USER'S MANUAL

NEC

μPD75336

4-BIT SINGLE-CHIP MICROCOMPUTER

μPD75336
μPD75P338

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

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

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

The devices listed in this document are not suitable for use in aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. If customers intend to use NEC devices for above applications or they intend to use "Standard" quality grade NEC devices for applications not intended by NEC, please contact our sales people in advance.

Application examples recommended by NEC Corporation

Standard: Computer, Office equipment, Communication equipment, Test and Measurement equipment, Machine tools, Industrial robots, Audio and Visual equipment, Other consumer products, etc.

Special: Automotive and Transportation equipment, Traffic control systems, Antidisaster systems, Anticrime systems, etc.

M7 92.6

6427525 0094845 516

B

Major Revisions in This Edition

Location	Description
Whole manual	uPD75P336GK Under development → Development completed
1-6	Amendment of Table 1-1 "Differences between uPD75336 and uPD75P336"
2-13	2.2.10 INT0, INT1 Addition of text covering STOP mode and HALT mode release
2-14	2.2.11 INT2 Addition of text covering STOP mode and HALT mode release
2-18	Amendment of 2.2.26 IC (uPD75336 only) item
5-7	Amendment of Figure 5-3 "Configurations of Ports 3n and 6n"
5-7	Amendment of Figure 5-4 "Configurations of Ports 2 and 7
5-8	Addition of cautions to Table 5-5 "Maximum Time Required for System Clock and CPU Clock Switching"
6-6	Amendment of text in 6.3 (1) "Interrupt request flags & interrupt enable flags"
8-1	Amendment of Table 8-1 "Hardware Status after Reset"
B-1	Appendix B "Development Tools" Version upgrade of relevant MS-DOS (Ver. 5.00/5.00A)

PREFACE

Intended Readership : This manual is intended for user engineers who understand the uPD75336 and uPD75P336 functions and wish to design an application system with those functions.

Purpose : This manual has been prepared to enable the users to have an understanding of the uPD75336 and uPD75P336 hardware functions.

How to Read this Manual: Readers of this manual are required to have a general knowledge of electric and logic circuits and microcomputers.

- Users who have previously used the uPD75328

  → The basic functions are the same. The uPD75336 can be used as the 16K version of the uPD75328. The uPD75336 has an improved CPU (instructions and registers) and timer. Refer to Appendix A. "Functional comparison of uPD75336, uPD75P336 and uPD75328" for details of additional functions and instructions.

- Users who use this manual for the uPD75P336

  → Unless there are functional differences, this manual describes the uPD75336 as a representative type. After checking the functional differences by referring to Section 1.3 "Differences between uPD75336 and uPD75P336", use this manual by replacing "uPD75336" with "uPD75P336".

. When checking instruction functions with knowledge of mnemonics

→ Refer to Appendix D.2 "Instruction Index (in Alphabetical Order)".

. When checking instructions with understanding of functions but not mnemonics

→ Check the mnemonics of functions by referring to Section 10.2 "Instruction Set and Operations" and then check the functions by referring to Section 10.4 "Instruction Functions and Use".

. When having an understanding of uPD75336 and uPD75P336

→ After checking the main functions by referring to Section 1.1 "Function Outline", read this manual in the order of the contents.

. For the electrical specifications of the uPD75336/75P336

→ See the separate Data Sheet.

Legend	:	Data representations:	Most significant digit on the left and least significant digit on the right
		Active-low representations	: $\overline{xxx}$ (line above pin and signal names)
		Memory map address	: Upper - low, lower - high
		*	: Description of * in the text
		NOTE	: Contents to be read with particular attention
		Remarks	: Supplementary description of text
		Numeric representations	: Binary number ... xxxx or xxxxB
			Decimal number ... xxxx
			Hexadecimal number .. xxxxH

Relevant Documents

Device Documentation

Product / Document	uPD75336	uPD75P336
Data Sheet	IC-7972	IC-8371
User's Manual	This manual	
Instruction Operation Table	IEM-5516	
75X Series Selection Guide	IF-151	

Development Tool Documentation

	Document Name		Document No.
Hardware	IE-75000-R/IE-75001-R User's Manual		EEU-846
	IE-75000-R-EM User's Manual		EEU-673
	EP-75336GC-R User's Manual		EEU-691
	EP-75336GK-R User's Manual		EEU-837
	PG-1500 User's Manual		EEU-651
Software	RA75X Assembler Package User's Manual	Operation Volume	EEU-731
		Language Volume	EEU-730
	PG-1500 Controller User's Manual		EEU-704

Other Documentation

Document Name	Document No.
Package Manual	IEI-635
Semiconductor Device Mounting Manual	IEI-616
NEC Semiconductor Device Quality Standards	IEI-620
NEC Semiconductor Device Reliability and Quality Control	IEM-5068
Electrostatic Discharge (ESD) Testing	MEM-539
Semiconductor Device Quality Assurance Guide	MEI-603
Microcomputer Related Product Guide - Other Manufacturers Volume	MEI-604

NOTE: The contents of the above documents are subject to change without notice. Please ensure that the latest versions are used for design work, etc.

CONTENTS

Contents of Figures

Contents of Tables

CHAPTER 1. OVERVIEW

The uPD75336 and uPD75P336 are 4-bit single-chip microcomputer 75X series products capable of data processing equal to an 8-bit microcomputer.

The uPD75336 features expanded ROM and RAM capacity compared with the uPD75328, low-voltage operation of the A/D converter and enhanced 8-bit data processing.

The uPD75P336 is a product in which the on-chip mask ROM of uPD75336 is replaced with a one-time programmable PROM. It is most suitable for preproduction and short-run production for system development.

The uPD75336 and uPD75P336 have the following features:

- uPD75328 upward compatible
- Instruction execution time variable function which is useful for high-speed operations and power saving.
- On-chip A/D converter with 8-bit resolution (successive approximation): 8 channels
- On-chip LCD controller/driver
- Improved timer functions: 4 channels
- Improved 8-bit data processing
- Ultra compact package (80-pin plastic TQFP (fine pitch) (□ 12 mm))
- The uPD75P336 can operate over the same power supply voltage range as the uPD75336 mask product : V_{DD} = 2.7 to 6.0 V

Since the uPD75336 has LCD display and A/D conversion functions and can operate at high-speeds with low power consumption, it can be applied in the following fields:

- Cameras
- VCR integrated cameras
- Air conditioners
- Sphygmomanometers

Through the adoption of a compact package, the uPD75336 can be used to control ultra-small devices.

Remarks: This manual describes the uPD75336 as a representative type.

Readers using this manual for the uPD75P336 should refer to "How to Read" described in "PREFACE".

1.1 FUNCTION OUTLINE

Item		Function			
Instruction execution time		When main-system clock is selected: 0.95, 1.91, 3.81, 15.3 us (when operated at 4.19 MHz) When subsystem clock is selected: 122 us (when operated at 32.768 kHz)			
On-chip memory	ROM	16256 x 8 bits			
	RAM	768 x 4 bits			
General register		. 4-bit manipulation: 8 x 4 banks . 8-bit manipulation: 4 x 4 banks			
I/O lines [Includes the LCD drive dual-function pin but not includes the LCD drive dedicated pin]		44 lines	8 lines	CMOS input pin	On-chip pull-up resistor specifiable by software (except P00)
			20 lines	CMOS input/output pin	
			8 lines	CMOS output pin	Dual-function segment pin
			8 lines	N-ch open-drain input/output	10 V withstand voltage. The pull-up resistor specifiable by using the mask option.
LCD controller/driver		. LCD drive output pin . Segment output pin: 20 (CMOS output dual-function pin: 8) . Common output pin : 4 . Number of segment selections: 12/16/20 segments . Display mode selection . Static . 1/2 duty, 1/2 bias . 1/3 duty, 1/2 bias . 1/3 duty, 1/3 bias . 1/4 duty, 1/3 bias . LCD drive voltage supply split resistor may be integrated (mask option).			

(to be continued)

(cont'd)

Item	Function
A/D converter	On-chip 8-bit resolution A/D converter (successive approximation) . 8-channel analog input . Low-voltage operation capability: V_{DD} = 2.7 to 6.0 V . A/D conversion speed: 40.1 us (when operated at 4.19 MHz)
Timer	4 channels { . 8-bit timer/event counter x 2 channels . 8-bit basic interval timer . Watch timer: 0.5 sec time interval generation and buzzer output capability (2 kHz, 4 kHz, 32 kHz)
Serial interface	. NEC standard serial bus interface (SBI) . Clock synchronous serial interface
Bit sequential buffer	Special bit manipulation memory: 16 bits
Clock output (PCL)	Φ, 524 kHz, 262 kHz, 65.5 kHz (when operated at 4.19 MHz)
Buzzer output (BUZ)	2 kHz, 4 kHz, 32 kHz (when main-system or subsystem clock operates)
Vectored interrupt	. External: 3 . Internal: 4
Test input	. External: 1 . Internal: 1
8-bit data processing	Transfer, add/subtract, increment/decrement and compare
System clock oscillator	. Ceramic/crystal oscillator for main-system clock oscillation: 4.194304 MHz . Crystal oscillator for subsystem clock oscillation: 32.768 kHz
Standby	STOP/HALT mode
Operating voltage	V_{DD} = 2.7 to 6.0 V
Package	80-pin plastic QFP (□ 14 mm) 80-pin plastic TQFP (fine pitch) (□ 12 mm)

1.2 ORDERING INFORMATION AND QUALITY GRADE

(1) Ordering Information

Ordering Code	Package	Program Memory
uPD75336GC-xxx-3B9	80-pin plastic QFP (□14 mm)	Mask ROM
uPD75336GK-xxx-BE9	80-pin plastic TQFP (fine pitch) (□12 mm)	Mask ROM
uPD75P336GC-3B9	80-pin plastic QFP (□14 mm)	One-time PROM
uPD75P336GK-BE9	80-pin plastic TQFP (fine pitch) (□12 mm)	One-time PROM

Remarks: xxx is the ROM code number.

(2) Quality Grade

Ordering Code	Package	Quality Grade
uPD75336GC-xxx-3B9	80-pin plastic QFP (□14 mm)	Standard
uPD75336GK-xxx-BE9	80-pin plastic TQFP (fine pitch) (□12 mm)	Standard
uPD75P336GC-3B9	80-pin plastic QFP (□14 mm)	Standard
uPD75P336GK-BE9	80-pin plastic TQFP (fine pitch) (□12 mm)	Standard

Remarks: xxx is the ROM code number.

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

1.3 DIFFERENCES BETWEEN uPD75336 AND uPD75P336

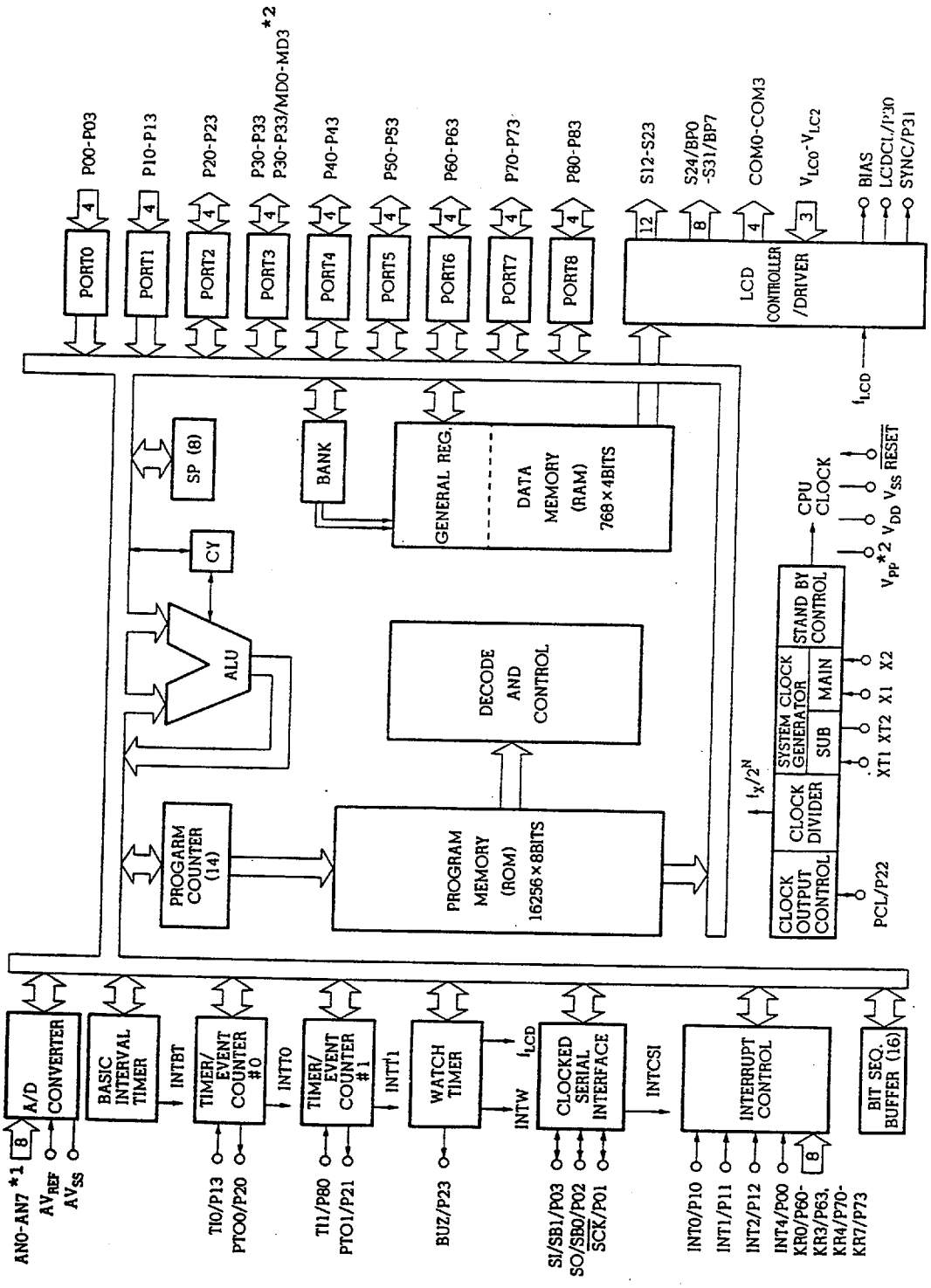
In the uPD75P336, the program memory of the on-chip mask ROM uPD75336 is replaced with one-time PROM that can be written to only once. The differences between the uPD75336 and the uPD75P336 are shown below. Please take careful note of these differences when moving from application system debugging and preproduction using one-time PROM, to volume production using the mask ROM product.

Table 1-1 Differences between uPD75336 and uPD75P336

Item		uPD75336	uPD75P336
Program memory		. Mask ROM . 16256 x 8 bits . 0000H to 3F7FH	. One-time PROM . 16256 x 8 bits . 0000H to 3F7FH
Data memory		768 x 4 bits	768 x 4 bits
Pull-up resister of ports 4 and 5		Mask option	None
Split resistor for LCD drive power supply			
Feedback resister for subsystem clock oscillation		Mask option	On chip
Pin connection	Nos. 50 to 53	P30 to P33	P30/MD0 to P33/MD3
	No. 69	IC	V_{PP}
Electrical specifications		Different consumption current, etc. See the Electrical Specifications section in the respective Data Sheets.	
Operating voltage range		V_{DD} = 2.7 to 6.0 V	
Packages		. 80-pin plastic QFP (□ 14 mm) . 80-pin plastic TQFP (fine pitch) (□ 12 mm)	
Others		As the circuit scale, mask layout, etc., are different, noise resistance, noise emission, etc., are different.	

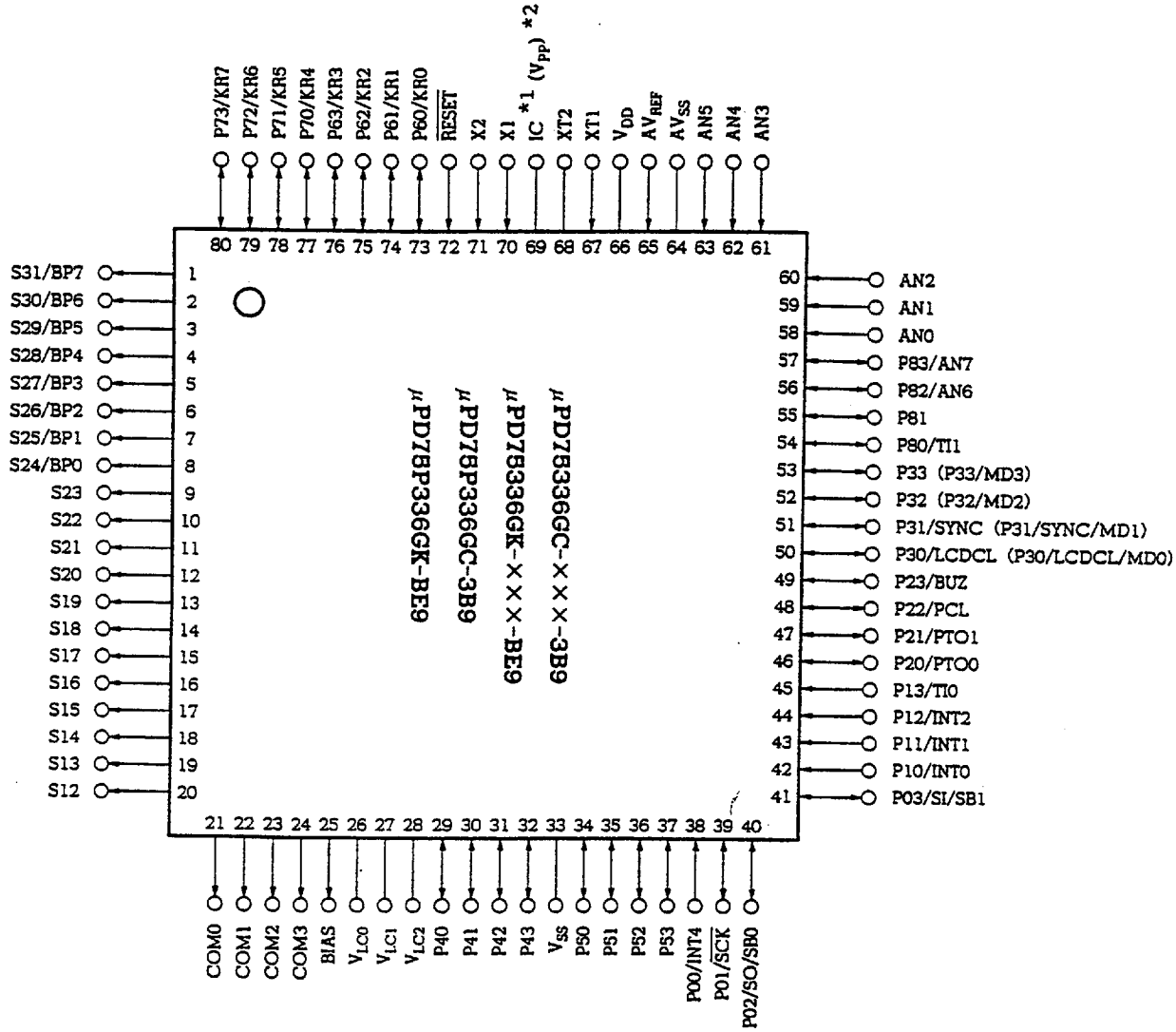
NOTE: Noise resistance and noise emission are different for PROM and mask ROM. When investigating the replacement of the PROM product with the mask ROM product in the transition from preproduction to volume production, thorough evaluation must be carried out with the mask ROM CS product (not the ES product).

1.4 BLOCK DIAGRAM



* 1: AN6/P82, AN7/P83
2: When uPD75P336 is used.

1.5 PIN CONFIGURATION



*1: IC pin: Internally Connected. Connect directly to V_{DD}.

2: In normal operation, connect V_{PP} directly to V_{DD}.

Pin Name

P00 to 03 : Port0
P10 to 13 : Port1
P20 to 23 : Port2
P30 to 33 : Port3
P40 to 43 : Port4
P50 to 53 : Port5
P60 to 63 : Port6
P70 to 73 : Port7
P80 to 83 : Port8
BP0 to 7 : Bit Port 0 to 7
KR0 to 7 : Key Return 0 to 7
AV_{REF} : Analog Reference
AV_{SS} : Analog Ground
AN0 to 7 : Analog Input 0 to 7
$\overline{SCK}$: Serial Clock
SI : Serial Input
SO : Serial Output
(MD0 to 3 : Mode Seloction)
(V_{PP} : Programming/Verifying Power Supply)

SB0, 1 : Serial Bus 0, 1
$\overline{RESET}$: Reset Input
S12 to 31 : Segment Output 12 to 31
COM0 to 3 : Common Output 0 to 3
$V_{LC0 \text{ to } 2}$: LCD Power Supply 0 to 2
BIAS : LCD Power Supply Bias Control
LCDCL : LCD Clock
SYNC : LCD Synchronization
TI0, 1 : Timer Input 0, 1
PTO0, 1 : Programmable Timer Output 0, 1
BUZ : Buzzer Clock
PCL : Programmable Clock
INT0, 1, 4 : External Vectored Interrupt 0, 1, 4
INT2 : External Test Input 2
X1, 2 : Main System Clock Oscillation 1, 2
XT1, 2 : Subsystem Clock Oscillation 1, 2
IC : Internally Connected
V_{DD} : Positive Power Supply
V_{SS} : Ground

Remarks: uPD75P336 in parentheses

CHAPTER 2. PIN FUNCTIONS

2.1 LIST OF uPD75336 PIN FUNCTIONS

Table 2-1 List of Digital Input/Output Port Pin Functions

Pin Name	Input/ Output	Dual-Function Pin	Function	8-Bit I/O	When Reset	Input/ Output Circuit Type *1
P00	Input	INT4	4-bit input port (PORT0). On-chip pull-up resistor specifiable in 3-bit units by software for P01 to P03.	x	Input	Ⓑ
P01	Input/ output	$\overline{SCK}$				Ⓕ - A
P02	Input/ output	SO/SB0				Ⓕ - B
P03	Input/ output	SI/SB1				Ⓜ - C
P10	Input	INT0	Noise elimination function available	x	Input	Ⓑ - C
P11		INT1	4-bit input port (PORT1). On-chip pull-up resistor specifiable in 4-bit units by software for this port.			
P12		INT2				
P13		TI0				
P20	Input/ output	PTO0	4-bit input/output port (PORT2). On-chip pull-up resistor specifiable in 4-bit units by software for this port.	x	Input	E - B
P21		PTO1				
P22		PCL				
P23		BUZ				
P30 *2	Input/ output	LCDCL *4 (MD0)	Programmable 4-bit input/output port (PORT3). Input/output specifiable bit by bit. On-chip pull-up resistor specifiable in 4-bit units by software for this port.	x	Input	E - B
P31 *2		SYNC *4 (MD1)				
P32 *2		(MD2) *4				
P33 *2		(MD3) *4				

(to be continued)

Table 2-1 List of Digital Input/Output Port Pin Functions (cont'd)

Pin Name	Input/ Output	Dual-Function Pin	Function	8-Bit I/O	When Reset	Input/ Output Circuit Type *1
P40 to P43 *2	Input/ output	—	N-ch open-drain 4-bit input/output port (PORT4). On-chip pull-up resistor specifiable bit-wise (mask option). *3 10 V resistance with open-drain	o	High level (with an on-chip pull-up resistor) or high impedance	M (M - A) *4
P50 to P53 *2	Input/ output	—	N-ch open-drain 4-bit input/output port (PORT5). On-chip pull-up resistor specifiable bit-wise (mask option). *3 10 V resistance with open-drain.	o	High level (with an on-chip pull-up resistor) or high impedance	M (M - A) *4
P60	Input/ output	KR0	Programmable 4-bit input/output port (PORT6). Input/output specifiable bit-wise. On-chip pull-up resistor specifiable in 4-bit units by software for this port.	o	Input	Ⓕ - A
P61		KR1				
P62		KR2				
P63		KR3				
P70	Input/ output	KR4	4-bit input/output port (PORT7). On-chip pull-up resistor specifiable in 4-bit units by software for this port.		Input	Ⓕ - A
P71		KR5				
P72		KR6				
P73		KR7				

(to be continued)

Table 2-1 List of Digital Input/Output Port Pin Functions (cont'd)

Pin Name	Input/ Output	Dual-Function Pin	Function	8-Bit I/O	When Reset	Input/ Output Circuit Type *1
P80 to P83	Input/ output	TI1	4-bit input/output port (PORT8) On-chip pull-up resistor specifiable in 4-bit units by software for this port.	x	Input	Ⓔ - E
		——				E - B
		AN6				Y - B
		AN7				
BP0	Output	S24	1-bit output port (BIT PORT) Also serves as a segment output pin.	x	*5	G - C
BP1		S25				
BP2		S26				
BP3		S27				
BP4	Output	S28				
BP5		S29				
BP6		S30				
BP7		S31				

*1: Circled circuits have a Schmitt trigger input.

2: Can directly drive LEDs.

3: uPD75P336 has no on-chip pull-up resistor by mask option.

4: uPD75P336 in parentheses

5: V_{LC1} is selected as an input source for BP0 to BP7. The output level varies depending on BP0 to BP7 and the V_{LC1} external circuit. An example is shown on the next page.

Example: Since BP0 to BP7 are interconnected through the uPD75336 as shown below, BP0 to BP7 output levels are determined by the values of R_1, R_2 and R_3.

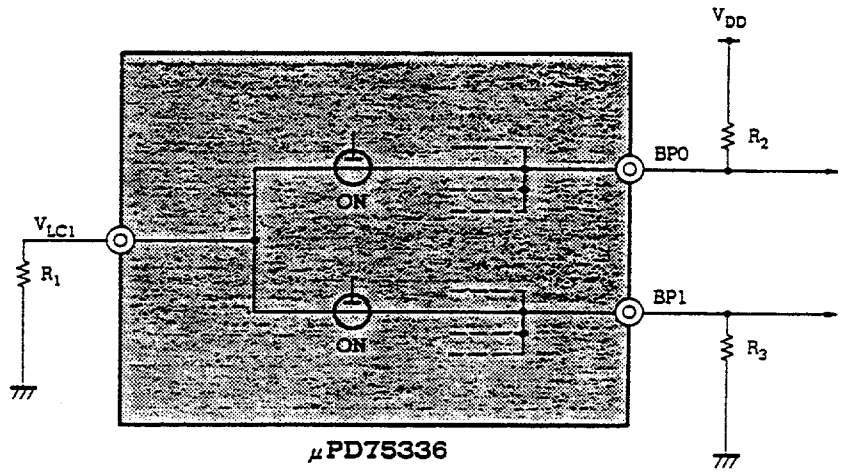


Table 2-2 List of Pins Other than Port Pins

Pin Name	Input/ Output	Dual-Function Pin	Function		When Reset	Input/ Output Circuit Type *1
TI0	Input	P13	External event pulse input for timer/event counter		Input	Ⓑ - C
TI1		P80				Ⓔ - E
PTO0	Output	P20	Timer/event counter output		Input	E - B
PTO1		P21				
PCL	Output	P22	Clock output		Input	E - B
BUZ	Output	P23	Frequency output (for buzzer or system clock trimming)		Input	E - B
$\overline{SCK}$	Input/ output	P01	Serial clock input/output		Input	Ⓕ - A
SO/SB0	Input/ output	P02	Serial data output Serial bus input/output		Input	Ⓕ - B
SI/SB1	Input/ output	P03	Serial data input Serial bus input/output		Input	Ⓜ - C
INT4	Input	P00	Edge-detected vectored interrupt input (valid for detection of both rising and falling edges)		Input	Ⓑ
INT0	Input	P10	Edge-detected vectored interrupt input (detected edge selection enabled)	Clocked	Input	Ⓑ - C
INT1		P11		Asyn-chro-nous		
INT2	Input	P12	Edge-detected vectored interrupt input (detected edge selection enabled)	Asyn-chro-nous	Input	Ⓑ - C
KR0 to KR3	Input	P60 to P63	Parallel falling edge detection testable input		Input	Ⓕ - A
KR4 to KR7	Input	P70 to P73	Parallel falling edge detection testable input		Input	Ⓕ - A

(to be continued)

Table 2-2 List of Pins Other than Port Pins (cont'd)

Pin Name	Input/ Output	Dual-Function Pin	Function	When Reset	Input/ Output Circuit Type *1
S12 to S23	Output	—	Segment signal output	*3	G - A
S24 to S31	Output	BP0 to BP7	Segment signal output	*3	G - C
COM0 to COM3	Output	—	Common signal output	*3	G - B
V_{LC0} to V_{LC2}	Input	—	LCD drive power On-chip split resistor (mask option) *4	—	—
BIAS	Output	—	External split resistor cut output	*5	—
LCDCL *2	Output	P30	Clock output for externally extended driver drive	Input	E - B
SYNC *2	Output	P31	Clock output for externally extended driver synchronization	Input	E - B
AN0 to AN5	Input	—	A/D converter analog signal input	Input	Y
AN6		P82			Y - B
AN7		P83			
AV_{REF}	Input	—	A/D converter reference voltage input	—	Z
AV_{SS}	Input	—	A/D converter reference GND potential	—	Z
X1	Input	—	Crystal/ceramic connection pins for main-system clock oscillation. External clock is input to X1 and its inverted phase is input to X2.	—	—
X2					

(to be continued)

Table 2-2 List of Pins Other than Port Pins (cont'd)

Pin Name	Input/ Output	Dual-Function Pin	Function	When Reset	Input/ Output Circuit Type *1
XT1	Input	—	Crystal connection pins for subsystem clock oscillation. External clock is input to XT1 and XT2 is left open. XT1 can also be used as a 1-bit input (test) pin.	—	—
XT2 *6	—				
$\overline{\text{RESET}}$	Input	—	System reset input	—	(B)
MD0 to MD3	Input/ output	P30 to P33	Only incorporated in uPD75P336. Mode selection for program memory (PROM) write/verify.	Input	E - B
IC	—	—	Only incorporated in uPD75336. Internally Connected. Connect directly to V_{DD}.	—	—
V_{PP}	—	—	Only incorporated in uPD75P336. Program voltage application for program memory (PROM) write/verify Connect directly to V_{DD} for normal operation. Apply +12.5 V for PROM write/verify.	—	—
V_{DD}	—	—	Positive power	—	—
V_{SS}	—	—	GND potential	—	—

*1: Circled circuits have a Schmitt trigger input.

2: Pins reserved for future system expansion. Currently used as P30 and P31 pins only.

3: The following V_{LCX} is selected as an input source for each display output.
S12 to S31: V_{LC1}, COM0 to COM2: V_{LC2}, COM3: V_{LC0}
Each display output level varies depending on each display output and V_{LCX} external circuit.

4: uPD75P336 does not incorporate a split resistor by mask option.

5: When a split resistor is incorporated
... Low level
When a split resistor is not incorporated
... High impedance

6: When no subsystem clock is used, refer to Section 5.2.2 (5).

2.2 DESCRIPTION OF PIN FUNCTIONS

2.2.1 P00 TO P03 (PORT0) ... INT4, $\overline{SCK}$, SO/SB0, AND SI/SB1 DUAL-FUNCTION INPUTS
P10 TO P13 (PORT1) ... INT0, 1, 2 AND TI0 DUAL-FUNCTION INPUTS

4-bit input port: Port 0 and port 1 input pins

Ports 0 and 1 have the following functions in addition to the input port functions.

- Port 0: Vectored interrupt input (INT4)
  Serial interface input/output ($\overline{SCK}$, SO/SB0, SI/SB1)
- Port 1: Vectored interrupt inputs (INT0, INT1)
  Edge-detected test input (INT2)
  External event pulse inputs to the timer/event counter (TI0, TI1)

The pin statuses of ports 0 and 1 can always be input irrespective of dual-function pin operations.

P00/INT4, P01/$\overline{SCK}$, P02/SO/SB0, P03/SI/SB1 inputs of port 0 and each pin of port 1 are Schmitt trigger inputs to prevent errors due to noise. P10 is equipped with a noise eliminator (refer to Section 6.3 (3) "INT0, INT1 and INT4 hardware" for details).

An on-chip pull-up resistor can be specified by the software for port 0 (P01 to P03) as a 3-bit unit or for port 1 (P10 to P13) as a 4-bit unit. This specification can be made by operating pull-up resistor specification register group A.

When the $\overline{RESET}$ signal is generated, all pins are set to the input port mode.

2.2.2 P20 TO P23 (PORT2) ... PTO0, PTO1, PCL AND BUZ DUAL-FUNCTION INPUTS/OUTPUTS

P30 TO P33 (PORT3) ... LCDCL AND SYNC DUAL-FUNCTION INPUTS/OUTPUTS

P40 TO P43 (PORT4),

P50 TO P53 (PORT5) ... N-ch OPEN-DRAIN MIDDLE-VOLTAGE (10 V) HIGH-CURRENT OUTPUT

P60 TO P63 (PORT6),

P70 TO P73 (PORT7),

P80 TO P83 (PORT8) ... TI1, AN6 AND AN7 DUAL-FUNCTION INPUTS/OUTPUTS

4-bit input/output port with output latch:
Ports 2 to 8 input/output pin

Port n (n = 2, 3, 6, 7) carries out the following functions in addition to the input/output port functions.

- Port 2 : Timer/event counter output (PTO0, PTO1)
  Clock output (PCL)
  Any frequency output (BUZ)
- Port 3 : LCD externally extended driver drive clock (LCDCL)
  LCD externally extended driver synchronizing clock (SYNC)
- Ports 6 and 7: Key interrupt inputs (KR0 to KR3, KR4 to KR7)

Port 3 is a high-current output and can directly drive LEDs.

Ports 4 and 5 are N-ch open-drain middle-voltage (10 V) and high-current outputs and can directly drive LEDs.

The input/output mode of each port is selected using the port mode register. Ports 2, 4, 5, 7 and 8 can be specified in 4-bit units and ports 3 and 6 can be specified in bit units.

An on-chip pull-up resistor can be specified in 4-bit units by software for ports 2, 3, 6, 7 and 8 using the pull-up resistor specification registers (POGA, POGB). An on-chip pull-up resistor can be specified bit-wise by mask option for ports 4 and 5 of the uPD75336.

8-bit unit input/output is possible for ports 4 and 5 in pairs or ports 6 and 7 in pairs. When $\overline{\text{RESET}}$ signal is generated, ports 2, 3, 6, 7 and 8 are set to the input mode (output high impedance) and ports 4 and 5 become high level (when a pull-up resistor is incorporated) or high impedance.

2.2.3 BP0 TO BP7 ... CD CONTROLLER/DRIVER SEGMENT SIGNAL OUTPUT (S24 TO S31) DUAL-FUNCTION OUTPUT

1-bit output port with output latch: Output pin of bit ports 0 to 7. Also serve as LCD controller/driver segment signal output pins (S24 to S31). Use this port for CMOS load drive.

2.2.4 TI0 ... PORT 1 DUAL-FUNCTION INPUT
TI1 ... PORT 8 DUAL-FUNCTION INPUT

External event pulse input pins of programmable timer/ event counters 0 and 1.

TI0 and TI1 are Schmitt trigger inputs.

2.2.5 PTO0, PTO1 ... PORT 2 DUAL-FUNCTION OUTPUTS

Output pins of programmable timer/event counters 0 and 1. Generate square wave pulses. When a programmable timer/ event counter signal is output, the output latch is cleared to "0" and port 2 bit of the port mode register is set to output mode "1".

The timer start instruction clears the output to "0".

2.2.6 PCL ... PORT 2 DUAL-FUNCTION OUTPUT

Programmable clock output pin. Used to supply clocks to the peripheral LSIs (such as a slave microcomputer and an A/D converter). When the $\overline{\text{RESET}}$ signal is generated, the clock mode register (CLOM) is cleared to "0", clock output is disabled and this pin is set to the normal port operating mode.

2.2.7 BUZ ... PORT 2 DUAL-FUNCTION OUTPUT

Frequency output pin. Used to generate buzzer sound or trim the system clock oscillator frequency by generating a frequency (2 kHz, 4 kHz, 32 kHz). Also serves as the P23 pin and is only valid when bit 7 (WM7) of the clock mode register (WM) is set to "1".

When the $\overline{\text{RESET}}$ signal is generated, WM7 is cleared to "0" and this pin is set to the normal port operating mode.

2.2.8 $\overline{\text{SCK}}$, SO/SB0, SI/SB1 ... PORT 0 DUAL-FUNCTION 3-STATE INPUTS/OUTPUTS

Input/output pins for the serial interface. Each pin operates according to the setting of the serial operating mode register (CSIM).

When the $\overline{\text{RESET}}$ signal is generated, the serial interface stops operating and becomes an input port.

Each pin is a Schmitt trigger input.

2.2.9 INT4 ... PORT 0 DUAL-FUNCTION INPUT

External vectored interrupt input pin with both active rising and falling edges. When a signal input to this pin changes from low to high or vice versa, the interrupt request flag is set.

INT4 is an asynchronous input and is acknowledged irrespective of the CPU operation clock when a signal having a specified high-level or low-level width is input.

INT4 can also be used to release the STOP mode and HALT mode. It also serves as a Schmitt trigger input.

2.2.10 INT0, INT1 ... PORT 1 DUAL-FUNCTION INPUTS

Edge-detected vectored interrupt input pins. INT0 has a noise elimination function. The detected edge can be selected using the edge detection mode registers (IM0, IM1).

(1) INT0 (IM0 bits 0 and 1)

- (a) Rising edge active
- (b) Falling edge active
- (c) Both rising and falling edge active
- (d) External interrupt signal input disabled

(2) INT1 (IM1 bit 0)

- (a) Rising edge active
- (b) Falling edge active

INT0 has a noise elimination function. The sampling clock for noise elimination can be changed at two levels. The width of an acknowledgeable signal varies depending on the CPU operation clock.

INT1 is an asynchronous input and is acknowledged irrespective of the CPU operation clock if there is an input having the specified high-level width.

When the $\overline{\text{RESET}}$ signal is generated, IM0 and IM1 are cleared to "0" and rising edge active is selected.

INT1 can also be used to release the STOP mode and HALT mode, but INT0 cannot.

INT0 and INT1 are Schmitt trigger inputs.

2.2.11 INT2 ... PORT 1 DUAL-FUNCTION INPUT

External test input pin with both rising and falling edge active. When INT2 is selected by the edge-detection mode register (IM2) or a signal input to this pin changes from low to high, the internal test flag (IRQ2) is set.

INT2 is an asynchronous input and is acknowledged irrespective of the CPU operation clock if there is an input having the specified high-level width.

When the $\overline{\text{RESET}}$ signal is generated, IM2 is cleared to "0" and the test flag (IRQ2) is set by INT2 pin rising edge input.

INT2 can also be used to release the STOP mode and HALT mode. It is a Schmitt-triggered input.

2.2.12 KR0 TO KR3 ... PORT 6 DUAL-FUNCTION INPUTS
KR4 TO KR7 ... PORT 7 DUAL-FUNCTION INPUTS

Key interrupt input pins. KR0 to KR7 are parallel falling edge detected interrupt input pins.

The interrupt format can be specified in accordance with the edge-detection mode register (IM2).

When the $\overline{\text{RESET}}$ signal is generated, input mode is set for ports 6 and 7.

2.2.13 S12 TO S23 ... OUTPUTS
S24 TO S31 ... BIT PORTS 0 TO 7 DUAL-FUNCTION OUTPUTS

Segment signal output pins capable of directly driving the LCD segment pin (front electrode). They execute 2- or 3-time multiplexing based on the static, 1/2 bias method or 3- to 4-time multiplexing based on the 1/3 bias method.

S12 to S23 are segment special-purpose output pins and S24 to S31 also serve as bit ports 0 to 7 output pins. They are switched for use by the display mode register (LCDM).

2.2.14 COM0 TO COM3 ... OUTPUTS

Common signal output pin capable of directly driving the LCD common pin (rear electrode). They generate common signals when 2-time multiplexing (COM0, 1 output) or 3-time multiplexing (COM0, 1, 2 outputs) is executed based on the static (COM0, 1, 2 and 3 outputs), 1/2 bias method, or 3-time multiplexing (COM0, 1, 2 outputs) or 4-time multiplexing (COM0, 1, 2, 3 outputs) is executed based on the 1/3 bias method.

2.2.15 V_{LC0} TO V_{LC2}

LCD drive power supply pins. The uPD75336 can have an on-chip dividing resistor so as to supply LCD drive power in accordance with each bias method to the V_{LC0} to V_{LC2} pins without the use of an external split resistor (mask option).

2.2.16 BIAS

Split resistor cut output pin. Connected to the V_{LC0} pin to cope with various LCD drive voltages in order to change the resistor division ratio. With an external resistor connected, this pin is used together with the V_{LC0} to V_{LC2} pins and V_{SS} pin for fine adjustment of the LCD drive power voltage value.

2.2.17 LCDCL

LCD externally extended driver drive clock output pin

2.2.18 SYNC

LCD externally extended drive cyclic clock output pin

2.2.19 AN0 TO AN5
AN6 & AN7 ... PORT 8 DUAL-FUNCTION INPUTS

8 analog signal input pins for the A/D converter

2.2.20 AV_{REF}

A/D converter reference voltage input pin

2.2.21 AV_{SS}

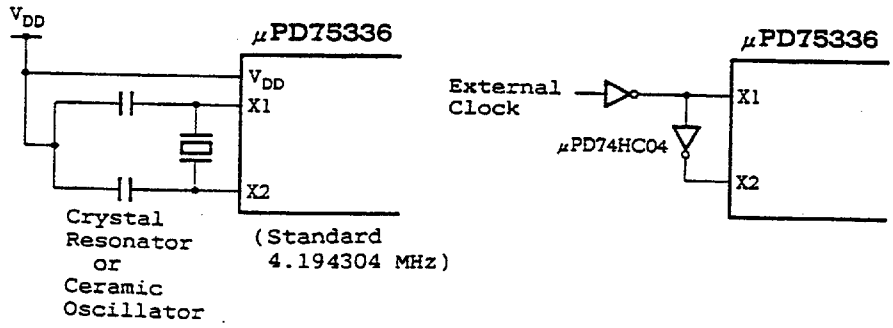
A/D converter GND pin. It should always be set the same voltage as V_{SS}.

2.2.22 X1, X2

Crystal/ceramic connection pins for main-system clock oscillation

External clocks can also be input to these pins.

(a) Crystal/ceramic oscillation (b) External clock

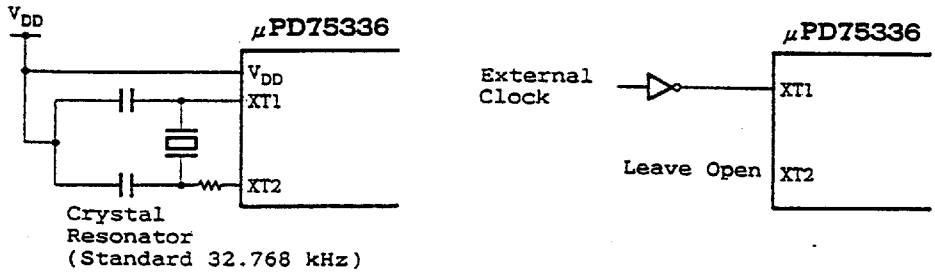


2.2.23 XT1, XT2

Crystal connection pins for subsystem clock oscillation.

External clocks can also be input to these pins.

(a) Crystal oscillation (b) External clock



Remarks: When no subsystem clock is used, refer to Section 5.2.2 (5).

2.2.24 $\overline{\text{RESET}}$

Low-level active reset input pin.

$\overline{\text{RESET}}$ input is an asynchronous input. When a signal having the specified low-level width is input irrespective of the operation clock, the $\overline{\text{RESET}}$ signal is generated and a system reset is applied with priority over all other operations.

In addition to normal CPU initialize/start operations, this pin is used to release the STOP mode and HALT mode.

$\overline{\text{RESET}}$ input is a Schmitt trigger input.

2.2.25 MD0 TO MD3 (uPD75P336 ONLY)

Only the uPD75P336 incorporates MD0 to MD3 pins.

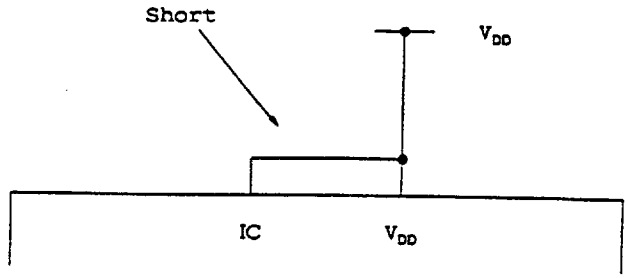
This pin is used to select the mode for write/verify for the program memory (one-time PROM).

2.2.26 IC (uPD75336 ONLY)

The IC (Internally Connected) pin is a pin for setting the test mode in order to test the uPD75336 when shipped by NEC. In normal operation, the IC pin should be connected directly to the V_{DD} pin, and the wiring length should be kept as short as possible.

The customer's program may not function normally if a potential difference arises between the IC pin and the V_{DD} pin when the wiring between the IC pin and V_{DD} pin is long, or external noise is applied to the IC pin.

o Connect the IC pin directly to the V_{DD} pin.



2.2.27 V_{PP} (uPD75P336 ONLY)

Program voltage input pin for program memory (one-time PROM) write/verify.

Connect this pin directly to V_{DD} for normal operation.

2.2.28 V_{DD}

Positive power supply pin.

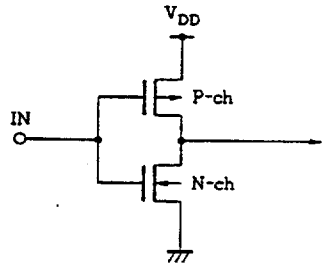
2.2.29 V_{SS}

GND potential

2.3 PIN INPUT/OUTPUT CIRCUITS

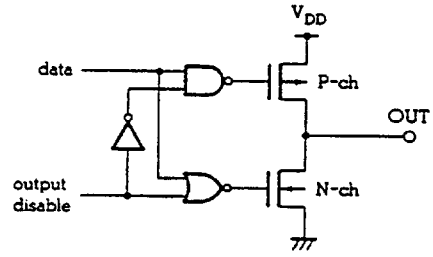
The input/output circuits of the uPD75336 pins are schematically shown below.

TYPE A (For TYPE E-B)



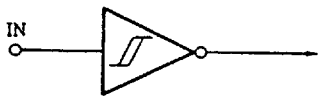
CMOS specified input buffer

TYPE D (For TYPEs E-B and F-A)



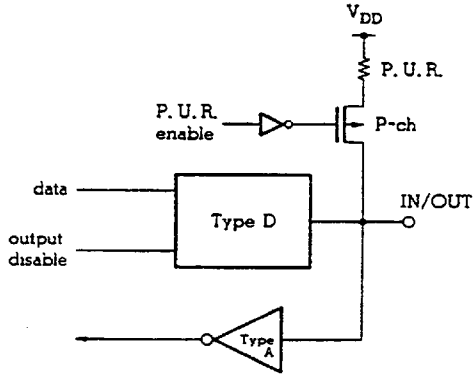
Push-pull output which can be set to output high impedance (with both P-ch and N-ch OFF)

TYPE B



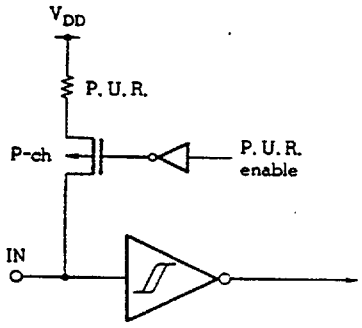
Schmitt trigger input having hysteresis characteristics

TYPE E-B



P.U.R. : Pull-Up Resistor

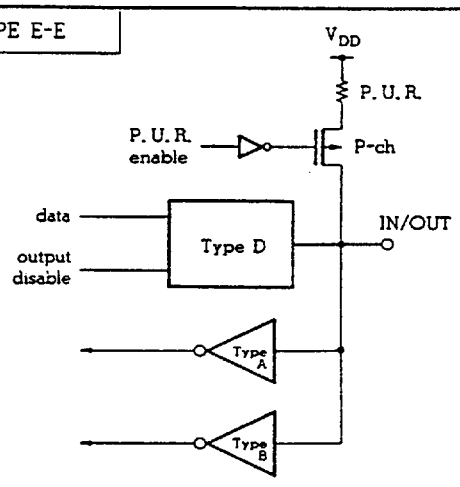
TYPE B-C



P.U.R. : Pull-Up Resistor

Schmitt trigger input having hysteresis characteristics

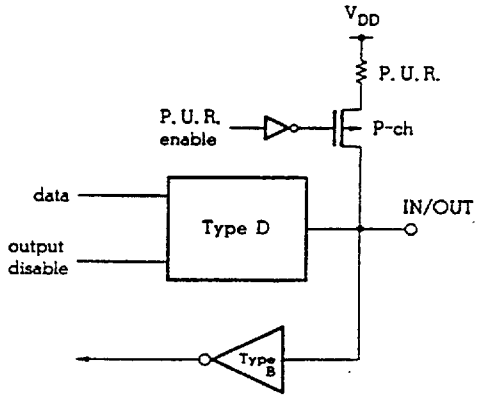
TYPE E-E



P.U.R. : Pull-Up Resistor

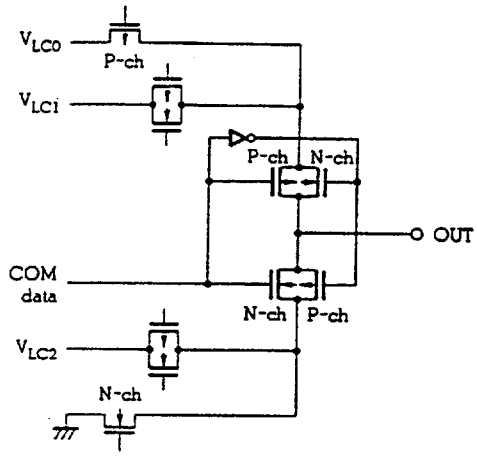
(to be continued)

TYPE F-A

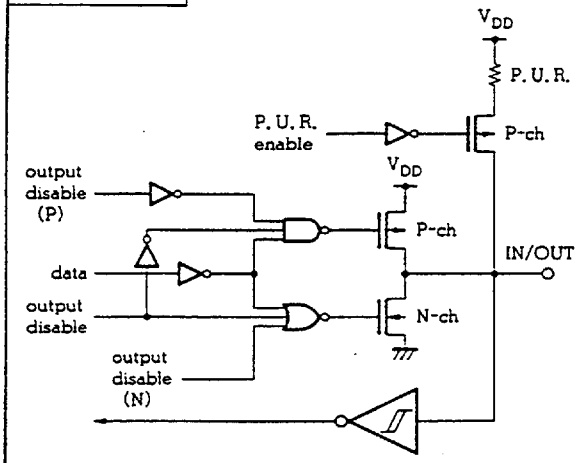


P.U.R. : Pull-Up Resistor

TYPE G-B

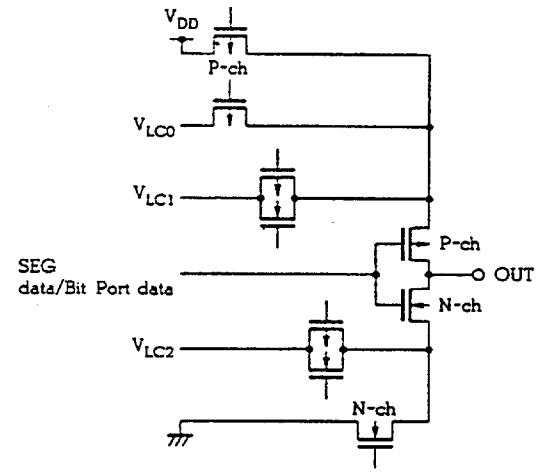


TYPE F-B

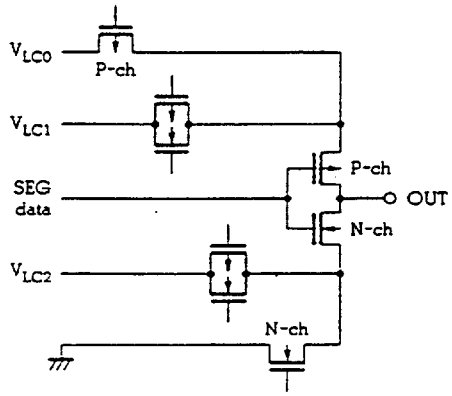


P.U.R. : Pull-Up Resistor

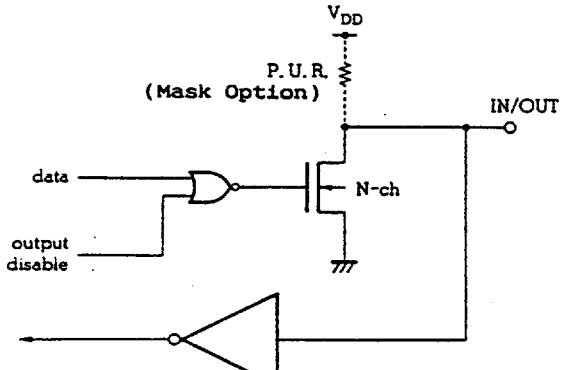
TYPE G-C



TYPE G-A



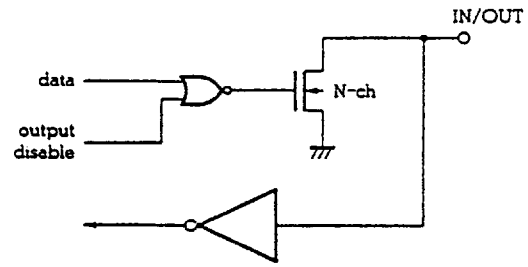
TYPE M



P.U.R. : Pull-Up Resistor

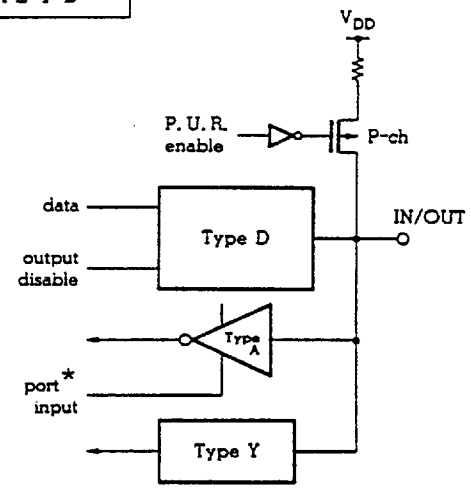
(to be continued)

TYPE M-A



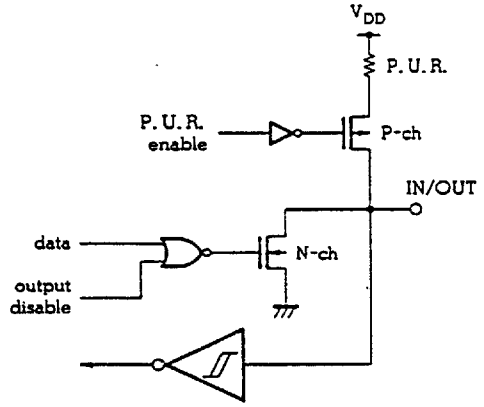
Medium-Voltage Input Buffer

TYPE Y-B



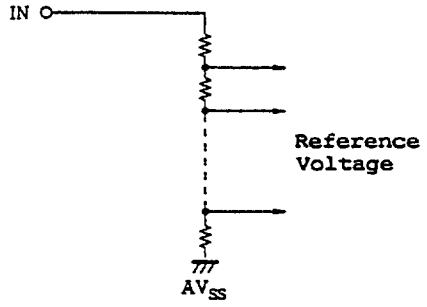
P.U.R. : Pull-Up Resistor

TYPE M-C

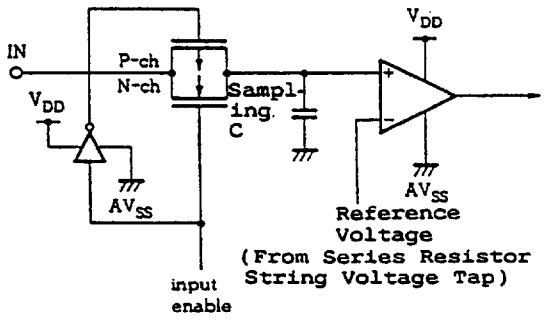


P.U.R. : Pull-Up Resistor

TYPE Z



TYPE Y



*: Becomes active when input instruction is executed.

2.4 RECOMMENDED CONNECTION OF UNUSED PINS

Table 2-3 Recommended Connection of Unused Pins

Pin	Recommended Connection
P00/INT4	Connect to V_{SS}.
P01/$\overline{SCK}$	Connect to V_{SS} or V_{DD}.
P02/SO/SB0	
P03/SI/SB1	
P10/INT0 to P12/INT2	Connect to V_{SS}.
P13/TI0	
P20/PTO0	Input status : Connect to V_{SS} or V_{DD}. Output status: Leave open.
P21/PTO1	
P22/PCL	
P23/BUZ	
P30/P33 (P30/MD0 to P33/MD3)*	
P40 to P43	
P50 to P53	
P60 to P63	
P70 to P73	
P80, P81	
P82/AN6, P83/AN7	
S12 to S23	Leave open.
S24/BP0 to S31/BP7	
COM0 to COM3	
V_{LC0} to V_{LC2}	Connect to V_{SS}.
BIAS	Only when all V_{LC0} to V_{LC2} are not used, connect to V_{SS}. In all other cases, leave open.

(to be continued)

Table 2-3 Recommended Connection of Unused Pins (cont'd)

Pin	Recommended Connection
XT1	Connect to V_{SS} or V_{DD}.
XT2	Leave open.
AN0 to AN5	Connect to V_{SS} or V_{DD}.
IC (V_{PP})*	Connect directly to V_{DD}.

* : Pins only for uPD75P336 in parentheses

2.5 MASK OPTION SELECTION

The following mask options are available for the pins (uPD75336 only).

Table 2-4 Mask Option Selection

Pin	Mask Option	
P40 to P43, P50 to P53	① Without pull-up resistor (Specifiable bit-wise)	② With pull-up resistor (Specifiable bit-wise)
BIAS, V_{LC0} to V_{LC2}	① Without split resistor for LCD drive power supply (Specify in 4 units)	② With split resistor for LCD drive power supply (Specify in 4 units)
XT1, XT2	① With feedback resistor (When subsystem clock is used)	② Without feedback resistor (When subsystem clock is not used)

CHAPTER 3. DATA MEMORY OPERATION AND MEMORY MAP

The 75X architecture employed for the uPD75336 has the following features:

- On-chip RAM with a maximum capacity of 4K words x 4 bits (12-bit address)
- Peripheral hardware extendibility

To achieve these excellent features, the following methods are employed.

(1) Data memory bank configuration
(2) Memory mapped I/O

Each feature is described in detail below.

3.1 DATA MEMORY BANK CONFIGURATION AND ADDRESSING MODE

3.1.1 DATA MEMORY BANK CONFIGURATION

The uPD75336 incorporates static RAM in addresses 000H to 2FFH of the data memory space, of which the 20 x 4 bits in addresses 1ECH to 1FFH are used as display data memory. Peripheral hardware (input/output ports, timers, etc.) is allocated to addresses F80H to FFFH.

For addressing the 12-bit address (4K words x 4 bits) data memory space, the uPD75336 has a memory bank configuration in which the least significant 8-bit address is directly or indirectly specified by an instruction and the most significant 4-bit address is specified by a memory bank (MB).

To specify the MB, the following hardware is incorporated:

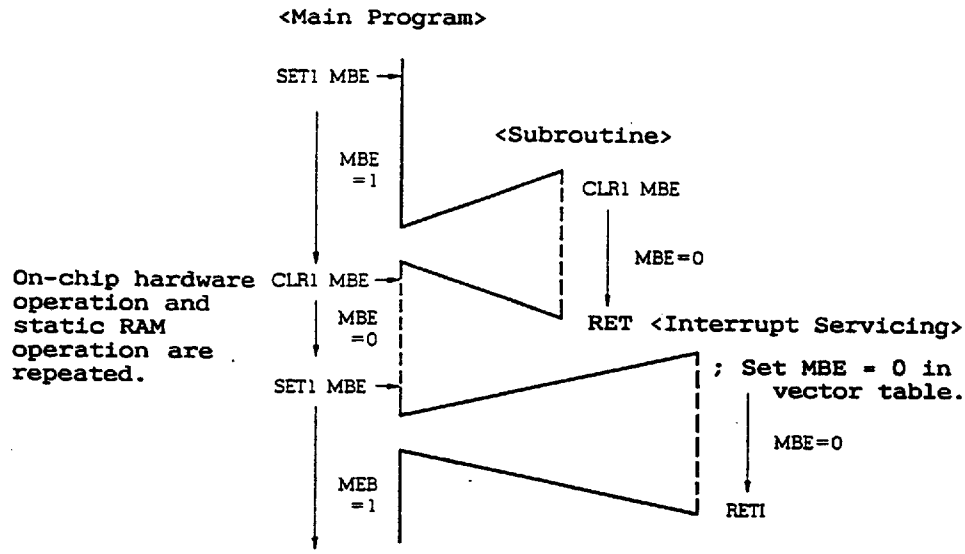
- Memory bank enable flag (MBE)
- Memory bank select register (MBS)

The MBS is a register to select the memory bank and can set 0, 1, 2 and 15 of the memory bank. The MBE is a flag to determine whether the memory bank selected by the MBS should be validated or not. As shown in Figure 3-1, when the MBE is 0, the memory bank (MB) to be specified is fixed irrespective of the MBS. When the MBE is 1, the data memory space can be expanded by switching the memory bank by setting the MBS.

For data memory space addressing, MBE = 1 is normally set and the data memory of the memory bank specified by the MBS is operated. Programming can be carried out efficiently by using the MBE = 0 or MBE = 1 mode for each program operation.

	Applicable Program Operation	Effects
MBE = 0 mode	o Interrupt servicing	MBS save/restore becomes unnecessary.
	o Repetition of on-chip hardware operation and static RAM operation	MBS change becomes unnecessary.
	o Subroutine operation	MBS save/restore becomes unnecessary.
MBE = 1 mode	o Normal program operation	

Figure 3-1 MBE = 0 Mode and MBE = 1 Mode Distinction



Remarks: ——— when MBE = 1; ----- when MBE = 0

Since the MBE is automatically saved/restored in subroutine processing, it can be changed freely in that processing. The MBE is also automatically saved/restored in interrupt servicing, and in addition, the MBE undergoing interrupt servicing can be specified upon interrupt servicing startup by setting the interrupt vector table. Thus, interrupt servicing can be carried out at high-speeds.

When changing the MBS in subroutine processing or interrupt servicing, save/restore it by the PUSH/POP instruction.

Set the MBE by the SET1/CLR1 instruction. Set the MBS by the SEL instruction.

Example 1: Clear the MBE and fix the memory bank.

```
CLR1  MBE; MBE ← 0
```

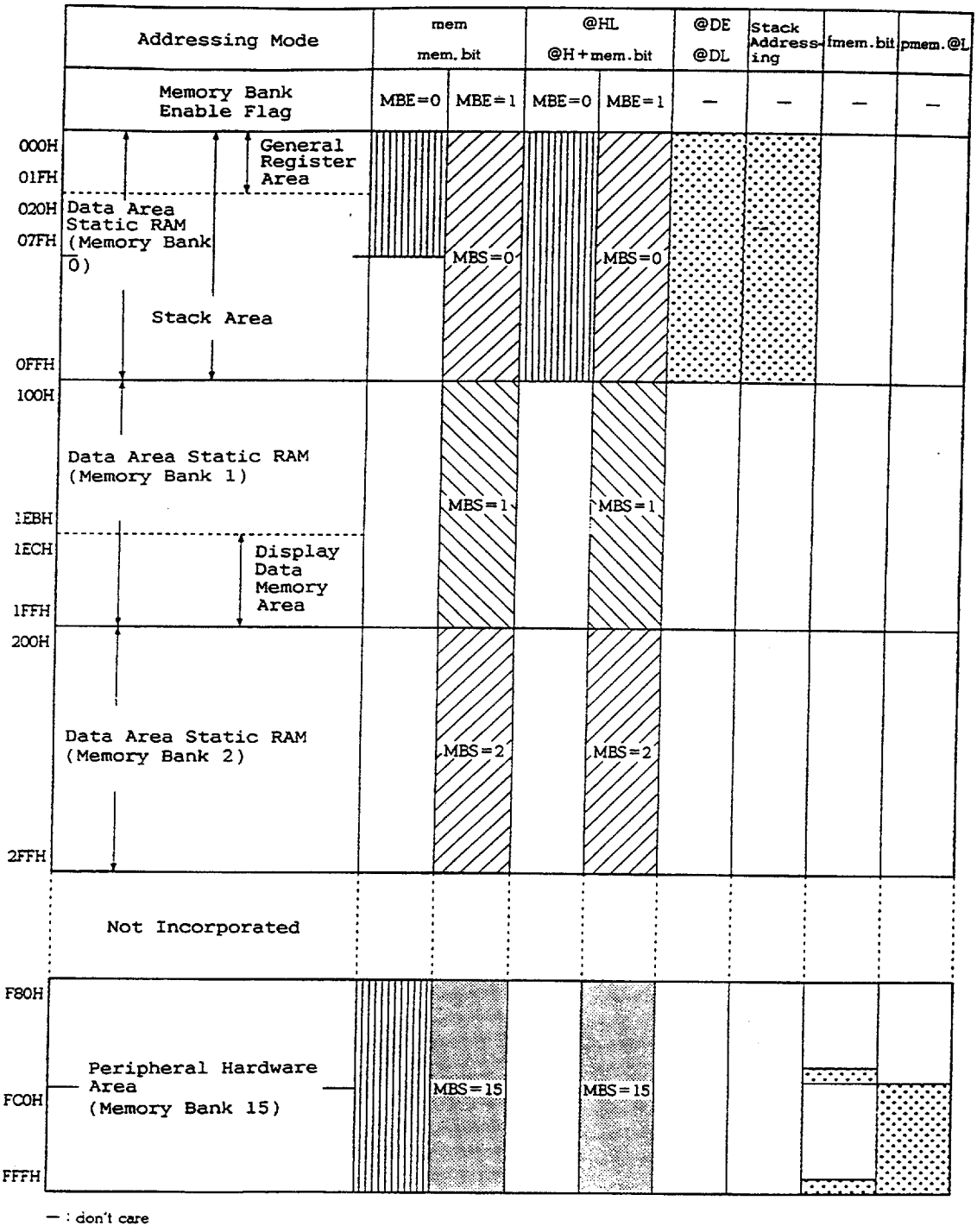
2: Select memory bank 1.

```
SET1  MBE; MBE ← 1
SEL   MB1; MBS ← 1
```

3.1.2 DATA MEMORY ADDRESSING MODE

In the 75X series architecture employed for the uPD75336, seven types of addressing modes shown in Figure 3-2 and Table 3-1 are available for efficient addressing for each bit length of data used for data memory space processing so that programming can be carried out efficiently.

Figure 3-2 Data Memory Configuration and Addressing Range in Each Addressing Mode



— : don't care

Table 3-1 Addressing Mode

Addressing Mode	Identifier	Address to be Specified
1-bit direct addressing	mem.bit	Bit indicated by bit of the address indicated by MB and mem When MBE = 0, { MB = 0 if mem = 00H to 7FH. MB = 15 if mem = 80H to FFH. When MBE = 1, MB = MBS
4-bit direct addressing	mem	Address indicated by MB and mem. When MBE = 0, { MB = 0 if mem = 00H to 7FH. MB = 15 if mem = 80H to FFH. When MBE = 1, MB = MBS
8-bit direct addressing		Address indicated by MB and mem (even address). When MBE = 0, { MB = 0 if mem = 00H to 7FH. MB = 15 if mem = 80H to FFH When MBE = 1, MB = MBS
4-bit register indirect addressing	@HL	Address indicated by MB and HL where MB = MBE·MBS
	@HL+ @HL-	Address indicated by MB and HL where MB = MBE·MBS. HL+ automatically increments L register after addressing. HL- automatically decrements L register after addressing.
	@DE	Address indicated by DE of memory bank 0
	@DL	Address indicated by DL of memory bank 0
8-bit register indirect addressing	@HL	Address indicated by MB and HL where MB = MBE·MBS (Bit 0 of L register is ignored)
Bit manipulation addressing	fmem.bit	Bit indicated by bit of the address indicated by fmem where fmem = { FB0H to FBFH (interrupt-related hardware) FF0H to FFFH (I/O port)
	pmem.@L	Bit indicated by the most significant 2 bits of the L register of the address indicated by the most significant 10 bits of pmem and the least significant 2 bits of the L register where pmem = FC0H to FFFH

(to be continued)

Table 3-1 Addressing Mode (cont'd)

Addressing Mode	Identifier	Address to be Specified
Bit manipulation addressing (cont'd)	@H+mem.bit	Bit indicated by bit of the address indicated by MB, H and the least significant 4 bits of mem where MB = MBE·MBS
Stack addressing		Address indicated by SP of memory bank 0

(1) 1-bit direct addressing (mem.bit)

In this addressing mode each bit of the whole data memory space is directly specified by an instruction operand.

In the MBE = 0 mode the memory bank (MB) to be specified is fixed to MB = 0 when the address specified by the operand is 000H to 07FH and MB = 15 when the address is F80H to FFFH. Thus, in the MBE = 0 mode both the data area 000H to 07FH and the peripheral hardware area F80H to FFFH can be addressed.

In the MBE = 1 mode MB = MBS is set and the data memory space to be specified can be expanded.

This addressing mode can be applied to four instructions; bit set/reset instructions (SET1/CLR1) and bit test instructions (SKT/SKF).

Example: Set FLAG1, reset FLAG2 and test whether FLAG3 is 0.

```
FLAG1  EQU   03FH.1; Bit 1 at address 3FH
FLAG2  EQU   087H.2; Bit 2 at address 87H
FLAG3  EQU   0A7H.0; Bit 0 at address A7H

       SET1  MBE    ; MBE ← 1
       SEL   MB0    ; MBS ← 0
       SET1  FLAG1  ; FLAG1 ← 1
       CLR1  FLAG2  ; FLAG2 ← 0
       SKF   FLAG3  ; FLAG3 = 0?
```

(2) 4-bit direct addressing (mem)

In this address mode the entire data memory space is directly specified in 4-bit units by an instruction operand.

As is the case with the 1-bit direct addressing mode, the specifiable areas are fixed to the data area 000H to 07FH and the peripheral hardware area F80H to FFFH in the MBE = 0 mode. In the MBE = 1 mode, MB = MBS and the data memory space to be specified is expanded to the entire available space.

This addressing mode is applied to the MOV, XCH, INCS, IN and OUT instructions.

Example 1: Enter port 4 and store it in "DATA1".

```
DATA1  EQU   5FH      ; "DATA1" at address
                         5FH
       CLR1  MBE      ; MBE ← 0
       IN    A, PORT4; A ← PORT4
       MOV   DATA1, A; (DATA1) ← A
```

Example 2: Generate "BUFF" data to port 8.

```
BUFF    EQU    11AH      ; "BUFF" at address
                           11AH
        SET1   MBE       ; MBE ← 1
        SEL    MB1       ; MBS ← 1
        MOV    A, BUFF   ; A ← (BUFF)
        SEL    MB15      ; MBS ← 15
        OUT    PORT8, A; PORT8 ← A
```

NOTE: When data related to an input/output port is stored in the static RAM of bank 1 as in this example, program efficiency is decreased. As in the example 1, programming can be carried out without changing MBS if input/output port related data is stored at addresses 000H to 07FH of bank 0.

(3) 8-bit direct addressing (mem)

In this addressing mode the entire data memory space is directly specified by an instruction operand in 8-bit units.

Addresses which can be specified by an operand are even addresses. 4-bit data of the address specified by an operand and 4-bit data of the address plus one undergo 8-bit processing in pairs with the 8-bit accumulator (XA register pair).

The memory bank to be specified is the same as with 4-bit direct addressing.

This addressing mode is applied to the MOV, XCH, IN and OUT instructions.

Example 1: Transfer 8-bit data at ports 4 and 5 to addresses 20H and 21H and generate the data previously placed at addresses 20H and 21H from ports 6 and 7.

```
DATA  EQU   020H
      CLR1  MBE        ; MBE ← 0
      IN    XA, PORT4; XA ← Ports 5 and 4
      XCH   XA, DATA ; XA ↔ (21H, 20H)
      OUT   PORT6, XA; Ports 7 and 6 ← XA
```

2: Fetch 8-bit data input to the serial interface shift register (SIO) and simultaneously set transfer data.

```
      SEL   MB15       ; MBS ← 15
      XCH   XA, SIO    ; XA ↔ SIO
```

(4) 4-bit register indirect addressing (@rpa)

In this addressing mode the data memory space is indirectly specified in 4-bit units using the data pointer (general register pair) specified by an instruction operand.

Three types of data pointers are available. They are the HL register pair which can specify the entire data memory space by specifying MB = MBE·MBS, the DE and DL register pairs which are always fixed to memory bank 0 irrespective of MBE and MBS specification. Programming can be performed efficiently by selecting the appropriate data pointer depending on the data memory bank to be used.

When the HL register pair is specified, the auto increment and auto decrement modes are available to add one to or subtract one from the L register, respectively, simultaneously when an instruction is executed. The program steps can be decreased by using these modes.

Example: Transfer 50H to 57H data to 110H to 117H.

```
DATA1   EQU   57H
DATA2   EQU   117H
        SET1  MBE                    ; MBE ← 1
        SEL   MB1                    ; MBS ← 1
        MOV   D, #DATA1 SHR 4        ; D ← 5
        MOV   HL, #DATA2 AND 0FFH;   HL ← 17H
LOOP:   MOV   A, @DL                 ; A ← (DL)
        XCH   A, @HL-                ; A ↔
                                       (HL),
                                       L ← L - 1
        BR    LOOP
```

The addressing mode with the HL register pair used as the data pointer is widely applied for data transfer, operations, comparison, input/output, etc. The addressing mode using the DE and DL register pairs is applied to the MOV and XCH instructions.

When used with a general register or the register pair increment and decrement instructions, the addresses in the data memory space can be freely updated in this addressing mode as shown in Figure 3-2.

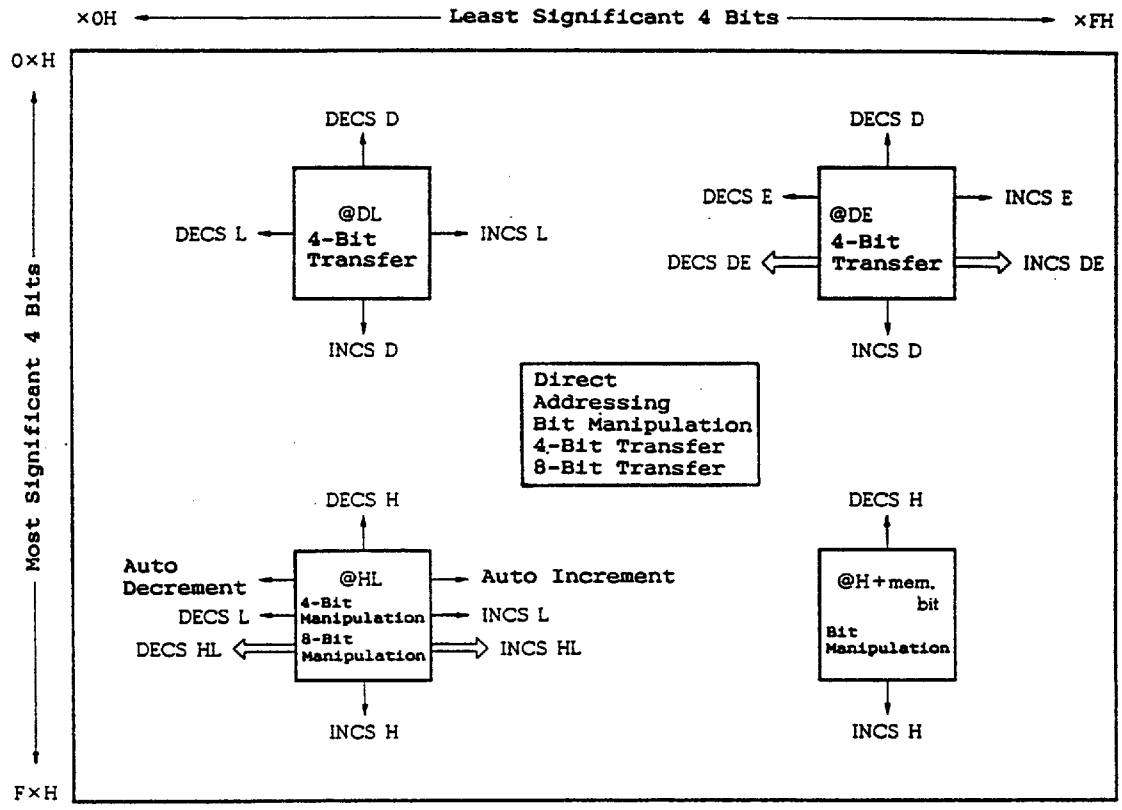
Example 1: Compare 50H to 57H data with 110H to 117H data.

```
DATA1   EQU   57H
DATA2   EQU   117H
        SET1  MBE
        SEL   MB1
        MOV   D, #DATA1 SHR 4
        MOV   HL, #DATA2 AND 0FFH
LOOP:   MOV   A, @DL
        SKE   A, @HL; A = (HL)?
        BR    NO      ; NO
        DECS  L       ; YES, L ← L - 1
        BR    LOOP
```

2: Clear data memory 00H to FFH to "0".

```
        CLR1  RBE
        CLR1  MBE
        MOV   XA, #00H
        MOV   HL, #04H
LOOP:   MOV   @HL, A  ; (HL) ← A
        INCS  HL      ; HL ← HL + 1
        BR    LOOP
```

Figure 3-3 Static RAM Address Updating Method



(5) 8-bit register indirect addressing (@HL)

In this addressing mode the entire data memory space is indirectly specified in 8-bit units using the data pointer (HL register pair).

4-bit data of address with bit 0 of data pointer (bit 0 of L register) and 4-bit data of the address plus one undergo 8-bit processing in pairs with the 8-bit accumulator (XA register).

As is the case with the HL register specified in the 4-bit register indirect addressing mode, the memory bank to be specified is MB = MBS·MBS.

This addressing mode is applied to the MOV, XCH and SKE instructions.

Example 1: Check if the count register (T0) value of timer/event counter 0 is equal to data at addresses 30H and 31H.

```
DATA    EQU    30H
        CLR1   MBE
        MOV    HL, #DATA
        MOV    XA, T0     ; XA ← Count
                                register 0
        SKE    XA, @HL    ; XA = (HL)?
```

2: Clear data memory 00H to FFH to 0.

```
        CLR1   RBE
        CLR1   MBE
        MOV    XA, #00H
        MOV    HL, #04H
LOOP:   MOV    @HL, XA    ; (HL) ← XA
        INCS   HL
        INCS   HL
        BR     LOOP
```

(6) Bit manipulation addressing

In this addressing mode bit manipulation (including Boolean processing and bit transfer) is carried out for each bit in all data memory spaces.

The 1-bit direct addressing mode can only be applied to the bit set, bit reset and bit test instructions. In contrast, in this addressing mode a wide variety of bit manipulations can be performed, such as Boolean processing by AND1, OR1 and XOR1 instructions, bit transfer by the MOV1 instruction and test & reset by the SKTCLR instruction.

The following three bit manipulation addressing methods are available for use depending on the data memory address to be used.

(a) Specific address bit direct addressing (fmem.bit)

In this addressing mode hardware which frequently requires bit manipulation irrespective of memory bank setting can be used. Such hardware includes input/output ports and interrupt related flags among peripheral hardware. Thus, this mode can be applied to data memory addresses FF0H to FFFH with input/output ports mapped and FB0H to FBFH with interrupt related hardware mapped. Hardware in these two data memory areas can perform bit manipulation freely by direct addressing irrespective of MBS and MBE settings.

Example 1: Generate inverted P02 in input from the P33 pin.

```
MOV1  CY, PORT0.2
NOT1  CY
MOV1  PORT3.3, CY
```

2: Test the timer 0 interrupt request flag (IRQT0). If it has been set, clear the request flag and reset P63.

```
SKTCLR  IRQT0   ; IRQT0 = 1?
BR      NO      ; NO
CLR1    PORT6.3; YES
```

3: When both P30 and P41 are 1, reset P53.


```
MOV1  CY, PORT3.0; CY ← P30
AND1  CY, PORT4.1; CY ∧ P41
NOT1  CY         ; CY ← CY
MOV1  PORT5.3, CY; P53 ← CY
```

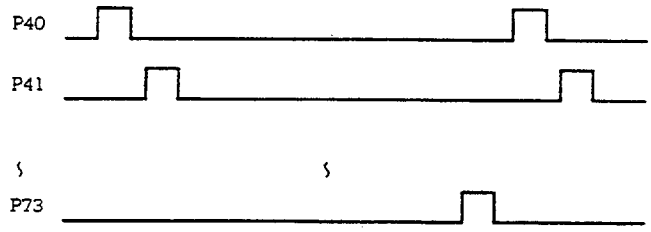
(b) Specific address bit register indirect addressing (pmem.@L)

In this addressing mode the bits of input/output ports among peripheral hardware are indirectly specified using a register and are manipulated continuously. This addressing mode can be applied to the FC0H to FFFH data memory addresses. In addition to input/output ports, a bit-manipulated memory (refer to Section 5.9 "Bit Sequential Buffer") for efficient use of this mode is also included.

In this addressing mode the most significant 10-bit address of the data memory address 12 bits is directly specified by an operand and the least significant 2-bit address and the bit address are indirectly specified using the L register. Thus, 16 bits (4 ports) can be continuously manipulated by L register specification.

This addressing mode also enables bit operation to be executed irrespective of MBE and MBS settings.

Example 1: Generate pulses sequentially to each bit of ports 4 to 7.




```
        MOV   L, #0
LOOP:   SET1  PORT4, @L; Bits (L1 and
                         L0) of ports
                         4 to 7 ← 1
        CLR1  PORT4, @L; Bits (L1 and
                         L0) of ports
                         4 to 7 ← 0
        INCS  L
        BR    LOOP
```

Example 2: When P30 is high, transfer 16-bit serial data input from P31 to the bit sequential buffer (BSB).

```
        MOV   L, #0
LOOP:   SKT   PORT3.0      ; P30 = 1?
        BR    LOOP
        MOV1  CY, PORT3.1; CY ← P31
        MOV1  BSB0.@L, CY; BSB bits
                           (L1 and
                           L0) ← CY
WAIT:   SKF   PORT3.0      ; P30 = 0?
        BR    WAIT
        INCS  L            ; L ← L + 1
        BR    LOOP
```

(c) Specific 1-bit direct addressing (@H+mem.bit)

In this addressing mode bit manipulation can be performed for each bit of the entire data memory space.

In this addressing mode the most significant 4-bit address of the data memory address of the memory bank specified by MB = MBE·MBS is indirectly specified using the H register and the least significant 4-bit address and the bit address are directly specified by an operand. In this addressing mode a wide variety of bit manipulations can be performed for each bit of the entire data memory.

Example: When bit 3 (FLAG1) at address 30H and bit 0 (FLAG2) at address 31H are both 0 or 1, reset bit 2 (FLAG3) at address 32H.


```
FLAG1   EQU   30H.3
FLAG2   EQU   31H.0
FLAG3   EQU   32H.2
        SEL   MB0
        MOV   H, #FLAG1 SHR 6
        MOV1  CY, @H+FLAG1    ; CY ← FLAG1
        XOR1  CY, @H+FLAG2    ; CY ← CY ⊻ FLAG2
        MOV1  @H+FLAG3, CY    ; FLAG3 ← CY
```

(7) Stack addressing

In this addressing mode save/restore operations are carried out in interrupt servicing or subroutine processing.

In this addressing mode the address indicated by the stack pointer (8 bits) of data memory bank 0 is specified.

This addressing mode is also applied to register save/restore operations by the PUSH/POP instruction in addition to operations in interrupt servicing and subroutine processing.

Example 1: Save/restore the register in subroutine processing.

```
SUB  PUSH  XA
     PUSH  HL
     PUSH  BS; MBS and RBS save
        :
     POP   BS
     POP   HL
     POP   XA
     RET
```

2: Transfer HL register pair contents to the DE register pair.

```
     PUSH  HL
     POP   DE; DE ← HL
```

3: Branch to the address indicated by the [XABC] register.

```
     PUSH  BC
     PUSH  XA
     RET      ; Branch to address XABC
```

3.2 GENERAL REGISTER BANK CONFIGURATION

The uPD75336 incorporates four register banks, each bank consisting of eight general registers, X, A, B, C, D, E, H and L. These general register are mapped onto addresses 00H to 1FH of memory bank 0 of the data memory (see Figure 3-5). A register bank enabled flag (RBE) and a register bank select register (RBS) are incorporated to specify the general registers.
RBS is a register which selects the register bank and RBE is a flag which determines if the register bank selected by RBS should be validated or not.

The register bank (RB) which becomes valid for instruction execution is represented as

RB = RBE·RBS

Table 3-2 RBE, RBS and Register Banks Selected

RBE	RBS				Register Bank
	3	2	1	0	
0	0	0	x	x	Fixed to bank 0
1	0	0	0	0	Bank 0 selected
			0	1	Bank 1 selected
			1	0	Bank 2 selected
			1	1	Bank 3 selected

Fixed to 0

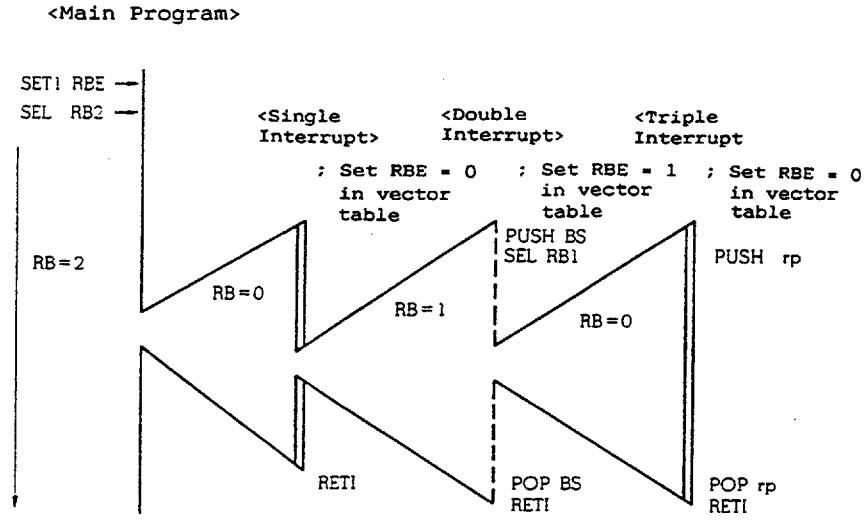
x: Don't care

Since RBE is automatically saved/restored in subroutine processing, it can be freely set in subroutine processing. In interrupt servicing, RBE is also automatically saved/restored, and when interrupt servicing starts, RBE undergoing interrupt servicing can be simultaneously set by interrupt vectored table setting. As shown in Table 3-3, interrupt servicing can be carried out at high speeds by using the appropriate register bank for normal processing and interrupt servicing. In single interrupt servicing, it is not necessary to save/restore general registers. In double interrupt servicing, only RBS is saved/restored

Table 3-3 Recommended Register Bank Usage in Normal and Interrupt Routines

Normal operation	RBE = 1 is set and register banks 2 and 3 are used.
Single interrupt servicing	RBE = 0 is set and register bank 0 is used.
Double interrupt servicing	RBE = 1 is set and register bank 1 is used. (AT the same time, RBS must be saved/ restored.)
Triple or more interrupt servicing	Register is saved/restored by PUSH and POP.

Figure 3-4 Register Bank Usage



Remarks: —— when RB = 2, ═══ when RB = 0, ----- when RB = 1.

When changing RBS in subrouting processing or interrupt servicing, save/restore RBS by the PUSH/POP instruction.

RBE is set by the SET1/CLR1 instruction. RBS is set by the SEL instruction.

```
Example:  SET1   RBE; RBE ← 1
          CLR1   RBE; RBE ← 0
          SEL    RB0; RBS ← 0
          SEL    RB3; RBS ← 3
```

The general register area incorporated in the uPD75336 can be used not only as 4-bit registers but also as 8-bit registers with register pairs. Programming using a general register as the core unit can be performed by transfer, operation, compare and increment/decrement instructions having functions equal to an 8-bit microcomputer.

(1) When used as 4-bit registers

When the general register area is used as 4-bit registers, a total of eight general registers, X, A, B, C, D, E, H and L specified by RB = RBE·RBS, can be used as shown in Figure 3-5. Among these registers, the A register plays an important role as a 4-bit accumulator for 4-bit data transfers, operations and comparisons. The other general registers can transfer, compare and increment/decrement data with the accumulator.

(2) When used as 8-bit registers

When the general register area is used as 8-bit registers, a total of eight 8-bit registers can be used as shown in Figure 3-5. They are register pairs XA, BC, DC and HL, of the register bank specified by RB = RBE·RBS, and register pairs XA', BC', DC' and HL', of the register bank with bit 0 of register bank (RB) inverted. The XA register pair plays an important role as an 8-bit accumulator for 8-bit data transfers, operations, and comparisons. Other register pairs can transfer, calculate, compare and increment/decrement data with the accumulator. The HL register pair functions mainly as a data pointer. The DE register pair functions as an auxiliary data pointer.

```
Example 1:  INCS   HL          ; HL ← HL + 1,
                                 skip when HL = 00H
            ADDS   XA, BC      ; XA ← XA + BC,
                                 skip on carry
            SUBC   DE', XA     ; DE' ← DE' - XA - CY
            MOV    XA, XA'     ; XA ← XA'
            MOVT   XA, @PCDE;  XA ← (PC13 to PC8 + DE)
                                 ROM, table reference
            SKE    XA, BC      ; skip when XA = BC
```

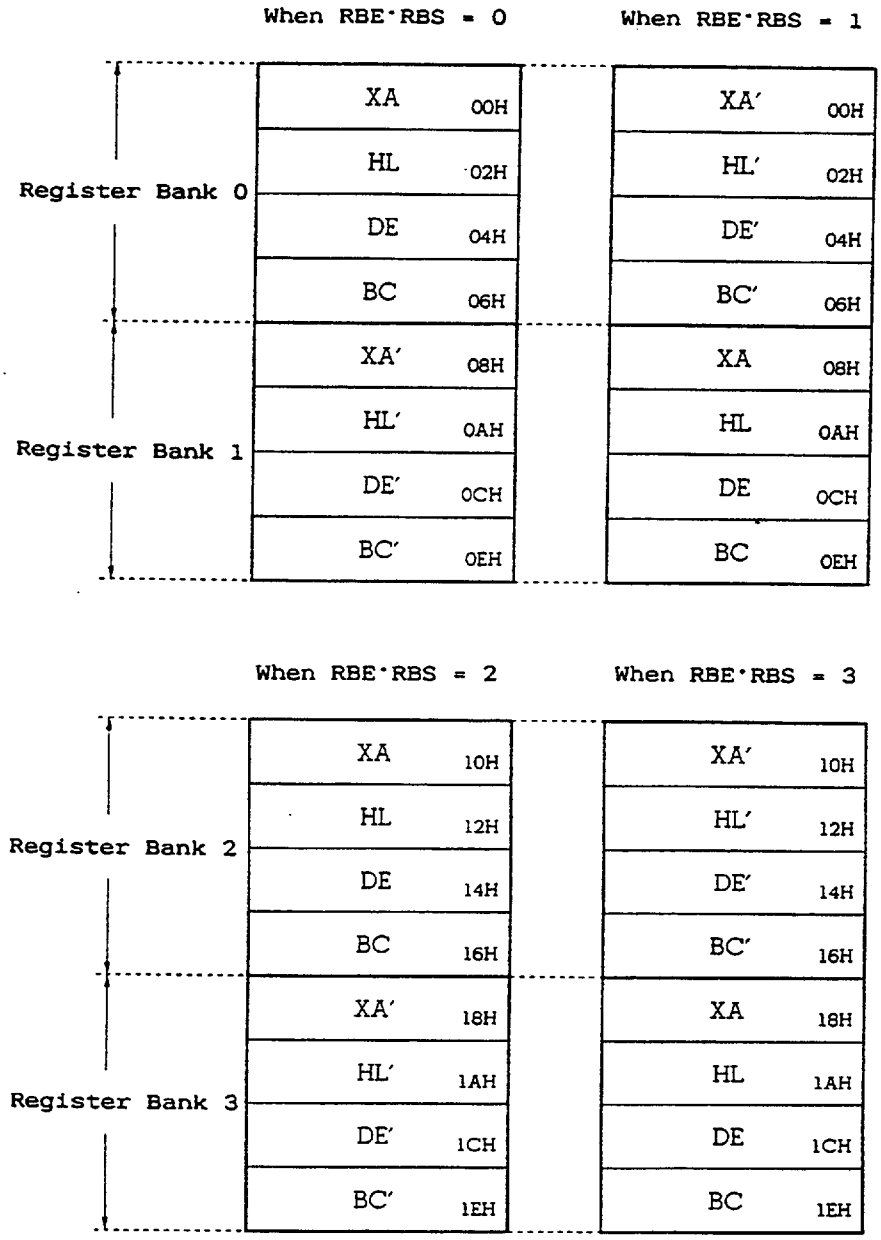
Example 2: Check if the count register (T0) value of timer/event counter 0 is greater than the BC' register pair value. If not, wait until the T0 value becomes greater than the BC' register pair value.

```
        CLR1   MBE
NO:     MOV    XA, T0 ; count register read
        SUBS   XA, BC'; XA ≥ BC?
        BR     YES    ; YES
        BR     NO     ; NO
```

Figure 3-5 General Register Configuration (4-Bit Processing)

X	01H	A	00H	Register Bank 0 (RBE · RBS = 0)
H	03H	L	02H	
D	05H	E	04H	
B	07H	C	06H	
X	09H	A	08H	Register Bank 1 (RBE · RBS = 1)
H	0BH	L	0AH	
D	0DH	E	0CH	
B	0FH	C	0EH	
X	11H	A	10H	Register Bank 2 (RBE · RBS = 2)
H	13H	L	12H	
D	15H	E	14H	
B	17H	C	16H	
X	19H	A	18H	Register Bank 3 (RBE · RBS = 3)
H	1BH	L	1AH	
D	1DH	E	1CH	
B	1FH	C	1EH	

Figure 3-6 General Register Configuration (8-Bit Processing)



3.3 MEMORY MAPPED I/O

As shown in Figure 3-7, the uPD75336 employs memory mapped I/O with the peripheral hardware including input/output ports and timers mapped onto addresses F80H to FFFH in the data memory space. Thus, there are no special instructions to control the peripheral hardware and all operations are controlled by memory manipulation instructions. (Some hardware control mnemonics are available to make the program easy to understand.)

When operating the peripheral hardware, the addressing modes listed in Table 3-4 can be used.

The display data memory mapped onto addresses 1ECH to 1FFH is manipulated by specifying memory bank 1.

Table 3-4 Addressing Modes Applicable when Operating the Peripheral Hardware

	Applicable Addressing Mode	Applicable Hardware
Bit manipulation	Specify by direct addressing mem.bit with MBE = 0 or (MBE = 1, MBS = 15)	All hardware devices enabled for bit manipulation
	Specify by direct addressing fmem.bit irrespective of MBE and MBS	IST0, IST1, MBE, RBE, IExxx, IRQxxx, PORTn.x
	Specify by indirect addressing pmem.@L irrespective of MBE and MBS	BSBn.x PORTn.x
4-bit manipulation	Specify by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15)	All hardware devices enabled for 4-bit manipulation
	Specify by register indirect addressing @HL with (MBE = 1, MBS = 15)	

(to be continued)

Table 3-4 Addressing Modes Applicable when Operating the Peripheral Hardware (cont'd)

	Applicable Addressing Mode	Applicable Hardware
8-bit manipulation	Specify by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15) (mem is an even address)	All hardware devices enabled for 8-bit manipulation
	Specify by register indirect addressing @HL with MBE = 1 and MBS = 15 (L register contents are even)	

```
Example:  CLR1    MBE       ; MBE = 0
          SET1    TM0.3     ; Timer 0 start
          EI      IE0       ; INT0 enabled
          DI      IEI1      ; INT1 disabled
          SKTCLR  IRQ2      ; INT2 request flag test, clear
          SET1    PORT4.@L  ; Port 4 set
          IN      A, PORT0  ; A ← Port 0
          OUT     PORT4, XA; Ports 5 and 4 ← XA
```

Figure 3-7 shows the uPD75336 I/O map.

In the figure, each item has the following meaning:

- Symbol .. Name indicating the on-chip hardware address Can be described in the instruction operand column.
- R/W ..... Indicates whether the corresponding hardware is enabled for read/write.
  - R/W: Read/write enabled
  - R : Read only
  - W : Write only

. No. of manipulable bits

..... Indicates the number of applicable bits before operating the corresponding hardware

o: Bit manipulation enabled in the units specified in the column (1-, 4- and 8-bit)

Δ: Only limited number of bits enabled for manipulation. For manipulable bits, refer to the remarks column.

-: Bit manipulation disabled in the units specified in the column (1-, 4- and 8-bit)

. Bit manipulation addressing

..... Indicates the applicable bit manipulation addressing before operating the corresponding hardware

Figure 3-7 uPD75336 I/O Map

Address	Hardware Name (Symbol) b3 b2 b1 b0	R/W	No. of Manipulable Bits 1-Bit	4-Bit	8-Bit	Bit Manipulation Addressing	Remarks
F80H	Stack pointer (SP)	R/W	-	-	o		Bit 0 is fixed to 0.
			-	-			
F85H	Basic interval timer mode register (BTM)	W	Δ	o	-	mem.bit	Only bit 3 is bit-manipulable.
F86H	Basic interval timer (BT)	R	-	-	o		
			-	-			
F8CH	Display mode register (LCDM)	W	Δ	-	o	mem.bit	Only bit 3 is bit-manipulable.
			-	-			
F8EH	Display control register (LCDC)	W	-	o	-		

(to be continued)

Figure 3-7 uPD75336 I/O Map (cont'd)

Address	Hardware Name (Symbol) b3	b2	b1	b0	R/W	No. of Manipulable Bits 1-Bit	4-Bit	8-Bit	Bit Manipulation Addressing	Remarks
F98H	Clock mode register (WM)				R/W	Δ (R)	-	o (W)	mem.bit	Only bit 3 is bit-testable. *1
						-	-			

FA0H	Timer/event counter 0 mode register (TM0)				W	Δ	-	o	mem.bit	Only bit 3 is bit-manipulable.
						-	-			
FA2H	TOE0*2				W	o	-	-	mem.bit	
FA4H	Timer/event counter 0 count register (T0)				R	-	-	o		
FA6H	Timer/event counter 0 modulo register (TMOD0)				W	-	-	o		
FA8H	Timer/event counter 1 mode register (TM1)				W	Δ	-	o	mem.bit	Only bit 3 is bit-manipulable.
						-	-			
FAAH	TOE1*2				W	o	-	-	mem.bit	
FACH	Timer/event counter 1 count register (T1)				R	-	-	o		
FAEH	Timer/event counter 1 modulo register (TMOD1)				W	-	-	o		
FB0H	IST1	IST0	MBE	RBE	R/W	o (R/W)	o (R/W)	o (R)	fmem.bit	Bit 8 is bit-manipulable only for R.
	Program status word (PSW)					-	-			
	CY	SK2	SK1	SK0						
FB2H	Interrupt priority select register (IPS)				W	-	o	-		Only bit 3 is manipulated by EI/DI instruction.

(to be continued)

Figure 3-7 uPD75336 I/O Map (cont'd)

Address	Hardware Name (Symbol) b3	b2	b1	b0	R/W	No. of Manipulable Bits 1-Bit	4-Bit	8-Bit	Bit Manipulation Addressing	Remarks
FB3H	Processor clock control register (PCC)				W	-	o	-		*3
FB4H	INT0 mode register (IM0)				W	-	o	-		Bit 2 is fixed to 0.
FB5H	INT1 mode register (IM1)				W	-	o			Bits 3, 2 and 1 are fixed to 0.
FB6H	INT2 mode register (IM2)				W	-	o	-		Bits 3 and 2 are fixed to 0.
FB7H	System clock control register (SCC)				W	o	-		fmem.bit	Bits 2 and 1 are fixed to 0.
FB8H	IE4	IRQ4	IEBT	IRQBT	R/W	o	o	-	fmem.bit	
FBAH			IEW	IRQW	R/W	o	o			
FBCH	IET1	IRQT1	IET0	IRQT0	R/W	o	o	-		
FBDH			IECSI	IRQCSI	R/W	o	o			
FBEH	IE1	IRQ1	IE0	IRQ0	R/W	o	o	-		
FBFH			IE2	IRQ2	R/W	o	o			

FC0H	Bit sequential buffer 0 (BSB0)	R/W	o	o	o	mem.bit pmem.@L		
FC1H	Bit sequential buffer 1 (BSB1)	R/W	o	o				
FC2H	Bit sequential buffer 2 (BSB2)	R/W	o	o	o			
FC3H	Bit sequential buffer 3 (BSB3)	R/W	o	o				

(to be continued)

Figure 3-7 uPD75336 I/O Map (cont'd)

Address	Hardware Name (Symbol) b3 / b2 / b1 / b0	R/W	No. of Manipulable Bits: 1-Bit	4-Bit	8-Bit	Bit Manipulation Addressing	Remarks
FD0H	Clock output mode register (CLOM)	W	-	o	-		
FD8H	SOC / EOC — A/D conversion mode register (ADM)	R/W	△ (R) (W)	-	o (W)	mem.bit	EOC (R) SOC (W)*4
			-	-			
FDAH	SA register (SA)	R	-	-	o		
			-	-			
FDCH	Pull-up resistor specification register group A (POGA)	W	-	-	o		
			-	-			
FDEH	Pull-up resistor specification register group B (POGB)	W	-	-	o		
			-	-			

FE0H	Serial operation mode register (CSIM)	R/W	-	-	o (W)		
FE1H	CSIE / COI / WUP		△ (R) (W)	o		mem.bit	Bits 3, 2 and 1 are bit-manipulable. *4
FE2H	CMDD / RELD / CMDT / RELT — SBI control register (SBIC)	R/W	o	-	-	mem.bit	R/W depends on the bit.
FE3H	BSYE / ACKD / ACKE / ACKT		o	-			
FE4H	Serial I/O shift register (SIO)	R/W	-	-	o		
			-	-			
FE6H	Slave address register (SVA)	W	-	-	o		
			-	-			
FE8H	Port mode register group A (PMGA)	W	-	-	o		
			-	-			

(to be continued)

Figure 3-7 uPD75336 I/O Map (cont'd)

Address	Hardware Name (Symbol) b3	b2	b1	b0	R/W	No. of Manipulable Bits 1-Bit	4-Bit	8-Bit	Bit Operation Addressing	Remarks
FECH	Port mode register group B (PMGB)				W	-	-	o		
						-	-			
FEEH	Port mode register group C (PMGC)				W	-	-	o		
						-	-			

FF0H	Port 0 (PORT0)				R	o	o	-	fmem.bit pmem.@L	
FF1H	Port 1 (PORT1)				R	o	o			
FF2H	Port 2 (PORT2)				R/W	o	o	-		
FF3H	Port 3 (PORT3)				R/W	o	o			
FF4H	Port 4 (PORT4)				R/W	o	o	o		
FF5H	Port 5 (PORT5)				R/W	o	o			
FF6H *5	KR3	KR2	KR1	KR0	R/W	o	o	o		
	Port 6 (PORT6)									
FF7H *5	KR7	KR6	KR5	KR4	R/W	o	o			
	Port 7 (PORT7)									
FF8H	Port 8 (PORT8)				R/W	o	o	-		

*1: 1-bit manipulation is available only for R and 8-bit manipulation is available only for W.

2: TOE0, 1: Timer/event counter 0 or 1 output enable flag

3: Bits 2 and 3 may manipulate during STOP/HALT instruction execution.

4: 1-bit manipulation differ with R/W bit-wise.
8-bit manipulation is available only for R.

5: KR0 to KR7 may only be read. For 4-bit parallel input, specify them with PORT6 or PORT7.

Remarks 1: IExxx is the interrupt enable flag.

2: IRQxxx is the interrupt request flag.

3: IME is the interrupt master enable flag.

CHAPTER 4. INTERNAL CPU FUNCTIONS

4.1 PROGRAM COUNTER (PC) ... 14 BITS

This is a 14-bit binary counter to hold the program memory address information.

Figure 4-1 Program Counter Configuration

PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PC

The program counter is normally incremented automatically in accordance with the number of bytes of an instruction each time the instruction is executed.

When a branch instruction (BR, BRCB) is executed, immediate data indicating the branch destination address and the register pair contents is loaded into all or some bits of the PC.

When a subroutine call instruction (CALL, CALLF) is executed or a vectored interrupt is generated, the PC contents (the return address which has already been incremented to fetch the next instruction) are saved into the stack memory (the data memory specified by the stack pointer) and then each jump destination address is loaded.

When a return instruction (RET, RETS, RETI) is executed, the stack memory contents are set in the PC.

When the $\overline{\text{RESET}}$ signal is generated, the least significant 6 bits at address 0000H of the program memory are set to PC13 to PC8 and the contents at address 0001H are set to PC7 to PC0 and the program is initialized. Therefore, the program can be started at any desired address.

4.2 PROGRAM MEMORY (ROM) ... 16256 WORDS x 8 BITS

This is a mask programmable ROM, with a configuration of 16256 words x 8 bits, which is used to store programs, interrupt vector tables, GETI instruction reference tables, table data, etc.

The program memory is addressed by the program counter. Table data can be referred to by the table reference instruction (MOVT).

The branch address range available for branch and subroutine call instructions is shown in Figure 4-2. In addition to these instructions, the BR PCDE, BR PCXA instructions can branch to the address with only the least significant 8 bits of the PC changed.

The program memory addresses are 0000H to 3F7F and the following addresses are especially assigned. (All areas except 0000H and 0001H can be used as normal program memory.)

- Addresses 0000H to 0001H

  Vector address table for writing the program start address for $\overline{\text{RESET}}$ and the RBE and MBE set values. (Can be reset and started at any desired address.)

- Addresses 0002H to 000DH

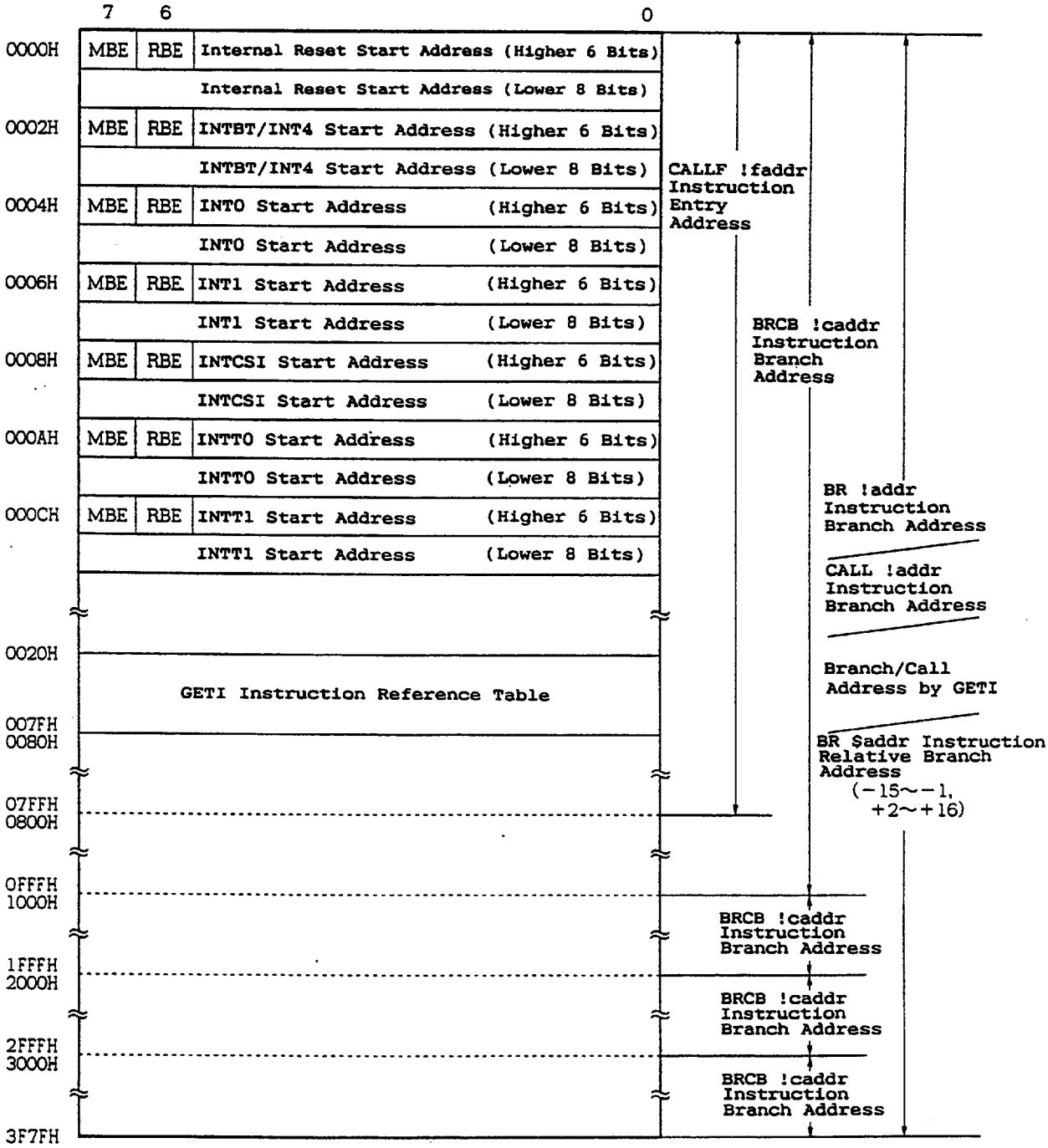
  Vector address table for writing the program start address to be set by each vectored interrupt and the RBE and MBE set values. (Interrupt servicing can be started at any desired address.)

- Addresses 0020H to 007FH

  Table area to be referred to by GETI instruction.*

*: The GETI instruction is an instruction which implements any 2-byte/3-byte instruction with one byte. It enables the number of program bytes to be decreased.

Figure 4-2 Program Memory Map



Remarks: In all cases other than those listed above, a branch to the address with only the least significant 8 bits of the PC changed is enabled by BR PCDE and BR PCXA instructions.

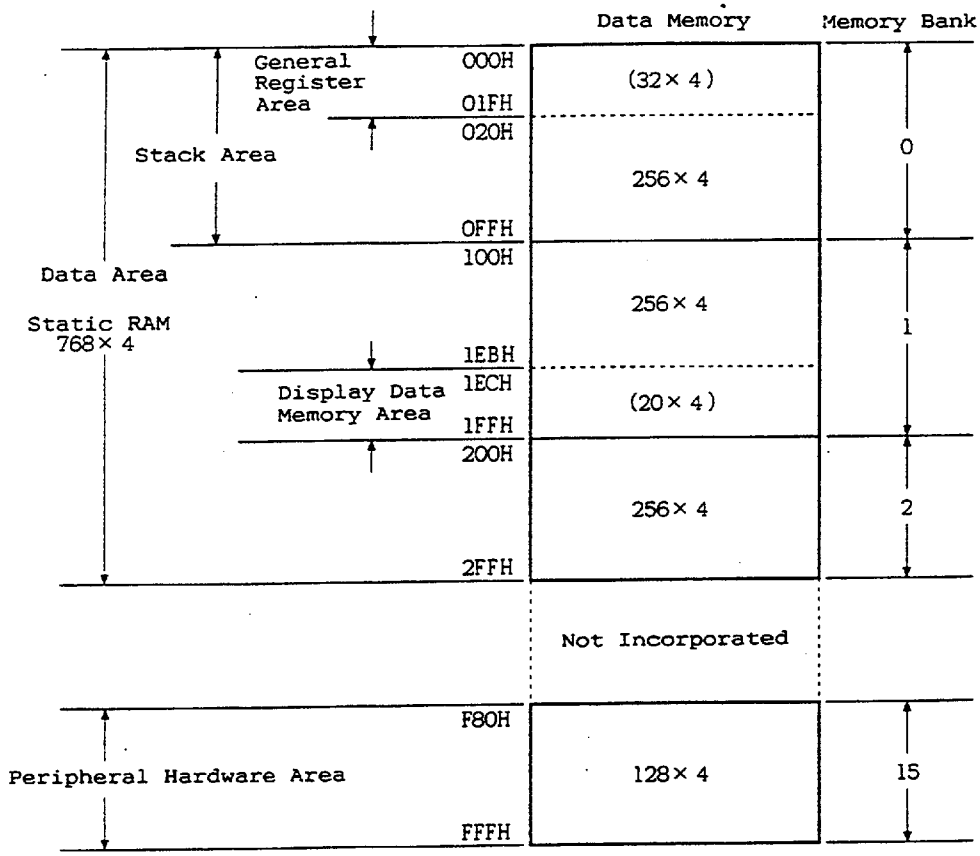
4.3 DATA MEMORY (RAM) ... 768 WORDS x 4 BITS

The data memory consists of a data area and a peripheral hardware area as shown in Figure 4-3.

The data memory has a bank configuration, each bank consisting of 256 words x 4 bits. The following memory banks are available:

. Memory banks 0, 1 and 2 (data area)
. Memory bank 15 (peripheral hardware area)

Figure 4-3 Data Memory Map



(1) Data area

The uPD75336 data area has a static RAM (768 words x 4 bits) configuration. The data area is used to store processing data and is manipulated by memory manipulation instructions.

The static RAM is mapped in memory banks 0, 1 and 2 (256 words x 4 bits each). Bank 0 is intended for mapping as a data area. It can also be used as a general register area (000H to 007H) and a stack area (000H to 0FFH). Bank 1 can also be used as a data area and a display data memory area. Bank 2 can only be used as a data area.

One address of the static RAM consists of 4 bits. However, the static RAM can be manipulated in 8-bit units by 8-bit memory manipulation instructions or bit-wise units by bit manipulation instructions. In the case of 8-bit manipulation instructions, specify an even address.

. General register area

This area can be manipulated by general register manipulation instructions or memory manipulation instructions.

There are 4-bank registers, each bank consisting of eight 4-bit registers. The section which is not used for general registers can be used as a data area or a stack area.

. Stack area

This area is set by an instruction and can be used as a save area for subroutine execution and interrupt servicing operation.

. Display data memory area

The area which is not used for display purposes can be used as a normal data area. This area can be manipulated bit-wise or in 4-bit units using memory manipulation instructions. It cannot be manipulated in 8-bit units (refer to Figure 4-4 "Display Data Memory Configuration" for details).

(2) Peripheral hardware area

This area is mapped onto addresses F80H to FFFH of memory bank 15. As is the case with the static RAM, the peripheral hardware area is manipulated by memory manipulation instructions. It should be noted that the peripheral hardware has different manipulable bit units for each address. The addresses for which no peripheral hardware has been allocated cannot be accessed because no data memory is incorporated (refer to Figure 3-7 "uPD75336 I/O Map").

For the procedure for using a specific area of data memory, refer to the following sections.

. General register area
... Section 4.4 "General Register"

. Stack memory area
... Section 4.6 "Stack Pointer"

. Display data memory
... Section 5.7.5 "Display Data Memory"

. Peripheral hardware
... Chapter 5. "Peripheral Hardware Functions"

The memory bank is specified by the 4-bit memory bank select register (MBS) (MBS = 0, 1, 15) when bank specification is enabled by the memory bank enable flag (MBE) (MBE = 1). If bank specification is disabled (MBS = 0), bank 0 or 15 is automatically specified according to the current addressing mode. Addresses in the bank are set by 8-bit immediate data or register pairs.

For details of memory bank selection and addressing, refer to Section 3.1 "Data Memory Bank Configuration and Addressing Mode".

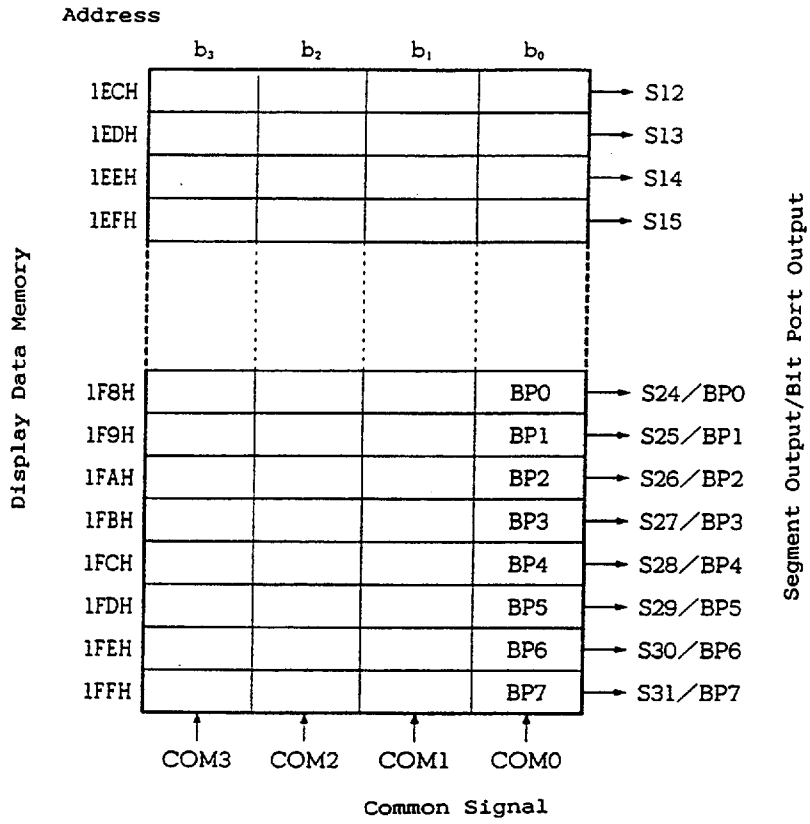
When reset, the data memory is indeterminate. Thus, be sure to initialize it to "0" at the beginning of the program (RAM clear). Otherwise, bugs may be generated.

Example: Clear the RAM at addresses 000H to 2FFH.

```
        STE1  RBE
        SEL   RB0
        SET1  MBE
        SEL   MB0
        MOV   XA, #00H
        MOV   HL, #04H
RAMC0:  MOV   @HL, A  ; 04H to FFH clear*
        INCS  HL      ; HL ← HL + 1
        BR    RAMC0
        SEL   MB1
RAMC1:  MOV   @HL, A  ; 100H to 1FFH clear
        INCS  HL      ; HL ← HL + 1
        BR    RAMC1
RAMC2:  MOV   @HL, A  ; 200H to 2FFH clear
        INCS  HL      ; HL ← HL + 1
        BR    RAMC2
```

*: Do not clear data memory 000H to 003H because it is used as general registers XA and HL.

Figure 4-4 Display Data Memory Configuration



The display data memory is manipulated bit-wise or in 4-bit units.

NOTE: The display data memory cannot be manipulated in 8-bit units.

Example: Clear 1ECH to 1FFH display data memory.

```
        SET1  MBE
        SEL   MB1
        MOV   HL. #0ECH
        MOV   A, #00H
LOOP:   MOV   @HL. A    ; Display data memory
                          is cleared to "0"
                          in 4-bit units at
                          one time.
        INCS  HL
        BR    LOOP
```

4.4 GENERAL REGISTER ... 8 x 4 BITS x 4 BANKS

The general registers are mapped onto special addresses of the data memory. There are 4-banks of registers, each bank consisting of eight 4-bit registers (B, C, D, E, H, L, X, A).

The register bank (RB) which becomes valid for instruction execution is given as

RB = RBE·RBS (RBS = 0 to 3)

Each general register is manipulated as a 4-bit unit. BC, DE, HL and XA form register pairs and are used for 8-bit manipulation. In addition to DE and HL, DL also makes up a pair and these three pairs can be used as data pointers.

In 8-bit unit manipulation, the register pairs of the register bank with register bank (RB) bit 0 inverted (0 ↔ 1, 2 ↔ 3) can be used as BC', DE', HL' and XA' in addition to BC, DE, HL and XA (refer to Section 3.2 "General Register Bank Configuration").

The general register area can be accessed by address specification as normal RAM whether or not it is used as registers.

Figure 4-5 General Register Configuration

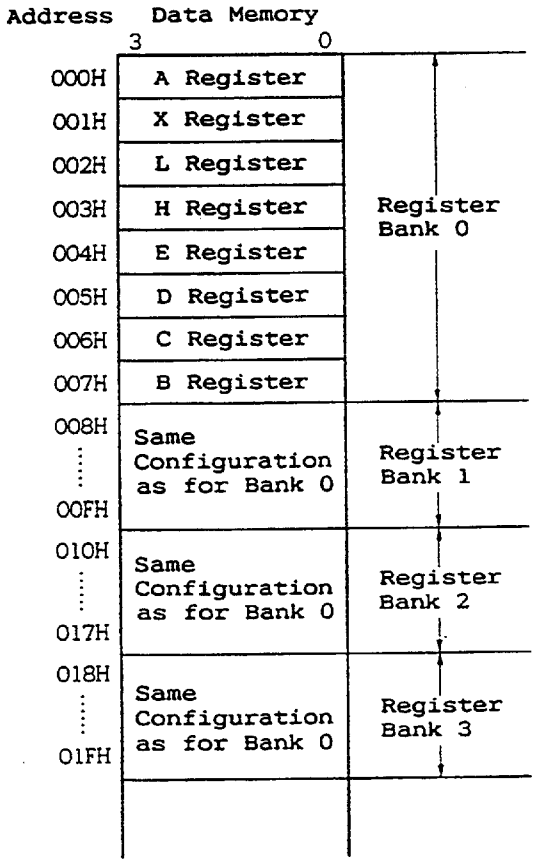
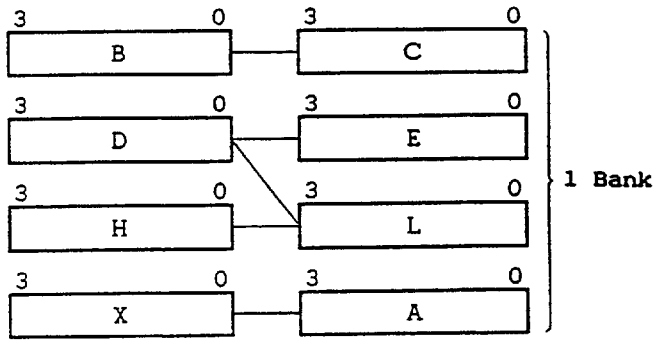


Figure 4-6 Register Pair Configuration

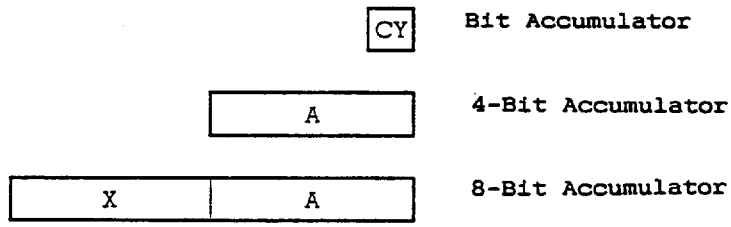


4.5 ACCUMULATOR

In the uPD75336, the A register and XA register pair function as an accumulator. 4-bit data processing instructions are executed mainly by the A register and 8-bit data processing instructions are executed mainly by the XA register pair.

In execution of a bit manipulation instruction, the carry flag (CY) functions as a bit accumulator.

Figure 4-7 Accumulator



4.6 STACK POINTER (SP) ... 8 BITS

In the uPD75336, the static RAM is used as a stack memory (LIFO type) and the 8-bit register which holds the start address information in the stack area is the stack pointer (SP).

The stack area is located at addresses 000H to 0FFH of memory bank 0 irrespective of MBE and MBS settings.

The SP is decremented before a data write (save) to the stack memory and incremented after a data read (restore) from the stack memory.

The data saved/restored by each stack operation is shown in Figures 4-9 and 4-10.

The SP is mapped in the data memory space and can be read and written by 8-bit memory manipulation instructions.

"0" is always written to SP0.

It is recommended to set the SP initial value to 00H and use the SP from the most significant address (0FFH) of data memory bank 0 as a stack area.

When the $\overline{\text{RESET}}$ signal is generated, SP contents become indeterminate. Thus, be sure to initialize the SP at the beginning of the program.

Example: SP initialization

```
SEL   MB15     ; or CLR1 MBE
MOV   XA, #00H
MOV   SP, XA   ; SP ← 00H
```

Figure 4-8 Stack Pointer Configuration

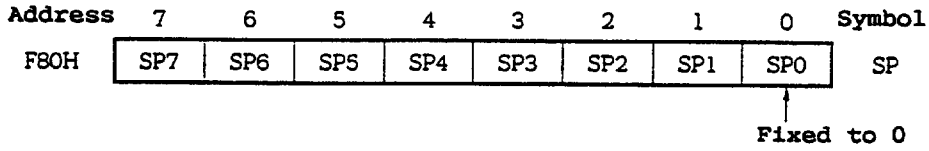


Figure 4-9 Data Saved into Stack Memory

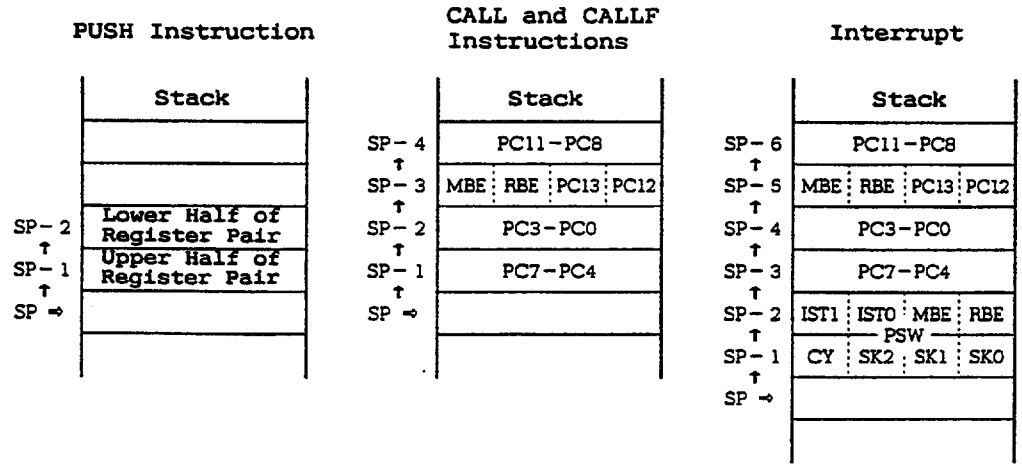
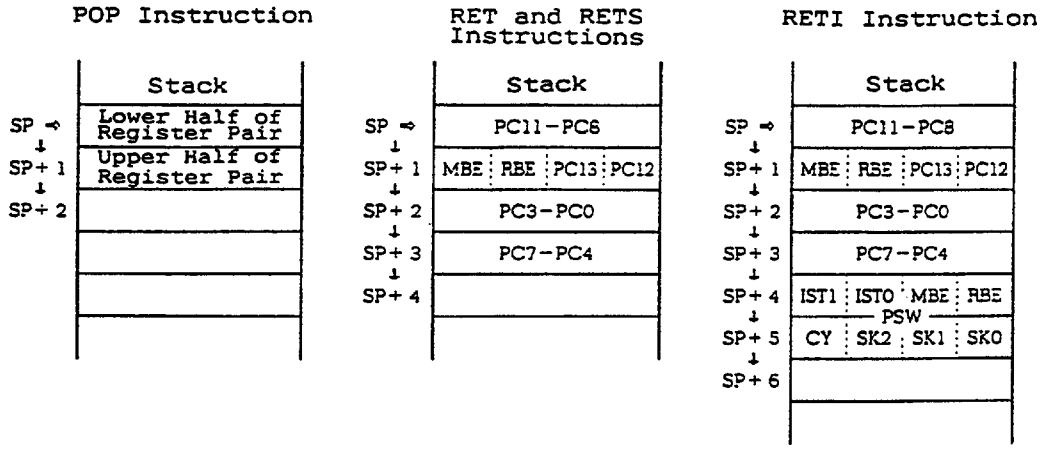


Figure 4-10 Data Restored from Stack Memory

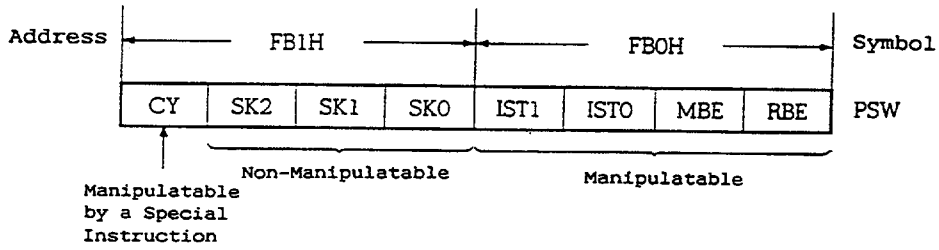


4.7 PROGRAM STATUS WORD (PSW) ... 8 BITS

The program status word (PSW) consists of various types of flags closely related to processor operation.

The PSW is mapped onto addresses FB0H and FB1H in the data memory space and the 4 bits at address FB0H can be manipulated by a memory manipulation instruction. Normal data memory manipulation instructions cannot be used on address FB1H.

Figure 4-11 Program Status Word Configuration



Part or all of the PSW is saved/restored to the stack memory when a subroutine call instruction or hardware interrupt is executed. PSW flags which are manipulated for stack operation are shown in Table 4-1.

Table 4-1 PSW Flags Saved/Restored in Stack Operation

		Flag Saved/Restored
Save	During CALL/CALLF instruction execution	MBE and RBE saved
	Upon hardware interruption	All PSW bits saved
Restore	During RET/RETS instruction execution	MBE and RBE restored
	During RETI instruction execution	All PSW bits restored

(1) Carry flag (CY)

The carry flag is a 1-bit flag used to store overflow or underflow generation information when a carry operation instruction (ADDC, SUBC) is executed.

It has a bit accumulator function for executing Boolean algebraic operations with the data memory specified by the bit address and storing the result.

Carry flag manipulation is carried out using a special instruction irrespective of other PSW bits.

When the $\overline{\text{RESET}}$ signal is generated, the carry flag becomes indeterminate.

Table 4-2 Carry Flag Manipulation Instructions

	Instruction (Mnemonic)	Carry Flag Operation and Processing
Carry flag manipulation special instruction	SET1 CY	CY set to "1"
	CLR1 CY	CY cleared to "0"
	NOT1 CY	CY contents inverted
	SKT CY	Skip if CY contents is "1"
Bit transfer instruction	MOV1 [*1] CY	CY contents transferred to the specified bit
	MOV1 CY, [*1]	Specified bit contents transferred to CY
Bit Boolean instruction	AND1 CY, [*1] OR1 CY, [*1] XOR1 CY, [*1]	Specified bit contents ANDed/ ORed/XORed with CY contents and the results set in CY
Interrupt servicing	During interrupt execution	Parallel save of other PSW bits and 8 bits to the stack memory
	RETI	Restore from the stack memory in parallel with other PSW bits

Remarks: [*1] indicates the following bit addressing operations.

. fmem.bit
. pmem.@L
. @H+mem.bit

Example: AND bit 3 at address 3FH with P33 and generate the result to P50.

```
MOV   H, #3H        ; set the most
                      significant 4-bit
                      address in the H
                      register
MOV1  CY, @H+0FH.3; CY ← 3FH bit 3
AND1  CY, PORT3.3 ; CY ← CY ∧ P33
MOV1  PORT5.0, CY ; P50 ← CY
```

(2) Skip flags (SK2, SK1, SK0)

These skip flags are used to store the skipped state and are automatically set/reset when the CPU executes an instruction.

The user cannot directly manipulate the skip flags as operands.

(3) Interrupt status flags (IST1, IST0)

The interrupt status flags are 2-bit flags used to store the status of the processing currently being executed.

Table 4-3 Interrupt Status Flag Specification Contents

IST1	IST0	Status of Processing being Executed	Processing Contents and Interrupt Control
0	0	Status 0	Normal program being executed. All interrupts acknowledgeable
0	1	Status 1	Low or high interrupt being executed. Only high interrupt acknowledgeable
1	0	Status 2	High interrupt being executed. All interrupts non-acknowledgeable
1	1	——	Setting prohibited

The interrupt priority control circuit (see Figure 6-1) identifies the flag contents and executes multiple interrupt control.

If an interrupt is acknowledged, the IST1 and IST0 contents are saved to the stack memory as part of the PSW and are automatically changed to the status higher by one level and the values prior to interruption by the RETI instruction are restored.

The interrupt status flag can be manipulated by a memory manipulation instruction and the processing status being executed can be changed by program control.

NOTE: Before manipulating the interrupt status flag, be sure to disable interruption by executing a DI instruction, and then enable interruption by executing an EI instruction after the manipulation.

(4) Memory bank enable flag (MBE)

This is a 1-bit flag used to specify the mode to generate the address information of the most significant 4 bits of the 12 bits of the data memory address.

The MBE can be set/reset at any time by a bit manipulation instruction irrespective of the memory bank setting.

When this flag is set to "1", the data memory address space is expanded depending on the memory bank select register (MBS) contents and the entire data memory space becomes addressable.

When this flag is reset to "0", the data memory address space is fixed irrespective of the MBS setting (see Figure 3-2).

When the $\overline{\text{RESET}}$ signal is generated, the bit 7 contents at address 0 of the program memory are set and the MBE flag is automatically initialized.

In vectored interrupt servicing, the bit 7 contents of the corresponding vector address table are set and the MBE status in interrupt servicing is automatically set.

Normally, set MBE = 0 for interrupt servicing and use the static RAM of memory bank 0.

(5) Register bank enable flag (RBE)

This is a 1-bit flag used to determine whether or not the general register bank configuration should be expanded.

RBE can be set/reset at any time by a bit manipulation instruction irrespective of the memory bank setting.

When this flag is set to "1", one general register can be selected from register banks 0 to 3 depending on the register bank select register (RBS) contents.

When this flag is reset to "0", register bank 0 is selected as a general register irrespective of the register bank select register (RBS) contents.

When the $\overline{\text{RESET}}$ signal is generated, the bit 6 contents at address 0 of the program memory are set and the RBE flag is automatically initialized.

When a vectored interrupt is generated, the bit 6 contents of the corresponding vector address table are set and the RBE status in interrupt servicing is automatically set. Normally, set RBE = 0 for interrupt servicing. Use register bank 0 for 4-bit operations and register banks 0 and 1 for 8-bit operations.

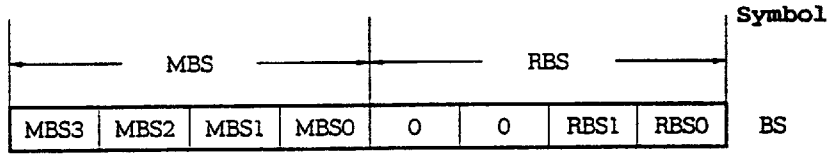
4.8 BANK SELECT REGISTER (BS)

The bank select register (BS) consists of a register bank select register (RBS) and a memory bank select register (MBS). RBS and MBS are used to specify the register bank and the memory bank to be used, respectively.

RBS and MBS are set by the SEL RBn and SEL MBn instructions, respectively.

BS can be saved/restored to the stack area as an 8-bit unit by the PUSH BS/POP BS instruction.

Figure 4-12 Bank Select Register Configuration



(1) Memory bank select register (MBS)

The memory bank select register is a 4-bit register used to store the most significant 4-bit address information of the data memory address (12 bits); the memory bank to be accessed is specified by the MBS contents. In the uPD75336, only banks 0, 1, 2 and 15 can be specified.

MBS is set by the SEL MBn instruction (n = 0, 1, 2, 15).

The address range for MBE and MBS setting is shown in Figure 3-2.

When the $\overline{\text{RESET}}$ signal is generated, MBS is initialized to "0".

(2) Register bank select register (RBS)

The register bank select register is used to specify the register bank for use as general registers and can set banks 0 to 3.

RBS is set by the SEL RBn instruction (n = 0 to 3).

When the $\overline{\text{RESET}}$ signal is generated, RBS is initialized to "0".

CHAPTER 5. PERIPHERAL HARDWARE FUNCTIONS

5.1 DIGITAL INPUT/OUTPUT PORTS

The uPD75336 employs memory mapped I/O and all input/output ports are mapped onto the data memory space.

Bit 0 at 1F8H to 1FFH functions as an output latch for port outputs 0 to 7(BP0 to BP7).

BP0 to BP7 are switched in 4-bit units by bits 6 and 7 of the display mode register (LCDM) between segment output and bit port output (refer to Figure 5-75 "Display Mode Register Format").

Bits at 1F8H to 1FFH which are not used as a bit port output latch can be used as display memory or static RAM. Each address can be operated bit-wise or in 4-bit units. 8-bit unit operation is not possible.

Figure 5-1 Digital Port Data Memory Addresses

Address	3	2	1	0	
FF0H	P03	P02	P01	P00	PORT 0
FF1H	P13	P12	P11	P10	PORT 1
FF2H	P23	P22	P21	P20	PORT 2
FF3H	P33	P32	P31	P30	PORT 3
FF4H	P43	P42	P41	P40	PORT 4
FF5H	P53	P52	P51	P50	PORT 5
FF6H	P63	P62	P61	P60	PORT 6
FF7H	P73	P72	P71	P70	PORT 7
FF8H	P83	P82	P81	P80	PORT 8
1F8H	—	—	—	BP0	
1F9H	—	—	—	BP1	
1FAH	—	—	—	BP2	
1FBH	—	—	—	BP3	
1FCH	—	—	—	BP4	
1FDH	—	—	—	BP5	
1FEH	—	—	—	BP6	
1FFH	—	—	—	BP7	

Remarks: Ports marked with —— can be used as data memory or display memory.

Input/output port operation instructions are listed in Table 5-1. As ports 4 to 7, a variety of control operations ranging from 4-bit input/output to 8-bit input/output and bit manipulations can be carried out.

BP0 to BP7 are bit-wise output ports.

Example 1: Test P13 status and generate different values at PORT4 and PORT5 depending on the test results.

```
SKT     PORT1,3   ; Skip if PORT1 bit 3 is 1.
MOV     XA, #18H  ; XA ← 18H
MOV     XA, #14H  ; XA ← 14H
SEL     MB15      ; Or CLR1 MBE
OUT     PORT4, XA; Ports 5 and 4 ← XA
```

2: SET1 PORT4.@L; Bit specified by L register at ports 4 to 7 is set to "1".

3: Generate 1 to BPO.

```
SET1    MBE
SEL     MB1;  Memory bank 1 is selected .
SET1    BPO;  BPO ← 1
```

5.1.1 TYPES, FEATURES AND CONFIGURATIONS OF DIGITAL INPUT/OUTPUT PORTS

Table 5-1 gives a list of digital input/output ports.

Figures 5-2 to 5-6 show port configurations.

Table 5-1 Types and Features of Digital I/O Ports

Port (Symbol)	Function	Operation/Feature	Remarks
PORT0	4-bit input	Reading or testing possible irrespective of dual-function pin operating mode	Also serves as a pin for INT4, $\overline{SCK}$, SO/SB0 and SI/SB1.
PORT1			Also serves as a pin for INT0 to INT2 and TI0.
PORT3*	4-bit input/output	Settable bit-wise to the input or output mode	Also serves as a pin for LCDCL and SYNC.
PORT6			Also serves as a pin for KR0 to KR3.
PORT2		Settable in 4-bit units to the input or output mode. Ports 6 and 7 can input/output 8-bit data as a pair.	Also serves as a pin for PTO0, PTO1, PCL and BUZ.
PORT7			Also serves as a pin for KR4 to KR7.
PORT8			Also serves as a pin for TI1, AN6 and AN7.
PORT4* PORT5*	4-bit input/output (N-ch open-drain with 10 V resistance)	Settable in 4-bit units to the input or output mode. Ports 4 and 5 can input/output 8-bit data as a pair.	On-chip pull-up resistor specifiable bit-wise by mask option
BP0 to BP7	1-bit output	Bit-wise data output. Switchable with LCD drive segment outputs S24 to S31 by software.	

*: Can directly drive LEDs.

P10 also serves as an external vectored interrupt input pin. It is an input with a noise eliminator (refer to Section 6.3 "Interrupt Control Circuit Hardware" for details).

BP0 to BP7 also serve as LCD drive segment outputs (S24 to S31) and are switched in 4- or 8-bit units by bits 6 and 7 of the display mode register (LCDM). BP0 to BP7 are bit-wise output ports which generate bit 0 data at addresses 1F8H to 1FFH of the display data memory (refer to Section 5.7.5 "Display Data Memory").

BP0 to BP7 have much smaller drive capabilities than other ports. Thus, they should be used to drive a CMOS load.

When $\overline{\text{RESET}}$ signal is generated, the output latch of ports 2 to 8 is cleared, the output buffer is turned OFF and the input mode is set. Refer to Table 2-1 for the BP0 to BP7 statuses in $\overline{\text{RESET}}$ input.

Figure 5-2 Configurations of Ports 0 and 1

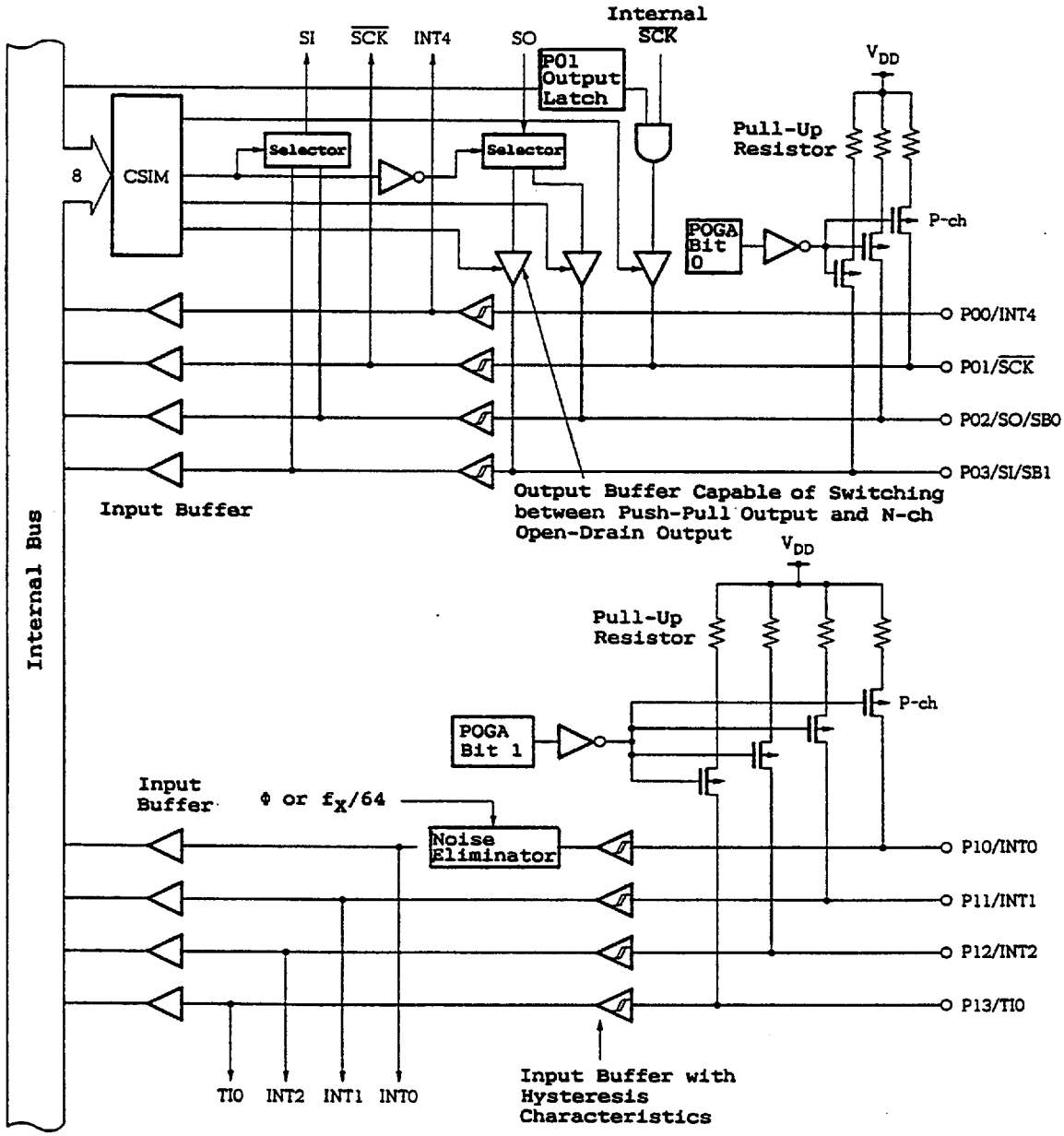


Figure 5-3 Configurations of Ports 3n and 6n (n = 0 to 3)

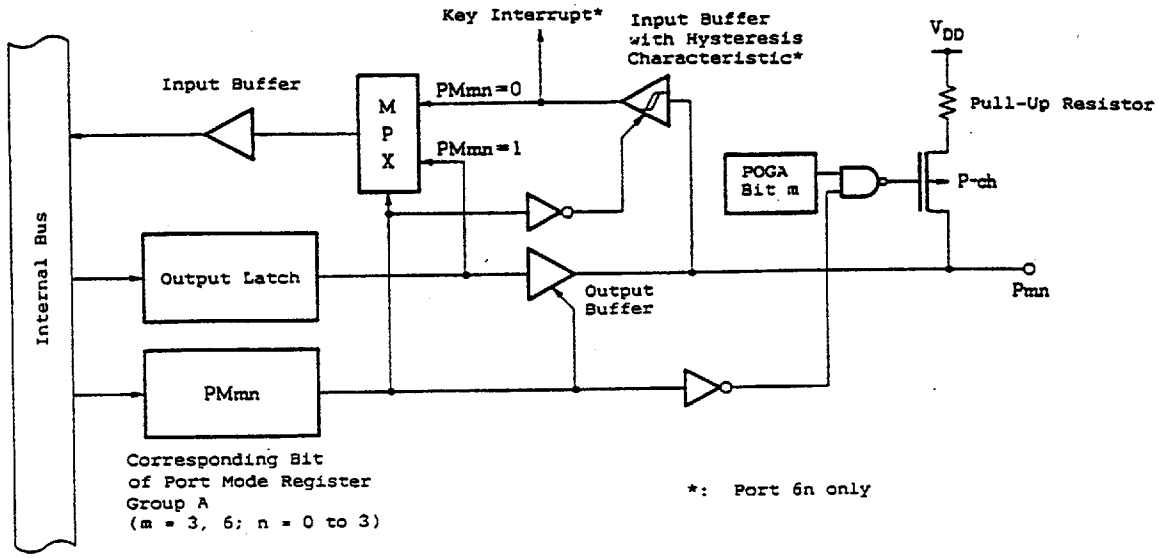


Figure 5-4 Configurations of Ports 2 and 7

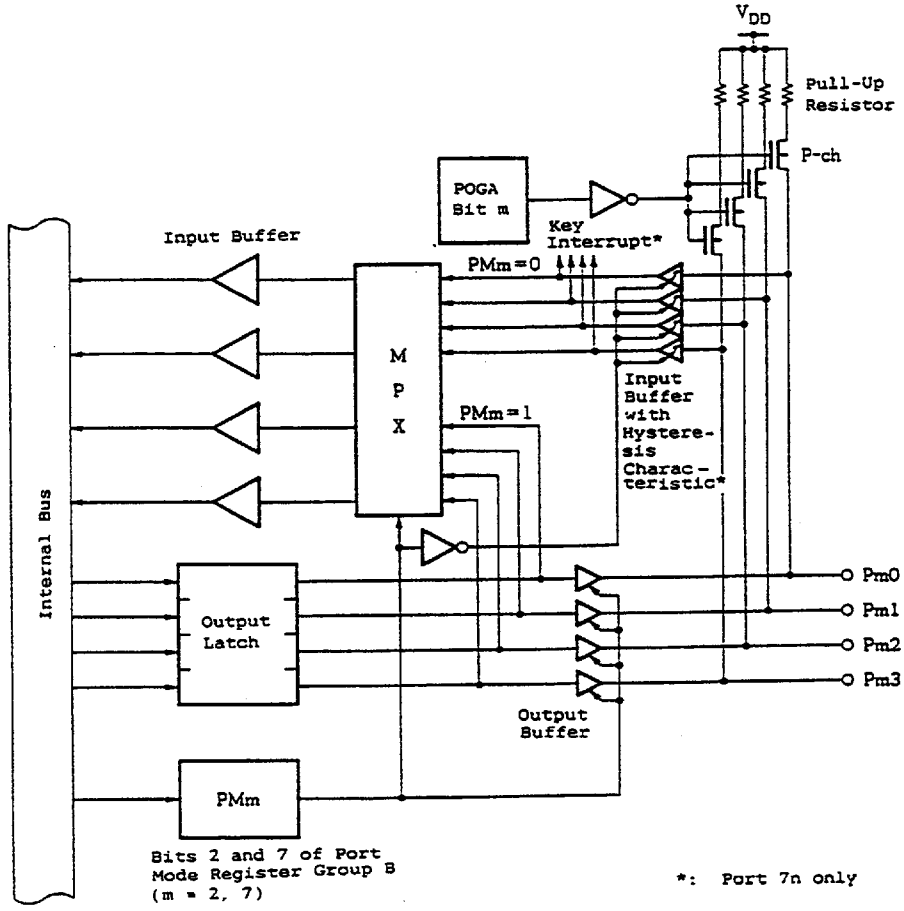


Figure 5-5 Configurations of Ports 4 and 5

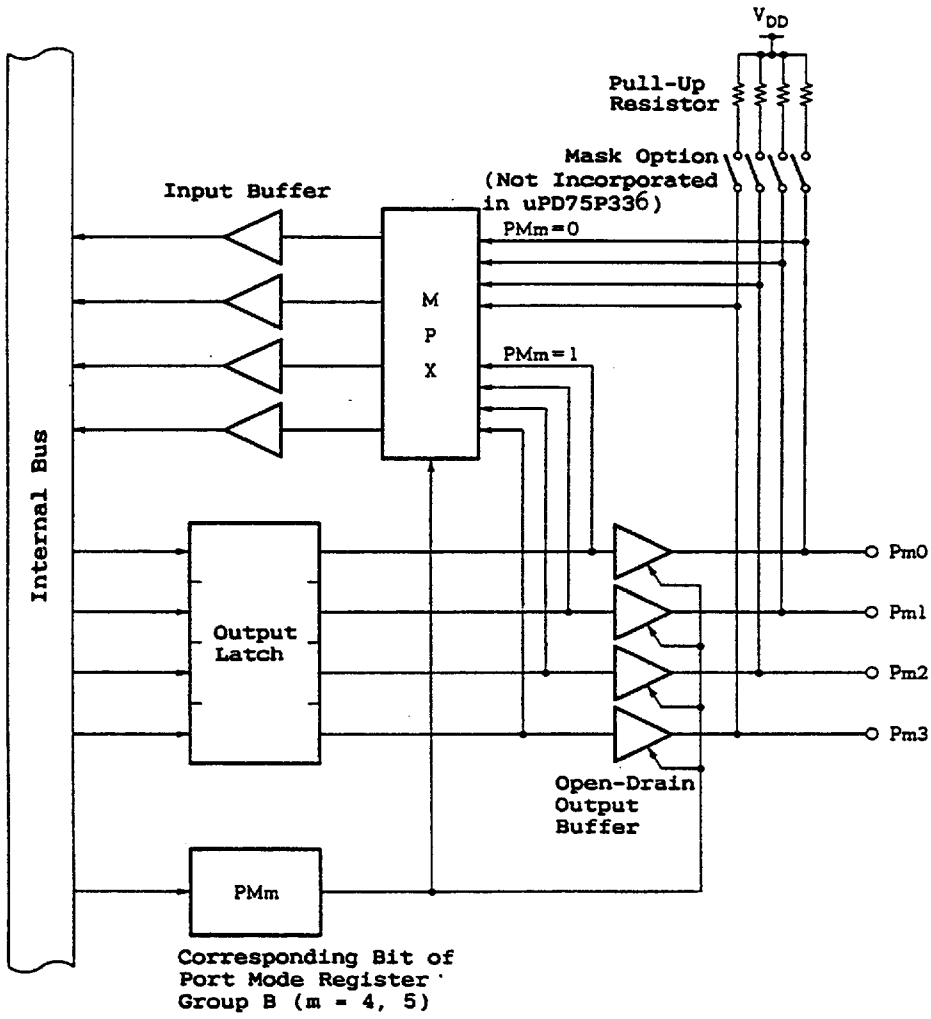
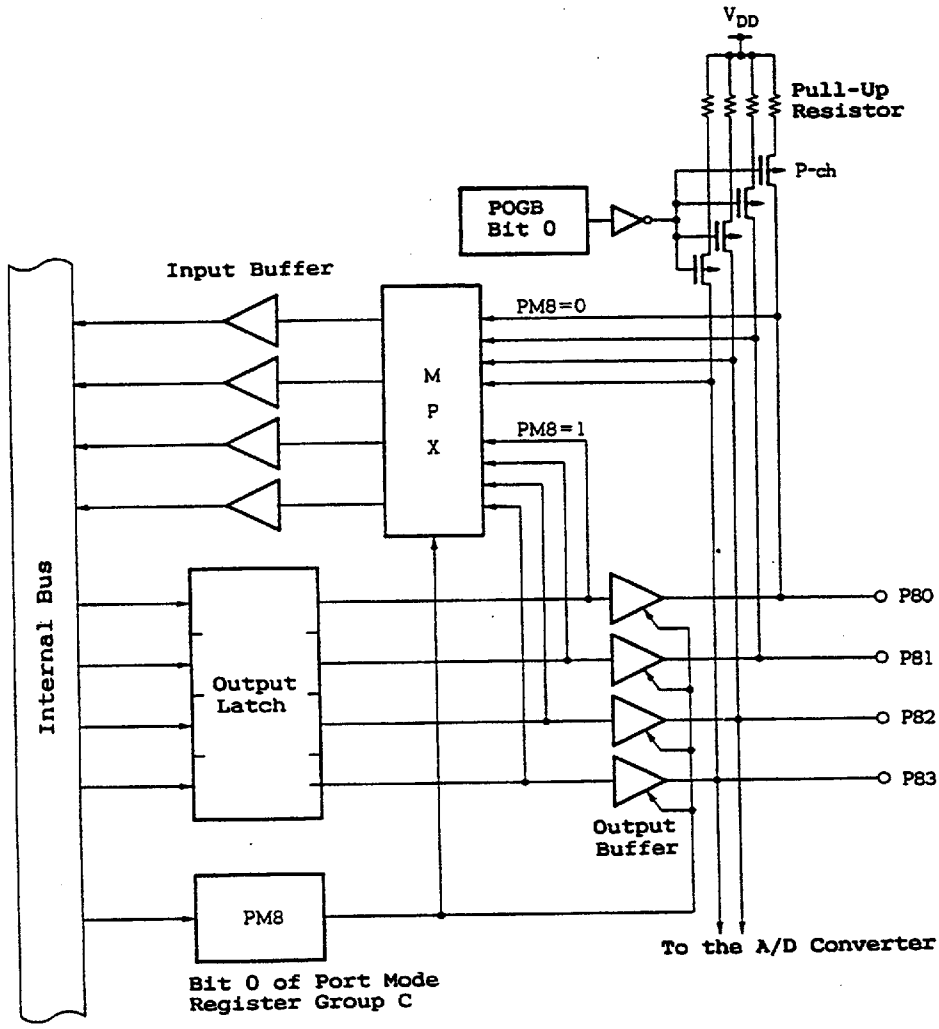


Figure 5-6 Port 8 Configuration



5.1.2 INPUT/OUTPUT MODE SETTING

The input/output mode of each input/output port is set by the port mode register as shown in Figure 5-7 on the next page. Input/output can be specified bit-wise for ports 3 and 6 by port mode register group A (PMGA). Input/output is specified in 4-bit units by the port mode register group B (PMGB) for ports 2, 4, 5 and 7 and by port mode register group (PMGC) for port 8.

When the bit of the corresponding port mode register is "0" or "1", each port functions as an input or output port, respectively.

When the output mode is selected by port mode register setting, the output latch contents are simultaneously output to the output pin. Thus, it is necessary to rewrite the output latch contents to the necessary value before setting the output mode.

Port mode register groups A, B and C each are set by an 8-bit memory manipulation instruction.

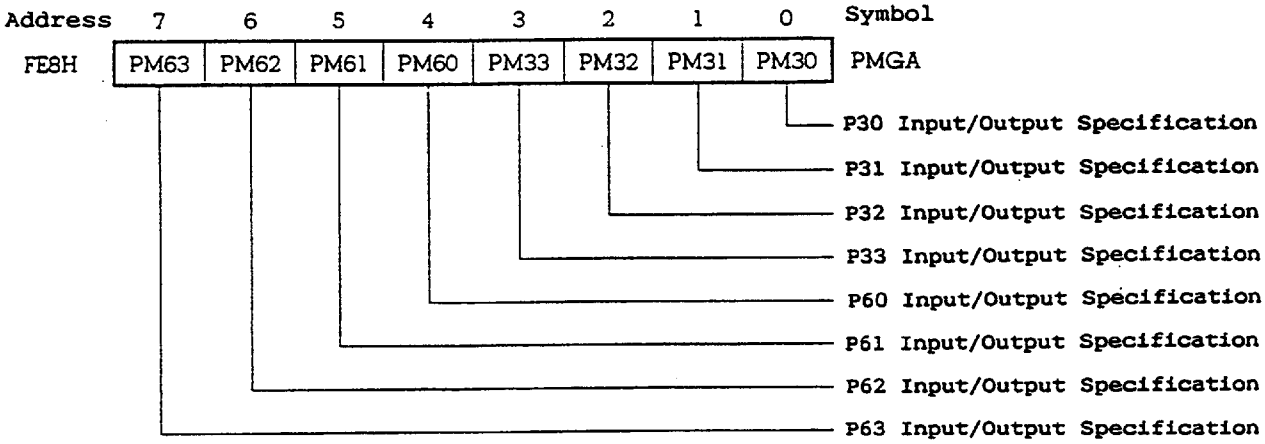
When the $\overline{\text{RESET}}$ signal is generated, all bits of each port mode register are cleared to "0". Thus, the output buffer is turned OFF and all ports are set to the input mode.

Example: Use P30, P31, P62 and P63 as input pins and P32, P33, P60 and P61 as output pins.

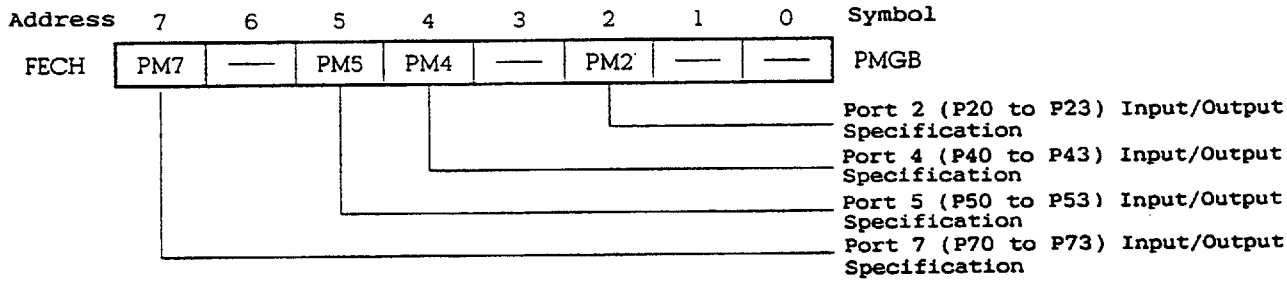
```
CLR1    MBE      ; Or SEL MB15
MOV     XA, #3CH
MOV     PMGA, XA
```

Figure 5-7 Port Mode Register Formats

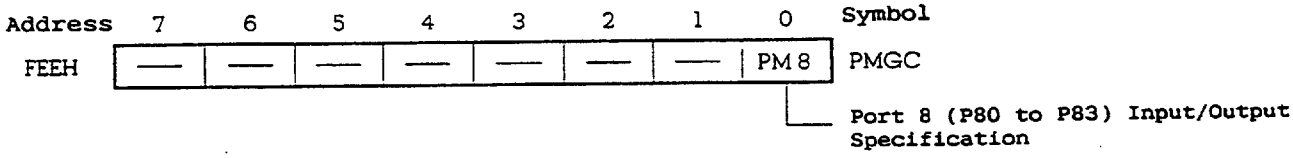
Port Mode Register Group A



Port Mode Register Group B



Port Mode Register Group C



	Specification Contents
0	Input mode (output buffer OFF)
1	Output mode (output buffer ON)

5.1.3 DIGITAL INPUT/OUTPUT PORT OPERATION INSTRUCTIONS

The on-chip input/output ports of the uPD75336 are all mapped onto the data memory space, and thus, all data memory manipulation instructions can be used. Table 5-2 lists the data memory manipulation instructions apparently effective for input/output pin operations and their application ranges.

(1) Bit manipulation instructions

Specific address bit direct addressing (fmem.bit) and specific address bit register indirect addressing (pmem.@L) can be used for digital input/output ports PORT0 to PORT8. Thus, port bit manipulation can be carried out freely irrespective of MBE and MBS settings.

Example: OR P50 with P41 and generate the result to P61.

```
        SET1  CY           ; CY ← 1
        AND1  CY, PORT5.0; CY ← CY ∧ P50
        OR1   CY, PORT4.1; CY ← CY ∨ P41
        SKT   CY
        BR    CLRP
        SET1  PORT6.1      ; P61 ← 1
         :
         :
CLRP:   CLR1  PORT6.1      ; P61 ← 0
```

(2) 4-bit manipulation instructions

In addition to IN/OUT instructions, all 4-bit memory manipulation instructions including MOV, XCH, ADDS and INCS can be used. It is necessary to select memory bank 15 prior to instruction execution.

Example 1: Generate accumulator contents to port 3.

```
SEL    MB15     ; Or CLR1 MBE
OUT    PORT3, A
```

2: Add the accumulator value to the data output to port 5 and generate the sum.

```
SET1   MBE
SEL    MB15
MOV    HL, #PORT5
ADDS   A, @HL     ; A ← A + PORT5
NOP
MOV    @HL, A     ; PORT5 ← A
```

3: Check if port 4 data is greater than the accumulator value.

```
SET1   MBE
SEL    MB15
MOV    HL, #PORT4
SUBS   A, @HL     ; A < PORT4
BR     NO         ; NO
                  : YES
```

(3) 8-bit manipulation instructions

IN/OUT, MOV, XCH and SKE instructions can be used for ports 4, 5, 6 and 7 which can perform 8-bit manipulation. As is the case with 4-bit manipulation, it is necessary to preselect memory bank 15.

Example: Generate BC register pair data to the output port specified by 8-bit data input from ports 4 and 5.

```
SET1  MBE
SEL   MB15
IN    XA, PORT4; XA ← Ports 5 and 4
MOV   HL, XA   ; HL ← XA
MOV   XA, BC   ; XA ← BC
MOV   @HL, XA  ; Port (L) ← XA
```

Table 5-2 Table of Input/Output Pin Operation Instructions

Instruction \ PORT	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PORT 5	PORT 6	PORT 7	PORT 8	Bit Port 0 to 7
IN A, PORTn *1	o									MOV A, mem *3, *4
IN XA, PORTn *1	—		—		o		o		-	—
OUT PORTn, A *1	-	-	o							MOV mem, A *3, *4
OUT PORTn, XA *1	—		—		o		o		-	—
SET1 PORTn.bit	-	-	o							SET1 BPn *3
SET1 PORTn.@L *2	-	-	o							—
CLR1 PORTn.bit	-	-	o							CLR1 BPn *3
CLR1 PORTn.@L *2	-	-	o							—
SKT PORTn.bit	o									SKT BPn *3
SKT PORTn.@L *2	o									—
SKF PORTn.bit	o									SKF BPn *3
SKF PORTn.@L *2	o									—
MOV1 CY, PORTn.bit	o									—
MOV1 CY, PORTn.@L *2	o									—
MOV1 PORTn.bit, CY	-	-	o							—

(to be continued)

Table 5-2 Table of Input/Output Pin Operation Instructions (cont'd)

Instruction \ PORT	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PORT 5	PORT 6	PORT 7	PORT 8	Bit Port 0 to 7
MOV1 PORTn.@L, CY *2	-	-	o							—
AND1 CY, PORTn.bit	o									AND1 CY, @H + BPn *3, *5
AND1 CY, PORTn.@L *2	o									—
OR1 CY, PORTn.bit	o									OR1 CY, @H + BPn *3, *5
OR1 CY, PORTn.@L *2	o									—
XOR1 CY, PORTn.bit	o									XOR1 CY, @H + BPn *3, *5
XOR1 CY, PORTn.@L *2	o									—

*1: Preset MBE = 0 or (MBE = 1, MBS = 15) before execution.

2: Specify the least significant 2 bits of the address and the bit address indirectly using the L register.

3: Preset (MBE = 1, MBS = 1) before execution.

4: Bit n of accumulator A corresponds to BPn.

5: Write FH to H register.

5.1.4 DIGITAL INPUT/OUTPUT PORT OPERATIONS

When a data memory manipulation instruction is executed for a digital input/output port, port and pin operations differ depending on the input/output mode setting (refer to Table 5-3).
As is clear from the input/output port configuration, this is because the data fetched into the internal bus becomes individual pin data in the input mode and output latch data in the output mode.

(1) Operations in the input mode

- Pin data is input when a test instruction such as the SKT instruction, a bit input instruction by MOV1 instruction or an instruction which fetches port data into the internal bus in 4- or 8-bit units (IN, OUT, operation and compare instructions) is executed.

- Accumulator data is latched into the output latch when an instruction which transfers accumulator contents to the port in 4- or 8-bit units (OUT and MOV instructions) is executed. The output buffer remains OFF.

- When the XCH instruction is executed, individual pin data is input to the accumulator and accumulator data is latched into the output latch. The output buffer remains OFF.

- When the INCS instruction is executed, individual pin data (4 bits) plus one is latched into the output latch. The output buffer remains OFF.

. When an instruction, such as the SET1, CLR1, MOV1 or SKTCLR instruction, which rewrites the data memory bit by bit is executed, the output latch of the specified bit can be rewritten as specified by the instruction. In this case, the output latch contents of the other bits become indeterminate.

(2) Operations in the output mode

. Output latch contents are input when a test instruction, a bit input instruction or an instruction which fetches port data into the internal bus in 4- or 8-bit units is executed.

. When an instruction to transfer accumulator contents in 4- or 8-bit units is executed, output latch data is rewritten and is simultaneously output from the pin.

. When the XCH instruction is executed, output latch contents are transferred to the accumulator, and the accumulator contents are latched into the output latch and are output from the pin.

. When the INCS instruction is executed, output latch contents plus one are latched into the output latch and are output from the pin.

. When a bit output instruction is executed, the bit of the specified output latch is rewritten and is output from the pin.

Table 5-3 Input/Output Port Operations

Instruction to be Executed	Port and Pin Operations	
	Input Mode	Output Mode
SKT [*] SKF [*]	Pin data test	Output latch data test
MOV1 CY, [*]	Pin data transfer to CY	Output latch data transfer to CY
AND1 CY, [*] OR1 CY, [*] XOR1 CY, [*]	Operation between pin data and CY	Operation between output latch data and CY

IN A, PORTn
IN XA, PORTn
MOV A, @HL
MOV XA, @HL
	Pin data transfer to accumulator	Output latch data transfer to accumulator

ADDS A, @HL
ADDC A, @HL
SUBS A, @HL
SUBC A, @HL
AND A, @HL
OR A, @HL
XOR A, @HL
	Operation between pin data and accumulator	Operation between output latch data and accumulator

SKE A, @HL SKE XA, @HL	Pin data and accumulator compare	Output latch data and accumulator compare

OUT PORTn, A
OUT PORTn, XA
MOV @HL, A
MOV @HL, XA
	Accumulator data transfer to output latch (with output buffer set to OFF)	Accumulator data transfer to output latch and output from pin

(to be continued)

Table 5-3 Input/Output Port Operations (cont'd)

Instruction to be Executed	Port and Pin Operations	
	Input Mode	Output Mode
XCH A, PORTn XCH XA, PORTn XCH A, @HL XCH XA, @HL	Pin data transfer to accumulator and accumulator data transfer to output latch (with output buffer set to OFF)	Data exchange between output latch and accumulator
INCS PORTn INCS @HL	Latch of pin data plus one to output latch	Output latch contents plus one
SET1 [*] CLR1 [*] MOV1 [*], CY SKTCLR [*]	Specified bit output latch rewritten as specified by instruction but all other bit output latches remain indeterminate	Output pin status change by instruction

*: Indicates two addressing modes, PORTn.bit and PORTn.@L.

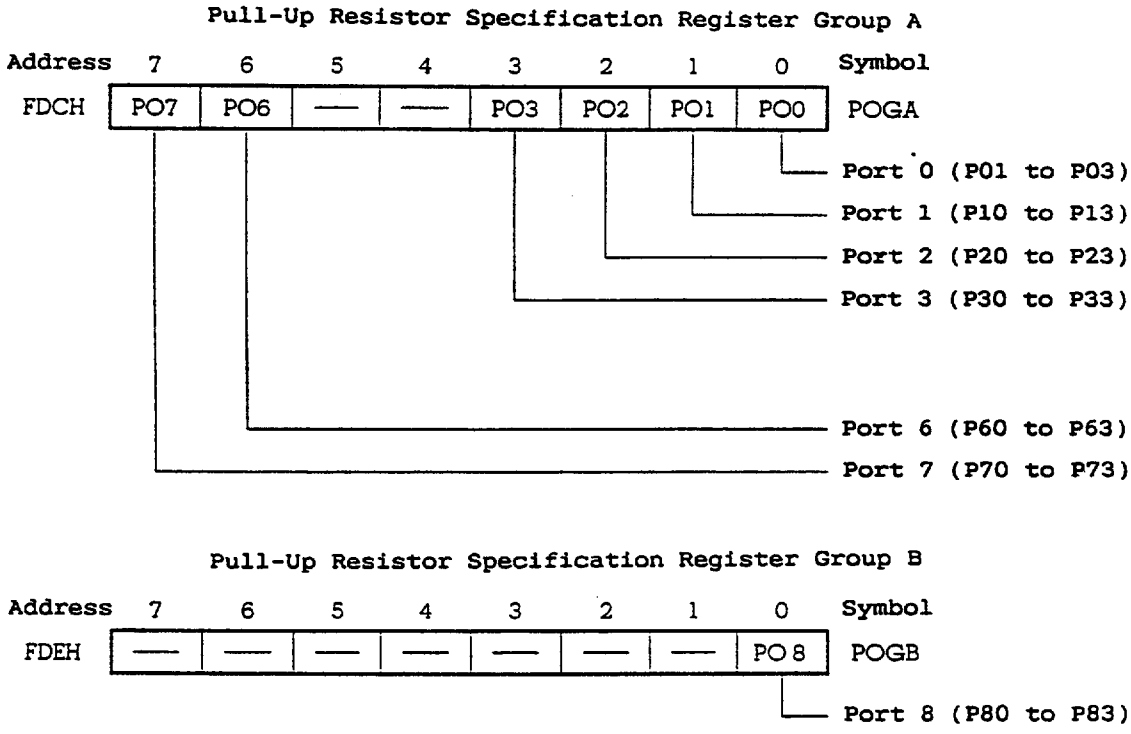
5.1.5 ON-CHIP PULL-UP RESISTOR

A pull-up resistor can be incorporated in each port of the uPD75336 (except P00 and BP0 to BP7). Pull-up resistor incorporation in the pin can be specified by software or mask option.

Pull-up register incorporation is specified by pull-up register specification register group A(POGA) and B(POGB).

On-chip pull-up resistor specification for ports 3 and 6 is only valid for pins specified for the input mode. The pins specified for the output mode have no on-chip pull-up resistors irrespective of POGA setting.

Figure 5-8 Pull-Up Resistor Specification Register Format



	Specification Contents
0	Pull-up resistor not incorporated.
1	Pull-up resistor incorporated.

NOTE: The uPD75P336 does not incorporate pull-up resistors by mask option.

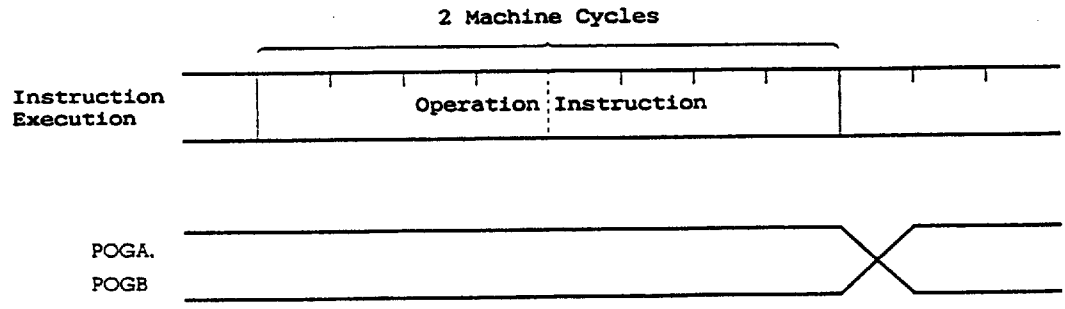
Table 5-4 On-Chip Pull-Up Resistor Specification Method

Port (Pin Name)	Specification Method	Special Bit
Port 0 (P01 to P03) *1	On-chip resistor specification by software in 3-bit units	POGA.0
Port 1 (P10 to P13)	On-chip resistor specification by software in 4-bit units	POGA.1
Port 2 (P20 to P23)		POGA.2
Port 3 (P30 to P33)		POGA.3
Port 6 (P60 to P63)		POGA.6
Port 7 (P70 to P73)		POGA.7
Port 8 (P80 to P83)		POGB.0
Port 4 (P40 to P43)	On-chip resistor specification by mask option in 1-bit units *2	——
Port 5 (P50 to P53)		

*1: No pull-up resistor can be incorporated in the P00 pin.

2: The uPD75P336 can incorporate no pull-up resistor in ports 4 and 5.

Pull-up resistor usage is switched by the pull-up resistor specification register (POGA, POGB) setting using the following timing.



After the on-chip pull-up resistor has been specified by rewriting POGA and POGB, execute an NOP instruction and an input/output instruction in that order, taking into account the external load capacity.

Example: Enter after specifying on-chip pull-up resistor for port 1.

```
SEL   MB15
MOV   XA, #02H; Pull-up resistor incorporated in
                port 1
MOV   POGA, XA  ┐
 :              │ Wait until stabilization
 :              │ considering external load
IN    A, PORT1  ┘ capacity.
```

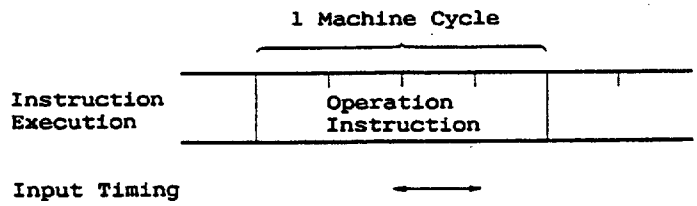
5.1.6 DIGITAL INPUT/OUTPUT PORT INPUT/OUTPUT TIMINGS

Data is output to the output latch and pin data or output latch data is fetched into the internal bus using the timings shown in Figure 5-9.

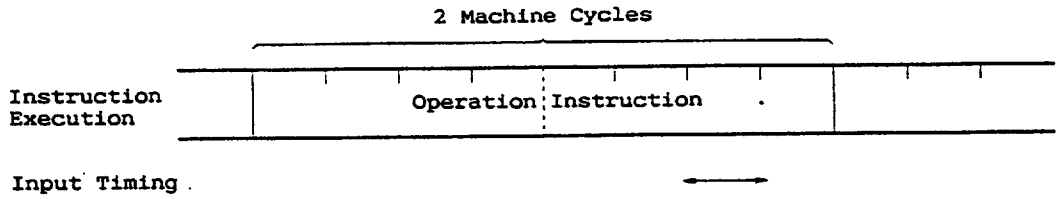
ON timings when on-chip pull-up resistor is specified by software are shown in Figure 5-10.

Figure 5-9 Digital Input/Output Port Input/Output Timings

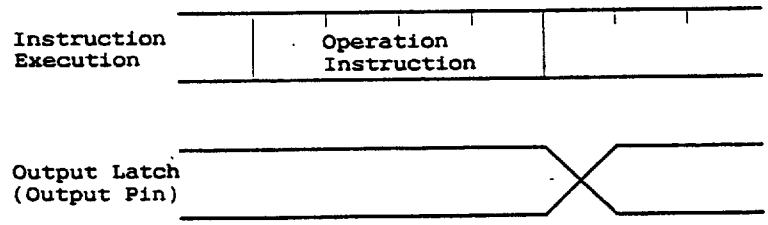
(a) When data is fetched by 1-machine cycle instruction



(b) When data is fetched by 2-machine cycle instruction



(c) When data is latched by 1-machine cycle instruction



(d) When data is latched by 2-machine cycle instruction

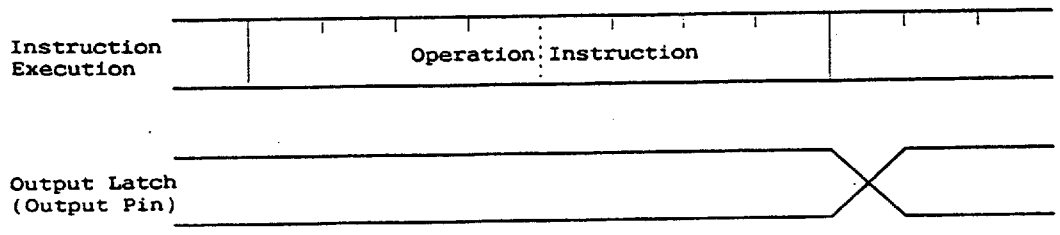
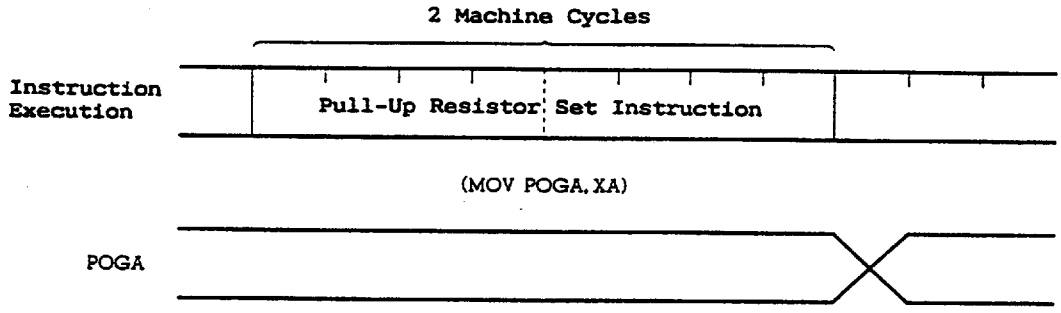


Figure 5-10 Pull-Up Resistor ON Timing by Software



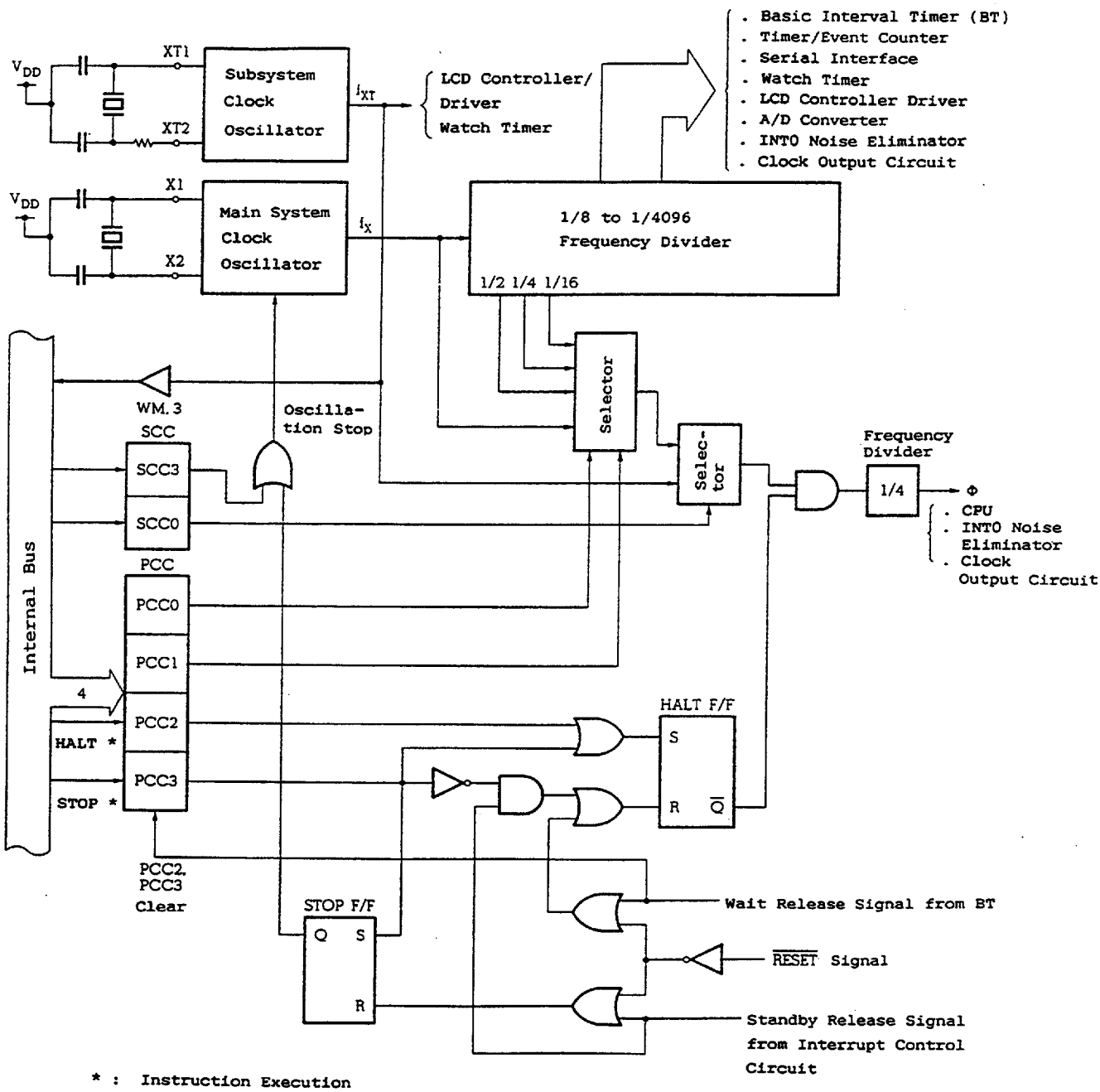
5.2 CLOCK GENERATOR

The clock generator supplies the CPU and peripheral hardware with various types of clocks and controls the CPU operation mode.

5.2.1 CLOCK GENERATOR CONFIGURATION

The clock generator is configured as shown in Figure 5-11.

Figure 5-11 Clock Generator Block Diagram



* : Instruction Execution

Remarks 1: f_X = Main-system clock frequency

2: f_{XT} = Subsystem clock frequency

3: Φ = CPU clock

4: PCC: Processor clock control register

5: SCC: System clock control register

6: 1 clock cycle (t_{CY}) of Φ is 1 machine cycle of an instruction.

5.2.2 CLOCK GENERATOR FUNCTIONS AND OPERATIONS

The clock generator generates the following clocks and controls the CPU operating modes including standby mode.

- Main-system clock f_X
- Subsystem clock f_{XT}
- CPU clock Φ
- Clock to peripheral hardware

The following clock generator operations are determined by the processor clock control register (PCC) and the system clock control register (SCC):

(a) When the $\overline{\text{RESET}}$ signal is generated, the lowest speed mode (15.3 us/4.19 MHz) of the main-system clock is selected (PCC = 0, SCC = 0).

(b) One of the four-level CPU clocks can be selected by setting the PCC with the main-system clock selected (0.95 us, 1.91 us, 3.81 us and 15.3 us/4.19 MHz).

(c) Two standby modes, the STOP and HALT modes, are available with the main-system clock selected.

(d) The clock generator can be operated at a super low-speed with low power consumption (122 us/32.768 kHz) by selecting the subsystem clock with SCC. In this case, the PCC set value has no effect on the CPU clock.

(e) Main-system clock oscillation can be stopped by SCC with the subsystem clock selected. The HALT mode can also be used but the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.)

(f) Divided main-system clocks are supplied to the peripheral hardware. Subsystem clocks can be directly supplied to the watch timer. Therefore, the clock function, the LCD control function executed by the clock from the watch timer, and the buzzer output function can be continued even in the standby status.

(g) When a subsystem clock is selected as the count clock of the watch timer, the watch timer and the LCD controller can continue to operate normally. However, because the other hardware devices operate with main-system clock, they cannot be used if the main-system clock is stopped.

(1) Processor clock control register (PCC)

The PCC is a 4-bit register used to select the CPU clock Φ with the least significant 2 bits and to control the CPU operating mode with the most significant 2 bits. Figure 5-12 on the next page shows the PCC format.

When bit 3 or 2 is set to "1", the standby mode is set. If the standby mode is released by the standby release signal, both bits are automatically cleared and the normal operating mode is set. (For details, refer to Chapter 7. "Standby Functions".)

The least significant 2 bits of the PCC are set by a 4-bit memory manipulation instruction (with the most significant 2 bits set to "0").

Bits 3 and 2 are set to "1" by the STOP and HALT instructions, respectively.

The STOP and HALT instructions can always by executed irrespective of the MBE contents.

The CPU clock can only be selected during operation with the main-system clock. When operating with the subsystem clock, the least significant 2 bits of the PCC are invalidated and $f_{XT}/4$ is fixed. The STOP instruction is also enabled only during operation with the main-system clock.

Example 1: Set the machine cycle to 0.95 us (f_X = 4.19 MHz).

```
SEL   MB15
MOV   A, #0011B
MOV   PCC,A
```

2: Set the machine cycle to 1.63 us (f_X = 4.91 MHz).

```
SEL   MB15
MOV   A, #0010B
MOV   PCC,A
```

3: Set the STOP mode (be sure to write NOP instruction after STOP and HALT instructions).

```
STOP
NOP
```

When the $\overline{\text{RESET}}$ signal is generated, the PCC is cleared to "0".

Figure 5-12 Processor Clock Control Register Format

Address: FB3H

Symbol: PCC

3	2	1	0
PCC3	PCC2	PCC1	PCC0

CPU Clock Select Bit
(When $f_X \leq 4.19$ MHz)

		SCC0 = 0 Values with f_X = 4.19 MHz are indicated in parentheses		SCC0 = 1 Values with f_{XT} = 32.768 kHz are indicated in parentheses	
		CPU Clock Frequency	1 Machine Cycle	CPU Clock Frequency	1 Machine Cycle
0	0	$\Phi = f_X/64$ (65.5 kHz)	15.3 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
0	1	$\Phi = f_X/16$ (262 kHz)	3.82 μs	Setting prohibited	
1	0	$\Phi = f_X/8$ (524 kHz)	1.91 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
1	1	$\Phi = f_X/4$ (1.05 MHz)	0.95 μs		

(When 4.19 MHz < $f_X \leq$ 5.0 MHz)

		SCC0 = 0 Values with f_X = 4.91 MHz are indicated in parentheses		SCC0 = 1 Values with f_{XT} = 32.768 kHz are indicated in parentheses	
		CPU Clock Frequency	1 Machine Cycle	CPU Clock Frequency	1 Machine Cycle
0	0	$\Phi = f_X/64$ (76.7 kHz)	13 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
0	1	$\Phi = f_X/16$ (307 kHz)	3.26 μs	Setting prohibited	
1	0	$\Phi = f_X/8$ (614 kHz)	1.63 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
1	1	Setting prohibited		Setting prohibited	

Remarks 1: f_X : Main-system clock oscillator output frequency

2: f_{XT}: Subsystem clock oscillator output frequency

CPU Operating Mode Control Bit

0	0	Normal operating mode
0	1	HALT mode
1	0	STOP mode
1	1	Setting prohibited

NOTE: When using a value of f_X such that 4.19 MHz < $f_X \leq$ 5 MHz, if the maximum speed: $\Phi = f_X/4$ (PCC1, PCC0 = 11) is set as the CPU clock frequency, 1 machine cycle becomes less than 0.95 us, with the result that the specified MIN value of 0.95 cannot be observed. Therefore, in this case PCC1, PCC0 = 11 cannot be set. Use PCC1, PCC0 = 10 or 01 or 00. As a result, the combination of f_X = 4.19 MHz and PCC1, PCC0 = 11 is the selected maximum CPU clock speed (1 machine cycle = 0.95 us).

(2) System clock control register (SCC)

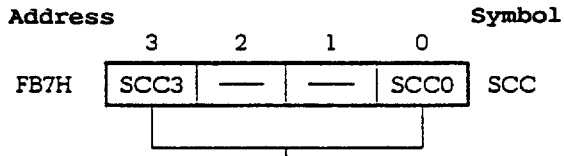
The SCC is a 4-bit registered used to select the CPU clock Φ with the least significant bit and to control main system clock oscillation stop page with the most significant bit. Figure 5-13 shows the SCC format.

Although bits 3 and 0 of SCC are located at the same data memory address, both bits cannot be changed simultaneously. Thus, these two bits of SCC are set by a bit manipulation instruction. Bits 3 and 0 of SCC can always be bit-manipulated irrespective of the MBE contents.

Main-system clock oscillation can be stopped by setting SCC bit 3 only when in operation with the subsystem clock. Oscillation when in operation with the main-system clock is stopped by the STOP instruction.

When the $\overline{\text{RESET}}$ signal is generated, the SCC is cleared to "0".

Figure 5-13 System Clock Control Register Format



SCC3	SCC0	CPU Clock Selection	Main-System Clock Oscillation
0	0	Main-system clock	Oscillation enabled
0	1	Subsystem clock	
1	0	Setting prohibited	
1	1	Subsystem clock	Oscillation stopped

NOTE 1: A maximum of $1/f_{XT}$ is required to change the system clock. Thus, when stopping the main-system clock oscillation, set SCC bit 3 following the elapse of more than the number of machine cycles shown in Table 5-5 after the subsystem clock has been changed to the subsystem clock.

2: If oscillation is stopped by setting SCC bit 3 while in operation with the main-system clock, the normal STOP mode is not set.

3: When "1" is set in SCC bit 3, X1 input is internally short-circuited to V_{SS} (GND potential) to prevent leakage from the crystal oscillator unit. Thus, when using an external clock for the main-system clock, do not set "1" in SCC bit 3.

4: When PCC = 0001B (with $\Phi = f_X/16$ selected), do not set "1" in SCC bit 0 switching from the main system clock to subsystem clock should be performed after setting PCC to the other state (PCC $\neq$ 0001B). Do not perform setting of PCC = 0001B when operating with subsystem clock.

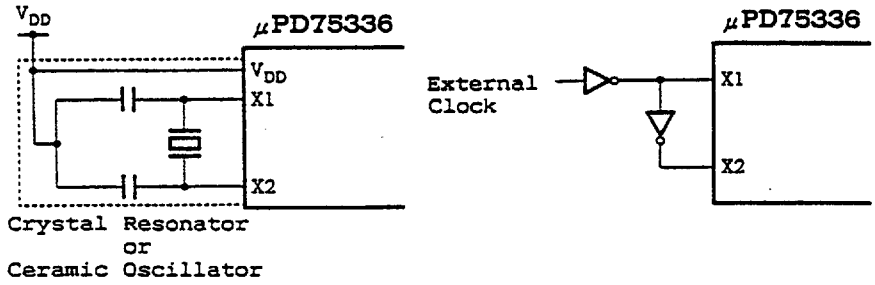
(3) System clock oscillator

(i) The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator connected to the X1 and X2 pins (with a standard frequency of 4.194304 MHz).

External clocks can be input to this oscillator. In this case, apply the clock signal to the X1 pin and the inverted clock signal to the X2 pin.

Figure 5-14 External Circuit of Main-System Clock Oscillator

(a) Crystal/ceramic oscillation (b) External clock



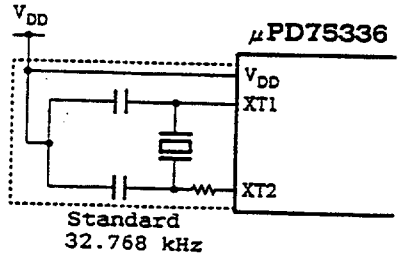
NOTE: When an external clock is input, STOP mode cannot be set. This is because the X1 pin is short-circuited to V_{SS} in the STOP Mode.

(ii) The subsystem clock oscillator oscillates with a crystal resonator (with a standard frequency of 32.768 kHz) connected to the XT1 and XT2 pins. External clock can be input to this oscillator. In this case, apply the clock signal to the XT1 pin and the inverted signal to the XT2 pin.

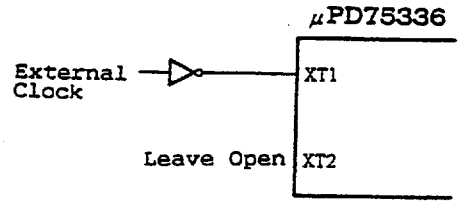
The XT1 pin status can be tested by bit 3 of the clock mode register (WM).

Figure 5-15 External Circuit of Subsystem Clock Oscillator

(a) Crystal oscillation



(b) External clock



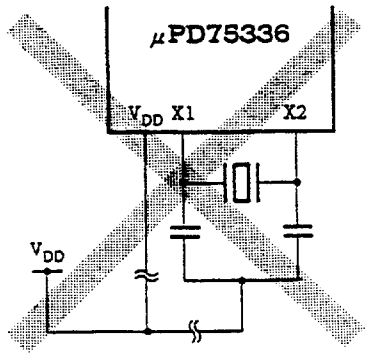
NOTE 1: When main system clock and subsystem clock oscillator are used, the shaded area in Figures 5-14 and 5-15 should be wired in order to avoid effects of wiring capacitance etc., as shown below.

- Minimize the length of wiring.
- Do not cross other signal lines, or position wiring close to a variable high current.
- The grounding point of the oscillator capacitor should always be of the same potential as V_{DD}. Do not connect it to the supply pattern where there is a high current.
- Do not pick up the signal form the oscillator.

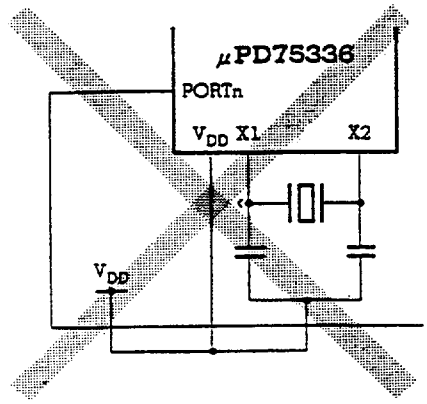
The subsystem clock oscillator is designed to be a low amplification circuit which consumes only a low current and misoperation caused by noise may occur more often than with the main system clock oscillator.
Therefore, when using the subsystem clock, extreme care is required for the wiring procedure.
Figure 5-16 shows bad examples of resonator connection circuits.

Figure 5-16 Bad Examples of Resonator Connection Circuit

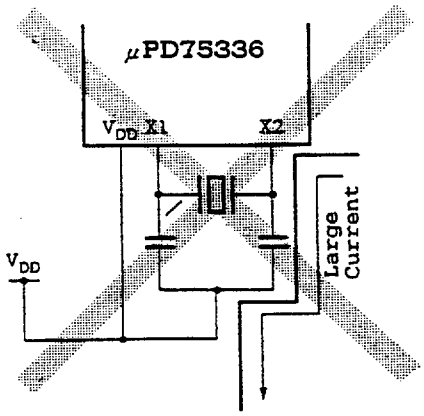
(a) Connection circuit is too long.



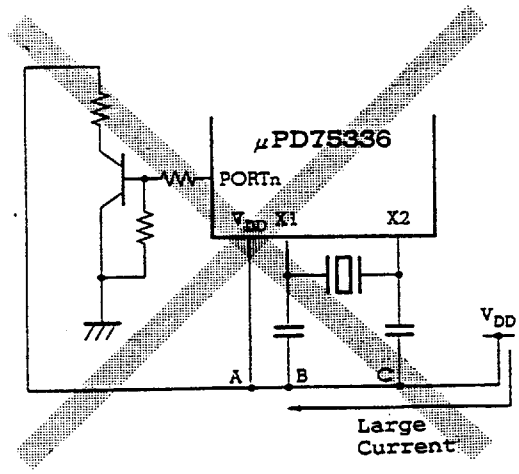
(b) Connection circuit crosses a signal line.



(c) A line carrying varying large current is running near the connection circlet.



(d) Current is flowing in the grounding line of the oscillation circuit. (The electric potential at points A, B and C fluctuates.)

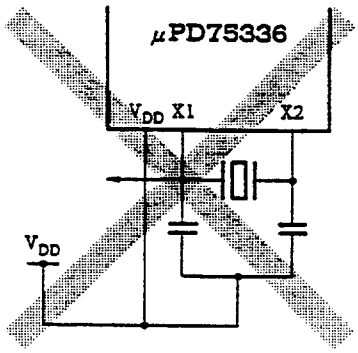


Remarks 1: When the subsystem clock is used, pins X1 and X2 in the figures above should be read as XT1 and XT2, respectively. Insert also a resistor serially on the XT2 side.

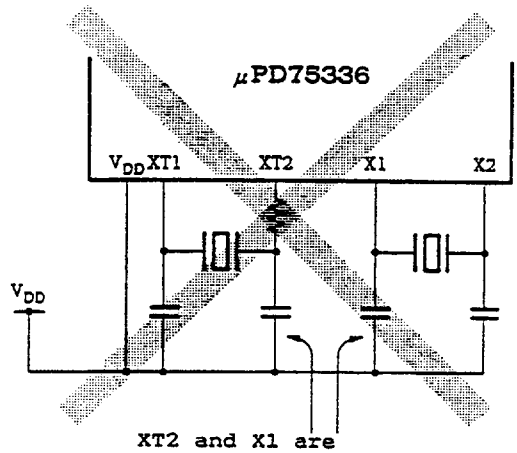
Remarks 2: Pin arrangements in the figures above are for illustration purposes only and may differ from those for the uPD75336.

Figure 5-16 Bad Examples of Resonator Connection Circuit (cont'd)

(e) Signal is being taken out of the connection circuit.



(f) The main system clock signal line is placed adjacent to and parallel with the subsystem signal line.

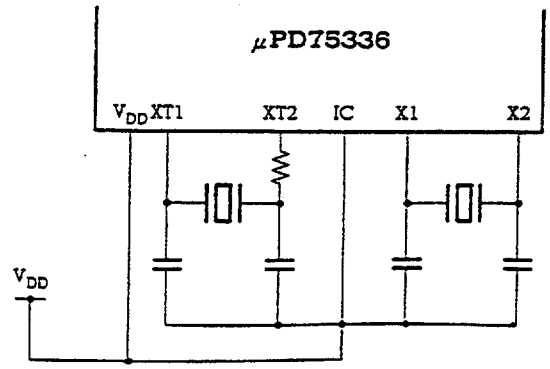


XT2 and X1 are arranged in parallel. (See NOTE 2 for an example of correction.)

Remarks 1: When the subsystem clock is used, pins X1 and X2 in the figures above should be read as XT1 and XT2, respectively. Insert also a resistor serially on the XT2 side.

2: Pin arrangements in the figures above are for illustration purposes only and may differ from those for the uPD75336.

NOTE 2: In Figure 5-16 (f), XT2 is arranged in parallel with X1. This may lead to erroneous operation due to the X1 crosstalk noise multiplied by XT2 crosstalk noise.
To prevent erroneous operation, X1 and XT2 should not be arranged in parallel, and the NC pin located between the two should be connected to V_{DD}.



(4) Frequency divider

The frequency divider divides the main-system clock oscillator output (f_X) and generates various clocks.

(5) When subsystem clock is not used

When it is not necessary to use a subsystem clock because of low power consumption operation, clock operation, etc. handle the XT1 and XT2 pins as follows:

XT1: Connect to V_{SS} or V_{DD}
XT2: Leave open

In this state, however, some leakage current may result via an internal feedback resistor of the subsystem clock oscillator. To minimize the leakage current, the internal feedback resistor can be removed by a mask option. In this case also, handle the XT1 and XT2 pins as described above (be sure to specify the mask option when ordering).

5.2.3 SYSTEM CLOCK AND CPU CLOCK SETTINGS

(1) Time required for system clock and CPU clock switching

The system clock and the CPU clock can be mutually switched with the least significant 2 bits of the SCC and the least significant bit of the SCC. This switching is not executed just after register rewriting and operation continues with the previous clock during the specified number of machine cycles. Thus, to stop main system clock oscillation, it is necessary to execute a STOP instruction or to set SCC bit 3 to "1" after the specified switching time.

Table 5-5 Maximum Time Required for System Clock and CPU Clock Switching

Set Value before Switching			Set Value after Switching														
SCC 0	PCC 1	PCC 0	SCC 0	PCC 1	PCC 0	SCC 0	PCC 1	PCC 0	SCC 0	PCC 1	PCC 0	SCC 0	PCC 1	PCC 0	SCC 0	PCC 1	PCC 0
			0	0	0	0	0	1	0	1	0	0	1	1	1	x	x
0	0	0				1 machine cycle			1 machine cycle			1 machine cycle			$f_X/64f_{XT}$ machine cycles (2 machine cycles)		
	0	1	4 machine cycles						4 machine cycles			4 machine cycles			Setting prohibited		
	1	0	8 machine cycles			8 machine cycles						8 machine cycles			$f_X/8f_{XT}$ machine cycles (16 machine cycles)		
	1	1	16 machine cycles			16 machine cycles			16 machine cycles						$f_X/4f_{XT}$ machine cycles (32 machine cycles)		
1	x	x	1 machine cycle			Setting prohibited			1 machine cycle			1 machine cycle					

Remarks 1 : Values with f_X = 4.19 MHz and f_{XT} = 32.768 kHz are indicated in parentheses.

2 : x : don't care

3 : CPU clock ϕ is a clock to be supplied to the internal CPU. The reciprocal of this clock is the minimum instruction time (defined as 1 machine cycle in this manual).

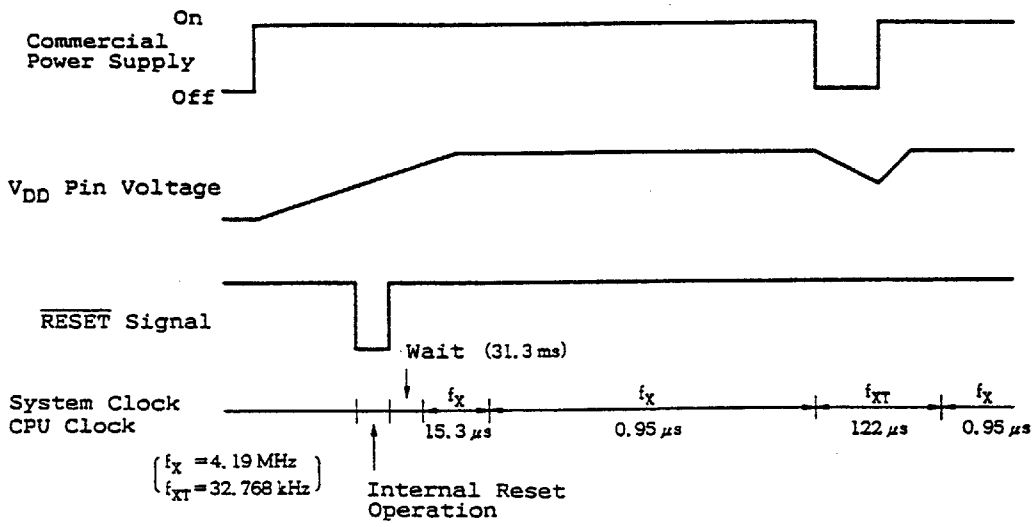
NOTE 1: When PCC = 0001B (with $\Phi = f_{XT}/16$ selected) do not set "1" in SCC bit 0. Switching from the main system clock to subsystem clock should be performed after setting PCC to the other state (PCC $\neq$ 0001B). Do not perform setting of PCC = 0001B when operating with subsystem clock.

2: The values of f_X and f_{XT} vary depending on conditions such as the dispersion of the resonator ambient temperature and load capacitance performance. When f_X is higher than the nominal value, and when f_{XT} is lower than the nominal value, in particular, the machine cycle figure obtained from the expressions $f_X/64f_{XT}$, $f_X/8f_{XT}$ and $f_X/4f_{XT}$ in the table will be larger than that obtained with the nominal values of f_X and f_{XT}. Therefore, when setting the wait time required to switch between the system clock and CPU clock, it should be made longer than the number of machine cycles obtained with the nominal values of f_X and f_{XT}.

(2) System clock and CPU clock switching procedure

System clock and CPU clock switching is described with reference to Figure 5-17.

Figure 5-17 System Clock and CPU Clock Switching



① When a $\overline{\text{RESET}}$ signal is generated, the CPU starts at the lowest speed (15.3 us/4.19 MHz) of the main system clock after the wait time (31.3 ms/ 4.19 MHz) for maintaining the oscillation stabilization time.

② The CPU rewrites the PCC and operates at its maximum available speed after the elapse of sufficient time for the V_{DD} pin voltage to increase to a voltage allowing the highest speed operation.

③ The CPU detects commercial power-off from the interrupt input*, sets SCC bit 0 to "1" and operates with the subsystem clock. (Subsystem clock oscillation must have started.) After the elapse of the time required for the CPU clock to switch to the subsystem clock (32 machine cycles), the CPU sets SCC Bit 3 to "1" and stops main system clock oscillation.

④ After the CPU detects the commercial power reset from the interrupt, it clears SCC bit 3 to "0" and starts main system clock oscillation. Following the elapse of the time required for oscillation stabilization, the CPU clears SCC bit 0 to "0" and operates at its highest speed.

*: Use of INT4 is effective.

5.2.4 CLOCK OUTPUT CIRCUIT

(1) Clock output circuit configuration

The clock output circuit is configured as shown in Figure 5-18.

(2) Clock output circuit functions

The clock output circuit generates clock pulses from the P22/PCL pin. It is used to generate remote controlled outputs and supply the peripheral LSI with clock pulses.

Use the following procedure to generate clock pulses:

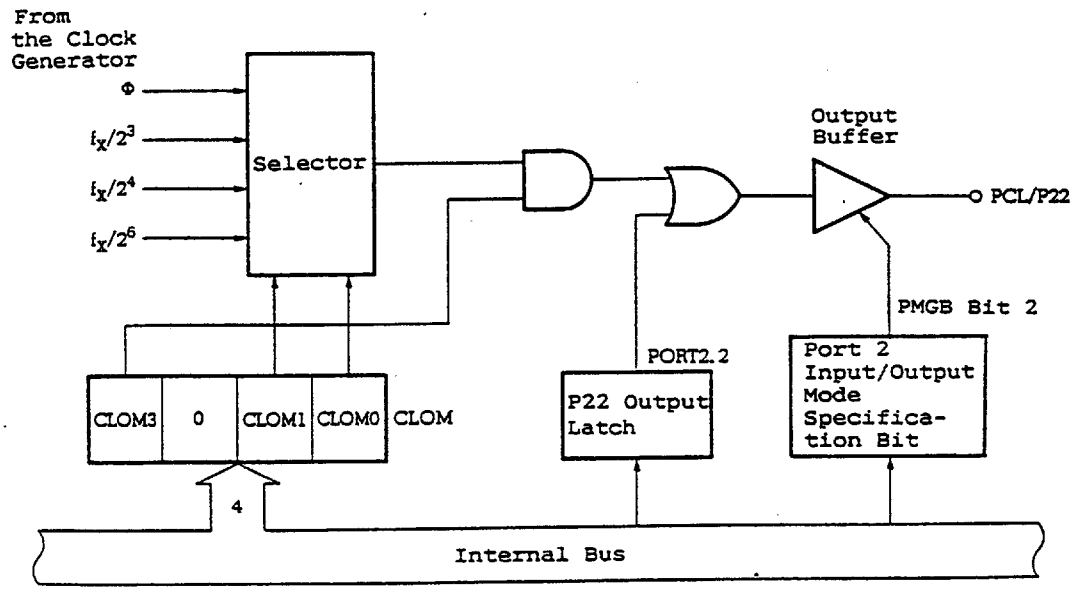
(a) Select the clock output frequency with clock output disabled.

(b) Write "0" in the P22 output latch.

(c) Set the port 2 input/output mode to output.

(d) Enable clock output.

Figure 5-18 Clock Output Circuit Configuration



Remarks: When the clock output enabled/disabled status is switched, pulses with short widths will not be output.

(3) Clock output mode register (CLOM)

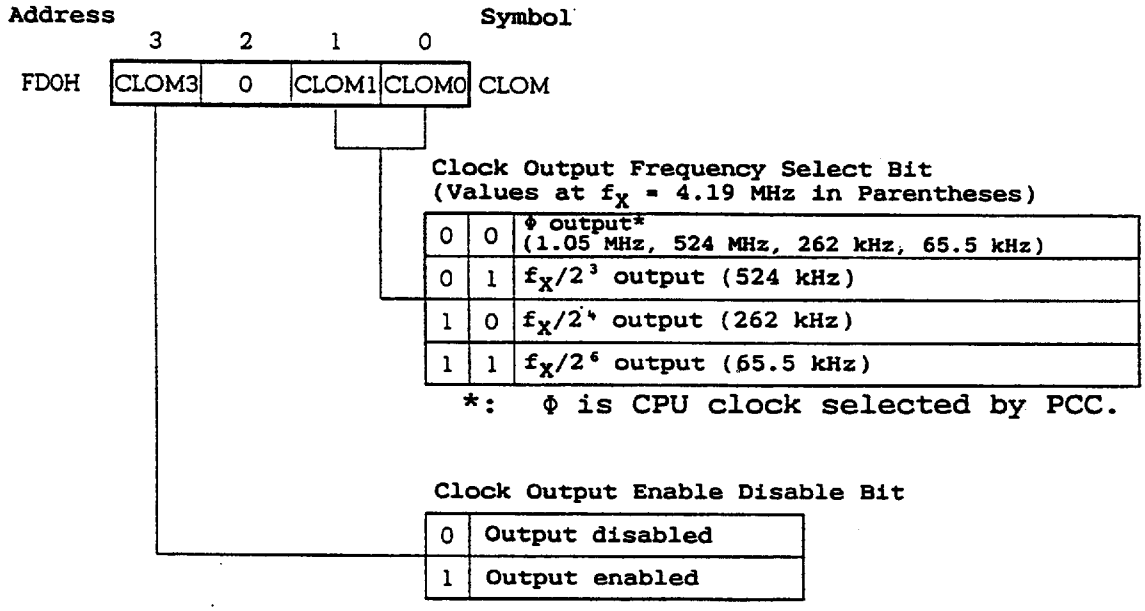
The CLOM is a 4-bit register used to control clock output. It is set by a 4-bit memory manipulation instruction. Data cannot be read from the CLOM.

Example: Generate CPU clock Φ from PCL/P22 pin.

```
SEL  MB15      ; Or CLR1 MBE
MOV  A, #1000B
MOV  CLOM, A
```

When the $\overline{\text{RESET}}$ signal is generated, the CLOM is cleared to "0" and clock output is disabled.

Figure 5-19 Clock Output Mode Register Format



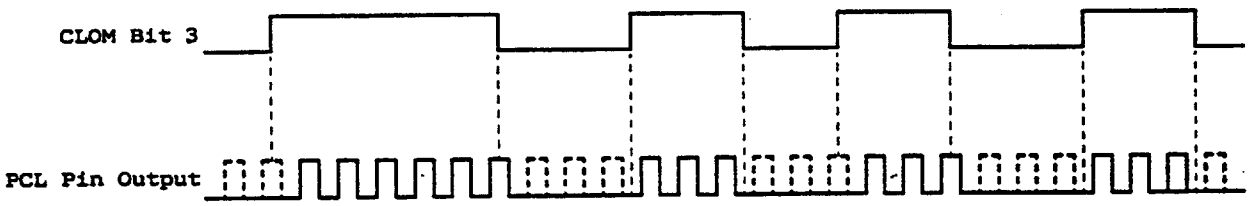
NOTE: Be sure to write "0" to bit 2 of CLOM.

(4) Example of application to remote controlled output

The uPD75336 clock output functions can be applied to remote controlled output. The remotely controlled output carrier frequency is selected by the clock frequency select bit of the clock output mode register. Pulse output is enabled/disabled by controlling the clock output enable/disable bit by software.

When switching the clock output enabled/disabled status, pulses with short widths will not be output.

Figure 5-20 Example of Application to Remote Controlled Output



5.3 BASIC INTERVAL TIMER

The uPD75336 is equipped with an 8-bit basic interval timer which has the following functions:

(a) Reference time generation (4 time intervals)

(b) Watchdog timer application to detect inadvertent program loop

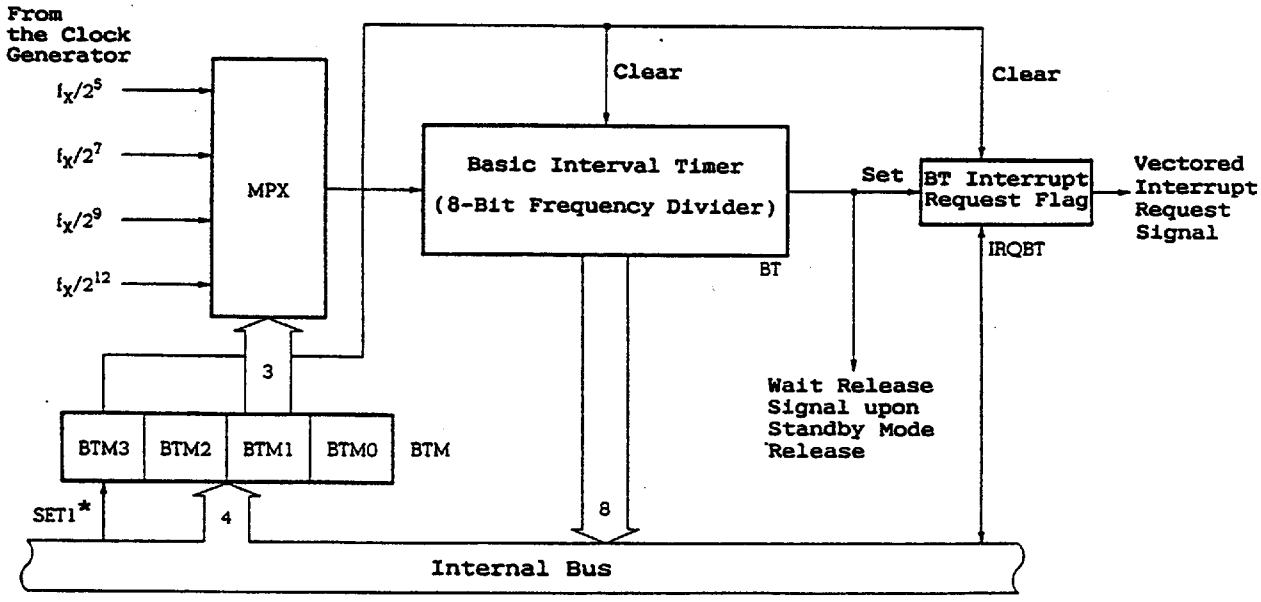
(c) Wait time selection and count upon standby mode release

(d) Count contents read

5.3.1 BASIC INTERVAL TIMER CONFIGURATION

The basic interval timer is configured as shown in Figure 5-21.

Figure 5-21 Basic Interval Timer Configuration



*: Instruction execution

5.3.2 BASIC INTERVAL TIMER MODE REGISTER (BTM)

The BTM is a 4-bit register used to control basic interval timer operations.

The BTM is set by a 4-bit memory manipulation instruction.

Bit 3 can be set independently by a bit manipulation instruction.

Example 1: Set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
SEL  MB15      ; Or CLR1 MBE
MOV  A, #1111B
MOV  BTM, A    ; BTM ← 1111B
```

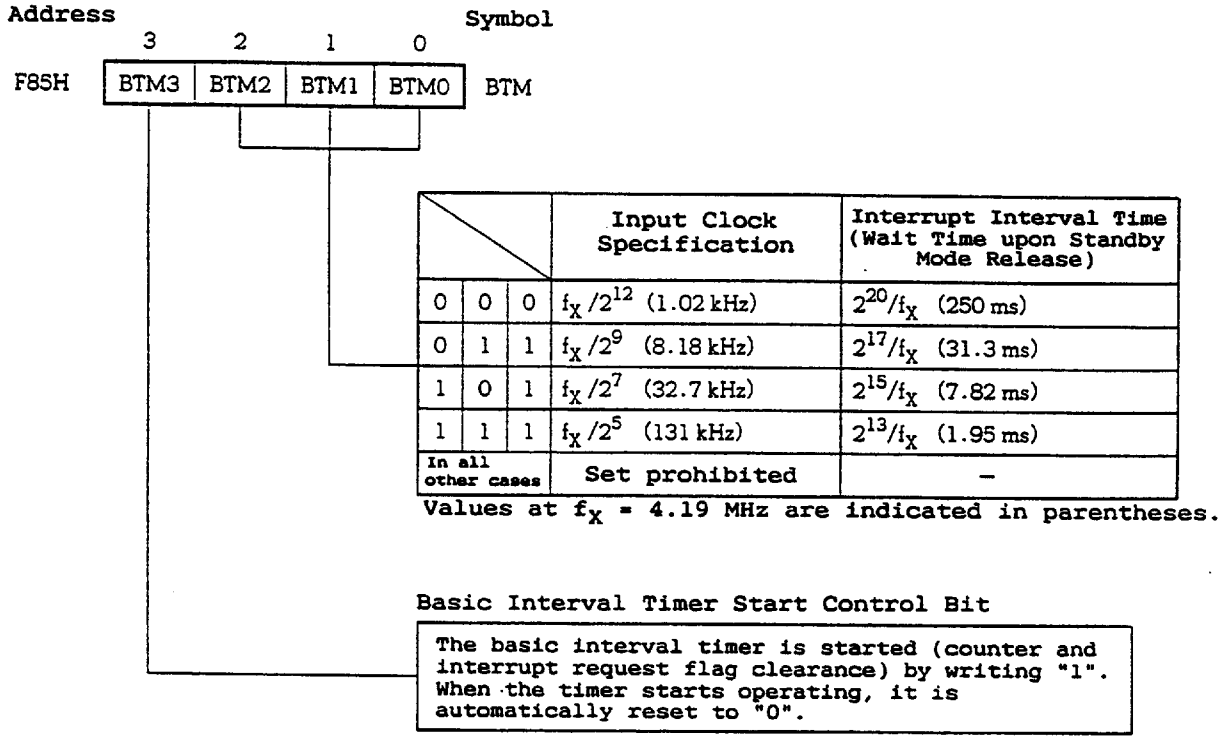
2: Clear BT and IRQBT (watchdog timer application).

```
SEL   MB15     ; Or CLR1 MBE
SET1  BTM. 3   ; BTM bit 3 is set to "1".
```

When bit 3 is set to "1", the basic interval timer contents and the basic interval timer interrupt request flag (IRQBT) are simultaneously cleared (basic interval timer start).

When the $\overline{\text{RESET}}$ signal is generated, the contents are cleared to "0" and the interrupt request signal generation interval time is set to its maximum value.

Figure 5-22 Basic Interval Timer Mode Register Format



			Input Clock Specification	Interrupt Interval Time (Wait Time upon Standby Mode Release)
0	0	0	$f_X/2^{12}$ (1.02 kHz)	$2^{20}/f_X$ (250 ms)
0	1	1	$f_X/2^9$ (8.18 kHz)	$2^{17}/f_X$ (31.3 ms)
1	0	1	$f_X/2^7$ (32.7 kHz)	$2^{15}/f_X$ (7.82 ms)
1	1	1	$f_X/2^5$ (131 kHz)	$2^{13}/f_X$ (1.95 ms)
In all other cases			Set prohibited	–

Values at f_X = 4.19 MHz are indicated in parentheses.

Basic Interval Timer Start Control Bit

The basic interval timer is started (counter and interrupt request flag clearance) by writing "1". When the timer starts operating, it is automatically reset to "0".

5.3.3 BASIC INTERVAL TIMER OPERATION

The basic interval timer (BT) is incremented by clock pulses from the clock generator and sets the interrupt request flag (IRQBT) due to an overflow. BT count operation cannot be stopped.

Four interrupt generation intervals are available by setting the BTM (Figure 5-22).

The basic interval timer and the interrupt request flag can be cleared by setting bit 3 of the BTM to "1" (interval timer start instruction).

The count status can be read from the basic interval timer (BT) by an 8-bit manipulation instruction. Data cannot be written to the BT.

NOTE: When reading the basic interval timer count contents, execute the read instruction twice and compare the two read contents so as not to read unstable data undergoing count updating. If the two values are both acceptable, use the second read value as the correct one. If they differ completely, execute reading again from the beginning.

Example: BT count contents read

```
        SET1   MBE
        SEL    MB15
        MOV    ML, #BT;  BT address set to HL
LOOP:   MOV    XA, @HL;  1st read
        MOV    BC, XA
        MOV    XA, @HL;  2nd read
        SKE    XA, BC
        BR     LOOP
```

To obtain the oscillation stabilization time from STOP mode release to system clock oscillation stabilization, the wait function is available to stop CPU operation until the basic interval timer overflows.

The wait time after the $\overline{\text{RESET}}$ signal is generated is fixed. However, if the STOP mode has been released by interrupt generation, the wait time can be selected by BTM setting. In that case, the wait time is equal to the interval time shown in Figure 5-22. BTM setting must be done before STOP mode setting. (For details, refer to Chapter 7. "Standby Functions.)

5.3.4 BASIC INTERVAL TIMER APPLICATION EXAMPLE

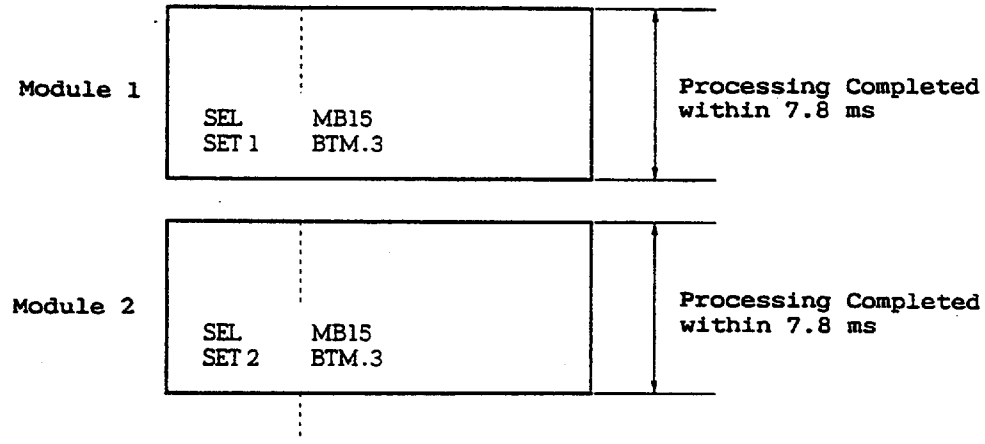
Example 1: Enable basic interval timer interruption and set the interrupt generation interval to 1.95 ms (at 4.19MHz).

```
SEL  MB15
MOV  A, #1111B
MOV  BTM, A    ; Set and start
EI             ; Interrupt enabled
EI   IEBT      ; BT interrupt enabled
```

2: Watchdog timer application

Divide the program into several modules which terminate processing within the BT set period and clear BT and IRQBT at the end of each module. If an interrupt is generated, the program is judged to have an inadvertent loop.

```
Initialization
SEL  MB15
MOV  A, #1101B; Interval set to 7.8 ms
MOV  BTM, A    ; Set and start
EI
EI   IEBT
```



Example 3: Set the wait time upon STOP mode release by interruption to 7.8 ms.

```
SEL   MB15
MOV   A, #1101B
MOV   BTM, A    ; BTM ← 1101B
STOP            ; STOP Mode set
NOP
```

4: Set the high-level width of a pulse to be input to INT4 interrupt (both-edge detection). (The pulse width should not exceed the BT set value which should be 7.8 ms or more.)

```
<INT4 interrupt routine (MBE = 0)>
LOOP:  MOV   XA, BT     ; 1st read
       MOV   BC, XA     ; Data storage
       MOV   XA, BT     ; 2nd read
       SKE   A, C
       BR    LOOP
       MOV   A, X
       SKE   A, B
       BE    LOOP
       SKT   PORT0.0    ; P00 = 1?
       BR    AA         ; NO
       MOV   XA, BC     ; Data storage into data
                          memory
       MOV   BUFF, XA
       CLR1  FLAG       ; Data available.  Flag
                          clear
       RETI
```

```
AA  :  MOV   HL, #BUFF
       MOV   A, C
       SUBC  A, @HL
       INCS  L
       MOV   C, A
       MOV   A, B
       SUBC  A, @HL
       MOV   B, A
       MOV   XA, BC
       MOV   BUFF, XA ; Data storage
       SET1  FLAG     ; Data available.  Flag
                        set
       RETI
```

5.4 WATCH TIMER

The uPD75336 incorporates one watch timer channel which has the following functions.

(a) Sets the test flag (IRQW) at 0.5 second intervals. The standby mode can be released by IRQW.

(b) 0.5 second intervals can be set with the main system or subsystem clock.

(c) The fast mode enables setting of a 128 times (3.91 ms) interval useful to program debugging and inspection.

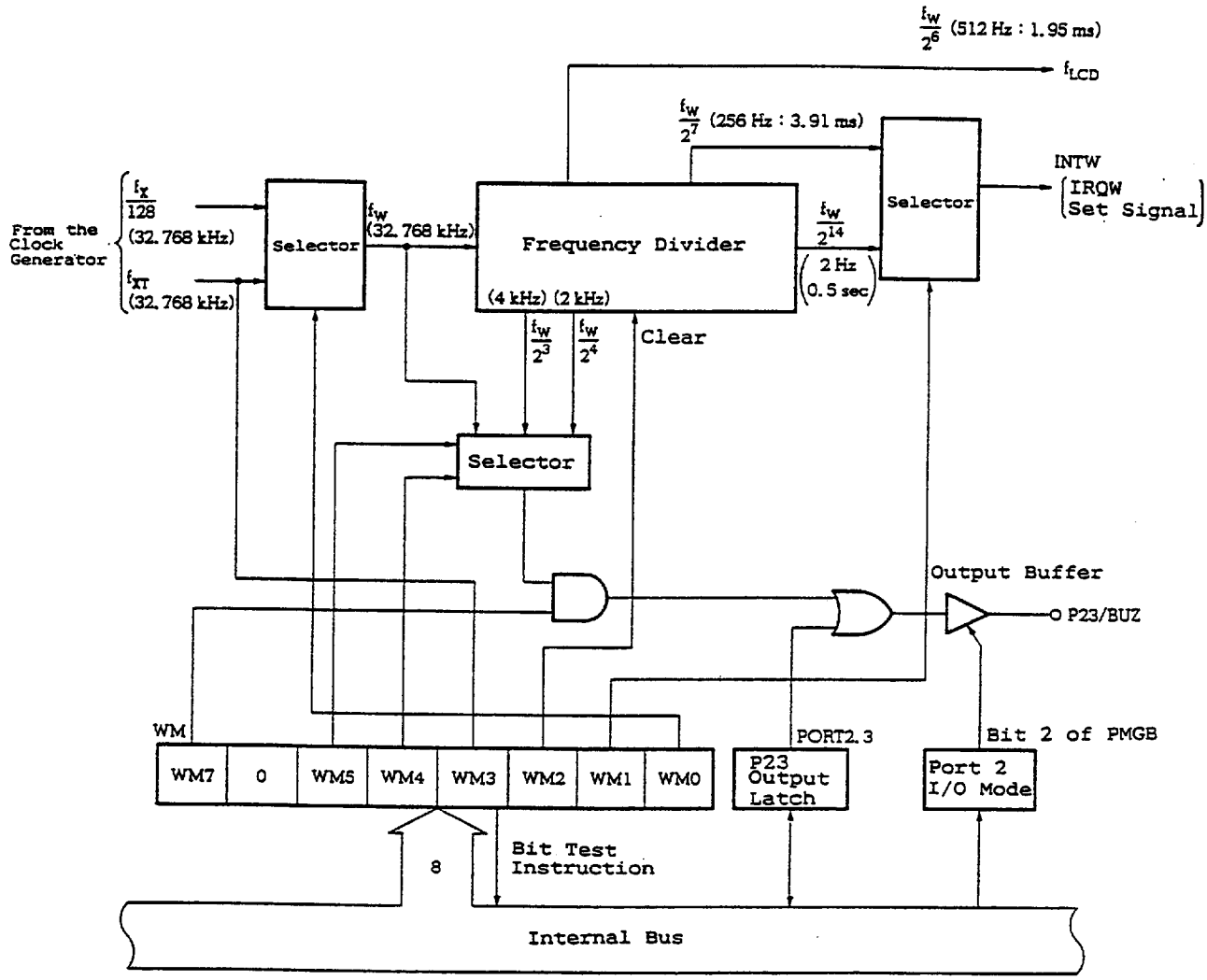
(d) The fixed frequencies (2.048 kHz, 4.096 kHz and 32.768 kHz) can be output to the P23/BUZ pin for use to generate buzzer sound or trim the system clock oscillator frequency.

(e) Since the frequency divider can be cleared, the clock can be started from zero seconds.

5.4.1 WATCH TIMER CONFIGURATION

The watch timer is configured as shown in Figure 5-23.

Figure 5-23 Watch Timer Block Diagram



Values at f_X = 4.194304 MHz and f_{XT} = 32.768 kHz are indicated in parentheses.

5.4.2 CLOCK MODE REGISTER

The clock mode register (WM) is an 8-bit register used to control the watch timer. Its format is shown in Figure 5-24.

The clock mode register is set by an 8-bit manipulation instruction, except for bit 3. Bit 3 is used to test the XT1 pin input level. A bit test enables the input level for the XT1 pin to be tested. Data cannot be written to this bit. When the $\overline{\text{RESET}}$ signal is generated, all bits except bit 3 are cleared to "0".

Example: Generate time with the main-system clock (4.19 MHz). Enable buzzer output.

```
CLR1  MBE
MOV   XA, #84H
MOV   WM, XA   ; WM set
```

Figure 5-24 Clock Mode Register Format

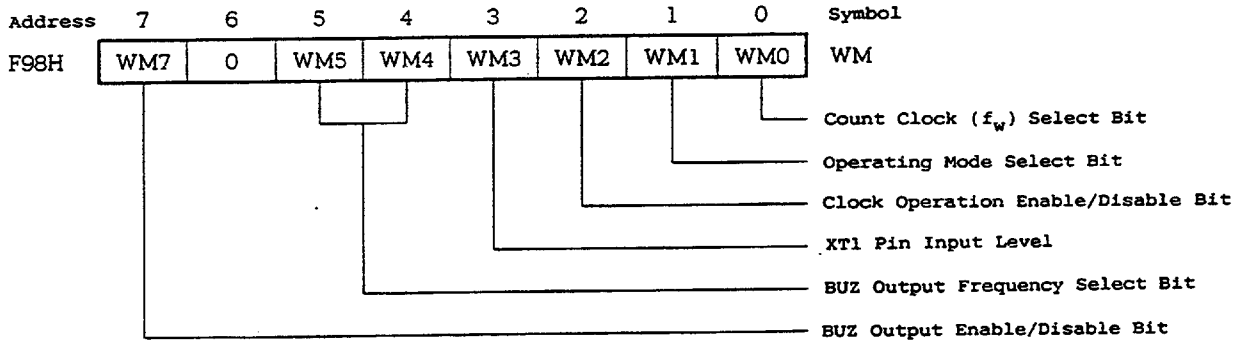


Figure 5-24 Clock Mode Register Format (cont'd)

Count Clock (f_W) Select Bit

WM0	0	System clock divided output: $f_X/128$ selected
	1	Subsystem clock: f_{XT} selected

Operation Mode Select Bit

WM1	0	Normal clock mode ($f_W/2^{14}$: IRQW set at 0.5 sec)
	1	Fast clock mode ($f_W/2^7$: IRQW set at 3.91 ms)

Clock Operation Enable/Disable Bit

WM2	0	Clock operation stop (frequency divider clear)
	1	Clock operable

XT1 Pin Input Level (Only Bit Test Enabled)

WM3	0	XT1 pin input at low level
	1	XT1 pin input at high level

BUZ Output Frequency Select Bit

WM5	WM4	BUZ output frequency
0	0	$\frac{f_W}{2^{14}}$ (2.048 kHz)
0	1	$\frac{f_W}{2^3}$ (4.096 kHz)
1	0	Setting prohibited
1	1	f_W (32.768 kHz)

BUZ Output Enable/Disable Bit

WM7	0	BUZ output disabled
	1	BUZ output enabled

Remarks: Values at f_W = 32.768 kHz are indicated in parentheses.

5.5 TIMER/EVENT COUNTER

5.5.1 TIMER/EVENT COUNTER CONFIGURATION

The uPD75336 incorporates to timer/event counter channels. Its format is shown in Figure 5-25.

(1) Differences between timer/event counter channel 0 and channel 1.

The two channels have the following differences:

Table 5-6 Differences between Timer/Event Counter Channel 0 and Channel 1

Differences	Channel 0	Channel 1
Select count pulse	$f_X/2^{10}$, $f_X/2^8$, $f_X/2^6$, $f_X/2^4$	$f_X/2^{12}$, $f_X/2^{10}$, $f_X/2^8$, $f_X/2^6$
Clock supply to serial interface	Possible	Not possible

(2) Timer/event counter functions

The timer/event counter has the following functions:

(a) Programmable interval timer operation

(b) Output of square waves with selected frequency to the PTOn pin

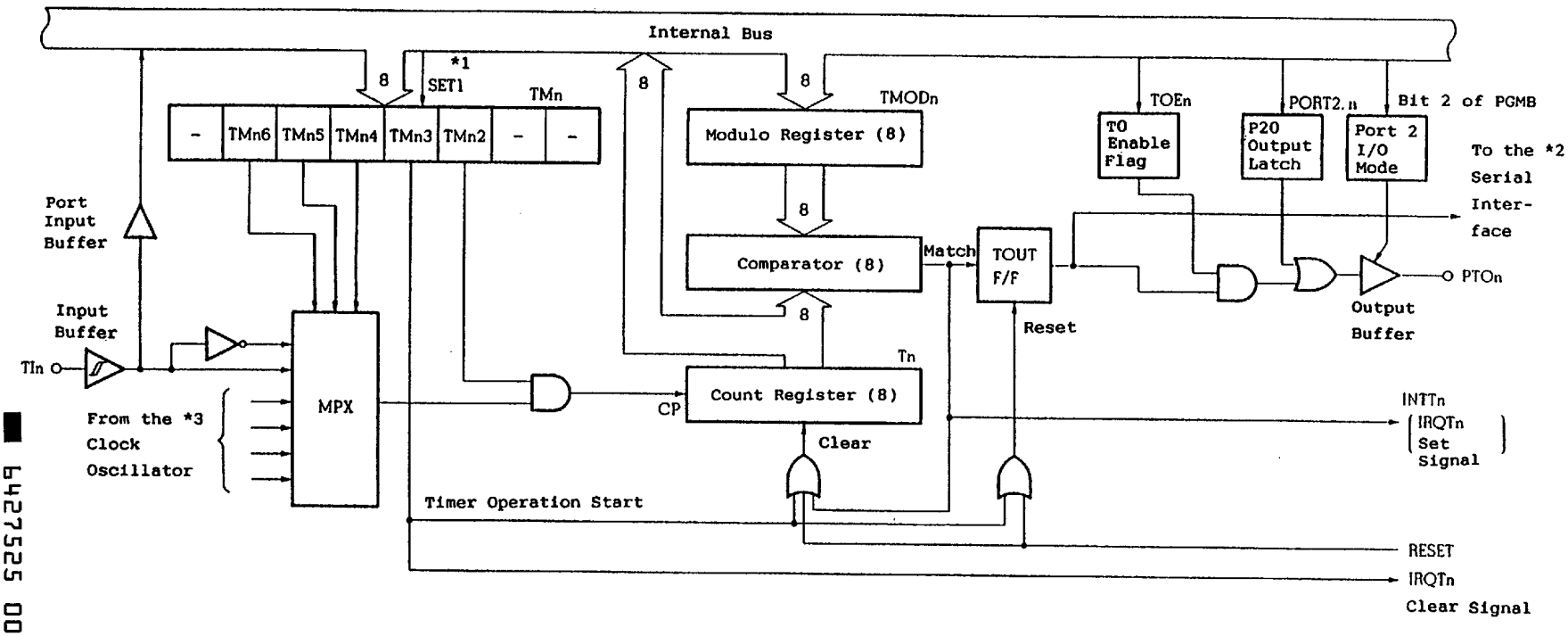
(c) Event counter operation

(d) Output of N-divided TIn pin input to the PTOn pin (frequency divider operation)

(e) Serial shift clock supply to the serial interface circuit

(f) Count state read function

Figure 5-25 Timer/Event Counter Block Diagram



Remarks : n= 0, 1 (n indicates channel number)

*1: Instruction execution
2: Only timer/event counter channel 0 enabled
3: Refer to Table 5-6 "Differences between Timer/ Event Counter Channel 0 and Channel 1" and Figure 5-11 "Clock Generator Block Diagram".

5.5.2 TIMER/EVENT COUNTER BASIC CONFIGURATION AND OPERATION

The timer/event counter can select several operating modes using the timer/event counter mode register (TM0/TM1). Its basic configuration and operations are as follows.

(1) Count pulse CP is selected by TMn setting and is input to the 8-bit count register (T0/T1).

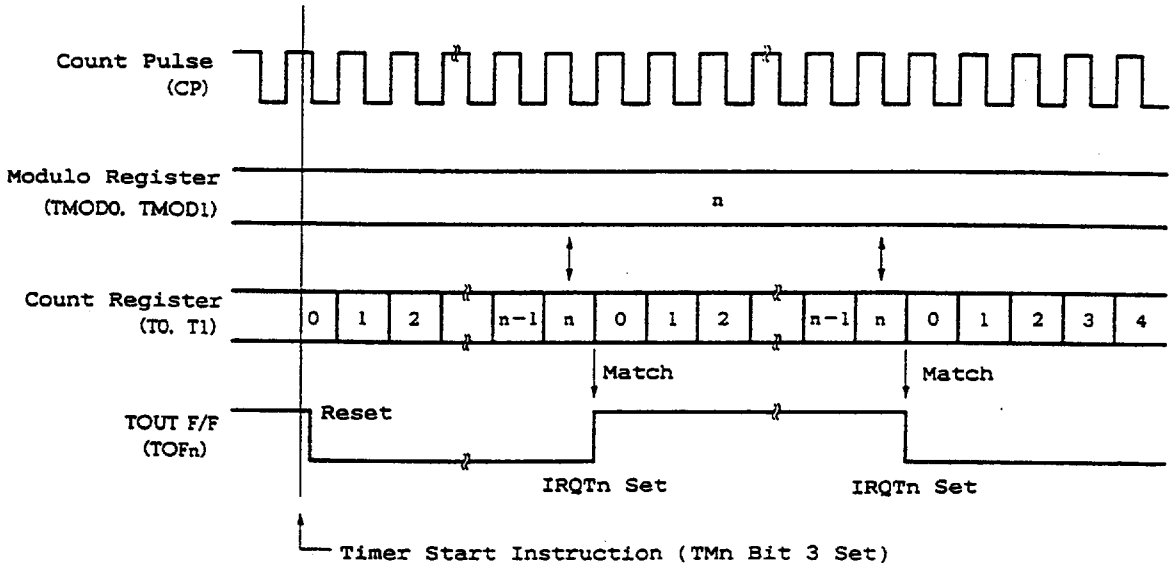
(2) Tn is a binary 8-bit up-counter which is incremented by one when CP is input. When the $\overline{\text{RESET}}$ signal is generated, TMn bit 3 is set (timer start) or a match signal is generated, Tn is cleared to "0". Data can be read from Tn by the 8-bit memory manipulation instruction. Data cannot be written to (T0/T1).

(3) Modulo register (TMOD0, TMOD1) is an 8-bit register used to determine the Tn count number. Values are set in TMODn by the 8-bit memory manipulation instruction but data cannot be read from TMODn. When the $\overline{\text{RESET}}$ signal is generated, TMODn is initialized to FFH.

(4) The comparator compares Tn and TMODn contents. When they match each other, the comparator generates a match signal and sets the interrupt request flag (IRQT0/IRQT1).

Figure 5-26 shows count operation timings.

Figure 5-26 Count Operation Timings



5.5.3 TIMER/EVENT COUNTER MODE REGISTERS (TM0, TM1)

The timer/event counter mode register (TM0, TM1) is an 8-bit register used to control the timer/event counter. Its format is shown in Figure 5-27.

TM0 and TM1 are set by the 8-bit memory manipulation instruction.

Bit 3 is a timer start bit and can be manipulated individually. When the timer starts operating, bit 3 is reset to "0".

When the $\overline{\text{RESET}}$ signal is generated, all bits of the timer mode register are cleared to "0".

Example 1: Start the timer in the interval timer mode with CP = 4.09 kHz.

```
SEL   MB15          ; Or CLR1 MBE
MOV   XA, #01001100B
MOV   TM0, XA       ; TM0 ← 4CH
```

Example 2: Restart the timer according to timer mode register setting.

```
SEL   MB15 ; Or CLR1 MBE
SET1  TM0.3; TM0. bit 3 ← 1
```

Figure 5-27 Timer/Event Counter Mode Register Format (Channels 0 and 1)

Address

Symbol

	7	6	5	4	3	2	1	0	
FA0H	–	TM06	TM05	TM04	TM03	TM02	–	–	TM0 (Channel 0)
FA8H	–	TM16	TM15	TM14	TM13	TM12	–	–	TM1 (Channel 1)

Operating Mode

	Count Operation
0	Stop (with count contents held)
1	Count operation

Timer Start Instruction Bit

Writing "1" clears the counter and the IRQT0 flag. If bit 2 has been set to "1", writing "1" starts the count operation.

Count Pulse (CP) Select Bit (n = 0, 1)

TMn6	TMn5	TMn4	Channel 0	Channel 1
			Count Pulse (CP)	
0	0	0	TIn input rising edge	
0	0	1	TIn input falling edge	
1	0	0	$f_X/2^{10}$ (4.09 kHz)	$f_X/2^{12}$ (1.02 kHz)
1	0	1	$f_X/2^{8}$ (16.4 kHz)	$f_X/2^{10}$ (4.09 kHz)
1	1	0	$f_X/2^{6}$ (65.5 kHz)	$f_X/2^{8}$ (16.4 kHz)
1	1	1	$f_X/2^{4}$ (262 kHz)	$f_X/2^{6}$ (65.5 kHz)
In all other cases			Setting prohibited	

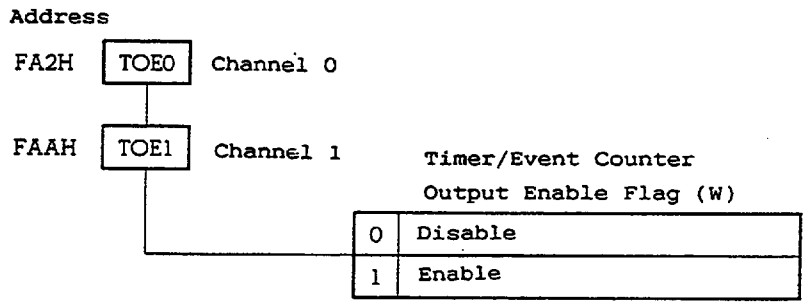
Remarks: Values at f_X = 4.19 MHz are indicated in parentheses.

5.5.4 TIMER/EVENT COUNTER OUTPUT ENABLE FLAGS (TOE0, TOE1)

The timer/event counter output enable flag (TOE0, TOE1) is a flag used to control the timer out F/F (TOUT F/F) status output enabling disabling for PTO0 and PTO1 pins. The timer out F/F is an F/F which is inverted by a match signal from the comparator. When bit 3 of the timer mode register (TM0, TM1) is set to "1", the timer out F/F is reset to "0".

When the $\overline{\text{RESET}}$ signal is generated, TOE0, TOE1 and the timer out F/F are cleared to "0".

Figure 5-28 Timer/Event Counter Output Enable Flag Format (Channels 0, 1)



5.5.5 TIMER/EVENT COUNTER OPERATING MODES

The count operation stop mode and the count operating mode are available by the mode register setting for the timer/event counter operation.

The following operations are enabled irrespective of the mode register setting:

① TIn pin signal input and test (P13 input of dual-function pins is testable) *1

② Output of timer out F/F status to PTOn *2

③ Modulo register (TMODn) setting

④ Count register (Tn) read

⑤ Interrupt request flag (IRQTn) set/clear/test

*1: When using timer/event counter channel 1 (TI1), set the P80 dual-function pin to the input mode.

2: When using the timer/event counter output pins (PTO0, PTO1), set the P20 and P21 dual-function pins as follows.

① Clear the P20 and P21 output latches.

② Set port 2 to the output mode.

③ Disable on-chip pull-up resistor at port 2.

(1) Count operation stop mode

When TMn bit 2 is "0", this mode is set. In this mode, the count operation is not carried out because count pulse (CP) supply to the count register is stopped.

(2) Count operating mode

When TMn bit 2 is "1", this mode is set. In this mode, the count pulses selected by bits 4 to 6 are supplied to the count register and the count operation is carried out as shown in Figure 5-29.

Timer operation can normally by started using the following procedure:

① Set the count value in the modulo register (TMODn).

② Set the operating mode, count clock and start instruction in the mode register (TMn).

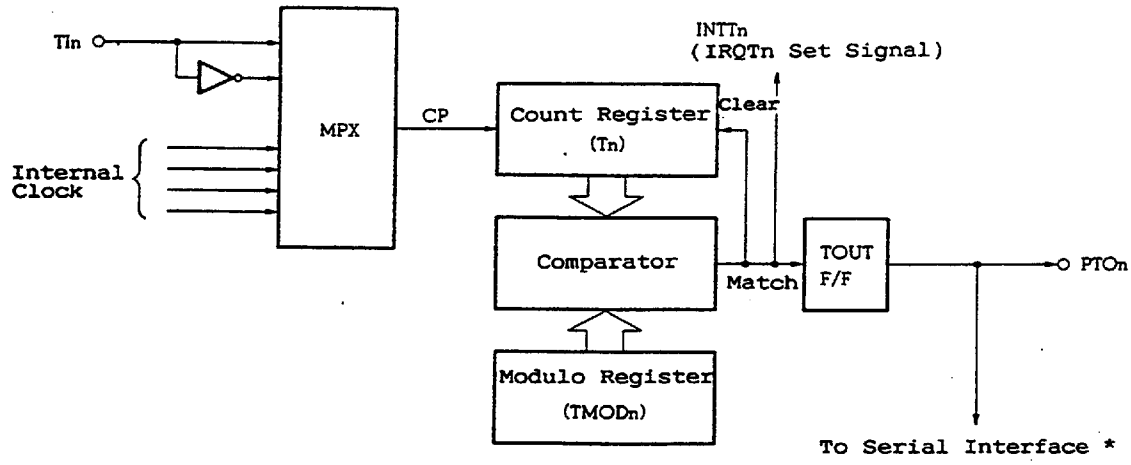
The modulo register is set by an 8-bit data transfer instruction.

NOTE: Set a value other than "0" in the modulo register.

Example: Set 3FH in the channel 0 modulo register.

```
SEL   MB15      ; Or CLR1 MBE
MOV   XA. #3FH
MOV   TMOD0, XA
```

Figure 5-29 Operation in Count Operating Mode



*: A signal to the serial interface can be output to timer/event counter channel 0 only.

5.5.6 TIMER/EVENT COUNTER TIME SETTING

[Timer setting time] (cycle) is obtained by dividing [modulo register contents + 1] by [count pulse frequency] selected by setting the timer mode register.

$$T\ (SEC) = \frac{N + 1}{f_{CP}} = (N + 1) \cdot (Resolution)$$

T (SEC) : Timer set time (sec)
f_{CP} (Hz): Count pulse frequency (Hz)
N : Modulo register value (N ≠ 0)

Once the timer is set, an interrupt request signal (IRQTn) is generated at the set interval.

Table 5-7 shows the resolution and maximum set time (when FFH is set in the modulo register) with each count pulse of the timer/event counter.

Example: Generate 30 ms time interval (f_X = 4.194304 MHz).

In this case, use a mode with the maximum set time of 62.5 ms.

The time interval is given as

$$\frac{30\ ms}{244\ us} = 122.9 \fallingdotseq 7AH$$

Set 79H in the modulo register.

```
SEL  MB15
MOV  XA, #79H
MOV  TMOD0, XA
```

Table 5-7 Resolution and Maximum Set Time
(with f_X = 4.19 MHz)

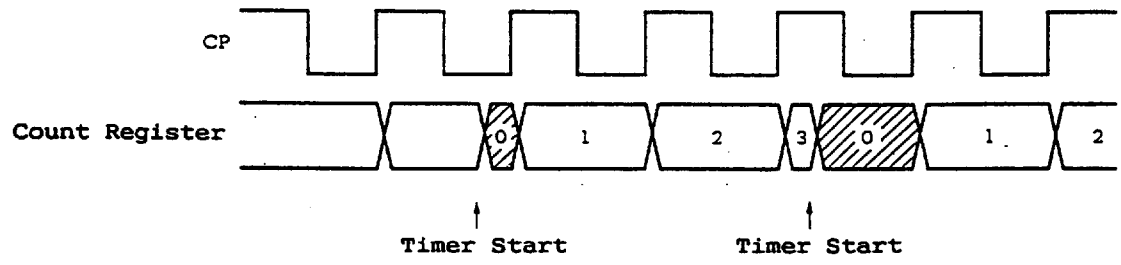
Mode Register			Timer Channel 0		Timer Channel 1	
TMn6	TMn5	TMn4	Resolution	Maximum Set Time	Resolution	Maximum Set Time
1	0	0	244 us	62.5 ms	980 us	250 ms
1	0	1	61.1 us	15.6 ms	244 us	62.5 ms
1	1	0	15.3 us	3.91 ms	61.1 us	15.6 ms
1	1	1	3.81 us	977 us	15.3 us	3.91 ms

5.5.7 TIMER/EVENT COUNTER APPLICATION PRECAUTIONS

(1) Errors upon timer start

The time from timer start (TMn bit 3 set) to match signal generation may have a maximum difference of one clock count pulse (CP) from the value calculated in Section 5.5.6 "Timer/Event Counter Time Setting". This is because the count register is cleared asynchronously with regard to the CP as shown in Figure 5-30.

Figure 5-30 Count Register Clear Timing



(2) Precautions in timer start

Timer start (TMn bit 3 set) clears the count register Tn and the interrupt request flag IRQTn. If the timer is in the operating mode and IRQTn set and timer start occur simultaneously, IRQTn may not be cleared. This poses no problem when IRQTn is used as a vectored interrupt. In IRQTn test applications, however, a problem is that although the timer has started, IRQTn has been set. Thus, when starting the timer at a timing at which IRQTn may be set, stop the timer (by setting TMn bit 2 to "0") and then restart it, or execute timer start twice.

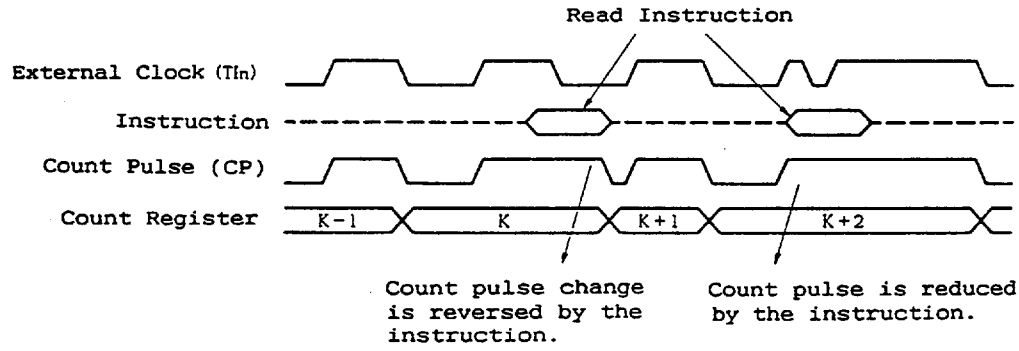
Example: Timer start at timing when IRQT0 may be set

```
SEL   MB15
MOV   XA, #0
MOV   TM0, XA ; Timer stop
MOV   XA, #4CH
MOV   TM0, XA ; Restart

Or

SEL   MB15
SET1  TM0.3
SET1  TM0.3   ; Restart
```

(3) Errors in count register read

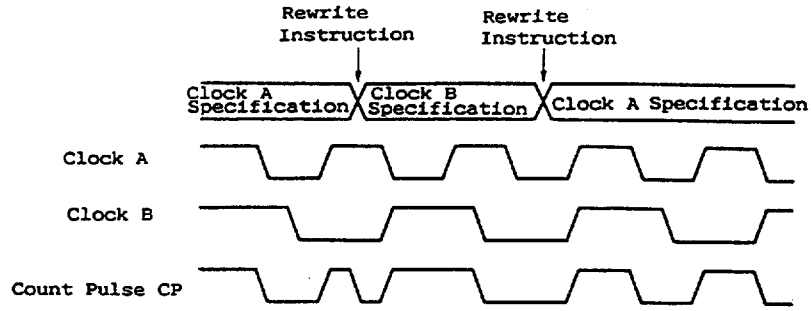
Count register contents can be read by an 8-bit data memory manipulation instruction. While this instruction is in operation, count pulse change is deferred so that the count register remains unchanged. Thus, when the count pulse signal source is set to TIn input, the count pulse is reduced by the instruction execution time (when the internal clock is used as a count pulse, this does not occur because the clock B in synchronization with the instruction).

Therefore, when applying TIn input as a count pulse and reading count register contents, the signal to be input must be one with a pulse width which will not cause incorrect counting if the count pulse is reduced. That is, the pulse to be input to TIn pin must have a width larger than one machine cycle in which the count is reserved by the read instruction.

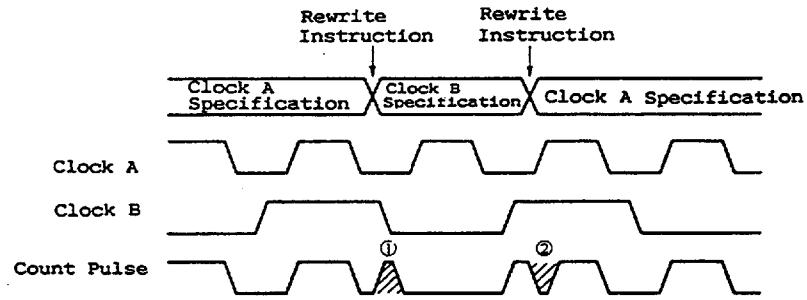


(4) Count pulse changing precautions

If the count pulse has been changed by rewriting the timer mode register, the specification becomes valid just after instruction execution.

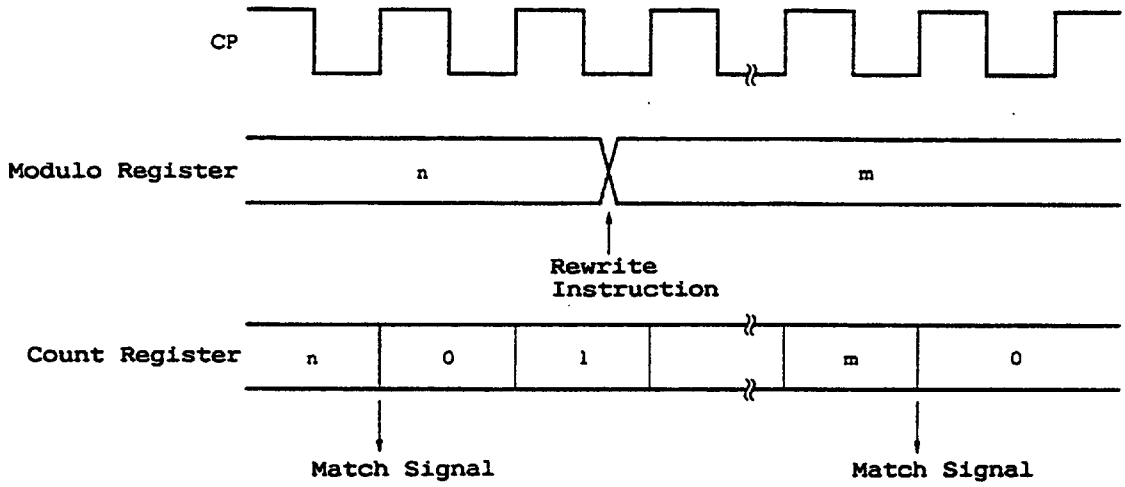


A count pulse spike (① or ②) may occur depending on combinations of clocks when the count pulse is changed. In this case, the counting may be incorrect or count register contents may be destroyed. To prevent this from occurring, when changing the count pulse, be sure to set bit 3 of the counter mode register to "1" and simultaneously restart the timer.

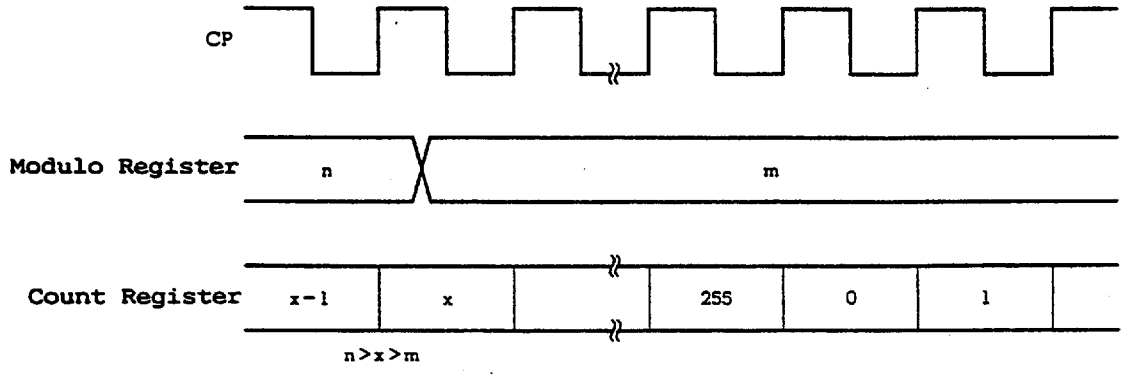


(5) Operation after modulo register change

The modulo register is changed when an 8-bit data memory manipulation instruction is executed.



If the modulo register changed value is smaller than the count register value, the count register continues to count till it overflows. After it overflows, it recounts from 0. Thus, if the value (m) after the modulo register change is smaller than the value (n) before the change, it is necessary to restart the timer after the modulo register is changed.



5.5.8 TIMER/EVENT COUNTER APPLICATIONS

(1) Use timer channel 0 as an interval timer which generates interrupts at 50 ms intervals.

- The most significant 4 bits of the mode register are set to 0100B and the maximum set time of 62.5 ms is selected.
- The least significant 4 bits of the mode register are set to 1100B.
- The modulo register is set as follows.

$$\frac{50\ ms}{244\ us} = 205 \fallingdotseq CDH$$

<Program example>

```
SEL   MB15
MOV   XA, #0CDH
MOV   TMOD0, XA       ; Modulo set
MOV   XA, #01001100B
MOV   TM0, XA         ; Mode set, timer start
EI                    ; Interrupt enabled
EI    IET0            ; Timer interrupt enabled
```

Remarks: In this application, the TI0 pin can be used as an input pin.

(2) When 100 pulses are input from the TI0 pin, an interrupt is generated (with pulse set to high active).

- The most significant 4 bits of the mode register are set to 0000 and the rising edge is selected.
- The least significant 4 bits of the mode register are set to 1100B.
- The modulo register is set to 99 = 100 - 1.

<Program example>

```
SEL   MB15
MOV   XA, #100 - 1
MOV   TMOD0, XA      ; Modulo set
MOV   XA, #00001100B
MOV   TM0, XA        ; Mode set
EI
EI    IET0           ; INTT0 enabled
```

5.6 SERIAL INTERFACE

5.6.1 SERIAL INTERFACE FUNCTIONS

The uPD75336 incorporates a clocked 8-bit serial interface, with four modes available.

These modes are outlined below.

(1) Operation-halted mode

This mode is used when no serial transfer is to be performed, and allows power dissipation to be reduced.

(2) 3-wire serial I/O mode

In this mode, 8-bit data transfer is performed using three lines: The serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

In the 3-wire serial I/O mode simultaneous transmission and reception is possible, increasing the data transfer processing speed.

Either the MSB or LSB can be specified as the start bit for an 8-bit data serial transfer, allowing connection to devices using either as the start bit.

The 3-wire serial I/O mode allows connection to 75X series and 78K series devices and various peripheral I/O devices.

(3) 2-wire serial I/O mode

In this mode, 8-bit data transfer is performed using two lines: The serial clock ($\overline{SCK}$) and the serial data bus (SB0 or SB1). As the output level to the two lines can be manipulated by software, communication with multiple devices is possible.

Also, since software manipulation of the output level is possible for $\overline{SCK}$ and SB0 (or SB1), this mode is compatible with any communication format. It is therefore possible to eliminate the handshaking line previously required for connection to multiple devices, allowing efficient use of input/output ports.

(4) SBI mode (serial bus interface mode)

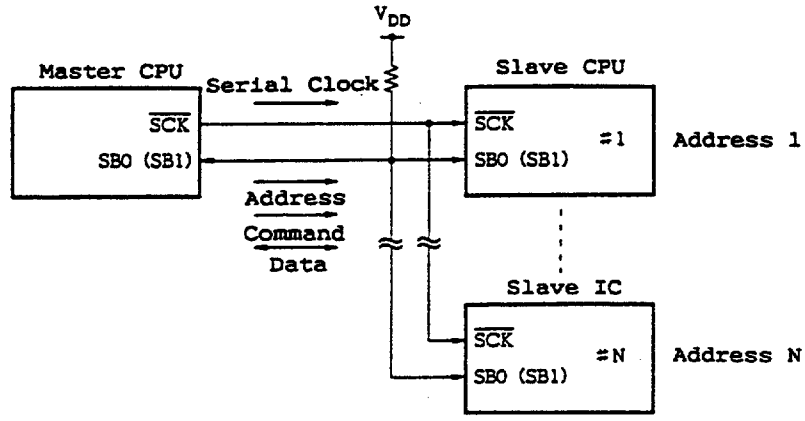
In the SBI mode, communication is performed with multiple devices by means of two lines: The serial clock ($\overline{SCK}$) and the serial data bus (SB0 or SB1).

This mode conforms to the NEC serial bus format.

In the SBI mode, the sender can output to the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. The receiver can determine by hardware whether the received data is an address, command or actual data.

This function allows input/output ports to be used efficiently, as with the 2-wire serial I/O mode, and also allows the serial interface control portion of the application program to be simplified.

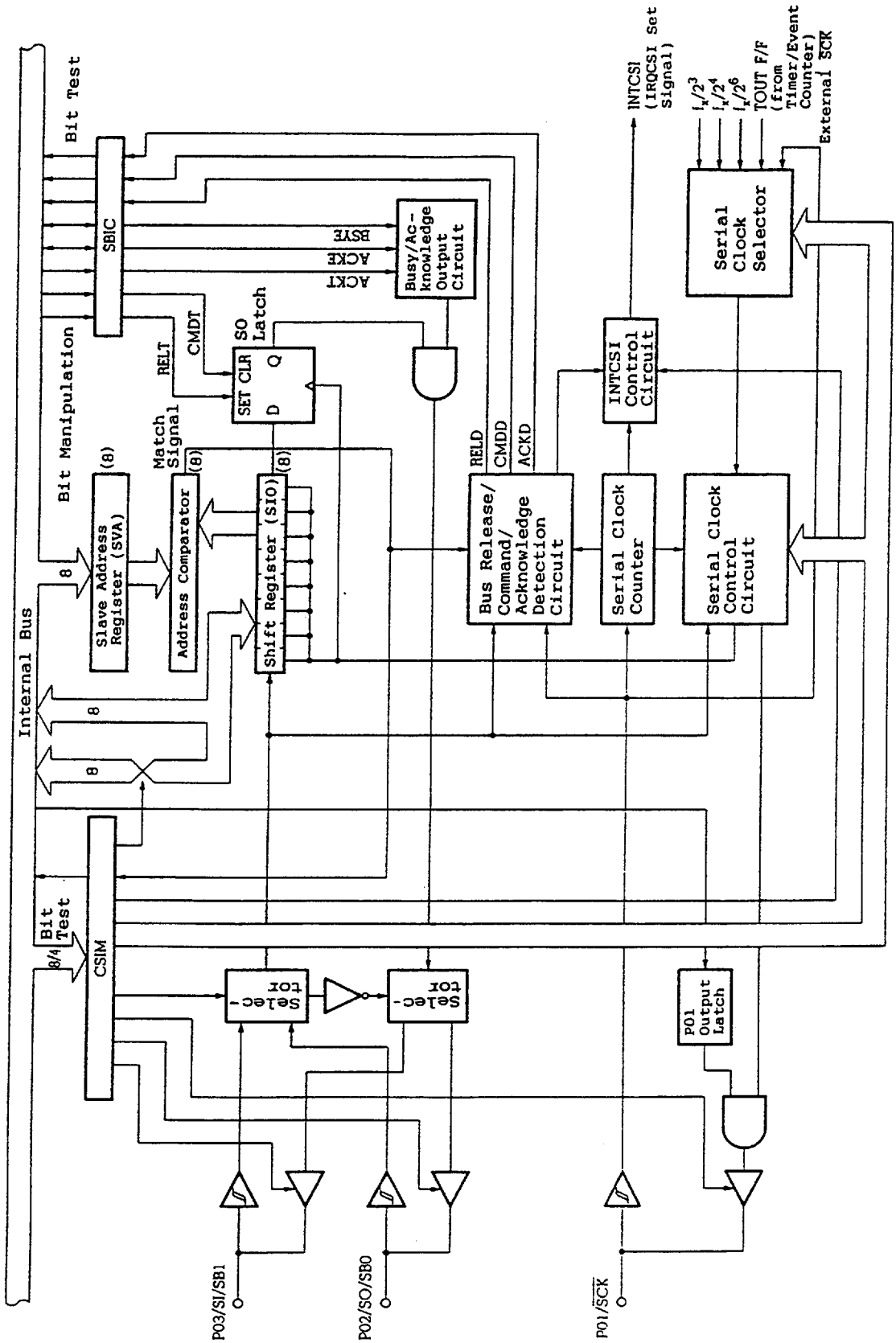
Figure 5-31 SBI System Configuration Example



5.6.2 SERIAL INTERFACE CONFIGURATION

The serial interface block diagram is shown in Figure 5-32.

Figure 5-32 Serial Interface Block Diagram



(1) Serial operating mode register (CSIM)

CSIM is an 8-bit register which specifies the serial interface operating mode, serial clock, wake-up function, etc. (See 5.6.3 (1) "Serial operating mode register" for details.)

(2) Serial bus interface control register (SBIC)

SBIC is an 8-bit register composed of bits which control the serial bus and flags which indicate various statuses of the input data from the serial bus, and is mainly used in the SBI mode. (See 5.6.3 (2) "Serial bus interface control register" for details.)

(3) Shift register (SIO)

The SIO register converts 8-bit serial data to parallel data and 8-bit parallel data to serial data. It performs transmission/reception operations (shift operations) in synchronization with the serial clock. A serial transfer is started by writing data to SIO. Actual transmission/reception operations are controlled by writes to the SIO. (See 5.6.3 (3) "Shift register" for details).

(4) SO latch

A latch which holds the SO/SB0 and SI/SB1 pin levels. Can be directly controlled by software. Set at the end of the 8th $\overline{SCK}$ pulse in the SBI mode. (See 5.6.3 (2) "Serial bus interface control register" for details.)

(5) Serial clock selector

Selects the serial clock to be used.

(6) Serial clock counter

Counts the serial clocks output and input in a transmission/reception operation, and checks that 8-bit data transmission/reception has been performed.

(7) Slave address register (SVA), address comparator

- In SBI mode

Used when the uPD75336 is used as a slave device. The slave sets its own specification number (slave address value) in the SVA register. The master outputs a slave address to select a specific slave.
The address comparator is used to compare the slave address received from the master with the SVA value, and if they match the relevant slave is determined to have been selected.

- In 2-wire serial I/O mode or SBI mode

When the uPD75336 transmits as the master or slave, the SVA register performs error detection. (See 5.6.3 (4) "Slave address register" for details.)

(8) INTCSI control circuit

Controls the generation of interrupt requests. Interrupt request (INTCSI) is generated in the following cases and interrupt request flag (IRQCSI) is set. (Refer to Figure 6-1 "Interrupt Control Circuit Block Diagram")

- In 3-wire and 2-wire serial I/O mode
  Interrupt request is generated on each count of 8 serial clock cycles.

- In SBI mode

  When WUP* = "0"... Interrupt request is generated on each count of 8 serial clock cycles.

  When WUP = "1" ... When the SVA and SIO values match after address reception, interrupt request is generated.

  *: WUP ... Wake-up function specification bit (CSIM bit 5)

(9) Serial clock control circuit

Controls the supply of the serial clock to the shift register. Also controls the clock output to the $\overline{SCK}$ pin when the internal system clock is used.

(10) Busy/acknowledge output circuit, bus release/command acknowledge detection circuit

Performs output and detection of various control signals in the SBI mode.

Does not operate in the 3-wire and 2-wire serial I/O mode.

(11) P01 output latch

Latch used for serial clock generation by software after completion of 8 serial clock cycles.

Set to "1" by reset input.

When the internal system clock is selected as the serial clock, the P01 output latch should be set to "1".

5.6.3 REGISTER FUNCTIONS

(1) Serial operating mode register (CSIM)

The format of the serial operating mode register (CSIM) is shown in Figure 5-33.

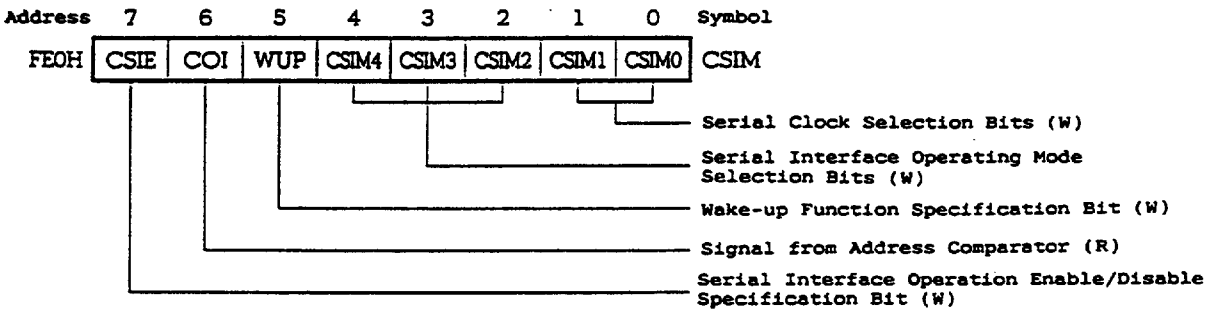
CSIM is an 8-bit register which specifies the serial interface operating mode, serial clock, wake-up function, etc. CSIM is manipulated by 8-bit memory manipulation instructions.

The high-order 3 bits can be manipulated bit by bit using the individual bit names.

Read/write capability differs from bit to bit (see Figure 5-33). Bit 6 can be tested only, and data written to this bit is invalid.

Reset input clears this register to 00H.

Figure 5-33 Serial Operating Mode Register (CSIM) Format



Remarks: (R): Read only
(W): Write only

Figure 5-33 Serial Operating Mode Register (CSIM) Format (cont'd)

Serial clock selection bits (W)

CSIM1	CSIM0	Serial Clock			$\overline{SCK}$ Pin Mode
		3-Wire Serial I/O Mode	SBI Mode	2-Wire Serial I/O Mode	
0	0	Input clock to $\overline{SCK}$ pin from off chip			Input
0	1	Timer/even counter output (TO)			Output
1	0	$f_X/2^4$ (262 kHz)*		$f_X/2^6$ (65.5 kHz)*	
1	1	$f_X/2^3$ (524 kHz)*			

*: ( ): When f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Operating Mode	Shift Register Bit Order	SO Pin Function	SI Pin Function
x	0	0	3-wire serial I/O mode	$SIO_{7 \text{ to } 0} \leftrightarrow XA$ (MSB-first transfer)	SO/PO2 (CMOS output)	SI/PO3 (Input)
		1		$SIO_{0 \text{ to } 7} \leftrightarrow XA$ (LSB-first transfer)		
0	1	0	SBI mode	$SIO_{7 \text{ to } 0} \leftrightarrow XA$ (MSB-first transfer)	SBO/PO2 (N-ch open-drain input/output)	PO3 input
1					PO2 input	SB1/PO3 (N-ch open-drain input/output)
0	1	1	2-wire serial I/O mode	$SIO_{7 \text{ to } 0} \leftrightarrow XA$ (MSB-first transfer)	SBO/PO2 (N-ch open-drain input/output)	PO3 input
1					PO2 input	SB1/PO3 (N-ch open-drain input/output)

Remarks: x: don't care

Figure 5-33 Serial Operating Mode Register (CSIM) Format (cont'd)

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer in each mode.
	1	Used only in SBI mode. IRQCSI is set only when the address received after bus release matches the slave address register data (wake-up status). SBO/SB1 is high impedance.

NOTE: If WUP = 1 is set during $\overline{\text{BUSY}}$ signal output, $\overline{\text{BUSY}}$ is not released. With the SBI, the $\overline{\text{BUSY}}$ signal is output after the $\overline{\text{BUSY}}$ release directive until the next fall of the serial clock ($\overline{\text{SCK}}$). When setting WUP = 1, it is necessary to confirm that the SBO (or SB1) pin has been driven high after $\overline{\text{BUSY}}$ is released before setting WUP = 1.

Signal from address comparator (R)

	Clearing Condition (COI = 0)	Setting Condition (COI = 1)
COI*	When slave address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

*: A COI read is valid only before the start of ate completion of a serial transfer. During a transfer an indeterminate value will be read. Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI0/SB1 Pins
CSIE	0	Shift operation disabled	Cleared	Retained	Port 0 function only
	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

Remarks 1: The operating mode can be selected according to the setting of CSIE, CSIM3, and CSIM2.

CSIE	CSIM3	CSIM2	Operating Mode
0	x	x	Operation-halted mode
1	0	x	3-wire serial I/O mode
1	1	0	SBI mode
1	1	1	2-wire serial I/O mode

x : don't care

2: The P01/$\overline{SCK}$ pin status depends on the setting of CSIE, CSIM1 and CSIM0 as shown below.

CSIE	CSIM1	CSIM0	P01/$\overline{SCK}$ Pin Status
0	0	0	Input port
0	0	1	High-level output
0	1	0	
0	1	1	
1	0	0	High impedance
1	0	1	Serial clock output (high-level output)
1	1	0	
1	1	1	

3: The following procedure should be used to clear CSIE during a serial transfer.

① Clear the interrupt enable flag (IECSI) to set the interrupt disabled state.

② Clear CSIE.

③ Clear the interrupt request flag (IRQCSI).

Example 1: This example selects $f_X/2^4$ as the serial clock, generates an IRQCSI serial interrupt at the end of each serial transfer, and selects the mode in which serial transfers are performed in the SBI mode with the SB0 pin as the serial data bus.

```
SEL  MB15            ; Or CLR1 MBE
MOV  XA, #10001010B
MOV  CSIM, XA        ; CSIM ← 10001010B
```

2: To enable serial transfers in accordance with the contents of CSIM.

```
SEL  MB15            ; Or CLR1 MBE
SET1 CSIE
```

(2) Serial bus interface control register (SBIC)

The format of the serial bus interface control register (SBIC) is shown in Figure 5-34.

SBIC is an 8-bit register composed of bits which control the serial bus and flags which indicate various statuses of the input data form the serial bus, and is mainly used in the SBI mode.

SBIC is manipulated by bit-manipulation instructions; it cannot be manipulated by 4-bit or 8-bit memory manipulation instructions.

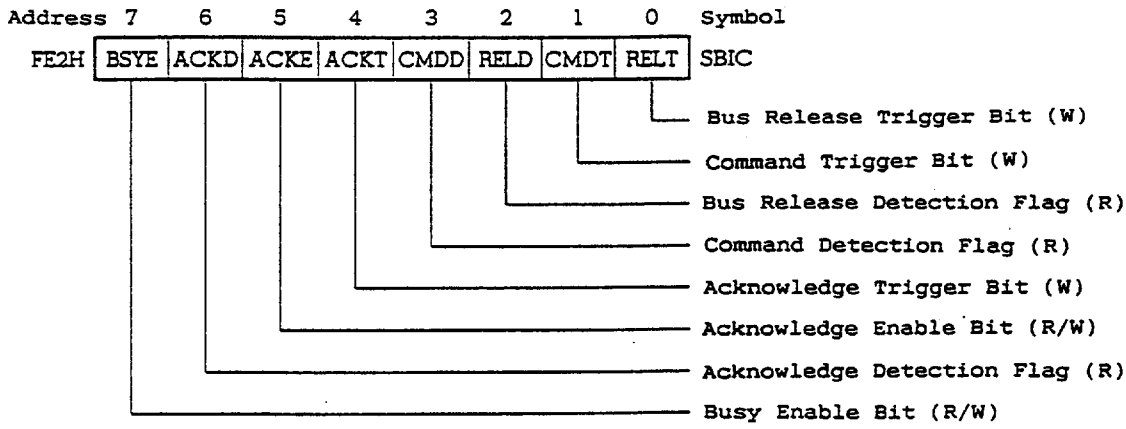
Read/write capability differs from bit to bit (see Figure 5-34).

Reset input clears this register to 00H.

NOTE: In the 3-wire and 2-wire serial I/O modes, only the following bits can be used:

- Bus release trigger bit (RELT)
  ... SO latch setting
- command trigger bit (CMDT)
  ... SO latch clearing

Figure 5-34 Serial Bus Interface Control Register (SBIC) Format



Remarks: (R) : Read only
(W) : Write only
(R/W): Read/write enabled

Figure 5-34 Serial Bus Interface Control Register (SBIC) Format (cont'd)

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).

NOTE: SBO (or SB1) must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Command trigger bit (W)

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).

NOTE: SBO (or SB1) must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Bus release detection flag (R)

RELD	Clearing Conditions (RELD = 0)	Setting Condition (RELD = 1)
	① When a transfer start instruction is executed ② When RESET is input ③ When CSIE = 0 (see Figure 5-33) ④ When SVA and SIO match when an address is received	When the bus release signal (REL) is detected

Figure 5-34 Serial Bus Interface Control Register (SBIC) Format (cont'd)

Command detection flag (R)

CMDD	Clearing Conditions (CMDD = 0)	Setting Condition (CMDD = 1)
	① When a transfer start instruction is executed ② When the bus release signal (REL) is detected ③ When RESET is input ④ When CSIE = 0 (see figure 5-33)	When the command signal (CMD) is detected

Acknowledge trigger bit (W)

ACKT	When ACKT is set after the end of a transfer, $\overline{ACK}$ is output in synchronization with the next $\overline{SCK}$. After the $\overline{ACK}$ signal is output, ACKT is automatically cleared (0).

NOTE 1: ACKT must not be set (1) before completion of a serial transfer or during a transfer.

2: ACKT cannot be cleared by software.

3: When ACKT is set, ACKE should be reset to 0.

Acknowledge enable bit (R/W)

ACKE			
	0	Disables automatic output of the acknowledge signal ($\overline{ACK}$) (output by ACKT is possible).	
	1	When set before end of transfer	$\overline{ACK}$ is output in synchronization with the 9th $\overline{SCK}$ clock cycle.
		When set after end of transfer	$\overline{ACK}$ is output in synchronization with $\overline{SCK}$ immediately after execution of the setting instruction.

Figure 5-34 Serial Bus Interface Control Register (SBIC) Format (cont'd)

Acknowledge detection flag (R)

ACKD	Clearing Conditions (ACKD = 0)	Setting Condition (ACKD = 1)
	① When transfer starts ② When RESET is input	When the acknowledge signal ($\overline{ACK}$) is detected (Synchronized with the rise of $\overline{SCK}$)

Busy enable bit (R/W)

BSYE		
	0	① Disabling of automatic busy signal output ② Busy signal output is stopped in synchronization with the fall of $\overline{SCK}$ immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of $\overline{SCK}$ following the acknowledge signal.

Example 1: To output the command signal.

```
SEL  MB15 ; Or CLR1 MBE
SET1 CMDT
```

2: To test RELD and CMDD, and perform different processing according to the type of receive data.
This interrupt routine is only performed when WUP = 1 and there is an address match.

```
        SEL   MB15
        SKF   RELD ; RELD test
        BR    !ADRS
        SKT   CMDD ; CMDD test
        BR    !DATA
CMD :     ....    ; Command interpret
DATA:     ....    ; Data processing
ADRS:     ....    ; Address decode
```

(3) Shift register (SIO)

The configuration around the shift register is shown in Figure 5-35. SIO is an 8bit register which carries out parallel-to-serial conversion and performs serial transmission/reception (shift operations) in synchronization with the serial clock.

A serial transfer is started by writing data to SIO.

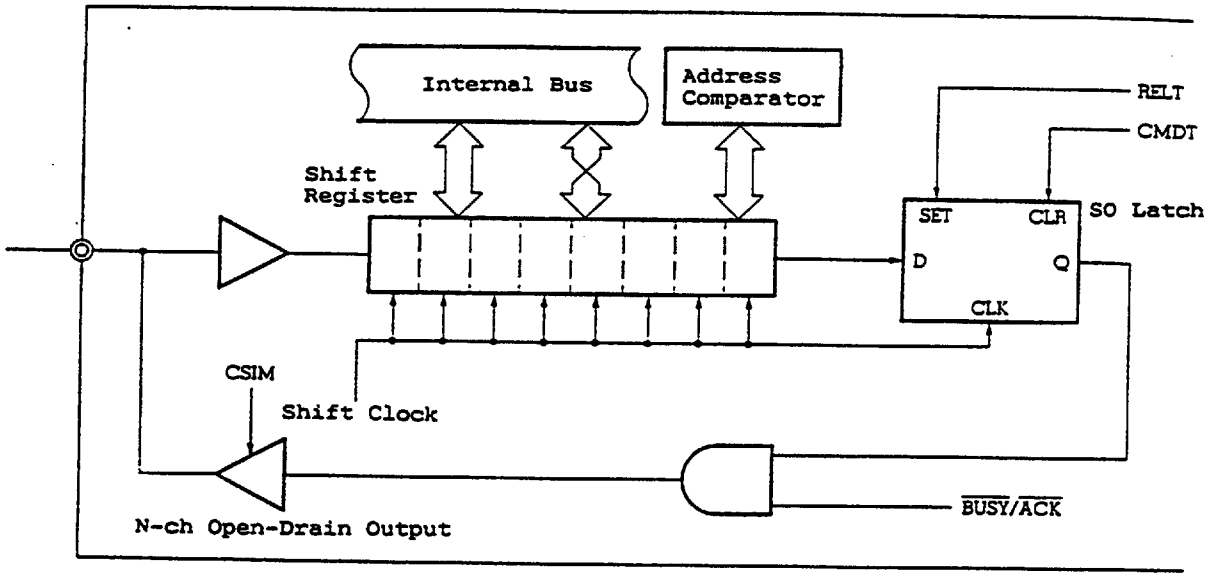
In transmission, the data written to SIO is output to the serial output (SO) or the serial data bus (SB0/SB1). In reception, data is read into SIO form the serial input (SI) or SB0/SB1.

SIO can be read or written to by an 8-bit manipulation instruction.

If RESET is input during its operation, the value of SIO is indeterminate. If RESET is input in the standby mode, the value of SIO is retained.

The shift operation stops after transmission/reception of 8 bits.

Figure 5-35 Configuration around Shift Register



SIO reading and the start of a serial transfer (write) are possible at the following times:

- When the serial interface enable/disable bit (CSIE) = 1, except when CSIE is set to "1" after data has been written into the shift register.
- When the serial clock has been masked after an 8-bit serial transfer.
- When $\overline{SCK}$ is high.

Ensure that $\overline{SCK}$ is high when data is written to or read from the SIO register.

In the 2-wire serial I/O mode and SBI mode data bus configuration, input pins and output pins have dual functions. Output pins have an N-ch open-drain configuration. Therefore, in a device in which reception is to be performed henceforth FFH should be set in the SIO register.

(4) Slave address register (SVA)

SVA is an 8-bit register used by the slave to set the slave address value (its own specification number).

It is a write-only register which is manipulated by an 8-bit manipulation instruction.

After RESET signal input, the value of SVA is indeterminate. However, when RESET is input in the standby mode, the value of SVA is retained.

The two functions of the SVA register are described below.

(a) Slave address detection (SBI mode)

Used when the uPD75336 is connected to the serial bus as a slave device.

The master outputs to its connected slaves a slave address to select a specific slave. If these two data items (the slave address output from the master and the SVA value) are found to match when compared by the address comparator, the relevant slave is determined to have been selected.

At this time, bit 6 (COI) of the serial operating mode register (CSIM) is set to "1".

When an address is received the bus release detection flag (RELD) is cleared (0) if a match is not detected. IRQCSI is set only when a match is detected when WUP = 1. This interrupt request indicates that a communication request has been issued from the master to the uPD75336.

(b) Error detection (2-wire serial I/O mode or SBI mode)

The SVA performs error detection in the following cases:

- When the uPD75336 transmits addresses, commands or data as the master device.
- When the uPD75336 receives data as a slave device.

See 5.6.6 (6) or 5.6.7 (8) "Error detection" for details.

5.6.4 OPERATION-HALTED MODE

The operation-halted mode is used when no serial transfer is performed, allowing power dissipation to be reduced.

In this mode, the shift register does not perform shift operations and can be used as an ordinary 8-bit register.

When the RESET signal is input the operation-halted mode is set. The P02/SO/SB0 and P03/SI/SB1 pins are fixed as input ports. P01/$\overline{SCK}$ can be used as an input port depending on the setting of the serial operating mode register.

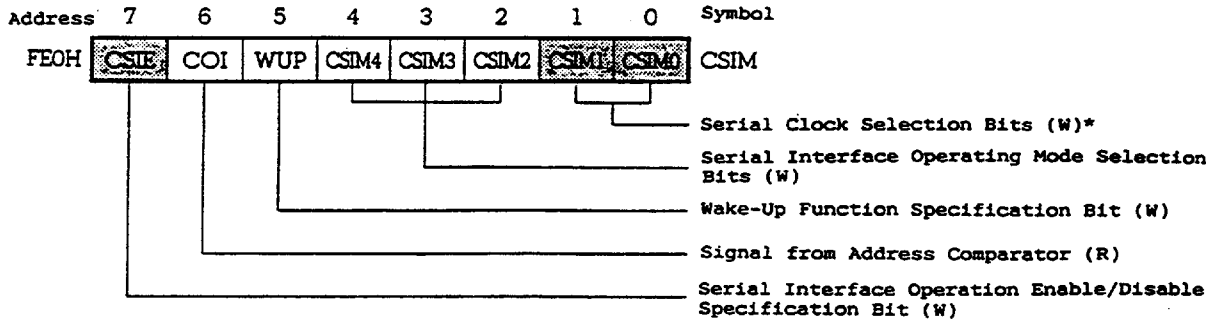
[Register setting]

Operation-halted mode setting is performed by the serial operating mode register (CSIM) (see 5.6.3 (1) "Serial operating mode register" for the contents of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions, but bit manipulation of CSIE is also possible.

Reset input clears this CSIM register to 00H.

■■■■ indicates bits used in the operation-halted mode.



*: Allow selection of P01/$\overline{SCK}$ pin status.

Remarks: (R): Read only
(W): Write only

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI/SB1 Pins
CSIE	0	Shift operation disabled	Cleared	Retained	Port 0 function only

Serial clock selection bits (W)

The P01/$\overline{SCK}$ pin status depends on the CSIM0 and CSIM1 settings as shown below.

CSIM1	CSIM0	P01/$\overline{SCK}$ Pin Status
0	0	High impedance
0	1	High level
1	0	
1	1	

The following procedure should be used to clear CSIE during a serial transfer.

① Clear the interrupt enable flag (IECSI) to set the interrupt disabled state.

② Clear CSIE.

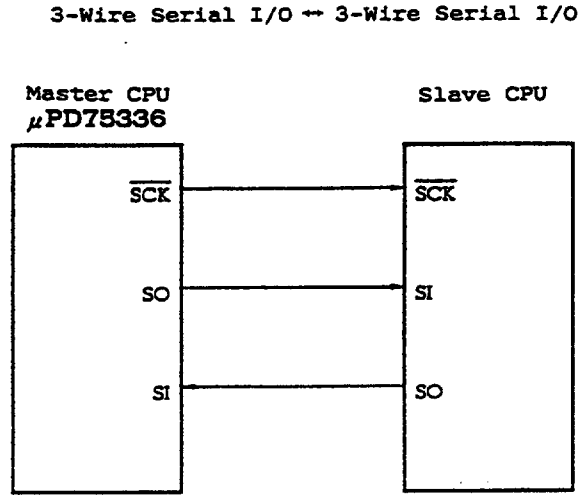
③ Clear the interrupt request flag (IRQCSI).

5.6.5 3-WIRE SERIAL I/O MODE OPERATION

The 3-wire serial I/O mode allows connection to the system used in the 75X series, uPD7500 series, 78K series, etc.

Communication is performed using three lines: The serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

Figure 5-36 Example of 3-Wire Serial I/O System Configuration



Remarks: A uPD75336 can also be used as the slave CPU.

(1) Register setting

When the device is used in the 3-wire serial I/O mode setting of the following two registers is used:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

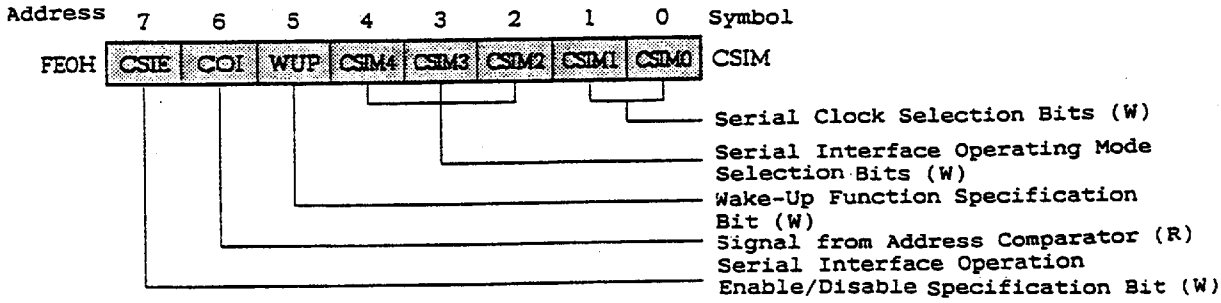
(a) Serial operating mode register (CSIM)

When the 3-wire serial I/O mode is used, CSIM is set as shown below (see 5.6.3 (1) "Serial operating mode register" for the contents of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shaded area indicates bits used in the 3-wire serial I/O mode.



Remarks: (R): Read only
(W): Write only

Serial clock selection bits (W)

CSIM1	CSIM0	Serial Clock	$\overline{SCK}$ Pin Mode
0	0	Input clock to $\overline{SCK}$ pin from off chip	Input
0	1	Timer/event counter output (TO)	Output
1	0	$f_X/2^4$ (262 kHz)*	
1	1	$f_X/2^3$ (524 kHz)*	

*: ( ): When f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Shift Register Bit Order	SO Pin Function	SI Pin Function
x	0	0	SIO $_{7 \text{ to } 0}$ ↔ XA (MSB-first transfer)	SO/P02 (CMOS output)	SI/P03 (Input)
		1	SIO $_{0 \text{ to } 7}$ ↔ XA (LSB-first transfer)		

Remarks: x: don't care

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer.

Signal from address comparator (R)

COI*	Clearing Condition (COI = 0)	Setting Condition (COI = 1)
	When salve address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

*: A COI read is valid only before the start or after completion of a serial transfer. During a transfer an indeterminate value will be read. Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

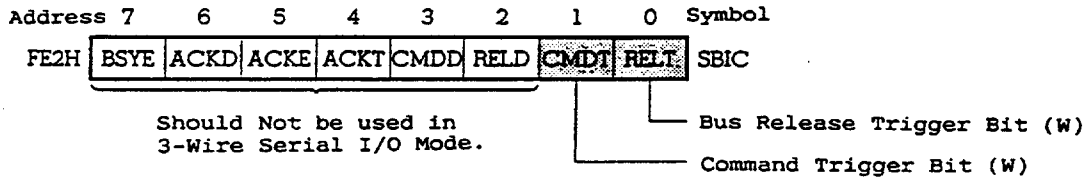
(b) Serial bus interface control register (SBIC)

When the 3-wire serial I/O mode is used, SBIC is set as shown below (see 5.6.3 (2) "Serial bus interface control register" for the contents of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

Shaded area indicates bit s used in the 3-wire serial I/O mode.



Remarks: (W): Write only

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).

Command trigger bit

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).

NOTE: Bits other than RELT and CMDT should not be used in the 3-wire serial I/O mode.

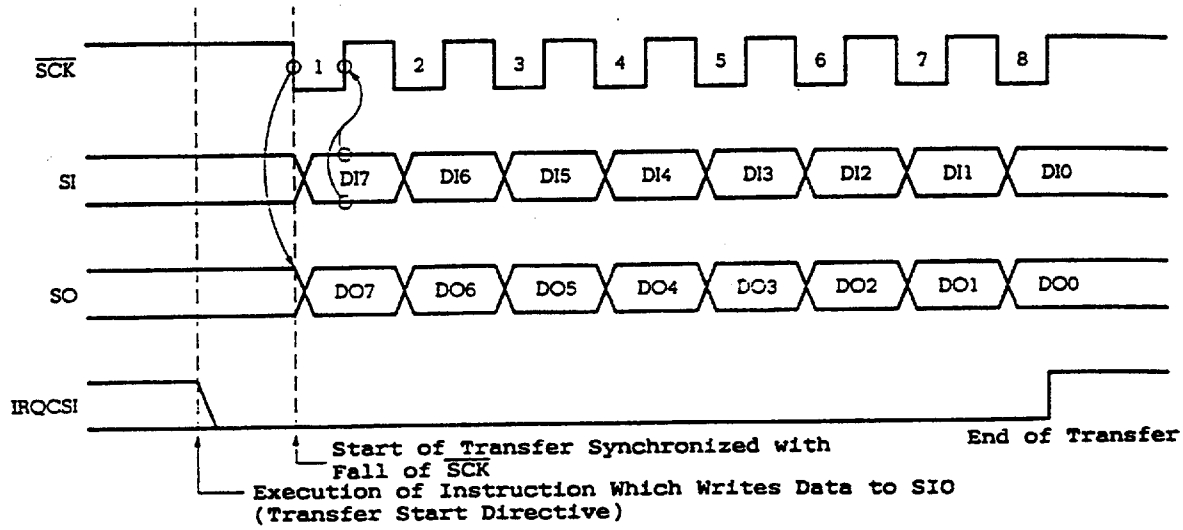
(2) Communication operation

In the 3-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in synchronization with the serial clock.

Shift register shift operations are performed in synchronization with the fall of the serial clock ($\overline{SCK}$). Then send data is held in the SO latch and output for the SO pin. Also, receive data input to the SI pin is latched in the shift register on the rise of $\overline{SCK}$.

At the end of an 8-bit transfer the operation of the shift register stops automatically and the IRQCSI interrupt request flag is set.

Figure 5-37 3-Wire Serial I/O Mode Timing



The SO pin becomes a CMOS output and outputs the SO latch status, and thus the SO pin output status can be manipulated in accordance with the setting of the RELT bit and CMDT bit.

However, manipulation should not be performed during a serial transfer.

The $\overline{SCK}$ pin output level is controlled by manipulating the P01 output latch in the output mode (internal system clock mode) (see 5.6.8 "$\overline{SCK}$ PIN OUTPUT MANIPULATION").

(3) Serial clock selection

Serial clock selection is performed by setting bits 0 and 1 of the serial operating mode register (CSIM). Any of the following 4 clocks can be selected.

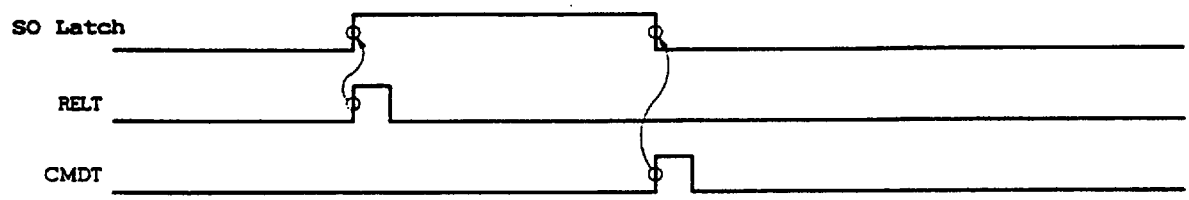
Table 5-8 Serial Clock Selection and Use (In 3-Wire Serial I/O Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial	Use
CSIM 1	CSIM 0	Source	Serial Clock Mask		
0	0	External $\overline{SCK}$	Automatically masked at end of 8-bit data transfer.	① In operation-halted mode (CSIE = 0) ② When serial clock is masked after end of 8-bit serial transfer ③ When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Half-duplex asynchronous transfer (software control)
1	0	$f_X/2^4$			Medium-speed serial transfer (software control)
1	1	$f_X/2^3$			High-speed serial transfer

(4) Signals

RELT and CMDT operation is shown in Figure 5-38.

Figure 5-38 RELT & CMDT Operation



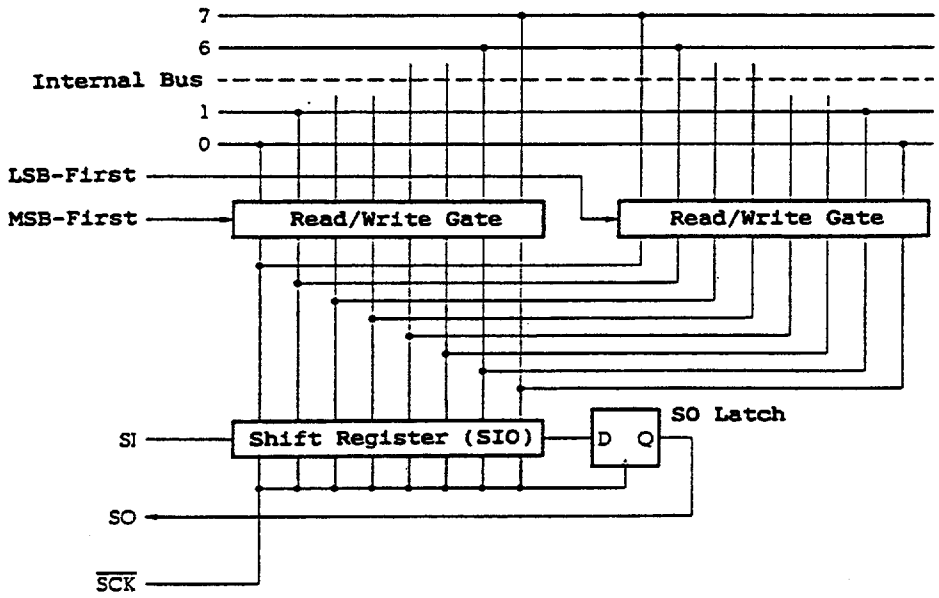
(5) MSB/LSB-first switching

The 3-wire serial I/O mode includes a function for selecting MSB-first or LSB-first transfer.

Figure 5-39 shows the shift register (SIO) and internal bus configuration. As shown in Figure 5-39, reading/writing can be performed by inverting the MSB/LSB.

MSB/LSB-first switching can be specified by bit 2 of the serial operating mode register (CSIM).

Figure 5-39 Transfer Bit Switching Circuit



Start bit switchover is implemented by switching the order in which data bits are written to the shift register (SIO). The SIO shift order is always the same.

Therefore, MSB/LSB start bit switching must be performed before writing data to the shift register.

(6) Start of transfer

When the following two conditions are met a serial transfer is started by setting transfer data in the shift register (SIO).

- The serial interface operation enable/disable bit (CSIE) = 1.
- After an 8-bit serial transfer, the internal serial clock is stopped or $\overline{SCK}$ is high.

NOTE: The transfer will not be started if CSIE is set to "1" after data is written into the shift register.

When an 8-bit transfer ends, the serial transfer stops automatically and the IRQCSI interrupt request flag is set.

Example: In the following example the data in the RAM specified by the HL register is transferred to SIO, and at the same time the SIO data is fetched into the accumulator and the serial transfer is started.

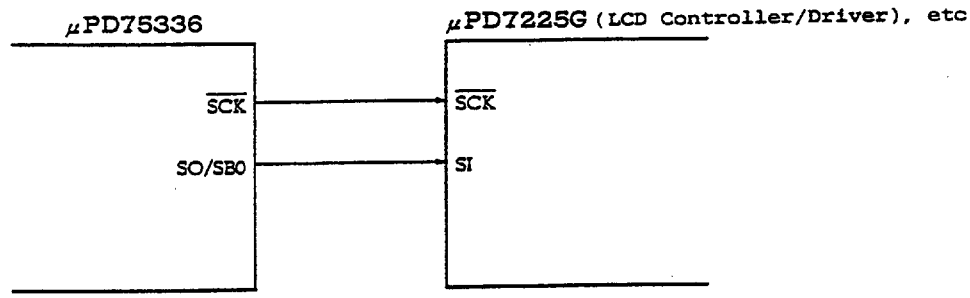
```
MOV  XA,  @HL; Fetch send data from RAM
SEL  MB15    ; Or CLR1 MBE
XCH  XA,  SIO; Exchange send data with
               receive data and start
               transfer
```

(7) 3-wire serial I/O mode applications

Example 1: To transfer data MSB-first (master operation) using a 262 kHz transfer clock (when operating at 4.19 MHz).
<Sample program>

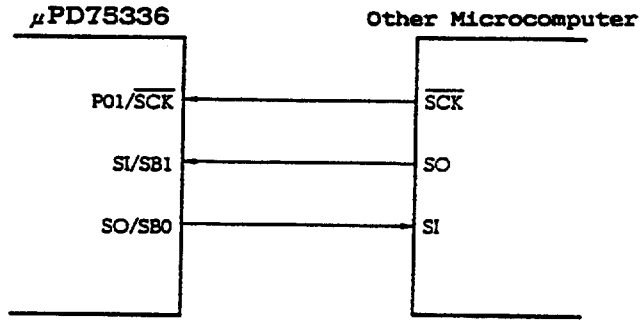
```
CLR1  MBE
MOV   XA, #10000010B
MOV   CSIM, XA        ; Transfer mode
                        setting
MOV   XA, TDATA       ; TDATA is transfer
                        data storage
                        address
MOV   SIO, XA         ; Transfer data
                        setting & start of
                        transfer
```

NOTE: From the second time onward, the transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA, SIO).



In this application the uPD75336 SI/SB1 pin can be for input.

Example 2: To transmit/receive LSB-first data using an external clock (slave operation). (In this case the function for reversing the MSB and LSB in shift register read/write operation is used.)



<Sample program>

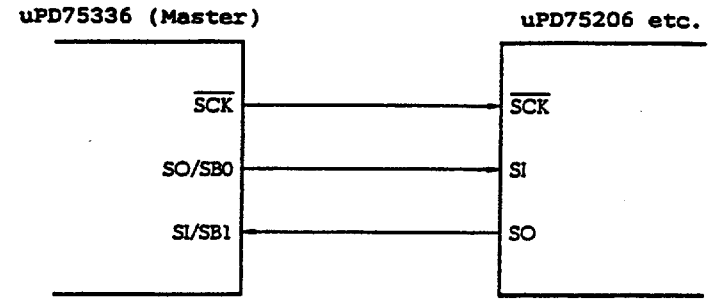
Main Routine

```
CLR1   MBE
MOV    XA, #84H
MOV    CSIM, XA ; Serial operation stopped,
                  MSB/LSB inversion mode,
                  external clock
MOV    XA, TDATA
MOV    SIO, XA  ; Transfer data setting &
                  start of transfer
EI     IECSI
EI
```

Interrupt Routine (MBE = 0)

```
MOV    XA, TDATA
XCH    XA, SIO  ; Receive data setting &
                  start of transfer
MOV    RDATA, XA; Receive data save
RETI
```

Example 3: To transmit/receive data at high speed using a 524 kHz transfer clock (when operating at 4.19 MHz).



<Program example> ··· Master side

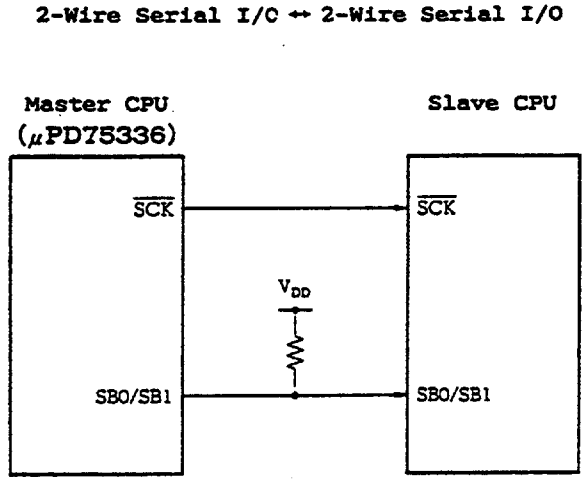
```
        CLR1    MBE
        MOV     XA, #10000011B
        MOV     CSIM, XA        ; Transfer
                                  mode
                                  setting
        MOV     XA, TDATA
        MOV     SIO, XA         ; Transfer
                                  data
                                  setting &
                                  start of
                                  transfer
         :
         :
         :
LOOP:   SKTCLR  IRQCSI          ; IRQCSI
                                  test
        BR      LOOP
        MOV     XA, SIO         ; Receive
                                  data
```

5.6.6 2-WIRE SERIAL I/O MODE OPERATION

The 2-wire serial I/O mode allows adaptation by means of the program to any communication format.

Communication is basically performed using two lines: The serial clock ($\overline{SCK}$) and serial data input/output (SB0 or SB1).

Figure 5-40 Example of 2-Wire Serial I/O System Configuration



Remarks: A uPD75336 can also be used as the slave CPU.

(1) Register setting

When the device is used in the 2-wire serial I/O mode setting of the following two registers is used:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

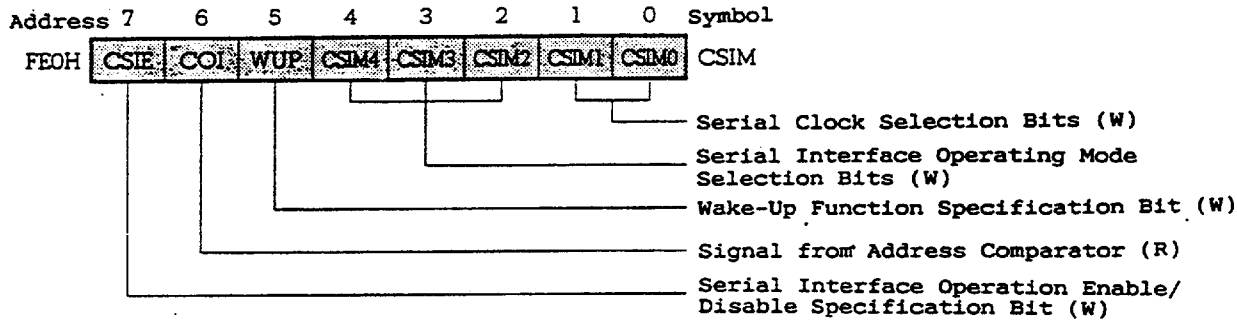
(a) Serial operating mode register (CSIM)

When the 2-wire serial I/O mode is used, CSIM is set as shown below (see 5.6.3 (1) "Serial operating mode register" for the contents of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shaded area indicates bits used in the 2-wire serial I/O mode.



Remarks: (R): Read only
(W): Write only

Serial clock selection bits (W)

CSIM1	CSIM0	Serial Clock	$\overline{SCK}$ Pin Mode
0	0	Input clock to $\overline{SCK}$ pin from off chip	Input
0	1	Timer/event counter output (TO)	Output
1	0	$f_X/2^6$ (65.5 kHz)*	
1	1		

*: ( ): When f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Shift Register Bit Order	SO Pin Function	SI Pin Function
0	1	1	SIO $_{7}$ to $_{0}$ ↔ XA (MSB-first transfer)	SB0/P02 (N-ch open-drain input/output)	P03 input
1				P02 input	SB1/P03 (N-ch open-drain input/output)

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer.

Signal from address comparator (R)

COI*	Clearing Condition (COI = 0)	Setting Condition (COI = 1)
	When salve address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

*: A COI read is valid only before the start or after completion of a serial transfer. During a transfer an indeterminate value will be read. Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

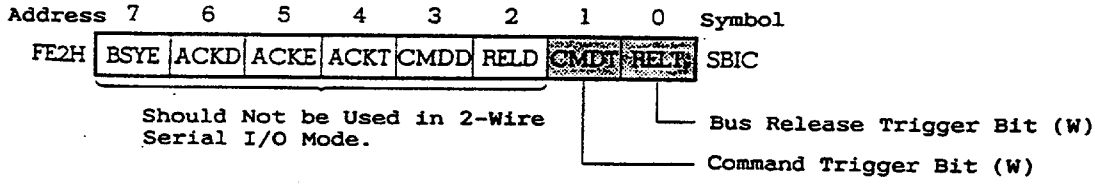
(b) Serial bus interface control register (SBIC)

When the 2-wire serial I/O mode is used, SBIC is set as shown below (see 5.6.3 (2) "Serial bus interface control register" for the contents of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

Shaded area indicates bit s used in the 2-wire serial I/O mode.



Remarks: (W): Write only

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).

Command trigger bit

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).

NOTE: Bits other than RELT and CMDT should not be used in the 2-wire serial I/O mode.

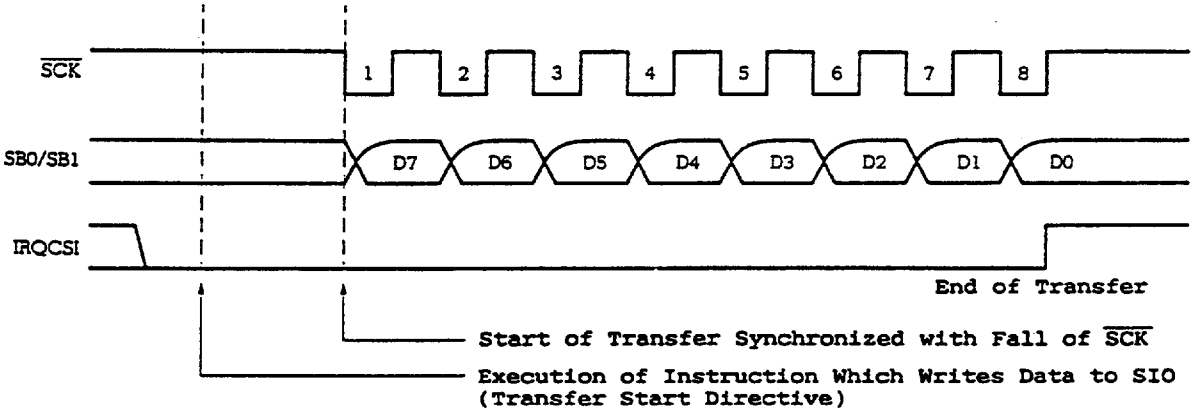
(2) Communication operation

In the 2-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in synchronization with the serial clock.

Shift register shift operations are performed in synchronization with the fall of the serial clock ($\overline{SCK}$). Then send data is held in the SO latch and output MSB-first from the SB0/P02 (or SB1/P03) pin. Also, receive data input to the SB0 (or SB1) pin is latched in the shift register on the rise of $\overline{SCK}$.

At the end of an 8-bit transfer the operation of the shift register stops automatically and the IRQCSI interrupt request flag is set.

Figure 5-41 2-Wire Serial I/O Mode Timing



Since the pin specified as the SB0 (or SB1) pin serial data bus is an N-ch open-drain input/output, it must be pulled high externally. Also, since the N-ch transistor must be turned off during data reception, FFH is written to SIO beforehand.

Since the SB0 (or SB1) pin outputs the SO latch status, the SB0 (or SB1) pin output status can be manipulated in accordance with the setting of the RELT bit and CMDT bit.

However, manipulation should not be performed during a serial transfer.

The $\overline{SCK}$ pin output level is controlled by manipulating the P01 output latch in the output mode (internal system clock mode) (see 5.6.8 "$\overline{SCK}$ Pin Output Manipulation").

(3) Serial clock selection

Serial clock selection is performed by setting bits 0 and 1 of the serial operating mode register (CSIM). Any of the following 3 clocks can be selected.

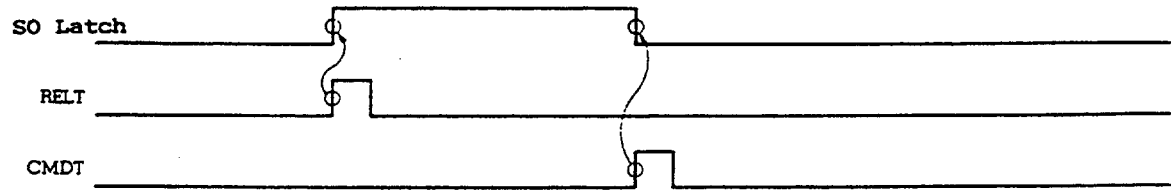
Table 5-9 Serial Clock Selection and Use (In 2-Wire Serial I/O Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial	Use
CSIM 1	CSIM 0	Source	Serial Clock Mask		
0	0	External $\overline{SCK}$	Automatically masked at end of 8-bit data transfer.	① In operation-halted mode (CSIE = 0) ② When serial clock is masked after end of 8-bit serial transfer ③ When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Arbitrary-speed serial transfer
1	0	$f_X/2^6$			Low-speed serial transfer
1	1				

(4) Signals

RELT and CMDT operation is shown in Figure 5-42.

Figure 5-42 RELT & CMDT Operation



(5) Start of transfer

When the following two conditions are met a serial transfer is started by setting transfer data in the shift register (SIO).

- The serial interface operation enable/disable bit (CSIE) = 1.
- After an 8-bit serial transfer, the internal serial clock is stopped or $\overline{SCK}$ is high.

NOTE 1: The transfer will not be started if CSIE is set to "1" after data is written into the shift register.

2: Since the N-ch transistor must be turned off during data reception, FFH should be written to SIO beforehand.

When an 8-bit transfer ends, the serial transfer stops automatically and the IRQCSI interrupt request flag is set.

(6) Error detection

In the 2-wire serial I/O mode, since the status of the serial bus SB0/SB1 pin during transmission is also written into the SIO shift register of the transmitting device, transmission errors can be detected in the following ways:

(a) Comparison of pre-transmission and post-transmission SIO data

In this case, a transmission error is judged to have been generated if the two data items are different.

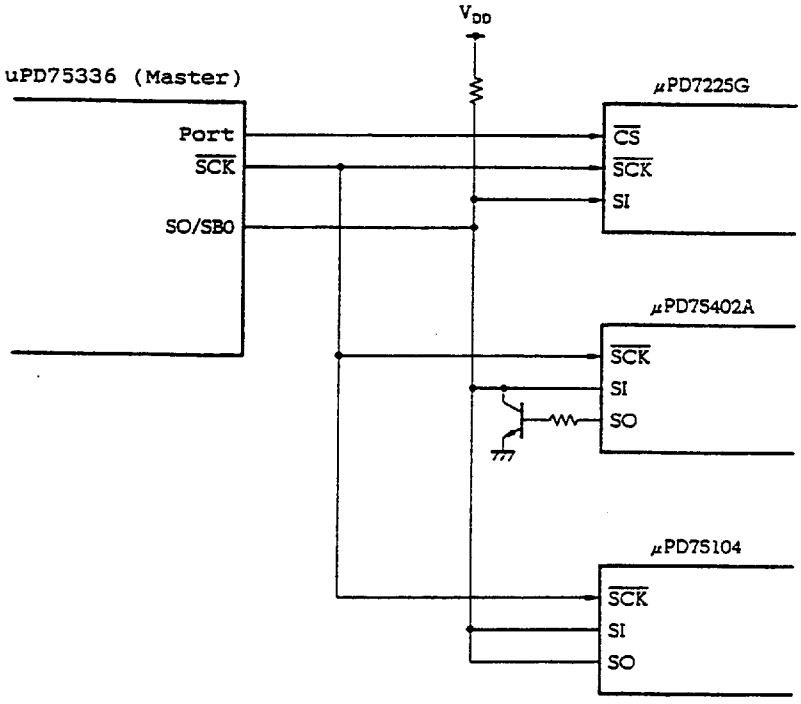
(b) Use of slave address register (SVA)

Transmission is performed after setting the send data in the SIO and SVA registers. After transmission the COI bit of the serial operating mode register (CSIM) (the match signal from the address comparator) is tested: "1" indicates normal transmission, and "0", a transmission error.

(7) 2-wire serial I/O mode applications

A serial bus is configured and multiple devices connected.

Example: Top configure a system with a uPD75336 as the master and a uPD75104, uPD75402A and uPD7225G connected as slaves.



The uPD75104 is connected via the SI pin and SO pin. When serial data is not being output the serial operating mode register is manipulated and the output buffer turned off to release the bus.

Since the uPD75402A SO pin cannot be placed in the high impedance state, it should be made an open collector output by connecting a transistor as shown in the figure. Then, when data is input the transistor is turned off by writing 00H to the shift register beforehand.

Which microcomputer outputs data when is determined in advance.

The serial clock is output by the uPD75336, which is the master microcomputer, and the other slave microcomputers all operate on an external clock.

5.6.7 SBI MODE OPERATION

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

The SBI is a single-master high-speed serial bus. Its format includes the addition of bus configuration functions to the clocked serial I/O system to enable communication to be performed with multiple devices using two signal lines. Consequently, when a serial bus is configured with multiple microcomputers and peripheral ICs, it is possible to reduce the number of ports used and the amount of wiring on the substrate.

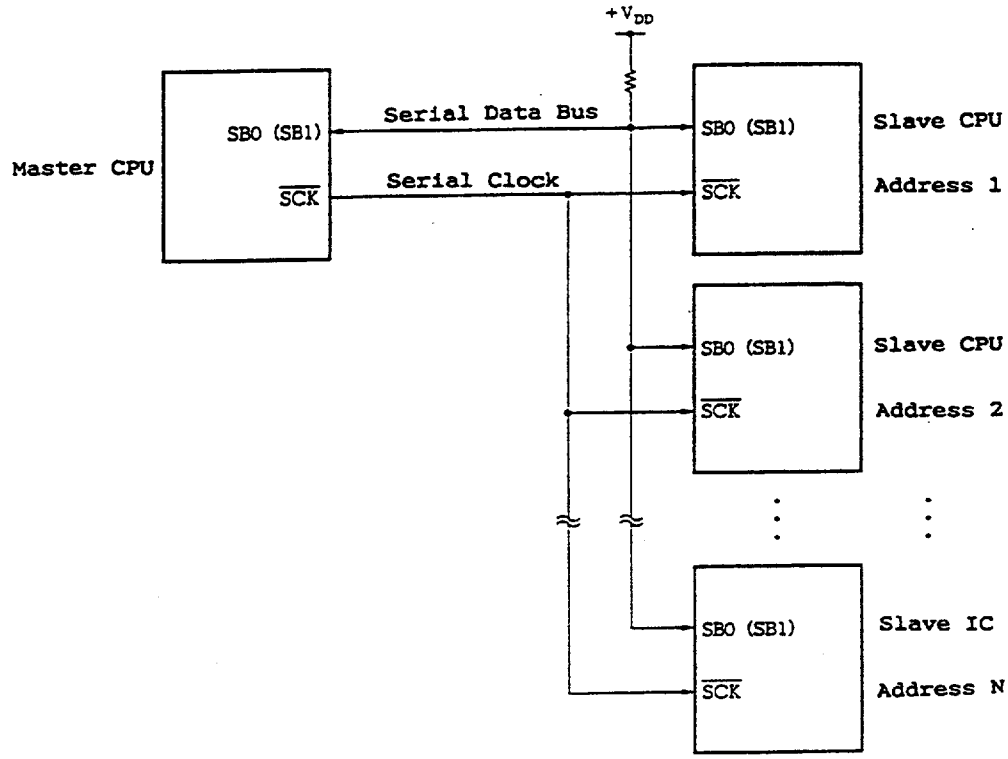
The master can output to a slave on the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. The slave can determine by hardware whether the received data is an address, command or actual data. This function allows the serial interface control portion of the application program to be simplified.

SBI functions are incorporated in number of devices including the 75X series, and 78K series 8/16-bit single-chip microcomputers.

An example of a serial bus configuration when CPUs and peripheral ICs with a serial interface conforming to the SBI are used is shown in Figure 5-43.

In the SBI the SB0 (SB1) serial data bus pin is an open-drain output and thus the data bus line is in the wired-OR state. The serial data bus line requires a pull-up resistor.

Figure 5-43 Example of SBI Serial Bus Configuration



NOTE: When master/slave exchange processing is performed, since serial clock line ($\overline{SCK}$) input/output switching is performed asynchronously between master and slave, a pull-up resistor is also required for the serial clock line ($\overline{SCK}$).

(1) SBI functions

Since conventional serial I/O methods have only data transfer functions, when a serial bus is configured with multiple devices connected a large number of ports and wires are required for Chip Select signal and command/data differentiation, busy status recognition, etc. If these controls re performed by software, the load incurred by software is very large.

With the SBI, a serial bus can be configured using only two lines: The serial clock, $\overline{SCK}$, and the serial data bus, SB0 (SB1). As a result, the number of microcomputer ports and the amount f substrate wiring can be significantly reduced.

SBI functions are described below.

(a) Address/Command/data differentiation function

Identifies serial data as an address, command or actual data.

(b) Chip selection by address

The master performs chip selection by address transmission.

(c) Wake-up function

A slave can identify address reception (chip selection) easily by means of the wake-up function (settable/releasable by software).

When the wake-up function is set, an interrupt (IRQCSI) is generated when a matching address is received.

As a result, non-selected CPUs can operate without regard to serial communications even when communication with multiple devices is performed.

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

Controls the acknowledge signal used to confirm serial data reception.

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

Controls the busy signal used to give notification of a slave busy status.

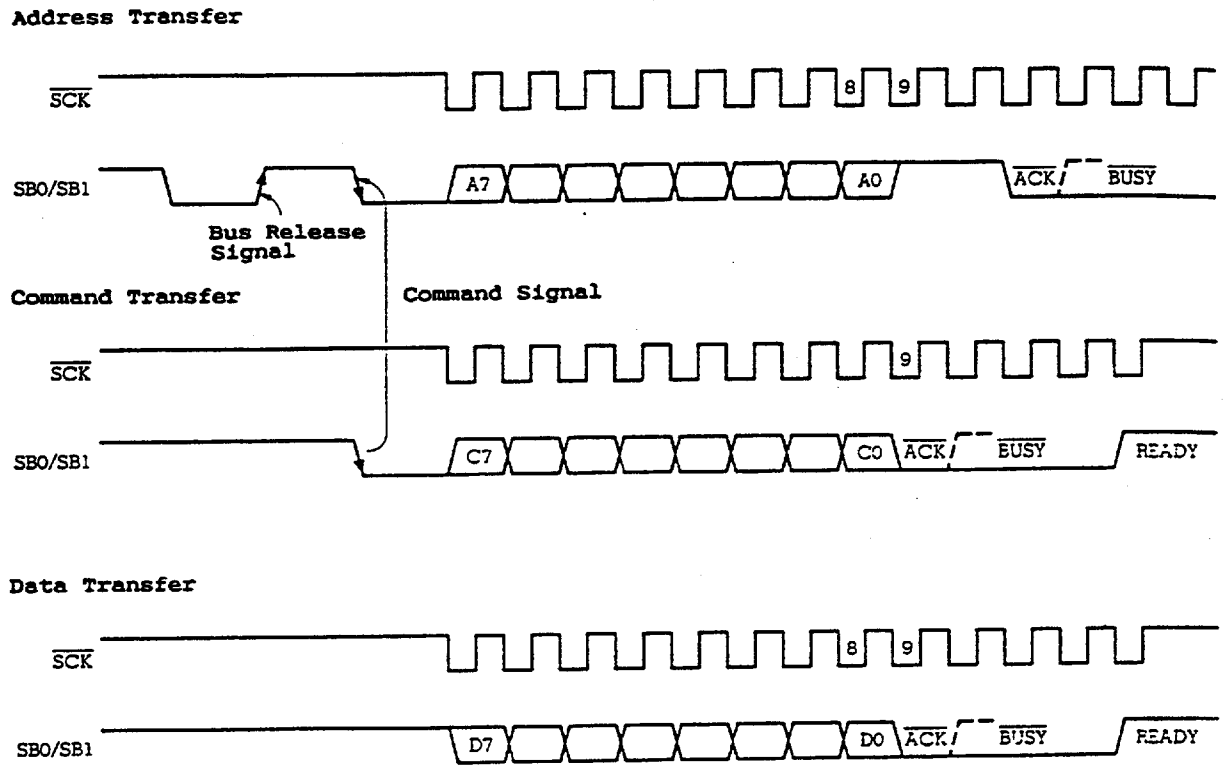
(2) SBI definition

The SBI serial data format and the meaning of the signals used are explained in the following section.

Serial data transmitted via the SBI is classified into three types: Commands, addresses and data.

Address, command and data timing is shown in Figure 5-44.

Figure 5-44 SBI Transfer Timing



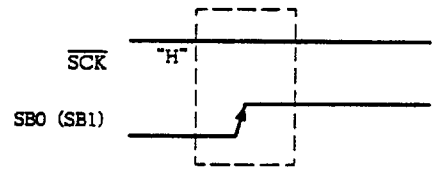
The bus release signal and command signal are output by the master. The $\overline{BUSY}$ signal is output by the slave. $\overline{ACK}$ can be output by either the master or slave (normally output by the 8-bit data receiver).

The serial clock is output by the master continuously from the start of an 8-bit data transfer until $\overline{BUSY}$ is released.

(a) Bus release signal (REL)

The bus release signal indicates that the SB0 (SB1) line has changed from low to high when the $\overline{SCK}$ line is high (when the serial clock is not being output). This signal is output by the master.

Figure 5-45 Bus Release Signal

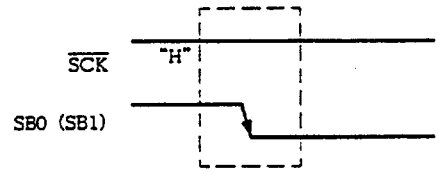


The bus release signal indicates that the master is about to send an address to a slave. Slaves incorporate hardware to detect the bus release signal.

(b) Command signal (CMD)

The command signal indicates that the SB0 (SB1) line has changed for high to low when the $\overline{SCK}$ line is high (when the serial clock is not being output). This signal is output by the master.

Figure 5-46 Command Signal

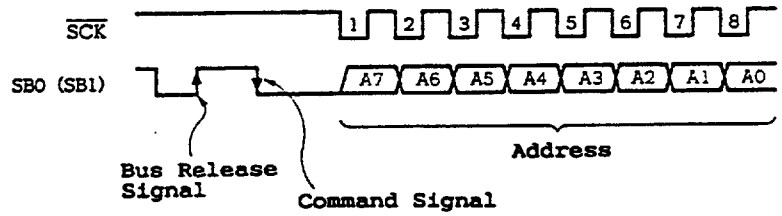


Slaves incorporate hardware to detect the bus release signal.

(c) Address

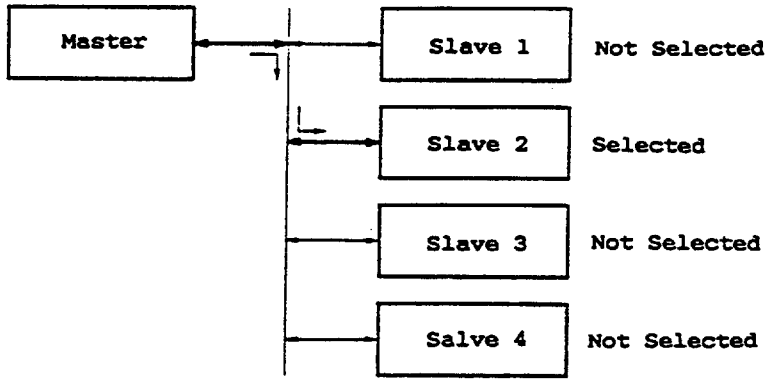
An address is 8-bit data output by the master to slaves connected to the bus line in order to select a particular slave.

Figure 5-47 Address



The 8-bit data following the bus release signal and command signal is defined as an address. In a slave this condition is detected by hardware and a check is performed by hardware to see if the 8-bit data matches the slave's own specification number (slave address). If the 8-bit data matches the slave address, that slave is determined to have been selected and communication is subsequently performed with the master until a disconnect directive is received from the master.

Figure 5-48 Slave Selection by Address



(d) Command & data

The master sends a command or data to the slave selected by address transmission.

Figure 5-49 Command

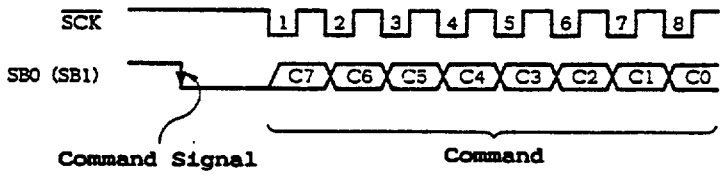
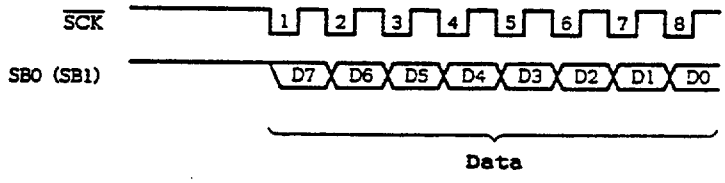


Figure 5-50 Data



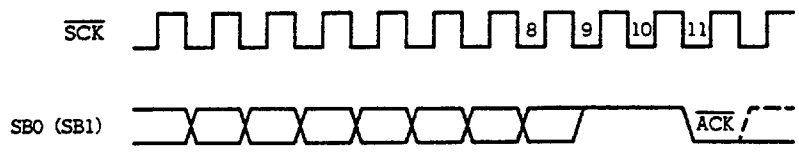
The 8-bit data following the command signal is defined as a command. 8-bit with no command signal is defined as data. The way in which commands and data are used can be freely decided according to the communication specifications.

(e) Acknowledge signal($\overline{ACK}$)

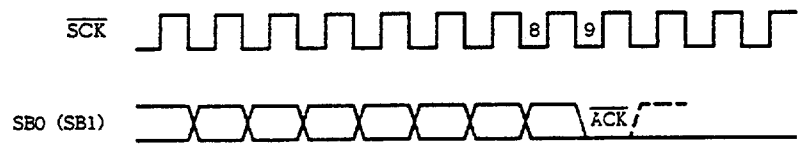
The acknowledge signal is used to confirm serial data reception between the sender and receiver.

Figure 5-51 Acknowledge Signal

[When output in synchronization with 11th $\overline{SCK}$ clock cycle]



[When output in synchronization with 9th $\overline{SCK}$ clock cycle]



The acknowledge signal is a one-shot pulse synchronized with the fall of $\overline{SCK}$ after an 8-bit data transfer. Its position is arbitrary and it can be synchronized with any $\overline{SCK}$ clock cycle.

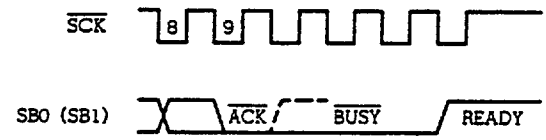
After 8-bit data transmission the sender checks whether the receiver has sent back an acknowledge signal. If an acknowledge signal is not returned within a specific time after data transmission, reception can be judged not to have been performed correctly.

(f) Busy signal ($\overline{BUSY}$), ready signal ($\overline{READY}$)

The busy signal notifies the master that a slave is preparing for data transmission/ reception.

The ready signal notifies the master that a slave is ready for data transmission/reception.

Figure 5-52 Busy Signal & Ready Signal



With the SBI a slave reports its busy status to the master by driving the SBO (SBI) line low.

The busy signal is output following the acknowledge signal output by the master or slave. Busy signal setting/release is performed in synchronization with the fall of $\overline{SCK}$. When the busy signal is released the master automatically terminates output of the $\overline{SCK}$ serial clock.

When he busy signal is released and the ready signal state is entered the master can start the next transfer.

(3) Register setting

When the device is used in the SBI mode, setting of the following two registers is used:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

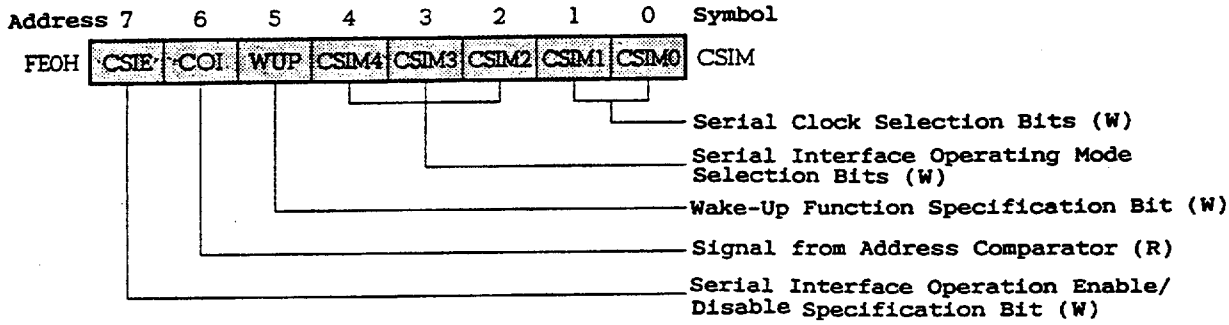
(a) Serial operating mode register (CSIM)

When the SBI mode is used, CSIM is set as shown below (see 5.6.3 (1) "Serial operating mode register" for the contents of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shaded area indicates bits used in the SBI mode.



Remarks: (R): Read only
(W): Write only

Serial clock selection bits (W)

CSIM1	CSIM0	Serial Clock	$\overline{SCK}$ Pin Mode
0	0	Input clock to $\overline{SCK}$ pin from off chip	Input
0	1	Timer/event counter output (T0)	Output
1	0	$f_X/2^4$ (262 kHz)*	
1	1	$f_X/2^3$ (524 kHz)*	

*: ( ): When f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Shift Register Bit Order	SO Pin Function	SI Pin Function
0	1	0	SIO $_{7}$ to $_{0}$ ↔ XA (MSB-first transfer)	SB0/P02 (N-ch open-drain input/output)	P03 input
1				P02 input	SB1/P03 (N-ch open-drain input/output)

Wake-up function specification bit (W)

WUP		
	0	IRQCSI set at end of every serial transfer with mask state of SBI mode.
	1	Used only slave in SBI mode. IRQCSI is set only when the address received after bus release matches the slave address register data (wake-up status). SB0/SB1 is high impedance.

NOTE: If WUP = 1 is set during $\overline{\text{BUSY}}$ signal output, $\overline{\text{BUSY}}$ is not released. With the SBI, the $\overline{\text{BUSY}}$ signal is output after the $\overline{\text{BUSY}}$ release directive until the next fall of the serial clock ($\overline{\text{SCK}}$). When setting WUP = 1, it is necessary to confirm that the SB0 (or SB1) pin has been driven high after $\overline{\text{BUSY}}$ is released before setting WUP = 1.

Signal from address comparator (R)

COI*	Clearing Condition (COI = 0)	Setting Condition (COI = 1)
	When slave address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

*: A COI read is valid only before the start or after completion of a serial transfer. During a transfer an indeterminate value will be read. Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

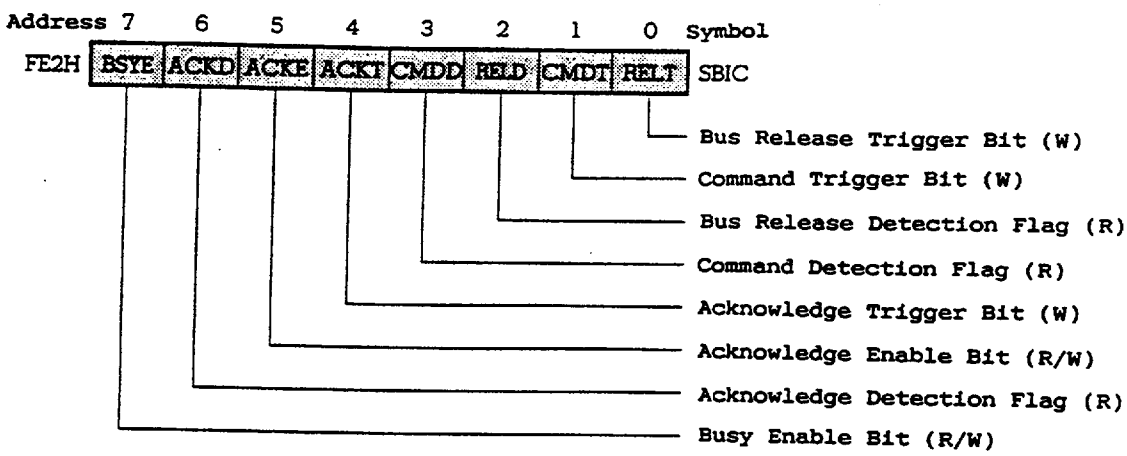
(b) Serial bus interface control register (SBIC)

When the SBI mode is used, SBIC is set as shown below (see 5.6.3 (2) "Serial bus interface control register" for the contents of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

Shaded area indicates bits used in the SBI mode.



Remarks: (R) : Read only
(W) : Write only
(R/W): Read/write enabled

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).

NOTE: SB0 (or SB1) must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Command trigger bit (W)

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).

NOTE: SB0 (or SB1) must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Bus release detection flag (R)

RELD	Clearing Conditions (RELD = 0)	Setting Condition (RELD = 1)
	① When a transfer start instruction is executed ② When RESET is input ③ When CSIE = 0 (see Figure 5-33) ④ When SVA and SIO match when an address is received	When the bus release signal (REL) is detected

Command detection flag (R)

CMDD	Clearing Conditions (CMDD = 0)	Setting Condition (CMDD = 1)
	① When a transfer start instruction is executed ② When the bus release signal (REL) is detected ③ When RESET is input ④ When CSIE = 0 (see Figure 5-33)	When the command signal (CMD) is detected

Acknowledge trigger bit (W)

ACKT	When ACKT is set after the end of a transfer, $\overline{\text{ACK}}$ is output in synchronization with the next $\overline{\text{SCK}}$. After the $\overline{\text{ACK}}$ signal is output, ACKT is automatically cleared (0).

NOTE 1: ACKT must not be set (1) before completion of a serial transfer or during a transfer.

2: ACKT cannot be cleared by software.

3: When ACKT is set, ACKE should be reset to 0.

Acknowledge enable bit (R/W)

ACKE	0	Disables automatic output of the acknowledge signal (output by ACKT is possible).	
	1	When set before end of transfer	The $\overline{ACK}$ signal is output in synchronization with the 9th $\overline{SCK}$ clock cycle.
		When set after end of transfer	The $\overline{ACK}$ signal is output in synchronization with $\overline{SCK}$ immediately after execution of the setting instruction.

Acknowledge detection flag (R)

ACKD	Clearing Conditions (ACKD = 0)	Setting Condition (ACKD = 1)
	① When transfer starts ② When RESET is input	When the acknowledge signal ($\overline{ACK}$) is detected (synchronized with the rise of $\overline{SCK}$)

Busy enable bit (R/W)

BSYE	0	① Disabling of automatic busy signal output ② Busy signal output is stopped in synchronization with the fall of $\overline{SCK}$ immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of $\overline{SCK}$ following the acknowledge signal.

(4) Serial clock selection

Serial clock selection is performed by setting bits 0 and 1 of the serial operating mode register (CSIM). Any of the following 4 clocks can be selected.

Table 5-10 Serial Clock Selection and Use (In SBI Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial	Use
CSIM 1	CSIM 0	Source	Serial Clock Mask		
0	0	External $\overline{SCK}$	Automatically masked at end of 8-bit data transfer.	① In operation-halted mode (CSIE = 0) ② When serial clock is masked after end of 8-bit serial transfer ③ When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Arbitrary-speed serial transfer
1	0	$f_X/2^4$			Medium-speed serial transfer
1	1	$f_X/2^3$			High-speed serial transfer

When the internal system clock is selected $\overline{SCK}$ stops at 8 pulses internally, bit externally the count continues until the slave is in the ready state.

(5) Signals

The operation of signals and flags in SBIC in the SBI mode are shown in Figures 5-53 to 5-58, and SBI signals are listed in Table 5-11.

Figure 5-53 RELT, CMDT, RELD & CMDD Operation (Master)

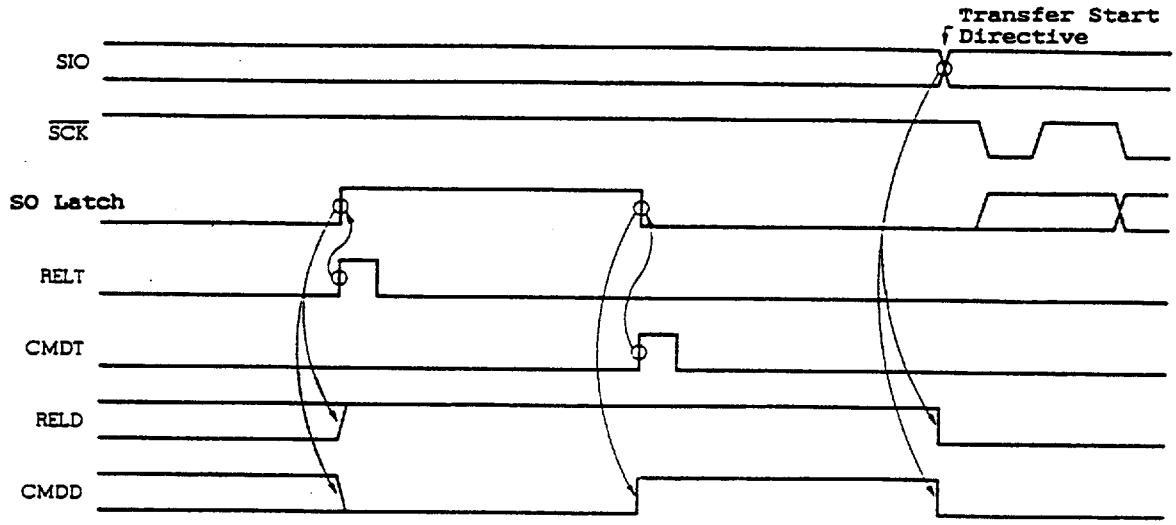


Figure 5-54 RELT, CMDT, RELD & CMDD Operation (Slave)

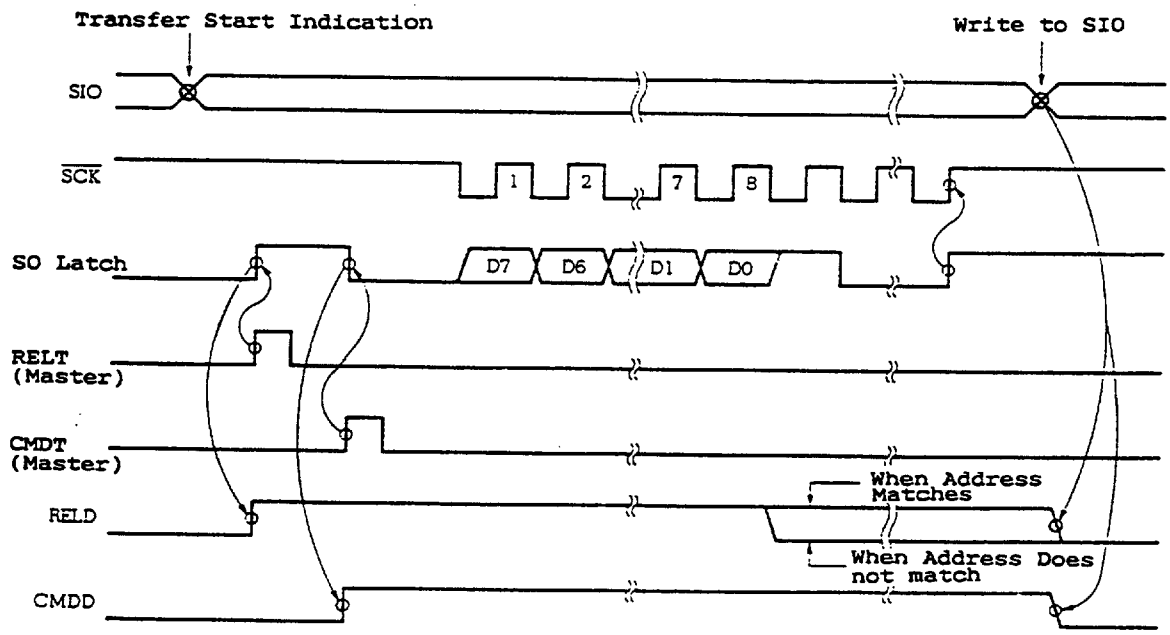
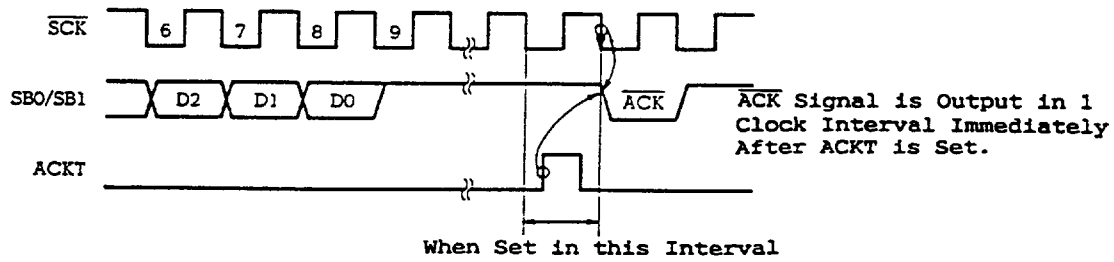


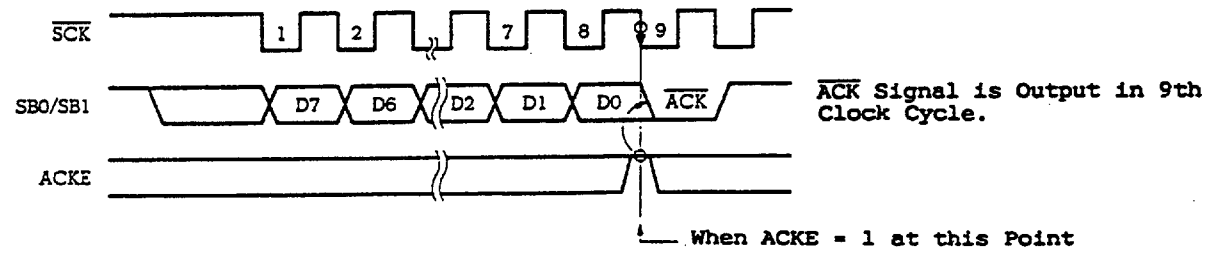
Figure 5-55 ACKT Operation



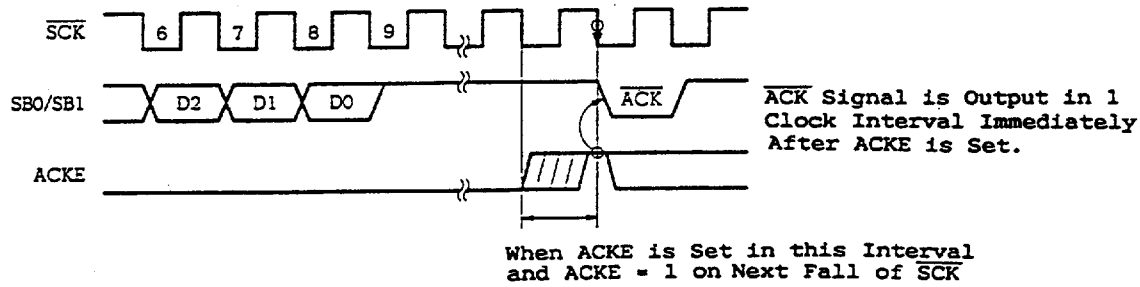
NOTE: ACKT must not be set before the end of a transfer.

Figure 5-56 ACKE Operation

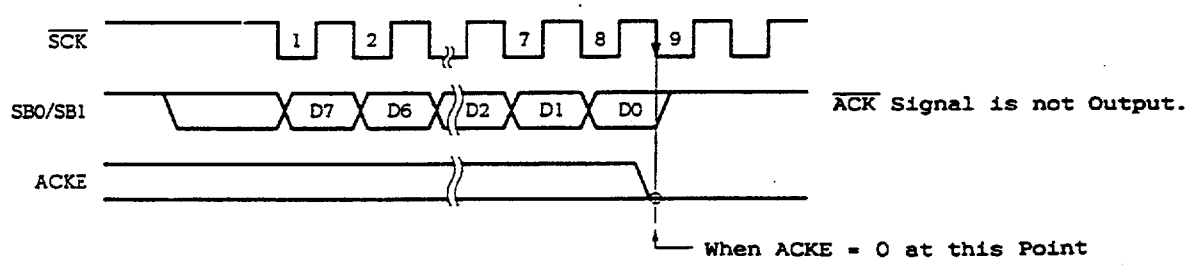
a. When ACKE = 1 on completion of transfer



b. When ACKE is set after completion of transfer



c. When ACKE = 0 on completion of transfer



d. When ACKE = 1 interval is short

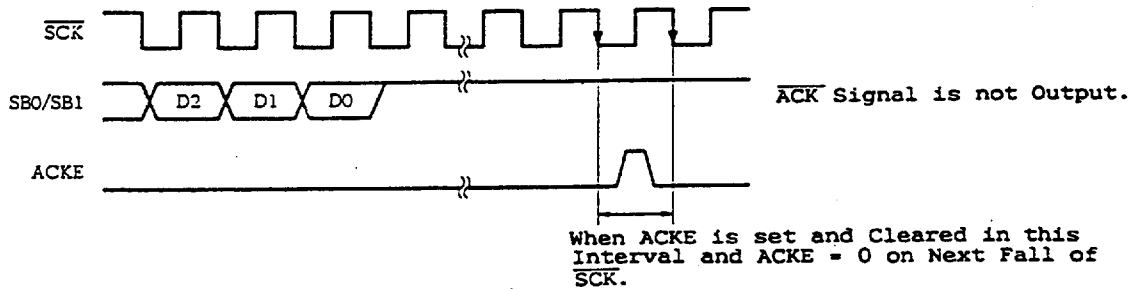
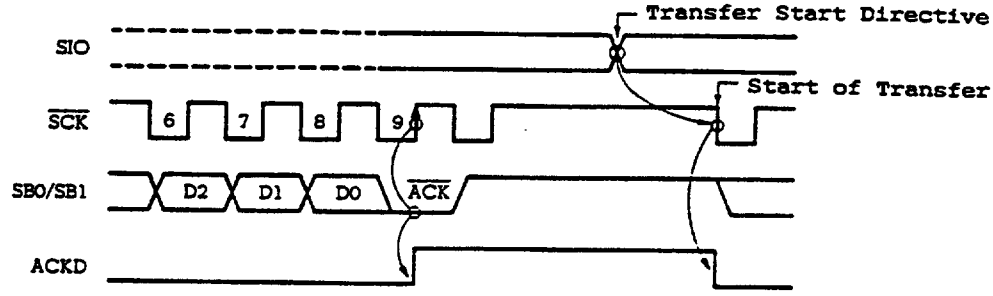
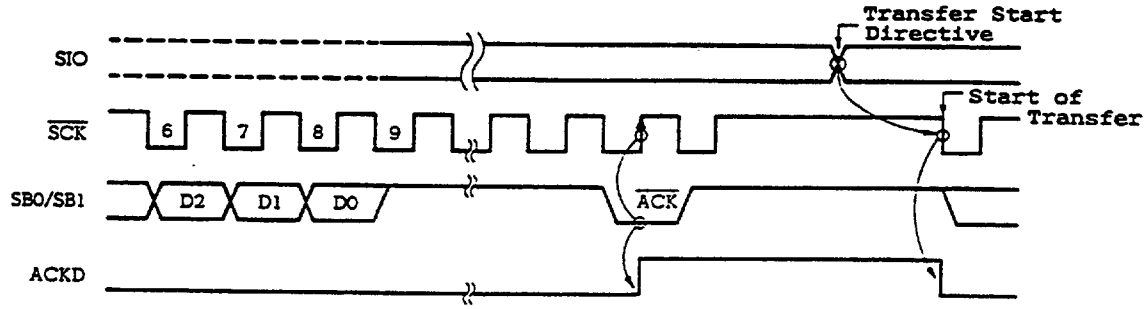


Figure 5-57 ACKD Operation

a. When $\overline{ACK}$ signal is output in 9th $\overline{SCK}$ clock interval



b. When $\overline{ACK}$ signal is output after 9th $\overline{SCK}$ clock interval



c. Clearing timing when transfer start directive is given during BUSY state

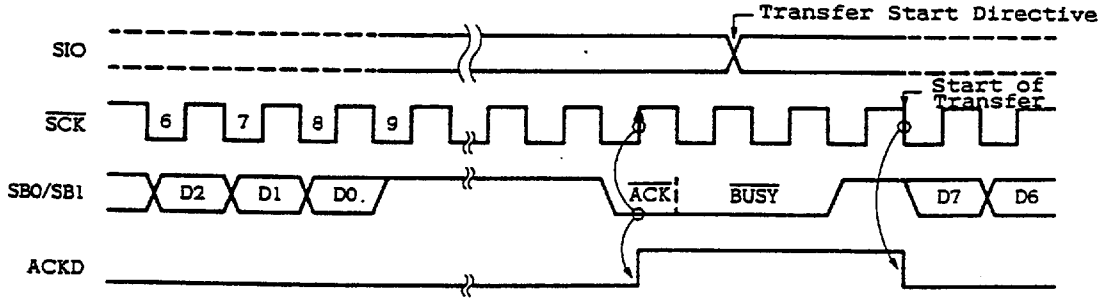


Figure 5-58 BSYE Operation

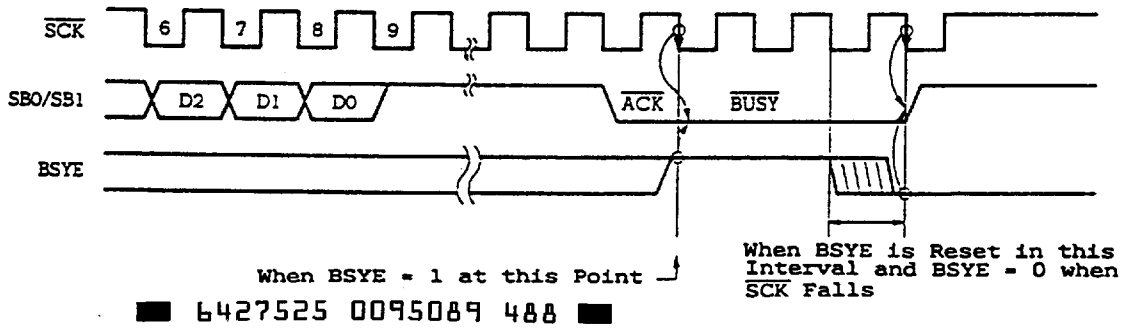
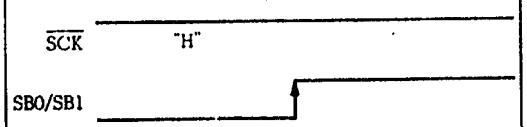
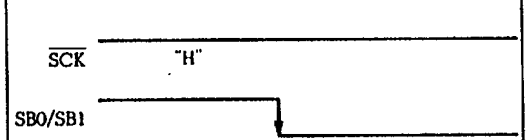
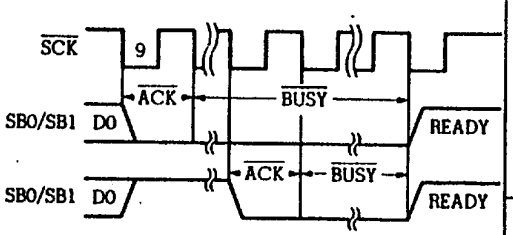


Table 5-11 Various Signals in SBI Mode

Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Effects on Flags	Meaning of Signal
Bus release signal (REL)	Master	SBO/SB1 rising edge when $\overline{SCK}$ = 1		. RELT set	. RELD set . CMDD clear	Outputs next CMD signal and indicates send data is address.
Command signal (CMD)	Master	SBO/SB1 falling edge when $\overline{SCK}$ = 1		. CMDT set	. CMDD set	i) After REL signal output send data is address. ii) Send data with no REL signal output is command.

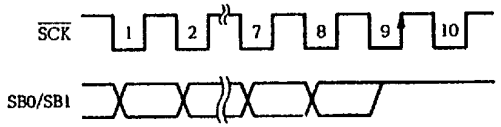
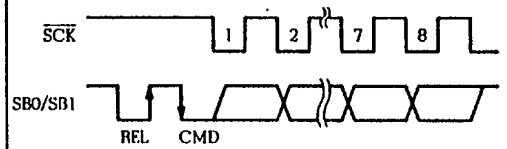
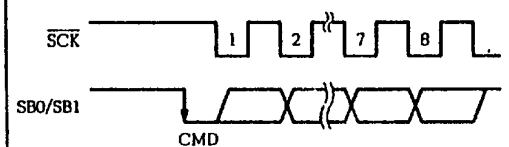
(to be continued)

Table 5-11 Various Signals in SBI Mode (cont'd)

Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Effects on Flags	Meaning of Signal
Acknowledge signal ($\overline{ACK}$)	Master/ slave	Low-level signal output to SBO/SB1 in 1 $\overline{SCK}$ clock interval after serial receive completion.	[Synchronous busy output] 	① ACKE = 1 ② ACKT set	. ACKD set	Receive completion
Busy signal ($\overline{BUSY}$)	Slave	[Synchronous busy signal] Low-level signal output to SBO/SB1 after Acknowledge signal		. BSYE = 1	—	Serial receive disabled due to processing
Ready signal (READY)	Slave	High-level signal output to SBO/SB1 before start or after completion of serial transfer		① BSYE = 0 ② Execution of instruction to write data to SIO (transfer start directive)	—	Serial reception enabled

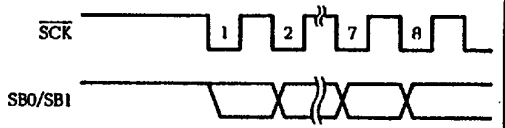
(to be continued)

Table 5-11 Various Signals in SBI Mode (cont'd)

Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Effects on Flags	Meaning of Signal
Serial clock ($\overline{SCK}$)	Master	Synchronous clock for output of address/command/data, $\overline{ACK}$ signal, synchronous $\overline{BUSY}$ signal. etc. Address/command/data is transferred in the first 8 cycles.		Execution of instruction to write to SIO when CSIE = 1 (serial transfer start directive) *2	IRQCSI set (rise of 9th $\overline{SCK}$ clock cycle) *1	Timing of signal output to serial data bus
Address (A7 to A0)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after REL signal and CMD signal output				Address value of slave device on serial bus
Command (C7 to C0)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after CMD signal only is output without output of REL signal.				Directive message to slave device

(to be continued)

Table 5-11 Various Signals in SBI Mode (cont'd)

Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Effects on Flags	Meaning of Signal
Data (D7 to D0)	Master/ slave	8-bit data transferred in synchronization with $\overline{SCK}$ with no output of either REL signal or CMD signal.		Execution of instruction to write to SIO when CSIE = 1 (serial transfer start directive) *2	IRQCSI set (rise of 9th $\overline{SCK}$ clock cycle) *1	Numeric value to be processed by slave or master device

*1: When WUP = 0, IRQCSI is always set on the 9th rise of $\overline{SCK}$.
When WUP = 1, IRQCSI is set only when an address is received and that address matches the value of the slave address register (SVA).

2: When in the $\overline{BUSY}$ state, the transfer starts after transition to the READY state.

(6) Pin configuration

The configuration of the serial clock pin ($\overline{SCK}$) and the serial data bus pin (SB0 or SB1) is as shown below.

(a) $\overline{SCK}$ ..... Pin for input/output of serial clock

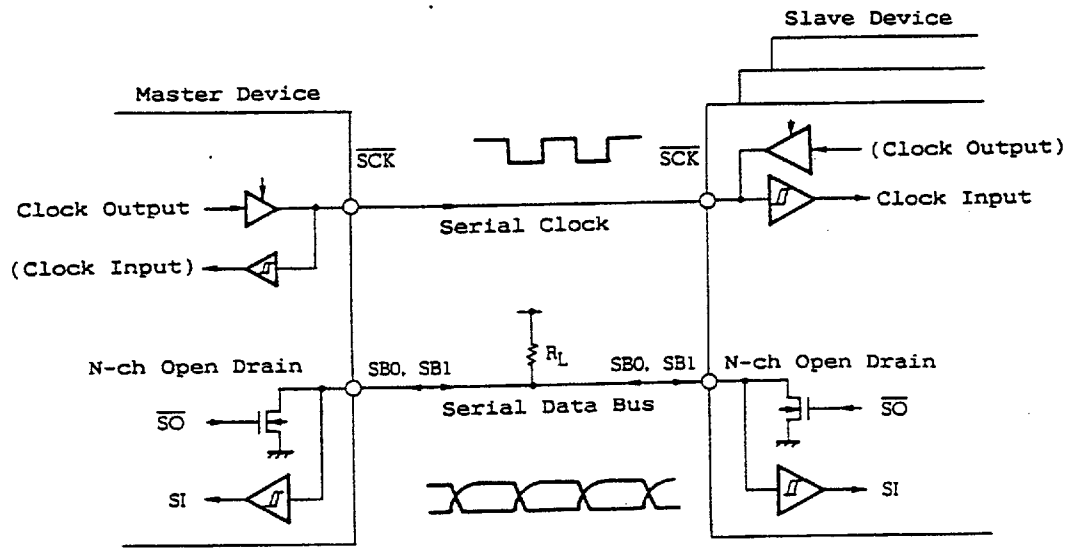
① Master ... CMOS, push-pull output

② Slave .... Schmitt input

(b) SB0, SB1 ..... Serial data input/output dual-function pin
For both master and slave, output is N-ch open drain, input is Schmitt input.

Since the serial data bus line output is N-ch open-drain, an external pull-up resistor is necessary.

Figure 5-59 Pin Configuration Diagram



NOTE: Since the N-ch transistor must be turned off during data reception, FFH should be written to SIO beforehand. It can always be turned off during transmission. However, when the wake-up function specification bit (WUP) is 1, the N-ch transistor is always off, and therefore FFH need not be written to SIO prior to reception.

(7) Address match detection method

In the SBI mode, master address communication is used to select a specific slave and start communication.

Address match detection is performed by hardware. A slave address register (SVA) is provided, and IRQCSI is set only when the address sent from the master and the value set in SVA match in the wake-up state (WUP = 1).

NOTE 1: Detection of the slave selected/ nonselected state is performed by detection of a match with a slave address received after bus release (when RELD = 1).
An address match interrupt (IRQCSI) generated when WUP = 1 is normally used for this match detection. Therefore, detection of selection/nonselection by slave address should be performed with WUP =1.

NOTE 2: For selection/nonselection detection without using an interrupt when WUP = 0, the address detection method is not used, but instead detection should be performed by transmission/reception of commands set beforehand by the program.

(8) Error detection

In the SBI mode, since the status of the serial bus SB0/SB1 pin during transmission is also written into the SIO shift register of the transmitting device, transmission errors can be detected in the following ways:

(a) Comparison of pre-transmission and post-transmission SIO data

In this case, a transmission error is judged to have been generated if the two data items are different.

(b) Use of slave address register (SVA)

Transmission is performed after setting the send data in the SIO and SVA registers. After transmission the COI bit of the serial operating mode register (CSIM) (the match signal from the address comparator) is tested: "1" indicates normal transmission, and "0", a transmission error.

(9) Communication operation

With the SBI, the master normally selects the slave device to be communicated with from among the multiple connected devices by outputting an address onto the serial bus.

After the target communication device has been determined, commands and data are exchanged between the master device and slave device, thus implementing serial communication.

Data communication timing charts are shown in Figures 5-60 through 5-63.

In the SBI mode, shift register shift operations are performed in synchronization with the fall of the serial clock ($\overline{SCK}$), and send data is latched in the SO latch and is output MSB-first from the SB0/P02 or SB1/P03 pin. Receive data input to the SB0 (or SB1) pin is latched in the shift register on the rise of $\overline{SCK}$.

Figure 5-60 Address Transmission from Master Device to Slave Device (WUP = 1)

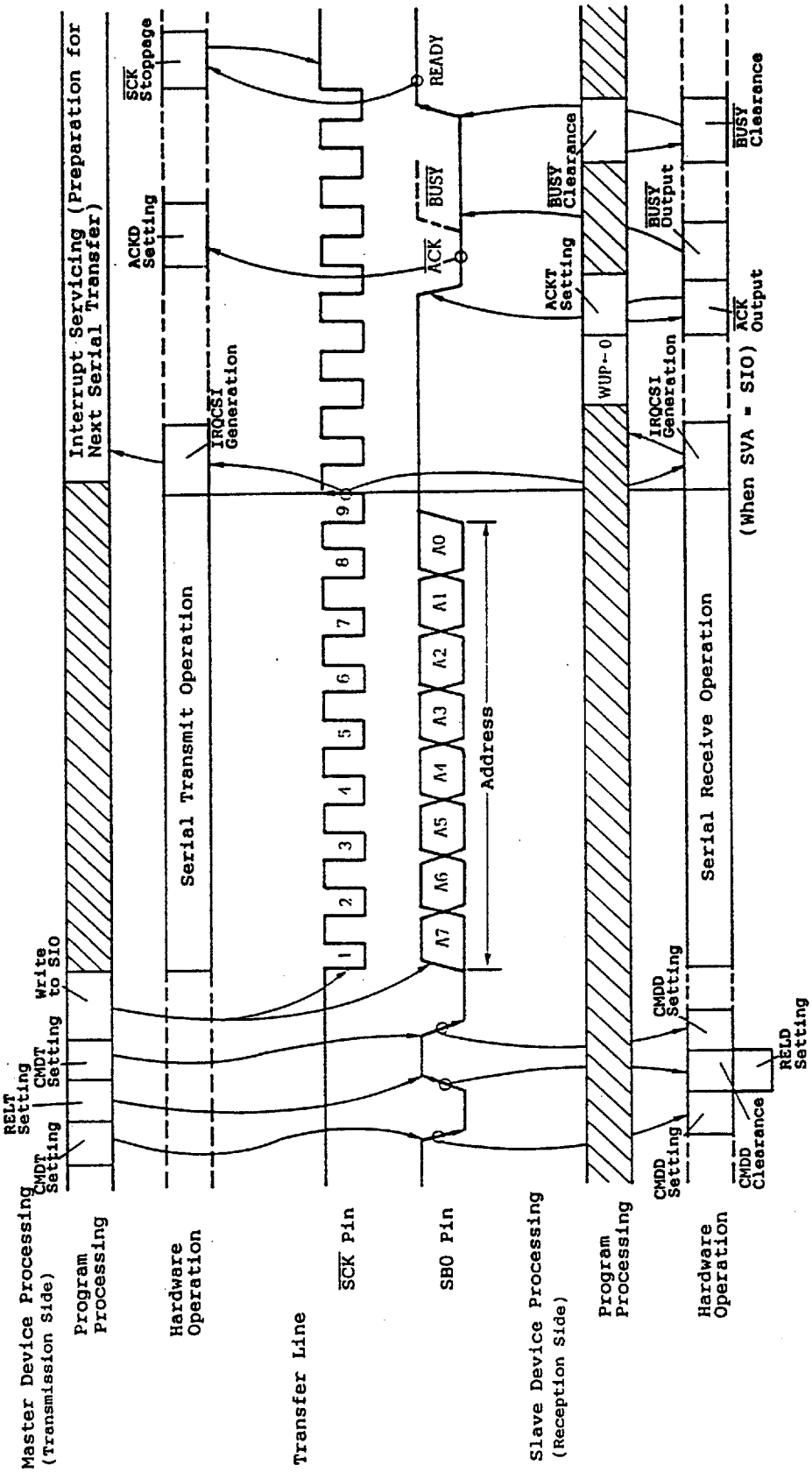


Figure 5-61 Command Transmission from Master Device to Slave Device

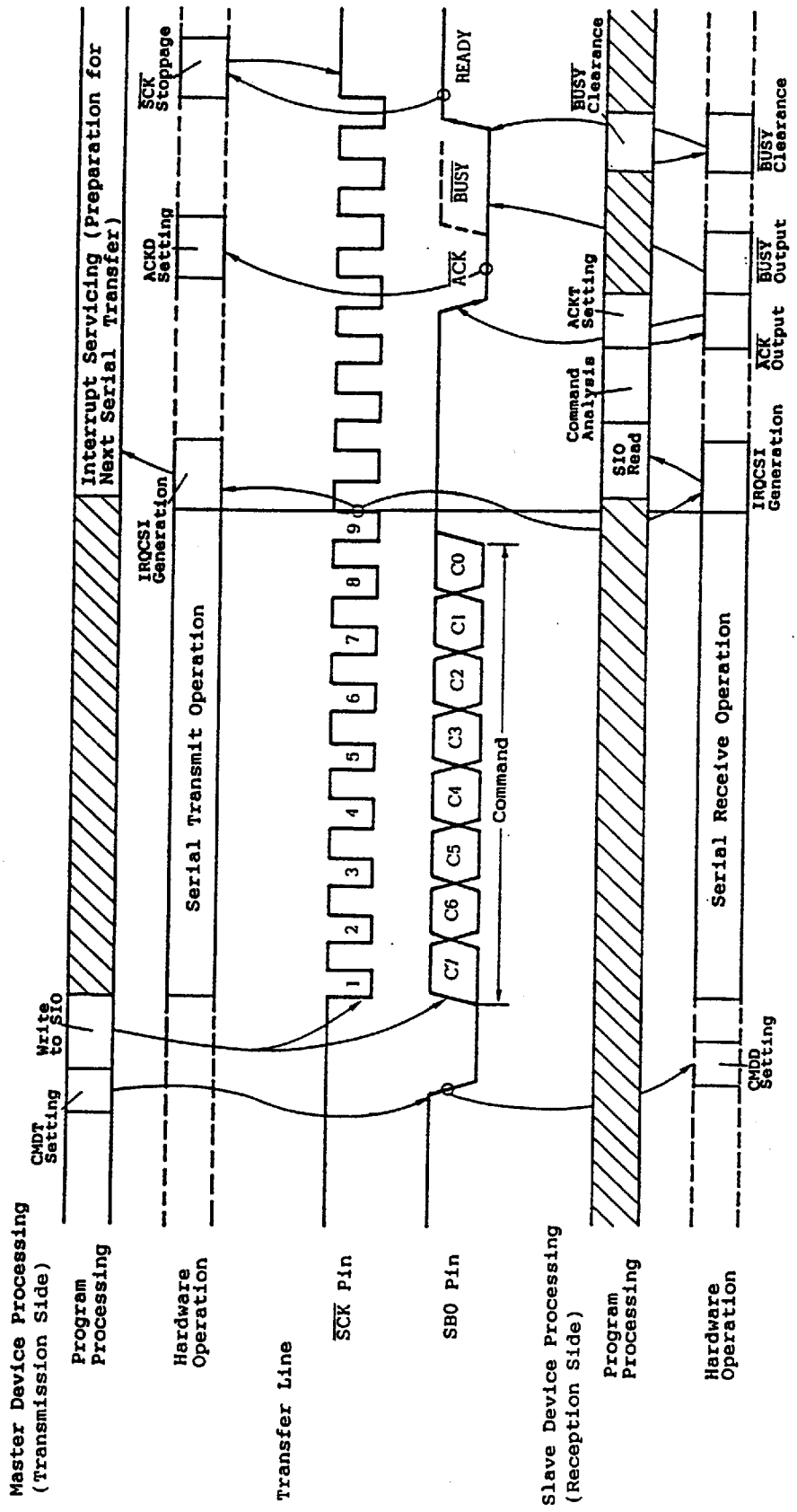


Figure 5-62 Data Transmission from Master Device to Slave Device

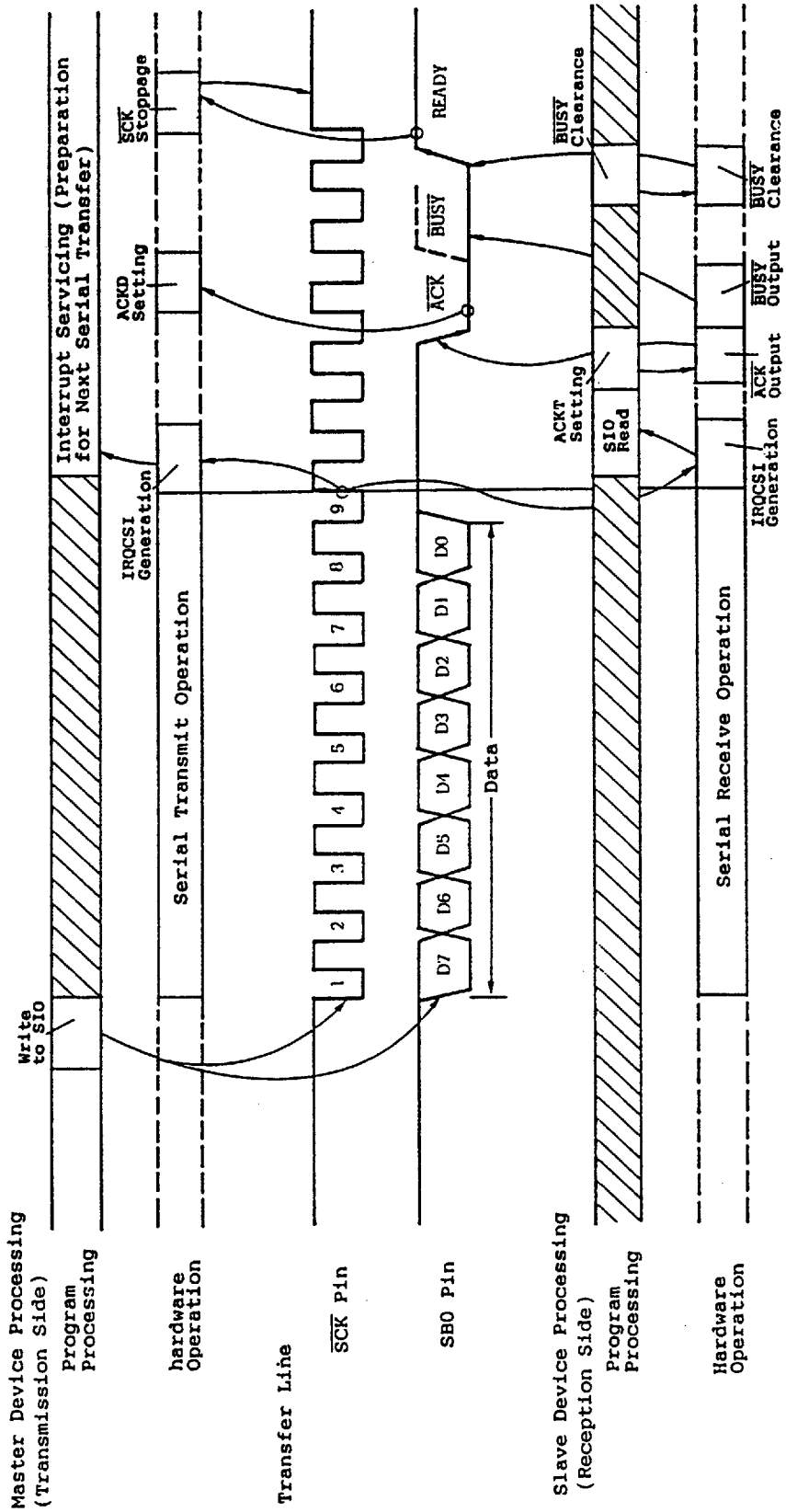
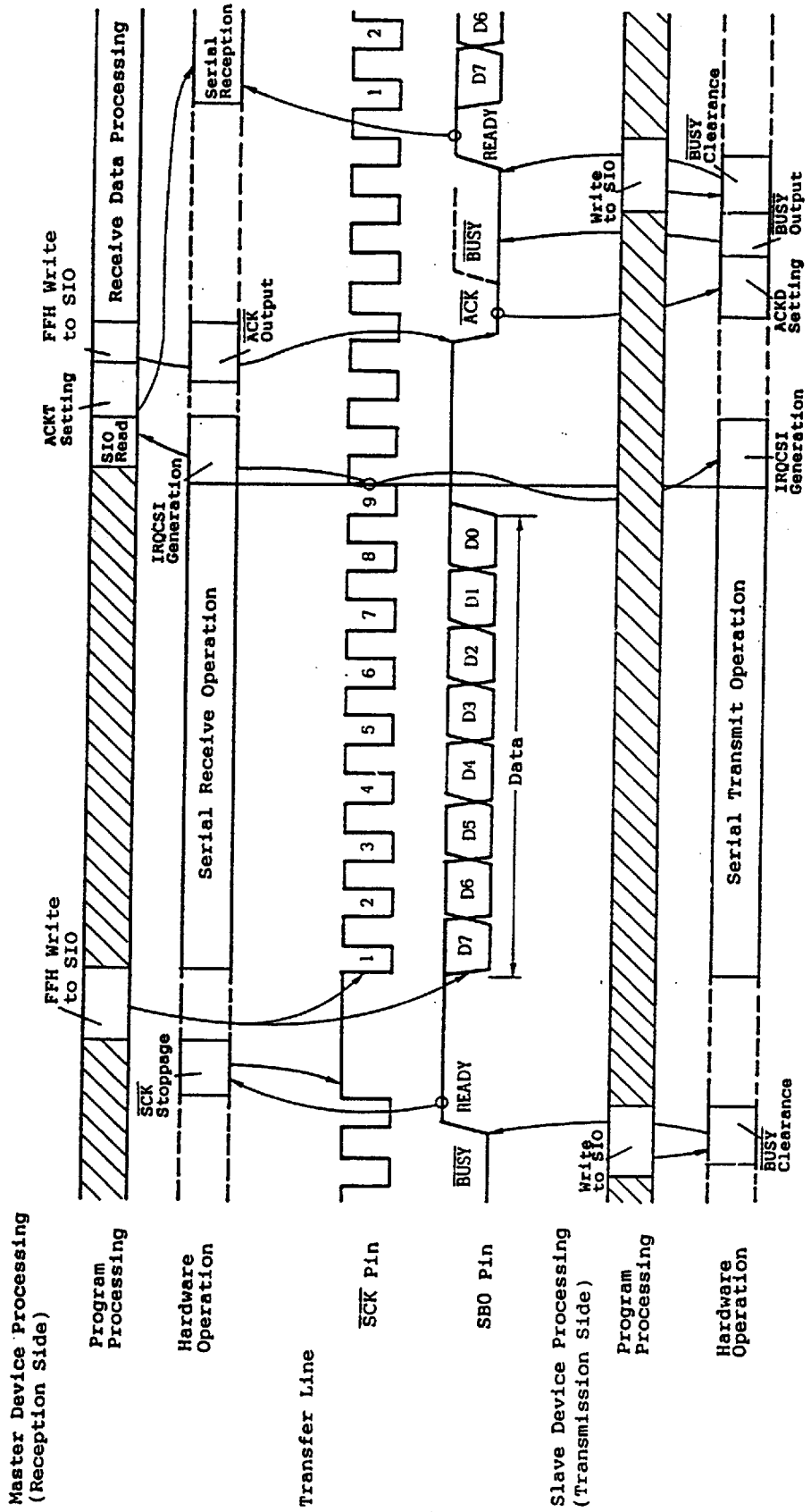


Figure 5-63 Data Transmission from Slave Device to Master Device



(10) Start of transfer

When the following two conditions are met a serial transfer is started by setting transfer data in the shift register (SIO).

- The serial interface operation enable/disable bit (CSIE) = 1.

- After an 8-bit serial transfer, the internal serial clock is stopped or $\overline{SCK}$ is high.

NOTE 1: The transfer will not be started if CSIE is set to "1" after data is written into the shift register.

2: Since the N-ch transistor must be turned off during data reception, FFH should be written to SIO beforehand. However, when the wake-up function specification bit (WUP) is 1, the N-ch transistor is always off, and therefore FFH need not be written to SIO prior to reception.

3: If data is written to SIO when the slave is in the busy state, that data is not lost.
The transfer starts when the busy state is released and the SB0 (or SB1) input becomes high (ready state).

When an 8-bit transfer ends, the serial transfer stops automatically and the IRQCSI interrupt request flag is set.

Example: In the following example the data in the RAM specified by the HL register is transferred to SIO, and at the same time the SIO data is fetched into the accumulator and the serial transfer is started.

```
MOV  XA, @HL; Fetch send data from RAM
SEL  MB15   ; or CLR1 MBE
XCH  XA, SIO; Exchange send data with
              receive data and start
              transfer
```

(11) Points to note concerning SBI mode

(a) Detection of the slave detected/nondetected state is performed by detection of a match with a slave address received after bus release (when RELD = 1).

An address match interrupt (IRQCSI) generated when WUP = 1 is normally used for this match detection. Therefore, detection of selection/nonselection by slave address should be performed with WUP = 1.

(b) For selection/nonselection detection without using an interrupt when WUP = 0, the address detection method is not used, but instead detection should be performed by transmission/reception of commands set beforehand by the program.

(c) When WUP is set to 1 during $\overline{\text{BUSY}}$ signal output, $\overline{\text{BUSY}}$ is not released. With the SBI, the $\overline{\text{BUSY}}$ signal is output following a $\overline{\text{BUSY}}$ release directive until the next fall of the serial clock ($\overline{\text{SCK}}$). When WUP is set to 1, a check must be performed to ensure that SB0 (SB1) has been driven high after $\overline{\text{BUSY}}$ is released before setting WUP to 1.

(12) SBI mode applications

This section introduces an example of applications in which serial data communication is performed in the SBI mode. In this application example the uPD75336 can operate as either the master CPU or a slave CPU on the serial bus.

Also, the master can be changed by a command.

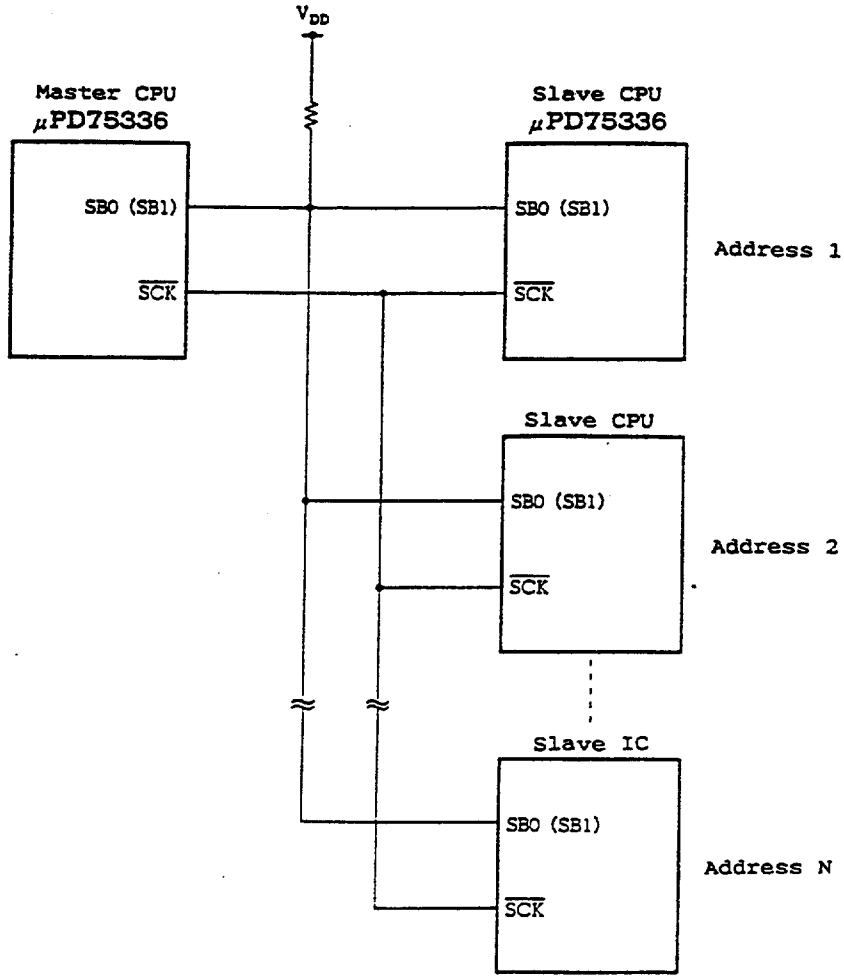
(a) Serial bus configuration

In the serial bus configuration in the application example given here the uPD75336 is assumed to be connected to the bus line as one of the devices on the serial bus.

Two uPD75336 pins are used: The serial data bus SB0 (P02/SO) and the serial clock $\overline{\text{SCK}}$ (P01).

A serial bus configuration example is shown in Figure 5-64.

Figure 5-64 Example of Serial Bus Configuration



(b) Description of commands

<Command Types>

In this application example the following commands are set.

① READ command : Performs data transfer from slave to master.

② WRITE command : Performs data transfer from master to slave.

③ END command : Indicates WRITE command completion to slave.

④ STOP command : Indicates WRITE command suspension to slave.

⑤ STATUS command: Reads slave-side status.

⑥ RESET command : Sets currently selected slave to nonselected status.

⑦ CHGMST command: Transfers mastership to slave side.

<Communication Procedure>

The procedure for communication between master and slave is shown below.

① The master starts communication by sending the address of the slave it wishes to communicate with to select the slave (chip selection).

The slave which receives the address returns $\overline{ACK}$ and performs communication with the master (changes from nonselected to selected status).

② Communication is performed between the slave selected in ① and the master by transferring commands and data.

Master and slave on a one-to-one basis, the other slaves must be in the nonselected status.

③ Communication is terminated when the slave changes to the nonselected status. This happens in the following cases:

- When a RESET command is sent from the master the selected slave changes to nonselected status.

- When the master is changed by the CHGMST command, the device which changes from master to slave assumes the nonselected status.

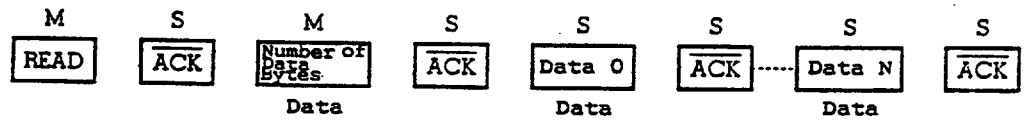
<Command Format>

The transfer format of each command is shown below.

① READ command

This command performs a read from a slave. A variable number of data bytes between 1 and 256 can be read; the number of data bytes is specified as a parameter from the master. If 00H is specified as the number of data bytes, a 256-byte data transfer is regarded as having been specified.

Figure 5-65 READ Command Transfer Format



Remarks: M: Output by master
S: Output by slave

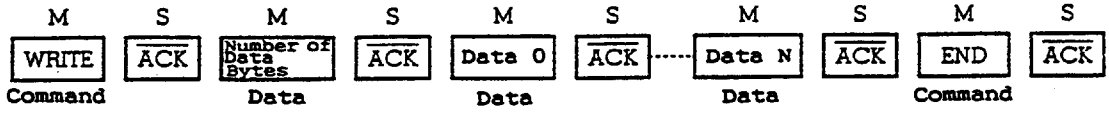
After receiving the number of data bytes, if the transmissible data is not less than that number of data bytes the slave returns $\overline{ACK}$. If there is insufficient data, $\overline{ACK}$ is not returned and an error is generated.

When the master receives data it sends $\overline{ACK}$ to the slave for each byte received.

② WRITE, END and STOP commands

These commands are used to write data to a slave. A variable number of data bytes between 1 and 256 can be written: The number of data bytes is specified as a parameter from the master. If 00H is specified as the number of data bytes, a 256-byte data transfer is specified.

Figure 5-66 WRITE & END Command Transfer Format



Remarks: M: Output by master
S: Output by slave

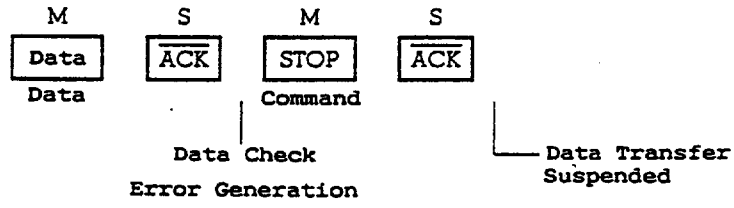
After receiving the number of data bytes, if the receive data storage area is not less than that number of data bytes the slave returns $\overline{ACK}$. If the storage area is insufficient, $\overline{ACK}$ is not returned and an error is generated.

When the master has sent all the data it sends an END command. The END command informs the slave that all the data has been transferred correctly.

The slave also receives an END command before all data has been received. In this case data received up to reception of the END command is valid.

In data transmission, the master compares the SIO contents before and after transmission to check whether the data has been correctly output to the bus. If the pre-transmission and post-transmission SIO contents are different, the master sends a STOP command to suspend the data transfer.

Figure 5-67 STOP Command Transfer Format



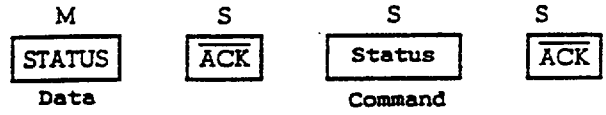
Remarks: M: Output by master
S: Output by slave

When the slave receives the STOP command, it invalidates the data byte received immediately prior to the command.

③ STATUS command

This command reads the currently selected slave status.

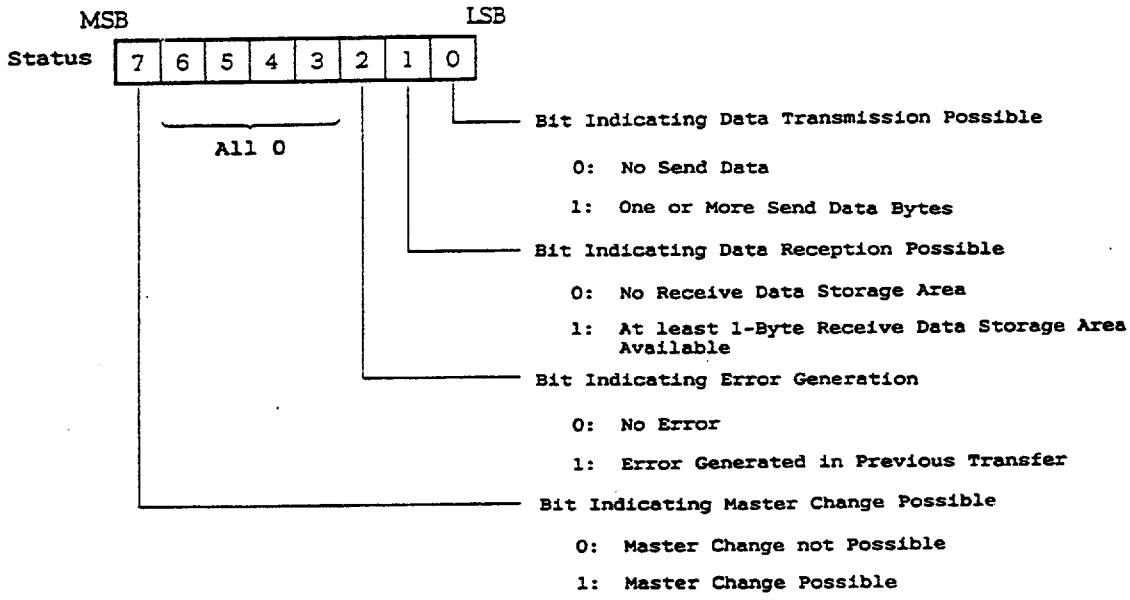
Figure 5-68 STATUS Command Transfer Format



Remarks: M: Output by master
S: Output by slave

The format of the status byte returned by the slave is shown below.

Figure 5-69 STATUS Command Status Format



When the master receives status data it returns $\overline{ACK}$ to the slave.

④ RESET command

This command changes the currently selected slave to the nonselected status. Transmission of the RESET command allows all slaves to be set to the nonselected status.

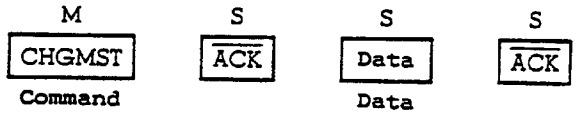
Figure 5-70 RESET Command Transfer Format

Remarks: M: Output by master
S: Output by slave

⑤ CHGMST command

This command passes mastership to the currently selected slave.

Figure 5-71 CHGMST Command Transfer Format



Remarks: M: Output by master
S: Output by slave

When the slave receives the CHGMST command it determines whether it is able to accept mastership, and returns data to the master. This data is as follows:

- 0FFH: Master change possible
- 00H : Master change not possible

When data is transferred the slave compares the pre-transfer and post-transfer contents of SIO: If they do not match it does not return $\overline{ACK}$ and an error is generated.

When the master receives data it returns $\overline{ACK}$ to the slave. If the receive data is FFH, it henceforth operates as a slave. After the slave sends OFFH data and $\overline{ACK}$ is returned from the master, it operates as the master.

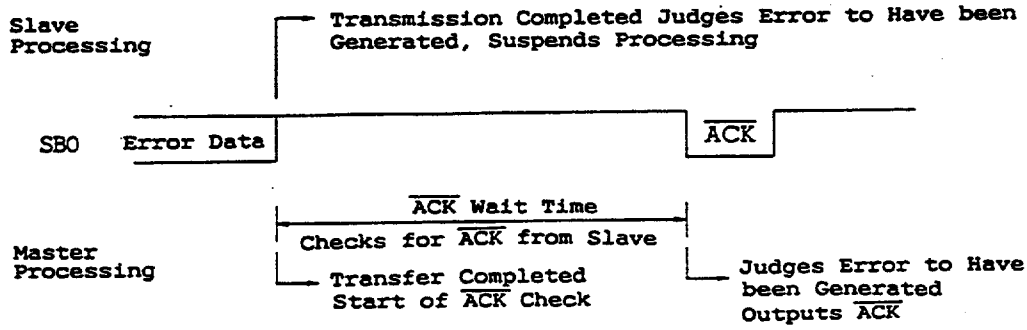
<Error Generation>

Operation in the event of a communication error is described below.

A slave notifies the master of error generation by not returning $\overline{ACK}$. Only when the slave is receiving, when an error is generated the error indication bit in the status byte is set and all executing command processing is canceled.

After transmission of one byte is completed the master checks for $\overline{ACK}$ from the slave. If $\overline{ACK}$ is not returned from the slave within a specific time after completion of transmission an error is judged to have been generated and the master output an $\overline{ACK}$ signal (as a dummy).

Figure 5-72 Master & Slave Operation in Case of Error Generation



Errors are shown below.

- Errors generated on slave side

① Error in command transfer format

② Reception of undefined command

③ Insufficient data transferred in READ command

④ Insufficient data storage area in WRITE command.

⑤ When data changes in READ, STATUS and CHGMST command data transmission.

When any of ① through ⑤ occurs, $\overline{ACK}$ is not returned.

- Errors generated on master side

When data changes during WRITE command data transmission, a STOP command is sent to the slave.

5.6.8 $\overline{SCK}$ PIN OUTPUT MANIPULATION

As an output latch is incorporated in the $\overline{SCK}$/P01 pin, it can be used for static output by means of software in addition to its normal serial clock function.

In addition, P01 output latch manipulation allows the $\overline{SCK}$ number to be set to any desired value by software (SO/SB0, SI/SB1 pin control is performed by the RELT and CMDT bits of SBIC).

The method of manipulating the $\overline{SCK}$/P01 pin is shown below.

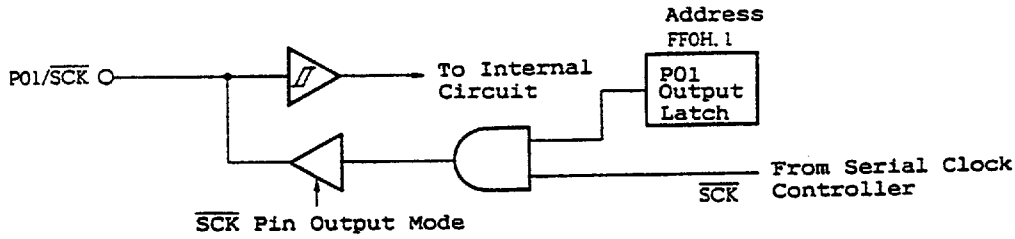
① The serial operation mode register (CSIM) is set ($\overline{SCK}$ pin: Output mode). While serial transfer is suspended, $\overline{SCK}$ = 1.

② The P01 output latch is manipulated by a bit-manipulation instruction.

Example: To output one $\overline{SCK}$ clock cycle by software.

```
SEL   MB15            ; or CLR1 MBE
MOV   XA, #00000011B; SCK (fX/2^3), output mode
MOV   CSIM, XA
CLR1  0FF0H.1         ; SCK/P01 ← 0
SET1  0FF0H.1         ; SCK/P01 ← 1
```

Figure 5-73 $\overline{SCK}$/P01 Pin Configuration



The P01 output latch is mapped onto bit 1 of address FF0H. $\overline{RESET}$ signal generation sets the P01 output latch to "1".

NOTE 1: The P01 output latch must be set to "1" during a normal serial transfer.

NOTE 2: The P01 output latch address cannot be specified as "PORT0.1" as shown below. The address (FF0H.1) must be written directly as the operand. However, when the instruction is executed, "MBE = 0" or "MBE = 1 and MBS = 15" needs to be set beforehand.

```
CLR1   PORT0.1  } Cannot be used
SET1   PORT0.1  }
```

```
CLR1   0FF0H.1  } Can be used
SET1   0FF0H.1  }
```

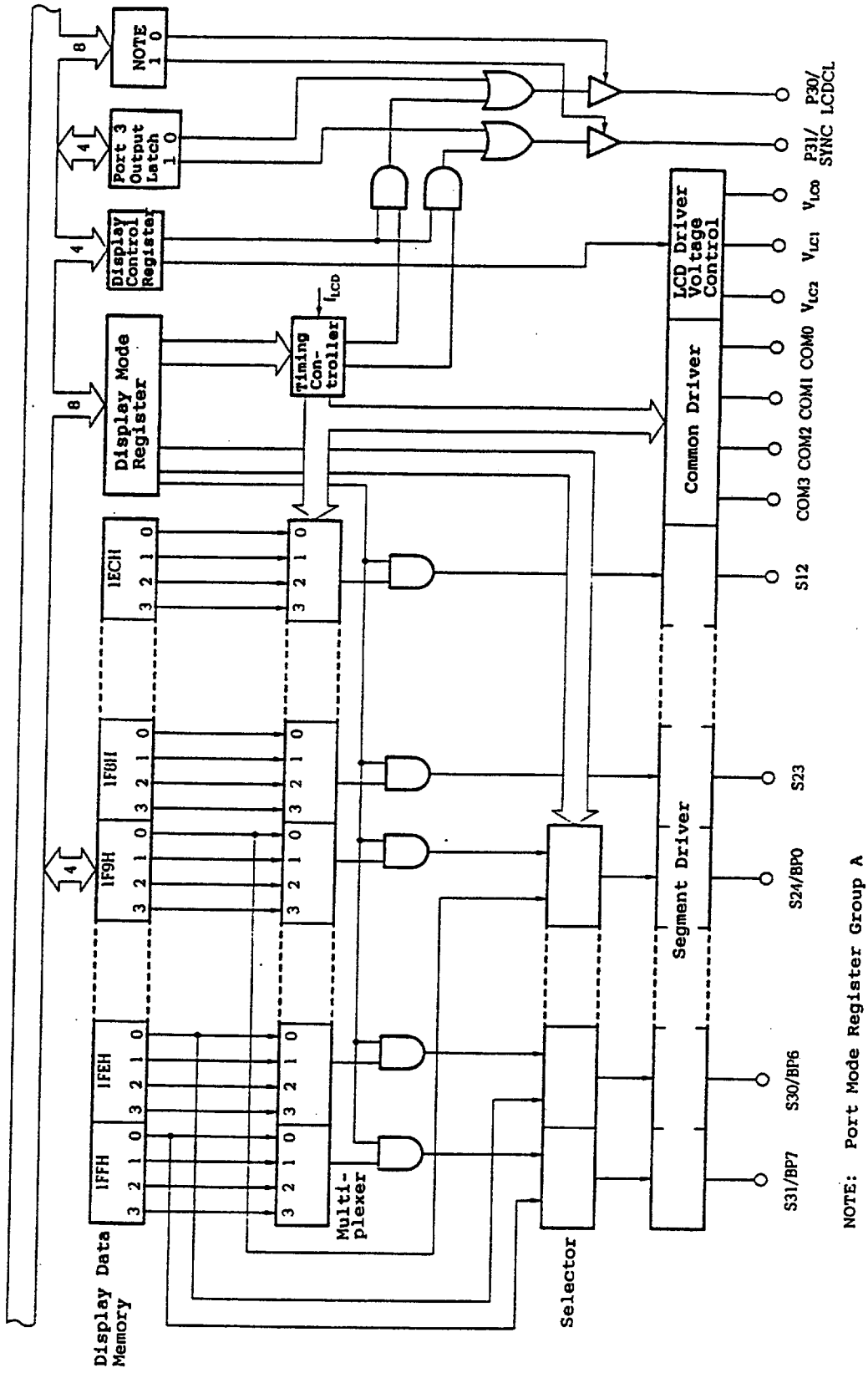
5.7 LCD CONTROLLER/DRIVER

5.7.1 LCD CONTROLLER/DRIVER CONFIGURATION

The uPD75336 incorporates a display controller which generates a segment signal and a common signal in accordance with display data memory data, a segment driver and a common driver which can directly drive the LCD panel.

The LCD controller/driver configuration is shown in Figure 5-74.

Figure 5-74 LCD Controller/Driver Block Diagram



NOTE: Port Mode Register Group A

5.7.2 LCD CONTROLLER/DRIVER FUNCTIONS

The on-chip LCD controller/driver of the uPD75336 has the following functions.

(a) Reads the display data memory automatically by DMA operation and generates segment and common signals.

(b) Can select display mode from among the following five modes.

① Static
② 1/2 duty (2-time multiplexing), 1/2 bias
③ 1/3 duty (3-time multiplexing), 1/2 bias
④ 1/3 duty (3-time multiplexing), 1/3 bias
⑤ 1/4 duty (4-time multiplexing), 1/3 bias

(c) Can select frame frequency in each display mode from among four frame frequencies.

(d) There are a maximum of 20 segment signal outputs (S12 to S31) and a maximum of 4 common outputs (COM0 to COM3).

(e) Segment signal outputs (S24 to S27, S28 to S31) can be switched as a group of 4 to the output ports (BP0 to BP3, BP4 to BP7).

(f) A split resistor for LCD drive power supply can be incorporated (mask option).

- Can handle various bias methods and LCD drive voltages.
- Current flow to the split resistor is cut when display is OFF.

(g) The display data memory not used for display can be used as normal data memory.

(h) Can operate with a subsystem clock.

Table 5-12 shows the maximum number of pixels which can be displayed in each display mode.

Table 5-12 Maximum Number of Pixels Displayed

Bias Method	Time Multi-plexing	COM Signal Used	Maximum No. of Pixels
-	Static	COM0 (COM1, 2, 3)	20 (20 segments x 1 common) *1
1/2	2	COM0, 1	40 (20 segments x 2 commons) *2
	3	COM0, 1, 2	60 (20 segments x 3 commons) *3
1/3	3		
	4	COM0, 1, 2, 3	80 (20 segments x 4 commons) *4

*1: 2 digits of 8 segment signals/digit on 8. -type LCD panel

2: 5 digits of 4 segment signals/digit on 8. -type LCD panel

3: 6 digits of 3 segment signals/digit on 8. -type LCD panel

4: 10 digits of 2 segment signals/digit on 8. -type LCD panel

5.7.3 DISPLAY MODE REGISTER

The display mode register (LCDM) is an 8-bit register used to specify the display mode, LCD clock, frame frequency, segment output/bit port output selection, and display output ON/OFF control.

The LCDM is set by an 8-bit memory manipulation instruction. Only bit 3 (LCDM3) can be set and cleared by a bit manipulation instruction.

When the $\overline{\text{RESET}}$ signal is generated, all bits are cleared to "0".

The display mode register format is shown on the next page.

Figure 5-75 Display Mode Register Format

Address	7	6	5	4	3	2	1	0	Symbol
F8CH	LCDM7	LCDM6	LCDM5	LCDM4	LCDM3	LCDM2	LCDM1	LCDM0	LCDM

Display Mode Selection

LCDM3	LCDM2	LCDM1	LCDM0	No. of Time Multiplexing	Bias Method
0	×	×	×	Display OFF*	
1	0	0	0	4	1/3
1	0	0	1	3	1/3
1	0	1	0	2	1/2
1	0	1	1	3	1/2
1	1	0	0	Static	
Others				Setting prohibited	

*: All segment signals are at non-selection level.

LCD Clock Selection

LCDM5	LCDM4	LCDCL
0	0	$f_W/2^9$ (64 Hz)
0	1	$f_W/2^8$ (128 Hz)
1	0	$f_W/2^7$ (256 Hz)
1	1	$f_W/2^6$ (512 Hz)

NOTE: LCDCL is only supplied when the clock timer is in operation. When using the LCD controller, set the clock mode register WM bit 2 to "1".

Segment Output/Bit Port Output Switching Specification

LCDM7	LCDM6	S24-S27	S28-S31	No. of Segment Outputs	No. of Bit Port Outputs
0	0	Segment output	Segment output	20	0
0	1	Segment output	Bit port output	16	4
1	0	Bit port output	Segment output	16	4
1	1	Bit port output	Bit port output	12	8

Frame Frequency (Hz)

Display Duty \ LCDCL	$f_W/2^9$ (64 Hz)	$f_W/2^8$ (128 Hz)	$f_W/2^7$ (256 Hz)	$f_W/2^6$ (512 Hz)
Static	64	128	256	512
1/2	32	64	128	256
1/3	21	43	85	171
1/4	16	32	64	128

At f_W = 32.768 kHz
At f_W = Clock Timer Input Clock ($f_X/128$ or f_{XT})

5.7.4 DISPLAY CONTROL REGISTER

The display control register (LCDC) controls the following LCD drive operations.

- Enabling/disabling common and segment output
- Cutting the current flow to the LCD drive power supply split resistor
- Enabling/disabling synchronous clock (LCDCL) and synchronous signal (SYNC) to the external segment signal expansion controller/driver

The LCDC is set by a 4-bit memory manipulation instruction.

When $\overline{\text{RESET}}$ is generated, all bits of the display control register are cleared to "0".

Figure 5-76 Display Control Register Format

Address	3	2	1	0	Symbol
F8EH	0	LCDC2	0	LCDC0	LCDC

Display and Output Status through LCDC0 and LCDM3

LCDC0	0	1	
LCDM3	×	0	1
COM0-3	"L" output (display OFF)	Output of common signal corresponding to display mode	Output of common signal corresponding to display mode
S12-S23 S24 to S31 segment specification pins	"L" output (display OFF)	Output of segment signal corresponding to display mode (non-selection level output, display OFF)	Output of segment signal corresponding to display mode (display ON)
S24 to S31 bit port specification pins	Output of bit 0 contents of the corresponding display data memory (bit port function)	Output of bit 0 contents of the corresponding display data memory (bit port function)	Output of bit 0 contents of the corresponding display data memory (bit port function)
Power supply to the dividing resistor (BIAS pin output)	OFF (high impedance) *1	ON (high level) *1	ON (high level) *1

LCDCL and SYNC Signal *2 Output Enable/Disable Specification Bits

LCDC2	0	LCDCL and SYNC signal output disabled
	1	LCDCL and SYNC signal output enabled

*1: Cases when LCD drive power supply split resistor is not incorporated are shown in parentheses.

2: LCDCL and SYNC signal outputs are reserved for future system expansion. Disable these signals for output at present.

NOTE: Figure 5-76 shows the cases when LCD drive power (V_{LC0} to V_{LC2}) is supplied by BIAS pin output. If the LCD drive power supply is not dependent on BIAS pin output, the display output and bit port output are not affected by the LCDC0 bit and are determined by LCDM setting only.

5.7.5 DISPLAY DATA MEMORY

The display data memory is mapped onto 1ECH to 1FFH.

This memory is an area which the LCD controller/driver reads by a DMA operation irrespective of CPU operation. The LCD controller controls the segment signals in accordance with the display data memory data. When S24 to S31 are used as bit ports, the bit 0 contents of the data written at addresses 1F8H to 1FFH of the display data memory are output from each bit port output pin.

If the display data memory is not used as an LCD display memory or port, it can be used as normal data memory.

The display data memory is manipulated bit-wise or in 4-bit units. 8-bit manipulation is not possible for this memory.

Figure 5-78 shows the relationships between each display data memory bit and segment/bit port output.

Figure 5-77 Data Memory Map

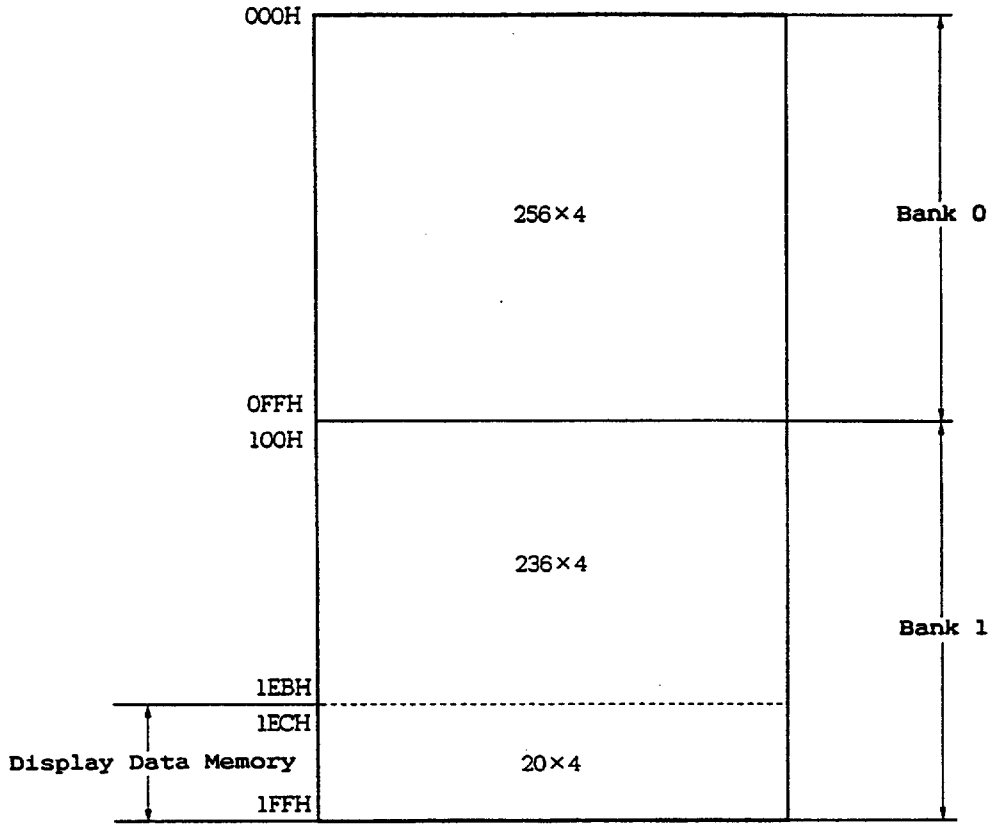
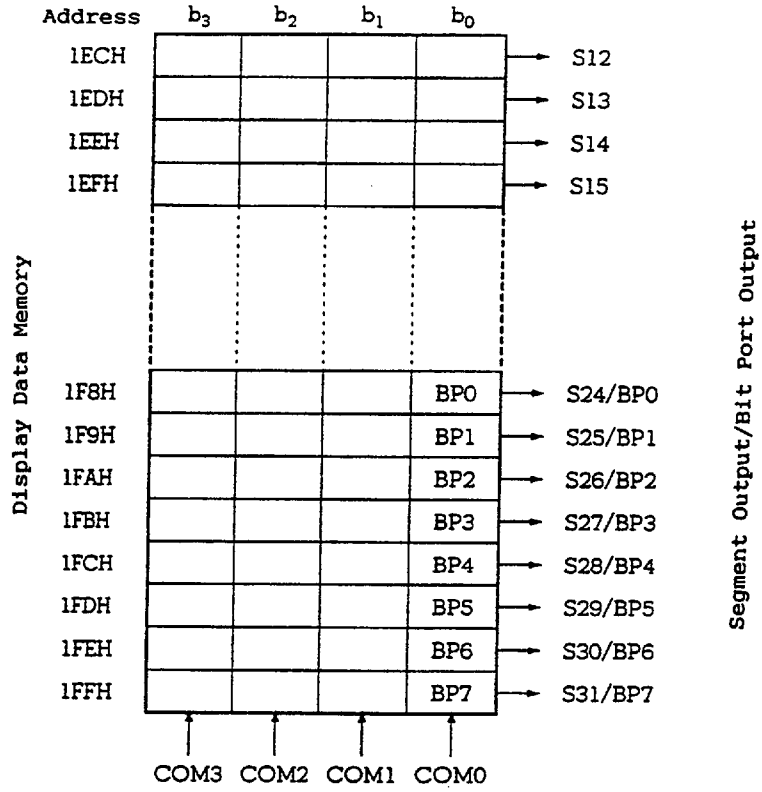


Figure 5-78 Relations between Display Data Memory and Common and Segment Signals



NOTE: When the 8-bit manipulation instruction is executed for the display data memory area (1ECH to 1FFH), the uPD75336 and the emulation chip mounted in the in-circuit emulator for evaluation operate differently as follows.

uPD75336	CPU Emulation Chip (uPD75000A)	Peripheral Hardware Emulation Chip (uPD75390)
8-bit unit read/ write disabled	8-bit unit normal read/write enabled	8-bit unit read/write disabled

When a program for 8-bit manipulation for the display data memory is created, it is not executed by the uPD75336. Do not execute 8-bit manipulation for the display data memory.

If read/write is executed on the in-circuit emulator by the 8-bit manipulation instruction for the display data memory, an access is made as follows.

- In a write, data is written to both the uPD75000A and uPD75390.
- In a read, data is read from the uPD75000A.

Thus, when 8-bit manipulation is executed on the in-circuit emulator for the display memory area, normal operation appears to be carried out because the uPD75000A is accessed. However, since LCD display is made for the data written to the uPD75390, data written in 8-bit units is not normally displayed.

5.7.6 COMMON SIGNAL AND SEGMENT SIGNAL

Each pixel on the LCD panel lights up when the potential difference between the corresponding common and segment signals reaches the specified voltage (LCD drive voltage V_{LCD}). If the potential difference is equal to or less than V_{LCD} or becomes 0 V, the LCD panel pixel goes out.

Because the LCD panel deteriorates if a DC voltage is applied to the common and segment signals, it is driven by an AC voltage.

(1) Common signals

Common signals are selected in the order shown in Table 5-13 according to the set number of time multiplexing and operation is repeated with the set time multiplexing as one cycle. In the static mode COM0, COM1, COM2 and COM3 are the same signals. Open COM2 and COM3 pins for 2-time multiplexing and open COM3 pin for 3-time multiplexing.

Table 5-13 COM Signals

COM Signal / No. of Time Multiplexing	COM0	COM1	COM2	COM3
Static				
2			Leave open	Leave open
3				Leave open
4				

(2) Segment signals

There are 20 segment signals corresponding to 20 locations in the data memory display data area (1ECH to 1FFH). Bits 0 to 3 are automatically read in synchronization with the timings of COM0 to COM3, respectively. When each bit content is 1 or 0, it is converted to a select voltage or a non-select voltage, respectively, and is output from the segment pin (S12 to S31).

Thus, check for combinations of LCD panel front electrodes (corresponding to the segment signals) and rear electrodes (corresponding to the common signals) in which a display pattern is formed in the display data area and write bit data corresponding to the pattern to be displayed in the ratio of 1:1.

Bits 1, 2 and 3 in the display data area based on the static method, bits 2 and 3 based on 2-time multiplexing, and bit 3 based on 3-time multiplexing are not accessed. Thus, these bits can be used for purposes other than display.

(3) Output waveforms of common and segment signals

Voltages at levels shown in Tables 5-14 to 5-16 are output to the common and segment signals. A $+V_{LCD}/-V_{LCD}$ ON voltage is set only when both become select signals. In all other combinations, an OFF voltage results.

Table 5-14 LCD Drive Voltage (Static)

Segment Signal Sn / Common Signal COM0	Select	Non-Select
	V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
V_{SS}/V_{LC0}	$+V_{LCD}/-V_{LCD}$	0 V/0 V

Table 5-15 LCD Drive Voltage (1/2 Bias Method)

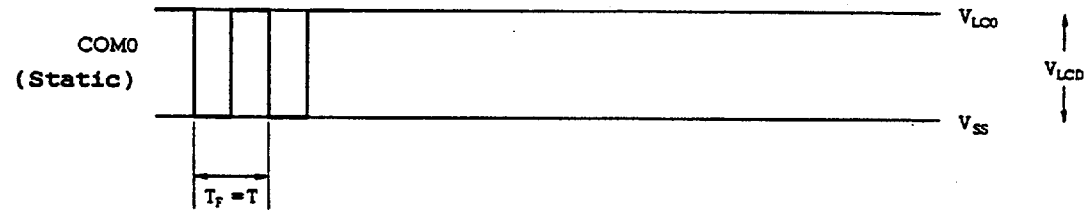
Segment Signal Sn / Common Signal COMm		Select	Non-Select
		V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
Select	V_{SS}/V_{LC0}	$+V_{LCD}/-V_{LCD}$	0 V/0 V
Non-select	$V_{LC1} = V_{LC2}$	$+\frac{1}{2}V_{LCD}/-\frac{1}{2}V_{LCD}$	$-\frac{1}{2}V_{LCD}/+\frac{1}{2}V_{LCD}$

Table 5-16 LCD Drive Voltage (1/3 Bias Method)

Segment Signal Sn / Common Signal COMm		Select	Non-Select
		V_{LC0}/V_{SS}	V_{LC2}/V_{LC1}
Select	V_{SS}/V_{LC0}	$+V_{LCD}/-V_{LCD}$	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$
Non-select	$V_{LC1} = V_{LC2}$	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$	$-\frac{1}{3}V_{LCD}/+\frac{1}{3}V_{LCD}$

Figures 5-79 to 5-81 show common signal waveforms and Figure 5-82 shows common and segment signal voltages and phases.

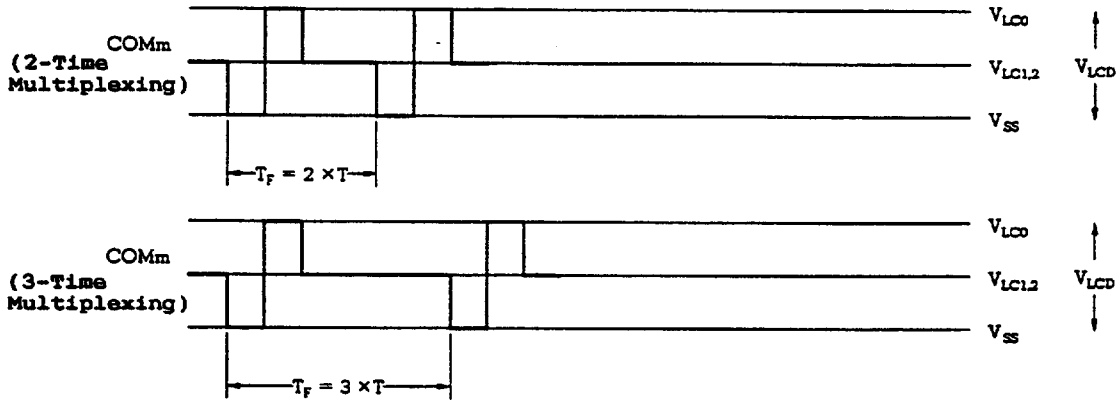
Figure 5-79 Common Signal Waveform (Static)



T : One Cycle of LCDCL

T_F : Frame Cycle

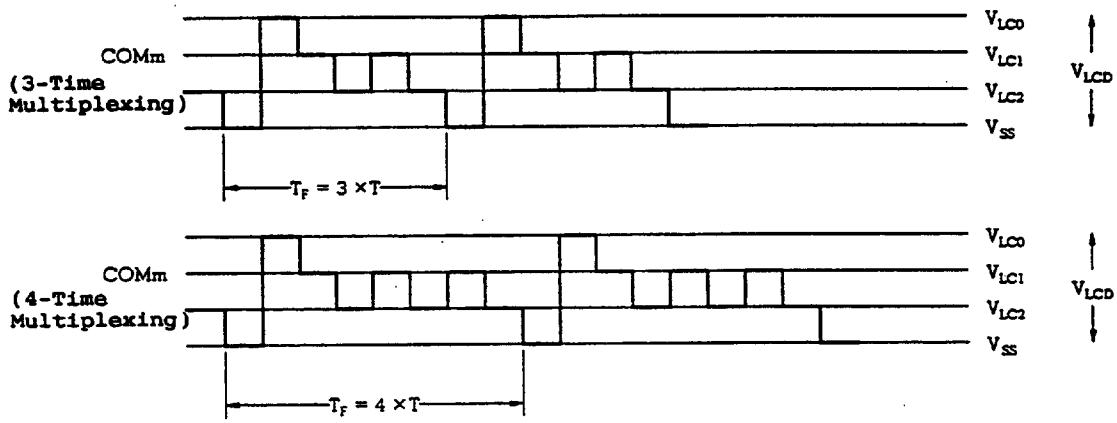
Figure 5-80 Common Signal Waveform (1/2 Bias Method)



T : One Cycle of LCDCL

T_F : Frame Cycle

Figure 5-81 Common Signal Waveform (1/3 Bias Method)

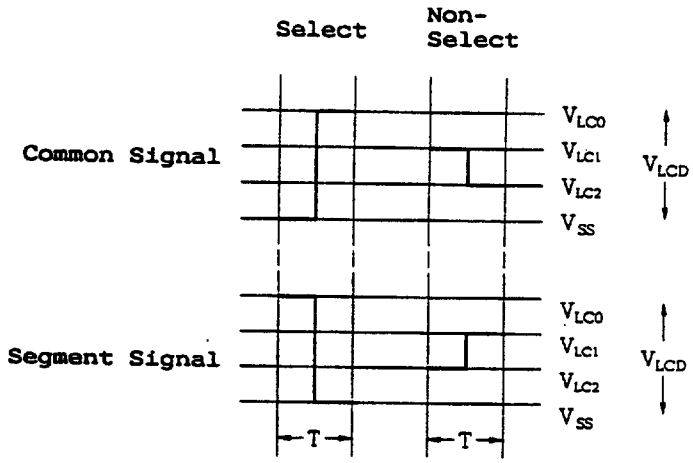


T : One Cycle of LCDCL

T_F : Frame Cycle

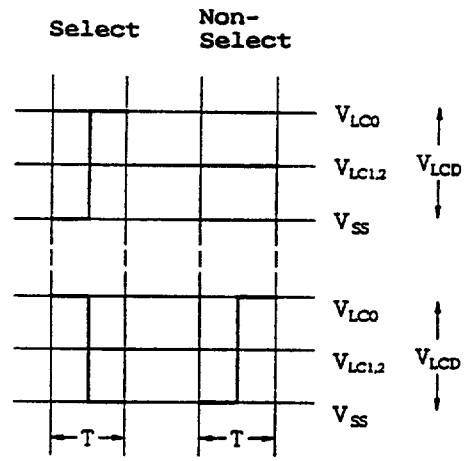
Figure 5-82 Common and Segment Signal Voltages and Phases

(a) 1/3 Bias method

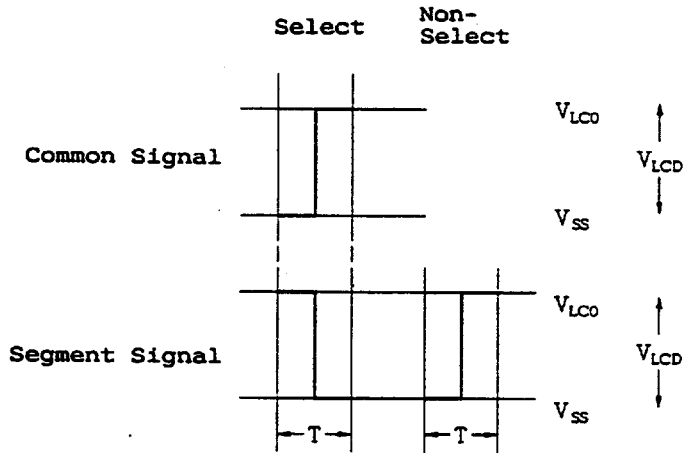


T: One Cycle of LCDL

(b) 1/2 Bias method



(c) Static display mode



5.7.7 V_{LC0}, V_{LC1} AND V_{LC2} POWER SUPPLIES FOR LCD DRIVE

The uPD75336 can incorporate split resistors in V_{LC0} to V_{LC2} pins for LCD drive power supply. It can supply LCD drive power based on each bias method without the use of external split resistors. The uPD75336 is also equipped with BIAS pins to cope with various LCD drive voltages. The BIAS pins are externally connected to V_{LC0} pins.

The following values are supplied as appropriate LCD drive power based on the static, 1/2 and 1/3 bias methods.

Table 5-17 LCD Drive Power Supply Values

Bias Method / LCD Drive Power Supply	Without Bias (Static Mode)	1/2	1/3
V_{LC0}	V_{LCD}	V_{LCD}	V_{LCD}
V_{LC1}	$2/3V_{LCD}$	$1/2V_{LCD}$ *	$2/3V_{LCD}$
V_{LC2}	$1/3V_{LCD}$		$1/3V_{LCD}$
V_{SS}	0 V	0 V	0 V

* : In the case of 1/2 bias, V_{LC1} and V_{LC2} pins must be connected externally.

Remarks : $V_{LCD} = 3/5V_{DD}$ when BIAS and V_{LC0} pins are open. (split resistors must be on chip by mask option.)

$V_{LCD} = V_{DD}$ when BIAS and V_{LC0} pins are connected.

LCD drive power supply examples in accordance with Table 5-17 are shown in Figure 5-83 (a), (b) and (c).

Current flow through the split resistors can be cut off by clearing bit 0 (LCDC0) or the display control register to "0".

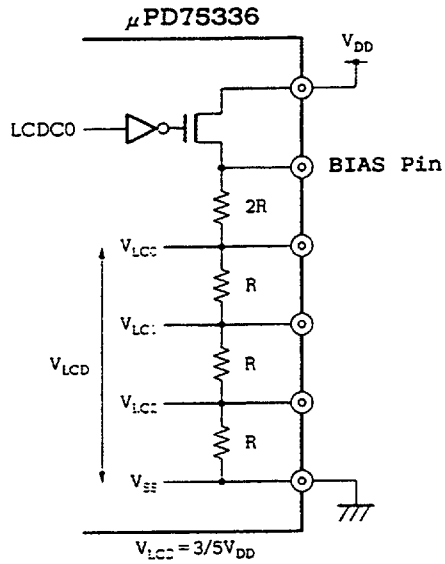
LCD power ON/OFF control is also effective to prevent a DC voltage from being applied to the LCD because the STOP instruction stops the LCD clock (if a system clock has been selected) when the watch timer is in operation with the main-system clock.

In other words, potential differences can be prevented between the LCD electrodes if the LCD clock stops by clearing bit 0 (LCDC0) of the display control register to "0" just before STOP instruction execution, thus equalizing all LCD drive power supplies to the same V_{SS} potential.

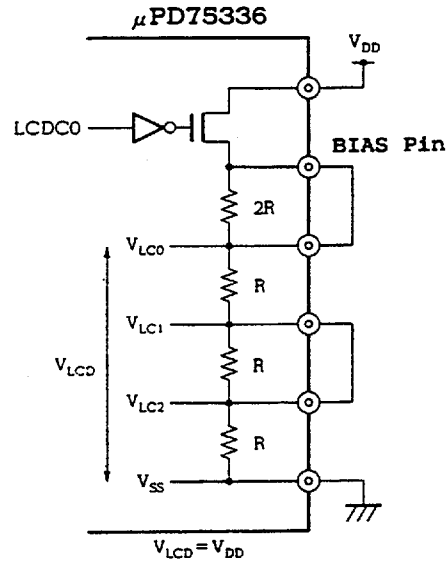
When the watch timer is in operation with the subsystem clock, LCD display can continue.

Figure 5-83 LCD Drive Power Supply Connection Examples (with On-Chip Split Resistors)

(a) 1/3 bias method and static display mode (Example with V_{DD} = 5 V, V_{LCD} = 3 V)



(b) 1/2 bias method (Example with V_{DD} = 5 V, V_{LCD} = 5 V)



(c) 1/3 bias method and static display mode (Example with V_{DD} = 5 V, V_{LCD} = 5 V)

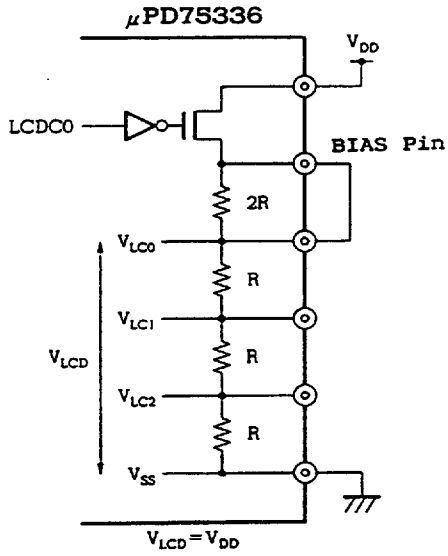
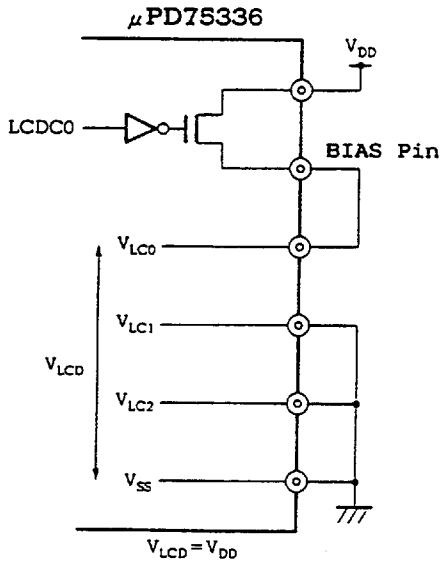
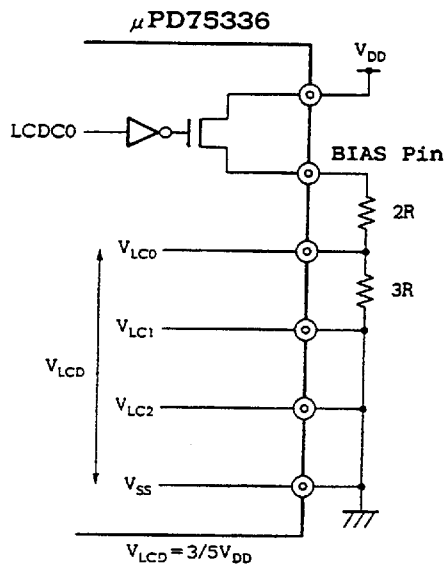


Figure 5-84 LCD Drive Power Supply Connection Examples (with External Split Resistors)

(a) Static display mode* (Example with V_{DD} = 5 V, V_{LCD} = 5 V)



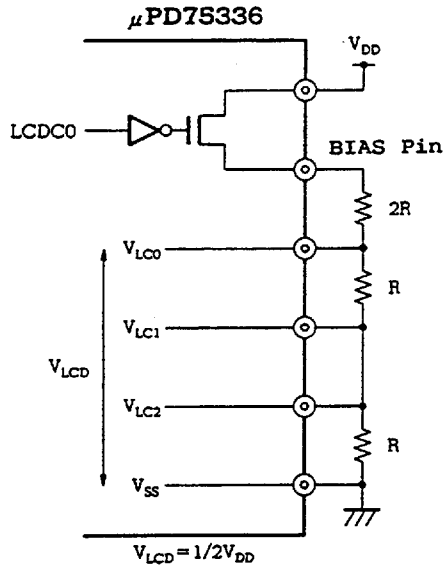
(b) Static display mode (Example with V_{DD} = 5 V, V_{LCD} = 3 V)



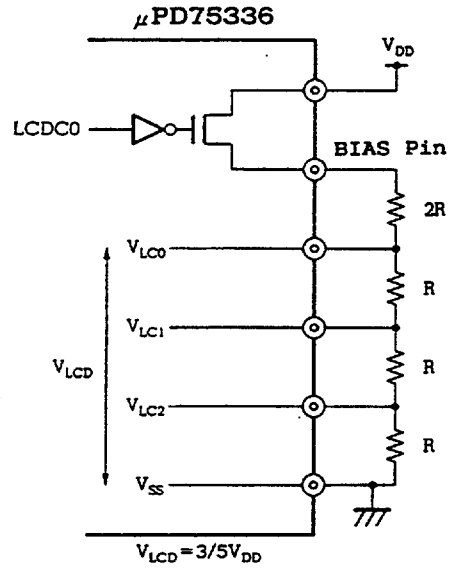
*: Set LCDC0 always to "1" (including the standby mode).

Figure 5-84 LCD Drive Power Supply Connection Examples (with External Split Resistors) (cont'd)

(c) 1/2 bias method (Example with V_{DD} = 5 V, V_{LCD} = 2.5 V)



(d) 1/3 bias method (Example with V_{DD} = 5 V, V_{LCD} = 3 V)



5.7.8 DISPLAY MODES

(1) Static display example

Figure 5-86 shows connection of a 3-digit LCD panel based on the static method with a display pattern shown in Figure 5-85 to the uPD75336 segment signals (S12 to S31) and common signal (COM0). The display example is 123 and the display data memory contents (addresses 1ECH to 1FFH) corresponding to this display.

This section provides a description using the first digit 3 (3) as an example. It is necessary to generate select and non-select voltages shown in Table 5-18 to the S12 to S18 pins at the COM0 common signal timing in accordance with the display pattern in Figure 5-85.

Table 5-18 Select and Non-Select Voltages (COM0)

Segment / Common	S12	S13	S14	S15	S16	S17	S18
COM0	Select	Select	Select	Non-select	Select	Non-select	Select

It is clear that 1110101 should be set at bit 0 of the display data memory (addresses 1ECH to 1F2H) corresponding to S12 to S18.

Figure 5-87 shows the S14, S15 and COM0 LCD drive waveforms. When S14 becomes a select voltage at the COM0 select timing, $+V_{LCD}/-V_{LCD}$ AC rectangular waveform at LCD ON level is generated.

Since the same waveform as that of COM0 is generated at COM1, COM2 and COM3, the drive capability can be improved by connecting COM0, COM1, COM2 and COM3.

Figure 5-85 Static LCD Display Pattern and Electrode Wiring

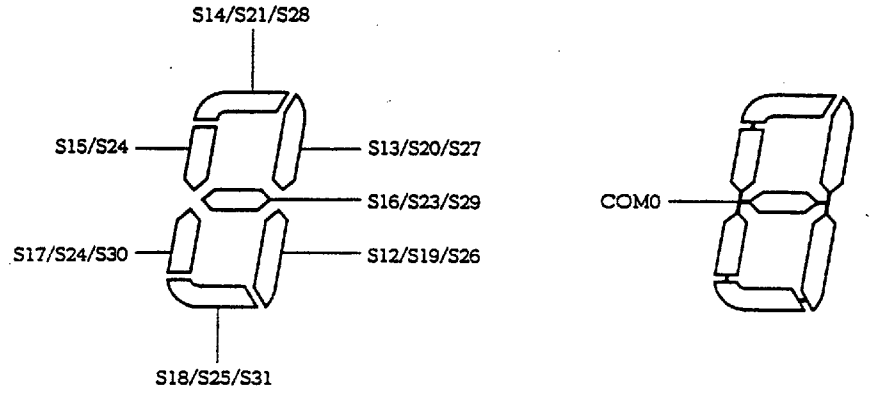


Figure 5-86 Static LCD Panel Wiring Example

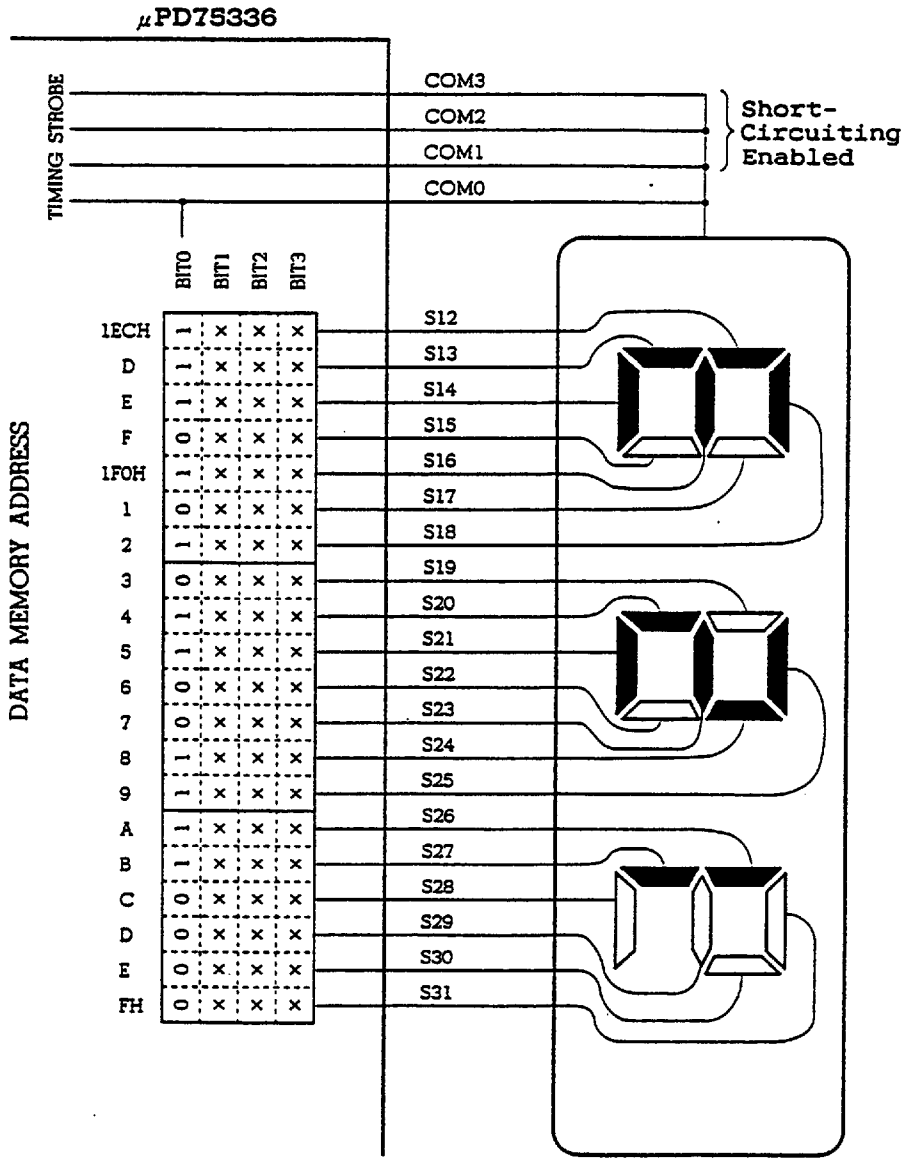
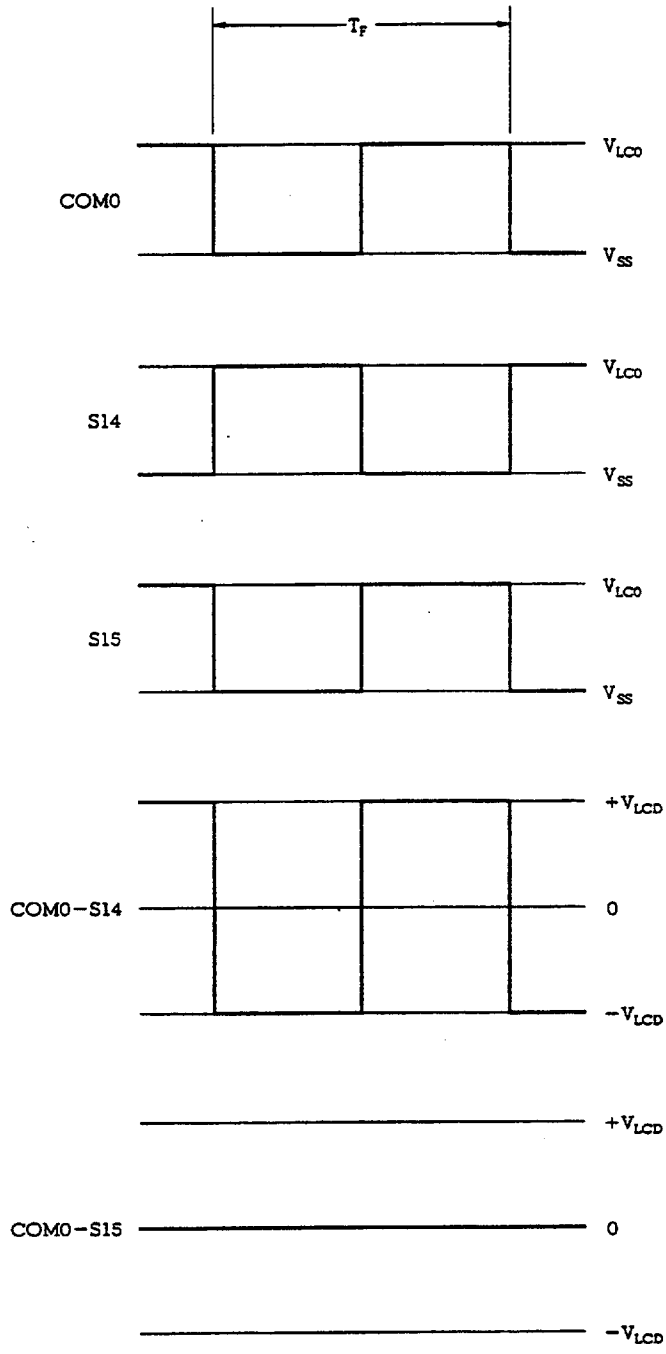


Figure 5-87 Static LCD Drive Waveform Example



(2) 2-time multiplexing display example

Figure 5-89 shows connection of a 5-digit LCD panel based on the 2-time multiplexing method with a display pattern shown in Figure 5-88 to the uPD75336 segment signals (S12 to S31) and common signals (COM0, COM1). The display example is 123.45 and the display data memory contents (addresses 1ECH to 1FFH) corresponding to this display.

This section provides a description using the third digit 3. (3.) as an example. It is necessary to generate select and non-select voltages shown in Table 5-19 to the S20 to S23 pins at the COM0 and COM1 common signal timings in accordance with the display pattern in Figure 5-88.

Table 5-19 Select and Non-Select Voltages (COM0, COM1)

Segment / Common	S20	S21	S22	S23
COM0	Select	Select	Non-select	Non-select
COM1	Select	Select	Select	Select

It is clear, for example, that xx10 should be set in the display data memory (address 1F7H) corresponding to S23.

Figure 5-90 shows the S23 and common signal LCD drive waveforms. When S23 becomes a select voltage at the COM1 select timing, a $+V_{LCD}/-V_{LCD}$ AC rectangular waveform at LCD ON level is generated.

Figure 5-88 2-Time Multiplexing LCD Display Parameter and Electrode Wiring

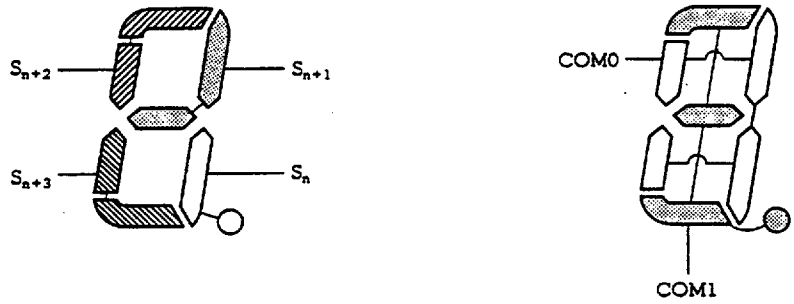
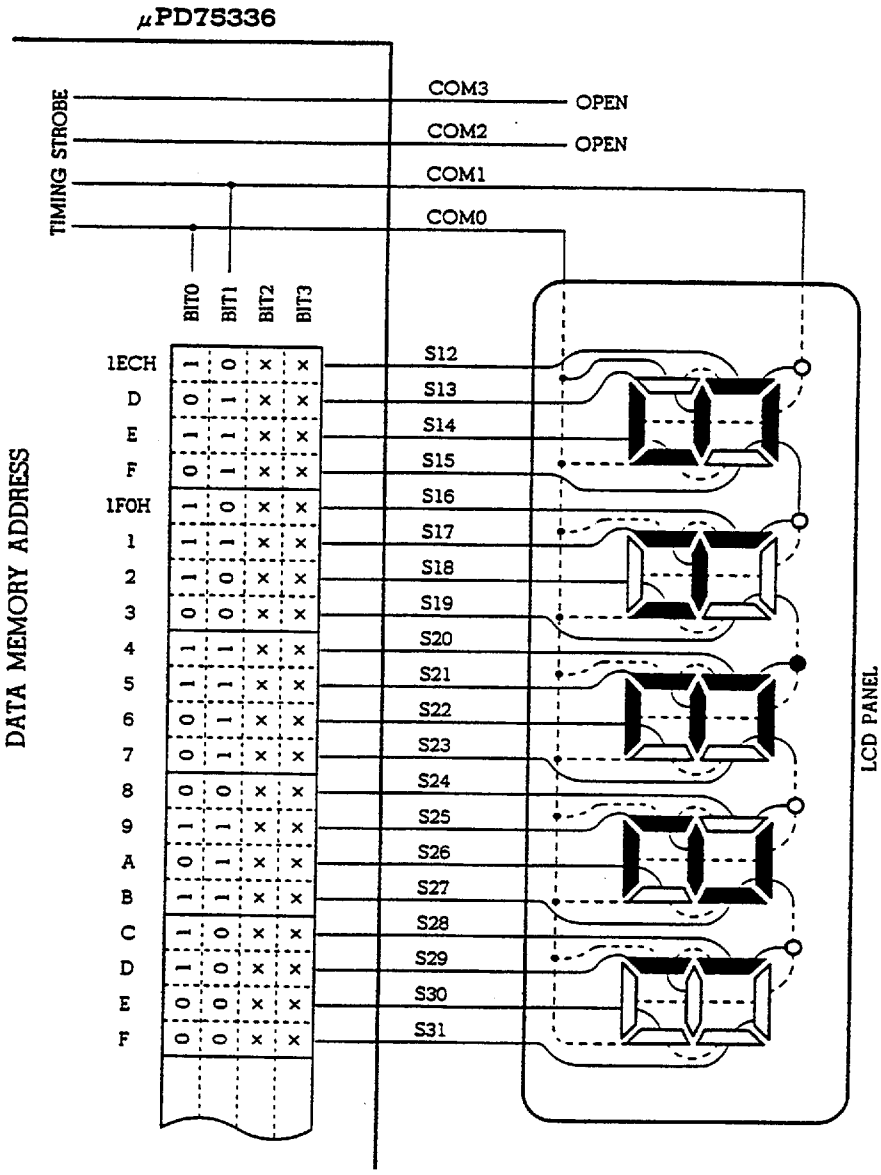
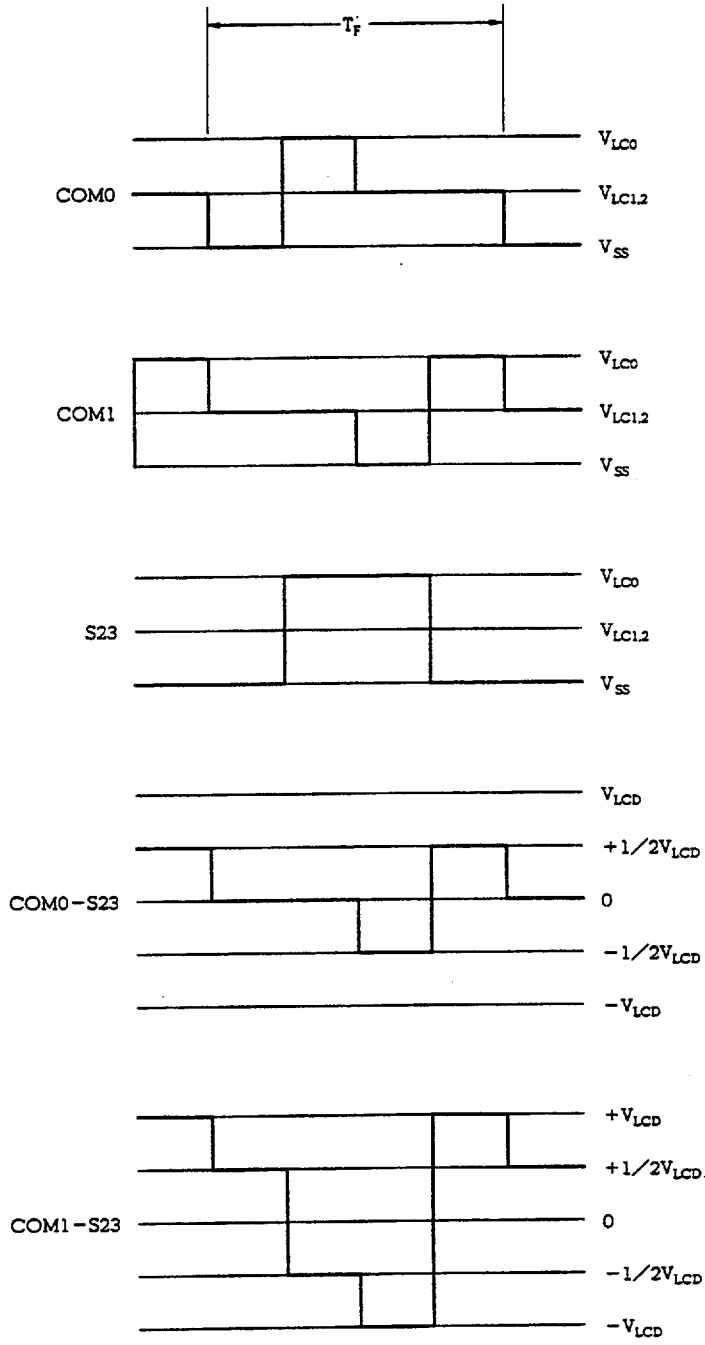


Figure 5-89 2-Time Multiplexing LCD Panel Wiring Example



x: Because of 2-time multiplexing display, any data can always be stored.

Figure 5-90 2-Time Multiplexing LCD Drive Waveform Example (1/2 Bias Method)



(3) 3-time multiplexing display example

Figure 5-92 shows connection of a 6-digit LCD panel based on the 3-time multiplexing method with a display pattern shown in Figure 5-91 to the uPD75336 segment signals (S12 to S29) and common signals (COM0 to COM2). The display example is 12345.6 and the display data memory contents (addresses 1ECH to 1FDH) corresponding to this display.

This section provides a description using the second digit 5. (5.) as an example. It is necessary to generate select and non-select voltages shown in Table 5-20 to the S15 to S17 pins at the COM0 to COM2 common signal timings in accordance with the display pattern in Figure 5-91.

Table 5-20 Select and Non-Select Voltages (COM0, 1, 2)

Segment / Common	S15	S16	S17
COM0	Non-select	Select	Select
COM1	Select	Select	Non-select
COM2	Select	Select	——

It is clear that X110 should be set in the display data memory (address 1EFH) corresponding to S15.

Figure 5-93 (1/2 bias method) and Figure 5-94 (1/3 bias method) show the S15 and common signal LCD drive waveforms. When S15 becomes a select voltage at the COM1/COM2 select timing, a $+V_{LCD}/-V_{LCD}$ AC rectangular waveform at LCD ON level is generated.

Figure 5-91 3-Time Multiplexing LCD Display Pattern and Electrode Wiring

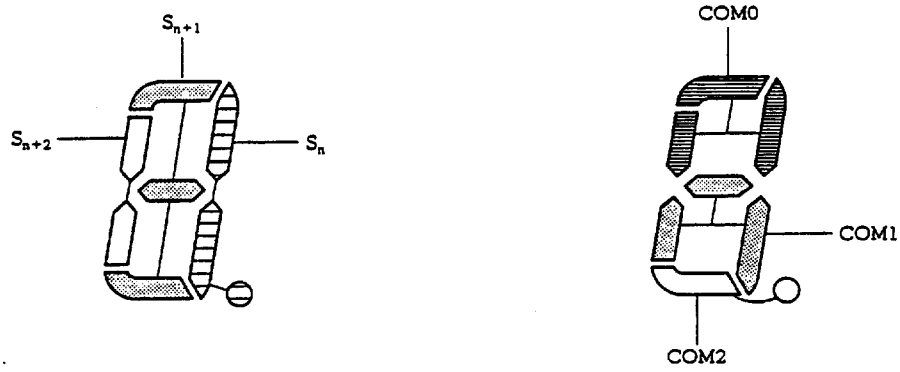
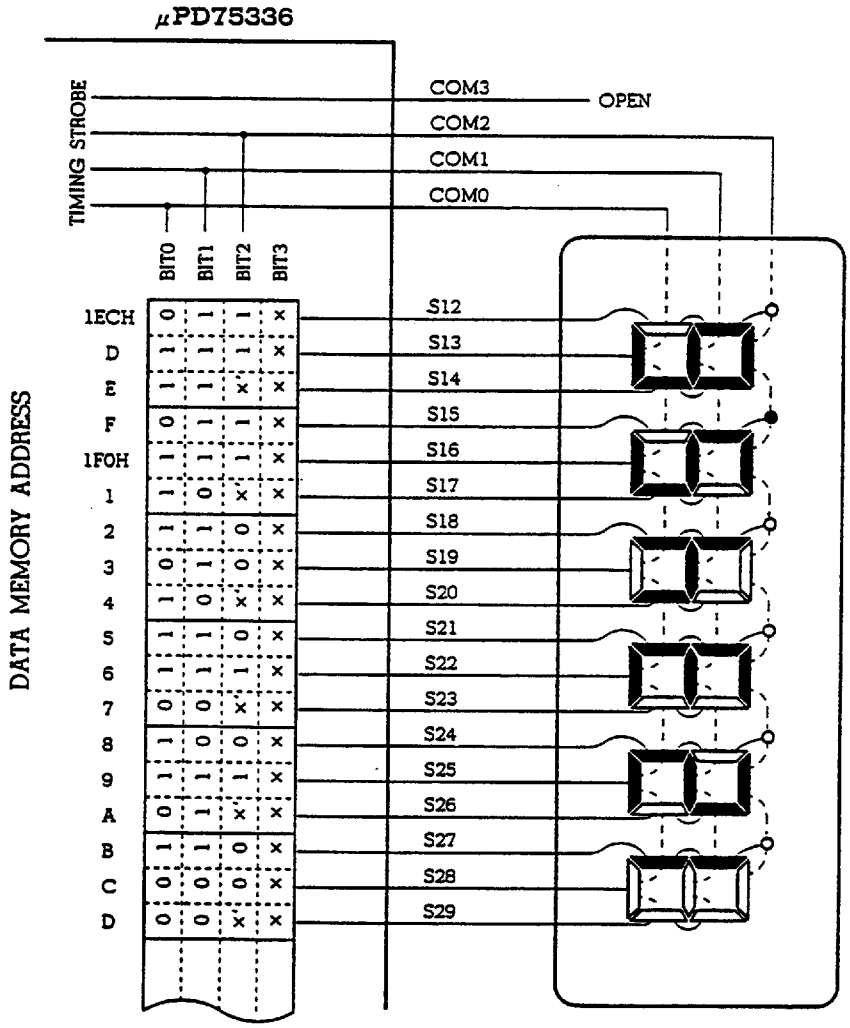


Figure 5-92 3-Time Multiplexing LCD Panel Wiring Example



LCD PANELx': Because there is no corresponding segment on the LCD panel, any data can be stored.

x : Because of 3-time multiplexing display, any data can always be stored.

Figure 5-93 3-Time Multiplexing LCD Drive Waveform Example (1/2 Bias Method)

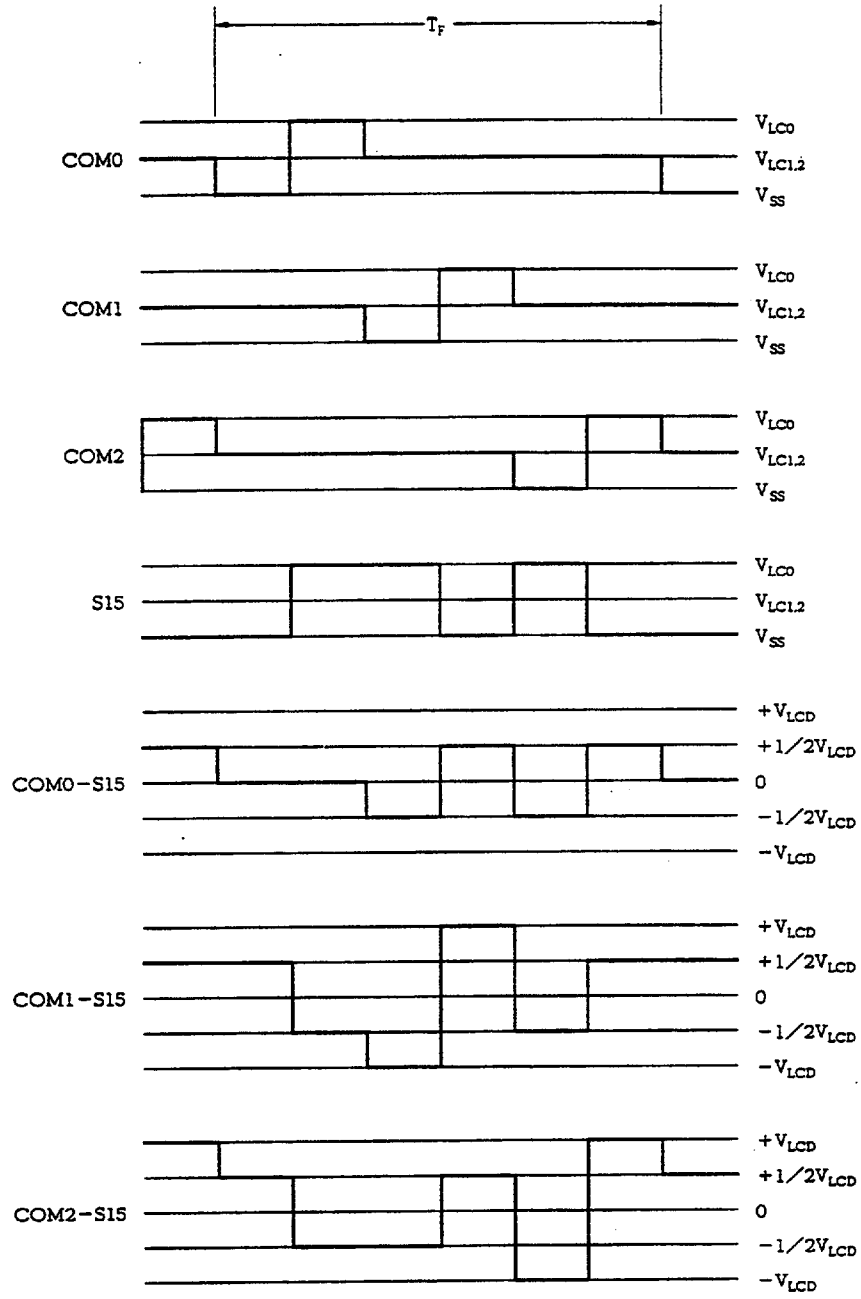
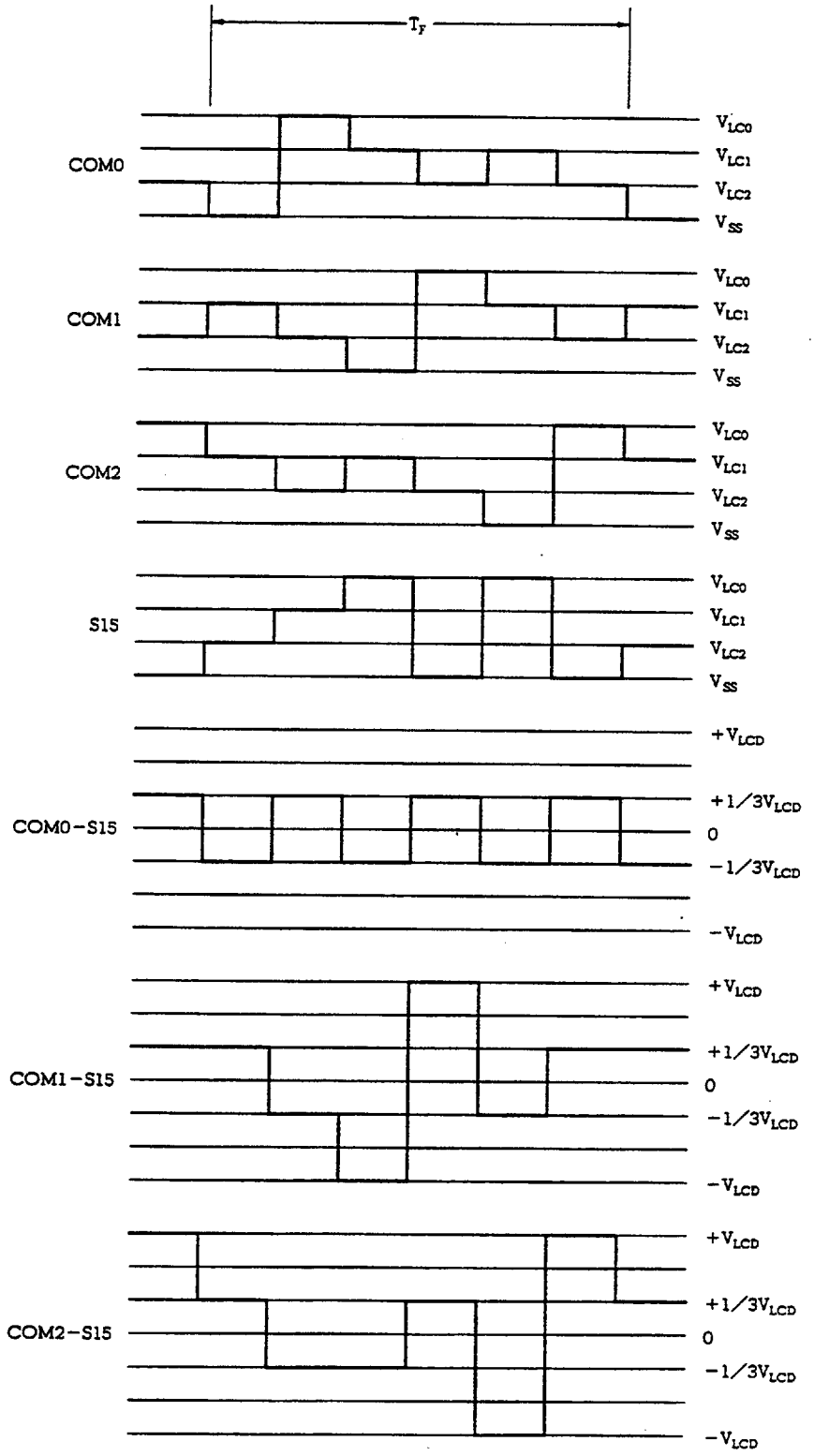


Figure 5-94 3-Time Multiplexing LCD Drive Waveform Example (1/3 Bias Method)



(4) 4-time multiplexing display example

Figure 5-96 shows connection of a 10-digit LCD panel based on the 4-time multiplexing method with a display pattern shown in Figure 5-95 to the uPD75336 segment signals (S12 to S31) and common signals (COM0 to COM3). The display example is 123456.7890 and the display data memory contents (addresses 1ECH to 1FFH) corresponding to this display.

This section provides a description using the fifth digit 6. (6.) as an example. It is necessary to generate select and non-select voltages shown in Table 5-21 to the S20 and S21 pins at the COM0 to COM3 common signal timings in accordance with the display pattern in Figure 5-95.

Table 5-21 Select and Non-Select Voltages (COM0 to COM3)

Segment / Common	S20	S21
COM0	Select	Select
COM1	Non-select	Select
COM2	Select	Select
COM3	Select	Select

It is clear that 1101 should be set in the display data memory (address 1F4H) corresponding to S20.

Figure 5-97 shows the S20 and COM0 and COM1 LCD drive waveforms (COM2 and COM3 waveforms are omitted for the sake of drawings). When S20 becomes a select voltage at the COM0 select timing, a $+V_{LCD}/-V_{LCD}$ AC rectangular waveform at LCD ON level is generated.

Figure 5-95 4-Time Multiplexing LCD Display Parameter and Electrode Wiring

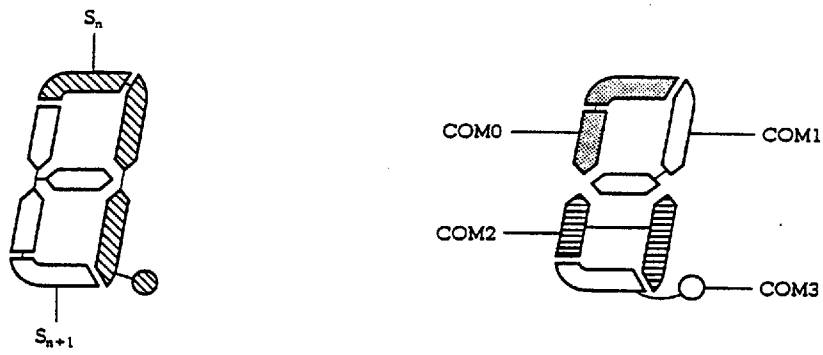


Figure 5-96 4-Time Multiplexing LCD Panel Wiring Example

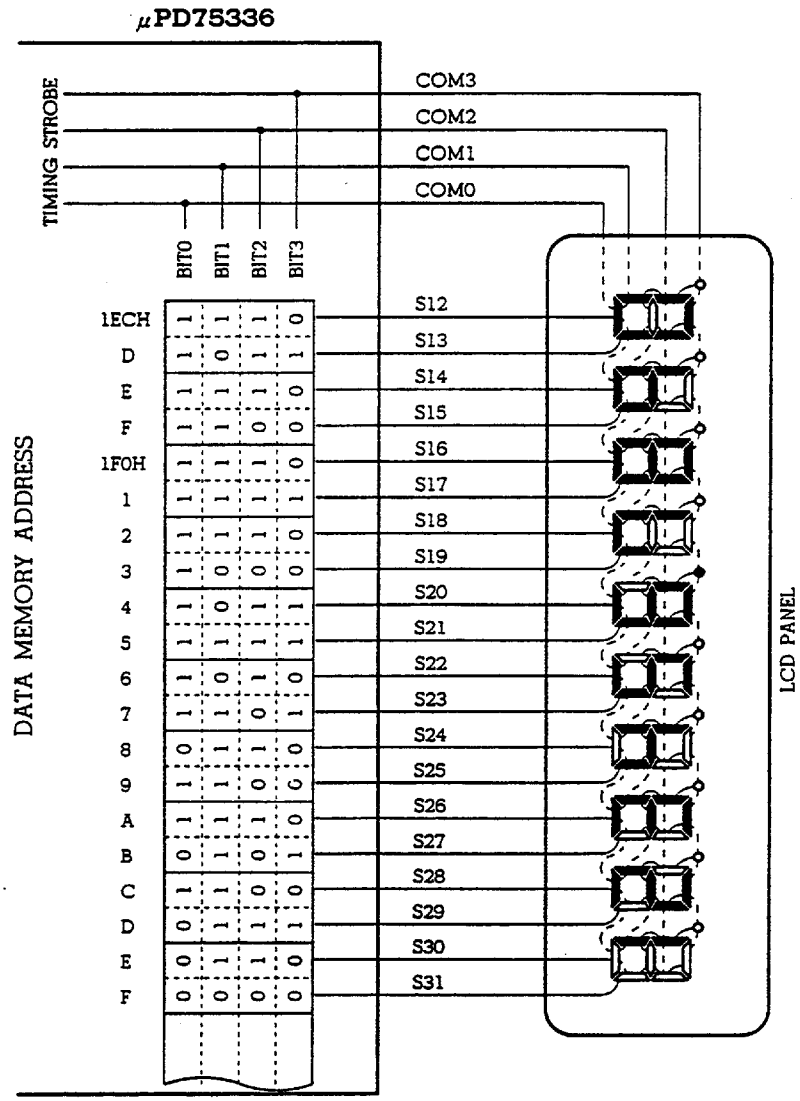
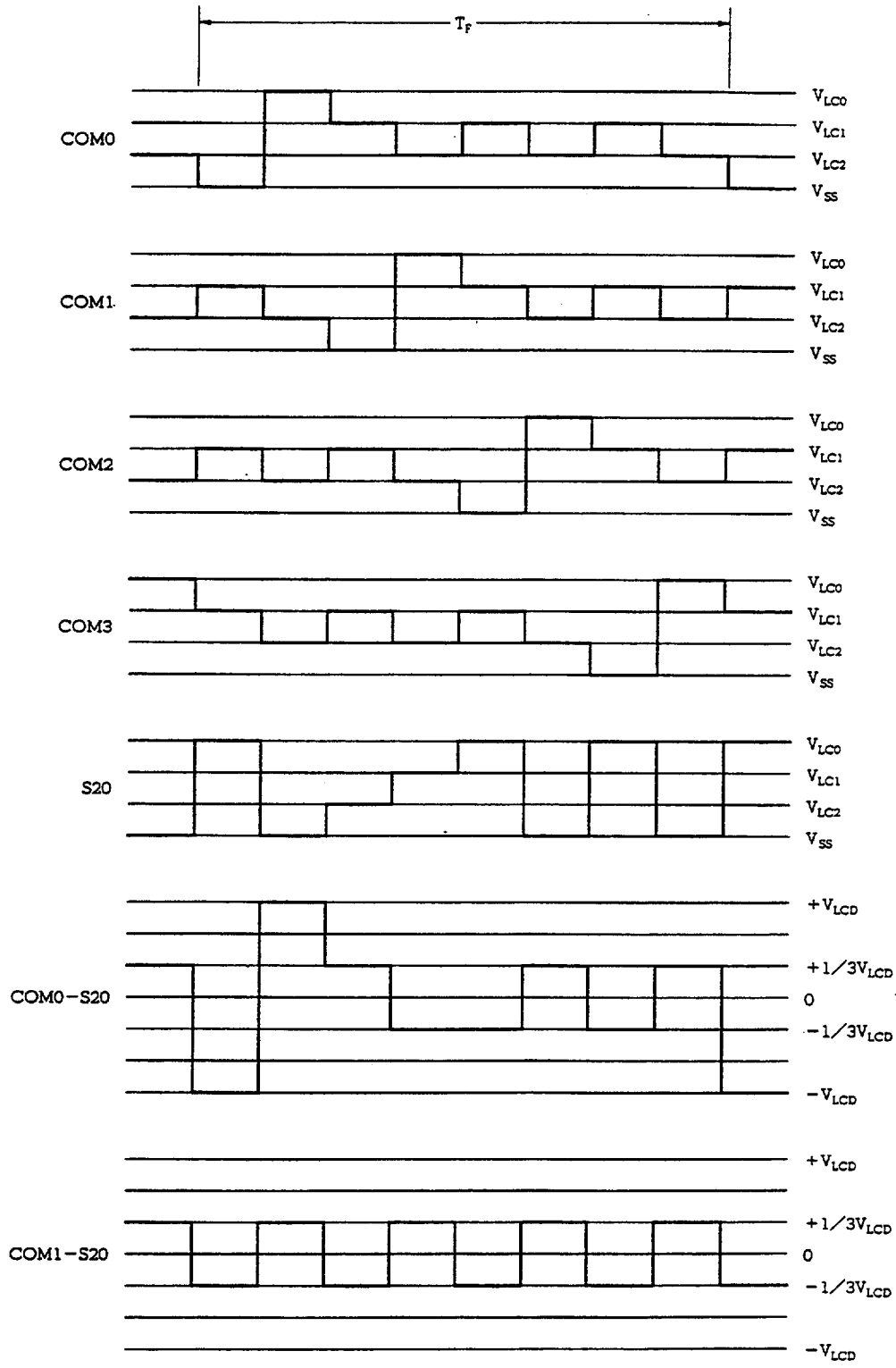


Figure 5-97 4-Time Multiplexing LCD Panel Waveform Example (1/3 Bias Method)



5.8 A/D CONVERTER

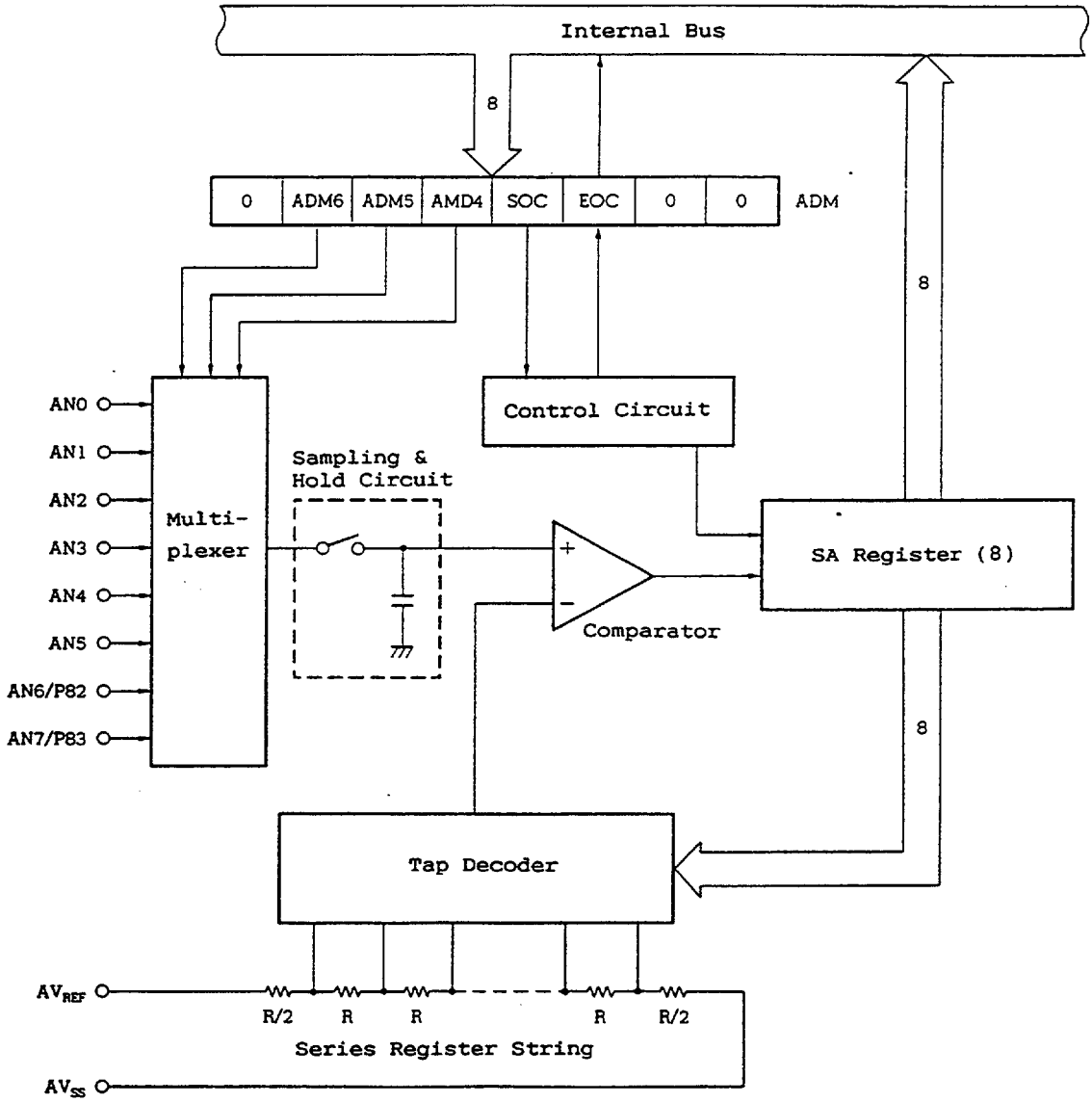
The uPD75336 incorporates an 8-bit precision analog/digital (A/D) converter having 8-channel analog inputs (AN0 to AN7).

The A/D converter employs the successive approximation method.

5.8.1 A/D CONVERTER CONFIGURATION

The A/D converter has the configuration shown in Figure 5-98.

Figure 5-98 A/D Converter Block Diagram



(1) A/D converter pins

(a) AN0 to AN7

8-channel analog signal input pins for the A/D converter. Analog signals to be converted from analog to digital are input to these pins.

AN6 and P82, and AN7 and P83 serve as dual-function pins in pairs.*

The A/D converter incorporates a sample & hold circuit and an analog input voltage is internally held during A/D conversion.

*: When AN6 and AN7 are used, the following settings are necessary before A/D conversion.

① Set port 8 to the input mode.
② Disable the on-chip pull-up resistor at port 8.

(For details, refer to Section 5.1 "DIGITAL INPUT/OUTPUT PORTS".)

NOTE: Use AN0 to AN7 input voltages within the specified range. If a voltage equal to or greater than V_{DD} or equal to or less than V_{SS} (even in the absolute maximum rating range) is input, the converted value of that channel becomes indeterminate and may cause effects on the converted value of another channel.

(b) AV_{REF}

Pin to which A/D converter reference voltage is input.

Signals input to AN0 to AN7 are converted to digital signals on the basis of voltages applied across AV_{REF} and AV_{SS}.

(c) AV_{SS}

A/D converter GND pin. It always should be set the same voltage as V_{SS}.

(2) A/D conversion mode register (ADM)

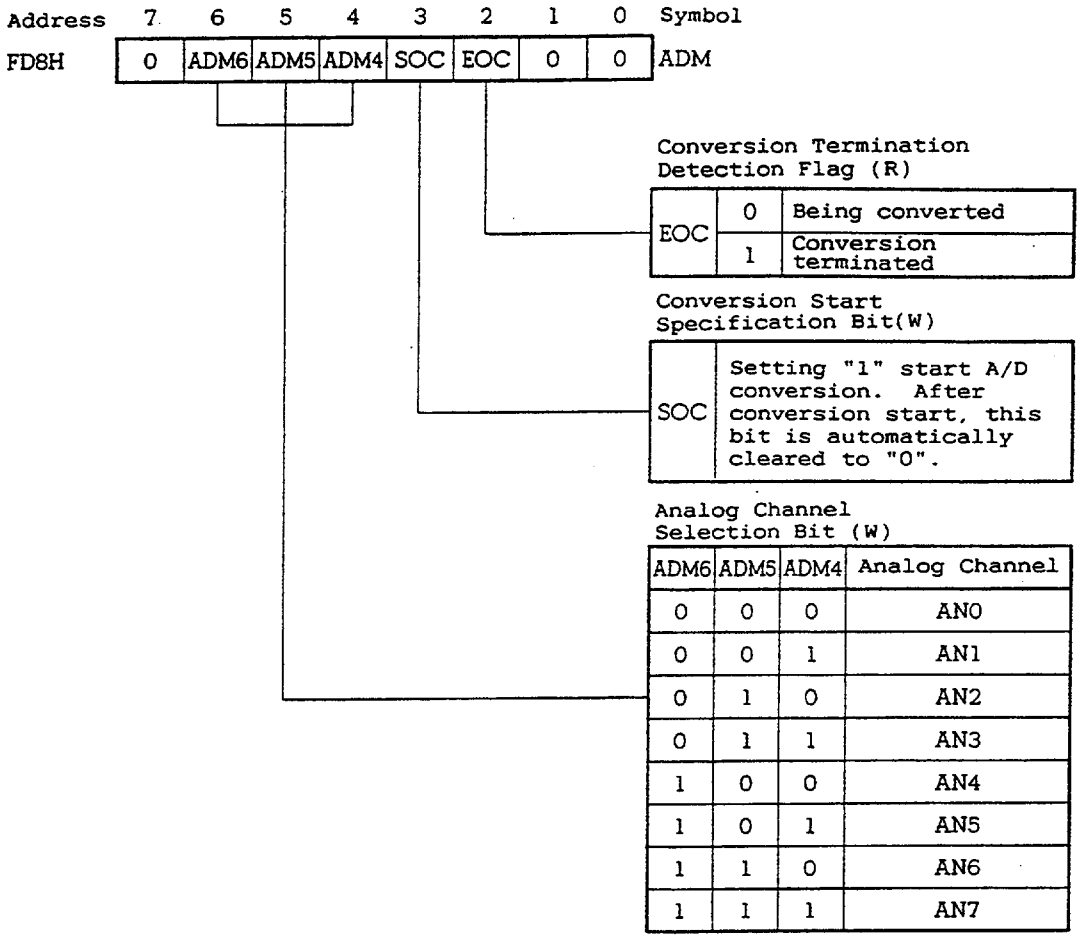
The ADM is an 8-bit register used to select the analog input channel, specify conversion start and detect the termination of conversion.

The ADM is set by an 8-bit manipulation instruction.

Bit 2 (EOC) and bit 3 (SOC) can be manipulated bit-wise.

When the $\overline{\text{RESET}}$ signal is generated , the ADM is initialized to 04H (only EOC is set to "1" and all other bits are cleared to "0").

Figure 5-99 A/D Conversion Mode Register Format



NOTE: A/D conversion starts with a maximum delay of 2 /f_X sec (3.81 us at f_X = 4.19 MHz) after SOC setting (refer to 5.8.2 "A/D Converter Operations").

(3) SA register(SA)

The SA (successive approximation)is an 8-bit register used to store A/D conversion results.

The SA is read by an 8-bit manipulation instruction. It is a read-only register and write operations and bit manipulation cannot be carried out.

When the $\overline{\text{RESET}}$ signal is generated, the SA is set to 7FH.

NOTE: When A/D conversion is started with the ADM register bit 3 (SOC) set to "1", the conversion results in the SA are destroyed and the SA remains indeterminate until new conversion results are stored.

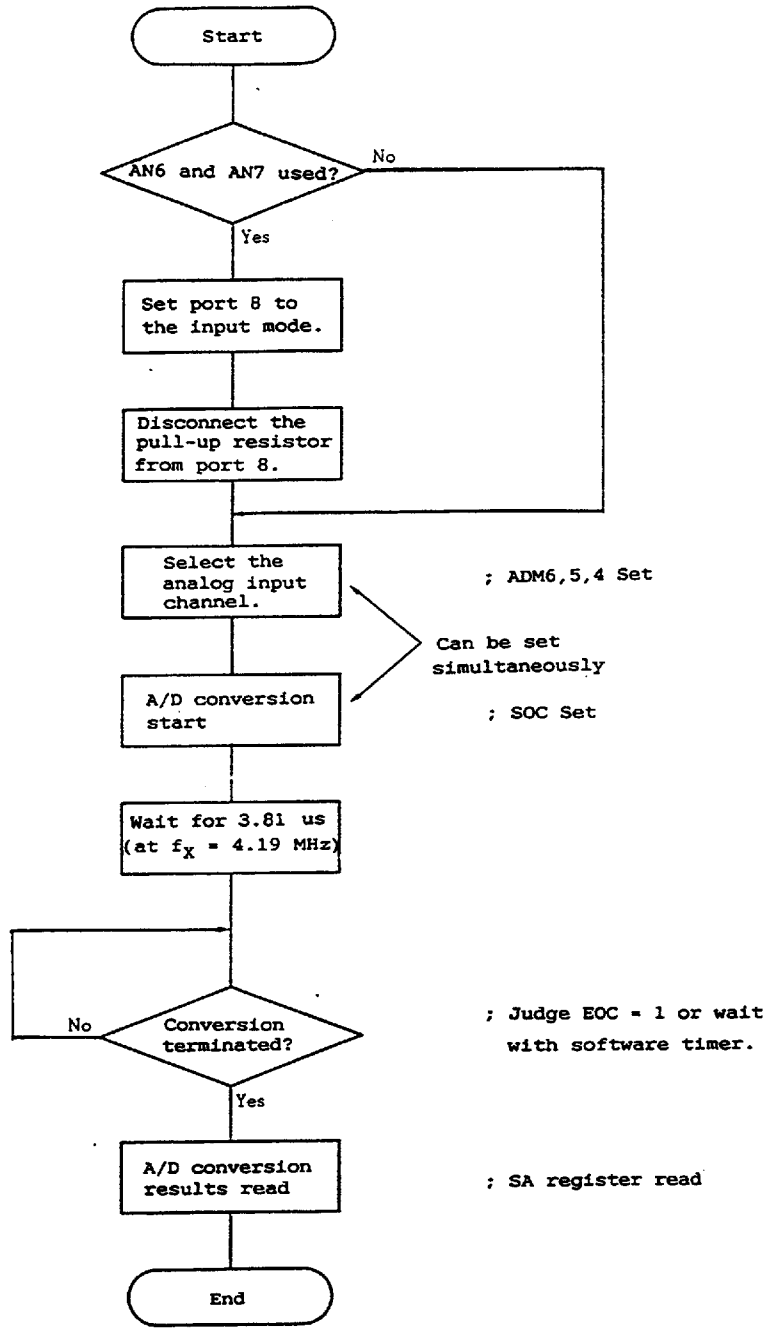
5.8.2 A/D CONVERTER OPERATIONS

The analog input signal to be converted to a digital signal is specified by setting bits 6, 5 and 4 (ADM6, 5, 4) of the A/D conversion mode register.

A/D conversion is started by setting ADM bit 3 (SOC) to "1". After setting, the SOC is automatically cleared to "0". A/D conversion is performed by hardware using the successive approximation method and the 8-bit conversion result data is stored in the SA register. Upon termination of conversion, ADM bit 2 (EOC) is set to "1".

Figure 5-100 shows an A/D conversion timing chart.

Operate the A/D converter using the following procedure.



NOTE: It takes the EOC a maximum of $2^4/fx$ (3.81 us at fx = 4.19 Mhz) to be cleared after A/D conversion start following SOC setting. Thus, execute the EOC test after the lapse of time specified in Table 5-22 following SOC setting. The A/D conversion time is also shown in Table 5-22.

Table 5-22 SCC and PCC Settings

SCC and PCC Set Values				A/D Conversion Time	Wait Time till EOC Test after SOC Setting	Wait Time till Termination of A/D Conversion after SOC Setting
SCC3	SCC0	PCC1	PCC0			
0	0	0	0	$168/f_X$ (40.1 us/ f_X = 4.19 MHz)	Wait not required	3 machine cycles
		0	1		1 machine cycle	11 machine cycle
		1	0		2 machine cycles	21 machine cycles
		1	1		4 machine cycles	42 machine cycles
0	1	x	x		Wait not required	Wait not required
1	x	x	x	Conversion operation stop	—	—

x: don't care

Figure 5-100 A/D Conversion Timing Chart

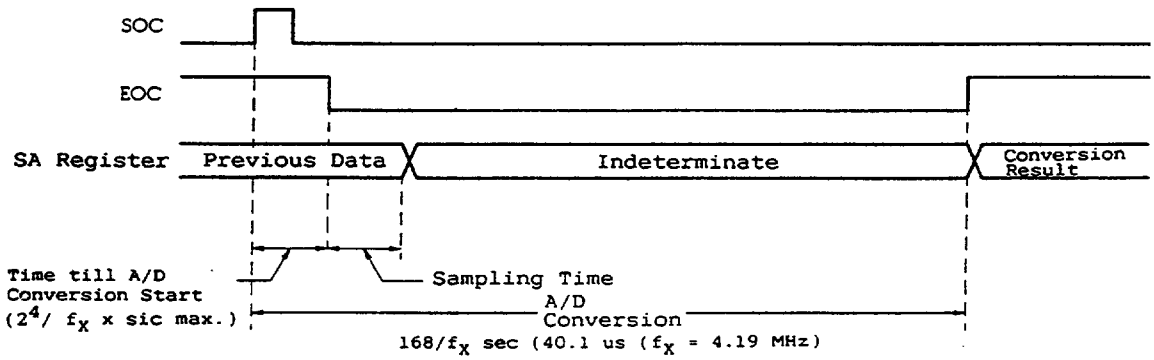
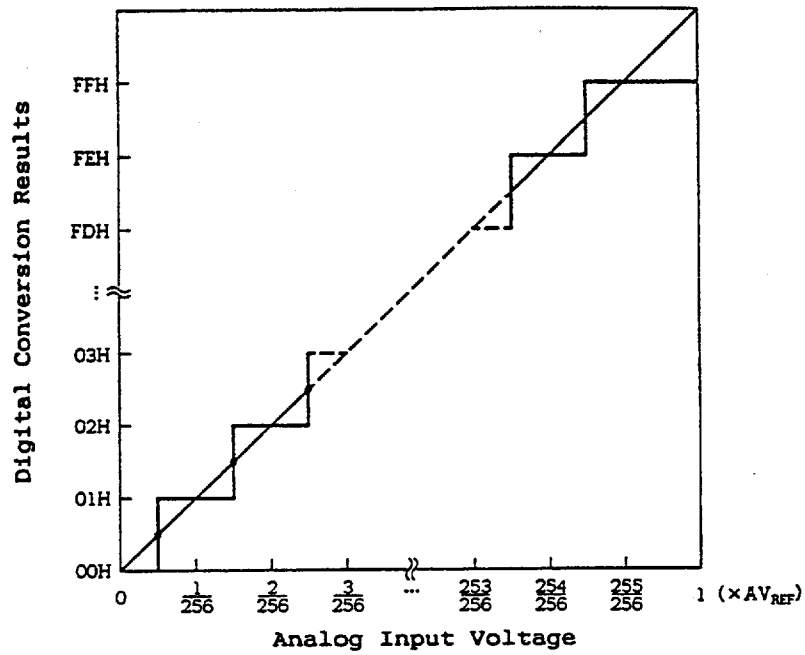


Figure 5-101 shows the relations between analog input voltages and A/D converted 8-bit digital data.

Figure 5-101 Relations between Analog Input Voltages and A/D Conversion Results (Ideal Case)

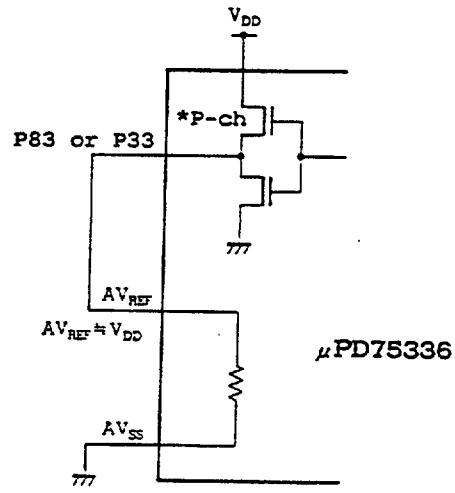


5.8.3 STANDBY MODE PRECAUTIONS

The A/D converter operates with the main-system clock. Thus, it stops operating in the STOP mode or the HALT mode with the subsystem clock. Because current flows to the AV_{REF} pin in this case also, it is necessary to cut this current to minimize the power consumption of the whole system. Since the P83 and P33 pins have higher drive capability than any other port, a voltage can be supplied directly to the AV_{REF} pin (see Figure 5-102). In this case, however, because the actual AV_{REF} voltage is not accurate, the converted value itself has no accuracy and can only be used for relative comparison. In the standby mode, power consumption can be minimized by generating a low level to P83 and P33.

The drive capability of the P83 and P33 pins of the uPD75390 peripheral hardware emulator used for emulation with EVAKIT is the same as that of any other port. Thus, when an example in Figure 5-102 is used, AV_{REF} for emulation becomes lower than AV_{REF} generated when uPD75336 is actually used, and the A/D conversion value does not become the same.

Figure 5-102 Example of How to Decrease Power Consumption in Standby Mode



*: The P83 and P33 pins have higher drive capability than any other port.

5.8.4 OPERATING PRECAUTIONS AND OTHERS

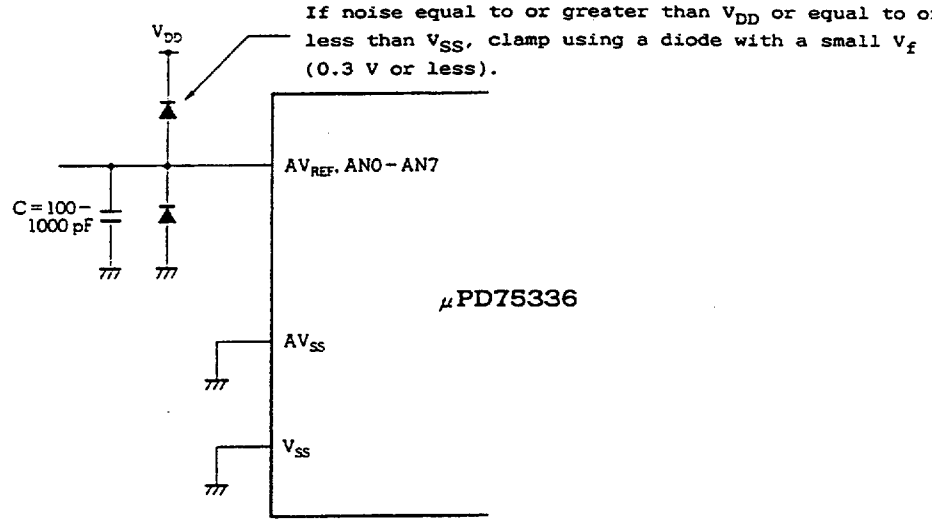
(a) AN0 to AN7 input ranges

Use AN0 to AN7 input voltages within the specified range. If a voltage equal to or greater than V_{DD} or equal to or less than V_{SS} (even in the absolute maximum range) is input, the converted value of that channel becomes indeterminate and may cause effects on the converted value of another channel.

(b) Countermeasure against noise

To maintain 8-bit precision, extra precaution must be taken against noise in the AV_{REF} and AN0 to AN7 pins. The higher the analog input source output impedance, the greater the effects. It is recommended to externally connect the C (see Figure 5-103), so as to reduce noise.

Figure 5-103 Analog Input Pin Treatment



(c) AN6/P82 and AN7/P83 pins

Analog input (AN6, AN7) pins serve as dual-function pins with input ports (P82, P83).

When performing A/D conversion with AN6 or AN7 selected, preset port 8 to the input mode and do not execute an input instruction during conversion, otherwise conversion accuracy may deteriorate.

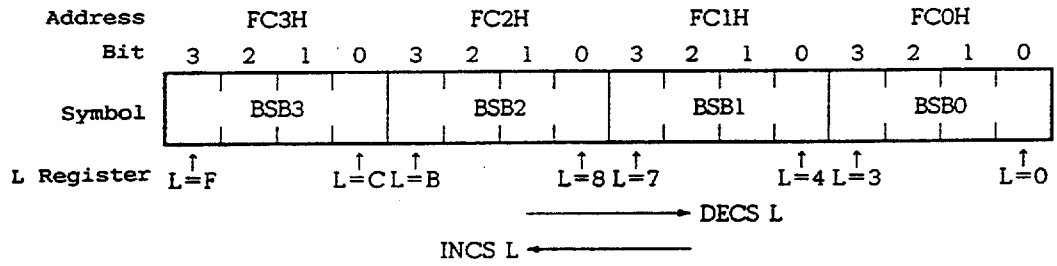
If a digital pulse is applied to pins next to the pin undergoing A/D conversion, the expected A/D conversion value may not be obtained because of coupling noise. Thus, do not apply pulses to the pins adjacent to the pin undergoing A/D conversion.

5.9 BIT SEQUENTIAL BUFFER: 16 BITS

The bit sequential buffers 0 to 3(BSB0 to BSB3) are special data memory for bit manipulation. Because bit manipulation can be performed easily by sequentially changing address and bit specification, this data memory is useful for bit-wise when processing of data with long bit lengths.

This data memory has a 16-bit configuration. It can perform pmem.@L addressing for a bit manipulation instruction and specify particular bits indirectly using the L register. In this case, processing can be carried out by sequentially shifting the specified bit simply by incrementing or decrementing the L register in the program loop.

Figure 5-104 Bit Sequential Buffer Format



Remarks 1: In pmem.@L addressing, the specified bit shifts in accordance with the L register.

2: In pmem.@L addressing, the BSB may operate independently of the MBE MBS specification.

Data can be operated in direct addressing as well. 1-bit data continuous input/output can be executed by combining 1-bit, 4-bit and 8-bit direct addressing and pmem.@L addressing. In the case of 8-bit manipulation, the most/least significant 8 bits are manipulated by specifying BSB0 and BSB2.

Example: Output 16-bit data of BUFF1 and BUFF2 serially from bit 0 of port 3.

```
        CLR1  MBE
        MOV   XA, BUFF1
        MOV   BSB0, XA ; BSB0 and BSB1 set
        MOV   XA, BUFF2
        MOV   BSB2, XA ; BSB2 and BSB3 set
        MOV   L, #0
LOOP0:  SKT   BSB0, @L ; BSB specified bit test
        BR    LOOP1
        NOP            ; Dummy (timing
                         adjustment)
        SET1  PORT3.0  ; Port 3 bit 0 set
        BR    LOOP2
LOOP1:  CLR1  PORT3.0  ; Port 3 bit 0 clear
        NOP            ; Dummy (timing
                         adjustment)
        NOP
LOOP2:  INCS  L        ; L ← L + 1
        BR    LOOP0
              RET
```

CHAPTER 6. INTERRUPT FUNCTIONS

On the uPD75336 there are 7 vectored interrupt sources and two testable inputs, enabling a wide variety of applications to be handled.

Moreover, the uPD75336's interrupt control circuit has the following special features, making possible extremely fast interrupt servicing.

(a) Acknowledgment enabling/disabling can be controlled by means of the interrupt master enable flag (IME) and the interrupt enable flags (IExxx).

(b) The interrupt service start address and MBE/RBE during interrupt servicing can be set arbitrarily using the vector table (for rapid starting of the actual interrupt service program).

(c) It is possible to raise the priority of any single interrupt source and enable multiple interrupts from that source.

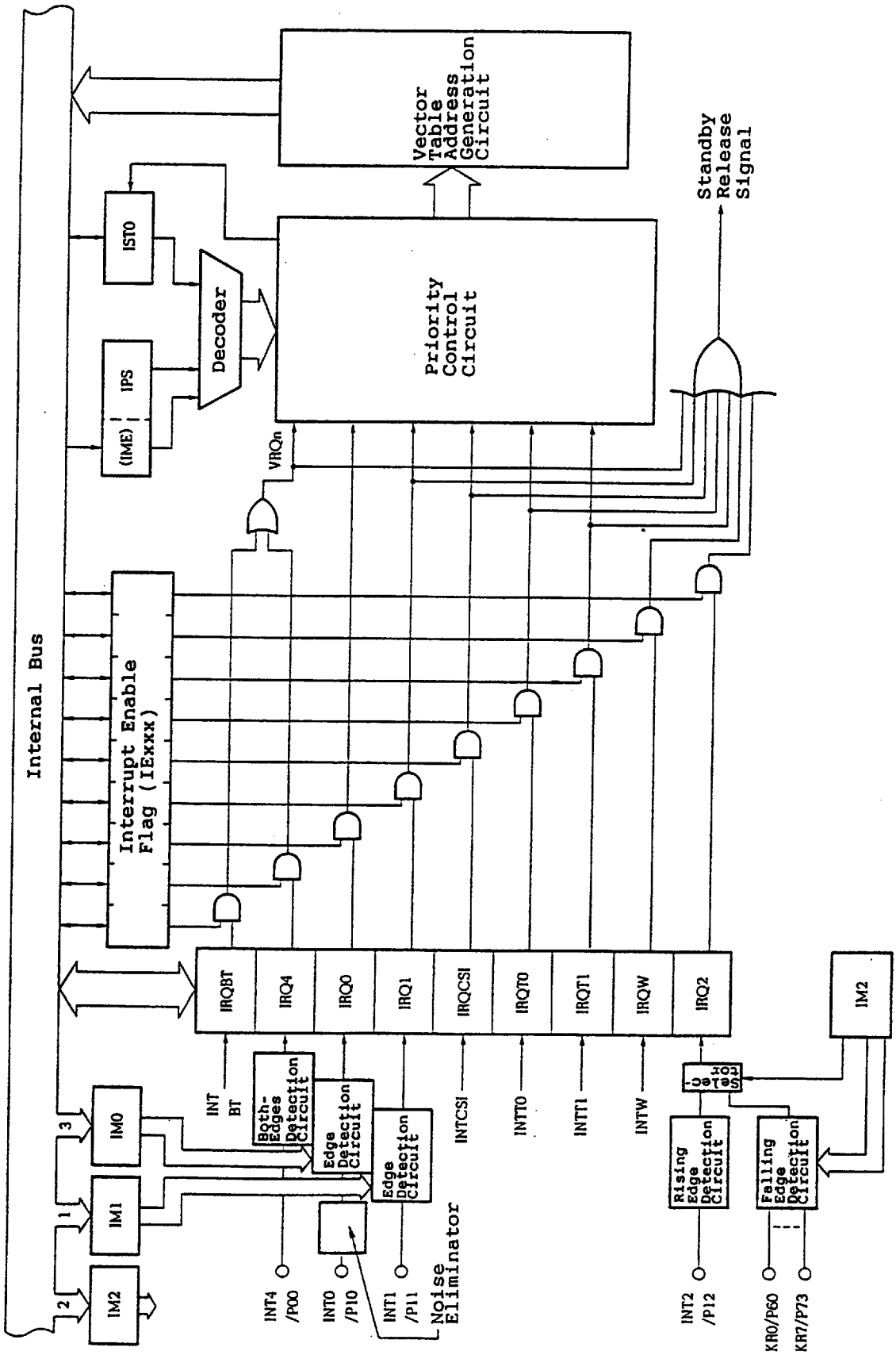
(d) Interrupt request flags (IRQxxx) can be tested and cleared (allowing checking of interrupt generation by software).

(e) Standby mode (STOP/HALT) can be released by an interrupt request (release source is selectable by means of an interrupt enable flag).

6.1 INTERRUPT CONTROL CIRCUIT CONFIGURATION

The interrupt control circuit is configured as shown in Figure 6-1, with each hardware item mapped onto data memory space.

Figure 6-1 Interrupt Control Circuit Block Diagram



6.2 INTERRUPT SOURCE TYPES AND VECTOR TABLE

The uPD75336's interrupt source types and interrupt vector table are shown in Table 6-1 and Figure 6-2.

Table 6-1 Interrupt Source Types

Interrupt	Source	Internal/ External	Interrupt Priority *1	Vectored Interrupt Request Signal (Vector Table Address)
INTBT	(Basic time interval signal from basic interval timer)	Internal	1	VRQ1 (0002H)
INT4	(Both rising edge and falling edge detection valid)	External		
INT0	(Rising/falling edge detection selection)	External	2	VRQ2 (0004H)
INT1 (0006H)		External	3	VRQ3
INTCSI	(Serial data transfer termination signal)	Internal	4	VRQ4 (0008H)
INTTO	(Match signal for programmable timer/ counter 0 count register and modulo register)	Internal	5	VRQ5 (000AH)
INTT1	(Match signal for programmable timer/ counter 1 count register and modulo register)	Internal	6	VRQ6 (000CH)
INT2 *3	(INT2 pin input rising edge detection or rising edge detection on any input to KR0 through KR7 *2)	External	Testable input signal (sets IRQ2 & IRQW)	
INTW *3	(Signal from clock timer)	Internal		

*1: The interrupt priority is the order of precedence when multiple interrupt requests occur simultaneously.

2: For details of INT2, see 6.3 (4) "INT2 and key interrupt 0 to 7(KR0 to KR7) hardware".

3: Test source. This is affected by the interrupt enable flag as well as the interrupt source, however, the vectored interrupt is not generated.

Figure 6-2 Interrupt Vector Table

Address			
0002H	MBE	RBE	INTBT/INT4 Start Address (High-Order 6 Bits)
			INTBT/INT4 Start Address (Low-Order 8 Bits)
0004H	MBE	RBE	INT0 Start Address (High-Order 6 Bits)
			INT0 Start Address (Low-Order 8 Bits)
0006H	MBE	RBE	INT1 Start Address (High-Order 6 Bits)
			INT1 Start Address (Low-Order 8 Bits)
0008H	MBE	RBE	INTCSI Start Address (High-Order 6 Bits)
			INTCSI Start Address (Low-Order 8 Bits)
000AH	MBE	RBE	INTTO Start Address (High-Order 6 Bits)
			INTTO Start Address (Low-Order 8 Bits)
000CH	MBE	RBE	INTT1 Start Address (High-Order 6 Bits)
			INTT1 Start Address (Low-Order 8 Bits)

The interrupt priority shown in the table shows the order in which interrupts are executed when multiple interrupt requests are generated simultaneously or when multiple interrupt requests are pending.

The vector table contains the interrupt service routine start addresses and the set value of the MBE, RBE with interrupt servicing in progress. Vector table setting is performed by the VENTn assembler pseudo-instruction.

Example: INTBT/INT4 vector table setting.

```
VENT1 MBE = 0, RBE = 0, GOTOBT
  ①     ②       ③       ④
```

① Address 0002 vector table
② Interrupt service routine MBE setting
③ Interrupt service routine RBE setting
④ Symbol indicating start address of interrupt service routine

NOTE: The vector table address specified by VENTn (n = 1 to 6) is address 2n.

Example: INTBT/INT4 and INTTO vector table setting.

```
VENT1   MBE = 0, RBE = 0, GOTOBT
VENT5   MBE = 0, RBE = 1, GOTOTO
```

6.3 INTERRUPT CONTROL CIRCUIT HARDWARE

(1) Interrupt request flags & interrupt enable flags

Nine types of interrupt request flags (IRQxxx) as shown below are provided corresponding to interrupt sources (interrupts: 7, test: 2).

INT0 Interrupt Request Flag (IRQ0)
INT1 Interrupt Request Flag (IRQ1)
INT2 Interrupt Request Flag (IRQ2)
INT4 Interrupt Request Flag (IRQ4)
BT Interrupt Request Flag (IRQBT)
Serial Interface Interrupt Request Flag (IRQCSI)
Timer/Event Counter 0 Interrupt Request Flag (IRQT0)
Timer/Event Counter 1 Interrupt Request Flag (IRQT1)
Watch Timer Interrupt Request Flag (IRQW)

An interrupt request flag is set "1" by the generation of an interrupt request and cleared "0" automatically when the interrupt service routine is executed. However, since IRQBT and IRQ4 share a vector address, the clear operation is different (see 6.6 "Vector Address Sharing Interrupt Servicing").

Nine types of interrupt enable flags (IExxx) as shown below are provided corresponding to interrupt sources.

INT0 Interrupt Enable Flag (IE0)
INT1 Interrupt Enable Flag (IE1)
INT2 Interrupt Enable Flag (IE2)
INT4 Interrupt Enable Flag (IE4)
BT Interrupt Enable Flag (IEBT)
Serial Interface Interrupt Enable Flag (IECSI)
Timer/Event Counter 0 Interrupt Enable Flag (IET0)
Timer/Event Counter 1 Interrupt Enable Flag (IET1)
Watch Timer Interrupt Enable Flag (IEW)

When the interrupt enable flag contents are "1", interrupts are enabled, when "0", interrupts are disabled.

When an interrupt request flag is set and the interrupt enable flag permits an interrupt, a vectored interrupt request (VRQn) is generated. This signal is also used to release standby mode.

The interrupt request flags and interrupt enable flags are manipulated by bit manipulation instructions and 4-bit memory manipulation instructions. In the case of bit manipulation instructions, direct manipulation is always possible irrespective of the MBE setting. In addition, the interrupt enable flags are manipulated by the EI IExxx instruction and the DI IExxx instruction. The SKTCLR is normally used for interrupt request flag testing.

```
Example:  EI       IE0    ; INT0 enabled
          DI       IE1    ; INT1 disabled
          SKTCLR   IRQCSI; skip and clear if IRQCSI is
                           1
```

When the interrupt request flag is set by an instruction, although no interrupt is generated, a vectored interrupt is executed in the same way as when an interrupt is generated.

The IRQ0, IRQBT, IRQCSI, IRQT0, IRQT1 and IRQW interrupt request flags are cleared (0) by generation of a $\overline{\text{RESET}}$ signal. IRQ1, IRQ2 and IRQ4 are undefined after $\overline{\text{RESET}}$ signal input, and should therefore be cleared (0) by software. □

The interrupt enable flags are cleared (0) by generation of a $\overline{\text{RESET}}$ signal. All interrupts are then disabled.

Table 6-2 Interrupt Request Flag Setting Signals

Interrupt Request Flag	Interrupt Request Flag Setting Signal	Interrupt Enable Flag
IRQBT	Set by basic time interval signal from basic interval timer.	IEBT
IRQ4	Set by INT4/P00 pin input signal rising or falling edge detection.	IE4
IRQ0	Set by INT0/P10 pin input signal edge detection. Detected edge is selected by INT0 mode register (IM0).	IE0
IRQ1	Set by INT1/P11 pin input signal edge detection. Detected edge is selected by INT1 mode register (IM1).	IE1
IRQCSI	Set by serial interface serial data transfer operation termination signal.	IECSI
IRQT0	Set by match signal from timer/event counter #0.	IET0
IRQT1	Set by match signal from timer/event counter #1.	IET1
IRQW	Set by signal from clock timer.	IEW
IRQ2	Set by INT2/P12 pin input rising edge detection or detection of falling edge of any input to pins KR0/P60 through KR7/P73.	IE2

(2) Interrupt priority selection register (IPS)

The interrupt priority selection register is used to select the "high-ranking interrupt" with multiple interrupt capability, this being specified by the low-order 3 bits.

Bit 3 is the interrupt master enable flag (IME) which specifies whether or not all interrupts are disabled.

IPS is set by a 4-bit memory manipulating instruction, but bit 3 is set/reset by an EI/DI instruction.

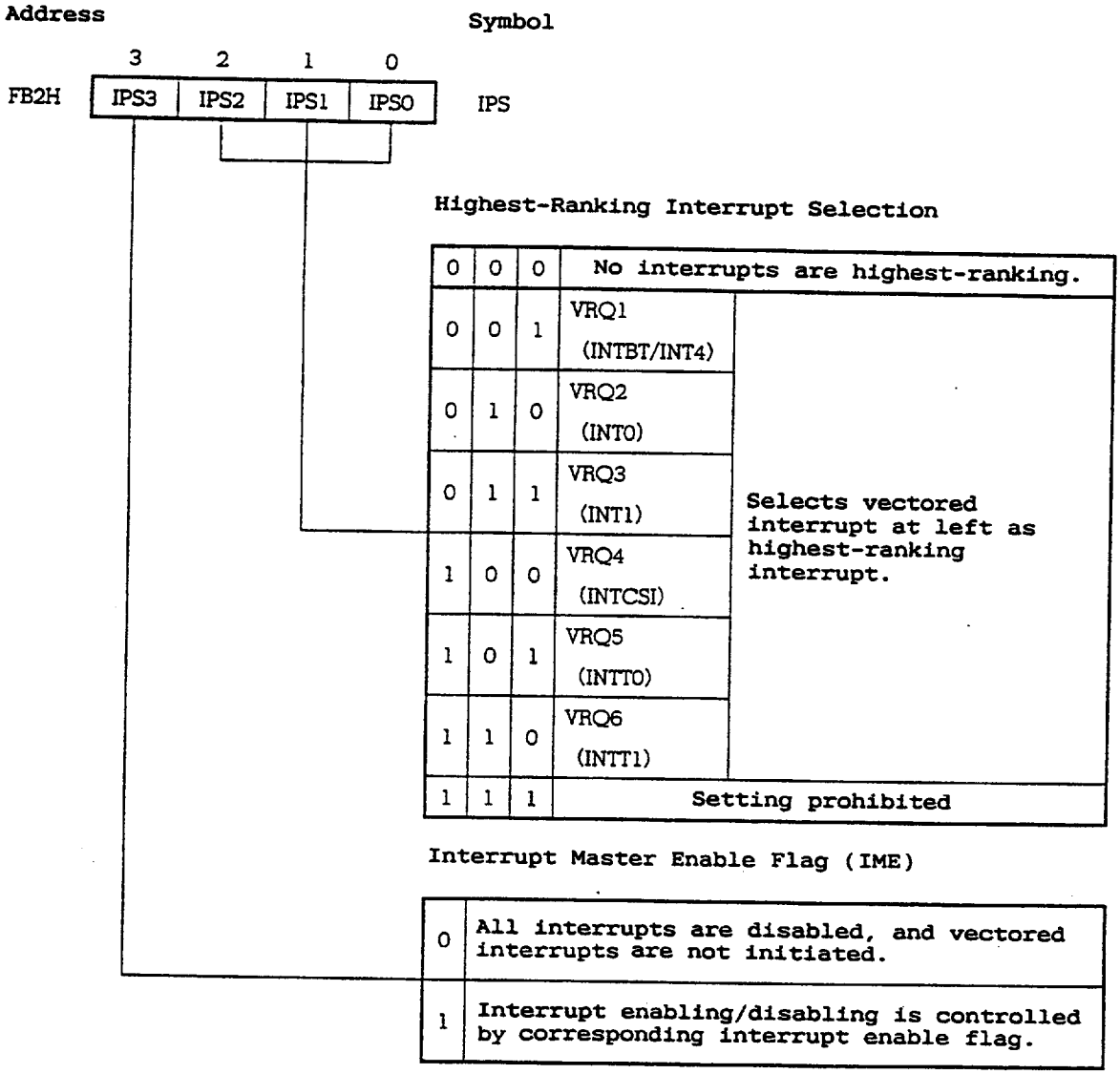
When the low-order 3 bits of IPS are modified, this should always be performed when interrupts are disabled (IME = 0).

```
Example:  DI              ; Disable interrupts
          CLR1  MBE
          MOV   A, #1011
          MOV   IPS, A    ; Make INT1 high-ranking
                            interrupt, and enable
                            interrupts
```

$\overline{\text{RESET}}$ signal generation clears all bits to "0".

The format of the interrupt priority selection register is shown in Figure 6-3.

Figure 6-3 Interrupt Priority Selection Register



(3) INT0, INT1 and INT4 hardware

(a) The configuration of INT0 is shown in Figure 6-4 (a). It is an external interrupt input for which rising edge or falling edge detection can be selected.

The INT0 pin also has a function for noise elimination by means of a sampling clock (refer to Figure 6-5 "Noise Elimination Circuit Input/ Output Timing"). In the noise elimination circuit, pulses narrower than 2 cycles* of the sampling clock are eliminated as noise. However, pulses greater than 1 cycle of the sampling clock may be acknowledged depending on sampling timing (refer to Figure 6-5 ② (a)). Pulses exceeding twice the width of the sampling clock can be surely acknowledged as interrupt signals.

In INT0, either Φ or $f_X/64$ sampling clock can be selected. The selection is performed by the edge detection mode register bit 3 (IM03) (refer to Figure 6-6 (a)).

Detected edge selection is performed by means of the edge detection mode register bit 0 (IM00) and bit 1 (IM01).

The format of IM0 is shown in Figure 6-6 (a). IM0 is set by a 4-bit manipulation instruction. Upon reset signal generation, all bits are cleared to "0" and rising edge detection is specified.

*: When sampling clock is Φ : $2t_{CY}$
When sampling clock is $f_X/64$: $128/f_X$

NOTE 1: Since INT0 sampling is performed by means of a clock, it does not operate in standby mode.

2: When INT0/P10 pin is input, it is input via noise elimination circuit. Therefore, pulses exceeding twice the width of the sampling clock also should be input.

(b) The configuration of INT1 is shown in Figure 6-4 (b). It is an external interrupt input for which rising edge or falling edge detection can be selected.

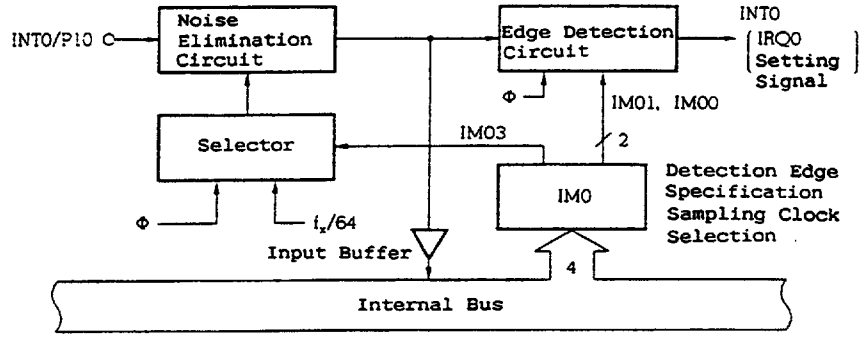
Detected edge selection is performed by means of the edge detection mode register (IM1).

The format of IM1 is shown in Figure 6-6 (b). IM1 is set by 4-bit manipulation instruction. Upon reset signal generation, all bits are cleared to "0" and rising edge detection is specified.

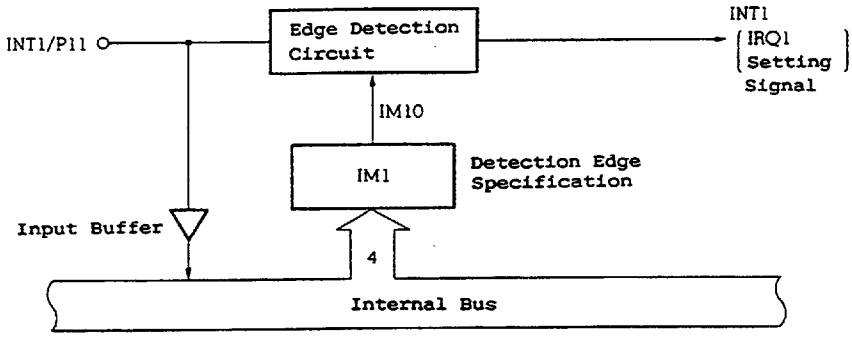
(c) The configuration of INT4 is shown in Figure 6-4 (c). It is an external interrupt input with both rising edge and falling edge detection capability.

Figure 6-4 Configuration of INT0, INT1 and INT4

(a) INT0 hardware



(b) INT1 hardware



(c) INT4 hardware

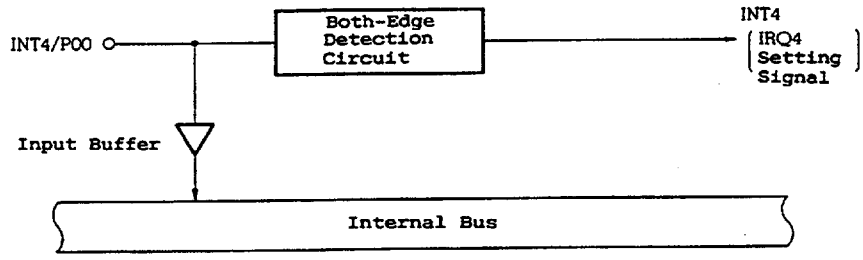
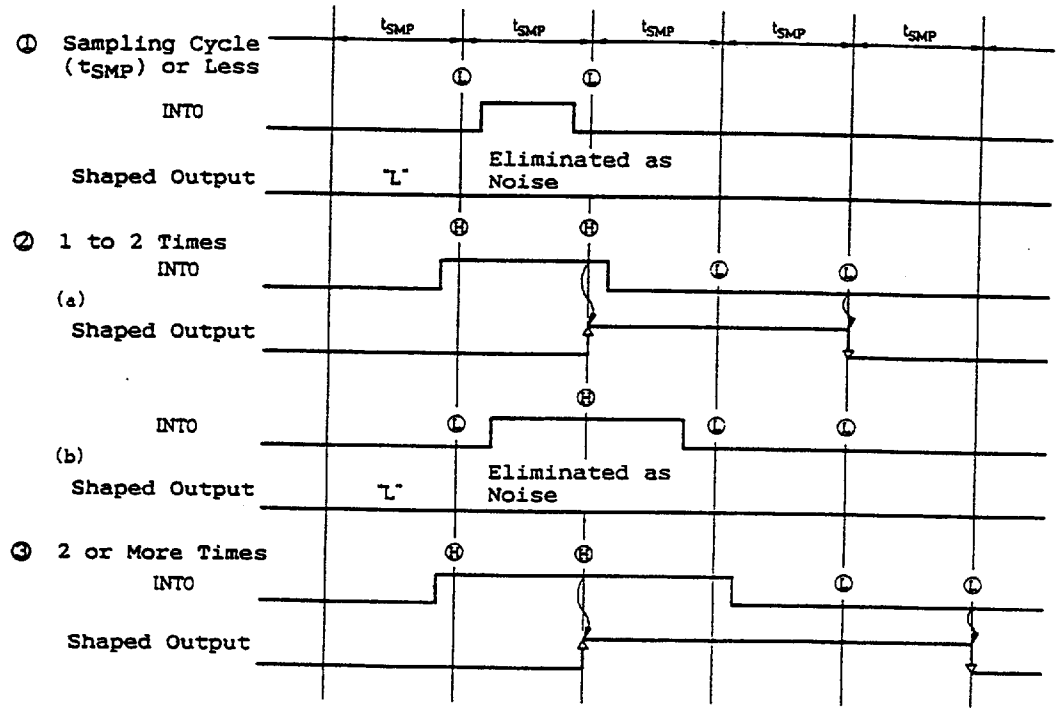


Figure 6-5 Noise Elimination Circuit Input/Output Timing



Remarks : $t_{SMP} = t_{CY}$ or $64/f_X$

Figure 6-6 Edge Detection Mode Register Format

(a) INT0 Edge Detection Mode Register (IM0)

Address: FB4H — Symbol: IM0

3	2	1	0
IM03	0	IM01	IM00

Detected Edge Specification

0	0	Rising edge Specification
0	1	Falling edge specification
1	0	Rising and falling edge specification
1	1	Ignored (Interrupt request flag is not set.)

Sampling Clock

0	Φ (0.95, 1.91, 3.81, 15.3 us: at 4.19 MHz operation)
1	f_X/64 (15.3 us: at 4.19 MHz operation)

(b) INT1 Edge Detection Mode Register (IM1)

FB5H — IM1

0	0	0	IM10

0	Rising edge specification
1	Falling edge specification

(c) INT2 Edge Detection Mode Register (IM2)

FB6H — IM2

0	0	IM21	IM20

IM21	IM20	INT2 Interrupt Source	Interrupt Input Pins
0	0	INT2 pin input rising edge specification	INT2 (1 pin)
0	1	Falling edge specification on any KRX pin input	KR4 to KR7 (4 pins)
1	0		KR2 to KR7 (6 pins)
1	1		KR0 to KR7 (8 pins)

A note is given on the following page.

NOTE: As the interrupt request flag may be set when the edge detection mode register is modified, the following procedure should be used: Disable interrupts and modify the mode register in advance, clear the interrupt request flag with the CLR1 instruction, and then enable interrupts again. Also, when $f_x/64$ is selected as the sampling clock by modifying IM0, the interrupt request flag must be cleared after the elapse of 16 machine cycles following the mode register modification.

(4) INT2 and key interrupt 0 to 7 (KR0 to KR7) hardware

The configuration of INT2 and KR0 to KR7 is shown in Figure 6-7. There are two types for IRQ2 setting, as follows. Either (a) or (b) can be selected by the edge detection mode resister (IM2).

(a) INT2 pin input rising edge detection

IRQ2 is set upon detection of an INT2 pin input rising edge.

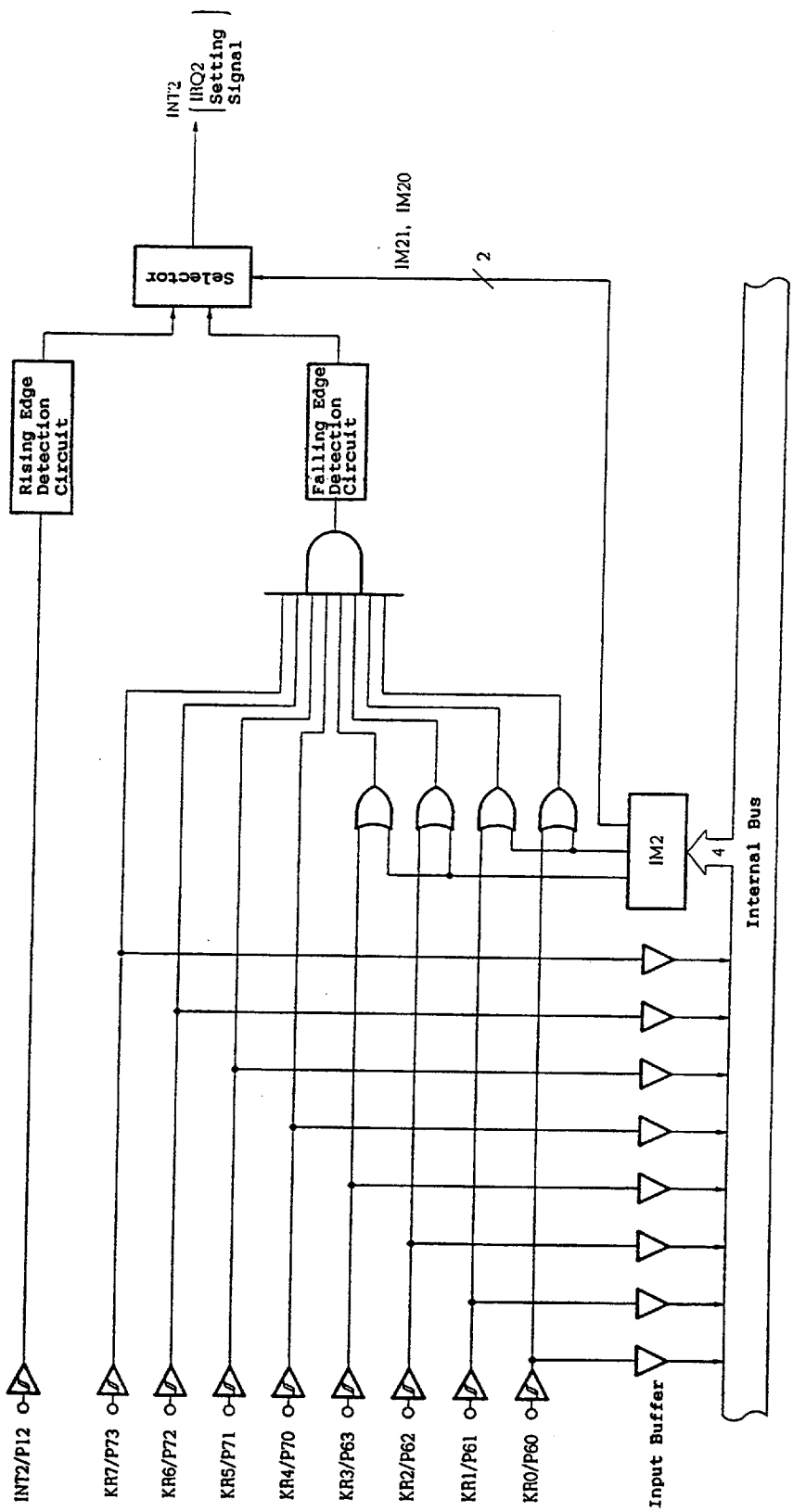
(b) Falling edge detection on any of pins KR0 to KR7 (key interrupt)

IRQ2 is set upon detection of a falling edge in the input of any of pins KR0 through KR7 selected by the edge detection mode register (IM2).

NOTE: If there is low-level input on even one of the pins selected for falling edge detection, IRQ2 is not set even if a falling edge is input on the other pins.

The format of IM2 is shown in Figure 6-6 (c). IM2 is set by a 4-bit manipulating instruction. Upon reset signal generation, all bits are cleared to "0" and rising edge detection is specified.

Figure 6-7 Configuration of INT2 and KR0 to KR7



(5) Interrupt status flags

The interrupt status flags (IST0, IST1) are flags in the PSW which indicate the status of the processing currently being executed by the CPU.

The interrupt priority control circuit controls multiple interrupts according to the contents of these flags as shown in Table 6-3.

Since IST0 and IST1 can be modified by bit manipulation instructions or 4-bit manipulation instructions, it is possible to perform multiple interrupts by changing the status during execution. IST0 and IST1 can be manipulated bit-wise at any time irrespective of the MBE setting.

When IST0 and IST1 are manipulated, it is always necessary to disable interrupts by executing a DI instruction prior to the manipulation, then to enable interrupts by execution of an EI instruction after the manipulation.

After IST1 and IST0 are saved to stack memory together with the rest of the PSW when an interrupt is acknowledged, their status is automatically changed to the next higher level. When an RETI instruction is executed, the original IST0/IST1 values are restored.

$\overline{\text{RESET}}$ signal generation clears the flag contents to "0".

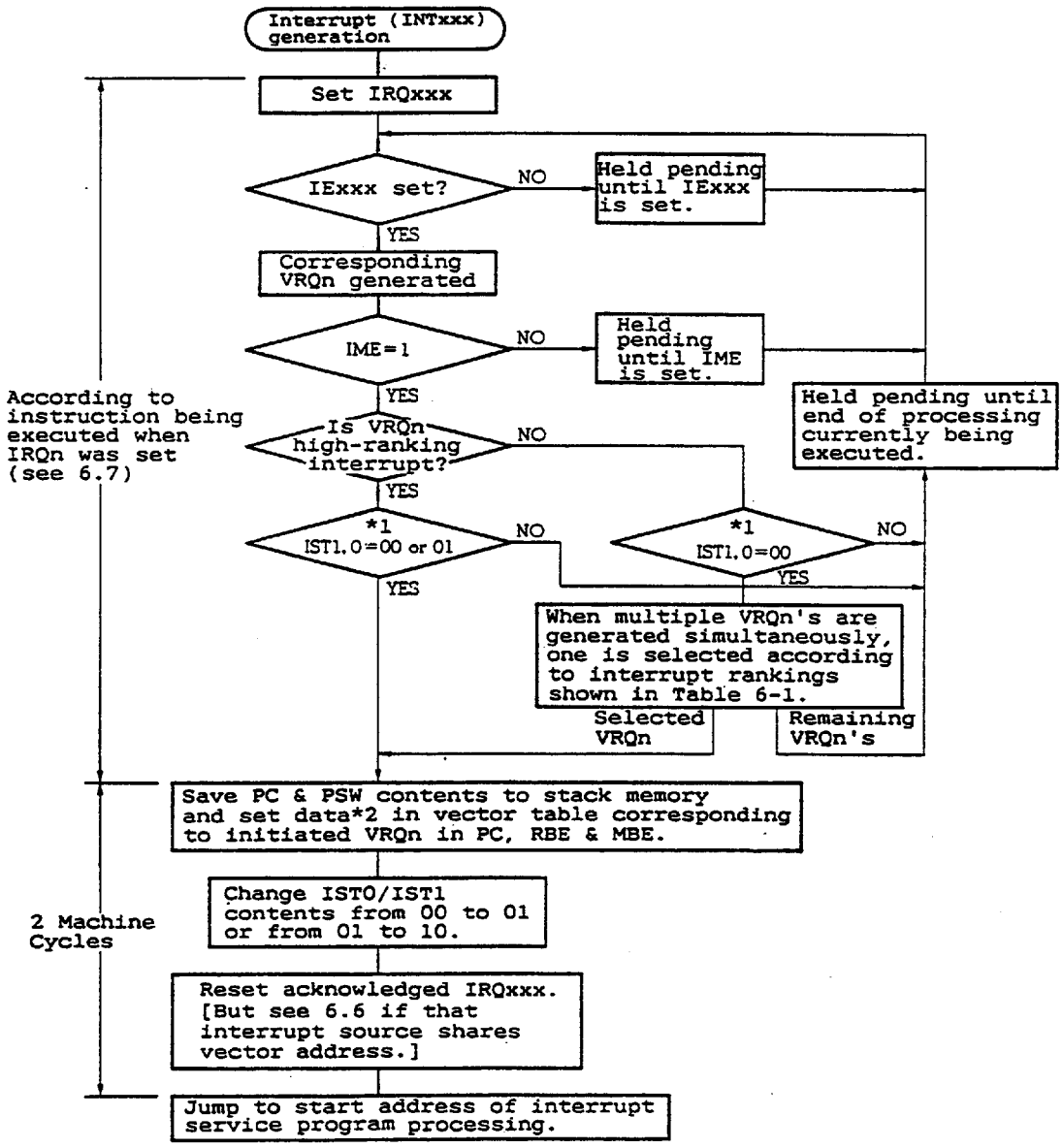
Table 6-3 IST1/IST0 Interrupt Servicing Status

IST1	IST0	Executing Processing Status	CPU Processing	Interrupt Requests for which Interrupt Acknowledgment is Possible	After Interrupt Acknowledgement	
					IST1	IST0
0	0	Status 0	Normal program processing in progress	Acknowledgment of all interrupts possible	0	1
0	1	Status 1	Low-ranking interrupt or high-ranking interrupt servicing in progress	Only high-ranking interrupts can be acknowledged	1	0
1	0	Status 2	High-ranking interrupt servicing in progress	Acknowledgment of all interrupts disabled	-	-
1	1	Setting prohibited				

6.4 INTERRUPT SEQUENCE

When an interrupt is generated, it is serviced by the following procedure.

Figure 6-8 Interrupt Service Sequence



*1: IST0, IST1: Interrupt status flags (PSW bits 3 & 2: See Table 6-3.)

2: Each vector table contains the start address of the interrupt service program and MBE & RBE set values when the interrupt starts.

6.5 MULTIPLE INTERRUPT SERVICE CONTROL

On the uPD75336, multiple interrupts can be performed in two ways, as shown below.

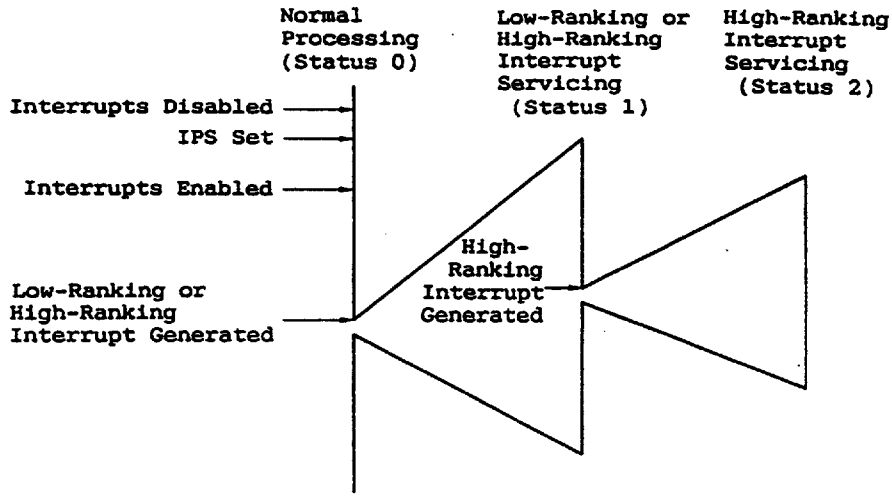
(1) Multiple interrupts with high-ranking interrupt specified

This is the basic multiple interrupt method on the uPD75336; one of the interrupt sources is selected and multiple interrupts (dual interrupts) are enabled for that interrupt source.

In other words, the high-ranking interrupts specified by the interrupt priority selection register (IPS) are enabled when the status of the processing being executed is 0 or 1, and other interrupts (low-ranking interrupts) are enabled only when the status is 0 (Refer to Figure 6-9).

Therefore, if it is wished to set multiple interrupt capability for only one of the interrupts used, using this method makes it possible to accomplish dual interrupts without performing operations such as enabling/disabling during the interrupt servicing, and to hold down the number of nesting levels to two.

Figure 6-9 Multiple Interrupts by High-Rank Specification



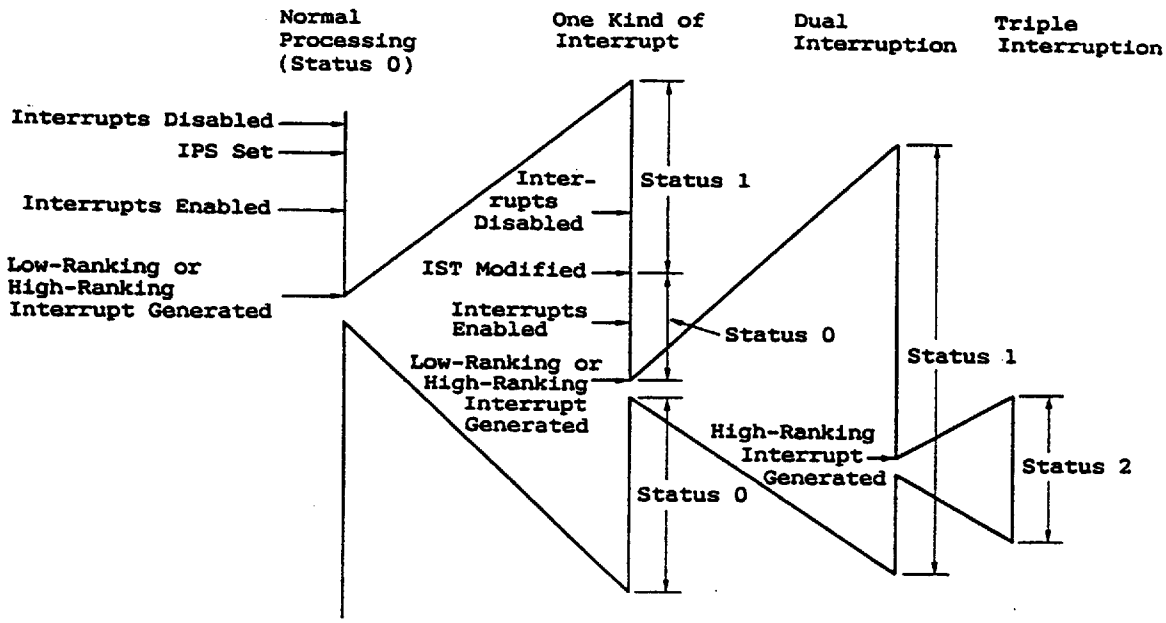
(2) Multiple interrupts through modification of interrupt status flag

If the interrupt status flag is modified by the program, multiple interrupts are enabled. That is, multiple interrupts are enabled by changing IST1 & IST0 to 0 & 0 and setting status 0 in the interrupt service program.

This method is used when it is wished to enable multiple interrupts involving two or more interrupts, or to perform triple-or-above multiple interruption.

IST1/IST0 modification is performed with interrupts disabled beforehand by means of the DI instruction.

Figure 6-10 Multiple Interrupts by Interrupt Status Flag Modification



6.6 VECTOR ADDRESS SHARING INTERRUPT SERVICING

Since INTBT and INT4 interrupt sources share a vector table, interrupt source selection is performed as shown below.

(1) When only one interrupt is used

The interrupt enable flag is set "1" for the necessary interrupt source of the two interrupt sources sharing the vector table, and the interrupt enable flag for the other source is cleared "0". In this case, when an interrupt request is generated by the enabled (IExxx = 1) interrupt source and acknowledged the corresponding interrupt request flag is reset (same operation as for interrupts not sharing a vector address).

(2) When both interrupts are used

The interrupt enable flags corresponding to the two interrupt sources are both set "1". In this case, the OR of the 2 interrupt source interrupt request flags constitutes the interrupt request.

In this case, even though an interrupt request is acknowledged by the setting of one or both of the interrupt request flags, neither interrupt request flag is reset.

In this case, therefore, it is necessary to determine within the interrupt service routine from which interrupt source the interrupt originated. This is done by checking the interrupt request flags with an SKTCLR instruction.

If both request flags are set when this interrupt request flag test-and-clear operation is performed, the interrupt request remains even if one of the request flags is cleared. When this interrupt is selected as a "high-ranking interrupt", dual interrupt servicing is entered by means of the remaining interrupt request.

In other words, the interrupt request for which testing is not performed is serviced first. Meanwhile, in the case of a "low-ranking interrupt", the remaining interrupt is held pending, and thus the interrupt request for which testing was performed is serviced first. Therefore, the method of shared interrupt discrimination differs according to whether or to "high-ranking interrupts" are involved.

Table 6-4 Shared Interrupt Discrimination

In case of "high-ranking interrupt"	Interrupts are disabled and interrupt request flag of interrupt to be given priority is tested.
In case of "low-ranking interrupt"	Interrupt request flag of interrupt source to be given priority is tested.

Example 1: When INTBT and INT4 are both used as "high-ranking interrupts", and INT4 is given priority.

```
        DI
        SKTCLR  IRQ4    ; IRQ4 = 1?
        BR      VSUBBT  ┐
           :            │
           :            ├ INT4 service
        EI              │ routine
        RETI            ┘
           :

VSUBBT: CLR1    IRQBT   ┐
           :            │
           :            ├ INTBT service
        EI              │ routine
        RETI            ┘
```

2: When INTBT and INT4 are both used as "low-ranking interrupts", and INT4 is given priority.

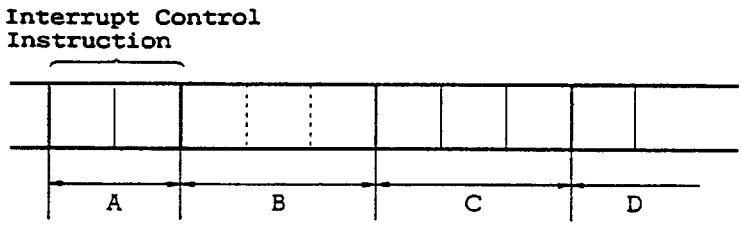
```
        SKTCLR  IRQ4    ; IRQ4 = 1?
        BR      VSUBBT  ┐
           :            │
           :            ├ INT4 service
           :            │ routine
        RETI            ┘
           :

VSUBBT: CLR1    IRQBT   ┐
           :            │
           :            ├ INTBT service
        RETI            │ routine
                        ┘
```

6.7 MACHINE CYCLES UNTIL INTERRUPT SERVICING

On the 75X, the machine cycles expended from setting of the interrupt request flag (IRQn) until the interrupt routine program is executed are as follows.

(1) When IRQn is set during interrupt control instruction execution

When IRQn is set during execution of an interrupt control instruction, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the next instruction.



A: Setting of IRQn
B: Execution of next instruction (between 1 and 3 machine cycles depending on instruction)
C: Interrupt servicing (3 machine cycles)
D: Execution of interrupt routine

Remarks 1: An interrupt control instruction is an instruction which manipulates interrupt-related hardware (data memory FBx address). These instructions comprise the DI and EI instructions.

Remarks 2: The 3 machine cycles of interrupt servicing include the time for manipulation of the stack on acknowledgment of an interrupt, etc.

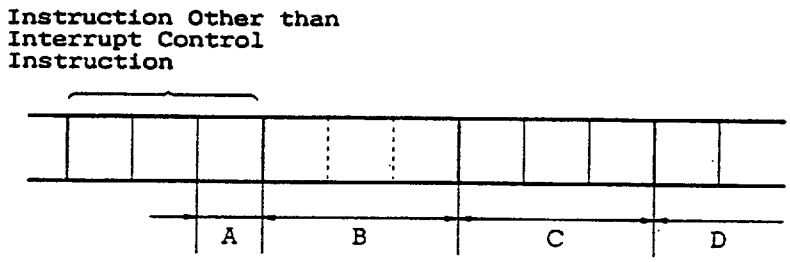
NOTE 1: If there are a number of consecutive interrupt control instructions, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the last interrupt control instruction executed.

2: If the DI instruction is executed using the IRQn setup timing (A in the above figure) or subsequent timing, the IRQn interrupt request is retained until the next EI instruction execution.

(2) When IRQn is set during execution of an instruction other than an interrupt control instruction

(a) When IRQn is set in the last machine cycle of the instruction being executed

In this case, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the instruction being executed.



A: Setting of IRQn

B: Execution of next instruction (between 1 and 3 machine cycles depending on instruction)

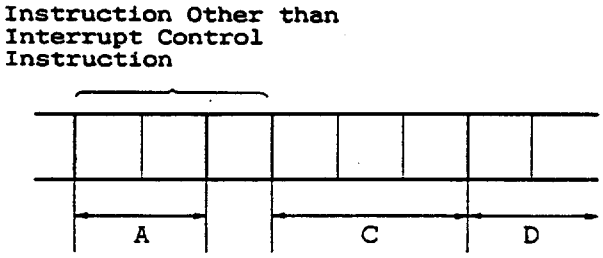
C: Interrupt servicing (3 machine cycles)

D: Execution of interrupt routine

NOTE: If the next instruction is an interrupt control instruction, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the last interrupt control instruction executed. Also, if the interrupt control instruction executed after IRQn is set is a DI instruction, the interrupt request by which IRQn was set is held pending.

(b) When IRQn is set before the last machine cycle of the instruction being executed

In this case, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following the instruction being executed.



A: Setting of IRQn

C: Interrupt servicing (3 machine cycles)

D: Execution of interrupt routine

6.8 EFFECTIVE USE OF INTERRUPTS

The following uses of the interrupt functions are effective.

(1) Setting MBE = 0 in the interrupt service routine

If the data memory to be used in the interrupt service routine is allocated in prioritized fashion to addresses 00H through 7FH and MBE = 0 is specified in the interrupt vector table, programming with memory bank transparency is possible.

If it is imperative to use memory bank 1 due to program considerations, memory bank 1 is selected by saving the memory bank selection register by means of a PUSH BS instruction.

(2) Split use of register banks in ordinary routines and interrupt routines

In ordinary routines, register banks 2 and 3 are used by setting RBE = 1 and RBS = 2. With a single interrupt service routine, using register bank 0 by setting RBE = 0 eliminates the need for register saving/restoration. With dual interrupt servicing, RBE = 1 is set, the register bank is saved by the PUSH BS instruction, and register bank 1 is used by setting RBS = 1.

(3) Use of software interrupts in debugging

Setting an interrupt request flag by means of an instruction results in the same operation as when an interrupt is generated. Irregular interrupt debugging or debugging when interrupts are generated simultaneously can be performed efficiently by setting the interrupt request flag by means of an instruction.

6.9 USE OF INTERRUPTS

When the interrupt function is used, the following settings are first carried out in the main program.

(a) The interrupt enable flag to be used is set (EI IExxx instruction).

(b) If INT0 and INT1 are used, the active edge is selected (IM0/IM1 setting).

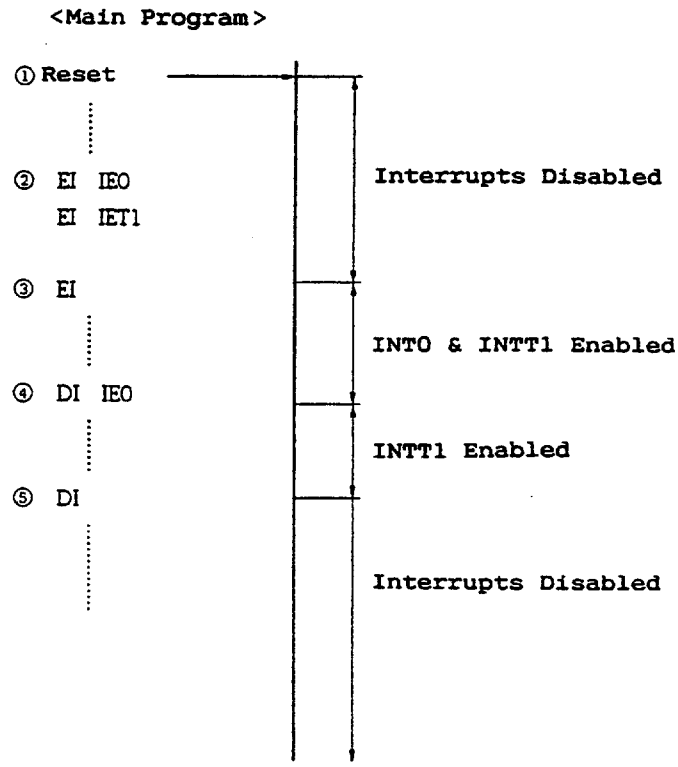
(c) When dual interruption is used (by means of high-ranking interrupts), IPS is set (IME can also be set at the same time).

(d) The interrupt master enable flag is set (EI instruction).

MBE and RBE are set by means of the vector table in the interrupt service program. However, register bank saving and setting is required for interrupts specified as "high-ranking interrupts".

Return from the interrupt service program is by means of an RETI instruction.

(1) Interrupt enabling/disabling



① All interrupts disabled by $\overline{\text{RESET}}$ signal.

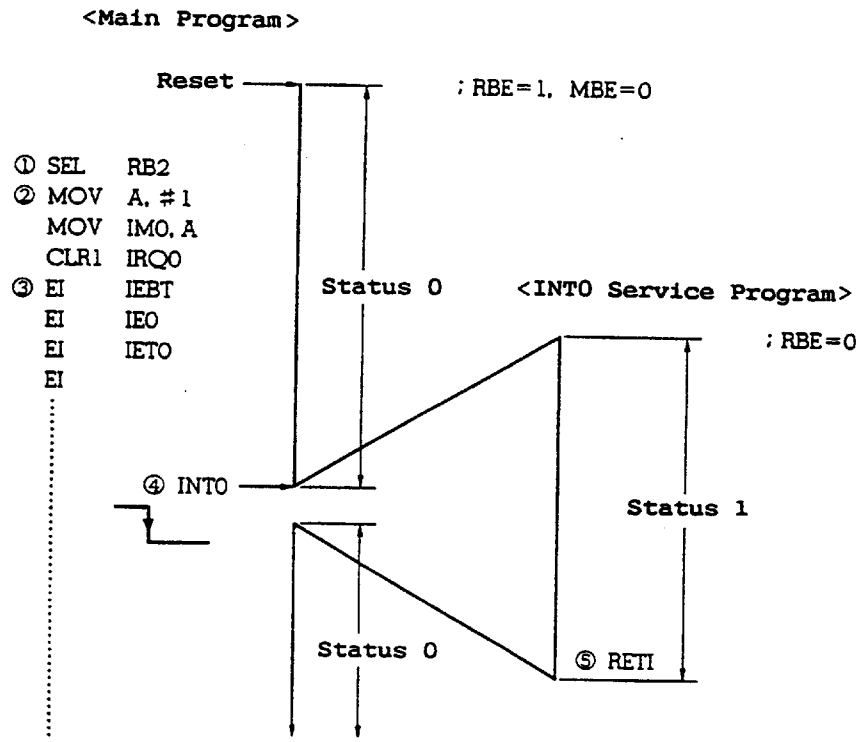
② Interrupt enable flag set by EI IExxx instruction.
At this stage, all interrupts are still disabled.

③ Interrupt master enable flag set by EI instruction.
At this stage, INT0 & INTT1 are enabled.

④ Interrupt enable flag cleared by DI IExxx instruction; INT0 disabled.

⑤ All interrupts disabled by DI instruction.

(2) Example using INTBT, INT0 (falling edge active), and INTT0, with no multiple interrupts (all low-ranking interrupts)



① All interrupts disabled and status 0 set by $\overline{\text{RESET}}$ signal.
RBE = 1 specified by reset vector table, register banks 2 & 3 used by means of SEL RB2 instruction.

② INT0 set to falling edge active.

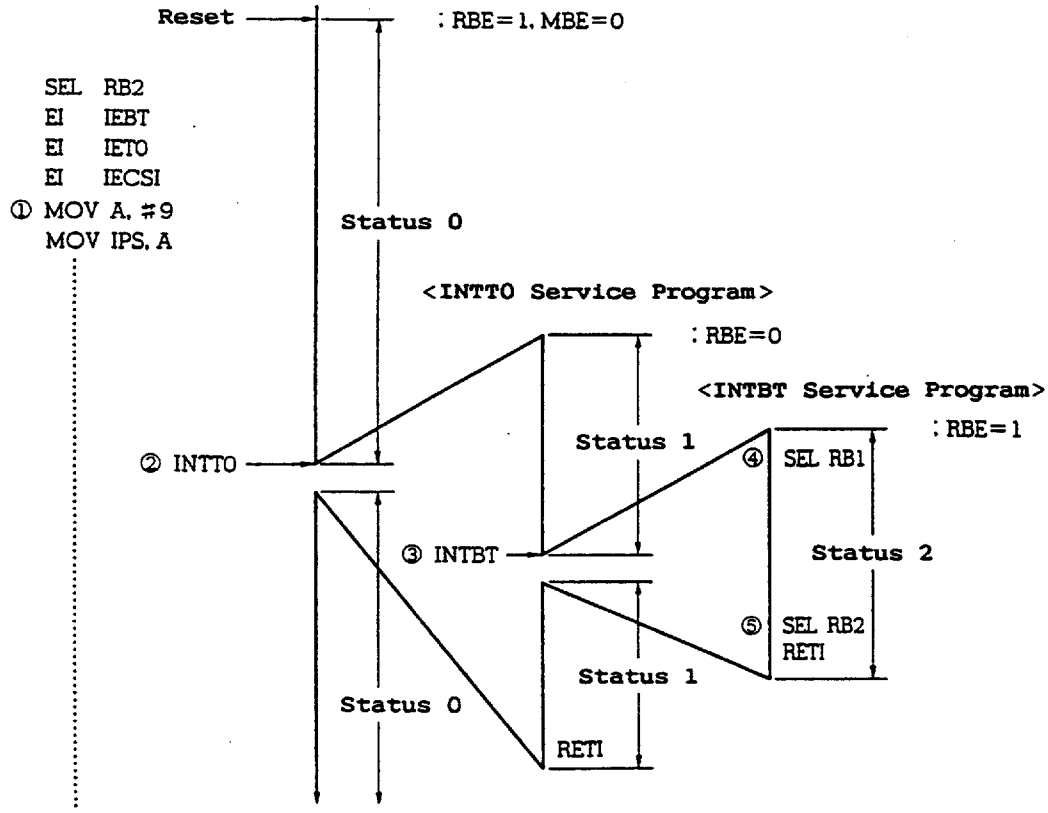
③ Interrupts enabled by EI and EI IExxx instructions.

④ On fall of INT0, INT0 interrupt service program is started. Status is changed to 1, and all interrupts are disabled.
RBE becomes 0, and register banks 0 & 1 are used.

⑤ Return from interrupt by means of RETI instruction.
Status is restored to 0 and interrupts are enabled.

Remarks: When all interrupts are used as "low-ranking interrupts" as in this example, register saving/restoration is completely unnecessary if register banks 2 and 3 are used by setting RBE = 1 and RBS = 2 in the main program, and register banks 0 and 1 are used by setting RBE = 0 in the interrupt service program.

(3) Multiple interrupt by means of "high-ranking interrupts" (INTBT high-ranking interrupts, INTTO & INTCSI low-ranking interrupts)



① INTBT set to "high-ranking interrupt" by IPS setting and interrupts enabled simultaneously.

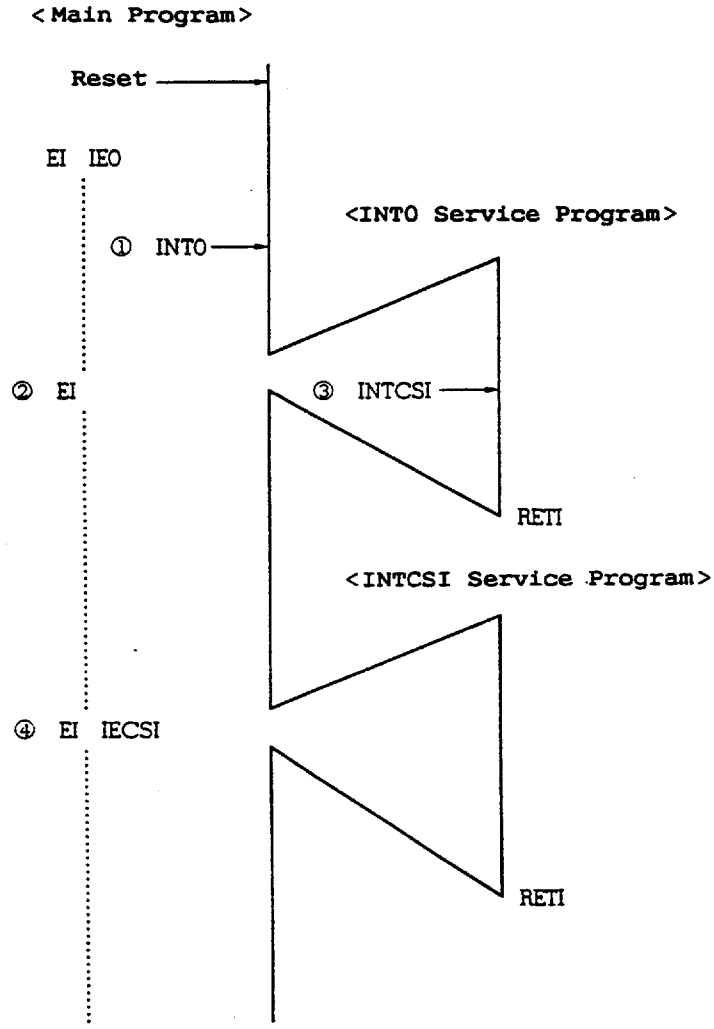
② Generation of low-ranking interrupt INTTO starts INTTO service program, sets status 1 and disables low-ranking interrupts. Register bank 0 is used by setting RBE = 0.

③ Generation of high-ranking interrupt INTBT results in dual interrupt servicing. Status 2 is set and all interrupts are disabled.

④ Register bank 1 is used by setting RBE = 1 and RBS = 1 (only the registers used need to be saved by PUSH instruction).

⑤ RBS is restored to 2 and return is performed. Status is restored to 1.

(4) Pending interrupt execution - interrupt input in interrupt disabled state -



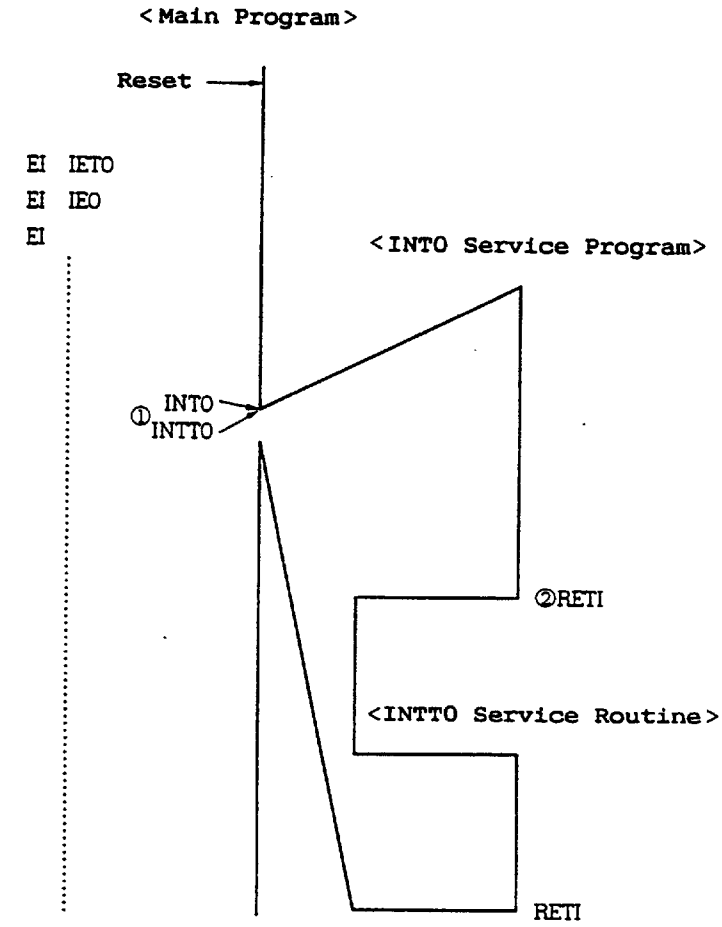
① Although INT0 is set in the interrupt disabled state, the request flag is held pending.

② The INT0 service program is started at the point at which interrupts are enabled by the EI instruction.

③ Same as ①

④ The INTCSI service program is started at the point at which the pending INTCSI is enabled.

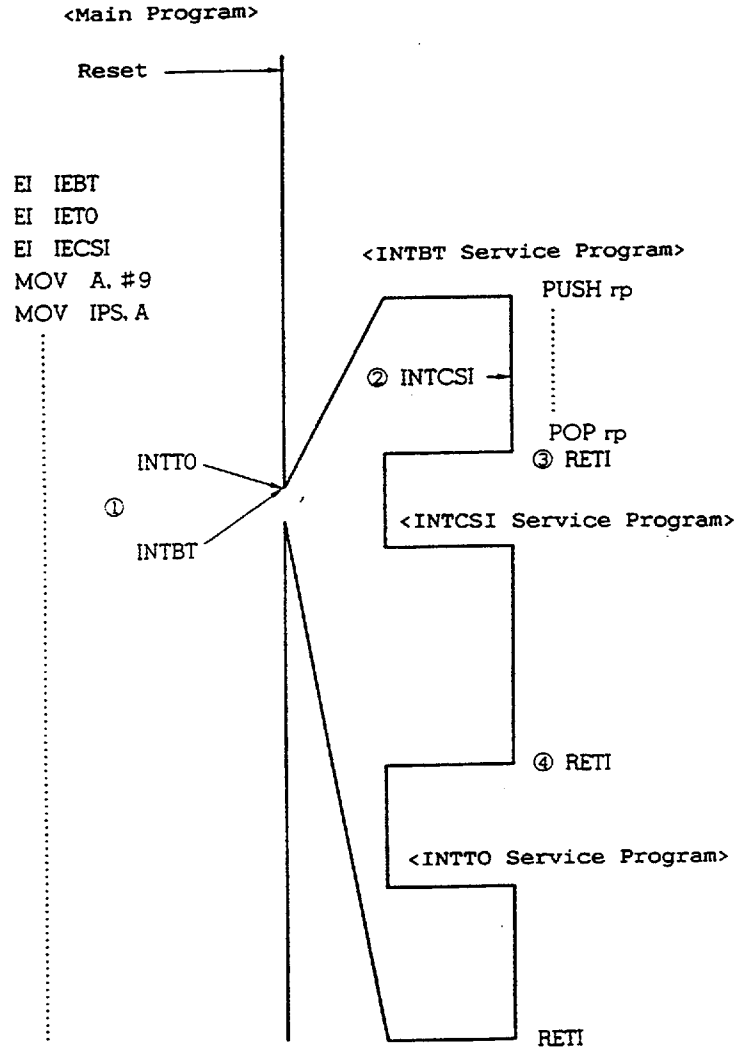
(5) Pending interrupt execution - simultaneous generation of two low-ranking interrupts -



① If "low-ranking interrupt" INTO and INTTO are generated simultaneously (during execution of the same instruction), INTO, which has the higher interrupt priority, is executed first (INTTO is held pending).

② When the INTO service program is ended by the RETI instruction, the pending INTTO service program is started.

(6) Pending interrupt execution - interrupt generation during interrupt servicing. (INTBT high-ranking interrupt, INTTO & INTCSI low-ranking interrupts)



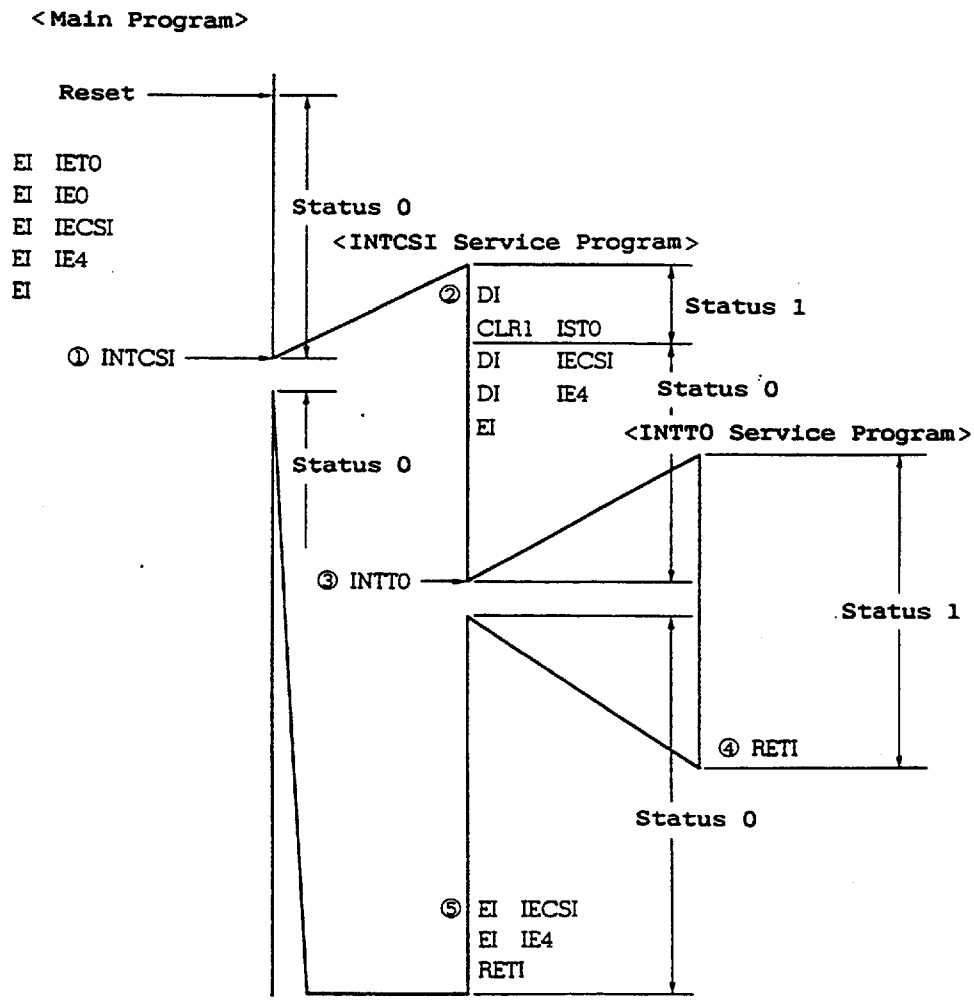
① If high-ranking interrupt INTBT and low-ranking interrupt INTTO are generated simultaneously, the high-ranking interrupt servicing is started. (If it is certain that no high-ranking interrupt will be generated during the high-ranking interrupt servicing, DI IExx is not necessary.)

② If a low-ranking interrupt is generated during execution of high-ranking interrupt, the interrupt is held pending.

③ When the high-ranking interrupt ends, servicing of INTCSI which has the highest ranking among the pending low-ranking interrupts is executed.

④ When INTCSI servicing ends, the pending INTTO is serviced.

(7) Two dual interrupts enabled. - Dual interrupt enabled for INTTO and INTO. INTCSI and INT4: Single interruption -



① Generation of an INTCSI interrupt for which dual interrupts are not enabled starts the INTCSI service program. Status is set to 1.

② Clearing ISTO sets the status to 0. INTCSI and INT4 for which dual interrupts are not enabled are disabled.

③ Generation of an INTTO interrupt for which dual interrupts are enabled results in dual interrupt execution, setting of status to 1, and disabling of all interrupts.

④ At the end of INTTO servicing, status is restored to 0.

⑤ The disabled INTCSI and INT4 are enabled and return is performed.

CHAPTER 7. STANDBY FUNCTIONS

The uPD75336 is provided with standby functions for reducing the system power consumption. The standby functions comprise the following two modes.

. STOP mode
. HALT mode

The functions of STOP mode and HALT mode are described below respectively.

(1) STOP mode

In this mode the main system clock oscillation circuit is stopped and the entire system stops. The CPU current consumption is considerably reduced.
In addition low-voltage (V_{DD} = up to 2 V) data memory retention is possible. This mode is thus effective for retaining the data memory contents at an extremely low current consumption.

As the uPD75336 STOP mode can be released by an interrupt request, intermittent operation is also possible. However, since a wait time is required for oscillation stabilization when the STOP mode is released, the HALT mode should be selected when processing must be started immediately by an interrupt request.

(2) HALT mode

In this mode the CPU operating clock is stopped, but oscillation by the system clock oscillation circuit continues. This mode does not allow current consumption to be reduced to the degree it is in the STOP mode, but it is effective when wishing to restart processing immediately by means of an interrupt request, or to perform intermittent operation such as clock operation.

In both modes, the contents of all registers, flags, and data memory directly prior to the setting of standby mode are retained. In addition, the status of input/output port output latches and output buffers is retained, and thus the input/output port statuses are processed beforehand so that the overall system current consumption is minimized.

Notes on use are given below.

NOTE 1: STOP mode can only be used when the system is operating on the main system clock (it is not possible to stop the oscillation of the subsystem clock). HALT mode can be used when operating on either the main system clock or the subsystem clock.

2: When the LCD controller/driver and clock timer operating clock is the main system clock f_X, setting STOP mode will stop their operation. Therefore, to continue their operation, it is necessary to switch the operating clock to the subsystem clock f_{XT} before setting STOP mode.

3: Standby mode and CPU clock/system clock switching allows efficient low current consumption and low-voltage operation, but the time specified in 5.2.3 "System Clock and CPU Clock Settings" is required in each case from selection of the new clock by control register manipulation until operation under the post-switchover clock begins. For this reason, when the standby mode is used in conjunction with the clock switchover function standby mode should be set after the time required for switchover has elapsed.

4: When the standby mode is used, the I/O port should be handled so that the current consumption is a minimum. In particular, the input port should not be left open. A low or high level must be input.

7.1 STANDBY MODE SETTING AND OPERATING STATUS

Table 7-1 Operating Status in Standby Mode

		STOP Mode	HALT Mode
Setting instruction		STOP instruction	HALT instruction
System clock at setting time		Main system clock	Main system clock Subsystem clock
Operating status	Clock generator	Main system clock oscillation only stopped	CPU clock Φ only stopped (oscillation continues)
	Basic interval timer	Operation stopped	Operation enabled only when main system clock oscillates. (IRQBT is set at basic time interval.)
	Serial interface	Operation enabled when external input $\overline{SCK}$ is selected as serial clock	Operation enabled only when external $\overline{SCK}$ input is selected as serial clock or only when main system clock oscillates.
	Timer/event counter	Operation enabled only when TIO pin input is specified as count clock	Operation enabled when TIn pin input is specified as count clock or only when main system clock oscillates.
	Watch timer	Operation enabled when f_{XT} is selected as count clock	Operation enabled
	LCD controller	Operation enabled when f_{XT} is selected as LCDCL	Operation enabled
	A/D converter	Operation stopped	Operation enabled*
	External interrupts	Operation enabled for INT1, 2 and 4 Operation disabled for INT0 only	
	CPU	Operation stopped	
Release signal		Interrupt request signal from hardware for which operation is enabled by interrupt enable flag, or $\overline{RESET}$ input	Interrupt request signal from hardware for which operation is enabled by interrupt enable flag, or $\overline{RESET}$ input

*: Only in case of main system clock oscillation

The STOP and HALT modes are set by the STOP and HALT instructions respectively. (The STOP and HALT instructions set PCC bit 3 and bit 2 respectively).

Ensure that an NOP instruction is always written directly after a STOP instruction or HALT instruction.

When the CPU operating clock is changed by means of the low-order 2 bits of the PCC, after the PCC has been overwritten as shown in Table 4-1 there may be a time lag before the CPU clock is changed. Therefore when changing the pre-standby-mode CPU clock and the post-standby-mode-release CPU clock, standby mode is set when the number of machine cycles required for the CPU clock change have elapsed after the PCC is rewritten.

In standby mode, all the data in registers and data memory (general registers, flags, mode registers, output latches, etc.) for which operation is stopped while in that mode is saved.

Notes are given below.

NOTE 1: When STOP mode is set, the X1 input is internally shorted to V_{SS} (GND potential) to suppress leakage of the crystal oscillator section. Therefore, use of STOP mode is not possible on systems which employ an external clock.

NOTE 2: Before setting the STANDBY mode, set the interrupt request flag in advance. If there is an interrupt source for which both the interrupt request flag and interrupt enable flag are set, the STANDBY mode, even if entered, is immediately released (see Figure 6-1 "Interrupt Control Circuit Block Diagram"). However, in case of the STOP mode, the HALT mode is entered immediately after execution of the STOP instruction, a wait follows for the duration of time set by the BTM register, and then the operating mode is restored.

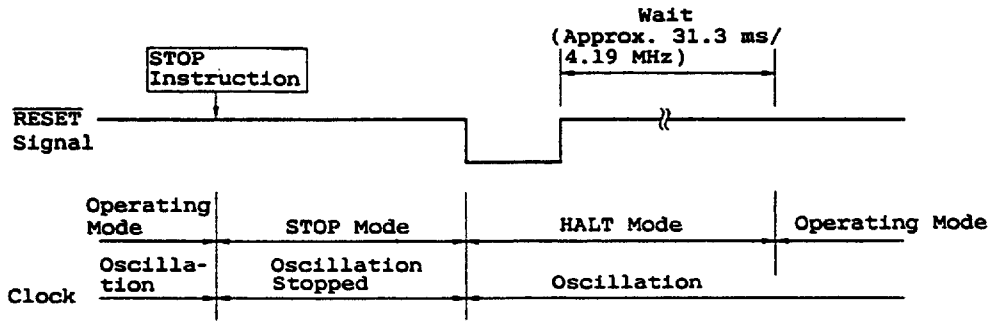
7.2 STANDBY MODE RELEASE

STOP mode and HALT mode are both released by generation of an interrupt request signal* enabled by an interrupt enable flag, and by $\overline{\text{RESET}}$ input. The release operation in each mode is shown in Figure 7-1.

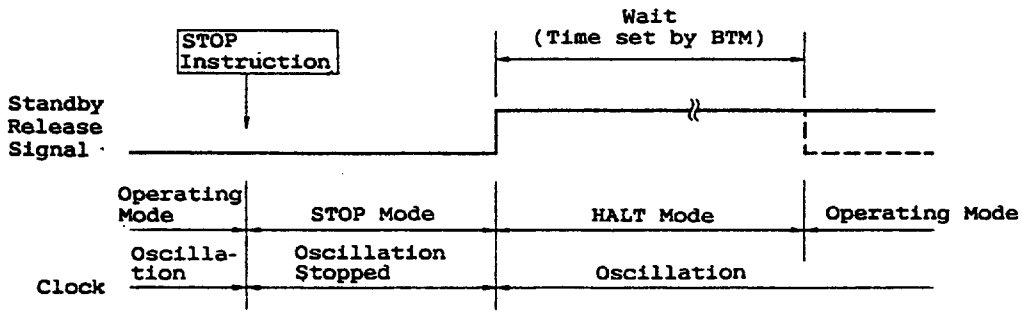
*: Except INT0

Figure 7-1 Standby Mode Release Operations

(a) STOP mode release by $\overline{\text{RESET}}$ input



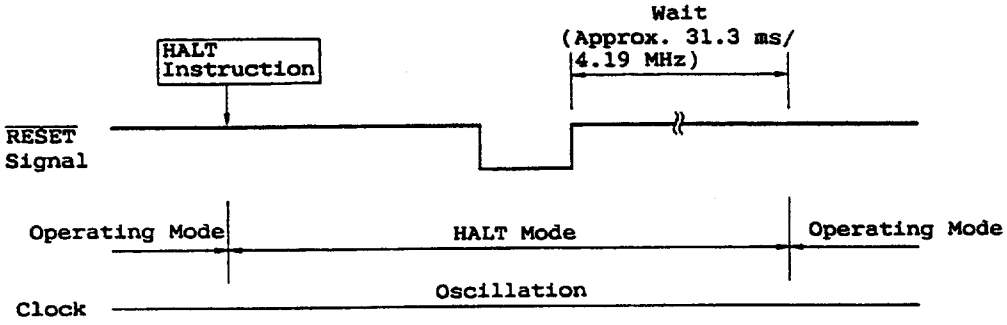
(b) STOP mode release by interrupt generation



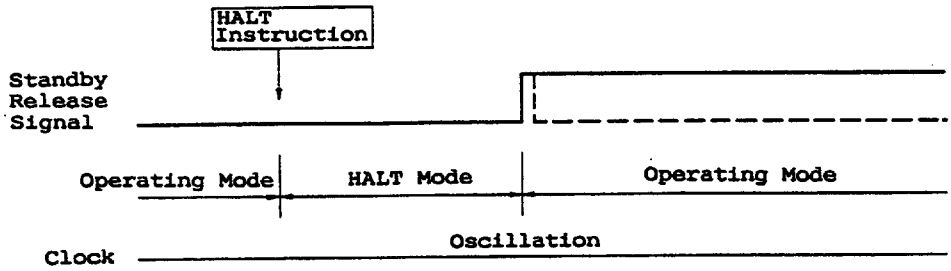
Remarks: The dotted line shows the case where the interrupt request which released standby mode is acknowledged (IME = 1).

Figure 7-1 Standby Mode Release Operations (cont'd)

(c) HALT mode release by $\overline{\text{RESET}}$ input



(d) HALT mode release by interrupt generation



Remarks: The dotted line shows the case where the interrupt request which released standby mode is acknowledged (IME = 1).

When STOP mode is released by program of an interrupt, the wait time is determined by the setting of BTM (see Table 7-2).

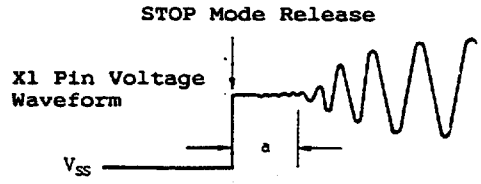
The time until oscillation stabilizes varies according to the type of resonator used and the power supply voltage when STOP mode is released. Therefore, the wait time is selected according to the conditions of use and BTM is set before setting STOP mode.

Table 7-2 Wait Time Selection by BTM

BTM3	BTM2	BTM1	BTM0	Wait Time* Figure in ( ) is for f_X = 4.19 MHz
-	0	0	0	Approx. $2^{20}/f_X$ (Approx. 250 ms)
-	0	1	1	Approx. $2^{17}/f_X$ (Approx. 31.3 ms)
-	1	0	1	Approx. $2^{15}/f_X$ (Approx. 7.82 ms)
-	1	1	1	Approx. $2^{13}/f_X$ (Approx. 1.95 ms)
Other than above				Setting prohibited

*: This time does not include the time until the start of oscillation after STOP mode release.

NOTE: In both the $\overline{\text{RESET}}$ input and the interrupt generation case, the wait time when STOP mode is released does not include the time until clock oscillation begins after STOP mode release ("a" in the diagram below).



7.3 OPERATION AFTER STANDBY MODE RELEASE

(1) When standby mode is released by $\overline{\text{RESET}}$ input, a normal reset operation is executed.

(2) When standby mode is released by generation of an interrupt request, whether or not a vectored interrupt is performed when the CPU restarts instruction execution is determined by the contents of the interrupt master enable flag (IME).

(a) When IME = 0

After standby mode is released, execution is restarted from the instruction (NOP instruction) following that in which standby mode was set.

The interrupt request flag value is retained.

(b) When IME = 1

After standby mode is released, a vectored interrupt is executed after the execution of the two instructions following that in which standby mode was set. However, a vectored interrupt is not generated when standby mode is released by means of INTW or INT2 (testable inputs), and in this case the same processing is performed as in (a) above.

7.4 USE OF STANDBY MODE

The procedure shown below is followed when using standby mode.

① Detection of the standby mode setting source, such as power cutoff by an interrupt input or port input. (Use of INT4 for power cutoff detection is effective.)

② Input/output port processing (processing to minimize current consumption).
In particular, the input port should not be left open. A low or high level should be input.

③ Setting of interrupt to release standby mode. (Use of INT4 is effective; non-releasing interrupt enable flags are cleared.)

④ Post-release operation setting (IME manipulation according to whether or not interrupt servicing is performed).

⑤ Post-release CPU clock setting. (When switching, the necessary number of machine cycles are allowed to elapse before standby mode is set.)

⑥ Selection of wait time used when release is performed.

⑦ Standby mode setting (STOP/HALT mode).

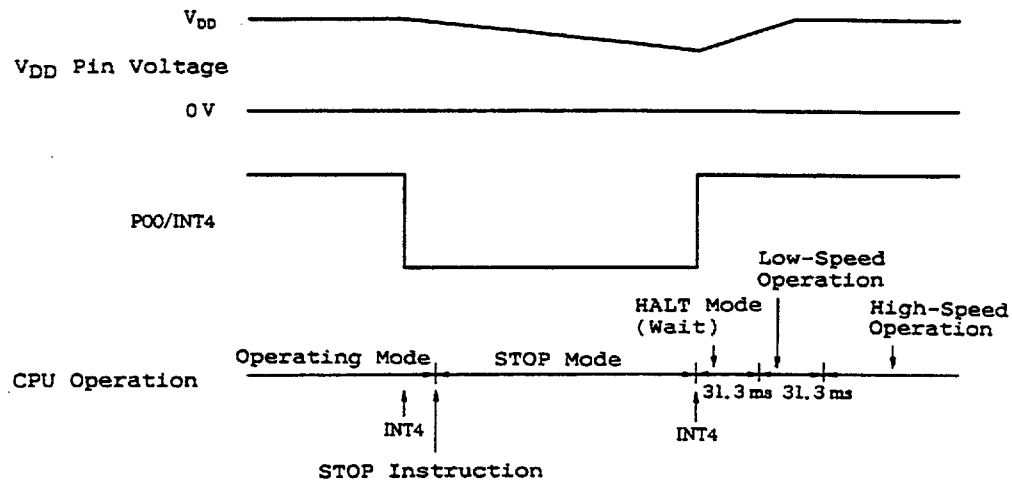
When used in conjunction with system clock switching, standby mode allows low-power-consumption and low-voltage operation.

(1) Example of use of STOP mode (f_X = 4.19 MHz operation)

<When STOP mode is used under the following conditions>

- STOP mode is set by an INT4 falling edge input and released by a rising edge input (INTBT is not used).
- All input ports are high impedance. (when a pin is processed externally to reduce current consumption at high impedance)
- Interrupts used in the program are INT0 and INTT0; however, these are not used for releasing STOP mode.
- Interrupts are also enabled after release.
- After release, operation starts under the slowest CPU clock.
- The wait time used when releasing is set to 31.3 ms.
- After release, a further 31.3 ms wait is performed to allow the power supply to stabilize. The P00/INT4 pin is checked twice to eliminate chattering.

<Timing chart>



<Sample program>

(INT4 service program, MBE = 0)

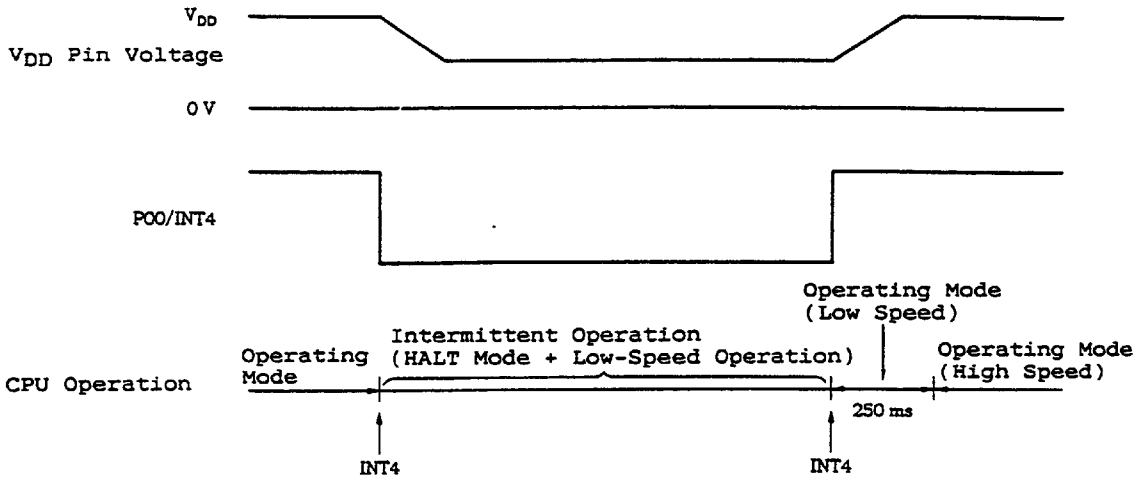
```
VSUB4:  SKT    PORT0.0  ; P00 = 1?
        BR     PDOWN    ; power-down
        SET1   BTM.3    ; power-on
WAIT :  SKT    IRQBT    ; 31.3 ms wait
        BR     WAIT
        SKT    PORT0.0  ; check chattering
        BR     PDOWN
        MOV    A, #0011B
        MOV    PCC, A   ; set high-speed mode
       [MOV    XA, #xxH]; set port mode register
       [MOV    PMGm, XA]
        EI     IE0
        EI     IET0
        RETI
PDOWN:  MOV    A, #0    ; low-speed mode
        MOV    PCC, A
        MOV    XA, #00H
        MOV    LCDM, XA ; LCD display off
        MOV    LCDC, A
        MOV    PMGA, XA ; input/output ports high
                          impedance
        MOV    PMGB, XA
        DI     IE0      ; INT0/INTT0 disabled
        DI     IET0
        MOV    A, #1011B
        MOV    BTM, A   ; wait time = 31.3 ms
        NOP
        STOP            ; set STOP mode
        NOP
        RETI
```

(2) Use of HALT mode (f_X = 4.19 MHz, f_{XT} = 32.768 kHz operation)

<In case of intermittent operation under the following conditions>

- Subsystem clock is switched to on fall of INT4.
- Main system clock oscillation is stopped and HALT mode is set.
- Intermittent operation performed in standby mode at 0.5 sec. intervals.
- System switches back to main system clock on rise of INT4.
- INTBT is not used.

<Timing chart>



<Sample program>

(Initialization)

```
        MOV     A, #0011B
        MOV     PCC, A      ; high-speed mode
        MOV     XA, #05
        MOV     WM, XA      ; subsystem clock
        EI      IE4
        EI      IEW
        EI                  ; interrupts enabled
```

(Main routine)

```
        SKT     PORT0.0     ; power supply OK?
        HALT                ; power-down mode
        NOP                 ; power supply OK?
        SKTCLR  IRQW        ; 0.5 flag set?
        BR      MAIN        ; NO
        CALL    WATCH       ; clock subroutine
MAIN :    :
          :
          :
```

(INT4 service routine)

```
VINT4:  SKT     PORT0.0     ; power supply OK?, MBE = 0
        BR      PDOWN
        CLR1    SCC.3       ; start main system clock
                              oscillation
        MOV     A, #1000B
        MOV     BTM, A
WAIT1:  SKT     IRQBT       ; 250 ms wait
        BR      WAIT1
        SKT     PORT0.0     ; check chattering
        BR      PDOWN
        CLR1    SCC.0       ; switch to main system clock
        RETI
```

```
PDOWN:  MOV     XA, #00H
        MOV     LCDM, XA ; LCD display off
        MOV     LCDC, A
        SET1    SCC.0    ; switch to subsystem clock
        MOV     A, #5
WAIT2:  INCS    A        ; wait of 32 machine cycles
                           or more (35 machine
                           cycles)*
        BR      WAIT2
        SET1    SCC.3    ; stop main system clock
                           oscillation
        RETI
```

*: See 5.2.3 "System Clock and CPU Clock Settings" for details of switching between the system clock and CPU clock.

NOTE: When switching from the main system clock to the subsystem clock, it is necessary to wait for the subsystem clock oscillation to stabilize before making the switchover.

CHAPTER 8. RESET FUNCTION

The uPD75336 is reset by $\overline{\text{RESET}}$ input and the various hardware units are initialized as shown in Table 8-1 to 8-3. Reset operation timing is shown in Figure 8-1.

Figure 8-1 $\overline{\text{RESET}}$ Input Reset Operation

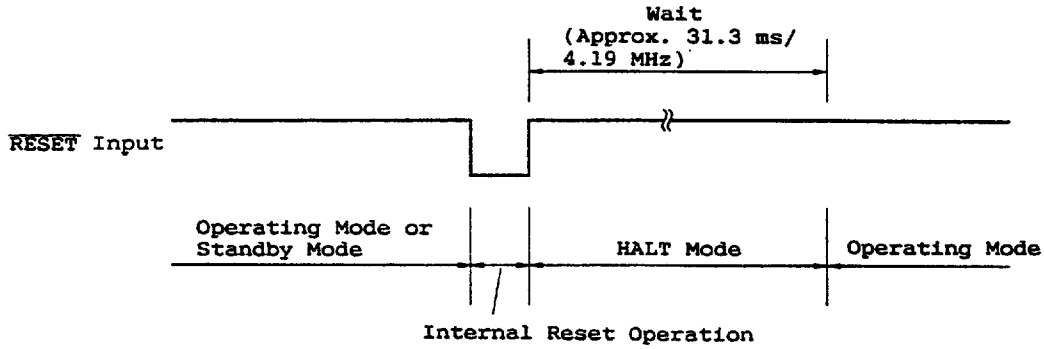


Table 8-1 Hardware Status after Reset

Hardware		$\overline{\text{RESET}}$ Input in Standby Mode	$\overline{\text{RESET}}$ Input during Operation
Program counter (PC)		Program memory address 0000H low-order 6 bits set in PC13 to PC8, address 0001H contents in PC7 to PC0.	Same as at left
PSW	Carry flag (CY)	Retained	Undefined
	Skip flags (SK0 to SK2)	0	0
	Interrupt status flag (IST0)	0	0
	Bank enable flags (MBE, RBE)	Program memory address 0000H bit 6 set in RBE, bit 7 in MBE.	Same as at left
Stack pointer (SP)		Undefined	Undefined

(to be continued)

Table 8-1 Hardware Status after Reset (cont'd)

Hardware		$\overline{\text{RESET}}$ Input in Standby Mode	$\overline{\text{RESET}}$ Input during Operation
Data memory (RAM)		Retained *1	Undefined
General registers (X,A, H, L, D, E, B, C)		Retained	Undefined
Bank selection registers (MBS, RBS)		0, 0	0, 0
Basic interval timer	Counter (BT)	Undefined	Undefined
	Mode register (BTM)	0	0
Timer/ event counter (n = 0, 1)	Counter (Tn)	0	0
	Modulo register (TMODn)	FFH	FFH
	Mode register (TMn)	0	0
	TOEn, TOUT F/F	0, 0	0, 0
Clock timer	Mode register (WM)	0	0
Serial interface	Shift register (SIO)	Retained	Undefined
	Operating mode register (CSIM)	0	0
	SBI control register (SBIC)	0	0
	Slave address register (SVA)	Retained	Undefined
Clock generation circuit, clock output circuit	Processor clock control register (PCC)	0	0
	System clock control register (SCC)	0	0
	Clock output mode register (CLOM)	0	0

(to be continued)

Table 8-1 Hardware Status after Reset (cont'd)

Hardware			$\overline{\text{RESET}}$ Input in Standby Mode	$\overline{\text{RESET}}$ Input during Operation
LCD controller	Display mode register (LCDM)		0	0
	Display control register (LCDC)		0	0
A/D converter	Mode register (ADM)		04H (EOC = 1)	04H (EOC = 1)
	SA register		7FH	7FH
Interrupt functions	Interrupt request flags (IRQxxx)	IRQ1, IRQ2 IRQ4	Undefined	Undefined
		Other than above	0	0
	Interrupt enable flags (IExxx)		0	0
	Interrupt priority selection register (IPS)		0	0
	INT0, INT1 and INT2 mode registers (IM0, IM1 and IM2)		0, 0, 0	0, 0, 0
Digital ports	Output buffer		Off	Off
	Output latch		Cleared (0)	Cleared (0)
	Input/output mode registers (PMGA, PMGB, PMGC)		0	0
	Pull-up resistor specification registers (POGA, POGB)		0	0
Bit sequential buffers (BSB0 to BSB3)			Retained	Undefined

(to be continued)

Table 8-1 Hardware Status after Reset (cont'd)

Hardware		$\overline{\text{RESET}}$ Input in Standby Mode	$\overline{\text{RESET}}$ Input during Operation
Pin status	P00 to P03, P10 to P13, P20 to P23, P30 to P33, P60 to P63, P70 to P73, P80 to P83	Input	Same as at left
	P40 to P43, P50 to P53	. When using internal pull-up resistor ... High level . When open-drain ... High-impedance	Same as at left
	S12 to S32, COM0 to COM3	*2	Same as at left
	BIAS	. With internal split resistor ... Low level . With no internal split resistor ... High-impedance	Same as at left

*1: Data in data memory addresses 0F8H to 0FDH is undefined after $\overline{\text{RESET}}$ input.

2: Display outputs are selected using the V_{LCX} shown below as the input source.

S12 to S31 : V_{LC1}
COM0 to COM2: V_{LC2}
COM3 : V_{LC0}

The display output level varies depending on the display output and V_{LCX} external circuit.

CHAPTER 9. PROM (PROGRAM MEMORY) WRITE AND VERIFY OPERATIONS

The program memory incorporated in the uPD75P336 is one-time PROM. The memory capacity is as follows:

uPD75P336: 16256 words x 8 bits

Write/verify operations on this one-time PROM are executed using the pins shown in the table below. Address updating is performed by means of clock input from the X1 pin, and not by address input.

Table 9-1 Pin for Using Program Memory Write/Verify Operating Modes

Pin Name	Function
X1, X2	Input of address update clock for program memory write/verify. Inverse of X1 pin signal is input to X2 pin.
MD0 to MD3 (P30 to P33)	Operating mode selection pins for program memory write/verify.
P40 to P43 (low-order 4 bits), P50 to P53 (high-order 4 bits)	8-bit data input/output pins for program memory write/verify.
V_{DD}	Supply voltage application pin. Applies 2.7 to 6.0 V in normal operation, and +6 V for program memory write/verify.
V_{PP}	Voltage application pin for program memory write/verify (normally V_{DD} potential).

NOTE 1: Since the uPD75P336 is not provided with an erase window, program memory contents cannot be erased.

NOTE 2: Pins not used in a program memory write/verify operation are handled as follows:

- All pins except XT2 ... Connect to V_{SS} with a pull-down resistor.
- XT2 ..... Leave open.

9.1 PROGRAM MEMORY WRITE/VERIFY OPERATING MODES

When +6 V is applied to the V_{DD} pin and +12.5 V to the V_{PP} pin, the uPD75P336 is placed in the program memory write/verify mode. This mode comprises one of the operating modes shown below according to the input signal to pins MD0 to MD3.

Table 9-2 Operating Modes

Operating Mode Setting						Operating Mode
V_{DD}	V_{PP}	MD0	MD1	MD2	MD3	
+6 V	+12.5 V	H	L	H	L	Program memory address zero-clear
		L	H	H	H	Write mode
		L	L	H	H	Verify mode
		H	x	H	H	Program inhibit mode

Remarks: x: L or H

9.2 PROGRAM MEMORY WRITE PROCEDURE

The procedure for writing to program memory is as shown below, allowing high-speed writing.

(1) Unused pins are connected to V_{SS} with a pull-down resistor. The X1 pin is driven low.

(2) 5 V is supplied to the V_{DD} and V_{PP} pins.

(3) 10 us wait.

(4) Program memory address zero-clear mode.

(5) +6 V is supplied to V_{DD}, +12.5 V to V_{PP}.

(6) Program inhibit mode.

(7) Data is written in 1 ms write mode.

(8) Program inhibit mode.

(9) Verify mode. If write is successful go to (10), otherwise repeat (7) through (9).

(10) (Number of times written in (7) through (9): X) x 1 ms additional writes.

(11) Program inhibit mode

(12) Program memory address is updated (+1) by inputting 4 pulses to the X1 pin.

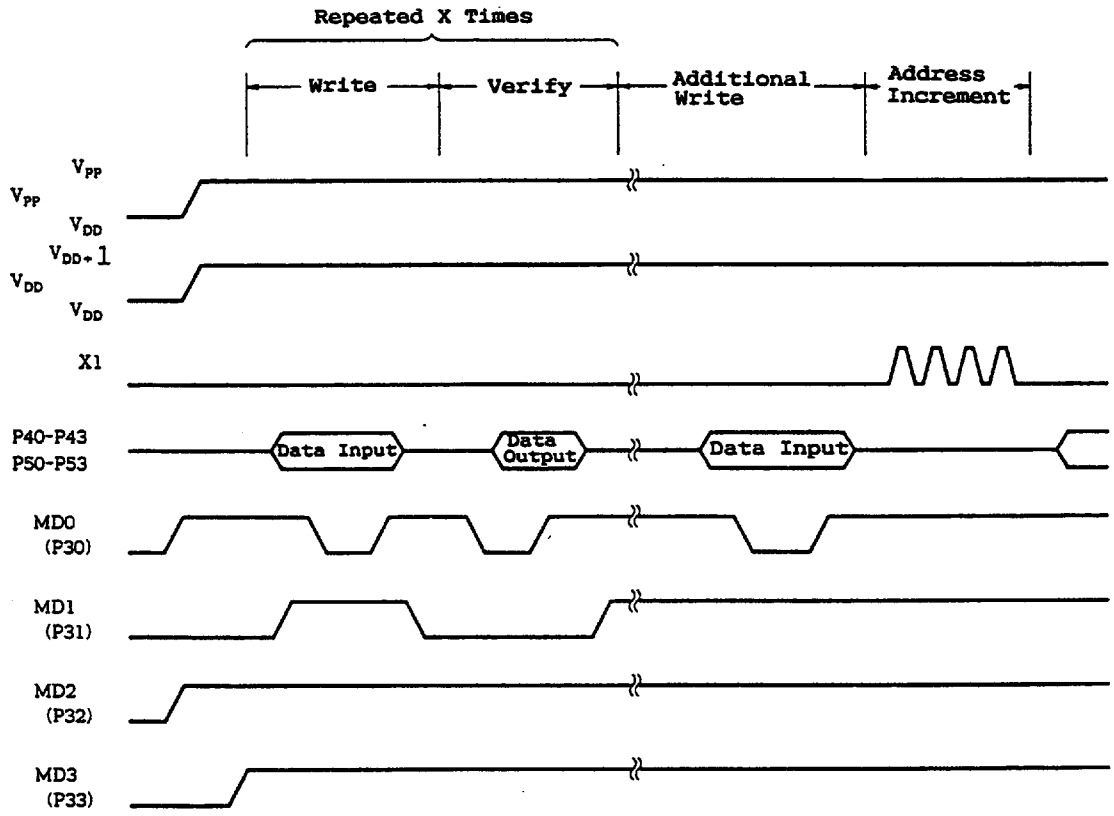
(13) Steps (7) through (12) are repeated until the last address.

(14) Program memory address zero-clear mode.

(15) V_{DD}/V_{PP} pin voltage is changed to 5 V.

(16) Power-off.

Steps (2) to (12) of this procedure are shown in the following figure.



9.3 PROGRAM MEMORY READ PROCEDURE

uPD75P336 program memory contents can be read using the following procedure. Reading is performed in verify mode.

(1) Unused pins are connected to V_{SS} with a pull-down resistor. The X1 pin is driven low.

(2) 5 V is supplied to the V_{DD} and V_{PP} pins.

(3) 10 us wait.

(4) Program memory address zero-clear mode.

(5) +6 V supplied to V_{DD}, +12.5 V to V_{PP}.

(6) Program inhibit mode.

(7) Verify mode. When clock pulses are input to the X1 pin, data is output sequentially, one address per 4-pulse-input cycle.

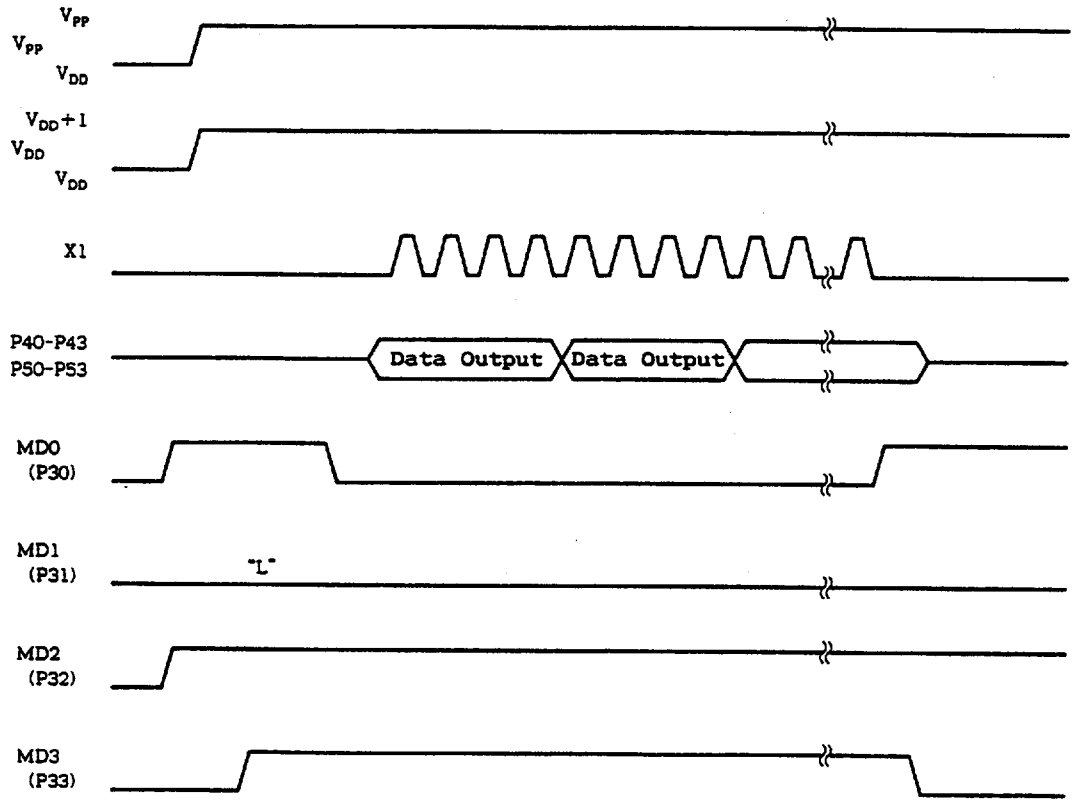
(8) Program inhibit mode.

(9) Program memory address zero-clear mode.

(10) V_{DD}/V_{PP} pin voltage is changed to 5 V.

(11) Power-off.

Steps (2) to (9) of this procedure are shown in the following figure.



CHAPTER 10. INSTRUCTION SET

The uPD75336 instruction set is an improved and extended version of that of the uPD7500 series, the predecessor of the 75X series. Thus, while inheriting the features of the uPD7500 series it constitutes a new epoch-making instruction set, with the following special characteristics.

(1) Bit-manipulating instructions with a wide range of uses

(2) Efficient 4-bit manipulating instructions

(3) 8-bit data transfer instructions

(4) GETI instruction for reduced program size

(5) Stacked instructions and radix adjustment instructions for improved program efficiency

(6) Table referencing instructions suitable for consecutive references

(7) One-byte relative branch instructions

(8) Easy-to-understand rearranged standard NEC mnemonics

In addition, see Chapter 3 "Data Memory Operation and Memory Map" for the addressing modes which can be used in data memory operations.

10.1 SPECIAL INSTRUCTIONS

An outline is given here of the special instructions in the uPD75336's instruction set.

10.1.1 GETI INSTRUCTION

The GETI instruction is used to convert the following instructions to 1-byte instructions.

(a) Total-space subroutine call instructions

(b) Branch instructions to the total space

(c) Any 2-byte 2-machine-cycle instruction (excluding BRCB and CALLF instructions)

(d) A combination of two 1-byte instructions

The GETI instruction references a table in program memory addresses 0020H to 007FH, and executes the referenced 2-byte data as one of the instructions in (a) through (d) above. Therefore the 48 instructions in (a) through (d) can be converted to 1-byte instructions.

Using the GETI instruction to convert frequently used instructions (a) through (d) to 1-byte instructions allows the number of program bytes to be considerably reduced.

10.1.2 BIT MANIPULATION INSTRUCTIONS

uPD75336 bit manipulations can be performed by the following wide variety of instructions.

(a)	Bit setting	:	SET1	mem. bit
			SET1	mem. bit*
(b)	Bit clearing	:	CLR1	mem. bit
			CLR1	mem. bit*
(c)	Bit testing	:	SKT	mem. bit
			SKT	mem. bit*
(d)	Bit testing	:	SKF	mem. bit
			SKF	mem. bit*
(e)	Bit testing and clearing	:	SKTCLR	mem. bit*
(f)	Boolean operations	:	AND1	CY, mem. bit*
			OR1	CY, mem. bit*
			XOR1	CY, mem. bit*
(g)	Bit transfer	:	MOV1	CY, mem. bit*
			MOV1	mem. bit*, CY

mem. bit* is the bit address indicated by bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).
In particular, since all the above bit manipulation instructions can always be used on input/output ports, input/output operations can be performed extremely efficiently.

10.1.3 STACKED INSTRUCTIONS

The uPD75336 is provided with the following two kinds of stacked instruction.

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"Stacked" means that these two kinds of instruction are located in consecutive addresses.

```
Example:  A0 :  MOV A, #0
          A1 :  MOV A, #1
          XA7:  MOV XA, #07
```

If stacked instructions are arranged in sequence as in this example, if the first address to be executed is A0, the following two instructions are replaced by a NOP instruction before being executed; if the first address to be executed is A1, then the following instruction only is replaced by NOP. In other words, only the first instruction executed is effective and the following instructions are all processed as NOP instructions.

Use of these stacked instructions allows the setting of constants in an accumulator (A register or XA register pair) and setting of constants in the data pointer (HL register pair) to be performed efficiently.

10.1.4 RADIX ADJUSTMENT INSTRUCTIONS

Depending on the application, the result of a 4-bit data addition or subtraction (performed in binary) must be converted to decimal or subjected to radix-6 adjustment, as with a time.

For this purpose the uPD75336 instruction set includes radix adjustment instructions for adjusting the result of a 4-bit data addition or subtraction to a number of any desired radix.

(a) Radix adjustment in addition

If the radix for adjustment is m, the following combination adds memory (HL) to the accumulator and performs m-ary adjustment of the result.

```
ADDS A, #16 - m
ADDC A, @HL     ; A, CY ← A + (HL) + CY
ADDS A, #m
```

Overflow is left in the carry flag.

If a carry is generated as the result of executing the ADDC A, @HL instruction, the following ADDS A, #n4 instruction is skipped. If no carry is generated the ADDS A, #n4 instruction is executed, but the instruction skipping function at this time is disabled, and the next instruction is not skipped even if a carry is generated as a result of the addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

Example: Decimal addition of accumulator and memory.

```
ADDS A, #6
ADDC A, @HL; A, CY ← A + (HL) + CY
ADDS A, #10
     :
     :
```

(b) Radix adjustment in subtraction

If the radix for adjustment is m, the following combination subtracts memory (HL) from the accumulator and performs m-ary adjustment of the result.

```
SUBC A, @HL
ADDS A, #m
```

Underflow is left in the carry flag.

If no borrow is generated as the result of executing the SUBC A, @HL instruction, the following ADDS A, #n4 instruction is skipped. If a borrow is generated the ADDS A, #n4 instruction is executed, but the instruction skipping function at this time is disabled, the next instruction is not skipped even if a carry is generated as a result of the addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

10.1.5 SKIP INSTRUCTIONS AND MACHINE CYCLES REQUIRED FOR SKIPPING

With the uPD75336 instruction set, a program is configured using conditional judgments by means of skipping.

"Skipping" is a function whereby if the skip condition is fulfilled when a skip instruction (instruction with a skip condition) is executed, the following instruction is jumped over (skipped) and the instruction after the skipped instruction is executed.

If a skip is generated, the number of machine cycles required for skipping is as shown below.

(a) When the instruction (the instruction to be skipped) following the skip instruction is a 3-byte instruction (either the BR !addr instruction or the CALL !addr instruction) : 2 machine cycles

(b) With instructions other than those in (a) : 1 machine cycle

10.2 INSTRUCTION SET AND OPERATIONS

(1) Operand notation and description method

Operands are described in the operand field of an instruction in accordance with the description method for the operand notation for that instruction (see the "RA75X Assembler Package User's Manual Language Volume (EEU-730)" for details). When there are several elements in the description method field, one is selected. Capital letters and symbols "+" and "-" are keywords and are written as they are.

With immediate data, the relevant number or label is written.

The symbols given in the various register and flag format diagrams in Chapter 3 to 6 can be written as labels instead of mem, fmem, pmem, bit, and so forth (but note that there are restrictions on labels that can be used for fmem and pmem).

Notation	Description Method
reg reg1	X, A, B, C, D, E, H, L X, B, C, D, E, H, L
rp rp1 rp2 rp' rp'1	XA, BC, DE, HL BC, DE, HL BC, DE XA, BC, DE, HL, XA', BC', DE', HL' BC, DE, HL, XA', BC', DE', HL'
rpa rpa1	HL, HL+, HL-, DE, DL DE, DL
n4 n8	4-bit immediate data or label 8-bit immediate data or label
mem bit	8-bit immediate data* or label 2-bit immediate data or label

(to be continued)

(cont'd)

Notation	Description Method
fmem. bit	IST1, IST0, MBE, RBE, IExxx, IRQxxx, PORTn.m (n = 0 to 8, m = 0 to 3)
pmem	PORTn (n = 0, 4)
addr	0000H to 3F7FH immediate data or label
caddr faddr taddr	12-bit immediate data or label 11-bit immediate data or label 20H to 7FH immediate data (bit0 = 0) or label
PORTn IExxx RBn MBn	PORT0 to PORT8 IEBT, IECSI, IET0, IET1, IE0 to IE2, IE4, IEW RB0 to RB3 MB0, MB1, MB2, MB15

*: For 8-bit data processing, only an even address can be specified.

(2) Legend for operation descriptions

A : A register; 4-bit accumulator
B : B register; 4-bit accumulator
C : C register; 4-bit accumulator
D : D register; 4-bit accumulator
E : E register; 4-bit accumulator
H : H register; 4-bit accumulator
L : L register; 4-bit accumulator
X : X register; 4-bit accumulator
XA : Register pair (XA); 8-bit accumulator
BC : Register pair (BC)
DE : Register pair (DE)
HL : Register pair (HL)
XA' : Extended register pair (XA')
BC' : Extended register pair (BC')
DE' : Extended register pair (DE')
HL' : Extended register pair (HL')
PC : Program counter
SP : Stack pointer

CY : Carry flag; bit accumulator
PSW : Program status word
MBE : Memory bank enable flag
RBE : Register bank enable flag
PORTn: Port n (n = 0 to 8)
IME : Interrupt master enable flag
IPS : Interrupt priority selection register
IExxx: Interrupt enable flag
RBS : Register bank selection register
MBS : Memory bank selection register
BS : Bank selection register
PCC : Processor clock control register
. : Address, bit delimiter
(xx) : Contents addressed by xx
xxH : Hexadecimal data

(3) Explanation of symbols in addressing area field

*1	MB=MBE · MBS MB=0, 1, 2, 15	Data Memory Addressing
*2	MB=0	
*3	MBE=0 : MB=0 (000H-07FH) MB=15 (F80H-FFFH) MBE=1 : MB=MBS MB=0, 1, 2, 15	
*4	MB=15, fmem=FB0H-FBFH, FF0H-FFFH	
*5	MB=15, pmem=FC0H-FFFH	
*6	addr=0000H-3F7FH	Program Memory Addressing
*7	addr= (CurrentPC)-15~(CurrentPC)-1 = (CurrentPC)+2~(CurrentPC)+16	
*8	caddr=n000H-nFFFH n=PC_{13}, PC_{12}=0-3	
*9	faddr=0000H-07FFH	
*10	taddr=0020H-007FH	

Remarks 1: MB indicates the accessible memory bank.
2: With *2 MB = 0 regardless of MBE/MBS.
3: With *4 & *5 MB = 15 regardless of MBE/MBS.
4: *6 to *10 indicate the respective addressable areas.

(4) Explanation of machine cycle field

S indicates the number of machine cycles required for the skip operation by an instruction with a skip function. The value of S varies as follows:

- When no skip is performed ..... S = 0
- When the instruction to be skipped is a 1-byte or 2-byte instruction ..... S = 1
- When the instruction to be skipped is a 3-byte instruction* ..... S = 2

*: BR !addr, CALL !addr instruction

NOTE: The GETI instruction is skipped in 1 machine cycle.

One machine cycle is equal to one cycle of the CPU clock ; any of four times can be selected by setting the PCC register (see 5.2.2 (1) "Processor clock control register").

NOTE 1	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Addressing Area	Skip Condition
Transfer	MOV	A, #n4	1	1	A←n4		Stack A
		reg1, #n4	2	2	reg1←n4		
		XA, #n8	2	2	XA←n8		Stack A
		HL, #n8	2	2	HL←n8		Stack B
		rp2, #n8	2	2	rp2←n8		
		A, @HL	1	1	A← (HL)	*1	
		A, @HL+	1	2+S	A← (HL), then L←L+1	*1	L=0
		A, @HL−	1	2+S	A← (HL), then L←L−1	*1	L=FH
		A, @rpal	1	1	A← (rpal)	*2	
		XA, @HL	2	2	XA← (HL)	*1	
		@HL, A	1	1	(HL) ←A	*1	
		@HL, XA	2	2	(HL) ←XA	*1	
		A, mem	2	2	A← (mem)	*3	
		XA, mem	2	2	XA← (mem)	*3	
		mem, A	2	2	(mem) ←A	*3	
		mem, XA	2	2	(mem) ←XA	*3	
		A, reg	2	2	A←reg		
		XA, rp'	2	2	XA←rp'		
		reg1, A	2	2	reg1←A		
		rp'1, XA	2	2	rp'1←XA		
	XCH	A, @HL	1	1	A↔ (HL)	*1	
		A, @HL+	1	2+S	A↔ (HL), then L←L+1	*1	L=0
		A, @HL−	1	2+S	A↔ (HL), then L←L−1	*1	L=FH
		A, @rpal	1	1	A↔ (rpal)	*2	
		XA, @HL	2	2	XA↔ (HL)	*1	
		A, mem	2	2	A↔ (mem)	*3	
		XA, mem	2	2	XA↔ (mem)	*3	
		A, reg1	1	1	A↔reg1		
		XA, rp'	2	2	XA↔rp'		
NOTE 2	MOVT	XA, @PCDE	1	3	XA← $(PC_{13-8}+DE)_{ROM}$		
		XA, @PCXA	1	3	XA← $(PC_{13-8}+XA)_{ROM}$		

NOTE 1: Instruction Group
2: Table reference

NOTE	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Address-ing Area	Skip Condition
Bit transfer	MOV1	CY, fmem. bit	2	2	CY←(fmem. bit)	*4	
		CY, pmem. @L	2	2	CY←($pmem_{7-2}$+L_{3-2}. bit (L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY←(H+mem_{3-0}. bit)	*1	
		fmem. bit, CY	2	2	(fmem. bit) ←CY	*4	
		pmem. @L, CY	2	2	($pmem_{7-2}$+L_{3-2}. bit(L_{1-0})) ←CY	*5	
		@H+mem. bit, CY	2	2	(H+mem_{3-0}. bit) ←CY	*1	
Operation	ADDS	A, #n4	1	1+S	A←A+n4		carry
		XA, #n8	2	2+S	XA←XA+n8		//
		A, @HL	1	1+S	A←A+(HL)	*1	//
		XA, rp'	2	2+S	XA←XA+rp'		//
		rp'1, XA	2	2+S	rp'1←rp'1+XA		//
	ADDC	A, @HL	1	1	A, CY←A+(HL)+CY	*1	
		XA, rp'	2	2	XA, CY←XA+rp'+CY		
		rp'1, XA	2	2	rp'1, CY←rp'1+XA+CY		
	SUBS	A, @HL	1	1+S	A←A−(HL)	*1	borrow
		XA, rp'	2	2+S	XA←XA−rp'		//
		rp'1, XA	2	2+S	rp'1←rp'1−XA		//
	SUBC	A, @HL	1	1	A, CY←A−(HL)−CY	*1	
		XA, rp'	2	2	XA, CY←XA−rp'−CY		
		rp'1, XA	2	2	rp'1, CY←rp'1−XA−CY		
	AND	A, #n4	2	2	A←A∧n4		
		A, @HL	1	1	A←A∧(HL)	*1	
		XA, rp'	2	2	XA←XA∧rp'		
		rp'1, XA	2	2	rp'1←rp'1∧XA		
	OR	A, #n4	2	2	A←A∨n4		
		A, @HL	1	1	A←A∨(HL)	*1	
		XA, rp'	2	2	XA←XA∨rp'		
		rp'1, XA	2	2	rp'1←rp'1∨XA		
	XOR	A, #n4	2	2	A←A∀n4		
		A, @HL	1	1	A←A∀(HL)	*1	
		XA, rp'	2	2	XA←XA∀rp'		
		rp'1, XA	2	2	rp'1←rp'1∀XA		

NOTE : Instruction Group

NOTE 1	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Addressing Area	Skip Condition
NOTE 2	**RORC**	A	1	1	$CY \leftarrow A_0$, $A_3 \leftarrow CY$, $A_{n-1} \leftarrow A_n$		
	NOT	A	2	2	$A \leftarrow \overline{A}$		
NOTE 3	**INCS**	reg	1	1+S	reg←reg+1		reg=0
		rp1	1	1+S	rp1←rp1+1		rp1=00H
		@HL	2	2+S	(HL) ← (HL)+1	*1	(HL)=0
		mem	2	2+S	(mem) ← (mem)+1	*3	(mem)=0
	DECS	reg	1	1+S	reg←reg−1		reg=FH
		rp'	2	2+S	rp'←rp'−1		rp'=FFH
Compare	**SKE**	reg, #n4	2	2+S	Skip if reg=n4		reg=n4
		@HL, #n4	2	2+S	Skip if (HL)=n4	*1	(HL)=n4
		A, @HL	1	1+S	Skip if A= (HL)	*1	A= (HL)
		XA, @HL	2	2+S	Skip if XA= (HL)	*1	XA= (HL)
		A, reg	2	2+S	Skip if A=reg		A=reg
		XA, rp'	2	2+S	Skip if XA=rp'		XA=rp'
NOTE 4	**SET1**	CY	1	1	CY←1		
	CLR1	CY	1	1	CY←0		
	SKT	CY	1	1+S	Skip if CY=1		CY=1
	NOT1	CY	1	1	$CY \leftarrow \overline{CY}$		

NOTE 1: Instruction Group
2: Accumulator Operating
3: Increment/decrement
4: Carry flag manipulating

NOTE	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Addressing Area	Skip Condition
Memory bit manipulating	SET1	mem. bit	2	2	(mem. bit) ←1	*3	
		fmem. bit	2	2	(fmem. bit) ←1	*4	
		pmem. @L	2	2	($pmem_{7-2}+L_{3-2}$. bit (L_{1-0}))←1	*5	
		@H+mem. bit	2	2	(H+mem_{3-0}. bit) ←1	*1	
	CLR1	mem. bit	2	2	(mem. bit) ←0	*3	
		fmem. bit	2	2	(fmem. bit) ←0	*4	
		pmem. @L	2	2	($pmem_{7-2}+L_{3-2}$. bit (L_{1-0}))←0	*5	
		@H+mem. bit	2	2	(H+mem_{3-0}. bit) ←0	*1	
	SKT	mem. bit	2	2+S	Skip if (mem. bit) = 1	*3	(mem. bit) = 1
		fmem. bit	2	2+S	Skip if (fmem. bit) = 1	*4	(fmem. bit) = 1
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}+L_{3-2}$. bit (L_{1-0})) = 1	*5	(pmem. @L) = 1
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}. bit) = 1	*1	(@H+mem. bit) = 1
	SKF	mem. bit	2	2+S	Skip if (mem. bit) = 0	*3	(mem. bit) = 0
		fmem. bit	2	2+S	Skip if (fmem. bit) = 0	*4	(fmem. bit) = 0
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}+L_{3-2}$. bit (L_{1-0})) = 0	*5	(pmem. @L) = 0
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}. bit) = 0	*1	(@H+mem. bit) = 0
	SKTCLR	fmem. bit	2	2+S	Skip if (fmem. bit) = 1 and clear	*4	(fmem. bit) = 1
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}+L_{3-2}$. bit (L_{1-0})) = 1 and clear	*5	(pmem. @L) = 1
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}. bit) = 1 and clear	*1	(@H+mem. bit) = 1
	AND1	CY, fmem. bit	2	2	CY←CY∧(fmem. bit)	*4	
		CY, pmem. @L	2	2	CY←CY∧($pmem_{7-2}+L_{3-2}$. bit (L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY←CY∧(H+mem_{3-0}. bit)	*1	
	OR1	CY, fmem. bit	2	2	CY←CY∨ (fmem. bit)	*4	
		CY, pmem. @L	2	2	CY←CY∨($pmem_{7-2}+L_{3-2}$. bit (L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY←CY∨ (H+mem_{3-0}. bit)	*1	
	XOR1	CY, fmem. bit.	2	2	CY←CY⊻ (fmem. bit)	*4	
		CY, pmem. @L	2	2	CY←CY⊻ ($pmem_{7-2}+L_{3-2}$. bit (L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY←CY⊻(H+mem_{3-0}. bit)	*1	

NOTE: Instruction Group

NOTE	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Addressing Area	Skip Condition
Branch	**BR**	addr	—	—	PC_{13-0}←addr { Most appropriate instruction of BR !addr, BRCB ! caddr and BR $addr is selected by assembler. }	*6	
		! addr	3	3	PC_{13-0}←addr	*6	
		$ addr	1	2	PC_{13-0}←addr	*7	
		PCDE	2	3	PC_{13-0}←PC_{13-8}+DE		
		PCXA	2	3	PC_{13-0}←PC_{13-8}+XA		
	BRCB	! caddr	2	2	PC_{13-0}←PC_{13}, PC_{12}+$caddr_{11-0}$	*8	
Subroutine/stack control	**CALL**	! addr	3	3	(SP−4)(SP−1)(SP−2)←PC_{11-0} (SP−3)←MBE, RBE, PC_{13}, PC_{12} PC_{13-0}←addr, SP←SP−4	*6	
	CALLF	! faddr	2	2	(SP−4)(SP−1)(SP−2)←PC_{11-0} (SP−3)←MBE, RBE, PC_{13}, PC_{12} PC_{13-0}←000+faddr, SP←SP−4	*9	
	RET		1	3	MBE, RBE, PC_{13}, PC_{12}←(SP+1) PC_{11-0}←(SP)(SP+3)(SP+2) SP←SP+4		
	RETS		1	3+S	MBE, RBE, PC_{13}, PC_{12}←(SP+1) PC_{11-0}←(SP)(SP+3)(SP+2) SP←SP+4 then skip unconditionally		Unconditional
	RETI		1	3	MBE, RBE, PC_{13}, PC_{12}, ←(SP+1) PC_{11-0}←(SP)(SP+3)(SP+2) PSW←(SP+4)(SP+5), SP←SP+6		
	PUSH	rp	1	1	(SP−1)(SP−2)←rp, SP←SP−2		
		BS	2	2	(SP−1)←MBS, (SP−2)←RBS, SP←SP−2		
	POP	rp	1	1	rp←(SP+1)(SP), SP←SP+2		
		BS	2	2	MBS←(SP+1), RBS←(SP), SP←SP+2		

NOTE: Instruction Group

NOTE 1	Mnemonic	Operands	No. of Bytes	Machine Cycles	Operation	Address-ing Area	Skip Condition
NOTE 2	EI		2	2	IME (IPS.3) ←1		
		IE×××	2	2	IE×××←1		
	DI		2	2	IME (IPS.3) ←0		
		IE×××	2	2	IE×××←0		
NOTE 3	IN*	A, PORTn	2	2	A←PORTn (n=0−8)		
		XA, PORTn	2	2	XA←PORTn+1, PORTn (n=4, 6)		
	OUT*	PORTn, A	2	2	PORTn←A (n=2−8)		
		PORTn, XA	2	2	PORTn+1, PORTn←XA (n=4, 6)		
NOTE 4	HALT		2	2	Set HALT Mode (PCC.2←1)		
	STOP		2	2	Set STOP Mode (PCC.3←1)		
	NOP		1	1	No Operation		
Special	SEL	RBn	2	2	RBS←n (n=0−3)		
		MBn	2	2	MBS←n (n=0, 1, 2, 15)		
	GETI	taddr	1	3	With TBR instruction PC_{13-0}←(taddr)$_{5-0}$+ (taddr+1)	*10	
					With TCALL instruction (SP−4)(SP−1)(SP−2)←PC_{11-0} (SP−3)←MBE, RBE, PC_{13}, PC_{12} PC_{13-0}←(taddr)$_{5-0}$+ (taddr+1) SP←SP−4		
					With other than TBR and TCALL instructions (taddr) (taddr + 1) instruction execution		Depending on referenced instruction

*: When an IN/OUT instruction is executed, it is necessary to set MBE = 0 or MBE = 1 and MBS = 15.

Remarks: The TBR and TCALL instructions are assembler pseudo-instruction for GETI instruction table definition.

NOTE 1: Instruction Group
2: Interrupt control
3: Input/output
4: CPU control

10.3 INSTRUCTION OPERATION CODES

(1) Explanation of operation code symbols

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

reg

reg1

P_2	P_1	P_0	reg-pair
0	0	0	XA
0	0	1	XA'
0	1	0	HL
0	1	1	HL'
1	0	0	DE
1	0	1	DE'
1	1	0	BC
1	1	1	BC'

rp'

rp'1

Q_2	Q_1	Q_0	addressing
0	0	0	@HL
0	1	0	@HL+
0	1	1	@HL−
1	0	0	@DE
1	0	1	@DL

@rpa

@rpa1

P_2	P_1	reg-pair
0	0	XA
0	1	HL
1	0	DE
1	1	BC

rp2

rp1

rp

N_3	N_2	N_1	N_0	IE×××
0	0	0	0	IEBT
0	0	1	0	IEW
0	1	0	0	IET0
0	1	0	1	IECSI
0	1	1	0	IE0
0	1	1	1	IE2
1	0	0	0	IE4
1	1	0	0	IET1
1	1	1	0	IE1

In: Immediate data for n4, n8

Dn: Immediate data for "mem"

Bn: Immediate data for "bit"

Nn: Immediate data for n, IExxx

Tn: Immediate data for taddr x 1/2

An: Immediate data for [relative address distance (2 to 16) from branch destination address] - 1

Sn: Immediate data for one's complement of [relative address distance (15 to 1) from branch destination address]

(2) Bit manipulation addressing operation codes

*1 in the operand field indicates that the following 3 kinds of addressing are possible:

- fmem.bit
- pmem.@L
- @H+mem.bit

The 2nd byte *2 of the operation code for the above addressing methods is as shown in the table below.

*1	2nd Byte of Operation Code	Accessible Bits
fmem.bit	1 0 B_1 B_0 F_3 F_2 F_1 F_0	Manipulable bits of FB0H through FBFH
	1 1 B_1 B_0 F_3 F_2 F_1 F_0	Manipulable bits of FF0H through FFFH
pmem.@L	0 1 0 0 G_3 G_2 G_1 G_0	Manipulable bits of FC0H through FFFH
@H+mem.bit	0 0 B_1 B_0 D_3 D_2 D_1 D_0	Manipulable bits of accessible memory bank

B_n: Immediate data for "bit"

F_n: Immediate data for "fmem" (indicates low-order 4 bits of address)

G_n: Immediate data for "pmem" (indicates bits 5 through 2 of address)

D_n: Immediate data for "mem" (indicates low-order 4 bits of address)

NOTE 1	Mnemonic	Operands	Operation Code		
			B_1	B_2	B_3
Transfer	MOV	A, #n4	0 1 1 1 I_3 I_2 I_1 I_0		
		reg1, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 1 R_2 R_1 R_0	
		rp, #n8	1 0 0 0 1 P_2 P_1 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @rpa	1 1 1 0 0 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 0	
		@HL, A	1 1 1 0 1 0 0 0		
		@HL, XA	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 0	
		A, mem	1 0 1 0 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		mem, A	1 0 0 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		mem, XA	1 0 0 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg	1 0 0 1 1 0 0 1	0 1 1 1 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 1 1 P_2 P_1 P_0	
		reg1, A	1 0 0 1 1 0 0 1	0 1 1 1 0 R_2 R_1 R_0	
		rp'1, XA	1 0 1 0 1 0 1 0	0 1 0 1 0 P_2 P_1 P_0	
	XCH	A, @rpa	1 1 1 0 1 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 1	
		A, mem	1 0 1 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg1	1 1 0 1 1 R_2 R_1 R_0		
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 0 P_2 P_1 P_0	
NOTE 2	MOVT	XA, @PCDE	1 1 0 1 0 1 0 0		
		XA, @PCXA	1 1 0 1 0 0 0 0		
NOTE 3	MOV1	CY, [*1]	1 0 1 1 1 1 0 1	*2	
		[*1], CY	1 0 0 1 1 0 1 1	*2	

NOTE 1: Instruction Group
2: Table reference
3: Bit transfer

NOTE 1	Mnemonic	Operands	Operation Code		
			B_1	B_2	B_3
Operation	ADDS	A, #n4	0 1 1 0 I_3 I_2 I_1 I_0		
		XA, #n8	1 0 1 1 1 0 0 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @HL	1 1 0 1 0 0 1 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 0 0 P_2 P_1 P_0	
	ADDC	A, @HL	1 0 1 0 1 0 0 1		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 1 0 P_2 P_1 P_0	
	SUBS	A, @HL	1 0 1 0 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 0 0 P_2 P_1 P_0	
	SUBC	A, @HL	1 0 1 1 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 1 0 P_2 P_1 P_0	
	AND	A, #n4	1 0 0 1 1 0 0 1	0 0 1 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 0 1 0 P_2 P_1 P_0	
	OR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 0 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 0 0 P_2 P_1 P_0	
	XOR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 1 0 P_2 P_1 P_0	
NOTE 2	RORC	A	1 0 0 1 1 0 0 0		
	NOT	A	1 0 0 1 1 0 0 1	0 1 0 1 1 1 1 1	

NOTE 1 : Instruction Group
2 : Accumulator operating

NOTE	Mnemonic	Operands	Operation Code		
			B_1	B_2	B_3
Increment/decrement	INCS	reg	1 1 0 0 0 R_2 R_1 R_0		
		rp1	1 0 0 0 1 P_2 P_1 0		
		@HL	1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0	
		mem	1 0 0 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
	DECS	reg	1 1 0 0 1 R_2 R_1 R_0		
		rp'	1 0 1 0 1 0 1 0	0 1 1 0 1 P_2 P_1 P_0	
Compare	SKE	reg, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 0 R_2 R_1 R_0	
		@HL, #n4	1 0 0 1 1 0 0 1	0 1 1 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 0 0 0 0 0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 1	
		A, reg	1 0 0 1 1 0 0 1	0 0 0 0 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 1 P_2 P_1 P_0	
Carry flag manipulation	SET1	CY	1 1 1 0 0 1 1 1		
	CLR1	CY	1 1 1 0 0 1 1 0		
	SKT	CY	1 1 0 1 0 1 1 1		
	NOT1	CY	1 1 0 1 0 1 1 0		
Memory bit manipulation	SET1	mem. bit	1 0 B_1 B_0 0 1 0 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 0 1 1 1 0 1	*2	
	CLR1	mem. bit	1 0 B_1 B_0 0 1 0 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 0 1 1 1 0 0	*2	
	SKT	mem. bit	1 0 B_1 B_0 0 1 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 1 1 1 1 1 1	*2	
	SKF	mem. bit	1 0 B_1 B_0 0 1 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 1 1 1 1 1 0	*2	
	SKTCLR	[*1]	1 0 0 1 1 1 1 1	*2	
	AND1	CY, [*1]	1 0 1 0 1 1 0 0	*2	
	OR1	CY, [*1]	1 0 1 0 1 1 1 0	*2	
	XOR1	CY, [*1]	1 0 1 1 1 1 0 0	*2	

NOTE : Instruction Group

NOTE 1	Mnemonic	Operands	Operation Code		
			B_1	B_2	B_3
Branch	BR	! addr	1 0 1 0 1 0 1 1	0 0 ← addr →	
		$ addr (+16) ≤ (+2) (−1) ≤ (−15)	0 0 0 0 A_3 A_2 A_1 A_0		
			1 1 1 1 S_3 S_2 S_1 S_0		
		PCDE	1 0 0 1 1 0 0 1	0 0 0 0 0 1 0 0	
		PCXA	1 0 0 1 1 0 0 1	0 0 0 0 0 0 0 0	
	BRCB	! caddr	0 1 0 1 ← caddr →		
Subroutine/stack control	CALL	! addr	1 0 1 0 1 0 1 1	0 1 ← addr →	
	CALLF	! faddr	0 1 0 0 0 ← faddr →		
	RET		1 1 1 0 1 1 1 0		
	RETS		1 1 1 0 0 0 0 0		
	RETI		1 1 1 0 1 1 1 1		
	PUSH	rp	0 1 0 0 1 P_2 P_1 1		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 1	
	POP	rp	0 1 0 0 1 P_2 P_1 0		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 0	
Input/output	IN	A, PORTn	1 0 1 0 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
		XA, PORTn	1 0 1 0 0 0 1 0	1 1 1 1 N_3 N_2 N_1 N_0	
	OUT	PORTn, A	1 0 0 1 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
		PORTn, XA	1 0 0 1 0 0 1 0	1 1 1 1 N_3 N_2 N_1 N_0	
NOTE 2	EI		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 1	1 0 N_5 1 1 N_2 N_1 N_0	
	DI		1 0 0 1 1 1 0 0	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 0	1 0 N_5 1 1 N_2 N_1 N_0	
NOTE 3	HALT		1 0 0 1 1 1 0 1	1 0 1 0 0 0 1 1	
	STOP		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 1	
	NOP		0 1 1 0 0 0 0 0		
Special	SEL	RBn	1 0 0 1 1 0 0 1	0 0 1 0 0 0 N_1 N_0	
		MBn	1 0 0 1 1 0 0 1	0 0 0 1 N_3 N_2 N_1 N_0	
	GETI	taddr	0 0 T_5 T_4 T_3 T_2 T_1 T_0		

NOTE 1: Instruction Group
2: Interrupt control
3: CPU control

10.4 INSTRUCTION FUNCTIONS AND USE

10.4.1 TRANSFER INSTRUCTIONS

MOV A, #n4

Function: A ← n4 n4 = I_3 to I_0: 0 to FH

Transfers the 4-bit immediate data n4 to the A register (4-bit accumulator). This instruction has a stacking effect (Group A) and if followed by a MOV A, #n4 or MOV XA, #n8 instruction, the stacked instructions following the executed instruction are processed as NOP instructions.

Examples:

(1) To set 0BH in the accumulator.

```
MOV A, #0BH
```

(2) To select data for output to port 3 from 0 through 2.

```
A0:  MOV A, #0
A1:  MOV A, #1
A2:  MOV A, #2
     OUT PORT3, A
```

MOV reg1, #n4

Function: reg1 ← n4 n4 = I_3 to I_0 0 to FH

Transfers the 4-bit immediate data n4 to A register reg1 (X, H, L, D, E, B, C).

MOV rp, #n8

Function: rp ← n8 n8 = I_7 to I_0: 00H to FFH

Transfers the 8-bit immediate data n8 to register pair rp (XA, HL, DE, BC). When XA or HL is selected as the register pair this instruction has a stacking effect. Stacking effects fall into two groups, Group A (MOV A, #n4 instruction & MOV XA, #n8 configuration) and Group B (MOV HL, #n8 instruction); if instructions of the same group are located consecutively, stacked instructions following the executed instruction are processed as NOP instructions.

Example:

To set 5FH in the DE register pair

MOV DE, #5FH

MOV A, @rpa

Function: A ← (rpa)
When rpa = HL+: skip if L = 0
When rpa = HL-: skip if L = FH

Transfers the contents of the data memory addressed by register pair rpa (HL, HL+, HL-, DE, DL) to the A register.
When auto-incrementing (HL+) is specified for rpa, after the data transfer the contents of the L register are automatically adjusted by +1, and if the result is 0, the next one instruction is skipped.

When auto-decrementing (HL-) is specified for rpa, after the data transfer the contents of L register are automatically adjusted by -1, and if the result is FH, the next one instruction is skipped.

MOV XA, @HL

Function: A ← (HL), X ← (HL + 1)

Transfers the contents of the data memory addressed by register pair HL to the A register, and transfers the contents of the next memory address to the X register.

If the contents of the L register are odd, the address with the LSB ignored is transferred.

Example:

To transfer the contents of addresses 3EH and 3FH to register pair XA.

```
MOV HL, #3EH
MOV XA, @HL
```

MOV @HL, A

Function: (HL) ← A

Transfers the contents of the A register to the data memory addressed by register pair HL.

MOV @HL, XA

Function: (HL) ← A, (HL + 1) ← X

Transfers the contents of the A register to the data memory addressed by register pair HL, and transfers the contents of the X register to the next memory address.

If the contents of the L register are odd, the address with the LSB ignored is specified.

MOV A, mem

Function: A ← (mem) mem = D_7 to D_0: 00H to FFH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register.

MOV XA, mem

Function: A ← (mem), X ← (mem + 1)
mem = D_7 to D_0: 00H to FEH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register, and the contents of the next address to the X register.

The address specified by mem must be an even address.

Example:

To transfer the data in addresses 40H and 41H to register pair XA.

MOV XA, 40H

MOV mem, A

Function: (mem) ← A mem = D_7 to D_0: 00H to FFH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem.

MOV mem, XA

Function: (mem) ← A, (mem + 1) ← X
mem = D_7 to D_0: 00H to FEH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem, and transfers the contents of the X register to the next memory address.

The address specified by mem must be an even address.

MOV A, reg

Function: A ← reg

Transfers the contents of register reg (X, A, H, L, D, E, B, C) to the A register.

MOV XA, rp'

Function: XA ← rp'

Transfers the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') to the register pair XA.

Example:

To transfer the data in register pair XA' to register pair XA.

MOV XA, XA'

MOV reg1, A

Function: reg1 ← A

Transfers the contents of the A register to register reg1 (X, H, L, D, E, B, C).

MOV rp'1, XA

Function: rp'1 ← XA

Transfers the contents of register pair XA to register pair rp'1 (HL, DE, BC, XA', HL', DE', BC').

XCH A, @rpa

Function: A ↔ (rpa)
When rpa = HL+: skip if L = 0
When rpa = HL-: skip if L = FH

Exchanges the contents of the A register with the contents of the data memory addressed by register pair rpa (HL, HL+, HL-, DE, DL).
When auto-incrementing (HL+)is specified for rpa, after the data exchange the contents of the L register are automatically adjusted by +1, and if the result is 0, the next instruction is skipped.

When auto-decrementing (HL-) is specified for rpa, after the data exchange the contents of L register are automatically adjusted by -1 and if the result is FH, the next one instruction is skipped.

Example:

To exchange the data in data memory addresses 20H to 2FH with the data in addresses 30H to 3FH.

```
        SEL  MB0
        MOV  D, #2
        MOV  HL, #30H
LOOP:   XCH  A, @HL ; A ←→ (3 x)
        XCH  A, @DL ; A ←→ (2 x)
        XCH  A, @HL+; A ←→ (3 x)
        BR   LOOP
```

XCH XA, @HL

Function: A ↔ (HL), X ↔ (HL + 1)

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL, and exchanges the contents of the X register with the contents of the next memory address.

If the contents of the L register are odd, the address with the LSB ignored is specified.

XCH A, mem

Function: A ↔ (mem) mem = D_7 to D_0: 00H to FEH

Exchanges the contents of the A register with the contents of the data memory addressed by 8-bit immediate data mem.

XCH XA, mem

Function: A ↔ (mem), X ↔ (mem + 1)
mem = D_7 to D_0: 00H to FEH

Exchanges the contents of the A register with the contents of the data memory addressed by 8-bit immediate data mem, and exchanges the contents of the X register with the contents of the next memory address.

The address specified by mem must be an even address.

XCH A, reg1

Function: A ↔ reg1

Exchanges the contents of the A register with the contents of register reg1 (X, H, L, D, E, B, C).

XCH XA, rp'

Function: XA ↔ rp'

Exchanges the contents of register pair XA with the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC').

10.4.2 TABLE REFERENCING INSTRUCTIONS

MOVT XA, @PCDE

Function: XA ← ROM (PC_{13} to PC_8 + DE)

Transfers to the A register the low-order 4 bits of the table data in the program memory addressed when the low-order 8 bits (PC_7 to PC_0) of the program counter (PC) are replaced with the contents of register pair DE, and transfers the high-order 4 bits of this table data to the X register.

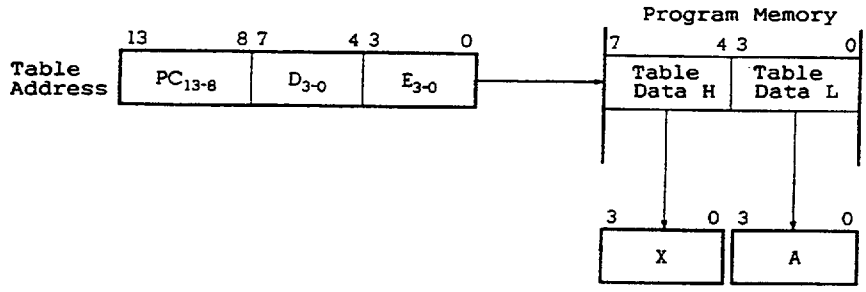
The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must previously have been programmed in the table area by an assembler pseudo-instruction (DB instruction).

The program counter is not affected by the execution of this instruction.

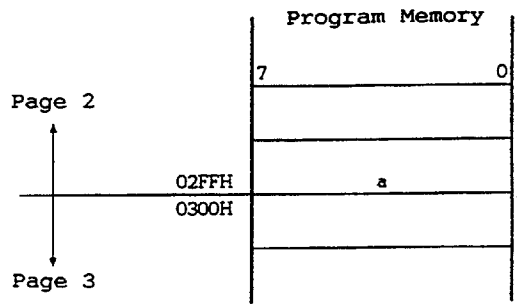
This instruction is effective for successive table data references.

Example:



Note:

The MOVT XA, @PCDE instruction normally references table data in the page in which that instruction is located, but if the instruction is in address xxFFH, the table data in the next page, to that page, is referenced.



If, for example, the MOVT XA, @PCDE instruction is in location "a" in the above figure, the table data in page 3, not page 2, specified by the contents of register pair DE is transferred to register pair XA.

Example:

To transfer the 16-byte data in program memory addresses xxF0H through xxFFH to data memory addresses 30H through 4FH.

```
SUB :  SEL   MB0
       MOV   HL, #30H          ; HL ← 30H
       MOV   DE, #0F0H         ; DE ← F0H
LOOP:  MOVT  XA, @PCDE         ; XA ← Table data
       MOV   @HL, XA           ; (HL) ← XA
       INCS  HL                ; HL ← HL + 2
       INCS  HL
       INCS  E                 ; E ← E + 1
       BR    LOOP
       RET
       ORG   xxF0H
       DB     xxH, xxH, .....; Table data
```

MOVT XA, @PCXA

Function: XA ← ROM (PC_{13} to PC_8 + XA)

Transfers to the A register the low-order 4 bits of the table data in the program memory addressed when the low-order 8 bits (PC_7 to PC_0) of the program counter (PC) are replaced with the contents of register pair XA, and transfers the high-order 4 bits of this table data to the X register.

The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must previously have been programmed in the table area by an assembler pseudo-instruction (DB instruction).

The program counter is not affected by the execution of this instruction.

Note:

As with the MOVT XA, @PCDE instruction, if this instruction is located in address xxFFH the table data in the page is transferred.

10.4.3 BIT TRANSFER INSTRUCTIONS

MOV1 CY, fmem.bit

MOV1 CY, pmem.@L

MOV1 CY, @H + mem.bit

Function: CY ← (bit specified by operand)

Transfers the contents of the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit) to the carry flag (CY).

MOV1 fmem.bit, CY

MOV1 pmem.@L, CY

MOV1 @H + mem.bit, CY

Function: (Bit specified by operand) ← CY

Transfers the contents of the carry flag (CY) to the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit).

Example:

To output the flag at bit 3 of data memory address 3FH to bit 2 of port 3.

```
FLAG  EQU 3FH. 3
SEL   MB0
MOV   H, #FLAG SHR6; H ← High-order 4 bits of FLAG
MOV1  CY, @H + FLAG; CY ← FLAG
MOV1  PORT3. 2, CY ; P32 ← CY
```

10.4.4 OPERATION INSTRUCTIONS

ADDS A, #n4

Function: A ← A + n4; Skip if carry.
n4 = I_3 to I_0: 0 to FH

Performs binary addition of the 4-bit immediate data n4 to the contents of the A register, and skips the next instruction if a carry is generated as a result of the addition. The carry flag is not affected.

Combining the ADDC A, @HL instruction and the SUBC A, @HL instruction gives a radix adjustment instruction (see 10.1 "Special Instructions").

ADDS XA, #n8

Function: XA ← XA + n8; Skip if carry.
n8 = I_7 to I_0: 00H to FFH

Performs binary addition of the 8-bit immediate data n8 to the contents of register pair XA, and skips the next instruction if a carry is generated as a result of the addition. The carry flag is not affected.

ADDS A, @HL

Function: A ← A + (HL); Skip if carry.

Performs binary addition of the contents of the data memory addressed by register pair HL to the contents of the A register, and skips the next instruction if a carry is generated as a result of the addition. The carry flag is not affected.

ADDS XA, rp'

Function: XA ← XA + rp': Skip if carry.

Performs binary addition of the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') to the contents of register pair XA, and skips the next instruction if a carry is generated as a result of the addition. The carry flag is not affected.

ADDS rp'1, XA

Function: rp' ← rp'1 + XA; Skip if carry.

Performs binary addition of the contents of register pair XA to the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC'), and skips the next instruction if a carry is generated as a result of the addition. The carry flag is not affected.

Example:

To left-shift a register pair.

```
MOV   XA, rp'1
ADDS  rp'1, XA
NOP
```

ADDC A, @HL

Function: A, CY ← A + (HL) + CY

Performs binary addition including the carry flag of the contents of the data memory addressed by register pair HL to the contents of the A register. If a carry is generated as a result of the addition the carry flag is set, and if no carry is generated the carry flag is reset.

If an ADDS A, #n4 instruction is placed after this instruction, if a carry is generated in this instruction the ADDS A, #n4 instruction is skipped. If no carry is generated the ADDS A, #n4 instruction is executed, and a function is produced whereby skipping of the ADDS A, #n4 instruction is disabled. Therefore, the combination of these instructions can be used as a radix adjustment instruction (see 10.1.4 "Radix Adjustment Instructions").

ADDC XA, rp'

Function: XA, CY ← XA + rp' + CY

Performs binary addition including the carry flag of the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') to the contents of register pair XA. If a carry is generated as a result of the addition the carry flag is set, and if no carry is generated the carry flag is reset.

ADDC rp'1, XA

Function: rp'1, CY ← rp'1 + XA + CY

Performs binary addition including the carry flag of the contents of register pair XA to the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC'). If a carry is generated as a result of the addition the carry flag is set, and if no carry is generated the carry flag is reset.

SUBS A, @HL

Function: A ← A - (HL); Skip if borrow

Subtracts the contents of the data memory addressed by register pair HL from the contents of the A register, and places the result in the A register. If a borrow is generated as a result of the subtraction, the next instruction is skipped.

The carry flag is not affected.

SUBS XA, rp'

Function: XA ← XA - rp'; Skip if borrow

Subtracts the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') from the contents of register pair XA, and places the result in register pair XA. If a borrow is generated as a result of the subtraction, the next instruction is skipped.

The carry flag is not affected.

Example:

To compare the relative size of data memory and register pair contents.

```
MOV   XA, mem
SUBS  XA, rp'
            ; (mem) ≥ rp'
            ; (mem) < rp'
```

SUBS rp'1, XA

Function: rp'1 ← rp'1 - XA; Skip if borrow

Subtracts the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC'), and places the result in the specified register pair rp'1. If a borrow is generated as a result of the subtraction, the next instruction is skipped.

The carry flag is not affected.

SUBC A, @HL

Function: A, CY ← A - (HL) - CY

Performs subtraction including the carry flag of the contents of the data memory addressed by register pair HL from the contents of the A register, and places the result in the A register. If a borrow is generated as a result of the subtraction the carry flag is set, and if no borrow is generated the carry flag is reset.

If an ADDS A, #n4 instruction is placed after this instruction, if no borrow is generated in this instruction the ADDS A, #n4 instruction is skipped. If a borrow is generated the ADDS A, #n4 instruction is executed, and a function is produced whereby skipping of the ADDS A, #n4 instruction is disabled. Therefore, the combination of these instructions can be used as a radix adjustment instruction (see 10.1.4 "Radix Adjustment Instructions").

SUBC XA, rp'

Function: XA, CY ← XA - rp' - CY

Performs subtraction including the carry flag of the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') from the contents of register pair XA, and places the result in register pair XA. If a borrow is generated as a result of the subtraction the carry flag is set, and if no borrow is generated the carry flag is reset.

SUBC rp'1, XA

Function: rp'1, CY ← rp'1 - XA - CY

Performs subtraction including the carry flag of the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC') and places the result in register pair rp'1. If a borrow is generated as a result of the subtraction the carry flag is set, and if no borrow is generated the carry flag is reset.

AND A, #n4

Function: A ← A ∧ n4 n4 = I_3 to I_0: 0 to FH

Obtains the logical product of the contents of the A register and the 4-bit immediate data n4, and places the result in the A register.

Example:

To zeroize the high-order 2 bits of the accumulator.

AND A, #0011B

AND A, @HL

Function: A ← A ∧ (HL)

Obtains the logical product of the contents of the A register and the contents of the data memory addressed by register pair HL, and places the result in the A register.

AND XA, rp'

Function: XA ← XA ∧ rp'

Obtains the logical product of the contents of register pair XA and the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC'), and places the result in register pair XA.

AND rp'1, XA

Function: rp'1 ← rp'1 ∧ XA

Obtains the logical product of the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC') and the contents of register pair XA, and places the result in the specified register pair.

OR A, #n4

Function: A ← A ∨ n4 n4 = I_3 to I_0: 0 to FH

Obtains the logical sum of the contents of the A register and the 4-bit immediate data n4, and places the result in the A register.

Example:

To set the low-order 3 bits of the accumulator to 1.

OR A, #0111B

OR A, @HL

Function: A ← A ∨ (HL)

Obtains the logical sum of the contents of the A register and the contents of the data memory addressed by register pair HL, and places the result in the A register.

OR XA, rp'

Function: XA ← XA ∨ rp'

Obtains the logical sum of the contents of register pair XA and the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC'), and places the result in register pair XA.

OR rp'1, XA

Function: rp'1 ← rp'1 ∨ XA

Obtains the logical sum of the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC') and the contents of register pair XA, and places the result in register pair rp'1.

XOR A, #n4

Function: A ← A ∀ n4 n4 = I_3 to I_0: 0 to FH

Obtains the exclusive logical sum of the contents of the A register and the 4-bit immediate data n4, and places the result in the A register.

Example:

To invert the high-order 4 bits of the accumulator.

XOR A, #1000B

XOR A, @HL

Function: A ← A ∀ (HL)

Obtains the exclusive logical sum of the contents of the A register and the contents of the data memory addressed by register pair HL, and places the result in the A register.

XOR XA, rp'

Function: XA ← XA ∀ rp'

Obtains the exclusive logical sum of the contents of register pair XA and the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC'), and places the result in register pair XA.

XOR rp'1, XA

Function: rp'1 ← rp'1 ∀ XA

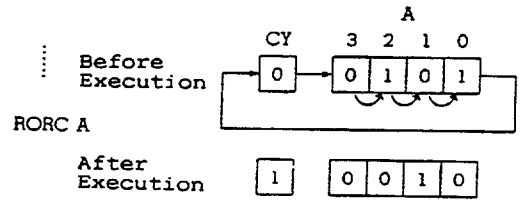
Obtains the exclusive logical sum of the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', BC') and the contents of register pair XA, and places the result in register pair rp'1.

10.4.5 ACCUMULATOR MANIPULATING INSTRUCTIONS

RORC A

Function: $CY \leftarrow A_0$, $A_{n-1} \leftarrow A_n$, $A_3 \leftarrow CY$ (n = 1 to 3)

Performs bit-wise right rotation including the carry flag of the contents of the A register (4-bit accumulator).



NOT A

Function: $A \leftarrow \overline{A}$

Obtains the one's complement (inverts each bit) of the A register (4-bit accumulator).

10.4.6 INCREMENT/DECREMENT INSTRUCTIONS

INCS reg

Function: reg ← reg + 1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L, D, E, B, C). If reg = 0 as a result of the increment, the next instruction is skipped.

INCS rpl

Function: rpl ← rpl + 1; Skip if rpl = 00H

Increments the contents of the register pair rpl (HL, DE, BC). If rpl = 00H as a result of the increment, the next instruction is skipped.

INCS @HL

Function: (HL) ← (HL) + 1; Skip if (HL) = 0

Increments the contents of the data memory addressed by register pair HL. If the contents of that data memory are 0 as a result of the increment, the next instruction is skipped.

INCS mem

Function: (mem) ← (mem) + 1; Skip if (mem) = 0,
mem = D_7 to D_0: 00H to FFH

Increments the contents of the data memory addressed by the 8-bit immediate data mem. If the contents of that data memory are 0 as a result of the increment, the next instruction is skipped.

DECS reg

Function: reg ← reg - 1; Skip if reg = FH

Decrements the contents of register reg (X, A, H, L, D, E, B, C). If reg = FH as a result of the decrement, the next instruction is skipped.

DECS rp'

Function: rp' ← rp' - 1; Skip if rp' = FFH

Decrements the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC'). If rp' = FFH as a result of the decrement, the next instruction is skipped.

10.4.7 COMPARE INSTRUCTIONS

SKE reg, #n4

Function: Skip if reg = n4 n4 = I_3 to I_0: 0 to FH

Skips the next instruction if the contents of register reg (X, A, H, L, D, E, B, C) and the 4-bit immediate data n4 are equal.

SKE @HL, #n4

Function: Skip if (HL) = n4 n4 = I_3 to I_0: 0 to FH

Skips the next instruction if the contents of the data memory addressed by register pair HL and the 4-bit immediate data n4 are equal.

SKE A, @HL

Function: Skip if A = (HL)

Skips the next instruction if the contents of the A register and the contents of the data memory addressed by register pair HL are equal.

SKE XA, @HL

Function: Skip if A = (HL) and X = (HL + 1)

Skips the next instruction if the contents of the A register and the contents of the data memory addressed by register pair HL are equal, and the contents of the X register and the contents of the next data memory address are equal.

If the contents of the L register are odd, the address with the LSB ignored is specified.

SKE A, reg

Function: Skip if A = reg

Skips the next instruction if the contents of the A register and the contents of register reg (X, A, H, L, D, E, B, C) are equal.

SKE XA, rp'

Function: Skip if XA = rp'

Skips the next instruction if the contents of register pair XA and the contents of the register pair rp' (XA, HL, DE, BC, XA', HL', DE', BC') are equal.

10.4.8 CARRY FLAG MANIPULATING INSTRUCTIONS

SET1 CY

Function: CY ← 1

Sets the carry flag.

CLR1 CY

Function: CY ← 0

Clears the carry flag.

SKT CY

Function: Skip if CY = 1

Skips the next instruction if the carry flag is 1.

NOT1 CY

Function: CY ← $\overline{CY}$

Complements the carry flag. The flag is changed from 0 to 1, or from 1 to 0.

10.4.9 MEMORY BIT MANIPULATION INSTRUCTIONS

SET1 mem.bit

Function: (mem.bit) ← 1 mem = D_7 to D_0 : 00H to FFH,
bit = B_1 and B_0 : 0 to 3

Sets the bit specified by the 2-bit immediate data "bit" of the address indicated by the 8-bit immediate data mem.

SET1 fmem.bit

SET1 pmem.@L

SET1 @H + mem.bit

Function: (Bit specified by operand) ← 1

Sets the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit).

CLR1 mem.bit

Function: (mem.bit) ← 0 mem = D_7 to D_0 : 00H to FFH,
bit = B_1 and B_0 : 0 to 3

Clears the bit specified by the 2-bit immediate data "bit" of the address indicated by the 8-bit immediate data mem.

CLR1 fmem.bit

CLR1 pmem.@L

CLR1 @H + mem.bit

Function: (Bit specified by operand) ← 0

Clears the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit).

SKT mem.bit

Function: Skip if (mem.bit) = 1
mem = D_7 to D_0 : 00H to FFH,
bit = B_1 and B_0: 0 to 3

Skips the next instruction if the bit specified by the 2-bit immediate data "bit" of the address indicated by the 8-bit immediate data mem is 1.

SKT fmem.bit

SKT pmem.@L

SKT @H + mem.bit

Function: Skip if (bit specified by operand) = 1

Skips the next instruction if the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit) is 1.

SKF mem.bit

Function: Skip if (mem.bit) = 0
mem = D_7 to D_0 : 00H to FFH,
bit = B_1 and B_0: 0 to 3

Skips the next instruction if the bit specified by the 2-bit immediate data "bit" of the address indicated by the 8-bit immediate data mem is 0.

SKF fmem.bit

SKF pmem.@L

SKF @H + mem.bit

Function: Skip if (bit specified by operand) = 0

Skips the next instruction if the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit) is 0.

SKTCLR fmem.bit

SKTCLR pmem.@L

SKTCLR @H + mem.bit

Function: Skip if (bit specified by operand) = 1 then clear

Skips the next instruction if the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit) is 1, then clears that bit to "0".

AND1 CY, fmem.bit

AND1 CY, pmem.@L

AND1 CY, @H + mem.bit

Function: CY ← CY ∧ (bit specified by operand)

Obtains the logical product of the contents of the carry flag and the contents of the data memory specified by bit manipulation addressing(fmem.bit, pmem.@L, @H + mem.bit), and places the result in the carry flag.

OR1 CY, fmem.bit

OR1 CY, pmem.@L

OR1 CY, @H + mem.bit

Function: CY ← CY ∨ (bit specified by operand)

Obtains the logical sum of the contents of the carry flag and the contents of the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit), and places the result in the carry flag.

XOR1 CY, fmem.bit

XOR1 CY, pmem.@L

XOR1 CY, @H + mem.bit

Function: CY ← CY ∀ (bit specified by operand)

Obtains the exclusive logical sum of the contents of the carry flag and the contents of the data memory bit specified by bit manipulation addressing (fmem.bit, pmem.@L, @H + mem.bit), and places the result in the carry flag.

10.4.10 BRANCH INSTRUCTIONS

BR addr

Function: PC_{13} to PC_0 ← addr
addr = 0000H to 3F7FH

Branches to the address specified by immediate data addr.

This instruction is an assembler pseudo-instruction, and during assembly is automatically replaced by the assembler with the most suitable of the following instructions: BR !addr, BRCB !caddr, or BR $addr.

BR !addr

Function: PC_{13} to PC_0 ← addr
addr = 0000H to 3F7FH

Transfers immediate data addr to the program counter (PC), and branches to the address specified by the PC.

BR $addr

Function: PC_{13} to PC_0 ← addr

Relative branch instruction with a branching range of (-15 to -1) and (+2 to +16) from the current address. It is not affected by page boundaries or block boundaries.

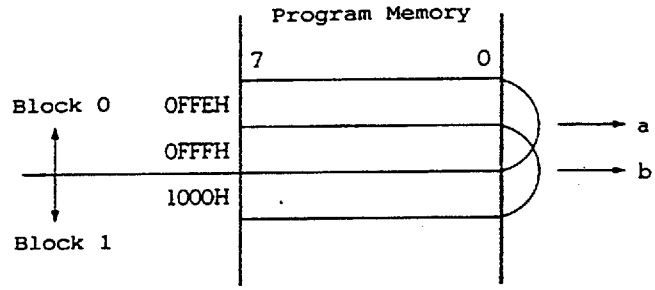
BRCB !caddr

Function: PC_{13} to PC_0 ← PC_{13}, PC_{12} + $caddr_{11}$ to $caddr_0$
caddr = n000H to nFFFH
n = PC_{13}, PC_{12} = 0 to 3

Branches to the address indicated by replacing the low-order bits of the program counter (PC_{11} to PC_0) with the 12-bit immediate data caddr.

Note:

The BRCB !caddr instruction usually branches within the block in which the instruction is located, but if the 1st byte is in address 0FFEH or address 0FFFH, it branches not to block 0 but to block 1.



If the BRCB !caddr instruction is located at "a" or "b" in the above diagram, the branch is performed to block 1, not to block 0.

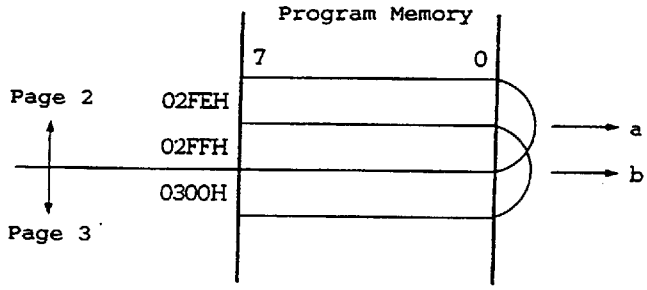
BR PCDE

Function: PC_7 to PC_4 ← D, PC_3 to PC_0 ← E

Branches to the address indicated by replacing the low-order 8 bits (PC_7 to PC_0) of the program counter with the contents of register pair DE. The high-order bits of the program counter are not affected.

Note:

The BR PCDE instruction usually branches within the page in which the instruction is located, but if the 1st byte of the operation code is in address xxFEH or address xxFFH, it branches not to that page but to the next page.



If, for example, the BR PCDE instruction is located at "a" or "b" in the above diagram, the branch is performed to the low-order 8-bit address specified by the contents of register pair DE not in page 2 but in page 3.

BR PCXA

Function: PC_7 to $PC_4 \leftarrow X$, PC_3 to $PC_0 \leftarrow A$

Branches to the address indicated by replacing the low-order 8 bits (PC_7 to PC_0) of the program counter with the contents of register pair XA. The high-order bits of the program counter are not affected.

Note:

As with the BR PCDE instruction, if the 1st byte is in address xxFEH or address xxFFH, a branch is made not within the same page but to the next page.

TBR addr

Function:

An assembler pseudo-instruction or GETI instruction table definition. Used when replacing a 3-byte BR !addr instruction with a 1-byte GETI instruction. 12-bit address data is specified for addr. For details, see the "RA75X Assembler Package User's Manual, Language Volume".

aadr = 0000H to 3F7FH

10.4.11 SUBROUTINE/STACK CONTROL INSTRUCTIONS

CALL !addr

Function:

$(SP - 1) \leftarrow PC_7$ to PC_4, $(SP - 2) \leftarrow PC_3$ to PC_0
$(SP - 3) \leftarrow$ MBE, RBE, PC_{13}, PC_{12}
$(SP - 4) \leftarrow PC_{11}$ to PC_8, PC_{13} to $PC_0 \leftarrow$ addr,
$SP \leftarrow SP - 4$

addr = 0000H to 3F7FH

Saves the contents of the program counter (return address) and MBE & RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by the 14-bit immediate data addr.

CALLF !faddr

Function:

$(SP - 1) \leftarrow PC_7$ to PC_4, $(SP - 2) \leftarrow PC_3$ to PC_0
$(SP - 3) \leftarrow$ MBE, RBE, PC_{13}, PC_{12}
$(SP - 4) \leftarrow PC_{11}$ to PC_8, $SP \leftarrow SP - 4$
PC_{13} to $PC_0 \leftarrow 000 +$ faddr

faddr = 0000H to 07FFH

Saves the contents of the program counter (PC; return address) and MBE & RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by the 11-bit immediate data faddr. Only addresses in the range 0000H to 07FFH (0 to 2047) can be called.

TCALL !addr

Function:

An assembler pseudo-instruction for GETI instruction table definition. Used when replacing a 3-byte CALL !addr instruction with a 1-byte GETI instruction. 12-bit address data is specified for addr. For details, see the "RA75X Assembler Package User's Manual, Language Volume".

addr = 0000H to 3F7FH

RET

Function: PC_{11} to $PC_8 \leftarrow (SP)$,
MBE, RBE, PC_{13}, $PC_{12} \leftarrow (SP + 1)$,
PC_3 to $PC_0 \leftarrow (SP + 2)$,
PC_7 to $PC_4 \leftarrow (SP + 3)$, $SP \leftarrow SP + 4$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE) and register bank enable flag (RBE), and then increments the contents of the SP.

Note:

Program status word (PSW) bits other than the MBE and RBE flags are not restored.

RETS

Function: PC_{11} to PC_8 ← (SP),
MBE, RBE, PC_{13}, PC_{12} ← (SP + 1),
PC_3 to PC_0 ← (SP + 2),
PC_7 to PC_4 ← (SP + 3), SP ← SP + 4
Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE) and register bank enable flag (RBE), increments the contents of the SP, and then skips unconditionally.

Note:

Program status word (PSW) bits other than the MBE and RBE flags are not restored.

RETI

Function: PC_{11} to PC_8 ← (SP),
MBE, RBE, PC_{13}, PC_{12} ← (SP + 1),
PC_3 to PC_0 ← (SP + 2),
PC_7 to PC_4 ← (SP + 3)
PSW_L ← (SP + 4), PSW_H ← (SP + 5)
SP ← SP + 6

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and the program status word, and then increments the contents of the SP.

This instruction is used to return from an interrupt service routine.

PUSH rp

Function: $(SP - 1) \leftarrow rp_H$, $(SP - 2) \leftarrow rp_L$, $SP \leftarrow SP - 2$

Saves the contents of register pair rp (XA, HL, DE, BC) to the data memory (stack) addressed by the stack pointer (SP), and then decrements the SP.

The higher register of the register pair (rp_H: X, H, D, B) is saved to the stack memory addressed by (SP - 1), and the lower register (rp_L: A, L, E, C) is saved to the stack memory addressed by (SP - 2).

PUSH BS

Function: $(SP - 1) \leftarrow MBS$, $(SP - 2) \leftarrow RBS$, $SP \leftarrow SP - 2$

Saves the contents of the memory bank selection register (MBS) and the register bank selection register (RBS) to the data memory (stack) addressed by the stack pointer (SP), and then decrements the SP.

POP rp

Function: $rp_L \leftarrow (SP)$, $rp_H \leftarrow (SP + 1)$, $SP \leftarrow SP + 2$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to register pair rp (XA, HL, DE, BC), and then increments the SP.

The contents of (SP) are restored to the lower register of the register pair (rp_H: A, L, E, C), and the contents of (SP + 1) are restored to the higher register (rp_H: X, H, D, B).

POP BS

Function: RBS ← (SP), MBS ← (SP + 1), SP ← SP + 2

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the register bank selection register (RBS) and the memory bank selection register (MBS), and then increments the SP.

10.4.12 INTERRUPT CONTROL INSTRUCTIONS

EI

Function: IME (IPS. 3) ← 1

Sets the interrupt master enable flag (bit 3 of the interrupt priority selection register) to "1" and enables interrupts. Whether or not interrupt acknowledgment is possible is controlled by the individual interrupt enable flags.

EI IExxx

Function: IExxx ← 1 xxx = N_5, N_2 to N_0

Sets an interrupt enable flag (IExxx) to "1", enabling interrupt acknowledgment. (xxx = BT, CSI, T0, T1, W, 0, 1, 2, 4)

DI

Function: IME (IPS. 3) ← 0

Resets the interrupt master enable flag (bit 3 of the interrupt priority selection register) to "0", disabling all interrupts irrespective of the contents of the individual interrupt enable flags.

DI IExxx

Function: IExxx ← 0 xxx = N_5, N_2 to N_0

Resets an interrupt enable flag (IExxx) to "0", disabling interrupt acknowledgment. (xxx = BT, CSI, T0, T1, W, 0, 1, 2, 4)

10.4.13 INPUT/OUTPUT INSTRUCTIONS

IN A, PORTn

Function: A ← PORTn n = N_3 to N_0: 0 to 8

Transfers the contents of the port specified by PORTn (n = 0 to 8) to the A register.

Note:

When this instruction is executed, it is necessary for MBE = 0 or (MBE = 1, MBS = 15) to be set. A value between 0 and 8 can be specified for n.

Output latch data (output mode) or pin data (input mode) is fetched according to the input/output mode specification.

IN XA, PORTn

Function: A ← PORTn, X ← PORTn + 1 n = N_3 to N_0: 4, 6

Transfers the contents of the port specified by PORTn (n = 4 or 6) to the A register, and transfers the contents of the next port to the X register.

Note:

Only 4 or 6 can be specified for n. When this instruction is executed, it is necessary for MBE = 0 or (MBE = 1, MBS = 15) to be set.

Output latch data (output mode) or pin data (input mode) is fetched according to the input/output mode specification.

OUT PORTn, A

Function: PORTn ← A n = N_3 to N_0: 2 to 8

Transfers the contents of the A register to the output latch of the port specified by PORTn (n = 2 to 8) to the A register.

Note:

When this instruction is executed, it is necessary for MBE = 0 or (MBE = 1, MBS = 15) to be set. A value between 2 and 8 can be specified for n.

OUT PORTn, XA

Function: PORTn ← A, PORTn + 1 ← X n = N_3 to N_0: 4, 6

Transfers the contents of the A register to the output latch of the port specified by PORTn (n = 4 or 6) to the A register, and transfers the contents of the X register to the output latch of the next port.

Note:

When this instruction is executed, it is necessary for MBE = 0 or (MBE = 1, MBS = 15) to be set.

Only 4 or 6 can be specified for n.

10.4.14 CPU CONTROL INSTRUCTIONS

HALT

Function: PCC. 2 ← 1

The instruction which sets the HALT mode (sets bit 2 of the processor clock control register).

Note:

The instruction following the HALT instruction must be an NOP instruction.

STOP

Function: PCC. 3 ← 1

The instruction which sets the STOP mode (sets bit 3 of the processor clock control register).

Note:

The instruction following the STOP instruction must be an NOP instruction.

NOP

Function:

Expends one machine cycle without performing any operation.

10.4.15 SPECIAL INSTRUCTIONS

SEL RBn

Function: RBS ← n n = N_1 and N_0: 0 to 3

Places the 2-bit immediate data n in the register bank selection register (RBS).

SEL MBn

Function: MBS ← n
n = N_3 to N_0: 0, 1, 2, 15

Transfers the 4-bit immediate data n to the memory bank selection register (MBS).

GETI taddr

Function: taddr = T_5 to T_0, 0: 20 H to 7FH

- When a table defined by the TBR instruction referenced
  PC_{13} to PC_0 ←
  $(taddr)_5$ to $(taddr)_0$ + (taddr + 1)
- When a table defined by the TCALL instruction is referenced
  (SP - 1) ← PC_7 to PC_4, (SP - 2) ← PC_3 to PC_0
  (SP - 3) ← MBE, RBE, PC_{13}, PC_{12}
  (SP - 4) ← PC_{11} to PC_8
  PC_{13} to PC_0 ←
  $(taddr)_5$ to $(taddr)_0$ + (taddr + 1)
  SP ← SP - 4
- When a table defined by other than the TBR or TCALL instruction is referenced
  Execute the instruction that has (taddr) (taddr + 1) as the operation code.

This instruction references the 2-byte data in the program memory addresses specified by (taddr) and (taddr + 1), and executes that data as an instruction.

The reference table area is 0020H to 007FH, and data is written in this area in advance. When writing the data, in the case of a 1-byte or 2-byte instruction the mnemonic is written as it is. In the case of a 3-byte call instruction or a 3-byte branch instruction, it is written by means of an assembler pseudo-instruction (TCALL, TBR).

Only an even address can be specified for taddr.

Note:

Two-byte instructions which can be placed in the reference table are restricted to 2-machine-cycle instructions (with the exception of the BRCB and CALLF instructions). When two 1-byte instructions are placed in the table, only the combinations shown below are permitted.

1st Byte Instruction	2nd Byte Instruction

MOV A, @HL
MOV @HL, A
XCH A, @HL
	INCS L
	DECS L
	INCS H
	DECS H
	INCS HL

MOV A, @DE XCH A, @DE	INCS E DECS E INCS D DECS D INCS DE
MOV A, @DL XCH A, @DL	INCS L DECS L INCS D DECS D

As the PC is not incremented during execution of a GETI instruction, after execution of the referenced instruction processing continues from the address following the GETI instruction.

If the instruction before the GETI instruction has a skip function, the GETI instruction is skipped in the same way as any other 1-byte instruction. Also, if the instruction referenced by the GETI instruction has a skip function, the instruction after the GETI instruction is skipped.

When an instruction with a stacking effect is referenced by the GETI instruction, it is executed as follows:

- If the instruction before the GETI instruction is also a stacking-effect instruction of the same group, when the GETI instruction is executed the stacking effect disappears and the referenced instruction is not skipped.

- If the instruction after the GETI instruction is also a stacking-effect instruction of the same group, the stacking effect produced by the referenced instruction is valid and the next instruction is skipped.

Example:

To replace the following with GETI instructions:

```
MOV   HL, #00H
MOV   XA, #FFH
CALL  SUB1
BR    SUB2
```

```
        ORG    20H
HL00 :  MOV    HL, #00H
XAFF :  MOV    XA, #FFH
CSUB1:  TCALL  SUB1
BSUB2:  TBR    SUB2
     :
     :
     :
     GETI  HL00 ; MOV   HL, #00H
     :
     :
     :
     GETI  BSUB2; BR    SUB2
     :
     :
     :
     GETI  CSUB1; CALL  SUB1
     :
     :
     :
     GETI  XAFF ; MOV   XA, #FFH
```

APPENDIX A. FUNCTIONAL COMPARISON OF uPD75336, uPD75P336 AND uPD75328

A.1 FUNCTIONAL DIFFERENCES

Product Name		uPD75336	uPD75P336	uPD75328
CPU core		75X-High End		75X-Standard
ROM (bytes)		16256		8064
RAM (x 4 bits)		768		512
General registers		4 bits x 8 x 4 banks		4 bits x 8 x 1 bank
Instruction cycle	Main system clock	0.95 us, 1.91 us, 3.81 us, 15.3 us (at 4.19 MHz operation)		0.95 us, 1.91 us 15.3 us (at 4.19 MHz operation)
	Subsystem clock	122 us (at 32.768 kHz operation)		
A/D converter		. 8-ch x 8-bit resolution (successive approximation type) . Low-voltage operation capability: V_{DD} = 2.7 to 6.0 V		. 6-ch x 8-bit resolution (successive approximation type) . Low-voltage operation capability: V_{DD} = 3.5 to 6.0 V
Timer/counter		. Basic interval timer x 1 . Timer/event counter x 2 . Watch timer x 1		. Basic interval timer x 1 . Timer/event counter x 1 . Watch timer x 1
Vectored interrupts		. External: 3 . Internal: 4		. External: 3 . Internal: 3
Test inputs		. External: 1 . Internal: 1		. External: 1 . Internal: 1

(to be continued)

(cont'd)

Product Name	uPD75336	uPD75P336	uPD75328
Buzzer output (BUZ)	2 kHz, 4 kHz, 32 kHz		2 kHz
8-bit data processing*	Transfer, addition/subtraction, increment/decrement, compare		Transfer
On-chip PROM product	uPD75P336	—	uPD75P328
Package	. 80 pin plastic QFP (□ 14 mm) . 80 pin plastic TQFP (□ 12 mm)		. 80 pin plastic QFP (□ 14 mm)

*: See A.2 "Differences between uPD75336 and uPD75328 Instructions" for details.

A.2 DIFFERENCES BETWEEN uPD75336 AND uPD75328 INSTRUCTIONS

uPD75336 instructions include the following in addition to those of the uPD75328.

Current users of the uPD75328 should refer to this table.

Transfer	MOV	A, @HL+
		A, @HL−
	XCH	A, @HL+
		A, @HL−
Bit transfer	MOV1	CY, fmem. bit
		CY, pmem. @L
		CY, @H+mem. bit
		fmem. bit, CY
		pmem. @L, CY
		@H+mem. bit, CY
NOTE 1	INCS	rp1
	DECS	rp'
NOTE 2	SKE	XA, @HL
		XA, rp'
NOTE 3	SEL	RBn

NOTE 1 : Increment/decrement
2 : Comparison
3 : Special

Branch	BR	PCDE
		PCXA
Operation	ADDS	XA, #n8
		XA, rp'
		rp' 1, XA
	ADDC	XA, rp'
		rp'1, XA
	SUBS	XA, rp'
		rp'1, XA
	SUBC	XA, rp'
		rp'1, XA
	AND	XA, rp'
		rp'1, XA
	OR	XA, rp'
		rp'1, XA
	XOR	XA, rp'
		rp'1, XA

APPENDIX B. DEVELOPMENT TOOLS

The following development tools are available for system development using the uPD75336.

Language Processor

RA75X relocatable assembler	Host Machine	OS	Supply Medium	Ordering Code (Product Name)
	PC-9800 series	MS-DOS™ [Ver. 3.30 to Ver. 5.00A*]	3.5-inch 2HD	uS5A13RA75X
			5-inch 2HD	uS5A10RA75X
	IBM PC/AT™	PC DOS™ (Ver. 3.1)	5-inch 2HC	uS7B10RA75X

*: A task swapping function is provided in Ver. 5.00/5.00A, but cannot be used with this software.

PROM Writing Tools

<table>
<tr><td rowspan="3">Hardware</td><td>PG-1500</td><td colspan="4">This PROM programmer allows easy programming via the keyboard or remote control of a typical PROM of 256K bits to 4M bits or a single-chip microcomputer with on-chip PROM by connecting the board provided to a separately available programmer adapter.</td></tr>
<tr><td>PA-75P328GC</td><td colspan="4">PROM programmer adapter for uPD75P336GC, used connected to the PG-1500.</td></tr>
<tr><td>PA-75P336GK</td><td colspan="4">PROM programmer adapter for uPD75P336GK, used connected to the PG-1500.</td></tr>
<tr><td rowspan="4">Software</td><td rowspan="4">PG-1500 controller</td><td colspan="4">Connects the PG-1500 and host machine via a serial and parallel interface, and controls the PG-1500 on the host machine.</td></tr>
<tr><td>Host Machine</td><td>OS</td><td>Supply Medium</td><td>Ordering Code (Product Name)</td></tr>
<tr><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS [Ver. 3.30 to Ver. 5.00A*]</td><td>3.5-inch 2HD</td><td>uS5A13PG1500</td></tr>
<tr><td>5-inch 2HD</td><td>uS5A10PG1500</td></tr>
<tr><td>IBM PC/AT</td><td>PC DOS (Ver. 3.1)</td><td>5-inch 2HC</td><td>uS7B10PG1500</td></tr>
</table>

*: A task swapping function is provided in Ver. 5.00/5.00A, but cannot be used with this software.

Remarks : The operation of RA75X relocatable assembler and PG-1500 controller is available only on the host machine and operation system described above.

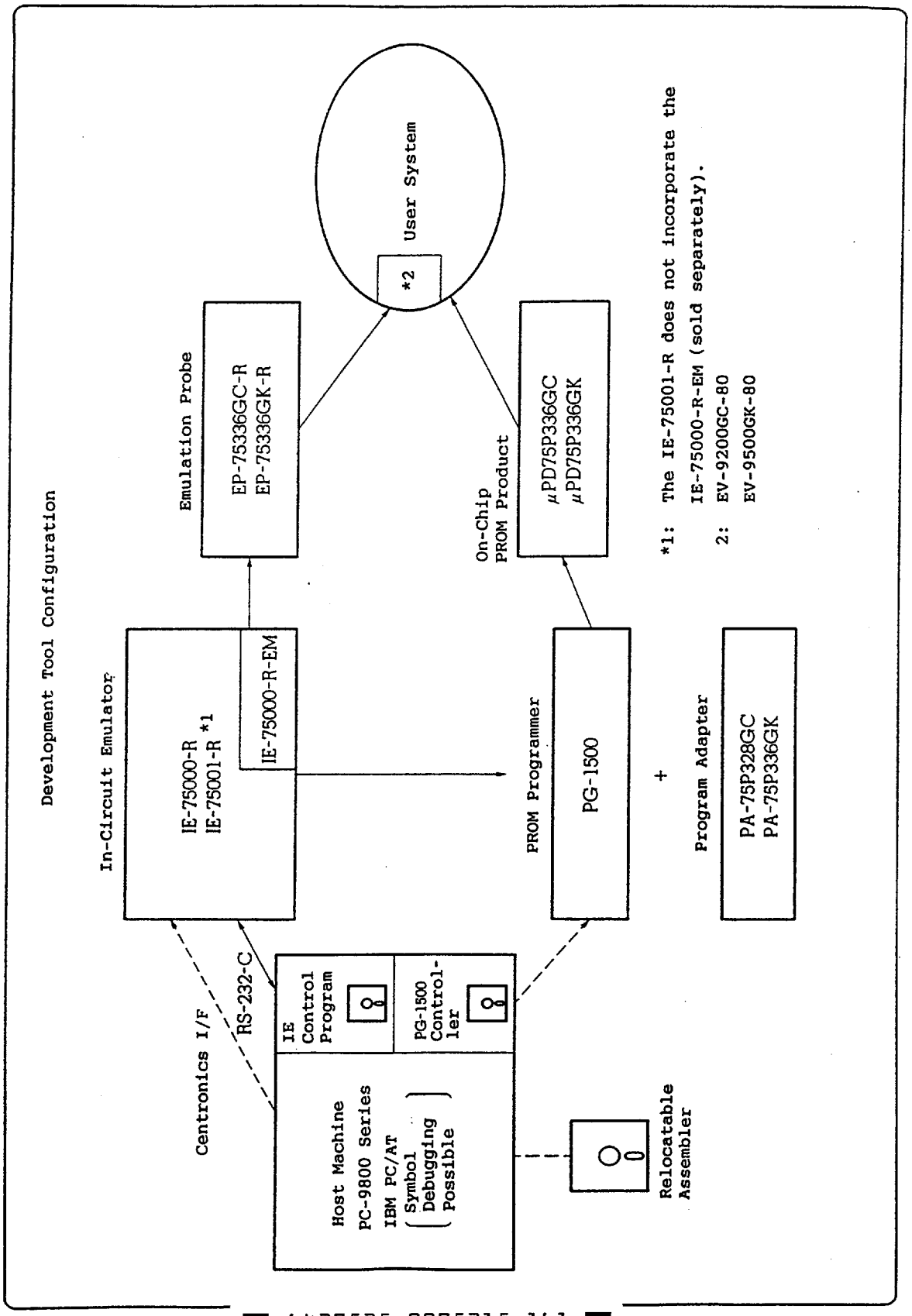
Debugging Tools

<table>
<tr><td rowspan="6">Hardware</td><td>IE-75000-R
*1</td><td colspan="3">The IE-75000-R is an in-circuit emulator for performing hardware and software debugging in the development of application systems using the 75X series. The IE-75000-R is used in combination with an emulation probe. Efficient debugging is possible by connecting to a host machine and PROM programmer.</td></tr>
<tr><td>IE-75000-R-EM</td><td colspan="3">Emulation board for evaluation of application systems using the 75X series. The IE-75000-R-EM is used in combination with the IE-75000-R or IE-75001-R. It is incorporated in the IE-75000-R.</td></tr>
<tr><td>IE-75001-R</td><td colspan="3">The IE-75001-R is an in-circuit emulator for performing hardware and software debugging in the development of application systems using the 75X series. The IE-75001-R is used in combination with a separately available emulation board IE-75000-R-EM and the emulation probe. Efficient debugging is possible by connecting to a host machine and PROM programmer.</td></tr>
<tr><td>EP-75336GC-R
EV-9200GC-80</td><td colspan="3">Emulation probe for the uPD75336GC. Used in conjunction with the IE-75000-R, or the IE-75001-R and the IE-75000-R-EM. The 80-pin conversion socket EV-9200GC-80 for easy connection with the user system is provided.</td></tr>
<tr><td>EP-75336GK-R
EV-9500GK-80</td><td colspan="3">Emulation probe for the uPD75336GK. Used in conjunction with the IE-75000-R, or the IE-75001-R and the IE-75000-R-EM. The 80-pin conversion adapter EV-9500GK-80 for easy connection with the user system is provided.</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Software</td><td>IE control program</td><td colspan="3">Connects the IE-75000-R and the IE-75001-R to the host machine via RS-232-C and Centronics I/F, and controls the IE-75000-R and the IE-75001-R on the host machine.</td></tr>
<tr><td colspan="2"></td><td>Host Machine</td><td>OS</td><td>Supply Medium</td><td>Ordering Code (Product Name)</td></tr>
<tr><td colspan="2"></td><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS [Ver.3.30 to Ver.5.00A*2]</td><td>3.5-inch 2HD</td><td>uS5A13IE75X</td></tr>
<tr><td colspan="2"></td><td>5-inch 2HD</td><td>uS5A10IE75X</td></tr>
<tr><td colspan="2"></td><td>IBM PC series</td><td>PC DOS (Ver.3.1)</td><td>5-inch 2HC</td><td>uS7B10IE75X</td></tr>
</table>

*1: Maintenance product

2: A task swapping function is provided in Ver. 5.00/5.00A, but cannot be used with this software.

Remarks: The operation of IE control program is available only on the host machine and operation system described above.



APPENDIX C. MASK ROM ORDERING PROCEDURE

After completion of the uPD75336 program, the following procedure should be used to order the mask ROM.

① Mask ROM order reservation

Inform an NEC agent or NEC sales department of your intention to make a mask ROM order.

② Order medium creation

The medium for a mask ROM order is UVEPROM or an 8-inch IBM-format floppy disk. When ordering using UVEPROM, 3 UVEPROMs with the same contents should be prepared (for a product with a mask option, the mask option data should be submitted on a mask option information sheet).

③ Required documents

The following documents need to be completed when ordering mask ROM.

A. Mask ROM order form

B. Mask ROM order checksheet

C. Mask option information sheet (only required for a product with a mask option)

④ Ordering

The medium described in ② and the documents listed in ③ above should be submitted to an NEC agent or NEC sales department by the order reservation date.

NOTE: For details, please refer to Information Document "ROM Code Ordering Procedure" (Document No.: IEM-834).

APPENDIX D. INSTRUCTION INDEX

D.1 INSTRUCTION INDEX (IN FUNCTIONAL ORDER)

(to be continued)

(cont'd)

D.2 INSTRUCTION INDEX (IN ALPHABETICAL ORDER)

(to be continued)

(cont'd)

Instruction		Page	Instruction		Page
RET		10-61	SKTCLR	fmem.bit	10-54
RETI		10-62	SKTCLR	pmem.@L	10-54
RETS		10-62	SKTCLR	@H + mem.bit	10-54
RORC	A	10-46	STOP		10-68
SEL	MBn	10-69	SUBC	A, @HL	10-41
SEL	RBn	10-69	SUBC	rp'1, XA	10-42
SET1	CY	10-51	SUBC	XA, rp'	10-41
SET1	fmem.bit	10-52	SUBS	A, @HL	10-39
SET1	mem.bit	10-52	SUBS	rp'1, XA	10-40
SET1	pmem.@L	10-52	SUBS	XA, rp'	10-40
SET1	@H + mem.bit	10-52	TBR	addr	10-59
SKE	A, reg	10-50	TCALL	!addr	10-61
SKE	A, @HL	10-49	XCH	A, @rpa	10-29
SKE	reg, #n4	10-49	XCH	A, mem	10-30
SKE	XA, rp'	10-50	XCH	A, reg1	10-30
SKE	XA, @HL	10-49	XCH	XA, @HL	10-30
SKE	@HL, #n4	10-49	XCH	XA, mem	10-30
SKF	fmem.bit	10-54	XCH	XA, rp'	10-31
SKF	mem.bit	10-54	XOR	A, #n4	10-45
SKF	pmem.@L	10-54	XOR	A, @HL	10-45
SKF	@H + mem.bit	10-54	XOR	rp'1, XA	10-46
SKT	CY	10-51	XOR	XA, rp'	10-45
SKT	fmem.bit	10-53	XOR1	CY, fmem.bit	10-55
SKT	mem.bit	10-53	XOR1	CY, pmem.@L	10-55
SKT	pmem.@L	10-53	XOR1	CY, @H + mem.bit	10-55
SKT	@H + mem.bit	10-53			

APPENDIX E. HARDWARE INDEX (ALPHABETICAL ORDER)

Hardware Index