To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

USER'S MANUAL

RENESAS

Phase-out/Discontinued

μPD75308

4-BIT SINGLE-CHIP MICROCONTROLLER

μPD75312B
μPD75316
μPD75316B
μPD75P308
μPD75P316
μPD75P316A
μPD75P316B
μPD75304
μPD75304B
μPD75306
μPD75306B
μPD75308
μPD75308B
μPD75312

Document No. U11023EJCV0UM00 (12th edition)
(Previous No. IEU-1263)
Date Published April 1996 P
Printed in Japan

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

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

License needed	:	μPD75P308K	μPD75P316AK	μPD75P316BKK-T

The customer must judge the need for license	:	μPD75304GF-××× -3B9	μPD75P308GF-3B9
		μPD75304BGC-×××-3B9	μPD75312GF-×××-3B9
		μPD75304BGF-×××-3B9	μPD75312BGC-×××-3B9
		μPD75304BGK-×××-BE9	μPD75312BGK-×××-BE9
		μPD75306GF-×××-3B9	μPD75316GF-×××-3B9
		μPD75306BGC-×××-3B9	μPD75316BGC-×××-3B9
		μPD75306BGF-×××-3B9	μPD75316BGK-×××-BE9
		μPD75306BGK-×××-BE9	μPD75P316GF-3B9
		μPD75308GF-×××-3B9	μPD75P316AGF-3B9
		μPD75308BGC-×××-3B9	μPD75P316BGC-3B9
		μPD75308BGF-×××-3B9	μPD75P316BGK-BE9
		μPD75308BGK-×××-BE9	

M7 94.11

MAJOR REVISIONS IN THIS VERSION

Section	Description
Whole manual	μPD75P316B has been developed.
p.203	The note in **Figure 5-83. Examples of LCD Drive Power Supply Connection (With External Split Resistor)** has been modified and a caution has been added to this figure.
p.235	The example in **6.6 Vector Address Sharing Interrupt Servicing** has been modified.
p.313	**APPENDIX A DEVELOPMENT TOOLS** The version of the supported OS has been up-graded.
p.325	**APPENDIX E REVISION HISTORY** has been added.

The mark ★ shows major revised points.

PREFACE

Intended Readership This manual describes the functions of the μPD7530x/7531x products, and is intended for user's technical personnel involved in the design of application systems using them.

Purpose The purpose of this manual is to explain to the user the hardware functions of the μPD75304, 75304B, 75306, 75306B, 75308, 75308B, 75P308, 75312, 75312B, 75316, 75316B, 75P316, 75P316A and 75P316B following the organization shown below.

Organization This manual is broadly organized into the following sections:

- General description
- Pin functions
- Architectural features and memory mapping
- Internal CPU functions
- Peripheral hardware functions
- Interrupt functions
- Standby functions
- Reset functions
- PROM write and verify
- Instruction set

Using this Manual Readers of this manual need to have a general understanding of electricity, logical circuits and microcontrollers.

- **Users employing this manual for the μPD7530x/7531x products**
  → Insofar as there are no particular functional differences between the various products, the μPD75308 is taken as the representative device for the descriptions in this manual. After checking the functional differences described in 1.3 "List of Functions of Series Products", "μPD75308" can be read as the respective product name when using the manual.
- **Users employing this manual for the μPD75P308K, μPD75P316AK or μPD75P316BKK-T**
  → In those sections of this manual which apply to both PROM and EPROM products, the term "PROM" is used to represent both. When using this manual for an EPROM product, "PROM" can be read as "EPROM".
- **Users who has previous experience of operating the μPD75304, 75306, 75308 and 75P308**
  → Read 1.3 "List of Functions of Series Products" to check for differences in functions between products.
- **When checking the instruction functions from mnemonic**
  → Refer to **APPENDIX C INSTRUCTION INDEX**.
- **When checking the function of the particular on-chip circuit**
  → Refer to **APPENDIX D HARDWARE INDEX**.
- **For a general understanding of the μPD7530x/7531x products.**
  → First read 1.1 Function Outline to get an understanding of the main functions, then read the manual according to the Contents.

- **For the electrical specifications of μPD7530x/7531x products**
  → Refer to the separate Data Sheet.
- **For application example of various functions of μPD7530x/7531x products**
  → Refer to the separate Application Note.

Explanatory Notes

Data notation	:	The most significant digit on left, the least significant digit on right
Active low notation	:	$\overline{\text{xxx}}$ (line above pin or signal name)
Memory map addresses	:	High-order address below, low-order address above
Note	:	Description of**Note** included in the text
Caution	:	Statement drawing particular attention
Remark	:	Supplementary description of text
Important item, emphasis	:	Boldface notation
Numeric notation	:	Binary number .... xxxx or xxxxB Decimal number .... xxxx Hexadecimal number .... xxxxH

Relevant Documents

■ Documents related to μPD75308

Document No. / Product	Brochure	Data Sheet	User's Manual	Instruction List	Application Note (Fundamental)	Application Note (SBI Application)
μPD75304	—	IC-2523	U11023E (This manual)	—	IEM-1239	IEM-1245
μPD75306						
μPD75308						
μPD75P308		IC-2472				
μPD75312		IC-2477				
μPD75316						
μPD75P316		IC-2651				
μPD75P316A	—	IC-2524				
μPD75304B	—	IC-2913				
μPD75306B						
μPD75308B						
μPD75312B		IC-3196				
μPD75316B						
μPD75P316B		IC-3189				

■ Documents related to whole 75X series

Data book	75X series vol.1 4-bit single-chip microcontroller	—
	75X series vol.2 4-bit single-chip microcontroller	—
Selection guide	75X series	IF-1027

■ **Documents related to development tool**

Document Name			Document Number
Hardware	IE-75000-R/IE-75001-R User's manual		EEU-1416
	IE-75000-R-EM User's manual		EEU-1294
	EP-75308GF-R User's manual		EEU-1301
	EP-75308BGC-R User's manual		EEU-1406
	EP-75308BGK-R User's manual		EEU-1408
	PG-1500 User's manual		EEU-1335
Software	RA75X assembler package user's manual	Operation	EEU-1346
		Language	EEU-1364
	PG-1500 controller user's manual	PC-9800 series (MS-DOS™) base	EEU-1291
		IBM PC series (PC DOS™) base	U10540E

Remark IE control programs are explained in the IE-75000-R/IE-75001-R User's Manual.

[MEMO]

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 GENERAL DESCRIPTION

The μPD75304, 75304B, 75306, 75306B, 75308, 75308B, 75P308, 75312, 75312B, 75316, 75316B, 75P316, 75P316A and 75P316B are 75X series products. They are 4-bit single-chip microcontrollers having an on-chip programmable LCD controller/driver and an on-chip NEC standard serial bus interface, with the features of high-speed and high-performance.

The features are described below.

- ROM capacity : Max. 16256 words x 8 bits
- RAM capacity : Max. 1024 words x 4 bits
- General register : Max. 8 units x 4 bits
- High-speed operation : Min. instruction execution time 0.95 μs
- Six interrupt sources and efficient interrupt servicing
- Efficient instruction system capable of operating 1/4/8-bit data
- Versatile timer functions : 3 channels
- Ultra low power clock operations in the standby mode (with an on-chip subsystem clock operating at power consumption level)

Products listed in Table 1-1 are available depending on the program memory sizes and types.

Table 1-1. Features of 75X Series Products

Product Name	Program Memory (ROM)	Remarks
μPD75304	4096 words x 8 bits	Mask ROM
μPD75304B	4096 words x 8 bits	Mask ROM, low-voltage operation capability
μPD75306	6016 words x 8 bits	Mask ROM
μPD75306B	6016 words x 8 bits	Mask ROM, low-voltage operation capability
μPD75308	8064 words x 8 bits	Mask ROM
μPD75308B	8064 words x 8 bits	Mask ROM, low-voltage operation capability
μPD75P308	8064 words x 8 bits	One-time PROM/EPROM
μPD75312	12160 words x 8 bits	Mask ROM
μPD75312B	12160 words x 8 bits	Mask ROM, low-voltage operation capability
μPD75316	16256 words x 8 bits	Mask ROM
μPD75316B	16256 words x 8 bits	Mask ROM, low-voltage operation capability
μPD75P316	16256 words x 8 bits	One-time PROM
μPD75P316A	16256 words x 8 bits	One-time PROM/EPROM
μPD75P316B	16256 words x 8 bits	One-time PROM/EPROM, low-voltage operation capability

The μPD75P308, 75P316A and 75P316B are products having an on-chip, one-time PROM or EPROM instead of a mask ROM. The packages are provided as follows:

One-time programmable : μPD75P308GF-3B9 (without a window)
μPD75P316AGF-3B9 (without a window)
μPD75P316BGC-3B9 (without a window)
μPD75P316BGK-BE9 (without a window)

Reprogrammable : μPD75P308K (with a window)
μPD75P316AK (with a window)
μPD75P316BKK-T (with a window)

The μPD75P316 is a product having an on-chip, one-time PROM instead of a mask ROM. The package is provided as follows:

One-time programmable : μPD75P316GF-3B9 (without a window)

Applications

- Electric home appliances VCR, audio equipment (such as CD players), etc.
- Others Telephones, cameras, sphygmomanometers, etc.

1.1 Function Outline

Function outline (1/2)

<table>
<tr><th colspan="2">Item</th><th colspan="4">Function</th></tr>
<tr><td colspan="2">Basic instructions</td><td colspan="4">41</td></tr>
<tr><td colspan="2">Instruction cycle</td><td colspan="4">0.95 μs, 1.91 μs, 15.3 μs (operating on 4.19 MHz main system clock), 122 μs (operating on 32.768 kHz subsystem clock)</td></tr>
<tr><td rowspan="2">On-chip memory</td><td>ROM</td><td colspan="4">• μPD75304, 75304B : 4096 words × 8 bits (ROM)
• μPD75306, 75306B : 6016 words × 8 bits (ROM)
• μPD75308, 75308B : 8064 words × 8 bits (ROM)
• μPD75P308 : 8064 words × 8 bits (PROM)
• μPD75312, 75312B : 12160 words × 8 bits (ROM)
• μPD75316, 75316B : 16256 words × 8 bits (ROM)
• μPD75P316, 75P316A, 75P316B : 16256 words × 8 bits (PROM)</td></tr>
<tr><td>RAM</td><td colspan="4">512 x 4 bits
1024 x 4 bits (μPD75312B, 75316B, 75P316A, 75P316B only)</td></tr>
<tr><td colspan="2">General registers</td><td colspan="4">• 4-bit manipulation : 8 units (X, A, B, C, D, E, H, L)
• 8-bit manipulation : 4 units (XA, BC, DE, HL)</td></tr>
<tr><td colspan="2">Accumulators</td><td colspan="4">• Bit accumulator (CY)
• 4-bit accumulator (A)
• 8-bit accumulator (XA)</td></tr>
<tr><td colspan="2">Instruction set</td><td colspan="4">• Many bit manipulation instructions
• Efficient 4-bit data operation instructions
• 8-bit data transfer instructions
• GETI instruction capable of implementing any 2-byte/3-byte instruction in one byte</td></tr>
<tr><td colspan="2" rowspan="4">I/O lines</td><td rowspan="4">40</td><td>8</td><td>CMOS input</td><td rowspan="2">Software pull-up capability: 23</td></tr>
<tr><td>16</td><td>CMOS input/output</td></tr>
<tr><td>8</td><td>CMOS output</td><td>Dual-purpose as segment pins</td></tr>
<tr><td>8</td><td>N-ch open-drain input/output</td><td>10 V withstand voltage, mask option pull-up capability[Note]1: 8</td></tr>
<tr><td colspan="2">LCD controller/driver</td><td colspan="4">• Segment number selection : 24, 28 and 32 segments (4 and 8 segments switchable to bit port output)
• Display mode selection : Static, 1/2 duty, 1/3 duty (1/2 bias), 1/3 duty (1/3 bias), 1/4 duty
• LCD drive division resistor incorporation capability by mask option</td></tr>
<tr><td colspan="2">Supply voltage range</td><td colspan="4">V_{DD} = 2.0 to 6.0 V : μPD75304B, 75306B, 75308B, 75312B, 75316B, 75P316B
V_{DD} = 2.7 to 6.0 V : μPD75304, 75306, 75308, 75312, 75316, 75P316A
V_{DD} = 5 V ± 5 % : μPD75P308, 75P316</td></tr>
</table>

Note Pull-up resistors by mask option are not provided for μPD75P308, 75P316, 75P316A, and 75P316B.

Phase-out/Discontinued

Function outline (2/2)

Item	Function	
Timer	3-channel	8-bit timer/event counter • Clock sources : 4 levels • Event count capability
		8-bit basic interval timer • Reference time generation : 1.95 ms, 7.82 ms, 31.3 ms, 250 ms (operating at 4.19 MHz) • Watchdog timer application capability
		Watch timer • 0.5 sec. time interval generation • Count clock source : Main system clock/subsystem clock switchable • Clock fast forward mode (3.9 ms time interval generation) • Buzzer output capability (2 kHz)
8-bit serial interface	Applicable to three modes • 3-wire serial I/O mode • 2-wire serial I/O mode • SBI mode	
	• LSB first/MSB first switchable	
Bit sequential buffer	Special bit manipulation memory: 16 bits • Ideal for remote control applications	
Clock output functions	Timer/event counter output (PTO0): Selected frequency square wave output	
	Clock output (PCL): Φ, 524 kHz, 262 kHz, 65.5 kHz (operating at 4.19 MHz)	
	Buzzer output (BUZ): 2 kHz (operating at 4.19 MHz or 32.768 kHz)	
Vectored interrupts	• External : 3 • Internal : 3	
Test input	• External : 1 • Internal : 1	
System clock oscillation circuits	• Ceramic/crystal oscillator for main system clock oscillation: 4.19430 MHz • Crystal oscillator for subsystem clock oscillation: 32.768 kHz	
Standby	STOP/HALT modes	
Packages	80-pin plastic QFP (14 x 14 mm) : μPD75304B, 75306B, 75308B, 75312B, 75316B, 75P316B 80-pin plastic QFP (14 x 20 mm) : μPD75304, 75304B, 75306, 75306B, 75308, 75308B, 75P308, 75312, 75316, 75P316, 75P316A 80-pin plastic QFP (12 x 12 mm) : μPD75304B, 75306B, 75308B, 75312B, 75316B, 75P316B 80-pin ceramic WQFN (LCC with window)[Note] : μPD75P308, 75P316A, 75P316B	

Note μPD75P308, 75P316A : 14 x 20 mm
μPD75P316B : 14 x 14 mm

1.2 Ordering Information

Part Number	Package	On-Chip ROM
μPD75304GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75304BGC-xxx3B9	80-pin plastic QFP (14 x 14 mm)	Mask ROM
μPD75304BGF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75304BGK-xxx-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	Mask ROM
μPD75306GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75306BGC-xxx-3B9	80-pin plastic QFP (14 x 14 mm)	Mask ROM
μPD75306BGF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75306BGK-xxx-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	Mask ROM
μPD75308GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75308BGC-xxx-3B9	80-pin plastic QFP (14 x 14 mm)	Mask ROM
μPD75308BGF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75308BGK-xxx-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	Mask ROM
μPD75P308GF-8B9	80-pin plastic QFP (14 x 20 mm)	One-time PROM
μPD75P308K	80-pin ceramic WQFN (LCC with window)	EPROM
μPD75312GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75312BGC-xxx-3B9	80-pin plastic QFP (14 x 14 mm)	Mask ROM
μPD75312BGK-xxx-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	Mask ROM
μPD75316GF-xxx-3B9	80-pin plastic QFP (14 x 20 mm)	Mask ROM
μPD75316BGC-xxx-3B9	80-pin plastic QFP (14 x 14 mm)	Mask ROM
μPD75316BGK-xxx-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	Mask ROM
μPD75P316GF-3B9	80-pin plastic QFP (14 x 20 mm)	One-time PROM
μPD75P316AGF-3B9	80-pin plastic QFP (14 x 20 mm)	One-time PROM
μPD75P316AK	80-pin ceramic WQFN (LCC with window)	EPROM
μPD75P316BGC-3B9	80-pin plastic QFP (14 x 14 mm)	One-time PROM
μPD75P316BGK-BE9	80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	One-time PROM
μPD75P316BKK-T	80-pin ceramic WQFN (LCC with window)	EPROM

Remark “xxx” indicates the ROM code number.

1.3 List of Functions of Series Products

(1) μPD75304, 75306, 75308, 75P308

Item \ Product		μPD75304	μPD75306	μPD75308	μPD75P308
Program memory		• Mask ROM • 000H to FFFH • 4096 bytes	• Mask ROM • 0000H to 177FH • 6016 bytes	• Mask ROM • 0000H to 1F7FH • 8064 bytes	• EPROM/one-time PROM • 0000H to 1F7FH • 8064 bytes
Data memory		512 x 4 bits (Banks 0 and 1: 256 x 4 bits)			
Instruction set	3-byte branch instructions	Not provided	Provided		
	Others	Commonly provided			
Program counter		12 bits	13 bits		
Mask option		• Internal pull-up resistor for ports 4 and 5 • Division resistor for LCD drive power supply			Not provided
No. 57 pin function		NC			V_{PP}
Supply voltage range		2.7 to 6.0 V			5 V ± 5 %
Package		80-pin plastic QFP (14 x 20 mm)			• 80-pin ceramic WQFN (LCC with window) • 80-pin plastic QFP (14 x 20 mm)

(2) μPD75304B, 75306B, 75308B

Item \ Product		μPD75304B	μPD75306B	μPD75308B
Program memory		• Mask ROM • 0000H to 0FFFH • 4096 bytes	• Mask ROM • 0000H to 177FH • 6016 bytes	• Mask ROM • 0000H to 1F7FH • 8064 bytes
Data memory		512 x 4 bits (Banks 0 and 1: 256 x 4 bits)		
Instruction set	3-byte branch instructions	Not provided	Provided	
	Others	Commonly provided		
Program counter		12 bits	13 bits	
Mask option		• Internal pull-up resistor for ports 4 and 5 • Division resistor for LCD drive power supply		
No. 57 pin function		NC		
Supply voltage range		2.0 to 6.0 V		
Package		• 80-pin plastic QFP (14 x 14 mm) • 80-pin plastic QFP (14 x 20 mm) • 80-pin plastic TQFP (12 x 12 mm)		

(3) μPD75312, 75316, 75P316, 75P316A

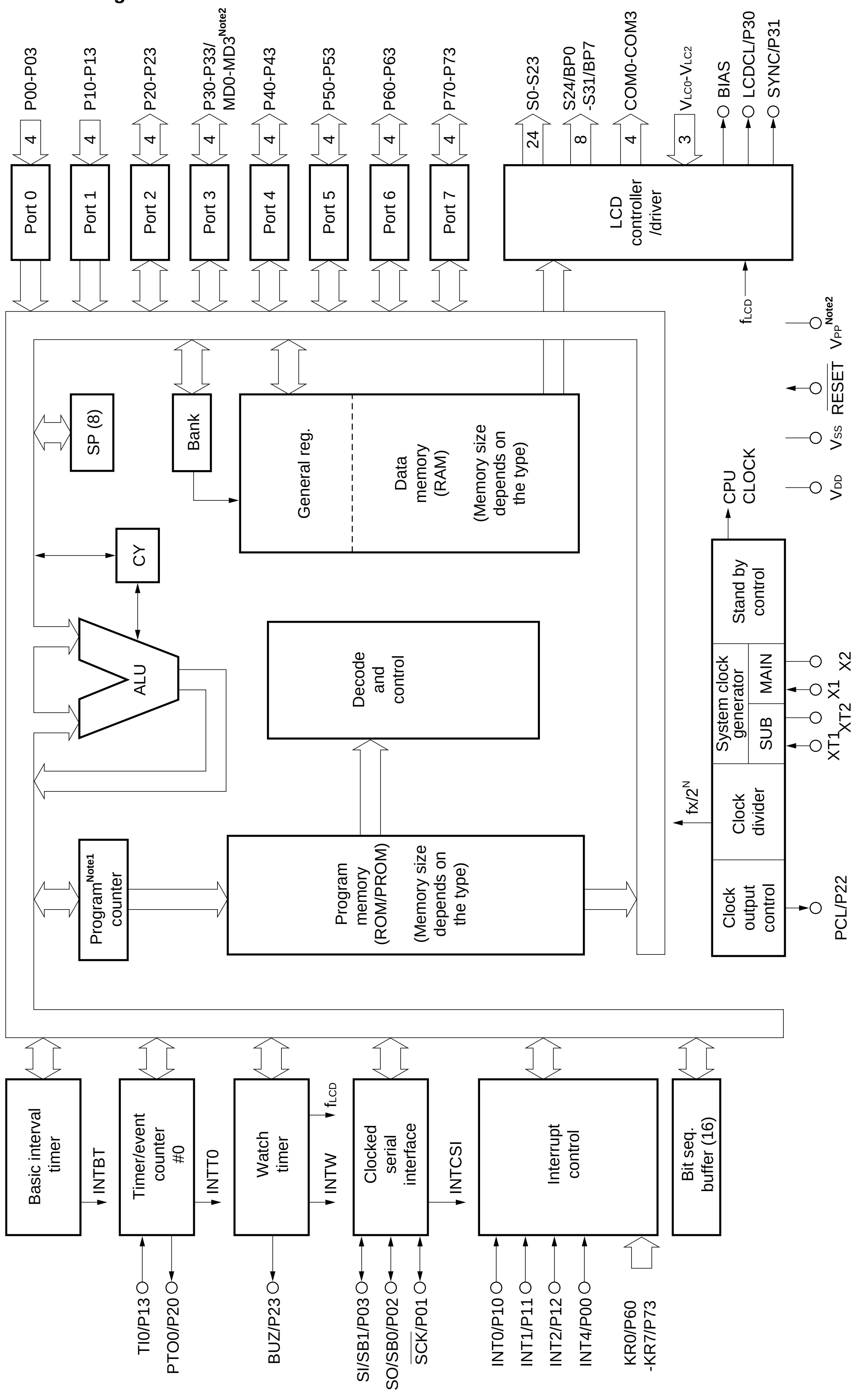
Item \ Product		μPD75312	μPD75316	μPD75P316	μPD75P316A
Program memory		• Mask ROM • 0000H to 2F7FH • 12160 bytes	• Mask ROM • 0000H to 3F7FH • 16256 bytes	• One-time PROM • 0000H to 3F7FH • 16256 bytes	• EPROM/one-time PROM • 0000H to 3F7FH • 16256 bytes
Data memory		512 x 4 bits (Banks 0 and 1: 256 x 4 bits)			1024 x 4 bits (Bank 0, 1, 2, 3: 256 x 4 bits)
Instruction set	3-byte branch instructions	Provided			
	Others	Commonly provided			
Program counter		14 bits			
Mask option		• Internal pull-up resistor for ports 4 and 5 • Division resistor for LCD drive power supply		Not provided	
No. 57 pin function		NC		V_{PP}	
Supply voltage range		2.7 to 6.0 V		5 V ± 5 %	2.7 to 6.0 V
Package		80-pin plastic QFP (14 x 20 mm)			• 80-pin ceramic WQFN (LCC with window) • 80-pin plastic QFP (14 x 20 mm)

(4) μPD75312B, 75316B, 75P316B

Item \ Product		μPD75312B	μPD75316B	μPD75P316B
Program memory		• Mask ROM • 0000H to 2F7FH • 12160 bytes	• Mask ROM • 0000H to 3F7FH • 16256 bytes	• One-time PROM • 0000H to 3F7FH • 16256 bytes
Data memory		1024 x 4 bits (Banks 0, 1, 2, 3: 256 x 4 bits)		
Instruction set	3-byte branch instructions	Provided		
	Others	Commonly provided		
Program counter		14 bits		
Mask option		• Internal pull-up resistor for ports 4 and 5 • Division resistor for LCD drive power supply		Not provided
No. 57 pin function		IC		V_{PP}
Supply voltage range		2.0 to 6.0 V		
Package		• 80-pin plastic QFP (14 x 14 mm) • 80-pin plastic TQFP (12 x 12 mm)		• 80-pin plastic QFP (14 x 14 mm) • 80-pin plastic TQFP (12 x 12 mm) • 80-pin ceramic WQFN (LCC with window)

Phase-out/Discontinued

1.4 Block Diagram



Notes 1. Bit length depends on the type.

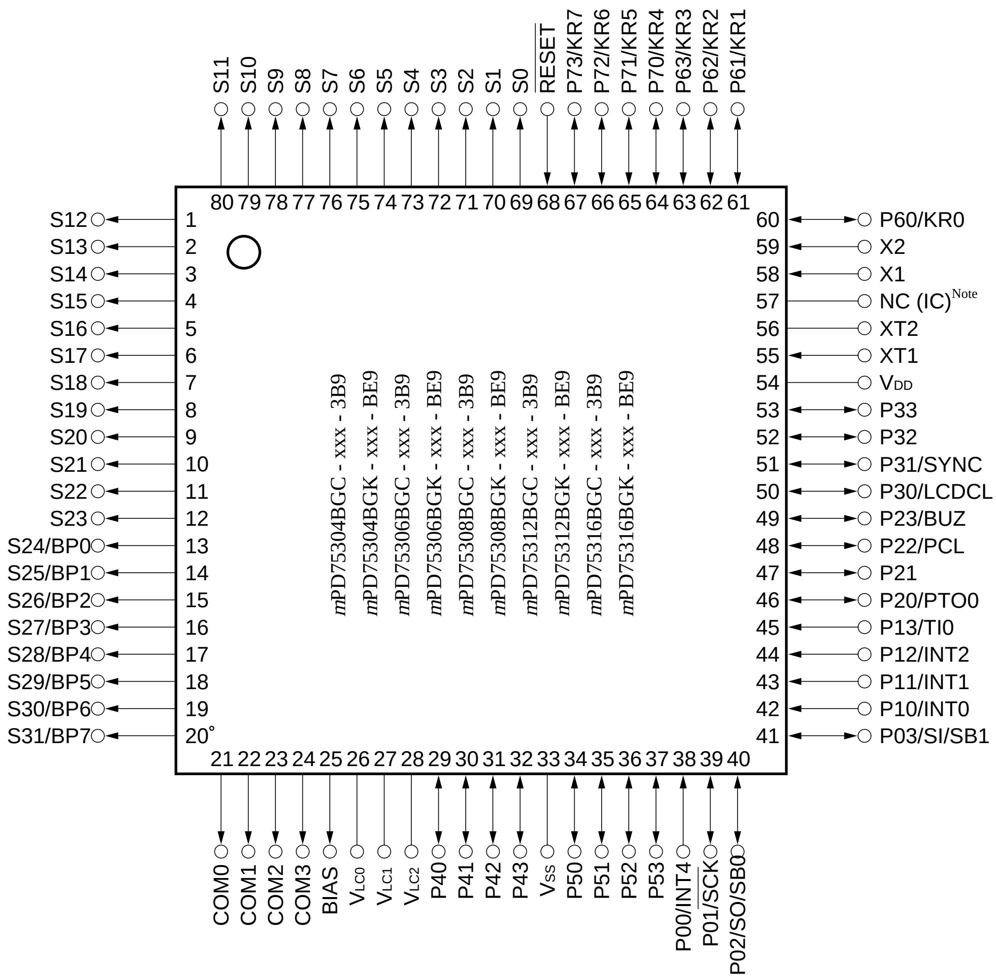
2. MD0 to MD3 and V_{PP} are used for PROM version.

Phase-out/Discontinued

1.5 Pin Configuration (Top View)

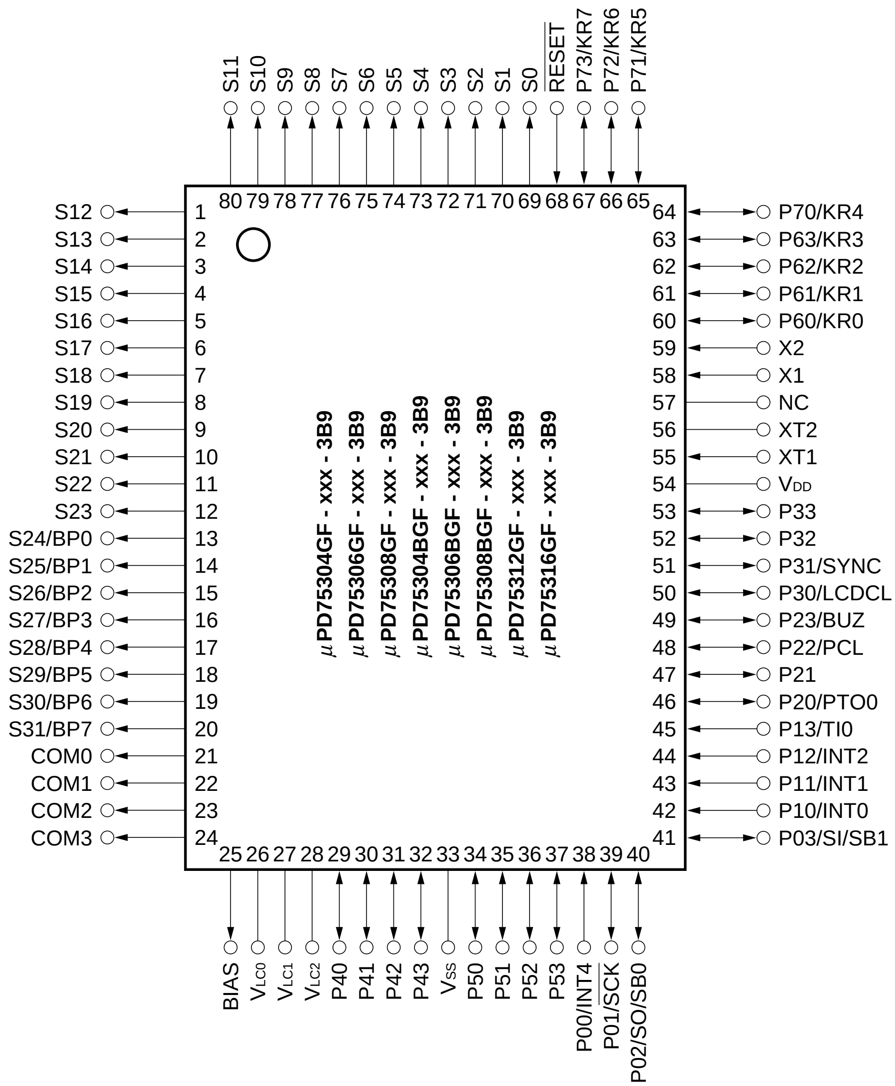
(1) Normal operating mode

(a) 80-pin plastic QFP (14 x 14 mm)
80-pin plastic TQFP (12 x 12 mm)



Note The IC pin is used for μPD75312B and 75316B.
IC pin: Internally Connected. Connect this pin with V_{DD} directly.

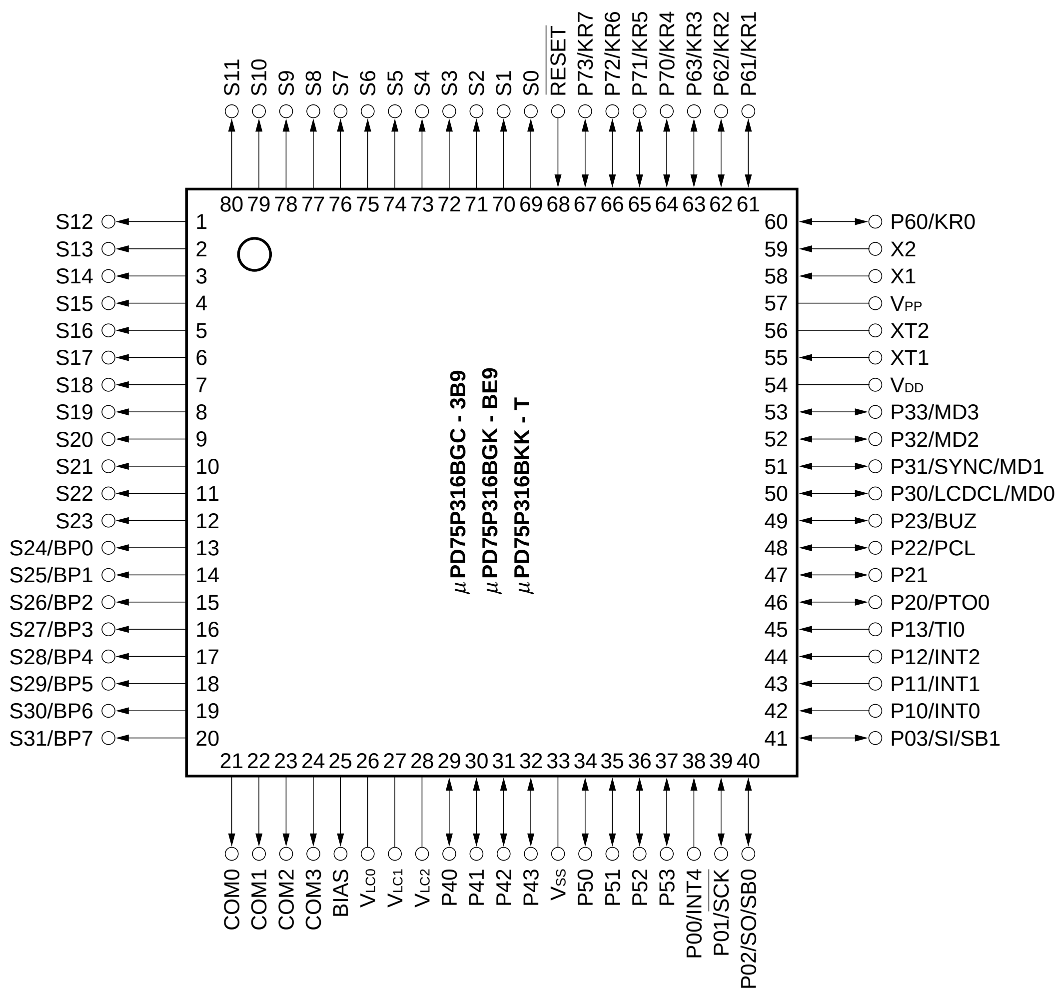
(b) 80-pin plastic QFP (14 x 20 mm)



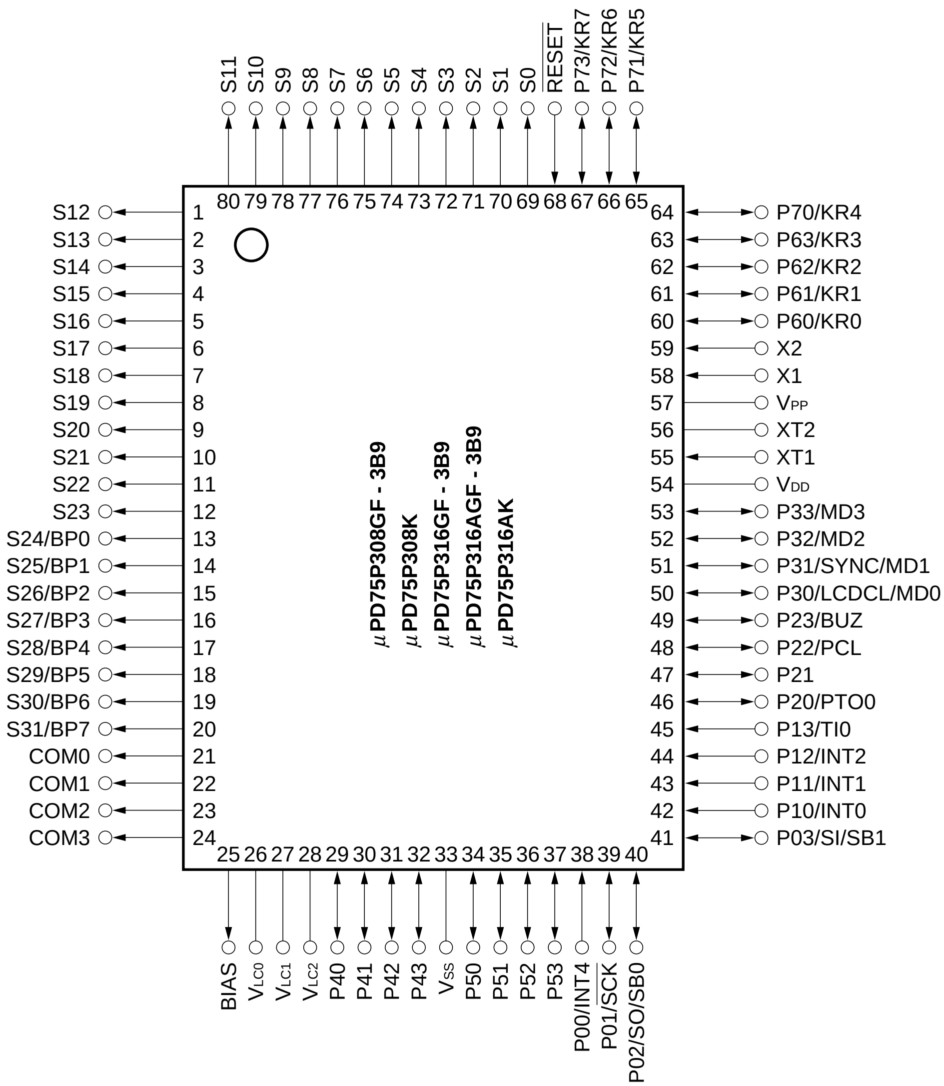
P00-03	: Port0	S0-31	: Segment Output 0-31
P10-13	: Port1	COM0-3	: Common Output 0-3
P20-23	: Port2	V_{LC0-2}	: LCD Power Supply 0-2
P30-33	: Port3	BIAS	: LCD Power Supply Bias Control
P40-43	: Port4	LCDCL	: LCD Clock
P50-53	: Port5	SYNC	: LCD Synchronization
P60-63	: Port6	TI0	: Timer Input 0
P70-73	: Port7	PTO0	: Programmable Timer Output 0
BP0-7	: Bit Port	BUZ	: Buzzer Clock
KR0-7	: Key Return	PCL	: Programmable Clock
$\overline{SCK}$	: Serial Clock	INT0, 1, 4	: External Vectored Interrupt 0, 1, 4
SI	: Serial Input	INT2	: External Test Input 2
SO	: Serial Output	X1, 2	: Main System Clock Oscillation 1, 2
SB0, 1	: Serial Bus 0,1	XT1, 2	: Subsystem Clock Oscillation 1, 2
$\overline{RESET}$	: Reset Input	NC	: No Connection

(2) PROM mode

(a) 80-pin plastic QFP (14 x 14 mm)
80-pin plastic TQFP (12 x 12 mm)
80-pin ceramic WQFN (LCC with window)



(b) 80-pin plastic QFP (14 x 20 mm)
80-pin ceramic WQFN (LCC with window)



P00-03	: Port0
P10-13	: Port1
P20-23	: Port2
P30-33	: Port3
P40-43	: Port4
P50-53	: Port5
P60-63	: Port6
P70-73	: Port7
BP0-7	: Bit Port
KR0-7	: Key Return
$\overline{\text{SCK}}$	: Serial Clock
SI	: Serial Input
SO	: Serial Output
SB0, 1	: Serial Bus 0, 1
$\overline{\text{RESET}}$	: Reset Input
MD0-3	: Mode 0-3

S0-31	: Segment Output 0-31
COM0-3	: Common Output 0-3
V_{LC0-2}	: LCD Power Supply 0-2
BIAS	: LCD Power Supply Bias Control
LCDCL	: LCD Clock
SYNC	: LCD Synchronization
TI0	: Timer Input 0
PTO0	: Programmable Timer Output 0
BUS	: 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
V_{PP}	: Programming Power Supply

2.1 List of Pin Functions

2.1.1 Normal operating mode

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

Pin Name	Input/ Output	Dual-Purpose Pin	Function	8-Bit I/O	When Reset	Input/Output Circuit Type[Note 1]
P00	Input	INT4	4-bit input port (PORT 0). P01 to P03 enables to specify incorporation of pull-up resistors by software in 3-bit units.	x	Input	B
P01	I/O	$\overline{SCK}$				F – A
P02	I/O	SO/SB0				F – B
P03	I/O	SI/SB1				M – C
P10	Input	INT0	With noise elimination function	x	Input	B – C
P11		INT1	4-bit input port (PORT 1). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.			
P12		INT2				
P13		TI0				
P20	I/O	PTO0	4-bit input/output port (PORT 2). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	x	Input	E – B
P21		–				
P22		PCL				
P23		BUZ				
P30[Note 2]	I/O	LCDCL	Programmable 4-bit input/output port (PORT 3). Bitwise input/output set enabled. This port enables to specify incorporation of pull-up resistors software in 4-bit units.	x	Input	E – B
P31[Note 2]		SYNC				
P32[Note 2]		–				
P33[Note 2]		–				
P40-P43[Note 2]	I/O	–	N-ch open drain 4-bit input/output port (Port 4). Bitwise pull-up resistor incorporate enabled (mask option). 10 V withstand in open-drain mode	○	High level (when pull-up resistors are incorporated) or high-impedance	M
P50-P53[Note 2]	I/O	–	N-ch open drain 4-bit input/output port (Port 5). Bitwise pull-up resistor incorporate enabled (mask option). 10 V withstand in open-drain mode		High level (when pull-up resistors are incorporated) or high-impedance	M
P60	I/O	KR0	Programmable 4-bit input/output port (PORT 6). Bitwise input/output set enabled. This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	○	Input	F – A
P61		KR1				
P62		KR2				
P63		KR3				

Notes 1. Circles indicate Schmitt trigger inputs.

2. Direct LED drive enabled

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

Pin Name	Input/ output	Dual-Purpose pin	Function	8-Bit I/O	When Reset	Input/Output Circuit Type[Note 1]
P70	I/O	KR4	4-bit input/output port (PORT 7). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	○	Input	F – A
P71		KR5				
P72		KR6				
P73		KR7				
BP0	Output	S24	1-bit output port (BIT PORT). Also serves as the segment output pin.	x	**Note 2**	G – C
BP1		S25				
BP2		S26				
BP3		S27				
BP4	Output	S28				
BP5		S29				
BP6		S30				
BP7		S31				

Notes 1. Circles indicate Schmitt trigger inputs.

2. BP0 to BP7 select V_{LC1} as the input source. The output level varies depending on the external circuit of BP0 to BP7 and V_{LC1}.

Example Since BP0 to BP7 are interconnected through the μPD75308 as shown below, the output level of BP0 to BP7 is determined by the values of R_1 to R_3.

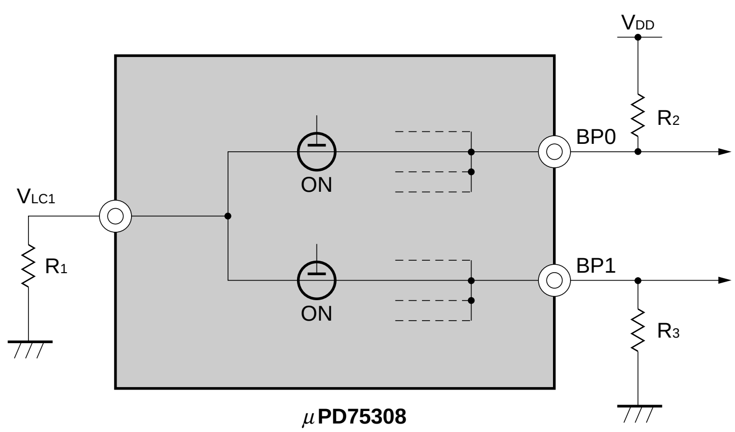


Table 2-2. List of Pin Functions Except Port Pins

Pin Name	Input/ Output	Dual- Purpose Pin	Function	When Reset	Input/Output Circuit Type[Note 1]
TI0	Input	P13	External event pulse input pin for the timer/event counter	Input	Ⓑ – C
PTO0	I/O	P20	Timer/event counter output pin	Input	E – B
PCL	I/O	P22	Clock output pin	Input	E – B
BUZ	I/O	P23	Fixed frequency output pin (for the buzzer or system clock trimming)	Input	E – B
$\overline{\text{SCK}}$	I/O	P01	Serial clock input/output pin	Input	Ⓕ – A
SO/SB0	I/O	P02	Serial data output pin Serial bus input/output pin	Input	Ⓕ – B
SI/SB1	I/O	P03	Serial data input pin Serial bus input/output pin	Input	Ⓜ – C
INT4	Input	P00	Edge detect vectored interrupt input pin (valid for both rising and falling edge detect)	Input	Ⓑ
INT0	Input	P10	Edge detect vectored interrupt input pin (detect edge select enabled) — Clock synchronous	Input	Ⓑ – C
INT1		P11	Asynchronous		
INT2	Input	P12	Edge detect testable input pin (rising edge detect) — Asynchronous	Input	Ⓑ – C
KR0-KR3	I/O	P60 to P63	Parallel falling edge detect testable input pin	Input	Ⓕ – A
KR4-KR7	I/O	P70 to P73	Parallel falling edge detect testable input pin	Input	Ⓕ – A
S0-S23	Output	–	Segment signal output pin	**Note 2**	G – A
S24-S31	Output	BP0 to BP7	Segment signal output pin	**Note 2**	G – C
COM0-COM3	Output	–	Common signal output pin	**Note 2**	G – B
V_{LC0}-V_{LC2}	–	–	LCD drive power pin having on-chip dividing resistors (mask option)	–	–
BIAS	Output		Output pin for cutting the externally mounted dividing resistors	**Note 3**	–
LCDCL[Note 4]	I/O	P30	Clock output pin for driving the externally expanded driver	Input	E – B
SYNC[Note 4]	I/O	P31	Clock output pin for synchronizing the externally expanded driver	Input	E – B
X1, X2	Input	–	Crystal/ceramic connection pin for main system clock oscillation. In the case of an external clock, an input is applied to X1 and the inverse input is applied to X2.	–	–
XT1	Input	–	Crystal connection pin for subsystem clock oscillation. In the case of an external clock, an input is applied to XT1, and XT2 is opened. XT1 can be used as a 1-bit input (test) pin.	–	–
XT2	–				
$\overline{\text{RESET}}$	Input	–	System reset input pin	–	Ⓑ
NC[Note 5]	–	–	No connection	–	–
IC[Note 6]	–	–	Internally Connected Connect this pin with V_{DD} directly.	–	–
V_{DD}	–	–	Positive power pin	–	–
V_{SS}	–	–	Ground potential pin	–	–

Notes 1. Circles indicate Schmitt trigger inputs.

2. The following V_{LCX} is selected as the input source for each display output.

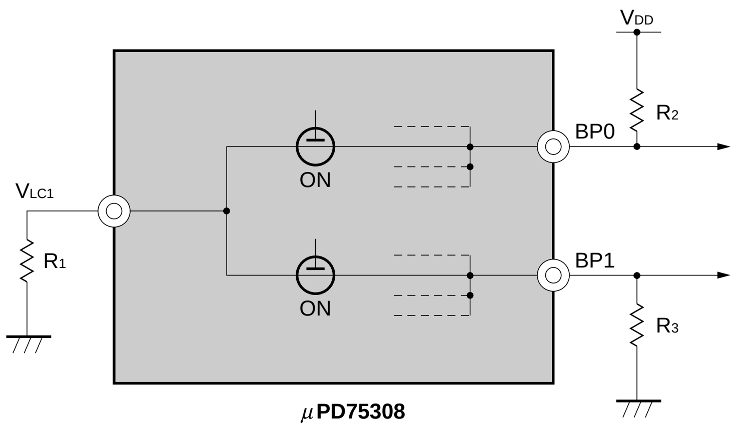
S0 to S31 : V_{LC1}

COM0 to COM2 : V_{LC2}

COM3 : V_{LC0}

Each display output level varies depending on the output value and the V_{LCX} external circuit type.

Example Since BP0 to BP7 are interconnected through the μPD75308 as shown below, the output level of BP0 to BP7 is determined by the values of R_1 to R_3.



3. When a dividing resistor is incorporated ... Low level

When a dividing resistor is not incorporated ... High impedance

4. Pins reserved for future system expansion. It is currently used only as P30 and P31 pins.

5. When the printed board is shared with the μPD75P308, 75P316, 75P316A and 75P316B, connect NC pin to V_{DD} (during emulation).

6. When μPD75312B or 75316B is used.

Phase-out/Discontinued

2.1.2 PROM mode

Table 2-3. List of Digital Input/Output Port Pin Functions

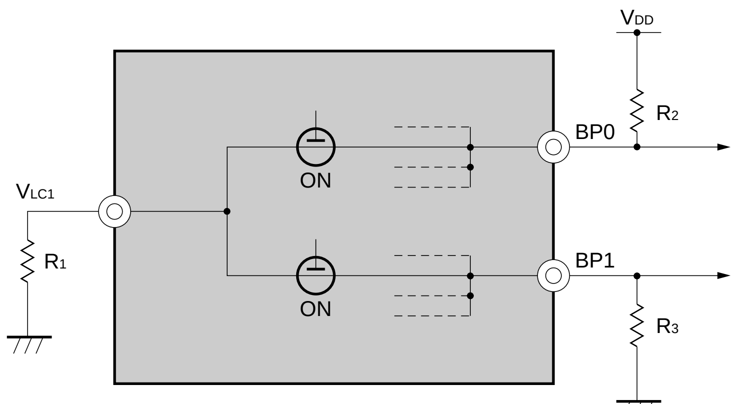
Pin Name	Input/ Output	Dual-Purpose Pin	Function	8-Bit I/O	When Reset	Input/Output Circuit Type Note 1
P00	Input	INT4	4-bit input port (PORT 0). P01 to P03 enables to specify incorporation of pull-up resistors by software in 3-bit units.	x	Input	B
P01	I/O	$\overline{SCK}$				F – A
P02	I/O	SO/SB0				F – B
P03	I/O	SI/SB1				M – C
P10	Input	INT0	With noise elimination function	x	Input	B – C
P11		INT1	4-bit input port (PORT 1). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.			
P12		INT2				
P13		TI0				
P20	I/O	PTO0	4-bit input/output port (PORT 2). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	x	Input	E – B
P21		–				
P22		PCL				
P23		BUZ				
P30 Note 2	I/O	LCDCL / MD0	Programmable 4-bit input/output port (PORT 3). Bitwise input/output set enabled. This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	x	Input	E – B
P31 Note 2		SYNC / MD1				
P32 Note 2		MD2				
P33 Note 2		MD3				
P40 to P43 Note 2	I/O	–	N-ch open-drain 4-bit input/output port (PORT 4). Data input/output pin (low-order 4 bits) for program memory (EPROM) write/verify.	○	High impedance	M – A
P50 to P53 Note 2	I/O	–	N-ch open-drain 4-bit input/output port (PORT 5). Data input/output pin (high-order 4 bits) for program memory (EPROM) write/verify.		High impedance	M – A
P60	I/O	KR0	Programmable 4-bit input/output port (PORT 6). Bitwise input/output set enabled. This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	○	Input	F – A
P61		KR1				
P62		KR2				
P63		KR3				
P70	I/O	KR4	4-bit input/output port (PORT 7). This port enables to specify incorporation of pull-up resistors by software in 4-bit units.	○	Input	F – A
P71		KR5				
P72		KR6				
P73		KR7				
BP0	Output	S24	1-bit output port (BIT PORT). Also serves as the segment output pin.	x	**Note 3**	G – C
BP1		S25				
BP2		S26				
BP3		S27				
BP4	Output	S28				
BP5		S29				
BP6		S30				
BP7		S31				

Notes 1. Circles indicate Schmitt trigger inputs.

2. Direct LED drive enabled

3. BP0 to BP7 select V_{LC1} as the input source. The output level varies depending on the external circuit of BP0 to BP7 and V_{LC1}.

Example Since BP0 to BP7 are interconnected through the μPD75308 as shown below, the output level of BP0 to BP7 is determined by the values of R_1 to R_3.



μPD75308

Table 2-4. List of Pin Functions Except Port Pins

Pin Name	Input/ Output	Dual-Purpose Pin	Function	When Reset	Input/Output Circuit Type[Note 1]
TI0	Input	P13	External event pulse input pin for the timer/event counter	–	B – C
PTO0	I/O	P20	Timer/event counter output pin	Input	E – B
PCL	I/O	P22	Clock output pin	Input	E – B
BUZ	I/O	P23	Fixed frequency output pin (for the buzzer or system clock trimming)	Input	E – B
$\overline{\text{SCK}}$	I/O	P01	Serial clock input/output pin	Input	F – A
SO/SB0	I/O	P02	Serial data output pin Serial bus input/output pin	Input	F – B
SI/SB1	I/O	P03	Serial data input pin Serial bus input/output pin	Input	M – C
INT4	Input	P00	Edge detect vectored interrupt input pin (valid for both rising and falling edge detect)	–	B
INT0	Input	P10	Edge detect vectored interrupt input pin (detect edge select enabled): Clock synchronous	–	B – C
INT1		P11	Asynchronous		
INT2	Input	P12	Edge detect testable input pin (rising edge detect): Asynchronous	–	B – C
KR0-KR3	I/O	P60 to P63	Testable input/output pin (parallel falling edge detect)	Input	F – A
KR4-KR7	I/O	P70 to P73	Testable input/output pin (parallel falling edge detect)	Input	F – A
S0-S23	Output	–	Segment signal output pin	**Note 2**	G – A
S24-S31	Output	BP0 to BP7	Segment signal output pin	**Note 2**	G – C
COM0-COM3	Output	–	Common signal output pin	**Note 2**	G – B
V_{LC0}-V_{LC2}	–	–	LCD drive power pin	–	–
BIAS	–	–	Output pin for cutting the externally mounted dividing resistors	**Note 3**	–
LCDCL[Note 4]	I/O	P30	Clock output pin for driving the externally expanded driver	Input	E – B
SYNC[Note 4]	I/O	P31	Clock output pin for synchronizing the externally expanded driver	Input	E – B
X1, X2	Input	–	Crystal/ceramic connection pin for main system clock oscillation. In the case of an external clock, an input is applied to X1 and the inverse input is applied to X2.	–	–
XT1	Input	–	Crystal connection pin for subsystem clock oscillation. In the case of an external clock, an input is applied to XT1, and XT2 is opened. XT1 can be used as a 1-bit input (test) pin.	–	–
XT2	–				
$\overline{\text{RESET}}$	Input	–	System reset input pin (low level active)	–	B
MD0-MD3	I/O	P30 to P33	Program memory (PROM) write/verify mode select pin	Input	E – B
V_{PP}	–	–	Program memory (PROM) write/verify program voltage apply pin. Connect to V_{DD} for normal operations. Apply a voltage of +12.5 V for PROM write/verify.	–	–
V_{DD}	–	–	Positive power pin	–	–
V_{SS}	–	–	Ground potential pin	–	–

Notes 1. Circles indicate Schmitt trigger inputs.

2. The following V_{LCX} is selected as the input source for each display output.

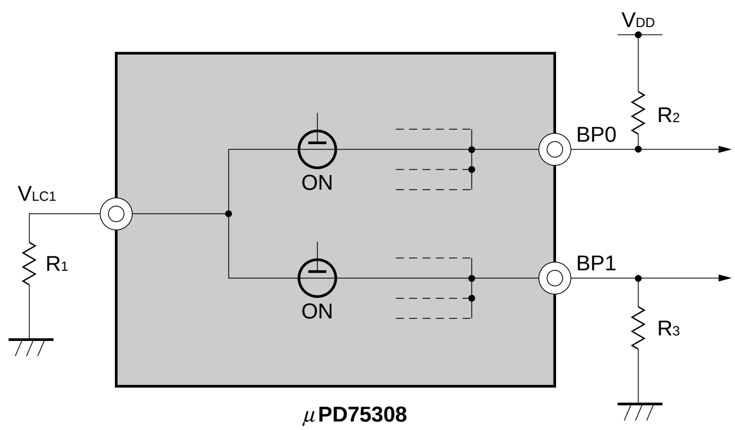
S0 to S31 : V_{LC1}

COM0 to COM2 : V_{LC2}

COM3 : V_{LC0}

Each display output level varies depending on the output value and the V_{LCX} external circuit type.

Example Since BP0 to BP7 are interconnected through the μPD75308 as shown below, the output level of BP0 to BP7 is determined by the values of R_1 to R_3.



3. When a dividing resistor is incorporated ... Low level

When a dividing resistor is not incorporated ... High impedance

4. Pins reserved for future system expansion. It is currently used only as P30 and P31 pins.

2.2 Description of Pin Functions

2.2.1 P00 to P03 (Port 0): INT4, $\overline{\text{SCK}}$, SO/SB0, SI/SB1 multi-purpose inputs
P10 to P13 (Port 1): INT0 to INT2, T10 multi-purpose inputs

4-bit input port: Input pins of ports 0 and 1

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

(1) Port 0 : Vectored interrupt input (INT4)
Serial interface input/output ($\overline{\text{SCK}}$, SO/SB0, SI/SB1)

(2) Port 1 : Vectored interrupt input (INT0, INT1)
Edge detect test input (INT2)
External event pulse input to timer/event counter (TI0)

The pins of ports 0 and 1 always serve for inputs irrespectively of the operation of the multi-purpose pins.

They are Schmitt trigger inputs in order not to malfunction due to noise. P10 is equipped with a noise eliminator (refer to **6.3 (3) INT0, INT1 and INT4 hardware** for details).

An on-chip pull-up resistor can be specified using the software for port 0 in 3-bit units (P01 to P03) and port 1 in 4-bit units (P10 to P13). This specification can be done by operating pull-up resistor specify register group A.

Each pin operates as an input port when $\overline{\text{RESET}}$ signal is generated.

2.2.2 P20 to P23 (Port 2): PTO0, PCL, BUZ multi-purpose input/output
P30 to P33 (Port 3): LCDCL, SYNC, MD0 to MD3[Note] multi-purpose input/output
P40 to P43 (Port 4),
P50 to P53 (Port 5): N-ch open-drain intermediate withstand voltage (10 V) high-current output
P60 to P63 (Port 6),
P70 to P73 (Port 7): 3-state input/output

4-bit input/output port with an output latch: Input/output pins of ports 2 to 7

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

(1) Port 2 : Timer/event counter output (PTO0)
Clock output (PCL)
Fixed frequency output (BUZ)

(2) Port 3 : Clock output for operating the LCD externally expanded driver (LCDCL)
Clock output for synchronizing the LCD externally expanded driver (SYNC)

(3) Ports 6, 7 : Key interrupt input (KR0 to KR3, KR4 to KR7)

Ports generate a high-current output and can directly drives the LED. Ports 4 and 5 generate a N-ch open drain intermediate withstand voltage (10 V) high-current output and can directly drives the LED.

Input/output mode selection is set using the port mode register. Port m (m = 2, 4, 5, 7) enables to set input/output in 4-bit units. Ports 3 and 6 enable to set input/output bitwise.

Port n enables to specify incorporation of pull-resistors by software in 4-bit units. This specification can be done by operating the pull-up resistor specify register group A (POGA). Ports 4 and 5 enable to specify incorporation of pull-up resistors bitwise by mask option.

Ports 4 and 5 and ports 6 and 7 enable in pairs to select input/output in 8-bit units.

The output latch of each port is cleared when the $\overline{\text{RESET}}$ signal is generated. As a result, port n is set to the input mode (output high impedance) and ports 4 and 5 are set to the high level (when a pull-up resistor is incorporated) or high impedance.

Note Only the μPD75P308, 75P316, 75P316A and 75P316B serve as MD0 to MD3.

2.2.3 BP0 to BP7: LCD controller/driver segment signal output (S24 to S31) dual-purpose output

1-bit output port with an output latch:

Output pins of bit ports 0 to 7. They also serve as the LCD controller/driver segment signal output pins.

2.2.4 TI0: Port 1 dual-purpose input

External event pulse input pin of the programmable timer/event counter.

TI0 serves as a Schmitt trigger input.

2.2.5 PTO0: Port 2 dual-purpose output

Output pin of the programmable timer/event counter. It generates square-wave pulses. To generate the programmable timer/event counter signal, the P20 output latch is cleared (0) and the bit of the port mode register port 2 is set to the output mode (1).

The output is cleared (0) by the timer start instruction.

2.2.6 PCL: Port 2 dual-purpose output

Programmable clock output pin. It is used to supply clocks to the peripheral LSI (slave microcontroller, A/D converter, etc.). When the $\overline{\text{RESET}}$ signal is generated, the clock mode register (CLOM) is cleared (0) and the clock is disabled for output with the result that PCL is set to the normal port operating mode.

2.2.7 BUZ: Port 2 dual-purpose output

Fixed frequency output pin. It is used to generate buzzer sound or to trim the system clock oscillation frequency by generating the fixed frequency (2.048 kHz). It also serves as P23 pin and is only effective when bit 7 (WM7) of the timer mode register (WM) is set (1). When the $\overline{\text{RESET}}$ signal is generated, WM7 is cleared (0) and BUZ is set to the normal port operating mode.

2.2.8 $\overline{\text{SCK}}$, SO/SB0, SI/SB1: Port 0 dual-purpose 3-state input/output

Input/output pin for the serial interface. It operates in accordance with the setting of the serial operating mode register (CSIM).

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

These ports serve as Schmitt trigger inputs.

2.2.9 INT4: Port 0 dual-purpose input

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

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

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

2.2.10 INT0, INT1: Port 1 dual-purpose inputs

They are edge detect vectored interrupt input pins. INT0 has the noise elimination function. They can select the detect edge with the edge detect mode registers (IM0, IM1).

(1) INT0 (bits 0, 1 of IM0)

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

(2) INT1 (bit 0 of IM1)

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

INT0 has the noise elimination function and can change the noise elimination sampling clock at two levels. The width of a signal to be acknowledged depends on the CPU operation clock.

INT1 is an asynchronous input and is acknowledged, irrespectively of the CPU operation clock, when the input has the specified high level width.

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

INT1 can also be for STOP mode and HALT mode release, but INT0 cannot.

INT0 and INT1 are Schmitt trigger inputs.

2.2.11 INT2: Port 1 dual-purpose input

External test input pin with both rising and falling edges set to active. If the signal input to this pin changes from low to high when INT2 is selected by the edge detect mode register (IM2), the internal test flag (IRQ2) is set.

INT2 is an asynchronous input and is acknowledged, irrespectively of the CPU operation clock, when the input has the specified high level width.

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

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

2.2.12 KR0 to KR3: Port 6 dual-purpose input
KR4 to KR7: Port 7 dual-purpose input

Key interrupt input pins. KR0 to KR7 are parallel falling edge detect interrupt input pins. An interrupt format can be specified in accordance with the edge detect mode register (IM2).

When the $\overline{\text{RESET}}$ signal is generated, the port 6/port 7 input mode is set.

2.2.13 S0 to S23
S24 to S31: Bit ports 0 to 7 dual-purpose outputs

Segment signal output pins which can directly drive the LCD segment pin (front electrode). They activate the static method, 2 or 3-time sharing of the 1/2 bias method or 3 or 4-time sharing of the 1/3 bias method.

S0 to S23 also serve as segment dedicated output pins and S24 to S31 also serve as the output pins of bit ports 0 to 7. They are switched using the display mode register (LCDM).

2.2.14 COM0 to COM3

Common signal output pins which can directly drive the LCD common pin (rear electrode). They generate the common signal when the static method (COM0, 1, 2, 3 outputs), 2-time sharing (COM0, 1 outputs) or 3-time sharing (COM0, 1, 2 outputs) of the 1/2 bias method, 3-time sharing (COM0, 1, 2 outputs) or 4-time sharing (COM0, 1, 2, 3 outputs) of the 1/3 bias method is activated.

2.2.15 V_{LC0} to V_{LC2}

LCD drive power supply pin. The μPD75308 can have an on-chip dividing resistor in the V_{LC0} to V_{LC2} pins so that an LCD drive power can be supplied in accordance with each bias method without the use of an externally mounted dividing resistor (mask option).

2.2.16 BIAS

Output pin for cutting the dividing resistor. Connected to the V_{LC0} pin to cope with various LCD drive voltages, it is used to change the resistance division ratio. Along with the V_{LC0} to V_{LC2} pins or V_{SS} pin, BIAS with an externally connected resistor is used to fine adjust the LCD drive power voltage values.

2.2.17 LCDCL

Clock output pin for driving the LCD externally expanded driver.

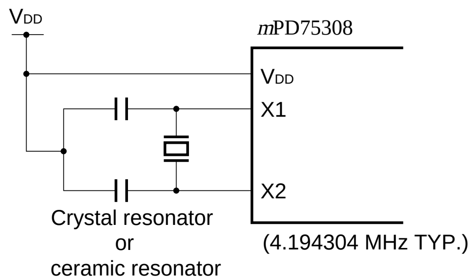
2.2.18 SYNC

Clock output pin for synchronizing the LCD externally expanded driver.

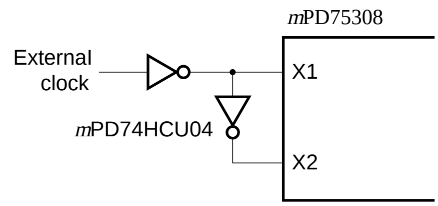
2.2.19 X1, X2

Crystal/ceramic connection pins for main system clock oscillation.
These pins enable to input external clocks.

(a) Crystal/ceramic oscillation



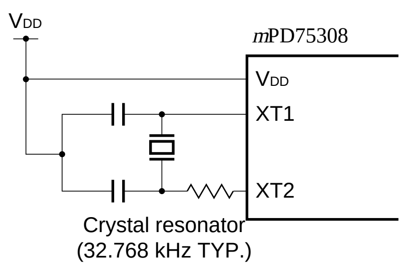
(b) External clock



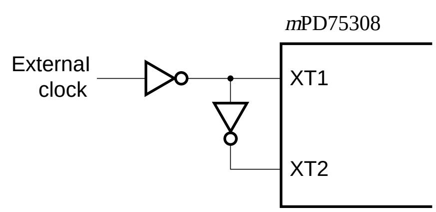
2.2.20 XT1, XT2

Crystal connection pin for subsystem clock oscillation.
These pins enable to input external clocks.

(a) Crystal oscillation



(b) External clock



2.2.21 $\overline{\text{RESET}}$

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 irrespectively of the operation clock, the $\overline{\text{RESET}}$ signal is generated and system reset is applied before all other operations are carried out.

In addition to the normal CPU initialize/start operation, this pin is used to release the standby (STOP/HALT) mode.

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

2.2.22 V_{DD}

Positive power supply pin

2.2.23 V_{SS}

Ground potential

2.2.24 V_{PP}

This pin function is only available for the μPD75P308, 75P316, 75P316A and 75P316B.

This is a voltage apply pin to write/verify for the PROM (program memory). For normal operations, V_{PP} is connected to V_{DD} directly. A voltage of +12.5 V is applied to write/verify for the PROM.

2.2.25 MD0 to MD3: Port 3 dual-purpose inputs/outputs

These pin functions are only available for the μPD75P308, 75P316, 75P316A and 75P316B.

They are used to select the operation mode to write/verify for the PROM (program memory).

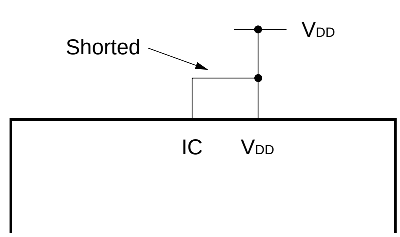
2.2.26 IC

This pin function is only available for the μPD75312B and 75316B.

The IC (Internally Connected) pin is used to set the test mode for testing the μPD75312B/75316B when shipped from NEC. In normal operation, the IC pin should be directly connected to V_{DD}, and the wiring should be kept as short as possible.

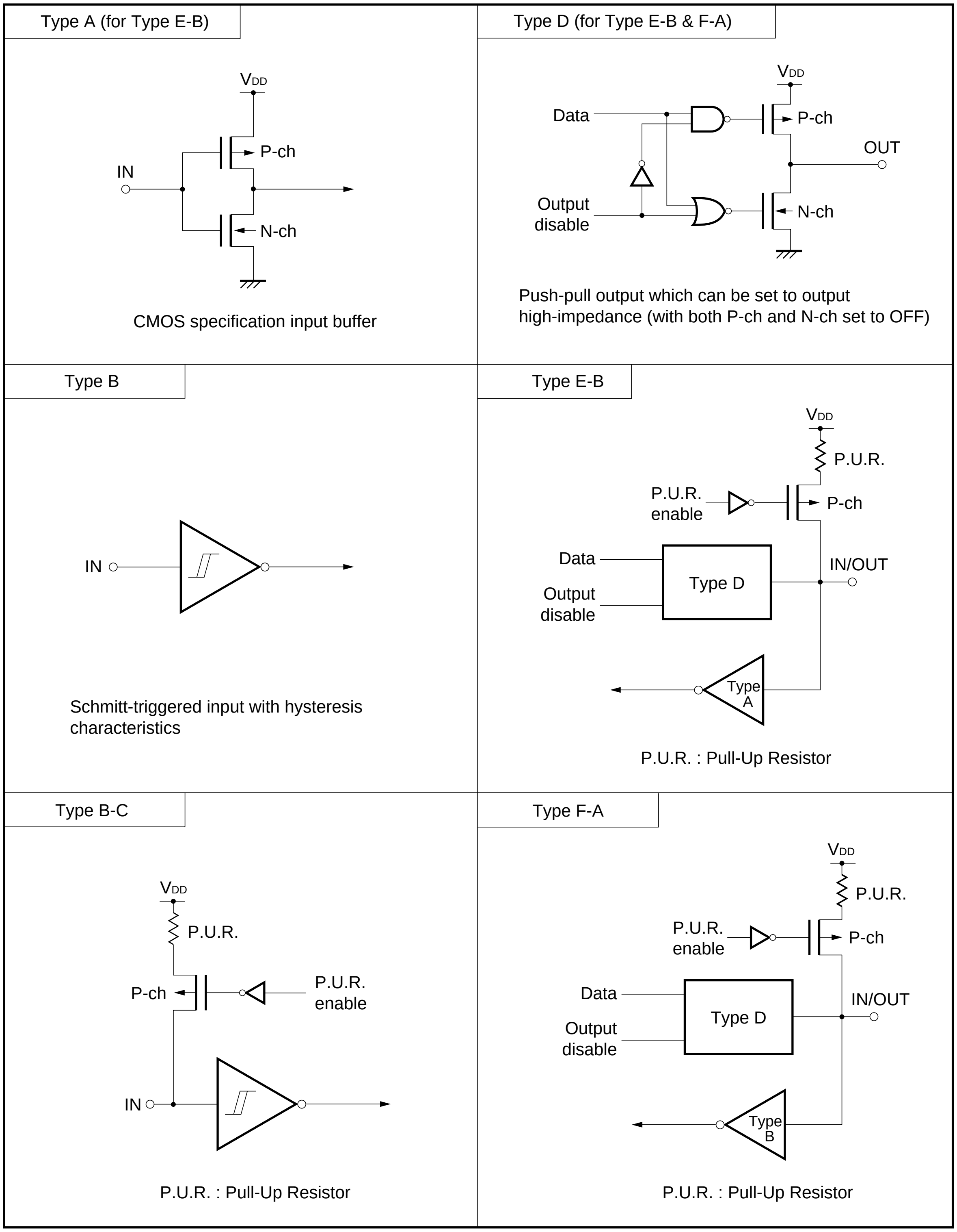
If a potential difference arises between the IC pin and V_{DD} when the routing of the wiring between the IC pin and V_{DD} pin is long, or when external noise is applied to the IC pin, for instance, the customer's program may not run correctly.

• **Connect the IC pin directly to the V_{DD} pin.**

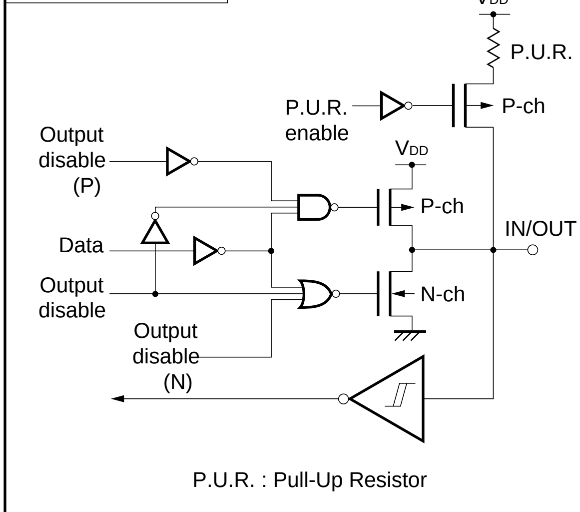


2.3 Input/Output Circuits of Pins

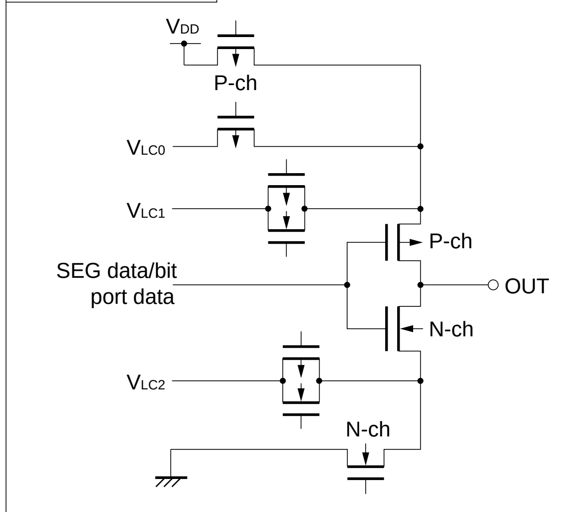
The input/output circuits of μPD75308 pins are shown in simplified form.



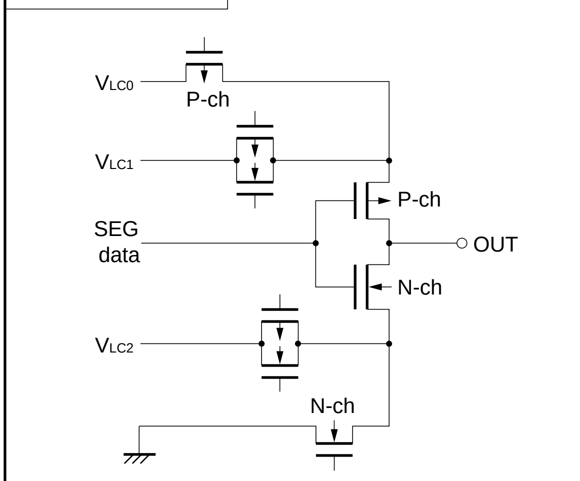
Type F-B



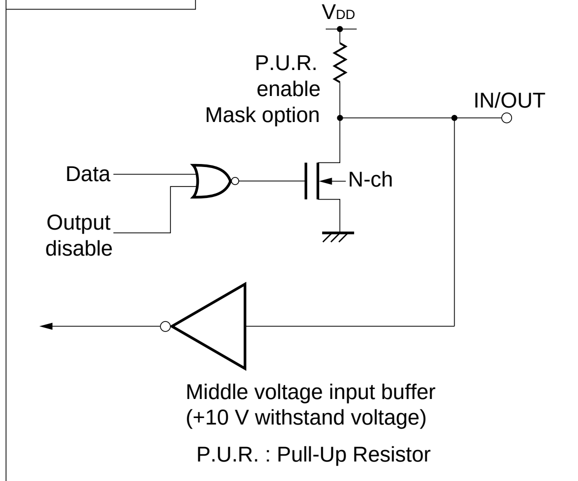
Type G-C



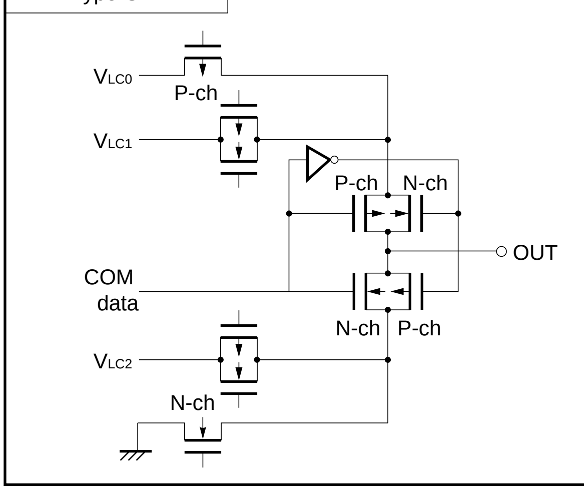
Type G-A



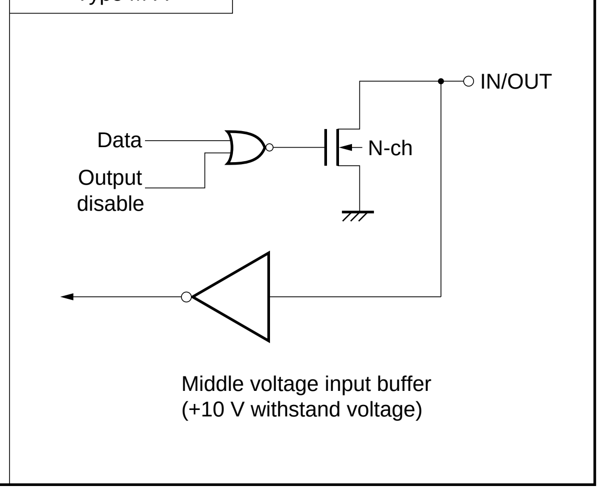
Type M

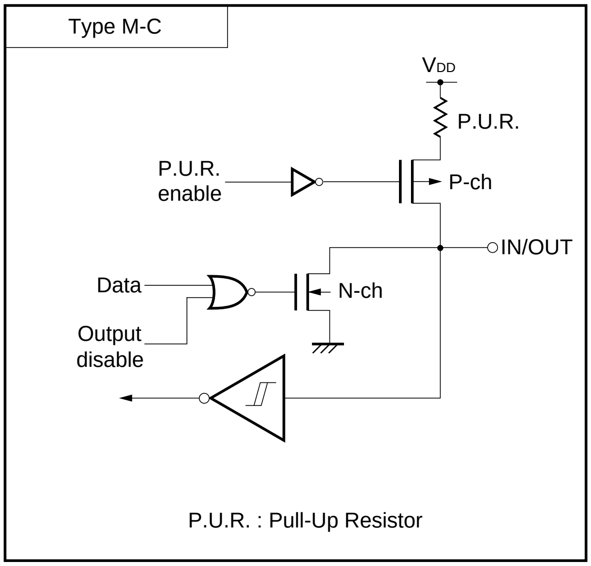


Type G-B



Type M-A





2.4 Selection of Mask Option

The following mask options are available for the pins. No mask options are available for the μPD75P308, 75P316, 75P316A and 75P316B.

Table 2-5. Selection of Mask Option

Pin Name	Mask Option
P40 to P43, P50 to P53	• Pull-up resistor available (specifiable as bit-wise) • Pull-up resistor not available (specifiable as bit-wise)
V_{LC0} to V_{LC2}, BIAS	• Dividing resistor for LCD drive power supply available (specifiable in 4-bit units) • Dividing resistor for LCD drive power supply not available (specifiable in 4-bit units)

2.5 Treatment of Unused Pins

Table 2-6. Treatment 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-P12/INT2	Connect to V_{SS}.
P13/TI0	
P20/PTO0	Input state : Connect to V_{SS} or V_{DD}. Output state : Leave unconnected.
P21	
P22/PCL	
P23/BUZ	
P30-P33	
P40-P43	
P50-P53	
P60-P63	
P70-P73	
S0-S23	Leave unconnected.
S24/BP0-S31/BP7	
COM0-COM3	
V_{LC0}-V_{LC2}	Connect to V_{SS}.
BIAS	Connect to V_{SS} only when none of V_{LC0} to V_{LC2} are used. Leave unconnected in all other cases.
XT1	Connect to V_{SS} or V_{DD}.
XT2	Leave unconnected.
V_{PP}	Connect to V_{DD} directly.
IC	

2.6 Caution of Use of P00/INT4 Pin and $\overline{\text{RESET}}$ Pin

This caution does not apply to the μPD75312B, 75316B, 75P316A or 75P316B.

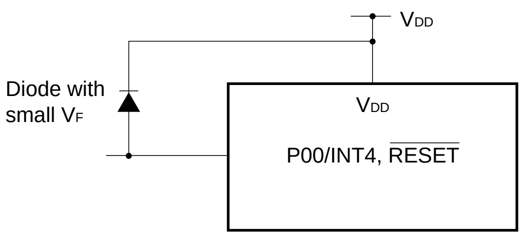
In addition to the functions shown in **2.2.9 INT4** and **2.2.21 $\overline{\text{RESET}}$**, the P00/INT4 pin and $\overline{\text{RESET}}$ pin have a function for setting the test mode in which the internal operation of the μPD75308 is tested (IC test only).

When a potential greater than V_{DD} is applied to either of these pins, the test mode is set. As a result, if noise exceeding V_{DD} is applied during normal operation, the test mode will be entered and normal operation may be impeded.

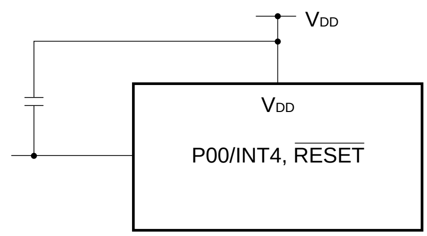
If, for example, the routing of the wiring between the P00/INT4 pin and $\overline{\text{RESET}}$ pin is long, the above problem may occur as the result of inter-wiring noise between these pins.

Therefore, wiring should be carried out so as to eliminate inter-wiring noise as far as possible. If it is not possible to eliminate noise, anti-noise measures should be taken using external parts as shown in the figures below.

- **Connection of diode with small V_F between P00/INT4/$\overline{\text{RESET}}$ pin and V_{DD}**



- **Connection of capacitor between P00/INT4/$\overline{\text{RESET}}$ pin and V_{DD}**



[MEMO]

CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP

In the architecture of the 75X series used for the μPD75308,

- Maximum capacity of 4K words x 4 bits (12-bit address) for the on-chip RAM
- Expandability of the peripheral hardware

have been adopted to realize the following features:

(1) Data memory bank configuration

(2) Memory mapped I/O

This chapter describes the data memory bank configuration and the memory mapped I/O.

3.1 Data Memory Bank Configuration and Addressing Mode

3.1.1 Data memory bank configuration

512 words x 4 bits (1024 words x 4 bits[Note]) of static RAM is included on-chip in addresses 000H through 1FFH (000H through 3FFH[Note]) of the data memory space. Of this, addresses 1E0H through 1FFH are 32-words x 4-bit display data memory. Peripheral hardware (input/output ports and timers etc.) is allocated to addresses F80H through FFFH.

For addressing the 12-bit address (4K words x 4 bits) data memory space, the μPD75308 has a memory bank configuration in which instructions are used to directly or indirectly specify the low-order 8-bit addresses and the memory bank is used to specify the high-order 4-bit addresses.

To specify the memory bank (MB), the μPD75308 has the following two on-chip hardware units:

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

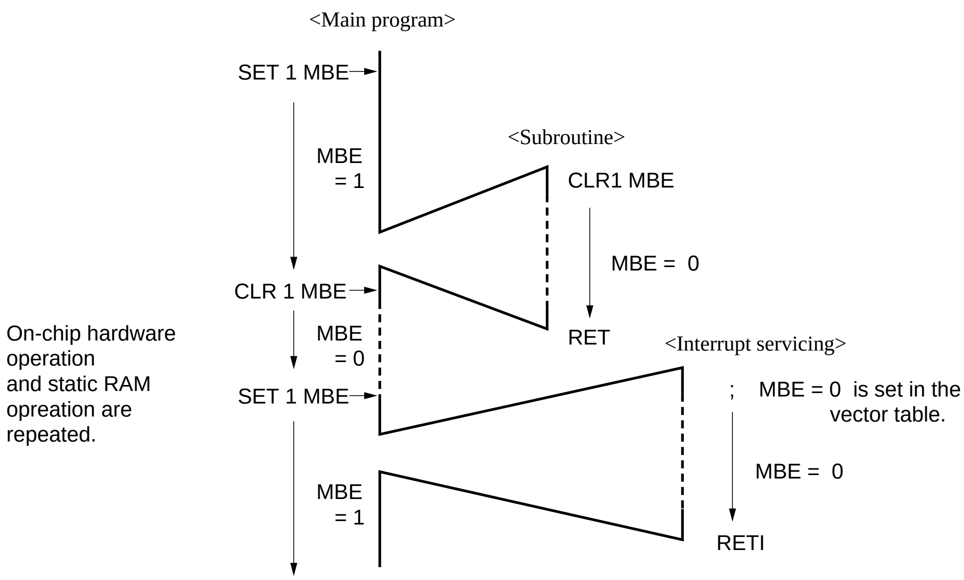
The MBS is a register to select the memory bank and banks 0, 1 and 15 (0, 1, 2, 3 and 15[Note]) can be set for the μPD75308. The MBE is a flag to determine if the memory bank selected by MBS should be validated. When MBE is 0, the selected memory bank (MB) is fixed irrespectively of the MBS as shown in Figure 3-1. When MBE is 1, the data memory space can be expanded by switching the memory bank according to MBS setting.

For data memory space addressing, MBE = 1 is usually set and the data memory of the memory bank specified by MBS is manipulated. To execute programming efficiently, the MBE = 0 mode or the MBE = 1 mode can be selected in each program processing.

Note μPD75312B, 75316B, 75P316A and 75P316B only

	Appropriate Program Processing	Effect
MBE = 0 mode	• Interrupt servicing	MBS save/restore is not necessary
	• Processing of repeating on-chip hardware operation and static RAM operation	MBS change is not necessary.
	• Subroutine processing	MBS save/restore is not necessary.
MBE = 1 mode	• Normal program processing	

Figure 3-1. Selection of MBE = 0 Mode and MBE = 1 Mode



Remark ———— : When MBE = 1, – – – – –: When MBE = 0

Since the MBE is automatically saved/restored during subroutine processing, it can be changed freely during that processing operation. In the interrupt servicing operation, the MBE is automatically saved/restored, and furthermore, MBE undergoing interrupt servicing can be specified upon start of interrupt servicing by setting the interrupt vector table so that high-speed interrupt servicing operations can be carried out efficiently.

When changing the MBS by executing the subroutine or interrupt servicing operation, the MBS is saved/ restored by the PUSH/POP instruction.

MBE is set by the SET1/CLR1 instruction. MBS is set by the SEL instruction.

Examples **1.** MBE is cleared and the memory bank is fixed.

```
CLR1    MBE    ; MBE ← 0
```

2. Memory bank 1 is selected.

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

3.1.2 Data memory addressing mode

In the 75X series architecture used for the μPD75308, seven addressing modes are available, as shown in Figure 3-2, for efficiently addressing the data memory space for each bit length of data to be processed.

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

In this addressing mode, the bits of the whole data memory space are directly specified by instruction operands. In the MBE = 0 mode, the memory bank (MB) is fixed to 0 when the operand specified addresses are 00H to 7FH and the MB is fixed to15 when the operand specified addresses are 80H to FFH.

Therefore both the 000H to 07FH data area and the F80H to FFFH peripheral hardware area can be addressed in the MBE = 0 mode.

In the MBE = 1 mode, MB becomes equal to MBS so that the specifiable data memory space can be expanded.

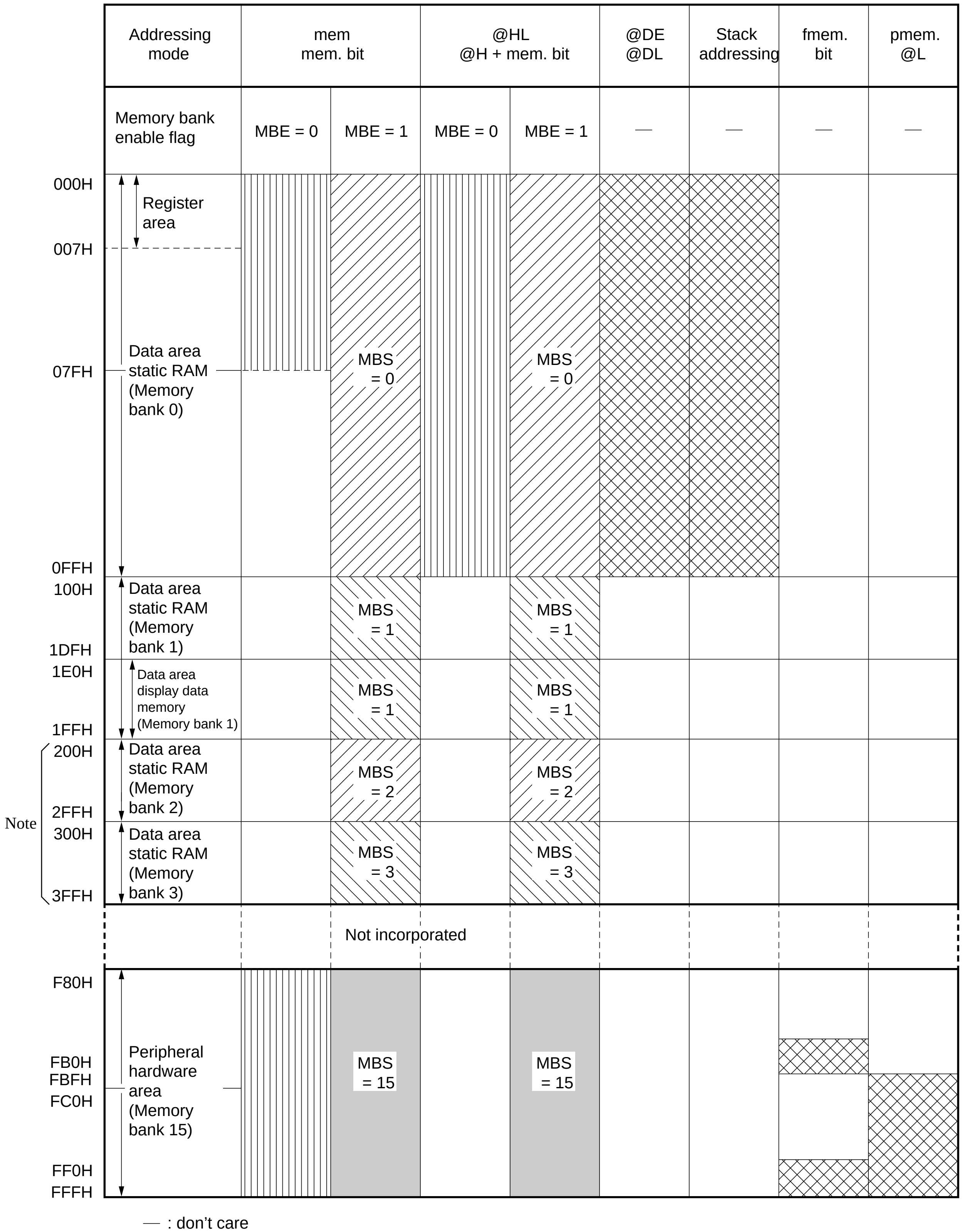
This addressing mode can be used for the bit set and reset instructions (SET1, CLR1) and the bit test instructions (SKT, SKF).

Example Test if FLAG1 is set, FLAG2 is reset, and FLAG3 is 0.

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

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

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



Note The μPD75312B, 75316B, 75P316A, 75P316B only

Figure 3-3. Addressing Mode

Addressing Mode	Format	Address to be Specified
1-bit direct addressing	mem. bit	Bit specified by ‘bit’ of address specified by ‘MB’ and ‘mem’ MBE = 0, When mem = 00H to 7FH, MB = 0 When mem = 80H to FFH, MB = 15 When MBE = 1, MB = MBS
4-bit direct addressing	mem	Address specified by ‘MB’ and ‘mem’ MBE = 0, When mem = 00H to 7FH, MB = 0 When mem = 80H to FFH, MB = 15 When MBE = 1, MB = MBS
8-bit direct addressing		Address specified by ‘MB’ and ‘mem (= even address)’ MBE = 0, When mem = 00H to 7FH, MB = 0 When mem = 80H to FFH, MB = 15 When MBE = 1, MB = MBS
4-bit register indirect addressing	@HL	Address specified by ‘MB’ and ‘HL’ when MB = MBE•MBS
	@DE	Address specified by ‘DE’ of memory bank 0
	@DL	Address specified by ‘DL’ of memory bank 0
8-bit register indirect addressing	@HL	Address specified by ‘MB’ and ‘HL’ (with L register having an even number content) when MB = MBE•MBS
Bit manipulation addressing	fmem.bit	Bit specified by ‘bit’ of address specified by ‘fmem’ fmem = FB0H to FBFH (interrupt related hardware) FF0H to FFFH (I/O port)
	pmem. @L	Bit specified by the low-order 2 bits of L register of the address specified by the high-order 10 bits of ‘pmem’ and the high-order 2 bits of L register when pmem = FC0H to FFFH
	@H+mem. bit	Bit specified by ‘bit’ of the address specified by MB, H and the low-order 4 bits of ‘mem’ when MB = MBE•MBS
Stack addressing		Address specified by SP of memory bank 0

Remark On the μPD75312B, 75316B, 75P316A, and 75P316B, MBS = 0, 1, 2, 3 or 15. Otherwise, MBS = 0, 1 or 15.

(2) 4-bit direct addressing (mem)

This addressing mode enables to directly specify the whole data memory space in 4-bit units by an instruction operand.

Like in 1-bit direct addressing, the specifiable area is fixed to the 000H to 07FH data area and the F80H to FFFH peripheral hardware area in the MBE = 0 mode.

In the MBE = 1 mode, MB becomes equal to MBS and the specifiable data memory space is expanded to the whole space.

This addressing mode is used for the MOV, XCH, INCS, IN and OUT instructions.

Caution As in example 1, program efficiency decreases if input/output port related data is stored in the static RAM of bank 1. If the data is stored at addresses 00H to 7FH of bank 0, programming can be performed without changing MBS as in the example 2.

Examples 1. 'BUFF' data is output to port 5.

```
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    PORT5, A    ; PORT5 ← A
```

2. Port 4 is input and is stored in 'DATA1'.

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

(3) 8-bit direct addressing (mem)

This addressing mode enanbles to directly specify the whole data memory space in 8-bit units by an instruction operand.

Only even addresses can be specified by operands. 4-bit data of the operand specified address and 4-bit data of the address added by one undergo 8-bit processing in pairs with the 8-bit accumulator (XA register pair).

The same memory bank as in 4-bit direct addressing is specified.

This addressing mode is used for the MOV, XCH, IN and OUT instructions.

Examples 1. 8-bit data of ports 4 and 5 are transferred to addresses 20H and 21H.

```
DATA   EQU    020H
       CLR1   MBE         ; MBE ← 0
       IN     XA, PORT4   ; X ← port 5, A ← port 4
       MOV    DATA, XA    ; (21H) ← X, (20H) ← A
```

2. As soon as 8-bit data input to the shift register (SIO) of the serial interface is fetched, transfer data is set and transfer start is instructed.

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

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

This addressing mode enables to indirectly specify the data memory space in 4-bit units with the data pointer (register pair of the general register) specified by an instruction operand.

Three types of data pointers are available: HL register pair which can specify the whole data memory space by specification of MB = MBE•MBS, DE register pair and DL register pair which are fixed to memory bank 0 irrespectively of MBE and MBS specification. Programming can be performed efficiently by selecting one of the three data pointers according to the bank of the data memory to be used.

Example 50H to 57H data is transferred to 110H to 117H.

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

The addressing mode using the HL register pair as the data pointer is widely used for data transfer, arithmetic operation, comparison, input/output and other relevant operations. The addressing mode using the DE/DL register pair is used for the MOV and XCH instructions.

In combination with the increment/decrement instruction of the general register or the register pair, the address of the data memory space can be updated freely as shown in Figure 3-4.

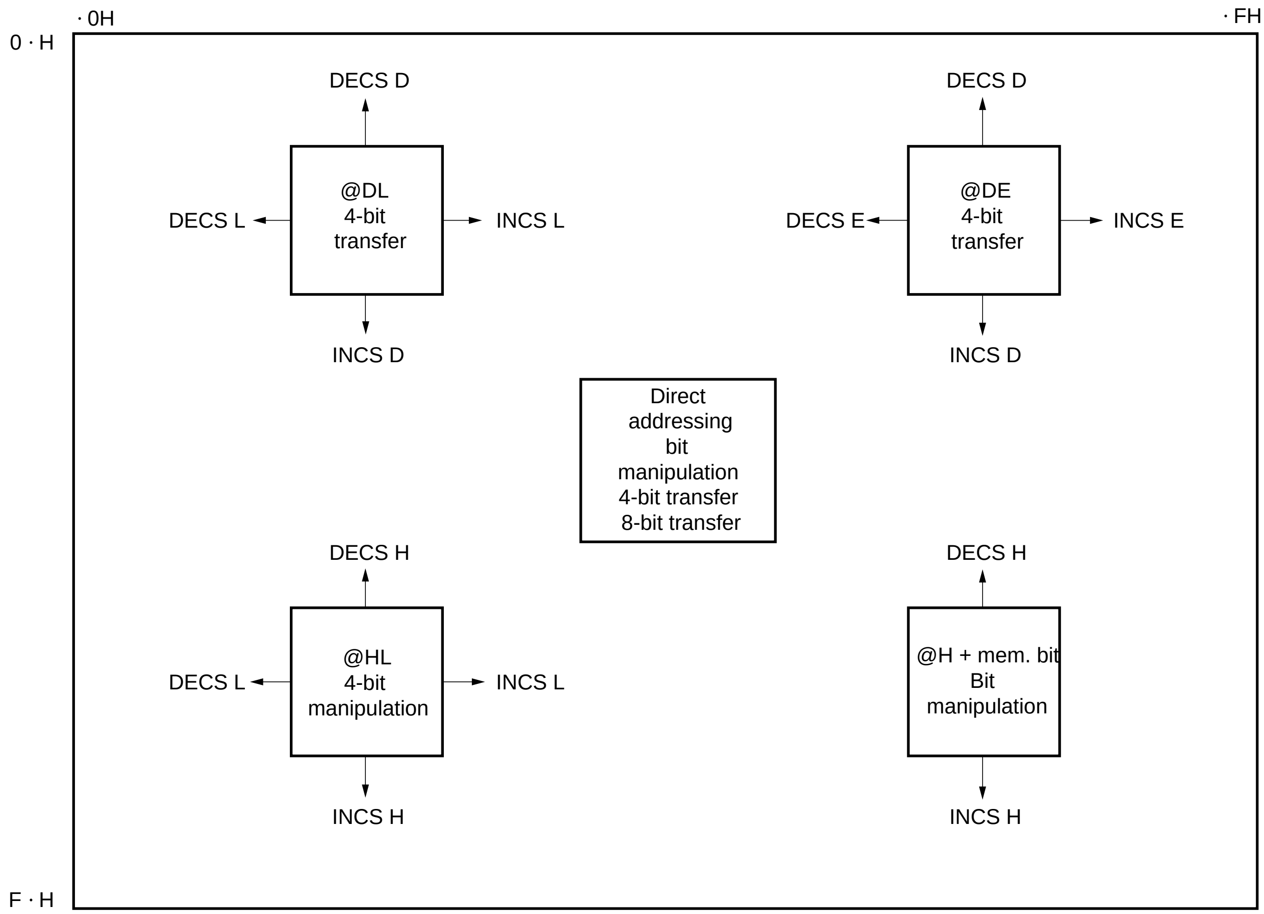
Example 1. 50H to 57H data is compared to 110H to 117H data.

```
DATA1   EQU     57H
DATA2   EQU     117H
        SET1    MBE
        SEL     MB1
        MOV     D, #DATA1 SHR4
        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. The 00H to FFH data memory is cleared to 0.

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

Figure 3-4. Static RAM Address Updating Procedure



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

This addressing mode enables to indirectly specify the whole data memory space in 8-bit units with the data pointer (HL register pair).

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

The specified memory bank is MB = MBE•MBS as when the HL register is specified by 4-bit register indirect addressing. This addressing mode is used for the MOV, XCH and SKE instructions.

Examples **1.** The counter register (T0) value of the timer/event counter 0 is checked if it 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     A, @HL      ; A = (HL)?
        BR      NO
        INCS    L
        MOV     A, X        ; A ← X
        SKE     A, @HL      ; A = (HL)?
```

2. The 00H to FFH data memory is cleared to 0.

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

(6) Bit manipulation addressing

This addressing mode enables to carry out bit manipulations (BOOLEAN processing, bit transfer, etc.) for each bit in the whole data memory space.

While the 1-bit direct addressing mode can only be used for the bit set, reset and test instructions, the bit manipulation addressing mode enables to carry out many bit manipulations including BOOLEAN processing by the AND1, OR1 and XOR1 instructions and test & reset by the SKTCLR instruction.

The following three bit manipulation addressing modes are available to be selected according to the data memory address in use.

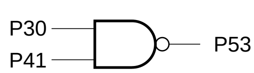
(a) Specific address bit direct addressing (fmem. bit)

This addressing mode enables to operate the peripheral hardware frequently executing bit manipulations, such as the input/output ports and interrupt related flags, irrespectively of memory bank setting. Thus, the memory addresses usable for this addressing mode are FF0H to FFFH at which input/output ports are mapped and FB0H to FBFH at which interrupt related hardware is mapped. The hardware located in these two data memory areas can carry out bit manipulation by direct addressing irrespectively of MBS and MBE settings.

Examples 1. The timer 0 interrupt request flag (IRQT0) is tested. If the flag has been set, it is cleared and P63 is reset.

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

2. If both P30 and P41 are 1, P53 is reset.




```
(i)             SET1    CY             ; CY ← 1
                AND1    CY, PORT3. 0   ; CY ^ P30
                AND1    CY, PORT4. 1   ; CY ^ P41
                SKT     CY             ; CY = 1?
                BR      SETP
                CLR1    PORT5. 3       ; P53 ← 0
                  :
        SETP:   SET1    PORT5. 3       ; P53 ← 1
                  :
(ii)            SKT     PORT3. 0       ; P30 = 1?
                BR      SETP
                SKT     PORT4. 1       ; P41 = 1?
                BR      SETP
                CLR1    PORT5. 3       ; P53 ← 0
                  :
        SETP:   SET1    PORT5. 3       ; P53 ← 1
```

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

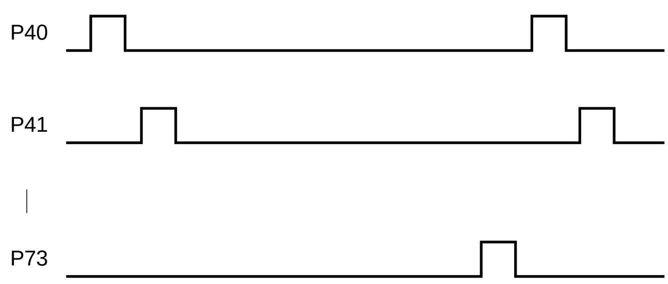
This addressing mode enables to continuously operate each bit of the input/output port among the peripheral hardware by indirect register addressing. This addressing mode can be used for the FC0H to FFFH data memory addresses.

In this addressing mode, the high-order 10 bits of the 12 bits of the data memory address are directly specified by an operand and the low-order 2-bit address and the bit address are indirectly specified using the L register.

Thus, 16 bits (4 ports) can be continuously operated by L register specification.

This addressing mode also enables to carry out bit manipulation irrespectively of MBE and MBS settings.

Example Pulses are sequentially output to the bits of ports 4 to 7.



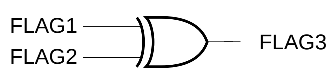

```
        MOV     L, #0
LOOP:   SET1    PORT4. @L   ; Bits (L1–0) of ports 4 to 7 ← 1
        CLR1    PORT4. @L   ; Bits (L1–0) of ports 4 to 7 ← 0
        INCS    L
        NOP
        BR      LOOP
```

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

This addressing mode enables to carry out bit manipulation for the bits in the whole data memory space.

In this addressing mode, the high-order 4 bits of the data memory address of the memory bank specified by MB = MBE•MBS are indirectly specified using the H register and the low-order 4-bit address and the bit address are directly specified by an operand. This addressing mode enables to carry out many bit operations for the bits in the whole data memory space.

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




```
FLAG1   EQU    30H. 3
FLAG2   EQU    31H. 0
FLAG3   EQU    32H. 2
        SEL    MB0
        MOV    H, #FLAG1 SHR 6
        CLR    CY,              ; CY ← 0
        OR1    CY, @H+FLAG1     ; CY ← CY ∨ FLAG1
        XOR1   CY, @H+FLAG2     ; CY ← CY ⊻ FLAG2
        SET1   @H+FLAG3         ; FLAG3 ← 1
        SKT    CY               ; CY = 1?
        CLR1   @H+FLAG3         ; FLAG3 ← 0
```

(7) Stack addressing

This addressing mode enables to carry out save/restore operations during interrupt or subroutine servicing. In this mode, the address indicated by the stack pointer (8 bits) of data memory bank 0 is specified. This addressing mode is also used for register save/restore operations by the PUSH/POP instruction.

Examples **1.** The register is saved/restored by subroutine processing.

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

2. HL register pair contents are transferred to the DE register pair.

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

3. Data is branched to the address specified by the [XABC] register.

```
        PUSH    BC
        PUSH    XA
        RET             ; branched to the address XABC
```

3.2 Memory Mapped I/O

As shown in Figure 3-2, the μPD75308 has a memory mapped I/O with the peripheral hardware such as input/output ports and timers mapped at addresses F80H to FFFH in the data memory space. Thus, memory manipulation instructions are used instead of special instructions to control the peripheral hardware. (Hardware control mnemonic is partly used to facilitate the understanding of the program.)

The addressing modes listed in Table 3-1 are available to operate the peripheral hardware. The display data memory mapped at addresses 1E0H to 1FFH is operated by specifying memory bank 1.

Table 3-1. Addressing Modes Applicable for Peripheral Hardware Operations

	Applicable Addressing Mode	Applicable Hardware
Bit manipulation	Direct addressing mode to be specified by 'mem. bit' with MBE = 0 or (MBE = 1, MBE = 15)	All hardware devices capable of carrying out bit manipulations
	Direct addressing mode to be specified by 'fmem. bit' irrespectively of MBE and MBS settings	IST0, MBE IExxx, IRQxxx, PORTn. x
	Indirect addressing mode to be specified by 'pmem. @L' irrespectively of MBE and MBS settings	BSBn. x PORTn. x
4-bit manipulation	Direct addressing mode to be specified by 'mem' with MBE = 0 or (MBE = 1, MBS = 15)	All hardware devices capable of carrying out 4-bit manipulations
	Register indirect addressing mode to be specified by '@HL' with (MBE = 1, MBS = 15)	
8-bit manipulation	Direct addressing mode to be specified by 'mem (= even address)' with MBE = 0 or (MBE = 1, MBS = 15)	All hardware devices capable of carrying out 8-bit manipulations
	Register indirect addressing mode to be specified by '@HL (with the L register content set to even number)' with MBE = 1 and MBS = 15	

Example

```
CLR1    MBE         ; MBE = 0
SET1    TM0. 3      ; Timer 0 start
EI      IE0         ; INT0 enable
DI      IE1         ; INT1 disable
SKTCLR  IRQ2        ; INT2 request flag test and clear
SET1    PORT4. @L   ; Port 4 set
MOV     X, #0       ; X = 0 is set
IN      A, PORT0    ; A ← port 0
OUT     PORT4, XA   ; Port 5, 4 ← XA
```

Figure 3-5 shows the I/O map of μPD75308.

Each item in the figure has the following meaning.

- Symbol : Name indicating the address of on-chip hardware
  Can be described in the instruction operand column.
- R/W : Indicates if the corresponding hardware is read/write enable.
  R/W : Read/write enable
  R : Only read enable
  W : Only write enable
- No. of manipulatable bits
  : Indicates the number of bits which can be operated when the corresponding hardware is operated.
  ○ : Bit manipulation enabled in 1, 4 or 8-bit units specified in the column
  △ : Operation enable for part of bits.
  Refer to the “Remarks” column for bits which can be operated.
  – : Bit manipulation disabled in 1, 4 or 8-bit units specified in the column
- Bit manipulation addressing
  : Indicates the applicable bit manipulation addressing mode for bit manipulations of the corresponding hardware.

Figure 3-5. μPD75308 I/O Map (1/3)

Address	Hardware Name (Symbol)				R/W	No. of Manipulatable Bits			Bit Manipulation	Remarks
	b3	b2	b1	b0		1 Bit	4 Bits	8 Bits	Addressing	
F80H	Stack pointer (SP)				R/W	–	–	○		Bit 0 fixed to 0
F85H	Basic interval timer mode register (BTM)				W	△	○	–	mem. bit	Only bit 3 can be operated
F86H	Basic interval timer (BT)				R	–	–	○		
F8CH	Display mode register (LCDM)				W	△	–	○	mem. bit	Only bit 3 can be operated
						–	–			
F8EH	Display control register (LCDC)				W	–	○	–		

Address	Hardware Name (Symbol)				R/W	1 Bit	4 Bits	8 Bits	Bit Manipulation Addressing	Remarks
F98H	Timer mode register (WM)				R/W	△ (R)	–	○	mem. bit	Only bit 3 can be tested Note 1
						–	–	(W)		

Address	Hardware Name (Symbol)				R/W	1 Bit	4 Bits	8 Bits	Bit Manipulation Addressing	Remarks
FA0H	Timer/event counter 0 mode register (TM0)				W	△	–	○	mem. bit	Only bit 3 can be operated
						–	–			
FA2H	TOE0 Note 2				W	○	–	–	mem. bit	
FA4H	Timer/event counter 0 count register (T0)				R	–	–	○		
FA6H	Timer/event counter 0 modulo register (TMOD0)				W	–	–	○		

Notes 1. 1-bit manipulation: R only; 8-bit manipulation: W only

2. TOE0 ... Timer/event counter 0 output enable flag

Figure 3-5. μPD75308 I/O Map (2/3)

Address	Hardware Name (Symbol)				R/W	No. of Manipulatable Bits			Bit Manipulation	Remarks
	b3	b2	b1	b0		1 Bit	4 Bits	8 Bits	Addressing	
FB0H	0	IST0	MBE	0	R/W	○ (R/W)	○ (R/W)	○	fmem.bit	8-bit manipulation: R Only
	Program status word (PSW)									
	CY	SK2	SK1	SK0		–	–	(R)		
FB2H	(IME)				–	–	–	–		Operation by EI/DI instruction
FB3H	Processor clock control register (PCC)				W	–	○			**Note**
FB4H	INT0 mode register (IM0)				W	–	○	–		Bit 2 fixed to 0
FB5H	INT1 mode register (IM1)				W	–	○			Bits 3, 2, 1 fixed to 0
FB6H	INT2 mode register (IM2)				W	–	○			Bits 3, 2 fixed to 0
FB7H	System clock control register (SCC)				W	○	–			Bits 2, 1 fixed to 0
FB8H	IE4	IRQ4	IEBT	IRQBT	R/W	○	○	–	fmem. bit	
FBAH			IEW	IRQW	R/W	○	○			
FBCH			IET0	IRQT0	R/W	○	○	–		
FBDH			IECSI	IRQCSI	R/W	○	○			
FBEH	IE1	IRQ1	IE0	IRQ0	R/W	○	○	–		
FBFH			IE2	IRQ2	R/W	○	○			

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

FD0H	Clock output mode register (CLOM)	W	–	○	–		
FDCH	Pull-up resistor specify register group A (POGA)	W	–	–	○		

Note Bits 3 and 2 can be manipulated at STOP/HALT instruction execution.

Remarks **1.** IExxx is an interrupt enable flag.
2. IRQxxx is an interrupt request flag.
3. IME is an interrupt master enable flag.

Figure 3-5. μPD75308 I/O Map (3/3)

Address	Hardware Name (Symbol)				R/W	No. of Manipulatable Bits			Bit Manipulation	Remarks
	b3	b2	b1	b0		1 Bit	4 Bits	8 Bits	Addressing	
FE0H	Serial operation mode register (CSIM)				R/W	–	–	○		
	CSIE	COI	WUP			△ (R) (W)	○	(W)	mem. bit	Bits 3, 2, 1 are bit-manipulatable[Note 1]
FE2H	CMDD	RELD	CMDT	RELT	R/W	○	–	–	mem. bit	R or W varies in every bits.
	SBI control register (SBIC)									
	BSYE	ACKD	ACKE	ACKT						
FE4H	Serial I/O shift register (SIO)				R/W	–	–	○		
FE6H	Slave address register (SVA)				W	–	–	○		
FE8H	PM33	PM32	PM31	PM30	W	–	–	○		
	Port mode register group A (PMGA)									
	PM63	PM62	PM61	PM60						
FECH	–	PM2	–	–	W	–	–	○		
	Port mode register group B (PMGB)									
	PM7	–	PM5	PM4						
FF0H	Port 0 (PORT 0)				R	○	○	–	fmem. bit	
FF1H	Port 1 (PORT 1)				R	○	○		pmem. @L	
FF2H	Port 2 (PORT 2)				R/W	○	○	–		
FF3H	Port 3 (PORT 3)				R/W	○	○			
FF4H	Port 4 (PORT 4)				R/W	○	○	○		
FF5H	Port 5 (PORT 5)				R/W	○	○			
FF6H[Note 2]	KR3	KR2	KR1	KR0	R/W	○	○	○		
	Port 6 (PORT 6)									
FF7H[Note 2]	KR7	KR6	KR5	KR4	R/W	○	○			
	Port 7 (PORT 7)									

Notes 1. In bit manipulation, R or W varies from bit to bit.
8-bit manipulation: W only

2. KR0 to KR7 are read only. They can be specified by PORT 6 or PORT 7 in the 4-bit parallel input mode.

CHAPTER 4 INTERNAL CPU FUNCTION

4.1 Program counter (PC):
12 bits (μPD75304, 75304B)
13 bits (μPD75306, 75306B, 75308, 75308B, 75P308)
14 bits (μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B)

The PC is a binary counter to hold the program memory address. The μPD75304, 75304B has a 12-bit configuration (refer to **Figure 4-1 (a)**). The μPD75306, 75306B, 75308, 75308B and 75P308 each have a 13-bit configuration (refer to **Figure 4-1 (b)**). The μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A and 75P316B each have a 14-bit configuration (refer to **Figure 4-1 (c)**).

Figure 4-1. Program Counter Configuration

(a) μPD75304, 75304B

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

(b) μPD75306, 75306B, 75308, 75308B, 75P308

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

(c) μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B

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

The program counter value is automatically incremented according to the number of bytes of the instruction each time an instruction is executed.

When a branch instruction (BR, BRCB) is executed, immediate data or register pair content indicating the branch point address is loaded into all or some bits in the PC.

When a subroutine call instruction (CALL, CALLF) is executed of a vectored interrupt is generated, the PC content (return address incremented to fetch the next instruction) is saved into the stack memory (data memory specified by the stack pointer) and the address in the jump destination is loaded.

When a return instruction (RET, RETS, RETI) is executed, the stack memory content is set into the PC.

When the $\overline{\text{RESET}}$ signal is generated, the program memory content is loaded into the program counter for initialization. Thus, the program can be started at any selected address.

μPD75304, 75304B:
- PC11 to PC8 ← Low-order 4 bits of address 000H
- PC7 to PC0 ← 8 bits of address 001H

μPD75306, 75306B, 75308, 75308B, 75P308:
- PC12 to PC8 ← Low-order 5 bits of address 0000H
- PC7 to PC0 ← 8 bits of address 0001H

μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B:
- PC13 to PC8 ← Low-order 6 bits of address 0000H
- PC8 to PC0 ← 8 bits of address 0001H

4.2 Program Memory (ROM):
4096 words x 8 bits (μPD75304, 75304B)
6016 words x 8 bits (μPD75306, 75306B)
8064 words x 8 bits (μPD75308, 75308B, 75P308)
12160 words x 8 bits (μPD75312, 75312B)
16256 words x 8 bits (μPD75316, 75316B, 75P316, 75P316A, 75P316B)

The program memory is used to store the program, interrupt vector table, GETI instruction reference table and table data. The mask programmable ROM serves as the program memory for the μPD75304, 75304B, 75306, 75306B, 75308, 75308B, 75312, 75312B, 75316, 75316B. The PROM serves as the program memory for the μPD75P308, 75P316, 75P316A and 75P316B.

Figures 4-2 to 4-6 show the program memory maps.

Addressing is performed by the program counter. Table data can be referred to by the table reference instruction (MOVT).

The address ranges which can be branched by the branch instruction and the subroutine call instruction are shown in Figures 4-2 to 4-6. The relative branch instruction (BR $addr) enables to branch to the [PC content –15 to –1, +2 to +16] addresses, irrespectively of block.

The program memory address ranges are as follows:

- 000H to FFH : μPD75304, 75304B
- 0000H to 177FH : μPD75306, 75306B
- 0000H to 1F7FH : μPD75308, 75308B, 75P308
- 0000H to 2F7FH : μPD75312, 75312B
- 0000H to 3F7FH : μPD75316, 75316B, 75P316, 75P316A, 75P316B

Special functions are assigned for the following addresses. All areas except 0000H to 0001H (000H to 001H[Note 1]) can be used as the normal program memory.

- 0000H to 0001H (000H to 001H[Note 1])
  Vector address table for writing the program status address and MBE set value when resetting. (Reset can be started from any selected address.)
- 0002H to 000BH (002H to 00BH[Note 1])
  Vector address table for writing the program start address and MBE set value by each vector interrupt. (Interrupt servicing can be started from any selected address.)
- 0020H to 007FH (020H to 07FH[Note 1])
  Table area which is referred to by GETI instruction[Note 2].

Notes 1. In the case of μPD75304, 75304B.

2. GETI instruction is an instruction to realize any 2-byte/3-byte instruction or two 1-byte instructions with one byte, and can reduce the number of program bytes.

Figure 4-2. Program Memory Map (μPD75304, 75304B)

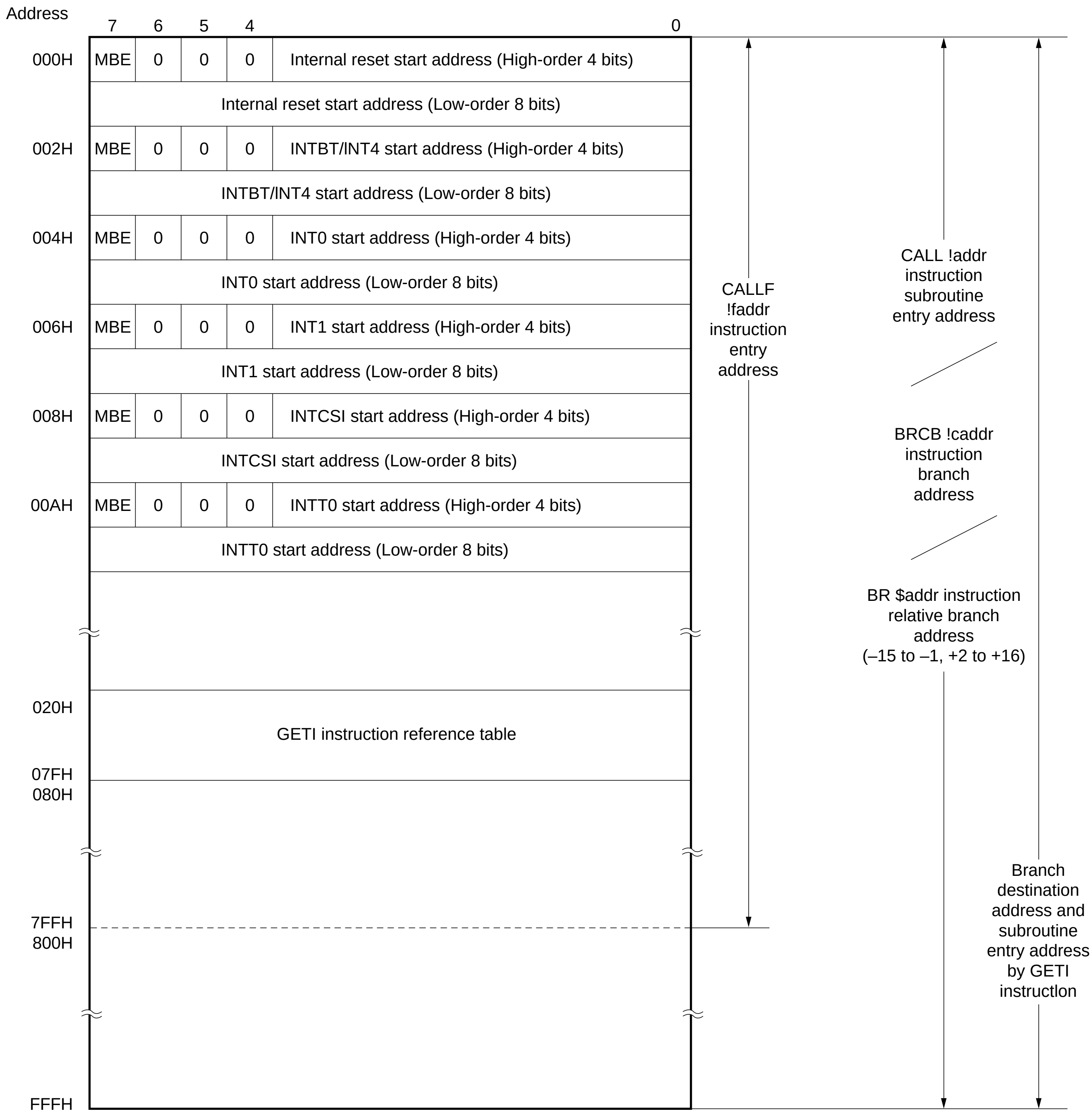


Figure 4-3. Program Memory Map (μPD75306, 75306B)

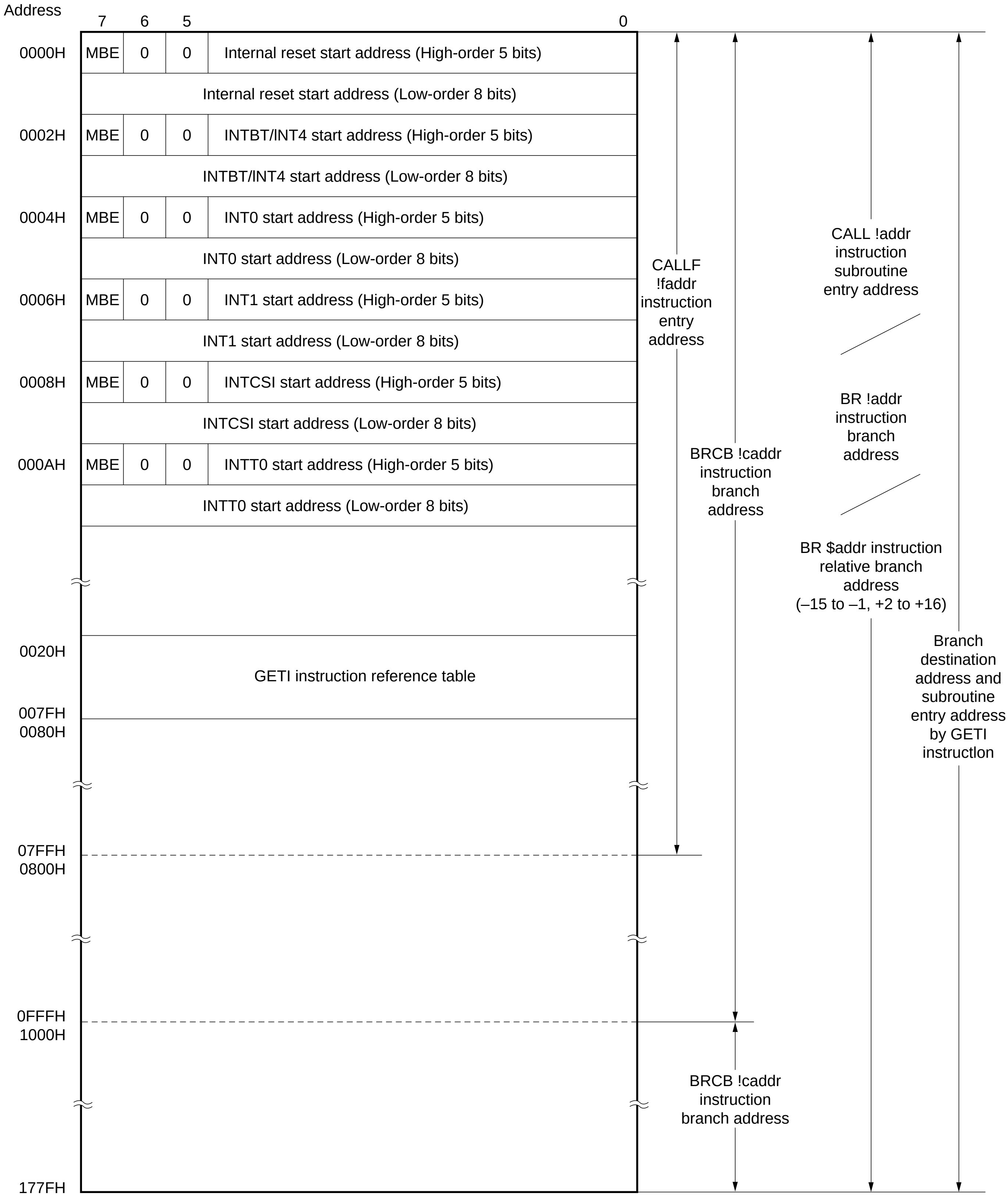


Figure 4-4. Program Memory Map (μPD75308, 75308B, 75P308)

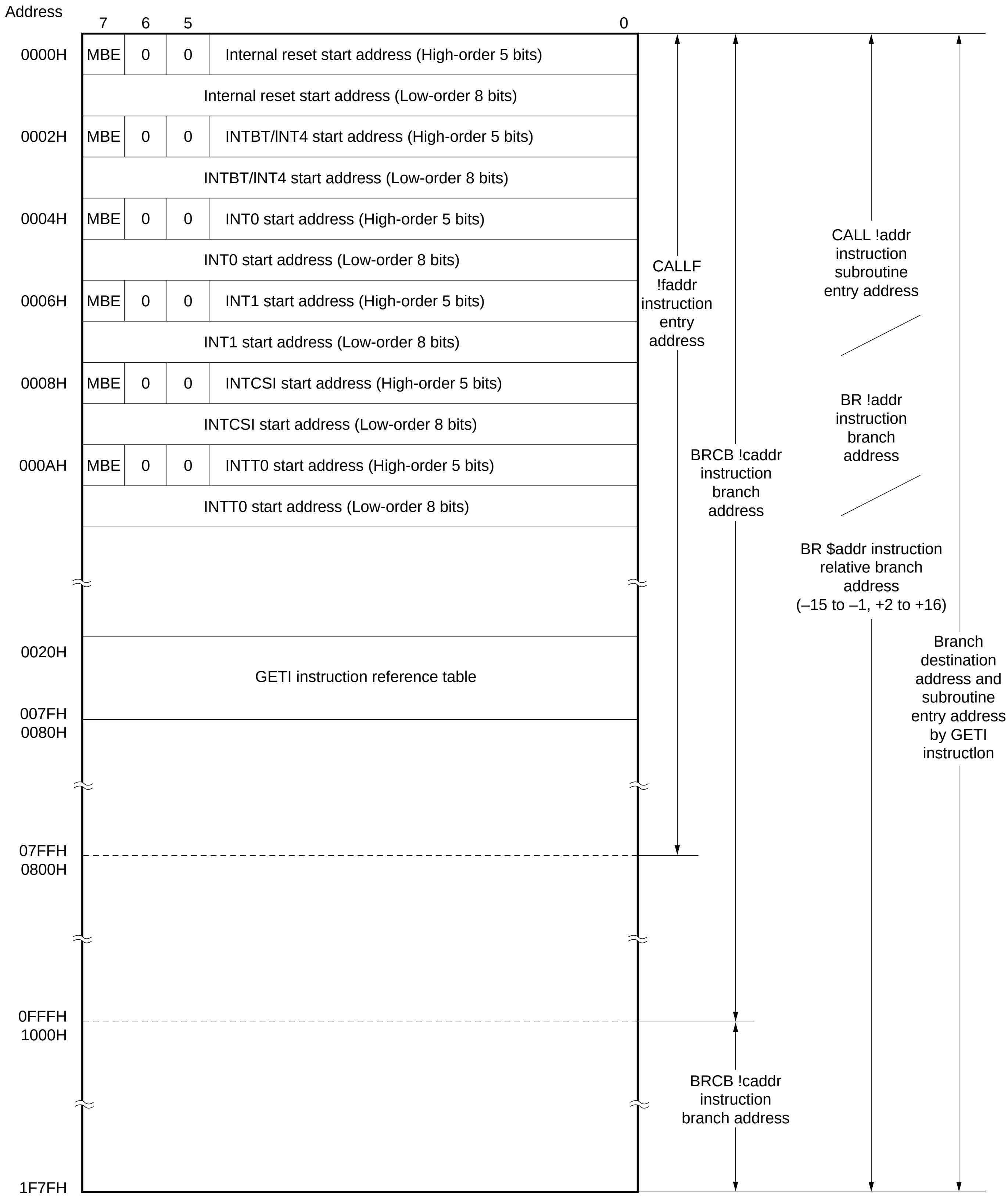


Figure 4-5. Program Memory Map (μPD75312, 75312B)

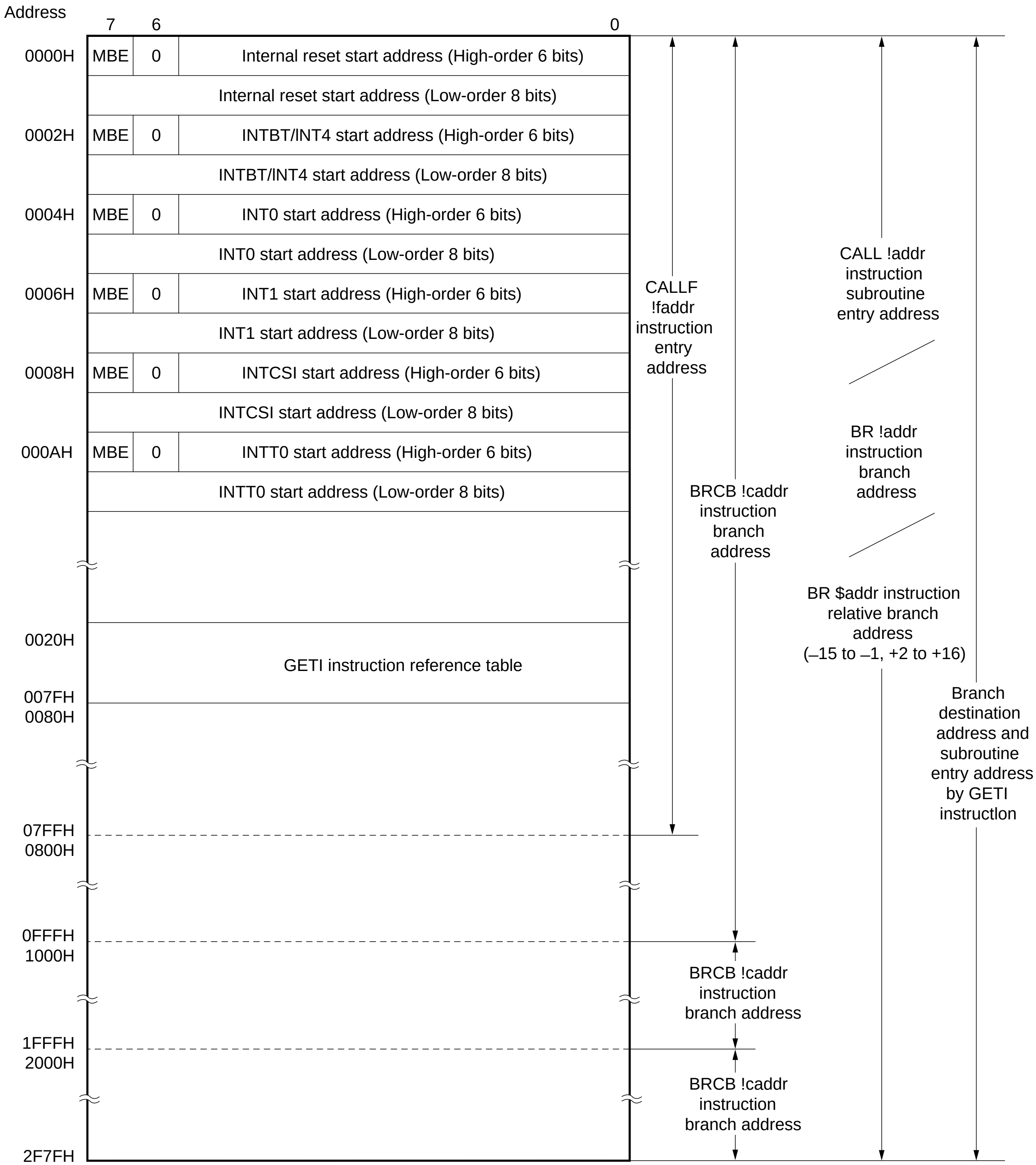
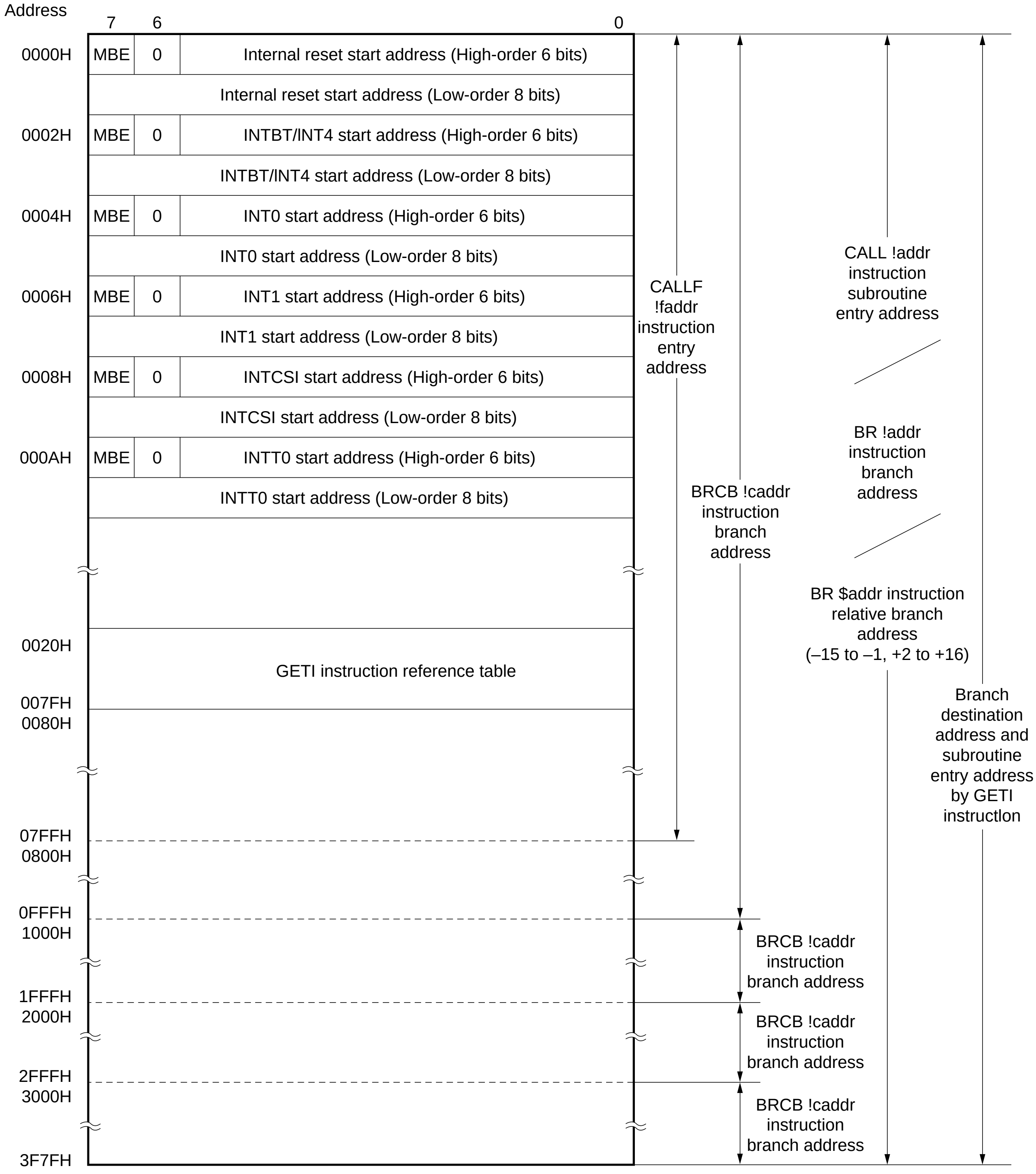


Figure 4-6. Program Memory Map (μPD75316, 75316B, 75P316, 75P316A, 75P316B)



4.3 Data Memory (RAM):

512 words x 4 bits (μPD75304, 75304B, 75306, 75306B, 75308, 75308B, 75P308, 75312, 75316, 75P316)

1024 words x 4 bits (μPD75312B, 75316B, 75P316A, 75P316B)

The data memory consists of a data area and a peripheral hardware area as shown in Figure 4-7.

The data memory has a bank configuration, with one bank consisting of 256 words x 4 bits. The memory banks are as follows:

- Memory banks 0 and 1 (data area)Note
- Memory bank 15 (peripheral hardware area)

Note Memory banks 0, 1, 2 and 3 on the μPD75312B, 75316B, 75P316A, 75P316B only.

4.3.1 Data memory configuration

(1) Data area

The data area comprises static RAM and is used for data storage and as stack memory during execution of subroutines and interrupts. Long-term retention of memory contents is possible even on battery backup power when CPU operation is halted in standby mode. Manipulation is by means of memory manipulation instructions.

A 256 x 4-bit area of static RAM is mapped onto each of memory banks 0 and 1Note 1. Bank 0 is mapped as a data area, but can also be used as a general register area (000H to 007H) and a stack memory area (000H to 0FFH). Bank 1 is used as display data memory (1E0H to 1FFH).

Static RAM has a 4-bit address configuration. However, it is possible to manipulate in 8-bit units using 8-bit memory manipulation instructions, and bit-wise using bit manipulation instructionsNote 2. In 8-bit manipulation instructions, an even address should be specified.

Notes 1. Memory banks 0, 1, 2 and 3 on the μPD75312B, 75316B, 75P316A and 75P316B only.

2. 8-bit manipulation is not possible on display data memory.

- **General register area**

  This area can be manipulated by either general register manipulation instructions or memory manipulation instructions. Up to eight 4-bit registers can be used. Any of the general registers not used by the program can be used as data area or stack area memory.

- **Stack memory area**

  The stack area is set up by an instruction, and can be used as a save area during execution of a subroutine or interrupt servicing.

- **Display data memory area**

  This is the area to which LCD display data is written. When driving and LCD, the data written to this display data memory area is automatically read by hardware and then displayed. Any portion not used for display can be used as part of the data area.

(2) Peripheral hardware area

The peripheral hardware area is mapped onto addresses F80H through FFFH of memory bank 15.

This area is manipulated by memory manipulation instructions in the same way as static RAM. With peripheral hardware, however, the bit units which can be manipulated vary from address to address. As data memory is not incorporated on chip for addresses not allocated to peripheral hardware, they cannot be accessed.

4.3.2 Data memory bank specification

The 4-bit memory bank selection register (MBS) is used to specify the memory bank (MBS = 0, 1 or 15[Note]) when the memory bank enable flag (MBE) specifies that bank selection is enabled (MBE = 1). When bank specification is disabled (MBS = 0), the memory bank is automatically specified as either bank 0 or bank 15 according to the current addressing mode. Addresses in the bank are addressed by 8-bit immediate data or a register pair, etc.

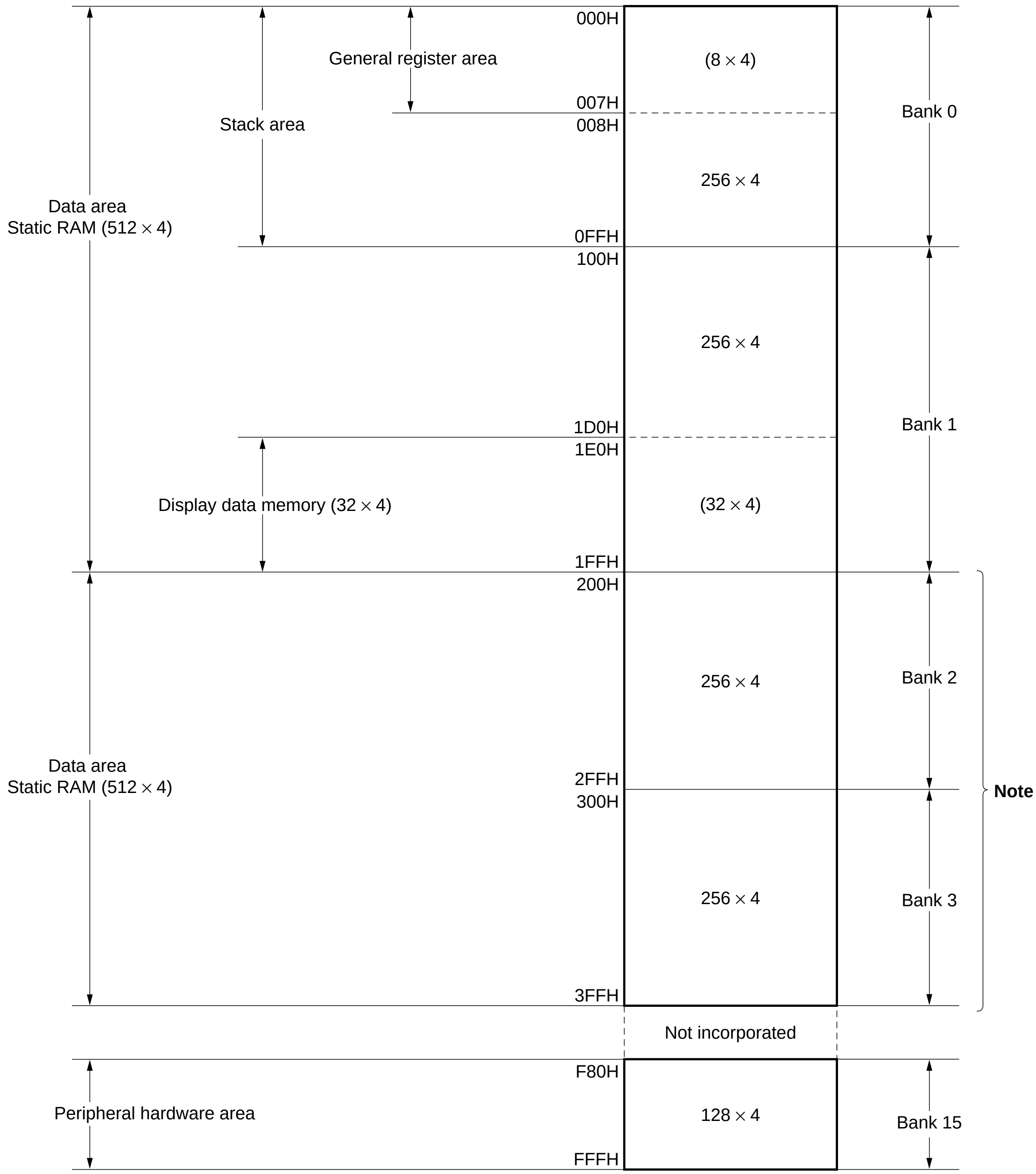
For details of memory bank selection and addressing, refer to **3.1 Data Memory Bank Configuration and Addressing Mode**.

For the use of specific areas in the data memory, please refer to the following sections.

- General register area .... **4.4 General Register**
- Stack memory area .... **4.6 Stack Pointer (SP)**
- Display data memory .... **5.7.5 Display data memory**
- Peripheral hardware .... **CHAPTER 5 PERIPHERAL HARDWARE FUNCTIONS**

Note MBS = 0, 1, 2, 3 or 15 on the μPD75312B, 75316B, 75P316A and 75P316B only.

Figure 4-7. Data Memory Map



Note The μPD75312B, 75316B, 75P316A, and 75P316B only

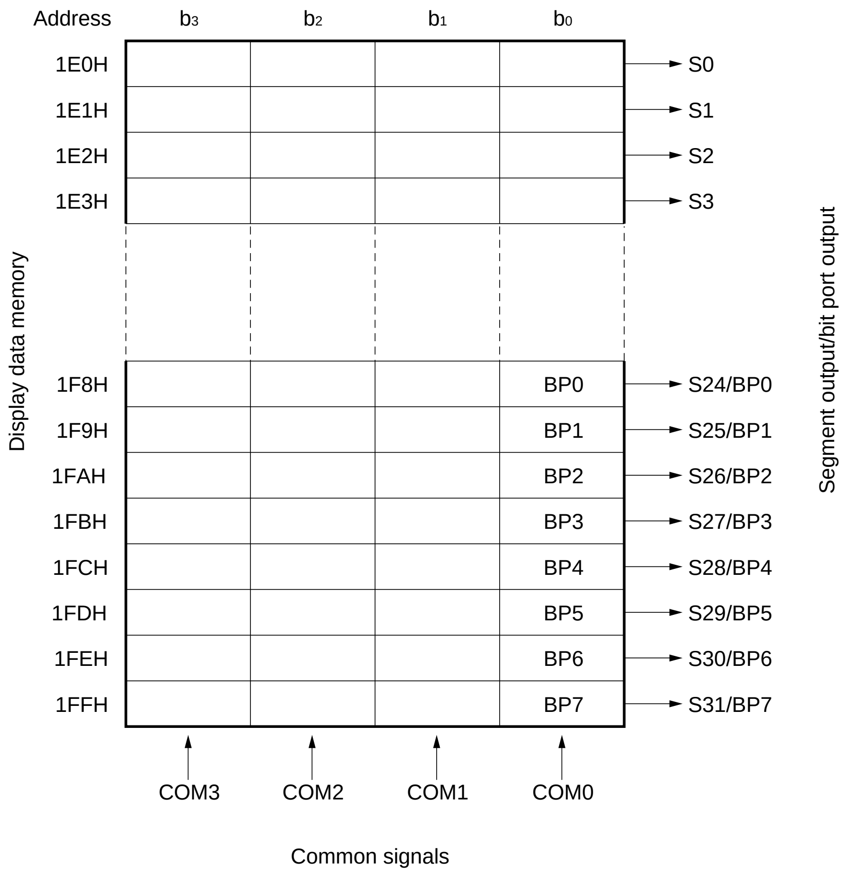
The data memory is indeterminate when reset. Thus, initialize it to zero at the beginning of the normal program (RAM clear). Make sure to do so, or bugging may result.

Example The RAM at addresses 000H to 1FFH is cleared.

```
        SET1    MBE
        SEL     MB0
        MOV     XA, #00H
        MOV     HL, #04H
RAMC0:  MOV     @HL, A          ; 04H to FFH clear[Note]
        INCS    L               ; L ← L + 1
        BR      RAMC0
        INCS    H               ; H ← H + 1
        BR      RAMC0
        SEL     MB1
RAMC1:  MOV     @HL, A          ; 100H to 1FFH clear
        INCS    L               ; L ← L + 1
        BR      RAMC1
        INCS    H               ; H ← H + 1
        BR      RAMC1
```

Note Since the data memory at addresses 000H to 003H is used as the general register XA or HL, it is not cleared to zero.

Figure 4-8. Display Data Memory Configuration



The display data memory is manipulated in 1 or 4-bit units.

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

Example The 1E0H to 1FFH display data memory is cleared.

```
        SET1    MBE
        SEL     MB1
        MOV     HL, #0E0H
        MOV     A, #00H
LOOP:   MOV     @HL, A       ; The display data memory is cleared to 0 all at once
                               in 4-bit units.
        INCS    L
        BR      LOOP
        INCS    H
        BR      LOOP
```

4.4 General Register: 8 x 4 bits

Eight 4-bit general registers (B, C, D, E, H, L, X, A) are mapped at the specific addresses of the data memory.

Each general register is operated in 4-bit units.

BC, DE, HL and XA make up register pairs to be used for 8-bit operations. DL also makes up a register pair and three pairs, DE, HL and DL, can be used as the data pointer.

The general register area can be accessed by addressing as a normal RAM whether it is used as a register or not.

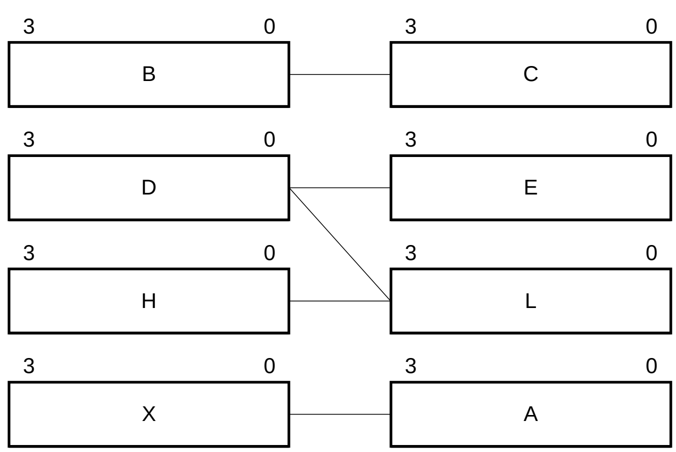
Figure 4-9. General Register Configuration (For 4-Bit Processing)

X	01H	A	00H
H	03H	L	02H
D	05H	E	04H
B	07H	C	06H

Figure 4-10. General Register Configuration (For 8-Bit Processing)

XA	00H
HL	02H
DE	04H
BC	06H

Figure 4-11. Register Pair Configuration

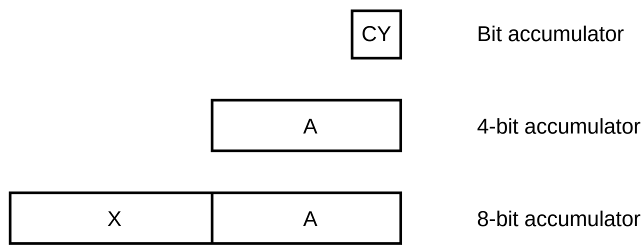


4.5 Accumulator

The μPD75308, the A register and the XA register pair function as accumulators. 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.

For execution of bit manipulation instructions, the carry flag (CY) functions as the bit accumulator.

Figure 4-12. Accumulator



4.6 Stack Pointer (SP): 8 bits

The μPD75308 has a data area which functions as the stack memory (LIFO format). The 8-bit register which holds the first address information of the stack area is the stack pointer (SP).

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

The SP is decremented ahead of write (save) operation for the stack memory and is incremented after read (restore) operation from the stack memory.

Figures 4-14 to 4-16 show the data to be saved/restored by each stack operation.

The SP sets the initial value by the 8-bit memory manipulation instruction and determines the stack area. It can also read the contents from the stack area.

"0" is always set for SP0.

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

Since the SP content becomes indeterminate when the $\overline{\text{RESET}}$ signal is generated, initialize the SP to the desired value at the beginning of the program.

Example SP initialize

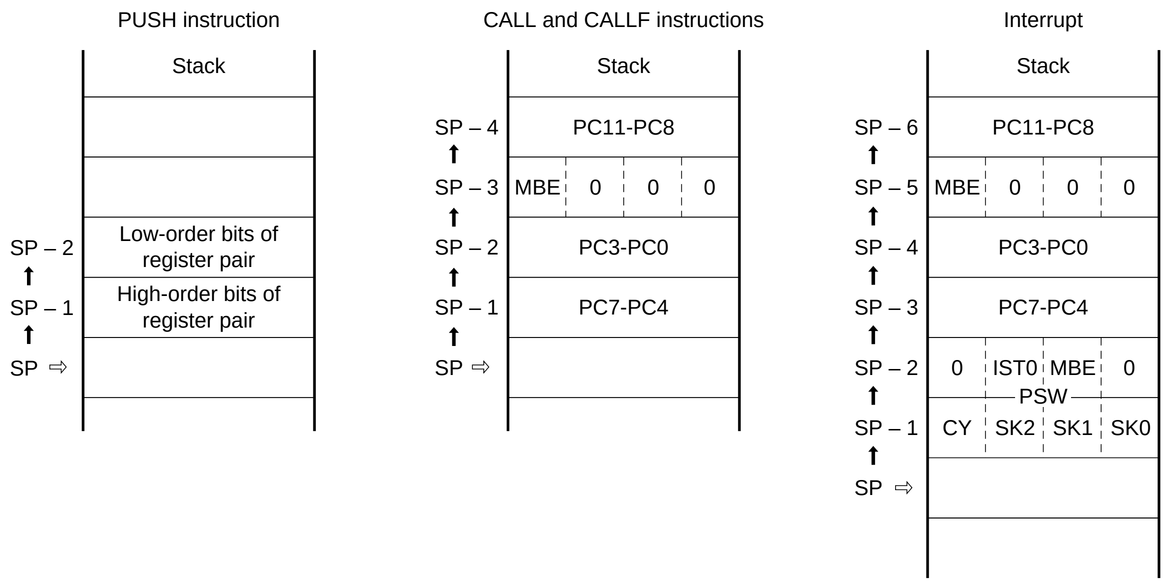
```
SEL   MB15      ; or CLR1 MBE
MOV   XA, #00H
MOV   SP, XA    ; SP ← 00H
```

Figure 4-13. Stack Pointer Format

Address	7	6	5	4	3	2	1	0	Symbol
F80H	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SP
								Fixed to 0	

Figure 4-14. Data Saved/Restored by Stack Operation (μPD75304, 75304B)

(a) Data Saved into Stack Memory



(b) Data Restored from Stack Memory

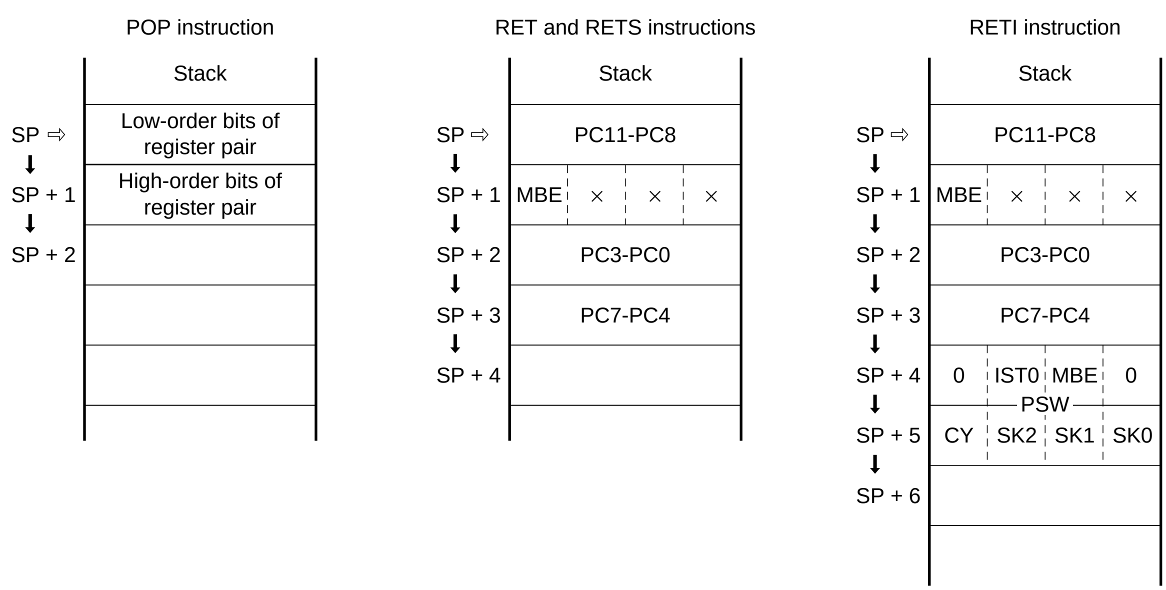
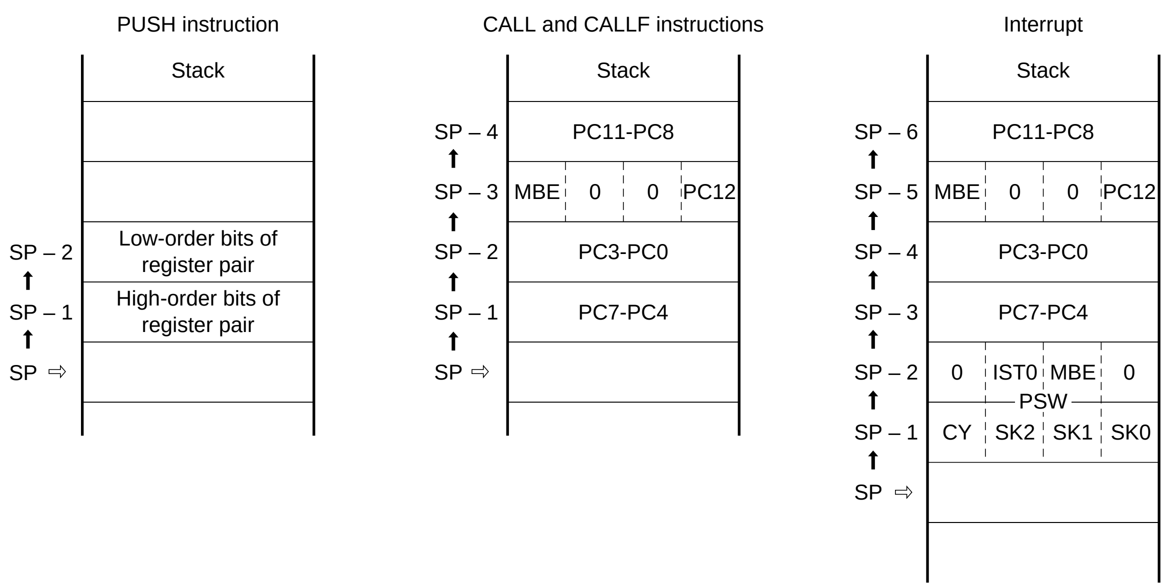


Figure 4-15. Data Saved/Restored by Stack Operation (μPD75306, 75306B, 75308, 75308B, 75P308)

(a) Data Saved into Stack Memory



(b) Data Restored from Stack Memory

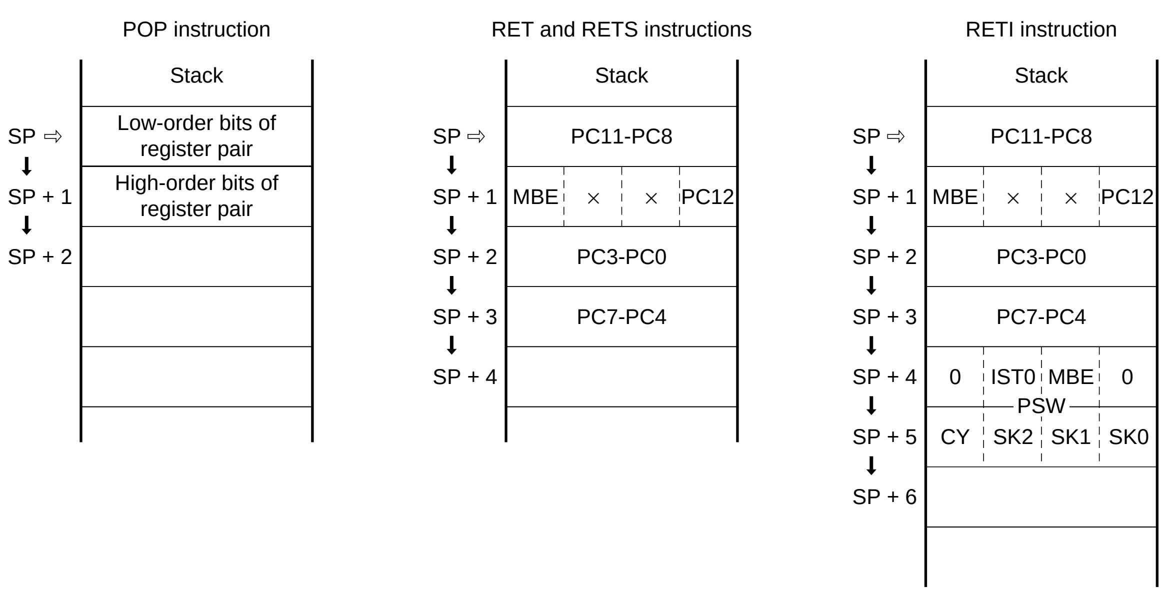
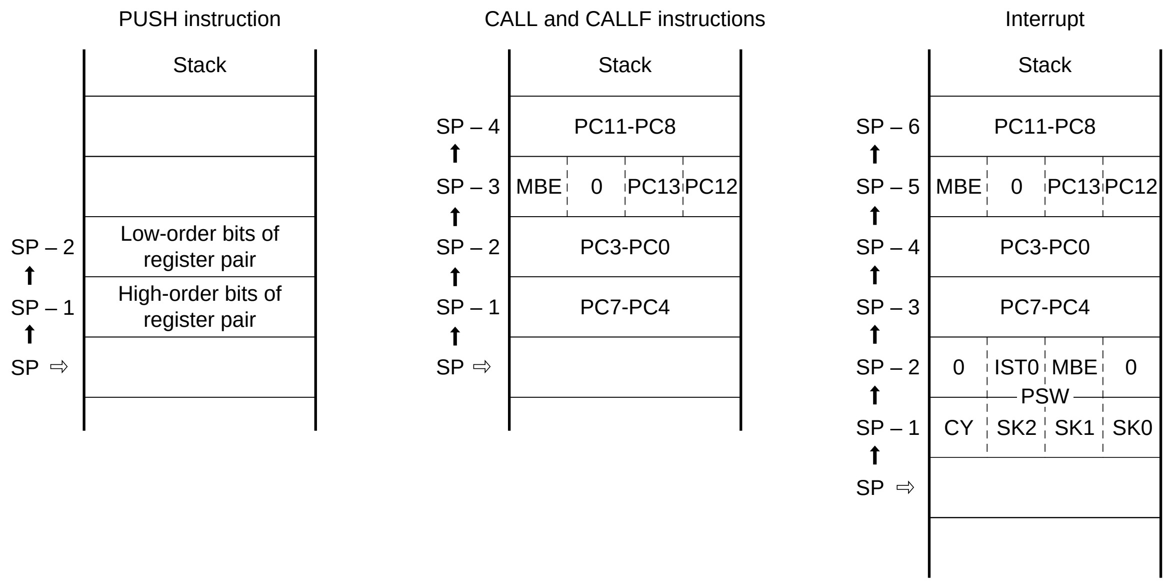
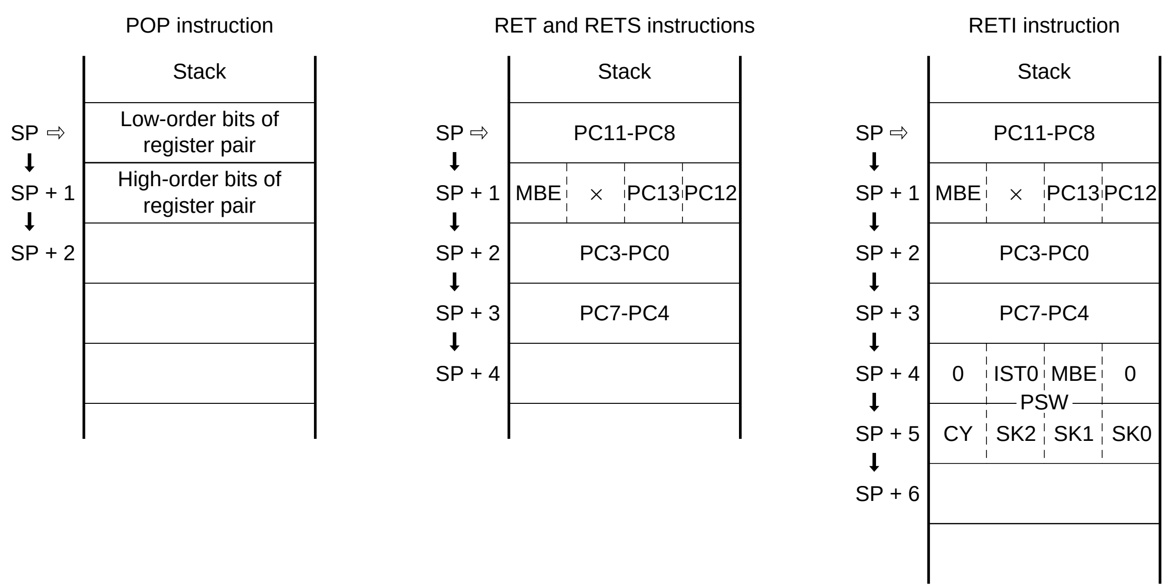


Figure 4-16. Data Saved/Restored by Stack Operation (μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A and 75P316B)

(a) Data Saved into Stack Memory



(b) Data Restored from Stack Memory



4.7 Program Status Word (PSW): 8 bits

The program status word (PSW) is configured of various flags closely related to processor operations.

The PSW is mapped at addresses FB0H and FB1H in the data memory space and two bits at address FB0H can be operated by a memory manipulation instruction.

Figure 4-17. Program Status Word Format

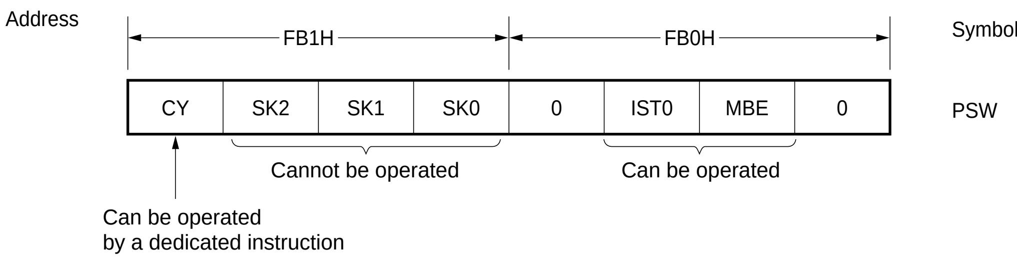


Table 4-1. PSW Flag Saved/Restored by Stack Operation

		Flag Saved/Restored
Save	When CALL and CALLF instructions are executed	MBE is saved
	When the hardware is interrupted	All PSW bits are saved
Restore	When RET and RETS instructions are executed	MBE is restored
	When RETI instruction is executed	All PSW bits are restored

(1) Carry flag (CY)

The carry flag is a 1-bit flag to store overflow/underflow occurrence information during execution of carry operation instructions (ADDC, SUBC).

The carry flag performs the bit accumulator function which enables to carry out Boolean logic operations with the bit address specified data memory and to store the operation results.

Carry flag operations are carried out using a dedicated instruction irrespectively of other PSW bits. The carry flag becomes indeterminate if the $\overline{\text{RESET}}$ signal is generated.

Table 4-2. Carry Flag Operation Instructions

	Instruction (Mnemonic)	Carry Flag Operation and Processing
Carry flag operation dedicated instruction	SET1 CY CLR1 CY NOT1 CY SKT CY	CY set (1) CY clear (0) CY content inverse Skip if CY content is 1
Bit Boolean instruction	AND1 CY, mem*.bit OR1 CY, mem*.bit XOR1 CY, mem*.bit	Specified bit content is AND/OR/XORed with CY content and the result is set in CY.
Interrupt servicing	When interrupt servicing is executed	Another PSW bit and 8 bits are saved in parallel into the stack memory.
	RETI	Restored in parallel with another PSW from the stack memory.

Remark mem*.bit indicates the following three bit operation addressing modes:

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

Example Bit 3 at address 3FH is ANDed with P33 and the result is set in CY.

```
SET1   CY            ; CY ← 1
CLR1   MBE           ; or SEL MB15
SKT    3FH.3         ; skip if bit 3 at address 3FH is 1.
CLR1   CY            ; CY ← 0
AND1   CY, PORT3.3   ; CY ← CY ^ P33
```

(2) Skip flags 0 to 2 (SK0 to SK2)

The skip flag stores the skip state. It is automatically set/reset when the CPU executes an instruction. The user cannot directly operate the skip flag as an operand.

(3) Interrupt status flag (IST0)

The interrupt status flag stores the status of processing currently being executed. (Refer to **Table 6-3 IST0 and Interrupt Servicing Status** for details.)

Table 4-3. Interrupt Status Flag Instruction Content

IST0	Status of Processing Being Executed	Processing Content and Interrupt Control
0	Status 0	All interrupts acknowledge enable during normal program processing
1	Status 1	All interrupts acknowledge disable during interrupt servicing

When an interrupt is acknowledged, the IST0 content is saved into the stack memory as part of PSW and is then set to 1 automatically and is set to 0 by the RETI instruction.
The interrupt status flag can be operated by the memory manipulation instruction and the status of processing being executed can be changed by program control.

Caution When operating this flag, make sure to disable the interrupt by executing the DI instruction and enable the interrupt by executing the EI instruction after operation.

(4) Memory bank enable flag (MBE)

This is a 1-bit flag to specify the address information generate mode for the high-order 4 bits of 12 bits of the data memory address.
The MBE can be set/reset by a bit manipulation instruction irrespectively of memory bank setting.

Example

```
SET1  MBE   ; MBE ← 1
CLR1  MBE   ; MBE ← 0
```

When the MBS is set to 1, the data memory address space is expanded and all data memory spaces become addressable.
When the MBS is set to 0, the data memory address space is fixed irrespectively of MBS (refer to **Figure 3-2 Data Memory Configuration and Addressing Range in Each Addressing Mode**).
When the $\overline{\text{RESET}}$ signal is generated, the content of bit 7 at address 0 of the program memory is set and is automatically initialized.
When the vectored interrupt servicing is carried out, the content of bit 7 of the corresponding vector address table is set and the MBE status in interrupt service is automatically set.
MBE = 0 is usually set in interrupt servicing for use with the data area of the memory bank.

4.8 Bank Select Register (BS)

Memory bank select register (MBS) used for memory bank specification is mapped in the bank select register (BS). The low-order 4 bits are fixed to 0.

The MBS is set by the SEL MBn instruction.

The BS can be saved/restored in 8-bit units in the stack area by the PUSH BS/POP BS instruction.

Figure 4-18. Bank Select Register Format

MBS								Symbol
7	6	5	4	3	2	1	0	
MBS3	MBS2	MBS1	MBS0	0	0	0	0	BS

(1) Memory bank select register (MBS)

The MBS is a 4-bit register to store the high-order 4-bit address information of the data memory address (12 bits). The memory bank to be accessed is specified by the content of this register. Three memory banks, 0, 1 and 15, can be specified.Note 1

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

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

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

Notes 1. On the μPD75312B, 75316B, 75P316A and 75P316B only, there are 5 banks: 0, 1, 2, 3 and 15.

2. On the μPD75312B, 75316B, 75P316A and 75P316B only, n = 0, 1, 2, 3 or 15.

[MEMO]

5.1 Digital Input/Output Port

The μPD75308 has the memory mapped I/O. All input/output ports are mapped in the data memory space.

Bit port outputs BP0 to BP7 at addresses 1F8H to 1FFH function as output latches.

BP0 to BP7 are switched in 4-bit units by bits 6 and 7 of the display mode register (LCDM) (refer to **Figure 5-74. Display Mode Register Format**).

Bits which are not used as output latches by the bit ports at 1F8H to 1FFH are used as display memory or static RAM. Each address can be operated in 1-bit/4-bit units.

Figure 5-1. Digital Port Data Memory Address

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
1F8H	–	–	–	BP0	
1F9H	–	–	–	BP1	
1FAH	–	–	–	BP2	
1FBH	–	–	–	BP3	
1FCH	–	–	–	BP4	
1FDH	–	–	–	BP5	
1FEH	–	–	–	BP6	
1FFH	–	–	–	BP7	

Remark Some addresses can be used as static RAM.

Table 5-2 lists input/output port manipulation instructions. Various types of control operations, including 4-bit input/output, 8-bit input/output and bit operations, can be carried out for PORT4 through PORT7.

BP0 to BP7 are bitwise output ports.

Phase-out/Discontinued

Examples **1.** Depending on the results of P13 status test, different values are output to ports 4 and 5.

```
SKT    PORT1. 3    ; Skip if bit 3 of port 1 is 1.
MOV    XA, #18H    ; XA ← 18H
MOV    XA, #14H    ; XA ← 14H
SEL    MB15        ; or CLR1 MBE
OUT    PORT4, XA   ; Port 5, 4 ← XA
```

2.

```
SET1   PORT4. @L   ; L register specified bit of ports 4 to 7 is set (1)
```

3. 1 is output to BP0.

```
SET1   MBE
SEL    MB1         ; Memory bank 1 is selected.
SET1   BP0         ; BP0 ← 1
```

5.1.1 Types, features and configurations of digital input/output ports

The different kinds of digital input/output ports are shown in Table 5-1.

The configuration of each port is shown in Figures 5-2 through 5-5.

Table 5-1. Digital Port Types and Features

Port (Pin Names)	Function	Operation/Features		Remarks
PORT0 (P00 to P03)	4-bit input	Can always be read or tested regardless of dual-purpose pin operating mode.		Pins have dual purpose as INT4, $\overline{SCK}$, SO/SB0, and SI/SB1.
PORT1 (P10 to P13)				Pins have dual purpose as INT0 to INT2, and TI0.
PORT2 (P20 to P23)	4-bit input/output	Input or output mode can be set in 4-bit units.		Pins have dual purpose as PTO0, PCL, and BUZ.
PORT3[Note 1] (P30 to P33)		Input or output can be set in 1-bit/4-bit units.		Pins have dual purpose as LCDCL, SYNC, and MD0 to MD3[Note 2]
PORT4[Note 1] (P40 to P43)	4-bit input/output (N-ch open-drain, 10 V withstand voltage)	Input or output can be set in 4-bit units.	8-bit data input/output is possible using ports 4 and 5 as a pair.	Bit-wise specification of on-chip pull-up resistor is possible.
PORT5[Note 1] (P50 to P53)				
PORT6 (P60 to P63)	4-bit input/output	Input or output can be set in 1-bit/4-bit units.	8-bit data input/output is possible using ports 6 and 7 as a pair.	Pins have dual purpose as KR0 to KR3.
PORT7 (P70 to P73)		Input or output can be set in 4-bit units.		Pins have dual purpose as KR4 to KR7.
BP0 to BP7	1-bit output	Bit-wise data output. Switchable by software to LCD drive segment outputs S24 to S31.		Drive capability is very small. For CMOS load drive.

Notes 1. Direct LED drive is possible.

2. PORT3 also functions as pins MD0 through MD3 on the μPD75P308, 75P316, 75P316A and 75P316B only.

P10 also serves as the external vector interrupt input pin, functioning an input with a noise eliminator (refer to **6.3 Various Hardware Types of Interrupt Control Circuit** for details).

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

BP0 to BP7 have an extremely low drive capability as compared to that of other ports. Thus, use them to drive the CMOS load.

When the $\overline{\text{RESET}}$ signal is generated, the output latches of ports 2 to 7 are cleared, the output buffer is turned off and the input mode is set.

Phase-out/Discontinued

Figure 5-2. Configuration of Ports 0 and 1

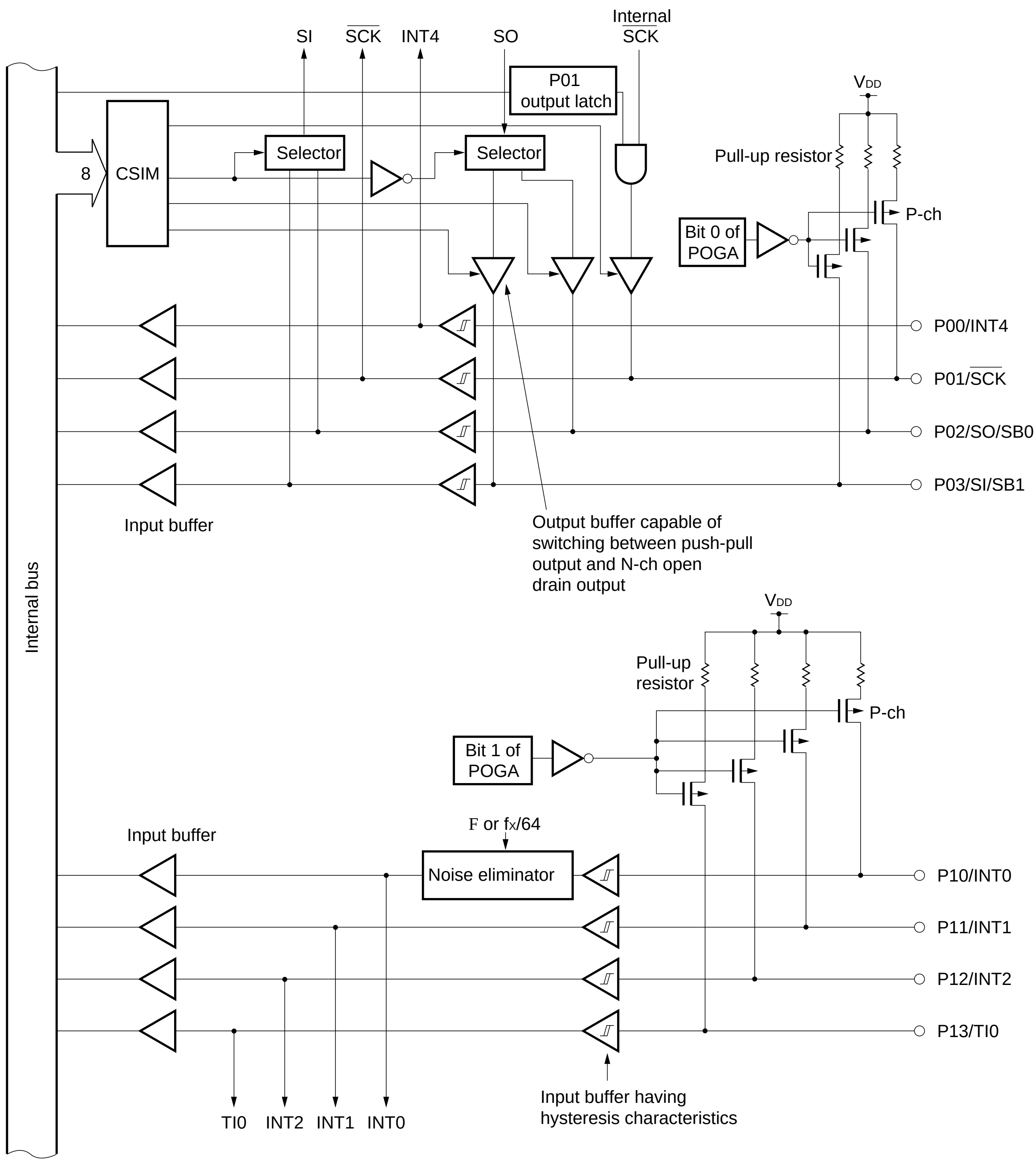
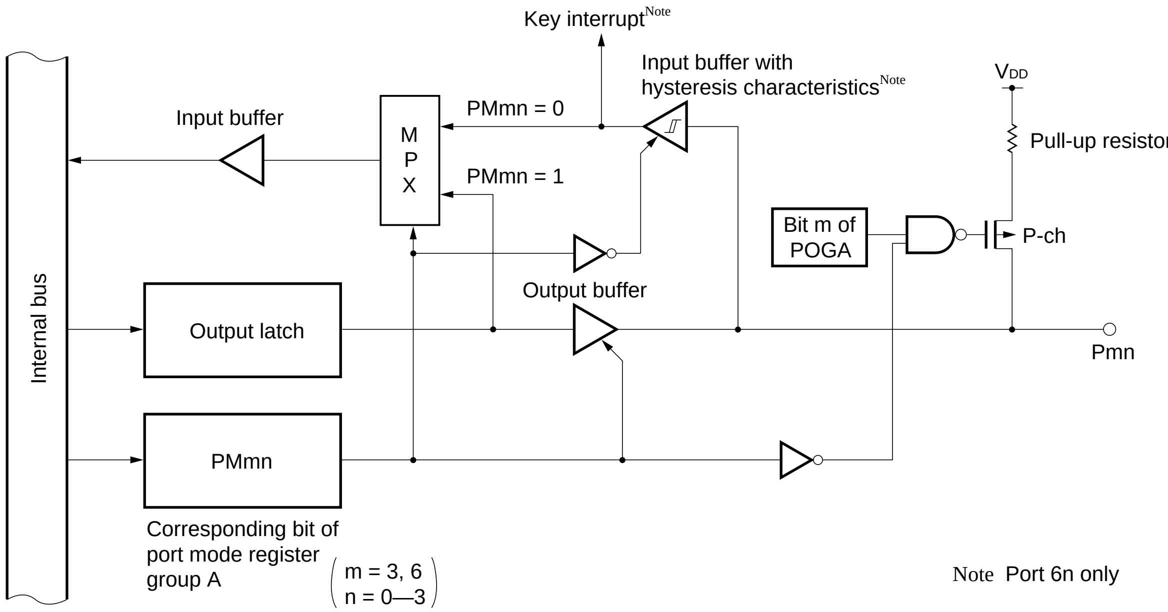
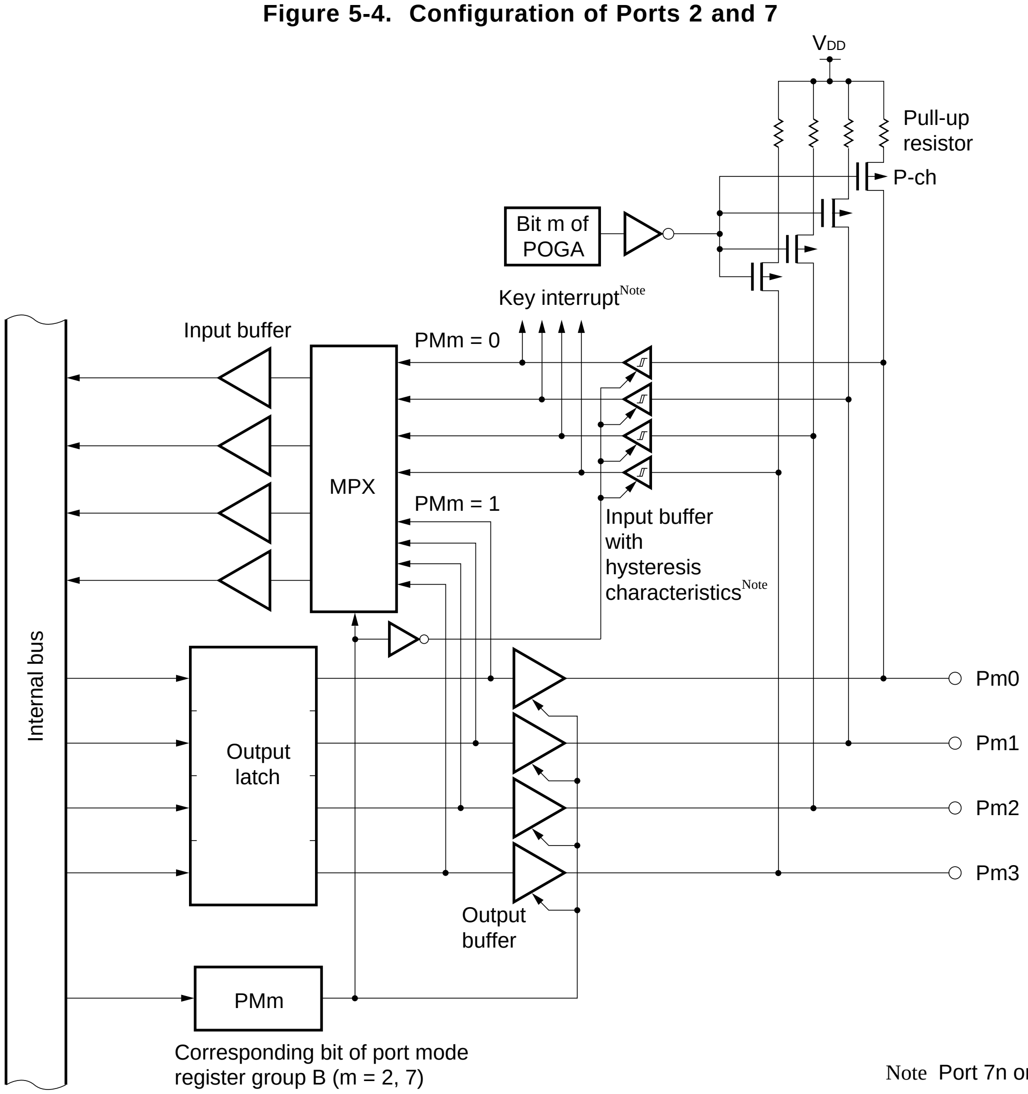


Figure 5-3. Configuration of Ports 3n and 6n (n = 0 to 3)



Note Port 6n only

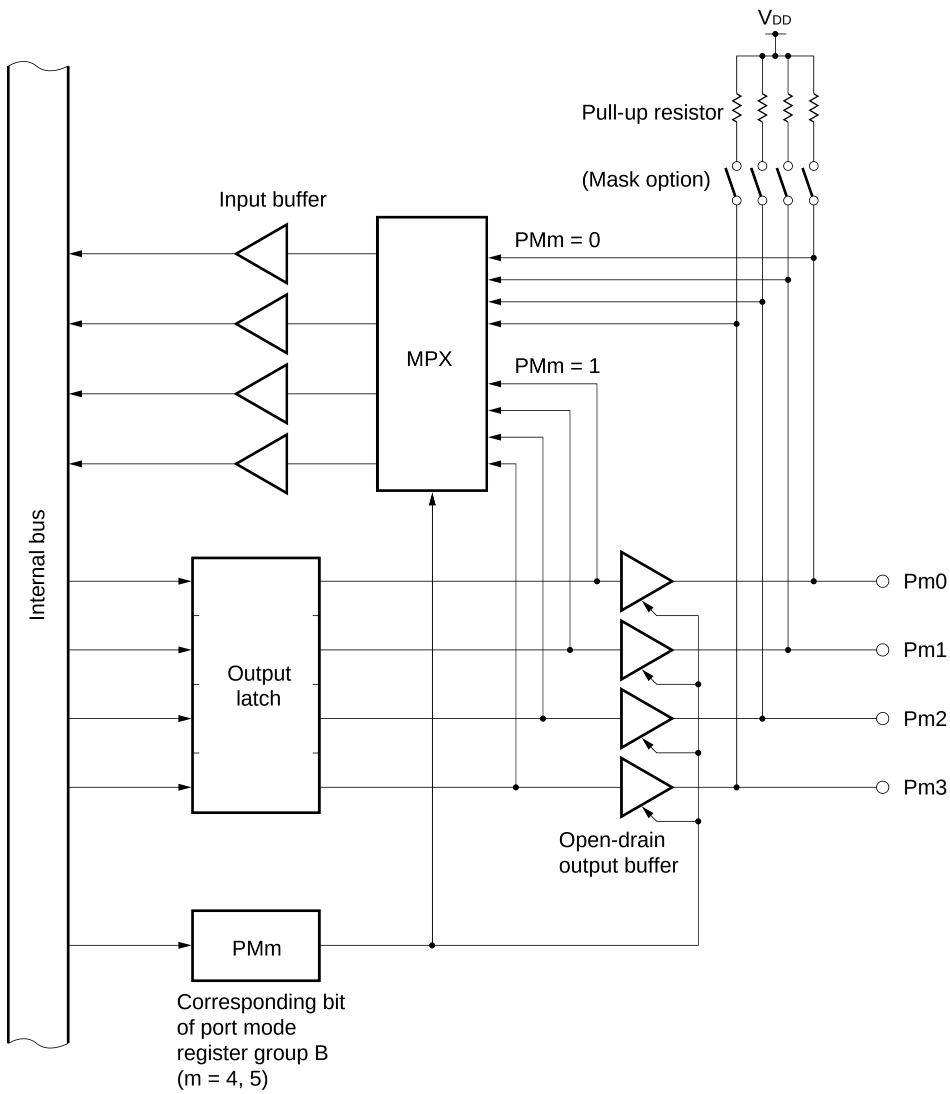
Figure 5-4. Configuration of Ports 2 and 7



Note Port 7n only

Phase-out/Discontinued

Figure 5-5. Configuration of Ports 4 and 5



5.1.2 Input/output mode setting

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

Each port functions as an input or output port when the port mode register bit is “0” or “1”, respectively.

When the output mode is selected by setting the port mode register, the output latch content is simultaneously generated to the output pin. Therefore, the output latch content must be rewritten as the necessary value before the output mode is set.

Port mode register groups A and B 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. As a result, the output buffer is turned off and all ports are set to the input mode.

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

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

Figure 5-6. Port Mode Register Format

Port Mode Register Group A

Address	7	6	5	4	3	2	1	0	Symbol
FE8H	PM63	PM62	PM61	PM60	PM33	PM32	PM31	PM30	PMGA

Symbol PMGA

PM3n PM6n	P3n, P6n pin input/output specification (n = 0 to 3)
0	Input mode (Output buffer off)
1	Output mode (Output buffer on)

Port Mode Register Group B

Address	7	6	5	4	3	2	1	0	Symbol
FECH	PM7	–	PM5	PM4	–	PM3	–	–	PMGB

Symbol PMGB

PMn	Port n input/output specification (n = 2, 4, 5, 7)
0	Input mode (Output buffer off)
1	Output mode (Output buffer on)

–: Can be 0 or 1.

5.1.3 Digital input/output port manipulation instruction

All on-chip input/output ports of the μPD75308 are mapped in the data memory space and thus all data memory manipulation instructions can be applied. Table 5-2 describes instructions seemingly effective for input/output pin operations among data memory manipulation instructions, together with 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 carried out for digital input/output ports PORT0 to PORT7 and thus port bit manipulations are enabled for them irrespectively of MBE and MBS settings.

Example P50 is ORed with p41 and the result is output 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 instruction

In addition to the IN/OUT instruction, all 4-bit memory manipulation instructions, including MOV, XCH, ADDS and INCS, can be used. Memory bank 15 must be selected before each 4-bit memory manipulation instruction is executed.

Examples 1. The accumulator content is output to port 3.

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

2. Data output at port 5 plus the accumulator value are output.

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

3. Whether port 4 data is larger than the accumulator value is tested.

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

Phase-out/Discontinued

(3) 8-bit manipulation instruction

For ports 4 and 5 and ports 6 and 7 which can be used for 8-bit manipulation as a pair, MOV/XCH/SKE instructions can be used in addition to the IN/OUT instructions. Memory bank 15 must be selected beforehand as with 4-bit manipulation.

Example The BC register pair data is output to the output port specified by the 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. List of Input/Output Pin Manipulation Instructions

<table>
<tr><th colspan="2"></th><th>PORT 0</th><th>PORT 1</th><th>PORT 2</th><th>PORT 3</th><th>PORT 4</th><th>PORT 5</th><th>PORT 6</th><th>PORT 7</th><th>BIT PORT 0 – 7</th></tr>
<tr><td>IN</td><td>A, PORTn[Note 1]</td><td colspan="8">○</td><td>MOV A, men[Note 3, 4]</td></tr>
<tr><td>IN</td><td>XA, PORTn[Note 1]</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">○</td><td colspan="2">○</td><td>–</td></tr>
<tr><td>OUT</td><td>PORTn, A[Note 1]</td><td>–</td><td>–</td><td colspan="6">○</td><td>MOV men, A[Note 3, 4]</td></tr>
<tr><td>OUT</td><td>PORTn.XA[Note 1]</td><td colspan="2">–</td><td colspan="2">–</td><td colspan="2">○</td><td colspan="2">○</td><td>–</td></tr>
<tr><td>SET1</td><td>PORTn.bit</td><td>–</td><td>–</td><td colspan="6">○</td><td>SET1 BPn[Note 3]</td></tr>
<tr><td>SET1</td><td>PORTn.@L[Note 2]</td><td>–</td><td>–</td><td colspan="6">○</td><td>–</td></tr>
<tr><td>CLR1</td><td>PORTn.bit</td><td>–</td><td>–</td><td colspan="6">○</td><td>CLR1 BPn[Note 3]</td></tr>
<tr><td>CLR1</td><td>PORTn.@L[Note 2]</td><td>–</td><td>–</td><td colspan="6">○</td><td>–</td></tr>
<tr><td>SKT</td><td>PORTn.bit</td><td colspan="8">○</td><td>SKT BPn[Note 3]</td></tr>
<tr><td>SKT</td><td>PORTn.@L[Note 2]</td><td colspan="8">○</td><td>–</td></tr>
<tr><td>SKF</td><td>PORTn.bit</td><td colspan="8">○</td><td>SKF BPn[Note 3]</td></tr>
<tr><td>SKF</td><td>PORTn.@L[Note 2]</td><td colspan="8">○</td><td>–</td></tr>
<tr><td>AND1</td><td>CY, PORTn.bit</td><td colspan="8">○</td><td>AND1 CY, @H+BPn[Note 3, 5]</td></tr>
<tr><td>AND1</td><td>CY, PORTn.@L[Note 2]</td><td colspan="8">○</td><td>–</td></tr>
<tr><td>OR1</td><td>CY, PORTn.bit</td><td colspan="8">○</td><td>OR1 CY, @H+BPn[Note 3, 5]</td></tr>
<tr><td>OR1</td><td>CY, PORTn.@L[Note 2]</td><td colspan="8">○</td><td>–</td></tr>
<tr><td>XOR1</td><td>CY, PORTn.bit</td><td colspan="8">○</td><td>XOR1 CY, @H+BPn[Note 3, 5]</td></tr>
<tr><td>XOR1</td><td>CY, PORTn.@L[Note 2]</td><td colspan="8">○</td><td></td></tr>
</table>

Notes 1. MBE = 0 or (MBE = 1, MBS = 15) must be set before execution.

2. The low-order 2 bits of the address and the bit address are indirectly specified with the L register.

3. (MBE = 1, MBS = 1) must be set before execution.

4. Bit 0 of accumulator A corresponds to BPn.

5. FH is written into the H register.

5.1.4 Digital input/output port operations

When a data memory manipulation instruction is carried out for the digital input/output port, port and pin operations vary depending on the input/output mode setting (refer to **Table 5-3**). This is because, as is clear from the input/output port configuration, data fetched into the internal bus becomes data of each pin in the input mode and data of the output latch in the output mode.

(1) Operations in the input mode

When a test instruction such as SKT or a 4/8-bit instruction which fetches port data into the internal bus (IN, OUT, arithmetic, and comparison instructions) is executed, data of each pin is operated.

When an instruction (OUT and MOV instructions) to transfer the accumulator content to the port in 4/8-bit units is executed, the accumulator data is latched into the output latch with the output buffer set to OFF.

When the XCH instruction is executed, each pin data is input to the accumulator and accumulator data is latched into the output latch with the output buffer set to OFF.

When the INCS instruction is executed, one is added to each pin data (4 bits) and the sum is latched into the output latch with the output buffer set to OFF.

When the SET1, CLR1 or SKTCLR instruction intended to rewrite the data memory bitwise is executed, the output latch of the specified bit can be rewritten as specified by the instruction but the contents of the output latches of other bits become indeterminate.

(2) Operations in the output mode

When the test instruction or an instruction to fetch port data into the internal bus in 4/8-bit units, the output latch content is operated.

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

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

When the INCS instruction is executed, the output latch content added by one is latched into the output latch and is output from the pin.

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

Table 5-3. Input/Output Port and Pin Operations

Instruction to be Executed	Port and Pin Operations	
	Input Mode	Output Mode
SKT ① SKF ①	Pin data test	Output latch data test
AND1 CY, ① OR1 CY, ① XOR1 CY, ①	Pin data and CY operation	Output latch data and CY operation

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

ADDS A, @HL
 ADDC A, @HL
 SUBS A, @HL
 SUBC A, @HL
 AND A, @HL
 OR A, @HL
 XOR A, @HL
	Pin data and accumulator operation	Output latch data and accumulator operation

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

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

XCH A, PORTn XCH XA, PORTn XCH A, @HL XCH XA, @HL	Pin data transfer to the accumulator and accumulator data transfer to the output latch (with the output buffer set to OFF)	Data exchange between output latch and accumulator
INCS PORTn INCS @HL	Pin data added by one and latched into the output latch	Output latch content added by one
SET1 ① CLR1 ① SKTCLR ①	The output latch of the specified bit is rewritten as specified by an instruction and the output latches of all other bits remain unchanged.	Output pin status is changed according to an instruction.

① : Indicates two addressing modes PORTn. bit and PORTn.@L.

5.1.5 Integration of pull-up resistor

A pull-up resistor can be integrated at each port pin of the μPD75308 (except P00 and BP0 to BP7). Pull-up resistor integration can be specified for pins by using the software of mask option.

The specification method for each port pin is shown in Table 5-4. Specification by software is performed based on the format shown in Figure 5-7.

Specification of the internal pull-up resistor for ports 3 and 6 is effective for pins specified as in the input mode. For pins specified as in the output mode, specification of the internal pull-up register is not possible, irrespective of POGA setting.

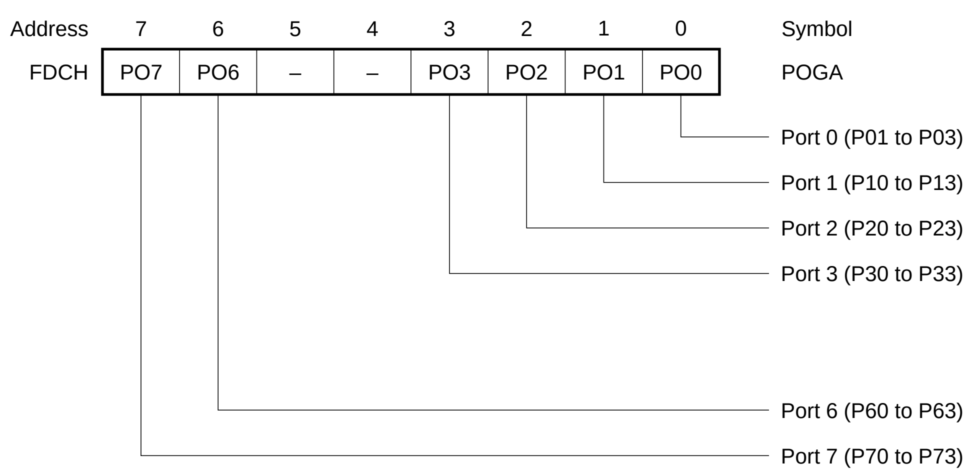
Table 5-4. Specification of Pull-Up Resistor Integration

Port (Pin Name)	Specification of Pull-Up Resistor Integration	POGA Bit
Port 0 (P01 to P03)[Note]	3-bit unit specification of integration using software	Bit 0
Port 1 (P10 to P13)	4-bit unit specification of integration using software	Bit 1
Port 2 (P20 to P23)		Bit 2
Port 3 (P30 to P33)		Bit 3
Port 6 (P60 to P63)		Bit 6
Port 7 (P70 to P73)		Bit 7
Port 4 (P40 to P43)	Bitwise specification of integration using mask option	–
Port 5 (P50 to P53)		

Note No pull-up resistors can be integrated at P00 pin.

Remark Pull-up resistors by mask option are not provided for μPD75P308, 75P316, 75P316A and 75P316B.

Figure 5-7. Pull-up Resistor Specification Register Group A



	Specification
0	No integration of pull-up resistor
1	Integration of pull-up resistor

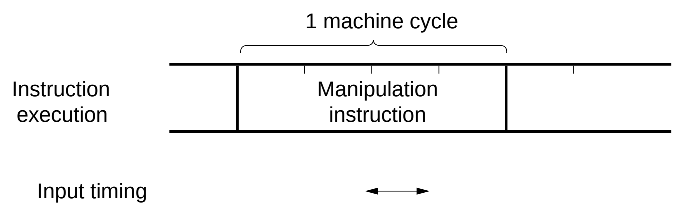
5.1.6 Input/output timing of digital input/output port

Figure 5-8 shows the timing of data output to the output latch and the timing of pin data or output latch data fetch into the internal bus.

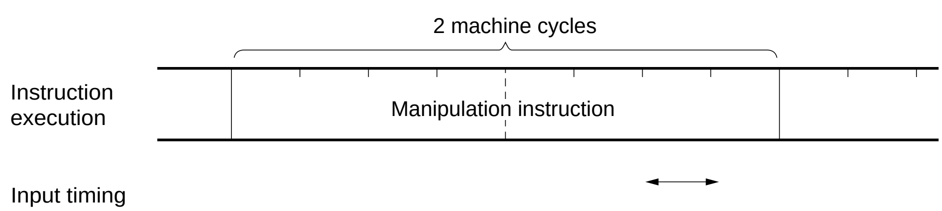
Figure 5-9 shows the ON timing when pull-up resistor integration is specified by software.

Figure 5-8. Input/Output Timing of Digital Input/Output Port

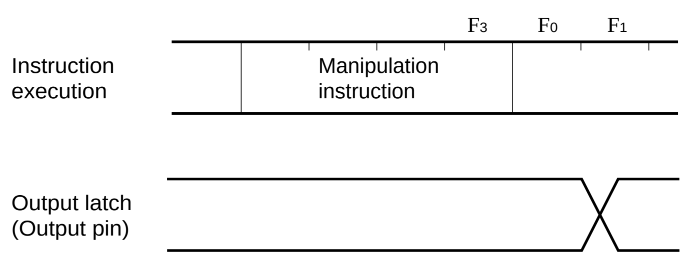
(a) Data fetch by 1-machine cycle instruction



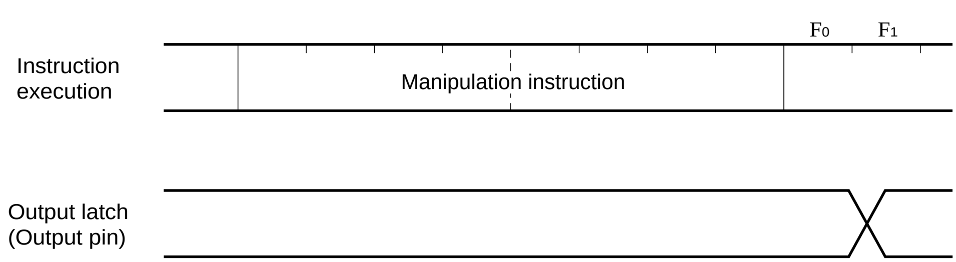
(b) Data fetch by 2-machine cycle instruction



(c) Data latch by 1-machine cycle instruction

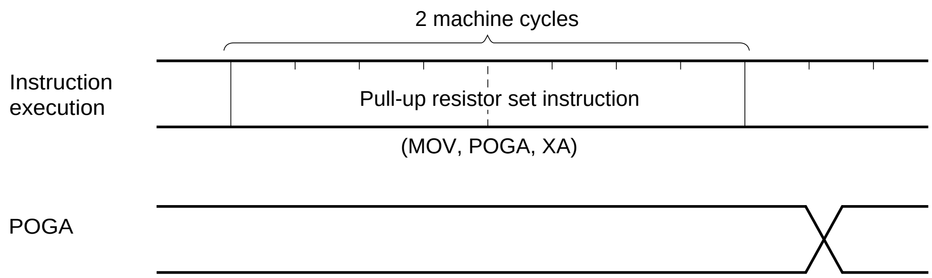


(d) Data latch by 2-machine cycle instruction



Phase-out/Discontinued

Figure 5-9. Pull-Up Resistor ON Timing by Software



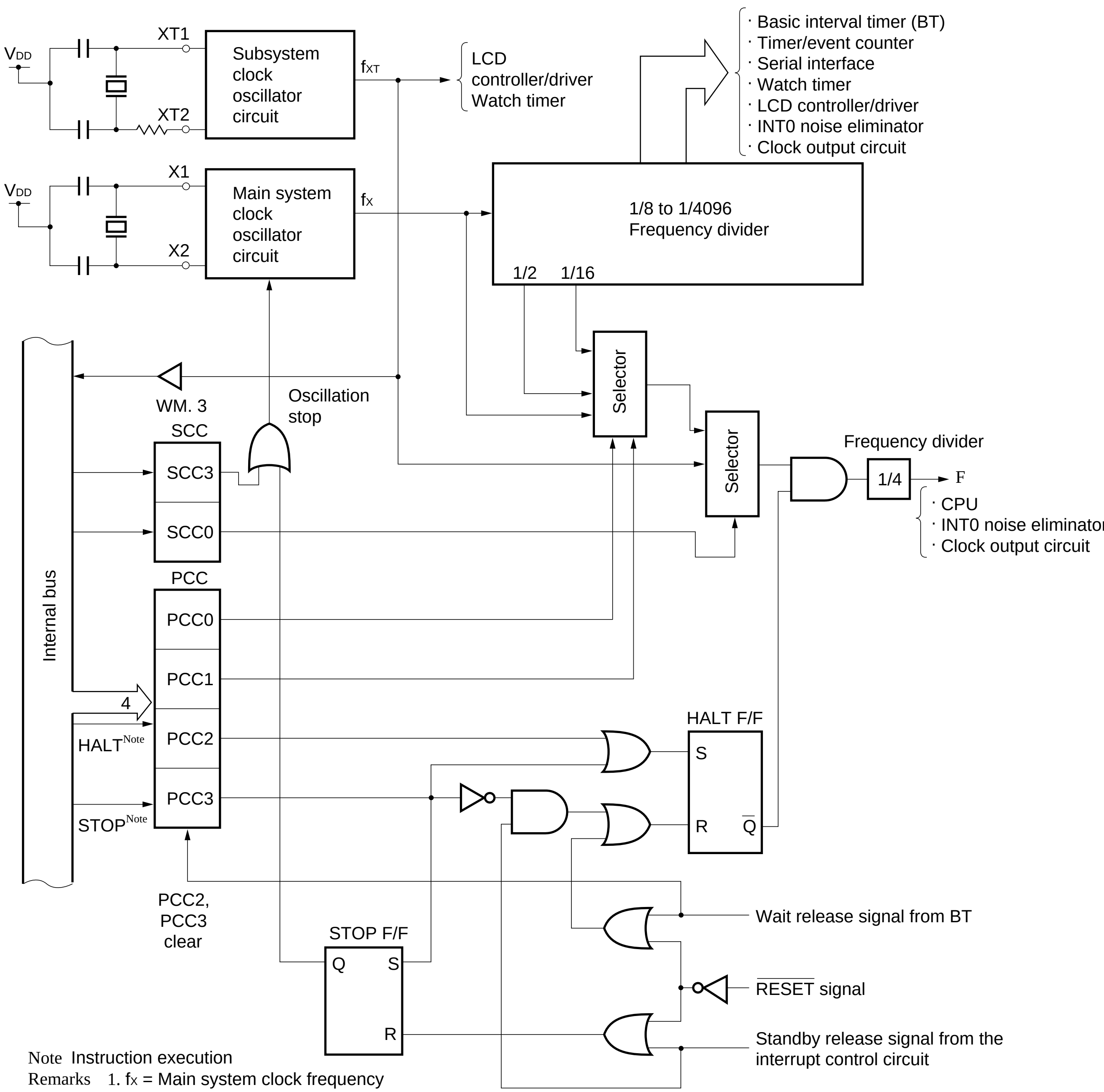
5.2 Clock Generator Circuit

The clock generator circuit supplies the CPU and peripheral hardware devices with various clocks and controls the CPU operating mode.

5.2.1 Clock generator circuit configuration

The clock generator circuit is configured as shown in Figure 5-10.

Figure 5-10. Clock Generator Circuit Block Diagram



Note Instruction execution

Remarks
1. f_X = Main system clock frequency
2. f_{XT} = Subsystem clock frequency
3. F = CPU clock
4. PCC: Processor clock control resistor
5. SCC: System clock control resistor
6. 1 clock cycle of F (t_{CY}) is 1 machine cycle.

5.2.2 Clock generator circuit functions and operations

The clock generator circuit generates various clocks and controls the CPU operation mode including the standby mode, etc.

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

Clock generator circuit operations are determined as follows by the processor clock control register (PCC) and system clock control register (SCC).

(a) When $\overline{\text{RESET}}$ signal is generated, the lowest speed mode (15.3 μs: When operated at 4.19 MHz) is selected (PCC = 0, SCC = 0)

(b) With the main system clock selected, 3-level CPU clocks can be selected by PCC setting (0.95 μs, 1.91 μs, 15.3 μs: When operated at 4.19 MHz).

(c) With the main system clock selected, two standby modes, STOP mode and HALT mode, can be used.

(d) With the subsystem clock selected by SCC, this circuit can operate at extremely low speeds with low current consumption (122 μs: When operated at 32.768 kHz). In this case, the PCC set value has no effects on the CPU clock.

(e) With the subsystem clock selected, main system clock oscillation can be stopped by SCC. This circuit can be used in the HALT mode but it cannot be used in the STOP mode (subsystem clock oscillation cannot be stopped).

(f) Clocks for the peripheral hardware are supplied by dividing the main system clock. Subsystem clocks can be directly supplied to the watch timer only. Thus, the clock function, the function of the LCD controller operated by clocks from the watch timer and the buzzer output function can be continued even in the standby mode.

(g) When the subsystem clock is selected, the watch timer and the LCD controller can continue to operate normally. All other hardware devices operate by main system clocks and thus cannot be used if the main system clock is stopped.

(1) Processor clock control register (PCC)

The PCC is a 4-bit register which selects CPU clock Φ with low-order 2 bits and executes CPU operating mode control with high-order 2 bits (refer to **Figure 5-11**).

When bit 3 or 2 is set (1), the standby mode is set. When the bit is released by the standby release signal, the bit is automatically cleared and the normal operating mode is set (refer to **CHAPTER 7 STANDBY FUNCTION** for details).

The low-order 2 bits of the PCC are set by the 4-bit memory manipulation instruction (with the low-order 2 bits set to 0).

Bits 3 and 2 are set (1) by the STOP and HALT instructions, respectively.

The STOP and HALT instructions can always be executed irrespectively of MBE content.

The CPU clock can only be selected when the circuit is in operation with the main system clock. When the circuits operated with the subsystem clock, the low-order 2 bits of the PCC are invalidated and are fixed to $f_{XT}/4$. The STOP instruction is also enabled only when the circuit is in operation with the main system clock.

Examples 1. The machine cycle is set to 0.95 μs (4.19 MHz).

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

2. The machine cycle is set to 1.63 μs (f_X = 4.91 MHz)

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

3. The STOP mode is set. (Make sure to write the NOP instruction after the STOP or HALT instruction.)

```
STOP
NOP
```

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

Figure 5-11. Processor Clock Control Register Format

Address	3	2	1	0	Symbol
FB3H	PCC3	PCC2	PCC1	PCC0	PCC

CPU Clock Selection Bit

When $f_X \leq 4.19$ MHz

		SCC = 0 When $f_X \leq 4.19$ MHz in Parentheses		SCC = 1 When $f_{XT} \leq 32.768$ kHz 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	Setting prohibited	–		
1	0	$\Phi = f_X/8$ (524 kHz)	1.91 μs		
1	1	$\Phi = f_X/4$ (1.05 MHz)	0.95 μs		

f_X: Main system clock oscillator output frequency
f_{XT}: Subsystem clock oscillator output frequency

When 4.19 MHz < $f_X \leq 5.0$ MHz

		SCC = 0 When $f_X \leq 4.19$ MHz in Parentheses		SCC = 1 When $f_{XT} \leq 32.768$ kHz 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	Setting prohibited	–		
1	0	$\Phi = f_X/8$ (614 kHz)	1.63 μs		
1	1	Setting prohibited	–		

CPU Operating Mode Control Bit

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

Caution When using a value of f_X such that 4.19 MHz < f_X - 5 MHz, if the fastest mode: $\Phi = f_X/4$ (PCC1, PCC0 = 11) is set as CPU clock frequency, 1 machine cycle becomes less than 0.95 μs. Thus the MIN. standard value 0.95 μs cannot be observed. Therefore, in this case, “PCC1, PCC0 = 11” cannot be set, set “PCC1, PCC0 = 10 or 00”. Consequently, the combination of “f_X = 4.19 MHz, PCC1, PCC0 = 11” is the selection for the maximum CPU clock speed (1 machine cycle = 0.95 μs).

(2) System clock control register (SCC)

The SCC is a 4-bit register which selects the CPU clock Φ with the least significant bit and controls main system clock oscillation stop with the most significant bit (refer to **Figure 5-12**).

Although SCC.0 and SCC.3 are located at the same data memory address, both bits cannot be changed simultaneously. Thus, SCC.0 and SCC.3 are set by bit manipulation instructions. They are always ready for bit manipulation irrespectively of MBE content.

Main system clock oscillation stop by setting SCC.3 is only enabled when the circuit is in operation with the subsystem clock. While the circuit is operating with the main system clock, oscillation is stopped by the STOP instruction.

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

Figure 5-12. System Clock Control Register Format

Address	3	2	1	0	Symbol
FB7H	SCC3	—	—	SCC0	SCC

SCC3	SCC0	CPU Clock Selection	Main System Clock Oscillation
0	0	Main system clock	Oscillation possible
0	1	Subsystem clock	
1	0	Setting prohibited	
1	1	Subsystem clock	Oscillation stop

Cautions 1. It takes a maximum of 1/f_{XT} to change the system clock. Thus, to stop main system clock oscillation, set SCC.3 following the lapse of a time longer than the machine cycle indicated in Table 5-5 after the subsystem clock has been changed.

2. If oscillation is stopped by setting SCC.3 when the circuit is in operation with the main system clock, the normal STOP mode is not set.

3. When "1" is set to SCC.3, X1 input is internally short-circuited (ground potential) to prevent the crystal oscillator circuit block from leaking. Thus, when using an external clock as the main system clock, "1" should not be set to SCC.3.

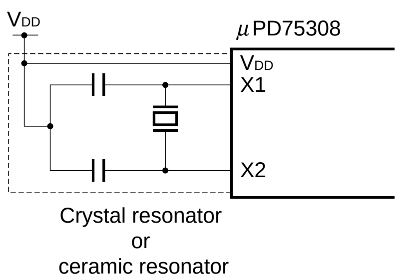
(3) System clock oscillator circuit

The main system clock oscillator circuit oscillates with the crystal resonator (4.194304 MHz TYP.) or the ceramic resonator connected to the X1 and X2 pins.

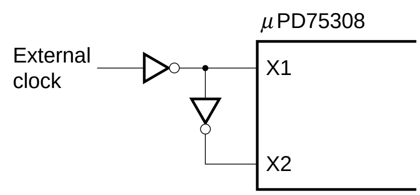
External clocks can also be input to this circuit. In this case, apply the clock signal to the X1 pin and the inverse signal to the X2 pin.

Figure 5-13. Externally Mounted Circuit for the Main System Clock Oscillator Circuit

(a) Crystal/ceramic oscillation



(b) External clock



Caution While an external clock is being input, the STOP mode cannot be set. This is because X1 pin is short-circuited to V_{SS} in the STOP mode.

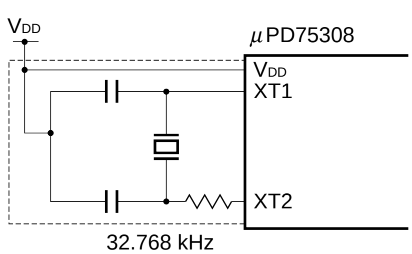
The subsystem clock oscillator circuit oscillates with the crystal resonator (32.768 kHz TYP.) connected to the XT1 and XT2 pins.

An external clock can be input. In this case, apply the clock signal to the XT1 pin and the inverted signal should be input to the XT2 pin.

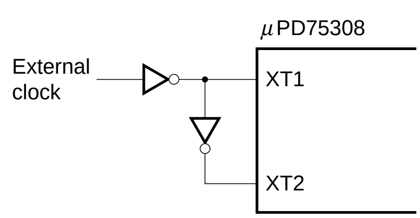
The XT1 pin status can be tested by checking bit 3 of the watch mode register (WM).

Figure 5-14. Externally Mounted Circuit for Subsystem Clock Oscillator Circuit

(a) Crystal oscillator



(b) External clock



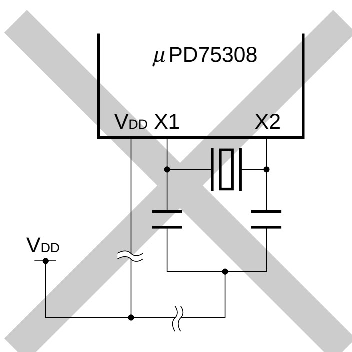
Cautions 1. When using the main system clock or subsystem clock oscillator, wiring enclosed by dotted line in [ ] Figures 5-13 and 5-14 should be made as follows to avert the adverse effect of wiring capacitance.

- **Wires should be kept as short as possible.**
- **Do not cross other signal lines. Do not route wires close to a fluctuating high-current line.**
- **Oscillation capacitor connection points should be always at the same electric potential as V_{DD}. Do not connect to a high-current power supply pattern.**
- **Do not take a signal from the oscillator.**

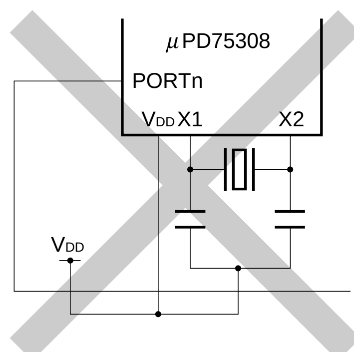
Note that the subsystem clock oscillator has small amplification in order to keep power consumption low.
Figure 5-15 shows an example of an incorrect resonator connection circuit.

Figure 5-15. Example of Incorrect Resonator Connection Circuit (1/2)

(a) Excessively long connection circuit wires



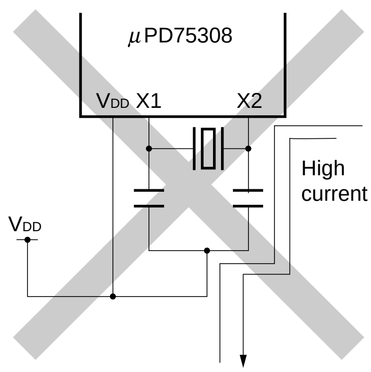
(b) Crossed signal lines



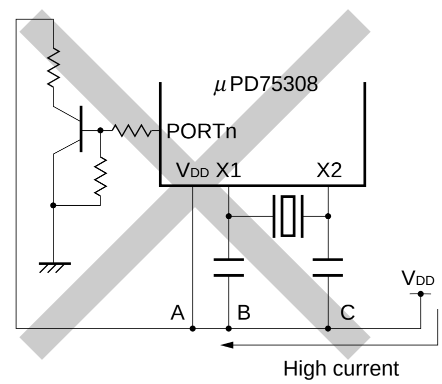
Remark When the subsystem clock is used, read XT1 and XT2 in place of X1 and X2. Also, a resistor should be inserted in series on XT2 side.

Figure 5-15. Example of Incorrect Resonator Connection Circuit (2/2)

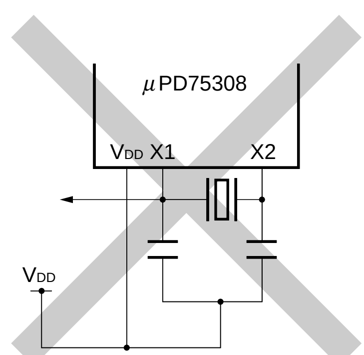
(c) Fluctuating high current is close to the signal line



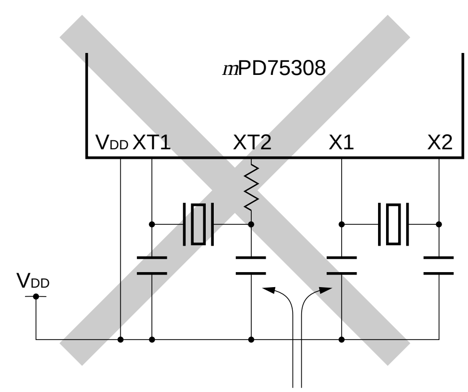
(d) Current is flowing in the oscillator power supply line.
(A, B and C electric potentials fluctuate)



(e) Signal is taken out.



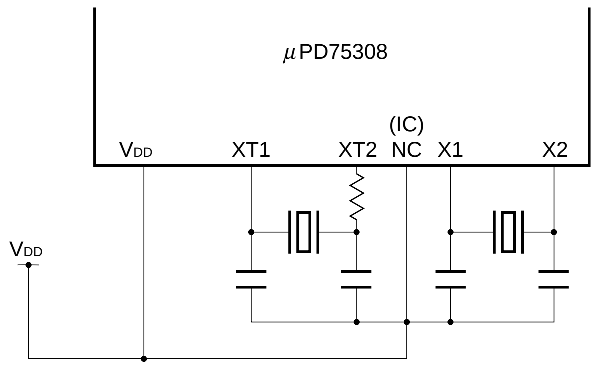
(f) Main system clock and subsystem clock signal lines are close and parallel to each other.



XT2 and X1 are wired in parallel.
(See Caution 2 for remedy.)

Remark When the subsystem clock is used, read XT1 and XT2 in place of X1 and X2. Also, a resistor should be inserted in series on XT2 side.

Cautions 2. In Figure 5-15 (f), XT2 and X1 are wired in parallel. Thus, crosstalk noise of X1 affects XT2, resulting in misoperation.
To avoid this, it is recommended that XT2 and X1 should not be wired in parallel and that the NC pin between XT2 and X1 should be connected to V_{DD}.
In the μPD75312B/75316B, NC pin is not used, but IC pin. Therefore, be sure to connect the pin to V_{DD} directly.



(4) Frequency divider

The frequency divider divides the main system clock oscillator output (f_X) to generate various kinds of clocks.

(5) When subsystem clock is not used

Unless the subsystem clock need be used for power dissipation and watch operations, deal with XT1 and XT2 pins as follows.

XT1 : Connect to V_{SS} and V_{DD}.
XT2 : Leave unconnected

However, in this situation, when the main system clock stops, a little leak current is generated from a feedback resistor in subsystem clock oscillator. This feedback resistor can be removed by mask option specification in order to prevent this leak current. In this case, also deal with XT1 and XT2 pins in the same way as above. (The mask option can be specified in ordering.)

5.2.3 System clock and CPU clock setting

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

The system clock can be switched to the CPU clock or vice versa the low-order 2 bits of PCC and the least significant bit of SCC. This switching is not executed just after the register is rewritten. The previous clock remains in operation during the specified machine cycle. Thus, to stop main system clock oscillation, it is necessary to execute the STOP instruction or set SCC.3.

Table 5-5. Maximum Time Required for System Clock/CPU Clock Switching

<table>
<tr><td colspan="3">Set Value before Switching</td><td colspan="12">Set Value after Switching</td></tr>
<tr><td>SCC</td><td>PCC</td><td>PCC</td><td>SCC0</td><td>PCC1</td><td>PCC0</td><td>SCC0</td><td>PCC1</td><td>PCC0</td><td>SCC0</td><td>PCC1</td><td>PCC0</td><td>SCC0</td><td>PCC1</td><td>PCC0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>x</td><td>x</td></tr>
<tr><td rowspan="3">0</td><td>0</td><td>0</td><td colspan="3"></td><td colspan="3">1-machine cycle</td><td colspan="3">1-machine cycle</td><td colspan="3">$\frac{f_X}{64 f_{XT}}$ machine cycle
(2-machine cycle)</td></tr>
<tr><td>1</td><td>0</td><td colspan="3">8-machine cycle</td><td colspan="3"></td><td colspan="3">8-machine cycle</td><td colspan="3">$\frac{f_X}{8 f_{XT}}$ machine cycle
(16-machine cycle)</td></tr>
<tr><td>1</td><td>1</td><td colspan="3">16-machine cycle</td><td colspan="3">16-machine cycle</td><td colspan="3"></td><td colspan="3">$\frac{f_X}{4 f_{XT}}$ machine cycle
(32-machine cycle)</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td colspan="3">1-machine cycle</td><td colspan="3">1-machine cycle</td><td colspan="3">1-machine cycle</td><td colspan="3"></td></tr>
</table>

When f_X = 4.19 MHz and f_{XT} = 32.768 kHz in parentheses
x: don't care

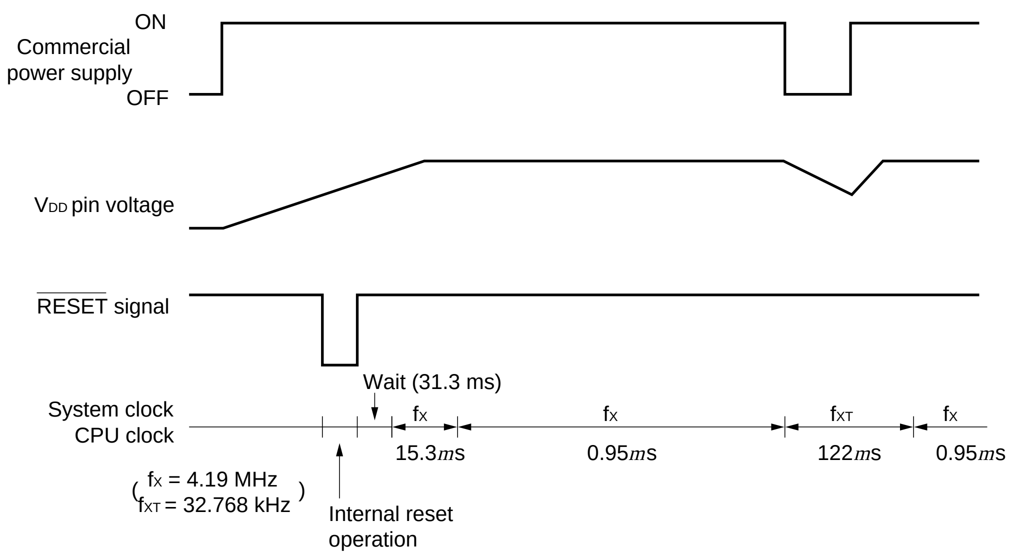
Caution The values of f_X and f_{XT} vary according to conditions such as variations in the resonator ambient temperature and load capacitance capacity. In particular, if f_X is higher than the nomial value or f_{XT} is lower than the nomial value, the machine cycles given by the expressions $f_X/64f_{XT}$, $f_X/8f_{XT}$ and $f_X/4f_{XT}$ in the table are greater than the machine cycles given by the nomial values of f_X and f_{XT}. Therefore, when setting the wait time required for system clock/CPU clock switchover, this should be made longer than the number of machine cycles given by the nominal values of f_X and f_{XT}.

Remark CPU clock Φ is a clock supplied to the internal CPU of the μPD75308. The inverse of the clock is the minimum instruction time (defined as 1-machine cycle in this manual).

(2) System clock/CPU clock switching procedure

The system clock/CPU clock switching procedure is described in accordance with Figure 5-16.

Figure 5-16. System Clock/CPU Clock Switching



① When the $\overline{\text{RESET}}$ signal is generated after the wait time (31.3 ms: When operated at 4.19 MHz) securing oscillation stabilization, the CPU starts operating at the lowest speed (15.3 μs: When operated at 4.19 MHz) of the main system clock.

② After a sufficient time has passed for the V_{DD} pin voltage to increase up to a value enabling the highest speed operation, the PCC is rewritten and the CPU operates at the highest speed.

③ If the turning off of the commercial power supply is detected from an interrupt (which INT4 can be effectively applied), SCC.0 is set and is operated with the subsystem clock (the subsystem clock oscillation should be started). After the lapse of the time (32-machine cycle) required for the clock to be switched to the subsystem clock, SCC.3 is set and the main system clock oscillation is stopped.

④ If the recovery of the commercial power supply is detected from an interrupt, SCC.3 is cleared and main system clock oscillation is started. After the lapse of the time required for oscillation to stabilize, SCC.0 is cleared and the CPU operates at the highest speed.

5.2.4 Clock output circuit

(1) Clock output circuit configuration

The clock output circuit is configured as shown in Figure 5-17.

(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 or to supply the peripheral LSI with clock pulses.

The following procedure is used to generate clock pulses.

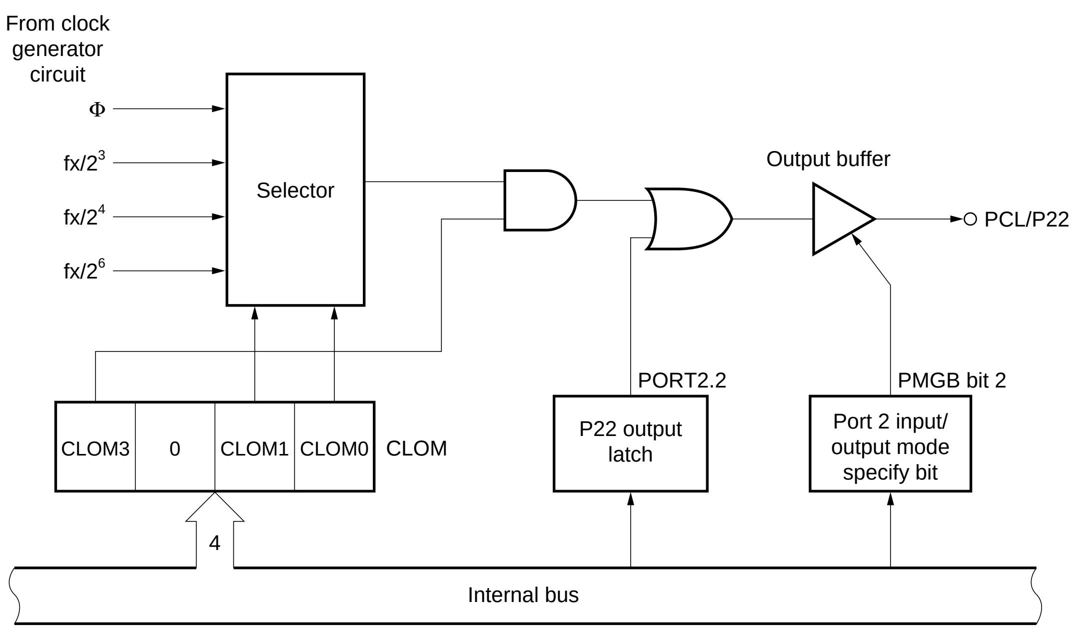
(a) The clock output frequency is selected. Clock output is disabled.

(b) 0 is written to the P22 output latch.

(c) Port 2 input/output mode is set to output.

(d) Clock output is enabled.

Figure 5-17. Clock Output Circuit Configuration



Remark When clock output enable/disable is switched, pulses having short widths are not output.

(3) Clock output mode register (CLOM)

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

Example CPU clock Φ is output from the 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 5-18. Clock Output Mode Register Format

Address	3	2	1	0	Symbol
FD0H	CLOM3	0	CLOM1	CLOM0	CLOM

Clock Output Frequency Select Bit

f_X = 4.19 MHz in Parentheses

CLOM1	CLOM0	
0	0	Φ output[Note] (1.05 MHz, 524 kHz, 65.5 kHz)
0	1	$f_X/2^3$ output (524 kHz)
1	0	$f_X/2^4$ output (262 kHz)
1	1	$f_X/2^6$ output (65.5 kHz)

Note Φ is a CPU clock to be selected by PCC.

Clock Output Enable/Disable Bit

CLOM3	
0	Output Disable
1	Output Enable

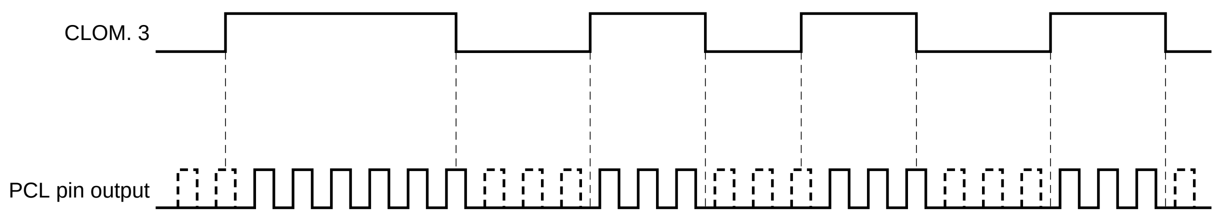
Caution Make sure to write 0 for bit 2 of CLOM.

(4) Example of remote controlled output application

The clock output function of the μPD75308 can be applied to remote controlled output. The remote controlled output carrier frequency is selected by the clock frequency select bit of the clock output mode register. Pulse output is enabled or disabled by controlling the clock output enable/disable bit using the software.

When clock output enable/disable is switched, pulses having small widths are not output.

Figure 5-19. Remote Controlled Output Application Example



5.3 Basic Interval Timer

The μPD75308 is equipped with an 8-bit basic interval timer and has the following functions.

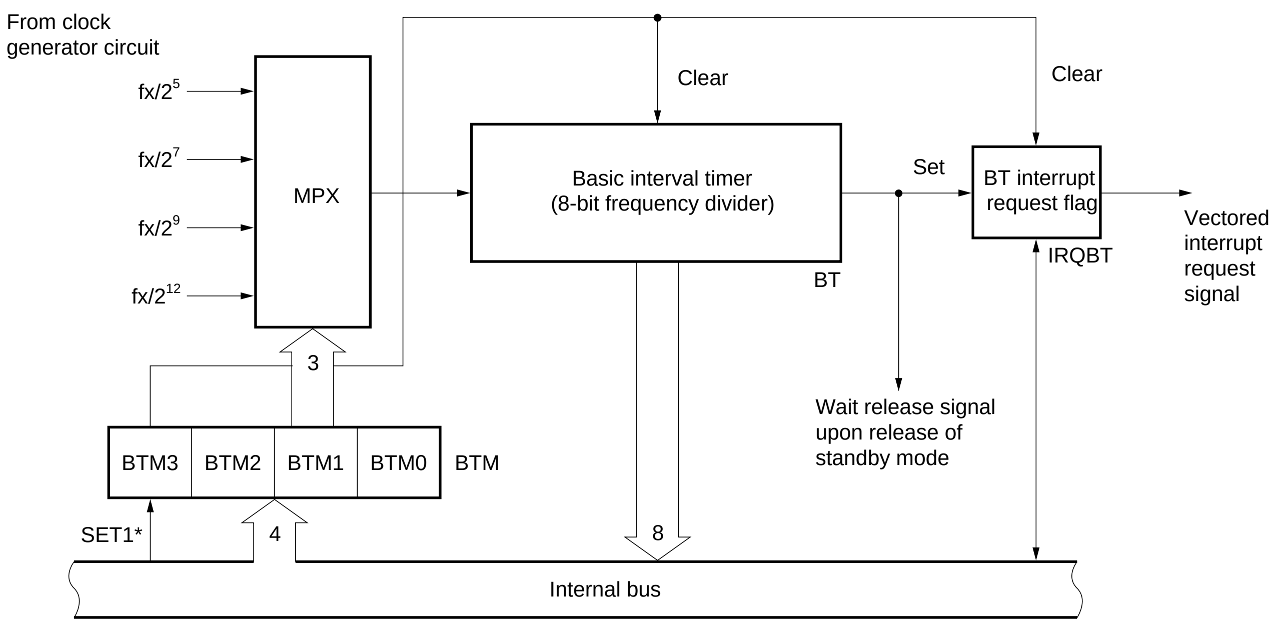
(a) Reference timer generation (4 time intervals)
(b) Selection and count of wait time upon release of the standby mode
(c) Count content read

The basic interval timer can also be used as a watchdog timer to detect a program overrun.

5.3.1 Basic interval timer configuration

The basic interval timer is configured as shown in Figure 5-20.

Figure 5-20. Basic Interval Timer Configuration



Remark * indicates instruction execution.

5.3.2 Basic interval timer mode register (BTM)

The BTM is a 4-bit register to control basic interval timer operations. It is set by a 4-bit memory manipulation instruction.

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

Examples 1. The interrupt generate interval is set to 1.95 ms (4.19 MHz).

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

2. BT and IRQBT are cleared (watchdog timer application)

```
SEL     MB15        ; or CLR1 MBE
SET1    BTM.3       ; BTM bit 3 is set (1).
```

When bit 3 is set (1), the basic interval timer content 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 basic interval timer content is cleared to 0 and the interrupt request signal generate interval time is set to the maximum value.

Figure 5-21. Basic Interval Timer Mode Register Format

Address	3	2	1	0	Symbol
F85H	BTM3	BTM2	BTM1	BTM0	BTM

BTM2	BTM1	BTM0	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	$f_X/2^{7}$ (32.768 kHz)	$2^{15}/f_X$ (7.82 ms)
1	1	1	$f_X/2^{5}$ (131 kHz)	$2^{13}/f_X$ (1.95 ms)
All other cases			Setting prohibited	—

f_X = 4.19 MHz in parentheses

Basic Interval Timer Start Control Bit (BTM3)

The basic interval timer is started (counter and interrupt request flag clear) by writing "1". When the operation starts, the basic interval timer is reset (0).

5.3.3 Basic interval timer operations

The basic interval timer (BT) is incremented by a clock from the clock generator circuit. If the BT overflows, the interrupt request flag (IRQBT) is set. The counting operation of the BT cannot be stopped.

Four interrupt generate intervals are available by BTM setting (refer to **Figure 5-21**).

The BT and the IRQBT can be cleared by setting BTM bit 3 to (1) (interval timer start instruction).

The count status of the basic interval timer (BT) can be read by an 8-bit manipulation instruction. Data write operations are not permitted.

Caution When reading the basic interval timer count content, execute the read instruction twice to prevent unstable data from being read during count update, and compare two read results. If the two values are almost equal, use the 2nd read result. If they differ considerably from each other, carry out the instruction executions again.

Example BT count content read

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

The wait function is available to stop CPU operation until the basic interval timer overflows. This function enables to establish the system clock oscillation stabilizing time upon STOP mode release.

The wait time after $\overline{\text{RESET}}$ signal generation is fixed. When releasing the STOP mode upon generation of an interrupt, the same wait time as the interval time described in Figure 5-21 can be selected by BTM setting. Set the BTM before setting the STOP mode (refer to **CHAPTER 7 STANDBY FUNCTION** for details).

5.3.4 Basic interval timer application examples

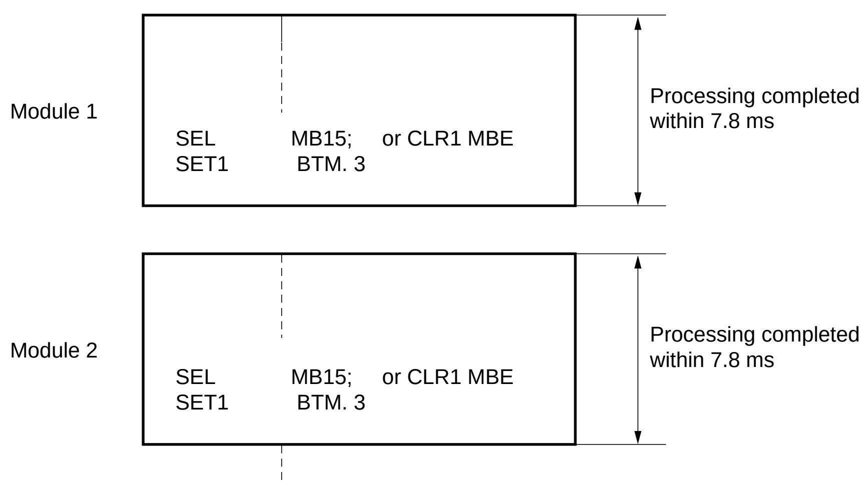
Examples 1. Basic interval timer interrupt is enabled and the interrupt generate interval is set to 1.95 ms (at 4.19 MHz operation).

```
SEL     MB15          ; or CLR1 MBE
MOV     A, #1111B
MOV     BTM, A        ; set and start
EI                    ; interrupt enabled
EI      IEBT          ; BT interrupt enabled
```

2. Watchdog timer application

The program is divided into several modules which terminate processing with the BT set time, and BT and IRQBT are cleared at the end of each module. If an interrupt is generated, an overrun is judged to have occurred.

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



3. Wait time for releasing the STOP mode upon interrupt generation is set to 7.8 ms.

```
SEL     MB15          ; or CLR1 MBE
MOV     A, #1101B
MOV     BTM, A        ; BTM ← 1101B
STOP                  ; STOP mode set
NOP
```

4. The high level width of a pulse to be input to the INT4 interrupt (both edge detect) is set. (Pulse width should be less than the BT set value. The BT set value should be 7.8 ms or more.)

<INT4 interrupt routine (MBE = 0)>

```
LOOP:   MOV     XA, BT          ; 1st read
        MOV     BC, XA          ; data store
        MOV     XA, BT          ; 2nd read
        SKE     A, C
        BR      LOOP
        MOV     A, X
        SKE     A, B
        BR      LOOP
        SKT     PORT0.0         ; P00 = 1?
        BR      AA              ; NO
        MOV     XA, BC          ; data store in the data memory
        MOV     BUFF, XA
        CLR1    FLAG            ; data attached 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 store
        SET1    FLAG            ; data attached flag set
        RETI
```

5.4 Watch Timer

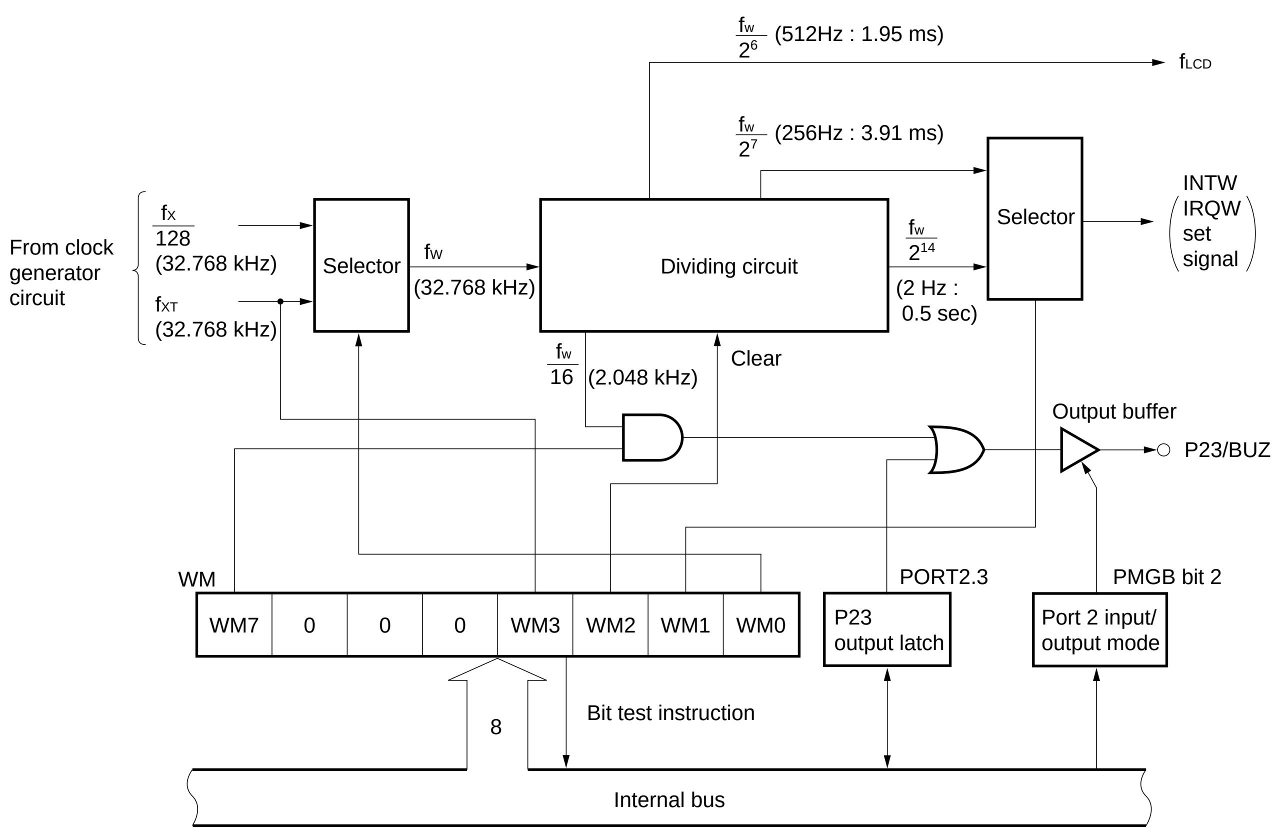
The μPD75308 is equipped with a one-channel watch timer which performs the following functions.

(a) Sets the test flag (IRQW) at 0.5 sec time intervals.
Can release the standby mode by IRQW.
(b) Can set the 0.5 sec interval by either the main system clock or the subsystem clock.
(c) The time interval can be multiplied by 128 (to 3.91 ms) in the fast mode so that program debugging and inspection can be carried out efficiently.
(d) Can generate the fixed frequency (2.048 kHz) to P23/BUZ pin so that a buzzer sound can be generated or the system clock oscillation frequency can be trimmed.
(e) Because the dividing circuit can be cleared, the watch can be started at the zero second.

5.4.1 Watch timer configuration

The watch timer is configured as shown in Figure 5-22.

Figure 5-22. Watch Timer Block Diagram



Remark f_X = 4.194304 MHz and f_{XT} = 32.768 kHz in Parentheses

5.4.2 Watch mode register

The watch mode register (WM) is an 8-bit register to control the watch timer. The format is shown in Figure 5-23.

All bits of the watch mode register, except bit 3, are set by 8-bit manipulation instructions. Bit 3 is used to test the XT1 pin input level (bit test). Data cannot be written. When the $\overline{RESET}$ signal is generated, all bits except bit 3 are cleared to 0.

Example The time is generated by the main system clock (4.19 MHz). Buzzer output is enabled.

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

Figure 5-23. Watch Mode Register Format

Address	7	6	5	4	3	2	1	0	Symbol
F98H	WM7	0	0	0	WM3	WM2	WM1	WM0	WM

Count clock (f_W) select bit

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

Operating mode select bit

WM1	0	Normal watch mode ($\frac{f_W}{2^{14}}$: IRQW is set by 0.5 sec)
	1	Fast watch mode ($\frac{f_W}{2^7}$: IRQW is set by 3.91 ms)

Watch operation enable/disable bit

WM2	0	Watch operation stop (dividing circuit clear)
	1	Watch operation enable

XT1 pin input level (enable for bit test only)

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

BUZ output enable/disable bit

WM7	0	BUZ output disable
	1	BUZ output enable

5.5 Timer/Event Counter

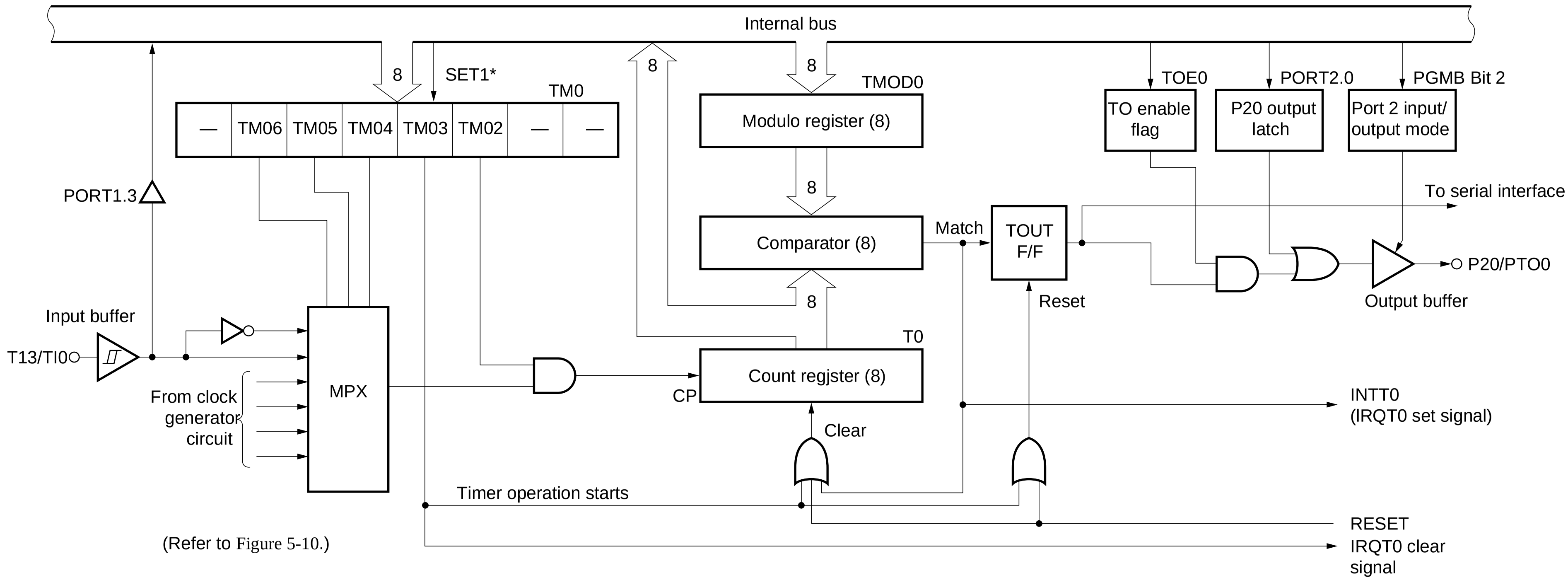
5.5.1 Timer/event counter configuration

The μPD75308 has a one-channel on-chip timer/event counter and is configured as shown in Figure 5-24.

The timer/event counter has the following functions.

(a) Programmable interval timer operation
(b) Output of square wave having any selected frequency to PT00 pin
(c) Event counter operation
(d) Output of TI0 pin input divided by N to PTO0 pin (frequency divider operation)
(e) Serial shift clock supply to serial interface circuit
(f) Count state read function

Figure 5-24. Timer/Event Counter Block Diagram



Remark * indicates instruction execution.

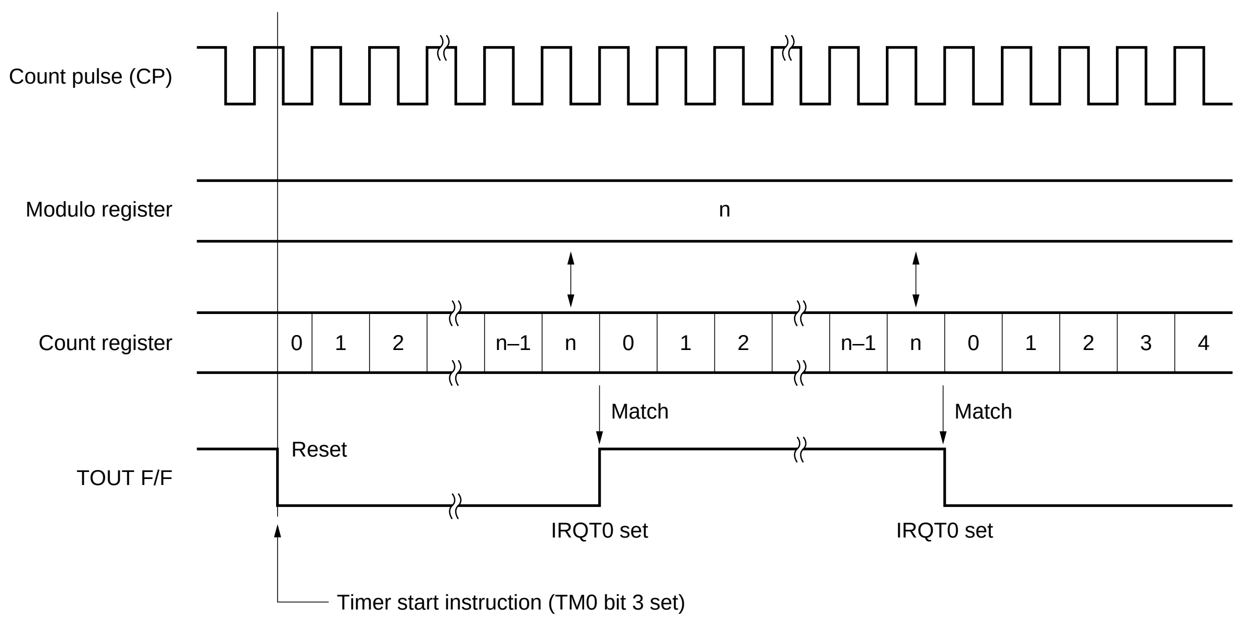
5.5.2 Basic configuration and operations of timer/event counter

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

(1) Count pulse CP is selected by TM0 setting and is input to the 8-bit count register T0.
(2) T0 is a binary, 8-bit up-counter which is incremented by 1 when CP is input. When the $\overline{\text{RESET}}$ signal is generated, TM0 bit 3 is set (upon timer start) or the match signal is generated, T0 is cleared to 0. T0 can be read by an 8-bit memory manipulation instruction. Data cannot be written to T0.
(3) Modulo register TMOD0 is an 8-bit register which determines the number of T0 counts. TMOD0 is set by an 8-bit memory manipulation instruction. Data cannot be read from TMOD0.
When the $\overline{\text{RESET}}$ signal is generated, TMOD0 is initialized to FFH.
(4) The comparator compares T0 content to TMOD0 content. When the contents match, the comparator generates a match signal and sets the interrupt request flag (IRQT0).

Figure 5-25 shows the count operation timing.

Figure 5-25. Count Operation Timing



5.5.3 Timer/event counter mode register (TM0) and timer/event counter output enable flag (TOE0)

The mode register (TM0) is an 8-bit register which controls the timer/event counter. Its format is shown in Figure 5-26.

The timer mode register is set by an 8-bit memory manipulation instruction.

Bit 3 is a timer start bit and can be set independently. It is automatically reset (0) when the timer starts operating.

Examples **1.** The timer is started in the CP = 4.09 kHz interval timer mode.

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

2. The timer is restarted in accordance with the timer mode register setting.

```
SEL     MB15                    ; or CLR1 MBE
SET1    TM0.3                   ; TM0. bit3 ← 1
```

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

The timer/event counter output enable flag (TOE0) is used to enable or disable output of timer out F/F (TOUT F/F) state to PT00 pin.

The timer out F/F (TOUT F/F) is an F/F which is inverted by a match signal coming from the comparator. It is reset by an instruction which sets bit 3 of the timer mode register (TM0).

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

Figure 5-26. Timer/Event Counter Mode Register Format

Address	7	6	5	4	3	2	1	0	
FA0H	—	TM06	TM05	TM04	TM03	TM02	—	—	TM0

Operation Mode

TM02	Count operation
0	Stop (count content hold)
1	Count operation

Timer Start Instruction Bit (TM03)

Clear the counter and IRQT0 flag by writing "1".
If bit 2 has been set (1), the count operation starts.

Count Pluse (CP) Select Bit

TM06	TM05	TM04	Count pulse (CP)
0	0	0	TI0 input rising edge
0	0	1	TI0 input falling edge
1	0	0	$f_X/2^{10}$ (4.09 kHz)
1	0	1	$f_X/2^{8}$ (16.4 kHz)
1	1	0	$f_X/2^{6}$ (65.5 kHz)
1	1	1	$f_X/2^{4}$ (262 kHz)
Other than above			Setting prohibited

Remark f_X = 4.19 MHz in parentheses

Figure 5-27. Timer/Event Counter Output Enable Flag Format

Address	3
FA2H	TOE0

Timer/Event Counter Output Enable Flag (W)

TOE0	
0	Disable
1	Enable

Remark (W): Write only

5.5.4 Timer/event counter operating mode

The count operation stop mode or the count operating mode is set according to the mode register setting.
The following operations can always be carried out irrespectively of the mode register setting.

① TI0 pin signal input and test (P13 dual-purpose pin input test enable)
② Timer out F/F state output to PTO0
③ Modulo register (TMOD0) set
④ Count register (T0) read
⑤ Interrupt request flag (IRQT0) set/clear/test

(a) Count operation stop mode

When TM0 bit 2 is set to 0, this mode is set. In this mode, count operations are not carried out because count pulse (CP) supply to the count register is stopped.

(b) Count operating mode

When TM0 bit 2 is set to 1, this mode is set. In this mode, the count pulse selected by bits 4 to 6 are supplied to the count register and count operations shown in Figure 5-25 are carried out.
Timer operation is normally started in the following sequence.

① The number of counts is set to the modulo register (TMOD0).
② Operating mode, count clock and start instruction are set to the mode register (TM0).

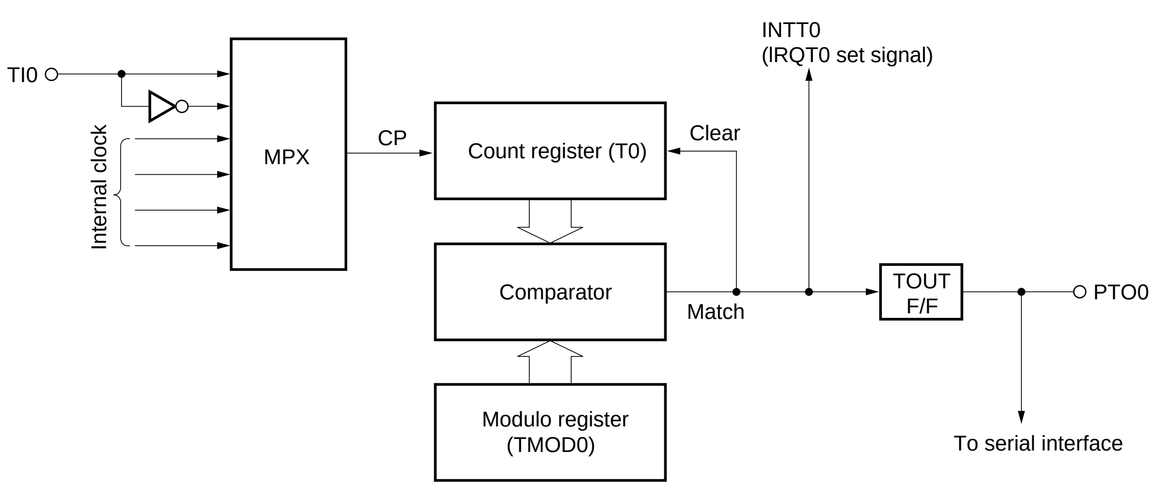
The modulo register is set by an 8-bit data transfer instruction.

Caution Set a value other than 0 to the modulo register.

Example 3FH is set to the modulo register of channel 0.

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

Figure 5-28. Count Operating Mode Operation



5.5.5 Timer/event counter time set

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

$$T_{(SEC)} = \frac{n + 1}{f_{CP}} = (n + 1) \bullet (\text{resolution})$$

$T_{(SEC)}$: Timer set time (sec)
f_{CP} (Hz) : Count pulse frequency (Hz)
n : Modulo register value ($n \neq 0$)

Once set, the timer generates the interrupt request signal (IRQT0) at the set intervals.

Table 5-6 shows the resolution and maximum set time (time to be set when FFH is set to the modulo register).

Example To set up a 30 ms time interval (f_X = 4.194304 MHz).
In this case, the mode with a maximum setting time of 62.5 ms is used.
This gives

$$\frac{30 \text{ ms}}{244 \ \mu\text{s}} = 123 \fallingdotseq 7BH$$

and 7AH is set in the modulo register.

```
SEL   MB15
MOV   XA, #7AH
MOV   TMOD0, XA
```

Table 5-6. Resolution and Maximum Time Set (4.19 MHz)

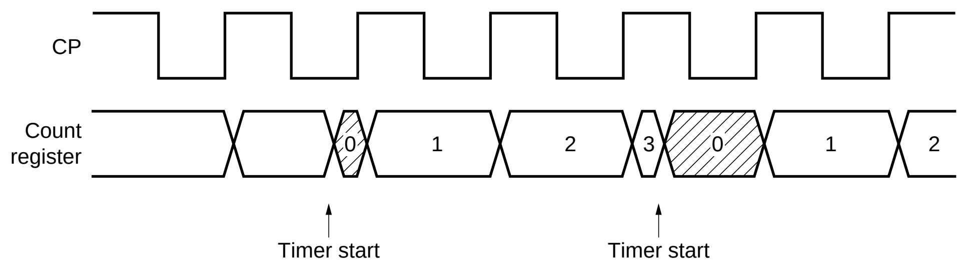
Mode Register			Timer Channel 0	
TM06	TM05	TM04	Resolution	Maximum Set Time
1	0	0	244 μs	62.5 ms
1	0	1	61.1 μs	15.6 ms
1	1	0	15.3 μs	3.91 ms
1	1	1	3.81 μs	977 μs

5.5.6 Precautions relating to timer/event counter application

(1) Timer start error

The time between timer start (TM0.3 set) and generation of a match signal will have a maximum error corresponding to one clock of count pulse (CP) as compared to the value calculated in **5.5.5**. This is because the count register is cleared asynchronously with the CP as shown in Figure 5-29.

Figure 5-29. Timer Start Error



(2) Cautions relating to timer start

Count register T0 and interrupt request flag IRQT0 are normally cleared by timer start (TM0 bit 3 set). If the timer is in the operating mode and IRQT0 set and timer start are carried out simultaneously, IRQT0 may not be cleared. This possesses no problem when IRQT0 is used as a vectored interrupt. However, if IRQT0 is to be tested, a problem occurs because IRQT0 is set although the timer has been started. Thus, when starting the timer at the timing when IRQT0 may be set, stop the timer first (by setting TM0 bit 2 to 0) and restart it, or execute the timer start twice.

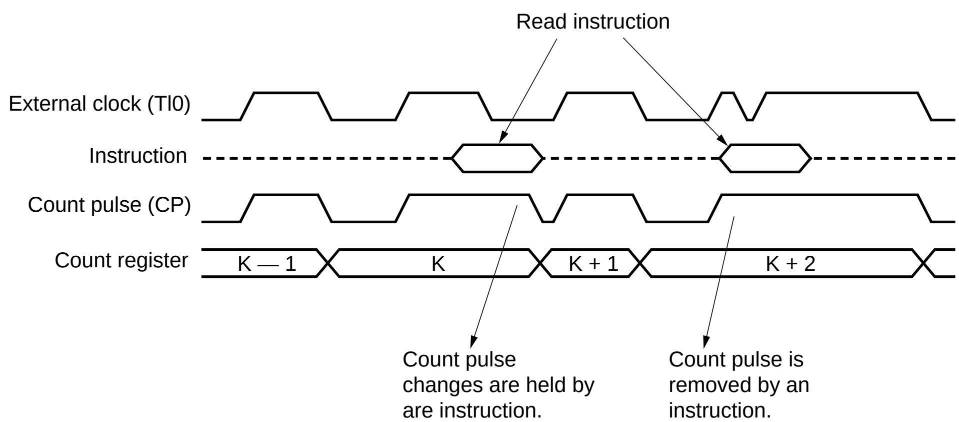
Example Timer start at the timing when IRQT0 may be set

```
SEL     MB15        ; or CLR1 MBE
MOV     XA, #0
MOV     TM0, XA     ; timer stop
MOV     XA, #4CH
MOV     TM0, XA     ; restart
or
SEL     MB15        ; or CLR1 MBE
SET1    TM0.3
SET1    TM0.3       ; restart
```

(3) Count register read errors

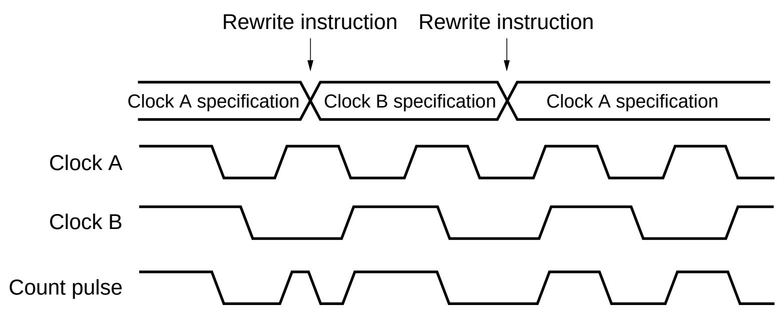
The count register content can always be read by an 8-bit data memory manipulation instruction. While this instruction is in operation, the count pulse and the count register are held from changing. Thus, when TI0 input is used as the count pulse signal source, count pulses are removed by the amount corresponding to the instruction execution time (if the internal clock is used as the count pulse, this pulse removal will not occur because of synchronization with the instruction).

Therefore, when applying TI0 input as the count pulse and reading the count register content, it is necessary to apply a signal having a pulse width which will not lead to an incorrect count if count pulses are removed. In other words, since the count is held for one machine cycle by a read instruction, a pulse to be input to the TI0 pin must have a width larger than the one machine cycle.

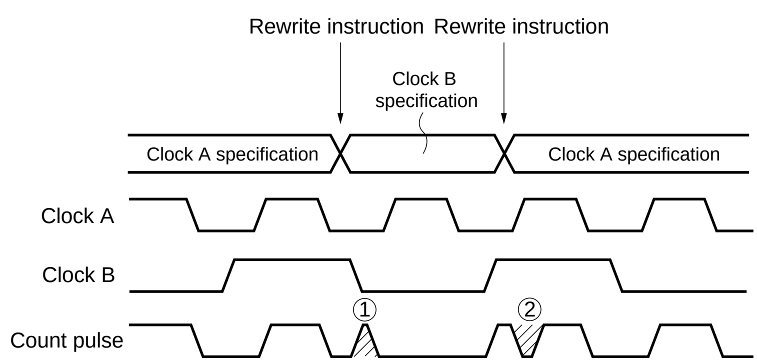


(4) Precaution relating to count pulse change

If the count pulse is changed by rewriting the timer mode register, the specification is validated just after the execution of an instruction.

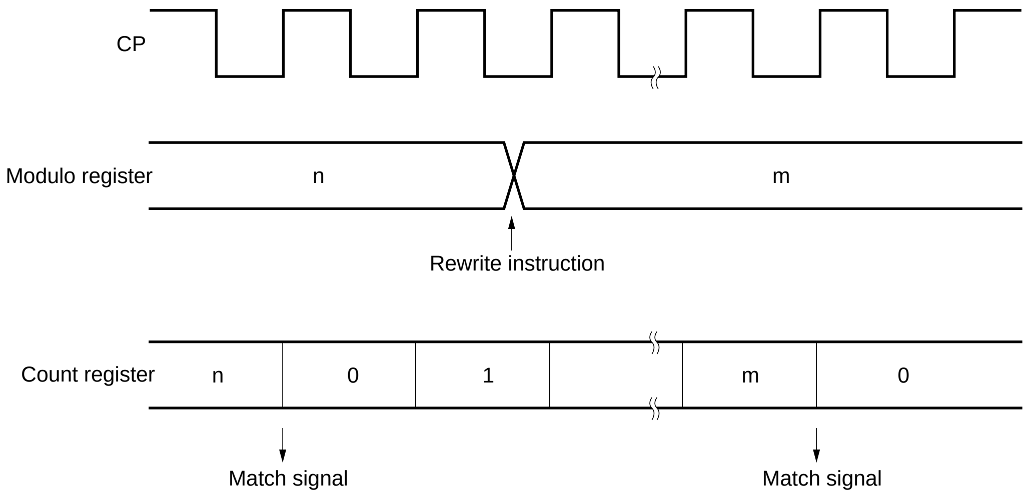


A whiskered count pulse (① or②) may be generated as shown below depending on clock combinations for count pulse change. In such cases, the counting may become incorrect or the counter register content may be destroyed. T0 prevents it from occurring, make sure to set bit 3 of the count mode register to 1 and simultaneously restart the timer when changing the count pulse.

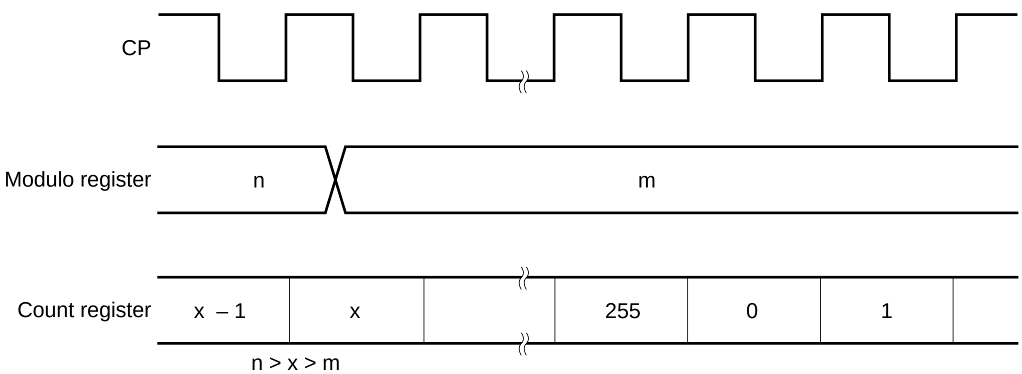


(5) Operations after modulo register change

Modulo register change is carried out upon execution of an 8-bit data memory manipulation instruction.



If the modulo register changed value is smaller than the count register value, the count register continues to count till it overflows. After that, it recounts from 0. Thus, the modulo register post-change value (m) is smaller than the pre-change value (n), it is necessary to first change the modulo register and then restart the timer.



5.5.7 Timer/event counter application

(1) Timer 0 is used as an interval timer to generate interrupts at 50 ms intervals.

- With the high-order 4 bits of the mode register set to 0100B, select the maximum set time of 62.5 ms.
- Set the low-order 4 bits of the mode register to 1100B.
- The modulo register set value is as follows.

$$\frac{50\ \text{ms}}{244\ \mu\text{s}} = 205 \fallingdotseq \text{CDH}$$

<Program example>

```
SEL     MB15              ; or CLR1 MBE
MOV     XA, #0CCH
MOV     TMOD0, XA         ; modulo set
MOV     XA, #01001100B
MOV     TM0, XA           ; mode set and timer start
EI                        ; interrupt enable
EI      IET0              ; timer interrupt enable
```

Remark In this application, TI0 pin can be used as an input pin.

(2) If the number of pulses input from TI0 pin becomes 100, an interrupt is generated (with the pulse set to high active).

- With the high-order 4 bits of the mode register set to 0000, select the rising edge.
- Set the low-order 4 bits of the mode register to 1100B.
- Set the modulo register to 99 = 100 – 1.

<Program example>

```
SEL     MB15              ; or CLR1 MBE
MOV     XA, #100–1
MOV     TMOD0, XA         ; modulo set
MOV     XA, #00001100B
MOV     TM0, XA           ; mode set
EI
EI      IET0              ; INTT0 enable
```

5.6 Serial Interface

5.6.1 Serial interface functions

The μPD75308 incorporates a clocked 8-bit serial interface, with four modes available. The functions of 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 ($\overline{\text{SCK}}$), 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 ($\overline{\text{SCK}}$) 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 $\overline{\text{SCK}}$ and SB0 (or SB1), this mode is compatible with any transfer 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)

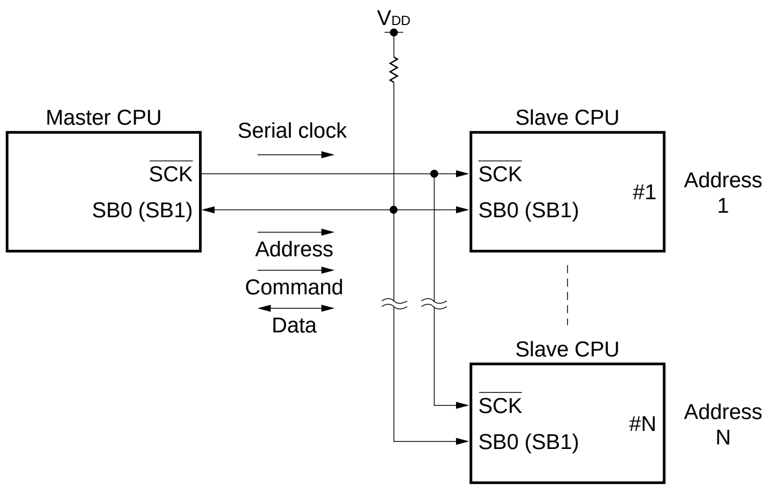
In the SBI mode, communication is performed with multiple devices by means of two lines: The serial clock ($\overline{\text{SCK}}$) and the serial data bus (SB0 or SB1).

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 receive 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 5-30. Example of SBI System Configuration

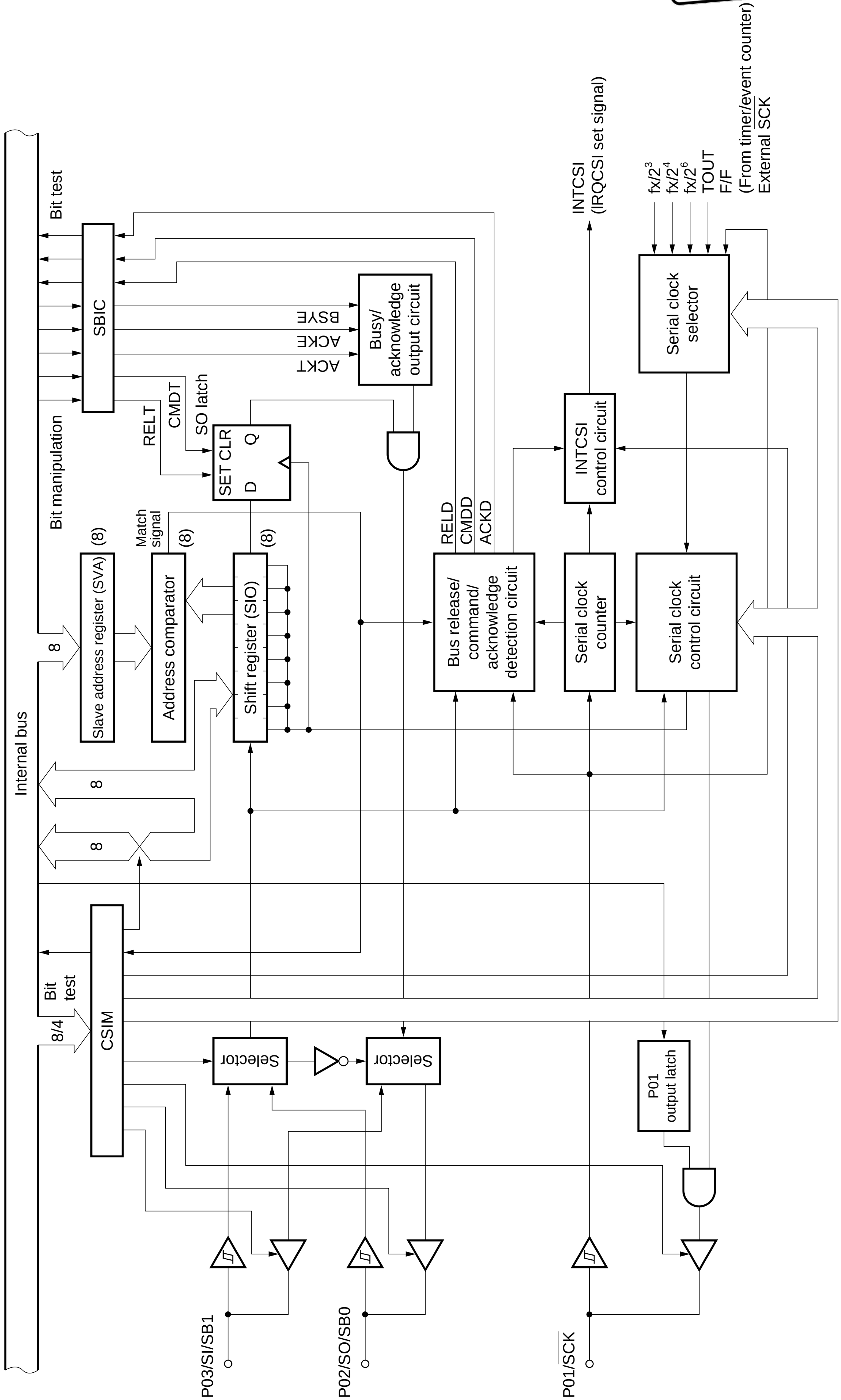


5.6.2 Serial interface configuration

The serial interface block diagram is shown in Figure 5-31.

Phase-out/Discontinued

Figure 5-31. 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 (CSIM)** 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. (Refer to **5.6.3 (2) Serial bus interface control register (SBIC)** 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 operation) in synchronization with the serial clock. Actual transmission/reception operations are controlled by writes to the SIO. (Refer to **5.6.3 (3) Shift register (SIO)** 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. (Refer to **5.6.3 (2) Serial bus interface control register (SBIC)** 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 μPD75308 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 μPD75308 transmits as the master or slave, the SVA register performs error detection. (Refer to **5.6.3 (4) Slave address register (SVA)** for details.)

(8) INTCSI control circuit

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

- **In 3-wire and 2-wire serial I/O mode**

  Generates interrupt requests on each count of 8 serial clock cycles.

- **In SBI mode**

  When WUP[Note] = "0" ...... Generates interrupt requests on each count of 8 serial clock cycles.

  When WUP = "1" .... Generates interrupt requests when the SVA and SIO values match after address reception.

  Note WUP .... Wake-up function specification bit (bit 5 of CSIM)

(9) Serial clock control circuit

Controls the supply of the serial clock to the shift register. Also controls the clock output to the $\overline{\text{SCK}}$ 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 5-32.

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 (refer to **Figure 5-32**). Bit 6 can be tested only, and data written to this bit is invalid.

Reset input clears this register to 00H.

Figure 5-32. Serial Operating Mode Register (CSIM) Format (1/3)

Address	7	6	5	4	3	2	1	0	Symbol
FE0H	CSIE	COI	WUP	CSIM4	CSIM3	CSIM2	CSIM1	CSIM0	CSIM

- CSIM1, CSIM0: Serial clock selection bits (W)
- CSIM4, CSIM3, CSIM2: Serial interface operating mode selection bits (W)
- WUP: Wake-up function specification bit (W)
- COI: Signal from address comparator (R)
- CSIE: Serial interface operation enable/disable specification bit (W)

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

Figure 5-32. Serial Operating Mode Register (CSIM) Format (2/3)

Serial clock selection bit

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 external source			Input
0	1	Timer/event counter output (T0)			Output
1	0	$f_X/2^4$ (262 kHz)		$f_X/2^6$ (65.5 kHz)	
1	1	$f_X/2^3$ (524 kHz)			

Remark ( ): When f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Operating Mode	Shift Register Bit Order	S0 Pin Function	SI Pin Function
x	0	0	3-wire serial I/O mode	SIO_7 to $SIO_0 \leftrightarrow$ XA (Transfer starting from MSB)	SO/P02 (CMOS output)	SI/P03 (Input)
		1		SIO_0 to $SIO_7 \leftrightarrow$ XA (Transfer starting from LSB)		
0	1	0	SBI mode	SIO_7 to $SIO_0 \leftrightarrow$ XA (Transfer starting from MSB)	SB0/P02 (N-ch open-drain I/O)	P03 input
1					P02 input	SB1/P03 (N-ch open-drain I/O)
0	1	1	2-wire serial I/O mode	SIO_7 to $SIO_0 \leftrightarrow$ XA (Transfer starting from MSB)	SB0/P02 (N-ch open-drain I/O)	P03 input
1					P02 input	SB1/P03 (N-ch open-drain I/O)

Remark x: Don't care

Figure 5-32. Serial Operating Mode Register (CSIM) Format (3/3)

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). SB0/SBI is high impedance.

Caution 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[Note]	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.

Note 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 and SI/SB1 Pins
CSIE	0	Shift operation disabled	Cleared	Retained	Port 0 function only
	1	Shift operation enabled	Count operation	Settable	Function in each mode and 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	Operating-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

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

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

3. The following procedure should be used to clear CSIE during a serial transfer.
1. Clear the interrupt enable flag to set the interrupt disable state.
2. Clear CSIE.
3. Clear the interrupt request flag.

Examples 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 5-33.

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.

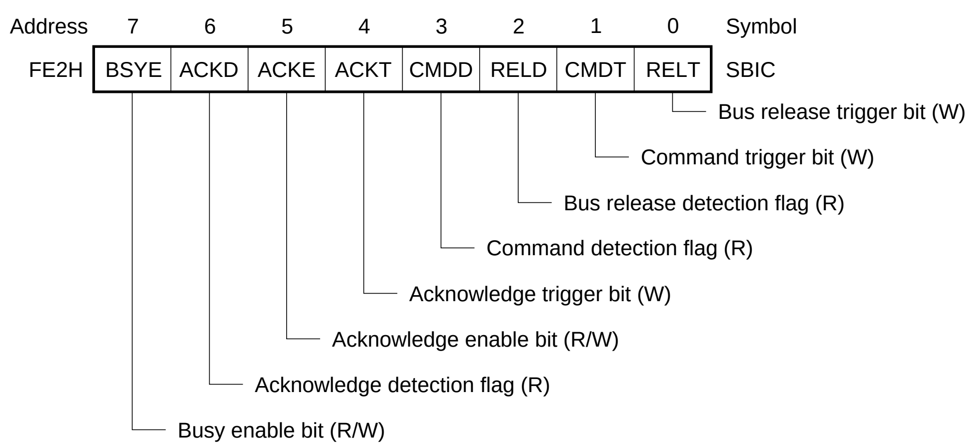
SBIC is manipulated by bit-manipulation instructions; it cannot be manipulated by 4-bit or 8-bit memory manipulation instructions.

Read/write capability differs from bit to bit (refer to **Figure 5-33**).

Reset input clears this register to 00H.

Caution 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 5-33. Serial Bus Interface Control Register (SBIC) Format (1/3)



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

Figure 5-33. Serial Bus Interface Control Register (SBIC) Format (2/3)

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).

Caution 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).

Caution 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 (refer to **Figure 5-32**) ④ When SVA and SIO do not 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 (refer to **Figure 5-32**)	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 SCK. After the $\overline{ACK}$ signal is output, ACKT is automatically cleared (0). **Cautions 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.**

Figure 5-33. Serial Bus Interface Control Register (SBIC) Format (3/3)

Acknowledge enable bit (R/W)

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

Acknowledge detection flag (R)

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

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 $\overline{\text{SCK}}$ immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of $\overline{\text{SCK}}$ following the acknowledge signal.

Examples 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      ; Test RELD
    BR      !ADRS
    SKT     CMDD      ; Test CMDD
    BR      !DATA
CMD  :  ......................  ; Command interpretation
DATA :  ......................  ; Data processing
ADRS :  ......................  ; Address decoding
```

(3) Shift register (SIO)

The configuration around the shift register is shown in Figure 5-34. SIO is an 8-bit 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.

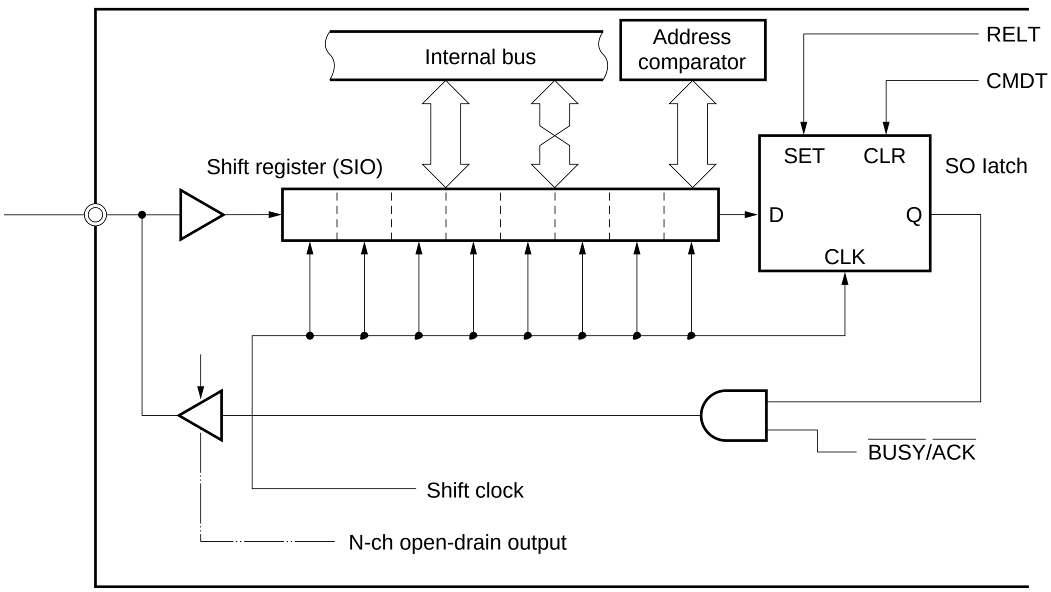
In transmission, the data written to SIO is output to the serial output (SO) or the serial data bus (SB0/SB1). In reception, data is read into SIO from the serial input (SI) or SB0/SB1.

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 5-34. Configuration Around Shift Register



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

- When the serial interface operation 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 purposes. 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 SVA register has the following two functions:

(a) Slave address detection

[In SBI mode]

Used when the μPD75308 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 μPD75308.

(b) Error detection

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

The SVA performs error detection in the following cases.

- When the μPD75308 transmits addresses, commands or data as the master device.
- When the μPD75308 receives data as a slave device.

(Refer to **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/$\overline{\text{SCK}}$ can be used as an input port depending on the setting of the serial operating mode register.

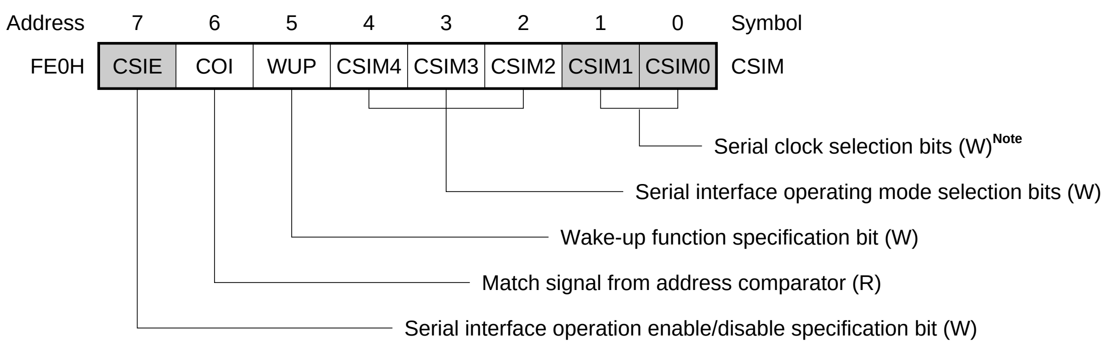
(1) Register setting

Operation-halted mode setting is performed by the serial operating mode register (CSIM) (refer to **5.6.3 (1) Serial operating mode register (CSIM)** for the full configuration of CSIM).

CSIM is manipulated by 8-bit manipulation instructions, but bit manipulation of CSIE is also possible. Manipulation is also possible using the bit name.

Reset input clears CSIM to 00H.

Shading indicates bits used in the operation-halted mode.



Note Allow selection of P01/$\overline{\text{SCK}}$ pin status.

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

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 and 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.

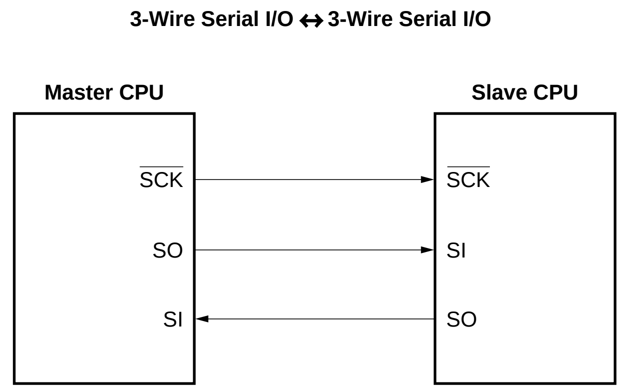
① Clear the interrupt enable flag (IECSI) to set the interrupt disable state.
② Clear CSIE.
③ Clear the interrupt request flag (IRQCSI).

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, μPD7500 series, 87AD series, etc.

Communication is performed using three lines: The serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

Figure 5-35. Example of 3-Wire Serial I/O System Configuration



(1) Register setting

3-wire serial I/O mode operation is set by means of the following two registers:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

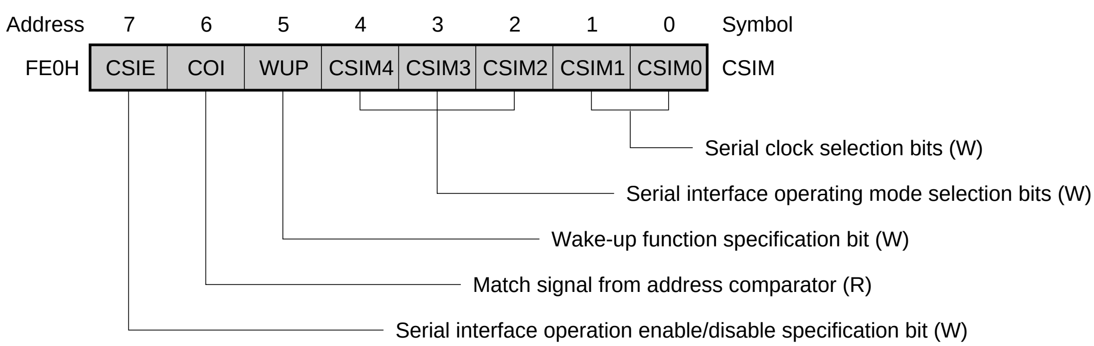
(a) Serial operating mode register (CSIM)

When the 3-wire serial I/O mode is used, CSIM is set as shown below (refer to **5.6.3 (1) Serial operating mode register (CSIM)** for the full configuration of CSIM).

CSIM is manipulated by 8-bit manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shading indicates bits used in the 3-wire serial I/O mode.



Remark (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)	

Remark ( ): Operating with f_X = 4.19 MHz

Serial interface operating mode selection bits (W)

CSIM4	CSIM3	CSIM2	Shift Register Bit Order	S0 Pin Function	SI Pin Function
x	0	0	$SIO_{7-0} \leftrightarrow XA$ (MSB-first transfer)	SO/P02 (CMOS output)	SI/P03 (Input)
		1	$SIO_{0-7} \leftrightarrow XA$ (LSB-first transfer)		

Remark 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[Note]	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.

Note 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 and SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode and 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 (refer to **5.6.3 (2) Serial bus interface control register (SBIC)** for the full configuration of SBIC).

SBIC is manipulated by bit manipulation instruction.

Reset input clears the SBIC register to 00H.

Shading indicates bits used in the 3-wire serial I/O mode.

Address	7	6	5	4	3	2	1	0	Symbol
FE2H	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	SBIC

Should not be used in 3-wire serial I/O mode. (bits 7 to 2)

Bus release trigger bit (W) (bit 0, RELT)

Command trigger bit (W) (bit 1, CMDT)

Remark (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).

Caution Bits other than RELT and CMDT should not be used in the 3-wire serial I/O mode.

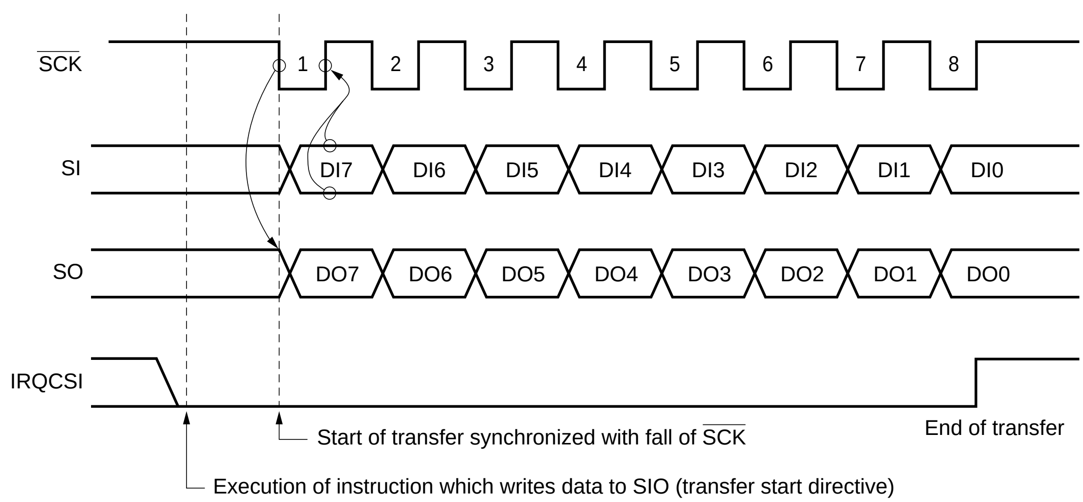
(2) Communication operation

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 from 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 5-36. 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) (refer to **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.

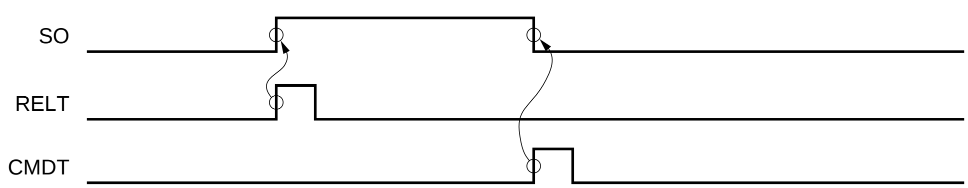
Table 5-7. Serial Clock Selection and Use (In 3-Wire Serial I/O Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial Transfer Start	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
1	1	$f_X/2^3$			High-speed serial transfer

(4) Signals

RELT and CMDT operation is shown in Figure 5-37.

Figure 5-37. RELT and CMDT Operations



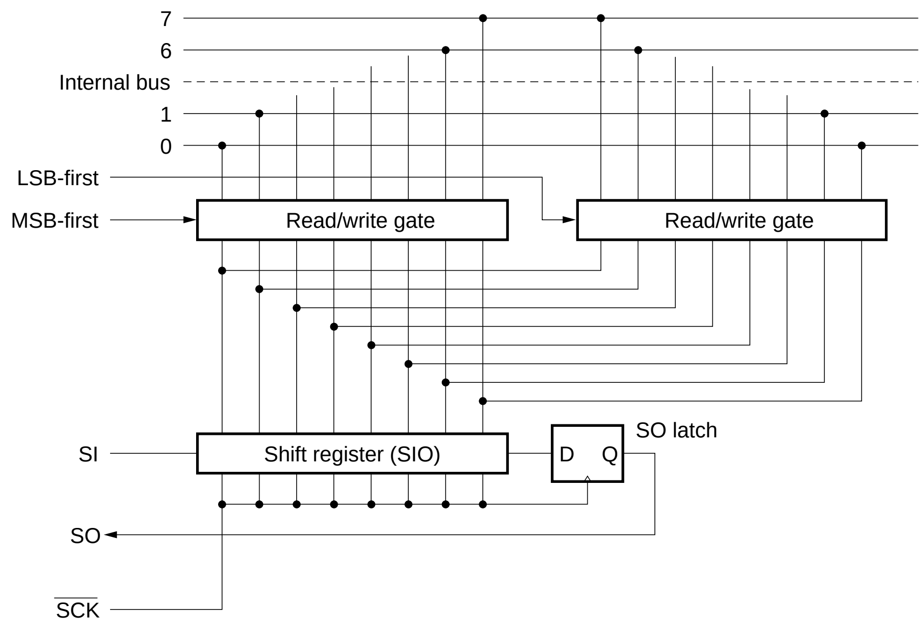
(5) MSB/LSB-first switching

The 3-wire serial I/O mode includes a function for selecting MSB-first or LSB-first transfer.

Figure 5-38 shows the shift register (SIO) and internal bus configuration. As shown in the figure, 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 5-38. 3-Wire I/O Mode Configuration



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 $\overline{\text{SCK}}$ is high.

Caution 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
```

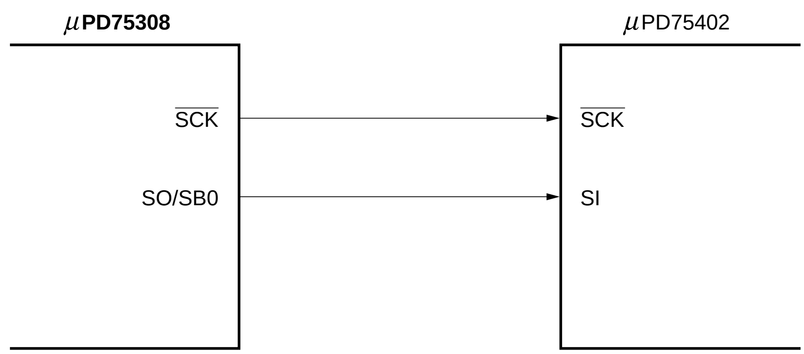
(7) 3-wire serial I/O mode applications

Examples 1. To transfer data MSB-first (master operation) using a 262 kHz transfer clock (when operating at 4.19 MHz).

<Sample program>

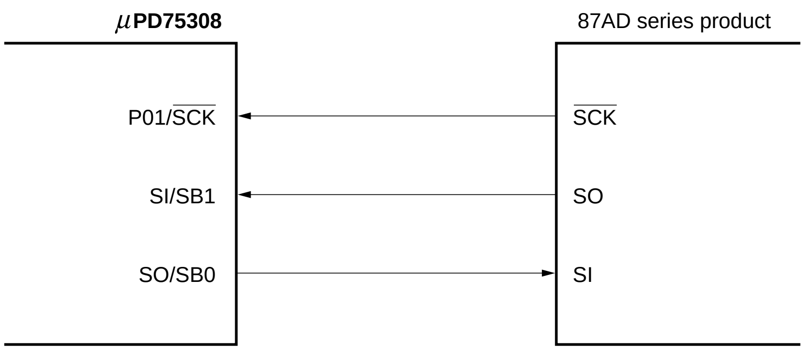
```
CLR1   MBE
MOV    XA, #10000010B
MOV    CSIM, XA        ; transfer mode setting
MOV    XA, TDATA       ; TDATA is transfer data storage address
MOV    SIO, XA         ; transfer data setting and start of transfer
```

Caution 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 μPD75308 SI/SB1 pin can be used for input.

Examples 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 operations is used.)



<Sample program>

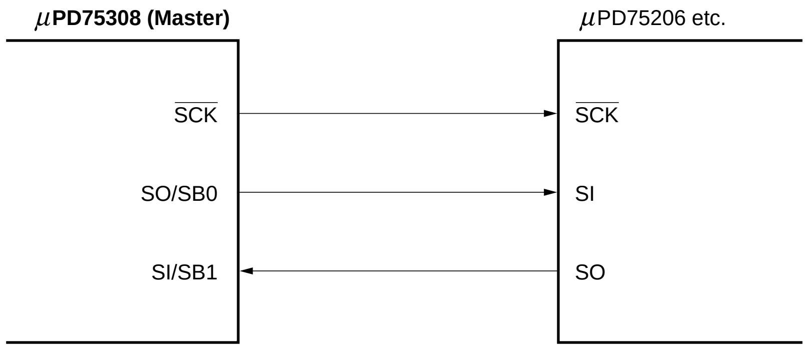
Main routine

```
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 and start of transfer
EI      IECSI
EI
```

Interrupt routine (MBE = 0)

```
MOV     XA, TDATA
XCH     XA, SIO      ; receive data-send data, start of transfer
MOV     RDATA, XA    ; receive data save
RETI
```

Examples 3. To transmit/receive data at high speed using a 524 kHz transfer clock (when operating at 4.19 MHz).



<Sample program> ... Master side

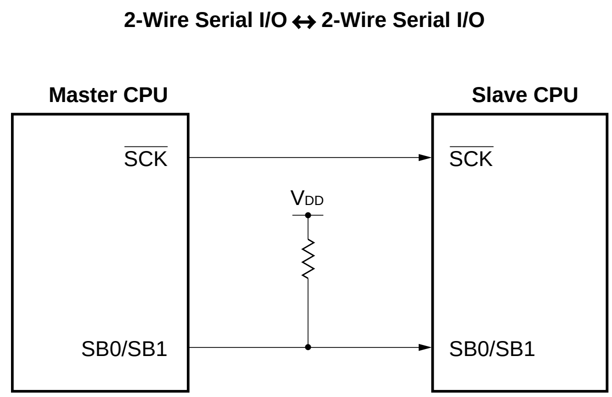
```
     CLR1      MBE
     MOV       XA, #10000011B
     MOV       CSIM, XA          ; transfer mode setting
     MOV       XA,  TDATA
     MOV       SIO, XA           ; transfer data setting and start of transfer
       ⋮
LOOP:  SKTCLR    IRQCSI          ; IRQCSI test
       BR        LOOP
       MOV       XA, SIO         ; receive data fetch
```

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 5-39. Example of 2-Wire Serial I/O System Configuration



(1) Register setting

2-wire serial I/O mode operation can be set by means of the following two registers:

- 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 (refer to **5.6.3 (1) Serial operating mode register (CSIM)** for the full configuration of CSIM).

CSIM is manipulated by 8-bit manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shading indicates bits used in the 2-wire serial I/O mode.

Address	7	6	5	4	3	2	1	0	Symbol
FE0H	CSIE	COI	WUP	CSIM4	CSIM3	CSIM2	CSIM1	CSIM0	CSIM

- CSIM1, CSIM0: Serial clock selection bits (W)
- CSIM4 to CSIM2: Serial interface operating mode selection bits (W)
- WUP: Wake-up function specification bit (W)
- COI: Match signal from address comparator (R)
- CSIE: Serial interface operation enable/disable specification bit (W)

Remark (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	$f_X/2^6$ (65.5 kHz)	
1	1		

Remark ( ): Operating with 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	$SIO_{7-0} \leftrightarrow XA$ (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[Note]	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.

Note 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 and SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode and 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 (refer to **5.6.3 (2) Serial bus interface control register (SBIC)** for the full configuration of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

Shading indicates bits used in the 2-wire serial I/O mode.

Address	7	6	5	4	3	2	1	0	Symbol
FE2H	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	SBIC

Bits 7 to 2: Should not be used in 2-wire serial I/O mode.

Bit 0 (RELT): Bus release trigger bit (W)

Bit 1 (CMDT): Command trigger bit (W)

Remark (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).

Caution Bits other than RELT and CMDT should not be used in the 2-wire serial I/O mode.

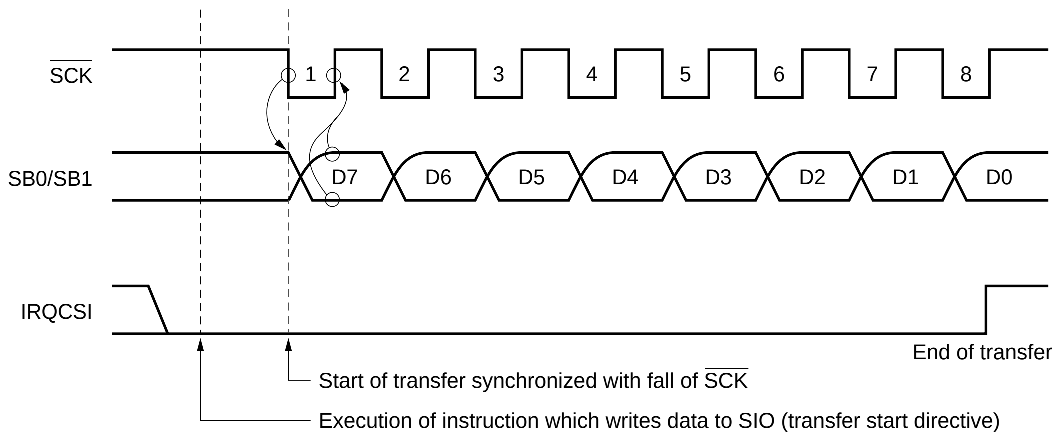
(2) Communication operation

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 ($\overline{\text{SCK}}$). 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 from the SB0 (or SB1) pin is latched in the shift register on the rise of $\overline{\text{SCK}}$.

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 5-40. 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 $\overline{\text{SCK}}$ pin output level is controlled by manipulating the P01 output latch in the output mode (internal system clock mode) (refer to **5.6.8 $\overline{\text{SCK}}$ 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.

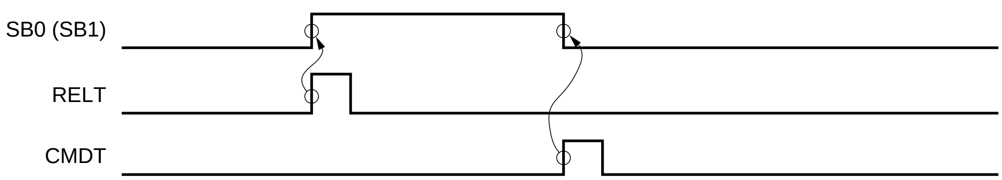
Table 5-8. Serial Clock Selection and Use (In 2-Wire Serial I/O Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial Transfer Start	Use
CSIM 1	CSIM 0	Source	Serial Clock Mask		
0	0	External $\overline{\text{SCK}}$	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 $\overline{\text{SCK}}$ 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 5-41.

Figure 5-41. RELT and CMDT Operations



(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 $\overline{\text{SCK}}$ is high.

Cautions 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.

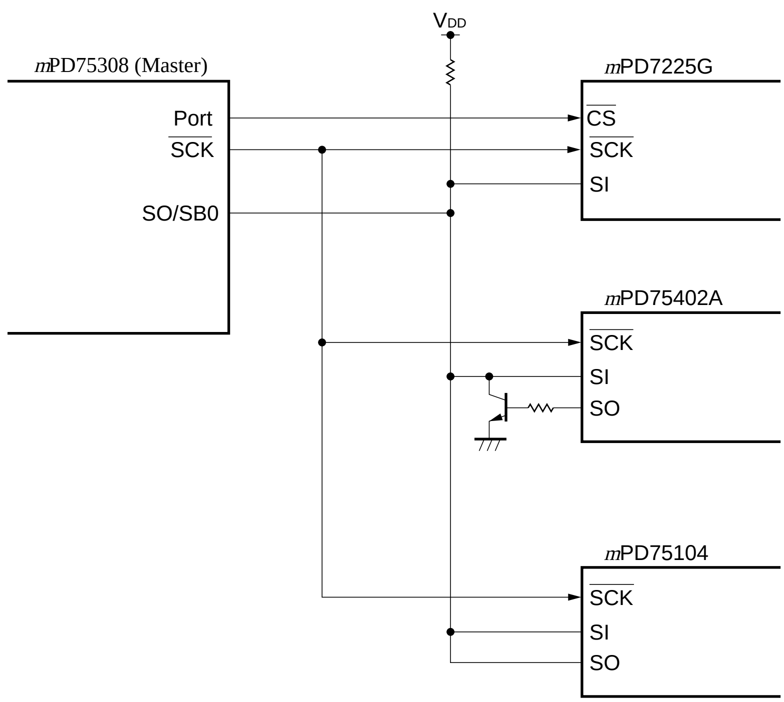
(b) Use of slave address register (SVA)

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 To configure a system with the μPD75308 as the master and the μPD75104, μPD75402A and μPD7225G connected as slaves.



In the bus configuration shown in this example connection is made via the SI pin and SO pin. When serial data is not output, FFH is written into the shift register beforehand and a high-level signal output to the SO pin, and the output buffer is turned off to release the bus.

Since the μPD75402A 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.

When each microcontroller outputs data is determined in advance.

The serial clock is output by the μPD75308, which is the master microcontroller, and the other slave microcontroller all operate on an external clock.

Phase-out/Discontinued

5.6.7 SBI mode operation

The SBI (serial bus interface) is a high-speed serial interface which conforms to the 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 bus system to enable communication to be performed with multiple devices using two signal lines. Consequently, when a serial bus is configured with multiple microcontrollers and peripheral ICs, it is possible to reduce the number of ports used and the amount of wiring on the substrate.

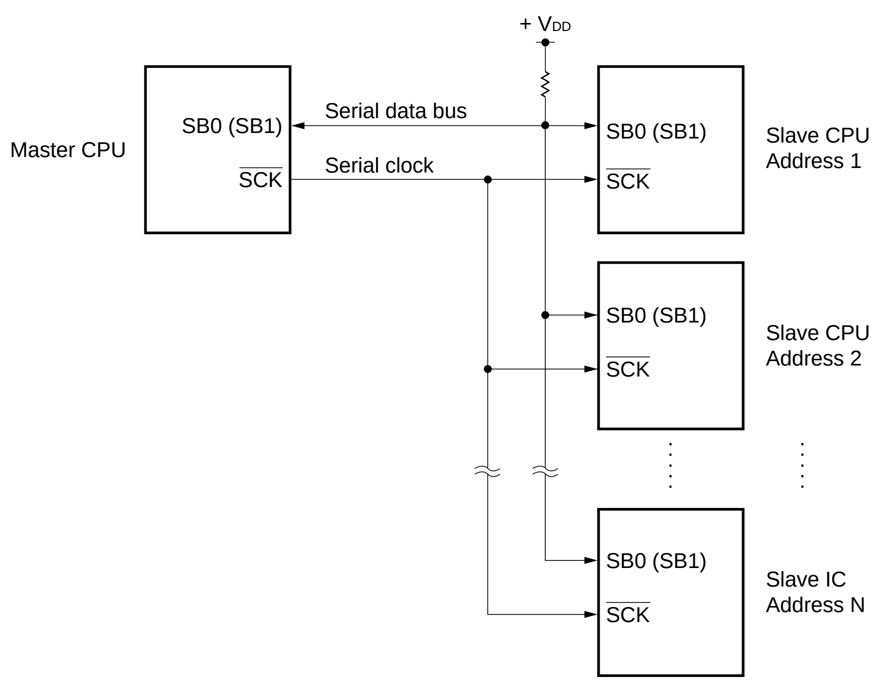
The master can output to a slave on the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. The slave can determine by hardware whether the receive data is an address, command or actual data. This function allows the serial interface control portion of the application program to be simplified.

SBI functions are incorporated in a number of devices including the 75X series, and 78K series 8-bit single-chip microcontrollers.

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 5-42.

In the SBI the SB0 (SB1) serial data bus pin is an open-drains output and thus the data bus line is in the wired-OR state. The serial data bus line requires a pull-up resistor.

Figure 5-42. Example of SBI Serial Bus Configuration



Caution 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 the Chip Select signal, command/data differentiation, busy status recognition, etc. If these controls are 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, $\overline{\text{SCK}}$, and the serial data bus, SB0 (SB1). As a result, the number of microcontroller ports and the amount of substrate wiring can be significantly reduced.

SBI functions are described below.

(a) Address/command/data identification 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 ($\overline{\text{ACK}}$) control function

Controls the acknowledge signal used to confirm serial data reception.

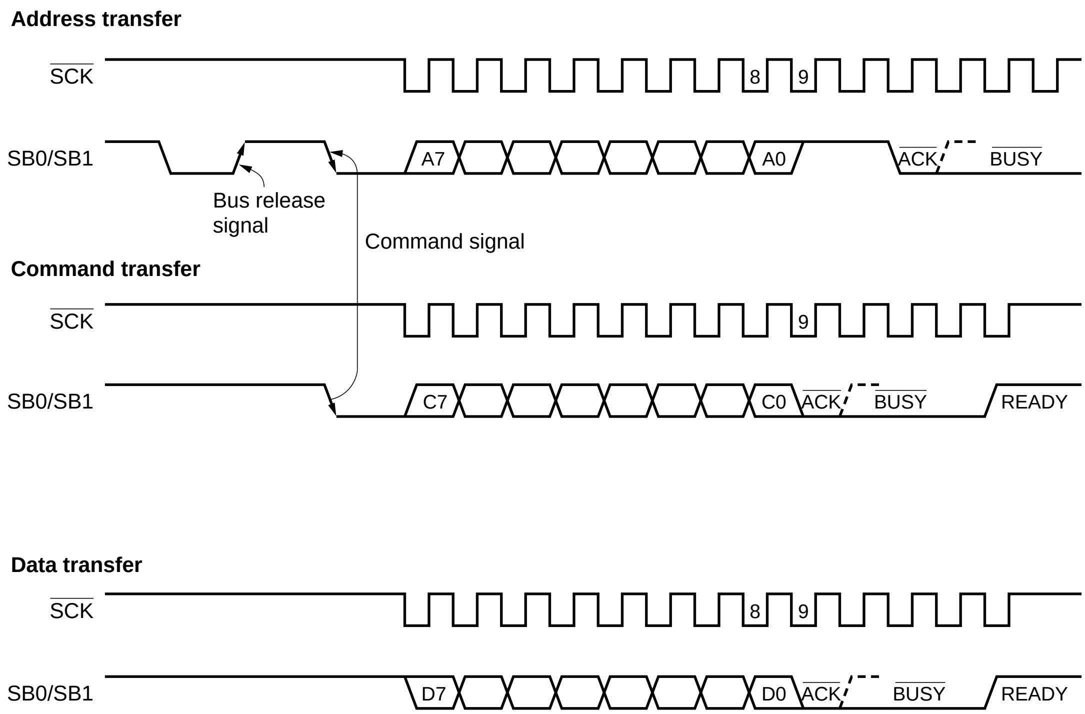
(e) Busy signal ($\overline{\text{BUSY}}$) control function

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. Serial data forms a frame using the configuration below.
Address, command and data timing is shown in Figure 5-43.

Figure 5-43. SBI Transfer Timing

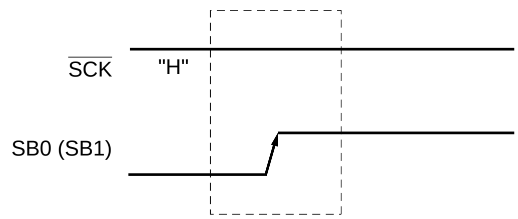


The bus release signal and command signal are output by the master. The $\overline{\text{BUSY}}$ signal is output by the slave. $\overline{\text{ACK}}$ can be output by either the master or slave (normally output by the 8-bit data receiver).
The serial clock is output by the master continuously from the start of an 8-bit data transfer until $\overline{\text{BUSY}}$ 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 5-44. Bus Release Signal

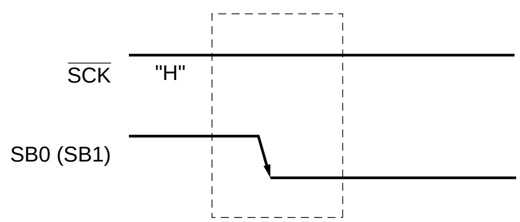


The bus release signal indicates that the master is about to send an address to a slave. Slaves incorporate hardware to detect the bus release signal.

(b) Command signal (CMD)

The command signal indicates that the SB0 (SB1) line has changed from 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 5-45. Command Signal

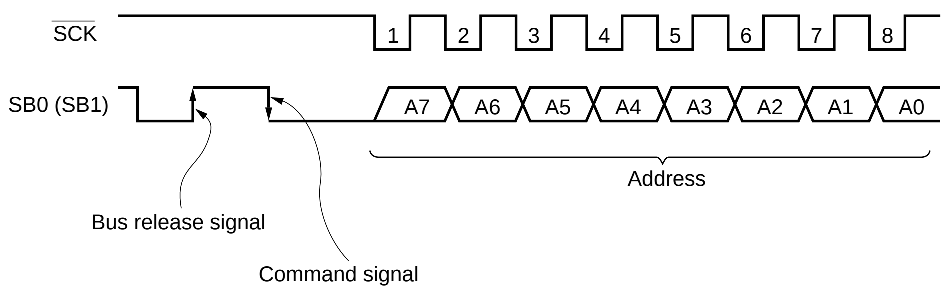


Slave incorporate hardware to detect the command signal.

(c) Address

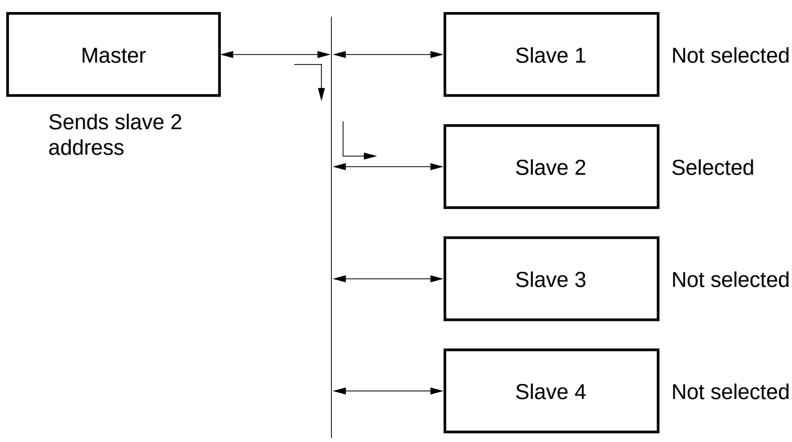
An address is 8-bit data output by the master to slaves connected to the bus line in order to select a particular slave.

Figure 5-46. 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 5-47. Slave Selection by Address



(d) Command and data

The master sends a command or data to the slave selected by address transmission.

Figure 5-48. Command

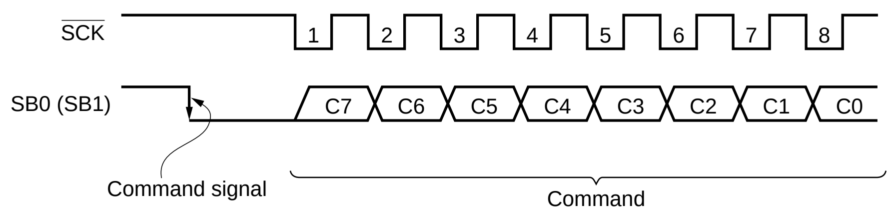
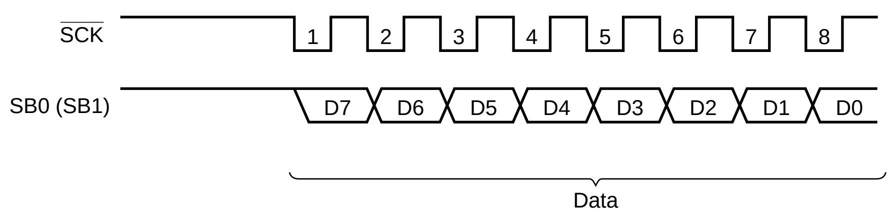


Figure 5-49. Data



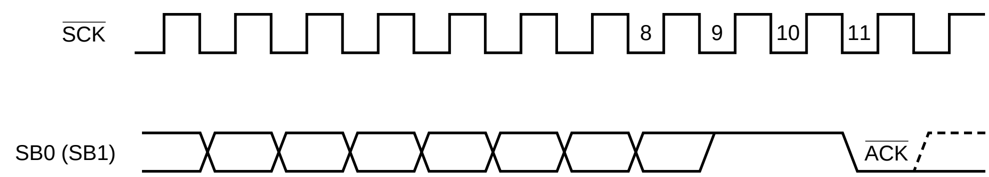
The 8-bit data following the command signal is defined as a command. 8-bit data with no command signal is defined as data. The way in which commands and data are used can be freely decided according to the communication specifications.

(e) Acknowledge signal ($\overline{\text{ACK}}$)

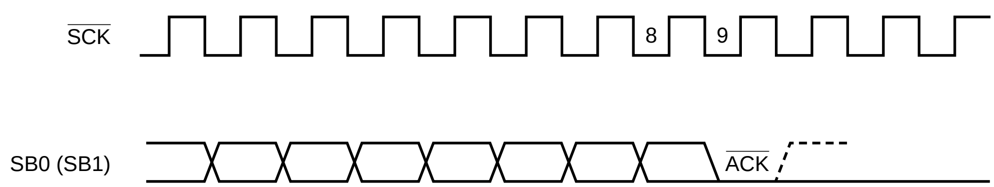
The acknowledge signal is used to confirm serial data reception between the sender and receiver.

Figure 5-50. Acknowledge Signal

[When output in synchronization with 11th $\overline{\text{SCK}}$ clock cycle]



[When output in synchronization with 9th $\overline{\text{SCK}}$ clock cycle]



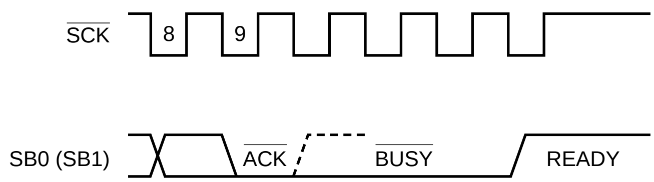
The acknowledge signal is an one-shot pulse synchronized with the fall of $\overline{\text{SCK}}$ after an 8-bit data transfer. Its position is arbitrary and it can be synchronized with any $\overline{\text{SCK}}$ clock cycle.

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 ($\overline{\text{BUSY}}$), ready signal (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 5-51. Busy Signal and Ready Signal



With the SBI a slave reports its busy status to the master by driving the SB0 (SB1) 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 $\overline{\text{SCK}}$. When the busy signal is released the master automatically terminates output of the $\overline{\text{SCK}}$ serial clock.
When the busy signal is released and the ready signal state is entered the master can start the next transfer.

(3) Register setting

SBI mode operation can be set by means of the following two registers:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

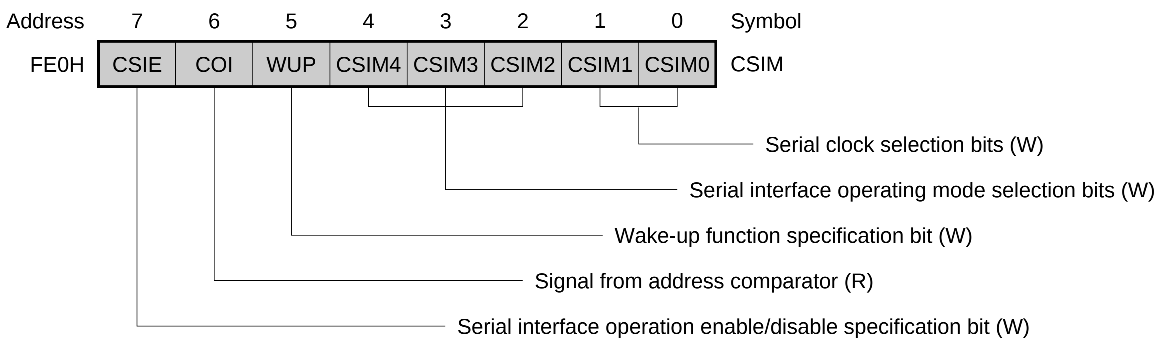
(a) Serial operating mode register (CSIM)

When the SBI mode is used, CSIM is set as shown below (refer to **5.6.3 (1) Serial operating mode register (CSIM)** for the full configuration of CSIM).

CSIM is manipulated by 8-bit manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

Shading indicates bits used in the SBI mode.



Remark (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)	

Remark ( ): Operating with 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	$SIO_{7-0} \leftrightarrow XA$ (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 in SBI mode mask state.
	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). SB0/SB1 is high impedance.

Caution **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[Note]	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.

Note 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 and port 0 function

(b) Serial bus interface control register (SBIC)

When the SBI mode is used, SBIC is set as shown below (refer to **5.6.3 (2) Serial bus interface control register (SBIC)** for the full configuration of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

Shading indicates bits used in the SBI mode.

Address	7	6	5	4	3	2	1	0	Symbol
FE2H	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	SBIC

- RELT: Bus release trigger bit (W)
- CMDT: Command trigger bit (W)
- RELD: Bus release detection flag (R)
- CMDD: Command detection flag (R)
- ACKT: Acknowledge trigger bit (W)
- ACKE: Acknowledge enable bit (R/W)
- ACKD: Acknowledge detection flag (R)
- BSYE: Busy enable bit (R/W)

Remark (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).

Caution 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).

Caution 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 (refer to **Figure 5-32**) 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 (refer to **Figure 5-32**)	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). **Cautions 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 acknowledge signal is output in synchronization with the 9th $\overline{\text{SCK}}$ clock cycle.
		When set after end of transfer	The acknowledge signal is output is synchronization with $\overline{\text{SCK}}$ immediately after execution of the setting instruction.

Acknowledge detection flag (R)

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

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 $\overline{\text{SCK}}$ immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of $\overline{\text{SCK}}$ 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 5-9. Serial Clock Selection and Use (In SBI Mode)

Mode Register		Serial Clock		Possible Timing for Shift Register R/W and Serial Transfer Start	Use
CSIM1	CSIM0	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, but 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 5-52 to 5-57, and SBI signals are listed in Table 5-10.

Figure 5-52. RELT, CMDT, RELD, and CMDD Operation (Master)

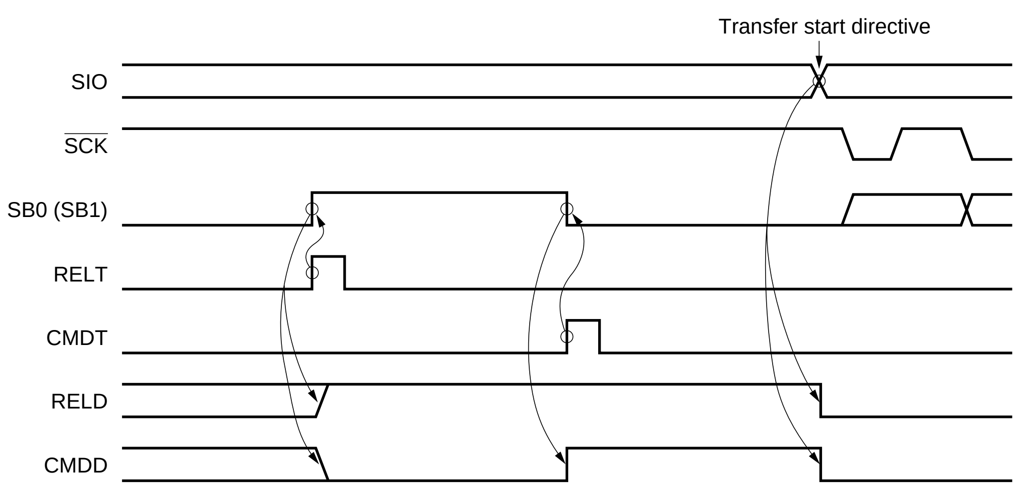


Figure 5-53. RELT, CMDT, RELD, and CMDD Operation (Slave)

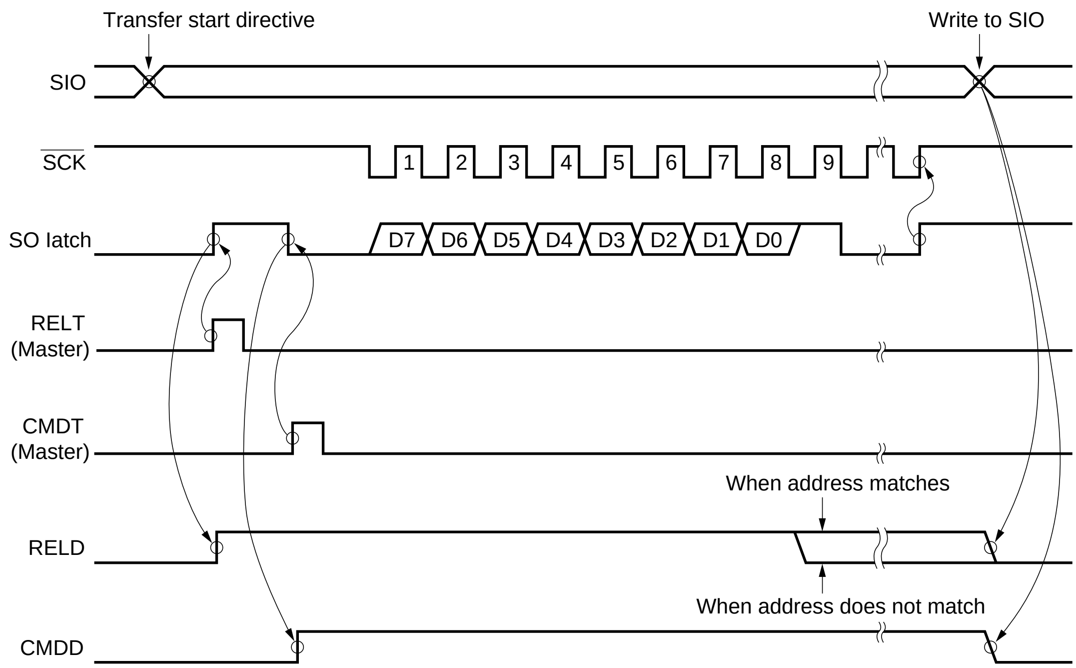
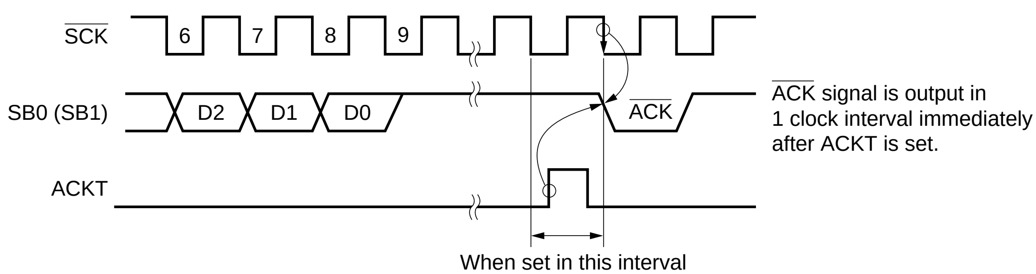


Figure 5-54. ACKT Operation

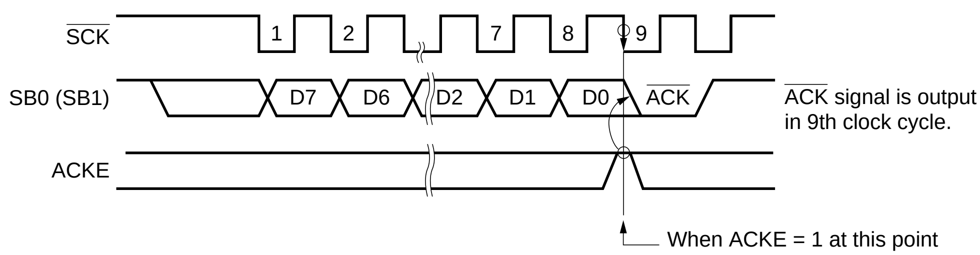


Caution ACKT must not be set before the end of a transfer.

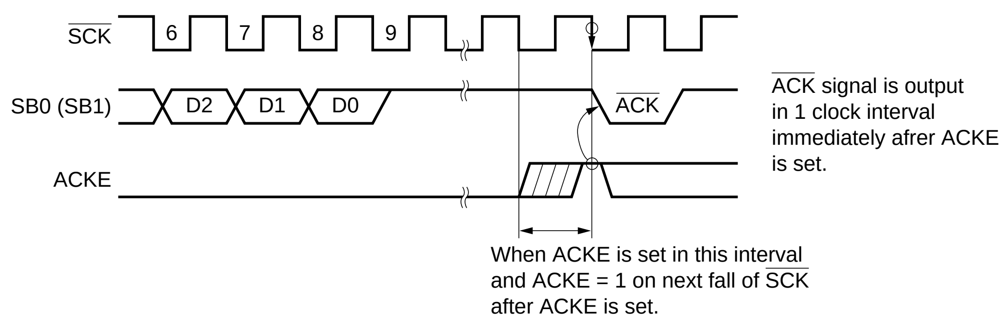
Phase-out/Discontinued

Figure 5-55. ACKE Operation

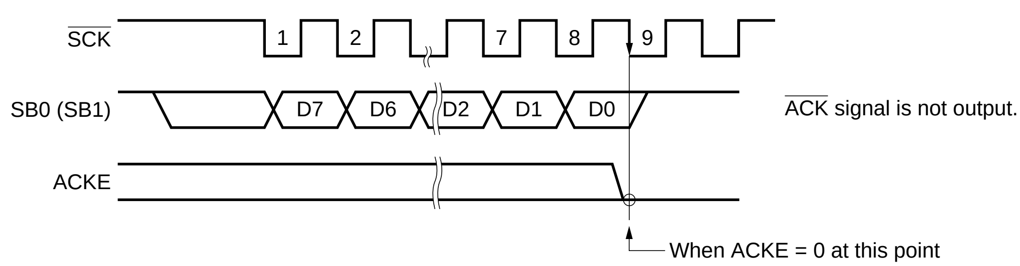
(a) When ACKE = 1 on completion of transfer



(b) When ACKE is set after completion of transfer



(c) When ACKE = 0 on completion of transfer



(d) When ACKE = 1 interval is short

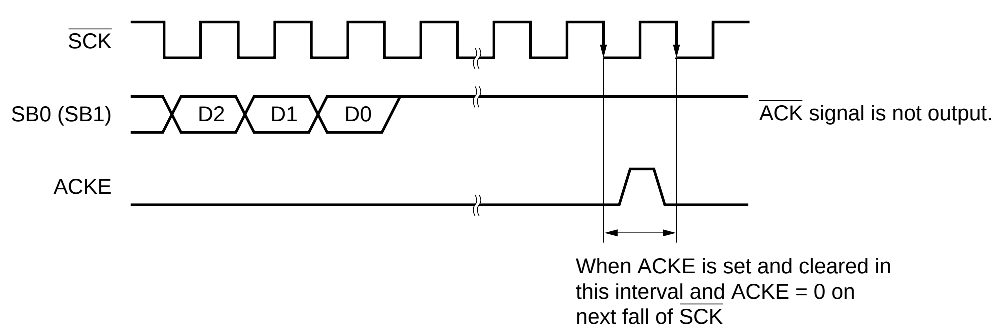
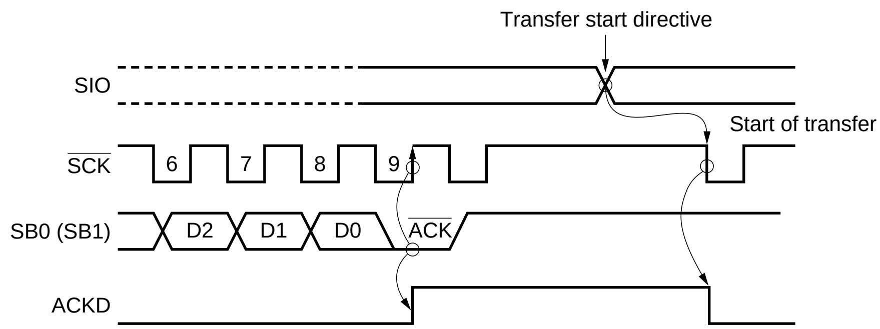
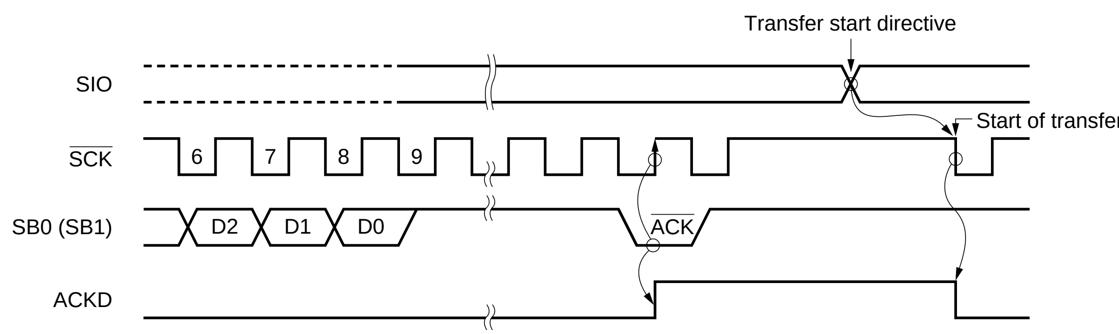


Figure 5-56. ACKD Operation

(a) When $\overline{\text{ACK}}$ signal is output in 9th $\overline{\text{SCK}}$ clock interval



(b) When $\overline{\text{ACK}}$ signal is output after 9th $\overline{\text{SCK}}$ clock interval



(c) Clearing timing when transfer start directive is given during busy state

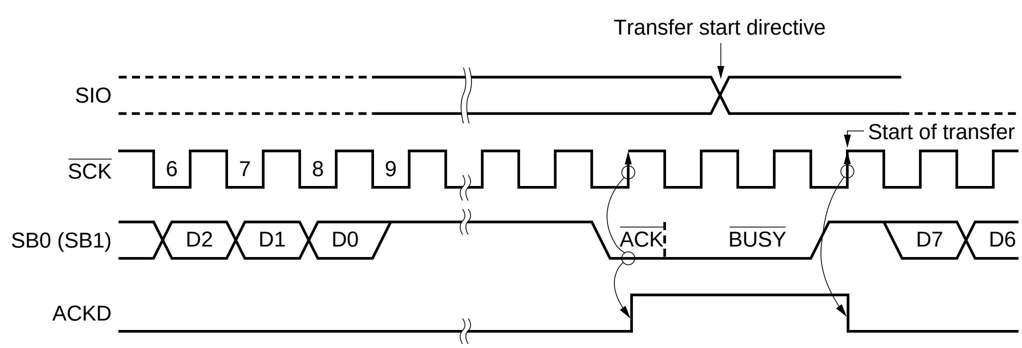
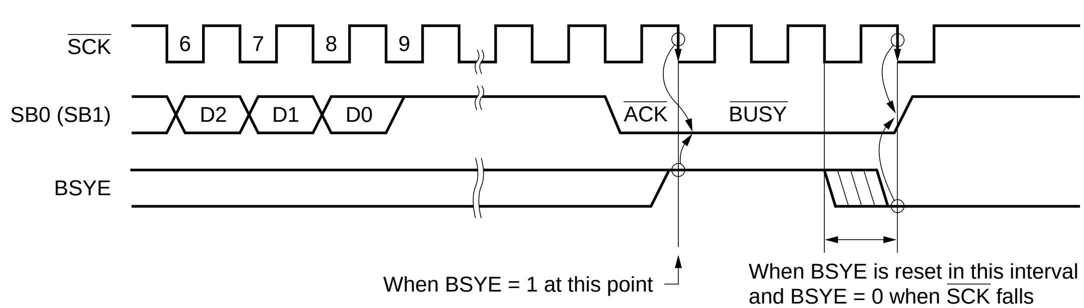


Figure 5-57. BSYE Operation



Phase-out/Discontinued

Table 5-10. Signals in SBI Mode (1/2)

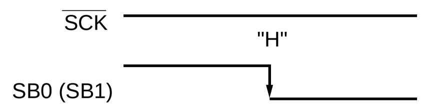
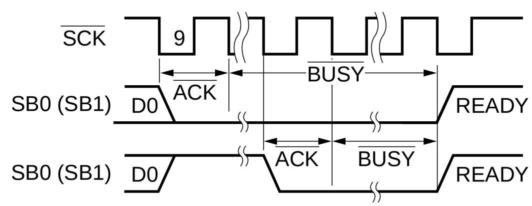
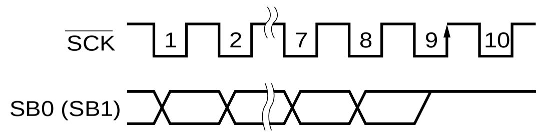
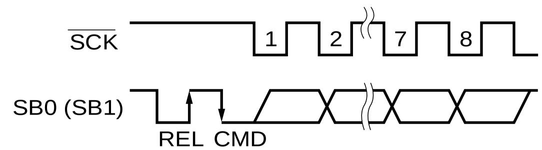
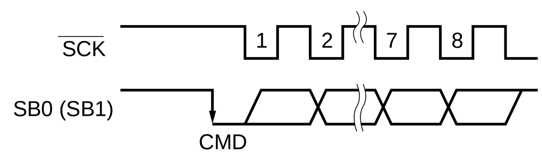
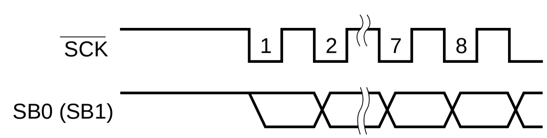
Signal Name	Outputting Device	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Bus release signal (REL)	Master	SB0 (SB1) rising edge when $\overline{\text{SCK}}$ = 1		• RELT set	• RELD set • CMDD cleared	Outputs next CMD signal and indicates send data is address.
Command signal (CMD)	Master	SB0 (SB1) falling edge when $\overline{\text{SCK}}$ = 1		• CMDT set	• CMDD set	(1) After REL signal output send data is address. (2) Send data with no REL signal output is command.
Acknowledge signal ($\overline{\text{ACK}}$)	Master/ slave	Low-level signal output to SB0 (SB1) in 1 $\overline{\text{SCK}}$ clock interval after serial receive completion.		① ACKE = 1 ② ACKT set	• ACKD set	Receive completion
Busy signal ($\overline{\text{BUSY}}$)	Slave	Low-level signal output to SB0 (SB1) after Acknowledge signal.		• BSYE = 1	–	Serial transmission/ reception disabled because processing is in progress.
Ready signal (READY)	Slave	High-level signal output to SB0 (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 transmission/ reception enabled

Table 5-10. Signals in SBI Mode (2/2)

Signal Name	Outputting Device	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Serial clock ($\overline{SCK}$)	Master	Synchronization clock for output of address/ command/data, $\overline{ACK}$ signal, synchronous $\overline{BUSY}$ signal, etc. Address/command/data is transferred in first 8 cycles.		Execution of instruction to write to SIO when CSIE = 1 (serial transfer start directive[Note 2])	IRQCSI set (rise of 9th $\overline{SCK}$ clock cycle)[Note 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
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.				Numeric value to be processed by slave or master device.

Notes 1. When WUP = 0, IRQCSI is always set on the 9th rise of $\overline{SCK}$.
When WUP = 1, IRQCSI is set on the 9th rise of $\overline{SCK}$ 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

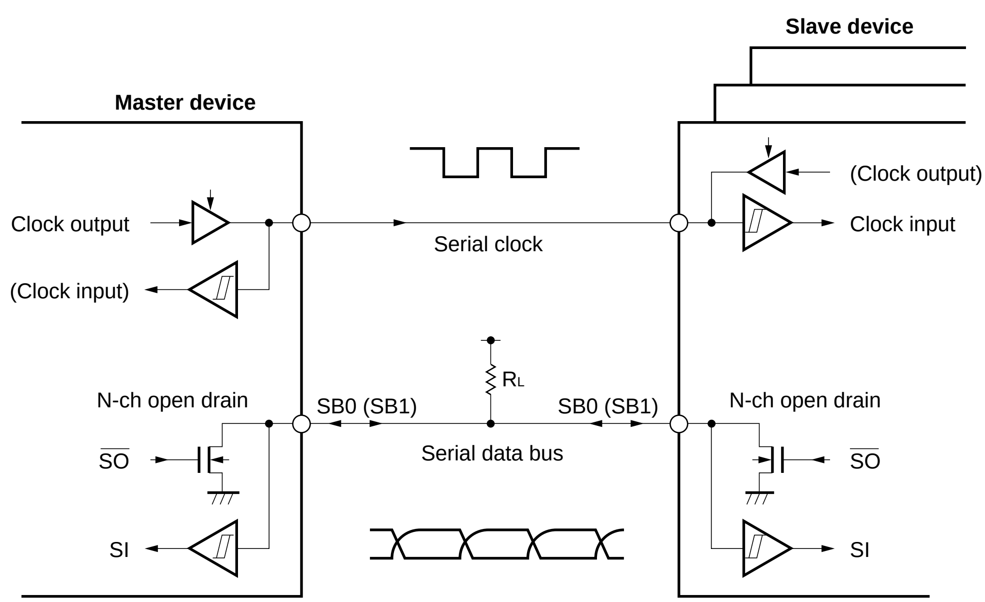
The configuration of the serial clock pin ($\overline{SCK}$) and the serial data bus pin SB0 (SB1) is as shown below.

(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-purpose 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 5-58. Pin Configuration Diagram



Caution Since the N-ch open drain 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).

Cautions 1. Detection of the slave selected/non-selected 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/non-selection by slave address should be performed with WUP = 1.

2. For selection/non-selection 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 5-59 through 5-62.

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 5-59. Address Transmission from Master Device to Slave Device (WUP = 1)

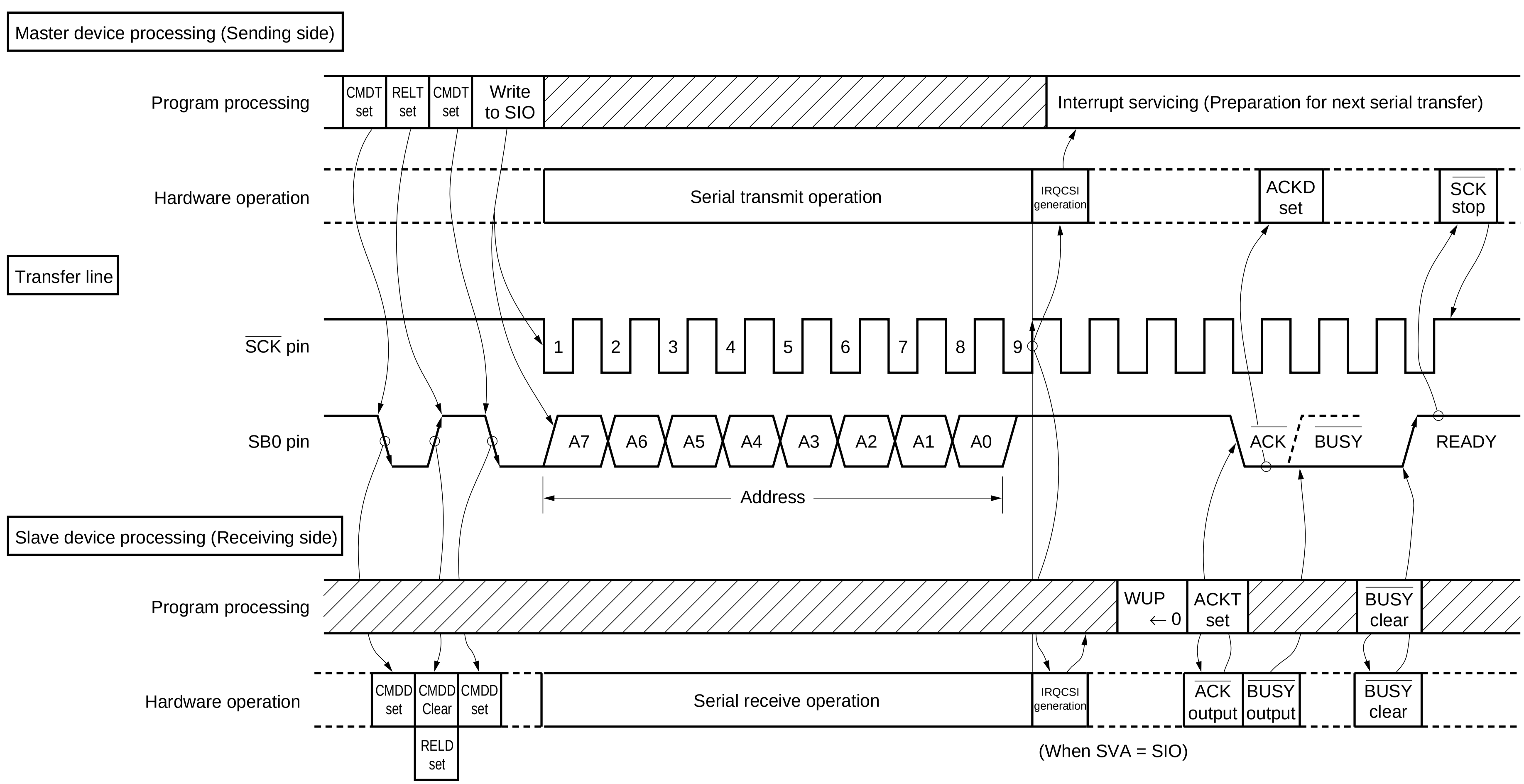
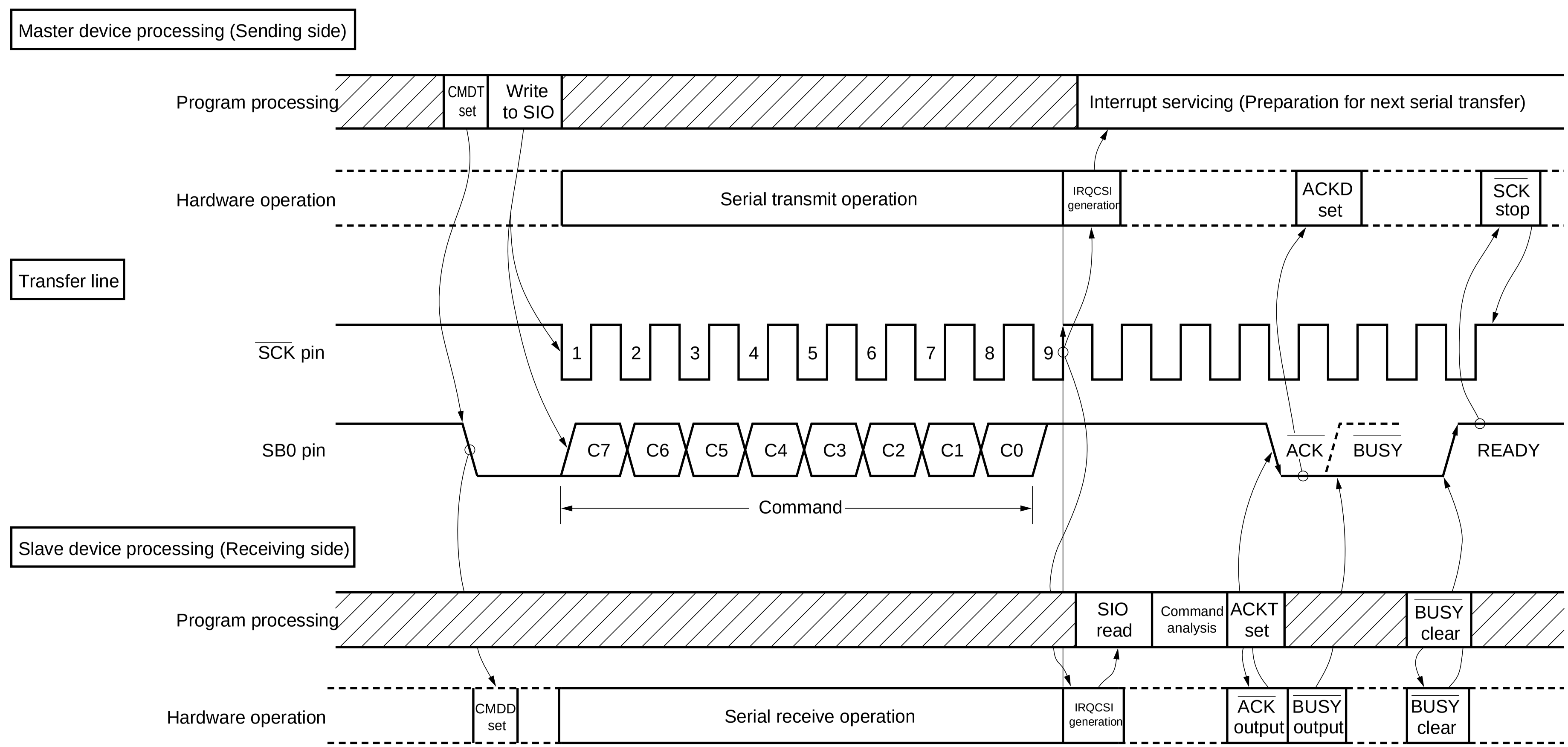


Figure 5-60. Command Transmission from Master Device to Slave Device



Phase-out/Discontinued

Figure 5-61. Data Transmission from Master Device to Slave Device

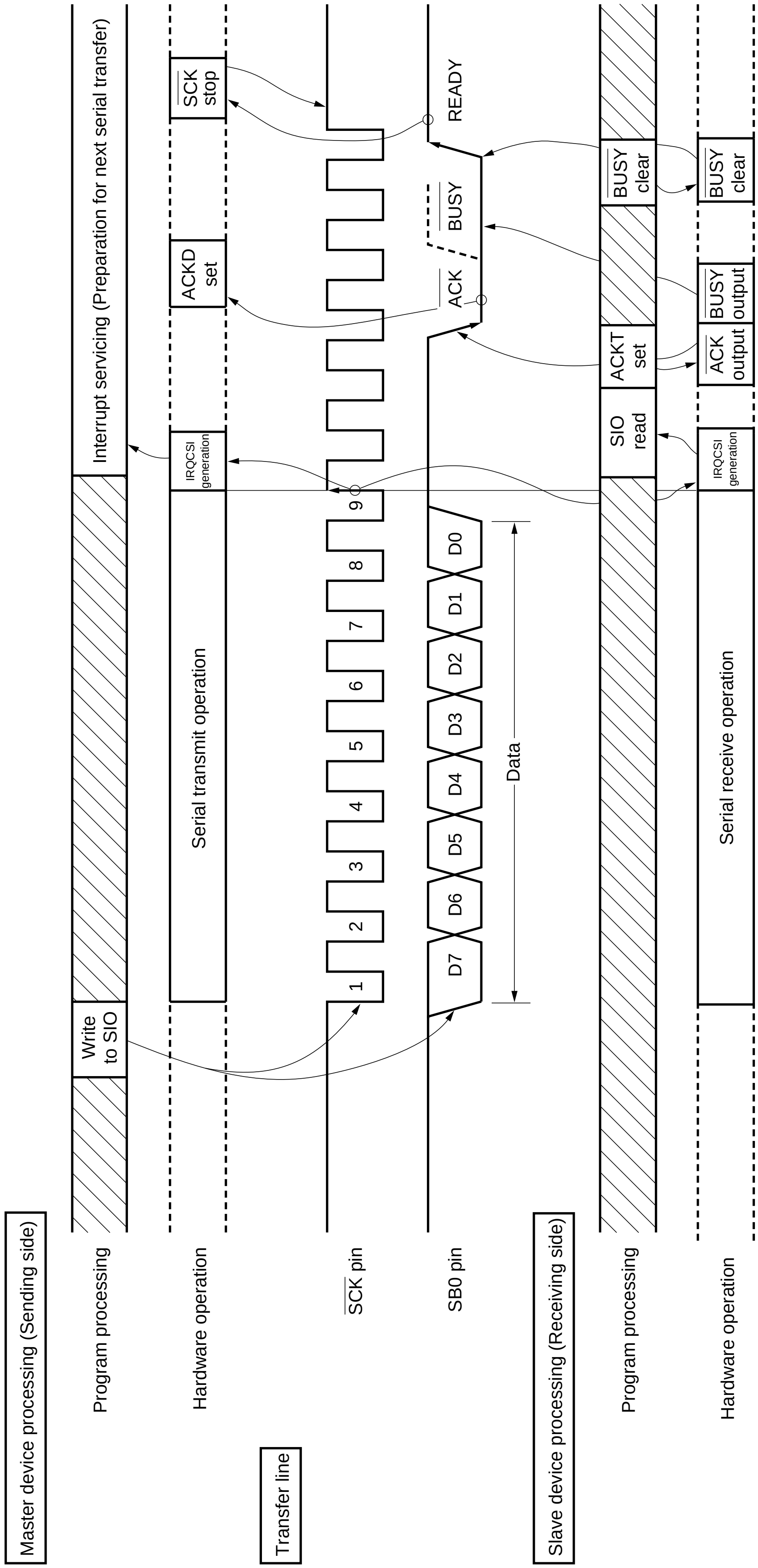
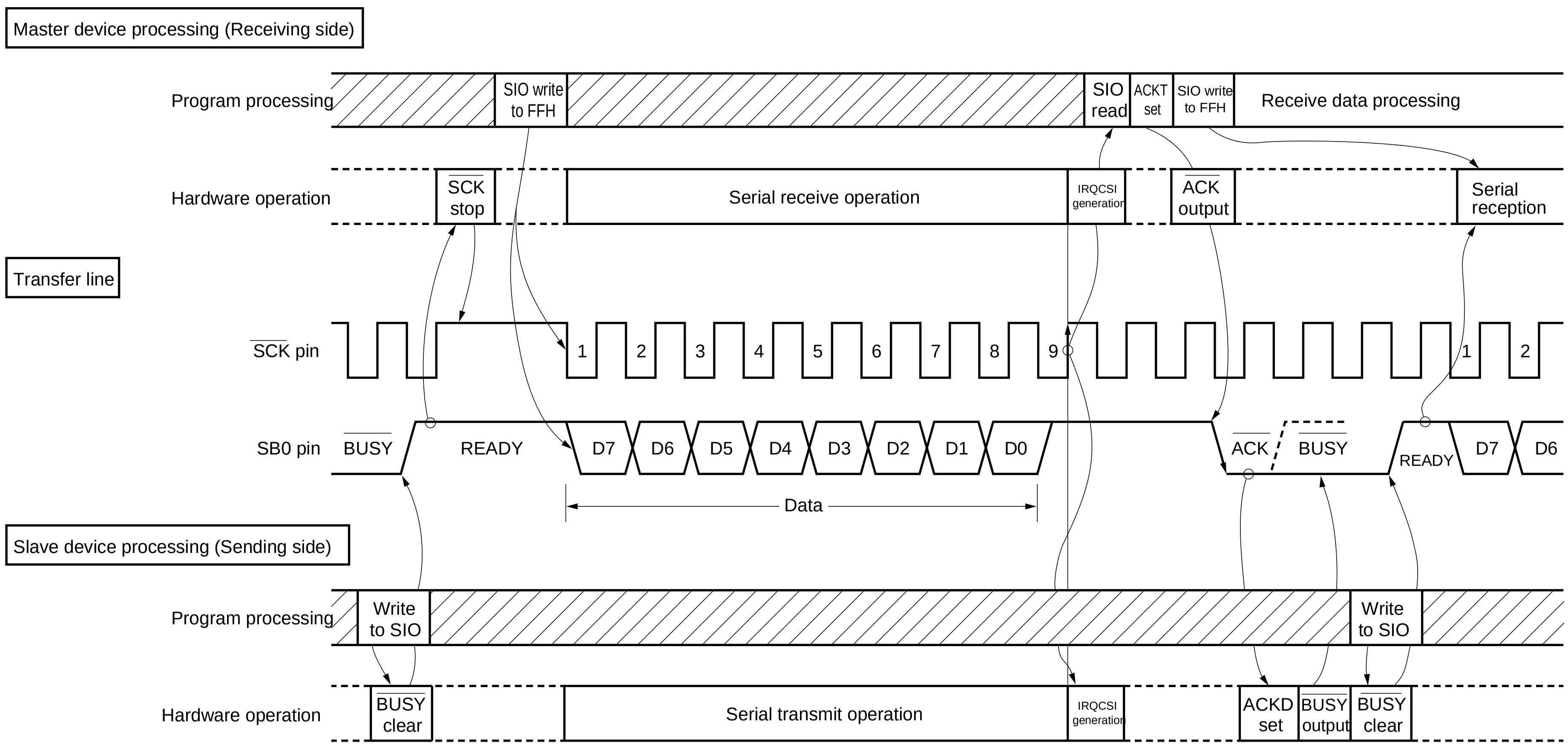


Figure 5-62. Data Transmission from Slave Device to Master Device



(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 $\overline{\text{SCK}}$ is high.

Cautions 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 selected/non-selected 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/non-selection by slave address should be performed with WUP = 1.

(b) For selection/non-selection 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 μPD75308 can operate as either the master CPU or a slave CPU on the serial bus.

Also, the master can be changed by a command.

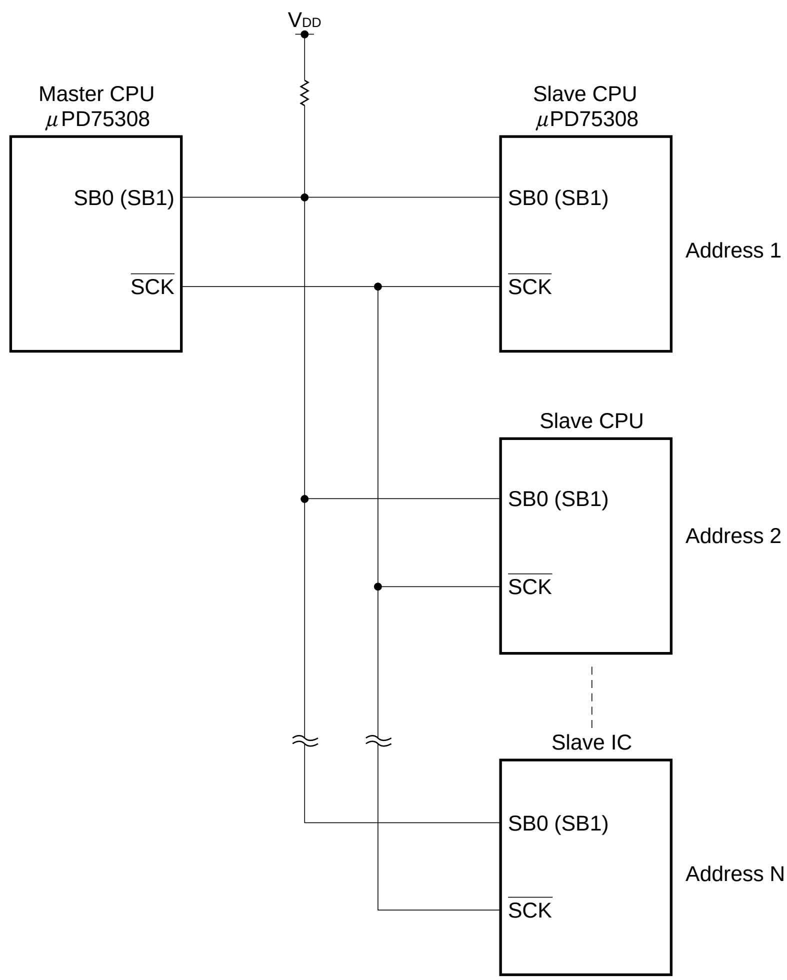
(a) Serial bus configuration

In the serial bus configuration in the application example given here the μPD75308 is assumed to be connected to the bus line as one of the devices on the serial bus.

Two μPD75308 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 5-63.

Figure 5-63. Example of Serial Bus Configuration



(b) Description of commands

(i) 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 : Read slave-side status.
6. RESET command : Sets currently selected slave to non-selected status.
7. CHGMST command : Transfers mastership to slave side.

(ii) 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 $\overline{\text{ACK}}$ and performs communication with the master (changes from non-selected to selected status).
2. Communication is performed between the slave selected in 1 and the master by transferring commands and data.
   Since command and data transfer is performed between master and slave on a one-to-one basis, the other slaves must be in the non-selected status.
3. Communication is terminated when the slave changes to the non-selected status. This happens in the following cases:
   - When a RESET command is sent from the master the selected slave changes to non-selected status.
   - When the master is changed by the CHGMST command, the device which changes from master to slave assumes the non-selected status.

(iii) 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 5-64. READ Command Transfer Format

M	S	M	S	S	S		S	S
READ	$\overline{\text{ACK}}$	Number of data bytes	$\overline{\text{ACK}}$	Data 0	$\overline{\text{ACK}}$	·····	Data N	$\overline{\text{ACK}}$
Command		Data		Data			Data	

Remark M : Output by master
S : Output by slave

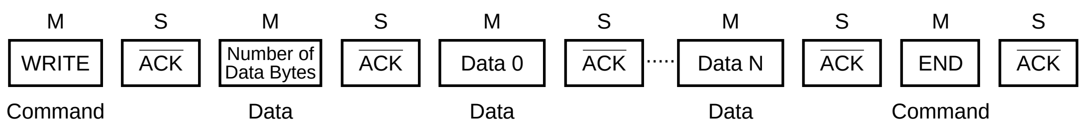
After receiving the number of data bytes, if the transmissible data is not less than that number of data bytes the slave returns $\overline{\text{ACK}}$. If there is insufficient data, $\overline{\text{ACK}}$ is not returned and an error is generated.

When the master receives data it sends $\overline{\text{ACK}}$ 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.

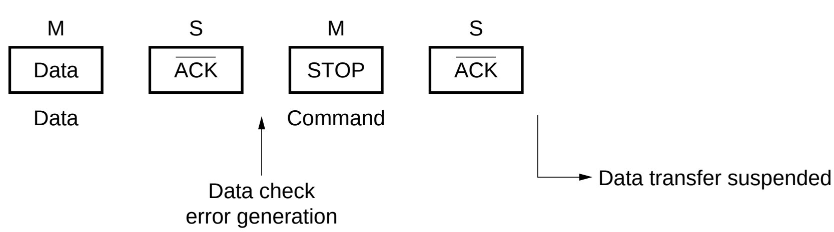
Figure 5-65. WRITE and END Command Transfer Format



Remark 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 $\overline{\text{ACK}}$. If the storage area is insufficient, $\overline{\text{ACK}}$ is not returned and an error is generated.
When the master has sent all the data it sends and 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 5-66. STOP Command Transfer Format



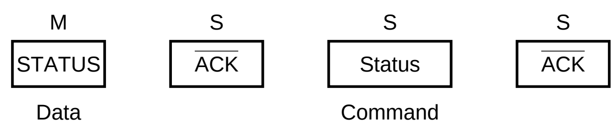
Remark M : Output by master
S : Output by slave

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.

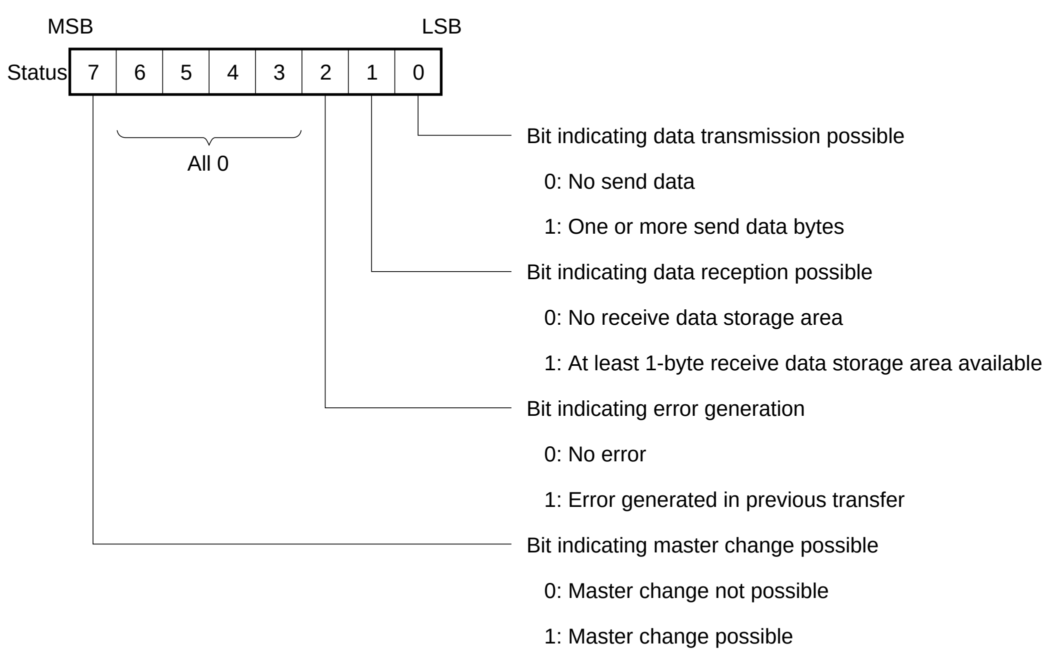
Figure 5-67. STATUS Command Transfer Format



Remark M : Output by master
S : Output by slave

The format of the status byte returned by the slave is shown below.

Figure 5-68. STATUS Command Status Format

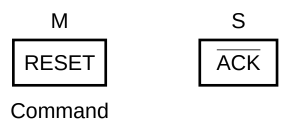


When the master receives status data it returns $\overline{\text{ACK}}$ to the slave.

④ **RESET command**

This command changes the currently selected slave to the non-selected status. Transmission of the RESET command allows all slaves to be set to the non-selected status.

Figure 5-69. RESET Command Transfer Format

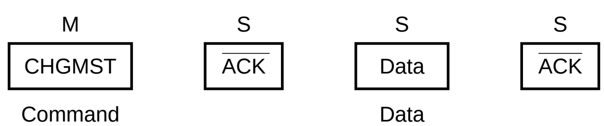


Remark M : Output by master
S : Output by slave

⑤ **CHGMST command**

This command passes mastership to the currently selected slave.

Figure 5-70. CHGMST Command Transfer Format



Remark 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 $\overline{\text{ACK}}$ and an error is generated.
When the master receives data it returns $\overline{\text{ACK}}$ to the slave. If the receive data is 0FFH, it henceforth operates as a slave. After the slave sends 0FFH data and $\overline{\text{ACK}}$ is returned from the master, it operates as the master.

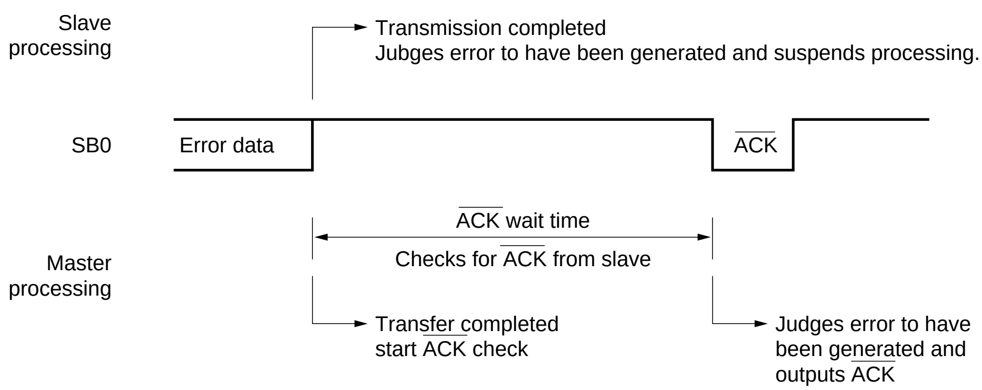
(iv) Error generation

Operation in the event of a communication error is described below.

A slave notifies the master of error generation by not returning $\overline{\text{ACK}}$. 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 $\overline{\text{ACK}}$ from the slave. If $\overline{\text{ACK}}$ 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 $\overline{\text{ACK}}$ signal (as a dummy).

Figure 5-71. Master and Slave Operation in Case of Error Generation



Error are as follows:

- **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. Data change during data transmission by a READ, STATUS or CHGMST command

When any of ① through ⑤ occurs, $\overline{\text{ACK}}$ is not returned to the master by the slave.

- **Errors generated on master side**

  When data changes during WRITE command data transmission, a STOP command is sent to the slave.

5.6.8 $\overline{\text{SCK}}$ pin output manipulation

As an output latch is incorporated in the $\overline{\text{SCK}}$/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 $\overline{\text{SCK}}$ number to be set to any desired value by software (SO/SB0 and SI/SB1 pin control is performed by the RELT and CMDT bits of SBIC).

The method of manipulating the $\overline{\text{SCK}}$/P01 pin is shown below.

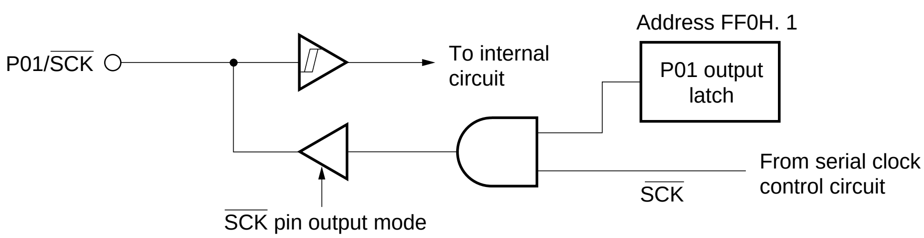
① The serial operating mode register (CSIM) is set ($\overline{\text{SCK}}$ pin: Output mode; serial operation: Possible). While serial transfer is suspended, $\overline{\text{SCK}}$ = 1.

② The P01 output latch is manipulated by a bit manipulation instruction.

Example To output one $\overline{\text{SCK}}$ 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 5-72. $\overline{\text{SCK}}$/P01 Pin Configuration



The P01 output latch is mapped onto bit 1 of address FF0H. $\overline{\text{RESET}}$ signal generation sets the P01 output latch to “1”.

Cautions 1. The P01 output latch must be set to “1” during a normal serial transfer.

2. The P01 output latch address cannot be specified as “PORT0.1” as shown below. The address (0FF0H.1) must be written directly as the operand. However, when the instruction is executed, MBE = 0 or (MBE = 1 and MSB = 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

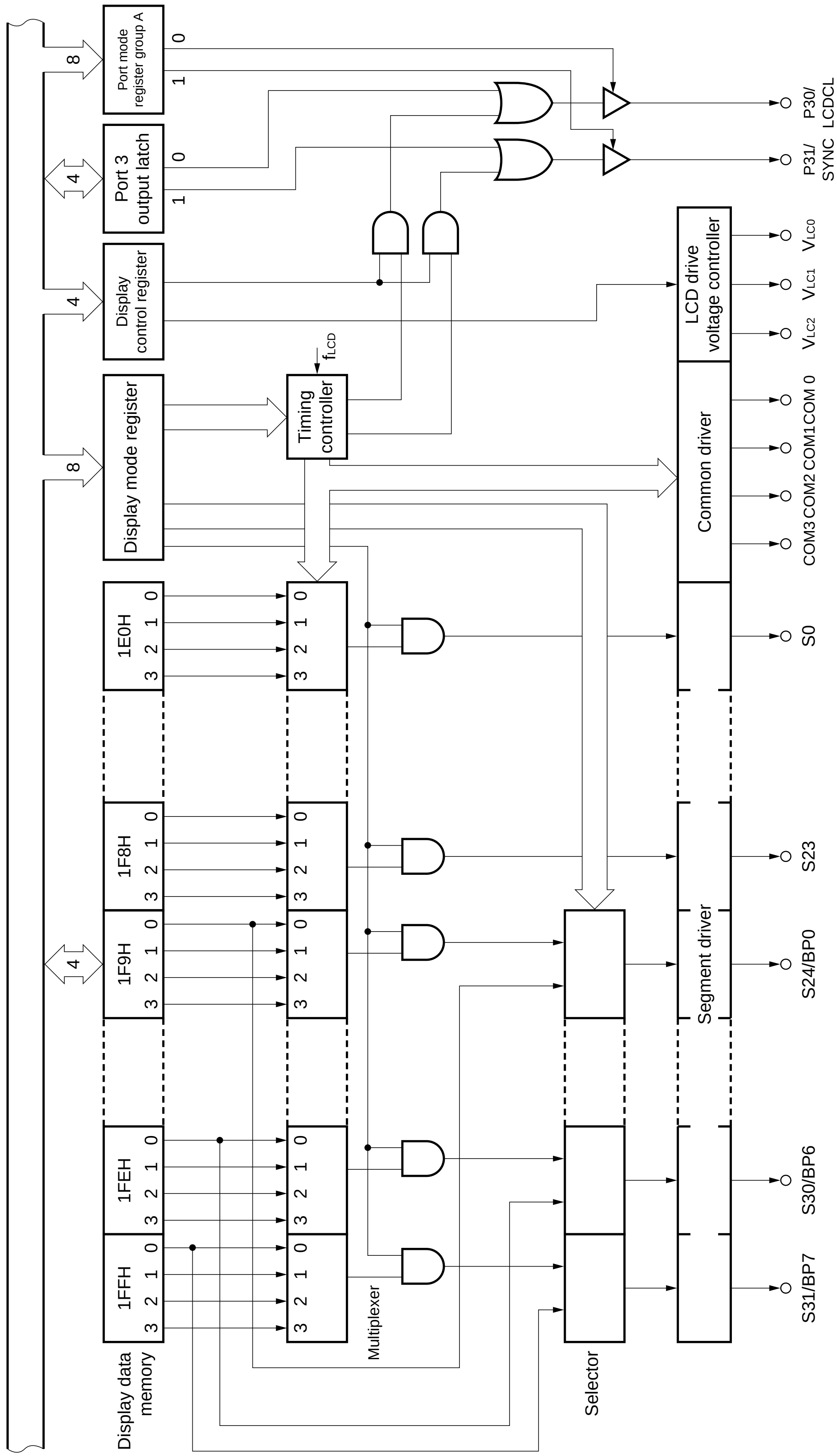
5.7.1 LCD controller/driver configuration

The μPD75308 incorporates a display controller which generates segment signals and common signals in accordance with data in display data memory, and a segment driver and common driver which can directly drive an LCD panel.

The configuration of the LCD controller/driver is shown in Figure 5-73.

Phase-out/Discontinued

Figure 5-73. LCD Controller/Driver Block Diagram



5.7.2 LCD controller/driver function

The functions of the LCD controller/driver incorporated in the μPD75308 are described below.

(a) Generates segment signals and common signals by automatically reading display data memory using DMA operations.

(b) One of 5 display modes can be selected:
 ① Static
 ② 1/2 duty (2-time division) 1/2 bias
 ③ 1/3 duty (3-time division) 1/2 bias
 ④ 1/3 duty (3-time division) 1/3 bias
 ⑤ 1/4 duty (4-time division) 1/3 bias

(c) In each display mode there is a choice of 4 frame frequencies.

(d) Maximum of 32 segment signal outputs (S0 to S31); 4 common outputs (COM0 to COM3)

(e) Segment signal outputs (S24 to S27, S28 to S31) can be switched to output ports (BP0 to BP3, BP4 to BP7) in 4-line units.

(f) On-chip division resistor available (mask option) for LCD drive power supply.
 • Various bias module, allowing correspondence to LCD drive voltage
 • Current to division resistor is cut when display is off.

(g) Display data memory not used for display can be used as normal data memory.

(h) Operation possible using subsystem clock

The maximum number of pixels which can be displayed in each display mode is shown in Table 5-11.

Table 5-11. Maximum Number of Display Pixels

Bias Modulus	Time Division	Common Signal Used	Maximum Number of Pixels
–	Static	COM0 (COM1, 2, 3)	32 (32 segment x 1 common)Note 1
1/2	2	COM0, 1	64 (32 segment x 2 common)Note 2
	3	COM0, 1, 2	96 (32 segment x 3 common)Note 3
1/3	3		
	4	COM0, 1, 2, 3	128 (32 segment x 4 common)Note 4

Notes 1. With 8. shaped LCD panel, 4 digits at 8 segment signals per digit.
2. With 8. shaped LCD panel, 8 digits at 4 segment signals per digit.
3. With 8. shaped LCD panel, 10 digits at 3 segment signals per digit.
4. With 8. shaped LCD panel, 16 digits at 2 segment signals per digit.

5.7.3 Display mode register

The display mode register (LCDM) is an 8-bit register which specifies the display mode, LCD clock, frame frequency, segment output/bit port output selection, and display output on/off control.

LCDM is set by an 8-bit memory manipulation instruction. Only bit 3 (LCDM3) can be set or cleared by a bit manipulation instruction.

Generation of the $\overline{\text{RESET}}$ signal clears all bits to "0".

Figure 5-74. Display Mode Register Format (1/2)

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	Number of Time Divisions	Bias Modulus
0	x	x	x	Display off[Note]	
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	
Other than above				Setting prohibited	

Note All segment signals at non-selected 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)

Caution LCDCL is supplied only when the watch timer is operating. When the LCD controller is used, bit 2 of the watch mode register WM should be set (1)

Figure 5-74. Display Mode Register Format (2/2)

Segment output/bit port output switchover specification

LCDM7	LCDM6	S24 to S27	S28 to S31	Number of Segment Outputs	Number of Bit Port Outputs
0	0	Segment output	Segment output	32	0
0	1	Segment output	Bit port output	28	4
1	0	Bit port output	Segment output	28	4
1	1	Bit port output	Bit port output	24	8

Frame frequency (Hz)

LCDCL / Display Duty	$f_W/2^9$ (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

Remark When f_W = 32.768 kHz

f_W: Input clock to watch timer (f_X/128 or f_{XT})

5.7.4 Display control register

The display control register (LCDC) performs the following LCD drive control functions.

- Enabling/disabling common and segment outputs
- Cutting current to division resistor for LCD drive power supply
- Enabling/disabling output of synchronization clock (LCDCL) and synchronization signal (SYNC) to controller/driver for external segment signal extension

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

Generation of the $\overline{\text{RESET}}$ signal clears (0) the entire display control register

Figure 5-75. Display Control Register Format

Address	3	2	1	0	Symbol
F8EH	0	LCDC2	0	LCDC0	LCDC

Display output status of LCDC0 and LCDM3

LCDC0	0	1	
LCDM3	x	0	1
COM0 to COM3	Outputs “L” (Display off)	Outputs common signal corresponding to display mode.	Outputs common signal corresponding to display mode.
S0 to S23 / S24 to S31 segment specification pins	Outputs “L”. (Display off)	Outputs segment signal corresponding to display mode. (Non-selected level output, display off)	Outputs segment signal corresponding to display mode. (Display off)
S24 to S32 bit port specification pins	Outputs contents of bit 0 of corresponding display data memory. (Bit port function)	Outputs contents of bit 0 of corresponding display data memory. (Bit port function)	Outputs contents of bit 0 of corresponding display data memory. (Bit port function)
Power supply to division resistor (BIAS pin output)	Off (high impedance)[Note]	On (high level)[Note]	On (high level)[Note]

Note Items in parentheses apply when split resistor for LCD drive power supply is not incorporated.

LCDCL and SYNC signal output enable/disable bit

LCDC2		
	0	LCDCL and SYNC signal output disabled
	1	LCDCL and SYNC signal output enabled

Caution LCDCL and SYNC signal outputs are provided for future system expansion. At present these signal outputs should be disabled.

5.7.5 Display data memory

Display data memory is mapped onto addresses 1E0H through 1FFH.

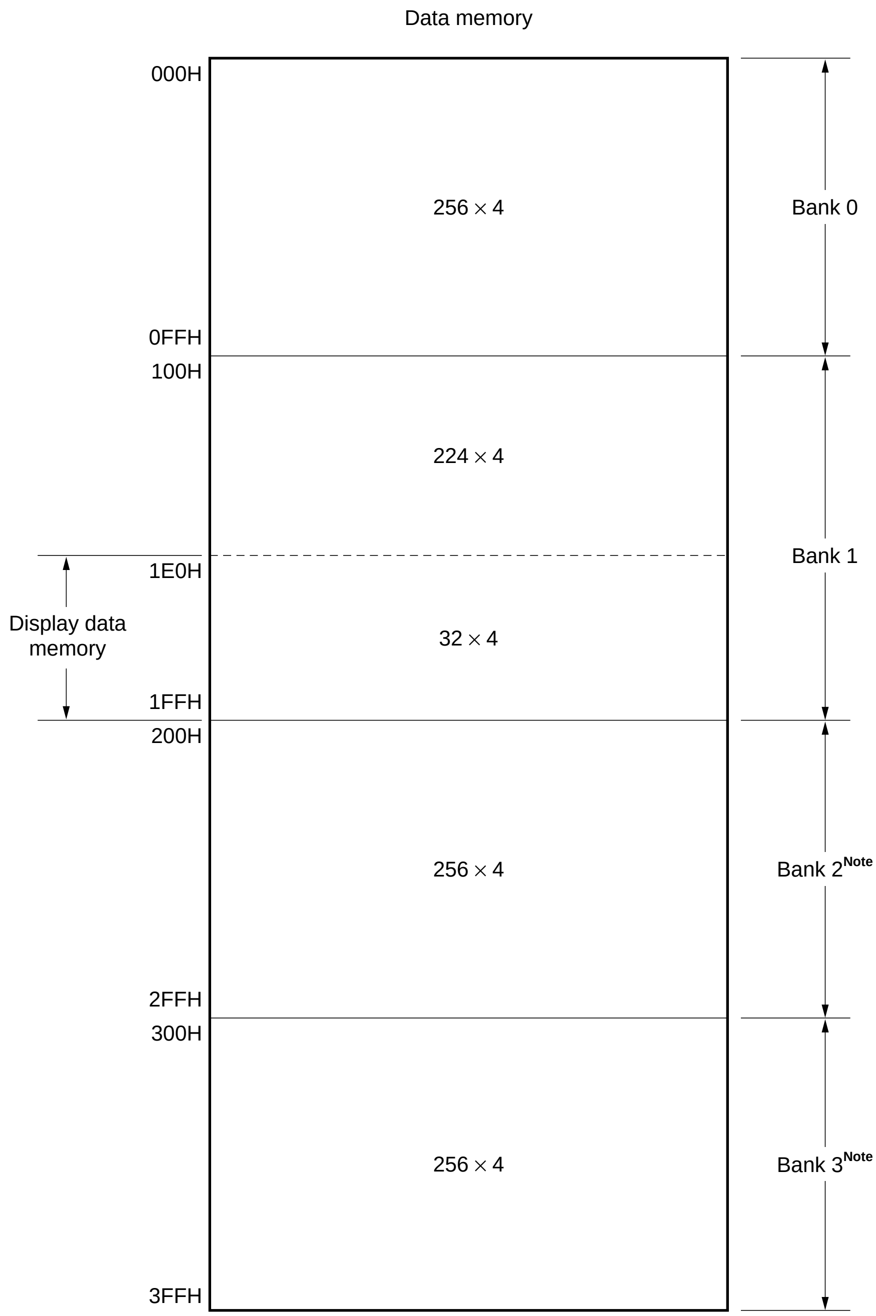
Display data memory is a memory area which the LCD controller/driver reads by DMA operations irrespective of CPU operation. The LCD controller controls segment signals according to the data in display data memory. When S24 through S31 are used as bit ports, the content of bit 0 of the data written into addresses 1F8H to 1FFH of display data memory is output from each bit port output pin.

The area which is not used as LCD display ports can be used as ordinary data memory.

Display data memory can be manipulated bit by bit or in 4-bit units. It cannot be manipulated in 8-bit units.

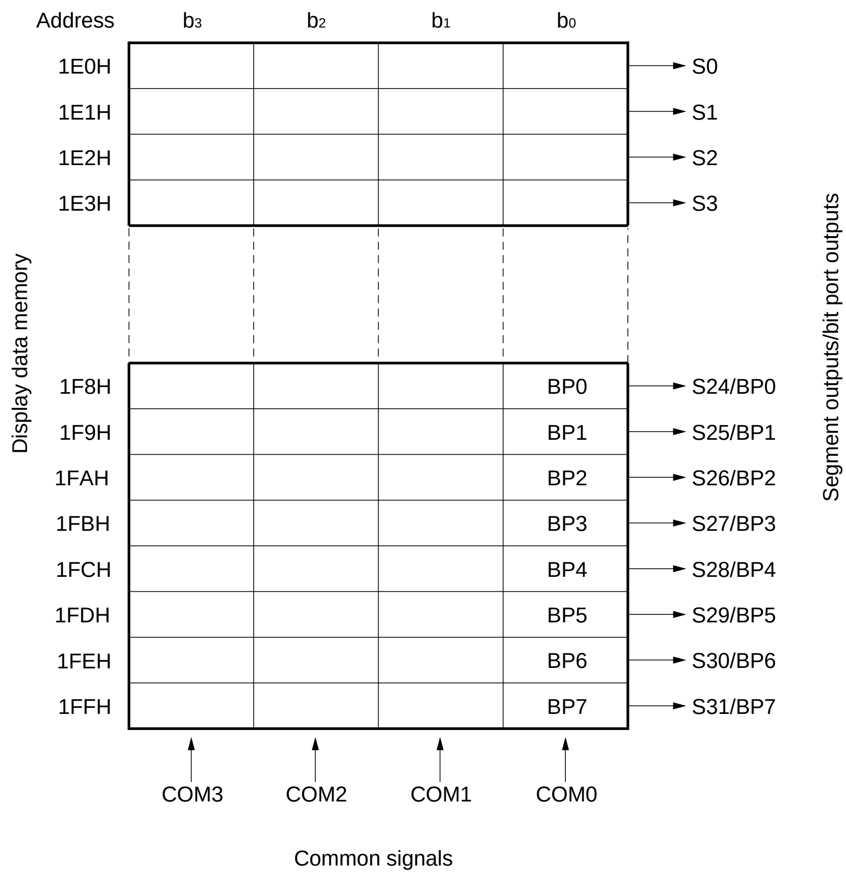
The correspondence between each bit of display data memory and segment outputs and bit port outputs is shown in Figure 5-77.

Figure 5-76. Data Memory Map



Note Banks 2 and 3 are provided on the μPD75312B, 75316B, 75P316A, 75P316B only.

Figure 5-77. Correspondence between Display Data Memory and Common Segments



5.7.6 Common signals and segment signals

A picture element of the LCD panel is illuminated when the potential difference between the corresponding common signal and segment signal reaches or exceeds a preset voltage (LCD drive voltage V_{LCD}). When the potential difference falls to less than V_{LCD} or 0 V, the picture element is extinguished.

Since deterioration results from adding a DC voltage to the common signal and segment signal, the LCD panel is driven by an AC voltage.

(1) Common signals

Common signals have the selection timing order shown in Table 5-12 corresponding to the assigned number of time divisions, with repeated operation of these performed as one cycle. In the static mode, the same signal is output for COM0, 1, 2, and 3. In the case of two time divisions, the COM2 and COM3 pins should be left open; with 3 time divisions, the COM3 pin should be left unconnected.

Table 5-12. Common Signals

Number of Time Divisions \ Common Signals	COM0	COM1	COM2	COM3
Static				
2			Leave unconnected	Leave unconnected
3				Leave unconnected
4				

(2) Segment signals

Segment signals are carried on 32 lines corresponding to 32 locations of the display data area (1E0H to 1FFH) in data memory. Bits 0 through 3 of each location are automatically read in synchronization with COM0 through COM3 selection timing respectively, then if the value of the bit is “1” conversion to the selection voltage is performed, and if “0” conversion to the non-selection voltage is performed, for output from the segment pins (S0 through S31).

From the above, confirmation is made of the combination of LCD panel front electrode (corresponding to the segment signal) and rear electrode (corresponding to the common signal) used to form a display pattern, then bit data corresponding one-to-one with the pattern to be displayed is written into the display data area.

As the following bits of the display area are not accessed, they can be used for purpose other than display: Bits 1/2/3 in the static system, bits 2/3 in the 2-time-division system, and bit 3 in the 3-time-division system.

(3) Common signal and segment signal output waveforms

The voltage levels shown in Tables 5-13 through 5-15 are output in the common signals and segment signals. The $+V_{LCD}/-V_{LCD}$ illumination voltage results only when both reach the selection voltage; other combinations result in the “extinguished” voltage.

Table 5-13. LCD Drive Voltages (Static)

Segment Signal Sn / Common Signal COM0	Selection	Non-Selection
	V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
V_{SS}/V_{LC0}	$+V_{LCD}/-V_{LCD}$	0 V/0 V

Table 5-14. LCD Drive Voltages (1/2 Bias Modulus)

Segment Signal Sn / Common Signal COMm		Selection	Non-Selection
		V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
Selection	V_{SS}/V_{LC0}	$\boxed{+V_{LCD}/-V_{LCD}}$	0 V/0 V
Non-selection	$V_{LC1} = V_{LC2}$	$+1/2V_{LCD}/-1/2V_{LCD}$	$-1/2V_{LCD}/+1/2V_{LCD}$

Table 5-15. LCD Drive Voltage (1/3 Bias Modulus)

Segment Signal Sn / Common Signal COMm		Selection	Non-Selection
		V_{LC0}/V_{SS}	V_{LC2}/V_{LC1}
Selection	V_{SS}/V_{LC0}	$\boxed{+V_{LCD}/-V_{LCD}}$	$+1/3V_{LCD}/-1/3V_{LCD}$
Non-selection	V_{LC1}/V_{LC2}	$+1/3V_{LCD}/-1/3V_{LCD}$	$-1/3V_{LCD}/+1/3V_{LCD}$

Figures 5-78 through 5-80 show the common signal waveforms, and Figure 5-81 shows the voltage and phase of the common signal and the segment signal.

Figure 5-78. Common Signal Waveform (Static)

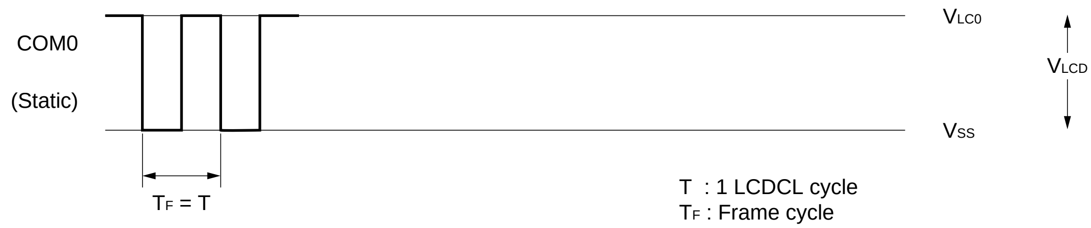


Figure 5-79. Common Signal Waveform (1/2 Bias Modulus)

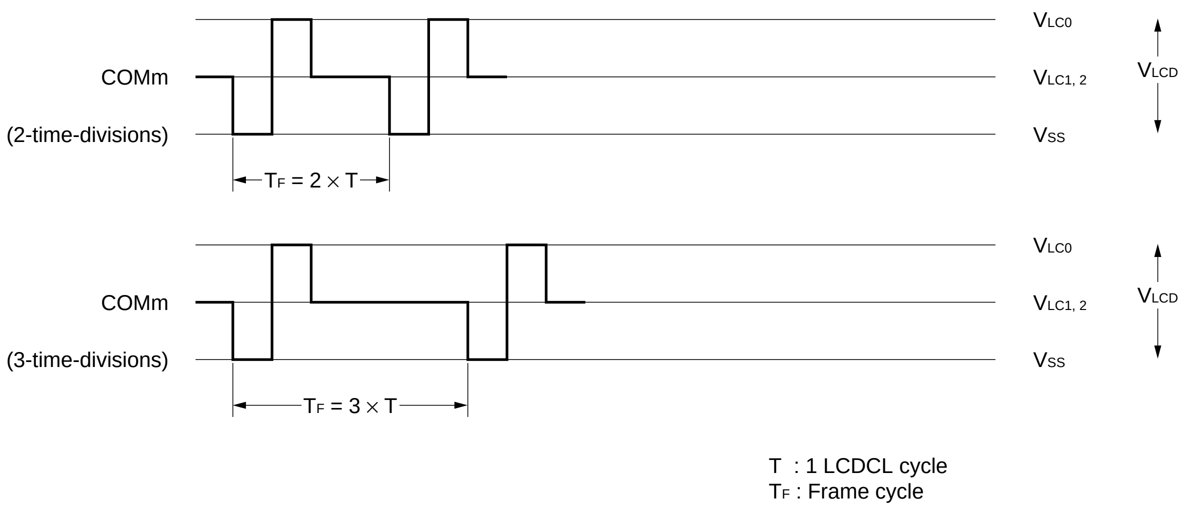


Figure 5-80. Common Signal Waveform (1/3 Bias Modulus)

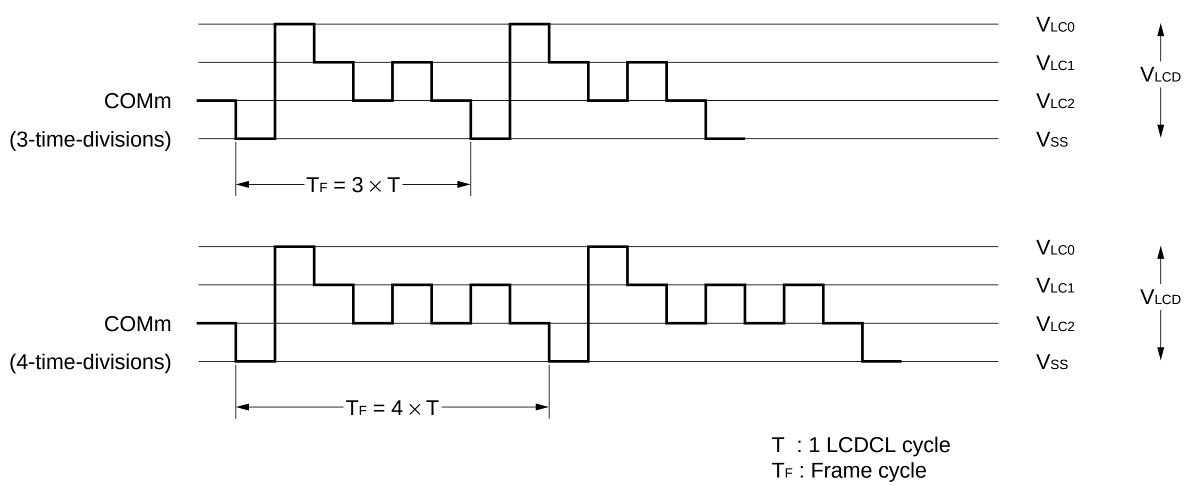
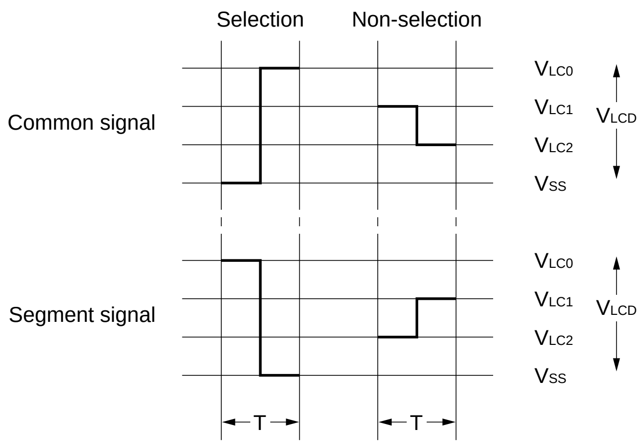


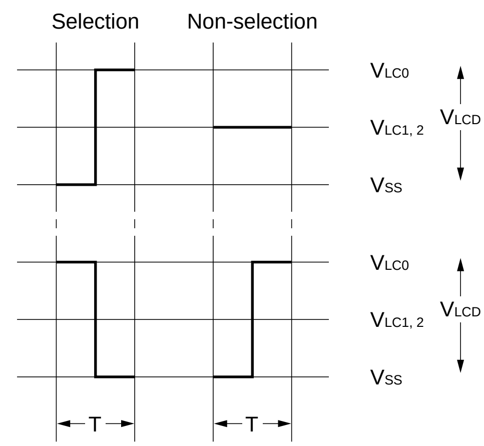
Figure 5-81. Voltage and Phase of Common Signal and Segment Signal

(a) 1/3 bias modulus

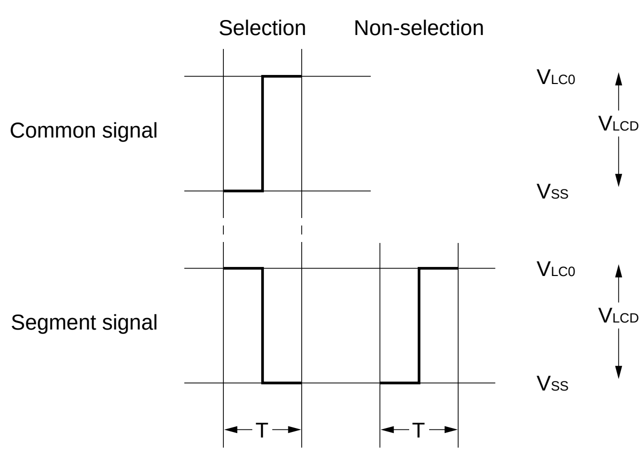


T: 1 LCDCL cycle

(b) 1/2 bias modulus



(c) Static display mode



5.7.7 LCD drive power V_{LC0}, V_{LC1}, V_{LC2} supply

In the μPD75308, a division resistor can be incorporated within pins V_{LC0} through V_{LC2} for the LCD drive power supply, allowing LCD drive power corresponding to each bias modulus to be supplied without an external division resistor. There is also a BIAS pin corresponding to the various LCD drive voltages; the BIAS pin and the V_{LC0} pin are connected externally.

The values shown in the following table are supplied as appropriate LCD drive power for the static, 1/2 and 1/3 bias modulus modes.

Table 5-16. LCD Drive Power Supply values

Bias Modulus / LCD Drive Power	No Bias (Static Display Mode)	1/2	1/3
V_{LC0}	V_{LCD}	V_{LCD}	V_{LCD}
V_{LC1}	$2/3V_{LCD}$	$1/2V_{LCD}$ Note1	$2/3V_{LCD}$
V_{LC2}	$1/3V_{LCD}$		$1/3V_{LCD}$
V_{SS}	0 V	0 V	0 V

Notes 1. In 1/2 bias mode, the V_{LC1} and V_{LC2} pins must be connected externally.

2. When BIAS pin and V_{LC0} pin are open, $V_{LCD} = 3/5V_{DD}$ (it is necessary to incorporate division resistor by mask option). When BIAS pin and V_{LC0} pin are connected, $V_{LCD} = V_{DD}$.

Figures 5-82 and 5-83 show LCD drive power supply examples based on Table 5-16.

Also, the current flowing in the division resistor can be cut by clearing (0) bit 0 (LCDC0) of the display control register.

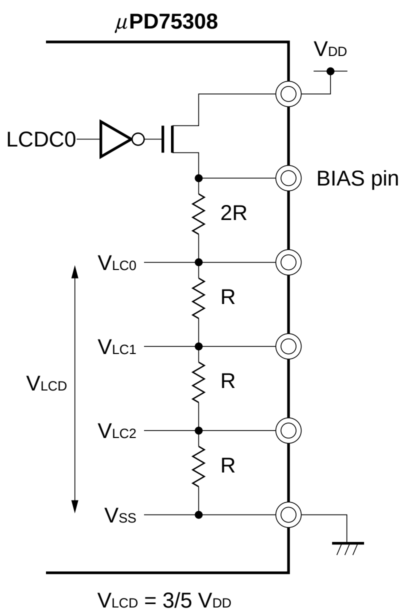
On/off control of this LCD power supply is also effective in preventing the LCD clock from being stopped by a STOP command (when the system clock has been selected) and a DC voltage from being applied to the LCD when the watch timer is operating by means of the main system clock.

In other words, by clearing (0) bit 0 (LCDC0) of the display control register and setting all LCD drive power supplies to the same potential, V_{SS}, directly before execution of the STOP instruction, it is possible to prevent the occurrence of a potential difference between the LCD electrodes even if the LCD clock stops.

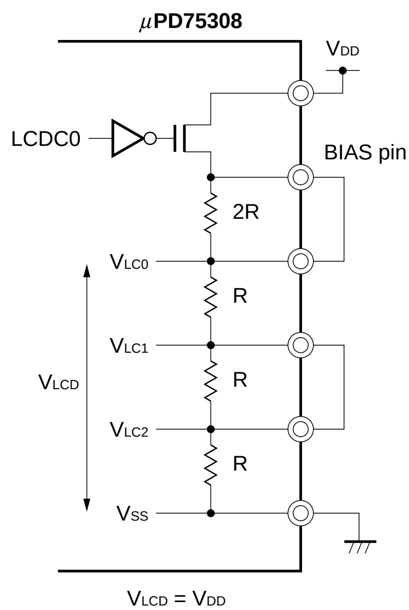
When the watch timer is operated by means of a subsystem clock, continuous LCD display is possible.

Figure 5-82. Examples of LCD Drive Power Supply Connection (With On-Chip Division Resistor)

(a) 1/3 bias modulus and static display mode (Example of V_{DD} = 5 V, V_{LCD} = 3 V)



(b) 1/2 bias modulus (Example of V_{DD} = 5 V, V_{LCD} = 5 V)



(c) 1/3 bias modulus and static display mode (Example of V_{DD} = 5 V, V_{LCD} = 5 V)

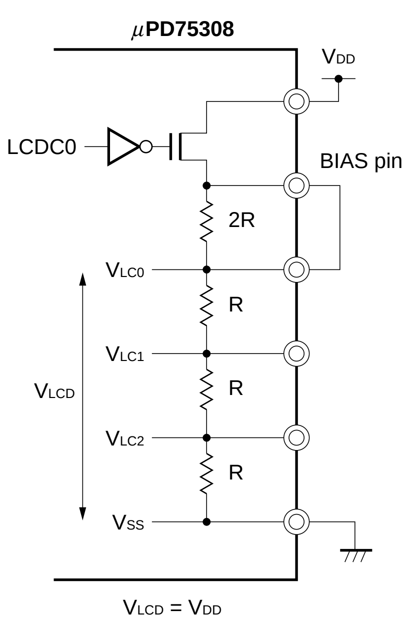
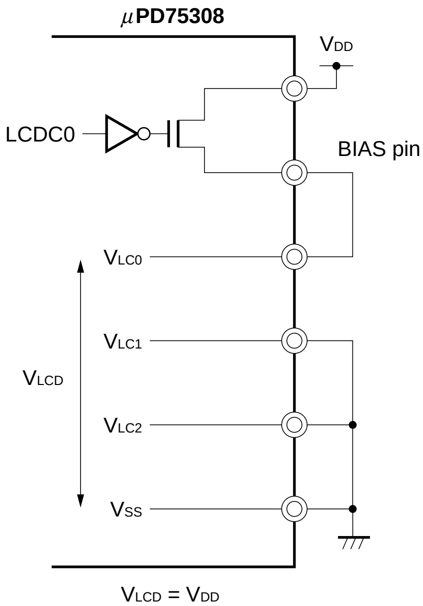
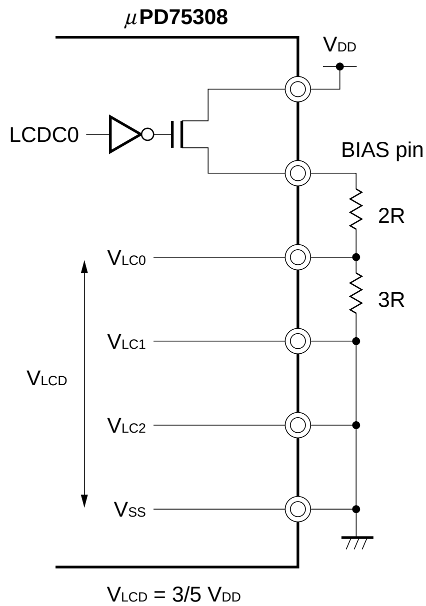


Figure 5-83. Examples of LCD Drive Power Supply Connection (With External Division Resistor)

(a) Static display mode[Note]
(Example of V_{DD} = 5 V, V_{LCD} = 5 V)

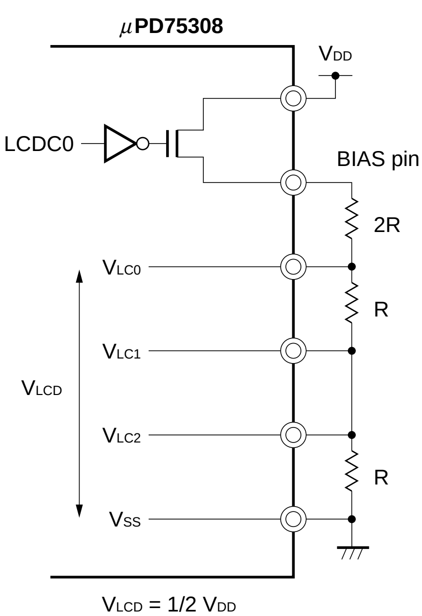


(b) Static display mode
(Example of V_{DD} = 5 V, V_{LCD} = 3 V)

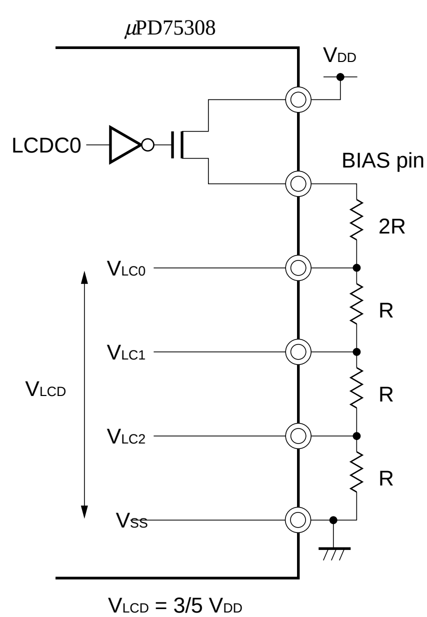


Note LCDC0 should always be set to 1 (including when in standby mode). ★
If LCDC0 is set to OFF while LCD display is performed, a DC voltage is applied to the LCD panel.

(c) 1/2 bias modulus
(V_{DD} = 5 V, V_{LCD} = 2.5 V)



(d) 1/3 bias modulus
(V_{DD} = 5 V, V_{LCD} = 3 V)



Caution To turn off the LCD panel, clear LCDM3 to issue a non-select signal. ★

5.7.8 Display modes

(1) Static display example

Figure 5-85 shows the connection between a static type 4-digit LCD panel with the display pattern shown in Figure 5-84, and the μPD75308's segment signals (S0 to S31) and common signal (COM0). The display example is "123.4", and the contents of the display data memory (addresses 1E0H through 1FFH) correspond to this value.

Here, the second digit "3." (3.) is considered for the sake of explanation. In accordance with the display pattern in Figure 5-84, using the COM0 common signal timing, the selection and non-selection voltages shown in Table 5-17 must be output to pins S8 through S15.

Table 5-17. Selection/Non-Selection Voltages for Pins S8 to S15 (Static Display Example)

Segment / Common	S8	S9	S10	S11	S12	S13	S14	S15
COM0	Selection	Selection	Selection	Selection	Non-selection	Selection	Non-selection	Selection

From this it can be seen that it is only necessary to set up "11110101" in bit 0 of the display data memory addresses (1E8H through 1EFH) corresponding to S8 through S15.

The LCD drive waveforms for S11, S12 and COM0 are shown in Figure 5-86. It can be seen that when S11 reaches the selection voltage in synchronization with COM0 selection timing, a $+V_{LCD}/-V_{LCD}$ (LCD illumination level) AC square wave is generated.

As the same waveform is output to COM1 through COM3 as to COM0, the drive performance can be increased by connecting COM0 through COM3.

Figure 5-84. Static Type LCD Display Pattern and Electrode Wiring

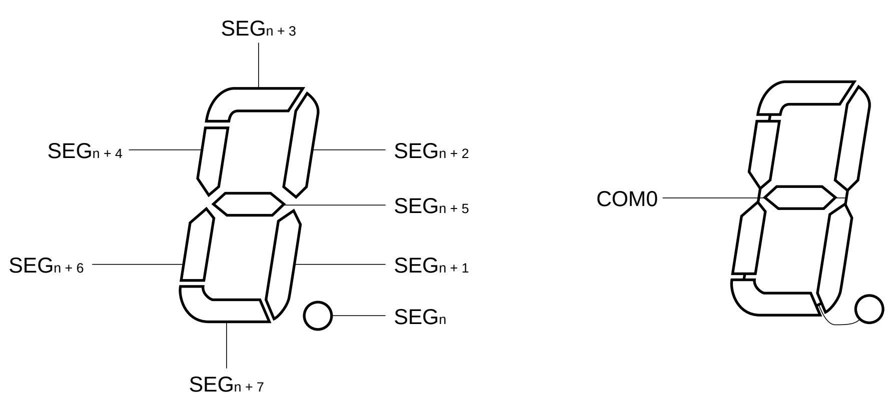


Figure 5-85. Static LCD Panel Wiring Example

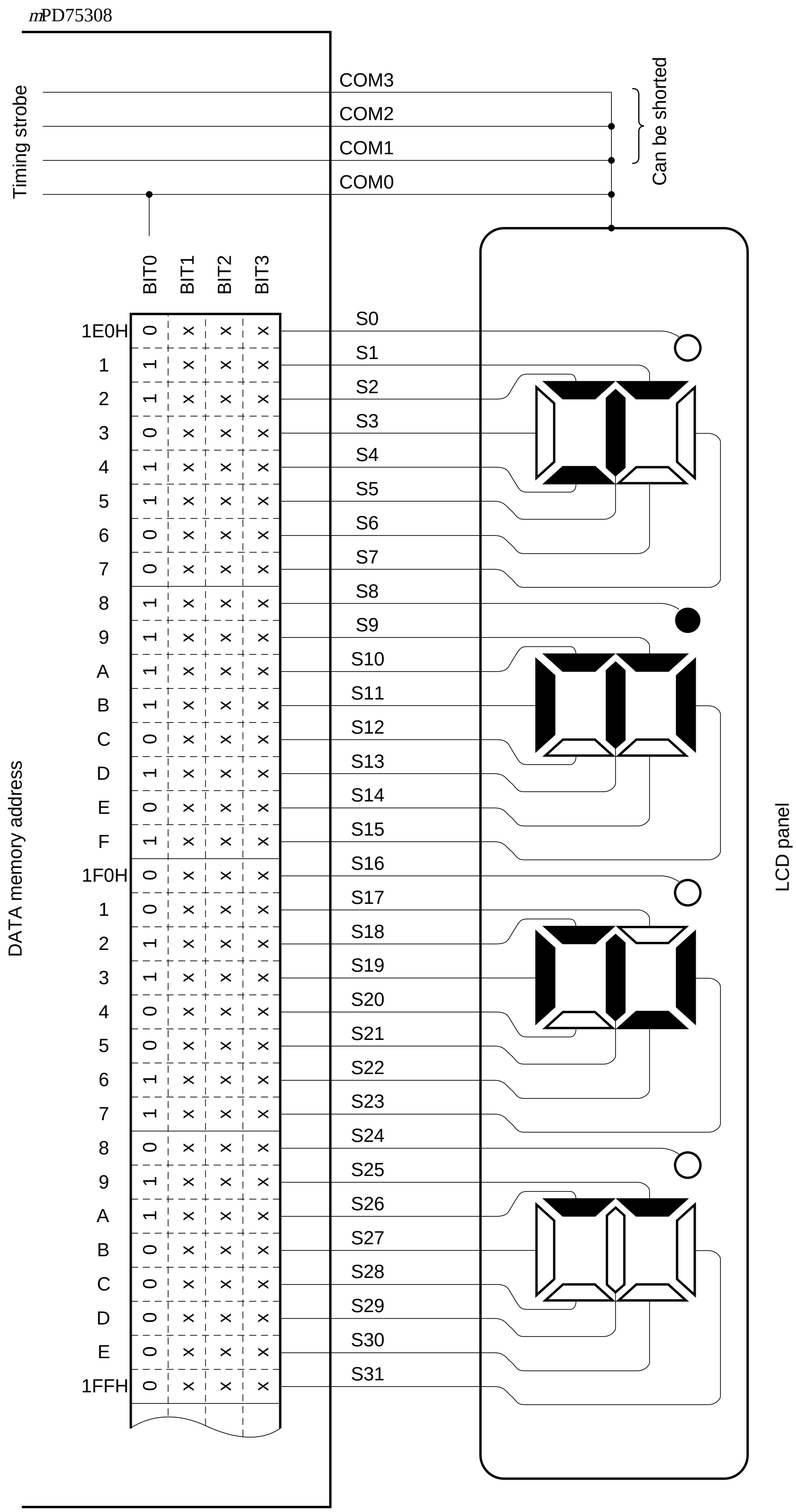
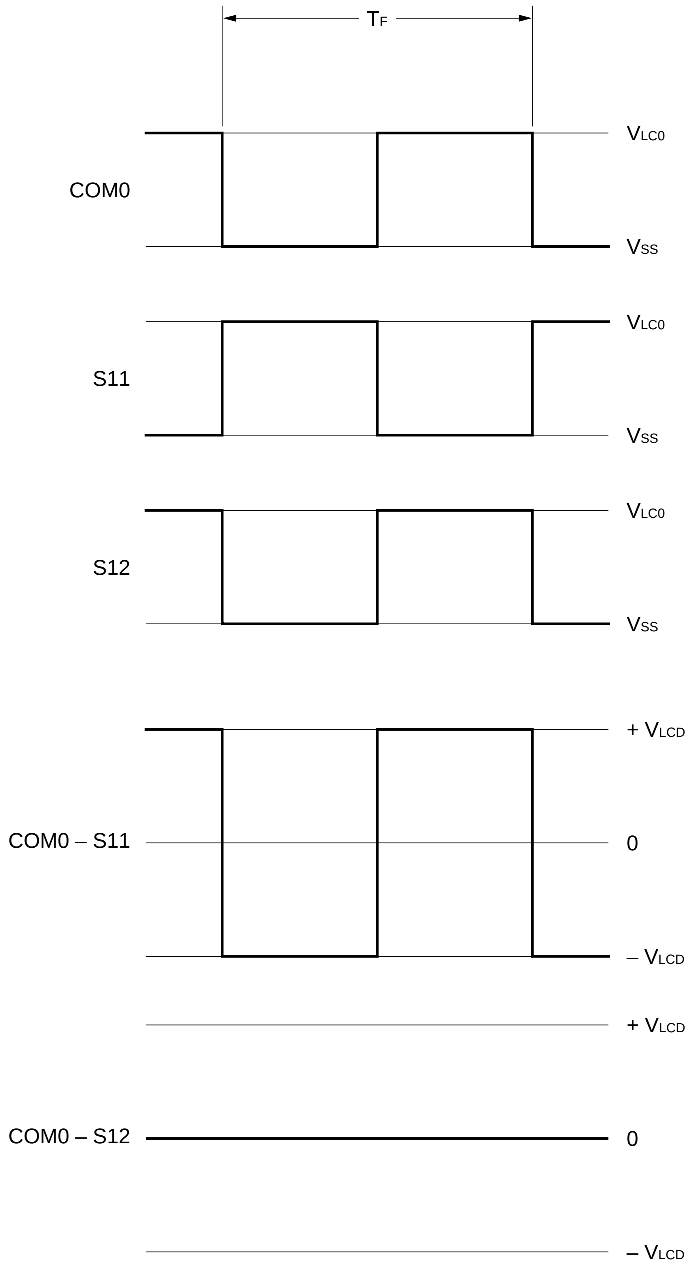


Figure 5-86. Example of Static LCD Drive Waveforms



(2) 2-time-division display example

Figure 5-88 shows the connection between a 2-time-division type 8-digit LCD panel with the display pattern shown in Figure 5-87, and the μPD75308's segment signals (S0 to S31) and common signals (COM0 and COM1). The display example is "123456.78", and the contents of the display data memory (addresses 1E0H through 1FFH) correspond to this value.

Here, the third digit "6." (6.) is considered for the sake of explanation. In accordance with the display pattern in Figure 5-87, using the COM0 and COM1 common signal timing, the selection and non-selection voltages shown in Table 5-18 must be output to pins S8 through S11.

Table 5-18. Selection/Non-Selection Voltages for Pins S8 to S11 (2-Time-Division Display Example)

Segment / Common	S8	S9	S10	S11
COM0	Selection	Non-selection	Selection	Selection
COM1	Selection	Selection	Selection	Selection

From this it can be seen that it is only necessary to set up, for example "xx10" in the display data memory (address 1E9H) corresponding to S9.

An example of the LCD drive waveforms between S9 and each common signal is shown in Figure 5-89. It can be seen that when S9 reaches the selection voltage in synchronization with COM1 selection timing, a $+V_{LCD}/-V_{LCD}$ (LCD illumination level) AC square wave is generated.

Figure 5-87. 2-Time-Division Type LCD Display Pattern and Electrode Wiring

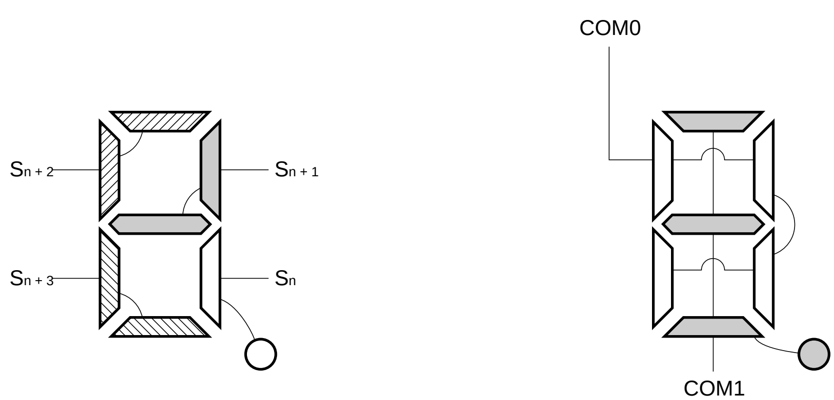
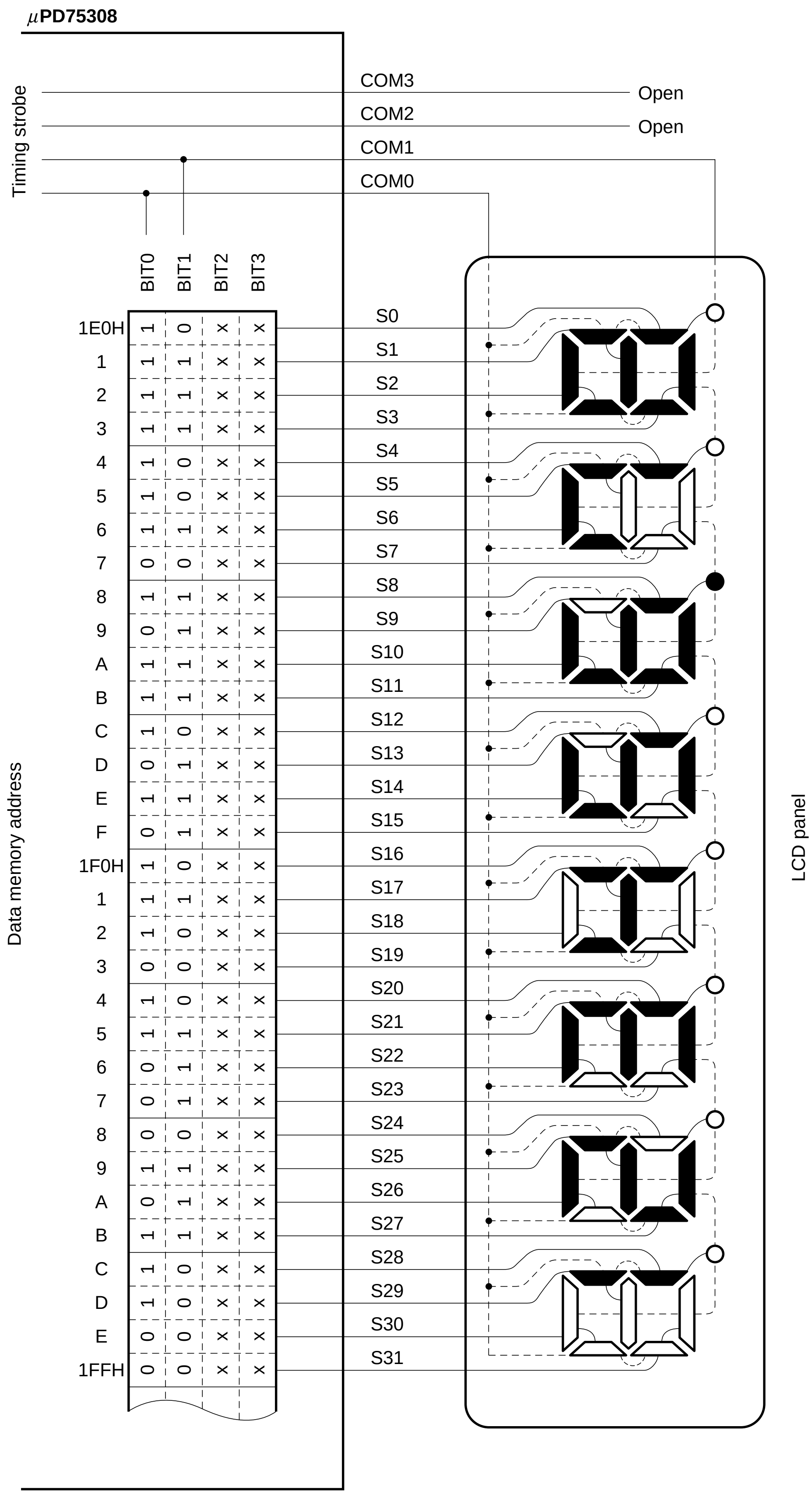
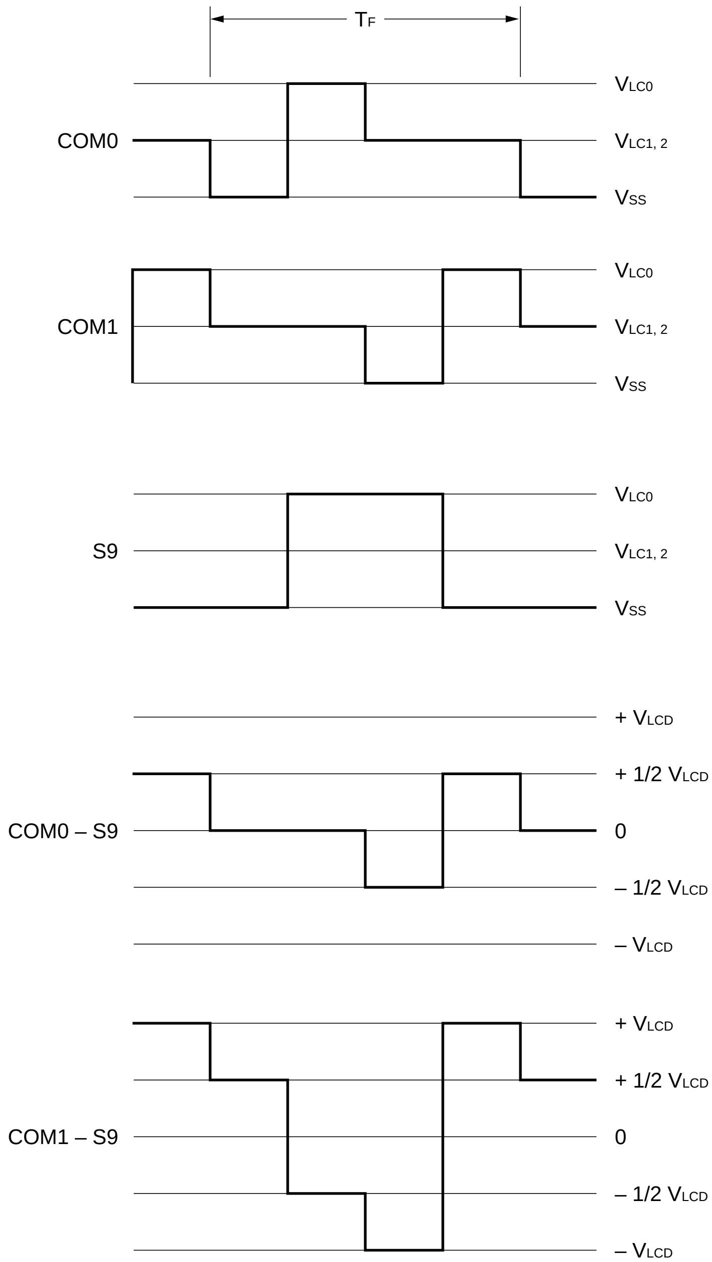


Figure 5-88. 2-Time-Division LCD Panel Wiring Example



Remark x: As this is a 2-time-division display, any data can be stored here at all times.

Figure 5-89. Example of 2-Time-Division LCD Drive Waveforms (1/2 Bias Modulus)



(3) 3-time-division display example

Figure 5-91 shows the connection between a 3-time-division type 10-digit LCD panel with the display pattern shown in Figure 5-90, and the μPD75308's segment signals (S0 to S29) and common signals (COM0 to COM2). The display example is "123456.7890", and the contents of the display data memory (addresses 1E0H through 1FDH) correspond to this value.

Here, the fifth digit "6." (6.) is considered for the sake of explanation. In accordance with the display pattern in Figure 5-90, using the COM0, COM1 and COM2 common signal timings, the selection and non-selection voltages shown in Table 5-19 must be output to pins S12 through S14.

Table 5-19. Selection/Non-Selection Voltages for Pins S12 to S14 (3-Time-Division Display Example)

Common \ Segment	S12	S13	S14
COM0	Non-selection	Selection	Selection
COM1	Selection	Selection	Selection
COM2	Selection	Selection	Non-selection

From this it can be seen that it is only necessary to set up "x110" in the display data memory (address 1ECH) corresponding to S12.

The LCD drive waveforms between S12 and each common signal are shown in Figure 5-92 (1/2 bias modulus) and 5-93 (1/3 bias modulus). It can be seen that when S12 is at the selection voltage in synchronization with COM1 selection timing, a $+V_{LCD}/-V_{LCD}$ (LCD illumination level) AC square wave is generated.

Figure 5-90. 3-Time-Division Type LCD Display Pattern and Electrode Wiring

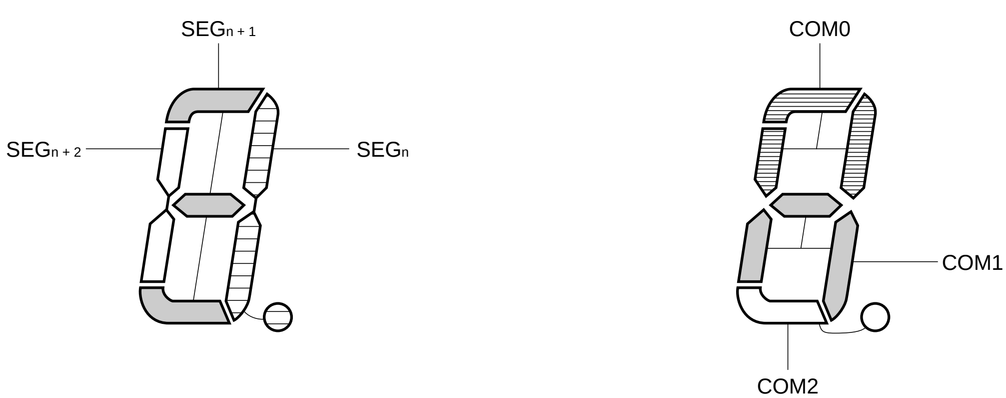
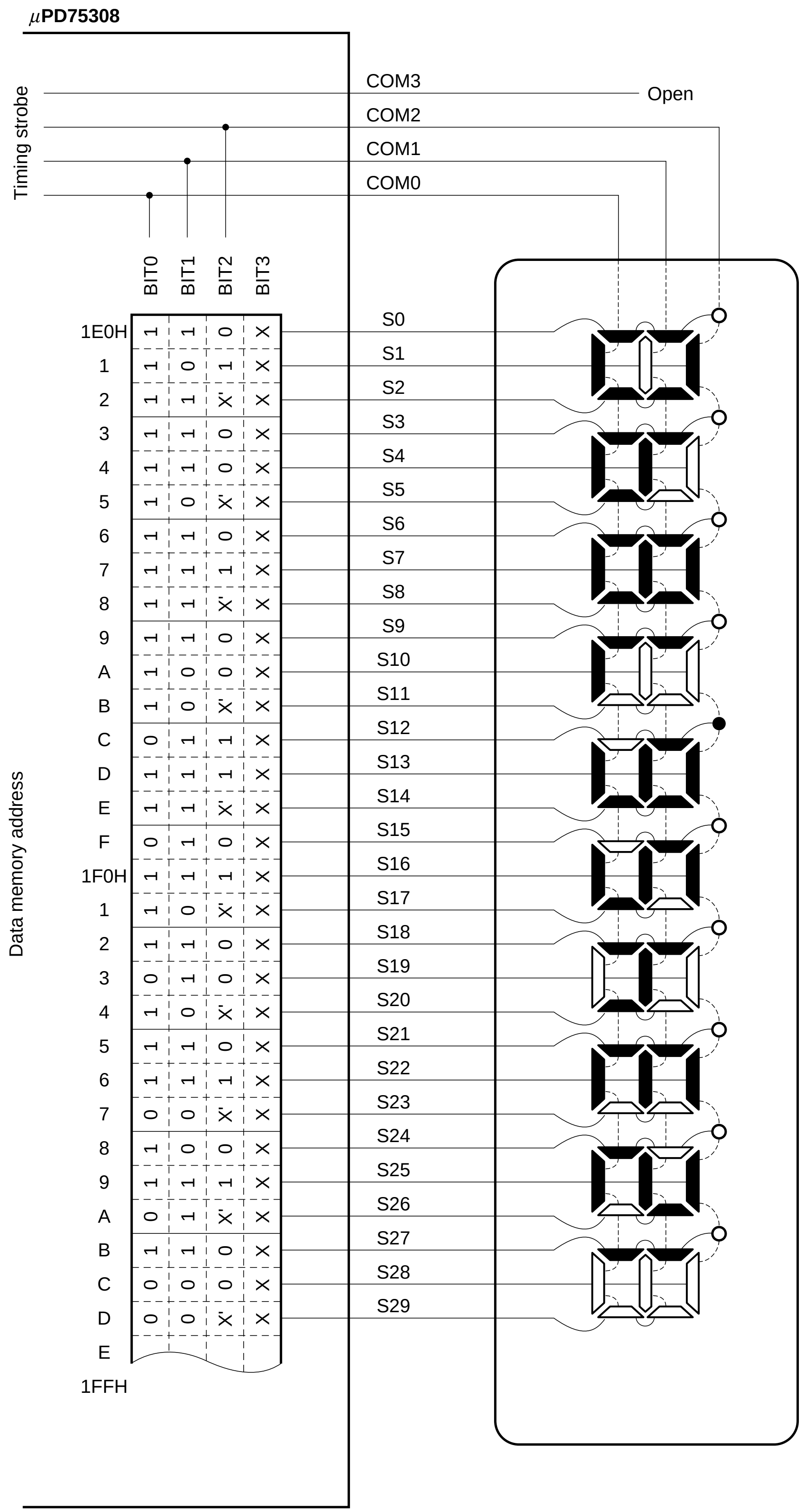


Figure 5-91. 3-Time-Division LCD Panel Wiring Example



Remark X' : As there is no segment corresponding to the LCD panel, any data can be stored here.
X : As this is a 3-time-division display, any data can be stored here at all times.

Figure 5-92. Example of 3-Time-Division LCD Drive Waveforms (1/2 Bias Modulus)

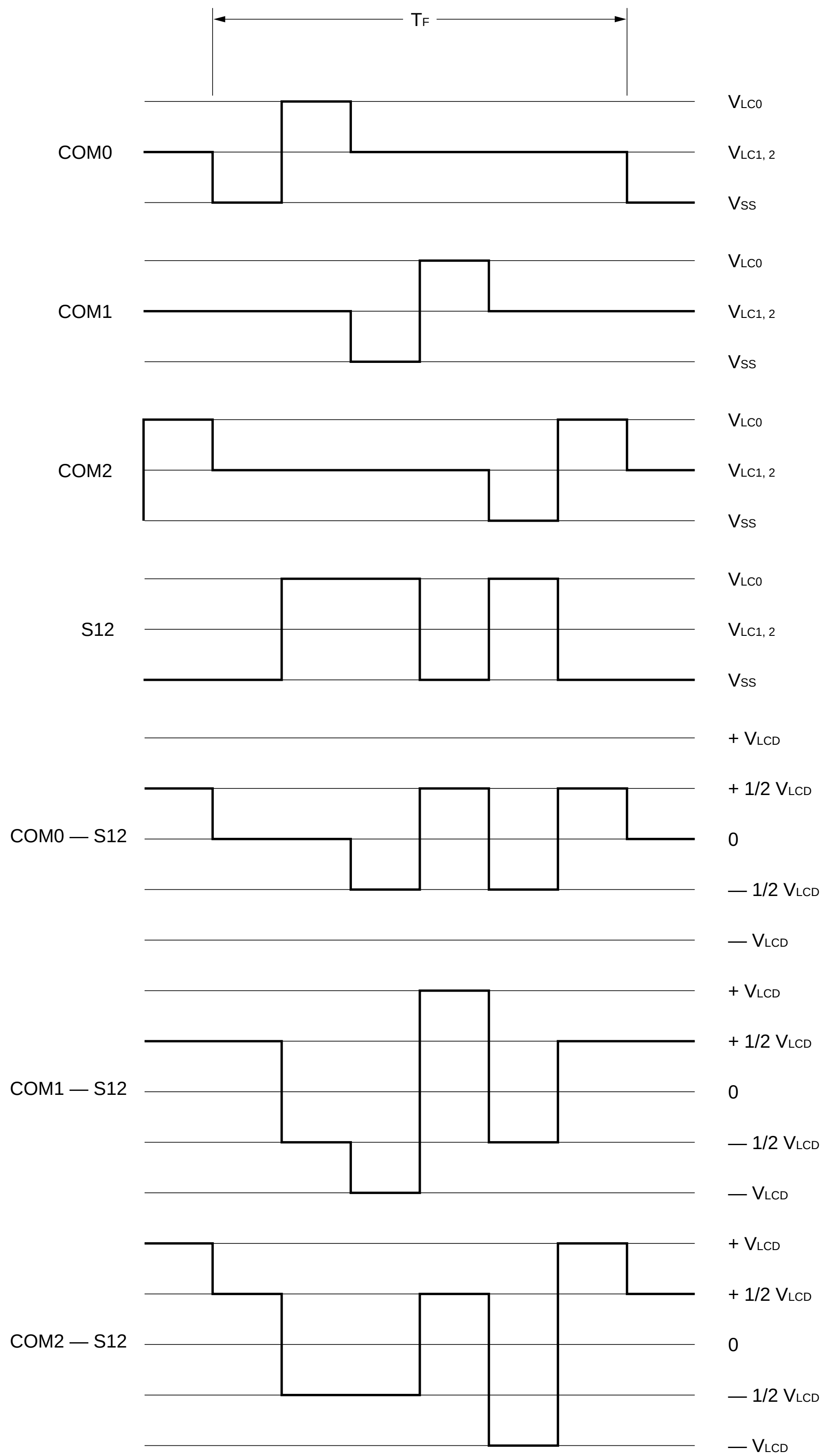
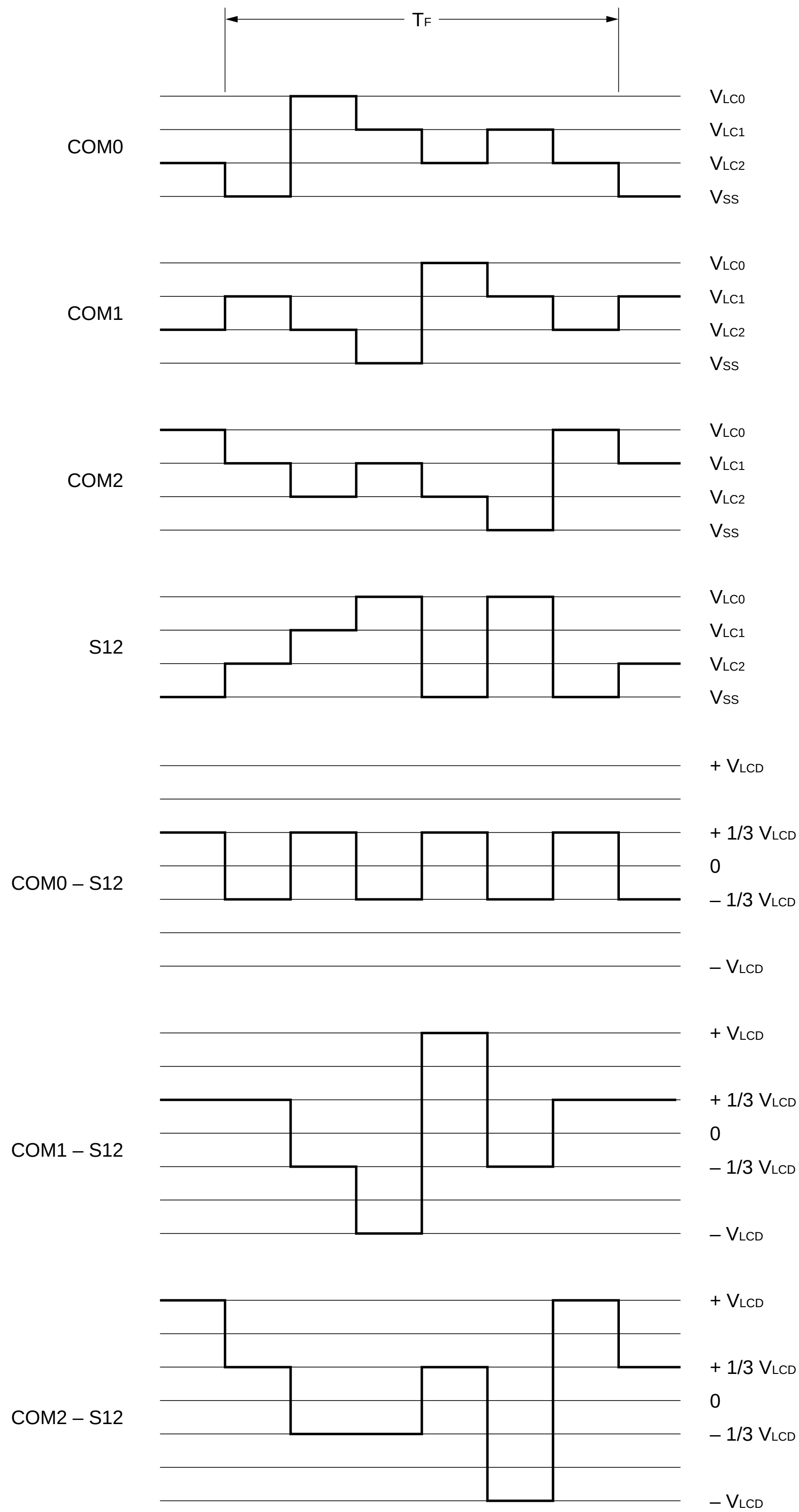


Figure 5-93. Example of 3-Time-Division LCD Drive Waveforms (1/3 Bias Modulus)



(4) 4-time-division display example

Figure 5-95 shows the connection between a 4-time-division type 16-digit LCD panel with the display pattern shown in Figure 5-94, and the μPD75308's segment signals (S0 to S31) and common signals (COM0 to COM3). The display example is "123456.7890123456", and the contents of the display data memory (addresses 1E0H through 1FFH) correspond to this value.

Here, the 11th digit "6." (6.) is considered for the sake of explanation. In accordance with the display pattern in Figure 5-94, using the COM0 through COM3 common signal timings, the selection and non-selection voltages shown in Table 5-20 must be output to pins S20 and S21.

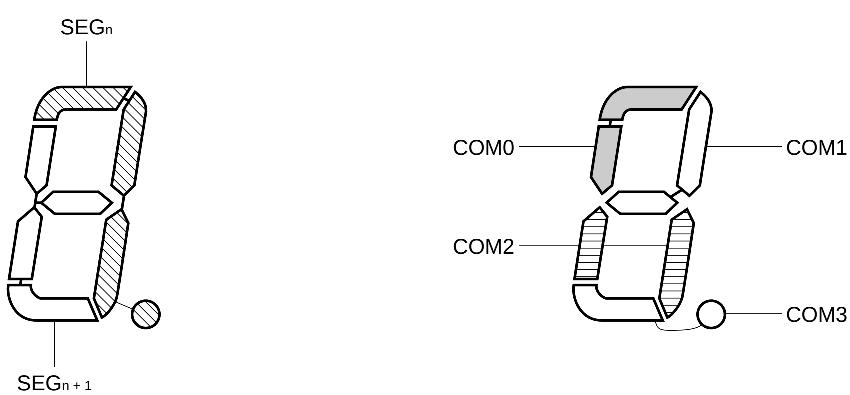
Table 5-20. Selection/Non-Selection Voltages for Pins S20 and S21 (4-Time-Division Display Example)

Common \ Segment	S20	S21
COM0	Selection	Selection
COM1	Non-selection	Selection
COM2	Selection	Selection
COM3	Selection	Selection

From this it can be seen that it is only necessary to set up "1101" in the display data memory (address 1F4H) corresponding to S20.

The LCD drive waveforms between S20 and the COM0 and COM1 common signals are shown in Figure 5-96 (for simplicity, waveforms to COM2 and COM3 are omitted). It can be seen that when S20 reaches the selection voltage in synchronization with COM0 selection timing, a $+V_{LCD}/-V_{LCD}$ (LCD illumination level) AC square wave is generated.

Figure 5-94. 4-Time-Division Type LCD Display Pattern and Electrode Wiring



Phase-out/Discontinued

Figure 5-95. 4-Time-Division LCD Panel Wiring Example

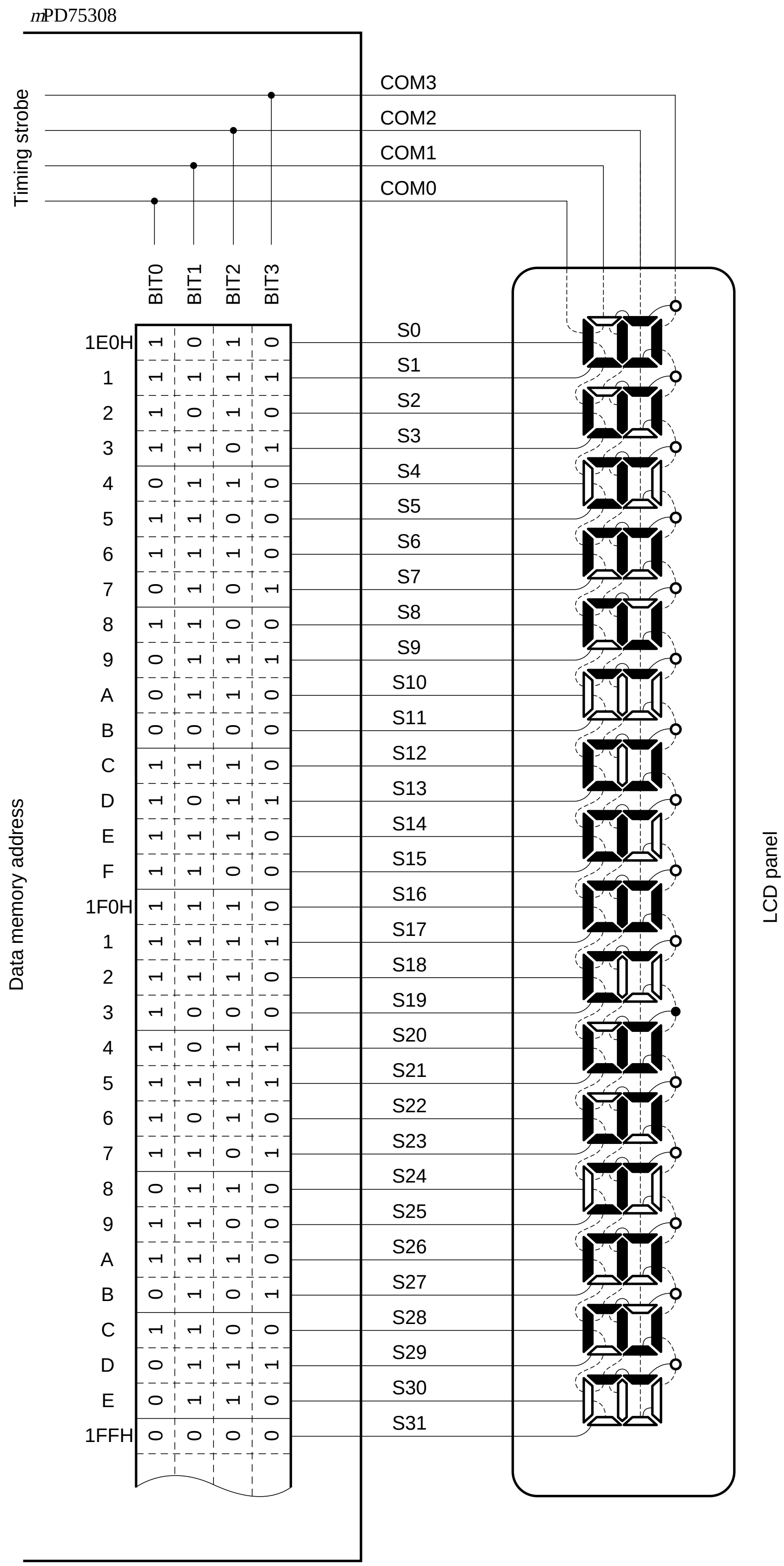
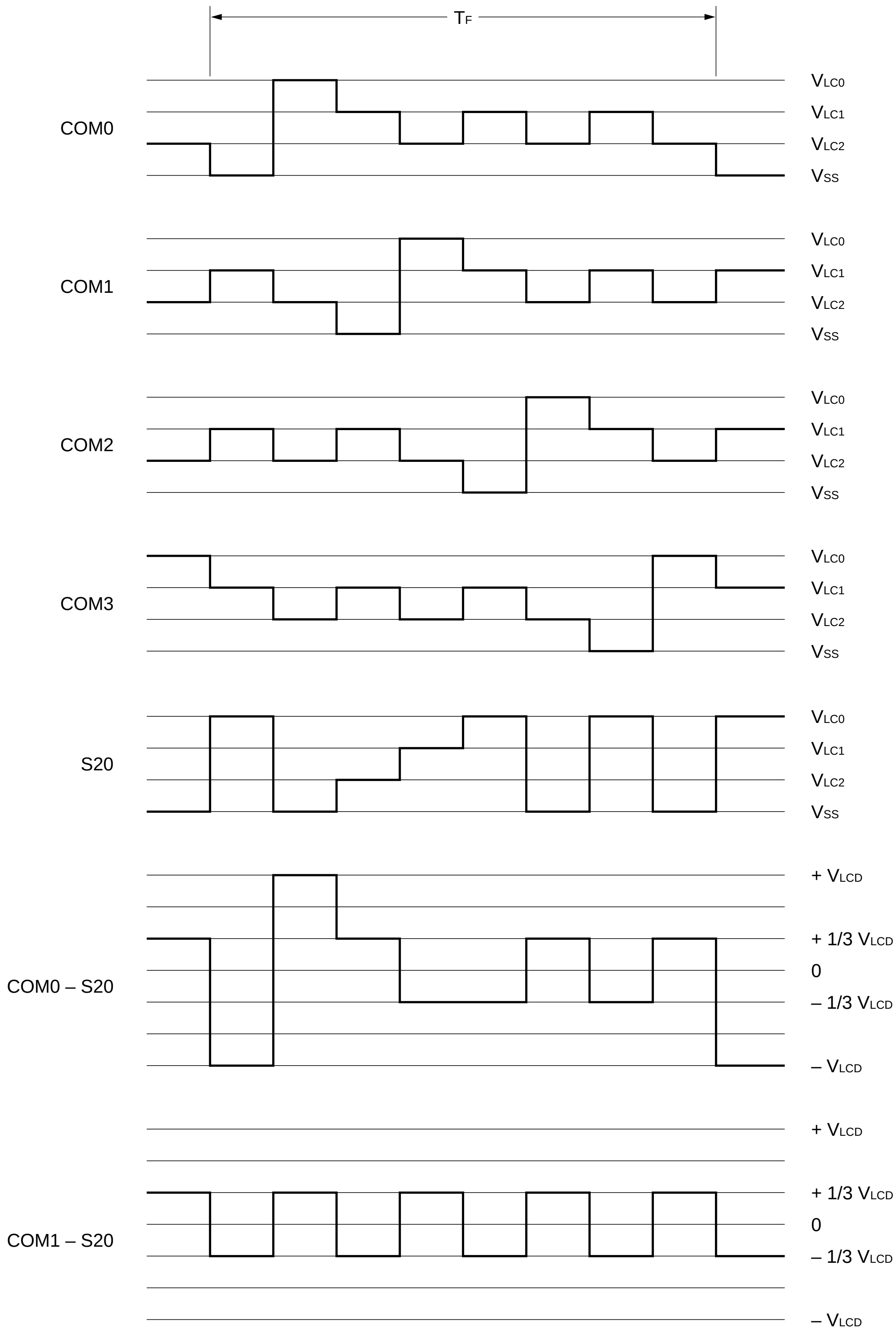


Figure 5-96. Example of 4-Time-Division LCD Drive Waveforms (1/3 Bias Modulus)

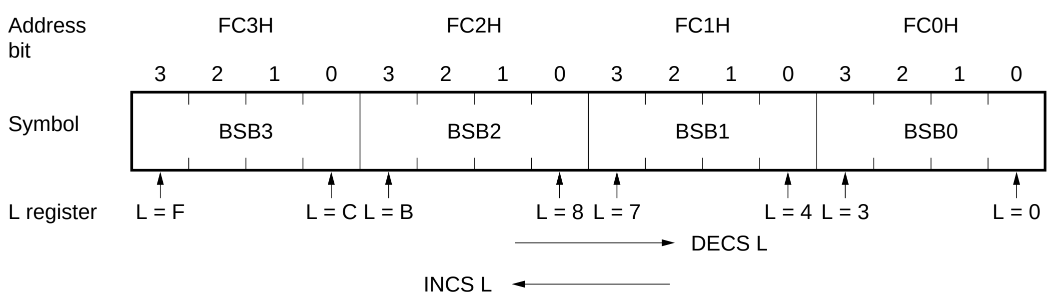


5.8 Bit Sequential Buffer ... 16 Bits

The bit sequential buffer is special data memory for bit manipulations. In particular, bit manipulation by sequential modification of the address and bit specification is facilitated, making this facility useful for bit-wise processing of data comprising many bits.

This data memory consists of 16 bits; bit manipulation instruction pmem.@L addressing can be used, and indirect bit specification is possible by means of the L register. In this case, processing can proceed while moving the specified bits sequentially by simply incrementing or decrementing the L register within the program loop.

Figure 5-97. Bit Sequential Buffer Format



Remark In pmem. @L addressing, the specification bit corresponding to the L register is moved.

Data manipulation is also possible using direct addressing. A combination of 1-bit, 4-bit, or 8-bit direct addressing and pmem. @L addressing can be used for continuous input or continuous output of 1-bit data, etc. In the case of 8-bit manipulation, the high-order and low-order 8 bits are handled separately by specifying BSB0 and BSB2.

Example To serially output two 16-bit data in BUFF 1, 2 from bit 0 of Port 3.

```
         CLR1    MBE
         MOV     XA, BUFF1
         MOV     BSB0, XA        ; set BSB0, 1
         MOV     XA, BUFF2
         MOV     BSB2, XA        ; set BSB2, 3
         MOV     L, #0
LOOP0:   SKT     BSB0, @L        ; test specified bit of BSB
         BR      LOOP1
         NOP                     ; dummy (timing adjustment)
         SET1    PORT3.0         ; set bit 0 of Port 3
         BR      LOOP2
LOOP1:   CLR1    PORT3.0         ; clear bit 0 of Port 3
         NOP                     ; dummy (timing adjustment)
         NOP
LOOP2:   INCS    L               ; L ← L + 1
         BR      LOOP0
         RET
```

[MEMO]

CHAPTER 6 INTERRUPT FUNCTION

The μPD75308 contains 6 vectored interrupt sources and 2 testable inputs, thus making various applications possible.

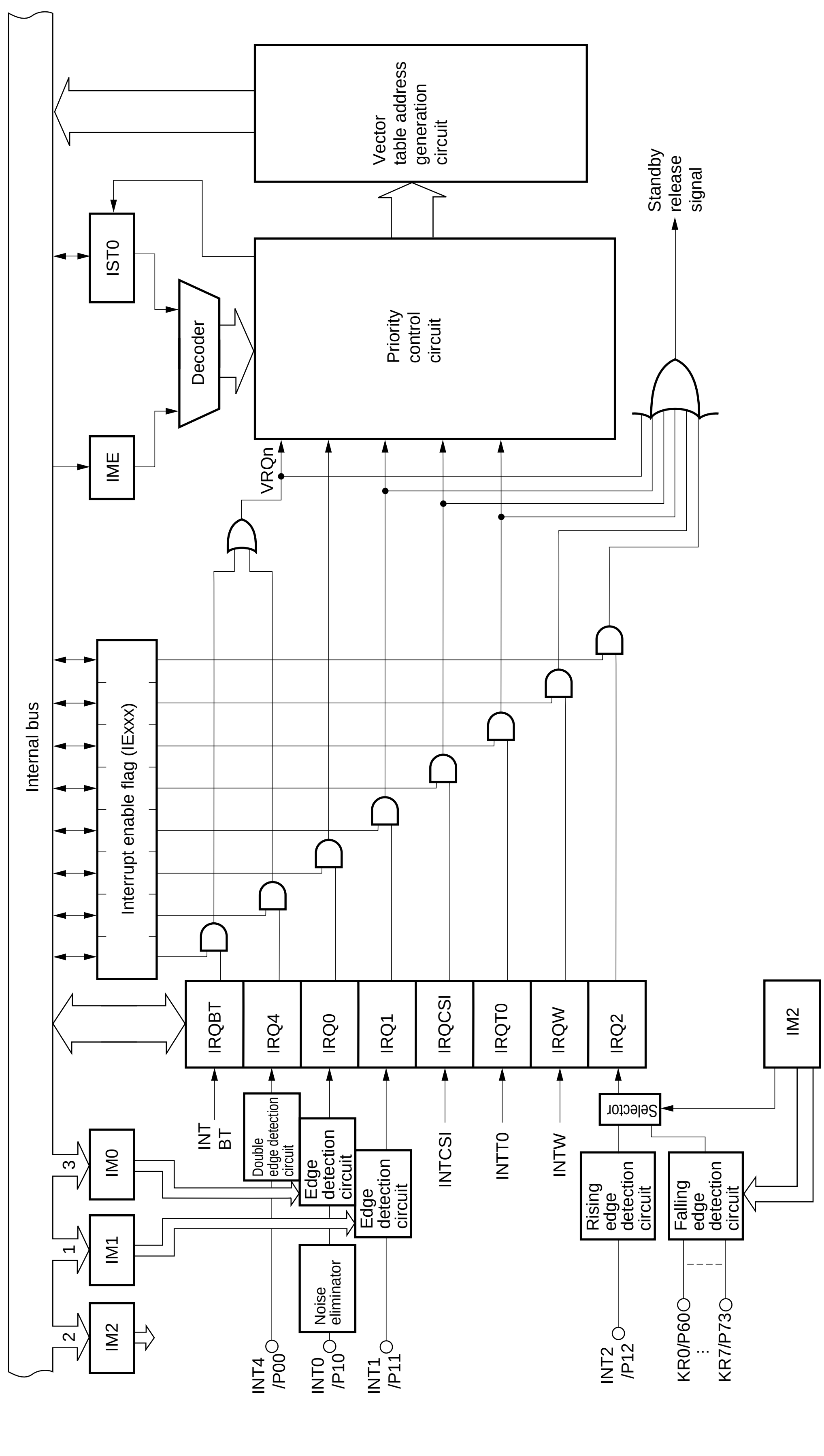
The μPD75308 interrupt control circuit has the following characteristics, making extremely high-speed interrupt servicing possible.

(a) Can control enabling or disabling of reception, based on the interrupt master enable flag (IME) and interrupt enable flag (IExxx).
(b) Can arbitrarily set the interrupt servicing start address and the MBE to interrupt servicing, based on the vector table (Actual interrupt service program starts earlier.)
(c) Can test and clear (can confirm the occurrence of an interrupt by means of the software) the interrupt request flag (IRQxxx)
(d) Can cancel (can select the cancellation source by the interrupt enable flag) the standby mode (STOP, HALT) by means of the interrupt request.

6.1 Interrupt Control Circuit Configurations

The interrupt control circuit configurations are shown in Figure 6-1. Each hardware is mapped in the data memory space.

Figure 6-1. Interrupt Control Circuit Block Diagram



Phase-out/Discontinued

6.2 Interrupt Source Types and Vector Table

The interrupt source types and the interrupt vector table are shown in Table 6-1 and Figure 6-2 respectively.

Table 6-1. Interrupt Source Types

Interrupt Source		Internal/ External	Interrupt Order[Note 1]	Vectored Interrupt Request Signal (Vector Table Address)
INTBT	(Basic time interval signal from basic interval timer)	Internal	1	VRQ1 (0002H)
INT4	(Both rising and falling edge detections are valid)	External		
INT0	(Rising or falling detection edge selection)	External	2	VRQ2 (0004H)
INT1		External	3	VRQ3 (0006H)
INTCSI	(Serial data transmission end signal)	Internal	4	VRQ4 (0008H)
INTT0	(Match signal between the programmable timer/ counter's count register and the module register)	Internal	5	VRQ5 (000AH)
INT2[Note 3]	(Rising edge detection of the input to the INT2 pin and falling edge detection of either of the inputs to KR0 to KR7)[Note 2]	External	Testable input signal (Set IRQ2 and IRQW.)	
INTW[Note 3]	(Signal from the watch timer)	Internal		

Notes 1. The interrupt order means the order of priority given to multiple interrupt requests.

2. For details of INT2, please refer to **6.3 (4) INT2 and key interrupt 0 to 7 (KR0 to KR7) hardware**.

3. This is a test source. It is affected by an interrupt enable flag in the same way as an interrupt source, but does not generate a vectored interrupt.

Figure 6-2. Interrupt Vector Table (1/2)

(a) In the case of μPD75304, 75304B

Address					
002H	MBE	0	0	0	INTBT/INT4 start address (High-order 4 bits)
	INTBT/INT4 start address (Low-order 8 bits)				
004H	MBE	0	0	0	INT0 start address (High-order 4 bits)
	INT0 start address (Low-order 8 bits)				
006H	MBE	0	0	0	INT1 start address (High-order 4 bits)
	INT1 start address (Low-order 8 bits)				
008H	MBE	0	0	0	INTCSI start address (High-order 4 bits)
	INTCSI start address (Low-order 8 bits)				
00AH	MBE	0	0	0	INTT0 start address (High-order 4 bits)
	INTT0 start address (Low-order 8 bits)				

Figure 6-2. Interrupt Vector Table (2/2)

(b) In the case of μPD75306, 75306B, 75308, 75308B and 75P308

Address				
002H	MBE	0	0	INTBT/INT4 start address (High-order 5 bits)
				INTBT/INT4 start address (Low-order 8 bits)
004H	MBE	0	0	INT0 start address (High-order 5 bits)
				INT0 start address (Low-order 8 bits)
006H	MBE	0	0	INT1 start address (High-order 5 bits)
				INT1 start address (Low-order 8 bits)
008H	MBE	0	0	INTCSI start address (High-order 5 bits)
				INTCSI start address (Low-order 8 bits)
00AH	MBE	0	0	INTT0 start address (High-order 5 bits)
				INTT0 start address (Low-order 8 bits)

(c) In the case of μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A and 75P316B

Address			
002H	MBE	0	INTBT/INT4 start address (High-order 6 bits)
			INTBT/INT4 start address (Low-order 8 bits)
004H	MBE	0	INT0 start address (High-order 6 bits)
			INT0 start address (Low-order 8 bits)
006H	MBE	0	INT1 start address (High-order 6 bits)
			INT1 start address (Low-order 8 bits)
008H	MBE	0	INTCSI start address (High-order 6 bits)
			INTCSI start address (Low-order 8 bits)
00AH	MBE	0	INTT0 start address (High-order 6 bits)
			INTT0 start address (Low-order 8 bits)

In the table above, the interrupt order indicates the order of executing the interrupts when multiple interrupt requests occur simultaneously or when multiple interrupt requests are held.

The interrupt servicing start address and the set value of the MBE in the interrupt servicing are written in the vector table. The vector table is set according to the assembler directive (VENTn).

Example INTBT/INT4 vector table setting

VENT1 MBE = 0, RBE = 0, GOTOBT

① ② ③ ④

① Vector table of address 0002
② MBE setting in interrupt servicing routine
③ Make sure to set “0” for RBE.
④ Symbol indicating the start address of the interrupt servicing routine

Cautions 1. The vector table address set by the VENTn (n = 1 to 5) instruction is 2n.

2. In the VENTn instruction, make sure to set "0" for RBE.

Example Vector table setting of INTBT/INT4 and INTT0

```
VENT1    MBE = 0, RBE = 0, GOTOBT
VENT5    MBE = 0, RBE = 0, GOTOT0
```

6.3 Various Hardware Types of Interrupt Control Circuit

(1) Interrupt request flag and interrupt enable flag

There are eight interrupt request flags (IRQxxx) corresponding to interrupt sources (interrupt: 6, test: 2), as follows.

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 interrupt request flag (IRQT0)
Watch timer interrupt request flag (IRQW)

The interrupt request flag is set (1) when the interrupt request occurs, and then cleared automatically (0) when the interrupt servicing is executed. However, the IRQBT and the IRQ4 differ in their clearance operations because they share the same vector table (refer to **6.6 Vector Address Sharing Interrupt Servicing**).
There are eight interrupt enable flags (IExxx) corresponding to interrupt request flags, as follows.

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 interrupt enable flag (IET0)
Watch timer interrupt enable flag (IEW)

The interrupt enable flag is provided for each interrupt request flag. An interrupt is enabled if the contents are “1” but disabled if “0”.
If the interrupt request flag is set and the interrupt enable flag is enabling the interrupt, the vectored interrupt request (VRQn) is generated. This signal is used even in the release of the standby mode.
The interrupt request flag and the interrupt enable flag are operated by the bit manipulation instruction and 4-bit memory manipulation instruction. In the case of the bit manipulation instruction, direct operations are always possible regardless of the MBE setting. The interrupt enable flag is operated by the EI IExxx instruction and the DI IExxx instruction. Normally, the SKTCLR instruction is used in the interrupt request flag test.

Example
```
EI       IE0      ; enable INT0
DI       IE1      ; disable INT1
SKTCLR   IRQCSI   ; If IRQCSI is 1, the interrupt request flag is skipped and cleared
```

If the interrupt request flag is set by the instruction, even if no interrupt has occurred, the vectored interrupt is executed as if the interrupt has occurred.
The interrupt request flag and the interrupt enable flag are cleared (0) by the occurrence of the $\overline{\text{RESET}}$ signal, thus disabling all the interrupts.

Table 6-2. Interrupt Request Flag Set Signal

Interrupt Request Flag	Interrupt Request Flag Set Signal	Interrupt Enable Flag
IRQBT	Set with basic time interval signal by basic interval timer	IEBT
IRQ4	Set by either the rising or the falling edge detection of the INT4/P00 pin input signal	IE4
IRQ0	Set by the edge detection of the INT0/P10 pin input signal. The detection edge is select by the INT0 mode register (IM0).	IE0
IRQ1	Set by the edge detection of the INT1/P11 pin input signal. The detection edge is selected by the INT1 mode register (IM1).	IE1
IRQCSI	Set by the serial data transmission operation end signal of the serial interface.	IECSI
IRQT0	Set by the match signal from the timer/event counter #0.	IET0
IRQW	Set by the signal from the watch timer.	IEW
IRQ2	Set by the rising edge detection of the INT2/P12 pin input signal or the falling edge detection of either of the inputs to KR0/P60 to KR7/P73 pins	IE2

(2) Interrupt master enable flag (IME)

The interrupt master enable flag specifies enabling/disabling of acknowledgment of all interrupts.
IME is set (1) and reset (0) by the EI and DI instructions respectively.
When a $\overline{\text{RESET}}$ signal is generated this flag is cleared to 0, and acknowledgement of all interrupts is disabled.

Figure 6-3. IME Format

Address	3
FB2H	IME

Interrupt Master Enable Flag (IME)

0	All interrupts are disabled, and vectored interrupts are not initiated.
1	Interrupt enabling/disabling is controlled by the corresponding interrupt enable flag.

(3) INT0, INT1 and INT4 hardware

(a) The configuration of INT0 is shown in Figure 6-4 (a). This pin is an external interrupt input for which rising or falling edge detection can be selected

INT0 has a function for noise elimination by means of a sampling clock (refer to **Figure 6-5 Input/ Output Timing of Noise Eliminator**). The noise eliminator eliminates as noise any pulse less than two sampling clock cycles[Note] in width. However, a pulse that is at least one sampling clock cycle in width may be acknowledged as an interrupt signal depending on the timing at which sampling is performed (refer to **Figure 6-5 ② (a)**). A pulse that is two sampling clock cycles or more in width can definitely be acknowledged as an interrupt signal.

INT0 has two sampling clocks, Φ and $f_X/64$, either of which can be selected and used. The selection is made by edge detection mode register bit 3 (IM03) (refer to **Figure 6-6 (a)**).

Selection of the detected edge is performed by an 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. Generation of the reset signal clears all bits to 0 and selects the rising edge specification.

Note When sampling clock is Φ : $2t_{CY}$
When sampling clock is $f_X/64$: $128/f_X$

Cautions 1. INT0 does not operate in the standby mode since sampling is performed by the clock.

2. Because the INT0/P10 pin is internally connected to a noise eliminator even when it is used as a port pin, input a pulse with a width of 2 cycles or more of the sampling clock to this pin.

(b) The configuration of INT1 is shown in Figure 6-4(b). This pin is an external interrupt input for which rising or falling edge detection can be selected.

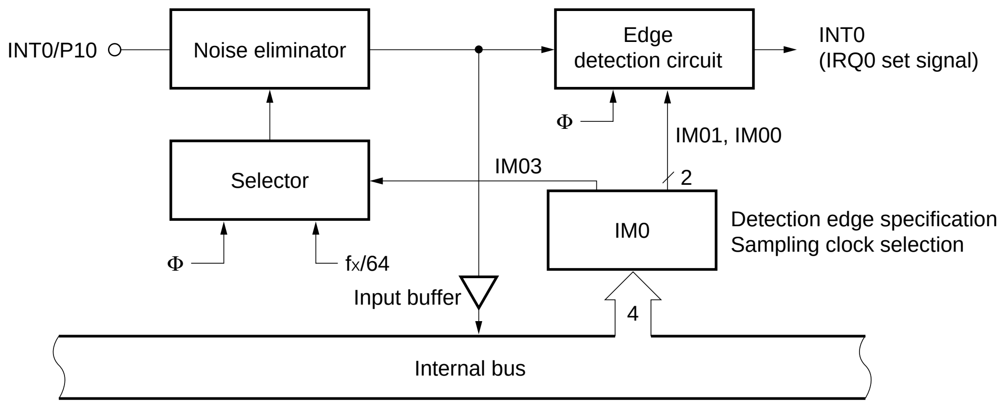
Selection of the detected edge is performed by an edge detection mode register (IM1).

The format of IM1 is shown in Figure 6-6(b). IM1 is set by a bit manipulation instruction. Generation of the reset signal clears all bits to 0 and selects the rising edge specification.

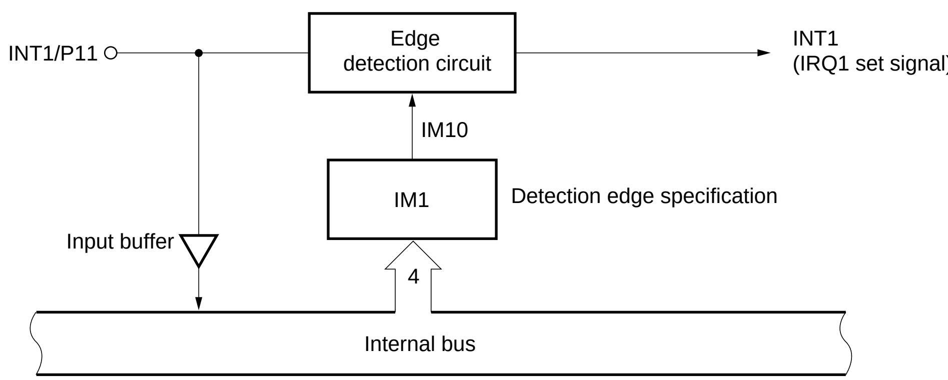
(c) The configuration of INT4 is shown in Figure 6-4(c). This pin is an external interrupt input which is capable of both rising and falling edge detection.

Figure 6-4. Configuration of INT0, INT1 and INT4

(a) INT0 hardware



(b) INT1 hardware



(c) INT4 hardware

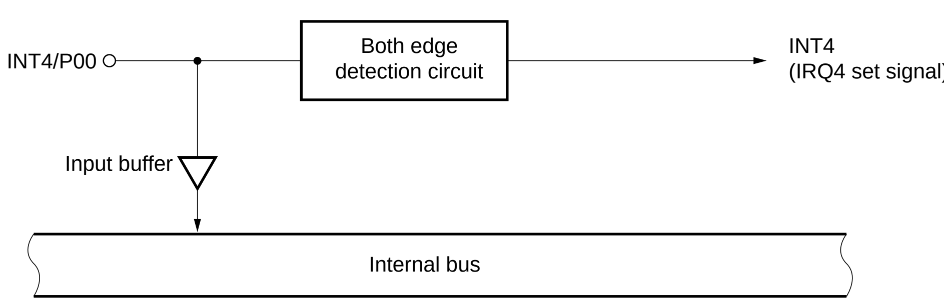
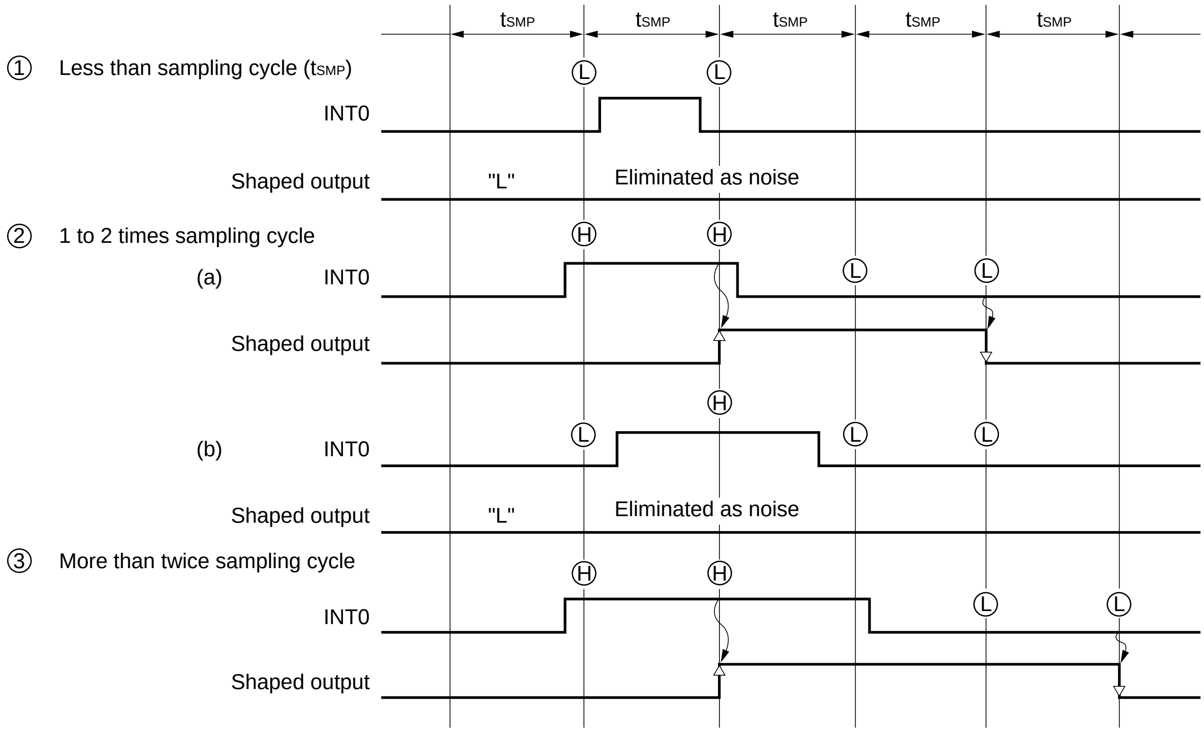


Figure 6-5. Input/Output Timing of Noise Eliminator



Remark $t_{SMP} = t_{CY}$ or $64/f_X$

Figure 6-6. Format of Edge Detection Mode Registers

(a) INT0 edge detection mode register (IM0)

Address	3	2	1	0	Symbol
FB4H	IM03	0	IM01	IM00	IM0

Edge Detection Specification

IM01	IM00	
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

IM03	
0	Φ (0.95, 1.91, 15.3 μs: 4.19 MHz operation)
1	f_X/64 (15.3 μs: 4.19 MHz operation)

(b) INT1 edge detection mode register (IM1)

Address	3	2	1	0	
FB5H	0	0	0	IM10	IM1

IM10	
0	Rising edge specification
1	Falling edge specification

(c) INT2 edge detection mode register (IM2)

FB6H	0	0	IM21	IM20	IM2

IM21	IM20	INT2 Interrupt Source	Interrupt Input Pin
0	0	INT2 pin input rising edge specification	INT2 (1 pin)
0	1	Any KRx pin input falling edge specification	KR4 to KR7 (4 pins)
1	0		KR2 to KR7 (6 pins)
1	1		KR0 to KR7 (8 pins)

Caution Since an interrupt request flag may be set when an edge detection mode register is modified, interrupts should be enabled after first disabling interrupts, then modifying the mode register and clearing the interrupt request flag with the CLR1 instruction. 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 mode register modification.

(4) INT2 and key interrupt 0 to 7 (KR0 to KR7) hardware

The configurations of INT2 and KR0 through KR7 are shown in Figure 6-7. There are two IRQ2 sets as follows, either of which can be selected by the edge detection mode register (IM2).

(a) INT2 pin input rising edge detection

IRQ2 is set when the rising edge of the INT2 pin input is detected.

(b) Falling edge detection in input of any of pins KR0 to KR7 (key interrupt)

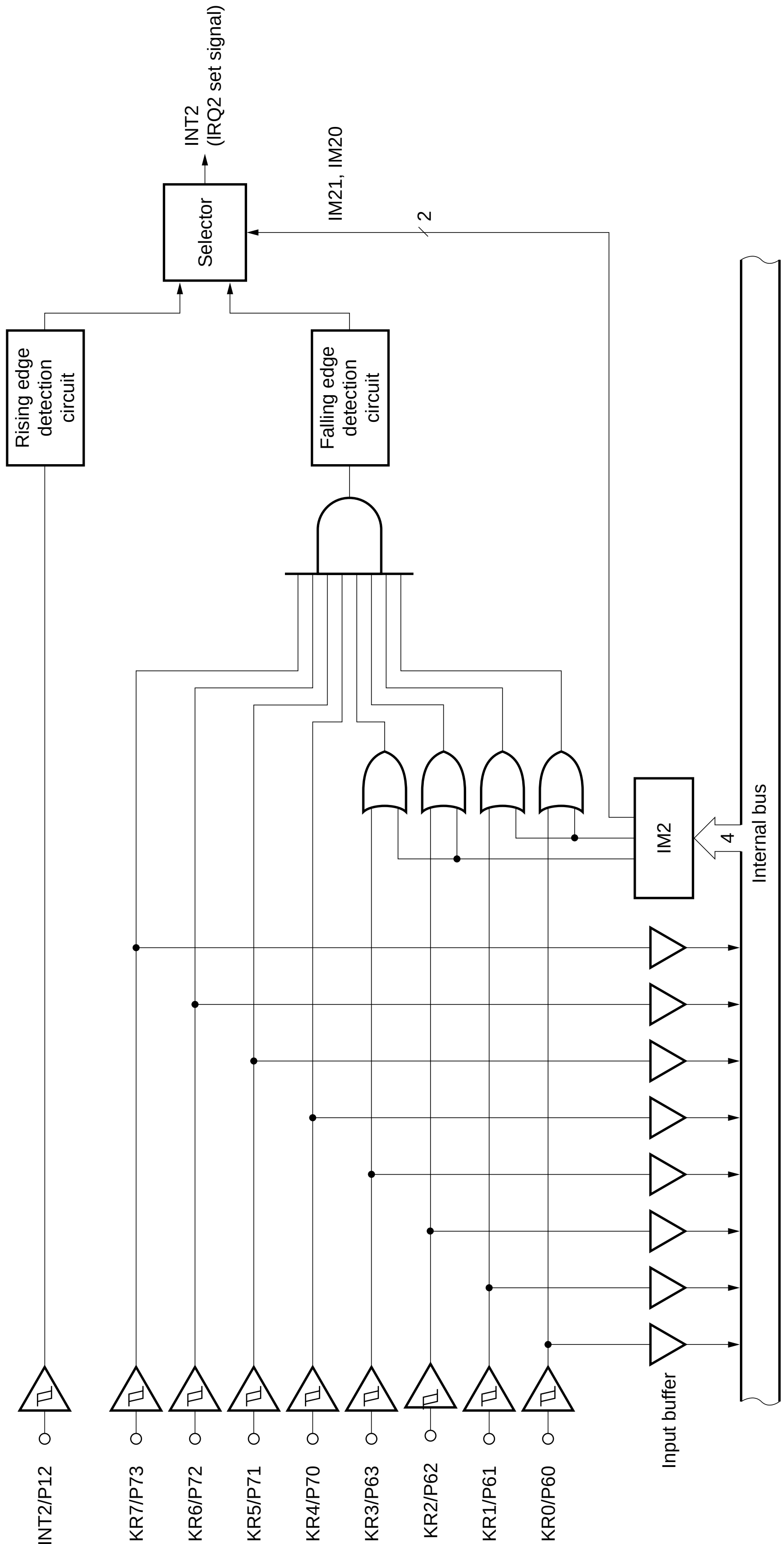
Pins to be used for interrupt input are selected by the edge detection mode register (IM2) from KR0 to KR7. IRQ2 is set when a falling edge is detected in the input of any of the pins.

Caution When a low-level signal is input on at least one of the pins selected as falling edge detection pins, IRQ2 is not set even if a falling edge is input on another pin.

The format of IM2 is shown in Figure 6-6(c). IM2 is set by a 4-bit manipulation instruction. Generation of the reset signal clears all bits to 0, selecting the INT2 rising edge specification.

Phase-out/Discontinued

Figure 6-7. INT2, KR0 to KR7 Configurations



(5) Interrupt status flag

The interrupt status flag (IST0) shows the status of the processing currently being executed by the CPU and is included in the PSW.

As shown in Table 6-3, the interrupt priority control circuit controls multiinterrupts, based on the contents of the flag.

Because IST0 can be changed by the 4-bit manipulation instruction or the bit manipulation instruction, it can also perform multiinterrupts by changing the status in execution. IST0 can be operated in terms of bit at all times regardless of the MBE setting.

Before operating IST0, make sure to disable the interrupt by executing the DI instruction; after operation, make sure to enable the interrupt by executing the EI instruction.

When receiving the interrupt, IST0 is saved in the stack memory together with other PSWs and then is automatically set to 1. When the RETI instruction has been executed, the original IST0 value (0) is restored.

When occurrence of the $\overline{\text{RESET}}$ signal, the flag contents are cleared (0).

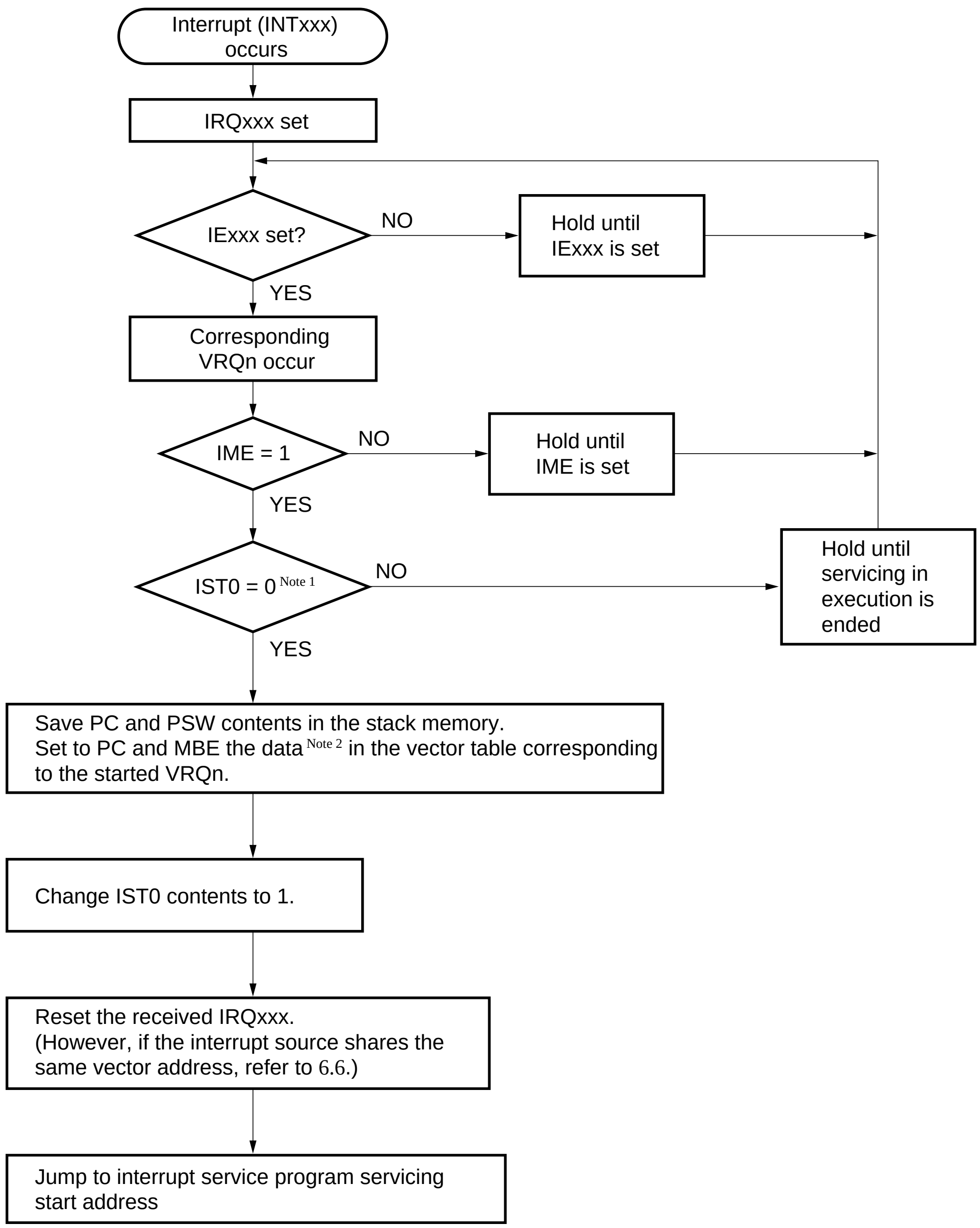
Table 6-3. IST0 and Interrupt Servicing Status

IST0	Processing status in Execution	CPU Processing Contents	Receivable Interrupt Request	After Receiving Interrupt IST0
0	Status 0	Normal program being processed	All interrupt reception enabled	1
1	Status 1	Interrupt being serviced	All interrupt reception disabled	–

6.4 Interrupt Sequence

If an interrupt occurs, it is serviced in the procedure shown in Figure 6-8.

Figure 6-8. Interrupt Servicing Procedure



Notes 1. IST0: Interrupt status flag (PSW's bit 2; refer to **Table 6-3**)

2. Store, in each vector table, the interrupt service program start address and the MBE set value in starting the interrupt.

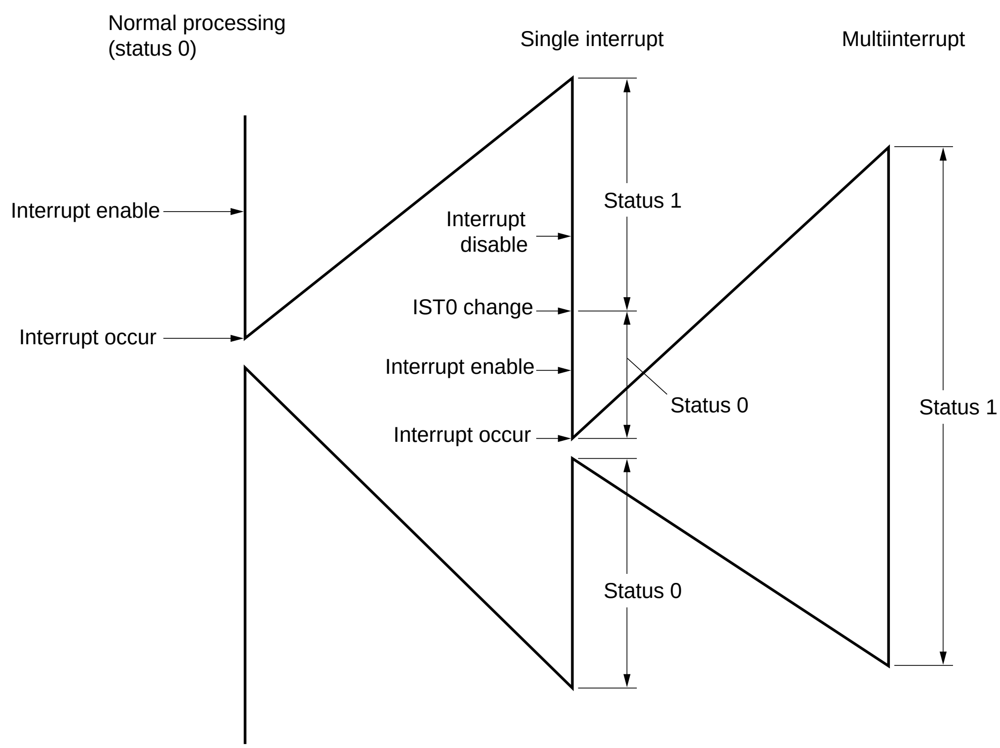
6.5 Multiinterrupt Servicing Control

The μPD75308 is capable of multiinterrupt in the following method.

As Table 6-3 shows, multiinterrupts are made possible by changing the interrupt status flag with the program, that is, by changing the interrupt servicing program IST0 to 0, and the status 0.

IST0 is changed with the interrupt disabled by the DI instruction beforehand.

Figure 6-9. Multiinterrupt by Interrupt Status Flag Change



6.6 Vector Address Sharing Interrupt Servicing

As INTBT and INT4 interrupt sources share the same vector table, the interrupt source is selected as shown below.

(1) When using the interrupts only on one side

Between the two interrupt source types that share the vector table, set (1) the interrupt enable flag of the interrupt source needed and clear (0) the other interrupt enable flag. In this case, the interrupt request is generated by the interrupt source of the enabled (IExxx = 1) side, and once received, its interrupt request flag is reset.

(2) When using the interrupts of both sides

Set (1) both of the interrupt enable flags that correspond to the two interrupt source types. In this case, the logical sum of the interrupt request flags of the two interrupt source types becomes the interrupt request.

In this case, even if the interrupt request has been received by setting the interrupt request flag of one side or both sides, the request flag of either side is not reset.

Therefore, in this case, it is necessary to determine, by using the service routine, which interrupt source the interrupt is from. It is performed by executing the DI instruction at the beginning of the service routine and checking the interrupt request flag with the SKTCLR instruction.

When this interrupt request flag has been tested and cleared, if the request flags of both sides are set, even if the flag of one side is cleared, the interrupt request remains. If IST0 is cleared, the remaining interrupt request starts the multiinterrupt servicing.

Remark When only the interrupts of one side are enabled, the source of the interrupt occurred being clear, the interrupt request flag is reset by the hardware when receiving the interrupt.
On the other hand, when the interrupts of both sides are enabled, either of the interrupt sources that have occurred is unclear and therefore the interrupt request flag cannot be reset by the hardware. Consequently, the interrupt request flag is checked by the software to determine the interrupt source. Flag resetting is performed by the software.

Example When giving priority to INT4 ★

```
         DI
         SKTCLR   IRQ4      ; IRQ4 = 1?
         BR       VSUBBT    ⎫
           :                ⎬ INT4 processing routine
EIRETI:  EI                 ⎪
         RETI               ⎭

VSUBBT:  SKTCLR   IRQBT     ; IRQBT = 1?Note
         BR       EIRETI    ; IRQBT v IRQ4 = 0
           :                ⎫ INTBT processing routine
         BR       EIRETI    ⎭
```

Note Be sure to also test interrupts that do not take precedence.
Otherwise, interrupt servicing is not executed as expected. The reason for this is as follows:

Example (If only interrupt that takes precedence is tested)

```
         Main routine
              :
         VENT1   MBE = 0, RBE = 0, VSUB4
              :
INT4 ──→ NOP
         CLR1    IRQ4
              :
```

```
Vector routine

VSUB4:   SKTCLR1  IRQ4
         BR       VSUBBT  ┐
            :             ├ INT4 processing routine
         RETI             ┘
            :
VSUBBT:  CLR1     IRQBT   ┐
            :             ├ IRQBT processing routine
         RETI             ┘
```

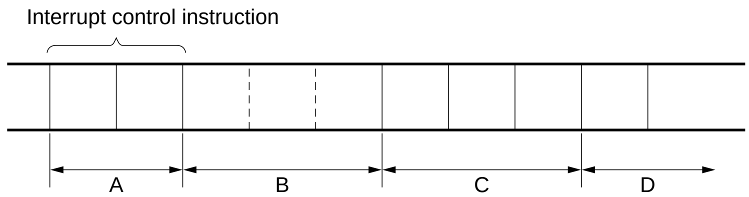
If the edge of INT4 is generated while the "NOP" instruction in the above main routine is executed, the CPU starts interrupt servicing. Because "NOP" is a one-machine cycle instruction, however, the CPU first executes the "CLR1 IRQ4" instruction and then starts the interrupt vector routine (refer to **6.7 Machine Cycles until Interrupt Servicing**).
The vector routine tests IRQ4 by executing the "SKTCLR1 IRQ4" instruction. However, because IRQ4 = 0, execution jumps to "VSUBBT". As a result, the IRQBT processing routine is executed.
For this reason, be sure to also test interrupts that do not take precedence.

6.7 Machine Cycles until Interrupt Servicing

On the μPD75308, the number of machine cycles executed from the time the interrupt request flag (IRQn) is set until the interrupt routine program is executed is as follows.

(1) When IRQn is set during execution of an interrupt control instruction

When the IRQn flag 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 : IRQn set
B : Execution of next instruction (between 1 and 3 machine cycles depending on the instruction)
C : Interrupt servicing (3 machine cycles)
D : Execution of interrupt routine

Remarks 1. An interrupt control instruction is an instruction which manipulates instruction related hardware (data memory FB x address). These comprise the DI instruction and the EI instruction.

2. The 3 machine cycles of interrupt servicing represent the time for stack manipulation on acknowledgement of the interrupt, etc.

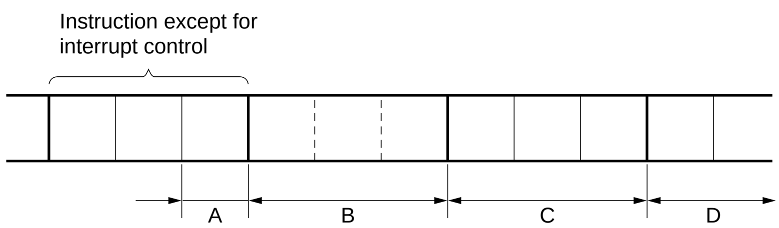
Cautions 1. In the case of consecutive interrupt control instructions, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the next instruction after the last interrupt control instruction executed.

2. If the DI instruction is executed when IRQn is set (A in Figure above) or subsequently, the set IRQn interrupt request is held pending until the next EI instruction is executed.

(2) When IRQn is set during execution of an instruction other than those in (1).

(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 next instruction after the instruction being executed.

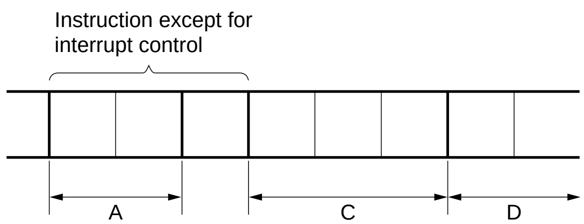


A : IRQn set
B : Execution of next instruction (between 1 and 3 machine cycles depending on the instruction)
C : Interrupt servicing (3 machine cycles)
D : Execution of interrupt routine

Caution When 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 next instruction after the last interrupt control instruction executed. When the interrupt control instruction executed after the instruction by which the IRQn flag 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 : IRQn set
C : Interrupt servicing (3 machine cycles)
D : Execution of interrupt routine

6.8 Effective Methods of Using Interrupts

The interrupt function can be used effectively as follows:

(1) In the interrupt servicing routine, make MBE = 0.

The 00H to 7FH address is given priority for allocation of the data memory used in the interrupt servicing routine. Thus, if MBE = 0 is specified in the interrupt vector table, it is possible to program without recognizing the memory bank.

If the program is obliged to use the memory bank 1, the memory bank select register is saved by the PUSH BS instruction and then memory bank 1 is selected.

(2) Use the software interrupt in debugging

Even if the interrupt request flag is set by the instruction, the system operates in the same manner as when the interrupt has occurred. Irregular or simultaneous interrupts can be debugged effectively by using the instruction to set the interrupt request flag.

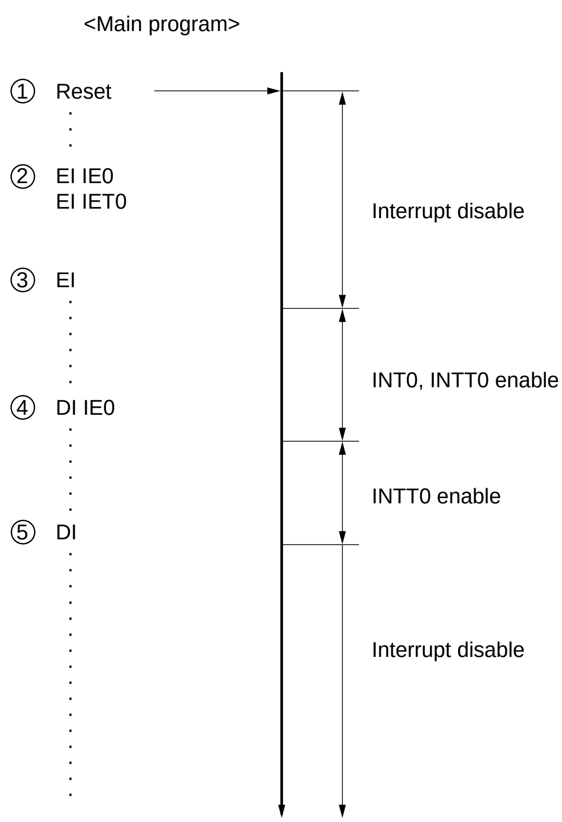
6.9 Application of Interrupts

When using the interrupt function, first of all set the main program as follows.

① Set the interrupt enable flag to be used (IE IExxx instruction)
② When using INT0 and INT1, select the active edge. (Set IM0 and IM1.)
③ Set (EI instruction) the interrupt master enable flag (IME).

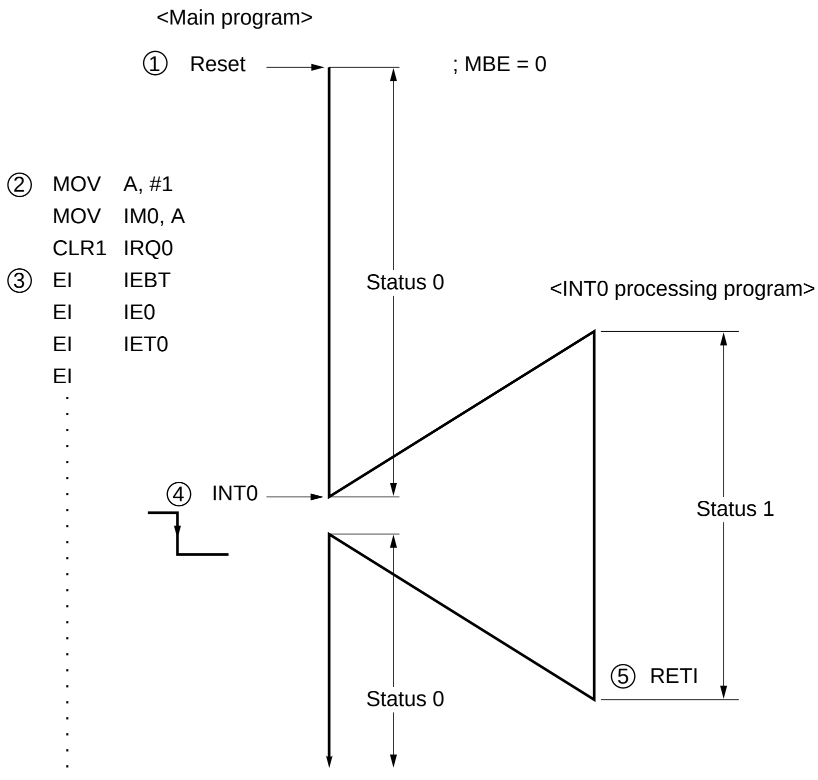
In the interrupt servicing program, MBE is set by the vector table.
The RETI instruction is used for restoration from the interrupt servicing program.

(1) Interrupt enable/disable



① All the interrupts are disabled by the $\overline{\text{RESET}}$ signal.
② Set the interrupt enable flag by the EI IExxx instruction. At this stage, all the interrupts are still disabled.
③ Set the interrupt master enable flag by the EI instruction. At this stage, INT0 and INTT0 are enabled.
④ The interrupt enable flag is cleared and INT0 is disabled by the DI IExxx instruction.
⑤ All the interrupts are disabled by the DI instruction.

(2) Example of using INTBT, INT0 (falling edge active) and INTT0: No multiinterrupts



① All the interrupts are disabled and the status is made 0 by the $\overline{\text{RESET}}$ signal.

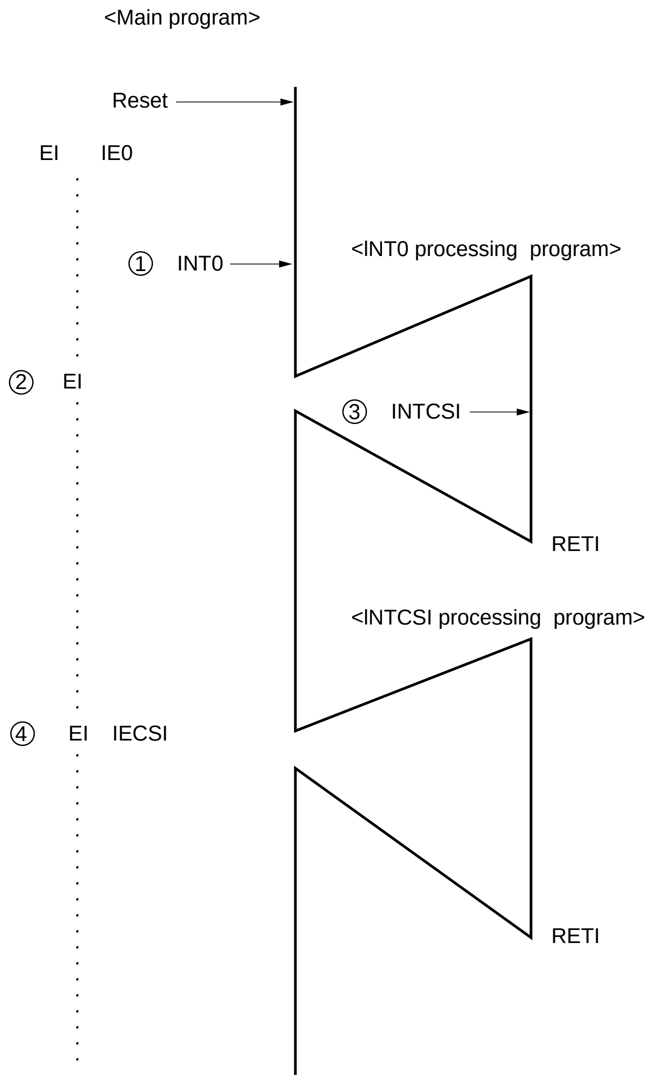
② Set INT0 to falling edge active.

③ Interrupt enable by EI and EI IExxx instructions.

④ Start the INT0 interrupt servicing program by INT0 falling. The status is changed to 1 and all the interrupts are disabled.

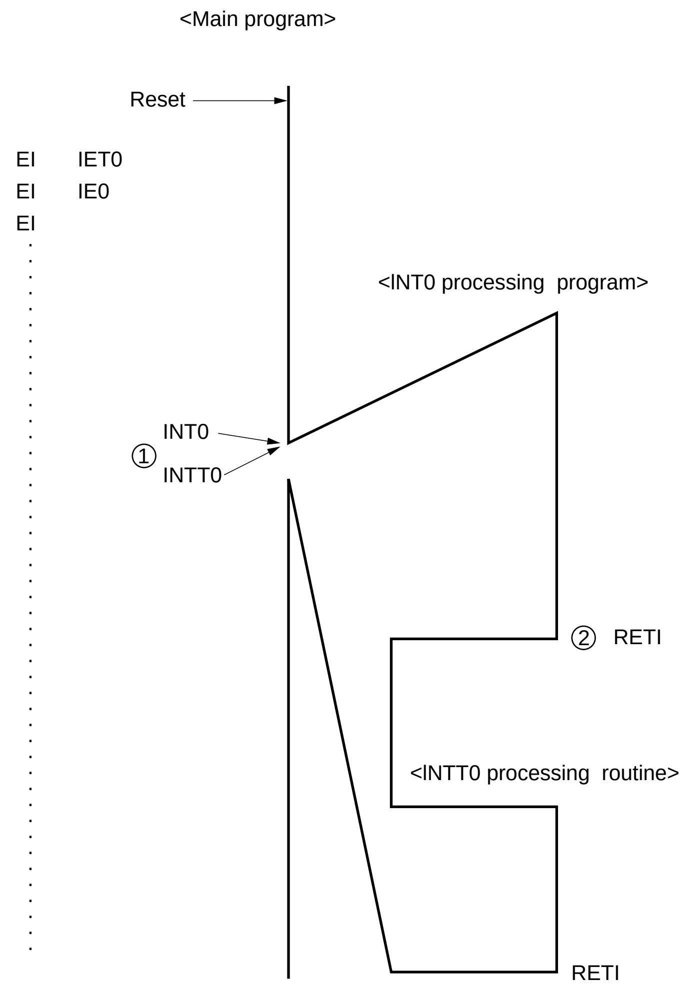
⑤ Restored from the interrupt by the RETI instruction. The status is returned to 0 and the interrupt is enabled.

(3) Execution of hold interrupt –Interrupt input during interrupt disabling–



① Even if INT0 is set during disabling, the request flag is held.
② INT0 processing program starts at the moment of interrupt enabling by the EI instruction.
③ Same as ①.
④ INTCSI processing program starts at the moment INTCSI in hold is enabled.

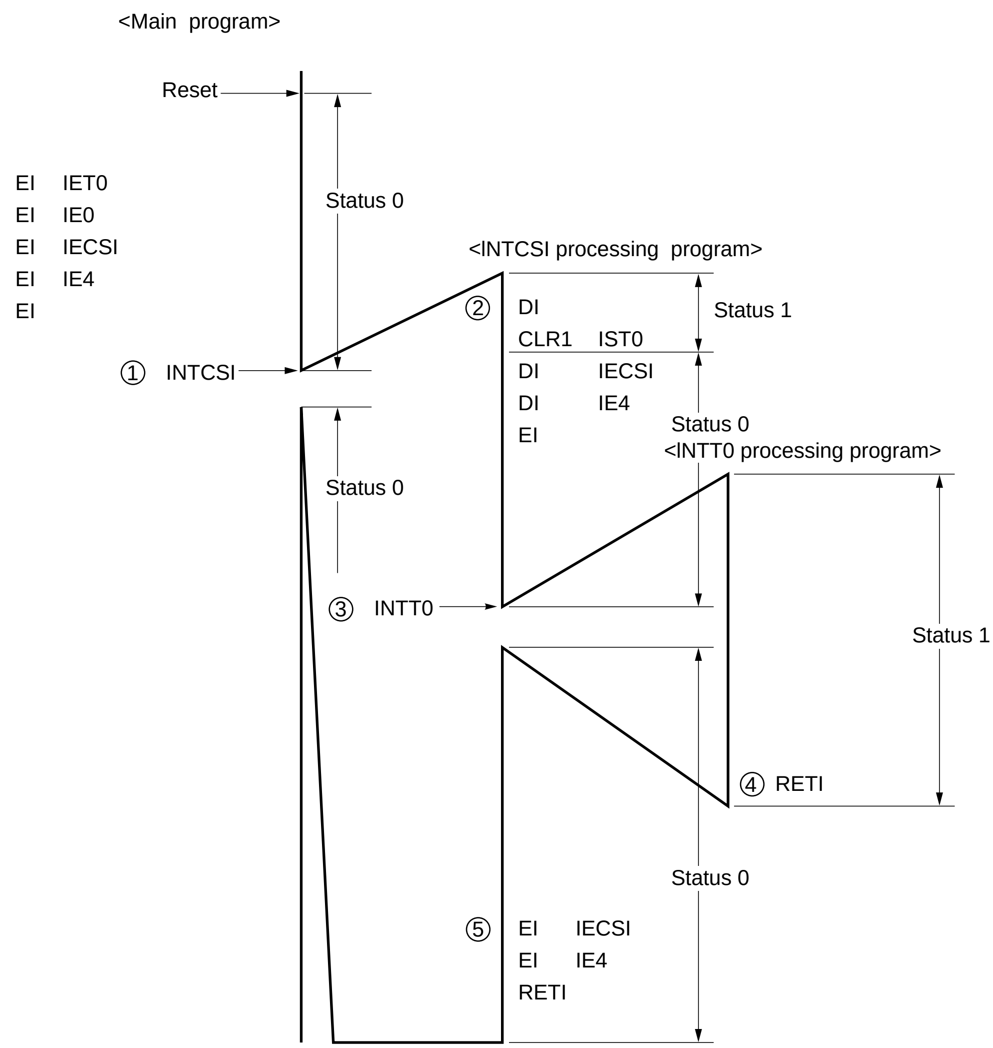
(4) Execution of interrupts in hold



① When INT0 and INTT0 occur simultaneously (during execution of the same instruction). INT0 earlier in the interrupt order is executed earlier (INTT0 is held).

② Using the RETI instruction, start the INTT0 processing program in hold if the INT0 processing program is ended.

(5) Enable two multiinterrupts – INTT0 and INT0 enable multiinterrupts. INTCS1 and INT4 enable single interrupt–



① Start the INTCSI processing program by the occurrence of the interrupt INTCSI which does not enable multiinterrupts. The status is 1.

② By clearing IST0, the status is made 0. Disable INTCSI and INT4 which do not enable multiinterrupts.

③ With the occurrence of INTT0 which enables multiinterrupts, the multiinterrupt is executed, the status is made 1 and all the interrupts are disabled.

④ The status is returned to 0 by ending the INTT0 processing.

⑤ Enable and restore INTCSI and INT4 that have been disabled.

CHAPTER 7 STANDBY FUNCTION

The μPD75308 has a standby function which allows the system power consumption to be reduced. The standby function comprises two modes:

- STOP mode
- HALT mode

The function of these modes is described below.

(1) STOP mode

This mode stops the main system clock oscillation circuit and halts the entire system. The power consumption of the CPU is thus considerably reduced.

Since low-voltage (V_{DD} = up to 2 V) data memory retention is also possible, this mode is useful for retaining data memory contents at an ultra-low consumption current.

As the μPD75308 STOP mode can be released by an interrupt request, intermittent operation is also possible. However, since a wait period is required to secure the oscillation stabilization time when the STOP mode is released, the HALT mode should be selected when it is necessary for servicing to start immediately in response to an interrupt request.

(2) HALT mode

In the HALT mode the CPU operation clock is stopped, but the system clock oscillation circuit continues to operate. Although this mode cannot reduce the consumption current as much as the STOP mode, it is effective when wanting to start servicing immediately in response to an interrupt request, or when wanting to perform intermittent operation such as clock operation.

In both modes, all register, flag and data memory contents immediately prior to setting the standby mode are retained. As the input/output port output latch and output buffer status are also retained, the input/output port status is processed beforehand to minimize the consumption current of the entire system.

Precautions concerning the use of these modes are given as follows.

Precautions:

1. **The STOP mode can be used only when the main system clock operating (the subsystem clock oscillation cannot be stopped). The HALT mode can be used in either the main system or subsystem clock operation status.**
2. **If the LCD controller/driver and watch timer's operation clock are at main system clock f_X and if it is set to the STOP mode, its operation is stopped. Therefore, when continuing the operation, it is necessary to switch the operation clock over to the subsystem clock f_{XT} before setting the STOP mode.**
3. **Changeover of the standby mode, the CPU clock and the system clock can achieve effective low-consumption current and low-voltage operations. However, as shown in 5.2.3 System clock and CPU clock setting, it takes time in either case from the selection of a new clock by operating the control register until the start of the operation with the newly-replaced clock. Therefore, when combining the clock changeover function and the standby mode, set to the standby mode after the time required for the changeover has elapsed.**
4. **When the standby mode is used, input/output ports should be handled so as to minimize the consumption current.**
   In particular, input ports should not be left open; either a low level or high level should be input without fail.

7.1 Standby Mode Setting and Operation Status

Table 7-1. Each Operation Status in Standby Mode

		STOP Mode	HALT Mode
Set instruction		STOP instruction	HALT instruction
System clock when setting		Settable only in main system clock	Settable either in main system clock or in subsystem clock
Operation status	Clock generation circuit	Stop only main system clock oscillation	Stop only CPU clock Φ (Oscillation continues)
	Basic interval timer	Stop operation	Operable only with main system clock oscillation (IRQBT is set at the standard time interval).
	Serial interface	Operable only when selecting external $\overline{\text{SCK}}$ input for serial clock	Operable with main system clock oscillation or when the external $\overline{\text{SCK}}$ input is selected as the serial clock.
	Timer/event counter	Operable only when TI0 pin input is specified for count clock	Operable with main system clock oscillation or when the TI0 pin input is specified for the count clock.
	Watch timer	Operable when f_{XT} is selected for the count clock	Operable
	LCD controller	Operable only when f_{XT} is selected for LCDCL	Operable
	External interrupt	INT1, 2 and 4 are operable. Only INT0 is inoperable.	
	CPU	Stop operation	
Cancellation signal		Interrupt request signal or $\overline{\text{RESET}}$ input from the operable hardware which is enabled by the interrupt enable flag	Interrupt request signal or $\overline{\text{RESET}}$ input from the operable hardware which is enabled by the interrupt enable flag

Use the STOP instruction to set the STOP mode, and the HALT instruction to set the HALT mode. (The STOP instruction and the HALT instruction respectively set PCC's bit 3 and bit 2.)

Make sure to write the NOP instruction after the STOP and HALT instructions.

When changing CPU's operation clock by PCC's low-order 2 bits, there is sometimes a time lapse from PCC rewriting to CPU clock change. Therefore, when changing the operation clock before the standby mode and the CPU clock after standby mode cancellation, set the standby mode after the machine cycles necessary for CPU clock change have elapsed since PCC rewriting.

In the standby mode, data is maintained in all the registers and the data memory, such as general registers, flags, mode registers, output latches which stop operations during standby mode.

Cautions 1. When set to the STOP mode, the X1 input is internally short circuited to V_{SS} (GND potential) to restrict the leakage of the crystal oscillation circuit unit.
Consequently, please do not use the STOP mode in the system which uses an external clock for the main system clock.

2. Before setting the standby mode, the interrupt request flag should be reset beforehand. When there is an interrupt source where both the interrupt request flag and interrupt enable flag are set, the standby mode is released as soon as it is entered (refer to Figure 6-1. Interrupt Control Circuit Block Diagram). However, if the STOP mode is set, the HALT mode is entered immediately after STOP instruction execution and the operating mode is restored after waiting for the BTM register set time.

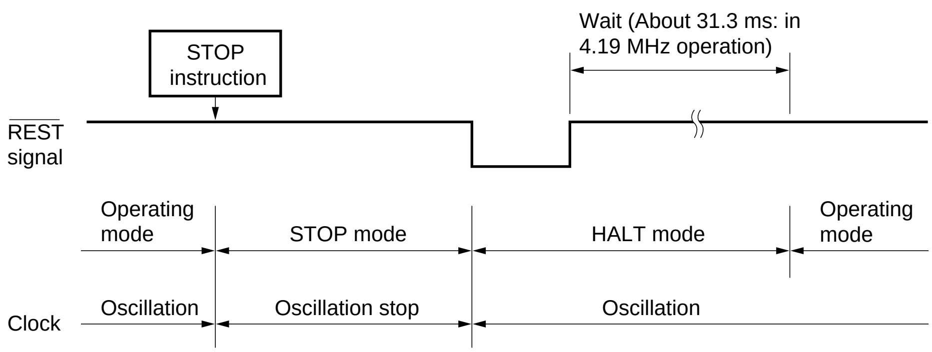
7.2 Cancellation of Standby Mode

Both the STOP mode and the HALT mode are canceled by the occurrence of the interrupt request signal[Note] enabled by the interrupt enable flag and the $\overline{\text{RESET}}$ input. The cancellation operation of each mode is shown in Figure 7-1.

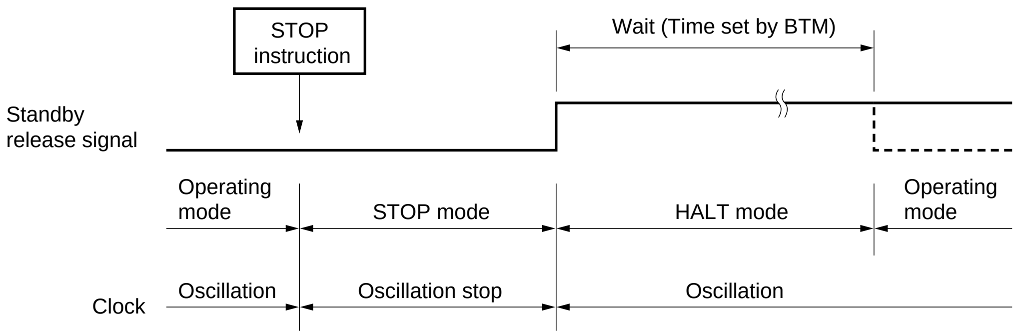
Note INT0 is not included.

Figure 7-1. Standby Mode Cancellation Operation (1/2)

(a) STOP mode cancellation by $\overline{\text{RESET}}$ input



(b) STOP mode cancellation by occurrence of interrupt



Remark The dotted line indicates the case in which the interrupt request which canceled the standby was received (IME = 1).

(c) HALT mode cancellation by $\overline{\text{RESET}}$ input

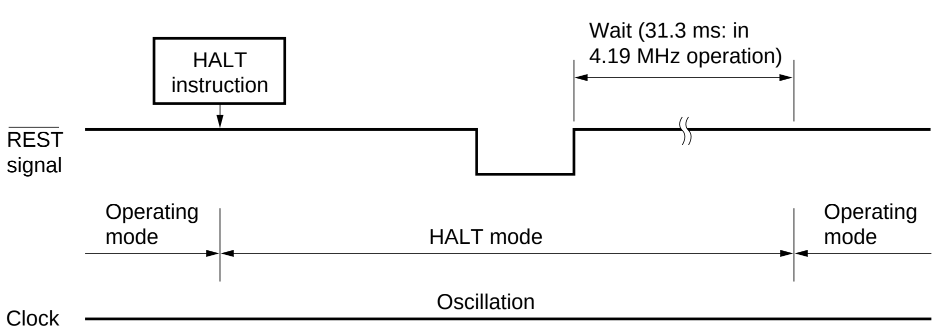
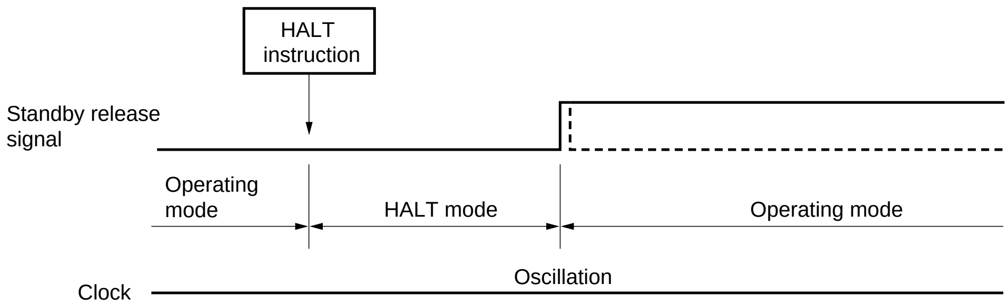


Figure 7-1. Standby Mode Cancellation Operation (2/2)

(d) HALT mode cancellation by occurrence of interrupt



Remark The dotted line indicates the case in which the interrupt request which canceled the standby was received (IME = 1).

When the STOP mode has been canceled by the occurrence of the interrupt, the wait time is determined by setting the BTM (refer to **Table 7-2**).

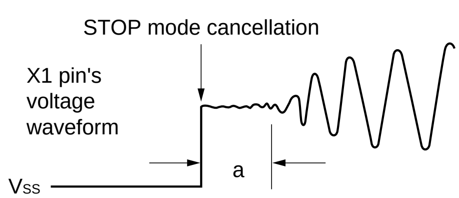
The time for settlement of the oscillation varies depending on the vibrator type used and the power voltage at the time of STOP mode cancellation. Therefore, select the WAIT time according to the use situation, and set the BTM before setting the STOP mode.

Table 7-2. Wait Time Selection by BTM

BTM3	BTM2	BTM1	BTM0	WAIT Time[Note] ( ) is when f_X = 4.19 MHz
–	0	0	0	About $2^{20}/f_X$ (about 250 ms)
–	0	1	1	About $2^{17}/f_X$ (about 31.3 ms)
–	1	0	1	About $2^{15}/f_X$ (about 7.82 ms)
–	1	1	1	About $2^{13}/f_X$ (about 1.95 ms)
Other than above				Setting prohibited

Note This time does not include the time from STOP mode cancellation to oscillation start.

Caution The wait time in STOP mode cancellation does not include the time (a in the diagram below) from STOP mode cancellation to clock oscillation start whether by $\overline{\text{RESET}}$ input or by occurrence of interrupt.



7.3 Operation After Standby Mode Cancellation

(1) Execute the normal reset operation if the standby mode was canceled by $\overline{\text{RESET}}$ input.

(2) When canceled by occurrence of the interrupt request, whether to perform the vectored interrupted or not when the CPU has restarted the instruction execution is determined by the contents of the interrupt enable flag (IME).

(a) When IME = 0

After standby mode cancellation, restart the execution from the instruction (NOP instruction) following the setting of the standby mode.

The interrupt request flag is maintained.

(b) When IME = 1

After standby mode cancellation, the vectored interrupt is executed after executing two instructions following the standby mode set instruction. However, if the standby mode was canceled by INTW and INT2 (testable input), operate the system in the same manner as in (a) because the vectored interrupt does not occur here.

7.4 Standby Mode Application

The standby mode is used according to the following procedures:

1. Detect the standby mode set factor such as power disconnection with the interrupt input or the port input (INT4 is effective in detecting power disconnection).
2. Process the input/output port (so that the consumption current is minimized).
   In particular, input ports should not be left open; either a low level or high level should be input without fail.
3. Specify the interrupt that cancels the standby mode. (INT4 is effective for this. Clear the interrupt enable flag not canceling the standby mode.)
4. Specify the operation after cancellation. (Operate the IME depending on whether to perform interrupt servicing or not.)
5. Specify the CPU clock after cancellation. (When changing the clock, let the machine cycles required until standby mode setting claps.)
6. Select the wait time in cancellation.
7. Set the standby mode (STOP, HALT instructions)

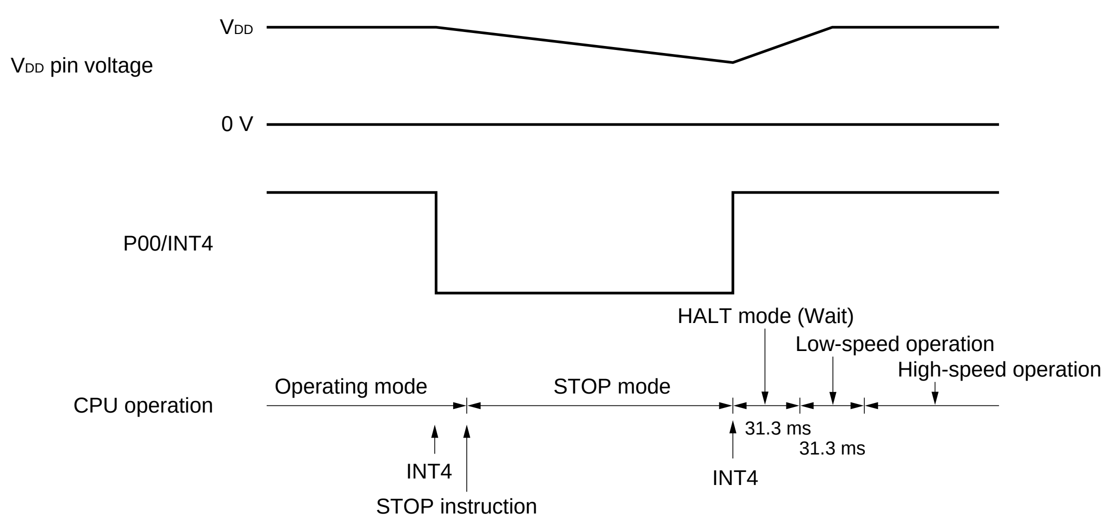
Furthermore, the standby mode can realize low-consumption current and low-voltage operation by combining with the system clock changeover function.

(1) Example of STOP mode application (f_X = 4.19 MHz operation)

<When using the STOP mode under the following conditions>

- Set the STOP mode by INT4's falling edge input and cancel it by the rising edge input. (INTBT is not used.)
- All the input/output ports shall be of high impedance (when pins are handled externally to reduce the consumption current in high impedance).
- Interrupts used in the program shall be INT0 and INTT0, provided that these are not used in STOP cancellation.
- Enable the interrupt even after cancellation.
- After cancellation, the system shall be started by the CPU clock at the lowest speed, which will then be changed over to high speed after 31.3 ms.
- The wait time in cancellation shall be about 31.3 ms.
- After cancellation, wait another 31.3 ms for power safety. And check the P00/INT4 pin twice and eliminate the chattering.

<Timing chart>



<Program example> (INT4 processing program, MBE = 0)

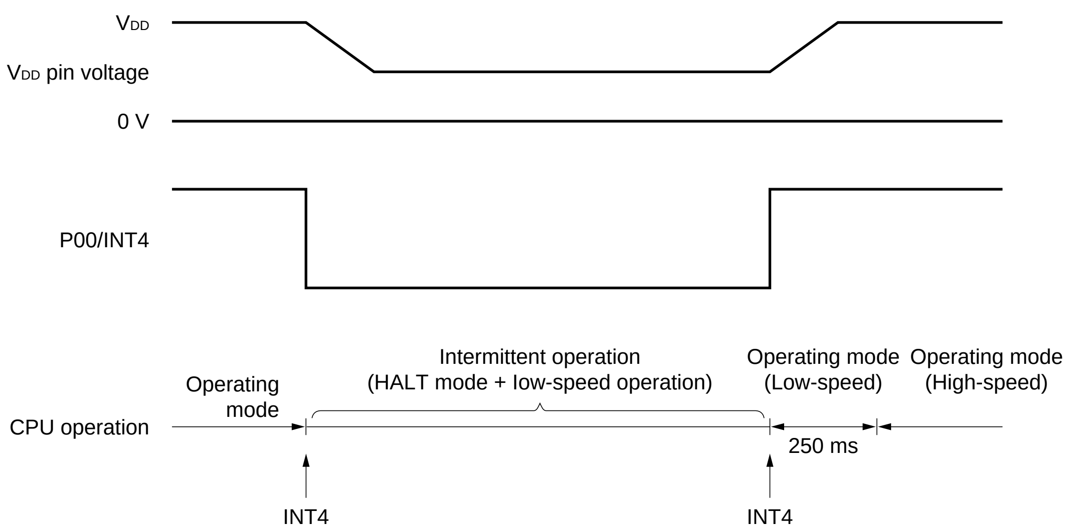
```
VSUB4:  SKT     PORT0.0     ; P00 = 1?
        BR      PDOWN       ; power down
        SET1    BTM.3       ; power on
WAIT:   SKT     IRQBT       ; 31.3 ms wait
        BR      WAIT
        SKT     PORT0.0     ; chattering check
        BR      PDOWN
        MOV     A, #0011B
        MOV     PCC, A      ; high-speed mode
       [MOV     XA, #xxH  ] ; port mode register set
       [MOV     PMGm, XA  ]
        EI      IE0
        EI      IET0
        RETI
PDOWN:  MOV     A, #0       ; lowest speed mode
        MOV     PCC, A
        MOV     XA, #00H
        MOV     LCDM, XA    ; LCD display off
        MOV     LCDC, A
        MOV     PMGA, XA    ; input/output port high impedance
        MOV     PMGB, XA
        DI      IE0         ; INT0, INTT0 disable
        DI      IET0
        MOV     A, #1011B
        MOV     BTM, A      ; wait time ≒ 31.3 ms
        NOP
        STOP                ; STOP mode set
        NOP
        RETI
```

(2) HALT mode application (f_X = 4.19 MHz, f_{XT} = 32.768 kHz operation)

<When operating intermittently under the following conditions>

- Changeover to the subsystem clock at INT4's falling.
- Stop main system clock's oscillation and set to the HALT mode.
- Operate intermittently at the interval of 0.5 sec during the standby mode.
- Changeover again to the main system clock at INT4's rising.
- INTBT shall not be used.

<Timing chart>



<Program example>

```
(Initialization)
            MOV     A, #0011B
            MOV     PCC, A        ; high-speed mode
            MOV     XA, #05H
            MOV     WM, XA        ; main system clock
            EI      IE4
            EI      IEW
            EI                    ; interrupt enable
(Main routine)
            SKT     PORT0.0       ; power OK?
            HALT                  ; power-down mode
            NOP                   ; power OK?
            SKTCLR  IRQW          ; is there 0.5 sec flag?
            BR      MAIN          ; NO
            CALL    WATCH         ; Watch subroutine
MAIN        :   :
                :
                :
```

```
(INT4 processing routine)
VINT4:   SKT     PORT0.0    ; power OK?, MBE = 0
         BR      PDOWN
         CLR1    SCC.3      ; main system clock oscillation start
         MOV     A, #1000B
         MOV     BTM, A
WAIT1:   SKT     IRQBT      ; 250 ms wait
         BR      WAIT1
         SKT     PORT0.0    ; chattering check
         BR      PDOWN
         CLR1    SCC.0      ; changeover to main system clock
         RETI
PDOWN:   MOV     XA, #00H
         MOV     LCDM, XA   ; LCD display off
         MOV     LCDC, A
         SET1    SCC.0      ; changeover to subsystem clock
         MOV     A, #5
WAIT2:   INCS    A          ; 32 or more machine cycles wait^Note (35 machine cycles)
         BR      WAIT2
         SET1    SCC.3      ; main system clock oscillation stop
         RETI
```

Note Refer to **5.2.3 System clock and CPU clock setting** for system clock/CPU clock switchover.

Caution Before changing the system clock from the main system clock to the subsystem clock, wait until the subsystem clock's oscillation settles.

CHAPTER 8 RESET FUNCTION

The μPD75308 is reset by the $\overline{\text{RESET}}$ input, and each hardware is initialized as shown in Table 8-1. Reset operation timing is shown in Figure 8-1.

Figure 8-1. Reset Operation by $\overline{\text{RESET}}$ Input

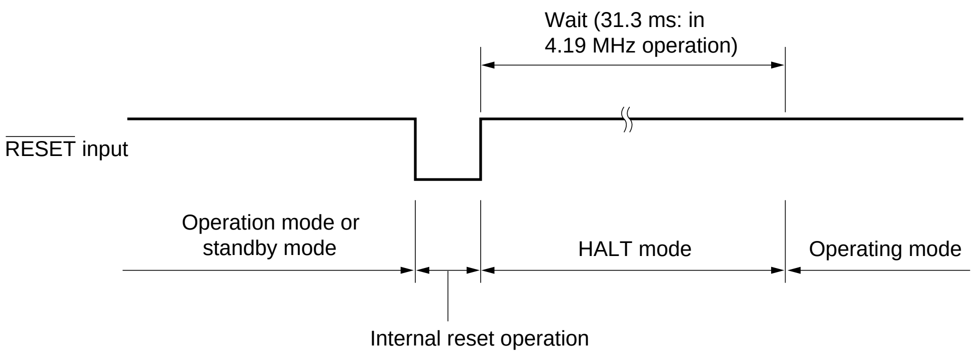


Table 8-1. Each Hardware Status After Resetting (1/2)

Hardware		$\overline{\text{RESET}}$ Input during Standby Mode	$\overline{\text{RESET}}$ Input during Operation
Program counter (PC)	μPD75304, 75304B	Set the low-order 4 bits of the 0000H address of the program memory to PC11 to PC8, and the contents of the 0001H address to PC7 to PC0.	Same as left
	μPD75306, 75306B, 75308, 75308B, 75P308	Set the low-order 5 bits of the 0000H address of the program memory to PC12 to PC8, and the contents of the 0001H address to PC7 to PC0.	Same as left
	μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B	Set the low-order 6 bits of the 0000H address of the program memory to PC13 to PC8, and the contents of the 0001H address to PC7 to PC0.	Same as left
PSW	Carrying flag (CY)	Retained	Undefined
	Skip flag (SK0 to SK2)	0	0
	Interrupt status flag (IST0)	0	0
	Bank enable flag (MBE)	Set bit 7 of the 0000H address of the program memory to the MBE.	Same as left
Stack pointer (SP)		Undefined	Undefined
Data memory (RAM)		Retained[Note]	Undefined
General register (X, A, H, L, D, E, B, C)		Retained	Undefined
Bank selection register (MBS)		0	0

Note The data of the 0F8H to 0FDH address of the data memory is made indefinite by $\overline{\text{RESET}}$ input.

Table 8-1. Each Hardware Status After Resetting (2/2)

Hardware		$\overline{\text{RESET}}$ Input during Stand-By Mode	$\overline{\text{RESET}}$ Input during Operation
Basic interval timer	Counter (BT)	Undefined	Undefined
	Mode register (BTM)	0	0
Timer/event counter	Counter (T0)	0	0
	Module register (TMOD0)	FFH	FFH
	Mode register (TM0)	0	0
	TOE0, TOUT F/F	0, 0	0, 0
Watch 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 and clock output circuit	Processor clock control register (PCC)	0	0
	System clock control register (SCC)	0	0
	Clock output mode register (CLOM)	0	0
LCD controller	Display mode register (LCDM)	0	0
	Display control register (LCDC)	0	0
Interrupt function	Interrupt request flag (IRQxxx)	Reset (0)	Reset (0)
	Interrupt enable flag (IExxx)	0	0
	Interrupt master enable flag (IME)	0	0
	INT0, 1 and 2 mode register (IM0, 1, 2)	0, 0, 0	0, 0, 0
Digital port	Output buffer	Off	Off
	Output latch	Clear (0)	Clear (0)
	I/O mode register (PMGA, B)	0	0
	Pull-up resistor specification register (POGA)	0	0
Bit sequential buffer (BSB0 to BSB3)		Retained	Undefined
Pin status	P00 to P03, P10 to P13, P20 to P23, P30 to P33, P60 to P63, P70 to P73	Input	Same as left
	P40 to P43, P50 to P53	• In integrated pull-up resistor: High level • In open drain: High impedance	Same as left
	S0 to S31, COM0 to COM3	**Note**	Same as left
	BIAS	• In integrated split resistor: Low level • When split resistor is not integrated: High impedance	Same as left

Note Each display output selects the following V_{LCX} as the input source.

S0 to S31 : V_{LC1}

COM0 to COM2 : V_{LC2}

COM3 : V_{LC0}

However, each display output level changes depending on each display output and V_{LCX}'s external circuit.

CHAPTER 9 PROM WRITING AND VERIFICATION

The program memory integrated in the μPD75P308 and 75P316A is one-time PROM or EPROM. The program memory integrated in the μPD75P316 and 75P316B is one-time PROM. The memory capacity is as follows.

- μPD75P308 : 8064 words x 8 bits
- μPD75P316 : 16256 words x 8 bits
- μPD75P316A : 16256 words x 8 bits
- μPD75P316B : 16256 words x 8 bits

For writing in and verifying this PROM, pins as shown in Table 9-1 are used. There is no address input. Instead, the addresses are updated by inputting the clock from the X1 pin.

Table 9-1. Pin Functions

Pin Name	Functions
X1, X2	Inputs the address updating clock when writing in and verifying the PROM, and inputs in the X2 pin its reversed signal.
MD0 to MD3	Selects the operating mode when writing in and verifying PROM.
P40 to P43 (Low-order 4 bits) P50 to P53 (High-order 4 bits)	Inputs/outputs the 8-bit data when writing in and verifying PROM.
V_{DD}	Applies the power voltage. Applies 5 V ±5 % in normal operation.[Note] Applies +6 V when writing in and verifying PROM.
V_{PP}	Applies the voltage when writing in and verifying PROM (normally, V_{DD} potential).

Note On the μPD75P316A only, 2.7 V to 6.0 V is applied in normal operation.

Cautions 1. In the μPD75P308K and 75P316AK equipped with the erase window, seal it up with a light blocking cover film except when erasing the PROM contents.

2. The one-time versions μPD75P308GF, 75P316GF, 75P316AGF, 75P316BGC and 75P316BGK are not equipped with the erase window, and so the PROM contents cannot be erased by ultraviolet rays.

9.1 Operating Mode when Writing in and Verifying PROM

The μPD75P308, 75P316, 75P316A and 75P316B turn to the PROM write/verify mode when +6 V is applied to the V_{DD} pin and +12.5 V to the V_{PP} pin. This mode is turned to the following operating modes by input signals to the MD0 to MD3 pins.

Table 9-2. Operating Mode

Operating Mode Specification						Operating Mode
V_{PP}	V_{DD}	MD0	MD1	MD2	MD3	
+12.5 V	+6 V	H	L	H	L	Program memory address 0 clear
		L	H	H	H	Write mode
		L	L	H	H	Verify mode
		H	x	H	H	Program inhibit mode

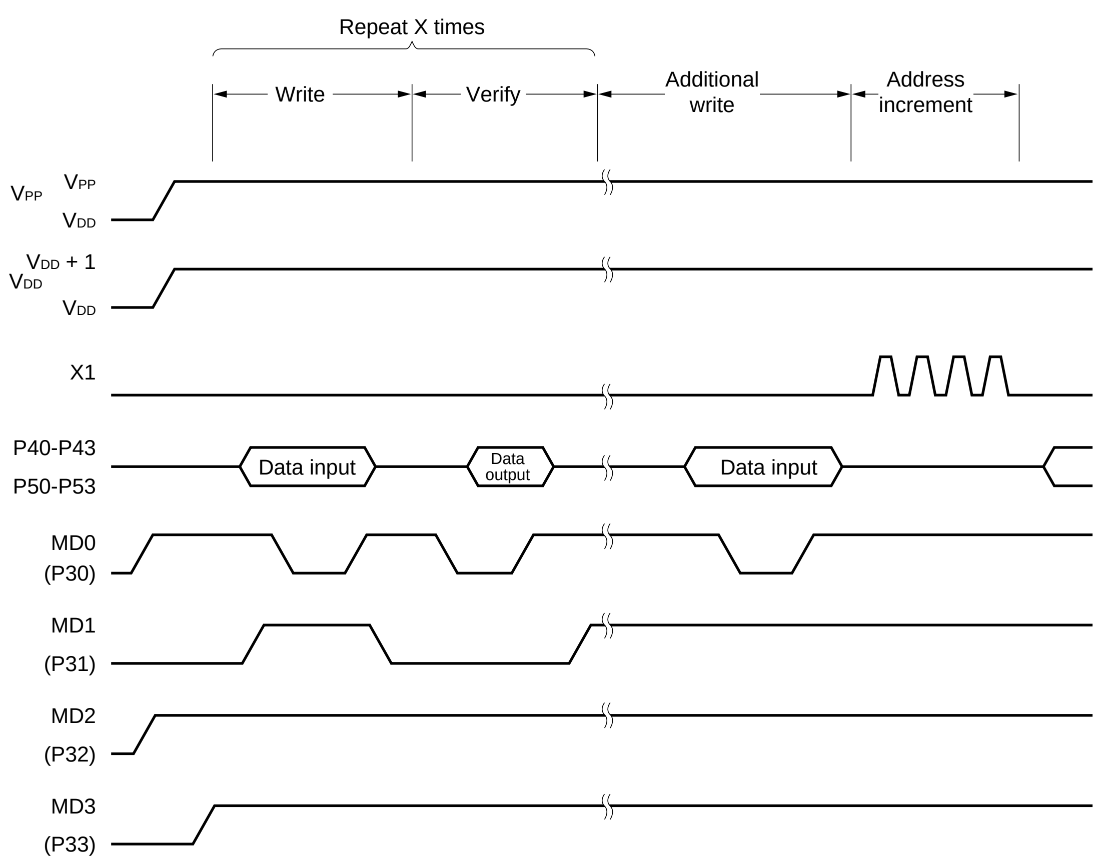
Remark x: L or H

9.2 PROM Write Procedure

The PROM write procedure is as follows. High speed writing is possible.

(1) Pull down the unused pin to V_{SS} through the resistance. The X1 pin is at low level.
(2) Supply 5 V to pins V_{DD} and V_{PP}.
(3) 10 μs wait
(4) 0 clear mode of the program memory address
(5) Supply +6 V to V_{DD}, and +12.5 V to V_{PP}.
(6) Program inhibit mode
(7) Write data in the 1 ms write mode.
(8) Program inhibit mode
(9) Verify mode. If written, move to (10). If not written repeat (7) to (9).
(10) Additional writing of (write time in (7) to (9): X) x 1 ms
(11) Program inhibit mode
(12) By inputting 4 pulses to the X1 pin, update the program memory address (+1).
(13) Repeat (7) to (12) to the last address.
(14) 0 clear mode of the program memory address
(15) Change the voltage of pins V_{DD} and V_{PP} to 5 V.
(16) Power off

Procedures (2) to (12) are shown in the diagram below.

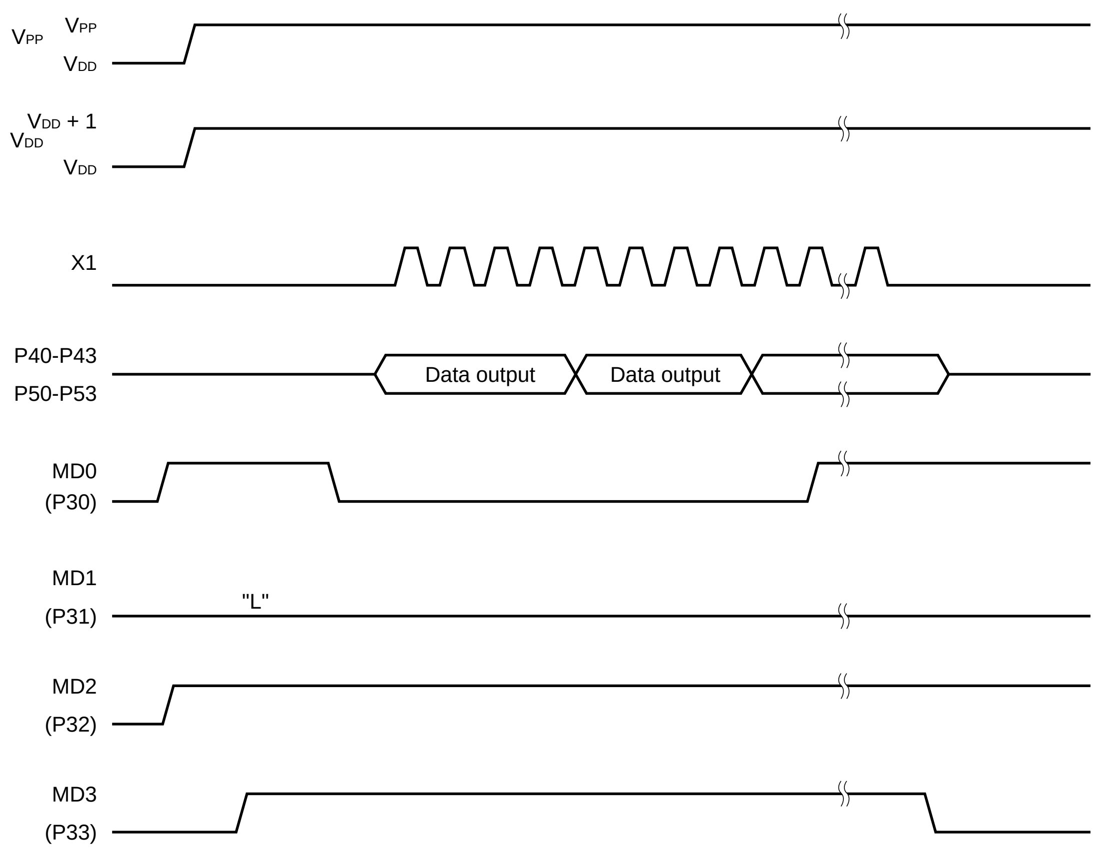


9.3 PROM Read Procedure

The PROM read procedure is as follows. Reads are performed in the verify mode.

(1) Pull down the unused pin to V_{SS} through the resistance. The X1 pin is at low level.
(2) Supply 5 V to pins V_{DD} to V_{PP}.
(3) 10 μs wait
(4) 0 clear mode of the program memory address
(5) Supply +6 V to V_{DD}, and +12.5 V to V_{PP}.
(6) Program inhibit mode
(7) If the clock pulse is input to the verify mode X1 pin, the data of 1 address is output in sequence at the cycle of four pulses.
(8) Program inhibit mode
(9) 0 clear mode of the program memory address
(10) Change the voltage of pins V_{DD} and V_{PP} to 5 V.
(11) Power off

Procedures (2) to (9) are shown in the diagram below.



9.4 Erasure

The data programmed into the μPD75P308K and 75P316AK can be erased by irradiation with ultraviolet rays through the window in the top.

Erasure is possible using ultraviolet rays with a wavelength of approximately 250 nm. The total amount of irradiation required for total erasure is 15 W•s/cm^2 (ultraviolet ray intensity x erasure time).

When using a commercially available ultraviolet lamp (254 nm wavelength, 12 mW/cm^2 intensity), erasure can be performed in approximately 15 to 20 minutes.

Cautions 1. Memory contents may also be erased by prolonged exposure to direct sunlight or light from a fluorescent lamp. To protect the memory contents, the window should be masked with a light-screening cover film. The light-screening cover film which NEC provides with its UV EPROM products should be used for this purpose.

2. During erasure, the distance between the ultraviolet lamp and the μPD75P308K/75P316AK should be no more than 2.5 cm.

Remark A longer exposure time may be necessary due to deterioration of the ultraviolet lamp or dirt etc. on the package window.

[MEMO]

CHAPTER 10 INSTRUCTION SET

The μPD75308 instruction set is a version revised and improved from the instruction set of the μPD7500 series, which is a forerunner of the 75X series. The new rennovative instruction set, still maintaining the continuity of the μPD7500 series, has the following characteristics.

(1) Bit manipulation instruction applicable in various ways
(2) Effective 4-bit manipulation instruction
(3) 8-bit data transfer instruction
(4) GETI instruction to shorten the program size
(5) Accumulation instruction and notation adjust instruction that improve program effects
(6) Table reference instruction suitable to continuous reference
(7) 1-byte relative branch instruction
(8) NEC standard mnemonics arranged for easy understanding

For addressing mode applicable in operating the data memory, refer to **CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP**.

10.1 Characteristic Instructions

Characteristic instructions in the μPD75308 instruction set will be overviewed below.

10.1.1 GETI instruction

The GETI instruction converts into 1-byte instructions the following.

(a) All-space subroutine call instruction
(b) All-space branch instruction
(c) Arbitrary 2-byte, 2 machine cycle instruction (However, the BRCB and CALLF instructions are excluded.)
(d) Combination of two 1-byte instructions

In the GETI instruction, refer to the table addressed by 0020H to 007FH in the program memory and execute the referred 2-byte data as the instruction for (a) to (d). Therefore, as many as 48 instructions for (a) to (d) can be converted to 1-byte instructions.

If the frequently used (a) to (d) instructions are converted to 1-byte by the GETI instruction, the number of program bytes can be shortened substantially.

10.1.2 Bit manipulation instruction

The μPD75308's bit manipulation can be performed by various instructions as shown below.

(a)	Bit set	: SET1	mem. bit
		SET1	mem. bit*
(b)	Bit clear	: CLR1	mem. bit
		CLR1	mem. bit*
(c)	Bit test	: SKT	mem. bit
		SKT	mem. bit*
(d)	Bit test	: SKF	mem. bit
		SKF	mem. bit*
(e)	Bit test and clear	: SKTCLR	mem. bit*
(f)	Boolean Arithmetic operation	: AND1	CY, mem. bit*
		OR1	CY, mem. bit*
		XOR1	CY, mem. bit*

Remark mem. bit* is the bit address indicated by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).

Especially, all the bit manipulation instructions described above can be applied to the input/output port, thus making its operation extremely effective.

10.1.3 Accumulation instruction

The μPD75308 is prepared with the following two types of accumulation instructions.

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"Accumulation" indicates that each of these two instruction types is placed in a continuous address.

Example

A0	: MOV	A, #0
A1	: MOV	A, #1
XA7	: MOV	XA, #07

As the example above shows, in a series of accumulation instructions, if the address to be initially executed is A0, the following two instructions are replaced by the NOP instruction to be executed, and if the address to be initially executed is A1, the following one instruction is replaced by the same instruction. In other words, only the initially executed instruction is valid, and all the other instructions following this are processed as NOP instructions.

By using this accumulation instruction, constants can be effectively set to the accumulator (A register, register pair XA) and to the data pointer (register pair HL).

10.1.4 Notation adjust instruction

In some applications, the results of 4-bit data additions or subtractions (performed in binary numerals) are converted to decimal numerals, or it is necessary to adjust them to hexa numerals as in time.

Therefore, the μPD75308 instruction set is prepared with the notation adjust instruction to arbitrarily adjust the 4-bit data addition or subtraction result to a notation.

(a) Notation adjust in addition

Assume that the notation value desired to be adjusted is m.

Based on the combinations of

```
ADDS    A, #16–m
ADDC    A, @HL       ; A, CY ← A + (HL) + CY
ADDS    A, #m
```

The accumulator and the memory (HL) are added and the added result is adjusted in the m-notation. The overflow is left behind in the carry flag.

If carry occurs as the result of executing the ADDC A, @HL instruction, skip the ADDS A, #n4 instruction that follows. If carry does not occur, the ADDS A, #n4 instruction is executed. At this time, this instruction's skip function is disabled and so, even if carry occurs as the result of addition, the following instruction is not skipped. Thus, the program can be written after the ADDS A, #n4 instruction.

Example The accumulator and the memory are added decimally.

```
ADDS    A, #6
ADDC    A, @HL    ; A, CY ← A + (HL) + CY
ADDS    A, #10
    :
```

(b) Notation adjust in subtraction

Assume that the notation value desired to be adjusted is m. The memory (HL) is subtracted from the accumulator, based on the combinations of

```
SUBC    A, @HL
ADDS    A, #m
```

and the result is adjusted in the m-notation. The underflow is left behind in the carry flag.

If borrow does not turn up as the result of executing the SUBC A, @HL instruction, skip the following ADDS A, #n4 instruction. If borrow turns up, the ADDS A, #n4 instruction is executed. At this time, this instruction's skip function is disabled and so, even if carry turns up as the result of addition, the following instruction is not skipped. Thus, the program can be written after the ADDS A, #n4 instruction.

10.1.5 Skip instruction and the number of machine cycles required for skipping

The μPD75308's instruction set is made to judge conditions and then organize programs by means of skip.

Skip is the function of executing the following instruction by skipping one instruction when the SKIP instruction (instruction with skip conditions) is executed and skip conditions are satisfied.

When skip has occurred, the number of machine cycles required for skipping is as follows.

(a) If the instruction (instruction to be skipped) following the SKIP instruction is a 3-byte instruction (two types: BR !addr instruction and CALL !addr instruction): 2 machine cycles

(b) If the instruction is other than (a): 1 machine cycle

Phase-out/Discontinued

10.2 Instruction Set and Its Operation

(1) Operand expression form and description method

Describe the operand in each instruction's operand column according to the method of describing the instruction's operand expression form. (Refer to **RA75X Assembler Package User's Manual-Language (EEU-1364)** for details.) If there are multiple description methods, select one of them. English letters in capital and codes +, – are the key words, and write them as they are.

In the case of immediate data, describe the appropriate numeric value or label.

The abbreviations of various types of registers and flags written in Figure 3-5 can be described as labels on behalf of mem, fmem, pmem and bit, etc. (However, there are limits to the labels that fmem and pmem can describe. For details, refer to **Table 3-1. Addressing Modes Applicable for Peripheral Hardware Operations** and **Figure 3-5. μPD75308 I/O Map**.)

<table>
<tr><th>Expression Format</th><th colspan="2">Description Method</th></tr>
<tr><td>reg
reg1</td><td colspan="2">X, A, B, C, D, E, H, L
X, B, C, D, E, H, L</td></tr>
<tr><td>rp
rp1
rp2</td><td colspan="2">XA, BC, DE, HL
BC, DE, HL
BC, DE</td></tr>
<tr><td>rpa
rpal</td><td colspan="2">HL, DE, DL
DE, DL</td></tr>
<tr><td>n4
n8</td><td colspan="2">4-bit immediate data or label
8-bit immediate data or label</td></tr>
<tr><td>mem Note 1
bit</td><td colspan="2">8-bit immediate data or label
2-bit immediate data or label</td></tr>
<tr><td>fmem
pmem</td><td colspan="2">FB0H to FBFH, FF0H to FFFH immediate data or label
FC0H to FFFH immediate data or label</td></tr>
<tr><td rowspan="5">addr,
caddr</td><td>μPD75304, 75304B</td><td>000H to FFFH immediate data or label</td></tr>
<tr><td>μPD75306, 75306B</td><td>0000H to 177FH immediate data or label</td></tr>
<tr><td>μPD75308, 75308B, 75P308</td><td>0000H to 1F7FH immediate data or label</td></tr>
<tr><td>μPD75312, 75312B</td><td>0000H to 2F7FH immediate data or label</td></tr>
<tr><td>μPD75316, 75316B, 75P316, 75P316A, 75P316B</td><td>0000H to 3F7FH immediate data or label</td></tr>
<tr><td>faddr</td><td colspan="2">11-bit immediate data or label</td></tr>
<tr><td>taddr</td><td colspan="2">20H to 7FH immediate data (provided that bit 0 = 0) or label</td></tr>
<tr><td>PORTn
IExxx
MBn</td><td colspan="2">PORT0 to PORT7
IEBT, IECSI, IET0, IE0, IE1, IE2, IE4, IEW
MB0, MB1, MB15 Note 2</td></tr>
</table>

Notes 1. Only even-number addresses can be described in mem in 8-bit data processing.

2. On the μPD75312B, 75316B, 75P316A 75P316B only:
MB0, MB1, MB2, MB3 and MB15.

(2) Common examples in explaining operations

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)
DL	: Register pair (DL)
HL	: Register pair (HL)
PC	: Program counter
SP	: Stack pointer
CY	: Carry flag; bit accumulator
PSW	: Program status word
MBE	: Memory bank enable flag
PORTn	: Port n (n = 0 to 7)
IME	: Interrupt master enable flag
IExxx	: Interrupt enable flag
MBS	: Memory bank selection register
PCC	: Processor clock control register
•	: Address bit delimit
(xx)	: Contents addressed by xx
xxH	: Hexadecimal data

(3) Explanation of symbols in the addressing area column

*1	MB = MBE•MBS (MBS = 0, 1, 15)[Note]		Data memory addressing
*2	MB = 0		
*3	MBE = 0 : MB = 0 (00H-7FH) MB = 15 (80H-FFH) MBE = 1 : MB = MBS (MBS = 0, 1, 15)[Note]		
*4	MB = 15, fmem = FB0H-FBFH, FF0H-FFFH		
*5	MB = 15, pmem = FC0H-FFFH		
*6	μPD75304, 75304B	addr = 000H-FFFH	Program memory addressing
	μPD75306, 75306B	addr = 0000H-177FH	
	μPD75308, 75308B, 75P308	addr = 0000H-1F7FH	
	μPD75312, 75312B	addr = 0000H-2F7FH	
	μPD75316, 75316B, 75P316, 75P316A, 75P316B	addr = 0000H-3F7FH	
*7	addr = (Current PC) – 15 to (Current PC) – 1, (Current PC) + 2 to (Current PC) + 16		
*8	μPD75304, 75304B	caddr = 000H-FFFH	
	μPD75306, 75306B	caddr = 0000H-0FFFH (PC_{12} = 0) 1000H-177FH (PC_{12} = 1)	
	μPD75308, 75308B, 75P308	caddr = 0000H-0FFFH (PC_{12} = 0) 1000H-1F7FH (PC_{12} = 1)	
	μPD75312, 75312B	caddr = 0000H-0FFFH (PC_{13} = 0, PC_{12} = 0) 1000H-1FFFH (PC_{13} = 0, PC_{12} = 1) 2000H-2F7FH (PC_{13} = 1, PC_{12} = 0)	
	μPD75316, 75316B, 75P316, 75P316A, 75P316B	caddr = 0000H-0FFFH (PC_{13} = 0, PC_{12} = 0) 1000H-1FFFH (PC_{13} = 0, PC_{12} = 1) 2000H-2FFFH (PC_{13} = 1, PC_{12} = 0) 3000H-3F7FH (PC_{13} = 1, PC_{12} = 1)	
*9	faddr = 0000H-07FFH		
*10	faddr = 0020H-007FH		

Note On the μPD75312B, 75316B, 75P316A and 75P316B only, MBS = 0, 1, 2, 3 or 15.

Remarks
1. MB indicates the accessible memory bank.
2. In *2, MB = 0 regardless of MBE and MBS.
3. In *4 and *5, MB = 15 regardless of MBE and MBS.
4. Each of *6 to *10 indicates the addressable area.

(4) Explanation of machine cycle columns

"S" indicates the number of machine cycles required for the skip-attached instruction to do skipping. The value of the S varies as follows.

- When not skipping .... S = 0
- If the instruction to be skipped is a 1- or 2-byte instruction .... S = 1
- If the instruction to be skipped is a 3-byte instruction (BR !addr, CALL !addr instruction)......... S = 2

Caution The GETI instruction is skipped at 1 machine cycle.

1 machine cycle corresponds to CPU clock Φ's 1 cycle portion (= t_{CY}), and, by setting the PCC, three kinds of time be selected (refer to **5.2.2 (1) Processor clock control register (PCC)**).

(5) Explanation of representative products in operation column

The products shown in the operation column (μPD75304, 75308, 75316) represent the following products.

μPD75034	μPD75304, 75304B
μPD75308	μPD75306, 75306B, 75308, 75308B, 75P308
μPD75316	μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Transfer instructions	MOV	A, #n4	1	1	A ← n4		Accumulation A
		reg1, #n4	2	2	reg1 ← n4		
		XA, #n8	2	2	XA ← n8		Accumulation A
		HL, #n8	2	2	HL ← n8		Accumulation B
		rp2, #n8	2	2	rp2 ← n8		
		A, @HL	1	1	A ← (HL)	*1	
		A, @rpa1	1	1	A ← (rpa1)	*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		
		rp1, XA	2	2	rp1 ← XA		
	XCH	A, @HL	1	1	A ↔ (HL)	*1	
		A, @rpa1	1	1	A ↔ (rpa1)	*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		
Table reference instructions	MOVT	XA, @PCDE	1	3	• **μPD75304** XA ← (PC_{11-8} + DE)$_{ROM}$		
					• **μPD75308** XA ← (PC_{12-8} + DE)$_{ROM}$		
					• **μPD75316** XA ← (PC_{13-8} + DE)$_{ROM}$		
		XA, @PCXA	1	3	• **μPD75304** XA ← (PC_{11-8} + XA)$_{ROM}$		
					• **μPD75308** XA ← (PC_{12-8} + XA)$_{ROM}$		
					• **μPD75316** XA ← (PC_{13-8} + XA)$_{ROM}$		
Arithmetic operation instructions	ADDS	A, #n4	1	1+S	A ← A + n4		carry
		A, @HL	1	1+S	A ← A + (HL)	*1	carry
	ADDC	A, @HL	1	1	A, CY ← A + (HL) + CY	*1	
	SUBS	A, @HL	1	1+S	A ← A – (HL)	*1	borrow
	SUBC	A, @HL	1	1	A, CY ← A – (HL) – CY	*1	
	AND	A, #n4	2	2	A ← A ^ n4		
		A, @HL	1	1	A ← A ^ (HL)	*1	

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Arithmetic operation instructions	OR	A, #n4	2	2	A ← A v n4		
		A, @HL	1	1	A ← A v (HL)	*1	
	XOR	A, #n4	2	2	A ← A $\veebar$ n4		
		A, @HL	1	1	A ← A $\veebar$ (HL)	*1	
Accumulator manipulation instructions	RORC	A	1	1	CY ← A_0, A_3 ← CY, A_{n-1} ← A_n		
	NOT	A	2	2	A ← $\overline{A}$		
Increment/ decrement instruction	INCS	reg	1	1+S	reg ← reg + 1		reg = 0
		@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
Comparison instructions	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)
		A, reg	2	2+S	Skip if A = reg		A = reg
Carry flag manipulation instructions	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 ← $\overline{CY}$		
Memory bit manipulation instructions	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	
		frmem. 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	(pmem. @L) = 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 v (fmem. bit)	* 4	
		CY, pmem. @L	2	2	CY ← CY v ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0}))	* 5	
		CY, @H + mem. bit	2	2	CY ← CY v (H + mem_{3-0}. bit)	* 1	

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Memory bit manipulation instructions	XOR1	CY, fmem. bit	2	2	$CY \leftarrow CY \forall$ (fmem. bit)	*4	
		CY, pmem. @L	2	2	$CY \leftarrow CY \forall (pmem_{7-2} + L_{3-2}.\ bit(L_{1-0}))$	*5	
		CY, @H + mem. bit	2	2	$CY \leftarrow CY \forall (H + mem_{3-0}.\ bit)$	*1	
Branch instructions	BR	addr	–	–	• **µPD75304** $PC_{11-0} \leftarrow addr$ (Using the assembler, select a more adequate instruction from between BRCB !caddr and BR $addr.)	*6	
					• **µPD75308** $PC_{12-0} \leftarrow addr$ (Using the assembler, select the most adequate instruction from among BR !addr, BRCB !caddr and BR $addr.)		
					• **µPD75316** $PC_{13-0} \leftarrow addr$ (Using the assembler, select the most adequate instruction from among BR !addr, BRCB !caddr and BR $addr.)		
		!addr	3	3	• **µPD75308** $PC_{12-0} \leftarrow addr$	*6	
					• **µPD75316** $PC_{13-0} \leftarrow addr$		
		$addr	1	2	• **µPD75304** $PC_{11-0} \leftarrow addr$	*7	
					• **µPD75308** $PC_{12-0} \leftarrow addr$		
					• **µPD75316** $PC_{13-0} \leftarrow addr$		
	BRCB	!caddr	2	2	• **µPD75304** $PC_{11-0} \leftarrow caddr_{11-0}$	*8	
					• **µPD75308** $PC_{12-0} \leftarrow PC_{12} + caddr_{11-0}$		
					• **µPD75316** $PC_{13-0} \leftarrow PC_{13}, PC_{12} + caddr_{11-0}$		
Subroutine stack control instructions	CALL	!addr	3	3	• **µPD75304** $(SP-4)\ (SP-1)\ (SP-2) \leftarrow PC_{11-0}$ $(SP-3) \leftarrow MBE, 0, 0, 0$ $PC_{11-0} \leftarrow addr, SP \leftarrow SP-4$	*6	
					• **µPD75308** $(SP-4)\ (SP-1)\ (SP-2) \leftarrow PC_{11-0}$ $(SP-3) \leftarrow MBE, 0, 0, PC_{12}$ $PC_{12-0} \leftarrow addr, SP \leftarrow SP-4$		
					• **µPD75316** $(SP-4)\ (SP-1)\ (SP-2) \leftarrow PC_{11-0}$ $(SP-3) \leftarrow MBE, 0, PC_{13}, PC_{12}$ $PC_{13-0} \leftarrow addr, SP \leftarrow SP-4$		

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Subroutine stack control instructions	CALLF	!faddr	2	2	• **μPD75304** (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, 0, 0, 0 SP ← SP–4	*9	
					• **μPD75308** (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, 0, 0, PC_{12} PC_{12-0} ← 00, faddr SP ← SP–4		
					• **μPD75316** (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, 0, PC_{13}, PC_{12} PC_{13-0} ← 000, faddr SP ← SP–4		
	RET		1	3	• **μPD75304** MBE, x, x, x ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PC_{11-0} ← 0, faddr SP ← SP+4		
					• **μPD75308** MBE, x, x, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4		
					• **μPD75316** MBE, x, PC_{13}, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4		
	RETS		1	3+S	• **μPD75304** MBE, x, x, x ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		Unconditional
					• **μPD75308** MBE, x, x, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		
					• **μPD75316** MBE, x, PC_{13}, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		
	RETI		1	3	• **μPD75304** MBE, x, x, x ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP + 6		
					• **μPD75308** MBE, x, x, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP + 6		
					• **μPD75316** MBE, x, PC_{13}, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP + 6		

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Subroutine stack control instructions	PUSH	rp	1	1	(SP–1) (SP–2) ← rp, SP ← SP–2		
		BS	2	2	(SP–1)←MBS,(SP–2)←0,SP←SP–2		
	POP	rp	1	1	rp ← (SP+1) (SP), SP ← SP+2		
		BS	2	2	MBS ← (SP+1), SP ← SP+2		
Interrupt control instructions	EI		2	2	IME ← 1		
		IExxx	2	2	IExxx ← 1		
	DI		2	2	IME ← 0		
		IExxx	2	2	IExxx ← 0		
Input/output instructions	IN Note 1	A, PORTn	2	2	A ← PORTn (n = 0–7)		
		XA, PORTn	2	2	XA ← PORTn+1, PORTn (n = 4, 6)		
	OUT Note 1	PORTn, A	2	2	PORTn ← A (n = 2–7)		
		PORTn, XA	2	2	PORTn+1, PORTn ← XA (n = 4, 6)		
CPU control instructions	HALT		2	2	Set HALT Mode (PCC. 2 ← 1)		
	STOP		2	2	Set STOP Mode (PCC. 3 ← 1)		
	NOP		1	1	No Operation		
Special instructions	SEL	MBn	2	2	MBS ← n (n = 0, 1, 15) Note 2		
	GETI	taddr	1	3	● ***μPD75304*** • **For table defined by TBR instruction** PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1)	*10	
					• **For table defined by TCALL instruction** (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, 0, 0, 0 PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1) SP ← SP–4		
					• **For table other than the above** (taddr) (taddr+1) instruction execution		Depends on the instruction referenced
					● ***μPD75308*** • **For table defined by TBR instruction** PC_{12-0} ← $(taddr)_{4-0}$ + (taddr+1)		
					• **For table defined by TCALL instruction** (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, 0, 0, PC_{12} PC_{12-0} ← $(taddr)_{4-0}$ + (taddr+1) SP ← SP–4		
					• **For table other than the above** (taddr) (taddr+1) instruction execution		Depends on the instruction referenced

Notes 1. When an IN/OUT instruction is executed, it is necessary to set MBE = 0, or MBE = 1 and MBS = 15.

2. For the μPD75312B, 75316B, 75P316A and 75P316B only, n = 0, 1, 2, 3 or 15.

Remark The TBR and TCALL instructions are assembler directives for GETI instruction table definition.

Instruction Group	Mnemonic	Operand	Number of Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Special instructions	GETI	taddr	1	3	● **μPD75316** • **For table defined by TBR instruction** $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr+1)$		
					• **For table defined by TCALL instruction** $(SP-4)\ (SP-1)\ (SP-2) \leftarrow PC_{11-0}$ $(SP-3) \leftarrow MBE, 0, PC_{13}, PC_{12}$ $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr+1)$ $SP \leftarrow SP-4$		
					• **For table other than the above** (taddr) (taddr+1) instruction execution		Depends on the instruction referenced

Note For the μPD75312B, 75316B, 75P316A and 75P316B only, n = 0, 1, 2, 3 or 15.

Remark The TBR and TCALL instructions are assembler directives for GETI instruction table definition.

10.3 Instruction Code of Each Instruction

(1) Explanation of instruction code symbol

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

Q_2	Q_1	Q_0	addressing
0	0	1	@HL
1	0	0	@DE
1	0	1	@DL

@rpa1 @rpa

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	IExxx
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	1	0	IE1

In : Immediate data to n4 and n8
Dn : Immediate data to mem
Bn : Immediate data to bit
Nn : Immediate data to n and IExxx
Tn : Immediate data to taddr x 1/2
An : Immediate data to [relative address distance (2 to 16) to branch destination address] – 1
Sn : Immediate data to the complement 1 of [relative address distance (15 to 1) to branch destination address]

(2) Instruction code of bit manipulation addressing

The *1 in the operand column indicates that there are the following three addressing types for this purpose:

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

The instruction code's second byte *2 corresponding to the addressing above is as follows:

*1	Instruction Code Second Byte								Accessible Bit
fmem. bit	1	0	B_1	B_0	F_3	F_2	F_1	F_0	Bit that can operate FB0H to FBFH
	1	1	B_1	B_0	F_3	F_2	F_1	F_0	Bit that can operate FF0H to FFFH
pmem. @L	0	1	0	0	G_3	G_2	G_1	G_0	Bit that can operate FC0H to FFFH
@H + mem. bit	0	0	B_1	B_0	D_3	D_2	D_1	D_0	Bit that can operate the accessible memory bank

Bn : Immediate data to bit

Fn : Immediate data to fmem (indicates low-order 4 bits of the address)

Gn : Immediate data to pmem (indicates bit 5 to 2 of the address)

Dn : Immediate data to mem (indicates low-order 4 bits of the address)

Istruction Group	Mnemonic	Operand	Instruction code		
			B_1	B_2	B_3
Transfer instruction	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 D_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 D_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 0	
		reg1, A	1 0 0 1 1 0 0 1	0 1 1 1 0 R_2 R_1 R_0	
		rp1, XA	1 0 1 0 1 0 1 0	0 1 0 1 0 P_2 P_1 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 D_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 0	
Table reference instruction	MOVT	XA, @PCDE	1 1 0 1 0 1 0 0		
		XA, @PCXA	1 1 0 1 0 0 0 0		
Arithmetic operation instructions	ADDS	A, #n4	0 1 1 0 I_3 I_2 I_1 I_0		
		A, @HL	1 1 0 1 0 0 1 0		
	ADDC	A, @HL	1 0 1 0 1 0 0 1		
	SUBS	A, @HL	1 0 1 0 1 0 0 0		
	SUBC	A, @HL	1 0 1 1 1 0 0 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		
	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		
	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		
Accumulator manipulation instructions	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	
Increment/ decrement instruction	INCS	reg	1 1 0 0 0 R_2 R_1 R_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		

Phase-out/Discontinued

Istruction Group	Mnemonic	Operand	Instruction code B1	Instruction code B2	Instruction code B3
Comparison instructions	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		
		A, reg	1 0 0 1 1 0 0 1	0 0 0 0 1 R_2 R_1 R_0	
Carry flag manipulation instruction	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 instructions	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	
		$\boxed{*1}$	1 0 0 1 1 1 0 1	$\boxed{*2}$	
	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	
		$\boxed{*1}$	1 0 0 1 1 1 0 0	$\boxed{*2}$	
	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	
		$\boxed{*1}$	1 0 1 1 1 1 1 1	$\boxed{*2}$	
	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	
		$\boxed{*1}$	1 0 1 1 1 1 1 0	$\boxed{*2}$	
	SKTCLR	$\boxed{*1}$	1 0 0 1 1 1 1 1	$\boxed{*2}$	
	AND1	CY, $\boxed{*1}$	1 0 1 0 1 1 0 0	$\boxed{*2}$	
	OR1	CY, $\boxed{*1}$	1 0 1 0 1 1 1 0	$\boxed{*2}$	
	XOR1	CY, $\boxed{*1}$	1 0 1 1 1 1 0 0	$\boxed{*2}$	
Branch instructions	BR	!addr	• **µPD75308** 1 0 1 0 1 0 1 1	0 0 0 ←	addr →
			• **µPD75316** 1 0 1 0 1 0 1 1	0 0 ←	addr →
		$addr	0 0 0 0 A_3 A_2 A_1 A_0 1 1 1 1 S_3 S_2 S_1 S_0		
	BRCB	!caddr	0 1 0 1 ←	caddr →	
Subroutine stack control instructions	CALL	!addr	• **µPD75304** 1 0 1 0 1 0 1 1	0 1 0 0 ←	addr →
			• **µPD75308** 1 0 1 0 1 0 1 1	0 1 0 ←	addr →
			• **µPD75316** 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	

Istruction Group	Mnemonic	Operand	Instruction code		
			B_1	B_2	B_3
Input/output instructions	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	
Interrupt control instructions	EI		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 0	
		IExxx	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	
		IExxx	1 0 0 1 1 1 0 0	1 0 N_5 1 1 N_2 N_1 N_0	
CPU control instructions	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 instructions	SEL	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		

10.4 Instruction Function and Application

10.4.1 Transfer instruction

MOV A, #n4

Function: A ← n4, n4 = $I_{3\text{-}0}$: 0H to FH

Transfers the 4-bit immediate data n4 to register A (4-bit accumulator). This has the accumulation effect (group A). If the MOV A, #n4 or MOV XA, #n8 instruction is entered after this, the accumulation instructions following the executed instruction are processed as NOP.

Application example:

① Set 0BH to the accumulator

MOV A, #0BH

② Select the data to be output to port 3 from among 0 to 2.

```
A0 : MOV A, #0
A1 : MOV A, #1
A2 : MOV A, #2
     OUT PORT3, A
```

MOV reg1, #n4

Function: reg1 ← n4, n4 = I_{3-0}: 0H to FH

Transfers the 4-bit immediate data n4 to register reg1 (X, H, L, D, E, B, C).

MOV rp, #n8

Function: rp ← n8, n8 = I_{7-0}: 00H to FFH

Transfers the 8-bit immediate data n8 to register pair rp (XA, HL, DE, BC).
When XA or HL is specified as rp, the accumulation effect is generated. The accumulation effect includes group A (MOV A, #n4 instruction and MOV XA, #n8 instruction) and group B (MOV HL, #n8 instruction). If instructions of the same group are entered in a row, accumulation instructions following the executed instruction are processed as NOP.

Application example:

Set 5FH to register pair DE.
MOV DE, #5FH

MOV A, @rpa

Function: A ← (rpa)

Transfers the contents of the data memory addressed by register pair rpa (HL, DE, DL) to register A.

MOV XA, @HL

Function: A ← (HL), X ← (HL + 1)

Transfers the contents of the data memory addressed by register pair HL to register A, and transfers the contents of the next address of the memory to register X.
However, if the contents of register L are in an odd number, the address whose lowest bit is ignored is specified.

Application example:
Transfer the data 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 register A to the data memory addressed by register pair HL.

MOV @HL, XA

Function: (HL) ← A, (HL + 1) ← X

Transfers the contents of register A to the data memory addressed by register pair HL, and transfers the contents of register X to the next address of the memory.
However, if the contents of register L is in an odd number, the address whose lowest bit is ignored is specified.

MOV A, mem

Function: A ← (mem), mem = D_{7-0}: 00H to FFH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to register A.

MOV XA, mem

Function: A ← (mem), X ← (mem + 1) mem = D_{7-0}: 00H to FFH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to register A, and transfers the contents of the next address to register X.
Addresses in an even number can be specified by mem.

Application example:
Transfer the data of addresses 40H and 41H to register pair XA.
MOV XA, 40H

MOV mem, A

Function: (mem) ← A, mem = D_{7-0}: 00H to FFH

Transfers the contents of register A to the data memory addressed by 8-bit immediate data mem.

MOV mem, XA

Function: (mem) ← A, (mem + 1) ← X mem = D_{7-0}: 00H to FFH

Transfers the contents of register A to the data memory addressed by 8-bit immediate data mem, and transfers the contents of register X to the next address of the memory.
Addresses in an even number can be specified by mem.

MOV A, reg

Function: A ← reg

Transfers the contents of register reg (X, A, H, L, D, E, B, C) to register A.

MOV XA, rp

Function: XA ← rp

Transfers the contents of register pair rp (XA, HL, DE, BC) to register pair XA.

MOV reg1, A

Function: reg1 ← A

Transfers the contents of register reg1 (X, H, L, D, E, B, C) to register A.

MOV rp1, XA

Function: rp1 ← XA

Transfers the contents of register pair XA to register pair rp1 (HL, DE, BC).

XCH XA, mem

Function: A ↔ (mem), X ↔ (mem + 1) mem = D_{7-0}: 00H to FFH

Exchanges the contents of register A with those of the data memory addressed by 8-bit immediate data mem, and exchanges the contents of register X with those of the next address of the memory. Addresses in an even number can be specified by mem.

XCH A, reg1

Function: A ↔ reg1

Exchanges the contents of register A with those of register reg1 (X, H, L, D, E, B, C).

XCH XA, rp

Fucntion: XA ↔ rp

Exchanges the contents of register pair XA with those of register pair rp (XA, HL, DE, BC).

XCH A, @rpa

Function: A ↔ (rpa)

Exchanges the contents of register A with those of the data memory addressed by register pair rpa (HL, DE, DL).

Application example:

Exchanges the data of addresses 20H to 2FH of the data memory with those of addresses 30H to 3FH.

```
        SEL     MB0
        MOV     D, #2
        MOV     HL, #30H
LOOP:   XCH     A, @HL      ; A ↔ (3x)
        XCH     A, @DL      ; A ↔ (2x)
        XCH     A, @HL      ; A ↔ (3x)
        INCS    L           ; L ← L + 1
        BR      LOOP
```

XCH XA, @HL

Function: A ↔ (HL), X ↔ (HL + 1)

Exchanges the contents of register A with those of the data memory addressed by register pair HL, and exchanges the contents of register X with those of the next address of the memory.
However, if the contents of register L are in an odd number, the address whose lowest bit is ignored is specified.

XCH A, mem

Function: A ↔ (mem), mem = D_{7-0}: 00H to FFH

Exchanges the contents of register A with those of the data memory addressed by 8-bit immediate data mem.

10.4.2 Table reference instruction

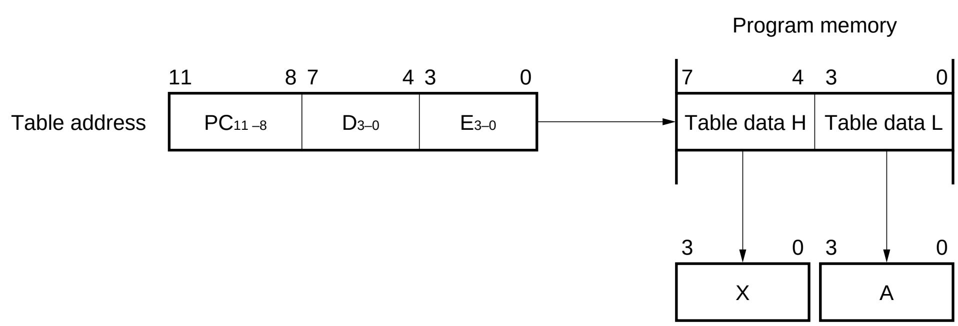
MOVT XA, @PCDE

Function:
- If μPD75304, 75304B
  XA ← ROM (PC_{11-8} + DE)
- If μPD75306, 75306B, 75308, 75308B, 75P308
  XA ← ROM (PC_{12-8} + DE)
- If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
  XA ← ROM (PC_{13-8} + DE)

Transfers the high-order bit of the program counter (PC) and the low-order 4 bits of the table data (ROM) in the program memory addressed by the contents of register pair DE to register A, and the high-order 4 bits to register X.
The high-order bit of the table address is determined by the contents of the program counter (PC) when executing this instruction.
In the table area, it is necessary to program necessary data beforehand with the assembler directive (DB instruction).
The program counter is not affected by the execution of this instruction.
This instruction is valid when referring to the table data more than once in a row.

Figure 10-1. Data Flow by Instruction Execution (1/2)

(a) If μPD75304, 75304B



(b) If μPD75306, 75306B, 75308, 75308B, 75P308

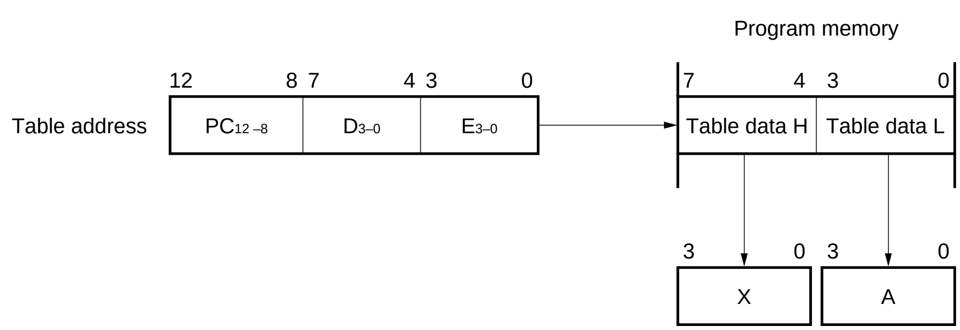
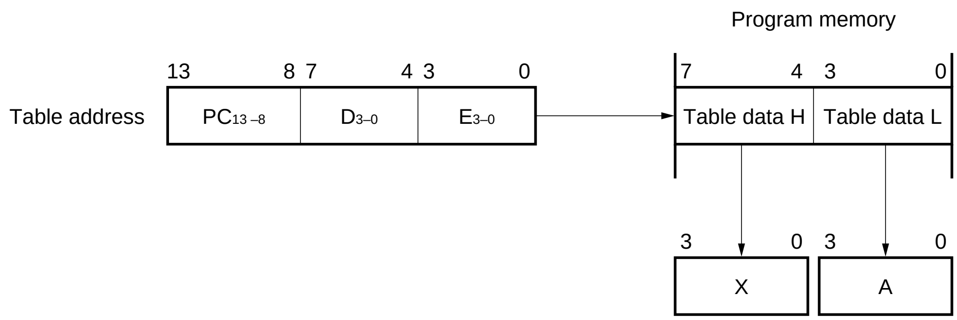


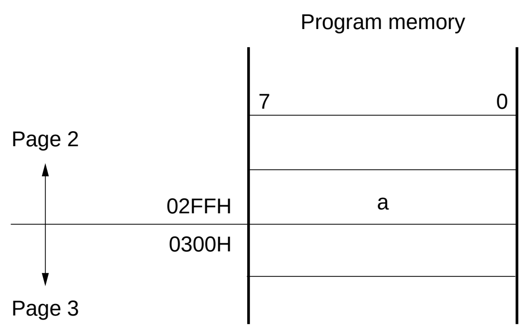
Figure 10-1. Data Flow by Instruction Execution (2/2)

(c) If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B



Precaution:

Normally, the MOVT XA, @PCDE instruction reference the table data of the page from which the instruction is located. However, if the instruction is located in address xxFFH, it does not reference the table data of this page but that of the next page.



For example, if the MOVT XA, @PCDE instruction is located in a as shown in the diagram above, the table data specified by the contents of the register pair DE not of page 2 but of page 3 is transferred to register pair XA.

Phase-out/Discontinued

Application example:

Transfers the 16-byte data of addresses xxF0H to xxFFH of the program memory to addresses 30H to 4FH the data memory.

```
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:
- If μPD75304, 75304B
  XA ← ROM (PC_{11-8} + XA)
- If μPD75306, 75306B, 75308, 75308B, 75P308
  XA ← ROM (PC_{12-8} + XA)
- If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
  XA ← ROM (PC_{13-8} + XA)

Transfers the high-order bit of the program counter (PC) and the low-order 4 bits of the table data in the program memory addressed by the contents of register pair XA to register A, and the high-order 4 bits to register X.

The high-order bit of the table address is determined by the contents of the program counter (PC) when executing this instruction.

In the table area, it is necessary to program necessary data beforehand with the assembler directive (DB instruction).

The program counter (PC) is not affected by the execution of this instruction.

Precaution:

As with MOVT XA, @PCDE, if the instruction is located in address xxFFH, the table data of the next page is transferred.

10.4.3 Arithmetic operation instruction

ADDS A, #n4

Function: A ← A + n4; Skip if carry. n4 = I_{3-0}: 0H to FH

Adds binary 4-bit immediate data n4 to the contents of register A. If carry turns up as the result of the addition, skip one instruction that follows. The carry flag is not affected in this case.
If instructions ADDC A, @HL and SUBC A, @HL are combined, they turn into the notation adjust instruction (refer to **10.1.4 Notation adjust instruction**).

ADDS A, @HL

Function: A ← A + (HL); Skip if carry.

Adds binarily the contents of the data memory addressed by register pair HL to the contents of register A. If carry occurs as the result of the addition, skip one instruction that follows. The carry flag is not affected in this case.

ADDC A, @HL

Function: A, CY ← A + (HL) + CY

Adds binarily the contents of the data memory addressed by register pair HL, including the carry flag, to the contents of register A. If carry turns up as the result of the addition, the carry flag is set. If carry does not occur, the carry flag is reset.
If the ADDS, #n4 instruction is entered after this instruction and if carry occurs in the latter instruction, the former instruction is skipped. If carry does not occur, the ADDS A, #n4 instruction is executed, and a function which disables the ADDS A, #n4 instruction's skip function is generated. Thus, these instructions can be combined to be utilized in the notation adjust (refer to **10.1.4 Notation adjust instruction**).

SUBC A, @HL

Function: A, CY ← A – (HL) – CY

Subtracts the contents of the data memory addressed by register pair HL, including the carry flag, from the contents of register A. If borrow occurs as the result, the carry flag is set. If borrow does not occur, the carry flag is reset.
If the ADDS, #n4 instruction is entered after this instruction and if borrow does not occur in the latter instruction, the former instruction is skipped. If borrow occurs, the ADDS A, #n4 instruction is executed, and a function which disables the ADDS A, #n4 instruction's skip function is generated. Thus, these instructions can be combined to be utilized in the notation adjust (refer to **10.1.4 Notation adjust instruction**).

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 register A. If borrow occurs as the result, skip one instruction that follows. The carry flag is not affected in this case.

AND A, #n4

Function: A ← A ^ n4, n4 = I_{3-0}: 0H to FH

Operates AND on the contents of register A and 4-bit immediate data n4. Sets the result to register A.

Application example:

Sets the high-order 2 bits of the accumulator to 0.

```
AND     A, #0011B
```

AND A, @HL

Function: A ← A ^ (HL)

Operates AND on the contents of register A and those of the data memory addressed by register pair HL. Sets the result to register A.

OR A, #n4

Function: A ← A v n4, n4 = I_{3-0}: 0H to FH

Operates OR on the contents of register A and 4-bit immediate data n4. Sets the result to register A.

Application example:

Sets the low-order 3 bits of the accumulator to 1.

```
OR      A, #0111B
```

OR A, @HL

Function: A ← A v (HL)

Operates OR on the contents of register A and those of the data memory addressed by register pair HL.

XOR A, #n4

Function: $A \leftarrow A \forall n4$, $n4 = I_{3-0}$: 0H to FH

Operates exclusive OR on the contents of register A and 4-bit immediate data n4. Sets the result to regiser A.

Application example:

Reverse the highest bit of the accumulator.

XOR A, #1000B

XOR A, @HL

Function: $A \leftarrow A \forall (HL)$

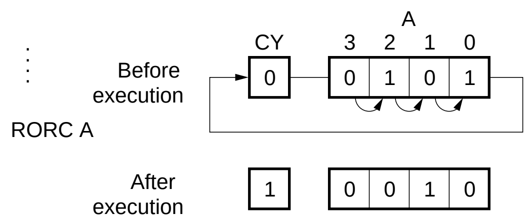
Operates exclusive OR on the contents of register A and those of the data memory addressed by register pair HL.

10.4.4 Accumulator manipulation instruction

RORC A

Function: $CY \leftarrow A_0$, $A_{n-1} \leftarrow A_n$, $A_3 \leftarrow CY$ (n = 1 - 3)

Rotates the contents of register A (4-bit accumulator), including the carry flag, to the right 1 bit after another.



NOT A

Function: $A \leftarrow \overline{A}$

Takes the complement of 1 (reverse each bit) of register A (4-bit accumulator).

10.4.5 Increment/decrement instruction

INCS reg

Function: reg $\leftarrow$ reg + 1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L, D, E, B, C). If reg = 0 as the result, the following one instruction is skipped.

INCS @HL

Function: (HL) $\leftarrow$ (HL) + 1; Skip if (HL) = 0

Increments the contents of the data memory addressed by register pair HL. If the contents of the data memory becomes 0 as the result, the following one instruction is skipped.

INCS mem

Function: (mem) $\leftarrow$ (mem) + 1; Skip if (mem) = 0, mem = D_{7-0}: 00H to FFH

Increments the contents of the data memory addressed by 8-bit immediate data mem. If the contents of the data memory becomes 0, the following one instruction is skipped.

DECS reg

Function: reg $\leftarrow$ reg – 1; Skip if reg = FH

Decrements the contents of register reg (X, A, H, L, D, E, B, C). If reg = FH as the result, the following one instruction is skipped.

10.4.6 Comparison instruction

SKE reg, #n4

Function: Skip if reg = n4, n4 = I_{3-0}: 0H to FH

Skips the following one instruction if the contents of register reg (X, A, H, L, D, E, B, C) are equal to 4-bit immediate data n4.

SKE @HL, #n4

Function: Skip if (HL) = n4, n4 = I_{3-0}: 0H to FH

Skips the following one instruction if the contents of the data memory addressed by register pair HL are equal to 4-bit immediate data n4.

SKE A, @HL

Function: Skip if A = (HL)

Skips the following one instruction if the contents of register A are equal to the contents of the data memory addressed by register pair HL.

SKE A, reg

Function: Skip if A = reg

Skips the following one instruction if the contents of register A are equal to the contents of register reg (X, A, H, L, D, E, B, C).

10.4.7 Carry flag manipulation instruction

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 following one instruction if the carry flag is 1.

NOT1 CY

Function: CY ← $\overline{\text{CY}}$

Reverses the carry flag; to 1 if 0, and to 0 if 1.

10.4.8 Memory bit manipulation instruction

SET1 mem. bit

Function: (mem. bit) ← 1, mem = D_{7-0}: 00H–FFH, bit = B_{1-0}: 0–3

Sets the bit which is specified by 2-bit immediate data bit of the address which is indicated by 8-bit immediate data mem.

SET1 fmem. bit
SET1 pmem. @L
SET1 @H + mem. bit

Function: (Bit specified by the operand) ← 1

Sets the bit of the data memory specified by bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).

CLR1 mem. bit

Function: (mem. bit) ← 0, mem = D_{7-0}: 00H–FFH, bit = B_{1-0}: 0–3

Clears the bit which is specified by 2-bit immediate data bit of the address which is indicated by 8-bit immediate data mem.

CLR1 fmem. bit
CLR1 pmem. @L
CLR1 @H + mem. bit

Function: (Bit specified by the operand) ← 0

Clears the bit of the data memory specified by bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit).

SKT mem. bit

Function: Skip if (mem. bit) = 1
mem = D_{7-0}: 00H–FFH, bit = B_{1-0}: 0–3

Skips the following one instruction if the bit specified by the 2-bit immediate data bit of the address indicated by 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 following one instruction if the bit of the data memory specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) is 1.

SKF mem. bit

Function: Skip if (mem. bit) = 0
mem = D_{7-0}: 00H–FFH, bit = B_{1-0}: 0–3

Skips the following one instruction if the bit specified by 2-bit immediate data bit of the address indicated by 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 following one instruction if the data memory bit specified by the 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

If the data memory bit specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit) is 1, the following one instruction is skipped and the bit is cleared to 0.

AND1 CY, fmem. bit
AND1 CY, pmem. @L
AND1 CY, @H + mem. bit

Function: CY ← CY ^ (bit specified by operand)

Operates AND on the contents of the carry flag and those of the data memory specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit), and then sets the result to the carry flag.

OR1 CY, fmem. bit
OR1 CY, pmem. @L
OR1 CY, @H + mem. bit

Function: CY $\leftarrow$ CY $\vee$ (bit specified by operand)

Operates OR on the contents of the carry flag and those of the bit of the data memory specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit), and then sets the result to the carry flag.

XOR1 CY, fmem. bit
XOR1 CY, pmem. @L
XOR1 CY, @H + mem. bit

Function: CY $\leftarrow$ CY $\veebar$ (bit specified by operand)

Operates exclusive OR on the contents of the carry flag and those of the bit of the data memory specified by the bit manipulation addressing (fmem. bit, pmem. @L, @H + mem. bit), and then sets the result to the carry flag.

10.4.9 Branch instruction

BR addr

Function: • If μPD75304, 75304B
$PC_{11-0} \leftarrow$ addr: addr = 000H to FFFH
• If μPD75306, 75306B, 75308, 75308B, 75P308
$PC_{12-0} \leftarrow$ addr: addr = 0000H to 177FH (μPD75306, 75306B)
addr = 0000H to 1F7FH (μPD75308, 75308B, 75P308)
• If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
$PC_{13-0} \leftarrow$ addr: addr = 0000H to 2F7FH (μPD75312, 75312B)
addr = 0000H to 3F7FH (μPD75316, 75316B, 75P316, 75P316A, 75P316B)

Branches to the address addressed by immediate data addr.
This instruction is the assembler's directive, and is replaced by the assembler during assembly by the most adequate instruction among instruction BR !addr, BRCB !caddr and BR $addr.
However, the μPD75304, 75304B does not have the BR !addr instruction.

BR !addr

Function: • If μPD75306, 75306B, 75308, 75308B, 75P308,
$PC_{12-0} \leftarrow$ addr: addr = 0000H to 1F7FH (μPD75306, 75306B)
addr = 0000H to 1F7FH (μPD75308, 75308B, 75P308)
• If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B,
$PC_{13-0} \leftarrow$ addr: addr = 0000H to 2F7FH (μPD75312, 75312B)
addr = 0000H to 3F7FH (μPD75316, 75316B, 75P316, 75P316A, 75P316B)

The immediate data addr is transferred to the program counter (PC) and then is branched to the address addressed by PC. Branching can be done to the entire space of the program memory.
However, the μPD75304, 75304B does not have the BR !addr instruction.

BR $addr

Function: PC ← addr, addr = (PC – 15) to (PC –1), (PC + 2) to (PC + 16)

This is the relative branch instruction with the branch range of (–15 to –1 and +2 to +16) from the current address. It is not affected by the page boundary or the block boundary.

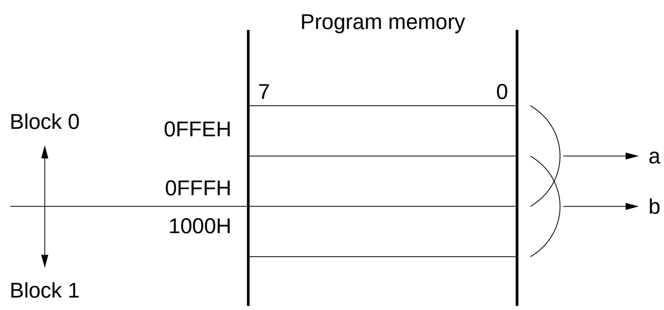
BRCB !caddr

Function: $PC_{11-0} \leftarrow$ caddr, caddr = A_{11-0}: 000H to FFFH

Branches to the address in which the program counter's low-order 12 bits (PC_{11-0}) are replaced by 12-bit immediate data caddr (A_{11-0}).
The μPD75304, 75304B's program counter consists of 11 bits, and so branching can be done by this instruction to the entire space.
The μPD75306, 75306B, 75308, 75308B, 75P308 cannot change PC_{12}, and so branching is done within the block.
The μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B cannot change PC_{13} and PC_{12}, and so branching is done within the block.

Precaution:

The BRCB !caddr instruction normally branches within the block in which this instruction exists. However, if the 1st byte is located in address 0FFEH or 0FFFH, it does not branch in block 0 but in block 1.



If the BRCB !caddr instruction is located in a or b in the diagram above, it does not branch in block 0 but in block 1.

TBR addr

Function:

This is the assembler directive to define the GETI instruction table. It is used when replacing the 3-byte BR instruction with the GETI instruction. The branch destination address of the 3-byte BR instruction is described in addr.
For details, refer to **RA75X Assembler Package User's Manual-Language (EEU-1364)**.

10.4.10 Subroutine stack control instruction

CALL !addr

Function: • If μPD75304, 75304B

$(SP-1) \leftarrow PC_{7-4}$, $(SP-2) \leftarrow PC_{3-0}$,
$(SP-3) \leftarrow$ MBE, 0, 0, 0, $(SP-4) \leftarrow PC_{11-8}$, $PC_{11-0} \leftarrow$ addr
$SP \leftarrow (SP-4)$
addr = A_{11-0}: 000H to FFFH

• If μPD75306, 75306B, 75308, 75308B, 75P308

$(SP-1) \leftarrow PC_{7-4}$, $(SP-2) \leftarrow PC_{3-0}$,
$(SP-3) \leftarrow$ MBE, 0, 0, PC_{12}, $(SP-4) \leftarrow PC_{12-8}$,
$PC_{12-0} \leftarrow$ addr, $SP \leftarrow (SP-4)$
addr = A_{12-0}: 0000H to 177FH (μPD75306, 75306B)
0000H to 1F7FH (μPD75308, 75308B, 75P308)

• If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B

$(SP-1) \leftarrow PC_{7-4}$, $(SP-2) \leftarrow PC_{3-0}$,
$(SP-3) \leftarrow$ MBE, 0, PC_{13}, PC_{12}, $(SP-4) \leftarrow PC_{13-8}$,
$PC_{13-0} \leftarrow$ addr, $SP \leftarrow (SP-4)$
addr = A_{13-0}: 0000H to 2F7FH (μPD75312, 75312B)
0000H to 3F7FH (μPD75316, 75316B, 75P316, 75P316A, 75P316B)

Branches to the address addressed by the immediate data addr after saving the contents of the program counter (PC: return address) and the MBE to the data memory (stack) addressed by the stack pointer (SP) and decrementing the stack pointer (SP).
Branching is done to the entire space of the program memory.

CALLF !faddr

Function: • If μPD75304, 75304B
$(SP - 1) \leftarrow PC_{7-4}$, $(SP - 2) \leftarrow PC_{3-0}$,
$(SP - 3) \leftarrow MBE, 0, 0, 0$, $(SP - 4) \leftarrow PC_{11-8}$,
$SP \leftarrow (SP - 4)$, $PC \leftarrow (0, A_{10-0})$
faddr = A_{10-0}: 000H to 7FFH

• If μPD75306, 75306B, 75308, 75308B, 75P308
$(SP - 1) \leftarrow PC_{7-4}$, $(SP - 2) \leftarrow PC_{3-0}$,
$(SP - 3) \leftarrow MBE, 0, 0, PC_{12}$, $(SP - 4) \leftarrow PC_{12-8}$,
$SP \leftarrow (SP - 4)$, $PC \leftarrow (00, A_{10-0})$
faddr = A_{10-0}: 000H to 7FFH

• If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
$(SP - 1) \leftarrow PC_{7-4}$, $(SP - 2) \leftarrow PC_{3-0}$,
$(SP - 3) \leftarrow MBE, 0, PC_{13}, PC_{12}$, $(SP - 4) \leftarrow PC_{13-8}$,
$SP \leftarrow (SP - 4)$, $PC \leftarrow (000, A_{10-0})$
faddr = A_{10-0}: 000H to 7FFH

Branches to the address addressed by the 11-bit immediate data faddr after saving the contents of the program counter (PC; return address) and the MBE to the data memory (stack) addressed by the stack pointer (SP) and decrementing the SP.
The range that can be called is limited to addresses 000H to 7FFH (0 to 2047).

TCALL addr

Function:

This is the assembler directive to define the GETI instruction table. It is used when replacing the 3-byte CALL instruction with the GETI instruction. The CALL address of the 3-byte CALL instruction is described in addr.
For details, refer to **RA75X Assembler Package User's Manual-Language (EEU-1364)**.

RET

Function:

- If μPD75304, 75304B
  $PC_{11-8} \leftarrow$ (SP), MBE, x, x, x, $\leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2),
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4)

- If μPD75306, 75306B, 75308, 75308B, 75P308
  $PC_{11-8} \leftarrow$ (SP), MBE, x, x, $PC_{12} \leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2)
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4)

- If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
  $PC_{11-8} \leftarrow$ (SP), MBE, x, PC_{13}, $PC_{12} \leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2),
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4)

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) in the program counter (PC) and the memory bank enable flag (MBE) and then increments the contents of the SP.

Precaution:

The program status word (PSW) is not restored other than in the MBE.

RETS

Function:

- If μPD75304, 75304B
  $PC_{11-8} \leftarrow$ (SP), MBE, x, x, x $\leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2),
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4), Then skip unconditionally

- If μPD75306, 75306B, 75308, 75308B, 75P308
  $PC_{11-8} \leftarrow$ (SP), MBE, x, x, $PC_{12} \leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2),
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4), Then skip unconditionally

- If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
  $PC_{11-8} \leftarrow$ (SP), MBE, x, PC_{13}, $PC_{12} \leftarrow$ (SP + 1), $PC_{3-0} \leftarrow$ (SP + 2),
  $PC_{7-4} \leftarrow$ (SP + 3), SP $\leftarrow$ (SP + 4), Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) in the program counter (PC) and the memory bank enable flag (MBE), increments the contents of the SP and then skips unconditionally.

Precaution:

The program status word (PSW) is not restored other than in the MBE.

RETI

Function: • If μPD75304, 75304B
$PC_{11-8} \leftarrow (SP)$, MBE, x, x, x $\leftarrow (SP + 1)$, $PC_{3-0} \leftarrow (SP + 2)$, $PC_{7-4} \leftarrow (SP + 3)$,
$PSW_L \leftarrow (SP + 4)$, $PSW_H \leftarrow (SP + 5)$, $SP \leftarrow (SP + 6)$

• If μPD75306, 75306B, 75308, 75308B, 75P308
$PC_{11-8} \leftarrow (SP)$, MBE, x, x, $PC_{12} \leftarrow (SP + 1)$, $PC_{3-0} \leftarrow (SP + 2)$
$PC_{7-4} \leftarrow (SP + 3)$, $PSW_L \leftarrow (SP + 4)$, $PSW_H \leftarrow (SP + 5)$
$SP \leftarrow (SP + 6)$

• If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B
$PC_{11-8} \leftarrow (SP)$, MBE, x, PC_{13}, $PC_{12} \leftarrow (SP + 1)$, $PC_{3-0} \leftarrow (SP + 2)$
$PC_{7-4} \leftarrow (SP + 3)$, $PSW_L \leftarrow (SP + 4)$, $PSW_H \leftarrow (SP + 5)$
$SP \leftarrow (SP + 6)$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) in the program counter (PC) and the program status word (PSW) and then increments the SP.
It is used when returning from interrupt servicing routine.

PUSH rp

Function: $(SP - 1) \leftarrow rp_H$, $(SP - 2) \leftarrow rp_L$, $SP \leftarrow SP - 2$

Saves the contents of the register pair rp (XA, HL, DE, BC) to the data memory (stack) addressed by the stack pointer (SP) and then decrements the SP.
The upper part (rp_H: X, H, D, B) of the register pair is saved to the stack addressed by (SP – 1), and the lower part (rp_L: A, L, E, C) to the stack addressed by (SP – 2).

PUSH BS

Function: $(SP - 1) \leftarrow MBS$, $(SP - 2) \leftarrow 0$, $SP \leftarrow SP - 2$

Saves the contents of the memory bank selection register (MBS) 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 $(SP + 1)$, $SP \leftarrow SP + 2$

Returns 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 in the lower part (rp_L: A, L, E, C) of the register pair, and the contents of (SP + 1) are restored in the upper part (rp_H: X, H, D, B).

POP BS

Function: MBS ← (SP + 1), SP ← SP + 2

Returns the contents of the data memory (stack) addressed by the stack pointer (SP) to the memory bank selection register (MBS) and then increments the SP.

10.4.11 Interrupt control instruction

EI

Function: IME ← 1

Sets (1) the interrupt mask enable flag and enables the interrupt. Whether the interrupt is received or not is controlled by each interrupt enable flag.

EI IExxx

Function: IExxx ← 1, xxx = N_5, N_{2-0}

Sets (1) the interrupt enable flag (IExxx) and puts the interrupt to reception status. (xxx = BT, CSI, T0, 0, 1, 2, 4, W)

DI

Function: IME ← 0

Resets (0) the interrupt mask enable flag and disables all the interrupts regardless of the contents of each interrupt enable flag.

DI IExxx

Function: IExxx ← 0, xxx = N_5, N_{2-0}

Resets (0) the interrupt enable flag (IExxx) and disables the reception of the interrupt. (xxx = BT, CSI, T0, 0, 1, 2, 4, W)

10.4.12 Input/output instruction

IN A, PORTn

Function: A ← PORTn, n = N_{3-0}: 0–7

Transfers the contents of the port specified by PORTn (n = 0 to 7) to register A.

Precaution:

When executing this instruction, it is necessary to set MBE = 0 or (MBE = 1 and MBS = 15). Only 0 to 7 can be specified to n.
By specifying the input/output mode, the output latch data (output mode) or the pin data (input mode) is incorporated.

IN XA, PORTn

Function: A ← PORTn, X ← PORTn + 1, n = N_{3-0}: 4, 6

Transfers the contents of the port specified by PORTn (n = 4, 6) to register A, and the contents of the next port to register X.

Precaution:

Only 4 or 6 can be specified as n. When executing this instruction, it is necessary to set MBE = 0 or (MBE = 1 and MBS = 15).
By specifying the input/output mode, the output latch data (output mode) or the pin data (input mode) is incorporated.

OUT PORTn, A

Function: PORTn ← A, n = N_{3-0}: 2–7

Transfers the contents of register A to the output latch of the port specified by PORTn (n = 2–7).

Precaution:

When executing this instruction, it is necessary to set MBE = 0 or (MBE = 1 and MBS = 15).
Only 2 to 7 can be specified to n.

OUT PORTn, XA

Function: PORTn ← A, PORTn + 1 ← X, n = N_{3-0}: 4, 6

Transfers the contents of register A to the output latch of the port specified by PORTn (n = 4, 6) and the contents of register X to the output latch of the next port.

Precaution:

When executing this instruction, it is necessary to set MBE = 0 or (MBE = 1 and MBS = 15).
Only 4 or 6 can be specified as n.

10.4.13 CPU control instruction

HALT

Function: PCC. 2 ← 1

Sets the HALT mode. (This instruction sets bit 2 of the processor clock control register.)

Precaution: Make one instruction following the HALT instruction a NOP instruction.

STOP

Function: PCC. 3 ← 1

Sets the STOP mode. (This instruction sets bit 3 of the processor clock control register.)

Precaution: Make one instruction following the STOP instruction a NOP instruction.

NOP

Function:

Uses up 1 machine cycle without doing anything.

10.4.14 Special instruction

SEL MBn

Function: MBS $\leftarrow$ n, n = N_{3-0}: 0, 1, 15[Note]

Transfers 4-bit immediate data n to the memory bank selection register (MBS). Only 0, 1 and 15 (on the μPD75312B, 75316B, 75P316A, 75P316B: 0, 1, 2, 3, and 15 can be specified as n.

Note On the μPD75312B, 75316B, 75P316A, 75P316B only, N = N_{3-0}: 0, 1, 2, 3, 15.

GETI taddr

Function: taddr = T_{5-0}, 0: 20H–7FH

(i) If TBR instruction

- **If μPD75304, 75304B**
  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr + 1)$

- **If μPD75306, 75306B, 75308, 75308B, 75P308**
  $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$

- **If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B**
  $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr + 1)$

(ii) If TCALL instruction

- **If μPD75304, 75304B**
  $(SP - 4)\ (SP - 1)\ (SP - 2) \leftarrow PC_{11-0}$
  $(SP - 3) \leftarrow$ MBE, 0, 0, 0
  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr + 1)$
  $SP \leftarrow (SP - 4)$

- **If μPD75306, 75306B, 75308, 75308B, 75P308**
  $(SP - 4)\ (SP - 1)\ (SP - 2) \leftarrow PC_{11-0}$
  $(SP - 3) \leftarrow$ MBE, 0, 0, PC_{12}
  $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$
  $SP \leftarrow (SP - 4)$

- **If μPD75312, 75312B, 75316, 75316B, 75P316, 75P316A, 75P316B**
  $(SP - 4)\ (SP - 1)\ (SP - 2) \leftarrow PC_{11-0}$
  $(SP - 3) \leftarrow$ MBE, 0, PC_{13}, PC_{12}
  $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr + 1)$
  $SP \leftarrow (SP - 4)$

(iii) If other than TBR or TCALL instructions, executes the instruction whose instruction code is (taddr) (taddr + 1).

Refers to the 2-byte data in the program memory address specified by (taddr), (taddr + 1) and executes it as the instruction.
The range of reference table is 0020H to 007FH, and data is written in beforehand. If the 1-byte or 2-byte instruction is used, data is written in mnemonic as it is. If the 3-byte branch instruction or 3-byte call instruction is used, data is written by the assembler directive (TBR, TCALL).
Addresses that can be specified as taddr are only in even numbers.

Precaution:

2-byte instructions that can be set in the reference table are limited to the 2-machine cycle instructions (BRCB and CALLF instruction excluded). When setting two 1-byte instructions, combinations are limited as shown below.

1st Byte Instruction	2nd Byte Instruction

MOV A, @HL
MOV @HL, A
XCA 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 PC cannot be incremented during GETI instruction execution, continues processing from the address following the GETI instruction after executing the reference instruction.
If the instruction preceding the GETI instruction has the skip function, the GETI instruction is skipped in the same manner as other 1-byte instructions. Also, if the instruction referenced by the GETI instruction has the skip function, the instruction immediately following the GETI instruction is skipped.
If an instruction with accumulation effect is referenced by the GETI instruction, it is executed as follows.

- If the instruction immediately preceding the GETI instruction has also the accumulation effect of the same group, the accumulation effect is removed if the GETI instruction is executed and the referenced instruction is not skipped.
- If the instruction immediately following the GETI instruction has also the accumulation effect, the accumulation generated by the referenced instruction is valid and the following instruction is skipped.

Application example:

Replace

```
MOV     HL, #00H
MOV     XA, #0FFH
CALL    SUB1
B       SUB2
```

with the GETI instruction.

```
CODE0    CSEG     IENT
HL00   : MOV      HL, #00H
XAFF   : MOV      XA, #0FFH
CSUB1  : TCALL    SUB1
BSUB2  : TBR      SUB2
           ⋮
         GETI     HL00       ; MOV HL, #00H
           ⋮
         GETI     BSUB2      ; BR SUB2
           ⋮
         GETI     CSUB1      ; CALL SUB1
           ⋮
         GETI     XAFF       ; MOV XA, #0FFH
           ⋮
```

APPENDIX A DEVELOPMENT TOOLS

The following development tools are available for system development using the μPD75308.

Language Processor

RA75X relocatable assembler	Host Machine			Part Number (Product Name)	
		OS	Distribution Media		
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD	μS5A13RA75X	
			5-inch 2HD	μS5A10RA75X	★
	IBM PC/AT™ or its compatible machine	Refer to **OS of IBM PC**.	3.5-inch 2HC	μS7B13RA75X	★
			5-inch 2HC	μS7B10RA75X	

PROM Writing Tools

<table>
<tr><td rowspan="3">Hardware</td><td>PG-1500</td><td colspan="4">This PROM programmer allows programming, in stand-alone mode or via operation from a host machine, of a single-chip microcontroller with on-chip PROM by connection of the board provided and a separately available programmer adapter. Programming of a typical PROM of between 256 and 1M bits is also possible.</td><td></td></tr>
<tr><td>PA-75P308GF
PA-75P308K</td><td colspan="4">PROM programmer adapter for μPD75P308, 75P316, 75P316A used for connection to the PG-1500.</td><td></td></tr>
<tr><td>PA-75P316BGC
PA-75P316BGK</td><td colspan="4">PROM programmer adopter for μPD75P316B, used for connection to the PG-1500.</td><td></td></tr>
<tr><td rowspan="7">Software</td><td rowspan="7">PG-1500
controller</td><td colspan="4">Connects PG-1500 and host machine via a serial and parallel interface, and controls the PG-1500 on the host machine.</td><td></td></tr>
<tr><td colspan="3">Host Machine</td><td rowspan="2">Part Number
(Product Name)</td><td></td></tr>
<tr><td></td><td>OS</td><td>Distribution Media</td><td></td></tr>
<tr><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS
(Ver. 3.30 to Ver. 6.2[Note])</td><td>3.5-inch 2HD</td><td>μS5A13PG1500</td><td></td></tr>
<tr><td>5-inch 2HD</td><td>μS5A10PG1500</td><td>★</td></tr>
<tr><td rowspan="2">IBM PC/AT
or its compatible machine</td><td rowspan="2">Refer to OS of IBM PC.</td><td>3.5-inch 2HD</td><td>μS7B13PG1500</td><td>★</td></tr>
<tr><td>5-inch 2HC</td><td>μS7B10PG1500</td><td></td></tr>
</table>

Note Ver. 5.00 or later provides a task swap function, but this function cannot be used in combination with this software.

Remark Operation of the relocatable assembler and PG-1500 controller is assured only on the host machines and operating system quoted above.

Phase-out/Discontinued

Debugging Tools

In-circuit emulators (IE-75000-R and IE-75001-R) are available as μPD75308 debugging tools. The system configuration is shown below.

<table>
<tr><td rowspan="6">Hardware</td><td>IE-75000-R[Note 1]</td><td colspan="4">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 can be performed by connecting the IE-75000-R to a host machine and PROM programmer.</td></tr>
<tr><td>IE-75000-R-EM</td><td colspan="4">Emulation board for evaluating application systems that use the 75X series.
Used in combination with the IE-75000-R or IE-75001-R.
Incorporated in the IE-75000-R.</td></tr>
<tr><td>IE-75001-R[Note 2]</td><td colspan="4">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 the separately available IE-75000-R-EM emulation board and emulation probe.
Efficient debugging can be performed by connecting the IE-75001-R to a host machine and PROM programmer.</td></tr>
<tr><td>EP-75308GF-R
EV-9200G-80</td><td colspan="4">Emulation probe for the μPD75308GF.
Connected to the IE-75000-R, or to the IE-75001-R and IE-75000-R-EM.
An 80-pin conversion socket (EV-9200G-80) is also provided to simplify connection to the user system.</td></tr>
<tr><td>EP-75308BGC-R
EV-9200GC-80</td><td colspan="4">Emulation probe for the μPD75308BGC.
Connected to the IE-75000-R, or to the IE-75001-R and IE-75000-R-EM.
An 80-pin conversion socket (EV-9200GC-80) is also provided to simplify connection to the user system.</td></tr>
<tr><td>EP-75308BGK-R
EV-9500GK-80</td><td colspan="4">Emulation probe for the μPD75308BGK.
Connected to the IE-75000-R, or to the IE-75001-R and IE-75000-R-EM.
An 80-pin conversion socket (EV-9500GK-80) is also provided to simplify connection to the user system.</td></tr>
<tr><td rowspan="7">Software</td><td rowspan="7">IE control program</td><td colspan="4">Connects the IE-75000-R or IE-75001-R to the host machine via RS-232-C or Centronics I/F interface, and controls the IE-75000-R or IE-75001-R on the host machine.</td></tr>
<tr><td rowspan="2">Host Machine</td><td colspan="2"></td><td rowspan="2">Part Number
(Product Name)</td></tr>
<tr><td>OS</td><td>Distribution Media</td></tr>
<tr><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS
(Ver. 3.30 to Ver. 6.2[Note])</td><td>3.5-inch 2HD</td><td>μS5A13IE75X</td></tr>
<tr><td>5-inch 2HD</td><td>μS5A10IE75X</td></tr>
<tr><td rowspan="2">IBM PC/AT
or its compatible machine</td><td rowspan="2">Refer to OS of IBM PC.</td><td>3.5-inch 2HC</td><td>μS7B13IE75X</td></tr>
<tr><td>5-inch 2HC</td><td>μS7B10IE75X</td></tr>
</table>

Notes 1. Maintenance product

2. IE = 75000-R-EM is sold separately.

3. Ver. 5.00 or later provides a task swap function, but this function cannot be used in combination with this software.

Remark Operation of the IE control program is assured only on the host machines and operating system quoted above.

OS of IBM PC ★

The following OS is supported as the OS for IBM PC.

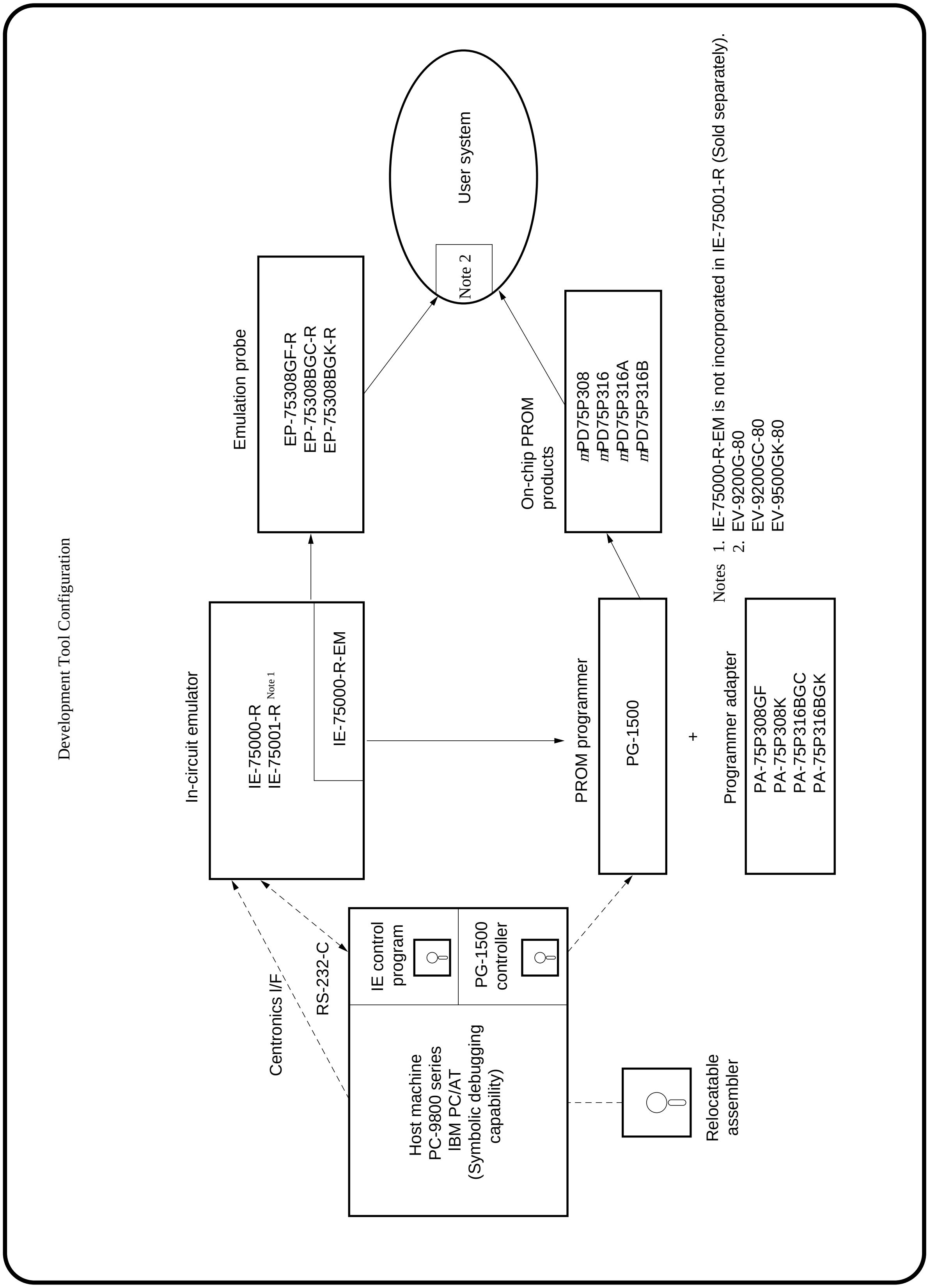
OS	Version
PC DOS	Ver. 3.1 to Ver. 6.3 J6.1/V[Note] to J6.3/V[Note]
MS-DOS	Ver. 5.0 to Ver. 6.22 5.0/V[Note] to 6.2/V[Note]
IBM DOS	J5.02/V[Note]

Note Only the English mode is supported.

Caution Ver. 5.00 or later provides a task swap function, but this function cannot be used in combination with this software.

Phase-out/Discontinued

Development Tool Configuration



Notes 1. IE-75000-R-EM is not incorporated in IE-75001-R (Sold separately).
2. EV-9200G-80
EV-9200GC-80
EV-9500GK-80

APPENDIX B MASK ROM ORDERING PROCEDURE

On completion of the μPD75308 program, the following procedure should be used to order the mask ROM.

① Mask ROM order reservation
Please notify your sales agent or NEC sales department of your intention to order a mask ROM.

② Creation of order medium
The medium for a mask ROM order is UVEPROM or a 3.5 or 5-inch IBM format floppy disk (outside Japan only). When UVEPROM is used for the order, three UV-EPROMs with identical contents should be prepared (in the case of products with a mask option, the mask option data should be submitted on a mask option information sheet).

③ Necessary documentation
When ordering a mask ROM, the following documents should be filled out.
A. Mask ROM order form
B. Mask ROM order check sheet
C. Mask option information sheet (only required for mask option products)

④ Ordering
The media created in ② and documentation filled out in ③ should be submitted to the agent or NEC sales department by the order reservation date.

Caution For details, please refer to the technical document ROM Code Ordering Procedure (IEM-1366).

[MEMO]

APPENDIX C INSTRUCTION INDEX

C.1 Instruction Index (in function)

[Branch Instructions]

[Interrupt Control Instructions]

[Input/Output Instructions]

[CPU Control Instructions]

[Special Instructions]

C.2 Instruction Index (in general)

[P]

[R]

[S]

[T]

[X]

APPENDIX D HARDWARE INDEX

[T]

[W]

APPENDIX E REVISION HISTORY ★

The following shows the revision history of this manual. “Chapter” indicates the chapter of the previous edition.

Edition	Revision	Chapter
11th edition	The μPD75P316BKK-T has been added to the applicable products.	Whole manual
	The μPD75312B and 75316B have been developed.	
	The PA-75P316BGC, PA-75P316BGK have been developed.	
	MS-DOS Ver. 5.00/5.00A for the PC-9800 series has been supported.	
	2.2.26 IC has been added.	CHAPTER 2 PIN FUNCTIONS
	2.6 Caution of Use of P00/INT4 Pin and $\overline{\text{RESET}}$ Pin has been added.	
	Figure 5-3. Configuration of Port 3n and 6n (n = 0 to 3) has been modified.	CHAPTER 5 PERIPHERAL HARDWARE FUNCTIONS
	Figure 5-4. Configuration of Ports 2 and 7 has been modified.	
	Figure 5-10. Clock Generator Circuit Block Diagram has been modified and a note has been added to the figure.	
	5.2.2 (4) Frequency divider has been added.	
	5.2.2 (5) When subsystem clock is not used has been added.	
	A note has been added to **Table 5-5. Maximum Time Required for System Clock/CPU Clock Switching**.	
	A note has been added to **Table 6-1. Interrupt Source Types**.	CHAPTER 6 INTERRUPT FUNCTION
	6.3 (3) INT0, INT1 and INT4 hardware has been modified.	
	Precautions have been added to **CHAPTER 7 STANDBY FUNCTION**.	CHAPTER 7 STAND-BY FUNCTION
	A note has been added to **7.4 Standby Mode Application**.	
	A note has been added to **7.4 (1) Example of STOP mode application**.	
12th edition	The μPD75P316B has been developed.	Whole manual
	The note in **Figure 5-83. Example of LCD Drive Power Supply Connection (With External Split Resistor)** has been modified and a caution has been added to this figure.	CHAPTER 5 PERIPHERAL HARDWARE FUNCTIONS
	The example in **6.6 Vector Address Sharing Interrupt Servicing** has been modified.	CHAPTER 6 INTERRUPT FUNCTION
	The version of the supported OS has been up-graded.	APPENDIX A DEVELOPMENT TOOLS

[MEMO]

Phase-out/Discontinued

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-551-0451

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❏	❏	❏	❏
Technical Accuracy	❏	❏	❏	❏
Organization	❏	❏	❏	❏

CS 95.12