To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

RENESAS

MOS INTEGRATED CIRCUIT

μPD78C17(A),78C18(A)

8-BIT SINGLE-CHIP MICROCOMPUTER (WITH A/D CONVERTER)

Phase-out/Discontinued

The μPD78C18(A) is an 8-bit CMOS microcomputer which integrates 16-bit ALU, ROM, RAM, an A/D converter, a multi-function timer/event counter, and a general-purpose serial interface onto a single chip, and whose memory (ROM/RAM) is externally expandable up to 31 Kbytes. The μPD78C18(A) can operate at low power consumption because of its CMOS architecure and is provided with a standby function that enables data retention with an even lower power consumption.

The μPD78C17(A) is the ROM-less version of the μPD78C18(A). Its memory (ROM/RAM) is expandable externally up to 63 Kbytes.

A detailed explanation of the functions is provided in the user's manual listed below. It should be read before starting design work. 87AD Series μPD78C18 User's Manual: IEU-1314

FEATURES

- High reliability compared with μPD78C17 and μPD78C18
- 159 types of instructions: 87AD series instruction set plus multiply/divide instructions and 16-bit operation instructions
- Instruction cycle: 0.8 μs (at 15-MHz operation)
- Internal ROM: 32768 x 8 bits (μPD78C18(A) only)
- Internal RAM: 1024 x 8 bits
- Up to 64 Kbytes of memory (ROM/RAM) can be directly addressed
- High-resolution 8-bit A/D converter: 8 analog inputs
- General-purpose serial interface: Asynchronous, synchronous, I/O interface modes
- Multi-function 16-bit timer/event counter
- Two 8-bit timers
- I/O lines Input/output ports : 28 (μPD78C17(A)), 40 (μPD78C18(A))
  Edge detection inputs : 4
- 11 interrupt functions external : 3, internal: 8
  (Non-maskable: 1, maskable: 10)
- Zero-cross detection function: (2 inputs)
- Standby function: HALT mode, hardware/software STOP mode
- Mask option pull-up resistors can be incorporated into Ports A, B, and C: (μPD78C18(A) only)

ORDERING INFORMATION

Part Number	Package	Quality Grade
μPD78C17GF(A)-3BE	64-pin plastic QFP(14 x 20 mm)	Special
μPD78C17GQ(A)-36	64-pin plastic QUIP	Special
μPD78C18GF(A)-xxx-3BE	64-pin plastic QFP(14 x 20 mm)	Special
μPD78C18GQ(A)-xxx-36	64-pin plastic QUIP	Special

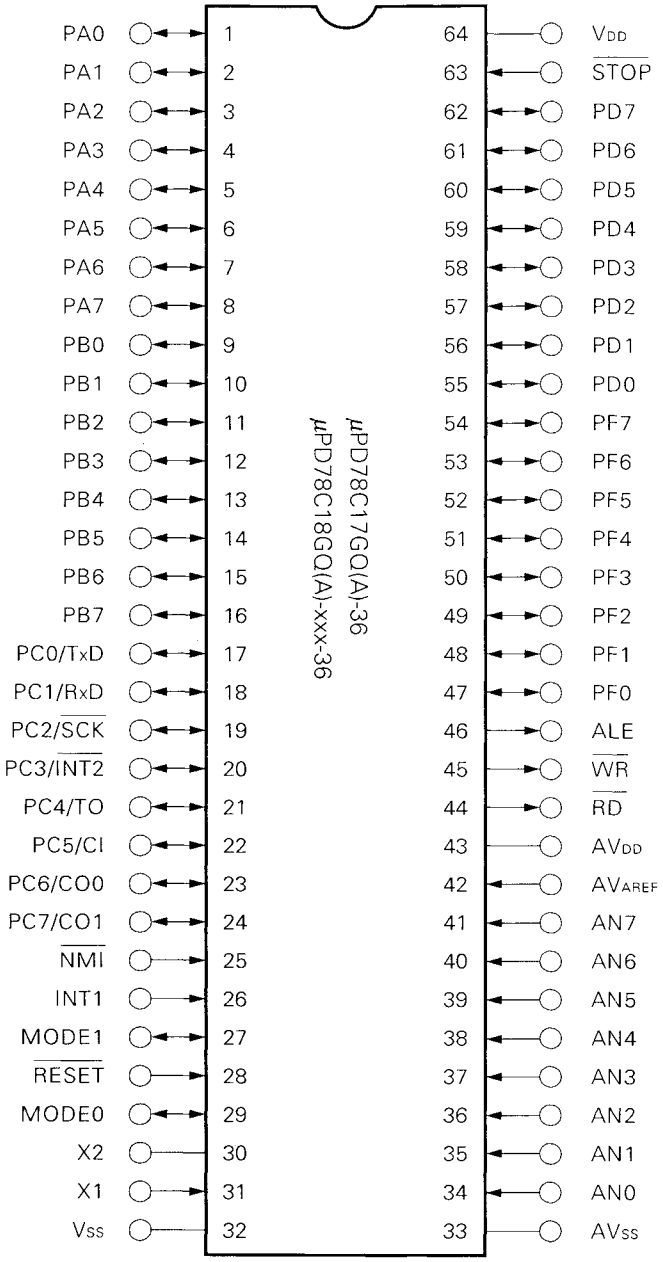
Please refer to "Quality grade on NEC semiconductor devices" (Document number IEI-1209) published by NEC corporation to know the specification of quality grade on the devices and its recommended applications.

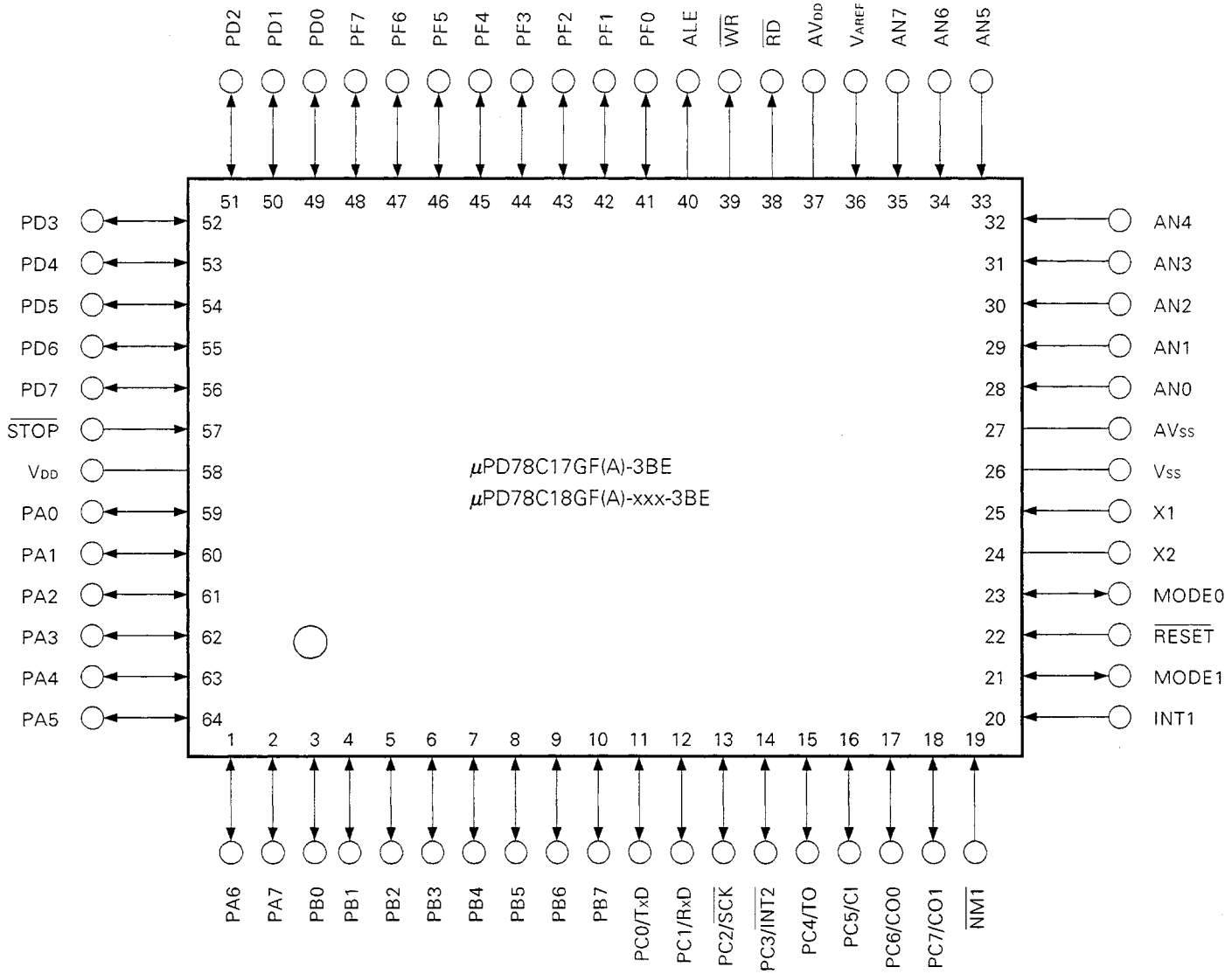
The information in this document is subject to change without notice.

Document No. IC-3127A
(O. D. No. IC-8624A)
Date Published April 1995 P
Printed in Japan

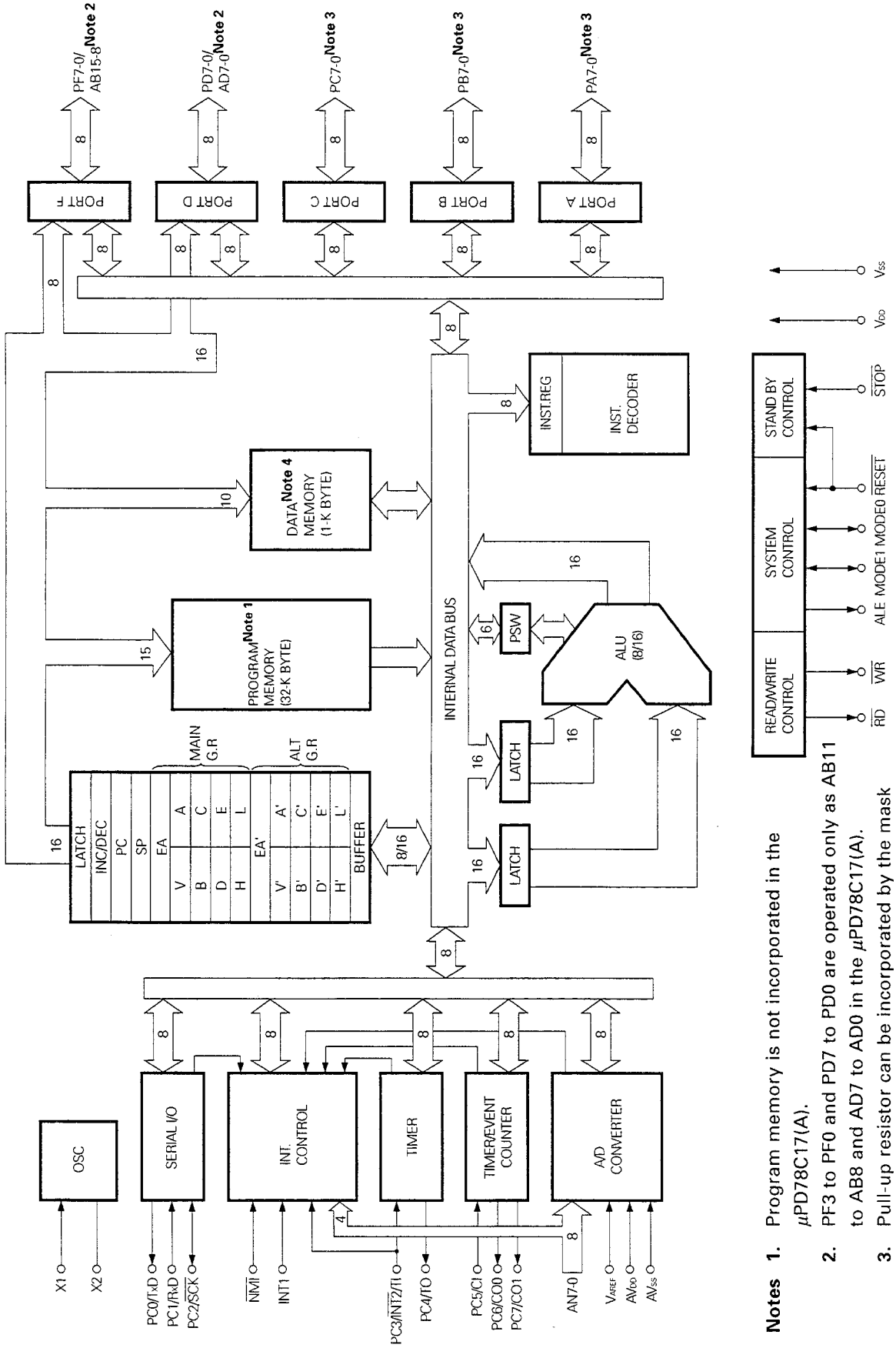
The mark ★ shows revised points.

PIN CONFIGURATION (TOP VIEW)




PD2 PD1 PD0 PF7 PF6 PF5 PF4 PF3 PF2 PF1 PF0 ALE WR RD AVDD VAREF AN7 AN6 AN5
51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
PD3 52
PD4 53
PD5 54
PD6 55
PD7 56
STOP 57
VDD 58
PA0 59
PA1 60
PA2 61
PA3 62
PA4 63
PA5 64
32 AN4
31 AN3
30 AN2
29 AN1
28 AN0
27 AVSS
26 VSS
25 X1
24 X2
23 MODE0
22 RESET
21 MODE1
20 INT1
μPD78C17GF(A)-3BE
μPD78C18GF(A)-xxx-3BE
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
PA6 PA7 PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7 PC0/TxD PC1/RxD PC2/SCK PC3/INT2 PC4/TO PC5/CI PC6/CO0 PC7/CO1 NMI

BLOCK DIAGRAM



Notes 1. Program memory is not incorporated in the μPD78C17(A).

2. PF3 to PF0 and PD7 to PD0 are operated only as AB11 to AB8 and AD7 to AD0 in the μPD78C17(A).

3. Pull-up resistor can be incorporated by the mask option in the μPD78C18(A).

4. Can be used only when RAE bit of MM register is 1. When it is 0, an external memory is necessary.

DIFFERENCE BETWEEN μPD78C17(A), 78C18(A) AND μPD78C17, 78C18

Part Number / Item	**μPD78C17(A), 78C18(A)**	μPD78C17, 78C18
Quality grade	Special	Standard
Electrical specifications	Input leakage current AN7 to AN0: ±1 μA (MAX.)	Input leakage current AN7 to AN0; ±10 μA (MAX.)
Package	• 64-pin plastic QFP (14 x 20 mm) • 64-pin plastic QUIP	• 64-pin plastic shrink DIP (750 mil) • 64-pin plastic QFP (14 x 20 mm) • 64-pin plastic QUIP

CONTENTS

1. PIN FUNCTIONS

1.1 LIST OF PIN FUNCTION (1/2)

Pin Name	I/O	Function	
PA7 to PA0 (Port A)	Input-output	8-bit input-output port, which can specify input/output (Port A) bit-wise.	
PB7 to PB0 (Port B)	Input-output	8-bit input-output port, which can specify input/output (Port B) bit-wise.	
PC0/TxD	Input-output/ Output	Port C 8-bit input-output port, which can specify input/output bit-wise.	Transmit Data Output pin for serial data.
PC1/RxD	Input-output/ Input		Receive Data Input pin for serial data.
PC2/$\overline{SCK}$	Input-output/ Input-output		Serial Clock Input-output pin for serial clock. It becomes output clock for the internal clock use, and input for the external.
PC3/$\overline{INT2}$/TI	Input-output/ Input/Input		Interrupt Request/Timer Input Maskable interrut input pin of the edge trigger (falling edge), or an external clock input pin for a timer. Also, it can be used as a zero-cross detection pin for AC input.
PC4/TO	Input-output/ Output		Timer Output Square wave defining one cycle of internal clock or timer counter time as half cycle is output.
PC5/CI	Input-output/ Input		Counter Input External pulse input pin to timer/event counter.
PC6/CO0 PC7/CO1	Input-output/ Output		Counter Output 0, 1 Programmable square wave output by timer/event counter.
PD7 to PD0/ AD7 to AD0	Input-output/ Input-output	Port D 8-bit input-output port, which can specify input/output in byte units (μPD78C18(A)).	Address/Data Bus When external memory is used, it becomes multiplexed address/data bus.
PF7 to PF0/ AB15 to AB8	Input-output/ Output	Port F 8-bit input-output port, which can specify input/output bit-wise.	Address Bus When external memory is used, it becomes address bus.
$\overline{WR}$ (Write Strobe)	Output	Strobe signal which is output for write operation of external memory. It becomes high in any cycle other than the data write machine cycle of external memory. When $\overline{RESET}$ signal is either low or in the hardware STOP mode, this signal becomes output high-impedance.	
$\overline{RD}$ (Read Strobe)	Output	Strobe signal which is output for read operation of external memory. It becomes high in any cycle other than the read machine cycle of external memory. When $\overline{RESET}$ signal is either low or in the hardware STOP mode, this signal becomes output high-impedance.	
ALE (Address Latch Enable)	Output	Strobe signal to latch externally the lower address information which is output to PD7 to PD0 pins to access external memory. When $\overline{RESET}$ signal is either low or in the hardware STOP mode, this signal becomes output high-impedance.	

1.1 LIST OF PIN FUNCTION (2/2)

Pin Name	I/O	Function
MODE0 MODE1 (Mode)	Input-output	The μPD78C18(A) sets MODE0 pin to "0" (low level), and MODE1 pin to "1" (high level)[Note] The μPD78C17(A) allows you to set MODE0, MODE1 pins to select 4 K, 16 K, or 63 Kbytes for the size of the memory which is installed externally. MODE0 | MODE1 | External Memory 0 | 0 | 4K bytes 1 | 0 | 16K bytes 1 | 1 | 63K bytes Also, when each of MODE0 and MODE1 pins is set to "1"[Note], it is synchronized to ALE to output a control signal.
$\overline{NMI}$ (Non-Maskable Interrupt)	Input	Non-maskable interrupt input pin of the edge trigger (falling edge)
INT1 (Interrupt Request)	Input	A maskable interrupt input pin of the edge trigger (rising edge). Also, it can be used as a zero-cross detection pin for AC input.
AN7 to AN0 (Analog Input)	Input	8 pins of analog input to A/D converter. AN7 to AN4 can be used as edge detection (falling edge) input.
V_{AREF} (Reference Voltage)	Input	A common pin serving both as a reference voltage input pin for A/D converter and as a control pin for A/D converter operation.
AV_{DD} (Analog V_{DD})		Power supply pin for A/D converter.
AV_{SS} (Analog V_{SS})		GND pin for A/D converter.
X1, X2 (Crystal)		Crystal connection pins for system clock oscillation. X1 should be input when a clock is supplied from outside. Inverted clock of X1 should be input in X2.
$\overline{RESET}$ (Reset)	Input	Low-level active system reset input.
$\overline{STOP}$ (Stop)	Input	Control signal input pin in hardware STOP mode. The oscillation stops when the low-level is input.
V_{DD}		Positive power supply pin.
V_{SS}		GND pin.

Note Pull-up. Pull-up resistor R is $4\ [k\Omega] \leq R \leq 0.4\ t_{CYC}\ [k\Omega]$ (t_{CYC} is ns unit).

Remark The μPD78C18(A) can incorporate (mask option) pull-up resistors on to ports A, B, and C.

1.2 PIN INPUT/OUTPUT CIRCUITS

Table 1-1 and 1-2, and figures (1) to (15) show input/output circuits of each pin in a schematic form.

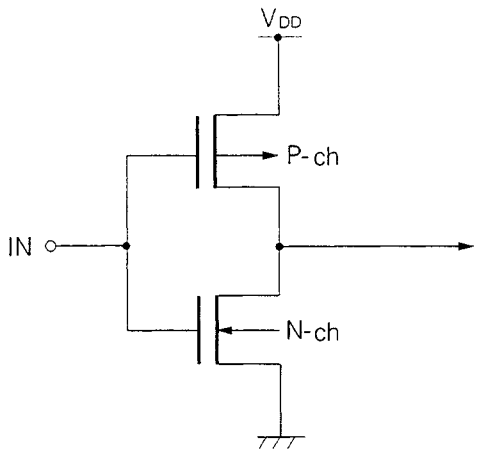
Table 1-1 Pin Type No. for μPD78C17(A)

Pin Name	Type No.	Pin Name	Type No.
PA7 to PA0	5	$\overline{\text{RESET}}$	2
PB7 to PB0	5	$\overline{\text{RD}}$	4
PC1 and PC0	5	$\overline{\text{WR}}$	4
PC2/$\overline{\text{SCK}}$	8	ALE	4
PC3/$\overline{\text{INT2}}$	10	$\overline{\text{STOP}}$	2
PC7 to PC4	5	MODE0	11
AD7 to AD0	5	MODE1	11
AB11 to AB8	5	AN3 to AN0	7
PF7 to PF4	5	AN7 to AN4	12
$\overline{\text{NMI}}$	2	V_{AREF}	13
INT1	9		

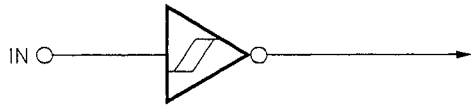
Table 1-2 Pin Type No. for μPD78C18(A)

Pin Name	Type No.	Pin Name	Type No.
PA7 to PA0	5-A	$\overline{\text{RESET}}$	2
PB7 to PB0	5-A	$\overline{\text{RD}}$	4
PC1 and PC0	5-A	$\overline{\text{WR}}$	4
PC2/$\overline{\text{SCK}}$	8-A	ALE	4
PC3/$\overline{\text{INT2}}$	10-A	$\overline{\text{STOP}}$	2
PC7 to PC4	5-A	MODE0	11
PD7 to PD0	5	MODE1	11
PF7 to PF0	5	AN3 to AN0	7
$\overline{\text{NMI}}$	2	AN7 to AN4	12
INT1	9	VAREF	13

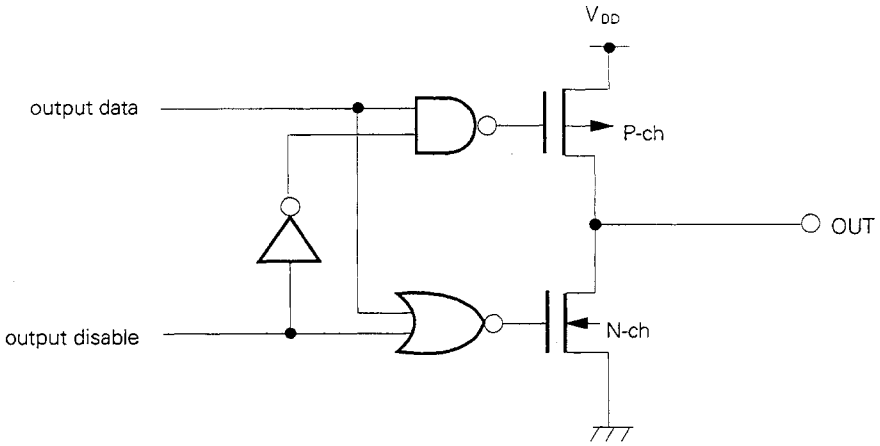
(1) Type 1



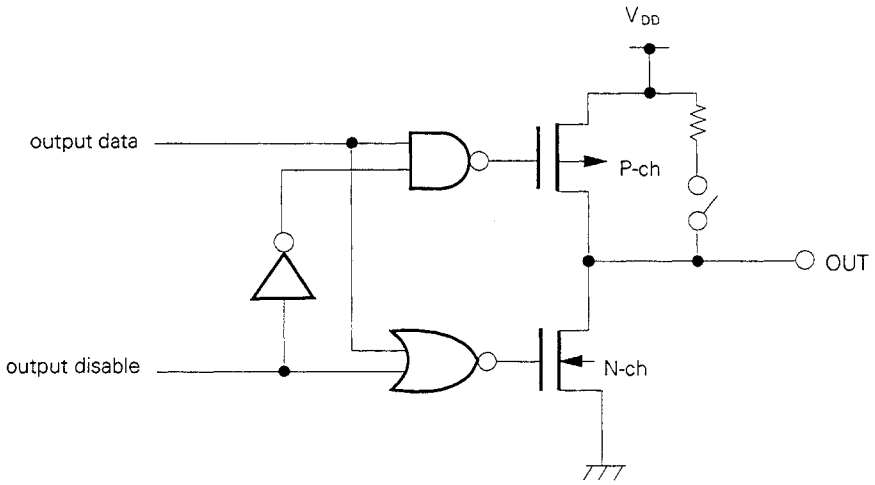
(2) Type 2



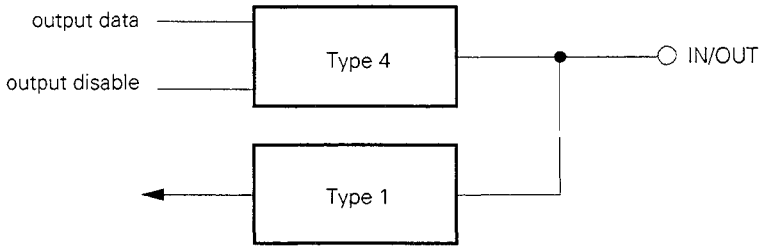
(3) Type 4



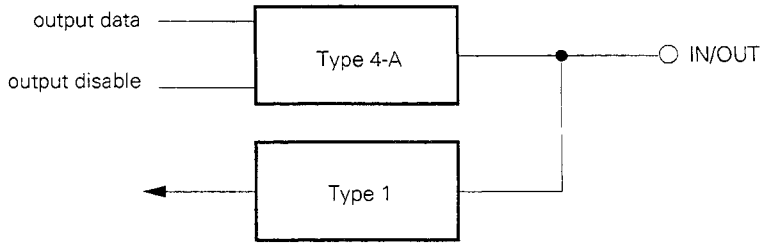
(4) Type 4-A



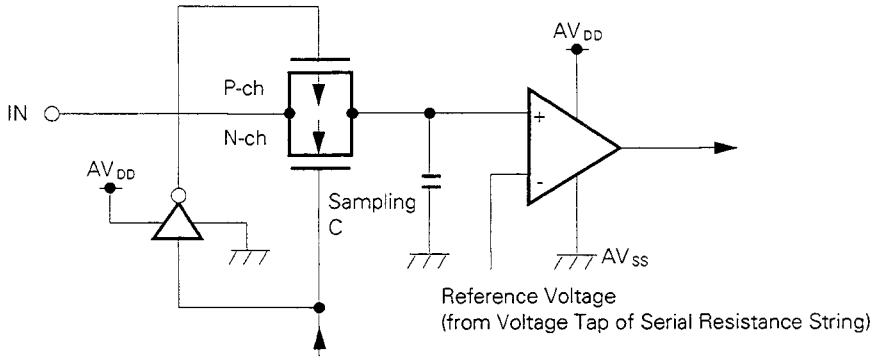
(5) Type 5



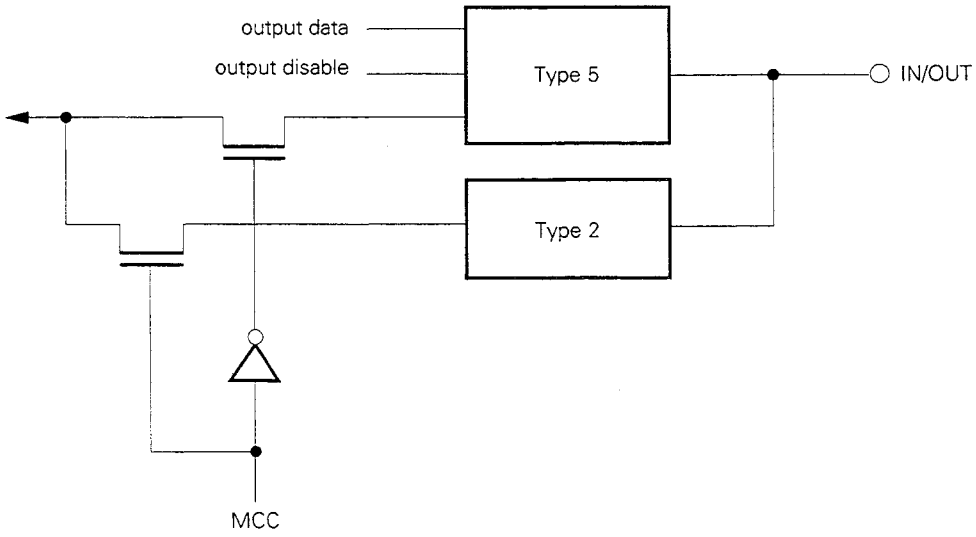
(6) Type 5-A



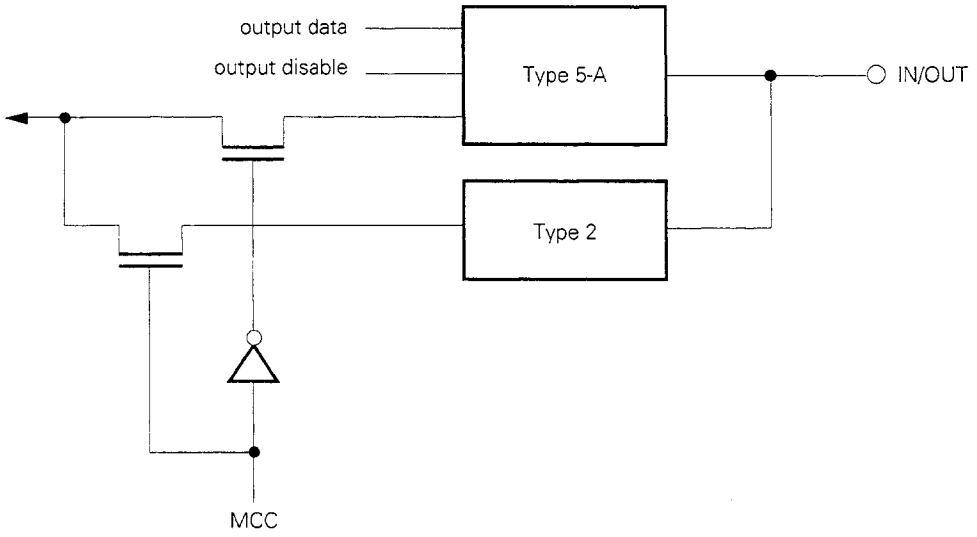
(7) Type 7



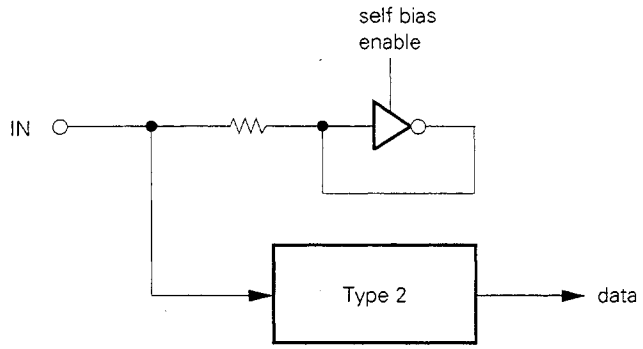
(8) Type 8



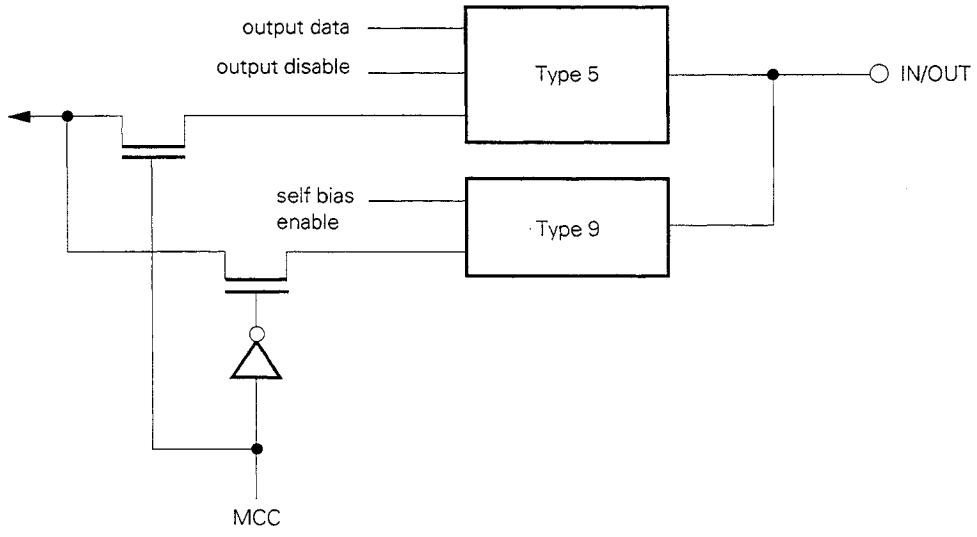
(9) Type 8-A



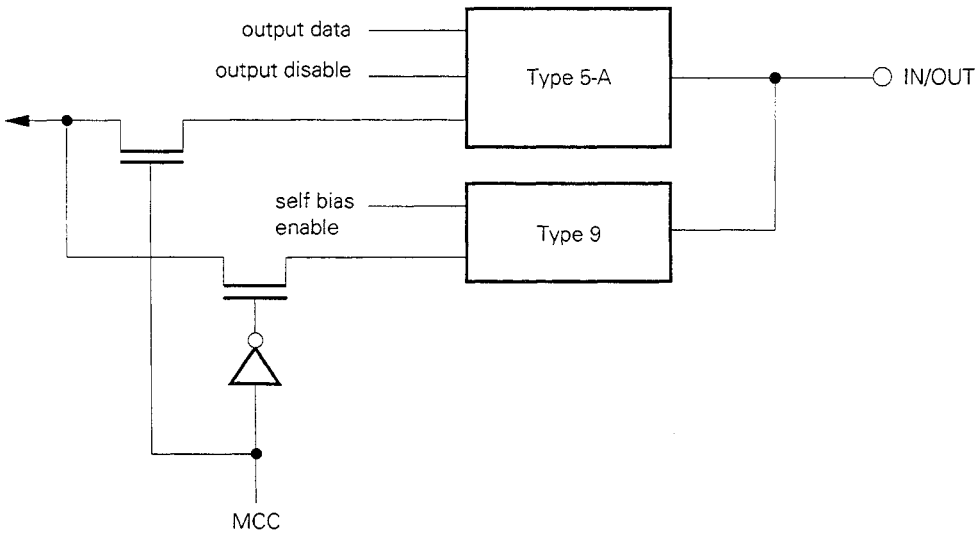
(10) Type 9



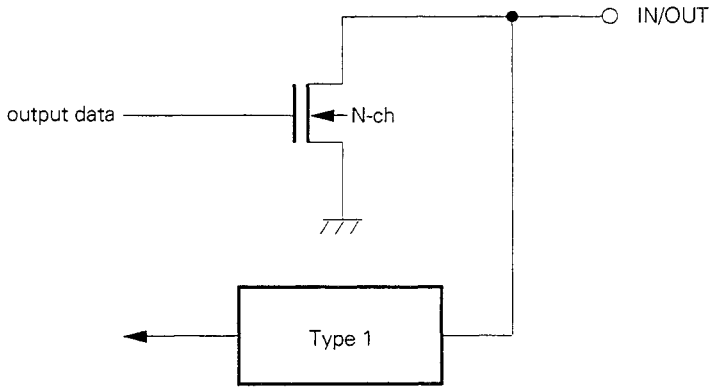
(11) Type 10



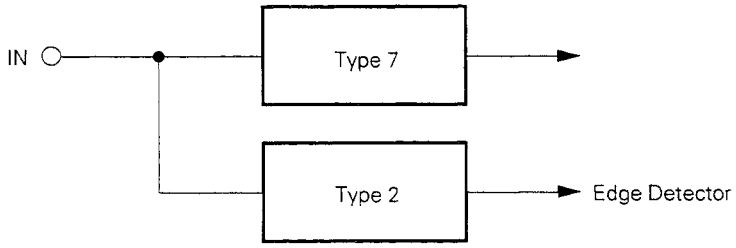
(12) Type 10-A



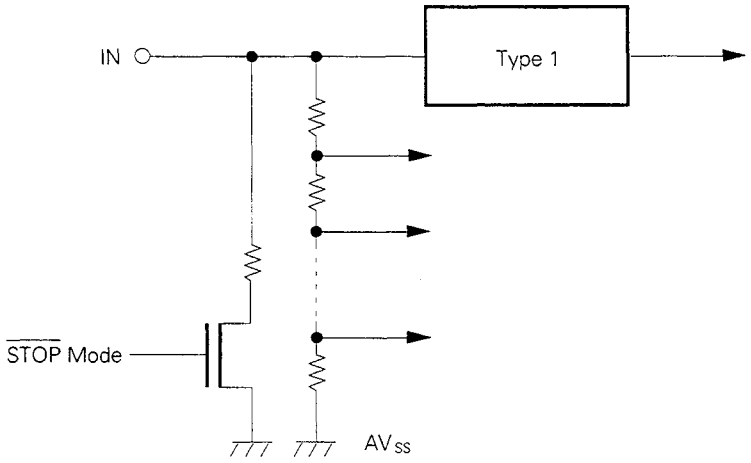
(13) Type 11



(14) Type 12



(15) Type 13



1.3 PIN MASK OPTIONS

The μPD78C18(A) has the following mask options, which can be selected bit-wise according to the application.

Pin Name	Mask Options
PA7 to PA0	Pull-up resistor can be incorporated
PB7 to PB0	
PC7 to PC0	

★ **Cautions 1. Zero-cross function will not operate properly if pull-up resistor is incorporated in PC3.**
2. The μPD78C17(A) has no mask option.

1.4 UNUSED PIN CONNECTIONS

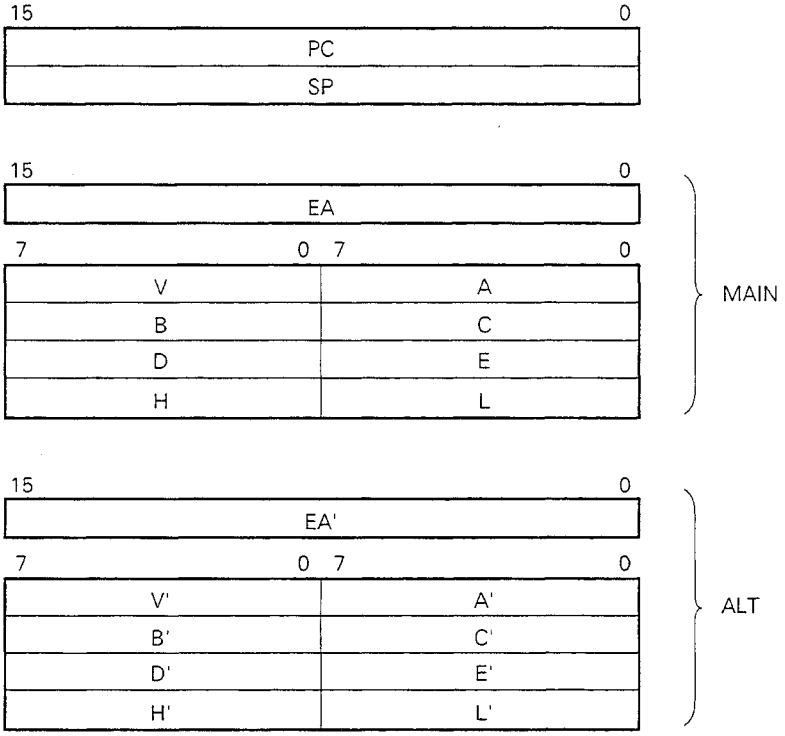
Pin	Recommended Connection
PA7 to PA0 PB7 to PB0 PC7 to PC0 PD7 to PD0 PF7 to PF0	Connect to V_{SS} or V_{DD} via resistor
$\overline{RD}$ $\overline{WR}$ ALE	Leave open
$\overline{STOP}$	Connect to V_{DD}
INT1, $\overline{NMI}$	Connect to V_{SS} or V_{DD}
AV_{DD}	Connect to V_{DD}
AV_{AREF} AV_{SS}	Connect to V_{SS}
AN7 to AN0	Connect to AV_{SS} or AV_{DD}

2. INTERNAL BLOCK FUNCTIONS

2.1 REGISTERS

The central registers are the sixteen 8-bit registers and four 16-bit registers shown in Fig. 2-1.

Fig. 2-1 Register Configuration



(a) General registers (B, C, D, E, H, L)

There are two sets of general registers (MAIN: B, C, D, E, H, L; ALT: B', C', D', E', H', L'). They function as auxiliary registers for the accumulator, and have a data pointer function as register pairs (BC, DE, HL; B'C', D'E', H'L'). Four register pairs, DE, D'E', HL, and H'L' in particular, have a base register function.

When the two sets are used, if an interrupt occurs in one set, the register contents are saved into the other register set without saving them into the memory so that interrupt servicing can be carried out. The other set of registers can also be used as data pointer expansion registers. Single-step automatic increment/decrement modes and a two-step automatic increment addressing mode are available for the register pairs, DE, HL, D'E', and H'L', so that the processing time can be reduced. BC, DE, and HL can be simultaneously replaced with the ALT register by means of the EXX instruction. The HL register can be independently replaced with the ALT register by means of the EXH instruction.

(b) Working register vector register (V)

When a working area is set in the memory space, the high-order 8 bits of the memory address are selected using the V register and the low-order 8 bits are addressed by the immediate data in the instruction. Thus, the memory area specified with the V register can be used as working registers with a 256 x 8-bit configuration.

Because a working register can be specified with a 1-byte address field, program reduction is possible by using the working area for software flags, parameters and counters. The V register can be replaced with the ALT register paired with an accumulator by means of the EXA instruction.

(c) Accumulator (A)

In the μPD78C17(A)/78C18(A), because an accumulator type architecture is used, 8-bit data processing such as 8-bit arithmetic and logical operation instructions is mainly performed by this accumulator.

This accumulator can be replaced with the ALT register paired with the vector register (V) by means of the EXA instruction.

(d) Expansion accumulator (EA)

16-bit data processing such as 16-bit arithmetic and logical operation instructions is mainly performed by EA.

This accumulator can be replaced with the ALT register EA' by means of the EXA instruction.

(e) Program counter (PC)

This is a 16-bit register which holds information on the next program address to be executed. This register is normally incremented automatically according to the number of bytes of the instruction to be fetched. When an instruction associated with a branch is executed, immediate data or register contents are loaded. $\overline{\text{RESET}}$ input clears this counter to 0000H.

(f) Stack pointer (SP)

This is a 16-bit register which holds the start address of the memory stack area (LIFO format).

SP contents are decremented when a CALL or PUSH instruction is executed or an interrupt is generated, and incremented when a RETURN or POP instruction is executed.

2.2 ARITHMETIC LOGIC UNIT (ALU) ...16 BITS

The ALU executes data processing such as 8-bit arithmetic and logical operations, shift and rotation, data processing such as 16-bit arithmetic and logical operations and shift operations, 8-bit multiplication and 16-bit by 8-bit division.

2.3 PROGRAM STATUS WORD (PSW)

This word consists of 6 types of flags which are set/reset according to instruction execution results. Three of these flags (Z, HC, and CY) can be tested by an instruction. PSW contents are automatically saved to the stack when an interrupt (external, internal, or SOFTI instruction) is generated, and restored by the RETI instruction. $\overline{\text{RESET}}$ input resets all bits to (0).

Fig. 2-2 PSW Configuration

7	6	5	4	3	2	1	0
0	Z	SK	HC	L1	L0	0	CY

(a) Z (Zero)

When the operation result is zero, this flag is set (1). In all other cases, it is reset (0).

(b) SK (Skip)

When the skip condition is satisfied, this flag is set (1). If the condition is not satisfied, it is reset (0).

(c) HC (Half Carry)

If an 8-bit operation generates a carry out of bit 3 or a borrow into bit 3, this flag is set (1). In all other cases, it is reset (0).

(d) L1

When the "MVI A, byte" instruction is stacked, this flag is set (1). In all other cases, it is reset (0).

(e) L0

When the "MVI L, byte;LXI H, word" instruction is stacked, this flag is set (1). In all other cases, it is reset (0).

(f) CY (Carry)

When a 16-bit operation generates a carry out of or a borrow into bit 7 or 15, this flag is set (1). In all other cases, it is reset (0).

When one of 35 types of ALU instructions, rotation instructions, or carry manipulation instructions is executed, various flags are affected as shown in Table 2-1.

Table 2-1 Flag Operations

Operation						D6	D5	D4	D3	D2	D0
reg. memory			immediate		skip	Z	SK	HC	L1	L0	CY
ADD ADC SUB SBB DADD DADC DSUB DSBB EADD ESUB	ADDW ADCW SUBW SBBW	ADDX ADCX SUBX SBBX	ADI ACI SUI SBI			↕	0	↕	0	0	↕
ANA ORA XRA DAN DOR DXR	ANAW ORAW XRAW	ANAX ORAX XRAX	ANI ORI XRI	ANIW ORIW		↕	0	●	0	0	●
ADDNC SUBNB GTA LTA DADDNC DSUBNB DGT DLT	ADDNCW SUBNBW GTAW LTAW	ADDNCX SUBNBX GTAX LTAX	ADINC SUINB GTI LTI	GTIW LTIW		↕	↕	↕	0	0	↕
ONA OFFA DON DOFF	ONAW OFFAW	ONAX OFFAX	ONI OFFI	ONIW OFFIW		↕	↕	●	0	0	●
NEA EQA DNE DEQ	NEAW EQAW	NEAX EQAX	NEI EQI	NEIW EQIW		↕	↕	↕	0	0	↕
INR DCR	INRW DCRW					↕	↕	↕	0	0	●
DAA						↕	0	↕	0	0	↕
RLR RLL SLR SLL DRLR DRLL DSLR DSLL						●	0	●	0	0	↕
SLRC SLLC						●	↕	●	0	0	↕
STC						●	0	●	0	0	1
CLC						●	0	●	0	0	0
			MVI A, byte			●	0	●	1	0	●
			MVI L, byte LXI H, word			●	0	●	0	1	●
					BIT SK SKN SKIT SKNIT	●	↕	●	0	0	●
					RETS	●	1	●	0	0	●
Other All instructions						●	0	●	0	0	●

↕ ..... Affected (Set or Reset)
1 ..... Set
0 ..... Reset
● ..... Not affected

2.4 MEMORY

The μPD78C17(A)/78C18(A) can address a maximum of 64 Kbytes of memory. The memory maps are shown in Figs. 2-3 and 2-4. The external memory area and the internal RAM area can be freely used as program memory and data memory. Because the access timing for internal memory and external memory are the same, processing can be executed at high speeds.

(a) Interrupt start addresses

The interrupt start addresses are all fixed as follows:

$\overline{NMI}$ 0004H
INTT0/INTT1 .....0008H
INT1/$\overline{INT2}$ 0010H
INTE0/INTE1 .....0018H
INTEIN/INTAD 0020H
INTSR/INTST 0028H
SOFTI 0060H

(b) Call address table

The call address of a 1-byte call instruction (CALT) can be stored in the 64-byte area (for 32 call addresses) from address 0080H to address 00BFH.

(c) Specific memory area

The reset start address, interrupt start addresses, and the call table are allocated to addresses 0000H to 00BFH, and this area takes account of these in use. Addresses 0800H to 0FFFH are directly addressable by a 2-byte call instruction (CALF).

The μPD78C18(A) has on-chip mask programmable ROM in addresses 0000H to 7FFFH.

(d) Internal data memory area

1K-byte RAM is incorporated in addresses FC00H to FFFFH. The RAM contents are retained for 1K-byte internal data memory area in standby operation.

(e) External memory area

With the μPD78C17(A), the external memory can be expanded in steps in 63K-byte area (0000H to FBFFH) by setting the MODE0 and MODE1 pins (see **Table 2-3**).

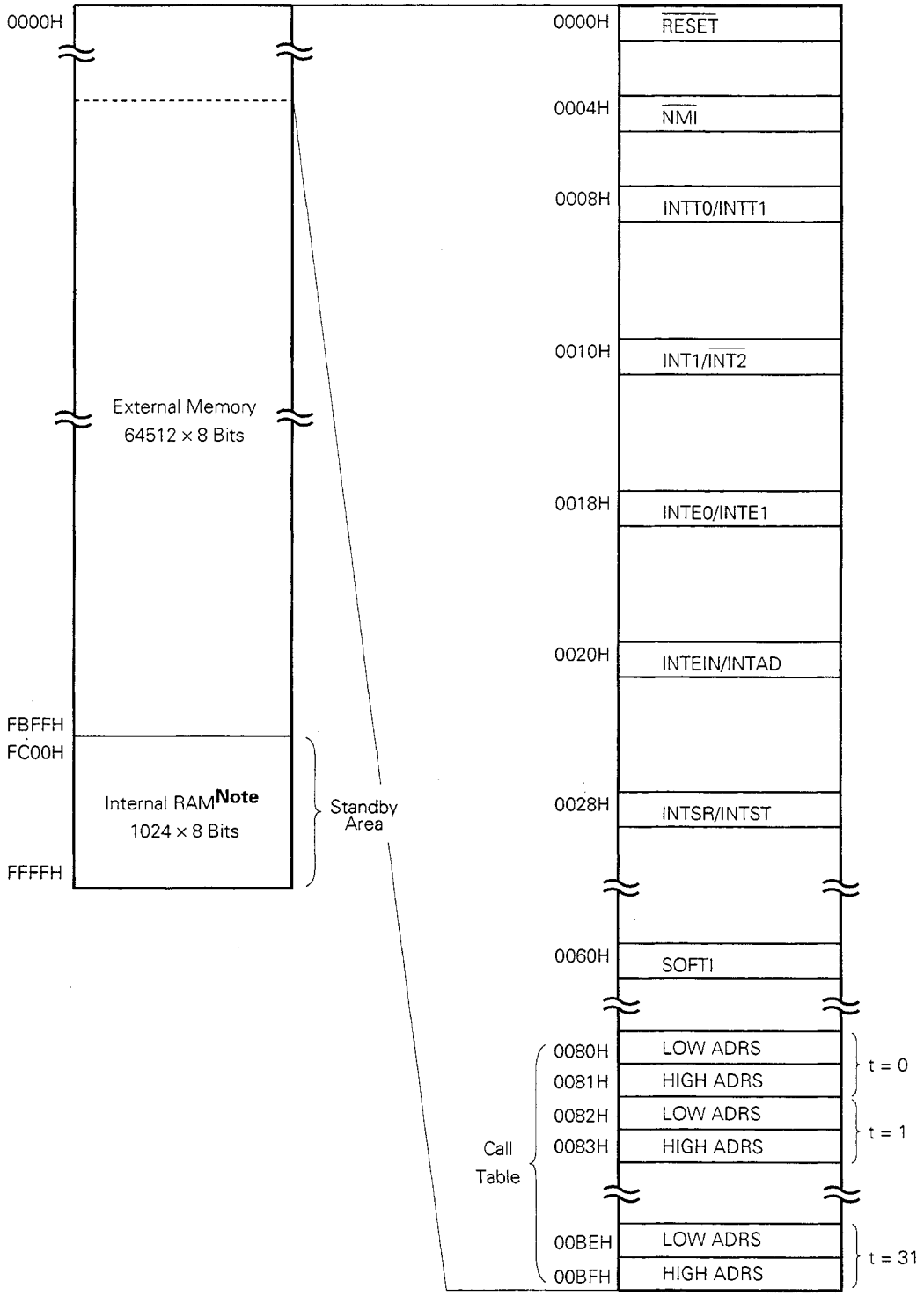
With the μPD78C18(A), the external memory can be expanded in steps in 31K-byte area (8000H to FBFFH) by setting the MEMORY MAPPING register (see **Fig. 2-13**).

The external memory is accessed using AD7 to AD0 (multiplexed address/data bus), AB7 to AB0 (address bus), and the $\overline{RD}$, $\overline{WR}$, and ALE signals. Both programs and data can be stored in the external memory.

(f) Working register area

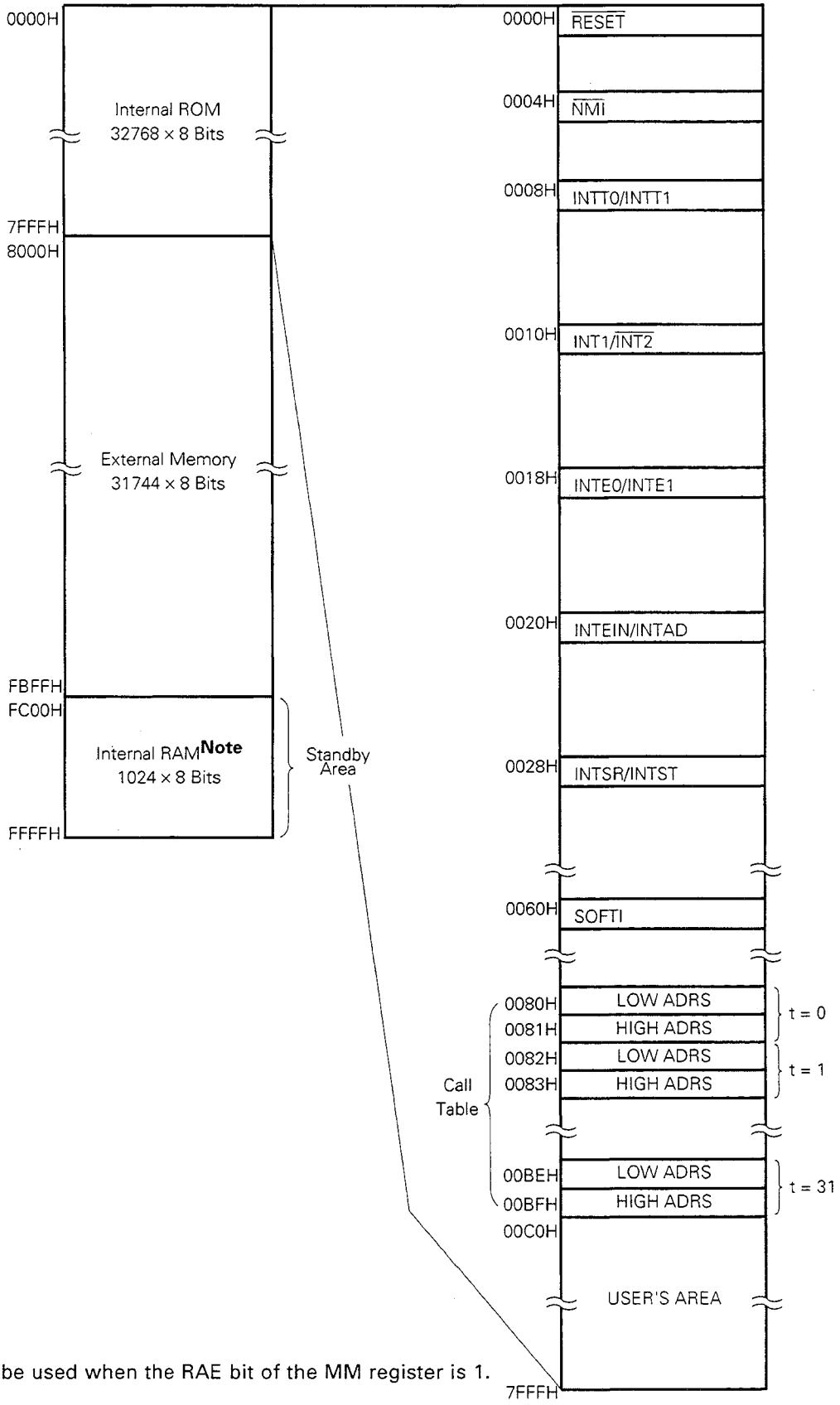
A 256-byte working register area can be set in any memory location (specified by the V register) and working register addressing is possible.

Fig. 2-3 μPD78C17(A) Memory Map



Note Can only be used when the RAE bit of the MM register is 1.

Fig. 2-4 μPD78C18(A) Memory Map



Note Can only be used when the RAE bit of the MM register is 1.

2.5 PORT FUNCTIONS

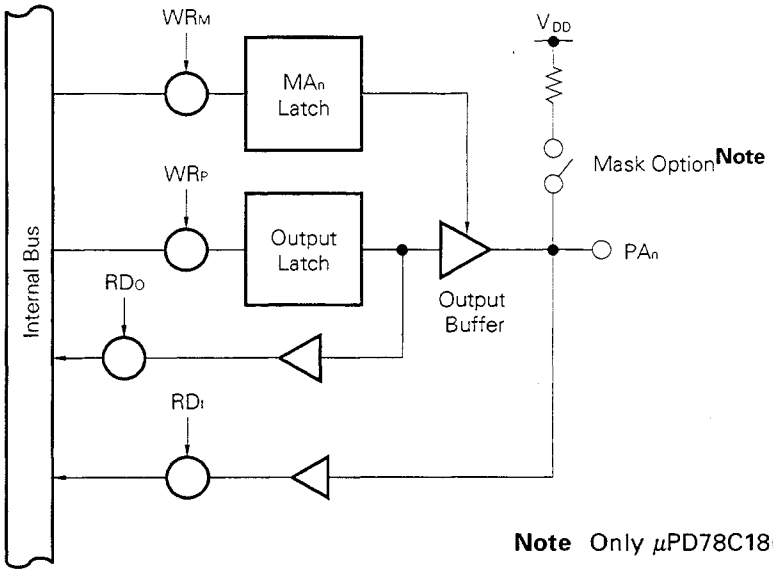
(1) PA7 to PA0 (PORT A)

This is an 8-bit input/output port which has input/output buffer and output latch functions. Port A can be set as to input or output bit-wise using the MODE A register. And μPD78C18(A) port A pull-up resistor specification is performed bit-wise by mask option.

Port A is set as follows when setting the input port or after reset.

High-impedance : Without pull-up resistor
High level : With pull-up resistor

Fig. 2-5 Port A

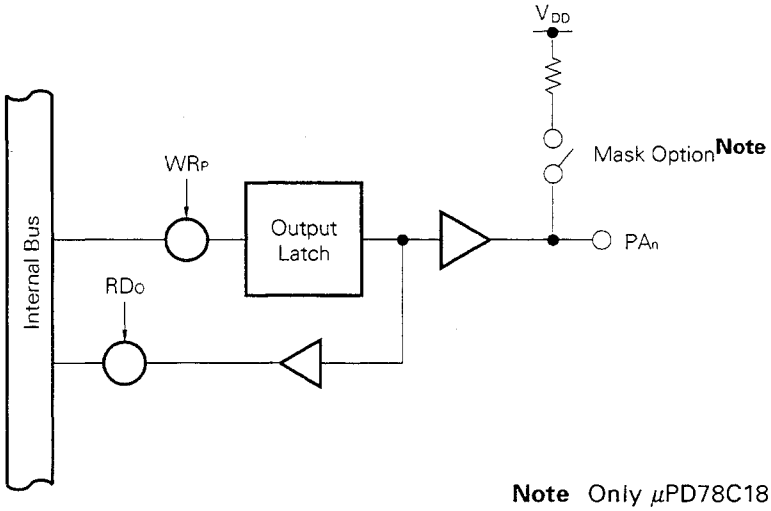


Note Only μPD78C18(A)

(a) When specified as output port (MAn = 0)

The output latch is effective, enabling data exchange by a transfer instruction between the output latch and the accumulator. Direct bit setting/resetting of output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Once data is written to the output latch, the data is held until a port A manipulation instruction is executed or the data is reset.

Fig. 2-6 Port A Specified as Output Port

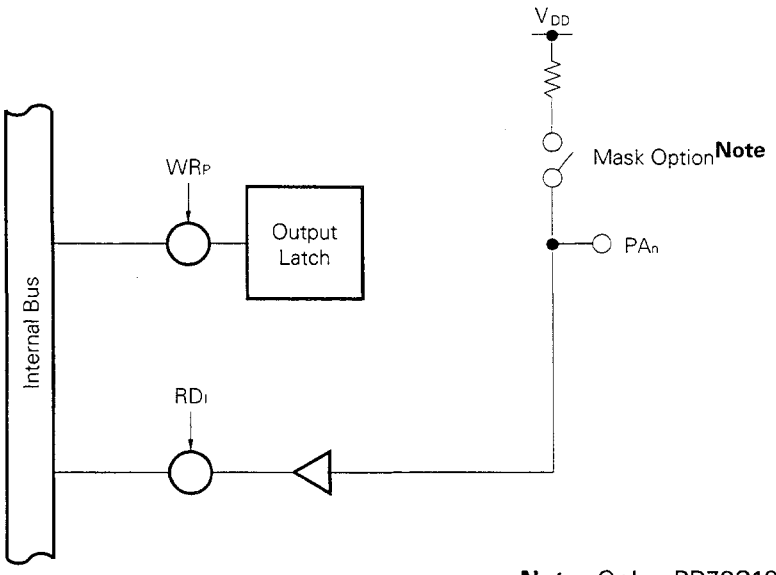


Note Only μPD78C18(A)

(b) When specified as input port (MAn = 1)

PA line contents can be loaded into an accumulator by a transfer instruction. They can also be directly tested bit-wise by an arithmetic or logical operation instruction without the use of an accumulator.

Fig. 2-7 Port A Specified as Input Port



Note Only μPD78C18(A)

Actual execution of an instruction which manipulates port A is performed in 8-bit units. If a port A read instruction (MOV A, PA) is executed, the input line contents of the port specified for input and the output latch contents of the port specified for output are loaded into an accumulator. When a port A write instruction (MOV PA, A) is executed, data is written to the output latch of both ports specified for input and output. However, the output latch contents of a bit specified as an input port cannot be loaded to the accumulator and are not output to an external pin (which functions as input pin), because the output buffer is off.

- MODE A register (MA)

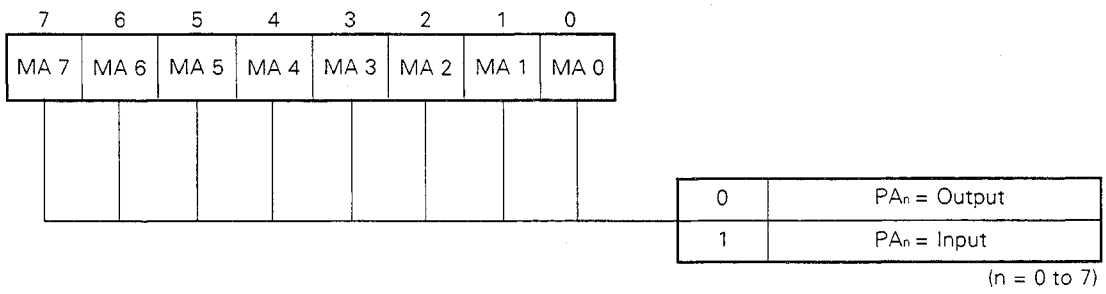
  8-bit register which specifies port A input/output.

  Port A input/output can be specified bit-wise. If the MODE A register corresponding bit is set (1), this register is input, and if the bit is reset (0), this register is output.

  After $\overline{\text{RESET}}$ input or in the hardware STOP mode, all the bits are set, and port A is in the input mode resulting in the below status.

  High-impedance : Without pull-up resistor
  High level : With pull-up resistor

Fig. 2-8 MODE A Register Format



(2) PB7 to PB0 (PORT B)

Like port A, port B is an 8-bit input/output port with input/output buffer and output latch functions. Port B can be set as an input or output port bit-wise using the MODE B register (MB). μPD78C18(A) port B pull-up resistor specification is performed bit-wise by mask option.

Port B is set as follows when setting the input port or after reset.

High-impedance : Without pull-up resistor
High level : With pull-up resistor

As with port A, direct bit setting/resetting of port B output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Data transfer to/from an accumulator is also possible.

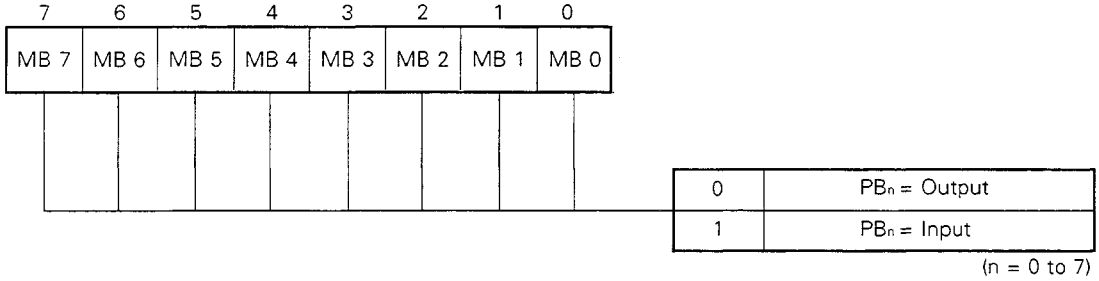
- MODE B Register (MB)

Like the MODE A register, the MODE B register is an 8-bit register which specifies port B input/output bit-wise.

After RESET input or in the hardware STOP mode, all the bits are set (1), and port B is in the input mode resulting in the status below.

High-impedance : Without pull-up resistor
High level : With pull-up resistor

Fig. 2-9 Mode B Register Format



(3) PC7 to PC0 (PORT C)

Port C (PC7 to PC0) is an 8-bit special input/output port which functions as various control signals as well as general-purpose input/output ports in which input/output is set bit-wise like port A. These are switched over bit-wise according to the setting of the MODE C register and MODE CONTROL C register as shown below.

Table 2-2 Operation of PC7 to PC0

	$MCC_n = 1$	$MCC_n = 0$	
	$MC_n = x$	$MCC_n = 0$	$MC_n = 1$
PC0	TxD output	Output	Input
PC1	RxD inpit	Output	Input
PC2	$\overline{SCK}$ input/output	Output	Input
PC3	$\overline{INT2}$/TI input	Output	Input
PC4	TO output	Output	Input
PC5	CI input	Output	Input
PC6	CO0 output	Output	Input
PC7	CO1 output	Output	Input

(n = 0 to 7)

μPD78C18(A) port C pull-up resistor specification is performed bit-wise by mask option.

In the operation when data is set in the general-purpose input/output ports, as with port A, direct bit setting/resetting/testing of port C output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Data transfer to/from an accumulator is also possible.

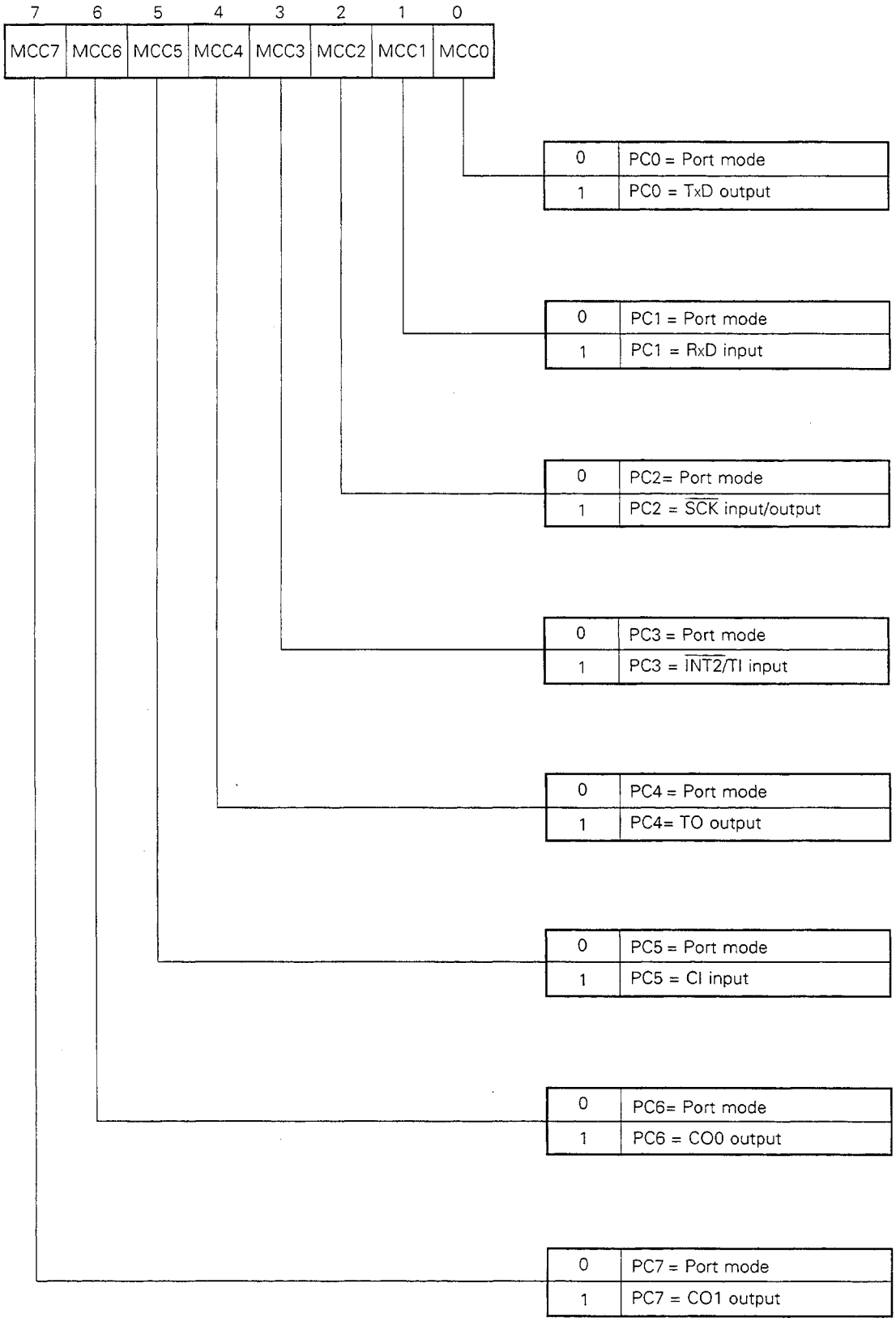
- MODE CONTROL C Register (MCC)

  8-bit register which specifies the port C port/ control signal input/output mode bit-wise.

  If the MODE CONTROL C register corresponding bits are set (1), PC7 to PC0 are in the control signal input/output mode, and if these are reset (0), in the port mode.

  After $\overline{RESET}$ input or in the hardware STOP mode, all the bits of the MODE CONTROL C register are reset (0), and the port mode is set.

Fig. 2-10 MODE CONTROL C Register Format



- MODE C register (MC)

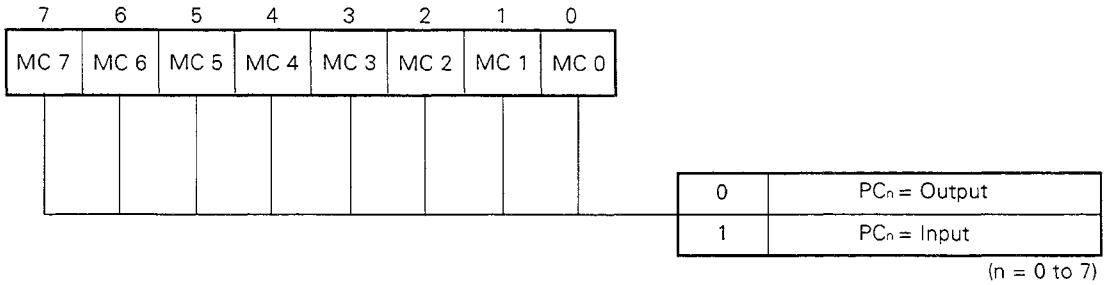
The MODE C register is an 8-bit register by which, like the MODE A register of port A, port C input/output specification is performed bit-wise.

Contents of the MODE C register corresponding to the bits set to the control mode by the MODE CONTROL C register are ignored.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, all bits of the MODE C register are set (1). Because all bits of the MODE CONTROL C register are reset (0), port C is an input port and the below state is set.

High-impedance : Without pull-up resistor
High level : With pull-up resistor

Fig. 2-11 MODE C register Format



(4) PD7 to PD0 (PORT D)

■ μPD78C17(A)

Can be used for address/data bus. These have no functions as a port.

■ μPD78C18(A)

8-bit general-purpose input/output ports shared as multiplexed address/data bus. These ports can be specified for input/output in byte units (8-bit unit) as general-purpose input/output ports, and function as multiplexed address/data bus when external expansion memory is connected. This switchover is performed by the MEMORY MAPPING register.

In the operation when data is set in the general-purpose input/output ports, unless input/output is specified in byte units, as with port A, direct bit setting/resetting/testing of port F output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Data transfer to/from an accumulator is also possible.

(5) PF7 to PF0 (PORT F)

■ μPD78C17(A)

General-purpose input/output ports shared as address bus.

These pins function as address outputs corresponding to the size of externally installed memory according to the MODE0 and MODE1 pin settings.

Pins which are not used for address output can be used for general-purpose input/output ports which have the same port function as for port A. Input/output setting is performed by the MODE F register.

Table 2-3 Operation of PF7 to PF0 μPD78C17(A)

MODE1	MODE0	PF 7	PF 6	PF 5	PF 4	PF 3	PF 2	PF 1	PF 0	External Address Space
0	0	Port	Port	Port	Port	AB11	AB10	AB9	AB8	4 Kbytes
0	1	Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	16 Kbytes
1	0	Setting prohibited								
1	1	AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	63 Kbytes

When this is set as general-purpose input/output ports, as with port A, direct bit setting/resetting/ testing of port C output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Data transfer to/from an accumulator is also possible.

- μPD78C17(A) MEMORY MAPPING register (MM)

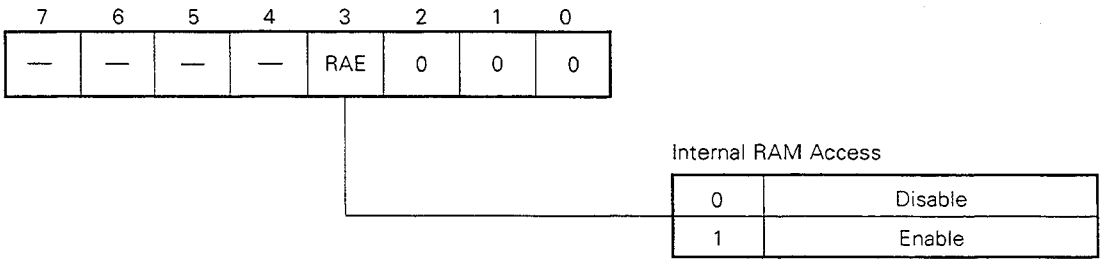
  A register which controls internal RAM access permission.

  Bit 3 (RAE) of the MEMORY MAPPING register controls whether or not internal RAM is permitted.

  When internal RAM is used in external extension and external memory is used in the area, RAE bit is set to "0" and internal RAM access is prohibited.

  Contents of RAE bit is retained, even if $\overline{\text{RESET}}$ signal is input in the normal operation. However, at power-on reset, RAE bit is undefined and RAE bit should be initialized by an instruction.

Fig. 2-12 μPD78C17(A) MEMORY MAPPING Register Format



■ μPD78C18(A)

8-bit general-purpose input/output ports shared as address bus.

Can specify input/output bit-wise as general-purpose input/output ports, and address signal is output according to external extension memory size when the external expansion memory of 256 bytes or greater is accessed.

This switchover is performed by the MEMORY MAPPING and MODE F registers.

PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	External Memory
Port	Port	Port	Port	Port	Port	Port	Port	Maximum 256 bytes
Port	Port	Port	Port	AB11	AB10	AB9	AB8	Maximum 4 Kbytes
Port	Port	AB13	AB12	AB11	AB10	AB9	AB8	Maximum 16 Kbytes
AB15	AB14	AB13	AB12	AB11	AB10	AB9	AB8	Maximum 31 Kbytes

When this is set as general-purpose input/ourput ports, as with port A, direct bit setting/resetting/testing of port C output latch contents is possible by an arithmetic or logical operation instruction without the use of an accumulator. Data transfer to/from an accumulator is also possible.

- μPD78C18(A) MEMORY MAPPING register (MM)

  4-bit register which specifies PD7 to PD0 and PF7 to PF0 port/extension mode and controls internal RAM access permission.

  Bits 0, 1, and 2 (MM0, MM1, MM2) in the MEMORY MAPPING register control specification of PD7 to PD0 port/extension mode, input/output, and PF7 to PF0 address line.

  When bits MM1 and MM2 in the MEMORY MAPPING register are "0", PD7 to PD0 and PF7 to PF0 are set as general-purpose input/output port, input/output of PD7 to PD0 is specified by MM0, and input/output of PF7 to PF0 is specified by the MODE F register.

  4 types of external extension memory (256 bytes, 4 Kbytes, 16 Kbytes, and 31 Kbytes) can be selected, and ports which are not used for address line are used as general-purpose input/output ports.

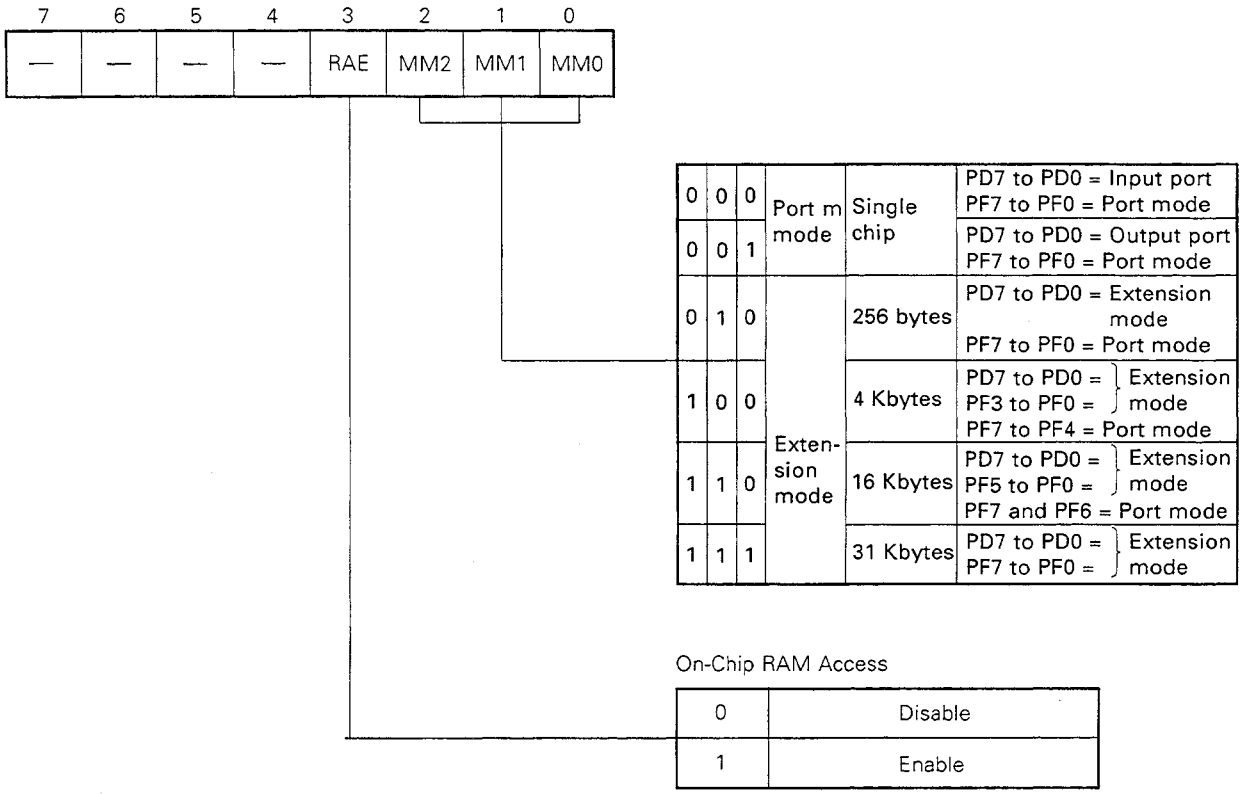
  Bit 3 (RAE) of the MEMORY MAPPING register controls whether or not the access to internal RAM is permitted.

  When internal RAM is not used in external extension and external memory uses the area, RAE bit is set to "0" and internal RAM access is prohibited.

  After $\overline{\text{RESET}}$ input or in the hardware STOP mode, bits MM0, MM1, and MM2 of the MEMORY MAPPING register are reset (0), and PD7 to PD0 are input ports (high-impedance).

  Even if the $\overline{\text{RESET}}$ signal is input to the RAE bit in the normal operation, contents in input time are retained. However, the RAE bit is undefined at power-on reset, the RAE bit should be initialized by an instruction.

Fig. 2-13 μPD78C18(A) MEMORY MAPPING Register Format

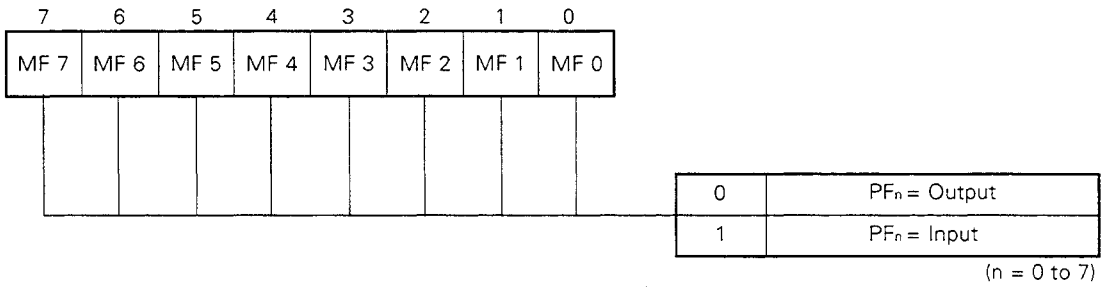


- MODE F register (MF)

The MODE F register specifies port F input/output in the same way as for the MODE A register in port A. However, contents of the MODE F register corresponding to port F bits specified as address line by the MEMORY MAPPING register are in the output mode.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, all the bits of the MODE F register are set (1) and port F is an input port (high-impedance).

Fig.2-14 MODE F Register Format



2.6 TIMER

This is an interval timer which has two 8-bit timers (TIMER0, TIMER1). These are programmable independently. By cascading these can also be used as 16-bit interval timer, and can be used for counting TI input.

The timer is composed of TIMER0 and TIMER1, as shown in 2-15, including 8-bit TIMER REG (TM0, TM1), 8-bit COMPARATOR, 8-bit UPCOUNTER, and TIMER F/F. Input selection, timer operation and TO output are controlled by the timer mode register (TMM).

In TIMER0, ϕ_{12} (1 μs: 12-MHz operation) and ϕ_{384} (32 μs: 12-MHz operation) internal clock and TI input are input. In TIMER1, not only these inputs but also TIMER0 match signal are input.

Because TIMER0 operates in the same way as TIMER1, TIMER0 operation is described below.

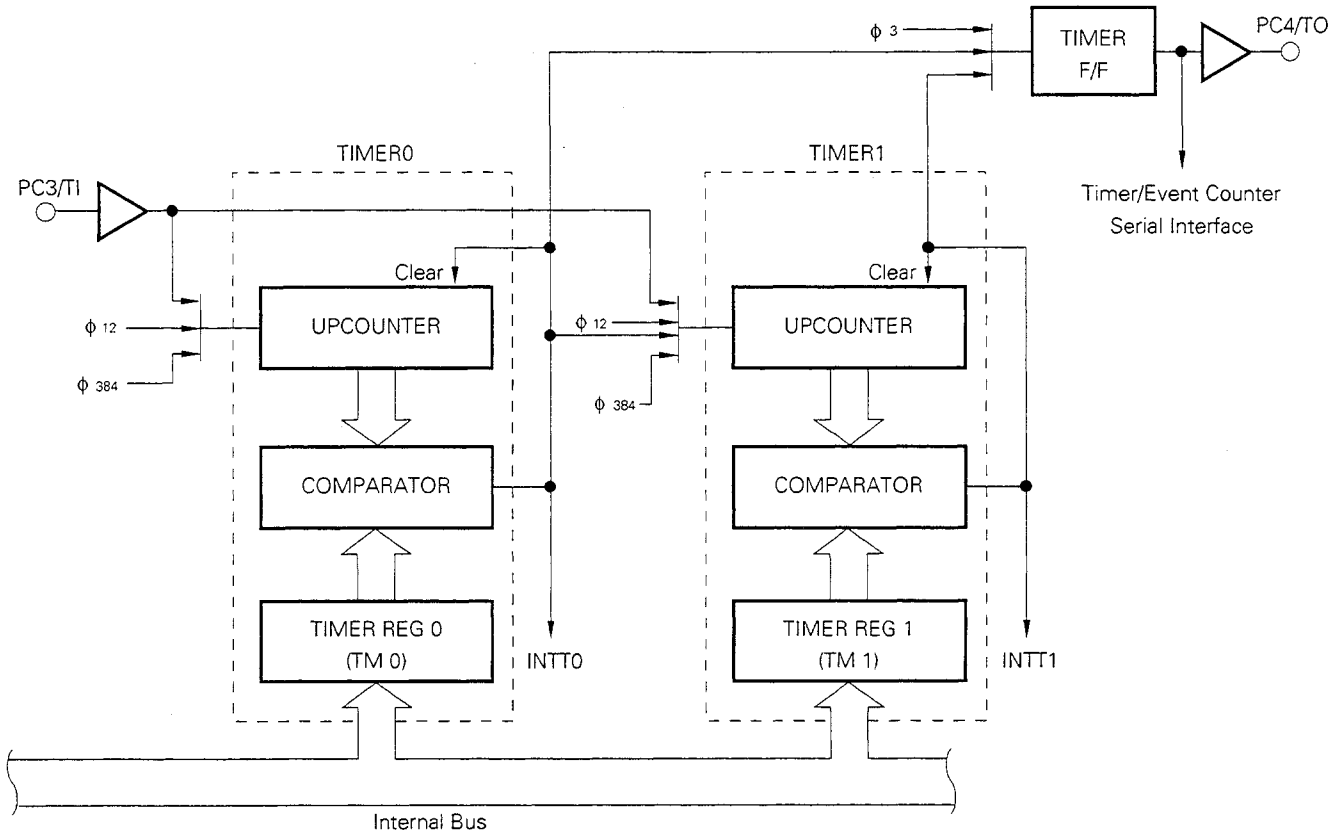
At first, a count value is set in TIMER REG0, and TIMER0 input and TIMER0 start data (bit 4 in the timer mode register = "0") are set in the timer mode register to start TIMER0. The UPCOUNTER is incremented one input at a time. The COMPARATOR always compares contents of the incremented UPCOUNTER with those of TIMER REG0, and if these match, the match signal (internal interrupt: INTT0) is generated. This match clears contents of UPCOUNTER and increment starts again from 00H. Therefore, the interval is set by count time, which is a count value set by TIMER REG0. This allows the timer to operate as an interval timer which generates interrupts repeatedly.

By setting (1) bit 1 (MKT0) of the interrupt mask register (MKL), internal interrupt (INTT0) is disabled.

The TO output has timers COMPARATOR match signal and TIMER F/F complemented by ϕ_3 (250 ns: 12-MHz operation) internal clocks, and can obtain a square wave which has a half period of the count time or ϕ_3. By setting the timer/event counter mode register (ETMM), this output can be used for the timer event counter reference time.

By setting the serial mode register (SMH), the timer can be used as the serial clock ($\overline{SCK}$) in serial interface.

Fig. 2-15 Timer Block Diagram



Remarks 1. ϕ_3 = fxx x 1/3

2. ϕ_{12} = fxx x 1/12 Where, fxx = oscillation frequency (MHz)

3. ϕ_{384} = fxx x 1/384

(1) Timer mode register (TMM)

This is an 8-bit register which controls TIMER0, TIMER1, and TIMER F/F operation (see **Fig. 2-16**).

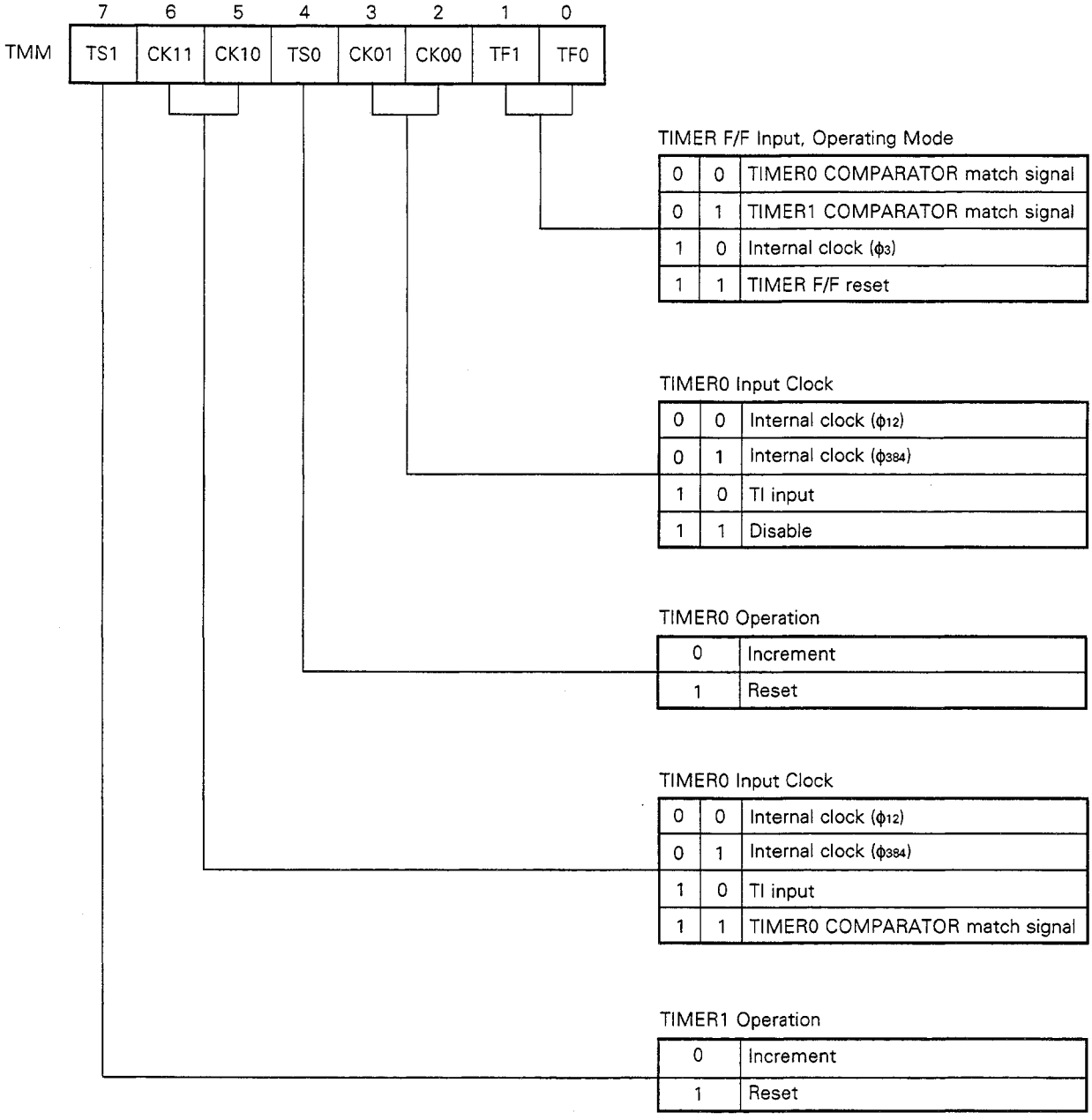
The timer mode register bits 0 and 1 (TF0, TF1) control the TIMER F/F operating mode, bits 2 and 3 (CK00, CK01) control TIMER0 input clock, bit 4 (TS0) controls TIMER0 operation. Bits 5 and 6 (CK10, CK11) control TIMER1 input clock, and bit 7 (TS1) controls TIMER1 operation.

TS0 and TS1 bits clear these UPCOUNTERs to 00H by "1", and stop increment. By changing "1" to "0", the UPCOUNTER starts increment from 00H.

The internal clock (ϕ_3) divides the oscillator frequency by 3, the internal clock (ϕ_{12}) divides it by 12, and the internal clock (ϕ_{384}) divides it by 384.

After $\overline{\text{RESET}}$ input, the timer mode register is set to FFH, the UPCOUNTERs in TIMER0 and TIMER1 are cleared in the suspended state, and TIMER F/F is reset.

Fig. 2-16 Timer Mode Register (TMM) Format



2.7 TIMER/EVENT COUNTER

The μPD78C17(A)/78C18(A) have a 16-bit multi-function timer/event counter which has the functions shown below.

- o Interval timer
- o External event counter
- o Frequency measurement
- o Pulse width measurement
- o Programmable square wave output
- o One pulse output

The timer/event counters are composed of 16-bit timer/event counter upcounter (ECNT), timer/event counter capture register (ECPT), comparator, timer/event counter REG0 and REG1 (ETM0, ETM1), and output/input and interrupt and clear control circuits.

ECNT is a 16-bit upcounter which counts an input pulse, and cleared by the clear control circuit.

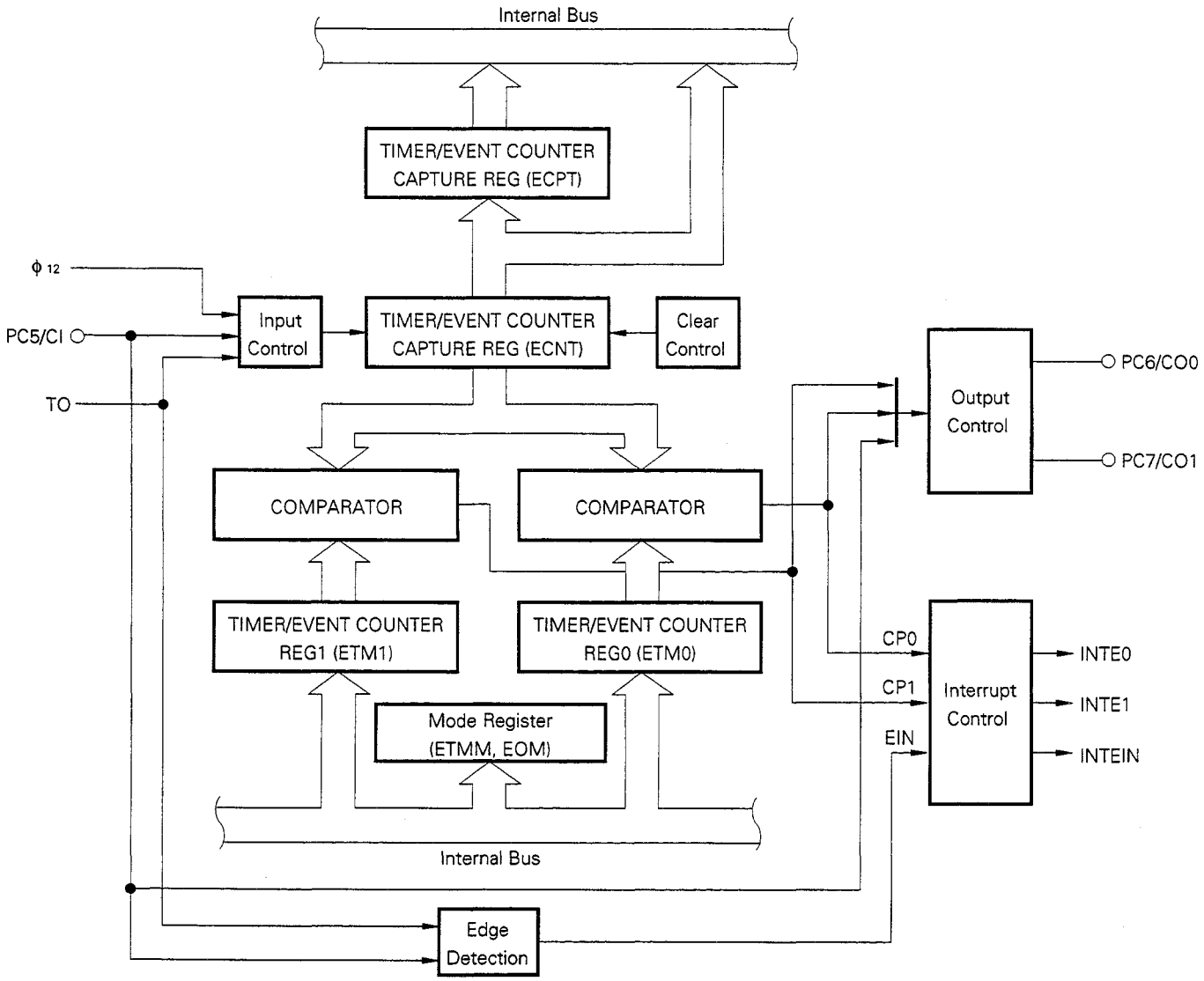
The ECPT register is a 16-bit buffer register which retains the contents of ECNT. The timing to latch contents of ECNT by the ECPT register is the falling edge of CI input when input to ECNT is an internal clock, and is the falling edge of TO output when input to ECNT is CI input.

The ETM0 and ETM1 registers are two 16-bit registers which set a number of counts and data is exchanged by 16-bit data transfer instructions via an extended accumulator.

The comparator compares contents of ECNT with contents of the ETM0 and ETM1 registers, and if these match, match signals are generated.

The interrupt control circuit is a circuit which controls interrupts from the timer/event counter. The following interrupt sources are generated. These are generated by three signals: the ECNT and ETM0 register match signal (INTE0), the ECNT and ETM1 register match signal (INTE1), and the CI input or timer output (TO) falling edge (INTEIN).

Fig. 2-17 Timer/Event Counter Block Diagram



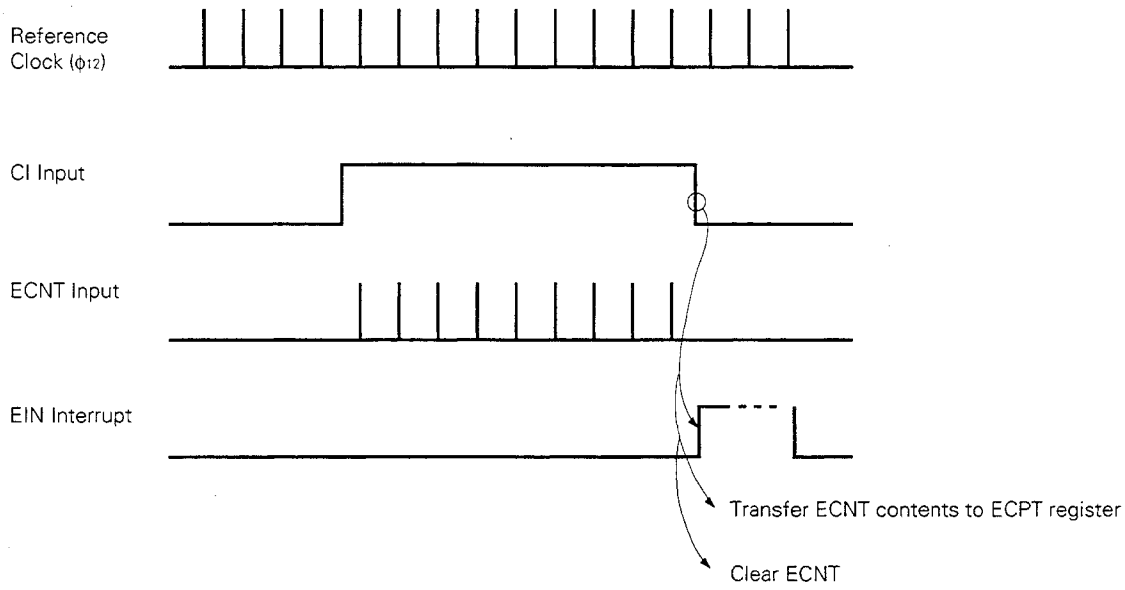
Remarks ϕ_{12} = fxx x 1/12, where fxx = oscillation frequency (MHz)

Next, using pulse width measurement as an example, the operation is described.

This operation purpose is measurement for high-level width of external pulse input to CI. This is performed by setting the timer/event counter mode register (ETMM) to 09H.

ECNT continues internal clock (ϕ_{12}) count while CI is high. If the external pulse which is input to CI falls, the contents of ECNT are transferred to the ECPT register. ECNT is cleared and an internal interrupt (INTEIN) is generated (see **Fig. 2-18**). Therefore, using contents of the ECPT register and internal clock period, the pulse width is measured.

Fig. 2-18 Pulse Width Measurement



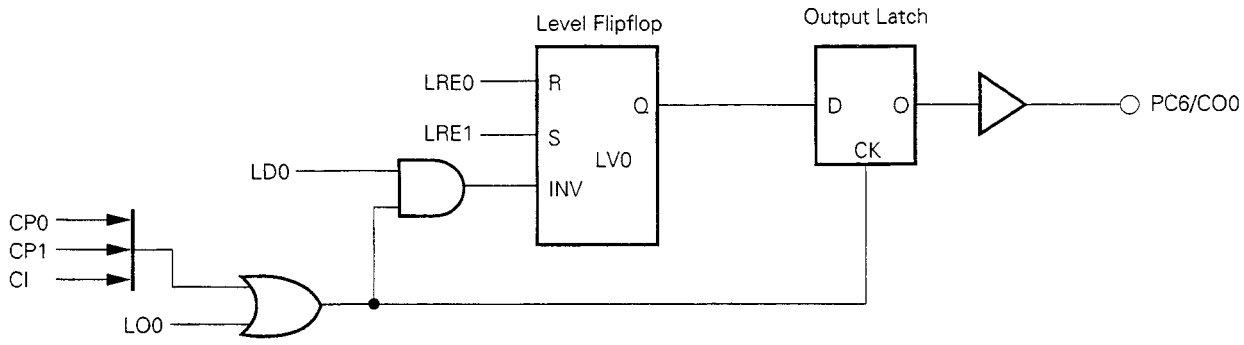
The μPD78C17(A)/78C18(A) have an output control circuit which outputs pulses which can be changed in pulse width and period by interlocking with the timer/event counter.

The output control circuit outputs are CO0 output and CO1 output. Because these share the same configuration, CO0 output is described. Fig. 2-19 shows the CO0 configuration. CO0 output is a master-slave type output. The first phase level F/F (LV0) retains the level which is output next, and the second phase output latch outputs the LV0 level to off-chip.

By setting the timer/event counter output mode register (EOM), LV0 can be set/reset. LV0 has a level inversion pin (INV) and LV0 level can be inverted at the output time by setting the timer/event counter mode register.

Timing when the output latch outputs LV0 level to off-chip is performed by output timing of the timer/event counter mode register setting.

Fig. 2-19 Output Control Circuit



Next, the operation which outputs a square wave to the CO0 pin is described.

At first, after ECNT is cleared, a count value (ETM0 < ETM1) is set in the ETM0 and ETM1 registers, and data for LV0 initial status specification and to enable LV0 level inversion is set in the timer/event counter output mode register.

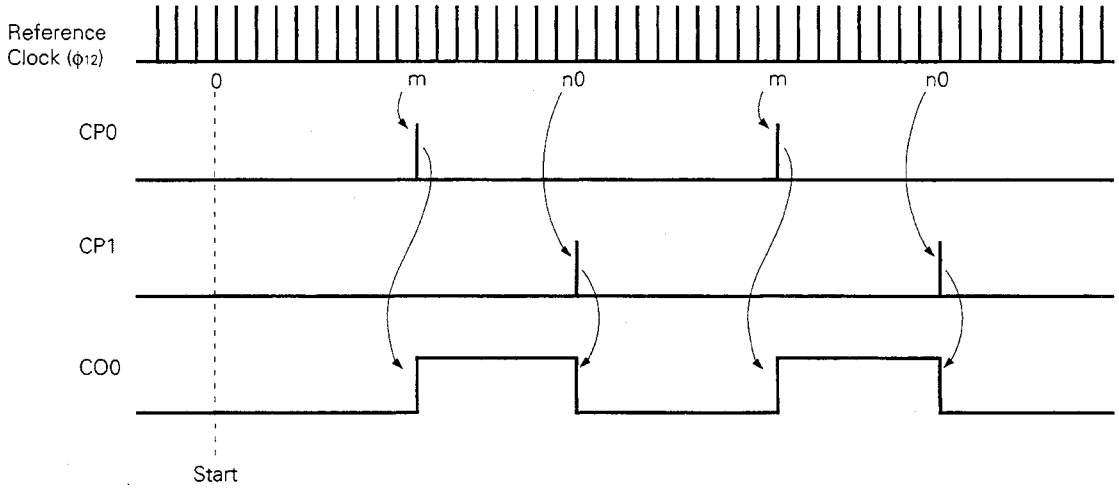
In the timer/event counter mode register, by setting an input to ECNT to ϕ_{12} (1 μs: 12-MHz operation) internal clock, the ECNT clear mode to the ECNT and ETM1 register match signal, and CO0 pin output timing to the ECNT and ETM0 register match signal or ECNT and ETM1 register match signal, the timer/event counter starts operation.

ECNT is incremented one ϕ_{12} internal clock at a time, the comparator compares incremented ECNT with the ETM0 and ETM1 registers, and if these match, the match signal (CP0, CP1) is generated. By this match signal, LV0 level is output to the CO0 pin, and LV0 level is inverted.

ECNT is cleared by the ECNT and ETM1 register match signal (CP1), ECNT increments again from 0000H, and the above-mentioned steps are repeated (see **Fig. 2-20**).

Therefore, a programmable square wave which has the ETM0 and ETM1 register count as a pulse width is output.

Fig. 2-20 Square Wave Output



Remarks ETM0 register = m
ETM1 register = n (m < n: m and n are count values.)

(1) Timer/event counter mode register (ETMM)

This is an 8-bit register which controls the timer/event counter (see **Fig. 2-21**).

The timer/event counter mode register bits 0 and 1 (ET0, ET1) control the timer event counter upcounter (ECNT) input clock, bits 2 and 3 (EM0, EM1) control the ECNT clear mode, bits 4 and 5 (CO00, CO01) control output timing when the output latch contents are output to the counter output0 (CO0). Bits 6 and 7 (CO10, CO11) control CO1 output timing.

The internal clock (ϕ_{12}) divides the oscillation frequency by 12.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, the timer/event counter mode register is reset to 00H.

Fig. 2-21 Timer/Event Counter Mode Register Format

	7	6	5	4	3	2	1	0
ETMM	CO11	CO10	CO01	CO00	EM 1	EM 0	ET 1	ET 0

ECNT Input Clock

0	0	Internal clock (ϕ_{12})
0	1	ϕ_{12} while CI input is in the high level
1	0	CI input
1	1	CI input while TO is in the high level

ECNT Clear Mode

0	0	Stop after clear
0	1	Free running Clear a full count at a time
1	0	Clear at the fall of CI input (ET1 = 0) Clear the fall of TO (ET1 = 1)
1	1	Clear by matching ECNT and ETM1

CO0 Output Timing

0	0	ECNT and ETM0 match
0	1	Setting prohibited
1	0	ECNT and ETM0 match, or CI input fall
1	1	ECNT and ETM0 match, or ECNT and ETM1 match

CO1 Output Timing

0	0	ECNT and ETM1 match
0	1	Setting prohibited
1	0	ECNT and ETM1 match, or CI input fall
1	1	ECNT and ETM0 match, or ECNT and ETM1 match

(2) Timer/event counter output mode register (EOM)

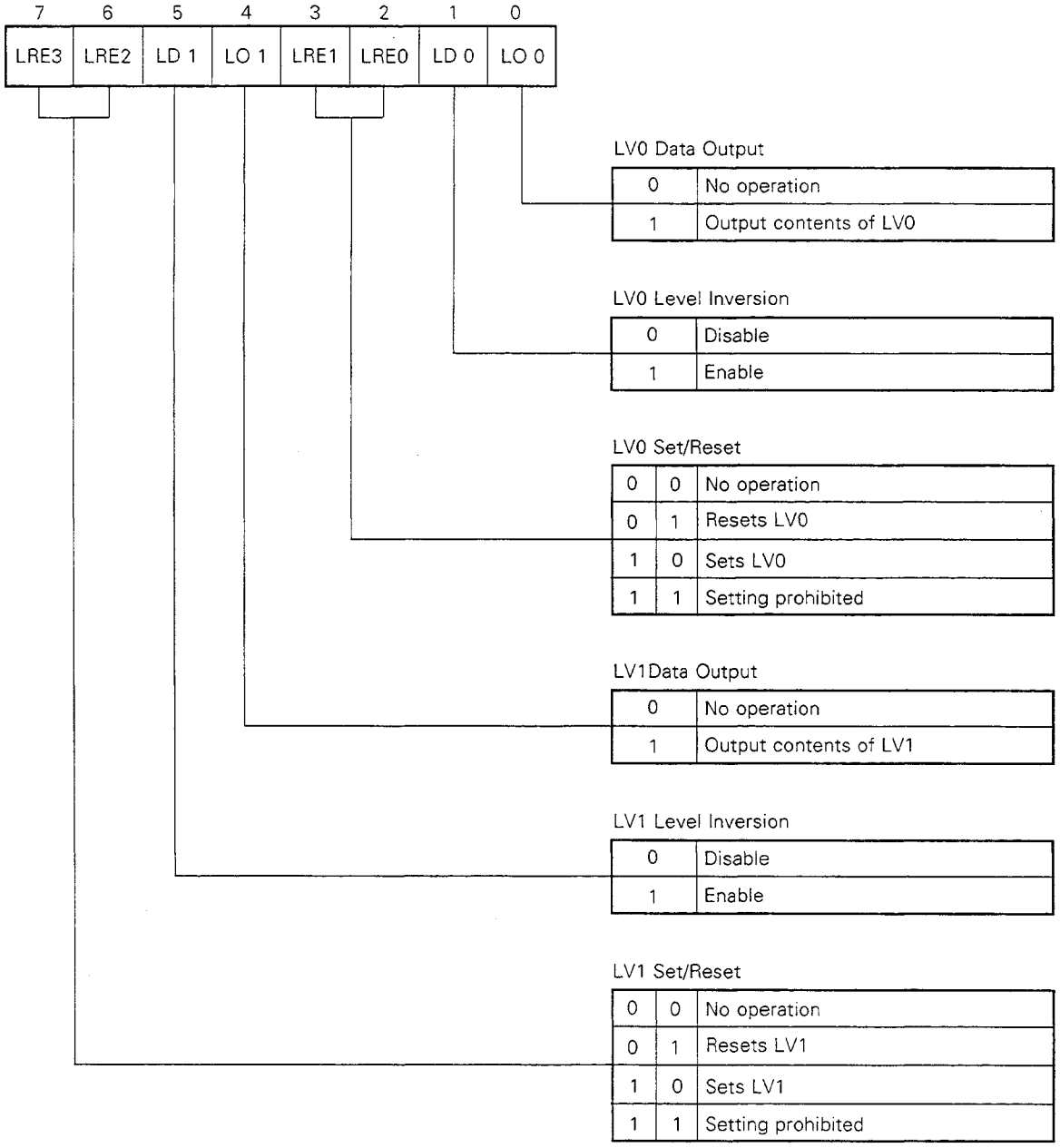
This is an 8-bit register which controls the timer/event counters CO0 and CO1 (Counter Output 0, 1) operating mode.

The timer/event counter output mode register bits 0 and 4 (LO0, LO1) control whether or not LV0 and LV1 level are output to the CO0 and CO1 pins, bits 1 and 5 (LD0, LD1) control whether or not LV0 and LV1 level are inverted at an output timing specified by the timer/event counter mode register, bits 2, 3, 6, and 7 (LRE0, LRE1, LRE2, LRE3) control LV0 and LV1 setting/resetting.

Bits LO0, LO1, LRE0, LRE1, LRE2, and LRE3 are automatically reset (0) after individual operations.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, the timer/event counter output mode register is reset to 00H.

Fig. 2-22 Timer/Event Counter Output Mode Register (EOM) Format



2.8 SERIAL INTERFACE

The μPD78C17(A)/78C18(A) have the serial interface using the transmit/receive method by start/stop bit. The three types of operating modes are shown below.

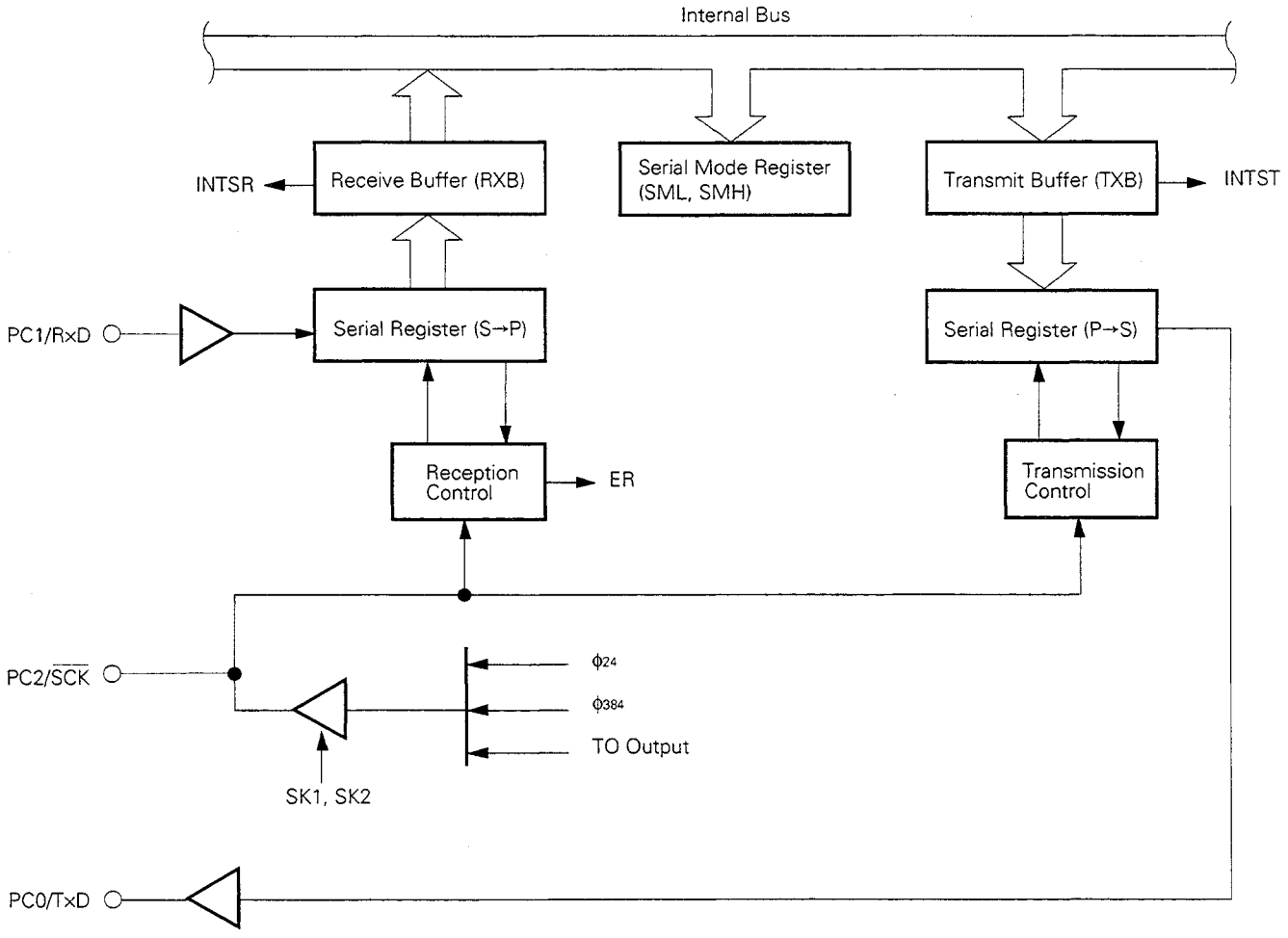
- Asynchronous (start-stop) mode : Establishes data bit synchronization and character synchronization by start bit.
- Synchronous mode : Data transfer is performed in synchronization with the serial clock.
- I/O interface mode : As for serial data transfer in the μPD7801/78C06A etc., data transfer is performed in synchronization with the serial clock.

The serial interface block is composed of the serial data input (RxD), serial data output (TxD), 3 serial clock input/output ($\overline{SCK}$) pins, transfer control block, two 8-bit serial registers for transmission and reception, and 8-bit transmission buffer and reception buffer (see **Fig. 2-23**).

As the serial registers and buffers for transmission and reception are provided, transmission or reception is individually performed (full-duplex double buffer transmitter/receiver).

However, the serial clock ($\overline{SCK}$) is shared in transmission and reception, and half-duplex transmission/reception is performed in the synchronous mode and I/O mode.

Fig. 2-23 Serial Interface Block Diagram



Remarks ϕ_{24} = fxx x 1/24 Where, fxx = oscillation frequency (MHz)
ϕ_{384} = fxx x 1/384

(1) Asynchronous mode

In case of the asynchronous mode, clock rate, character length, number of stop bits, parity enable, and odd or even parity specifications can be controlled by the serial mode register (SML).

Transmission operation is enabled by setting (1) bit 2 (TxE) of the serial mode register (SMH).

If data is written to the transmission buffer by the "MOV TXB, A" instruction and preceding data transfer is terminated, contents of the transmission buffer are transferred to the serial register automatically. The start bit (1 bit), parity bit (odd/even number, no parity), and stop bit (1 or 2 bits) are automatically added to data which is transferred to the serial register. And this data is transmitted from the TxD pin starting from the least significant bit (LSB).

If the transmit buffer is empty, the internal interrupt (INTST) is generated.

Transmission data is transmitted from the TxD pin at the fall of $\overline{SCK}$ in the clock rate of x 1, x 1/16, or x 1/64 serial clock ($\overline{SCK}$).

The maximum data transfer speed in transmission is set by $\overline{SCK}$ and clock rate in 12-MHz operation as shown below.

$\overline{SCK}$ / Clock Rate	Internal Clock		External Clock	
	$\overline{SCK}$	Data Transfer Speed	$\overline{SCK}$	Data Transfer Speed
x 1	500 kHz	500 kbps	660 kHz	660 kbps
x 16	2 MHz	125 kbps	2 MHz	125 kbps
x 64		31.25 kbps		31.25 kbps

When TxE is "0" or the serial register has no transmitted data, the TxD pin is in the marking state (1).

By setting bit 2 (MKST) of the interrupt mask register (MKH), the internal interrupt (INTST) is disabled.

Fig. 2-24 Asynchronous Data Format

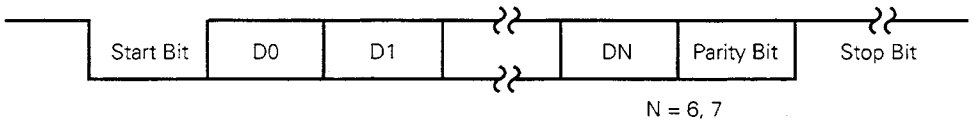


Fig. 2-25 Serial Mode Register Format in Asynchronous Mode

	7	6	5	4	3	2	1	0
SML	S 2	S1	EP	PEN	L 2	L 1	B 2	B 1

Clock Rate

0	1	x 1
1	0	x 16
1	1	x 64

Character Length

0	0	Setting prohibited
0	1	Setting prohibited
1	0	7 bits
1	1	8 bits

Parity Enable

0	Disable
1	Enable

Even Parity Generation/Check

0	Odd number
1	Even number

Number of Stop Bits

0	0	Setting prohibited
0	1	1 bit
1	0	Setting prohibited
1	1	2 bits

	7	6	5	4	3	2	1	0
SMH	0	0	0	0	RxE	TxE	SK2	SK1

$\overline{SCK}$ Selection

0	0	Internal clock (TO output)
0	1	Internal clock (ϕ_{384})
1	0	Internal clock (ϕ_{24})
1	1	External clock

Transmission Enable

0	Disable
1	Enable

Reception Enable

0	Disable
1	Enable

Receive operation is enabled by setting (1) bit 3 (RxE) of the serial mode register (SMH).

The start bit is confirmed by detecting the low level of RxD input and the low level after 1 or 2 bits. Reception is performed by sampling character bit, parity bit, and stop bit following the low level. When data specified in the serial register from RxD is input, data is transferred to the receive buffer. If the receive buffer is full, the internal interrupt (INTSR) is generated.

By setting (1) bit 1 (MKSR) of the interrupt mask register (MKH), the internal interrupt (INTSR) is disabled.

In reception, odd or even parity is checked (when PEN bit = 1). If data do not match (parity error), if stop bit is low (framing error), or if the next data is transferred to the receive buffer when the receive buffer is full (overrun error), the error flag is set (1).

However, because error interrupt mechanism is not provided, test is executed by the skip instruction (SKIT, SKNIT).

The serial clock ($\overline{SCK}$) can be selected as an external or internal clock by the serial mode register (SMH).

Three types of ϕ_{24}, ϕ_{384}, or TO outputs can be selected as internal clock. This clock can be output to off-chip. Or the external serial clock can be input.

By using the internal clock (TO output) as $\overline{SCK}$, the data transfer speed can be flexibly changed by program.

The maximum data transfer speed in reception is set by $\overline{SCK}$ and the clock rate in 12-MHz operation as shown below.

$\overline{SCK}$ / Clock Rate	Internal Clock		External Clock	
	$\overline{SCK}$	Data Transfer Speed	$\overline{SCK}$	Data Transfer Speed
x 1[Note2]	500 kHz	500 kbps	660 kHz 1 MHz	660 kbps 1 Mbps[Note1]
x 16	2 MHz	125 kbps	2 MHz	125 kbps
x 64		31.25 kbps		31.25 kbps

Notes 1. If data of transfer speed 660 kbps to 1 Mbps is received, 2 stop bits are required.

2. In x 1 clock rate, RxD and $\overline{SCK}$ synchronization needs to be externally established.

For an example, when data is transferred in the data transfer speed of 110 to 9600 bps, when the timer input clock is set as internal clock (ϕ_{12}), the timer count value (C) is shown below.

Oscillation Frequency (MHz) / Data Transfer Speed (bps) / N	7.3728		11.0592		14.7456	
	16	64	16	64	16	64
9600	C = 2	-	C = 3	-	C = 4	C = 1
4800	4	C = 1	6	-	8	2
2400	8	2	12	C = 3	16	4
1200	16	4	24	6	32	8
600	32	8	48	12	64	16
300	64	16	96	24	128	32
150	128	32	192	48	256	64
110	175	44	262	65	370	88

(2) Synchronous mode

In the synchronous mode, data transfer is performed with 8-bit character length fixed, and with no parity bit. Therefore, the serial mode register (SML) is set to 0CH (see **Fig. 2-26**).

Transmission operation is enabled by setting (1) bit 3 (TxE) of the serial mode register (SMH).

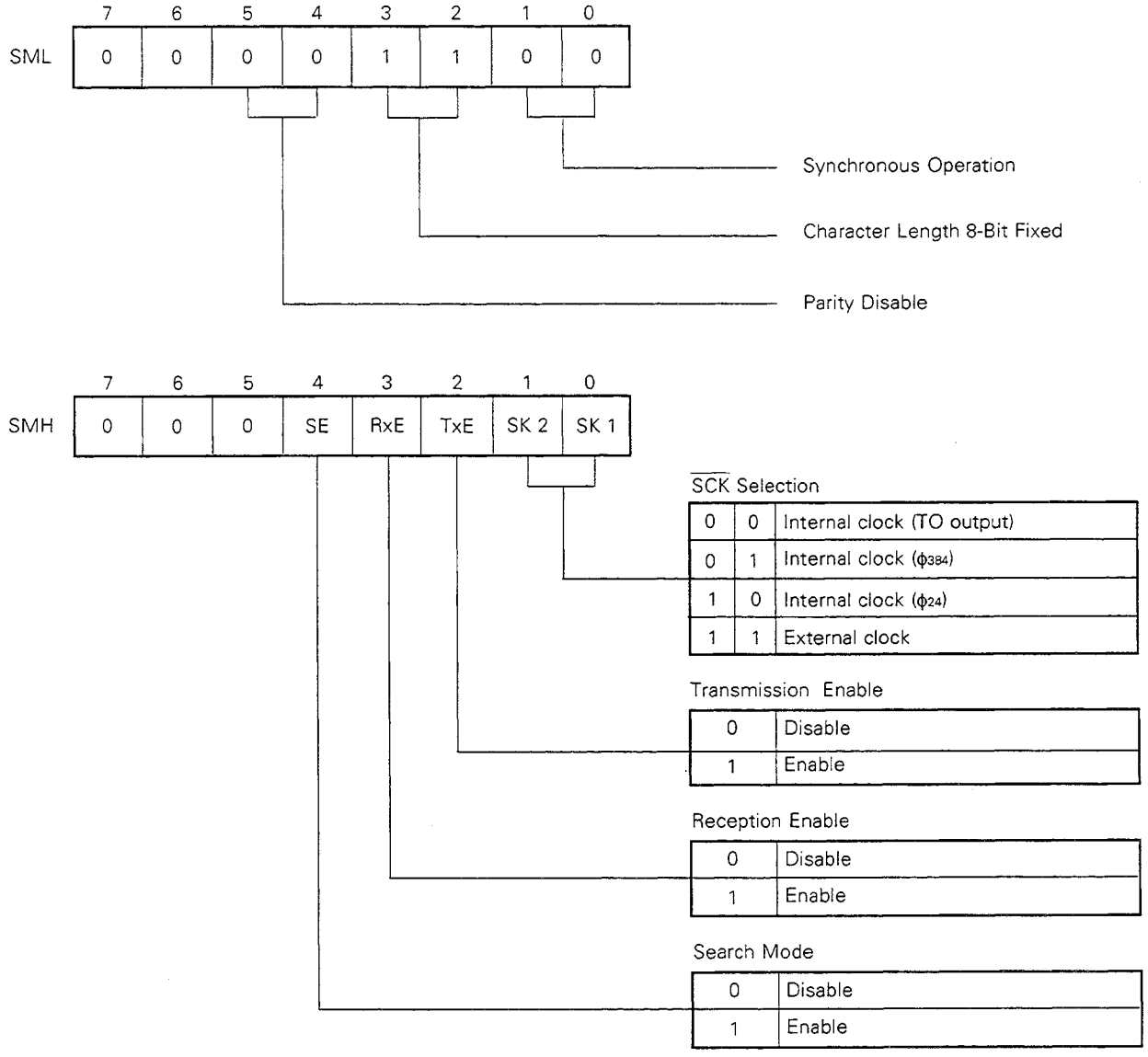
If data is written to the transmit buffer by the "MOV TXB, A" instruction and preceding data transfer is terminated, the contents of the transmit buffer are automatically transferred to the serial register and converted to serial data, and data starting from LSB are transmitted from TxD at the falling edge of $\overline{SCK}$. The serial data is transferred in the same rate as for $\overline{SCK}$.

Data transfer speed in transmission is maximum 500 kbps when an internal clock is used for $\overline{SCK}$ and maximum 1 Mbps when an external clock is used (12-MHz operation).

When data is transferred from the transmit buffer to the serial register and the transmit buffer is empty, the internal interrupt (INTST) is generated.

When TxE is "0" or the serial register has no transmitted data, the TxD pin is in the marking state (1).

Fig. 2-26 Serial Mode Register Format in Synchronous Mode



In the synchronous mode, 2 types of receive operation can be selected. This mode can be controlled by SE bit of the serial mode register (SMH).

By setting SE bit (1), the search mode is set. On each 1-bit reception from the RxD pin, the contents of the serial register are transferred to the receive buffer and the internal interrupt (INTSR) is generated. Because the μPD78C17(A)/78C18(A) don't have a synchronous character detection circuit by hardware, a synchronous character detection is required by software.

If receive synchronization is established after a synchronous character is detected, SE bit is reset (0). By resetting the SE bit, the character mode is set. On each 8-bit data reception, the contents of the serial register are transferred to the receive buffer and the internal interrupt (INTSR) is generated.

By setting (1) MKSR bit of the interrupt mask register, the internal interrupt (INTSR) is disabled.

In the synchronous mode, data is output from TxD at the falling edge of $\overline{SCK}$, and data is input from RxD at the rising edge of $\overline{SCK}$.

$\overline{SCK}$ can be selected as an internal clock or external clock by setting the serial mode register (SMH).

Data transfer speed in reception is maximum 500 kbps when an internal clock is used for $\overline{SCK}$ and maximum 660 kbps when an external clock is used (12-MHz operation).

(3) I/O interface mode

When input/output is extended to off-chip or I/O controllers (A/D converter, liquid crystal display controller, etc.) are connected to this chip, this mode is effective.

In the I/O interface mode, data transfer is performed starting from the most significant bit (MSB) with 8-bit character length fixed, and with no parity bits. Therefore, the serial mode register (SML) should be set to 0CH and bit 5 (IOE) of the serial mode register (SMH) is set to "1".

This mode establishes synchronization by controlled $\overline{SCK}$ (8 cycles of the serial clock) and $\overline{SCK}$ should be high except during data transfer.

The transmission operation is enabled by setting (1) bit 2 (TxE) of the serial mode register (SMH).

If data is written by the "MOV TXB, A" instruction, data is transferred to the serial register automatically, and is output from TxD at the falling edge of controlled $\overline{SCK}$. The transmit buffer is empty, the internal interrupt (INTST) is generated.

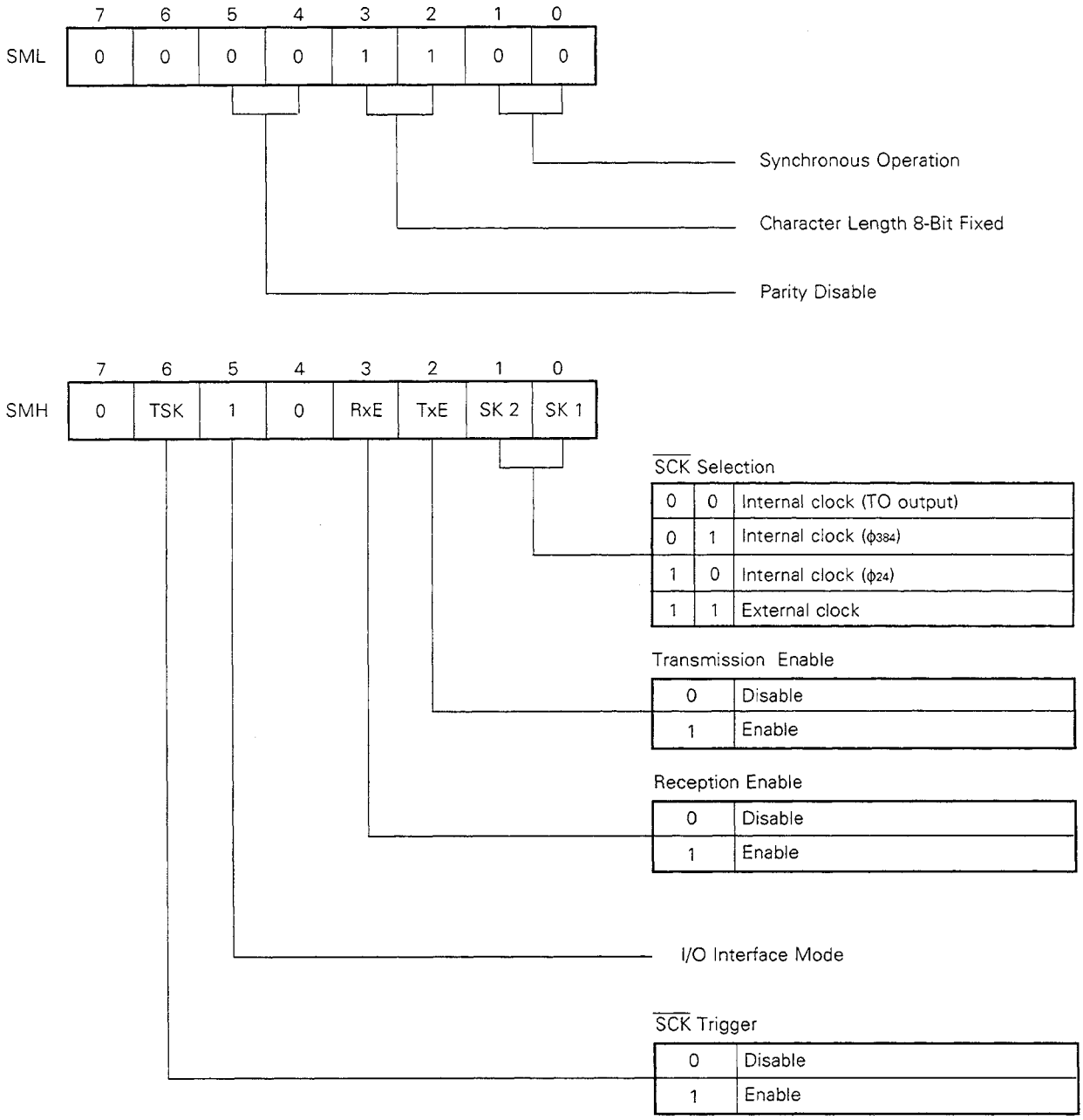
Data transfer speed in transmission is maximum 500 kbps when an internal clock is used for $\overline{SCK}$ and maximum 1 Mbps when an external clock is used (12-MHz operation).

The reception operation is enabled by setting (1) bit 3 (RxE) of the serial mode register (SMH), and receive data is input to the serial register at the rising edge of controlled $\overline{SCK}$. When the serial register receives 8-bit data, data is transferred from the serial register to the receive buffer and the internal interrupt (INTSR) is generated.

$\overline{SCK}$ can be selected as an internal clock or external clock by the serial mode register (SMH).

Data transfer speed in reception is maximum 500 kbps when an internal clock is used for $\overline{SCK}$ and maximum 660 kbps when an external clock is used for $\overline{SCK}$ (12-MHz operation). 6 states or more is required in 8th $\overline{SCK}$ high-level width.

Fig. 2-27 Serial Mode Register Format in I/O Interface Mode



(4) Serial mode register (SML, SMH)

These are two 8-bit registers which control the serial interface operation (see **Figs. 2-28** and **2-29**).

The serial mode low register (SML) bits 0 and 1 (B1, B2) control switchover of the asynchronous mode and synchronous operation and clock rate in the asynchronous mode, bits 2 and 3 (L1, L2) control character length, bit 4 (PEN) controls parity enable, bit 5 (EP) controls odd or even parity, and bits 6 and 7 (S1, S2) control a number of stop bits.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, the serial mode low register (SML) is set to 48H.

The serial mode high register (SMH) bits 0 and 1 (SK1, SK2) control whether an internal clock or external clock is used as the serial clock ($\overline{\text{SCK}}$), bit 2 (TxE) controls the transmission operation, bit 3 (RxE) controls the reception operation, bit 4 (SE) controls whether or not the search mode is set in the synchronous mode. Bit 5 (IOE) controls whether the synchronous mode or I/O interface mode is set, and bit 6 (TSK) starts the serial clock when data is received using the internal clock in the I/O interface mode. The TSK bit is automatically reset (0) after the serial clock starts.

When the serial clock is specified as an internal clock, the $\overline{\text{SCK}}$ value is determined by the following expressions.

Internal clock (ϕ_{24}):

$\overline{\text{SCK}} = fxx / 24$

Internal clock (ϕ_{384}):

$\overline{\text{SCK}} = fxx / 384$

Internal clock (TO output):

Timer input clock is ϕ_{12}:

$\overline{\text{SCK}} = fxx / (24 \times C)$

Timer input clock is ϕ_{384}:

$\overline{\text{SCK}} = fxx / (768 \times C)$

TIMER F/F input is ϕ_3:

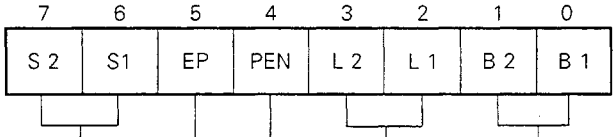
$\overline{\text{SCK}} = fxx / 6$

However, fxx is set in the oscillation frequency, $\overline{\text{SCK}}$ is set in the serial clock, and C is set in the timer count value.

When TIMER F/F input is ϕ_3 in case of the internal clock (TO output), the asynchronous mode can only be used when the clock rate is 16 or 64.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, the serial mode high register (SMH) is reset to 00H.

Fig. 2-28 Serial Mode Low Register (SML) Format



Clock Rate

0	0	Synchronous Operation
0	1	x 1
1	0	x 16
1	1	x 64

Character Length

0	0	Setting prohibited
0	1	Setting prohibited
1	0	7 bits
1	1	8 bits

Parity Enable

0	Disable
1	Enable

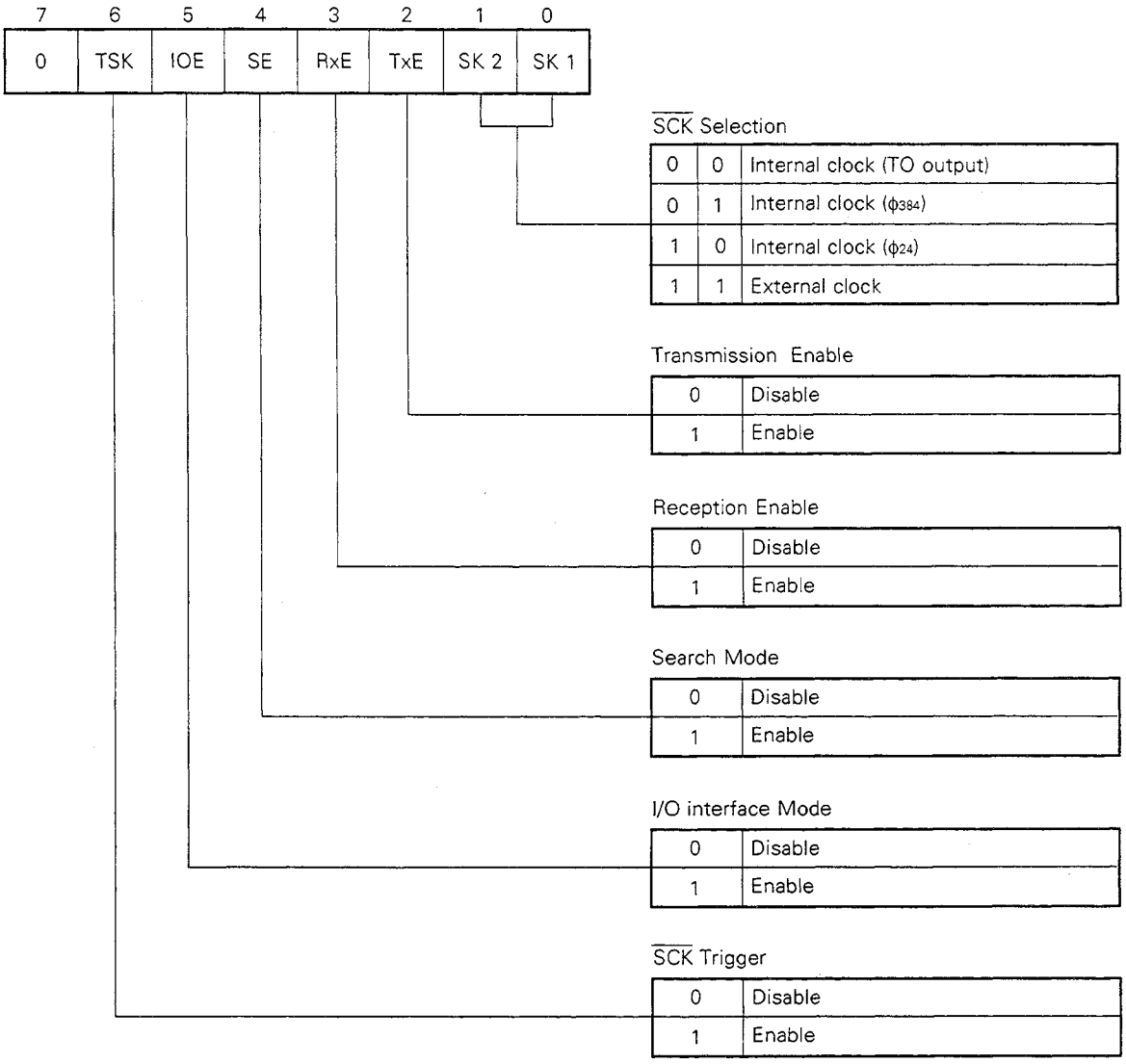
Even Parity Generation/Check

0	Odd number
1	Even number

Number of Stop Bits

0	0	Setting prohibited
0	1	1 bit
1	0	Setting prohibited
1	1	2 bits

Fig. 2-29 Serial Mode High Register (SMH) Format



2.9 ANALOG/DIGITAL CONVERTER

The μPD78C17(A)/78C18(A) have on-chip 8-bit high-speed and high-resolution analog/digital (A/D) converter with 8-multiplexed analog input (AN7 to AN0), and 4 "Conversion Result" registers (CR0 to CR3) to retain a conversion result. This A/D converter uses the successive approximation method.

In the A/D converter operation, either the scan mode or select mode can be selected by software.

In the select mode, one of analog inputs is selected by the A/D channel mode register before starting A/D conversion. Conversion values are stored to CR0 through CR3 sequentially. In the scan mode, Analog conversion values AN0 to AN3 or AN4 to AN7 are stored to CR0 through CR3 sequentially. This mode switchover is specified by the A/D channel mode register.

In case of the select mode, one of the analog inputs is selected by the A/D channel mode register and the A/D conversion starts. Conversion values are stored to CR0 through CR3 sequentially. When four CR registers are set to conversion values, the internal interrupt (INTAD) is generated. The A/D converter continues A/D conversion and sequential storage of conversion values beginning with CR0 until the A/D channel mode register is changed.

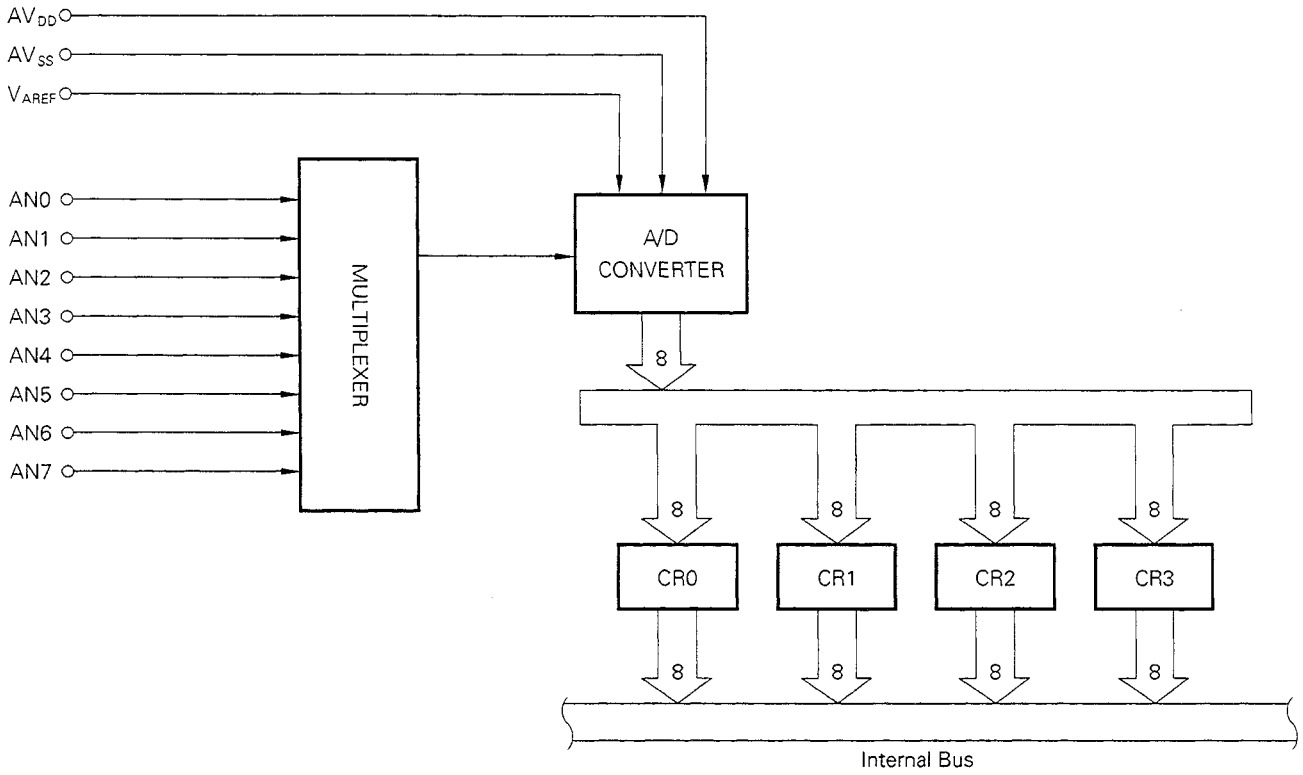
In case of the scan mode, the analog input AN0 to AN3 (ANI2 = 0) or AN4 to AN7 (ANI2 = 1) can be selected.

If bit 3 (ANI2) of the A/D channel mode register is set to "0", analog inputs AN0, AN1, AN2, AN3 and AN0 are selected in that order. These input A/D conversion values CR0, CR1, CR2, CR3, and CR0 are stored in that order. If ANI2 of the A/D channel mode register is set to "1", analog inputs AN4, AN5, AN6, AN7, and AN4 are selected in that order, and these input A/D conversion values CR0, CR1, CR2, CR3, and CR0 in that order. In the scan mode, like in the select mode, when four CR registers are set to conversion values, the internal interrupt (INTAD) is generated.

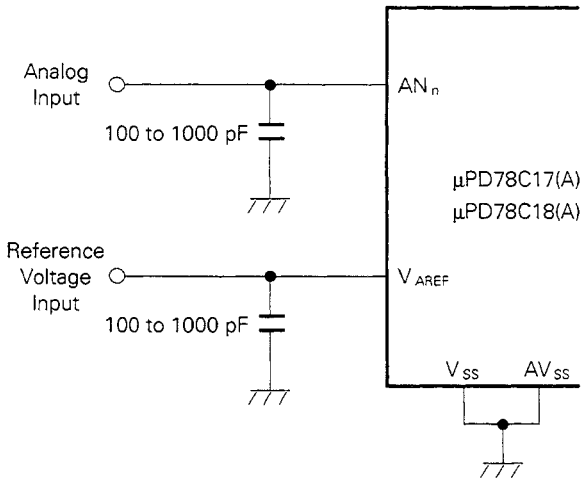
In the scan mode, too, the above-mentioned operation is repeated until the A/D channel mode register is changed.

By setting (1) bit 0 (MKAD) of the interrupt mask register (MKH), the internal interrupt (INTAD) is disabled.

Fig. 2-30 A/D Converter Block Diagram



Caution Capacitors should be connected to the analog input pins and reference voltage input pins in order to prevent mulfunction due to noise.



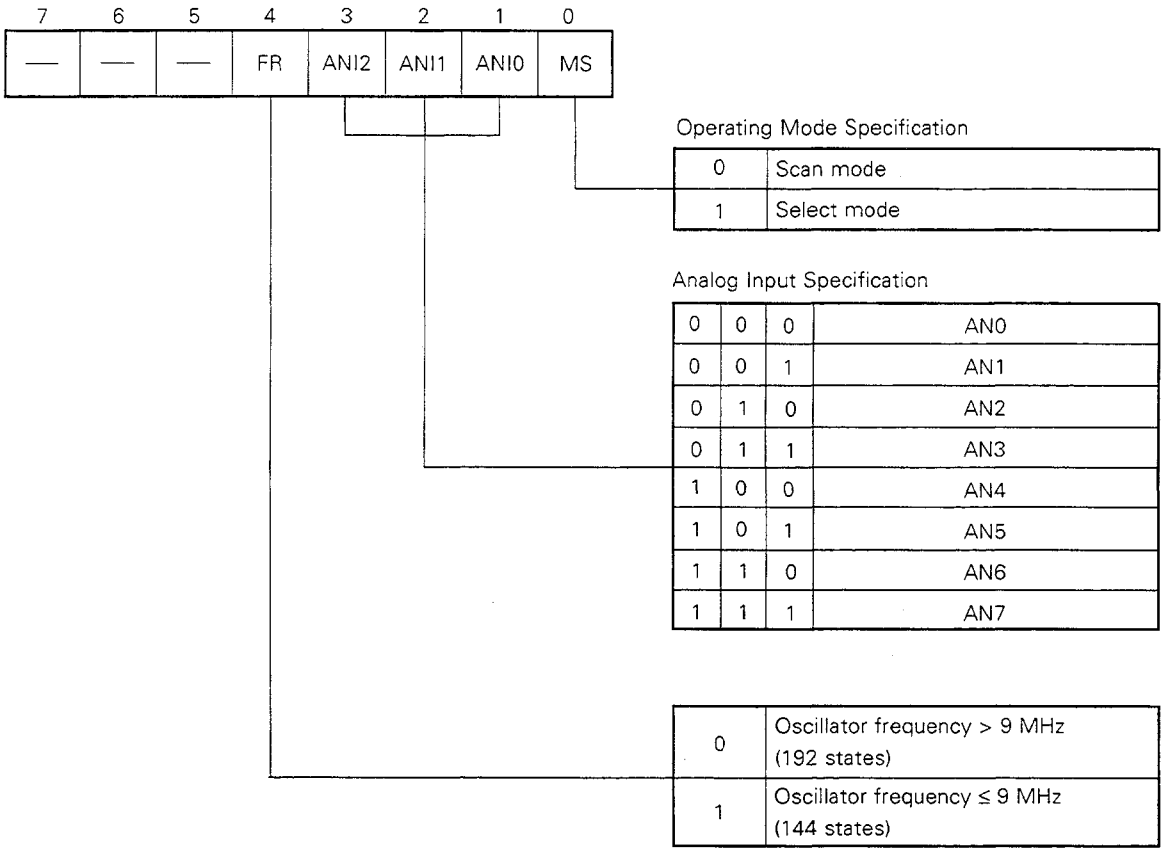
(1) A/D channel mode register (ANM)

This is an 8-bit register which controls A/D converter operation. Bit 0 (MS) of the A/D channel mode register controls the operating mode, bits 1, 2, and 3 (ANI0, ANI1, ANI2) controls A/D conversion input, and bit 4 (FR) controls A/D operation according to change of the oscillator frequency.

In the A/D channel mode register, the operating mode specification is written, and the contents of this register are read. Therefore, in the A/D interrupt generation, analog input data distinction is possible.

After $\overline{\text{RESET}}$ input or in the hardware STOP mode, the A/D channel mode register is set to 00H.

Fig. 2-31 A/D Channel Mode Register Format



(2) A/D converter operation control method

The A/D converter can stop conversion operation by controlling the V_{AREF} input voltage. If a voltage greater than V_{IH1} is input to the V_{AREF} pin, the A/D converter starts conversion operation and the conversion results are guaranteed in V_{AREF} = 3.4 V to AV_{DD}. If the V_{AREF} pin input voltage is set to less than V_{IL1} during the conversion operation, the A/D converter conversion operation stops. At this time, contents of CR0 to CR3 are undefined.

Even if the V_{AREF} input voltage is changed for A/D converter stop control, the A/D channel mode register (ANM) is not affected. Therefore, if the V_{AREF} input voltage is greater than 3.4 V, the A/D converter restarts operation beginning with storage of conversion values to CR0 in the mode directly before the stop state is set.

Even if the V_{AREF} input voltage level is changed, the detection function of AN4 to AN7 input edge is not affected.

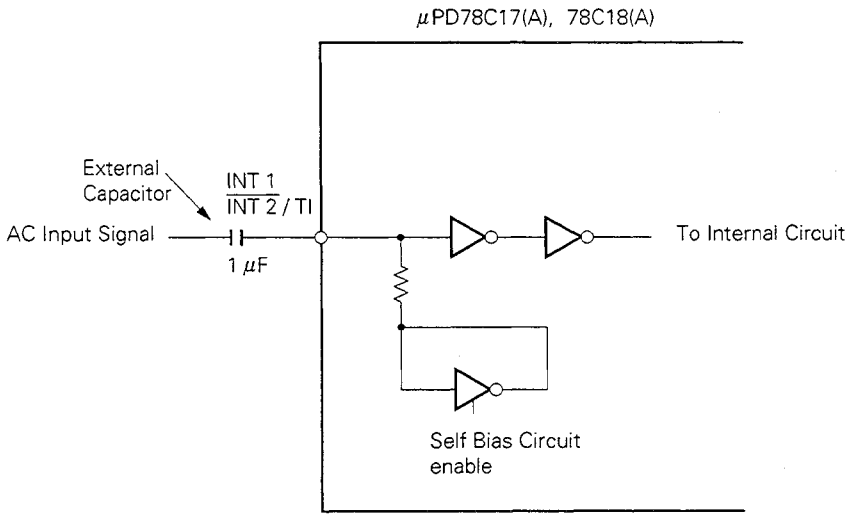
Caution When V_{AREF} is low, inputs AN0 to AN7 in the range of AV_{SS} to AV_{DD} are necessary.

2.10 ZERO-CROSS DETECTOR

The INT1 pin and $\overline{INT2}$/TI (shared as PC3) pin can be made to execute zero-cross detection operations by setting the zero-cross mode register.

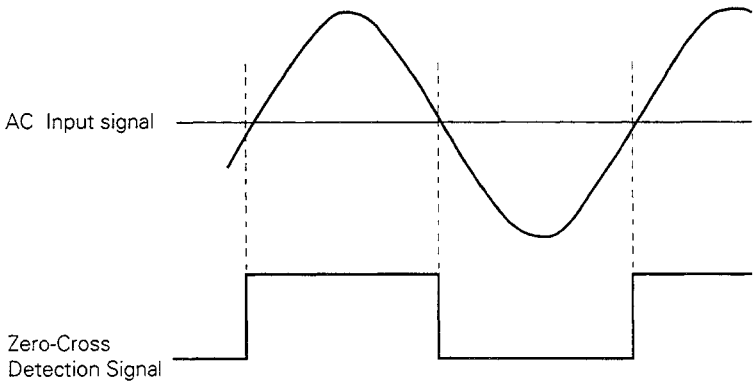
The zero-cross detector has a self-bias type high-gain amplifier. It biases the input to the switching point and generates digital displacement in response to a small input displacement.

Fig. 2-32 Zero-Cross Detector



The zero-cross detector detects a negative-to-positive or positive-to-negative transition of the AC signal input through an external capacitor and generates a digital pulse which changes from 0 to 1 or 1 to 0 at each transition point.

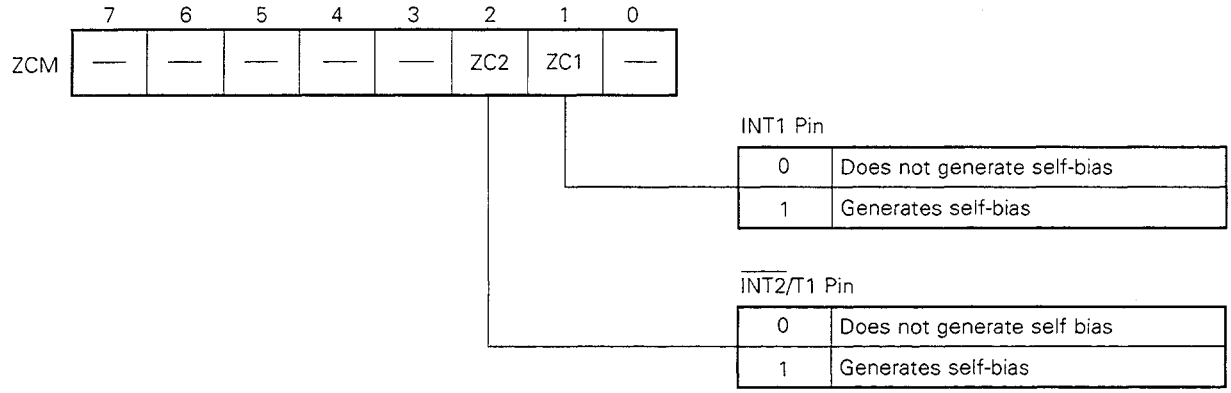
Fig. 2-33 Zero-Cross Detection Signal



A digital pulse generated in the zero-cross detector of the INT1 pin is sent to the interrupt control circuit. The INTF1 interrupt request flag is set at the zero-cross point from the negative to the positive state of the AC signal (rising edge), and if INT1 interrupt is enabled, interrupt servicing is started. A digital pulse generated in the $\overline{INT2}$/TI pin zero-cross detector is sent to the interrupt control circuit and interrupt servicing can be started at the zero-cross point from the positive to the negative state of the AC signal (falling edge) as with the INT1 pin, and can also be used as a timer input clock.

The format of the zero-cross mode register (ZCM), which controls self-bias for zero-cross detection of the INT1 and $\overline{INT2}$/TI pins, is shown in Fig. 2-34.

Fig. 2-34 Zero-Cross Mode Register Format



When the ZC1 and ZC2 bits of the zero-cross mode register are set to "0", a self-bias for zero-cross detection of each pin is not generated and each pin responds as a normal digital input.

When the ZC1 and ZC2 bits are set to "1", a self-bias is generated and an AC input signal zero-cross can be detected by connecting a capacitor to each pin. Each pin with ZC1 and ZC2 bits set to "1" can be directly driven without the use of an external capacitor. In this case, each pin responds as a digital input. However, an input load current is necessary and an external circuit output driver must be considered. Thus, when no zero-cross detection is executed and each pin is used simply as an interrupt input or timer input, the ZC1 and ZC2 bits of the zero-cross mode register should be set to "0".

$\overline{RESET}$ input sets both the ZC1 and ZC2 bits to "1" and a self-bias is generated.

The zero-cross function of the $\overline{INT2}$/TI (shared as PC3) pin can operate only when the control mode is specified by the MODE CONTROL C register (MCC). In the port mode, the zero-cross detection function does not operate.

Caution Unlike other CMOS circuits, a supply current is always present in the zero-cross detector because of its operation points. This also applies in the standby modes (HALT and software/hardware STOP modes). Thus, when the zero-cross detector is operated (with self-bias generation: ZCX = 1), slightly more current flows than without zero-cross detector operation, and its effect is greater in the software/hardware STOP mode.

3. INTERRUPT FUNCTIONS

There are 3 kinds of external interrupt request and 8 kinds of internal interrupt requests. The 11 kinds of interrupt requests are divided into 6 groups, each of which is assigned a different priority and interrupt address.

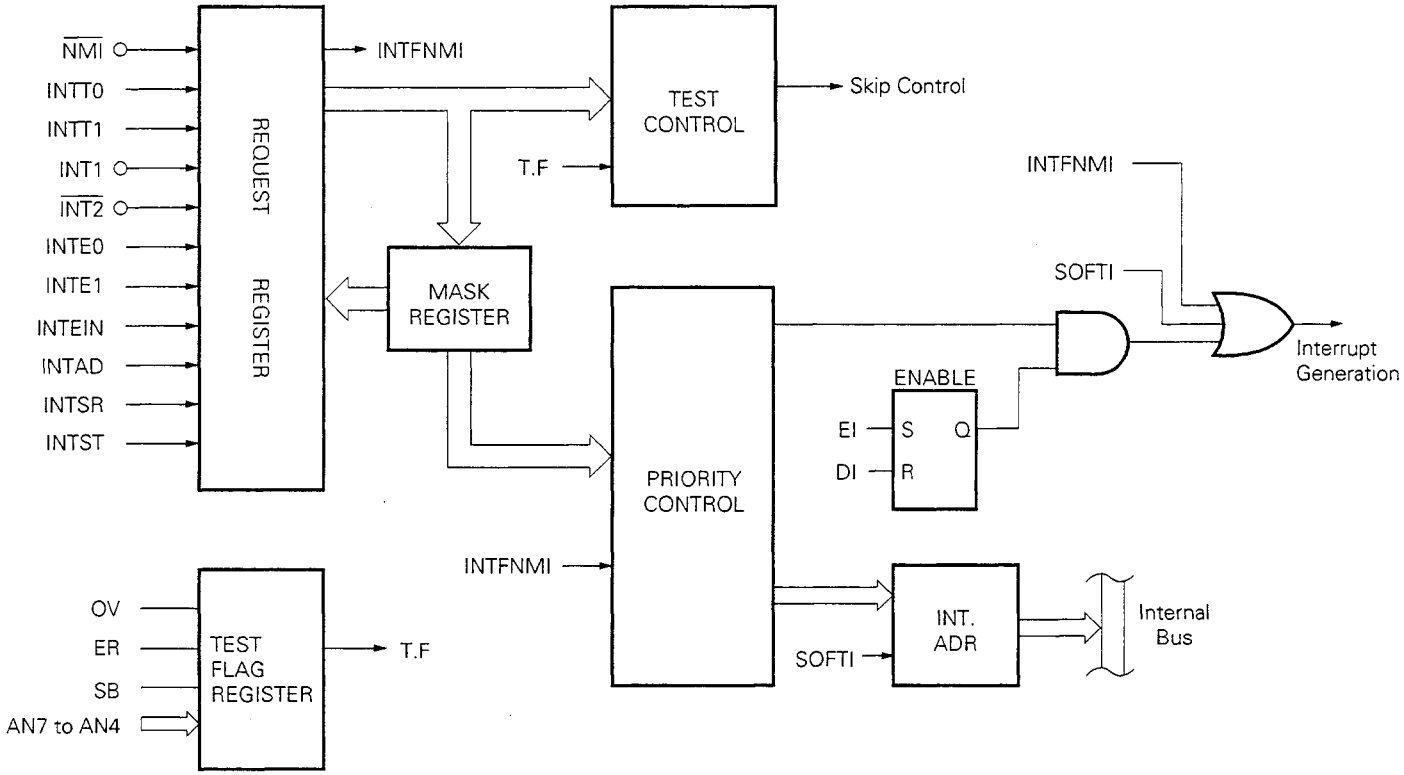
The priority of these interrupt sources and interrupt addresses are as follows.

Priority	Interrupt Address	Interrupt Request		External/ Internal
1	4	$\overline{\text{NMI}}$	Falling edge	External
2	8	INTT0	Match signal from TIMER0	Internal
		INTT1	Match signal from TIMER1	
3	16	INT1	Rising edge	External
		$\overline{\text{INT2}}$	Falling edge	
4	24	INTE0	Match signal from timer/event counter	Internal
		INTE1	Match signal from timer/event counter	
5	32	INTEIN	CI pin or TO fall signal	Internal
		INTAD	A/D converter interrupt	
6	40	INTSR	Serial reception interrupt	Internal
		INTST	Serial transmission interrupt	

3.1 INTERRUPT CONTROL CIRCUIT CONFIGURATION

The interrupt control circuit consists of a request register, a mask register, a priority control, a test control, an interrupt enable F/F, and a test flag register (see **Fig. 3-1**).

Fig. 3-1 Interrupt Control Circuit Block Diagram



(a) REQUEST REGISTER

This register consists of 11 interrupt request flags which are set by the different interrupt requests. A flag is reset when an interrupt request is acknowledged or a skip instruction (SKIT or SKNIT) is executed. $\overline{\text{RESET}}$ input resets all flags.

There are 11 types of interrupt request flags.

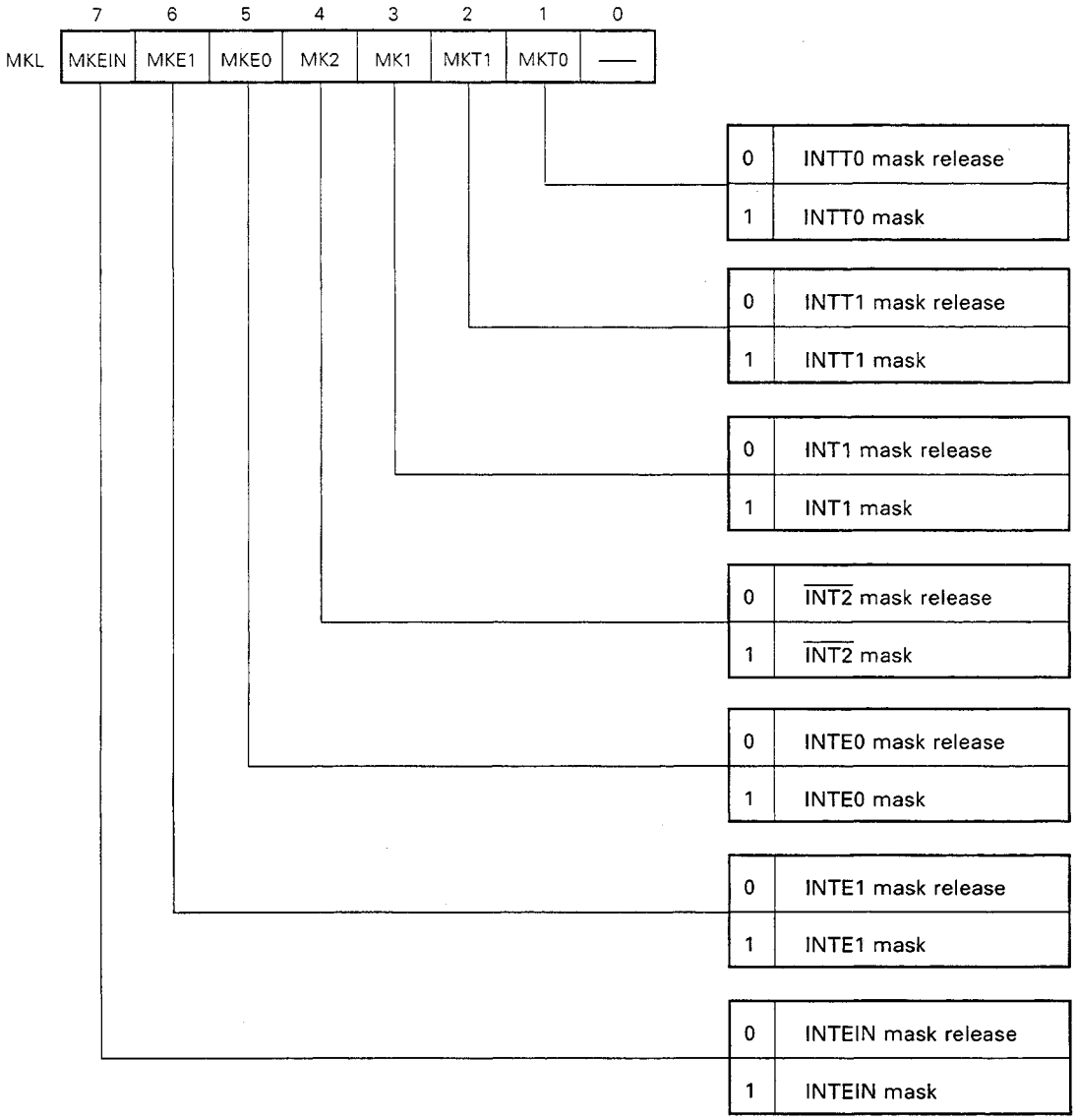
- INTFNMI
  Set (1) by a falling edge input to the $\overline{\text{NMI}}$ pin. Unlike other interrupt request flags, this flag cannot be tested by a skip instruction.
- INTFT0
  Set (1) by TIMER0 COMPARATOR match signal.
- INTFT1
  Set (1) by TIMER1 COMPARATOR match signal.
- INTF1
  Set (1) by a rising edge input to the INT1 pin.
- INTF2
  Set (1) by a falling edge input to the $\overline{\text{INT2}}$ pin.
- INTFE0
  Set (1) by a match signal when timer/event counter ECNT and ETM0 register contents match.
- INTFE1
  Set (1) by a match signal when timer/event counter ECNT and ETM1 register contents match.
- INTFEIN
  Set (1) by a falling edge of the timer/event counter CI input or timer output (TO).
- INTFAD
  Set (1) when A/D converter conversion values are transferred to the four registers CR0 to CR3.
- INTFSR
  Set (1) when the serial interface receive buffer becomes full.
- INTFST
  Set (1) when the serial interface transmit buffer becomes empty.

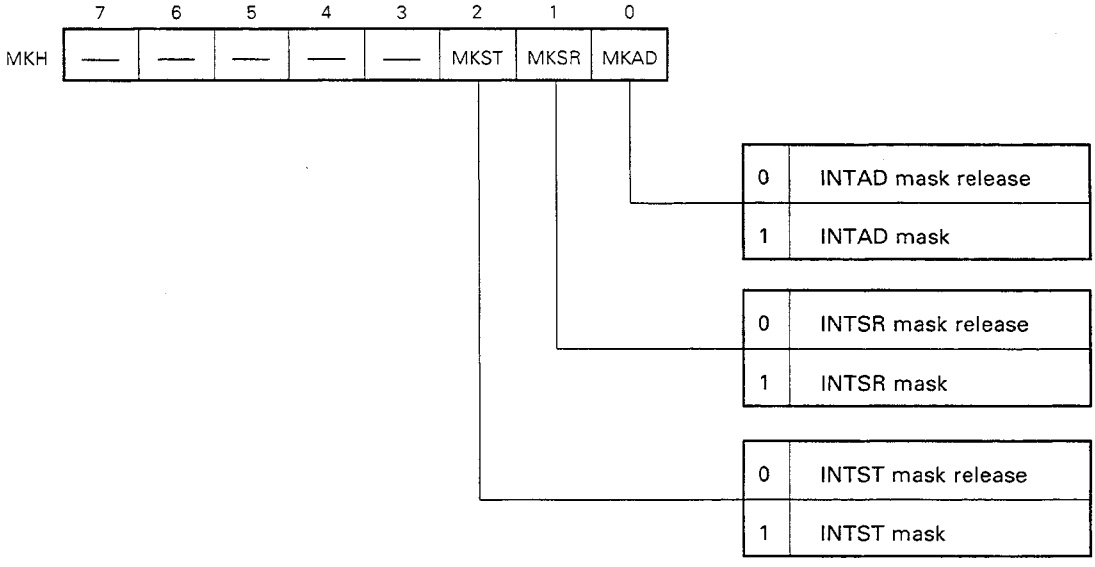
(b) MASK REGISTER

This is a 10-bit mask register which handles all interrupt requests except non-maskable interrupts ($\overline{\text{NMI}}$). It can be set (1) or reset (0) bit-wise by an instruction. An interrupt request is masked (disabled) or enabled when the corresponding bit of the mask register is "1" or "0", respectively.

All bits of the mask register are set by $\overline{\text{RESET}}$ input and all interrupt requests except non-maskable interrupts are masked. All bits of the mask register are set in the hardware STOP mode.

Fig. 3-2 Mask Register (MKL, MKH) Format





(c) PRIORITY CONTROL circuit

This circuit controls the 6 priority levels described earlier. If two or more interrupt request flags are set simultaneously, the interrupt with the highest priority according to the priority is acknowledged.

(d) TEST CONTROL circuit

This circuit comes into operation when a skip instruction (SKIT or SKNIT) is executed to test interrupt request flags (except INTFNMI) for each interrupt source, $\overline{NMI}$ pin states, and test flags.

(e) INTERRUPT ENABLE F/F (IE F/F)

This is a flip-flop which is set by the EI instruction and reset by the DI instruction. This flip-flop is reset when an interrupt is acknowledged, and by $\overline{RESET}$ input, too. Interrupts are enabled when this flip-flop is set, and disabled when it is reset.

(f) TEST FLAG REGISTER

This register consists of 7 test flags which do not generate interrupt requests. These flags are tested or reset by the skip instructions (SKIT, SKNIT).

- **OV**
  Set (1) when the timer/event counter ECNT overflows.
- **ER**
  Set (1) in the event of a parity error, framing error or overrun error in serial interface.
- **SB**
  Set (1) if V_{DD} pin increases from a level lower than specified to a level higher than specified.
- **AN7 to AN4**
  Set (1) by a falling edge input to pins AN7 to AN4.

3.2 NON-MASKABLE INTERRUPT OPERATION

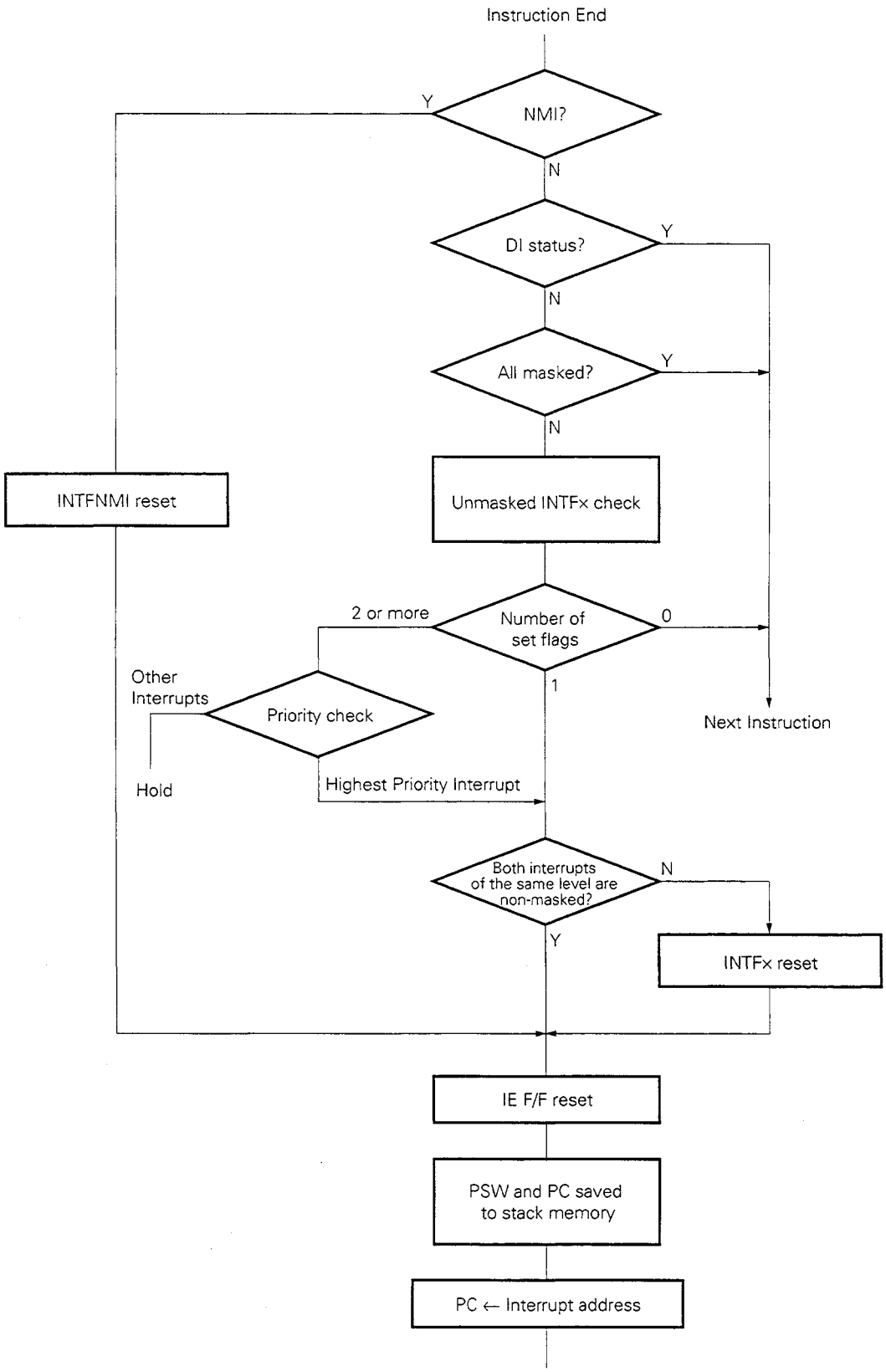
When the interrupt request flag (INTFNMI) is set by a falling edge input to the $\overline{NMI}$ pin, a non-maskable interrupt is acknowledged by means of the following procedure irrespective of the EI/DI state (see **Fig. 3-3**).

(i) A check is made to see if INTFNMI is set at the end of each instruction. If INTFNMI is set, a non-maskable interrupt is acknowledged and INTFNMI is reset.
(ii) When the non-maskable interrupt is acknowledged, the IE F/F is reset and all interrupts except for non-maskable interrupts and the SOFTI instruction are placed in the disabled state (DI state).
(iii) PSW, PC high byte, and PC low byte are saved into the stack memory in that order.
(iv) The program jumps to the interrupt address (0004H).

These interrupt operations are automatically carried out in 16 states.

The interrupt request flag (INTFNMI) cannot be tested by the skip instruction. However, the $\overline{NMI}$ pin status can be tested by the skip instruction (SKIT NMI, SKNIT NMI). Therefore, by testing the $\overline{NMI}$ pin status with the skip instructions in several times in the non-maskable interrupt service routine, noise of comparatively long period or periodical noise can be removed. The $\overline{NMI}$ pin status is not changed even if the status is tested by the skip instruction.

Fig. 3-3 Interrupt Operation Procedure



3.3 MASKABLE INTERRUPT OPERATION

Interrupt requests except non-maskable interrupts and the SOFTI instruction are maskable interrupts which can be enabled/disabled (IE F/F set/reset) by the EI/DI instructions and can be masked individually by means of the mask register.

When an external maskable interrupt is recognized as a normal interrupt signal by an active level input for more than the specified time, an interrupt request flag is set. If an internal interrupt request is generated, an interrupt request flag is immediately set. Once the interrupt request flag is set, both the external and internal interrupts are serviced using the following procedure (see **Fig. 3-3**).

(i) In the EI state (IE F/F = 1), a check is made to see if the interrupt request flag has been set at the end checked at end of each instruction. If the flag has been set, the interrupt cycle starts. However, interrupt requests masked by the mask register are not checked.
(ii) If two or more interrupt request flags have been set simultaneously, their priorities are checked. The interrupt with the highest priority is acknowledged and the others are held pending.
(iii) When an interrupt request is acknowledged, the interrupt request flag is automatically reset. If two types of interrupt requests with the same priority have both been unmasked by the mask register, the interrupt request flag is not reset. This is because the two types are identified by software at a later stage.
(iv) When an interrupt request is acknowledged, the IE F/F is reset, and all interrupts except non-maskable interrupts and the SOFTI instruction are placed in the disabled state (DI state).
(v) The PSW, upper PC byte, and lower PC byte are saved to the stack memory in that order.
(vi) The program jumps to the interrupt address.

These interrupt operations are automatically carried out in 16 states.

The pending interrupt requests are acknowledged if there are no other interrupt requests of higher priority when interrupts are enabled by execution of the EI instruction.

With maskable interrupts there are two types of interrupt requests with the same priority and same interrupt address. Unmasking both types, unmasking one type, or masking both kinds can be selected by setting the mask register.

(1) When both types are unmasked

The corresponding bits of the mask register for two types of interrupt requests are both set to "0". In this case, the interrupt request is the logical sum of the two interrupt request flags.

If an interrupt request is acknowledged in accordance with the interrupt operation as a result of setting one or both interrupt request flags having the same priority and the program jumps to the interrupt address, the interrupt request flag is not reset. Therefore, the interrupt request is identified by executing a skip instruction which tests the interrupt request flag at the beginning of the interrupt service routine, and the interrupt request flag is reset.

(2) When one type is unmasked

For two types of interrupt requests having the same priority, the corresponding bit of the mask register for the interrupt request to be unmasked is set to "0" and the other bit is set to "1". In this case, if an interrupt request is generated by setting the unmasked interrupt request flag and that interrupt request is acknowledged in accordance with the interrupt operation, the interrupt request flag is automatically reset.

When the masked interrupt request flag is set, that interrupt request is held pending. When the pending interrupt request is unmasked, it is acknowledged if there are no other interrupt requests of higher priority in the interrupt enable state.

(3) When both types are masked

The corresponding bits of the mask register for two types of interrupt request are both set to "1". In this case, the interrupt requests are held pending are not acknowledged when the interrupt request flag is set. When the pending interrupt requests are unmasked, they are acknowledged if there are no other interrupt requests of higher priority in the interrupt enabled state.

3.4 INTERRUPT OPERATION BY SOFTI INSTRUCTION

When the SOFTI instruction is executed, the program jumps unconditionally to the interrupt address (0060H). The SOFTI instruction interrupt is not affected by the IE F/F, and the IE F/F is not affected when this instruction is executed.

The servicing procedure for an interrupt generated by the SOFTI instruction is as follows:

(i) The PSW, upper PC byte, and lower PC byte are saved to the stack memory in that order.
(ii) The program jumps to the interrupt address (0060H).

Caution If the skip condition is satisfied by the instruction (arithmetic or logical operation, increment/decrement, shift, skip, or RETS instruction) immediately before the SOFTI instruction, the SOFTI instruction is executed and not skipped. When SOFTI instruction is executed, the SK flag of the PSW is saved as set (1) to the stack area. Thus, when the return is made from the SOFTI service routine, the PSW SK flag remains set and the instruction following the SOFTI instruction is skipped.

4. STANDBY FUNCTIONS

Three standby modes are available for the μPD78C17(A)/78C18(A) to save power consumption in the program standby mode (the HALT mode, software STOP mode, and hardware STOP mode).

4.1 HALT MODE

When the HLT instruction is executed, the HALT mode is set unless the interrupt request flag of the unmasked interrupt is set. In the HALT mode the CPU clock stops and program execution also stops. However, the contents of all registers and internal RAM just before the stoppage are retained. In the HALT mode, the timer, timer/event counter, serial interface, A/D converter, and interrupt control circuit are operational.

Table 4-1 shows the status of the μPD78C17(A)/78C18(A) output pins in the HALT mode.

Table 4-1 Output Pin Statuses

Output Pin	Single Chip[Note1]	External Expansion
PA7 to PA0	Data retained	Data retained
PB7 to PB0	Data retained	Data retained
PC7 to PC0	Data retained	Data retained
PD7 to PD0	Data retained	High-impedance
PF7 to PF0	Data retained	Next address retained[Note2] Data retained[Note3]
$\overline{WR}$, $\overline{RD}$	High-level	High-level
ALE	High-level	High-level

Notes 1. μPD78C18(A) only
2. Address output pin
3. Port data output pin

Caution Because an interrupt request flag is used to release the HALT mode, HLT instruction execution does not set the HALT mode if even a single interrupt request flag for an unmasked interrupt is set. Thus, when setting the HALT mode when there is a possibility that an interrupt request flag may have been set (when there is a pending interrupt), one of the following procedures should be followed: First process the pending interrupt; or, reset the interrupt request flag by executing a skip instruction; or, mask all interrupts except those used to release the HALT mode.

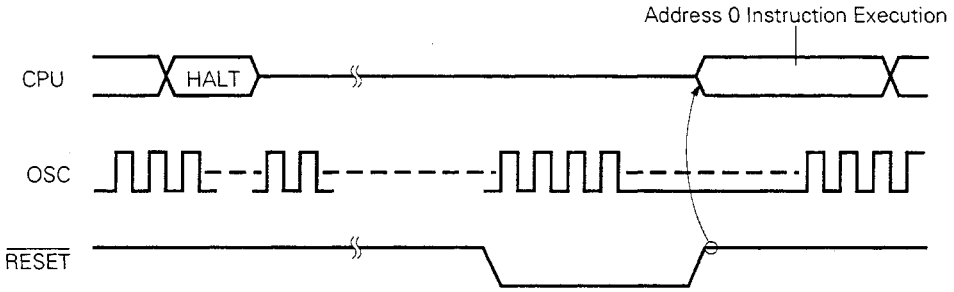
4.2 HALT MODE RELEASE

(1) Release by $\overline{\text{RESET}}$ signal

When the $\overline{\text{RESET}}$ signal changes from the high to low level in the HALT mode, the HALT mode is released and the reset state is set. When the $\overline{\text{RESET}}$ signal returns to the high level, the CPU starts program execution at address 0.

When the $\overline{\text{RESET}}$ signal is input, the RAM contents are retained but the contents of other registers are undefined.

Fig. 4-1 HALT Mode Release Timing ($\overline{\text{RESET}}$ Signal Input)



(2) Release by interrupt request flag

The HALT mode is released if at least one interrupt request flag is set by the generation of a non-maskable interrupt ($\overline{NMI}$) or one of ten unmasked maskable interrupts (INTT0, INTT1, INT1, $\overline{INT2}$, INTE0, INTE1, INTEIN, INTAD, INTST, and INTSR).

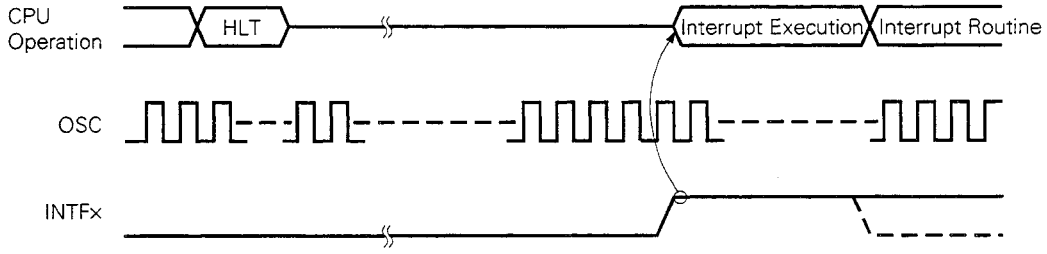
When the HALT mode is released by a non-maskable interrupt, the instruction following the HLT instruction is not executed and the program jumps to the interrupt address (0004H) irrespective of the interrupt enabled/disabled (EI/DI) state.

When the HALT mode is released by a maskable interrupt, operation after release differs depending on whether the EI or DI state is set.

(i) EI state

The instruction following the HLT instruction is not executed and the program jumps to the corresponding interrupt address.

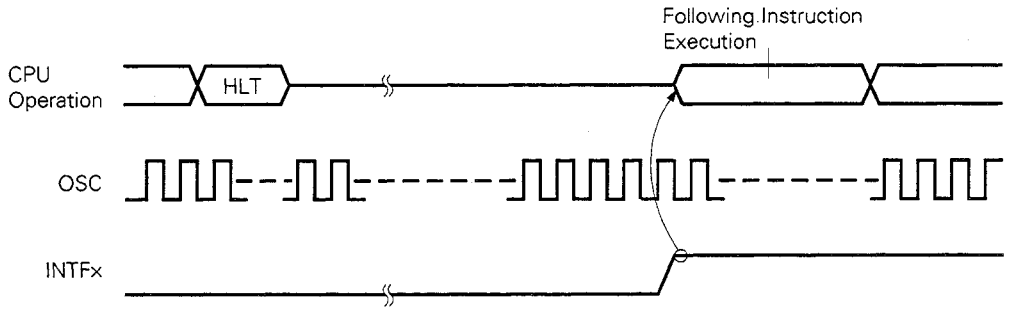
Fig. 4-2 HALT Mode Release Timing (in EI State)



(ii) DI state

Execution restarts with the instruction following the HLT instruction (without jumping to the interrupt address). Because the interrupt request flag used for release remains set, it should be reset by a skip instruction when required.

Fig. 4-3 HALT Mode Release Timing (in DI State)



4.3 SOFTWARE STOP MODE

When the STOP instruction is executed, the software STOP mode is set unless the interrupt request flag for an unmasked external interrupt is set. In the software STOP mode, all clocks stop. When this mode is set, program execution stops and the contents of all registers and internal RAM are retained (the timer upcounter is cleared to 00H). Only the $\overline{NMI}$ and $\overline{RESET}$ signals used to release the software STOP mode are valid, and all other functions stop.

The statuses of the μPD78C17(A)/78C18(A) output pins in the software STOP mode are the same as for the HALT mode, as shown in Table 4-1.

Cautions 1. Internal interrupts should be masked before executing the STOP instruction to prevent errors due to an internal interrupt during the oscillation stabilization time at release of the software STOP mode.

2. The TIMER1 coincidence signal is used as the signal to start CPU operation to secure an oscillation stabilization period after the software STOP mode has been released by setting the non-maskable interrupt request flag. Thus, it is necessary to set a count value in TIMER REG which takes account of the oscillation stabilization time, and to set the timer mode register to the timer operating state, before executing the STOP instruction.

4.4 SOFTWARE STOP MODE RELEASE

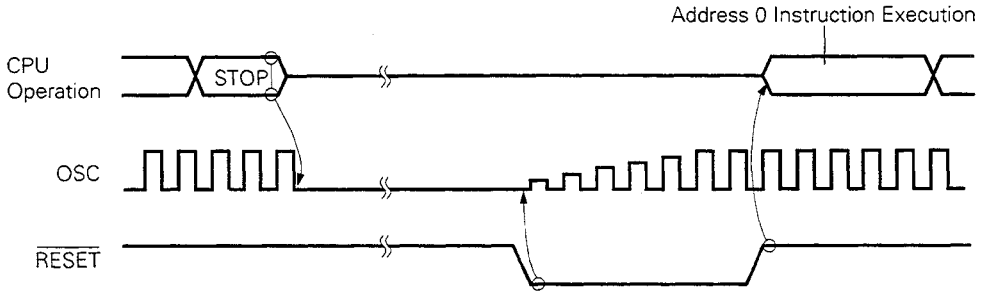
(1) Release by $\overline{RESET}$ signal

When the $\overline{RESET}$ signal changes from the high to low level in the software STOP mode, the software STOP mode is released and clock oscillation starts as soon as the reset state is set. When the $\overline{RESET}$ signal is driven high after oscillation has stabilized, the CPU starts program execution at address 0.

When the $\overline{RESET}$ signal changes from the high to low level, clock oscillation starts but it takes time for oscillation to stabilize. The $\overline{RESET}$ signal low-level width must therefore be longer than the oscillation stabilization time.

When the $\overline{RESET}$ signal is input, the RAM contents are retained but the contents of other registers are undefined.

Fig. 4-4 Software STOP Mode Release Timing ($\overline{RESET}$ Input)



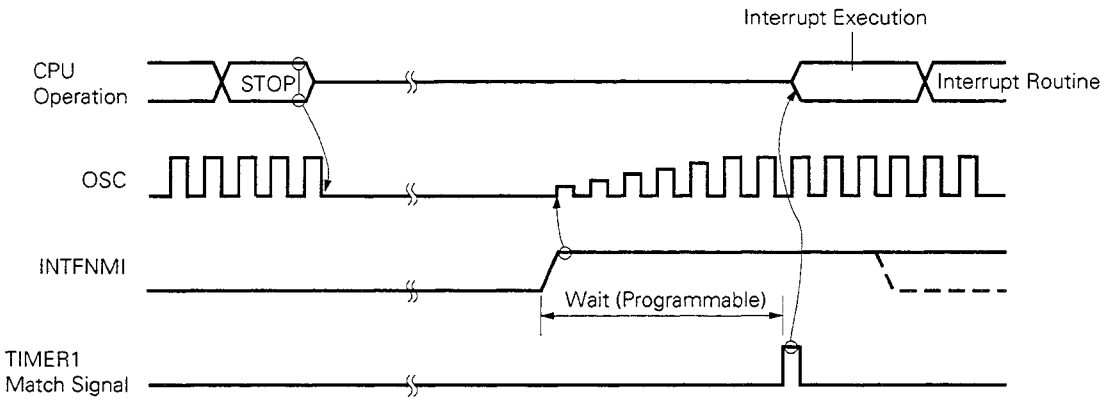
If the software STOP mode is released by the $\overline{RESET}$ signal, program execution starts at address 0 as in the case of a normal power-on reset. The SB (Standby) flag can be used to identify the program execution mode. The SB flag is set (1) when the V_{DD} pin rises from the specified low level or below to the specified high level or above, and is reset (0) by executing a skip instruction. Thus, by testing the SB flag using a skip instruction in the program executed after $\overline{RESET}$ input, a set SB flag indicates a power-on start, and a reset SB flag indicates a start due to release of the software STOP mode.

(2) Release by interrupt request flag

When the non-maskable interrupt request flag is set in the software STOP mode, the software STOP mode is released and simultaneously clock oscillation starts. When clock oscillation starts, the timer upcounter starts counting up from 00H in accordance with the setting before execution of the STOP instruction. CPU operation is started by a match signal (wait time taking account of the oscillation stabilization time) from the TIMER1 UPCOUNTER. In this case, the UPCOUNTER match signal does not set the interrupt request flag. The timer mode register of the timer after generation of the match signal is set to FFH and timer operation is stopped.

After the elapse of the oscillation stabilization time, the program jumps to the interrupt address (0004H) irrespective of the interrupt enabled/ disabled (EI/DI) state and without executing the instruction following the STOP instruction.

Fig. 4-5 Software STOP Mode Release Timing



4.5 HARDWARE STOP MODE

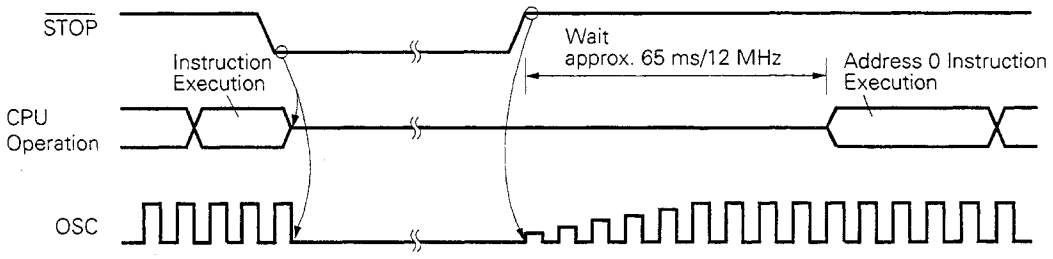
When the $\overline{\text{STOP}}$ signal changes from the high to low level, the hardware STOP mode is set. In this mode all clocks stop. When the hardware STOP mode is set, program execution stops and the internal RAM contents just before stoppage are retained, and the $\overline{\text{STOP}}$ signal used to release the hardware STOP mode is valid. All other functions stop and the reset state is set.

In the hardware STOP mode, the μPD78C17(A)/78C18(A) output pins become high-impedance.

4.6 HARDWARE STOP MODE RELEASE

When the $\overline{\text{STOP}}$ signal changes from the low to high level in the hardware STOP mode, the hardware STOP mode is released and simultaneously clock oscillation starts. After the elapse of the wait time (approximately 65 ms at 12 MHz) which takes account of the oscillation stabilization time, the CPU starts program execution at address 0 (see **Fig. 4-6**).

Fig. 4-6 Hardware STOP Mode Release Timing



The hardware STOP mode is not released by a high-to-low transition of the $\overline{\text{RESET}}$ signal. When the $\overline{\text{STOP}}$ signal changes from low to high while the $\overline{\text{RESET}}$ signal is low, the hardware STOP mode is released and clock oscillation starts. If the $\overline{\text{RESET}}$ signal returns from the low to high level, the CPU starts program execution at address 0 without waiting for the elapse of the oscillation stabilization time (see **Fig. 4-7**). For also the case where the $\overline{\text{RESET}}$ signal changes from high to low immediately after the hardware STOP mode is released (the $\overline{\text{STOP}}$ signal changes from low to high), the program is executed when the $\overline{\text{RESET}}$ signal returns from low to high (see **Fig. 4-8**).

The oscillation stabilization time should therefore be taken into account when returning the $\overline{\text{RESET}}$ signal to the high level.

After $\overline{\text{RESET}}$ signal input RAM contents are retained, but the contents of other registers are undefined.

Fig. 4-7 Hardware STOP Mode Release Timing

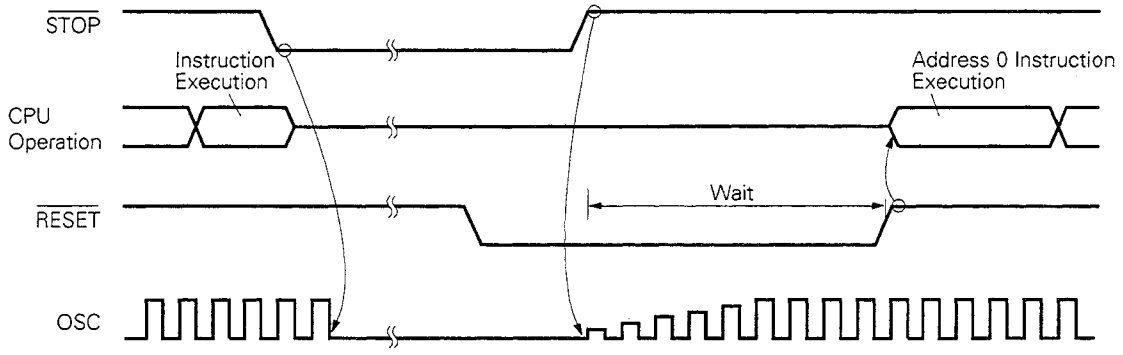
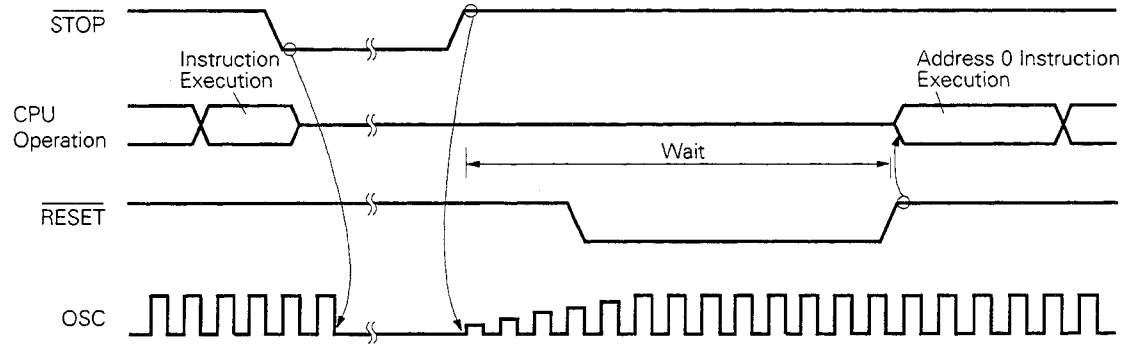


Fig. 4-8 Hardware STOP Mode Release Timing



In the case of a hardware STOP mode release, as with a release of the software STOP mode by means of the $\overline{\text{RESET}}$ signal, it is possible to differentiate between a power-on start and a start due to release of the hardware STOP mode by testing the SB flag using a skip instruction.

4.7 LOW SUPPLY VOLTAGE DATA RETENTION MODE

The low supply voltage data retention mode can be set by decreasing the V_{DD} supply voltage to 2.5 V after setting the software/hardware STOP mode. RAM contents can be retained with lower power consumption than in the software/hardware STOP mode.

Caution The software/hardware STOP mode should not be released while in the low supply voltage data retention mode. V_{DD} must be raised to the normal operating voltage before the release is performed.

5. RESET OPERATIONS

When $\overline{\text{RESET}}$ input becomes low, then system reset is activated to create the following status.

- o INTERRUPT ENABLE F/F is reset and interrupt is disabled.
- o All the interrupt mask registers are set (1) and interrupt is masked.
- o An interrupt request flag is reset (0) and hold interrupt is eliminated.
- o All PSWs are reset (0).
- o 0000H is loaded into the program counter (PC).
- o The MODE A, MODE B, MODE C, and MODE F registers are set to FFH and the bits (MM0, 1, and 2) of the MODE CONTROL C and MEMORY MAPPING registers are respectively reset (0), then all the ports (A, B, C, D, and F) become input port (high-impedance).
- o All the test flags but SB flag are reset (0).
- o A timer mode register is set to FFH, and TIMER F/F is reset.
- o The mode register (ETMM, EOM) of a timer/event counter is reset (0).
- o The serial mode high register(SMH) of serial interface is reset (0), while the serial mode low register (SML) is set to 48H.
- o The A/D channel mode register of the A/D converter is reset (0).
- o $\overline{\text{WR}}$, $\overline{\text{RD}}$, ALE signals become high-impedance.
- o The ZC1, ZC2 bits of the zero-cross mode register (ZCM) are set (1).
- o Data memory and the following register contents are undefined.
- o The internal timing generator is initialized.

Stack pointer (SP)
Expansion accumulator (EA, EA'), accumulator (A, A')
General register (B, C, D, E, H, L, B', C', D', E', H', L')
Output latch of each port
TIMER REG0, 1 (TM0, TM1)
TIMER/EVENT COUNTER REG0, 1 (ETM0, ETM1)
RAE bit of MEMORY MAPPING register
SB flag of test flag

When $\overline{\text{RESET}}$ input becomes high, the reset status is released. Then, execution of the program is started from 0000H. The contents of various kinds of registers must be initialized or re-initialized in the program, if necessary.

6. INSTRUCTION SET

6.1 IDENTIFIER/DESCRIPTION OF OPERAND

Identifier	Description
r r1 r2	V, A, B, C, D, E, H, L EAH, EAL, B, C, D, E, H, L A, B, C
sr sr1 sr2 sr3 sr4	PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, SML, EOM, ETMM, TMM, MM, MCC, MA, MB, MC, MF, TXB, TM0, TM1, ZCM PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM, RXB, CR0, CR1, CR2, CR3 PA, PB, PC, PD, PF, MKH, MKL, ANM, SMH, EOM, TMM ETM0, ETM1 ECMT, ECPT
rp rp1 rp2 rp3	SP, B, D, H V, B, D, H, EA SP, B, D, H, EA B, D, H
rpa rpa1 rpa2 rpa3	B, D, H, D+, H+, D–, H– B, D, H B, D, H, D+, H+, D–, H–, D+byte, H+A, H+B, H+EA, H+byte D, H, D++, H++, D+byte, H+A, H+B, H+EA, H+byte
wa	8-bit immediate data
word byte bit	16-bit immediate data 8-bit immediate data 3-bit immediate data
f	CY, HC, Z
irf	NMI[Note], FT0, FT1, F1, F2, FE0, FE1, FEIN, FAD, FSR, FST, ER, OV, AN4, AN5, AN6, AN7, SB

Note NMI can also be described as FNMI.

Remarks

1. sr to sr4 (special register)

PA	:	PORT A	ETMM	:	TIMER/EVENT COUNTER MODE
PB	:	PORT B	EOM	:	TIMER/EVENT COUNTER OUTPUT MODE
PC	:	PORT C	ANM	:	A/D CHANNEL MODE
PD	:	PORT D	CR0 to CR3	:	A/D CONVERSION RESULT 0 to 3
PF	:	PORT F	TXB	:	Tx BUFFER
MA	:	MODE A	RXB	:	Rx BUFFER
MB	:	MODE B	SMH	:	SERIAL MODE High
MC	:	MODE C	SML	:	SERIAL MODE Low
MCC	:	MODE CONTROL C	MKH	:	MASK High
MF	:	MODE F	MKL	:	MASK Low
MM	:	MEMORY MAPPING	ZCM	:	ZERO CROSS MODE
TM0	:	TIMER REG0			
TM1	:	TIMER REG1			
TMM	:	TIMER MODE			
ETM0	:	TIMER/EVENT COUNTER REG0			
ETM1	:	TIMER/EVENT COUNTER REG1			
ECNT	:	TIMER/EVENT COUNTER UPCOUNTER			
ECPT	:	TIMER/EVENT COUNTER CAPTURE			

2. rp to rp3 (register pair)

SP	:	STACK POINTER
B	:	BC
D	:	DE
H	:	HL
V	:	VA
EA	:	EXTENDED ACCUMULATOR

3. rpa to rpa3 (rp addressing)

B	:	(BC)
D	:	(DE)
H	:	(HL)
D+	:	(DE)+
H+	:	(HL)+
D–	:	(DE)–
H–	:	(HL)–
D++	:	(DE)++
H++	:	(HL)++
D + byte	:	(DE + byte)
H + A	:	(HL + A)
H + B	:	(HL + B)
H + EA	:	(HL + EA)
H + byte	:	(HL + byte)

4. f (flag)

CY	:	CARRY
HC	:	HALF CARRY
Z	:	ZERO

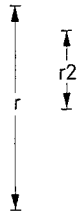
5. irf (interrupt flag)

NMI	:	NMI INPUT
FT0	:	INTFT0
FT1	:	INTFT1
F1	:	INTF1
F2	:	INTF2
FE0	:	INTFE0
FE1	:	INTFE1
FEIN	:	INTFEIN
FAD	:	INTFAD
FSR	:	INTFSR
FST	:	INTFST
ER	:	ERROR
OV	:	OVERFLOW
AN4 to AN7	:	ANALOG INPUT 4 to 7
SB	:	STANDBY

6.2 SYMBOL DESCRIPTION OF INSTRUCTION CODE

r

R_2	R_1	R_0	reg
0	0	0	V
0	0	1	A
0	1	0	B
0	1	1	C
1	0	0	D
1	0	1	E
1	1	0	H
1	1	1	L

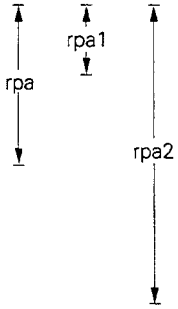


r1

T_2	T_1	T_0	reg
0	0	0	EAH
0	0	1	EAL
0	1	0	B
0	1	1	C
1	0	0	D
1	0	1	E
1	1	0	H
1	1	1	L

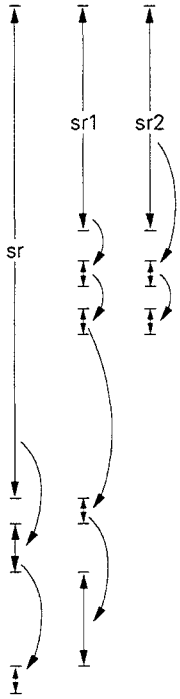
rpa

A_3	A_2	A_1	A_0	addressing
0	0	0	0	———
0	0	0	1	(BC)
0	0	1	0	(DE)
0	0	1	1	(HL)
0	1	0	0	(DE)+
0	1	0	1	(HL)+
0	1	1	0	(DE)-
0	1	1	1	(HL)-
1	0	1	1	(DE + byte)
1	1	0	0	(HL + A)
1	1	0	1	(HL + B)
1	1	1	0	(HL + EA)
1	1	1	1	(HL + byte)



sr

S_5	S_4	S_3	S_2	S_1	S_0	Special-reg
0	0	0	0	0	0	PA
0	0	0	0	0	1	PB
0	0	0	0	1	0	PC
0	0	0	0	1	1	PD
0	0	0	1	0	1	PF
0	0	0	1	1	0	MKH
0	0	0	1	1	1	MKL
0	0	1	0	0	0	ANM
0	0	1	0	0	1	SMH
0	0	1	0	1	0	SML
0	0	1	0	1	1	EOM
0	0	1	1	0	0	ETMM
0	0	1	1	0	1	TMM
0	1	0	0	0	0	MM
0	1	0	0	0	1	MCC
0	1	0	0	1	0	MA
0	1	0	0	1	1	MB
0	1	0	1	0	0	MC
0	1	0	1	1	1	MF
0	1	1	0	0	0	TXB
0	1	1	0	0	1	RXB
0	1	1	0	1	0	TM0
0	1	1	0	1	1	TM1
1	0	0	0	0	0	CR0
1	0	0	0	0	1	CR1
1	0	0	0	1	0	CR2
1	0	0	0	1	1	CR3
1	0	1	0	0	0	ZCM



rpa3

C_3	C_2	C_1	C_0	addressing
0	0	1	0	(DE)
0	0	1	1	(HL)
0	1	0	0	(DE)++
0	1	0	1	(HL)++
1	0	1	1	(DE + byte)
1	1	0	0	(HL + A)
1	1	0	1	(HL + B)
1	1	1	0	(HL + EA)
1	1	1	1	(HL + byte)

irf

I_4	I_3	I_2	I_1	I_0	INTF
0	0	0	0	0	NMI
0	0	0	0	1	FT0
0	0	0	1	0	FT1
0	0	0	1	1	F1
0	0	1	0	0	F2
0	0	1	0	1	FE0
0	0	1	1	0	FE1
0	0	1	1	1	FEIN
0	1	0	0	0	FAD
0	1	0	0	1	FSR
0	1	0	1	0	FST
0	1	0	1	1	ER
0	1	1	0	0	OV
1	0	0	0	0	AN4
1	0	0	0	1	AN5
1	0	0	1	0	AN6
1	0	0	1	1	AN7
1	0	1	0	0	SB

sr3

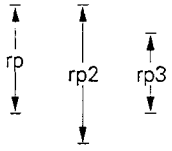
U_0	special-reg
0	ETM0
1	ETM1

sr4

V_0	special-reg
0	ECNT
1	ECPT

rp

P_2	P_1	P_0	reg-pair
0	0	0	SP
0	0	1	BC
0	1	0	DE
0	1	1	HL
1	0	0	EA



rp1

Q_2	Q_1	Q_0	reg-pair
0	0	0	VA
0	0	1	BC
0	1	0	DE
0	1	1	HL
1	0	0	EA

f

F_2	F_1	F_0	flag
0	0	0	—
0	1	0	CY
0	1	1	HC
1	0	0	Z

6.3 INSTRUCTION EXECUTION TIME

One state indicated in this section consists of three clock cycles. For example, one state takes 200 ns (1/15 ns × 3) at 15-MHz operation, and when executing a 4-state instruction, the minimum execution time is 0.8 μs.

Note 1	Mnemonic		Operand	Instruction Code B1	B2	B3	B4	State	Operation	Skip Condition
8-bit data transfer instructions	MOV		r1, A	0 0 0 1 1 T_2 T_1 T_0				4	r1 ← A	
			A, r1	0 0 0 0 1 T_2 T_1 T_0				4	A ← r1	
		*	sr, A	0 1 0 0 1 1 0 1	1 1 S_5 S_4 S_3 S_2 S_1 S_0			10	sr ← A	
		*	A, sr1	0 1 0 0 1 1 0 0	1 1 S_5 S_4 S_3 S_2 S_1 S_0			10	A ← sr1	
			r, word	0 1 1 1 0 0 0 0	0 1 1 0 1 R_2 R_1 R_0	Low Adrs	High Adrs	17	r ← (word)	
			word, r	0 1 1 1 0 0 0 0	0 1 1 1 1 R_2 R_1 R_0	Low Adrs	High Adrs	17	(word) ← r	
	MVI	*	r, byte	0 1 1 0 1 R_2 R_1 R_0	← Data →			7	r ← byte	
			sr2, byte	0 1 1 0 0 1 0 0	S_3 0 0 0 0 S_2 S_1 S_0	Data		14	sr2 ← byte	
	MVIW	*	wa, byte	0 1 1 1 0 0 0 1	← Offset →	Data		13	(V. wa) ← byte	
	MVIX	*	rpa1, byte	0 1 0 0 1 0 A_1 A_0	← Data →			10	(rpa1) ← byte	
	STAW	*	wa	0 1 1 0 0 0 1 1	← Offset →			10	(V. wa) ← A	
	LDAW	*	wa	0 0 0 0 0 0 0 1	← Offset →			10	A ← (V. wa)	
	STAX	*	rpa2	A_3 0 1 1 1 A_2 A_1 A_0	Data*[1]			7/13*[3]	(rpa2) ← A	
	LDAX	*	rpa2	A_3 0 1 0 1 A_2 A_1 A_0	Data*[1]			7/13*[3]	A ← (rpa2)	
	EXX			0 0 0 1 0 0 0 1				4	B ↔ B', C ↔ C', D ↔ D' E ↔ E', H ↔ H', L ↔ L'	
	EXA			0 0 0 1 0 0 0 0				4	V, A ↔ V', A', EA ↔ EA'	
	EXH			0 1 0 1 0 0 0 0				4	H, L ↔ H', L'	
	BLOCK			0 0 1 1 0 0 0 1				13 (C + 1)	(DE)+ ← (HL)+, C ← C – 1 End if borrow	
Note 2	DMOV		rp3, EA	1 0 1 1 0 1 P_1 P_0				4	$rp3_L$ ← EAL, $rp3_H$ ← EAH	
			EA, rp3	1 0 1 0 0 1 P_1 P_0				4	EAL ← $rp3_L$, EAH ← $rp3_H$	

Notes
1. Instruction Group
2. 16-bit data transfer instructions

Note 1	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
16-bit data transfer instructions	DMOV	sr3, EA	0 1 0 0 1 0 0 0	1 1 0 1 0 0 1 U_0			14	sr3 ← EA	
		EA, sr4	↓	1 1 0 0 0 0 0 V_0			14	EA ← sr4	
	SBCD	word	0 1 1 1 0 0 0 0	0 0 0 1 1 1 1 0	Low Adrs	High Adrs	20	(word) ← C, (word + 1) ← B	
	SDED	word	↓	0 0 1 0 1 1 1 0	↓	↓	20	(word) ← E, (word + 1) ← D	
	SHLD	word	↓	0 0 1 1 1 1 1 0	↓	↓	20	(word) ← L, (word + 1) ← H	
	SSPD	word	↓	0 0 0 0 1 1 1 0	↓	↓	20	(word) ← SP_L, (word + 1) ← SP_H	
	STEAX	rpa3	0 1 0 0 1 0 0 0	1 0 0 1 C_3 C_2 C_1 C_0	Data*2		14/20*3	(rpa3) ← EAL, (rpa3 + 1) ← EAH	
	LBCD	word	0 1 1 1 0 0 0 0	0 0 0 1 1 1 1 1	Low Adrs	High Adrs	20	C ← (word), B ← (word + 1)	
	LDED	word	↓	0 0 1 0 1 1 1 1	↓	↓	20	E ← (word), D ← (word + 1)	
	LHLD	word	↓	0 0 1 1 1 1 1 1	↓	↓	20	L ← (word), H ← (word + 1)	
	LSPD	word	↓	0 0 0 0 1 1 1 1	↓	↓	20	SP_L ← (word), SP_H ← (word + 1)	
	LDEAX	rpa3	0 1 0 0 1 0 0 0	1 0 0 0 C_3 C_2 C_1 C_0	Data*2		14/20*3	EAL ← (rpa3), EAH ← (rpa3 + 1)	
	PUSH	rp1	1 0 1 1 0 Q_2 Q_1 Q_0				13	(SP – 1) ← $rp1_H$, (SP – 2) ← $rp1_L$ SP ← SP – 2	
	POP	rp1	1 0 1 0 0 Q_2 Q_1 Q_0				10	$rp1_L$ ← (SP), $rp1_H$ ← (SP + 1) SP ← SP + 2	
	LXI *	rp2, word	0 P_2 P_1 P_0 0 1 0 0	← Low Byte →	High Byte		10	rp2 ← word	
	TABLE		0 1 0 0 1 0 0 0	1 0 1 0 1 0 0 0			17	C ← (PC + 3 + A) B ← (PC + 3 + A + 1)	
Note 2	ADD	A, r	0 1 1 0 0 0 0 0	1 1 0 0 0 R_2 R_1 R_0			8	A ← A + r	
		r, A	↓	0 1 0 0			8	r ← r + A	
	ADC	A, r	↓	1 1 0 1			8	A ← A + r + CY	
		r, A	↓	0 1 0 1			8	r ← r + A + CY	

Notes
1. Instruction Group
2. 8-bit operation instructions (register)

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
8-bit operation instructions (register)	ADDNC	A, r	0 1 1 0 0 0 0 0	1 0 1 0 0 R_2 R_1 R_0			8	A ← A + r	No Carry
		r, A		0 0 1 0			8	r ← r + A	No Carry
	SUB	A, r		1 1 1 0			8	A ← A − r	
		r, A		0 1 1 0			8	r ← r − A	
	SBB	A, r		1 1 1 1			8	A ← A − r − CY	
		r, A		0 1 1 1			8	r ← r − A − CY	
	SUBNB	A, r		1 0 1 1			8	A ← A − r	No Borrow
		r, A		0 0 1 1			8	r ← r − A	No Borrow
	ANA	A, r		1 0 0 0 1 R_2 R_1 R_0			8	A ← A ∧ r	
		r, A		0 0 0 0			8	r ← r ∧ A	
	ORA	A, r		1 0 0 1			8	A ← A ∨ r	
		r, A		0 0 0 1			8	r ← r ∨ A	
	XRA	A, r		1 0 0 1 0 R_2 R_1 R_0			8	A ← A ∀ r	
		r, A		0 0 0 1			8	r ← r ∀ A	
	GTA	A, r		1 0 1 0 1 R_2 R_1 R_0			8	A − r − 1	No Borrow
		r, A		0 0 1 0			8	r − A − 1	No Borrow
	LTA	A, r		1 0 1 1			8	A − r	Borrow
		r, A		0 0 1 1			8	r − A	Borrow
	NEA	A, r		1 1 1 0			8	A − r	No Zero
		r, A		0 1 1 0			8	r − A	No Zero

Note Instruction Group

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
8-bit operation instructions (register)	EQA	A, r	0 1 1 0 0 0 0 0	1 1 1 1 1 R_2 R_1 R_0			8	A – r	Zero
		r, A		0 1 1 1			8	r – A	Zero
	ONA	A, r		1 1 0 0			8	A ∧ r	No Zero
	OFFA	A, r		1 1 0 1			8	A ∧ r	Zero
8-bit operation instructions (memory)	ADDX	rpa	0 1 1 1 0 0 0 0	1 1 0 0 0 A_2 A_1 A_0			11	A ← A + (rpa)	
	ADCX	rpa		1 1 0 1			11	A ← A + (rpa) + CY	
	ADDNCX	rpa		1 0 1 0			11	A ← A + (rpa)	No Carry
	SUBX	rpa		1 1 1 0			11	A ← A – (rpa)	
	SBBX	rpa		1 1 1 1			11	A ← A – (rpa) – CY	
	SUBNBX	rpa		1 0 1 1			11	A ← A – (rpa)	No Borrow
	ANAX	rpa		1 0 0 0 1 A_2 A_1 A_0			11	A ← A ∧ (rpa)	
	ORAX	rpa		1 0 0 1			11	A ← A ∨ (rpa)	
	XRAX	rpa		1 0 0 1 0 A_2 A_1 A_0			11	A ← A ⊻ (rpa)	
	GTAX	rpa		1 0 1 0 1 A_2 A_1 A_0			11	A – (rpa) – 1	No Borrow
	LTAX	rpa		1 0 1 1			11	A – (rpa)	Borrow
	NEAX	rpa		1 1 1 0			11	A – (rpa)	No Zero
	EQAX	rpa		1 1 1 1			11	A – (rpa)	Zero
	ONAX	rpa		1 1 0 0			11	A ∧ (rpa)	No Zero
	OFFAX	rpa		1 1 0 1			11	A ∧ (rpa)	Zero

Note Instruction Group

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
Immediate data operation instructions	ADI *	A, byte	0 1 0 0 0 1 1 0	← Data →			7	A ← A + byte	
		r, byte	0 1 1 1 0 1 0 0	0 1 0 0 0 R_2 R_1 R_0	Data		11	r ← r + byte	
		sr2, byte	0 1 1 0 ↓	S_3 1 0 0 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 + byte	
	ACI *	A, byte	0 1 0 1 0 1 1 0	← Data →			7	A ← A + byte + CY	
		r, byte	0 1 1 1 0 1 0 0	0 1 0 1 0 R_2 R_1 R_0	Data		11	r ← r + byte + CY	
		sr2, byte	0 1 1 0 ↓	S_3 1 0 1 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 + byte + CY	
	ADINC *	A, byte	0 0 1 0 0 1 1 0	← Data →			7	A ← A + byte	No Carry
		r, byte	0 1 1 1 0 1 0 0	0 0 1 0 0 R_2 R_1 R_0	Data		11	r ← r + byte	No Carry
		sr2, byte	0 1 1 0 ↓	S_3 0 1 0 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 + byte	No Carry
	SUI *	A, byte	0 1 1 0 0 1 1 0	← Data →			7	A ← A – byte	
		r, byte	0 1 1 1 0 1 0 0	0 1 1 0 0 R_2 R_1 R_0	Data		11	r ← r – byte	
		sr2, byte	0 1 1 0 ↓	S_3 1 1 0 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 – byte	
	SBI *	A, byte	0 1 1 1 0 1 1 0	← Data →			7	A ← A – byte – CY	
		r, byte	0 1 1 1 0 1 0 0	0 1 1 1 0 R_2 R_1 R_0	Data		11	r ← r – byte – CY	
		sr2, byte	0 1 1 0 ↓	S_3 1 1 1 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 – byte – CY	
	SUINB *	A, byte	0 0 1 1 0 1 1 0	← Data →			7	A ← A – byte	No Borrow
		r, byte	0 1 1 1 0 1 0 0	0 0 1 1 0 R_2 R_1 R_0	Data		11	r ← r – byte	No Borrow
		sr2, byte	0 1 1 0 ↓	S_3 0 1 1 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 – byte	No Borrow
	ANI *	A, byte	0 0 0 0 0 1 1 1	← Data →			7	A ← A ∧ byte	
		r, byte	0 1 1 1 0 1 0 0	0 0 0 0 1 R_2 R_1 R_0	Data		11	r ← r ∧ byte	

Note Instruction Group

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
Immediate data operation instructions	ANI	sr2, byte	0 1 1 0 0 1 0 0	S_3 0 0 0 1 S_2 S_1 S_0	Data		20	sr2 ← sr2 ∧ byte	
	ORI *	A, byte	0 0 0 1 0 1 1 1	←Data→			7	A ← A ∨ byte	
		r, byte	0 1 1 1 0 1 0 0	0 0 0 1 1 R_2 R_1 R_0	Data		11	r ← r ∨ byte	
		sr2, byte	0 1 1 0 ↓	S_3 0 0 1 1 S_2 S_1 S_0	↓		20	sr2 ← sr2 ∨ byte	
	XRI *	A, byte	0 0 0 1 0 1 1 0	←Data→			7	A ← A ⊻ byte	
		r, byte	0 1 1 1 0 1 0 0	0 0 0 1 0 R_2 R_1 R_0	Data		11	r ← r ⊻ byte	
		sr2, byte	0 1 1 0 ↓	S_3 0 0 1 0 S_2 S_1 S_0	↓		20	sr2 ← sr2 ⊻ byte	
	GTI *	A, byte	0 0 1 0 0 1 1 1	←Data→			7	A – byte– 1	No Borrow
		r, byte	0 1 1 1 0 1 0 0	0 0 1 0 1 R_2 R_1 R_0	Data		11	r – byte – 1	No Borrow
		sr2, byte	0 1 1 0 ↓	S_3 0 1 0 1 S_2 S_1 S_0	↓		14	sr2 – byte – 1	No Borrow
	LTI *	A, byte	0 0 1 1 0 1 1 1	←Data→			7	A – byte	Borrow
		r, byte	0 1 1 1 0 1 0 0	0 0 1 1 1 R_2 R_1 R_0	Data		11	r – byte	Borrow
		sr2, byte	0 1 1 0 ↓	S_3 0 1 1 1 S_2 S_1 S_0	↓		14	sr2 – byte	Borrow
	NEI *	A, byte	0 1 1 0 0 1 1 1	←Data→			7	A – byte	No Zero
		r, byte	0 1 1 1 0 1 0 0	0 1 1 0 1 R_2 R_1 R_0	Data		11	r – byte	No Zero
		sr2, byte	0 1 1 0 ↓	S_3 1 1 0 1 S_2 S_1 S_0	↓		14	sr2 – byte	No Zero
	EQI *	A, byte	0 1 1 1 0 1 1 1	←Data→			7	A – byte	Zero
		r, byte	0 1 1 1 0 1 0 0	0 1 1 1 1 R_2 R_1 R_0	Data		11	r – byte	Zero
		sr2, byte	0 1 1 0 ↓	S_3 1 1 1 1 S_2 S_1 S_0	↓		14	sr2 – byte	Zero

Note Instruction Group

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
Immediate data operation instructions	ONI *	A, byte	0 1 0 0 0 1 1 1	←Data→			7	A ∧ byte	No Zero
		r, byte	0 1 1 1 0 1 0 0	0 1 0 0 1 R_2 R_1 R_0	Data		11	r ∧ byte	No Zero
		sr2, byte	0 1 1 0 ↓	S_3 1 0 0 1 S_2 S_1 S_0	↓		14	sr2 ∧ byte	No Zero
	OFFI *	A, byte	0 1 0 1 0 1 1 1	←Data→			7	A ∧ byte	Zero
		r, byte	0 1 1 1 0 1 0 0	0 1 0 1 1 R_2 R_1 R_0	Data		11	r ∧ byte	Zero
		sr2, byte	0 1 1 0 ↓	S_3 1 0 1 1 S_2 S_1 S_0	↓		14	sr2 ∧ byte	Zero
Working register operation instructions	ADDW	wa	0 1 1 1 0 1 0 0	1 1 0 0 0 0 0 0	offset		14	A ← A + (V. wa)	
	ADCW	wa	↓ ↓ ↓	1 1 0 1	↓		14	A ← A + (V. wa) + CY	
	ADDNCW	wa	↓ ↓ ↓	1 0 1 0	↓		14	A ← A + (V. wa)	No Carry
	SUBW	wa	↓ ↓ ↓	1 1 1 0	↓		14	A ← A – (V. wa)	
	SBBW	wa	↓ ↓ ↓	1 1 1 1	↓		14	A ← A – (V. wa) – CY	
	SUBNBW	wa	↓ ↓ ↓	1 0 1 1	↓		14	A ← A – (V. wa)	No Borrow
	ANAW	wa	↓ ↓ ↓	1 0 0 0 1 0 0 0	↓		14	A ← A ∧ (V. wa)	
	ORAW	wa	↓ ↓ ↓	1 0 0 1	↓		14	A ← A ∨ (V. wa)	
	XRAW	wa	↓ ↓ ↓	1 0 0 1 0 0 0 0	↓		14	A ← A ∨̄ (V. wa)	
	GTAW	wa	↓ ↓ ↓	1 0 1 0 1 0 0 0	↓		14	A – (V. wa) – 1	No Borrow
	LTAW	wa	↓ ↓ ↓	1 0 1 1	↓		14	A – (V. wa)	Borrow
	NEAW	wa	↓ ↓ ↓	1 1 1 0	↓		14	A – (V. wa)	No Zero
	EQAW	wa	↓ ↓ ↓	1 1 1 1	↓		14	A – (V. wa)	Zero
	ONAW	wa	↓ ↓ ↓	1 1 0 0	↓		14	A ∧ (V. wa)	No Zero

Note Instruction Group

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
Working register operation instructions	OFFAW	wa	0 1 1 1 0 1 0 0	1 1 0 1 1 0 0 0	Offset		14	A ∧ (V. wa)	Zero
	ANIW *	wa, byte	0 0 0 0 0 1 0 1	← Offset →	Data		19	(V. wa) ← (V. wa) ∧ byte	
	ORIW *	wa, byte	0 0 0 1				19	(V. wa) ← (V. wa) ∨ byte	
	GTIW *	wa, byte	0 0 1 0				13	(V. wa) – byte – 1	No Borrow
	LTIW *	wa, byte	0 0 1 1				13	(V. wa) – byte	Borrow
	NEIW *	wa, byte	0 1 1 0				13	(V. wa) – byte	No Zero
	EQIW *	wa, byte	0 1 1 1				13	(V. wa) – byte	Zero
	ONIW *	wa, byte	0 1 0 0				13	(V. wa) ∧ byte	No Zero
	OFFIW	wa, byte	0 1 0 1				13	(V. wa) ∧ byte	Zero
16-bit operation instructions	EADD	EA, r2	0 1 1 1 0 0 0 0	0 1 0 0 0 0 R_1 R_0			11	EA ← EA + r2	
	DADD	EA, rp3	0 1 0 0	1 1 0 0 0 1 P_1 P_0			11	EA ← EA + rp3	
	DADC	EA, rp3		1 1 0 1			11	EA ← EA + rp3 +CY	
	DADDNC	EA, rp3		1 0 1 0			11	EA ← EA + rp3	No Carry
	ESUB	EA, r2	0 0 0 0	0 1 1 0 0 0 R_1 R_0			11	EA ← EA – r2	
	DSUB	EA, rp3	0 1 0 0	1 1 1 0 0 1 P_1 P_0			11	EA ← EA – rp3	
	DSBB	EA, rp3		1 1 1 1			11	EA ← EA – rp3 – CY	
	DSUBNB	EA, rp3		1 0 1 1			11	EA ← EA – rp3	No Borrow
	DAN	EA, rp3		1 0 0 0 1 1 P_1 P_0			11	EA ← EA ∧ rp3	
	DOR	EA, rp3		1 0 0 1			11	EA ← EA ∨ rp3	
	DXR	EA, rp3		1 0 0 1 0 1 P_1 P_0			11	EA ← EA ∨̄ rp3	

Note Instruction Group

Note 1	Mnemonic	Operand	Instruction Code B1	B2	B3	B4	State	Operation	Skip Condition
16-bit operation instructions	DGT	EA, rp3	0 1 1 1 0 1 0 0	1 0 1 0 1 1 P_1 P_0			11	EA – rp3 – 1	No Borrow
	DLT	EA, rp3		1 0 1 1			11	EA – rp3	Borrow
	DNE	EA, rp3		1 1 1 0			11	EA – rp3	No Zero
	DEQ	EA, rp3		1 1 1 1			11	EA – rp3	Zero
	DON	EA, rp3		1 1 0 0			11	EA ∧ rp3	No Zero
	DOFF	EA, rp3		1 1 0 1			11	EA ∧ rp3	Zero
Note 2	MUL	r2	0 1 0 0 1 0 0 0	0 0 1 0 1 1 R_1 R_0			32	EA ← A × r2	
	DIV	r2		0 0 1 1			59	EA ← EA ÷ r2, r2 ← Remainder	
Addition/subtraction instructions	INR	r2	0 1 0 0 0 0 R_1 R_0				4	r2 ← r2 + 1	Carry
	INRW *	wa	0 0 1 0 0 0 0 0	←Offset→			16	(V. wa) ← (V. wa) + 1	Carry
	INX	rp	0 0 P_1 P_0 0 0 1 0				7	rp ← rp + 1	
		EA	1 0 1 0 1 0 0 0				7	EA ← EA + 1	
	DCR	r2	0 1 0 1 0 0 R_1 R_0				4	r2 ← r2 – 1	Borrow
	DCRW *	wa	0 0 1 1 0 0 0 0	←Offset→			16	(V. wa) ← (V. wa) – 1	Borrow
	DCX	rp	0 0 P_1 P_0 0 0 1 1				7	rp ← rp – 1	
		EA	1 0 1 0 1 0 0 1				7	EA ← EA – 1	
Note 3	DAA		0 1 1 0 0 0 0 1				4	Decimal Adjust Accumulator	
	STC		0 1 0 0 1 0 0 0	0 0 1 0 1 0 1 1			8	CY ← 1	
	CLC			0 0 1 0 1 0 1 0			8	CY ← 0	
	NEGA			0 0 1 1 1 0 1 0			8	A ← $\overline{A}$ + 1	

Note
1. Instruction Group
2. Multiply/divide instructions
3. Other operation instructions

Note	Mnemonic	Operand	Instruction Code B1	Instruction Code B2	Instruction Code B3	Instruction Code B4	State	Operation	Skip Condition
Rotation/shift instructions	RLD		0 1 0 0 1 0 0 0	0 0 1 1 1 0 0 0			17	Rotate Left Digit	
	RRD			1 0 0 1			17	Rotate Right Digit	
	RLL	r2		0 1 R_1 R_0			8	$r2_{m+1} \leftarrow r2_m$, $r2_0 \leftarrow CY$, $CY \leftarrow r2_7$	
	RLR	r2		0 0 R_1 R_0			8	$r2_{m-1} \leftarrow r2_m$, $r2_7 \leftarrow CY$, $CY \leftarrow r2_0$	
	SLL	r2		0 0 1 0 0 1 R_1 R_0			8	$r2_{m+1} \leftarrow r2_m$, $r2_0 \leftarrow 0$, $CY \leftarrow r2_7$	
	SLR	r2		0 0 R_1 R_0			8	$r2_{m-1} \leftarrow r2_m$, $r2_7 \leftarrow 0$, $CY \leftarrow r2_0$	
	SLLC	r2		0 0 0 0 0 1 R_1 R_0			8	$r2_{m+1} \leftarrow r2_m$, $r2_0 \leftarrow 0$, $CY \leftarrow r2_7$	Carry
	SLRC	r2		0 0 R_1 R_0			8	$r2_{m-1} \leftarrow r2_m$, $r2_7 \leftarrow 0$, $CY \leftarrow r2_0$	Carry
	DRLL	EA		1 0 1 1 0 1 0 0			8	$EA_{n+1} \leftarrow EA_n$, $EA_0 \leftarrow CY$, $CY \leftarrow EA_{15}$	
	DRLR	EA		0 0 0 0			8	$EA_{n-1} \leftarrow EA_n$, $EA_{15} \leftarrow CY$, $CY \leftarrow EA_0$	
	DSLL	EA		1 0 1 0 0 1 0 0			8	$EA_{n+1} \leftarrow EA_n$, $EA_0 \leftarrow 0$, $CY \leftarrow EA_{15}$	
	DSLR	EA		0 0 0 0			8	$EA_{n-1} \leftarrow EA_n$, $EA_{15} \leftarrow 0$, $CY \leftarrow EA_0$	
Jump instructions	JMP *	word	0 1 0 1 0 1 0 0	← Low Adrs →	High Adrs		10	PC ← word	
	JB		0 0 1 0 0 0 0 1				4	$PC_H \leftarrow B$, $PC_L \leftarrow C$	
	JR	word	1 1 ← jdisp 1 →				10	PC ← PC + 1 + jdisp 1	
	JRE *	word	0 1 0 0 1 1 1 ←	jdisp →			10	PC ← PC + 2 + jdisp	
	JEA		0 1 0 0 1 0 0 0	0 0 1 0 1 0 0 0			8	PC ← EA	
Call Instructions	CALL *	word	0 1 0 0 0 0 0 0	← Low Adrs →	High Adrs		16	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$ PC ← word, SP ← SP − 2	
	CALB		0 1 0 0 1 0 0 0	0 0 1 0 1 0 0 1			17	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$ $PC_H \leftarrow B$, $PC_L \leftarrow C$, SP ← SP − 2	
	CALF *	word	0 1 1 1 1 ←	fa →			13	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$ $PC_{15-11} \leftarrow 00001$, $PC_{10-0} \leftarrow fa$, SP ← SP − 2	

Note Instruction Group

Note 1	Mnemonic	Operand	Instruction Code B1	B2	B3	B4	State	Operation	Skip Condition
Note 2	CALT	word	1 0 0 ← ta →				16	(SP – 1) ← (PC + 1)H, (SP – 2) ← (PC + 1)L PCL ← (128+2ta), PCH ← (129+2ta), SP ← SP–2	
	SOFTI		0 1 1 1 0 0 1 0				16	(SP – 1) ← PSW, (SP – 2) ← (PC + 1)H, (SP – 3) ← (PC + 1)L, PC ← 0060H, SP ← SP – 3	
Return instructions	RET		1 0 1 1 1 0 0 0				10	PCL ← (SP), PCH ← (SP + 1) SP ← SP + 2	
	RETS		1 0 0 1				10	PCL ← (SP), PCH ← (SP + 1), SP ← SP + 2 PC ← PC + n	Unconditional skip
	RETI		0 1 1 0 0 0 1 0				13	PCL ← (SP), PCH ← (SP + 1) PSW ← (SP + 2), SP ← SP + 3	
Skip instructions	BIT *	bit, wa	0 1 0 1 1 B2 B1 B0	← Offset →			10	Skip if (V. wa) bit = 1	(V. wa)bit = 1
	SK	f	0 1 0 0 1 0 0 0	0 0 0 0 1 F2 F1 F0			8	Skip if f = 1	f = 1
	SKN	f		0 0 0 1			8	Skip if f = 0	f = 0
	SKIT	irf		0 1 0 I4 I3 I2 I1 I0			8	Skip if irf = 1, then reset irf	irf = 1
	SKNIT	irf		0 1 1 I4 I3 I2 I1 I0			8	Skip if irf = 0 Reset irf, if irf = 1	irf = 0
CPU control instructions	NOP		0 0 0 0 0 0 0 0				4	No Operation	
	EI		1 0 1 0 1 0 1 0				4	Enable Interrupt	
	DI		1 0 1 1 1 0 1 0				4	Disable Interrupt	
	HLT		0 1 0 0 1 0 0 0	0 0 1 1 1 0 1 1			12	Set Halt Mode	
	STOP		0 1 0 0 1 0 0 0	1 0 1 1 1 0 1 1			12	Set Stop Mode	

* **1.** Data is B2 if rpa2 = D + byte, H + byte.

2. Data is B3 if rpa3 = D + byte, H + byte.

3. In the State item, a figure is in the right side of slash if rpa2 and rpa3 are D + byte, H + A, H + B, H + EA, H + byte.

Remarks The idle state when each instruction is skipped is different from the execution state as shown below.

1-byte instruction	: 4 states	3-byte instruction (with *)	: 10 states
2-byte instruction (with *)	: 7 states	3-byte instruction	: 11 states
2-byte instruction	: 8 states	4-byte instruction	: 14 states

Notes **1.** Instruction Group

2. Call instructions

7. LIST OF MODE REGISTERS

Name of Mode Registers		Read/ Write	Function
MA	MODE A register	W	Specifies bit-wise the input/output of the port A.
MB	MODE B register	W	Specifies bit-wise the input/output of the port B.
MCC	MODE CONTROL C register	W	Specifies bit-wise the port/control mode of the port C.
MC	MODE C register	W	Specifies bit-wise the input/output of the port C which is in port mode.
MM	MEMORY MAPPING register	W	Specifies the port/expansion mode of port D and port F.
MF	MODE F register	W	Specifies bit-wise the input/output of the port F which is in port mode.
TMM	Timer mode register	R/W	Specifies operating mode of timer.
ETMM	Timer/event counter mode register	W	Specifies the operating mode of timer/event counter.
EOM	Timer/event counter output mode register	R/W	Control the output level of CO0 and CO1.
SML	Serial mode register	W	Specifies the operating mode of serial interface.
SMH		R/W	
MKL	interrupt mask register	R/W	Specifies the enable/disable of the interrupt request.
MKH			
ANM	A/D channel mode register	R/W	Specifies the operating mode of A/D converter.
ZCM	Zero-cross mode register	W	Specifies the operation of zero-cross detector circuit.

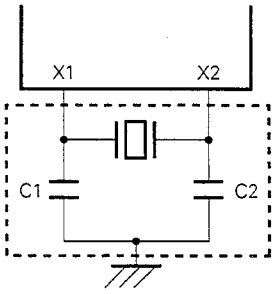
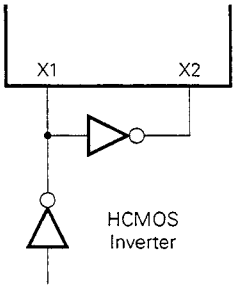
8. ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (T_A = 25 °C)

PARAMETER	SYMBOL	TEST CONDITIONS	RATINGS	UNIT
Power supply voltage	V_{DD}		−0.5 to +7.0	V
	AV_{DD}		AV_{SS} to V_{DD} + 0.5	V
	AV_{SS}		−0.5 to +0.5	V
Input voltage	V_I		−0.5 to V_{DD} + 0.5	V
Output voltage	V_O		−0.5 to V_{DD} + 0.5	V
Output current, low	I_{OL}	Per pin	4.0	mA
		Total of all output pins	100	mA
Output current, high	I_{OH}	Per pin	−2.0	mA
		Total of all output pins	−50	mA
A/D converter reference input voltage	V_{AREF}		−0.5 to AV_{DD} + 0.3	V
Operating ambient temperature	T_A		−40 to +85	°C ★
Storage temperature	T_{stg}		−65 to +150	°C

Caution If the absolute maximum rating of even one of the above parameters is exceeded even momentarily, the quality of the product may be degraded. The absolute maximum ratings, therefore, specify the values exceeding which the product may be physically damaged. Be sure to use the product with these rated values never exceeded. ★

OSCILLATOR CHARACTERISTICS (T_A = −40 to +85 °C, V_{DD} = AV_{DD} = +5.0 V ±10 %, V_{SS} = AV_{SS} = 0 V, V_{DD} − 0.8 V ≤ AV_{DD} ≤ V_{DD}, 3.4 V ≤ V_{AREF} ≤ AV_{DD})

RESONATOR	RECOMMENDED CIRCUIT	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
Ceramic or crystal resonator		Oscillation frequency (fxx)	A/D converter not used	4	15	MHz
			A/D converter used	5.8	15	MHz
External clock		X1 input frequency (fx)	A/D converter not used	4	15	MHz
			A/D converter used	5.8	15	MHz
		X1 rise time, fall time (t_r, t_f)		0	20	ns
		X1 input high, low level width ($t_{\phi H}$, $t_{\phi L}$)		20	250	ns

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

2. Ensure that no other signal lines are routed through the area enclosed with dotted lines.

CAPACITANCE (T_A = 25 °C, V_{DD} = V_{SS} = 0 V)

PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNIT
Input capacitance	C_I	f_C = 1 MHz Unmeasured pins returned to 0 V			10	pF
Output capacitance	C_O				20	pF
Input-output capacitance	C_{IO}				20	pF

DC CHARACTERISTICS (T_A = −40 to +85 °C, V_{DD} = AV_{DD} = +5.0 V ±10 %, V_{SS} = AV_{SS} = 0 V)

PARAMETER	SYMBOL	TEST CONDITIONS		MIN.	TYP.	MAX.	UNIT
Input voltage, low	V_{IL1}	All except $\overline{RESET}$, $\overline{STOP}$, $\overline{NMI}$, $\overline{SCK}$, INT1, TI, AN7 to AN4		0		0.8	V
	V_{IL2}	$\overline{RESET}$, $\overline{STOP}$, $\overline{NMI}$, $\overline{SCK}$, INT1, TI, AN7 to AN4		0		0.2V_{DD}	V
Input voltage, high	V_{IH1}	All except $\overline{RESET}$, $\overline{STOP}$, $\overline{NMI}$, $\overline{SCK}$, INT1, TI, AN7 to AN4, X1, X2		2.2		V_{DD}	V
	V_{IH2}	$\overline{RESET}$, $\overline{STOP}$, $\overline{NMI}$, $\overline{SCK}$, INT1, TI, AN7 to AN4, X1, X2		0.8V_{DD}		V_{DD}	V
Output voltage, low	V_{OL}	I_{OL} = 2.0 mA				0.45	V
Output voltage, high	V_{OH}	I_{OH} = −1.0 mA		V_{DD} − 1.0			V
		I_{OH} = −100 μA		V_{DD} − 0.5			V
Input current	I_{LI}	INT1[Note1], TI (PC3)[Note2]; 0 V ≤ V_I ≤ V_{DD}				±200	μA
Input leakage current	I_{LI}	All except INT1, TI (PC3), AN7 to AN0; 0 V ≤ V_I ≤ V_{DD}				±10	μA
		AN7 to AN0; 0 V ≤ V_I ≤ V_{DD}				±1	μA
Output leakage current	I_{LO}	0 V ≤ V_O ≤ V_{DD}				±10	μA
AV_{DD} power supply current	AI_{DD1}	Operating mode f_{XX} = 15 MHz			0.5	1.3	mA
	AI_{DD2}	STOP mode			10	20	μA
V_{DD} power supply current	I_{DD1}	Operating mode f_{XX} = 15 MHz			16	30	mA
	I_{DD2}	HALT mode f_{XX} = 15 MHz			7	13	mA
Data retention voltage	V_{DDDR}	Hardware/software STOP mode		2.5			V
Data retention current	I_{DDDR}	Hardware/software[Note3]	V_{DDDR} = 2.5 V		1	15	μA
		STOP mode	V_{DDDR} = 5 V ±10 %		10	50	μA
Pull-up resistor	R_L	Ports A, B, and C	3.5 V ≤ V_{DD} ≤ 5.5 V, V_I = 0 V	17	27	75	kΩ

Notes 1. If self-bias should be generated by ZCM register.
2. If the control mode is set by MCC register, and self-bias should be generated by ZCM register.
3. If self-bias is not generated.

AC CHARACTERISTICS (T_A = −40 to +85 °C, V_{DD} = AV_{DD} = +5.0 V ±10 %, V_{SS} = AV_{SS} = 0 V)
Read/write Operation:

PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	MAX.	UNIT
X1 input cycle time	t_{CYC}		66	250	ns
Address setup time (to ALE ↓)	t_{AL}	f_{XX} = 15 MHz, CL = 100 pF	30		ns
Address hold time (from ALE ↓)	t_{LA}		35		ns
$\overline{RD}$ ↓ delay time from address	t_{AR}		100		ns
Address float time from $\overline{RD}$ ↓	t_{AFR}	C_L = 100 pF		20	ns
Data input time from address	t_{AD}	f_{XX} = 15 MHz, C_L = 100 pF		250	ns
Data input time from ALE ↓	t_{LDR}			135	ns
Data input time from $\overline{RD}$ ↓	t_{RD}			120	ns
$\overline{RD}$ ↓ delay time from ALE ↓	t_{LR}		15		ns
Data hold time (from $\overline{RD}$ ↑)	t_{RDH}	C_L = 100 pF	0		ns
ALE ↑ delay time from $\overline{RD}$ ↑	t_{RL}	f_{XX} = 15 MHz, C_L = 100 pF	80		ns
$\overline{RD}$ low-level width	t_{RR}	In Data Read f_{XX} = 15 MHz, C_L = 100 pF	215		ns
		In OP Code Fetch f_{XX} = 15 MHz, C_L = 100 pF	415		ns
ALE high-level width	t_{LL}	f_{XX} = 15 MHz, C_L = 100 pF	90		ns
$\overline{M1}$ setup time (to ALE ↓)	t_{ML}	f_{XX} = 15 MHz	30		ns
$\overline{M1}$ hold time (from ALE ↓)	t_{LM}		35		ns
$\overline{IO}$/M setup time (to ALE ↓)	t_{IL}		30		ns
$\overline{IO}$/M hold time (from ALE ↓)	t_{LI}		35		ns
$\overline{WR}$ ↓ delay time from address	t_{AW}	f_{XX} = 15 MHz, C_L = 100 pF	100		ns
Data output time from ALE ↓	t_{LDW}			180	ns
Data output time from $\overline{WR}$ ↓	t_{WD}	C_L = 100 pF		100	ns
$\overline{WR}$ ↓ delay time from ALE ↓	t_{LW}	f_{XX} = 15 MHz, C_L = 100 pF	15		ns
Data setup time (to $\overline{WR}$ ↑)	t_{DW}		165		ns
Data hold time (from $\overline{WR}$ ↑)	t_{WDH}		60		ns
ALE ↑ delay time from $\overline{WR}$ ↑	t_{WL}		80		ns
$\overline{WR}$ low-level width	t_{WW}		215		ns

Serial Operation :

PARAMETER	SYMBOL	TEST CONDITIONS		MIN.	MAX.	UNIT
$\overline{SCK}$ cycle time	t_{CYK}	$\overline{SCK}$ input	**Note1**	800		ns
			Note2	400		ns
		$\overline{SCK}$ output		1.6		μs
$\overline{SCK}$ low-level width	t_{KKL}	$\overline{SCK}$ input	**Note1**	335		ns
			Note2	160		ns
		$\overline{SCK}$ output		700		ns
$\overline{SCK}$ high-level width	t_{KKH}	$\overline{SCK}$ input	**Note1**	335		ns
			Note2	160		ns
		$\overline{SCK}$ output		700		ns
RxD setup time (to $\overline{SCK}$ ↑)	t_{RXK}	**Note1**		80		ns
RxD hold time (from $\overline{SCK}$ ↑)	t_{KRX}	**Note1**		80		ns
TxD delay time from $\overline{SCK}$ ↓	t_{KTX}	**Note1**			210	ns

Notes 1. If clock rate is × 1 in asynchronous mode, synchronous mode, or I/O interface mode.
2. If clock rate is × 16 or × 64 in asynchronous mode.

Remark The numeric values in the table are those when f_{XX} = 15 MHz, C_L = 100 pF.

Zero-Cross Characteristics :

PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	MAX.	UNIT
Zero-cross detection input	V_{ZX}	AC combination 60-Hz sine wave	1	1.8	VAC_{P-P}
Zero-cross accuracy	A_{ZX}			±135	mV
Zero-cross detection input frequency	f_{ZX}		0.05	1	kHz

Other Operation :

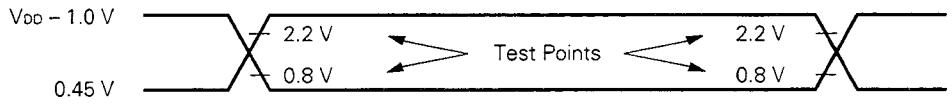
PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	MAX.	UNIT
TI high, low-level width	t_{TIH}, t_{TIL}		6		t_{CYC}
CI high, low-level width	t_{CI1H}, t_{CI1L}	Event count mode	6		t_{CYC}
	t_{CI2H},t_{CI2L}	Pulse width test mode	48		t_{CYC}
$\overline{NMI}$ high, low-level width	t_{NIH}, t_{NIL}		10		μs
INT1 high, low-level width	t_{I1H}, t_{I1L}		36		t_{CYC}
$\overline{INT2}$ high, low-level width	t_{I2H}, t_{I2L}		36		t_{CYC}
AN7 to AN4, low-level width	t_{ANH}, t_{ANL}		36		t_{CYC}
$\overline{RESET}$ high, low-level width	t_{RSH}, t_{RSL}		10		μs

A/D CONVERTER CHARACTERISTICS (T_A = −40 to +85 °C, V_{DD} = +5.0 V ±10 %, V_{SS} = AV_{SS} = 0 V, V_{DD} − 0.5 V ≤ AV_{DD} ≤ V_{DD}, 3.4 V ≤ V_{AREF} ≤ AV_{DD})

PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNIT
Resolution			8			Bits
Absolute accuracy[Note]		3.4 V ≤ V_{AREF} ≤ AV_{DD}, 66 ns ≤ t_{CYC} ≤ 170 ns			±0.8 %	FSR
		4.0 V ≤ V_{AREF} ≤ AV_{DD}, 66 ns ≤ t_{CYC} ≤ 170 ns			±0.6 %	FSR
		T_A = −10 to +70 °C, 4.0 V ≤ V_{AREF} ≤ AV_{DD}, 66 ns ≤ t_{CYC} ≤ 170 ns			±0.4 %	FSR
Conversion time	t_{CONV}	66 ns ≤ t_{CYC} ≤ 110 ns	576			t_{CYC}
		110 ns ≤ t_{CYC} ≤ 170 ns	432			t_{CYC}
Sampling time	t_{SAMP}	66 ns ≤ t_{CYC} ≤ 110 ns	96			t_{CYC}
		110 ns ≤ t_{CYC} ≤ 170 ns	72			t_{CYC}
Analog input voltage	V_{IAN}	AN7 to AN0 (including unused pins)	−0.3		V_{AREF} + 0.3	V
★ Analog input impedance	R_{AN}			50		MΩ
Reference voltage	V_{AREF}		3.4		AV_{DD}	V
V_{AREF} current	I_{AREF1}	Operating mode		1.5	3.0	mA
	I_{AREF2}	STOP mode		0.7	1.5	mA
AV_{DD} power supply current	AI_{DD1}	Operating mode f_{XX} = 15 MHz		0.5	1.3	mA
	AI_{DD2}	STOP mode		10	20	μA

Note Quantization error (±1/2 LSB) is not included.

AC Timing Test Point



t_{CYC}-Dependent AC Characteristics Expression

PARAMETER	EXPRESSION	MIN./MAX.	UNIT
t_{AL}	2T – 100	MIN.	ns
t_{LA}	T – 30	MIN.	ns
t_{AR}	3T – 100	MIN.	ns
t_{AD}	7T – 220	MAX.	ns
t_{LDR}	5T – 200	MAX.	ns
t_{RD}	4T – 150	MAX.	ns
t_{LR}	T – 50	MIN.	ns
t_{RL}	2T – 50	MIN.	ns
t_{RR}	4T – 50 (In data read)	MIN.	ns
	7T – 50 (In OP code fetch)		
t_{LL}	2T – 40	MIN.	ns
t_{ML}	2T – 100	MIN.	ns
t_{LM}	T – 30	MIN.	ns
t_{IL}	2T – 100	MIN.	ns
t_{LI}	T – 30	MIN.	ns
t_{AW}	3T – 100	MIN.	ns
t_{LDW}	T + 110	MAX.	ns
t_{LW}	T – 50	MIN.	ns
t_{DW}	4T – 100	MIN.	ns
t_{WDH}	2T – 70	MIN.	ns
t_{WL}	2T – 50	MIN.	ns
t_{WW}	4T – 50	MIN.	ns
t_{CYK}	6T ($\overline{SCK}$ input)[Note1]/12T ($\overline{SCK}$ input)[Note2]	MIN.	ns
	24T ($\overline{SCK}$ output)		
t_{KKL}	2.5T + 5 ($\overline{SCK}$ input)[Note1]/5T + 5 ($\overline{SCK}$ input)[Note2]	MIN.	ns
	12T – 100 ($\overline{SCK}$ output)		
t_{KKH}	2.5T + 5 ($\overline{SCK}$ input)[Note1]/5T + 5 ($\overline{SCK}$ input)[Note2]	MIN.	ns
	12T – 100 ($\overline{SCK}$ output)		

Notes 1. If clock rate is ×16, ×64 in asynchronous mode.

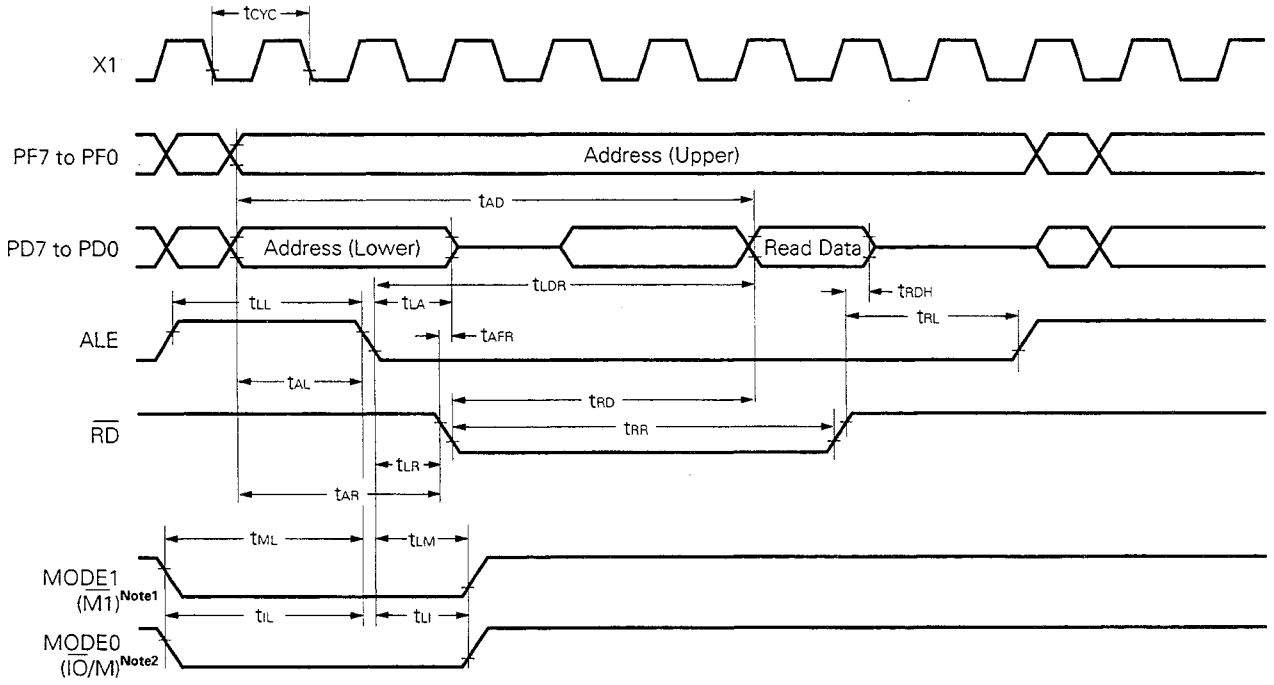
2. If clock rate is ×1, in asynchronous mode, synchronous mode, or I/O interface mode.

Remarks 1. $T = t_{CYC} = 1/f_{XX}$

2. Other items which are not listed in this table are not dependent on oscillator frequency (f_{XX}).

Timing Waveform

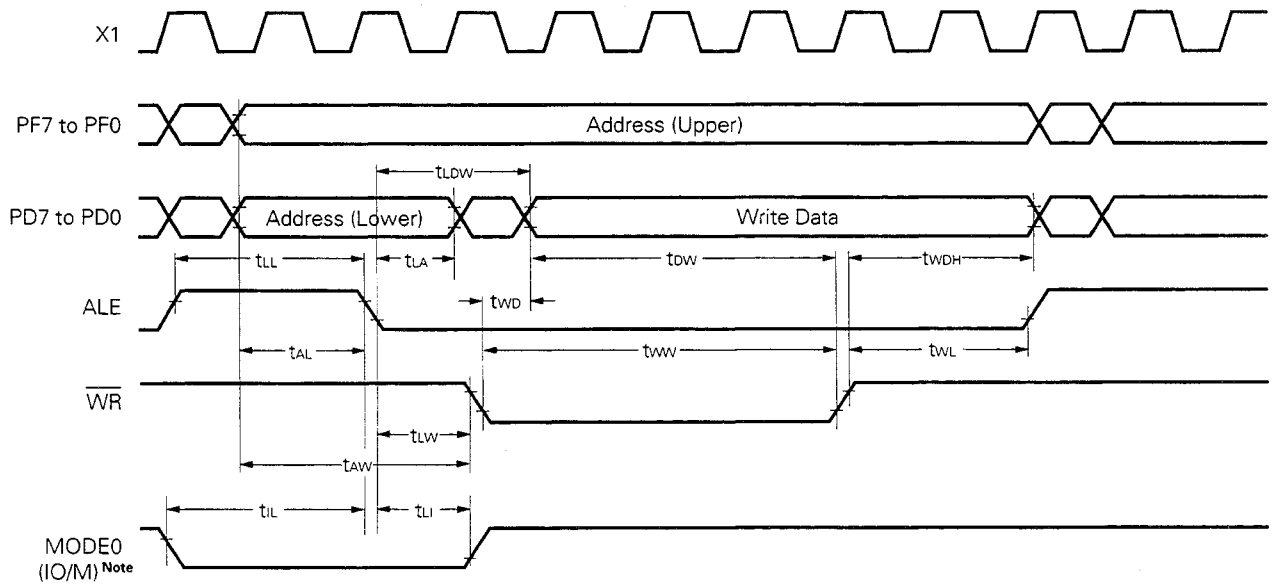
Read Operation



Notes 1. When MODE1 pin is pulled up, $\overline{M1}$ signal is output to MODE1 pin in the 1st OP code fetch cycle.

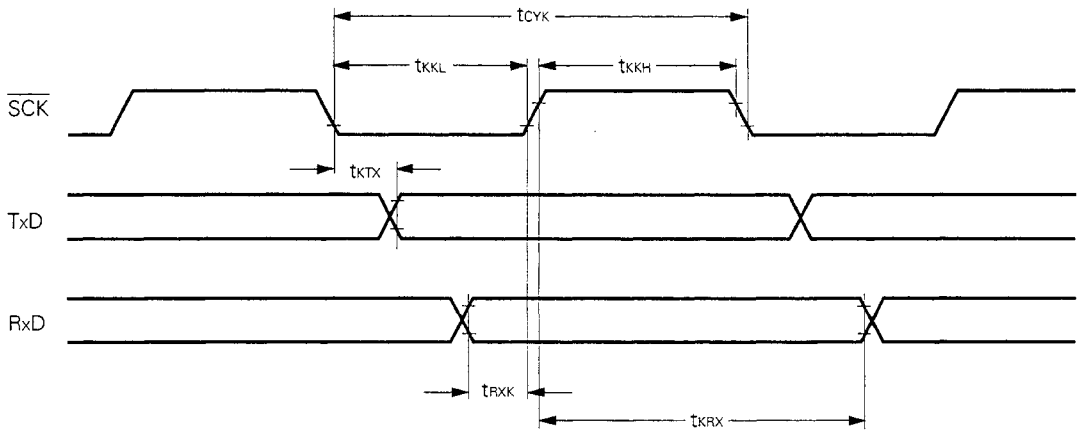
2. When MODE0 pin is pulled up, $\overline{IO}$/M signal is output to MODE0 pin in sr to sr2 register read cycle.

Write Operation

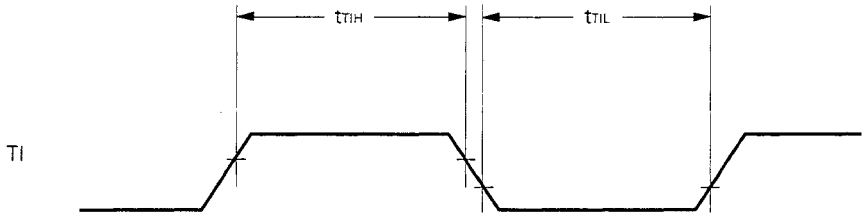


Note When MODE0 pin is pulled up, $\overline{IO}$/M signal is output to MODE0 pin in sr to sr2 register write cycle.

Serial Operation

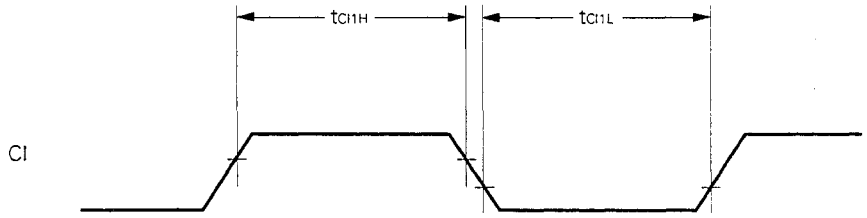


Timer Input Timing

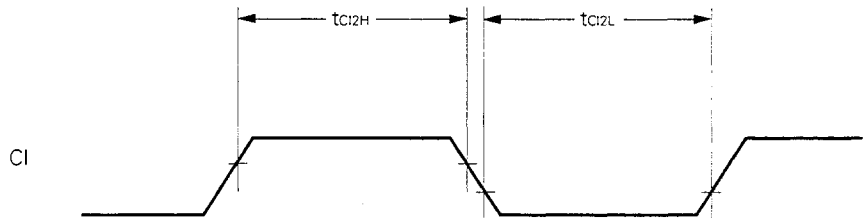


Timer/event Counter Input Timing

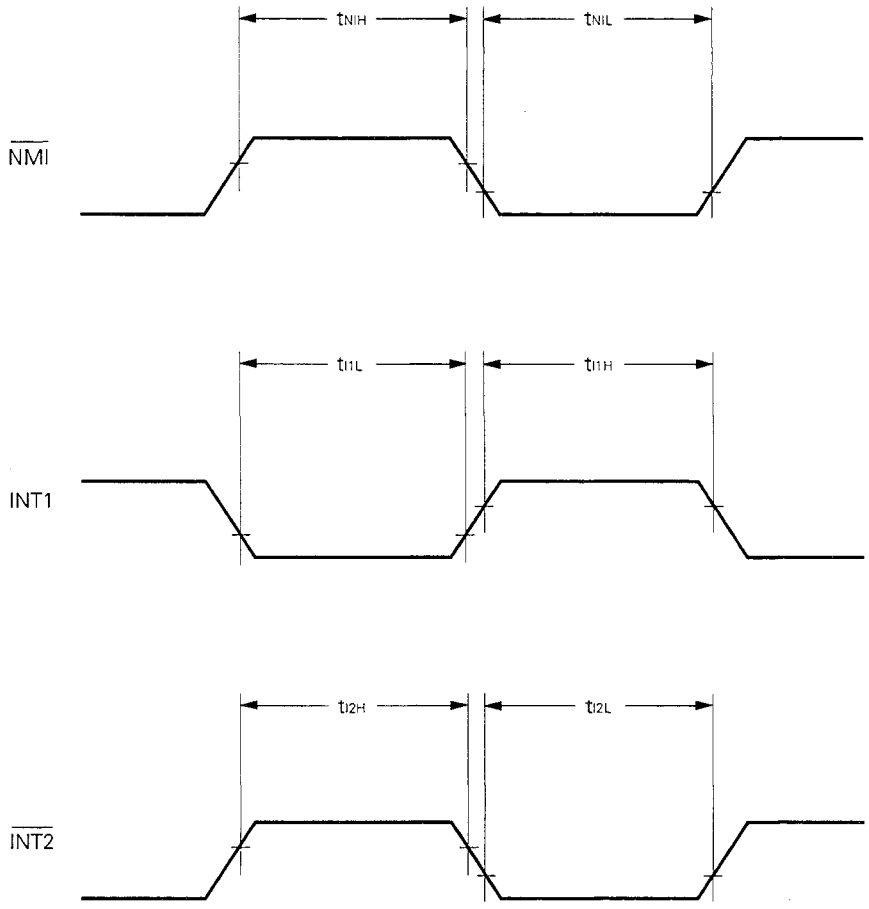
Event Counter Mode



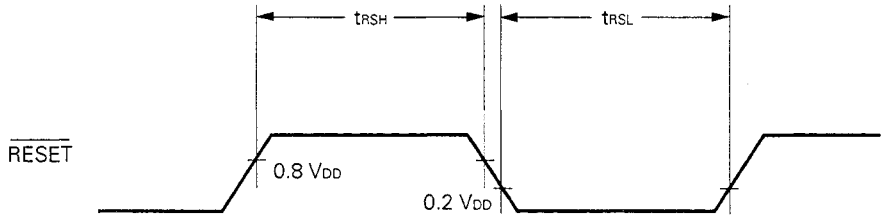
Pulse Width Test Mode



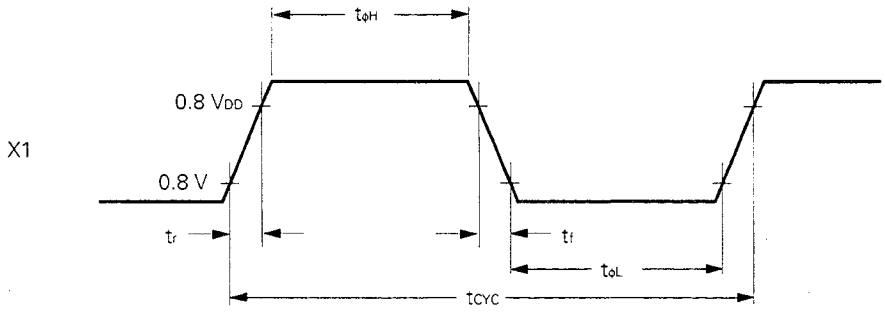
Interrupt Input Timing



Reset Input Timing



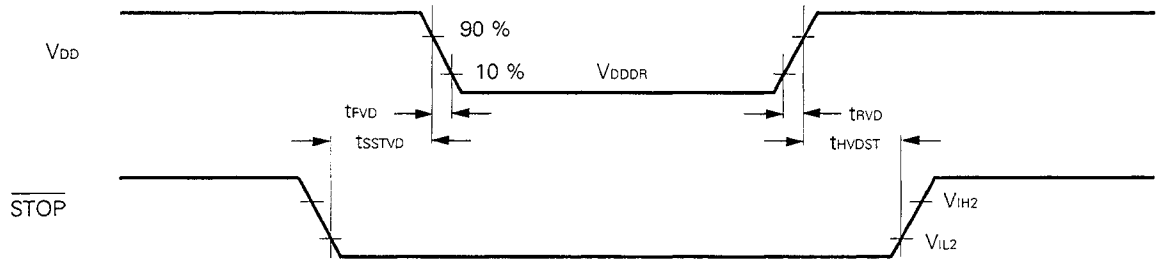
External Clock Timing



DATA MEMORY STOP MODE LOW POWER SUPPLY VOLTAGE DATA RETENTION (T_A = −40 to +85 °C)

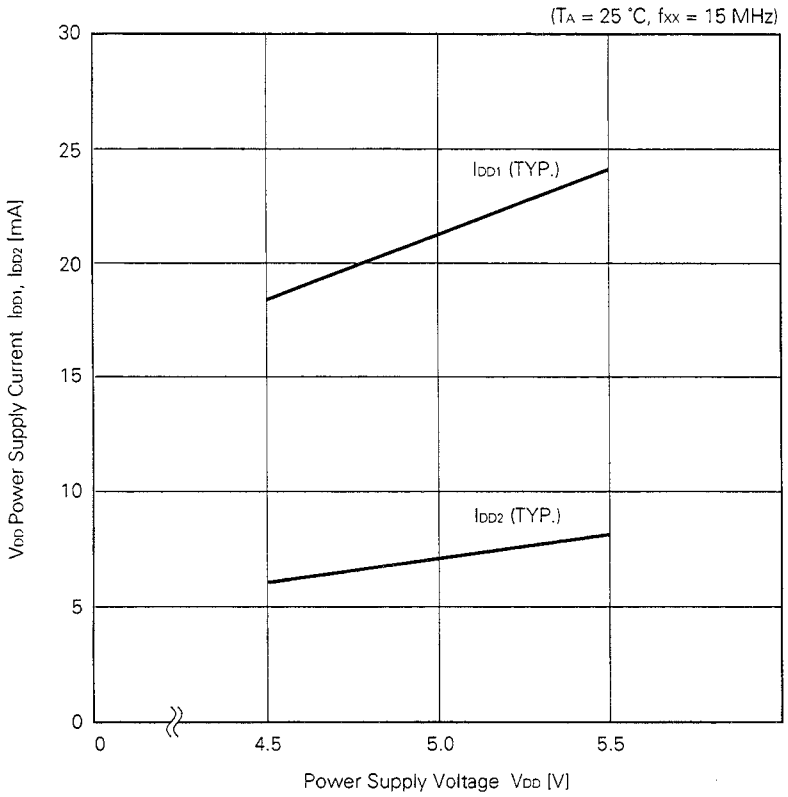
PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNIT
Data retention power supply voltage	V_{DDDR}		2.5		5.5	V
Data retention power supply current	I_{DDDR}	V_{DDDR} = 2.5 V		1	15	μA
		V_{DDDR} = 5 V ±10 %		10	50	μA ★
V_{DD} rise/fall time	t_{RVD}, t_{FVD}		200			μs
$\overline{STOP}$ setup time (to V_{DD})	t_{SSTVD}		12T + 0.5			μs
$\overline{STOP}$ hold time (from V_{DD})	t_{HVDST}		12T + 0.5			μs

Data Retention Timing

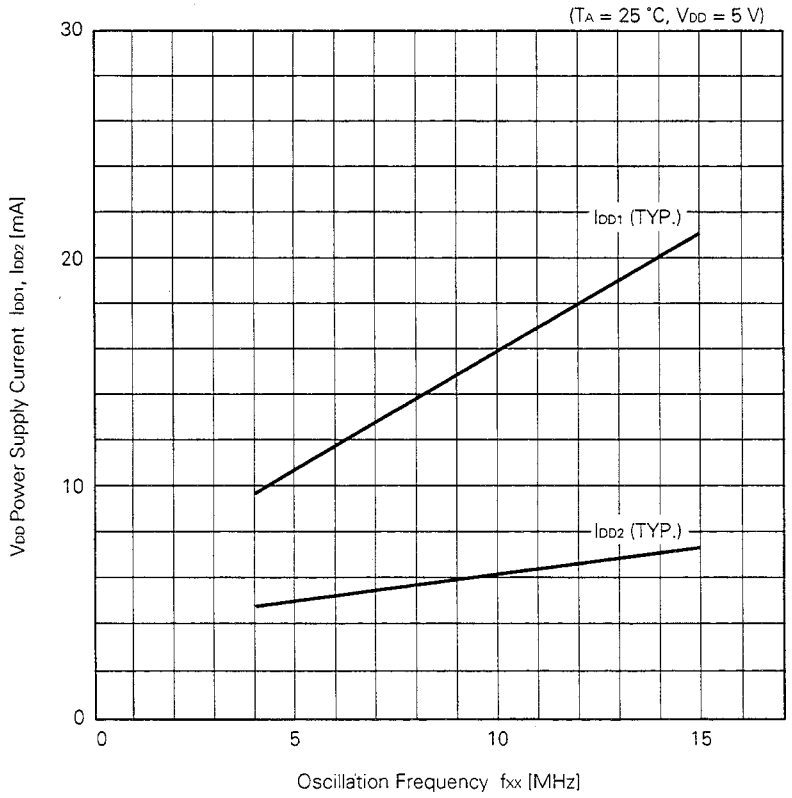


9. CHARACTERISTIC CURVES

I_{DD1}, I_{DD2} vs V_{DD}

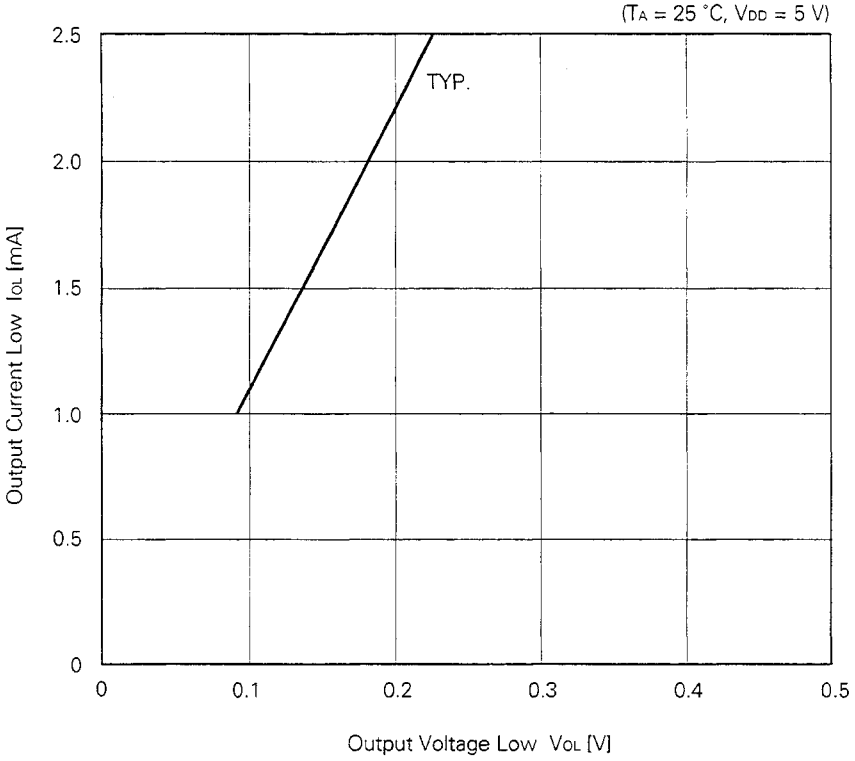


I_{DD1}, I_{DD2} vs fxx



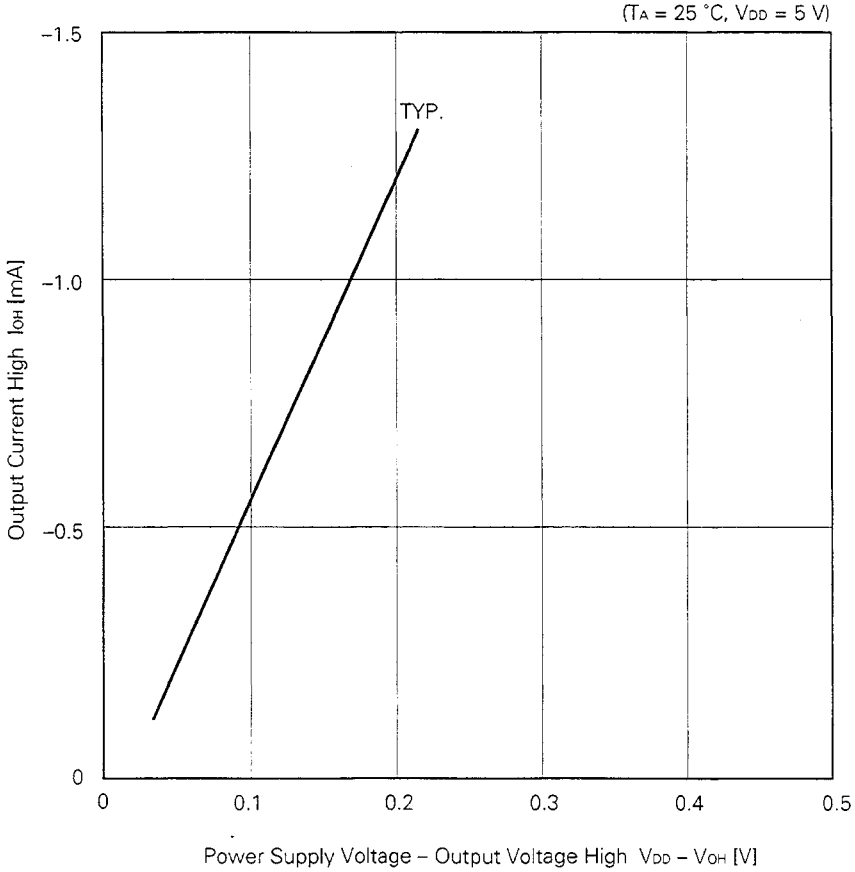
I_{OL} vs V_{OL}

(T_A = 25 °C, V_{DD} = 5 V)

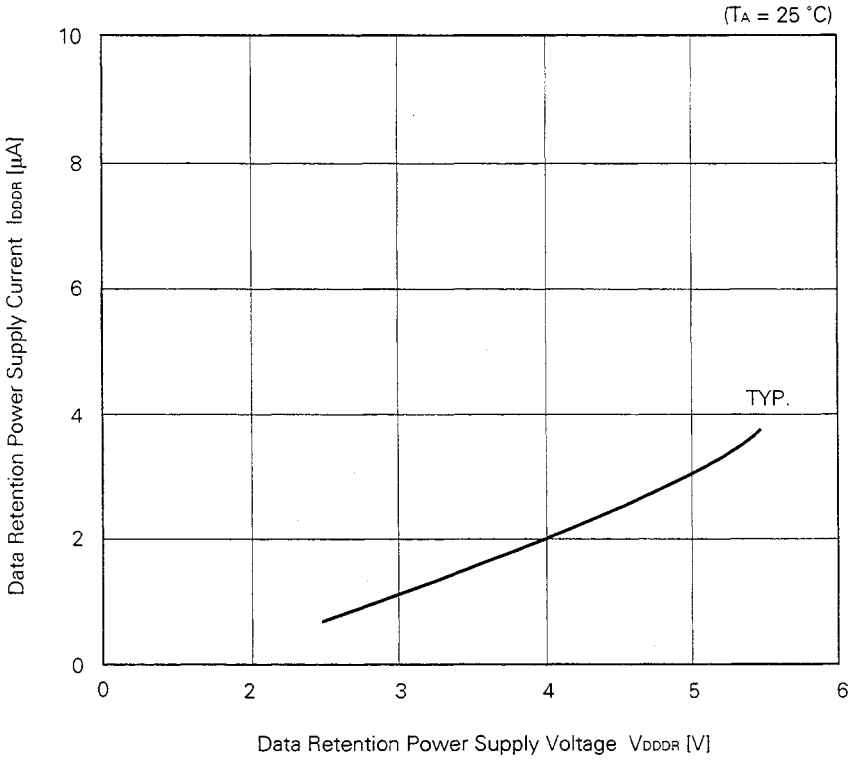


I_{OH} vs V_{OH}

(T_A = 25 °C, V_{DD} = 5 V)

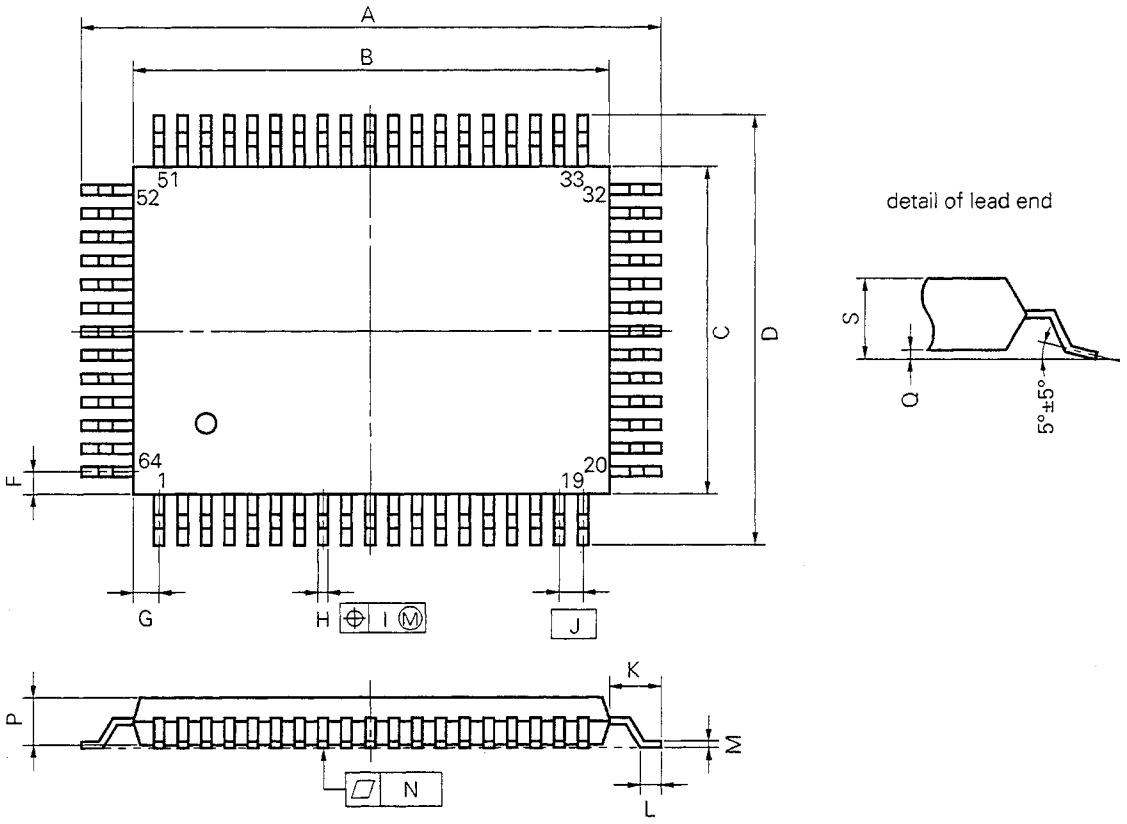


I_{DDDR} vs V_{DDDR}



10. PACKAGE DRAWINGS

64 PIN PLASTIC QFP (14×20)



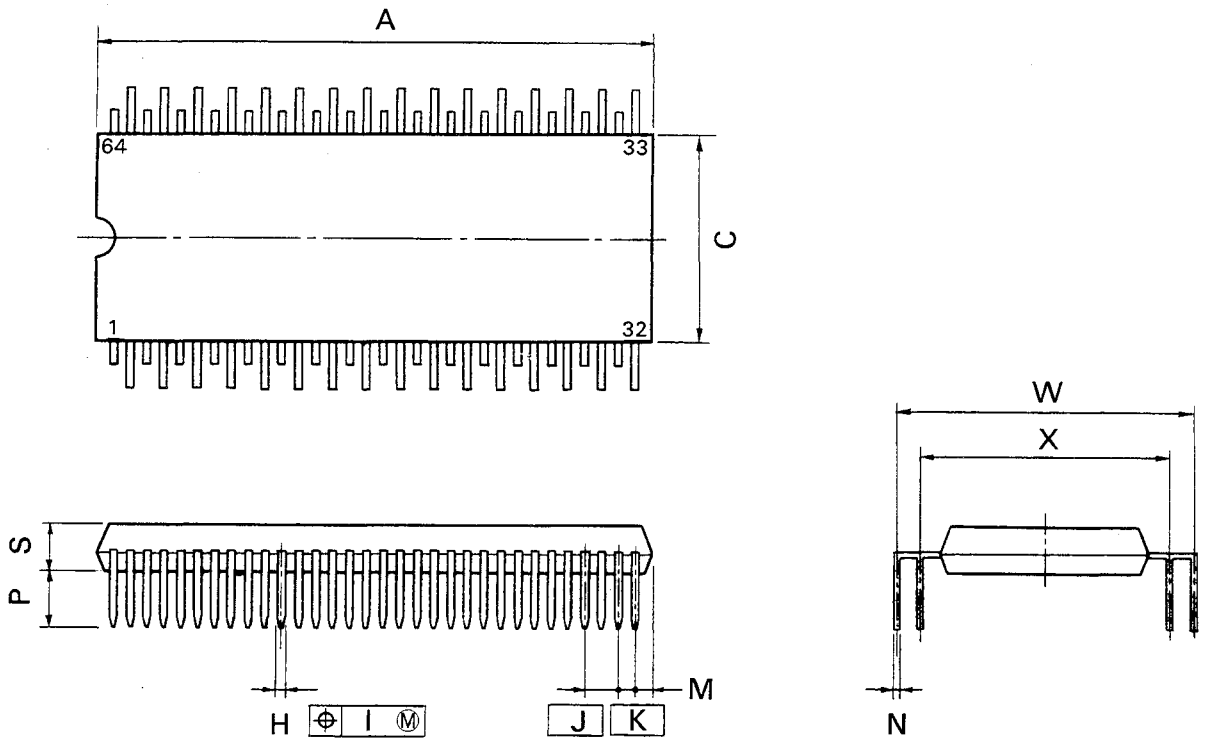
P64GF-100-3B8,3BE,3BR-1

NOTE

Each lead centerline is located within 0.20 mm (0.008 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	23.6±0.4	0.929±0.016
B	20.0±0.2	$0.795^{+0.009}_{-0.008}$
C	14.0±0.2	$0.551^{+0.009}_{-0.008}$
D	17.6±0.4	0.693±0.016
F	1.0	0.039
G	1.0	0.039
H	0.40±0.10	$0.016^{+0.004}_{-0.005}$
I	0.20	0.008
J	1.0 (T.P.)	0.039 (T.P.)
K	1.8±0.2	$0.071^{+0.008}_{-0.009}$
L	0.8±0.2	$0.031^{+0.009}_{-0.008}$
M	$0.15^{+0.10}_{-0.05}$	$0.006^{+0.004}_{-0.003}$
N	0.12	0.005
P	2.7	0.106
Q	0.1±0.1	0.004±0.004
S	3.0 MAX.	0.119 MAX.

64 PIN PLASTIC QUI



P64GQ-100-36

NOTE

Each lead centerline is located within 0.25 mm (0.010 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	$41.5^{+0.3}_{-0.2}$	$1.634^{+0.012}_{-0.008}$
C	16.5	0.650
H	$0.50^{\pm0.10}$	$0.020^{+0.004}_{-0.005}$
I	0.25	0.010
J	2.54 (T.P.)	0.100 (T.P.)
K	1.27 (T.P.)	0.050 (T.P.)
M	$1.1^{+0.25}_{-0.15}$	$0.043^{+0.011}_{-0.006}$
N	$0.25^{+0.10}_{-0.05}$	$0.010^{+0.004}_{-0.003}$
P	$4.0^{\pm0.3}$	$0.157^{+0.013}_{-0.012}$
S	$3.6^{\pm0.1}$	$0.142^{+0.004}_{-0.005}$
W	$24.13^{\pm1.05}$	$0.950^{\pm0.042}$
X	$19.05^{\pm1.05}$	$0.750^{\pm0.042}$

11. RECOMMENDED SOLDERING CONDITIONS ★

This μPD78C17(A)/78C18(A) should be soldered and mounted under the following recommended conditions.

For details of the conditions, refer to the document **"Surface Device Mounting Technology Manual"** (IEI-1207).

For soldering methods and conditions other than those recommended below, contact an NEC sales representative.

Table 11-1 Surface Mounting Type Soldering Conditions

μPD78C17GF(A)-3BE: 64-pin plastic QFP (14 × 20 mm)
μPD78C18GF(A)-xxx-3BE: 64-pin plastic QFP (14 × 20 mm)

Soldering Method	Soldering Conditions	Recommended Condition Symbol
Infrared ray reflow	Package peak temperature: 235 °C, Time: Within 30 s (at 210 °C or higher), Count: Twice or less <Attention> (1) Perform the second reflow when the device temperature has come down to the room temperature from the heating by the first reflow. (2) Do not wash the soldered portion with flux following the first reflow.	IR35-00-2
VSP	Package peak temperature: 215 °C, Time: Within 40 s (at 200 °C or higher), Count: Twice or less <Attention> (1) Perform the second reflow when the device temperature has come down to the room temperature from the heating by the first reflow. (2) Do not wash the soldered portion with flux following the first reflow.	VP15-00-2
Wave soldering	Solder bath temperature: 260 °C or lower, Time: Within 10 s, Count: Once, Preheating temperature: 120 °C MAX. (package surface temperature)	WS60-00-1
Partial heating	Pin temperature: 300 °C or lower, Time: Within 3 s (per pin row)	—

Caution Do not use several soldering methods in combination (except partial heating).

Table 11-2 Through-Hole Type Soldering Condition

μPD78C17GQ(A)-36: 64-pin plastic QUIP
μPD78C18GQ(A)-xxx-36: 64-pin plastic QUIP

Soldering Method	Soldering Conditions
Wave soldering (pin only)	Solder bath temperature: 260 °C or lower, Time: Within 10 s
Partial heating	Pin temperature: 300 °C, Time: Within 3 s (per pin)

Caution Wave soldering must be applied to pins only, and care must be taken to prevent solder from coming into direct with the package body.

12. DIFFERENCES BETWEEN μPD78C18(A) AND μPD78C14(A)/78C12A(A)

Item \ Part Number	**μPD78C18(A)**	μPD78C14(A)	μPD78C12A(A)
Internal ROM	32 K × 8	16 K × 8	8 K × 8
Internal RAM	1 K × 8	256 × 8	
Port A to Port C	On-chip pull-up resistor selectable bit-wise by mask option	No on-chip pull-up resistor	On-chip pull-up resistor selectable bit-wise by mask option
Package	• 64-pin plastic QFP (14 × 20 mm) • 64-pin plastic QUIP	• 64-pin plastic QFP (14 × 20 mm) • 64-pin plastic QUIP • 68-pin plastic QFJ	

APPENDIX DEVELOPMENT TOOLS ★

The following development tools are available to develop a system which uses the μPD78C17(A) or 78C18(A).

Language Processor

87AD series relocatable assembler (RA87)	This is a program which converts a program written in mnemonic to an object code for which microcomputer execution is possible. Besides, it contains a function to automatically create a symbol/table, and optimize a branch instruction.			
	Host Machine	OS	Supply Medium	Ordering Code (Product Name)
	PC-9800 series	MS-DOS™ Ver. 2.11 to Ver. 5.00A[Note]	3.5-inch 2HD	μS5A13RA87
			5-inch 2HD	μS5A10RA87
	IBM PC/AT™	PC DOS™ (Ver. 3.1)	3.5-inch 2HC	μS7B13RA87
			5-inch 2HC	μS7B10RA87

PROM Write Tools

Hardware	PG-1500	With a provided board and an optional programmer adapter connected, this PROM programmer can manipulate from a stand-alone or host machine to perform programming on single-chip microcomputer which incorporates PROM. It is also capable of programming a typical PROM ranging from 256 K to 4 Mbits.		
	PA-78CP14GF/GQ	PROM programmer adapter for the μPD78CP18(A). Used by connecting to PG-1500.		
	PA-78CP14GF	For the μPD78CP18GF(A)-3BE		
	PA-78CP14GQ	For the μPD78CP18GQ(A)-36		
Software	PG-1500 controller	Connected PG-1500 to a host machine by using serial and parallel interface, to control the PG-1500 on a host machine.		
		Host Machine	OS	Supply Medium	Ordering Code (Product Name)
		PC-9800 series	MS-DOS Ver. 2.11 to Ver. 5.00A[Note]	3.5-inch 2HD	μS5A13PG1500
				5-inch 2HD	μS5A10PG1500
		IBM PC/AT	PC DOS (Ver. 3.1)	5-inch 2HC	μS7B10PG1500

Note Ver. 5.00/5.00A are provided with the task swap function, but it cannot be used with this software.

Remark Operation of assemblers and the PG-1500 controller are guaranteed only on the host machines and operating systems quoted above.

Debugging tools

An in-circuit emulator (IE-78C11-M) is available as a program debugging tool for the μPD78C17(A) and 78C18(A). The following table shows its system configuration.

Hardware	IE-78C11-M	The IE-78C11-M is an in-circuit emulator which works with 87AD series. It can be connected to a host machine to perform efficient debugging.			
Software	IE-78C11-M control program (IE controller)	Connects the IE-78C11-M to host machine by using the RS-233C, then controls the IE-78C11-M on host machine.			
		Host Machine	OS	Supply Medium	Ordering Code (Product Name)
		PC-9800 series	MS-DOS [Ver. 2.11 to Ver. 3.30D]	3.5-inch 2HD	μS5A13IE78C11
				5-inch 2HD	μS5A10IE78C11
		IBM PC/AT	PC DOS (Ver. 3.1)	5-inch 2HC	μS7B10IE78C11

Remark Operation of the IE controller is guaranteed only on the host machine and operating systems quoted above.

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

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

The export of these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

License not needed: μPD78C17GF(A)-3BE, 78C17GQ(A)-36

The customer must judge
the need for license: μPD78C18GF(A)-xxx-3BE, 78C18GQ(A)-xxx-36

M4 94.11