NEC

User's Manual

μPD754304

4-bit Single-Chip Microcontrollers

μPD754302
μPD754304
μPD75P4308

Document No. U10123EJ2V1UM00 (2nd edition)
Date Published November 1999 N CP (K)

[MEMO]

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 this product from Japan is regulated by the Japanese government. To export this product may be prohibited without governmental license, the need for which must be judged by the customer. The export or re-export of this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

- **The information in this document is subject to change without notice. Before using this document, please confirm that this is the latest version.**
- **Not all devices/types available in every country. Please check with local NEC representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.
- NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.
- Descriptions of circuits, software, and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software, and information in the design of the customer's equipment shall be done under the full responsibility of the customer. NEC Corporation assumes no responsibility for any losses incurred by the customer or third parties arising from the use of these circuits, software, and information.
- While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.
- NEC devices are classified into the following three quality grades:
  "Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated “quality assurance program“ for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.
  - Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
  - Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
  - Specific: Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

  The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

M7D 98.12

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Rodovia Presidente Dutra, Km 214
07210-902-Guarulhos-SP Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

J99.1

Major Revisions in this Edition

Page	Description
Throughout	The development status of the μPD754302, 754304, 75P4308 is changed from “Under development” to “Development completed.”
	The input withstand voltage of the port 5’s pins in the open-drain mode is changed from 12 V to 13 V.
	COI is added to serial operation mode register (CSIM).
	Slave address register (SVA) is added.
p.141	An address comparator is added to **Figure 5-35 Serial Interface Block Diagram**.
p.225	**CHAPTER 10 MASK OPTION** is added.
p.253	Modification of the instruction list in 11.3 Op Code of Each Instruction.
p.301-302	The versions of the supported operating systems are upgraded in **APPENDIX B DEVELOPMENT TOOLS**.
p.315	**APPENDIX F REVISION HISTORY** is added.

The mark ★ shows major revised points.

INTRODUCTION

Readers: This manual is intended for user engineers who understand the functions of the μPD754302, 7534304, and 75P4308 4-bit single-chip microcontrollers, and wish to design application systems using any of these microcontrollers.

Purpose: This manual describes the hardware functions of the μPD754302, 754304, and 75P4308 in the organization described below.

Organization: This manual contains the following information:

- General
- Pin Functions
- Features of Architecture and Memory Map
- Internal CPU Functions
- Peripheral Hardware Functions
- Interrupt Functions and Test Functions
- Standby Functions
- Reset Function
- Writing and Verifying PROM
- Mask options
- Instruction Set

How to Read This Manual:

It is assumed that readers for this manual have general knowledge on electricity, logic circuits, and microcontrollers.

- If you have experience of using the μPD750004,
  → Read **APPENDIX A μPD750004, μPD754304, AND μPD75P4308 FUNCTION LIST** to check differences between the μPD750004 and the microcontrollers described in this manual.

- To check the functions of an instruction whose mnemonic is known,
  → Refer to **APPENDIX D INSTRUCTION INDEX**.

- To check the functions of a specific internal circuit,
  → Refer to **APPENDIX E HARDWARE INDEX**.

- To understand the overall functions of the μPD754302, 754304, and 75P4308,
  → Read this manual in the order of Table of Contents.

Legend

Data significance	:	Left: higher, right: lower
Active low	:	$\overline{\times\times\times}$ (top bar over signal or pin name)
Address of memory map	:	Top: low, Bottom: high
Note	:	Footnote
Caution	:	Important information
Remark	:	Supplement
Important point and emphasis	:	Bold letters
Numeric notation	:	Binary ·····×××× or ××××B
		Decimal ·····××××
		Hexadecimal ····××××H

Related Documents The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Documents related to devices

Document name	Document number	
	English	Japanese
μPD754304 User's Manual	This manual	U10123J
μPD754302, 754304, 754302(A), 754304(A) Data Sheet	U10797E	U10797J
μPD75P4308 Data Sheet	U10909E	U10909J
μPD754304 Instruction List	—	IEM-5605
75XL Series Selection Guide	U10453E	U10453J

Documents related to development tools ★

Document name			Document number	
			English	Japanese
Hardware	IE-75000-R/IE-75001-R User's Manual		EEU-1416	EEU-846
	IE-75300-R-EM User's Manual		U11354E	U11354J
	EP-754304GS-R User's Manual		U10667E	U10677J
	PG-1500 User's Manual		U11940E	U11940J
Software	RA75X	Operation	U12622E	U12622J
	Assembler Package User's Manual	Language	U12385E	U12385J
	PG-1500 Controller User's Manual	PC-9800 series (MS-DOS™-based)	EEU-1291	EEU-704
		IBM PC series (PC DOS™-based)	U10540E	U10540J

Other documents ★

Document name	Document number	
	English	Japanese
SEMICONDUCTORS SELECTION GUIDE Products & Packages (CD-ROM)	X13769X	
Semiconductor Device Mounting Technology Manual	C10535E	C10535J
Quality Grades on NEC Semiconductor Devices	C11531E	C11531J
NEC Semiconductor Device Reliability/Quality Control System	C10983E	C10983J
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E	C11892J
Guide to Microcomputer-Related Products by Third Party	—	U11416J

Caution The above related documents are subject to change without notice. Be sure to use the latest edition when you design your system.

[MEMO]

TABLE OF CONTENTS

LIST OF FIGURES (1/3)

LIST OF FIGURES (2/3)

LIST OF FIGURES (3/3)

LIST OF TABLES

CHAPTER 1 GENERAL

The μPD754302, 754304, and 75P4308 are 4-bit single-chip microcontrollers in the NEC's 75XL series, a successor to the 75X series that boasts a wealth of variations. These three devices are called the μPD754304 subseries as a general term.

The 75XL Series is the successor to the 75X Series CPUs and attains a wide range of operating voltages and high-speed operation. Not only is it upward compatible with previous products, but is suited to applications that use batteries.

The features of the μPD754308 are as follows:

- Low-voltage operation: V_{DD} = 1.8 to 5.5 V
- Variable instruction execution time useful for high-speed operation and power saving
  0.95 μs, 1.91 μs, 3.81 μs, 15.3 μs (at 4.19 MHz)
  0.67 μs, 1.33 μs, 2.67 μs, 10.7 μs (at 6.0 MHz)
- Powerful timer function: 3 channels
- Small package (36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)

The μPD75P4308 is provided with a one-time PROM that can be electrically written and is pin-compatible with the μPD754302 and 754304. This one-time PROM model is convenient for experimental development or small-scale production of an application system.

Application Fields

- VCRs
- Audio (CD players)
- Remote controllers
- Telephones
- Cameras, etc.

Remark Unless otherwise specified, the μPD754304 is regarded as the representative model, and description throughout this manual is focused on this model.

1.1 Functional Outline

Parameter			Function	
Instruction execution time			• 0.95, 1.91, 3.81, 15.3 μs (@ 4.19 MHz with main system clock) • 0.67, 1.33, 2.67, 10.7 μs (@ 6.0 MHz with main system clock)	
On-chip memory		ROM	2048 × 8 bits (μPD754302)	
			4096 × 8 bits (μPD754304)	
			8192 × 8 bits (μPD75P4308)	
		RAM	256 × 4 bits	
General-purpose register			• 4-bit operation: 8 × 4 banks • 8-bit operation: 4 × 4 banks	
Input/ output port	CMOS input		8 pins	Of these, seven have software-specifiable on-chip pull-up resistors.
	CMOS input/output		18 pins	Eighteen have software-specifiable on-chip pull-up resistors.
★	N-ch open-drain input/output		4 pins	Each withstands up to 13 V and has a mask-option pull-up resistor **Note**
	Total		30 pins	
Timer			3 channels • 8-bit timer/event counter: 2 channels (Can also be used as a jointed 16-bit timer/ event counter • Basic interval timer/watchdog timer: 1 channel	
Serial interface			• 3-wire serial I/O mode ... MSB or LSB can be selected for transferring top bit. • 2-wire serial I/O mode	
Bit sequential buffer (BSB)			16 bits	
Clock output (PCL)			• Φ, 524, 262, 65.5 kHz (@ 4.19 MHz with main system clock) • Φ, 750, 375, 93.8 kHz (@ 6.0 MHz with main system clock)	
Vectored interrupts			External: 3, Internal: 4	
Test input			External: 1	
System clock oscillator			Ceramic or crystal oscillator	
Standby function			STOP/HALT mode	
Operating ambient temperature			T_A = –40 to +85 ˚C	
Power supply voltage			V_{DD} = 1.8 to 5.5 V	
Package			36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)	

★ **Note** The N-ch open-drain I/O port pins of the μPD75P4308 are not connected to pull-up resistors by mask option, and are always open.

1.2 Ordering Information

Part number	Package	Internal ROM
μPD754302GS-×××	36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)	Mask ROM
μPD754304GS-×××	36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)	Mask ROM
μPD75P4308GS	36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)	One-time PROM

Remark ××× indicates a ROM code suffix.

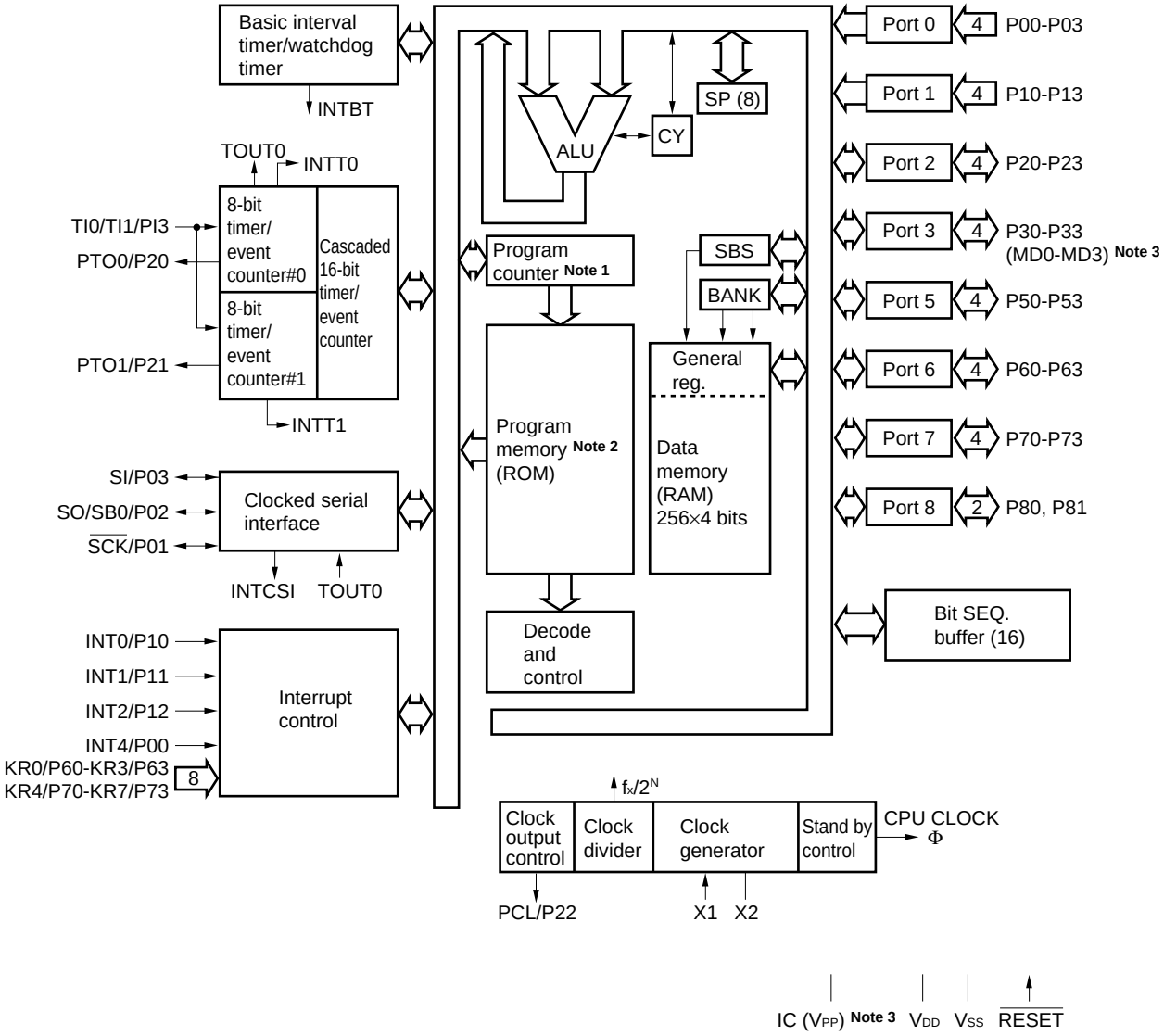
1.3 Differences among Subseries Products

Item			μPD754302	μPD754304	μPD75P4308
ROM (bytes)			· Mask ROM · 2048 bytes · 0000H-07FFH	· Mask ROM · 4096 bytes · 0000H-0FFFH	· One-time PROM · 8192 bytes · 0000H-1FFFH
RAM (× 4 bits)			256		
Program counter			11 bits	12 bits	13 bits
★ Mask option	PORT5 pull-up resister		Yes (can be specified on chip or not)		No (on chip not possible)
	Wait time during $\overline{\text{RESET}}$		Yes (can select either 2^{17}/fx or 2^{15}/fx) **Note**		No (fixed at 2^{15}/fx) **Note**
Pin connection		Pins 5 to 8	P30 to P33		P30/MD0 to P33/MD3
		Pin 19	IC		V_{PP}
Power supply voltage			V_{DD} = 1.8 to 5.5 V		
Package			36-pin plastic shrink SOP (300 mil)		
Other			Noise immunity and noise radiation differ because circuit scale and mask layout differ.		

Note The oscillation stabilization wait time becomes 21.8 ms at 6.0 MHz and 31.3 ms at 4.19 MHz if $2^{17}/f_x$ is selected; it becomes 5.46 ms at 6.0 MHz and 7.81 ms at 4.19 MHz if $2^{15}/f_x$ is selected.

Caution The noise immunity and noise radiation of the PROM model differ from those of the mask ROM model. If you replace the PROM model with the mask ROM model in the course of experimental production to mass production, perform thorough evaluation by using the CS model (not ES model) of the mask ROM model.

1.4 Block Diagram



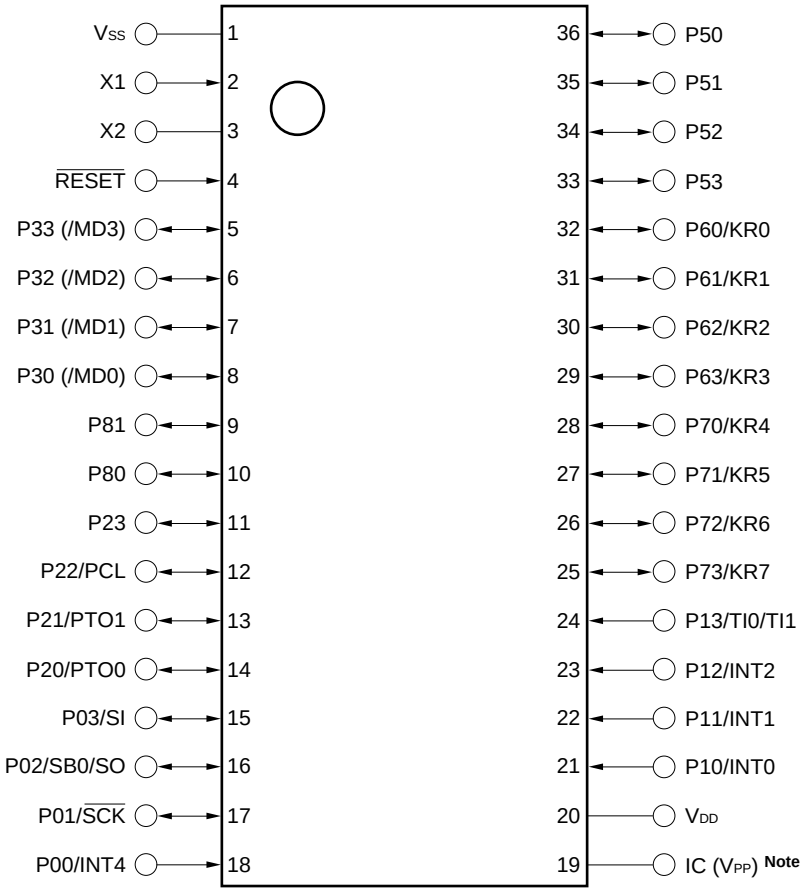
Notes 1. The counters in the μPD754302, 754304, and 75P4308 consist of 11, 12, and 13 bits, respectively.

2. 2048 × 8 bits in μPD754302. 4096 × 8 bits in μPD754304. 8192 × 8 bits in μPD75P4308.

3. The pin names or functions in parentheses applies with the μPD75P4308.

1.5 Pin Configuration (Top View)

- **36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)**
  μPD754302GS-×××
  μPD754304GS-×××
  μPD75P4308GS



Note Directly connect the IC (V_{PP}) pin to V_{DD}.

Remark The pin names or functions in parentheses applies with the μPD75P4308

Pin Identification

P00 to P03	:	Port 0	TI0, TI1	:	Timer Input 0, 1
P10 to P13	:	Port 1	PTO0, PTO1	:	Programmable Timer Output 0, 1
P20 to P23	:	Port 2	PCL	:	Programmable Clock
P30 to P33	:	Port 3	INT0, INT1, INT4	:	External Vectored Interrupt 0, 1, 4
P50 to P53	:	Port 5	INT2	:	External Test Input 2
P60 to P63	:	Port 6	X1, X2	:	System Clock Oscillation 1, 2
P70 to P73	:	Port 7	V_{DD}	:	Positive Power Supply
P80, P81	:	Port 8	V_{SS}	:	Ground
KR0 to KR7	:	Key Return 0 to 7	V_{PP}	:	Programming Power Supply
$\overline{SCK}$	:	Serial Clock	IC	:	Internally Connected
SI	:	Serial Input	MD0 to MD3	:	Mode Selection
SO	:	Serial Output			
SB0	:	Serial data Bus 0			
$\overline{RESET}$	:	Reset Input			

[MEMO]

CHAPTER 2 PIN FUNCTION

2.1 Pin Functions of μPD754304

Table 2-1. Pin Functions of Digital I/O Ports (1/2)

Pin name	Input/output	Alternate function	Function	8-bit I/O	After reset	I/O circuit type Note 1
P00	Input	INT4	4-bit input port (PORT0). For P01 to P03, on-chip pull-up resistors can be specified by software in 3-bit units.	No	Input	[B]
P01	Input/Output	$\overline{SCK}$				[F] -A
P02	Input/Output	SO/SB0				[F] -B
P03	Input	SI				[B] -C
P10	Input	INT0	4-bit input port (PORT1). On-chip pull-up resistors can be specified by software in 4-bit units. With noise eliminating function (Only P10/INT0)	No	Input	[B] -C
P11		INT1				
P12		INT2				
P13		TI0/TI1				
P20	Input/Output	PTO0	4-bit input/output port (PORT2). On-chip pull-up resistors can be specified by software in 4-bit units.	No	Input	E-B
P21		PTO1				
P22		PCL				
P23		–				
P30 Note 2	Input/Output	(MD0) Note 3	Programmable 4-bit input/output port (PORT3). This port can be specified input/output bit-wise. On-chip pull-up resistor can be specified by software in 4-bit units.	No	Input	E-B
P31 Note 2		(MD1) Note 3				
P32 Note 2		(MD2) Note 3				
P33 Note 2		(MD3) Note 3				
P50-P53 Note 2, 4	Input/Output	–	N-ch open-drain 4-bit input/output port (PORT5). Each pin withstands up to 13 V in open-drain mode. A pull-up resistor can be contained bit-wise (mask option). Note 5 The higher 4 bits are used as data input/output pins in the program memory (PROM) write/verify mode.	No	High level (when pull-up resistors are provided) or high-impedance	M-D (M-E) Note 3 ★

Notes 1. The circuit types enclosed with [ ] are Schmitt-triggered input circuits.

2. LED can be directly driven.

3. The pin names or functions in parentheses are available with the μPD75P4308.

4. If on-chip pull-up resistors are not specified by mask option (when used as N-ch open-drain input port), low level input leakage current increases when input or bit manipulation instruction is executed.

5. These pins of the μPD75P4308 are not provided with pull-up resistors by mask option and are always open.

Table 2-1. Pin Functions of Digital I/O Ports (2/2)

Pin name	Input/output	Alternate function	Function	8-bit I/O	After reset	I/O circuit type Note
P60	Input/Output	KR0	Programmable 4-bit input/output port (PORT6). This port can be specified for input/output bit-wise. On-chip pull-up resistors can be specified by software in 4-bit units. The lower 4 bits are used as data input/output pins in the program memory (PROM) write/verify mode.	Yes	Input	[F]-A
P61		KR1				
P62		KR2				
P63		KR3				
P70	Input/Output	KR4	4-bit input/output port (PORT7). On-chip pull-up resistors can be specified by software in 4-bit units.		Input	[F]-A
P71		KR5				
P72		KR6				
P73		KR7				
P80	Input/Output	–	2-bit input/output port (PORT8). On-chip pull-up resistors can be specified by software in 2-bit units.	No	Input	E-B
P81		–				

Note The circuit types enclosed with [ ] are Schmitt-triggered input circuits.

Table 2-2. Pin Function of Pins Other Than Port Pins

Pin name	Input/output	Alternate function	Function		After reset	I/O circuit type Note 1
TI0/TI1	Input	P13	Inputs external event pulses to the timer/ event counter.		Input	[B]-C
PTO0	Input/Output	P20	Timer/event counter output		Input	E-B
PTO1		P21				
PCL		P22	Clock output			
$\overline{SCK}$	Input/Output	P01	Serial clock input/output		Input	[F]-A
SO/SB0		P02	Serial data output Serial data bus input/output			[F]-B
SI		P03	Serial data input			[B]-C
INT4	Input	P00	Edge detection vectored interrupt input (both rising edge and falling edge detection are valid)		Input	[B]
INT0	Input	P10	Edge detection vectored interrupt input (detection edge can be selected). INT0/P10 has a noise elimination function.	Clock synchronous/asynchronous selection	Input	[B]-C
INT1		P11		Asynchronous		
INT2	Input	P12	Rising edge-detection-testable input	Asynchronous	Input	[B]-C
KR0 to KR3	Input/Output	P60 to P63	Parallel falling edge detection testable input		Input	[F]-A
KR4 to KR7		P70 to P73				
X1, X2	Input	–	Crystal/ceramic connection pin for the main system clock oscillator. When inputting the external clock, input the external clock to pin X1, and the reverse phase of the external clock to pin X2.		–	–
$\overline{RESET}$	Input	–	System reset input (low-level active)		–	[B]
IC Note 2	–	–	Internally connected. Connect this pin directly to V_{DD}.		–	–
V_{DD}	–	–	Positive power supply		–	–
V_{SS}	–	–	Ground potential		–	–
V_{PP} Note 3	–	–	Supplies voltage necessary for writing/ verifying program memory (PROM). Apply +12.5 V to this pin to write/verify PROM.		–	–
MD0 to MD3 Note 3	Input/Output	P30 to P33	Select modes for writing/verifying program memory (PROM).		Input	E-B

Notes 1. The circuit types enclosed with [ ] are Schmitt-triggered input circuits.

2. Used as V_{PP} for μPD75P4308.

3. Only μPD75P4308 supports.

2.2 Pin Functions

2.2.1 P00 to P03 (PORT0) ··· input shared with INT4, $\overline{SCK}$, SO/SB0, and SI
P10 to P13 (PORT1) ··· input shared with INT0, 1, 2 and TI0/TI1

4-bit input port: These pins are the input pins of ports 0 and 1, respectively.

Ports 0 and 1 also have the following functions, in addition to the input port function:

(1) Port 0 : Vector interrupt input (INT4)
Serial interface I/Os ($\overline{SCK}$, SO/SB0, SI)

(2) Port 1 : Vector interrupt inputs (INT0, INT1)
Edge detection test input (INT2)
External event pulse input to timer/event counter (TI0/TI1)

The status of each pin of ports 0 and 1 can be always input regardless of the operation of the shared pins.

The pins of port 0 and port 1 are Schmitt trigger input pins to prevent malfunctioning due to noise. In addition, the P10 pin is provided with a noise rejecter circuit (for details, refer to (3) Hardware of INT0, INT1, and INT4 in section 6.3).

Software enables Port 0 in 3-bit units (P01 to P03) and Port 1 in 4-bit units (P10 to P13) to connect with on-chip pull-up resistors. Whether or not the on-chip pull-up resistors are connected is specified by using the pull-up resistor specification register group A (POGA).

When the $\overline{RESET}$ signal is asserted, all the pins are set in the input mode.

2.2.2 P20 to P23 (PORT2) ··· I/O shared with PTO0, PTO1, and PCL
P30 to P33 (PORT3) ··· I/O shared with MD0 to MD3
★ ### P50 to P53 (PORT5) ··· N-ch open-drain, medium withstand voltage (13 V), high-current output
P60 to P63 (PORT6) and P70 to P73 (PORT7) ··· I/O shared with KR0 to KR3 and KR4 to KR7

4-bit I/O ports with output latch: I/O pins of ports 2, 3, 5, 6, and 7

In addition to the I/O port function, port n (n = 2, 3, 6, or 7) has the following functions:

- Port 2 : Timer/event counter outputs (PTO0, PTO1)
  Clock output (PCL)
- Port 3 : Mode selection when writing or verifying the program memory (PROM) (MD0 to MD3) **Note**
- Ports 6 and 7 : Key interrupt input (KR0 to KR3 and KR4 to KR7)

Note MD0 to MD3 used only in μPD75P4308.

Port 3 can output a high current and therefore can directly drive an LED.

Port 5 is an N-ch open-drain, medium-voltage (13 V) port. ★

Ports 2, 5, and 7 can be set in input or output mode in 4-bit units, and ports 3 and 6 can be set in input or output mode in 1-bit units.

Ports 2, 3, 6, and 7 can be connected with on-chip pull-up resistors in 4-bit units via software, by manipulating the pull-up resistor specification registers group A (POGA). Port 5 of the μPD754304 can be connected with a pull-up resistor in 1-bit units by mask option. However, the ports of the μPD75P4308 cannot be connected with a pull-up resistor by mask option and are always open.

Ports 6 and 7 can be set in input or output mode in pairs in 8-bit units. When the $\overline{\text{RESET}}$ signal is asserted, each port and the output latch are cleared and ports 2, 3, 6, and 7 are set in input mode (output high impedance). Port 5 is set at the high-level (when the pull-up resistor is connected) or high-impedance state.

2.2.3 P80, P81 (PORT8)

2-bit I/O port with output latch: These I/O pins are for port 8.

Port 8 can set the connections for the software selectable internal pull-up resistors in 2-bit units by manipulating pull-up resistor specification register group B (POGB).

2.2.4 TI0/TI1 ··· inputs shared with port 1

These are the external pulse event input pins of programmable timers/event counters 0 through 2.

TI0 through TI1 are Schmitt trigger input pins.

2.2.5 PTO0, PTO1 ··· outputs shared with port 2

These are the output pins of programmable timers/event counters 0 through 2, and output square wave pulses. To output the signal of a programmable timer/event counter, clear the output latch of the corresponding pin of port 2 to “0”, and set the bit corresponding to port 2 of the port mode register to “1” to set the output mode.

The outputs of these pins are cleared to “0” by the timer start instruction.

2.2.6 PCL ··· output shared with port 2

This is a programmable clock output pin and is used to supply the clock to a peripheral LSI (such as a slave microcomputer). When the $\overline{\text{RESET}}$ signal is asserted, the contents of the clock mode register (CLOM) are cleared to “0”, disabling the output of the clock. In this case, the PCL pin can be used as an ordinary port pin.

2.2.7 $\overline{SCK}$, SO/SB0, and SI ⋯ 3-state I/Os shared with port 0

These are serial interface I/O pins and operate in accordance with the setting of the serial operation mode register (CSIM).

When the $\overline{RESET}$ signal is asserted, the serial interface operation is stopped, and these pins served as input port pins.

All these pins are Schmitt trigger input pins.

2.2.8 INT4 ⋯ input shared with port 0

This is an external vector interrupt input pin and becomes active at both the rising and falling edges. When the signal input to this pin changes from high to low level or vice versa, the interrupt request flag is set.

INT4 is an asynchronous input pin and the interrupt is acknowledged when a high-level or low-level signal is input to this pin for a fixed time, regardless of the operating clock of the CPU.

INT4 can also be used to release the STOP and HALT modes. This pin is a Schmitt trigger input pin.

2.2.9 INT0 and INT1 ⋯ inputs shared with port 1

These pins input interrupt signals that are detected by the edge. INT0 has a noise rejection function. The edge to be detected can be specified by using the edge detection mode registers (IM0 and IM1).

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

(a) Active at rising edge
(b) Active at falling edge
(c) Active at both rising and falling edges
(d) External interrupt signal input disabled

(2) INT1 (bit 0 of IM1)

(a) Active at rising edge
(b) Active at falling edge

INT0 has a noise rejection function and the sampling clock that rejects noise can be changed in two steps. The width of the signal that is acknowledged differs depending on the CPU operating clock.

INT1 is an asynchronous input pin. The signal input to this pin is acknowledged as long as the signal has a specific high-level width, regardless of the operating clock of the CPU.

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

Both INT0 and INT1 can be used to release the STOP and HALT modes. However, when the noise rejection circuit is selected, INT0 cannot be used to release the STOP and HALT modes.

INT0 and INT1 are Schmitt trigger input pins.

2.2.10 INT2 ⋯ input shared with port 1

This pin inputs an external test signal that is active at the rising edge. When INT2 is selected by the edge detection mode register (IM2), and when the signal input to this pin goes low, an internal test flag (IRQ2) is set.

INT2 is an asynchronous input. The signal input to this pin is acknowledged as long as it has a specific high-level width, regardless of the operating clock of the CPU.

When the $\overline{\text{RESET}}$ signal is asserted, the contents of IM2 are cleared to “0”, and the test flag (IRQ2) is set at the rising edge of the INT2 pin.

INT2 can be used to release the STOP and HALT modes. It is a Schmitt trigger input pin.

2.2.11 KR0 to KR3 ⋯ inputs shared with port 6
KR4 to KR7 ⋯ inputs shared with port 7

These are key interrupt input pins, which input interrupt signals that are detected in parallel at falling edge.

The interrupt format can be specified by using the edge detection mode register (IM2).

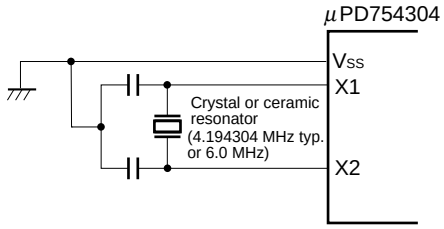
When the $\overline{\text{RESET}}$ signal is asserted, these pins serve as ports 6 and 7 pins and set in input mode.

2.2.12 X1 and X2

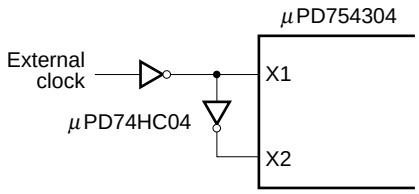
These pins connect a crystal/ceramic oscillator for main system clock oscillation.

An external clock can also be input to these pins.

(a) Crystal/ceramic oscillation



(b) External clock



2.2.13 $\overline{\text{RESET}}$

This pin inputs a low-active $\overline{\text{RESET}}$ signal.

The $\overline{\text{RESET}}$ signal is an asynchronous input signal and is asserted when a signal with a specific low-level width is input to this pin regardless of the operating clock. The $\overline{\text{RESET}}$ signal takes precedence over all the other operations.

This pin can not only be used to initialize and start the CPU, but also to release the STOP and HALT modes.

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

2.2.14 V_{DD}

Positive power supply pin.

2.2.15 V_{SS}

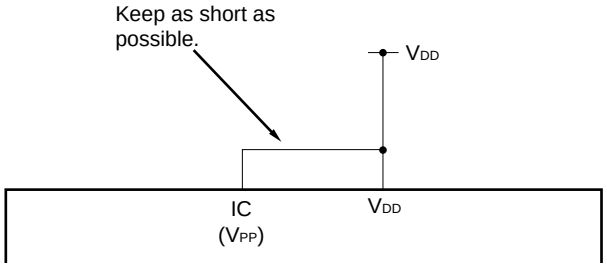
GND.

2.2.16 IC (μPD754302, and 754304 only)

The IC (Internally Connected) pin sets a test mode in which the μPD754304 is tested before shipment. It is usually best to connect the IC pin directly to the V_{DD} pin with as short a wiring length as possible.

If a voltage difference is generated between the IC and V_{DD} pins because the wiring length between the IC and V_{DD} pins is too long, or because external noise is superimposed on the IC pin, your program may not be correctly executed.

- **Directly connect the IC pin to the V_{DD} pin.**



2.2.17 V_{PP} (μPD75P4308 only)

This pin inputs a program voltage when the program memory (one-time PROM) is written or verified.

It is usually best to connect this pin directly to the V_{DD} (refer to the figure above). Apply 12.5 V to this pin when the PROM is written or verified.

2.2.18 MD0 to MD3 (μPD75P4308 only) ⋯ inputs/outputs shared with Port 3

These pins are provided to the μPD75P4308 only, and are used to select a mode when the program memory (one-time PROM) is written or verified.

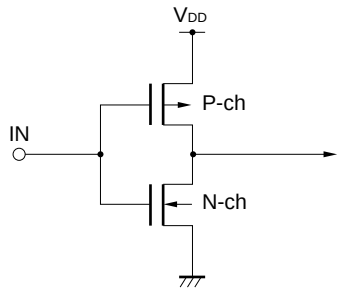
2.3 Pin Input/Output Circuits

The μPD754304 pin input/output circuits are shown schematically.

Figure 2-1. Pin I/O Circuits (1/2)

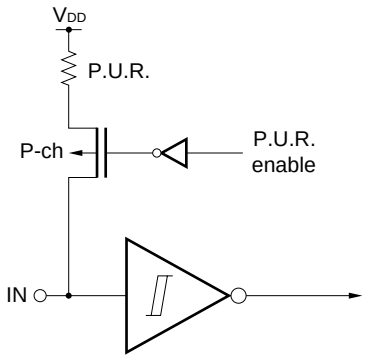
(1/2)

TYPE A



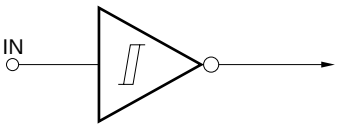
CMOS specification input buffer.

TYPE B-C



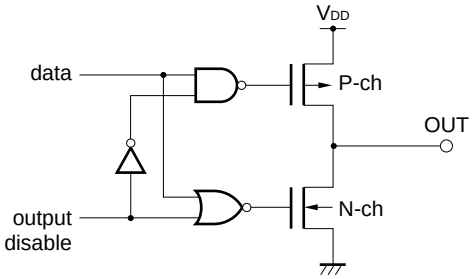
P.U.R. : Pull-Up Resistor

TYPE B



Schmitt trigger input having hysteresis characteristic.

TYPE D

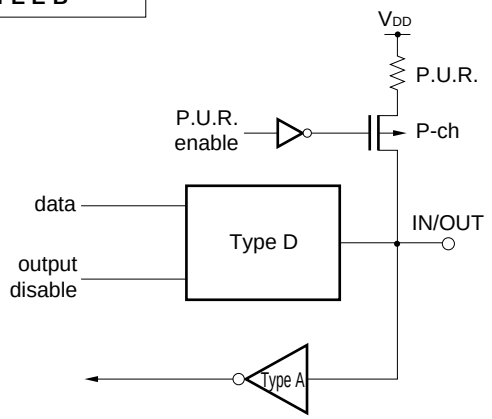


Push-pull output that can be placed in output high-impedance (both P-ch, N-ch off).

Figure 2-1. Pin I/O Circuits (2/2)

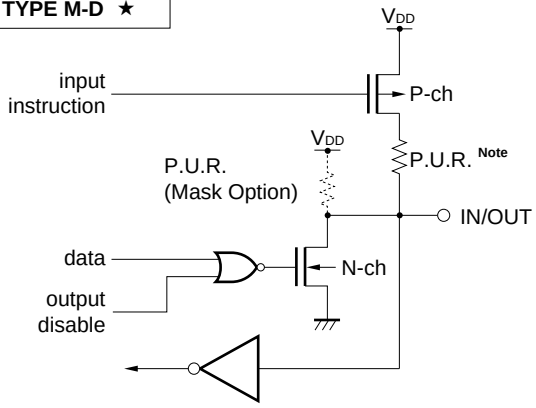
(2/2)

TYPE E-B



P.U.R. : Pull-Up Resistor

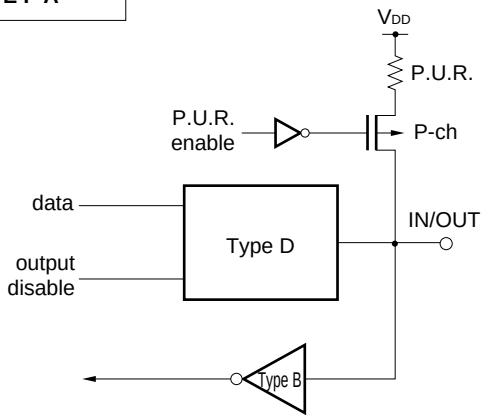
TYPE M-D ★



Medium-voltage input buffer
(Withstands 13 V max.)

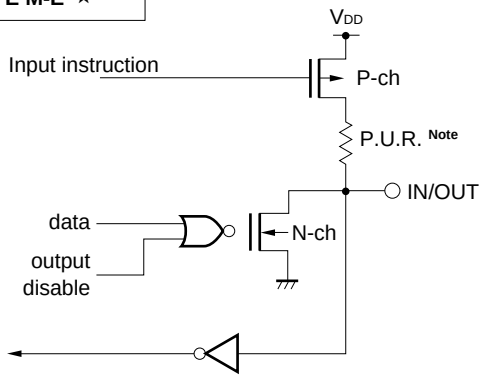
Note P.U.R. stands for pull-up resistor. This resistor functions when an input instruction is executed and the low level is input.

TYPE F-A



P.U.R. : Pull-Up Resistor

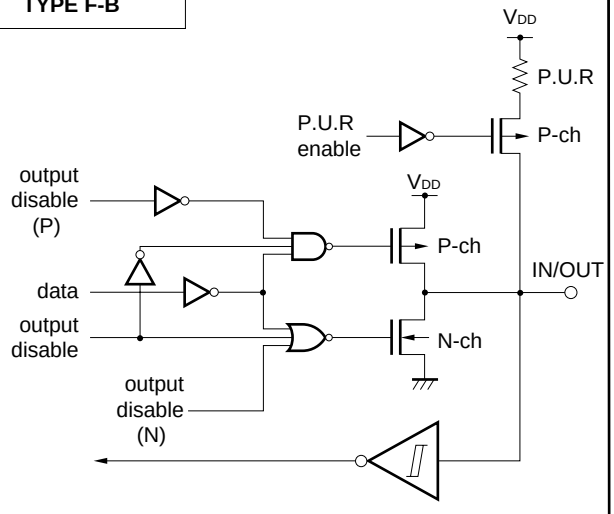
TYPE M-E ★



Medium-voltage input buffer
(Withstands 13 V max.)

Note P.U.R. stands for pull-up resistor. This resistor functions when an input instruction is executed and the low level is input.

TYPE F-B



P.U.R : Pull-Up Resistor

2.4 Recommended Connections for Unused Pins

Table 2-3. List of Recommended Connections for 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	
P10/INT0-P12/INT2	Connect to V_{SS}.
P13/TI0/TI1	
P20/PTO0	Input mode: Connect to V_{SS} or V_{DD} individually via resistor.
P21/PTO1	Output mode: Leave unconnected.
P22/PCL	
P23	
P30 (/MD0) **Note** to P33 (/MD3) **Note**	
P50-P53	
P60/KR0-P63/KR3	
P70/KR4 to P73/KR7	
P80 and P81	
IC (V_{PP}) **Note**	Connect to V_{DD} directly.

Note The function or assignment in parentheses is applied only to the μPD75P4308.

[MEMO]

CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP

The 75XL architecture employed for the μPD754304 has the following features:

- Internal RAM: 4 Kwords × 4 bits MAX. (12-bit address)
- Expandability of peripheral hardware

To realize these superb features, the following methods are employed:

(1) Bank configuration of data memory
(2) Bank configuration of general-purpose registers
(3) Memory mapped I/O

This chapter describes each of these features.

3.1 Bank Configuration of Data Memory and Addressing Mode

3.1.1 Bank configuration of data memory

The μPD754304 is provided with static RAM of 256 words × 4 bits at the addresses 000H through 0FFH of the data memory space. Peripheral hardware units (such as I/O ports and timers) are allocated to addresses F80H through FFFH.

The μPD754304 employs memory bank configuration that directly or indirectly specifies the lower 8 bits of an address by an instruction and the higher 4 bits of the address by a memory bank when the data memory space of 12-bit address (4 Kwords × 4 bits) is addressed.

To specify a memory bank (MB), the following hardware units are provided:

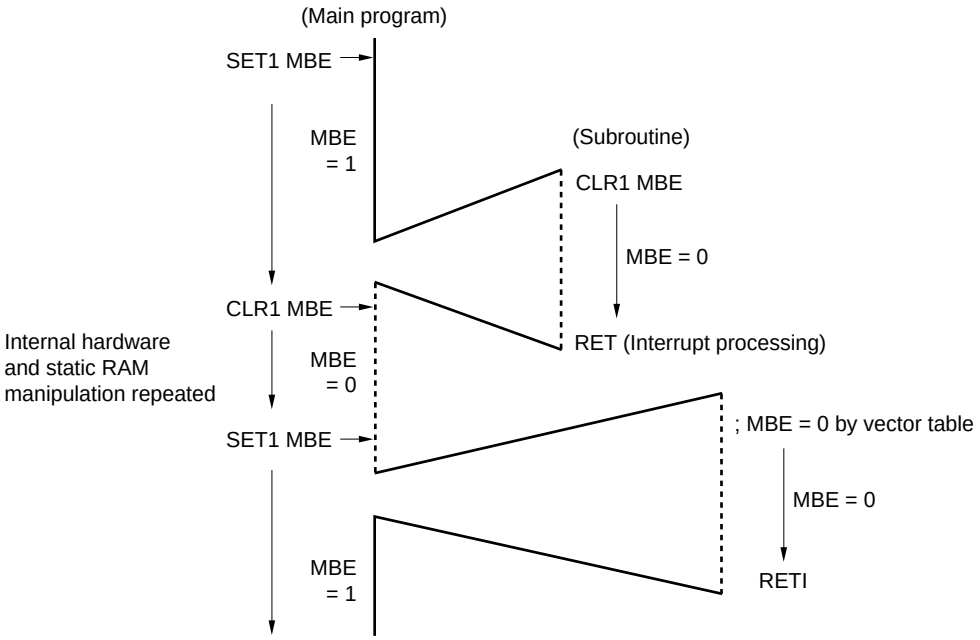
- Memory bank enable flag (MBE)
- Memory bank select register (MBS)

MBS is a register that selects a memory bank. Memory bank 0 or 15 can be selected. MBE is a flag that enables or disables the memory bank selected by MBS. When MBE is 0, the specified memory bank (MB) is fixed, regardless of MBS, as shown in **Figure 3-1**. When MBE is 1, however, a memory bank is selected according to the setting of MBS, so that the data memory space can be expanded.

To address the data memory space, MBE is usually set to 1 and the data memory of the memory bank specified by MBS is manipulated. By selecting a mode of MBE = 0 or a mode of MBE = 1 for each processing of the program, programming can be efficiently carried out.

	Adapted program processing	Effect
MBE = 0 mode	• Interrupt processing	Saving/restoring MBS unnecessary
	• Processing repeating internal hardware manipulation and stack RAM manipulation	Changing MBS unnecessary
	• Subroutine processing	Saving/restoring MBS unnecessary
MBE = 1 mode	• Normal program processing	

Figure 3-1. Selecting MBE = 0 Mode and MBE = 1 Mode



Because MBE is automatically saved or restored during subroutine processing, it can be changed even while subroutine processing is under execution. MBE can also be saved or restored automatically during interrupt processing, so that MBE during interrupt processing can be specified as soon as the interrupt processing is started, by setting the interrupt vector table. This feature is useful for high-speed interrupt processing.

To change MBS by using subroutine processing or interrupt processing, save or restore it to stack by using the PUSH or POP instruction.

MBE is set by using the SET1 or CLR1 instruction. Use the SEL instruction to set MBS.

Examples 1. To clear MBE and fix memory bank

```
CLR1    MBE     ; MBE ← 0
```

2. To select memory bank 1

```
SET1    MBE     ; MBE ← 1
SEL     MB0     ; MBS ← 0
```

3.1.2 Addressing mode of data memory

The 75XL architecture employed for the μPD754308 provides the seven types of addressing modes as shown in **Figures 3-2** and **3-3**, and Table 3-1, so that the data memory space can be efficiently addressed by the bit length of the data to be processed, and so that programming can be carried out efficiently.

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

This mode is for directly addressing each bit of the entire data memory space by using the operand of an instruction.

The memory bank (MB) to be specified is fixed to 0 in the mode of MBE = 0 if the address specified by the operand ranges from 00H to 7FH, and to 15 if the address specified by the operand is 80H to FFH. In the mode of MBE = 0, therefore, both the data area of addresses 000H through 07FH and the peripheral hardware area of F80H through FFFH can be addressed.

In the mode of MBE = 1, MB = MBS; therefore, the entire data memory space can be addressed.

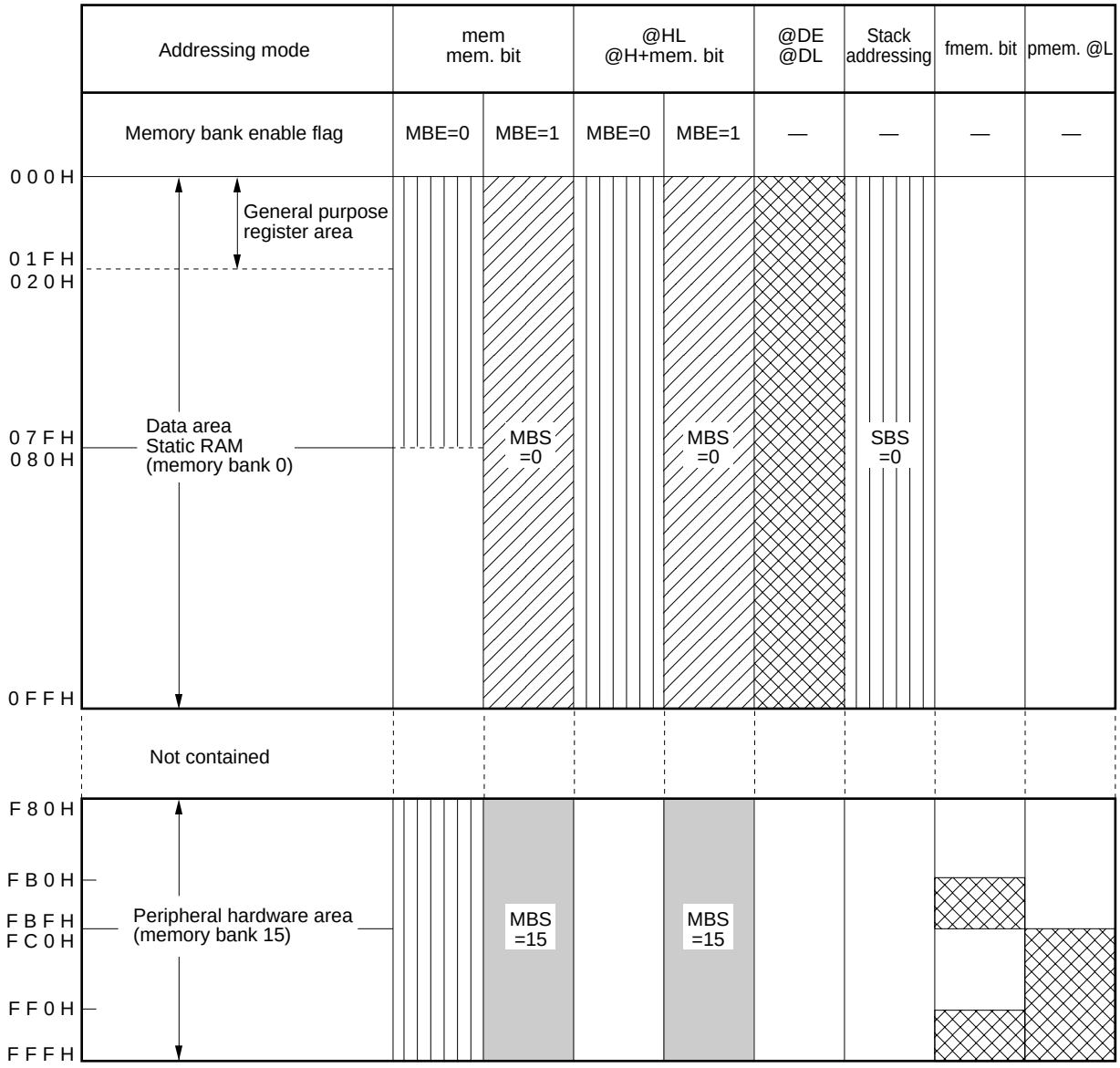
This addressing mode can be used with four instructions: bit set and reset (SET1 and CLR1) instructions, and bit test instructions (SKT and SKF).

Example To set FLAG1, reset FLAG2, and test whether FLAG3 is 0

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

        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 for Each Addressing Mode



Remark – : don't care

Table 3-1. Addressing Mode

Addressing mode	Identifier	Specified address
1-bit direct addressing	mem.bit	Bit of address indicated by MB and mem. The bit is addressed by "bit". • When MBE = 0 when mem = 00H-7FH : MB = 0 when mem = 80H-FFH : MB = 15 • When MBE = 1 : MB = MBS
4-bit direct addressing	mem	Address indicated by MB and mem. • When MBE = 0 when mem = 00H-7FH : MB = 0 when mem = 80H-FFH : MB = 15 • When MBE = 1 : MB = MBS
8-bit direct addressing		Address indicated by MB and mem (mem is an even address). • When MBE = 0 when mem = 00H-7FH : MB = 0 when mem = 80H-FFH : MB = 15 • When MBE = 1 : MB = MBS
4-bit register indirect addressing	@HL	Address indicated by MB and HL. MB = MBE•MBS
	@HL+ @HL–	Address indicated by MB and HL. MB = MBE•MBS When HL+ is given, L register is automatically incremented after addressing. When HL– is given, L register is automatically decremented after addressing.
	@DE	Memory bank 0 address indicated by DE.
	@DL	Memory bank 0 address indicated by DL.
8-bit register indirect addressing	@HL	Address indicated by MB and HL (L register contents are even). MB = MBE•MBS.
Bit manipulation addressing	fmem.bit	Bit of address indicated by fmem. The bit is addressed by "bit". fmem = { FB0H-FBFH (hardware related to interrupt) FF0H-FFFH (I/O port)
	pmem.@L	Bit of address indicated by high-order 10-bit of pmem and high-order 2-bit of L register. The bit is addressed by low-order 2-bit of L register. pmem = FC0H-FFFH
	@H+mem.bit	Bit of address indicated by MB, H, and low-order 4-bit of mem. The bit is addressed by "bit". MB = MBE•MBS
Stack addressing	–	The address indicated by SP of memory banks 0 and 1 selected by setting SBS.

(2) 4-bit direct addressing (mem)

This addressing mode is to directly address the entire memory space in 4-bit units by using the operand of an instruction.

Like the 1-bit direct addressing mode, the area that can be addressed is fixed to the data area of addresses 000H through 07FH and the peripheral hardware area of F80H through FFFH in the mode of MBE = 0. In the mode of MBE = 1, the entire data memory space can be addressed by becoming MB = MBS.

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

Caution If data related to I/O ports is stored to the stack RAM in bank 1 as shown in Example 1 below, the program efficiency is degraded. To program without changing MBS as shown in Example 2, store the data related to I/O ports to the addresses 00H through 7FH of bank 0.

Examples 1. To output data of “BUFF” to port 5

```
BUFF    EQU     90H         ; “BUFF” is at address 90H
        SET1    MBE         ; MBE ← 1
        SEL     MB0         ; MBS ← 0
        MOV     A, BUFF     ; A ← (BUFF)
        SEL     MB15        ; MBS ← 15
        OUT     PORT5, A    ; PORT5 ← A
```

2. To input data from port 5 and store it to “DATA1”

```
DATA1   EQU     5FH         ; “DATA1” is at address 5FH
        CLR1    MBE         ; MBE ← 0
        IN      A, PORT5    ; A ← PORT5
        MOV     DATA1, A    ; (DATA1) ← A
```

(3) 8-bit direct addressing (mem)

This addressing mode is for directly addressing the entire data memory space in 8-bit units by using the operand of an instruction.

The address that can be specified by the operand is an even address. The 4-bit data of the address specified by the operand and the 4-bit data of the the address higher than the specified address are used in pairs and processed in 8-bit units by the 8-bit accumulator (XA register pair).

The memory bank that is addressed is the same as that addressed in the 4-bit direct addressing mode.

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

Examples 1. To transfer the 8-bit data of ports 6 and 7 to addresses 20H and 21

```
DATA    EQU     020H
        CLR1    MBE         ; MBE ← 0
        IN      XA, PORT6   ; X ← port 7, A ← port 6
        MOV     DATA, XA    ; (21H) ← X, (20H) ← A
```

2. To load the 8-bit data input to the shift register (SIO) of the serial interface and, at the same time, set transfer data to instruct the start of transfer

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

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

This addressing mode is for indirectly addressing the data memory space in 4-bit units by using a data pointer (a pair of general-purpose registers) specified by the operand of an instruction.

As the data pointer, three register pairs can be specified: HL that can address the entire data memory space by using MBE and MBS, and DE and DL that always address memory bank 0, regardless of the specification by MBE and MBS. By selecting a register pair depending on the data memory bank to be used, programming can be carried out efficiently.

Example To transfer data 50H through 57H to addresses 10H through 17H

```
DATA1   EQU   57H
DATA2   EQU   17H
        SET1  MBE                  ; MBE ← 1
        SEL   MB0                  ; MBS ← 0
        MOV   D, #DATA1 SHR 4
        MOV   HL, #DATA2 AND 0FFH  ; HL ← 17H
LOOP:   MOV   A, @DL               ; A ← (DL)
        XCH   A, @HL–              ; A ← (HL), L ← L –1
        BR    LOOP
```

The addressing mode that uses register pair HL as the data pointer is widely used to transfer, operate, compare, and input/output data. The addressing mode using register pair DE or DL is used with the MOV and XCH instructions.

By using this addressing mode in combination with the increment/decrement instruction of a general-purpose register or a register pair, the addresses of the data memory can be updated as shown in **Figure 3-3**.

Examples 1. To compare data 50H through 57H with data 10H through 17H

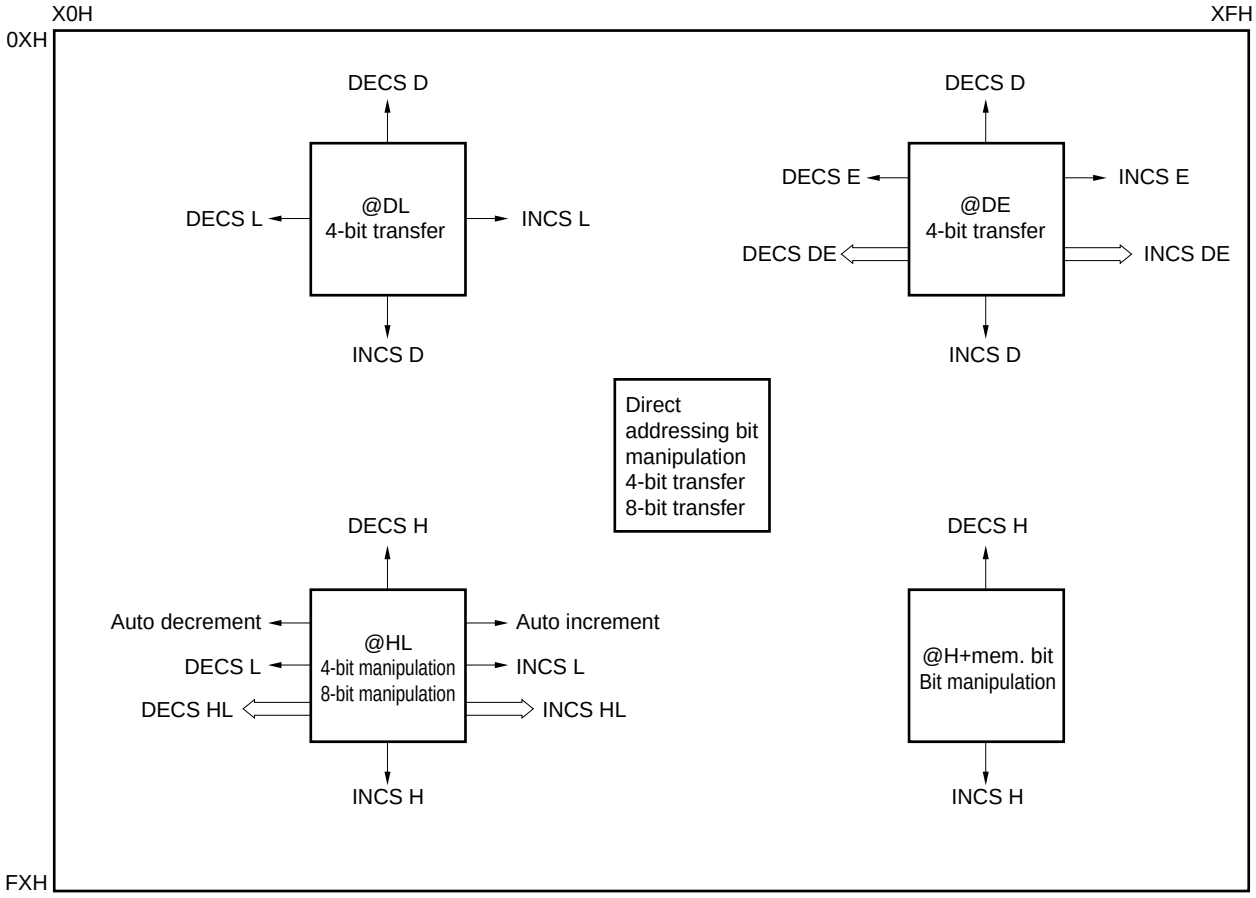
```
DATA1   EQU   57H
DATA2   EQU   17H
        SET1  MBE
        SEL   MB0
        MOV   D, #DATA1 SHR 4
        MOV   HL, #DATA2 AND 0FFH
LOOP:   MOV   A, @DL
        SKE   A, @HL        ; A = (HL)?
        BR    NO            ; NO
        DECS  L             ; YES, L ← L – 1
        BR    LOOP
```

2. To clear data memory of 00H through FFH

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

Note Since data memory addresses 00H to 03H are used as the general-purpose registers XA and HL, they are not cleared.

Figure 3-3. Static RAM Address Update Method



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

This addressing mode is to indirectly address the entire data memory space in 8-bit units by using a data pointer (HL register pair).

In this addressing mode, data is processed in 8-bit units, that is, the 4-bit data at an address specified by the data pointer with bit 0 (bit 0 of the L register) cleared to 0 and the 4-bit data at the address higher are used in pairs and processed with the data of the 8-bit accumulator (XA register).

The memory bank to be specified turns MB = MBE • MBS, which is the same case the HL register is specified in the 4-bit register indirect addressing mode. This addressing mode is applicable to the MOV, XCH, and SKE instructions.

Examples 1. To compare whether the count register (T0) value of timer/event counter 0 is equal to the data at addresses 30H and 31H

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

2. To clear data memory at 00H through FFH

```
        CLR1   RBE
        CLR1   MBE
        MOV    XA, #00H
        MOV    HL, #04H
LOOP:   MOV    @HL, A        ; (HL) ← A, Clear 04H to FFH Note
        INCS   HL
        NCS    HL
        BR     LOOP
```

Note Since the data memory addresses 00H to 03H are used as the general-purpose registers XA and HL, they are not cleared.

(6) Bit manipulation addressing

This addressing mode is used to perform the bit manipulation to each bit in the entire memory space (such as Boolean processing and bit transfer).

While the 1-bit direct addressing mode can be only used with the instructions that set, reset, or test a bit, this addressing mode can be used in various ways, such as Boolean processing by the AND1, OR1, and XOR1 instructions, bit transfer by the MOV1 instruction, and test and reset by the SKTCLR instruction.

Bit manipulation addressing can be implemented in the following three ways, which can be selected depending on the data memory address to be used.

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

This addressing mode is to manipulate the hardware units that use bit manipulation especially often, such as I/O ports and interrupt-related flags, regardless of the setting of the memory bank. Therefore, the data memory addresses to which this addressing mode is applicable are FF0H through FFFH, to which the I/O ports are mapped, and FB0H through FBFH, to which the interrupt-related hardware units are mapped. The hardware units in these two data memory areas can be manipulated in bit units at any time in the direct addressing mode, regardless of the setting of MBS and MBE.

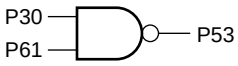
Example 1. Invert the input at pin P02 and output it from pin P33

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

2. To test timer 0 interrupt request flag (IRQT0) and, if it is set, clear the flag and reset P63

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

3. To reset P53 if both P30 and P61 pins are 1




```
MOV1        CT, PORT3.0     ; CY ← P30
AND1        CY, PORT6.1     ; CY ∧ P61
NOT1        CY              ; CY ← CY
MOV1        PORT5.3, CY     ; P53 ← CY
```

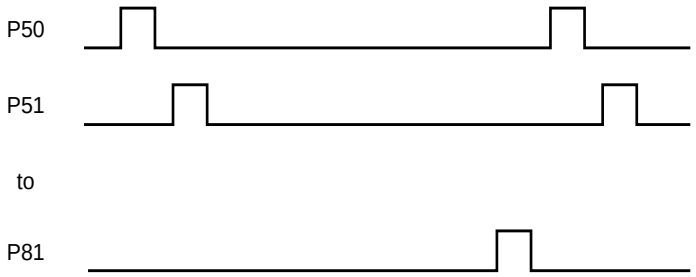
(b) Specific address bit register indirect addressing (pmem, @L)

This addressing mode is used to indirectly specify and successively manipulate the bits of the peripheral hardware units, such as I/O ports. The data memory addresses to which this addressing mode can be applied are FC0H through FFFH.

This addressing mode specifies the higher 10 bits of a data memory address directly by using an operand, and the lower 2 bits by using the L register. Therefore, 16 bits (4 ports) can be successively manipulated depending on the specification of the L register.

This addressing mode can also be used independently of the setting of MBE and MBS.

Example To output pulses to the respective bits (14 bits) of ports 5 to 8



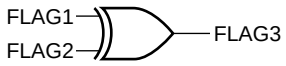

```
        MOV     L, #0100B
LOOP:   SET1    PORT5.@L    ; Bits of ports 5 to 8 (L1-0) ← 1
        CLR1    PORT5.@L    ; Bits of ports 5 to 8 (L1-0) ← 0
        INCS    L
        NOP
        BR      LOOP
```

(c) Special 1-bit direct addressing (@H+mem, bit)

This addressing mode enables bit manipulation in the entire memory space.

The higher 4 bits of the data memory address of the memory bank specified by MB = MBE·MBS are indirectly specified by the H register, and the lower 4 bits and the bit address are directly specified by the operand. This addressing mode can be used to manipulate the respective bits of the entire data memory area in various ways.

Example To reset bit 2 (FLAG3) at address 32H if both bits 3 (FLAG1) at address 30H and bit 0 (FLAG2) at address 31H are 0 or 1




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

(7) Stack addressing

This addressing mode is used to save or restore data when interrupt processing or subroutine processing is executed.

The address of data memory bank 0 pointed to by the stack pointer (8 bits) is specified in this addressing mode. This addressing is also used to save or restore register contents by using the PUSH or POP instruction, in addition to during interrupt processing or subroutine processing.

Examples 1. To save or restore register contents during subroutine processing

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

2. To transfer contents of register pair HL to register pair DE

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

3. To branch to address specified by registers [XABC]

```
        PUSH    BC
        PUSH    XA
        RET                 ; To branch address XABC
```

3.2 Bank Configuration of General-Purpose Registers

The μPD754304 is provided with four register banks with each bank consisting of eight general-purpose registers: X, A, B, C, D, E, H, and L. The general-purpose register area consisting of these registers is mapped to the addresses 00H through 1FH of memory bank 0 (refer to **Figure 3-5. General-Purpose Register Structure (4-bit processing**). To specify a general-purpose register bank, a register bank enable flag (RBE) and a register bank select register (RBS) are provided. RBS selects a register bank, and RBE determines whether the register bank selected by RBS is valid or not. The register bank (RB) that is enabled when an instruction is executed is as follows:

RB = RBE·RBS

Table 3-2. Register Bank Selected by RBE and RBS

RBE	RBS				Register bank
	3	2	1	0	
0	0	0	×	×	Fixed to bank 0
1	0	0	0	0	Bank 0 selection
			0	1	Bank 1 selection
			1	0	Bank 2 selection
			1	1	Bank 3 selection

Fixed to 0

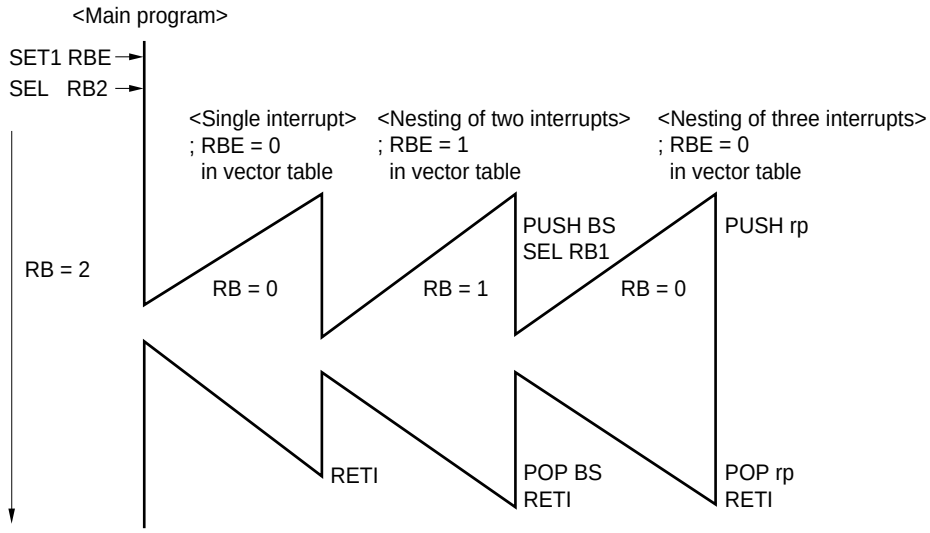
Remark × = don't care

RBE is automatically saved or restored during subroutine processing, and therefore can be set while subroutine processing is under execution. When interrupt processing is executed, RBE is automatically saved or restored, and RBE can be set during interrupt processing depending on the setting of the interrupt vector table as soon as the interrupt processing is started. Consequently, if different register banks are used for normal processing and interrupt processing as shown in **Table 3-3**, it is not necessary to save or restore general-purpose registers when an interrupt is processed, and only RBS needs to be saved or restored if two interrupts are nested, so that the interrupt processing speed can be increased.

Table 3-3. Example of Using Different Register Banks for Normal Routine and Interrupt Routine

Normal processing	Uses register banks 2 or 3 with RBE = 1
Single interrupt processing	Uses register bank 0 with RBE = 0
Nesting processing of two interrupts	Uses register bank 1 with RBE = 1 (at this time, RBS must be saved or restored)
Nesting processing of three or more interrupts	Registers must be saved or restored by PUSH or POP instructions

Figure 3-4. Example of Using Register Banks



If RBS is to be changed in the course of subroutine processing or interrupt processing, it must be saved or restored by using the PUSH or POP instruction.

RBE is set by using the SET1 or CLR1 instruction. RBS is set by using the SEL instruction.

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

The general-purpose register area provided to the μPD754304 can be used not only as 4-bit registers, but also as 8-bit register pairs. This feature allows the μPD754304 to provide transfer, operation, comparison, and increment/decrement instructions comparable to those of 8-bit microcomputers and allows you to program mainly with general-purpose registers.

(1) To use as 4-bit registers

When the general-purpose register area is used as a 4-bit register area, a total of eight general-purpose registers, X, A, B, C, D, E, H, and L, specified by RBE and RBS can be used as shown in **Figure 3-5**. Of these registers, A plays a central role in transferring, operating, and comparing 4-bit data as a 4-bit accumulator. The other registers can transfer, compare, and increment or decrement data with the accumulator.

(2) To use as 8-bit registers

When the general-purpose register area is used as an 8-bit register area, a total of eight 8-bit register pairs can be used as shown in **Figure 3-6**: register pairs XA, BC, DE, and HL of a register bank specified by RB = RBE·RBS, and register pairs XA', BC', DE', and HL' of the register bank whose bit 0 is complemented in respect to the register bank (RB). Of these register pairs, XA serves as an 8-bit accumulator, playing the central role in transferring, operating, and comparing 8-bit data. The other register pairs can transfer, compare, and increment or decrement data with the accumulator. The HL register pair is mainly used as a data pointer. The DE and DL register pairs are also used as auxiliary data pointers.

Examples 1.

```
INCS    HL          ; Skips if HL ← HL+1, HL=00H
ADDS    XA, BC      ; Skips if XA ← XA+BC, carry
SUBC    DE', XA     ; DE' ← DE' – XA – CY
MOV     XA, XA'     ; XA ← XA'
MOVT    XA, @PCDE   ; XA ← (PC12-8+DE)ROM, table reference
SKE     XA, BC      ; Skips if XA = BC
```

2. To test whether the value of the count register (T0) of timer/event counter is greater than the value of register pair BC' and, if not, wait until it becomes greater

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

Figure 3-5. General-Purpose Register Configuration (for 4-bit operation)

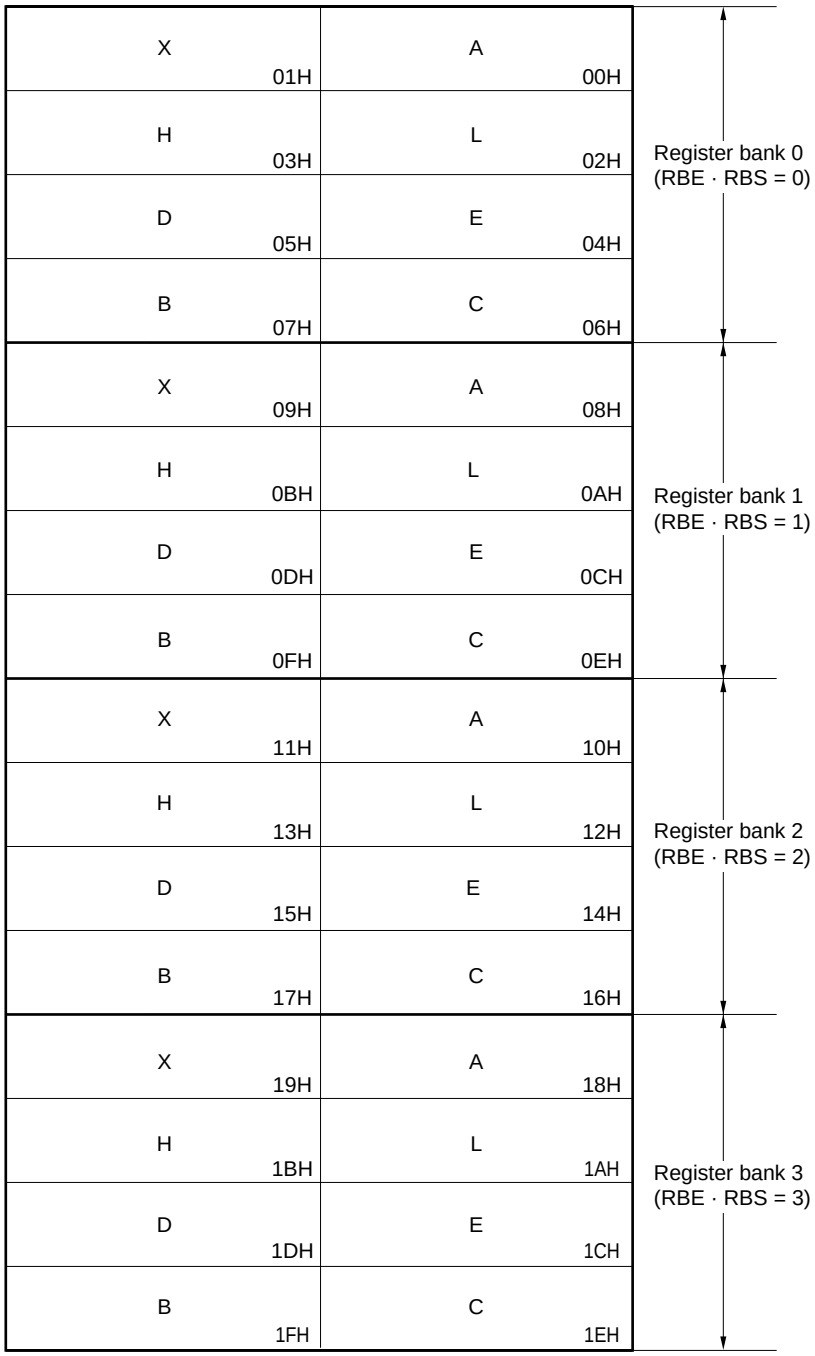
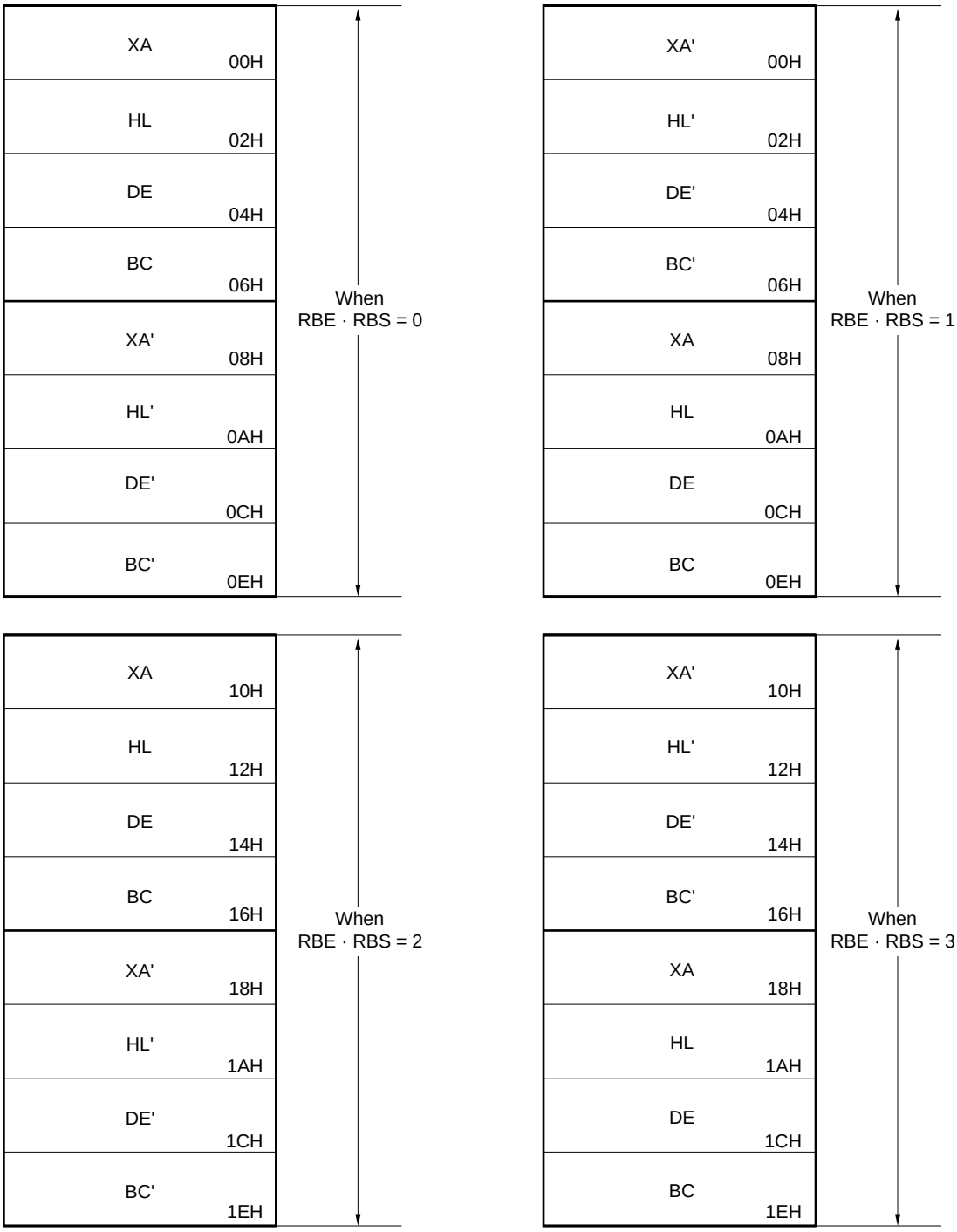


Figure 3-6. General-Purpose Register Configuration (for 8-bit operation)



3.3 Memory-Mapped I/O

The μPD754304 employs memory-mapped I/O where peripheral hardware such as the input/output ports and timers are mapped in data memory space addresses F80H-FFFH, as shown in **Figure 3-2**. Thus, special instructions to control the peripheral hardware are not provided and memory manipulation instructions are all used to control the peripheral hardware (Some hardware control mnemonics are provided for easy understanding of programs).

To manipulate the peripheral hardware, the addressing modes listed in **Table 3-4** can be used.

Table 3-4. Addressing Modes Applicable to Operating the Peripheral Hardware

	Applicable addressing mode	Applicable hardware
Bit manipulation	Specified by a direct addressing mem.bit with MBE = 0 or (MBE = 1, MBS = 15).	All the hardware for which bit manipulation is possible
	Specified by direct addressing fmem.bit regardless of MBE and MBS.	IST1, IST0, MBE, RBE IEXXX, IRQXXX, PORTn.X
	Specified by indirect addressing pmem.@L regardless of MBE and MBS.	BSBn.X PORTn.X
4-bit manipulation	Specified by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15).	All the hardware for which 4-bit manipulation is possible
	Specified by register indirect addressing @HL with (MBE = 1, MBS = 15).	
8-bit manipulation	Specified by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15). Note that mem must be an even-number address.	All the hardware for which 8-bit manipulation is possible
	Specified by register indirect addressing @HL with (MBE = 1, MBS = 15). Note that the contents of the L register are an even number.	

Example
```
CLR1    MBE      ; MBE = 0
SET1    TM0.3    ; Starts timer 0
EI      IE0      ; Enables INT0
DI      IE1      ; Disables INT1
SKTCLR  IRQ2     ; Tests and clears INT2 request flag
SET1    PORT5.@L ; Sets port 5
```

The I/O map of the μPD754304 is shown in **Figure 3-7**.
The meanings of the items in **Figure 3-7** are as follows.

- Hardware name (symbol) ····· Aname indicating the address of on-chip hardware. Can be described in the operand column of instruction.

- R/W ····· Indicates whether the given hardware is read/write enabled or not.
  R/W : read/write enabled
  R : read only
  W : write only

- Manipulation unit ····· Indicates the number of bits in which the hardware device can be manipulated.
  Yes : Bit manipulation is possible in the unit (1/4/8 bits) used in the column.
  Δ : A part of bits can be manipulated. Refer to “Remarks” for the bits that can be manipulated.
  — : Bit manipulation is impossible in the unit (1/4/8 bits) used in the column.

- Bit manipulation addressing ····· Indicates the usable bit manipulation addressing when bit manipulation is performed on the hardware.

Figure 3-7. μPD754304 I/O Map (1/5)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
F80H	Stack pointer (SP)				R/W	–	–	Yes	–	Bit 0 is fixed to 0.
F82H	Register bank selection register (RBS)				R	–	Yes	Yes	–	**Note 1**
	Bank selection register (BS)									
F83H	Memory bank selection register (MBS)					–	Yes			
F84H	Stack bank selection register (SBS)				R/W	–	Yes	–	mem.bit	
F85H	Basic interval timer mode register (BTM)				W	Δ	Yes	–	mem.bit	Bit manipulation can be performed only on bit 3.
F86H	Basic interval timer (BT)				R	–	–	Yes	–	
F8BH	WDTM Note 2				W	Yes	–	–	mem.bit	Bit manipulation can be performed only on bit 3.

Notes 1. The manipulation is possible separately with RBS and MBS in the 4-bit manipulation.
The manipulation is possible with BS in the 8-bit manipulation.
Write data in the MBS and RBS with the SEL MBn and SEL RBn instructions.

2. WDTM: Watchdog Timer Enable flag (W); Cannot be cleared, once set, by an instruction.

Figure 3-7. μPD754304 I/O Map (2/5)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FA0H	Timer/event counter 0 mode register (TM0)				R/W	Δ (W)	–	Yes	mem.bit	Bit manipulation can be performed only on bit 3
						–	–	(R/W)	–	
FA2H	TOE0 Note 1				W	Yes	–	–	mem.bit	
FA4H	Timer/event counter 0 count register (T0)				R	–	–	Yes	–	
FA6H	Timer/event counter 0 modulo register (TMOD0)				R/W	–	–	Yes	–	
FA8H	Timer/event counter 1 mode register (TM1)				R/W	Δ (W)	–	Yes	mem.bit	Bit manipulation can be performed only on bit 3
						–	–	(R/W)	–	
FAAH	TOE1 Note 2				W	Yes	–	–	mem.bit	
FACH	Timer/event counter 1 count register (T1)				R	–	–	Yes	–	
FAEH	Timer/event counter 1 modulo register (TMOD1)				R/W	–	–	Yes	–	

Notes 1. TOE0: Timer/event counter 0 output enable flag (W)

2. TOE1: Timer/event counter 1 output enable flag (W)

Figure 3-7. μPD754304 I/O Map (3/5)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FB0H	IST1	IST0	MBE	RBE	R/W	Yes(R/W)	Yes(R/W)	Yes	fmem.bit	8-bit manipulation is read only.
	Program status word (PSW)									
	CY	SK2	SK1	SK0		–	–	(R)		
FB2H	Interrupt priority selection register (IPS)				R/W	–	Yes	–		**Note 1**
FB3H	Processor clock control register (PCC)				R/W	–	Yes	–		**Note 2**
FB4H	INT0 edge detection mode register (IM0)				R/W	–	Yes	–	–	
FB5H	INT1 edge detection mode register (IM1)				R/W	–	Yes			Bits 3, 2, and 1 fixed at 0.
FB6H	INT2 edge deection mode register (IM2)				R/W	–	Yes			Bits 3 and 2 fixed at 0.
FB8H	INTA register (INTA)				R/W	Yes	Yes	–	fmem.bit	
	IE4	IRQ4	IEBT	IRQBT						
FBAH	INTE register (INTE)				R/W	Yes	Yes			
	IET1	IRQT1	IET0	IRQT0						
FBDH	INTF register (INTF)				R/W	Yes	Yes			
	IET2	IRQT2	IECSI	IRQCSI						
FBEH	INTG register (INTG)				R/W	Yes	Yes	–		
	IE1	IRQ1	IE0	IRQ0						
FBFH	INTH register (INTH)				R/W	Yes	Yes			
			IE2	IRQ2						

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

Remarks 1. IEXXX : Interrupt enable flag

2. IRQXXX: Interrupt request flag

Notes 1. Only bit 3 can be manipulated with an EI/DI instruction.

2. Bits 3 and 2 can be manipulated bitwise when a STOP or HALT instruction is executed.

Figure 3-7. μPD754304 I/O Map (4/5)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FD0H	Clock output mode register (CLOM)				W	–	Yes	–	–	
FDCH	Pull-up resistor specification register group A (POGA)				R/W	–	–	Yes	–	
FDEH	Pull-up resistor specification register group B (POGB)				R/W	–	–	Yes	–	

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FE0H	Serial operation mode register (CSIM)				W	–	–	Yes	–	
★	CSIE	COI				Yes	–		mem.bit	
FE2H			CMDT	RELT	W	Yes	–	–	mem.bit	
	SBI control register (SBIC)									
FE4H	Serial I/O shift register (SIO)				R/W	–	–	Yes	–	
★ FE6H	Slave address register (SVA)				R/W	–	–	Yes	–	
FE8H	PM33	PM32	PM31	PM30	R/W	–	–	Yes	–	
	Port mode register group A (PMGA)									
	PM63	PM62	PM61	PM60						
FECH	–	PM2	–	–	R/W	–	–	Yes	–	
	Port mode register group B (PMGB)									
	PM7	–	PM5	–						
FEEH	–	–	–	PM8	R/W	–	–	Yes	–	
	Port mode register group C (PMGC)									
	–	–	–	–						

Figure 3-7. μPD754304 I/O Map (5/5)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FF0H	Port 0 (PORT0)				R	Yes	Yes	–	fmem.bit pmem.@L	
FF1H	Port 1 (PORT1)				R	Yes	Yes			
FF2H	Port 2 (PORT2)				R/W	Yes	Yes	–		
FF3H	Port 3 (PORT3)				R/W	Yes	Yes			
FF5H	Port 5 (PORT5)				R/W	Yes	Yes	–		
FF6H	KR3	KR2	KR1	KR0	R/W	Yes	Yes	Yes		
	Port 6 (PORT6)									
FF7H	KR7	KR6	KR5	KR4	R/W	Yes	Yes			
	Port 7 (PORT7)									
FF8H	Port 8 (PORT8)				R/W	Yes	Yes	–		

[MEMO]

CHAPTER 4 INTERNAL CPU FUNCTIONS

4.1 Switching Function between Mk I Mode and Mk II Mode

4.1.1 Difference between Mk I and Mk II modes

The CPU of μPD754304 has the following two modes: Mk I and Mk II, either of which can be selected. The mode can be switched by the bit 3 of the Stack Bank Select register (SBS).

- Mk I mode : Can be used in the 75XL CPU with a ROM capacity of up to 16 Kbytes.
- Mk II mode : Can be used in all the 75XL CPU's including those devices whose ROM capacity is more than 16 Kbytes.

Table 4-1 lists the differences between the Mk I and Mk II modes.

Table 4-1. Differences between Mk I Mode and Mk II Mode

	Mk I mode	Mk II mode
Number of stack bytes for subroutine instructions	2 bytes	3 bytes
BRA !addr1 instruction CALLA !addr1 instruction	Not available	Available
MOVT XA, @BCXA instruction MOVT XA, @BCDE instruction BR BCXA instruction BR BCDE instruction	Available	Available
CALL !addr instruction	3 machine cycles	4 machine cycles
CALLF !faddr instruction	2 machine cycles	3 machine cycles

Caution Mk II mode is for maintaining a software compatibility with devices in the 75X series or 75XL series whose program memory is more than 24 Kbytes. Therefore, Mk I mode is recommended for applications with a focus on the ROM efficiency or speed.

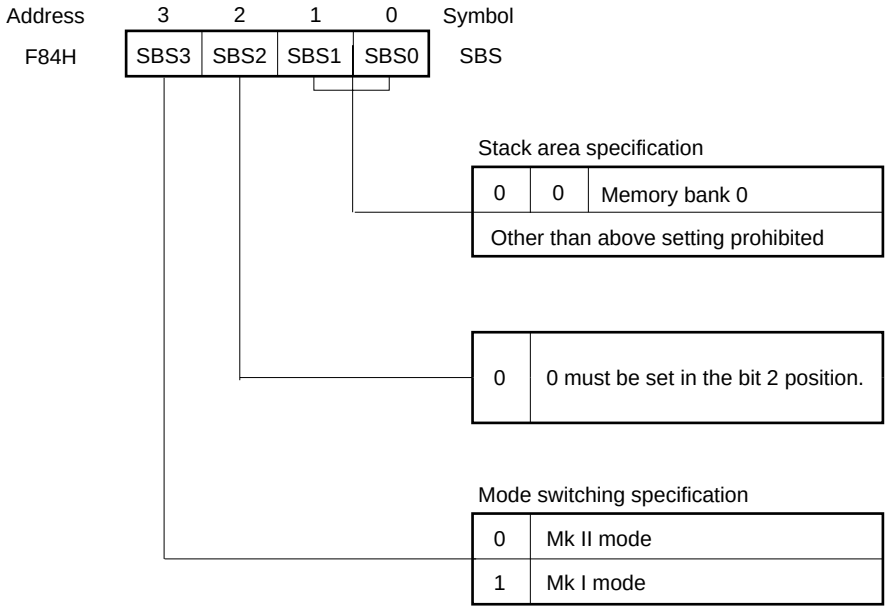
4.1.2 Setting method of Stack Bank Select register (SBS)

Switching between the Mk I mode and Mk II mode can be done by the SBS. **Figure 4-1** shows the format.

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

When using the Mk I mode, the SBS must be initialized to 1000B at the beginning of a program. When using the Mk II mode, it must be initialized to 0000B.

Figure 4-1. Stack Bank Select Register Format



Caution Because SBS. 3 is set to "1" after a $\overline{\text{RESET}}$ signal is generated, the CPU operates in the Mk I mode. When executing an instruction in the Mk II mode, set SBS. 3 to "0" to select the Mk II mode.

4.2 Program Counter (PC) ····· 11-bit (μPD754302)
12-bit (μPD754304)
13-bit (μPD75P4308)

This is a binary counter that holds an address of the program memory.

Figure 4-2. Program Counter Structure

(a) μPD754302

PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

(b) μPD754304

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

(c) μPD75P4308

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

The value of the program counter (PC) is usually automatically incremented by the number of bytes of an instruction each time an instruction has been executed.

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

When a subroutine call instruction (CALL, CALLA, or CALLF) is executed or when a vector interrupt occurs, the contents of the PC (a return address already incremented to fetch the next instruction) are saved to the stack memory (data memory specified by the stack pointer) and then the jump destination address is loaded to the PC.

When the return instruction (RET, RETS, or RETI) is executed, the contents of the stack memory are set to the PC.

When the $\overline{\text{RESET}}$ signal is asserted, the program counter (PC) is loaded with the contents of addresses 000H and 001H in the program memory as shown below. The program can start from the arbitrary address according to the contents of the 000H and 001H addresses.

μPD754302:
PC10 - PC8 ← (Program memory 000H)$_{2\text{-}0}$
PC7 - PC0 ← (Program memory 001H)$_{7\text{-}0}$
μPD754304:
PC11 - PC8 ← (Program memory 000H)$_{3\text{-}0}$
PC7 - PC0 ← (Program memory 001H)$_{7\text{-}0}$
μPD75P4308:
PC12 - PC8 ← (Program memory 000H)$_{4\text{-}0}$
PC7 - PC0 ← (Program memory 001H)$_{7\text{-}0}$

4.3 Program Memory (ROM) ····· 2048× 8 bits (μPD754302: Mask ROM)
4096 × 8 bits (μPD754304: Mask ROM)
8192 × 8 bits (μPD75P4308: One-time PROM)

The program memory is provided to store the programs, interrupt vector table, reference table of the GETI instruction, and table data.

It is addressed by the program counter. Table data can be referenced by the Table Reference instruction (MOVT).

The range of addresses to which branches can be taken by a Branch instruction and Subroutine Call instruction is shown in **Figure 4-3**. A branch can take place to address (contents of PC–15 to –1, +2 to +16) by a Relative Branch instruction (BR $addr instruction).

The address range of the program memory of each model is as follows:

- 0000H-07FFH: μPD754302
- 0000H-0FFFH: μPD754304
- 0000H-1FFFH: μPD75P4308

Special functions are assigned to the following addresses. All the addresses other than 0000H and 0001H can be usually used as program memory addresses.

- Addresses 0000-0001H
  Vector table wherein the program start address and the values set for the RBE and MBE at the time a $\overline{\text{RESET}}$ signal is generated are written. Reset and start are possible at an arbitrary address.

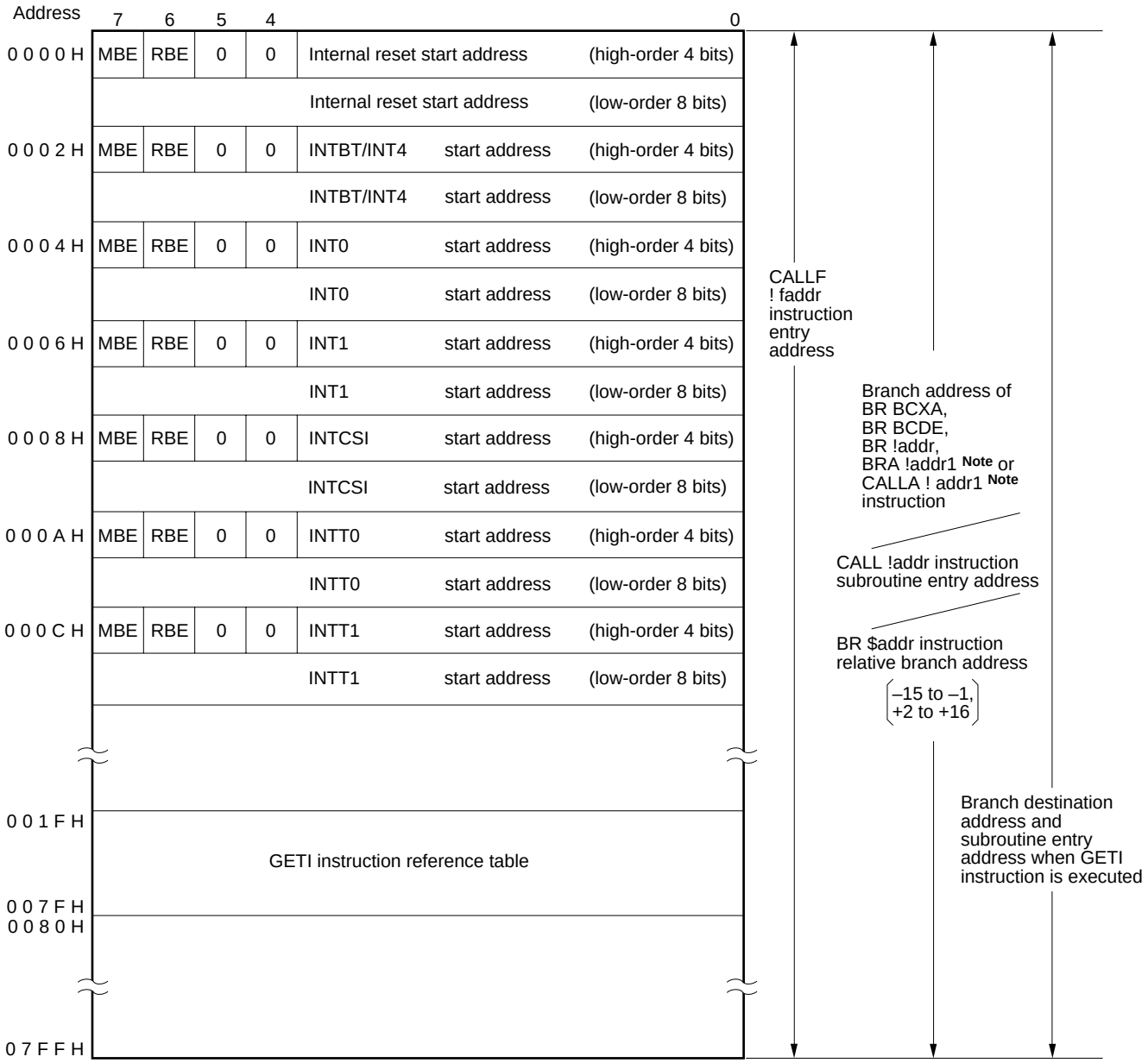
- Addresses 0002-000DH
  Vector table wherein the program start address and values set for the RBE and MBE by the vectored interrupts are written. Interrupt execution can be started at an arbitrary address.

- Addresses 0020-007FH
  Table area referenced by the GETI instruction. **Note**

Note The GETI instruction realizes a 1-byte instruction on behalf of an arbitrary 2-byte instruction, 3-byte instruction, or two 1-byte instructions. It is used to decrease the program steps (See **11.1.1 GETI instruction**).

Figure 4-3. Program Memory Map (1/3)

(a) μPD754302

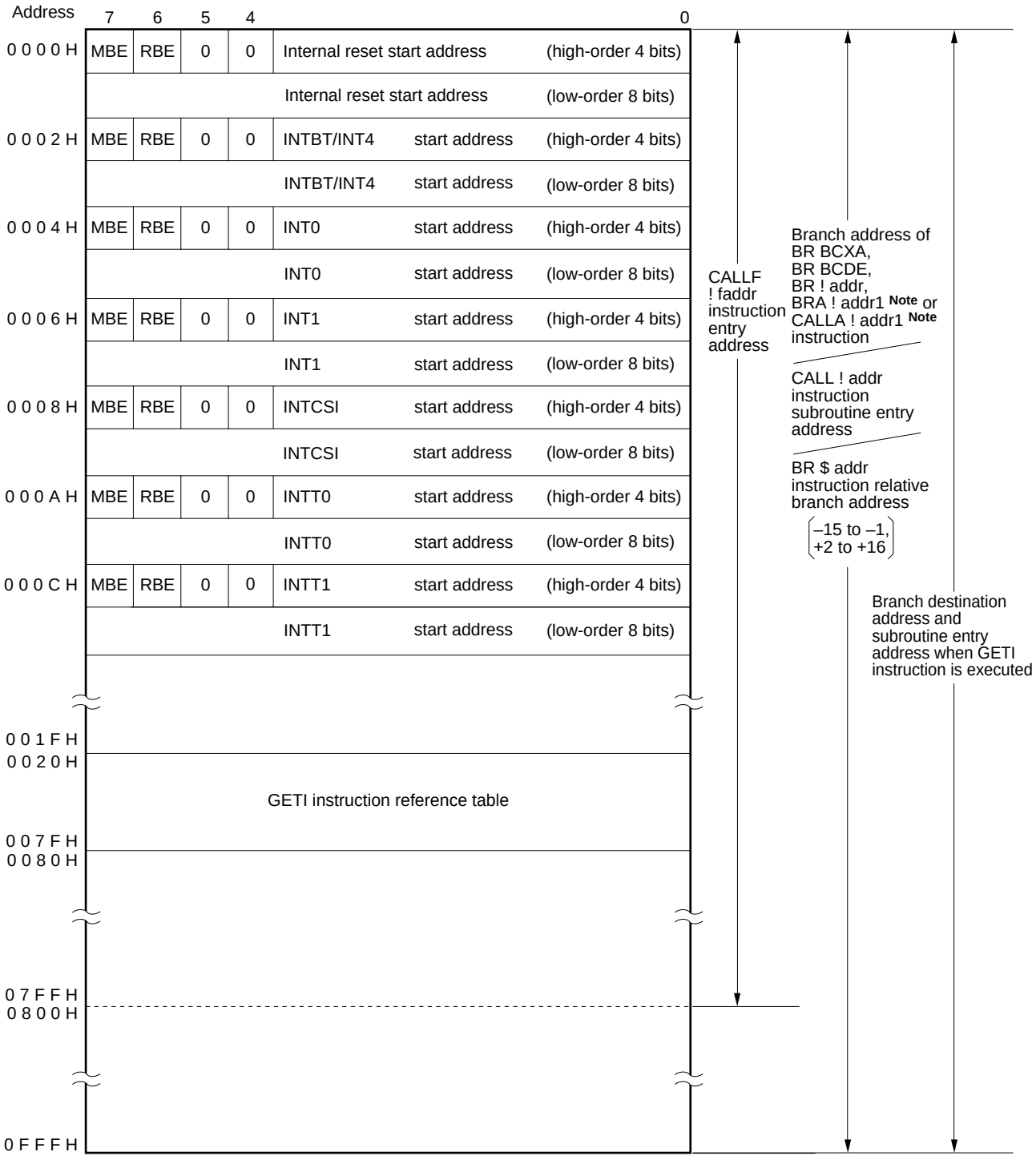


★ **Note** The BRA !addr1 and CALLA !addr1 instructions can be used only in the Mk II mode.

Remark In addition to the above, a branch can be taken to the address indicated by changing only the low-order eight bits of PC by executing the BR PCDE or BR PCXA instruction.

Figure 4-3. Program Memory Map (2/3)

(b) μPD754304

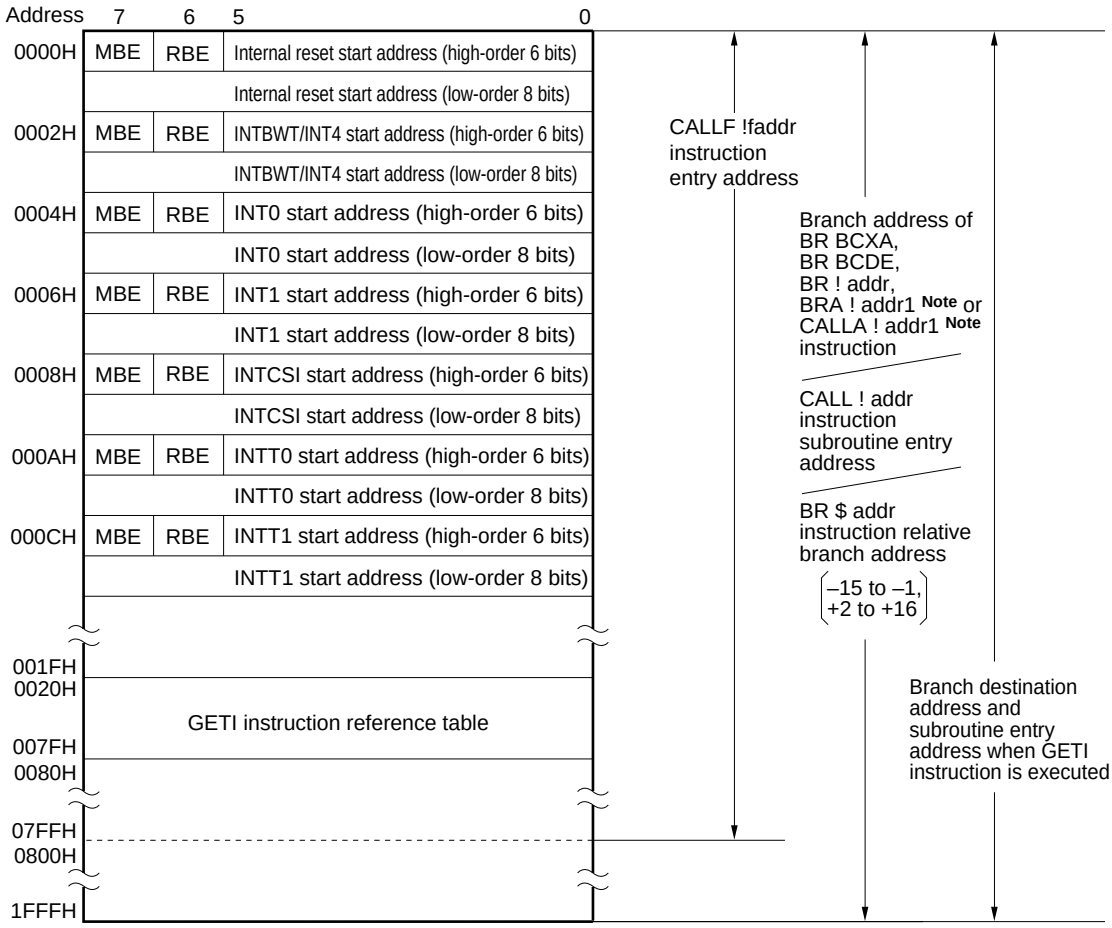


Note The BRA !addr1 and CALLA !addr1 instructions can be used only in the Mk II mode. ★

Remark In addition to the above, a branch can be taken to the address indicated by changing only the low-order eight bits of PC by executing the BR PCDE or BR PCXA instruction.

Figure 4-3. Program Memory Map (3/3)

(c) μPD75P4308



★ **Note** BRA !addr1 and CALLA !addr1 instructions can be used only in the Mk II mode.

Remark In addition to the above, a branch can be taken to the address indicated by changing only the low-order eight bits of PC by executing the BR PCDE or BR PCXA instruction.

4.4 Data Memory (RAM) ··· 256 words × 4 bits

The data memory consists of data areas and a peripheral hardware area as shown in **Figure 4-7**.
The data memory consists of the following banks, with each bank made up of 256 words × 4 bits:

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

4.4.1 Configuration of data memory

(1) Data area

A data area consists of static RAM, and is used to store data and as a stack memory when a subroutine or interrupt is executed. The contents of this area can be backed up for a long time by batteries even when the CPU is stopped in the standby mode. The data area is manipulated by using memory manipulation instructions.
Static RAM is mapped to memory banks 0 in units of 256 × 4 bits. Although bank 0 is mapped as a data area, it can also be used as a general-purpose register area (000H through 01FH) and as a stack area (000H through 0FFH).
One address of the static RAM consists of 4 bits. However, it can be manipulated in 8-bit units by using an 8-bit memory manipulation instruction, or in 1-bit units by using a bit manipulation instruction. To use an 8-bit manipulation instruction, specify an even address.

- **General-purpose register area**
  This area can be manipulated by using a general-purpose register manipulation instruction or memory manipulation instruction. Up to eight 4-bit registers can be used. The registers not used by the program can be used as part of the data area or stack area. (See **4.5. General-Purpose Registers**.)

- **Stack area**
  The stack area is set by an instruction and is used as a saving area when a subroutine or interrupt processing is executed. (See **4.7 Stack Pointer (SP) and Stack Bank Selection Register (SBS)**.)

(2) Peripheral hardware area

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

This area is manipulated by using a memory manipulation instruction, in the same manner as the static area. Note, however, that the bit units in which the peripheral hardware units can be manipulated differ depending on the address. The addresses to which no peripheral hardware unit is allocated cannot be accessed because these addresses are not provided to the data memory. (See **Figure 3-7. μPD754302 I/O Map**)

4.4.2 Specifying bank of data memory

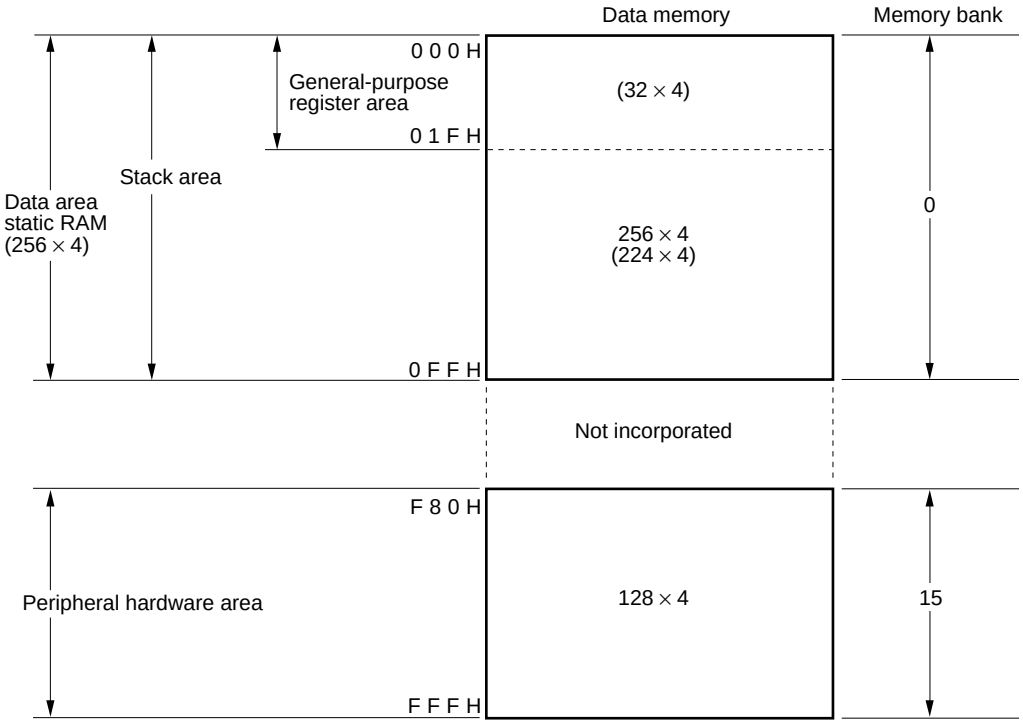
A memory bank is specified by setting a 4-bit memory bank select register (MBS) to 0 or 15 when bank specification is enabled by setting a memory bank enable flag (MBE) to 1. When bank specification is disabled (MBE = 0), bank 0 or 15 is automatically specified depending on the addressing mode selected at that time. The addresses in the bank are specified by 8-bit immediate data or a register pair.

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

For how to use a specific area of the data memory, refer to the following chapter or sections:

- General-purpose register area ..... See **4.5 General-Purpose Registers**
- Stack area ..... See **4.7 Stack Pointer (SP) and Stack Bank Selection Register (SBS)**.
- Peripheral hardware area ..... See **CHAPTER 5 PERIPHERAL HARDWARE FUNCTION.**

Figure 4-4. Data Memory Map



The contents of the data memory are undefined at reset. Therefore, they must be initialized at the beginning of program execution (RAM clear). Otherwise, unexpected bugs may occur.

Example To clear RAM at addresses 000H through 0FFH

```
        SET1    MBE
        SEL     MB0
        MOV     XA, #00H
        MOV     HL, #04H
RAMC0:  MOV     @HL, A      ; Clears 04H-FFH Note
        INCS    L           ; L ← L+1
        BR      RAMC0
        INCS    H           ; H ← H+1
        BR      RAMC0
```

Note Since the data memory address 000H to 003H are used as the general-purpose registers XA and HL, they are not cleared.

4.5 General-Purpose Registers ··· 8× 4 bits × 4 banks

The general-purpose registers are mapped in specific addresses of the data memory. There are four registers banks each consisting of eight 4-bit registers (B, C, D, E, H, L, X, and A).

The register bank (RB) which becomes valid during instruction execution is determined by the following expression:

RB = RBE•RBS (RBS = 0 to 3)

Each general-purpose register is manipulated in 4-bit units. In addition, register pairs BC, DE, HL, and XA can also be used for 8-bit manipulation. The DL register can also be paired as well as DE and HL; these three register pairs can be used as data pointers.

When two general-purpose registers are manipulated in 8-bit units, register pairs BC', DE', HL', and XA' of the register bank (0 ↔ 1, 2 ↔ 3) specified by the complement of bit 0 of the register bank (RB) can be used, in addition to BC, DE, HL, and XA (refer to **3.2 Bank Configuration of General-Purpose Registers**).

The general purpose register area can be addressed as normal RAM for an access regardless of whether or not the area is used as registers.

Figure 4-5. General-Purpose Register Configuration

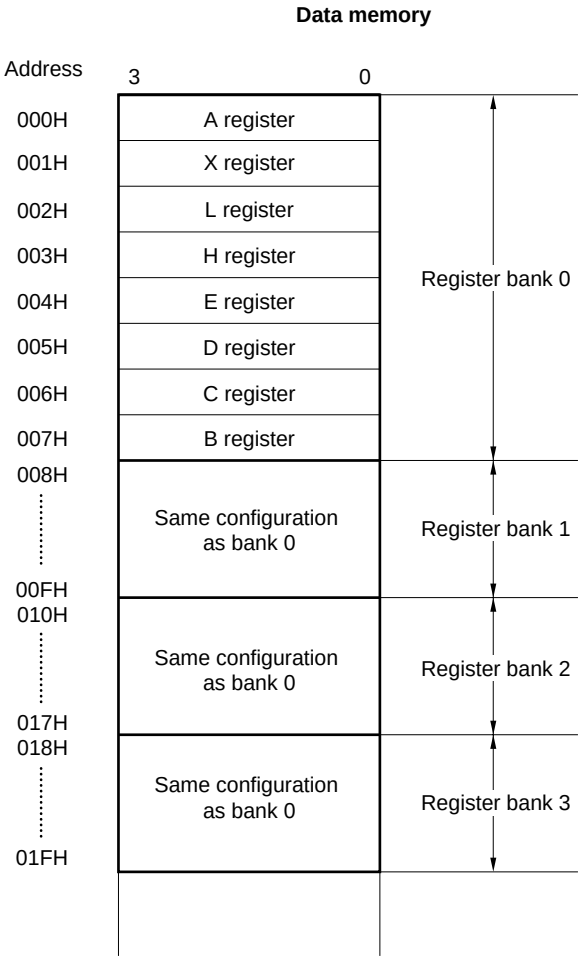
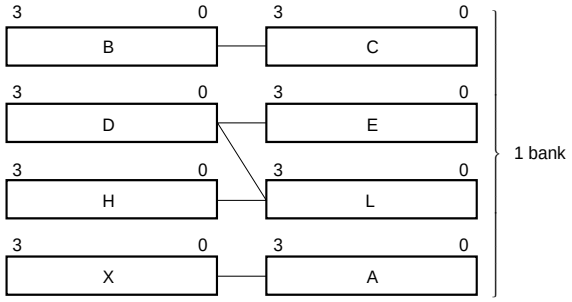


Figure 4-6. Register Pair Configuration

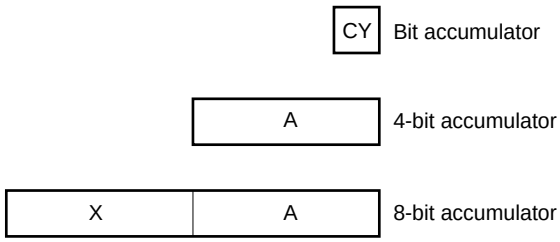


4.6 Accumulators

The μPD754304 uses the A register and XA register pair as accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.

The carry flag (CY) is used for a bit accumulator during execution of bit manipulation instructions.

Figure 4-7. Accumulators



4.7 Stack Pointer (SP) and Stack Bank Selection Register (SBS)

The μPD754304 uses static RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.

The stack area is addresses 000H-0FFH of memory bank 0 or 1. Specify one memory bank using 2-bit SBS (See **Table 4-2**).

Table 4-2. Stack Area Selected by SBS

SBS		Stack area
SBS1	SBS0	
0	0	Memory bank 0
Other than above		Setting prohibited

The SP decrements before a write (save) operation in the stack memory and increments after a read (restore) operation from it. SBS is set by 4-bit memory manipulation instruction (see **Figure 4-1**).

Figures 4-9 to **4-12** show the data saved and restored by the stack operations.

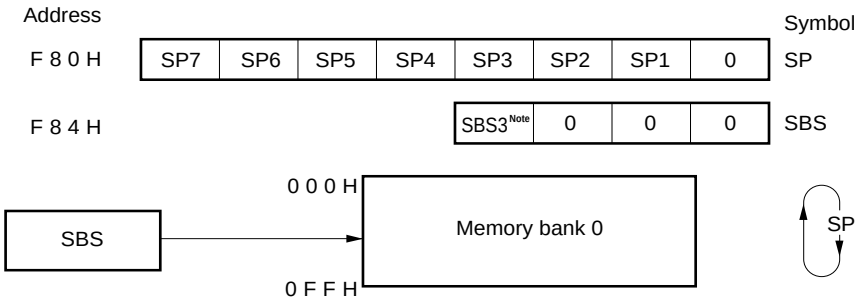
The initial value of the SP is set by an 8-bit memory manipulation instruction and the initial value of the SBS is set by a 4-bit memory manipulation instruction to determine a stack area. Its contents can also be read.

When 00H is set in the SP as the initial value, data is stacked first in the highest-order address (0FFH) in the memory bank (0) specified by the SBS.

The stack area is limited to the memory bank specified by the SBS, and data is returned to 0FFH in the same bank when further stacking operation is performed in addresses starting with 000H. Data cannot be stacked over the boundary of memory bank without rewriting the SBS.

Because generation of the $\overline{\text{RESET}}$ signal causes SP to become undefined and SBS to be set to 1000B, be sure to load SP and SBS with user-desired values on the first stage of the program.

Figure 4-8. Stack Pointer and Stack Bank Selection Register Configuration



Note Switching between the Mk I mode and Mk II mode can be done by a SBS3. The stack bank select function can be used in both the Mk I mode and Mk II mode (See **4.1 Switching Function between Mk I Mode and Mk II Mode**).

Example SP Initialization

Memory bank 1 is assigned to the stack area and data is stacked in addresses starting with 0FFH.

```
SEL     MB15        ; or CLR1 MBE
MOV     A, #8H
MOV     SBS, A      ; Assign memory bank 1 as the stack area.
MOV     XA, #00H
MOV     SP, XA      ; SP ← 00H
```

Figure 4-9. Data Saved in Stack Memory (Mk I mode)

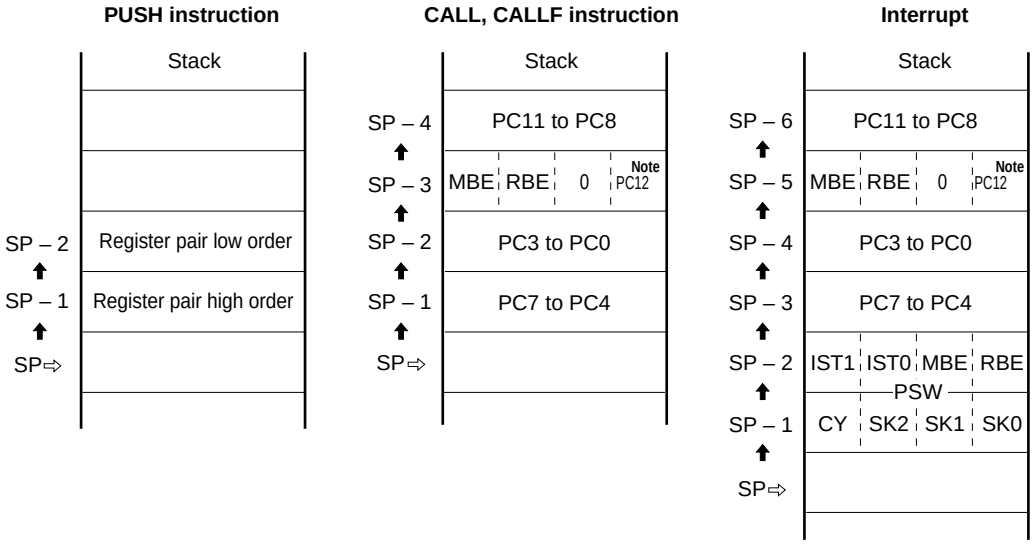
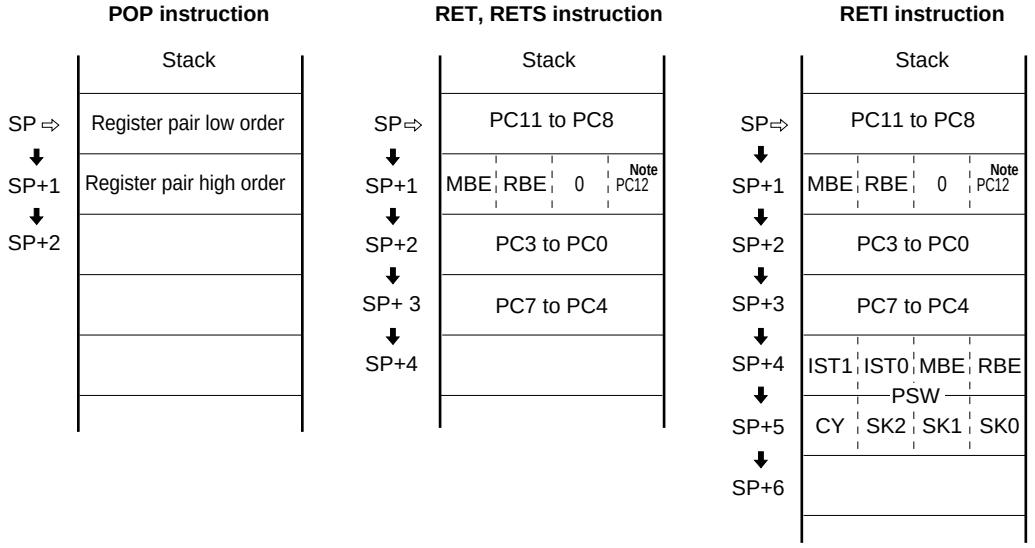


Figure 4-10. Data Restored from Stack Memory (Mk I mode)



Note For the μPD754302, PC11 and 12 are set to 0. For the μPD754304, PC12 is set to 0.

Figure 4-11. Data Saved in Stack Memory (Mk II mode)

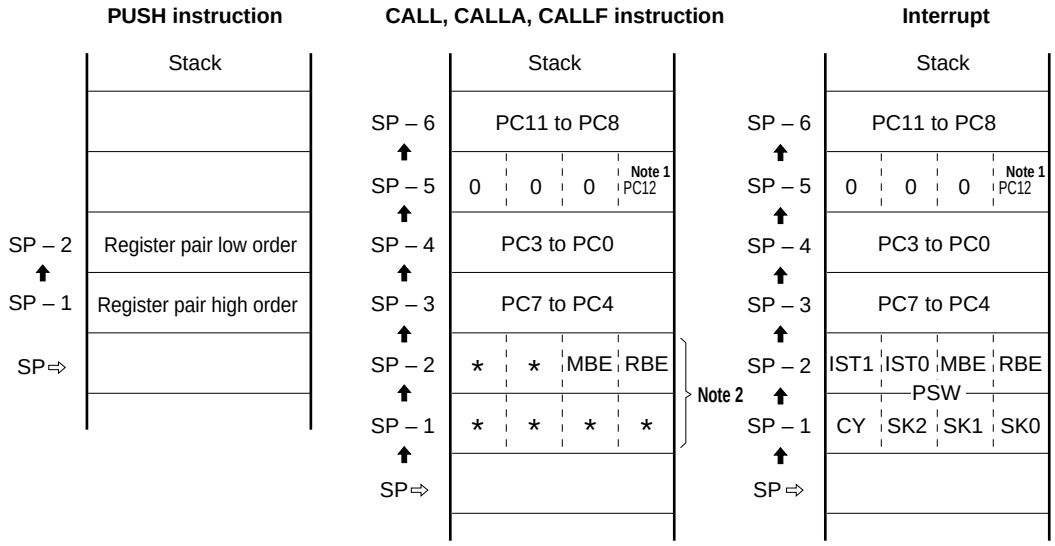
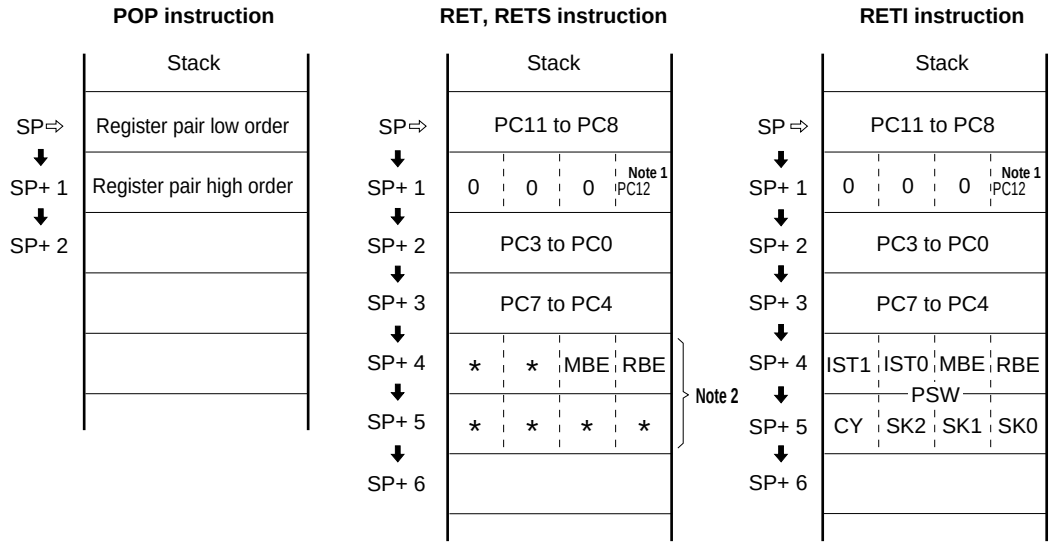


Figure 4-12. Data Restored from Stack Memory (MkII mode)



Notes 1. For the μPD754302, PC11 and 12 are set to 0. For the μPD754304, PC12 is set to 0.

2. PSW other than MBE and RBE is not saved/restored.

Remark The asterisks (*) in the above illustrations means undefined.

4.8 Program Status Word (PSW) ··· 8 bits

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

PSW is mapped in memory space addresses FB0H and FB1H, and four bits of address FB0H can be manipulated by executing a memory manipulation instruction.

Figure 4-13. Program Status Word Format

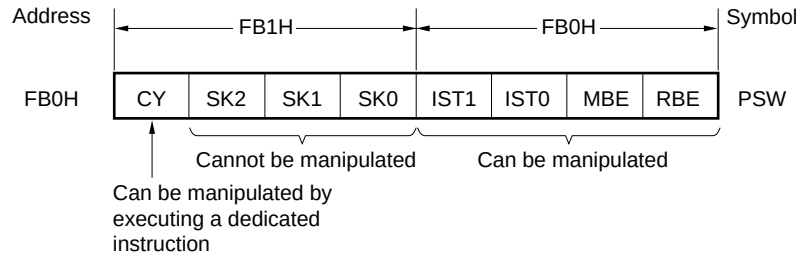


Table 4-3. PSW Flags Saved and Restored during Stack Operation

		Flags saved and restored
Save	When CALL, CALLA or CALLF instruction is executed	MBE and RBE are saved
	When hardware interrupt is executed	All PSW bits are saved
Restore	When RET or RETS instruction is executed	MBE and RBE are restored
	When RETI instruction is executed	All PSW bits are restored

(1) Carry flag (CY)

The carry flag (CY) is a 1-bit flag which stores overflow or underflow occurrence information when an operation instruction with carry (ADDC or SUBC) is executed. The carry flag also serves as a bit accumulator. Boolean algebra operation is performed between the bit accumulator and the data memory specified by bit address, and the result can be stored in the accumulator.

The carry flag is manipulated by executing a dedicated instruction independently of other PSW bits.

When a $\overline{\text{RESET}}$ is input, the carry flag becomes undefined.

Table 4-4. Carry Flag Manipulation Instructions

	Instruction (mnemonics)	Carry flag manipulation and processing
Carry flag manipulation dedicated instruction	SET1 CY CLR1 CY NOT1 CY SKT CY	Set CY to 1 Reset CY to 0 Reverses the CY status Skip if CY contains 1
Bit transfer instruction	MOV1 mem*.bit, CY MOV1 CY,mem*.bit	Transfer CY contents to the specified bit Transfer the specified bit contents to CY
Bit Boolean instruction	AND1 CY,mem*.bit OR1 CY,mem*.bit XOR1 CY,mem*.bit	AND, OR, and XOR in the specified bit contents and CY contents and set the result in CY
Interrupt processing	When interrupt is executed	Save CY and other PSW bits in stack memory in parallel
	RETI	Restore CY and other PSW bits in parallel from stack memory

Remark mem*.bit indicates following three bit manipulation addressing

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

Example AND address 3FH bit 3 and P33 and output the result to P50.

```
MOV     H, #3H          ; Set high-order 4-bit address in H register
MOV1    CY, @H+0FH.3    ; CY ← 3FH BIT 3
AND1    CY, PORT3.3     ; CY ← CY ^ P33
MOV1    PORT5.0, CY     ; P50 ← CY
```

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

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

(3) Interrupt status flag (IST1, IST0)

The interrupt status flag is a 2-bit flag which stores the status of the current processing being performed (For details, see **Table 6-3. IST1 and IST0 and Interrupt Processing Status**).

Table 4-5. Interrupt Status Flag Indication

IST1	IST0	Status of processing being performed	Processing indication and interrupt control
0	0	Status 0	During normal program processing. Acknowledgment of all interrupts is enabled.
0	1	Status 1	During low-priority or high-priority interrupt processing. High-priority interrupt acknowledgment is enabled.
1	0	Status 2	During high-priority interrupt processing. Acknowledgment of all interrupt is disabled.
1	1	–	Setting prohibited

The interrupt priority control circuit (see **Figure 6-1. Interrupt Control Circuit Block Diagram**) judges the interrupt status flag contents to control multiple interrupt.

If an interrupt is acknowledged, the IST1 and IST0 contents are saved in the stack memory as a part of PSW, then automatically changed to the upper status. When the RETI instruction is executed, the value before the interrupt service routine is entered is restored.

The interrupt status flag can be manipulated by executing a memory manipulating instruction. The status of processing being performed can also be changed under the program control.

Caution To manipulate the flag, be sure to execute a DI instruction to disable interrupts before manipulation and execute an EI instruction to enable interrupts after manipulation.

(4) Memory bank enable flag (MBE)

The memory bank enable flag (MBE) is a 1-bit flag to specify the address information generation mode of the high-order four bits of a 12-bit data memory address.

MBE can be set or reset at any time, regardless of the setting of the memory bank.

When MBE is set to 1, the data memory address space is expanded and all the data memory space can be addressed.

When MBE is reset to 0, the data memory address space is fixed regardless of the MBS contents (See **Figure 3-2. Data Memory Configuration and Addressing Range for Each Addressing Mode**).

When a $\overline{\text{RESET}}$ signal is input, the contents of bit 7 of program memory address 0 is set in MBE for automatic initialization.

When vectored interrupt processing is performed, the bit 7 contents of the corresponding vector address table are set and the MBE state during the interrupt service is automatically set.

Normally, in interrupt processing, MBE is set to 0 for use of static RAM of memory bank 0.

(5) Register bank enable flag (RBE)

The register bank enable flag (RBE) is a 1-bit flag to control whether or not the register bank configuration of the general-purpose registers is expanded.

RBE can be set or reset at any time, regardless of the setting of the memory bank.

When RBE is set to 1, general-purpose registers of one bank can be selected among register banks 0-3 according to the register bank selection register (RBS) contents.

When RBE is reset to 0, register bank 0 is always selected for general-purpose registers regardless of the register bank selection register (RBS) contents.

When a $\overline{\text{RESET}}$ signal is input, the bit 6 contents of program memory address 0 are set in RBE for automatic initialization.

When a vectored interrupt occurs, the bit 6 contents of the corresponding vector address table are set and the RBE state during the interrupt service is automatically set. Normally, in interrupt processing, RBE is set to 0 for use of register bank 0 for 4-bit operation or register bank 0 and 1 for 8-bit operation.

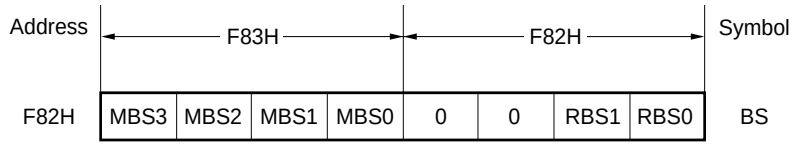
4.9 Bank Selection Register (BS)

The bank selection register (BS) consists of the register bank selection register (RBS) and memory bank selection register (MBS) to specify the register bank and memory bank to be used.

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

BS can be saved in and restored from the stack memory in 8-bit units by executing the PUSH BS and POP BS instructions.

Figure 4-14. Bank Selection Register Format



(1) Memory bank selection register (MBS)

The memory bank selection register (MBS) is a 4-bit register which stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified by the register contents (For the μPD754304, only banks 0 and 15 can be specified).

MBS is set by executing the SEL MBn instruction (n = 0 or 15).

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

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

(2) Register bank selection register (RBS)

The register bank selection register (RBS) is a register to specify the register bank used as general- purpose registers. One of banks 0 to 3 can be selected.

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

When a $\overline{\text{RESET}}$ signal is input, RBS is initialized to 0.

Table 4-6. RBE, RBS, and Selected Register Bank

RBE	RBS				Register bank
	3	2	1	0	
0	0	0	×	×	Fixed to bank 0
1	0	0	0	0	Bank 0 selection
			0	1	Bank 1 selection
			1	0	Bank 2 selection
			1	1	Bank 3 selection

Fixed to 0

× : Don’t care

CHAPTER 5 PERIPHERAL HARDWARE FUNCTION

5.1 Digital I/O Port

Memory mapped I/O is employed for the μPD754304. All the I/O ports are mapped in the data memory space.

Figure 5-1. Digital Ports Data Memory Addresses

Address	3	2	1	0	
F F 0 H	P03	P02	P01	P00	PORT0
F F 1 H	P13	P12	P11	P10	PORT1
F F 2 H	P23	P22	P21	P20	PORT2
F F 3 H	P33	P32	P31	P30	PORT3
F F 5 H	P53	P52	P51	P50	PORT5
F F 6 H	P63	P62	P61	P60	PORT6
F F 7 H	P73	P72	P71	P70	PORT7
F F 8 H	–	–	P81	P80	PORT8

Table 5-2 lists the input/output ports manipulation instructions for Port 6 and Port 7 in addition to 4-bit input/output, 8-bit input/output and bit manipulation can be performed. These enable many types of control.

Examples 1. The status shown on P13 is tested and the values depending on the results of test are output to ports 6 and 7.

```
SKT     PORT1.3     ; Skip if bit 3 of port 1 is 1.
MOV     XA, #18H    ; XA ← 18H
MOV     XA, #14H    ; XA ← 14H  ⟍ String effect
SEL     MB15        ; or CLR1 MBE
OUT     PORT6, XA   ; ports 7, 6 ← XA
```

2. SET1 PORT6. @L ; The bit (in ports 6 and 7) specified by the L register is set to 1.

5.1.1 Types, features, configuration of digital I/O ports

Table 5-1 lists the types of digital I/O ports.

The configurations of the ports are shown in **Figures 5-2** to **5-6**.

Table 5-1. Types and Features of Digital Ports

Port	Function	Operation & features	Remarks
PORT0	4-bit input	Read and test are always possible regardless of operation mode of the dual function pins.	Also used for the INT4, $\overline{SCK}$, SO/SB0 and SI pins.
PORT1			Also used for the INT0 to INT2 and TI0/TI1 pins.
PORT2	4-bit I/O	Can be set to input mode or output mode in 4-bit units. Ports 6 and 7 are paired and data can be input/output in 8-bit units.	Also used for the PTO0 to PTO2 and PCL pins.
PORT7			Also used for the KR4 to KR7 pins.
PORT3 **Note 1**		Can be set to output mode bit-wise.	Also used for the MD0 to MD3 **Note 2** pins.
PORT6			Also used for the KR0 to KR3 pins.
★ PORT5 **Note 1**	4-bit I/O (N-channel open-drain, 13 V DC rating)	Can be set to input mode or output mode in 4-bit units.	Can be specify the on-chip of the pull-up resister in 1-bit unit with a mask option **Note 3**.
PORT8	2-bit I/O	Can be set to input or output mode in 2-bit units.	–

Notes 1. Can directly drive a LED.

2. The μPD75P4308 is only shared.

3. The μPD75P4308 provides no mask option and a pull-up resistor can not be on-chipped.

P10 is also used for an external vectored interrupt input pin and has a noise eliminator (See **6.3 Hardware Controlling Interrupt Functions**).

The occurrence of the $\overline{RESET}$ signal causes the output latches of ports 2, 3, and 5 through 8 to be cleared. The output buffer goes off and these ports become in input mode.

Figure 5-2. Port 0, 1 Configuration

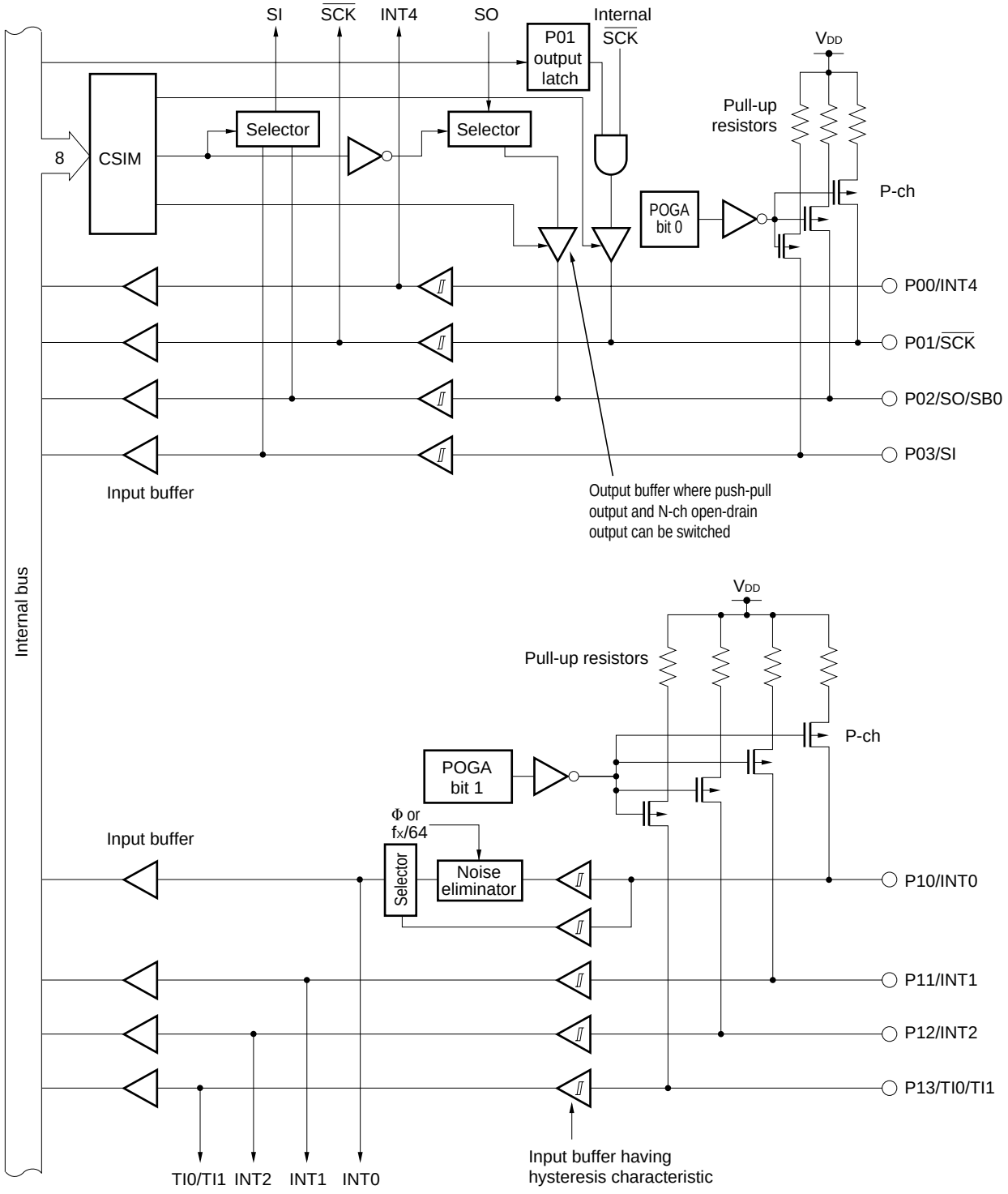


Figure 5-3. Port 2, 7 Configuration

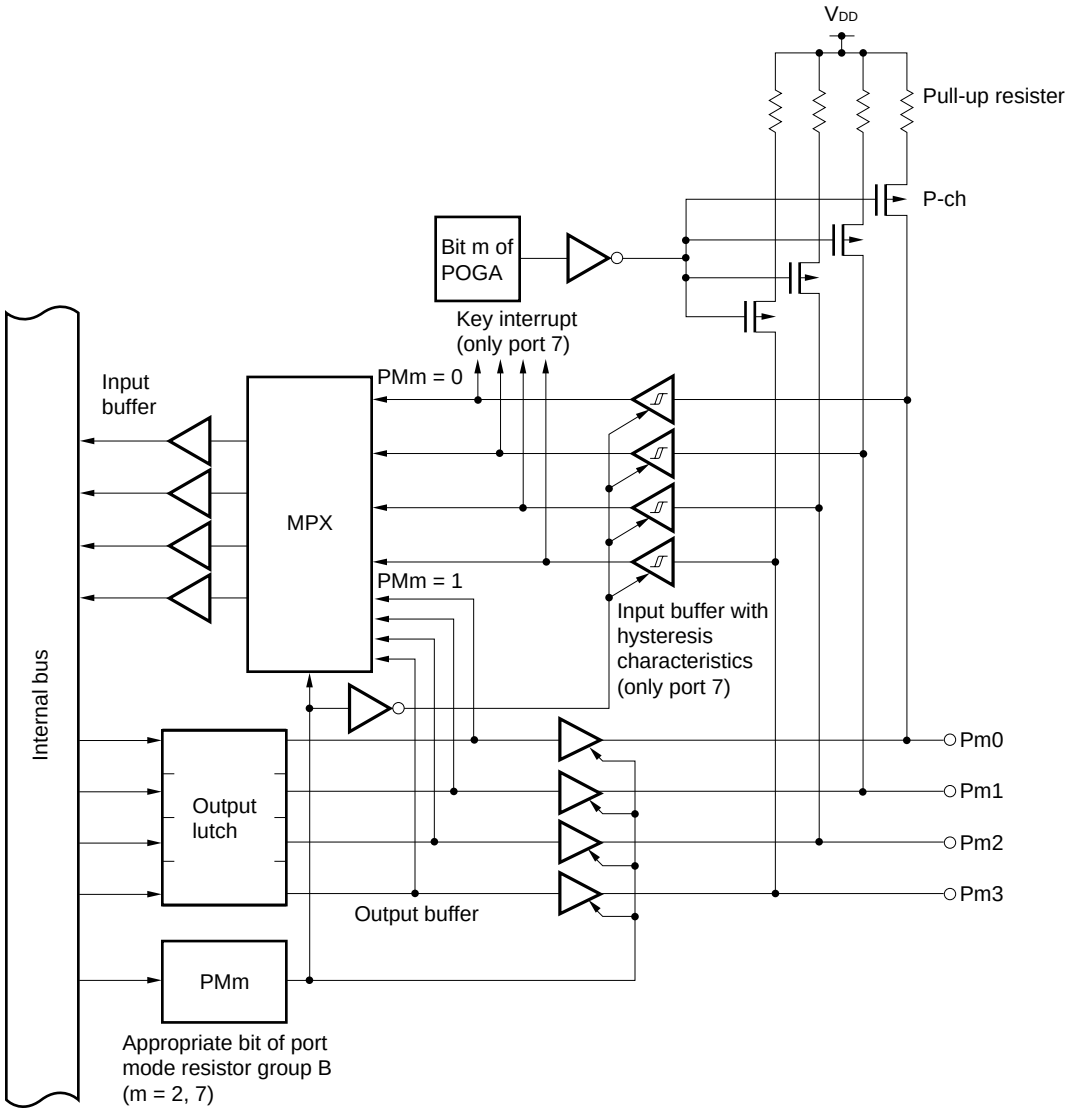


Figure 5-4. Port 3, Port 6 Configuration

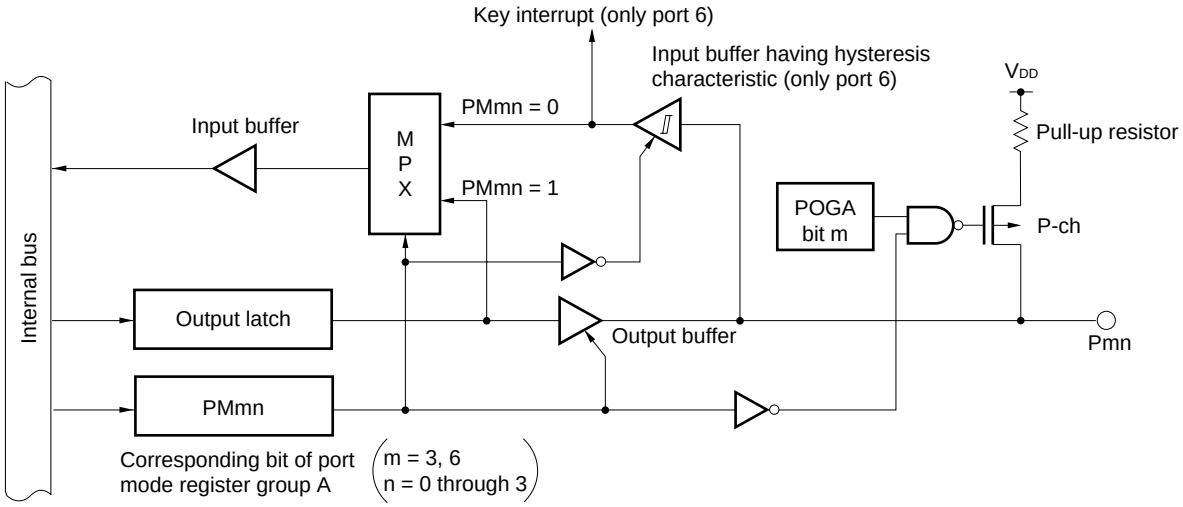


Figure 5-5. Port 5 Configuration

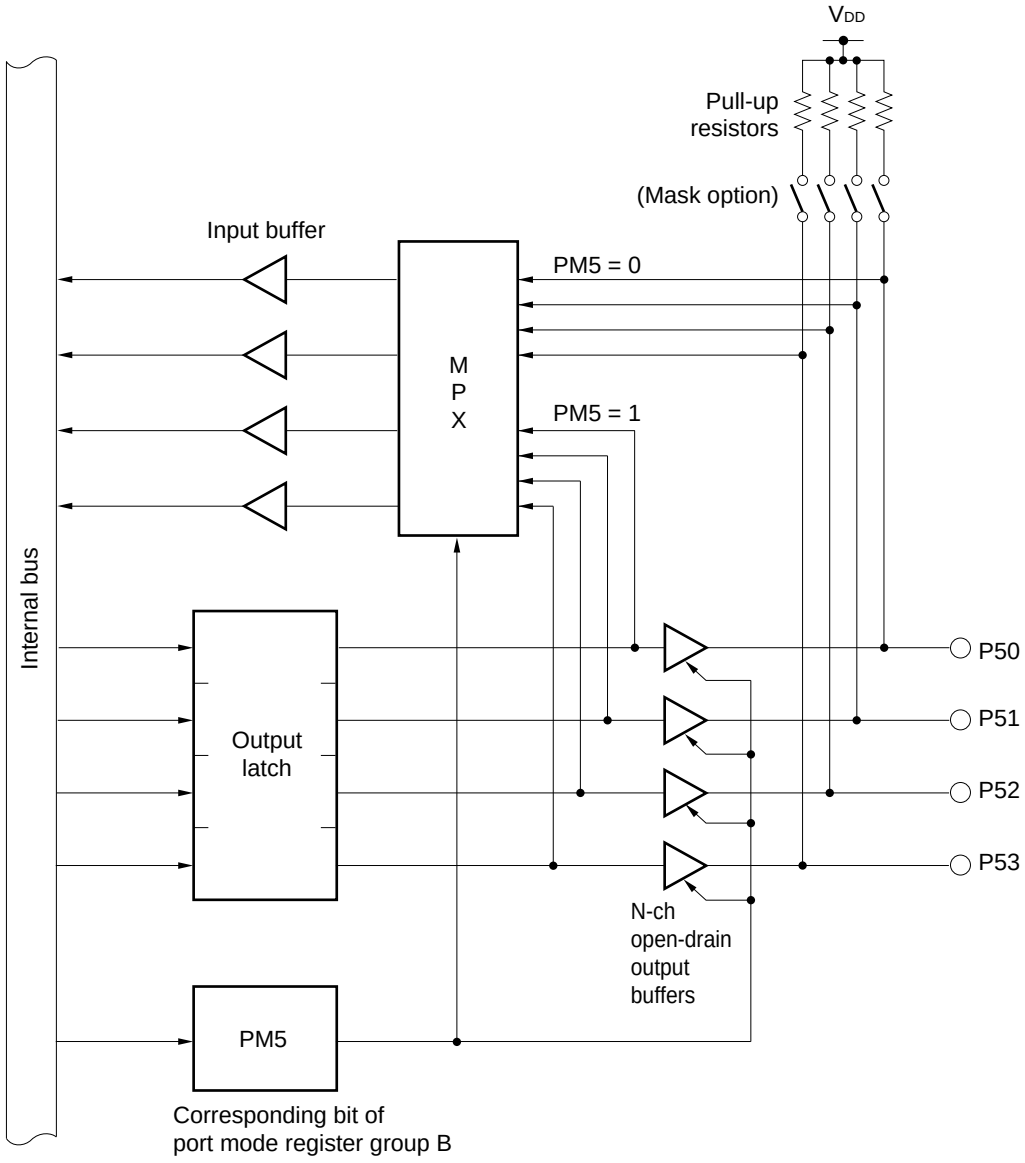
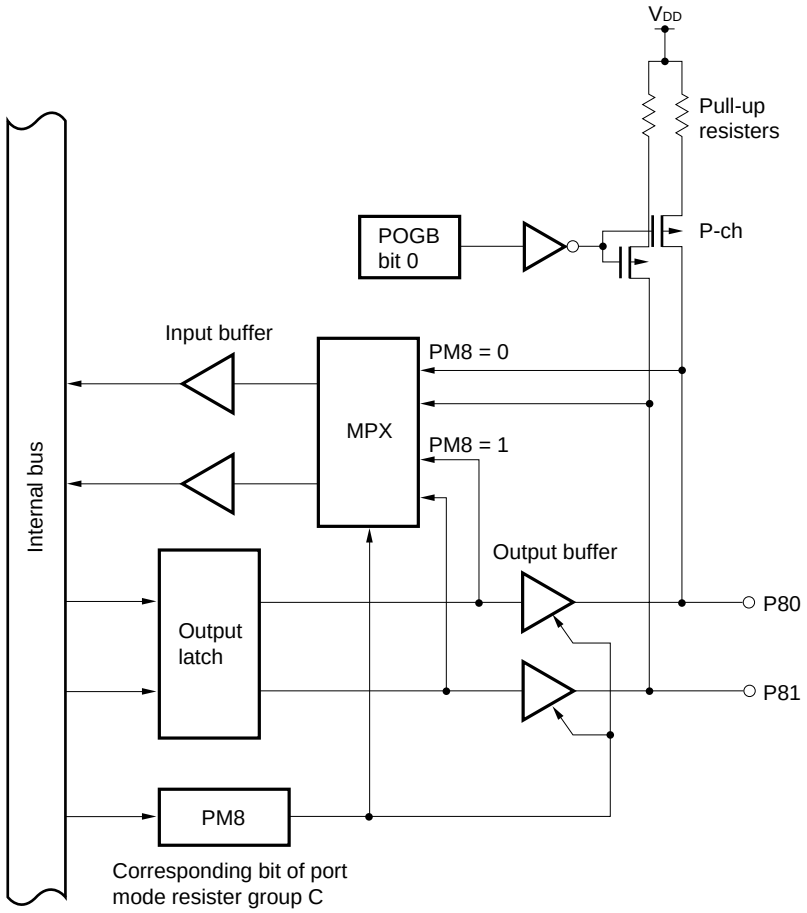


Figure 5-6. Port 8 Configuration



5.1.2 Setting I/O mode

The input or output mode of each I/O port is set by the corresponding port mode register as shown in **Figure 5-7**. Ports 3 and 6 can be set in the input or output mode in 1-bit units by using port mode register group A (PMGA). Ports 2, 5, and 7 are set in the input or output mode in 4-bit units by using port mode register group B (PMGB). Port mode group register group C (PMGC) is used to set the input or output mode of port 8 in 2-bit units.

Each port is set in the input mode when the corresponding port mode register bit is “0” and in the output mode when the corresponding register bit is “1”.

When a port is set in the output mode by the corresponding port mode register, the contents of the output latch are output to the output pin(s). Before setting the output mode, therefore, the necessary value must be written to the output latch.

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

When the $\overline{\text{RESET}}$ signal is asserted, all the bits of each port mode register are cleared to 0, turning off the output buffer and setting the corresponding port in the input mode.

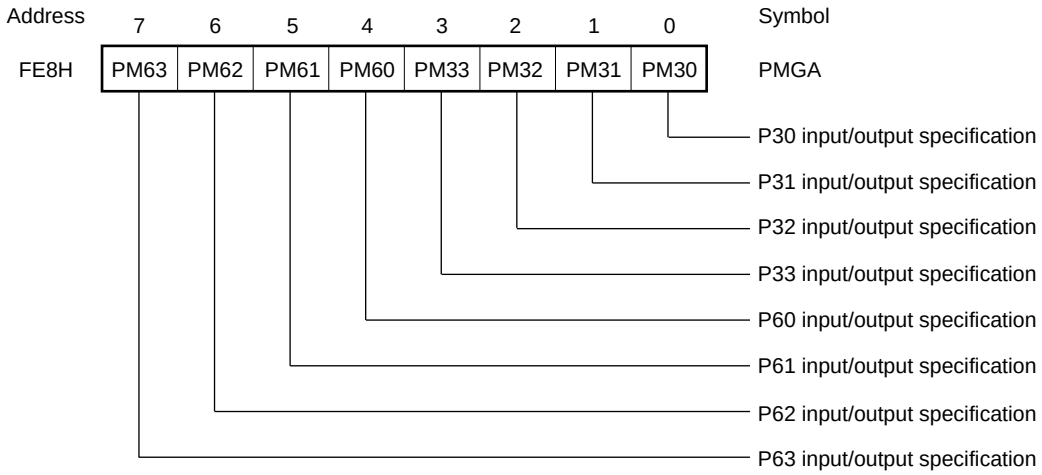
Example To use P30, 31, 62, and 63 as input pins and P32, 33, 60, and 61 as output pins

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

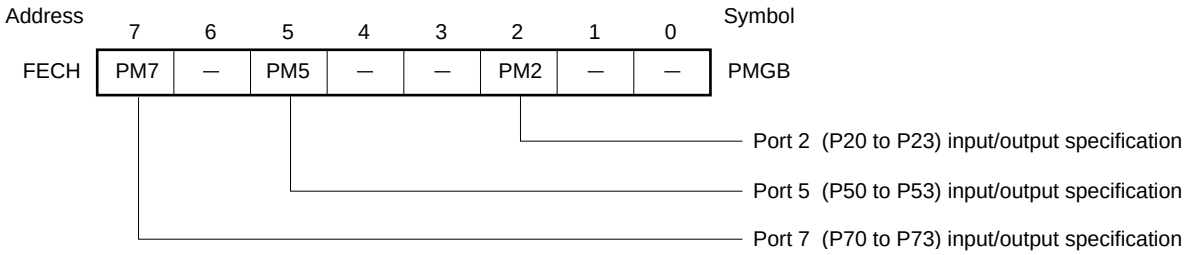
Figure 5-7. Port Mode Register Formats

	Specification
0	Input mode (output buffer off)
1	Output mode (output buffer on)

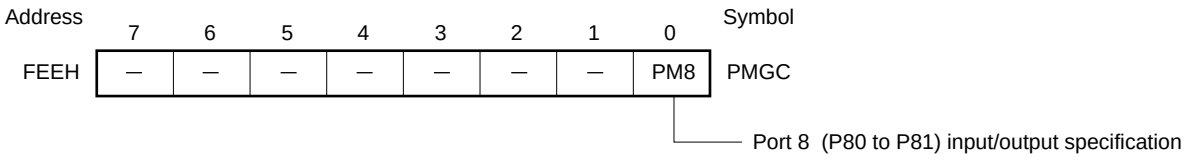
Port mode register group A



Port mode register group B



Port mode register group C



5.1.3 Digital I/O port manipulation instruction

Because all the I/O ports of the μPD754304 are mapped to the data memory space, they can be manipulated by using data memory manipulation instructions. Of these data memory manipulation instructions, those considered to be especially useful for manipulating the I/O pins and their range of applications are shown in **Table 5-3**.

(1) Bit manipulation instruction

Because the specific address bit direct addressing (fmem.bit) and specific address bit register indirect addressing (pmem.@L) are applicable to digital I/O ports 0 through 3, 5, 6, 7 and 8, the bits of these ports can be manipulated regardless of the specifications by MBE and MBS.

Example To OR P70 and P61, and output to P31

```
        SET1    CY              ; CY ← 1
        AND1    CY, PORT7.0     ; CY ← CY ∧ P70
        OR1     CY, PORT6.1     ; CY ← CY ∨ P61
        SKT     CY
        BR      CLRP
        SET1    PORT3.1         ; P31 ← 1
          ⋮
CLRP:   CLR1    PORT3.1         ; P31 ← 0
```

(2) 4-bit manipulation instruction

In addition to the IN and OUT instructions, all the 4-bit memory manipulation instructions such as MOV, XCH, ADDS, and INCS can be used to manipulate the ports in 4-bit units. Before executing these instructions, however, memory bank 15 must be selected.

Examples 1. To output the contents of the accumulator to port 3

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

2. To add the value of the accumulator to the data output to port 5

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

3. To test whether the data of port 6 is greater than the value of the accumulator

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT6
SUBS    A, @HL          ; A < PORT6
BR      NO              ; NO
                        ; YES
```

(3) 8-bit manipulation instruction

In addition to the IN and OUT instructions, the MOV, XCH, and SKE instructions can be used to manipulate ports 6 and 7, which can be manipulated in 8-bit units. In this case also, memory bank 15 must be selected, just as when 4-bit manipulation instructions are used to manipulate ports.

Example To output the data of register pair BC to the output port specified by the 8-bit data input from ports 6 and 7

```
SET1    MBE
SEL     MB15
IN      XA, PORT6       ; XA ← ports 7 and 6
MOV     HL, XA          ; HL ← XA
MOV     XA, BC          ; XA ← BC
MOV     @HL, XA         ; Port (L) ← XA
```

Table 5-2. I/O Pin Manipulation Instructions

Instruction \ PORT	PORT 0	PORT 1	PORT 2	PORT 3	PORT 5	PORT 6	PORT 7	PORT 8
IN A, PORTn Note 1					√			
IN XA, PORTn Note 1	–		–		–	√		–
OUT PORTn, A Note 1	–				√			
OUT PORTn, XA Note 1	–		–		–	√		–
SET1 PORTn. bit	–				√			
SET1 PORTn. @L Note 2	–				√			
CLR1 PORTn. bit	–				√			
CLR1 PORTn. @L Note 2	–				√			
SKT PORTn. bit					√			
SKT PORTn. @L Note 2					√			
SKF PORTn. bit					√			
SKF PORTn. @L Note 2					√			
★ MOV1 CY, PORTn. bit					√			
★ MOV1 CY, PORTn. @L Note 2					√			
★ MOV1 PORTn. bit, CY	–	–			√			
★ MOV1 PORTn. @L, CY Note 2	–	–			√			
AND1 CY, PORTn. bit					√			
AND1 CY, PORTn. @L Note 2					√			
OR1 CY, PORTn. bit					√			
OR1 CY, PORTn. @L Note 2					√			
XOR1 CY, PORTn. bit					√			
XOR1 CY, PORTn. @L Note 2					√			

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

2. The low-order 2 bits and the bit addresses of the address must be indirectly specified by the L register.

5.1.4 Operation of digital I/O port

The operations of each port and port pin when a data memory manipulation instruction is executed to manipulate a digital I/O port differs depending on whether the port is set in the input or output mode (refer to **Table 5-3**). This is because, as can be seen from the configuration of the I/O port, the data of each pin is loaded to the internal bus in the input mode, and the data of the output latch is loaded to the internal bus in the output mode.

(1) Operation in input mode

When a test instruction such as SKT, a bit input instruction such as MOV1, or an instruction that loads port data to the internal bus, such as IN, OUT, an operation, or a comparison instruction, is executed, the data of each pin is manipulated. ★

When an instruction that transfers the contents of the accumulator in 4- or 8-bit units, such as OUT or MOV, is executed, the data of the accumulator is latched to the output latch. The output buffer remains off.

When the XCH instruction is executed, the data of each pin is input to the accumulator, and the data of the accumulator is latched to the output latch. The output buffer remains off.

When the INCS instruction is executed, the data which 1 is added to the data of each pin (4 bits) is latched to the output latch. The output buffer remains off.

When an instruction that rewrites the data memory contents in 1-bit units, such as SET1, CLR1, MOV1, or SKTCLR, is executed, the contents of the output latch of the specified bit can be rewritten as specified by the instruction, but the contents of the output latches of the other bits are undefined.

(2) Operation in output mode

When a test instruction, bit input instruction, or an instruction that loads port data to the internal bus is executed, the contents of the output latch are manipulated.

When an instruction that transfers the contents of the accumulator in 4- or 8-bit units is executed, the data of the output latch is rewritten and at the same time output from the port pins.

When the XCH instruction is executed, the contents of the output latch are transferred to the accumulator, and the contents of the accumulator are latched to the output latches of the specified port and output from the port pins.

When the INCS instruction is executed, the contents of the output latches of the specified port are incremented by 1 and output from the port pins.

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

Table 5-3. Operation When an I/O Port Is Manipulated

Instruction executed	Operation of port and pins	
	Input mode	Output mode
SKT ① SKF ①	Tests pin data	Tests output latch data
★ MOV1 CY, ①	Transfers pin data to CY	Transfers output latch data to CY
AND1 CY, ① OR1 CY, ① XOR1 CY, ①	Performs operation between pin data and CY	Performs operation between output latch data and CY

IN A, PORTn
IN XA, PORTn
MOV A, @HL
MOV XA, @HL
	Transfers pin data to accumulator	Transfers output latch data to accumulator

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

SKE A, @HL SKE XA, @HL	Compares pin data with accumulator	Compares output latch data with accumulator

OUT PORTn, A
OUT PORTn, XA
MOV @HL, A
MOV @HL, XA
	Transfers accumulator data to output latch (output buffer remains off)	Transfers accumulator data to output latch and outputs data from pins

XCH A, PORTn XCH XA, PORTn XCH A, @HL XCH XA, @HL	Transfers pin data to accumulator and accumulator data to output latch (output buffer remains off)	Exchanges data between output latch and accumulator
INCS PORTn INCS @HL	Increments pin data by 1 and latches it to output latch	Increments output latch contents by 1
SET1 ① CLR1 ① ★ MOV1 ① , CY SKTCLR ①	Rewrites output latch contents of specified bit as specified by instruction but output latch contents of other bits are undefined	Changes status of output pin as specified by instruction

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

5.1.5 Connecting pull-up resistors

Each port pin of the μPD754304 can be connected to an internal pull-up resistor (except the P00 pin). Some pins can be connected with a pull-up resistor via software and the others can be connected by mask option.

Table 5-4 shows how to specify connection of the on-chip pull-up resistor to each port. The on-chip pull-up resistor is connected via software in the format shown in **Figure 5-8**.

The on-chip pull-up resistor can be connected only to the pins of ports 3 and 6 in the input mode. To the pins set to output mode, the on-chip pull-up resistors cannot be connected regardless of the setting of POGA.

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

Port (pin name)	Specification method	Specified bit
PORT0 (P01-P03) **Note**	Specifies to connect the on-chip pull-up resistors by software in 3-bit units.	POGA.0
PORT1 (P10-P13)	Specifies to connect the on-chip pull-up resistors by software in 4-bit units.	POGA.1
PORT2 (P20-P23)		POGA.2
PORT3 (P30-P33)		POGA.3
PORT6 (P60-P63)		POGA.6
PORT7 (P70-P73)		POGB.7
PORT8 (P80, P81)	Specifies to connect the on-chip pull-up resistors by software in 2-bit units.	POGB.0
PORT5 (P50-P53)	Specifiable by mask option in 1-bit units.	–

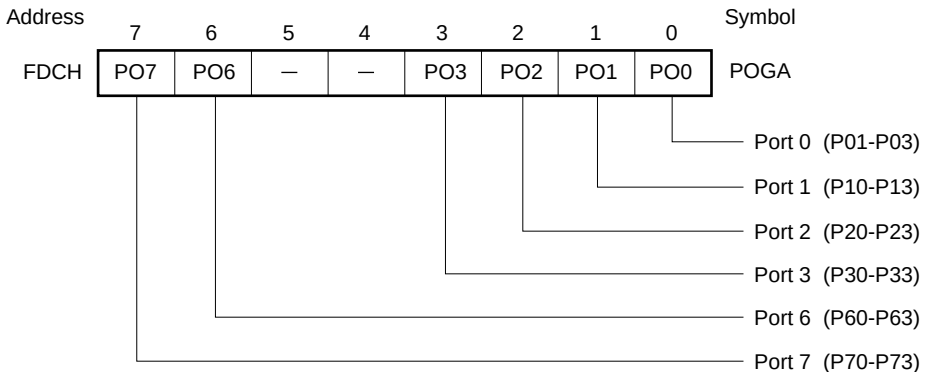
Note The P00 pin cannot be specified to connect an on-chip pull-up resistor.

Remark The port pins of the μPD75P4308 are not connected to a pull-up resistor by mask option and are always open.

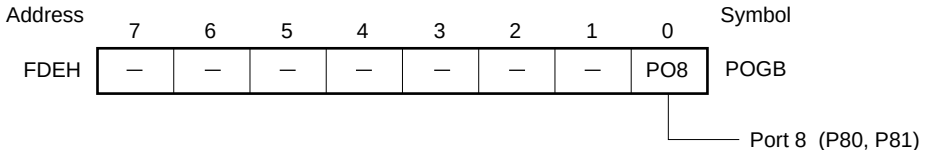
Figure 5-8. Pull-Up Resistor Specify Register Format

	Specification
0	Disables on-chip pull-up resistor.
1	Enables on-chip pull-up resistor.

Pull-up resistor specify register group A



Pull-up resistor specify register group B



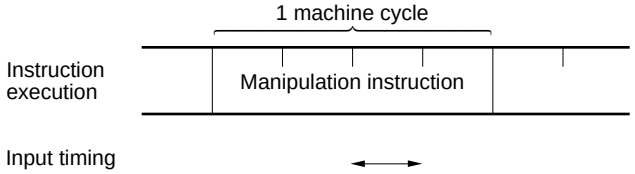
5.1.6 I/O timing of digital I/O port

Figure 5-9 shows the timing by which data is output to the output latch and the timing by which the pin data or the data of the output latch is loaded to the internal bus.

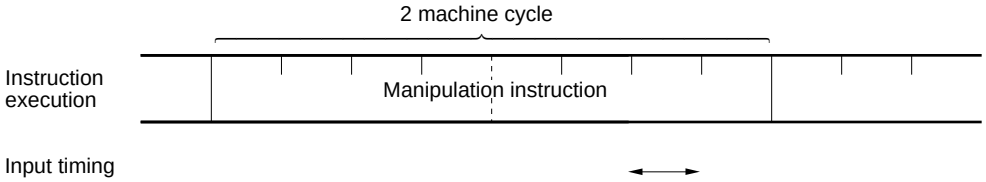
Figure 5-10 shows the ON timing when an on-chip pull-up resistor is connected to a port pin via software.

Figure 5-9. I/O Timing of Digital I/O Port

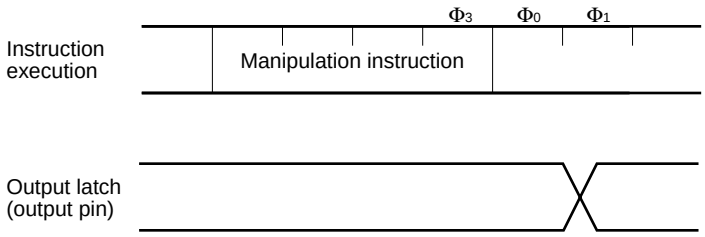
(a) When data is loaded by 1-machine cycle instruction



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



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



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

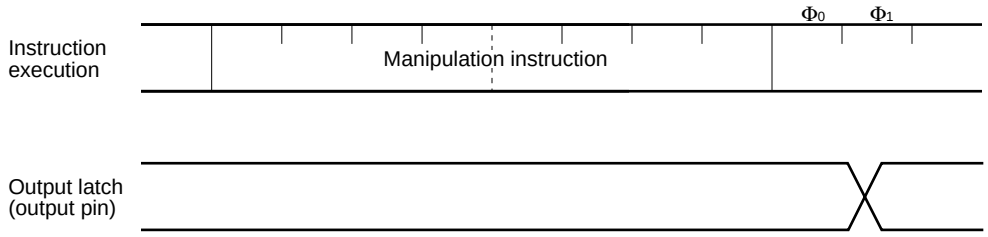
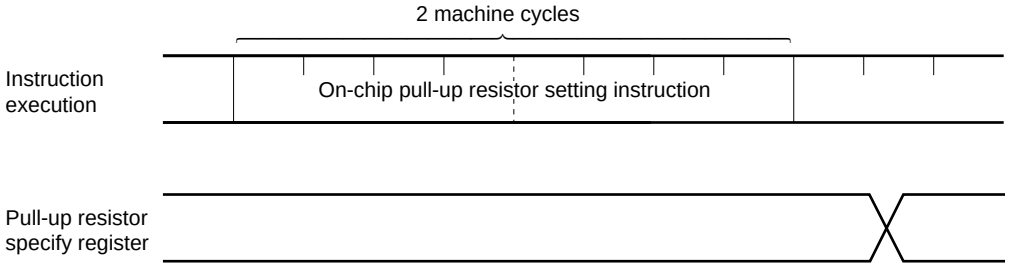


Figure 5-10. ON Timing of On-Chip Pull-up Resistor Connected via Software



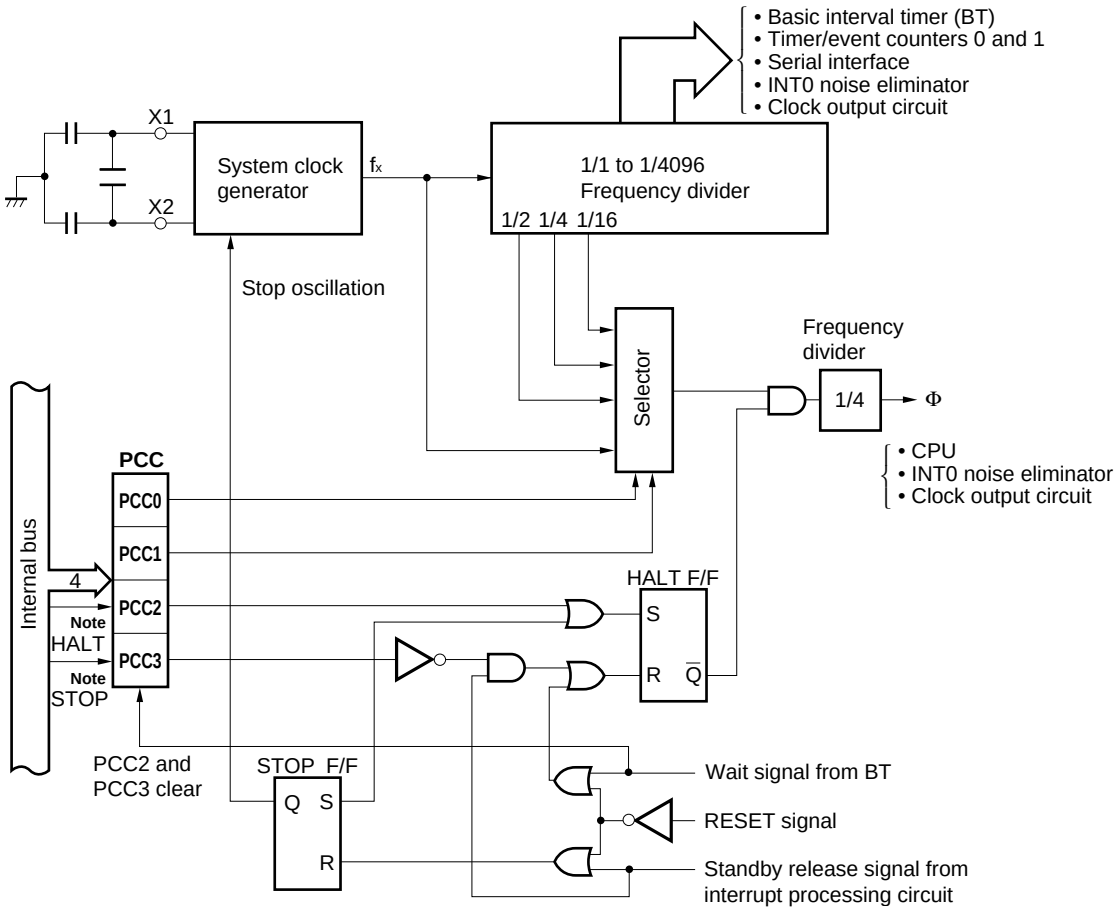
5.2 Clock Generator

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

5.2.1 Clock generator configuration

The configuration of the clock generator is shown in **Figure 5-11**.

Figure 5-11. Clock Generator Block Diagram



Note Instruction execution

Remarks 1. f_X = System clock frequency

2. Φ = CPU clock

3. PCC: Processor Clock Control Register

4. One clock cycle (t_{CY}) of the CPU clock is equal to one mechine cycle of the insturction.

5.2.2 Clock generator function and operation

The clock generator provides the following clock signals and controls the operating mode of the CPU such as standby mode.

- System clock f_X
- CPU clock Φ
- Clock to peripheral hardware

The clock generator operates according to how the processor clock control register (PCC) and system clock control register (SCC) are set, as described below:

(a) When the $\overline{\text{RESET}}$ signal is generated, the minimum speed mode of the system clock (10.7 μs at 6.00- MHz operation) is selected. (PCC = 0)

(b) When the system clock is selected, one of four CPU clock frequencies can be selected (0.67 μs, 1.33 μs, 2.67 μs, and 10.7 μs at 6.00-MHz operation) by setting PCC.

(c) When the system clock is selected, the standby mode (STOP or HALT) can be used.

(d) The system clock is divided to generate a clock supplied to peripheral hardware.

(e) The serial interface and timer/event counter can continue the operation when they have selected an external clock as a clock. However, the other hardware operate with the system clock, therefore it cannot be used when the system clock stops.

(1) Processor clock control register (PCC)

The PCC is a 4-bit register whereof the low-order 2 bits select the CPU clock F and high-order 2 bits control the CPU operating mode (See **Figure 5-12**).

When bit 2 or bit 3 is set to "1" exclusively, the PCC is set in the standby mode. When it is released from the mode by a Standby Release signal, both bits 2 and 3 are automatically cleared for normal operations (See **CHAPTER 7 STANDBY FUNCTION**).

The low-order 2 bits of the PCC are set by a 4-bit memory manipulation instruction. (The high-order 2 bits must be set to "0".)

Bits 2 and 3 are set to "1" by a HALT instruction and STOP instruction, respectively.

These instructions can be executed regardless of the contents of MBE.

Examples 1. The machine cycle is set to the fastest mode (0.67 μs: during 6.00 MHz operation).

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

2. The machine cycle is set to 1.91 μs at 4.19 MHz.

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

3. The PCC is set to the STOP mode. (An NOP instruction must be entered following the STOP instruction or HALT instruction.)

```
STOP
NOP
```

The PCC is cleared to "0" by the $\overline{\text{RESET}}$ signal.

Figure 5-12. Processor Clock Control Register Format

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

CPU operating mode control bit

PCC3	PCC2	Operating mode
0	0	Normal operating mode
0	1	HALT mode
1	0	STOP mode
1	1	Setting disabled

CPU clock selection

(fx = 6.0 MHz)

PCC1	PCC0	CPU clock frequency	1 machine cycle
0	0	Φ = fx/64 (93.7 kHz)	10.7 μs
0	1	Φ = fx/16 (375 kHz)	2.67 μs
1	0	Φ = fx/8 (750 kHz)	1.33 μs
1	1	Φ = fx/4 (1.5 MHz)	0.67 μs

(fx = 4.19 MHz)

PCC1	PCC0	CPU clock frequency	1 machine cycle
0	0	Φ = fx/64 (65.5 kHz)	15.3 μs
0	1	Φ = fx/16 (261.8 kHz)	3.81 μs
1	0	Φ = fx/8 (524 kHz)	1.91 μs
1	1	Φ = fx/4 (1.05 MHz)	0.95 μs

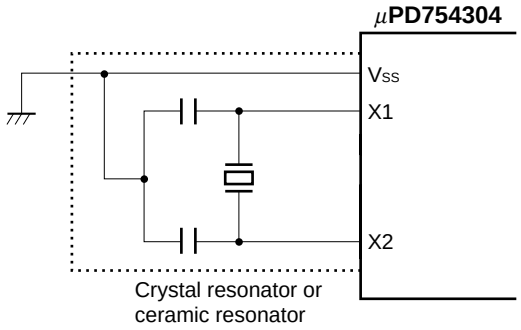
Remark fx: System clock generator output frequency

(2) System clock oscillators

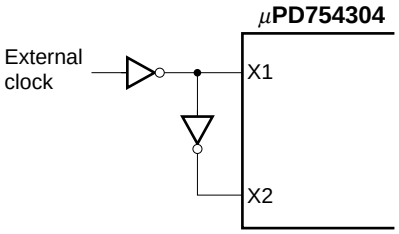
The system clock oscillator oscillates with a crystal resonator or ceramic resonator connected to the X1 and X2 pins. External clock can also be input.

Figure 5-13. Main System Clock Oscillator External Circuit

(a) Crystal or ceramic oscillation



(b) External clock



Cautions 1. When external clock is input in the system clock, the STOP mode cannot be set.

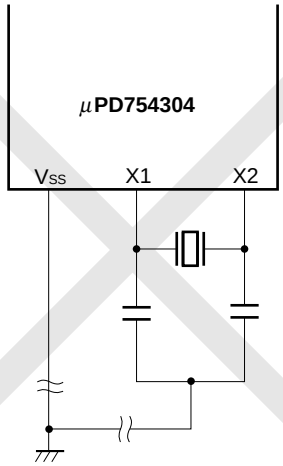
2. Wire the portion enclosed by broken lines in Figure 5-13 as follows to prevent adverse influence by wiring capacitance when using the system clock oscillation circuit.

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any other signal lines.**
- **Do not route the wiring in the vicinity of a line through which a high alternating current flows.**
- **Always keep the potential at the connecting point of the capacitor of the oscillation circuit at the same level as V_{SS}. Do not connect the wiring to ground lines through which a high current flows.**
- **Do not extract signals from the oscillation circuit.**

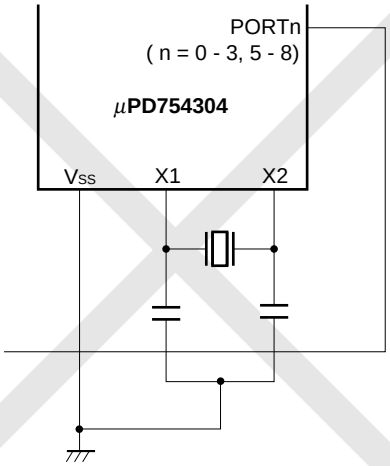
Figure 5-14 shows examples of connecting the resonator incorrectly.

Figure 5-14. Example of Connecting Resonator Incorrectly (1/2)

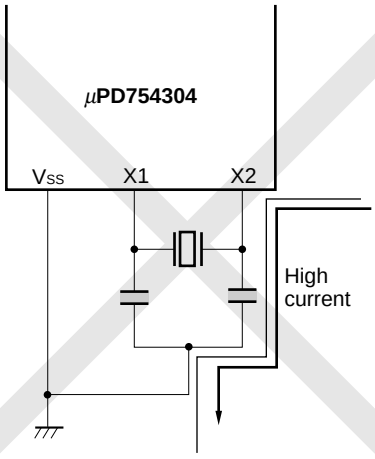
(a) Wiring length too long



(b) Crossed signal line



(c) High alternating current close to signal line



(d) Current flowing through power line of oscillation circuit (potential at points A, B, and C changes)

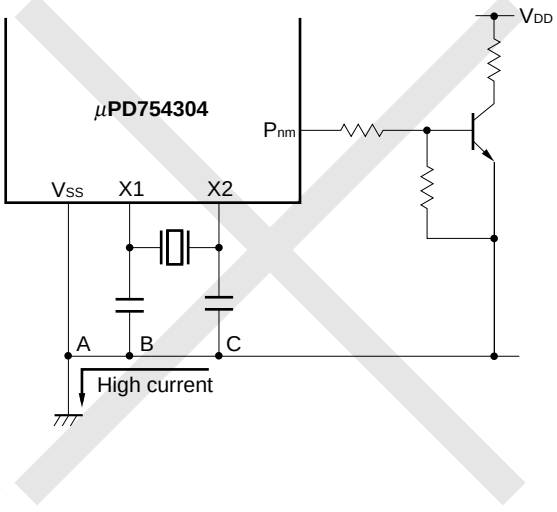
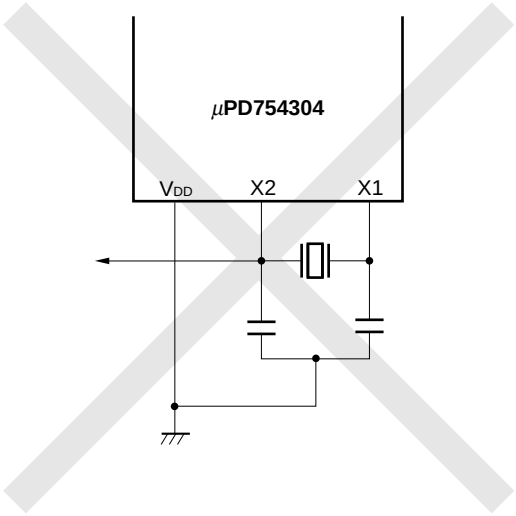


Figure 5-14. Example of Connecting Resonator Incorrectly (2/2)

(e) Signal extracted



(3) Divider circuit

The divider circuit divides the output of the system clock oscillation circuit (f_X) to generate various clocks.

5.2.3 CPU clock settings

(1) Time needed to switch the CPU clock

The CPU clock can be switched by the least significant 2 bits of PCC. This switching is not performed immediately after register switching and operates at the constant clock prior to switching during a constant machine cycle. Accordingly, when the system clock oscillation stops, after this switching time elapses, the STOP instruction must be executed.

Table 5-5. Maximum Time Needed to Switch the CPU Clock

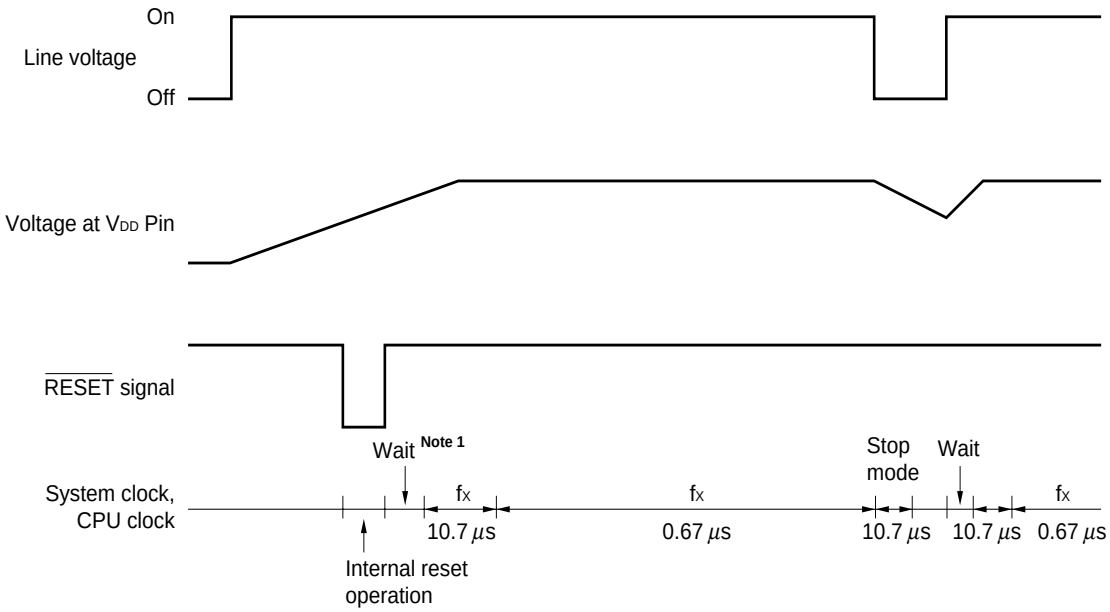
Setting Before Switching		Setting After Switching							
PCC1	PCC0	PCC1	PCC0	PCC1	PCC0	PCC1	PCC0	PCC1	PCC0
		0	0	0	1	1	0	1	1
1	0			1 machine cycle		1 machine cycle		1 machine cycle	
0	1	4 machine cycles				4 machine cycles		4 machine cycles	
1	0	8 machine cycles		8 machine cycles				8 machine cycles	
1	1	16 machine cycles		16 machine cycles		16 machine cycles			

Remark The CPU clock Φ is the clock that is supplied to the internal CPU. This reciprocal of the frequency is the minimum instruction time (defined in this manual as 1 machine cycle).

(2) CPU clock switching procedure

The switching procedure for CPU clock is explained according to **Figure 5-15**.

Figure 5-15. Switching CPU Clock



(f_X = 6.00 MHz)

<1> After the wait time has elapsed **Note 1** for stable oscillation by the $\overline{\text{RESET}}$ signal, the CPU starts operation with the slowest speed (10.7 μs: 6.00 MHz at operation time, 15.3 μs: 4.19 MHz at operation time) of the main system clock.

<2> After a time long enough for the voltage at the V_{DD} pin to rise to a value by which the CPU can operate in the highest speed has elapsed, the contents of the PCC are written and the CPU starts operation in the highest speed.

<3> After the turning off of the commercial power supply is detected by an interrupt input **Note 2** and the time required for PCC to be rewritten and operate at the slowest speed elapses, the STOP mode is entered.

<4> The resumption of the commercial power supply is detected by an interrupt input **Note 2**, and the STOP mode is released. After the wait time (time set by BTM) to ensure a stable oscillation period elapses, the CPU restarts operation at the slowest system clock. After sufficient time elapses so that the voltage at pin V_{DD} rises to the voltage capable of operating at the maximum speed, PCC is rewritten and operates at the fastest speed.

Notes 1. The following two times can be selected by the mask option.
2^{15}/fx (21.8 ms: 6.0 MHz operation, 31.3 ms: 4.19 MHz)
2^{17}/fx (5.46 ms: 6.0 MHz operation, 7.81 ms: 4.19 MHz)
However, μPD75P4308 does not have the mask option and is fixed at 2^{15}/fx.

2. Valid when using INT4

5.2.4 Clock Output Circuit

(1) Clock output circuit configuration

The configuration of the clock output circuit is shown in **Figure 5-16**.

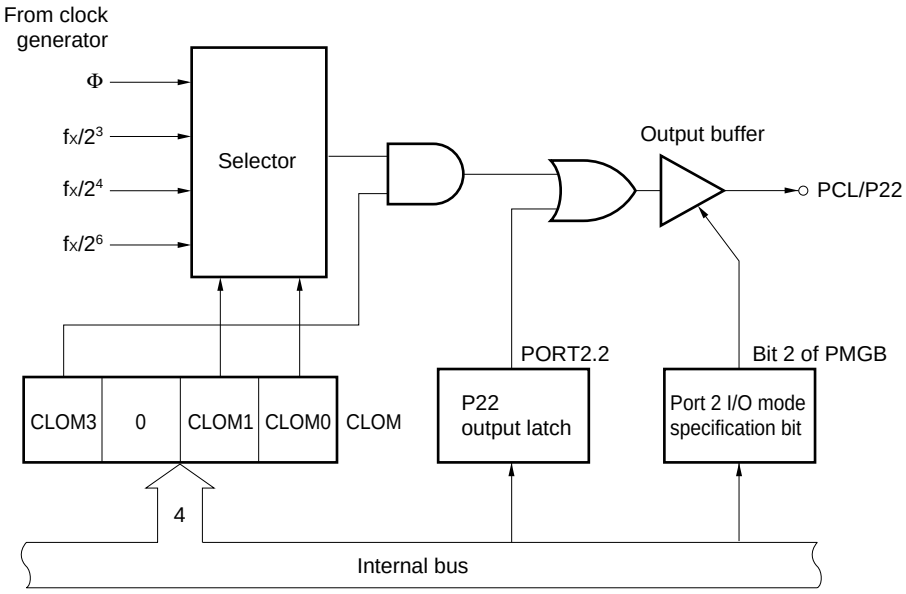
(2) Clock output circuit function

The clock output circuit is provided to output the clock pulses from the P22/PCL pin to the remote control outputs and peripheral LSI's.

The clock pulses must be output in the following steps.

(a) Select a clock output frequency. Prohibit clock output.
(b) Write "0" in the output latch at P22.
(c) Set the I/O mode of the port 2 to output.
(d) Enable clock output.

Figure 5-16. Clock Output Circuit Block Diagram



Remark Special care has been taken in designing the chip so that small-width pulses may not be output when switching clock output enable/disable.

(3) Clock output mode register (CLOM)

The CLOM is a 4-bit register which controls clock output.

It must be set by a 4-bit memory manipulation instruction and cannot be read out.

Example The CPU clock Φ is output from the PCL/P22 pin.

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

CLOM is cleared to “0” by a $\overline{\text{RESET}}$ signal generation and the clock output is disabled.

Figure 5-17. Clock Output Mode Register Format

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

Clock output frequency select bit

When f_X = 6.00 MHz

0	0	Φ output [Note] (1.5 MHz, 750 kHz, 375 kHz, 93.8 kHz)
0	1	$f_X/2^3$ output (750 kHz)
1	0	$f_X/2^4$ output (375 kHz)
1	1	$f_X/2^6$ output (93.8 kHz)

When f_X = 4.19 MHz

0	0	Φ output [Note] (1.05 MHz, 524 kHz, 262 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 the CPU clock selected by the PCC.

Clock output enable/disable bit

0	Output disabled.
1	Output enabled.

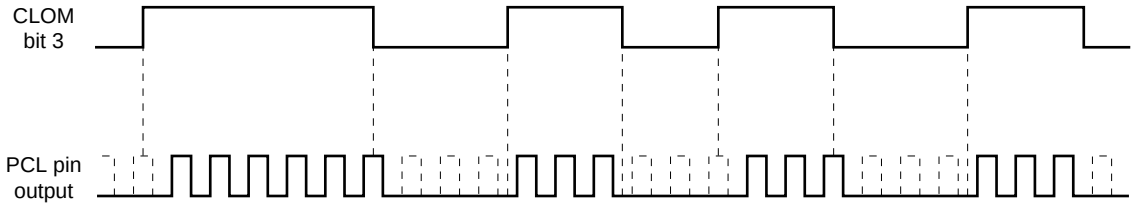
Caution Be sure to set bit 2 of the CLOM to “0”.

(4) Application example of remote control output

The μPD754304 clock output function can be used for remote control. The carrier frequency of remote control output is selected by the clock frequency select bit of the clock output mode register. The pulse output enable/disable is selected by controlling the clock output enable/disable bit by software.

Special attention is paid not to output small-width pulses when switching clock output enable/disable.

Figure 5-18. Application Example of Remote Control Output



5.3 Basic Interval Timer/Watchdog Timer

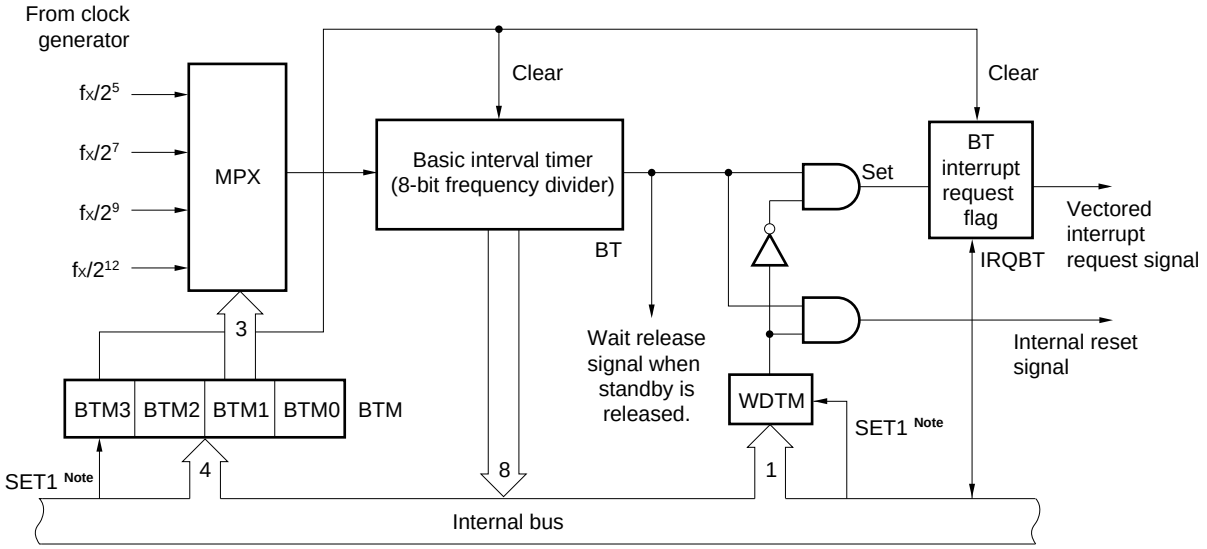
The μPD754304 is provided with the 8-bit basic interval timer/watchdog timer and has the following functions.

(a) Interval timer operation to generate a reference time interrupt
(b) Watchdog timer operation to detect a runaway of program and reset the CPU
(c) Selects and counts the wait time when the standby mode is released
(d) Reads the contents of counting

5.3.1 Basic interval timer/watchdog timer configuration

The configuration of the basic interval timer/watchdog timer is shown in **Figure 5-19**.

Figure 5-19. Basic Interval Timer/Watchdog Timer Block Diagram



Note Instruction execution

5.3.2 Basic interval timer mode register (BTM)

The BTM is a 4-bit register which controls the operations of the basic interval timer (BT).

It is set by a 4-bit memory manipulation instruction.

Bit 3 can be set by a bit manipulation instruction.

Example The interrupt generation interval is set to 1.37 ms (6.00 MHz).

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

When bit 3 is set to “1”, the contents of BT are cleared and the interrupt request flag of the basic interval timer/watchdog timer (IRQBT) is also cleared (the start of the basic interval timer/watchdog timer).

Its contents are cleared to “0” by a $\overline{\text{RESET}}$ signal generation and the interrupt request signal generation interval is set to the longest time.

Figure 5-20. Basic Interval Timer Mode Register Format

Address: F85H

Symbol: BTM

3	2	1	0
BTM3	BTM2	BTM1	BTM0

When f_X = 6.00 MHz

			Clock input specification	Interrupt interval time (wait time when standby is released)
0	0	0	$f_X/2^{12}$ (1.46 kHz)	$2^{20}/f_X$ (175 ms)
0	1	1	$f_X/2^{9}$ (11.7 kHz)	$2^{17}/f_X$ (21.8 ms)
1	0	1	$f_X/2^{7}$ (46.9 kHz)	$2^{15}/f_X$ (5.46 ms)
1	1	1	$f_X/2^{5}$ (188 kHz)	$2^{13}/f_X$ (1.37 ms)
Other than the above			Setting prohibited.	–

When f_X = 4.19 MHz

			Clock input specification	Interrupt interval time (wait time when standby is released)
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)	$2^{15}/f_X$ (7.81 ms)
1	1	1	$f_X/2^{5}$ (131 kHz)	$2^{13}/f_X$ (1.95 ms)
Other than the above			Setting prohibited.	–

Start control bit of basic interval timer/watchdog timer

The basic interval timer/watchdog timer starts by writing “1”

(The counter and interrupt request flag are cleared).

Reset to "0" when the operation starts.

5.3.3 Watchdog timer enable flag (WDTM)

The WDTM is a flag which enables reset signal generation by overflow.

It is set by a bit manipulation instruction. Once it is set, it cannot be cleared by an instruction.

Example Setting of watchdog timer

```
SEL     MB15        ; or CLR1 MBE
SET1    WDTM
  :
SET1    BTM.3       ; Bit 3 of BTM is set to "1".
```

The contents are cleared to "0" by a $\overline{\text{RESET}}$ signal generation.

Figure 5-21. Watchdog Timer Enable Flag (WDTM) Format

Address

F8BH.3 WDTM

0	BT mode: Sets the IRQBT by the overflow of basic interval timer (BT).
1	WT mode: Generates an internal Reset signal by the overflow of basic interval timer (BT).

5.3.4 Basic interval timer (BT) operations

When WDTM is set to “0”, the interrupt request flag (IRQBT) is set by the overflow of the basic interval timer (BT) and it operates as the interval timer. The basic interval timer (BT) always increments by the clock sent from the clock generator and the counting operation cannot be stopped.

Four interrupt generation intervals can be set by BTM (See **Figure 5-20**).

By setting bit 3 of the BTM to “1”, the basic interval timer (BT) and IRQBT can be cleared (start specification as the interval timer).

The counting status can be read out from the basic interval timer (BT) by an 8-bit manipulation instruction. Note that data cannot be entered.

Perform the timer operations as follows. (These can be set at the same time.)

<1> Set an interval time to the BTM.
<2> Set bit 3 of BTM to “1”.

Example Interrupts are generated every 1.37 ms (during 6.00 MHz operation).

```
SET1    MBE
SEL     MB15
MOV     A, #1111B
MOV     BTM, A      ; Time setting and start
EI                  ; Interrupt enabled
EI      IEBT        ; BT interrupt enabled
```

5.3.5 Watchdog timer operations

When WDTM is set to “1” in the basic interval timer/watchdog timer, it performs as the watchdog timer wherein ★ an internal reset signal is generated by an overflow of the basic interval timer (BT). No reset signal, however, is generated during the oscillation wait time following the STOP instruction has been released (When the WDTM is set to “1” once, it can be cleared only by resetting). The basic interval timer (BT) always increments by the clock sent from the clock generator and its counting operation cannot be stopped.

In the watchdog timer mode, program runaway is detected by utilizing the interval timer wherein the BT overflows. Four intervals can be selected by bits 0-2 of the BTM (See **Figure 5-20**). Select one of them suitable for user’s system. Set an interval and divide the program so that it can be executed in the interval and execute the instruction which clears the BT at the ends of the divided program. If the instruction which clears the BT is not reached within the time set (that is, the program is not executed normally = runaway), the BT overflows and an internal reset signal is generated to forcibly terminate the program. Namely, it indicates a program runaway has occurred and been detected.

Set the watchdog timer with the following procedure.

<1> Set the interval in the BTM.
<2> Set bit 3 of the BTM to “1”. } Initialization
<3> Set WDTM to “1”.
<4> Then, set bit 3 of the BTM to “1” within the interval.
The above steps <1> and <3> can be set at the same time.

Example The basic interval timer/watchdog timer is used as the watchdog timer with 5.46 ms (during 6.00 MHz operation).

The program is divided into several modules which end within the time set for the BTM (5.46 ms) and the BT is cleared at the ends of the modules. In case a runaway occurs, the BT is not cleared within the time set, therefore it overflows and an internal Reset signal is generated.

Initialization :

```
SET1    MBE
SEL     MB15
MOV     A, #1101B
MOV     BTM, A      : Sets time and starts
SET1    WDTM        : Enables watchdog timer
  ⋮
```

(Then, the bit 3 of the BTM is set to “1” every 5.46 ms.)

Module 1 :

```
  ⋮
SET1    MBE
SEL     MB15
SET1    BTM.3
```

Processing is completed within 5.46 ms.

Module 2 :

```
  ⋮
SET1    MBE
SEL     MB15
SET1    BTM.3
  ⋮
```

Processing is completed within 5.46 ms

5.3.6 Other functions

The basic interval timer/watchdog timer has the following functions regardless of the basic interval timer (BT) operation and watchdog timer operation.

<1> Selects and counts the wait time after the standby mode is released.
<2> Reads the contents of counter.

(1) Selects and counts the wait time after the STOP mode is released

At the time the STOP mode is released, the system clock needs time for stablilizing oscillation. For this purpose, the wait function is provided for the CPU to halt its operation until the basic interval timer (BT) overflows. The wait time after a $\overline{\text{RESET}}$ signal generation is fixed by the mask option. However, it can be selected by setting the BTM when the STOP mode is released by an interrupt generated. In this case, the wait time is the same as the interval shown in **Figure 5-24**. The BTM must be set before the STOP mode is set. For details, refer to **CHAPTER 7 STANDBY FUNCTION**.

Example The wait time is set to 5.46 ms at the time the STOP mode is released by an interrupt (during 6.00 MHz operation).

```
SET1    MBE
SEL     MB15
MOV     A, #1101B
MOV     BTM, A      ; Sets time
STOP                ; Sets the STOP mode
NOP
```

(2) Reads the counting operation

The basic interval timer (BT) can read the counting status by an 8-bit manipulation instruction. Note that data cannot be entered.

Caution When reading the counting contents of the BT, execute the read instruction twice in order not to read uncertain data while counting continues. If the two values read out are reasonable, take the last one as the count data. If they are completely different, try the operation again.

Examples 1. The counting contents of the BT is read out.

```
        SET1    MBE
        SEL     MB15
        MOV     HL, #BT     ; Sets the address of BT to HL.
LOOP :  MOV     XA, @HL     ; First reading
        MOV     BC, XA
        MOV     XA, @HL     ; Second reading
        SKE     XA, BC
        BR      LOOP
```

2. The high-level width of the pulses which are input to an INT4 interrupt (both edges are detected) is set. The pulse width is assumed not to exceed the value set for the BT. The value set for the BTM is assumed to be 5.46 ms or more (during 6.00-MHz operation).

<INT4 interrupt routine (MBH = 0)>

```
LOOP :  MOV     XA, BT      ; First reading
        MOV     BC, XA      ; Stores data
        MOV     XA, BT      ; Second reading
        SKE     A, C
        BR      LOOP
        MOV     A, X
        SKE     A, B
        BR      LOOP
        SKT     PORT0.0     ; P00=1?
        BR      AA          ; NO
        MOV     XA, BC      ; Stores data in the data memory
        MOV     BUFF, XA
        CLR1    FLAG        ; Data exists. Clears the flag.
        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    ; Stores data
        SET1    FLAG        ; Data exists. Sets the flag.
        RETI
```

5.4 Timer/Event Counter

The μPD754304 has two channels of timer/event counters. The timer/event counter has the following functions.

(a) Programmable interval timer operation
(b) Square wave output of any frequency to the PTOn pin.
(c) Event counter operation
(d) Divides the frequency of signal input via the TIn pin to 1-Nth of the original signal and outputs the divided frequency to the PTOn pin (frequency divider operation).
(e) Supplies the serial shift clock to the serial interface circuit.
(f) Calls the counting status.

The timer/event counter operates in the following two modes as set by the mode register.

Table 5-6. Operation Modes of Timer/Event Counter

Mode \ Channel	Channel 0	Channel 1
8-bit timer/event counter mode	Usable	Usable
16-bit timer/event counter mode	Usable	

5.4.1 Configuration of timer/event counter

Figures 5-22 and **5-23** shows the configuration of the timer/event counter.

Figure 5-22. Timer/Event Counter Block Diagram (channel 0)

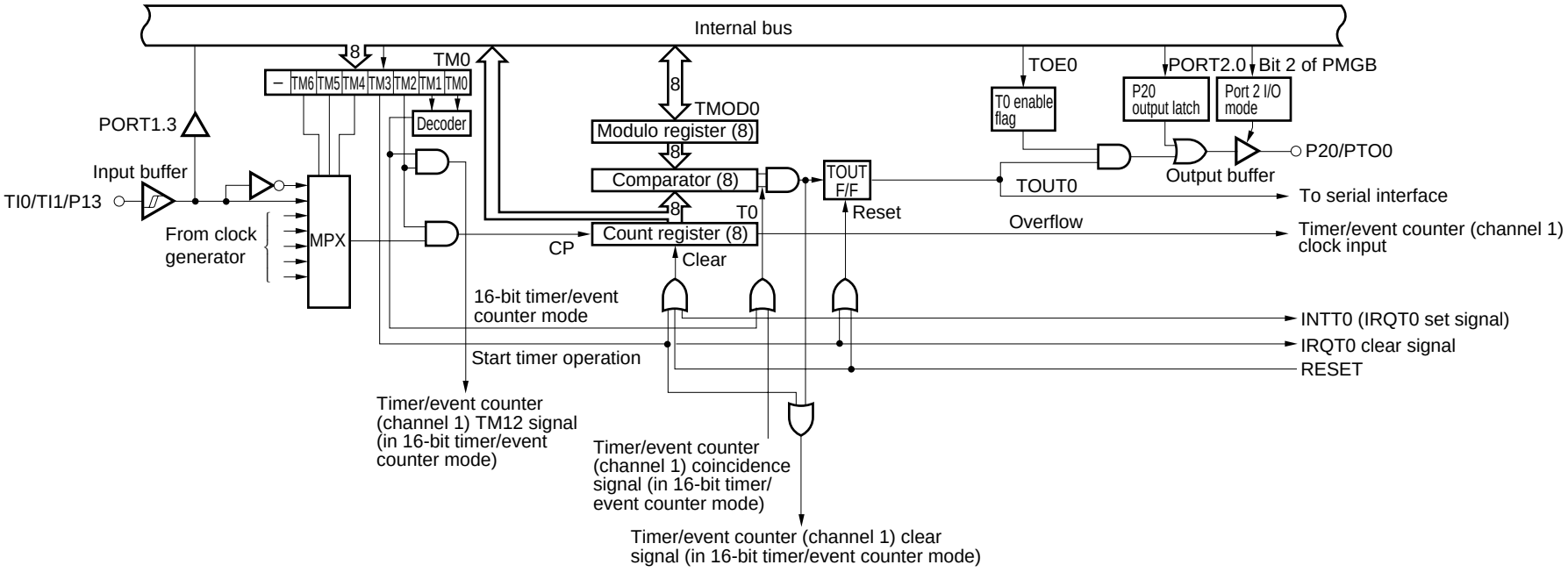
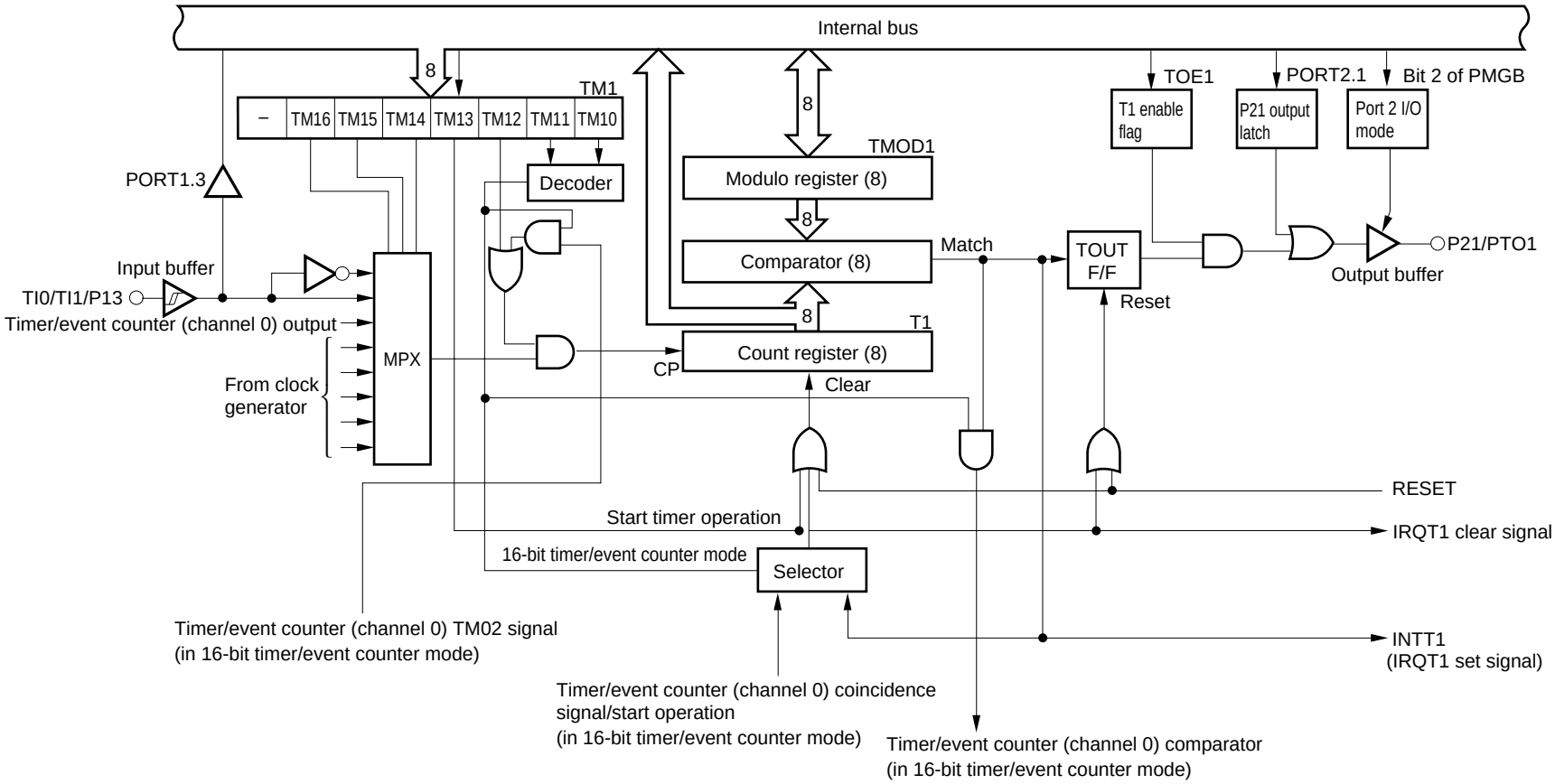


Figure 5-23. Timer/Event Counter Block Diagram (channel 1)



(1) Timer/event counter mode register (TM0, TM1)

The mode register (TMn) is an 8-bit register which controls the timer/event counter.

Its format is shown in **Figures 5-24** and **5-25**.

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

Bit 3 is a timer start bit and can be operated bit-wise. It is automatically reset to "0" when the timer operation starts.

All the bits of the timer/event counter mode register are cleared to "0" by a $\overline{\text{RESET}}$ signal generation.

Examples 1. Start the timer in the interval timer mode of CP = 5.86 kHz (during 6.00 MHz operation).

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

2. Restart the timer according to the setting of the timer/event counter mode register.

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

Figure 5-24. Timer/Event Counter Mode Register (channel 0) Format

Address	7	6	5	4	3	2	1	0	Symbol
FA0H	—	TM06	TM05	TM04	TM03	TM02	TM01	TM00	TM0

Count pulse (CP) selection bit

(When f_X = 6.00 MHz)

TM06	TM05	TM04	Count pulse (CP)
0	0	0	TI0 rising edge
0	0	1	TI0 falling edge
0	1	1	$f_X/2^2$ (1.5 MHz)
1	0	0	$f_X/2^{10}$ (5.86 kHz)
1	0	1	$f_X/2^8$ (23.4 kHz)
1	1	0	$f_X/2^6$ (93.8 kHz)
1	1	1	$f_X/2^4$ (375 kHz)
Other than above			Setting prohibited.

(When f_X = 4.19 MHz)

TM06	TM05	TM04	Count pulse (CP)
0	0	0	TI0 rising edge
0	0	1	TI0 falling edge
0	1	1	$f_X/2^2$ (1.05 MHz)
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.

Timer start indication bit

TM03	When 1 is written into the bit, the counter and IRQT0 flag are cleared. If bit 2 is set to 1, count operation is started.

Operation mode

TM02	Count operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM01	TM00	Mode
0	0	8-bit timer/event counter mode
1	0	16-bit timer/event counter mode
Other than above		Setting prohibited.

Figure 5-25. Timer/Event Counter Mode Register (Channel 1) Format

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	—	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1

Count pulse (CP) select bit

(When f_X = 6.00 MHz)

TM16	TM15	TM14	Count pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Overflow of timer/event counter channel 2
0	1	1	$f_X/2^2$ (1.5 MHz)
1	0	0	$f_X/2^{12}$ (1.46 kHz)
1	0	1	$f_X/2^{10}$ (5.86 kHz)
1	1	0	$f_X/2^8$ (23.4 kHz)
1	1	1	$f_X/2^6$ (93.8 kHz)

(When f_X = 4.19 MHz)

TM16	TM15	TM14	Count pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Overflow of timer/event counter channel 2
0	1	1	$f_X/2^2$ (1.05 MHz)
1	0	0	$f_X/2^{12}$ (1.02 kHz)
1	0	1	$f_X/2^{10}$ (4.09 kHz)
1	1	0	$f_X/2^8$ (16.4 kHz)
1	1	1	$f_X/2^6$ (65.5 kHz)

Timer start indication bit

TM13	When 1 is written into the bit, the counter and IRQT1 flag are cleared. If bit 2 is set to 1, count operation is started.

Operation mode

TM12	Count operation
0	Stop (retention of count contents)
1	Count operation

Operation mode select bit

TM11	TM10	Mode
0	0	8-bit timer/event counter mode
1	0	16-bit timer/event counter mode
Other than the above		Setting prohibited.

(2) Timer/event counter output enable flag (TOE0, TOE1)

The timer/event counter output enable flag (TOE0, TOE1) controls the output enable/disable to the PTO0 and PTO1 pins in the timer out F/F (TOUT F/F) status.

The timer out F/F flips by the match signal sent from the comparator. When bit 3 of the timer/event counter mode register (TM0, TM1) is set to “1”, the timer out F/F is cleared to “0”.

TOE0, TOE1, and timer out F/F are cleared to “0” by a $\overline{\text{RESET}}$ signal generation.

Figure 5-26. Timer/Event Counter Output Enable Flag Format

Address	3	2	1	0	
FA2H	TOE0	—	—	—	Channel 0
FAAH	TOE1	—	—	—	Channel 1

Timer/event counter output enable flag (W)

TOEn	Timer output
0	Disabled (output the low level).
1	Enabled.

5.4.2 8-bit timer/event counter mode operation

It is used as an 8-bit timer/event counter in this mode. It performs an 8-bit programmable interval timer and event counter operation.

(1) Register setting

The following three registers and one flag are used in the 8-bit timer/event counter mode.

- Timer/event counter mode register (TMn)
- Timer/event counter count register (Tn)
- Timer/event counter modulo register (TMODn)
- Timer/event counter output enable flag (TOEn)

(a) Timer/event counter mode register (TMn)

When the 8-bit timer/event counter mode is used, TMn must be set as shown in **Figure 5-27** (for the format of the TMn, see **Figures 5-24** to **5-25**).

The TMn is manipulated by an 8-bit manipulation instruction. Bit 3 is a timer start indication bit and can be manipulated bit-wise and is automatically cleared to “0” when the timer starts.

The TMn is cleared to 00H when an internal reset signal is generated.

Figure 5-27. Timer/Event Counter Mode Register Setup (8-bit) (1/2)

(a) In the case of timer/event counter (channel 0)

Address	7	6	5	4	3	2	1	0	Symbol
FA0H	—	TM06	TM05	TM04	TM03	TM02	TM01	TM00	TM0

Count pulse (CP) selection bit

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

Timer start indication bit

TM03	When “1” is written into the bit, the counter and IRQT0 flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM02	Count operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM01	TM00	Mode
0	0	8-bit timer/event counter mode

Figure 5-27. Timer/Event Counter Mode Register Setup (8-bit) (2/2)

(b) In the case of timer/event counter (channel 1)

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	—	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1

Count pulse (CP) selection bit

TM16	TM15	TM14	Count pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Timer/event counter channel 0 overflow
0	1	1	$f_X/2^2$
1	0	0	$f_X/2^{12}$
1	0	1	$f_X/2^{10}$
1	1	0	$f_X/2^8$
1	1	1	$f_X/2^6$

Timer start indication bit

TM13	When “1” is written to the bit, the counter and IRQT1 flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM12	Count operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM11	TM10	Mode
0	0	8-bit timer/event counter mode

(b) Timer/event counter output enable flag (TOEn)

For 8-bit timer/event counter output, set TOEn as shown in **Figure 5-28**.

Figure 5-28. Format of the Timer/Event Counter Output Enable Flag

Address	3	2	1	0	
FA2H	TOE0	—	—	—	Channel 0
FAAH	TOE1	—	—	—	Channel 1

Timer/event counter output enable flag (W)

TOEn	Timer output
0	Disable (output low level).
1	Enable.

(2) 8-bit timer/event counter time setting

[Time setting value] (count-up cycle) is found by dividing [modulo register content + 1] by [count pulse (CP) frequency] selected by setting the mode register.

$$T\ (sec) = \frac{n+1}{f_{CP}} = (n + 1) \times (Resolution)$$

T (sec) : Time value to be set in the timer (seconds)
f_{CP} (Hz): Count pulse frequency (Hz)
n : Modulo register content ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQTn) is generated at the intervals set in the timer. **Table 5-7** lists the resolution and maximum allowable time setting (that is, time when FFH is set in the modulo register) for each count pulse to the 8-bit timer/event counter.

★ **Table 5-7. Resolution and Maximum Allowable Time Setting (8-bit timer mode)**

(a) When timer/event counter (channel 0)

Mode register			During 6.00-MHz operation		During 4.19-MHz operation	
TM06	TM05	TM04	Resolution	Max. time setting	Resolution	Max. time setting
0	1	1	667 ns	171 μs	952 ns	244 μs
1	0	0	171 μs	43.7 ms	244 μs	62.5 ms
1	0	1	42.7 μs	10.9 ms	61.0 μs	15.6 ms
1	1	0	10.7 μs	2.73 ms	15.3 μs	3.91 ms
1	1	1	2.67 μs	683 μs	3.82 μs	977 μs

(b) When timer/event counter (channel 1)

Mode register			During 6.00-MHz operation		During 4.19-MHz operation	
TM16	TM15	TM14	Resolution	Max. time setting	Resolution	Max. time setting
0	1	1	667 ns	171 μs	952 ns	244 μs
1	0	0	683 μs	175 ms	980 μs	250 ms
1	0	1	171 μs	43.7 ms	244 μs	62.5 ms
1	1	0	42.7 μs	10.9 ms	61.0 μs	15.6 ms
1	1	1	10.7 μs	2.73 ms	15.3 μs	3.91 ms

(3) 8-bit timer/event counter operation

The 8-bit timer/event counter operates as follows.

Figure 5-29 shows the configuration of the 8-bit timer/event counter.

<1> The count pulse (CP) is selected by setting the mode register (TMn) and is input to the count register (Tn).

<2> The Tn is compared with the modulo register (TMODn), and if they are equal, a match signal is generated and the interrupt request flag (IRQTn) is set. At the same time, the timer out flip-flop (TOUT F/F) flips.

Figure 5-30 is a timing chart of the 8-bit timer/event counter.

The 8-bit timer/event counter normally begins operation in the following procedure.

<1> Set a count in the TMODn.

<2> Set the operating mode, count pulse, and start indication in the TMn.

Caution Set a value other than 00H in the modulo register (TMODn).

When using the timer/event counter output pin (PTOn), set the dual function pin P2n as follows.

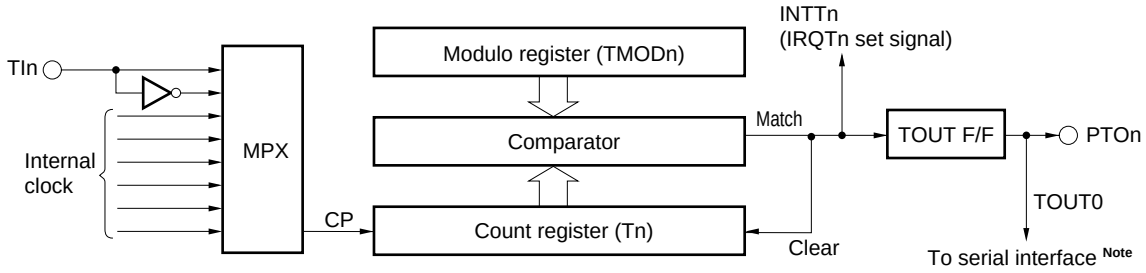
<1> Clear the output latch of P2n.

<2> Set port 2 to the output mode.

<3> Make a status wherein the pull-up resistor is not incorporated in port 2.

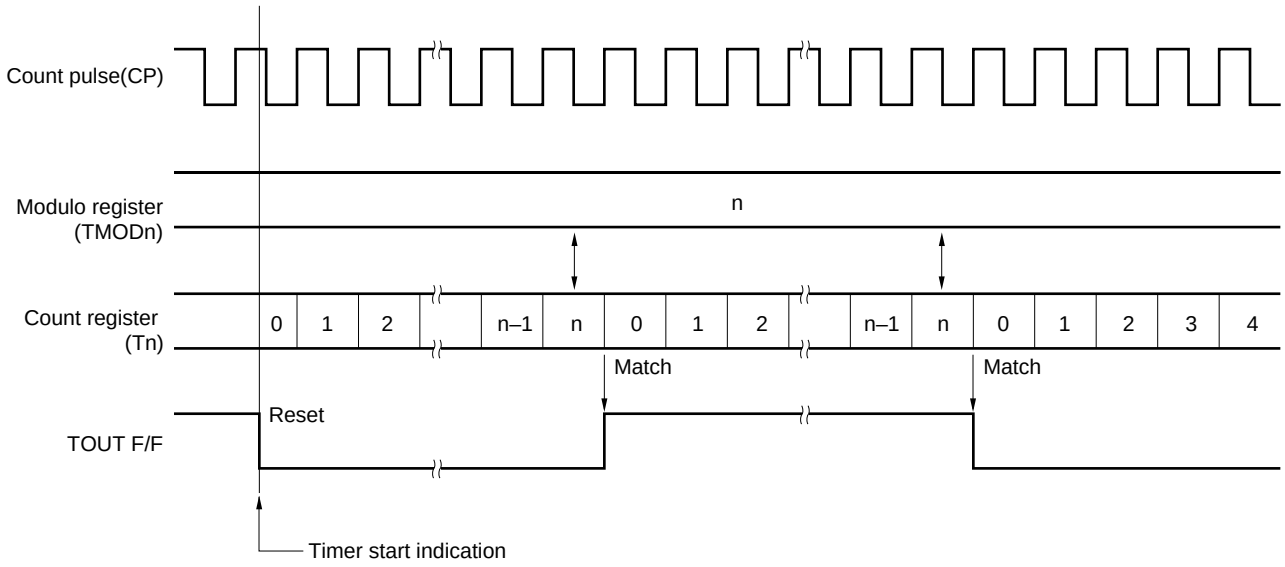
<4> Set the timer/event counter output enable flag (TOEn) to 1.

Figure 5-29. Configuration of Timer/Event Counter



Note Only the channel 0 signal of the timer/event counter can be output to the serial interface.

Figure 5-30. Count Operation Timing



(4) 8-bit timer/event counter mode application

(a) Use as an interval timer which causes an interrupt to occur at 50 ms intervals (@f_X = 4.19 MHz).

- Set the high-order four bits of the mode register (TMn) to 0100B to select the longest setup time 62.5 ms.
- Set the low-order four bits of the TMn to 1100B.
- The value set in the modulo register (TMODn) is as follows:

$$\frac{50\ \text{ms}}{244\ \mu\text{s}} = 205,\ \ 205 - 1 = \text{CCH}$$

<Program example>

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

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

(b) Generate an interrupt when the number of pulses input from the TIn pin reaches 100.
(The pulses are active high.)

- Set the high-order four bits of the mode register (TMn) to 0000 to select rising edge.
- Set the low-order four bits of the TMn to 1100B.
- The value set in the modulo register is 100 – 1 = 63H.

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #100 – 1
MOV     TMODn, XA       ; Set modulo
MOV     XA, #00001100B
MOV     TMn, XA         ; Set mode, start count
EI
EI      IETn            ; Enable INTTn
```

5.4.3 16-bit timer/event counter mode operation

Used as a 16-bit timer/event counter in this mode. It performs 16-bit programmable interval timer and event counter. When it is used in the 16-bit timer/event counter mode, the channel 0 and channel 1 of the timer/event counter are used in combination.

(1) Register setting

The following six registers are used in the 16-bit timer/event counter mode.

- Timer/event counter mode registers (TM0, TM1)
- Timer/event counter count registers (T0, T1)
- Timer/event counter modulo registers (TMOD0, TMOD1)
- Timer/event counter enable flag (TOE0)

(a) Timer/event counter mode registers (TM0, TM1)

When using the 16-bit timer/event counter mode, set the TM0 and TM1 as follows. For the formats of the TM0 and TM1, see **Figure 5-24. Timer/Event Counter Mode Register (channel 1) Format** and **Figure 5-25. Timer/Event Counter Mode Register (channel 2) Format**, respectively.

The TM0 and TM1 are manipulated by 8-bit manipulation instructions. Bit 3 (TM03) of TM0 is a timer start indication bit and can be manipulated bit-wise and is automatically cleared to 0 when the timer starts.

The TM0 and TM1 are cleared to 00H by an internal reset signal.

The flag indicated by the full lines expresses a bit used in the 16-bit timer/event counter mode.

The flag indicated by the broken lines must not be used in the 16-bit timer/event counter mode. Set 0.

Figure 5-31. Timer/Event Counter Mode Register Setup (16-bit mode)

Address	7	6	5	4	3	2	1	0	Symbol
FA0H	—	TM06	TM05	TM04	TM03	TM02	TM01	TM00	TM0
FA8H	—	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1 Note

Count pulse (CP) selection bit (n = 0, 1)

TMn6	TMn5	TMn4	TM0	TM1 Note
0	0	0	TI0 rising edge	TI1 rising edge
0	0	1	TI0 falling edge	TI1 falling edge
0	1	0	Setting prohibited	Overflow of count register (T0)
0	1	1	$f_X/2^2$	$f_X/2^2$
1	0	0	$f_X/2^{10}$	$f_X/2^{12}$
1	0	1	$f_X/2^8$	$f_X/2^{10}$
1	1	0	$f_X/2^6$	$f_X/2^8$
1	1	1	$f_X/2^4$	$f_X/2^6$

Timer start indication bit

TM03	When “1” is written to the bit, the counter and IRQTn flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM02	Count operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM11	TM10	TM01	TM00	Mode
1	0	1	0	16-bit timer/event counter mode

Note In the 16-bit timer/event counter mode, set TM1 = 00100010B.

(b) Timer/event counter output enable flag (TOE0)

For the 16-bit timer/event counter output, set TOE0 as shown in **Figure 5-32**.

Figure 5-32. Format of the Timer/Event Counter Output Enable Flag

Address	3	2	1	0	
FA2H	TOE0	—	—	—	Channel 0

Timer/event counter output enable flage (W)

TOE0	Timer output
0	Disabled (outputs the low level signal).
1	Enabled.

(2) 16-bit timer/event counter time setting

[Time setting value] (count-up cycle) is found by dividing [modulo register content + 1] by [count pulse (CP) frequency] selected by setting the mode register.

$$T\ (sec) = \frac{n+1}{f_{CP}} = (n + 1) \times (Resolution)$$

T (sec) : Time value to be set in the timer (seconds)
f_{CP} (Hz) : Count pulse frequency (Hz)
n : Modulo register content ($n \neq 0$)

Once the timer is set, an interrupt request signal (IRQT0) is generated at the intervals set in the timer.

Table 5-8 lists the resolution and maximum allowable time setting (that is, time when FFH is set in the modulo register) for each count pulse to the timer/event counter.

Table 5-8. Resolution and Maximum Allowable Time Setting (16-bit timer mode) ★

When timer/event counter (channel 0 and 1)

Mode register			During 6.00-MHz operation		During 4.19-MHz operation	
TM06	TM05	TM04	Resolution	Max. time setting	Resolution	Max. time setting
0	1	1	667 ns	43.7 ms	952 ns	62.5 ms
1	0	0	171 μs	11.2 s	244 μs	16.0 s
1	0	1	42.7 μs	2.80 s	61.0 μs	4.00 s
1	1	0	10.7 μs	699 ms	15.3 μs	1.00 s
1	1	1	2.67 μs	175 ms	3.82 μs	250 ms

Cautions 1. In the 16-bit timer/event counter mode, set TM16 = 0, TM15 = 1, and TM14 = 0.

2. The resolution is set by CP in timer channel 0.

(3) 16-bit timer/event counter operation

The 16-bit timer/event counter operates as follows.

Figure 5-33 shows the configuration of the 16-bit timer/event counter.

<1> The count pulse (CP) is selected by setting the mode registers (TM0 and TM1), and is input to the count register (T0). The overflow of the T0 is input to the count register (T1).

<2> The contents of the T0 and those of the modulo register (TMOD0) are compared, and if they are equal, a match signal is generated.

<3> The contents of the T1 are compared with those of the modulo register (TMOD1), and if they are equal, a match signal is generated.

<4> If the match signals of <2> and <3> above are the same, an interrupt request flag (IRQT0) is set. At the 16-bit same time, the timer out flip-flop (TOUT F/F) flips.

Figure 5-34 shows the timing chart of the 16-bit timer/event counter operation.

The 16-bit timer/event counter normally starts the operation in the following procedure.

<1> Set the high-order 8-bits of the count expressed by a 16-bit width in the TMOD1.

<2> Set the low-order 8-bits of the count expressed by a 16-bit width in the TMOD0.

<3> Set the operating mode and count pulse in the TM1. (TM1 = 00100010B)

<4> Set the operating mode, count pulse, and start indication in the TM0.

Cautions 1. Set a value other than 00H to the modulo register (TMOD0).

2. Set the timer/event counter interrupt enable flag (IET1) to 0 (disabled).

When using the timer/event counter output pin (PTO0), set the dual function pins P20 as follows.

<1> Clear the output latch of P20.

<2> Set port 2 to the output mode.

<3> Make a status wherein the port 2's on-chip pull-up resistor is not incorporated.

<4> Set the timer/event counter output enable flag (TOE0) to 1.

Figure 5-33. 16-bit Timer/Event Counter Operation Configuration

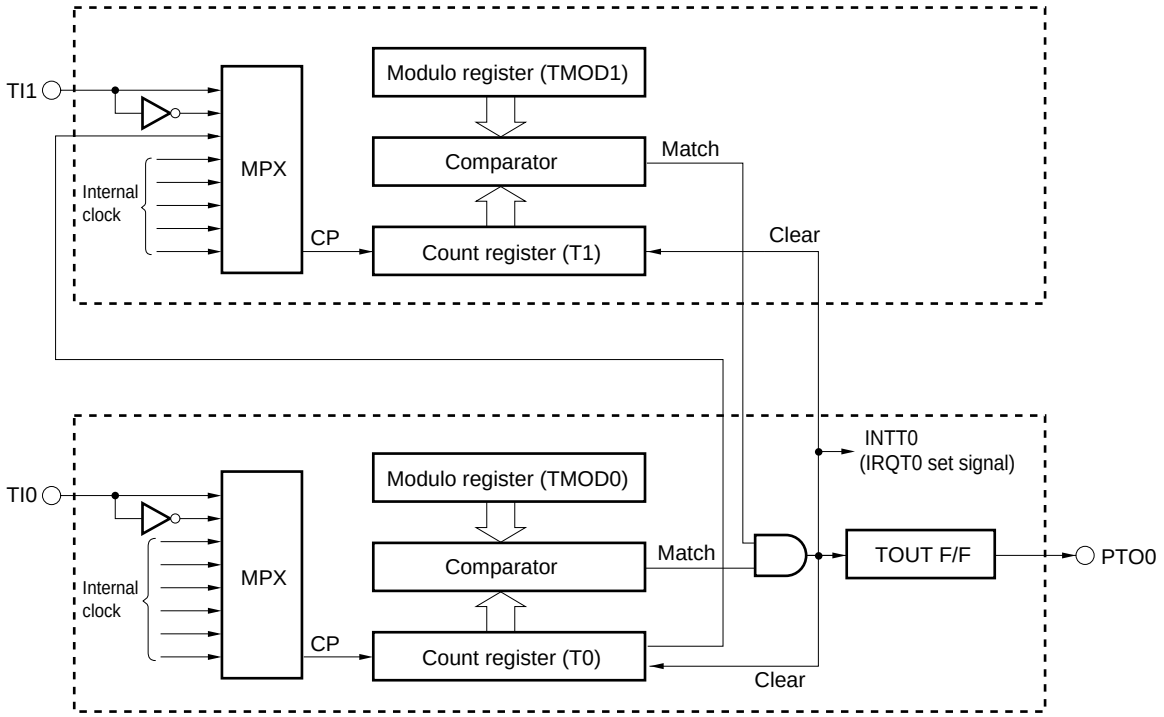
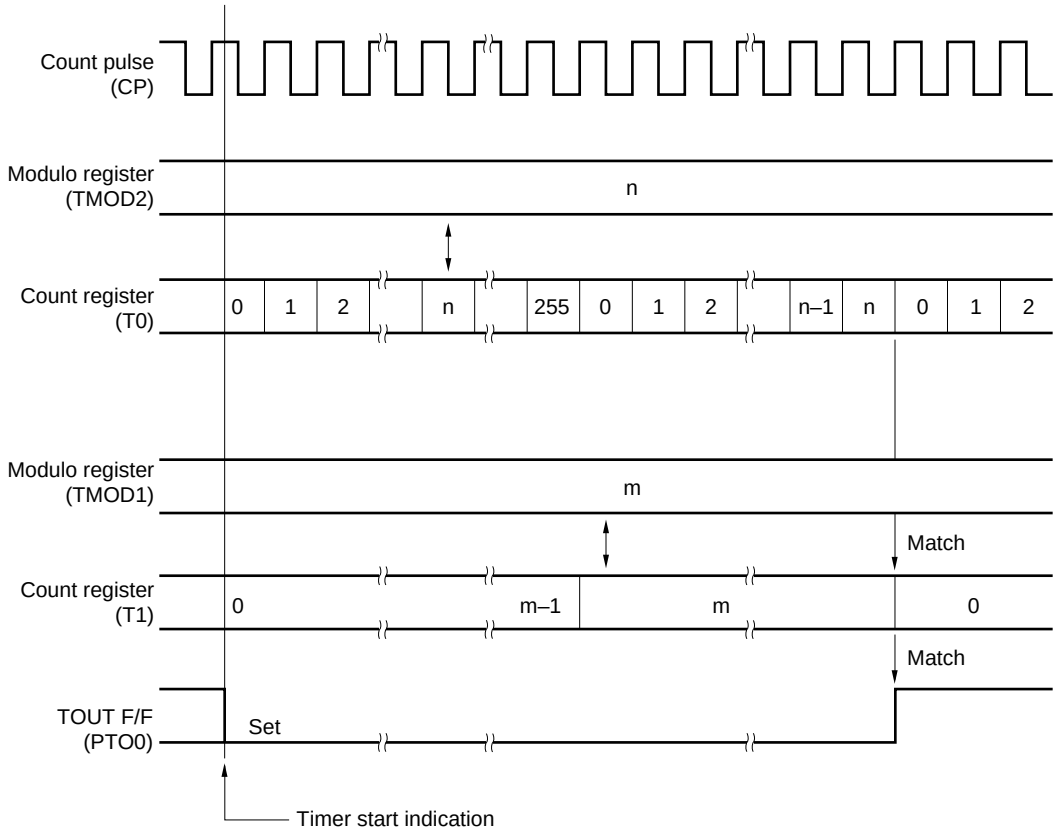


Figure 5-34. Count Operation Timing



(4) 16-bit timer/event counter mode application

(a) Application used as an interval timer generating interrupts every 5 seconds (@ f_X = 4.19 MHz)

- Set the high-order 4-bits of the mode register (TM1) to 0010B and select the overflow of the count register (T0).
- Set the high-order 4-bits of the TM0 to 0100B and select the longest setup time 16.0 sec.
- Set the low-order 4-bits of the TM1 to 0010B and select the 16-bit timer/event counter mode.
- Set the low-order 4-bits of the TM0 to 1110B and select the 16-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo registers (TMOD0, TMOD1) as follows.

$$\frac{5 \text{ sec}}{244\ \mu s} = 20491.8,\ \ 20492 - 1 = 500BH$$

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #050H
MOV     TMOD1, XA       ; Sets the modulo (for high-order 8-bits).
MOV     XA, #00BH
MOV     TMOD0, XA       ; Sets the modulo (for low-order 8-bits).
MOV     XA, #00100010B
MOV     TM1, XA         ; Sets the mode.
MOV     XA, #01001110B
MOV     TM0, XA         ; Sets the mode and starts the timer.
DI      IET1            ; Disables the timer (channel 1) interrupts.
EI                      ; Enables the interrupts.
EI      IET0            ; Enables the timer (channel 0) interrupts.
```

Remark In this example TI0/TI1 can be used as the input pins.

(b) When the pulse (input from the TI0 pin) count reaches 1000, the interrupts are generated
(The pulses are active high).

- Set the high-order 4 bits of the mode register (TM1) to 0010B and select the overflow of the count register (T0).
- Set the high-order 4 bits of the TM0 to 0000B and select the rising edge of the TI0 input.
- Set the low-order 4 bits of the TM1 to 0010B and select the 16-bit timer/event counter mode.
- Set the low-order 4 bits of the TM0 to 1110B and select the 16-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo registers (TMOD0, TMOD1) to 1000 – 1 = 999 = 03E7H. Set the TMOD0 to 03H. Set the TMOD1 to E7H.

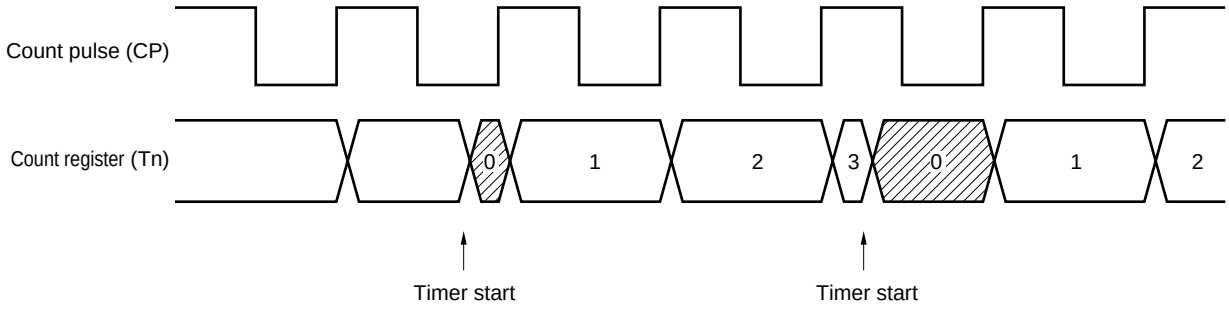
<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #003H
MOV     TMOD1, XA       ; Sets the modulo (for high-order 8 bits).
MOV     XA, #0E7H
MOV     TMOD0, XA       ; Sets the modulo (for low-order 8 bits).
MOV     XA, #00100010B
MOV     TM1, XA         ; Sets the mode.
MOV     XA, #00001110B
MOV     TM0, XA         ; Sets the mode and starts the timer.
DI      IET1            ; Disables the timer (channel 1) interrupts.
EI
EI      IET0            ; Enables the timer (channel 0) interrupts.
```

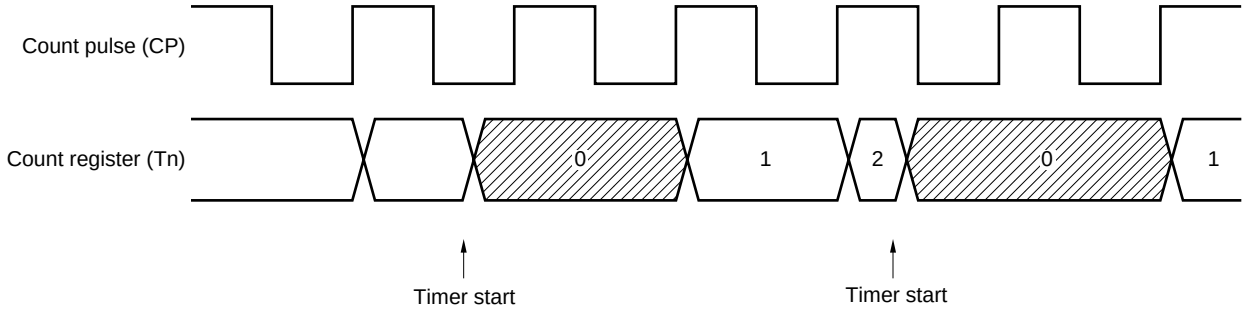
5.4.4 Notes on using the timer/event counter

(1) Error when starting the timer

During the time from the timer start (bit 3 of the TMn is set to “1”) to the match signal generation, an error of one count pulse (CP) at maximum is produced with respect to the value obtained by the formula: (value set in modulo register + 1) × resolution. This is because the count register Tn is cleared asynchronously with the CP as shown below.



When the frequency of CP is one machine cycle or more, the time from the timer start (bit 3 of the TMn is set to “1”) to the match signal generation has a discrepancy of two clock pulses at maximum to the value obtained by the formula: (value set in modulo register + 1) × resolution. This is because the Tn is cleared asynchronously with the CP based on the CPU clock as shown below.



(2) Caution on starting the timer

The count register Tn and interrupt request flag IRQTn are always cleared when the timer starts (bit 3 of the TMn is set to “1”). On the other hand, when the timer is operating and the IRQTn is set and the timer starts at the same timing, the IRQTn may not be able to be cleared. This does not cause trouble when the IRQTn is used as a vectored interrupt. However, when the IRQTn is tested, it appears to be set although the timer has started. Therefore, when the timer starts at the timing when the IRQTn may be set high, the timer must stop (bit 2 of the TMn is set to “0”) and then restart or the timer start operation must be done twice.

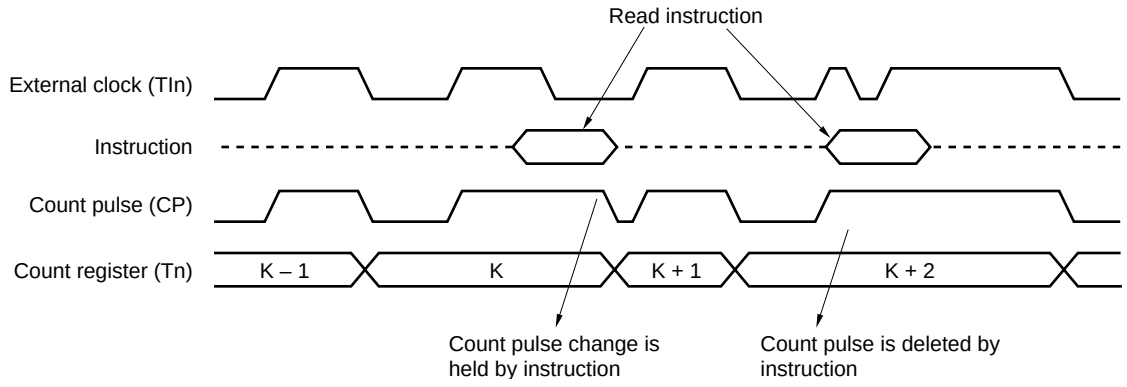
Example Timer start at the timing when the IRQTn may be set high

```
SEL     MB15
MOV     XA, #0
MOV     TMn, XA         ; Timer stop
MOV     XA, #4CH
MOV     TMn, XA         ; Restart
or
SEL     MB15
SET1    TMn.3
SET1    TMn.3           ; Restart
```

(3) Error when reading the count register

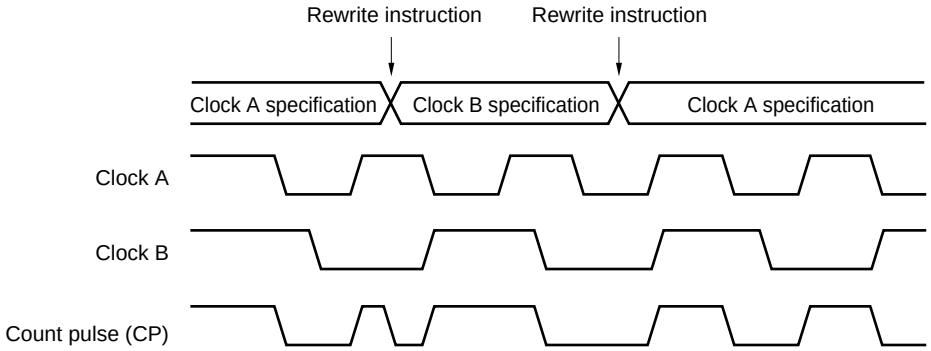
The count register (Tn) can be read any time by an 8-bit data memory operation instruction. When the instruction is being executed, the count pulse (CP) does not change and the contents of the Tn are kept unchanged. When the power supply for the CP is input from the TIn, the CP is cut during the instruction execution time. When the internal clock is used as the CP, it is synchronous with instructions, and therefore this phenomenon does not occur.

As stated above, when the TIn is input as the CP to read the Tn, a signal (which has a pulse width that does not give rise to incorrect counting even if the CP is cut) must be input. That is, the time during which count is suspended by a read instruction is one machine cycle, therefore the pulse that is input to the TIn must be wider than it.

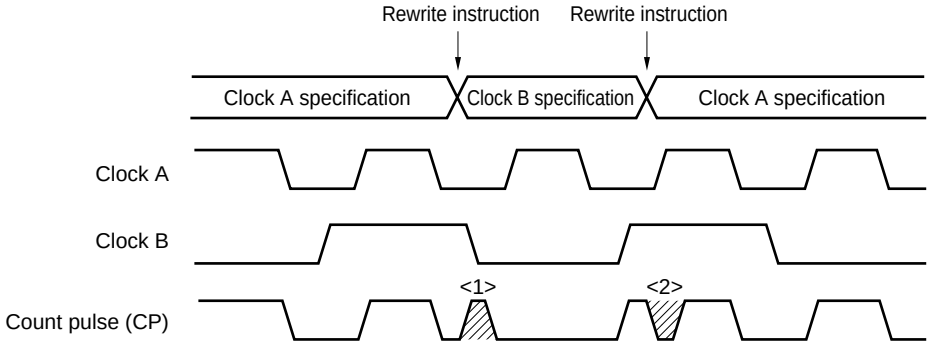


(4) Caution on changing the count pulse

When the count pulse (CP) is changed by rewriting the timer/event counter mode register (TMn), the specification for the change is valid immediately after the instruction is executed.

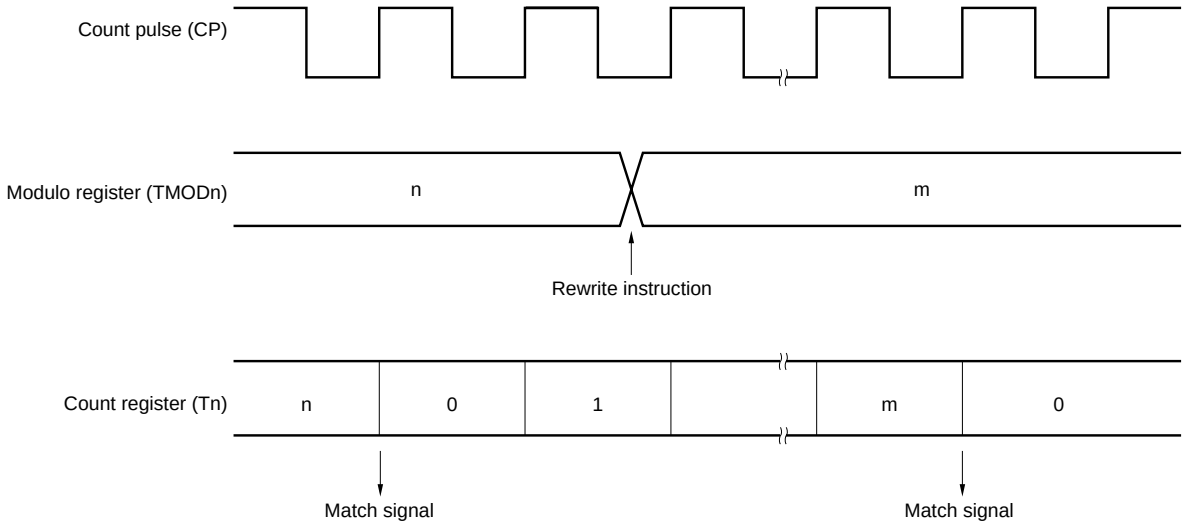


Depending on the combination of clock pulses at the time the CP is changed, whisker-like clock pulses (<1> or <2> in the illustration below) may be produced. In this case, incorrect counting may occur and the count register (Tn) may be disrupted. Therefore, when changing the CP, set bit 3 of the TMn to “1” and restart the timer simultaneously.

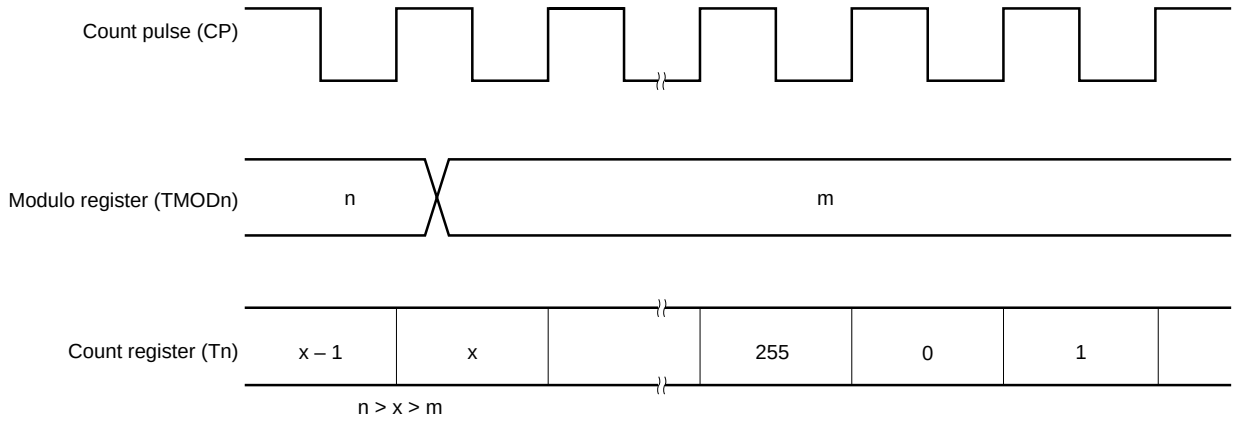


(5) Operation after changing the modulo register

The contents of the modulo register (TMODn) are rewritten by an 8-bit data memory manipulation instruction.



If the value of the TMODn after a change is smaller than the value of the count register (Tn), the Tn continues counting and overflows to restart counting from 0. Therefore, if the value (m) after the TMODn is changed is smaller than the value (n) before they are changed, the timer must restart after they are changed.



5.5 Serial interface

5.5.1. Serial interface function

The μPD754304 incorporates a clock-synchronous 8-bit serial interface consisting of the following three modes.

(1) Operation stop mode

This mode is used when serial transfer is not performed. The power dissipation can be reduced.

(2) 3-wire serial I/O mode

This mode performs data transfer in 8-bit units with three lines: serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

In addition, it enables high speed data transfer through simultaneous transmission and reception.

Because the top bit of 8-bit data for serial transfer is switchable to MSB or LSB, μPD754304 can be connected to any device regardless of whether its top bit is MSB or LSB.

Connection to 75XL series, 75X series, 78K series, and various types of peripheral I/O devices are possible.

(3) 2-wire serial I/O mode

This mode performs data transfer in 8 bit units with two lines: serial clock ($\overline{SCK}$), and serial data bus (SB0).

Communication to several devices by manipulating the output level to two lines with software is possible.

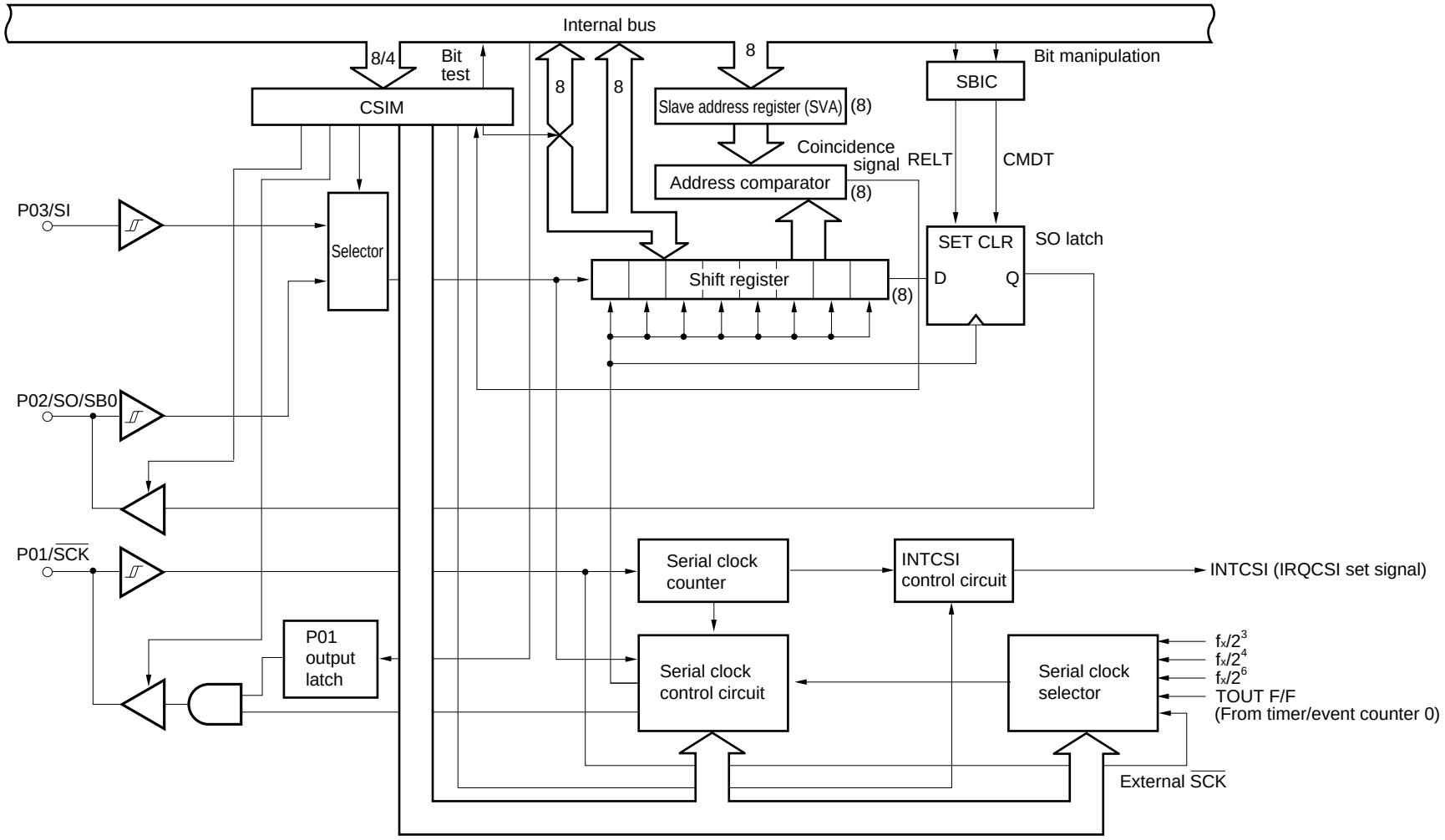
The levels of output to $\overline{SCK}$ and SB0 can be controlled by software so that they can accept any data transfer.

This eliminates the need for lines that are used for hand shaking when connecting two or more devices, thus enabling more efficient use of I/O port.

5.5.2 Configuration of serial interface

Figure 5-35 shows a block diagram of the serial interface.

Figure 5-35. Serial Interface Block Diagram ★



(1) Serial operation mode register (CSIM)

This 8-bit register specifies the operation mode and serial clock wake-up function of the serial interface. (For details, refer to (1) Serial operation mode register (CSIM) in **5.5.3.**)

(2) Serial bus interface control register (SBIC)

This 8-bit register consists of bits that control the status of the serial bus and flags that indicate the various statuses of the data input from the serial bus. It is mainly used in the SBI mode. (For details, refer to (2) Serial bus interface control register (SBIC) in **5.5.3.**)

(3) Shift register (SIO)

This register converts 8-bit serial data into parallel data or 8-bit parallel data into serial data. It performs transmission or reception (shift operation) in synchronization with the serial clock. Actual transmission or reception is controlled by writing data to the SIO. (For details, refer to (3) Shift register (SIO) in **5.5.3.**)

(4) SO latch

This latch holds the levels of the SO/SB0 and SI/SB1 pins. It can also be controlled directly via software. (For details, refer to paragraph (2) Serial bus interface control register (SBIC) in **5.5.3.**)

(5) Serial clock selector

This selects the serial clock to be used.

(6) Serial clock counter

This counter counts the number of serial clocks output or input when transmission or reception operation is performed, to check whether 8 bits of data have been transmitted or received.

★ **(7) Slave address register (SVA) and address comparator**

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

  When the μPD754304 is used as a slave or master, this register and comparator detects an error. (For details, refer to **(4) Slave address register (SVA)** in **5.5.3.**)

(8) INTCSI control circuit

This circuit controls generation of an interrupt request. The interrupt request (INTCSI) is generated in the following cases. When the interrupt request is generated, an interrupt request flag (IRQCSI) is set. (Refer to **Figure 6-1. Interrupt Control Circuit Block Diagram**.)

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

  An interrupt request is generated each time eight serial clocks have been counted.

(9) Serial clock control circuit

This circuit controls the supply of the serial clock to the shift register. It also controls the clock output to the $\overline{\text{SCK}}$ pin when the internal system clock is used.

(10) P01 output latch (SCKP)

This latch generates the serial clock via software after eight serial clocks have been generated.

It is set to “1” when the reset signal is input.

To select the internal system clock as the serial clock, set the P01 output latch to “1”.

5.5.3 Register function

(1) Serial operation mode register (CSIM)

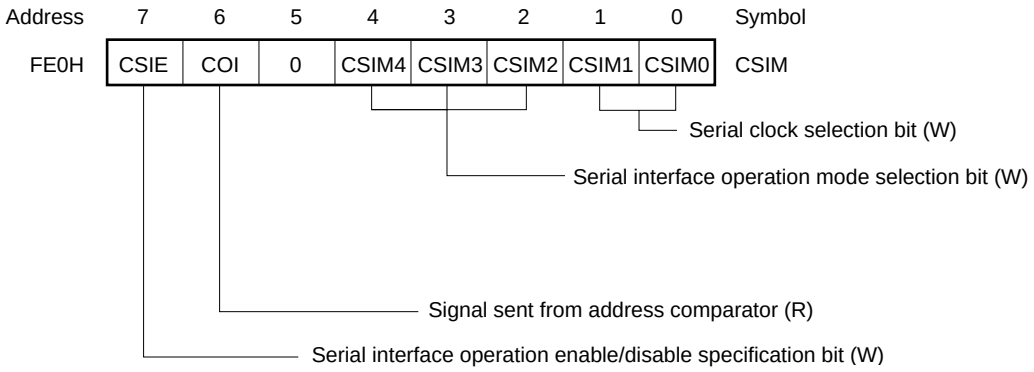
Figure 5-36 shows the format of the serial operation mode register (CSIM).

The CSIM is an 8-bit register which specifies the serial interface operation mode, serial clock, and wake-up function.

The register is manipulated by an 8-bit memory manipulation instruction. Bit 7 and 6 of the register can be operated bitwise. In this case, the name of each bit is used to manipulate.

All the bits are cleared to 0, when a $\overline{\text{RESET}}$ signal is generated.

Figure 5-36. Serial Operation Mode Register (CSIM) Format (1/2)



Caution Always set bit 5 of CSIM to 0.

Remarks 1. (R) Read only.

2. (W) Write only.

Figure 5-36. Serial Operation Mode Register (CSIM) Format (2/2)

Serial interface operation enable/disable specification bit (W)

		Shift register operation	Serial clock counter	IRQCSI flag	SO/SB0, SI pins
CSIE	0	Disable the shift operation.	Clear	Hold	Dedicated to port 0 function
	1	Enables the shift operation.	Count operation	Enable setting	Both for the function in each mode and port 0

Signal from address comparator (R) ★

COI Note	Condition to be cleared (COI = 0)	Condition to be set (COI = 1)
	The contents of the slave address register (SVA) are not the same as those of the shift register.	The contents of the slave address register (SVA) are the same as those of the shift register.

Note The reading of the COI is valid only before and after serial data transfer. Uncertain data is read out during transfer. The COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operating mode selection bit (W)

CSIM4	CSIM3	CSIM2	Operation mode	Shift register bit order	SO pin function	SI pin function
×	0	0	3-wire serial I/O mode	$SIO_{7-0} \leftrightarrow XA$ (transferred at MSB top)	SO/P02 (CMOS output)	SI/P03 (input)
		1		$SIO_{0-7} \leftrightarrow XA$ (transferred at LSB top)		
0	1	1	2-wire serial I/O mode	$SIO_{7-0} \leftrightarrow XA$ (transferred at MSB top)	SB0/P02 (N-channel open-drain I/O)	P03 input
Other than above			Setting prohibited.			

Remark × : don't care

Serial clock selection bit (W)

CSIM1	CSIM0	Serial clock		$\overline{SCK}$ pin mode
		3-wire serial I/O mode	2-wire serial I/O mode	
0	0	Clock input from outside to $\overline{SCK}$ pin		Input
0	1	Timer/event counter output (TOUT0)		Output
1	0	$f_X/2^4$ (during 375 kHz: 6.00 MHz operation, 262 kHz: 4.19 MHz operation)	$f_X/2^6$ (during 93.8 kHz: 6.00-MHz operation, 65.47 kHz: 4.19-MHz operation)	
1	1	$f_X/2^3$ (during 750 kHz: 6.00 MHz operation, 524 kHz: 4.19 MHz operation)		

Remarks 1. Each mode can be selected by the CSIE, CSIM3, and CSIM2 settings.

CSIE	CSIM3	CSIM2	Operating Mode
0	×	×	Operating stop mode
1	0	×	3-wire serial I/O mode
1	1	1	2-wire serial I/O mode

2. The P01/$\overline{\text{SCK}}$ pin is the following state based on the CSIE, CSIM1, and CSIM0 settings.

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

3. Follow the procedure below when clearing CSIE in a serial transfer.

<1> Clear the interrupt enable flag (IECSI) and enter the interrupt disable state.
<2> Clear CSIE.
<3> Clear the interrupt reguest flag (IRQCSI).

Examples 1. Select the fx/2^4 serial clock. The serial interrupt IRQSCI is generated at the end of each serial transfer. The mode for serial transfer with the MSB first in the 3-wire mode is selected.

```
SEL     MB15              ;or CLR1 MBE
MOV     XA, #10000010B
MOV     CSIM, XA          ;CSIM ← 10000010B
```

2. The state where serial transfer is in accordance with CSIM

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

(2) Serial bus interface control register (SBIC)

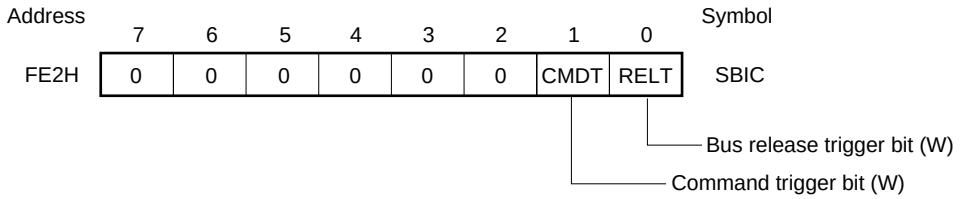
Figure 5-37 shows the format of the serial bus interface control register (SBIC).

The SBIC is an 8-bit register which is composed of the bits controlling the serial bus.

It is manipulated by a bit manipulation instruction. It can not be manipulated by an 8-bit/4-bit manipulation instruction.

All the bits are cleared to 0 when $\overline{\text{RESET}}$ signal is generated.

Figure 5-37. Serial Bus Interface Control Register (SBIC) Format



Remark (W) is write only.

Command trigger bit (W)

CMDT	Trigger output control bit for the command signal (CMD). When it is set (CMDT = 1), the SO latch is cleared to 0 and then CMDT bit is automatically cleared to 0.

Caution The CMDT bit must not be set during serial data transfer, but before or after it.

Bus release trigger bit (W)

RELT	Trigger output control bit for the bus release signal (REL). When it is set (RELT = 1), the SO latch is set to 1 and then RELT bit is automatically cleared to 0.

Caution The RELT bit must not be set during serial data transfer, but before or after it.

(3) Shift register (SIO)

Figure 5-38 shows the configuration of the system comprising the shift register and peripheral devices. The SIO is an 8-bit register which performs parallel-to-serial conversion and shift operation in synchronization with the serial clock.

Serial data transfer starts when data is entered into the SIO.

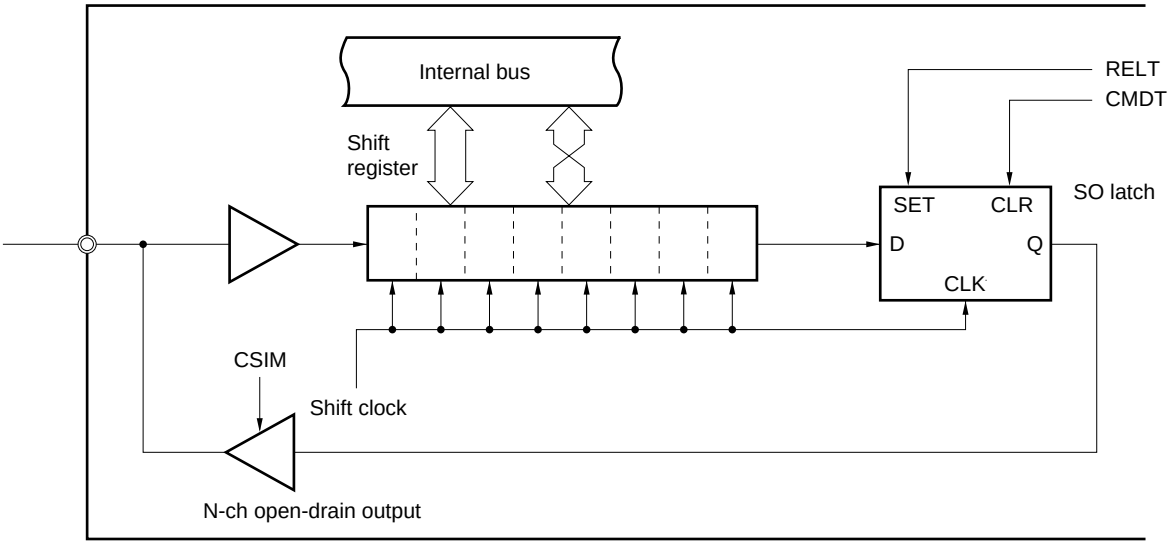
During data transmission, the data written in the SIO is output to the serial output (SO) or serial data bus (SB0 or SB1). During receiving, data is read from the serial input (SI) or SB0 (or SB1) to the SIO.

Data can be read and written by an 8-bit operation instruction.

When a $\overline{\text{RESET}}$ signal is generated during an operation, the contents of the SIO are uncertain. When the $\overline{\text{RESET}}$ signal is generated in the standby mode, the contents of the SIO are held.

The shift operation stops after data is transmitted or received in an 8-bit unit.

Figure 5-38. System Comprising Shift Register and Peripheral Devices Configuration



Data can be read and written (serial transfer start) form/to SIO in the following timings.

- The serial interface operation enable/disable bit (CSIE) is set to 1 except when the CSIE is set to 1 after data is written in the shift register.
- The serial clock is masked after the 8-bit serial data transfer ends.
- The $\overline{\text{SCK}}$ is high.

Be sure to write or read data to or from the SIO when $\overline{\text{SCK}}$ is high.

The input pin of the data bus is shared with the output pin in the two-wire serial I/O mode and SBI mode. The output pin is of the N-ch open-drain configuration. Therefore, put FFH in the SIO of the device that is to receive data.

(4) Slave address register (SVA) ★

SVA is an 8-bit register that sets a slave address (specification number).

It is operated by an 8-bit manipulation instruction.

The contents of the SVA becomes uncertain when a $\overline{\text{RESET}}$ signal is generated. However, when the $\overline{\text{RESET}}$ signal is generated in the standby mode, the contents of the SVA are held.

The function for having SVA is described next.

Detection of errors (in 2-wire serial I/O mode)

The SVA detects an error in the following cases:

- When the μPD754304 operates as the master and transmits addresses, commands, and data
- When the μPD754304 transmits data as a slave device

For details, refer to **5.5.6 (6) Error detection**.

5.5.4 Operation stop mode

The operation stop mode is used when serial transfer is not performed, to reduce the power dissipation.

In this mode, the shift register does not perform shift operations. Therefore, it can be used as an ordinary 8-bit register.

When the reset signal is input, operation stop mode is set. The P02/SO/SB0 and P03/SI pins are set to the input port mode. The P01/$\overline{\text{SCK}}$ pin can be used as an input port pin if so specified by the serial operation mode register.

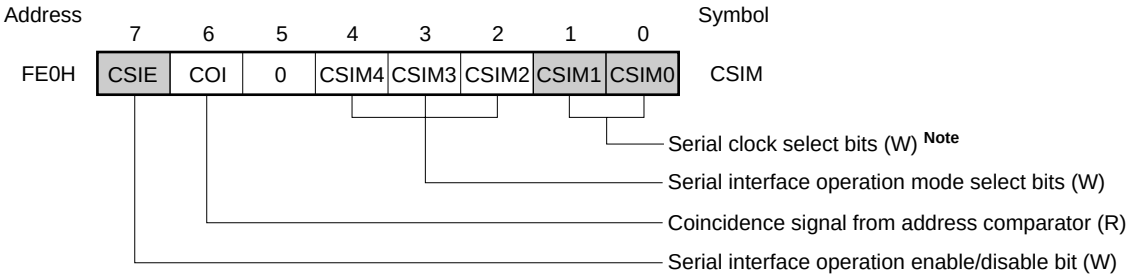
[Register setting]

Operation stop mode is set by using the serial operation mode register (CSIM). (For the format of the CSIM, refer to (1) Serial operation mode register (CSIM) in **5.5.3.**)

The CSIM is manipulated in 8-bit units. However, the CSIE bit of this register can be manipulated in 1-bit units. The name of the bit can be used for manipulation.

The CSIM is set to 00H at reset.

The shaded portions in the figure below indicate the bits used in operation stop mode.



Note This bit can select the status of the P01/$\overline{\text{SCK}}$ pin.

Remark (R) : read only
(W) : write only

Serial interface operation enable/disable bit (W)

		Operation of shift register	Serial clock counter	IRQCSI flag	SO/SB0 and SI pins
CSIE	0	Shift operation disabled	Cleared	Retained	Dedicated to port 0 function

Serial clock select bit (W)

The P01/$\overline{\text{SCK}}$ pin is set to the following status according to the setting of the CSIM0 and CSIM1 bits.

CSIM1	CSIM0	Status of P01/$\overline{\text{SCK}}$ pin
0	0	High impedance
0	1	High level
1	0	
1	1	

Clear the CSIE bit using the following procedure during serial transfer:

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

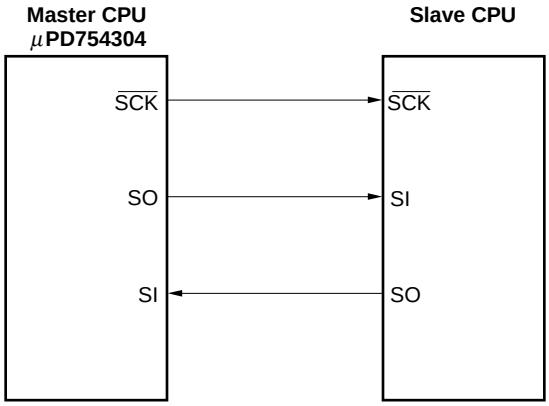
5.5.5 Operation in 3-wire serial I/O mode

In the 3-wire operation mode, the μPD754304 can be connected to microcomputers in the 75XL series, 75X series, μPD7500 series, and 87AD series.

In this mode, communication is established by using three lines: serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

Figure 5-39. Example of System Configuration in 3-Wire Serial I/O Mode

3-wire serial I/O ↔ 3-wire serial I/O



Remark The μPD754304 can be also used as a slave CPU.

(1) Register setting

When 3-wire serial I/O mode is used, the following two registers must be set:

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

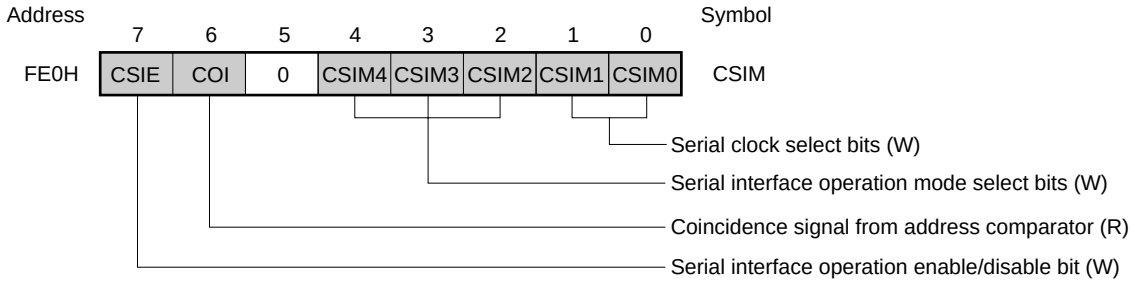
(a) Serial operation mode register (CSIM)

When 3-wire serial I/O mode is used, set CSIM as shown below. (For the format of CSIM, refer to (1) Serial operation mode register (CSIM) in 5.5.3.)

CSIM is manipulated by using 8-bit manipulation instructions. Bits 7 and 6 can also be manipulated in 1-bit units.

The contents of the CSIM are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in 3-wire serial I/O mode.



★

Remark (R) : read only
(W) : write only

Serial interface operation enable/disable bit (W)

		Operation of shift register	Serial clock counter	IRQCSI flag	SO/SB0 and SI pins
CSIE	1	Shift operation enabled	Count operation	Can be set	Function in each mode and port 0 function shared

Signal from address comparator (R) ★

COI **Note**	Clear condition (COI = 0)	Clear condition (COI = 1)
	When the slave address register (SVA) data and shift register data do not coincide	When slave address register (SVA) data and shift register data coincide

Note COI can be read before the start of serial transfer and after completion of the serial transfer. An undefined value is obtained if this bit is read during transfer. Data written to COI by an 8-bit manipulation instruction is ignored.

Serial interface operation mode select bit (W)

CSIM4	CSIM3	CSIM2	Bit order of shift register	SO pin function	SI pin function
×	0	0	$SIO_{7-0} \leftrightarrow XA$ (MSB first)	SO/P02 (CMOS output)	SI/P03 (input)
		1	$SIO_{0-7} \leftrightarrow XA$ (LSB first)		

Remark × : don't care

Serial clock select bit (W)

CSIM1	CSIM0	Serial clock	$\overline{SCK}$ pin mode
0	0	External clock input to $\overline{SCK}$ pin	Input
0	1	Timer/event counter output (TOUT0)	Output
1	0	$f_X/2^4$ (262 kHz) **Note**	
1	1	$f_X/2^3$ (524 kHz) **Note**	

Note The frequency in parentheses applies when f_X = 4.19 MHz

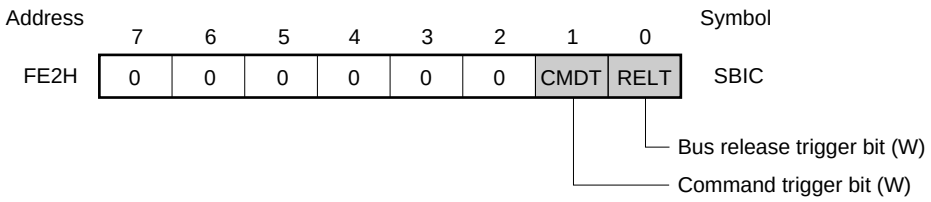
(b) Serial bus interface control register (SBIC)

When the three-wire serial I/O mode is used, set SBIC as shown below. (For the format of SBIC, refer to (2) Serial bus interface control register (SBIC) in **5.5.3.**)

This register is manipulated by using bit manipulation instructions.

The contents of SBIC are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the three-wire serial I/O mode.



Remark (W) : write only

Command trigger bit

CMDT	This bit controls the output trigger of a command signal (CMD). When this bit is set to 1, the SO latch is cleared to 0. After that, the CMDT bit is automatically cleared to 0.

Bus release trigger bit (W)

RELT	This bit controls the output trigger of a bus release signal (REL). When this bit is set to 1, the SO latch is set to 1. After that, the RELT bit is automatically cleared to 0.

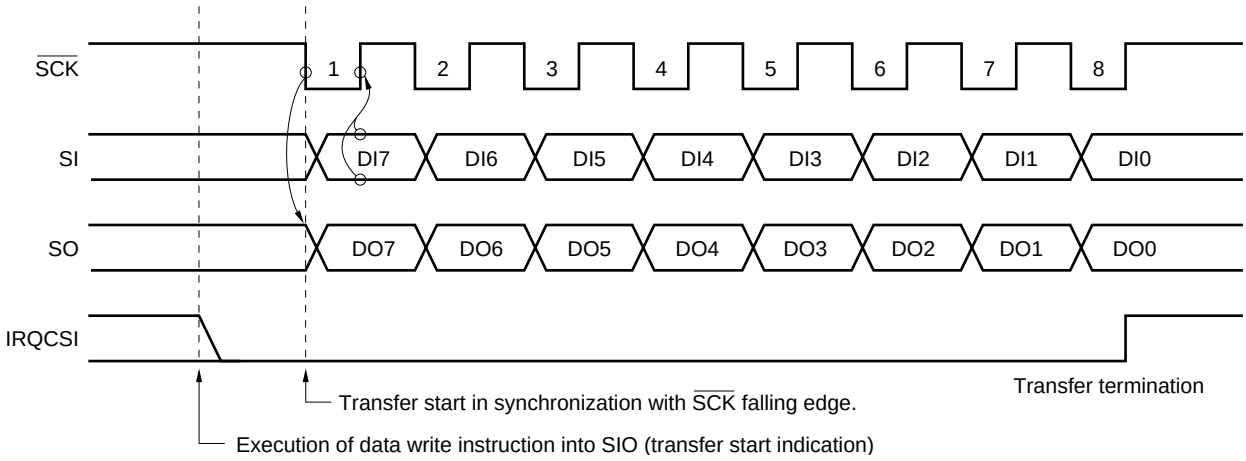
(2) Communication operation

The 3-wire serial I/O mode transmit/receive data in 8-bit units. Data is transferred one bit at a time in synchronization with a given serial clock.

Shift register shift operation is performed in synchronization with the serial clock ($\overline{SCK}$) falling edge. Transmit data is retained in the SO latch and output from the SO pin. On the $\overline{SCK}$ rising edge, receive data input to the SI pin is latched in the shift register.

When 8-bit transfer terminates, shift register operation automatically stops and the interrupt request flag (IRQCSI) is set.

Figure 5-40. 3-wire Serial I/O Mode Timing



Because the SO pin is a CMOS output pin and outputs the status of the SO latch, the output status of the SO pin can be manipulated by setting the RELT and CMDT bits.

However, do not perform this manipulation during serial transfer.

The output status of the $\overline{SCK}$ pin can be controlled by manipulating the P01 latch in the output mode (mode of the internal system clock). (Refer to **5.5.7 $\overline{SCK}$ pin output manipulation**.)

(3) Selecting the serial clock

The serial clock is selected by using bits 0 and 1 of the serial operation mode register (CSIM). The following four types of serial clocks can be selected:

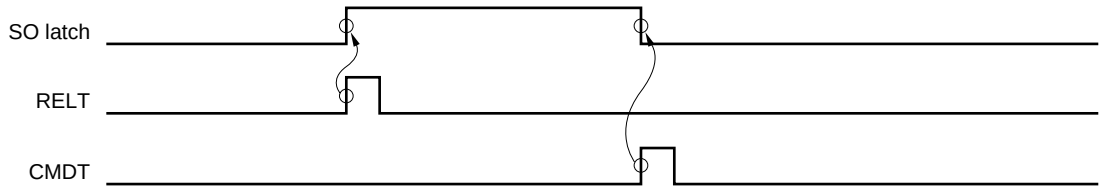
Table 5-9. Selection of Serial Clock and Applications (in 3-wire serial I/O mode)

Mode register		Serial clock		Timing at which shift register can be read/ written and serial transfer can be started	Application
CSIM 1	CSIM 0	Source	Masking serial clock		
0	0	External $\overline{SCK}$	Automatically masked at end of transfer of 8-bit data	<1> In operation stop mode (CSIE = 0) <2> If serial clock is masked after 8-bit serial transfer <3> When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Half duplex start-stop synchronization 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

Figure 5-41 illustrates the operation of RELT and CMDT.

Figure 5-41. Operation of RELT and CMDT



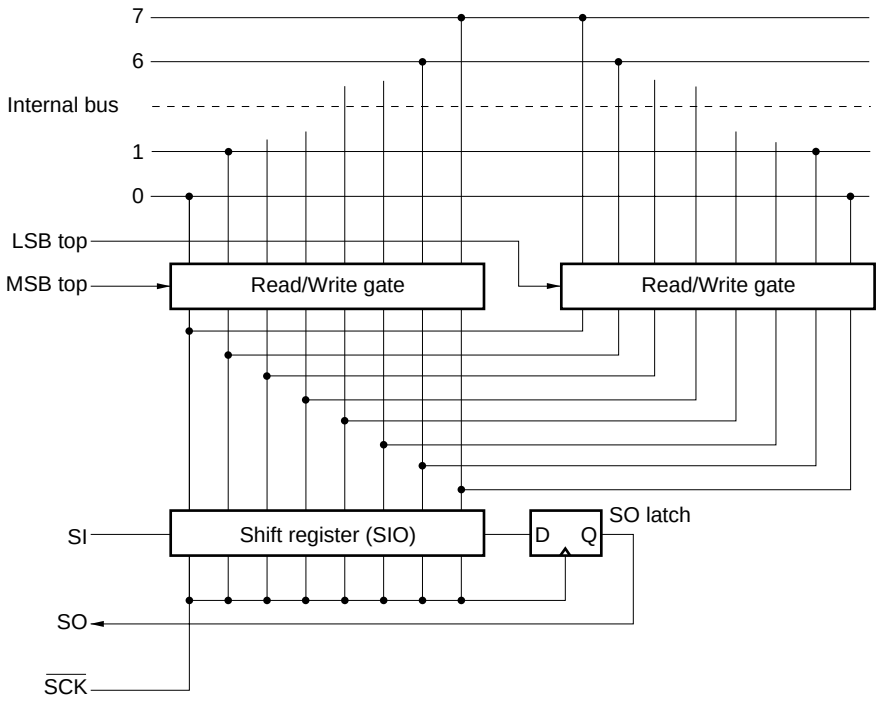
(5) Transfer top bit change between MSB and LSB

The 3-wire serial I/O mode enables selection of the most significant bit (MSB) or least significant bit (LSB) for the transfer top bit.

Figure 5-42 shows the shift register (SIO) and internal bus configuration. As shown in **Figure 5-42**, MSB and LSB can be reversed for read/write.

MSB or LSB can be specified as the transfer top bit by setting serial operation mode register (CSIM) bit 2.

Figure 5-42. Transfer Bit Change Circuit



The transfer top bit is switched by changing the bit order of data write into the shift register (SIO). The SIO shift order is always the same.

Change the transfer top bit between MSB and LSB before writing data into the shift register.

(6) Starting transfer

Serial transfer is started when the transfer data is placed in the shift register (SIO), if the following two conditions are satisfied:

- Serial interface operation enable/disable bit (CSIE) = 1
- The internal serial clock is stopped after 8-bit serial transfer or $\overline{SCK}$ is high

Caution Transfer is not started even if CSIE is set to "1" after the data has been written to the shift register.

When an 8-bit transfer has been completed, the serial transfer is automatically stopped, and an interrupt request flag (IRQCSI) is set.

Example To transfer the RAM data specified by the HL register to SIO and, at the same time, load the data in SIO to the accumulator and start serial transfer

```
MOV     XA, @HL         ; Fetches out transfer data from RAM
SEL     MB15            ; or CLR1 MBE
XCH     XA, SIO         ; Exchanges transmit data and receive data, and starts transfer
```

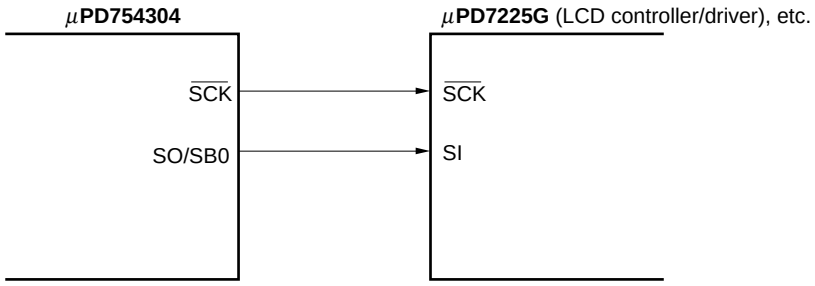
(7) Applications using 3-wire serial I/O mode

(a) To transfer data, MSB first, with 262-kHz transfer clock (at 4.19 MHz) (master operation)

<Program example>

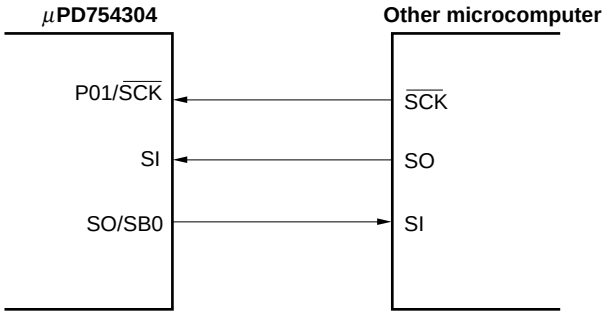
```
CLR1    MBE
MOV     XA, #10000010B
MOV     CSIM, XA        ; Sets transfer mode
MOV     XA, TDATA       ; TDATA is address storing transfer data
MOV     SIO, XA         ; Sets transfer data and starts transfer
```

Caution After transfer has been started for the first time, transfer can be started by writing data to SIO (by using MOV SIO, XA or XCH XA, SIO) the second and later times.



In this example, the SI pin of the μPD754304 can be used as an input pin.

(b) To transfer data, LSB first, with an external clock (slave operation)
(In this example, the shift register is read/written using a function that reverses the MSB-LSB order.)



<Program example>

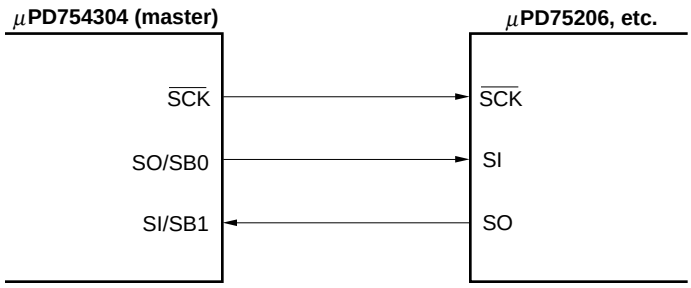
Main routine

```
CLR1   MBE
MOV    XA, #84H
MOV    CSIM, XA        ; Stops serial operation, MSB/LSB inverse mode, external clock
MOV    XA, TDATA
MOV    SIO, XA         ; Sets transfer data and starts transfer
EI     IECSI
EI
```

Interrupt routine (MBE = 0)

```
MOV    XA, TDATA
XCH    XA, SIO         ; Receive data ↔ transfer data, starts transfer
MOV    RDATA, XA       ; Saves receive data
RETI
```

(c) To transmit or receive data at high speeds using a 524-kHz (at 4.19 MHz) transfer clock



<Program example> ··· Master

```
        CLR1    MBE
        MOV     XA, #10000011B
        MOV     CSIM, XA        ;   Sets transfer mode
        MOV     XA, TDATA
        MOV     SIO, XA         ;   Sets transfer data and starts transfer
          :
LOOP:   SKTCLR  IRQCSI          ;   Test IRQCSI
        BR      LOOP
        MOV     XA, SIO         ;   Receives data
```

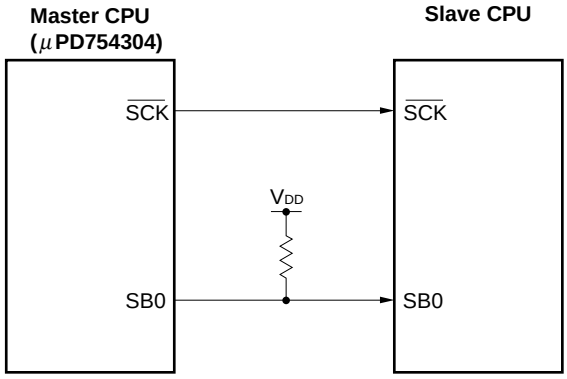
5.5.6 Operation in 2-wire serial I/O mode

The 2-wire serial I/O mode can correspord to an arbitrary communication format by programs.

Basically, communication is established by using two lines: serial clock ($\overline{SCK}$) and serial data input/output (SB0).

Figure 5-43. Example of System Configuration in 2-Wire Serial I/O Mode

2-wire serial I/O ↔ 2-wire serial I/O



Remark The μPD754304 can be also used as a slave CPU.

(1) Register setting

When the 2-wire serial I/O mode is used, the following two registers must be set:

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

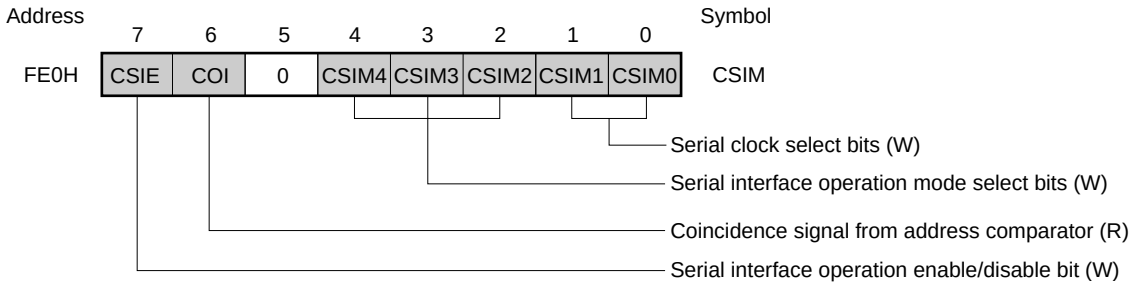
(a) Serial operation mode register (CSIM)

When the 2-wire serial I/O mode is used, set CSIM as shown below. (For the format of CSIM, refer to (1) Serial operation mode register (CSIM) in **5.5.3.**)

CSIM is manipulated by using an 8-bit manipulation instructions. Bits 7 and 6 can also be manipulated in 1-bit units.

The contents of the CSIM are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the 2-wire serial I/O mode.



★

Remark (R) : read only
(W): write only

Serial interface operation enable/disable bit (W)

		Operation of shift register	Serial clock counter	IRQCSI flag	SO/SB0 or SI pin
CSIE	1	Shift operation enabled	Count operation	Can be set	Function in each mode and port 0 function shared

Signal from address comparator (R) ★

COI **Note**	Clear condition (COI = 0)	Set condition (COI = 1)
	When slave address register (SVA) data and shift register data do not coincide	When slave address register (SVA) data and shift register data coincide

Note COI can be read before the start of a serial transfer or after completion of a serial transfer. An undefined value is read if this bit is read during transfer. Data written to COI by an 8-bit manipulation instruction is ignored.

Serial interface operation mode select bit (W)

CSIM4	CSIM3	CSIM2	Bit order of shift register	SO pin function	SI pin function
0	1	1	$SIO_{7-0} \leftrightarrow XA$ (MSB first)	SBO/P02 (N-ch open-drain I/O)	P03 input

Serial clock select bit (W)

CSIM1	CSIM0	Serial clock	$\overline{SCK}$ pin mode
0	0	External clock input to $\overline{SCK}$ pin	Input
0	1	Timer/event counter output (TOUT0)	Output
1	0	$f_X/2^6$ (65.5 kHz) **Note**	
1	1		

Note The frequency in parentheses applies when f_X = 4.19 MHz.

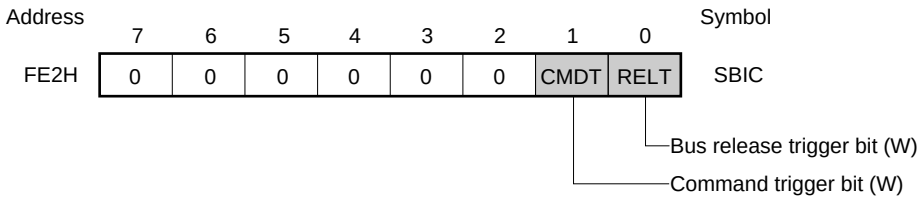
(b) Serial bus interface control register (SBIC)

When the 2-wire serial I/O mode is used, set SBIC as shown below. (For the format of SBIC, refer to (2) Serial bus interface control register (SBIC) in **5.5.3.**)

This register is manipulated by using bit manipulation instructions.

The contents of SBIC are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the 2-wire serial I/O mode.



Remark (W): write only

Command trigger bit

CMDT	This bit controls the output trigger of a command signal (CMD). When this bit is set to 1 (CMDT = 1), the SO latch is cleared to 0. After that, the CMDT bit is automatically cleared to 0.

Bus release trigger bit (W)

RELT	This bit controls the output trigger of a bus release signal (REL). When this bit is set to 1 (RELT = 1), the SO latch is set to 1. After that, the RELT bit is automatically cleared to 0.

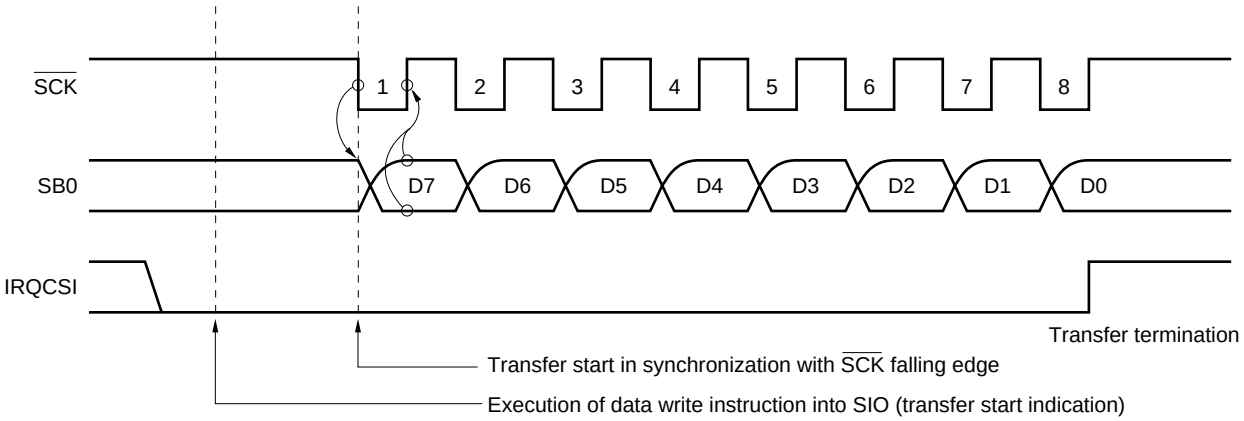
(2) Communication operation

The 2-wire serial I/O mode transmits/receives data in 8-bit units. Data is transferred one bit at a time in synchronization with a given serial clock.

Shift register shift operation is performed in synchronization with the serial clock ($\overline{SCK}$) falling edge. Transmit data is retained in the SO latch and output starting at the MSB from the SB0/P02 pin. On the $\overline{SCK}$ rising edge, receive data input from the SB0 pin is latched in the shift register.

When an 8-bit transfer terminates, shift register operation automatically stops and the interrupt request flag (IRQCSI) is set.

Figure 5-44. 2-wire Serial I/O Mode Timing



The SB0 pin specified for the serial data bus becomes N-ch open-drain input/output, thus must be pulled high with an external pull-up resistor. Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance.

Since the SB0 pin outputs the SO latch state, the SB0 pin output state can be manipulated by setting the RELT and CMDT bits.

However, do not perform this manipulation during serial transfer.

In the output mode (internal system clock mode), the $\overline{SCK}$ pin output state can be controlled if the P01 output latch is manipulated (See **5.5.7 $\overline{SCK}$ pin output manipulation**).

(3) Selecting the serial clock

The serial clock is selected by using the bits 0 and 1 of the serial operation mode register (CSIM). Four types of serial clocks can be selected:

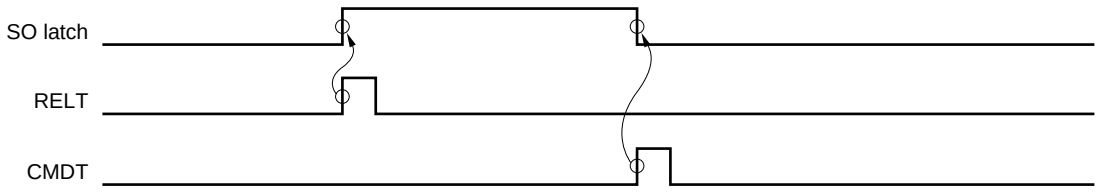
Table 5-10. Selection of Serial Clock and Applications (in 2-wire serial I/O mode)

Mode register		Serial clock		Timing at which shift register can be read/ written and serial transfer can be started	Application
CSIM 1	CSIM 0	Source	Masking serial clock		
0	0	External $\overline{SCK}$	Automatically masked at end of transfer of 8-bit data	<1> In operation stop mode (CSIE = 0) <2> If serial clock is masked after 8-bit serial transfer <3> When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Serial transfer at any speed
1	0	$f_X/2^6$			Low-speed serial transfer
1	1				

(4) Signals

Figure 5-45 illustrates the operation of RELT and CMDT.

Figure 5-45. Operation of RELT and CMDT



(5) Starting transfer

Serial transfer is started when the transfer data is placed in the shift register (SIO), if the following two conditions are satisfied:

- Serial interface operation enable/disable bit (CSIE) = 1
- The internal serial clock is stopped after 8-bit serial transfer, or $\overline{SCK}$ is high

Cautions 1. Transfer is not started even if CSIE is set to "1" after the data has been written to the shift register.

2. Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance.

When an 8-bit transfer has been completed, serial transfer is automatically stopped, and an interrupt request flag (IRQCSI) is set.

(6) Error detection

In 2-wire serial I/O mode, because the status of the serial bus SB0 or SB1 during transmission is also loaded to the shift register SIO of the device transmitting data, an error can be detected by the following methods:

(a) By comparing SIO data before and after transmission

If the two data differ from each other, it can be assumed that a transmission error has occurred.

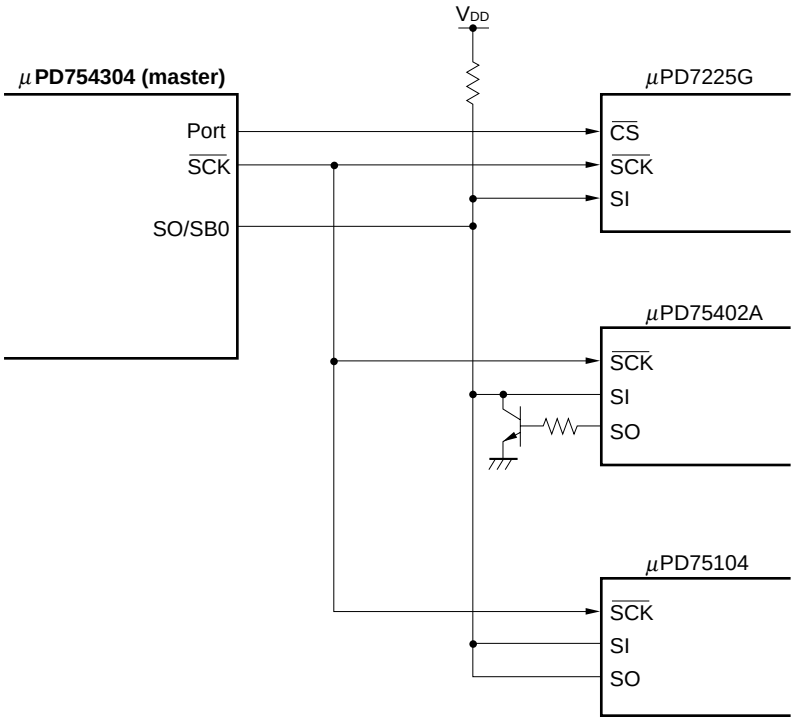
(b) By using the slave address register (SVA) ★

The transmit data is placed in SIO and SVA and transmission is executed. After transmission, the COI bit (coincidence signal from the address comparator) of the serial operation mode register (CSIM) is tested. If this bit is "1", the transmission has been completed normally. If it is "0", it can be assumed that a transmission error has occurred.

(7) Application using 2-wire serial I/O mode

2-wire serial I/O mode can be used to connect multiple devices by configuring a serial bus.

Example To configure a system by connecting the μPD754304 as the master and μPD75104, μPD75402A, and μPD7225G as slaves



The SI and SO pins of the μPD75104 are connected together. When serial data is not output, the serial operation mode register is manipulated so that the output buffer is turned off to release the bus.

Because the SO pin of the μPD75402A cannot go into a high-impedance state, a transistor is connected to the SO pin as shown in the figure, so that the SO pin can be used as an open-collector output pin. When data is input to the μPD75402A, the transistor is turned off by writing 00H to the shift register in advance.

The timing of when each microcomputer outputs data is determined in advance.

The serial clock is output by the μPD754304, which is the master. All the slave microcomputers operate on an external clock.

5.5.7 $\overline{\text{SCK}}$ pin output manipulation

The $\overline{\text{SCK}}$/P01 pin, which incorporates an output latch, can also produce static output by software control in addition to a normal serial clock.

The number of $\overline{\text{SCK}}$s can be set as desired by using software to control the P01 output latch (the SO/SB0 pins are controlled by setting the SBIC RELT and CMDT bits).

The $\overline{\text{SCK}}$/P01 pin output control method is described below:

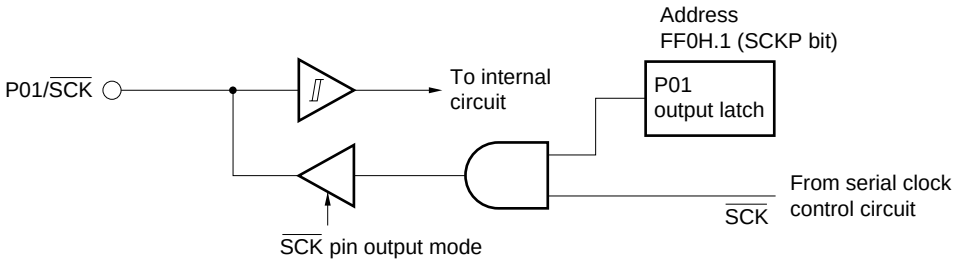
<1> The serial operation mode register (CSIM) is set ($\overline{\text{SCK}}$ pin: Output mode). $\overline{\text{SCK}}$ from serial clock control circuit is set to 1 during serial transfer stops.

<2> The P01 output latch is controlled by using bit manipulation (operation) instructions.

Example To output one clock pulse to $\overline{\text{SCK}}$/P01 pin by using software.

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #10000011B  ; SCK (fx/2^3), output mode
MOV     CSIM, XA
CLR1    0FF0H.1         ; SCK/P01 ← 0
SET1    0FF0H.1         ; SCK/P01 ← 1
```

Figure 5-46. $\overline{\text{SCK}}$/P01 Pin Configuration



The P01 output latch is mapped in bit 1 of address FF0H. When the $\overline{\text{RESET}}$ signal is generated, the P01 output latch is set to 1.

Cautions 1. The P01 output latch must be set to 1 during normal serial transfer.

2. Do not use "PORT0.1" to specify the P01 output latch address. Write directly the address (0FF0H.1) in operand. At that time, set MBS to 0, or set MBE to 1 and MBS to 15.

Do not use

```
CLR1    PORT0.1
SET1    PORT0.1
```

Use

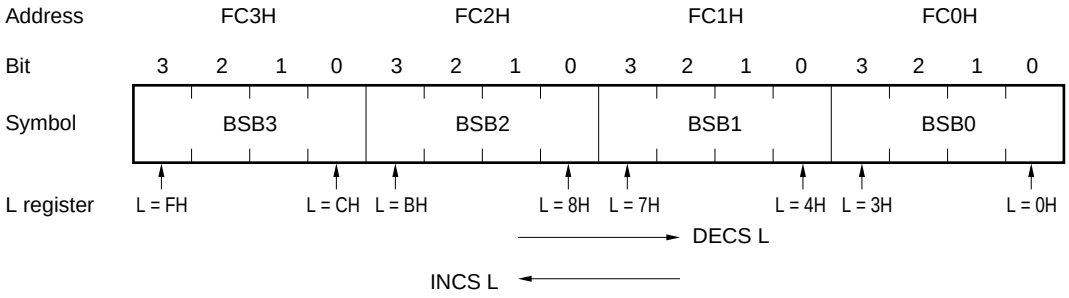
```
CLR1    0FF0H.1
SET1    0FF0H.1
CLR1    SCKP
SET1    SCKP
```

5.6 Bit Sequential Buffer ··· 16 Bits

The bit sequential buffer (BSB) is a special data memory for bit manipulation and the bit manipulation can be easily performed by changing the address specification and bit specification in sequence, therefore it is useful when processing a long data bit-wise.

The data memory is composed of 16 bits and the pmem.@L addressing of a bit manipulation instruction is possible. The bit can be specified indirectly by the L register. In this case, processing can be done by moving the specified bit in sequence by incrementing and decrementing the L register in the program loop.

Figure 5-47. Bit Sequential Buffer Format



Remarks 1. In the pmem.@L addressing, the specified bit moves corresponding to the L register.

2. In the pmem.@L addressing, the BSB can be manipulated regardless of MBE/MSB specification.

Data can be operated by direct addressing. It can be used for continuous input and output of 1-bit data together with the 1-bit/4-bit/8-bit direct addressing and pmem.@L addressing. For 8-bit manipulation, the BSB0 and BSB2 must be specified to manipulate the high-order 8-bits and low-order 8 bits.

Example The 16-bit data of BUFF 1/2 is output serially starting with the bit 0 of port 3.

```
        CLR1    MBE
        MOV     XA, BUFF1
        MOV     BSB0, XA        ; Sets BSB0,1.
        MOV     XA, BUFF2
        MOV     BSB2, XA        ; Sets BSB2,3.
        MOV     L, #0
LOOP0 : SKT     BSB0, @L        ; Tests the specification bit of BSB.
        BR      LOOP1
        NOP                     ; Dummy (timing adjustment)
        SET1    PORT3.0         ; Sets the bit 0 of port 3.
        BR      LOOP2
LOOP1 : CLR1    PORT3.0         ; Clears the bit 0 of port 3.
        NOP                     ; Dummy (timing adjustment)
        NOP
LOOP2 : INCS    L               ; L ← L+1
        BR      LOOP0
        RET
```

CHAPTER 6 INTERRUPT FUNCTION AND TEST FUNCTION

The μPD754304 has seven vector interrupt sources and one test input that can be used for various applications.

The interrupt control circuit of the μPD754304 has unique features and can process interrupts at extremely high speed.

(1) Interrupt function

(a) Vectored interrupt function for hardware control, enabling/disabling the interrupt acceptance by the interrupt enable flag (IE×××) and interrupt master enable flag (IME).

(b) Can set any interrupt start address.

(c) Multiple interrupts wherein the order of priority can be specified by the interrupt priority select register (IPS).

(d) Test function of interrupt request flag (IRQ×××). An interrupt generated can be checked by software.

(e) Release the standby mode. A release interrupt can be selected by the interrupt enable flag.

(2) Test function

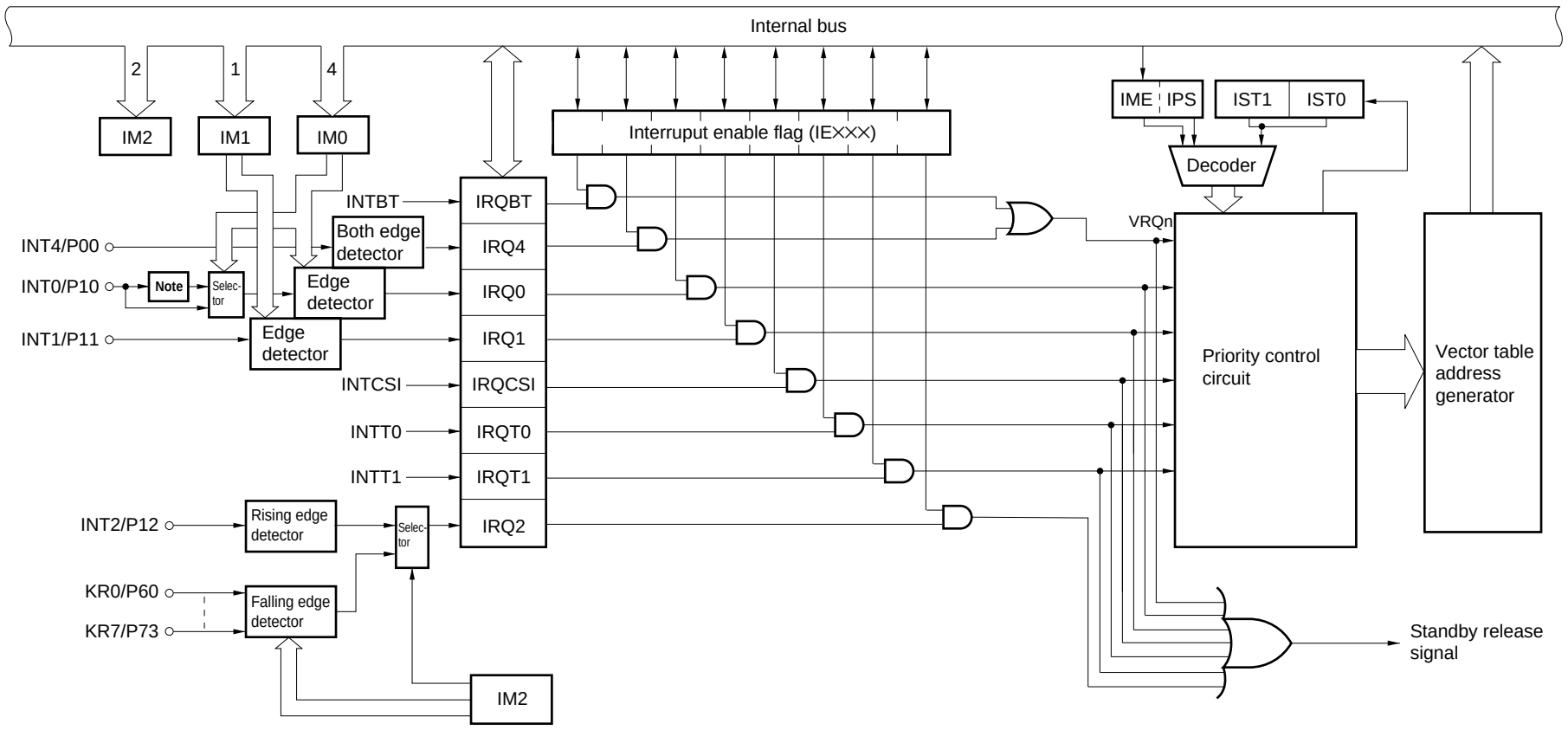
(a) Test request flag (IRQ×××) generation can be checked by software.

(b) Release the standby mode. The test source to be released can be selected by the test enable flag.

6.1 Configuration of Interrupt Control Circuit

The interrupt control circuit is configured as shown in **Figure 6-1**, and each hardware unit is mapped to the data memory space.

Figure 6-1. Interrupt Control Circuit Block Diagram



Note Noise eliminator (Standby release is disable when noise eliminator is selected.)

6.2 Types of Interrupt Sources and Vector Tables

The μPD754304 has the following seven types of interrupt sources, and multiple interrupts by software control are allowed.

Table 6-1. Types of Interrupt Sources

Interrupt source		Internal/ external	Interrupt priority Note	Vectored interrupt request signal (vector table address)
INTBT	(Reference interval signal sent from the basic interval timer/watchdog timer)	Internal	1	VRQ1 (0002H)
INT4	(Detection by both rising edge and falling edge is valid.)	External		
INT0	(Selects rising edge or falling edge.)	External	2	VRQ2 (0004H)
INT1		External	3	VRQ3 (0006H)
INTCSI	(Serial data transfer end signal)	Internal	4	VRQ4 (0008H)
INTT0	(Match signal between the count register and modulo register of the timer/event counter 0)	Internal	5	VRQ5 (000AH)
INTT1	(Match signal between the count register and modulo register of the timer/event counter 1)	Internal	6	VRQ6 (000CH)

Note The priority of interrupts is applied when several interrupt requests are generated simultaneously.

Figure 6-2. Interrupt Vector Table

Address			
0000H	MBE	RBE	Internal reset start address (high-order 6 bits)
			Internal reset start address (low-order 8 bits)
0002H	MBE	RBE	INTBT/INT4 start address (high-order 6 bits)
			INTBT/INT4 start address (low-order 8 bits)
0004H	MBE	RBE	INT0 start address (high-order 6 bits)
			INT0 start address (low-order 8 bits)
0006H	MBE	RBE	INT1 start address (high-order 6 bits)
			INT1 start address (low-order 8 bits)
0008H	MBE	RBE	INTCSI start address (high-order 6 bits)
			INTCSI start address (low-order 8 bits)
000AH	MBE	RBE	INTT0 start address (high-order 6 bits)
			INTT0 start address (low-order 8 bits)
000CH	MBE	RBE	INTT1 start address (high-order 6 bits)
			INTT1 start address (low-order 8 bits)

The priority column in Table 6-1 indicates the priority according to which interrupts are executed if two or more interrupts occur at the same time, or if two or more interrupt requests are kept pending.

To the vector table, write the start address of interrupt processing, and the set values of MBE and RBE during interrupt processing. The vector table is set by using an assembler pseudoinstruction (VENTn).

Example Setting of vector table of INTBT/INT4

```
VENT1     MBE=0,  RBE=0,  GOTOBT
  ↑         ↑       ↑       ↑
 <1>       <2>     <3>     <4>
```

<1> Vector table of address 0002
<2> Setting of MBE in interrupt processing routine
<3> Setting of RBE in interrupt processing routine
<4> Symbol indicating start address of interrupt processing routine

Caution The vector address set by the VENTn (n = 1-7) instruction is 2n.

Example Setting of vector tables of INTBT/INT4 and INTT0

```
VENT1     MBE=0, RBE=0, GOTOBT
VENT5     MBE=0, RBE=1, GOTOT0
```

6.3 Hardware Controlling Interrupt Function

(1) Interrupt request flag and interrupt enable flag

The μPD754304 has the following eight interrupt request flags (IRQ×××) corresponding to the respective interrupt sources:

INT0 interrupt request flag (IRQ0)	Serial interface interrupt request flag (IRQCSI)
INT1 interrupt request flag (IRQ1)	
INT2 interrupt request flag (IRQ2)	Timer/event counter 0 interrupt request flag (IRQT0)
INT4 interrupt request flag (IRQ4)	Timer/event counter 1 interrupt request flag (IRQT1)
BT interrupt request flag (IRQBT)	

Each interrupt request flag is set to "1" when the corresponding interrupt request is generated, and is automatically cleared to "0" when the interrupt processing is executed. However, because IRQBT and IRQ4 share the vector address, these flags are cleared differently from the other flags (refer to **6.6 "Vector Address Share Interrupt Service"**).

The μPD754304 also has eight interrupt enable flags (IE×××) corresponding to the respective interrupt request flags.

INT0 interrupt enable flag (IE0)	Serial interface interrupt enable flag (IECSI)
INT1 interrupt enable flag (IE1)	Timer/event counter 0 interrupt enable flag (IET0)
INT2 interrupt enable flag (IE2)	Timer/event counter 1 interrupt enable flag (IET1)
INT4 interrupt enable flag (IE4)	BT interrupt enable flag (IEBT)

When the interrupt enable flag is "1", the interrupt is enabled; and when it is "0", the interrupt is disabled.

When the interrupt request flag is set and interrupt enable flag enables the interrupt, a vectored interrupt request (VRQn) is generated. It is also used to release the standby mode.

The interrupt request flag and interrupt enable flag are operated by a bit manipulation instruction and 4-bit memory manipulation instruction. When the bit instruction is used, they can be directly manipulated at any time regardless of MBE setting. The interrupt enable flag is manipulated by an EI IE××× instruction and DI IE××× instruction. A SKTCLR instruction is normally used to test the interrupt request flag.

```
Example  EI      IE0     ; Enables INT0.
         DI      IE1     ; Disables INT1.
         SKTCLR  IRQCSI  ; Skips and clears when IRQCSI is 1.
```

When the interrupt request flag is set by an instruction, a vectored interrupt is executed as if an interrupt were generated.

The interrupt request flag and interrupt enable flag are cleared to “0” when a $\overline{\text{RESET}}$ signal is generated and all the interrupts are inhibited.

Table 6-2. Set Signals for Interrupt Request Flags

Interrupt request flag	Set signal for interrupt request flag	Interrupt enable flag
IRQBT	Set by the reference interval signal by the basic interval timer/watchdog timer.	IEBT
IRQ4	Set by the detection of both rising edge and falling edge of an INT4/P00 pin.	IE4
IRQ0	Reset by the edge detection of an INT0/P10 pin input signal. The detection edge is selected by the INT0 edge detection mode register (IM0).	IE0
IRQ1	Reset by the edge detection of an INT1/P11 pin input signal. The detection edge is selected by the INT1 edge detection mode register (IM1).	IE1
IRQCSI	Set by a serial data transfer end signal of the serial interface.	IECSI
IRQT0	Set by a match signal sent from the timer/event counter 0.	IET0
IRQT1	Set by a match signal sent from the timer/event counter 1.	IET1

(2) Interrupt priority selection register (IPS)

The interrupt priority selection register selects the higher-order-priority interrupts in a system wherein multiple interrupts are allowed. Its low-order 3 bits are used for specification.

Bit 3 is the interrupt master enable flag (IME) which specifies whether all the interrupts are prohibited or not.

The IPS is set by a 4-bit memory manipulation instruction. Bit 3 is set and reset by an EI/DI instruction.

To change the contents of the lower 3 bits of IPS, the interrupt must be disabled (IME = 0).

```
Example  DI              ; Disables interrupt
         CLR1    MBE
         MOV     A, #1011B
         MOV     IPS, A  ; Gives higher priority to INT1 and enables interrupt
```

All the bits are cleared to “0” when a $\overline{\text{RESET}}$ signal is generated.

Figure 6-3. Interrupt Priority Selection Register

Address	3	2	1	0	Symbol
FB2H	IPS3	IPS2	IPS1	IPS0	IPS

Selection of higher-order-priority interrupts

0	0	0	No interrupts are handled as higher-order-priority interrupts.	
0	0	1	VRQ1 (INTBT/INT4)	The above vectored interrupts are regarded as higher-order priority interrupts.
0	1	0	VRQ2 (INT0)	
0	1	1	VRQ3 (INT1)	
1	0	0	VRQ4 (INTCSI)	
1	0	1	VRQ5 (INTT0)	
1	1	0	VRQ6 (INTT1)	
1	1	1	Setting prohibited	

Interrupt master enable flag (IME)

0	Disables all the interrupts and no vectored interrupt is started.
1	Controls interrupt enable/disable by the corresponding interrupt enable flag.

(3) Hardware of INT0, INT1, and INT4

(a) Figure 6-4(a) shows the configuration of the INT0. An external interrupt is input to select the detection edge, that is, rising edge or falling edge.

The INT0 has a noise eliminator by means of the sampling clock. (See **Figure 6-5. Noise Detection Circuit Input/Output Timing**.) The noise eliminator removes the pulses which are narrower than the two cycles of the sampling clock **Note**. However, a pulse which is wider than the one cycle of the sampling clock may be accepted in some cases as an interrupting signal depending on a sampling timing. The pulses which are wider than the two cycles of sampling clock are accepted all the time as interrupting signals. (See **Figure 6-5 <2>(a)**.)

The INT0 has the two sampling clocks: F and fX/64, either of which can be selected. One of them is selected by the bit 3 (IM03) of the INT0 edge detection mode register (IM0). (See **Figure 6-6(a)**.)

The detection edge is selected by the bits 0/1 (IM00/IM01) of the INT0 edge detection mode register (IM0).

Figure 6-6(a) shows the format of the IM0. It is set by a 4-bit manipulation instruction. All the bits are cleared to "0" by a $\overline{\text{RESET}}$ signal generated and the rising edge specification is adopted.

Note $2t_{CY}$ when the sampling clock is Φ.
$128/f_X$ when the sampling clock is $f_X/64$.

Cautions 1. Pulses are input to the INT0/P10 pin via a noise eliminator if it is used as a port, therefore pulses longer than the two cycles of sampling clock must be input.

2. When the noise eliminator is selected, that is IM02 is set to 0, the INT0 performs sampling by a clock, therefore it does not operate in the standby mode. Consequently, if the standby mode is to be released by the INT0, the noise eliminator must not be selected, that is the IM02 must be set to 1.

(b) Figure 6-4(b) shows the configuration of the INT1. An external interrupt is input to select the detection edge, rising edge or falling edge.

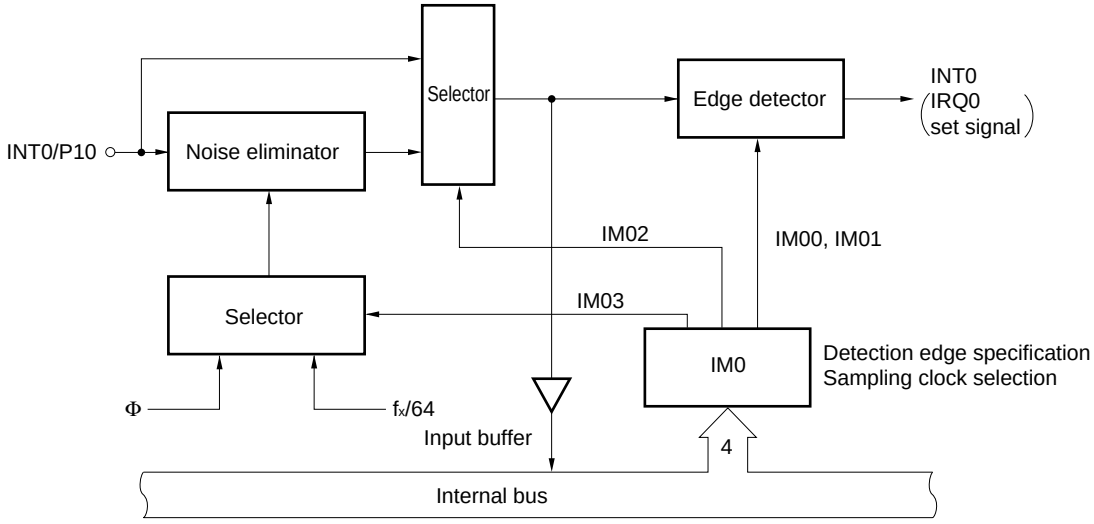
The detection edge is selected by the INT1 edge detection mode register (IM1).

Figure 6-6(b) shows the format of the IM1. It is set by a bit manipulation instruction. All the bits are cleared to "0" when the $\overline{\text{RESET}}$ signal is generated and the rising edge specification is adopted.

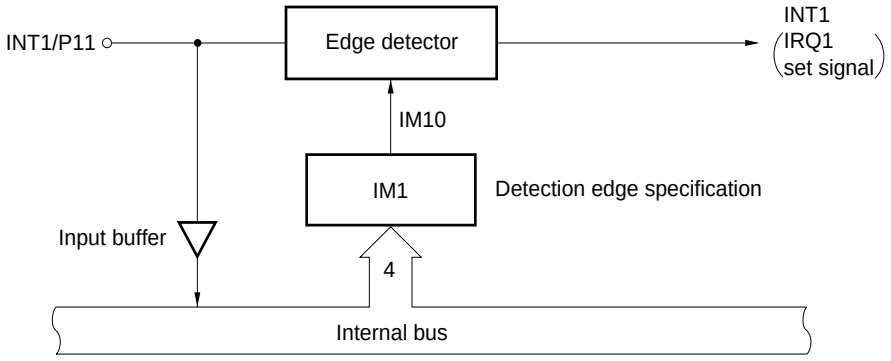
(c) Figure 6-4(c) shows the configuration of the INT4. An external interrupt is input so that both the rising edge and falling edge can be detected.

Figure 6-4. Configurations of INT0, INT1, and INT4

(a) INT0 hardware



(b) INT1 hardware



(c) INT4 hardware

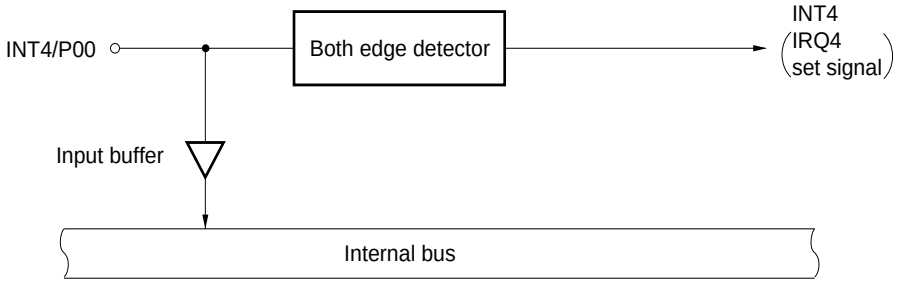
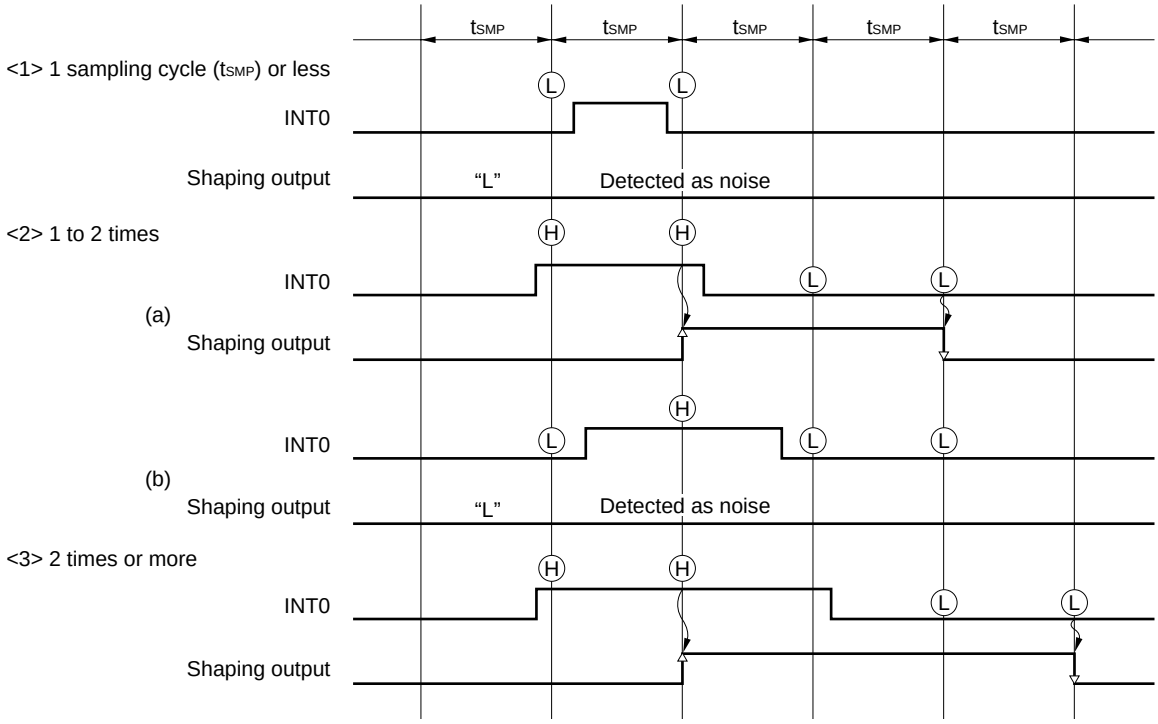


Figure 6-5. Noise Detection Circuit Input/Output Timing



Remark $t_{SMP} = t_{CY}$ or $64/f_X$

Figure 6-6. Edge Detection Mode Register Format

(a) INT0 edge detection mode register (IM0)

Address	3	2	1	0	Symbol
FB4H	IM03	IM02	IM01	IM00	IM0

IM01	IM00	Detection edge specification
0	0	Rising edge specification
0	1	Falling edge specification
1	0	Rising edge/falling edge specification
1	1	Ignored (The interrupt request flag is not set).

IM02	Noise eliminator select bit	Sampling	Standby release
0	Selects the noise eliminator.	Enabled	Impossible
1	Does not select the noise eliminator.	Disabled	Possible

IM03	Sampling clock
0	Φ (0.67 μs, 1.33 μs, 2.67 μs, 10.7 μs: 6.00 MHz operation)
1	f_X/64 (10.7 μs : 6.00 MHz operation)

(b) INT1 edge detection mode register (IM1)

Address					Symbol
	3	2	1	0	
FB5H	0	0	0	IM10	IM1

IM10	Detection edge specification
0	Rising edge specification
1	Falling edge specification

Caution When the edge detection mode register is changed, the interrupt request flag may be set in some case, therefore the interrupts must be prohibited beforehand to select the mode register and the interrupt request flag must be cleared by a CLR1 instruction, and then the interrupts must be enabled. When f_X/64 is selected as the sampling clock by changing the IM0, the interrupt request flag must be cleared when 16 machine cycles have elapsed since the mode register was changed.

(4) Interrupt status flag

The interrupt status flags (IST0 and IST1) indicate the status of the processing currently executed by the CPU and are included in PSW.

The interrupt priority control circuit controls multiple interrupts according to the contents of these flags as shown in **Table 6-3**.

IST0 and IST1 can be changed by using a 4-bit or bit manipulation instruction, and multiple interrupts can be performed with the status under execution changed. IST0 and IST1 can be manipulated in 1-bit units regardless of the setting of MBE.

Before manipulating IST0 and IST1, be sure to execute the DI instruction to disable the interrupt. Execute the EI instruction after manipulating the flags to enable the interrupt.

IST1 and IST0 are saved to the stack memory along with the other flags of PSW when an interrupt is acknowledged, and their statuses are automatically changed higher by one. When the RETI instruction is executed, the original values of IST1 and IST0 are restored.

The contents of these flags are cleared to “0” when the $\overline{\text{RESET}}$ signal is asserted.

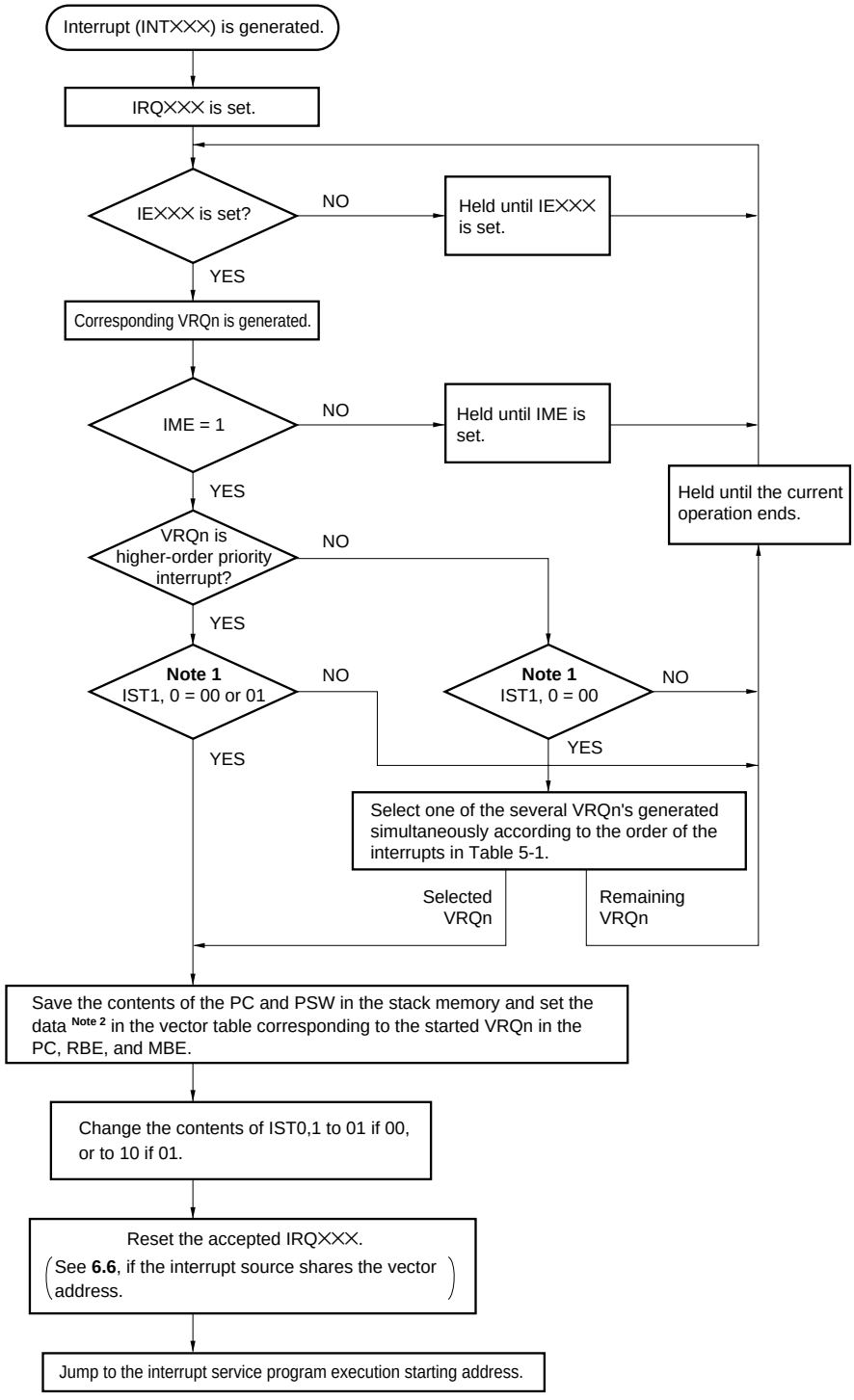
Table 6-3. IST1 and IST0 and Interrupt Processing Status

IST1	IST0	Status of processing under execution	Processing by CPU	Interrupt request that can be acknowledged	After interrupt acknowledged	
					IST1	IST0
0	0	Status 0	Executes normal program	All interrupts can be acknowledged	0	1
0	1	Status 1	Processes interrupt with low or high priority	Interrupt with high priority can be acknowledged	1	0
1	0	Status 2	Processes interrupt with high priority	Acknowledging all interrupts is disabled	–	–
1	1	Setting prohibited				

6.4 Interrupt Sequence

When an interrupt is generated, it is executed in the following procedure.

Figure 6-7. Interrupt Processing Sequence



Notes 1. IST1, 0 : interrupt status flag (bits 2, 3 of PSW; see **Table 6-3**)

2. Each vector table stores the interrupt service program starting address and values set for the MBE and RBE at the time the interrupt service starts.

6.5 Multiple Interrupt Service Control

The μPD754304 accepts multiple interrupts in the following two methods.

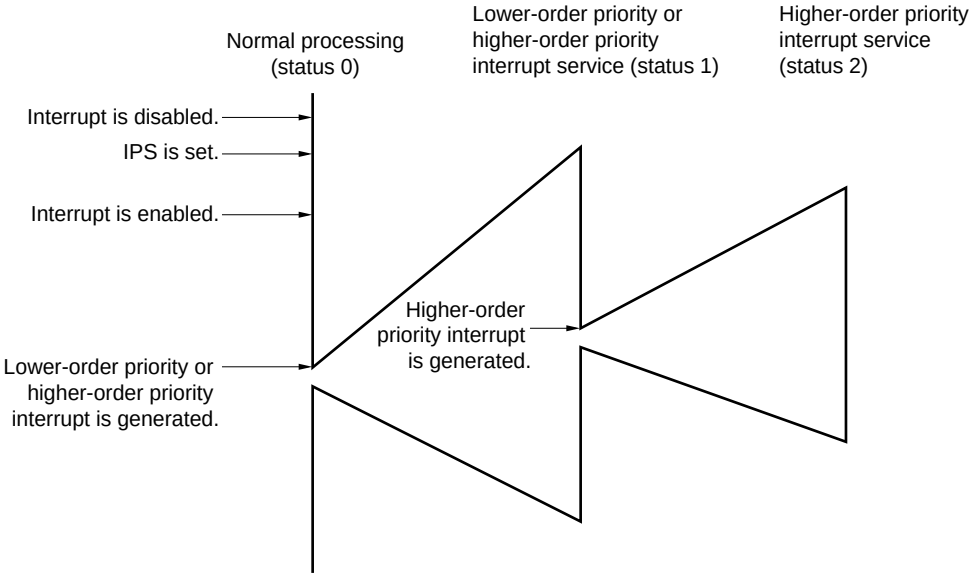
(1) Multiple interrupts wherein higher-order priority interrupts are specified

This is a standard multiple interrupts method of the μPD754304, wherein one of the interrupt sources is selected to enable the multiple interrupts (double interrupts) of the interrupt.

That is, the higher-order priority interrupts specified by the interrupt priority select register (IPS) can be accepted when the status of the current operation is 0 and 1, and the other lower-order priority interrupts can be serviced when the status is 0 (See **Figure 6-8** and **Table 6-3**).

Therefore, if this method is used when you wish to nest only one interrupt, operations such as enabling and disabling interrupts while the interrupt is processed need not to be performed, and the nesting level can be kept to 2.

Figure 6-8. Multiple Interrupts by Higher-Order Priority Interrupts



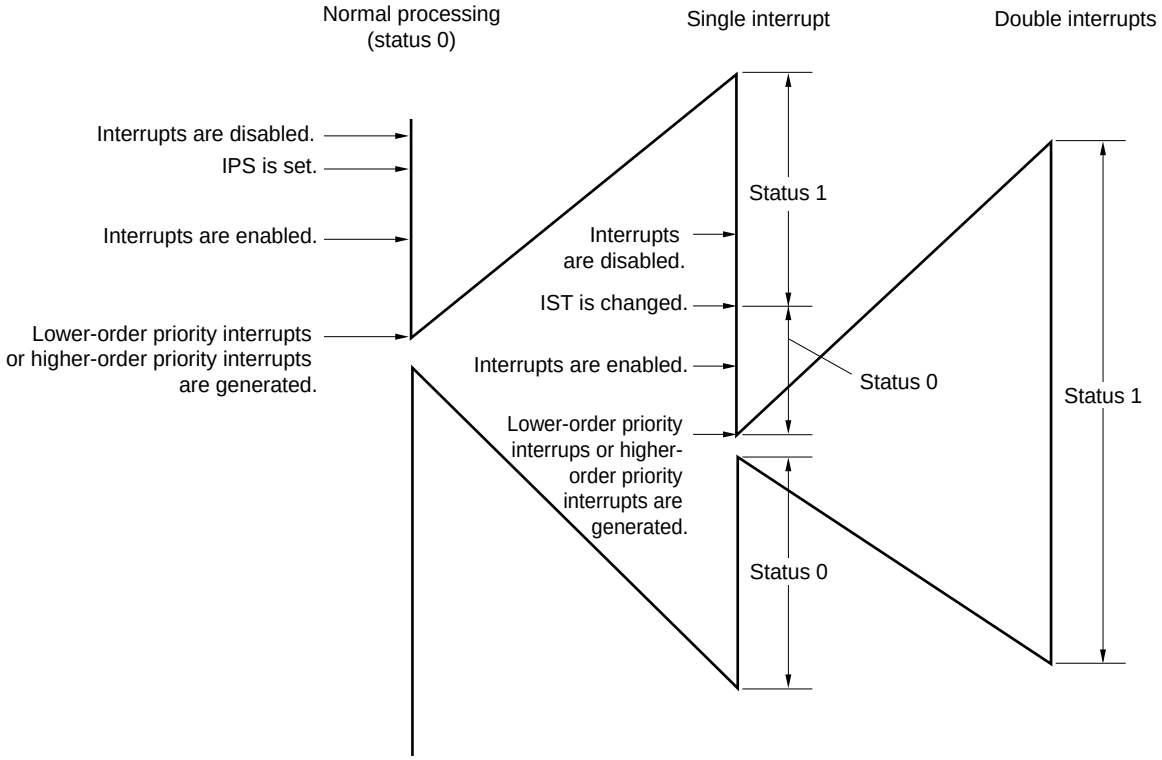
(2) Multiple interrupts changing the interrupt status flag

If the interrupt status flag is changed by the program, multiple interrupts can be accepted. That is, if IST1 and IST0 are changed to “0, 0” (status 0), multiple interrupts can be serviced.

This method is used when multiple interrupts (two or more interrupts) are to be accepted or more than double interrupts are to be performed.

The IST1 and IST0 must be changed beforehand in a status in which the interrupts are prohibited by a DI instruction.

Figure 6-9. Multiple Interrupts by Changing the Interrupt Status Flag



6.6 Vector Address Share Interrupt Service

Because INTBT and INT4 interrupt sources share the vector table addresses, interrupt source selection is made as described below:

(1) To use one interrupt only

Set the interrupt enable flag to 1 for the required one of the two interrupt sources sharing the vector table addresses, and clear the interrupt enable flag of the other interrupt source. In this case, an interrupt request is generated from the interrupt source corresponding to the interrupt enable flag that is set to 1 (IE×××= 1). When the interrupt request is acknowledged, the interrupt request flag is cleared.

(2) To use both interrupts

Set both the interrupt enable flags of the two interrupt sources to 1. In this case, an interrupt request is made by ORing the interrupt request flags of the two interrupt sources.

Even if an interrupt request is acknowledged when either or both of the interrupt request flags are set to 1, the interrupt request flags are not reset.

Therefore, the interrupt service routine must decide which interrupt source the interrupt is generated from. This is accomplished by executing the DI instruction at the SKTCLR instruction to check the interrupt request flags. If both the request flags are set when this request flag is tested or cleared, the interrupt request remains even if one of the request flags is cleared. If this interrupt is selected as having the higher priority, nesting processing is started by the remaining interrupt request.

Consequently, the interrupt request not tested is processed first. If the selected interrupt has the lower priority, the remaining interrupt is kept pending and therefore, the interrupt request tested is processed first. Therefore, an interrupt sharing a vector address with another interrupt is identified differently, depending whether it has the higher priority, as shown in **Table 6-4**.

Table 6-4. Identifying Interrupt Sharing Vector Table Address

With higher priority	Interrupt is disabled and interrupt request flag of interrupt that takes precedence is tested
With lower priority	Interrupt request flag of interrupt that takes precedence is tested

Examples 1. To use both INTBT and INT4 as having the higher priority and give priority to INT4

```
        DI
        SKTCLR   IRQ4      ;  IRQ4 = 1 ?
        BR       VSUBBT
          :                    Processing routine
        EI                     of INT4
        RETI
          :
VSUBBT: CLR1     IRQBT
          :
          :                    Processing routine
          :                    of INTBT
        EI
        RETI
```

2. To use both INTBT and INT4 as having the lower priority and give priority to INT4

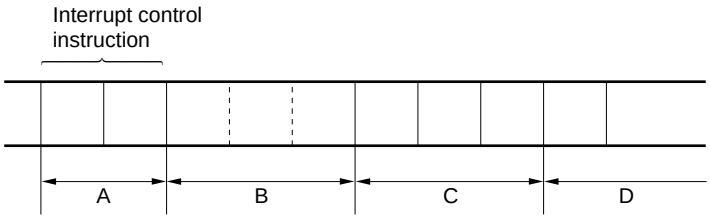
```
        SKTCLR   IRQ4      ;  IRQ4 = 1 ?
        BR       VSUBBT
          :
          :                    Processing routine
          :                    of INT4
        RETI
          :
VSUBBT: CLR1     IRQBT
          :
          :                    Processing routine
          :                    of INTBT
        RETI
```

6.7 Machine Cycles until Interrupt Processing

The number of machine cycles required since an interrupt request flag (IRQn) has been set until the interrupt routine is executed is as follows:

(1) If IRQn is set while interrupt control instruction is executed

If IRQn is set while an interrupt control instruction is executed, the next one instruction is executed. Then three machine cycles of interrupt processing is performed and the interrupt routine is executed.



A : Sets IRQn

B : Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)

C : Interrupt processing (3 machine cycles)

D : Executes interrupt routine

Cautions 1. If two or more interrupt control instructions are described in a row, these control instructions will be all executed consecutively. After having executed the last one, interrupt processing of three machine cycles will be performed, followed by the interrupt routine.

2. If the DI instruction is executed when or after IRQn is set (A in the above figure), the interrupt request corresponding to IRQn that has been set is kept pending until the EI instruction is executed next time.

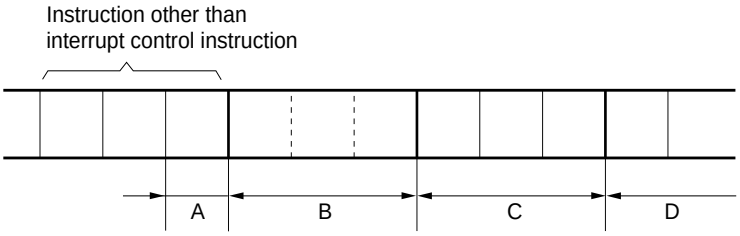
Remarks 1. An interrupt control instruction manipulates the hardware units related to interrupt (address FB×H of the data memory). The DI and EI instructions are interrupt control instructions.

2. The 3 machine cycles are the time, such as the stock manipulation, which is operated at the acknowledgment of interrupts.

(2) If IRQn is set while instruction other than (1) is executed

(a) If IRQn is set at the last machine cycle of the instruction under execution

In this case, the one instruction following the instruction under execution is executed, three machine cycles of interrupt processing are performed, and finally the interrupt routine is executed.

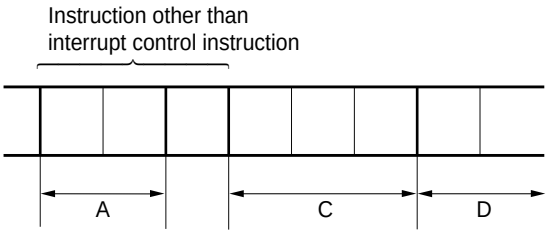


A : Sets IRQn
B : Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)
C : Interrupt processing (3 machine cycles)
D : Executes interrupt routine

Caution If the next instruction is an interrupt control instruction, the one instruction following the interrupt control instruction executed last is executed, three machine cycles of interrupt processing are performed, and finally the interrupt routine is executed. If the DI instruction is executed after IRQn has been set, the interrupt request corresponding to the set IRQn is kept pending.

(b) If IRQn is set before the last machine cycle of the instruction under execution

In this case, three machine cycles of processing are performed after execution of the current instruction, and then the interrupt routine is executed.



A : Sets IRQn
C : Interrupt processing (3 machine cycles)
D : Executes interrupt routine

6.8 Effective Usage of Interrupt

Use the interrupt function effectively as follows:

(1) Set MBE to 0 in interrupt processing routine.

If the memory used in the interrupt routine is allocated to addresses 00H through 7FH, and MBE is cleared to 0 by the interrupt vector table, you can program without having to consider the memory bank.

If it is necessary to use memory bank 1, save the memory bank select register by using the PUSH BS instruction and then select memory bank 1.

(2) Use different register banks for the normal routine and interrupt routine.

The normal routine uses register banks 2 and 3 with RBE = 1 and RBS = 2. If the interrupt routine is for one nested interrupt, use register bank 0 with RBE = 0, so that you do not have to save or restore the registers. When two or more interrupts are nested, set RBE to 1, save the register bank by using the PUSH BS instruction, and set RBS to 1 to select register bank 1.

(3) Use the software interrupt for debugging.

Even if an interrupt request flag is set by an instruction, the same operation as when an interrupt occurs is performed. For debugging of an irregular interrupt or debugging when two or more interrupts occur at the same time, the efficiency can be enhanced by setting the interrupt flag by an instruction.

6.9 Application of Interrupt

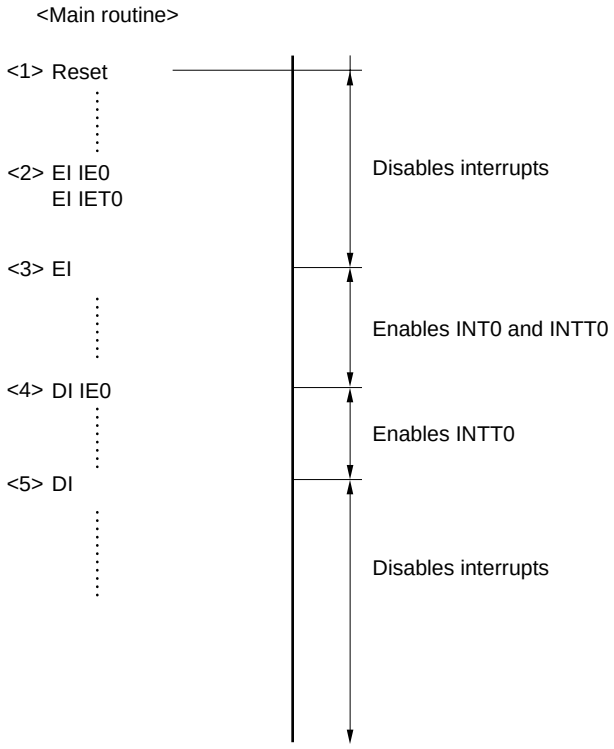
To use the interrupt function, first set as follows by the main routine:

(a) Set the interrupt enable flag of the interrupt used (by using the EI IE×××× instruction).
(b) To use INT0 or INT1, select the active edge (set IM0 or IM1).
(c) To use nesting (of an interrupt with the higher priority), set IPS (IME can be set at the same time).
(d) Set the interrupt master enable flag (by using the EI instruction).

In the interrupt routine, MBE and RBE are set by the vector table. However, when the interrupt specified as having the higher priority is processed, the register bank must be saved and set.

To return from the interrupt routine, use the RETI instruction.

(1) Enabling or disabling interrupt



<1> All the interrupts are disabled by the $\overline{\text{RESET}}$ signal.

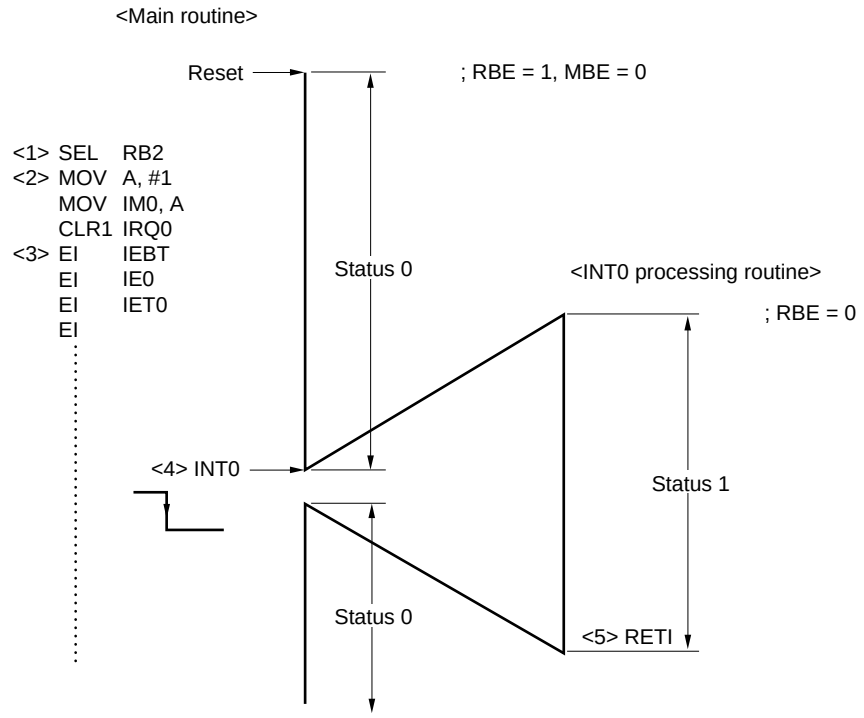
<2> An interrupt enable flag is set by the EI IE××× instruction.
At this stage, the interrupts are still disabled.

<3> The interrupt master enable flag is set by the EI instruction.
INT0 and INTT0 are enabled at this time.

<4> The interrupt enable flag is cleared by the DI IE××× instruction, and INT0 is disabled.

<5> All the interrupts are disabled by the DI instruction.

(2) Example of using INTBT, INT0 (falling edge active), and INTT0. No multiple interrupt (all interrupts have lower priority)



<1> All the interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
RBE = 1 is specified by the reset vector table. The SEL SB2 instruction uses register banks 2 and 3.

<2> INT0 is specified to be active at the falling edge.

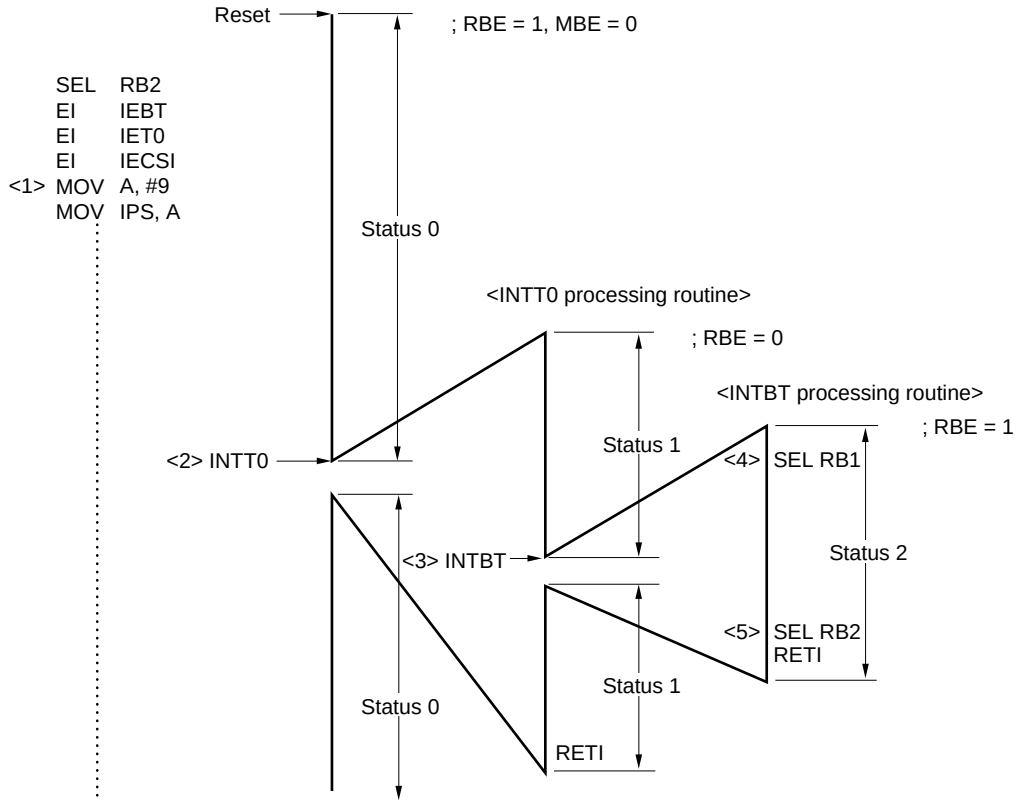
<3> The interrupt is enabled by the EI, EI IEXXX instruction.

<4> The INT0 interrupt routine is started at the falling edge of INT0. The status is changed to 1, and all the interrupts are disabled.
RBE = 0, and register banks 0 and 1 are used.

<5> Execution returns from the interrupt routine when the RETI instruction is executed. The status is returned to 0 and the interrupt is enabled.

Remark If all the interrupts are used as having the lower priority as shown in this example, saving or restoring the register bank is not necessary if RBE = 1 and RBS = 2 for the main routine and register banks 2 and 3 are used, and RBE = 0 for the interrupt routine and register banks 0 and 1 are used.

(3) Multiple interrupts with higher priority
(INTBT has higher priority and INTT0 and INTCSI have lower priority)



<1> INTBT is specified as having the higher priority by setting of IPS, and the interrupt is enabled at the same time.

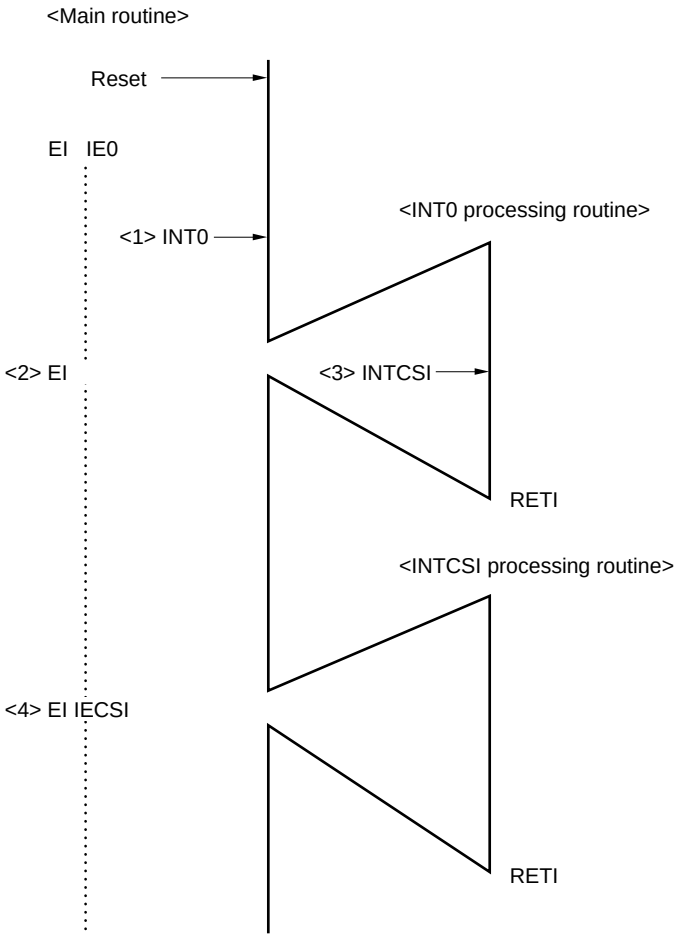
<2> INTT0 processing routine is started when INTT0 with the lower priority occurs. Status 1 is set and the other interrupts with the lower priority are disabled. RBE = 0 to select register bank 0.

<3> INTBT with the higher priority occurs. Double interrupts are executed. The status is changed to 2 and all the interrupts are disabled.

<4> RBE = 1 and RBS = 1 to select register bank 1 (only the registers used may be saved by the PUSH instruction).

<5> RBS is returned to 2, and execution returns to the main routine. The status is returned to 1.

(4) Executing pending interrupt – interrupt input while interrupts are disabled –



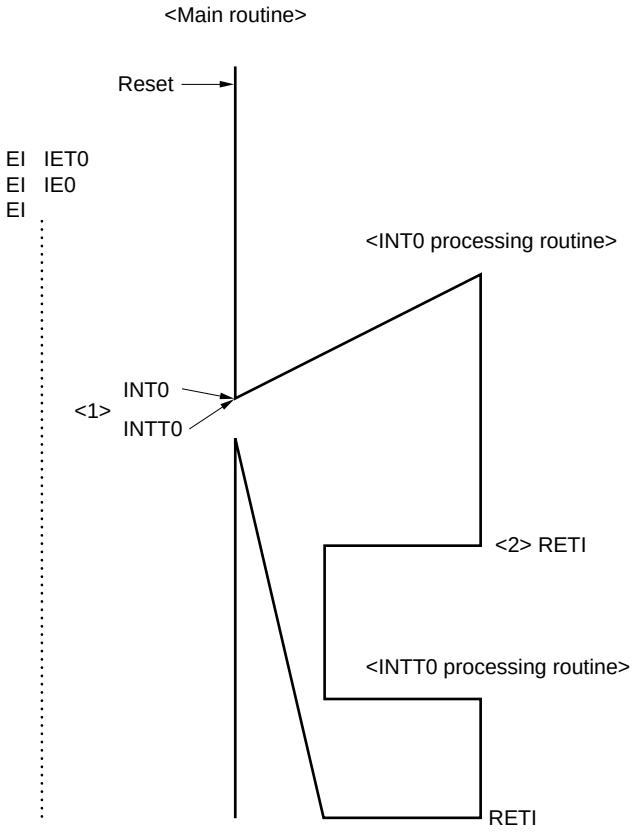
<1> The request flag is kept pending even if INT0 is set while the interrupts are disabled.

<2> INT0 processing routine is started when the interrupts are enabled by the EI instruction.

<3> Same as <1>.

<4> INTCSI processing routine is started when the pending INTCSI is enabled.

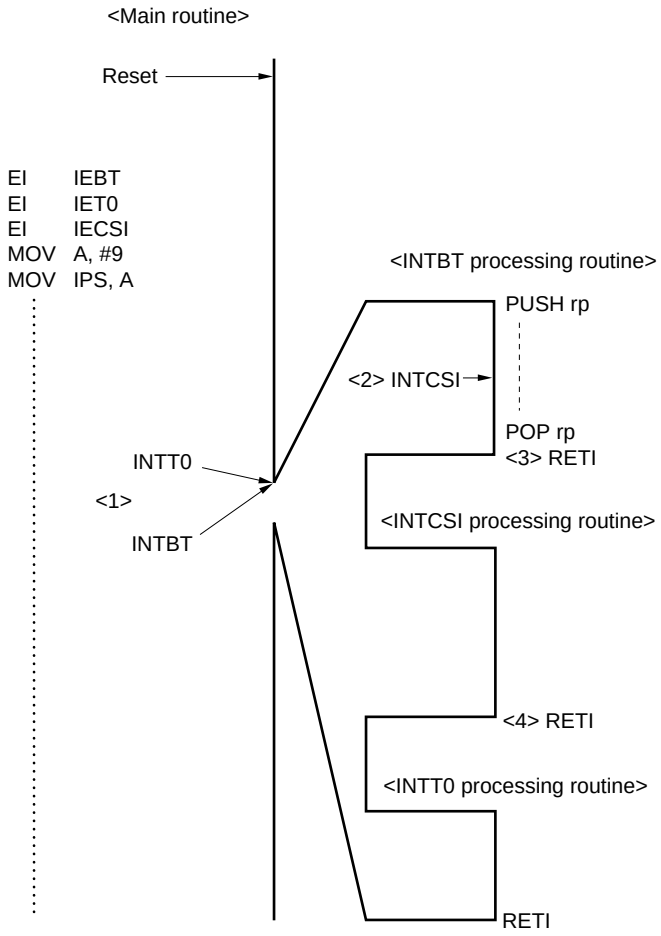
(5) Executing pending interrupt – two interrupts with lower priority occur simultaneously –



<1> If INT0 and INTT0 with the lower priority occur at the same time (while the same instruction is executed), INT0 with the higher priority is executed first (INTT0 is kept pending).

<2> When the INT0 processing routine is terminated by the RETI instruction, the pending INTT0 processing routine is started.

(6) Executing pending interrupt – interrupt occurs during interrupt processing (INTBT has higher priority and INTT0 and INTCSI have lower priority) –



<1> If INTBT with the higher priority and INTT0 with the lower priority occur at the same time, the processing of the interrupt with the higher priority is started (if there is no possibility that an interrupt with the higher priority occurs while another interrupt with the higher priority is processed, DI IE×× is not necessary).

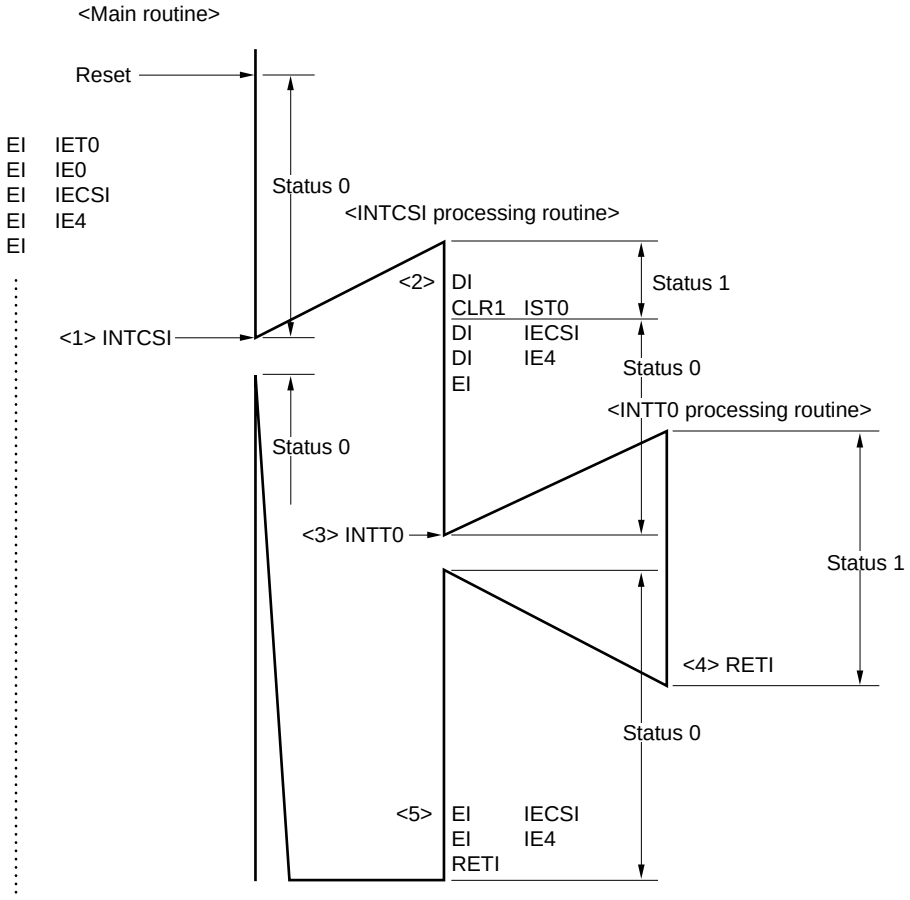
<2> If an interrupt with the lower priority occurs while the interrupt with the higher priority is executed, the interrupt with the lower priority is kept pending.

<3> When the interrupt with the higher priority has been processed, INTCSI with the higher priority of the pending interrupts is executed.

<4> When the processing of INTCSI has been completed, the pending INTT0 is processed.

(7) Enabling two double interrupts

– INTT0 and INT0 permits double interrupts and INTCSI and INT4 are a single interrupt –



<1> A generation of INTCSI without permission of double interrupts starts the INTCSI processing program. The status is 1.

<2> The status is changed to 0 by clearing IST0. INTCSI and INT4 that do not enable double interrupts are disabled.

<3> A generation of INTT0 with permission of double interrupts executes double interrupts. The status is changed to 1, and all the interrupts are disabled.

<4> The status is returned to 0 when INTT0 processing is completed.

<5> The disabled INTCSI and INT4 are enabled, and execution returns to the main routine.

6.10 Test Function

6.10.1 Types of test sources

The μPD754304 has two types of test sources. Of these, INT2 is provided with two types of edge-detection testable inputs.

Table 6-5. Types of Test Sources

Test source	Internal/external
INT2 (detects rising edge input to INT2 or falling edge of input to KR0 to KR3)	External

6.10.2 Hardware devices controlling the test function

(1) Test request flag, test enable flag

The test request flag (IRQXXX) is set to “1” when a test request (INTXXX) is generated. When the test processing is completed, it must be cleared to “0” by software.

The test enable flag (IEXXX) is annexed to each test request flag. When it is “1”, a standby release signal is enabled; and when it is “0”, the signal is disabled.

When both the test request flag and test enable flag are set to “1”, a standby release signal is generated.

Table 6-6 lists the set signals for the test request flags.

Table 6-6. Set Signal for Test Request Flag

Test request flag	Set signal for test request flag	Test enable flag
IRQ2	Set by either the rising edge detection of a signal input to the INT2/P12 pin or the falling edge detection of a signal input to the KR0/P60-KR3/P63 pins. The detection edge is selected by the INT2 edge detection mode register (IM2).	IE2

(2) INT2, key interrupt (KR0 to KR7) hardware

Figure 6-10 shows the configuration of INT2 and KR0 to KR7.

The IRQ2 set signal is output by the edge detection at the following two series of pins. The pin is selected by the INT2 edge detection mode register (IM2).

(a) Rising edge detection of input to INT2 pin

The IRQ2 is set when the rising edge of a signal input to the INT2 pin is detected.

(b) Falling edge detection of a signal input to KR0-KR7 pins (key interrupt)

A pin which is used for the interrupt input is selected among the KR0-KR7 pins by the INT2 edge detection mode register (IM2). The IRQ2 is set by the falling edge detection of a signal input to a selected pin.

Figure 6-11 shows the format of the IM2. The IM2 is set by a 4-bit manipulation instruction. All the bits are cleared to "0" by a $\overline{\text{RESET}}$ signal and the rising edge specification by the INT2 is adopted.

Figure 6-10. INT2 and KR0 to KR1 Block Diagram

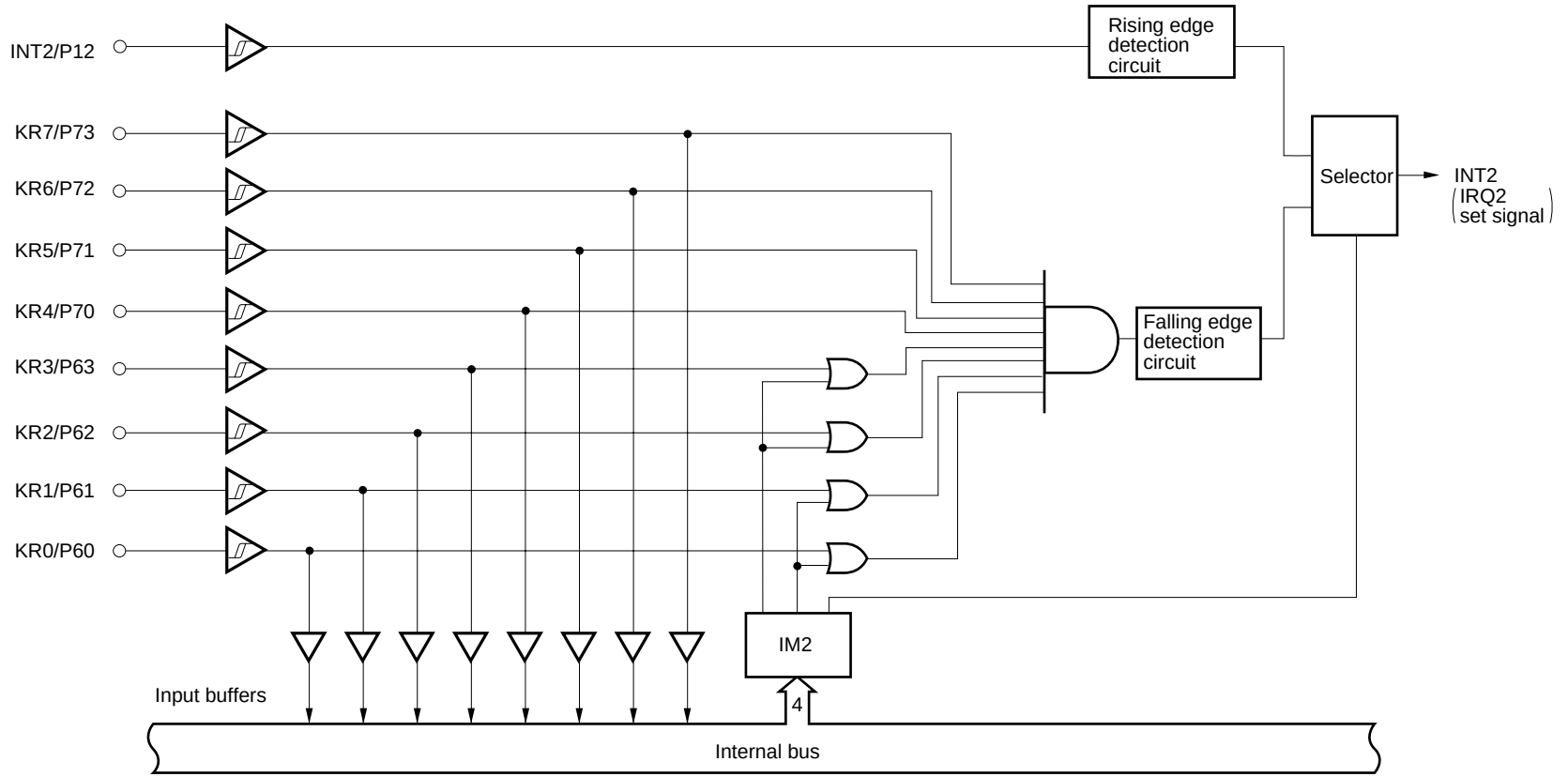


Figure 6-11. Format of INT2 Edge Detection Mode Register (IM2)

Address	3	2	1	0	Symbol
FB6H	0	0	IM21	IM20	IM2

IM21	IM20	INT2 test source	Interrupt input pin
0	0	Rising edge specification by the INT2 pin input	INT2 (1)
0	1	Falling edge specification of any KRX pin input	KR4 to KR7 (4)
1	0		KR2 to KR7 (6)
1	1		KR0 to KR7 (8)

Cautions 1. If the edge detection mode register is changed, the test request flag may be set in some cases; therefore the test input must be disabled beforehand to change the mode register and the test request flag must be cleared by a CLR1 instruction, and then a test input must be enabled.

2. When a low level signal is input to a pin among those pins selected for falling edge detection, the IRQ2 is not set even if falling edges are input to the other pins.

[MEMO]

CHAPTER 7 STANDBY FUNCTION

The μPD754304 has a standby function that reduces the power dissipation of the system. This standby function can be implemented in the following two modes:

- STOP mode
- HALT mode

The functions of the STOP and HALT modes are as follows:

(1) STOP mode

In this mode, the main system clock oscillation circuit is stopped and therefore, the entire system is stopped. The power dissipation of the CPU is substantially reduced.

Moreover, the contents of the data memory can be retained at a low voltage (V_{DD} = 1.8 V MIN.). This mode is therefore useful for retaining the data memory contents with an extremely low current dissipation.

The STOP mode of the μPD754304 can be released by an interrupt request; therefore, the microcomputer can operate intermittently. However, because a wait time is required for stabilizing the oscillation of the clock oscillation circuit when the STOP mode has been released, use the HALT mode if processing must be started immediately after the standby mode has been released by an interrupt request.

(2) HALT mode

In this mode, the operating clock of the CPU is stopped. Oscillation of the system clock oscillation circuit continues. This mode does not reduce the power dissipation as much as the STOP mode, but it is useful when processing must be resumed immediately when an interrupt request is issued, or for an intermittent operation such as a watch operation.

In either mode, all the contents of the registers, flags, and data memory immediately before the standby mode is set are retained. Moreover, the contents of the output latches and output buffers of the I/O ports are also retained; therefore, the statuses of the I/O ports are processed in advance so that the current dissipation of the overall system can be minimized.

The following page describes the points to be noted in using the standby mode.

Cautions 1. Efficient operation with a low current dissipation at a low voltage can be performed by selecting the standby mode, CPU clock, and system clock. In any case, however, the time described in 5.2.3 Setting of system clock and CPU clock is required until the operation is started with the new clock when the clock has been changed by manipulating the control register. To use the clock selecting function and standby mode in combination, therefore, set the standby mode after the time required for selection has elapsed.

2. To use the standby mode, the I/O ports must be processed so that the current dissipation can be minimized. Especially, do not open the input port, and be sure to input low or high level to it.

7.1 Standby Mode Setting and Operation Status

Table 7-1. Operation Status in Standby Mode

Item \ Mode		STOP mode	HALT mode
Set instruction		STOP instruction	HALT instruction
Operation status	Clock generator	Only the main system clock stops oscillation.	Only the CPU clock Φ halts (oscillation continues).
	Basic interval timer/ Watchdog timer	Operation stops.	Enables operation (The IRQBT is set in the reference interval).
	Serial interface	Operable only when an external $\overline{SCK}$ input is selected as the serial clock.	Enables operation
	Timer/event counter	Operable only when a signal input to the TI0 to TI2 pins is specified as the count clock.	Enables operation
	External interrupt	The INT1, 2, and 4 are operable. Only the INT0 is not operated **Note**.	
	CPU	The operation stops.	
Release signal		Interrupt request signal sent from the operable hardware enabled by the interrupt enable flag or $\overline{RESET}$ signal input.	

Note Enables operation only when the noise eliminator is not used (IM02 = 1) by bit 2 of the edge detection mode register(IM0).

The STOP mode is set by a STOP instruction and the HALT mode is set by a HALT instruction. The STOP instruction and HALT instruction set bit 3 and bit 2, respectively, of the PCC.

A NOP instruction must be placed following the STOP instruction and HALT instruction.

When changing the CPU clock by the low-order 2 bits of the PCC, there may be a time difference between PCC updating and CPU clock change as shown in **Table 5-5. Maximum Time Required to Switch System to/from CPU Clocks**. Consequently, when the operating clock before the standby mode and the CPU clock after the standby mode is released are to be changed, the PCC must be updated and then the standby mode must be set after the machine cycle necessary to change the CPU clock has elapsed.

In the standby mode, the data items stored in all the registers and data memory such as the general-purpose register, flags, mode registers, and output latch which stop in the mode are held.

Cautions 1. The STOP mode must not be used in a system in which an external clock is used.

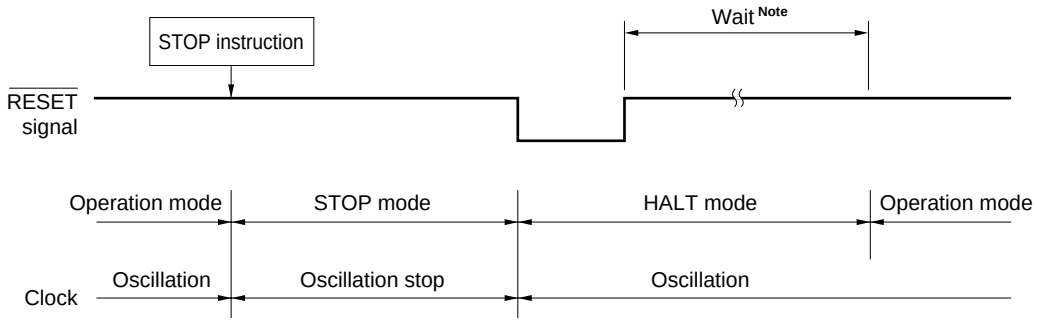
2. Before setting the standby mode, reset all the interrupt request flags.
If there is an interrupt source in which both the test request flag and test enable flag are set, the standby mode is released at the moment the system enters it (See Figure 6-1 Interrupt Control Circuit Block Diagram). However, when the STOP mode is set, the system enters the HALT mode immediately after a STOP instruction is executed and then returns to the operating mode after waiting for a time which is set in the BTM register.

7.2 Standby Mode Release

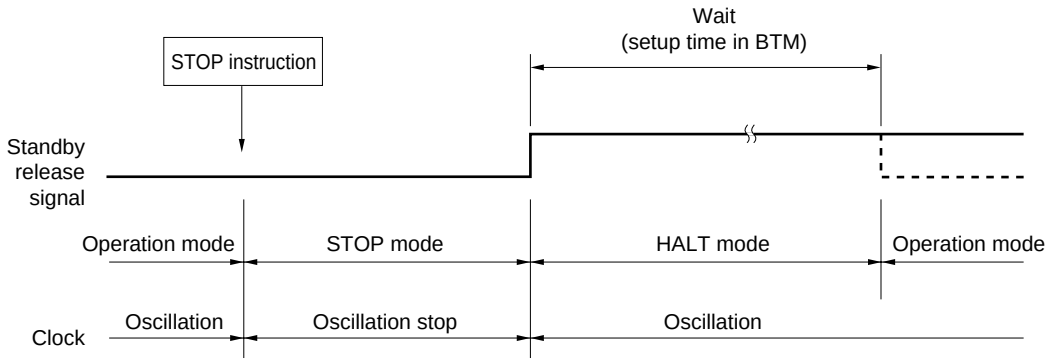
The standby mode (STOP or HALT) is released when an interrupt request signal enabled with an interrupt enable flag occurs or a $\overline{\text{RESET}}$ signal is input. Figure 7-1 shows the standby mode release operation.

Figure 7-1. Standby Mode Release Operation (1/2)

(a) STOP mode release when a $\overline{\text{RESET}}$ signal is generated



(b) STOP mode release when an interrupt occurs



Note Following two wait time can be specified by the mask option.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz, 31.3 ms at 4.19 MHz)

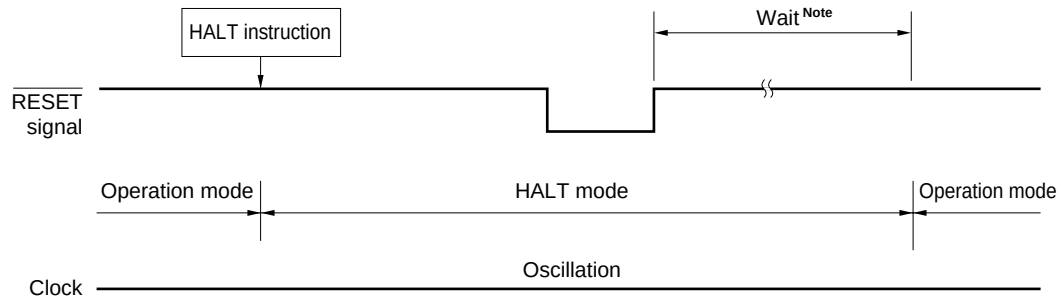
$2^{15}/f_X$ (5.46 ms at 6.00 MHz, 7.81 ms at 4.19 MHz)

★ However, the μPD75P4308 has no mask options and it is fixed to $2^{15}/f_X$.

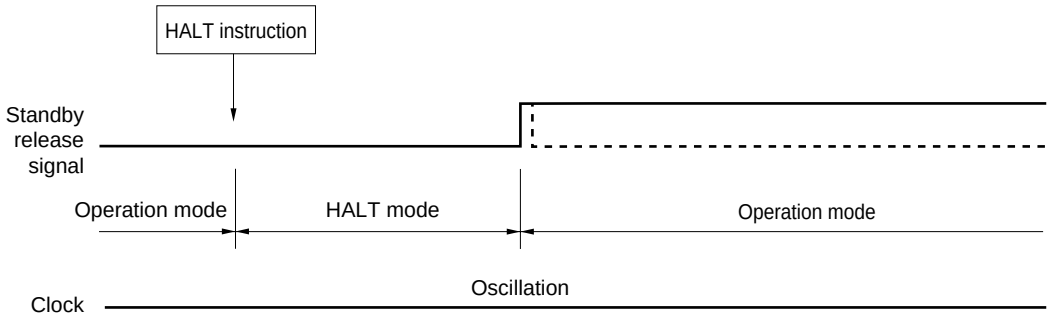
Remark Broken line: When the interrupt request to release the standby mode is acknowledged.

Figure 7-1. Standby Mode Release Operation (2/2)

(c) HALT mode release when a $\overline{\text{RESET}}$ signal is generated



(d) HALT mode release when an interrupt occurs



Note Following two wait times can be specified by the mask option.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz, 31.3 ms at 4.19 MHz)

$2^{15}/f_X$ (5.46 ms at 6.00 MHz, 7.81 ms at 4.19 MHz)

However, the μPD75P4308 has no mask options and it is fixed to $2^{15}/f_X$. ★

Remark Broken line: When the interrupt request to release the standby mode is acknowledged.

When the STOP mode has been released by an interrupt, the wait time is determined by the setting of BTM (refer to **Table 7-2**).

The time required for the oscillation to stabilize varies depending on the type of the oscillator used and the supply voltage when the STOP mode has been released. Therefore, select the appropriate wait time depending on a given condition, and set BTM before setting the STOP mode.

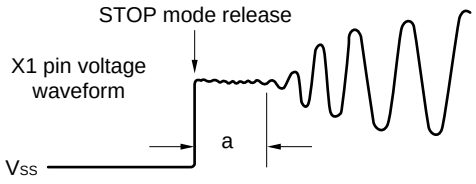
Table 7-2. Wait Time Selection by Using BTM

BTM3	BTM2	BTM1	BTM0	Wait time Note	
				When f_X = 6.00 MHz	When f_X = 4.19 MHz
–	0	0	0	About $2^{20}/f_X$ (about 175 ms)	About $2^{20}/f_X$ (about 250 ms)
–	0	1	1	About $2^{17}/f_X$ (about 21.8 ms)	About $2^{17}/f_X$ (about 31.3 ms)
–	1	0	1	About $2^{15}/f_X$ (about 5.46 ms)	About $2^{15}/f_X$ (about 7.81 ms)
–	1	1	1	About $2^{13}/f_X$ (about 1.37 ms)	About $2^{13}/f_X$ (about 1.95 ms)
Other than the above				Setting prohibited	

Note This time does not include the time until oscillation is started after the STOP mode is released.

Caution The wait time that elapses when the STOP mode has been released does not include the time that elapses until the clock oscillation is started after the STOP mode has been released ('a' in Figure 7-2), regardless of whether the STOP mode has been released by the $\overline{\text{RESET}}$ signal or occurrence of an interrupt.

Figure 7-2. The wait time when STOP mode is released



7.3 Operation After Releasing the Standby Mode

(1) When the standby mode is released by a $\overline{\text{RESET}}$ signal generation, the normal reset operation is performed.

(2) When the standby mode is released by generation of an interrupt, whether a vectored interrupt is to be serviced or not at the time the CPU resumes instruction execution is determined by the contents of the interrupt master enable flag (IME).

(a) When IME = 0

Following the release of the standby mode, the instruction execution resumes starting with the instruction subsequent to the standby mode setting instruction. The interrupt request flag is held.

(b) When IME = 1

After the standby mode is released, two instructions are executed and then a vectored interrupt is executed. However, if the standby mode is released by the INTW and INT2 (testable inputs), a vectored interrupt is not generated; therefore the operations identical to (a) above are performed.

7.4 Application of Standby Mode

Use the standby mode in the following procedure:

<1> Detect the cause that sets the standby mode, such as an interrupt input or power failure by port input (use of INT4 to detect a power failure is recommended).

<2> Process the I/O ports (process so that the current dissipation is minimized).
It is important not to open the input port. Be sure to input a low or high level to it.

<3> Specify an interrupt that releases the standby mode (use of INT4 is effective. Clear the interrupt enable flags of the interrupts that do not release the standby mode).

<4> Specify the operation to be performed after the standby mode has been released (manipulate IME depending on whether interrupt processing is performed or not).

<5> Specify the CPU clock to be used after the standby mode has been released (to change the clock, make sure that the necessary machine cycles elapse before the standby mode is set).

<6> Select the wait time to elapse after the standby mode has been released.

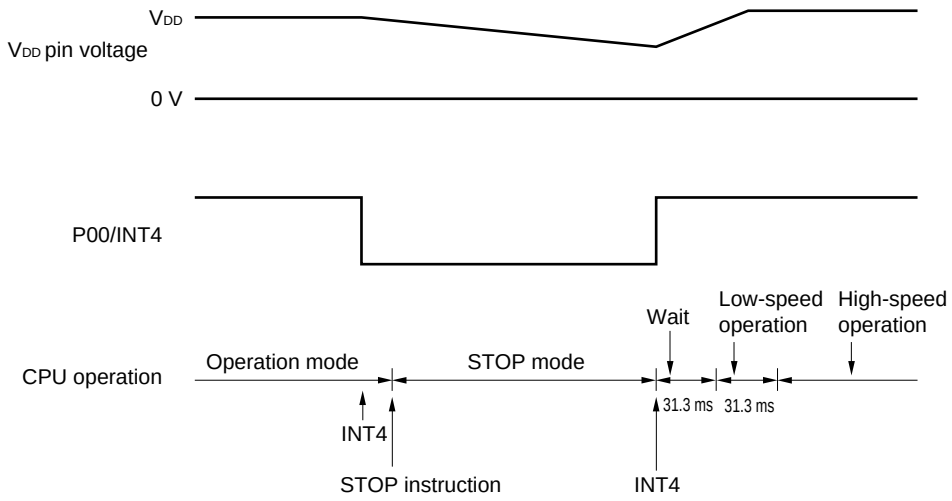
<7> Set the standby mode (by using the STOP or HALT instruction).

Application example of STOP mode (f_X = 4.19 MHz)

<When using the STOP mode under the following conditions>

- The STOP mode is set at the falling edge of INT4 and released at the rising edge (INTBT is not used).
- All the I/O ports go into a high-impedance state (if the pins are externally processed so that the current dissipation is reduced in a high-impedance state).
- Interrupts INT0 and INTT0 are used in the program. However, these interrupts are not used to release the STOP mode.
- The interrupts are enabled even after the STOP mode has been released.
- After the STOP mode has been released, operation is started with the slowest CPU clock. Switch the mode to the high operation in 31.3 ms
- The wait time that elapses after the mode has been released is about 31.3 ms.
- A wait time of 31.3 ms elapses until the power supply stabilizes after the mode has been released. The P00/INT4 pin is double-checked to eliminate chattering.

<Timing chart>



<Program example>

(INT4 processing program, MBE = 0)

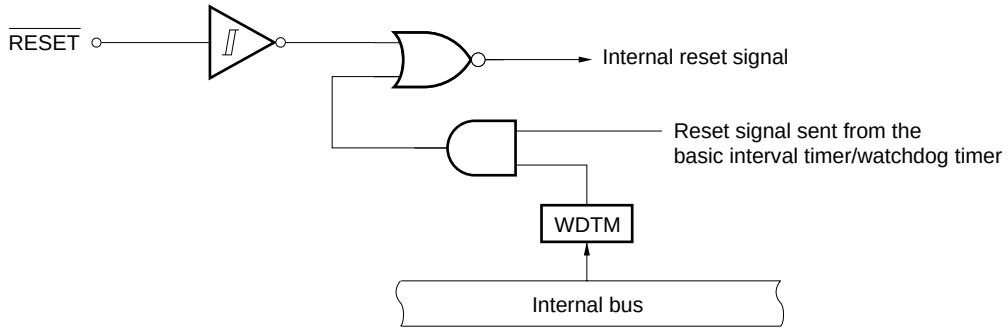
```
VSUB4:  SKT     PORT0.0          ;  P00 = 1 ?
        BR      PDOWN            ;  Power down
        SET1    BTM.3            ;  Power on
WAIT:   SKT     IRQBT            ;  Waits for 31.3 ms
        BR      WAIT
        SKT     PORT0.0          ;  Checks chattering
        BR      PDOWN
        MOV     A, #0011B
        MOV     PCC, A           ;  Sets high-speed mode
      ( MOV     XA, #XXH         )  ;  Sets port mode register
      ( MOV     PMGm, XA         )
        EI      IE0
        EI      IET0
        RETI
PDOWN:  MOV     A, #0            ;  Lowest-speed mode
        MOV     PCC, A
        MOV     XA, #00H
        MOV     PMGA, XA         ;  I/O port in high-impedance state
        MOV     PMGB, XA
        DI      IE0              ;  Disables INT0 and INTT0
        DI      IET0
        MOV     A, #1011B
        MOV     BTM, A           ;  Wait time ≒ 31.3 ms
        NOP
        STOP                     ;  Sets STOP mode
        NOP
        RETI
```

[MEMO]

CHAPTER 8 RESET FUNCTION

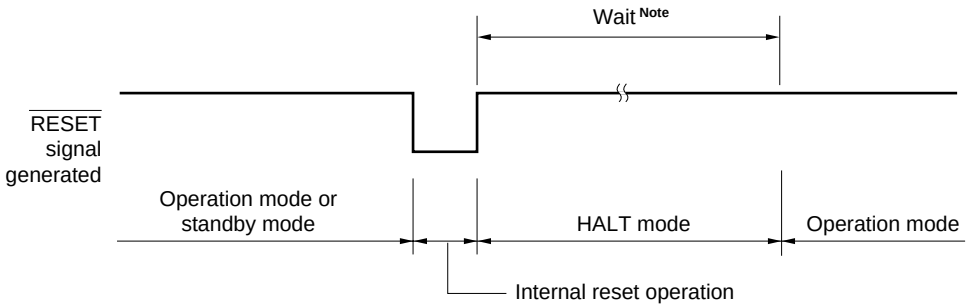
There are two reset inputs: external $\overline{\text{RESET}}$ signal and reset signal sent from the basic interval timer/watchdog timer. When either one of the reset signals are input, an internal reset signal is generated. **Figure 8-1** shows the circuit diagram of the above two inputs.

Figure 8-1. Configuration of Reset Function



Generation of the $\overline{\text{RESET}}$ signal causes each device to be initialized as listed in **Table 8-1. Figure 8-2** shows the timing chart of the reset operation.

Figure 8-2. Reset Operation by $\overline{\text{RESET}}$ Signal Generation



Note The following two times can be selected by the mask option in the μPD754302 and 754304.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz or 31.3 ms at 4.19 MHz)

$2^{15}/f_X$ (5.46 ms at 6.00 MHz or 7.81 ms at 4.19 MHz)

However, the μPD75P4308 is fixed to $2^{15}/fx$.

Table 8-1. Status of Each Device After Reset (1/2)

Hardware			$\overline{\text{RESET}}$ signal generation in the standby mode	$\overline{\text{RESET}}$ signal generation in operation
Program counter (PC)		μPD754302	Sets the low-order 3 bits of program memory's address 0000H to the PC10 to PC8 and the contents of address 0001H to the PC7 to PC0.	Sets the low-order 3 bits of program memory's address 0000H to the PC10 to PC8 and the contents of address 0001H to the PC7 to PC0.
		μPD754304	Sets the low-order 4 bits of program memory's address 0000H to the PC11 to PC8 and the contents of address 0001H to the PC7 to PC0.	Sets the low-order 4 bits of program memory's address 0000H to the PC11 to PC8 and the contents of address 0001H to the PC7 to PC0.
		μPD75P4308	Sets the low-order 5 bits of program memory address 0000H to PC12 to PC8 and contents of address 0001H to PC7 to PC0.	Sets the low-order 5 bits of program memory address 0000H to PC12 to PC8 and contents of address 0001H to PC7 to PC0.
PSW	Carry flag (CY)		Held	Undefined
	Skip flag (SK0-SK2)		0	0
	Interrupt status flag (IST0, IST1)		0	0
	Bank enable flag (MBE, RBE)		Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.	Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.
Stack pointer (SP)			Undefined	Undefined
Stack bank select register (SBS)			1000B	1000B
Data memory (RAM)			Held **Note**	Undefined
General-purpose register (X, A, H, L, D, E, B, C)			Held	Undefined
Bank select register (MBS, RBS)			0, 0	0, 0
Basic interval timer/watchdog timer	Counter (BT)		Undefined	Undefined
	Mode register (BTM)		0	0
	Watchdog timer enable flag (WDTM)		0	0
Timer/event counter (T0)	Counter (T0)		0	0
	Modulo register (TMOD0)		FFH	FFH
	Mode register (TM0)		0	0
	TOE0, TOUT F/F		0, 0	0, 0
Timer/event counter (T1)	Counter (T1)		0	0
	Modulo register (TMOD1)		FFH	FFH
	Mode register (TM1)		0	0
	TOE1, TOUT F/F		0, 0	0, 0

Note The data stored in data memory's addresses 0F8H to 0FDH is undefined by a $\overline{\text{RESET}}$ signal generation.

Table 8-1. Status of Each Device After Reset (2/2)

Hardware		$\overline{\text{RESET}}$ signal generation in the standby mode	$\overline{\text{RESET}}$ signal generation in operation
Serial interface	Shift register (SIO)	Held	Undefined
	Operation mode register (CSIM)	0	0
	SBI control register (SBIC)	0	0
	Slave address register (SVA)	Held	Undefined ★
Clock generator, clock output circuit	Processor clock control register (PCC)	0	0
	Clock output mode register (CLOM)	0	0
Interrupt function	Interrupt request flag (IRQ×××)	Reset (0)	Reset (0)
	Interrupt enable flag (IE×××)	0	0
	Interrupt priority selection register (IPS)	0	0
	INT0, 1, 2 mode registers (IM0, IM1, IM2)	0, 0, 0	0, 0, 0
Digital port	Output buffer	Off	Off
	Output latch	Cleared (0)	Cleared (0)
	I/O mode registers (PMGA, B, C)	0	0
	Pull-up resistor setting register (POGA, B)	0	0
Bit sequential buffer (BSB0 to BSB3)		Held	Undefined

[MEMO]

CHAPTER 9 WRITING AND VERIFYING PROM (PROGRAM MEMORY)

The program memory of the μPD75P4308 is a one-time PROM. The memory capacity is as follows:

μPD75P4308: 8192 words × 8 bits

To write or verify this one-time PROM, the pins shown in **Table 9-1** are used. Note that no address input pins are used and that the address is updated by inputting a clock from the X1 pin.

Table 9-1. Pins Used to Write or Verify Program Memory

Pin name	Function
X1, X2	Inputs clock to update address when program memory is written or verified. Complement of X1 pin is input to X2 pin.
MD0 to MD3 (P30 to P33)	Select operation mode when program memory is written or verified
P60 to P63 (low-order 4 bits) P50 to P53 (high-order 4 bits)	Input or output 8-bit data when program memory is written or verified
V_{DD}	Supplies power supply voltage. Supplies 1.8 to 5.5 V for normal operation and +6 V when program memory is written or verified
V_{PP}	Applies program voltage for writing or verifying program memory (usually, V_{DD})

Cautions 1. The program memory contents of the μPD75P4308 cannot be erased by ultraviolet rays because the μPD75P4308 is not provided with a window for erasure.

2. Connect the pins not used for writing or verifying the program memory to V_{SS} via pull-down resistor.

9.1 Operation Mode for Writing/Verifying Program Memory

When +6 V is applied to the V_{DD} pin of the μPD75P4308 and +12.5 V is applied to the V_{PP} pin, the program memory write/verify mode is set. In this mode, the following operation modes can be selected by using the MD0 through MD3 pins.

Table 9-2. Operation Mode

Specifies operation mode						Operation mode
V_{PP}	V_{DD}	MD0	MD1	MD2	MD3	
+12.5 V	+6 V	H	L	H	L	Clears program memory address to 0
		L	H	H	H	Write mode
		L	L	H	H	Verify mode
		H	×	H	H	Program inhibit mode

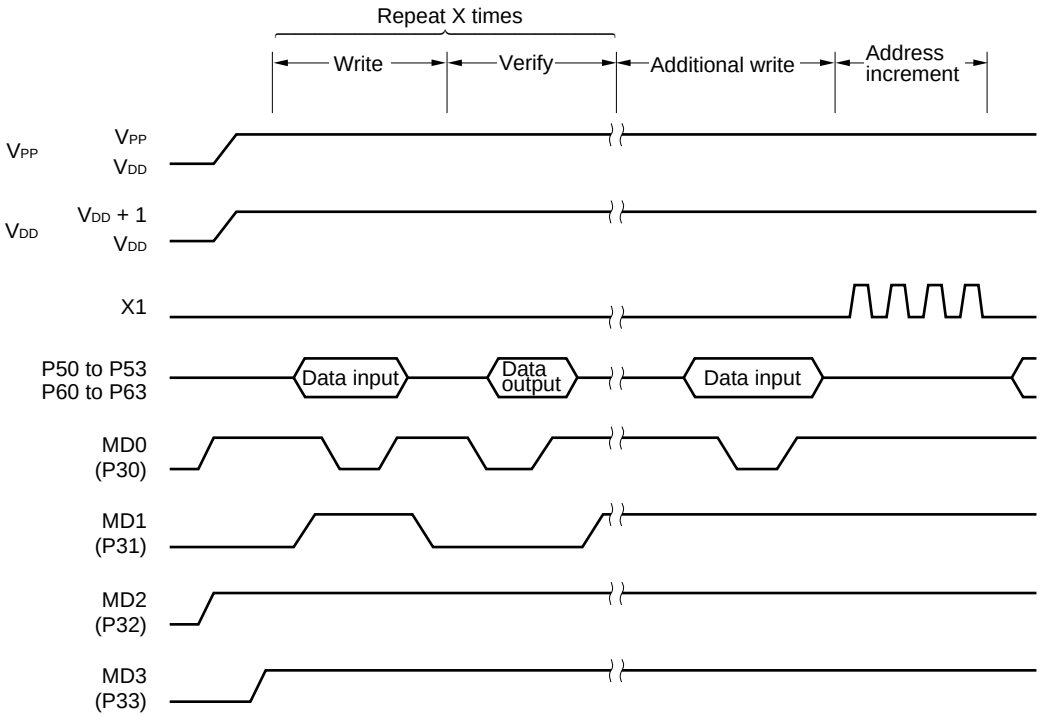
×: L or H

9.2 Writing Program Memory

The program memory can be written in the following procedure at high speed:

(1) Pull down the unused pins to V_{SS} via a resistor. The X1 pin is low level.
(2) Supply 5 V to the V_{DD} and V_{PP} pins.
(3) Wait for 10 μs.
(4) Set the program memory address 0 clear mode.
(5) Supply 6 V to V_{DD} and 12.5 V to V_{PP}.
(6) Set the program inhibit mode.
(7) Write data in the 1-ms write mode
(8) Set the program inhibit mode.
(9) Set the verify mode. If the data have been correctly written, proceed to (10). If not, repeat (7) through (9).
(10) Additional writing of (number of times data have been written in (7) through (9): X) × 1 ms
(11) Set the program inhibit mode.
(12) Input a pulse four times to the X1 pin to update the program memory address (by one).
(13) Repeat (7) through (12) until the last address is written.
(14) Set the program memory address 0 clear mode.
(15) Change the voltage applied to the V_{DD} and V_{PP} pins to 5 V.
(16) Turn off the power supply.

Steps (2) through (12) above are illustrated below.

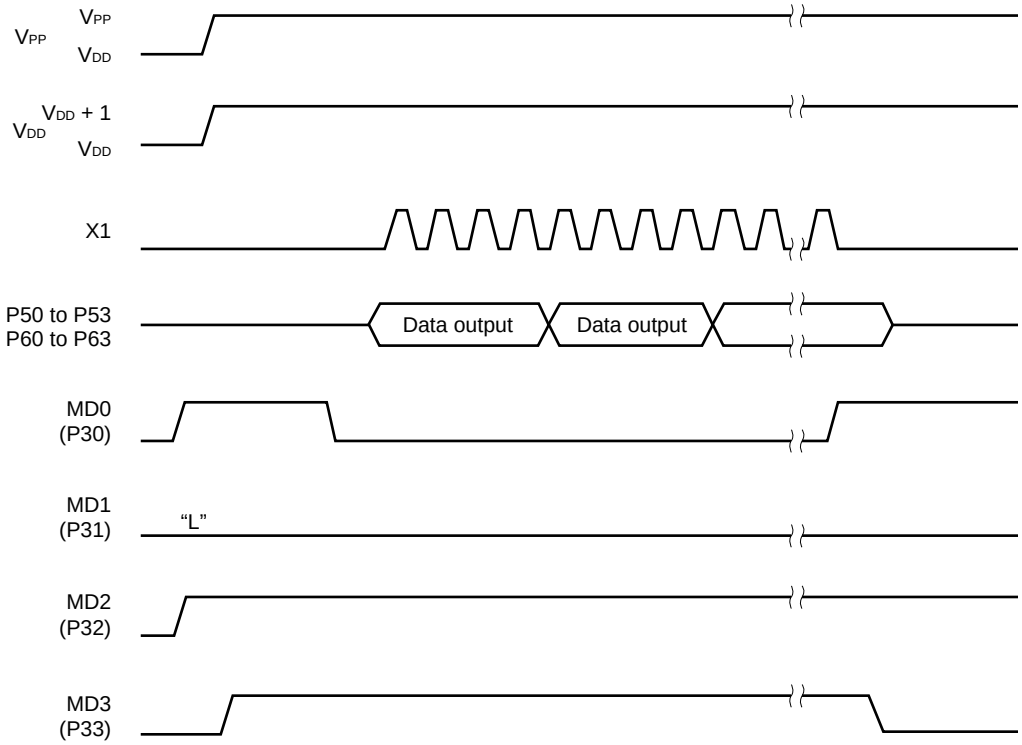


9.3 Reading Program Memory

The contents of the program memory can be read for the μPD75P4308 in the following procedure:

(1) Pull down the unused pins to V_{SS} via a resistor. The X1 pin is low level.
(2) Supply 5 V to the V_{DD} and V_{PP} pins.
(3) Wait for 10 μs.
(4) Set the program memory address 0 clear mode.
(5) Supply 6 V to V_{DD} and 12.5 V to V_{PP}.
(6) Set the program inhibit mode.
(7) Verify mode. Data at each address is sequentially output while four clock pulses are input to the X1 pin.
(8) Set the program inhibit mode.
(9) Set the program memory address 0 clear mode.
(10) Change the voltage applied to the V_{DD} and V_{PP} pins to 5 V.
(11) Turn off the power supply.

Steps (2) through (9) above are illustrated below.



9.4 Screening of One-Time PROM

Because of its structure, it is difficult for NEC to completely test the one-time PROM product before shipment. It is therefore recommended that screening be performed to verify the PROM contents after the necessary data has been written to the PROM and the product has been stored under the following conditions.

Storage temperature	Storage time
125 °C	24 hours

[MEMO]

CHAPTER 10 MASK OPTION ★

10.1 Pins

The pins of the μPD754304 have the following mask options:

Table 10-1. Selecting Mask Option of Pin

Pin	Mask Option
P50 to P53	Pull-up resistor can be connected in 1-bit units.

P50 through P53 (port 5) can be connected with pull-up resistors by mask option. The mask option can be specified in 1-bit units.

If the pull-up resistor is connected by mask option, port 5 goes high on reset. If the pull-up resistor is not connected, the port goes into a high-impedance state on reset.

The port 5 of the μPD75P4308 does not have a mask option and is always open.

10.2 Mask Option of Standby Function

The standby function of the μPD754304 allows you to select wait time by using a mask option. The wait time is required for the CPU to return to the normal operation mode after the standby function has been released by the $\overline{\text{RESET}}$ signal (for details, refer to **7.2 Standby Mode Release**).

The following two wait times can be selected:

<1> $2^{17}/f_X$ (21.8 ms when f_X = 6.00 MHz; 31.3 ms when f_X = 4.19 MHz)
<2> $2^{15}/f_X$ (5.46 ms when f_X = 6.00 MHz; 7.81 ms when f_X = 4.19 MHz)

The μPD75P4308 does not have a mask option and its wait time is fixed to $2^{15}/f_X$.

[MEMO]

CHAPTER 11 INSTRUCTION SET

The instruction set of the μPD754304 is based on the instruction set of the 75X series and therefore, maintains compatibility with the 75X series, but has some improved features. They are:

(1) Bit manipulation instructions for various applications
(2) Efficient 4-bit manipulation instructions
(3) 8-bit manipulation instructions comparable to those of 8-bit microcomputers
(4) GETI instruction reducing program size
(5) String-effect and base number adjustment instructions enhancing program efficiency
(6) Table reference instructions ideal for successive reference
(7) 1-byte relative branch instruction
(8) Easy-to-understand, well-organized NEC's standard mnemonics

For the addressing modes applicable to data memory manipulation and the register banks valid for instruction execution, refer to **3.2 Bank Configuration of General-Purpose Registers**.

11.1 Unique Instructions

This section describes the unique instructions of the instruction set of the μPD754304.

11.1.1 GETI instruction

The GETI instruction converts the following instructions into 1-byte instructions:

(a) Subroutine call instruction to the entire space
(b) Branch instruction to the entire space
(c) Any 2-byte, 2-machine cycle instruction (except BRCB and CALLF instructions)
(d) Combination of two 1-byte instructions

The GETI instruction references a table at addresses 0020H through 007FH of the program memory and executes the referenced 2-byte data as an instruction of (a) to (d). Therefore, 48 types of instructions can be converted into 1-byte instructions.

If instructions that are frequently used are converted into 1-byte instructions by using this GETI instruction, the number of bytes of the program can be substantially decreased.

11.1.2 Bit manipulation instruction

The μPD754304 has reinforced bit test, bit transfer, and bit Boolean (AND, OR, and XOR) instructions, in addition to the ordinary bit manipulation (set and clear) instructions.

The bit to be manipulated is specified in the bit manipulation addressing mode. Three types of bit manipulation addressing modes can be used. The bits manipulated in each addressing mode are shown in Table 11-1.

Table 11-1. Types of Bit Manipulation Addressing Modes and Specification Range

Addressing	Peripheral hardware that can be manipulated	Addressing range of bit that can be manipulated
fmem.bit	RBE, MBE, IST1, IST0, SCC, IE×××, IRQ×××	FB0H through FBFH
	PORT0 to 3, 5 to 8	FF0H through FFFH
pmem.@L	BSB0 to 3, PORT0 to 3, 5 to 8	FC0H through FFFH
@H + mem.bit	All peripheral hardware units that can be manipulated bitwise	All bits of memory bank specified by MB that can be manipulated bitwise

Remarks 1. ××× : 0, 1, 2, 4, BT, T0, T1, T2, W, CSI

2. MB = MBE·MBS

11.1.3 String-effect instruction

The μPD754304 has the following two types of string-effect instructions:

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"String effect" means locating these two types of instructions at contiguous addresses.

Example
```
A0  : MOV  A, #0
A1  : MOV  A, #1
XA7 : MOV  XA, #07
```

When string-effect instructions are arranged as shown in this example, and if the address executed first is A0, the two instructions following this address are replaced with the NOP instructions. If the address executed first is A1, the following one instruction is replaced with the NOP instruction. In other words, only the instruction that is executed first is valid, and all the string-effect instructions that follow are processed as NOP instructions.

By using these string-effect instructions, constants can be efficiently set to the accumulator (A register or register pair XA) and data pointer (register pair HL).

11.1.4 Base number adjustment instruction

Some application requires that the result of addition or subtraction of 4-bit data (which is carried out in binary number) be converted into a decimal number or into a number with a base of 6, such as time.

Therefore, the μPD754304 is provided with base number adjustment instructions that adjusts the result of addition or subtraction of 4-bit data into a number with any base.

(a) Base adjustment of result of addition

Where the base number to which the result of addition executed is to be adjusted is m, the contents of the accumulator and memory (HL) are added in the following combination, and the result is adjusted to a number with a base of m:

```
ADDS   A, #16–m
ADDC   A, @HL    ; A, CY ← A + (HL) + CY
ADDS   A, #m
```

Occurrence of an overflow is indicated by the carry flag.

If a carry occurs as a result of executing the ADDC A, @HL instruction, the ADDS A, #n4 instruction is skipped. If a carry does not occur, the ADDS A, #n4 instruction is executed. At this time, however, the skip function of the instruction is disabled, and the following instruction is not skipped even if a carry occurs as a result of addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

Example To add accumulator and memory in decimal

```
ADDS   A, #6
ADDC   A, @HL    ; A, CY ← A + (HL) + CY
ADDS   A, #10
       ⋮
```

(b) Base adjustment of result of subtraction

Where the base number into which the result of subtraction executed is to be adjusted is m, the contents of memory (HL) are subtracted from those of the accumulator in the following combination, and the result of subtraction is adjusted to a number with a base of m:

```
SUBC A, @HL
ADDS A, #m
```

Occurrence of an underflow is indicated by the carry flag.

If a borrow does not occur as a result of executing the SUBC A, @HL instruction, the following ADDS A, #n4 instruction is skipped. If a borrow occurs, the ADDS A, #n4 instruction is executed. At this time, the skip function of this instruction is disabled, and the following instruction is not skipped, even if a carry occurs as a result of addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

11.1.5 Skip instruction and number of machine cycles required for skipping

The instruction set of the μPD754304 configures a program where instructions may be or may not be skipped if a given condition is satisfied.

If a skip condition is satisfied when a skip instruction is executed, the instruction next to the skip instruction is skipped and the instruction after the next is executed.

When a skip occurs, the number of machine cycles required for skipping is:

(a) If the instruction that follows the skip instruction (i.e., the instruction to be skipped) is a 3-byte instruction (BR !addr, BRA !addr1, CALL !addr, or CALLA !addr1 instruction): 2 machine cycles

(b) Instruction other than (a): 1 machine cycle

11.2 Instruction Sets and their Operations

(1) Expression formats and description methods of operands

The operand is described in the operand column of each instruction in accordance with the description method for the operand expression format of the instruction. For details, refer to **"RA75X ASSEMBLER PACKAGE (Ver 4.5× for Language) (EEU-1363)"**. If there are several elements, one of them is selected. Capital letters and the + and – symbols are key words and are described as they are.

For immediate data, appropriate numbers and labels are described.

Instead of the labels such as mem, fmem, pmem, and bit, the symbols of the registers shown in Figure 3-7 can be described. However, there are restrictions in the labels that can be described for fmem and pmem. For details, see **Table 3-1. Addressing Modes** and **Figure 3-7. μPD754304 I/O Map**.

Representation format	Description method
reg reg1	X, A, B, C, D, E, H, L X, B, C, D, E, H, L
rp rp1 rp2 rp' rp'1	XA, BC, DE, HL BC, DE, HL BC, DE XA, BC, DE, HL, XA', BC', DE', HL' BC, DE, HL, XA', BC', DE', HL'
rpa rpa1	HL, HL+, HL–, DE, DL DE, DL
n4 n8	4-bit immediate data or label 8-bit immediate data or label
mem bit	8-bit immediate data or label **Note** 2-bit immediate data or label
fmem pmem	FB0H-FBFH, FF0H-FFFH immediate data or label FC0H-FFFH immediate data or label
addr addr1 (Only during Mk II mode)	0000H-07FFH immediate data or label (μPD754302) 0000H-0FFFH immediate data or label (μPD754304) 0000H-1FFFH immediate data or label (μPD75P4308)
caddr	12-bit immediate data or label
faddr	11-bit immediate data or label
taddr	20H-7FH immediate data (where bit0 = 0) or label
PORTn IE××× RBn MBn	PORT0 to PORT3, PORT5 to PORT8 IEBT, IET0, IE0 to IE2, IE4, IECSI RB0 to RB3 MB0, MB15

Note mem can be only used for even address in 8-bit data processing.

(2) Legend in explanation of operation

A	: A register, 4-bit accumulator
B	: B register
C	: C register
D	: D register
E	: E register
H	: H register
L	: L register
X	: X register
XA	: XA register pair; 8-bit accumulator
BC	: BC register pair
DE	: DE register pair
HL	: HL register pair
XA'	: XA' expanded register pair
BC'	: BC' expanded register pair
DE'	: DE' expanded register pair
HL'	: HL' expanded register pair
PC	: Program counter
SP	: Stack pointer
CY	: Carry flag, bit accumulator
PSW	: Program status word
MBE	: Memory bank enable flag
RBE	: Register bank enable flag
PORTn	: Port n (n = 0 to 3, 5 to 8)
IME	: Interrupt master enable flag
IPS	: Interrupt priority selection register
IE×××	: Interrupt enable flag
RBS	: Register bank selection register
MBS	: Memory bank selection register
PCC	: Processor clock control register
.	: Separation between address and bit
(××)	: The contents addressed by ××
××H	: Hexadecimal data

(3) Explanation of symbols under addressing area column

*1	MB = MBE • MBS (MBS = 0, 15)	
*2	MB = 0	
*3	MBE = 0 : MB = 0 (00H-7FH) MB = 15 (F80H-FFFH) MBE = 1 : MB = MBS (MBS = 0, 15)	
*4	MB = 15, fmem = FB0H-FBFH, FF0H-FFFH	
*5	MB = 15, pmem = FC0H to FFFH	
*6	μPD754302	addr, addr1 = 0000H-07FFH
	μPD754304	addr, addr1 = 0000H-0FFFH
	μPD75P4308	addr, addr1 = 0000H-1FFFH
*7	addr, addr1 = (Current PC) – 15 to (Current PC) – 1 (Current PC) + 2 to (Current PC) + 16	
*8	μPD754302	caddr = 0000H-07FFH
	μPD754304	caddr = 0000H-0FFFH (PC_{12} = 0)
	μPD75P4308	caddr = 0000H-0FFFH (PC_{12} = 0) or 1000H-1FFFH (PC_{12} = 1)
*9	faddr = 0000H-07FFH	
*10	taddr = 0020H-007FH	
*11	Only during MKII mode addr1 = 0000H-07FFH (μPD754302) 0000H-0FFFH (μPD754304) 0000H-1FFFH (μPD75P4308)	

Data memory addressing

Program memory addressing

Remarks 1. MB indicates memory bank that can be accessed.

2. In *2, MB = 0 independently of how MBE and MBS are set.

3. In *4 and *5, MB = 15 independently of how MBE and MBS are set.

4. *6 to *11 indicate the areas that can be addressed.

(4) Explanation of number of machine cycles column

S denotes the number of machine cycles required by skip operation when a skip instruction is executed. The value of S varies as follows.

- When no skip is made: S = 0
- When the skipped instruction is a 1- or 2-byte instruction: S = 1
- When the skipped instruction is a 3-byte instruction **Note** : S = 2

Note 3-byte instruction: BR !addr, BRA !addr1, CALL !addr or CALLA !addr1 instruction

Caution The GETI instruction is skipped in one machine cycle.

One machine cycle is equal to one cycle of CPU clock (= t_{CY}); time can be selected from among four types by setting PCC (See **Figure 5-12 Processor Clock Control Register Format**).

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Transfer instruction	MOV	A, #n4	1	1	A ← n4		String effect A
		reg1, #n4	2	2	reg1 ← n4		
		XA, #n8	2	2	XA ← n8		String effect A
		HL, #n8	2	2	HL ← n8		String effect B
		rp2, #n8	2	2	rp2 ← n8		
		A, @HL	1	1	A ← (HL)	*1	
		A, @HL+	1	2+S	A ← (HL), then L ← L+1	*1	L = 0
		A, @HL–	1	2+S	A ← (HL), then L ← L–1	*1	L = FH
		A, @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		
		rp'1, XA	2	2	rp'1 ← XA		
	XCH	A, @HL	1	1	A ↔ (HL)	*1	
		A, @HL+	1	2+S	A ↔ (HL), then L ← L+1	*1	L = 0
		A, @HL–	1	2+S	A ↔ (HL), then L ← L–1	*1	L = FH
		A, @rpa1	1	1	A ↔ (rpa)	*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'		

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Table reference	MOVT	XA, @PCDE	1	3	·μPD754302 XA ← $(PC_{10-8}+DE)_{ROM}$		
					·μPD754304 XA ← $(PC_{11-8}+DE)_{ROM}$		
					·μPD75P4308 XA ← $(PC_{12-8}+DE)_{ROM}$		
		XA, @PCXA	1	3	·μPD754302 XA ← $(PC_{10-8}+XA)_{ROM}$		
					·μPD754304 XA ← $(PC_{11-8}+XA)_{ROM}$		
					·μPD75P4308 XA ← $(PC_{12-8}+XA)_{ROM}$		
		XA, @BCDE	1	3	XA ← $(BCDE)_{ROM}$ **Note**	*6	
		XA, @BCXA	1	3	XA ← $(BCXA)_{ROM}$ **Note**	*6	
Bit transfer	MOV1	CY, fmem.bit	2	2	CY ← (fmem.bit)	*4	
		CY, pmem.@L	2	2	CY ← $(pmem_{7-2}+L_{3-2}.bit(L_{1-0}))$	*5	
		CY, @H+mem.bit	2	2	CY ← $(H+mem_{3-0}.bit)$	*1	
		fmem.bit, CY	2	2	(fmem.bit) ← CY	*4	
		pmem.@L, CY	2	2	$(pmem_{7-2}+L_{3-2}.bit(L_{1-0}))$ ← CY	*5	
		@H+mem.bit, CY	2	2	$(H+mem_{3-0}.bit)$ ← CY	*1	
Operation	ADDS	A, #n4	1	1+S	A ← A+n4		carry
		XA, #n8	2	2+S	XA ← XA+n8		carry
		A, @HL	1	1+S	A ← A+(HL)	*1	carry
		XA, rp'	2	2+S	XA ← XA+rp'		carry
		rp'1, XA	2	2+S	rp'1 ← rp'1+XA		carry
	ADDC	A, @HL	1	1	A, CY ← A+(HL)+CY	*1	
		XA, rp'	2	2	XA, CY ← XA+rp'+CY		
		rp'1, XA	2	2	rp'1, CY ← rp'1+XA+CY		

Note For the μPD754302, set 0 to the B register and the high-order 1 bit of the C register.
For the μPD754304, set 0 to the B register.
For the μPD75P4308, only the low-order 1 bit is valid for the B register.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Operation	SUBS	A, @HL	1	1+S	A ← A–(HL)	*1	borrow
		XA, rp'	2	2+S	XA ← XA–rp'		borrow
		rp'1, XA	2	2+S	rp'1 ← rp'1–XA		borrow
	SUBC	A, @HL	1	1	A, CY ← A–(HL)–CY	*1	
		XA, rp'	2	2	XA, CY ← XA–rp'–CY		
		rp'1, XA	2	2	rp'1, CY ← rp'1–XA–CY		
	AND	A, #n4	2	2	A ← A ∧ n4		
		A, @HL	1	1	A ← A ∧ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∧ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∧ XA		
	OR	A, #n4	2	2	A ← A ∨ n4		
		A, @HL	1	1	A ← A ∨ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∨ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∨ XA		
	XOR	A, #n4	2	2	A ← A ∀ n4		
		A, @HL	1	1	A ← A ∀ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∀ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∀ XA		
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 and decrement instructions	INCS	reg	1	1+S	reg ← reg+1		reg=0
		rp1	1	1+S	rp1 ← rp1+1		rp1=00H
		@HL	2	2+S	(HL) ← (HL)+1	*1	(HL)=0
		mem	2	2+S	(mem) ← (mem)+1	*3	(mem)=0
	DECS	reg	1	1+S	reg ← reg–1		reg=FH
		rp'	2	2+S	rp' ← rp'–1		rp'=FFH
Comparison instruction	SKE	reg, #n4	2	2+S	Skip if reg = n4		reg=n4
		@HL, #n4	2	2+S	Skip if (HL) = n4	*1	(HL) = n4
		A, @HL	1	1+S	Skip if A = (HL)	*1	A = (HL)
		XA, @HL	2	2+S	Skip if XA = (HL)	*1	XA = (HL)
		A, reg	2	2+S	Skip if A = reg		A=reg
		XA, rp'	2	2+S	Skip if XA = rp'		XA=rp'

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Carry flag manipulation instruction	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	
		fmem. bit	2	2	(fmem.bit) ← 0	*4	
		pmem. @L	2	2	($pmem_{7-2}$+L_{3-2}.bit(L_{1-0})) ← 0	*5	
		@H+mem. bit	2	2	(H+mem_{3-0}.bit) ← 0	*1	
	SKT	mem. bit	2	2+S	Skip if (mem.bit)=1	*3	(mem.bit)=1
		fmem. bit	2	2+S	Skip if (fmem.bit)=1	*4	(fmem.bit)=1
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))=1	*5	(pmem.@L)=1
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}.bit)=1	*1	(@H+mem.bit)=1
	SKF	mem. bit	2	2+S	Skip if (mem.bit)=0	*3	(mem.bit)=0
		fmem. bit	2	2+S	Skip if (fmem.bit)=0	*4	(fmem.bit)=0
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))=0	*5	(pmem.@L)=0
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}.bit)=0	*1	(@H+mem.bit)=0
	SKTCLR	fmem. bit	2	2+S	Skip if (fmem.bit)=1 and clear	*4	(fmem.bit)=1
		pmem. @L	2	2+S	Skip if ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))=1 and clear	*5	(pmem.@L)=1
		@H+mem. bit	2	2+S	Skip if (H+mem_{3-0}.bit)=1 and clear	*1	(@H+mem.bit)=1
	AND1	CY, fmem. bit	2	2	CY ← CY ∧ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ∧ ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ∧ (H+mem_{3-0}.bit)	*1	
	OR1	CY, fmem. bit	2	2	CY ← CY ∨ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ∨ ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ∨ (H+mem_{3-0}.bit)	*1	
	XOR1	CY, fmem. bit	2	2	CY ← CY ∀ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ∀ ($pmem_{7-2}$+L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ∀ (H+mem_{3-0}.bit)	*1	

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Branch instructions	BR	addr	–	–	• μPD754302 PC_{10-0} ← addr (Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.)	*6	
					• μPD754304 PC_{11-0} ← addr (Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.)		
					• μPD75P4308 PC_{12-0} ← addr (Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.)		
		addr1 Note	–	–	• μPD754302 PC_{10-0} ← addr (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)	*11	
					• μPD754304 PC_{11-0} ← addr1 (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)		
					• μPD75P4308 PC_{12-0} ← addr1 (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)		
		! addr	3	3	• μPD754302 PC_{10-0} ← addr	*6	
					• μPD754304 PC_{11-0} ← addr		
					• μPD75P4308 PC_{12-0} ← addr		
		$addr	1	2	• μPD754302 PC_{10-0} ← addr	*7	
					• μPD754304 PC_{11-0} ← addr		
					• μPD75P4308 PC_{12-0} ← addr		

Note The above operations in the shaded boxes can be performed only in the Mk II mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Branch instruction	BR	$addr1 Note 1	1	2	· μPD754302 PC_{10-0} ← addr1	*7	
					· μPD754304 PC_{11-0} ← addr1		
					· μPD75P4308 PC_{12-0} ← addr1		
		PCDE	2	3	· μPD754302 PC_{10-0} ← PC_{10-8}+DE		
					· μPD754304 PC_{11-0} ← PC_{11-8}+DE		
					· μPD75P4308 PC_{12-0} ← PC_{12-8}+DE		
		PCXA	2	3	· μPD754302 PC_{10-0} ← PC_{10-8}+XA		
					· μPD754304 PC_{11-0} ← PC_{11-8}+XA		
					· μPD75P4308 PC_{12-0} ← PC_{12-8}+XA		
		BCDE	2	3	· μPD754302 PC_{10-0} ← BCDE Note 2	*6	
					· μPD754304 PC_{11-0} ← BCDE Note 3		
					· μPD75P4308 PC_{12-0} ← BCDE Note 4		
		BCXA	2	3	· μPD754302 PC_{10-0} ← BCXA Note 2	*6	
					· μPD754304 PC_{11-0} ← BCXA Note 3		
					· μPD75P4308 PC_{12-0} ← BCXA Note 4		

Notes 1. The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

2. “0” must be set to B register.

3. Only low-order one bit is valid in B register.

4. Only low-order two bits are valid in B register.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Branch instruction	BRA **Note**	!addr1	3	3	• μPD754302 $PC_{10-0} \leftarrow$ addr1	*6	
					• μPD754304 $PC_{11-0} \leftarrow$ addr1		
					• μPD75P4308 $PC_{12-0} \leftarrow$ addr1		
	BRCB	!caddr	2	2	• μPD754302 $PC_{10-0} \leftarrow caddr_{11-0}$	*8	
					• μPD754304 $PC_{11-0} \leftarrow caddr_{11-0}$		
					• μPD75P4308 $PC_{12-0} \leftarrow PC_{12}+caddr_{11-0}$		
Subroutine stack control instructions	CALLA	!addr1	3	3	• μPD754302 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{10-0}$ (SP–5) ← 0, 0, 0, 0 PC_{10-0} ← addr1, SP ← SP–6	*11	
					• μPD754304 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← addr1, SP ← SP–6		
					• μPD75P4308 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) ← 0, 0, $PC_{13, 12}$ PC_{12-0} ← addr1, SP ← SP–6		
	CALL	!addr	3	3	• μPD754302 (SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) $\leftarrow PC_{10-0}$ PC_{10-0} ← addr, SP ← SP–4	*6	
					• μPD754304 (SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) $\leftarrow PC_{11-0}$ PC_{11-0} ← addr, SP ← SP–4		
					• μPD75P4308 (SP–3) ← MBE, RBE, 0, PC_{12} (SP–4) (SP–1) (SP–2) $\leftarrow PC_{11-0}$ PC_{12-0} ← addr1, SP ← SP–4		
				4	• μPD754302 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{10-0}$ (SP–5) ← 0, 0, 0, 0 PC_{10-0} ← addr, SP ← SP–6		
					• μPD754304 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← addr, SP ← SP–6		
					• μPD75P4308 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) ← 0, 0, 0, PC_{12} PC_{12-0} ← addr, SP ← SP–6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Subroutine stack control instructions	CALLF **Note**	!faddr	2	2	· μPD754302 (SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) ← PC_{10-0} PC_{10-0} ← faddr, SP ← SP–4	*9	
					· μPD754304 (SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) ← PC_{11-0} PC_{11-0} ← 0+faddr, SP ← SP–4		
					· μPD75P4308 (SP–3) ← MBE, RBE, $PC_{13, 12}$ (SP–4) (SP–1) (SP–2) ← PC_{11-0} PC_{12-0} ← 00+faddr, SP ← SP–4		
				3	· μPD754302 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{10-0} (SP–5) ← 0, 0, 0, 0 PC_{10-0} ← faddr, SP ← SP–6		
					· μPD754304 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← 0+faddr, SP ← SP–6		
					· μPD75P4308 (SP–2) ← ×, ×, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, PC_{12} PC_{12-0} ← 00+faddr, SP ← SP–6		
	RET **Note**		1	3	· μPD754302 PC_{10-0} ← (SP) (SP+3) (SP+2) MBE, RBE, 0, 0 ← (SP+1), SP ← SP+4		
					· μPD754304 PC_{11-0} ← (SP) (SP+3) (SP+2) MBE, RBE, 0, 0 ← (SP+1), SP ← SP+4		
					· μPD75P4308 PC_{11-0} ← (SP) (SP+3) (SP+2) MBE, RBE, 0, PC_{12} ← (SP+1) SP ← SP+4		
				3	· μPD754302 ×, ×, MBE, RBE ← (SP+4) 0, 0, 0, 0, ← (SP+1) PC_{10-0} ← (SP) (SP+3) (SP+2), SP ← SP+6		
					· μPD754304 ×, ×, MBE, RBE ← (SP+4) MBE, 0, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2), SP ← SP+6		
					· μPD75P4308 ×, ×, MBE, RBE ← (SP+4) 0, 0, 0, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Subroutine stack control instructions	RETS **Note**		1	3+S	• μPD754302 MBE, RBE, 0, 0 ← (SP+1) PC_{10-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		Unconditional
					• μPD754304 MBE, 0, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		
					• μPD75P4308 MBE, RBE, 0, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) SP ← SP+4 then skip unconditionally		
				3+S	• μPD754302 0, 0, 0, 0 ← (SP+1) PC_{10-0} ← (SP) (SP+3) (SP+2) ×, ×, MBE, RBE ← (SP+4) SP ← SP+6 then skip unconditionally		
					• μPD754304 0, 0, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) ×, ×, MBE, RBE ← (SP+4) SP ← SP+6 then skip unconditionally		
					• μPD75P4308 0, 0, 0, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) ×, ×, MBE, RBE ← (SP+4) SP ← SP+6 then skip unconditionally		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Subroutine stack control instructions	RETI **Note**		1	3	· μPD754302 MBE, RBE, 0, 0 ← (SP+1) PC_{10-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		Unconditional
					· μPD754304 MBE, RBE, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		
					· μPD75P4308 MBE, RBE, 0, PC_{12} ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		
					· μPD754302 0, 0, 0, 0 ← (SP+1) PC_{10-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		
					· μPD754304 0, 0, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		
					· μPD75P4308 0, 0, 0, PC_{12} ← SP+1 PC_{11-0} ← (SP) (SP+3) (SP+2) PSW ← (SP+4) (SP+5), SP ← SP+6		
	PUSH	rp	1	1	(SP–1)(SP–2) ← rp, SP ← SP–2		
		BS	2	2	(SP–1) ← MBS, (SP–2) ← RBS, SP ← SP–2		
	POP	rp	1	1	rp ← (SP+1) (SP), SP ← SP+2		
		BS	2	2	MBS ← (SP+1), RBS ← (SP), SP ← SP+2		
Interrupt control instructions	EI		2	2	IME (IPS.3) ← 1		
		IE×××	2	2	IE××× ← 1		
	DI		2	2	IME (IPS.3) ← 0		
		IE×××	2	2	IE××× ← 0		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Input/output instructions	IN Note	A, PORTn	2	2	A ← PORTn (n = 0 to 3, 5 to 8)		
		XA, PORTn	2	2	XA ← PORTn+1, PORTn (n = 6)		
	OUT Note	PORTn, A	2	2	PORTn ← A (n = 2, 3, 5 to 8)		
		PORTn, XA	2	2	PORTn+1, PORTn ← XA (n = 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	RBn	2	2	RBS ← n (n = 0 to 3)		
		MBn	2	2	MBS ← n (n = 0, 15)		

Note For an execution of the IN and/or OUT instruction, be sure to set MBE = 0 or (MBE = 1, MBS = 15).

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Special instructions	GETI **Note**	taddr	1	3	• μPD754302 • When TBR instruction $PC_{10-0} \leftarrow (taddr)_{2-0} + (taddr+1)$	*10	
					• When TCALL instruction (SP–4) (SP–1) (SP–2) ← PC_{10-0} (SP–3) ← MBE, RBE, 0, 0 $PC_{10-0} \leftarrow (taddr)_{2-0} + (taddr+1)$ SP ← SP–4		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction
					• μPD754304 • When TBR instruction $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$		
					• When TCALL instruction (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, RBE, 0, PC_{12} $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$ SP ← SP–4		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction
					• μPD75P4308 • When TBR instruction $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr+1)$		
					• When TCALL instruction (SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, RBE, 0, PC_{12} $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr+1)$ SP ← SP–4		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction

Note The TBR and TCALL instructions are the table definition assembler pseudo instructions of the GETI instruction.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Special instructions	GETI **Note**	taddr	1	3	• μPD754302 • When TBR instruction $PC_{10-0} \leftarrow (taddr)_{2-0} + (taddr+1)$	*10	
				4	• When TCALL instruction (SP–6) (SP–3) (SP–4) $\leftarrow PC_{10-0}$ (SP–5) $\leftarrow$ 0, 0, 0, 0 (SP–2) $\leftarrow \times, \times$, MBE, RBE $PC_{10-0} \leftarrow (taddr)_{2-0} + (taddr+1)$ SP $\leftarrow$ SP–6		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction
				3	• μPD754304 • When TBR instruction $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$		
				4	• When TCALL instruction (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) $\leftarrow$ 0, 0, 0, PC_{12} (SP–2) $\leftarrow \times, \times$, MBE, RBE $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$ SP $\leftarrow$ SP–6		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction
				3	• μPD75P4308 • When TBR instruction $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr+1)$		
				4	• When TCALL instruction (SP–6) (SP–3) (SP–4) $\leftarrow PC_{11-0}$ (SP–5) $\leftarrow$ MBE, RBE, PC_{12} (SP–2) $\leftarrow \times, \times$, MBE, RBE $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr+1)$ SP $\leftarrow$ SP–6		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction

Note The TBR, TCALL instructions are the table definition assembler pseudo instructions of the GETI instructions.

The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

11.3 Op Code of Each Instruction

(1) Description of symbol of op code

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

reg, reg1

P_2 P_1 P_0	reg-pair
0 0 0	XA
0 0 1	XA’
0 1 0	HL
0 1 1	HL’
1 0 0	DE
1 0 1	DE’
1 1 0	BC
1 1 1	BC’

rp’, rp’1

Q_2 Q_1 Q_0	addressing
0 0 0	@HL
0 1 0	@HL+
0 1 1	@HL–
1 0 0	@DE
1 0 1	@DL

@rpa, @rpa1

P_2 P_1	reg-pair
0 0	XA
0 1	HL
1 0	DE
1 1	BC

rp2, rp1, rp

N_5 N_2 N_1 N_0	IE×××
0 0 0 0	IEBT
0 0 1 0	IEW
0 1 0 0	IET0
0 1 0 1	IECSI
0 1 1 0	IE0
0 1 1 1	IE2
1 0 0 0	IE4
1 1 0 0	IET1
1 1 0 1	IET2
1 1 1 0	IE1

I_n : immediate data for n4 or n8
D_n : immediate data for mem
B_n : immediate data for bit
N_n : immediate data for n or IE×××
T_n : immediate data for taddr × 1/2
A_n : immediate data for [relative address distance from branch destination address (2 – 16)] – 1
S_n : immediate data for 1’s complement of [relative address distance from branch destination address (15 – 1)]

(2) Op code for bit manipulation addressing

∗1 in the operand field indicates the following three types:

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

The second byte ∗2 of the op code corresponding to the above addressing is as follows:

∗1	2nd byte of of code								Accessible bit
fmem. bit	1	0	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FB0H-FBFH that can be manipulated
	1	1	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FF0H-FFFH that can be manipulated
pmem. @L	0	1	0	0	G_3	G_2	G_1	G_0	Bit of FC0H-FFFH that can be manipulated
@H+mem. bit	0	0	B_1	B_0	D_3	D_2	D_1	D_0	Bit of accessible memory bank that can be manipulated

B_n : immediate data for bit

F_n : immediate data for fmem (indicates low-order 4 bits of address)

G_n : immediate data for pmem (indicates bits 5-2 of address)

D_n : immediate data for mem (indicates low-order 4 bits of address)

Instruction	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Transfer	MOV	A, #n4	0 1 1 1 I_3 I_2 I_1 I_0		
		reg1, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 1 R_2 R_1 R_0	
		rp, #n8	1 0 0 0 1 P_2 P_1 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @rpa1	1 1 1 0 0 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 0	
		@HL, A	1 1 1 0 1 0 0 0		
		@HL, XA	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 0	
		A, mem	1 0 1 0 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		mem, A	1 0 0 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		mem, XA	1 0 0 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg	1 0 0 1 1 0 0 1	0 1 1 1 1 R_2 R_1 R_0	
		XA, rp’	1 0 1 0 1 0 1 0	0 1 0 1 1 P_2 P_1 P_0	
		reg1, A	1 0 0 1 1 0 0 1	0 1 1 1 0 R_2 R_1 R_0	
		rp’1, XA	1 0 1 0 1 0 1 0	0 1 0 1 0 P_2 P_1 P_0	
	XCH	A, @rpa1	1 1 1 0 1 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 1	
		A, mem	1 0 1 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg1	1 1 0 1 1 R_2 R_1 R_0		
		XA, rp’	1 0 1 0 1 0 1 0	0 1 0 0 0 P_2 P_1 P_0	
Table lookup	MOVT	XA, @PCDE	1 1 0 1 0 1 0 0		
		XA, @PCXA	1 1 0 1 0 0 0 0		
		XA, @BCXA	1 1 0 1 0 0 0 1		
		XA, @BCDE	1 1 0 1 0 1 0 1		
Bit transfer	MOV1	CY, ∗1	1 0 1 1 1 1 0 1	*2	
		∗1, CY	1 0 0 1 1 0 1 1	*2	

Instruction	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Operation	ADDS	A, #n4	0 1 1 0 I_3 I_2 I_1 I_0		
		XA, #n8	1 0 1 1 1 0 0 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @HL	1 1 0 1 0 0 1 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 0 0 P_2 P_1 P_0	
	ADDC	A, @HL	1 0 1 0 1 0 0 1		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 1 0 P_2 P_1 P_0	
	SUBS	A, @HL	1 0 1 0 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 0 0 P_2 P_1 P_0	
	SUBC	A, @HL	1 0 1 1 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 1 0 P_2 P_1 P_0	
	AND	A, #n4	1 0 0 1 1 0 0 1	0 0 1 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 0 1 0 P_2 P_1 P_0	
	OR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 0 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 0 0 P_2 P_1 P_0	
	XOR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 1 0 P_2 P_1 P_0	
Accumulator manipula-tion	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	

Instruction	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Increment/ decrement	INCS	reg	1 1 0 0 0 R_2 R_1 R_0		
		rp1	1 0 0 0 1 P_2 P_1 0		
		@HL	1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0	
		mem	1 0 0 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
	DECS	reg	1 1 0 0 1 R_2 R_1 R_0		
		rp'	1 0 1 0 1 0 1 0	0 1 1 0 1 P_2 P_1 P0	
Comparison	SKE	reg, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 0 R_2 R_1 R_0	
		@HL, #n4	1 0 0 1 1 0 0 1	0 1 1 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 0 0 0 0 0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 1	
		A, reg	1 0 0 1 1 0 0 1	0 0 0 0 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 1 P_2 P_1 P_0	
Carry flag manipulation	SET1	CY	1 1 1 0 0 1 1 1		
	CLR1	CY	1 1 1 0 0 1 1 0		
	SKT	CY	1 1 0 1 0 1 1 1		
	NOT1	CY	1 1 0 1 0 1 1 0		
Memory bit manipula- tion	SET1	mem. bit	1 0 B_1 B_0 0 1 0 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		*1	1 0 0 1 1 1 0 1	*2	
	CLR1	mem. bit	1 0 B_1 B_0 0 1 0 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		*1	1 0 0 1 1 1 0 0	*2	
	SKT	mem. bit	1 0 B_1 B_0 0 1 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		*1	1 0 1 1 1 1 1 1	*2	
	SKF	mem. bit	1 0 B_1 B_0 0 1 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		*1	1 0 1 1 1 1 1 0	*2	
	SKTCLR	*1	1 0 0 1 1 1 1 1	*2	
	AND1	CY, *1	1 0 1 0 1 1 0 0	*2	
	OR1	CY, *1	1 0 1 0 1 1 1 0	*2	
	XOR1	CY, *1	1 0 1 1 1 1 0 0	*2	

Instruction	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Branch	BR	! addr	1 0 1 0 1 0 1 1	0 0 ← addr →	
		$ addr (+16) to (+2)	0 0 0 0 A_3 A_2 A_1 A_0		
		$ addr (–1) to (–15)	1 1 1 1 S_3 S_2 S_1 S_0		
		PCDE	1 0 0 1 1 0 0 1	0 0 0 0 0 1 0 0	
		PCXA	1 0 0 1 1 0 0 1	0 0 0 0 0 0 0 0	
		BCDE	1 0 0 1 1 0 0 1	0 0 0 0 0 1 0 1	
		BCXA	1 0 0 1 1 0 0 1	0 0 0 0 0 0 0 1	
	BRA	! addr1	1 0 1 1 1 0 1 0	0 ← addr1 →	
	BRCB	! caddr	0 1 0 1 ← caddr →		
Subroutine/ stack control	CALL	! addr	1 0 1 0 1 0 1 1	0 1 ← addr →	
	CALLA	! addr1	1 0 1 1 1 0 1 1	0 ← addr1 →	
	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	
I/O	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	EI		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 1	1 0 N_5 1 1 N_2 N_1 N_0	
	DI		1 0 0 1 1 1 0 0	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 0	1 0 N_5 1 1 N_2 N_1 N_0	
CPU control	HALT		1 0 0 1 1 1 0 1	1 0 1 0 0 0 1 1	
	STOP		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 1	
	NOP		0 1 1 0 0 0 0 0		
Special	SEL	RBn	1 0 0 1 1 0 0 1	0 0 1 0 0 0 N_1 N_0	
		MBn	1 0 0 1 1 0 0 1	0 0 0 1 N_3 N_2 N_1 N_0	
	GETI	taddr	0 0 T_5 T_4 T_3 T_2 T_1 T_0		

11.4 Instruction Function and Application

This section describes the functions and applications of the respective instructions. The instructions that can be used and the functions of the instructions differ between the Mk I and Mk II modes of the μPD754302, 754304, and 75P4308. Read the descriptions on the following pages according to the following guidance:

How to read

( ) : This instruction can be used commonly to all the following:
- μPD754302
- μPD754304
- μPD75P4308

} In Mk I and Mk II modes

(I) : This instruction can be used only in the Mk I mode of the μPD754302, 754304, and 75P4308.

(II) : This instruction can be used only in the Mk II mode of the μPD754302, 754304, and 75P4308.

(I/II) : This instruction can be used commonly in the Mk I and Mk II modes of the μPD754302, 754304, and 75P4308, but the function may differ between the Mk I and Mk II modes.
In the Mk I mode, refer to the description under the heading [Mk I mode]. In the Mk II mode, read the description under the heading [Mk II mode].

Remark In this section, it is assumed that the 12-bit program counter of the μPD754304 is used. Note that the program counter of the μPD754302 is 11 bits wide, and that of the μPD75P4308 is 13 bits wide.

11.4.1 Transfer instructions

MOV A, #n4

Function: A ← n4 n4 = I_{3-0}: 0-FH

Transfers 4-bit immediate data n4 to the A register (4-bit accumulator). This instruction has a string effect (group A). Therefore, if this instruction is followed by another MOV A, #n4 or MOV XA, #n8, the following instruction will be processed as a NOP instruction.

Application example

(1) To set 0BH to the accumulator

```
MOV A, #08H
```

(2) To select data output to port 3 from 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} 0-FH

Transfers 4-bit immediate data n4 to A register reg1 (X, H, L, D, E, B, or C).

MOV XA, #n8

Function: XA ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair XA. This instruction has a string effect, and if two or more of this instruction are executed in succession or if this instruction is followed by the MOV A, #n4 instruction, the instruction following this instruction is treated as NOP.

MOV HL, #n8

Function: HL ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair HL. This instruction has a string effect. If two or more of this instructions are executed in succession, those that follow the first instruction are treated as NOP.

MOV rp2, #n8

Function: rp2 ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair rp2 (BC, DE).

MOV A, @HL

Function: A ← (HL)

The contents of the data memory addressed by the HL register pair are transferred to the A register.

MOV A, @HL+

Function: A ← (HL), L ← L+1
skip if L = 0H

The contents of the data memory addressed by the HL register pair are transferred to the A register. Then the contents of the L register are automatically incremented by one. If the contents of the L register become OH as a result, the next one instruction is skipped.

MOV A, @HL–

Function: A ← (HL), L ← L–1
skip if L = FH

The contents of the data memory addressed by the HL register pair are transferred to the A register. Then the contents of the L register are automatically decremented by one. If the contents of the L register become FH as a result, the next one instruction is skipped.

MOV A, @rpa1

Function: A ← (Register pair specified with an operand)
when register pair rpa = HL+ : skip if L = 0
when register pair rpa = HL– : skip if L = FH

Transfers the contents of the data memory addressed by a register pair rpa (HL, HL+, HL–, DE, or DL) to the A register.

If autoincrement is specified as a register pair rpa (HL+), the contents of the L register are automatically incremented by one after the data has been transferred. If the contents of the L register become 0 as a result, the next one instruction is skipped.

If autodecrement is specified as a register pair rpa (HL–), the contents of the L register are automatically decremented by one after the data has been transferred. If the contents of the L register become FH as a result, the next one instruction is skipped.

MOV XA, @HL

Function: A ← (HL), X ← (HL+1)

Transfers the contents of the data memory addressed by register pair HL to the A register, and the contents of the next memory address to the X register.

If the contents of the L register are a odd number, an address whose least significant bit is ignored is transferred.

Application example

To transfer the data at addresses 3EH and 3FH to register pair XA

```
MOV HL, #3EH
MOV XA, @HL
```

MOV @HL, A

Function: (HL) ← A

Transfers the contents of the A register to the data memory addressed by register pair HL.

MOV @HL, XA

Function: (HL) ← A, (HL+1) ← X

Transfers the contents of the A register to the data memory addressed by register pair HL, and the contents of the X register to the next memory address.

However, if the contents of the L register are a odd number, an address whose least significant bit is ignored is transferred.

MOV A, mem

Function: A ← (mem) mem = D_{7-0}: 00H-FFH

Transfers the contents of the data memory addressed by 8-bit immediate data to the A register.

MOV XA, mem

Function: A ← (mem), X ← (mem+1) mem = D_{7-0}: 00H-FEH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register and the contents of the next address to the X register.

The address that can be specified by mem is an even address.

Application example

To transfer the data at addresses 40H and 41H to register pair XA

```
MOV XA, 40H
```

MOV mem, A

Function: (mem) ← A mem = D_{7-0}: 00H-FFH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem.

MOV mem, XA

Function: (mem) ← A, (mem+1) ← X mem = D_{7-0}: 00H-FFH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data and the contents of the X register to the next memory address.

The address that can be specified by mem is an even address.

MOV A, reg

Function: A ← reg

Transfers the contents of register reg (X, A, H, L, D, E, B, or C) to the A register.

MOV XA, rp'

Function: XA ← rp'

Transfers the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') to register pair XA.

Application example

To transfer the data of register pair XA' to register pair XA

```
MOV XA, XA'
```

MOV reg1, A

Function: reg1 ← A

Transfers the contents of the A register to register reg1 (X, H, L, D, E, B, or C).

MOV rp'1, XA

Function: rp'1 ← XA

Transfers the contents of register pair XA to register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC').

XCH A, @HL

Function: A ↔ (HL)

Exchanges the contents of the A register with the contents of the data memory addressed by a register pair HL.

XCH A, @HL+

Function: A ↔ (HL) , L ← L + 1
skip if L = OH

Exchanges the contents of the A register with the contents of the data memory addressed by a register pair HL. Then the contents of the L register are automatically incremented by one. If the contents of the L register become OH as a result, the next one instruction is skipped.

XCH A, @HL–

Function: A ↔ (HL) , L ← L – 1
skip if L = FH

Exchanges the contents of the A register with the contents of the data memory addressed by a register pair HL. Then the contents of the L register are automatically decremented by one. If the contents of the L register become FH as a result, the next one instruction is skipped.

XCH A, @rpa1

Function: A ↔ (Register pair rpa specified with an operand)

when register pair rpa = HL+ : skip if L = 0
when register pair rpa = HL– : skip if L = FH

Exchanges the contents of the A register with the contents of the data memory addressed by a register pair rpa (HL, HL+, HL–, DE, or DL).

If autoincrement is specified as a register pair rpa (HL+), the contents of the L register are automatically incremented by one after the data have been exchanged. If the increment result is 0, the next one instruction is skipped.

If autodecrement is specified as a register pair rpa (HL–), the contents of the L register are automatically decremented by one after the data have been exchanged. If the decrement result is FH, the next one instruction is skipped.

Application example

To exchange the data at data memory addresses 20H through 2FH with the data at addresses 30H through 3FH

```
        SEL   MB0
        MOV   D, #2
        MOV   HL, #30H
LOOP:   XCH   A, @HL    ; A ↔ (3×)
        XCH   A, @DL    ; A ↔ (2×)
        XCH   A, @HL+   ; A ↔ (3×)
        BR    LOOP
```

XCH XA, @HL

Function: A ↔ (HL), X ↔ (HL+1)

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL, and the contents of the X register with the contents of the next address.

If the contents of the L register are odd numbers, however, an address whose least significant bit is ignored is specified.

XCH A, mem

Function: A ↔ (mem) mem = D_{7-0}: 00H-FFH

Exchanges the contents of the A register with the contents of the data memory addressed by 8-bit immediate data mem.

XCH XA, mem

Function: A ↔ (mem), X ↔ (mem+1) mem = D_{7-0}: 00H-FEH

Exchanges the contents of the A register with the data memory contents addressed by 8-bit immediate data mem, and the contents of the X register with the contents of the next memory address.

The address that can be specified by mem is an even address.

XCH A, reg1

Function: A ↔ reg1

Exchanges the contents of the A register with the contents of register reg1 (X, H, L, D, E, B, or C).

XCH XA, rp'

Function: XA ↔ rp'

Exchanges the contents of register pair XA with the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC').

11.4.2 Table reference instruction

MOVT XA, @PCDE

Function: μPD754304 XA ← ROM (PC_{11-8}+DE)

Transfers the low-order 4 bits of the table data in the program memory addressed to the A resister when the low-order 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair DE, and the high-order 4 bits to the X register.

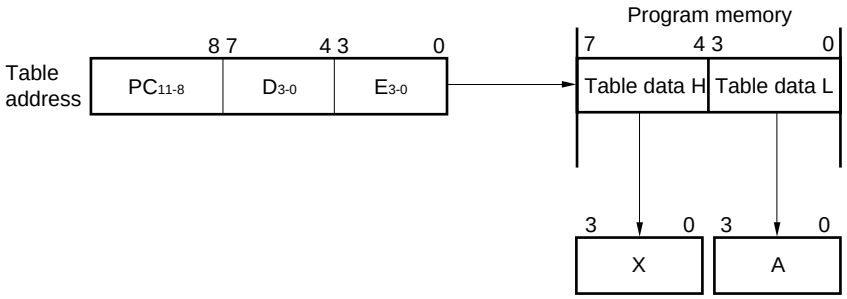
The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudoinstruction (DB instruction).

The program counter is not affected by execution of this instruction.

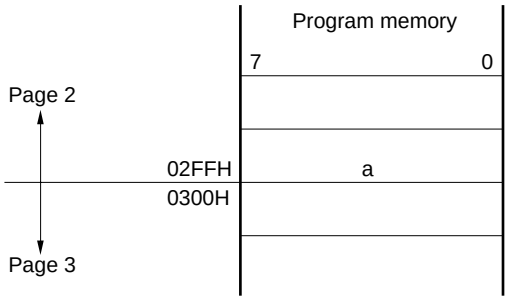
This instruction is useful for successively referencing table data.

Example In the case of μPD754304



Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

Caution The MOVT XA, @PCDE instruction usually references the table data in page where the instruction exists. If the instruction is at address ××FFH, however, the table data in the page where the instruction exists is not referenced, but the table data in the next page is.



For example, if the MOV XA, @PCDE instruction is located at position a in the above figure, the table data in page 3, not page 2, specified by the contents of register pair DE is transferred to register pair XA.

Application example

To transfer the 16-byte data at program memory addresses ××F0H through ××FFH to data memory addresses 30H through 4FH

```
SUB:   SEL    MB0
       MOV    HL, #30H      ; HL ← 30H
       MOV    DE, #0F0H     ; DE ← F0H
LOOP:  MOVT   XA, @PCDE     ; XA ← table data
       MOV    @HL, XA       ; (HL) ← XA
       INCS   HL            ; HL ← HL+2
       INCS   HL
       INCS   E             ; E ← E+1
       BR     LOOP
       RET
       ORG    ××F0H
       DB     ××H, ××H, ⋯   ; table data
```

MOVT XA, @PCXA

Function: μPD754304 XA ← ROM (PC_{11-8}+XA)

Transfers the low-order 4 bits of the table data in the program memory addressed to the A resister when the low-order 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair XA, and the high-order 4 bits to the X register.

The table address is determined by the contents of the PC when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudoinstruction (DB instruction). The PC is not affected by execution of this instruction.

Caution If an instruction exists at address ××FFH, the table data of the next page is transferred, in the same manner as MOVT XA, @PCDE.

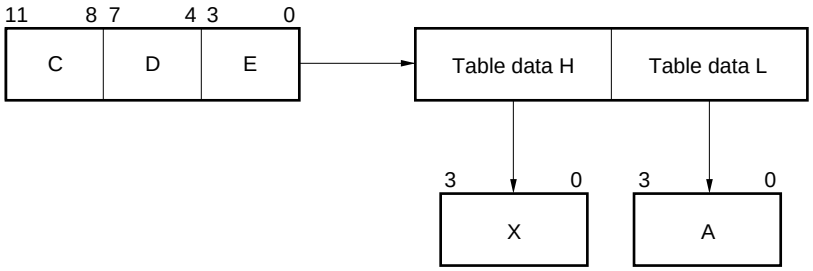
Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

MOVT XA, @BCDE

Function: μPD754304 XA ← (BCDE) ROM

Transfers the low-order 4 bits of the table data (8-bit) in the program memory addressed by the least significant bit of register B and the contents of registers C, D, and E, to the A register, and the high-order 4 bits to the X register.

The necessary data must be programmed to the table area in advance by using an assembler pseudoinstruction (DB instruction). The PC is not affected by execution of this instruction.

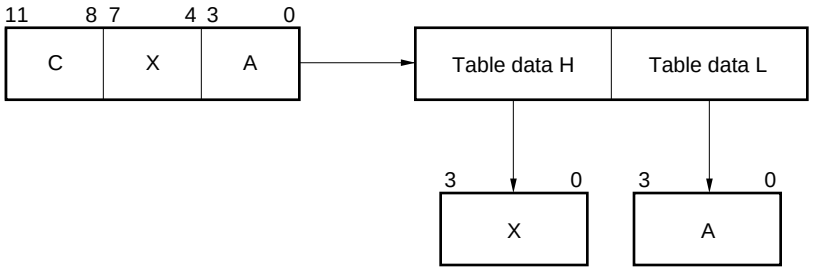


MOVT XA, @BCXA

Function: μPD754304 XA ← (BCXA) ROM

Transfers the low-order 4 bits of the table data (8-bit) in the program memory addressed by the least significant bit of register B and the contents of registers C, X, and A, to the A register, and the higher 4 bits to the X register.

The necessary data must be programmed to the table area in advance by using an assembler pseudoinstruction (DB instruction). The PC is not affected by execution of this instruction.



Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

11.4.3 Bit transfer instruction

MOV1 CY, fmem. bit

MOV1 CY, pmem. @L

MOV1 CY, @H+mem. bit

Function: CY ← (bit specified by operand)

Transfers the contents of the data memory addressed in the bit manipulating addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) to the carry flag (CY).

MOV1 fmem. bit, CY

MOV1 pmem. @L, CY

MOV1 @H+mem. bit, CY

Function: (Bit specified by operand) ← CY

Transfers the contents of the carry flag (CY) to the data memory bit addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit).

Application example

To output the flag of bit 3 at data memory address 3FH to the bit 2 of port 3

```
FLAG   EQU 3FH.3
SEL    MB0
MOV    H, #FLAG SHR6 ; H ← higher 4 bits of FLAG
MOV1   CY, @H+FLAG   ; CY ← FLAG
MOV1   PORT3. 2, CY  ; P32 ← CY
```

11.4.4 Operation instructions

ADDS A, #n4

Function: A ← A + n4; Skip if carry. n4 = I_{3-0}: 0-FH

Adds 4-bit immediate data n4 to the contents of the A register in binary. If a carry occurs as a result, the next instruction is skipped. The carry flag is not affected.

If this instruction is used in combination with ADDC A, @HL or SUBC A, @HL instruction, it can be used as a base number adjustment instruction (refer to **11.1 Unique Instructions**).

ADDS XA, #n8

Function: XA ← XA + n8; Skip if carry. n8 = I_{7-0}: 00H-FFH

Adds 8-bit immediate data n8 to the contents of register pair XA in binary. If a carry occurs as a result, the next instruction is skipped. The carry flag is not affected.

ADDS A, @HL

Function: A ← A + (HL); Skip if carry.

Adds the contents of the data memory addressed by register pair HL to the contents of the A register in binary. If a carry occurs as a result, the next instruction is skipped. The carry flag is not affected.

ADDS XA, rp'

Function: XA ← XA + rp'; Skip if carry.

Adds the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') to the contents of register pair XA in binary. If a carry occurs as a result, the next instruction is skipped. The carry flag is not affected.

ADDS rp'1, XA

Function: rp' ← rp'1 + XA; Skip if carry.

Adds the contents of register pair XA to register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC') in binary. If a carry occurs as a result, the next instruction is skipped. The carry flag is not affected.

Application example

To shift a register pair to the left

```
MOV    XA, rp'1
ADDS   rp'1, XA
NOP
```

ADDC A, @HL

Function: A, CY ← A + (HL) + CY

Adds the contents of the data memory addressed by register pair HL to the contents of the A register in binary, including the carry flag. If a carry occurs as a result, the carry flag is set; if not, the carry flag is reset.

If the ADDS A, #n4 instruction is placed following this instruction, and if a carry occurs as a result of executing this instruction, the ADDS A, #n4 instruction is skipped. If a carry does not occur, the ADDS A, #n4 instruction is executed, and a function that disables the skip function of the ADDS A, #n4 instruction is effected. Therefore, these instructions can be used in combination for base number adjustment (refer **11.1 Unique Instructions**).

ADDC XA, rp'

Function: XA, CY ← XA + rp' + CY

Adds the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') to the contents of register pair XA in binary, including the carry. If a carry occurs as a result, the carry flag is set; if not, the carry flag is reset.

ADDC rp'1, XA

Function: rp'1, CY ← rp'1 + XA + CY

Adds the contents of register pair XA to the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC') in binary, including the carry flag. If a carry occurs as a result, the carry flag is set; if not, the carry flag is reset.

SUBS A, @HL

Function: A ← A – (HL); Skip if borrow.

Subtracts the contents of the data memory addressed by register pair HL from the contents of the A register, and sets the result to the A register. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

SUBS XA, rp'

Function: XA ← XA – rp'; Skip if borrow.

Subtracts the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') from the contents of register pair XA, and sets the result to register pair XA. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

Application example

To compare specified data memory contents with the contents of a register pair

```
MOV    XA, mem
SUBS   XA, rp'
               ; (mem) ≥ rp'
               ; (mem) < rp'
```

SUBS rp'1, XA

Function: rp' ← rp'1 + XA; Skip if borrow.

Subtracts the contents of register pair XA from register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to specified register pair rp'1. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

SUBC A, @HL

Function: A, CY ← A – (HL) – CY

Subtracts the contents of the data memory addressed by register pair HL to the contents from the A register, including the carry flag, and sets the result to the A register. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

If an ADDS A, #n4 instruction is placed following this instruction, and if a borrow does not occur as a result of executing this instruction, the ADDS A, #n4 instruction is skipped. If a borrow occurs, the ADDS A, #n4 instruction is executed, and a function that disables the skip function of the ADDS A, #n4 instruction is effected. Therefore, these instructions can be used in combination for base number adjustment (refer to **11.1 Unique Instructions**).

SUBC XA, rp'

Function: XA, CY ← XA – rp' – CY

Subtracts the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') from the contents of register pair XA, including the carry, and sets the result to register pair XA. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

SUBC rp'1, XA

Function: rp'1, CY ← rp'1 – XA – CY

Subtracts the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), including the carry flag, and sets the result to specified register pair rp'1. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

AND A, #n4

Function: A ← A ∧ n4 n4 = I_{3-0}: 0-FH

ANDs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To clear the high-order 2 bits of the accumulator to 0

```
AND A, #0011B
```

AND A, @HL

Function: A ← A ∧ (HL)

ANDs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

AND XA, rp'

Function: XA ← XA ∧ rp'

ANDs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

AND rp'1, XA

Function: rp' 1 ← rp'1 ∧ XA

ANDs the contents of register pair XA with register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair.

OR A, #n4

Function: A ← A ∨ n4 n4 = I_{3-0}: 0-FH

ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To set the low-order 3 bits of the accumulator to 1

```
OR A, #0111B
```

OR A, @HL

Function: A ← A ∨ (HL)

ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

OR XA, rp'

Function: XA ← XA ∨ rp'

ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

OR rp'1, XA

Function: rp' 1 ← rp'1 ∨ XA

ORs the contents of register pair XA with register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair rp'1.

XOR A, #n4

Function: A ← A ▽ n4 n4 = I_{3-0}: 0-FH

Exclusive-ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To invert the high-order 4 bits of the accumulator

```
XOR A, #1000B
```

XOR A, @HL

Function: A ← A ▽ (HL)

Exclusive-ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

XOR XA, rp'

Function: XA ← XA ▽ rp'

Exclusive-ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

XOR rp'1, XA

Function: rp' 1 ← rp'1 ▽ XA

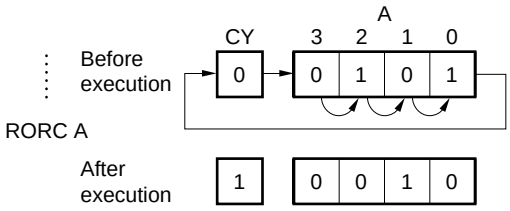
Exclusive-ORs the contents of register pair XA with register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair rp'1.

11.4.5 Accumulator manipulation instructions

RORC A

Function: CY ← A_0, A_{n-1} ← A_n, A_3 ← CY (n = 1-3)

Rotates the contents of the A register (4-bit accumulator) 1 bit to the right with the carry flag.



NOT A

Function: A ← $\overline{A}$

Takes 1's complement of the A register (4-bit accumulator) (inverts the bits of the accumulator).

11.4.6 Increment/decrement instruction

INCS reg

Function: reg ← reg + 1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L, D, E, B, or C). If reg = 0 as a result, the next instruction is skipped.

INCS rp1

Function: rp1 ← rp1+1; Skip if rp1 = 00H

Increments the contents of register pair rp1 (HL, DE, or BC). If rp1 = 00H as a result, the next instruction is skipped.

INCS @HL

Function: (HL) ← (HL) + 1; Skip if (HL) = 0

Increments the contents of the data memory addressed by pair register HL. If the contents of the data memory become 0 as a result, the next instruction is skipped.

INCS mem

Function: (mem) ← (mem) + 1; Skip if (mem) = 0, mem = D_{7-0}: 00H-FFH

Increments the contents of the data memory addressed by 8-bit immediate data mem. If the contents of the data memory become 0 as a result, the next instruction is skipped.

DECS reg

Function: reg ← reg – 1; Skip if reg = FH

Decrements the contents of register reg (X, A, H, L, D, E, B, or C). If reg = FH as a result, the next instruction is skipped.

DECS rp'

Function: rp' ← rp' – 1; Skip if rp' = 00H

Decrements the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE' or BC'). If rp' = FFH as a result, the next instruction is skipped.

11.4.7 Compare instructions

SKE reg, #n4

Function: Skip if reg = n4 n4 = I_{3-0}: 0-FH

Skips the next instruction if the contents of register reg (X, A, H, L, D, E, B, or C) are equal to 4-bit immediate data n4.

SKE @HL, #n4

Function: Skip if (HL) = n4 n4 = I_{3-0}: 0-FH

Skips the next 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 next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL.

SKE XA, @HL

Function: Skip if A = (HL) and X = (HL + 1)

Skips the next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL and if the contents of the X register are equal to the contents of the next memory address.

However, if the contents of the L register are an odd number, the address is determined as if the least significant bit had been zero.

SKE A, reg

Function: Skip if A = reg

Skips the next one instruction if the contents of the A register are equal to register reg (X, A, H, L, D, E, B, or C).

SKE XA, rp'

Function: Skip if XA = rp'

Skips the next one instruction if the contents of register pair XA are equal to the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC').

11.4.8 Carry flag manipulation instructions

SET1 CY

Function: CY ← 1

Sets the carry flag.

CLR1 CY

Function: CY ← 0

Clears the carry flag.

SKT CY

Function: Skip if CY = 1

Skips the next one instruction if the carry flag is 1.

NOT1 CY

Function: CY ← $\overline{CY}$

Inverts the carry flag. Therefore, sets the carry flag to 1 if it is 0, and clears the flag to 0 if it is 1.

11.4.9 Memory bit manipulation instructions

SET1 mem.bit

Function: (mem. bit) ← 1 mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Sets the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

SET1 fmem. bit

SET1 pmem. @L

SET1 @H+mem. bit

Function: (bit specified by operand) ← 1

Sets the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @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 specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

CLR1 fmem. bit

CLR1 pmem. @L

CLR1 @H+mem. bit

Function: (bit specified by operand) ← 0

Clears the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @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 next instruction if the bit specified by 2-bit immediate data bit at the address specified 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 next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @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 next instruction if the bit specified by 2-bit immediate data bit at the address specified 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 next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) is 0.

SKTCLR fmem. bit

SKTCLR pmem. @L

SKTCLR @H+mem. bit

Function: Skip if (bit specified by operand) = 1; then clear

Skips the next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) is 1, and then clears the bit to “0”.

AND1 CY, fmem. bit

AND1 CY, pmem. @L

AND1 CY, @H+mem. bit

Function: CY ← CY ∧ (bit specified by operand)

ANDs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

OR1 CY, fmem. bit

OR1 CY, pmem. @L

OR1 CY, @H+mem. bit

Function: CY ← CY ∨ (bit specified by operand)

ORs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

XOR1 CY, fmem. bit

XOR1 CY, pmem. @L

XOR1 CY, @H+mem. bit

Function: CY ← CY ⩛ (bit specified by operand)

Exclusive-ORs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

11.4.10 Branch instructions

BR addr

Function: μPD754304 PC_{12-0} ←addr
addr = 0000H-1FFFH

Branches to an address specified by immediate data addr.

This instruction is an assembler pseudoinstruction and is replaced by the assembler at assembly time with the optimum instruction from the BR !addr, BRCB !caddr, and BR $addr instructions.

II BR addr1

Function: μPD754304 PC_{12-0} ←addr1
addr1 = 0000H-1FFFH

Branches to an address specified by immediate data addr1.

This instruction is an assembler pseudoinstruction and is replaced by the assembler at assembly time with the optimum instruction from the BRA !addr1, BR !addr, BRCB !caddr, and BR $addr1 instructions.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

(II) **BRA !addr1**

Function: μPD754304 $PC_{11-0} \leftarrow$ addr1

( ) **BR !addr**

Function: μPD754304 $PC_{11-0} \leftarrow$addr
addr = 0000H-1FFFH

Transfers immediate data addr to the program counter (PC) and branches to the address specified by the PC.

(I) **BR $addr**

Function: μPD754304 $PC_{11-0} \leftarrow$ addr
addr = (PC–15) to (PC–1), (PC+2) to (PC+16)

This is a relative branch instruction that has a branch range of (–15 to –1) and (+2 to +16) from the current address. It is not affected by a page boundary or block boundary.

(II) **BR $addr1**

Function: μPD754304 $PC_{11-0} \leftarrow$addr1
addr = (PC–15) to (PC–1), (PC+2) to (PC+16)

This is a relative branch instruction that has a branch range of (–15 to –1) and (+2 to +16) from the current address. It is not affected by a page boundary or block boundary.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

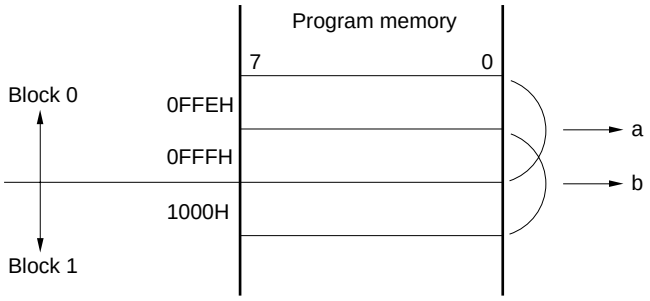
BRCB !caddr

Function: μPD754304 $PC_{11-0} \leftarrow$ caddr
caddr = A_{11-0}: 000H - FFFH

Branches to an address specified by the low-order 12 bits of the program counter (PC_{11-0}) replaced with 12-bit immediate data caddr (A_{11-0}).

The μPD754302 and μPD75304 have a 11-bit and 12-bit program counter, respectively. The BRCB !caddr instruction can branch execution to an entire space. However, the μPD75P4308 cannot change the PC_{12} and the instruction branches exection within the block where the instruction exists.

Caution The BRCB !caddr instruction usually branches execution within the block where the instruction exists. If the first byte of this instruction is at address 0FFEH or 0FFFH, however, execution does not branch to block 0 but to block 1.



If the BRCB !caddr instruction is at position a or b in the figure above, execution branches to block 1, not block 0.

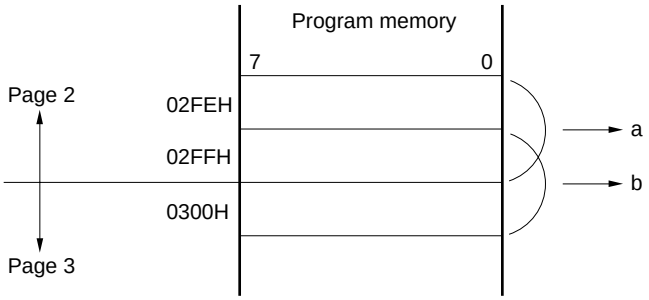
Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

BR PCDE

Function: $PC_{7-4} \leftarrow D$, $PC_{3-0} \leftarrow E$

Branches to the address specified by the low-order 8 bits of the program counter (PC_{7-0}) replaced with the contents of register pair DE. The higher bits of the program counter are not affected.

Caution The BR PCDE instruction usually branches execution within the page where the instruction exists. If the first byte of the op code is at address ××FEH or ××FFH, however, execution does not branch in that page, but to the next page.



For example, if the BR PCDE instruction is at position a or b in the above figure, execution branches to the low-order 8-bit address specified by the contents of register pair DE in page 3, not in page 2.

BR PCXA

Function: $PC_{7-4} \leftarrow X$, $PC_{3-0} \leftarrow A$

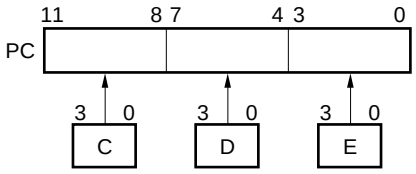
Branches to the address specified by the low-order 8 bits of the program counter (PC_{7-0}) replaced with the contents of register pair XA. The higher bits of the program counter are not affected.

Caution This instruction branches execution to the next page, not to the same page, if the first byte of the op code is at address ××FEH or ××FFH, in the same manner as the BR PCDE instruction.

BR BCDE

Function: μPD754304 $PC_{11-0} \leftarrow$ BCDE

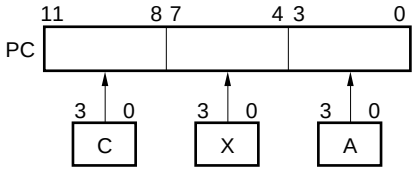
Branches to the address specified by the contents of the program counter replaced with the contents of registers C, D, and E.



BR BCXA

Function: $PC_{11-0} \leftarrow$ BCXA

Branches to the address specified by the contents of the program counter replaced with the contents of registers B, C, X, and A.



TBR addr

Function:

This is an assembler pseudoinstruction for table definition by the GETI instruction. It is used to replace a 3-byte BR !addr instruction with a 1-byte GETI instruction. Code the 12-bit address data as addr. For details, refer to the **RA75X Assembler Package User's Manual – Language (EEU-1363)**.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

11.4.11 Subroutine/stack control instructions

(II) CALLA !addr1

Function: μ**PD754304**

(SP–2) ← ×, ×, MBE, RBE, (SP–3) ← PC$_{7-4}$
(SP–4) ← PC$_{3-0}$, (SP–5) ← 0,
(SP–6) ← PC$_{11-8}$
PC$_{11-0}$ ← addr1, SP ← SP–6

(I/II) CALL !addr

Function: μ**PD754304**

[Mk I mode]
(SP–1) ← PC$_{7-4}$, (SP–2) ← PC$_{3-0}$
(SP–3) ← MBE, RBE, 0, 0
(SP–4) ← PC$_{11-8}$, PC$_{11-0}$ ← addr, SP ← SP–8

addr = 0000H–1FFFH

[Mk II mode]
(SP–2) ← ×, ×, MBE, RBE
(SP–3) ← PC$_{7-4}$, (SP–4) ← PC$_{3-0}$
(SP–5) ← 0, 0, 0, 0, (SP–6) ← PC$_{11-8}$
PC$_{12-0}$ ← addr, SP ← SP–6

addr = 0000H-1FFFH

Saves the contents of the program counter (return address), MBE, and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by 14-bit immediate data addr.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

I/II CALL !faddr

Function: μ**PD754304**

[Mk I mode]

(SP–1) ← PC$_{7-4}$, (SP–2) ← PC$_{3-0}$

(SP–3) ← MBE, RBE, 0, 0

(SP–4) ← PC$_{11-8}$, SP ← SP–4

PC$_{12-0}$ ← 00+faddr

faddr = 0000H–07FFH

[Mk II mode]

(SP–2) ← ×, ×, MBE, RBE

(SP–3) ← PC$_{7-4}$, (SP–4) ← PC$_{3-0}$

(SP–5) ← 0, 0, 0, 0, (SP–6) ← PC$_{11-8}$

SP ← SP–6

PC$_{11-0}$ ← 00+faddr

faddr = 0000H–07FFH

Saves the contents of the program counter (return address), MBE, and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by 11-bit immediate data faddr. The address range from which a subroutine can be called is limited to 0000H to 07FFH (0 to 2047).

TCALL !addr

Function

This is an assembler pseudoinstruction for table definition by the GETI instruction. It is used to replace a 3-byte CALL !addr instruction with a 1-byte GETI instruction. Code 12-bit address data as addr. For details, refer to the **RA75X Assembler Package User's Manual – Language (EEU-1363)**.

Remark TThe function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

(I/II) RET

Function: μPD754304

[Mk I mode] **$PC_{11-8} \leftarrow$ (SP),**
MBE, RBE, 0, 0 $\leftarrow$ (SP+1)
$PC_{3-0} \leftarrow$ (SP+2),
$PC_{7-4} \leftarrow$ (SP+3), SP $\leftarrow$ SP+4

[Mk II mode] **$PC_{11-8} \leftarrow$ (SP), $\times$, $\times$, $\times$, $\times \leftarrow$ (SP+1)**
$PC_{3-0} \leftarrow$ (SP+2), $PC_{7-4} \leftarrow$ (SP+3)
$\times$, $\times$, MBE, RBE $\leftarrow$ (SP+4)
SP $\leftarrow$ SP+6

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), and then increments the contents of the SP.

Caution None of the flags of the program status word (PSW), except MBE and RBE, are restored.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

(I/II) RETS

Function: μPD754304

[Mk I mode] **$PC_{11-8} \leftarrow$ (SP),**
MBE, RBE, 0, 0 $\leftarrow$ (SP+1)
$PC_{3-0} \leftarrow$ (SP+2), $PC_{7-4} \leftarrow$ (SP+3), SP $\leftarrow$ SP+4
Then skip unconditionally

[Mk II mode] **$PC_{11-8} \leftarrow$ (SP), $\times$, $\times$, $\times$, $\times \leftarrow$ (SP+1)**
$PC_{3-0} \leftarrow$ (SP+2), $PC_{7-4} \leftarrow$ (SP+3)
$\times$, $\times$, MBE, RBE $\leftarrow$ (SP+4),
SP $\leftarrow$ SP+6
Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), increments the contents of the SP, and then skips unconditionally.

Caution None of the flags of the program status word (PSW), except MBE and RBE, are restored.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

RETI

(I/II)

Function: μPD754304

[Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, RBE, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2),$
$PC_{7-4} \leftarrow (SP+3)$
$PSW_L \leftarrow (SP+4), PSW_H \leftarrow (SP+5)$
$SP \leftarrow SP+6$

[Mk II mode] $PC_{11-8} \leftarrow (SP), \times, \times, \times, \times \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2),$
$PC_{7-4} \leftarrow (SP+3)$
$PSW_L \leftarrow (SP+4), PSW_H \leftarrow (SP+5)$
$SP \leftarrow SP+6$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and program status word (PSW), and then increments the contents of the SP.

This instruction is used to return execution from an interrupt processing routine.

PUSH rp

Function: $(SP-1) \leftarrow rp_H, (SP-2) \leftarrow rp_L, SP \leftarrow SP-2$

Saves the contents of register pair rp (XA, HL, DE, or BC) to the data memory (stack) addressed by the stack pointer (SP), and then decrements the contents of the SP.

The high-order 4 bits of the register pair (rp_H: X, H, D, or B) are saved to the stack addressed by (SP-1), and the low-order 4 bits (rp_L: A, L, E, or C) are saved to the stack addressed by (SP-2).

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

PUSH BS

Function: (SP–1) ← MBS, (SP–2) ← RBS, SP ← SP–2

Saves the contents of the memory bank select register (MBS) and register bank select register (RBS) to the data memory (stack) addressed by the stack pointer (SP), and then decrements the contents of the SP.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

POP rp

Function: rp_L ← (SP), rp_H ← (SP+1), SP ← SP+2

Restores the contents of the data memory addressed by the stack pointer (SP) to register pair rp (XA, HL, DE, or BC), and then increments the contents of the stack pointer.

The contents of (SP) are restored to the low-order 4 bits of the register pair (rp_L: A, L, E, or C), and the contents of (SP+1) are restored to the high-order 4 bits (rp_H: X, H, D, or B).

POP BS

Function: RBS ← (SP), MBS ← (SP+1), SP ← SP+2

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the register bank select register (RBS) and memory bank select register (MBS), and then increments the contents of the SP.

11.4.12 Interrupt control instructions

EI

Function: IME (IPS.3) ← 1

Sets the interrupt mask enable flag (bit 3 of the interrupt priority select register) to “1” to enable interrupts. Acknowledging an interrupt is controlled by an interrupt enable flag corresponding to the interrupt.

EI IE×××

Function: IE××× ← 1 ××× = N_5, $N_{2\text{-}0}$

Sets a specified interrupt enable flag (IE×××) to “1” to enable acknowledging the corresponding interrupt (××× = BT, CSI, T0, T1, 0, 1, 2, or 4).

DI

Function: IME (IPS.3) ← 0

Resets the interrupt mask enable flag (bit 3 of the interrupt priority select register) to “0” to disable all interrupts, regardless of the contents of the respective interrupt enable flags.

DI IE×××

Function: IE××× ← 1 ××× = N_5, $N_{2\text{-}0}$

Resets a specified interrupt enable flag (IE×××) to “0” to disable acknowledging the corresponding interrupt (××× = BT, CSI, T0, T1, 0, 1, 2, or 4).

11.4.13 Input/output instructions

IN A, PORTn

Function: A ← PORTn n = N_{3-0}: 0-3, 5-8

Transfers the contents of a port specified by PORTn (n = 0-3, 5-8) to the A register.

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 0 to 3 and 5 to 8 can be specified to n.
The data of the output latch is loaded to the A register in the output mode, and the data of the port pins are loaded to the register in the input mode by specifying Input/output mode.

IN XA, PORTn

Function: A ← $PORT_n$, X ← $PORT_{n+1}$ n = N_{3-0}: 6

Transfers the contents of the port specified by $PORT_n$ (n = 6) to the A register, and transfers the contents of the next port to the X register.

Caution Only 6 can be specified as n. When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15).
The data of the output latch is loaded to the A and X registers in the output mode, and the data of the port pins are loaded to the registers in the input mode by specifying Input/output mode.

OUT PORTn, A

Function: PORTn ← A n = N_{3-0}: 2, 3, 5-8

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 2, 3, 5-8).

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 2, 3 and 5 to 8 can be specified as n.

OUT PORTn, XA

Function: $PORT_n$, ← A, $PORT_{n+1}$ ← X n = N_{3-0}: 6

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 6), and the contents of the X register to the output latch of the next port.

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 6 can be specified as n.

11.4.14 CPU control instructions

HALT

Function: PCC.2 ← 1

Sets the HALT mode (this instruction sets bit 2 of the processor clock control register).

Caution Make sure that a NOP instruction follows the HALT instruction.

STOP

Function: PCC.3 ← 1

Sets the STOP mode (this instruction sets bit 3 of the processor clock control register).

Caution Make sure that a NOP instruction follows the STOP instruction.

NOP

Function: Does nothing but consumes 1 machine cycle.

11.4.15 Special instructions

SEL RBn

Function: RBS ← n n = N_{1-0}: 0-3

Sets 2-bit immediate data n to the register bank select register (RBS).

SEL MBn

Function: MBS ← n n = N_{3-0}: 0, 1, 15

Transfers 4-bit immediate data n to the memory bank select register (MBS).
Only 0 and 15 can be specified to n.

(I/II) GETI taddr

Function: **μPD754304**

taddr = T_{5-0}, 0: 20H-7FH

[Mk I mode]

- **When a table defined by the TBR instruction is referenced**

  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$

- **When a table defined by the TCALL instruction is referenced**

  $(SP-1) \leftarrow PC_{7-4}$, $(SP-2) \leftarrow PC_{3-0}$

  $(SP-3) \leftarrow$ MBE, RBE, 0, 0

  $(SP-4) \leftarrow PC_{11-8}$

  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$

  $SP \leftarrow SP-4$

- **When a table defined by an instruction other than TBR or TCALL is referenced**

  Executes instruction with (taddr) (taddr+1) as op code

[Mk II mode]

- **When a table defined by the TBR instruction is referenced**

  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$

- **When a table defined by the TCALL instruction is referenced**

  $(SP-2) \leftarrow \times, \times$, MBE, RBE

  $(SP-3) \leftarrow PC_{7-4}$, $(SP-4) \leftarrow PC_{3-0}$

  $(SP-5) \leftarrow 0, 0, 0, 0$, $(SP-6) \leftarrow PC_{11-8}$

  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$

  $SP \leftarrow SP-6$

- **When a table defined by an instruction other than TBR and TCALL is referenced**

  Executes instruction with (taddr) (taddr+1) as op code

References the 2-byte data at the program memory address specified by (taddr), (taddr+1) and executes it as an instruction.

The area of the reference table consists of addresses 0020H through 007FH. Data must be written to this area in advance. When the data to be written is 1-byte or 2-byte instructions, code the mnemonics directly.

When a 3-byte call instruction or 3-byte branch instruction is used, data is written by using an assembler pseudoinstruction (TCALL or TBR).

Only an even address can be specified by taddr.

Remark The function described here applies to the μPD754304 that has a 12-bit program counter. Note that the program counter of the μPD754302 is 11 bits wide and that of the μPD75P4308 is 13 bits wide.

Caution Only the 2-machine cycle instructions can be placed in the reference table as a 2-byte instructions (except the BRCB and CALLF instructions). Two 1-byte instructions can be used only in the following combinations:

Instruction of 1st byte	Instruction of 2nd byte

MOV A, @HL
MOV @HL, A
XCH A, @HL
	INCS L
	DECS L
	INCS H
	DECS H
	INCS HL

MOV A, @DE XCH A, @DE	INCS E DECS E INCS D DECS D INCS DE
MOV A, @DL XCH A, @DL	INCS L DECS L INCS D DECS D

The contents of the PC are not incremented while the GETI instruction is executed. Therefore, after the referenced instruction has been executed, processing continues from the address following that of the GETI instruction.

If the instruction preceding the GETI instruction has a skip function, the GETI instruction is skipped in the same manner as other 1-byte instructions. If the instruction referenced by the GETI instruction has a skip function, the instruction that follows the GETI instruction is skipped.

If an instruction having a string effect is referenced by the GETI instruction, it is executed as follows:

- If the instruction preceding the GETI instruction has the string effect in the same group as the referenced instruction, the string effect is lost and the referenced instruction is not skipped when GETI is executed.
- If the instruction next to GETI has the string effect in the same group as the referenced instruction, the string effect by the referenced instruction is valid, and the instruction following that instruction is skipped.

Application example

```
( MOV   HL, #00H )
( MOV   XA, #FFH )
( CALL  SUB1     )   Replaced by GETI
( BR    SUB2     )
            ORG     20H
HL00 :      MOV     HL, #00H
XAFF :      MOV     XA, #FFH
CSUB1 :     TCALL   SUB1
BSUB2 :     TBR     SUB2
              :
              :
            GETI    HL00        ; MOV HL, #00H
              :
              :
            GETI    BSUB2       ; BR SUB2
              :
              :
            GETI    CSUB1       ; CALL SUB1
              :
              :
            GETI    XAFF        ; MOV XA, #FFH
```

APPENDIX A μPD750004, μPD754304, AND μPD75P4308 FUNCTION LIST

(1/2)

Item		μPD750004	μPD754304	μPD75P4308
Program memory		Mask ROM 0000H-0FFFH (4096 × 8 bits)	Mask ROM 0000H-0FFFH (4096 × 8 bits)	One-time PROM 0000H-1FFFH (8192 × 8 bits)
Data memory		000H-1FFH (512 × 4 bits)	000H-0FFH (256 × 4 bits)	
CPU		75XL CPU		
Instruction execution time	When selecting main system clock	• 0.67 μs, 1.33 μs 2.67 μs, 10.7 μs (6.0 MHz) • 0.95 μs, 1.91 μs, 3.81 μs, 15.3 μs (4.19 MHz)		
	When selecting the subsystem clock	• 122 μs (32.768 kHz)	No subsystem clock	
I/O ports	CMOS input	8 (software selectable pull-up resistors: 7)		
	CMOS I/O	18 (software selectable pull-up resistors)		
	N-channel, open drain I/O (max. 13 V)	8 (mask-option selectable pull-up resistors)	4 (mask-option selectable pull-up resistors)	4 (no mask option)
	Total	34	30 (no port 4 pins)	
Timer		4 channels • Basic interval timer/ watchdog timer • 8-bit timer/event counter • 8-bit timer • watch timer	3 channels • Basic interval timer/watchdog timer • 8-bit timer/event counter 0 ($fx/2^2$ added) • 8-bit timer/event counter 1 (TI1, $fx/2^2$ added) (Can be used as the 16-bit timer/event counter)	
Clock output (PCL)		• Φ, 524 kHz, 262 kHz, 65.5 kHz (main system clock: 4.19 MHz) • Φ, 750 kHz, 375 kHz, 93.8 kHz (main system clock: 6.0 MHz)		
Buzzer output (BUZ)		Yes	No	
Serial interface		3 modes possible • 3-wire serial I/O mode – Can switch between the MSB or the LSB first • 2-wire serial I/O mode • SBI mode	2 modes possible • 3-wire serial I/O mode – Can switch between the MSB or the LSB first • 2-wire serial I/O mode	

★

(2/2)

Item	μPD750004	μPD754304	μPD75P4308
Watch mode register (WM)	Yes	No	
System clock control register (SCC)			
Suboscillator control register (SOS)			
MBS register	MB0, 1	Only MB0	
Stack area (SBS1 and SBS0)			
TM0 and TM1 registers	Bits 0, 1, and 7 are fixed to 0.	–	
Vectored interrupts	External: 3, Internal: 4		
Test inputs	External: 1, Internal: 1	External: 1	
Test enable flag (IEW)	Yes	No	
Test request flag (IRQW)			
Power supply voltage	V_{DD} = 2.2 to 5.5 V	V_{DD} = 1.8 to 5.5 V	
Operating ambient temperature	T_A = –40 to +85 ˚C		
Package	• 42-pin plastic shrink DIP (600 mil) • 44-pin plastic QFP (10 × 10 mm)	• 36-pin plastic shrink SOP (300 mil, 0.8 mm pitch)	

APPENDIX B DEVELOPMENT TOOLS

The following development tools are provided for system development using the μPD754304.

In 75XL series, the relocatable assembler which is common to the series is used in combination with the device file of each product.

Language processor

RA75X relocatable assembler	Host machine	OS	Distribution media	Part number (product name)
	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™ and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13RA75X
			5-inch 2HC	μS7B10RA75X

★

Device file	Host machine	OS	Distribution media	Part number (product name)
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2 **Note**)	3.5-inch 2HD	μS5A13DF754304
			5-inch 2HD	μS5A10DF754304
	IBM PC/AT and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13DF754304
			5-inch 2HC	μS7B10DF754304

★

Note Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operations of the assembler and device file are guaranteed only on the above host machines and OSs.

PROM write tools

Hardware	PG-1500	PG-1500 is a PROM programmer which enables you to program single-chip microcontrollers including PROM by stand-alone or host machine operation by connecting an attached board and optional programmer adapter to PG-1500. It also enables you to program typical PROM devices of 256K bits to 4M bits.			
	PA-75P4308GS	PROM programmer adapter for the μPD75P4308GS. Connect the programmer adapter to PG-1500 for use.			
Software	PG-1500 controller	PG-1500 and a host machine are connected by serial and parallel interfaces and PG-1500 is controlled on the host machine.			
		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	μS5A13PG1500
★				5-inch 2HD	μS5A10PG1500
		IBM PC/AT and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13PG1500
				5-inch 2HC	μS7B10PG1500

Note Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operation of the PG-1500 controller is guaranteed only on the above host machines and OSs.

Debugging tool

The in-circuit emulators (IE-75000-R and IE-75001-R) are available as the program debugging tool for the μPD754304.

The system configurations are described as follows.

Hardware	IE-75000-R Note 1	In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X series and 75XL series. When developing a μPD754304 subseries, the emulation board IE-75300-R-EM and emulation probe EP-754304GS-R which are sold separately must be used with the IE-75000-R. By connecting with the host machine and the PROM programmer, efficient debugging can be made. It contains the emulation board IE-75000-R-EM which is connected.			
	IE-75001-R	In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X series and 75XL series. When developing a μPD754304 subseries, the emulation board IE-75300-R-EM and emulation probe EP-754304GS-R which are sold separately must be used with the IE-75001-R. By connecting with the host machine and the PROM programmer, efficient debugging can be made.			
	IE-75300-R-EM	Emulation board for evaluating the application systems that use a μPD754304 subseries. It must be used with the IE-75000-R or IE-75001-R.			
	EP-754304GS-R EV-9500GS-36	Emulation probe for the μPD754304GS. It must be connected to the IE-75000-R (or IE-75001-R) and IE-75300-R-EM. It is supplied with the flexible board EV-9500GS-36 which facilitates connection to a target system.			
Software	IE control program	Connects the IE-75000-R or IE-75001-R to a host machine via RS-232-C and Centronix interface and controls the above hardware on a host machine.			
		Host machine	OS	Distribution media	Part No. (product name)
		PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2 Note 2)	3.5-inch 2HD	μS5A13IE75X
				5-inch 2HD	μS5A10IE75X
		IBM PC/AT and its compatible machine	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13IE75X
				5-inch 2HC	μS7B10IE75X

★

Notes 1. Maintenance parts

2. Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operation of the IE control program is guaranteed only on the above host machines and OSs.

OS for IBM PC

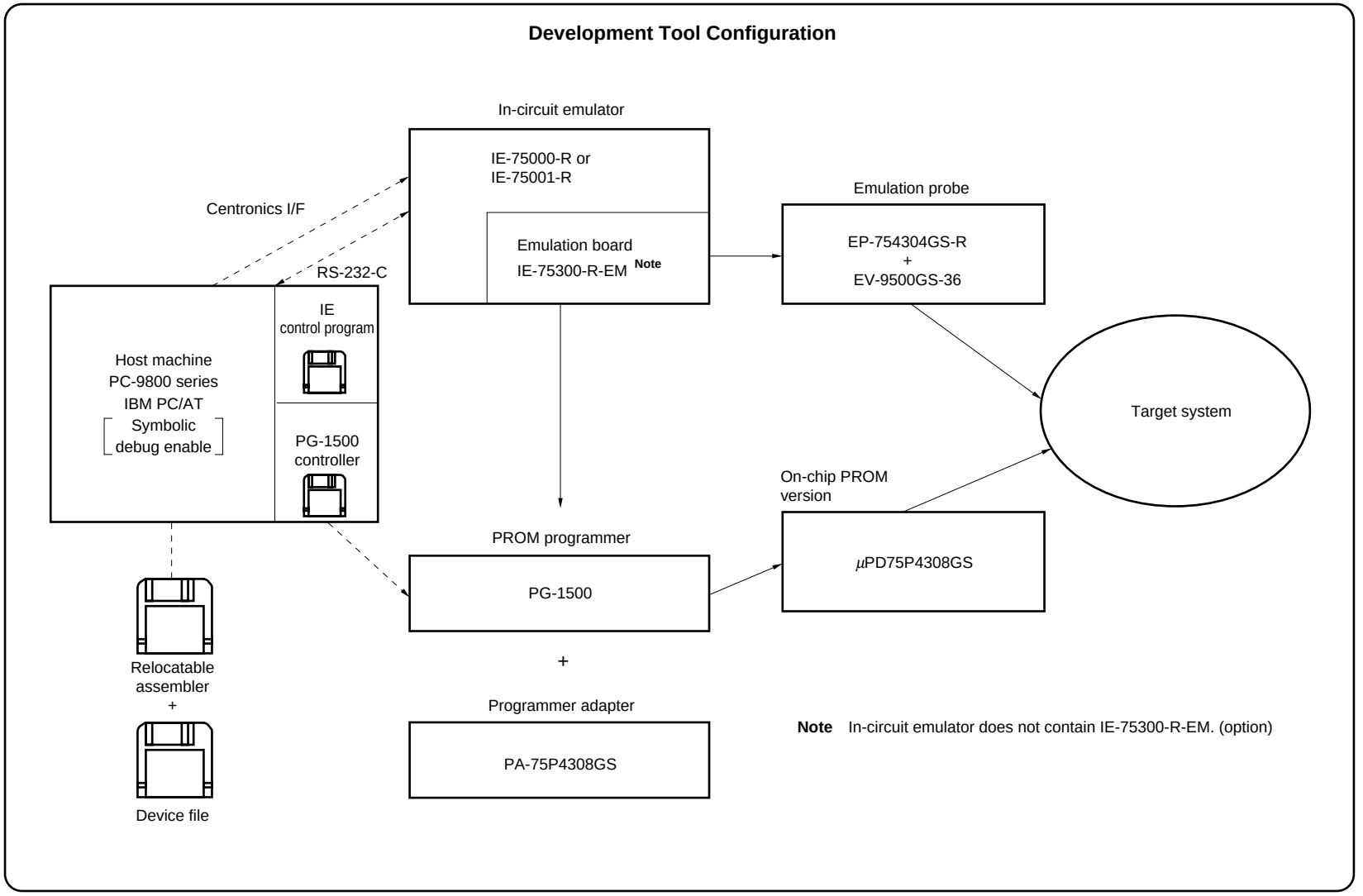
The following IBM PC OSs are supported.

OS	Version
★ PC DOS	Ver. 5.02 to Ver. 6.3 J6.1/V **Note** to J6.3/V **Note**
★ MS-DOS	Ver. 5.00 to Ver. 6.22 5.0/V **Note** to 6.2/V **Note**
IBM DOS™	J5.02/V **Note**

Note Only English version is supported.

Caution Ver. 5.0 and later have the task swap function, but it cannot be used for this software.

Development Tool Configuration



Note In-circuit emulator does not contain IE-75300-R-EM. (option)

[MEMO]

APPENDIX C ORDERING MASK ROMS

After your program has been developed, place an order for a mask ROM using the following procedure:

<1> Reservation for ordering mask ROM

Inform NEC of your schedule to place an order for the mask ROM (NEC's response may be delayed if it is not informed in advance).

<2> Preparation of ordering media

Use a UV-EPROM or 3.5-inch (or 5-inch) IBM-format floppy disk as the mask ROM ordering media. Prepare three UV-EPROMs having the same contents (write down the mask option data on the mask option information sheet).

<3> Preparation of necessary documents

Fill out the following documents when ordering the mask ROM:

A. Mask ROM Ordering Sheet
B. Mask ROM Ordering Check Sheet
C. Mask Option Information Sheet

<4> Ordering

Submit the media prepared in <2> and documents prepared in <3> to NEC by the reserved date.

For details, see the **ROM Code Ordering Method (IEM-834)** [Note] for reference.

Note This document number is that of Japanese version.

[MEMO]

APPENDIX D INSTRUCTION INDEX

D.1 Instruction Index (by function)

D.2 Instruction Index (alphabetical order)

APPENDIX E HARDWARE INDEX

[MEMO]

APPENDIX F REVISION HISTORY ★

The table shown below lists the major revised points in each edition of this manual. Note that, in the “Including Chapter” column, the chapter numbers and names of each edition are shown, not necessarily this edition.

Edition	Description	Including Chapter
Second	The development status of the μPD754302, 754304, and 75P4308 is changed from “Under development” to “Development completed.”	Throughout
	The input withstand voltage of the port 5's pins in the open-drain mode is changed from 12 V to 13 V.	
	The COI is added to the serial operation mode register (CSIM).	
	The slave address register (SVA) is added.	
	The address comparator is added to the serial interface block diagram.	**CHAPTER 5 PERIPHERAL HARDWARE FUNCTION**
	The explanation about mask options is added.	**CHAPTER 10 MASK OPTION**
	Modification of the instruction list.	**CHAPTER 11 INSTRUCTION SET**
	The versions of the supported operating systems are upgraded.	**APPENDIX B DEVELOPMENT TOOLS**

[MEMO]

NEC

Facsimile Message

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.

From:

Name

Company

Tel. FAX

Address

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
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	❑	❑	❑	❑