To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

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

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

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

USER'S MANUAL

Phase-out/Discontinued

μPD78138

8-BIT SINGLE-CHIP MICROCOMPUTER

μPD78134A
μPD78136
μPD78138
μPD78P138

Document No. IEU−1324B
(O. D. No. IEU−740C)
Date Published July 1993 P
Printed in Japan

USER'S MANUAL

NEC

Phase-out/Discontinued

μPD78138

8-BIT SINGLE-CHIP MICROCOMPUTER

μPD78134A
μPD78136
μPD78138
μPD78P138

SUMMARY OF CONTENTS

CHAPTER 1 GENERAL 1-1

CHAPTER 2 PIN FUNCTIONS 2-1

CHAPTER 3 CPU FUNCTIONS 3-1

CHAPTER 4 CLOCK GENERATOR 4-1

CHAPTER 5 PORT FUNCTIONS 5-1

CHAPTER 6 REAL-TIME OUTPUT PORT (RTP) 6-1

CHAPTER 7 SERIAL INTERFACE 7-1

CHAPTER 8 SUPER TIMER UNIT 8-1

CHAPTER 9 A/D CONVERTER 9-1

CHAPTER 10 CLOCK OUTPUT (CLO) 10-1

CHAPTER 11 INTERRUPT FUNCTION 11-1

CHAPTER 12 STANDBY FUNCTION 12-1

CHAPTER 13 RESET FUNCTION 13-1

CHAPTER 14 EXAMPLE OF APPLICATION 14-1

CHAPTER 15 THE uPD78P138 15-1

CHAPTER 16 INSTRUCTION SET 16-1

CHAPTER 17 DIFFERENCES BETWEEN THE uPD78138 SERIES AND uPD78134 17-1

APPENDIXES A-1

Cautions on CMOS Devices

① Countermeasures against static electricity for all MOSs

Caution: When handling MOS devices, take care so that they are not electrostatically charged.
Strong static electricity may cause dielectric breakdown in gates. When transporting or storing MOS devices, use conductive trays, magazine cases, shock absorbers, or metal cases that NEC uses for packaging and shipping. Be sure to ground MOS devices during assembling. Do not allow MOS devices to stand on plastic plates or do not touch pins.
Also handle boards on which MOS devices are mounted in the same way.

② CMOS-specific handling of unused input pins

Caution: Hold CMOS devices at a fixed input level.
Unlike bipolar or NMOS devices, if a CMOS device is operated with no input, an intermediate-level input may be caused by noise. This allows current to flow in the CMOS device, resulting in a malfunction.
Use a pull-up or pull-down resistor to hold a fixed input level. Since unused pins may function as output pins at unexpected times, each unused pin should be separately connected to the V_{DD} or GND pin through a resistor.
If handling of unused pins is documented, follow the instructions in the document.

③ Statuses of all MOS devices at initialization

Caution: The initial status of a MOS device is unpredictable when power is turned on.
Since characteristics of a MOS device are determined by the amount of ions implanted in molecules, the initial status cannot be determined in the manufacture process. NEC has no responsibility for the output statuses of pins, input and output settings, and the contents of registers at power on. However, NEC assures operation after reset and items for mode setting if they are defined.
When you turn on a device having a reset function, be sure to reset the device first.

MAJOR REVISIONS

Page	Description
Throughout	The uPD78134A and uPD78138 have already been developed.
P.1-3	The quality grade for the EPROM versions of the uPD78P138 has been changed from standard to not applied.
Preface, PP.1-1, 1-7, 1-9, 15-1	Caution on the EPROM versions of the uPD78P138 has been added.
P.1-11	A product has been added to the list in Section 1.7.
 P.5-9 P.5-21 P.5-32 P.5-38	Caution on manipulating a port using an instruction such as a bit manipulation instruction has been added to: Section 5.3.2 Section 5.5.2 Section 5.7.2 Section 5.8.2
P.8-51, P.8-53	The delay between a CLR1 input signal and a signal output from the noise eliminator has been modified.
P.11-18 to P.11-20	Section 11.4.2 has been added.
P.16-61 to P.16-188	The examples of some instructions have been added to Section 16.6.
P.17-1 to P.17-3	Chapter 17 has been added.
P.A-1 to P.A-3	Appendix A has been modified.

PREFACE

Users: This manual is for engineers who intend to learn the capabilities of the uPD78134A, uPD78136, uPD78138, and uPD78P138 for application program development.

The EPROM versions of the uPD78P138 are not intended for use in mass-produced products; they do not have reliability high enough for such purposes. Their use should be restricted to functional evaluation in experiment or trial manufacture.

Purpose: The purpose of this manual is to help users understand the hardware capabilities of the uPD78134A, uPD78136, uPD78138, and uPD78P138.

Organization:

This manual includes the following items:

- General
- Pin functions
- CPU functions
- Peripheral functions
- Interrupt function
- Other built-in peripheral functions
- Standby function
- Reset function
- Applicable examples
- uPD78P138 (PROM product)
- Instruction functions

Guidance: Before using this manual, the user should have a general knowledge of the electronics, logical circuit, and microcomputer fields.

- To use this manual as a uPD78134A, uPD78136, or uPD78P138 manual:
  Where there is no functional difference, only the uPD78138 is described, and its descriptions are applicable to the uPD78134A, uPD78136, and uPD78P138.

- To use this manual as a uPD78P138 manual:
  In this manual, the description of the PROM is for both a one-time PROM and EPROM.
  To use this manual as a uPD78P138 manual, each reference to PROM should be understood as a one-time PROM or EPROM.

- To check the detailed functions of an instruction whose mnemonic is self-evident:
  Look it up in Appendix B, "Instruction Index" (in alphabetic order).

- To check an instruction whose mnemonic is not self-evident but whose functions are clear:
  Check the mnemonic of the instruction in Section 16.1 and check the function of the instruction in Section 16.6.

- To understand the general functions of the uPD78134A, uPD78136, uPD78138, and uPD78P138:
  Read the entire manual in the order of the table of contents.

- For the electrical characteristics of the uPD78134A, uPD78136, uPD78138, and uPD78P138: See the separate Data Sheet. □

- For application examples of the functions of the uPD78134A, uPD78136, uPD78138, and uPD78P138: See the separate Application Note. □

Notation: Data weight: Higher digits on the left side
Lower digits on the right side

Active low: $\overline{xxx}$ (Pins and signal names are overscored.)

Note: Explanation of the indicated part of the text

Caution: Information requesting the user's special attention

Remarks: Supplementary information

Numeric value: Binary: xxxxB or xxxx
Decimal: xxxx
Hexadecimal: xxxxH

Related publications:

Product Publication	uPD78134A	uPD78136	uPD78138	uPD78P138
Pamphlet	IF-260			
Data sheet	IC-8447			IC-8051
User's manual	This manual			
Mode register summary sheet	IEM-5529			
Instruction set	IEM-5530			
Instruction summary sheet	IEM-5531			
Application note VCR servo program	IEA-708			

CONTENTS

CHAPTER 1 GENERAL 1-1
1.1 Features 1-2
1.2 Applications 1-3
1.3 Ordering Information and Quality Grade Standard 1-3
1.4 Functional Overview 1-4
1.5 Block Diagram 1-6
1.6 Pin Configuration (Top View) 1-7
1.6.1 Normal operation mode (uPD78134A, uPD78136, uPD78138, and uPD78P138) 1-7
1.6.2 PROM programming mode (uPD78P138) 1-9
1.7 Functions 1-11

CHAPTER 2 PIN FUNCTIONS 2-1
2.1 Lists of Pin Functions 2-1
2.1.1 Normal operation mode 2-1
2.1.2 PROM programming mode (uPD78P138) 2-5
2.2 Pin Functions 2-6
2.2.1 Normal operation mode 2-6
2.2.2 PROM programming mode (uPD78P138) 2-19
2.3 Input/Output Circuits and Connection of Unused Pins 2-21

CHAPTER 3 CPU FUNCTIONS 3-1
3.1 Memory Space 3-1
3.1.1 Internal program memory space 3-6
3.1.2 Internal data memory space 3-7
3.1.3 Special function register (SFR) space 3-7
3.1.4 Data memory addressing 3-7
3.2 Registers 3-12
3.2.1 Program counter (PC) 3-12
3.2.2 Program status word (PSW) 3-12
3.2.3 Stack pointer (SP) 3-15
3.2.4 General registers 3-17

3.2.5 Special function registers (SFRs) 3-19
3.3 External Expansion Functions 3-27
3.3.1 Bus interface function 3-27
3.3.2 Memory mapping register (MM) 3-29
3.3.3 Memory map in external expansion 3-31

CHAPTER 4 CLOCK GENERATOR 4-1
4.1 Configuration and Functions 4-1
4.2 Cautions 4-3
4.2.1 When an external clock is applied 4-3
4.2.2 When crystal/ceramic oscillation is used 4-4

CHAPTER 5 PORT FUNCTIONS 5-1
5.1 Functions and Outline of the Ports 5-1
5.2 Port 0 (P00-P07) 5-4
5.2.1 Hardware configuration 5-4
5.2.2 Setting the I/O mode and/or control mode 5-5
5.2.3 Internal pull-up resistors 5-6
5.3 Port 1 (P10-P17) 5-7
5.3.1 Hardware configuration 5-8
5.3.2 Setting the I/O mode and/or control mode 5-9
5.3.3 Internal pull-up resistor 5-10
5.4 Port 2 (P20-P27) 5-12
5.4.1 Hardware configuration 5-13
5.4.2 Setting the I/O mode and/or control mode 5-13
5.4.3 Internal pull-up resistors 5-14
5.5 Port 3 (P30-P37) 5-16
5.5.1 Hardware configuration 5-17
5.5.2 Setting the I/O mode and/or control mode 5-21
5.5.3 Internal pull-up resistors 5-23
5.6 Port 4 (P40-P47) 5-26
5.6.1 Hardware configuration 5-27
5.6.2 Setting the I/O mode and/or control mode 5-27
5.6.3 Internal pull-up resistors 5-29
5.7 Port 5 (P50-P57) 5-31
5.7.1 Hardware configuration 5-32

5.7.2 Setting the I/O mode and/or control mode 5-32
5.7.3 Internal pull-up resistors 5-34
5.8 Port 6 (P60-P67) 5-36
5.8.1 Hardware configuration 5-37
5.8.2 Setting the I/O mode and/or control mode 5-38
5.8.3 Internal pull-up resistors 5-40
5.9 Port 7 (P70, P71) 5-42
5.9.1 Hardware configuration 5-42
5.9.2 Setting the I/O mode and/or control mode 5-43
5.9.3 Internal pull-up resistors 5-44

CHAPTER 6 REAL-TIME OUTPUT PORT (RTP) 6-1
6.1 Configuration and Functions of RTP 6-1
6.2 Hardware Configuration 6-2
6.3 Structure of the Buffer Register 6-2
6.4 RTP Control Register 6-4
6.5 RTP Operation 6-6

CHAPTER 7 SERIAL INTERFACE 7-1
7.1 Functions 7-1
7.2 Configuration 7-1
7.3 Control Registers 7-5
7.3.1 Serial interface mode register (CSIM) 7-5
7.3.2 Serial bus interface control register
(SBIC)................................ 7-6
7.4 Three-wire Serial I/O Mode 7-9
7.4.1 Basic operation timing 7-9
7.4.2 When only send operation is enabled 7-11
7.4.3 When only receive operation is enabled 7-13
7.4.4 When send and receive operations
are enabled 7-14
7.4.5 Action taken when shift operation is not
in phase with the serial clock 7-16
7.5 SBI Mode 7-17
7.5.1 SBI features 7-17
7.5.2 Serial interface configuration 7-19

7.5.3 Address match detection method 7-22
7.5.4 SBI mode control registers 7-22
7.5.5 Shift register (SIO) 7-27
7.6 Communication Operation and Signals 7-28
7.6.1 Bus release signal (REL) 7-29
7.6.2 Command signal (CMD) 7-29
7.6.3 Address 7-30
7.6.4 Command data 7-31
7.6.5 Acknowledge signal ($\overline{\text{ACK}}$) 7-32
7.6.6 Busy signal ($\overline{\text{BUSY}}$) and ready signal (READY) 7-33
7.6.7 Signals 7-33
7.6.8 Communiation operation 7-43
7.6.9 Clearing the busy signal 7-48
7.6.10 Wake-up setting operation 7-48
7.6.11 Starting send/receive operation 7-49
7.7 Cautions 7-49

CHAPTER 8 SUPER TIMER UNIT 8-1
8.1 Overview of the Super Timer Unit 8-2
8.1.1 Configuration of the super timer unit 8-2
8.1.2 Functional overview of the timer units 8-6
8.2 Timer 0 Unit 8-9
8.2.1 Configuration of the timer 0 unit 8-9
8.2.2 Event counter (EC) 8-11
8.2.3 Timer 0 (TM0) 8-17
8.2.4 Operating mode 8-18
8.2.5 Setting timer 0 unit control registers 8-22
8.2.6 Examples of using the timer 0 unit 8-29
8.3 Timer 1 Unit 8-34
8.3.1 Configuration of the timer 1 unit 8-34
8.3.2 Event divider 8-38
8.3.3 Pulse width detection circuit (TM3) 8-38
8.3.4 Configuration of timer 1 (TM1) 8-44
8.3.5 Output modes of timer 1 output pins 8-45
8.3.6 Operation of timer 1 8-46

8.3.7 Digital noise eliminator 8-50
8.3.8 Setting timer 1 unit control registers 8-66
8.4 Free Running Counter (FRC) Unit 8-74
8.4.1 Configuration of the free running counter unit 8-74
8.4.2 Capture register 2 (CPT2) 8-76
8.4.3 Operation of the FRC 8-79
8.4.4 Setting FRC unit control registers 8-80
8.4.5 Application of the FRC to a VCR 8-82
8.5 Timer 2 Unit 8-92
8.5.1 Configuration of the timer 2 unit 8-93
8.5.2 Setting the register to control the timer 2 unit 8-94
8.6 PWM Output Unit 8-95
8.6.1 Configuration of the PWM output unit 8-95
8.6.2 Operation of the PWM output unit 8-96
8.6.3 Setting the register to control PWM output unit 8-102
8.6.4 Registers other than the control register 8-104
8.6.5 Application of the PWM output units 8-104

CHAPTER 9 A/D CONVERTER 9-1
9.1 Functions of A/D Converter 9-1
9.2 Hardware Configuration of A/D Converter 9-2
9.3 Control Register for the A/D Converter 9-6
9.4 A/D Converter Operation 9-9
9.4.1 Basic operation of A/D converter 9-9
9.4.2 A/D converter operation mode 9-12
9.5 A/D Converter Interrupt Request 9-15

CHAPTER 10 CLOCK OUTPUT (CLO) 10-1
10.1 Configuration and Functions of CLO 10-1
10.2 CLO Control Register 10-4

CHAPTER 11 INTERRUPT FUNCTION 11-1
11.1 Interrupt Request Sources 11-2
11.1.1 Nonmaskable interrupt request 11-4
11.1.2 Maskable interrupt request 11-4
11.2 External Interrupt Request Functions 11-5
11.2.1 External interrupt control registers 11-5
11.3 Interrupt Processing Control Registers 11-9
11.4 Interrupt Processing 11-15
11.4.1 Multiple interrupt processing 11-16
11.4.2 When interrupt requests and macro services are temporarily held 11-18
11.5 Macro Service Functions 11-21
11.5.1 Overview of macro services 11-21
11.5.2 Macro service control register 11-22
11.5.3 Macro service modes and interrupt requests 11-27
11.5.4 Data transfer mode 11-27
11.5.5 Real-time output port control mode 11-34
11.5.6 Counter mode 11-58
11.5.7 Data pattern identification mode 11-59
11.5.8 Points to be noted 11-69

CHAPTER 12 STANDBY FUNCTION 12-1
12.1 Configuration of Standby Function Control Circuit 12-2
12.2 Setting the STOP Mode and Operation States in the STOP Mode 12-4
12.3 Releasing the STOP Mode 12-5
12.3.1 Releasing the STOP mode by NMI input 12-5
12.3.2 Releasing the STOP mode by $\overline{\text{RESET}}$ input 12-6

CHAPTER 13 RESET FUNCTION 13-1

CHAPTER 14 EXAMPLE OF APPLICATION 14-1
14.1 Example of Application to the Normal-type Video Cassette Recorder 14-1

14.2 Example of Application to the Camcoder 14-3
14.3 Using the Super Timer Unit in a VCR
Servo System 14-5
14.3.1 Controlling the drum motor 14-7
14.3.2 Controlling the capstan motor 14-12
14.4 Block Diagram of the Software Digital Servo
System in a VCR 14-17

CHAPTER 15 THE uPD78P138 15-1
15.1 Differences of uPD78P138 from uPD78134A,
uPD78136, and uPD78138 15-2
15.1.1 Internal memory size change register
(IMS) 15-3
15.2 Programming in the uPD78P138 15-5
15.2.1 Operation mode 15-5
15.2.2 Procedure for writing on PROM 15-6
15.2.3 Procedure for reading from PROM 15-9
15.3 Erasure Characteristics Only for
the uPD78P138K 15-10
15.4 Protective Film Covering the Erasure Window
Only for the uPD78P138K 15-10

CHAPTER 16 INSTRUCTION SET 16-1
16.1 Operations 16-1
16.1.1 Legend 16-1
16.1.2 List of operations 16-5
16.2 Instruction Codes 16-21
16.2.1 Legend 16-21
16.2.2 List of instruction codes 16-24
16.3 Number of Clocks of the Instructions 16-35
16.3.1 Legend 16-35
16.3.2 Numbers of clocks of the instructions 16-37
16.4 Instruction Addressing 16-49
16.4.1 Relative addressing 16-49
16.4.2 Immediate addressing 16-50
16.4.3 Table indirect addressing 16-51

16.4.4 Register addressing 16-51
16.5 Operand Addressing 16-52
16.5.1 Register addressing 16-52
16.5.2 Immediate addressing 16-53
16.5.3 Short direct addressing 16-54
16.5.4 Special function register (SFR) addressing 16-56
16.5.5 Register indirect addressing 16-57
16.5.6 Indexed addressing 16-59
16.6 Explanation of Instructions 16-61
16.6.1 8-bit data transfer instructions 16-61
16.6.2 16-bit data transfer instructions 16-73
16.6.3 8-bit arithmetic/logical instructions 16-76
16.6.4 16-bit arithmetic/logical instructions 16-117
16.6.5 Multiply/divide instructions 16-123
16.6.6 Increment/decrement instructions 16-125
16.6.7 Shift/rotate instructions 16-127
16.6.8 BCD correction instructions 16-135
16.6.9 Bit manipulation instructions 16-136
16.6.10 Call return instructions 16-158
16.6.11 Stack manipulation instructions 16-162
16.6.12 Unconditional branch instructions 16-164
16.6.13 Conditional branch instructions 16-166
16.6.14 CPU control instructions 16-187

CHAPTER 17 DIFFERENCES BETWEEN THE uPD78138 SERIES AND uPD78134 17-1

APPENDIX A DEVELOPMENT TOOLS A-1

APPENDIX B INDEX OF INSTRUCTIONS B-1

APPENDIX C INDEX OF REGISTERS C-1

APPENDIX D INDEX OF REGISTER ABBREVIATIONS D-1

LIST OF FIGURES

Figure	Title	Page
2-1	Pin Input/Output Circuits	2-23
3-1	Memory Map (uPD78134A)	3-3
3-2	Memory Map (uPD78136)	3-4
3-3	Memory Map (uPD78138 and uPD78P138)	3-5
3-4	Memory Map and Addressing of Data Memory	3-8
3-5	Format of Program Counter (PC)	3-12
3-6	Program Status Word (PSW) Format	3-13
3-7	Stack Pointer (SP) Format	3-16
3-8	Data Saved in Stack Memory	3-16
3-9	Data Restored from Stack Memory	3-16
3-10	Format of General Registers	3-18
3-11	Read Timing	3-28
3-12	Write Timing	3-29
3-13	Format of Memory Mapping Register (MM)	3-30
3-14	Memory Map in External Expansion (uPD78134A) ...	3-32
3-15	Memory Map in External Expansion (uPD78136)	3-33
3-16	Memory Map in External Expansion (uPD78138 and uPD78P138)	3-34
4-1	Block Diagram of the Clock Generator	4-1
4-2	External Circuitry of the Clock Oscillator	4-2
4-3	Signal Extraction Point in External Clock Input	4-3
4-4	Cautions for Resonator Connection Circuitry	4-5
4-5	Examples of Wrong Resonator Connection Circuitry	4-5
5-1	Port Configuration	5-1
5-2	Block Diagram of Port 0	5-4
5-3	Format of the Port 0 Mode Register	5-5

5-4 Format of Real-time Output Port Control Register (RTPC) 5-6

5-5 Block Diagram of Port 1 5-8

5-6 Format of the Port 1 Mode Register 5-9

5-7 Format of the Register for Optional Pull-up Resistors 5-10

5-8 Specification for Pulling Voltage in Port 1 High 5-11

5-9 Block Diagram of Port 2 5-13

5-10 Format of the Register for Optional Pull-up Resistors 5-14

5-11 Specification for Pulling Voltage in Port 2 High 5-15

5-12 Block Diagram of Port 3 5-17

5-13 Format of the Port 3 Mode Register 5-21

5-14 Format of the Port 3 Mode Control Register (PMC3) 5-22

5-15 Format of the Register for Optional Pull-up Resistors 5-24

5-16 Specification for Pulling Voltage in Port 3 High 5-25

5-17 Block Diagram of Port 4 5-27

5-18 Port 4 Operation Mode Set by $\overline{\text{EA}}$ and MM 5-28

5-19 Format of the Register for Optional Pull-up Resistors 5-29

5-20 Specification for Pulling Voltage in Port 4 High 5-30

5-21 Block Diagram of Port 5 5-32

5-22 Port 5 Operation Mode Set by $\overline{\text{EA}}$ and MM 5-33

5-23 Format of the Port 5 Mode Register (PM5) 5-35

5-24 Format of the Register for Optional Pull-up Resistors 5-35

5-25 Specification for Pulling Voltage in Port 5 High 5-35

5-26 Block Diagram of Port 6 5-37

5-27 P64 and P65 Operation Modes Set by $\overline{\text{EA}}$ and MM 5-39

5-28 Format of the Port 6 Mode Register 5-39

5-29 Format of the Register for Optional Pull-up Resistors 5-40

5-30 Specification for Pulling Voltage in Port 6 High 5-41

5-31 Block Diagram of Port 7 5-42

5-32 Format of Port 7 Mode Register (PM7) 5-43

5-33 Format of the Register for Optional Pull-up Resistors 5-44

5-34 Specification for Pulling Voltage in Port 7 High 5-45

6-1 Block Diagram of RTP 6-2

6-2 Structure of Buffer Registers P0H and P0L 6-3

6-3 Format of Real-time Output Port Control Register (RTPC) 6-5

6-4 Timing of RTP Operation 6-8

7-1 Block Diagram of Serial Interface 7-2

7-2 Format of Serial Interface Mode Register (CSIM) 7-5

7-3 Format of Serial Bus Interface Control Register (SBIC) 7-8

7-4 Example of 3-Wire Serial I/O System Configuration 7-9

7-5 Timing of 3-Wire Serial I/O Mode 7-10

7-6 Example of Connection with 2-Wire Serial I/O Device 7-11

7-7 Example of SBI-Based Serial Bus Configuration 7-19

7-8 Block Diagram of Serial Interface 7-20

7-9 Pin Configuration 7-21

7-10 Format of Serial Interface Mode Register (CSIM) 7-23

7-11 Format of SBIC Register 7-24

7-12 Configuration of Shift Register and Peripheral Circuitry 7-27

7-13 Timing of SBI Transfer 7-28

7-14 Bus Release Signal 7-29

7-15 Command Signal 7-30

7-16 Address 7-30

7-17 Slave Selection Using an Address 7-31

7-18 Command 7-31

7-19 Data 7-31

7-20 Acknowledge Signal 7-32

7-21 Busy Signal and Ready Signal 7-33

7-22 Operation of RELT, CMDT, RELD, and CMDD 7-34

7-23 Operation of ACKT 7-34

7-24 Operation of ACKE 7-35

7-25 Operation of ACKD 7-36

7-26 Operation of BSYE 7-37

7-27 Address Transfer Operation from Master Device to Slave Device 7-44

7-28 Command Transfer Operation from Master Device to Slave Device 7-45

7-29 Data Transfer Operation from Master Device to Slave Device 7-46

7-30 Data Transfer Operation from Slave Device to Master Device 7-47

8-1 Configuration of the Super Timer Unit 8-4

8-2 Configuration of Timer 0, Timer 1, and Free Running Counter 8-5

8-3 Configuration of the Timer 0 Unit 8-10

8-4 Configuration of the Event Counter in the Internal Pulse Generation Mode 8-11

8-5 Operation Timing in the Internal Pulse Generation Mode (Generation of VCR Head Switching Signal) 8-13

8-6 Configuration of the Event Counter in the General Event Divider Mode 8-14

8-7 Operation Timing in the General Event Divider Mode 8-16
8-8 Configuration of the Timer 0 General Output Mode 8-18
8-9 Configuration of the Timer 0 RS Output Mode 8-19
8-10 Configuration of Timer 0 Delay Pulse Output Mode 1 8-19
8-11 Configuration of Timer 0 Delay Pulse Output Mode 2 8-20
8-12 Format of Input Control Register (ICR) 8-23
8-13 Format of Timer Control Register 0 (TMC0) 8-24
8-14 Format of Timer 0 Output Mode Register (TOM0) 8-25
8-15 Format of Timer 0 Output Control Register (TOC0) 8-27
8-16 Operation Timing of Timer 0 Delay Pulse Output Modes 1 and 2 8-30
8-17 Generating Head Switching Signals by Using the Timer 0 Unit 8-32
8-18 Timing Example of Head Switching Signals with the Timer 0 Unit 8-33
8-19 Configuration of the Timer 1 Unit 8-36
8-20 Configuration of the Timer 1 Clear Input Section 8-37
8-21 Configuration of the Pulse Width Detection Circuit (TM3) 8-41
8-22 Timing of Pulse Width Detection Circuit Operation 8-42
8-23 Configuration of the Prescaler 8-43
8-24 Configuration of the Timer 1 (TM1) 8-44
8-25 Output Modes of PTO11 8-45
8-26 Reference Counter Operation of Timer 1 8-47
8-27 Programmable Timer Output Operation of Timer 1 8-49
8-28 Timer 1 Phase Difference Detection Operation ... 8-50

8-29 Configuration of the CLR1 Input Section (Digital Noise Eliminator) 8-53
8-30 Operation of the Digital Noise Eliminator 8-54
8-31 Vertical Synchronizing Signal Extraction Using the Digital Noise Eliminator 8-56
8-32 COMPSYNC Signal Used for Even/Odd Field Determination 8-58
8-33 Example of INTCLR1 Setting 8-59
8-34 Even/Odd Field Determination Using the Digital Noise Eliminator 8-64
8-35 Example of CLR1 Input Detection Error Due to Noise 8-65
8-36 Format of the Event Divider Control Register (EDVC) 8-67
8-37 Format of External Capture Input Mode Register 1 (INTM1) 8-68
8-38 Format of Prescaler Mode Register (PRM3) 8-69
8-39 Format of Capture Mode Register (CPTM) 8-70
8-40 Format of Timer Control Register 0 (TMC0) 8-71
8-41 Format of Timer 1 Output Mode Register (TOM1) 8-72
8-42 Format of Timer 1 Output Control Register (TOC1) 8-72
8-43 Format of Input Control Register (ICR) 8-73
8-44 Configuration of the Free Running Counter 8-75
8-45 Configuration of CPT2 (CPT2H, CPT2L) 8-77
8-46 CPT2L Capture Data 8-77
8-47 Example of FRC Count Operation 8-78
8-48 Format of Timer Control Register 1 (TMC1) 8-80
8-49 Format of Capture Mode Register (CPTM) 8-81
8-50 Example of Using the FRC in a VCR (Detecting Drum Motor Speed Error Using Capture Register 2) 8-82
8-51 Example of Using FRC Capture Registers 0 (CPT0)... 8-83

8-52 Example of Using the FRC and TM1 in Playback (Operating TM1 as the Internal Phase Reference Timer) 8-84
8-53 Operation of the FRC and TM1 in Playback 8-85
8-54 Example of Using the FRC and TM1 in Recording (Capturing the FRC Contents on Input of the Phase Reference Signal) 8-87
8-55 Operation of the FRC and TM1 in Recording 8-88
8-56 FRC Capture Operation in Recording (CPT0) 8-91
8-57 Configuration of the Timer 2 Unit 8-93
8-58 Format of Timer Control Register 1 (TMC1) 8-94
8-59 Configuration of the PWM Output Unit 8-95
8-60 Active Level Setting for PWM Output 8-98
8-61 Example 1 of PWM Output Timing (PWM Pulse Width Switching Cycle: $2^{12}/f_{CLK}$) ... 8-100
8-62 Example 2 of PWM Output Timing (PWM Pulse Width Switching Cycle: $2^{8}/f_{CLK}$) 8-101
8-63 Format of the PWM Control Register (PWMC) 8-103

9-1 Block Diagram of A/D Converter 9-3
9-2 Example of Connecting Capacitors to A/D Converter Pins 9-4
9-3 Format of A/D Conversion Mode Register (ADM) ... 9-8
9-4 Basic Operation of A/D Conversion 9-11
9-5 A/D Conversion Started by Hardware 9-11
9-6 Rewriting ADM Contents during A/D Conversion ... 9-12
9-7 Operation Timing in Select Mode 9-13
9-8 Operation Timing in Scan Mode 9-14

10-1 Application Example for the Clock Output Function 10-1
10-2 Block Diagram of the Clock Output Circuit 10-2
10-3 CLO Pin at Initialization 10-3
10-4 Format of Clock Output Mode Register (CLOM) 10-5
10-5 Example of Setting the CLOM Register 10-6

11-1 Noise Elimination at an External Interrupt Request Pin 11-5

11-2 Format of the External Interrupt Mode Register 0 (INTM0) 11-7

11-3 Format of the External Capture Input Mode Register (INTM1) 11-8

11-4 Format of the Interrupt Request Flag Register (IF0) 11-12

11-5 Format of the Interrupt Mask Register (MK0) 11-13

11-6 Format of the Interrupt Service Mode Register (ISM0) 11-13

11-7 Format of the Priority Specification Flag Register (PR0) 11-14

11-8 Interrupt Processing Algorithm 11-15

11-9 Sequence of Macro Service Processing 11-22

11-10 Macro Service Control Words 11-24

11-11 Format of the Macro Service Mode Register (High-order Three Bits) 11-25

11-12 Format of a Macro Service Mode Register 11-26

11-13 Setting of a Macro Service Mode Register in the Data Transfer Mode 11-28

11-14 Addressing in the Data Transfer Mode 11-30

11-15 Processing Sequence in the Data Transfer Mode 11-31

11-16 Example of Data Transfer 11-33

11-17 Addressing in the Real-time Output Port Control Mode 11-35

11-18 Processing Sequence in the Real-time Output Port Control Mode 11-39

11-19 Example of Setting the Macro Service Mode Register in the Basic Operation Mode 11-41

11-20 Open-loop Control for the Stepping Motor via the Realtime Output Port 11-42

11-21 Output Timing in the Real-time Output Port Mode 11-43

11-22 Data Transfer Control Timing 11-44

11-23 1-phase Excitation of a 4-phase Stepping Motor 11-46
11-24 1-2 Phase Excitation of a 4-phase Stepping Motor 11-47
11-25 Example of Setting the Macro Service Mode Register for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion) 11-49
11-26 Block Diagram for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion) 11-50
11-27 Timing Diagram 1 for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion) 11-51
11-28 Example of Setting the Macro Service Mode Register for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals) 11-52
11-29 Block Diagram for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals) 11-53
11-30 Timing Diagram 2 for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals) 11-54
11-31 Example of Macro Service Operation 11-55
11-32 Example of a Counter Mode Operation 11-58
11-33 Addressing in the Data Pattern Identification Mode 11-60
11-34 Note on Setting a Data Comparing Area 11-61
11-35 Example of Setting the Macro Service Mode Register in the Data Pattern Identification Mode (with Comparison) 11-63
11-36 Example of an Application in the Data Pattern Identification Mode (VCR Index Search Control) 11-

12-1 Standby Status Transition 12-1

12-2 Configuration of a Standby Function
Control Circuit 12-2

12-3 Format of the Standby Control Register
(STBC) 12-3

12-4 Releasing the STOP Mode by NMI Input 12-5

13-1 Accepting a Reset Signal 13-1

13-2 Reset at Power-on 13-2

14-1 Example of Application to the Normal-type
Video Cassette Recorder 14-2

14-2 Example of Application to the Camcoder 14-4

14-3 Using the Super Timer Unit in the VCR
Servo System 14-6

14-4 Controlling the Drum Motor 14-9

14-5 Drum Speed Control 14-10

14-6 Drum Phase Control 14-11

14-7 Controlling the Capstan Motor 14-14

14-8 Capstan Speed Control 14-15

14-9 Capstan Phase Control 14-16

14-10 Block Diagram of the VCR Servo System 14-18

15-1 Format of the Internal Memory Size Change
Register (IMS) 15-4

15-2 PROM Write/Verify Timing Chart 15-7

15-3 Flowchart of Writing Procedure 15-8

15-4 PROM Read Timing Chart 15-9

LIST OF TABLES

Table	Title	Page
1-1	Main Differences between the Products	1-1
1-2	Functions	1-11
2-1	Port 2 Dual-function Pins	2-7
2-2	Port 3 Modes	2-12
2-3	Port 6 Operation Modes	2-15
2-4	I/O Circuit Type of Each Pin and Connection of Unused Pins	2-21
3-1	Memory Areas of Each Product	3-1
3-2	Vector Table	3-6
3-3	ISP Flag Format	3-13
3-4	Specification of a Register Bank	3-14
3-5	Correspondence between Function Names and Absolute Names	3-18
3-6	Special Function Registers (SFRs)	3-21
3-7	External Expansion Modes and Pin Functions	3-28
5-1	Port Functions	5-2
5-2	Function of Port 0	5-4
5-3	Functions of Port 1	5-7
5-4	Functions of Port 2	5-12
5-5	Functions of Port 3	5-16
5-6	Functions of Port 4	5-26
5-7	Functions of Port 5	5-31
5-8	Functions of Port 6	5-36
5-9	Functions of Port 7	5-42
6-1	Output Trigger for Real-time Output Port (When RTPC POMH = POML = 1)	6-6

7-1 Read/Write Operation of SBIC Register 7-7

7-2 Various Signals Used in SBI Mode 7-38

7-3 Conditions for Clearing $\overline{\text{BUSY}}$ 7-48

8-1 Components of the Super Timer Unit 8-2

8-2 Resolution and Maximum Count Time of Each Timer (at 12 MHz) 8-8

8-3 Resolution and Maximum Count Time of Timer 0 ... 8-17

8-4 Timer 0 Output Pins and Output Modes that can be Set 8-21

8-5 TOMO Setting Values and Output Modes of Timer 0 Outputs 8-21

8-6 Resolution of Timer 1 (at 12 MHz) 8-45

8-7 Digital Noise Eliminator Specification 8-51

8-8 FRC Capture Registers and Capture Triggers 8-74

8-9 Resolution of the FRC (at 12 MHz) 8-79

8-10 Resolution of Timer 2 (at 12 MHz) 8-92

9-1 Mode Generating INTAD 9-2

9-2 Conditions for Generating Interrupt Requests in Different A/D Converter Operating States 9-15

11-1 Interrupt Request Processing Modes 11-1

11-2 Interrupt Request Sources 11-2

11-3 Valid Edges and Control Registers of External Interrupt Pins 11-6

11-4 Flags Corresponding to Each Interrupt Request Source 11-9

11-5 Multiple Interrupt Processing 11-17

11-6 Macro Service Modes and Interrupt Request Sources 11-27

11-7 Interrupt Requests Sources in the Data Transfer Mode and SFR 11-28

11-8 Functions of the Macro Service Pointers 11-34

11-9 Macro Service Processing Time 11-68

11-10 Address Ranges that cannot be Used by the Macro Service Function 11-69

12-1 Operation States in the STOP Mode 12-4

13-1 Hardware Statuses after Reset 13-2

14-1 Timer Functions in the VCR Servo System 14-5

15-1 Differences of uPD78P138 from uPD78134A, uPD78136, and uPD78138 15-2

15-2 Operation Mode when Programming on the PROM 15-5

CHAPTER 1 GENERAL

The uPD78138 is an 8-bit single-chip microcomputer. It contains a high-speed, high-performance 8-bit CPU.

With on-chip peripheral hardware, the uPD78138 can be used in VCRs and other devices that require digital servo control via the software.

By mass-storage built-in ROM, the uPD78138 can support system control on one chip in addition to servo control, thus further miniaturizing the application set.

The uPD78P138 with the PROM is also provided, which is suited for evaluation and trial manufacture during system development, early stage start-up of applications, and short-run and multiple-device production.

> The EPROM versions of the uPD78P138 are not intended for use in mass-produced products; they do not have reliability high enough for such purposes. Their use should be restricted to functional evaluation in experiment or trial manufacture.

Table 1-1 Main Differences between the Products

Product	ROM	RAM	ROM type
uPD78134A	16K bytes	384 bytes	Mask ROM
uPD78136	24K bytes	640 bytes	
uPD78138	32K bytes		
uPD78P138			One-time PROM/EPROM

1.1 Features

- High-speed instruction execution via internal multiplexed bus: 0.33 us (at 12 MHz)

- Built-in super timer unit that best suits VCR servo control

  - Speed and phase control of drums, capstans, and motors
  - Head switch signal output of two channels including audio and video
  - Vertical synchronizing signal detection function
  - Input pulse duty ratio determining function
  - Built-in two-channel PWM output circuit that can specify active levels

- Additional functions that improve responsibility of servo control

  - Signed multiply instruction
  - Variable PWM output carrier frequency (23.4/46.9 kHz)

- Built-in real-time output port that can vary output patterns at any interval. (Suited for outputting the VCR head switch and controlling a step motor.)

- Powerful interrupt functions providing two service modes

  - Vector interrupt function
  - Macro service function (Facilitates automatic data transfer and AMSS function on VCRs.)

- Built-in pull-up resistor eliminating the need for external resistors

- 80-pin plastic QFP (14 mm x 20 mm)

1.2 Applications

The uPD78138 applies to servo controlling of VCRs (normal type and camcorder type), HDDs, FDDs, DATs, and CDPs.

VCR:	Video Cassette Recorder
HDD:	Hard Disc Drive
FDD:	Floppy Disk Drive
DAT:	Digital Audio Tape Recorder
CDP:	Compact Disc Player

1.3 Ordering Information and Quality Grade Standard

(1) Ordering information

Part number	Package	On-chip ROM
uPD78134AGF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
uPD78136GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
uPD78138GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
uPD78P138GF-3B9	80-pin plastic QFP (14 x 20 mm)	One-time PROM
uPD78P138K	80-pin LCC	EPROM

Remark: xxx is a ROM code.

(2) Quality grade

Part number	Package	On-chip ROM
uPD78134AGF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Standard
uPD78136GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Standard
uPD78138GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Standard
uPD78P138GF-3B9	80-pin plastic QFP (14 x 20 mm)	Standard
uPD78P138K	80-pin LCC	Not applied

Please refer to "Quality Grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

1.4 Functional Overview

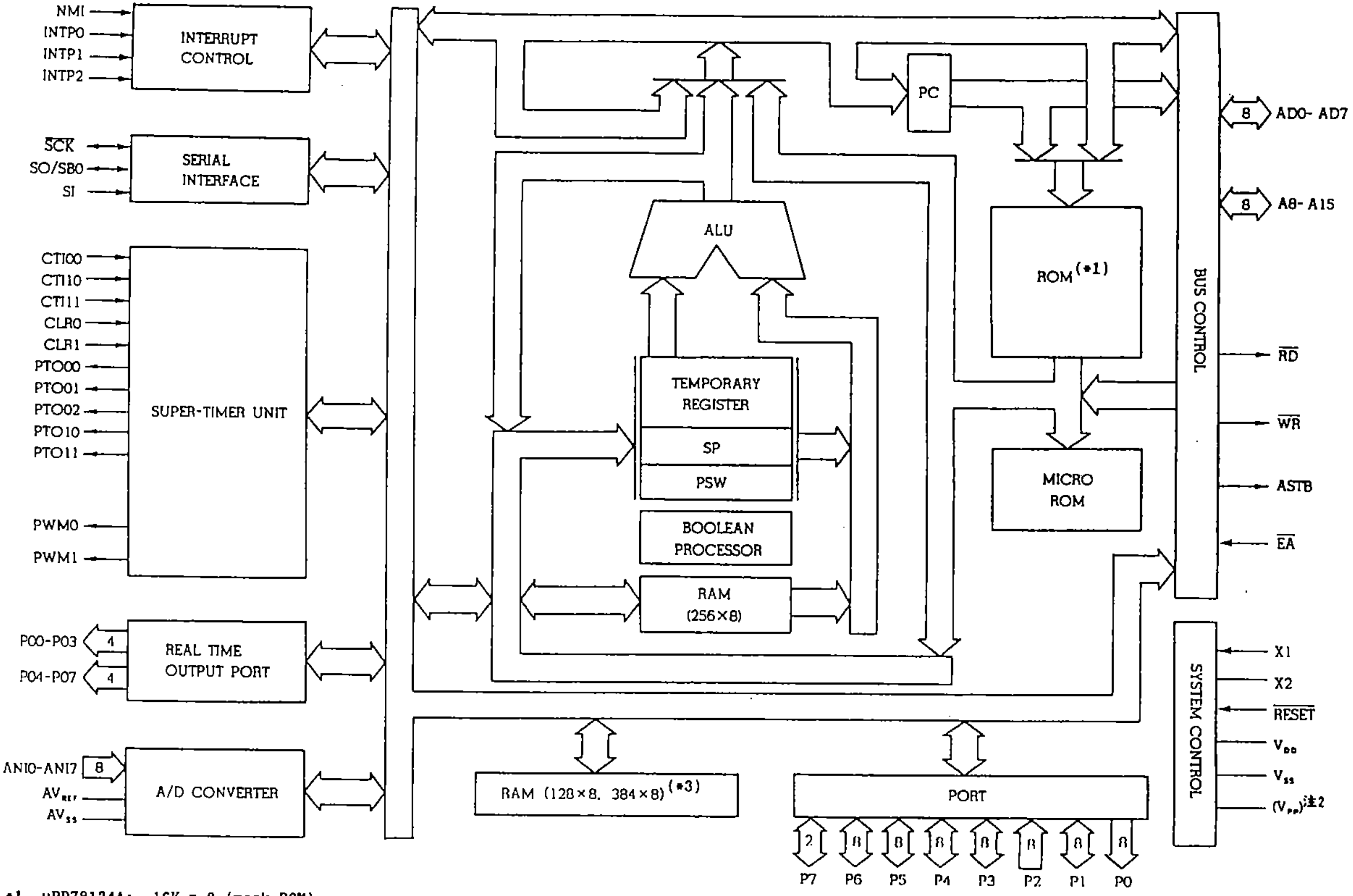
Item	Function
Number of basic instructions	64
Minimum instruction execution time	0.33 us (at 12 MHz)
Internal memory	. Program memory: 16256 x 8 bits (mask ROM, uPD78134A) 24576 x 8 bits (mask ROM, uPD78136) 32768 x 8 bits (mask ROM, uPD78138) 32768 x 8 bits (PROM, uPD78P138) . Data memory: 384 x 8 bits (uPD78134A), 640 x 8 bits
Memory expansion	Externally expandable up to 64K bytes
General register	8 bits x 8 x 4 banks (memory mapping)
Instruction set	. 16-bit addition, subtraction, comparison . Signed multiply instruction (signed 16 bits x unsigned 8 bits) . Unsigned multiply/divide instruction (16 bits x 8 bits, 16 bits ÷ 8 bits) . Bit manipulation instruction (transfer, Boolean operation, set, reset, test) . BCD correction instruction
I/O line	. 66 total Input port: 10 Output port: 12 I/O port: 36 Analog input: 8
Super timer unit	. Timer: 16 bits x 3 7 bits x 1 . Counter: 18 bits x 1 . Capture register: 18 bits x 1 16 bits x 4 7 bits x 1 . Compare register: 16 bits x 6 7 bits x 1 . PWM output: 12 bits x 2 (variable active level, variable carrier frequency (23.4/46.9 kHz))

(to be continued)

(Cont'd)

Item	Function
Real-time output port	. Timer-connected port output function . 4 bits x 2 or 8 bits x 1
Serial interface	. Either NEC format serial bus interface (SBI) or 3-wire serial interface can be selected.
A/D converter	. 8-bit resolution x 8 inputs . Conversion time: 30 us/1 analog input (at 12 MHz)
Interrupt	. Interrupt source: 17 (5 external and 12 internal) . One of the two service modes can be selected (macro service/vector interrupt). . Variable 2-level interrupt priority
Standby	STOP mode
Pull-up resistor	44, built-in (Enable/disable built-in can be specified via software.)

1.5 Block Diagram



*1 uPD78134A: 16K x 8 (mask ROM)
uPD78136: 24K x 8 (mask ROM), uPD78138: 32K x 8 (mask ROM),
uPD78P138: 32K x 8 (PROM)

*2 V_{PP} is a positive power supply pin for PROM and is only available on the uPD78P138.

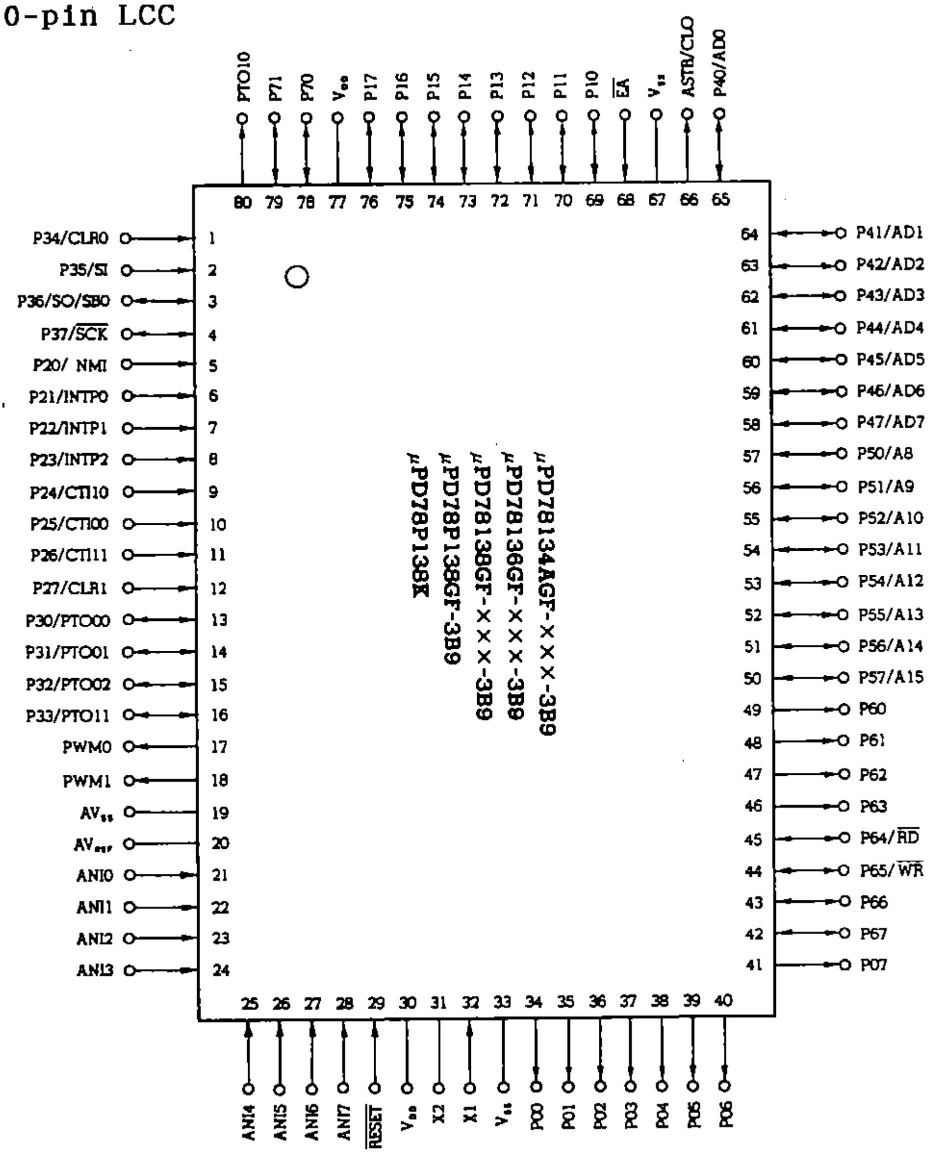
*3 uPD78134A: 128 x 8 bits, uPD78136, uPD78138, and uPD78P138: 384 x 8 bits

1.6 Pin Configuration (Top View)

1.6.1 Normal operation mode (uPD78134A, uPD78136, uPD78138, and uPD78P138)

80-pin plastic QFP (14 x 20 mm)

80-pin LCC



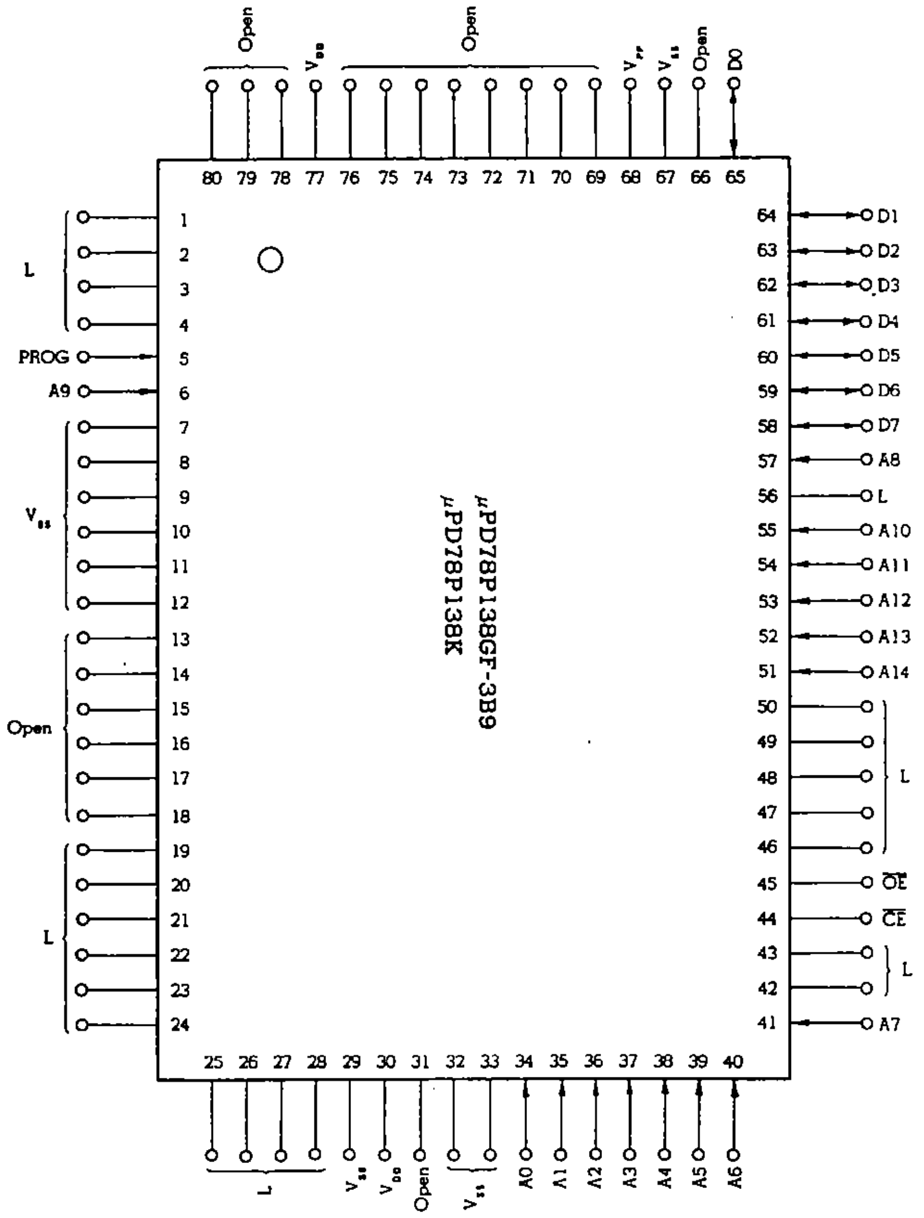
The EPROM versions of the uPD78P138 are not intended for use in mass-produced products; they do not have reliability high enough for such purposes. Their use should be restricted to functional evaluation in experiment or trial manufacture.

P00-P07:	Port0
P10-P17:	Port1
P20-P27:	Port2
P30-P37:	Port3
P40-P47:	Port4
P50-P57:	Port5
P60-P67:	Port6
P70, P71:	Port7
PWM0, PWM1:	Pulse Width Modulation Output
CLO:	Clock Output
ANI0-ANI7:	Analog Input
AV$_{REF}$:	Reference Voltage
AV$_{SS}$:	Analog V$_{SS}$
X1, X2:	Crystal
$\overline{\text{RESET}}$:	Reset

CTI00, CTI10,: CTI11	Capture Trigger Input
CLR0, CLR1:	Timer Clear Input
PTO00-PTO02,: PTO10, PTO11	Programmable Timer Input
NMI:	Nonmaskable Interrupt
INTP0-INTP2:	Interrupt From Peripherals
SI:	Serial Input
SO:	Serial Output
SB0:	Serial Bus
$\overline{\text{SCK}}$:	Serial Clock
AD0-AD7:	Address Data
A8-A15:	Address
$\overline{\text{RD}}$:	Read
$\overline{\text{WR}}$:	Write
ASTB:	Address Strobe
$\overline{\text{EA}}$:	External Access

1.6.2 PROM programming mode (uPD78P138)

80-pin plastic QFP (14 x 20 mm)
80-pin LCC



The EPROM versions of the uPD78P138 are not intended for use in mass-produced products; they do not have reliability high enough for such purposes. Their use should be restricted to functional evaluation in experiment or trial manufacture.

A0-A14:	Address	$\overline{CE}$:	Chip Enable
D0-D7:	Data	$\overline{OE}$:	Output Enable
PROG:	Program	V_{PP}:	Program Voltage

Cautions 1. L: Individually connect to a pull-down resistor and fix to low level.

2. V_{SS}: Connect to ground.

3. Open: Must be left open.

1.7 Functions

Table 1-2 Functions

Product Item	uPD78134A	uPD78136	uPD78138	uPD78P138	uPD78134
Minimum instruction execution time	. 0.33 us (at 12 MHz)				
ROM size	16K bytes (Mask ROM)	24K bytes (Mask ROM)	32K bytes (Mask ROM)	32K bytes (PROM)	16K bytes (Mask ROM)
RAM size	384 bytes	640 bytes			384 bytes
I/O ports	. 58 total { Input: 10 Output: 12 I/O: 36 }				
	. A/D: 8				
Real-time output ports	. 8 (Connected to a timer outputting a trigger signal)				
Super timer unit	. Timers: 16 bits x 3 7 bits x 1				
	. Counter: 18-bit free running counter x 1				
	. Capture registers: 18 bits x 1 16 bits x 4 7 bits x 1				
	. Compare registers: 16 bits x 6 7 bits x 1				
	. PWM outputs: 12 bits x 2 channels (carrier frequency: 23.4 kHz/ 46.9 kHz selectable)				. PWM outputs: 12 bits x 2 channels
Multiply instructions	. MULUW: 16-bit absolute value x 8-bit absolute value . MULSW: 16-bit complement x 8-bit absolute value				. MULUW: 16-bit absolute value x 8-bit absolute value

(to be continued)

Table 1-2 Functions (Cont'd)

Product / Item	uPD78134A	uPD78136	uPD78138	uPD78P138	uPD78134
A/D converter	. 8-bit resolution x 8 channels (conversion time: 30 us)				
Serial interface	. Three-wire SIO or SBI mode selectable: 1 channel				
Interrupt function	. Internal: 5, external: 12				
Evaluation chip	. uPD78P138 (with one-time PROM and window)				
Package	. 80-pin plastic QFP (14 x 20 mm excluding the dimensions of the pins) . 80-pin LCC				
V_{SYNC} separation circuit	. Threshold pulse width for elimination selectable (5.3 us/12.0 us)				. Threshold pulse width for elimination fixed (6.7 us)

CHAPTER 2 PIN FUNCTIONS

2.1 Lists of Pin Functions

2.1.1 Normal operation mode

(1) Port pins

Pin name	I/O	Dual-function pin	Function
P00-P07	O	—	Port 0 (P0): Can be specified to output or high impedance 8 bits by 8 bits. Also function as an 8 bits x 1 or 4 bits x 2 real-time output port.
P10-P17	I/O	—	Port 1 (P1): Can be specified to input or output bit by bit. Can directly drive LED. Software pull-up resistor (P10-P17) can be built in.
P20	I	NMI	Port 2 (P2): Software pull-up resistor (P22-P27) can be built in.
P21		INTP0	
P22		INTP1	
P23		INTP2	
P24		CTI10	
P25		CTI00	
P26		CTI11	
P27		CLR1	
P30	I/O	PTO00	Port 3 (P3): P30-P33: I/O port (Can be specified to input or output bit by bit.) P34, P35: Input port P36, P37: I/O port Software pull-up resistor (P30-P37) can be built in.
P31		PTO01	
P32		PTO02	
P33		PTO11	

(to be continued)

(Cont'd)

Pin name	I/O	Dual-function pin	Function
P34	I	CLR0	Port 3 (P3): P30-P33: I/O port (Can be specified to input or output bit by bit.) P34, P35: Input port P36, P37: I/O port Software pull-up resistor (P30-P37) can be built in.
P35	I	SI	
P36	I/O	SO/SB0	
P37	I/O	$\overline{\text{SCK}}$	
P40-P47	I/O	AD0-AD7	Port 4 (P4): Can be specified to input or output 8 bits by 8 bits. Software pull-up resistor (P40-P47) can be built in.
P50-P57	I/O	A8-A15	Port 5 (P5): Can be specified to input or output bit by bit. Software pull-up resistor (P50-P57) can be built in.
P60-P63	O	—	Port 6 (P6): P60-P63: Output port P64-P67: I/O port (Can be specified to input or output bit by bit.) Software pull-up resistor (P64-P67) can be built in.
P64	I/O	$\overline{\text{RD}}$	
P65	I/O	$\overline{\text{WR}}$	
P66, P67	I/O	—	
P70, P71	I/O	—	Port 7 (P7): Can be specified to input or output 2 bits by 2 bits. Software pull-up resistor (P70, P71) can be built in.

(2) Non-port pins

Pin name	I/O	Dual-function pin	Function
PWM0, PWM1	O	—	Super timer unit PWM output
ANI0-ANI7	I	—	Analog voltage input to A/D converter
AV_{REF}		—	Reference voltage input to A/D converter
AV_{SS}		—	Ground potential of A/D converter
NMI	I	P20	Non-maskable interrupt request input Either rising edge or falling edge can be selected via mode register (INTM0).
INTP0	I	P21	External interrupt request input Rising edge, falling edge, or rising and falling edges can be selected via mode register (INTM0).
INTP1	I	P22	External interrupt request input Can select rising edge, falling edge, or rising and falling edges by mode register (INTM0).
INTP2		P23	
SI	I	P35	Serial data input (3-wire serial I/O mode)
SO	I/O	P36/SB0	Serial data output (3-wire serial I/O mode)
SB0	I/O	P36/SO	Serial data input (SBI mode)
$\overline{SCK}$	I/O	P37	Serial clock input/output
CTI00	I	P25	Super timer unit capture trigger input
CTI10		P24	
CTI11		P26	
CLR0	I	P34	Super timer unit timer clear signal input
CLR1		P27	
PTO00	I/O	P30	Super timer unit timer output
PTO01		P31	
PTO02		P32	

(to be continued)

(Cont'd)

Pin name	I/O	Dual-function pin	Function
PTO10	O	—	Super timer unit timer output
PTO11	I/O	P33	
AD0-AD7	I/O	P40-P47	Time multiplexing address/data bus for when external memory is connected
A8-A15	O	P50-P57	Address output port for when external memory is connected
$\overline{RD}$	O	P64	Strobe signal output for reading external memory
$\overline{WR}$	O	P65	Strobe signal output for writing external memory
ASTB	O	CLO	Timing signal output that externally latches address data for accessing external memory
CLO	O	ASTB	Clock output
$\overline{EA}$	I	—	External expansion function control input
X1	I	—	Crystal connection for system clock signal oscillation Input the externally supplied clock signal to X1 and input its inverted phase to X2.
X2	—	—	
RESET	I	—	System reset input Contains an analog delay noise reduction circuit.
V_{DD}		—	Positive power supply
V_{SS}		—	GND potential

2.1.2 PROM programming mode (uPD78P138)

Pin name	I/O	Function
PROG	I	PROM programming mode setting
$\overline{\text{RESET}}$		
A0-A14		Address bus
D0-D7	I/O	Data bus
$\overline{\text{CE}}$	I	PROM enable input
$\overline{\text{OE}}$		Read strobe to PROM
V_{PP}	—	Write power supply
V_{DD}		Positive power supply
V_{SS}		GND

2.2 Pin Functions

2.2.1 Normal operation mode

(1) P00-P07 (Port 0): 3-state output

Eight-bit output dedicated pins of port 0 (8-bit output dedicated port with an output latch). Use the port 0 mode register (PM0) to specify the output mode or high impedance status eight bits by eight bits for these pins.

P00-P07 pins function as an 8-bit real-time output port, which output the contents of the buffer registers (P0L, P0H) at any inverval. P00-P07 pins can be divided into two groups (P00-P03 and P04-P07), either group functions as a 4-bit real-time output port. Use the real-time output port control register (RTPC) to specify the function as a normal output port or real-time output port.

When $\overline{\text{RESET}}$ is input, these pins are set to output high impedance status and the contents of the output latch becomes indefinite.

(2) P10-P17 (Port 1): Input/output

Eight-bit input/output pins of port 1 (8-bit I/O port with an output latch). Use the port 1 mode register (PM1) to specify input or output bit by bit to these pins.

These pins can handle a large current to directly drive an LED chip.

When $\overline{\text{RESET}}$ is input, port 1 is set to the all-bit input mode (output high impedance) and the contents of the output latch becomes indefinite.

Port 1 contains a software pull-up resistor. Use bit 1 of the pull-up resistor option register (PUO) to specify built-in pull-up resistor.

(3) P20-P27 (Port 2): Input

Eight-bit input pins of port 2 (8-bit input port).

These pins also function as various control pins. These pins can always read pin levels independent of the other function.

These eight pins employ Schmitt trigger input to eliminate operational mistakes caused by noise.

Table 2-1 lists the dual-function pins of port 2.

Table 2-1 Port 2 Dual-function Pins

Port 2	Dual-function pin	Port 2	Dual-function pin
P20	NMI input	P24	CTI10 input
P21	INTP0 input	P25	CTI00 input
P22	INTP1 input	P26	CTI11 input
P23	INTP2 input	P27	CLR1 input

Caution: NMI input accepts interrupt requests regardless of the interrupt enable/disable status. See Chapter 11 for details.

Port 2 contains a software pull-up resistor in its six bits P22-P27. Use bit 2 of the pull-up resistor option register (PU0) to specify a built-in pull-up resistor.

Caution: P20 and P21 pins do not contain any software pull-up resistor.

The functions of the dual-function pins are described below.

(a) NMI input

External non-maskable interrupt request input pin. Use the external interrupt mode register (INTM0) to specify one of the detection modes, rising edge or falling edge, for this pin.

This pin contains an analog delay noise reduction circuit.

(b) INTP0 input

External interrupt request input pin. Use the external interrupt mode register (INTM0) to specify one of the three detection modes, rising edge, falling edge, or rising and falling edges, for this pin.

(c) INTP1, INTP2

External interrupt request input pins. Use the external interrupt mode register (INTM0) to specify one of the three detection modes, rising edge, falling edge, or rising and falling edges, for these pins.

(d) CTI00 input

Capture trigger input pin of the super timer unit.

This pin detects the rising edge and captures the contents of the free running counter (FRC) in capture register 2 (CPT2H, CPT2L). (Capture register 2 consists of 18 bits.)

When timer 0 clear pulse is internally generated, this pin becomes the count clock pulse input pin of the event counter (EC), which is in the input block of timer 0.

(e) CTI10

Capture trigger input pin of the super timer unit. Use the external capture input mode register (INTM1) to specify one of the detection modes, rising edge or rising and falling edges, for this pin.

Valid edge detection signal is divided into programmable units by the event divider, which is in the input block of timer 1, and becomes the capture trigger of capture register 3 (CPT3) of the free running counter (FRC).

It also becomes capture trigger of the capture register 2 (CPT12) of timer 1, if so specified in the capture mode register (CPTM).

(f) CTI11

Capture trigger input pin of the super timer unit.

Use the external capture input mode register (INTM1) to specify one of the detection modes, rising edge or falling edge, for this pin.

It becomes the capture trigger of capture register 2 (CPT12) of timer 1, if so specified in the capture mode register (CPTM).

(g) CLR1

Timer 1 clear input pin of the super timer unit.

Use the external capture input mode register (INTM1) to specify one of the detection modes, rising edge or falling edge, for this pins.

CLR1 pin contains a digital noise reduction circuit.

This pin clears timer 1 when CLR1 is input.
This pin also functions as the capture trigger of capture register 0 (CPT0) of the free running counter (FRC) if the capture mode register (CPTM) is set.

(4) P30-P37 (Port 3): P30-P33: 3-state input/output
P34, P35: Input
P36, P37: Input/output

Eight-bit input/output pins of port 3 (P30-P33 are a 4-bit I/O port with an output latch, P34 and P35 are an input port, and P36 and P37 are an I/O port).

In addition to the function of an I/O port, these pins also function as various control signal pins.

Use the port 3 mode control register (PMC3) to specify operation mode of the individual P30-P33, P36, and P37 pins bit by bit as shown in Table 2-2.

Pins P34 and P35 can read pin levels (operate as an I/O port) even though they are fixed to the control mode.

P34, P35, P36 and P37 pins employ Schmitt trigger input to eliminate operational mistakes caused by noise.

When $\overline{\text{RESET}}$ is input, these pins are set to the input port (output high impedance) and the contents of the output latch become indefinite.

Port 3 contains a software pull-up resistor.

Use bit 3 of pull-up resistor option register (PUO) to specify built-in pull-up resistor.

Table 2-2 Port 3 Modes

(n = 0 - 7)

PMC3n P3n	PMC3n = 0	PMC3n = 1
	Port mode	Control signal input/output mode
P30	I/O port	PT000 output
P31	I/O port	PT001 output
P32	I/O port	PT002 output
P33	I/O port	PT010 output
P34	—(*)	CLR0 input
P35	—(*)	SI input
P36	I/O port	SO/SB0 input/output
P37	I/O port	$\overline{SCK}$ input/output

* Pins P33 and P34 can read the pin levels (input port operation) even though they are fixed to the control signal input/output mode.

(a) Port mode

Use the port 3 mode register (PM3) to specify input or output bit by bit to pins P30-P33, if these pins entered the port mode via the PMC3 register.

Pins P34 and P35 can read the pin levels even though they are fixed to the control signal input/output mode.

Pins P36 and P37 function as an I/O port, if these pins entered the port mode via the PMC3 register.

(b) Control signal input/output mode

(i) PT000, PT001, PT002, PT010

Programmable timer output pins of the super timer unit.

(ii) CLR0

Timer 0 clear signal input pin of the super timer unit. Both rising and falling edges are active.

(iii) SI

Serial data input pin.

(iv) SO/SB0

SO is a serial data output pin in the 3-wire serial I/O mode. SB0 is a serial bus input/output pin in the SBI mode.

(v) $\overline{\text{SCK}}$

Serial clock input/output pin.

(5) P40-P47 (Port 4): 3-state input/output

Eight-bit input/output pins of port 4 (8-bit I/O port with an output latch). Use the memory mapping register (MM) to specify input or output 8 bits by 8 bits for these pins.

These bits also function as a time multiplexing address/data bus (AD0-AD7) for externally expanded memory or I/O.

Remark: See Section 3.3 for details of the external expansion function.

When $\overline{\text{RESET}}$ is input, these pins are set to the input port (output high impedance) and the contents of the output latch become indefinite.

Port 4 contains a software pull-up resistor.

Use bit 4 of the pull-up resistor option register (PU0) to specify a built-in pull-up resistor.

(6) P50-P57 (Port 5): 3-state input/output

Eight-bit input/output pins of port 5 (8-bit I/O port with an output latch). Use the port 5 mode register (PM5) to specify input or output bit by bit for these pins.

These pins also function as an address bus (A8-A15) for externally expanded memory or I/O.

When $\overline{\text{RESET}}$ is input, these pins are set to the input port (output high impedance) and the contents of the output latch become indefinite.

Port 5 contains a software pull-up resistor.

Use bit 5 of the pull-up resistor option register (PU0) to a specify built-in pull-up resistor.

(7) P60-P67 (Port 6): P60-P63: Output port
P64-P67: 3-state input/output

Eight-bit input/output pins of port 6 (8-bit I/O port with an output latch).

In addition to functioning as a port, these pins also output control signals as described in Table 2-3. Use the memory mapping register (MM) to specify the control signal output mode.

When $\overline{\text{RESET}}$ is input, P60-P63 pins are set to output port and output low level. When $\overline{\text{RESET}}$ is input, P64-P67 are set to input port (output high impedance).

When $\overline{\text{RESET}}$ is input, the contents of the output latch become x0H.

Table 2-3 Port 6 Operation Modes

Pin	Port mode	Control signal output mode	To operate port 6 as a control pin
P60	Output port	—	—
P61			
P62			
P63			
P64	I/O port	$\overline{\text{RD}}$ output	Specify $\overline{\text{EA}}$ pin = 0. Specify external expansion mode in MM2-MM0 bit of MM register.
P65	I/O port	$\overline{\text{WR}}$ output	
P66	I/O port	—	—
P67			

(a) Port mode

P60-P63 pins are output dedicated port.

P64-P67 pins can be specified to input or output via the port 6 mode register (PM6) bit by bit.

(b) Control signal output mode

(i) $\overline{RD}$ (Read Strobe)

Strobe signal output pin for reading external memory. Input low level in the $\overline{EA}$ pin or use the memory mapping register (MM) to specify the operation modes.

(ii) $\overline{WR}$ (Write Strobe)

Strobe signal output pin for writing to external memory. Input low level in the $\overline{EA}$ pin or use the memory mapping register (MM) to specify this operation mode.

Remark: See Section 3.3 for the $\overline{RD}$ and $\overline{WR}$ operations.

P64-P67 pins contain a software pull-up resistor.

Use bit 6 of the pull-up resistor option register (PUO) to specify built-in pull-up resistor.

Cautions 1. P60-P63 pins do not contain a software pull-up resistor.

2. P60-P63 pins output low level after reset is released.

(8) P70-P71 (Port 7): 3-state input/output

Two-bit input/output pins of port 7 (2-bit I/O port with an output latch).

Use the port 7 mode register (PM7) to specify input or output 2 bits by 2 bits for these pins.

When $\overline{\text{RESET}}$ is input, port 7 is set to input port (output high impedance) and the contents of the output latch become indefinite.

Port 7 contains a software pull-up resistor.

Use bit 7 of the pull-up resistor option register (PUO) to specify built-in pull-up resistor.

(9) PWM0, PWM1 (Pulse Width Modulation Output): Output

PWM pulse output pins from the super timer unit.

(10) PTO10 (Programmable Timer Out): Output

Output pin from timer 1 (TM1) of the super timer unit.

(11) ANI0-ANI7 (Analog Input): Input

Eight analog signal input pins to A/D converter.

(12) AV_{REF} (Reference Voltage)

Reference voltage input pin of the A/D converter and power supply pin of the A/D converter.

(13) AV_{SS} (Analog V_{SS})

GND pin of the A/D converter.

(14) $\overline{EA}$ (External Access): Input

The uPD78134A, uPD78136, and uPD78138 have a ROM-less mode, in which external memory which is not on-chip in the ROM accesses the program memory.

When the $\overline{EA}$ pin is set to high, the on-chip ROM is accessed. When the $\overline{EA}$ pin is set to low, the external memory is accessed in the ROM-less mode.

Fix this pin to high to use the on-chip 16K, 24K, or 32K-byte mask ROM.

(15) ASTB/CLO (Address Strobe/Clock Output): Output

Timing signal output pin that externally latches address data for accessing external memory.

In the single-chip mode, i.e., external memory is not used, this pin functions as a clock output pin that supplies clock signals to external peripheral LSI chips and microcomputers.

(16) X1, X2 (Crystal)

Crystal connection pin for system clock oscillation. Input the externally supplied clock signal to X1 and input its inverted phase to X2.

(17) $\overline{\text{RESET}}$ (Reset): Input

Low level active reset input pin.

(18) V_{DD}

Positive power supply pin.

(19) V_{SS}

Ground voltage pin.

2.2.2 PROM programming mode (uPD78P138)

(1) PROG: Input

Input pin that sets the uPD78P138 in the PROM programming mode. When the input voltage of this pin is at 12.5 V and when $\overline{\text{RESET}}$ input is low, the uPD78P138 enters the PROM programming mode.

(2) $\overline{\text{RESET}}$: Input

Input pin that sets the uPD78P138 in the PROM programming mode. When this pin is low and when the input voltage of this pin is at 12.5 V, the uPD78P138 enters the PROM programming mode.

(3) A0-A14 (Address Bus): Input

Address bus that selects on-chip PROM address (0000H-7FFFH).

(4) D0-D7 (Data Bus): Input/output

Data bus. Programs are written in or read from the on-chip PROM via this bus.

(5) $\overline{CE}$ (Chip Enable): Input

This pin inputs the enable signal from the on-chip PROM. When this signal is active, programs can be written or read.

(6) $\overline{OE}$ (Output Enable): Input

This pin inputs the read strobe signal in the on-chip PROM. When $\overline{CE}$ = L, activate this signal. The program (one byte) in the on-chip PROM cell selected by A0-A14 is read on to D0-D7.

(7) V_{PP} (Programming Power Supply)

Power supply pin for program writing. When V_{PP} = 12.5 V, $\overline{OE}$ = H, and $\overline{CE}$ = L, programs on D0-D7 are written in the on-chip PROM cell selected by A0-A14.

(8) V_{DD}

Positive power supply pin.

(9) V_{SS}

Ground voltage pin.

2.3 Input/Output Circuits and Connection of Unused Pins

Table 2-4 I/O Circuit Type of Each Pin and Connection of Unused Pins

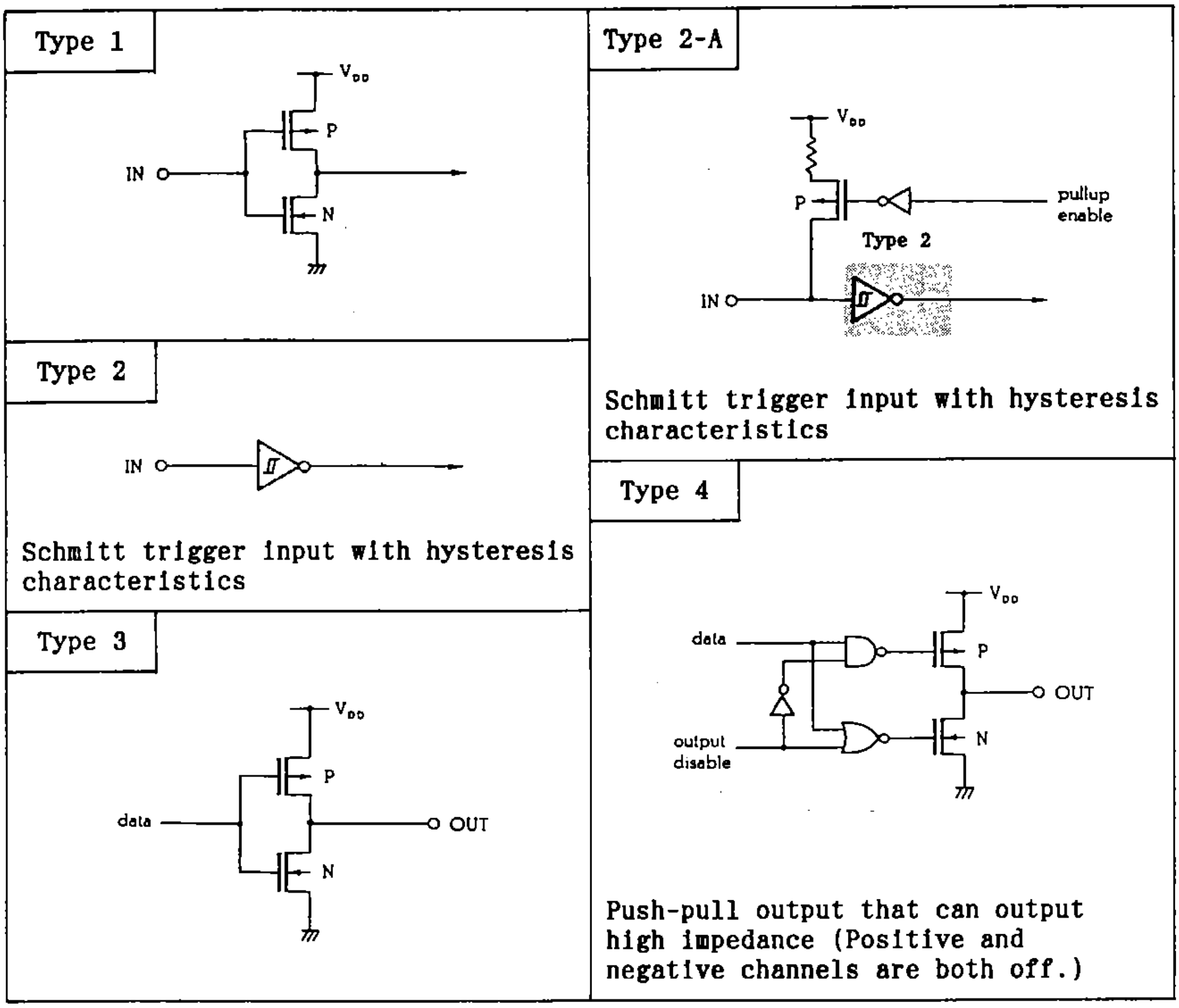
Pin	I/O circuit type	Recommended connection of unused pins
P00-P07	4	Open
P10-P17	5-A	Input: Connected to V_{DD} via pull-up resistor Output: Open
P20/NMI	2	Connected to V_{DD}
P21/INTP0		
P22/INTP1	2-A	
P23/INTP2		
P24/CTI10		
P25/CTI00		
P26/CTI11		
P27/CLR1		
P30/PTO00	5-A	Input: Connected to V_{DD} via pull-up resistor Output: Open
P31/PTO01		
P32/PTO02		
P33/PTO11		
P34/CLR0	2-A	Connected to V_{DD}
P35/SI		

(to be continued)

Table 2-4 I/O Circuit Type of Each Pin and Connection of Unused Pins (Cont'd)

Pin	I/O circuit type	Recommended connection of unused pins
P36/SO/SB0	10-A	Connected to V_{DD} via pull-up resistor
P37/$\overline{SCK}$	8-A	
P40-P47/AD0-AD7	5-A	Input: Connected to V_{DD} via pull-up resistor Output: Open
P50-P57/A8-A15		
P60-P63	3	Open
P64/$\overline{RD}$	5-A	Input: Connected to V_{DD} via pull-up resistor Output: Open
P65/$\overline{WR}$		
P66, P67		
P70, P71		
PWM0, PWM1	3	Open
PTO10		
ANI0-ANI7	7	Connected to V_{SS}
$\overline{EA}$	1	–
ASTB/CLO	3	Open
$\overline{RESET}$	2	–
AV_{REF}	-	Connected to V_{SS}
AV_{SS}		

Fig. 2-1 Pin Input/Output Circuits



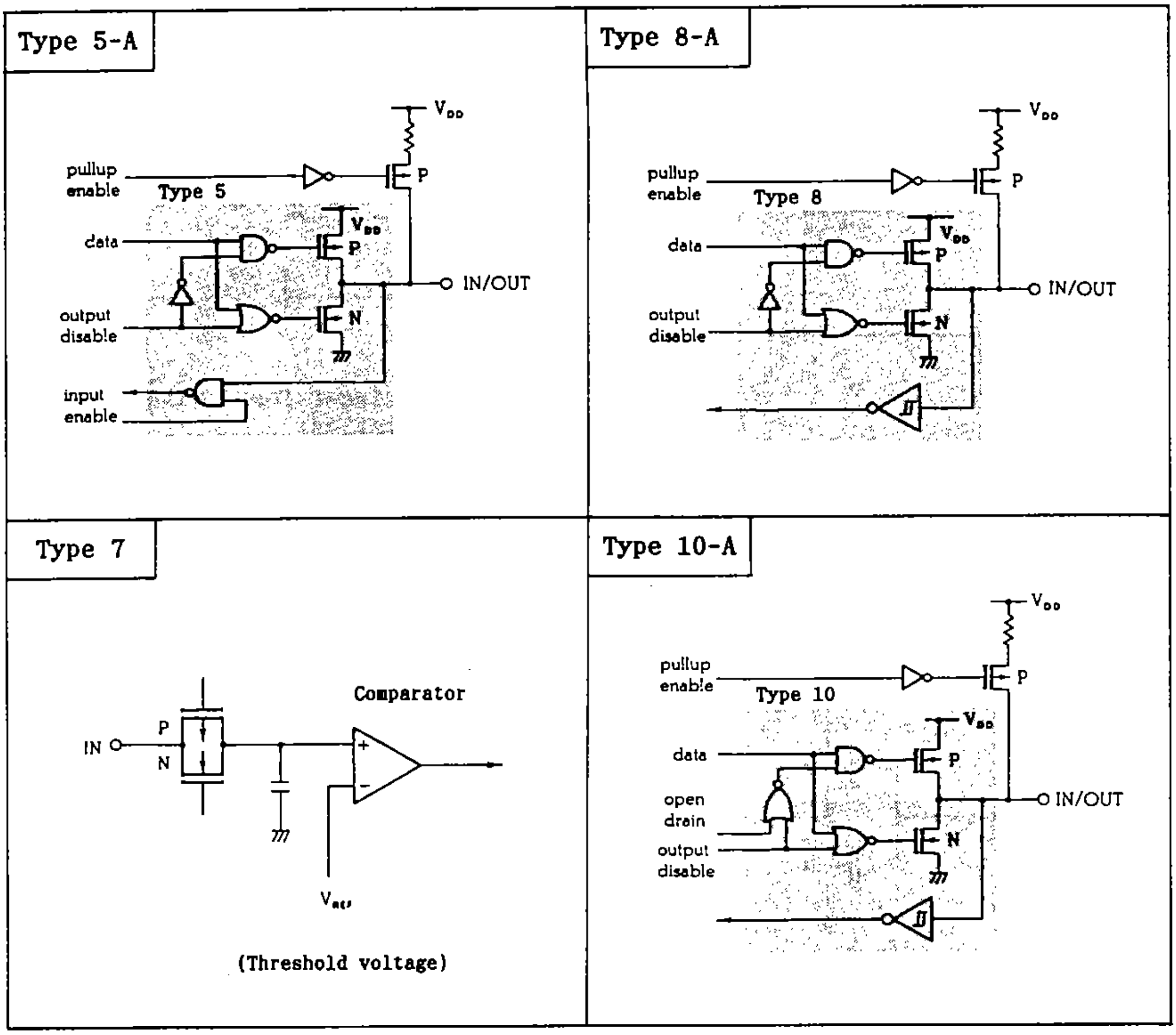
Type 2: Schmitt trigger input with hysteresis characteristics

Type 2-A: Schmitt trigger input with hysteresis characteristics

Type 4: Push-pull output that can output high impedance (Positive and negative channels are both off.)

(to be continued)

Fig. 2-1 Pin Input/Output Circuits (Cont'd)



3.1 Memory Space

The uPD78138 allows access to a memory space of up to 64K bytes. Figures 3-1 to 3-3 show the memory space. Program memory is mapped differently according to the microcomputer products uPD78134A, uPD78136, uPD78138, and uPD78P138 and the state of the external access pin ($\overline{EA}$).

(1) $\overline{EA}$ = high

Program memory is mapped in internal ROM and external memory. The sizes of the memory areas differ depending on the products.

External memory is accessed in the external memory expansion mode. The external memory area can also be used as data memory.

Data memory is mapped in the internal RAM. The sizes of the memory areas differ depending on the products.

Table 3-1 Memory Areas of Each Product

Product	Internal ROM	External memory	Internal RAM
uPD78134A	16384 bytes (0000H-3FFFH)	48512 bytes (4000H-FD7FH)	384 bytes (FD80H-FEFFH)
uPD78136	24576 bytes (0000H-5FFFH)	40064 bytes (6000H-FC7FH)	640 bytes (FC80H-FEFFH)
uPD78138 uPD78P138	32768 bytes (0000H-7FFFH)	31872 bytes (8000H-FC7FH)	

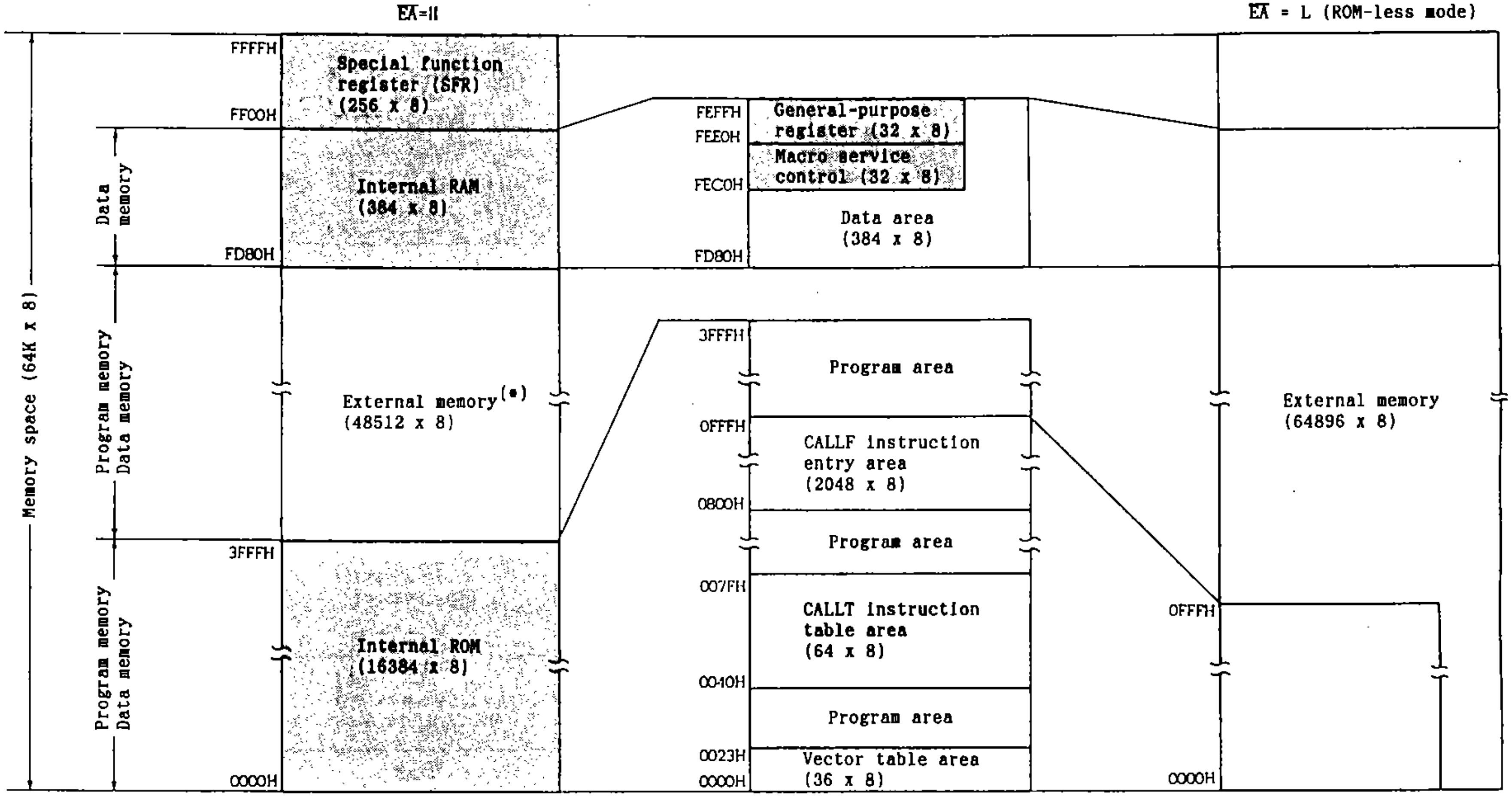
(2) $\overline{\text{EA}}$ = low (ROM-less mode)

All program memory is mapped in external memory in the ROM-less mode. This area can also be used as data memory.

Data memory is mapped in the internal RAM.

In the ROM-less mode, the memory areas are assigned to the same locations in the uPD78134A, uPD78136, uPD78138, and uPD78P138.

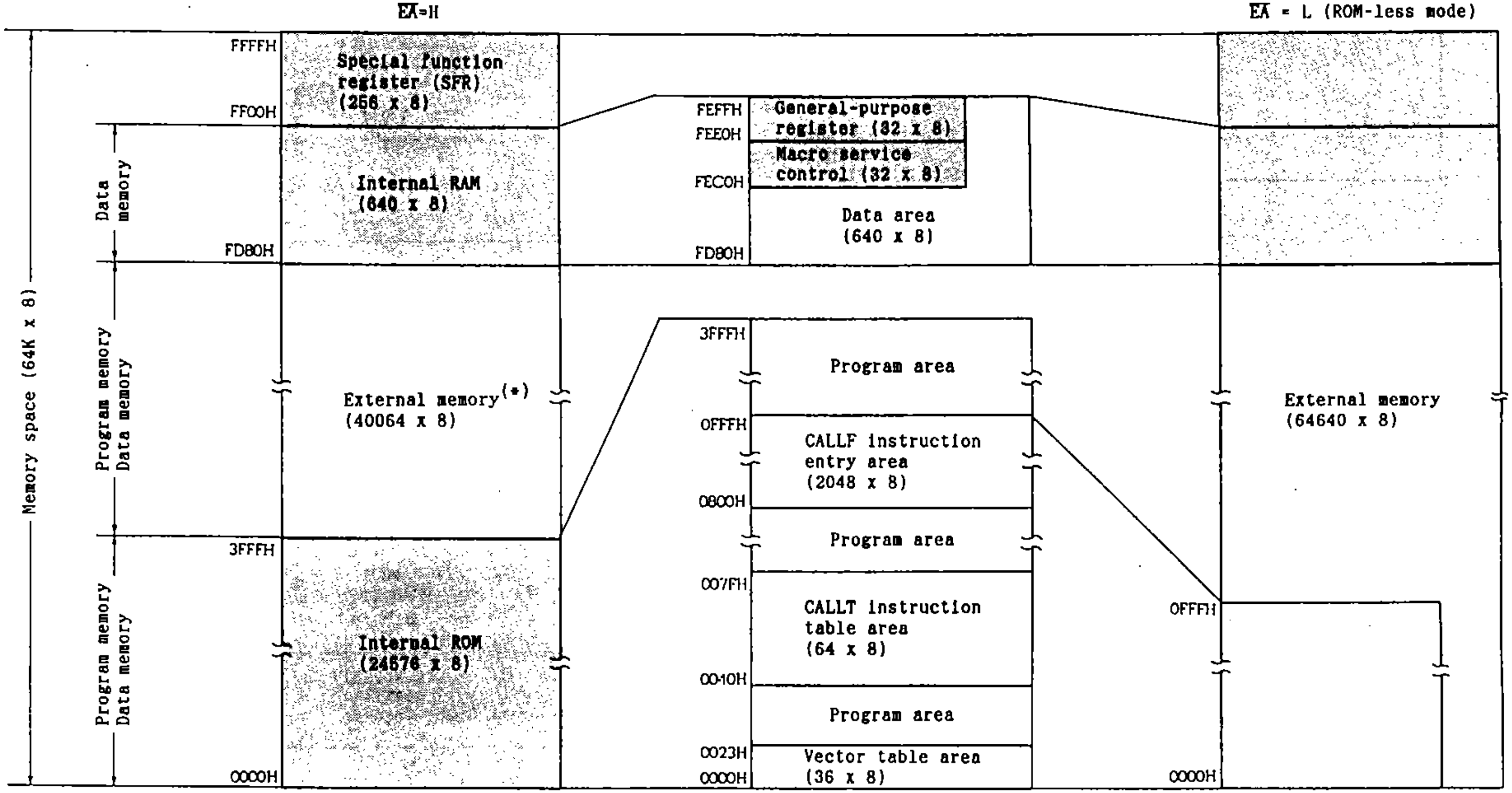
Fig. 3-1 Memory Map (uPD78134A)


EA=H
EA = L (ROM-less mode)
FFFFH
Special function register (SFR) (256 x 8)
FF00H
Internal RAM (384 x 8)
FD80H
Data memory
External memory(*) (48512 x 8)
Program memory Data memory
3FFFH
Internal ROM (16384 x 8)
Program memory Data memory
0000H
Memory space (64K x 8)
FEFFH
General-purpose register (32 x 8)
FEE0H
Macro service control (32 x 8)
FEC0H
Data area (384 x 8)
FD80H
3FFFH
Program area
0FFFH
CALLF instruction entry area (2048 x 8)
0800H
Program area
007FH
CALLT instruction table area (64 x 8)
0040H
Program area
0023H
Vector table area (36 x 8)
0000H
External memory (64896 x 8)
0FFFH
0000H

* Accessed in the external memory expansion mode

Remark: ▨ indicates an internal memory area.

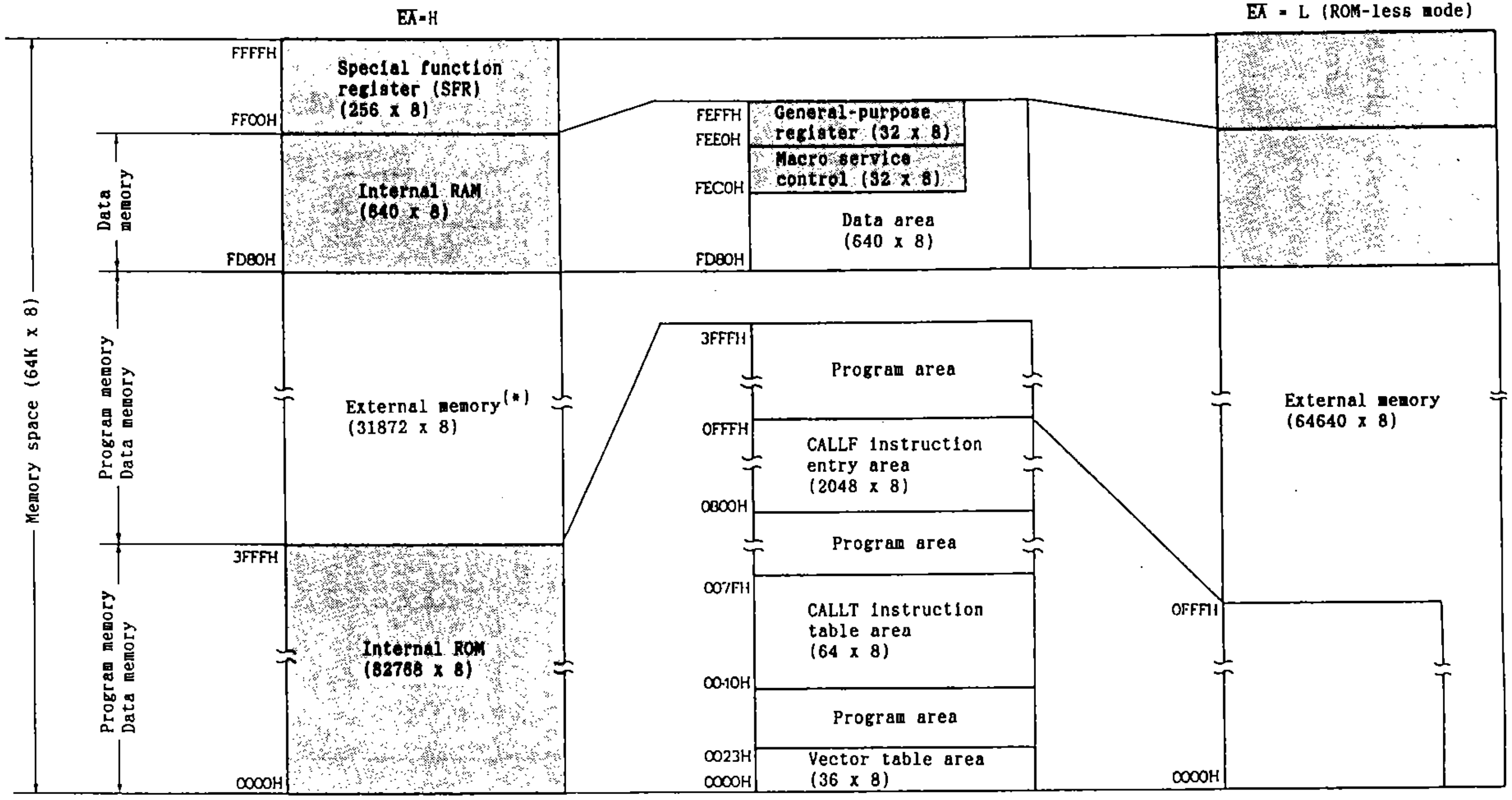
Fig. 3-2 Memory Map (uPD78136)


EA=H
EA = L (ROM-less mode)
FFFFH
Special function register (SFR) (256 x 8)
FF00H
Data memory
Internal RAM (640 x 8)
FD80H
Memory space (64K x 8)
Program memory Data memory
External memory(*) (40064 x 8)
3FFFH
Program memory Data memory
Internal ROM (24576 x 8)
0000H
FEFFH
General-purpose register (32 x 8)
FEE0H
Macro service control (32 x 8)
FEC0H
Data area (640 x 8)
FD80H
3FFFH
Program area
0FFFH
CALLF instruction entry area (2048 x 8)
0800H
Program area
007FH
CALLT instruction table area (64 x 8)
0040H
Program area
0023H
Vector table area (36 x 8)
0000H
External memory (64640 x 8)
0FFFH
0000H

* Accessed in the external memory expansion mode

Remark: ▨ indicates an internal memory area.

Fig. 3-3 Memory Map (uPD78138 and uPD78P138)



* Accessed in the external memory expansion mode

Remark: ▨ indicates an internal memory area.

3.1.1 Internal program memory space

An area from 0000H to 3FFFH (16K bytes: uPD78134A), from 0000H to 5FFFH (24K bytes: uPD78136), or from 0000H to 7FFFH (32K bytes: uPD78138 and uPD78P138) is assigned to internal program memory (internal ROM). This area holds programs and table data, and it is usually addressed by the program counter (PC).

The internal program memory space is assigned to the following areas:

(1) Vector table area

A 22-byte area from 0000H to 0015H holds program start addresses used when branches are caused by $\overline{\text{RESET}}$ input and interrupt requests. The lower eight bits of a 16-bit program start address are stored at an even address and the higher eight bits are stored at an odd address.

Table 3-2 Vector Table

Vector table address	Interrupt request
0000H	Reset (RESET input)
0002H	NMI
0004H	INTP0
0006H	INTCPT3
0008H	INTCPT2
000AH	INTCR12
000CH	INTCR00
000EH	INTCLR1
0010H	INTCR10
0012H	INTCR01
0014H	INTCR02
0016H	INTCR11
0018H	INTCPT10
001AH	INTTM
001CH	INTCSI
001EH	INTTB
0020H	INTP1/INTAD
0022H	INTP2

(2) CALLT instruction table area

A 64-byte area from 0040H to 007FH holds subroutine entry addresses for a 1-byte call instruction (CALLT).

(3) CALLF instruction entry area

Locations from 0800H to 0FFFH can be addressed for a direct subroutine call with a 2-byte call instruction (CALLF).

3.1.2 Internal data memory space

An area from FD80H to FEFFH for the uPD78134A or an area from FC80H to FEFFH for the uPD78136, uPD78138, and uPD78P138 is assigned to 640 bytes of general-purpose static RAM. In 32 bytes from FEE0H to FEFFH in that area four banks of general registers are mapped, and in another 32-byte area from FEC0H to FEDFH in the area, macro service channels are mapped.

Data memory can also be used as stack memory.

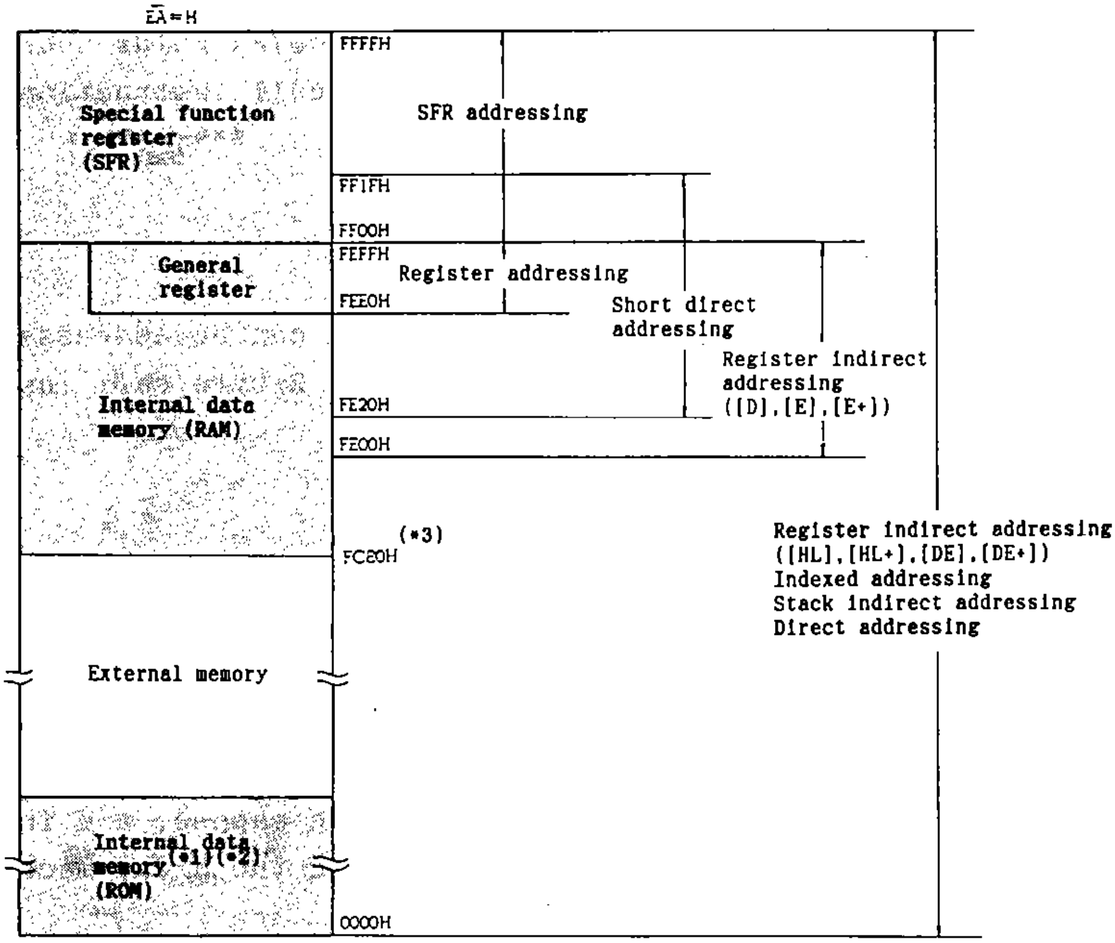
3.1.3 Special function register (SFR) space

The special function registers (SFRs) for the on-chip peripheral hardware are mapped in an area from FF00H to FFFFH. Locations in which no SFR is mapped cannot be accessed. (See Section 3.2.5.)

3.1.4 Data memory addressing

Figure 3-4 shows the memory map of data memory space and SFR space and the applied addressing scheme.

Fig. 3-4 Memory Map and Addressing of Data Memory



EA=H
Special function register (SFR)
General register
Internal data memory (RAM)
External memory
Internal data memory (ROM) (*1)(*2)
FFFFH
FF1FH
FF00H
FEFFH
FEE0H
FE20H
FE00H
FC80H
(*3)
0000H
SFR addressing
Register addressing
Short direct addressing
Register indirect addressing ([D],[E],[E+])
Register indirect addressing ([HL],[HL+],[DE],[DE+])
Indexed addressing
Stack indirect addressing
Direct addressing

*1 If $\overline{EA}$ is low, external memory is mapped.

*2 Do not place the stack pointer in the SFR area or, if $\overline{EA}$ is high, in the ROM area.

*3 FD80H for the uPD78134A; FC80H for the uPD78136, uPD78138, and uPD78P138

Remark: ▨ indicates internal memory.

(1) Register addressing

With this addressing scheme, a general register mapped into a particular location in data memory is addressed. The addressed general register is in the register bank specified by the RBS0 and RBS1 flags in the PSW.

Coding example: XCH A,r
When specifying the C register in r, code the following:

```
XCH  A,C
```

(2) Short direct addressing

The short direct addressing scheme applies to an area from FE20H to FEFFH in the internal data memory space and an area from FF00H to FF1FH in the SFR space.

In accessing 16-bit data, 2 bytes of data specified with even-odd consecutive addresses are accessed regardless of whether the address specification data is odd or even.

Coding example: ADDC saddr,A
When address FE50H is specified in saddr, code the following:

```
ADDC  0FE50H,A
```

(3) SFR addressing

The SFR addressing scheme applies to the special function registers (SFRs) mapped in the SFR area from FF00H to FFFFH.

Coding example: MOV A,sfr

When the SIO register is specified in sfr, code the following:

```
MOV  A,SIO
```

(4) Register indirect addressing

The register indirect addressing scheme addresses a data memory location indirectly through the contents of the register coded in an operand. The specified register is in the register bank specified by the RBS0 and RBS1 flags in the PSW.

Register indirect addressing with the HL register pair or the DE register pair can address any location in the entire space including internal ROM.

Only the MOV instruction can automatically increment the contents of the register or the register pair by one after the instruction is executed.

Coding example: SUB A,[r4]

When the E register is specified in r4, code the following:

```
SUB  A,[E]
```

(5) Indexed addressing

The indexed addressing scheme uses the result of adding the 16-bit immediate data coded in an operand and the contents of the 8-bit register coded in another operand. The specified 8-bit register is in the register bank specified by the RBS0 and RBS1 flags in the PSW.

Any location in the entire space including internal ROM can be addressed.

Coding example: MOV A,word[r1]

When FEA0H is specified in word and the B register is specified in r1, code the following:

```
MOV  A,0FEA0H[B]
```

(6) Stack indirect addressing

With the stack indirect addressing scheme, the 64K-byte stack area can be addressed indirectly according to the stack pointer (SP) content.

This addressing scheme applies when the PUSH or POP instruction is executed, when save or restore is performed in interrupt handling, or when a subroutine call or a return from that subroutine is performed.

Coding example: PUSH rp

When the DE register pair is specified in rp, code the following:

```
PUSH  DE
```

3.2 Registers

3.2.1 Program counter (PC)

The program counter is a 16-bit binary counter to hold address information of the instruction to be executed next. Usually, the program counter is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data or the contents of a register are set in the PC.

$\overline{\text{RESET}}$ input causes the data at 0000H in the internal ROM to be loaded into the lower eight bits of PC, and the data at 0001H to be loaded into the higher eight bits.

Fig. 3-5 Format of Program Counter (PC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

3.2.2 Program status word (PSW)

The program status word is an 8-bit register consisting of flags set or reset according to the result of executing an instruction. All the eight bits can be read from or written into the PSW at a time, and a particular flag can also be operated on with a bit manipulation instruction. The PSW contents are saved in a stack when an interrupt request is generated or when the PUSH PSW instruction is executed, and it is restored by the RETI or POP PSW instruction.

$\overline{\text{RESET}}$ input clears all the flags, setting the PSW to 02H.

Fig. 3-6 Program Status Word (PSW) Format

	7	6	5	4	3	2	1	0
PSW	IE	Z	RBS1	AC	RBS0	0	ISP	CY

Caution: Be sure to write 0 in bit 2.

(1) Carry flag (CY)

The carry flag retains an overflow or underflow that may be generated at the execution of an addition or subtraction instruction. It also retains a value shifted out as a result of executing a shift rotate instruction. When a bit manipulation instruction is executed, it can function as a bit accumulator.

(2) Interrupt priority status flag (ISP)

The interrupt priority status flag controls the priority of maskable vector interrupts that can be accepted currently. See Table 3-3.

When a maskable vector interrupt is accepted, the contents of the priority specification flag (PR0) of that interrupt are stored. For the priority specification flag, see Chapter 11.

Table 3-3 ISP Flag Format

ISP	Maskable vector interrupt that can be accepted
0	Interrupt with its priority specification flag set to 0 (high-priority interrupt)
1	Interrupt can be accepted regardless of the contents of the priority specification flag

(3) Register bank selection flags (RBS0 and RBS1)

Two bits of register bank selection flags select one of the four register banks.

Table 3-4 Specification of a Register Bank

RBS1	RBS0	Specified register bank
0	0	Register bank 0
0	1	Register bank 1
1	0	Register bank 2
1	1	Register bank 3

(4) Auxiliary carry flag (AC)

When a carry out of bit 3 or a borrow from bit 3 is produced as a result of an arithmetic/logical operation, the auxiliary carry flag is set to 1. Otherwise, it is reset to 0. The flag is used when the BCD correction instruction is executed.

(5) Zero flag (Z)

The zero flag is set when the arithmetic/logical operation result is zero. Otherwise, this flag is reset.

(6) Interrupt request enable flag (IE)

The interrupt request enable flag controls the CPU to enable or disable an interrupt request. If the flag is 0, the DI state is entered, disabling any interrupt request other than nonmaskable interrupts. If the flag is 1, the EI state is entered, enabling an interrupt request according to the corresponding interrupt mask flag.

The interrupt request enable flag is set to 1 by executing the EI instruction, and it is reset to 0 by executing the DI instruction or after an interrupt is accepted.

3.2.3 Stack pointer (SP)

The stack pointer is a 16-bit register to hold the start address of the stack area (LIFO form).

Stack memory can be placed at any location in the data memory area(Note). It can also be placed in external memory.

The SP content is predecremented when stack memory is written to (save operation), and is postincremented when stack memory is read from (restoration).

SP can be accessed with special instructions.

Note: . FD80H to FEFFH for the uPD78134A
. FC80H to FEFFH for the uPD78136, uPD78138, and uPD78P138

Fig. 3-7 Stack Pointer (SP) Format

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

Caution: RESET input makes the SP content undefined. Before calling a subroutine, be sure to initialize SP with an initialization program.

Fig. 3-8 Data Saved in Stack Memory

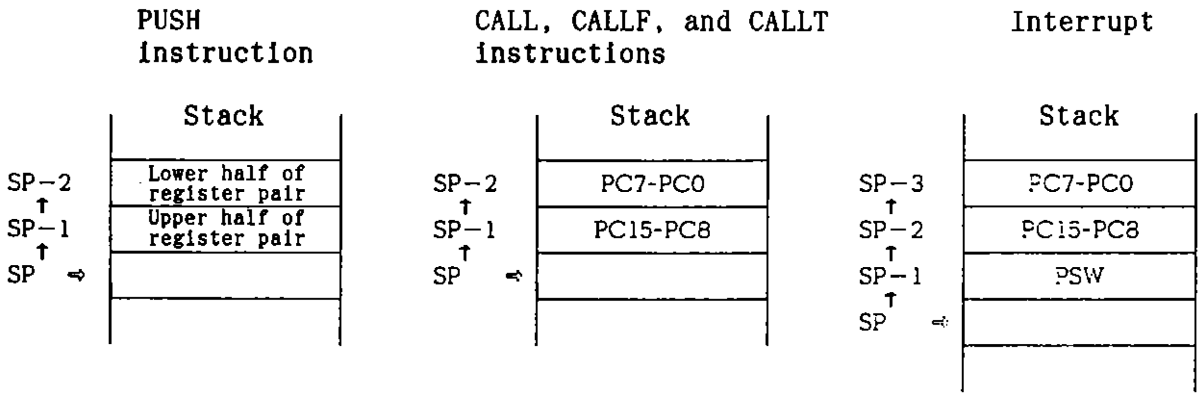
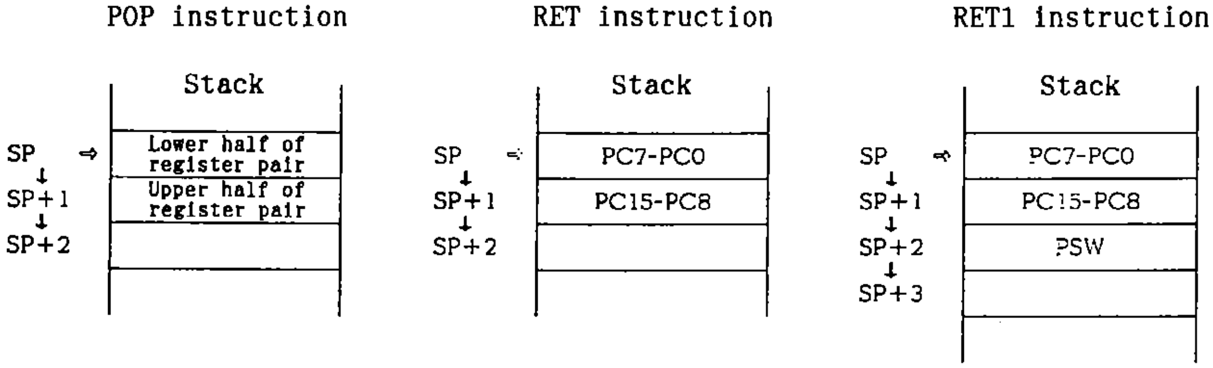


Fig. 3-9 Data Restored from Stack Memory



3.2.4 General registers

General registers are mapped in data memory locations from FEE0H to FEFFH. Eight registers X, A, C, B, E, D, L, and H, each consisting of eight bits, are grouped into one bank. There are four banks of registers in total.

The register bank valid for instruction execution depends on the setting of the register bank selection flags (RBS0 and RBS1) in PSW.

The eight bits of each general register are manipulated at one time. Pairs of 8-bit registers (AX, BC, DE, and HL) can also be manipulated in 16-bit units.

Function names X, A, C, B, E, D, L, H, AX, BC, DE, and HL are assigned to the individual registers to identify their particular functions. Also, the registers can be coded with absolute names (R0 to R7, RP0 to RP3). In the uPD78138, function names and absolute names correspond on a one-to-one basis. See Table 3-5.

The general register area can be addressed for access as normal data memory regardless of whether the registers are mapped there.

With the four register banks, uPD78138 programs can be coded so that different register banks are used between normal processing and interrupt processing for efficiency.

Table 3-5 Correspondence between Function Names and Absolute Names

Function name	Absolute name
X	R0
A	R1
C	R2
B	R3
E	R4
D	R5

Function name	Absolute name
L	R6
H	R7
AX	RP0
BC	RP1
DE	RP2
HL	RP3

Fig. 3-10 Format of General Registers

	8-bit processing			16-bit processing
FEE0H	A E1H	X E0H	Register bank 3 (RBS1, RBS0 = 11)	AX E0H
	B E3H	C E2H		BC E2H
	D E5H	E E4H		DE E4H
	H E7H	L E6H		HL E6H
	A E9H	X E8H	Register bank 2 (RBS1, RBS0 = 10)	AX E8H
	B EBH	C EAH		BC EAH
	D EDH	E ECH		DE ECH
	H EFH	L EEH		HL EEH
	A F1H	X F0H	Register bank 1 (RBS1, RBS0 = 01)	AX F0H
	B F3H	C F2H		BC F2H
	D F5H	E F4H		DE F4H
	H F7H	L F6H		HL F6H
	A F9H	X F8H	Register bank 0 (RBS1, RBS0 = 00)	AX F8H
	B FBH	C FAH		BC FAH
	D FDH	E FCH		DE FCH
FEFFH	H FFH	L FEH		HL FEH

3.2.5 Special function registers (SFRs)

Special function registers are assigned special functions, such as the mode register and control register of the on-chip peripheral hardware, and are mapped in the 256-byte space from FF00H to FFFFH.

The 32-byte area from FF00H to FF1FH can be accessed by short direct addressing. So SFRs which are accessed frequently, such as a timer compare register, capture register, and ports, can be mapped in that area, allowing short-word data processing with fewer clock pulses.

SFRs can be manipulated in various ways with arithmetic/logical instructions, move instructions, and bit manipulation instructions.

Caution: Addresses not assigned SFRs cannot be accessed. Access to such an address may cause normal operation to fail.

Table 3-6 lists the special function registers (SFRs).
The items in Table 3-6 mean:

. Abbreviation

A symbol indicating the address of an included SFR. It can be coded in the operand field of an instruction.

- R/W

Indicates whether SFR can be read from and/or written to.

R/W:	Can be read from and written to.
R:	Can be read from.
W:	Can be written to.

- Manipulation bit unit

Indicates the number of bits in SFR that can be manipulated at one time. An SFR which allows 16-bit manipulation can be coded in the sfrp operand. For address specification, an even address is specified.

An SFR which allows bit-based manipulation can be coded in a bit manipulation instruction.

- At resetting

Indicates a register state immediately after $\overline{\text{RESET}}$ input.

Table 3-6 Special Function Registers (SFRs)

Address	Special function register (SFR) name	Abbreviation	R/W	Manipulation bit unit			At resetting
				1	8	16	
FF00H	Port 0	P0	R/W	o	o	-	Undefined
FF01H	Port 1	P1		o	o		
FF02H	Port 2	P2	R	o	o	-	
FF03H	Port 3	P3	R/W	o	o		
FF04H	Port 4	P4		o	o	-	
FF05H	Port 5	P5		o	o		
FF06H	Port 6	P6		o	o	-	xxxx0000
FF07H	Port 7	P7		o	o		Undefined
FF08H	16-bit timer 0 compare register 0	CR00		-	-	o	
FF09H				-	-		
FF0AH	16-bit timer 0 compare register 1	CR01		-	-	o	
FF0BH				-	-		
FF0CH	16-bit timer 0 compare register 2	CR02		-	-	o	
FF0DH				-	-		
FF0EH	16-bit timer 1 compare register 0	CR10		-	-	o	
FF0FH				-	-		
FF10H	16-bit timer 1 compare register 1	CR11		-	-	o	
FF11H				-	-		
FF12H	16-bit timer 1 capture register 2	CR12	R	-	-	o	
FF13H				-	-		

(to be continued)

Table 3-6 Special Function Registers (SFRs) (Cont'd)

Address	Special function register (SFR) name	Abbreviation	R/W	Manipulation bit unit			At resetting
				1	8	16	
FF14H	16-bit FRC capture register 0	CPT0	R	-	-	o	Undefined
FF15H				-	-		
FF16H	16-bit FRC capture register 1	CPT1		-	-	o	
FF17H				-	-		
FF18H	18-bit FRC capture register 2	CPT2H		-	-	o	
FF19H				-	-		
FF1AH	16-bit FRC capture register 3	CPT3		-	-	o	
FF1BH				-	-		
FF1CH	18-bit FRC capture register 2	CPT2L		o	o	-	xx000000
FF1DH	Prescaler mode register	PRM3	R/W	o	o	-	0xxxx000
FF1EH	16-bit timer 2 compare register	CR20		-	-	o	Undefined
FF1FH				-	-		
FF20H	Port 0 mode register	PM0	W	-	o	-	FFH
FF21H	Port 1 mode register	PM1		-	o	-	
FF23H	Port 3 mode register	PM3		-	o	-	
FF25H	Port 5 mode register	PM5		-	o	-	
FF26H	Port 6 mode register	PM6		-	o	-	F0H
FF27H	Port 7 mode register	PM7		-	o	-	FFH

(to be continued)

Table 3-6 Special Function Registers (SFRs) (Cont'd)

Address	Special function register (SFR) name	Abbreviation	R/W	Manipulation bit unit			At resetting
				1	8	16	
FF30H	16-bit timer register 0	TM0	R	-	-	o	Undefined for 16 clock pulses Cleared to 0 after the 17th clock pulse
FF31H				-	-		
FF32H	16-bit timer register 1	TM1		-	-	o	
FF33H				-	-		
FF34H	16-bit free running counter	FRC		-	-	o	
FF35H				-	-		
FF36H	16-bit timer register 2	TM2		-	-	o	
FF37H				-	-		
FF38H	Timer control register 0	TMC0	W	-	o	-	0xx00000
FF39H	Timer control register 1	TMC1	R/W	-	o		00H
FF3AH	Capture mode register	CPTM	W	-	o	-	xxxxx000
FF3DH	7-bit timer register 3	TM3	R	-	o	-	00H
FF3EH	7-bit timer 3 compare register	CR30	R/W	-	o	-	x1111111
FF3FH	7-bit timer 3 capture register	CPT30	R	-	o	-	Undefined
FF40H	Register for optional pull-up resistor	PUO	R/W	o	o	-	00H
FF43H	Port 3 mode control register	PMC3		o	o	-	30H
FF4AH	Port 0 buffer register	POL		o	o	-	Undefined

(to be continued)

Table 3-6 Special Function Registers (SFRs) (Cont'd)

Address	Special function register (SFR) name	Abbreviation	R/W	Manipulation bit unit			At resetting
				1	8	16	
FF4BH	Port 0 buffer register	P0H	R/W	o	o	-	Undefined
FF4CH	Real-time output port control register	RTPC		o	o	-	00H
FF50H	Input control register	ICR	W	-	o	-	0x0x0xxx
FF53H	Event divider control register	EDVC		-	o	-	Undefined
FF54H	Event counter compare register 1	ECC1		-	o	-	xx111111
FF55H	Event counter compare register 0	ECC0		-	o	-	xx111111
FF56H	Event counter	EC	R	-	o	-	xx000000
FF58H	Timer 0 output mode register	TOM0	W	-	o	-	xx000000
FF59H	Timer 0 output control register	TOC0		-	o	-	xx000000
FF5AH	Timer 1 output mode register	TOM1		-	o	-	xxxx0000
FF5BH	Timer 1 output control register	TOC1	R/W (*)	-	o	-	xxxx0000
FF68H	A/D conversion mode register	ADM	R/W	o	o	-	00H
FF6AH	A/D conversion result register	ADCR	R	-	o	-	Undefined
FF70H	PWM control register	PWMC	R/W	o	o	-	05H

(to be continued)

* Only bit 0 of TOC1 can be read.

Table 3-6 Special Function Registers (SFRs) (Cont'd)

Address	Special function register (SFR) name	Abbreviation		R/W	Manipulation bit unit			At resetting
					1	8	16	
FF72H	PWM0 modulo register	PWM0		W	-	-	o	Undefined
FF73H					-	-		
FF74H	PWM1 modulo register	PWM1			-	-	o	
FF75H					-	-		
FF7FH	Clock output mode register	CLOM		R/W	o	o	-	00H
FF80H	Serial interface mode register	CSIM		R/W	o	o	-	00H
FF82H	Serial bus interface control register	SBIC		R/W	o	o	-	00H
FF86H	Serial shift register	SIO			-	o	-	Undefined
FFC0H	Standby control register	STBC			-	o	-	00H
FFC4H	Memory mapping register	MM		W	-	o	-	20H
FFCFH	Internal memory size switch register(*)	IMS			-	o	-	FDH
FFE0H	Interrupt request flag register	IF0L	IF0	R/W	o	o	o	00H
FFE1H		IF0H			o	o		00H
FFE4H	Interrupt mask register	MK0L	MK0		o	o	o	FFH
FFE5H		MK0H			o	o		FFH

(to be continued)

* The internal memory size switch register (IMS) is included only in the uPD78P138. It is not included in the uPD78134A, uPD78136, or uPD78138. So in the uPD78134A, uPD78136, and uPD78138, address FFCFH must not be accessed.

Table 3-6 Special Function Registers (SFRs) (Cont'd)

Address	Special function register (SFR) name	Abbreviation		R/W	Manipulation bit unit			At resetting
					1	8	16	
FFE8H	Priority specification flag register	PR0L	PR0	R/W	o	o	o	FFH
FFE9H		PR0H			o	o		FFH
FFECH	Interrupt service mode register	ISM0L	ISM0		o	o	o	00H
FFEDH		ISM0H			o	o		00H
FFF4H	External interrupt mode register	INTM0			o	o	-	50H
FFF5H	External capture input mode register	INTM1			o	o		0000xx01

o: Allowed
-: Not allowed

3.3 External Expansion Functions

The uPD78138 allows up to 64K bytes of program memory, data memory, or I/O to added externally.

3.3.1 Bus interface function

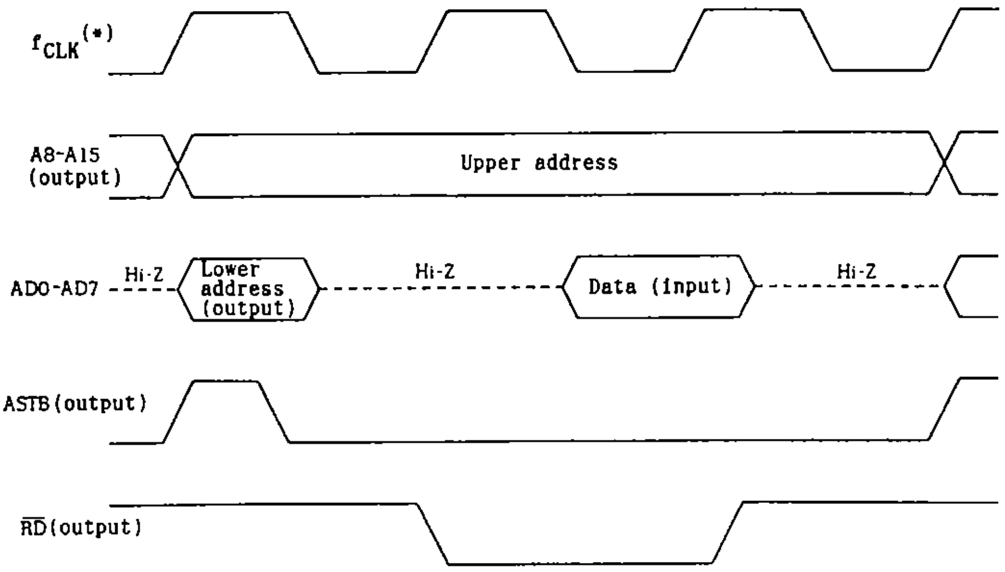
The external expansion is made by using the bus interface functions such as address and data buses (AD0 to AD7, A8 to A15), and read, write and address strobe signals ($\overline{RD}$, $\overline{WR}$, and ASTB). Figures 3-11 and 3-12 show the basic bus interface timing.

Except ASTB, the bus interface pins are also used as port pins. The lower address part/data bus pins (AD0 to AD7) also function as port 4 (P40 to P47), the address upper part bus pins (A8 to A15) function as port 5 (P50 to P57), and the $\overline{RD}$ and $\overline{WR}$ pins function as port 6 (P64 and P65). These pins function as the bus interface by specifying an external expansion mode in the memory mapping register (MM) and external access pin ($\overline{EA}$). Table 3-7 lists the external expansion modes of the uPD78138 and the pin functions.

Table 3-7 External Expansion Modes and Pin Functions

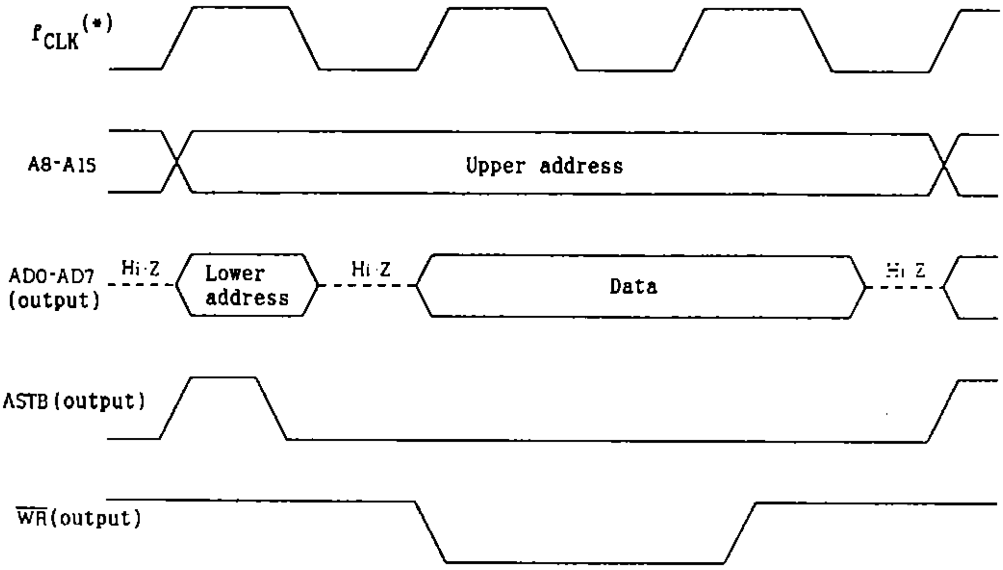
External expansion mode	P40-P47	P50-P57	P64	P65
Single-chip mode	Port 4	Port 5	Port 6	
256-byte expansion mode	AD0-AD7		$\overline{RD}$	$\overline{WR}$
48K-byte expansion mode		A8-A15		
64K-byte expansion mode (ROM-less mode)				

Fig. 3-11 Read Timing



* f_{CLK}: System clock frequency

Fig. 3-12 Write Timing



* f_{CLK}: System clock frequency

Caution: An external device cannot be mapped in the internal RAM area (FC80H to FEFFH) and the SFR area (FF00H to FFFFH) so that they overlap.

If the overlapped space is manipulated, and internal RAM and SFRs are automatically subject to manipulation. Although an address signal and the ASTB signal are output, the $\overline{RD}$ and $\overline{WR}$ signals are not output. That is, these signal lines remain high.

3.3.2 Memory mapping register (MM)

The memory mapping register (MM) is an 8-bit register to control the external expansion function. The register specifies the bus interface function, the number of waits, and the instruction fetch cycle. Figure 3-13 is the format of the memory mapping register.

Fig. 3-13 Format of Memory Mapping Register (MM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
MM	IFCH	0	PW21	PW20	0	MM2	MM1	MM0	FFC4H	20H	W

$\overline{\text{EA}}$	MM2	MM1	MM0	Mode		P40-P47		P50-P57	P65	P64
1	0	0	0	Single-chip mode		Port mode	Input	Port mode (*1)	Port mode (*1)	
	0	0	1				Output			
	0	1	1	External expansion mode	256-byte expansion	AD0-AD7			$\overline{\text{WR}}$	$\overline{\text{RD}}$
	1	1	1		(*2)			A8-A15		
0	x	x	x		64K-byte expansion					

$\overline{\text{EA}}$: External access pin

PW21	PW20	Specification of the number of waits for external memory access
0	0	0
0	1	1
1	0	2
1	1	Not to be set

IFCH	Internal fetch cycle control
0	Same instruction execution cycle as external ROM fetch cycle
1	High-speed internal fetch operation (Instruction is executed faster than external ROM fetch.)

*1 I/O is specified by port mode registers.
*2 32K-byte expansion (uPD78138 and uPD78P138)
40K-byte expansion (uPD78136)
48K-byte expansion (uPD78134A)

Cautions 1. When a reset occurs, IFCH (bit 7 of MM) is set to 0, and CPU processing becomes slower than internal fetch operation. For access to internal ROM, set IFCH to 1 to speed up CPU processing.

2. After the reset state is released, the IFCH bit can be set only once. If the setting of this bit is changed more than once, the system may malfunction.

3. Be sure to write 0 in MM bit 6.

3.3.3 Memory map in external expansion

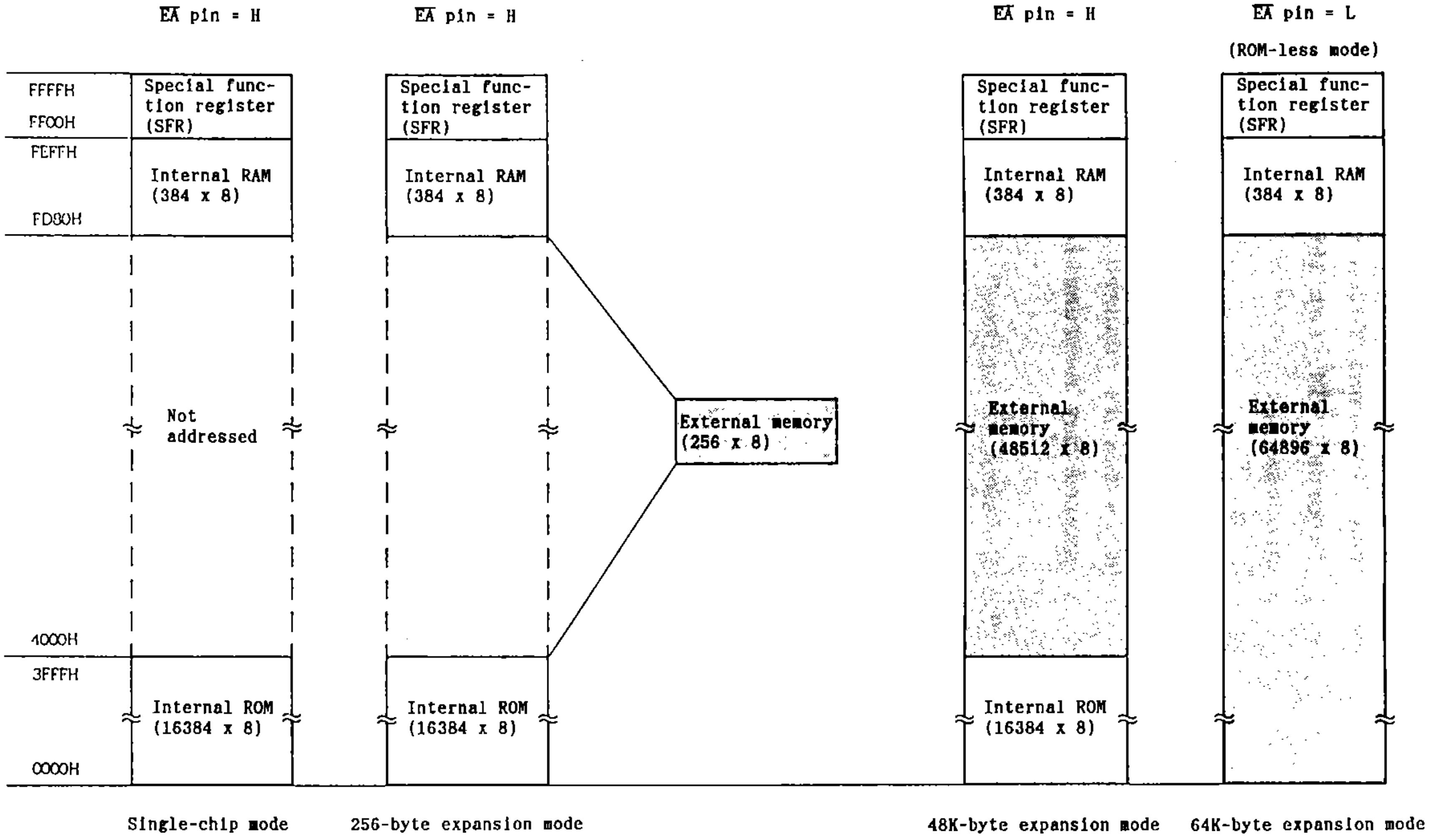
Figures 3-14 to 3-16 are memory maps when external expansion is done.

When an external area is addressed in the 256-byte expansion mode, the lower eight bits of the address (A0 to A8) are output. Thus, 256 bytes in the external area can be accessed.

In the 64K-byte expansion mode (ROM-less mode), an internal ROM area is not accessed.

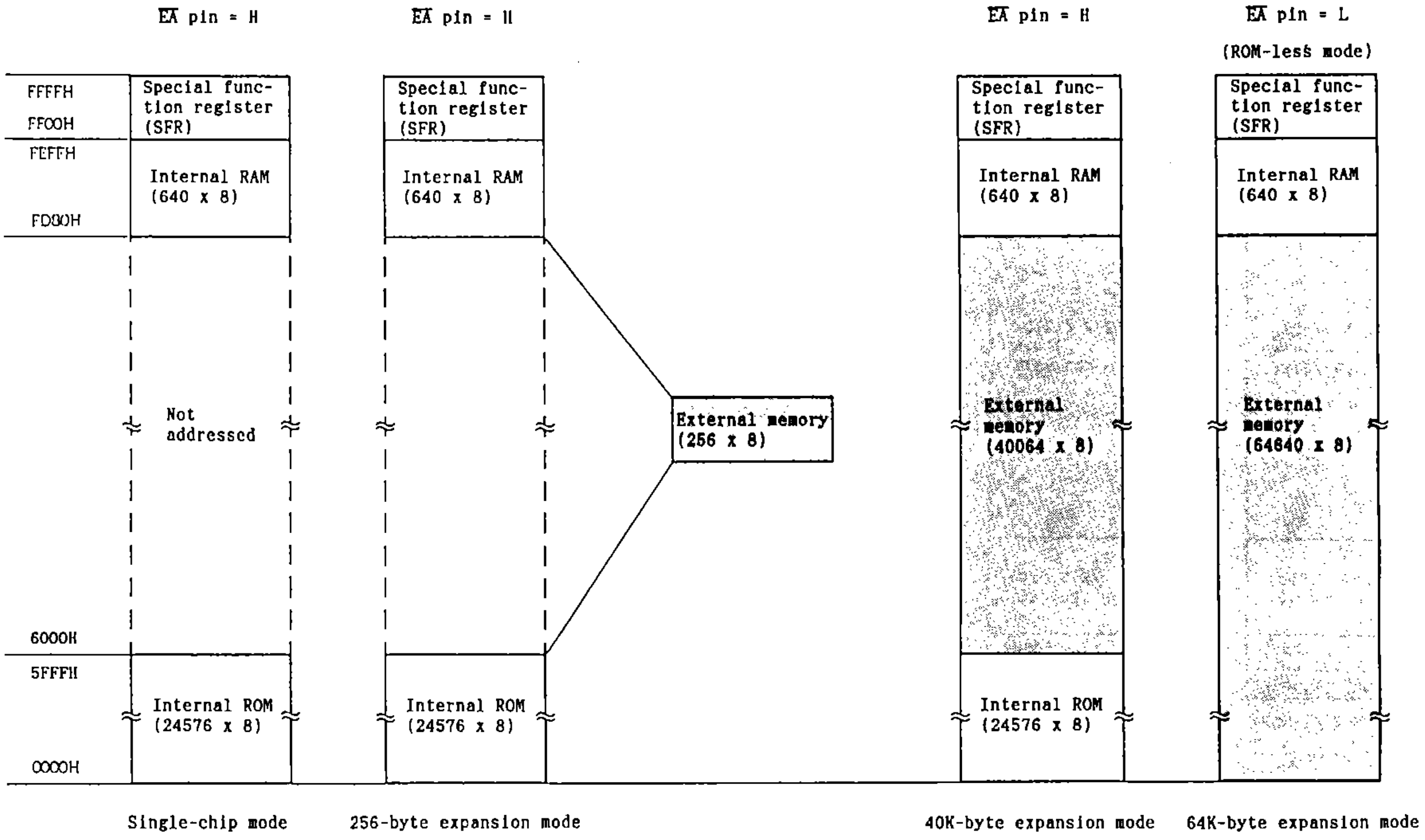
Data in internal RAM and special function registers (SFRs) is not fetched as instructions. This means that a program cannot be stored in these areas. Programs must be stored in an area from 0000H to FD7FH for the uPD78134A and from 0000H to FC7FH for the uPD78136, uPD78138, and uPD78P138.

Fig. 3-14 Memory Map in External Expansion (uPD78134A)



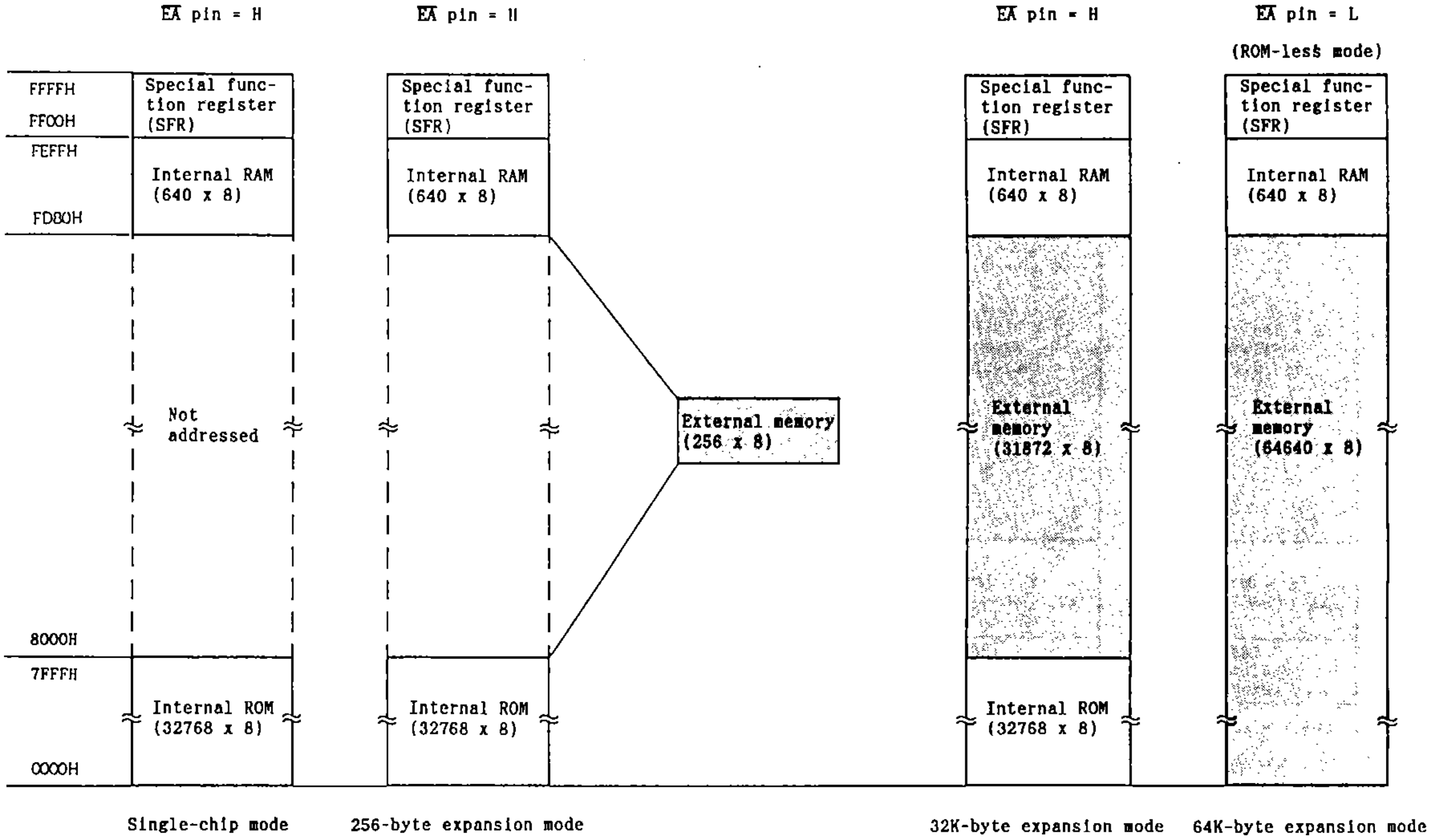
Remark: ▨ External expansion area

Fig. 3-15 Memory Map in External Expansion (uPD78136)



Remark: □ External expansion area

Fig. 3-16 Memory Map in External Expansion (uPD78138 and uPD78P138)

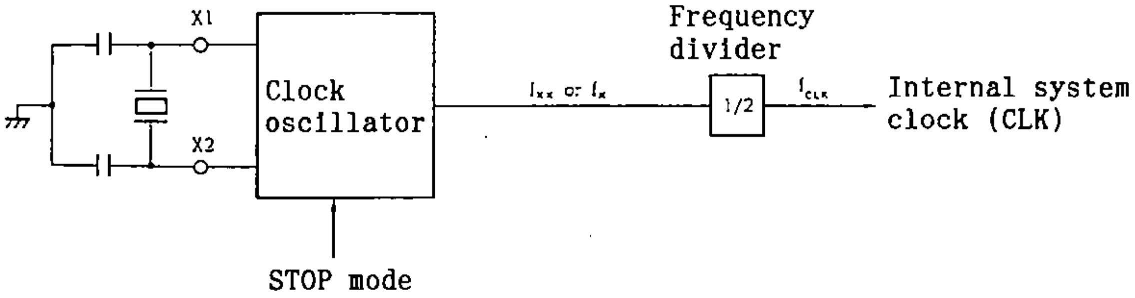


Remark: ▨ External expansion area

4.1 Configuration and Functions

The clock generator generates and controls an internal system clock signal (CLK) supplied to the CPU. Figure 4-1 is the configuration of the clock generator.

Fig. 4-1 Block Diagram of the Clock Generator



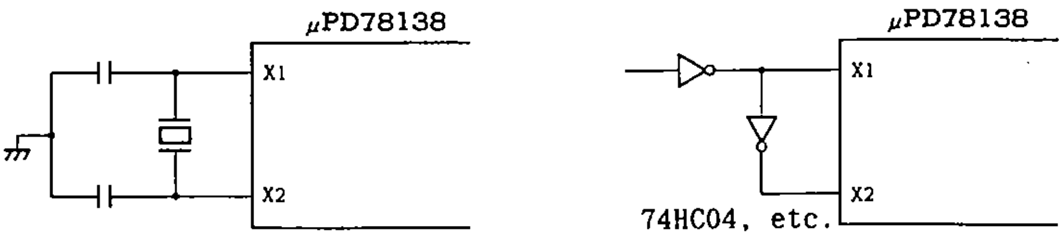
Remarks: f_{XX}: Crystal or ceramic oscillator frequency
f_X: External clock frequency
f_{CLK}: Internal system clock frequency
(1/2 f_{XX} or 1/2 f_X)

The clock oscillator oscillates a clock signal with a crystal or ceramic resonator connected to the X1 and X2 pins. When the standby mode (STOP) is set, the clock oscillator stops oscillation. (See Chapter 12.)

The clock oscillator can also accept external clock signal input. To do this, apply the clock signal to the X1 pin, and apply the inverted signal to the X2 pin.

The frequency divider divides the clock oscillator output (f_{XX} when a crystal or ceramic resonator is used, or f_X when an external clock is used) by two to produce an internal system clock signal (CLK).

Fig. 4-2 External Circuitry of the Clock Oscillator



(a) With a crystal or ceramic resonator

(b) With an external clock

Remark: Choosing between a crystal resonator and a ceramic oscillator

In general, crystal resonators produce stable frequencies. So crystal resonators are suitable for high-precision time control (for example, for clock or frequency measurement use). Ceramic resonators produce less stable frequencies than crystal resonators. But ceramic resonators start oscillation in a shorter time, are more compact, and are less costly. So ceramic resonators are useful for normal applications (requiring less precise time control). In addition, ceramic resonators containing a capacitor are available for a reduced part count and mounting space.

4.2 Cautions

The user must exercise caution with the clock generator as described below.

4.2.1 When an external clock is applied

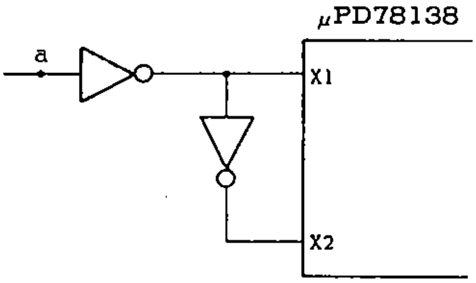
(1) When applying an external clock, never use the STOP mode. Otherwise, a destruction may occur, or the reliability may deteriorate.

(2) When applying an external clock, apply, to the X2 pin, the inverted signal of the signal applied to the X1 pin. Otherwise, malfunctioning due to noise may occur more frequently.

(3) When applying an external clock, use an HCMOS or a device having an equivalent drive capability.

(4) Never extract signals from the X1 or X2 pin. When X1 and X2 signal output is required, use point a in Figure 4-3.

Fig. 4-3 Signal Extraction Point in External Clock Input



(5) Minimize the wiring from the X1 pin to the X2 pin through the inverter.

4.2.2 When crystal/ceramic oscillation is used

(1) Since the generator is a high-frequency analog circuit, the user must be careful in handling. In particular, the user must observe the following:

. Minimize the wiring.

. Never cause the wires to cross other signal lines or run near a line over which a high current flows.

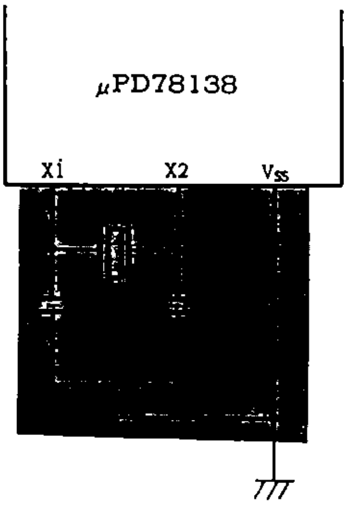
. Ground the capacitor of the generator so that the grounding point is always at the same potential as the V_{SS} pin. Never connect the capacitor to a ground pattern carrying a high current.

. Never extract a signal from the generator circuit.

The subsystem clock generator is a low-amplification circuit for reduced current consumption. This means that the subsystem clock generator is more sensitive to noise than the main system clock generator. When the subsystem clock circuitry is used, it must be wired carefully.

If normal and stable oscillation is not provided, the microcomputer cannot operate normally in a stable manner. When the user needs a high-precision oscillator frequency, it is recommended that the user consult with an oscillator supplier.

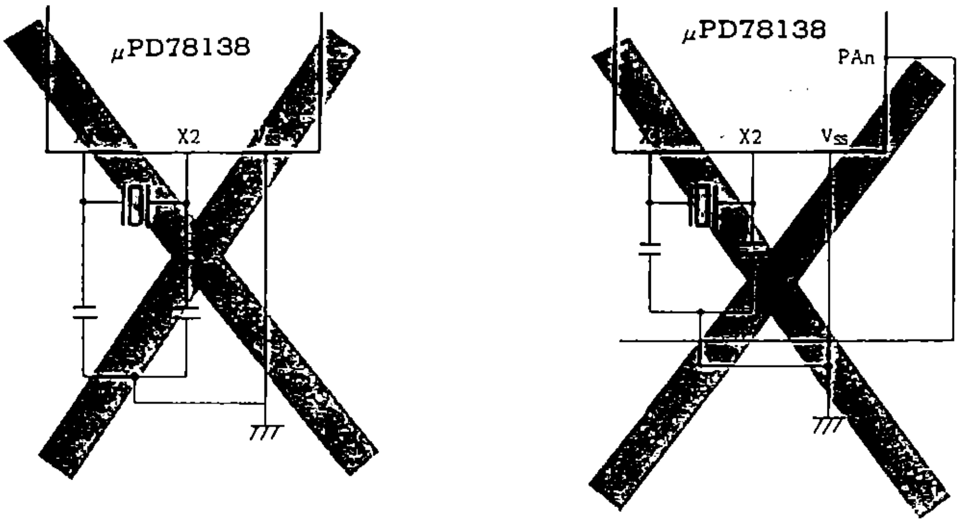
Fig. 4-4 Cautions for Resonator Connection Circuitry



Cautions 1. Place the oscillator circuit as close as possible to the X1 and X2 pins.

2. Never run other signal lines in the shaded area (■).

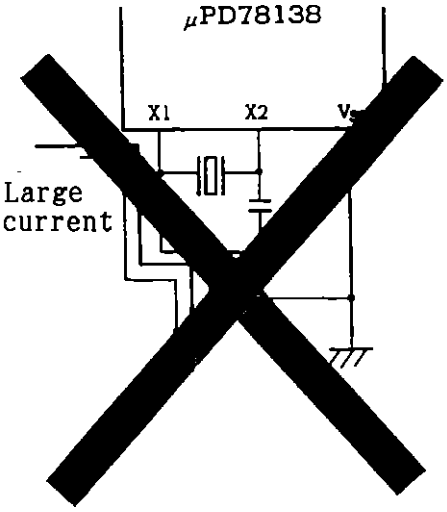
Fig. 4-5 Examples of Wrong Resonator Connection Circuitry



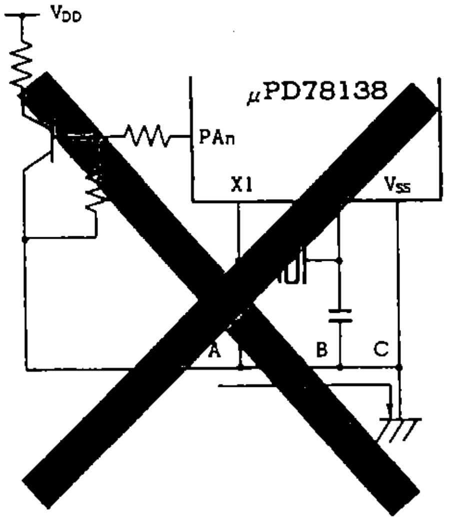
(a) Connection circuit wiring is too long.

(b) There is another signal line crossing.

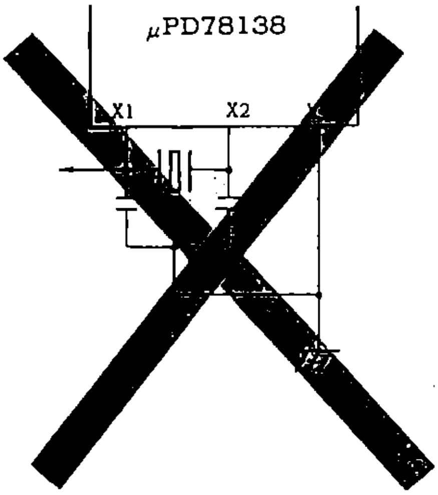
(c) A high varying current flows near a signal line.



(d) A current flows over the ground line of the generator circuit. (The potentials of points A, B, and C change.)



(e) A signal is extracted.



(2) At the time of power-on or return from the STOP mode, some waiting time is required before a stable oscillation can be obtained. In general, when a crystal resonator is used, several milliseconds are required. When a ceramic resonator is used, several hundreds of microseconds are required.

The two factors described below determine a time required for stable oscillation. Allow a sufficient time.

① $\overline{\text{RESET}}$ input at power-on (reset period)

② $\overline{\text{RESET}}$ input (reset period), (time period during which the NMI signal is active + automatically used timer), or automatically used timer at return from the STOP mode

CHAPTER 5 PORT FUNCTIONS

5.1 Functions and Outline of the Ports

The uPD78138 is provided with the ports shown in Figure 5-1, which allow a wide variety of control capabilities.

Table 5-1 indicates the functions of the ports. The use of internal pull-up resistors can be specified by software.

Fig. 5-1 Port Configuration

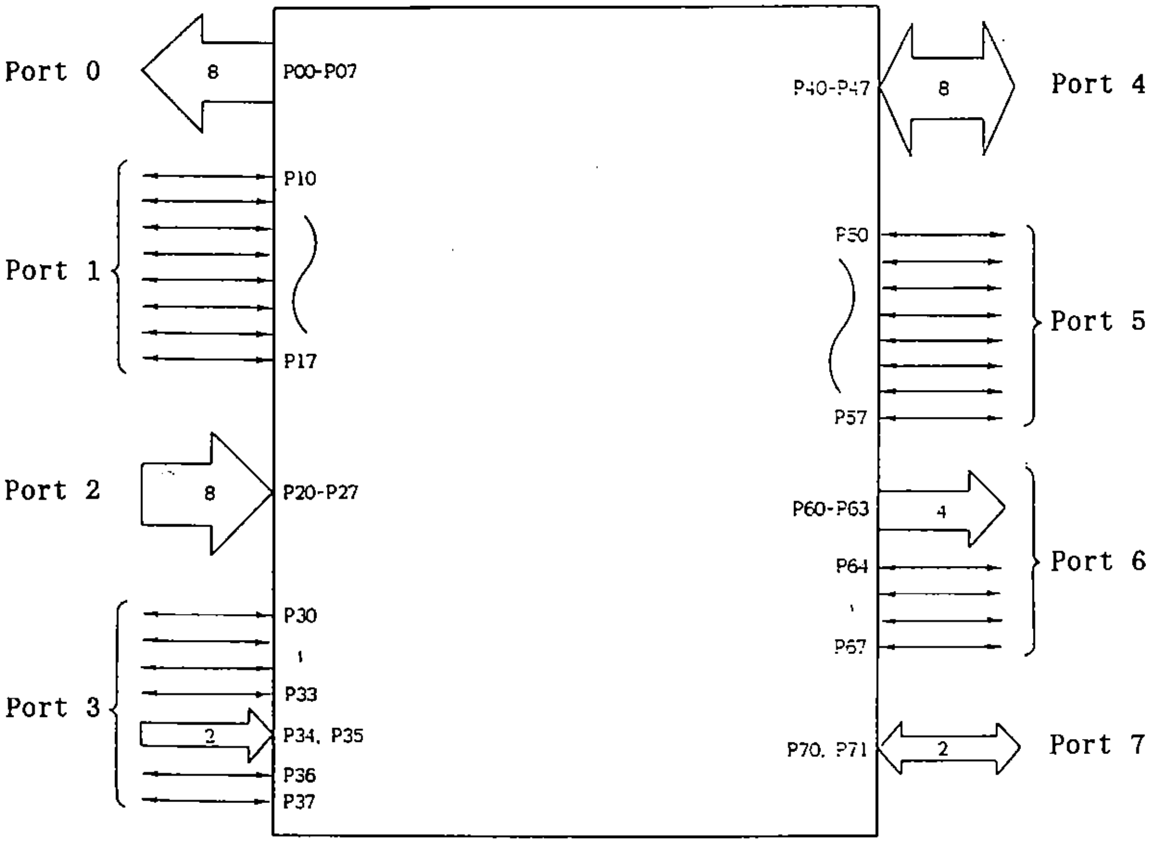


Table 5-1 Port Functions

Port	Pin	I/O	Function
Port 0	P00-P07	Output	8-bit output port . Specifiable in units of 8 bits for output or high-impedance . Also functions as a real-time output port of one 8-bit channel or two 4-bit channels.
Port 1	P10-P17	I/O	8-bit I/O port . Specifiable for input or output bit by bit. . Can directly drive an LED. . The use of the pull-up resistors can be specified by software for the pins in the input mode at one time.
Port 2	P20, P21	Input	8-bit input port . Also functions as external interrupt pins and trigger pins. . The use of the pull-up resistors can be specified by software for pins P22 to P27 (six pins) at one time.
	P22-P27		
Port 3	P30-P33	I/O	8-bit I/O port . Also functions as control pins. . Allows input or output to be specified bit by bit for P30-P33, P36, and P37 . The use of the pull-up resistors can be specified by software for the pins in the input mode at one time.
	P34, P35	Input	
	P36, P37	I/O	
Port 4	P40-P47	I/O	8-bit I/O port . Also functions as a time-multiplexing address /data bus for external expansion. . Specifiable for input or output in units of 8 bits. . The use of the pull-up resistors can be specified by software for the eight pins at one time.
Port 5	P50-P57	I/O	8-bit I/O port . Address bus for external expansion . Specifiable for input or output bit by bit . The use of the pull-up resistors can be specified by software for the pins in the input mode at one time.

(to be continued)

Table 5-1 Port Functions (Cont'd)

Port	Pin	I/O	Function
Port 6	P60-P63	Output	8-bit I/O port . Also functions as $\overline{RD}$ and $\overline{WR}$, control signal pins for external expansion. . Allows input or output to be specified bit by bit for P64-P67. . The use of the pull-up resistors can be specified by software for the pins in the input mode at one time.
	P64-P67	I/O	
Port 7	P70, P71	I/O	2-bit I/O port . Allows input or output to be specified in units of 2 bits. . The use of the pull-up resistors can be specified by software for the two pins at one time.

Caution: Port 4 can contain a pull-up resistor both in the input and output modes. Port 3 can contain a pull-up resistor in the input and control modes. The other ports can contain a pull-up resistor only in the input mode.

5.2 Port 0 (P00-P07)

Port 0 is an 8-bit output port. The port turns off the output buffer and places it in the high-impedance state. Port 0 also functions as a real-time output port. (See Chapter 6.)

Table 5-2 shows the function of port 0.

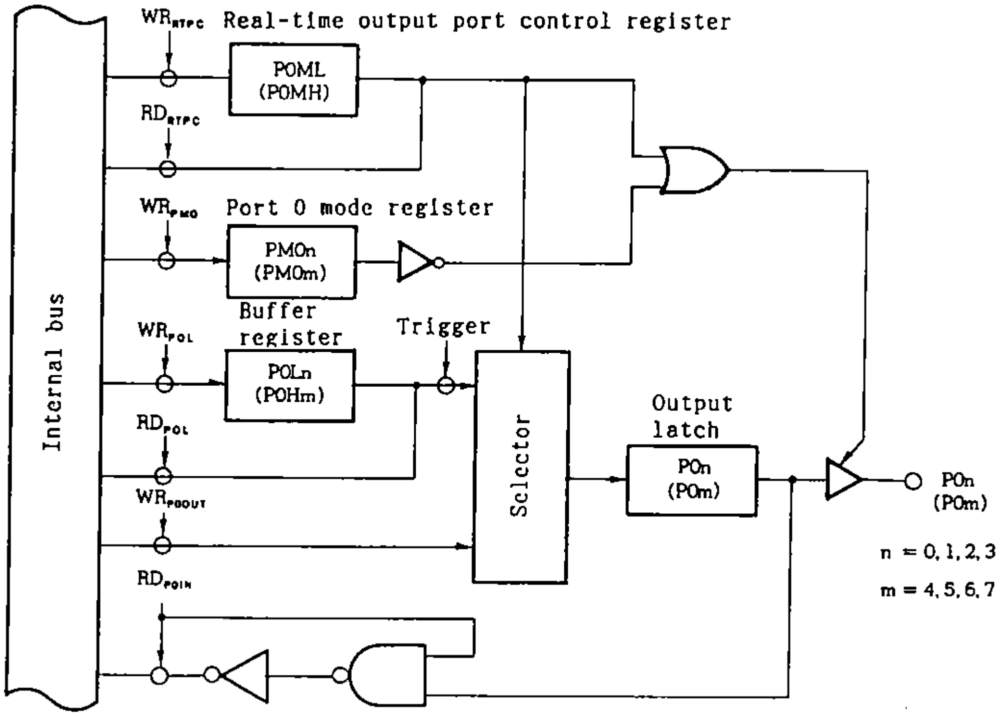
Table 5-2 Function of Port 0

Pin	Port function	Also usable as	Non-port function
P00 to P07	Output	Real-time output port 0	Also functions as an 8-bit real-time output port or two 4-bit real-time output port.

5.2.1 Hardware configuration

Figure 5-2 shows the hardware configuration of port 0.

Fig. 5-2 Block Diagram of Port 0



5.2.2 Setting the I/O mode and/or control mode

The I/O mode of port 0 is set by the port 0 mode register (PM0) as shown in Figure 5-3. PM0 is set with an 8-bit data transfer instruction. So PM0 cannot be manipulated or read on a bit-by-bit basis.

Fig. 5-3 Format of the Port 0 Mode Register

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	FF20H	FFH	W

PM0	P0n pin I/O specification (n = 0-7)
00H	Output mode (output buffer on)
FFH	High-impedance (output buffer off)
Other than above	Not to be set

When port 0 is to be used as a real-time output port, bits P0ML and P0MH of the real-time output port control register (RTPC) must be set to 1.

Setting P0ML and P0MH turns on the output buffer of each pin, regardless of the content of PM0, allowing the buffer content to be output on the pin.

RTPC is an 8-bit register and is written to with 8-bit or bit manipulation instructions.

Fig. 5-4 Format of Real-time Output Port Control Register (RTPC)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
RTPC	BYTE	0	0	POMH	EXTR	0	0	POML	FF4CH	00H	R/W

POML	Specification of P0L function
0	Port mode
1	Real-time output port mode

EXTR	Data transfer from buffer register to output latch by INTP0
0	Disabled
1	Enabled BYTE=0: Transfers P0L only. BYTE=1: Transfers P0L and P0H.

POMH	Specification of P0H function
0	Port mode
1	Real-time output port mode

BYTE	Operation mode of real-time output port
0	4-bit separate real-time output port
1	8-bit real-time output port

Caution: Be sure to write a 0 in bits 1, 2, 5, and 6.

5.2.3 Internal pull-up resistors

Port 0 is not provided with internal pull-up resistors.

5.3 Port 1 (P10-P17)

Port 1 is an 8-bit I/O port. Each bit of the port can be specified independently as input or output.

Port 1 is capable of high-current driving, and so it is suitable for driving an LED.

Port 1 is provided with software-controlled pull-up resistors. The resistors can be used only in the input mode.

Table 5-3 shows the functions of port 1.

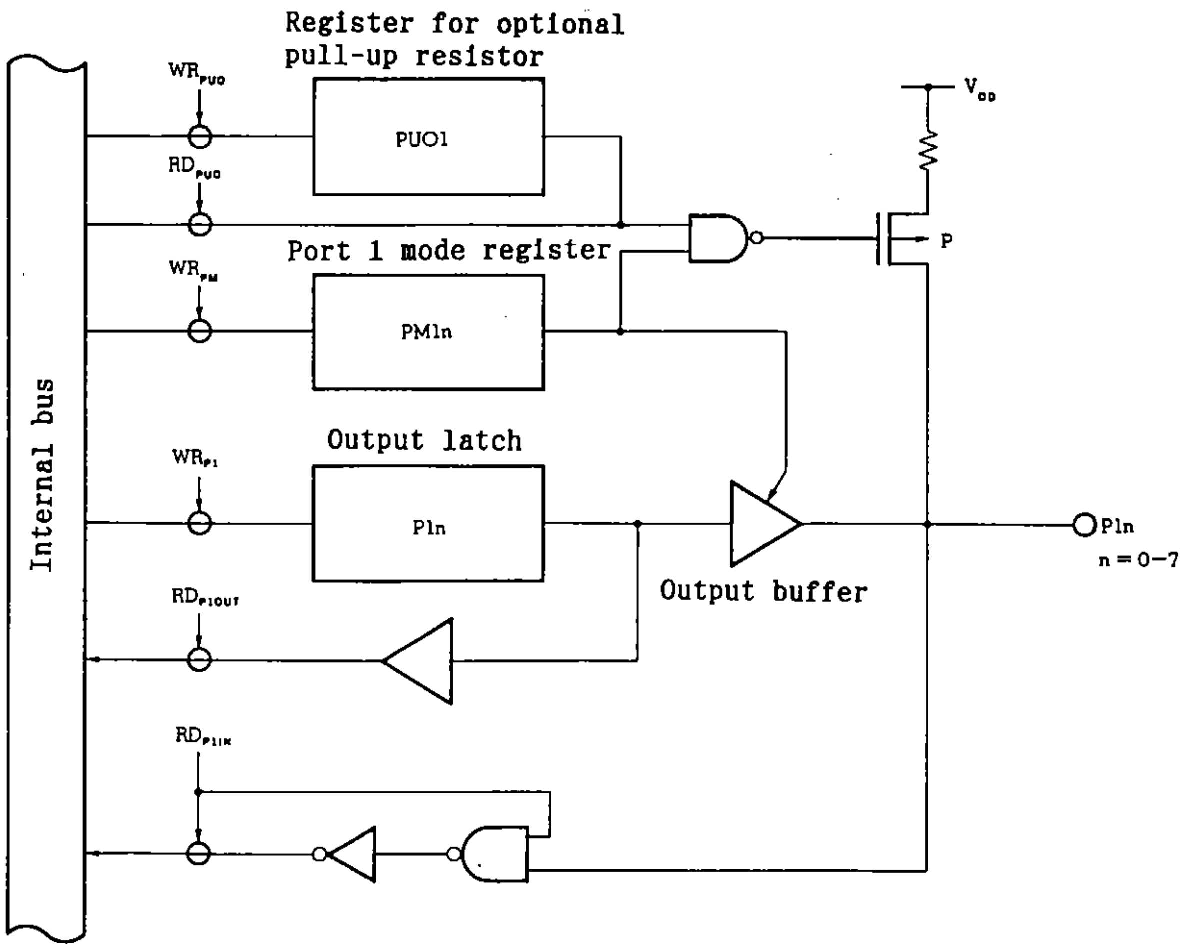
Table 5-3 Functions of Port 1

Pin	Port function	Also usable as	Non-port function
P10 to P17	I/O	-	-

5.3.1 Hardware configuration

Figure 5-5 shows the hardware configuration of port 1.

Fig. 5-5 Block Diagram of Port 1



5.3.2 Setting the I/O mode and/or control mode

The I/O mode is set for each pin of port 1 by port 1 mode register (PM1) as shown in Figure 5-6.

PM1 is set with an 8-bit data transfer instruction. So PM1 cannot be manipulated or read on a bit-by-bit basis.

Fig. 5-6 Format of the Port 1 Mode Register

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10	FF21H	FFH	W

PM1n	P1n pin I/O specification (n = 0-7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Caution: A bit manipulation instruction manipulates a bit, but it accesses a port in units of 8 bits. When a bit manipulation instruction is used for a port including both input and output pins, the contents of the output latches for the pins that are in the input mode become undefined (excluding the manipulated pins). Take care particularly when there is a pin used by switching between input and output. This applies also when a port is manipulated with other instructions.

5.3.3 Internal pull-up resistor

Port 1 is provided with optional internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of pull-up resistors is specified with the following two resistors:

- Bit 1 (PU01) of the register for optional pull-up resistors (PUO)
- Port 1 mode register (PM1)

The use of a pull-up resistor can be specified for the pins that are placed in the input mode at one time.

If PU01 is set to 1, the pull-up resistors for the pins that are placed in the input mode by PM1 are used.

Fig. 5-7 Format of the Register for Optional Pull-up Resistors

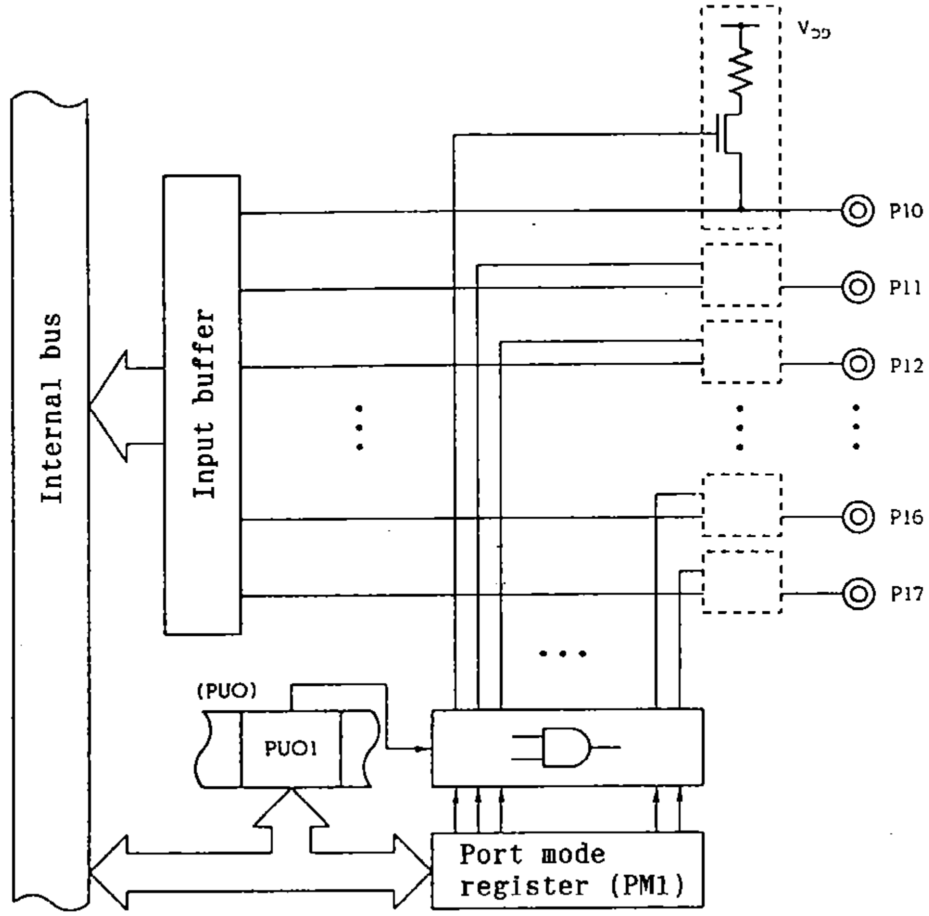
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU01	Use of pull-up resistors in port 1
0	Pull-up resistors are not used in port 1.
1	Pull-up resistors are used in port 1.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

If PU01 is 1, and if PU01 is associated with port 1, a pull-up resistor is made available to bits in the port which are specified as inputs by the port 1 mode register (PM1) for that port.

Fig. 5-8 Specification for Pulling Voltage in Port 1 High



5.4 Port 2 (P20-P27)

Port 2 is an 8-bit port for input only. The pins of the port also function as external interrupt and trigger pins. (See Chapters 8 and 11.) P22 to P27 can contain software-controlled pull-up resistors.

Table 5-4 lists the functions of port 2.

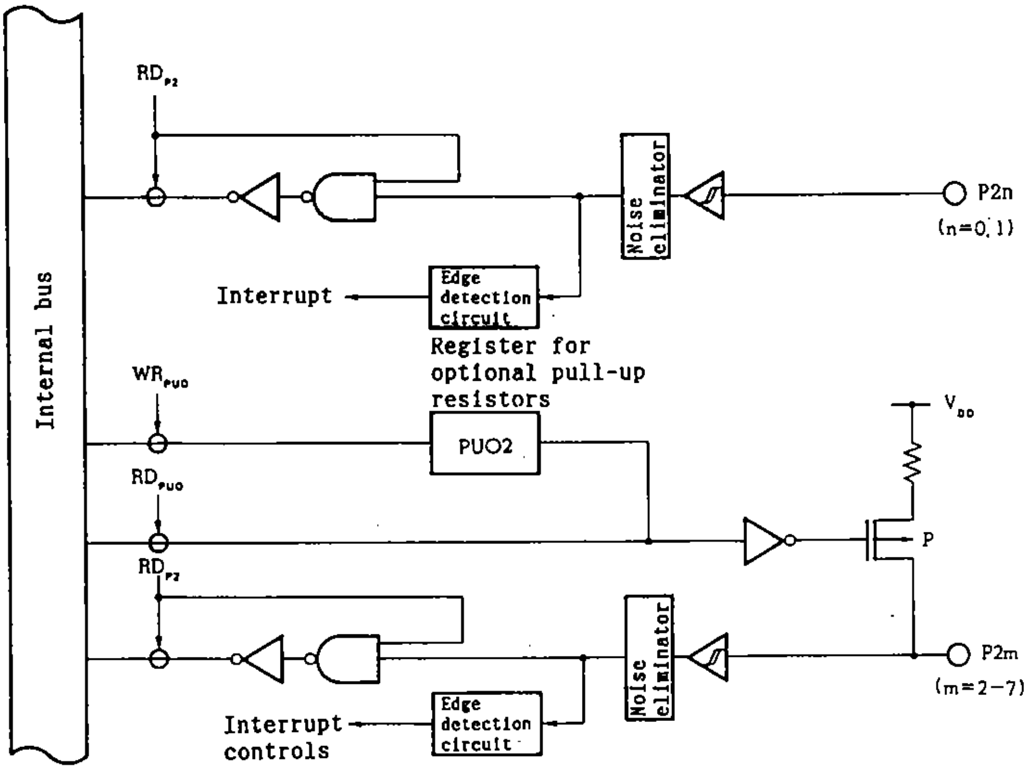
Table 5-4 Functions of Port 2

Pin	Port function	Also usable as	Non-port function
P20	Input	NMI	Nonmaskable interrupt request input pin
P21		INTP0	Cancels the external interrupt input/HALT mode.
P22		INTP1	Cancels the external interrupt input/HALT mode.
P23		INTP2	Cancels the external interrupt input/HALT mode.
P24		CTI10	FRC CPT3 capture trigger input
P25		CTI00	FRC CPT2 capture trigger input
P26		CTI11	TM1 CR12 capture trigger input
P27		CLR1	FRC CPT0 capture trigger input

5.4.1 Hardware configuration

Figure 5-9 shows the hardware configuration of port 2.

Fig. 5-9 Block Diagram of Port 2



5.4.2 Setting the I/O mode and/or control mode

Port 2 is an input port. So there is no register to set the input mode.

The port is always ready for control signal input.
The signal to be input must be determined with an internal control register in hardware components.

5.4.3 Internal pull-up resistors

Port 2 is provided with pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of the pull-up resistors for the six pins P22 to P27 is specified by PU02 of the register for optional pull-up resistors (PUO) at one time. (The resistors cannot be specified on a bit-by-bit basis.)

Note that P20 and P21 do not contain a pull-up resistor.

Fig. 5-10 Format of the Register for Optional Pull-up Resistors

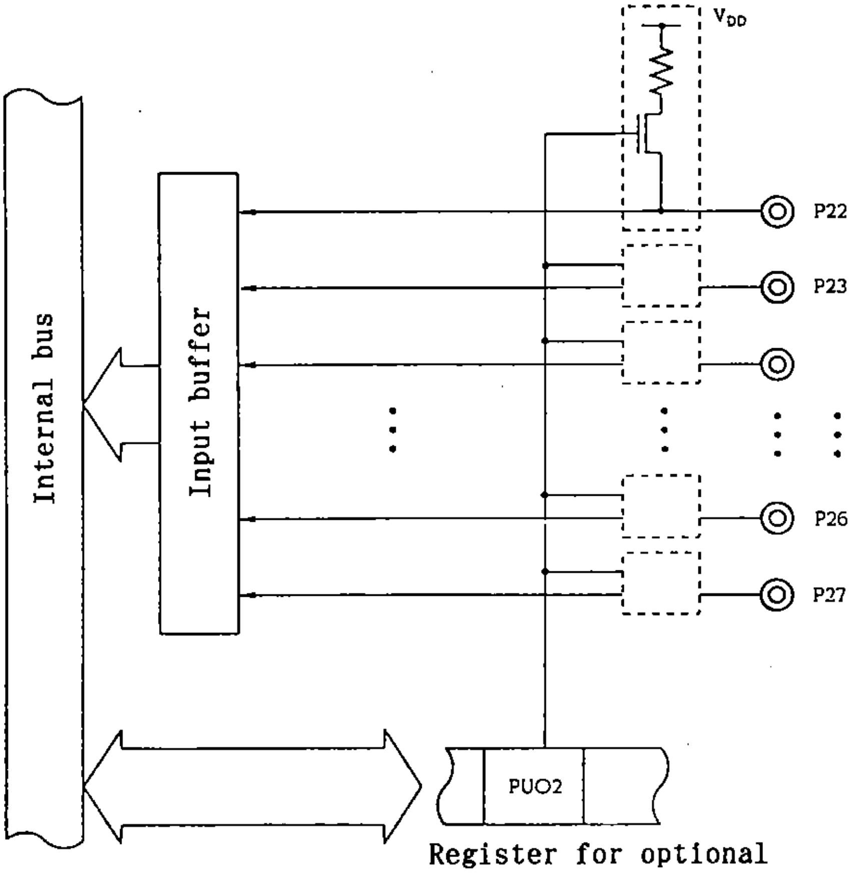
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU02	Use of pull-up resistors in port 2
0	Pull-up resistors are not used in port 2.
1	Pull-up resistors are used for P22-P27.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

Port 2 can be connected to V_{DD} with pull-up resistors just according to the specification in the PUO register.
The use of pull-up resistors can be set for all bits at a time.

Fig. 5-11 Specification for Pulling Voltage in Port 2 High



Register for optional pull-up resistors (PUO)

Cautions 1. A pull-up resistor is not provided for P20 and P21.

2. Because P22 to P27 are not pulled high immediately after reset, some dual functions of these pins may set associated interrupt request flags. So clear the interrupt request flags after the use of pull-up resistors is specified with an initialization routine.

5.5 Port 3 (P30-P37)

Port 3 is an 8-bit special I/O port. P30 to P33, P36, and P37 can be used as an I/O port bit by bit, and P34 and P35 can be used as a port for input only.

P30 to P33 also function as timer output pins, P34 functions as clear signal input for timer 0, and P35 to P37 function as the pins for serial interface 0.

Port 3 is provided with software-controlled pull-up resistors.

Table 5-5 lists the functions of port 3.

Table 5-5 Functions of Port 3

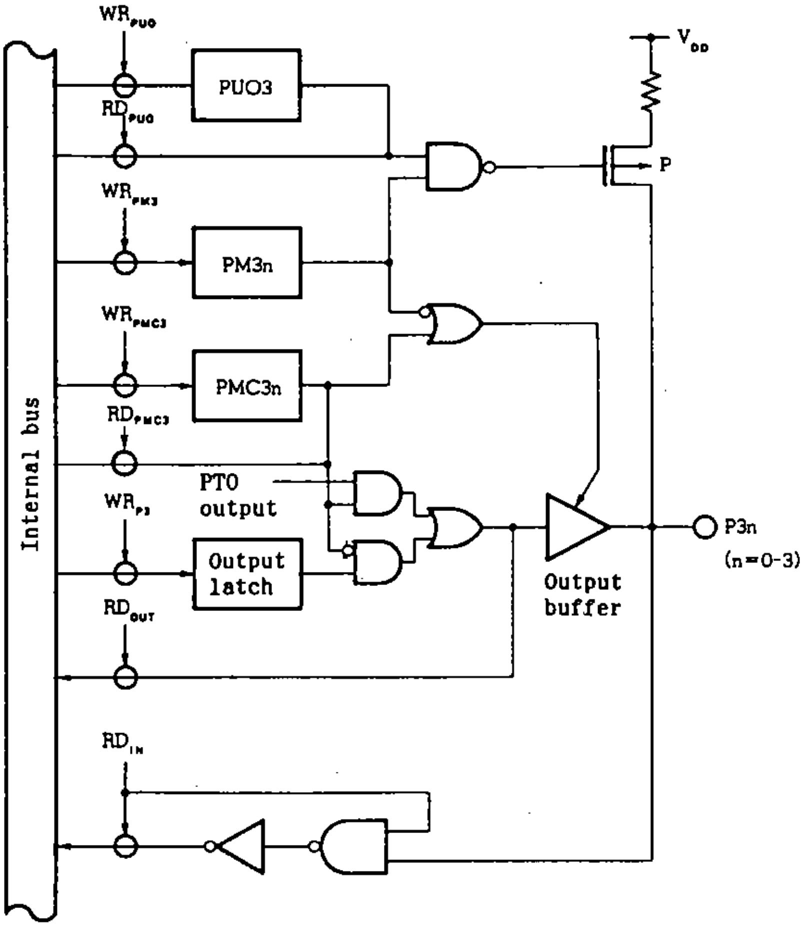
Pin	Port function	Also usable as	Non-port function
P30	I/O	PTO00	Super timer unit TM0 output pin (Matching of TM0 and CR00)
P31		PTO01	Super timer unit TM0 output pin (Matching of TM0 and CR01)
P32		PTO02	Super timer unit TM0 output pin (Matching of TM0 and CR02)
P33		PTO11	Super timer unit TM1 output pin (Matching of TM1 and CR11)
P34	Input	CLR0	Super timer unit TM0 clear input
P35		SI	Serial interface data input pin (3-wire serial I/O)
P36	I/O	SO/SB0	Serial interface data output pin (3-wire serial I/O)/serial interface data I/O pin (SBI)
P37		$\overline{\text{SCK}}$	Serial interface clock I/O pin

5.5.1 Hardware configuration

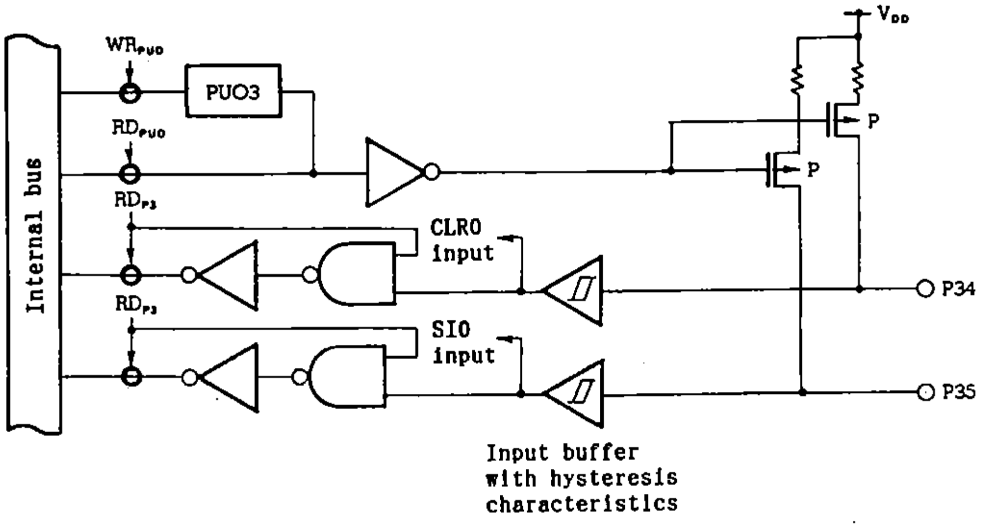
Figure 5-12 shows the hardware configuration of port 3.

Fig. 5-12 Block Diagram of Port 3

(a) P30 to P33

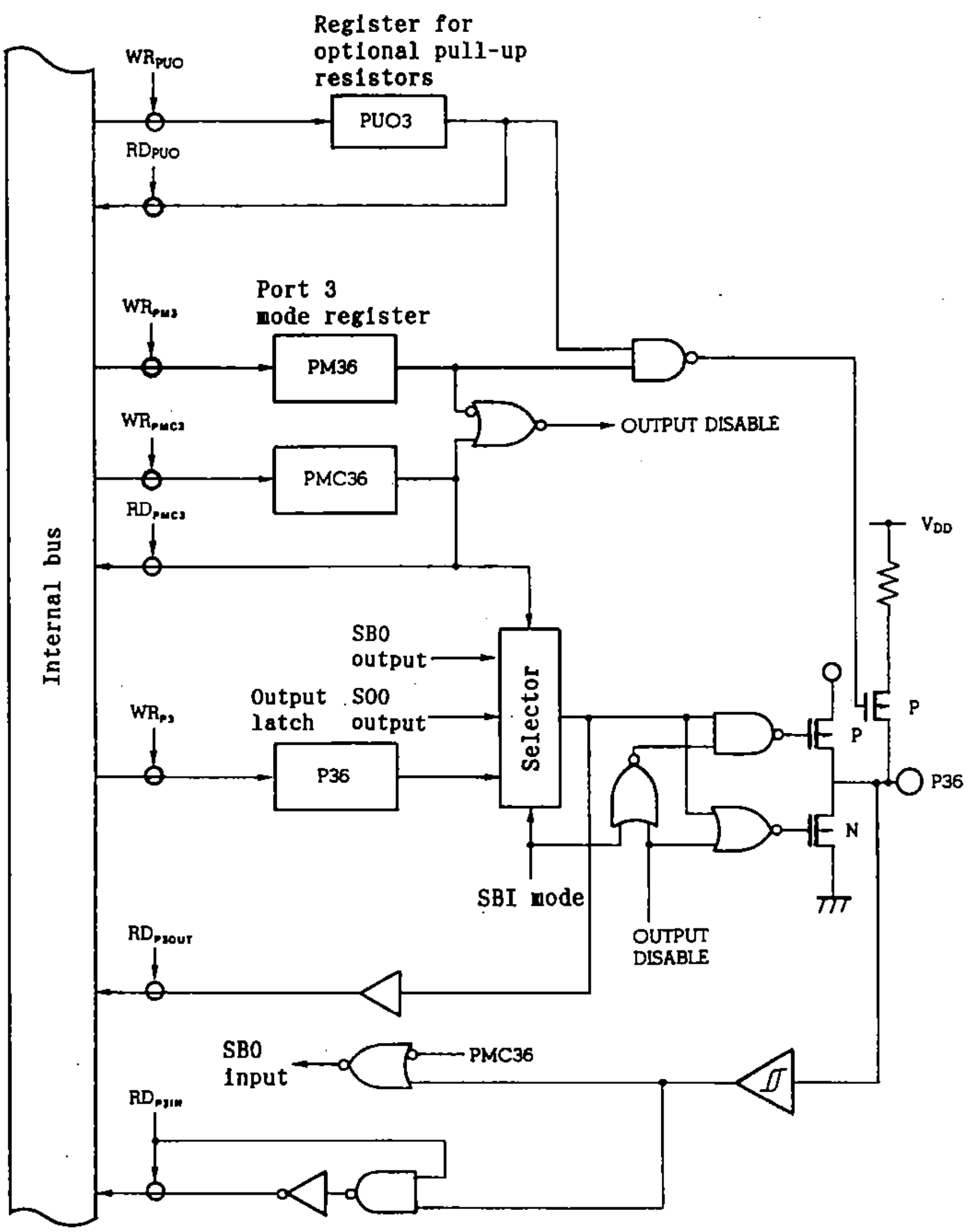


(b) P34, P35

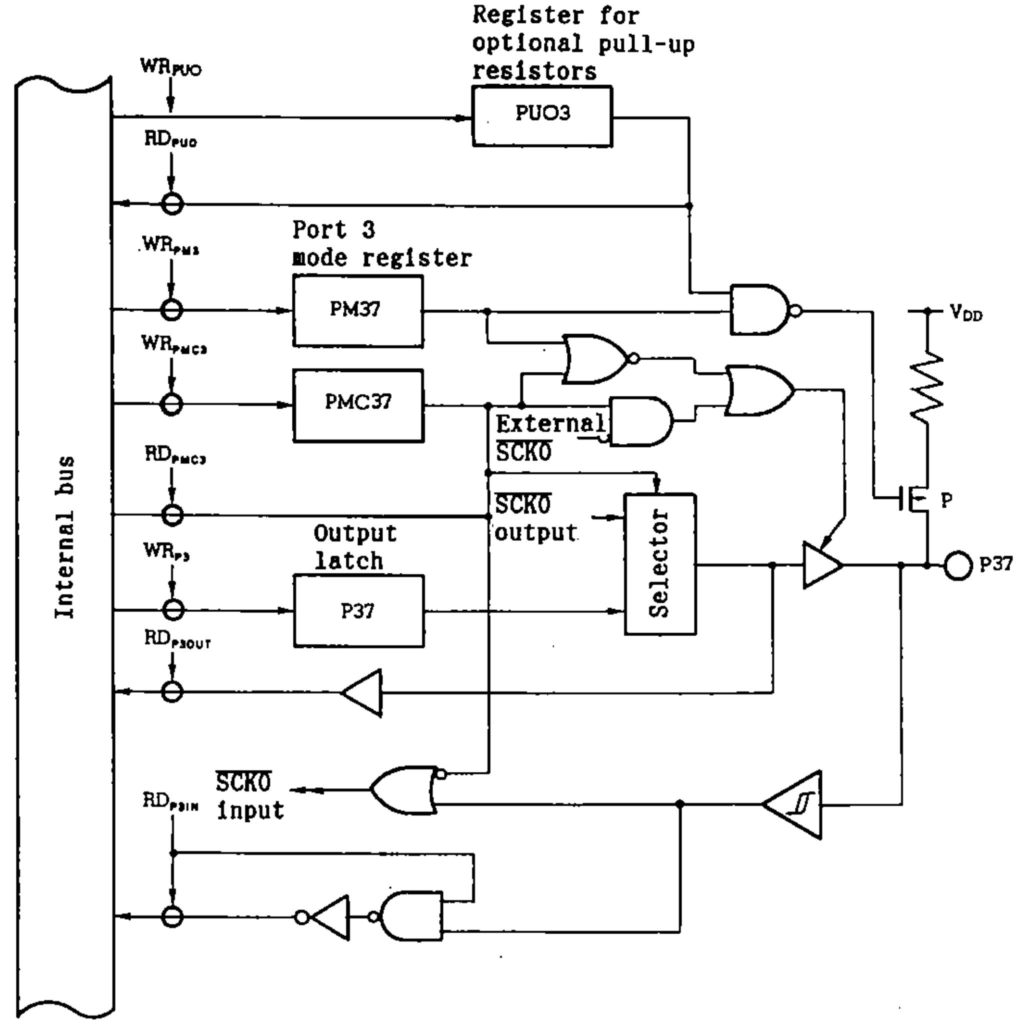

V_{DD}
WR_{PUO}
PUO3
RD_{PUO}
Internal bus
RD_{P3}
CLR0 input
P
P
P34
RD_{P3}
SIO input
P35
Input buffer with hysteresis characteristics

Phase-out/Discontinued

(c) P36



(d) P37


Register for optional pull-up resistors
PUO3
WR_{PUO}
RD_{PUO}
Port 3 mode register
WR_{PM3}
PM37
WR_{PMC3}
PMC37
External $\overline{SCK0}$
RD_{PMC3}
$\overline{SCK0}$ output
Selector
Output latch
WR_{P3}
P37
RD_{P3OUT}
Internal bus
V_{DD}
P
P37
$\overline{SCK0}$ input
RD_{P3IN}

5.5.2 Setting the I/O mode and/or control mode

The I/O mode can be set for each pin of port 3 with the port 3 mode register (PM3) as shown in Figure 5-13.

PM3 is set with an 8-bit data transfer instruction. (So PM3 cannot be manipulated or read on a bit-by-bit basis.)

Fig. 5-13 Format of the Port 3 Mode Register

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM3	PM37	PM36	1	1	PM33	PM32	PM31	PM30	FF23H	FFH	W

PM3n	P3n pin mode (n = 0-3, 6, 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

P30 to P33 also function as timer output pins, P34 functions as clear signal input for timer 0, and P35 to P37 function as the pins for serial interface 0.
The control modes of these dual-function pins are selected by the port 3 mode control register (PMC3).

Fig. 5-14 Format of the Port 3 Mode Control Register (PMC3)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PMC3	PMC37	PMC36	1	1	PMC33	PMC32	PMC31	PMC30	FF43H	30H	R/W

PMC3n	P3n pin control mode specification (n = 0-3)
0	I/O port mode
1	PTO output mode (output buffer on)

PMC36	P36 pin control mode specification
0	Input port mode
1	SO0/SB0 I/O mode

PMC37	P37 pin control mode specification
0	Input port mode
1	$\overline{SCK}$ output mode (Output buffer turned on by internal $\overline{SCK}$ specification)

Caution: A bit manipulation instruction manipulates a bit, but it accesses a port in units of 8 bits. When a bit manipulation instruction is used for a port including both input and output pins, the contents of the output latches for the pins that are in the input mode become undefined (excluding the manipulated pins). Take care particularly when there is a pin used by switching between input and output. This applies also when a port is manipulated with other instructions.

5.5.3 Internal pull-up resistors

Port 3 is provided with internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of pull-up resistors is specified with the following two resistors:

- Bit 3 (PUO3) of the register for optional pull-up resistors (PUO)
- Port 3 mode register (PM3)

The use of a pull-up resistor can be specified for the pins that are placed in the input mode at one time.

If PUO3 is set to 1, the pull-up resistors for the pins that are placed in the input mode by PM3 are used.

Even when the control mode is specified for a pin, the specification of a pull-up resistor is allowed for that pin. When a pin is in the control mode and it is not to be connected to a pull-up resistor, the corresponding bit of PM3 must be set to 0 (output mode).

Fig. 5-15 Format of the Register for Optional Pull-up Resistors

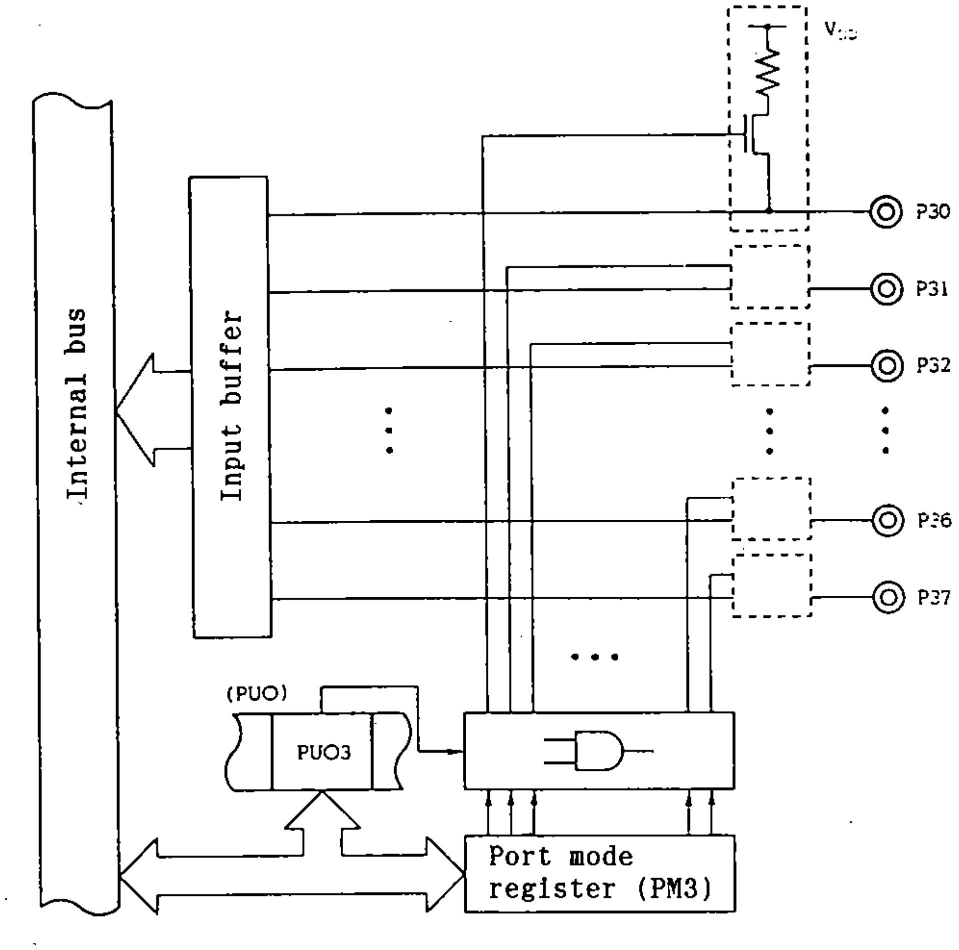
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PUOn	Use of pull-up resistors in port 3
0	Pull-up resistors are not used in port 3.
1	Pull-up resistors are used in port 3.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

If PUO3 is 1, and if PUO3 is associated with port 3, a pull-up resistor is made available to bits in the port which are specified as inputs by the port 3 mode register (PM3) for that port.

Fig. 5-16 Specification for Pulling Voltage in Port 3 High



Cautions 1. Even in the control mode, pull-up resistors are made available to port 3. To prevent the use of pull-up resistors for control inputs or outputs, set the associated bits in PM3 to 0 (output port).

2. In the port mode, P34 and P35 are always used in the input mode, so specifying the use of pull-up resistors with the PUO register makes the pull-up registers for P34 and P35 available.

5.6 Port 4 (P40-P47)

Port 4 is an 8-bit I/O port. Input or output can be specified in 8-bit units.

This port also functions as the address/data bus (AD0 to AD7) when external memory or I/O is connected.

Port 4 is provided with software-controlled pull-up resistors.

Table 5-6 lists the functions of port 4.

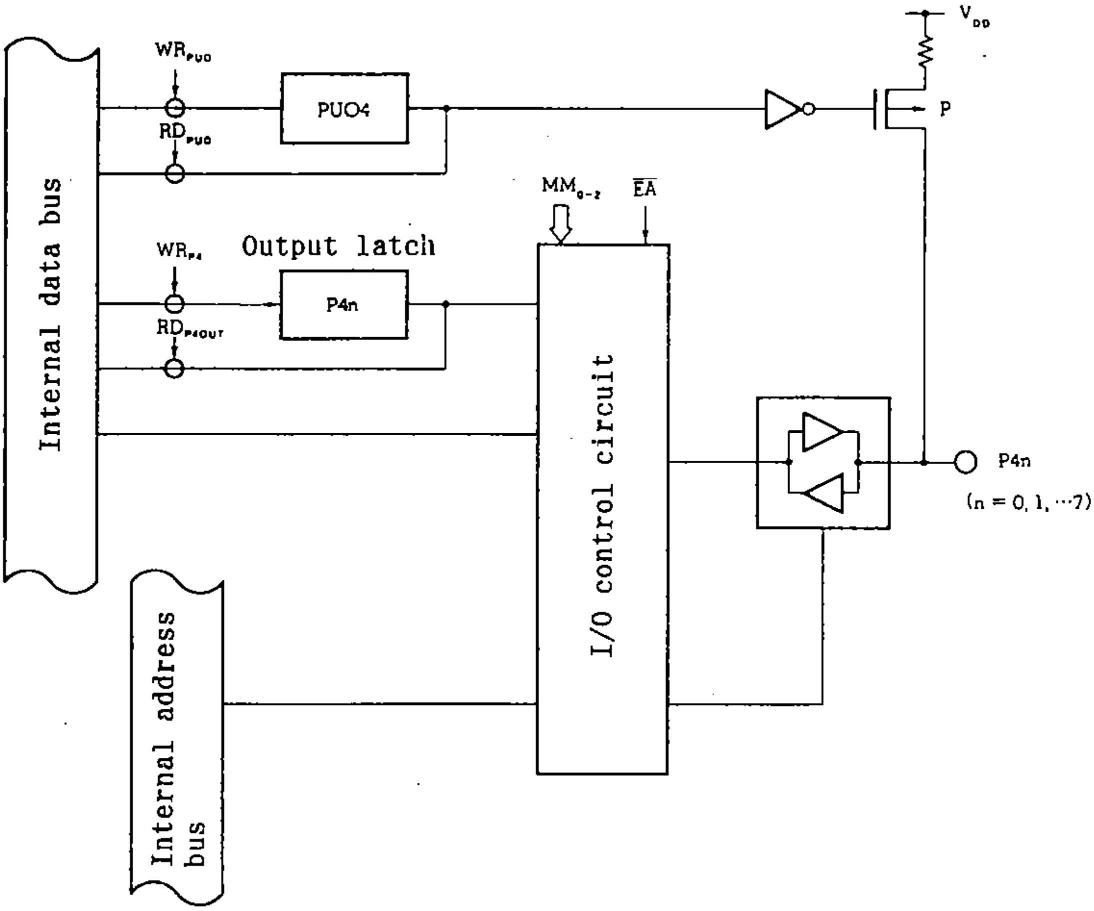
Table 5-6 Functions of Port 4

Pin	Port function	Also usable as	Non-port function
P40	I/O	AD0	Time multiplexing address/data bus at external expansion Lower address: A0-A7 Data : D0-D7
P41		AD1	
P42		AD2	
P43		AD3	
P44		AD4	
P45		AD5	
P46		AD6	
P47		AD7	

5.6.1 Hardware configuration

Figure 5-17 shows the hardware configuration of port 4.

Fig. 5-17 Block Diagram of Port 4



5.6.2 Setting the I/O mode and/or control mode

The I/O mode of port 4, either the port mode or address/data bus mode, is selected by the external access pin ($\overline{\text{EA}}$) and memory mapping register (MM), as shown in Figure 5-18.

Fig. 5-18 Port 4 Operation Mode Set by $\overline{\text{EA}}$ and MM

	7	6	5	4	3	2	1	0	Address	When reset	R/W
MM	IFCH	0	PW21	PW20	0	MM2	MM1	MM0	FF4CH	20H	W

$\overline{\text{EA}}$ pin	MM2	MM1	MM0	Port 4 operation mode	
1	0	0	0	Port operation (P40-P47)	Input port
	0	0	1		Output port
	0	1	1	Address/data bus operation (AD0-AD7)	256-byte expansion
	1	1	1		(*1)
0	x	x	x		64K-byte expansion (*2)

*1 32K-byte expansion (uPD78138 and uPD78P138)
40K-byte expansion (uPD78136)
48K-byte expansion (uPD78134A)

*2 ROM-less mode

Remark: $\overline{\text{EA}}$: External access pin

MM: Memory mapping register

5.6.3 Internal pull-up resistors

Port 4 is provided with internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of internal pull-up resistors is specified by PU04 of the register for optional pull-up resistors (PUO) for the eight bits at one time. (Bit-wise specification is not permitted.)

The pull-up resistors can be connected to port 4 regardless of the I/O mode.

Fig. 5-19 Format of the Register for Optional Pull-up Resistors

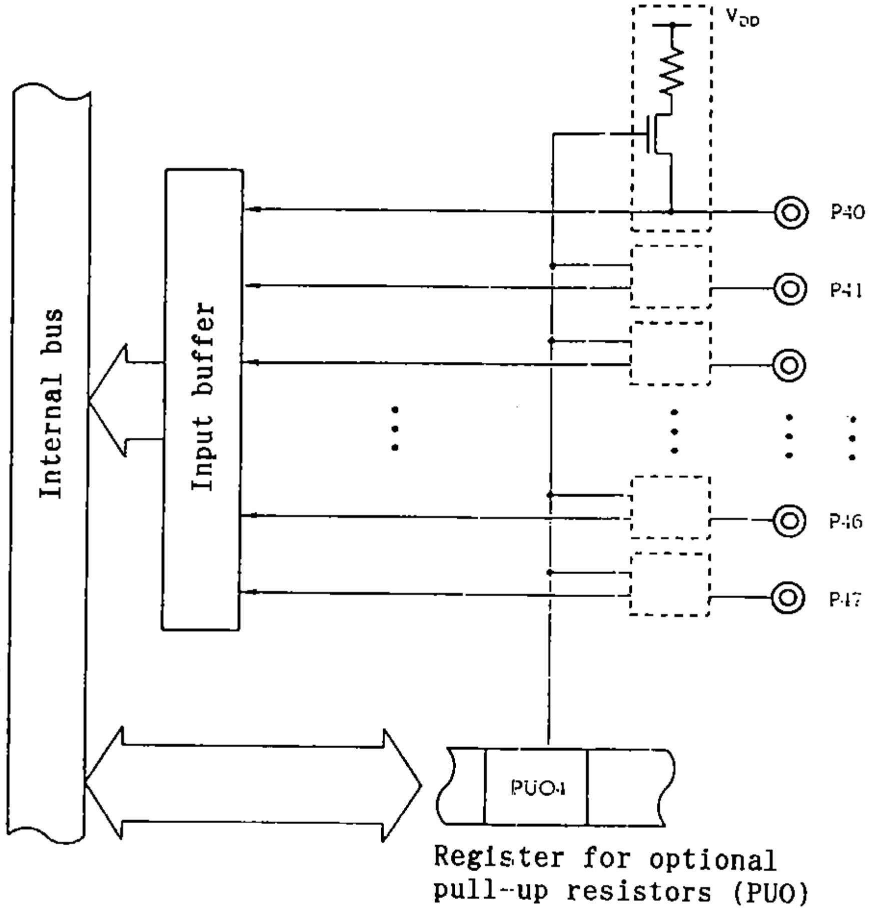
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU04	Use of pull-up resistors in port 4
0	Pull-up resistors are not used in port 4.
1	Pull-up resistors are used in port 4.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

Port 4 can be connected to V_{DD} with pull-up resistors just according to the specification in the PUO register. The use of pull-up resistors can be set for all bits at a time.

Fig. 5-20 Specification for Pulling Voltage in Port 4 High



Caution: Pull-up resistors can be used for port 4 regardless of whether the port is in the input output mode.

5.7 Port 5 (P50-P57)

Port 5 is an 8-bit I/O port. Input/output can be specified for each bit of port 5.

The port also functions as the address bus (A8 to A15) when external memory or I/O is connected.

Port 5 is provided with software-controlled pull-up resistors.

Table 5-7 shows the functions of port 5.

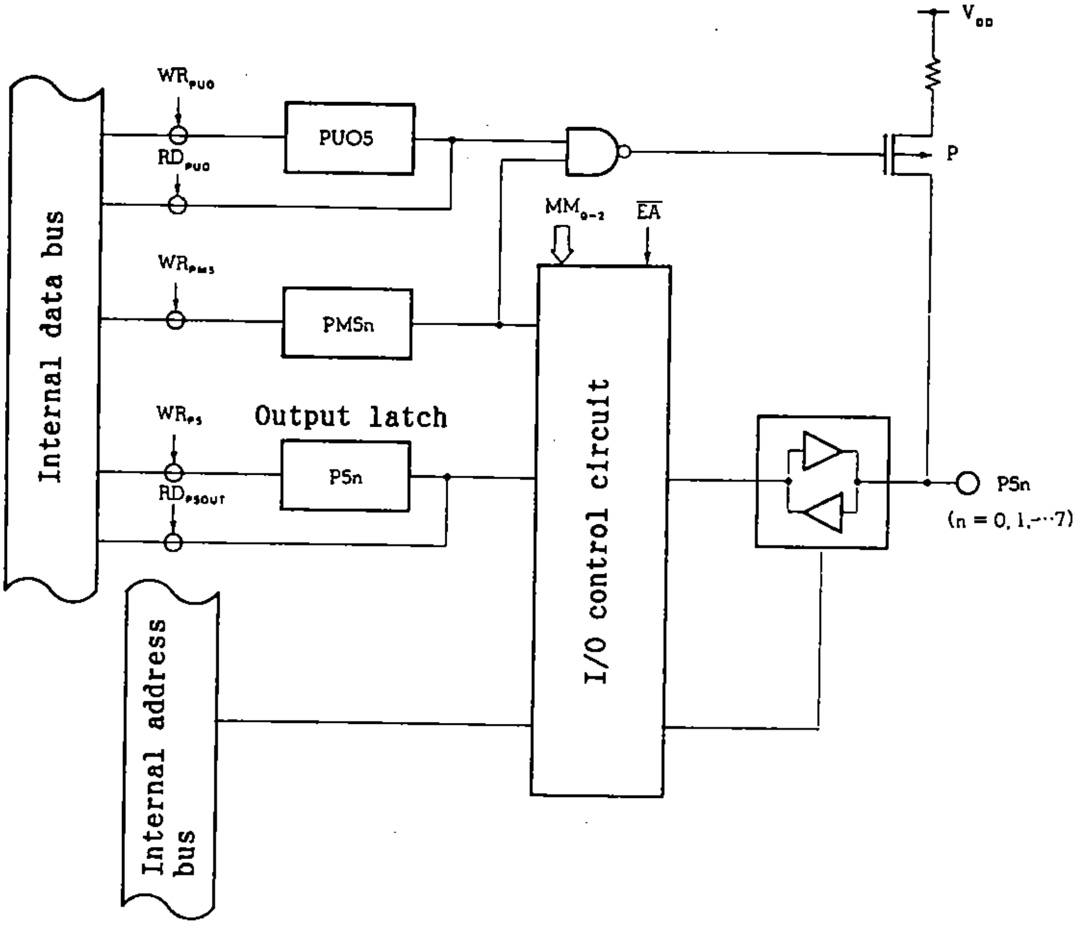
Table 5-7 Functions of Port 5

Pin	Port function	Also usable as	Non-port function
P50	I/O	A8	Time multiplexing address bus at external expansion Higher address: A8-A15
P51		A9	
P52		A10	
P53		A11	
P54		A12	
P55		A13	
P56		A14	
P57		A15	

5.7.1 Hardware configuration

Figure 5-21 shows the hardware configuration of port 5.

Fig. 5-21 Block Diagram of Port 5



5.7.2 Setting the I/O mode and/or control mode

The I/O mode of port 5, either the port mode or address/ data bus mode, is selected by the external access pin ($\overline{EA}$) and memory mapping register (MM), as shown in Figure 5-22.

Input/output for the port is specified bit by bit by the port 5 mode register as shown in Figure 5-23.

Fig. 5-22 Port 5 Operation Mode Set by $\overline{EA}$ and MM

	7	6	5	4	3	2	1	0	Address	When reset	R/W
MM	IFCH	0	PW21	PW20	0	MM2	MM1	MM0	FFC4H	20H	W

$\overline{EA}$ pin	MM2	MM1	MM0	Port 5 operation mode	
1	0	0	x	Port operation (P50-P57)	I/O is specified by PM5.
	0	1	1		
	1	1	1	Address bus operation (A8-A15)	(*1)
0	x	x	x		64K-byte expansion (*2)

*1 32K-byte expansion (uPD78138, uPD78P138)
40K-byte expansion (uPD78136)
48K-byte expansion (uPD78134A)

*2 ROM-less mode

Remark: $\overline{EA}$: External access pin
MM: Memory mapping register

Fig. 5-23 Format of the Port 5 Mode Register (PM5)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50	FF25H	FFH	W

PM5n	P5n pin I/O specification (n = 0-7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Caution: A bit manipulation instruction manipulates a bit, but it accesses a port in units of 8 bits. When a bit manipulation instruction is used for a port including both input and output pins, the contents of the output latches for the pins that are in the input mode become undefined (excluding the manipulated pins). Take care particularly when there is a pin used by switching between input and output. This applies also when a port is manipulated with other instructions.

5.7.3 Internal pull-up resistors

Port 5 is provided with internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of the internal pull-up resistors is specified with the following two resistors:

. Bit 5 (PUO5) of the register for optional pull-up resistors (PUO)
. Port 5 mode register (PM5)

The use of pull-up resistors can be specified for each pin of port 5.

If PUO5 is set to 1, the pull-up resistors for the pins that are placed in the input mode by PM5 are used.

Fig. 5-24 Format of the Register for Optional Pull-up Resistors

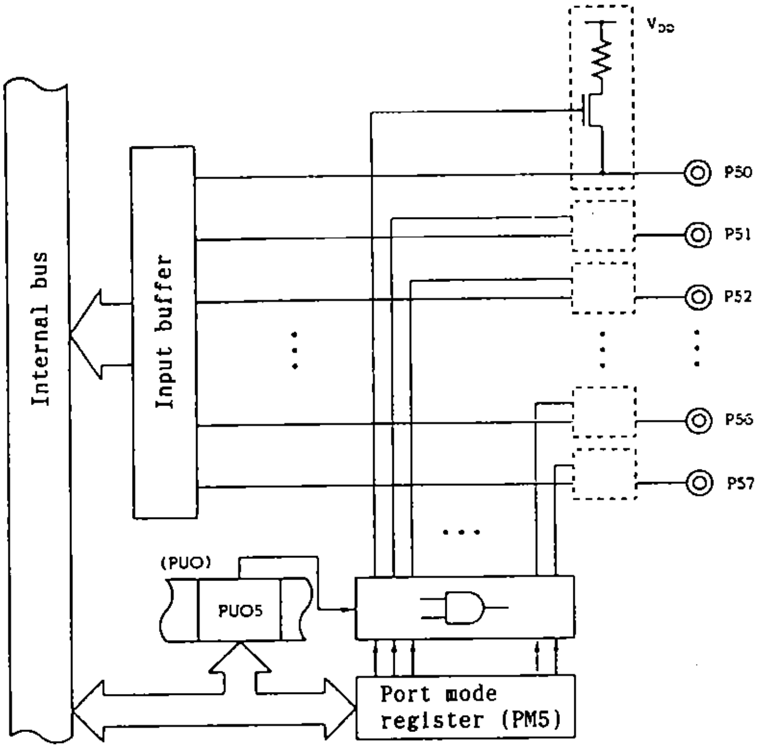
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU05	Use of pull-up resistors in port 5
0	Pull-up resistors are not used in port 5.
1	Pull-up resistors are used in port 5.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

If PU05 is 1, and if PU05 is associated with port 5, a pull-up resistor is made available to bits in the port which are specified as inputs by the port 5 mode register (PM5) for that port.

Fig. 5-25 Specification for Pulling Voltage in Port 5 High



5.8 Port 6 (P60-P67)

Port 6 consists of an output port on its four low-order bits (P60 to P63) and an I/O port on its four high-order bits (P64 to P67). For the I/O port, input or output can be specified on a bit-by-bit basis.

Bits 4 and 5 (P64 and P65) of port 6 also function as $\overline{RD}$ and $\overline{WR}$ control outputs, respectively, when external memory or I/O is connected.

For the four high-order bits of port 6, software-controlled pull-up resistors can be used.

Table 5-8 shows the functions of port 6.

Table 5-8 Functions of Port 6

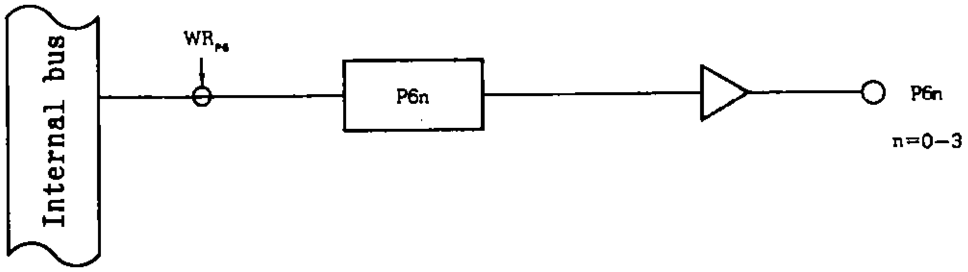
Pin	Port function	Also usable as	Non-port function
P60	Output	-	-
P61			
P62			
P63			
P64	I/O	$\overline{RD}$	Strobe signal output for external memory read
P65		$\overline{WR}$	Strobe signal output for external memory write
P66		-	-
P67			

5.8.1 Hardware configuration

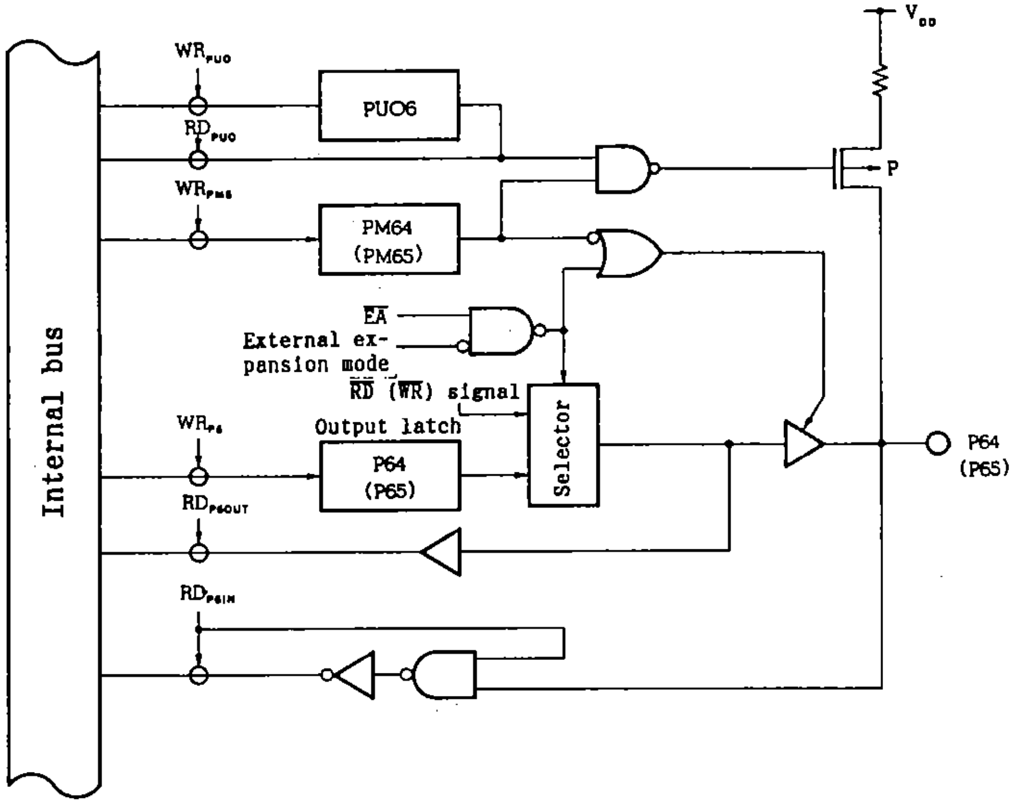
Figure 5-26 shows the hardware configuration of port 6.

Fig. 5-26 Block Diagram of Port 6

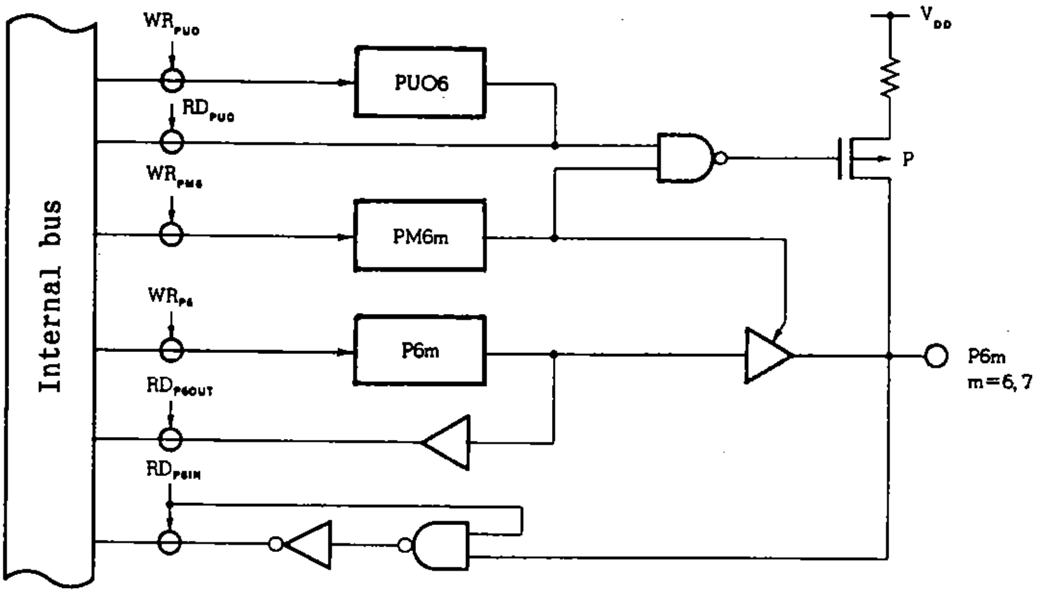
(a) P60 to P63



(b) P64 and P65



(c) P66, P67



5.8.2 Setting the I/O mode and/or control mode

The operation mode of bits 4 and 5 (P64 and P65) of port 6, either the port mode or external memory control mode, is selected by the external access pin ($\overline{EA}$) and memory mapping register (MM).

The input or output mode is set for each of the four high-order bits of port 6 by the port 6 mode register (PM6), as shown in Figure 5-28. PM6 is set with an 8-bit data transfer instruction. (So PM6 cannot be manipulated or read on a bit-by-bit basis.)

Fig. 5-27 P64 and P65 Operation Modes Set by $\overline{EA}$ and MM

	7	6	5	4	3	2	1	0	Address	When reset	R/W
MM	IFCH	0	PW21	PW20	0	MM2	MM1	MM0	FFC4H	20H	W

$\overline{EA}$ pin	MM2	MM1	MM0	P64	P65
1	x	0	x	Port operation (I/O is specified by PM6.)	
	x	1	x	$\overline{RD}$	$\overline{WR}$
0	x	x	x		

Remark: $\overline{EA}$: External access pin
MM: Memory mapping register

Fig. 5-28 Format of the Port 6 Mode Register

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM6	PM67	PM66	PM65	PM64	0	0	0	0	FF26H	F0H	W

PM6n	P6n pin I/O specification (n = 4-7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Caution: A bit manipulation instruction manipulates a bit, but it accesses a port in units of 8 bits. When a bit manipulation instruction is used for a port including both input and output pins, the contents of the output latches for the pins that are in the input mode become undefined (excluding the manipulated pins). Take care particularly when there is a pin used by switching between input and output. This applies also when a port is manipulated with other instructions.

5.8.3 Internal pull-up resistors

The four high-order bits (P64 to P67) of port 6 are provided with internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of the internal pull-up resistors is set with the following two registers:

- Bit 6 (PU06) of the register for optional pull-up resistors (PUO)
- Port 6 mode register (PM6)

The use of a pull-up resistor can be specified for the pins that are placed in the input mode at one time.

If PU06 is set to 1, the pull-up resistors for the pins that are placed in the input mode by PM6 are used.

Fig. 5-29 Format of the Register for Optional Pull-up Resistors

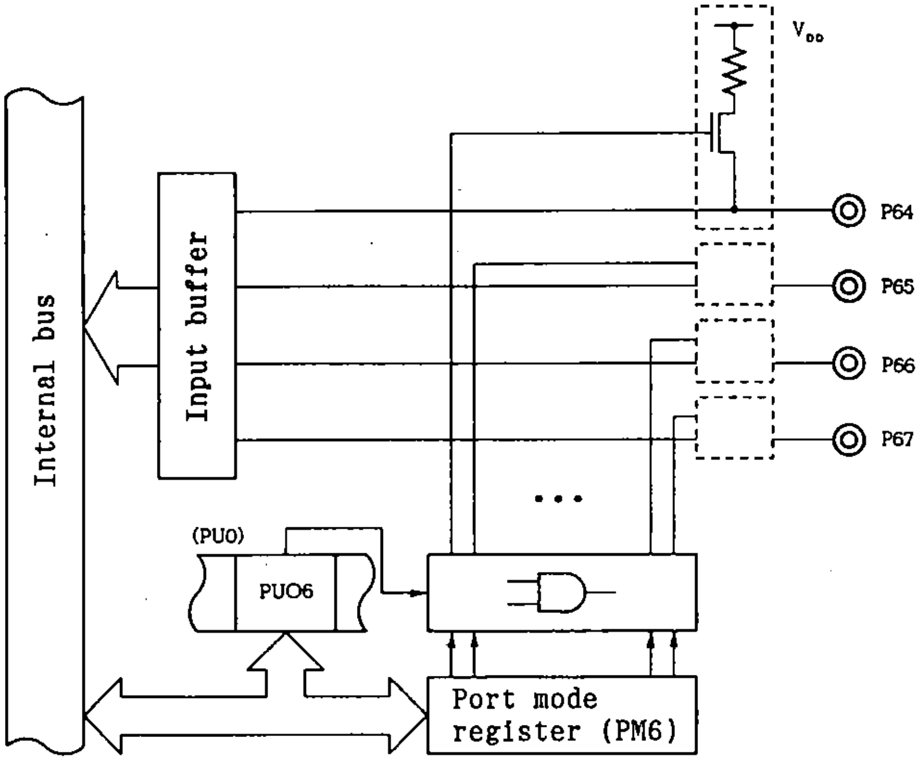
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU06	Use of the pull-up resistor for P6n pin (n = 4-7)
0	Pull-up resistors are not used in port 6.
1	Pull-up resistors are used in port 6.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

If PU06 is 1, and if PU06 is associated with port 6, a pull-up resistor is made available to bits in the port which are specified as inputs by the port 6 mode register (PM6) for that port.

Fig. 5-30 Specification for Pulling Voltage in Port 6 High



Caution: Pull-up resistors are not contained for the four low-order bits (P60 to P63) in port 6.

5.9 Port 7 (P70, P71)

Port 7 is a 2-bit I/O port. Input/output can be specified for two bits of port 7.

Port 7 is provided with software-controlled pull-up resistors.

Table 5-9 shows the functions of port 7.

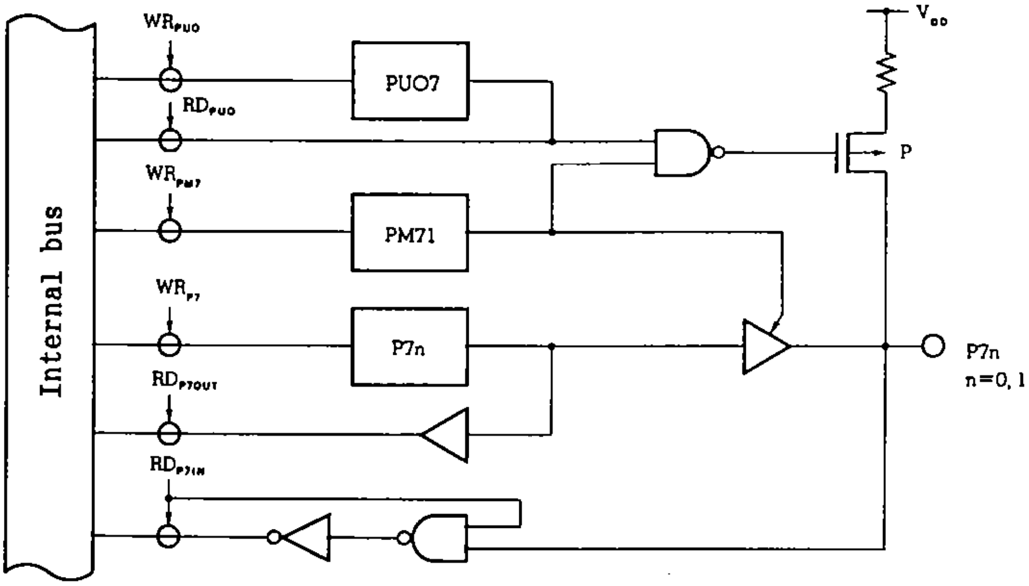
Table 5-9 Functions of Port 7

Pin	Port function	Also usable as	Non-port function
P70	I/O	-	-
P71			

5.9.1 Hardware configuration

Figure 5-31 shows the hardware configuration of port 7.

Fig. 5-31 Block Diagram of Port 7



5.9.2 Setting the I/O mode and/or control mode

The I/O mode is set for two bits of port 7 by port 7 mode register (PM7) as shown in Figure 5-32.

PM7 is set with an 8-bit data transfer instruction. So PM7 cannot be manipulated or read on a bit-by-bit basis.

Fig. 5-32 Format of Port 7 Mode Register (PM7)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM7	-	-	-	-	-	-	PM71	-	FF27H	FFH	W

PM71	Specification of I/O for P70 and P71 pins
0	Output mode (output buffer available)
1	Input mode (output buffer not available)

Caution: Either 00H or FFH must be written to port 7 mode register (PM7).

. Put P70 and P71 in the input mode
PM7 ← 00H

. Put P70 and P71 in the output mode
PM7 ← FFH

5.9.3 Internal pull-up resistors

Port 7 is provided with internal pull-up resistors. When pull-up resistors are needed, using these internal pull-up resistors, instead of mounting separate pull-up resistors, reduces the number of components for smaller mounting space.

The use of the internal pull-up resistors is set with the following two registers:

- Bit 7 (PU07) of the register for optional pull-up resistors (PUO)
- Port 7 mode register (PM7)

The use of a pull-up resistor can be specified for two bits of port 6.

If PU06 is set to 1, the pull-up resistors for the pins that are placed in the input mode by PM7 are used.

Fig. 5-33 Format of the Register for Optional Pull-up Resistors

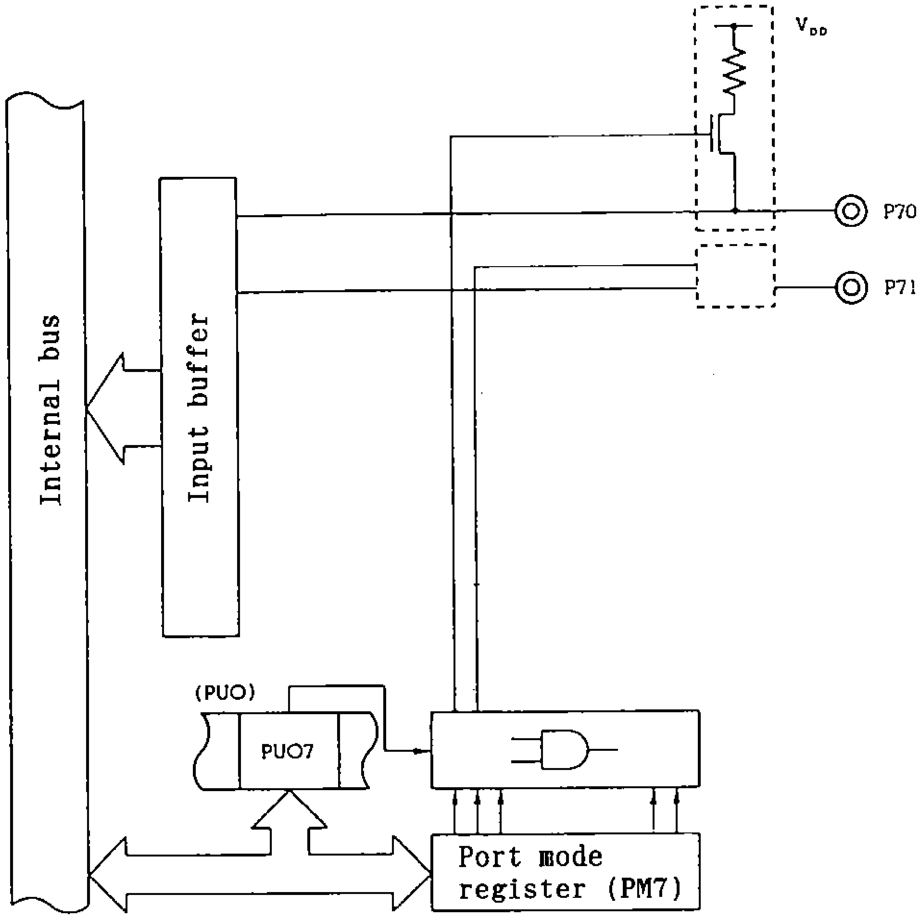
	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUO	PU07	PU06	PU05	PU04	PU03	PU02	PU01	0	FF40H	00H	R/W

PU07	Use of the pull-up resistor for P7n pin (n = 0, 1)
0	Pull-up resistors are not used in port 7.
1	Pull-up resistors are used in port 7.

Remark: When the STOP mode is entered, it is advisable to set PUO to 00H to reduce current consumption.

If PU07 is 1, and if PU07 is associated with port 7, a pull-up resistor is made available to bits in the port which are specified as inputs by the port 7 mode register (PM7) for that port.

Fig. 5-34 Specification for Pulling Voltage in Port 7 High



Caution: Port 7 is two-bit wide, consisting of P70 and P71.

CHAPTER 6 REAL-TIME OUTPUT PORT (RTP)

6.1 Configuration and Functions of RTP

As shown in Figure 6-1, the real-time output port consists of port 0 and buffer registers P0H and P0L, and other hardware.

Upon generation of a timer interrupt or external interrupt, data stored in the buffer register is transferred to the hardware output latch for output. This function is referred to as a real-time output function. A port used for this function is referred to as the real-time output port (RTP).

RTP of the uPD78138 is an 8-bit real-time output port. Real-time output data that can be handled may be in one of the following forms. The mode, either the real-time output port mode or port mode, is selected in 4-bit units.

. Two 4-bit-wide channels
. One 8-bit-wide channel

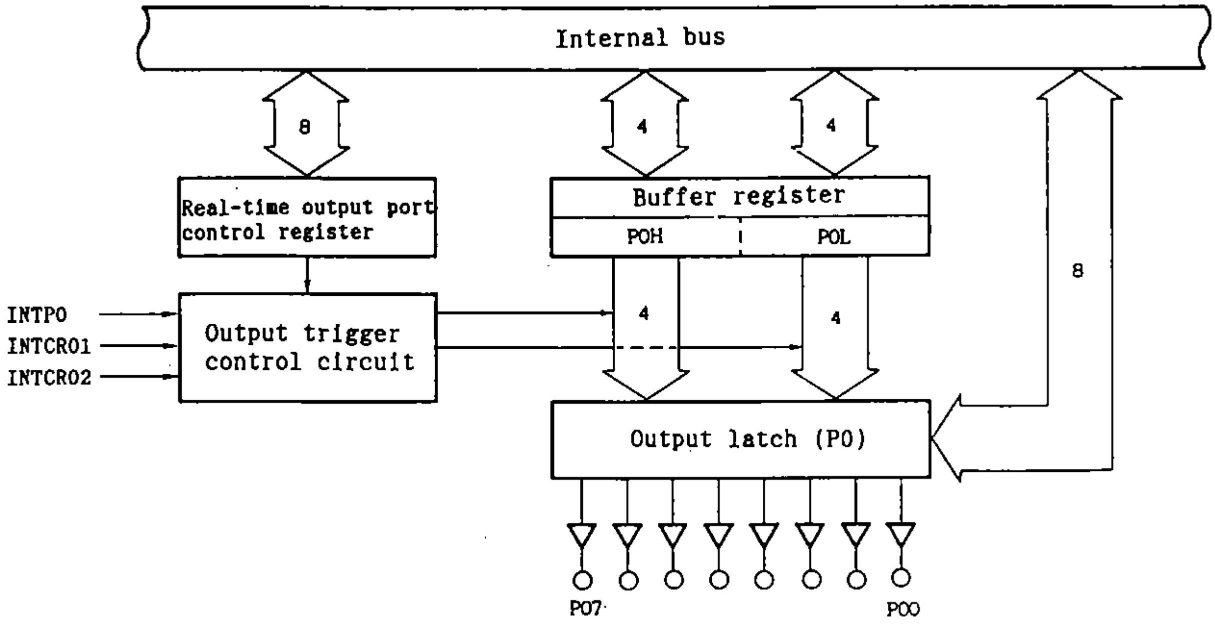
The following triggers cause the buffer register contents to be transferred to the output latch:

. INTCR01: Signal issued when the contents of 16-bit timer 0 and compare register CR01 match
. INTCR02: Signal issued when the contents of 16-bit timer 0 and compare register CR02 match
. INTP0: Valid edge input to the P21/INTP0 pin

6.2 Hardware Configuration

Figure 6-1 shows the hardware configuration of RTP.

Fig. 6-1 Block Diagram of RTP



6.3 Structure of the Buffer Register

As shown in Figure 6-2, buffer registers P0H and P0L are mapped in separate locations in the SFR area. If the real-time output function is specified for two 4-bit-wide channels, buffer registers P0H and P0L are loaded with data independently of each other. If the real-time output function is specified for one 8-bit-wide channel, 8-bit data can be loaded in both buffer registers P0H and P0L by performing write operation with only one of the registers specified.

Fig. 6-2 Structure of Buffer Registers P0H and P0L

	4 high-order bits	4 low-order bits
FF4AH		P0L
FF4BH	P0H	

Example of setting data in the buffer registers

. For two 4-bit-wide channels

```
MOV P0L, #05H ;  Sets the P0L register to 0101B.
MOV P0H, #0C0H;  Sets the P0H register to 1100B.
```

. For one 8-bit-wide channel

```
MOV P0L, #0C5H;  Sets the P0L register to 0101B, and
                 the P0H register to 1100B.
Or,
MOV P0H, #0C5H
```

Caution: When P0L is read, the P0H contents are also read for the four high-order bits.
When P0H is read, the P0L contents are also read for the four high-order bits.

6.4 RTP Control Register

The real-time output port is controlled by the real-time output port control register (RTPC). RTPC is an 8-bit register and is read from or written to with 8-bit or bit manipulation instructions.

Figure 6-3 shows the format of RTPC. $\overline{\text{RESET}}$ input sets the RTPC register to 00H.

Fig. 6-3 Format of Real-time Output Port Control Register (RTPC)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
RTPC	BYTE	0	0	POMH	EXTR	0	0	POML	FF4CH	00H	R/W

POML	Specification of P0L function
0	Port mode
1	Real-time output port mode

EXTR	Data transfer from buffer register to output latch by INTP0
0	Disabled
1	Enabled BYTE=0: Transfers P0L only. BYTE=1: Transfers P0L and P0H.

POMH	Specification of P0H function
0	Port mode
1	Real-time output port mode

BYTE	Operation mode of real-time output port
0	4-bit separate real-time output port
1	8-bit real-time output port

Caution: Be sure to write a 0 in bits 1, 2, 5, and 6.

Table 6-1 Output Trigger for Real-time Output Port (When RTPC P0MH = P0ML = 1)

RTPC		Number of real-time output bits	Output trigger	
BYTE	EXTR		P0H	P0L
0	0	Four bits	INTCR01	INTCR02
	1		INTCR01	INTCR02 or INTP0
1	0	Eight bits	INTCR02	
	1		INTCR02 or INTP0	

6.5 RTP Operation

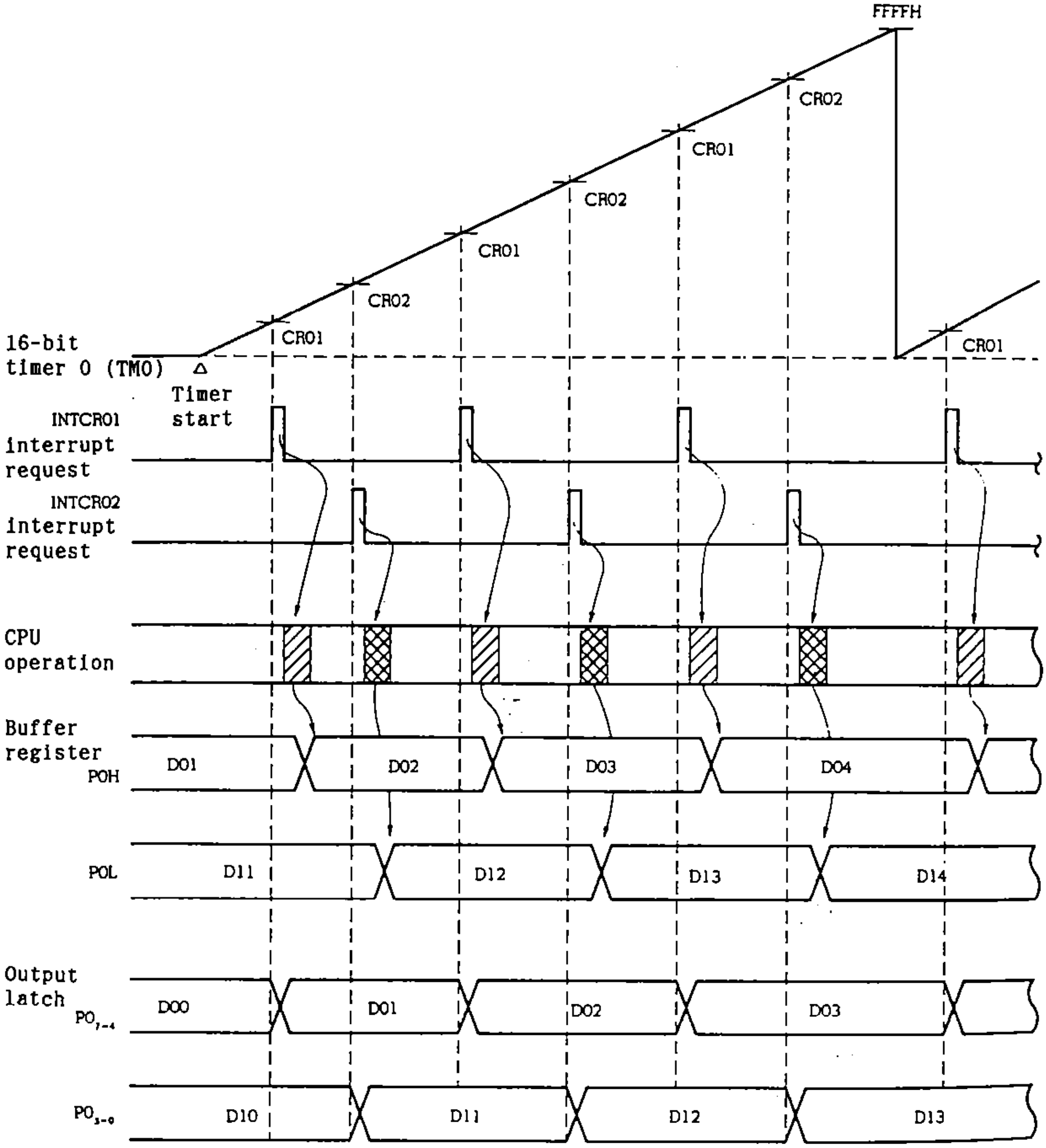
If the real-time output port function is specified for port 0, the contents of buffer registers P0H and P0L are fed into the output latch in synchronization with the occurrence of one of the trigger conditions listed in Table 6-1, then they are output on the pins in port 0.

For example, the signals issued when the contents of 16-bit timer 0 (TM0) and compare registers CR01 and CR02 match are selected as the output trigger sources (INTCR01 and INTCR02). Then, the output data on the pins in port 0 can change to the buffer register contents at an interval set in a selected compare register.

Using the real-time output port function with the macro service function enables the output data on the output pins in port 0 to change at given intervals. See Chapter 11 for information on the macro service.

Figure 6-4 gives an example of the operation timing of the real-time output port. The higher four bits of data on P04 to P07 in port 0 are rewritten, triggered by an interrupt (INTCR01) generated when the contents of 16-bit timer 0 and compare register CR01 match, and the lower four bits of data on P00 to P03 in port 0 are rewritten, triggered by an interrupt (INTCR02) generated when the contents of 16-bit timer 0 and compare register CR02 match.

Fig. 6-4 Timing of RTP Operation



▨ ▩ Buffer register contents are rewritten by software processing or macro service.

CHAPTER 7 SERIAL INTERFACE

7.1 Functions

Serial interface of the uPD78138 has two operation modes.

(1) Three-wire serial I/O mode (starting from MSB)

The 3-wire serial I/O mode uses three lines for 8-bit data transfer: serial clock ($\overline{SCK}$) and serial buses (SO and SI). This mode is applicable when the uPD78138 is connected to a peripheral I/O device or display controller containing a conventional clock synchronous serial interface.

(2) Serial bus interface (SBI) mode (starting from MSB)

The serial bus mode enables communication with more than one device by using two lines including the serial clock ($\overline{SCK}$) and serial data bus (SB0).

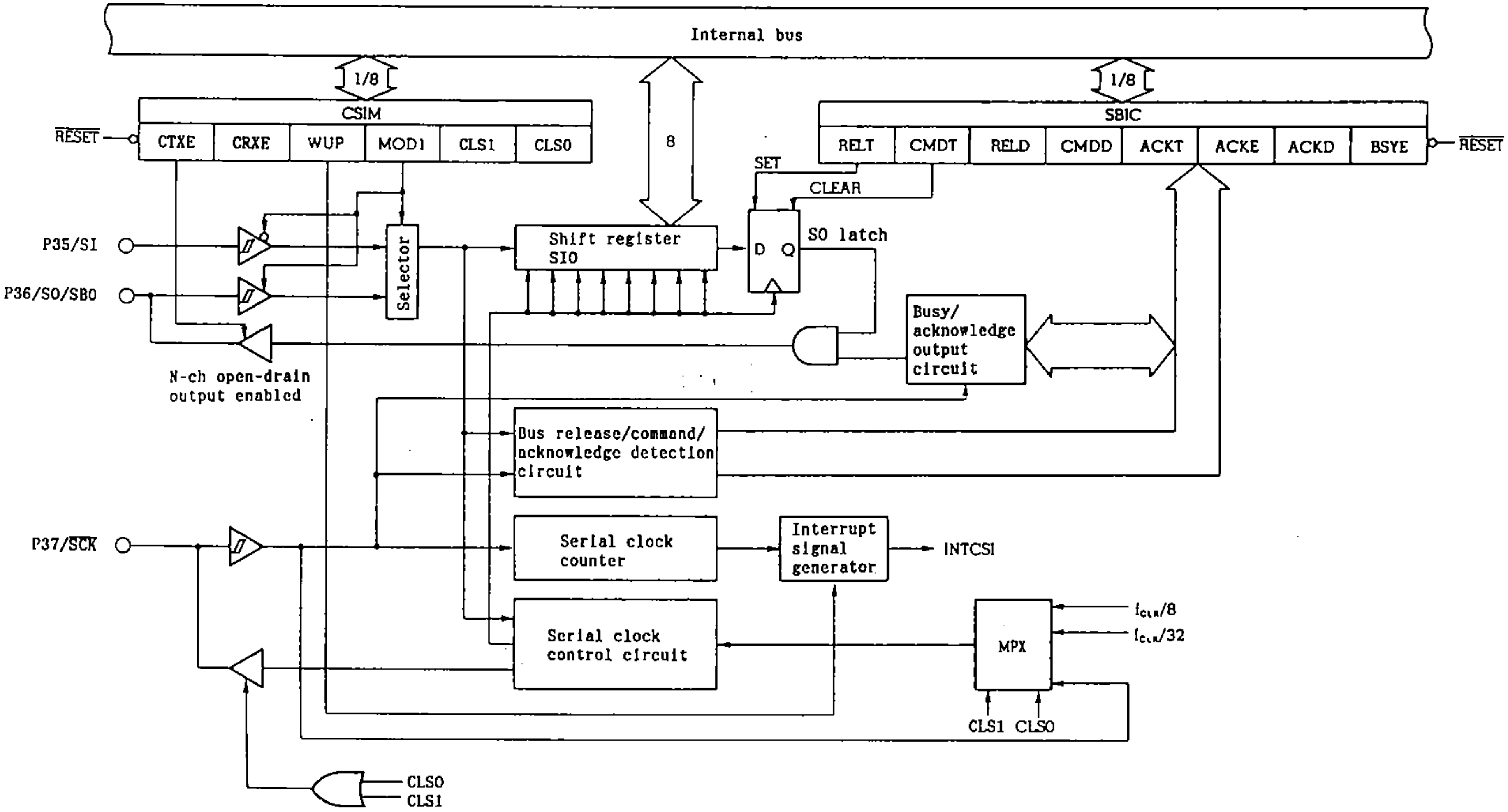
The serial bus interface mode conforms to the NEC serial bus format.

In the SBI mode, an address for selecting a device subject to serial communication, commands issued to that device, and actual data can be output onto the serial data bus. So the serial bus interface mode requires no lines for handshaking that used to be needed when more than one device is connected using a conventional clock synchronous serial interface. This helps efficient use of the I/O ports.

7.2 Configuration

The configuration of serial interface is shown below.

Fig. 7-1 Block Diagram of Serial Interface



(1) Shift register (SIO)

Shift register (SIO) converts 8-bit serial data to 8-bit parallel data, or vice versa. This register can be used for both send and receive operations.

Data is shifted in (received) and shifted out (sent) from the MSB side. Actual send/receive operation is controlled by writing to or reading from SIO.

This register can be read from or written to with an 8-bit manipulation instruction. The $\overline{\text{RESET}}$ signal causes this register to be undefined.

(2) SO latch

The SO latch holds the output level of the SO/SB0 pin. In the serial bus interface (SBI) mode, this latch can be directly controlled by software.

(3) Serial clock selector

The serial clock selector selects a serial clock to be used.

(4) Serial clock counter

The serial clock counter counts the serial clock to be output or input during send/receive operation, and checks whether 8-bit data has been sent or received.

(5) Interrupt signal generator

The interrupt signal generator controls whether to generate an interrupt request when the serial clock counter counts eight serial clock pulses. The interrupt signal generator generates an interrupt request when eight serial clock pulses are counted in the 3-wire serial I/O mode or when conditions are satisfied in the SBI mode.

(6) Serial clock control circuit

The serial clock control circuit controls the serial clock to be supplied to shift register. This circuit also controls the clock to be output on the $\overline{\text{SCK}}$ pin when the internal clock is used.

(7) Busy/acknowledge output circuit and bus release/ command/acknowledge detection circuit

The busy/acknowledge output circuit and bus release/command/acknowledge detection circuit output and detect various control signals in the SBI mode. These circuits do not operate in the 3-wire serial I/O mode.

7.3 Control Registers

7.3.1 Serial interface mode register (CSIM)

The CSIM register is an 8-bit register used to specify a serial interface operation mode, serial clock, wake-up function, and so forth.

The CSIM register can be read from or written to with an 8-bit manipulation instruction or bit manipulation instruction. Figure 7-2 shows the format of the register.

Fig. 7-2 Format of Serial Interface Mode Register (CSIM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
CSIM	CTXE	CRXE	WUP	0	MOD1	0	CLS1	CLS0	FF80H	00H	R/W

CLS1	CLS0	Serial clock selection		$\overline{\text{SCK}}$ pin	Master/slave selection in SBI mode
0	0	External clock		Input	Slave
0	1	Setting prohibited		Output	Master
1	0	Internal clock	$f_{CLK}/32$		
1	1		$f_{CLK}/8$		

WUP	MOD1	Operation mode	Control over wake-up function
0	0	3-wire serial I/O mode	Generates interrupt request for each serial transfer.
0	1	SBI mode	
1	1		Generates interrupt request only upon address reception.

CRXE	Receive operation
0	Disabled
1	Enabled

CTXE	Send operation
0	Disabled
1	Enabled

Caution: Do not change the state from CTXE = 0 and CRXE = 1 to CTXE = 1 and CRXE = 0 or from CTXE = 1 and CRXE = 0 to CTXE = 0 and CRXE = 1 using a single operation. If that is attempted, the serial clock counter malfunctions to terminate the first communication after the change before 8 bits are transferred. Use the following two instructions to perform the above change:

Example: To change the state from CTXE = 1 and CRXE = 0 to CTXE = 0 and CRXE = 1

```
CLR1 CTXE
SET1 CRXE
```

7.3.2 Serial bus interface control register (SBIC)

The SBIC register is an 8-bit register consisting of bits that control the serial bus state and bits that indicate the states of input data sent from the serial bus. The register can be used only in the SBI mode; it cannot not be manipulated in the 3-wire serial I/O mode.

The register can be manipulated with an 8-bit manipulation instruction or bit manipulation instruction. Table 7-1 indicates allowable read and/or write operation for each bit. When the register is read, 0's are read from the write-only bits. Figure 7-3 shows the format.

The $\overline{\text{RESET}}$ signal sets the register to 00H.

The ACKD, CMDD, and RELD flags are cleared when send/ receive operation is disabled (CTXE = CRXE = 0).

Table 7-1 Read/Write Operation of SBIC Register

	7	6	5	4	3	2	1	0
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT
	R/W	R	R/W	W	R	R	W	W

Remark: R/W: Read and write
R: Read only
W: Write only

Fig. 7-3 Format of Serial Bus Interface Control Register (SBIC)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	FF82H	00H	R/W

RELT	Bus release signal (REL) trigger output control
0	Not output
1	Output

CMDT	Command signal (CMD) trigger output control
0	Not output
1	Output

RELD	Bus release signal (REL) detection
0	Not detected
1	Detected

CMDD	Command signal (CMD) detection
0	Not detected
1	Detected

ACKT	Acknowledge signal ($\overline{\text{ACK}}$) trigger output control
0	Not output
1	Output

ACKE	Automatic acknowledge signal ($\overline{\text{ACK}}$) output
0	Disabled
1	Enabled

ACKD	Acknowledge signal ($\overline{\text{ACK}}$) detection
0	Not detected
1	Detected

BSYE	Automatic synchronous busy signal ($\overline{\text{BUSY}}$) output
0	Disabled
1	Enabled

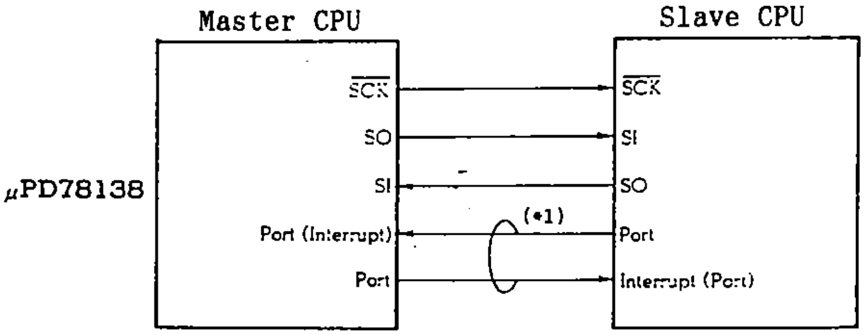
7.4 Three-wire Serial I/O Mode

The 3-wire serial I/O mode is used for communication with a device containing a conventional clock synchronous serial interface.

Basically, three lines including serial clock ($\overline{\text{SCK}}$), serial data output (SO), and serial data input (SI) are used for communication. When multiple devices are connected, additional lines for handshaking are required.

Fig. 7-4 Example of 3-Wire Serial I/O System Configuration

3-wire serial I/O ⟷ 3-wire serial I/O



*1 Handshaking line

7.4.1 Basic operation timing

In the 3-wire serial I/O mode, data is transferred block by block, with each block consisting of 8 bits. A block of data is transferred bit by bit starting with the MSB in phase with the serial clock.

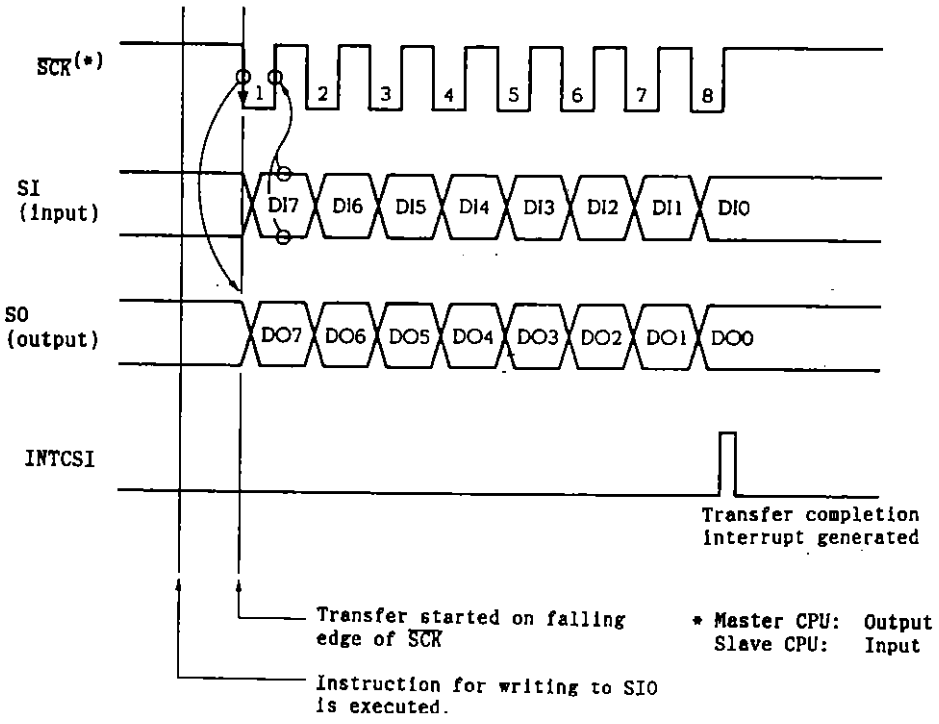
Send data is output on a falling edge of $\overline{\text{SCK}}$.
Receive data is sampled on a rising edge of $\overline{\text{SCK}}$.

An interrupt request (INTCSI) is generated on the eighth rising edge of $\overline{\text{SCK}}$.

If the internal clock is used for $\overline{\text{SCK}}$, $\overline{\text{SCK}}$ output is stopped on the eighth rising edge of $\overline{\text{SCK}}$; $\overline{\text{SCK}}$ remains high until the next data send or receive operation is started.

Figure 7-5 shows the timing of the 3-wire serial I/O mode.

Fig. 7-5 Timing of 3-Wire Serial I/O Mode



Transfer completion interrupt generated

Transfer started on falling edge of $\overline{\text{SCK}}$

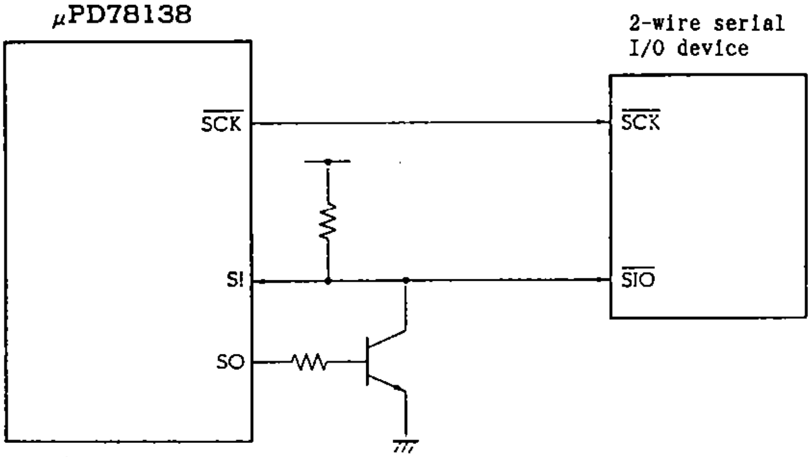
Instruction for writing to SIO is executed.

* Master CPU: Output
Slave CPU: Input

In the 3-wire serial I/O mode, the SO pin is configured as a CMOS push-pull output.

Remark: For connection with a 2-wire serial I/O device, connect a buffer to the SO pin as shown in Figure 7-6. In Figure 7-6, the buffer inverts the output level. Accordingly, to SIO, write the invert of desired data to be output.

Fig. 7-6 Example of Connection with 2-Wire Serial I/O Device



7.4.2 When only send operation is enabled

Send operation is performed when the CTXE bit of the clock synchronous serial interface mode register (CSIM) is set to 1. Send operation is started by writing to shift register (SIO) when the CTXE bit is set to 1.

When the CTXE bit is cleared to 0, the SO pin goes into the high-impedance state.

(1) When the internal clock is selected as the serial clock

When send operation is started, the serial clock is output on the $\overline{\text{SCK}}$ pin. At the same time, data from SIO is sequentially output on the SO pin on a falling edge of the serial clock. In addition, the signal on the SI pin is shifted into SIO on a rising edge of the serial clock.

It takes up to one clock of $\overline{\text{SCK}}$ from the start of send operation to the first falling edge of $\overline{\text{SCK}}$.

When sending is inhibited (the CTXE bit is reset to 0) during send operation, the $\overline{\text{SCK}}$ clock output is stopped at the next rising edge and the send operation is stopped. The interrupt request (INTCSI) does not occur. The output impedance of the SO pin becomes high and the contents of the SIO register become undefined.

(2) When an external clock is selected as the serial clock

After send operation is started, data from SIO is sequentially output on the SO pin on a falling edge of the serial clock applied to the $\overline{\text{SCK}}$ pin. At the same time, the signal on the SI pin is shifted into SIO on a rising edge of the signal applied to the $\overline{\text{SCK}}$. If the serial clock is applied to the $\overline{\text{SCK}}$ pin when send operation is not started yet, shift operation is not performed, and the output level on the SO pin does not change.

When sending is inhibited (the CTXE bit is reset to 0) during send operation, the send operation is stopped and subsequent $\overline{\text{SCK}}$ inputs are ignored. The interrupt request (INTCSI) does not occur. The output impedance of the SO pin becomes high and the contents of the SIO register become undefined.

7.4.3 When only receive operation is enabled

Receive operation is performed when the CRXE bit of the CSIM register is set to 1. Receive operation is started by changing the setting of the CRXE bit from 0 to 1 or by reading SIO.

(1) When the internal clock is selected as the serial clock

When receive operation is started, the serial clock is output on the $\overline{\text{SCK}}$ pin, and data on the SI pin is sequentially loaded into SIO on a rising edge of the serial clock.

It takes up to one clock of $\overline{\text{SCK}}$ from the start of receive operation to the first falling edge of $\overline{\text{SCK}}$.

When reception is inhibited (the CRXE bit is reset to 0) during receive operation, the $\overline{\text{SCK}}$ clock output is stopped at the next rising edge and the receive operation is stopped. The interrupt request (INTCSI) does not occur. The contents of the SIO register become undefined.

(2) When an external clock is selected as the serial clock

After receive operation is started, data on the SI pin is sequentially loaded into SIO on a rising edge of the serial clock applied to the $\overline{\text{SCK}}$ pin. If the serial clock is applied to the $\overline{\text{SCK}}$ pin when receive operation is not started yet, shift operation is not performed.

When reception is inhibited (the CRXE bit is reset to 0) during receive operation, the receive operation is stopped and subsequent $\overline{\text{SCK}}$ inputs are ignored. The interrupt request (INTCSI) does not occur. The contents of SIO register become undefined.

7.4.4 When send and receive operations are enabled

When both the CTXE and CRXE bits of the CSIM register are set to 1, send operation and receive operation (send/ receive operation) can be performed at the same time. Send/receive operation is started by setting the CRXE bit from 0 to 1 or by writing to SIO when the CTXE bit is set to 1.

When send/receive operation is started for the first time, the CRXE bit is always set from 0 to 1. This means that send/receive operation starts immediately, and can output undefined data. So before enabling send/receive operation, write the first send data to SIO when both send and receive operations are disabled (the CTXE bit and CRXE bit are reset to 0).

When send/receive operation is disabled (CTXE = 0, CRXE = 0), the SO pin goes into the high-impedance state.

(1) When the internal clock is selected as the serial clock

When send/receive operation is started, the serial clock is output on the $\overline{\text{SCK}}$ pin. At the same time, data from SIO is sequentially output on the SO pin on a falling edge of the serial clock. In addition, data on the SI pin is sequentially shifted into SIO on a rising edge of the serial clock.

It takes up to one clock of $\overline{\text{SCK}}$ from the start of send/receive operation to the first falling edge of $\overline{\text{SCK}}$.

When sending or reception is inhibited during send/ receive operation, only the inhibited operation is stopped. When only sending is inhibited, the output impedance of the SO pin becomes high. When only reception is inhibited, the contents of the SIO register become undefined.

When sending and reception are inhibited at the same time, the $\overline{\text{SCK}}$ clock output is stopped at the next rising edge and the send and receive operations are stopped. The contents of the SIO register become undefined and the interrupt request (INTCSI) does not occur. The output impedance of the SO pin becomes high.

(2) When an external clock is selected as the serial clock

After send/receive operation is started, data from SIO is sequentially output on the SO pin on a falling edge of the serial clock applied to the $\overline{\text{SCK}}$ pin. At the same time, data on the SI pin is sequentially shifted into SIO on a rising edge of the signal.

If the serial clock is applied to the $\overline{SCK}$ pin when send/receive operation is not started yet, shift operation is not performed, and the output level of the SO pin does not change.

When sending or reception is inhibited during send/ receive operation, only the inhibited operation is stopped. When only sending is inhibited, the output impedance of the SO pin becomes high. When only the reception is inhibited, the contents of the SIO register become undefined.

When sending and reception are inhibited at the same time, the send and receive operations are stopped and subsequent $\overline{SCK}$ inputs are ignored. The contents of the SIO register become undefined and the interrupt request (INTCSI) does not occur. The output impedance of the SO pin becomes high.

7.4.5 Action taken when shift operation is not in phase with the serial clock

If an external clock is selected for the serial clock, noise, for example, can cause a mismatch between the number of serial clock pulses and shift operation. In such a case, the serial clock counter is initialized by disabling both send operation and receive operation (by resetting the CTXE bit and CRXE bit to 0). So the serial clock pulse that is first applied after the next send or receive operation is enabled can be used as the first clock pulse to restore the synchronization of shift operation with the serial clock.

7.5 SBI Mode

The serial bus interface (SBI) is a high-speed serial interface that conforms to the NEC serial bus format.

To allow communication with multiple devices on a single-master, high-speed serial bus using two signal lines, the SBI has a bus configuration function added to the clock synchronous serial I/O method. So the SBI can reduce ports and wires on boards when multiple microcomputers and peripheral ICs are used to configure a serial bus.

For information about the SBI functions, also refer to "Serial Bus Interface (SBI) User's Manual (IEM-5040)."

7.5.1 SBI features

Conventional serial I/O methods provide only data transfer functions. Therefore, when a serial bus is configured connecting multiple devices, many ports and wires are required to identify chip select signals, commands, and data, and to detect busy states. If an attempt is made to control these jobs by software, an increased software load results.

The SBI method can configure a serial bus with two signal lines: serial clock $\overline{SCK}$ and serial data bus SB0. For this reason, the number of ports on a microcomputer can be reduced, and wiring on a circuit board can be simplified.

The SBI functions are described below.

(1) Address/command/data identification function

Serial data is classified into three types: address, command, and data.

(2) Address-based chip select function

The master selects a slave chip by address transfer.

(3) Wake-up function

A slave can easily check address reception (for chip select identification) with the wake-up function. This function can be set or reset by software.

If the wake-up function is set, serial reception interrupt (INTCSI) is generated only when the slave receives its address.

So, in communication with multiple devices, a CPU other than a selected slave can operate independently of serial communication.

(4) Acknowledge signal ($\overline{\text{ACK}}$) control function

The acknowledge signal used to confirm the reception of serial data can be controlled.

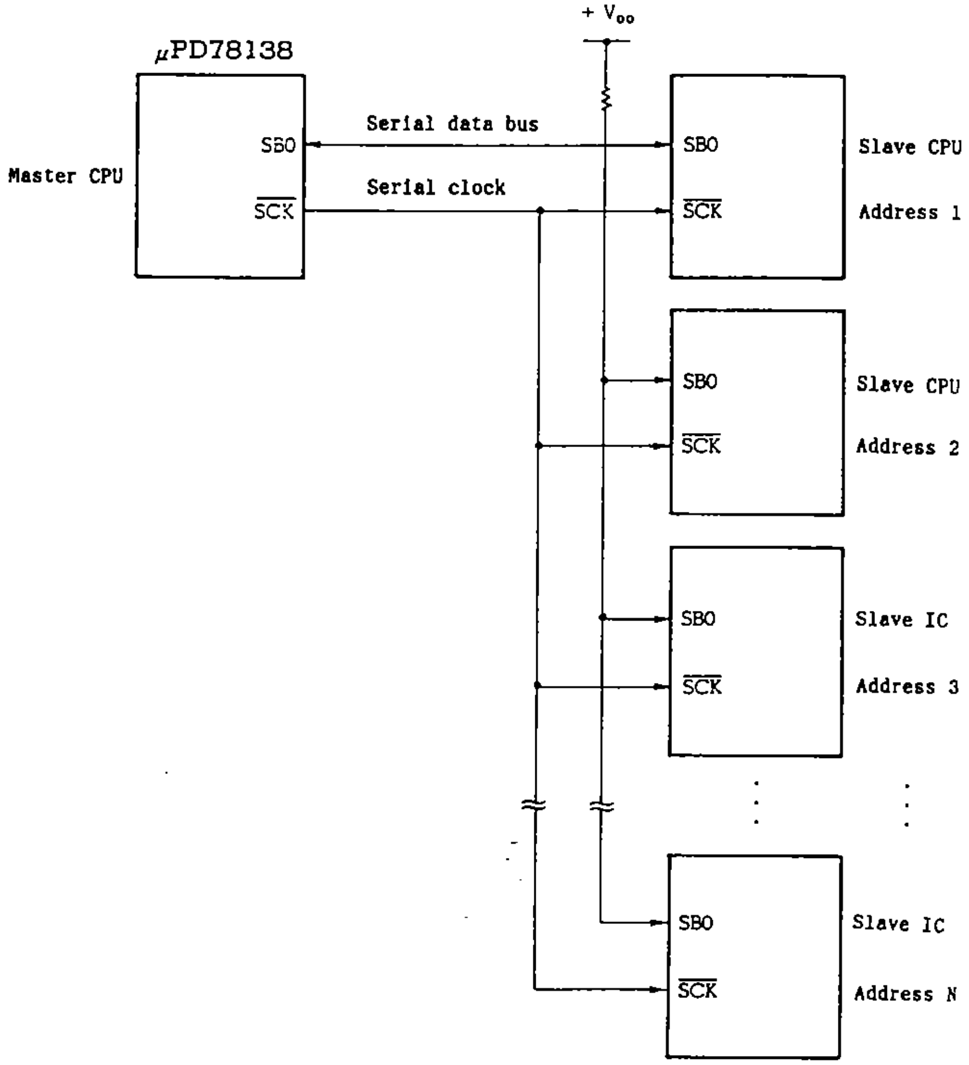
(5) Busy signal ($\overline{\text{BUSY}}$) control function

The busy signal used to post the busy state of a slave can be controlled.

Figure 7-7 shows an example of serial bus configuration that contains peripheral ICs and CPUs with an SBI-based serial interface.

In the SBI mode, the serial data bus pin SB0 is configured as an open-drain output. So the serial data bus line is placed in the wired OR state. A pull-up resistor is required for the serial data bus line.

Fig. 7-7 Example of SBI-Based Serial Bus Configuration

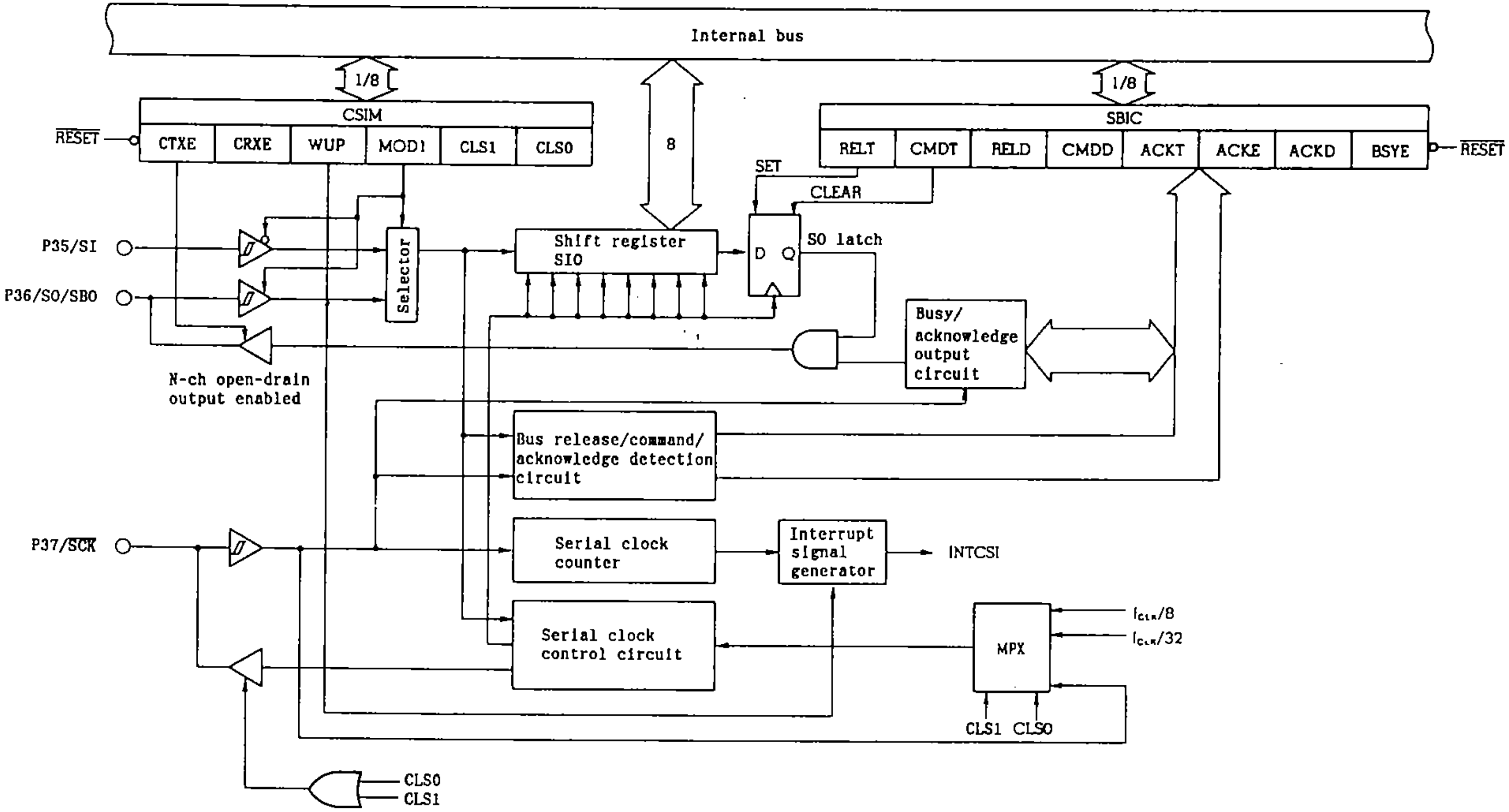


Caution: To switch between the master and slave, a pull-up resistor is required also for the serial clock line ($\overline{SCK}$) because $\overline{SCK}$ input/output switching is performed between the master and slave asynchronously.

7.5.2 Serial interface configuration

Figure 7-8 shows the block diagram of serial interface channel 0.

Fig. 7-8 Block Diagram of Serial Interface



The serial clock pin ($\overline{SCK}$) and serial data bus pin (SB0) are configured as described below.

(1) $\overline{SCK}$: Pin for serial clock I/O

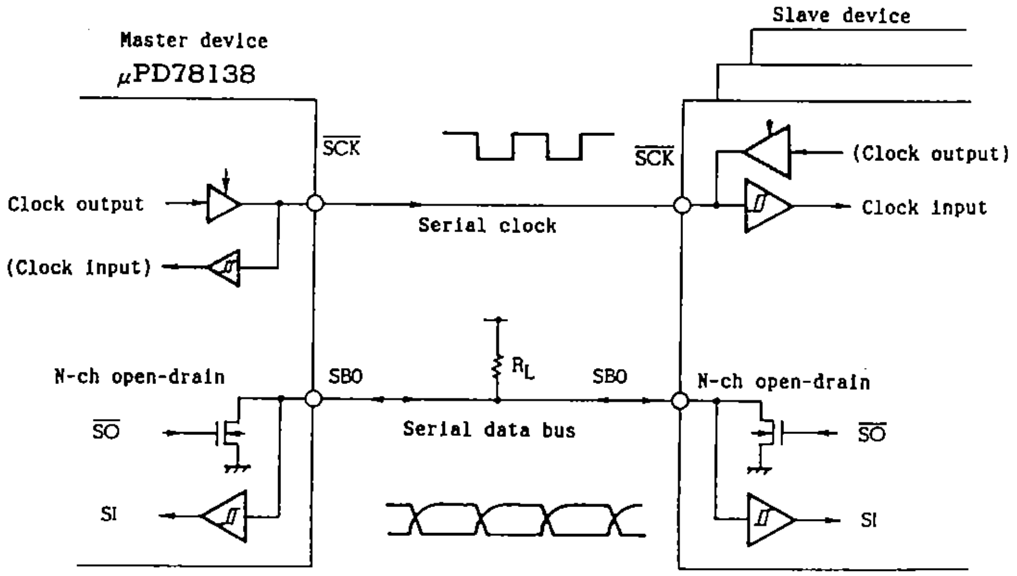
. Master: CMOS, push-pull output
. Slave: Schmitt input

(2) SB0: Pin usable for both serial data input and output

For both the master and slave, the output is configured as an N-ch open drain output, and the input is configured as a Schmitt input.

The serial data bus line output is configured as an N-ch open-drain output, so it requires an external pull-up resistor.

Fig. 7-9 Pin Configuration



7.5.3 Address match detection method

In the SBI mode, communication starts when the master selects a particular slave device by outputting an address.

A slave detects an address match by software. In the wake-up state (WUP = 1), a slave generates a serial transfer completion interrupt request only when its address is received.

In match address receive processing by software, the wake-up state is released (WUP ← 0), then a preparation is made to receive subsequent commands and data.

7.5.4 SBI mode control registers

(1) Clock synchronous serial interface mode register (CSIM0)

The CSIM register is an 8-bit register used to specify a serial interface operation mode, serial clock, wake-up function, and so forth.

Figure 7-10 shows the format of the CSIM register.

The CSIM register can be read from or written to with an 8-bit manipulation instruction or bit manipulation instruction. The register has a read/write attribute bit by bit.

The $\overline{\text{RESET}}$ signal sets the CSIM register to 00H.

Fig. 7-10 Format of Serial Interface Mode Register (CSIM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
CSIM	CTXE	CRXE	WUP	0	MOD1	0	CLS1	CLS0	FF80H	00H	R/W

CLS1	CLS0	Serial clock selection		$\overline{SCK}$ pin	Master/slave selection in SBI mode
0	0	External clock		Input	Slave
0	1	Setting prohibited		Output	Master
1	0	Internal clock	$f_{CLK}/32$		
1	1		$f_{CLK}/8$		

WUP	MOD1	Operation mode	Control over wake-up function
0	0	3-wire serial I/O mode	Generates interrupt request for each serial transfer.
0	1	SBI mode	
1	1		Generates interrupt request only upon address reception.

CRXE	Receive operation
0	Disabled
1	Enabled

CTXE	Send operation
0	Disabled
1	Enabled

Caution: Do not change the state from CTXE = 0 and CRXE = 1 to CTXE = 1 and CRXE = 0 or from CTXE = 1 and CRXE = 0 to CTXE = 0 and CRXE = 1 using a single operation. If that is attempted, the serial clock counter malfunctions to terminate the first communication after the change before 8 bits are transferred. Use the following two instructions to perform the above change:

Example: To change the state from CTXE = 1 and CRXE = 0 to CTXE = 0 and CRXE = 1

```
CLR1 CTXE
SET1 CRXE
```

(2) Serial bus interface control register (SBIC)

The SBIC register is an 8-bit register consisting of bits that control the serial bus state and flags that indicate the states of input data sent from the serial bus.

The register can be read from or written to with an 8-bit manipulation instruction or bit manipulation instruction. The allowable read and/or write operation depends on each bit. Figure 7-11 shows the format.

The $\overline{\text{RESET}}$ signal sets the register to 00H.

Fig. 7-11 Format of SBIC Register (1/3)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	FF82H	00H	R/W

Bus release trigger bit (W)

RELT	Bus release signal (REL) trigger output control bit. Setting the RELT bit sets the SO latch to 1, then the RELT bit is automatically cleared to 0.

Command trigger bit (W)

CMDT	Command signal (CMD) trigger output control bit. Setting the CMDT bit clears the SO latch to 0, then the CMDT bit is automatically cleared to 0.

Fig. 7-11 Format of SBIC Register (2/3)

Bus release detection flag (R)

	Conditions for being cleared (RELD = 0)	Conditions for being set (RELD = 1)
RELD	① Transfer start instruction execution ② RESET signal input ③ CTXE = CRXE = 0	Detection of the bus release signal (REL)

Command detection flag (R)

	Conditions for being cleared (CMDD = 0)	Conditions for being set (CMDD = 1)
CMDD	① Transfer start instruction execution ② Detection of the bus release signal (REL) ③ RESET signal input ④ CTXE = CRXE = 0	Detection of the command signal (CMDD)

Acknowledge trigger bit (W)

ACKT	When the ACKT bit is set after completion of transfer, $\overline{\text{ACK}}$ is output in phase with the next $\overline{\text{SCK}}$. After the $\overline{\text{ACK}}$ signal is output, the ACKT bit is automatically cleared. Caution: ① Never set this bit to 1 before completion of serial transfer. ② ACKT cannot be cleared by software. ③ Set ACKE to 0 before setting ACKT.

Fig. 7-11 Format of SBIC Register (3/3)

Acknowledge enable bit (R/W)

ACKE	0	Disables automatic output of the acknowledge signal.	
	1	Before transfer completion	Outputs $\overline{\text{ACK}}$ in phase with the ninth $\overline{\text{SCK}}$ pulse.
		After transfer completion	Outputs $\overline{\text{ACK}}$ in phase with $\overline{\text{SCK}}$ immediately after set instruction execution.

Acknowledge detection flag (R)

ACKD	Conditions for being cleared (ACKD = 0)	Conditions for being set (ACKD = 1)
	① Start of transfer ② RESET signal input ③ CTXE = CRXE = 0 ④ Detection of bus release (in slave mode only)	Detection of the acknowledge signal ($\overline{\text{ACK}}$)

Busy enable bit (R/W)

BSYE	0	① Disables automatic output of the busy signal. ② Stops busy signal output on a falling edge of $\overline{\text{SCK}}$ immediately after clear instruction execution.
	1	Outputs the busy signal on a falling edge of $\overline{\text{SCK}}$ after the acknowledge signal.

Remark: (R): Read only

(W): Write only

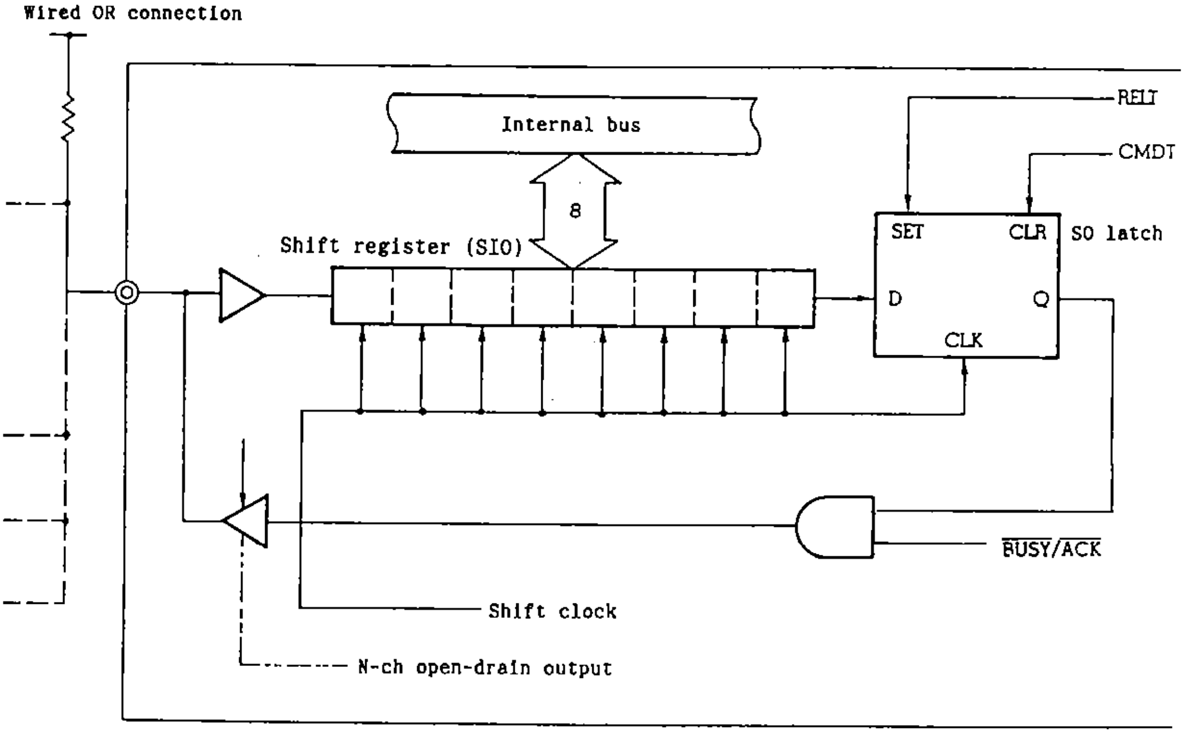
(R/W): Read and write

7.5.5 Shift register (SIO)

The SIO register is a shift register for parallel-serial conversion.

Data written into SIO is output onto the serial data bus, and data is loaded into SIO from the serial data bus. Figure 7-12 shows the configuration of shift register and its peripheral circuitry.

Fig. 7-12 Configuration of Shift Register and Peripheral Circuitry



In the SBI data bus configuration, an input pin is used also as an output pin. The output pin is configured as an N-ch open-drain output, and has a wired OR configuration with an external pull-up resistor. So a device attempting to receive data must load FFH into shift register (SIO) or disable send operation.

7.6 Communication Operation and Signals

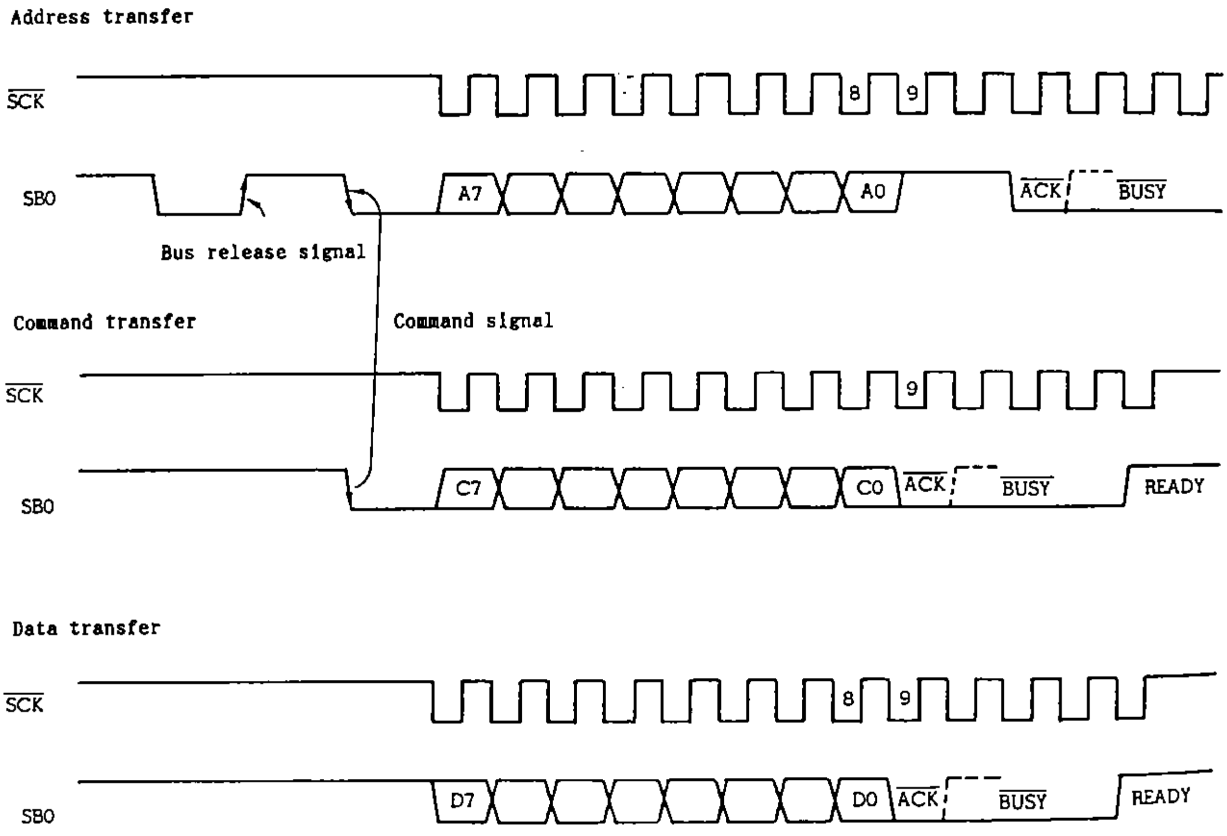
This section explains the formats of serial data, and the functions of signals used.

Serial data transferred in the SBI mode is classified into three types: address, data, and command. Serial data makes up one frame according to the following format:

(Bus release signal) + (command signal) + 8-bit data + $\overline{\text{ACK}}$ + ($\overline{\text{BUSY}}$)

Figure 7-13 shows the transfer timing of addresses, data, and commands.

Fig. 7-13 Timing of SBI Transfer



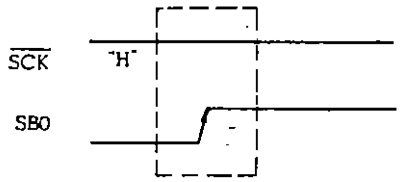
The bus release signal and command signal are output by the master, and $\overline{\text{BUSY}}$ is output by a slave. $\overline{\text{ACK}}$ can be output by either the master or a slave. (The master or a slave that receives 8-bit data usually outputs $\overline{\text{ACK}}$.)

The serial clock is output by the master for a period of time from the start of 8-bit data transfer to the clearing of $\overline{\text{BUSY}}$.

7.6.1 Bus release signal (REL)

The bus release signal is the SB0 line signal going low to high when the $\overline{\text{SCK}}$ line is high (the serial clock is not output). The bus release signal is output by the master.

Fig. 7-14 Bus Release Signal

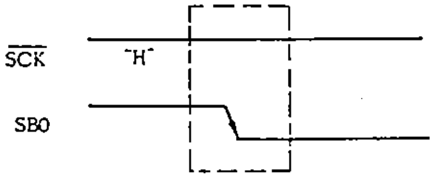


The bus release signal indicates that the master is to send an address to slaves. The slaves contain hardware to detect the bus release signal.

7.6.2 Command signal (CMD)

The command signal is the SB0 line signal going from high to low when the $\overline{\text{SCK}}$ line is high (the serial clock is not output). The command signal is output by the master.

Fig. 7-15 Command Signal

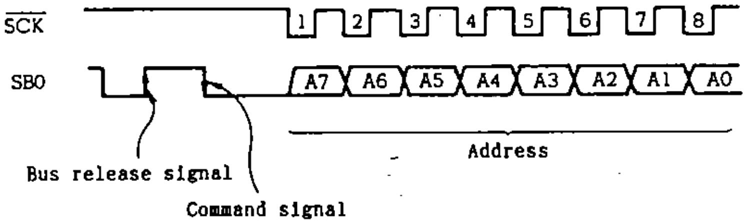


The slaves contain hardware to detect the command signal.

7.6.3 Address

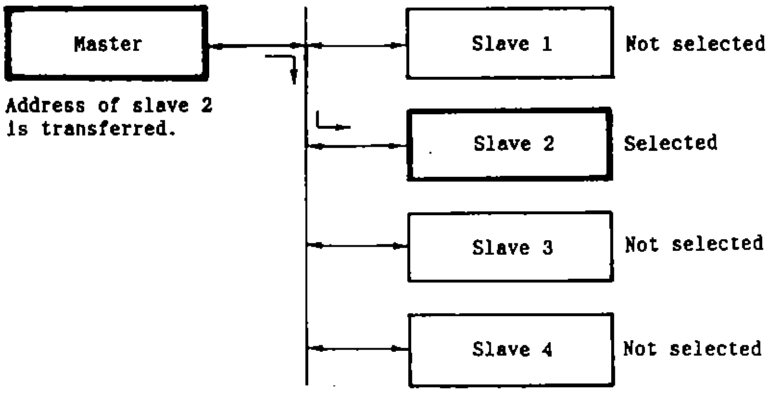
An address is 8-bit data and is output by the master to the connected slaves to select a particular slave.

Fig. 7-16 Address



Eight-bit data after the bus release signal and command signal is defined as an address. A slave detects this condition by hardware, and checks whether the 8-bit data matches the number assigned to the slave (slave address) by hardware or software. If the 8-bit data matches the slave address, that slave is selected. The selected slave continues to communicate with the master until disconnection is directed by the master.

Fig. 7-17 Slave Selection Using an Address



7.6.4 Command data

The master sends commands to the slave selected by sending an address. The master also transfers data to and from the slave.

Fig. 7-18 Command

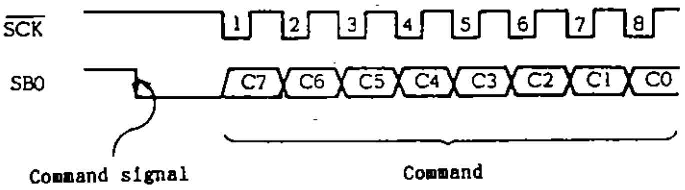
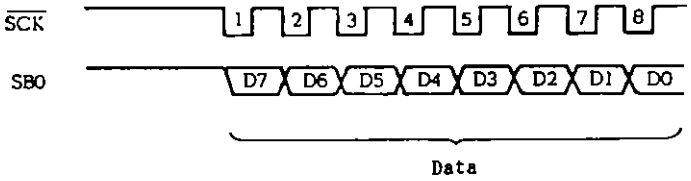


Fig. 7-19 Data



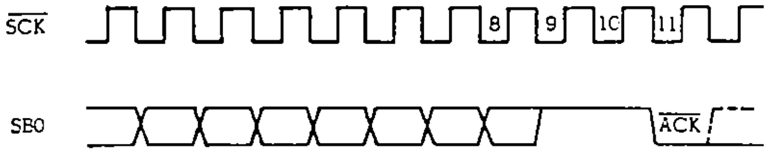
Eight-bit data after the command signal is defined as a command. Eight-bit data with no preceding command signal is defined as data. The usage of commands and data can be arbitrarily determined according to communication specifications.

7.6.5 Acknowledge signal ($\overline{\text{ACK}}$)

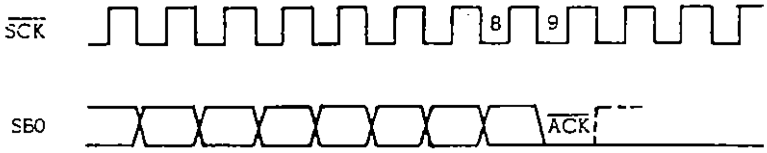
The acknowledge signal confirms the reception of data transferred between a sender and receiver.

Fig. 7-20 Acknowledge Signal

[When output in phase with the eleventh clock of $\overline{\text{SCK}}$]



[When output in phase with the ninth clock of $\overline{\text{SCK}}$]



The acknowledge signal is a one-shot pulse output on a falling edge of $\overline{\text{SCK}}$ after 8-bit data transfer.
This signal may be synchronized with any clock of $\overline{\text{SCK}}$.

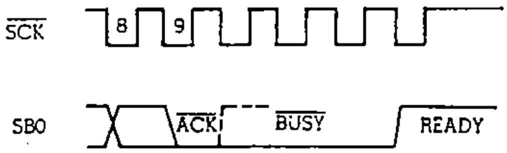
The sender checks if the receiver returns the acknowledge signal after 8-bit data transfer. If the acknowledge signal is not returned after a specified period of time, the sender can assume that the reception failed.

7.6.6 Busy signal ($\overline{\text{BUSY}}$) and ready signal (READY)

A slave uses the busy signal to inform the master that the slave is not ready yet for data transfer.

A slave uses the ready signal to inform the master that the slave is ready for data transfer.

Fig. 7-21 Busy Signal and Ready Signal



In the SBI mode, a slave pulls the SBO line low to inform the master that the slave is busy.

The busy signal is output after the acknowledge signal output by the master or slave. The busy signal is set or cleared on a falling edge of $\overline{\text{SCK}}$. When the busy signal is cleared, the master automatically stops serial clock ($\overline{\text{SCK}}$) output.

The master can restart transfer when the busy signal is cleared and the ready signal is set.

7.6.7 Signals

Figures 7-22 through 7-26 show the various signals generated in the SBI mode, and SBIC flag operation. Table 7-2 lists the signals used in the SBI mode.

Fig. 7-22 Operation of RELT, CMDT, RELD, and CMDD

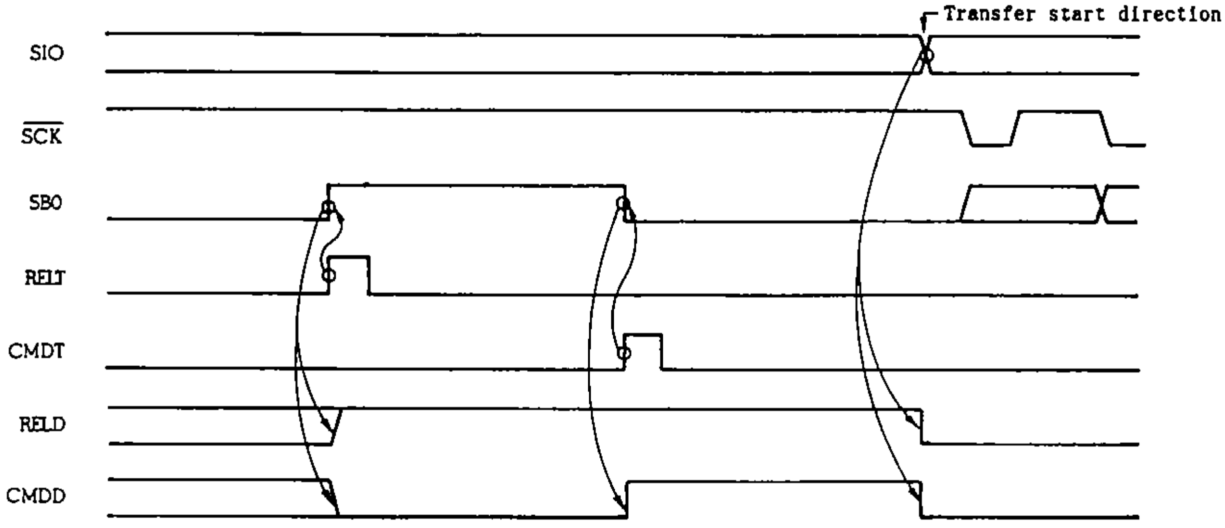
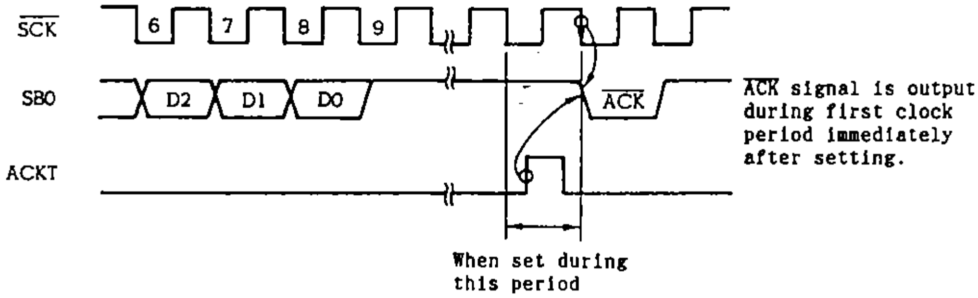
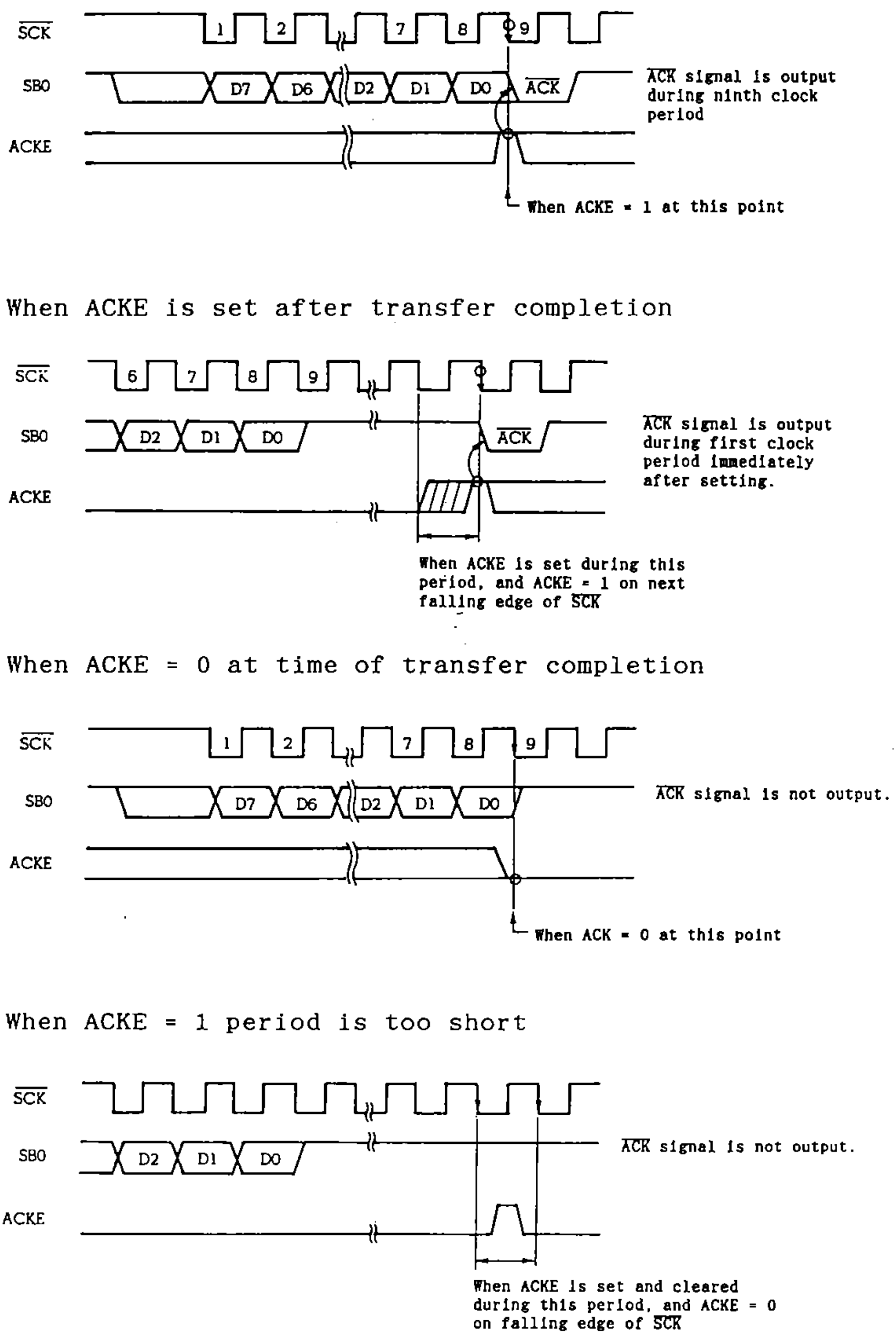


Fig. 7-23 Operation of ACKT


SCK
6
7
8
9
SBO
D2
D1
D0
ACK
ACKT
ACK signal is output during first clock period immediately after setting.
When set during this period

Caution: Do not set ACKT before transfer completion.

Fig. 7-24 Operation of ACKE



(a) When ACKE = 1 at time of transfer completion

SCK
SBO
ACKE
1
2
7
8
9
D7
D6
D2
D1
D0
ACK
ACK signal is output during ninth clock period
When ACKE = 1 at this point

(b) When ACKE is set after transfer completion

SCK
SBO
ACKE
6
7
8
9
D2
D1
D0
ACK
ACK signal is output during first clock period immediately after setting.
When ACKE is set during this period, and ACKE = 1 on next falling edge of SCK

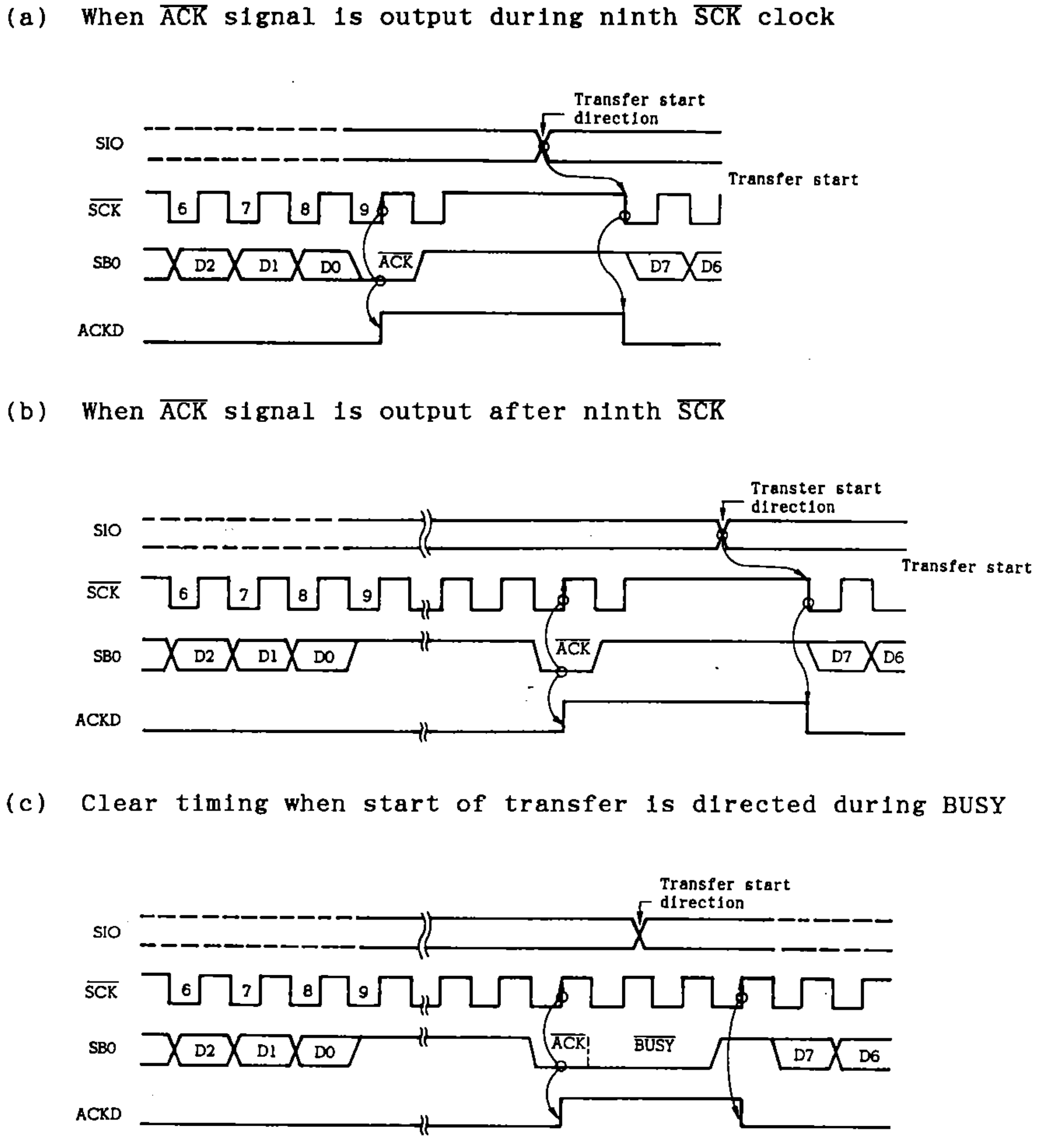
(c) When ACKE = 0 at time of transfer completion

SCK
SBO
ACKE
1
2
7
8
9
D7
D6
D2
D1
D0
ACK signal is not output.
When ACK = 0 at this point

(d) When ACKE = 1 period is too short

SCK
SBO
ACKE
D2
D1
D0
ACK signal is not output.
When ACKE is set and cleared during this period, and ACKE = 0 on falling edge of SCK

Fig. 7-25 Operation of ACKD



(a) When $\overline{\text{ACK}}$ signal is output during ninth $\overline{\text{SCK}}$ clock

(b) When $\overline{\text{ACK}}$ signal is output after ninth $\overline{\text{SCK}}$

(c) Clear timing when start of transfer is directed during BUSY

Fig. 7-26 Operation of BSYE

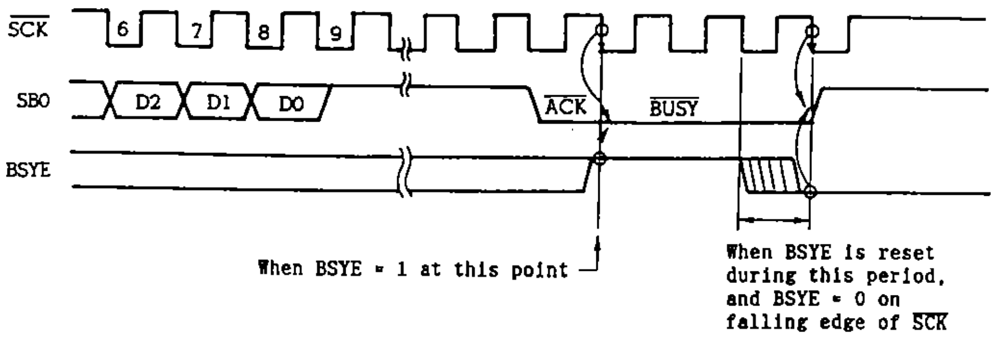

SCK
SBO
BSYE
6
7
8
9
D2
D1
D0
ACK
BUSY
When BSYE = 1 at this point
When BSYE is reset during this period, and BSYE = 0 on falling edge of SCK

Table 7-2 Various Signals Used in SBI Mode (1/5)

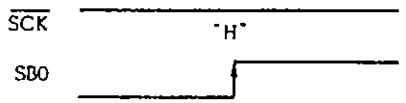
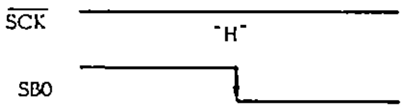
Signal name	Output device	Definition	Timing chart	Condition for output	Flag operation	Meaning of signal
Bus release signal (REL)	Master	Rising edge of SBO when $\overline{\text{SCK}}$ = 1		. RELT is set.	. RELD is set. . CMDD is cleared.	Indicates that CMD signal will follow this signal, and send data is address.
Command signal (CMD)	Master	Falling edge of SBO when $\overline{\text{SCK}}$ = 1		. CMDT is set.	. CMDD is set.	(1) Send data after REL signal output is address. (2) Send data with no REL signal preceding is command.

Table 7-2 Various Signals Used in SBI Mode (2/5)

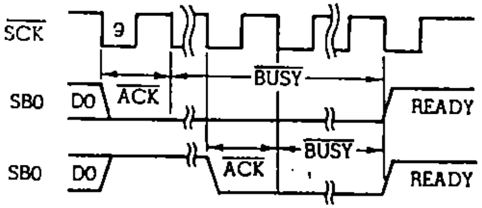
Signal name	Output device	Definition	Timing chart	Condition for output	Flag operation	Meaning of signal
Acknowledge signal ($\overline{\text{ACK}}$)	Master/ slave	Low-level signal output on SB0 during one $\overline{\text{SCK}}$ clock period after serial receive operation is completed	 $\overline{\text{SCK}}$ 9 $\overline{\text{BUSY}}$ SB0 DO $\overline{\text{ACK}}$ READY SB0 DO $\overline{\text{ACK}}$ $\overline{\text{BUSY}}$ READY	① ACKE = 1 ② ACKT is set.	. ACKD is set.	Completion of receive operation
Busy signal ($\overline{\text{BUSY}}$)	Slave	Low-level signal output on SB0 after acknowledge signal		. BSYE = 1	-	Indicates that processing is in progress, so serial send/ receive operation is impossible.

Table 7-2 Various Signals Used in SBI Mode (3/5)

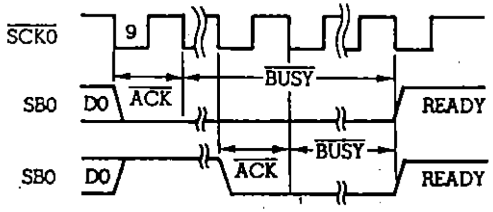
Signal name	Output device	Definition	Timing chart	Condition for output	Flag operation	Meaning of signal
Ready signal (READY)	Slave	High-level signal output on SB0 before or after serial transfer	 SCK0 9 BUSY SB0 D0 ACK READY SB0 D0 ACK BUSY READY	① BSYE = 0 ② Data write to SIO when CTXE = 1 (serial transfer start direction) *2 ③ Execution of instruction to read data from SIO when CTXE = 0 and CRXE = 1 ④ CRXE bit going from 0 to 1	-	Indicates that serial send/receive opera-is possible.

Table 7-2 Various Signals Used in SBI Mode (4/5)

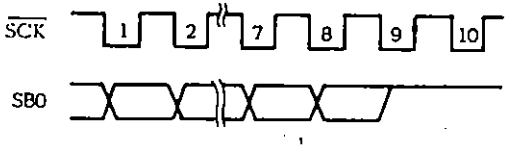
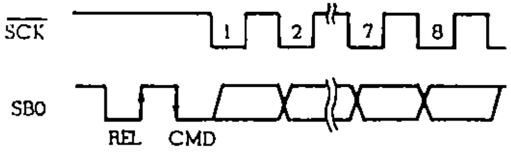
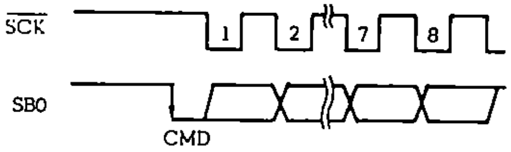
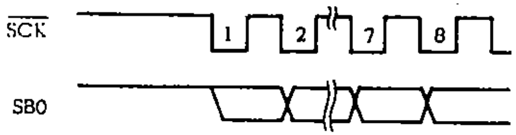
Signal name	Output device	Definition	Timing chart	Condition for output	Flag operation	Meaning of signal
Serial clock ($\overline{\text{SCK}}$)	Master	Sync clock for outputting address/command/data, $\overline{\text{ACK}}$ signal, sync $\overline{\text{BUSY}}$ signal, etc. Address/command/data are transferred during first 8 clock periods.	 $\overline{\text{SCK}}$ SBO 1 2 7 8 9 10	① Execution of instruction to write data to SIO when CTXE = 1 (serial transfer start direction) (*2)	CSIIF is set (on eighth rising edge of clock)(*1)	Timing of signal output on serial data bus
Address (A7-A0)	Master	8-bit data transferred in phase with $\overline{\text{SCK}}$ after REL signal and CMD signal are output	 $\overline{\text{SCK}}$ SBO 1 2 7 8 REL CMD	② Execution of instruction to read data from SIO when CTXE = 0 and CRXE = 1		Address value of slave device on serial bus

Table 7-2 Various Signals Used in SBI Mode (5/5)

Signal name	Output device	Definition	Timing chart	Condition for output	Flag operation	Meaning of signal
Command (C7-C0)	Master	8-bit data transferred in phase with $\overline{SCK}$ after only CMD signal is output, with REL signal not output		③ CRXE bit going from 0 to 1		Direction and message to slave device
Data (D7-D0)	Master/ slave	8-bit data transferred in phase with $\overline{SCK}$, with REL and CMD signals not output				Data processed by slave or master

*1 When WUP = 0, CSIIF is always set on the eighth rising edge of $\overline{SCK}$
When WUP = 1, CSIIF is set on the eighth rising edge of $\overline{SCK}$ only when an address is received.

*2 In data send/receive operation, transfer operation is started after the state is changed from the $\overline{BUSY}$ state to the READY state.

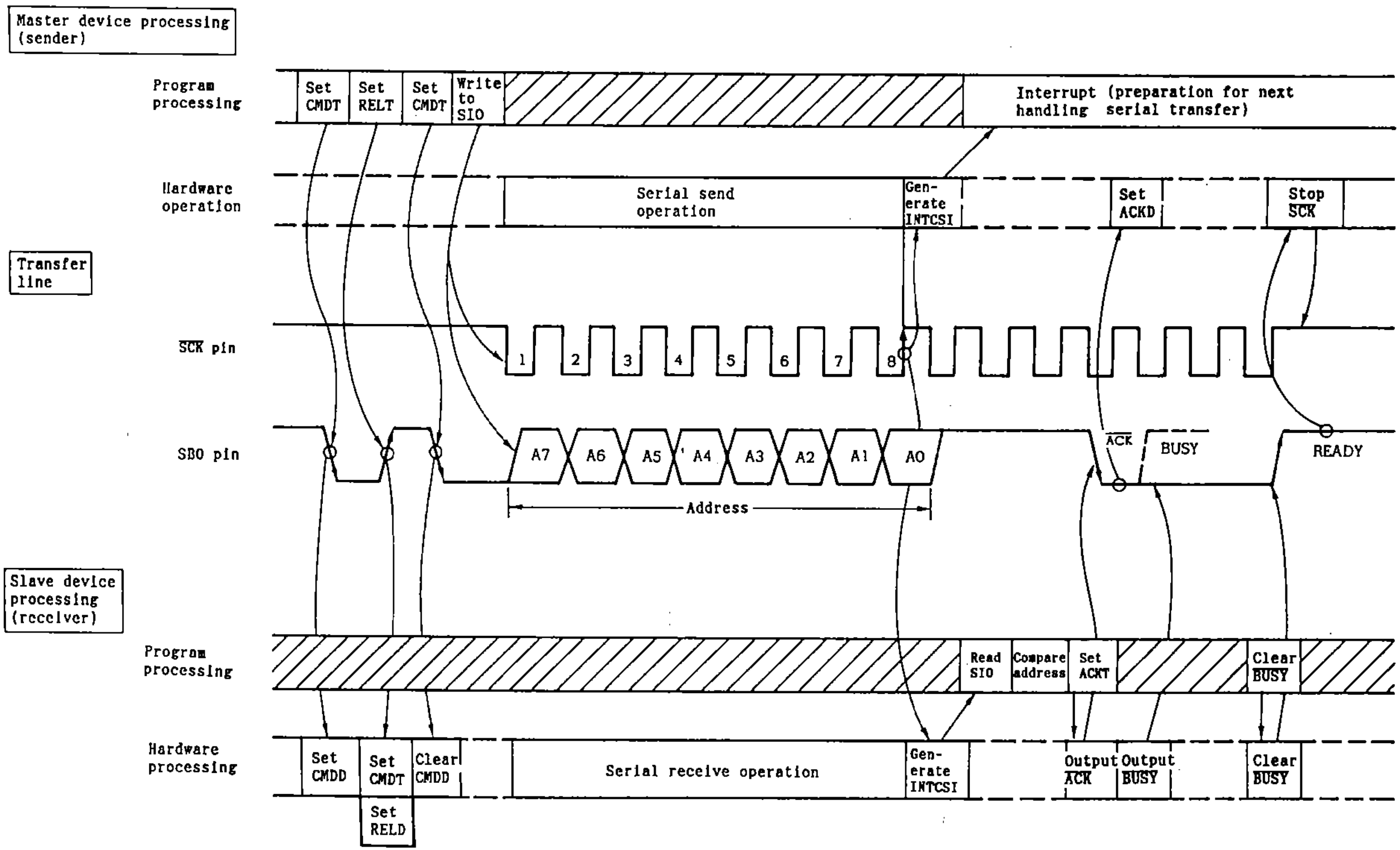
7.6.8 Communication operation

In the SBI mode, the master selects a desired slave device from multiple slave devices by outputting the address of the desired slave onto the serial bus.

After selecting a slave device subject to communication, the master and slave device exchange commands and data with each other to perform serial communication.

Figures 7-27 through 7-30 show the timing charts of data communication operations.

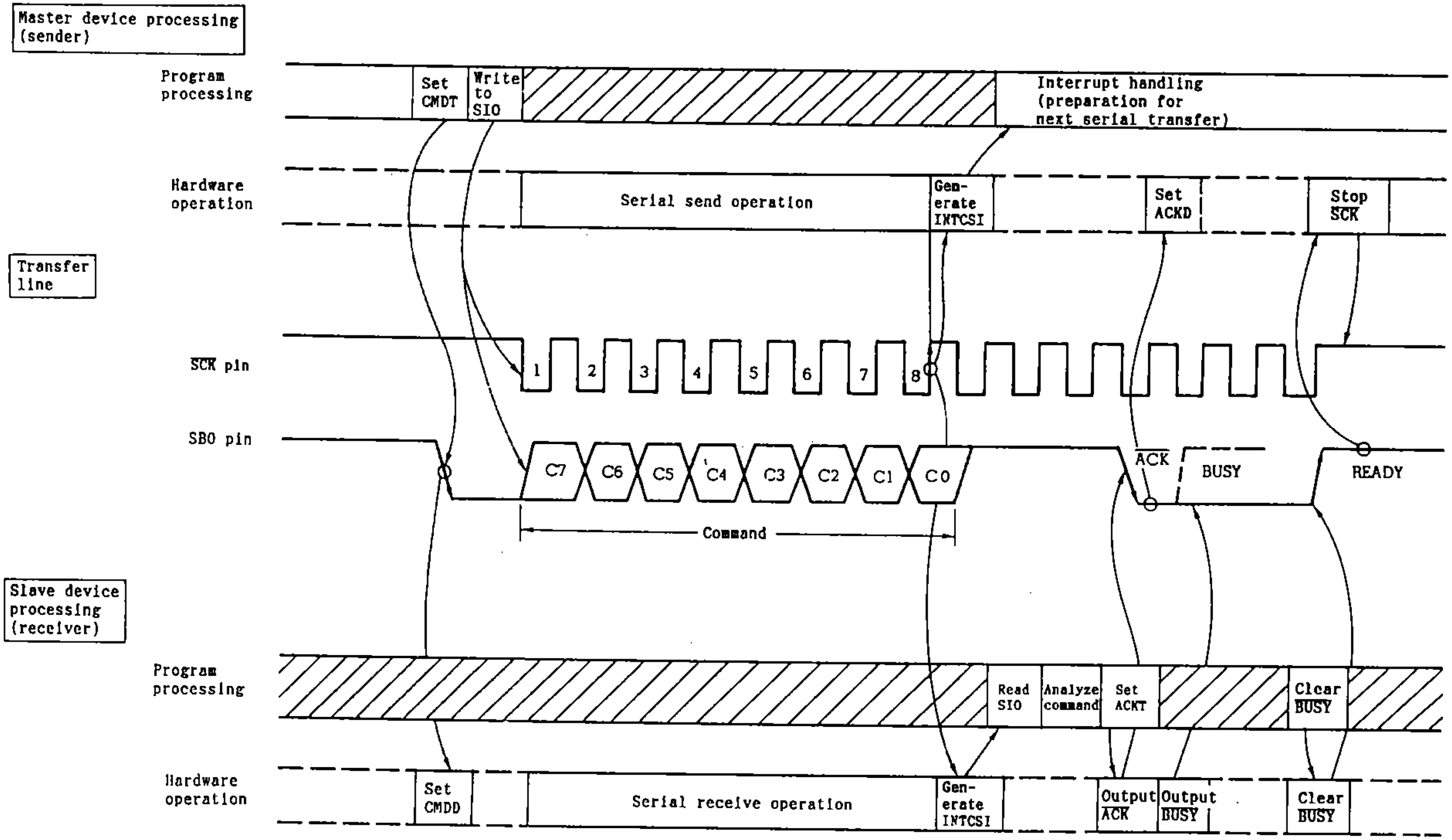
Fig. 7-27 Address Transfer Operation from Master Device to Slave Device



Remark: This timing is based on the following conditions:

- The master is allowed to perform send operation only.
- The slave is allowed to perform receive operation only. ACKE = 0 and BSYE = 1

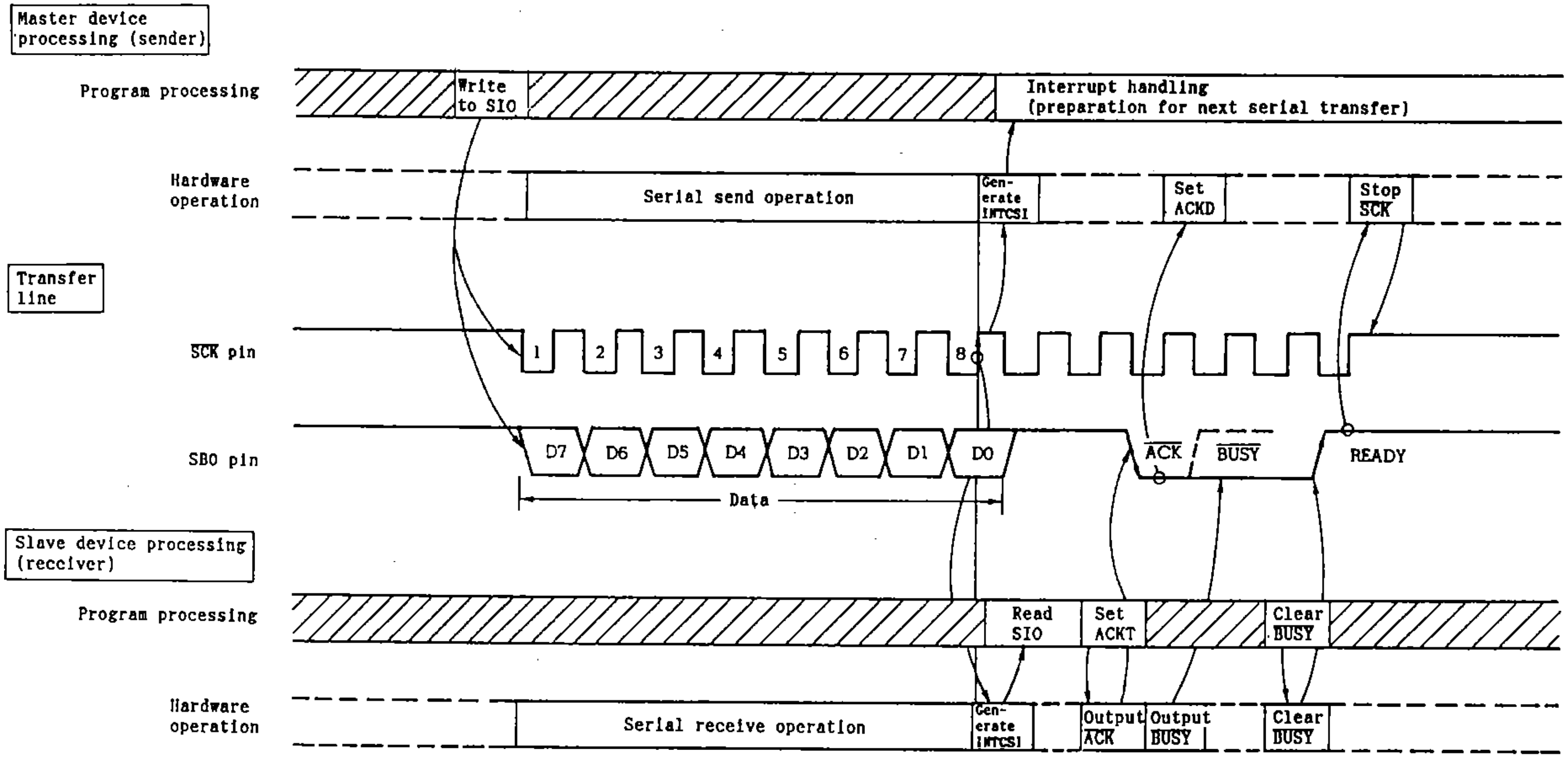
Fig. 7-28 Command Transfer Operation from Master Device to Slave Device



Remark: This timing is based on the following conditions:

- The master is allowed to perform send operation only.
- The slave is allowed to perform receive operation only. ACKE = 0 and BSYE = 1

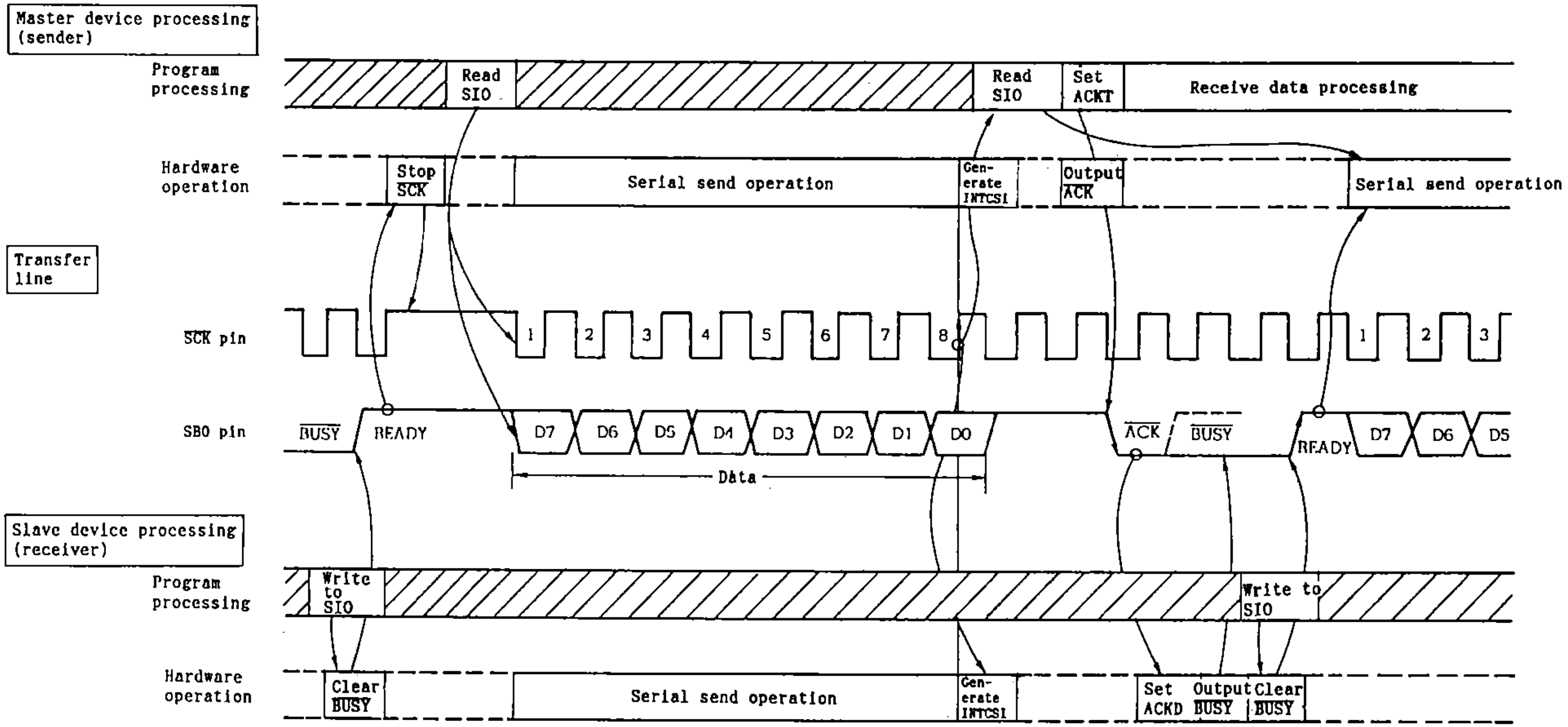
Fig. 7-29 Data Transfer Operation from Master Device to Slave Device



Remark: This timing is based on the following conditions:

- The master is allowed to perform send operation only.
- The slave is allowed to perform receive operation only. ACKE = 0 and BSYE = 1

Fig. 7-30 Data Transfer Operation from Slave Device to Master Device



Remark: This timing is based on the following conditions:

- The master is allowed to perform send operation only. ACKE = 0
- The slave is allowed to perform receive operation only. BSYE = 1

7.6.9 Clearing the busy signal

The condition for clearing the busy signal depends on whether send and/or receive operation is enabled. This takes high-speed transfer operation using an SBI macro service into consideration. Table 7-3 indicates the conditions for clearing $\overline{\text{BUSY}}$.

Table 7-3 Conditions for Clearing $\overline{\text{BUSY}}$

Send/receive enabled or disabled		$\overline{\text{BUSY}}$ clearing condition
CTXE	CRXE	
0	0	None
0	1	BSYE ← 0 or SIO read access
1	0	BSYE ← 0 or SIO write access(*)
1	1	

* If the next operation is a receive operation, FFH is to be written to SIO.

7.6.10 Wake-up setting operation

If WUP is set to 1 during busy state, the wake-up state is set immediately after the ready state is entered.

In the wake-up state, the interrupt (INTCSI) occurs only when an address is received, and the acknowledge signal ($\overline{\text{ACK}}$) is not detected.

7.6.11 Starting send/receive operation

Send/receive operation is started in the same way as for clearing the busy signal. Even when the start of send/receive operation is indicated, the start is held while the slave device is outputting the busy ($\overline{\text{BUSY}}$) signal. The operation is started when the busy signal is cleared.

7.7 Cautions

(1) To switch between the master and slave, a pull-up resistor is required also for the serial clock line ($\overline{\text{SCK}}$) because $\overline{\text{SCK}}$ input/output switching is performed between the master and slave asynchronously.

(2) Do not set ACKT before transfer operation is completed.

(3) Do not change the state from CTXE = 0 and CRXE = 1 to CTXE = 1 and CRXE = 0 or from CTXE = 1 and CRXE = 0 to CTXE = 0 and CRXE = 1 using a single operation. If that is attempted, the serial clock counter malfunctions to terminate the first communication after the change before 8 bits are transferred. Use the following two instructions to perform the above change:

Example: To change the state from CTXE = 1 and CRXE = 0 to CTXE =0 and CRXE = 1

```
CLR1 CTXE
SET1 CRXE
```

CHAPTER 8 SUPER TIMER UNIT

The uPD78138 contains a super timer unit to facilitate digital servo control by software. The super timer unit consists of an 18-bit counter, three 16-bit timers, and two PWM output channels with a 12-bit resolution. With these timer units, various pulse signals can be output, and various pulse widths can be measured.

The uPD78138 allows the user to select either 23.4- or 46.9-kHz carrier frequency for PWM output. Selecting 46.9-kHz carrier frequency improves the response time in servo control because a small value can be specified as the time constant for the external low-pass filter for carrier elimination.

The super timer unit of the uPD78138 has many functions that facilitate VCR servo control.

8.1 Overview of the Super Timer Unit

8.1.1 Configuration of the super timer unit

Table 8-1 indicates the basic components of the super timer unit of the uPD78138.

Table 8-1 Components of the Super Timer Unit

Unit name	Timer/counter	Register	Remarks
Timer 0	16-bit timer x 1 (TM0)	16-bit compare register x 3	Contains an auxiliary 6-bit counter.
Free running counter	18-bit counter x 1 (FRC)	16-bit capture register x 3 18-bit capture register x 1	Contains a digital noise eliminator.
Timer 1	16-bit timer x 1 (TM1)	16-bit compare register x 2 16-bit capture register x 1	. Contains an auxiliary 6-bit counter. . Pulse width detection function
	7-bit count register x 1 (TM3)	7-bit compare register x 1 7-bit capture register x 1	
Timer 2	16-bit timer x 1 (TM2)	16-bit compare register x 1	
PWM output	12-bit counter x 2 (PWM0, PWM1)	16-bit modulo register x 2 (with 12 bits used)	. Selectable active level of output . Selectable carrier frequency

Figure 8-1 shows the configuration of the super timer unit.

The most significant feature of the super timer unit of the uPD78138 includes timer 0 (TM0), the free running counter (FRC), and timer 1 (TM1).

Figure 8-2 shows the configuration of these timer units. The timer units facilitate VCR index search, DC motor control, and servo control using software.

Fig. 8-1 Configuration of the Super Timer Unit

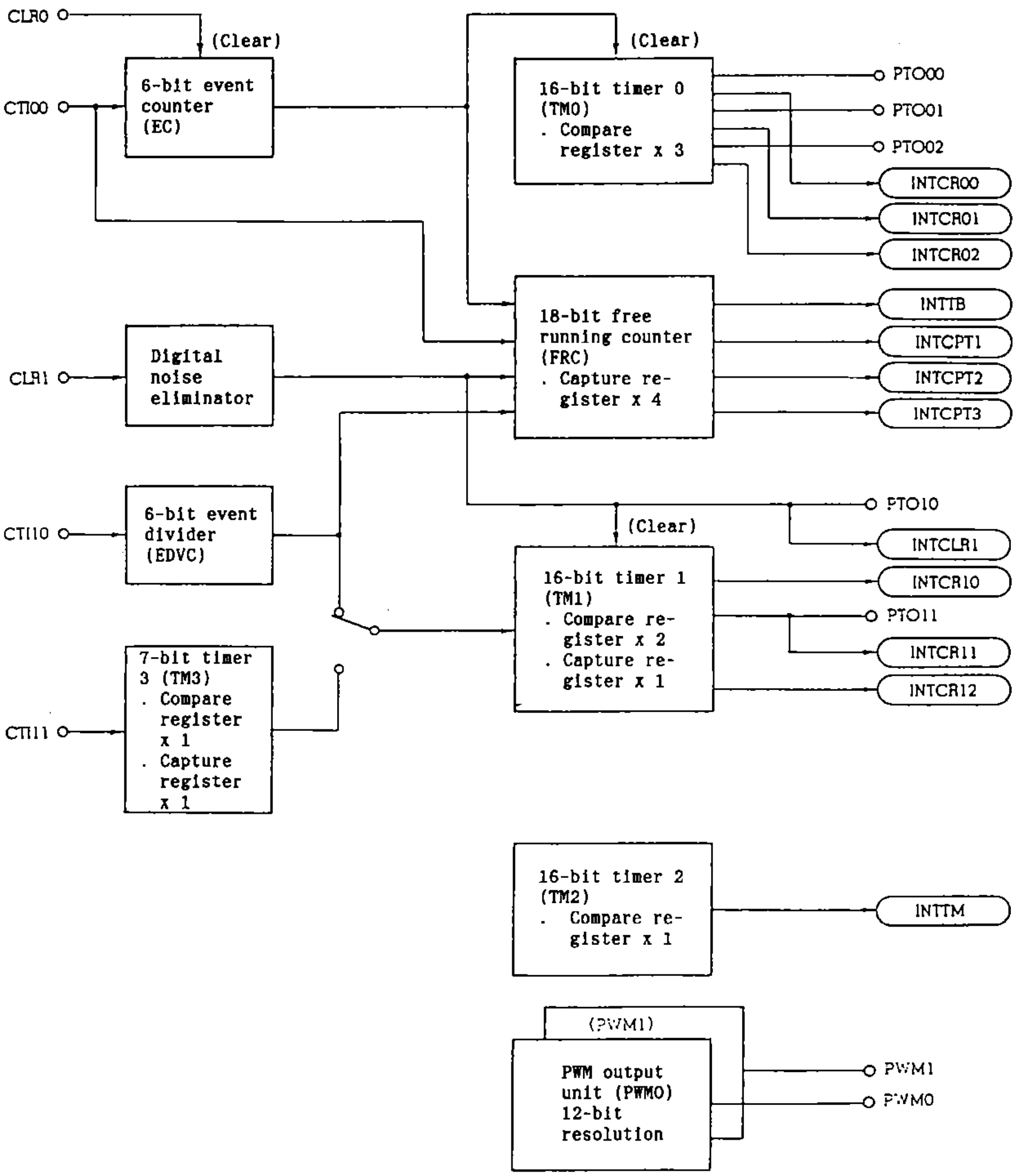
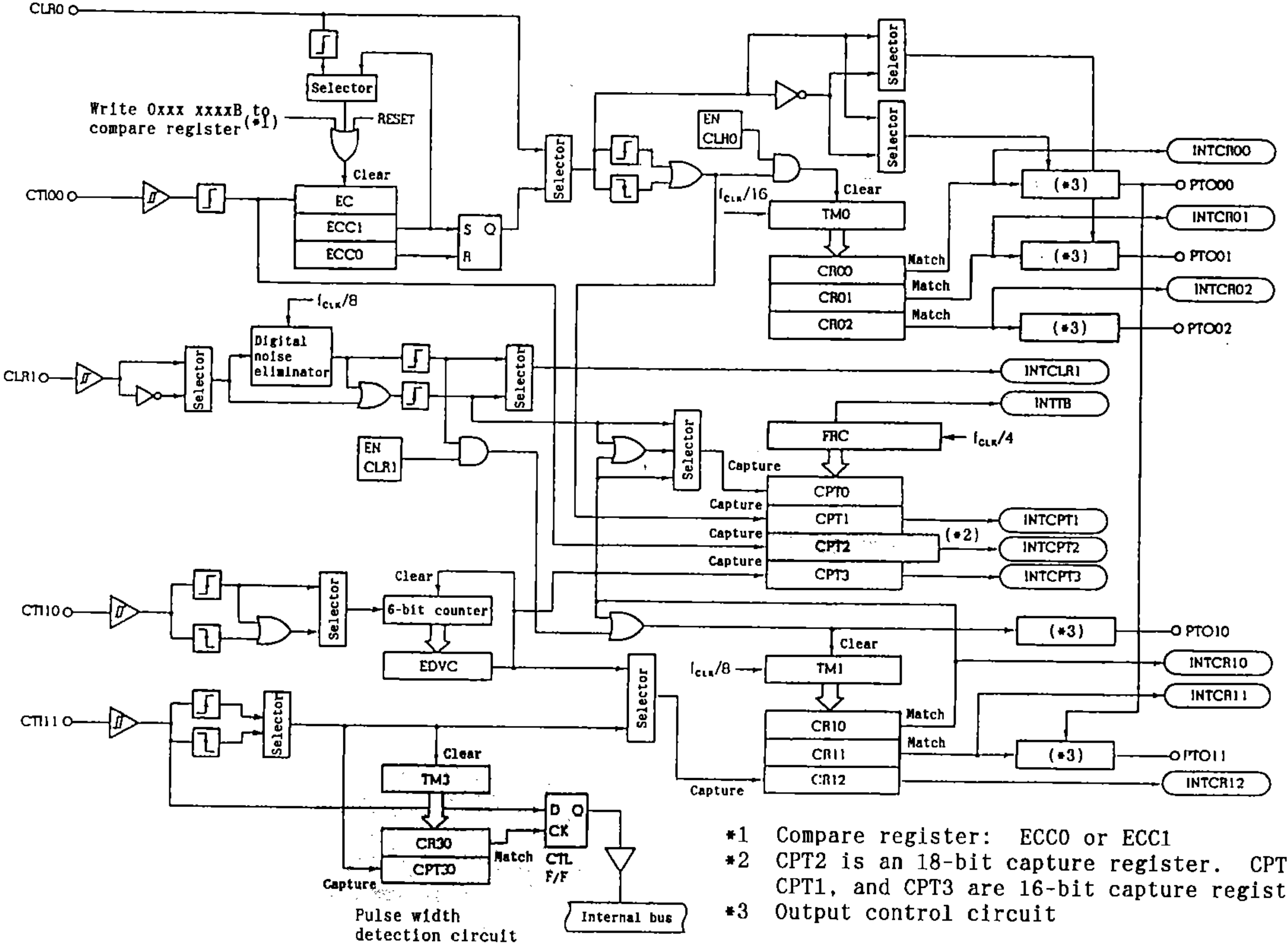


Fig. 8-2 Configuration of Timer 0, Timer 1, and Free Running Counter



*1 Compare register: ECC0 or ECC1

*2 CPT2 is an 18-bit capture register. CPT0, CPT1, and CPT3 are 16-bit capture registers.

*3 Output control circuit

8.1.2 Functional overview of the timer units

(1) Timer 0 (TM0) unit: 16-bit timer

Timer 0 is a timer unit suitable for pulse output timing control. By using an external input signal, TM0 enables the timing of pulse output to be delayed by programming. Three channels of pulse output are available, and can be used, for example, for VCR sound and video head switching signals.

(2) Timer 1 (TM1) unit: 16-bit timer

Timer 1 is a timer unit for generating a reference signal for internal processing. Timer 1 can be used for various applications such as pulse output and reference signal generation using external trigger input. Timer 1 also allows programmable delay pulse output as with timer 0. Timer 1 contains timer 3 (TM3), which is a 7-bit timer. Timer 3 can be used for external pulse width detection and period measurement.

(3) Free running counter (FRC) unit: 18-bit counter

The free running counter can be used for external pulse period measurement. This unit contains four capture registers, so that period measurements can be performed for four triggers in parallel. Since an 18-bit counter is used for this unit, a high-precision phase and speed detection is possible for a VCR drum rotating at high speed.

(4) Timer 2 (TM2) unit: 16-bit timer

Timer 2 is a general 16-bit timer unit. When the contents of the compare register match the contents of timer 2, timer 2 is automatically cleared, and functions as an interval timer to initiate an interrupt at the same time.

(5) PWM output (PWM0, PWM1) unit: 12-bit PWM

This unit is a pulse width modulation (PWM) output unit with a 12-bit resolution, and contains two channels. An active level, high or low, can be selected for each channel independently. This unit is most suitable for DC motor speed control.

Table 8-2 Resolution and Maximum Count Time of Each Timer (at 12 MHz)

Unit name	Input clock frequency	Resolution	Maximum count time
Timer 0	375 kHz (f_{CLK}/16)	2.67 us	175 ms
Timer 1	750 kHz (f_{CLK}/8)	1.33 us	87 ms
Free running counter	For CPT2L: 6 MHz (f_{CLK})	166 ns	43.7 ms
	For CPT0, CPT1, CPT2H, and CPT3: 1.5 MHz (f_{CLK}/4)	666 ns	
Timer 2	375 kHz (f_{CLK}/16)	2.67 us	175 ms
Timer 3	187.50 kHz (f_{CLK}/32)	5.33 us	0.68 ms
	46.88 kHz (f_{CLK}/128)	21.3 us	2.73 ms
	11.72 kHz (f_{CLK}/512)	85.3 us	10.9 ms
	2.93 kHz (f_{CLK}/2048)	341.3 us	43.7 ms
	External input pulse (f_{CLK}/2 at maximum)	Depends on external input pulse	External input pulse x 128

8.2 Timer 0 Unit

8.2.1 Configuration of the timer 0 unit

Figure 8-3 shows the configuration of the timer 0 unit.

The timer 0 unit consists of a 6-bit event counter (EC) and 16-bit timer 0 (TM0).

(1) Six-bit event counter (EC)

This counter generates a timer 0 clear pulse signal from signals applied to the CLR0 and CTI00 pins, and also divides a pulse signal applied to the CTI00 pin.

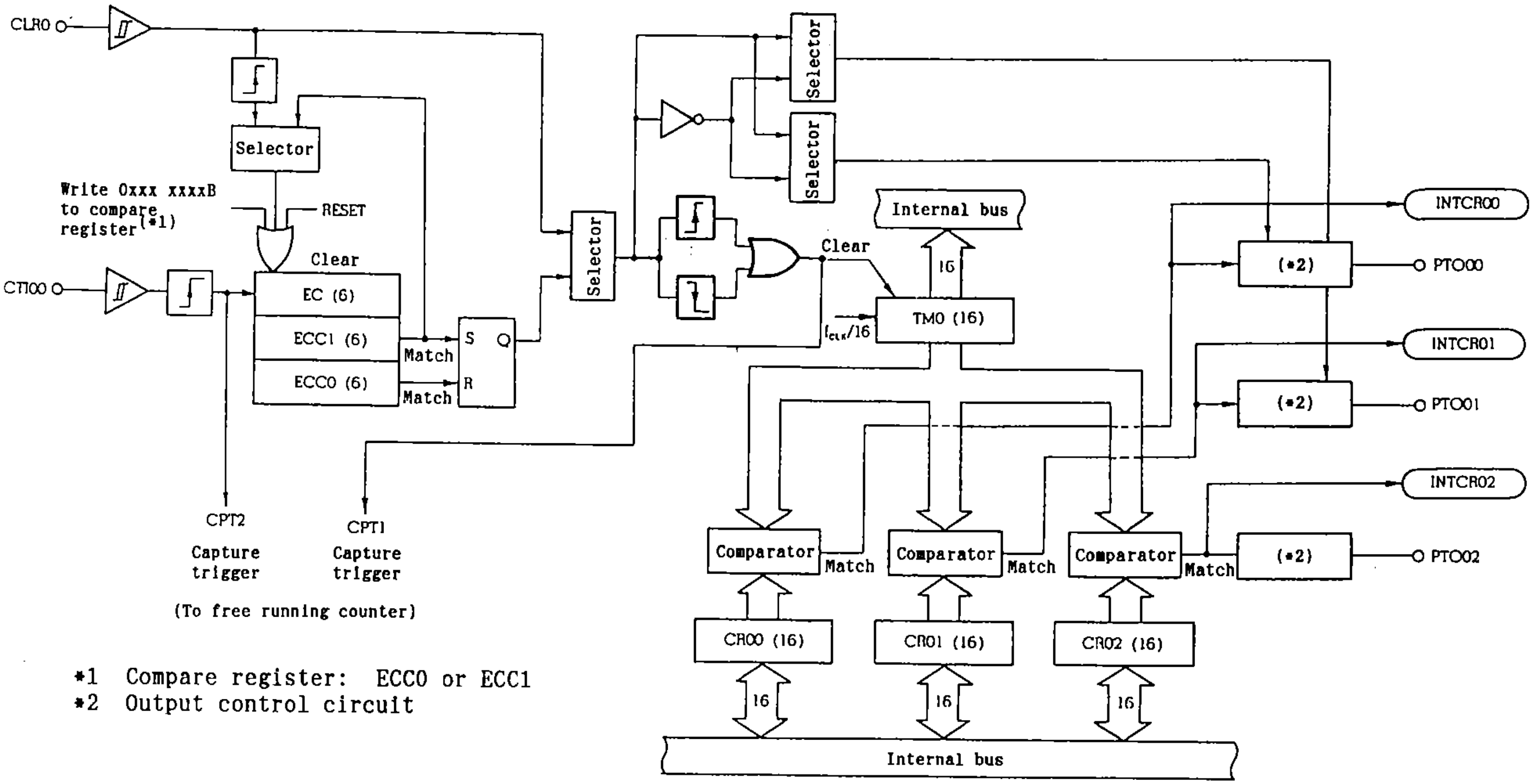
(2) Sixteen-bit timer 0 (TM0)

This timer consists of one 16-bit timer and three compare registers (CR00, CR01, CR02).

This timer has a programmable delay pulse output function, which delays, by some amount, a pulse signal applied to the CLR0 pin or a pulse signal internally generated by the event counter.

In an application to a VCR, for example, the timer 0 unit is used to generate a head switching signal from drum FG and PG signals.

Fig. 8-3 Configuration of the Timer 0 Unit



*1 Compare register: ECC0 or ECC1
*2 Output control circuit

8.2.2 Event counter (EC)

The event counter (EC) internally generates pulses from signals applied to the CLR0 and CTI00 pins.

The event counter consists of one 6-bit counter, two 6-bit compare registers, and one flip-flop for operation control.

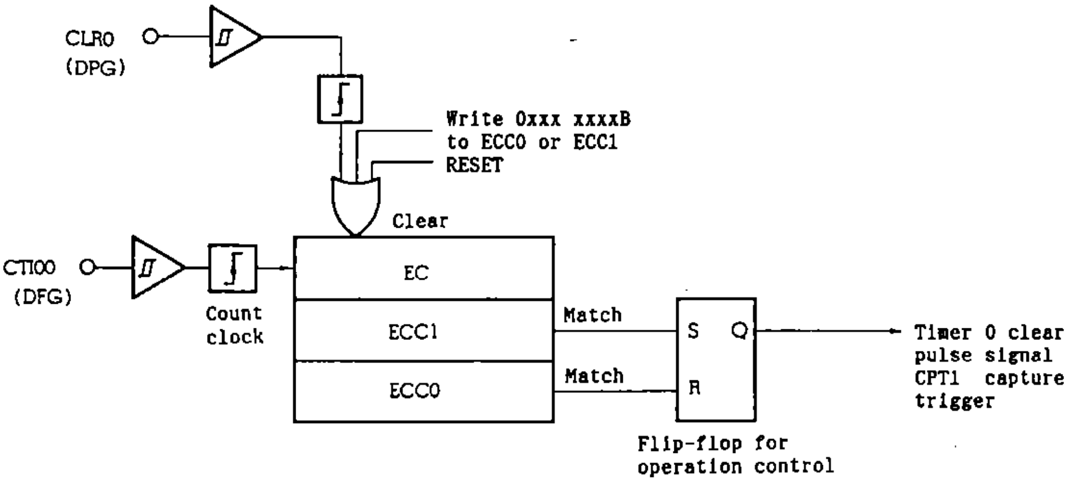
The event counter operates in one of two modes:

. Internal pulse generation mode

. General event divider mode

(1) Internal pulse generation mode

Fig. 8-4 Configuration of the Event Counter in the Internal Pulse Generation Mode



The internal pulse generation mode is used to generate an internal pulse signal from signals applied to the CLR0 and CTI00 pins.

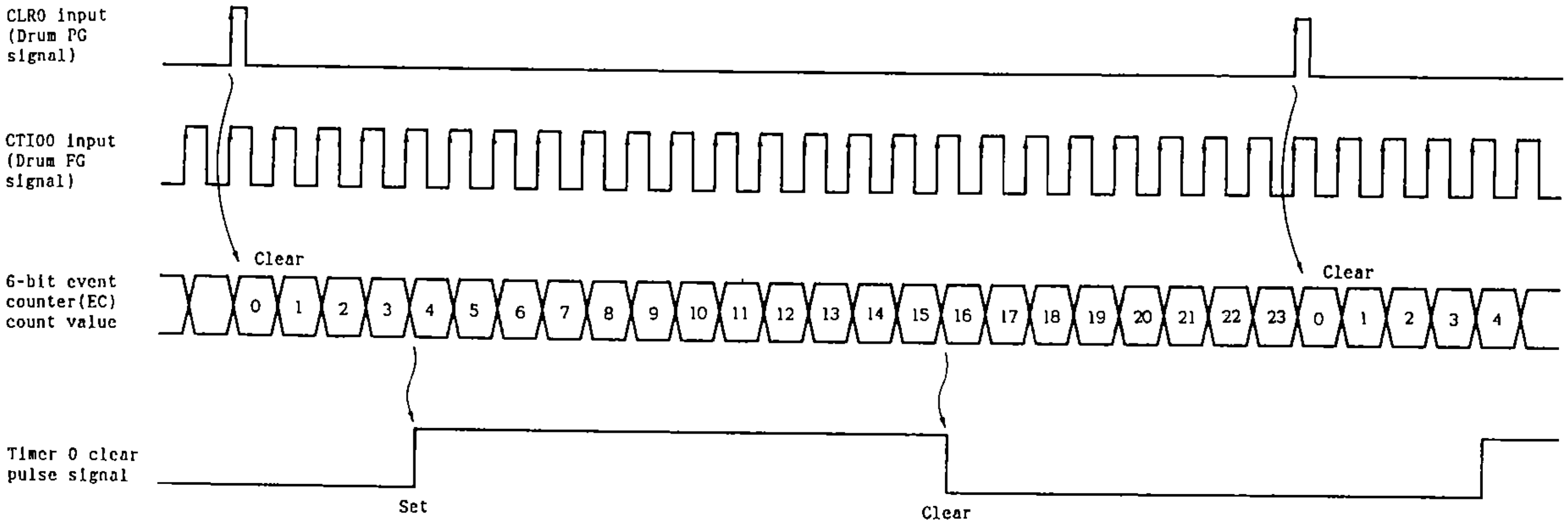
A CLR0 pin input signal clears the 6-bit event counter, and the 6-bit counter counts up with a CTI00 pin input signal.

When the value set in ECC1 matches the value of the EC, the flip-flop for operation control is set. When the value set in ECC0 matches the value of the EC, the flip-flop is reset.

This mode is used to generate a pulse signal that is set or reset according to the value set in ECC1 or ECC0 at a CLR0 pin input cycle. In an application to a VCR, for example, this mode is especially useful in internally generating a head switching signal from a drum PG signal and drum FG signal.

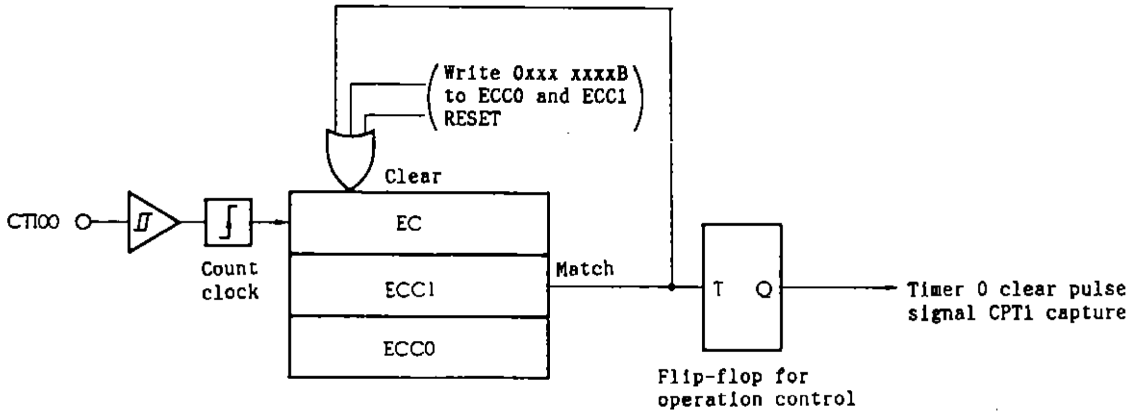
As an example, we consider the generation of a head switching signal with a 50% duty cycle from a drum motor with 24 FG signals (motor that generates 24 FG signals in one rotation). Figure 8-5 shows the operation timing. To generate, as a head switching signal, a pulse signal that is set with the fourth FG signal pulse and is reset with the 16th FG signal after a drum PG signal is applied, 03H is to be loaded into ECC1 and 0EH into ECC0.

Fig. 8-5 Operation Timing in the Internal Pulse Generation Mode (Generation of VCR Head Switching Signal)



(2) General event divider mode

Fig. 8-6 Configuration of the Event Counter in the General Event Divider Mode


Write 0xxx xxxxB to ECC0 and ECC1
RESET
Clear
CTI00
Count clock
EC
ECC1
ECC0
Match
T
Q
Flip-flop for operation control
Timer 0 clear pulse signal CPT1 capture

In the general event divider mode, the event counter is used as a general event divider to divide a CTI00 pin input pulse signal. In this case, the flip-flop operates as a T flip flop that inverts output each time the value of the EC matches the value of ECC1.

Figure 8-7 shows the operation timing of the general event divider mode when ECC1 holds 03H.

Cautions 1. In either the internal pulse generation mode or general event divider mode, the 6-bit event counter (EC) is cleared when 0xxx xxxxB (data with 0 in the highest-order bit) is written into ECC1 and ECC0.

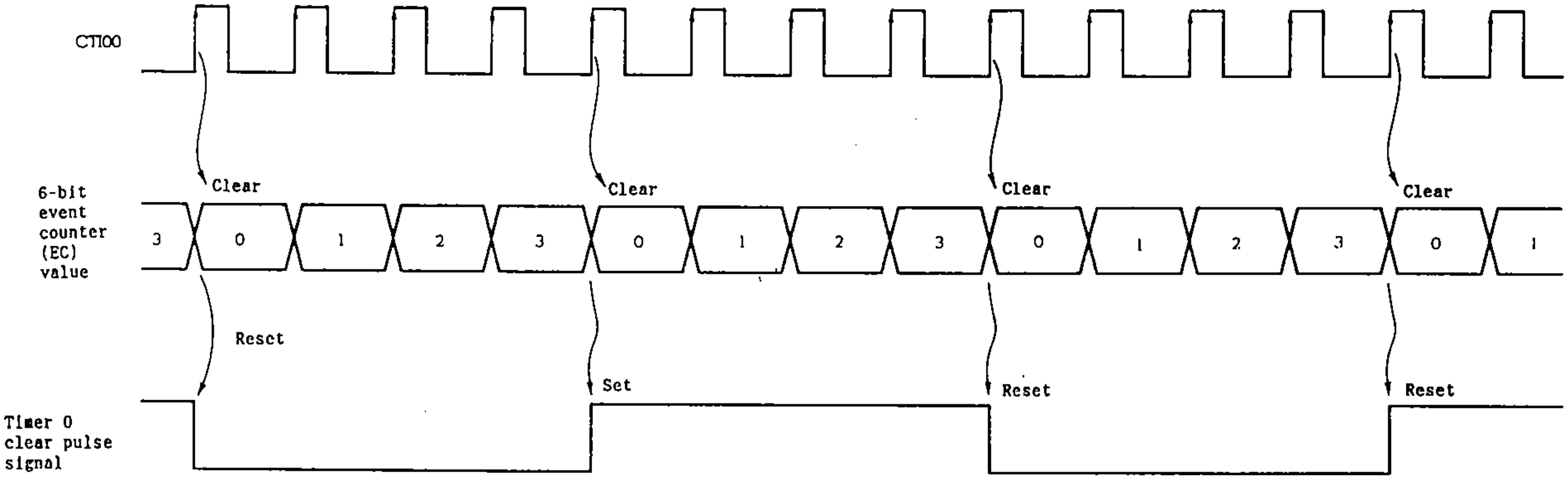
When 1xxx xxxxB (data with 1 in the highest-order bit) is written into ECC1 and ECC0, the event counter is not cleared and counting is continued.

Cautions 2. When the same value is set in ECC1 and ECC0, EC may match ECC1 and ECC0 at the same time. If this occurs, the flip-flop output is reset. (Reset with higher priority)

3. If a 0 is set in ECC1 in the general event divider mode, the timer 0 clear pulse signal is not output.

4. Changing bit 5 (ECMOD bit) of the input control register (ICR) does not affect the flip-flop controlling operation.
For example, after the operation mode of the event counter is changed from the internal pulse generation mode to the general event divider mode while the flip-flop is set, the flip-flop remains set.

Fig. 8-7 Operation Timing in the General Event Divider Mode



8.2.3 Timer 0 (TM0)

Timer 0 is a read-only 16-bit counter, and is cleared to 0000H on a rising or falling edge of a timer 0 clear pulse signal.

By using bit 7 of the input control register, the user can determine whether a CLR0 pin input signal or a pulse signal internally generated with the event counter (EC) is to be used as a timer 0 clear pulse.

The count clock of the timer can be selected from two types, $f_{CLK}/8$ and $f_{CLK}/16$.

Table 8-3 indicates the resolution and maximum count time of timer 0 when f_{CLK} = 6 MHz.

Table 8-3 Resolution and Maximum Count Time of Timer 0

Input clock	Resolution	Maximum count time
375 kHz ($f_{CLK}/16$)	2.67 us	175 ms

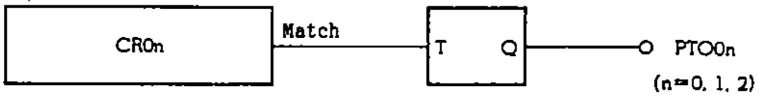
Caution: TM0 is undefined for 16 clock pulses (1 clock = 1 internal system clock: f_{CLK}) after reset release. Start TM0 on the 17th clock pulse or later.

8.2.4 Operating mode

Four modes can be used for the timer 0 output pins. The modes include the general output mode, RS mode, delay pulse output mode 1, and delay pulse output mode 2. Timer 0 output mode register (TOM0) is used to set the output mode for the timer 0 output pins (see Figure 8-14).

(1) General output mode

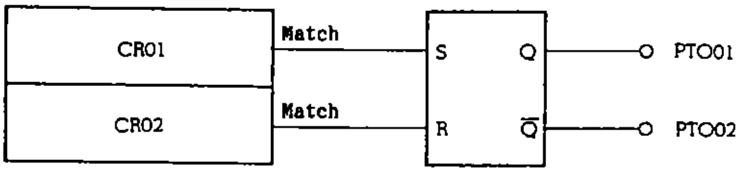
Fig. 8-8 Configuration of the Timer 0 General Output Mode



In this mode, the level of an output pin is inverted based on toggle operation each time the value of timer 0 matches the value set in the compare register.

(2) RS output mode

Fig. 8-9 Configuration of the Timer 0 RS Output Mode



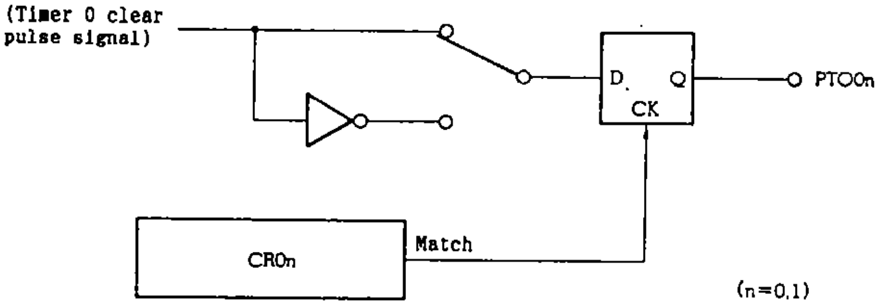
The RS output mode can be used only for PT001 and PT002.

The RS mode cannot be set for the PT000 pin. A PT001 output signal is set to 1 when the value of TM0 matches the value set in CR01. A PT001 output signal is reset when the value of TM0 matches the value set in CR02.

On PT002, the inverted signal of a PT001 output signal is output.

(3) Delay pulse output mode 1

Fig. 8-10 Configuration of Timer 0 Delay Pulse Output Mode 1



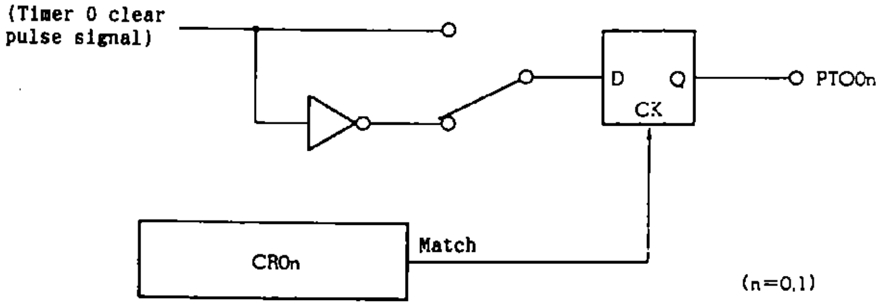
This mode latches and outputs the level of a timer 0 clear pulse signal at the time when the value of timer 0 matches the value set in CR00 or CR01.

This mode can be set for PT000 and PT001.

This mode cannot be set for PT002.

(4) Delay pulse output mode 2

Fig. 8-11 Configuration of Timer 0 Delay Pulse Output Mode 2



This mode latches and outputs the inverted signal of a timer 0 clear pulse signal at the time when the value of timer 0 matches the value set in CR00 or CR01.

This mode can be set for PT000 and PT001.

This mode cannot be set for PT002.

Table 8-4 indicates the relationships between the timer 0 output pins and output modes that can be set.

Table 8-4 Timer 0 Output Pins and Output Modes that can be Set

Output pin mode	PT000	PT001	PT002
General output mode	o	o	o
RS output mode	x	o	o
Delay pulse output mode 1	o	o	x
Delay pulse output mode 2	o	o	x

(o: Can be set
x: Cannot be set

Table 8-5 indicates the correspondence between TOMO setting values and output modes of PT00n (n = 0, 1, 2).

Table 8-5 TOMO Setting Values and Output Modes of Timer 0 Outputs

TOMO setting value	PT000	PT001	PT002
xx000000	General output mode	General output mode	General output mode
xx010110	Delay pulse output mode 1	RS output mode (Q output)	RS output mode ($\overline{Q}$ output)
xx010111	Delay pulse output mode 2	RS output mode (Q output)	RS output mode ($\overline{Q}$ output)
xx010100	General output mode	RS output mode (Q output)	RS output mode ($\overline{Q}$ output)
xx001010	Delay pulse output mode 1	Delay pulse output mode 1	General output mode
xx001011	Delay pulse output mode 2	Delay pulse output mode 1	General output mode
xx001110	Delay pulse output mode 1	Delay pulse output mode 2	General output mode
xx001111	Delay pulse output mode 2	Delay pulse output mode 2	General output mode

8.2.5 Setting timer 0 unit control registers

The operation of the timer 0 unit can be controlled by the following registers:

ICR: Input control register (for control of the event counter and clear pulse)

TMC0: Timer 0 control register (for control of timer 0 operation)

TOM0: Timer 0 output mode register (for control of the PT00n output mode)

TOC0: Timer 0 output control register (for PT00n output control)

The following figures show the formats of the registers.

Fig. 8-12 Format of Input Control Register (ICR)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
ICR	SEL CLR0	-	EC MOD	V SYNCS	SEL CLR1	-	-	-	FF50H	0x0x0xxx	W

SEL CLR1	Selection of an INTCLR1 interrupt source
0	Vertical synchronizing signal input mode
1	Composite synchronizing signal input mode

V SYNCS	Selection of a V_{sync} separation pulse width
0	Eliminates pulses of less than 5.3 us ($32/f_{CLK}$).
1	Eliminates pulses of less than 12.0 us ($72/f_{CLK}$).

ECMOD	Specification of the operating mode of the event counter
0	General event divider mode
1	Internal pulse generation mode

SEL CLR0	Selection of a timer 0 clear pulse
0	CLR0 pin input (Bypasses the 6-bit event counter.)
1	Pulse generated internally by the event counter (EC)

Caution: Rewriting SELCLR0 (bit 7 of ICR) may cause an INTCPT1 interrupt request. Clear the interrupt request flag using the instruction after rewriting.

Fig. 8-13 Format of Timer Control Register 0 (TMC0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TMC0	CS1	-	-	EN CLR1	CS0	0	0	EN CLR0	FF38H	0xx00000	W

ENCLR0	TM0 clear signal enable bit
0	Prevents TM0 from being cleared by masking a TM0 clear pulse (free running mode).
1	Clears TM0 by a clear pulse.

CS0	TM0 count control
0	Clears TM0 and stops counting.
1	Counts.

ENCLR1	TM1 clear signal enable bit
0	Prevents TM1 from being cleared by masking CLR1 input.
1	Clears TM1 by CLR1 input.

CS1	Control of TM1 counting
0	Clears TM1 and stops counting.
1	Counts.

Fig. 8-14 Format of Timer 0 Output Mode Register (TOMO)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TOMO	-	-	0(*)	MOD 020	MOD 011	MOD 010	MOD 001	MOD 000	FF58H	xx000000	W

MOD 0n1	MOD 0n0	PT00n output mode (n = 0-2)
0	0	General output mode
0	1	RS output mode
1	0	Delay pulse output mode 1
1	1	Delay pulse output mode 2

* PT002 cannot be set to delay pulse output modes 1 and 2. PT002 can be set only to the general output mode and RS mode.

Cautions 1. TOMO can only be written to in 8-bit units. No bit manipulations and read operations are allowed.

2. Writing data to the timer 0 output mode register (TOMO) places the output pins of timer 0 (PT000, PT001, and PT002) in the following state:

(1) When a pin is used in the general output mode

The pin state as set in the timer 0 output control register (TOCO) appears.

(2) When a pin is used in the RS output mode

The Q output of the RS flip-flop goes low, and the $\overline{Q}$ output goes high (reset).

(3) When a pin is used in the delay pulse output mode 1 or 2

The Q output of the D flip-flop goes low (reset).

Fig. 8-15 Format of Timer 0 Output Control Register (TOCO)

	7	6	5	4	3	2	1	0	Adress	When reset	R/W
TOCO	-	-	ENT002	ALV2	ENT001	ALV1	ENT000	ALV0	FF59H	xx000000	W

Control PT002. Control PT001. Control PT000.

ENT00n	Timer output enable/disable specification
0	Disable (always inactive level)
1	Enable

ALVn	Specification of an active level for the timer 0 output pin
0	Active low
1	Active high

Cautions 1. The register TOCO can only be written to, and allow only 8-bit manipulations. No bit manipulations and read operations are allowed.

2. Writing data to the timer 0 output control register (TOCO) places the output pins of timer 0 (PT000, PT001, and PT002) in the following state:

(1) When the bit that enables or disables timer 0 output (ENTO0n) is changed from 0 to 1

Immediately after a 1 is set in the bit, an inactive level is output on the pin. The output level is inverted by a match signal which is issued when the timer 0 contents match the compare register contents.

(2) When the bit that controls the active level of the timer 0 output pin (ALV0n) is changed from 0 to 1 or vice versa

The active level of the output pin changes as soon as ALV0n is rewritten regardless whether timer 0 is operating or not.

8.2.6 Examples of using the timer 0 unit

(1) Timer 0 delay pulse output modes

The uPD78138 allows the setting of a delay amount for delay pulse output modes 1 and 2 with a program.

In delay pulse output mode 1, the level of a timer 0 clear pulse signal is latched and output using a match between the value of timer 0 and CR00 or CR01 as a trigger.

In delay pulse output mode 1, maximum delay amount is half of the timer 0 clear pulse signal pariod, and can be expressed using the phase angle as $0 \leqq \tau < 180$ (degrees).

On the other hand, in delay pulse output mode 2, the inverted signal of a timer 0 clear pulse signal is latched using a match -between the value of timer 0 and CR00 or CR01 as a trigger.

In delay pulse output mode 2, delay amount τ of timer 0 output can be expressed using the phase angle as $180 \leqq \tau < 360$ (degrees).

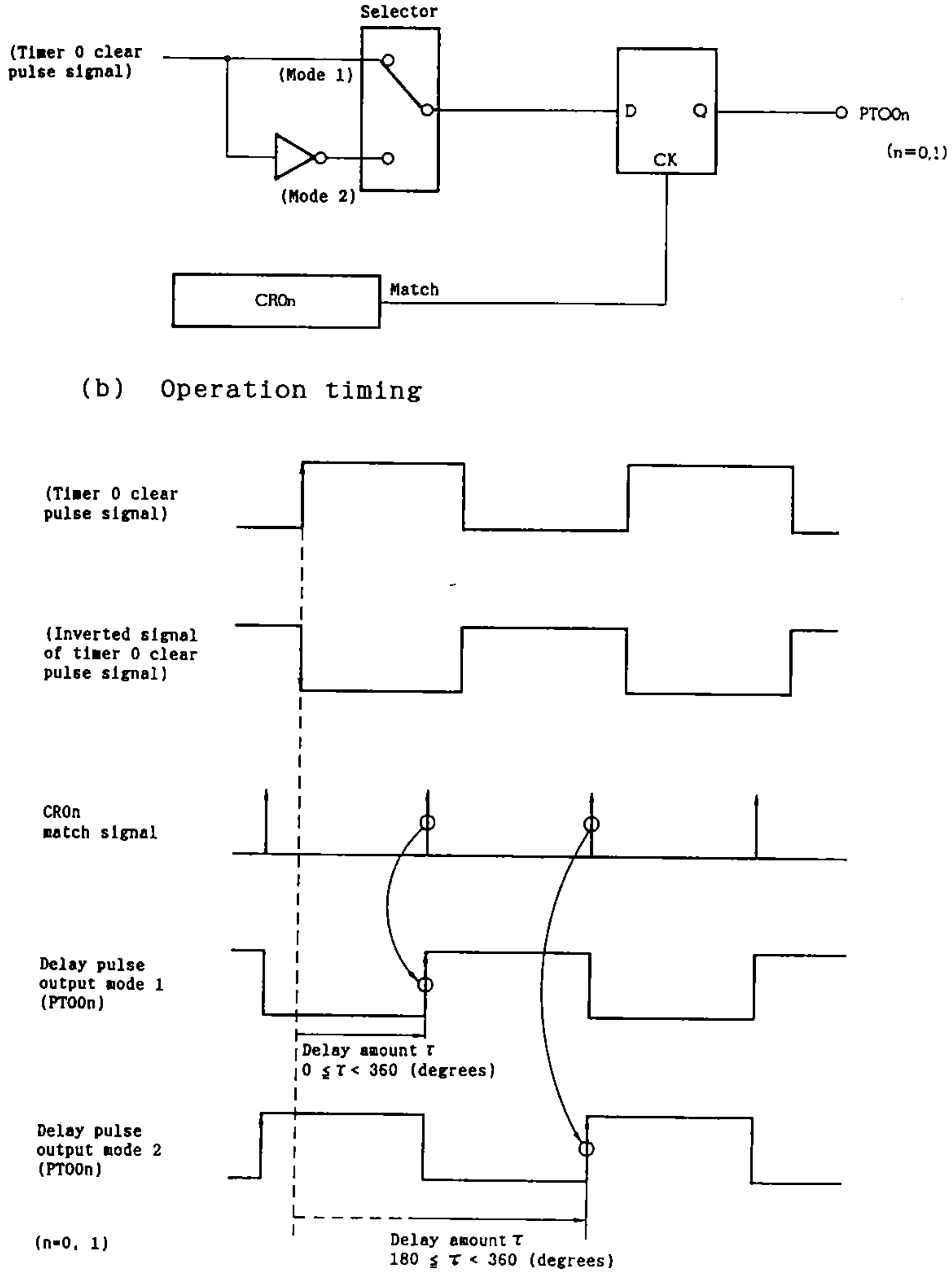
So when the timer 0 output pins (PT000, PT001) are used fro VCR head switching signals, for example, delay amount can be changed within the range $0 \leqq \tau < 360$ (degrees) by using delay pulse output mode 1 or 2.

Figure 8-16 shows the timing of these modes.

Fig. 8-16 Operation Timing of Timer 0 Delay Pulse Output Modes 1 and 2

(a) Configuration of timer 0 delay pulse output modes

(b) Operation timing


Selector
(Timer 0 clear pulse signal)
(Mode 1)
(Mode 2)
D
Q
CK
PTOOn
(n=0,1)
CR0n
Match
(Timer 0 clear pulse signal)
(Inverted signal of timer 0 clear pulse signal)
CR0n match signal
Delay pulse output mode 1 (PTOOn)
Delay amount τ
0 ≦ τ < 360 (degrees)
Delay pulse output mode 2 (PTOOn)
(n=0, 1)
Delay amount τ
180 ≦ τ < 360 (degrees)

(2) Generating VCR head switching signal

An application of timer 0 is shown in Figures 8-17 and 8-18.

In the application, the timer 0 unit is used to generate VCR head switching signals.

The drum motor used is assumed to output an FG signal on which 24 pulses are issued every motor rotation.

The CLR0 input pin receives the FG signal from the drum motor, and the CTI00 input pin receives the FG signal from the drum motor. By setting ECC of the event counter (EC) to 03H and ECC0 to 0EH, a timer 0 clear pulse signal with duty cycle of 50% can be generated so that the signal goes high on the fourth FG signal pulse after the drum PG signal is input, and the signal goes low on the 16th FG signal pulse.

A digital value equivalent to the delay d1 of the head switching signal output on the PT000 pin is set in the timer 0 compare register (CR00) beforehand.

Also, a digital value equivalent to the delay d2 of the head switching signal output on the PT001 pin is set in CR01.

As a result, two pulse signals with different delays can appear on the PT000 and PT001 output pins.

The relationships between the digital values set in the compare registers and delay amounts d1 and d2 can be expressed as follows:

$\tau d1$ = (value set in CR00) x 16/f_{CLK} [s]

$\tau d2$ = (value set in CR01) x 16/f_{CLK} [s]

(f_{CLK} = 6 MHz when the internal system clock is used, or 12 MHz when an external clock is used)

Fig. 8-17 Generating Head Switching Signals by Using the Timer 0 Unit

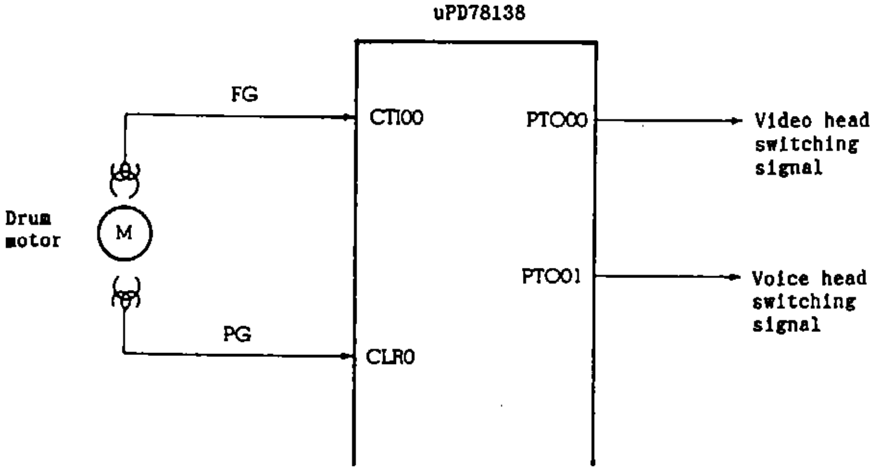
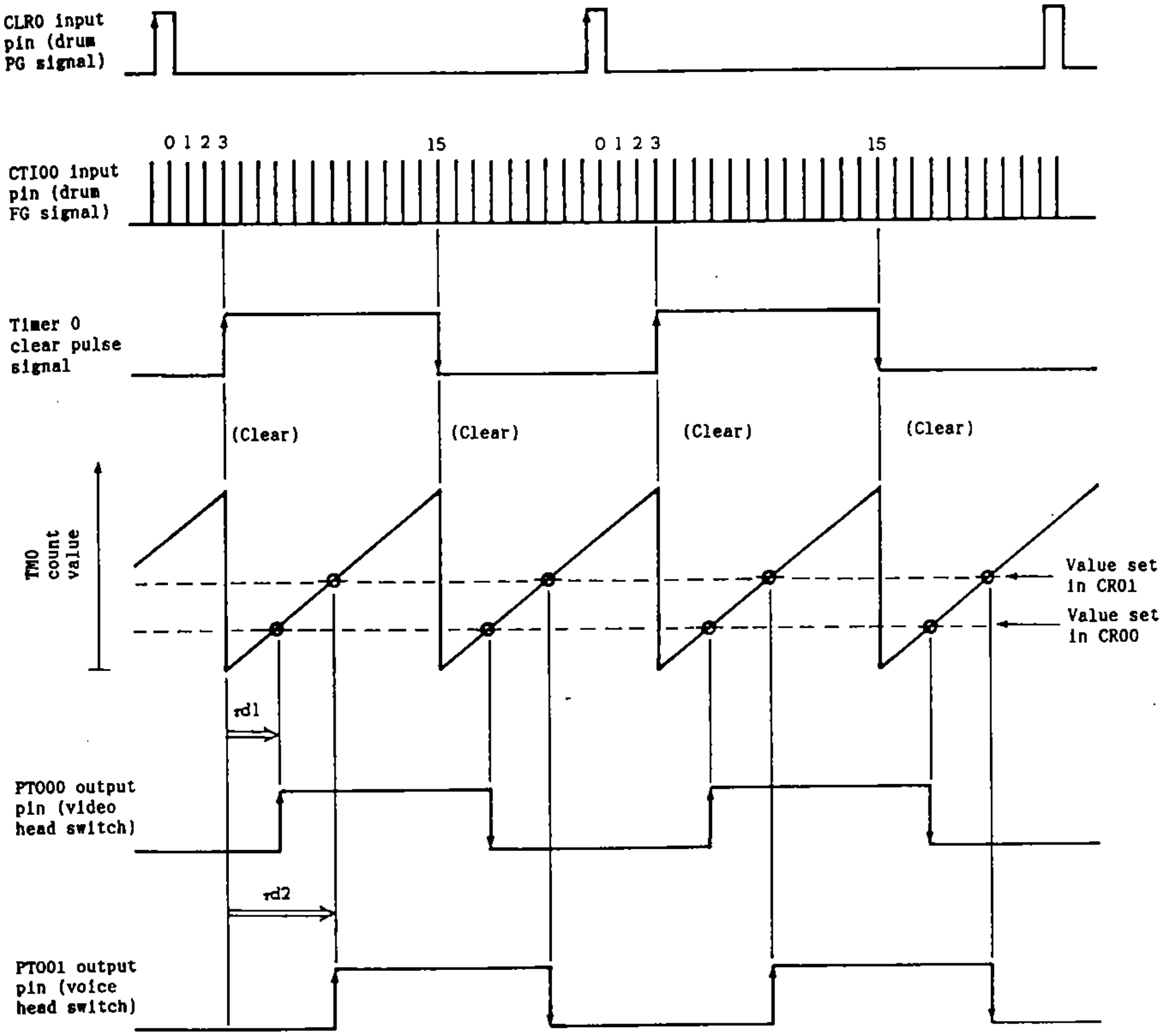


Fig. 8-18 Timing Example of Head Switching Signals with the Timer 0 Unit


CLR0 input pin (drum PG signal)
0 1 2 3
15
0 1 2 3
15
CTI00 input pin (drum FG signal)
Timer 0 clear pulse signal
(Clear)
(Clear)
(Clear)
(Clear)
TM0 count value
Value set in CR01
Value set in CR00
τd1
PTO00 output pin (video head switch)
τd2
PTO01 output pin (voice head switch)

τd1, τd2: Delays of the head switching signals

8.3 Timer 1 Unit

8.3.1 Configuration of the timer 1 unit

Figures 8-19 and 8-20 show the configuration of the timer 1 unit.

The timer 1 unit consists of a 6-bit event divider, pulse width detection circuit (TM3), and 16-bit timer (TM1).

(1) Six-bit event divider

The 6-bit event divider is used to divide a CTI10 input pulse signal.

In an application to a VCR, for example, a capstan FG (CFG) signal is applied to the CTI10 input pin. When a high CFG signal frequency is used for high-speed search, the divider can divide the CFG frequency.

(2) Pulse width detection circuit (TM3)

The pulse width detection circuit consists of a circuit for outputting a CR12 capture trigger signal upon detection of a valid edge on the CTI11 input pin and timer 3 (TM3), which is a 7-bit timer for determining the pulse width of a pulse signal applied to the CTI11 input pin.

In an application to a VCR, for example, these circuits are used as described below.

A playback control signal (PBCTL signal) is applied to the CTI11 input pin, and CR12 captures capstan phase information for capstan phase control.

On the other hand, the pulse width detection circuit determines the duty cycle of a PBCTL signal.

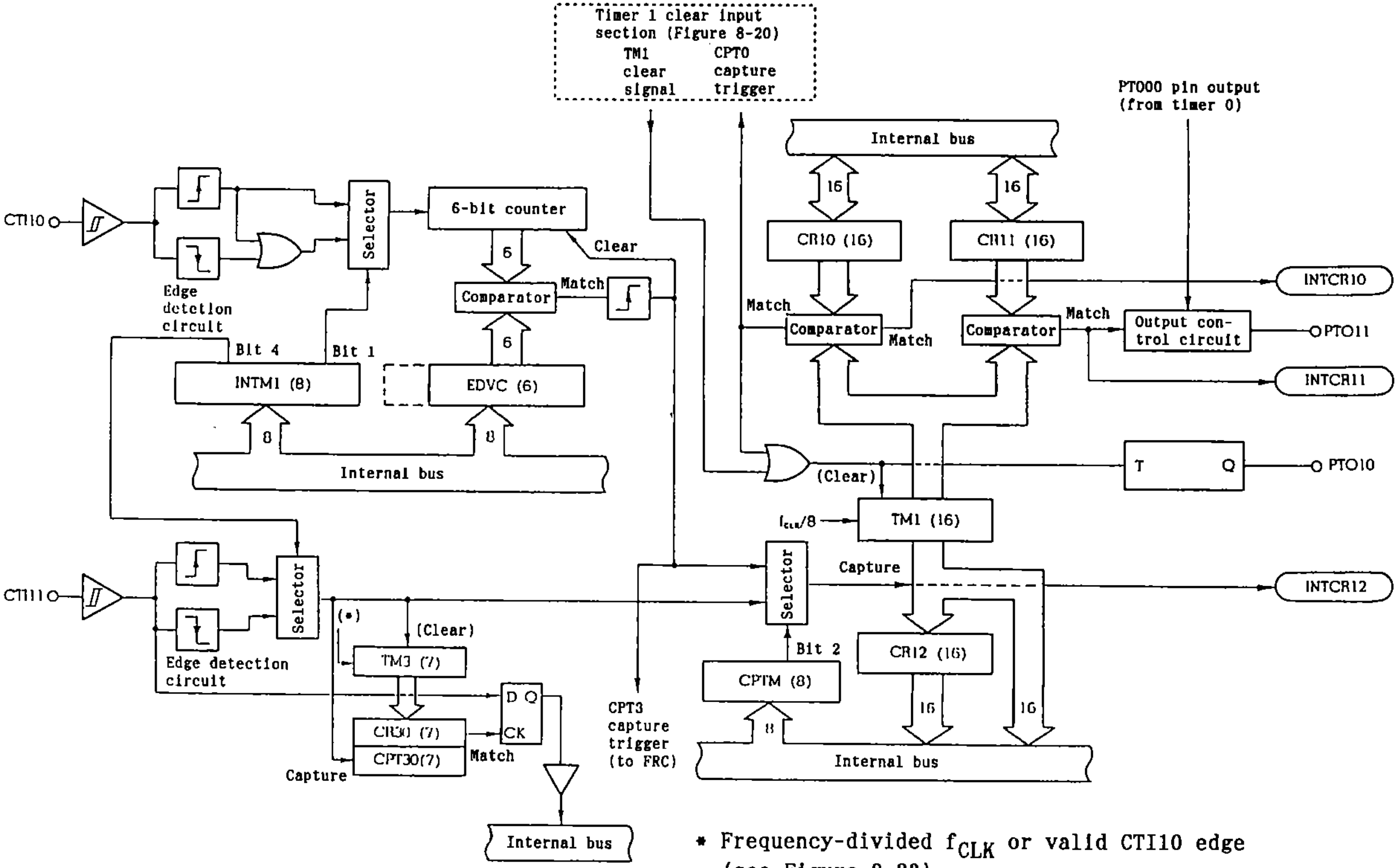
For example, the pulse width detection circuit enables a VHS index search system (VISS) signal to be detected by hardware.

(3) Sixteen-bit timer 1 (TM1)

Sixteen-bit timer 1 (TM1) operates as an interval timer for generating INTCR10 interrupts at regular intervals, and also operates as a reference counter that functions in phase with an external event.

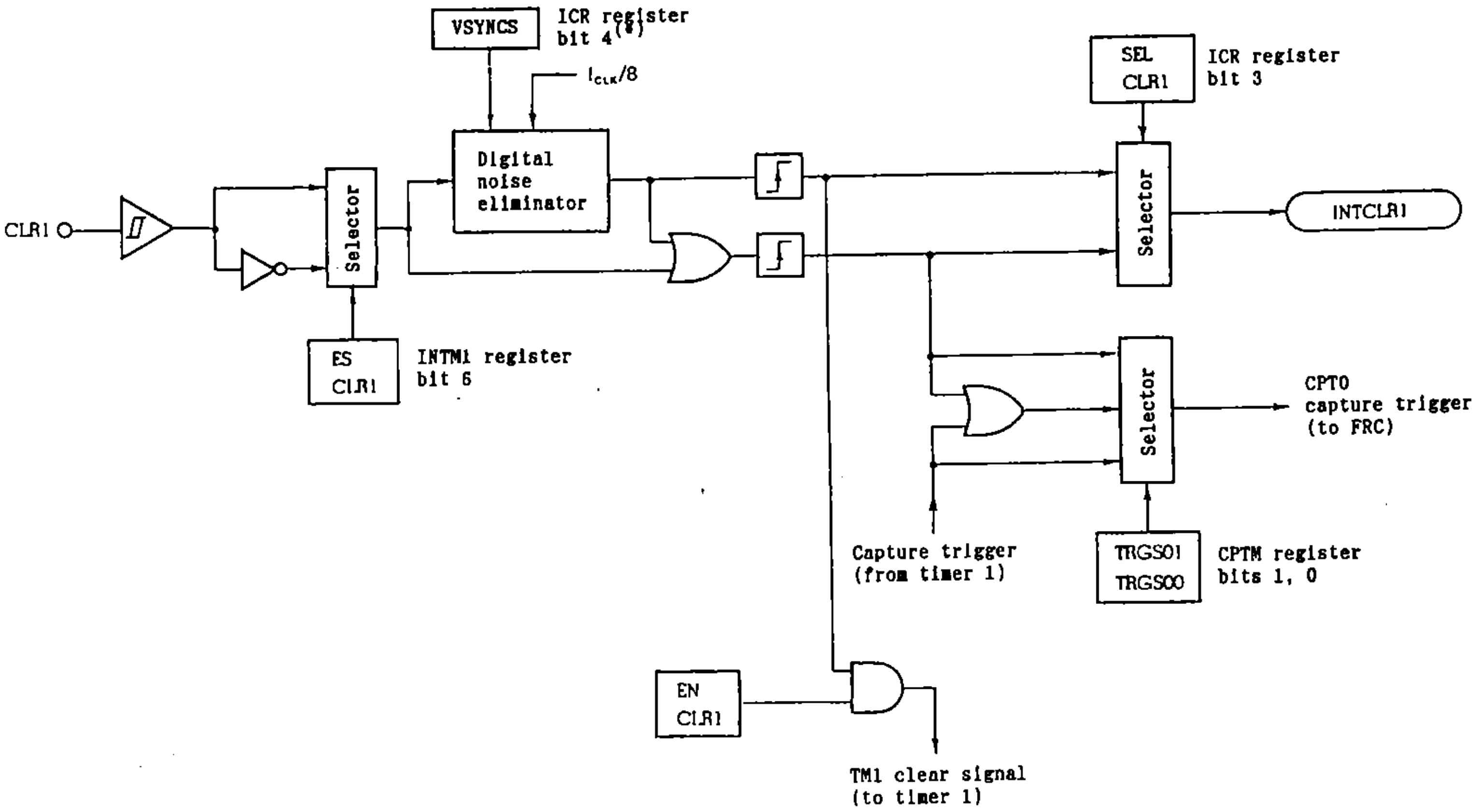
This timer has two pulse output pins; one pin is used for output synchronized with the TM1 clear timing, and the other pin is used for programmable pulse output.

Fig. 8-19 Configuration of the Timer 1 Unit



* Frequency-divided f_{CLK} or valid CTI10 edge (see Figure 8-23)

Fig. 8-20 Configuration of the Timer 1 Clear Input Section



* Pulses with a width in either of the following ranges are removed as noise. Either range is selected according to the setting of bit 4 of the ICR register.

. $(8/f_{CLK}) \times 4$ or less
. $(8/f_{CLK}) \times 9$ or less

In an application to a VCR, for example, timer 1 can be used as described below.

In the playback mode, timer 1 is used as a reference counter for generating interrupts at regular intervals. In the recording mode, timer 1 is used to generate a phase reference signal synchronized with an external vertical synchronizing signal (V_{sync}).

Timer 1 is also used for capstan phase control and for generating a recording control signal (RECCTL signal).

8.3.2 Event divider

The event divider is a 6-bit event divider that can divide a pulse signal applied to the CTI10 pin according to the value held in the event divider control register (EDVC).

When data is written into EDVC, the 6-bit counter for dividing a CTI10 input signal is cleared, and the signal is divided by the value set in EDVC.

An edge detection mode for a CTI10 input signal is specified using bit 1 of the external capture input mode register (INTM1).

8.3.3 Pulse width detection circuit (TM3)

The pulse width detection circuit detects the duty ratio of a pulse signal applied to the CTI11 input pin.

The pulse width detection circuit consists of the following hardware:

- 7-bit timer counter (TM3)
- 7-bit compare register (CR30)
- 7-bit capture register (CPT30)
- Control flip-flop (CTL F/F)

Figure 8-25 shows the configuration of the pulse width detection circuit.

(1) Operation of the pulse width detection circuit

The pulse width detection circuit operates in one of the two modes described below.

(a) General timer operation

Upon valid edge input to CTI11, the count value of timer 3 (TM3) is captured in the capture register CPT30. Then TM3 is cleared to zero and count operation is restarted.

At the same time, an INTCR12 interrupt request is generated.

(b) Pulse width detection operation

As in general timer operation, the count value of TM3 is captured in CPT30 upon valid edge input to CTI11. This count value corresponds to one input pulse period.

Then the value corresponding to an input pulse width detection point is loaded into the compare register CR30. When a match signal between TM3 and CR30 occurs, the CTI11 pin level is latched in the control flip-flop. This value can be checked by reading bit 7 of prescaler mode register 3 (PRM3). This function enables the duty ratio of a pulse signal applied to the CTI11 pin to be detected.

If the period of an input pulse signal changes in duty ratio detection, a pulse width detection point is determined using the value in CPT30. In detecting a pulse signal with a 50% duty ratio, for example, half of the value of CPT30 is loaded into CR30. In this case, the value of CPT30 indicates the pulse width one period before the pulse applied to the CTI11 pin. Figure 8-22 shows the timing.

Phase-out/Discontinued

Fig. 8-21 Configuration of the Pulse Width Detection Circuit (TM3)

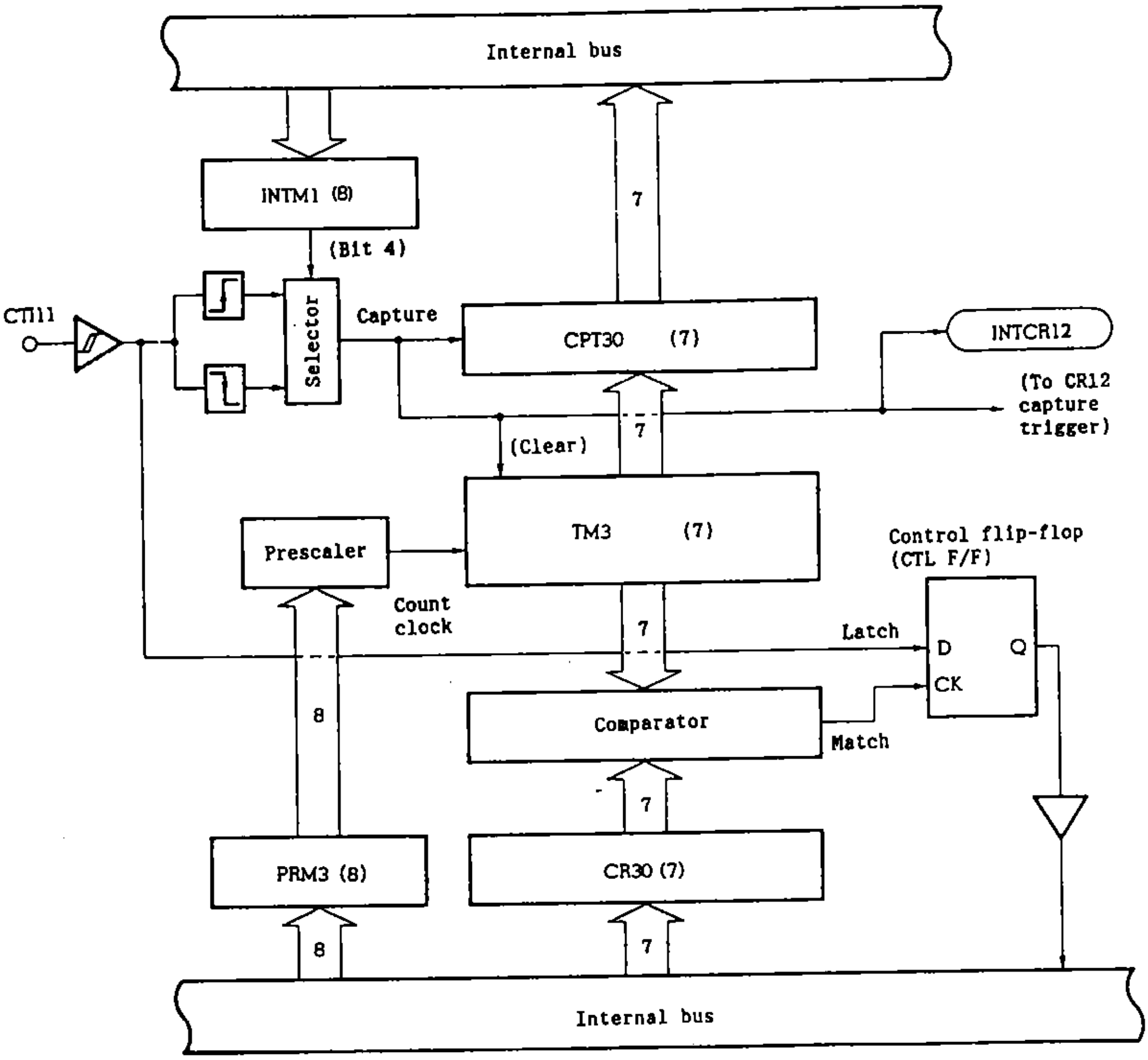
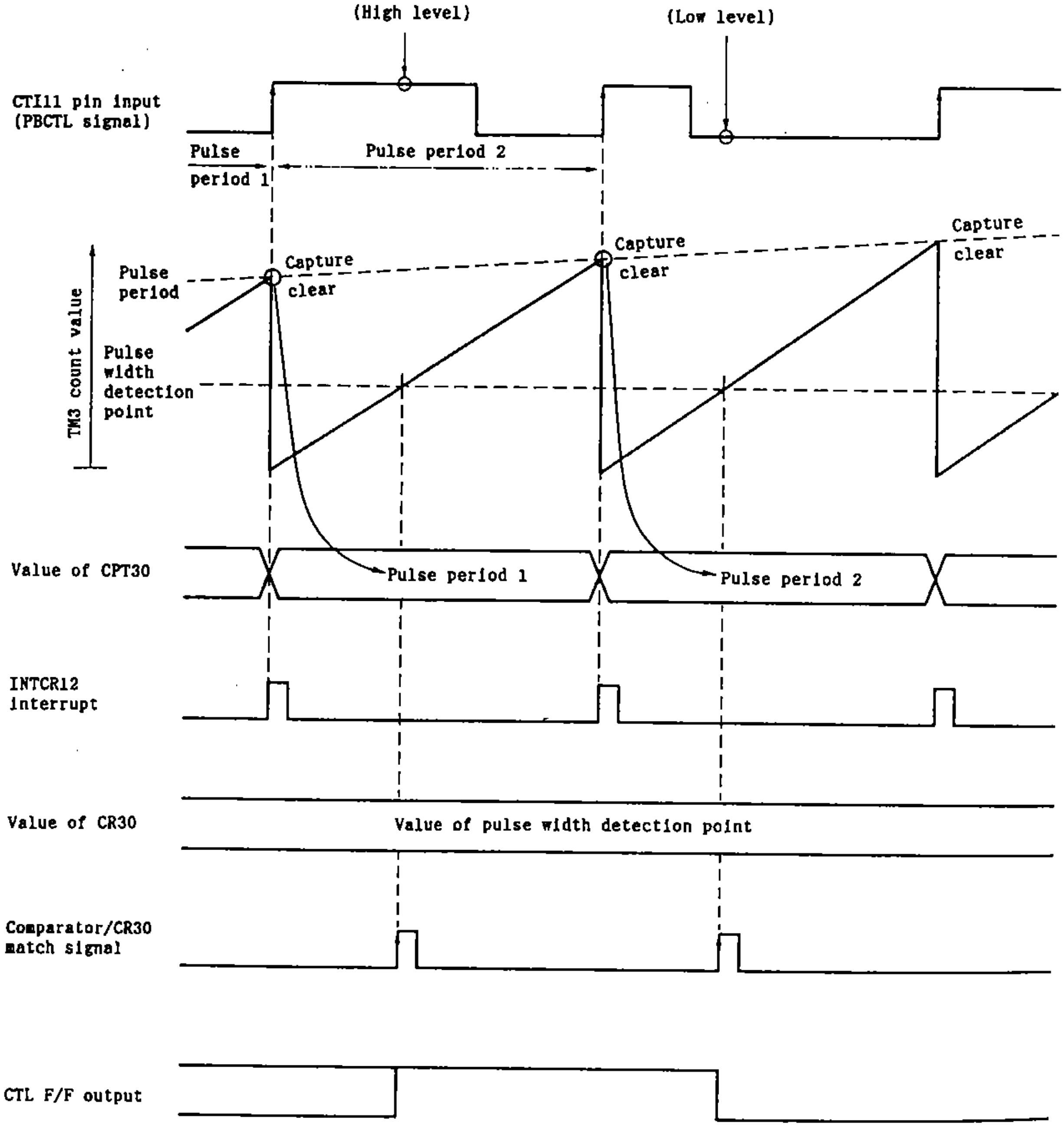


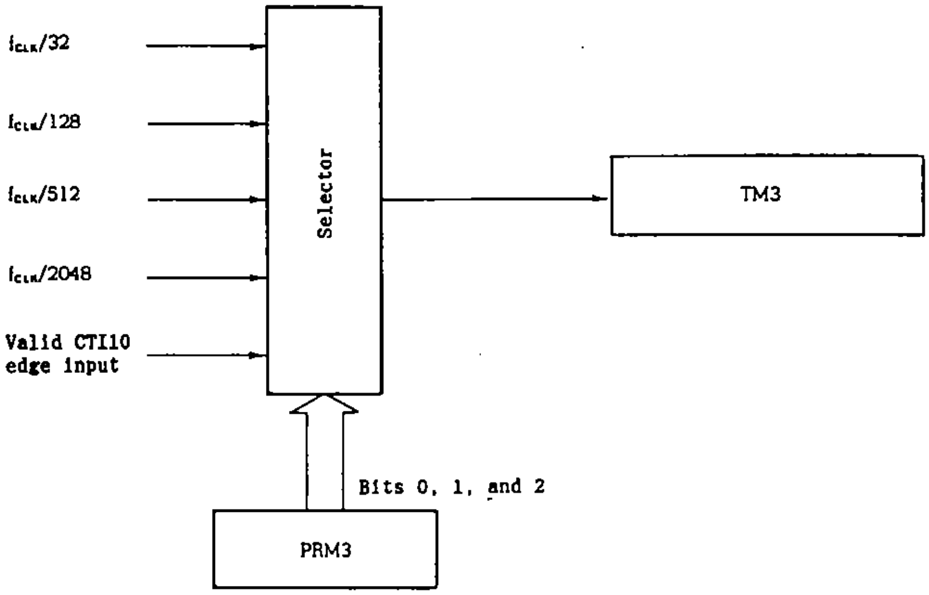
Fig. 8-22 Timing of Pulse Width Detection Circuit Operation



(2) Prescaler

The count clock of the 7-bit timer (TM3) in the pulse width detection circuit can be set to one of five different values by using the prescaler. Figure 8-23 shows the configuration of the prescaler.

Fig. 8-23 Configuration of the Prescaler



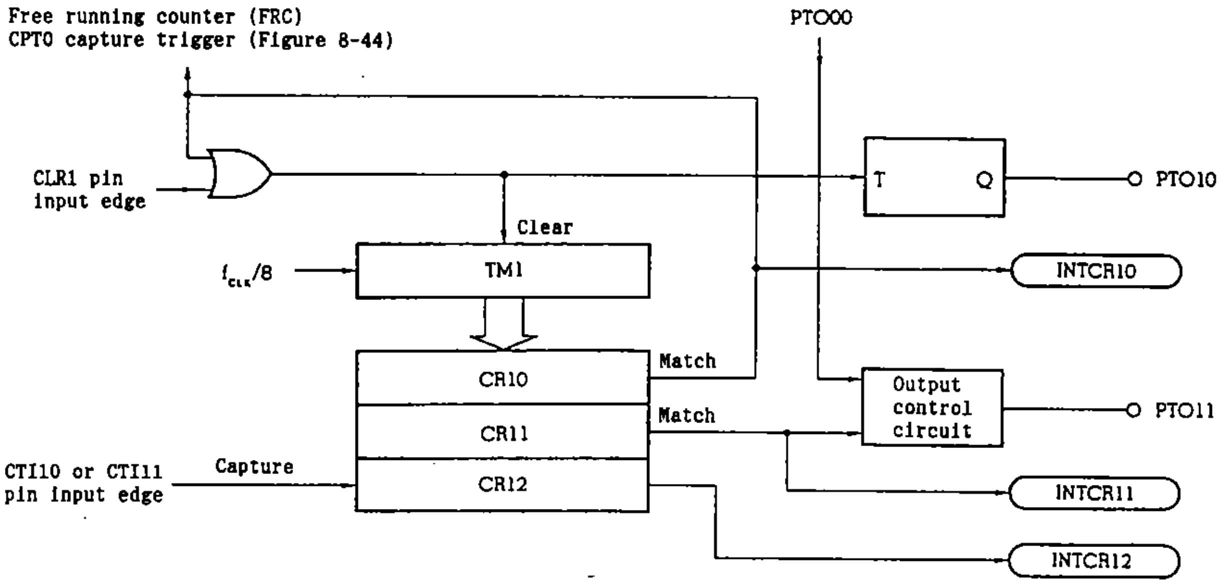
The prescaler allows the count clock of the 7-bit timer (TM3) to be set to one of five values: $f_{CLK}/32$, $f_{CLK}/128$, $f_{CLK}/512$, $f_{CLK}/2048$, and valid edge input to CTI10.

Prescaler mode register 3 (PRM3) is used for count clock specification.

8.3.4 Configuration of timer 1 (TM1)

Figure 8-24 shows the configuration of timer 1 of the uPD78138.

Fig. 8-24 Configuration of Timer 1 (TM1)



Timer 1 has two timer output pins.

PTO10 is a T flip-flop that inverts output when a match signal between TM1 and CR10 occurs.

PTO11 has two modes: the general output mode and delay pulse output mode. Figure 8-25 shows the output modes.

In the general output mode, PTO11 functions as a T flip-flop that inverts output when a match signal between TM1 and CR11 occurs.

In the delay pulse output mode, the output state of PT000 can be output on the PT010 pin after it is delayed by an arbitrary amount with respect to the internal reference signal.

Table 8-6 indicates the resolution and maximum count time of TM1 when f_{CLK} = 6 MHz (at 12 MHz).

Table 8-6 Resolution of Timer 1 (at 12 MHz)

Input clock	Resolution	Maximum count time
750 kHz (f_{CLK}/8)	1.33 us	87.4 ms

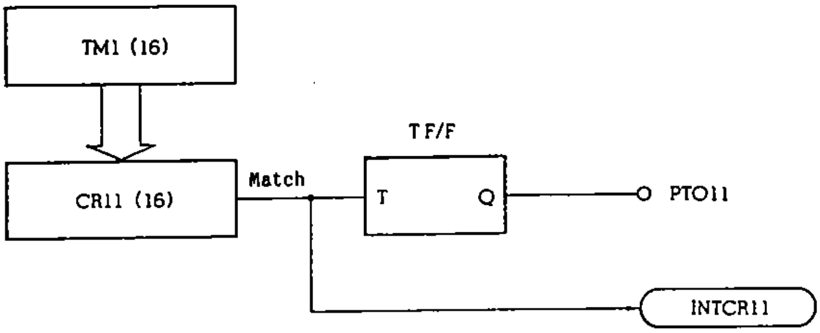
Caution: TM1 is undefined for 16 clock pulses (16/f_{CLK}) after reset release. Start TM1 count operation at the 17th clock pulse or later.

8.3.5 Output modes of timer 1 output pins

Timer 1 has two timer output pins. The output modes are classified into the general output mode and delay pulse output mode.

Fig. 8-25 Output Modes of PT011

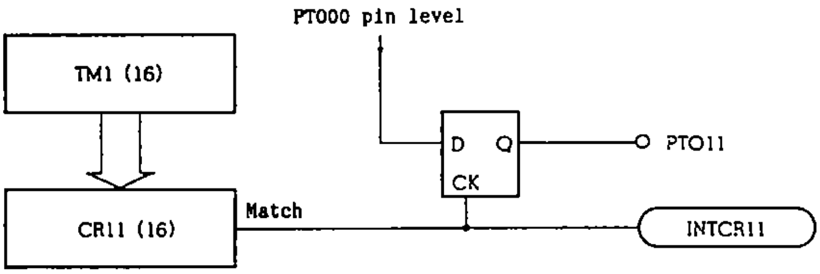
(a) General output mode



(to be continued)

Fig. 8-25 Output Modes of PTO11 (Cont'd)

(b) Delay pulse output mode



Caution: Only the general output mode can be set for the PTO10 pin, which cannot be used in the delay pulse output mode.

8.3.6 Operation of timer 1

(1) Operation as a reference counter (CR10 function).

(a) When CLR1 input is not used

The value corresponding to some interval is to be loaded into the CR10 register beforehand. With this setting, when a match signal between

TM1 and CR10 occurs, TM1 is cleared and INTCR10 interrupts occur at regular intervals.

Figure 8-26(a) shows the timing of this operation.

(b) When CLR1 input is used

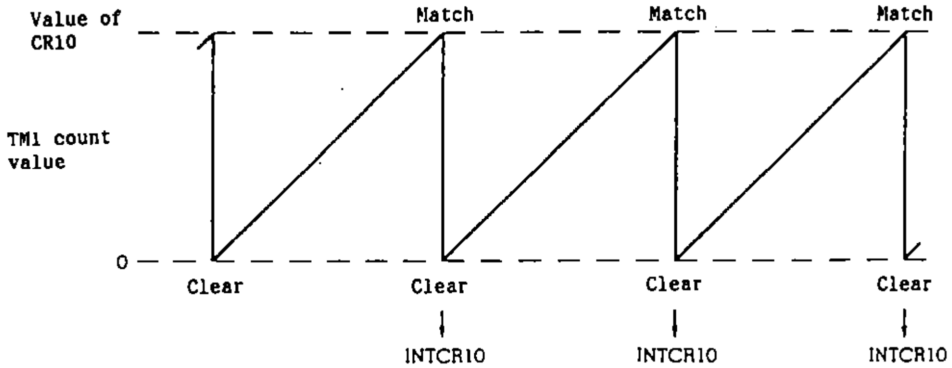
Each time an external event signal is applied to the CLR1 pin, TM1 is cleared. Timer 1 always operates as a reference counter synchronized with an external event.

Apply an event signal to the CLR1 pin at regular intervals, and set the CR10 register to the same value as this interval period. Then if no event signal is applied to the CLR1 pin for a cause, the reference counter (TM1) is internally cleared automatically to correct the internal reference signal.

Figure 8-26(b) shows the timing of this operation.

Fig. 8-26 Reference Counter Operation of Timer 1

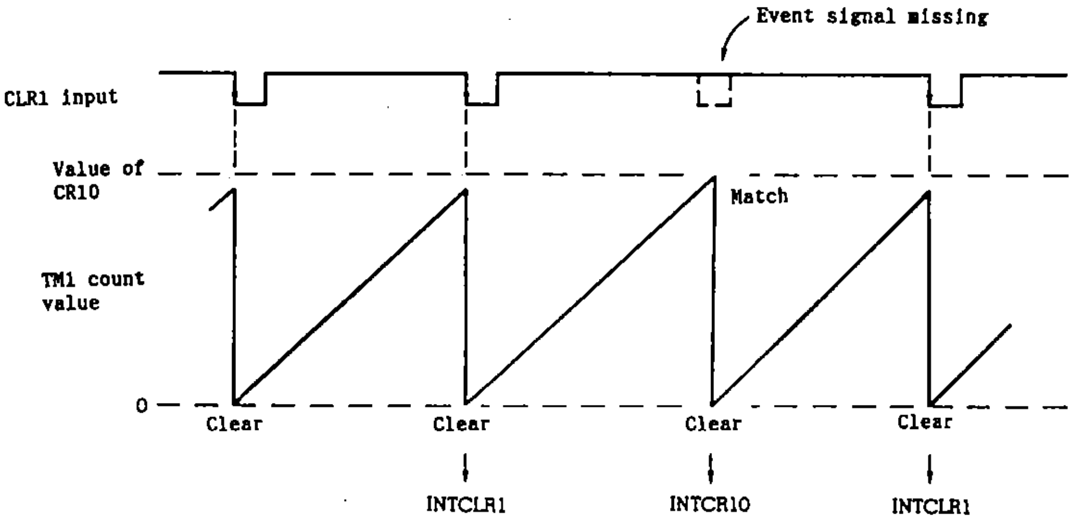
(a) When CLR1 input is not used



(to be continued)

Fig. 8-26 Reference Counter Operation of Timer 1 (Cont'd)

(b) When CLR1 input is used



(2) Programmable pulse output (CR11 function)

The level of the PT000 pin delayed by an arbitrary amount with respect to the internal reference signal can be output on the PT010 pin.

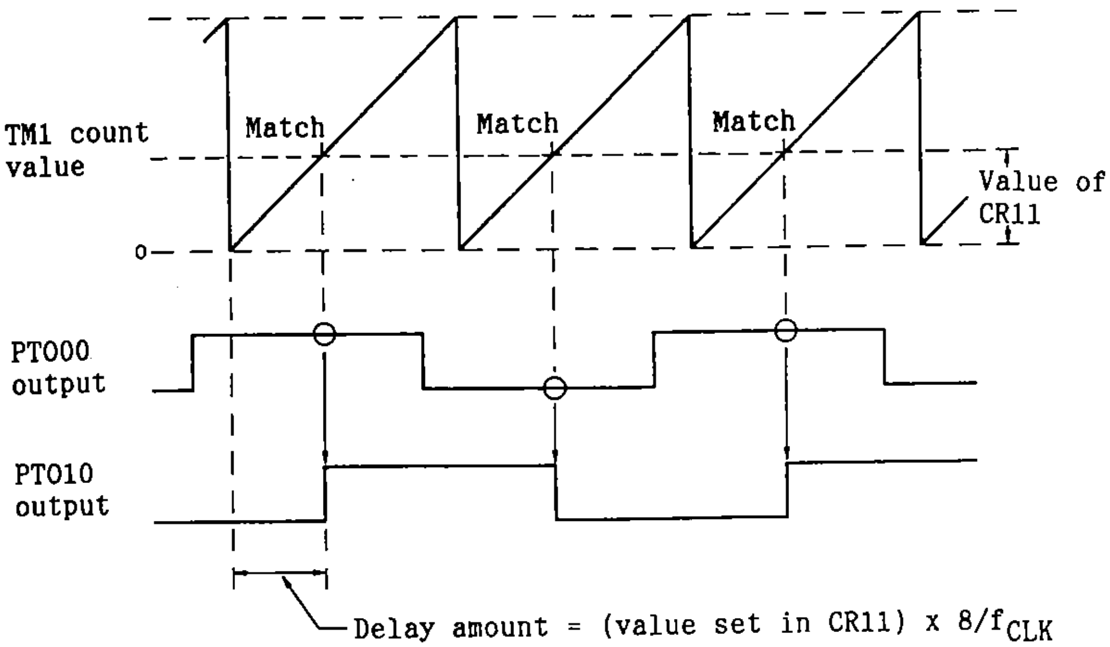
Figure 8-27 shows the timing of this operation.

The relationship between a value set in CR11 and delay amount is:

Delay amount = (value set in CR11) x $8/f_{CLK}$

f_{CLK}: Internal system clock

Fig. 8-27 Programmable Timer Output Operation of Timer 1



(3) Phase difference detection (CR12 function)

A delay amount of external event signal occurrence on the CTI10 or CTI11 pin with respect to the internal reference signal can be detected.

When an event signal is applied to the CTI10 or CTI11 pin, the count value of TM1 is captured in the CR12 register. So this capture value exactly matches a delay amount with respect to the reference signal.

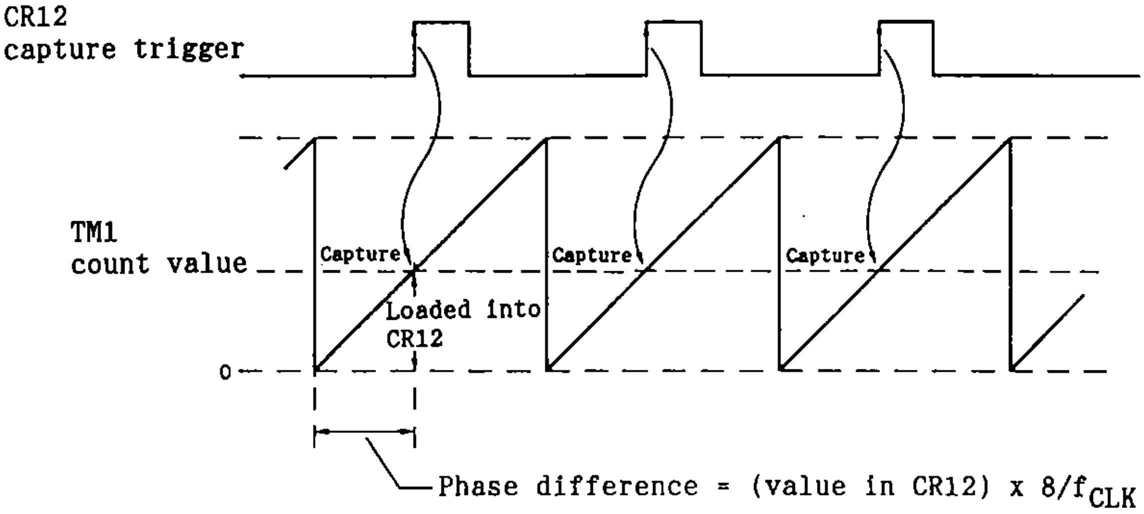
Figure 8-28 shows the timing of this operation.

The relationship between a capture value in CR12 and delay amount is:

Delay amount = (capture value in CR12) x 8/f_{CLK}

f_{CLK}: Internal system clock

Fig. 8-28 Timer 1 Phase Difference Detection Operation



8.3.7 Digital noise eliminator

The uPD78138 contains a digital noise eliminator in the CLR1 input section. The digital noise eliminator generates a capture 0 trigger signal for the free running counter, timer 1 clear signal, and INTCLR1 interrupt signal.

Figure 8-29 shows the configuration of the CLR1 input section.

(1) Operation of the digital noise eliminator

The digital noise eliminator samples an input signal at intervals of $f_{CLK}/8$.

The noise elimination pulse width can be selected from two types: 4 samples and 9 samples. Bit 4 of the input control register (ICR) is used for this selection.

If the pin has the same level consecutively for the time corresponding to the noise elimination pulse width, the level is assumed to be valid, and the level is output at the next sampling timing.

A signal output from the digital noise eliminator after noise removal lags a CLR1 input signal. This delay amount d is expressed as shown in Table 8-7.

The digital noise eliminator allows a CLR1 pin input edge to be specified. Bit 6 of the external capture input mode register (INTM1) is used for this specification.

Table 8-7 Digital Noise Eliminator Specification

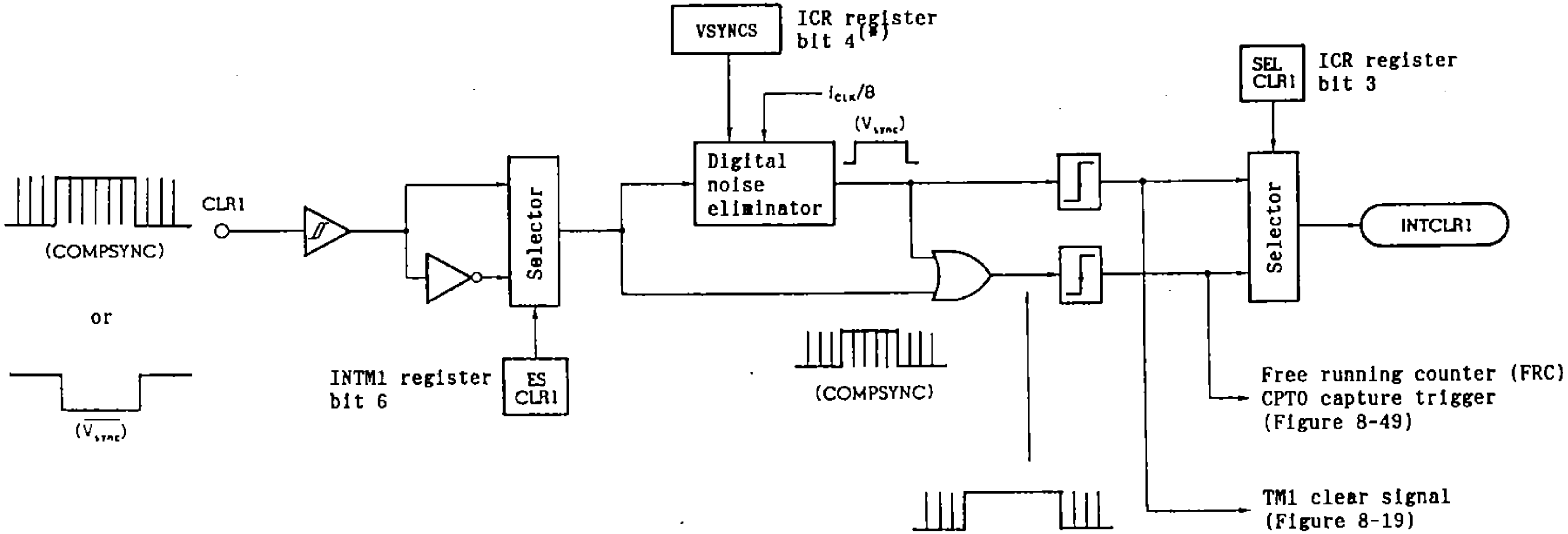
Bit 4 of ICR	Width of the pulse to be eliminated	Width of the pulse that is passed	Delay
0	5.3 us	6.7 us or more	5.3 us ≦ τd < 6.7 us
1	12.0 us	13.3 us or more	12.0 us ≦ τd < 13.3 us

One of two INTCLR1 interrupt occurrence sources can be selected for the digital noise eliminator.
The INTCLR1 interrupt occurrence source is specified using bit 3 of the input control register (ICR).

Clearing bit 3 of the ICR to 0 selects the rising edge of the pulse passed through the digital noise eliminator as an INTCLR1 interrupt occurrence source (vertical synchronizing signal input mode).

Setting bit 3 of the ICR to 1 selects the rising edge of the signal obtained by ORing the signal passed through the digital noise eliminator with the signal that is not yet passed through the eliminator as an INTCLR1 interrupt occurrence source (composite synchronizing signal input mode).

Fig. 8-29 Configuration of the CLR1 Input Section (Digital Noise Eliminator)



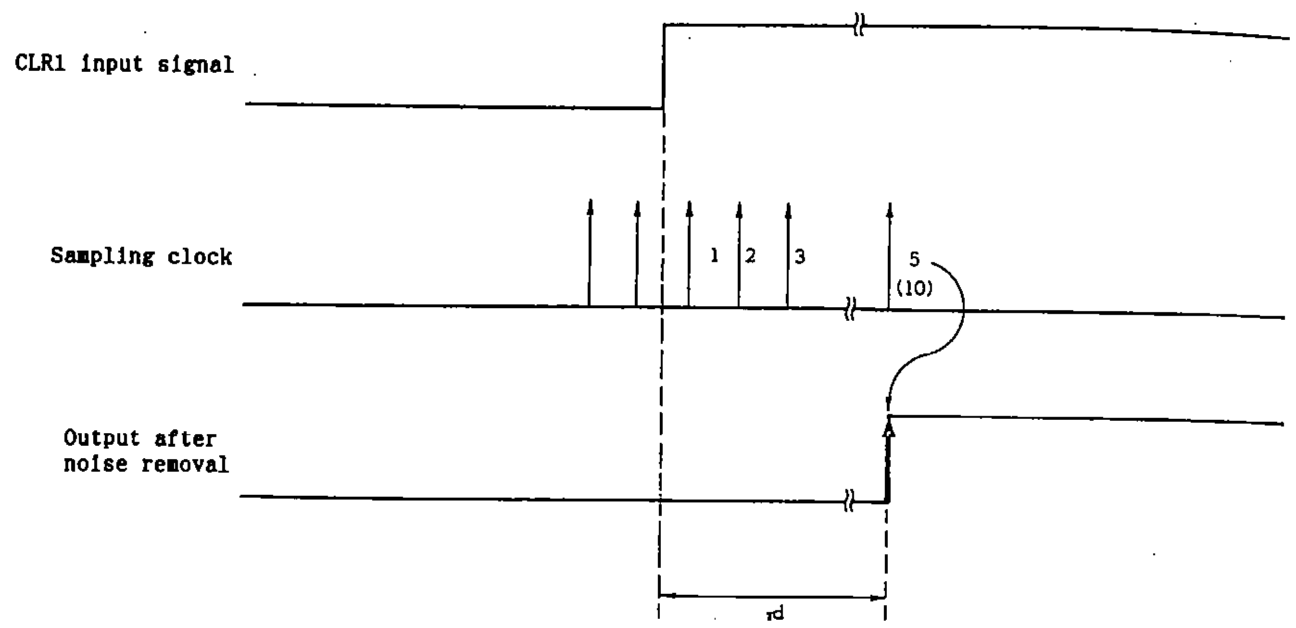
* Pulses with a width in either of the following ranges are removed as noise. Either range is selected according to the setting of bit 4 of the ICR register.

. $(8/f_{CLK}) \times 4$ or less

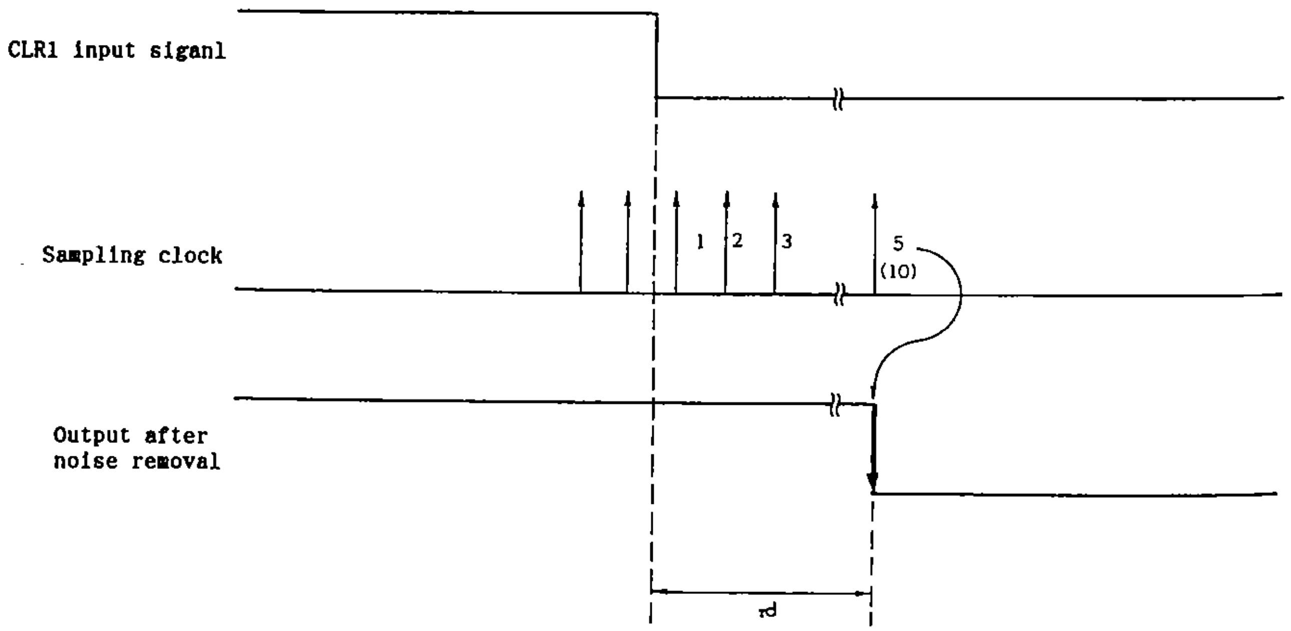
. $(8/f_{CLK}) \times 9$ or less

Fig. 8-30 Operation of the Digital Noise Eliminator

(1) Rising Edge Detection



(2) Falling Edge Detection



(2) Application of the digital noise eliminator to a VCR

Two application examples are described below. One example extracts a vertical synchronizing signal (V_{sync}) from a VCR composite synchronizing signal (COMPSYNC). In the other example, the digital noise eliminator is used for even/odd field determination.

(a) Vertical synchronizing signal (V_{sync}) extraction

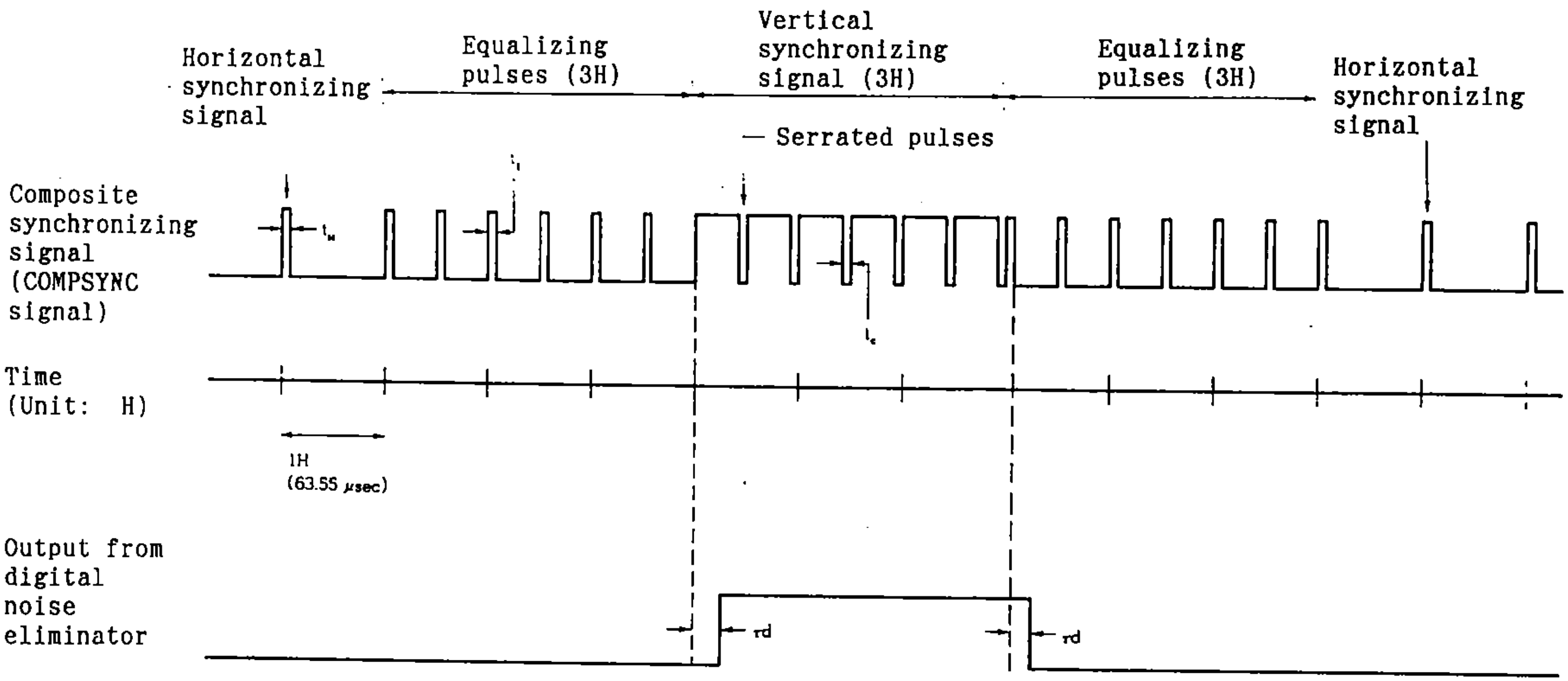
From a COMPSYNC signal, a V_{sync} signal can be extracted only by passing the COMPSYNC signal through the digital noise eliminator. As shown in Figure 8-31, a NTSC COMPSYNC signal consists of a horizontal synchronizing signal (H_{sync}), vertical synchronizing signal (V_{sync}: including serrated pulses), and equalizing pulses. When such a COMPSYNC signal is applied to the digital noise eliminator, the H_{sync} signal, equalizing pulses, and serrated pulses are removed, and only V_{sync} signal output can be obtained.

One of $32/f_{CLK}$ and $72/f_{CLK}$ can be specified as the noise elimination pulse width for the uPD78138.

In this case, a V_{sync} signal output from the digital noise eliminator lags the V_{sync} signal in the applied COMPSYNC signal by up to $40/f_{CLK}$ (6.7 us at 12 MHz) or up to $80/f_{CLK}$ (13.3 us at 12 MHz).

If $32/f_{CLK}$ is selected, a V_{SYNC} signal with less delay can be obtained. $72/f_{CLK}$ should be selected when the radio reception is poor.

Fig. 8-31 Vertical Synchronizing Signal Extraction Using the Digital Noise Eliminator


Horizontal synchronizing signal
Equalizing pulses (3H)
Vertical synchronizing signal (3H)
Equalizing pulses (3H)
Horizontal synchronizing signal
Serrated pulses
Composite synchronizing signal (COMPSYNC signal)
t_H
t_E
t_C
Time (Unit: H)
1H (63.55 μsec)
Output from digital noise eliminator
τd
τd

Remarks 1. NTSC-based pulse widths

t_H (pulse width of horizontal synchronizing signal):	4.8 us (0.075 H)
t_E (equalizing pulse width):	2.5 us (0.04 H)
t_C (serrated pulse width):	4.4 us (0.07 H)
τd (delay due to digital sampling):	Max. 6.7 us or 13.3 us

2. The unit of time H is the horizontal synchronizing signal period.

(b) Even/odd field determination

The position of the sixth equalizing pulse of a COMPSYNC signal can be used for even/odd field determination (Figure 8-32). This determination uses the digital noise eliminator and the CLR1 pin interrupt INTCLR1.

For the INTCLR1 interrupt, an input signal detection edge can be selected with the external capture input mode register (INTM1), and an interrupt source can be selected with the input control register (ICR).

Figures 8-33(a) to (c) show examples of setting.

An INTCLR1 interrupt source can be selected as described below.

. Applying a COMPSYNC signal to the CLR1 pin (used for Figures 8-33(a) and (b))

In Figure 8-33(a), an INTCLR1 interrupt is generated on a rising edge of V_{sync} extracted from the COMPSYNC signal.

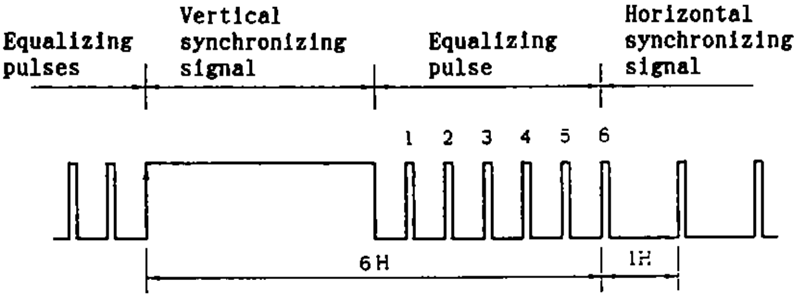
In Figure 8-33(b), an INTCLR1 interrupt is generated on a rising edge of V_{sync} with only serrated pulses removed from the COMPSYNC signal or on a rising edge of equalizing pulses.

- Applying a $\overline{V_{sync}}$ signal to the CLR1 pin (used for Figure 8-33(c))

  In Figure 8-33(c), an INTCLR1 interrupt is generated on a falling edge of $\overline{V_{sync}}$.

Fig. 8-32 COMPSYNC Signal Used for Even/Odd Field Determination

(a) Odd field


Equalizing pulses
Vertical synchronizing signal
Equalizing pulse
Horizontal synchronizing signal
1
2
3
4
5
6
6H
1H

(b) Even field

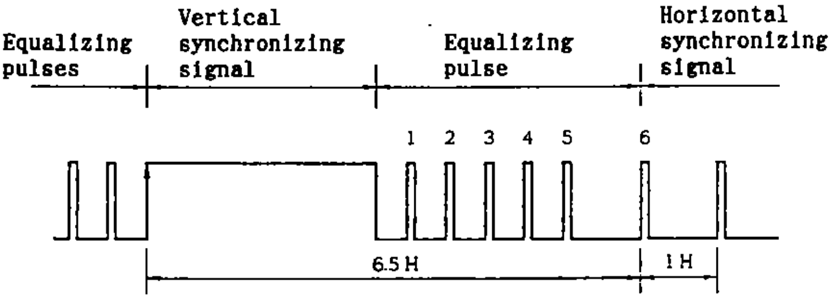
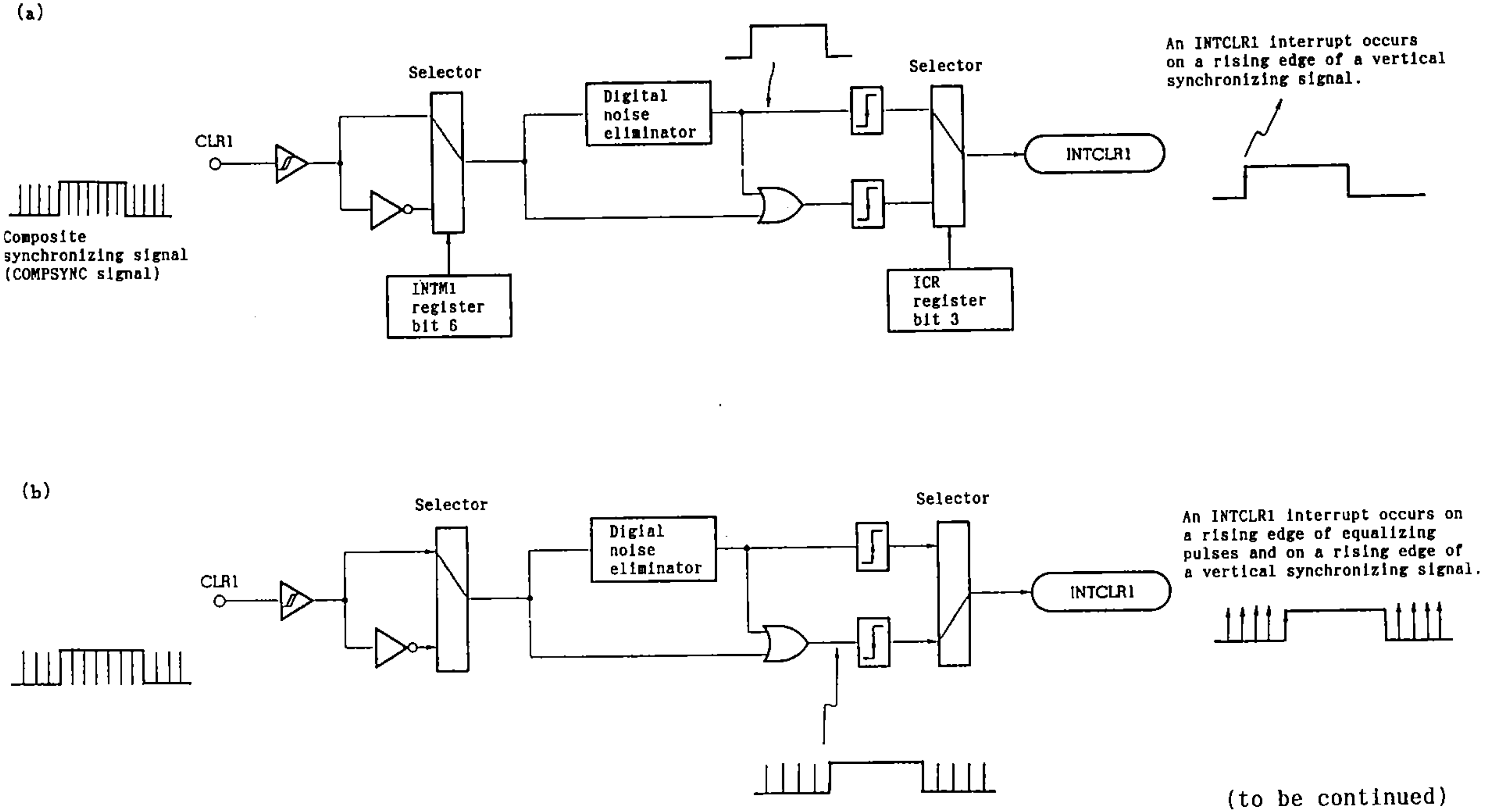

Equalizing pulses
Vertical synchronizing signal
Equalizing pulse
Horizontal synchronizing signal
1
2
3
4
5
6
6.5H
1H

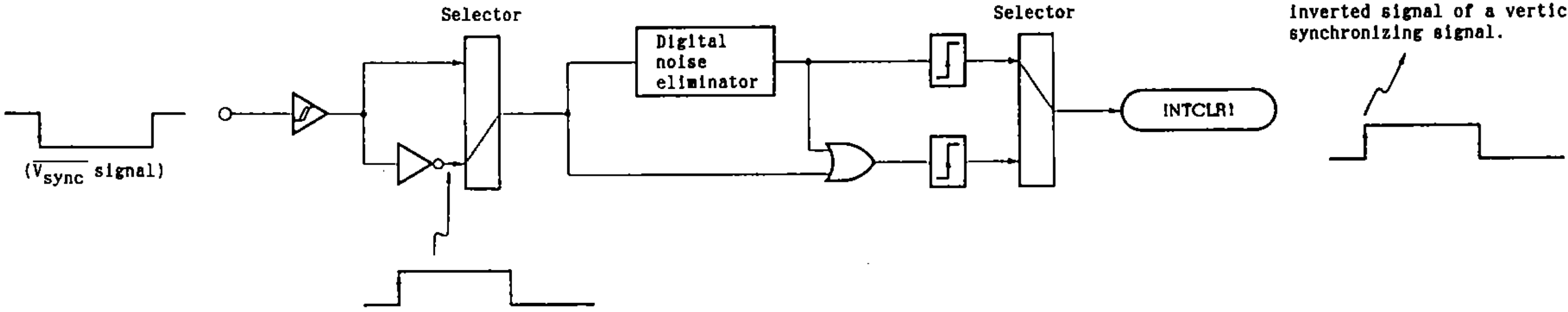
Fig. 8-33 Example of INTCLR1 Setting


(a)
Selector
Digital noise eliminator
Selector
An INTCLR1 interrupt occurs on a rising edge of a vertical synchronizing signal.
CLR1
INTCLR1
Composite synchronizing signal (COMPSYNC signal)
INTM1 register bit 6
ICR register bit 3
(b)
Selector
Digial noise eliminator
Selector
An INTCLR1 interrupt occurs on a rising edge of equalizing pulses and on a rising edge of a vertical synchronizing signal.
CLR1
INTCLR1

(to be continued)

(Cont'd)

(c)



An INTCLR1 interrupt occurs on a rising edge of the inverted signal of a vertical synchronizing signal.

Even/odd field determination is performed by applying a COMPSYNC signal to CLR1. So, an INTCLR1 interrupt source is selected in Figures 8-33(a) and (b).

The procedure for even/odd field determination is explained below.

① An INTCLR1 interrupt source is set as shown in Figure 8-33(a).

Example:

. Set INTM1 bit 6 to 1:
Specifies a CLR1 rising edge.

. Reset ICR bit 3 to 0:
Specifies the vertical synchronizing signal mode.

This setting removes the equalizing pulses preceding V_{sync}.

② An INTCLR1 interrupt is generated on a rising edge of V_{sync}. At the same time, the contents of the free running counter (FRC) are loaded into capture register 0 (CPT0) at edge input to CLR1. Then the contents of CPT0 are stored in memory by using the INTCLR1 interrupt service routine. Let N1 be the contents.

③ An INTCLR1 interrupt source is set as shown in Figure 8-33(b).

Example:

. Set ICR bit 3 to 1:
Specifies the composite synchronizing signal mode.

This setting generates an INTCLR1 interrupt on each rising edge of the equalizing pulses and vertical synchronizing signal.

④ The sixth equalizing pulse can be detected by counting INTCLR1 occurrences. This count operation can be facilitated using the count mode of the macro service. (See Section 11.5.6 in Chapter 11.) This count mode starts vectored interrupt handling only when a specified number of interrupts have occurred.

⑤ The contents of CPT0 at the time the sixth equalizing pulse is detected in ④ are read. This processing is performed as vectored interrupt handling. Let N2 be the read contents of CPT0.

⑥ As shown in Figure 8-32, the position of the sixth equalizing pulse differs between the even field and odd field. When one period of the horizontal synchronizing signal is 1 H, the sixth equalizing pulse for the even field is placed at 6.5 H after a rising edge of the vertical synchronizing signal. On the other hand, the sixth equalizing pulse for the odd field is placed at 6.0 H.

The contents (N1) of CPT0 loaded in ② indicate a rising edge of the vertical synchronizing signal. On the other hand, the contents (N2) of CPT0 read in ⑤ indicate the position of the sixth equalizing pulse. This means that even/odd field determination can be performed by finding the difference between N1 and N2 and checking whether the difference is an integral multiple of H (one horizontal synchronizing signal period).

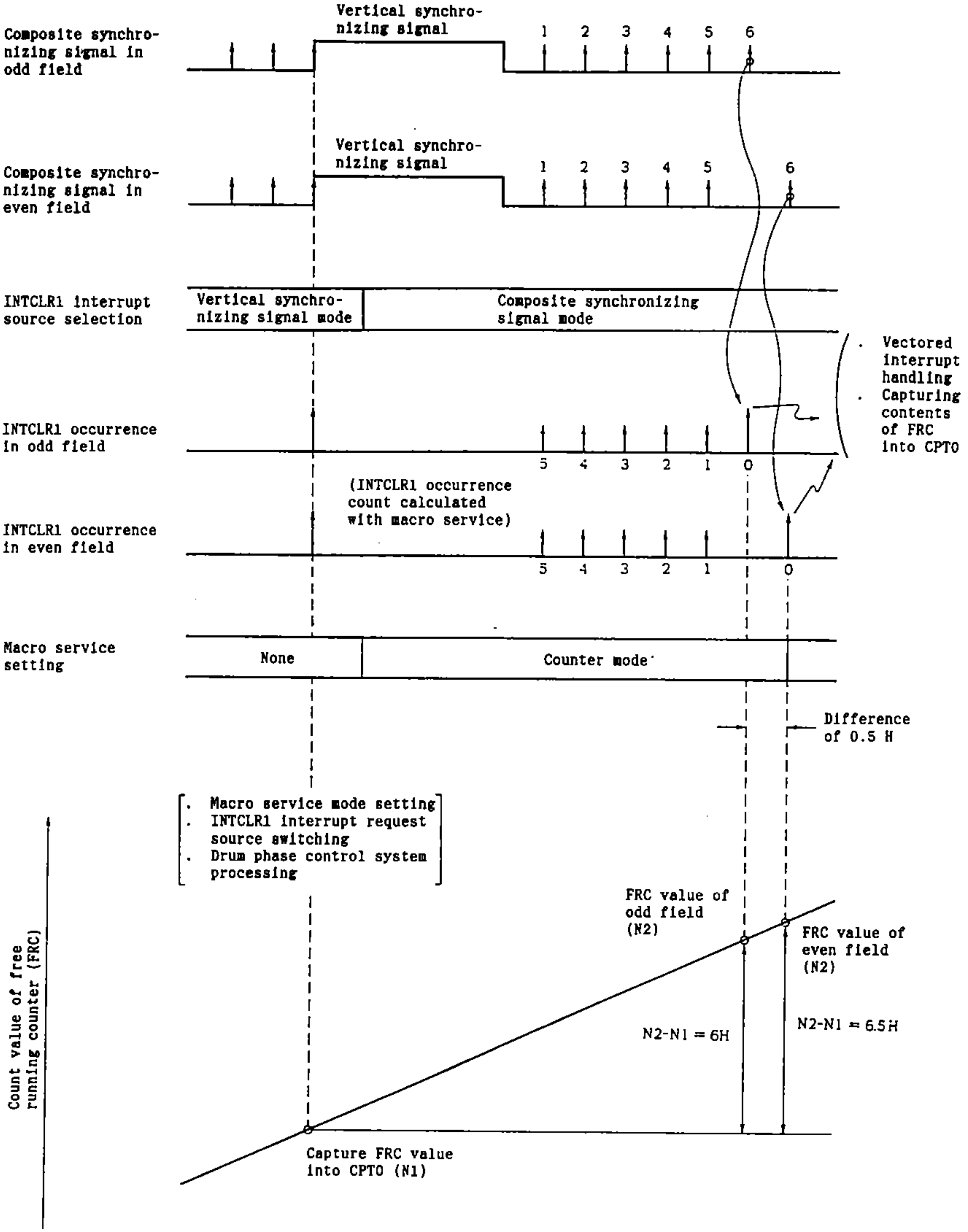
Remarks 1. See Figure 8-37 for the format of INTM1, and Figure 8-43 for the format of ICR.

2. See Section 8.4.3 for the operation of CPT0 of the FRC.

Figure 8-34 illustrates the description above.

In this example, even/odd field determination is performed using the sixth equalizing pulse of a vertical synchronizing signal. However, a COMPSYNC signal actually applied often contains noise, so that equalizing pulses may not be detected. If equalizing pulses cannot be counted normally, the difference between the contents of CPT0 at the time of horizontal synchronizing signal input and the contents of CPT0 at the time of a rising edge of a vertical synchronizing signal is found. By checking whether the difference is an integral multiple of the period of the horizontal synchronizing signal, even/odd field determination can be performed.

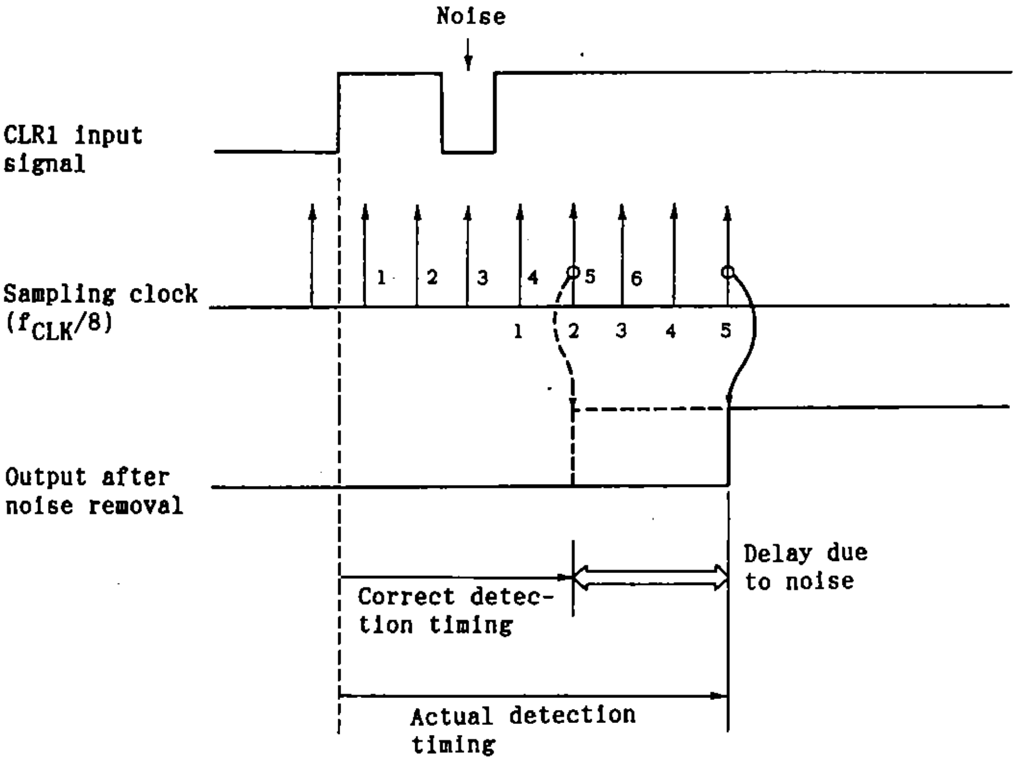
Fig. 8-34 Even/Odd Field Determination Using the Digital Noise Eliminator


Composite synchronizing signal in odd field
Vertical synchronizing signal
1 2 3 4 5 6
Composite synchronizing signal in even field
Vertical synchronizing signal
1 2 3 4 5 6
INTCLR1 interrupt source selection
Vertical synchronizing signal mode
Composite synchronizing signal mode
. Vectored interrupt handling
. Capturing contents of FRC into CPT0
INTCLR1 occurrence in odd field
5 4 3 2 1 0
(INTCLR1 occurrence count calculated with macro service)
INTCLR1 occurrence in even field
5 4 3 2 1 0
Macro service setting
None
Counter mode
Difference of 0.5 H
. Macro service mode setting
. INTCLR1 interrupt request source switching
. Drum phase control system processing
FRC value of odd field (N2)
FRC value of even field (N2)
N2-N1 = 6H
N2-N1 = 6.5H
Count value of free running counter (FRC)
Capture FRC value into CPT0 (N1)

Remark: 1H: One horizontal synchronizing signal period

Caution: If a narrow noise appears on a CLR1 input signal in phase with the sampling timing as shown in Figure 8-35, the digital noise eliminator malfunctions. In practice, the detection of a rising edge of the CLR1 input signal may lag the correct detection timing.

Fig. 8-35 Example of CLR1 Input Detection Error Due to Noise



8.3.8 Setting timer 1 unit control registers

The operation of the timer 1 unit can be controlled by the following registers:

EDVC: Event divider control register (for event divider control)

INTM1: External capture input mode register 1 (for specifying an external input edge)

PRM3: Prescaler mode register (for control of a TM3 prescaler)

CPTM: Capture mode register (for specifying a CR12 capture trigger)

TMC0: Timer 0 control register (for control of Timer 1 operation)

TOM1: Timer 1 output mode register (for control of the PTO1n output mode)

TOC1: Timer 1 output control register (for PTO1n output control)

ICR: Input control register (for control of the digital noise eliminator)

The following figures show the formats of the registers.

Fig. 8-36 Format of the Event Divider Control Register (EDVC)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
EDVC	—	—	EDV5	EDV4	EDV3	EDV2	EDV1	EDV0	FF53H	Not defined	W

EDV5	EDV4	EDV3	EDV2	EDV1	EDV0	Specification for dividing CTI10 input
0	0	0	0	0	0	Not to be set
0	0	0	0	0	1	Through output No dividing
0	0	0	0	1	0	Divided by 2
⋮						⋮
1	1	1	1	1	0	Divided by 62
1	1	1	1	1	1	Divided by 63

Fig. 8-37 Format of External Capture Input Mode Register 1 (INTM1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
INTM1	0	ES CLR1	0	ES11	-	-	ES10	1	FFF5H	0000xx01	R/W

ES10	Specification of a CTI10 input detection edge
0	Rising edge
1	Either the rising or falling edge

ES11	Specification of CTI11 input detection edge
0	Falling edge
1	Rising edge

ES CLR1	Specification of valid CLR1 input edge
0	Falling edge
1	Rising edge

Caution: Changing the setting in bit 6 of the external capture input mode register (INTM1) may cause an INTCLR1 interrupt request.

Fig. 8-38 Format of Prescaler Mode Register (PRM3)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PRM3	FFLVL	0	0	0	-	PRM32	PRM31	PRM30	FF1DH	0xxxx000	R/W

PRM32	PRM31	PRM30	Specification of prescaler output frequency (at 12 MHz)
0	0	0	$f_{CLK}/32$ (187.5 kHz)
0	0	1	$f_{CLK}/128$ (46.875 kHz)
0	1	0	$f_{CLK}/512$ (11.719 kHz)
0	1	1	$f_{CLK}/2048$ (2.9297 kHz)
1	0	0	Valid CTI10 edge input (external clock)
Other than the above			Not to be set

FFLVL	Flag for storing the output level of the flip-flop for control of the pulse width detection circuit.

Fig. 8-39 Format of Capture Mode Register (CPTM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
CPTM	-	-	-	-	-	TRGS 12	TRGS 01	TRGS 00	FF3AH	xxxxx000	W

TRGS 01	TRGS 00	Specification of a CPT0 capture trigger
0	0	Match signal between TM1 and CR10
0	1	CLR1 input edge detection signal
1	0	Not to be set
1	1	Signal obtained by ORing the match signal between TM1 and CR10 with the CLR1 input edge detection signal

TRGS 12	Specification of a CR12 capture trigger
0	CTI11 input edge detection signal
1	CTI10 input division signal

Fig. 8-40 Format of Timer Control Register 0 (TMC0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TMC0	CS1	-	-	EN CLR1	CS0	0	0	EN CLR0	FF38H	0xx00000	W

ENCLR0	TM0 clear signal enable bit
0	Prevents TM0 from being cleared by masking a TM0 clear pulse (free running mode).
1	Clears TM0 using a clear pulse.

CS0	TM0 count operation control
0	Clears TM0 and stops counting.
1	Counts.

ENCLR1	TM1 clear signal enable bit
0	Prevents TM1 from being cleared by masking CLR1 input.
1	Clears TM1 by CLR1 input.

CS1	Control of TM1 counting
0	Clears TM1 and stops counting.
1	Counts.

Fig. 8-41 Format of Timer 1 Output Mode Register (TOM1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TOM1	-	-	-	-	MOD 111	MOD 110	0	0	FF5AH	xxxx0000	W

Fixed to 0 (PTO10 is always set to general output mode.)

MOD 111	MOD 110	PTO11 output mode specification
0	0	General output mode
0	1	Not to be set
1	0	Delay pulse output mode 1
1	1	Not to be set

Fig. 8-42 Format of Timer 1 Output Control Register (TOC1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TOC1	-	-	-	-	ENTO 11	ALV 11	ENTO 10	ALV 10	FF5BH	xxxx0000	R/W

ALV 11n	Setting of active level of timer 1 output pin
0	Active low
1	Active high

ENTO 11n	Control of timer 1 output enable/ disable
0	Disable (with output always set at inactive level)
1	Enable

Fig. 8-43 Format of Input Control Register (ICR)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
ICR	SEL CLR0	-	EC MOD	V SYNCS	SEL CLR1	-	-	-	FF50H	0x0x0xxx	W

SEL CLR1	Selection of an INTCLR1 interrupt source
0	Vertical synchronizing signal input mode
1	Composite synchronizing signal input mode

V SYNCS	Selection of a V_{sync} separation pulse width
0	Eliminates pulses of less than 5.3 us ($32/f_{CLK}$).
1	Eliminates pulses of less than 12.0 us ($72/f_{CLK}$).

ECMOD	Specification of the operating mode of the event counter
0	General event divider mode
1	Internal pulse generation mode

SEL CLR0	Selection of a timer 0 clear pulse
0	CLR0 pin input (Bypasses the 6-bit event counter.)
1	Pulse generated internally by the event counter (EC)

8.4 Free Running Counter (FRC) Unit

8.4.1 Configuration of the free running counter unit

As shown in Figure 8-44, the free running counter unit consists of an 18-bit free running counter (FRC), an 18-bit capture register (CPT2), and three 16-bit capture registers (CPT0, CPT1, CPT3).

The free running counter captures the count value of the FRC into a capture register upon occurrence of a capture trigger. By comparing each captured value, the occurrence cycle of capture triggers that occur periodically can be measured. There are four capture registers, so occurrence cycle measurements can be made for four types of capture triggers in parallel.

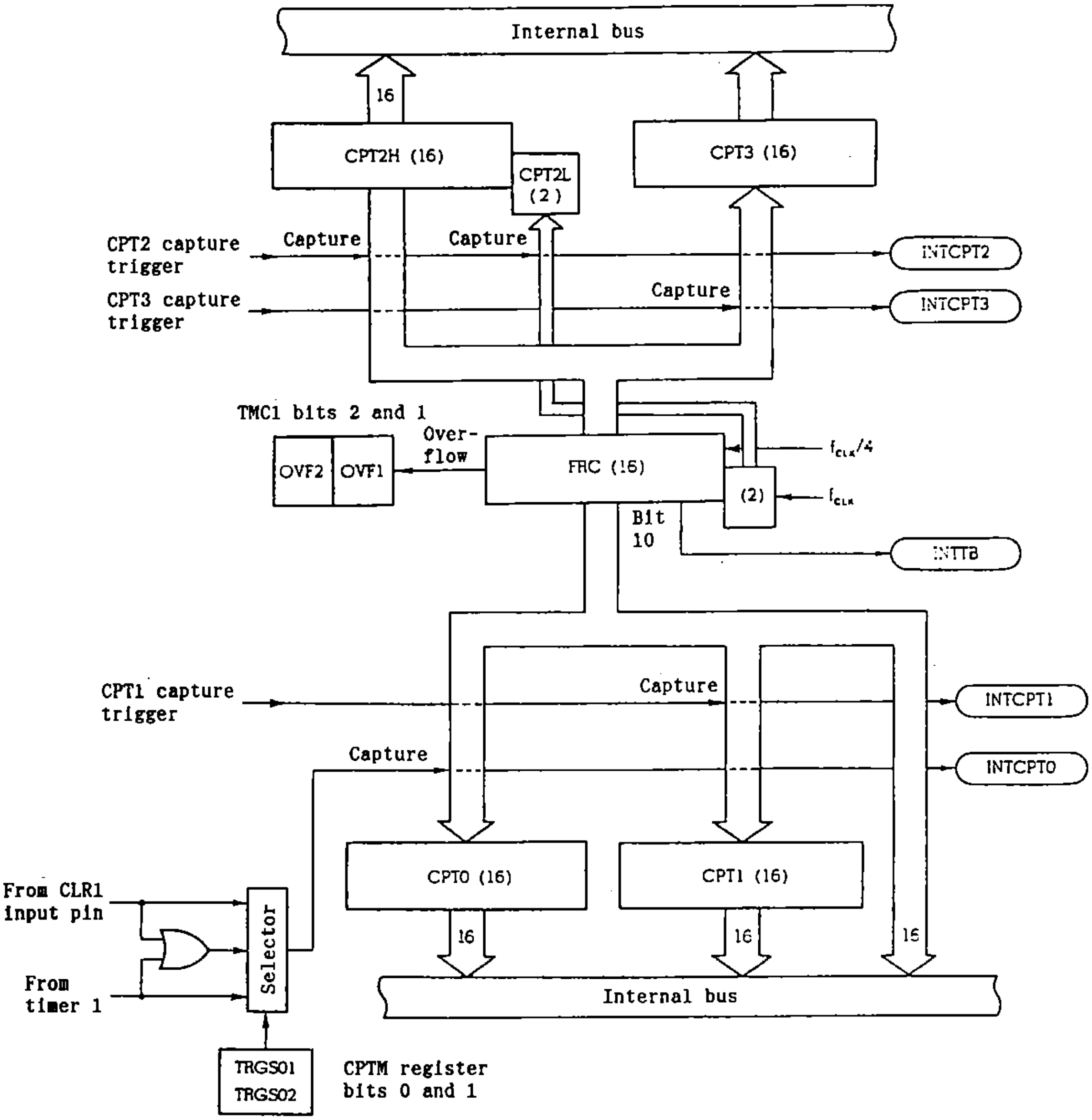
Table 8-8 lists the capture registers and their respective capture triggers.

Table 8-8 FRC Capture Registers and Capture Triggers

Capture register	Capture trigger	Interrupt request
CPT0	Detection signal of CLR1 input rising or falling edge	INTCLR1
	Match signal between TM1 and CR10	INTCR10
	Occurrence of either of the above two triggers	INTCLR1 or INTCR10
CPT1	Detection signal of both timer 0 clear pulse rising edge and falling edge	INTCPT1
CPT2 (CPT2H, CPT2L)	Detection signal of CTI00 input rising edge	INTCPT2
CPT3	Divided signal of CTI10 input edge detection signal	INTCPT3

Caution: A CPT0 capture trigger is specified using bits 0 and 1 of the capture mode register (CPTM) (Figure 8-49).

Fig. 8-44 Configuration of the Free Running Counter



The FRC is an 18-bit counter that counts up with clock input.

The FRC consists of two counters: one is a 16-bit counter that counts up with a $f_{CLK}/4$ clock, and the other is a 2-bit counter that counts up with f_{CLK}.

2-bit counter that counts up with f_{CLK}.
Four capture registers are provided for the FRC. Among these registers, CPT0, CPT1, and CPT3 are 16-bit capture registers, and CPT2 is an 18-bit capture register.

8.4.2 Capture register 2 (CPT2)

The 18-bit capture register (CPT2) consists of two registers: CPT2H and CPT2L.

The CPT2H register is a 16-bit register that holds the higher 16-bit count value of the 18-bit FRC. The CPT2L register is an 8-bit register that holds the lower 2-bit data of the 18-bit FRC. The lower 2-bit data of the FRC is held in the two high-order bits of the CPT2L register, and zero is held in the six low-order bits of the CPT2L register.

Figure 8-45 shows the configuration of CPT2 (CPT2H, CPT2L) of the FRC.

Figure 8-46 shows how data is held in CPT2L.

An example of FRC count operation is shown in Figure 8-47.

When f_{CLK} = 6 MHz, the 16 high-order bits of the FRC count with a count clock of $f_{CLK}/4$ (= 1.5 MHz).

The higher bit of the two low-order bits counts with $f_{CLK}/2$ (= 3 MHz or 333 ns), and the lower bit counts with f_{CLK} (= 6 MHz or 167 ns).

When a rising edge is applied to the CTI00 pin according to the timing shown in Figure 8-47, the count value (N) of the 16 high-order bits is held in CPT2H, and 01000000B is held in CPT2L.

Fig. 8-45 Configuration of CPT2 (CPT2H, CPT2L)

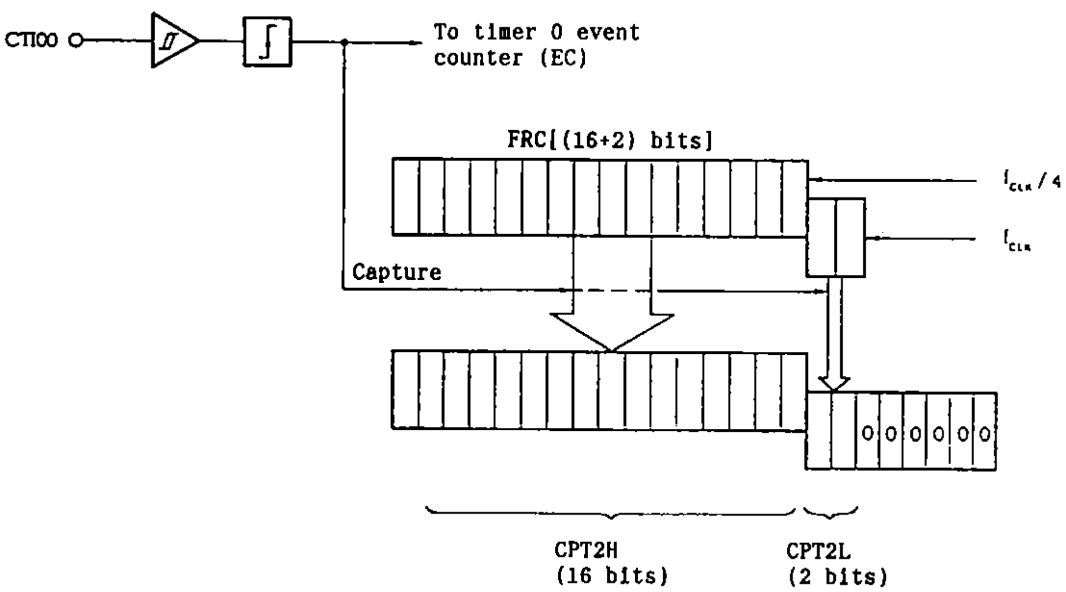
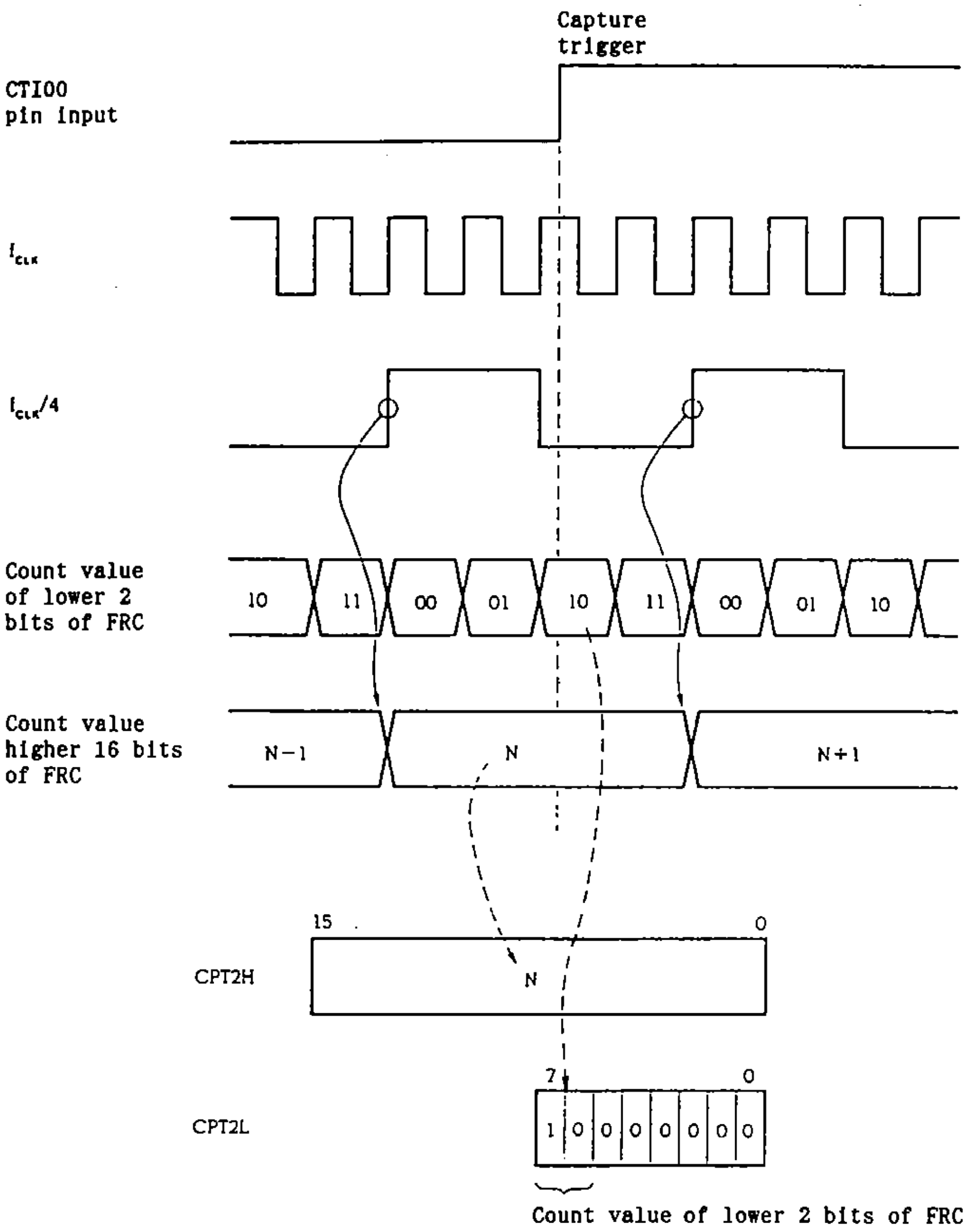


Fig. 8-46 CPT2L Capture Data

	7	6	5	4	3	2	1	0
CPT2L			0	0	0	0	0	0

Lower 2-bit data of FRC

Fig. 8-47 Example of FRC Count Operation


Capture trigger
CTI00 pin input
f_{CLK}
$f_{CLK}/4$
Count value of lower 2 bits of FRC
10
11
00
01
10
11
00
01
10
Count value higher 16 bits of FRC
N−1
N
N+1
15
0
CPT2H
N
7
0
CPT2L
1
0
0
0
0
0
0
0
Count value of lower 2 bits of FRC

8.4.3 Operation of the FRC

Table 8-9 indicates the resolution and maximum count time of the FRC when f_{CLK} = 6 MHz (at 12 MHz).

Table 8-9 Resolution of the FRC (at 12 MHz)

Input clock	Resolution	Maximum count time
6 MHz (f_{CLK})	167 ns when CPT2L is used	43.7 ms
	667 ns when CPT0, CPT1, CPT2H, and CPT3 are used	

The FRC can be started by setting bit 3 of TMC1 to 1.

Bits 1 and 2 (OVF1, OVF2) of TMC1 are the overflow flag of the FRC. Bit 1 (OVF1) is set for the first overflow. For any additional overflows, bit 2 (OVF2) is set. OVF1 and OVF2 cannot be cleared by writing zero, but can be cleared by reading TMC1.

The FRC has a function of generating a timer base interrupt (INTTB).

When bit 10 of the FRC is set to 1, the FRC generates an interrupt in a periodic manner.

Caution: The FRC is undefined for 16 clock pulses ($16/f_{CLK}$) after reset release. Start FRC count operation at the 17th clock pulse or later.

A capture trigger for the CPT0 register of the FRC can be specified using the capture mode register (CPTM).

8.4.4 Setting FRC unit control registers

The operation of the FRC can be controlled using the following registers:

TMC1: Timer control register 1 (for control of FRC operation)

CPTM: Capture mode register (for selection of capture sources)

Fig. 8-48 Format of Timer Control Register 1 (TMC1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TMC1	CS2	0	0	0	CSFRC	OVF2	OVF1	0	FF39H	00H	R/W

OVF2	OVF1	FRC overflow flag
0	0	No overflows occurred.
0	1	One overflow occurred.
1	1	More than one overflow occurred.

CSFRC	FRC count control
0	Clears FRC and stops counting. (No time base interrupt occurs.)
1	Counts. (A time base interrupt occurs.)

CS2	Timer 2 count operation control
0	Clears timer 2 and stops counting.
1	Counts.

Remark: Bits 7 and 3 of TMC1 are used only for write operation, and bits 2 and 1 are used only for read operation.

Fig. 8-49 Format of Capture Mode Register (CPTM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
CPTM	-	-	-	-	-	TRGS 12	TRGS 01	TRGS 00	FF3AH	xxxxx000	W

TRGS 01	TRGS 00	Specification of a CPT0 capture trigger
0	0	Match signal between TM1 and CR10
0	1	CLR1 input edge detection signal
1	0	Not to be set
1	1	Signal obtained by ORing the match signal between TM1 and CR10 with the CLR1 input edge detection signal

TRGS 12	Specification of a CR12 capture trigger
0	CTI11 input edge detection signal
1	CTI10 input division signal

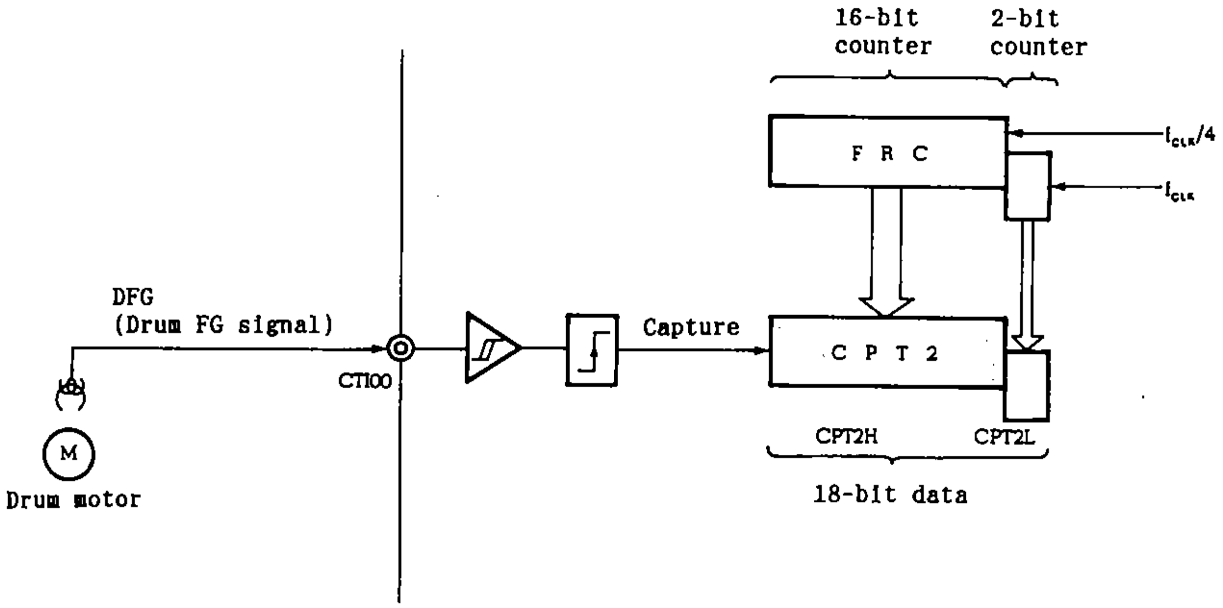
8.4.5 Application of the FRC to a VCR

(1) Application to a drum speed control system

The drum rotation speed control system of a VCR servo system requires highest-precision control. For this control system, the high-precision 18-bit FRC of the uPD78138 and the associated 18-bit capture register CPT2 (CPT2H and CPT2L) can be used.

Specifically, a drum FG (DFG) signal is applied to the CTI00 pin, and a rising edge of this signal is used as a capture trigger to load the count value of the FRC into CPT2 (CPT2H, CPT2L) as shown in Figure 8-50. The value in CPT2 represents the rotation speed of the drum, and this value is used for drum speed control.

Fig. 8-50 Example of Using the FRC in a VCR (Detecting Drum Motor Speed Error Using Capture Register 2)

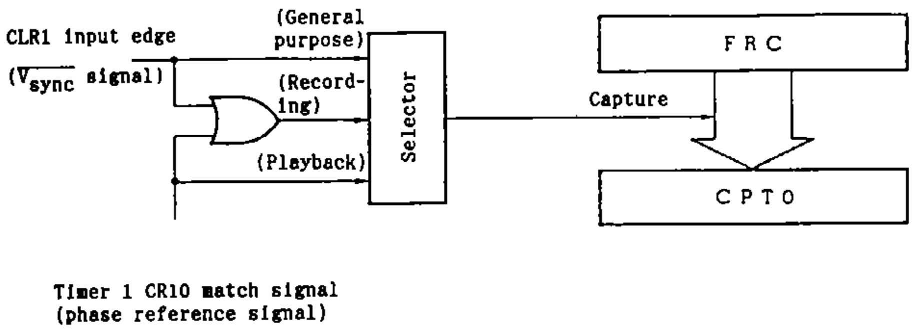


(2) Drum phase control system

For a VCR drum head switching signal and drum phase control, capture register 0 (CPT0) allowing selection from three types of capture triggers is used. Figure 8-51 shows the FRC CPT0 capture trigger configuration. Drum phase control uses a head switching signal as a capture trigger to load the count value of the FRC into capture register 1 (CPT1), and uses a phase reference signal or vertical synchronizing signal (V_{sync}) as a capture trigger to load the count value of the FRC into capture register 0 (CPT0). Then the difference between the value of CPT1 and the value of CPT0 is checked for drum phase control.

In this case, the method of control for playback slightly differs from that for recording.

Fig. 8-51 Example of Using FRC Capture Register 0 (CPT0)



(a) Playback

Timer 1 is used as the reference counter, and a match signal of the compare register (CR10) is used as the phase reference signal. This phase reference signal is used as a capture trigger to load the value of the FRC into CPT0.

Figure 8-52 gives an example of using the FRC and TM1 in playback.

Figure 8-53 shows how the FRC and TM1 operate in playback.

Fig. 8-52 Example of Using the FRC and TM1 in Playback (Operating TM1 as the Internal Phase Reference Timer)

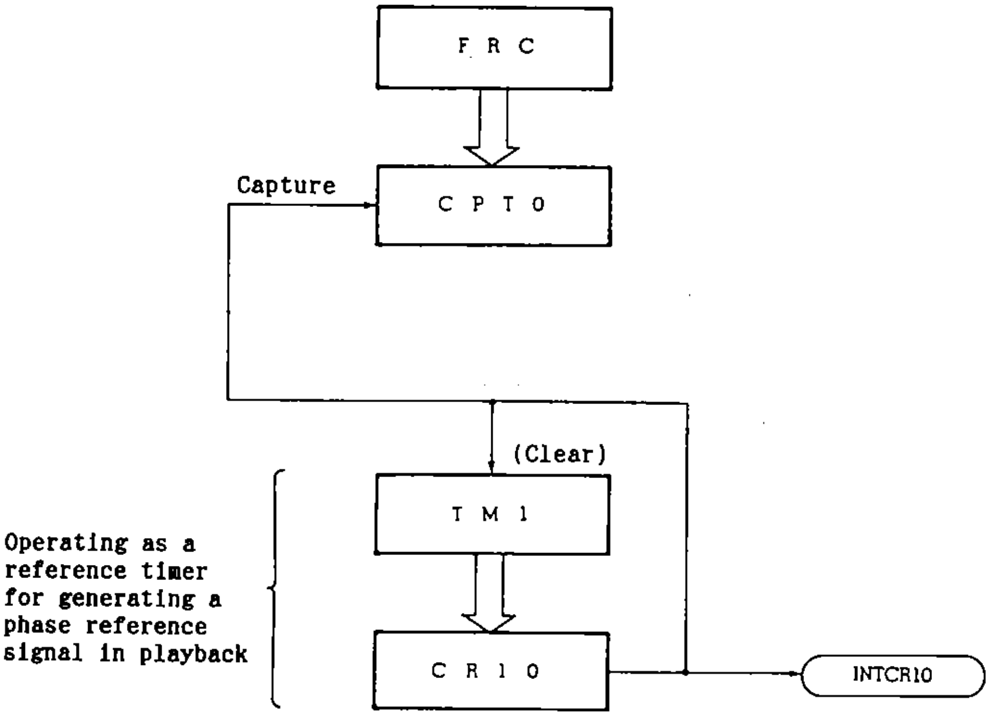
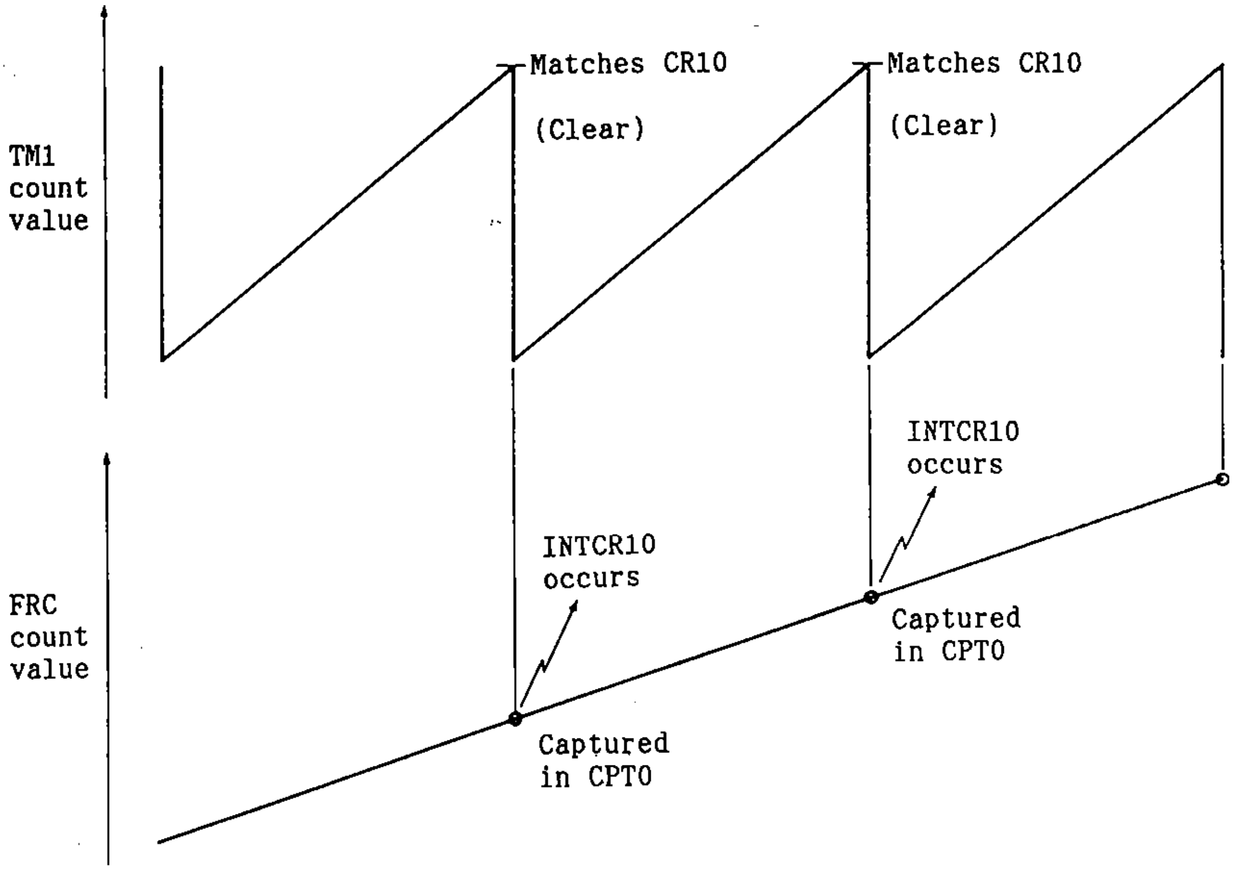

FRC
Capture
CPT0
(Clear)
TM1
Operating as a reference timer for generating a phase reference signal in playback
CR10
INTCR10

Fig. 8-53 Operation of the FRC and TM1 in Playback



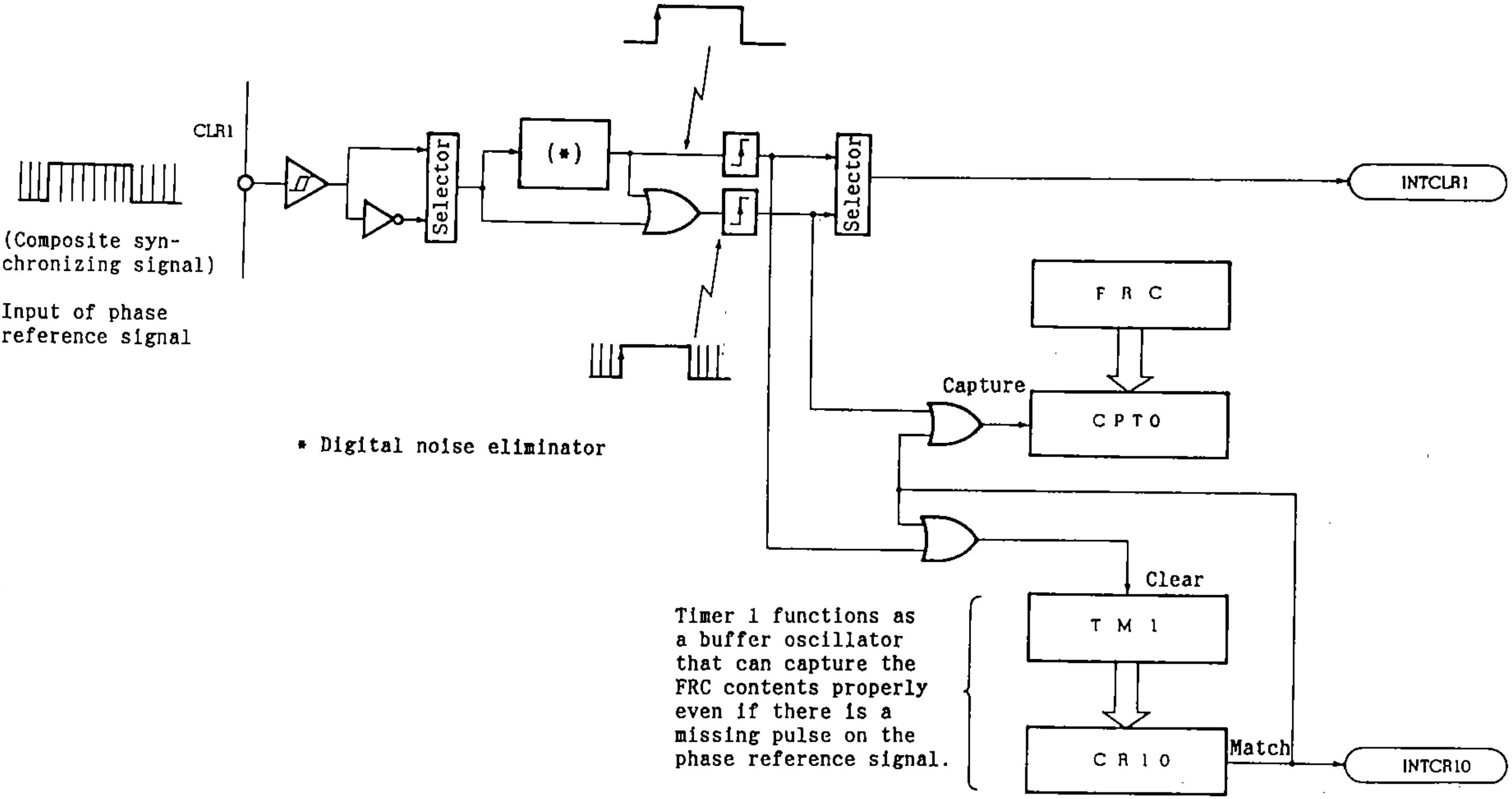
(b) Recording

In recording, the disjunction of a vertical synchronizing (V_{sync}) signal applied to the CLR1 pin and a phase reference signal generated from CR10 of timer 1 is used as a capture trigger. If only a V_{sync} signal is used as a capture trigger, abnormal phase control may result due to V_{sync} lost by noise. The method of using such a disjunction ensures normal phase control. In this case, the period of the V_{sync} signal must be the same as the period of the phase reference signal. This setting ensures normal phase control if the V_{sync} signal is lost for a cause; in this case, the phase reference signal can take place of the V_{sync} signal.

Figure 8-54 gives an example of using the FRC and TM1 in recording.

Figure 8-55 shows how the FRC and TM1 operate in recording.

Fig. 8-54 Example of Using the FRC and TM1 in Recording (Capturing the FRC Contents on Input of the Phase Reference Signal)



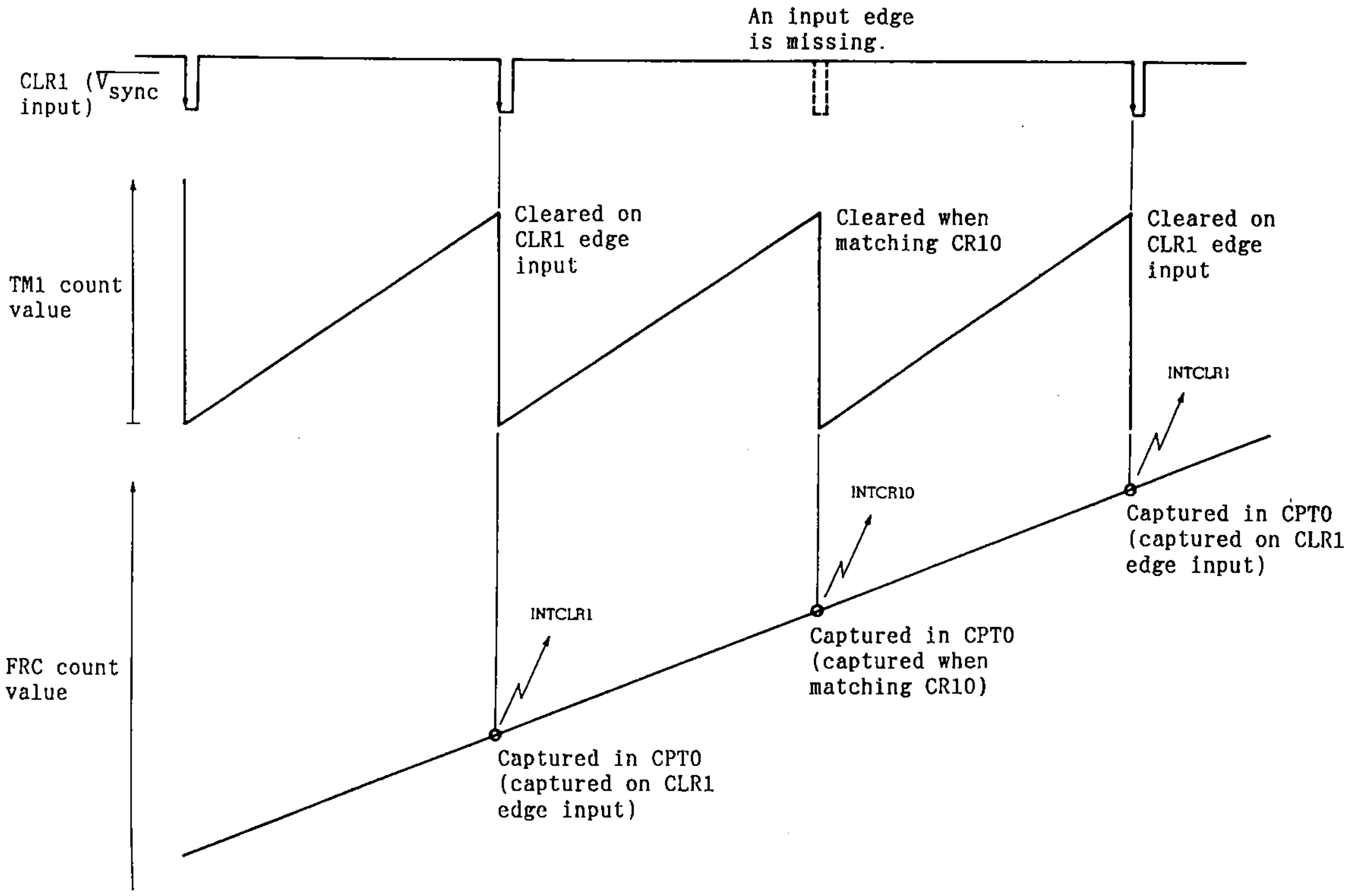
(Composite synchronizing signal)

Input of phase reference signal

* Digital noise eliminator

Timer 1 functions as a buffer oscillator that can capture the FRC contents properly even if there is a missing pulse on the phase reference signal.

Fig. 8-55 Operation of the FRC and TM1 in Recording



Caution: Notes on capturing the FRC value during recording

When attempting phase control as shown in Figure 8-54, the FRC count value captured in CPT0 on the rising edge of a vertical synchronizing signal (V_{sync}) must be saved before the rising edge of an equalizing pulse that follow V_{sync} is input.

That is, the contents of CPT0 must be saved within 200 us after the rising edge of V_{sync} is input.

This must be done because of the following reason.

When a composite synchronizing signal is applied to the CLR1 pin and a V_{sync} signal is extracted in a digital noise eliminator incorporated in the uPD78138 for phase control, the count value of the FRC is captured on the rising edge of an equalizing pulse signal as well as on the rising edge of V_{sync}. However, only the count value captured on the rising edge of V_{sync} is used for phase control. Therefore, to perform phase control correctly, the value of CPT0 must be saved before the rising edge of the equalizing pulse is input.

Time taken from the rising edge of V_{sync} to the rising edge of the equalizing pulse is as follows:

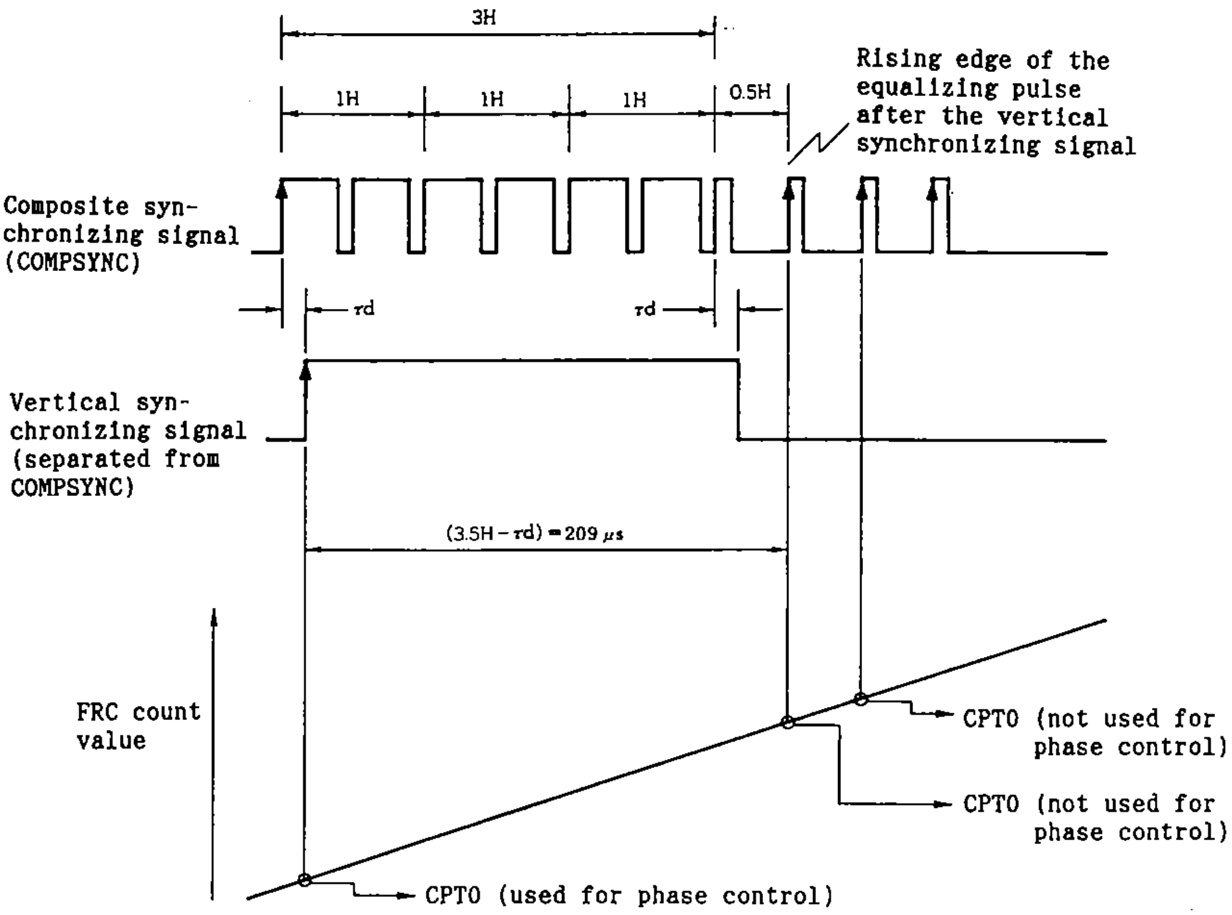
- Pulse width of vertical synchronizing signal (V_{sync}): 3 H
  (H: Horizontal synchronizing signal period; 1 H = 63.55 us)

- Rise time of equalizing pulse after vertical synchronizing signal:
  0.5 H

- Maximum delay time (τd) of digital noise eliminator: 13.3 us

Time taken from rising edge of V_{sync} to rising edge of equalizing pulse
$= (3 + 0.5) \times 63.55 - 13.3$
$\fallingdotseq 209$ us

Fig. 8-56 FRC Capture Operation in Recording (CPT0)


3H
1H
1H
1H
0.5H
Rising edge of the equalizing pulse after the vertical synchronizing signal
Composite syn-chronizing signal (COMPSYNC)
τd
τd
Vertical syn-chronizing signal (separated from COMPSYNC)
(3.5H − τd) = 209 μs
FRC count value
CPT0 (not used for phase control)
CPT0 (not used for phase control)
CPT0 (used for phase control)

Remark: τd: Delay arising from digital sampling. 13.3 us at maximum (when operating at 12 MHz)

H: Horizontal synchronizing signal period
1 H = 63.55 us

8.5 Timer 2 Unit

The timer 2 unit consists of a 16-bit counter (TM2), 16-bit compare register (CR20), and 16-bit comparator, as shown in Figure 8-56.

TM2 is a 16-bit binary up-counter, and is incremented by one each time a counter clock (f_{CLK}/16) pulse is applied. When the value of TM2 matches the value of CR20, TM2 is cleared to 0000H, and a timer interrupt (INTTM) occurs at the same time.

So the timer functions as an interval timer whose interval is determined by the CR20 register.

The count operation of TM2 is controlled by bit 7 (CS2) of timer control register 1 (TMC1). TMC1 is an 8-bit register that allows both read and write operations, but does not allow bit manipulations. A $\overline{\text{RESET}}$ input signal sets TMC1 to 00H (Figure 8-58).

Table 8-10 indicates the interval time of the timer when f_{CLK} = 6 MHz.

Table 8-10 Resolution of Timer 2 (at 12 MHz)

Input clock	Resolution	Full-count interval (CR20 = FFFFH is set.)
375 kHz (f_{CLK}/16)	2.67 us	174.8 ms

TM2 allows only read operation, and CR20 allows both read and write operations. TM2 and CR20 allow access on a 16-bit basis only.

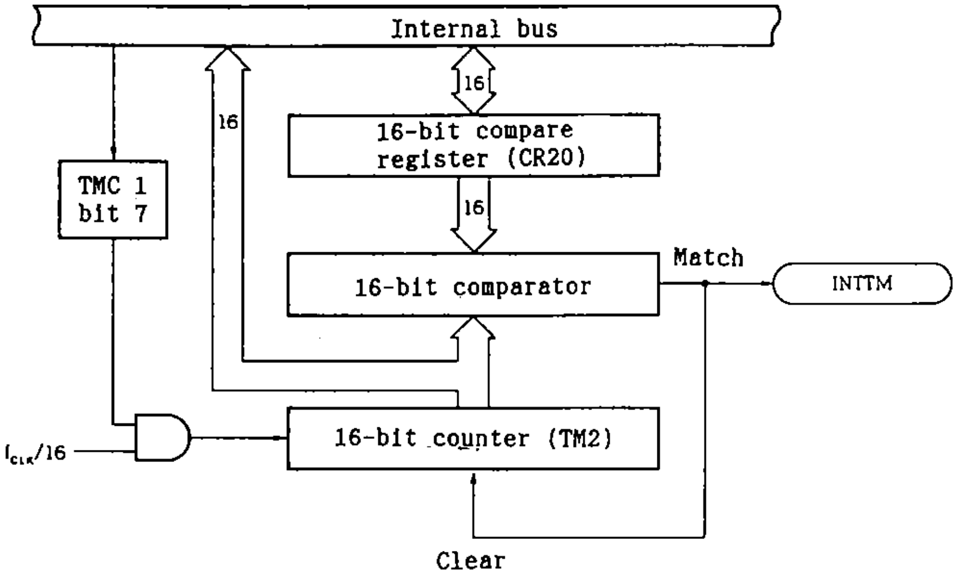
A $\overline{\text{RESET}}$ input signal makes CR20 undefined, and clears TM2 when 16 clock pulses have elapsed after reset release.

Caution: TM2 is undefined for 16 clock pulses ($16/f_{CLK}$) after reset release. Start TM2 timer operation at the 17th clock pulse or later.

8.5.1 Configuration of the timer 2 unit

Figure 8-57 shows the configuration of the timer 2 unit.

Fig. 8-57 Configuration of the Timer 2 Unit



8.5.2 Setting the register to control the timer 2 unit

Operation of the timer 2 unit can be controlled by timer control register 1 (TMC1).

Fig. 8-58 Format of Timer Control Register 1 (TMC1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
TMC1	CS2	0	0	0	CSFRC	OVF2	OVF1	0	FF39H	00H	R/W

OVF2	OVF1	FRC overflow flag
0	0	No overflow occurred.
0	1	One overflow occurred.
1	1	More than one overflow occurred.

CSFRC	Controls counting of the FRC.
0	Clears and stops counting. (Does not generate a time base interrupt.)
1	Counts. (Generates a time base interrupt.)

CS2	Controls counting of timer 2.
0	Clears and stops counting.
1	Counts.

8.6 PWM Output Unit

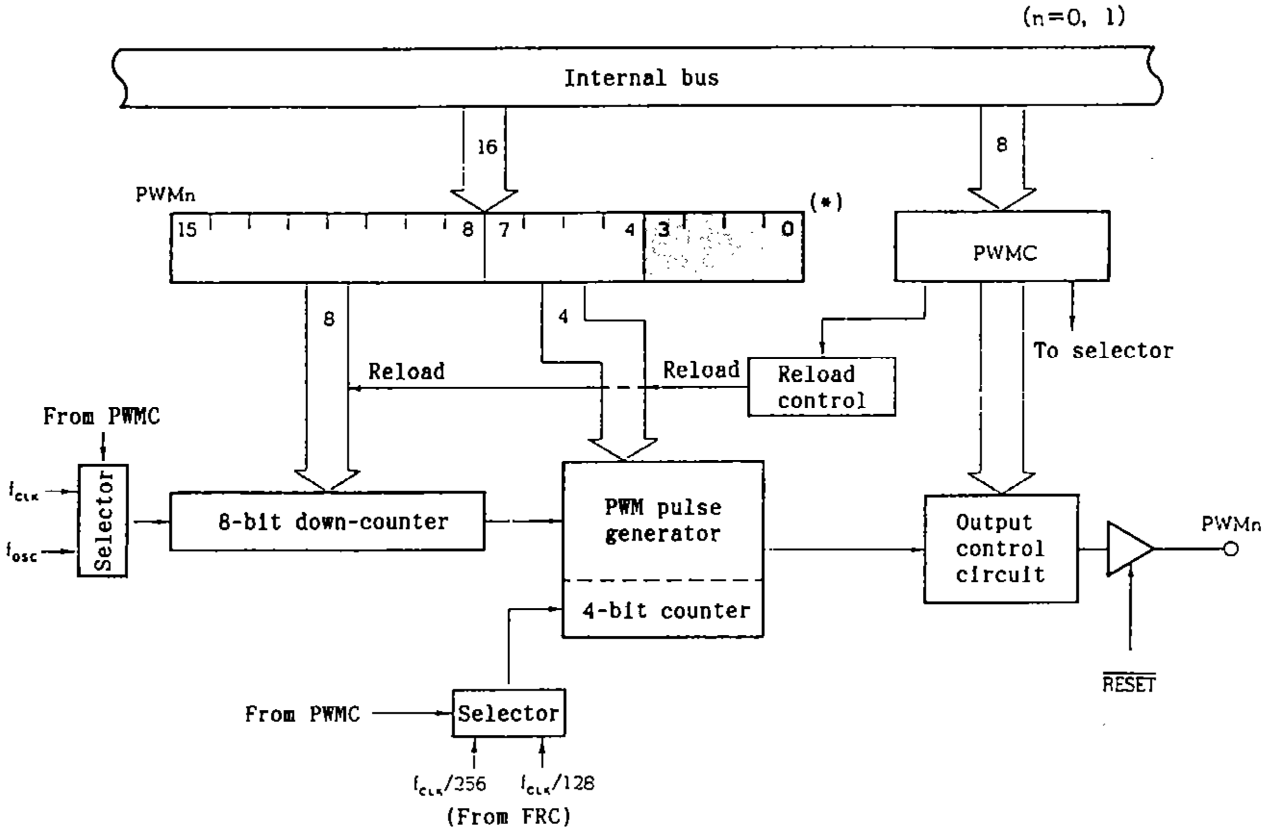
The uPD78138 contains two 12-bit pulse width modulation (PWM) output circuits. The PWM output unit allows the user to select either the 23.4- or 46.9-kHz carrier frequency for PWM output. It also allows selection between active high or active low for the active level of a PWM output pulse signal.

In addition, the PWM output port can be used as a general output port.

8.6.1 Configuration of the PWM output unit

Figure 8-59 shows the configuration of the PWM output unit.

Fig. 8-59 Configuration of the PWM Output Unit



* See Section 8.6.4.

Remark: $f_{CLK} = f_{OSC}/2$

8.6.2 Operation of the PWM output unit

(1) PWM pulse output enable/disable

The PWM output unit allows the user to select either 23.4- or 46.9-kHz carrier frequency (PWM pulse repetition interval) for PWM output (at 12 MHz). The PWM pulse width is determined by the contents of the PWM modulo register (PWM0, PWM1).

When a PWM pulse signal is to be output, data must be set in the PWM modulo registers, then the EN0 and EN1 bits of the PWMC register must be set to 1.

With these settings, PWM pulses with the active level specified by ALV0 and ALV1 of the PWMC register are output on the PWM output pins.

When the EN0 and EN1 bits of the PWMC register are cleared to 0, the PWM output unit immediately stops PWM output operation; and the inactive level appears on the PWM output pins.

Caution: The PWM output control circuit operates with a clock signal supplied from the free running counter (FRC). So PWM output operation cannot be performed when the FRC is not in operation. The FRC performs count operation when bit 3 of timer control register 1 (TMC1) is set to 1 (Figure 8-48).

(2) Active level specification for a PWM pulse signal

The ALV0 and ALV1 bits of the PWMC register specify the active level of PWM pulse signals output on the PWM output pins.

When ALV0 and ALV1 are set to 1, an active high pulse signal is output. When these bits are reset to 0, an active low pulse signal is output.

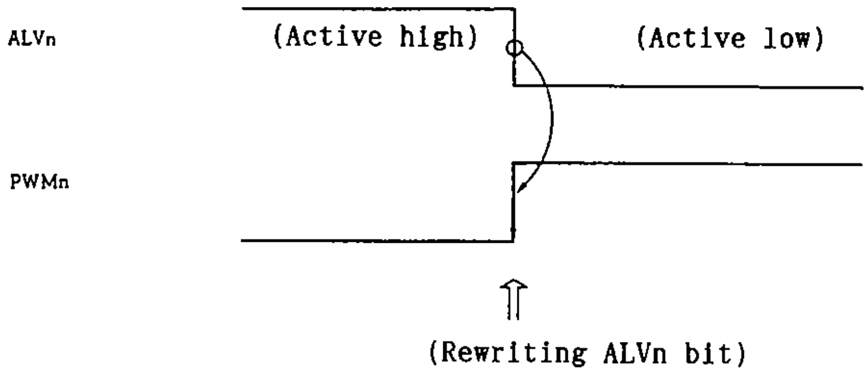
When the settings of ALV0 and ALV1 are changed, the PWM active level immediately changes. Figure 8-60 shows the active level setting of PWM output and pin state.

In Figure 8-60(a), the setting of ALVn (n = 0, 1) is changed when the ENn (n = 0, 1) of PWMC is reset to 0, and PWM output is disabled. When PWM output is disabled, the inactive level appears on the PWM pulse output pins. So by rewriting ALVn (n = 0, 1), the PWMn pin (n = 0, 1) can be used as a general output port.

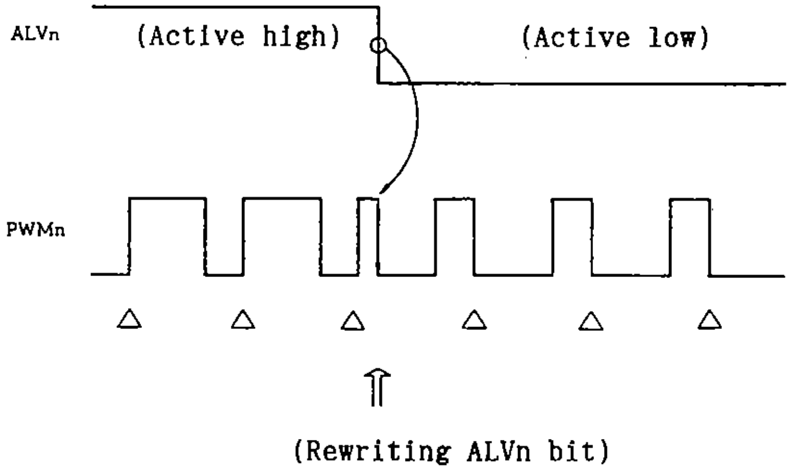
In Figure 8-60(b), the setting of ALVn (n = 0, 1) is changed when the ENn (n = 0, 1) of the PWMC register is set to 1, and PWM output is enabled.

Fig. 8-60 Active Level Setting for PWM Output

(a) When PWM pulse output is disabled (ENn = 0: n = 0, 1)



(b) When PWM pulse output is enabled (ENn = 1: n = 0, 1)



(3) Specification of a PWM pulse width switching cycle

PWM output is started and the pulse width is changed every 16 PWM pulse cycles (2^{12}/PWM operating frequency) or for each PWM pulse cycle (2^8/PWM operating frequency). A PWM pulse width switching cycle can be specified with the SYNn bit (n = 0, 1) of the PWMC register.

When the SYNn bit (n = 0, 1) is reset to 0, pulse width switching is performed every 16 PWM pulse cycles (2^{12}/PWM operating frequency). This means that up to 2^{12} clock pulses (342 us when PWM operating frequency = 12 MHz) are required before pulses with the width corresponding to the data loaded into the PWM modulo register are output.

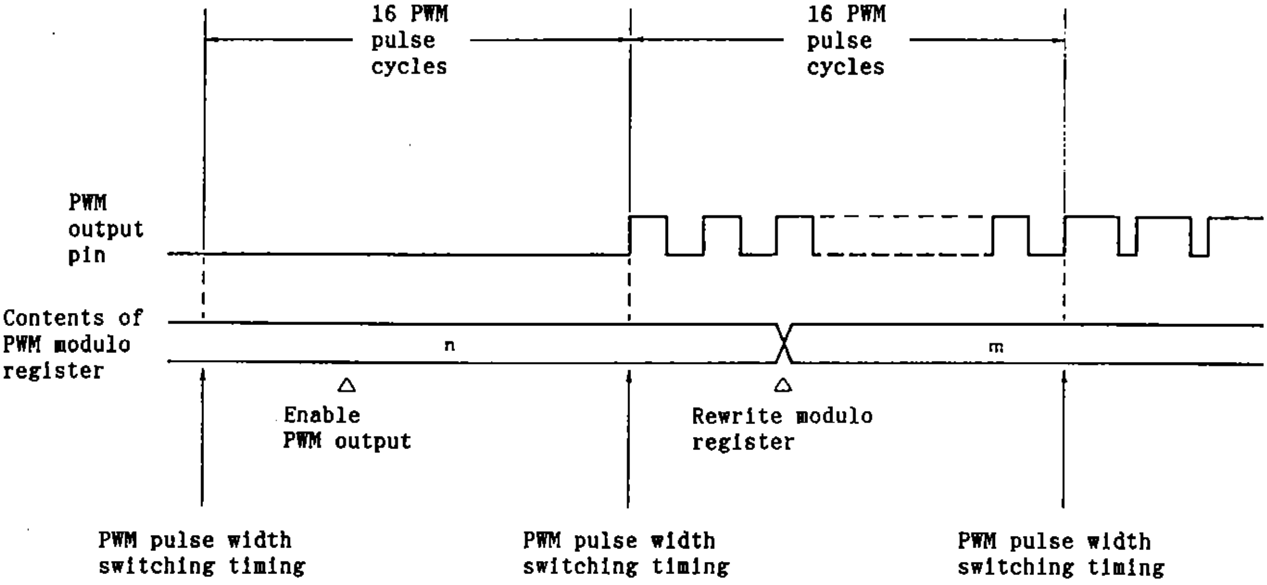
Figure 8-61 shows an example of PWM output timing.

On the other hand, when the SYNn bit (n = 0, 1) is set to 1, pulse width switching is performed for each pulse cycle (2^8/PWM operating frequency). In this case, up to 2^8 clock pulses (42 us when PWM operating frequency = 12 MHz) are required before pulses with the width corresponding to the data loaded into the PWM modulo register are output.

However, when 2^8/PWM operating frequency is specified as a pulse width switching cycle (that is, when the SYNn bit is set to 1), note that a pulse width resolution from eight bits to 12 bits is obtained. This resolution is inferior to a resolution obtained when 2^{12}/PWM operating frequency is specified.

Figure 8-62 shows an example of PWM output timing when a switching cycle of 2^8/PWM operating frequency is specified.

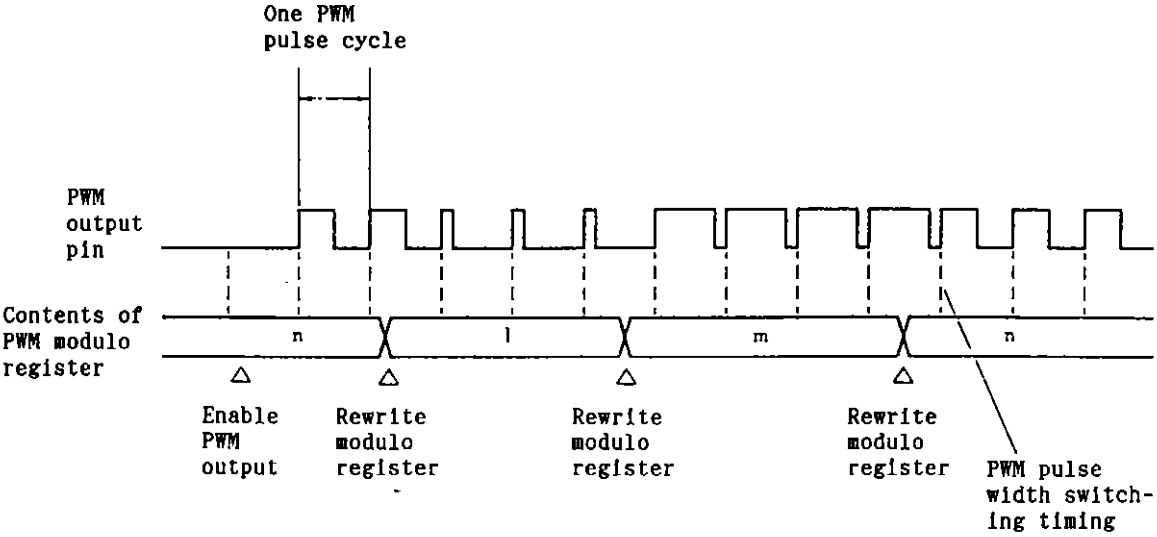
Fig. 8-61 Example 1 of PWM Output Timing (PWM Pulse Width Switching Cycle: $2^{12}/f_{CLK}$)


16 PWM pulse cycles
16 PWM pulse cycles
PWM output pin
Contents of PWM modulo register
n
m
Enable PWM output
Rewrite modulo register
PWM pulse width switching timing
PWM pulse width switching timing
PWM pulse width switching timing

Remarks 1. Pulse width switching is performed every 16 PWM pulse cycles.

2. The PWM pulse resolution is 12-bit.

Fig. 8-62 Example 2 of PWM Output Timing (PWM Pulse Width Switching Cycle: $2^8/f_{CLK}$)



Remarks 1. Pulse width switching is performed for each PWM pulse cycle.

2. The PWM pulse resolution is from 8-bit to 12-bit within 16 PWM pulse cycles after the PWM modulo register is rewritten.

3. The value n, m, or l represents the contents of the PWM modulo register.

(4) PWM pulse width

The PWM pulse width is determined by the 12-bit data from bit 15 to bit 4 of a PWM modulo register.

If the high-order eight bits of the PWM modulo register (bits 15 to 8) shows 00H, no PWM pulse signal is output regardless of the value set in the low-order four bits (bits 7 to 4). Be sure to set the PWM modulo register to a value not less than 0100H. The duty cycle of the PWM output with a particular pulse width is expressed by the following:

Duty cycle for a pulse width (%) = ((value in PWM modulo register bits 15 to 4) + 1)/2^{12} x 100

where,

the value of PWM modulo register bits 15 to 4 $\geq$ 010H

Caution: The PWM output control circuit operates with a clock signal supplied from the free running counter (FRC). So PWM pulse output operation is not performed when the FRC is not in operation.

Setting bit 3 of the timer control register 1 (TMC1) to 1 causes the FRC to start counting.

8.6.3 Setting the register to control PWM output unit

Operation of the PWM output units can be controlled by the following register:

PWMC: PWM control register, which controls operation of PWM0 and PWM1

The format of the register is shown below.

Fig. 8-63 Format of the PWM Control Register (PWMC)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PWMC	SYN1	CLS1	SYN0	CLS0	EN1	ALV1	EN0	ALV0	FF70H	05H	R/W

(n = 0, 1)

ALVn	Setting of PWM output active level on PWMn pin
0	Active low
1	Active high

ENn	Control over PWM output on PWMn pin
0	Disables output, with pin level set to inactive level.
1	Enables PWM output.

CLSn	Selection of PWM operating frequency
0	f_{CLK} (carrier frequency: 23.4 kHz)(*)
1	f_{OSC} (carrier frequency: 46.9 kHz)(*)

* When operating at 12 MHz

SYNn	Specification of PWM pulse width switching cycle
0	Switches width every 16 PWM pulse cycles ($2^{12}/f_{CLK}$).
1	Switches width for each PWM pulse cycle ($2^8/f_{CLK}$).

Remark: f_{OSC}: External oscillator frequency
f_{CLK}: Internal system clock ($f_{CLK} = f_{OSC}/2$)

8.6.4 Registers other than the control register

PWM modulo registers (PWM0, PWM1)

The PWM0 and PWM1 registers are 16-bit registers that determine the pulse width of a PWM pulse signal. Data can be set using a 16-bit data transfer instruction.

The registers are write-only registers, and do not allow read operation.

Bits 15 to 4 of the PWM0 and PWM1 registers determine a 12-bit PWM pulse width (12-bit resolution). Bits 3 to 0 are ignored; PWM output is not affected when 1 or 0 is written to these bits.

A $\overline{\text{RESET}}$ input signal makes the contents of the modulo registers undefined, so data must be set using the initialization program before PWM output is enabled.

8.6.5 Application of the PWM output units

Application of the PWM output units to a VCR

Since PWM0 and PWM1 feature high-speed, high-resolution PWM output, they are suitable where real-time processing and high precision are critical.

In a VCR unit, PWM0 and PWM1 can be used for driving a drum motor and capstan motor.

CHAPTER 9 A/D CONVERTER

9.1 Functions of A/D Converter

The uPD78138 contains a built-in analog-to-digital (A/D) converter with eight multiplexed analog inputs (ANI0 to ANI7).

Analog-to-digital conversion is done by successive approximation. The converted result is stored in the 8-bit A/D conversion result register (ADCR). Fast and very accurate conversion is enabled (conversion requires only 30 us when the system operates at 12 MHz).

A/D conversion is started in one of the following modes:

o Hardware start: Conversion is started by trigger input (INTP1).

o Software start: Conversion is started by setting an appropriate bit in the A/D conversion mode register (ADM).

After started, conversion is done in one of the following modes:

o Scan mode: Selects analog inputs sequentially for conversion, and obtains digital data converted from analog inputs on the all pins.

o Select mode: Converts analog input on a particular pin continuously.

The above modes and the stop of conversion are specified with ADM.

When a converted result is transferred to ADCR, interrupt request INTAD is generated (except the select mode when conversion is started in the software start mode). Macro service can therefore transfer converted results to memory successively.

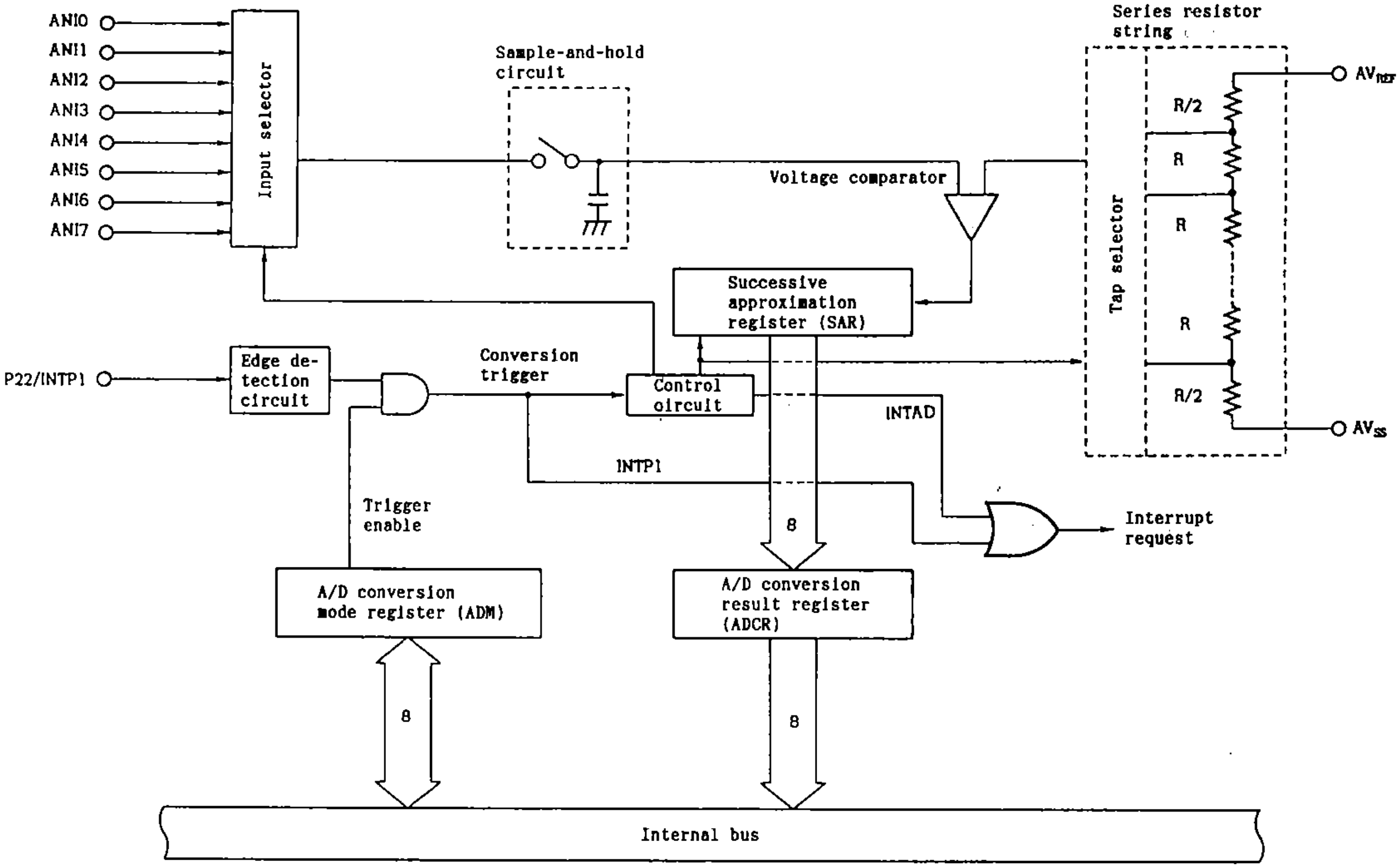
Table 9-1 Mode Generating INTAD

Mode Start	Scan mode	Select mode
Hardware start	Generated	Generated
Software start	Generated	Not generated

9.2 Hardware Configuration of A/D Converter

Figure 9-1 shows the configuration of the A/D converter.

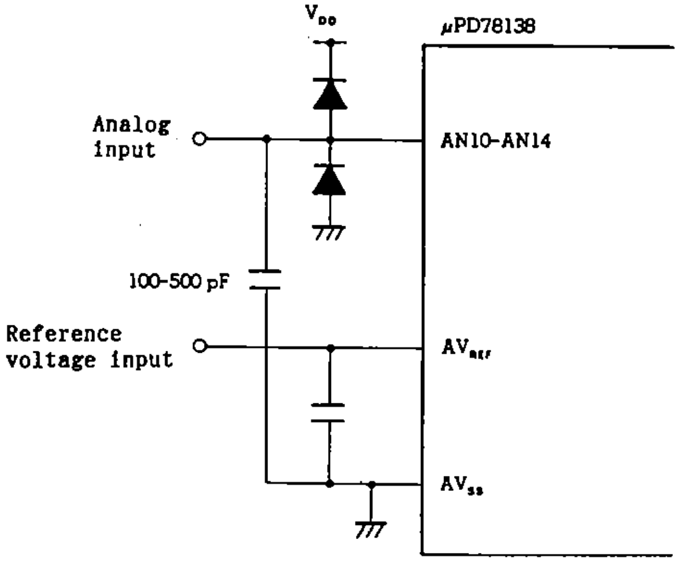
Fig. 9-1 Block Diagram of A/D Converter



Cautions 1. Connect capacitors to analog input pins (ANI0 to ANI7) and reference voltage input pin (AV_{REF}) to prevent miss-operation due to noise.

2. Be careful not to apply voltage exceeding the range from AV_{SS} to AV_{REF} to the ANI0 to ANI7 pins during A/D conversion or when these pins are not used. When it is possible that a noise with a voltage above AV_{REF} or below AV_{SS} is applied, clamp the pins with diodes with low V_F.

Fig. 9-2 Example of Connecting Capacitors to A/D Converter Pins



(1) Input circuit

The input circuit selects an analog input as specified in the A/D conversion mode register (ADM), and sends the analog input to the sample-and-hold circuit according to the current operation mode.

(2) Sample-and-hold circuit

The sample-and-hold circuit samples each of analog inputs sent successively, and holds the analog input being converted to a digital form.

(3) Voltage comparator

The voltage comparator compares the analog input with the voltage at a voltage tap in the series resistor string.

(4) Series resistor string

The series resistor string generates voltage steps for converting analog input into a digital form.

The series resistor string is connected between the reference voltage pin (AV_{REF}) and GND pin (AV_{SS}) for the A/D converter. The string consists of 255 equivalent resistors and two resistors having a half of the resistance of the 255 resistors so that 256 voltage levels can be produced in equivalent steps between the two pins.

One of the voltage taps in the series resistor string is selected by the tap decoder controlled by SAR.

(5) Successive approximation register (SAR)

SAR is an 8-bit register to accept a result of comparing the voltage at a voltage tap in the series resistor string with the voltage of the analog input bit-wise from the most significant bit (MSB).

When comparison results are set down to the least significant bit (LSB) of SAR, A/D conversion is terminated, and the conversion result in SAR is transferred to the A/D conversion result register (ADCR) and held there. At the same time, an A/D conversion termination interrupt request (INTAD) is generated from SAR.

(6) Edge detection circuit

The edge detection circuit detects a valid edge in the input on the interrupt request input pin (INTP1), then generates an external interrupt request signal (INTP1) and an external trigger for A/D conversion.

The valid edge of the INTP1 pin input is specified by the external interrupt mode register 0 (INTM0). (See Figure 11-11.) The external trigger is enabled or disabled by the ADM register. (See Section 9.3.)

9.3 Control Register for the A/D Converter

The A/D converter is controlled by the A/D conversion mode register (ADM).

The ADM register is an 8-bit register that controls the operation of the A/D converter.

Eight-bit manipulation instructions and bit manipulation instructions can be used to read from or write to the register. Figure 9-3 shows the format of the ADM register.

Bit 0 (MS) controls the operation mode.

Bits 1, 2, and 3 (ANIS0, ANIS1, and ANIS2) select analog input to be converted into digital form.

Bit 6 (TRG) enables A/D conversion to be synchronized with an external signal. If the CS bit is 1, setting the TRG bit initializes conversion operation every time a valid edge is received on the INTP1 pin as an external trigger. Resetting the TRG bit to 0 leaves conversion continuing until it is terminated regardless of the INTP1 pin input.

Bit 7 (CS) controls A/D conversion. If the CS bit is set to 1, conversion starts, and if the bit is reset to 0, entire conversion operation is stopped even when conversion is being in progress. In this case, requests for ADCR register update and INTAD interrupt are not generated.

$\overline{\text{RESET}}$ input sets the ADM register to 00H.

Fig. 9-3 Format of A/D Conversion Mode Register (ADM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
ADM	CS	TRG	0	FR	ANIS2	ANIS1	ANIS0	MS	FF68H	00H	R/W

ANIS2	ANIS1	ANIS0	MS	Specification of A/D conversion mode	
0	0	0	0	Scan mode	Scan ANI0 input.
0	0	1	0		Scan ANI0 and ANI1 inputs.
0	1	0	0		Scan ANI0-ANI2 inputs.
0	1	1	0		Scan ANI0-ANI3 inputs.
1	0	0	0		Scan ANI0-ANI4 inputs.
1	0	1	0		Scan ANI0-ANI5 inputs.
1	1	0	0		Scan ANI0-ANI6 inputs.
1	1	1	0		Scan ANI0-ANI7 inputs.
0	0	0	1	Select mode	Select ANI0 input.
0	0	1	1		Select ANI1 input.
0	1	0	1		Select ANI2 input.
0	1	1	1		Select ANI3 input.
1	0	0	1		Select ANI4 input.
1	0	1	1		Select ANI5 input.
1	1	0	1		Select ANI6 input.
1	1	1	1		Select ANI7 input.

FR	Conversion speed control	
0	180 states	When oscillator frequency > 8 MHz
1	120 states	When oscillator frequency ≤ 8 MHz

TRG	External pin trigger control
0	External trigger disabled
1	External trigger enabled

CS	A/D conversion control
0	Stop A/D conversion.
1	Start A/D conversion.

9.4 A/D Converter Operation

9.4.1 Basic operation of A/D converter

Analog-to-digital conversion is performed in the following procedure:

(1) Select an analog input and specify an operation mode with the A/D conversion mode register (ADM).

(2) Set bit 7 (CS) in the ADM register to 1 to start A/D conversion.

(3) As conversion starts, the most significant bit in SAR (bit 7) is automatically set to 1.

(4) As bit 7 in SAR is set, the tap decoder selects a voltage tap in the series resistor string so that the voltage level is (1/2)· AV_{REF}.

(5) The voltage comparator compares the voltage at the tap in the series resistor string with the voltage of analog input. If the voltage of the analog input is higher than (1/2) AV_{REF}, the MSB of SAR is left set. If the voltage is less than (1/2) AV_{REF}, the MSB is reset.

(6) Next, bit 6 in SAR is automatically set to 1, proceeding to the next comparison. One of the following voltage taps in the series resistor string is selected according to the value set in bit 7:

- Bit 7 = 1: (3/4) AV_{REF}
- Bit 7 = 0: (1/4) AV_{REF}

The voltage comparator compares the voltage at the selected voltage tap with the voltage of the analog input. Bit 6 in SAR is set depending on the result of comparison as follows:
Analog input voltage ≥ Voltage tap:
Bit 6 is set to 1.
Analog input voltage < Voltage tap:
Bit 6 is set to 0.

(7) Comparison continues in the same way down to the least-significant bit (bit 0) (called the binary search method).

(8) When comparison is terminated for the eight bits, SAR has held a valid result in digital form. This value is transferred to the ADCR register and latched in it.

At the same time, an A/D conversion end interrupt request (INTAD) is generated. INTAD must be processed as a vectored interrupt or macro service. (See Figure 9-4.)

Caution: Be careful not to apply voltage exceeding the range from AV_{SS} - 0.3 to AV_{REF} + 0.3 to the ANI0 to ANI7 pins during A/D conversion or when they are not used.

The start of A/D conversion can be synchronized with an external signal. When the CS bit is set to 1 after bit 6 (TRG) in the ADM register is set to 1 by software, an external signal ready state is entered. Every time a valid edge is applied to the INTP1 pin, initialization is performed, and A/D conversion starts. (See Figure 9-5.) A/D conversion continues until the CS bit is reset to 0 by software.

When the ADM register is written to during A/D conversion, the conversion operation is initialized and started from the beginning. (See Figure 9-6.)

$\overline{\text{RESET}}$ input makes the ADCR register contents indefinite.

Fig. 9-4 Basic Operation of A/D Conversion

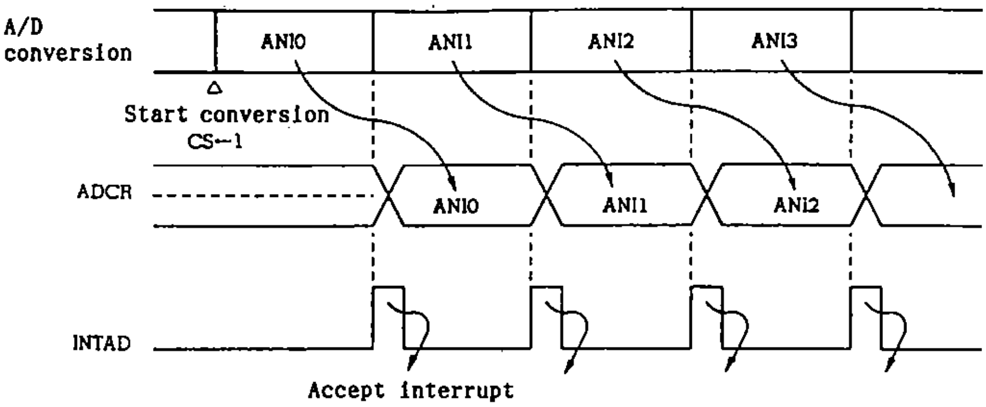


Fig. 9-5 A/D Conversion Started by Hardware

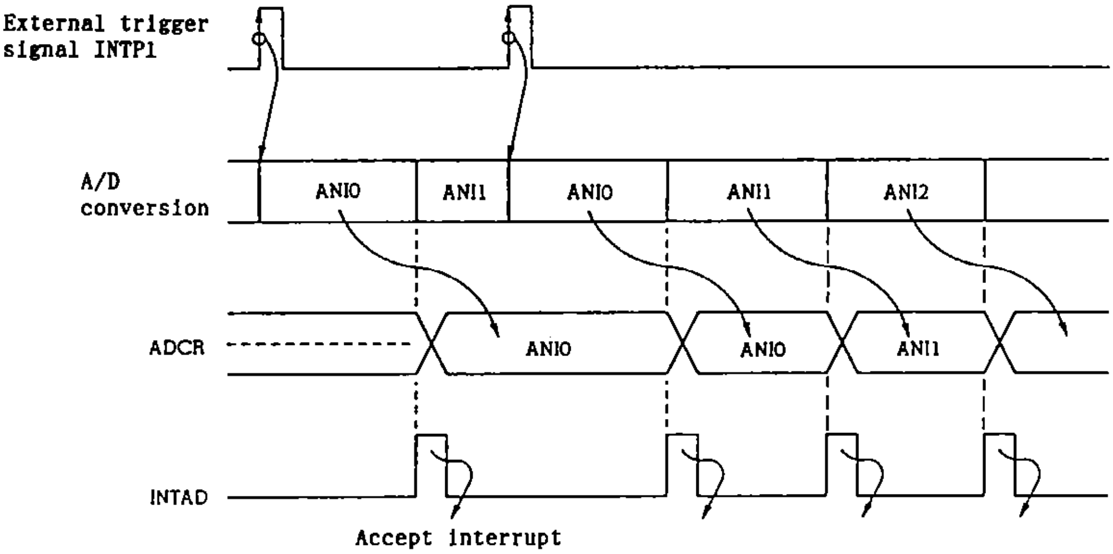
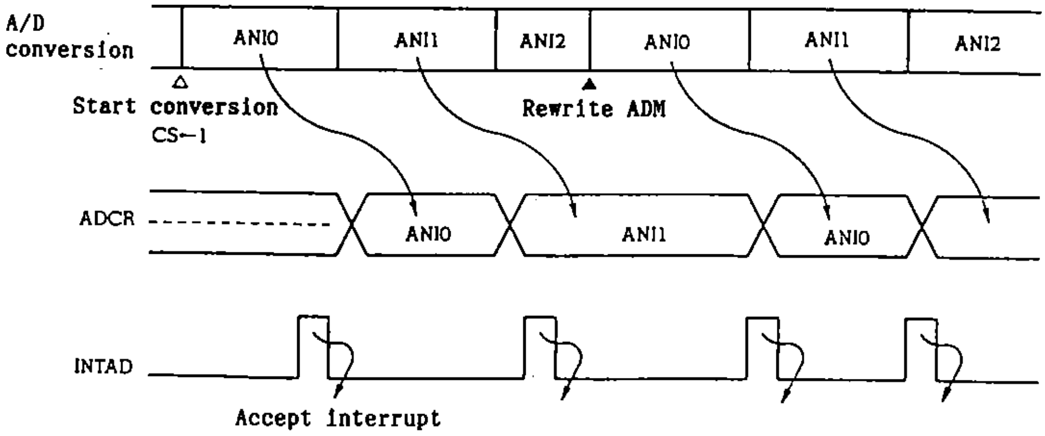


Fig. 9-6 Rewriting ADM Contents during A/D Conversion



9.4.2 A/D converter operation mode

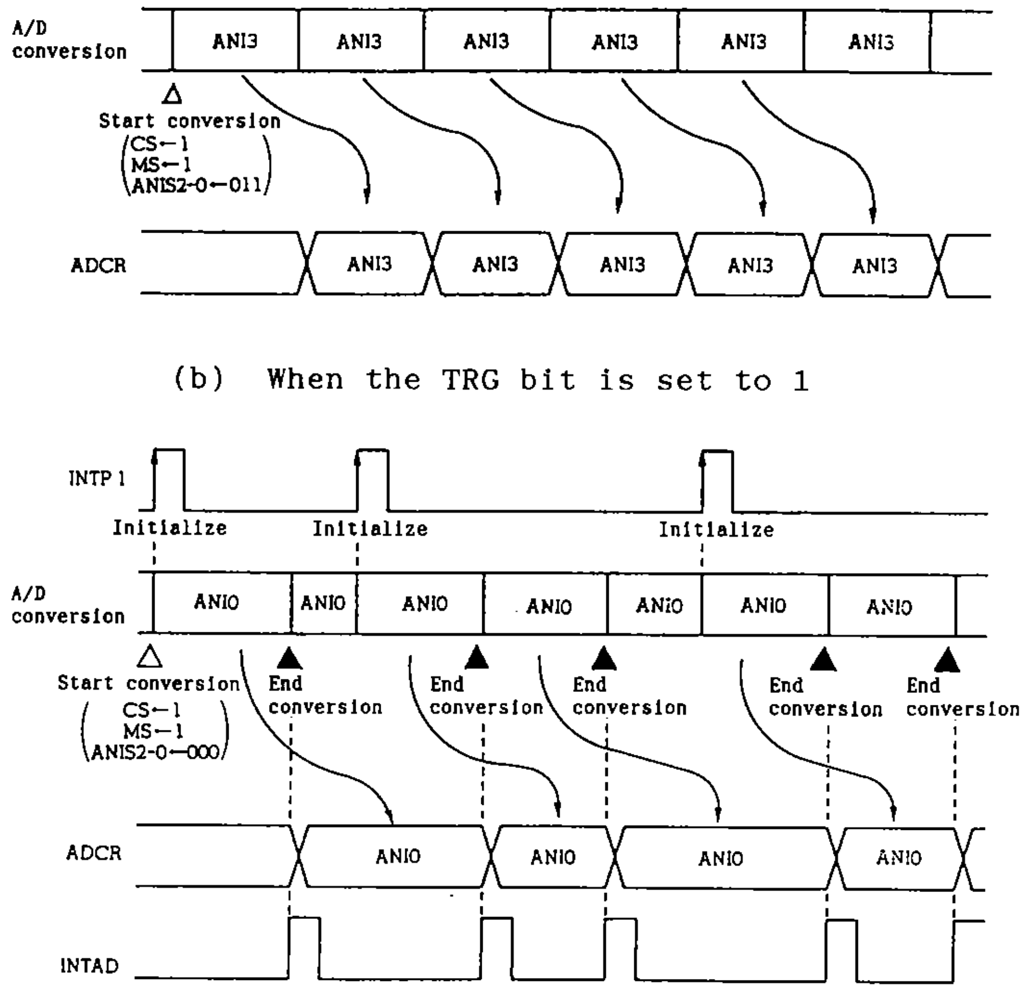
The A/D converter operates in either the scan mode or select mode. The mode is selected by bit 0 (MS) in the A/D conversion mode register (ADM). The selected mode continues until the ADM register contents are rewritten.

(1) Select mode

One analog input is specified with bits 1 to 3 (ANI0 to ANI2) in the ADM register for starting A/D conversion. The conversion result is stored in the A/D conversion result register (ADCR).

If bit 6 (TRG) in the ADM register is set to enable external trigger, an A/D conversion end interrupt request (INTAD) is generated.

Fig. 9-7 Operation Timing in Select Mode



(a) When the TRG bit is set to 0

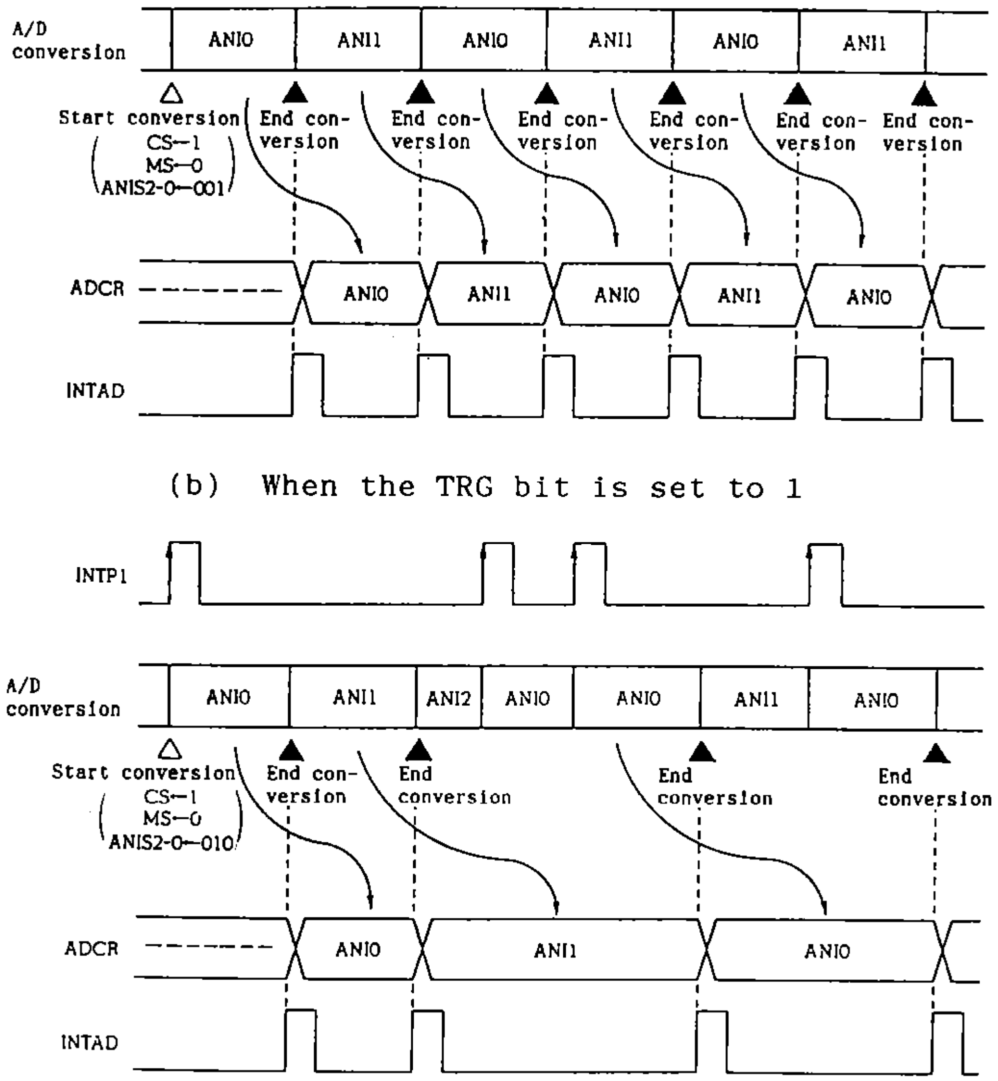
(b) When the TRG bit is set to 1

(2) Scan mode

The scan mode converts the inputs on the analog input pins specified by bits 1 to 3 (ANIS0 to ANIS2) in the A/D conversion mode register (ADM) sequentially.

For example, if ANIS2 to ANIS0 in the ADM register is 001, ANI0 and ANI1 are scanned repeatedly from ANI0 to ANI1 to ANI0 to ANI1... In the scan mode, every time an input has been converted, the converted value is stored in the ADCR register, and an A/D conversion end interrupt request (INTAD) is generated.

Fig. 9-8 Operation Timing in Scan Mode



(a) When the TRG bit is set to 0

(b) When the TRG bit is set to 1

9.5 A/D Converter Interrupt Request

The A/D converter generates an A/D conversion end interrupt request (INTAD) every time A/D conversion is terminated, except in the select mode.

The control flags associated with INTAD also function as control flags associated with external interrupt request INTP1. The timing of generating an interrupt request, therefore, differs according to the A/D converter operation state specified by the ADM register, as listed in Table 9-2.

Interrupt service caused by INTAD is controlled with interrupt control registers in the same way as for INTP1. For details, see Chapter 11.

Table 9-2 Conditions for Generating Interrupt Requests in Different A/D Converter Operating States

A/D converter	Interrupt request flag	Mask flag	Interrupt request	Interrupt request condition
In stopped state	PIF1	PMK1	INTP1	Valid edge input on INTP1 pin
Scan mode			INTAD	When A/D conversion is terminated
Select mode			INTP1	Valid edge input on INTP1 pin
A/D conversion started by hardware			INTAD	When A/D conversion is terminated

Caution: Handling of an A/D converter interrupt request (INTAD)

An A/D converter interrupt request is generated with a period of approximately 30 us when the system operates at 12 MHz. In an application where motor digital servo control is done by software, mask the INTAD associated with the current operation. Otherwise, an INTAD may frequently occur during operation with the servo system, which adversely affects the servo characteristics.

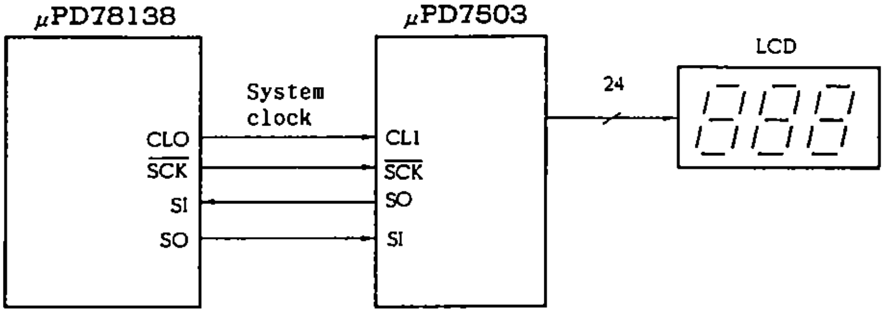
CHAPTER 10 CLOCK OUTPUT (CLO)

10.1 Configuration and Functions of CLO

A square wave with a 50% duty cycle can be output on the ASTB/CLO pin to clock a peripheral device or another microcomputer. The clock output mode register (CLOM) determines whether clock output is enabled or disabled, and sets the frequency.

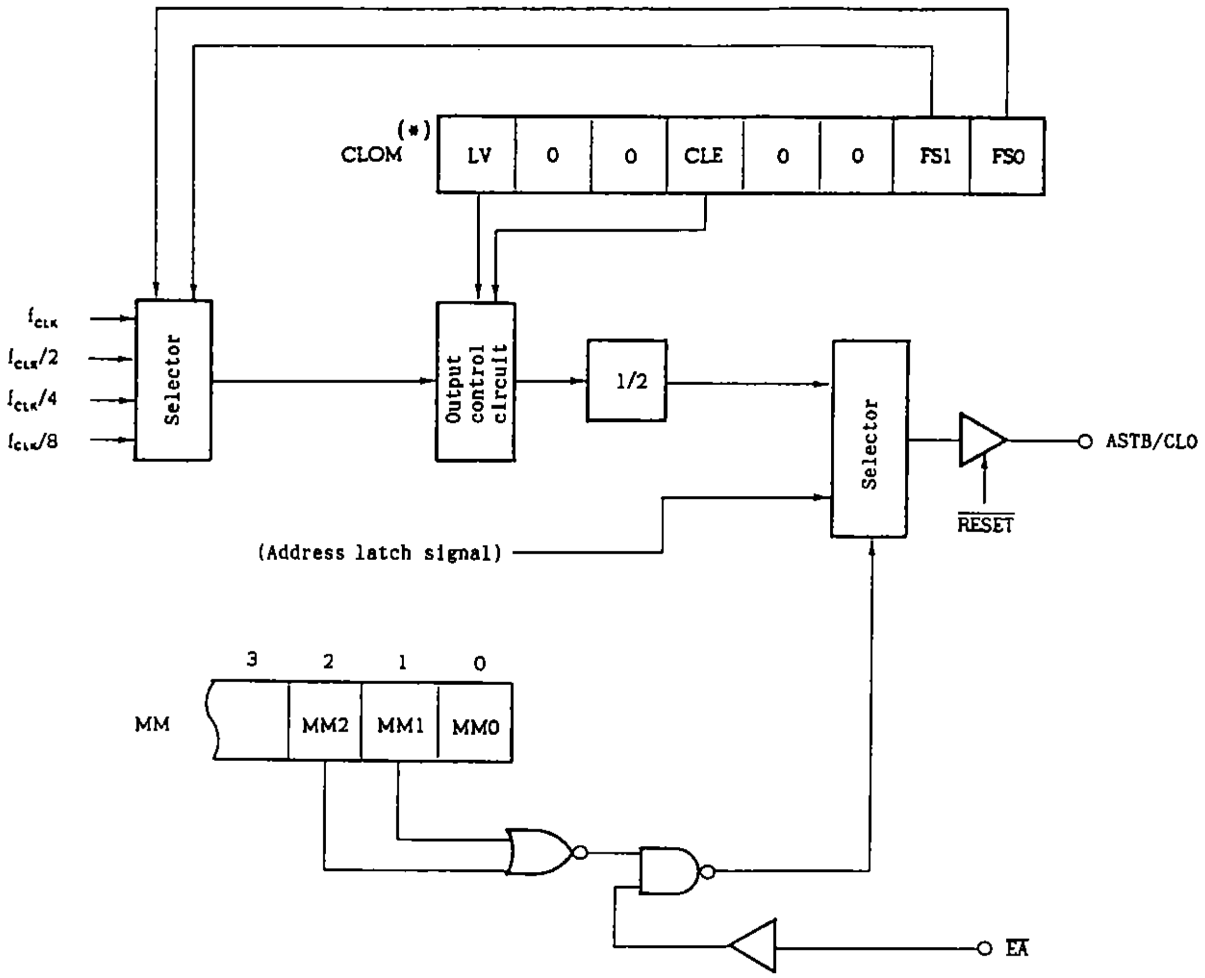
The frequency is set so that the dividing ratio is f_{CLK}/n where n is 2, 4, 8, or 16. ($f_{CLK} = f_{OSC}/2$: f_{OSC} denotes the oscillating frequency of the resonator.)

Fig. 10-1 Application Example for the Clock Output Function



When clock output is disabled, the CLO pin is used as a 1-bit-wide output port.

Fig. 10-2 Block Diagram of the Clock Output Circuit



* See figure 10-4 for the format.

Remark: f_{CLK}: Internal system clock

Figure 10-2 shows the configuration of the clock output circuit.

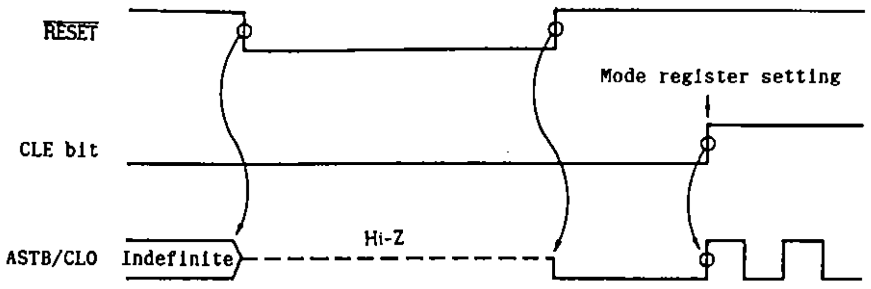
The clock output pin (CLO pin) also functions as the address latch strobe pin (ASTB pin) for the external expansion mode. The clock output function, therefore, is available only in the single-chip mode.

To use the clock output function, make the external access pin ($\overline{EA}$) low, and clear bits 1 and 2 (MM1 and MM2) in the memory mapping register (MM) to 0. The clock output function is not available in the external expansion mode. For details on the MM register, see Section 3.3. The CLO pin can be used as output port P60 when clock output is disabled. (The output value is specified by bit 7 of the CLOM register.)

Cautions 1. There is no operation mode to make the ASTB/CLO pin in the high impedance state. This pin becomes high impedance only during reset. After reset is released, a low signal appears on the pin. Figure 10-3 shows the state of the CLO pin when initialization is performed.

2. In the STOP mode, do not use the clock output function. Be sure to set CLE to 0 in this mode. (CLE: Bit 4 of the clock output mode register (CLOM))

Fig. 10-3 CLO Pin at Initialization



10.2 CLO Control Register

The clock output mode register (CLOM) controls clock output. Figure 10-4 shows the format of CLOM. CLOM is an 8-bit register that can be read from or written to with 8-bit manipulation instructions and bit manipulation instructions. To use CLO as a bit port, the high or low output must be specified with bit 7 (LV) in CLOM.
$\overline{\text{RESET}}$ input sets CLOM to 00H.

Fig. 10-4 Format of Clock Output Mode Register (CLOM)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
CLOM	LV	0	0	CLE	0	0	FS1	FS0	FF7FH	00H	R/W

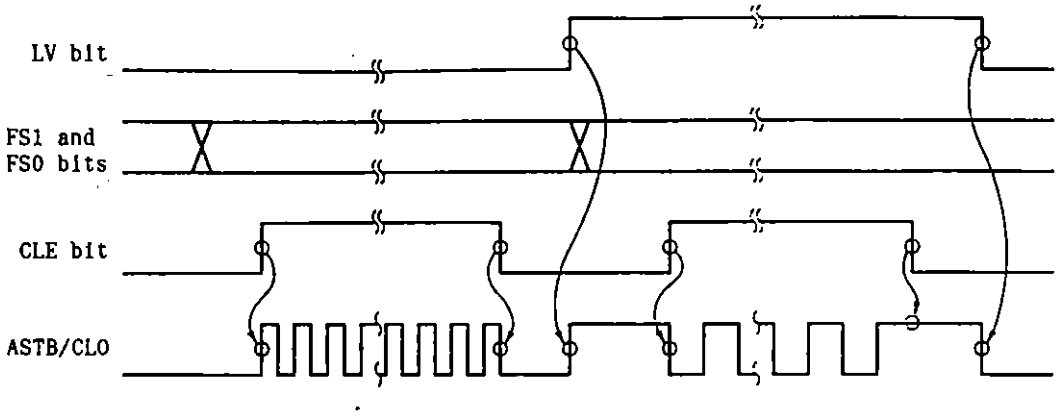
FS1	FS0	Specification of clock output frequency (when operated at 12 MHz)
0	0	$f_{CLK}/2$ (3.0 MHz)
0	1	$f_{CLK}/4$ (1.5 MHz)
1	0	$f_{CLK}/8$ (750 kHz)
1	1	$f_{CLK}/16$ (375 kHz)

Remark: f_{CLK}: Internal system clock

CLE	Clock output on CLO pin
0	Disabled. Output level is specified by LV bit.
1	Enabled. Output frequency is specified by FS1 and FS0 bits.

LV	CLO pin output level control (when CLE = 0)
0	Low level output
1	High level output

Fig. 10-5 Example of Setting the CLOM Register



Caution: The LV, FS1, and FS0 bits must be rewritten while the clock output is disabled (CLE = 0).

Remark: Manipulation on the LV or CLE bit does not cause spike noise on the CLO pin.

CHAPTER 11 INTERRUPT FUNCTION

The uPD78138 has two interrupt request processing modes. Table 11-1 lists the two modes. The program can optionally set these two modes. In the macro service mode, however, interrupts can be handled only for the interrupt request sources having the macro service processing mode. Table 11-2 lists these interrupt request sources.

Table 11-1 Interrupt Request Processing Modes

Interrupt request processing mode	Processed by	Contents of PC and PSW	Processing mode
Vectored interrupt	Software	With save and return operations	A branch to any service program is made and the interrupt is executed there.
Macro service	Hardware (Firmware)	Held	Processing set beforehand, such as memory I/O data transfer, is performed.

11.1 Interrupt Request Sources

The uPD78138 has 17 interrupt request sources (see Table 5-2). An interrupt vector table is assigned to each source.

Table 11-2 Interrupt Request Sources

Interrupt request type	Default priority	Interrupt source		Macro service processing mode	Vector table address
		Name	Interrupt trigger		
Non-maskable	-	NMI	Pin input edge detection	-	0002H
Maskable	0	INTP0	Pin input edge detection	Yes	0004H
	1	INTCPT3	EDVC output signal (CPT3 register capture)		0006H
	2	INTCPT2	CTI00 pin input edge detection (CPT2 register capture)		0008H
	3	INTCR12	CTI11 pin input edge detection, EDVC output signal (CPT12 register capture)		000AH
	4	INTCR00	TM0-CR00 match signal		000CH
	5	INTCLR1	CLR1 pin input edge detection		000EH
	6	INTCR10	TM1-CR10 match signal		0010H

(to be continued)

Table 11-2 Interrupt Request Sources (Cont'd)

Interrupt request type	Default priority	Interrupt source		Macro service processing mode	Vector table address
		Name	Interrupt trigger		
Maskable	7	INTCR01	TM0-CR01 match signal	Yes	0012H
	8	INTCR02	TM0-CR02 match signal		0014H
	9	INTCR11	TM1-CR11 match signal		0016H
	10	INTCPT1	Pin input edge detection, EC output signal (CPT1 register capture)		0018H
	11	INTTM	TM2-CR20 match signal		001AH
	12	INTCSI	Serial transfer end		001CH
	13	INTTB	Time base from FRC		001EH
	14	INTP1/INTAD	Pin input edge detection, A/D conversion end		0020H
	15	INTP2	Pin input edge detection		0022H

EDVC:	Event divider compare register
EC:	Event counter
TM0/1:	16-bit timers 0 and 1
CRxx:	Compare register (xx = 00, 01, 02, 10, 11, 20)
CPTxx:	Capture register (xx = 1, 2, 3, 12)
FRC:	18-bit free running counter

Remarks 1. An INTP1 interrupt is also used as an INTAD interrupt (A/D conversion end interrupt).

Remarks 2. The default priority indicates the priority used when two or more interrupts occur simultaneously.

11.1.1 Nonmaskable interrupt request

Nonmaskable interrupt requests are accepted unconditionally even in the interrupt disable (DI) state. Such requests are not subject to interrupt priority control, that is, they have the highest priority of all interrupts. Multiprocessing by a nonmaskable interrupt request, however, can be accepted only when bit 0 (NMIS) of the interrupt status register (IST) is reset to 0.

Nonmaskable interrupt requests are made by input to pin NMI. When a valid edge specified in bit 0 (ESNMI) of the external interrupt mode register 0 (INTM0) is detected during input of a request to pin NMI, an interrupt request is issued.

11.1.2 Maskable interrupt request

Maskable interrupt requests are subject to mask control according to the setting of the interrupt mask register (MK0).

When two or more maskable interrupt requests are issued at a time, the maskable interrupt request having the highest default priority is processed first. See Table 11-2 for the default priorities of maskable interrupt requests. Setting the priority specification flag register (PR0) can divide interrupt priorities into two groups, a higher priority group and lower priority group. However, macro services can be accepted irrespective of priority control.

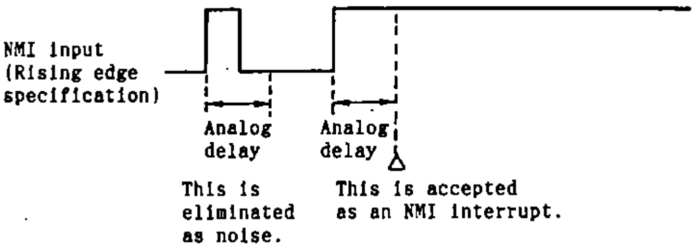
11.2 External Interrupt Request Functions

An external interrupt request is issued when a valid edge specified by the external interrupt mode register (INTM0) or external capture input mode register (INTM1) is detected during input to pin NMI, INTP0 to INTP2, CTI00, or CTI11.

The NMI input pin has an internal noise eliminator with an analog delay feature. Input signals having insufficient duration are eliminated as noise. (See Figure 11-1.)

Pins INTP0 to INTP2, CTI00, and CTI11 have a Schmitt trigger with hysteresis characteristics.

Fig. 11-1 Noise Elimination at an External Interrupt Request Pin



11.2.1 External interrupt control registers

For each external interrupt pin, a valid edge can be specified in the external interrupt mode register (INTM0) or external capture input mode register (INTM1).

Table 11-3 Valid Edges and Control Registers of External Interrupt Pins

External interrupt pin	Valid edge	Control register
NMI	. Rising edge . Falling edge	INTM0
INTP0	. Rising edge . Falling edge . Rising and falling edges	
INTP1	. Rising edge . Falling edge	
INTP2		
CTI00	. Rising edge only	-
CTI11	. Rising edge . Falling edge	INTM1

(1) External interrupt mode register (INTM0)

INTM0 is an 8-bit register which specifies a valid edge on pins NMI and INTP0 to INTP2.

Eight-bit manipulation instructions and bit manipulation instructions are used to read data from INTM0 and write data into INTM0.

Fig. 11-2 Format of the External Interrupt Mode Register 0 (INTM0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
INTM0	ES2	1	ES1	1	ES01	ES00	0	ES NMI	FFF4H	50H	R/W

ES NMI	NMI pin input detection edge specification
0	Falling edge
1	Rising edge

ES01	ES00	INTP0 pin input detection edge specification
0	0	Falling edge
0	1	Rising edge
1	0	Use prohibited
1	1	Rising and falling edges

ES1	INTP1 pin input detection edge specification
0	Rising edge
1	Rising and falling edges

ES2	INTP2 pin input detection edge specification
0	Rising edge
1	Rising and falling edges

(2) External capture input mode register (INTM1)

INTM1 is an 8-bit register which specifies a valid edge on pins CTI10, CTI11, and CLR1. Only pin CTI11 has the external interrupt function.

Eight-bit manipulation instructions and bit manipulation instructions are used to read data from INTM1 and write data into INTM1.

Fig. 11-3 Format of the External Capture Input Mode Register (INTM1)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
INTM1	0	ES CLR1	0	ES11	—	—	ES10	1	FFF5H	0000xx01	R/W

ES10	CTI10 pin input detection edge specification
0	Rising edge
1	Rising and falling edges

ES11	CTI11 pin input detection edge specification
0	Falling edge
1	Rising edge

ES CLR1	CLR1 pin input detection edge specification
0	Falling edge
1	Rising edge

11.3 Interrupt Processing Control Registers

The following four registers control interrupt processing.

- Interrupt request flag register (IF0)
- Interrupt mask register (MK0)
- Interrupt service mode register (ISM0)
- Priority specification register (PR0)

The above four registers are all 16-bit read/write registers. These registers can be manipulated in 8- or 16-bit units. A bit manipulation instruction is used to set or reset the bit of these registers. Figures 11-4 through 11-7 show the formats of each of the four registers.

Table 11-4 lists the names of the interrupt request flag, interrupt mask flag, interrupt service mode flag, and priority specification flag corresponding to each interrupt request source.

Table 11-4 Flags Corresponding to Each Interrupt Request Source

Interrupt request source	Interrupt request flag	Interrupt mask flag	Interrupt service mode flag	Priority specification flag
INTP0	PIF0	PMK0	PISM0	PPR0
INTCPT3	CPIF3	CPMK3	CPISM3	CPPR3
INTCPT2	CPIF2	CPMK2	CPISM2	CPPR2
INTCR12	CRIF12	CRMK12	CRISM12	CRPR12
INTCR00	CRIF00	CRMK00	CRISM00	CRPR00
INTCLR1	CLIF1	CLMK1	CLISM1	CLPR1
INTCR10	CRIF10	CRMK10	CRISM10	CRPR10

(to be continued)

Table 11-4 Flags Corresponding to Each Interrupt Request Source (Cont'd)

Interrupt request source	Interrupt request flag	Interrupt mask flag	Interrupt service mode flag	Priority specification flag
INTCR01	CRIF01	CRMK01	CRISM01	CRPR01
INTCR02	CRIF02	CRMK02	CRISM02	CRPR02
INTCR11	CRIF11	CRMK11	CRISM11	CRPR11
INTCPT1	CPIF1	CPMK1	CPISM1	CPPR1
INTTM	TMIF	TMMK	TMISM	TMPR
INTCSI	CSIIF	CSIMK	CSIISM	CSIPR
INTTB	TBIF	TBMK	TBISM	TBPR
INTP1/ INTAD	PIF1	PMK1	PISM1	PPR1
INTP2	PIF2	PMK2	PISM2	PPR2

(1) Interrupt request flag register (IF0)

The interrupt request flag register is a 16-bit register consisting of the interrupt request flags.

Each interrupt request flag is set to 1 when corresponding interrupt request is issued. In this way, vectored interrupt processing is accepted. Performing macro service processing clears IF0 to 0.

Input of a $\overline{\text{RESET}}$ signal resets IF0 to 0000H.

(2) Interrupt mask register (MK0)

The interrupt mask register is a 16-bit register consisting of the interrupt mask flags. Each interrupt mask flag controls enabling/disabling of a corresponding interrupt request.

Input of a $\overline{\text{RESET}}$ signal resets MK0 to FFFFH, disabling all maskable interrupts.

(3) Interrupt service mode register (ISM0)

The interrupt service mode register is a 16-bit register consisting of the interrupt service mode flags. When an interrupt service mode flag is set to 0, a corresponding interrupt request is processed by a vectored interrupt. When an interrupt service mode flag is set to 1, a corresponding interrupt request is processed by a macro service. After a macro service request is processed a specified number of times, the flag is cleared to 0.

Input of a $\overline{\text{RESET}}$ signal resets ISM0 to 0000H, specifying processing by a vectored interrupt.

(4) Priority specification flag register (PR0)

The priority specification flag register is a 16-bit register consisting of the priority specification flags for accepting an interrupt. PR0 is used to control multiple interrupt processing.

Either the higher priority group and lower priority group can be set. When the priority specification flag is set to 0, the corresponding interrupt request is assigned to the higher priority group. When the priority specification flag is set to 1, the corresponding interrupt request is assigned to the lower priority group.

When a vectored interrupt having lower priority is being processed in the EI state, multiple vectored interrupts having lower or higher priority can be accepted. When a vectored interrupt having higher priority is being processed in the EI state, only multiple vectored interrupts having higher priority can be accepted. However, macro services are accepted irrespective of the priority specification.

Input of a $\overline{\text{RESET}}$ signal resets PR0 to FFFFH and all interrupt requests are assigned to the lower priority group.

Fig. 11-4 Format of the Interrupt Request Flag Register (IF0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
IF0L	CRIF01	CRIF10	CLIF1	CRIF00	CRIF12	CPIF2	CPIF3	PIF0	FFE0H	00H	R/W
IF0H	PIF2	PIF1	TBIF	CSIIF	TMIF	CPIF1	CRIF11	CRIF02	FFE1H	00H	R/W

Interrupt request flag

0	No interrupt request is issued.
1	An interrupt request is issued.

Fig. 11-5 Format of the Interrupt Mask Register (MK0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
MK0L	CRMK01	CRMK10	CLMK1	CRMK00	CRMK12	CPMK2	CPMK3	PMK0	FFE4H	FFH	R/W
MK0H	PMK2	PMK1	TBMK	CSIMK	TMMK	CPMK1	CRMK11	CRMK02	FFE5H	FFH	R/W

Interrupt request mask flag

0	Interrupt processing is allowed.
1	Interrupt processing is held.

Fig. 11-6 Format of the Interrupt Service Mode Register (ISM0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
ISM0L	CRISM01	CRISM10	CLISM1	CRISM00	CRISM12	CPISM2	CPISM3	PISM0	FFECH	00H	R/W
ISM0H	PISM2	PISM1	TBISM	CSIISM	TMISM	CPISM1	CPISM11	CRISM02	FFEDH	00H	R/W

Interrupt service mode flag

0	Processed by a vectored interrupt
1	Processed by a macro service

Fig. 11-7 Format of the Priority Specification Flag Register (PR0)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
PR0L	CRPR01	CRPR10	CLPR1	CRPR00	CRPR12	CPPR2	CPPR3	PPR0	FFE8H	FFH	R/W
PR0H	PPR2	PPR1	TBPR	CSIPR	TMPR	CPPR11	CRPR11	CRPR02	FFE9H	FFH	R/W

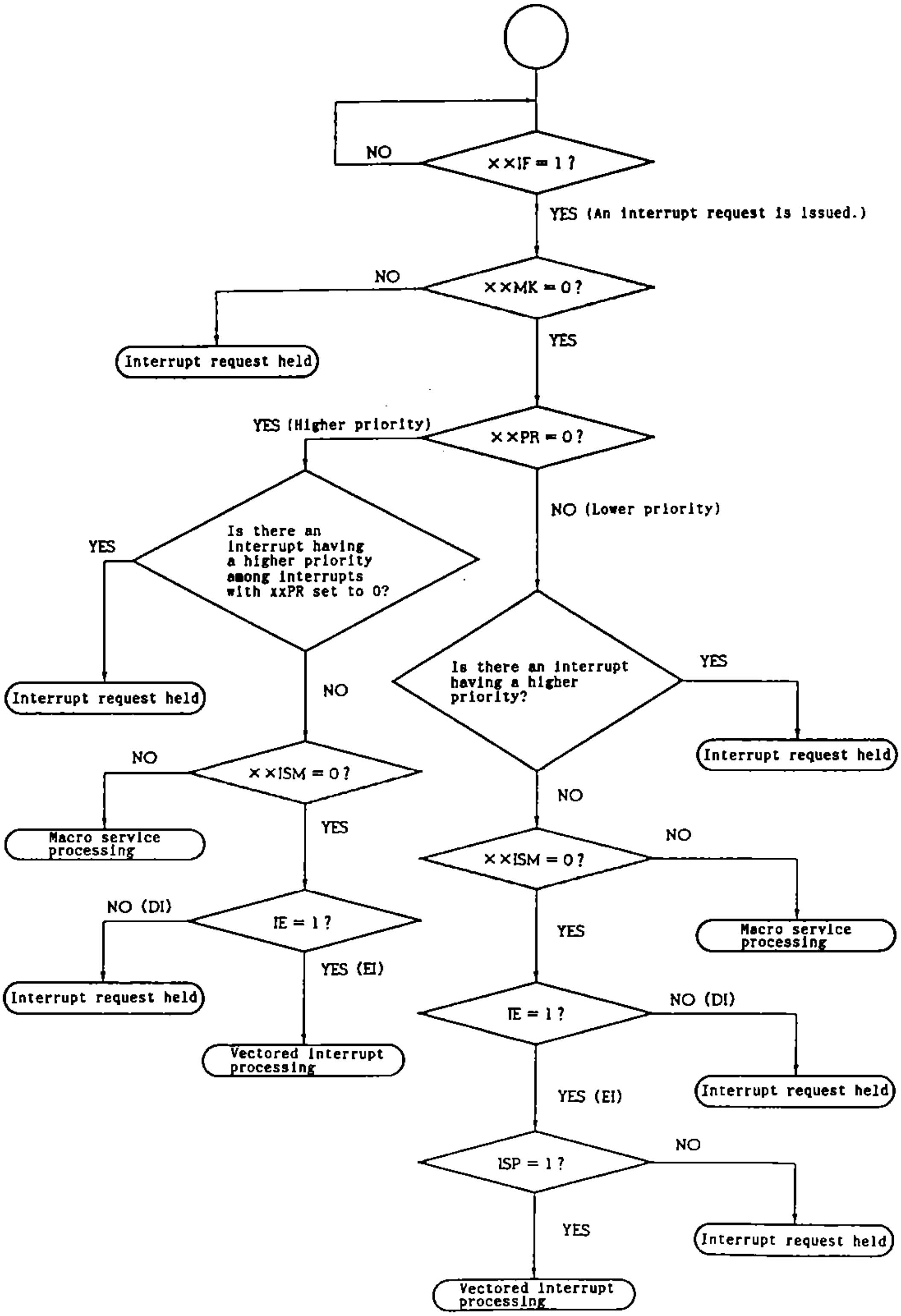
Priority specification flag

0	Higher priority level
1	Lower priority level

11.4 Interrupt Processing

Figure 11-8 shows the interrupt processing algorithm.

Fig. 11-8 Interrupt Processing Algorithm



11.4.1 Multiple interrupt processing

The uPD78138 can perform multiple interrupt processing, which means another interrupt can be accepted during processing of an interrupt. Multiple interrupts are controlled according to the default priorities or programmable priorities.

Default priority control processes multiple interrupts which occur simultaneously according to the priorities which have been assigned to the interrupts (default priorities). See Table 11-2 for the default priorities. Programmable priority control divides interrupt requests into two groups, higher priority group and lower priority group, according to the setting of the corresponding bit of the priority specification flag register (PR0). Table 11-5 lists the interrupt requests which can be accepted while another interrupt is being processed.

Table 11-5 Multiple Interrupt Processing

Interrupting request (source)	Interrupted request (destination)	
Interrupt having lower programmable priority	DI	. Nonmaskable interrupts . Maskable interrupts by macro service processing
	EI(*1)	. Nonmaskable interrupts . All maskable interrupts
Interrupt having higher programmable priority	DI	. Nonmaskable interrupts . Maskable interrupts by macro service processing
	EI(*1)	. Nonmaskable interrupts . Maskable interrupts having a higher programmable priority
Nonmaskable interrupts	DI	. Nonmaskable interrupts(*2) . Maskable interrupts by macro service processing
	EI(*1)	. Nonmaskable interrupts(*2) . All maskable interrupts

*1 The DI state is automatically enabled immediately after an interrupt request has been accepted. To set the EI state, execute an EI instruction.

*2 Bit 0 (NMIS) of the interrupt status register (IST) is set to 1 during acceptance of a nonmaskable interrupt. If the NMIS bit is set to 1, another nonmaskable interrupt does not occur. To enable multiple nonmaskable interrupt processing, the NMIS flag must have been reset to 0.

11.4.2 When interrupt requests and macro services are temporarily held

□

When one of the following instructions is issued, reception of all maskable interrupts and processing of macro services are temporarily held until execution of the next instruction is complete.

. EI
. DI
. RETI
. POP PSW
. MOV PSW, A
. MOV PSW, #byte
. Bit manipulation instructions for the PSW (Excluding BT PSW.bit, $addr16, BF PSW.bit, $addr16, SET1 CY, CLR1 CY, and NOT1 CY)
. Manipulation instructions for each of registers MK0L, MK0H, and MK1L
. Manipulation instructions for each of registers IF0L, IF0H, and IF1L
. Manipulation instructions for each of registers PR0L, PR0H, and PR1L
. Manipulation instructions for each of registers ISM0L, ISM0H, and ISM1L

Cautions 1. When a register related to interrupts such as IFxx is polled using an instruction such as BF, do not specify the instruction as its branch destination. For such a program, all interrupts and macro services are held until the condition under which no branch is caused is satisfied.

Examples 1. Incorrect specification

```
                 ⋮                ; All interrupts and
        LOOP:  BF IF0H.3, $LOOP; macro services are
               xxx                ; held until bit 3 at
                 ⋮                ; address IF0H is set
                                  ; to 1.  Interrupts
                                  ; and macro services
                                  ; are processed after
                                  ; execution of the
                                  ; instruction
                                  ; following the BF
                                  ; instruction.
```

2. Example 1 of correct specification

```
                 ⋮                ; Interrupts and
        LOOP:  NOP                ; macro services are
               BF IF0H.3, $LOOP; not held long
                 ⋮                ; because they are
                                  ; processed after
                                  ; execution of the
                                  ; NOP instruction.
```

Examples 3. Example 2 of correct specification

```
                 :             ; Interrupts and
LOOP:   BT IF0H.3, $NEXT; macro services are
        BR $LOOP          ; not held long
NEXT:                     ; because they are
                 :             ; processed after
                          ; execution of the BR
                          ; instruction.
```

Remark: It is useful to use the BTCLR instruction instead of the BT instruction because the BTCLR instruction automatically clears the flag.

Cautions 2. When the above instructions are required to be executed consecutively, interrupts and macro services are held for a long time. To prevent this, insert NOP and other instructions so that interrupts and macro services can be processed.

11.5 Macro Service Functions

The interrupt processing function of the uPD78138 consists of a built-in vectored interrupt function and macro service function.

11.5.1 Overview of macro services

The macro service function executes a specific service set by the firmware when an interrupt request is issued. The mode register sets this service and the hardware processes this service. Since this function is not performed via the CPU, the statuses (SP and PSW) of the CPU need not be saved and returned each time an interrupt occurs. Thus, CPU service time is improved.

There are 16 maskable interrupt requests which can process a macro service. (See Table 11-2.)

The following macro services are provided.

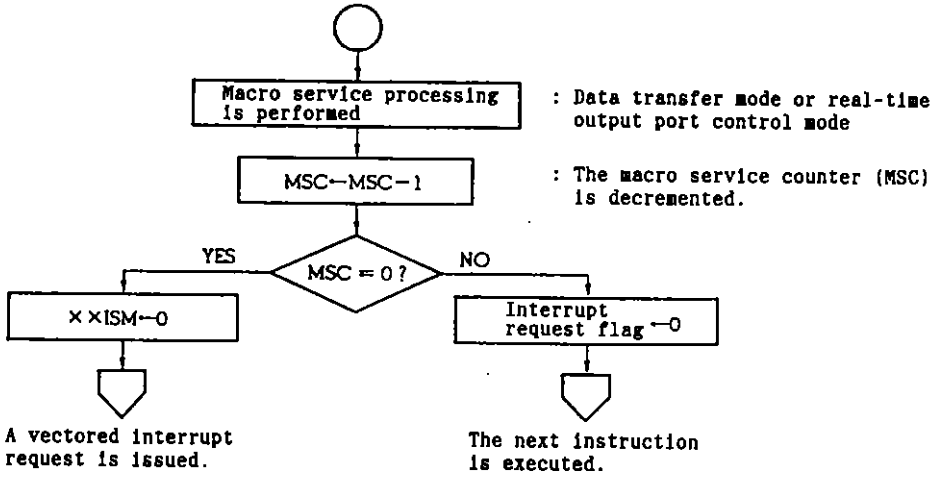
- Data transfer mode
- Real-time output port control mode
- Counter mode
- Data pattern identification mode

The processing of a macro service is specified according to the algorithm shown in Figure 11-8 and the macro service is processed according to the sequence shown in Figure 11-9.

Macro service processing can be accepted irrespective of the interrupt state (DI or EI). To disable macro service processing, set the mask flag of the interrupt mask register (MK0) to 1. Note that macro service processing can be accepted during execution of a vectored interrupt processing program.

Fig. 11-9 Sequence of Macro Service Processing

When an interrupt request which can specify macro service processing occurs


Macro service processing is performed
: Data transfer mode or real-time output port control mode
MSC←MSC−1
: The macro service counter (MSC) is decremented.
YES
MSC = 0 ?
NO
××ISM←0
Interrupt request flag ←0
A vectored interrupt request is issued.
The next instruction is executed.

11.5.2 Macro service control register

(1) Macro service control word

The uPD78138 macro service function is controlled with the macro service mode registers and the macro service channel pointers.

A macro service mode register sets the macro service processing mode and a macro service channel pointer sets the address of a macro service channel.

Each combination of a macro service mode register and a macro service channel pointer at addresses FEC0H to FEDFH of the internal RAM is mapped as a macro service control word for each macro service. Figure 11-10 shows the macro service control words.

The address of a macro service control word is determined corresponding to each interrupt which can process a macro service.

The macro service mode registers and the macro service channel pointers must be set before a macro service is processed.

Remark: A macro service channel is a data memory location accessed by macro service processing.

Fig. 11-10 Macro Service Control Words

	General register	
FEDFH	Channel pointer	INTCPT3
FEDEH	Mode register	
FEDDH	Channel pointer	INTCPT2
FEDCH	Mode register	
FEDBH	Channel pointer	INTCR12
FEDAH	Mode register	
FED9H	Channel pointer	INTCR00
FED8H	Mode register	
FED7H	Channel pointer	INTCLR1
FED6H	Mode register	
FED5H	Channel pointer	INTCPT1
FED4H	Mode register	
FED3H	Channel pointer	INTCSI
FED2H	Mode register	
FED1H	Channel pointer	INTCR01
FED0H	Mode register	
FECFH	Channel pointer	INTCR02
FECEH	Mode register	
FECDH	Channel pointer	INTP1/INTAD
FECCH	Mode register	
FECBH	Channel pointer	INTTB
FECAH	Mode register	
FEC9H	Channel pointer	INTP0
FEC8H	Mode register	
FEC7H	Channel pointer	INTCR10
FEC6H	Mode register	
FEC5H	Channel pointer	INTCR11
FEC4H	Mode register	
FEC3H	Channel pointer	INTTM
FEC2H	Mode register	
FEC1H	Channel pointer	INTP2
FEC0H	Mode register	

(2) Macro service mode register

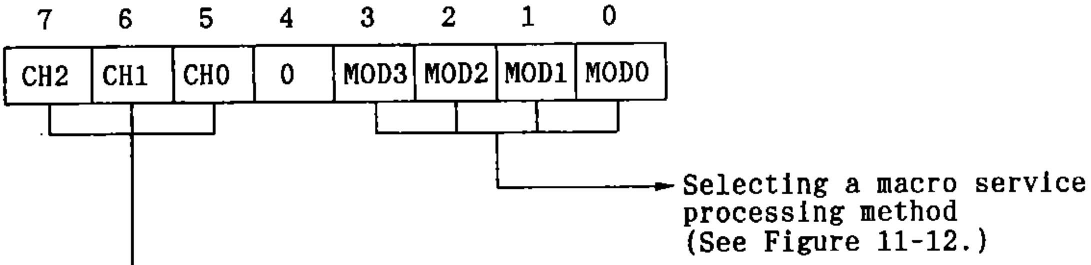
A macro service mode register is an 8-bit register which specifies a macro service operation.

A macro service mode register, which is a part of a macro service control word, contains set values on the internal RAM. (See Figure 11-10.)

The high-order three bits (bits 5 to 7) of a macro service mode register specify the operation mode and the low-order four bits (bits 0 to 3) specify the processing method in each operation mode.

Figures 11-11 and 11-12 show the format of a macro service mode register.

Fig. 11-11 Format of the Macro Service Mode Register (High-order Three Bits)


7
6
5
4
3
2
1
0
CH2
CH1
CH0
0
MOD3
MOD2
MOD1
MOD0
Selecting a macro service processing method (See Figure 11-12.)

CH2	CH1	CH0	Selecting a macro service operation mode
0	1	0	Data transfer mode
1	1	0	Real-time output port control mode (The timer macro service pointer is held after transfer.)
1	1	1	Real-time output port control mode (The timer macro service pointer (low order) is incremented after transfer.)
1	0	0	Counter mode
0	0	0	Data pattern identification mode

Fig. 11-12 Format of a Macro Service Mode Register

7	6	5	4	3	2	1	0
CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0

			CH0	0	0 ... The timer macro service pointer is held after transfer. 1 ... The timer macro service pinter (low order) is incremented after transfer.				0	0	
			CH1	1	1				0	0	
			CH2	0	1				0	1	
MOD3	MOD2	MOD1	MOD0	Data transfer mode	Real-time output port control mode				Counter mode	Data pattern identification mode	
0	0	0	0	Data transfer from memory to SFR							
0	0	0	1	Data transfer from SFR to memory							
1	0	0	0		Macro service for real-time output port control	Without ring control	Data transfer only	Four low-order bits P00-P03		Data pattern identification mode	Shift only
1	0	0	1					Four high-order bits P04-P07	Counter mode		With comparison
1	0	1	0				With automatic addition	Four low-order bits P00-P03			
1	0	1	1					Four high-order bits P04-P07			
1	1	0	0			With ring control	Data transfer only	Four low-order bits P00-P03			
1	1	0	1					Four high-order bits P04-P07			
1	1	1	0				With automatic addition	Four low-order bits P00-P03			
1	1	1	1					Four high-order bits P04-P07			

11.5.3 Macro service modes and interrupt requests

The macro service mode depends upon the type of interrupt request source. Table 11-6 lists the correspondence between the macro service modes and interrupt request sources.

Table 11-6 Macro Service Modes and Interrupt Request Sources

Macro service mode	Interrupt request source which can process the corresponding macro service
Data transfer mode	INTCSI, INTAD
Real-time output port control mode	INTCR01, INTCR02
Counter mode	All maskable interrupt requests
Data pattern identification mode	INTCR12

11.5.4 Data transfer mode

(1) Overview of function

In the data transfer mode, data is transferred between the internal memory location and the special function register (SFR).

Table 11-7 lists the interrupt request sources which can process a macro service in the data transfer mode and the source/destination SFR.

Table 11-7 Interrupt Requests Sources in the Data Transfer Mode and SFR

Interrupt request sources in the data transfer mode	Source/destination SFR
INTAD	ADCR register
INTCSI	SIO register

In the data transfer mode, data can be transferred from memory to the SFR or vice versa.

The four low-order bits (bits 0 to 3) of a macro service mode register set the transfer direction.

Figure 11-13 shows the setting of a macro service mode register in the data transfer mode.

Fig. 11-13 Setting of a Macro Service Mode Register in the Data Transfer Mode

7	6	5	4	3	2	1	0	
CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0	Macro service operation
0	1	0	0	0	0	0	0	Data transfer from memory to SFR
				0	0	0	1	Data transfer from SFR to memory

Figure 11-14 shows the addressing in the data transfer mode.

A combination of a macro service channel pointer and the macro service counter specifies the buffer address of the source or destination internal RAM (FE00H to FEFFH).

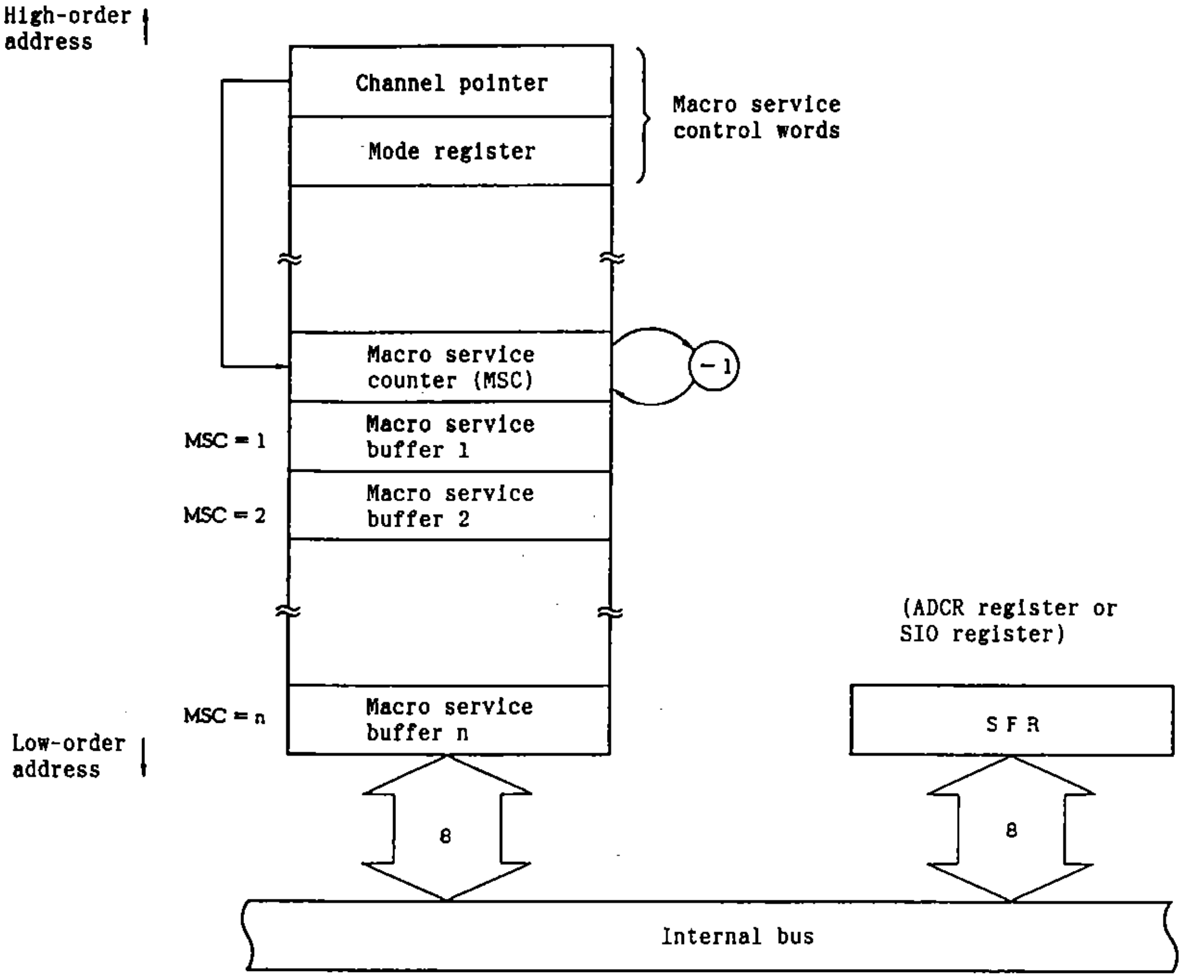
The value set on the macro service counter indicates the number of macro services. Each time a byte of data is transferred, the value on the macro service counter is decremented by one. When the value on the macro service counter is 0, a vectored interrupt occurs.

In the data transfer mode, only the ADCR or SIO register is used as a source/destination SFR.

Figure 11-15 shows the processing sequence in the data transfer mode.

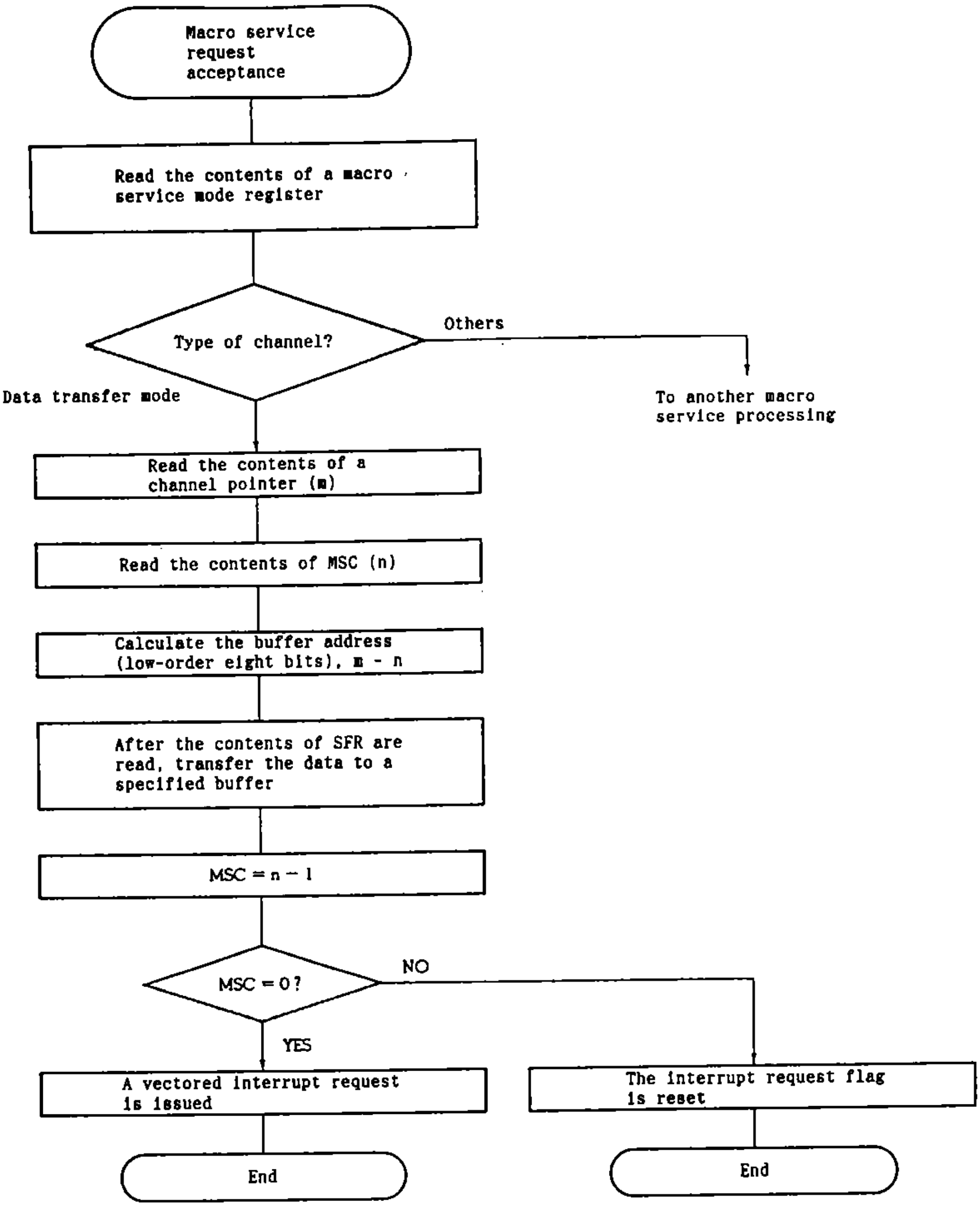
Caution: The macro service counter (MSC) is decremented for each macro service. When MSC is 0, a vectored interrupt occurs. To process a macro service again after this, set MSC again.

Fig. 11-14 Addressing in the Data Transfer Mode



Address of a macro service buffer (eight low-order bits) = Contents of a channel pointer minus contents of the macro service counter

Fig. 11-15 Processing Sequence in the Data Transfer Mode


Macro service request acceptance
Read the contents of a macro service mode register
Type of channel?
Others
To another macro service processing
Data transfer mode
Read the contents of a channel pointer (m)
Read the contents of MSC (n)
Calculate the buffer address (low-order eight bits), m - n
After the contents of SFR are read, transfer the data to a specified buffer
MSC = n - 1
MSC = 0 ?
NO
YES
A vectored interrupt request is issued
The interrupt request flag is reset
End
End

(2) Example

Figure 11-16 shows an example of transferring data received from the synchronous serial interface to a buffer area in the internal RAM.

The macro service channel pointer of INTCSI is mapped at FED3H. The value set to this channel pointer indicates the low-order 8-bit address of the macro service counter (MSC). The high-order 8-bit address is fixed to FEH.

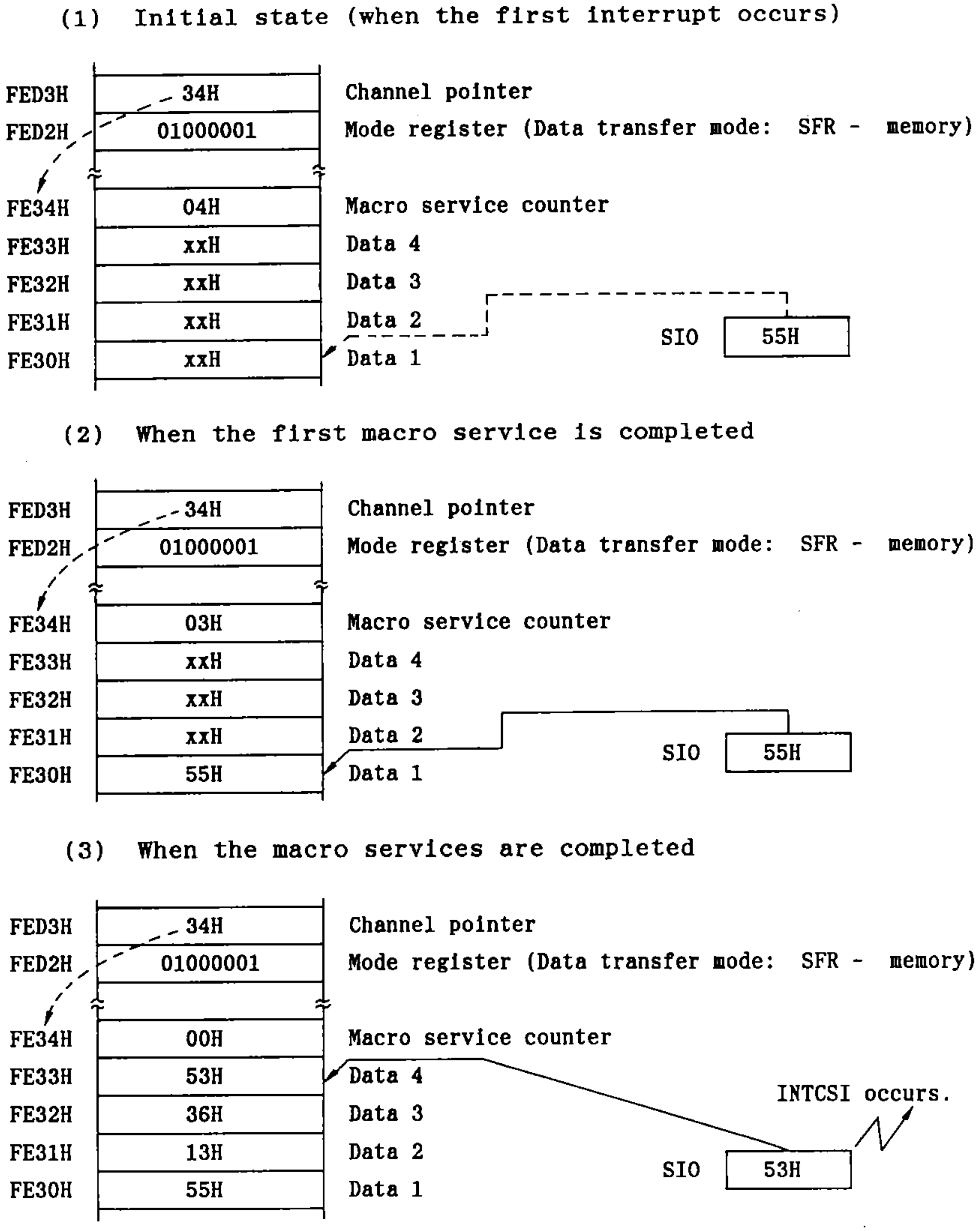
The memory location indicated by the address set on MSC is allocated as a buffer area in the low-order bits of the MSC address.

When a macro service request is issued from the serial interface, the contents of the shift register (SIO) are stored in the memory location indicated by the MSC address minus the MSC contents. After that, the data set on MSC is decremented.

The contents of the shift register are sequentially stored in the buffer register from the lowest-order address. When data in the shift register is stored the number of times set on MSC, that is, when MSC is 0, a vectored interrupt request is issued.

After a vectored interrupt occurs, set again MSC, which remains 0, with the interrupt service program.

Fig. 11-16 Example of Data Transfer



(1) Initial state (when the first interrupt occurs)

FED3H	34H	Channel pointer
FED2H	01000001	Mode register (Data transfer mode: SFR - memory)
FE34H	04H	Macro service counter
FE33H	xxH	Data 4
FE32H	xxH	Data 3
FE31H	xxH	Data 2
FE30H	xxH	Data 1

(2) When the first macro service is completed

FED3H	34H	Channel pointer
FED2H	01000001	Mode register (Data transfer mode: SFR - memory)
FE34H	03H	Macro service counter
FE33H	xxH	Data 4
FE32H	xxH	Data 3
FE31H	xxH	Data 2
FE30H	55H	Data 1

(3) When the macro services are completed

FED3H	34H	Channel pointer
FED2H	01000001	Mode register (Data transfer mode: SFR - memory)
FE34H	00H	Macro service counter
FE33H	53H	Data 4
FE32H	36H	Data 3
FE31H	13H	Data 2
FE30H	55H	Data 1

11.5.5 Real-time output port control mode

(1) Overview of function

In the real-time output port control mode, data output to the real-time output port and the output timing can be controlled easily.

An interrupt request (INTCR01 or INTCR02) issued from 16-bit timer 0 (TM0) controls the real-time output port.

There are two macro service pointers in this mode: the timer macro service pointer and the data macro service pointer. The timer macro service pointer indicates the output timing data area in the 64K-byte memory space and the data macro service pointer indicates the output data area. Table 11-8 lists these macro service pointers.

Table 11-8 Functions of the Macro Service Pointers

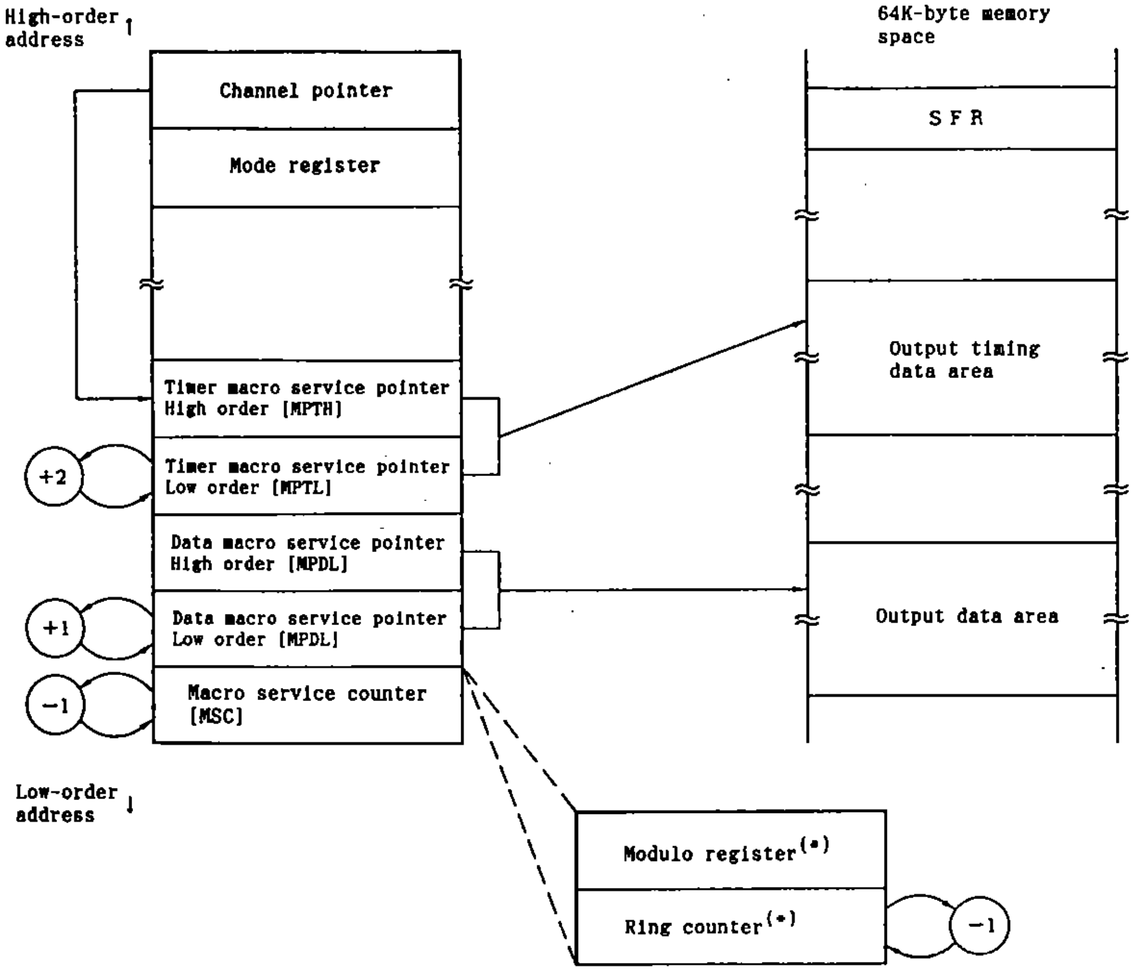
Macro service pointer	Area pointed to	Destination SFR
For timer (MPTH, MPTL)	Output timing data area	CR01, CR02
For data (MPDH, MPDL)	Output data area	P0L, P0H

Figure 11-17 shows the addressing in the real-time output port control mode.

The output data area stores data to be output to the buffer register (P0L and P0H) of the real-time output port.

The output timing data area stores the value for the data output cycle. The data is sequentially transferred to the compare register (CR01 or CR02) of 16-bit timer 0.

Fig. 11-17 Addressing in the Real-time Output Port Control Mode


High-order address
64K-byte memory space
Channel pointer
Mode register
SFR
Output timing data area
Timer macro service pointer High order [MPTH]
Timer macro service pointer Low order [MPTL]
+2
Data macro service pointer High order [MPDL]
Data macro service pointer Low order [MPDL]
+1
Output data area
Macro service counter [MSC]
−1
Low-order address
Modulo register(*)
Ring counter(*)
−1

* The modulo register and ring counter are built in to provide ring control.
  In the mode not subject to ring control, the modulo register and ring counter are not built in.
  The modulo register is used for the ring counter, that is, when the ring counter is 0, the contents of the modulo register are reloaded.

Remark: See (b) and (c) of (2) in this section for details of ring control.

The data macro service pointer (MPDL and MPDH) is used as a pointer to the output data area. The timer macro service pointer (MPTL and MPTH) is used as a pointer to the output timing data area.

In the real-time output port control mode, the following operations can be selected by a macro service mode register.

① Ring control

Ring control is valid when fixed data patterns are output to the real-time output port repeatedly.

② Output timing data transfer or automatic addition

The timing at which data is output to the real-time output port is controlled with the compare register (CR01 or CR02) of timer 0. The data in the compare register is transferred from memory. Alternatively, the contents of memory are added to the contents of the compare register.

③ Hold or increment by the timer macro service pointer (MPT)

The source address (pointed to by MPT) for the timing at which data is output to the real-time output port can be held or incremented according to the specification after a macro service is processed.

In an application where data is output to the real-time output port at fixed intervals, MPT hold is selected.

④ High order (P0H) and low order (P0L) of the buffer register

The destination of the data output to the real-time output port is selected.

The data macro service pointer (MPD) is incremented by one each time a macro service is processed. The macro service mode register specifies whether the timer macro service pointer (MPT) is incremented or the data for that pointer is held. When incrementing is selected, MPT is incremented by two.

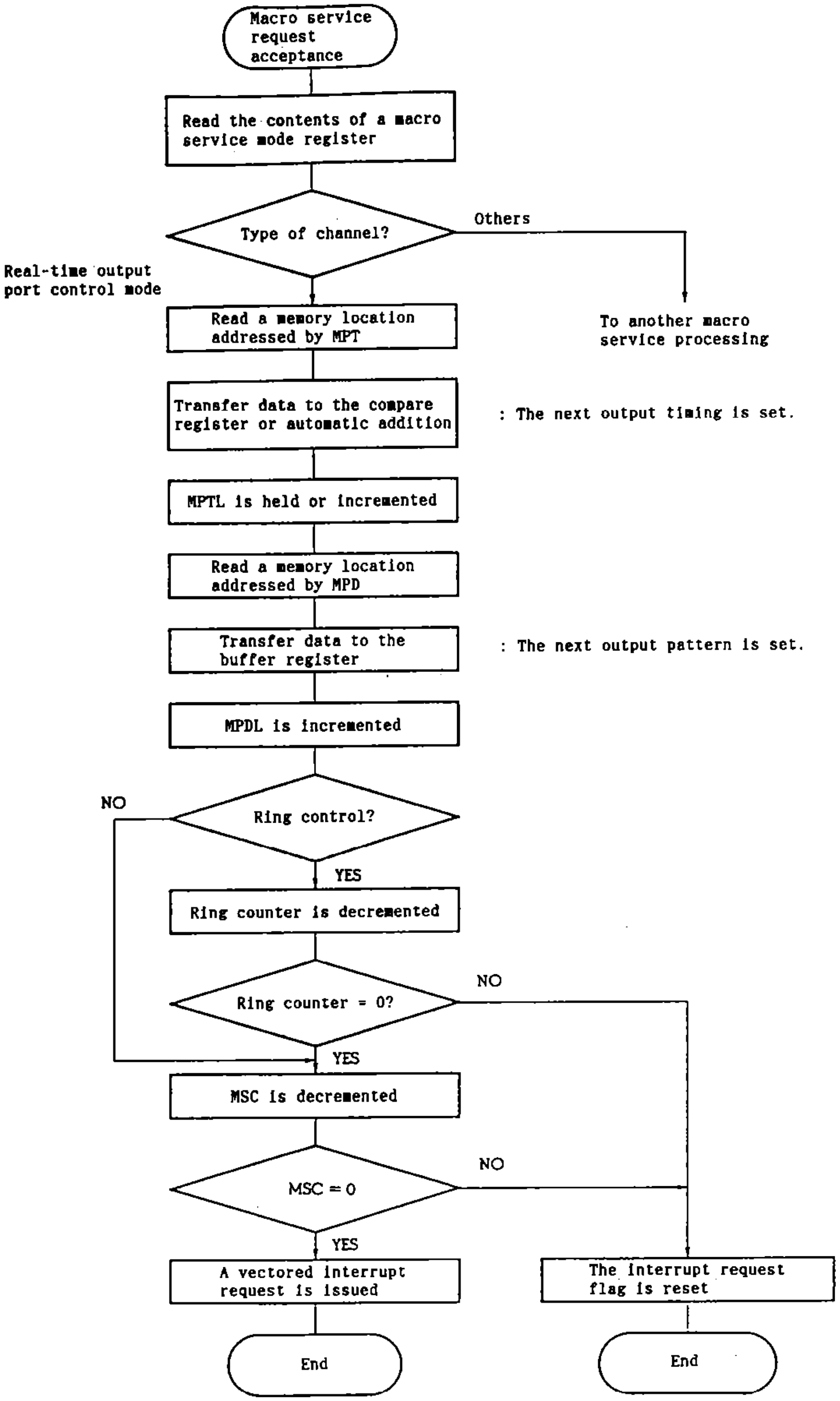
After a macro service is processed, the macro service counter (MSC) is decremented by one. When MSC is 0, a vectored interrupt occurs.

Figure 11-18 shows the processing sequence in the real-time output port control mode.

Cautions 1. When MPD and MPT are incremented, only the eight low-order bits are incremented in macro service processing. (The eight high-order bits remain unchanged.) That is, even if 1FFFH is set, the result of incrementing is not 2000H, but 1F00H. To increment the eight high-order bits, set them again with a vectored interrupt.

Cautions 2. Since output data in macro service processing is single-byte data, the data macro service pointer is incremented by one. However, since output timing data is two-byte data, the timer macro service pointer is incremented by two.

Fig. 11-18 Processing Sequence in the Real-time Output Port Control Mode


Macro service request acceptance
Read the contents of a macro service mode register
Type of channel?
Others
To another macro service processing
Real-time output port control mode
Read a memory location addressed by MPT
Transfer data to the compare register or automatic addition
: The next output timing is set.
MPTL is held or incremented
Read a memory location addressed by MPD
Transfer data to the buffer register
: The next output pattern is set.
MPDL is incremented
Ring control?
NO
YES
Ring counter is decremented
Ring counter = 0?
NO
YES
MSC is decremented
MSC = 0
NO
YES
A vectored interrupt request is issued
The interrupt request flag is reset
End
End

(2) Example

(a) Example in the basic operation mode

In the real-time output port control mode, normally updated data is transferred from the two data areas which have been set in the 64K-byte space to the buffer register (P0L and P0H) of the real-time output function and the compare register (CR01) of 16-bit timer 0 (TM0).

This can directly control the patterns output to the real-time output port and the output interval and simplify control of the open-loop for the stepping motor.

Figure 11-20 shows an example of open-loop control for the stepping motor in the basic operation mode. Figure 11-21 shows the timing at which data is output to the real-time output port in the example of Figure 11-20. In this example, the four low-order bits (P00 to P03) of the real-time output port are used to drive the stepping motor. The output interval timing is set when the value of 16-bit timer 0 (TM0) and the compare register (CR01) match.

Figure 11-19 shows an example of setting the macro service mode register in this example.

Fig. 11-19 Example of Setting the Macro Service Mode Register in the Basic Operation Mode

CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0	Macro service processing
1	1	1	0	1	0	0	0	MPT increment, without ring control, data transfer only (P00-P03)

Fig. 11-20 Open-loop Control for the Stepping Motor via the Realtime Output Port

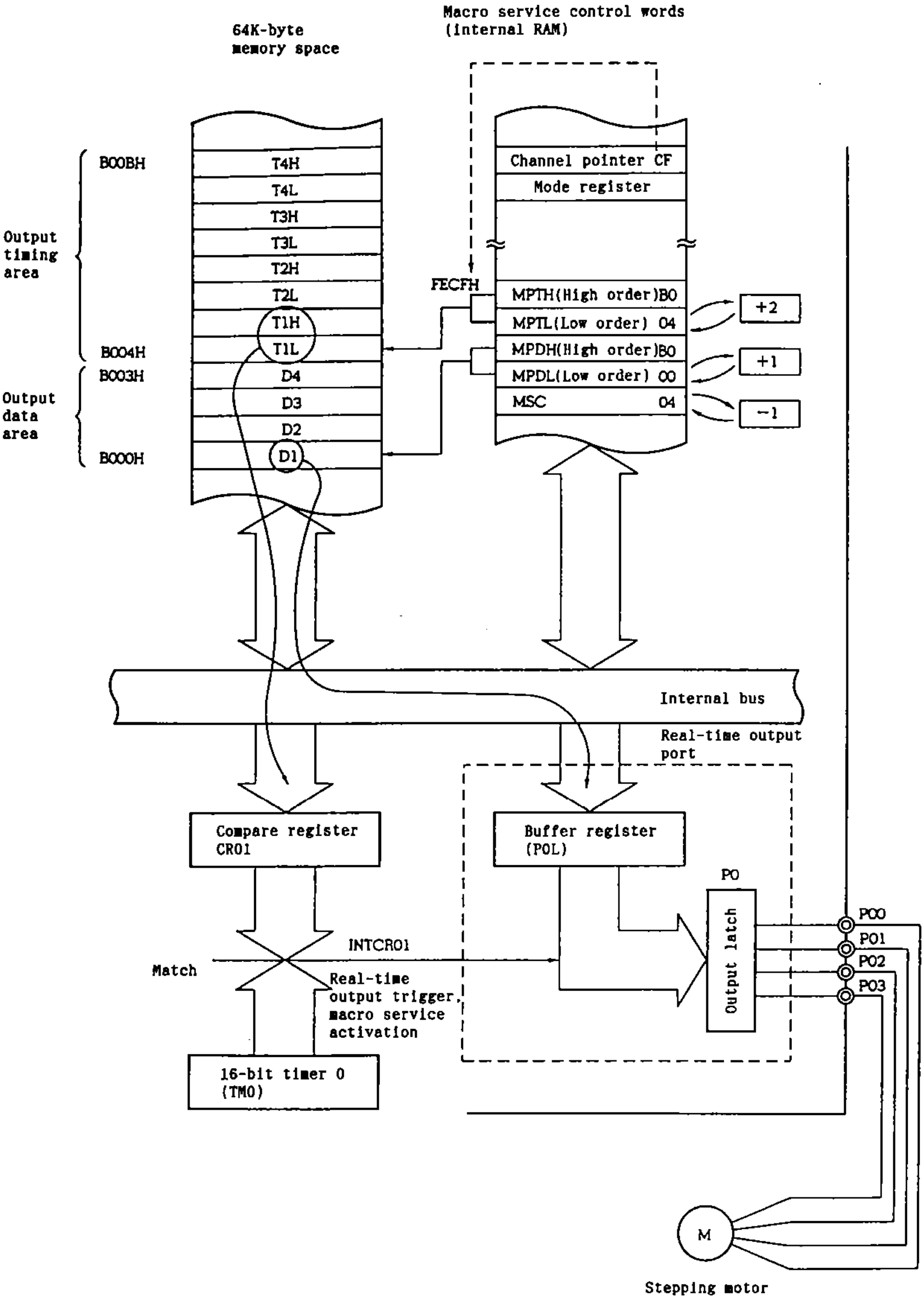


Fig. 11-21 Output Timing in the Real-time Output Port Mode

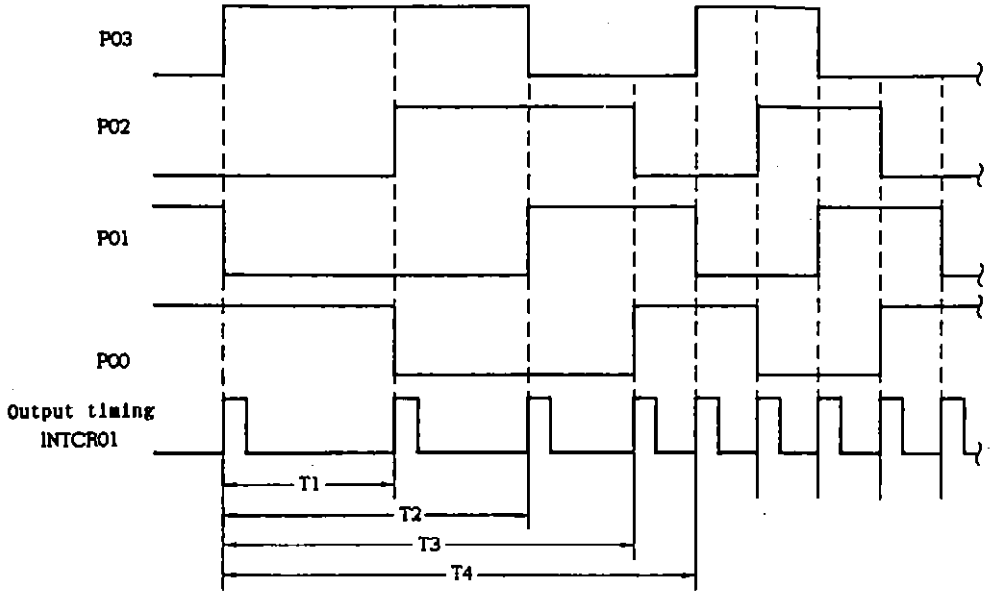


Figure 11-22 shows the timing diagram for data transfer control timer 0.

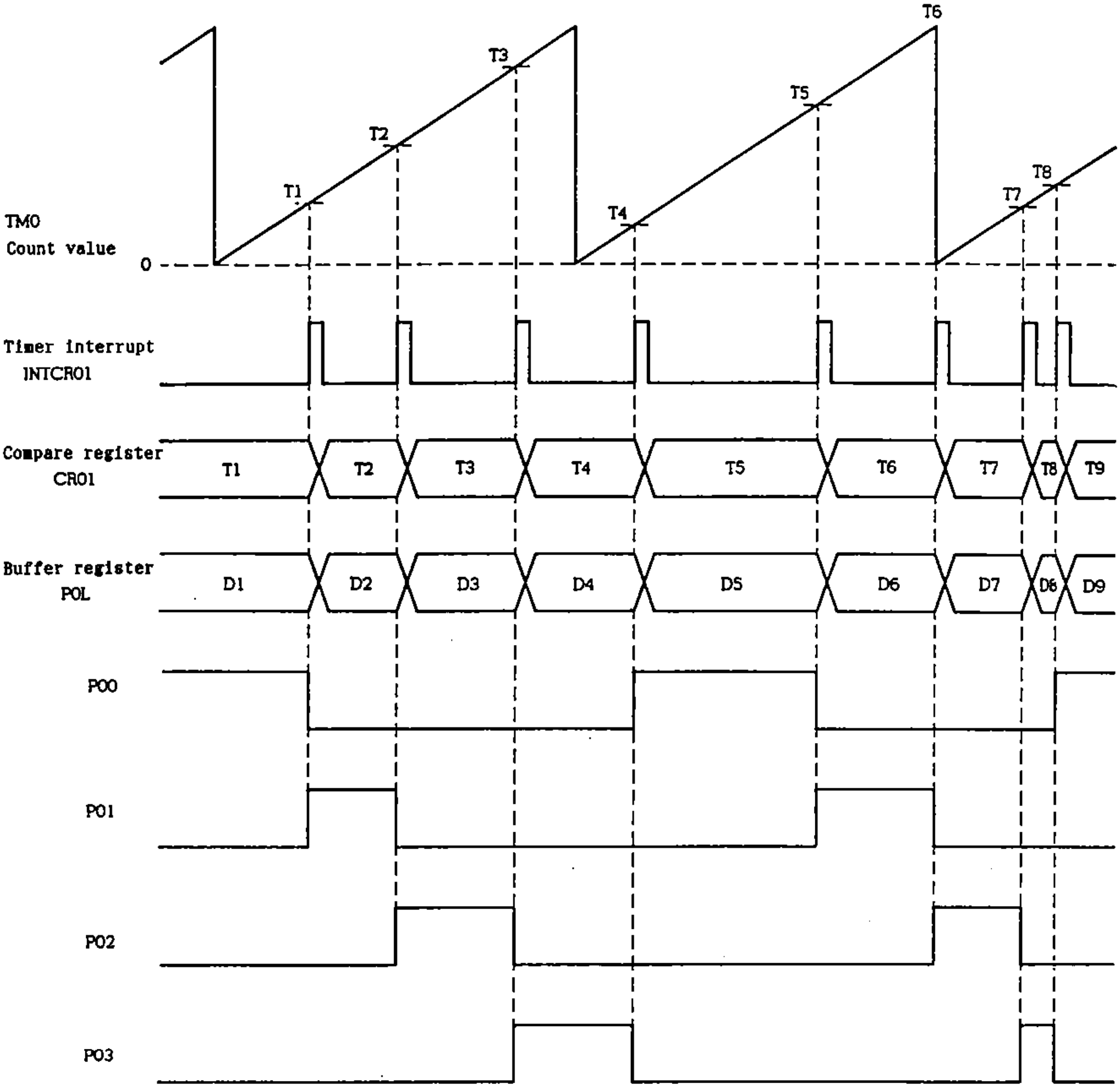
After changing the output data patterns (P00 to P03) to D1 to D4, set the macro service counter (MSC) to 04H to cause a vectored interrupt.

When timer 0 (TM0) and the compare register (CR01) match, the 16-bit data in a memory location addressed by the timer macro service pointer (MPT) is read and transferred to CR01. After this, MPTL is incremented by two.

Next, the 8-bit data stored in a memory location addressed by the data macro service pointer (MPD) is transferred to the buffer register (P0L) of the real-time output port and MPDL is incremented by one.

Finally, MSC is decremented by one and it is determined whether a vectored interrupt occurs.

Fig. 11-22 Data Transfer Control Timing



(b) Overview of automatic addition control and ring control

(i) Automatic addition control

In the basic operation, the value of the output timing data stored in the 64K-byte space is transferred to the compare register of timer 0. However, under automatic addition control, the output timing data (Δt) specified by the macro service pointer (MPT) is added to the contents of the compare register and the result of addition is written back into the compare register.

Use of automatic addition control eliminates the need to calculate the set value of the compare register in the program every time.

(ii) Ring control

The method for controlling the stepping motor depends upon the configuration of the stepping motor and the phase excitation method (1-phase or 2-phase excitation).

Figure 11-23 shows an example of the timing at which a 4-phase stepping motor is driven by 1-phase excitation. Figure 11-24 shows an example of the timing at which the 4-phase stepping motor is driven by 1-2 phase excitation.

For 1-phase excitation, a cycle consists of four patterns of output data. For 1-2 phase excitation, a cycle consists of eight patterns of output data.

Under ring control, a fixed cycle of output data patterns is repeated in the ring format and is output sequentially.

Thus, the data ROM area can be reduced.

When the result of decrementing the ring counter is 0, the macro service counter (MSC) is decremented by one (see Figure 11-18).

When MSC is 0 under ring control, an interrupt request is also issued.

Fig. 11-23 1-phase Excitation of a 4-phase Stepping Motor

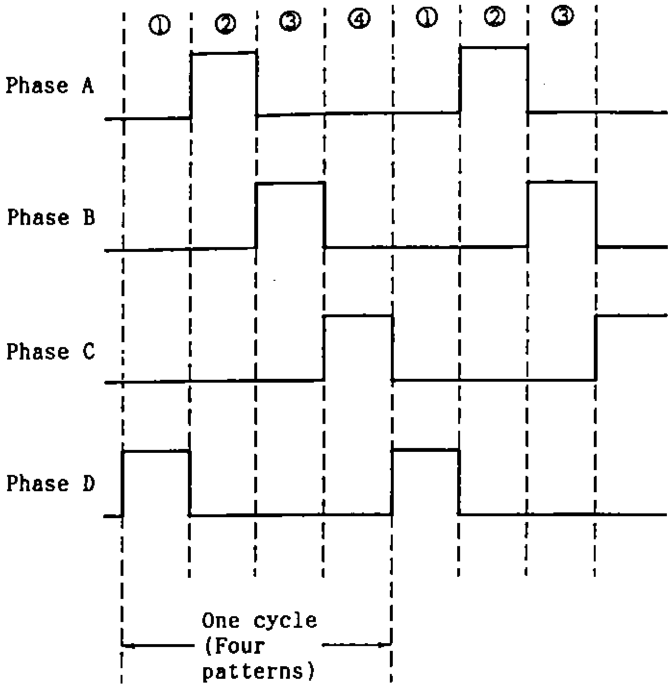
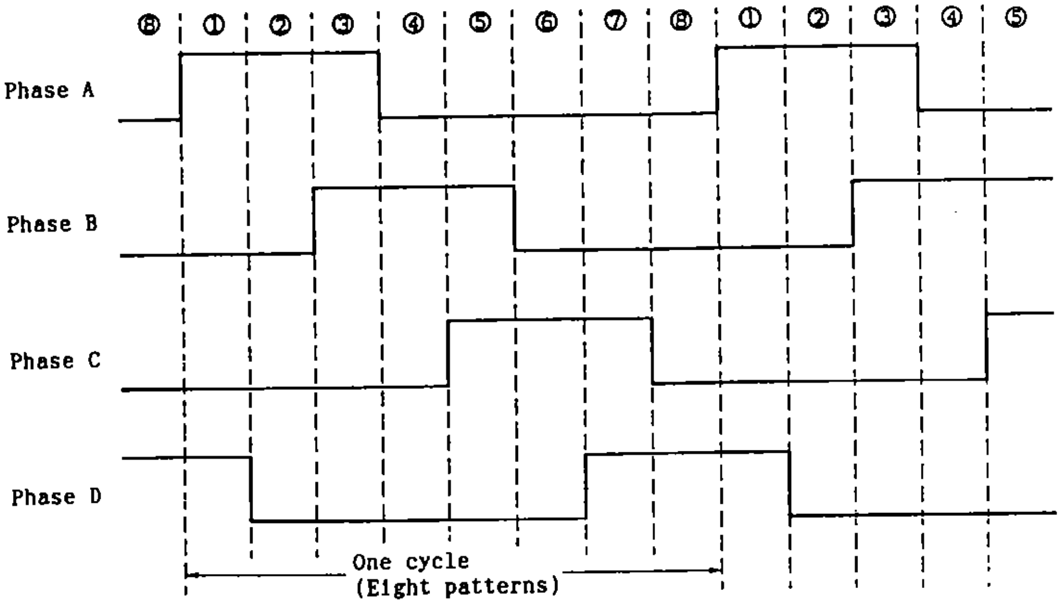


Fig. 11-24 1-2 Phase Excitation of a 4-phase Stepping Motor



(c) Examples of using automatic addition control and ring control together

Using addition control and ring control together efficiently controls the uniform motion, acceleration, and deceleration of the stepping motor.

Figure 11-26 shows the block diagram when controlling the uniform motion of the stepping motor driven by 1-2 phase excitation.

The output interval for the uniform motion is constant, that is, a fixed value is always added to the compare register (CR01) of timer 0. The timer macro service pointer (MPTH and MPTL) is therefore held and the address indicated by MPT stores the digital value (Δt) to be added to CR01 under automatic addition control.

1-2 phase excitation performs ring control, regarding eight patterns, D0 to D7, as a cycle. The ring counter and the modulo register therefore stores 07H.

When the value of timer 0 (TM0) and the compare register (CR01) match, the data macro service pointer (MPD) is incremented. After eight patterns of data are output, MPD returns to the address where the first data (D0) is stored, then the eight patterns are output repeatedly. Thus, the fixed data patterns are output in the ring format.

The modulo register is used to reload the initial value when the ring counter contents are 0.

Figure 11-27 shows the timing diagram when controlling the uniform motion of the stepping motor driven by 1-2 phase excitation.

For uniform motion control, set the mode in which the timer macro service pointer (MPT) is held.

Figure 11-25 shows an example of setting the macro service mode register in this case.

Fig. 11-25 Example of Setting the Macro Service Mode Register for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion)

CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0	Macro service processing
1	1	0	0	1	1	1	0	MPT hold, ring control, automatic addition (P00-P03)

Figure 11-29 shows the block diagram when controlling the stepping motor driven by 2-phase excitation at varying output intervals.

Since the output timing varies in this case, the digital value to be added automatically to the compare register (CR01) also changes. The values to be added are stored as 16-bit data (⊿t1 to ⊿t9) in a memory location addressed by the timer macro service pointer (MPT).

Two phase excitation performs ring control, regarding four output data patterns, D0 to D3, as a cycle.

Phase-out/Discontinued

Fig. 11-26 Block Diagram for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion)

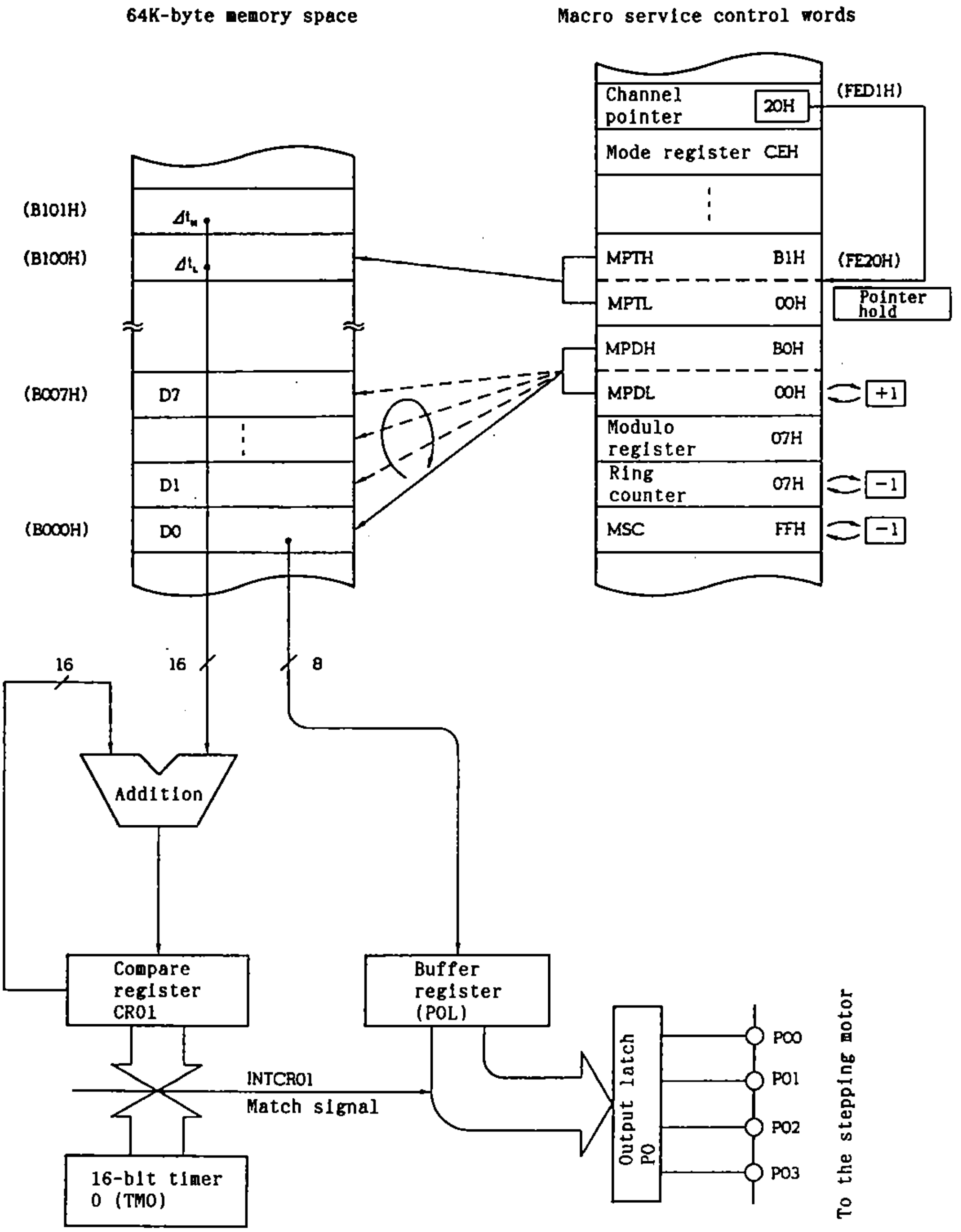
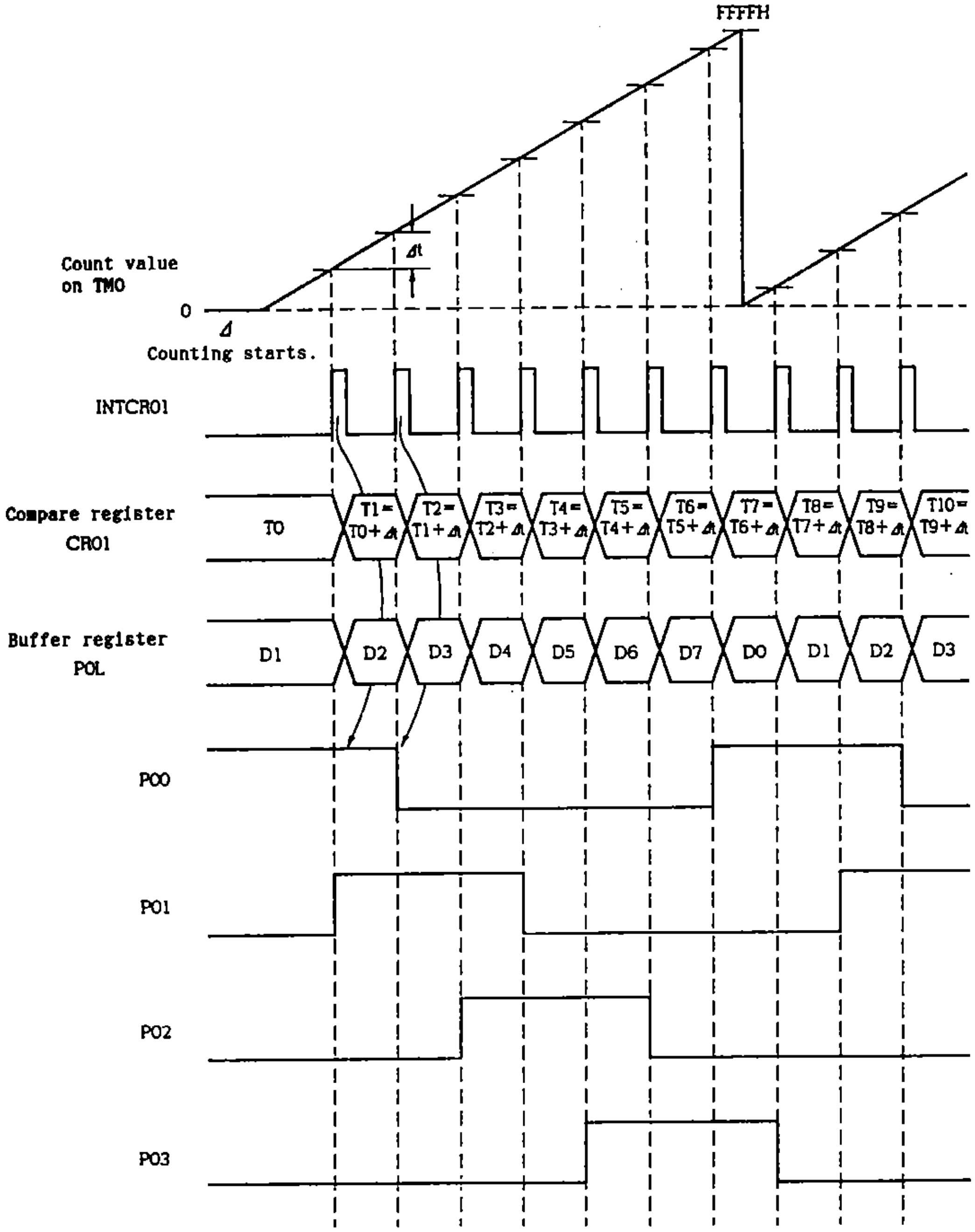


Fig. 11-27 Timing Diagram 1 for Automatic Addition Control Plus Ring Control (1-2 Phase Excitation Uniform Motion)



Caution: Set the mode in which MPT is held.

The ring counter therefore stores 03H.

Figure 11-30 shows the timing diagram when controlling the stepping motor driven by 2-phase excitation at varying output intervals.

Figure 11-28 shows an example of setting the macro service mode register in this case.

Fig. 11-28 Example of Setting the Macro Service Mode Register for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals)

CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0	Macro service processing
1	1	1	0	1	1	1	0	MPT increment, ring control, automatic addition (P00-P03)

Fig. 11-29 Block Diagram for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals)

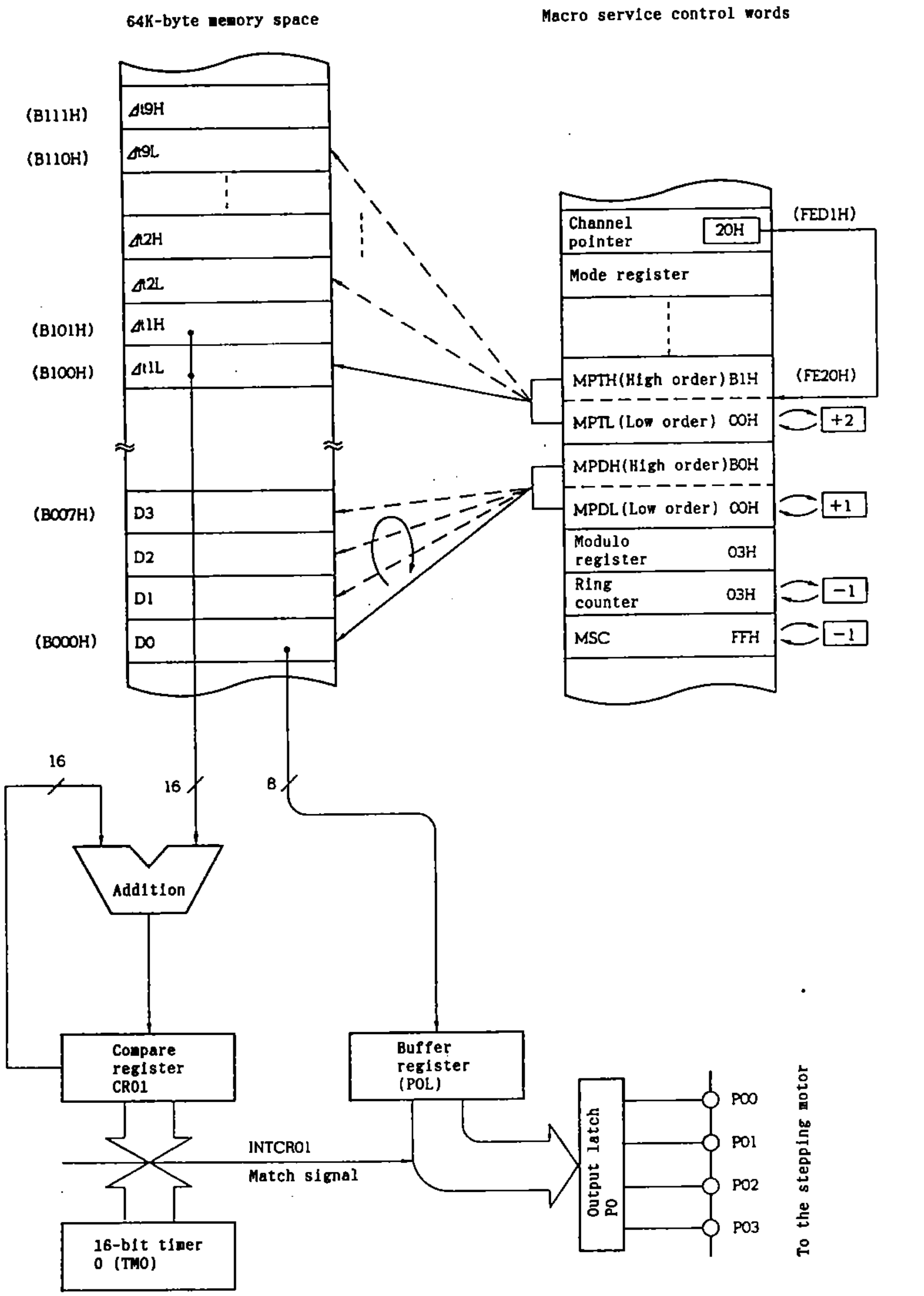
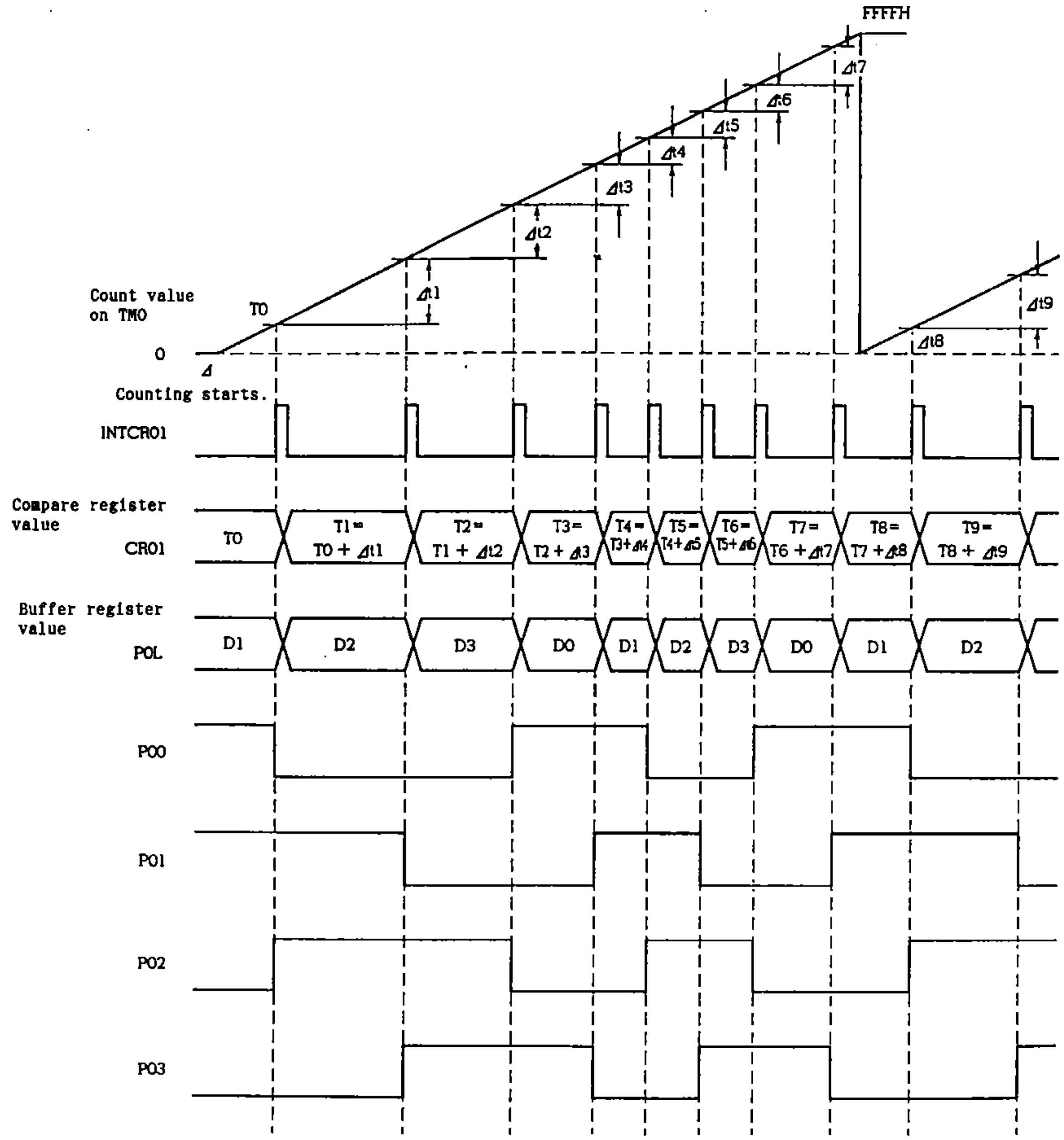


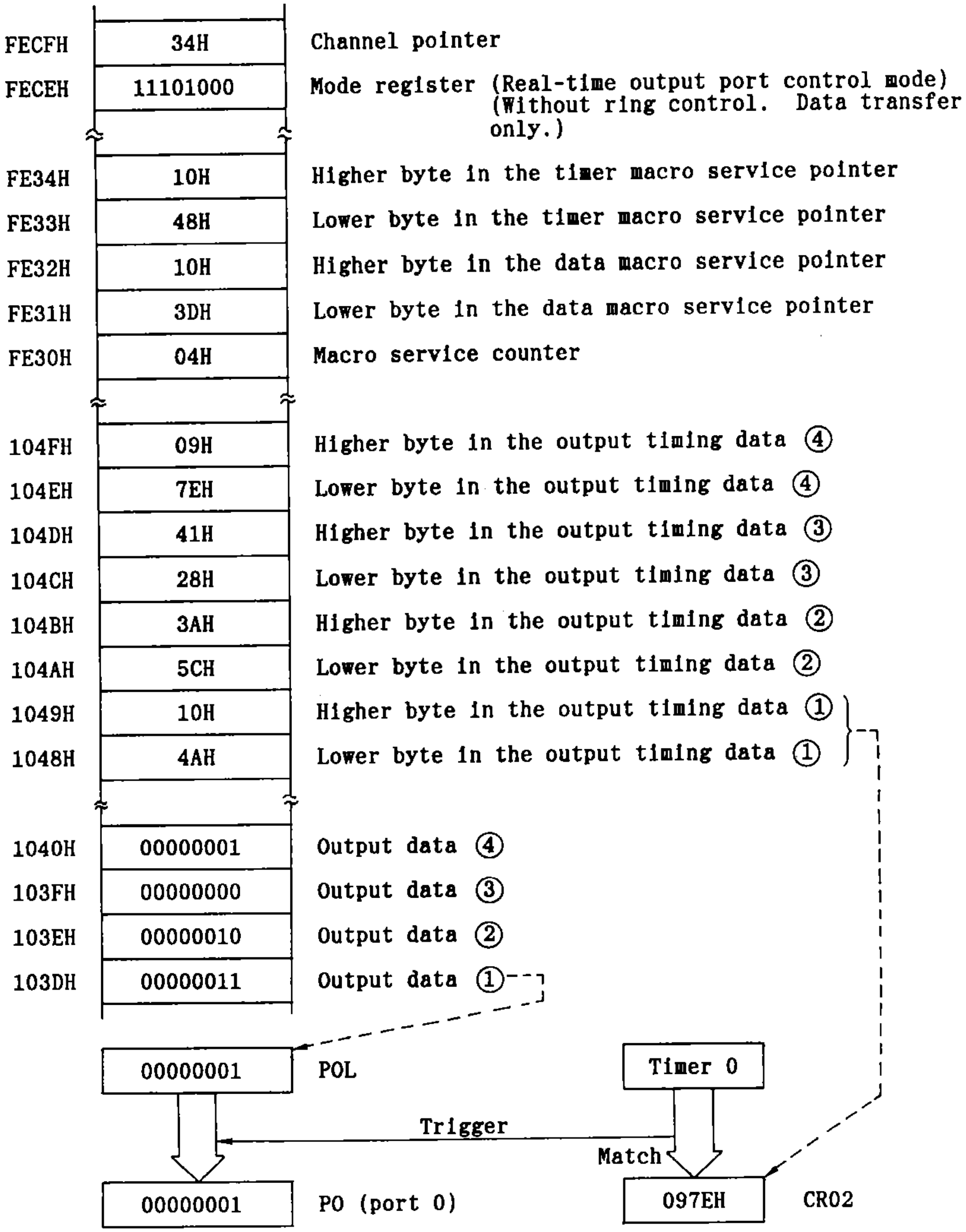
Fig. 11-30 Timing Diagram 2 for Automatic Addition Control Plus Ring Control (2-phase Excitation at Varying Intervals)



Caution: Set the mode in which MPT is incremented.

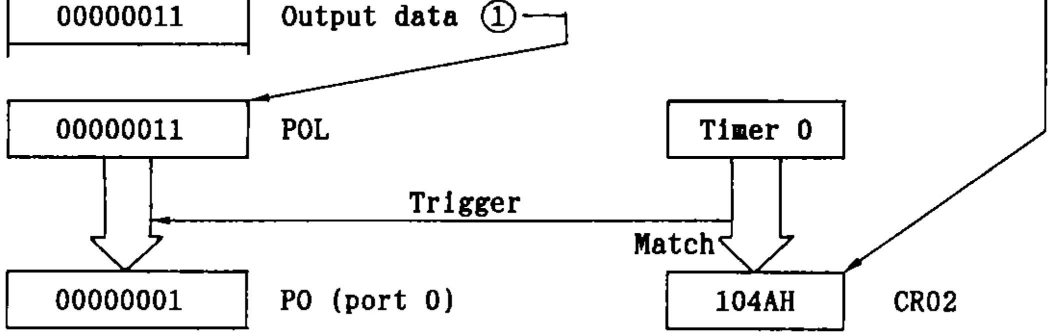
Fig. 11-31 Example of Macro Service Operation

(1) Initial state (when the first interrupt occurs)

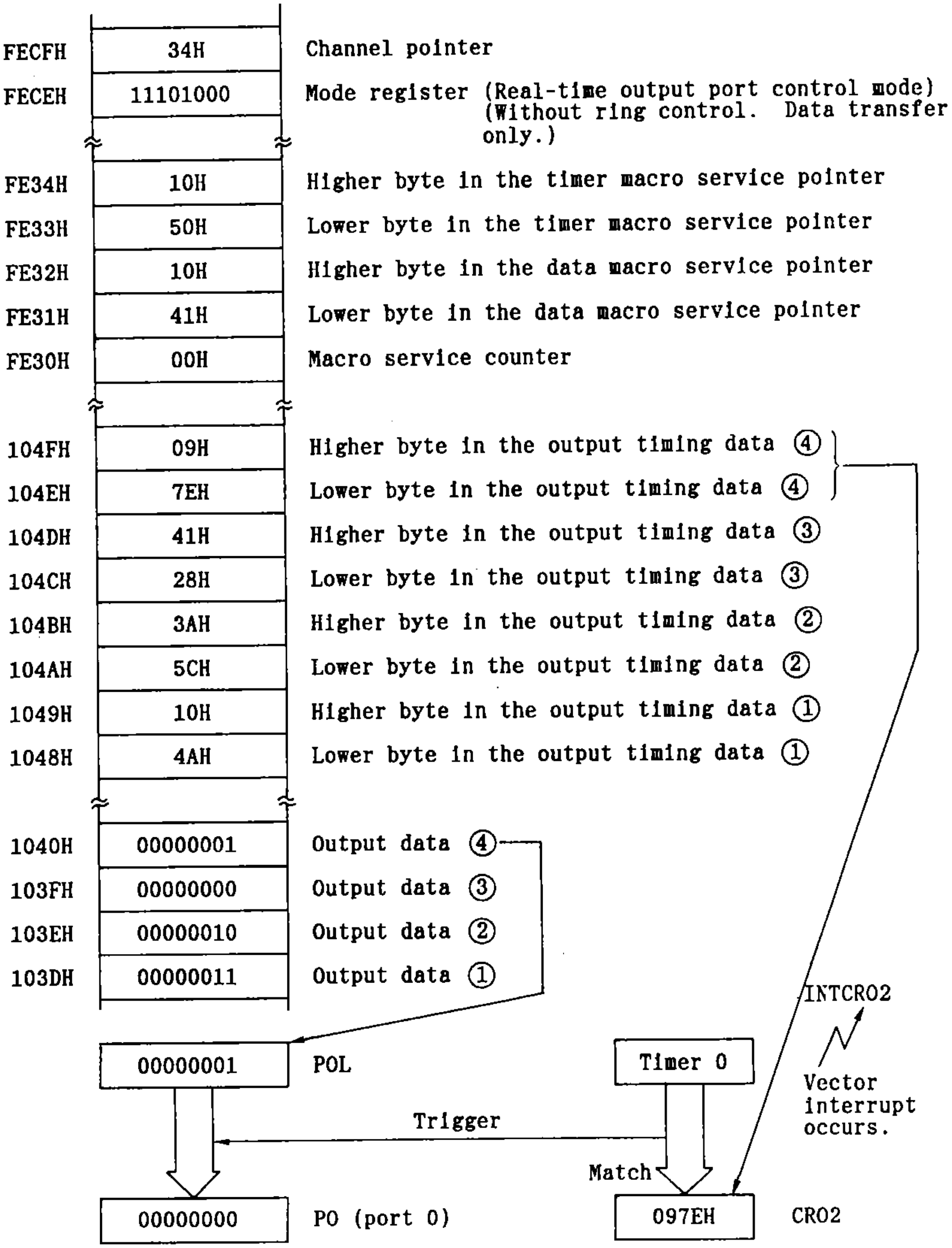

FECFH
34H
Channel pointer
FECEH
11101000
Mode register (Real-time output port control mode)
(Without ring control. Data transfer only.)
FE34H
10H
Higher byte in the timer macro service pointer
FE33H
48H
Lower byte in the timer macro service pointer
FE32H
10H
Higher byte in the data macro service pointer
FE31H
3DH
Lower byte in the data macro service pointer
FE30H
04H
Macro service counter
104FH
09H
Higher byte in the output timing data ④
104EH
7EH
Lower byte in the output timing data ④
104DH
41H
Higher byte in the output timing data ③
104CH
28H
Lower byte in the output timing data ③
104BH
3AH
Higher byte in the output timing data ②
104AH
5CH
Lower byte in the output timing data ②
1049H
10H
Higher byte in the output timing data ①
1048H
4AH
Lower byte in the output timing data ①
1040H
00000001
Output data ④
103FH
00000000
Output data ③
103EH
00000010
Output data ②
103DH
00000011
Output data ①
00000001
POL
Timer 0
Trigger
Match
00000001
P0 (port 0)
097EH
CR02

(2) When the first macro service is completed

FECFH	34H	Channel pointer
FECEH	11101000	Mode register (Real-time output port control mode) (Without ring control. Data transfer only.)
FE34H	10H	Higher byte in the timer macro service pointer
FE33H	4AH	Lower byte in the timer macro service pointer
FE32H	10H	Higher byte in the data macro service pointer
FE31H	3EH	Lower byte in the data macro service pointer
FE30H	03H	Macro service counter
104FH	09H	Higher byte in the output timing data ④
104EH	7EH	Lower byte in the output timing data ④
104DH	41H	Higher byte in the output timing data ③
104CH	28H	Lower byte in the output timing data ③
104BH	3AH	Higher byte in the output timing data ②
104AH	5CH	Lower byte in the output timing data ②
1049H	10H	Higher byte in the output timing data ①
1048H	4AH	Lower byte in the output timing data ①
1040H	00000001	Output data ④
103FH	00000000	Output data ③
103EH	00000010	Output data ②
103DH	00000011	Output data ①



(3) When the macro services are completed


FECFH
34H
Channel pointer
FECEH
11101000
Mode register (Real-time output port control mode)
(Without ring control. Data transfer only.)
FE34H
10H
Higher byte in the timer macro service pointer
FE33H
50H
Lower byte in the timer macro service pointer
FE32H
10H
Higher byte in the data macro service pointer
FE31H
41H
Lower byte in the data macro service pointer
FE30H
00H
Macro service counter
104FH
09H
Higher byte in the output timing data ④
104EH
7EH
Lower byte in the output timing data ④
104DH
41H
Higher byte in the output timing data ③
104CH
28H
Lower byte in the output timing data ③
104BH
3AH
Higher byte in the output timing data ②
104AH
5CH
Lower byte in the output timing data ②
1049H
10H
Higher byte in the output timing data ①
1048H
4AH
Lower byte in the output timing data ①
1040H
00000001
Output data ④
103FH
00000000
Output data ③
103EH
00000010
Output data ②
103DH
00000011
Output data ①
00000001
POL
Timer 0
INTCR02
Vector interrupt occurs.
Trigger
Match
00000000
P0 (port 0)
097EH
CR02

11.5.6 Counter mode

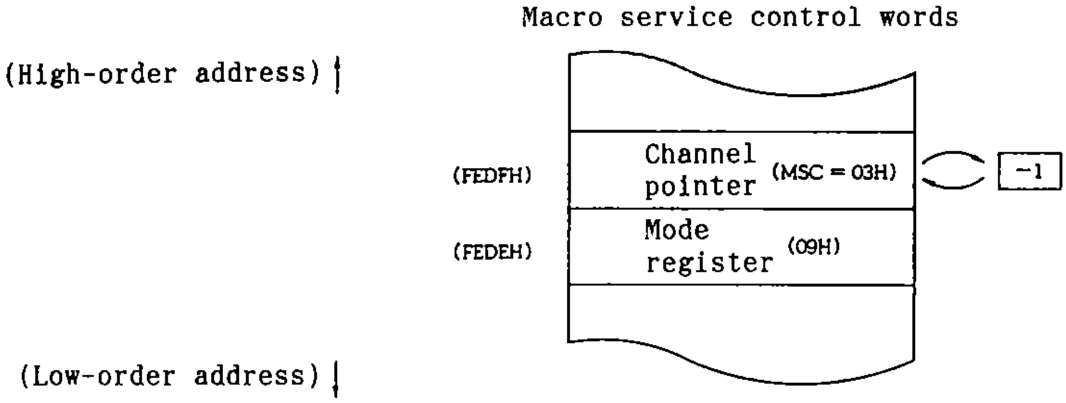
In the counter mode, a macro service is processed for all maskable interrupt requests.

A macro service channel pointer corresponding to each macro service request is used as the macro service counter (MSC).

When a macro service request is issued, the data set on MSC is decremented by one. When the value on MSC is 0, that is, a macro service request is issued the number of times set on MSC, a vectored interrupt occurs. After a vectored interrupt occurs, MSC remains 0. Use the interrupt service program to set MSC again. A macro service in this mode functions as the counter which frequency-divides the number of interrupt requests issued.

Figure 11-32 shows an example in which a macro service frequency-divides the number of INTCPT3 interrupt requests issued by 3.

Fig. 11-32 Example of a Counter Mode Operation



11.5.7 Data pattern identification mode

(1) Overview of function

In the data pattern identification mode, the data output from the control flip flop (CTL F/F) which is built in the pulse width measurement circuit (timer 3) of the super-timer unit is sequentially shifted to the right then stored in the buffer set in the RAM area.

The pulse width measurement circuit (see Figure 8-21) measures the duty of a pulse input to pin CTI11. When the measured duty is greater than the set value, 1 is latched in CTL F/F. When the measured duty is smaller than the set value, 0 is latched in CTL F/F. Storing this data string sequentially in RAM, for example, enables an index search signal of a VCR to be detected.

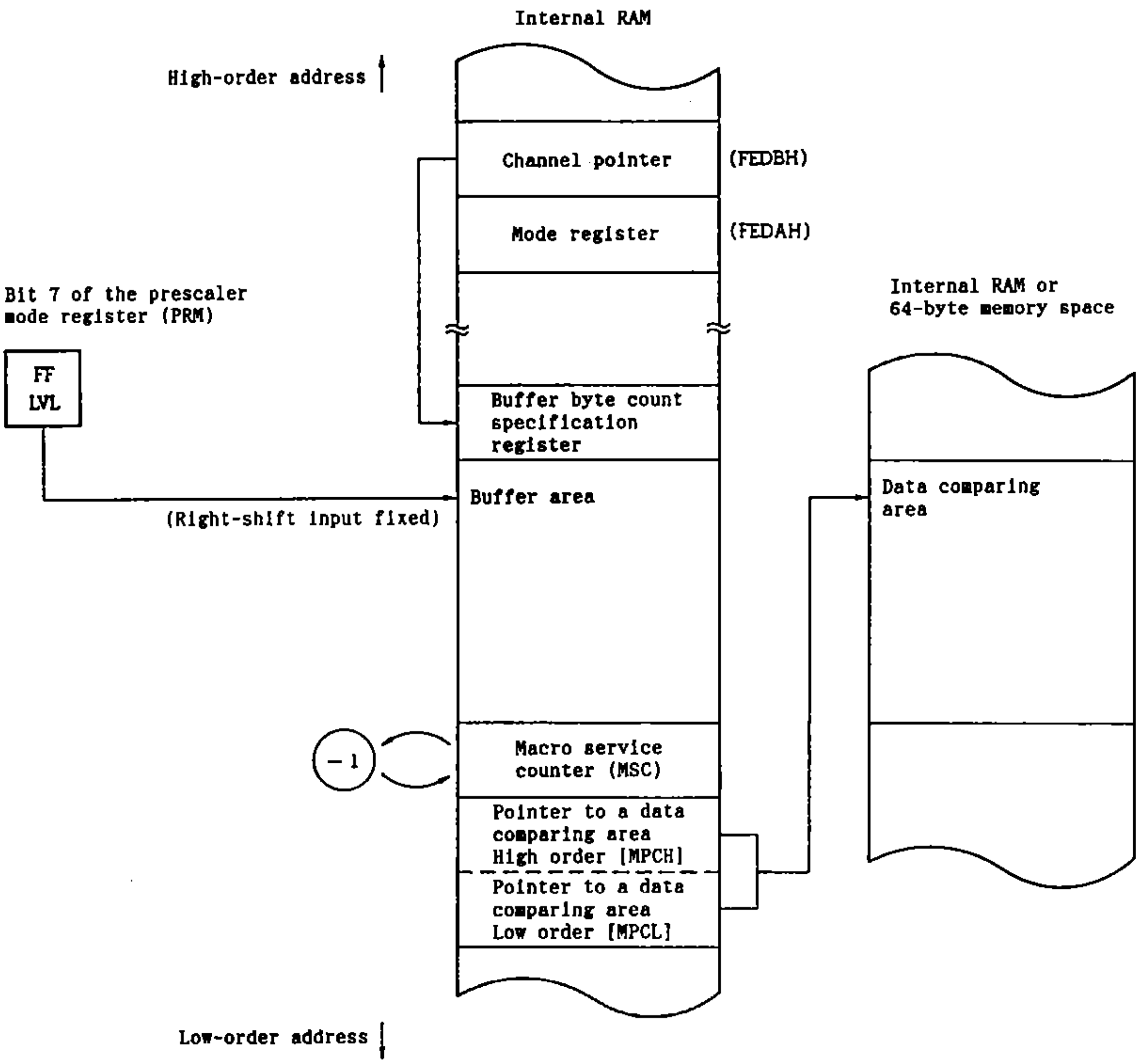
Figure 11-33 shows the addressing in the data pattern identification mode. Since this macro service is valid only for INTCR12 interrupt requests, a channel pointer is set to FEDBH and a mode register is set to FEDAH.

The channel pointer specifies the address of the buffer byte count specification register. This register specifies the number of bytes for data to be stored. A buffer area is allocated for this number of bytes.

The data output from CTL F/F is stored in bit 7 of prescaler mode register 3 (PRM3). This data is sequentially shifted to the right then stored in a buffer area.

The macro service counter (MSC) is decremented each time a bit of data is stored in a buffer area.

Fig. 11-33 Addressing in the Data Pattern Identification Mode


Internal RAM
High-order address
Channel pointer
(FEDBH)
Mode register
(FEDAH)
Bit 7 of the prescaler mode register (PRM)
FF
LVL
Internal RAM or 64-byte memory space
Buffer byte count specification register
Buffer area
(Right-shift input fixed)
Data comparing area
−1
Macro service counter (MSC)
Pointer to a data comparing area High order [MPCH]
Pointer to a data comparing area Low order [MPCL]
Low-order address

A vectored interrupt normally occurs when one of the following conditions is satisfied.

① When the contents of the macro service counter (MSC) are 0
(When an INTCR12 interrupt request is issued the number times set on MSC)

② When the data stored in a buffer area and the data in a data comparing area match (The data comparing area is set separately from the buffer area.)

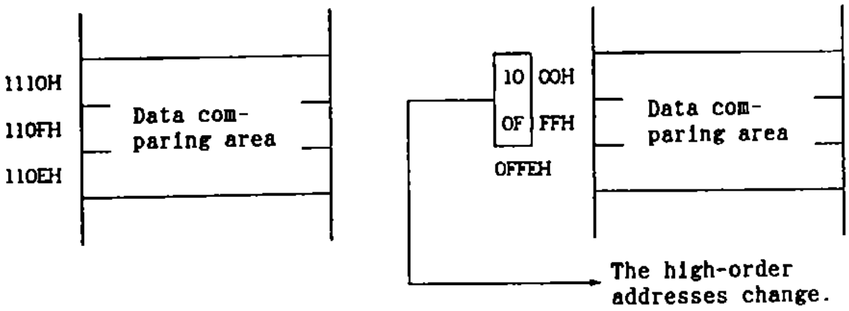
A vectored interrupt occurs when one of the above two conditions is satisfied according to the setting of the macro service mode register or when only condition ① is satisfied.

The data to be compared with the data stored in a buffer area is set in the address indicated by the pointer to a data comparing area (MPC).

The data comparing area may be specified in an internal RAM space or a program space in memory.

Cautions 1. A data comparing area cannot be set in the area whose high-order 8-bit address varies. (See Figure 11-34.)

Fig. 11-34 Note on Setting a Data Comparing Area



O (Can be set.)

× (Cannot be set.)

2. The time required to process a macro service in the data pattern identification mode depends upon the number of set buffer bytes.

The buffer byte count specification register can store only a value of 31H or less.

Cautions 3. To store the value of CTL F/F in a buffer area, the shift direction is fixed to the right.

(2) Example

Figure 11-36 shows an example of an application in the data pattern identification mode. In this example, the data pattern identification mode function is used for controlling the index search function of a VCR.

Pin CTI11 receives playback control (PBCTL) signals from a VCR. The pulse width detection circuit, which is built in the timer 1 (TM1) input part of the super-timer unit, judges the duty of an input control signal.

When digital data 0 or 1 is coded according to the duty of a playback control signal just as a VISS/VASS signal of a VHS VCR, the data coded by the pulse width detection circuit is decoded then right-shifted into a buffer area.

In an example of Figure 11-36, three bytes of data (01H, 23H, and 45H) are input to buffer areas. When the values in the buffer areas and the values set in the corresponding data comparing areas match, a vectored interrupt occurs.

Since three bytes of data are input, three bytes of buffer area are allocated. The buffer byte count specification register is therefore set to 03H.

When three bytes (24 bits) of data are stored, a value of greater than 18H is set to the macro service counter (MSC).

In this example, 19H is set.

When three bytes of data are stored by this setting, the data in the buffer areas are compared with the data in data comparing areas. If a match is found as a result of comparison, a vectored interrupt occurs.

If a match is not found because of noise, MSC is set to 0 by input of the 25th bit data and a vectored interrupt occurs.

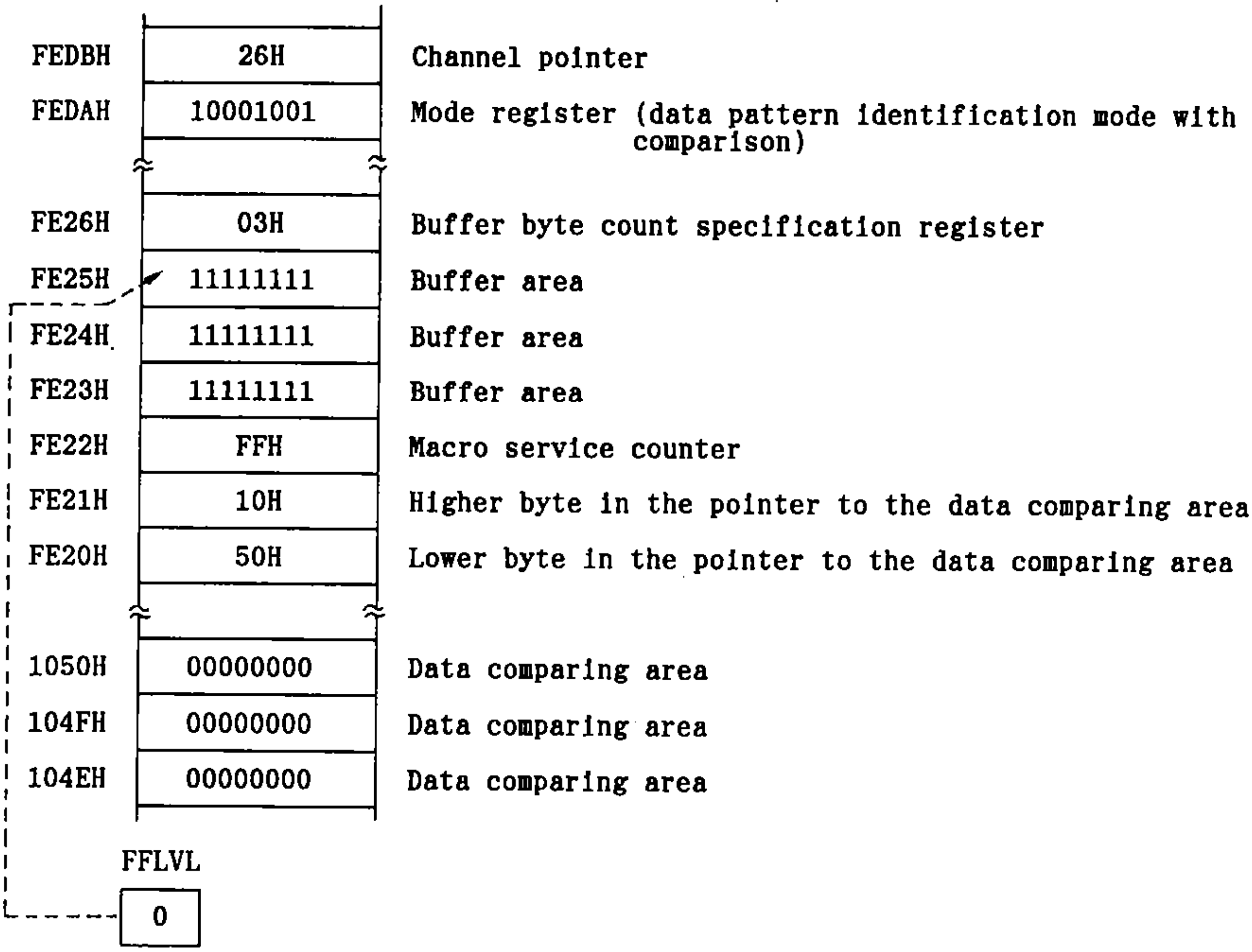
In this example, the macro service mode register is set as follows.

Fig. 11-35 Example of Setting the Macro Service Mode Register in the Data Pattern Identification Mode (with Comparison)

7	6	5	4	3	2	1	0	
CH2	CH1	CH0	0	MOD3	MOD2	MOD1	MOD0	Macro service processing
1	0	0	0	1	0	0	1	Data pattern identification mode, with comparison

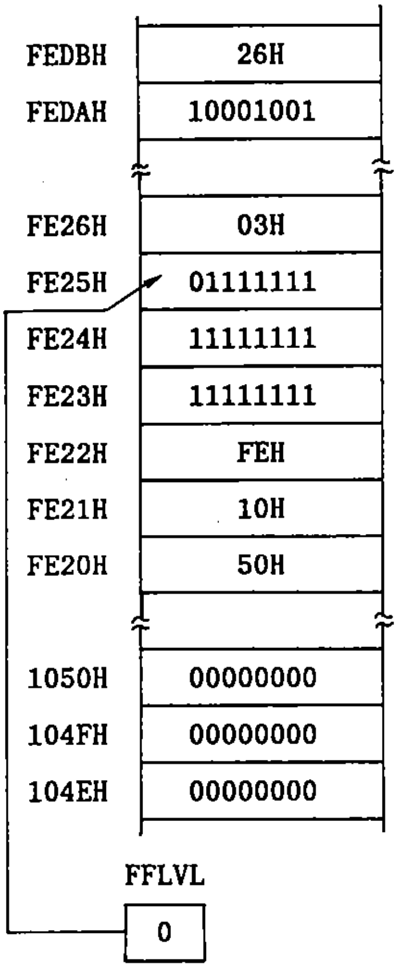
Fig. 11-36 Example of an Application in the Data Pattern Identification Mode (VCR Index Search Control)

(1) Initial state (when a value is latched in FFLVL)



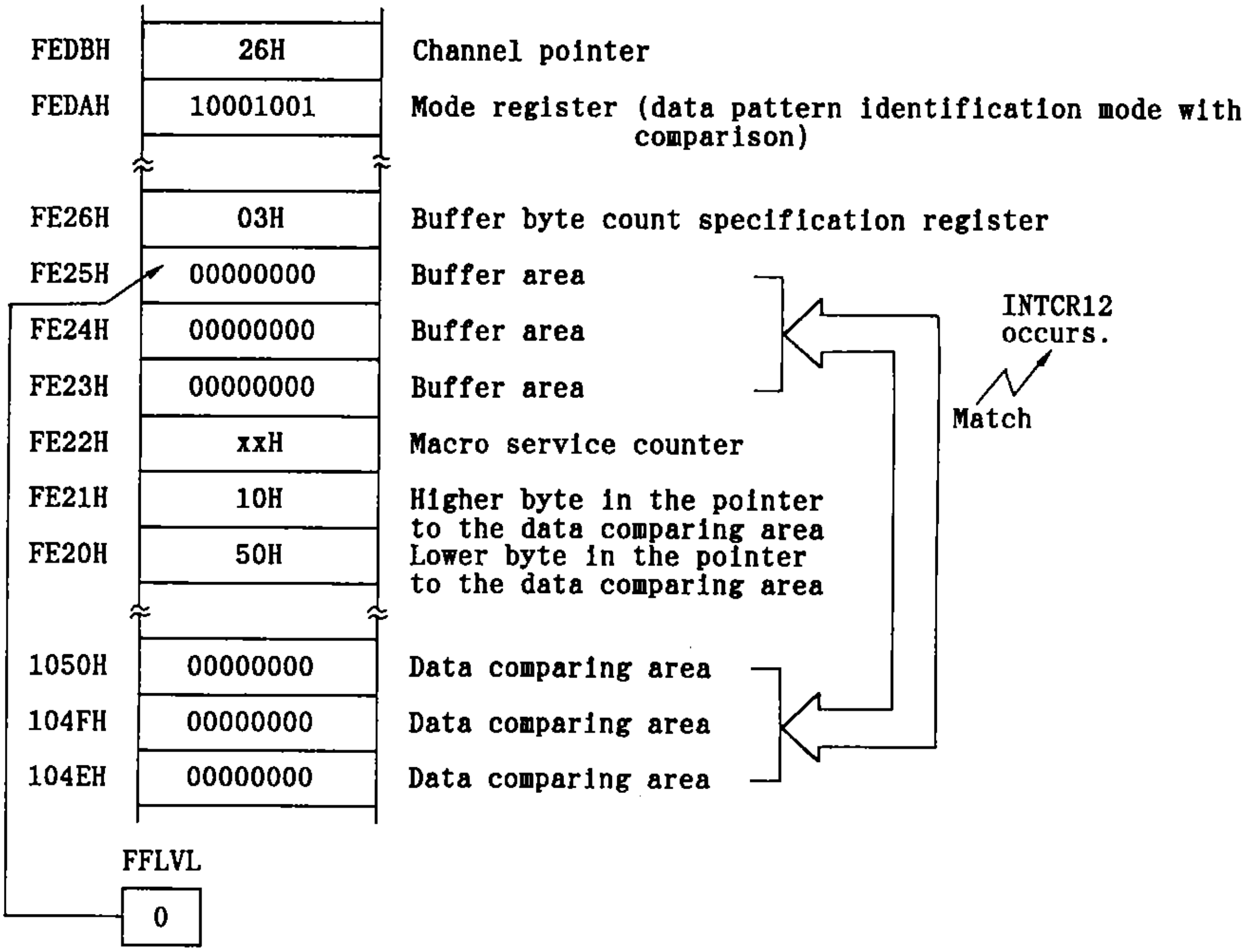
FEDBH	26H	Channel pointer
FEDAH	10001001	Mode register (data pattern identification mode with comparison)
FE26H	03H	Buffer byte count specification register
FE25H	11111111	Buffer area
FE24H	11111111	Buffer area
FE23H	11111111	Buffer area
FE22H	FFH	Macro service counter
FE21H	10H	Higher byte in the pointer to the data comparing area
FE20H	50H	Lower byte in the pointer to the data comparing area
1050H	00000000	Data comparing area
104FH	00000000	Data comparing area
104EH	00000000	Data comparing area

(2) When the first macro service is completed



FEDBH	26H	Channel pointer
FEDAH	10001001	Mode register (data pattern identification mode with comparison)
FE26H	03H	Buffer byte count specification register
FE25H	01111111	Buffer area
FE24H	11111111	Buffer area
FE23H	11111111	Buffer area
FE22H	FEH	Macro service counter
FE21H	10H	Higher byte in the pointer to the data comparing area
FE20H	50H	Lower byte in the pointer to the data comparing area
1050H	00000000	Data comparing area
104FH	00000000	Data comparing area
104EH	00000000	Data comparing area

(3) When the contents of the buffer area and data comparing area match (VISS detected)



FEDBH	26H	Channel pointer
FEDAH	10001001	Mode register (data pattern identification mode with comparison)
FE26H	03H	Buffer byte count specification register
FE25H	00000000	Buffer area
FE24H	00000000	Buffer area
FE23H	00000000	Buffer area
FE22H	xxH	Macro service counter
FE21H	10H	Higher byte in the pointer to the data comparing area
FE20H	50H	Lower byte in the pointer to the data comparing area
1050H	00000000	Data comparing area
104FH	00000000	Data comparing area
104EH	00000000	Data comparing area

(3) Number of clocks required for macro service processing

The number of clocks required for macro service processing depends upon the type of macro service processing. (See Table 11-9.)

Table 11-9 lists the number of clocks when the memory expansion mode register (MM) is set to 1000xxxxB and an instruction is executed in an internal ROM.

Table 11-9 Macro Service Processing Time

Type of macro service processing				Number of clocks
Data transfer mode		Memory to SFR		19
		SFR to memory		20
Real-time output port control macro	Without ring control	Data transfer only	Four low-order bits (P0L)	50
			Four high-order bits (P0H)	51
		With automatic addition	Four low-order bits (P0L)	57
			Four high-order bits (P0H)	58
	With ring control (*1)	Data transfer only	Four low-order bits (P0L)	55
			Four high-order bits (P0H)	56
		With automatic addition	Four low-order bits (P0L)	62
			Four high-order bits (P0H)	63
Counter mode				10
Data pattern identification mode		Shift only		20 + 6n(*2)
		(Shift) + (Comparison)		32 + 12n

*1 When the ring counter is 0 for ring control, five clocks are added to the corresponding values in the table.

*2 n indicates the value set by the buffer byte count specification register.

11.5.8 Points to be noted

Table 11-10 lists the ranges of addresses that cannot be used by the macro service function.

Table 11-10 Address Ranges that cannot be Used by the Macro Service Function

	Condition	Description
Limit on use of addresses for the macro service channel	Unconditional	The macro service channel must not be set within the range of FED0H to FEDFH.
Limit on use of addresses when the external expansion function is used	Single-chip mode	There is no limit on the available addresses for external expansion.
	256-byte expansion mode	External expansion addresses 0AH and 0CH must not be used *.
	External memory expansion mode	External expansion address FF0AH and FF0CH must not be used *.
	ROM-less mode	

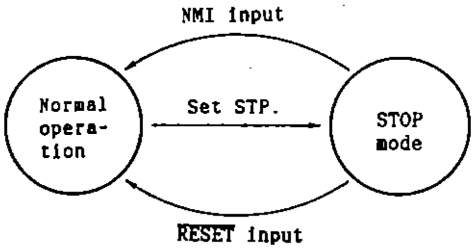
* For applications that use the macro service, do not use the above addresses as external addresses in any part of a program.

CHAPTER 12 STANDBY FUNCTION

The uPD78138 has a STOP mode as a standby function to reduce power consumption of the system. In this mode, the oscillator is stopped to stop the entire system. Data can be held at very low power consumption in which only a leakage current flows.

The STOP mode is set when the STOP flag (STP) is set to 1 by software. The STOP mode is released by a nonmaskable interrupt (NMI) or reset ($\overline{\text{RESET}}$) input. Figure 12-1 shows standby status transition.

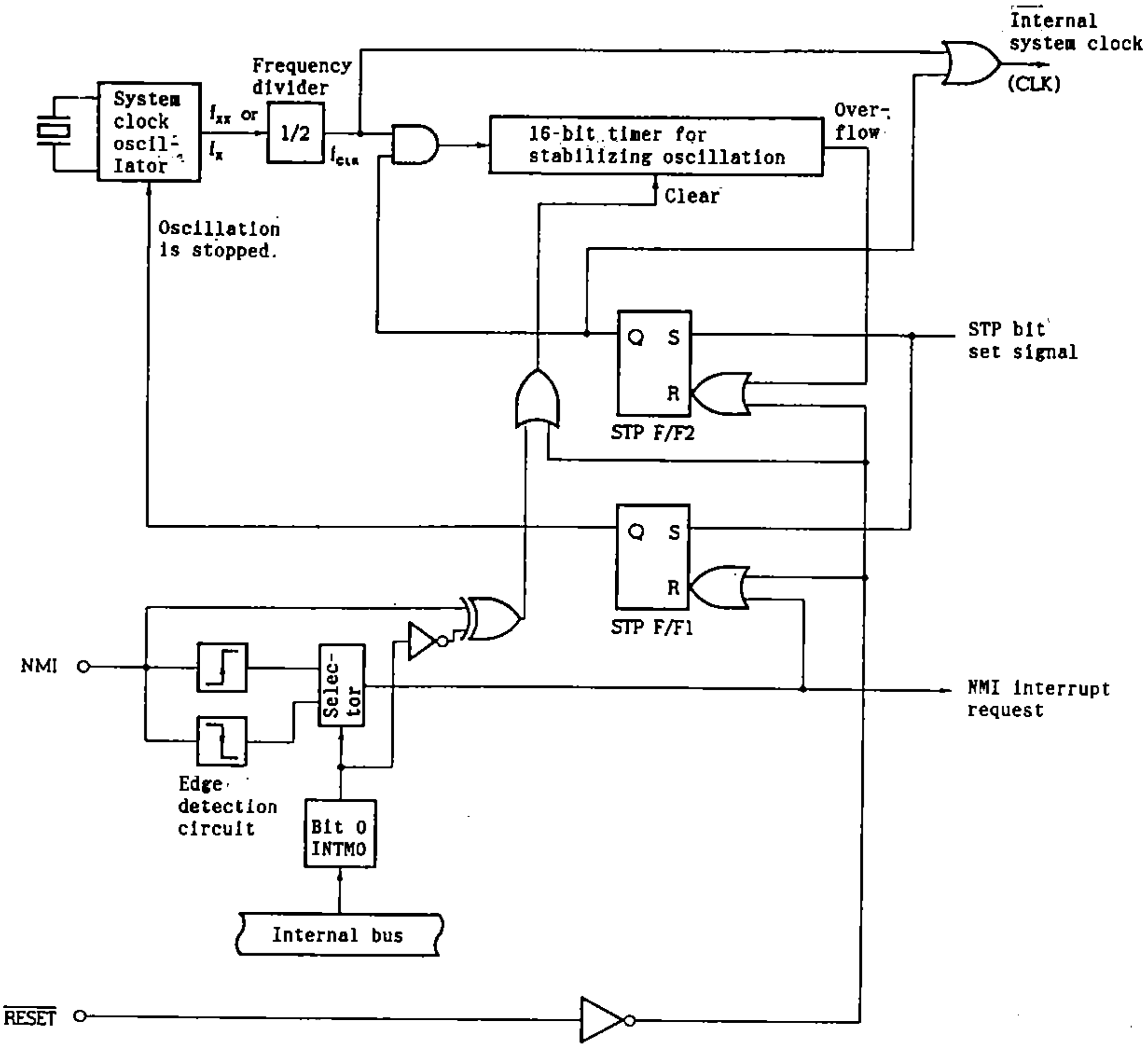
Fig. 12-1 Standby Status Transition



12.1 Configuration of Standby Function Control Circuit

Figure 12-2 shows the configuration of a standby function control circuit.

Fig. 12-2 Configuration of a Standby Function Control Circuit



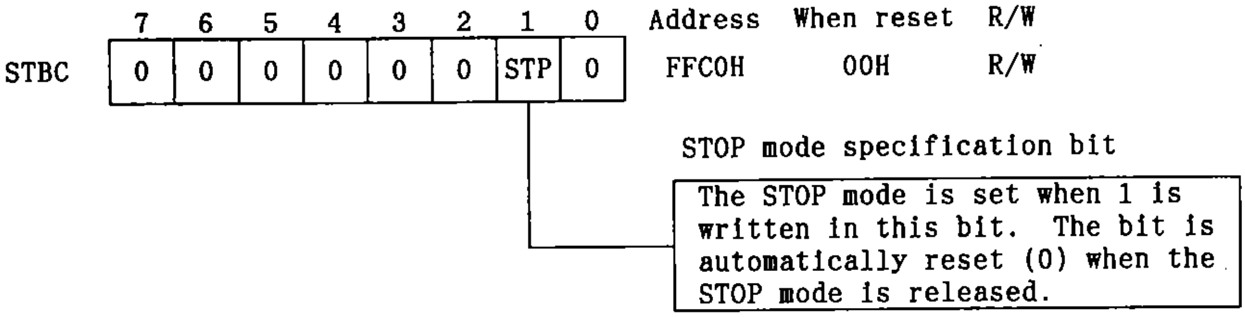
Remark: INTM0 indicates an external interrupt mode register (see Figure 11-2).

(1) Standby control register (STBC)

The STBC register is an 8-bit register which controls the standby mode. The contents of the STBC register can be read and written. They can, however, be written only by a dedicated instruction (MOV STBC, #byte). Figure 12-3 shows the format of the STBC register.

The STBC register will be reset to 00H by $\overline{\text{RESET}}$ input.

Fig. 12-3 Format of the Standby Control Register (STBC)



	7	6	5	4	3	2	1	0	Address	When reset	R/W
STBC	0	0	0	0	0	0	STP	0	FFC0H	00H	R/W

STOP mode specification bit

The STOP mode is set when 1 is written in this bit. The bit is automatically reset (0) when the STOP mode is released.

12.2 Setting the STOP Mode and Operation States in the STOP Mode

The STOP mode is set by setting the STP bit in the STBC register to 1.

Eight-bit data can be written in the STBC register only by a dedicated instruction. The MOV STBC, #02H instruction must be therefore specified to set the STOP mode.

Table 12-1 Operation States in the STOP Mode

Clock oscillator	Stopped
Internal system clock	Stopped
CPU	Stopped
I/O line	The status before setting the STOP mode is retained.
Each internal block	Stopped
Data retention	All internal data such as the CPU status and the contents of the RAM are retained.

Cautions 1. When the STOP mode is set, pin X1 is internally connected to V_{SS} (GND potential) to prevent leak at the clock oscillator. The STOP mode must not therefore be set in the system using an external clock.

2. The STOP mode must not be set while the A/D converter is operating.

12.3 Releasing the STOP Mode

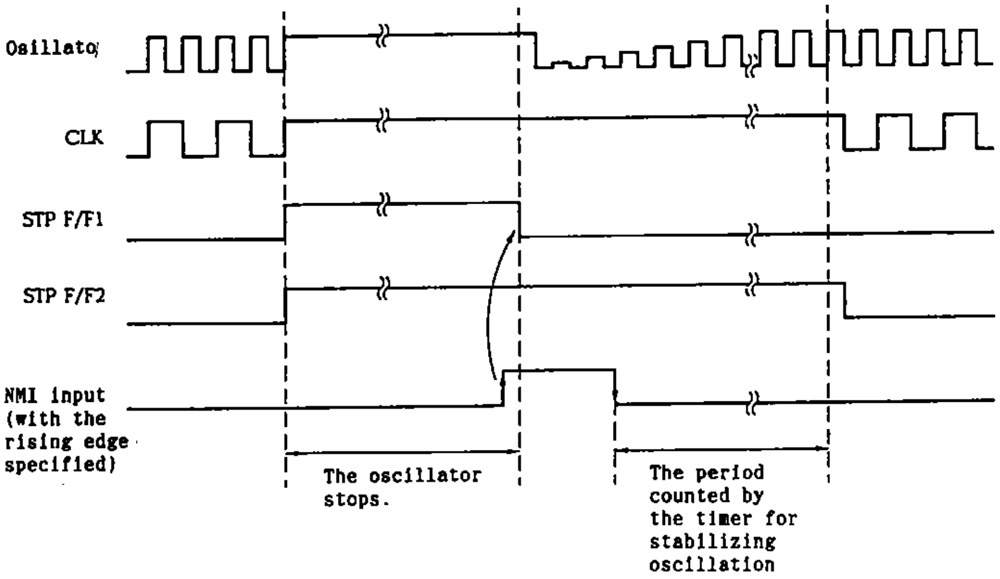
The STOP mode can be released by NMI or $\overline{\text{RESET}}$ input.

12.3.1 Releasing the STOP mode by NMI input

The oscillator restarts when the edge specified in the external interrupt mode register (INTM0) is detected in NMI input. When the NMI input level reaches the original level, the 16-bit counter for stabilizing oscillation starts counting. When the counter overflows, generation of the internal system clock is started. The system is therefore in the wait state during the high or low level width after detecting the NMI input edge and counter overflow time. This waiting time allows the oscillation to stabilize (see Figure 12-4).

After the STOP mode is released, the system branches to the NMI interrupt service program.

Fig. 12-4 Releasing the STOP Mode by NMI Input


Osillato
CLK
STP F/F1
STP F/F2
NMI input (with the rising edge specified)
The oscillator stops.
The period counted by the timer for stabilizing oscillation

12.3.2 Releasing the STOP mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input is changed from high to low, the system is put in the reset state and the oscillator restarts.

Releasing the STOP mode by $\overline{\text{RESET}}$ input is different from doing by NMI input. The system does not enter the wait state by the counter for stabilizing oscillation. When the terminal level is changed from low to high, the system starts instruction execution even if the oscillator operates unstable. The low level width must be used enough to reserve time for stabilizing oscillation.

The contents of data memory are retained as they were before setting the STOP mode in the different way from normal reset operation.

CHAPTER 13 RESET FUNCTION

When input to pin $\overline{\text{RESET}}$ becomes low, the system is reset and each hardware component is put in the status shown in Table 13-1.

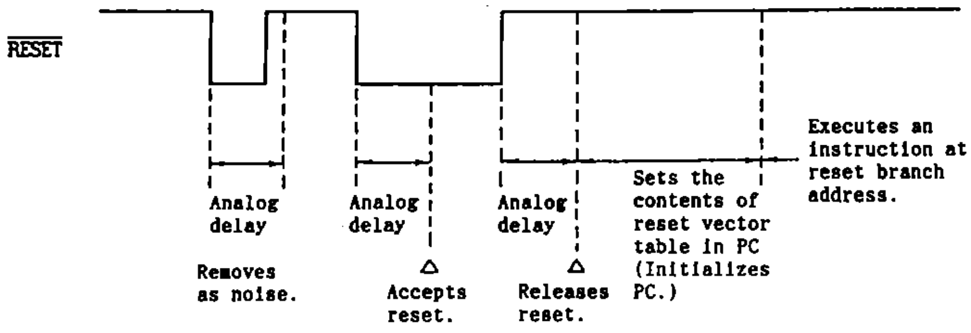
When input to pin $\overline{\text{RESET}}$ becomes high, the reset status is released. The contents at address 0000H in a reset vector table are then set in bits 7 to 0 in the program counter (PC) and the contents of bits 7 to 0 at address 0001H are set in bits 15 to 8 in the PC. The branch is taken in this way and program execution starts at the branch destination address. Reset start is possible at any address.

Initialize the contents of registers in the program as required.

A noise eliminator using analog delay is provided for the $\overline{\text{RESET}}$ input pin to prevent miss-operation due to noise (Figure 13-1).

For the reset operation at power-on, reserve time for the oscillation to stabilize from power-on to releasing the reset signal as shown in Figure 13-2.

Fig. 13-1 Accepting a Reset Signal



RESET
Analog delay
Removes as noise.
Analog delay
Accepts reset.
Analog delay
Releases reset.
Sets the contents of reset vector table in PC (Initializes PC.)
Executes an instruction at reset branch address.

Fig. 13-2 Reset at Power-on

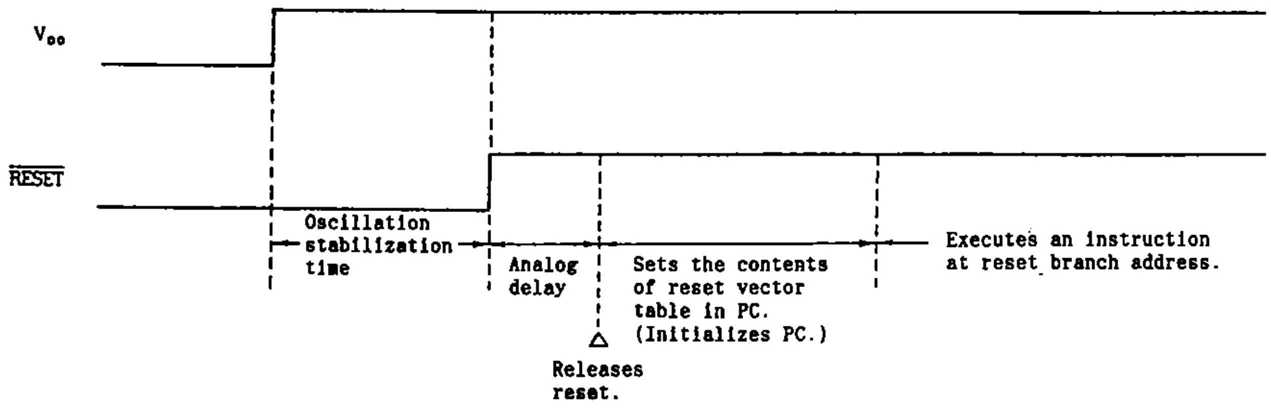

V_DD
RESET
Oscillation stabilization time
Analog delay
Sets the contents of reset vector table in PC. (Initializes PC.)
Releases reset.
Executes an instruction at reset branch address.

Table 13-1 Hardware Statuses after Reset

Hardware		Status after reset
Program counter (PC)		The contents of a reset vector table (0000H, 0001H) are set.
Stack pointer (SP)		Undefined
Program status word (PSW)		02H
Built-in RAM	Data memory	Undefined(*)
	General registers (X, A, C, B, E, D, L, and H)	
I/O line	P00-P07	High impedance (output buffer off)
	P20-P27, and P34-P37	Input
	P10-P17, P30-P33, P40-P47, P50-P57, P64-P67, P70, and P71	Input (output buffer off)
	P60-P63	Low-level output

(to be continued)

* Retains the value before setting the STOP mode when the STOP mode is released by RESET input.

Table 13-1 Hardware Statuses after Reset (Cont'd)

Hardware			Status after reset
Output latch	Ports 0, 1, 3, 4, 5, and 7		Undefined
	Port 6		xxxx0000
Port mode register	PM0, PM1, PM3, PM5, PM7		FFH
	PM6		F0H
Port 3 mode control register (PMC3)			30H
Memory mapping register (MM)			20H
Register for optional pull-up resistor (PUO)			00H
Super timer unit	Counters (TM0, TM1, FRC, and TM2)		Up to 16 clocks after releasing the reset signal: Undefined 17 clocks or more after releasing the reset signal: Zero clear
	Compare registers (CR00, CR01, CR02, CR10, CR11, CR20)		Undefined
	Capture registers (CR12, CPT0, CPT1, CPT2H, CPT2L, CPT3)		
	Timer control register 0 (TMC0)		0xx00000
	Timer control register 1 (TMC1)		00H
	Capture mode register (CPTM)		xxxxx000
	Input control register (ICR)		0x000xxx
	External capture input mode register (INTM1)		0000xx01
	Event counters (CTI00 input section)	EC	xx000000
		ECC0	xx111111
		ECC1	xx111111
	Event divider control register (CTI10 input section)	EDVC	Undefined

(to be continued)

Table 13-1 Hardware Statuses after Reset (Cont'd)

Hardware			Status after reset
Super timer unit	Pulse width detection circuit (CTI11 input section)	TM3	00H
		PRM3	0xxxx000
		CR30	x1111111
		CPT30	Undefined
	Timer output mode register	TOM0	xx000000
		TOM1	xxxx0000
	Timer output control register	TOC0	xx000000
		TOC1	xxxx0000
	PTO10 output (PTO10)		High-level output
	PWM outputs (PWM0 and PWM1)		Low-level output
	PWM control register (PWMC)		05H
	PWM modulo registers (PWM0 and PWM1)		Undefined
Real-time output port	Port 0 buffer registers (P0L and P0H)		Undefined
	Real-time output port control register (RTPC)		00H
A/D converter	Mode register (ADM)		00H
	A/D conversion result register (ADCR)		Undefined
Serial interface	Mode register (CSIM)		00H
	Shift register (SIO)		Undefined
	Serial bus control register (SBIC)		00H

(to be continued)

Table 13-1 Hardware Statuses after Reset (Cont'd)

Hardware		Status after reset
Interrupt	Interrupt request flag registers (IF0H) (IF0L)	00H 00H
	Interrupt mask registers (MK0H) (MK0L)	FFH FFH
	Interrupt priority specification flag registers (PR0H) (PR0L)	FFH FFH
	Interrupt service mode registers (ISM0H) (ISM0L)	00H 00H
	External interrupt mode registers (INTM0)	50H
Standby control register (STBC)		00H
Clock output mode register (CLOM)		00H
ASTB/CLO output		Low-level output
Internal memory size change register (IMS)(*)		FDH

* The IMS is available only with the uPD78P138. The uPD78134A, uPD78136, and uPD78138 do not have it.

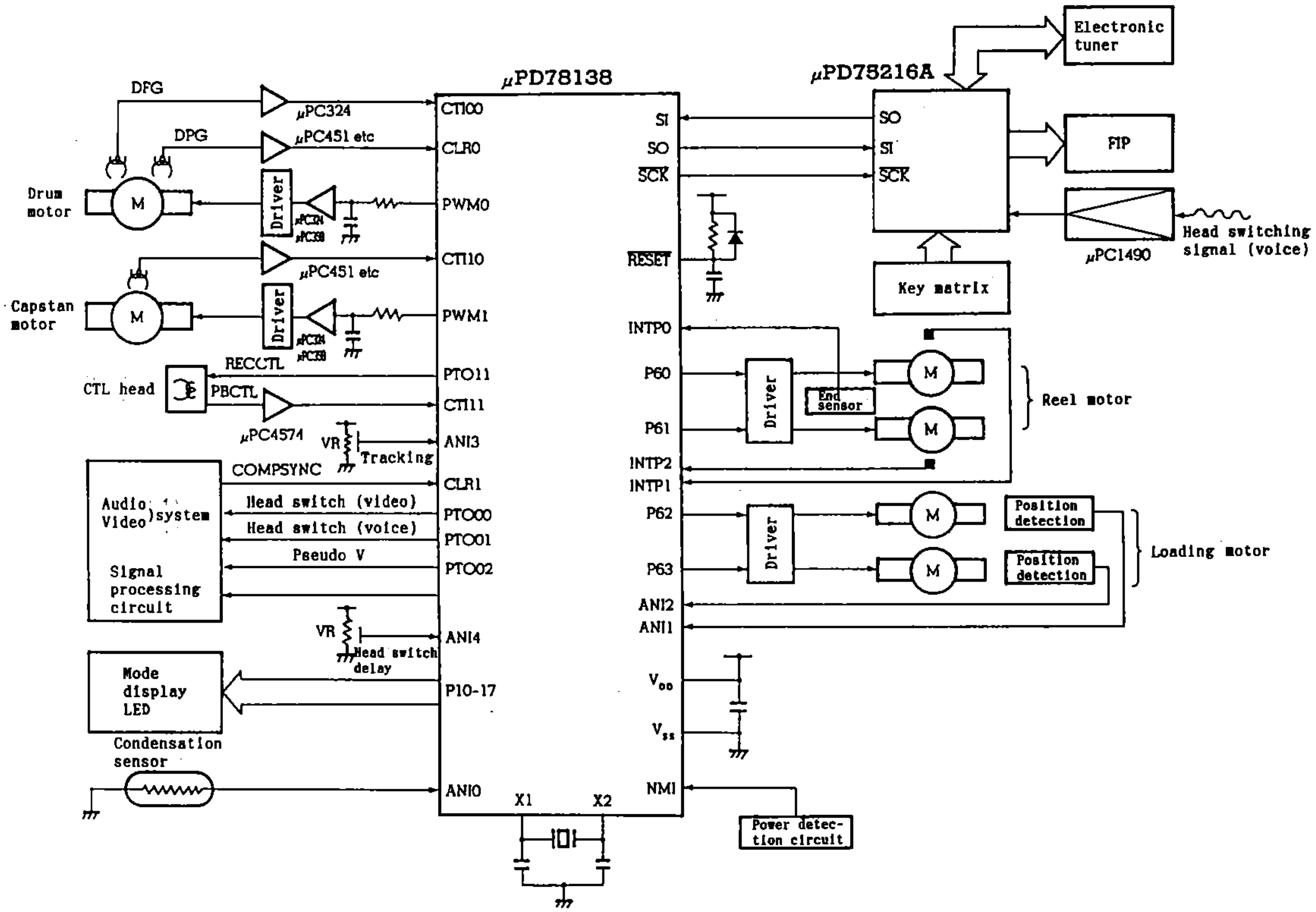
CHAPTER 14 EXAMPLE OF APPLICATION

14.1 Example of Application to the Normal-type Video Cassette Recorder

Figure 14-1 shows a block diagram of a normal-type video cassette recorder.

In this example, the uPD78138 is used for system control and digital servo control, and the uPD75216A is used for controlling a timer, key entry, and FIP display.

Fig. 14-1 Example of Application to the Normal-type Video Cassette Recorder

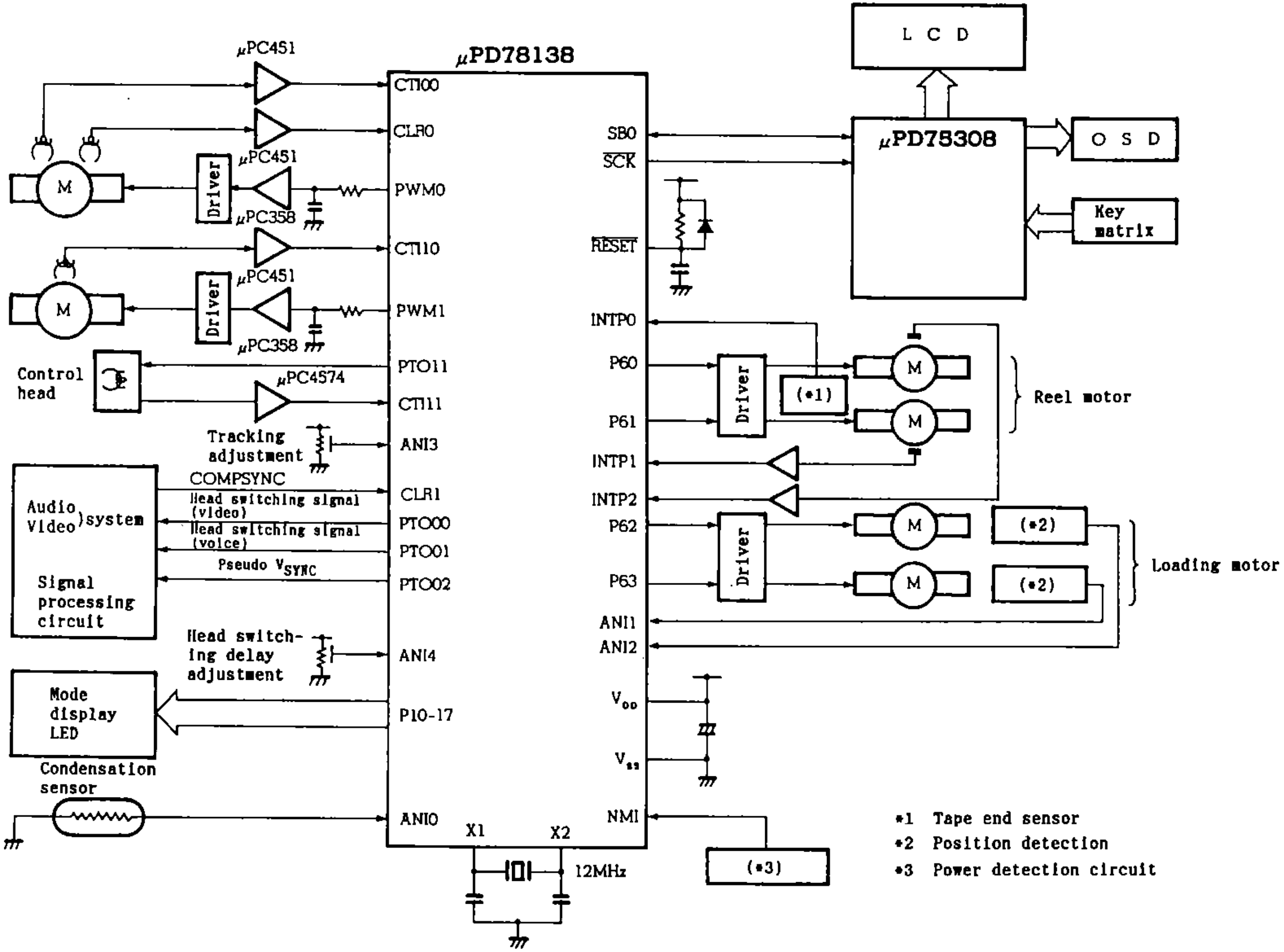


14.2 Example of Application to the Camcoder

Figure 14-2 shows a block diagram of a camcoder.

In this example, the uPD78138 is used for system control and digital servo control, and the uPD75308 is used for controlling LCD display, key entry, and on-screen display (OSD).

Fig. 14-2 Example of Application to the Camcoder



14.3 Using the Super Timer Unit in a VCR Servo System

The example given in this section assumes a VHS-format, normal-type hi-fi VCR.

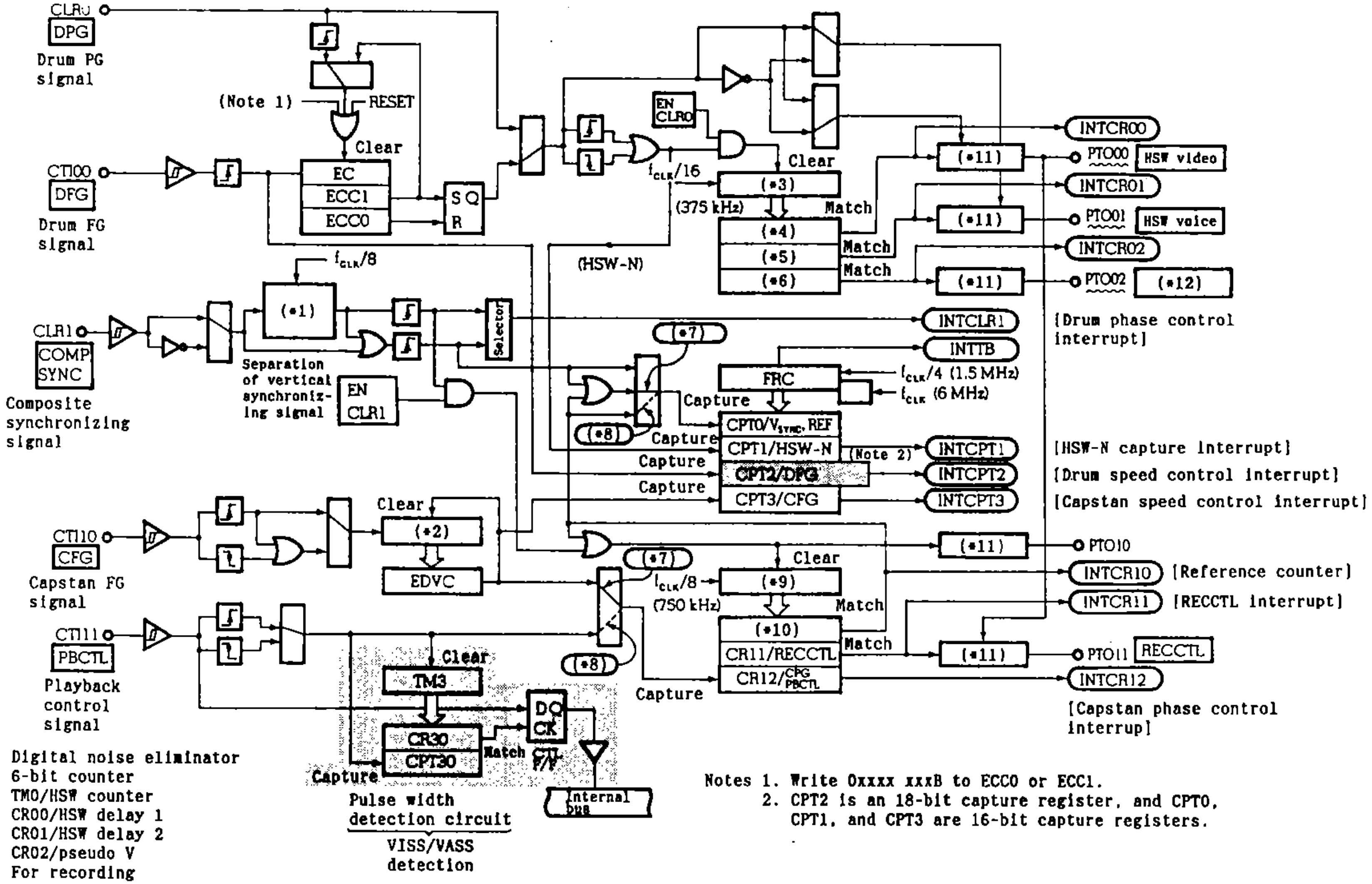
Figure 14-3 gives an example of using the super timer unit in a VCR servo system.

Table 14-1 lists the functions of timers in the VCR servo system.

Table 14-1 Timer Functions in the VCR Servo System

Timer name	Function
Timer 0	. Generating video and voice head switching signals . Generating a pseudo vertical synchronizing signal
Timer 1	. 30-Hz reference timer in playback mode . Generating a recording control signal in recording mode . Capstan phase control . Detecting an index signal (VISS/VASS)
Free running counter	. Drum speed control . Drum phase control . Capstan speed control
Timer 2	. Timer for masking vertical synchronizing signal input interrupts
PWM output unit	. PWM0: For driving the drum motor . PWM1: For driving the capstan motor

Fig. 14-3 Using the Super Timer Unit in the VCR Servo System



*1 Digital noise eliminator
*2 6-bit counter
*3 TM0/HSW counter
*4 CR00/HSW delay 1
*5 CR01/HSW delay 2
*6 CR02/pseudo V
*7 For recording
*8 For playback
*9 TM1/REF counter
*10 CR10/REF. buffer
*11 Output control circuit
*12 Pseudo V_{SYNC}

Notes 1. Write 0xxxx xxxB to ECC0 or ECC1.
2. CPT2 is an 18-bit capture register, and CPT0, CPT1, and CPT3 are 16-bit capture registers.

14.3.1 Controlling the drum motor

The CPT0, CPT1, and CPT2 registers in the free running counter unit and the CR10 register in the timer 1 unit are used for controlling the drum motor.

The DFG signal and DPG signal from the drum motor are input signals. The COMPSYNC signal is the reference signal.

The FRC value is stored in the CPT2 register at the rising edge of the DFG signal from the drum motor.
At the same time, the drum control interrupt (INTCPT2) occurs.

The INTCPT2 interrupt routine calculates the speed of the drum motor by subtracting the previous captured value from the current captured value.

The error is calculated from the speed of the drum motor calculated above and the target value of the speed.

$$
\begin{aligned}
E_{DV} &= (N_{DF(n)} - N_{DF(n-1)}) - N_{DFL} \\
&= \Delta N_{DF} - N_{DFL}
\end{aligned}
$$

E_{DV}:	Error of the drum motor speed
$N_{DF(n)}$:	Current captured value of the CPT2
$N_{DF(n-1)}$:	Previous captured value of the CPT2
N_{DFL}:	Target value (speed when the drum motor stably rotates)
ΔN_{DF}:	Current speed of the drum motor
E_{DP}:	Drum phase error

Drum phase control for playback synchronizes the phases of the head switching signal (mentioned later) and the reference signal.

The reference signal is generated by the reference timer which consists of timer 1 and compare register CR01 and clears itself.

For VCR, the period of the reference signal is the same as the TV broadcast frame frequency.

E_{DP} = (captured value by the internal reference signal) - (captured value by the phase signal) - (target value of phase control)

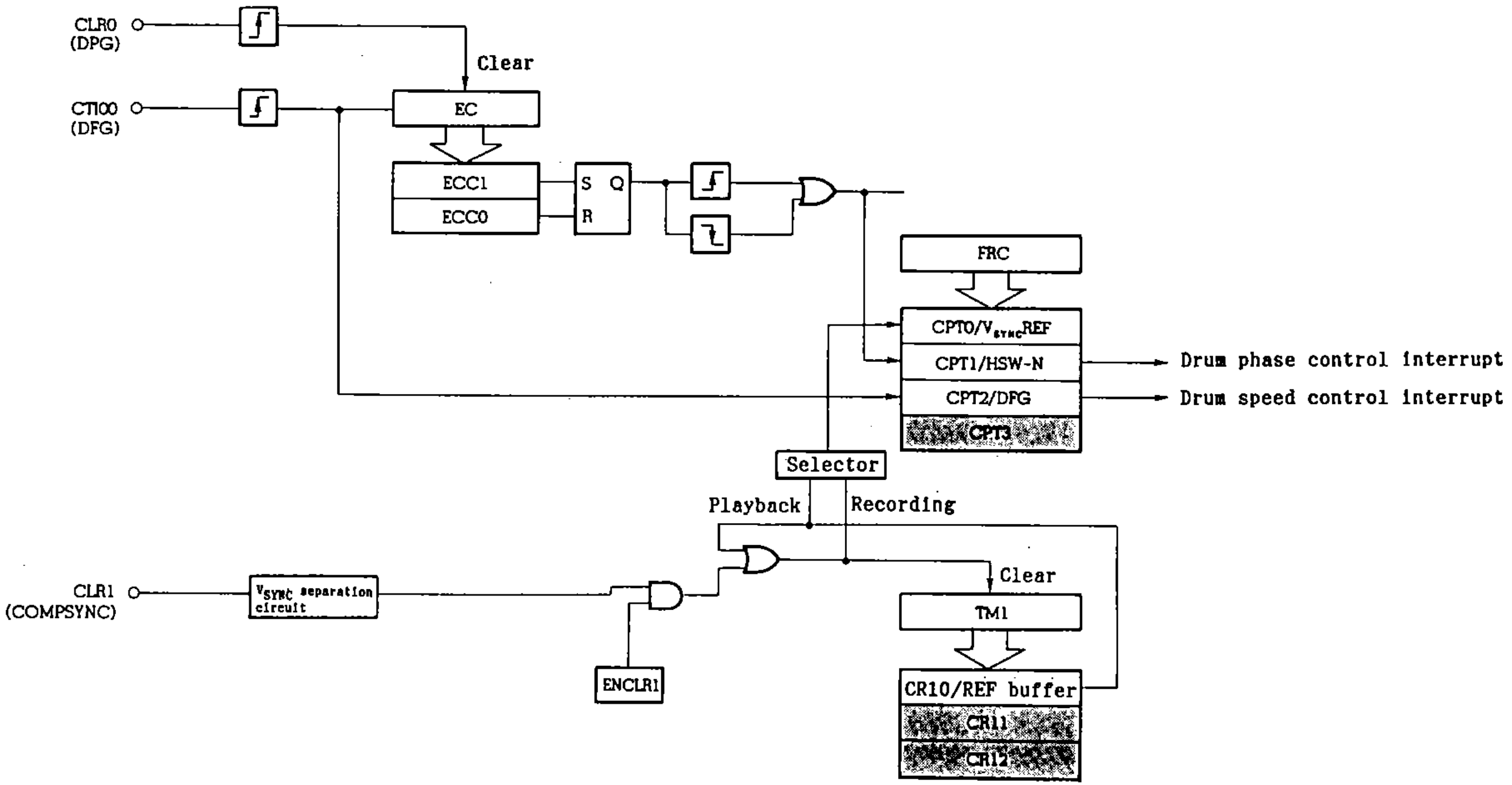
= (CPT0 value) - (CPT1 value) - (target value)

Drum phase control for recording synchronizes the phases of the rotation of the drum speed and the external reference signal (vertical synchronizing signal).

E_{DP} = (captured value by the reference signal) - (captured value by the phase signal) - (target value of phase control)

= (CPT0 value) - (CPT1 value) - (target value)

Fig. 14-4 Controlling the Drum Motor



Remark: ▨ : Registers not used in the control

Fig. 14-5 Drum Speed Control

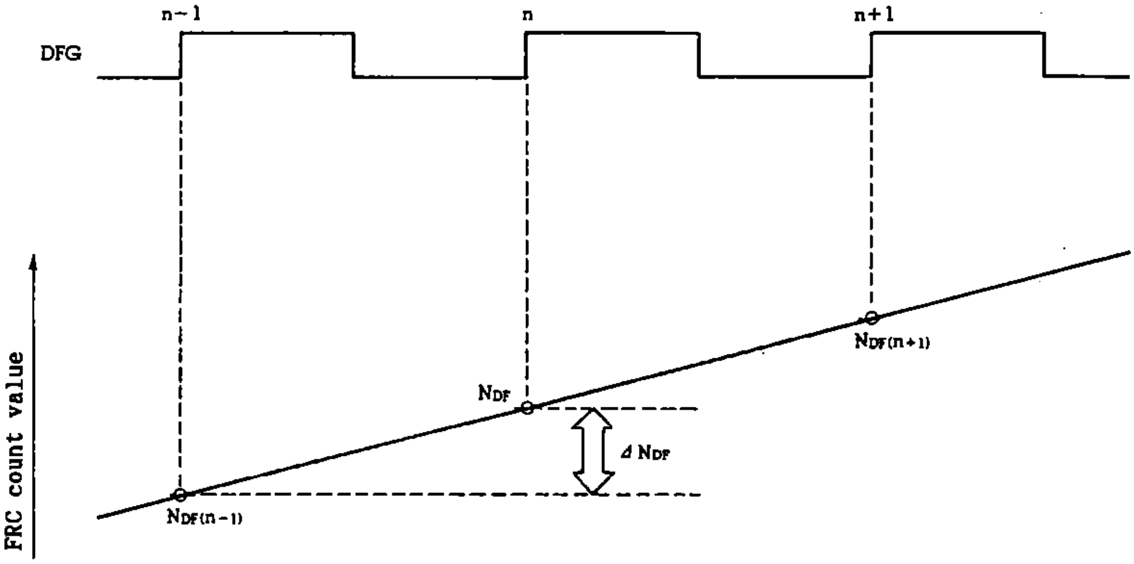
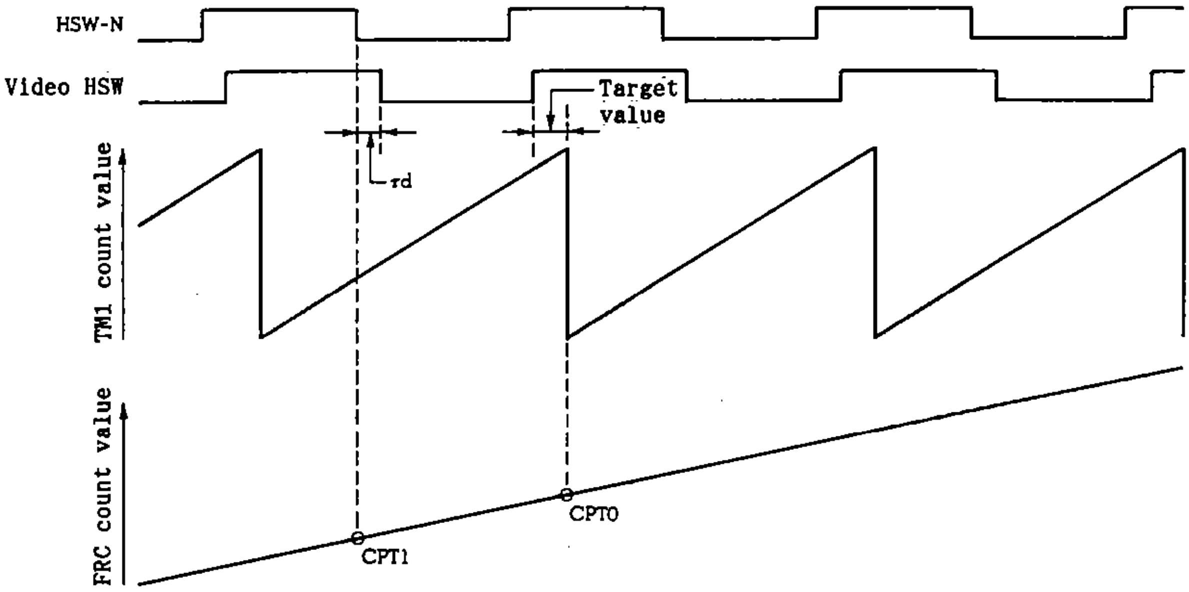
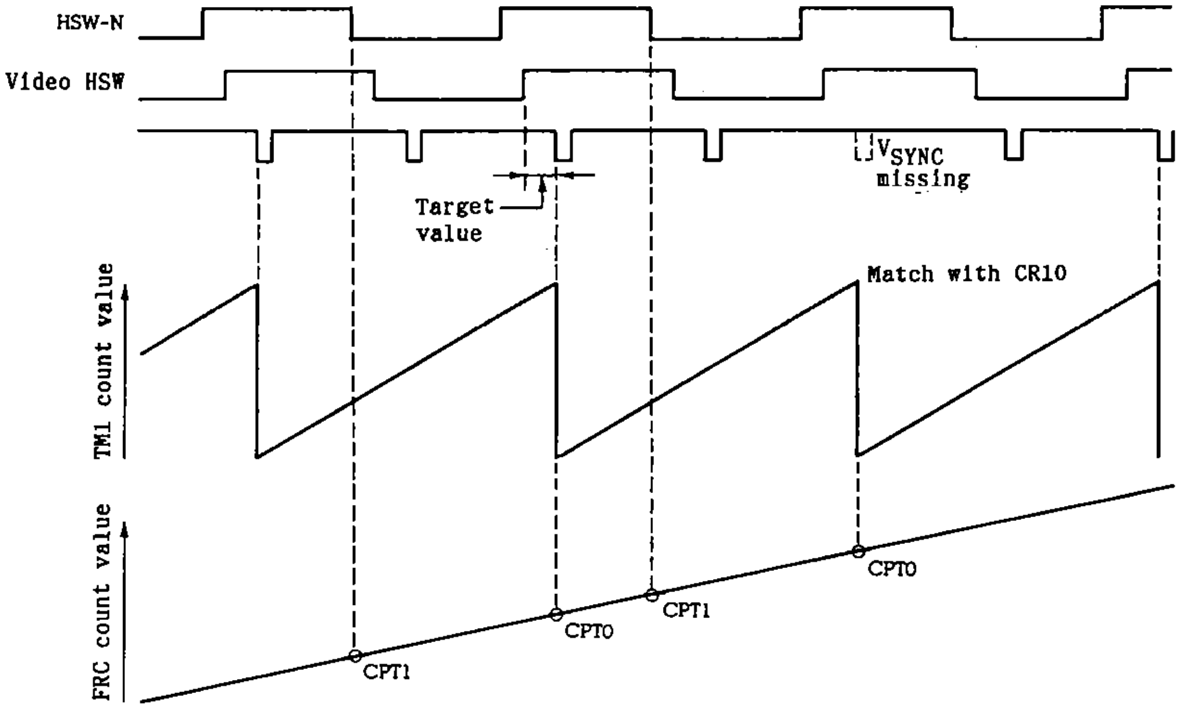


Fig. 14-6 Drum Phase Control

(a) For playback



(b) For recording



14.3.2 Controlling the capstan motor

The CPT3 register in the free running counter unit and the CR10 and CR12 registers in the timer 1 unit are used for controlling the capstan motor.

The CFG signal from the capstan motor and the PBCTL signal (for playback) are input signals. The COMPSYNC signal (for recording) is the reference signal.

The FRC value is stored in the CPT3 register at the rising edge of the CFG signal from the capstan motor.
At the same time, the capstan control interrupt (INTCPT3) occurs.

The INTCPT3 interrupt routine calculates the speed of the capstan motor by subtracting the previous captured value from the current captured value.

The error is calculated from the speed of the capstan motor calculated above and the target value of the speed.

$$\begin{aligned} E_{CV} &= (N_{CF(n)} - N_{CF(n-1)}) - N_{CFL} \\ &= \Delta N_{CF} - N_{CFL} \end{aligned}$$

E_{CV}: Error of the capstan motor speed

$N_{CF(n)}$: Current captured value of the CPT3

$N_{CF(n-1)}$: Previous captured value of the CPT3

N_{CFL}: Target value (speed when the capstan motor stably rotates)

ΔN_{CF}: Current speed of the capstan motor

E_{CP}: Capstan phase error

Capstan phase control for playback synchronizes the phases of the head switching signal and the PBCTL signal.

When the phase of the drum motor is controlled, the phases of the head switching signal and TM1 are already synchronized. The value captured by PBCTL has information on the capstan motor phase.

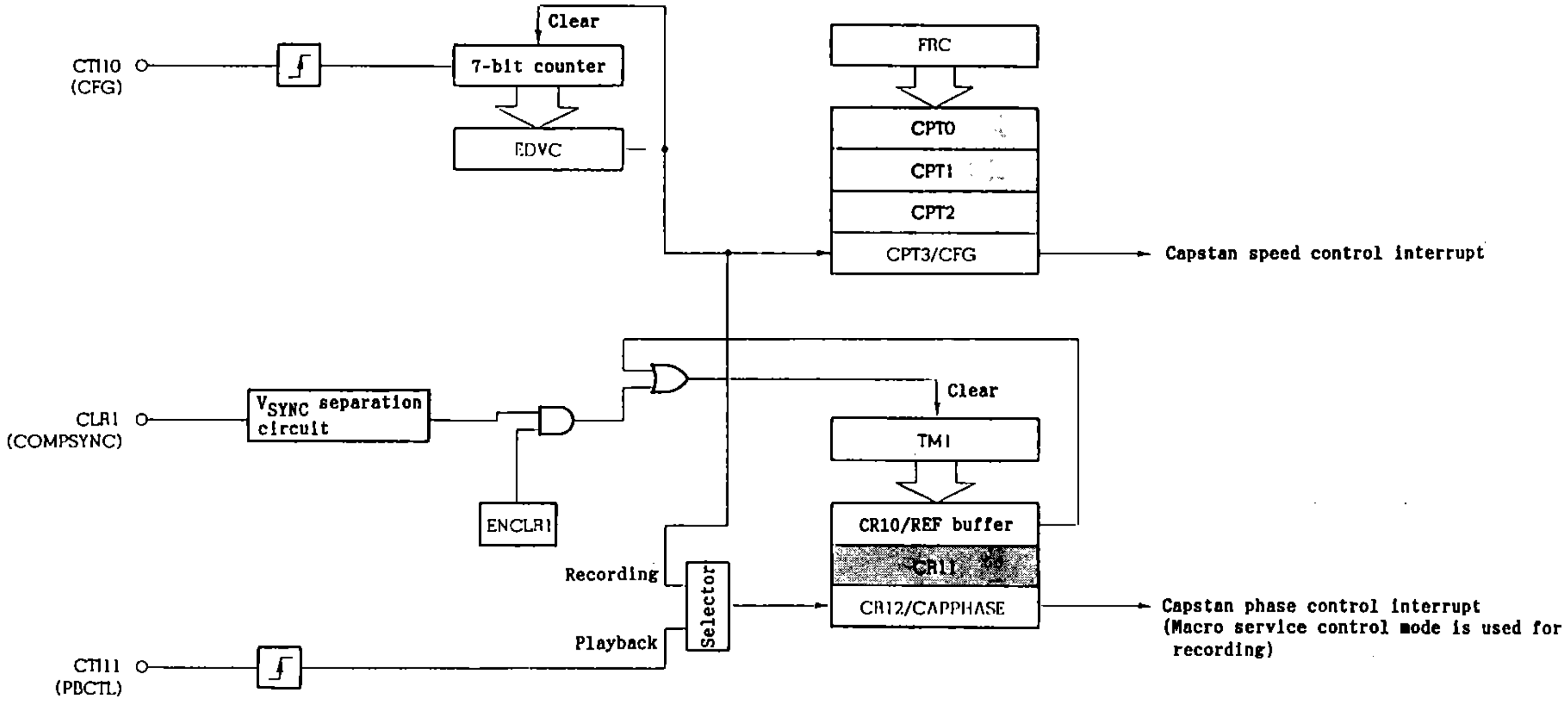
E_{CP} = (captured value by the PBCTL) - (target value of phase control)
= (CP12 value) - (target value)

There is no specified absolute phase referred to as an external signal in capstan phase control for recording.

The target phase can be arbitrarily set.

E_{CP} = (captured value by the CFG divider signal) - (target value of phase control)
= (CP12 value) - (target value)

Fig. 14-7 Controlling the Capstan Motor



Remark: □ : Registers not used in the control

Fig. 14-8 Capstan Speed Control

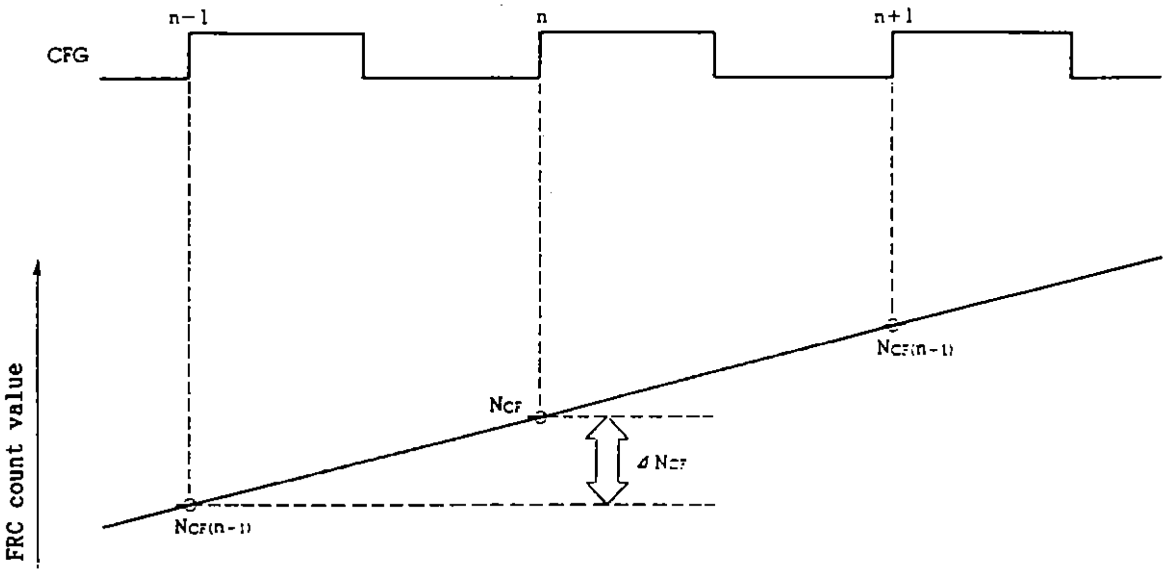
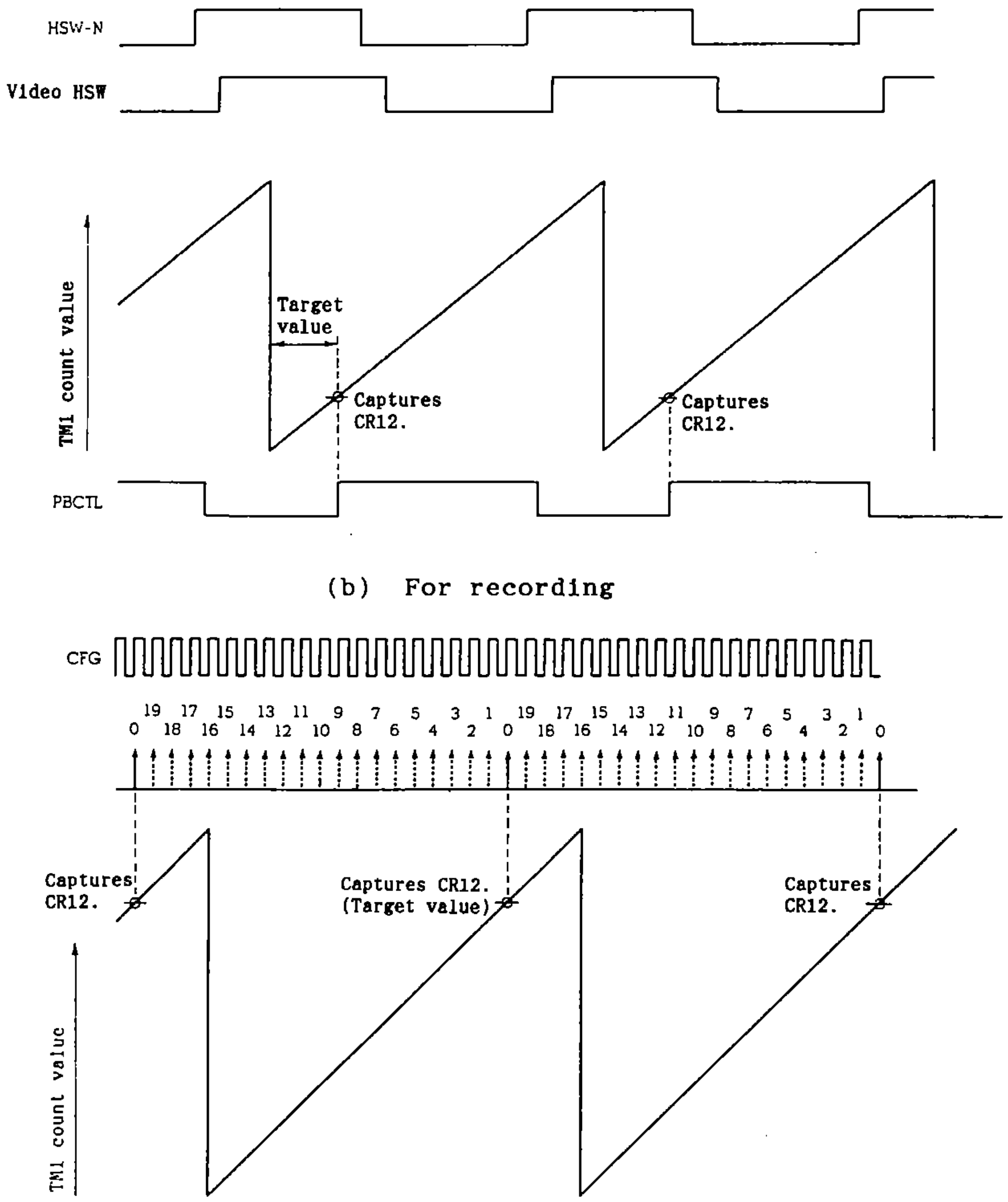


Fig. 14-9 Capstan Phase Control



(a) For playback

(b) For recording

Remark: ⁝ : Macro service

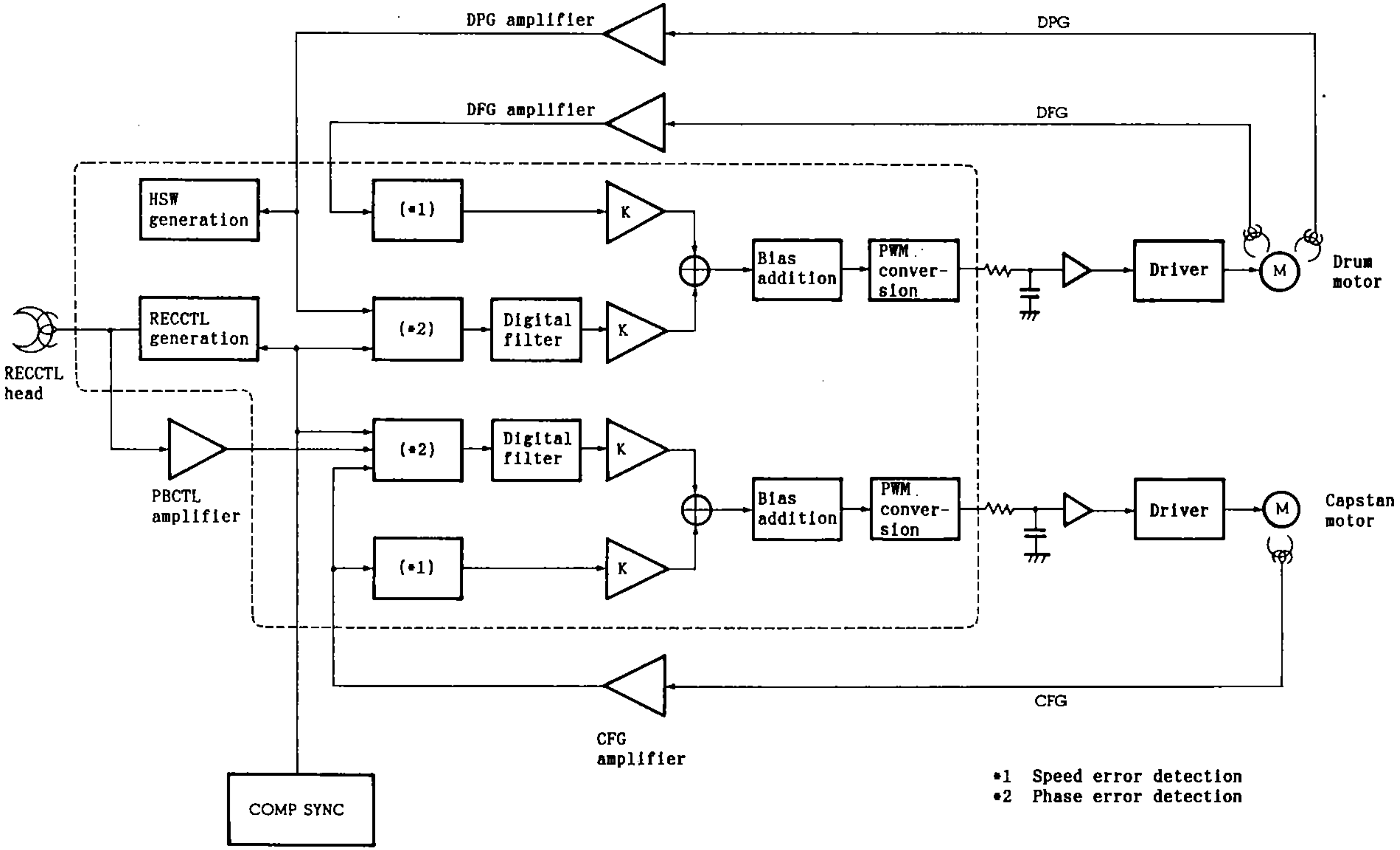
↑ : Vectored interrupt

14.4 Block Diagram of the Software Digital Servo System in a VCR

Figure 14-4 shows a block diagram of the software digital servo system in a VCR.

In the diagram, part enclosed by a dotted line is software processing performed by the uPD78138.

Fig. 14-10 Block Diagram of the VCR Servo System (Processing Enclosed by a Dotted Line is Performed by the uPD78138)



CHAPTER 15 THE uPD78P138

The uPD78P138 is an 8-bit single-chip microcomputer produced by replacing the on-chip mask ROM in the uPD78138 with a PROM.

The uPD78P138 not only makes evaluation and trial manufacture during system development easier, but also enables early stage start-up of applications, and short-run and multiple-device production.

> The EPROM versions of the uPD78P138 are not intended for use in mass-produced products; they do not have reliability high enough for such purposes. Their use should be restricted to functional evaluation in experiment or trial manufacture.

15.1 Differences of uPD78P138 from uPD78134A, uPD78136, and uPD78138

The uPD78P138 is produced by replacing the mask ROM in the uPD78134A, uPD78136, and uPD78138 with a PROM on which data can be written. Table 15-1 shows the differences between these products.

Table 15-1 Differences of uPD78P138 from uPD78134A, uPD78136, and uPD78138

Item	uPD78P138	uPD78134A	uPD78136	uPD78138
Program memory (ROM)	. PROM . 32768 bytes	. Mask ROM . 16384 bytes	. Mask ROM . 24576 bytes	. Mask ROM . 32768 bytes
Data memory (RAM)	640 bytes	384 bytes	640 bytes	
Internal memory size change register (IMS)	Provided (See Figure 15-1.)	Not provided		
Pin connection	In the uPD78P138, the functions to read/write the PROM are added to the pins.			

15.1.1 Internal memory size change register (IMS)

The internal memory size change register (IMS) specifies the effective area of the memory (ROM and RAM) in the uPD78P138.

The IMS is set when the uPD78P138 is used to evaluate a product whose ROM or RAM is smaller than that of the uPD78P138. Using this function eliminates bugs generated by an overflow of the ROM or RAM from application programs. For reference, the sizes of the ROM and RAM for each product are listed below.

Product	On-chip ROM	On-chip RAM
uPD78P138	32768 bytes (PROM)	640 bytes
uPD78138	32768 bytes (mask ROM)	640 bytes
uPD78136	24576 bytes (mask ROM)	640 bytes
uPD78134A	16384 bytes (mask ROM)	384 bytes

Data can be written into the IMS by an 8-bit manipulation instruction, but cannot be read from the IMS.

When the system is reset, the IMS is initialized to FDH, and enters the mode in which the ROM is set at 32768 bytes and the RAM at 640 bytes.

Figure 15-1 shows the format of the IMS.

Fig. 15-1 Format of the Internal Memory Size Change Register (IMS)

	7	6	5	4	3	2	1	0	Address	When reset	R/W
IMS	ROM3	ROM2	ROM1	ROM0	RAM3	RAM2	RAM1	RAM0	FFCFH	FDH	W

RAM3	RAM2	RAM1	RAM0	Internal RAM size change
1	1	0	1	640 bytes
0	1	1	1	384 bytes

Caution: No values other than above shall be set.

ROM3	ROM2	ROM1	ROM0	Internal ROM size change
1	1	1	1	32768 bytes
1	1	1	0	24576 bytes
0	1	0	1	16384 bytes

Caution: No values other than above shall be set.

Cautions 1. Do not use the IMS when the uPD78P138 is used to evaluate the uPD78138, because the internal ROM and RAM of the uPD78138 are of the same size as the uPD78P138.

2. When porting an application program which uses the IMS register of the uPD78P138 to a product having mask ROM, such as the uPD78134A, uPD78136, or uPD78138, be sure to remove the instructions that manipulate the IMS register. This has to be done because products having mask ROM do not contain the IMS register.

15.2 Programming in the uPD78P138

The programmable memory in the uPD78P138 is 32768 x 8 bits of electrically writable PROM. Use pins PROG and $\overline{\text{RESET}}$ to set a PROM programming mode when programming on the PROM.

The uPD78P138 provides programming characteristics compatible with the uPD27C256A.

15.2.1 Operation mode

The uPD78P138 changes to a program write/verify mode when +6 V is applied to pin V_{DD} and +12.5 V to pin V_{PP}. This mode varies to each operation mode shown in Table 15-2 depending on the setting of pins $\overline{\text{CE}}$ and $\overline{\text{OE}}$.

If the read mode is set in the uPD78P138, the contents of PROM can be read.

Table 15-2 Operation Mode when Programming on the PROM

Pin / Mode	PROG	$\overline{\text{RESET}}$	$\overline{\text{CE}}$	$\overline{\text{OE}}$	V_{PP}	V_{DD}	D0-D7
Program write	+12.5 V	L	L	H	+12.5 V	+6 V	Data input
Program verify			H	L			Data output
Program inhibit			H	H			High impedance
Read			L	L	+5 V	+5 V	Data output
Output disable			L	H			High impedance
Standby			H	L/H			High impedance

Caution: Do not set both $\overline{\text{CE}}$ and $\overline{\text{OE}}$ to L when V_{PP} is set to +12.5 V and V_{DD} to +6 V.

15.2.2 Procedure for writing on PROM

The following is a procedure for writing on PROM. Data can be written at high speed.

(1) Always set the $\overline{\text{RESET}}$ pin to low. Apply +12.5 V to the PROG pin. Handle other unused pins as shown in Section 1.6.2.

(2) Apply +6 V to the V_{DD} pin and +12.5 V to the V_{PP} pin.

(3) Set an initial address.

(4) Input write data.

(5) Input a 1 ms program pulse (active low) to the $\overline{\text{CE}}$ pin.

(6) Verify mode: If data is written, go to step (8); otherwise, repeats steps (4) to (6). If no data is written yet after it is repeated 25 times, go to step (7).

(7) Assume the device to be defective and stop write operation.

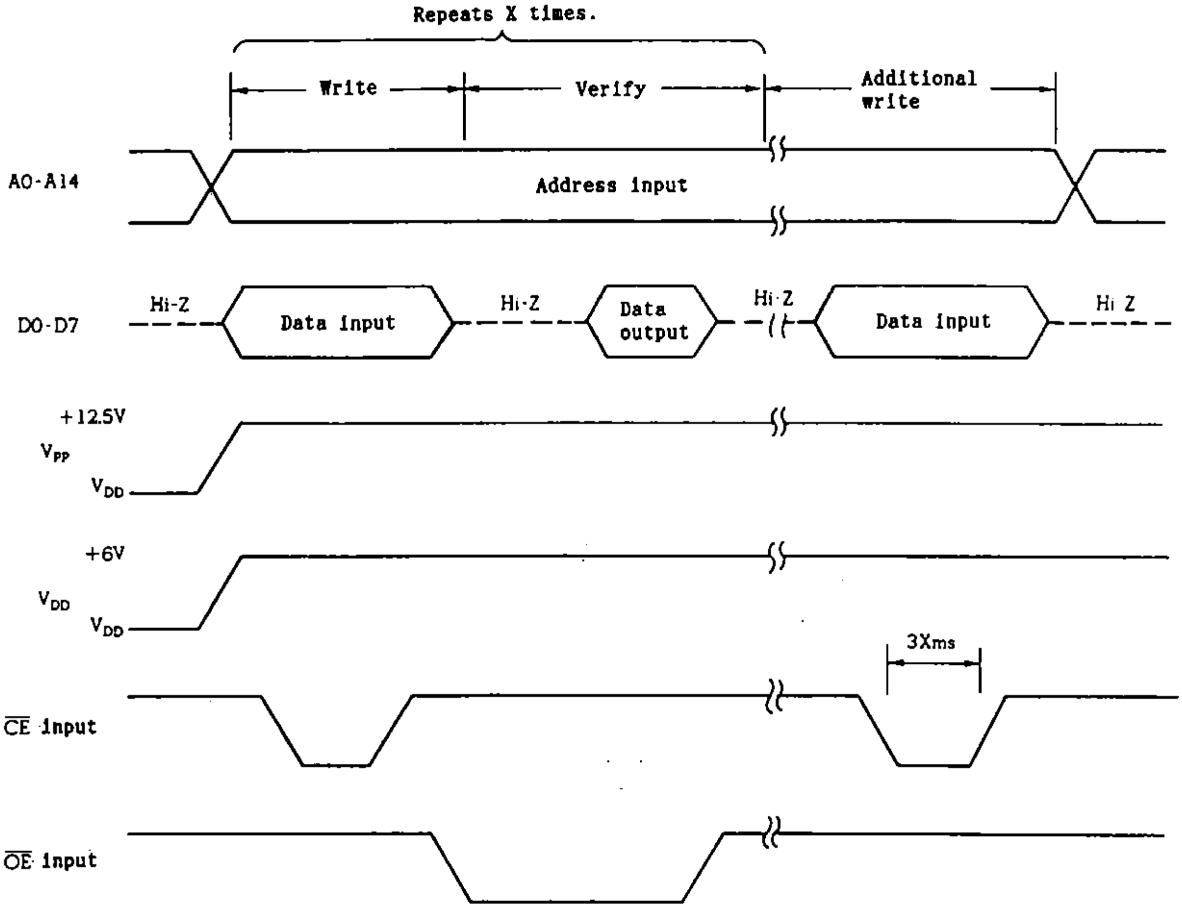
(8) Input write data and a program pulse of number of times steps (4) to (6) were repeated: X) x 3 ms (additional writing).

(9) Increment the address.

(10) Repeat steps (4) to (9) until the address exceeds the last address.

Figure 15-2 is a timing chart of these steps (2) to (8).

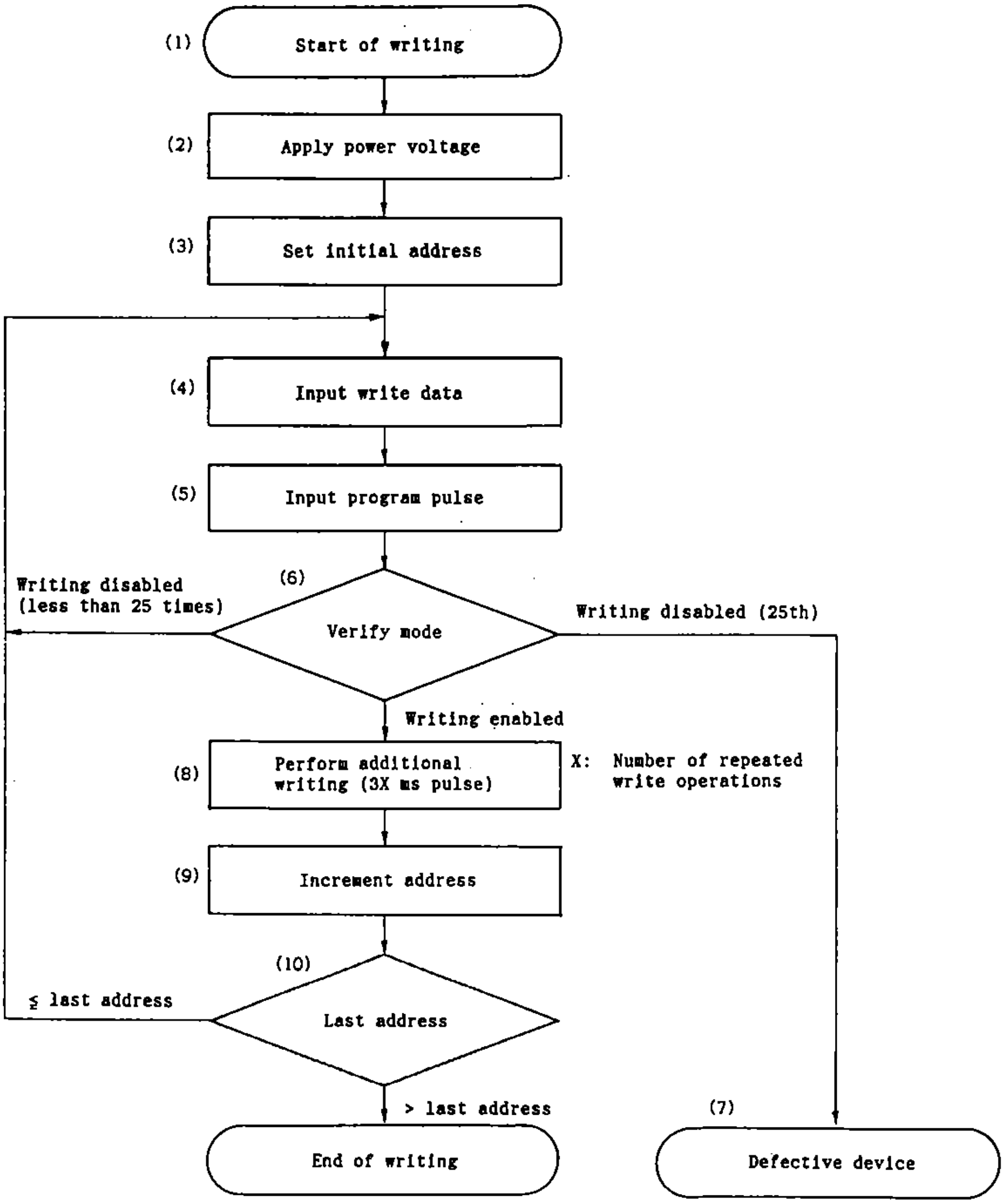
Fig. 15-2 PROM Write/Verify Timing Chart



Cautions 1. Apply V_{DD} before V_{PP} and diconnect V_{DD} after V_{PP}.

2. Set V_{PP} including overshoot so that it is less than +13 V.

Fig. 15-3 Flowchart of Writing Procedure



15.2.3 Procedure for reading from PROM

The contents of PROM can be read out to the external data bus (D0 to D7) in the following steps:

(1) Always set the $\overline{\text{RESET}}$ pin to low. Apply +12.5 V to the PROG pin. Handle other unused pins as shown in Section 1.6.2.

(2) Apply +5 V to the V_{DD} and V_{PP} pins.

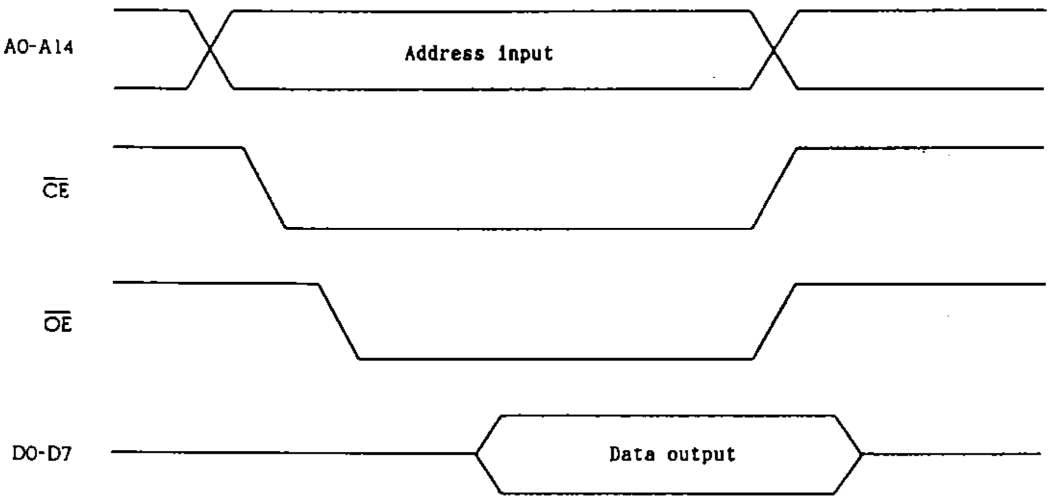
(3) Input the address of data to be read into the A0 to A14 pins.

(4) Read mode

(5) Output the data on the D0 to D7 pins.

Figure 15-4 is a timing chart of these steps (2) to (5).

Fig. 15-4 PROM Read Timing Chart



15.3 Erasure Characteristics Only for the uPD78P138K

The programmed data of the uPD78P138K can be erased by exposure to light with a wavelength less than approx. 400 nm (all of the EPROM data are set to FFH).

To erase the contents of program memory in the uPD78P138K, expose the erasure window to ultraviolet light with the wavelength of 254 nm. The amount of light required to completely erase the contents of program memory is a minimum of 15 W·s/cm^2 (intensity of ultraviolet light x erasing time). It takes about 15 to 20 minutes to expose the erasure window to a 12000-uW/cm^2 ultraviolet lamp. It may, however, take more time due to the fallen performance of this ultraviolet lamp, dirt on the package window, or suchlike. Note that the uPD78P138K should be placed less than 2.5 cm from the ultraviolet lamp during erasure. In addition, if a filter is attached to the ultraviolet lamp, remove the filter before erasure.

15.4 Protective Film Covering the Erasure Window Only for the uPD78P138K

The erasure window must be covered with a protective film when not erasing the contents of EPROM. This is to prevent the contents of memory from being erased erroneously by exposure to light other than the EPROM-contents erasing lamp. This is also to prevent any malfunction of the internal circuits other than the EPROM due to the light.

CHAPTER 16 INSTRUCTION SET

16.1 Operations

16.1.1 Legend

(1) Operand notation and coding format

Operands are coded in the operand field of each instruction as listed in the coding column of the table below. For details of the operand format, refer to the relevant assembler specifications. When several coding forms are presented, any one of them is selected. Uppercase letters and the symbols, +, #, !, $, /, and [], are keywords and must be written as they are. These symbols have the following meanings.

+:	Auto increment
#:	Immediate data
!:	Address by immediate addressing
$:	Address by relative addressing
/:	Bit inversion
[]:	Indirect addressing

For immediate data, an appropriate numeric or label must be written. The symbols +, #, !, $, /, and [] must not be omitted when describing labels.

Notation	Coding
r,r'	X(R0), A(R1), C(R2), B(R3), E(R4), D(R5), L(R6), H(R7)
r1	A, B
r2	B, C
r3	D, E, E+
r4	D, E
rp,rp'	AX(RR0), BC(RP1), DE(RP2), HL(RP3)
sfr	Special function register abbreviation
sfrp	Special function register abbreviation (16-bit manipulation register)
saddr	FE20H-FF1FH Immediate data or label
saddrp	FE20H-FF1EH Immediate data (bit 0 = 0, however) or label (for 16-bit manipulation)
!addr16	0000H-FFFFH Immediate data or label: Immediate addressing
$addr16	0000H-FFFFH Immediate data or label: Relative addressing
addr11	800H-FFFH Immediate data or label
addr5	40H-7EH Immediate data (bit 0 = 0, however) or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data (0 to 7)
RBn	RB0-RB3

Remarks 1. Absolute names (R0 to R7 and RP0 to RP3) can be specified in r, r', rp, and rp', as well as functional names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL).

2. Immediate addressing is effective for entire address spaces. Relative addressing is effective for the locations within a displacement range of -128 to +127 from the starting address of the next instruction.

(2) Legend

A:	Register A; 8-bit accumulator
X:	Register X
B:	Register B
C:	Register C
D:	Register D
E:	Register E
H:	Register H
L:	Register L
R0-R7	Register 0 to register 7 (absolute name)
AX:	Register pair (AX); 16 bit accumulator
BC:	Register pair (BC)
DE:	Register pair (DE)
HL:	Register pair (HL)
RP0-RP3:	Register pair 0 to register pair 3 (absolute name)
PC:	Program counter
SP:	Stack pointer
PSW:	Program status word
CY:	Carry flag
AC:	Auxiliary carry flag
Z:	Zero flag
RBS0-RBS1:	Register bank select flag
IE:	Interrupt request enable flag
STBC:	Standby control register
jdisp8:	8-bit signed data (displacement: -128 to +127)
( ):	Contents at an address enclosed in parentheses or at an address indicated in a register enclosed in parentheses
xxH:	Hexadecimal number
x_H, x_L:	Eight high-order bits and eight low-order bits of 16-bit register pair

(3) Numeric symbols in clock field

One clock cycle of an instruction is equivalent to a clock cycle of the internal system clock; $1/f_{CLK}$. (See Chapter 4.)

The number of clocks varies according to whether the instruction is located in the internal ROM or in the external memory, or according to the memory area to be accessed. See Section 16.3 for details.

The digit in the clock field is the number of clocks when the instruction is fetched from the internal ROM.

(4) Notational symbols in flag operation field

Symbol	Explanation
(Blank)	No change
0	Cleared to zero.
1	Set to 1.
x	Set or reset according to the result.
R	Saved values are restored.

16.1.2 List of operations

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit data transfer	MOV	r,#byte	2	2	r ← byte			
		saddr,#byte	3	3/5	(saddr) ← byte			
		sfr,#byte(*1)	3	5	sfr ← byte			
		r,r'	2	2	r ← r'			
		A,r	1	2	A ← r			
		A,saddr	2	2/4	A ← (saddr)			
		saddr,A	2	3/5	(saddr) ← A			
		A,sfr	2	4	A ← sfr			
		sfr,A	2	5	sfr ← A			
		A,[r3](*2)	1	5	A ← (FE00H+r3) r3=00H-FFH			
		[r3],A(*2)	1	5	(FE00H+r3) ← A r3=00H-FFH			

(to be continued)

*1 If STBC is written in sfr, a different dedicated instruction having the different byte and clock counts is generated. (See CPU control instructions.)

*2 If E+ is written in r3, the contents of register E are incremented by 1 after instruction execution, and the number of clocks is 6.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit data transfer	MOV	A,[HL]	1	5-7	A ← (HL)			
		[HL],A	1	5/7	(HL) ← A			
		A,[HL+]	1	8-10	A ← (HL), HL ← HL+1			
		[HL+],A	1	8/10	(HL) ← A, HL ← HL+1			
		A,[DE]	1	5-7	A ← (DE)			
		[DE],A	1	5/7	(DE) ← A			
		A,[DE+]	1	8-10	A ← (DE), DE ← DE+1			
		[DE+],A	1	8/10	(DE) ← A, DE ← DE+1			
		A,!addr16	4	6-8	A ← !addr16			
		!addr16,A	4	6/8	!addr16 ← A			
		A,word[r1]	4	7-9	A ← (word+r1)			
		word[r1],A	4	7/9	(word+r1) ← A			
		PSW,#byte	3	5	PSW ← byte	x	x	x
		PSW,A	2	5	PSW ← A	x	x	x
		A,PSW	2	4	A ← PSW			
	XCH	A,r	1	4	A ←→ r			
		A,saddr	2	4/8	A ←→ (saddr)			
		A,sfr	3	10	A ←→ sfr			
		A,[r4]	1	9	A ←→ (FE00H+r4) r4=00H-FFH			
		A,[HL]	2	9/13	A ←→ (HL)			
		A,[DE]	2	9/13	A ←→ (DE)			
		A,word[r1]	4	9/13	A ←→ (word+r1)			

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
16-bit data transfer	MOVW	rp,#word	3	3	rp ← word			
		saddrp,#word	4	4/8	(saddrp) ← word			
		sfrp,#word	4	8	sfrp ← word			
		rp,rp'	2	4	rp ← rp'			
		AX,saddrp	2	6/10	AX ← (saddrp)			
		saddrp,AX	2	5/9	(saddrp) ← AX			
		AX,sfrp	2	10	AX ← sfrp			
		sfrp,AX	2	9	sfrp ← AX			
8-bit arithmetic/ logical	ADD	A,#byte	2	2	A,CY ← A+byte	x	x	x
		saddr,#byte	3	4/8	(saddr),CY ← (saddr)+byte	x	x	x
		sfr,#byte	4	10	sfr,CY ← sfr+ byte	x	x	x
		r,r'	2	3	r,CY ← r+r'	x	x	x
		A,saddr	2	3/5	A,CY ← A+(saddr)	x	x	x
		A,sfr	3	7	A,CY ← A+sfr	x	x	x
		A,[r4]	2	7	A,CY ← A+ (FE00H+r4) r4=00H-FFH	x	x	x
		A,[HL]	2	8-10	A,CY ← A+(HL)	x	x	x
		A,[DE]	2	8-10	A,CY ← A+(DE)	x	x	x
		A,word[r1]	4	8-10	A,CY ← A+ (word+r1)	x	x	x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit arithmetic-logical	ADDC	A,#byte	2	2	A,CY ← A+byte+CY	x	x	x
		saddr,#byte	3	4/8	(saddr),CY ← (saddr)+byte+CY	x	x	x
		sfr,#byte	4	10	sfr,CY ← sfr+ byte+CY	x	x	x
		r,r'	2	3	r,CY ← r+r'+CY	x	x	x
		A,saddr	2	3/5	A,CY ← A+(saddr) +CY	x	x	x
		A,sfr	3	7	A,CY ← A+sfr+CY	x	x	x
		A,[r4]	2	7	A,CY ← A+(FE00H+ r4)+CY r4=00H-FFH	x	x	x
		A,[HL]	2	8-10	A,CY ← A+(HL)+CY	x	x	x
		A,[DE]	2	8-10	A,CY ← A+(DE)+CY	x	x	x
		A,word[r1]	4	8-10	A,CY ← A+ (word+r1)+CY	x	x	x
	SUB	A,#byte	2	2	A,CY ← A+byte	x	x	x
		saddr,#byte	3	4/8	(saddr),CY ← (saddr)+byte	x	x	x
		sfr,#byte	4	10	sfr,CY ←sfr+byte	x	x	x
		r,r'	2	3	r,CY ← r+r'	x	x	x
		A,saddr	2	3/5	A,CY ← A-(saddr)	x	x	x
		A,sfr	3	7	A,CY ← A-sfr	x	x	x
		A,[r4]	2	7	A,CY ← A-(FE00H+ r4) r4=00H-FFH	x	x	x
		A,[HL]	2	8-10	A,CY ← A-(HL)	x	x	x
		A,[DE]	2	8-10	A,CY ← A-(DE)	x	x	x
		A,word[r1]	4	8-10	A,CY ← A- (word+r1)	x	x	x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit arithmetic/ logical	SUBC	A,#byte	2	2	A,CY ← A-byte-CY	x	x	x
		saddr,#byte	3	4/8	(saddr),CY ← (saddr)-byte-CY	x	x	x
		sfr,#byte	4	10	sfr,CY ← sfr-byte-CY	x	x	x
		r,r'	2	3	r,CY ← r-r'-CY	x	x	x
		A,saddr	2	3/5	A,CY ← A-(saddr)-CY	x	x	x
		A,sfr	3	7	A,CY ← A-sfr-CY	x	x	x
		A,[r4]	2	7	A,CY ← A-(FE00H+r4)-CY r4=00H-FFH	x	x	x
		A,[HL]	2	8-10	A,CY ← A-(HL)-CY	x	x	x
		A,[DE]	2	8-10	A,CY ← A-(DE)-CY	x	x	x
		A,word[r1]	4	8-10	A,CY ← A-(word+r1)-CY	x	x	x
	AND	A,#byte	2	2	A ← A ∧ byte	x		
		saddr,#byte	3	4/8	(saddr) ← (saddr) ∧ byte	x		
		sfr,#byte	4	10	sfr sfr ∧ byte	x		
		r,r'	2	3	r ← r ∧ r'	x		
		A,saddr	2	3/5	A ← A ∧ (saddr)	x		
		A,sfr	3	7	A ← A ∧ sfr	x		
		A,[r4]	2	7	A ← A ∧ (FE00H+r4) r4=00H-FFH	x		
		A,[HL]	2	8-10	A ← A ∧ (HL)	x		
		A,[DE]	2	8-10	A ← A ∧ (DE)	x		
		A,word[r1]	4	8-10	A ← A ∧ (word+r1)	x		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit arithmetic/ logical	OR	A,#byte	2	2	A ← A ∨ byte	x		
		saddr,#byte	3	4/8	(saddr) ← (saddr) ∨ byte	x		
		sfr,#byte	4	10	sfr ← sfr ∨ byte	x		
		r,r'	2	3	r ← r ∨ r'	x		
		A,saddr	2	3/5	A ← A ∨ (saddr)	x		
		A,sfr	3	7	A ← A ∨ sfr	x		
		A,[r4]	2	7	A ← A ∨ (FE00H+ r4) r4=00H-FFH	x		
		A,[HL]	2	8-10	A ← A ∨ (HL)	x		
		A,[DE]	2	8-10	A ← A ∨ (DE)	x		
		A,word[r1]	4	8-10	A ← A ∨ (word+r1)	x		
	XOR	A,#byte	2	2	A ← A ∀ byte	x		
		saddr,#byte	3	4/8	(saddr) ← (saddr) ∀ byte	x		
		sfr,#byte	4	10	sfr ← sfr ∀ byte	x		
		r,r'	2	3	r ← r ∀ r'	x		
		A,saddr	2	3/5	A ← A ∀ (saddr)	x		
		A,sfr	3	7	A ← A ∀ sfr	x		
		A,[r4]	2	7	A ← A ∀ (FE00H+ r4) r4=00H-FFH	x		
		A,[HL]	2	8-10	A ← A ∀ (HL)	x		
		A,[DE]	2	8-10	A ← A ∀ (DE)	x		
		A,word[r1]	4	8-10	A ← A ∀ (word+r1)	x		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
8-bit arithmetic/ logical	CMP	A,#byte	2	2	A-byte	x	x	x
		saddr,#byte	3	3/5	(saddr)-byte	x	x	x
		sfr,#byte	4	7	sfr-byte	x	x	x
		r,r'	2	3	r-r'	x	x	x
		A,saddr	2	3/5	A-(saddr)	x	x	x
		A,sfr	3	7	A-sfr	x	x	x
		A,[r4]	2	7	A-(FE00H+r4) r4=00H-FFH	x	x	x
		A,[HL]	2	8-10	A-(HL)	x	x	x
		A,[DE]	2	8-10	A-(DE)	x	x	x
		A,word[r1]	4	8-10	A-(word+r1)	x	x	x
16-bit arithmetic/ logical	ADDW	AX,#word	3	4	AX,CY ← AX+word	x	x	x
		AX,rp	2	6	AX,CY ← AX+rp	x	x	x
		AX,saddrp	2	7/11	AX,CY ← AX+ (saddrp+1) (saddrp)	x	x	x
		AX,sfrp	3	13	AX,CY ← AX+sfrp	x	x	x
	SUBW	AX,#word	3	4	AX,CY ← AX-word	x	x	x
		AX,rp	2	6	AX,CY ← AX-rp	x	x	x
		AX,saddrp	2	7/11	AX,CY ← AX- (saddrp+1) (saddrp)	x	x	x
		AX,sfrp	3	13	AX,CY ← AX-sfrp	x	x	x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
16-bit arithmetic/ logical	CMPW	AX,#word	3	3	AX-word	x	x	x
		AX,rp	2	5	AX-rp	x	x	x
		AX,saddrp	2	6/10	AX-(saddrp)	x	x	x
		AX,sfrp	3	12	AX-sfrp	x	x	x
Multiply/ divide	MULSW	r$^{(*)}$	2	47	AX(16 high-order bits), r1(8 low-order bits) ← AX(signed 16 bits) x r1(absolute 8 bits)			
	MULUW	r$^{(*)}$	2	47	AX(16 high-order bits), r(8 low-order bits) ← AX x r			
	DIVUW	r$^{(*)}$	2	74	AX(quotient), r(remainder) ← AX ÷ r			
Increment/ decrement	INC	r	1	2	r ← r+1	x	x	
		saddr	2	3/7	(saddr) ← (saddr)+1	x	x	
	DEC	r	1	2	r ← r-1	x	x	
		saddr	2	3/7	(saddr) ← (saddr)-1	x	x	
	INCW	rp	1	3	rp ← rp+1			
	DECW	rp	1	3	rp ← rp-1			

(to be continued)

* Except for the registers A and X.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Shift/ rotate	ROR	r,n	2	3+2n	$(CY,r_7 \leftarrow r_0, r_{m-1} \leftarrow r_m)$ x n n=0-7			x
	ROL	r,n	2	3+2n	$(CY,r_0 \leftarrow r_7, r_{m+1} \leftarrow r_m)$ x n n=0-7			x
	RORC	r,n	2	3+2n	$(CY \leftarrow r_0, r_7+CY, r_{m-1} \leftarrow r_m)$ x n n=0-7			x
	ROLC	r,n	2	3+2n	$(CY \leftarrow r_7, r_0 \leftarrow CY, r_{m+1} \leftarrow r_m)$ x n n=0-7			x
	SHR	r,n	2	3+2n	$(CY \leftarrow r_0, r_7 \leftarrow 0, r_{m-1} \leftarrow r_m)$ x n n=0-7	x	0	x
	SHL	r,n	2	3+2n	$(CY \leftarrow r_7, r_0 \leftarrow 0, r_{m+1} \leftarrow r_m)$ x n n=0-7	x	0	x
	SHRW	rp,n	2	3+3n	$(CY \leftarrow rp_0, rp_{15} \leftarrow 0, rp_{m-1} \leftarrow rp_m)$ x n n=0-7	x	0	x
	SHLW	rp,n	2	3+3n	$(CY \leftarrow rp_{15}, rp_0 \leftarrow 0, rp_{m+1} \leftarrow rp_m)$ x n n=0-7	x	0	x
	ROR4	[r4]	2	22	$A_{3-0} \leftarrow (FE00+r4)_{3-0}, (FE00+r4)_{7-4} \leftarrow A_{3-0}, (FE00+r4)_{3-0} \leftarrow (FE00+r4)_{7-4}$			
	ROL4	[r4]	2	23	$A_{3-0} \leftarrow (FE00+r4)_{7-4}, (FE00+r4)_{3-0} \leftarrow A_{3-0}, (FE00+r4)_{7-4} \leftarrow (FE00+r4)_{3-0}$			

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
BCD correction	ADJBA		1	3	Decimal Adjust Accumulator after Addition	x	x	x
	ADJBS		1	3	Decimal Adjust Accumulator after Subtract	x	x	x
Bit manipulation	MOV1	CY,saddr.bit	3	5/7	CY ← (saddr.bit)			x
		CY,sfr.bit	3	7	CY ← sfr.bit			x
		CY,A.bit	2	5	CY ← A.bit			x
		CY,X.bit	2	5	CY ← X.bit			x
		CY,PSW.bit	2	5	CY ← PSW.bit			x
		saddr.bit,CY	3	8/12	(saddr.bit) ← CY			
		sfr.bit,CY	3	12	sfr.bit ← CY			
		A.bit,CY	2	8	A.bit ← CY			
		X.bit,CY	2	8	X.bit ← CY			
		PSW.bit,CY	2	7	PSW.bit ← CY	x	x	
	AND1	CY,saddr.bit	3	5/7	CY ← CY∧ (saddr.bit)			x
		CY,/saddr.bit	3	5/7	CY ← CY∧ ($\overline{\text{saddr.bit}}$)			x
		CY,sfr.bit	3	7	CY ← CY∧sfr.bit			x
		CY,/sfr.bit	3	7	CY ← CY∧$\overline{\text{sfr.bit}}$			x
		CY,A.bit	2	5	CY ← CY∧A.bit			x
		CY,/A.bit	2	5	CY ← CY∧$\overline{\text{A.bit}}$			x
		CY,X.bit	2	5	CY ← CY∧X.bit			x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Bit manipulation	AND1	CY,/X.bit	2	5	CY ← CY ∧ $\overline{\text{X.bit}}$			x
		CY,PSW.bit	2	5	CY ← CY ∧ PSW.bit			x
		CY,/PSW.bit	2	5	CY ← CY ∧ $\overline{\text{PSW.bit}}$			x
	OR1	CY,saddr.bit	3	5/7	CY ← CY ∨ (saddr.bit)			x
		CY,/saddr.bit	3	5/7	CY ← CY ∨ ($\overline{\text{saddr.bit}}$)			x
		CY,sfr.bit	3	7	CY ← CY ∨ sfr.bit			x
		CY,/sfr.bit	3	7	CY ← CY ∨ $\overline{\text{sfr.bit}}$			x
		CY,A.bit	2	5	CY ← CY ∨ A.bit			x
		CY,/A.bit	2	5	CY ← CY ∨ $\overline{\text{A.bit}}$			x
		CY,X.bit	2	5	CY ← CY ∨ X.bit			x
		CY,/X.bit	2	5	CY ← CY ∨ $\overline{\text{X.bit}}$			x
		CY,PSW.bit	2	5	CY ← CY ∨ PSW.bit			x
		CY,/PSW.bit	2	5	CY ← CY ∨ $\overline{\text{PSW.bit}}$			x
	XOR1	CY,saddr.bit	3	5/7	CY ← CY $\veebar$ (saddr.bit)			x
		CY,sfr.bit	3	7	CY ← CY $\veebar$ sfr.bit			x
		CY,A.bit	2	5	CY ← CY $\veebar$ A.bit			x
		CY,X.bit	2	5	CY ← CY $\veebar$ X.bit			x
		CY,PSW.bit	2	5	CY ← CY $\veebar$ PSW.bit			x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Bit manipulation	SET1	saddr.bit	2	3/7	(saddr.bit) ← 1			
		sfr.bit	3	10	sfr.bit ← 1			
		A.bit	2	6	A.bit ← 1			
		X.bit	2	6	X.bit ← 1			
		PSW.bit	2	5	PSW.bit ← 1	x	x	x
	CLR1	saddr.bit	2	3/7	(saddr.bit) ← 0			
		sfr.bit	3	10	sfr.bit ← 0			
		A.bit	2	6	A.bit ← 0			
		X.bit	2	6	X.bit ← 0			
		PSW.bit	2	5	PSW.bit ← 1	x	x	x
	NOT1	saddr.bit	3	6/10	(saddr.bit) ← ($\overline{\text{saddr.bit}}$)			
		sfr.bit	3	10	sfr.bit ← $\overline{\text{sfr.bit}}$			
		A.bit	2	6	A.bit ← $\overline{\text{A.bit}}$			
		X.bit	2	6	X.bit ← $\overline{\text{X.bit}}$			
		PSW.bit	2	5	PSW.bit ← $\overline{\text{PSW.bit}}$	x	x	x
	SET1	CY	1	2	CY ← 1			1
	CLR1	CY	1	2	CY ← 0			0
	NOT1	CY	1	2	CY ← $\overline{\text{CY}}$			x

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Call/ return	CALL	!addr16	3	11/15	(SP-1)(SP-2) ← PC+3,PC ←!addr16 SP ← SP-2			
		rp	2	12-16	(SP-1)(SP-2) ← PC+2,PC ← rp SP ← SP-2			
	CALLF	!addr11	2	11/15	(SP-1)(SP-2) ← PC+2,PC_{12-11} ←01, PC_{10-0} ← !addr11, SP ← SP-2			
	CALLT	[addr5]	1	14/18	(SP-1)(SP-2) ← PC+1,PC_H ← (addr5+1),PC_L ← (addr5), SP ← SP-2			
	RET		1	10/14	PC_L ← (SP), PC_H ← (SP+1), SP ← SP+2			
	RET1		1	15/21	PC_L ← (SP), PC_H ← (SP+1), PSW ← (SP+2), SP ← SP+3	R	R	R
Stack manipulation	PUSH	rp	1	8/12	(SP-1) ← rp_H, (SP-2) ← rp_L, SP ← SP-2			
		PSW	1	5/7	(SP-1) ← PSW, SP ← SP-1			
	POP	rp	1	11/15	rp_L ← (SP), rp_H (SP+1), SP ← SP+2			
		PSW	1	6/8	PSW ← (SP), SP ← SP+1	R	R	R

(to be continued)

Remark: When high-order 8 bits (SP8-SP15) are changed in the call return or the stuck manipulation instructions, the number of clocks is incremented by one or two.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Stack manipulation	MOVW	SP,#word	4	8	SP ← word			
		SP,AX	2	9	SP ← AX			
		AX,SP	2	10	AX ← SP			
Unconditional branch	BR	!addr16	3	5	PC ← !addr16			
		rp	2	6	$PC_H \leftarrow rp_H$, $PC_L \leftarrow rp_L$			
		$addr16	2	4	PC ← PC+2+jdisp8			
Conditional branch	BC BL	$addr16	2	4(2)	PC ← PC+2+jdisp8 if CY=1			
	BNC BNL	$addr16	2	4(2)	PC ← PC+2+jdisp8 if CY=0			
	BZ BE	$addr16	2	4(2)	PC ← PC+2+jdisp8 if Z=1			
	BNZ BNE	$addr16	2	4(2)	PC ← PC+2+jdisp8 if Z=0			
	BT	saddr.bit, $addr16	3	7(5)	PC ← PC+3+jdisp8 if (saddr.bit)=1			
		sfr.bit,$addr16	4	9(7)	PC ← PC+4+jdisp8 if sfr.bit=1			
		A.bit,$addr16	3	7(5)	PC ← PC+3+jdisp8 if A.bit=1			

(to be continued)

Remark: Values in parentheses in the clock field of the conditional branch instruction indicate the number of clocks when no branch was taken.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag		
						Z	AC	CY
Conditional branch	BT	X.bit,$addr16	3	7(5)	PC ← PC+3+jdisp8 if X.bit=1			
		PSW.bit,$addr16	3	7(5)	PC ← PC+3+jdisp8 if PSW.bit=1			
	BF	saddr.bit, $addr16	4	7(5)	PC ← PC+4+jdisp8 if (saddr.bit)=0			
		sfr.bit,$addr16	4	9(7)	PC ← PC+4+jdsip8 if sfr.bit=0			
		A.bit,$addr16	3	7(5)	PC ← PC+3+jdsip8 if A.bit=0			
		X.bit,$addr16	3	7(5)	PC ← PC+3+jdisp8 if X.bit=0			
		PSW.bit,$addr16	3	7(5)	PC ← PC+3+jdisp8 if PSW.bit=0			
	BTCLR	saddr.bit, $addr16	4	9(5)	PC ← PC+4+jdisp8 if (saddr.bit)=1 then reset (saddr.bit)			
		sfr.bit, $addr16	4	13(7)	PC ← PC+4+jdisp8 if sfr.bit=1 then reset sfr.bit			
		A.bit,$addr16	3	9(5)	PC ← PC+3+jdisp8 if A.bit=1 then reset A.bit			
		X.bit,$addr16	3	9(5)	PC ← PC+3+jdisp8 if X.bit=1 then reset X.bit			
		PSW.bit, $addr16	3	8(5)	PC ← PC+3+jdisp8 if PSW.bit=1 then reset PSW.bit	x	x	x

(to be continued)

Remark: Values in parentheses in the clock field of the conditional branch instruction indicate the number of clocks when no branch was taken.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clock	Operation	Flag
						Z AC CY
Conditional branch	DBNZ	r2,$addr16	2	5(3)	r2 ← r2-1, then PC ← PC+2+jdisp8 if r2≠0	
		saddr,$addr16	3	6(4)	saddr ← saddr-1, then PC ← PC+3+jdisp8 if saddr≠0	
CPU control	MOV	STBC,#byte	4	9	STBC ← byte	
	SEL	RBn	2	2	RBS1-0 ← n n=0~3	
	NOP		1	2	No Operation	
	EI		1	2	IE ← 1(Enable Interrupt)	
	DI		1	2	IE ← 0(Disable Interrupt)	

Remark: Values in parentheses in the clock field of the conditional branch instruction indicate the number of clocks when no branch was taken.

16.2 Instruction Codes

16.2.1 Legend

(1) Symbols of instruction codes

r, r'

R_2	R_1	R_0	reg	
R_6	R_5	R_4		
0	0	0	R0	X
0	0	1	R1	A
0	1	0	R2	C
0	1	1	R3	B
1	0	0	R4	E
1	0	1	R5	D
1	1	0	R6	L
1	1	1	R7	H

r1

R_5	reg
0	A
1	B

r2

R_0	reg
0	C
1	B

r3

R_1	R_0	reg
0	0	E
0	1	E+
1	0	D

r4

R_1	reg
R_2	
R_4	
0	E
1	D

rp, rp'

P_1	P_0	reg-pair	
P_2	P_1		
P_6	P_5		
0	0	RP0	AX
0	1	RP1	BC
1	0	RP2	DE
1	1	RP3	HL

B_n: Immediate data for the bit operand

N_n: Immediate data for the n operand

Data: 8-bit immediate data for the byte operand

Low/High Byte:
16-bit immediate data for the word operand

Saddr-offset:
Offset data for eight low-order bits of 16-bit address for the saddr operand

Sfr-offset:
Offset data for eight low-order bits of 16-bit address of special function register (sfr)

Low/High Offset:
16-bit offset data for the word operand in indirect addressing

Low/High Addr.:
16-bit immediate data for the addr16 operand

jdisp: Signed 2's complement data (8 bits) indicating the relative address displacement from the starting address of the instruction next to the branch address

fa: 11 low-order bits of immediate data for the addr11 operand

ta: Five low-order bits of immediate data for (addr5 x 1/2)

Caution: If registers or register pairs are specified as both the first and second operands in the operand field, the instruction code is as follows.

In a register specification byte, four high-order bits are used for the first operand specification code, and four low-order bits are used for the second operand specification code.

Example: MOV r,r'

Instruction Code

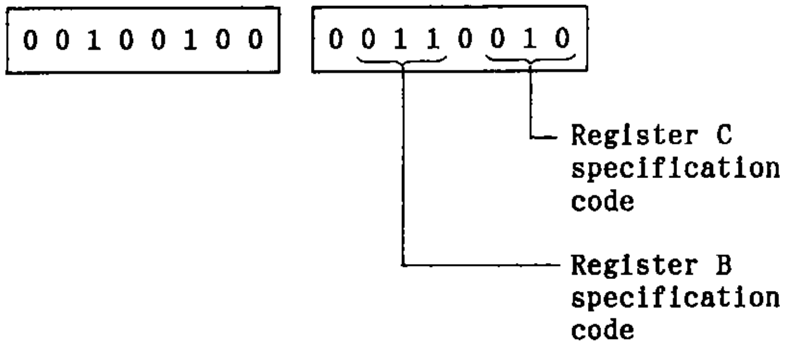
0 0 1 0 0 1 0 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0

If register B is specified as the first operand and register C as the second operand, the following instruction must be written:

MOV B,C

Then, the instruction code is as follows:

Instruction Code



16.2.2 List of instruction codes

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
8-bit data transfer	MOV	r,#byte	1 0 1 1 1 R_2 R_1 R_0	← Data →		
		saddr,#byte	0 0 1 1 1 0 1 0	← Saddr-offset →	Data	
		sfr,#byte	0 0 1 0 1 0 1 1	← Sfr-offset →	Data	
		r,r'	0 0 1 0 0 1 0 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,r	1 1 0 1 0 R_2 R_1 R_0			
		A,saddr	0 0 1 0 0 0 0 0	← Saddr-offset →		
		saddr,A	0 0 1 0 0 0 1 0	← Saddr-offset →		
		A,sfr	0 0 0 1 0 0 0 0	← Sfr-offset →		
		sfr,A	0 0 0 1 0 0 1 0	← Sfr-offset →		
		A,[r3]	0 1 1 1 1 1 R_1 R_0			
		[r3],A	0 1 1 1 1 0 R_1 R_0			
		A,[HL]	0 1 0 1 1 1 0 1			
		[HL],A	0 1 0 1 0 1 0 1			
		A,[HL+]	0 1 0 1 1 0 0 1			
		[HL+],A	0 1 0 1 0 0 0 1			
		A,[DE]	0 1 0 1 1 1 0 0			
		[DE],A	0 1 0 1 0 1 0 0			
		A,[DE+]	0 1 0 1 1 0 0 0			
		[DE+],A	0 1 0 1 0 0 0 0			
		A,!addr16	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 0	Low Addr.	High Addr.
		!addr16,A	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 1	Low Addr.	High Addr.
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 0 0 0 0	Low Offset	High Offset
		word[r1],A	0 0 0 0 1 0 1 0	1 0 R_5 1 0 0 0 0	Low Offset	High Offset
		PSW,#byte	0 0 1 0 1 0 1 1	1 1 1 1 1 1 1 0	Data	
		PSW,A	0 0 0 1 0 0 1 0	1 1 1 1 1 1 1 0		
		A,PSW	0 0 0 1 0 0 0 0	1 1 1 1 1 1 1 0		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
8-bit data transfer	XCH	A,r	1 1 0 1 1 R_2 R_1 R_0			
		A,saddr	0 0 1 0 0 0 0 1	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1	Sfr-Offset	
		A,[r4]	0 1 1 1 1 R_2 1 1			
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 0 1 0 0		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 0 1 0 0		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 0 1 0 0	Low Offset	High Offset
16-bit data transfer	MOVW	rp,#word	0 1 0 0 0 P_2 P_1 0	← Low Byte →	High Byte	
		saddrp,#word	0 0 0 0 1 1 0 0	← Saddr-offset →	Low Byte	High Byte
		sfrp,#word	0 0 0 0 1 0 1 1	← Sfr-offset →	Low Byte	High Byte
		rp,rp'	0 0 1 0 0 1 0 0	0 P_6 P_5 0 1 P_2 P_1 0		
		AX,saddrp	0 0 0 1 1 1 0 0	← Saddr-offset →		
		saddrp,AX	0 0 0 1 1 0 1 0	← Saddr-offset →		
		AX,sfrp	0 0 0 1 0 0 0 1	← Sfr-offset →		
		sfrp,AX	0 0 0 1 0 0 1 1	← Sfr-offset →		
8-bit arith-metic/logical	ADD	A,#byte	1 0 1 0 1 0 0 0	← Data →		
		saddr,#byte	0 1 1 0 1 0 0 0	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 0	Sfr-Offset	Data
		r,r'	1 0 0 0 1 0 0 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 0 0 0	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 0	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 0 0 0		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 0 0 0		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 0 0 0		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 0 0 0	Low Offset	High Offset

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
8-bit arithmetic/ logical	ADDC	A,#byte	1 0 1 0 1 0 0 1	← Data →		
		saddr,#byte	0 1 1 0 1 0 0 1	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 1	Sfr-offset	Data
		r,r'	1 0 0 0 1 0 0 1	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 0 0 1	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 1	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 0 0 1		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 0 0 1		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 0 0 1		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 0 0 1	Low Offset	High Offset
	SUB	A,#byte	1 0 1 0 1 0 1 0	← Data →		
		saddr,#byte	0 1 1 0 1 0 1 0	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 0	Sfr-offset	Data
		r,r'	1 0 0 0 1 0 1 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 0 1 0	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 0	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 0 1 0		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 0 1 0		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 0 1 0		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 0 1 0	Low Offset	High Offset
	SUBC	A,#byte	1 0 1 0 1 0 1 1	← Data →		
		saddr,#byte	0 1 1 0 1 0 1 1	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 1	Sfr-offset	Data
		r,r'	1 0 0 0 1 0 1 1	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 0 1 1	← Saddr-offset →		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
8-bit arithmetic/logical	SUBC	A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 1	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 0 1 1		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 0 1 1		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 0 1 1		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 0 1 1	Low Offset	High Offset
	AND	A,#byte	1 0 1 0 1 1 0 0	← Data →		
		saddr,#byte	0 1 1 0 1 1 0 0	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 0	Sfr-offset	Data
		r,r'	1 0 0 0 1 1 0 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 1 0 0	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 0	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 1 0 0		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 1 0 0		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 1 0 0		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 1 0 0	Low Offset	High Offset
	OR	A,#byte	1 0 1 0 1 1 1 0	← Data →		
		saddr,#byte	0 1 1 0 1 1 1 0	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 0	Sfr-offset	Data
		r,r'	1 0 0 0 1 1 1 0	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 1 1 0	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 0	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 1 1 0		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 1 1 0		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 1 1 0		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 1 1 0	Low Offset	High Offset

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
8-bit arithmetic/logical	XOR	A,#byte	1 0 1 0 1 1 0 1	← Data →		
		saddr,#byte	0 1 1 0 1 1 0 1	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 1	Sfr-offset	Data
		r,r'	1 0 0 0 1 1 0 1	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 1 0 1	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 1	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 1 0 1		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 1 0 1		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 1 0 1		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 1 0 1	Low Offset	High Offset
	CMP	A,#byte	1 0 1 0 1 1 1 1	← Data →		
		saddr,#byte	0 1 1 0 1 1 1 1	← Saddr-offset →	Data	
		sfr,#byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 1	Sfr-offset	Data
		r,r'	1 0 0 0 1 1 1 1	0 R_6 R_5 R_4 0 R_2 R_1 R_0		
		A,saddr	1 0 0 1 1 1 1 1	← Saddr-offset →		
		A,sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 1	Sfr-offset	
		A,[r4]	0 0 0 1 0 1 1 0	0 1 1 R_4 1 1 1 1		
		A,[HL]	0 0 0 1 0 1 1 0	0 1 0 1 1 1 1 1		
		A,[DE]	0 0 0 1 0 1 1 0	0 1 0 0 1 1 1 1		
		A,word[r1]	0 0 0 0 1 0 1 0	0 0 R_5 1 1 1 1 1	Low Offset	High Offset
16-bit arithmetic/logical	ADDW	AX,#word	0 0 1 0 1 1 0 1	← Low Byte →	High Byte	
		AX,rp	1 0 0 0 1 0 0 0	0 0 0 0 1 P_2 P_1 0		
		AX,saddrp	0 0 0 1 1 1 0 1	← Saddr-offset →		
		AX,sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 0 1	Sfr-offset	

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
16-bit arithmetic/ logical	SUBW	AX,#word	0 0 1 0 1 1 1 0	← Low Byte →	High Byte	
		AX,rp	1 0 0 0 1 0 1 0	0 0 0 0 1 P_2 P_1 0		
		AX,saddrp	0 0 0 1 1 1 1 0	← Saddr-offset →		
		AX,sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 0	Sfr-offset	
	CMPW	AX,#word	0 0 1 0 1 1 1 1	← Low Byte →	High Byte	
		AX,rp	1 0 0 0 1 1 1 1	0 0 0 0 1 P_2 P_1 0		
		AX,saddrp	0 0 0 1 1 1 1 1	← Saddr-offset →		
		AX,sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 1	Sfr-offset	
Multiply/ divide	MULSW	r	0 0 0 0 0 1 0 1	0 0 1 1 0 R_2 R_1 R_0		
	MULUW	r	0 0 0 0 0 1 0 1	0 0 0 0 0 R_2 R_1 R_0		
	DIVUW	r	0 0 0 0 0 1 0 1	0 0 0 1 1 R_2 R_1 R_0		
Increment/ decrement	INC	r	1 1 0 0 0 R_2 R_1 R_0			
		saddr	0 0 1 0 0 1 1 0	← saddr-offset →		
	DEC	r	1 1 0 0 1 R_2 R_1 R_0			
		saddr	0 0 1 0 0 1 1 1	← saddr-offset →		
	INCW	rp	0 1 0 0 0 1 P_1 P_0			
	DECW	rp	0 1 0 0 1 1 P_1 P_0			
Shift/ rotate	ROR	r,n	0 0 1 1 0 0 0 0	0 1 N_2 N_1 N_0 R_2 R_1 R_0		
	ROL	r,n	0 0 0 1	0 1 N_2 N_1 N_0 R_2 R_1 R_0		
	RORC	r,n	0 0 0 0	0 0 N_2 N_1 N_0 R_2 R_1 R_0		
	ROLC	r,n	0 0 0 1	0 0 N_2 N_1 N_0 R_2 R_1 R_0		
	SHR	r,n	0 0 0 0	1 0 N_2 N_1 N_0 R_2 R_1 R_0		
	SHL	r,n	0 0 0 1	1 0 N_2 N_1 N_0 R_2 R_1 R_0		
	SHRW	rp,n	0 0 0 0	1 1 N_2 N_1 N_0 P_2 P_1 0		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
Shift/ rotate	SHLW	rp,n	0 0 1 1 0 0 0 1	1 1 N_2 N_1 N_0 P_2 P_1 0		
	ROR4	[r4]	0 0 0 0 0 1 0 1	1 0 0 0 1 0 R_1 1		
	ROL4	[r4]	0 0 0 0 0 1 0 1	1 0 0 1 1 0 R_1 1		
BCD correction	ADJBA		0 0 0 0 1 1 1 0			
	ADJBS		0 0 0 0 1 1 1 1			
Bit manipulation	MOV1	CY,saddr.bit	0 0 0 0 1 0 0 0	0 0 0 0 0 B_2 B_1 B_0	Saddr-offset	
		CY,sfr.bit	1 0 0 0	1 B_2 B_1 B_0	Sfr-offset	
		CY,A.bit	0 0 1 1	1 B_2 B_1 B_0		
		CY,X.bit	0 0 1 1	0 B_2 B_1 B_0		
		CY,PSW.bit	0 0 1 0	0 B_2 B_1 B_0		
		saddr.bit,CY	1 0 0 0	0 0 0 1 0 B_2 B_1 B_0	Saddr-offset	
		sfr.bit,CY	1 0 0 0	1 B_2 B_1 B_0	Sfr-offset	
		A.bit,CY	0 0 1 1	1 B_2 B_1 B_0		
		X.bit,CY	0 0 1 1	0 B_2 B_1 B_0		
		PSW.bit,CY	0 0 1 0	0 B_2 B_1 B_0		
	AND1	CY,saddr.bit	0 0 0 0 1 0 0 0	0 0 1 0 0 B_2 B_1 B_0	Saddr-offset	
		CY,/saddr.bit		0 0 1 1 0 B_2 B_1 B_0	Saddr-offset	
		CY,sfr.bit		0 0 1 0 1 B_2 B_1 B_0	Sfr-offset	
		CY,/sfr.bit		0 0 1 1 1 B_2 B_1 B_0	Sfr-offset	
		CY,A.bit	0 0 1 1	0 0 1 0 1 B_2 B_1 B_0		
		CY,/A.bit		0 0 1 1 1 B_2 B_1 B_0		
		CY,X.bit		0 0 1 0 0 B_2 B_1 B_0		
		CY,/X.bit		0 0 1 1 0 B_2 B_1 B_0		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code					
			B1		B2		B3	B4
Bit manipulation	AND1	CY,PSW.bit	0 0 0 0	0'0 1 0	0 0 1 0	0 B_2 B_1 B_0		
		CY,/PSW.bit	↓	0 0 1 0	0 0 1 1	0 B_2 B_1 B_0		
	OR1	CY,saddr.bit	0 0 0 0	1 0 0 0	0 1 0 0	0 B_2 B_1 B_0	Saddr-offset	
		CY,/saddr.bit	↓	↓	0 1 0 1	0 B_2 B_1 B_0	Saddr-offset	
		CY,sfr.bit	↓	↓	0 1 0 0	1 B_2 B_1 B_0	Sfr-offset	
		CY,/sfr.bit	↓	↓	0 1 0 1	1 B_2 B_1 B_0	Sfr-offset	
		CY,A.bit	↓	0 0 1 1	0 1 0 0	1 B_2 B_1 B_0		
		CY,/A.bit	↓	↓	0 1 0 1	1 B_2 B_1 B_0		
		CY,X.bit	↓	↓	0 1 0 0	0 B_2 B_1 B_0		
		CY,/X.bit	↓	↓	0 1 0 1	0 B_2 B_1 B_0		
		CY,PSW.bit	↓	0 0 1 0	0 1 0 0	0 B_2 B_1 B_0		
		CY,/PSW.bit	↓	0 0 1 0	0 1 0 1	0 B_2 B_1 B_0		
	XOR1	CY,saddr.bit	0 0 0 0	1 0 0 0	0 1 1 0	0 B_2 B_1 B_0	Saddr-offset	
		CY,sfr.bit	↓	1 0 0 0	↓	1 B_2 B_1 B_0	Sfr-offset	
		CY,A.bit	↓	0 0 1 1	↓	1 B_2 B_1 B_0		
		CY,X.bit	↓	0 0 1 1	↓	0 B_2 B_1 B_0		
		CY,PSW.bit	↓	0 0 1 0	↓	0 B_2 B_1 B_0		
	SET1	saddr.bit	1 0 1 1	0 B_2 B_1 B_0	← Saddr-offset →			
		sfr.bit	0 0 0 0	1 0 0 0	1 0 0 0	1 B_2 B_1 B_0	Sfr-offset	
		A.bit	↓	0 0 1 1	↓	1 B_2 B_1 B_0		
		X.bit	↓	0 0 1 1	↓	0 B_2 B_1 B_0		
		PSW.bit	↓	0 0 1 0	↓	0 B_2 B_1 B_0		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
Bit manipulation	CLR1	saddr.bit	1 0 1 0 0 B_2 B_1 B_0	← Saddr-offset →		
		sfr.bit	0 0 0 0 1 0 0 0	1 0 0 1 1 B_2 B_1 B_0	Sfr-offset	
		A.bit	│ 0 0 1 1	│ 1 B_2 B_1 B_0		
		X.bit	│ 0 0 1 1	│ 0 B_2 B_1 B_0		
		PSW.bit	↓ 0 0 1 0	↓ 0 B_2 B_1 B_0		
	NOT1	saddr.bit	0 0 0 0 1 0 0 0	0 1 1 1 0 B_2 B_1 B_0	Saddr-offset	
		sfr.bit	│ 1 0 0 0	│ 1 B_2 B_1 B_0	Sfr-offset	
		A.bit	│ 0 0 1 1	│ 1 B_2 B_1 B_0		
		X.bit	│ 0 0 1 1	│ 0 B_2 B_1 B_0		
		PSW.bit	↓ 0 0 1 0	↓ 0 B_2 B_1 B_0		
	SET1	CY	0 1 0 0 0 0 0 1			
	CLR1	CY	0 1 0 0 0 0 0 0			
	NOT1	CY	0 1 0 0 0 0 1 0			
Call/ return	CALL	!addr16	0 0 1 0 1 0 0 0	← Low Addr. →	High Addr.	
		rp	0 0 0 0 0 1 0 1	0 1 0 1 1 P_2 P_1 0		
	CALLF	!addr11	1 0 0 1 0 ←	fa →		
	CALLT	[addr5]	1 1 1 ← ta →			
	RET		0 1 0 1 0 1 1 0			
	RETI		0 1 0 1 0 1 1 1			
Stack manipulation	PUSH	rp	0 0 1 1 1 1 P_1 P_0			
		PSW	0 1 0 0 1 0 0 1			
	POP	rp	0 0 1 1 0 1 P_1 P_0			
		PSW	0 1 0 0 1 0 0 0			
	MOVW	SP,#word	0 0 0 0 1 0 1 1	1 1 1 1 1 1 0 0	Low Byte	High Byte
		SP,AX	0 0 0 1 0 0 1 1	1 1 1 1 1 1 0 0		
		AX,SP	0 0 0 1 0 0 0 1	1 1 1 1 1 1 0 0		

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
Unconditional branch	BR	!addr16	0 0 1 0 1 1 0 0	← Low Addr. →	High Addr.	
		rp	0 0 0 0 0 1 0 1	0 1 0 1 1 P_2 P_1 0		
		$addr16	0 0 0 1 0 1 0 0	← jdisp →		
Conditional branch	BC	$addr16	1 0 0 0 0 0 1 1	← jdisp →		
	BL					
	BNC	$addr16	↓ 0 0 1 0	← jdisp →		
	BNL					
	BZ	$addr16	↓ 0 0 0 1	← jdisp →		
	BE					
	BNZ	$addr16	↓ 0 0 0 0	← jdisp →		
	BNE					
	BT	saddr.bit, $addr16	0 1 1 1 0 B_2 B_1 B_0	← Saddr-offset →	jdisp	
		sfr.bit, $addr16	0 0 0 0 1 0 0 0	1 0 1 1 1 B_2 B_1 B_0	Sfr-offset	jdisp
		A.bit, $addr16	↓ 0 0 1 1	↓ 1 B_2 B_1 B_0	jdisp	
		X.bit, $addr16	↓ 0 0 1 1	↓ 0 B_2 B_1 B_0	jdisp	
		PSW.bit, $addr16	↓ 0 0 1 0	↓ 0 B_2 B_1 B_0	jdisp	
	BF	saddr.bit, $addr16	0 0 0 0 1 0 0 0	1 0 1 0 0 B_2 B_1 B_0	Saddr-offset	jdisp
		sfr.bit, $addr16	↓ 1 0 0 0	↓ 1 B_2 B_1 B_0	Sfr-offset	jdisp
		A.bit, $addr16	↓ 0 0 1 1	↓ 1 B_2 B_1 B_0	jdisp	
		X.bit, $addr16	↓ 0 0 1 1	↓ 0 B_2 B_1 B_0	jdisp	
		PSW.bit, $addr16	↓ 0 0 1 0	↓ 0 B_2 B_1 B_0	jdisp	

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Instruction code			
			B1	B2	B3	B4
Conditional branch	BTCLR	saddr.bit, \$addr16	0 0 0 0 1 0 0 0	1 1 0 1 0 B_2 B_1 B_0	Saddr-offset	jdisp
		sfr.bit, \$addr16	↓ 1 0 0 0	↓ 1 B_2 B_1 B_0	Sfr-offset	jdisp
		A.bit, \$addr16	↓ 0 0 1 1	↓ 1 B_2 B_1 B_0	jdisp	
		X.bit, \$addr16	↓ 0 0 1 1	↓ 0 B_2 B_1 B_0	jdisp	
		PSW.bit, \$addr16	↓ 0 0 1 0	↓ 0 B_2 B_1 B_0	jdisp	
	DBNZ	r2,\$addr16	0 0 1 1 0 0 1 R_0	← jdisp →		
		saddr, \$addr16	0 0 1 1 1 0 1 1	← Saddr-offset →	jdisp	
CPU control	MOV	STBC,#byte	0 0 0 0 1 0 0 1	1 1 0 0 0 0 0 0	$\overline{\text{Data}}$	Data
	SEL	RBn	0 0 0 0 0 1 0 1	1 0 1 0 1 0 N_1 N_0		
	NOP		0 0 0 0 0 0 0 0			
	EI		0 1 0 0 1 0 1 1			
	DI		0 1 0 0 1 0 1 0			

16.3 Number of Clocks of the Instructions

16.3.1 Legend

(1) Number of clocks according to the different memory spaces

The number of clocks of the instructions varies according to the memory area where the instruction to be executed is located or where data is read from or written into.

Memory areas are classified as follows:

(1) Instruction fetch (memory in which the instruction is located)

Fetch from
Internal ROM: Values in these columns indicate the number of clocks when a program in the internal ROM is executed with MM register IFCH = 1 (high-speed fetch).

Fetch from
External ROM: Values in these columns indicate the number of clocks when a program is executed in the external programmable memory.

(ii) Access memory (where data is read from or written into)

. IROM:	Internal program memory
. IRAM:	Area FE00H-FEFFH of the internal RAM
. PRAM:	Area FC80H-FDFFH of the internal RAM
. SFR:	Special function register
. EMEM:	External memory

(2) Numeric symbols in clock field

One clock cycle of an instruction is equivalent to a clock cycle of the internal system clock; $1/f_{CLK}$. (See Chapter 4.)

(i) n in the clock field of the shift/rotate instruction indicates the number of bits to be shifted.

(ii) The value in parentheses in the clock field of the conditional branch instruction indicates the number of clocks when no branch was taken.

(iii) Values in parentheses in the clock fields of the call/return and stack manipulation instructions are the numbers of clocks required to change the high-order eight bits of the stack pointer.

(iv) Blank fields indicate the memory areas which cannot be accessed.

Phase-out/Discontinued

16.3.2 Numbers of clocks of the instructions

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
8-bit data transfer	MOV	r,#byte	2		2					6			
		saddr,#byte	3		3		5			9		9	
		sfr,#byte(*1)	3				5					9	
		r,r'	2		2					6			
		A,r	1		2					3			
		A,saddr	2		2		4			6		6	
		saddr,A	2		3		5			6		8	
		A,sfr	2				4					6	
		sfr,A	2				5					6	
		A,[r3](*2)	1		5(6)					6(7)			
		[r3],A(*2)	1		5(6)					6(7)			
		A,[HL]	1	6	5	7	7	7	7	6	8	8	8
		[HL],A	1		5	7	7	7		6	8	8	8
		A,[HL+]	1	9	8	10	10	10	10	9	11	11	11
		[HL+],A	1		8	10	10	10		9	11	11	11
		A,[DE]	1	6	5	7	7	7	7	6	8	8	8
		[DE],A	1		5	7	7	7		6	8	8	8
		A,[DE+]	1	9	8	10	10	10	10	9	11	11	11
		[DE+],A	1		8	10	10	10		9	11	11	11
		A,!addr16	4	7	6	8	8	8	15	14	16		16
		!addr16,A	4		6	8	8	8		14	16		17
		A,word[r1]	4	8	7	9	9	9	15	14	16	16	16
		word[r1],A	4		7	9	9	9		14	16	16	16
		PSW,A	2				5					6	
		A,PSW	2				4					6	
		PSW,#byte	3				5					9	

(to be continued)

Remark: The items marked an asterisk are explained on the next page.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
8-bit data transfer	XCH	A,r	1		4					4			
		A,saddr	2		4		8			6		10	
		A,sfr	3				10					13	
		A,[r4]	1		9					(*3) 10(9)			
		A,[HL]	2		9	13	13	13		12	16	16	16
		A,[DE]	2		9	13	13	13		12	16	16	16
		A,word[r1]	4		9	13	13	13		16	20	20	20

(to be continued)

*1 If STBC is written in sfr, a different dedicated instruction having the different byte and clock counts is generated. (See CPU control instructions.)

*2 If E+ is written in r3, the contents of register E are incremented by 1 after instruction execution, and the number is changed to the values enclosed in parentheses.

*3 10 clocks for the uPD78136, uPD78138, uPD78P138, and the IE-78130-R.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
16-bit data transfer	MOVW	rp,#word	3		3					9			
		saddrp,#word	4		4		8			12		12	
		sfrp,#word	4				8					12	
		rp,rp'	2		4					6			
		AX,saddrp	2		6		10			8		12	
		saddrp,AX	2		5		9			7		12	
		AX,sfrp	2				10					12	
		sfrp,AX	2				9					12	
8-bit arith-metic/ logical	ADD	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[r1]	4	9	8	10	10	10	14	15	17	17	17
	ADDC	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
8-bit arith-metic/ logical	ADDC	A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[rl]	4	9	8	10	10	10	14	15	17	17	17
	SUB	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[rl]	4	9	8	10	10	10	14	15	17	17	17
	SUBC	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[rl]	4	9	8	10	10	10	14	15	17	17	17
	AND	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
8-bit arith-metic/ logical	AND	r,r'	2		3					7	·		
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[r1]	4	9	8	10	10	10	14	15	17	17	17
	OR	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[r1]	4	9	8	10	10	10	14	15	17	17	17
	XOR	A,#byte	2		2					6			
		saddr,#byte	3		4		8			9		11	
		sfr,#byte	4				10					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
8-bit arithmetic/logical	XOR	A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[r1]	4	9	8	10	10	10	14	15	17	17	17
	CMP	A,#byte	2		2					6			
		saddr,#byte	3		3		5			9		11	
		sfr,#byte	4				7					14	
		r,r'	2		3					7			
		A,saddr	2		3		5			6		7	
		A,sfr	3				7					10	
		A,[r4]	2		7					11			
		A,[HL]	2	9	8	10	10	10	13	12	14	14	14
		A,[DE]	2	9	8	10	10	10	13	12	14	14	14
		A,word[r1]	4	9	8	10	10	10	14	15	17	17	17
16-bit arithmetic/logical	ADDW	AX,#word	3		4					9			
		AX,rp	2		6					8			
		AX,saddrp	2		7		11			9		13	
		AX,sfrp	3				13					16	
	SUBW	AX,#word	3		4					9			
		AX,rp	2		6					8			
		AX,saddrp	2		7		11			9		13	
		AX,sfrp	3				13					16	
	CMPW	AX,#word	3		3					9			
		AX,rp	2		5					7			
		AX,saddrp	2		6		10			8		12	
		AX,sfrp	3				12					15	

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
Multiply/divide	MULSW	$r^{(*)}$	2		47					49			
	MULUW	$r^{(*)}$	2		47					49			
	DIVUW	$r^{(*)}$	2		74					76			
Increment/decrement	INC	r	1		2					3			
		saddr	2		3		7			6		7	
	DEC	r	1		2					3			
		saddr	2		3		7			6		7	
	INCW	rp	1		3					3			
	DECW	rp	1		3					3			
Shift/rotate	ROR	r,n	2		3+2n					5+2n			
	ROL	r,n	2		3+2n					5+2n			
	RORC	r,n	2		3+2n					5+2n			
	ROLC	r,n	2		3+2n					5+2n			
	SHR	r,n	2		3+2n					5+2n			
	SHL	r,n	2		3+2n					5+2n			
	SHRW	rp,n	2		3+3n					5+3n			
	SHLW	rp,n	2		3+3n					5+3n			
	ROR4	[r4]	2		22					24			
	ROL4	[r4]	2		23					25			
BCD correction	ADJBA		1		3					3			
	ADJBS		1		3					3			
Bit manipulation	MOV1	CY,saddr.bit	3		5		7			9		9	
		CY,sfr.bit	3				7					9	
		CY,A.bit	2		5					7			

(to be continued)

* Except for the register A or X.

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
Bit manipulation	MOV1	CY,X.bit	2		5					7			
		CY,PSW.bit	2				5					7	
		saddr.bit,CY	3		8		12			12		14	
		sfr.bit,CY	3				12					14	
		A.bit,CY	2		8					10			
		X.bit,CY	2		8					10			
		PSW.bit,CY	2				7					9	
	AND1	CY,saddr.bit	3		5		7			9		11	
		CY,/saddr.bit	3		5		7			9		11	
		CY,sfr.bit	3				7					11	
		CY,/sfr.bit	3				7					11	
		CY,A.bit	2		5					7			
		CY,/A.bit	2		5					7			
		CY,X.bit	2		5					7			
		CY,/X.bit	2		5					7			
		CY,PSW.bit	2				5					7	
		CY,/PSW.bit	2				5					7	
	OR1	CY,saddr.bit	3		5		7			9		11	
		CY,/saddr.bit	3		5		7			9		11	
		CY,sfr.bit	3				7					11	
		CY,/sfr.bit	3				7					11	
		CY,A.bit	2		5					7			
		CY,/A.bit	2		5					7			
		CY,X.bit	2		5					7			
		CY,/X.bit	2		5					7			
		CY,PSW.bit	2				5					7	
		CY,/PSW.bit	2				5					7	

(to be continued)

(Cont'd)

Instruction set	Mnemonic	Operand	Byte	Clocks									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
Bit manipulation	XOR1	CY,saddr.bit	3		5		7			9		11	
		CY,sfr.bit	3				7					11	
		CY,A.bit	2		5					7			
		CY,X.bit	2		5					7			
		CY,PSW.bit	2				5					7	
	SET1	saddr.bit	2		3		7			6		11	
		sfr.bit	3				10					14	
		A.bit	2		6					8			
		X.bit	2		6					8			
		PSW.bit	2				5					7	
	CLR1	saddr.bit	2		3		7			6		11	
		sfr.bit	3				10					14	
		A.bit	2		6					8			
		X.bit	2		6					8			
		PSW.bit	2				5					7	
	NOT1	saddr.bit	3		6		10			10		14	
		sfr.bit	3				10					14	
		A.bit	2		6					8			
		X.bit	2		6					8			
		PSW.bit	2				5					7	
	SET1	CY	1				2					3	
	CLR1	CY	1				2					3	
	NOT1	CY	1				2					3	

Instruction set	Mnemonic	Operand	Byte	SP	Clocks			
					Fetch from internal ROM		Fetch from external ROM	
					IROM→IROM	IROM→EMEM	EMEM→IROM	EMEN→IROM
Call/ return	CALL	!addr16	3	IRAM	11(12)		15(16)	17(18)
				PRAM	15(16)		19(20)	21(22)
				EMEM	15(16)		19(20)	21(22)
		rp	2	IRAM	12(13)		13(14)	15(16)
				PRAM	16(17)		17(18)	19(20)
				EMEM	16(17)		17(18)	19(20)
	CALLF	!addr11	2	IRAM	11(12)		12(13)	14(15)
				PRAM	15(16)		16(17)	18(19)
				EMEM	15(16)		16(17)	18(19)
	CALLT	[addr5]	1	IRAM	14(15)		14(15)	20(21)
				PRAM	18(19)		18	24(25)
				EMEM	18(19)		18	24(25)
	RET		1	IRAM	10(11)		10(11)	11(12)
				PRAM	14(15)		14(15)	15(16)
				EMEM	14(15)		14(15)	15(16)
	RET1		1	IRAM	15(16)		15(16)	15(16)
				PRAM	21(22)		21(22)	21(22)
				EMEM	21(22)		21(22)	21(22)

Instruction set	Mnemonic	Operand	Byte	Clock									
				Fetch from internal ROM					Fetch from external ROM				
				IROM	IRAM	PRAM	SFR	EMEM	IROM	IRAM	PRAM	SFR	EMEM
Stack manipulation	PUSH	PSW	1		5(7)	7(9)		7(9)		5(7)	7(9)		7(9)
		rp	1		8(9)	12(13)		12(13)		8(9)	12(13)		12(13)
	POP	PSW	1		6	8		8		6	8		8
		rp	1		11(12)	15(16)		15(16)		11(12)	15(16)		15(16)
	MOVW	SP,#word	4				8					12	
		SP,AX	2				9					11	
		AX,SP	2				10					12	

Instruction set	Mnemonic	Operand	Byte	Clocks					
				Fetch from internal ROM			Fetch from external ROM		
				No branch	INT→INT	INT→EXT	No branch	EXT→INT	EXT→EXT
Unconditional branch	BR	!addr16	3		5			9	11
		rp	2		6			8	10
		$addr16	2		4			7	9
Conditional branch	BC	$addr16	2	2	4		6	7	9
	BL		2	2	4		6	7	9
	BNC	$addr16	2	2	4		6	7	9
	BNL		2	2	4		6	7	9
	BZ	$addr16	2	2	4		6	7	9
	BE		2	2	4		6	7	9
	BNZ	$addr16	2	2	4		6	7	9
	BNE		2	2	4		6	7	9
	BT	saddr.bit,$addr16	3	5	7		9		12
		sfr.bit,$addr16	4	7	9		13		16
		A.bit,$addr16	3	5	7		9		12
		X.bit,$addr16	3	5	7		9		12
		PSW.bit,$addr16	3	5	7		9		12
	BF	saddr.bit,$addr16	4	5	7		12		15
		sfr.bit,$addr16	4	7	9		13		16
		A.bit,$addr16	3	5	7		9		12
		X.bit,$addr16	3	5	7		9		12
		PSW.bit,$addr16	3	5	7		9		12
	BTCLR	saddr.bit,$addr16	4	5	9		12		15
		sfr.bit,$addr16	4	7	13		13		18
		A.bit,$addr16	3	5	9		9		12
		X.bit,$addr16	3	5	9		9		12
		PSW.bit,$addr16	3	5	8		9		12
	DBNZ	r2,$addr16	2	3	5		6		9
		saddr,$addr16	3	4	6		9		12

Instruction set	Mnemonic	Operand	Byte	Clocks	
				Fetch from internal ROM	Fetch from external ROM
CPU control	NOP		1	2	3
	EI		1	2	3
	DI		1	2	3
	SEL	RBn	2	2	6
	MOV	STBC,#byte	4	9	15

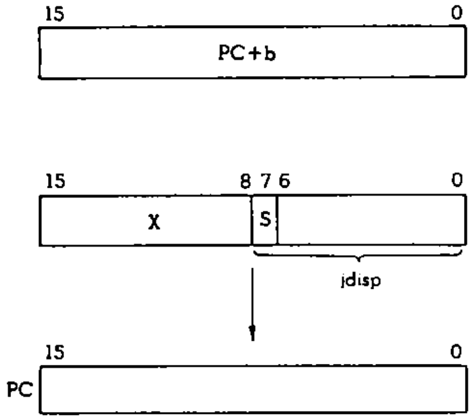
16.4 Instruction Addressing

The instruction address is determined by the program counter (PC) contents. Normally, whenever one instruction is executed, the PC is automatically incremented according to the number of the bytes of the fetched instruction (increment by one per byte). When an instruction involving a branch is executed, branch address information is loaded into the PC according to the addressing described below and a branch is taken.

16.4.1 Relative addressing

The result of adding the low-order 8-bit immediate data of a given instruction code (displacement: jdisp) to the top address of the next instruction is loaded into the program counter (PC) and a branch is taken. The displacement is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

This is performed when the BR $addr16 instruction or the conditional branch instruction is executed.



...b indicates the number of bytes of the instruction.

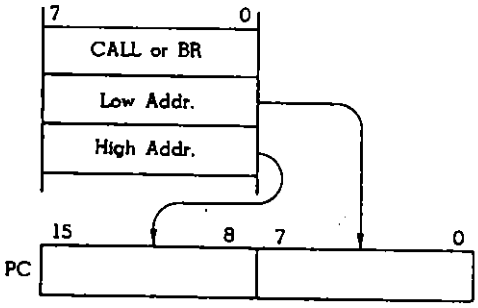
When S = 0, all bits in X are 0s.
When S = 1, all bits in X are 1s.

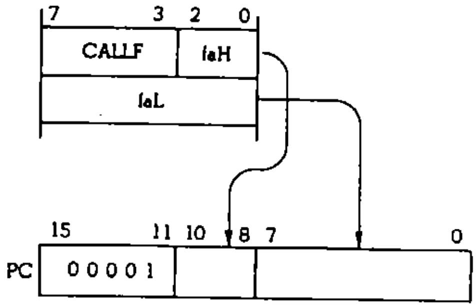
16.4.2 Immediate addressing

The immediate data in the instruction is loaded into the PC and a branch is taken.

This is performed when the CALL !addr16, BR !addr16, or CALLF !addr11 instruction is executed.

For the CALLF !addr11 instruction, a branch is taken to the fixed area whose five high-order bits contain a specific address.

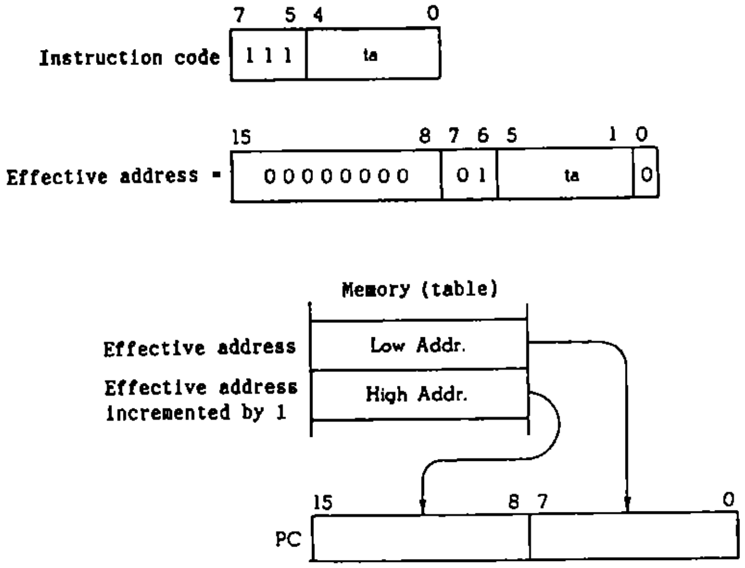




16.4.3 Table indirect addressing

The contents (branch address) of the table in the specific location addressed by the immediate data in five low-order bits in the instruction code are loaded into the PC and a branch is taken.

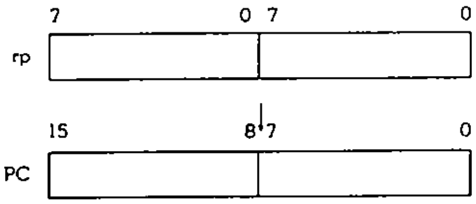
This is performed when the CALLT[addr5] instruction is executed.



16.4.4 Register addressing

The contents of the register pair (RP3 to RP0) specified by the instruction are loaded into the PC and a branch is taken.

This is performed when the BR rp, or CALL rp instruction is executed.



16.5 Operand Addressing

The addressing modes of registers and memory to be operated on in instruction execution are described below:

16.5.1 Register addressing

The general register to be specified is addressed as an operand by the contents of the register specification code such as Rn or Pn in an instruction in the register bank specified by the register bank selection flag (RBS1 and RBS0).

The register addressing is made when an instruction having any of the following operand identifiers is executed. When an 8-bit register is addressed, eight signals are specified with three bits in the instruction code. When a 16-bit register is addressed, four signals are specified with two bits in the instruction code.

Identifier	Description
r,r'	X(R0), A(R1), C(R2), B(R3), E(R4), D(R5), L(R6), H(R7)
r1	A, B
r2	B, C
r3	D, E, E+
r4	D, E
rp,rp'	AX(RP0), BC(RP1), DE(RP2), HL(RP3)

Functional names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL) can be specified in r, r', rp, and rp', as well as absolute names (R0 to R7 and RP0 to RP3). The function names correspond to the absolute names as shown in Table 3-4.

Example 1: MOV A,r

Instruction code

1	1	0	1	0	R_2	R_1	R_0

To specify register C in r, enter the following:

MOV A,C

The instruction code is as follows:

Instruction code

1	1	0	1	0	0	1	0

Example 2: INCW rp

Instruction code

0	1	0	0	0	1	P_1	P_0

To specify register pair DE in rp, enter the following:

INCW DE

The instruction code is as follows:

Instruction code

0	1	0	0	0	1	1	0

16.5.2 Immediate addressing

The 8-bit data or 16-bit data to be operated on is contained in a given instruction code.

The immediate addressing is made when an instruction having one of the following operand identifiers is executed:

Identifier	Description
byte	Label, numeric value of up to 8 bits
word	Label, numeric value of up to 16 bits

Example: ADD A, #byte

Instruction code

1	0	1	0	1	0	0	0

Data

To specify 77H in byte, enter the following:

ADD A, #77H

The instruction code is as follows:

Instruction code

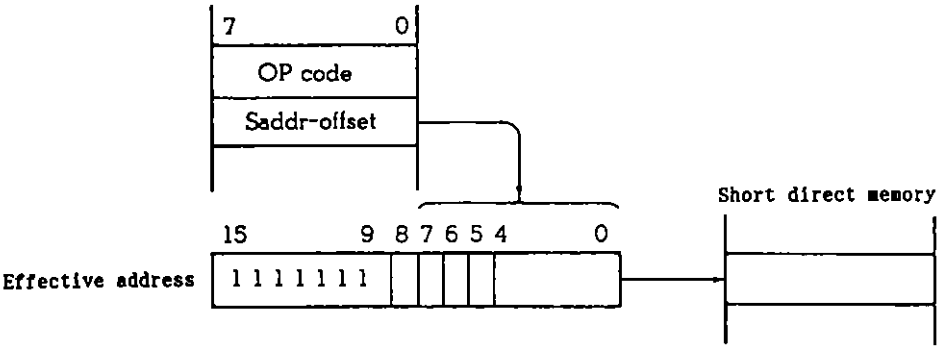
1	0	1	0	1	0	0	0
0	1	1	1	0	1	1	1

16.5.3 Short direct addressing

The memory location to be operated on in fixed space is directly addressed by the 8-bit immediate data in a given instruction.

This addressing is applied to 256-byte space from FE20H to FF1FH. The internal RAM (short direct memory) is mapped from FE20H to FEFFH and the special function register (SFR) is mapped from FF00H to FF1FH.

Bit 8 of an effective address is set to 0 when 8-bit immediate data is 20H to FFH. Or, the bit is set to 1 when the immediate data is 00H to 1FH.



This addressing is performed when an instruction having one of the saddr or saddrp operands is executed. The 2-byte data in the memory locations addressed by the effective address and by the next address (data at an even-odd address pair where the lowest bit of an effective address is ignored) are accessed by an instruction having saddrp.

Identifier	Description
saddr	Label, numeric value of FE20H to FF1FH
saddrp	Label, even numeric value of FE20H to FF1EH

Example: MOV saddr, #byte

Instruction code 0 0 1 1 1 0 1 0

Saddr-offset

Data

To specify FE20H in saddr of the first operand and 50H in byte of the second operand, enter the following:

```
MOV 0FE20H, #50H
```

The instruction code is as follows:

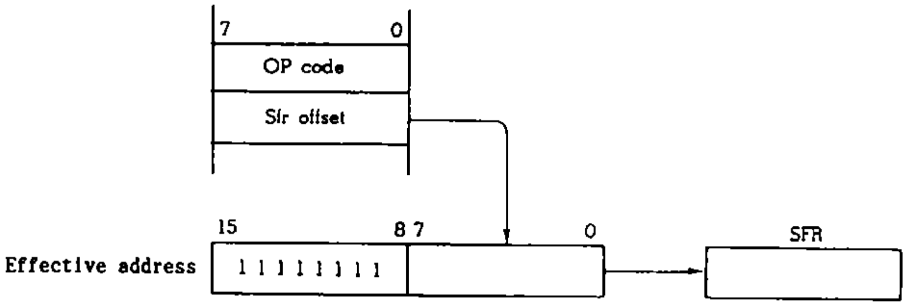
Instruction code

0	0	1	1	1	0	1	0
0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0

16.5.4 Special function register (SFR) addressing

The special function register (SFR) mapped into a memory location is addressed by the 8-bit immediate data in an instruction.

The space into which the SFR to be addressed is mapped is a 256-byte space from FF00H to FFFFH. The SFR mapped into FF00H-FF1FH is not accessed by SFR addressing, but accessed by short direct addressing.



Identifier	Description
sfr	Special function register name
sfrp	16-bit manipulation special function register name

Example: MOV sfr, A

Instruction code

0	0	0	1	0	0	1	0

Sfr-offset

To specify PM0 in sfr, enter the following:

MOV PM0, A

The instruction code is as follows:

Instruction code

0	0	0	1	0	0	1	0

0	0	1	0	0	0	0	0

16.5.5 Register indirect addressing

The memory location to be operated on is addressed by the contents of the 8-bit register and register pair HL indicated by the register specification code in an instruction in the register bank specified by the register bank select flag (RBS1-RBS0) as an operand address.

This addressing is made when an instruction having any of the following operand identifiers is executed:

Identifier	Description
[r3]	[D], [E], [E+]
[r4]	[D], [E]
[HL]	[HL]

Register indirect addressing using register E provides the function of incrementing the contents of register E by one to prepare for next addressing.

[E+] must be specified in the operand field in this case.

Example 1: MOV A, [r3]

Instruction code

0	1	1	1	1	1	R_1	R_0

To specify [E+] in [r3], enter the following:

MOV A, [E+]

The instruction code is as follows:

Instruction code

0	1	1	1	1	1	0	1

Example 2: ROR4 [r4]

Instruction code

0	0	0	0	0	1	0	1
1	0	0	0	1	0	R_1	1

To specify register E in r4, enter the following:

ROR4 [E]

The instruction code is as follows:

Instruction code

0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1

Example 3: ADD A, [r4]

Instruction code

0	0	0	1	0	1	1	0
0	1	1	R_4	1	0	0	0

To specify [D] in [r4], enter the following:

ADD A, [D]

The instruction code is as follows:

Instruction code

0	0	0	1	0	1	1	0
0	1	1	1	1	0	0	0

16.5.6 Indexed addressing

The memory location to be operated on is addressed by the sum of the contents of the 8-bit register (A, B), indicated by the addressing specification bit (R_5) in a given instruction in the register bank specified by the register bank selection flag (RBS1 - RBS0), and a 16-bit immediate data in operand as an operand address.

This addressing is made when an instruction having any of the following operand identifiers is executed:

Identifier	Description
word [r1]	word [A], word [B]

Example: ADDC A, word [r1]

Instruction code

0	0	0	0	1	0	1	0
0	0	R_5	1	0	0	0	0
Low				Offset			
High				Offset			

To specify indexed addressing by the sum of the contents of register B and 1F10H, enter the following:

```
   ADDC A, 1F10H [B]
```

The instruction code is as follows:

Instruction code	0	0	0	0	1	0	1	0
	0	0	1	1	0	0	0	0
	0	0	0	1	0	0	0	0
	0	0	0	1	1	1	1	1

16.6 Explanation of Instructions

16.6.1 8-bit data transfer instructions

MOV r, #byte

Function: r ← byte byte=00H-FFH
Transfers the 8-bit immediate data specified in the second operand to the 8-bit register specified in the first operand.

Flag operation: No change

Example: MOV A, #4DH: Sets 4DH in register A.

MOV saddr, #byte

Function: (saddr) ← byte saddr=FE20H-FF1FH
byte=00H-FFH
Transfers the 8-bit immediate data specified in the second operand to the short direct memory addressed in the first operand.
Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation: No change

Example: MOV 0FE40H, #40H: Stores 40H at address FE40H.

MOV sfr, #byte

Function: sfr ← byte byte=00H-FFH
Transfers the 8-bit immediate data specified in the second operand to the special function register (sfr) specified in the first operand.

Caution: If STBC is specified in sfr, the dedicated instruction code which is different from that of this instruction is generated (see Section 16.5.14).

Flag operation: No change

Example: MOV PM5, #0H: Specifies port 5 as an output port.

MOV r, r'

Function: r ← r'

Transfers the contents of the 8-bit register specified in the second operand to the 8-bit register specified in the first operand.

Flag operation: No change

Example:

SEL RB0: Specifies bank 0.

MOV H, A: Transfers the contents of register A to register H.

MOV A, r

Function: A ← r

Transfers the contents of the 8-bit register specified in the second operand to register A.

Flag operation: No change

Example: None

MOV A, saddr

Function: A ← (saddr) saddr=FE20H-FF1FH

Transfers the contents of the short direct memory addressed in the second operand to register A.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation: No change

Example: MOV A, 0FE40H: Transfers the contents at address FE40H to register A in the specified bank.

MOV saddr, A

Function: (saddr) ← A saddr=FE20H-FF1FH

Transfers the contents of register A to the short direct memory specified in the first operand.

Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation: No change

Example:
SEL RB2: Specifies bank 2.
MOV A, X :
MOV 0FE60H, A Transfers the contents of register X to the memory location at address FE60H.

MOV A, sfr

Function: A ← sfr

Transfers the contents of the special function register specified in the second operand to register A.

Flag operation: No change

Example: MOV A, SIO: Transfers serial receive data to register A in the current register bank.

MOV sfr, A

Function: sfr ← A

Transfers the contents of register A to the special function register specified in the first operand.

Flag operation: No change

Example: MOV P1, A: Set the contents of register A in port 1 output latch.

MOV PM1, #00H: Set port 1 to the output enable mode.

MOV A, [r3]

Function: A ← (FE00H+r3) r3=00H-FFH

Transfers the contents of the memory addressed in the second operand to register A. The address of the memory to be accessed is fixed to FEH in eight high-order bits, and the contents of the 8-bit register entered in the second operand are specified as the address in eight low-order bits.

The value in range of 00H to FFH must be set in the 8-bit register.

If E+ is specified in r3, the contents of register E are automatically incremented by one after data transfer.

Flag operation: No change

Example: MOV E, #60H: E ← 60H

MOV A, [E+]: A ← (FE60H), E ← E+1

MOV [r3], A

Function: (FE00H+r3) ← A r3=00H-FFH

Transfers the contents of register A to the memory addressed in the first operand. The address of the memory to be accessed is fixed to FEH in eight high-order bits, and the contents of the 8-bit register entered in the first operand are specified as the address in eight low-order bits.

The value in range of 00H to FFH must be set in the 8-bit register.

If E+ is specified in r3, the contents of register E are automatically incremented by one after data transfer.

Flag operation: No change

Example: None

MOV A, [HL]

Function: A ← (HL)

Transfers the contents of the memory addressed by the contents of register pair HL to register A.

This instruction enables table data to be read from the internal ROM.

Flag operation: No change

Example: None

MOV [HL], A

Function: (HL) ← A

Transfers the contents of register A to the memory addressed by the contents of register pair HL.

Flag operation: No change

Example: None

MOV A, [HL+]

Function: A ← (HL),

HL ← HL+1

Transfers the contents of the memory addressed by the contents of register pair HL to the contents of register A. The contents of register pair HL are then automatically incremented by one. This instruction enables table data to be read from the internal ROM.

Flag operation: No change

Example: None

MOV [HL+], A

Function: (HL) ← A,

HL ← HL+1

Transfers the contents of register A to the memory addressed by the contents of register pair HL.

The contents of register pair HL are then automatically incremented by one.

Flag operation: No change

Example: None

MOV A, [DE]

Function: A ← (DE),

Transfers the contents of the memory addressed by the contents of register pair DE to the contents of register A.

This instruction enables table data to be read from the internal ROM.

Flag operation: No change

Example: None

MOV [DE], A

Function: (DE) ← A,
Transfers the contents of register A to the memory addressed by the contents of register pair DE.

Flag operation: No change

Example: None

MOV A, [DE+]

Function: A ← (DE),
DE ← DE+1
Transfers the contents of the memory addressed by the contents of register pair DE to the contents of register A. The contents of register pair DE are then automatically incremented by one. This instruction enables table data to be read from the internal ROM.

Flag operation: No change

Example: None

MOV [DE+], A

Function: (DE) ← A,
DE ← DE+1
Transfers the contents of register A to the memory addressed by the contents of register pair DE.

The contents of register pair DE are then automatically incremented by one.

Flag operation: No change

Example: None

MOV A, !addr16

Function: A ← (addr16)

Transfers the contents of the memory specified in the second operand to the contents of register A.

Enter 16-bit immediate data (addr16) in the second operand.

Flag operation: No change

Example: MOV A, !0500H: Transfers the contents of the area at address 0500H to register A.

MOV !addr16, A

Function: (addr16) ← A

Transfers the contents of register A to the memory specified in the first operand.

Enter 16-bit immediate data (addr16) in the first operand.

Flag operation: No change

Example: None

MOV A, word [r1]

Function: A ← (word+r1)
Transfers the contents of the memory specified in the second operand to register A. The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register. This instruction enables table data to be read from the internal ROM.

Flag operation: No change

Example:

MOV B, #08H:	B ← 08H
MOV A, 0FE50H[B]:	A ← (FE50H+08H)

MOV word [r1], A

Function: (word+r1) ← A
Transfers the contents of register A to the memory specified in the first operand. The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the first operand and the contents of the 8-bit register.

Flag operation: No change

Example: None

MOV PSW, #byte

Function: PSW ← byte   byte=00H-FFH
Transfers the 8-bit immediate data specified in the second operand to PSW.

Flag operation:

Z	AC	CY
x	x	x

Example: None

MOV PSW, A

Function: PSW ← A

Transfers the contents of register A to PSW.

Flag operation:

Z	AC	CY
x	x	x

Example: None

MOV A, PSW

Function: A ← PSW

Transfers the contents of PSW to register A.

Flag operation: No change

Example: MOV A, PSW

XCH A, r

Function: A ↔ r

Exchanges the contents between register A and the 8-bit register specified in the second operand.

Flag operation: No change

Example: None

XCH A, saddr

Function: A ↔ (saddr) saddr=FE20H-FF1FH
Exchanges the contents between register A and the short direct memory addressed in the second operand.
Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation: No change

Example: XCH A, 0FEBCH: Exchanges the contents between register A and the area at address FEBCH.

XCH A, sfr

Function: A ↔ sfr
Exchanges the contents between register A and the special function register specified in the second operand.

Flag operation: No change

Example: XCH A, SIO: Exchanges the contents between register A and the serial shift register.

XCH A, [r4]

Function: A ↔ (+r4) r4=00H-FFH
Exchanges the contents between register A and the memory addressed in the second operand. The address of the memory to be accessed is fixed to FEH in eight high-order bits, and the contents of the 8-bit register entered in the second operand are specified as the address in eight low-order bits.

The value in range of 00H to FFH must be set in the 8-bit register.

Flag operation: No change

Example: None

XCH A, [HL]

Function: A <-> (HL)

Exchanges the contents between register A and the memory addressed by the contents of register pair HL.

Flag operation: No change

Example: None

XCH A, [DE]

Function: A <-> (DE)

Exchanges the contents between register A and the memory addressed by the contents of register pair HL.

Flag operation: No change

Example: None

XCH A, word [r1]

Function: A <-> word [r1]

Exchanges the contents between register A and the memory addressed in the second operand. The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation: No change

Example: None

16.6.2 16-bit data transfer instructions

MOVW rp, #word

Function: rp ← word word=0000H-FFFFH

Transfers the 16-bit immediate data specified in the second operand to the 16-bit register pair specified in the first operand.

Flag operation: No change

Example: MOVW RP0, #0AA55H: Transfers AA55H to register pair AX.

MOVW saddrp, #word

Function: (saddrp) ← word saddrp=FE20H-FF1EH
word=0000H-FFFFH

Transfers the 16-bit immediate data specified in the second operand to the 2-byte area in the short direct memory addressed in the first operand. Enter the address or label of the short direct memory in saddrp of the first operand. Only an even address is effective, however.

Flag operation: No change

Example: MOVW 0FE80H, #0000H: Transfers 0000H to the area at addresses FE81H and FE80H. □

MOVW sfrp, #word

Function: sfrp ← word word=0000H-FFFFH

Transfers the 16-bit immediate data specified in the second operand to the 16-bit special function register specified in the first operand.

Flag operation: No change

Example: MOVW PWM0, #0FF00H: Sets FF00H in register PWM0.

MOVW rp, rp'

Function: rp ← rp'

Transfers the contents of the 16-bit register pair specified in the second operand to the 16-bit register pair specified in the first operand.

Flag operation: No change

Example: None

MOVW AX, saddrp

Function: AX ← (saddrp) saddrp=FE20H-FF1EH

Transfers the contents of the 2-byte area in the short direct memory addressed in the second operand to register pair AX.

Enter the address or label of the short direct memory in saddrp of the second operand. Only an even address is effective, however.

Flag operation: No change

Example: MOVW AX, 0FE80H:

Transfers the contents at addresses FE81H and FE80H to register pair AX.

MOVW saddrp, AX

Function: (saddrp) ← AX saddrp=FE20H-FF1EH

Transfers the contents of register pair AX to the 2-byte area in the short direct memory addressed in the first operand.

Enter the address or label of the short direct memory in saddrp of the first operand. Only an even address is effective, however.

Flag operation: No change

Example: None

MOVW AX, sfrp

Function: AX ← sfrp

Transfers the contents of the 16-bit special function register specified in the second operand to register pair AX.

Flag operation: No change

Example: MOVW AX, CPT0: Transfers the contents of register CPT0 to register pair AX.

MOVW sfrp, AX

Function: sfrp ← AX

Transfers the contents of register pair AX to the 16-bit special function register specified in the first operand.

Flag operation: No change

Example: None

16.6.3 8-bit arithmetic/logical instructions

ADD A, #byte

Function: A, CY ← A+byte byte=00H-FFH

Adds the 8-bit immediate data specified in the second operand in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: ADD A, #40H: Adds 40H to the contents of register A in binary.

ADD saddr, #byte

Function: (saddr), CY ← (saddr)+byte

saddr=FE20H-FF1FH
byte=00H-FFH

Adds the 8-bit immediate data specified in the second operand in binary to the contents of the short direct memory addressed by the first operand. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x	x	x

Example: ADD 0FE80H, #80H: Adds 80H to the contents of the area at address FE80H in binary.

ADD sfr, #byte

Function: sfr, CY ← sfr+byte   byte=00H-FFH

Adds the 8-bit immediate data specified in the second operand in binary to the contents of the special function register specified in the first operand. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: ADD P6, #1H: Adds the contents of port 6 output latch and 1H in binary and set the result in the output latch.

ADD r, r'

Function: r, CY ← r+r'

Adds the contents of the register specified in the second operand in binary to the contents of the register specified in the first operand.

The carry flag is set when a carry has resulted from this addition.

The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADD A, saddr

Function: A, CY ← A+(saddr)   saddr=FE20H-FF1FH

Adds the contents of the short direct memory addressed in the second operand to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADD A, sfr

Function: A, CY ← A+sfr

Adds the contents of the special function register specified in the second operand in binary to the contents of register A.

The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADD A, [r4]

Function: A, CY ← A+(FE00H+r4) r4=00H-FFH

Adds the contents of the memory addressed in the second operand in binary to the contents of register A. The high-order eight bits of the address of the memory to be accessed are always FEH. The low-order eight bits are specified by the contents of the 8-bit register specified in the second operand.

The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

The value in the range from 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example:

```
MOV E, #45H:  E ← 45H
ADD A, [E]:   A, CY ← A+(FE45H)
```

□

ADD A, [HL]

Function: A, CY ← A+(HL)

Adds the contents of the memory addressed by the register pair HL in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADD A, [DE]

Function: A, CY ← A+(DE)

Adds the contents of the memory addressed by the contents of register pair DE in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADD A, word [r1]

Function: A, CY ← A+word[r1]
Adds the contents of the memory addressed in the second operand in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.
The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, #byte

Function: A, CY ← A+byte+CY byte=00H-FFH
Adds the 8-bit immediate data specified in the second operand including the carry flag to the contents of register A, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC saddr, #byte

Function: (saddr), CY ← (saddr)+byte+CY
saddr=FE20H-FF1FH
byte=00H-FFH

Adds the 8-bit immediate data specified in the second operand including the carry flag to the contents of the short direct memory addressed by the first operand. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.
Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC sfr, #byte

Function: sfr, CY ← sfr+byte+CY byte=00H-FFH

Adds the contents of the 8-bit immediate data specified in the second operand including the carry flag in binary to the contents of the special function register specified in the first operand. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC r, r'

Function: r, CY ← r+r'+CY

Adds the contents of the 8-bit register specified in the second operand including the carry flag to the contents of the 8-bit register specified in the first operand, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, saddr

Function: A, CY ← A+(saddr)+CY saddr=FE20H-FF1FH

Adds the contents of the short direct memory addressed by the second operand including the carry flag to the contents of register A in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, sfr

Function: A, CY ← A+sfr+CY

Adds the contents of the special function register specified in the second operand including the carry flag to register A in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, [r4]

Function: A, CY ← A+(FE00H+r4)+CY r4=00H-FFH

Adds the contents of the memory addressed by the second operand including the carry flag in binary to the contents of register A. The high-order eight bits of the address of the memory to be accessed are always FEH. The low-order eight bits are specified by the contents of the 8-bit register specified in the second operand. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

The value in the range from 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, [HL]

Function: A, CY ← A+(HL)+CY

Adds the contents of the memory addressed by the contents of register pair HL including the carry flag in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, [DE]

Function: A, CY ← A+(DE)+CY

Adds the contents of the memory addressed by the contents of register pair DE including the carry flag in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDC A, word [r1]

Function: A, CY ← A+word[r1]+CY

Adds the contents of the memory addressed by the second operand including the carry flag in binary to the contents of register A. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: ADDC A, 1234H[B]:

Adds the contents of the area at address 1234H + (the contents of register B) including the CY flag to the contents of register A and stores the result in register A.

SUB A, #byte

Function: A, CY ← A-byte byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction.

The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: SUB A, #40H: Subtracts 40H from the contents of register A in binary. □

SUB saddr, #byte

Function: (saddr), CY ← (saddr)-byte

saddr=FE20H-FF1FH
byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of the short direct memory addressed by the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x	x	x

Example: SUB 0FE80H, #80H: Subtracts 80H from the contents of the area at address FE80H in binary. □

SUB sfr, #byte

Function: sfr, CY ← sfr-byte byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of the special function register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: SUB P0, #1H: Subtracts 1H from the contents of the port 0 output latch in binary and sets the result in the port 0 output latch.

SUB r, r'

Function: r, CY ← r-r'

Subtracts the contents of the 8-bit register specified in the second operand from the contents of the 8-bit register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, saddr

Function: A, CY ← A-(saddr)   saddr=FE20H-FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction. Enter the address or the label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, sfr

Function: A, CY ← A-sfr

Subtracts the contents of the special function register specified in the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, [r4]

Function: A, CY ← A-(FE00H+r4) r4=00H-FFH

Subtracts the contents of the memory addressed by the second operand from the contents of register A. The eight high-order bits of the address of the memory to be accessed are always FEH. The eight low-order bits are specified by the contents of the 8-bit register specified in the second operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction. The value in the range of 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, [HL]

Function: A, CY ← A-(HL)

Subtracts the contents of the memory addressed by the contents of register pair HL from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, [DE]

Function: A, CY ← A-(DE)

Subtracts the contents of the memory addressed by the contents of register pair DE from the contents of register A. The carry flag is set when a carry has resulted from this subtraction. The carry flag is reset when a carry has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUB A, word[r1]

Function: A, CY ← A-word[r1]

Subtracts the contents of the memory addressed by the second operand from the contents of register A.

The carry flag is set when a carry has resulted from this subtraction. The carry flag is reset when a carry has not resulted from the subtraction.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example:

MOV B, #08H:	B ← 08H
SUB A, 0FE50H[B]:	A, CY ← A-(FE50H+ 08H)

SUBC A, #byte

Function: A, CY ← A-byte-CY byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand including the carry flag from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC saddr, #byte

Function: (saddr), CY ← (saddr)-byte-CY

saddr=FE20H-FF1FH
byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand including the carry flag from the contents of the short direct memory addressed by the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction. Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC sfr, #byte

Function: sfr, CY ← sfr-byte-CY byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand including the carry flag from the contents of the special function register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC r, r'

Function: r, CY ← r-r'-CY

Subtracts the contents of the 8-bit register specified in the second operand including the carry flag from the contents of the 8-bit register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC A, saddr

Function: A, CY ← A-(saddr)-CY saddr=FE20H-FF1FH

Subtracts the contents of the short direct memory addressed by the second operand including the carry flag from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC A, sfr

Function: A, CY ← A-sfr-CY

Subtracts the contents of the special function register specified in the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC A, [r4]

Function: A, CY ← A-(FE00H+r4)-CY r4=00H-FFH

Subtracts the contents of the memory addressed by the second operand including the carry flag from the contents of register A. The eight high-order bits of the address of the memory to be accessed are always FEH. The eight low-order bits are specified by the contents of the 8-bit register specified in the second operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction. The value in the range of 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC A, [HL]

Function: A, CY ← A-(HL)-CY

Subtracts the contents of the memory addressed by the contents of register pair HL including the carry flag from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBC A, [DE]

Function: A, CY ← A-(DE)-CY

Subtracts the contents of the memory addressed by the contents of register pair DE including the carry flag from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: SUBC A, [DE]:

Subtracts the contents of the area at the address indicated by the contents of register pair DE including the carry flag from the contents of register A and stores the result in register A.

SUBC A, word[r1]

Function: A, CY ← A-word[r1]-CY

Subtracts the contents of the memory addressed by the second operand including the carry flag from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

AND A, #byte

Function: A ← A ∧ byte   byte=00H-FFH

ANDs the contents of register A and the 8-bit immediate data specified in the second operand and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: AND A, #0FFH: ANDs the contents of register A and FFH.

AND saddr, #byte

Function: (saddr) ← (saddr) ∧ byte

saddr=FE20H-FF1FH
byte=00H-FFH

ANDs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the short direct memory addressed by the first operand.
Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x		

Example: AND 0FE40H, #0F0H:

Resets only the four low-order bits of the contents of the area at address FE40H. (The four high-order bits do not change.)

AND sfr, #byte

Function: sfr ← sfr ∧ byte   byte=00H-FFH

ANDs the contents of the special function register specified in the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the special function register specified in the first operand.

Flag operation:

Z	AC	CY
x		

Example: AND P6, #0FH:

Resets only the four high-order bits in the port 6 output latch. (The low-order four bits do not change.)

AND r, r'

Function: r ← r ∧ r'

ANDs the contents of the 8-bit register specified in the first operand and the 8-bit register specified in the second operand, and sets the result in the 8-bit register specified in the first operand.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, saddr

Function: A ← A ∧ (saddr) saddr=FE20H-FF1FH

ANDs the contents of register A and the short direct memory specified in the second operand, and sets the result in register A.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, sfr

Function: A ← A ∧ sfr

ANDs the contents of register A and the special function register specified in the second operand, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, [r4]

Function: $A \leftarrow A \wedge (FE00H+r4)$ r4=00H-FFH

ANDs the contents of register A and the memory addressed by the second operand, and sets the result in register A. The eight high-order bits of the address of the memory to be accessed are always FEH and the eight low-order bits are specified by the contents of the 8-bit register specified in the second operand. The value in the range of 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, [HL]

Function: $A \leftarrow A \wedge (HL)$

ANDs the contents of register A and the memory addressed by the contents of register pair HL, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, [DE]

Function: A ← A ∧ (DE)

ANDs the contents of register A and the memory addressed by the contents of register pair DE, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

AND A, word [r1]

Function: A ← A ∧ word[r1]

ANDs the contents of register A and the memory addressed by the second operand, and sets the result in register A.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

OR A, #byte

Function: A ← A ∨ byte byte=00H-FFH

ORs the contents of register A and the 8-bit immediate data specified in the second operand, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: OR A, #0FH: Outputs 1 from the four low-order bits of register A. (The four high-order bits do not change.)

OR saddr, #byte

Function: (saddr) ← (saddr) ∨ byte

saddr=FE20H-FF1FH
byte=00H-FFH

ORs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the short direct memory specified in the first operand.
Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x		

Example: None

OR sfr, #byte

Function: sfr ← sfr ∨ byte byte=00H-FFH

ORs the contents of the special function register specified in the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the special function register specified in the first operand.

Flag operation:

Z	AC	CY
x		

Example:

OR P1, #F0H : Outputs 1 from the four high order bits of port 1. (The four low-order bits do not change.)

MOV PM1, #00H

OR r, r'

Function: r ← r ∨ r'

ORs the contents of the 8-bit register specified in the first operand and the 8-bit register specified in the second operand, and sets the result in the 8-bit register specified in the first operand.

Flag operation:

Z	AC	CY
x		

Example: None

OR A, saddr

Function: A ← A ∨ (saddr) saddr=FE20H-FF1FH

ORs the contents of register A and the short direct memory addressed by the second operand, and sets the result in the register A.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x		

Example: OR A, 0FE98H:

ORs the contents of register A and the contents of the area at address FE98H bit by bit and stores the result in register A.

OR A, sfr

Function: A ← A ∨ sfr

ORs the contents of register A and the special function register specified in the second operand, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

OR A, [r4]

Function: A ← A ∨ (FE00H+r4)   r=00H-FFH

ORs the contents of register A and the memory addressed by the second operand, and sets the result in register A. The eight high-order bits of the address of the memory to be accessed are always FEH and the eight low-order bits are specified by the contents of the 8-bit register specified in the second operand. The value in the range of 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

OR A, [HL]

Function: A ← A ∨ (HL)

ORs the contents of register A and the memory addressed by the contents of register pair HL, and sets the results in register A.

Flag operation:

Z	AC	CY
x		

Example:

```
MOVW HL, #FEDOH;  ORs the contents at
OR A, [HL]        address FEDOH and the
                  contents of register A.
```

OR A, [DE]

Function: A ← A ∨ (DE)

ORs the contents of register A and the memory addressed by the contents of register pair DE, and sets the results in register A.

Flag operation:

Z	AC	CY
x		

Example: None

OR A, word [r1]

Function: A ← A ∨ word[r1]

ORs the contents of register A and the memory addressed by the second operand, and sets the result in register A.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, #byte

Function: A ← A ⊻ byte byte=00H-FFH

Exclusive ORs the contents of register A and the 8-bit immediate data specified in the second operand, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: XOR A, #0FFH: Inverts the contents of register A.

XOR saddr, #byte

Function: (saddr) ← (saddr) ⊻ byte

saddr=FE20H-FF1FH

byte=00H-FFH

Exclusive ORs the contents of the short direct memory addressed by the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the short direct memory addressed by the first operand. Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x		

Example: None

XOR sfr, #byte

Function: sfr ← sfr ⊻ byte byte=00H-FFH

Exclusive ORs the contents of the special function register specified in the first operand and the 8-bit immediate data specified in the second operand, and sets the result in the special function register.

Flag operation:

Z	AC	CY
x		

Example: XOR P1, #0FFH: Inverts the contents of the port 1 output latch.

XOR r, r'

Function: r ← r ⊻ r'

Exclusive ORs the contents of the 8-bit register specified in the first operand and the 8-bit register specified in the second operand, and sets the result in the 8-bit register specified in the first operand.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, saddr

Function: A ← A ∀ (saddr) saddr=FE20H-FF1FH

Exclusive ORs the contents of register A and the short direct memory addressed by the second operand, and sets the result in register A.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x		

Example: XOR A, 0FE50H:

Exclusive ORs the contents of register A and the contents of the area at address FE50H bit by bit and stores the result in register A.

XOR A, sfr

Function: A ← A ∀ sfr

Exclusive ORs the contents of register A and the special function register specified in the second operand, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, [r4]

Function: $A \leftarrow A \veebar (FE00H+r4)$ r4=00H-FFH

Exclusive ORs the contents of register A and the memory addressed by the second operand, and sets the result in register A. The eight high-order bits of the address of the memory to be accessed are always FEH and the eight low-order bits are specified by the contents of the 8-bit register specified in the second operand.

The value in the range of 00H to FFH must be set in the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, [HL]

Function: $A \leftarrow A \veebar (HL)$

Exclusive ORs the contents of register A and the memory addressed by the contents of register pair HL, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, [DE]

Function: $A \leftarrow A \veebar (DE)$

Exclusive ORs the contents of register A and the memory addressed by the contents of register pair DE, and sets the result in register A.

Flag operation:

Z	AC	CY
x		

Example: None

XOR A, word [r1]

Function: $A, CY \leftarrow A \veebar word[r1]$

Exclusive ORs the contents of register A and the memory addressed by the second operand, and sets the result in register A.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x		

Example: None

CMP A, #byte

Function: A - byte byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of register A do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: CMP A, #10H: Compares the contents of register A with 10H.

CMP saddr, #byte

Function: (saddr) - byte saddr=FE20H-FF1FH
byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of the short direct memory addressed by the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of the short direct memory do not change after instruction execution.

Enter the address or label of the short direct memory in saddr of the first operand.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP sfr, #byte

Function: sfr - byte byte=00H-FFH

Subtracts the 8-bit immediate data specified in the second operand from the contents of the special function register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
The contents of the special function register do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP r, r'

Function: r, CY ← r-r'

Subtracts the contents of the 8-bit register specified in the second operand from the contents of the 8-bit register specified in the first operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
The contents of the 8-bit register do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP A, saddr

Function: A - (saddr) saddr=FE20H-FF1FH

Subtracts the contents of the short direct memory addressed by the second operand from the the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of register A and the short direct memory do not change after instruction execution.

Enter the address or label of the short direct memory in saddr of the second operand.

Flag operation:

Z	AC	CY
x	x	x

Example: CMP A, #0FED0H: Compares the contents of register A with the contents of the area at FED0H. □

CMP A, sfr

Function: A - sfr

Subtracts the contents of the special function register specified in the second operand from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of register A and the special function register do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP A, [r4]

Function: A - (FE00H+r4) r4=00H-FFH

Subtracts the contents of the memory addressed in the second operand from the contents of register A. The eight high-order bits of the address of the memory to be accessed are always FEH and the eight low-order bits are specified by the contents of the 8-bit register specified in the second operand. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The value in the range of 00H to FFH must be set in the 8-bit register.

The contents of register A and the memory do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP A, [HL]

Function: A - (HL)

Subtracts the contents of the memory addressed by the contents of register pair HL from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of register A and the memory do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP A, [DE]

Function: A - (DE)

Subtracts the contents of the memory addressed by the contents of register pair DE from the contents of register A. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a carry has not resulted from the subtraction.

The contents of register A do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMP A, word [r1]

Function: A - word[r1]

Subtracts the contents of the memory addressed by the second operand from the contents of register A.

The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a carry has not resulted from the subtraction.

The contents of register A do not change after instruction execution.

The address of the memory to be accessed is specified as the value of the sum of 16-bit immediate data entered in the second operand and the contents of the 8-bit register.

Flag operation:

Z	AC	CY
x	x	x

Example: None

16.6.4 16-bit arithmetic/logical instructions

ADDW AX, #word

Function: AX, CY ← AX+word word=0000H-FFFFH

Adds the 16-bit immediate data specified in the second operand to the contents of register pair AX, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: ADDW AX, #0ABCDH:

Adds ABCDH to the contents of register pair AX in binary and stores the result in register pair AX.

ADDW AX, rp

Function: AX, CY ← AX+rp

Adds the contents of the 16-bit register pair specified in the second operand to the contents of register pair AX, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDW AX, saddrp

Function:

AX, CY ← AX+(saddrp) saddrp=FE20H-FF1EH

Adds the contents of the 2-byte area in the short direct memory addressed by the second operand to the contents of register pair AX, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Enter the address or label of the short direct memory in saddrp of the second operand. Only an even address must be entered, however.

Flag operation:

Z	AC	CY
x	x	x

Example: None

ADDW AX, sfrp

Function:

AX, CY ← AX+sfrp

Adds the contents of the 16-bit special function register specified in the second operand to the contents of register pair AX, in binary. The carry flag is set when a carry has resulted from this addition. The carry flag is reset when a carry has not resulted from the addition.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBW AX, #word

Function: AX, CY ← AX-word word=0000H-FFFFH

Subtracts the 16-bit immediate data specified in the second operand from the contents of register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBW AX, rp

Function: AX, CY ← AX-rp

Subtracts the contents of the 16-bit register pair specified in the second operand from the contents of register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBW AX, saddrp

Function: AX, CY ← AX-(saddrp) saddrp=FE20H-FF1EH
Subtracts the contents of the two-byte area in the short direct memory addressed by the second operand from the contents of register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
Enter the address or label of the short direct memory in saddrp of the second operand. Only an even address must be entered, however.

Flag operation:

Z	AC	CY
x	x	x

Example: None

SUBW AX, sfrp

Function: AX, CY ← AX-sfrp
Subtracts the contents of the 16-bit special function register specified in the second operand from the contents of register pair AX.
The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

Flag operation:

Z	AC	CY
x	x	x

Example: SUBW AX, CR01:
Subtracts the contents of register CR01 from the contents of register pair AX in binary and stores the result in register pair AX.

CMPW AX, #word

Function: AX - word word=0000H-FFFFH
Subtracts the contents of the 16-bit immediate data specified in the second operand from register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
The contents of register pair AX do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMPW AX, rp

Function: AX - rp
Subtracts the contents of the 16-bit register pair specified in the second operand from the contents of register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of the register pairs specified in the first and second operands do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

CMPW AX, saddrp

Function: AX - (saddrp) saddrp=FE20H-FF1EH

Subtracts the contents of the short direct memory addressed by the second operand from the contents of register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.

The contents of register pair AX and the short direct memory do not change after instruction execution.

Enter the address or label of the short direct memory in saddrp of the second operand. Only an even address must be specified, however.

Flag operation:

Z	AC	CY
x	x	x

Example: CMPW AX, 0FE43H:

Compares the contents of register pair AX with the contents of the area at address FE43H.

CMPW AX, sfrp

Function: AX - sfrp
Subtracts the contents of the 16-bit special function register specified in the second operand from register pair AX. The carry flag is set when a borrow has resulted from this subtraction. The carry flag is reset when a borrow has not resulted from the subtraction.
The contents of register pair AX and the 16-bit special function register specified in the second operand do not change after instruction execution.

Flag operation:

Z	AC	CY
x	x	x

Example: None

16.6.5 Multiply/divide instructions

MULSW r

Function: AX, r ← AX x r
Multiplies the contents of register pair AX (signed 16-bit data) by the contents of the 8-bit register specified in the operand (absolute 8-bit data), and sets the 16 high-order bits (signed) of the result in register pair AX and the 8 low-order bits of the result in the 8-bit register specified in the operand.
Specify an 8-bit register other than registers A and X in r.

Flag operation: No change

Example: None

MULUW r

Function: AX, r ← AX x r

Multiplies the contents of register pair AX (absolute 16-bit data) by the contents of the 8-bit register specified in the operand (absolute 8-bit data), and sets the 16 high-order bits of the result in register pair AX and the 8 low-order bits of the result in the 8-bit register specified in the operand.

Flag operation: No change

Example:

```
MOV B, #45H:  B ← 45H
MULUW B:      AX, B ← AX x 45H
```

DIVUW r

Function: AX, r ← AX ÷ r

Divides the contents of register pair AX (absolute 16-bit data) by the contents of the 8-bit register specified in the operand (absolute 8-bit data), and sets the quotient in register pair AX and the remainder in the register specified in the operand.

Flag operation: No change

Example:

```
MOV E, #15H:  E ← 15H
DIVUW E:      AX, E ← AX ÷ 15H
```

16.6.6 Increment/decrement instructions

INC r

Function: r ← r + 1
Increases the contents of the 8-bit register specified in the operand by one.

Flag operation:

Z	AC	CY
x	x	

Example: INC B: Increases the contents of register B by one.

INC saddr

Function: (saddr) ← (saddr)+1 saddr=FE20H-FF1FH
Increases the contents of the short direct memory addressed by the operand by one.
Enter the address or label of the short direct memory in saddr of the operand.

Flag operation:

Z	AC	CY
x	x	

Example: INC TB1: Increases the contents of the short direct memory of label TB1 by one.

DEC r

Function: r ← r - 1
Decreases the contents of the 8-bit register specified in the operand by one.

Flag operation:

Z	AC	CY
x	x	

Example: DEC A: Decreases the contents of register A by one.

DEC saddr

Function: (saddr) ← (saddr)-1 saddr=FE20H-FF1FH

Decreases the contents of the short direct memory addressed by the operand by one.

Enter the address or label of the short direct memory in saddr of the operand.

Flag operation:

Z	AC	CY
x	x	

Example: None

INCW rp

Function: rp ← rp + 1

Increases the contents of the 16-bit register pair specified in the operand by one.

Flag operation: No change

Example: INCW DE: DE ← DE + 1

DECW rp

Function: rp ← rp - 1

Decreases the contents of the 16-bit register pair specified in the operand by one.

Flag operation: No change

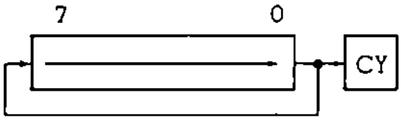
Example: DECW HL: HL ← HL - 1

Phase-out/Discontinued

16.6.7 Shift/rotate instructions

ROR r, n

Function: $(CY, r_7 \leftarrow r_0, r_{m-1} \leftarrow r_m) \times n$ n=0-7



Rotates the contents of the 8-bit register specified in the first operand right the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the LSB in the 8-bit register are both transferred to the MSB and set in the carry flag. When n = 0, these operations are not performed.

Flag operation:

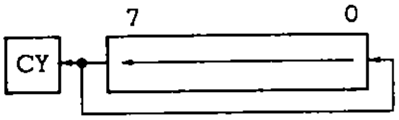
Z	AC	CY
		x

Example: ROR C, 4: Rotates the contents of register C right four bits.

□

ROL r, n

Function: $(CY, r_0 \leftarrow r_7, r_{m+1} \leftarrow r_m) \times n$ n=0-7



Rotates the contents of the 8-bit register specified in the first operand left the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the MSB in the 8-bit register are both transferred to the LSB and set in the carry flag. When n = 0, these operations are not performed.

Flag operation:

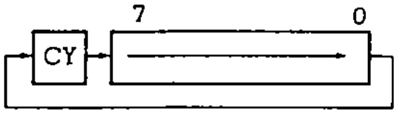
Z	AC	CY
		x

Example: ROL L, 2: Rotates the contents of register L left two bits.

RORC r, n

Function: $(CY \leftarrow r_0,\ r_7 \leftarrow CY,\ r_{m-1} \leftarrow r_m)$ x n

n=0-7



Rotates the contents of the 8-bit register specified in the first operand including the carry flag right the number of bits specified by the 3-bit immediate data specified in the second operand.

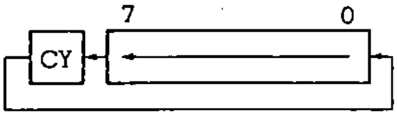
Flag operation:

Z	AC	CY
		x

Example: RORC B, 1: Rotates the contents of register B including the CY flag right one bit.

ROLC r, n

Function: $(CY \leftarrow r_7,\ r_0 \leftarrow CY,\ r_{m+1} \leftarrow r_m) \times n$
n=0-7



Rotates the contents of the 8-bit register specified in the first operand including the carry flag left the number of bits specified by the 3-bit immediate data specified in the second operand.

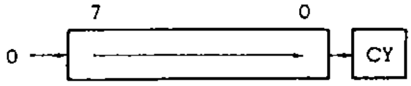
Flag operation:

Z	AC	CY
		x

Example: ROLC A, 3: Rotates the contents of register A including the CY flag left three bits.

SHR r, n

Function: $(CY \leftarrow r_0,\ r_7 \leftarrow 0,\ r_{m-1} \leftarrow r_m) \times n$ n=0-7



Shifts the contents of the 8-bit register specified in the first operand right the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the LSB in the 8-bit register are shifted into the carry flag and the MSB is set to 0. When n = 0, these operations are not performed.

Flag operation:

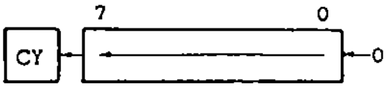
Z	AC	CY
x	0	x

Example: SHR A, 1: Divides the contents of register A by two. (The remainder is set in the CY.)

SHL r, n

Function: $(CY \leftarrow r_7,\ r_0 \leftarrow 0,\ r_{m+1} \leftarrow r_m)$ x n

n=0-7



Shifts the contents of the 8-bit register specified in the first operand left the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the MSB in the 8-bit register are shifted into the carry flag and the LSB is set to 0. When n = 0, these operations are not performed.

Flag operation:

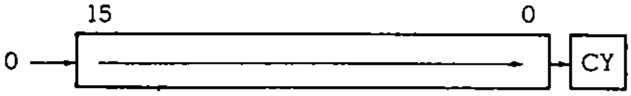
Z	AC	CY
x	0	x

Example: SHL L, 3: Shifts the contents of register L left three bit positions. The contents of bit 5 before shifted are set in the carry flag.

SHRW rp, n

Function: $(CY \leftarrow rp_0, rp_{15} \leftarrow 0, rp_{m-1} \leftarrow rp_m) \times n$

n=0-7



Shifts the contents of the 16-bit register pair specified in the first operand right the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the LSB in the 16-bit register pair are shifted into the carry flag and the MSB is set to 0. When n = 0, these operations are not performed.

Flag operation:

Z	AC	CY
x	0	x

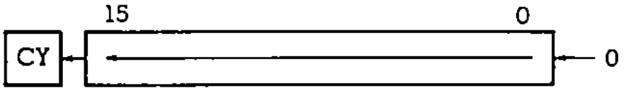
Example: SHRW AX, 3: Divides the contents of register pair AX by 8.

□

SHLW rp, n

Function: $(CY \leftarrow rp_{15}, rp_0 \leftarrow 0, rp_{m+1} \leftarrow rp_m) \times n$

n=0-7



Shifts the contents of the 16-bit register pair specified in the first operand left the number of bits specified by the 3-bit immediate data specified in the second operand. The contents of the MSB in the 16-bit register pair are shifted into the carry flag and the LSB is set to 0. When n = 0, these operations are not performed.

Flag operation:

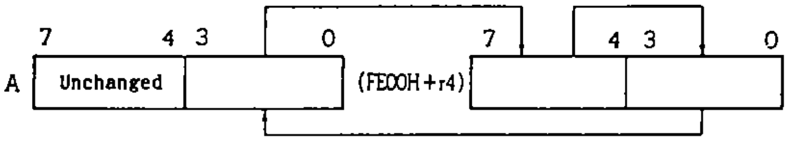
Z	AC	CY
x	0	x

Example: SHLW AX, 1: Multiplies the contents of register AX by two.

ROR4 [r4]

Function: $A_{3-0} \leftarrow (FE00H+r4)_{3-0}$,
$(FE00H+r4)_{7-4} \leftarrow A_{3-0}$,
$(FE00H+r4)_{3-0} \leftarrow (FE00H+r4)_{7-4}$

r4=00H-FFH



Rotates the contents of the four low-order bits in register A and the four high-order bits and four low-order bits of the memory addressed by the operand right in units of four bits. The eight high-order bits of the address of the memory to be accessed are always FEH, and the eight low-order bits are specified by the contents of the 8-bit register specified in the operand.
The value in range of 00H to FFH must be set in the 8-bit register.
Execution of this instruction have no effect on the four high-order bits in register A.

Flag operation: No change

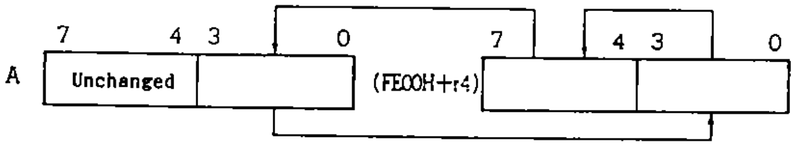
Example: ROR4 [E]: Rotates the contents of register A and memory at address FE00H + (the contents of register E) right.

	A		(FE00H+E)	
	7 4	3 0	7 4	3 0
Before execution	0 0 0 0	0 0 1 0	0 1 0 1	0 0 1 1
After execution	0 0 0 0	0 0 1 1	0 0 0 1	0 1 0 1

ROL4 [r4]

Function: $A_{3-0} \leftarrow (FE00H+r4)_{7-4}$,
$(FE00H+r4)_{3-0} \leftarrow A_{3-0}$,
$(FE00H+r4)_{7-4} \leftarrow (FE00H+r4)_{3-0}$

r4=00H-FFH



Rotates the contents of the four low-order bits in register A and the four high-order bits and four low-order bits of the memory addressed by the operand left in units of four bits. The eight high-order bits of the address of the memory to be accessed are always FEH, and the eight low-order bits are specified by the contents of the 8-bit register specified in the operand.
The value in range of 00H to FFH must be set in the 8-bit register.
Execution of this instruction have no effect on the four high-order bits in register A.

Flag operation: No change

Example: ROL4 [E]: Rotates the contents of register A and memory at address FE00H + (the contents of register E) left.

	A		(FE00H+E)	
	7 4	3 0	7 4	3 0
Before execution	0 0 0 1	0 0 1 0	0 1 0 0	1 0 0 0
After execution	0 0 0 1	0 1 0 0	1 0 0 0	0 0 1 0

16.6.8 BCD correction instructions

ADJBA

Function: Judges the contents of register A, the carry flag (CY), and the auxiliary carry flag (AC) and makes decimal correction as shown in the following table. This instruction does not have an effect until decimal (BCD) data are added.

Condition		Operation
$A_{3-0} \leqq 9$ AC = 0	$A_{7-4} \leqq 9$ and CY = 0	A ← A
	$A_{7-4} \geqq 10$ or CY = 1	A ← A + 01100000B
$A_{3-0} \geqq 10$ AC = 0	$A_{7-4} < 9$ and CY = 0	A ← A + 00000110B
	$A_{7-4} \geqq 9$ or CY = 1	A ← A + 01100110B
AC = 1	$A_{7-4} \leqq 9$ and CY = 0	A ← A + 00000110B
	$A_{7-4} \geqq 10$ or CY = 1	A ← A + 01100110B

Flag operation:

Z	AC	CY
x	x	x

Example: MOV A, #88H

ADD A, #79H: A=01H, CY=1, and AC=1

A ← A+66H, A=67H, and CY=1

88 + 79 = 167

```
         (   10001000   88H
ADD      { +)01111001   79H
        ((  ⤺00000001   01H
        |     CY  AC
ADJBA   {
        |  +)01100110   66H
        (    01100111   67H
```

ADJBS

Function: Judges the contents of register A, the carry flag (CY), and the auxiliary carry flag (AC) and makes decimal correction as shown in the following table. This instruction does not have an effect until subtraction between decimal (BCD) data is performed.

Condition		Operation
AC = 0	CY = 0	A ← A
	CY = 1	A ← A - 01100000B
AC = 1	CY = 0	A ← A - 00000110B
	CY = 1	A ← A - 01100110B

Flag operation:

Z	AC	CY
x	x	x

Example: None

16.6.9 Bit manipulation instructions

MOV1 CY, saddr.bit

Function: CY ← (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

Transfers the contents of the short direct memory bit addressed by the second operand to the carry flag.
Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: None

MOV1 CY, sfr.bit

Function: CY ← sfr.bit bit=0-7

Transfers the contents of the bit addressed by the 3-bit immediate data in the special function register specified in the second operand to the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

MOV1 CY, A.bit

Function: CY ← A.bit bit=0-7

Transfers the contents of the bit addressed by the 3-bit immediate data in register A specified in the second operand to the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

MOV1 CY, X.bit

Function: CY ← X.bit bit=0-7

Transfers the contents of the bit addressed by the 3-bit immediate data in register X specified in the second operand.

Flag operation:

Z	AC	CY
		x

Example: None

MOV1 CY, PSW.bit

Function: CY ← PSW.bit bit=0-7

Transfers the contents of the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand to the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

MOV1 saddr.bit, CY

Function: (saddr.bit) ← CY saddr=FE20H-FF1FH
bit=0-7

Transfers the contents of the carry flag to the short direct memory bit addressed by the first operand.

Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation: No change

Example: None

MOV1 sfr.bit, CY

Function: sfr.bit ← CY bit=0-7
Transfers the contents of the carry flag to the bit addressed by the 3-bit immediate data in the special function register specified in the first operand.

Flag operation: No change

Example: MOV1 P3.3, CY: Transfers the contents of the CY flag to bit 3 of port 3.

MOV1 A.bit, CY

Function: A.bit ← CY bit=0-7
Transfers the contents of the carry flag to the bit addressed by the 3-bit immediate data in register A specified in the first operand.

Flag operation: No change

Example: None

MOV1 X.bit, CY

Function: X.bit ← CY bit=0-7
Transfers the contents of the carry flag to the bits addressed by the 3-bit immediate data in register X specified in the first operand.

Flag operation: No change

Example: None

MOV1 PSW.bit, CY

Function: PSW.bit ← CY bit=0-7

Transfers the contents of the carry flag to the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the first operand.

Flag operation: No change

Example: None

AND1 CY, saddr.bit

Function: CY ← CY ^ (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

ANDs the contents of the short direct memory bit addressed by the second operand and the carry flag, and sets the result in the carry flag.

Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: AND1 CY, 0FE7FH.3:

ANDs the contents of bit 3 at address FE7FH and the CY flag and stores the result in the CY flag.

AND1 CY, /saddr.bit

Function: $CY \leftarrow CY \wedge (\overline{\text{saddr.bit}})$ saddr=FE20H-FF1FH

bit=0-7

ANDs the contents of the inverted short direct memory bit addressed by the second operand and the carry flag, and sets the result in the carry flag.

Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, sfr.bit

Function: $CY \leftarrow CY \wedge \text{sfr.bit}$ bit=0-7

ANDs the contents of the bit addressed by the 3-bit immediate data in the special function register specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, /sfr.bit

Function: $CY \leftarrow CY \wedge \overline{sfr.bit}$ bit=0-7

ANDs the contents of the inverted bit addressed by the 3-bit immediate data in the special function register specified in the second operand, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, A.bit

Function: $CY \leftarrow CY \wedge A.bit$ bit=0-7

ANDs the contents of the bit addressed by the 3-bit immediate data in register A specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, /A.bit

Function: $CY \leftarrow CY \wedge \overline{A.bit}$ bit=0-7

ANDs the contents of the inverted bit addressed by the 3-bit immediate data in register A specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, X.bit

Function: $CY \leftarrow CY \wedge X.bit$ bit=0-7

ANDs the contents of the bit addressed by the immediate data in register X specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, /X.bit

Function: $CY \leftarrow CY \wedge \overline{X.bit}$ bit=0-7

ANDs the contents of the inverted bit addressed by the 3-bit immediate data in register X specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, PSW.bit

Function: $CY \leftarrow CY \wedge PSW.bit$ bit=0-7

ANDs the contents of the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

AND1 CY, /PSW.bit

Function: $CY \leftarrow CY \wedge \overline{PSW.bit}$ bit=0-7

ANDs the contents of the inverted bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: AND1 CY, /PSW.6:

ANDs the value obtained by inverting bit 6 (Z flag) in the PSW and the contents of the CY flag and stores the result in the CY flag.

OR1 CY, saddr.bit

Function:

CY ← CY ∨ (saddr.bit)   saddr=FE20H-FF1FH
bit=0-7

ORs the contents of the short direct memory bit addressed by the second operand and the carry flag, and sets the result in the carry flag.

Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, /saddr.bit

Function:

CY ← CY ∨ ($\overline{\text{saddr.bit}}$)   saddr=FE20H-FF1FH
bit=0-7

ORs the contents of the inverted short direct memory bit addressed by the second operand and the carry flag, and sets the result in the carry flag.

Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, sfr.bit

Function: CY ← CY ∨ sfr.bit bit=0-7

ORs the contents of the bit addressed by the 3-bit immediate data in the special function register specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: OR1 CY, P2.5:

ORs the contents of bit 5 of port 2 and the CY flag and stores the result in the CY flag.

OR1 CY, /sfr.bit

Function: CY ← CY ∨ $\overline{\text{sfr.bit}}$ bit=0-7

ORs the contents of the inverted bit addressed by the 3-bit immediate data in the special function register specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, A.bit

Function: $CY \leftarrow CY \vee A.bit$ bit=0-7

ORs the contents of the bit addressed by the 3-bit immediate data in register A specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, /A.bit

Function: $CY \leftarrow CY \vee \overline{A.bit}$ bit=0-7

ORs the contents of the inverted bit addressed by the 3-bit immediate data in register A specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, X.bit

Function: $CY \leftarrow CY \vee X.bit$ bit=0-7

ORs the contents of the bit addressed by the 3-bit immediate data in register X specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, /X.bit

Function: CY ← CY ∨ $\overline{\text{X.bit}}$ bit=0-7

ORs the contents of the inverted bit addressed by the 3-bit immediate data in register X specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: OR1 CY, /X.0: ORs the value obtained by inverting bit 0 in register X and the contents of the CY flag and stores the result in the CY flag.

OR1 CY, PSW.bit

Function: CY ← CY ∨ PSW.bit bit=0-7

ORs the contents of the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

OR1 CY, /PSW.bit

Function: $CY \leftarrow CY \vee \overline{PSW.bit}$ bit=0-7

ORs the contents of the inverted bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand and the carry flag, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

XOR1 CY, saddr.bit

Function: $CY \leftarrow CY \veebar (saddr.bit)$ saddr=FE20H-FF1FH
bit=0-7

Exclusive ORs the contents of the short direct memory bit addressed by the second operand and the carry flag, and sets the result in the carry flag.
Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation:

Z	AC	CY
		x

Example: None

XOR1 CY, sfr.bit

Function: CY ← CY ⊻ sfr.bit bit=0-7

Exclusive ORs the contents of the bit addressed by the 3-bit immediate data in the special function register specified in the second operand, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

XOR1 CY, A.bit

Function: CY ← CY A.bit bit=0-7

Exclusive ORs the contents of the bit addressed by the 3-bit immediate data in register A specified in the second operand and the carry flag, and sets the contents in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: XOR1 CY, A.7: Exclusive ORs the contents of bit 7 in register A and the CY flag and stores the result in the CY flag.

XOR1 CY, X.bit

Function: CY ← CY ⊻ X.bit bit=0-7

Exclusive ORs the contents of the bit addressed by the 3-bit immediate data in register X specified in the second operand and the carry flag, and sets the contents in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

XOR1 CY, PSW.bit

Function: CY ← CY ⊻ PSW.bit bit=0-7

Exclusive ORs the contents of the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the second operand, and sets the result in the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

SET1 saddr.bit

Function: (saddr.bit) ← 1 saddr=FE20H-FF1FH
bit=0-7

Sets the short direct memory bit addressed by the operand to 1.
Enter the address or label of the short direct memory bit in saddr.bit of the operand.

Flag operation: No change

Example: None

SET1 sfr.bit

Function: sfr.bit ← 1 bit=0-7

Sets the bit addressed by the 3-bit immediate data in the special function register specified in the operand to 1.

Flag operation: No change

Example: SET1 ADM.7 : Sets bit 7 in the A/D conversion mode register to 1. (Starts A/D conversion.)

SET1 A.bit

Function: A.bit ← 1 bit=0-7

Sets the bit addressed by the 3-bit immediate data in register A of the operand to 1.

Flag operation: No change

Example: None

SET1 X.bit

Function: X.bit ← 1 bit=0-7

Sets the bit addressed by the 3-bit immediate data in register X of the operand to 1.

Flag operation: No change

Example: None

SET1 PSW.bit

Function: PSW.bit ← 1 bit=0-7

Sets the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the operand to 1.

Flag operation: The flag addressed by the operand is set to 1.

Example: None

CLR1 saddr.bit

Function: (saddr.bit) ← 0 saddr=FE20H-FF1FH
bit=0-7

Clears the short direct memory bit addressed by the operand to 0.

Enter the address or label of the short direct memory in saddr.bit of the operand.

Flag operation: No change

Example: None

CLR1 sfr.bit

Function: sfr.bit ← 0 bit=0-7

Clears the bit addressed by the 3-bit immediate data in the special function register specified in the operand to 0.

Flag operation: No change

Example: CLR1 ADM.7: Clears bit 7 in the A/D conversion mode register to 0. (Stops A/D conversion.)

CLR1 A.bit

Function: A.bit ← 0 bit=0-7
Clears the bit addressed by the 3-bit immediate data in register A specified in the operand to 0.

Flag operation: No change

Example: None

CLR1 X.bit

Function: X.bit ← 0 bit=0-7
Clears the bit addressed by the 3-bit immediate data in register X specified in the operand to 0.

Flag operation: No change

Example: None

CLR1 PSW.bit

Function: PSW.bit ← 0 bit=0-7
Clears the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the operand to 0.

Flag operation: The flag addressed by the operand is cleared to 0.

Example: None

NOT1 saddr.bit

Function: $(\text{saddr.bit}) \leftarrow (\overline{\text{saddr.bit}})$

saddr=FE20H-FF1FH

bit=0-7

Inverts the contents of the short direct memory bit addressed by the operand. Enter the address or label of the short direct memory in saddr.bit of the operand.

Flag operation: No change

Example: NOT1 0FE35H.0: Inverts the contents of bit 0 at address FE35H.

NOT1 sfr.bit

Function: $\text{sfr.bit} \leftarrow \overline{\text{sfr.bit}}$ bit=0-7

Inverts the contents of the 3-bit immediate data in the special function register specified in the operand.

Flag operation: No change

Example: None

NOT1 A.bit

Function: $\text{A.bit} \leftarrow \overline{\text{A.bit}}$ bit=0-7

Inverts the contents of the bit addressed by the 3-bit immediate data in register A specified in the operand.

Flag operation: No change

Example: None

NOT1 X.bit

Function: $X.bit \leftarrow \overline{X.bit}$ bit=0-7

Inverts the contents of the bit addressed by the 3-bit immediate data in register X specified in the operand.

Flag operation: No change

Example: None

NOT1 PSW.bit

Function: $PSW.bit \leftarrow \overline{PSW.bit}$ bit=0-7

Inverts the contents of the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the operand.

Flag operation: The contents of the flag addressed by the operand are inverted.

Example: None

SET1 CY

Function: CY ← 1

Sets the carry flag to 1.

Flag operation:

Z	AC	CY
		1

Example: None

CLR1 CY

Function: CY ← 0

Clears the carry flag to 0.

Flag operation:

Z	AC	CY
		0

Example: None

NOT1 CY

Function: CY ← $\overline{\text{CY}}$

Inverts the contents of the carry flag.

Flag operation:

Z	AC	CY
		x

Example: None

16.6.10 Call return instructions

CALL !addr16

Function: $(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$,
PC ← addr16, SP ← SP-2
addr16=0000H-FFFFH

Saves the first address (return address) of the next instruction in the memory (stack) addressed by a stack pointer (SP), decreases the contents of the SP, and causes a branch to the address indicated by the 16-bit immediate data specified in the operand.

Flag operation: No change

Example: CALL !3059H: Calls the subroutine having 3059H as its first address.

CALL rp

Function: $(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$,
PC ← rp, SP ← SP-2

Saves the first address (return address) of the next instruction in the memory (stack) addressed by a stack pointer (SP), decreases the contents of the SP, sets the contents of the 16-bit register pair in the program counter (PC), then causes a branch to the address indicated by the PC.

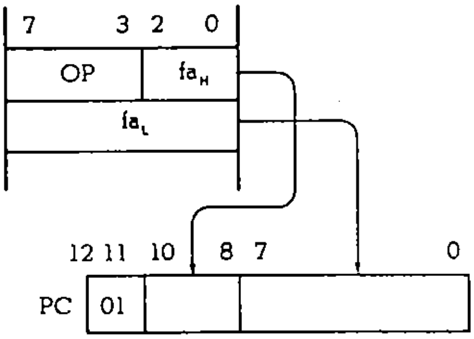
Flag operation: No change

Example: None

CALLF !addr11

Function: (SP-1) ← $(PC+2)_H$, (SP-2) ← $(PC+2)_L$, PC_{12-11} ← 01, PC_{10-0} ← fa, SP ← SP-2

addr11=0800H-OFFFH



Saves the first address (return address) of the next instruction in the memory (stack) addressed by a stack pointer (SP), decreases the contents of the SP, and causes a branch to the address addressed with the effective address which consists of 11-bit immediate data fa in the instruction code.

The call range is limited to addresses 0800H to OFFFH. Enter a branch address in addr11 of the operand directly with a label or numeric value, considering an entry address range.

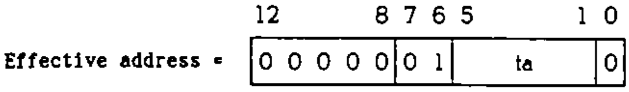
Flag operation: No change

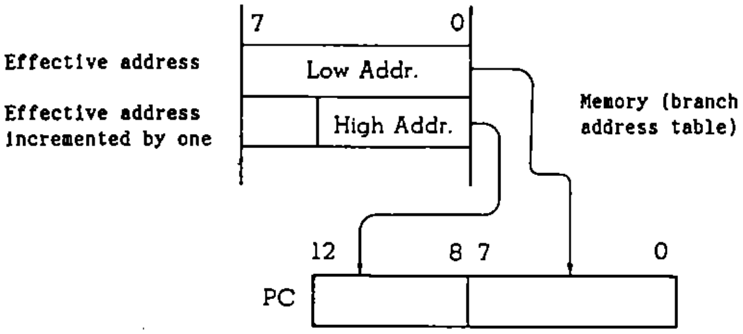
Example: None

CALLT [addr5]

Function: $(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$,
$PC_H \leftarrow (0000000001, ta, 1)$,
$PC_L \leftarrow (0000000001, ta, 0)$, $SP \leftarrow SP-2$

addr5=40H-7EH




7
0
Effective address
Low Addr.
Effective address incremented by one
High Addr.
Memory (branch address table)
12
8 7
0
PC

Saves the first address (return address) of the next instruction in the memory (stack) addressed by a stack pointer (SP), decreases the contents of the SP, sets the contents of the memory (branch address table) addressed with the effective address which consists of the 5-bit immediate data ta in the instruction code in the program counter (PC), and causes a branch to the address indicated by the memory contents.

The branch address table must be placed at addresses 0040H to 007FH. Enter the address of the branch address table in addr5 of the operand directly with a label or numeric value.

Flag operation: No change

Example: CALL [TBL1]: Causes a branch to the address indicated by the contents of the table specified by label TBL1.

RET

Function: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$, $SP \leftarrow SP+2$
Restores the contents of the memory (stack) addressed by a stack pointer to the program counter (PC) and increases the contents of the SP.

Flag operation: No change

Example: None

RETI

Function: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$, $PSW \leftarrow (SP+2)$, $SP \leftarrow SP+3$
Restores the contents of the memory (stack) addressed by a stack pointer (SP) to the program counter (PC) and program status word (PSW) and increases the contents of the SP.
This instruction is used during return from an interrupt handling routine.

Flag operation:

Z	AC	CY
R	R	R

Example: None

16.6.11 Stack manipulation instructions

PUSH rp

Function: $(SP-1) \leftarrow rp_H$, $(SP-2) \leftarrow rp_L$,
$SP \leftarrow SP-2$
Saves the contents of the 16-bit register pair specified in the operand to memory (stack) and decreases the contents of the SP.

Flag operation: No change

Example: PUSH AX: Saves the contents of register pair AX onto the stack.

PUSH PSW

Function: $(SP-1) \leftarrow PSW$, $SP \leftarrow SP-1$
Saves the contents of the program status word (PSW) in the memory (stack) addressed by a stack pointer (SP) and decreases the contents of the SP.

Flag operation: No change

Example: None

POP rp

Function: $rp_L \leftarrow (SP)$, $rp_H \leftarrow (SP+1)$, $SP \leftarrow SP+2$
Restores the contents of the memory (stack) addressed by a stack pointer (SP) to the 16-bit register pair specified in the operand and increases the contents of the SP.

Flag operation: No change

Example: POP AX: Restores the contents of the stack addressed by a stack pointer to register pair AX. □

POP PSW

Function: PSW ← (SP), SP ← SP+1

Restores the contents of the memory (stack) addressed by a stack pointer (SP) to the program status word (PSW) and decreases the contents of the SP.

Flag operation:

Z	AC	CY
R	R	R

Example: None

MOVW SP, #word

Function: SP ← word word=0000H-FFFFH

Transfers the 16-bit immediate data specified in the second operand to a stack pointer (SP).

Flag operation: No change

Example: MOVW SP, #0FE1FH: Stores FE1FH in a stack pointer. □

MOVW SP, AX

Function: SP ← AX

Transfers the contents of register pair AX to a stack pointer (SP).

Flag operation: No change

Example: None

MOVW AX, SP

Function: AX ← SP
Transfers the contents of a stack pointer (SP) to register pair AX.
Flag operation: No change
Example: None

16.6.12 Unconditional branch instructions

BR !addr16

Function: PC ← addr16 addr16=0000H-FFFFH
Transfers the 16-bit immediate data specified in the operand to the program counter (PC) and causes a branch to the address indicated by the PC.
A branch can be taken to address 0000H to FFFFH in memory.
Flag operation: No change
Example: BR BLK3: Causes a branch to the address indicated by label BLK3.

BR rp

Function: PC_H ← rp_H, PC_L ← rp_L
Transfers the contents of the 16-bit resister pair specified in the operand to the program counter (PC) and causes a branch to the address indicated by the PC.
A branch can be taken to address 0000H to FFFFH in memory.
Flag operation: No change

Caution: Do not specify address FD80H to FFFFH in rp because an instruction cannot be fetched at the address in the range.

Example: None

BR $addr16

Function: PC ← PC+2+jdisp8

addr16=(PC-126) to (PC+129)

Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction code to the first address of the next instruction and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

16.6.13 Conditional branch instructions

BC $addr16
BL $addr16

Function: PC ← PC+2+jdisp8 if CY=1
addr16=(PC-126) to (PC+129)
Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction code to the first address of the next instruction to the program counter (PC) when a carry flag is 1 and causes a branch to the address indicated by the PC.
jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.
Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: BC $300H: Causes a branch to address 0300H when CY=1.
(The branch destination address must be between the first address of the next instruction - 128 and the address + 127.)

BNC $addr16

BNL $addr16

Function: PC ← PC+2+jdisp8 if CY=0

addr16=(PC-126) to (PC+129)

Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction code to the first address of the next instruction to the program counter (PC) when a carry flag is 0 and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: CMP A, B

BNC $1500H: Causes a branch to address 1500H when the contents of register A are larger than the contents of register B. (The branch destination address must be between the first address of the next instruction - 128 and the address + 127.)

BZ $addr16

BE $addr16

Function: PC ← PC+2+jdisp8 if Z=1

addr16=(PC-126) to (PC+129)

Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction code to the first address of the next instruction to the program counter (PC) when a zero flag is 1 and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example:

```
DEC 0FE80H.   Cause a branch to the
BZ $JMP       address indicated by label
              JMP when the contents of
              the memory addressed
              by FE80H go to 0 after
              they are decremented by
              one.  (The branch
              destination shall be
              within the range (-128 to
              +127) from the first
              address.)
```

BNZ $addr16

BNE $addr16

Function: PC ← PC+2+jdisp8 if Z=0

addr16=(PC-126) to (PC+129)

Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction code to the first address of the next instruction to the program counter (PC) when a zero flag is 0 and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BT saddr.bit, $addr16

Function: PC ← PC+3+jdisp8 if (saddr.bit)=1
addr16=(PC-125) to (PC+130)
saddr=FE20H-FF1FH
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter when the bit of the short direct memory addressed by the first operand is 1, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter the address or label of the short direct memory bit in saddr.bit of the first operand and enter a branch address in addr16 of the second operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BT sfr.bit, $addr16

Function: PC ← PC+4+jdisp8 if sfr.bit=1

addr16=(PC-124) to (PC+131)

bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the fourth byte in an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the special function register specified in the first operand is 1, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the second operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BT A.bit, $addr16

Function: PC ← PC+3+jdisp8 if A.bit=1
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte in an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register A specified in the first operand is 1, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: BT A.3, $JMP1: Causes a branch to the address indicated by label JMP1 when bit 3 in register A is 1.

BT X.bit, $addr16

Function: PC ← PC+3+jdisp8 if X.bit=1

addr16=(PC-125) to (PC+130)

bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte in an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register X specified in the first operand is 1, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BT PSW.bit, $addr16

Function: PC ← PC+3+jdisp8 if PSW.bit=1
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte in an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the first operand is 1, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BF saddr.bit, $addr16

Function: PC ← PC+4+jdisp8 if (saddr.bit)=0

addr16=(PC-124) to (PC+131)

saddr=FE20H-FF1FH

bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the fourth byte in an instruction code to the first address of the next instruction to the program counter (PC) when the short direct memory bit addressed by the first operand is 0, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter the address or label of the short direct memory bit in saddr.bit of the first operand and enter a branch address in addr16 of the second operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BF sfr.bit, $addr16

Function: PC ← PC+4+jdisp8 if sfr.bit=0
addr16=(PC-124) to (PC+131)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the fourth byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the special function register specified in the first operand is 0, and causes a branch to the address indicated by the PC.
jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.
Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: BF P2.2, $1549H: Causes a branch to address 1549H when bit 2 of port 2 is 0. (The branch destination address must be between the first address of the next instruction - 128 and the address +127.)

BF A.bit, $addr16

Function: PC ← PC+3+jdisp8 if A.bit=0
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register A specified in the first operand is 0, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BF X.bit, $addr16

Function: PC ← PC+3+jdisp8 if X.bit=0
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte in an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register X specified in the first operand is 0, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BF PSW.bit, $addr16

Function: PC ← PC+3+jdisp8 if PSW.bit=0
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the program status word (PSW) specified in the first operand is 0, and causes a branch to the address indicated by the PC.
jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.
Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BTCLR saddr.bit, $addr16

Function: PC ← PC+4+jdisp8 if (saddr.bit)=1 then clear

addr16=(PC-124) to (PC+131)
saddr=FE20H-FF1FH
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the fourth byte of an instruction code to the first address of the next instruction to the program counter (PC) when the short direct memory bit addressed by the first operand is 1, causes a branch to the address indicated by the PC and clears the bit to 0.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter the address or label of the short direct memory bit in saddr.bit of the first operand and enter a branch address in addr16 of the second operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BTCLR sfr.bit, $addr16

Function: PC ← PC+4+jdisp8 if sfr.bit=1 then clear

addr16=(PC-124) to (PC+131)

bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the fourth byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the special function register specified in the first operand is 1, causes a branch to the address indicated by the PC, and clears the bit to 0.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BTCLR A.bit, $addr16

Function: PC ← PC+3+jdisp8 if A.bit=1 then clear
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register A specified in the first operand is 1, causes a branch to the address indicated by the PC, and clears the bit to 0.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BTCLR X.bit, $addr16

Function: PC ← PC+3+jdisp8 if X.bit=1 then clear
addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in register X specified in the first operand is 1, causes a branch to the address indicated by the PC, and clears the bit to 0.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

BTCLR PSW.bit, $addr16

Function: PC ← PC+3+jdisp8 if PSW.bit=1 then clear

addr16=(PC-125) to (PC+130)
bit=0-7

Transfers the value obtained by adding 8-bit displacement jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) when the bit addressed by the 3-bit immediate data in the program status word specified in the first operand is 1, causes a branch to the address indicated by the PC, and clears the bit to 0.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: If the specified flag is 1, it is reset.

Example: BTCLR PSW.6, $0F6EH:

Resets flag Z if the flag is 1 and causes a branch to address F6EH.

DBNZ r2, $addr16

Function: r2 ← r2-1, then PC ← PC+2+jdisp8
if r2≠0
addr16=(PC-126) to (PC+129)

Transfers the value obtained by adding 8-bit displacement value jdisp in the second byte of an instruction to the first address of the next instruction to the program counter (PC) if the result is not 0 after decreasing the contents of the 8-bit register specified in the first operand, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: DBNZ B, $1215H: Causes a branch to address 1215H when the result is not 0 after decreasing the contents of register B. (The branch destination address must be between the first address of the next instruction - 128 and the address + 127.)

DBNZ saddr, $addr16

Function: (saddr) ← (saddr)-1,
then PC ← PC+3+jdisp8 if (saddr) = 0
addr16=(PC-125) to (PC+130)
saddr=FE20H-FF1FH

Transfers the value obtained by adding 8-bit displacement value jdisp in the third byte of an instruction code to the first address of the next instruction to the program counter (PC) if the result is not 0 after decreasing the contents of the short direct memory addressed by the first operand, and causes a branch to the address indicated by the PC.

jdisp is treated as signed two's complement data (-128 to +127) and bit 7 is used as a sign bit.

Enter a branch address in addr16 of the operand directly with a label or numeric value, considering the branch range.

Flag operation: No change

Example: None

16.6.14 CPU control instructions

MOV STBC, #byte

Function: STBC ← byte byte=00H-FFH
Sets the 8-bit immediate data specified in the second operand in the standby control register (STBC).
This instruction is an instruction code specific to setting the STBC register.

Flag operation: No change

Example: MOV STBC, #02H: Sets the STOP mode.

SEL RBn

Function: RBS1 and RBS0 ← n n=0-3
Sets 2-bit immediate data N_{1-0} in the register bank selection flags (RBS1 and RBS0) and selects the register bank specified in the operand.

Flag operation: No change

Example: SEL RB2: Selects register bank 2 as the register bank to be used for the next and subsequent instructions.

NOP

Function: Uses two clocks without doing anything.

Flag operation: No change

Example: None

EI

Function: IE ← 1

Sets an interrupt request enable flag (IE) to 1. Each interrupt request control register controls whether a maskable interrupt is received.

Flag operation: No change

Example: None

DI

Function: IE ← 0

Clears an interrupt request enable flag (IE) to 0. Reception of all maskable interrupts are disabled. A macro service request is not, however, disabled.

Flag operation: No change

Example: None

CHAPTER 17 DIFFERENCES BETWEEN THE uPD78138 SERIES AND uPD78134

The differences between the uPD78138 series (uPD78134A, uPD78136, uPD78138, and uPD78P138) and uPD78134 are as follows:

① ROM/RAM size

Item	uPD78134	uPD78134A	uPD78136	uPD78138	uPD78P138
ROM	16K bytes (Mask ROM)		24K bytes (Mask ROM)	32K bytes (Mask ROM)	32K bytes (PROM)
RAM	384 bytes		640 bytes		

② Added instructions

The following 15 instructions are added in the uPD78138 series.

. Signed multiply instruction:	MULSW r
. 8-bit data transfer instruction:	MOV A, [HL+]
	MOV [HL+], A
	MOV A, [DE]
	MOV [DE], A
	MOV A, [DE+]
	MOV [DE+], A
	MOV A, !addr16
	MOV !addr16, A
	XCH A, [HL]
	XCH A, [DE]
	XCH A, word[r1]
. 8-bit arithmetic/logical instruction:	ALU A, [DE]
	ALU A, word[r1]
. Call instruction:	CALL rp

Remark: Legend

A:	Register A
rp:	AX, BC, DE, or HL
r:	A, X, B, C, D, E, H, or L
r1:	A or B
word:	16-bit immediate data or label
!addr16:	0000H-FFFFH immediate data or label
HL:	Register pair
DE:	Register pair
ALU:	Generic for all the mnemonics of the 8-bit arithmetic/logical instructions (ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP)

③ Operation when a value is written in an event counter compare register (ECC0 or ECC1)

. uPD78134: Clears the event counter (EC).

. uPD78138 series: Clears the event counter (EC) when 0xxxxxxxB is written.
Does not clear the event counter (EC) when 1xxxxxxxB is written.

④ Threshold width for pulse elimination for the digital noise eliminator

. uPD78134: 40/f_{CLK} fixed.

. uPD78138 series: 32/f_{CLK} or 72/f_{CLK} selectable

⑤ PWM carrier frequency

.uPD78134: 23.4 kHz fixed.

.uPD78138 series: 23.4 kHz or 46.9 kHz selectable

⑥ Restriction is lifted for the real-time output port control mode in macro service

In the uPD78138 series, the restriction that the output timing data must be stored in ROM in the real-time output port control mode in macro service in the uPD78134 is lifted.

Remark: Refer to the following manuals for the details of the uPD78134.

. uPD78134 Data Sheet (IC-7839)

. uPD78134 User's Manual (IEU-668)

APPENDIX A DEVELOPMENT TOOLS

The following development tools are readily available for development of systems using the uPD78138.

[Hardware]

IE-78130-R(*)	The IE-78130-R is an in-circuit emulator for the uPD78138. This emulator is connected to the host machine when debugging is performed. Symbolic debugging is enabled and object files can be transferred between the emulator and the host machine, thus enabling effective debugging results. The IE-78130-R contains two channels of the RS-232-C serial interfaces so that it can also be connected to PROM programmer PG-1500. The emulator also contains a Centronics interface so that object and symbol files can be downloaded at a high speed.
EP-78130GF-R	Emulation probe for the uPD78138. One EV-9200G-80 (80-pin conversion socket) is attached. This socket facilitates user system development.
EV-9200G-80	Socket to be mounted on the PC board for the user system, produced for an 80-pin plastic QFP (14 x 20 mm excluding the dimensions of the pins). This socket is used together with the EP-78130GF-R.
EV-9900	Jig for removing the uPD78P138K from the EV-9200G-80.
PG-1500	A PROM programmer with which programs can be written into single-chip microcomputers containing a PROM in standalone mode or by remote control from a host machine, when connected with the accessory board and optional programmer adapter. This PROM programmer can also be used to write programs into commonly used 256K- to 4M-bit PROMs.
PA-78P138GF PA-78P138K	PROM programmer adapters for the uPD78P138 to be used together with a PROM programmer such as the PG-1500.

* The IE-78130-R purchased before Jan. 1990 can be updated to emulate the uPD78134A, uPD78136, uPD78138, and uPD78P138A. Consult an NEC service personnel for details.

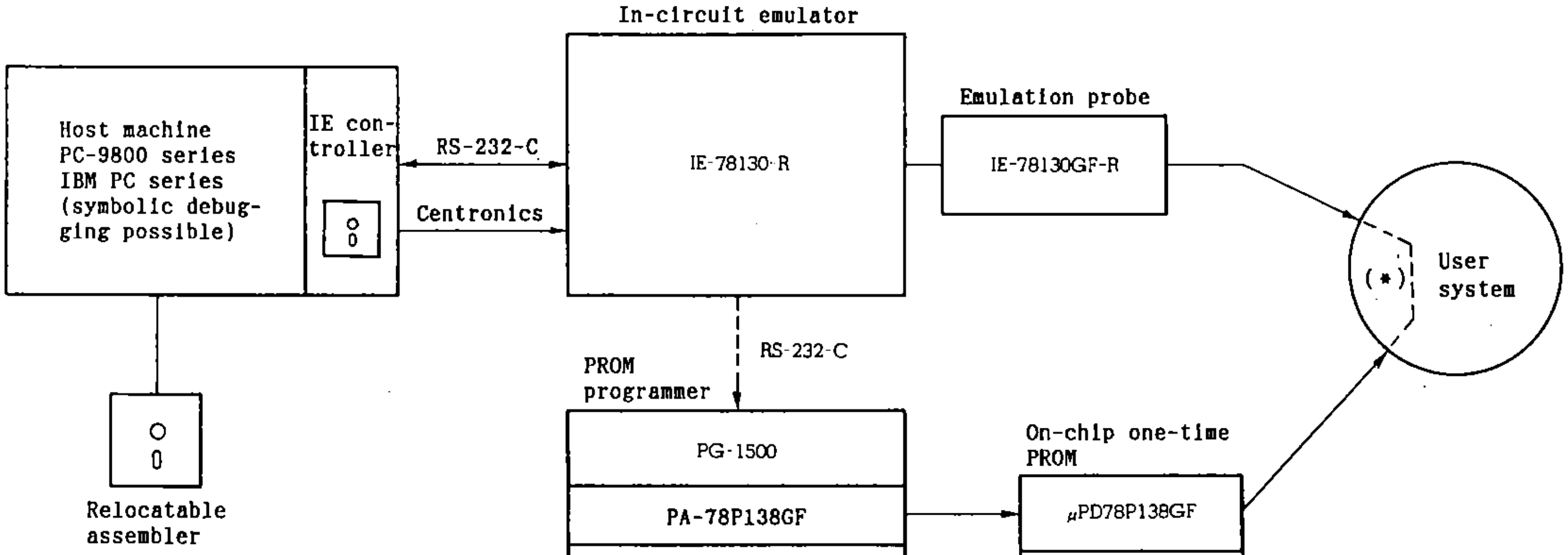
[Software]

RA78K/I relocatable assembler	This relocatable program can be used for all 78K/I series emulators. With its macro functions, it allows the user to improve program development efficiency. A structured-programming assembler is also provided, which enables explicit description of program control structures. This assembler could improve productivity in program production and maintenance.			
	Host machine	OS	Distribution media	Part number
	PC-9800 series	MS-DOS^{TM} (Ver.3.10 to Ver.5.00A$^{(*)}$)	3.5-inch 2HD	uS5A13RA78K1
			5-inch 2HD	uS5A10RA78K1
	IBM PC series	PC DOS^{TM} (Ver.3.1)	5-inch 2HC	uS7B10RA78K1
IE-78130-R control program (IE controller)	This program allows the user to control the IE-78130-R from the host machine.			
	Host machine	OS	Distribution media	Part number
	PC-9800 series	MS-DOS (Ver.3.10 to Ver.5.00A$^{(*)}$)	3.5-inch 2HD	uS5A13IE78130
			5-inch 2HD	uS5A10IE78130
	IBM PC series	PC DOS (Ver.3.1)	5-inch 2HC	uS7B10IE78130
PG-1500 controller	This program enables the host machine to control the PG-1500 under the serial intrreface and parallel interface.			
	Host machine	OS	Distribution media	Part number
	PC-9800 series	MS-DOS (Ver.3.10 to Ver.5.00A$^{(*)}$)	3.5-inch 2HD	uS5A13PG1500
			5-inch 2HD	uS5A10PG1500
	IBM PC series	PC DOS (Ver.3.1)	5-inch 2HC	uS7B10PG1500

* In version 5.00/5.00A, the task swapping function is disabled.

Remark: IE controller and assembler operations are guaranteed only on the host machine and by the OS mentioned above.

Configuration of Development Tools



* EV-9200G-80

APPENDIX B INDEX OF INSTRUCTIONS

Instruction	Page	Instruction	Page
ADD A, saddr	16-78	BC $addr16	16-166
ADD A, sfr	16-78	BE $addr16	16-168
ADD A, #byte	16-76	BF A.bit, $addr16	16-177
ADD A, [DE]	16-80	BF PSW.bit, $addr16	16-179
ADD A, [HL]	16-80	BF saddr.bit, $addr16	16-175
ADD A, [r4]	16-79	BF sfr.bit, $addr16	16-176
ADD A, word [r1]	16-81	BF X.bit, $addr16	16-178
ADD r, r'	16-77	BL $addr16	16-166
ADD saddr, #byte	16-76	BNC $addr16	16-167
ADD sfr, #byte	16-77	BNE $addr16	16-169
ADDC A, saddr	16-83	BNL $addr16	16-167
ADDC A, sfr	16-84	BNZ $addr16	16-169
ADDC A, #byte	16-81	BR rp	16-164
ADDC A, [DE]	16-85	BR !addr16	16-164
ADDC A, [HL]	16-85	BR $addr16	16-165
ADDC A, [r4]	16-84	BT A.bit, $addr16	16-172
ADDC A, word [r1]	16-86	BT PSW.bit, $addr16	16-174
ADDC r, r'	16-83	BT saddr.bit, $addr16	16-170
ADDC saddr, #byte	16-82	BT sfr.bit, $addr16	16-171
ADDC sfr, #byte	16-82	BT X.bit, $addr16	16-173
ADDW AX, rp	16-117	BTCLR A.bit, $addr16	16-182
ADDW AX, saddrp	16-118	BTCLR PSW.bit, $addr16	16-184
ADDW AX, sfrp	16-118	BTCLR saddr.bit, $addr16	16-180
ADDW AX, #word	16-117	BTCLR sfr.bit, $addr16	16-181
ADJBA	16-135	BTCLR X.bit, $addr16	16-183
ADJBS	16-136	BZ $addr16	16-168
AND A, saddr	16-99	CALL !addr16	16-158
AND A, sfr	16-99	CALL rp	16-158
AND A, #byte	16-97	CALLF !addr11	16-159
AND A, [DE]	16-101	CALLT [addr5]	16-160
AND A, [HL]	16-100	CLR1 A.bit	16-154
AND A, [r4]	16-100	CLR1 CY	16-157
AND A, word [r1]	16-101	CLR1 PSW.bit	16-154
AND saddr, #byte	16-97	CLR1 saddr.bit	16-153
AND sfr, #byte	16-98	CLR1 sfr.bit	16-153
AND r, r'	16-98	CLR1 X.bit	16-154
AND1 CY, A.bit	16-142	CMP A, saddr	16-113
AND1 CY, PSW.bit	16-144	CMP A, sfr	16-114
AND1 CY, saddr.bit	16-140	CMP A, #byte	16-111
AND1 CY, sfr.bit	16-141	CMP A, [DE]	16-115
AND1 CY, X.bit	16-143	CMP A, [HL]	16-115
AND1 CY, /A.bit	16-142	CMP A, [r4]	16-114
AND1 CY, /PSW.bit	16-144	CMP A, word [r1]	16-116
AND1 CY, /saddr.bit	16-141	CMP r, r'	16-112
AND1 CY, /sfr.bit	16-142	CMP saddr, #byte	16-111
AND1 CY, /X.bit	16-143	CMP sfr, #byte	16-112

(to be continued)

(Cont'd)

Instruction		Page	Instruction		Page
CMPW	AX, rp	16-121	MOVW	saddrp, AX	16-75
CMPW	AX, saddrp	16-122	MOVW	saddrp, #word	16-73
CMPW	AX, sfrp	16-123	MOVW	sfrp, AX	16-75
CMPW	AX, #word	16-121	MOVW	sfrp, #word	16-74
DBNZ	r2, $addr16	16-185	MOVW	SP, AX	16-163
DBNZ	saddr, $addr16	16-186	MOVW	SP, #word	16-163
DEC	r	16-125	MOV1	A.bit, CY	16-139
DEC	saddr	16-126	MOV1	CY, A.bit	16-137
DECW	rp	16-126	MOV1	CY, PSW.bit	16-138
DI		16-188	MOV1	CY, saddr.bit	16-136
DIVUW	r	16-124	MOV1	CY, sfr.bit	16-137
EI		16-188	MOV1	CY, X.bit	16-138
INC	r	16-125	MOV1	PSW.bit, CY	16-140
INC	saddr	16-125	MOV1	saddr.bit, CY	16-138
INCW	rp	16-126	MOV1	sfr.bit, CY	16-139
MOV	A, PSW	16-70	MOV1	X.bit, CY	16-139
MOV	A, r	16-62	MULSW	r	16-123
MOV	A, saddr	16-62	MULUW	r	16-124
MOV	A, sfr	16-63	NOP		16-187
MOV	A, word [r1]	16-69	NOT1	A.bit	16-155
MOV	A, [DE]	16-66	NOT1	CY	16-157
MOV	A, [DE+]	16-67	NOT1	PSW.bit	16-156
MOV	A, [HL]	16-65	NOT1	saddr.bit	16-155
MOV	A, [HL+]	16-66	NOT1	sfr.bit	16-155
MOV	A, [r3]	16-64	NOT1	X.bit	16-156
MOV	A, !addr16	16-68	OR	A, saddr	16-103
MOV	PSW, A	16-70	OR	A, sfr	16-104
MOV	PSW, #byte	16-69	OR	A, #byte	16-101
MOV	r, r'	16-62	OR	A, [DE]	16-105
MOV	r, #byte	16-61	OR	A, [HL]	16-105
MOV	saddr, A	16-63	OR	A, [r4]	16-104
MOV	saddr, #byte	16-61	OR	A, word [r1]	16-105
MOV	sfr, A	16-64	OR	r, r'	16-103
MOV	sfr, #byte	16-61	OR	saddr, #byte	16-102
MOV	STBC, #byte	16-187	OR	sfr, #byte	16-102
MOV	word [r1], A	16-69	OR1	CY, A.bit	16-147
MOV	[DE], A	16-67	OR1	CY, PSW.bit	16-148
MOV	[DE+], A	16-67	OR1	CY, saddr.bit	16-145
MOV	[HL], A	16-65	OR1	CY, sfr.bit	16-146
MOV	[HL+], A	16-66	OR1	CY, X.bit	16-147
MOV	[r3], A	16-65	OR1	CY, /A.bit	16-147
MOV	!addr16, A	16-68	OR1	CY, /PSW, bit	16-149
MOVW	AX, saddrp	16-74	OR1	CY, /saddr.bit	16-145
MOVW	AX, sfrp	16-75	OR1	CY, /sfr.bit	16-146
MOVW	AX, SP	16-164	OR1	CY, /X.bit	16-148
MOVW	rp, rp'	16-74	PUSH	PSW	16-162
MOVW	rp, #word	16-73	PUSH	rp	16-162

(to be continued)

(Cont'd)

Instruction	Page	Instruction	Page
POP PSW	16-163	SUBC A, [DE]	16-96
POP rp	16-162	SUBC A, [HL]	16-95
RET	16-161	SUBC A, [r4]	16-95
RETI	16-161	SUBC A, word [r1]	16-96
ROL r, n	16-127	SUBC r, r'	16-93
ROLC r, n	16-129	SUBC saddr, #byte	16-92
ROL4 [r4]	16-133	SUBC sfr, #byte	16-93
ROR r, n	16-127	SUBW AX, rp	16-119
RORC r, n	16-128	SUBW AX, saddrp	16-120
ROR4 [r4]	16-132	SUBW AX, sfrp	16-120
SEL RBn	16-187	SUBW AX, #word	16-119
SET1 A.bit	16-152	XCH A, r	16-70
SET1 CY	16-156	XCH A, saddr	16-71
SET1 PSW.bit	16-153	XCH A, sfr	16-71
SET1 saddr.bit	16-151	XCH A, [DE]	16-72
SET1 sfr.bit	16-152	XCH A, [HL]	16-72
SET1 X.bit	16-152	XCH A, [r4]	16-71
SHL r, n	16-130	XCH A, word [r1]	16-72
SHLW rp, n	16-132	XOR A, saddr	16-108
SHR r, n	16-129	XOR A, sfr	16-108
SHRW rp, n	16-131	XOR A, #byte	16-106
SUB A, saddr	16-89	XOR A, [DE]	16-110
SUB A, sfr	16-89	XOR A, [HL]	16-109
SUB A, #byte	16-86	XOR A, [r4]	16-109
SUB A, [DE]	16-91	XOR A, word [r1]	16-110
SUB A, [HL]	16-90	XOR r, r'	16-107
SUB A, [r4]	16-90	XOR saddr, #byte	16-106
SUB A, word [r1]	16-91	XOR sfr, #byte	16-107
SUB r, r'	16-88	XOR1 CY, A.bit	16-150
SUB saddr, #byte	16-87	XOR1 CY, PSW.bit	16-151
SUB sfr, #byte	16-88	XOR1 CY, saddr.bit	16-149
SUBC A, saddr	16-94	XOR1 CY, sfr.bit	16-150
SUBC A, sfr	16-94	XOR1 CY, X.bit	16-151
SUBC A, #byte	16-92		

APPENDIX C INDEX OF REGISTERS

[A]

A/D conversion mode register (ADM) 9-8

A/D conversion result register (ADCR) 9-1

[C]

capture mode register (CPTM) 8-70, 8-81

clock output mode register (CLOM) 10-5

[E]

18-bit FRC capture register 2
(CPT2H, CPT2L) 8-74, 8-77

18-bit free running counter (FRC) 8-2, 8-6, 8-74

event counter compare register 0 (ECC0) 8-14

event counter compare register 1 (ECC1) 8-14

event counter (EC) 8-11, 8-14

event divider control register (EDVC) 8-67

external capture input mode register
(INTM1) 8-68, 11-8

external interrupt mode register (INTM0) 11-7

[I]

input control register (ICR) 8-23, 8-73

internal memory size change register (IMS) ... 15-4

interrupt mask register (MK0) 11-10, 11-13

interrupt request flag register (IF0) 11-10, 11-12

interrupt service mode register (ISM0) 11-11, 11-13

[M]

macro service mode register 11-25, 11-26

memory mapping register (MM) 3-30, 5-27
5-33, 5-39

[P]

port 0 (P00-P07) 5-4

port 0 buffer register (P0H, P0L) 6-2

port 0 mode register (PM0) 5-5

port 1 (P10-P17) 5-7

port 1 mode register (PM1) 5-9

port 2 (P20-P27) 5-12

port 3 (P30-P37) 5-16

port 3 mode control register (PMC3) 5-22

port 3 mode register (PM3) 5-21

port 4 (P40-P47) 5-26

port 5 (P50-P57) 5-31

port 5 mode register (PM5) 5-33

port 6 (P60-P67) 5-36

port 6 mode register (PM6) 5-39

port 7 (P70,P71) 5-42

port 7 mode register (PM7) 5-43

prescaler mode register 3 (PRM3) 8-69

priority specification flag register (PR0) ... 11-11, 11-14

PWM control register (PWMC) 8-103

PWM0 modulo register (PWM0) 8-2, 8-7, 8-95

PWM1 modulo register (PWM1) 8-2, 8-7, 8-95

[R]

real-time output port control register
(RTPC) 5-6, 6-5

register for optional pull-up resistor
(PUO) 5-10, 5-14, 5-24, 5-29, 5-35, 5-40, 5-44

[S]

serial bus interface control register
(SBIC) 7-8

serial shift register (SIO) 7-3

7-bit timer 3 capture register (CPT30) 8-39

7-bit timer 3 compare register (CR30) 8-39

7-bit timer register 3 (TM3) 8-2, 8-34, 8-39

16-bit FRC capture register 0 (CPT0) 8-74

16-bit FRC capture register 1 (CPT1) 8-74

16-bit FRC capture register 3 (CPT3) 8-74

16-bit timer 0 compare register 0 (CR00) 8-9

16-bit timer 0 compare register 1 (CR01) 8-9

16-bit timer 0 compare register 2 (CR02) 8-9

16-bit timer 1 capture register 2 (CR12) 8-36, 8-44

16-bit timer 1 compare register 0 (CR10) 8-36, 8-44

16-bit timer 1 compare register 1 (CR11) 8-36, 8-44

16-bit timer 2 compare register (CR20) 8-92

16-bit timer register 0 (TM0) 8-2, 8-6, 8-9, 8-17

16-bit timer register 1 (TM1) 8-2, 8-6, 8-35, 8-44

16-bit timer register 2 (TM2) 8-2, 8-7, 8-92

standby control register (STBC) 12-3

synchronous serial interface mode register (CSIM) 7-5, 7-23

[T]

timer 0 output control register (TOC0) 8-27

timer 0 output mode register (TOM0) 8-25

timer 1 output control register (TOC1) 8-72

timer 1 output mode register (TOM1) 8-72

timer control register 0 (TMC0) 8-24, 8-71

timer control register 1 (TMC1) 8-80, 8-94

APPENDIX D INDEX OF REGISTER ABBREVIATIONS

[A]

ADCR 9-1

ADM 9-8

[C]

CLOM 10-5

CPTM 8-70, 8-81

CPT0 8-74

CPT1 8-74

CPT2H 8-74, 8-77

CPT2L 8-74, 8-77

CPT3 8-74

CPT30 8-39

CR00 8-9

CR01 8-9

CR02 8-9

CR10 .... 8-36, 8-44

CR11 .... 8-36, 8-44

CR12 .... 8-36, 8-44

CR20 8-92

CR30 8-39

CSIM 7-5, 7-23

[E]

EC 8-11, 8-14

ECC0 8-14

ECC1 8-14

EDVC 8-67

[F]

FRC 8-2, 8-6
8-74

[I]

ICR 8-23, 8-73

IF0 11-10,11-12

IMS 15-4

INTM0 11-7

INTM1 .... 8-68, 11-8

ISM0 .... 11-11, 11-13

[M]

MK0 .... 11-10, 11-13

MM 3-30, 5-28
5-33, 5-39

[P]

P0 5-4

P1 5-7

P2 5-12

P3 5-16

P4 5-26

P5 5-31

P6 5-36

P7 5-42

P0H............. 6-2

P0L............. 6-2

PM0............. 5-5

PM1............. 5-9

PM3 5-21

PM5 5-33

PM6 5-39

PM7 5-43

PMC3 5-22

PR0 .... 11-11, 11-14

PRM3 8-69

PUO 5-10, 5-14,
5-24, 5-29,
5-35, 5-40,
5-44

PWMC 8-103

PWM0 .... 8-2, 8-7,
8-95

PWM1 .... 8-2, 8-7,
8-95

[R]

RTPC 5-6, 6-5

[S]

SBIC 7-8

SIO 7-3

STBC 12-3

[T]

TM0 .... 8-2, 8-6,
8-9, 8-17

TM1 .... 8-2, 8-6,
8-35, 8-44

TM2 .... 8-2, 8-7,
8-92

TM3 .... 8-2, 8-34,
8-39

TMC0 .... 8-24, 8-71

TMC1 .... 8-80, 8-94

TOC0 8-27

TOC1 8-72

TOM0 8-25

TOM1 8-72