NEC

User's Manual

μPD754144, 754244

4-Bit Single-Chip Microcontrollers

μPD754144
μPD754244

Document No. U10676EJ3V0UM00 (3rd edition)
Date Published November 2002 N CP(K)

© NEC Electronics Corporation 1997
Printed in Japan

Download from Www.Somanuals.com. All Manuals Search And Download.

[MEMO]

Download from Www.Somanuals.com. All Manuals Search And Download.

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.

This product cannot be used for an IC card (SMART CARD).

EEPROM is a trademark of NEC Electronics Corporation.
MS-DOS is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.
IBM DOS, PC/AT, and PC DOS are trademarks of International Business Machines Corporation.

Download from Www.Somanuals.com. All Manuals Search And Download.

These commodities, technology or software, must be exported in accordance
with the export administration regulations of the exporting country.
Diversion contrary to the law of that country is prohibited.

- **The information in this document is current as of July, 2002. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics 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 customer's equipment shall be done under the full responsibility of customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product 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 and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact NEC Electronics sales representative in advance to determine NEC Electronics's willingness to support a given application.

(Note)

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

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11

Download from Www.Somanuals.com. All Manuals Search And Download.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, please contact the NEC Electronics 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 America, Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
  Fax: 091-504 28 60

- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
  Fax: 01-30-67 58 99

- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
  Fax: 02-66 75 42 99

- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
  Fax: 040-244 45 80

- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 80 820
  Fax: 08-63 80 388

- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133
  Fax: 01908-670-290

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 Shanghai, Ltd.
Shanghai, P.R. China
Tel: 021-6841-1138
Fax: 021-6841-1137

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

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 6253-8311
Fax: 6250-3583

J02.11

Download from Www.Somanuals.com. All Manuals Search And Download.

Major Revisions in This Edition

Pages	Description
p.210	Correction of description in figure in **7.9 Application of Interrupt (6) Executing pending interrupt - interrupt occurs during interrupt service (INTBT has higher priority and INTT0 and INTT2 have lower priority)**
p.253	Correction of instruction code of “BR BCDE” in **11.3 Opcode of Each Instruction**
p.296	Deletion of flash-related products in configuration diagram in **APPENDIX A DEVELOPMENT TOOLS**
p.297 in 2nd edition	Deletion of **APPENDIX A LIST OF FUNCTIONS OF μPD754144, 754244, AND 75F4264**

The mark ★ shows major revised points.

Download from Www.Somanuals.com. All Manuals Search And Download.

INTRODUCTION

Readers This manual is intended for user engineers who wish to understand the functions of the μPD754144 and 754244 and design application systems using these microcontrollers.

Purpose This manual is intended to give users an understanding of the hardware functions of the μPD754144 and 754244 described in the **Organization** below.

Organization This manual contains the following information.

- General
- Pin Functions
- Features of Architecture and Memory Map
- Internal CPU Functions
- EEPROM™
- Peripheral Hardware Functions
- Interrupt Functions and Test Functions
- Standby Functions
- Reset Functions
- Mask option
- Instruction Set

How to Read This Manual It is assumed that the readers of this manual have general knowledge of electrical engineering, logic circuits, and microcontrollers.

- **To users who use this manual as a manual for μPD754144 (RC oscillation, f_{CC})**
  → Unless otherwise specified, the μPD754244 (crystal/ceramic oscillation, f_X) is treated as the representative model in this manual. Check the functional differences between the μPD754144 and μPD754244 by referring to **1.3 Differences Between μPD754144 and 754244**, and take the μPD754244 as μPD754144 and f_X as f_{CC}.

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

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

- **To understand the overall functions of the μPD754144 and 754244,**
  → Read this manual in the order of the **CONTENTS**.

Conventions

Data significance:	Higher digits on the left and lower digits on the right	
Active low representations:	$\overline{\times\times\times}$ (overscore over pin and signal name)	
Note:	Footnote for item marked with **Note** in the text.	
Caution:	Information requiring particular attention	
Remark:	Supplementary information	
Numerical representations:	Binary	... ×××× or ××××B
	Decimal	... ××××
	Hexadecimal	... ××××H

Download from Www.Somanuals.com. All Manuals Search And Download.

★ **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 No.
μPD754144, 754244 Data Sheet	U10040E
μPD754144, 754244 User's Manual	This manual
75XL Series Selection Guide	U10453E

Documents related to development tools (software) (user's manuals)

Document Name		Document No.
RA75X Assembler Package	Operation	U12622E
	Language	U12385E
	Structured Assembler Preprocessor	U12598E

Documents related to development tools (hardware) (user's manuals)

Document Name	Document No.
IE-75000-R/IE-75001-R In-Circuit Emulator	EEU-1455
IE-75300-R-EM Emulation Board	U11354E
EP-754144GS-R Emulation Probe	U10695E

Other documents

Document Name	Document No.
SEMICONDUCTOR SELECTION GUIDE - Products & Packages -	X13769E
Semiconductor Device Mounting Technology Manual	C10535E
Quality Grades on NEC Semiconductor Devices	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983E
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E

Caution The related documents listed above are subject to change without notice. Be sure to use the latest version of each document for designing.

Download from Www.Somanuals.com. All Manuals Search And Download.

TABLE OF CONTENTS

Download from Www.Somanuals.com. All Manuals Search And Download.

Download from Www.Somanuals.com. All Manuals Search And Download.

Download from Www.Somanuals.com. All Manuals Search And Download.

Download from Www.Somanuals.com. All Manuals Search And Download.

LIST OF FIGURES (1/3)

Download from Www.Somanuals.com. All Manuals Search And Download.

LIST OF FIGURES (2/3)

Download from Www.Somanuals.com. All Manuals Search And Download.

LIST OF FIGURES (3/3)

Figure No.	Title	Page

Download from Www.Somanuals.com. All Manuals Search And Download.

LIST OF TABLES

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 1 GENERAL

The μPD754144 and 754244 are 4-bit single-chip microcontrollers in the NEC 75XL Series, the successor to the 75X Series that boasts a wealth of variations.

The μPD754144 and 754244 have extended CPU functions compared to the μPD75048, a 75X Series product with on-chip EEPROM, enabling high-speed and low voltage (1.8 V) operation.

This model is available in a small plastic SSOP (7.62 mm (300)).

The features of the μPD754144 are as follows:

- Low-voltage operation: V_{DD} = 1.8 to 6.0 V
  128-bit (16 × 8 bits) EEPROM capable of low voltage (1.8 V) operation on chip
- Variable instruction execution time useful for high-speed operation and power saving
  μPD754144: RC oscillator (resistors and capacitors are externally provided)
  4, 8, 16, 64 μs (at f_{CC} = 1.0 MHz)
  μPD754244: Crystal/ceramic oscillator
  0.95 μs, 1.91 μs, 3.81 μs, 15.3 μs (at f_X = 4.19 MHz)
  0.67 μs, 1.33 μs, 2.67 μs, 10.7 μs (at f_X = 6.0 MHz)
- Four timer channels
- Key return reset function for key-less entry
- Small package (20-pin plastic SSOP (7.62 mm (300))

APPLICATIONS

- Automotive appliances such as keyless entry, small data carriers, etc.

Remark Unless otherwise specified, the μPD754244 (crystal/ceramic oscillation, f_X) is treated as the representative model in this manual. When you use this manual for the μPD754144 (RC oscillation, f_{CC}), take the μPD754244 as the μPD754144 and f_X as f_{CC}.

Download from Www.Somanuals.com. All Manuals Search And Download.

1.1 Functional Outline

<table>
<tr><th colspan="2">Item</th><th colspan="2">μPD754144</th><th>μPD754244</th></tr>
<tr><td colspan="2">Instruction execution time</td><td colspan="2">• 4, 8, 16, 64 μs (at f_{CC} = 1.0 MHz)</td><td>• 0.95, 1.91, 3.81, 15.3 μs (at f_X = 4.19 MHz)
• 0.67, 1.33, 2.67, 10.7 μs (at f_X = 6.00 MHz)</td></tr>
<tr><td rowspan="3">On-chip memory</td><td>Mask ROM</td><td colspan="3">4096 × 8 bits (0000H to 0FFFH)</td></tr>
<tr><td>RAM</td><td colspan="3">128 × 4 bits (000H to 07FH)</td></tr>
<tr><td>EEPROM</td><td colspan="3">16 × 8 bits (400H to 41FH)</td></tr>
<tr><td colspan="2">System clock oscillator</td><td colspan="2">RC oscillator
(External resistor and capacitor)</td><td>Crystal/ceramic oscillator</td></tr>
<tr><td colspan="2">General-purpose registers</td><td colspan="3">• 4-bit operation: 8 × 4 banks
• 8-bit operation: 4 × 4 banks</td></tr>
<tr><td rowspan="3">I/O ports</td><td>CMOS input</td><td>4</td><td colspan="2">Pull-up resistors can be incorporated by mask option</td></tr>
<tr><td>CMOS I/O</td><td>9</td><td colspan="2">On-chip pull-up resistors can be specified by software</td></tr>
<tr><td>Total</td><td colspan="3">13</td></tr>
<tr><td colspan="2">Timers</td><td colspan="3">4 channels
• 8-bit timer counter: 3 channels
(can be used as 16-bit timer counter)
• Basic interval/watchdog timer: 1 channel</td></tr>
<tr><td colspan="2">Programmable threshold port</td><td colspan="3">2 channels</td></tr>
<tr><td colspan="2">Bit sequential buffer</td><td colspan="3">16 bits</td></tr>
<tr><td colspan="2">Vectored interrupt</td><td colspan="3">External: 1, Internal: 5</td></tr>
<tr><td colspan="2">Test input</td><td colspan="3">External: 1 (with key return reset function)</td></tr>
<tr><td colspan="2">Standby function</td><td colspan="3">STOP/HALT mode</td></tr>
<tr><td colspan="2">Operating ambient temperature</td><td colspan="3">T_A = –40 to +85°C</td></tr>
<tr><td colspan="2">Power supply voltage</td><td colspan="3">V_{DD} = 1.8 to 6.0 V</td></tr>
<tr><td colspan="2">Package</td><td colspan="3">• 20-pin plastic SOP (7.62 mm (300))
• 20-pin plastic SSOP (7.62 mm (300))</td></tr>
</table>

Download from Www.Somanuals.com. All Manuals Search And Download.

1.2 Ordering Information

Part Number	Package
μPD754141GS-×××-BA5	20-pin plastic SOP (7.62 mm (300))
μPD754141GS-×××-GJG	20-pin plastic SSOP (7.62 mm (300))
μPD754244GS-×××-BA5	20-pin plastic SOP (7.62 mm (300))
μPD754244GS-×××-GJG	20-pin plastic SSOP (7.62 mm (300))

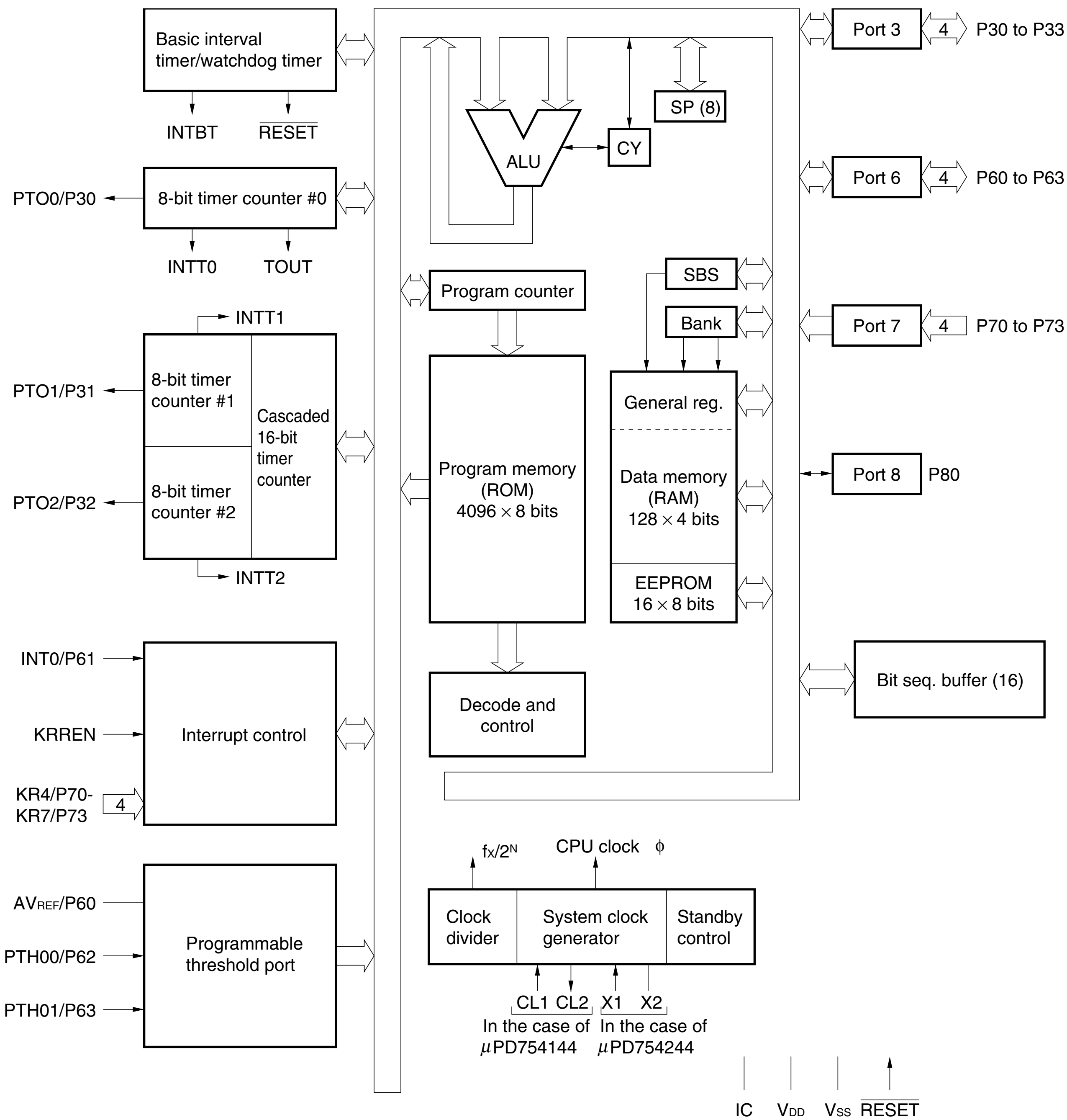
Remark ××× indicates ROM code suffix.

1.3 Differences Between Series Products

Item		μPD754144	μPD754244
Instruction execution time		4, 8, 16, 64 μs (at f_{CC} = 1.0 MHz)	• 0.95, 1.91, 3.81, 15.3 μs (at f_X = 4.19 MHz) • 0.67, 1.33, 2.67, 10.7 μs (at f_X = 6.0 MHz)
System clock oscillator		RC oscillator (resistors and capacitors are externally provided)	Crystal/ceramic oscillator
Startup time after reset		Fixed to 56 μs (at 1 MHz)	Can be selected by mask option from the following two: • $2^{17}/f_X$ (31.3 ms: at 4.19 MHz, 21.8 ms: at 6.0 MHz) • $2^{15}/f_X$ (7.81 ms: at 4.19 MHz, 5.46 ms: at 6.0 MHz)
Standby mode release time		$2^9/f_{CC}$	Can be selected by BTM setting from the following four: $2^{20}/f_X$, $2^{17}/f_X$, $2^{15}/f_X$, $2^{13}/f_X$
Pin connection	pin 2, pin 3	CL1, CL2	X1, X2

Download from Www.Somanuals.com. All Manuals Search And Download.

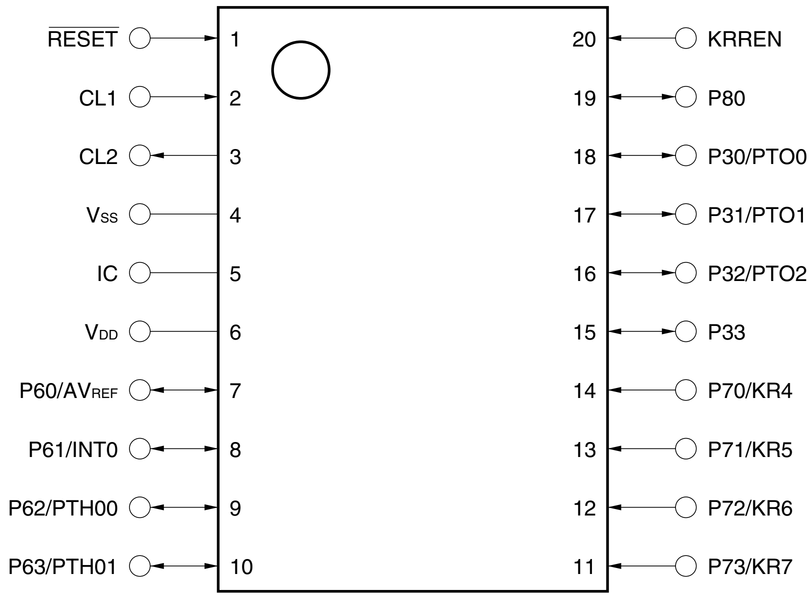
1.4 Block Diagram



Download from Www.Somanuals.com. All Manuals Search And Download.

1.5 Pin Configuration (Top View)

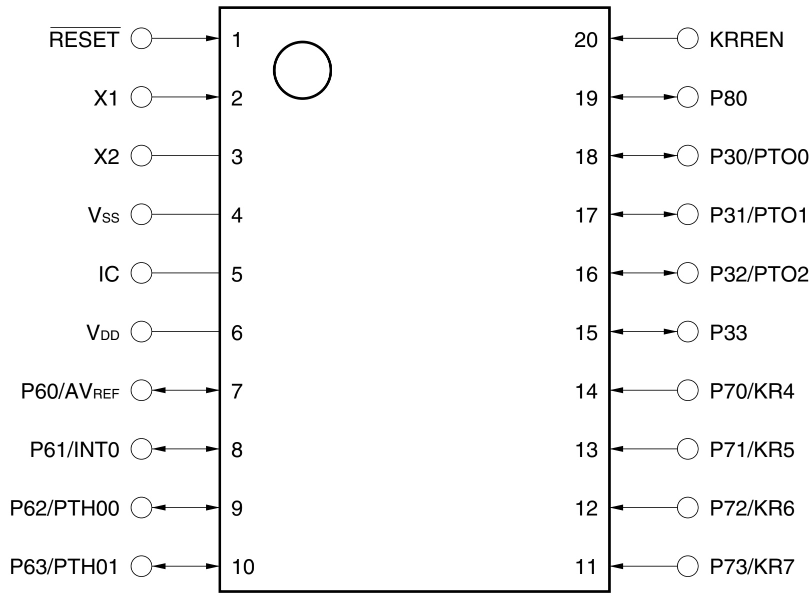
- **Pin configuration of μPD754144**
  - 20-pin plastic SOP (7.62 mm (300))
    μPD754144GS-×××-BA5
  - 20-pin plastic SSOP (7.62 mm (300))
    μPD754144GS-×××-GJG



IC: Internally Connected (Directly connect to V_{DD}.)

Download from Www.Somanuals.com. All Manuals Search And Download.

- **Pin configuration of μPD754244**
  - 20-pin plastic SOP (7.62 mm (300))
    μPD754244GS-×××-BA5
  - 20-pin plastic SSOP (7.62 mm (300))
    μPD754244GS-×××-GJG



IC: Internally Connected (Directly connect to V_{DD}.)

Download from Www.Somanuals.com. All Manuals Search And Download.

Pin Name

P30 to P33:	Port 3
P60 to P63:	Port 6
P70 to P73:	Port 7
P80:	Port 8
KR4 to KR7:	Key return 4 to 7
INT0:	External vectored interrupt 0
PTH00, PTH01:	Programmable threshold port analog input 0, 1
PTO0 to PTO2:	Programmable timer output 0 to 2
KRREN:	Key return reset enable
CL1, CL2:	RC oscillator
X1, X2:	Crystal/ceramic oscillator
IC:	Internally connected
$\overline{\text{RESET}}$:	Reset
AV_{REF}:	Analog reference
V_{SS}:	Ground
V_{DD}:	Positive power supply

Download from Www.Somanuals.com. All Manuals Search And Download.

2.1 Pin Functions of μPD754244

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

Pin Name	I/O	Alternate Function	Function	8-Bit I/O	After Reset	I/O Circuit TypeNote 1
P30	I/O	PTO0	Programmable 4-bit I/O port (Port 3). Input/output can be specified in 1-bit units. On-chip pull-up resistor can be specified by software in 4-bit units.	×	Input	E-B
P31		PTO1				
P32		PTO2				
P33		–				
P60	I/O	AV_{REF}	Programmable 4-bit I/O port (Port 6). Input/output can be specified in 1-bit units. An on-chip pull-up resistor can be specified by software in 4-bit unitsNote 2. A noise eliminator is selectable for P61/ INT0.	×	Input	Ⓕ-A
P61		INT0				
P62		PTH00				
P63		PTH01				
P70	Input	KR4	4-bit input port (Port 7). A pull-up resistor can be incorporated (mask option).	×	Input	Ⓑ-A
P71		KR5				
P72		KR6				
P73		KR7				
P80	I/O	–	1-bit I/O port (PORT8). An on-chip pull-up resistor can be specified by software.	×	Input	Ⓕ-A

Notes 1. Circled characters indicate Schmitt-triggered input.

2. Do not specify connection of an on-chip pull-up resistor when using a programmable threshold port.

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 2-2. Functions of Non-Port Pins

Pin Name	I/O	Alternate Function	Function		After Reset	I/O Circuit Type[Note]
PTO0	Output	P30	Timer counter output pins.		Input	E-B
PTO1		P31				
PTO2		P32				
INT0	Input	P61	Edge-detected vectored interrupt input (edge to be detected is selectable). Noise eliminator is selectable.	Noise eliminator/ asynchronous selectable	Input	Ⓕ-A
KR4 to KR7	Input	P70 to P73	Falling edge-detected testable input.		Input	Ⓑ-A
PTH00	Input	P62	Variable threshold voltage 2-bit analog input.		Input	Ⓕ-A
PTH01		P63				
KRREN	Input	–	Key return reset enable pin. Reset signal is generated at falling edge of KRn when KRREN = high in STOP mode.		Input	Ⓑ
AV_{REF}	Input	P60	Reference voltage input pin.		Input	Ⓕ-A
CL1	Input	–	Provided in μPD754144 only. These pins connect R and C for system clock oscillation. No external clock can be input to these pins.		–	–
CL2	Output					
X1	Input	–	Provided in μPD754244 only. These pins connect crystal/ceramic oscillator for system clock oscillation. When external clock is used, input it to X1 and inverse phase to X2.		–	–
X2	–					
$\overline{\text{RESET}}$	Input	–	System reset input pin (active-low)		–	Ⓑ-A
IC	–	–	Internally connected. Connect directly to V_{DD}.		–	–
V_{DD}	–	–	Positive power supply pin.		–	–
V_{SS}	–	–	Ground potential.		–	–

Note Circled characters indicate Schmitt triggered input.

Download from Www.Somanuals.com. All Manuals Search And Download.

2.2 Description of Pin Functions

2.2.1 P30 to P33 (Port 3) ... I/O pins shared with PTO0 to PTO2
P60 to P63 (Port 6) ... I/O pins shared with AV_{REF}, INT0, PTH00, PTH01
P80 (Port 8) ... I/O pin

These are 4-bit I/O ports with output latches (ports 3 and 6) and a 1-bit I/O port with an output latch (port 8).
Ports 3 and 6 also have the following functions, in addition to the I/O port function.

- Port 3: Timer counter output (PTO0 to PTO2)
- Port 6: Programmable threshold port reference voltage input (AV_{REF})
  Vectored interrupt input (INT0)
  Threshold variable voltage input (PTH00, PTH01)

The input or output mode of ports 3 and 6 is selected by port mode register group A (PMGA), and the input or output mode of port 8 is selected by port mode register group C (PMGC). Ports 3 and 6 can be set to the input or output mode in 1-bit units.

Ports 3, 6, and 8 can also be connected to an internal pull-up resistor by software. This is done by manipulating the pull-up resistor specification registers (POGA and POGB). Specify connection of the pull-up resistor to ports 3 and 6 in 4-bit units. Connection of the pull-up resistor to port 8 can be specified in 1-bit units.

I/O for ports 3 and 6 is possible in 4-bit or 1-bit units. Manipulation in 8-bit units is not possible.

Generation of the $\overline{\text{RESET}}$ signal sets the input mode.

2.2.2 P70 to P73 (Port 7) ... input pins shared with KR4 to KR7
Port 7 is a 4-bit input port.

This port also has a key interrupt input (KR4 to KR7) function, in addition the input port function.

Each pin is always set to input irrespective of the operation of alternate function pins. These pins have Schmitt-triggered input to prevent malfunction due to noise.

Internal pull-up resistors are specifiable by a mask option in 1-bit units.

2.2.3 PTO0 to PTO2 ... output pins shared with port 3
These are the output pins of timer counters 0 to 2, and output square-wave pulses. To output the signal of a timer counter, clear the output latch of the corresponding pin of port 3 to “0”. Then, set the bit corresponding to port 3 of port mode register group A (PMGA) to “1” to set the output mode.

The output of the TOUT F/F pin is cleared to “0” by the timer start instruction.

For details, refer to **6.4.2 (3) Timer counter operation (8-bit)**.

Download from Www.Somanuals.com. All Manuals Search And Download.

2.2.4 INT0 ... input pin shared with port 6

This pin inputs the vectored interrupt signal detected by the edge. A noise eliminator is selectable for INT0. The edge to be detected can be specified by using the edge detection mode register (IM0).

(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

INT0 is an asynchronous input pin, and a signal having a specific high-level width input to this pin can be acknowledged as an interrupt, regardless of the operating clock of the CPU. In addition, an internal noise eliminator can be connected to this pin by software, and the sampling clock that is used for noise elimination can be changed in two steps. In this case, the width of the signal that can be acknowledged differs depending on the CPU operating clock.
When the $\overline{\text{RESET}}$ signal is asserted, IM0 is cleared to “0”, and the rising edge is selected as the active edge.
INT0 can be used to release the STOP and HALT modes. However, when the noise eliminator is selected, INT0 cannot be used to release the STOP and HALT modes.
INT0 is a Schmitt-triggered input pin.

2.2.5 KR4 to KR7 ... input pins shared with port 7

These are key interrupt input pins. KR4 to KR7 are parallel falling edge-detected interrupt input pins.
The interrupt source can be specified for “KR4 to KR7” by using the edge detection mode register (IM2).
When the $\overline{\text{RESET}}$ signal is asserted, these pins serve as port 7 pins and are set to the input mode.

2.2.6 KRREN

This is a key return reset function selection pin. It is always set to input.
When the KRREN pin is high and it is in STOP mode, a falling input on pins KR4/P70 to KR7/P73 generates a system reset. At this time, STOP mode is released.
When the KRREN pin is low, pins KR4/P70 to KR7/P73 function as normal input pins or release standby.

2.2.7 TH00 and TH01 ... input pins shared with port 6

These are the input pins of the programmable threshold port (threshold voltage variable analog input port).
Setting the programmable threshold port mode register (PTHM) can change the threshold voltage in 16 stages.

Download from Www.Somanuals.com. All Manuals Search And Download.

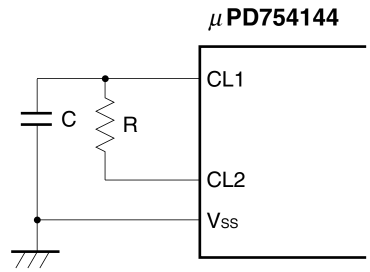
2.2.8 AV$_{REF}$ ... input pin shared with port 6

This is a reference voltage input pin. An analog reference voltage for the programmable threshold port is input.

2.2.9 CL1 and CL2 (μPD754144 only)

These pins are used to connect the RC oscillator resistor (R) and capacitor (C) of the system clock oscillator. No external clock can be input.

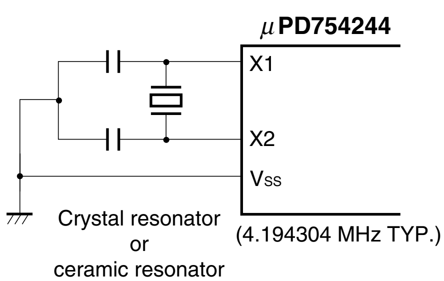
RC oscillation



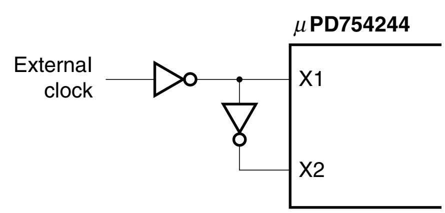
2.2.10 X1 and X2 (μPD754244 only)

These pins connect a crystal/ceramic oscillator for system clock oscillation.
An external clock can also be input to these pins.

(a) Ceramic/crystal oscillation



(b) External clock



2.2.11 $\overline{\text{RESET}}$

This pin inputs an active-low 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-triggered input pin.

This pin can be connected to an internal pull-up resistor by a mask option.

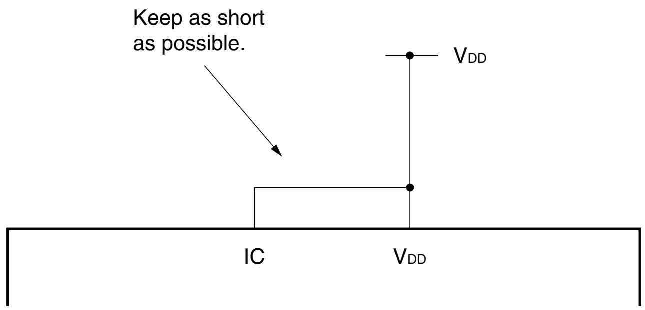
Download from Www.Somanuals.com. All Manuals Search And Download.

2.2.12 IC

The IC (Internally Connected) pin sets the test mode in which the μPD754244 is tested before shipment. Usually, you should directly connect the IC pin 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 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.



2.2.13 V_{DD}

Positive power supply pin.

2.2.14 V_{SS}

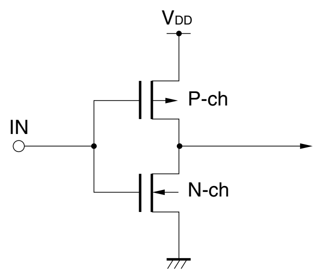
GND.

Download from Www.Somanuals.com. All Manuals Search And Download.

2.3 Pin I/O Circuits

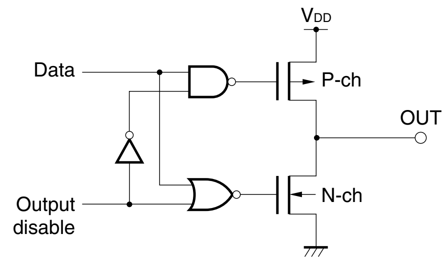
The following diagrams show the I/O circuits of the pins of the µPD754244. Note that in these diagrams the I/O circuits have been slightly simplified.

Type A



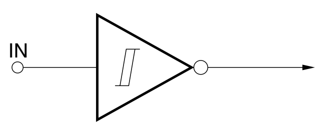
CMOS specification input buffer.

Type D



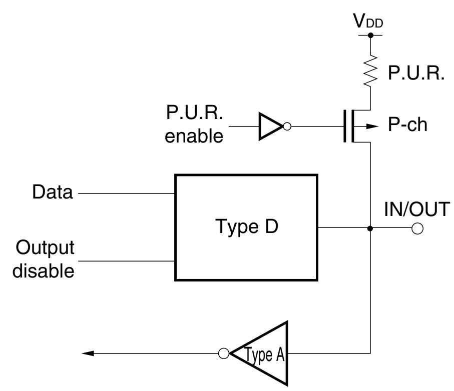
Push-pull output that can be placed in output high-impedance (both P-ch, N-ch off).

Type B



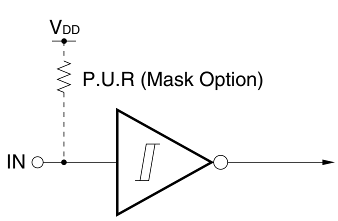
Schmitt-triggered input with hysteresis characteristics.

Type E-B



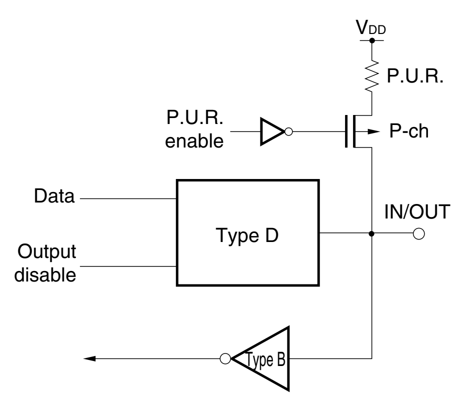
P.U.R. : Pull-Up Resistor

Type B-A



P.U.R. : Pull-Up Resistor

Type F-A



P.U.R. : Pull-Up Resistor

Download from Www.Somanuals.com. All Manuals Search And Download.

2.4 Processing of Unused Pins

Table 2-3. Recommended Connection of Unused Pins

Pin	Recommended Connection
P30/PTO0	Input: Independently connect to V_{SS} or V_{DD} via a resistor. Output: Leave open.
P31/PTO1	
P32/PTO2	
P33	
P60/AV_{REF}	
P61/INT0	
P62/PTH00	
P63/PTH01	
P70/KR4	Connect to V_{DD}.
P71/KR5	
P72/KR6	
P73/KR7	
P80	Input: Independently connect to V_{SS} or V_{DD} via a resistor. Output: Leave open.
KRREN	When this pin is connected to V_{DD}, the internal reset signal is generated at the falling edge of the KRn pin in the STOP mode. When this pin is connected to V_{SS}, the internal reset signal is not generated even if the falling edge of the KRn pin is detected in the STOP mode.
IC	Connect directly to V_{DD}.

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP

The 75XL architecture employed for the μPD754244 has the following features.

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

To realize these superb features, the following techniques have been employed.

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

This chapter describes these features.

3.1 Bank Configuration of Data Memory and Addressing Modes

3.1.1 Bank configuration of data memory

The μPD754244 is provided with a static RAM at the addresses 000H to 07FH of memory bank 0 of the data memory space. EEPROM (16 × 8 bits) is allocated to addresses 400H to 41FH of memory bank 4, and peripheral hardware units (such as I/O ports and timers) are allocated to addresses F80H to FFFH of memory bank 15.

The μPD754244 employs a 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, to address the data memory space of 12-bit address (4K words × 4 bits).

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

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

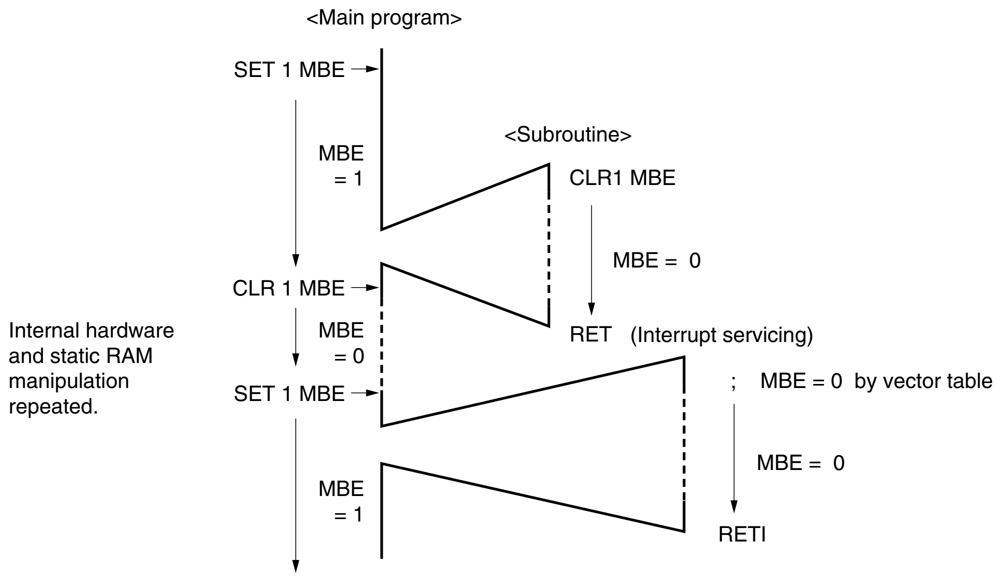
MBS is a register that selects a memory bank. Memory banks 0, 4, and 15 can be set. MBE is a flag that enables or disables the memory bank selected by MBS. When MBE is 0, the specified memory bank (MB) is fixed, regardless of MBS, as shown in Figure 3-1. When MBE is 1, however, a memory bank is selected 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 the mode of MBE = 0 or the mode of MBE = 1 for each processing of the program, programming can be efficiently carried out.

	Adapted Program Processing	Effect
MBE = 0 mode	• Interrupt servicing	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	

Download from Www.Somanuals.com. All Manuals Search And Download.

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



Remark   Solid line: MBE = 1, dotted line: MBE = 0

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

To change MBS by using subroutine processing or interrupt servicing, save or restore it to the 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 4

```
SET1   MBE   ; MBE ← 1
SEL    MB4   ; MBE ← 4
```


Download from Www.Somanuals.com. All Manuals Search And Download.

3.1.2 Addressing mode of data memory

The 75XL architecture employed for the μPD754244 provides the seven types of addressing modes shown in Table 3-1. This means that the data memory space can be efficiently addressed by the bit length of the data to be processed and that programming can be carried out efficiently.

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

This mode is used to directly address 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 to 07FH and the peripheral hardware area of F80H to FFFH can be addressed.

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

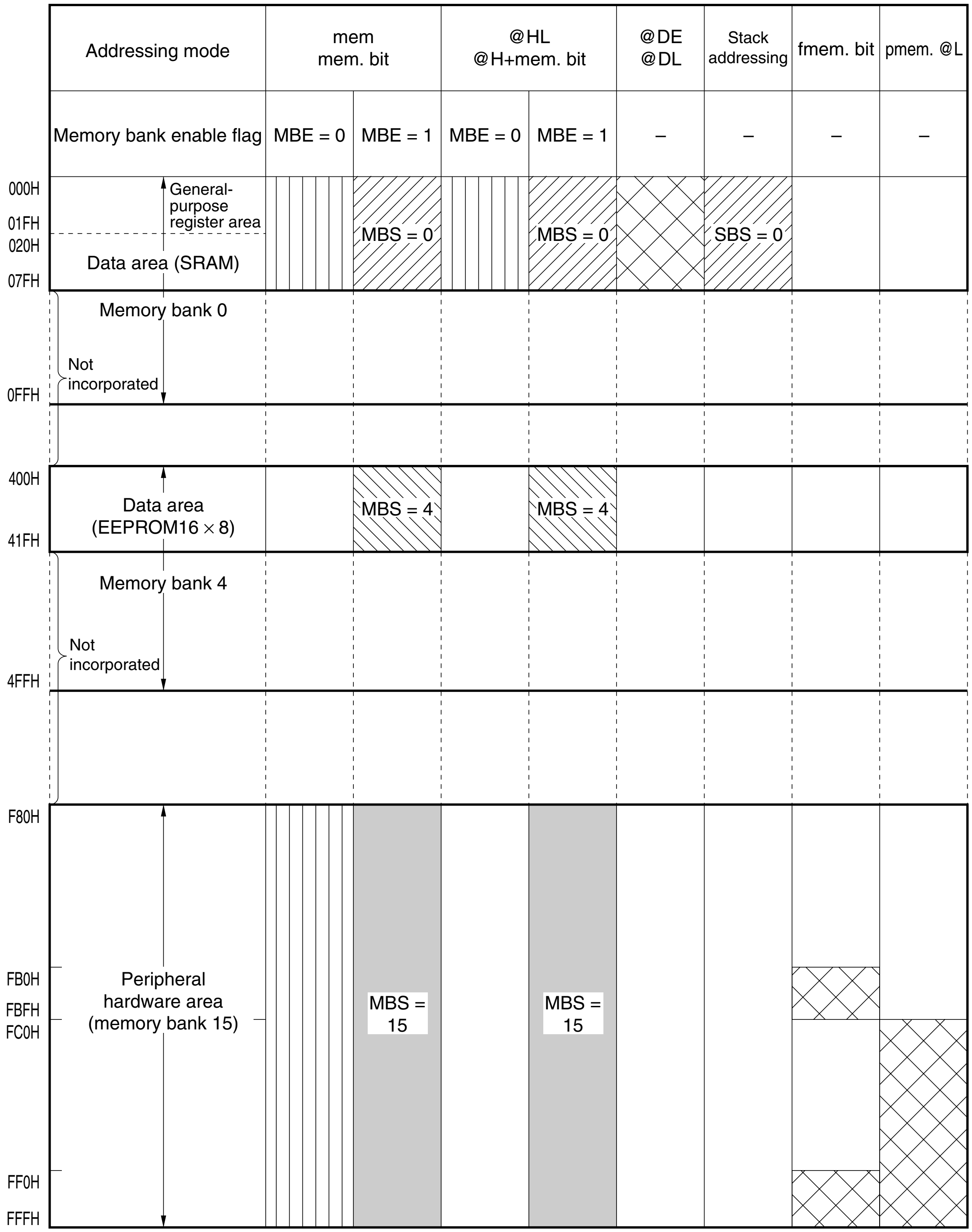
This addressing mode can be used with four instructions: the bit set and reset instructions (SET1 and CLR1), and the two bit test (SKT and SKF).

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

```
FLAG1  EQU   03FH.1   ;  Bit 1 of address 3FH
FLAG2  EQU   057H.2   ;  Bit 2 of address 57H
FLAG3  EQU   077H.0   ;  Bit 0 of address 77H

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


Download from Www.Somanuals.com. All Manuals Search And Download.

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



Remark – : don't care

Caution EEPROM can be manipulated by the following 8-bit manipulation instructions only.

MOV	**XA, @HL**	**XCH**	**XA, @HL**
MOV	**XA, mem**	**XCH**	**XA, mem**
MOV	**@HL, XA**	**SKE**	**XA, @HL**
MOV	**mem, XA**		

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 3-1. Addressing Modes

Addressing Mode	Representation	Specified Address
		• When MBE = 0 When mem = 00H to 7FH: MB = 0 When mem = 80H to FFH: MB = 15 • When MBE = 1: MB = MBS
4-bit direct addressing	mem	Address specified by MB and mem. • When MBE = 0 When mem = 00H to 7FH: MB = 0 When mem = 80H to FFH: MB = 15 • When MBE = 1: MB = MBS
8-bit direct addressing		Address specified by MB and mem (mem is even address) • When MBE = 0 When mem = 00H to 7FH: MB = 0 When mem = 80H to FFH: MB = 15 • When MBE = 1: MB = MBS
4-bit register indirect addressing	@HL	Address specified by MB and HL. Where, MB = MBE MBS
	@HL+ @HL–	Address specified by MB and HL. However, MB = MBE MBS. HL+ automatically increments L register after addressing. HL– automatically decrements L register after addressing.
	@DE	Address specified by DE in memory bank 0
	@DL	Address specified by DL in memory bank 0
8-bit register indirect addressing	@HL	Address specified by MB and HL (contents of L register are even number) Where, MB = MBE MBS
Bit manipulation addressing	fmem.bit	Bit specified by bit at address specified by fmem fmem = FB0H to FBFH (interrupt-related hardware) FF0H to FFFH (I/O port)
	pmem.@L	Bit specified by lower 2 bits of L register at address specified by higher 10 bits of pmem and lower 2 bits of L register. Where, pmem = FC0H to FFFH
	@H+mem.bit	Bit specified by bit at address specified by MB, H, and lower 4 bits of mem. Where, MB = MBE MBS
Stack addressing	—	Address specified by SP in memory bank 0

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) 4-bit direct addressing (mem)

This addressing mode is used 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 to 07FH and the peripheral hardware area of F80H to FFFH in the mode of MBE = 0. In the mode of MBE = 1, MB = MBS, and the entire data memory space can be addressed.

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

(3) 8-bit direct addressing (mem)

This addressing mode is used to directly address 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 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.

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

This addressing mode is used to indirectly address 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. The user selects a register pair depending on the data memory bank to be used in order to carry out programming efficiently.

When register HL is specified, auto increment/decrement mode can be used. This mode is used to increment or decrement register L automatically by 1 at the same time as each instruction is executed, therefore it can reduce the number of program steps.

Example To transfer data 50H to 57H to addresses 60H to 67H

```
DATA1   EQU    57H
DATA2   EQU    67H
        SET1   MBE
        SEL    MB0
        MOV    D, #DATA1 SHR4
        MOV    HL, #DATA2 AND 0FFH ;  HL ← 17H
LOOP :  MOV    A, @DL              ;  A ← (DL)
        XCH    A, @HL              ;  A ← (HL)
        DECS   L                   ;  L ← L – 1
        BR     LOOP
```

The addressing mode 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

Examples 1. To compare data 50H to 57H with data 60H to 67H

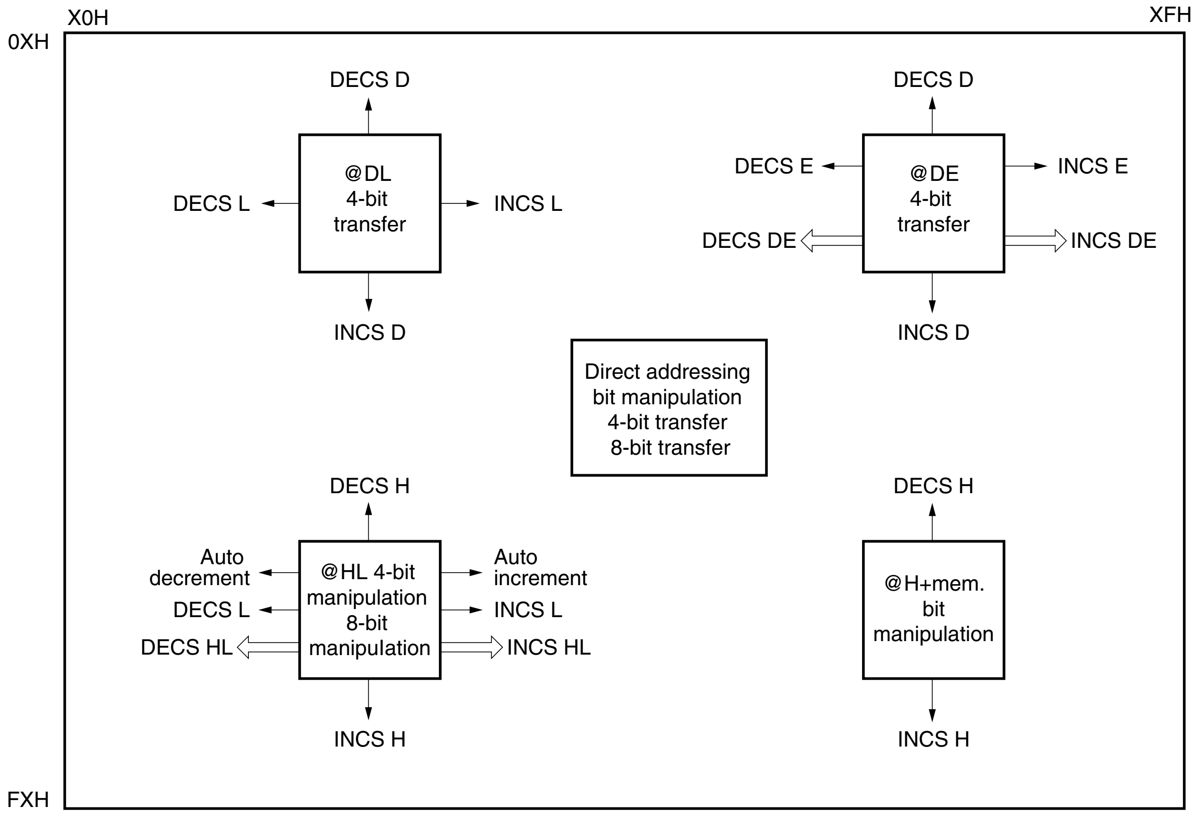
```
DATA1   EQU   57H
DATA2   EQU   67H
        SET1  MBE
        SEL   MB0
        MOV   D, #DATA1 SHR 4
        MOV   HL, #DATA2 AND 0FFH
LOOP  : MOV   A, @DL
        SKE   A, @HL    ; A = (HL)?
        BR    NO        ; NO
        DECS  L         ; YES, L ← L – 1
        BR    LOOP
```

2. To clear data memory of 004H to 07FH

```
        CLR1  RBE
        CLR1  MBE
        MOV   XA, #00H
        MOV   HL, #04H
LOOP  : MOV   @HL, A    ;  (HL) ← A
        INCS  L         ;  L ← L+1
        BR    LOOP
        INCS  H         ;  H ← H+1
        NOP
        SKE   H, #08H
        BR    LOOP
```


Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-3. Updating Address of Static RAM



Download from Www.Somanuals.com. All Manuals Search And Download.

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

This addressing mode is used 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 is specified in the same manner as when the HL register is specified in the 4-bit register indirect addressing mode, by using MBE and MBS. This addressing mode is applicable to the MOV, XCH, and SKE instructions.

Examples 1. To compare whether the count register (T0) value of timer 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     ;  XA = (HL)?
```

2. To clear data memory at 004H to 07FH

```
        CLR1    RBE
        CLR1    MBE
        MOV     XA, #00H
        MOV     HL, #04H
LOOP :  MOV     @HL, XA     ;  (HL) ← XA
        INCS    L
        INCS    L
        BR      LOOP
        INCS    H
        NOP
        SKE     H, #08H
        BR      LOOP
```


Download from Www.Somanuals.com. All Manuals Search And Download.

(6) Bit manipulation addressing

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

While the 1-bit direct addressing mode can only be 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, 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.

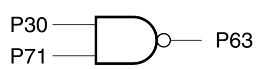
(a) Specific address bit direct addressing (fmem.bit)

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 to FFFH, to which the I/O ports are mapped, and FB0H to 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.

Examples **1.** To test the 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
```

2. To reset P63 if both P30 and P71 pins are 1



(i)

```
        SET1  CY            ;  CY ← 1
        AND1  CY, PORT3.0   ;  CY ∧ P30
        AND1  CY, PORT7.1   ;  CY ∧ P71
        SKT   CY            ;  CY = 1?
        BR    SETP
        CLR1  PORT6.3       ;  P63 ← 0
          :
SETP :  SET1  PORT6.3       ;  P63 ← 1
          :
```

(ii)

```
        SKT   PORT3.0   ;  P30 = 1?
        BR    SETP
        SKT   PORT7.1   ;  P71 = 1?
        BR    SETP
        CLR1  PORT6.3   ;  P63 ← 0
          :
SETP:   SET1  PORT6.3   ;  P63 ← 1
```


Download from Www.Somanuals.com. All Manuals Search And Download.

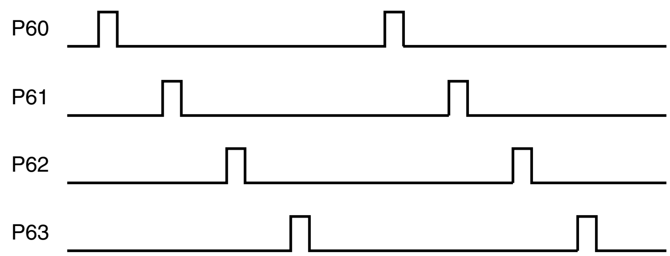
(b) Specific address bit register indirect addressing (pmem, @L)

This addressing mode is 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 to FFFH.

This addressing mode specifies the higher 10 bits of a 12-bit data memory address directly by using an operand, and the lower 2 bits by using 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 of port 6




```
LOOP2 :  MOV    L, #0
LOOP1 :  SET1   PORT6.@L;  Bits of port 6 (L1-0) ← 1
         CLR1   PORT6.@L;  Bits of port 6 (L1-0) ← 0
         INCS   L
         SKE    L, #4H
         BR     LOOP1
         BR     LOOP2
```

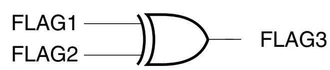

Download from Www.Somanuals.com. All Manuals Search And Download.

(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 MBE and 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 bit 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
        CLR1    CY              ;  CY ← 0
        OR1     CY, @H+FLAG1    ;  CY ← CY ∨ FLAG1
        XOR1    CY, @H+FLAG2    ;  CY ← CY ∀ FLAG2
        SET1    @H+FLAG3        ;  FLAG3 ← 1
        SKT     CY              ;  CY = 1?
        CLR1    @H+FLAG3        ;  FLAG3 ← 0
```


Download from Www.Somanuals.com. All Manuals Search And Download.

(7) Stack addressing

This addressing mode is used to save or restore data when interrupt servicing 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.

In addition to being used during interrupt servicing or subroutine processing, this addressing is also used to save or restore register contents by using the PUSH or POP instruction.

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
```


Download from Www.Somanuals.com. All Manuals Search And Download.

3.2 Bank Configuration of General-Purpose Registers

The μPD754244 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 to 1FH of memory bank 0 (refer to **Figure 3-5 Configuration of General-Purpose Registers (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 selected
			0	1	Bank 1 selected
			1	0	Bank 2 selected
			1	1	Bank 3 selected

Fixed to 0 (RBS bits 3 and 2)

Remark × = don't care

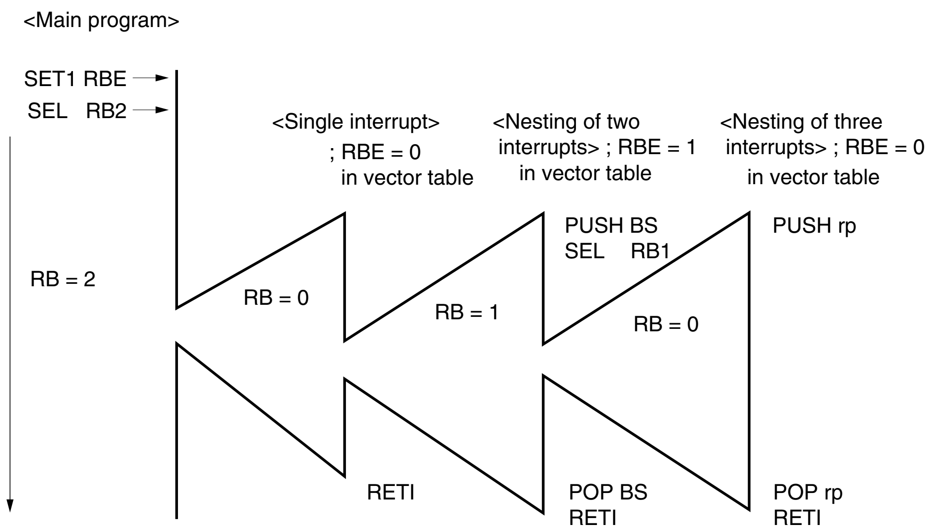
RBE is automatically saved or restored during subroutine processing and therefore can be set while subroutine processing is under execution. When interrupt servicing is executed, RBE is automatically saved or restored, and RBE can be set during interrupt servicing depending on the setting of the interrupt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

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

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

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-4. Example of Using Register Banks



If RBS is to be changed in the course of subroutine processing or interrupt servicing, 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 in the μPD754244 can be used not only as 4-bit registers but also as 8-bit register pairs. This feature allows the μPD754244 to provide transfer, operation, comparison, and increment/decrement instructions comparable to those of 8-bit microcontrollers and allows you to program using mainly general-purpose registers.

Download from Www.Somanuals.com. All Manuals Search And Download.

(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 RBE and 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 and carry occurs
SUBC   DE', XA     ; DE' ← DE' – XA – CY
MOV    XA, XA'     ; XA ← XA'
MOVT   XA, @PCDE ; XA ← (PC11–8+DE) ROM, table reference
SKE    XA, BC      ; Skips if XA = BC
```

2. To test whether the value of the count register (T0) of timer counter 0 is greater than the value of register pair BC' and, if not, to wait until it becomes greater

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

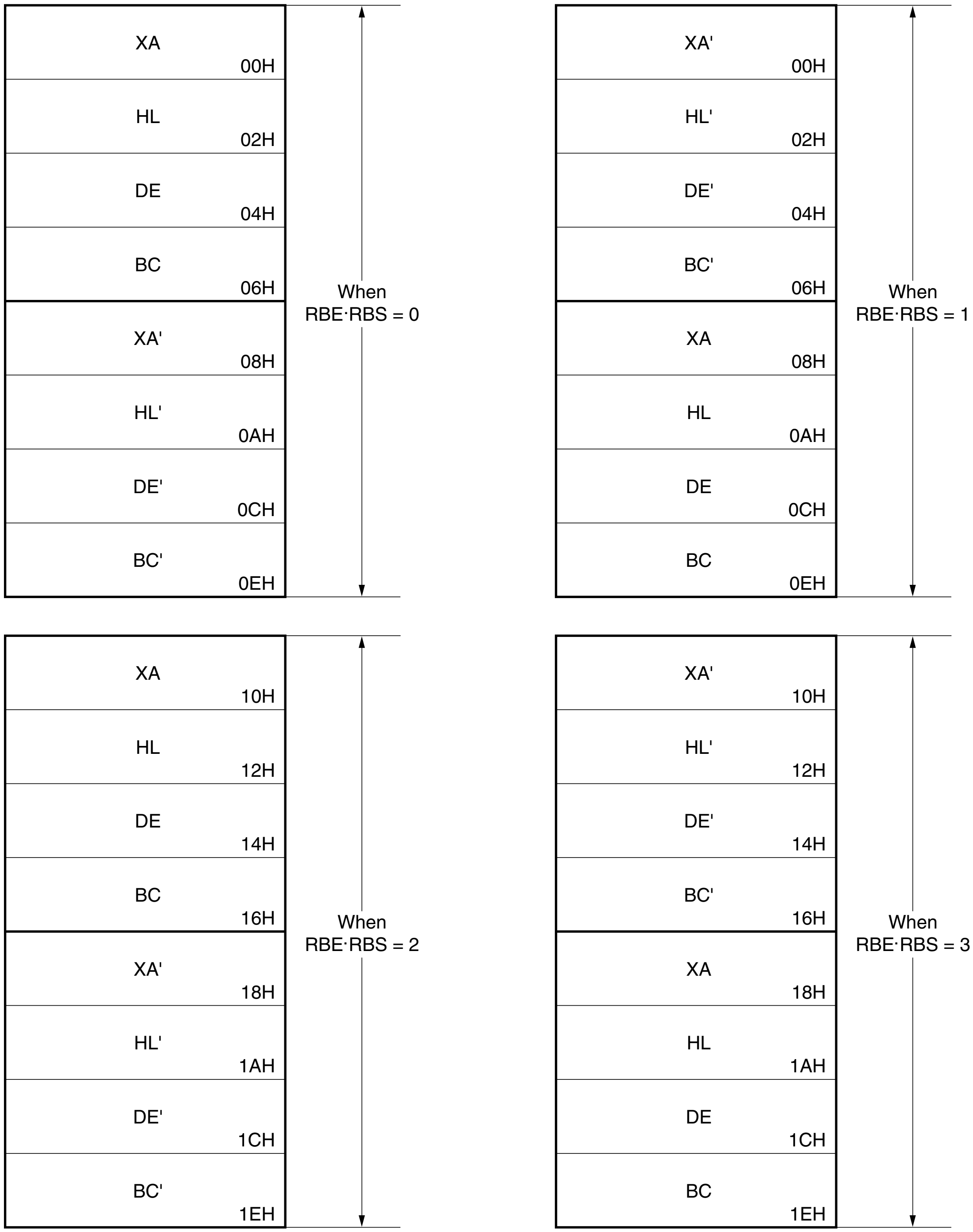

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-5. Configuration of General-Purpose Registers (4-Bit Processing)

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

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-6. Configuration of General-Purpose Registers (8-Bit Processing)



Download from Www.Somanuals.com. All Manuals Search And Download.

3.3 Memory-Mapped I/O

The μPD754244 employs memory-mapped I/O that maps peripheral hardware units such as I/O ports and timers to addresses F80H to FFFH on the data memory space, as shown in Figure 3-2. Therefore, no special instructions to control the peripheral hardware units are provided, and all the hardware units are controlled by using memory manipulation instructions. (Some mnemonics that make the program easy to read are provided for hardware control.)

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

Table 3-4. Addressing Modes Applicable to Peripheral Hardware Unit Manipulation

	Applicable Addressing Mode	Hardware Units
Bit manipulation	Specified in direct addressing mode mem.bit with MBE = 0 or (MBE = 1, MBS = 15)	All hardware units that can be manipulated in 1-bit units
	Specified in direct addressing mode fmem.bit regardless of setting of MBE and MBS	IST1, IST0, MBE, RBE IE×××, IRQ×××, PORTn.×
	Specified in indirect addressing mode pmem.@L regardless of setting of MBE and MBS	BSBn.× PORTn.×
4-bit manipulation	Specifies in direct addressing mode mem with MBE = 0 or (MBE = 1, MBS = 15)	All hardware units that can be manipulated in 4-bit units
	Specified in register indirect addressing @HL with (MBE = 1, MBS = 15)	
8-bit manipulation	Specified in direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15), where mem is even number.	All hardware units that can be manipulated in 8-bit units
	Specified in register indirect addressing @HL with MBE = 1, MBS = 15, where contents of L register are even number	

Example

```
CLR1    MBE         ; MBE = 0
SET1    TM0. 3      ; Starts timer 0
EI      IE0         ; Enables INT0
DI      IET1        ; Disables INTT1
SKTCLR  IRQ2        ; Tests and clears INT2 request flag
SET1    PORT3, @L   ; Sets port 3
IN      A, PORT6    ; A ← port 6
```


Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7 shows the I/O map of the μPD754244.

The meanings of the symbols shown in this figure are as follows.

- Symbol .... Name indicating the address of an internal hardware unit
  Can be written in operands of instructions
- R/W .... Indicates whether the hardware unit in question can be read or written
  R/W: Read/write
  R: Read only
  W: Write only
- Number of bits that can be manipulated .... Indicates the bit units in which the hardware unit in question can be manipulated
  ○ : Can be manipulated in specified units (1, 4, or 8 bits)
  △ : Only some bits can be manipulated. For the bits that can be manipulated, refer to Remarks.
  – : Cannot be manipulated in specified units (1, 4, or 8 bits).
- Bit manipulation addressing .... Indicates a bit manipulation addressing mode that can be used to manipulate the hardware unit in question in 1-bit units

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (1/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
F80H	Stack pointer (SP)				R/W	–	–	○	–	Bit 0 is fixed to 0.
F82H	Register bank selection register (RBS)				R	–	○	○	–	**Note 1**
	Bank selection register (BS)									
F83H	Memory bank selection register (MBS)					–	○			
F84H	Stack bank selection register (SBS)				R/W	–	○	–	–	
F85H	Basic interval timer mode register (BTM)				W	△	○	–	mem.bit	Bit manipulation can be performed only on bit 3.
F86H	Basic interval timer (BT)				R	–	–	○	–	
F88H	Modulo register for setting timer counter 2 high-level period (TMOD2H)				R/W	–	–	○	–	
F8AH	Unmounted									
F8BH	WDTM[Note 2]	–	–	–	W	○	–	–	mem.bit	
F8CH to F8FH	Unmounted									

Notes 1. Manipulation is possible separately with RBS and MBS in 4-bit manipulation. Manipulation is possible with BS in 8-bit manipulation. Write data into MBS and RBS with the SEL MBn (n = 0, 4 or 15) and SEL RBn (n = 0-3) instructions.

2. WDTM: Watchdog timer enable flag (W); Once set, cannot be cleared by an instruction.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (2/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
F90H	Timer counter 2 mode register (TM2)				R/W	△ (W)	–	○	–	Bit manipulation can be performed only on bit 3
						–	–		–	
F92H	TOE2	REMC	NRZB	NRZ	R/W	○	○	○	–	Bit 3 can only be written
	Timer counter 2 control register (TC2)									
	0	–	–	–		–	–			Only 0 can be written to bit 3
F94H	Timer counter 2 count register (T2)				R	–	–	○	–	
F96H	Timer counter 2 modulo register (TMOD2)				R/W	–	–	○	–	
F98H to F9FH	Unmounted									

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (3/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
FA0H	Timer counter 0 mode register (TM0)				R/W	△ (W)	–	○	mem.bit	Bit manipulation can be performed only on bit 3
						–	–		–	
FA2H	TOE0[Note 1]	–	–	–	W	○	–	–	mem.bit	
FA3H	Unmounted									
FA4H	Timer counter 0 count register (T0)				R	–	–	○	–	
FA6H	Timer counter 0 modulo register (TMOD0)				R/W	–	–	○	–	
FA8H	Timer counter 1 mode register (TM1)				R/W	△ (W)	–	○	mem.bit	Bit manipulation can be performed only on bit 3
						–	–		–	
FAAH	TOE1[Note 2]	–	–	–	W	○	–	–	mem.bit	
FABH	Unmounted									
FACH	Timer counter 1 count register (T1)				R	–	–	○	–	
FAEH	Timer counter 1 modulo register (TMOD1)				R/W	–	–	○	–	

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

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

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (4/8)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Number of bits that can be manipulated 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FB0H	IST1	IST0	MBE	RBE	R/W	○(R/W)	○(R/W)	○(R)	fmem.bit	R only possible as 8-bit manipulation.
	Program status word (PSW)									
	CY[Note 1]	SK2[Note 1]	SK1[Note 1]	SK0[Note 1]		△ Note 2	–			
FB2H	Interrupt priority selection register (IPS)				R/W	–	○	–		**Note 3**
FB3H	Processor clock control register (PCC)				R/W	–	○	–		**Note 4**
FB4H	INT0 edge detection mode register (IM0)				R/W	–	○	–	–	
FB5H	Unmounted									
FB6H	INT2 edge detection mode register (IM2)[Note 5]				R/W	–	○	–	–	
FB7H	Unmounted									
FB8H	INTA register (INTA)				R/W	○	○	–	fmem.bit	Bit manipulation can be performed by reserved word only.
	–	–	IEBT	IRQBT						
FB9H	INTB register (INTB)				R/W	○	○	–		
	IEEE	IRQEE	–	–						
FBAH	Unmounted									
FBBH										
FBCH	INTE register (INTE)				R/W	○	○	–	fmem.bit	Bit manipulation can be performed by reserved word only.
	IET1	IRQT1	IET0	IRQT0						
FBDH	INTF register (INTF)				R/W	○	○	–		
	IET2	IRQT2	–	–						
FBEH	INTG register (INTG)				R/W	○	○	–		
	–	–	IE0	IRQ0						
FBFH	INTH register (INTH)				R/W	○	○	–		
	–	–	IE2	IRQ2						

Remarks 1. IE××× is an interrupt enable flag.

2. IRQ××× is an interrupt request flag.

Notes 1. These are not registered as reserved words.

2. Use the CY manipulation instruction to write to CY.

3. IME (bit 3) can only be manipulated by an EI/DI instruction.

4. PCC3 (bit 3) and PCC2 (bit 2) can be manipulated by a STOP/HALT instruction.

5. This register specifies the falling edge of the KRn pin as the set signal of the interrupt request flag (IRQ2). This register is initialized to 00H after reset. Therefore, write 01H to set the falling edge of the KRn pin to IRQ2.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (5/8)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Number of bits that can be manipulated 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FC0H	Bit sequential buffer 0 (BSB0)				R/W	○	○	○	mem.bit	
FC1H	Bit sequential buffer 1 (BSB1)				R/W	○	○		pmem.@L	
FC2H	Bit sequential buffer 2 (BSB2)				R/W	○	○	○		
FC3H	Bit sequential buffer 3 (BSB3)				R/W	○	○			
FC4H	Unmounted									
FC5H										
FC6H	Reset detection flag register (RDF) KRF	WDF	–	–	R/W	△	○	–	mem.bit	Manipulation can be performed only on bits 2 and 3.
FC7H to FCDH	Unmounted									
FCEH	EWE[Note 1]	EWST[Note 1]	–	–	R/W	○	–	○	mem.bit	A write to an unmounted area is invalid, and the read value is undefined.
FCFH	EEPROM write control register (EWC) ERE[Note 1]	EWTC6[Note 2]	EWTC5[Note 2]	EWTC4[Note 2]		△	–			

Notes 1. In bit manipulation: EWE = R/W, EWST = R only, ERE = R/W.

2. These are not registered as reserved words.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (6/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
FD0H to FD3H	Unmounted									
FD4H	Programmable threshold port (PTH0)				R	○	○	–	mem.bit	
FD5H	Unmounted									
FD6H	PTHM3[Note]	PTHM2[Note]	PTHM1[Note]	PTHM0[Note]	R/W	–	–	○	mem.bit	A write to bit 4 or bit 5 is invalid, and the read value is undefined.
	Programmable threshold port mode register (PTHM)									
	PTHM7[Note]	PTHM6[Note]	–	–						
FD8H to FDBH	Unmounted									
FDCH	PO3[Note]	–	–	–	R/W	–	–	○	–	A write to an unmounted area is invalid, and the read value is undefined.
	Pull-up resistor specification register group A (POGA)									
	–	PO6[Note]	–	–						
FDEH	–	–	–	PO8[Note]	R/W	–	–	○	–	
	Pull-up resistor specification register group B (POGB)									
	–	–	–	–						

Note These are not registered as reserved words.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (7/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
FE0H to FE7H	Unmounted									
FE8H	PM33	PM32	PM31	PM30	R/W	○	–	○	–	
	Port mode register group A (PMGA)									
	PM63[Note]	PM62[Note]	PM61[Note]	PM60[Note]						
FEAH to FEDH	Unmounted									
FEEH	–	–	–	PM8[Note]	R/W	–	–	○	–	A write to an unmounted area is invalid, and the read value is undefined.
	Port mode register group C (PMGC)									
	–	–	–	–						

Note These are not registered as reserved words.
However, bit manipulation is possible by using 0FE9.0 to 0FE9.3.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 3-7. μPD754244 I/O Map (8/8)

Address	Hardware name (symbol)				R/W	Number of bits that can be manipulated			Bit manipulation addressing	Remarks
	b3	b2	b1	b0		1-bit	4-bit	8-bit		
FF0H to FF2H	Unmounted									
FF3H	Port 3 (PORT3)				R/W	○	○	–	fmem.bit pmem.@L	
FF4H	Unmounted									
FF5H										
FF6H	Port 6 (PORT6)				R/W	○	○	–	fmem.bit pmem.@L	
FF7H[Note 1]	Port 7 (PORT7)				R	○	○	–		
	KR7	KR6	KR5	KR4						
FF8H	Port 8 (PORT8)				R/W	△	○	–		A write to an unmounted area is invalid, and the read value is undefined.
	–	–	–	P80[Note 2]						
FF9H to FFFH	Unmounted									

Notes 1. KR4 to KR7 can only be read in 1-bit units. In 4-bit parallel input, PORT7 is used for specification.

2. These are not registered as reserved words.

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 4 INTERNAL CPU FUNCTION

4.1 Function to Select MkI and MkII Modes

4.1.1 Difference between MkI and MkII modes

The CPU of the μPD754244 has two modes to be selected: MkI and MkII. These modes can be selected by using bit 3 of the stack bank select register (SBS).

- MkI mode: In this mode, the μPD754144 is upwardly-compatible with the 75X Series.
  This mode can be used with the CPU in the 75XL Series having a ROM capacity of up to 16 KB.
- MkII mode: In this mode, the μPD754144 is not compatible with the 75X Series.
  This mode can be used with all the CPUs in the 75XL Series, including the models having a ROM capacity of 16 KB or higher.

Table 4-1. Differences Between MkI and MkII Modes

	MkI Mode	MkII Mode
Number of stack bytes of subroutine instruction	2 bytes	3 bytes
BRA !addr1 instruction CALLA !addr1 instruction	Not provided	Provided
CALL !addr instruction	3 machine cycles	4 machine cycles
CALLF !faddr instruction	2 machine cycles	3 machine cycles

Caution The MkII mode supports a program area exceeding 16 KB for the 75X and 75XL Series. This mode enhances software compatibility of the μPD754244 with a product with a program area of more than 16 KB.
When the MkII mode is selected, The number of stack bytes increases by one byte per stack, as compared with the MkI mode, when the subroutine call instruction is executed. When the CALL !addr or CALLF !faddr instruction is used, the machine cycle is extended by 1 cycle. To emphasize the use efficiency of the RAM or processing capability more than software compatibility, therefore, use the MkI mode.

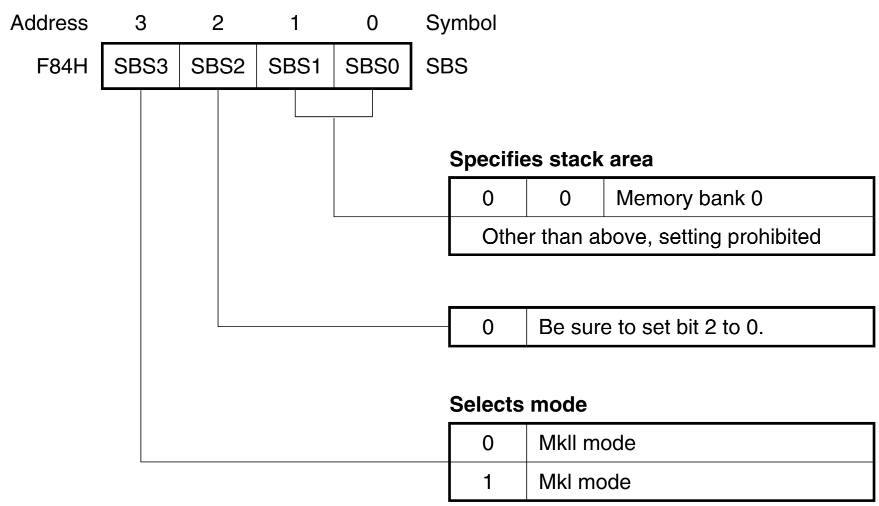
Download from Www.Somanuals.com. All Manuals Search And Download.

4.1.2 Setting stack bank select register (SBS)

The MkI mode or MkII mode is selected by using the stack bank select register (SBS). Figure 4-1 shows the format of this register.

The stack bank select register is set by using a 4-bit memory manipulation instruction. To use the MkI mode, be sure to initialize the stack bank select register to 1000B at the beginning of the program. To use the MkII mode, initialize the register to 0000B.

Figure 4-1. Format of Stack Bank Select Register



Caution The SBS.3 bit is set to "1" after the $\overline{RESET}$ signal has been asserted. Therefore, the CPU operates in the MkI mode. To use the instructions in the MkII mode, clear SBS.3 to "0" to set the MkII mode.

Download from Www.Somanuals.com. All Manuals Search And Download.

4.2 Program Counter (PC) ··· 12 bits

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

Figure 4-2. Configuration of Program Counter

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 that 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). Then, the jump destination address is loaded to the PC.

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

When the $\overline{\text{RESET}}$ signal is asserted, the contents of the program counter (PC) are initialized to the contents of address 0000H and 0001H of the program memory, and the program can be started from any address according to the contents.

$PC_{11-8} \leftarrow (0000H)_{3-0}$, $PC_{7-0} \leftarrow (0001H)_{7-0}$

Download from Www.Somanuals.com. All Manuals Search And Download.

4.3 Program Memory (ROM) ··· 4096 × 8 bits

The program memory stores a program, interrupt vector table, the reference table of the GETI instruction, and table data.

The program memory is addressed by the program counter. The table data can be referenced by using a table reference instruction (MOVT).

Figure 4-3 shows address ranges in which execution can be branched by a branch or subroutine call instruction. A relative branch instruction (BR $addr1 instruction) can branch execution to an address of [contents of PC –15 to –1 or +2 to +16], regardless of the block boundary.

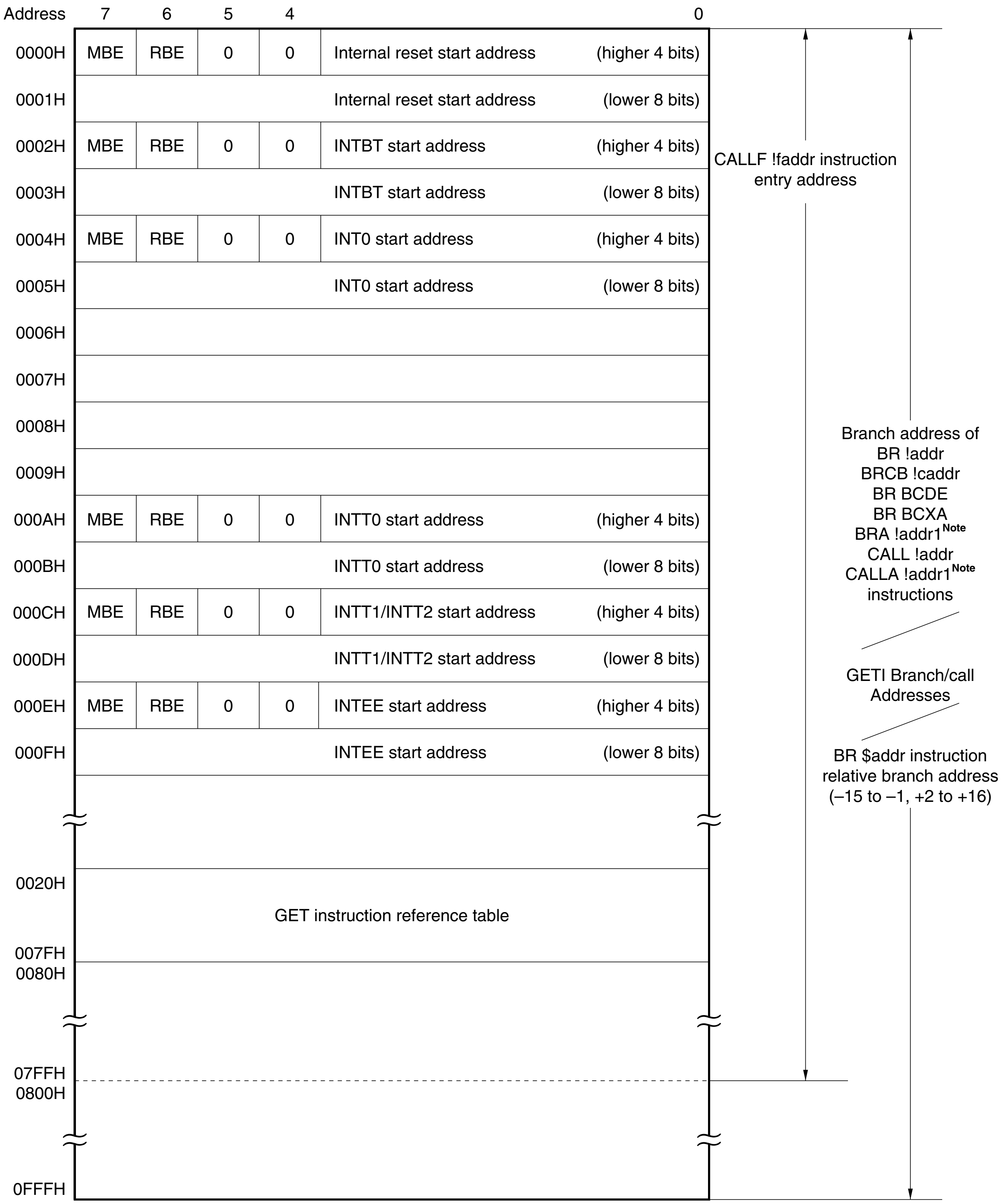
The address range of the program memory of each model is 0000H to 0FFFH, and among them, special functions are assigned to the following addresses. All the addresses other than 0000H and 0001H can be used as normal program memory addresses.

- **Addresses 0000H and 0001H**
  These addresses store the start address from which program execution is to be started when the $\overline{\text{RESET}}$ signal is asserted, and the vector table to which the set values of RBE and MBE are written. Program execution can be reset and started from any address.
- **Addresses 0002H to 000FH**
  These addresses store the start address from which program execution is to be started when a vector interrupt occurs, and the vector table to which the set values of RBE and MBE are written. Interrupt servicing can be started from any address.
- **Addresses 0020H to 007FH**
  These addresses constitute a table area that can be referenced by the GETI instruction[Note].

Note The GETI instruction implements any 2- or 3-byte instruction, or two 1-byte instructions with 1 byte. It is used to decrease the number of program steps (refer to **11.1.1 GETI instruction**).

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 4-3. Program Memory Map



Note Can be used in the MkII mode only.

Remark In addition to the above, a branch can be made to an address with the lower 8-bits only of the PC changed by means of a BR PCDE or BR PCXA instruction.

Download from Www.Somanuals.com. All Manuals Search And Download.

4.4 Data Memory (RAM) ... 128 words × 4 bits

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

- Memory bank 0 (data areas)
- Memory bank 4 (EEPROM)
- Memory bank 15 (peripheral hardware area)

4.4.1 Configuration of data memory

(1) Data area

A data area consists of a 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 retained for a long time by battery backup even when the CPU is halted in standby mode. The data area is manipulated by using memory manipulation instructions.

Static RAM is mapped to memory bank 0 in units of 128 words × 4 bits only. Although bank 0 is mapped as a data area, it can also be used as a general-purpose register area (000H to 01FH) and as a stack area (000H to 07FH).

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.

- **Stack area**
  The stack area is set by an instruction and is used as a saving area when a subroutine or interrupt service is executed.

(2) EEPROM (Electrically Erasable PROM)

In EEPROM memory bank 4 (400H to 4FFH), only 16 words × 8 bits at 400H to 41FH are mapped.
Reading/writing of EEPROM is performed in 8-bit units.
Since 420H to 4FFH of memory bank 4 is an unmounted area, any value written to this area is ignored and the read value becomes undefined.

(3) Peripheral hardware area

The peripheral hardware area is mapped to addresses F80H to FFFH of memory bank 15.
This area is manipulated by using a memory manipulation instruction, in the same manner as the static RAM. 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

4.4.2 Specifying bank of data memory

A memory bank is specified by a 4-bit memory bank select register (MBS) when bank specification is enabled by setting a memory bank enable flag (MBE) to 1 (MBS = 0, 4, or 15). When bank specification is disabled (MBS = 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.

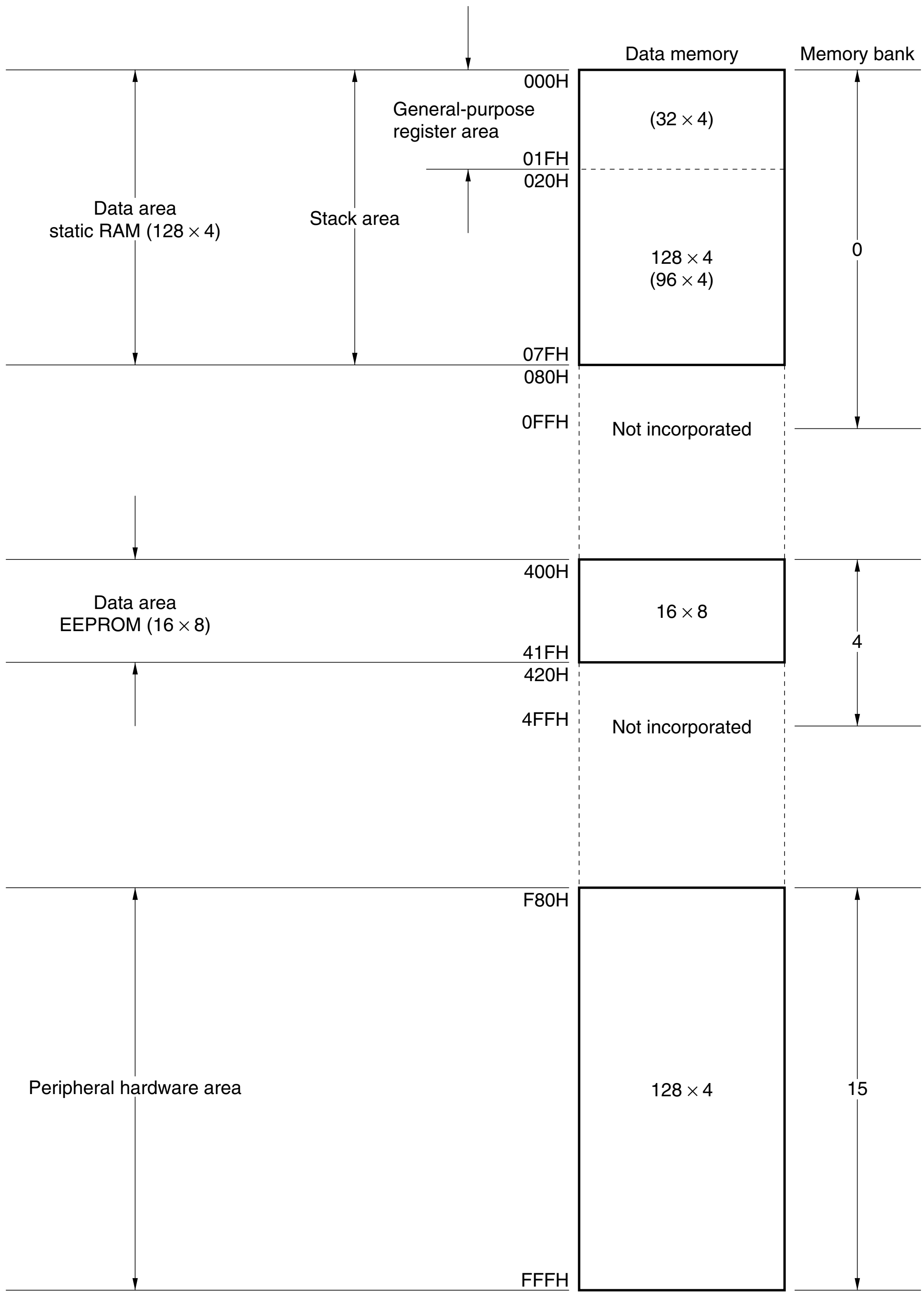
For the details of memory bank selection and addressing, refer to **3.1 Bank Configuration of Data Memory and Addressing Mode**.

For how to use a specific area of the data memory, refer to the following.

- General-purpose register area .... **4.5 General-Purpose Registers**
- Stack area .... **4.7 Stack Pointer (SP) and Stack Bank Select Register (SBS)**
- EEPROM .... **CHAPTER 5 EEPROM**
- Peripheral hardware area .... **CHAPTER 6 PERIPHERAL HARDWARE FUNCTION**

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 4-4. Data Memory Map



Download from Www.Somanuals.com. All Manuals Search And Download.

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 to 07FH

```
          SET1   MBE
          SEL    MB0
          MOV    XA, #00H
          MOV    HL, #04H
RAMC0 :   MOV    @HL, A     ;Clears 004H to 07FHNote
          INCS   L          ;L ← L+1
          BR     RAMC0
          INCS   H          ;H ← H+1
          NOP
          SKE    H, #08H
          BR     RAMC0
```

Note Data memory addresses 000H to 003H are not cleared because they are used as general-purpose register pairs XA and HL.

Download from Www.Somanuals.com. All Manuals Search And Download.

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

General-purpose registers are mapped to the specific addresses of the data memory. Four banks of registers, with each bank consisting of eight 4-bit registers (B, C, D, E, H, L, X, and A), are available.

The register bank (RB) that becomes valid when an instruction is executed is determined by the following expression.

RB = RBE· RBS (RBS = 0 to 3)

Each general-purpose register is manipulated in 4-bit units. Moreover, two registers can be used in pairs, such as BC, DE, HL, and XA, and manipulated in 8-bit units. Register pairs DE, HL, and DL are also used as data pointers.

When registers are manipulated in 8-bit units, the register pairs of the register bank (RB) with bit 0 inverted (0 ↔ 1, 2 ↔ 3), BC', DE', HL', and XA', can also 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 and accessed as an ordinary RAM area, regardless of whether the registers in this area are used or not.

Figure 4-5. Configuration of General-Purpose Register Area

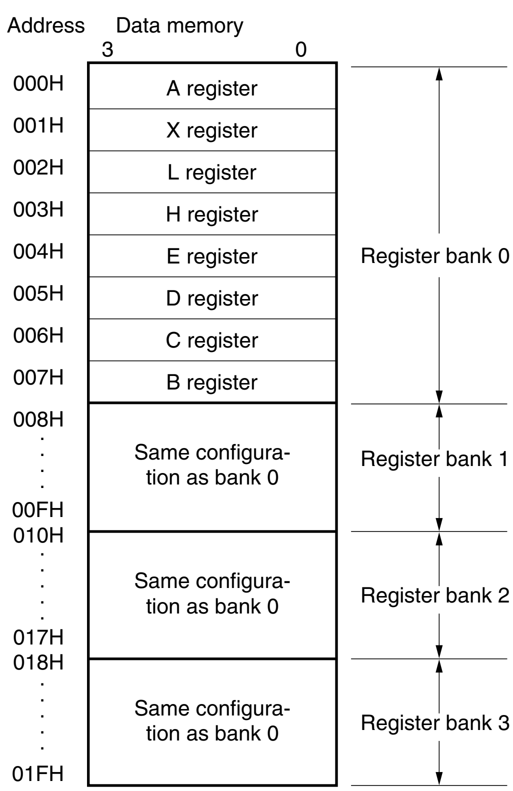
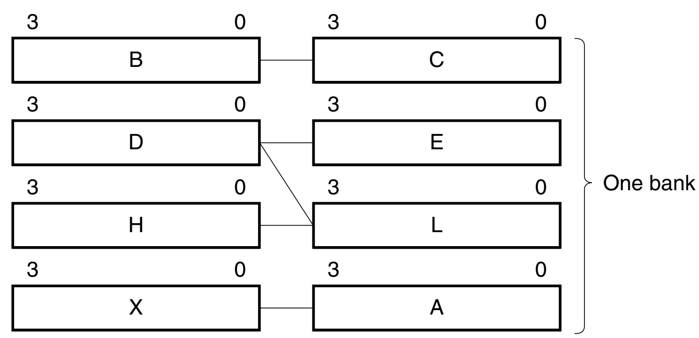


Figure 4-6. Configuration of Register Pair



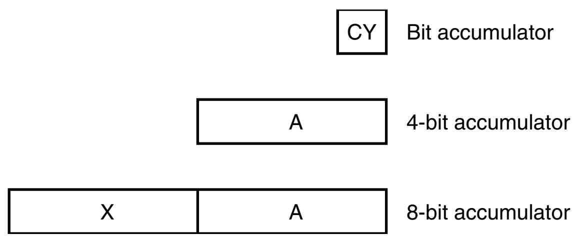
Download from Www.Somanuals.com. All Manuals Search And Download.

4.6 Accumulator

With the μPD754244, the A register or XA register pair functions as an accumulator. The A register plays a central role in 4-bit data processing, while the XA register pair is used for 8-bit data processing.

When a bit manipulation instruction is used, the carry flag (CY) is used as a bit accumulator.

Figure 4-7. Accumulator



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

The μPD754244 uses a static RAM as the stack memory (LIFO). The stack pointer (SP) is an 8-bit register that holds information on the first address of the stack area.

The stack area consists of addresses 000H to 07FH of memory bank 0. A memory bank is specified by 2-bit SBS (refer to **Table 4-2**).

Table 4-2. Stack Area Selected by SBS

SBS		Stack Area
SBS1	SBS2	
0	0	Memory bank 0
Other than above, setting prohibited		

The value of SP is decremented before data is written (saved) to the stack area, and is incremented after data has been read (restored) from the stack memory.

The data saved or restored to or from the stack are as shown in Figures 4-9 to 4-12.

The initial values of SP and SBS are respectively set by an 8-bit memory manipulation instruction and 4-bit memory manipulation instruction, to determine the stack area. The values of SP and SBS can also be read.

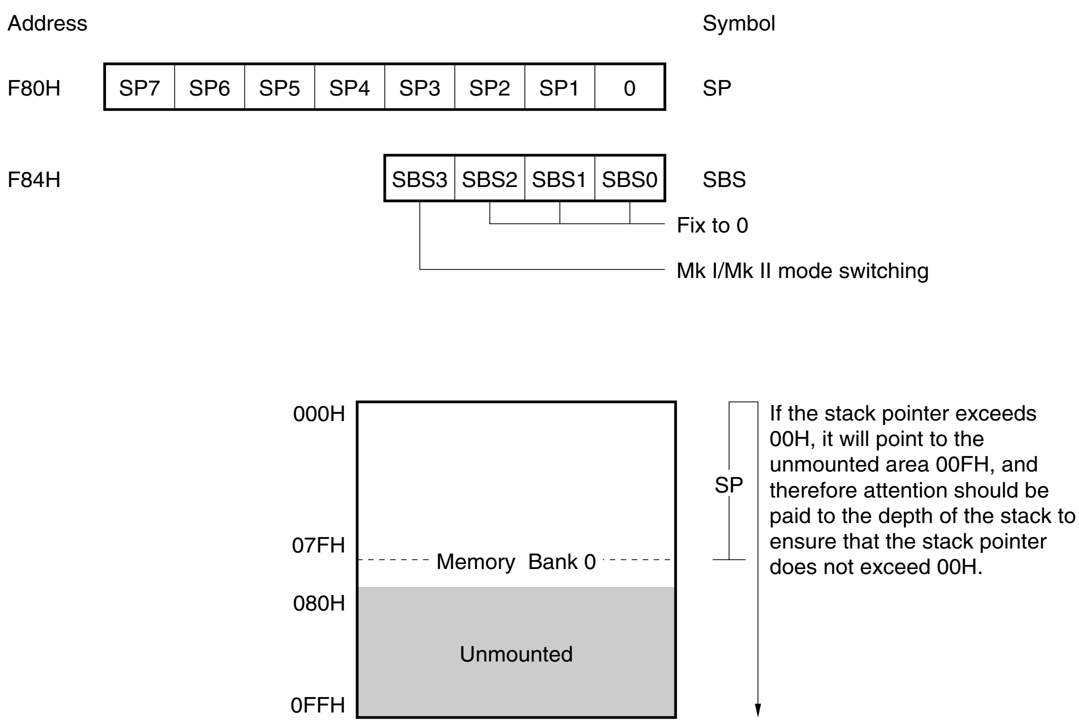
Download from Www.Somanuals.com. All Manuals Search And Download.

When 00H is set to SP as the initial value, memory bank 0 specified by SBS is used as the stack area, starting from the highest address (07FH).

The stack area can be used only in memory bank 0. If stack operation is performed from address 000H onwards, the stack pointer will point to unmounted area 0FFH. Therefore, be careful not to allow the stack pointer to exceed 000H.

The contents of SP become undefined and the contents of SBS become 1000B when the $\overline{\text{RESET}}$ signal is asserted. Therefore, be sure to initialize these to the desired values at the beginning of the program.

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



Example of SP initialization

To set the stack area in memory bank 0, and perform stack operations from address 07FH.

```
SEL  MB15      ; Or CLR1 MBE
MOV  A, #0
MOV  SBS, A    ; Specify memory bank 0 as stack area
MOV  XA, #80H
MOV  SP, XA    ; SP ← 80H (stack operations from 7FH)
```


Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 4-9. Data Saved to Stack Memory (MkI Mode)

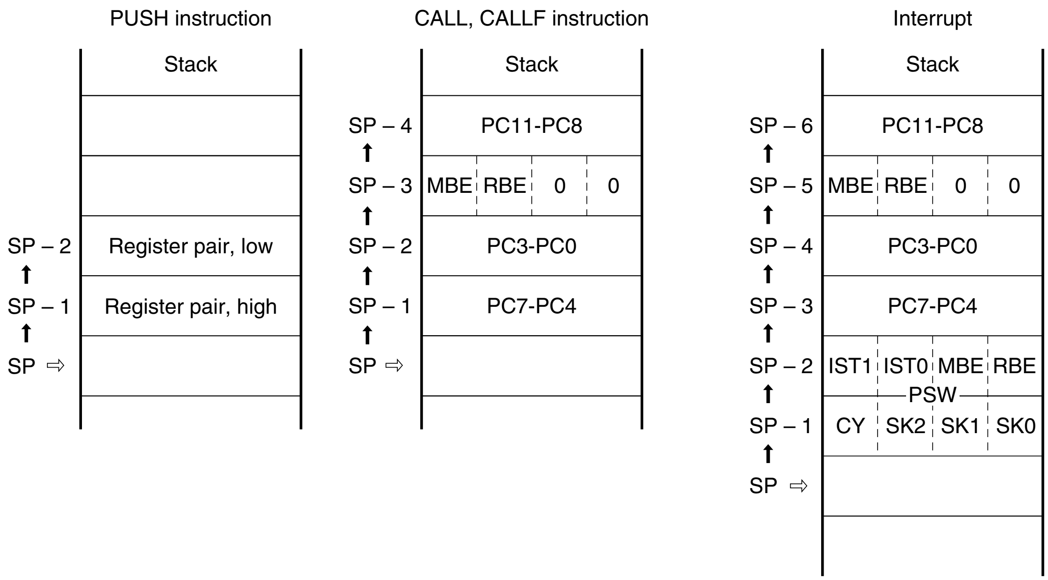
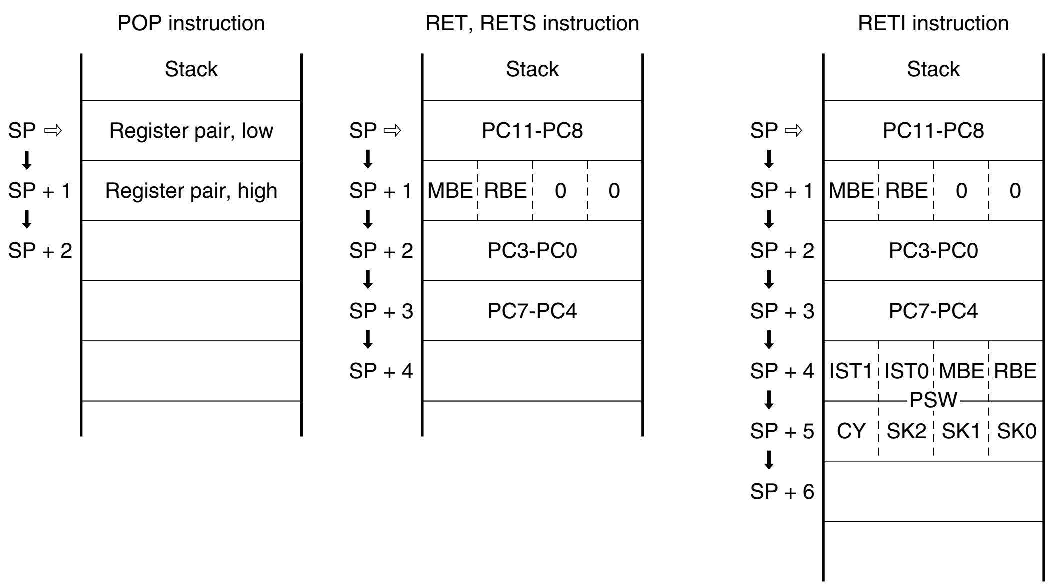


Figure 4-10. Data Restored from Stack Memory (MkI Mode)



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 4-11. Data Saved to Stack Memory (MkII Mode)

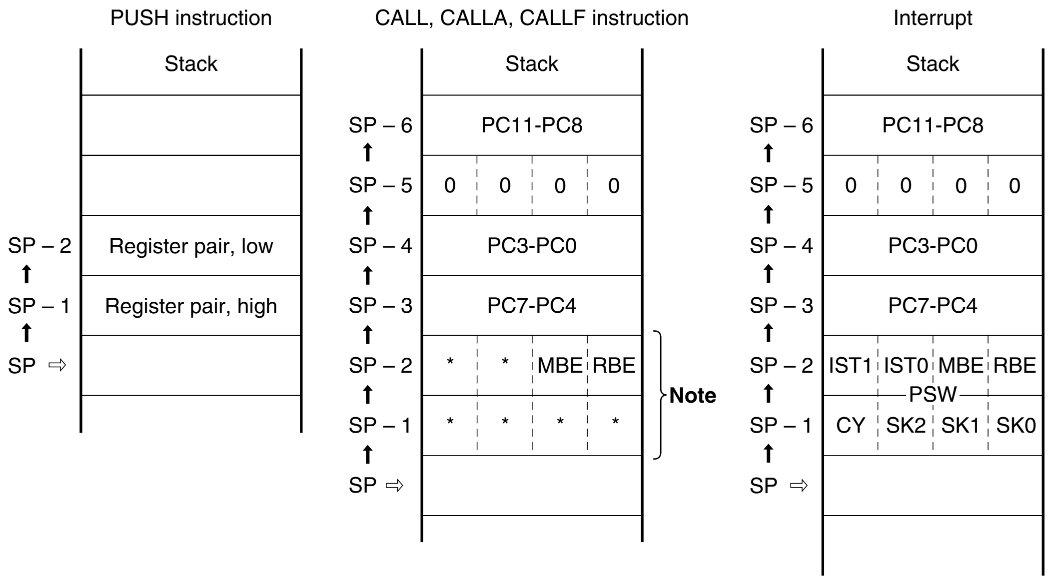
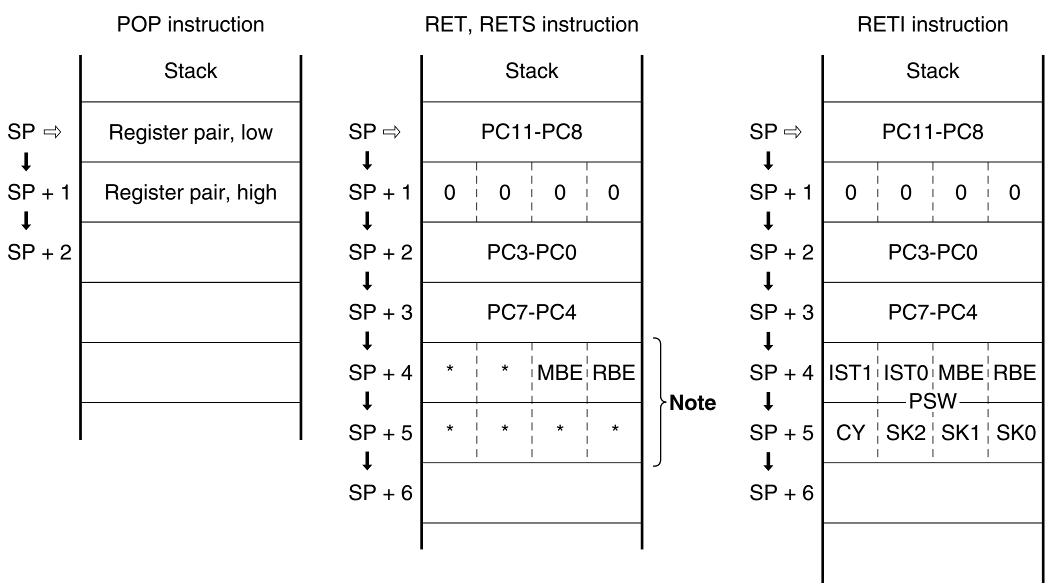


Figure 4-12. Data Restored from Stack Memory (MkII Mode)



Note The contents of PSW other than MBE and RBE are not saved or restored.

Remark *: Undefined

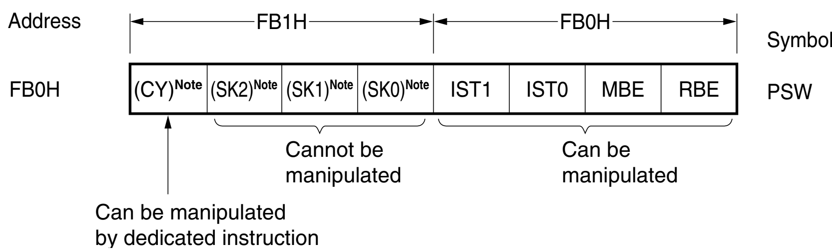
Download from Www.Somanuals.com. All Manuals Search And Download.

4.8 Program Status Word (PSW) ... 8 Bits

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

PSW is mapped to addresses FB0H and FB1H of the data memory space, and the 4 bits of address FB0H can be manipulated by using a memory manipulation instruction.

Figure 4-13. Configuration of Program Status Word



Note Not reserved as a reserved word.

Table 4-3. PSW Flags Saved/Restored to/from Stack

		Flag Saved or Restored
Save	When CALL, CALLA, or CALLF instruction is executed	MBE and RBE are saved
	When hardware interrupt occurs	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 records the occurrence of an overflow or underflow when an operation instruction with a carry (ADDC or SUBC) is executed.

The carry flag also functions as a bit accumulator and can store the result of a Boolean operation performed between a specified bit address and data memory.

The carry flag is manipulated by using a dedicated instruction and is independent of the other PSW bits.

The carry flag becomes undefined when the $\overline{\text{RESET}}$ signal is asserted.

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 4-4. Carry Flag Manipulation Instruction

	Instruction (Mnemonic)	Operation and Processing of Carry Flag
Carry flag manipulation instruction	SET1 CY	Sets CY to 1
	CLR1 CY	Clears CY to 0
	NOT1 CY	Inverts content of CY
	SKT CY	Skips if content of CY is 1
Bit transfer instruction	MOV1 mem*.bit, CY	Transfers content of CY to specified bit
	MOV1 CY, mem*.bit	Transfers content of specified bit to CY
Bit Boolean instruction	AND1 CY, mem*.bit	Takes ANDs, ORs, or XORs content of specified bit with content of CY and sets result to CY
	OR1 CY, mem*.bit	
	XOR1 CY, mem*.bit	
Interrupt service	In interrupt execution	Saved to stack memory in parallel with other PSW bits in 8-bit units
	RETI	Restored from stack memory with other PSW bits

Remark mem*.bit indicates the following three bit manipulation addressing modes.

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

Example To AND bit 3 at address 3FH with P33 and output result to P60

```
MOV     H, #3H          ; Sets higher 4 bits of address to H register
MOV1    CY, @H+0FH.3    ; CY ← bit 3 of 3FH
AND1    CY, PORT3.3     ; CY ← CY∧ P33
MOV1    PORT6.0, CY     ; P60 ← CY
```

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

The skip flags record the skip status, and are automatically set or reset when the CPU executes an instruction. These flags cannot be manipulated directly by the user as operands.

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Interrupt status flags (IST1 and IST0)

The interrupt status flags record the status of the processing under execution (for details, refer to **Table 7-3 IST, IST0, and Interrupt Servicing**).

Table 4-5. Contents of Interrupt Status Flags

IST1	IST0	Status of Processing Being Executed	Processing and Interrupt Control
0	0	Status 0	Normal program is being executed. All interrupts can be acknowledged
0	1	Status 1	Interrupt with lower or higher priority is serviced. Only an interrupt with higher priority can be acknowledged
1	0	Status 2	Interrupt with higher priority is serviced. All interrupts are disabled from being acknowledged
1	1	—	Setting prohibited

The interrupt priority controller (refer to **Figure 7-1 Block Diagram of Interrupt Control Circuit**) identifies the contents of these flags and controls the nesting of interrupts.

The contents of IST1 and 0 are saved to the stack along with the other bits of PSW when an interrupt is acknowledged, and the status is automatically updated by one. When the RETI instruction is executed, the values before the interrupt was acknowledged are restored to the interrupt status flags.

These flags can be manipulated by using a memory manipulation instruction, and the processing status under execution can be changed by program.

Caution To manipulate these flags, be sure to execute the DI instruction to disable the interrupts before manipulation. After manipulation, execute the EI instruction to enable the interrupts.

(4) Memory bank enable flag (MBE)

This flag specifies the address information generation mode of the higher 4 bits of the 12 bits of a data memory address.

MBE can be set or reset at any time by using a bit manipulation instruction, regardless of the setting of the memory bank.

When this flag is set to “1”, the data memory address space is expanded, and the entire data memory space can be addressed.

When MBE is reset to “0”, the data memory address space is fixed, regardless of MBS (refer to **Figure 3-2 Configuration of Data Memory and Addressing Ranges of Respective Addressing Modes**).

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

When a vector interrupt is serviced, bit 7 of the corresponding vector address table is set. Also, the status of MBE when the interrupt is serviced is automatically set.

Usually, MBE is reset to 0 for interrupt servicing, and the static RAM in memory bank 0 is used.

Download from Www.Somanuals.com. All Manuals Search And Download.

(5) Register bank enable flag (RBE)

This flag specifies whether the register bank of the general-purpose registers is expanded or not.

RBE can be set or reset at any time by using a bit manipulation instruction, regardless of the setting of the memory bank.

When this flag is set to “1”, one of four general-purpose register banks 0 to 3 can be selected depending on the contents of the register bank select register (RBS).

When RBE is reset to “0”, register bank 0 is always selected, regardless of the contents of the register bank select register (RBS).

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

When a vector interrupt occurs, the contents of bit 6 of the corresponding vector address table are set to RBE. Also, the status of RBE when the interrupt is serviced is automatically set. Usually, RBE is reset to 0 during interrupt servicing. Register bank 0 is selected for 4-bit processing, and register banks 0 and 1 are selected for 8-bit processing.

Download from Www.Somanuals.com. All Manuals Search And Download.

4.9 Bank Select Register (BS)

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

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

BS can be saved to or restored from the stack area in 8-bit units by the PUSH BS or POP BS instruction.

Figure 4-14. Configuration of Bank Select Register

Address	F83H				F82H				Symbol
F82H	MBS3	MBS2	MBS1	MBS0	0	0	RBS1	RBS0	BS

(1) Memory bank select register (MBS)

The memory bank select register is a 4-bit register that records the higher 4 bits of a 12-bit data memory address. This register specifies the memory bank to be accessed. With the μPD754244, however, only banks 0, 4 and 15 can be specified.

MBS is set by the SEL MBn instruction (n = 0, 4, 15).

The address range specified by MBE and MBS is as shown in Figure 3-2.

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

Table 4-6. MBE, MBS, and Memory Bank Selected

MBE	MBS				Memory Bank
	3	2	1	0	
0	×	×	×	×	Fixed to memory bank 0
1	0	0	0	0	Selects memory bank 0
	0	1	0	0	Selects memory bank 4
	1	1	1	1	Selects memory bank 15
Other than above					Setting prohibited

× = don’t care

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Register bank select register (RBS)

The register bank select register specifies a register bank to be used as general-purpose registers. It can select bank 0 to 3.

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

When the $\overline{\text{RESET}}$ signal is asserted, RBS is initialized to “0”.

Table 4-7. RBE, RBS, and Register Bank Selected

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

(RBS bits 3 and 2: Fixed to 0)

× = don’t care

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 5 EEPROM

The μPD754244 incorporates not only a 128-word × 4-bit static RAM but also a 16-word × 8-bit EEPROM (Electrically Erasable PROM) as data memory.

EEPROM, unlike static RAM, can retain its contents when the power is turned off.

Unlike EPROM, contents can electrically be erased without using ultraviolet rays.

It is, therefore, suitable for application fields such as keyless entry and as a data carrier.

EEPROM is mapped onto memory bank 4 of the data memory. EEPROM is manipulated by an 8-bit memory manipulation instruction.

5.1 EEPROM Configuration

EEPROM consists of the EEPROM main unit at memory bank 4 of the data memory and the EEPROM control block.

The EEPROM control block consists of an EEPROM write control register (EWC) that controls manipulation of EEPROM and a block that detects write completion and generates an interrupt signal.

5.2 EEPROM Features

(1) Once data is written to EEPROM, it can retain the contents, even if the power is turned off.

(2) As with the static RAM, manipulation (automatic erasure/automatic writing) is possible using an 8-bit memory manipulation instruction.
However, there are restrictions on the instructions that can be executed. Refer to **5.5.1 EEPROM manipulation instructions**.

(3) EEPROM performs automatic erasure/automatic writing within the time set by the dedicated EEPROM write timer clock selection bit (EWTC). Therefore, the load on the software that controls the write time can be alleviated.
- Write time ··· Set EWTC4 to EWTC6 so that the write time is as follows.
  With μPD754144 ··· $18 \times 2^8/f_{CC}$ (4.6 ms: f_{CC} = 1.0 MHz)
  With μPD754244 ··· 4.0 ms MIN., 10.0 ms MAX.
  Clear the EEPROM write enable/disable control bit (EWE) to 0 after writing.
  Any instruction other than one related to EEPROM writing can be executed even in an EEPROM write operation.
- Number of writes
  T_A = –40 to +70°C ··· 100,000 times/byte
  T_A = –40 to +85°C ··· 80,000 times/byte

(4) When writing is completed, an EEPROM write end interrupt is generated.

(5) EEPROM can independently check whether writing is possible or not using the write status flag. A bit manipulation instruction is used for this check (refer to **5.3 EEPROM Write Control Register (EWC)**.

Download from Www.Somanuals.com. All Manuals Search And Download.

5.3 EEPROM Write Control Register (EWC)

The EEPROM write control register (EWC) is an 8-bit register used to control manipulation of EEPROM. Figure 5-1 shows its configuration.

Figure 5-1. Format of EEPROM Write Control Register

Address	7	6	5	4	3	2	1	0	Symbol
FCEH	ERE	EWTC6	EWTC5	EWTC4	EWE	EWST	–	–	EWC

EEPROM read enable flag

ERE	EEPROM
0	EEPROM read disabled (suppresses current)
1	EEPROM read enabled (after 15 μs)

Dedicated EEPROM write timer clock selection bit

EWTC6	EWTC5	EWTC4	Selection of count clock
0	0	0	$18 \times 2^{13}/f_X$
0	0	1	$18 \times 2^{12}/f_X$
0	1	0	$18 \times 2^{11}/f_X$
0	1	1	$18 \times 2^{10}/f_X$
1	0	0	$18 \times 2^{9}/f_X$
1	0	1	$18 \times 2^{8}/f_X$
Other than above			Setting prohibited

f_X = system clock oscillation frequency

EEPROM write enable/disable control bit

EWE	EEPROM write operation
0	Disable
1	Enable

EEPROM write status flag

EWST	Write status
0	EEPROM write enable
1	EEPROM being written (EEPROM writing is not possible. If writing is attempted, it is ignored.)

Download from Www.Somanuals.com. All Manuals Search And Download.

Cautions 1. The write time depends on the system clock oscillation frequency.

2. Set EWTC4-EWTC6 so that the write time is as follows.
With μPD754144 ··· 18 × 2^8/fcc (4.6 ms: fcc = 1.0 MHz)
With μPD754244 ··· 4.0 ms MIN., 10.0 ms MAX.
Clear EWE to 0 after writing.

3. Be sure to clear (0) the ERE flag before executing a STOP instruction to disable reading. If the ERE flag is set (1), a current of approximately 10 μA always flows in the read circuit. Therefore, be sure to clear (0) the ERE flag before executing a STOP instruction to stop the current supply to the read circuit.

4. Be sure to clear (0) the EWE flag before executing a STOP instruction to disable writing.

EWC is set by an 8-bit memory manipulation instruction.

Bits 4 to 6 of EWC are the dedicated EEPROM write timer clock selection bits (EWTC).

EWTC sets the count clock when EEPROM automatic erasure/automatic writing is performed. EEPROM performs automatic erasure/automatic writing for each time set by EWTC.

Bit 2 of EWC is a write status flag (EWST). This flag can be used to check in 1-bit units whether writing is currently performed or writing is possible. When writing is started, EWST is automatically write disabled (1). A bit memory manipulation instruction is used to check this.

$\overline{\text{RESET}}$ input clears all EWC bits to 0.

Example EEPROM is write enabled and the write time is set to 18 × 2^8/fx.

```
SEL   MB15
MOV   XA, #01011000B
MOV   EWC, XA
```

5.4 Interrupt Related to EEPROM Control

Table 5-5 shows the interrupt related to EEPROM control.

For the details of the interrupt function, refer to **CHAPTER 7 INTERRUPT AND TEST FUNCTIONS.**

Table 5-1. Interrupt Related to EEPROM Control

Interrupt Source	EEPROM Interrupt Request Flag	EEPROM Interrupt Request Flag	Vector Table Address	Interrupt Request Flag Setting Source
INTEE (EEPROM write end interrupt)	IRQEE	IEEE	VRQ7 (000EH)	When the write time set by EWC has elapsed.

Caution The INTOW interrupt (EEPROM overwrite interrupt) used in the μPD75048 is not provided.

Download from Www.Somanuals.com. All Manuals Search And Download.

5.5 EEPROM Manipulation Method

5.5.1 EEPROM manipulation instructions

Instructions that can be used to manipulate the EEPROM are shown below, divided into read instructions and write instructions.

(1) Read manipulation instructions

Instruction Group	Mnemonic	Operand
Transfer instruction	MOV MOV	XA, @HL XA, mem
Compare instruction	SKE	XA, @HL

Remark Operation instruction such as ADDS, AND, etc., cannot be used.

(2) Write manipulation instructions

Instruction Group	Mnemonic	Operand
Transfer instruction	MOV	@HL, XA
	MOV	mem, XA
	XCH	XA, @HL
	XCH	XA, mem

Remark INCS (increment/decrement instruction) cannot be used.

An 8-bit memory manipulation instruction is used to manipulate EEPROM. Furthermore, a bit memory manipulation instruction can be used to check EWST.

A 4-bit memory manipulation instruction cannot be used.

Download from Www.Somanuals.com. All Manuals Search And Download.

5.5.2 Read manipulation

The following procedure is used to read EEPROM.

EWST, ERE and EWE can be set simultaneously by an 8-bit memory manipulation instruction to EWC.

<1> Check that the write status flag (EWST) is 0 (write enabled = writing is currently not being performed).
<2> Set the write enable/disable control bit (EWE) to 0 (write disabled).
<3> Execute the read instruction.

Figure 5-2. EEPROM Write Control Register in EEPROM Read Manipulation

Address	7	6	5	4	3	2	1	0	Symbol
FCEH	ERE	EWTC6	EWTC5	EWTC4	EWE	EWST	–	–	EWC

Operating mode selection bit

ERE	EWE	EWST	Mode
1	0	0	EEPROM read enable mode

Cautions 1. Be sure to check that EWST is 0 before reading. If an EEPROM read instruction is executed during an EEPROM write operation, the value read becomes undefined.

2. There are restrictions on the read instruction. Refer to 5.5.1 EEPROM manipulation instructions for details.

3. Setting ERE to 1 enables EEPROM read and increases the current consumption. Therefore, set ERE to 0 when EEPROM is not being read.

4. Execute the read instruction approximately 15 μs or more after setting ERE.

5. Setting EWE to 1 enables EEPROM write and increases the current consumption. Therefore, set EWE to 0 when EEPROM is not being written to.

Example After checking the write status flag (EWST), 8-bit data (0A, 0BH of memory bank 4) is read.

```
SET1    MBE
SEL     MB15
SKF     EWST
BR      A2
SEL     MB4
MOV     XA, #0AH
MOV     HL, @HL
```


Download from Www.Somanuals.com. All Manuals Search And Download.

5.5.3 Write manipulation

Use the following procedure to write to EEPROM.

Any instruction other than one related to EEPROM writing can be executed even during an EEPROM write operation.

EWST, EWTC and EWE can be set simultaneously by an 8-bit memory manipulation instruction to EWC.

<1> Check that the read status (EWST) is 0 (write enabled = currently not being written).
<2> Use EWTC4 to EWTC6 to set write time.
<3> Set the write enable/disable control bit (EWE) to 1 (write enabled).
<4> Execute the write instruction.

Figure 5-3. EEPROM Write Control Register in EEPROM Write Manipulation

Address	7	6	5	4	3	2	1	0	Symbol
FCEH	ERE	EWTC6	EWTC5	EWTC4	EWE	EWST	–	–	EWC

Operating mode selection bit

ERE	EWE	EWST	Mode
0	1	0	EEPROM write enabled mode

Dedicated EEPROM write timer clock selection bit

EWTC6	EWTC5	EWTC4	Count clock selection
0	0	0	$18 \times 2^{13}/f_X$
0	0	1	$18 \times 2^{12}/f_X$
0	1	0	$18 \times 2^{11}/f_X$
0	1	1	$18 \times 2^{10}/f_X$
1	0	0	$18 \times 2^{9}/f_X$
1	0	1	$18 \times 2^{8}/f_X$
Other than above			Setting prohibited

f_X = System clock oscillation frequency

Download from Www.Somanuals.com. All Manuals Search And Download.

Example Set the write time to 18 × 2^8/fx and after checking the EEPROM write status flag (EWST), write 8-bit data (0AH) at 08H of memory bank 4.

```
        SET1   MBE
        SEL    MB15                  ; Selection of bank 15
        MOV    XA, #01011000B        ; Write enable
        MOV    EWC, XA               ; Set the write time to 18 × 2^8/fx
        SKF    EWST
        BR     A1
        SEL    MB4
        MOV    XA, #0AH
        MOV    08H, XA               ; Write
        CLR1   MBE   ┐
WAIT;   SKF    EWST  │  <A>
        BR     WAIT  ┘
        CLR1   EWE
```

Caution **The development tool simulates writing data to EEPROM by writing the data to RAM. Therefore, it seems as if EEPROM was normally written even if the wait time <A> is not long enough. However, data cannot be written correctly to the device unless the wait time <A> is long enough (4.0 ms MIN., 10.0 ms MAX.) (the contents written to the EEPROM are undefined). After writing the data, clear EWE to 0.**

Cautions on EEPROM writing are shown below. Be sure to read them before writing to EEPROM. When performing consecutive writing, write once after the current write operation is finished. Set EWTC4 to EWTC6 so that data can be written to the EEPROM once within the following time.

- With μPD754144 ... 18 × 2^8/fcc (4.6 ms: fcc = 1.0 MHz)
- With μPD754244 ... 4.0 ms MIN., 10.0 ms MAX.

Clear EWE to 0 after writing.

Writing can be completed and time can be managed in the following ways.

(1) Using write end interrupt

After one data item is written, wait for generation of a write end interrupt, while performing processing other than writing. When a write end interrupt is generated, start the next write operation.

(2) Using write status flag

Execute polling on the write status flag and wait until it becomes 0.

When the write status flag becomes 0, it indicates the write end, and so start the next write operation.

(3) Using timer

Use the timer counter or basic interval timer to wait[Note] for the write time set by EWTC4 to EWTC6.

(4) Using software

Use the software timer to wait[Note] for the write time set by EWTC4 to EWTC6.

Note Make sure that a wait time longer than the write time specified by EWTC4 to EWTC6 elapses. If EWE is cleared within the time set by EWTC4 to EWTC6, the contents written to the EEPROM are undefined.

Download from Www.Somanuals.com. All Manuals Search And Download.

5.6 Cautions on EEPROM Writing

Cautions on EEPROM writing are shown below.
Be sure to read these before writing to EEPROM.

Cautions

1. Before writing, make sure that EWST is 0. While EEPROM is being written, if a write instruction is executed again, the instruction executed later is ignored.

2. There are restrictions on the write instruction. Refer to 5.5.1 EEPROM manipulation instructions for details.

3. Set EWTC4 to EWTC6 so that the write time is as follows.
With μPD754144 ... 18 $\times$ 2^8/fcc (4.6 ms: fcc = 1.0 MHz)
With μPD754244 ... 4.0 ms MIN., 10.0 ms MAX.
Clear EWE to 0 after writing.

4. When performing write operations consecutively, be sure to wait until the current write is finished before executing the next one.

5. Even if the HALT mode is set while EEPROM is being written, the write operation continues. However, hardware which is stopped by the CPU or in HALT mode stops.
Be careful about how you control the write time.

6. If STOP mode is set during an EEPROM operation, writing is stopped.
The address data being written becomes undefined.

7. If writing is disabled by EWE during an EEPROM write operation, writing is stopped.
The address data being written becomes 0.

8. The μPD754244 is shipped with the EEPROM contents set to 0.

9. Setting EWE to 1 enables EEPROM writing and increases the current consumption.
Therefore, set EWE to 0 when EEPROM writing is not being performed.

10. Setting ERE to 1 enables EEPROM reading and increases the current consumption.
Therefore, set ERE to 0 when EEPROM reading is not being performed.

11. The development tool simulates writing data to EEPROM by writing the data to RAM. Therefore, it seems as if EEPROM was normally written even if the write time is not long enough. However, data cannot be written correctly to the device unless the write time is long enough (4.0 ms MIN., 10.0 ms MAX.) (the contents written to the EEPROM are undefined). After writing the data, clear EWE to 0.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.1 Digital I/O Ports

The μPD754244 uses memory mapped I/O, and all the I/O ports are mapped to the data memory space.

Figure 6-1. Data Memory Address of Digital Ports

Address	3	2	1	0	
FF0H	—				
FF1H	—				
FF2H	—				
FF3H	P33	P32	P31	P30	Port 3
FF4H	—				
FF5H	—				
FF6H	P63	P62	P61	P60	Port 6
FF7H	P73	P72	P71	P70	Port 7
FF8H	–	–	–	P80	Port 8

Table 6-2 lists the instructions that manipulate the I/O ports. Ports 3 and 6 can be manipulated in 4-I/O and 1-bit units. They are used for various control operations.

Examples 1. To test the status of P73 and outputs different values to port 3 depending on the result

```
SKT   PORT7.3    ; Skips if bit 3 of port 7 is 1
MOV   XA, #8H    ; XA ← 8H  } String effect
MOV   XA, #4H    ; XA ← 4H  }
SEL   MB15       ; or CLR1 MBE
OUT   PORT3, A   ; Port 3 ← A
```

2.
```
SET1  PORT6.@L   ; Sets the bits of port 6 specified by the L register to "1"
```


Download from Www.Somanuals.com. All Manuals Search And Download.

6.1.1 Types, features, and configurations of digital I/O ports

Table 6-1 shows the types of digital I/O ports.

Figures 6-2 to 6-9 show the configuration of each port.

Table 6-1. Types and Features of Digital Ports

Port	Function	Operation and Features	Remarks
PORT3	4-bit I/O	Can be set to input or output mode in 1-bit units.	Also used for PTO0 to PTO2 pins.
PORT6			Also used for AVREF, INT0, PTH00, and PTH01 pins.
PORT7	4-bit input	4-bit input only port On-chip pull-up resistor can be specified by mask option in 1-bits units.	Also used for KR4 to KR7 pins.
PORT8	1-bit I/O	Can be set to input or output mode in 1-bit units.	–

P61 is shared with an external vector interrupt input pin and a noise eliminator is selectable (for details, refer to **7.3 Hardware Controlling Interrupt Function**).

When the $\overline{\text{RESET}}$ signal is asserted, the output latches of ports 3, 6, and 8 are cleared to 0, the output buffers are turned off, and the ports are set to the input mode.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-2. P3n Configuration (n = 0 to 2)

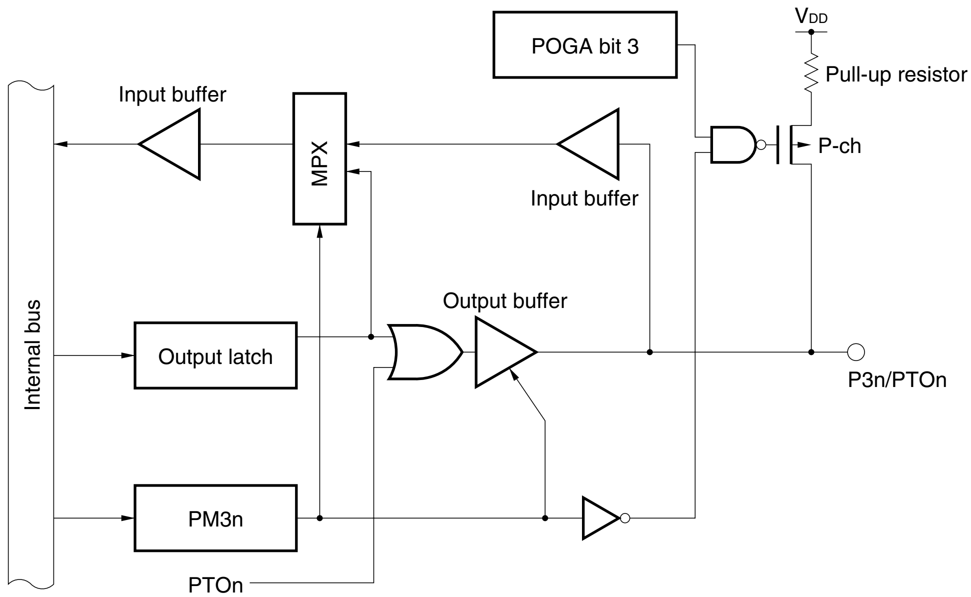
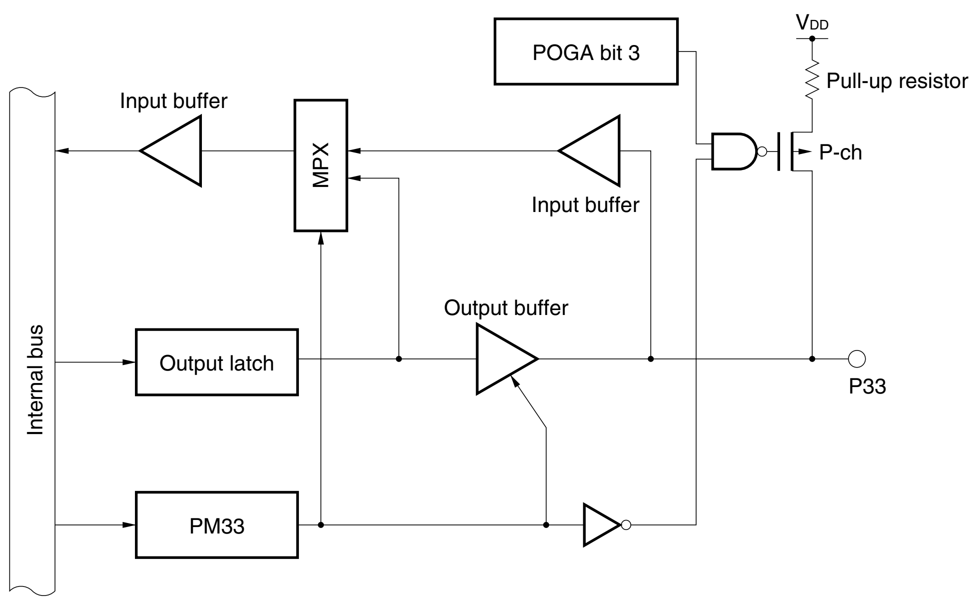


Figure 6-3. P33 Configuration



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-4. P60 Configuration

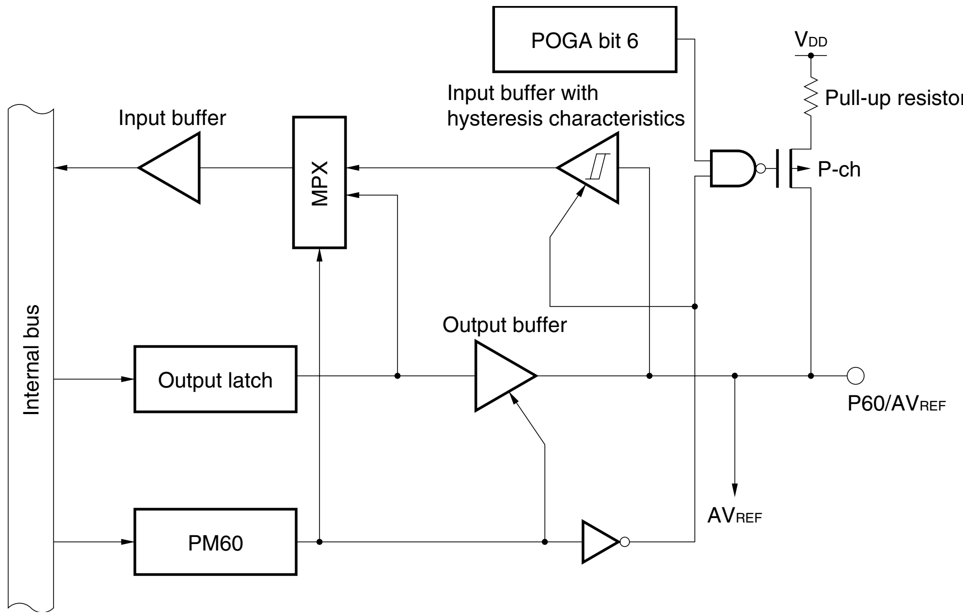
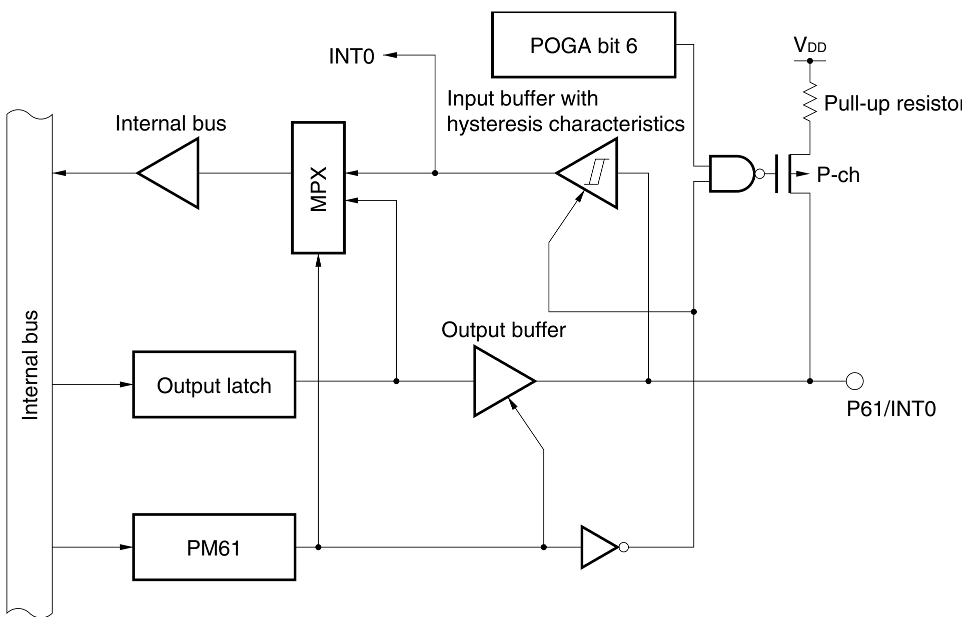


Figure 6-5. P61 Configuration



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-6. P62 Configuration

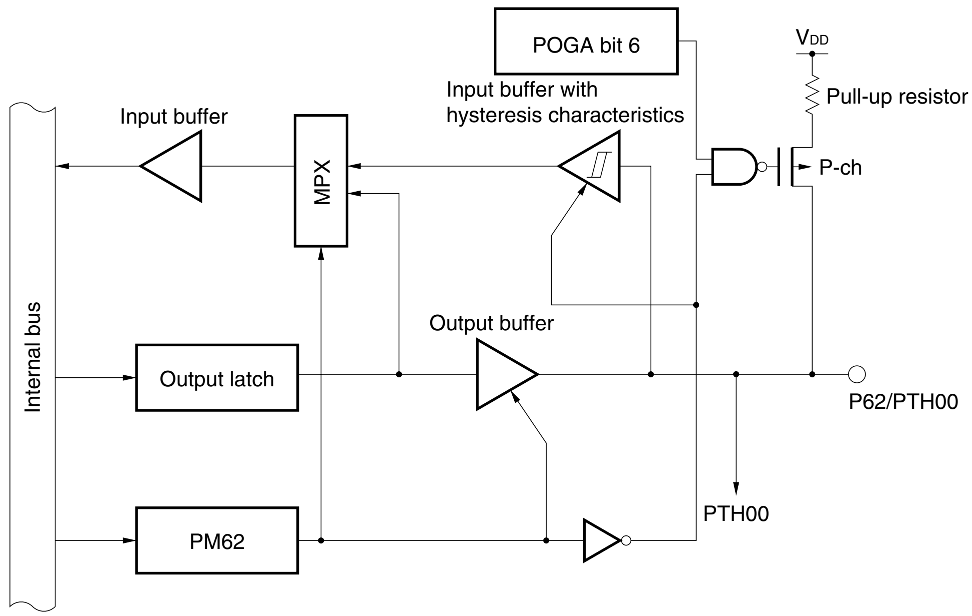
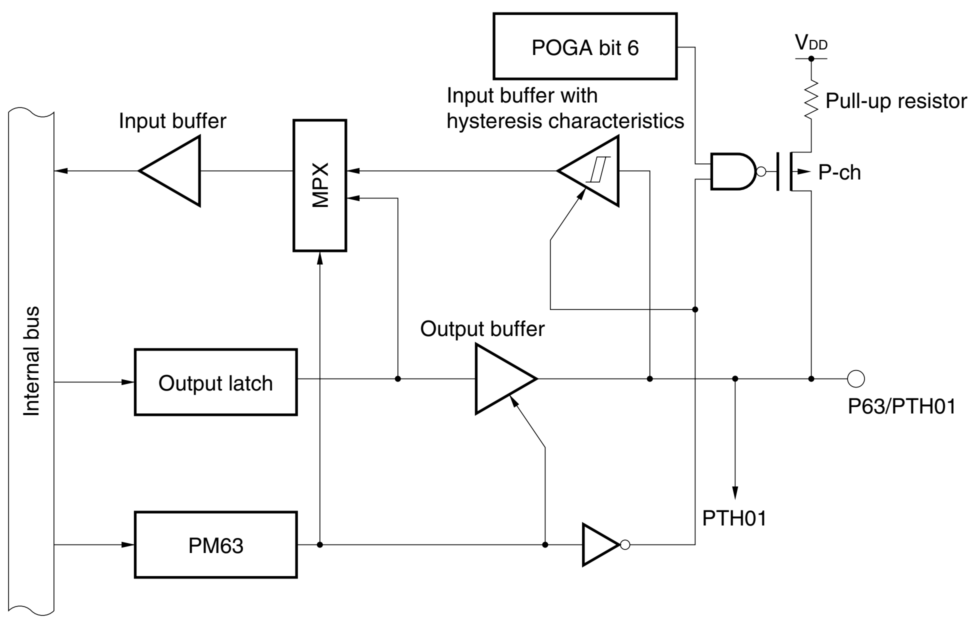


Figure 6-7. P63 Configuration



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-8. P7n Configuration (n = 0 to 3)

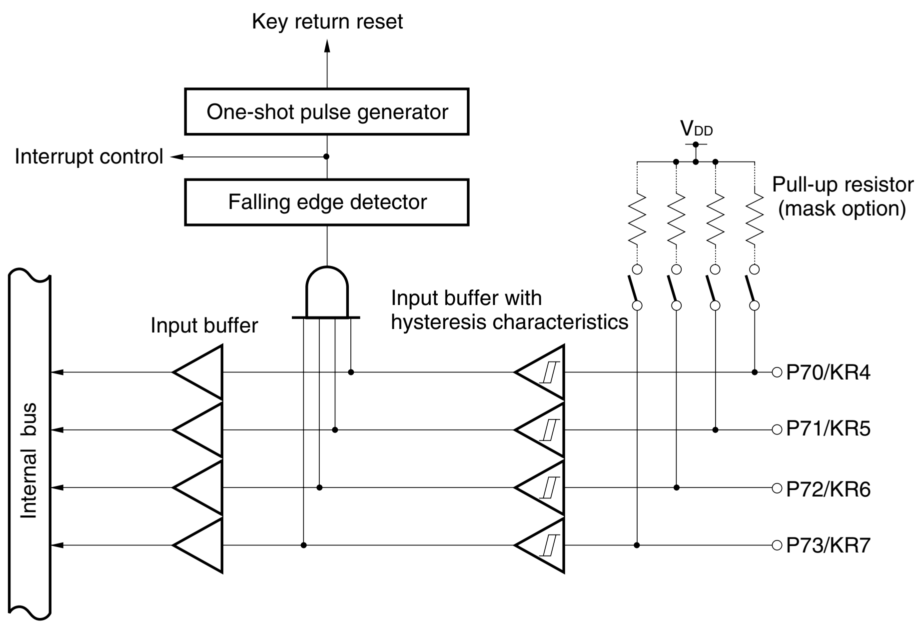
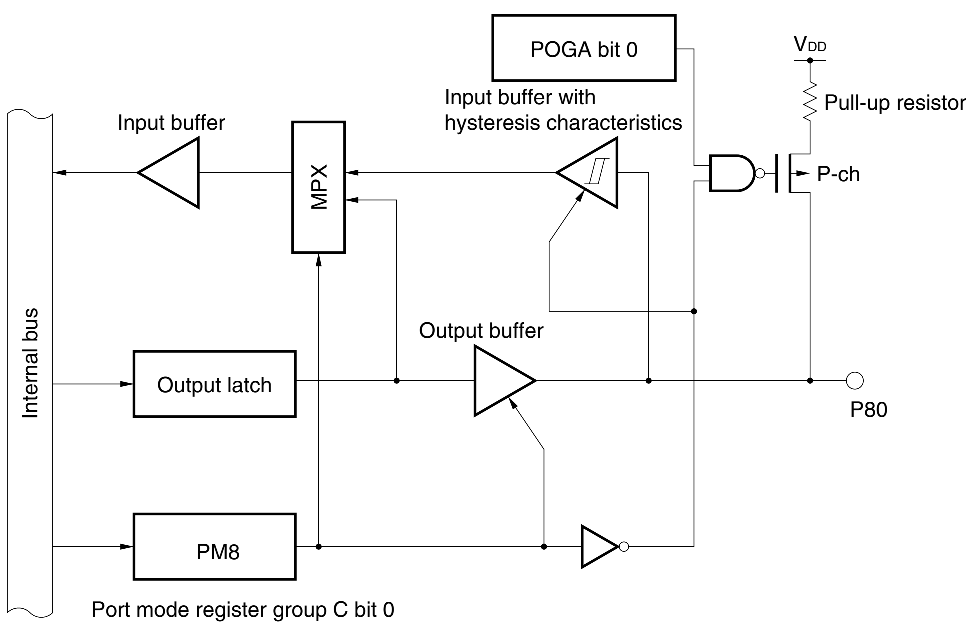


Figure 6-9. P80 Configuration



Download from Www.Somanuals.com. All Manuals Search And Download.

6.1.2 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 6-10. Ports 3 and 6 can be set to the input or output mode in 1-bit units by using port mode register group A (PMGA). Port 8 is set to the input or output mode by using port mode register group C (PMGC).

Each port is set to 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 to 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 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, the output buffer is turned off, and the corresponding port is set to 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
```


Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-10. Format of Each Port Mode Register

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

Port mode register group A

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

- PM30: Sets P30 to input or output mode
- PM31: Sets P31 to input or output mode
- PM32: Sets P32 to input or output mode
- PM33: Sets P33 to input or output mode
- PM60: Sets P60 to input or output mode
- PM61: Sets P61 to input or output mode
- PM62: Sets P62 to input or output mode
- PM63: Sets P63 to input or output mode

Port mode register group C

Address	7	6	5	4	3	2	1	0	Symbol
FEEH	–	–	–	–	–	–	–	PM8	PMGC

- PM8: Sets port 8 (P80) to input or output mode

Download from Www.Somanuals.com. All Manuals Search And Download.

6.1.3 Digital I/O port manipulation instruction

Because all the I/O ports of the μPD754244 are mapped to the data memory space, they can be manipulated by using data memory manipulation instructions. Table 6-2 shows these data memory manipulation instructions, which are considered to be especially useful for manipulating the I/O pins and their range of applications.

(1) Bit manipulation instruction

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

Example To OR P30 and P61 and output to P80

```
MOV1    CY, PORT3.0;  CY ← P30
OR1     CY, PORT6.1;  CY ← CY∨ P61
MOV1    PORT8.0, CY;  P80 ← CY
```

(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

```
SET1    MBE
SEL     MB15        ;  or CLR1 MBE
OUT     PORT3, A
```

2. To add the value of the accumulator to the data output to port 6

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT6
ADDS    A, @HL      ;  A ← A+PORT6
NOP
MOV     @HL, A      ;  PORT6 ← A
```

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

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT3
SUBS    A, @HL      ;  A<PORT3
BR      NO          ;  NO
                    ;  YES
```


Download from Www.Somanuals.com. All Manuals Search And Download.

Table 6-2. I/O Pin Manipulation Instructions

<table>
<tr><th>Instruction \ PORT</th><th>PORT3</th><th>PORT6</th><th>PORT7</th><th>PORT8</th></tr>
<tr><td>IN A, PORTn[Note 1]</td><td colspan="4">○</td></tr>
<tr><td>IN XA, PORTn[Note 1]</td><td colspan="4">–</td></tr>
<tr><td>OUT PORTn, A[Note 1]</td><td colspan="2">○</td><td>–</td><td>○</td></tr>
<tr><td>OUT PORTn, XA[Note 1]</td><td colspan="4">–</td></tr>
<tr><td>MOV A, PORTn[Note 1]</td><td colspan="4">○</td></tr>
<tr><td>MOV XA, PORTn[Note 1]</td><td colspan="4">–</td></tr>
<tr><td>MOV PORTn, A[Note 1]</td><td colspan="4">○</td></tr>
<tr><td>MOV PORTn, XA[Note 1]</td><td colspan="4">–</td></tr>
<tr><td>XCH A, PORTn[Note 1]</td><td colspan="4">○</td></tr>
<tr><td>XCH XA, PORTn[Note 1]</td><td colspan="4">–</td></tr>
<tr><td>MOV1 CY, PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>MOV1 CY, PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>MOV1 PORTn. bit, CY</td><td colspan="2">○</td><td>–</td><td>○</td></tr>
<tr><td>MOV1 PORTn. @L, CY[Note 2]</td><td colspan="2">○</td><td>–</td><td>○</td></tr>
<tr><td>INCS PORTn[Note 1]</td><td colspan="4">○</td></tr>
<tr><td>SET1 PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>SET1 PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>CLR1 PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>CLR1 PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>SKT PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>SKT PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>SKF PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>SKTCLR PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>SKTCLR PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>SKF PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>AND1 CY, PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>AND1 CY, PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>OR1 CY, PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>OR1 CY, PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
<tr><td>XOR1 CY, PORTn. bit</td><td colspan="4">○</td></tr>
<tr><td>XOR1 CY, PORTn. @L[Note 2]</td><td colspan="4">○</td></tr>
</table>

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

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

Download from Www.Somanuals.com. All Manuals Search And Download.

6.1.4 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 differ depending on whether the port is set to the input or output mode (refer to **Table 6-3**). 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 in 4-bit units such as IN, MOV, operation, or comparison instructions, is executed, the data of each pin is manipulated.

When an instruction that transfers the contents of the accumulator in 4-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 (4 bits) of each pin incremented by one (+1) 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 in 4-bit units 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-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. 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 6-3. Operation When I/O Port Is Manipulated

Instruction Executed	Operation of Port and Pin	
	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
MOV A, 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
MOV PORTn, A
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 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 PORT 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. However, 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.1.5 Connecting pull-up resistor

Each port pin of the μPD754244 can be connected to a pull-up resistor. Some pins can be connected to a pull-up resistor via software and others can be connected by a mask option.

Table 6-4 shows how to specify the connection of the pull-up resistor to each port pin. The pull-up resistor is connected via software in the format shown in Figure 6-11.

The pull-up resistor can be connected only to the pins of ports 3, 6, and 8 in the input mode. When the pins are set to the output mode, the pull-up resistor cannot be connected regardless of the setting of POGA and POGB.

Table 6-4. Specifying Connection of Pull-up Resistor

Port (Pin Name)	Specifying Connection of Pull-up Resistor	Specified Bit
Port 3 (P30-P33)	Connection of pull-up resistor specified in 4-bit units via software	POGA.3
Port 6 (P60-P63)		POGA.6
Port 7 (P70-P73)	Connection of pull-up resistor specified in 1-bit units by mask option	—
Port 8 (P80-P83)	Connection of pull-up resistor specified in 1-bit units via software	POGB.0

Figure 6-11. Format of Pull-up Resistor Specification Register

	Specification
0	Pull-up resistor not connected
1	Pull-up resistor connected

Pull-up resistor specification register group A

Address	7	6	5	4	3	2	1	0	Symbol
FDCH	–	PO6	–	–	PO3	–	–	–	POGA

PO3: Port 3 (P30 to P33)

PO6: Port 6 (P60 to P63)

Pull-up resistor specification register group B

Address	7	6	5	4	3	2	1	0	Symbol
FDEH	–	–	–	–	–	–	–	PO8	POGB

PO8: Port 8 (P80)

Download from Www.Somanuals.com. All Manuals Search And Download.

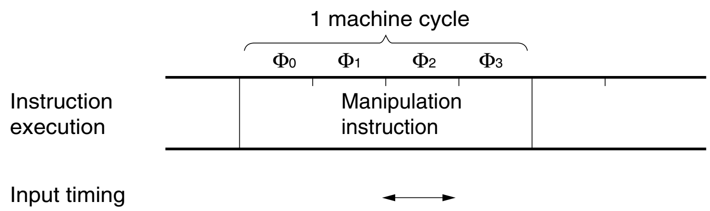
6.1.6 I/O timing of digital I/O port

Figure 6-12 shows the timing at which data is output to the output latch and the timing at which the pin data or the data of the output latch is loaded to the internal bus.

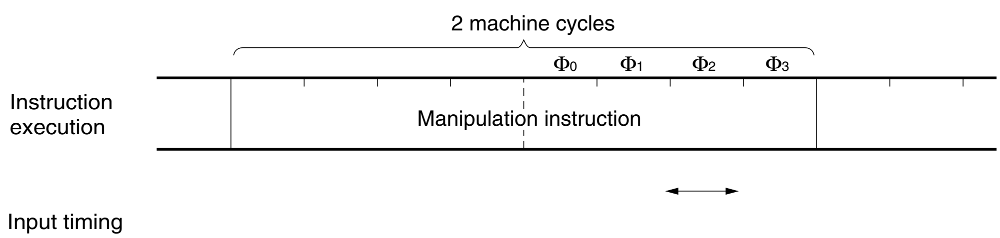
Figure 6-13 shows the ON timing when an on-chip pull-up resistor connection is specified via software.

Figure 6-12. I/O Timing of Digital I/O Port

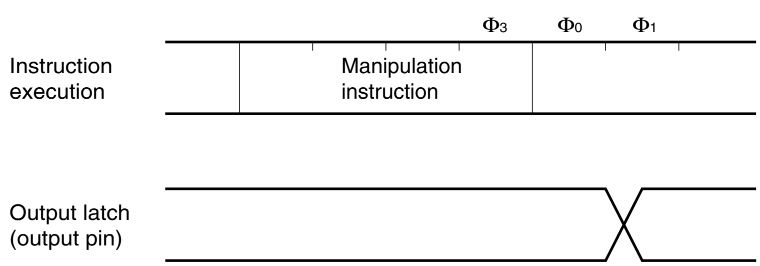
(a) When data is loaded by 1-machine-cycle instruction



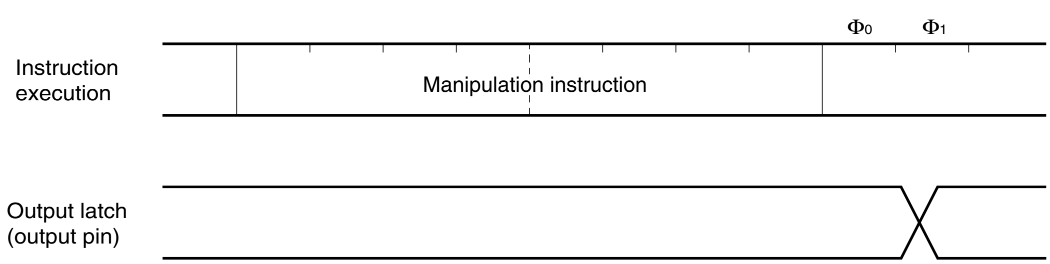
(b) When data is loaded by 2-machine-cycle instruction



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

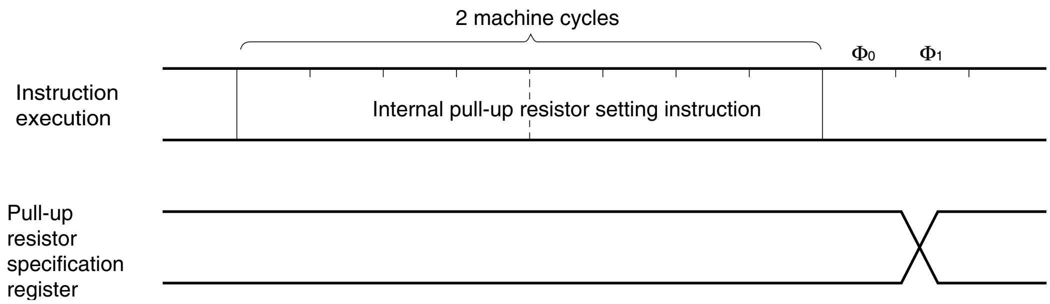


(d) When data is latched by 2-machine-cycle instruction



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-13. ON Timing of Internal Pull-up Resistor Connected via Software



Download from Www.Somanuals.com. All Manuals Search And Download.

6.2 Clock Generator

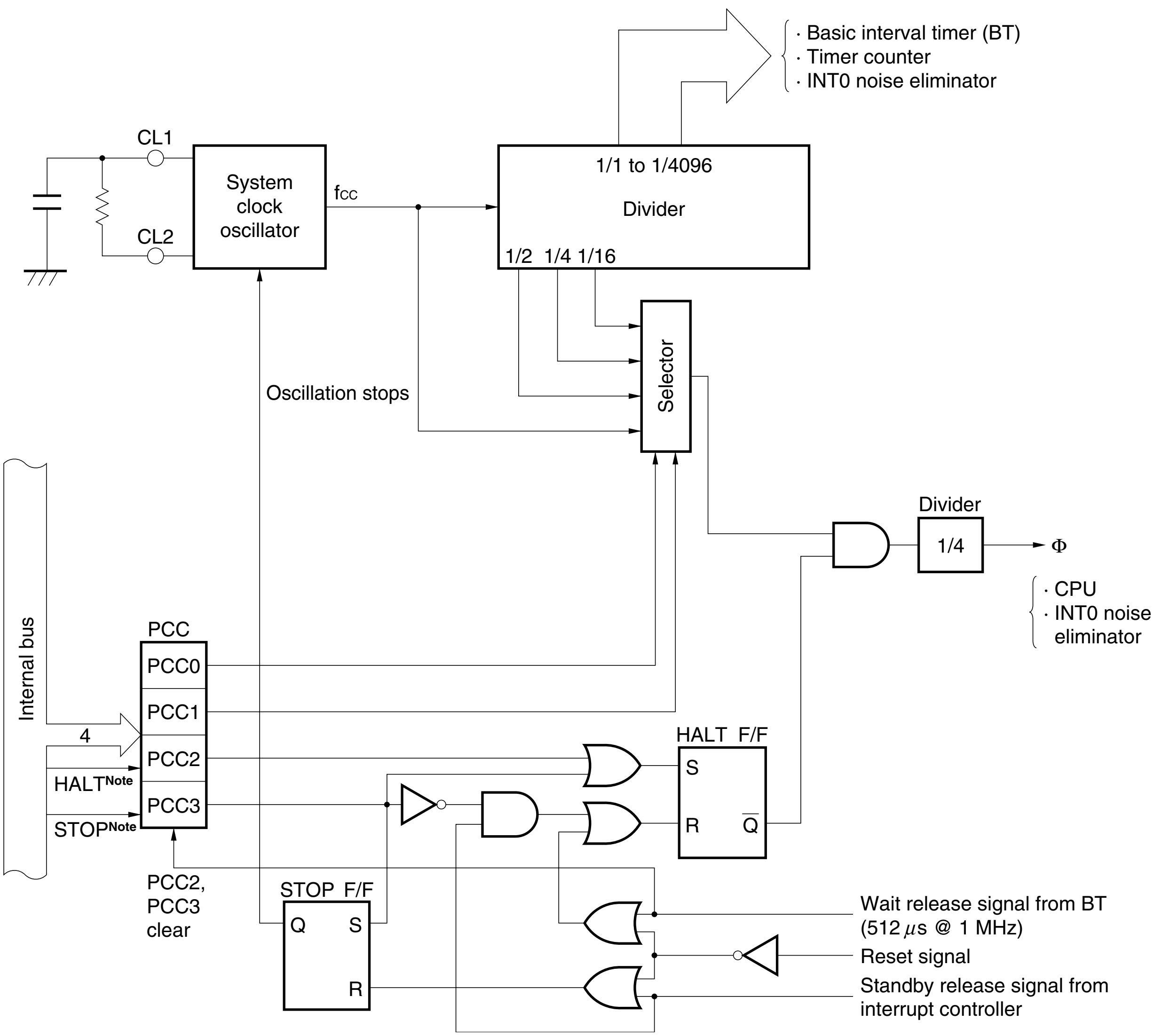
The clock generator supplies various clocks to the CPU and peripheral hardware units and controls the operation mode of the CPU.

6.2.1 Configuration of clock generator

Figure 6-14 shows the configuration of the clock generator.

Figure 6-14. Block Diagram of Clock Generator (1/2)

(a) μPD754144 (RC oscillation)



Note Instruction execution

Remarks 1. fCC: System clock frequency

2. Φ = CPU clock

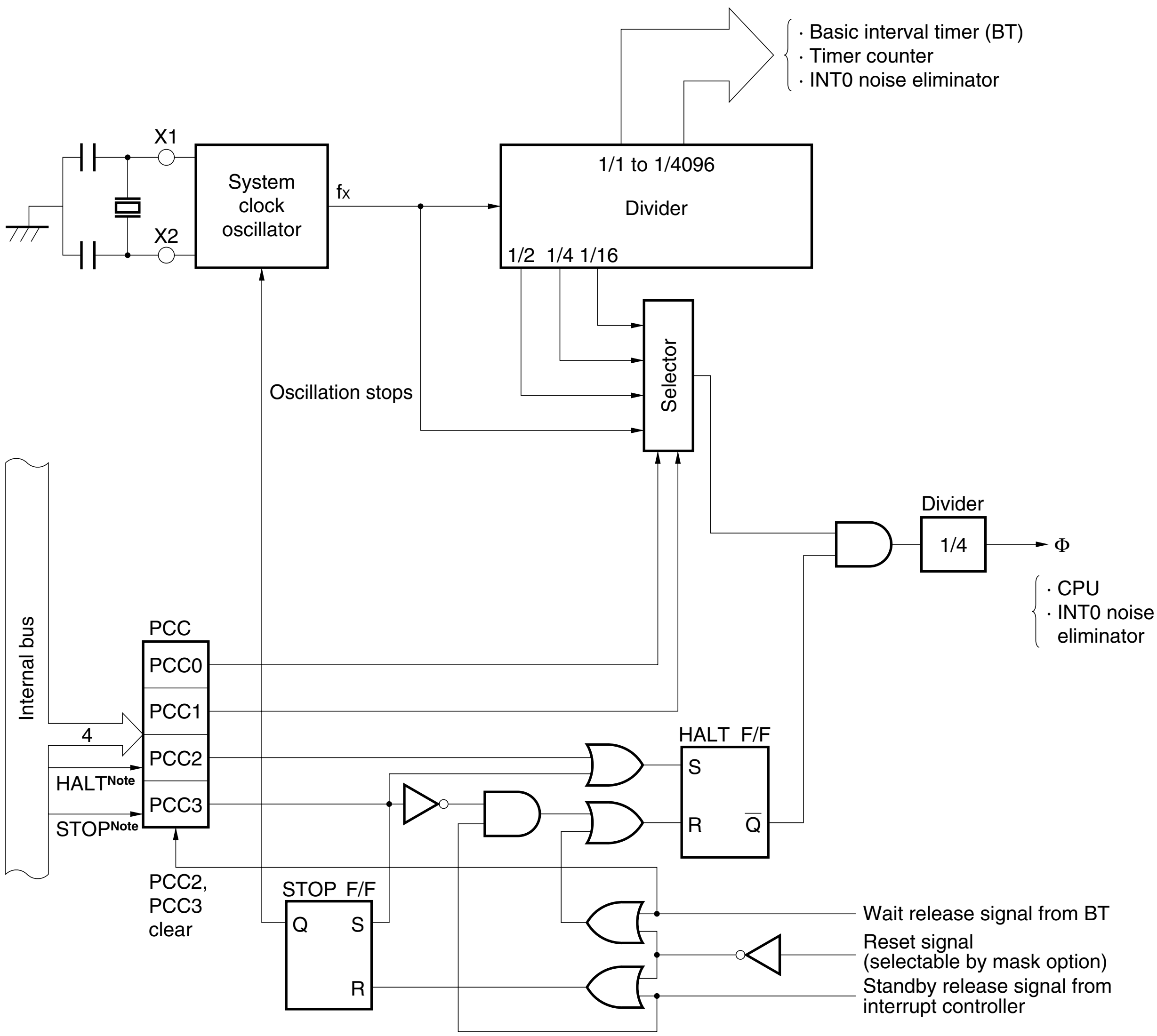
3. PCC: Processor Clock Control Register

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

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-14. Block Diagram of Clock Generator (2/2)

(b) μPD754244 Crystal/Ceramic Oscillation



Note Instruction execution

Remarks 1. fX: System clock frequency

2. Φ = CPU clock

3. PCC: Processor Clock Control Register

4. One clock cycle (tCY) of the CPU clock is equal to one machine cycle of the instruction.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.2.2 Function and operation of clock generator

The clock generator generates the following types of clocks and controls the operation mode of the CPU in the standby mode.

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

The operation of the clock generator is determined by the processor clock control register (PCC) as follows.

(a) When the $\overline{\text{RESET}}$ signal is asserted, the slowest mode of the system clock**Note 1** is selected (PCC = 0).

(b) The CPU clock can be changed in four steps**Note 2** by PCC.

(c) Two standby modes, STOP and HALT, can be used.

(d) The system clock is divided and supplied to the peripheral hardware units.

Notes 1. μPD754144: 64 μs at f_{CC} = 1.0 MHz
μPD754244: 15.3 μs at f_X = 4.19 MHz, 10.7 μs at 6.0 MHz

2. μPD754144: 4, 8, 16, 64 μs at f_{CC} = 1.0 MHz
μPD754244: 0.95, 1.91, 3.81, 15.3 μs at f_X = 4.19 MHz
0.67, 1.33, 6.67, 10.7 μs at f_X = 6.0 MHz

Download from Www.Somanuals.com. All Manuals Search And Download.

(1) Processor clock control register (PCC)

PCC is a 4-bit register that selects the CPU clock Φ with the lower 2 bits and controls the CPU operation mode with the higher 2 bits (refer to **Figure 6-15**).
When either bit 3 or 2 of this register is set to "1", the standby mode is set. When the standby mode has been released by the standby release signal, both the bits are automatically cleared and the normal operation mode is set (for details, refer to **CHAPTER 8 STANDBY FUNCTION**).
The lower 2 bits of PCC are set by a 4-bit memory manipulation instruction (clear the higher 2 bits to "0").
Bits 3 and 2 are set to "1" by the STOP and HALT instructions, respectively.
The STOP and HALT instructions can always be executed regardless of the contents of MBE.

Examples 1. To set the fastest machine cycle mode[Note 1]

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

2. To set the machine cycle of the μPD754244 to 1.33 μs (f_X = 6.0 MHz)[Note 2]

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

3. To set STOP mode (be sure to write a NOP instruction after STOP and HALT instructions)

```
STOP
NOP
```

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

Notes 1. μPD754144: 4 μs (f_{CC} = 1.0 MHz)
μPD754244: 0.67 μs (f_X = 6.0 MHz), or 0.95 μs (f_X = 4.19 MHz)

2. μPD754144: 8 μs (f_{CC} = 1.0 MHz)
μPD754244: 1.91 μs (f_X = 4.19 MHz)

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-15. Format of Processor Clock Control Register

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

CPU operating mode control bits

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

CPU clock selection bits

(μPD754144: When f_{CC} = 1.0 MHz)

PCC1	PCC0	CPU clock frequency	1 machine cycle
0	0	$\Phi = f_{CC}/64$ (15.6 kHz)	64 μs
0	1	$\Phi = f_{CC}/16$ (62.5 kHz)	16 μs
1	0	$\Phi = f_{CC}/8$ (125 kHz)	8 μs
1	1	$\Phi = f_{CC}/4$ (250 kHz)	4 μs

(μPD754244: When f_X = 6.0 MHz)

PCC1	PCC0	CPU clock frequency	1 machine cycle
0	0	$\Phi = f_X/64$ (93.8 kHz)	10.7 μs
0	1	$\Phi = f_X/16$ (375 kHz)	2.67 μs
1	0	$\Phi = f_X/8$ (750 kHz)	1.33 μs
1	1	$\Phi = f_X/4$ (1.5 MHz)	0.67 μs

(μPD754244: When f_X = 4.19 MHz)

PCC1	PCC0	CPU clock frequency	1 machine cycle
0	0	$\Phi = f_X/64$ (65.5 kHz)	15.3 μs
0	1	$\Phi = f_X/16$ (262 kHz)	3.81 μs
1	0	$\Phi = f_X/8$ (524 kHz)	1.91 μs
1	1	$\Phi = f_X/4$ (1.05 MHz)	0.95 μs

Remark f_{CC} and f_X: System clock oscillation frequency

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) System clock oscillator

(a) μPD754144 (RC oscillation)

The system clock oscillator oscillates by means of a resistor (R) and capacitor (C) connected to the CL1 and CL2 pins.

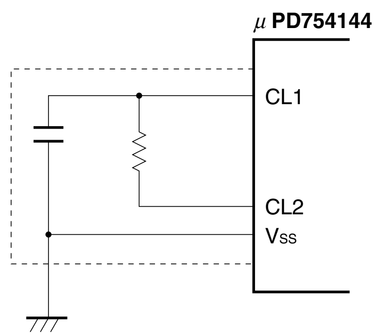
An external clock cannot be input for RC oscillation.

The relationship between the output frequency of the system clock oscillator (f_{CC}), resistance (R), and capacitance (C) is as follows.

$$f_{CC} = \frac{1}{2RC}$$

Cautions f_{CC} may have a frequency deviation due to fluctuation of the supply voltage or temperature.

Figure 6-16. RC Oscillation External Circuit



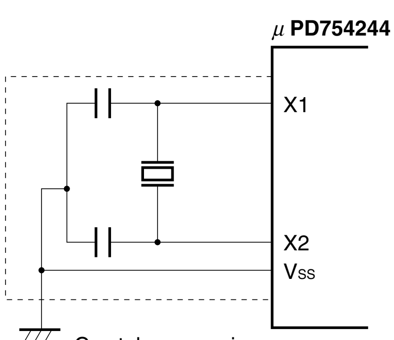
(b) μPD754244 (Crystal/ceramic oscillation)

The system clock oscillator oscillates by means of crystal or ceramic resonator connected to the X1 and X2 pins (6.0 MHz or 4.19 MHz TYP.).

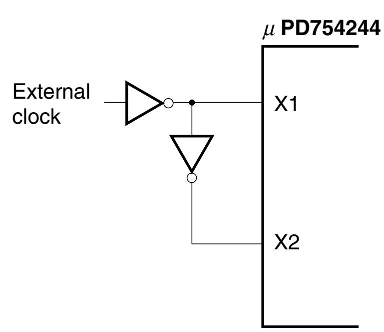
An external clock can also be input.

Figure 6-17. Crystal/Ceramic Oscillation External Circuit

(i) Crystal/ceramic oscillation



(ii) External clock



Download from Www.Somanuals.com. All Manuals Search And Download.

Cautions 1. The X2 pin of the μPD754244 is internally pulled up to V_{DD} by a resistor of 50 kΩ (typ.) in the STOP mode.

2. Wire the portion enclosed by the dotted lines in Figures 6-16 and 6-17 as follows to prevent adverse influence by wiring capacitance when using the system clock oscillator.

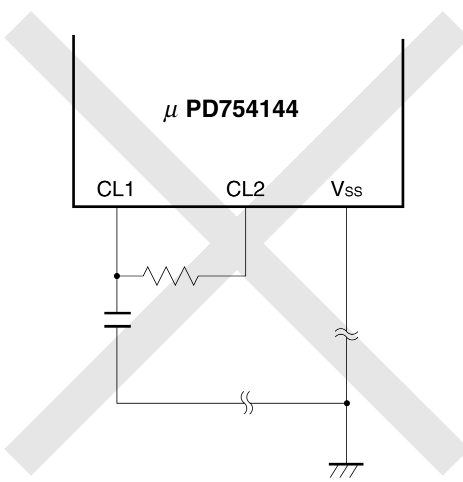
- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any other signal lines.**
- **Do not route the wiring in the vicinity of a line through which a high alternating current is flowing.**
- **Always make the potential at the connecting point of the capacitor of the oscillator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

Figure 6-18 shows incorrect examples of connecting the resonator.

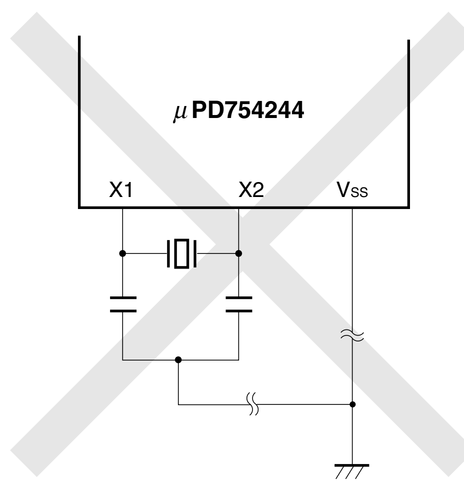
Figure 6-18. Example of Incorrect Resonator Connection (1/3)

(a) Wiring length too long

• μPD754144



• μPD754244

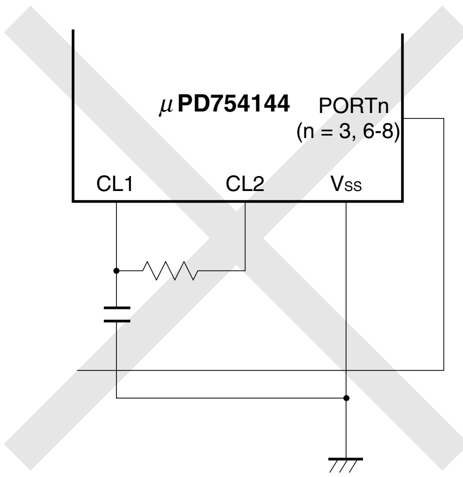


Download from Www.Somanuals.com. All Manuals Search And Download.

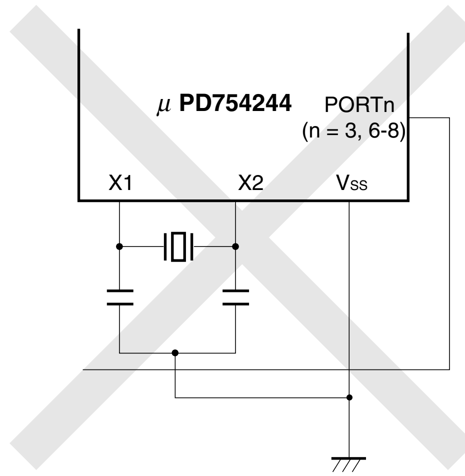
Figure 6-18. Example of Incorrect Resonator Connection (2/3)

(b) Crossed signal line

• μPD754144

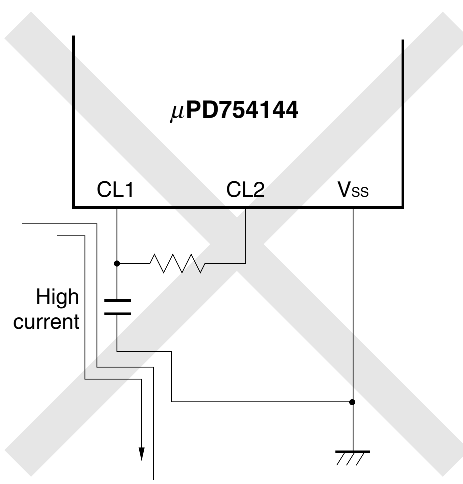


• μPD754244

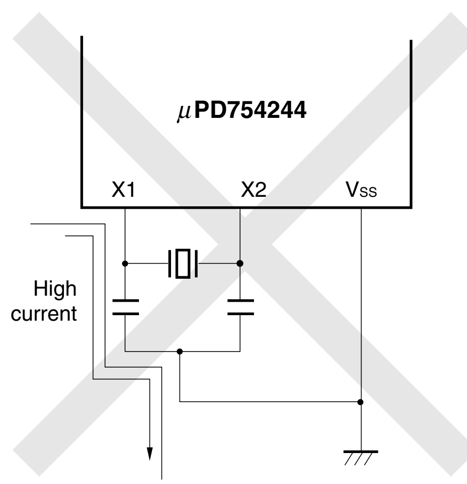


(c) High alternating current close to signal line

• μPD754144



• μPD754244

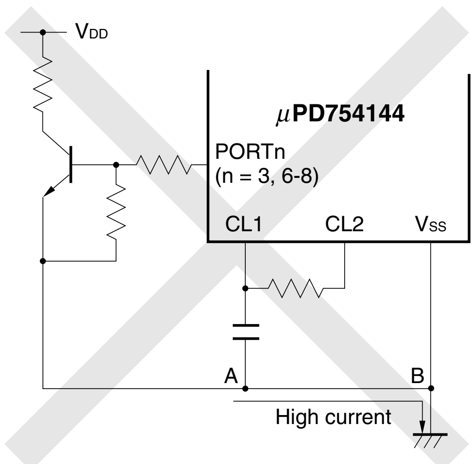


Download from Www.Somanuals.com. All Manuals Search And Download.

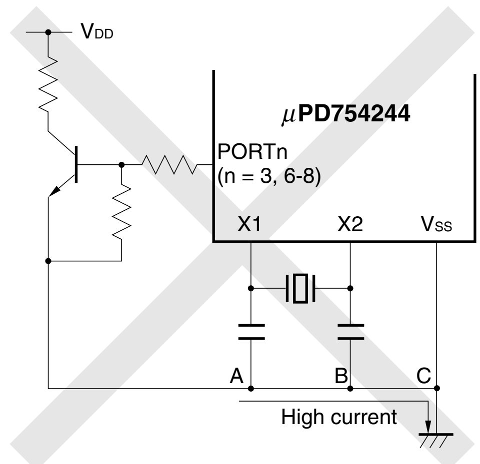
Figure 6-18. Example of Incorrect Resonator Connection (3/3)

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

• μPD754144

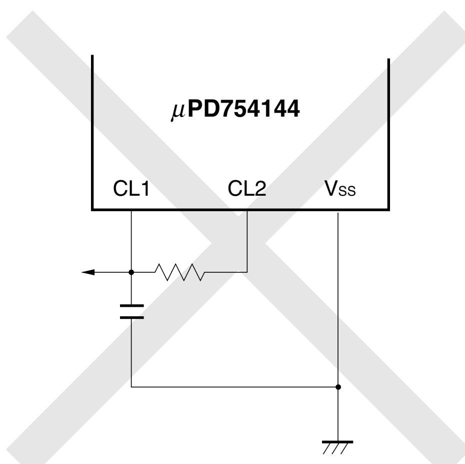


• μPD754244

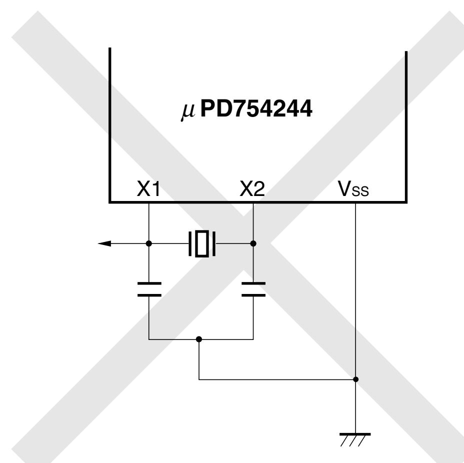


(e) Signal fetched

• μPD754144



• μPD754244



(3) Divider circuit

The divider circuit divides the output of the system clock oscillator to create various clock signals.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.2.3 Setting CPU clock

(1) Time required to switch CPU clock

The CPU clock can be switched by using the lower 2 bits of PCC. The processor does not operate with the selected clock, however, immediately after data has been written to the registers; it operates with the pre-change clock for the duration of a certain number of machine cycles. To stop oscillation of the system clock, therefore, execute the STOP instruction after a specific time has elapsed.

Table 6-5. Maximum Time Required for CPU Clock Switching

Set Value Before Switching		Set Value After Switching			
PCC1	PCC0	PCC1 = 0, PCC0 = 0	PCC1 = 0, PCC0 = 1	PCC1 = 1, PCC0 = 0	PCC1 = 1, PCC0 = 1
0	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	

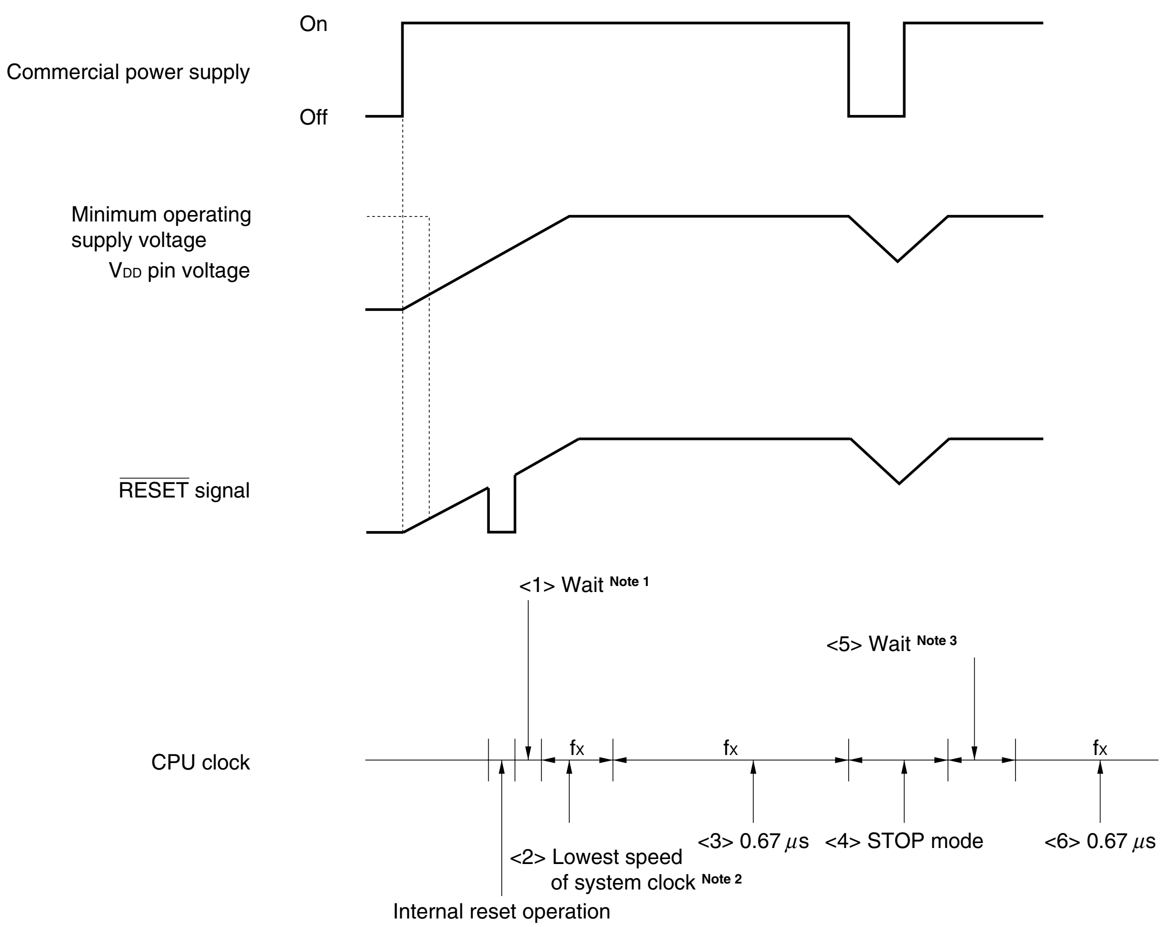
Caution The value of f_X changes depending on conditions such as the ambient temperature of the resonators, and variations in load capacitance performance.
Particularly when f_X is higher than the nominal value, the machine cycle in the table becomes bigger than the machine cycle obtained by the nominal value. Therefore, when setting the wait time required for switching the CPU clock, set it longer than the machine cycle obtained by the f_X nominal value.

(2) CPU clock switching procedure

The switching procedure of the CPU clock is explained according to Figure 6-19.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-19. CPU Clock Switching Example



<1> Wait time[Note 1] to secure the oscillation stabilization time in response to $\overline{\text{RESET}}$ signal generation.

<2> The CPU starts operating at the lowest system clock speed[Note 2].

<3> The PCC is rewritten and the device operates at maximum speed after the elapse of sufficient time for the V_{DD} pin voltage to increase to a level which allows maximum speed operation.

<4> Interruption of the commercial power is detected by means of interrupt input, etc., and the STOP mode is entered.

<5> Wait time[Note 3] to secure the oscillation stabilization time after restoration of commercial power is detected by means of an interrupt, etc., and the device is released from the STOP mode.

<6> Operates normally.

Notes 1. μPD754144: Fixed to $56/f_{CC}$ (56 μs at 1.0 MHz).
μPD754244: The wait time can be selected by a mask option.
Can be selected from $2^{15}/f_X$ = 7.81 ms or $2^{17}/f_X$ = 31.3 ms at 4.19 MHz, and from $2^{15}/f_X$ = 5.46 ms or $2^{17}/f_X$ = 21.8 ms at 6.0 MHz.

2. μPD754144: 64 μs at f_{CC} = 1.0 MHz.
μPD754244: 15.3 μs at 4.19 MHz and 10.7 μs at 6.0 MHz

3. μPD754144: $2^9/f_{CC}$ (512 μs at 1.0 MHz)
μPD754244: The following four times can be selected by BTM:
$2^{20}/f_X$, $2^{17}/f_X$, $2^{15}/f_X$, $2^{13}/f_X$

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3 Basic Interval Timer/Watchdog Timer

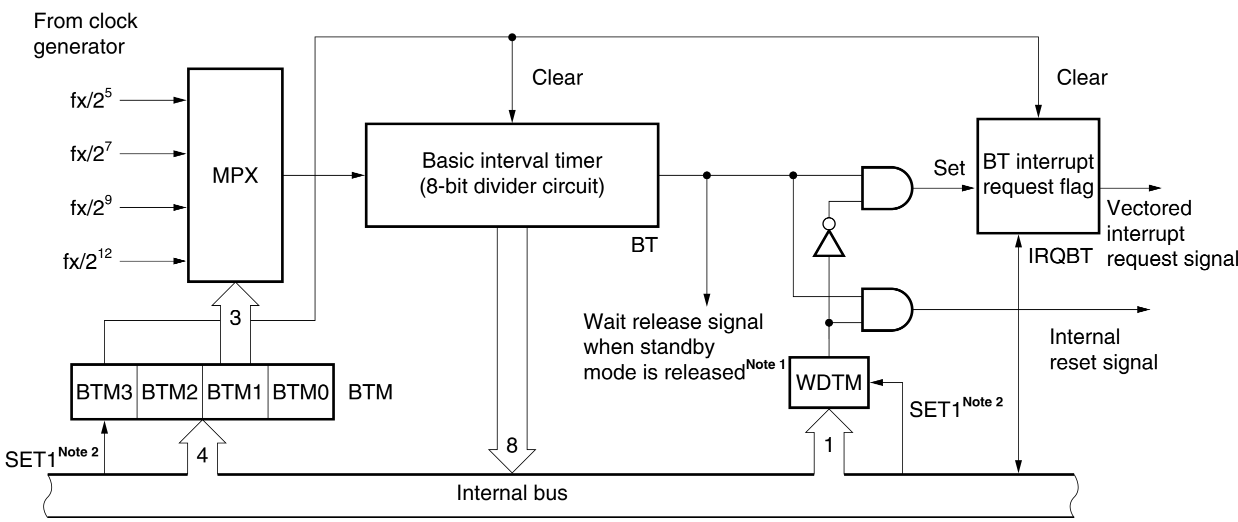
The μPD754244 has an 8-bit basic interval timer/watchdog timer that has the following functions.

(a) Interval timer operation to generate reference time interrupt
(b) Watchdog timer operation to detect program hang-up and reset CPU
(c) To select and count wait time when standby mode is released (μPD754244 only)
(d) To read count value

6.3.1 Configuration of basic interval timer/watchdog timer

Figure 6-20 shows the configuration of the basic interval timer/watchdog timer.

Figure 6-20. Block Diagram of Basic Interval Timer/Watchdog Timer



Notes 1. In the case of the μPD754144 (RC oscillation), it is not possible to select the wait time after the release of standby mode or after a reset. The μPD754144 has almost no oscillation stabilization wait time and returns to normal operating mode after counting 2^9/fcc (512 μs at 1.0 MHz).
In the case of μPD754244 (crystal/ceramic oscillation), it is possible to select the wait time after the release of standby mode.
Refer to **CHAPTER 8 STANDBY FUNCTION** and **CHAPTER 9 RESET FUNCTION**, for details.

2. Execution of the instruction.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3.2 Basic interval timer mode register (BTM)

BTM is a 4-bit register that controls the operation of the basic interval timer (BT).

This register is set by a 4-bit memory manipulation instruction.

Bit 3 of BT can be manipulated by a bit manipulation instruction.

Example To set the interrupt generation interval of the μPD754244 to 1.37 ms (at f_X = 6.0 MHz)[Note]

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

Note It is 1.95 ms when the μPD754244 is operating at f_X = 4.19 MHz.
In the case of μPD754144, it is fixed to $2^9/f_{CC}$ (512 μs at 1.0 MHz).

When bit 3 of this register is set to "1", the contents of BT are cleared, and at the same time, the basic interval timer/watchdog timer interrupt request flag (IRQBT) is cleared (the basic interval timer/watchdog timer is started).

When the $\overline{\text{RESET}}$ signal is asserted, the contents of this register are cleared to "0", and the generation interval time of the interrupt request signal is set to the longest value.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-21. Format of Basic Interval Timer Mode Register

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

μPD754144: f_{CC} = 1.0 MHz

BTM2	BTM1	BTM0	Specifies input clock	Interrupt interval time
0	0	0	$f_{CC}/2^{12}$ (244 Hz)	$2^{20}/f_{CC}$ (1.05 s)
0	1	1	$f_{CC}/2^{9}$ (1.95 kHz)	$2^{17}/f_{CC}$ (131 ms)
1	0	1	$f_{CC}/2^{7}$ (7.81 kHz)	$2^{15}/f_{CC}$ (32.8 ms)
1	1	1	$f_{CC}/2^{5}$ (31.3 kHz)	$2^{13}/f_{CC}$ (8.19 ms)
Other			Setting prohibited	–

μPD754244: f_X = 6.0 MHz

BTM2	BTM1	BTM0	Specifies input clock	Interrupt interval time (wait time when standby mode is released)Note
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			Setting prohibited	–

μPD754244: f_X = 4.19 MHz

BTM2	BTM1	BTM0	Specifies input clock	Interrupt interval time (wait time when standby mode is released)Note
0	0	0	$f_X/2^{12}$ (1.02 kHz)	$2^{20}/f_X$ (250 ms)
0	1	1	$f_X/2^{9}$ (8.19 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			Setting prohibited	–

Basic interval timer/watchdog timer start control bit (BTM3)

When "1" is written to this bit, the basic interval timer/watchdog timer is started (counter and interrupt request flag are cleared). When the timer starts operating, this bit is automatically reset to "0".

Note In the μPD754244 only, wait time is selectable when standby mode is released.
In the μPD754144, wait time is always fixed to $2^9/f_{CC}$ (512 μs at 1.0 MHz).

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3.3 Watchdog timer enable flag (WDTM)

WDTM is a flag that enables assertion of the reset signal when an overflow occurs.

This flag is set by a bit manipulation instruction. Once this flag has been set, it cannot be cleared by an instruction.

Example To set watchdog timer function

```
SEL     MB15    ; or CLR1 MBE
SET1    WDTM
     :
SET1    BTM.3   ; Sets bit 3 of BTM to "1"
```

The contents of this flag are cleared to 0 when the $\overline{\text{RESET}}$ signal is asserted.

Figure 6-22. Format of Watchdog Timer Enable Flag (WDTM)

Address

F8BH.3 WDTM

0	BT mode Sets IRQBT when basic interval timer (BT) overflows
1	WT mode Asserts internal reset signal when basic interval timer (BT) overflows

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3.4 Operation as basic interval timer

When WDTM is reset to “0”, the interrupt request flag (IRQBT) is set by the overflow of the basic interval timer (BT), and the basic interval timer/watchdog timer operates as the basic interval timer. BT is always incremented by the clock supplied by the clock generator and its counting operation cannot be stopped.

Four time intervals at which an interrupt occurs can be selected by BTM (μPD754244 only. Refer to **Figure 6-21**).

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

The count value of BT can be read by using an 8-bit manipulation instruction. No data can be written to BT.

Start the timer operation as follows (<1> and <2> may be performed simultaneously).

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

Example To generate an interrupt at intervals of the μPD754244 of 1.37 ms (at f_X = 6.0 MHz)[Note]

```
SET1  MBE
SEL   MB15
MOV   A, #1111B
MOV   BTM, A      ; Sets time and starts
EI                ; Enables interrupt
EI    IEBT        ; Enables BT interrupt
```

Note It is 1.95 ms when the μPD754244 is operating at f_X = 4.19 MHz.
In the case of the μPD754144, it is 8.19 ms at f_{CC} = 1.0 MHz.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3.5 Operation as watchdog timer

The basic interval timer/watchdog timer operates as a watchdog timer that asserts the internal reset signal when an overflow occurs in the basic interval timer (BT), if WDTM is set to “1”. However, if the overflow occurs during the oscillation wait time that elapses after the STOP instruction has been released, the reset signal is not asserted. (Once WDTM has been set to “1”, it cannot be cleared by any means other than reset.) BT is always incremented by the clock supplied from the clock generator, and its count operation cannot be stopped.

In the watchdog timer mode, a program hang-up is detected by using the interval time at which BT overflows. As this interval time, four values can be selected by using bits 2 to 0 of BTM (μPD754244 only. Refer to **Figure 6-21**). Select the interval time best-suited to detecting a hang-up that may occur in your system. Set an interval time, divide the program into several modules that can be executed within the set interval time, and execute an instruction that clears BT at the end of each module. If this instruction that clears BT is not executed within the set interval time (in other words, if a module of the program is not normally executed, i.e., if a hang-up occurs), BT overflows, the internal reset signal is asserted, and the program is terminated forcibly. Consequently, assertion of the internal reset signal indicates occurrence and detection of a program hang-up.

Set the watchdog timer as follows (<1> and <2> may be performed simultaneously).

<1> Set interval time to BTM.
<2> Set bit 3 of BTM to “1”.
<3> Set WDTM to “1”.
(<1> to <3>: Initial setting)
<4> After setting <1> to <3> above, set bit 3 of BTM to “1” within the interval time.

Download from Www.Somanuals.com. All Manuals Search And Download.

Example To use the μPD754244 as a watchdog timer with a time interval of 5.46 ms (at f_X = 6.0 MHz).**Note** Divide the program into several modules, each of which is completed within the set time of BTM (5.46 ms), and clear BT at the end of each module. If a hang-up occurs, BT is not cleared within the set time. As a result, BT overflows, and the internal reset signal is asserted.

Initial setting:

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

(After that, set bit 3 of BTM to "1" every 5.46 ms.)

Module 1:

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

Processing completed within 5.46 ms

Module 2:

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

Processing completed within 5.46 ms

:

Note It is 7.81 ms when the μPD754244 is operating at f_X = 4.19 MHz.
In the case of the μPD754144, it is 32.8 ms at f_{CC} = 1.0 MHz.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.3.6 Other functions

The basic interval timer/watchdog timer has the following functions, regardless of the operations as the basic interval timer or watchdog timer.

<1> Selects and counts wait time after standby mode has been released
<2> Reads count value

(1) Selecting and counting wait time after STOP mode has been released Note 1

When the STOP mode has been released, a wait time elapses during which the operation of the CPU is stopped until the basic interval timer (BT) overflows, so that oscillation of the system clock becomes stabilized.
The wait time that elapses after the $\overline{\text{RESET}}$ signal has been asserted is fixed by a mask option. When the STOP mode is released by an interrupt, however, the wait time can be selected by BTM. The wait time in this case is the same as the interval time shown in Figure 6-21. Set BTM before setting the STOP mode (for details, refer to **CHAPTER 8 STANDBY FUNCTION**).

Example To set a wait time of 5.46 ms that elapses when the STOP mode has been released by an interrupt (at f_X = 6.0 MHz)**Note 2**

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

Notes **1.** The μPD754244 only. In the μPD754144, the wait time is fixed to $2^9/f_{CC}$ (512 μs at 1.0 MHz).
2. It is 7.81 ms when the μPD754244 is operating at f_X = 4.19 MHz.

(2) Reading count value

The count value of the basic interval timer (BT) can be read by using an 8-bit manipulation instruction. No data can be written to the basic interval timer.

Caution To read the count value of BT, execute the read instruction twice to prevent undefined data from being read while the count value is updated. Compare the two read values. If the values are similar, take the latter value as the result. If the two values are completely different, redo from the beginning.

Example To read count value of BT

```
       SET1   MBE
       SEL    MB15
       MOV    HL, #BT    ; Sets address of BT to HL
LOOP:  MOV    XA, @HL    ; Reads first time
       MOV    BC, XA
       MOV    XA, @HL    ; Reads second time
       SKE    XA, BC
       BR     LOOP
```


Download from Www.Somanuals.com. All Manuals Search And Download.

6.4 Timer Counter

The μPD754244 incorporates a three-channel timer counter. The timer counter has the following functions.

(a) Programmable interval timer operation
(b) Square wave output of any frequency to PTO0-PTO2 pins
(c) Count value read function

The timer counter can operate in the following four modes as set by the mode register.

Table 6-6. Mode List

Mode \ Channel	Channel 0	Channel 1	Channel 2	TM11	TM10	TM21	TM20	Refer to
8-bit timer counter mode	○	○	○	0	0	0	0	6.4.2
PWM pulse generator mode	×	×	○	0	0	0	1	6.4.3
16-bit timer counter mode	×	○		1	0	1	0	6.4.4
Carrier generator mode	×	○		0	0	1	1	6.4.5

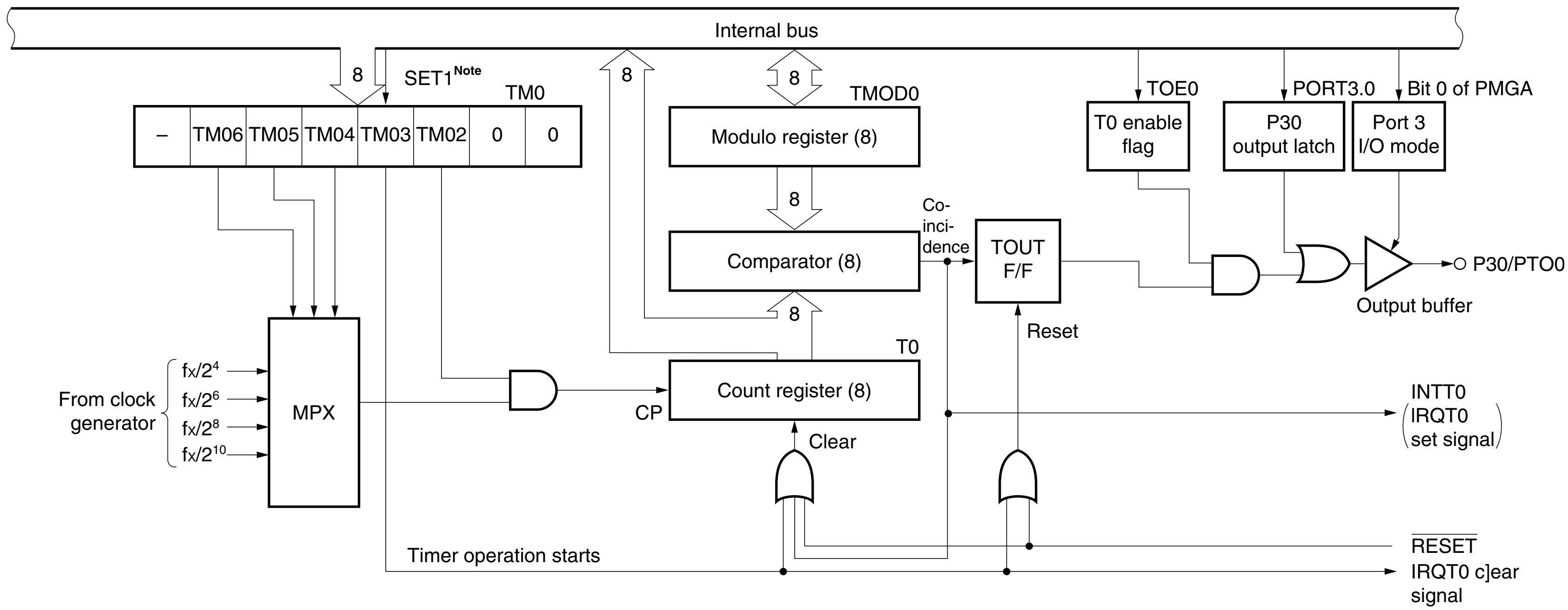
Remark ×**:** Corresponding function is not available.

6.4.1 Configuration of timer counter

The configuration of the timer counter is shown in Figures 6-23 to 6-25.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-23. Block Diagram of Timer Counter (Channel 0)

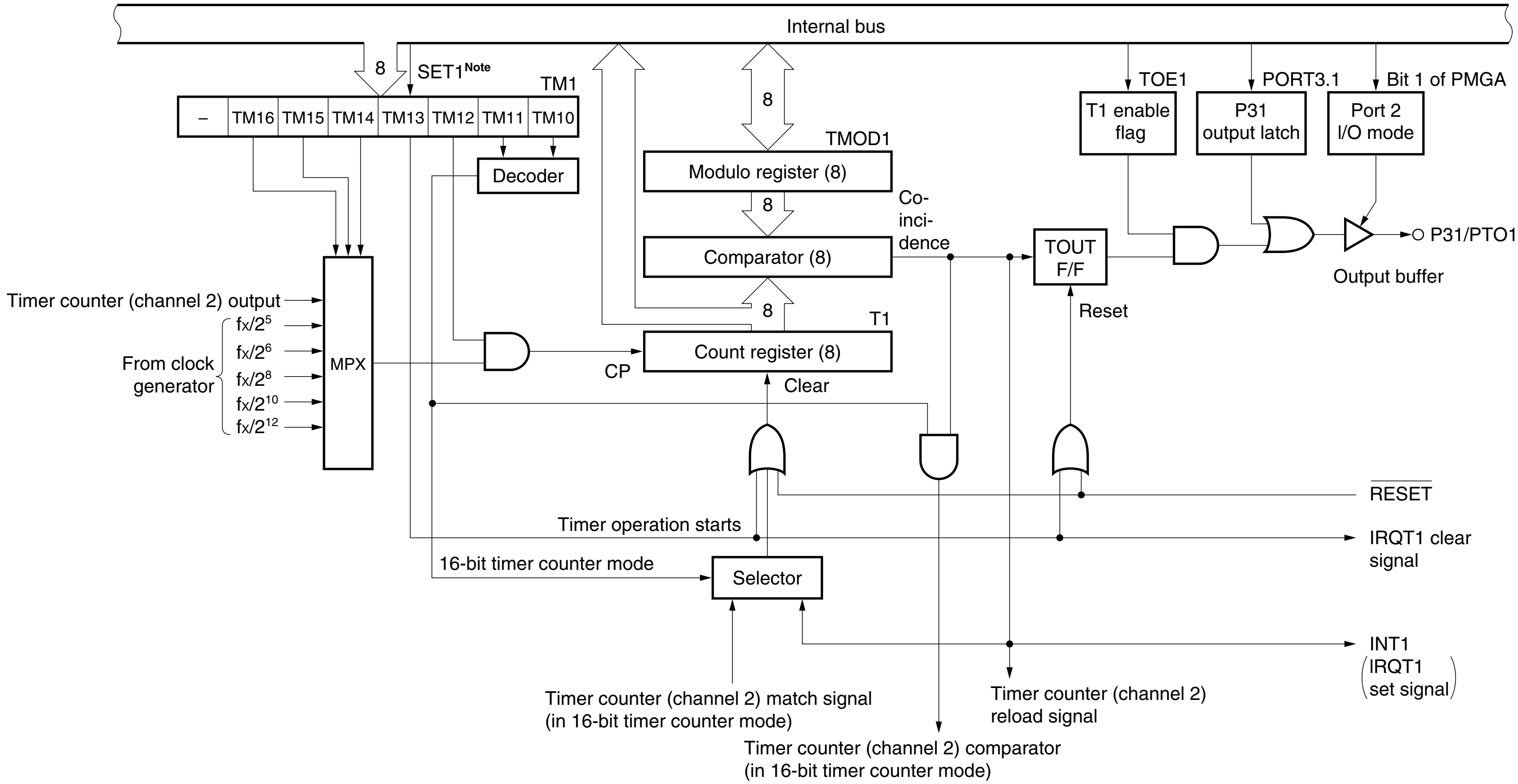


Note Execution of the instruction

Caution Be sure to clear bits 1 and 0 to 0 when setting data to TM0.

Download from Www.Somanuals.com. All Manuals Search And Download.

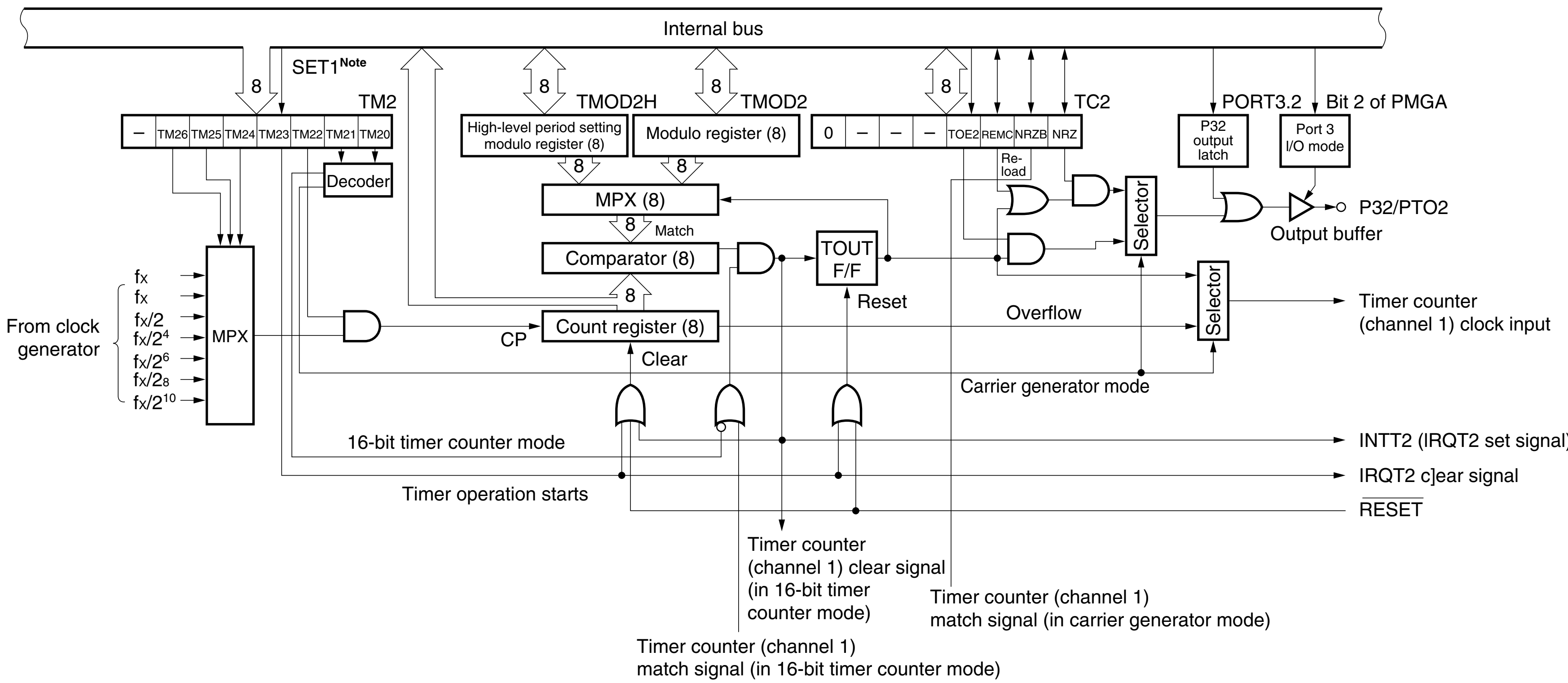
Figure 6-24. Block Diagram of Timer Counter (Channel 1)



Note Execution of the instruction

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-25. Block Diagram of Timer Counter (Channel 2)



Note Execution of the instruction

Caution Be sure to clear bit 7 to 0 when setting data to TC2.

Download from Www.Somanuals.com. All Manuals Search And Download.

(1) Timer counter mode registers (TM0, TM1, TM2)

A timer counter mode register (TMn) is an 8-bit register that controls the corresponding timer counter. Figures 6-26 to 6-28 show the formats of the various mode registers.
The timer counter mode register is set by an 8-bit memory manipulation instruction.
Bit 3 of this register is a timer start bit and can be manipulated in 1-bit units independently of the other bits. This bit is automatically reset to "0" when the timer starts operating.
All the bits of the timer counter mode register are cleared to "0" when the $\overline{RESET}$ signal is asserted.

Examples 1. To start timer in interval timer mode of CP = 5.86 kHz (at f_X = 6.0 MHz)Note

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

2. To restart timer according to setting of timer counter mode register

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

Note CP = 4.10 kHz when the μPD754244 is operating at fx = 4.19 MHz.
CP = 977 kHz when the μPD754144 is operating at f_{CC} = 1.0 MHz.

Remark n = 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-26. Format of Timer Counter Mode Register (Channel 0)

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

Count pulse (CP) select bit

μPD754144: f_{CC} = 1.0 MHz

TM06	TM05	TM04	Count pulse (CP)
1	0	0	$f_{CC}/2^{10}$ (977 Hz)
1	0	1	$f_{CC}/2^{8}$ (3.91 kHz)
1	1	0	$f_{CC}/2^{6}$ (15.6 kHz)
1	1	1	$f_{CC}/2^{4}$ (62.5 kHz)
Other			Setting prohibited

μPD754244: f_X = 6.0 MHz

TM06	TM05	TM04	Count pulse (CP)
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			Setting prohibited

μPD754244: f_X = 4.19 MHz

TM06	TM05	TM04	Count pulse (CP)
1	0	0	$f_X/2^{10}$ (4.10 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			Setting prohibited

Timer start command bit

TM03	Clears counter and IRQT0 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM02	Count operation
0	Stops (count value retained)
1	Count operation

Note Be sure to clear bits 0 and 1 to 0 when setting data to TM0.

Caution After a reset, all bits of TM0 become "0", therefore when operating the timer it is necessary to set the count pulse value first. Moreover, when any value other than the above is written to CP, the count pulse set becomes 0 and TM0 does not operate as a timer.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-27. Format of Timer Counter Mode Register (Channel 1) (1/2)

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

Count pulse (CP) select bit

μPD754144: f_{CC} = 1.0 MHz

TM16	TM15	TM14	Count pulse (CP)
0	1	0	Overflow of timer counter (channel 2)
0	1	1	$f_{CC}/2^5$ (31.3 kHz)
1	0	0	$f_{CC}/2^{12}$ (244 Hz)
1	0	1	$f_{CC}/2^{10}$ (977 Hz)
1	1	0	$f_{CC}/2^8$ (3.91 kHz)
1	1	1	$f_{CC}/2^6$ (15.6 kHz)
Other			Setting prohibited

μPD754244: f_X = 6.0 MHz

TM16	TM15	TM14	Count pulse (CP)
0	1	0	Overflow of timer counter (channel 2)
0	1	1	$f_X/2^5$ (188 kHz)
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)
Other			Setting prohibited

μPD754244: f_X = 4.19 MHz

TM16	TM15	TM14	Count pulse (CP)
0	1	0	Overflow of timer counter (channel 2)
0	1	1	$f_X/2^5$ (131 kHz)
1	0	0	$f_X/2^{12}$ (1.02 kHz)
1	0	1	$f_X/2^{10}$ (4.10 kHz)
1	1	0	$f_X/2^8$ (16.4 kHz)
1	1	1	$f_X/2^6$ (65.5 kHz)
Other			Setting prohibited

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-27. Format of Timer Counter Mode Register (Channel 1) (2/2)

Timer start command bit

TM13	Clears counter and IRQT1 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM12	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM11	TM10	Mode
0	0	8-bit timer counter mode[Note]
1	0	16-bit timer counter mode
Other		Setting prohibited

Note This mode is used as a carrier generator mode when used in combination with TM20, TM21 (=11) of timer counter mode register (channel 2).

Caution After a reset, all bits of TM1 become "0", therefore when operating the timer it is necessary to set the count pulse value first. Moreover, when any setting prohibited value is set, the count pulse set becomes 0 and TM0 does not operate as a timer.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-28. Format of Timer Counter Mode Register (Channel 2) (1/2)

Address	7	6	5	4	3	2	1	0	Symbol
F90H	–	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) select bit

μPD754144: f_{CC} = 1.0 MHz

TM26	TM25	TM24	Count pulse (CP)
0	1	0	$f_{CC}/2$ (500 MHz)
0	1	1	f_{CC} (1.0 MHz)
1	0	0	$f_{CC}/2^{10}$ (977 Hz)
1	0	1	$f_{CC}/2^{8}$ (3.91 kHz)
1	1	0	$f_{CC}/2^{6}$ (15.6 kHz)
1	1	1	$f_{CC}/2^{4}$ (62.5 kHz)
Other			Setting prohibited

μPD754244: f_X = 6.0 MHz

TM26	TM25	TM24	Count pulse (CP)
0	1	0	$f_X/2$ (3.00 MHz)
0	1	1	f_X (6.0 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			Setting prohibited

μPD754244: f_X = 4.19 MHz

TM26	TM25	TM24	Count pulse (CP)
0	1	0	$f_X/2$ (2.10 MHz)
0	1	1	f_X (4.19 MHz)
1	0	0	$f_X/2^{10}$ (4.10 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			Setting prohibited

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-28. Format of Timer Counter Mode Register (Channel 2) (2/2)

Timer start command bit

TM23	Clears counter and IRQT2 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM22	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM21	TM20	Mode
0	0	8-bit timer counter mode
0	1	PWM pulse generator mode
1	0	16-bit timer counter mode
1	1	Carrier generator mode

Caution After a reset, all bits of TM2 become "0", therefore when operating the timer it is necessary to set the count pulse value first. Moreover, when any setting prohibited value is set, the count pulse set becomes 0 and TM0 does not operate as a timer.

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Timer counter output enable flags (TOE0, TOE1)

Timer counter output enable flags TOE0 and TOE1 enable or disable output to the PTO0 and PTO1 pins in the timer out F/F (TOUT F/F) status.

The timer out F/F is inverted by a match signal from the comparator. When bit 3 (timer start command bit) of timer counter mode register TM0 or 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” when the $\overline{\text{RESET}}$ signal is asserted.

Figure 6-29. Format of Timer Counter Output Enable Flag

Address		
FA2H	TOE0	Channel 0
FAAH	TOE1	Channel 1

Timer counter output enable flag (W)

0	Disabled
1	Enabled

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Timer counter control register (TC2)

The timer counter control register (TC2) is an 8-bit register that controls the timer counter (channel 2). Figure 6-30 shows the format of this register.

This register controls timer output enable carrier generator mode used in combination with the timer counter (channel 1).

TC2 is set by an 8- or 4-bit manipulation instruction and bit manipulation instruction.

All the bits of TC2 are cleared to 0 when the internal reset signal is asserted.

Figure 6-30. Format of Timer Counter Control Register

Address	7	6	5	4	3	2	1	0	Symbol
F92H	0[Note]	–	–	–	TOE2	REMC	NRZB	NRZ	TC2

Timer counter output enable flag

TOE2	Timer output
0	Disabled (low level output)
1	Enabled

Remote controller output control flag

REMC	Remote controller output
0	Outputs carrier pulse to PTO2 pin when NRZ = 1
1	Outputs high level to PTO2 pin when NRZ = 1

No return zero buffer flag

NRZB	Area to store no return zero data to be output next. Transferred to NRZ when interrupt of timer counter (channel 1) occurs

No return zero flag

NRZ	No return zero data
0	Outputs low level to PTO2 pin
1	Outputs carrier pulse to PTO2 pin

Note Be sure to clear bits 7 to 0 when setting data to TC2.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.4.2 Operation in 8-bit timer counter mode

In this mode, the timer counter is used as an 8-bit timer counter. In this case, the timer counter operates as an 8-bit programmable interval timer or counter.

(1) Register setting

In the 8-bit timer counter mode, the following four registers are used:

- Timer counter mode register (TMn)
- Timer counter control register (TC2)[Note]
- Timer counter count register (Tn)
- Timer counter modulo register (TMODn)

Note Channels 0 and 1 of the timer counter use the timer counter output enable flags (TOE0 and TOE1).

(a) Timer counter mode register (TMn)

In the 8-bit timer counter mode, set TMn as shown in Figure 6-31 (for the format of TMn, refer to **Figures 6-26** to **6-28**).

TMn is manipulated by an 8-bit manipulation instruction. Bit 3 is a timer start command bit which can be manipulated in 1-bit units. This bit is automatically cleared to 0 when the timer starts operating.

TMn is cleared to 00H when the internal reset signal is asserted.

Remark n = 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-31. Setting of Timer Counter Mode Register (1/3)

(a) Timer counter (channel 0)

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

Count pulse (CP) select bit

TM06	TM05	TM04	Count pulse (CP)
1	0	0	$fx/2^{10}$
1	0	1	$fx/2^{8}$
1	1	0	$fx/2^{6}$
1	1	1	$fx/2^{4}$
Other			Setting prohibited

Timer start command bit

TM03	Clears counter and IRQT0 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM02	Count operation
0	Stops (count value retained)
1	Count operation

Note Be sure to clear bits 0 and 1 to 0 when setting data to TM0.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-31. Setting of Timer Counter Mode Register (2/3)

(b) Timer 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) select bit

TM16	TM15	TM14	Count pulse (CP)
0	1	0	Overflow of timer counter (channel 2)
0	1	1	$fx/2^5$
1	0	0	$fx/2^{12}$
1	0	1	$fx/2^{10}$
1	1	0	$fx/2^8$
1	1	1	$fx/2^6$
Other			Setting prohibited

Timer start command bit

TM13	Clears counter and IRQT1 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM12	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM11	TM10	Mode
0	0	8-bit timer counter mode

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-31. Setting of Timer Counter Mode Register (3/3)

(c) Timer counter (channel 2)

Address	7	6	5	4	3	2	1	0	Symbol
F90H	–	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) select bit

TM26	TM25	TM24	Count pulse (CP)
0	1	0	fx/2
0	1	1	fx
1	0	0	$fx/2^{10}$
1	0	1	$fx/2^8$
1	1	0	$fx/2^6$
1	1	1	$fx/2^4$
Other			Setting prohibited

Timer start command bit

TM23	Clears counter and IRQT2 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM22	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM21	TM20	Mode
0	0	8-bit timer counter mode

Download from Www.Somanuals.com. All Manuals Search And Download.

(b) Timer counter control register (TC2)

In the 8-bit timer counter mode, set TC2 as shown in Figure 6-32 (for the format of TC2, refer to **Figure 6-30 Format of Timer Counter Control Register**).

TC2 is manipulated by an 8- or 4-bit, or bit manipulation instruction.

The value of TC2 is cleared to 00H when the internal reset signal is asserted.

The flags shown by a solid line in the figure below are used in the 8-bit timer counter mode.

Do not use the flags shown by a dotted line in the figure below in the 8-bit timer counter mode (clear these flags to 0).

Figure 6-32. Setting of Timer Counter Control Register

7	6	5	4	3	2	1	0	Symbol
0	–	–	–	TOE2	REMC	NRZB	NRZ	TC2

Timer counter output enable flag

TOE2	Timer output
0	Disabled (low level output)
1	Enabled

Figure 6-33. Setting of Timer Counter Output Enable Flag

Address		
FA2H	TOE0	Channel 0
FAAH	TOE1	Channel 1

Timer counter output enable flag (W)

0	Disabled (low level output)
1	Enabled

(2) Time setting of timer counter

Download from Www.Somanuals.com. All Manuals Search And Download.

[Timer set time] (cycle) is calculated by dividing [contents of modulo register + 1] by [count pulse (CP) frequency] selected by the mode register.

$$T\ (sec) = \frac{n+1}{f_{CP}} = (n+1) \cdot (resolution)$$

where,

T (sec): Timer set time (seconds)

f_{CP} (Hz): CP frequency (Hz)

n: Contents of modulo register ($n \neq 0$)

Once the timer has been set, interrupt request flag IRQTn is set at the set time interval of the timer.

Table 6-7 shows the resolution of each count pulse of the timer counter and the longest set time (time when FFH is set to the modulo register).

Table 6-7. Resolution and Longest Set Time (8-Bit Timer Counter Mode) (1/3)
(TM10 = 0, TM11 = 0, TM20 = 0, TM21 = 0)

(a) μPD754244: at 6.0 MHz

8-bit timer counter (channel 0)

Mode Register			8-bit Timer Counter (Channel 0)	
TM06	TM05	TM04	Resolution	Longest set time
1	0	0	171 μs	43.7 ms
1	0	1	42.7 μs	10.9 ms
1	1	0	10.7 μs	2.73 ms
1	1	1	2.67 μs	683 μs

8-bit timer counter (channel 1)

Mode Register			8-bit Timer Counter (Channel 1)	
TM16	TM15	TM14	Resolution	Longest set time
0	1	1	5.33 μs	1.37 ms
1	0	0	683 μs	175 ms
1	0	1	171 μs	43.7 ms
1	1	0	42.7 μs	10.9 ms
1	1	1	10.7 μs	2.73 ms

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 6-7. Resolution and Longest Set Time (8-Bit Timer Counter Mode) (2/3)
(TM10 = 0, TM11 = 0, TM20 = 0, TM21 = 0)

8-bit timer counter (channel 2)

Mode Register			8-bit Timer Counter (Channel 2)	
TM26	TM25	TM24	Resolution	Longest set time
0	1	0	333 ns	85.3 μs
0	1	1	167 ns	42.7 μs
1	0	0	171 μs	43.7 ms
1	0	1	42.7 μs	10.9 ms
1	1	0	10.7 μs	2.73 ms
1	1	1	2.67 μs	683 μs

(b) μPD754244: at 4.19 MHz

8-bit timer counter (channel 0)

Mode Register			8-bit Timer Counter (Channel 0)	
TM06	TM05	TM04	Resolution	Longest set time
1	0	0	244 μs	62.5 ms
1	0	1	61.0 μs	15.6 ms
1	1	0	15.3 μs	3.91 ms
1	1	1	3.81 μs	977 μs

8-bit timer counter (channel 1)

Mode Register			8-bit Timer Counter (Channel 1)	
TM16	TM15	TM14	Resolution	Longest set time
0	1	1	7.63 μs	1.95 ms
1	0	0	977 μs	250 ms
1	0	1	244 μs	62.5 ms
1	1	0	61.0 μs	15.6 ms
1	1	1	15.3 μs	3.91 ms

8-bit timer counter (channel 2)

Mode Register			8-bit Timer Counter (Channel 2)	
TM26	TM25	TM24	Resolution	Longest set time
0	1	0	477 ns	122 μs
0	1	1	238 ns	61.0 μs
1	0	0	244 μs	62.5 ms
1	0	1	61.0 μs	15.6 ms
1	1	0	15.3 μs	3.91 ms
1	1	1	3.81 μs	977 μs

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 6-7. Resolution and Longest Set Time (8-Bit Timer Counter Mode) (3/3)
(TM10 = 0, TM11 = 0, TM20 = 0, TM21 = 0)

(c) μPD754144: at 1.0 MHz

8-bit timer counter (channel 0)

Mode Register			8-bit Timer Counter (Channel 0)	
TM06	TM05	TM04	Resolution	Longest set time
1	0	0	1024 μs	262ms
1	0	1	256 μs	65.5 ms
1	1	0	64 μs	16.4 ms
1	1	1	16 μs	4.10 μs

8-bit timer counter (channel 1)

Mode Register			8-bit Timer Counter (Channel 1)	
TM16	TM15	TM14	Resolution	Longest set time
0	1	1	32 μs	8.19 ms
1	0	0	4096 μs	1049 ms
1	0	1	1024 μs	262 ms
1	1	0	256 μs	65.5 ms
1	1	1	64 μs	16.4 ms

8-bit timer counter (channel 2)

Mode Register			8-bit Timer Counter (Channel 2)	
TM26	TM25	TM24	Resolution	Longest set time
0	1	0	2 μs	512 μs
0	1	1	1 μs	256 μs
1	0	0	1024 μs	262ms
1	0	1	256 μs	65.5 ms
1	1	0	64 μs	16.4 ms
1	1	1	16 μs	4.10 ms

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Timer counter operation (8-bit)

The timer counter operates as follows.

Figure 6-34 shows the configuration when the timer counter operates.

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

<2> The contents of Tn are compared with those of the modulo register (TMODn). When the contents of these registers match, 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) is inverted.

Figure 6-35 shows the timing of the timer counter operation.

The timer counter operation is usually started using the following procedure.

<1> Set the number of counts to TMODn.

<2> Sets the operation mode, count pulse, and start command to TMn.

Caution Set a value other than 00H to the timer counter modulo register (TMODn).

To use the timer counter output pin (PTOn), set the P3n pin as follows.

<1> Clear the output latch of P3n.

<2> Set port 3 in the output mode.

<3> Disconnect the on-chip pull-up resistor from port 3.

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

Remark n = 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-34. Configuration When Timer Counter Operates

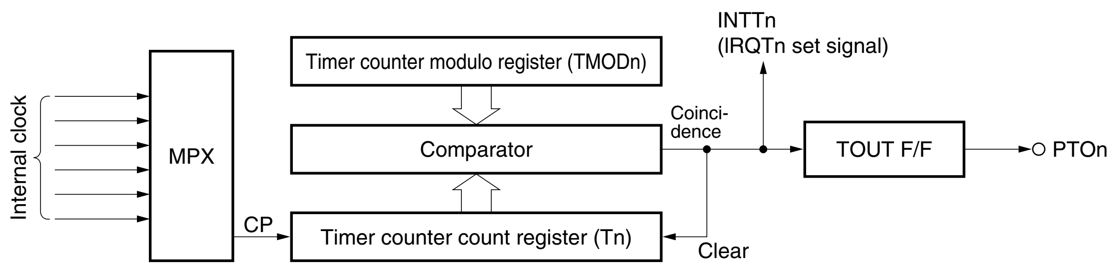
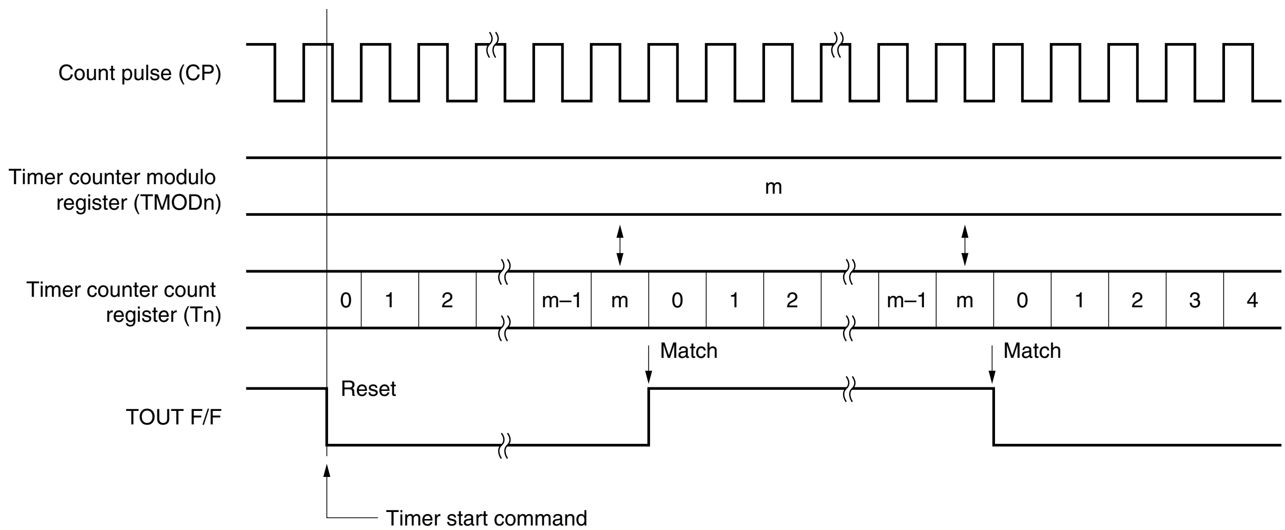


Figure 6-35. Count Operation Timing



Remark m: Set value of timer counter modulo register

n : 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Application of 8-bit timer counter mode

As an interval timer that generates an interrupt at 50 ms intervals[Note]

- Set the higher 4 bits of the timer counter mode register (TMn) to 0100B, and select 62.5 ms (at f_X = 4.19 MHz of μPD754244) as the longest set time.
- Set the lower 4 bits of TMn to 1100B.
- The set value of the timer counter modulo register (TMODn) is as follows:

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

<Program example>

```
SEL   MB15          ; or CLR1 MBE
MOV   XA, #0CCH
MOV   TMODn, XA     ; Sets modulo
MOV   XA, #01001100B
MOV   TMn, XA       ; Sets mode and starts timer
EI                  ; Enables interrupt
EI    IETn          ; Enables timer interrupt
```

Note This example applies to the operation of the μPD754244 at f_X = 4.19 MHz. With f_X = 6.0 MHz operation of the μPD754244 and f_{CC} = 1.0 MHz operation of the μPD754144, the longest set time and the interval time are different even if the settings are the same.

Remark n = 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

6.4.3 Operation in PWM pulse generator mode (PWM mode)

In this mode, the timer counter (channel 2) is used as a PWM pulse generator.

The timer counter operates as an 8-bit PWM pulse generator.

When the timer counter (channel 2) is used as a PWM pulse generator, the timer counters (channel 0 and 1) can be used as 8-bit timer counter.

(1) Register setting

In the PWM mode, the following five registers are used.

- Timer counter mode register (TM2)
- Timer counter control register (TC2)
- Timer counter count register (T2)
- Timer counter high-level period setting modulo register (TMOD2H)
- Timer counter modulo register (TMOD2)

(a) Timer counter mode register (TM2)

In the PWM mode, set TM2 as shown in Figure 6-36 (for the format of TM2, refer to **Figure 6-28 Format of Timer Counter Mode Register (Channel 2)**).

TM2 is manipulated by an 8-bit manipulation instruction. Bit 3 is a timer start command bit which can be manipulated in 1-bit units and is automatically cleared to 0 when the timer starts operating.

TM2 is also cleared to 00H when the internal reset signal is asserted.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-36. Setting of Timer Counter Mode Register

Address	7	6	5	4	3	2	1	0	Symbol
F90H	–	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) select bit

TM26	TM25	TM24	Count pulse (CP)
0	1	0	fx/2
0	1	1	fx
1	0	0	$fx/2^{10}$
1	0	1	$fx/2^{8}$
1	1	0	$fx/2^{6}$
1	1	1	$fx/2^{4}$
Other			Setting prohibited

Timer start command bit

TM23	Clears counter and IRQT2 flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TM22	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM21	TM20	Mode
0	1	PWM pulse generator mode

Remark When the timer counter (channel 2) is used as the PWM pulse generator mode, set the operation mode select bits TM10 and TM11 of the time counter (channel 1) to 0.

Download from Www.Somanuals.com. All Manuals Search And Download.

(b) Timer counter control register (TC2)

In the PWM mode, set TC2 as shown in Figure 6-37 (for the format of TC2, refer to **Figure 6-30 Format of Timer Counter Control Register)**.

TC2 is manipulated by an 8-, 4-, or bit manipulation instruction.

TC2 is cleared to 00H when the internal reset signal is asserted.

The flags shown by a solid line in the figure below are used in the PWM mode.

Do not use the flags shown by a dotted line in the PWM mode (set these flags to 0).

Figure 6-37. Setting of Timer Counter Control Register

7	6	5	4	3	2	1	0	Symbol
0	–	–	–	TOE2	REMC	NRZB	NRZ	TC2

Timer counter output enable flag

TOE2	Timer output
0	Disabled (low-level output)
1	Enabled

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) PWM pulse generator operation

The timer counter (channel 2) in PWM pulse generator mode has two registers, a high-level period setting timer counter modulo register (TMOD2H) and a low-level period setting timer counter modulo register (TMOD2). Figure 6-38 shows the PWM pulse generator configuration.

Each modulo register inverts its signal when the time set to each elapses. Therefore, pulses output from the PTO0 pin can be set arbitrarily for each modulo register.

The PWM pulse generator operates as follows. It repeats <2> and <3>, generating pulses until operation stops.

<1> A count pulse (CP) is selected by the timer counter mode register (TM2), and is input to the timer counter count register (T2).

<2> The contents of T2 are compared with those of the high-level period setting timer counter modulo register (TMOD2H). If the contents of the two registers match, a match signal is generated, and the timer output flip-flop (TOUT F/F) is inverted.
The count compare modulo register is switched to the low-level period setting timer counter modulo register (TMOD2).

<3> The contents of T2 are compared with those of the timer counter modulo register (TMOD2). When the contents of the two registers match, a match signal is generated, and an interrupt request flag (IRQT2) is set. At the same time, TOUT F/F is inverted. Then the count compare modulo register is switched to the high-level period setting timer counter modulo register (TMOD2H).

<4> The operations <2> and <3> are alternately repeated, and pulse wave form is generated.

Figure 6-39 shows the timing of the PWM pulse generator operation.

The PWM pulse generator operation is usually started in the following procedure.

<1> Set the number of counts of high-level width to TMOD2H.

<2> Set the number of counts of low-level width to TMOD2.

<3> Set an operation mode, count pulse, and start command to TM2.

Caution Set a value other than 00H to the timer counter modulo register (TMOD2) and high-level period setting timer counter modulo register (TMOD2H).

To use the timer counter output pin (PTO2), set the P32 pin as follows.

<1> Clear the output latch of P32.

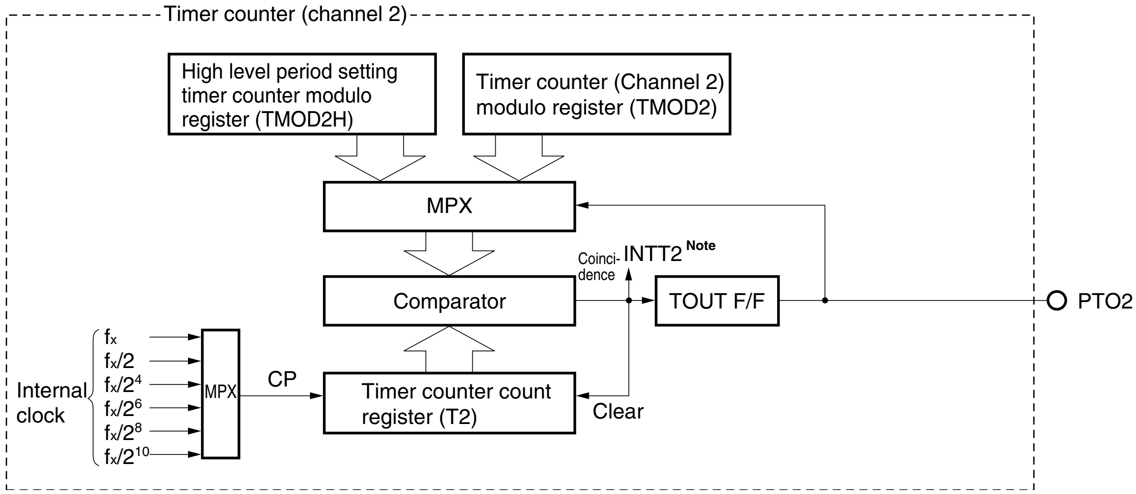
<2> Set port 3 in the output mode.

<3> Disconnect the pull-up resistor from port 3.

<4> Set the timer counter output enable flag (TOE2) to 1.

Download from Www.Somanuals.com. All Manuals Search And Download.

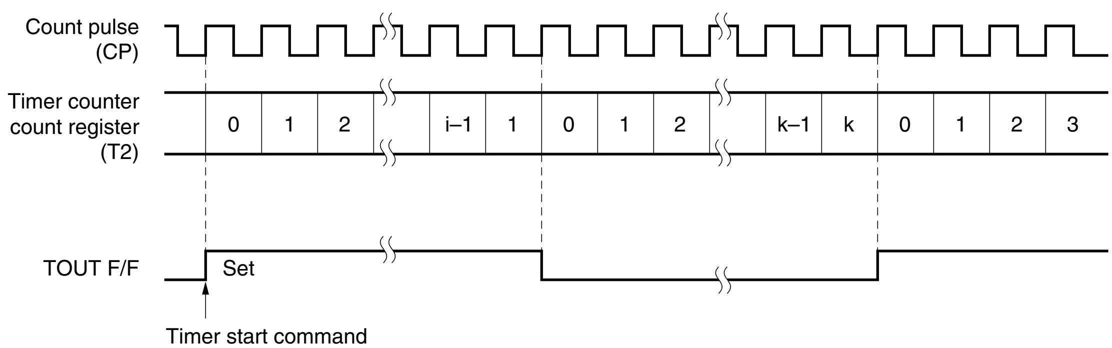
Figure 6-38. PWM Pulse Generator Operating Configuration



Note This is the IRQT2 set signal. It is only set when TMOD2 matches T2.

Figure 6-39. PWM Pulse Generator Operating Timing

Timer counter (channel 2) operation and carrier clock
(Modulo register H (TMOD2H) = 1, modulo register (TMOD2) = k)



Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Application of PWM mode

To output a pulse with a frequency of 38.0 kHz (cycle of 26.3 μs) and a duty factor of 1/3 to the PTO2 pin[Note]

- Set the higher 4 bits of the timer counter mode register (TM2) to 0011B and select 61.0 μs as the longest set time.
- Set the lower 4 bits of TM2 to 1101B, and select the PWM mode and count operation, and issue the timer start command.
- Set the timer counter output enable flag (TOE2) to "1" to enable timer output.
- Set the high-level period setting timer counter modulo register (TMOD2H) as follows.

$$\frac{1}{3} \cdot \frac{26.3\ \mu s}{238\ ns} - 1 = 36.8 - 1 \fallingdotseq 36 = 24H$$

- The set value of the timer counter modulo register (TMOD2) is as follows.

$$\frac{2}{3} \cdot \frac{26.3\ \mu s}{238\ ns} - 1 = 73.7 - 1 \fallingdotseq 73 = 49H$$

<Program example>

```
SEL     MB15            ; or CLR1 MBE
SET1    TOE2            ; Enables timer output
MOV     XA, #024H
MOV     TMOD2H, XA      ; Sets modulo (high-level period)
MOV     XA, #49H
MOV     TMOD2, XA       ; Sets modulo (low-level period)
MOV     XA, #00111101B
MOV     TM2, XA         ; Sets mode and starts timer
```

Note This example applies to the operation of the μPD754244 at f_X = 4.19 MHz. With f_X = 6.0 MHz operation of the μPD754244 and f_{CC} = 1.0 MHz operation of the μPD754144, the cycles are different even if the settings are the same.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.4.4 Operation in 16-bit timer counter mode

In this mode, two timer counter channels, 1 and 2, are used in combination to implement 16-bit programmable interval timer or event timer operation.

(1) Register setting

In the 16-bit timer counter mode, the following seven registers are used.

- Timer counter mode registers TM1 and TM2
- Timer counter control register TC2**Note**
- Timer count registers T1 and T2
- Timer count modulo registers TMOD1 and TMO2

Note Timer counter channel 1 uses the timer counter output enable flag (TOE1).

(a) Timer counter mode registers (TM1 and TM2)

In the 16-bit timer counter mode, TM1 and TM2 are set as shown in Figure 6-40 (for the formats of TM1 and TM2, refer to **Figure 6-27 Format of Timer Counter Mode Register (Channel 1)** and **Figure 6-28 Format of Timer Counter Mode Register (Channel 2)**).

TM1 and TM2 are manipulated by an 8-bit manipulation instruction. Bit 3 of these registers is a timer start command bit that can be manipulated in 1-bit units and is automatically cleared to 0 when the timer starts operating.

TM1 and TM2 are cleared to 00H when the internal reset signal is asserted.

The flags shown by a solid line in Figure 6-39 are used in the 16-bit timer counter mode.

Do not use the flags shown by a dotted line in the 16-bit timer counter mode (clear these flags to 0).

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-40. Setting of Timer Counter Mode Registers

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	–	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1
F90H	–	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) select bit

TMn6	TMn5	TMn4	TM1	TM2
0	1	0	Overflow of count register (T2)	fx/2
0	1	1	$fx2^{5}$	fx
1	0	0	$fx/2^{12}$	$fx/2^{10}$
1	0	1	$fx/2^{10}$	$fx/2^{8}$
1	1	0	$fx/2^{8}$	$fx/2^{6}$
1	1	1	$fx/2^{6}$	$fx/2^{4}$
Other			Setting prohibited	

Timer start command bit

TM23	Clears counter and IRQTn flag when "1" is written. Starts count operation if bit 2 is set to "1".

Remark n = 1 and 2

Operation mode

TM22	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM21	TM20	TM11	TM10	Mode
1	0	1	0	16-bit timer counter mode

Download from Www.Somanuals.com. All Manuals Search And Download.

(b) Timer counter control register (TC2)

In the 16-bit timer counter mode, set TC2 as shown in Figure 6-41 (for the format of TC2, refer to **Figure 6-30 Format of Timer Counter Control Register**).

TC2 is manipulated by an 8-, 4-, or bit manipulation instruction.

TC2 is cleared to 00H when the internal reset signal is asserted.

The flags shown by a solid line in Figure 6-40 are used in the 16-bit timer counter mode.

Do not use the flags shown by a dotted line in the 16-bit timer counter mode (clear these flags to 0).

Figure 6-41. Setting of Timer Counter Control Register

7	6	5	4	3	2	1	0	Symbol
0	–	–	–	TOE2	REMC	NRZB	NRZ	TC2

Timer counter output enable flag

TOE2	Timer Output
0	Disabled (low level output)
1	Enabled

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Time setting of timer counter

[Timer set time] (cycle) is calculated by dividing [contents of modulo register + 1] by [count pulse (CP) frequency] selected by the mode register.

$$T \text{ (sec)} = \frac{n+1}{f_{CP}} = (n+1) \cdot \text{(resolution)}$$

where,

T (sec): Timer set time (seconds)

f_{CP} (Hz): CP frequency (Hz)

n: Contents of modulo register ($n \neq 0$)

Once the timer has been set, interrupt request flag IRQT2 is set at the set time interval of the timer.

Table 6-8 shows the resolution of each count pulse of the timer counter and the longest set time (time when FFH is set to the modulo registers 1 and 2).

Table 6-8. Resolution and Longest Set Time (16-Bit Timer Counter Mode) (1/2)
(TM10 = 0, TM11 = 1, TM20 = 0, TM21 = 1)

(a) μPD754144: at 1.0 MHz

Mode Register			16-Bit Timer Counter	
TM26	TM25	TM24	Resolution	Longest Set Time
0	1	0	2 μs	131 ms
0	1	1	1 μs	65.5 ms
1	0	0	1024 μs	67.1 s
1	0	1	256 μs	16.8 s
1	1	0	64 μs	4.19 s
1	1	1	16 μs	1.05 s

(b) μPD754244: at 6.0 MHz

Mode Register			16-Bit Timer Counter	
TM26	TM25	TM24	Resolution	Longest Set Time
0	1	0	333 ns	21.8 ms
0	1	1	167 ns	10.9 ms
1	0	0	171 μs	11.2 s
1	0	1	42.7 μs	2.80 s
1	1	0	10.7 μs	699 ms
1	1	1	2.67 μs	175 ms

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 6-8. Resolution and Longest Set Time (16-Bit Timer Counter Mode) (2/2)
(TM10 = 0, TM11 = 1, TM20 = 0, TM21 = 1)

(c) μPD754244: at 4.19 MHz

Mode Register			16-Bit Timer Counter	
TM26	TM25	TM24	Resolution	Longest Set Time
0	1	0	477 ns	31.3 ms
0	1	1	238 ns	15.6 ms
1	0	0	244 μs	16.0 s
1	0	1	61.0 μs	4.0 s
1	1	0	15.3 μs	1.0 s
1	1	1	3.81 μs	250 ms

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Timer counter operation (at 16-bit)

The timer counter operates as follows.

Figure 6-42 shows the configuration when the timer counter operates.

<1> The count pulse (CP) is selected by timer counter mode registers TM1 and TM2 and is input to timer counter count register T2. The overflow of T2 is input to count register T1.

<2> The contents of T1 are compared with those of timer counter modulo register TMOD1. When the contents of these registers match, a match signal is generated.

<3> The contents of T2 are compared with those of timer counter modulo register TMOD2. When the contents of these registers match, a match signal is generated.

<4> If the match signals in <2> and <3> overlap, interrupt request flag IRQT2 is set. At the same time, timer out flip-flop TOUT F/F is inverted.

Figure 6-43 shows the operation timing of the timer counter operation.

The timer counter operation is usually started by the following procedure.

<1> Set the higher 8 bits of the number of counts indicated as 16 bits wide to TMOD1.

<2> Set the lower 8 bits of the number of counts indicated as 16 bits wide to TMOD2.

<3> Set the count pulse to TM1.

<4> Set the operation mode, count pulse, and start command to TM2.

Caution Be sure to set a value other than 00H to timer counter modulo register TMOD2. Also, set "0" to IET1.

To use timer counter output pin PTO2, set the P32 pin as follows:

<1> Clear the output latch of P32.

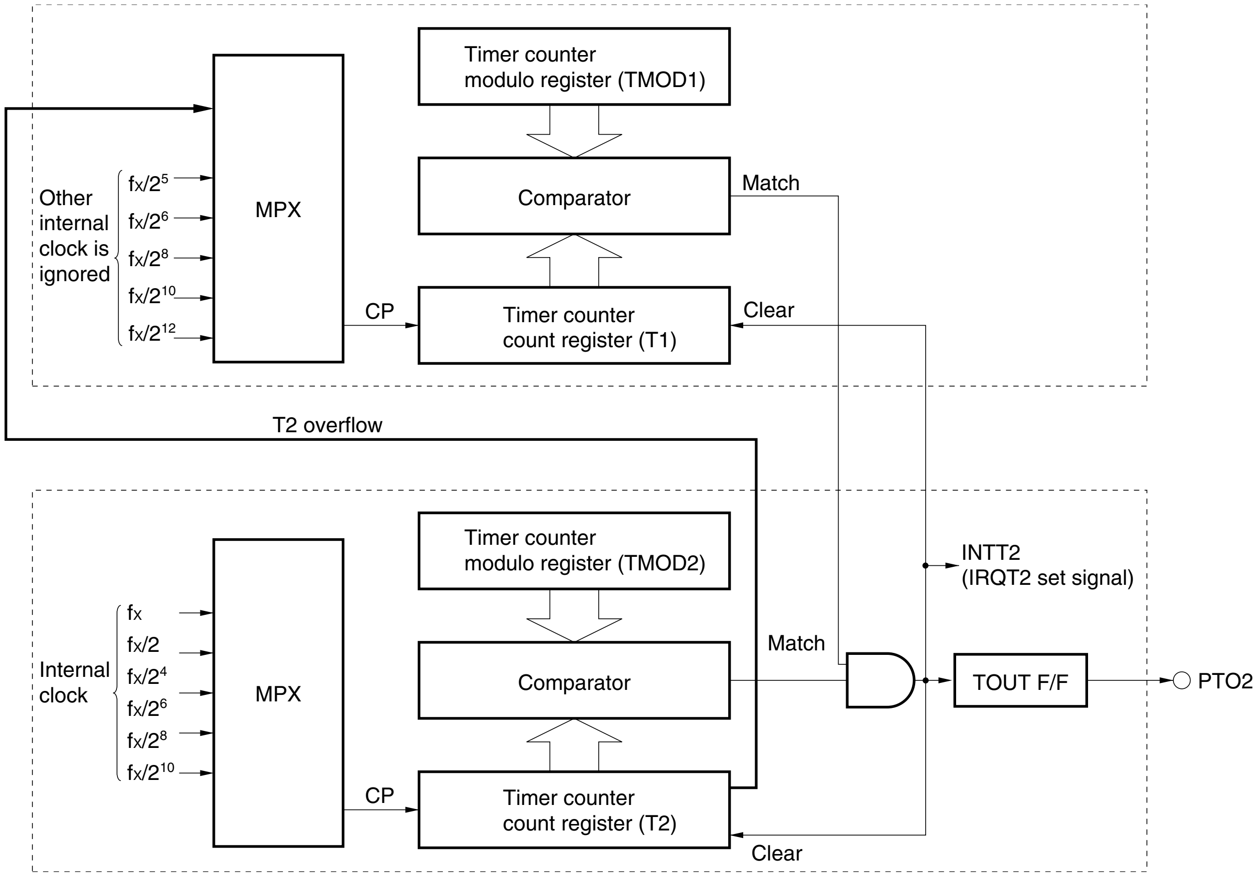
<2> Set port 3 in the output mode.

<3> Disconnect the internal pull-up resistor from port 3.

<4> Set timer counter output enable flag TOE2 to 1.

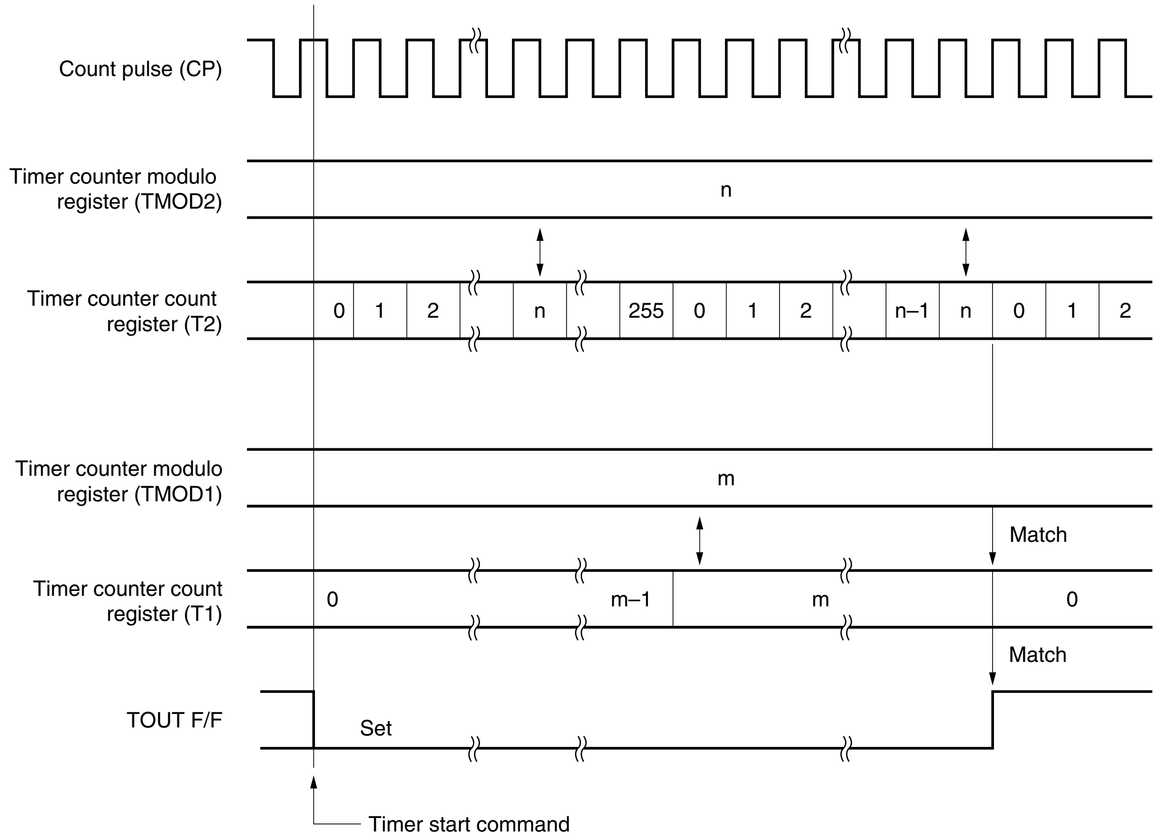
Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-42. Configuration When Timer Counter Operates



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-43. Timing of Count Operation



Remark m: Set value of timer counter module register (TMOD1)

n: Set value of timer counter modulo register (TMOD2)

Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Application of 16-bit timer counter mode

As an interval timer that generates an interrupt at 5-second intervals[Note]

- Set the higher 4 bits of the mode register (TM1) to 0010B, and select the overflow of timer counter count register (T2).
- Set the higher 4 bits of TM2 to 0100B and select 16.0 second as the longest set time.
- Set the lower 4 bits of TM1 to 0010B and select the 16-bit timer counter mode.
- Set the lower 4 bits of TM2 to 1110B, select the 16-bit timer counter mode and count operation. Then, issue the timer start command.
- The set values of the timer counter modulo registers (TMOD1 and TMOD2) are as follows.

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

<Program example>

```
SEL     MB15           ;  or CLR1 MBE
MOV     XA, #050H
MOV     TMOD1, XA      ;  Sets modulo (higher 8 bits)
MOV     XA, #00B
MOV     TMOD2, XA      ;  Sets modulo (lower 8 bits)
MOV     XA, #00100010B
MOV     TM1, XA        ;  Sets mode
MOV     XA, #01001110B
MOV     TM2, XA        ;  Sets mode and starts timer
DI      IET1           ;  Disables timer (channel 1) interrupt
EI                     ;  Enables interrupts
EI      IET2           ;  Enables timer (channel 2) interrupt
```

Note This example applies to the operation of the μPD754244 at f_X = 4.19 MHz. With f_X = 6.0 MHz operation of the μPD754244 and f_{CC} = 1.0 MHz operation of the μPD754144, the longest set time and interval time are different even if the settings are the same.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.4.5 Operation in carrier generator mode (CG mode)

In the PWM mode, timer counter channels 1 and 2 operate in combination to implement an 8-bit carrier generator operation.

When using CG mode, use it in combination with channel 1 and channel 2 of the timer counter.

Timer counter channel 1 generates a remote controller signal.

Timer counter channel 2 generates a carrier clock.

(1) Register setting

In the CG mode, the following eight registers are used.

- Timer counter mode registers TM1 and TM2
- Timer counter control register TC2[Note]
- Timer counter count registers T1 and T2
- Timer counter modulo registers TMOD1 and TMOD2
- Timer counter high-level period setting modulo register TMOD2H

Note Timer counter channel 1 uses the timer counter output enable flag (TOE1).

(a) Timer counter mode registers (TM1 and TM2)

In the CG mode, set TM1 and TM2 as shown in Figure 6-44 (for the formats of TM1 and TM2, refer to **Figure 6-27 Format of Timer Counter Mode Register (Channel 1)** and **Figure 6-28 Format of Timer Counter Mode Register (Channel 2)**).

TM1 and TM2 are manipulated by an 8-bit manipulation instruction. Bit 3 of TM1 and TM2 is timer start command bit which can be manipulated in 1-bit units and is automatically cleared to 0 when the timer starts operating.

TM1 and TM2 are also cleared to 00H when the internal reset signal is asserted.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-44. Setting of Timer Counter Mode Register (n = 1, 2)

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	–	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1
F90H	–	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) select bit

TMn6	TMn5	TMn4	TM1	TM2
0	1	0	Carrier clock input	$f_X/2$
0	1	1	$f_X/2^5$	f_X
1	0	0	$f_X/2^{12}$	$f_X/2^{10}$
1	0	1	$f_X/2^{10}$	$f_X/2^8$
1	1	0	$f_X/2^8$	$f_X/2^6$
1	1	1	$f_X/2^6$	$f_X/2^4$
Other			Setting prohibited	

Timer start command bit

TMn3	Clears counter and IRQTn flag when "1" is written. Starts count operation if bit 2 is set to "1".

Operation mode

TMn2	Count operation
0	Stops (count value retained)
1	Count operation

Operation mode select bit

TM21	TM20	TM11	TM10	Mode
1	1	0	0	Carrier generator mode

Remark n = 1, 2

Download from Www.Somanuals.com. All Manuals Search And Download.

(b) Timer counter control register (TC2)

In the CG mode, set the timer counter output enable flag (TOE1) and TC2 as shown in Figure 6-45 (for the format of TC2, refer to **Figure 6-30 Format of Timer Counter Control Register**).

TOE1 is manipulated by a bit manipulation instruction. TC2 is manipulated by an 8-, 4-, or bit manipulation instruction.

TOE1 and TC2 are cleared to 00H when the internal reset signal is asserted.

The flags shown by a solid line in the figure below are used in the CG mode.

Do not use the flags shown by a dotted line in the CG mode (clear these flags to 0).

Figure 6-45. Setting of Timer Counter Output Enable Flag

Address	
FAAH	TOE1

Timer counter output enable flag (W)

TOE1	
0	Disabled
1	Enabled

Figure 6-46. Setting of Timer Counter Control Register

7	6	5	4	3	2	1	0	Symbol
0	–	–	–	TOE2	REMC	NRZB	NRZ	TC2

Remote controller output control flag

REMC	Remote controller output
0	Outputs carrier pulse to PTO2 pin when NRZ = 1
1	Outputs high level to PTO2 pin when NRZ = 1

No return zero buffer flag

NRZB	Area to store no return zero data to be output next. Transferred to NRZ when timer counter (channel 1) interrupt occurs

No return zero flag

NRZ	No return zero data
0	Outputs low level to PTO2 pin (Carrier clock stopped)
1	Outputs carrier pulse to PTO2 pin

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Carrier generator operation

The carrier generator operation is performed as follows. Figure 6-47 shows the configuration of the timer counter in the carrier generator mode.

(a) Timer counter (channel 1) operation

The timer counter (channel 1) in carrier generator mode determines the time required to output the carrier clock generated by the timer counter (channel 2) to the PTO2 pin, and the time to stop the output.
Moreover, the overflow time of the timer counter (channel 1) determines the interval of loading from the no return zero buffer flag (NRZB) of the timer counter (channel 2) to the no return zero flag (NRZ).

<1> A count pulse (CP) is selected by the timer counter mode register (TM1), and is input to the timer counter count register (T1).

<2> The contents of T1 are compared with those of the timer counter modulo register (TMOD1). When the contents of the two registers match, an interrupt request flag (IRQT1) is set. At the same time, the timer out flip-flop (TOUT F/F) is inverted, and generates reload signal from NRZB to NRZ.

(b) Timer counter (channel 2) operation

The timer counter (channel 2) in carrier generator mode generates the carrier clock to be output to the PTO2 pin.
Moreover, according to an overflow signal of the timer counter (channel 1), it reloads from the no return zero buffer flag (NRZB) to the no return zero flag (NRZ).
NRZ determines whether the carrier clock generated should be output to the PTO2 pin or not.

Operation of the timer counter (channel 2) is carried out according to the following procedure. The timer counter repeats <2> and <3>, generating carrier waves until operation stops.

<1> A count pulse (CP) is selected by the timer counter mode register (TM2), and is input to the timer counter count register (T2).

<2> The contents of T2 are compared with those of the high-level period setting timer counter modulo register (TMOD2H). If the contents of the two registers match, a match signal is generated, and the timer output flip-flop (TOUT F/F) is inverted. At the same time, the count comparison modulo register is switched to the low-level period setting timer counter modulo register (TMOD2).

<3> The contents of T2 are compared with those of the timer counter modulo register (TMOD2). When the contents of the two registers match, a match signal is generated, and an interrupt request flag (IRQT2) is set. At the same time, TOUT F/F is inverted and the count comparison modulo register is switched to the high-level period setting timer counter modulo register (TMOD2H).

Download from Www.Somanuals.com. All Manuals Search And Download.

<4> The operations <2> and <3> are repeated.
<5> The no return zero data is reloaded from NRZB to NRZ when timer counter channel 1 generates an interrupt.
<6> A carrier clock or high level is output when NRZ is set to 1 by the remote controller output flag (REMC). When NRZ = 0, a low level is output.

Figure 6-48 shows the timing of the carrier generator operation.
The carrier generator operation is usually started by the following procedure.
<1> Set the number of counts of high-level width of the carrier clock to TMOD2H.
<2> Set the number of counts of low-level width of the carrier clock to TMOD2.
<3> Set the output waveform to REMC.
<4> Set the operation mode, count pulse, and start command to TM2.
<5> Set the number of counts of the NRZ switching timing to TMOD1.
<6> Set the operation mode, count pulse, and start command to TM1.
<7> Set the no return zero data to be output next to NRZB before timer counter channel 1 generates an interrupt.

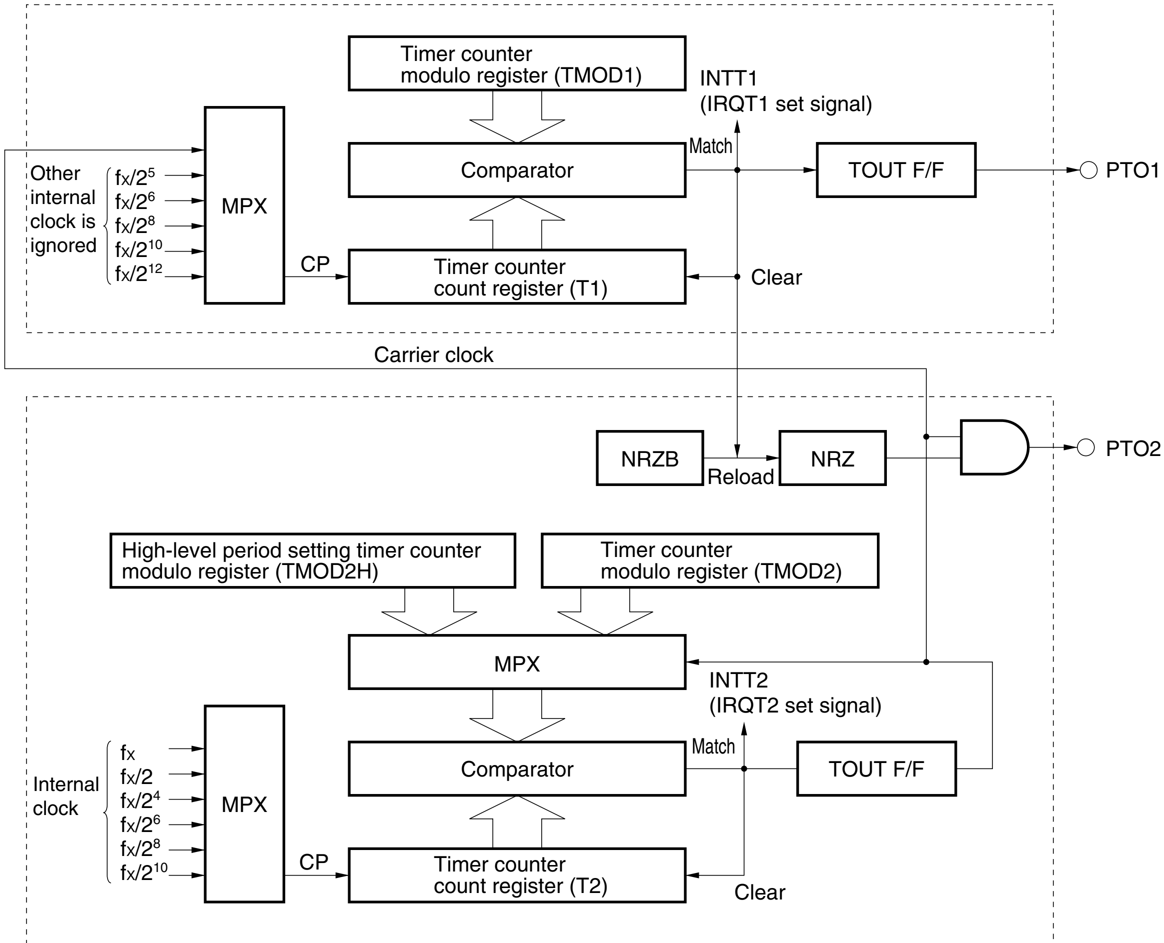
Caution Set a value other than 00H to the timer counter modulo registers (TMOD1, TMOD2, and TMOD2H).

To use the timer counter output pin (PTO1), set the P31 pin as follows.

<1> Clear the output latch of P31.
<2> Set port 3 in the output mode.
<3> Disconnect the internal pull-up resistor from port 3.
<4> Set the timer counter output enable flag (TOE1) to 1.

Download from Www.Somanuals.com. All Manuals Search And Download.

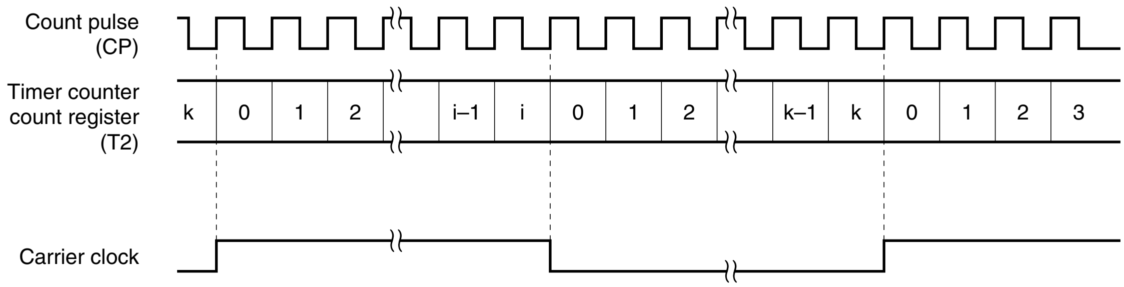
Figure 6-47. Configuration in Carrier Generator Mode



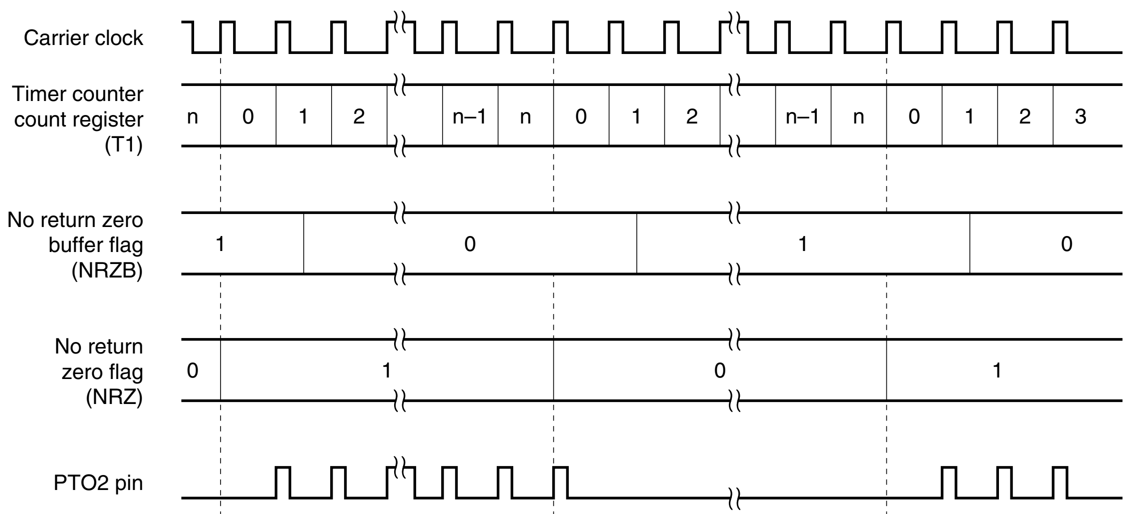
Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-48. Carrier Generator Operation Timing

<1> Timer (channel 2) operation and carrier clock
(Modulo register H (TMOD2H) = i, Modulo register (TMOD2) = k)

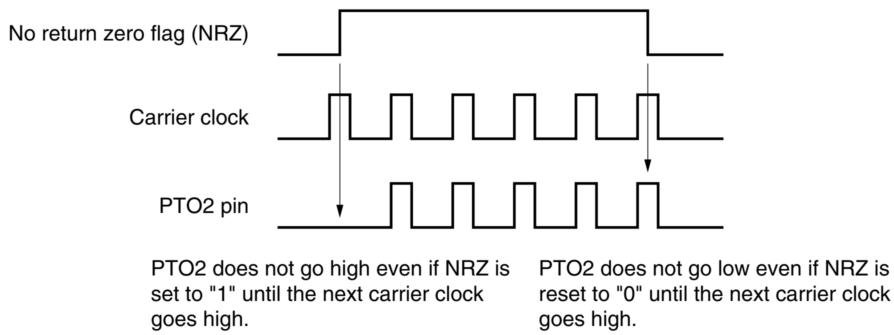


<2> Carrier clock, timer (channel 1), NRZB, NRZ, and PTO2 pin
(Modulo register (TMOD1) = n, Timer (channel 1) count pulse = Carrier clock)



Download from Www.Somanuals.com. All Manuals Search And Download.

Remark If a timer (channel 1) interrupt is generated when the PTO2 pin is low and the carrier clock is high (NRZ = 0, carrier clock = high level), the carrier is output to the PTO2 pin from the pulse after the carrier clock.
If a timer (channel 1) interrupt is generated when the PTO2 pin is high and the carrier clock is high (NRZ = 1, carrier clock = high level), the PTO2 pin does not become low until the end of the carrier clock being output.
This processing is performed to keep the width of the high-level pulse output from the PTO2 pin constant regardless of the NRZ switching timing (see figure below).



Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Application of CG mode

To use the timer counter as a carrier generator for remote controller signal transmission

The examples shown below apply to the operation of the μPD754244 at f_X = 4.19 MHz. With f_X = 6.0 MHz operation of the μPD754244 and f_{CC} = 1.0 MHz operation of the μPD754144, the cycles and signal output periods are different even if the settings are the same.

<1> To generate a carrier clock with a frequency of 38.0 kHz (cycle of 26.3 μs) and a duty factor of 1/3

- Set the higher 4 bits of the timer counter mode register (TM2) to 0011B and select 61.0 μs as the longest set time.
- Set the lower 4 bits of TM2 to 1111B, and select the CG mode and count operation. Then, issue the timer start command.
- Set the timer counter output enable flag (TOE2) to "1" to enable timer output.
- Set the high-level period setting timer counter modulo register (TMOD2H) as follows.

$$\frac{1}{3} \cdot \frac{26.3\ \mu s}{238\ ns} - 1 = 36.8 - 1 \fallingdotseq 36 = 24H$$

- The set value of the timer counter modulo register (TMOD2) is as follows.

$$\frac{2}{3} \cdot \frac{26.3\ \mu s}{238\ ns} - 1 = 73.7 - 1 \fallingdotseq 73 = 49H$$

<Program example>

```
SEL    MB15             ;  or CLR1 MBE
MOV    XA, #024H
MOV    TMOD2H, XA       ;  Sets modulo (high-level period)
MOV    XA, #49H
MOV    TMOD2, XA        ;  Sets modulo (low-level period)
MOV    XA, #00111111B
MOV    TM2, XA          ;  Sets mode and starts timer
```


Download from Www.Somanuals.com. All Manuals Search And Download.

<2> To output a leader code with a 9 ms period to output a carrier clock and a 4.5 ms period to output a low level (Refer to the figure below.)

- Set the higher 4 bits of the timer counter mode register (TM1) to 0110B and select 15.6 ms as the longest set time.
- Set the lower 4 bits of TM1 to 1100B. Then, select the 8-bit timer counter mode, count operation, and timer start command.
- The initial set value of the timer counter modulo register (TMOD1) is as follows.

$$\frac{9\text{ ms}}{61\ \mu\text{s}} - 1 = 147.5 - 1 \fallingdotseq 147 = 93\text{H}$$

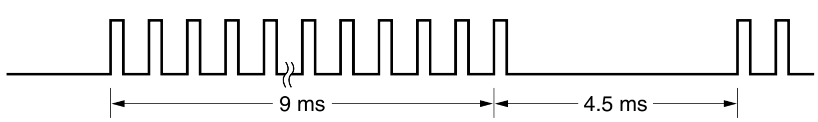
- The set value for rewriting TMOD1 is as follows.

$$\frac{4.5\text{ ms}}{61\ \mu\text{s}} - 1 = 73.8 - 1 \fallingdotseq 73 = 49\text{H}$$

- Set the higher 4 bits of TC2 to 0000B.
- Set the lower 4 bits of TC2 to 0000B. The carrier clock is output when no return zero data is "1", and the no return zero data to be output next is cleared to "0".

<Program example>

```
SEL     MB15            ;   or CLR1 MBE
MOV     XA, #093H
MOV     TMOD1, XA       ;   Sets modulo (carrier clock output period)
MOV     XA, #00000000B
MOV     TC2, XA
SET1    NRZ             ;   Sets no return zero data to "1"
MOV     XA, #01101100B
MOV     TM1, XA         ;   Sets mode and starts timer
EI                      ;   Enables interrupt
EI      IET1            ;   Enables interrupt of timer counter channel 1

; <Subroutine>
MOV     XA, #049H
MOV     TMOD1, XA       ;   Rewrites modulo (low-level output period)
RETI
```



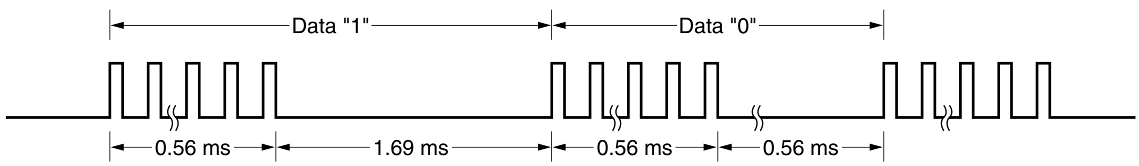
Download from Www.Somanuals.com. All Manuals Search And Download.

<3> To output a custom code with a 0.56 ms period to output a carrier clock when data is “1”, a 1.69 ms to output a low level, a 0.56 ms to output a carrier clock when data is “0”, and a 0.56 ms period to output a low level (refer to the figure below).

- Set the higher 4 bits of the timer counter mode register (TM1) to 0011B and select 1.95 ms as the longest set time.
- Set the lower 4 bits of TM1 to 1100B. Then, select the 8-bit timer counter mode, count operation, and timer start command.
- The initial set value of the timer counter modulo register (TMOD1) is as follows.

$$\frac{0.56 \text{ ms}}{7.64\ \mu\text{s}} - 1 = 73.3 - 1 = 72 = 48\text{H}$$

- During the period in which the carrier output of TMOD1 is not performed, processing is executed for the duration of the same as the output period when data is “0” and for the duration three times that of the output period when data is “1” (software repeats three times the period in which carrier output is not performed when data is “0”).
- Set the higher 4 bits of TC2 to 0000B.
- Set the lower 4 bits of TC2 to 0000B. The carrier clock is output when the no return zero data is “1”. The no return zero data to be output next is cleared to “0”.
- Set the transmit data (“0” or “1”) to the bit sequential buffer.



Download from Www.Somanuals.com. All Manuals Search And Download.

<Program example>

In the following example, it is assumed that the output latch of the PTO2 pin is cleared to “0” and that the output mode has been set. It is also assumed that the carrier clock is generated with the status of the program in the preceding example (2).

```
; SEND_CARIER_DATA_PRO
             SEL     MB15             ; or CLR1 MBE
             MOV     HL, #00H         ; Sets pointer of BSB (bit sequential buffer) to L.
                                        Uses H as bit data temporary saving area of BSB
; CG_Init & Send_1st_Data
             MOV     XA, #48H
             MOV     TMOD1, XA        ; Sets modulo register (carrier clock output period)
             MOV     XA, #00000000B   ; Enables output of carrier clock, and initializes NRZB and
                                        NRZ to 0
             MOV     TC2, XA
             SET1    NRZ              ; Sets no return zero flag to “1”
             MOV     XA, #01101100B   ; Selects count pulse and 8-bit timer counter mode
             MOV     TM1, XA          ; Enables timer counter operation and issues timer start
                                        command
; Send_1st_Data
             CALL    !GET_DATA        ; Gets data from BSB
             CALL    !SEND_D_0        ; Outputs carrier with data 0 and 1 and first low level output
                                        period setting processing
             SKE     H, #1H           ; If bit 0 is 1, proceeds to second additional processing of low
                                        level output period
             BR      SEND_1_F         ; If bit 0 is 0, outputs low level and transfers control to search
                                        of next data
             CALL    !SEND_D_1        ; Second additional processing of low level output period.
                                        Transfers control to data transmission processing of BSB
                                        bit 0-F with PTO2 pin outputting low
; SEND_1_F:                           ;    Data transmission processing of bit 0-F of BSB
             SET1    NRZB             ; Sets NRZB to 1 so that carrier of data to be transmitted next
                                        is output by IRQT1 generated next during low level output
                                        period of preceding data
             INCS    L                ; Counts data being transmitted and ends data transmission
                                        when L changes from 0FH to 0H
             BR      LOOP_C_0
             BR      SEND_END
LOOP_C_0:    SKTCLR  IRQT1            ; Waits for low level output of preceding data (confirmation
                                        of end of preceding data)
             BR      LOOP_C_0
                                      ; Starts carrier output
             CLR1    NRZB             ; Clears NRZB to 0 in advance so that first low level output
                                        is performed by IRQT1 generated next
             CALL    !GET_DATA
             CALL    !SEND_D_0
             SKE     H, #1H           ; If data obtained is 1, proceeds to second additional processing
                                        of low level output period (SEND_D_1)
```


Download from Www.Somanuals.com. All Manuals Search And Download.

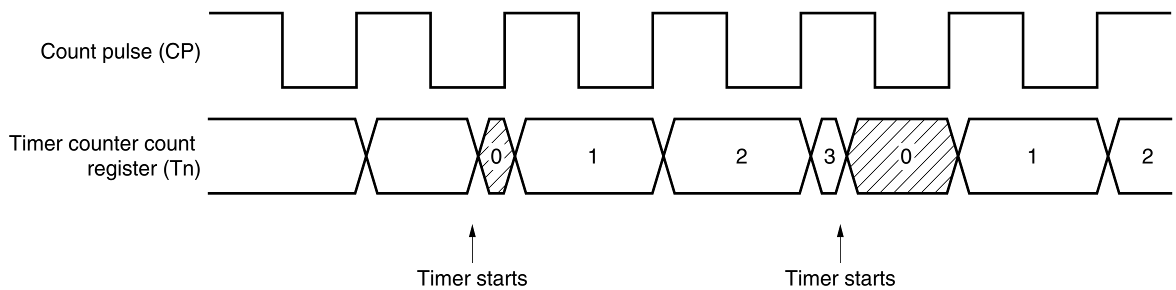
```
            BR      SEND_1_F      ; If data is 0, proceeds to transmission processing of next
                                    data with PTO2 pin outputting low level
            CALL    !SEND_D_1
            BR      SEND_1_F
SEND_END :                          ;    Completes transmission of 16 bits of data

; <subroutine>
GET_DATA:                         ; Searches data of BSB indicated by @L. Sets value to H
                                    register
            SKT     BSB0.@L
            MOV     A, #0
            MOV     A, #1
            MOV     H, A
            RET
SEND_D_0 :                        ; Processing to set carrier output of data 0 and 1 and first low
                                    level output
LOOP_1ST :SKTCLR    IRQT1
            BR      LOOP_1ST      ; Waits for carrier output
            RET                   ; Starts output of first low level
SEND_D_1 :
            CLR1    NRZB          ; Sets second low level output if data is 1
LOOP_2ND :  SKTCLR  IRQT1
            BR      LOOP_2ND      ; Waits for first low level output
                                  ; Starts second low level output
            CLR1    NRZB          ; Sets third low level output
LOOP_3RD :  SKTCLR  IRQT1
            BR      LOOP_3RD      ; Waits for second low level output
                                  ; Starts third low level output
            RET
```


Download from Www.Somanuals.com. All Manuals Search And Download.

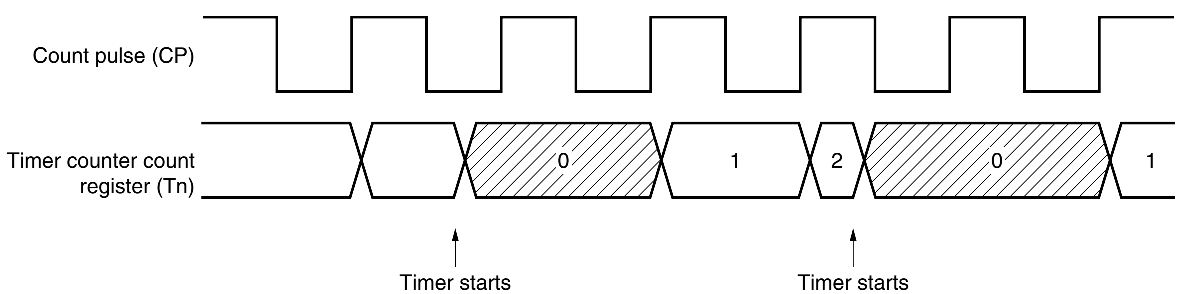
6.4.6 Notes on using timer counter

(1) Error when timer starts

After the timer has been started (bit 3 of TMn has been set to "1"), the time required for generation of the match signal, which is calculated by the expression (contents of modulo register + 1) × resolution, deviates by up to one clock of the count pulse (CP). This is because timer counter count register Tn is cleared asynchronously to CP, as shown below.



If the frequency of CP is greater than one machine cycle, the time required for generation of the match signal, which is calculated by the expression (modulo register contents + 1) × resolution, deviates by up to CP2 clock after the timer has been started (bit 3 of TMn has been set to "1"). This is because Tn is cleared asynchronously to CP, based on to CPU clock, as shown below.



Remark n = 0 to 2

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Note on starting timer

Usually, count register Tn and interrupt request flag IRQTn are cleared when the timer is started (bit 3 of TMn is set to "1"). However, if the timer is in an operation mode, and if IRQTn is set as soon as the timer is started, IRQTn may not be cleared. This does not pose any problem when IRQTn is used as a vector interrupt. In an application where IRQTn is being tested, however, IRQTn is not set after the timer has been started and this poses a problem. Therefore, there is a possibility that the timer could be started as soon as IRQTn is set to 1, either stop the timer once (by clearing the bit 2 of TMn to "0"), or start the timer two times.

Example If there is a possibility that timer could be started as soon as IRQTn is set

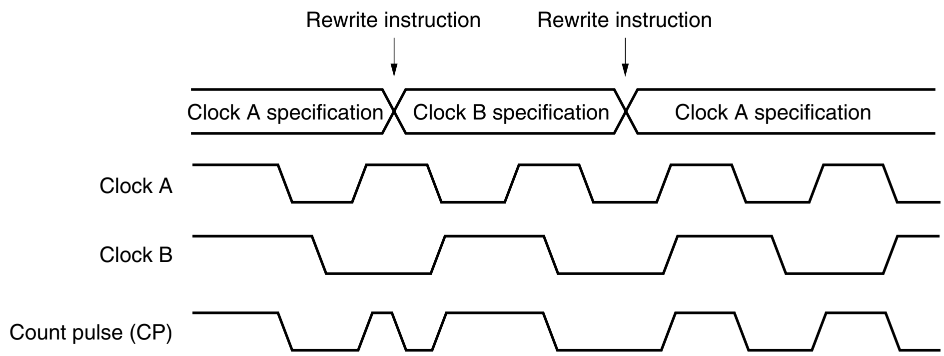
```
SEL     MB15
MOV     XA, #0
MOV     TMn, XA     ; Stops timer
MOV     XA, #4CH
MOV     TMn, XA     ; Restarts
Or,
SEL     MB15
SET1    TMn.3
SET1    TMn.3       ; Restarts
```

Remark n = 0 to 2

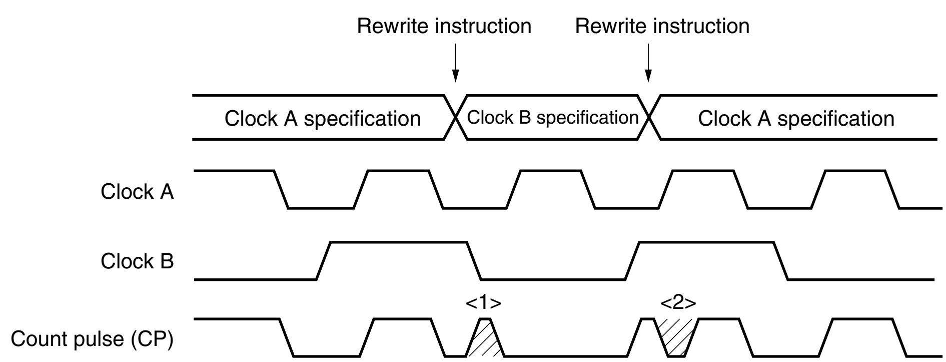
Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Notes on changing count pulse

When it is specified to change the count pulse (CP) by rewriting the contents of the timer counter mode register (TMn), the specification becomes valid immediately after execution of the instruction that commands the specification.



A whisker-like CP (<1> or <2 > in the figure below) may be generated depending on the combination of the clocks for changing CP. In this case, a miscount may occur or the contents of the count register (Tn) may be destroyed. To change CP, be sure to set the bit 3 of TMn bit to “1” and restart the timer at the same time.

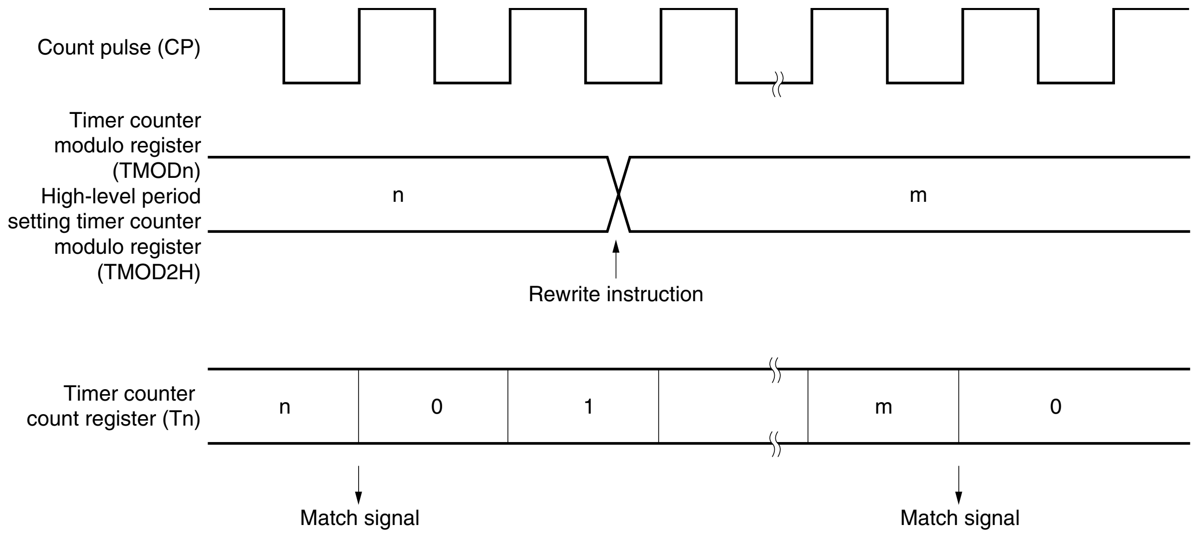


Remark n = 0 to 2

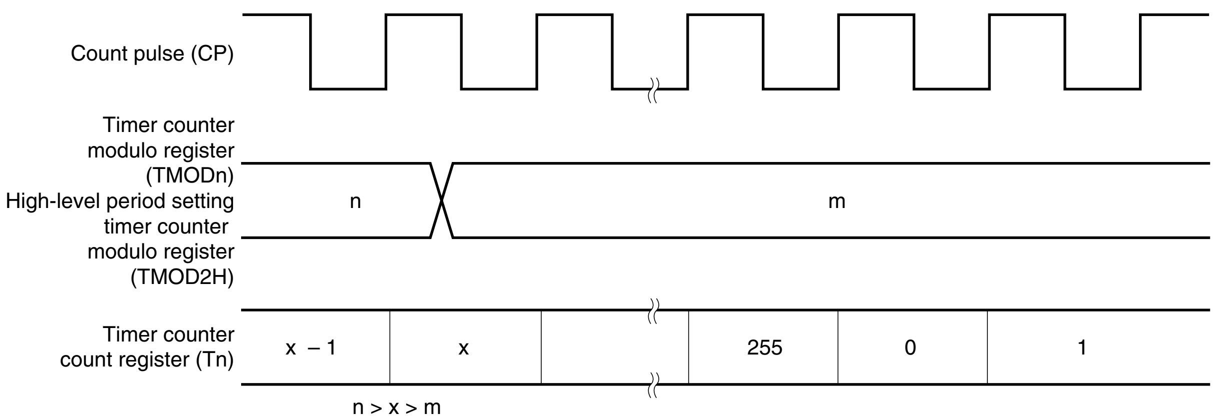
Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Operation after changing modulo register

The contents of the timer counter modulo register (TMODn) and high-level period setting timer counter modulo register (TMOD2H) are changed as soon as an 8-bit data memory manipulation instruction has been executed.



If the value of TMODn after the change is less than the value of the timer counter count register (Tn), Tn continues counting. When an overflow occurs, Tn starts counting again from 0. If the values of TMODn and TMOD2H after the change are less than the values before change (n), it is necessary to restart the timer after changing TMODn and TMOD2H.

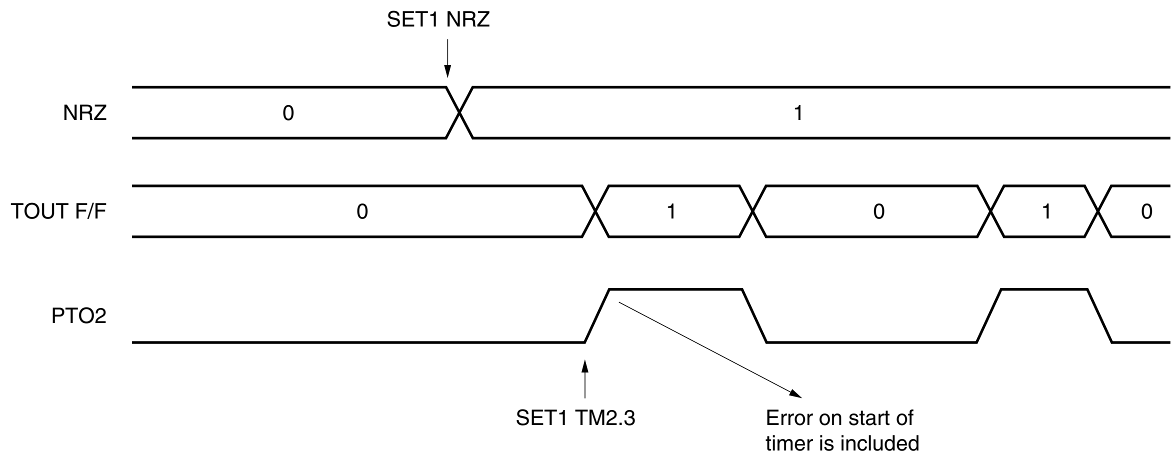


Download from Www.Somanuals.com. All Manuals Search And Download.

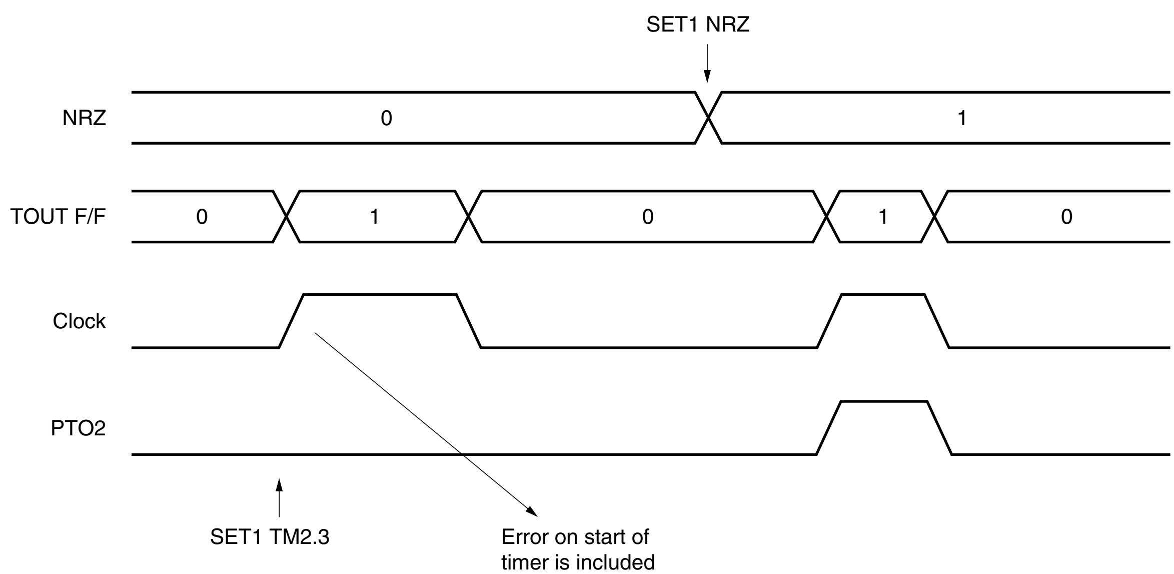
(5) Note on application of carrier generator (on starting)

When the carrier clock is generated, after the timer has been started (by setting bit 3 of TM2 to “1”), the high-level period of the initial carrier clock may deviate by up to one clock of the count pulse (CP) (up to two clocks of CP if the frequency of CP is higher than one machine cycle) from the value calculated by the expression (contents of modulo register + 1) × resolution (for details, refer to **(1) Error when timer starts**).

To output a carrier as the initial code, if the timer is started (by setting bit 3 of TM2 to “1”) after the no return zero flag (NRZ) has been set to “1”, the high-level period of the initial carrier clock includes the possibility of an error that may occur when the timer is started.



Therefore, to output a carrier as the initial code, set NRZ to “1” after the timer has been started (by setting bit 3 of TM2 to “1”).



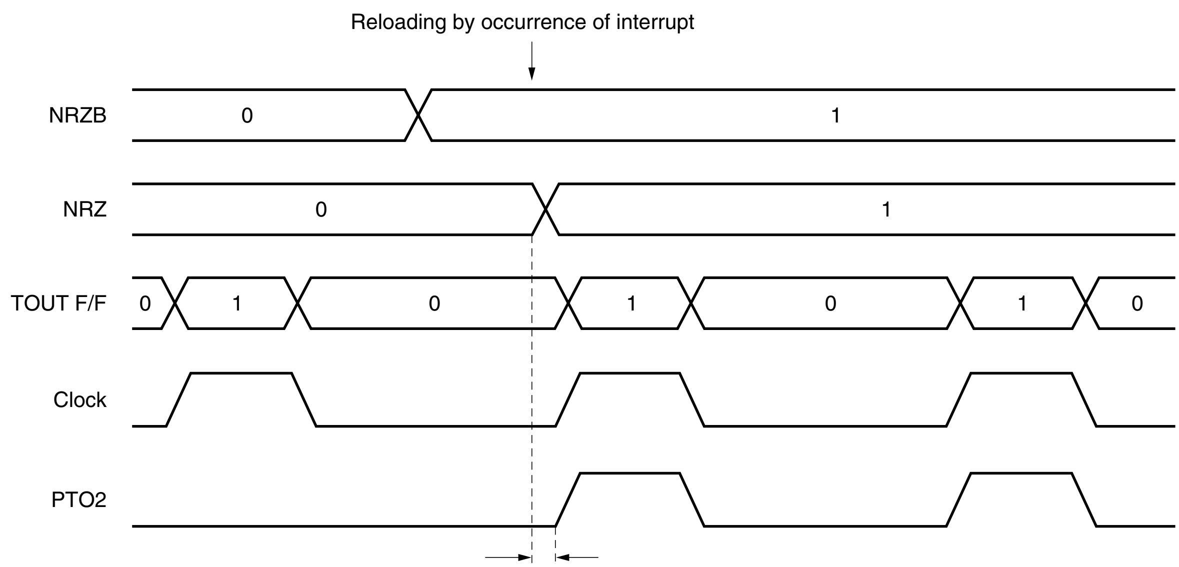
Download from Www.Somanuals.com. All Manuals Search And Download.

(6) Notes on application of carrier generator (reload)

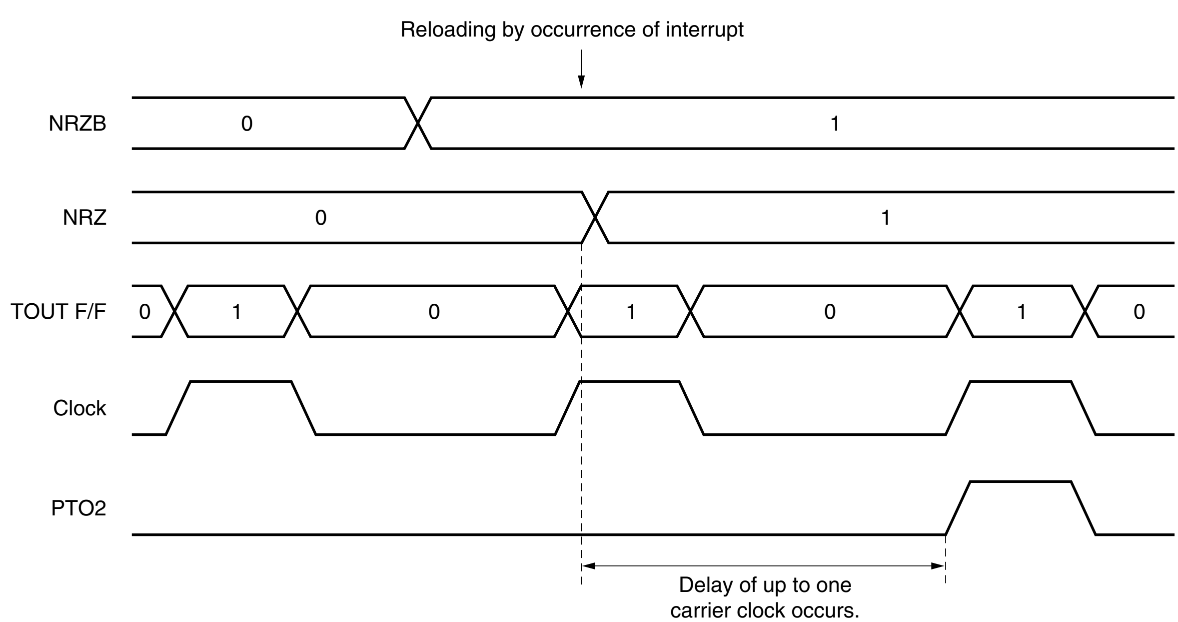
To output a carrier to the PTO2 pin, the time required for the initial carrier to be generated deviates by up to one carrier clock after reloading (the contents of the no return zero buffer flag (NRZB) are transferred to the no return zero flag (NRZ) by occurrence of the interrupt of timer counter channel 1, and the contents of NRZ are updated to “1”).

This is because reloading is performed asynchronously to the carrier clock, as illustrated below in order to hold constant the high-level period of the carrier.

<If delay after reloading is minimum>



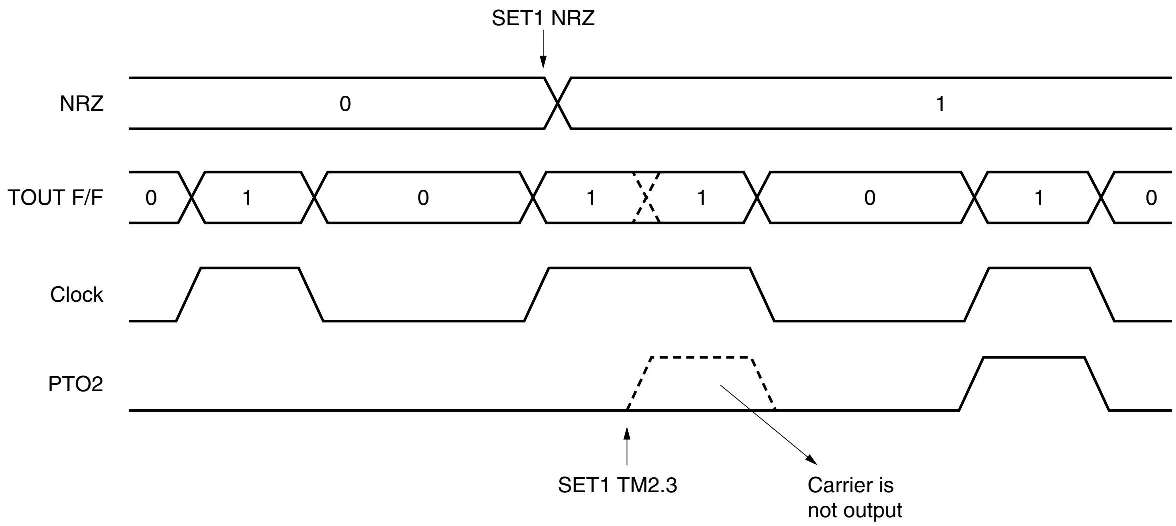
<If delay after reloading is maximum>



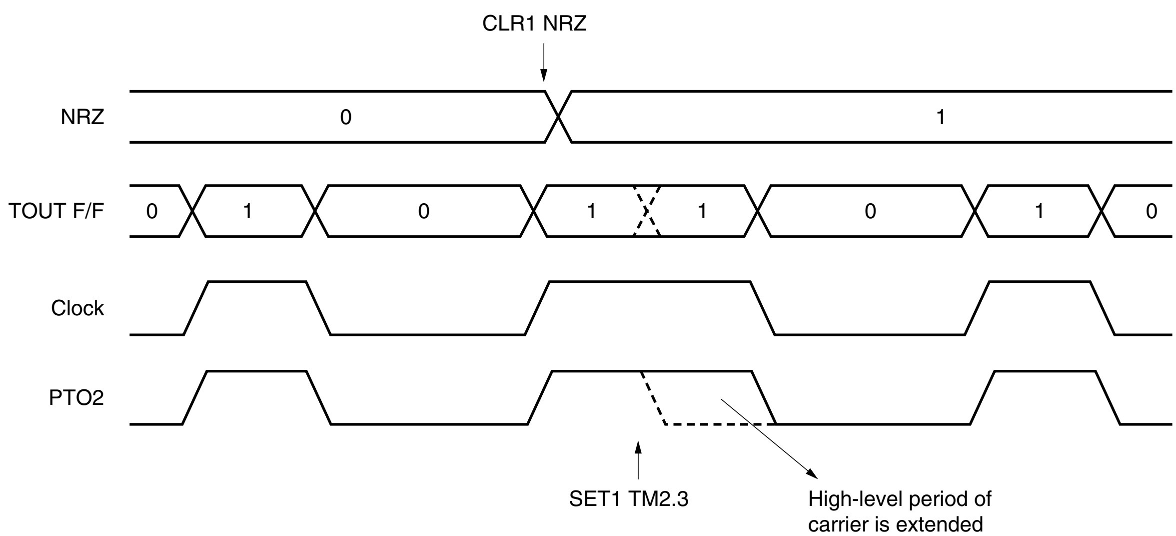
Download from Www.Somanuals.com. All Manuals Search And Download.

(7) Notes on application of carrier generator (restarting)

If forced reloading is performed by directly rewriting the contents of the no return zero flag (NRZ) and then the timer is restarted (by setting bit 3 of TM2 to "1") when the carrier clock is high (TOUT F/F holds "1"), the carrier may not be output to the PTO2 pin as shown below.



Likewise, if forced reloading is performed by directly rewriting the contents of NRZ and the timer is restarted (by setting bit 3 of TM2 to "1") when the carrier clock is high (TOUT F/F holds "1"), the high-level period of the carrier output to the PTO2 pin may be extended as shown below.



Download from Www.Somanuals.com. All Manuals Search And Download.

6.5 Programmable Threshold Port (Analog Input Port)

The μPD754244 provides analog input pins (PTH00, PTH01) whose threshold voltage (reference voltage) is selectable within sixteen steps. The following operations can be performed with these analog input pins.

(1) Comparator operation
(2) 4-bit resolution A/D converter operation (controlled by software)

Caution When using a programmable threshold port, do not specify connection of an internal pull-up resistor to port 6.

6.5.1 Configuration and operation of programmable threshold port

The configuration of the programmable threshold port is shown in Figure 6-49.

The input voltage via the PTH00 and PTH01 pins is compared with the threshold voltage (V_{REF}) specified by the programmable threshold port mode (PTHM) register, and the results are stored in the input latch of the programmable threshold port.

When V_{REF} > port input voltage: 0
When V_{REF} < port input voltage: 1

The conversion terminates when the conversion time specified by bit 6 of PTHM has elapsed after setting V_{REF} by the lower four bits of PTHM, and the conversion result is stored in the input latch of the programmable threshold port.

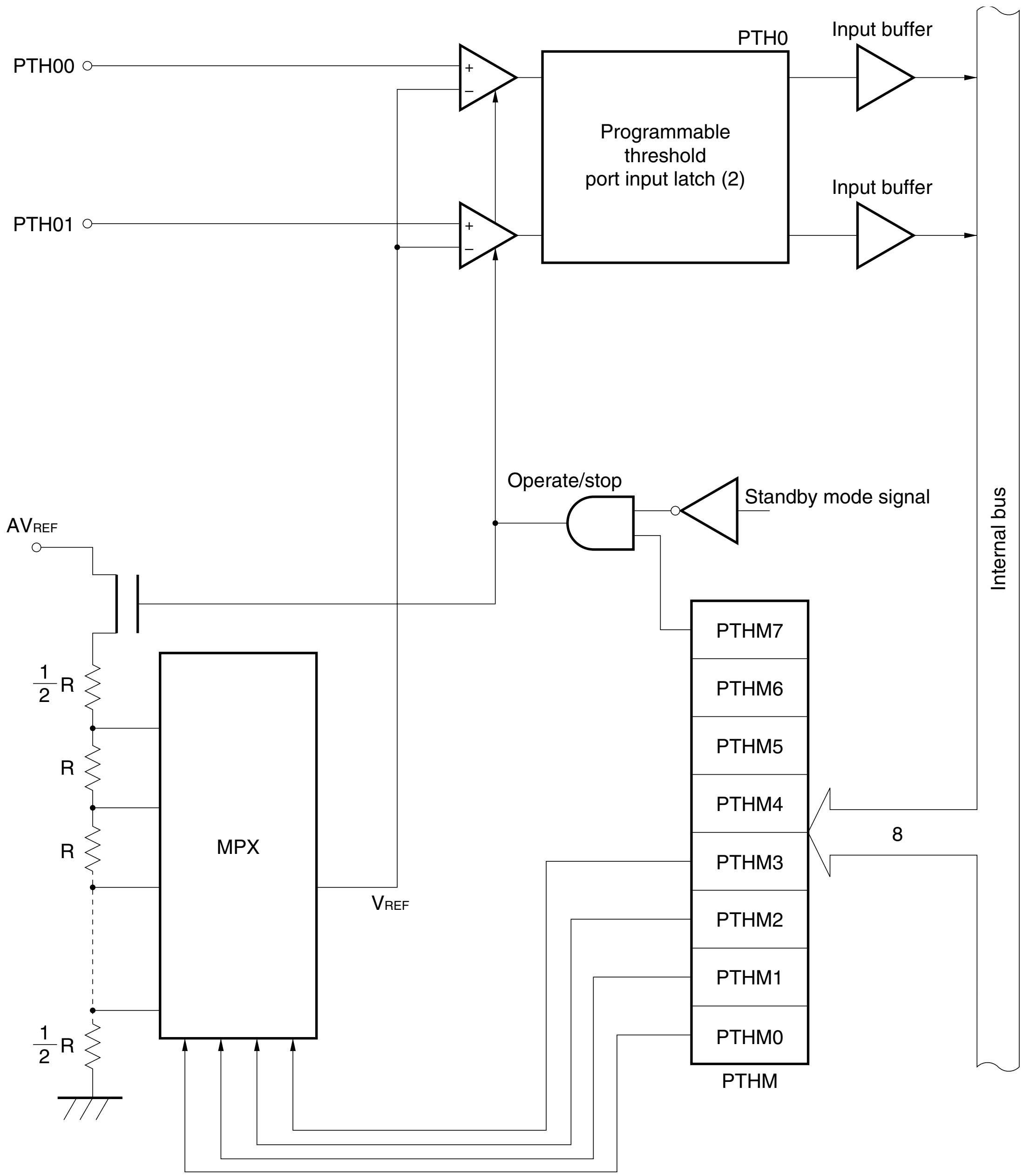
As a result, the conversion result must be read after the conversion time specified by bit 6 of PTHM has elapsed after modifying V_{REF} by PTHM modification.

When PCC = 0000B (low-speed operation mode), be sure to clear bit 6 of PTHM to "0" to select a long conversion time. When PCC = 0010B or 0011B, bit 6 of PTHM can be set to "1" to select the high-speed conversion.

The contents of the input latch can be read or tested by using a memory manipulation instruction, and the contents of the input latch become undefined by $\overline{\text{RESET}}$ signal generation.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 6-49. Block Diagram of Programmable Threshold Port



Download from Www.Somanuals.com. All Manuals Search And Download.

6.5.2 Programmable threshold port mode (PTHM) register

PTHM is an 8-bit register that controls the programmable threshold port operation, and it is set by an 8-bit memory manipulation instruction.

The threshold voltage can be selected by specifying the lower four bits of PTHM within 16 steps as follows.

$$\left(AV_{REF} \times \frac{0.5}{16} - AV_{REF} \times \frac{15.5}{16} \right)$$

It is also possible to reduce the current consumption if the comparator operation is stopped (PTHM7 = 0). All bits of PTHM are initialized to “0” by $\overline{\text{RESET}}$ signal generation, and the operation stop mode is entered.

Cautions 1. Bit 4 and 5 of PTHM must be set to “0”.

2. If a STOP or HALT instruction is executed during comparator operation (PTHM7 = 1), the operation stop. Therefore, a PTH0 read manipulation must be executed after the standby mode is released, input voltage is applied to the PTH00 and PTH01, and the conversion time has elapsed.

3. The conversion result of the programmable threshold port becomes undefined when the operation is stopped. Read the conversion result during comparator operation.

Figure 6-50. Format of Programmable Threshold Port Mode (PTHM) Register

Address	7	6	5	4	3	2	1	0	Symbol
FD6H	PTHM7	PTHM6	0	0	PTHM3	PTHM2	PTHM1	PTHM0	PTHM

Comparator operation mode specification

PTHM7	Operation mode
0	Comparator operation stopped
1	Comparator operating

Conversion time (maximum) selection[Note]

PTHM6	Conversion time
0	17 × 32/f_X (130 μs)
1	17 × 8/f_X (32.4 μs)

Values in parentheses are applicable when f_X = 4.19 MHz in the μPD754244.

Threshold voltage selection

Values set to PTHM3-PTHM0	Threshold voltage (V_{REF})
n (0 ≤ n ≤ 15)	$AV_{REF} \times \frac{(n + 0.5)}{16}$

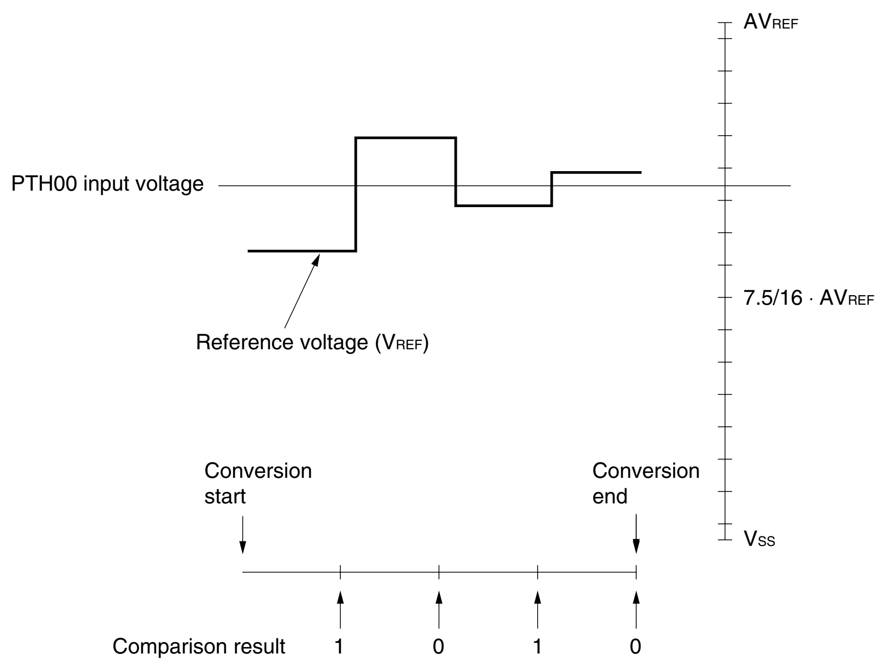
Note “PTHM6 = 1” can only be set when PCC = 0010B or 0011B.

Download from Www.Somanuals.com. All Manuals Search And Download.

6.5.3 Programmable threshold port application

(1) An analog input voltage input to the PTH00 pin is A/D converted with 4-bit resolution.

Figure 6-51. Application Example of Programmable Threshold Port



<Program example> The conversion result is stored in bit sequential buffer BSB0 (refer to **6.6 Bit Sequential Buffer**).

```
ADCONV: SET    MBE
        SEL    MB15
        MOV    HL, #0D3H         ; H ← higher 4 bits of PTH0, L ← bit 3 specification
        MOV    XA, #0C0H
        MOV    BSB0, A           ; BSB0 ← 0
LOOP:   SET1   BSB0, @L
        MOV    A, BSB0
        MOV    PTHM, XA          ; Comparison start
        MOV    A, #04H           ; 36-machine-cycle wait
WAIT:   INCS   A
        BR     WAIT
        MOV1   CY, @H+PTH0.0     ; Conversion result input
        MOV1   BSB0, @L, CY      ; Conversion result store
        DECS   L
        BR     LOOP
                                 ; Conversion end
```

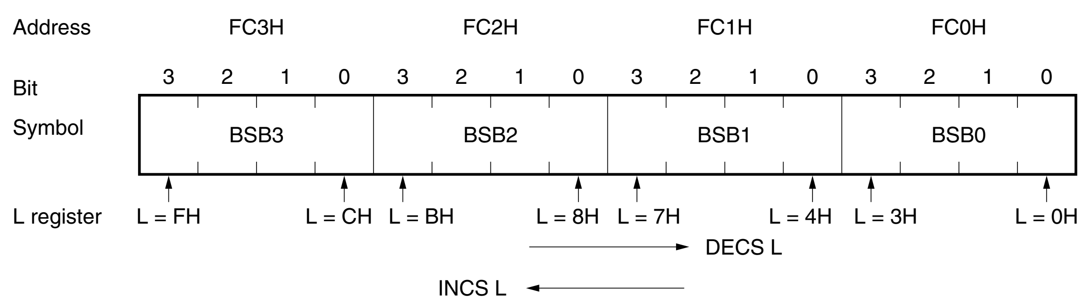

Download from Www.Somanuals.com. All Manuals Search And Download.

6.6 Bit Sequential Buffer ... 16 Bits

The bit sequential buffer (BSB) is a special data memory used for bit manipulation. It can manipulate bits by sequentially changing the address and bit specification. Therefore, this buffer is useful for processing data with a long bit length in bit units.

This data memory is configured of 16 bits and can be addressed by a bit manipulation instruction in the pmem.@L addressing mode. Its bits can be indirectly specified by the L register. The processing can be executed by only incrementing or decrementing the L register in a program loop and by moving the specified bit sequentially.

Figure 6-52. Format of Bit Sequential Buffer



Remarks 1. The specified bit is moved according to the L register in the pmem.@L addressing mode.

2. BSB can be manipulated at any time in the pmem.@L addressing mode, regardless of the specification of MBE and MBS.

The data in this buffer can also be manipulated even in direct addressing mode. By using 1-, 4-, or 8-bit direct addressing mode and pmem.@L addressing mode in combination, 1-bit data can be successively input or output. To manipulate BSB in 8-bit units, the higher and lower 8 bits are manipulated by specifying BSB0 and BSB2.

Download from Www.Somanuals.com. All Manuals Search And Download.

Example For serial output of the 16-bit data of BUFF1, 2 from 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 specified bit of BSB
        BR      LOOP1
        NOP                   ; Dummy (to adjust timing)
        SET1    PORT3.0       ; Sets bit 0 of port 3
        BR      LOOP2
LOOP1:  CLR1    PORT3.0       ; Clears bit 0 of port 3
        NOP                   ; Dummy (to adjust timing)
        NOP
LOOP2:  INCS    L             ; L ← L + 1
        BR      LOOP0
        RET
```


Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 7 INTERRUPT AND TEST FUNCTIONS

The μPD754244 has six vectored interrupt sources and one test input that can be used for various applications. The interrupt controller of the μPD754244 has unique features and can service interrupts at extremely high speed.

(1) Interrupt function

(a) Hardware-controlled vectored interrupt functions that can control acknowledgment of an interrupt by using an interrupt enable flag (IE×××) and interrupt master enable flag (IME)
(b) Any interrupt start address can be set.
(c) Interrupt nesting function that can specify priority by using an interrupt priority select register (IPS)
(d) Test function of interrupt request flag (IRQ×××) (Occurrence of an interrupt can be checked by software.)
(e) Releases standby mode (The interrupt that is used to release the standby mode can be selected by the interrupt enable flag.)

(2) Test function

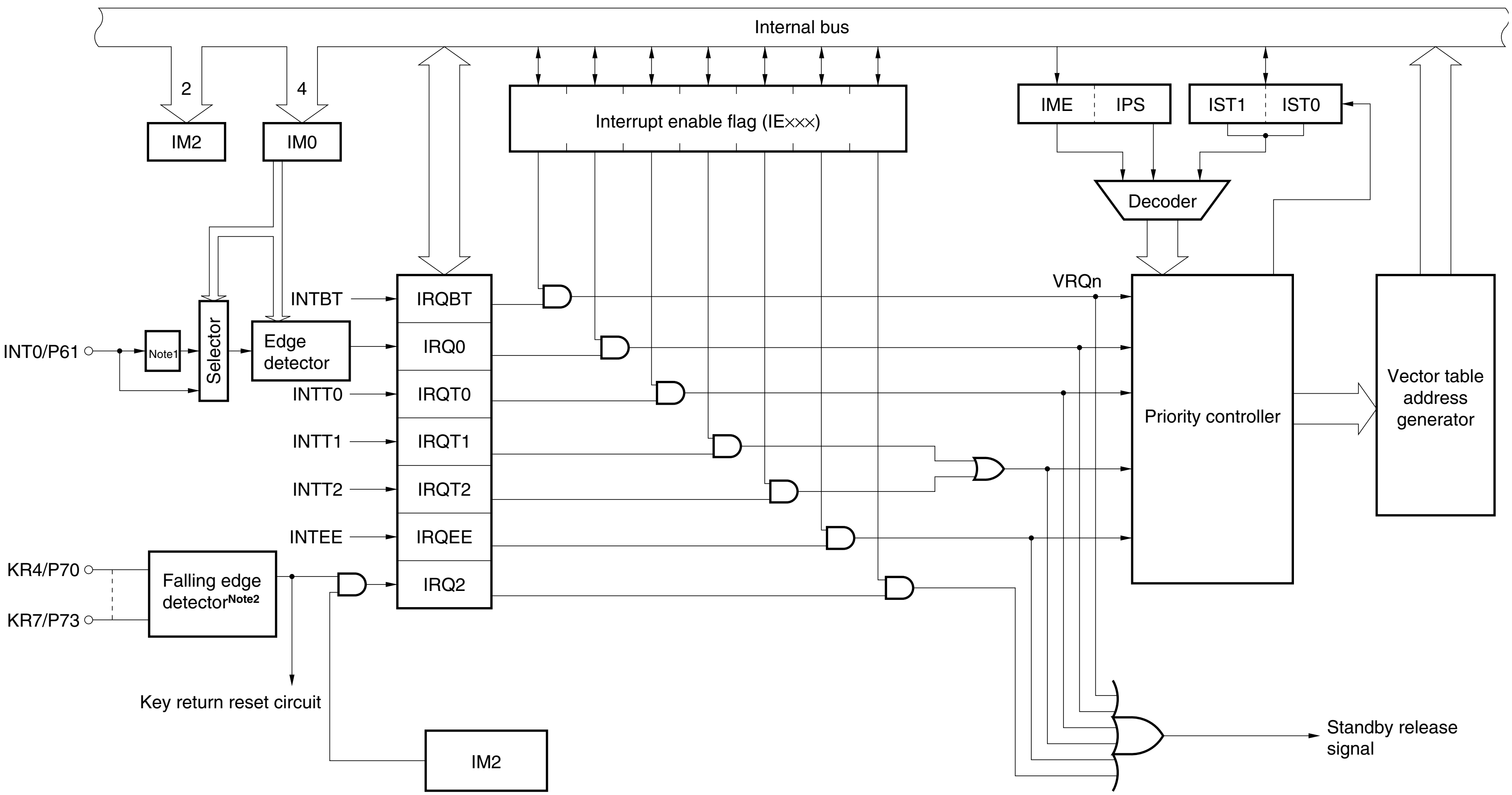
(a) Checks setting of a test request flag (IRQ2) via software
(b) Releases standby mode (The test source that releases the standby mode can be selected by the test enable flag.)

7.1 Configuration of Interrupt Controller

The interrupt controller is configured as shown in Figure 7-1, and each hardware unit is mapped to the data memory space.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-1. Block Diagram of Interrupt Controller



Notes 1. Noise eliminator (Standby release is disable when noise eliminator is selected.)

2. Does not have the INT2 pin. The interrupt request flag (IRQ2) is set at the KRn pin falling edge when IM20 = 1 and IM21 = 0.

Download from Www.Somanuals.com. All Manuals Search And Download.

7.2 Types of Interrupt Sources and Vector Table

The μPD754244 has the following six interrupt sources and nesting of interrupts can be controlled by software.

Table 7-1. Types of Interrupt Sources

Interrupt Source		Internal/External	Interrupt Priority[Note]	Vectored Interrupt Request Signal (Vector Table Address)
INBT	(reference time interval signal from basic interval timer/watchdog timer)	Internal	1	VRQ1 (0002H)
INT0	(rising edge or falling edge is selected)	External	2	VRQ2 (0004H)
INTT0	(signal indicating match between count register of timer counter 0 and modulo register)	Internal	3	VRQ5 (000AH)
INTT1	(signal indicating match between count register of timer counter 1 and modulo register)	Internal	4	VRQ6 (000CH)
INTT2	(signal indicating match between count register of timer counter 2 and modulo register)	Internal		
INTEE	(signal indicating writing of EEPROM has ended)	Internal	5	VRQ7 (000EH)

Note If two or more interrupts occur at the same time, the interrupts are processed according to this priority.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-2. Interrupt Vector Table

Address					
0002H	MBE	RBE	0	0	INTBT start address (higher 4 bits)
					INTBT start address (lower 8 bits)
0004H	MBE	RBE	0	0	INT0 start address (higher 4 bits)
					INT0 start address (lower 8 bits)
0006H					
0008H					
000AH	MBE	RBE	0	0	INTT0 start address (higher 4 bits)
					INTT0 start address (lower 8 bits)
000CH	MBE	RBE	0	0	INTT1, INTT2 start address (higher 4 bits)
					INTT1, INTT2 start address (lower 8 bits)
000EH	MBE	RBE	0	0	INTEE start address (higher 4 bits)
					INTEE start address (lower 8 bits)

The priority column in Table 7-1 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 held pending.

Write the start address of interrupt servicing to the vector table , and the set values of MBE and RBE during interrupt servicing. The vector table is set by using an assembler quasi directive (VENTn: n = 1, 2, or 5 to 7).

Example Setting of vector table of INTBT

```
VENT1   MBE=0,  RBE=0,  GOTOBT
  ↑       ↑       ↑       ↑
 <1>     <2>     <3>     <4>
```

<1> Vector table of address 0002
<2> Setting of MBE in interrupt servicing routine
<3> Setting of RBE in interrupt servicing routine
<4> Symbol indicating start address of interrupt servicing routine

Caution The contents described as the operand of VENTn (n = 1, 2, or 5 to 7) (MBE, RBE, or start address) are stored in the vector table address at address 2n.

Example Setting of vector tables of INTBT and INTT0

```
VENT1   MBE=0, RBE=0, GOTOBT; INTBT start address
VENT5   MBE=0, RBE=1, GOTOT0; INTT0 start address
```


Download from Www.Somanuals.com. All Manuals Search And Download.

7.3 Hardware Controlling Interrupt Function

(1) Interrupt request flag and interrupt enable flag

The μPD754244 has the following six interrupt request flags (IRQ×××) corresponding to the respective interrupt sources.

INT0 interrupt request flag (IRQ0)
BT interrupt request flag (IRQBT)
EEPROM interrupt request flag (IRQEE)
Timer counter 0 interrupt request flag (IRQT0)
Timer counter 1 interrupt request flag (IRQT1)
Timer counter 2 interrupt request flag (IRQT2)

Each interrupt request flag is set to “1” when the corresponding interrupt request is generated, and is automatically cleared to “0” when the interrupt servicing is executed. However, because IRQT1 and IRQT2 share the vector address, these flags are cleared differently from the other flags (refer to **7.6 Servicing of Interrupts Sharing Vector Address**).
The μPD754144 also has six interrupt enable flags (IE×××) corresponding to the respective interrupt request flags.

INT0 interrupt enable flag (IE0)
BT interrupt enable flag (IEBT)
EEPROM interrupt enable flag (IEEE)
Timer counter 0 interrupt enable flag (IET0)
Timer counter 1 interrupt enable flag (IET1)
Timer counter 2 interrupt enable flag (IET2)

The interrupt enable flag enables the corresponding interrupt when it is “1”, and disables the interrupt when it is “0”.
If an interrupt request flag is set and the corresponding interrupt enable flag enables the interrupt, a vector interrupt (VRQn: n = 1, 2, or 5 to 7) occurs. This signal is also used to release the standby mode.
The interrupt request flags and interrupt enable flags are manipulated by a bit manipulation or 4-bit manipulation instruction. When a bit manipulation instruction is used, the flags can be directly manipulated, regardless of the setting of MBE. The interrupt enable flags are manipulated by the EI IE××× and DI IE××× instructions. To test an interrupt request flag, the SKTCLR instruction is usually used.

Example

```
EI        IE0       ; Enables INT0
DI        IET1      ; Disables INTT1
SKTCLR    IRQBT     ; Skips and clears if IRQBT is 1
```

When an interrupt request flag is set by an instruction, a vector interrupt is executed even if an interrupt does not occur, in the same manner as when the interrupt occurs.
The interrupt request flags and interrupt enable flags are cleared to “0” when the $\overline{\text{RESET}}$ signal is asserted, disabling all the interrupts.

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 7-2. Signals Setting Interrupt Request Flags

Interrupt Request Flag	Signal Setting Interrupt Request Flag	Interrupt Enable Flag
IRQBT	Set by reference time interval signal from basic interval timer watchdog timer	IEBT
IRQ0	Set by detection of edge of INT0/P61 pin input signal. Edge to be detected is selected by INT0 edge detection mode register (IM0)	IE0
IRQT0	Set by match signal from timer counter 0	IET0
IRQT1	Set by match signal from timer counter 1	IET1
IRQT2	Set by match signal from timer counter 2	IET2
IRQEE	Set by EEPROM write end signal	IEEE

(2) Interrupt priority select register (IPS)

The interrupt priority select register selects an interrupt with a higher priority that can be nested. The lower 3 bits of this register are used for this purpose.

Bit 3 is an interrupt master enable flag (IME) that enables or disables all the interrupts.

IPS is set by a 4-bit memory manipulation instruction, but bit 3 is set or reset by the EI or 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, #1001
MOV     IPS, A      ; Gives higher priority to INTBT and enables interrupt
```

When the $\overline{\text{RESET}}$ signal is asserted, all the bits of this register are cleared to “0”.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-3. Interrupt Priority Select Register

Address	3	2	1	0	Symbol
FB2H	IPS3	IPS2	IPS1	IPS0	IPS

Selection of higher-priority interrupts

IPS2	IPS1	IPS0		
0	0	0	No interrupts are handled as higher-priority interrupts.	
0	0	1	VRQ1 (INTBT)	Vectored interrupt on left is selected as higher priority.
0	1	0	VRQ2 (INT0)	
0	1	1	Setting prohibited Note	
1	0	0		
1	0	1	VRQ5 (INTT0)	Vectored interrupt on left is selected as higher priority.
1	1	0	VRQ6 (INTT1, INTT2)	
1	1	1	VRQ7 (INTEE)	

Interrupt master enable flag (IME)

IPS3	
0	Disables all the interrupts and no vectored interrupt is started.
1	Controls interrupt enable/disable by the corresponding interrupt enable flag.

Note If this value is set in the IPS register then the state is the same as if it had been set to IPS = X000B (Does not give high priority to any interrupt.)

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Hardware of INT0

(a) Figure 7-4 shows the configuration of INT0, which is an external interrupt input that can be detected at the rising or falling edge depending on the specification.

INT0 also has a noise elimination function which uses a sampling clock (refer to **Figure 7-5 I/O Timing of Noise Eliminator**). The noise eliminator eliminates a pulse having a width narrower than 2 cycles[Note] of the sampling clock as noise. However, a pulse having a width wider than one cycle of the sampling clock may be acknowledged as the interrupt signal depending on the timing of sampling (refer to **Figure 7-5 <2> (a)**). A pulse having a width wider than two cycles of the sampling clock is always acknowledged as the interrupt without fail.

INT0 has two sampling clocks for selection: Φ and $f_X/64$. These sampling clocks are selected by using bit 3 (IM03) of the INT0 edge detection mode register (IM0) (refer to **Figure 7-6**).

The edge of INT0 to be detected is selected by using bits 0 and 1 of IM0.

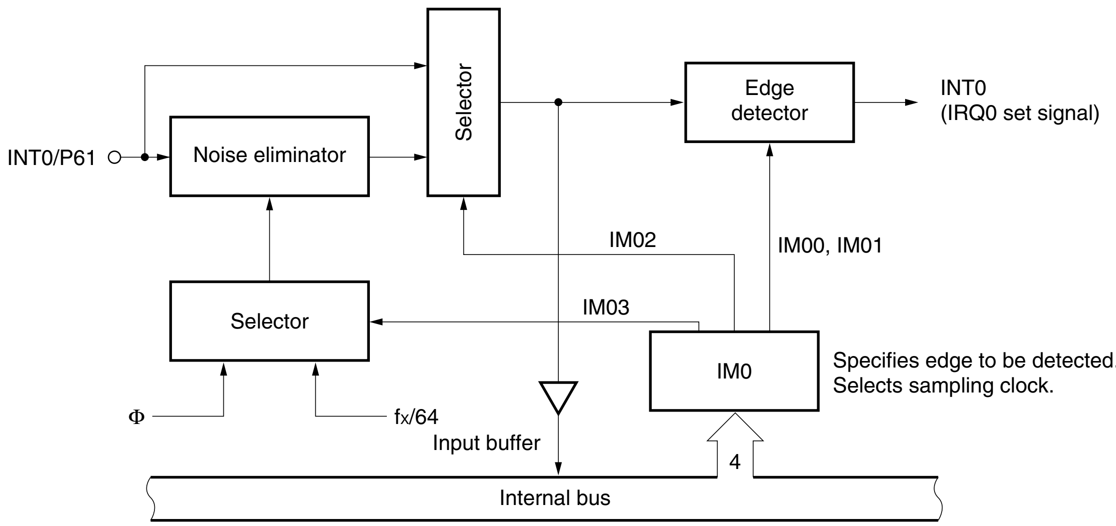
Figure 7-6 shows the format of IM0. This register is manipulated by a 4-bit manipulation instruction. All the bits of this register are cleared to “0” when the $\overline{\text{RESET}}$ signal is asserted, and the rising edge of INT0 is specified to be detected.

Note When sampling clock is Φ: $2t_{CY}$
When sampling clock is $f_X/64$: $128/f_X$

Cautions
1. **Even when a signal is input to the INT0/P61 pin in the port mode, it is input through the noise eliminator. Therefore, input a signal having a width wider than two cycles of the sampling clock.**
2. **When the noise eliminator is selected (by clearing IM02 to 0), INT0 does not operate in the standby mode because it performs sampling by using the clock. (The noise eliminator does not operate unless the CPU clock Φ is supplied to it.) Therefore, do not select the noise eliminator if it is necessary to release the standby mode by INT0 (set IM02 to 1).**

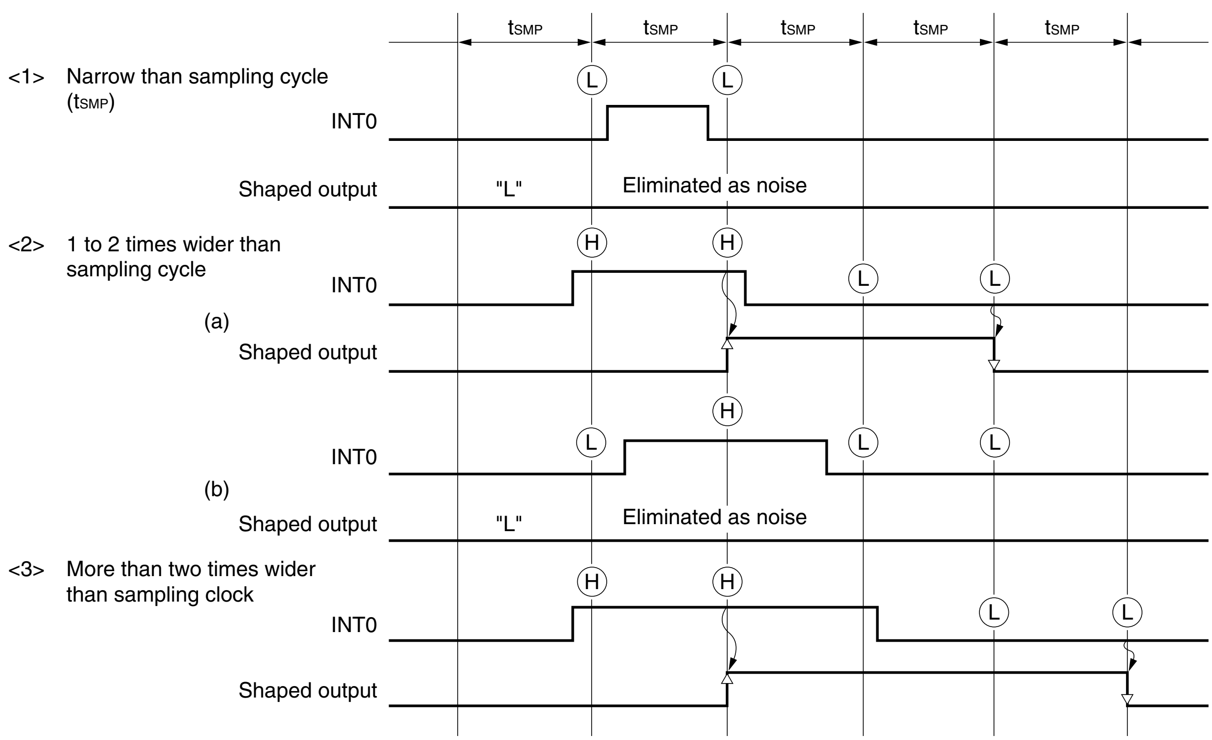
Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-4. Configuration of INT0



Note Even if fx/64 is selected, the HALT mode cannot be released by INT0.

Figure 7-5. I/O Timing of Noise Eliminator



Remark $t_{SMP} = t_{CY}$ or $64/f_X$

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-6. Format of INT0 Edge Detection Mode Register (IM0)

Address	3	2	1	0	Symbol
FB4H	IM03	IM02	IM01	IM00	IM0

IM01	IM00	Specifies edge to be detected
0	0	Rising edge
0	1	Falling edge
1	0	Both rising and falling edges
1	1	Ignored (interrupt request flag is not set)

IM02	Noise eliminator select bit	Sampling	Standby release
0	Selects noise eliminator	Enabled	Disabled
1	Does not select noise eliminator	Disabled	Enabled

IM03	Sampling clock
0	Φ [Note]
1	$f_X/64$ [Note]

Note This value differs depending on the system clock frequency (f_X).

Caution When the contents of the edge detection mode register are changed, the interrupt request flag may be set. Therefore, you should disable interrupts before changing the contents of the mode register. Then, clear the interrupt request flag by using the CLR1 instruction to enable the interrupts. If the contents of IM0 are changed and the sampling clock of $f_X/64$ is selected, clear the interrupt request flag 16 machine cycles after the contents of the mode register have been changed.

Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Interrupt status flag

The interrupt status flags (IST0 and IST1) indicate the status of the processing currently being executed by the CPU and are included in PSW.

The interrupt priority controller controls nesting of interrupts according to the contents of these flags as shown in Table 7-3.

Because IST0 and IST1 can be changed by using a 4-bit or bit manipulation instruction, interrupts can be nested 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 interrupts. Execute the EI instruction after manipulating the flags to enable interrupts.

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 one higher. 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 7-3. IST1 and IST0 and Interrupt Servicing Status

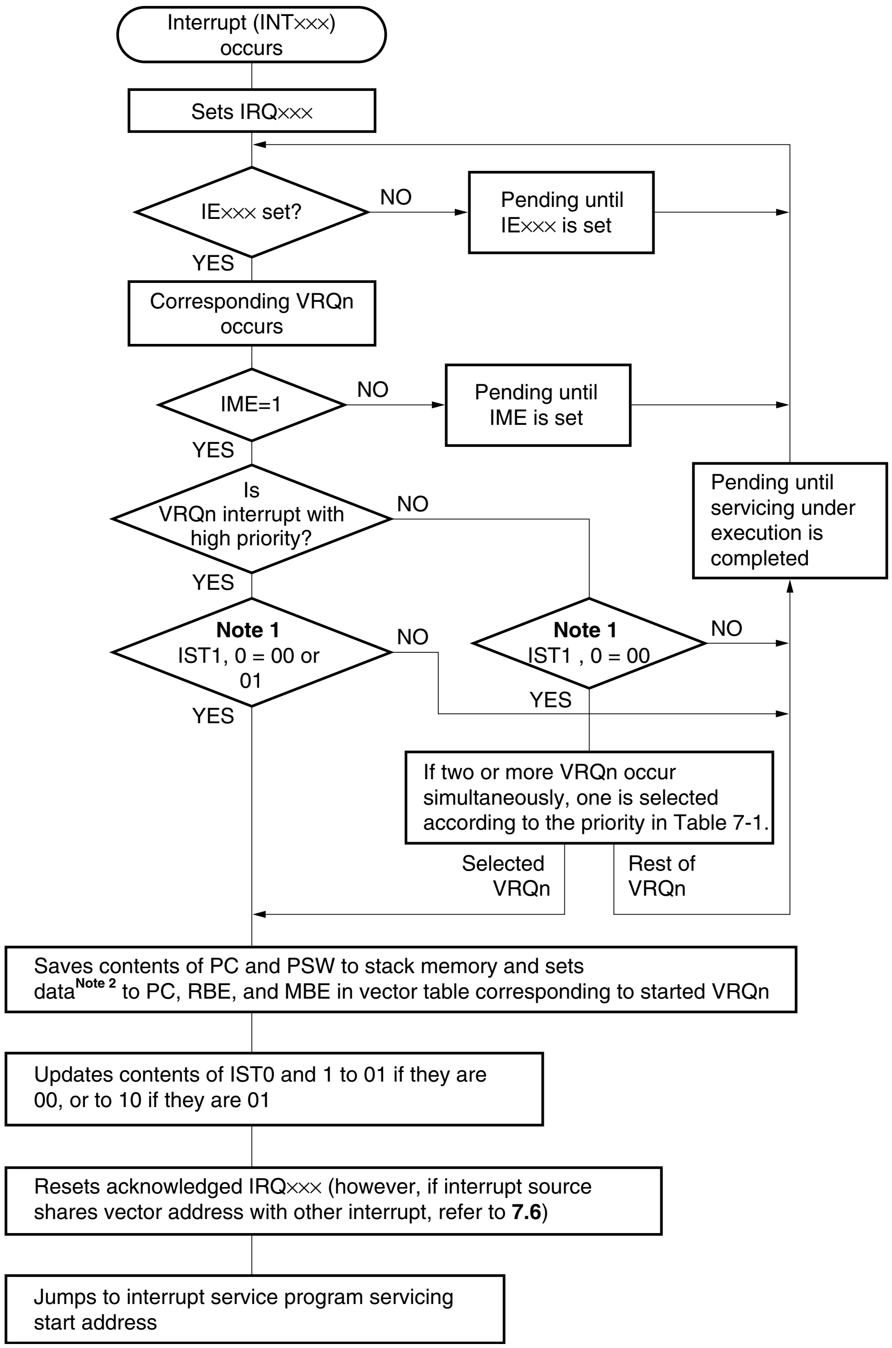
IST1	IST0	Status of Processing under Execution	Processing by CPU	Interrupt Request That Can Be Acknowledged	After Interrupt Acknowledged	
					IST1	IST0
0	0	Status 0	Executes normal program	All interrupts can be acknowledged	0	1
0	1	Status 1	Services interrupt with low or high priority	Interrupt with high priority can be acknowledged	1	0
1	0	Status 2	Services interrupt with high priority	Acknowledging all interrupts is disabled	–	–
1	1	Setting prohibited				

Download from Www.Somanuals.com. All Manuals Search And Download.

7.4 Interrupt Sequence

When an interrupt occurs, it is processed according to the procedure illustrated below.

Figure 7-7. Interrupt Servicing Sequence



Notes **1.** IST1 and 0: Interrupt status flags (bits 3 and 2 of PSW; Refer to **Table 7-3**.)

2. Each vector table stores the start address of an interrupt service program and the preset values of MBE and RBE when the interrupt is started.

Download from Www.Somanuals.com. All Manuals Search And Download.

7.5 Nesting Control of Interrupts

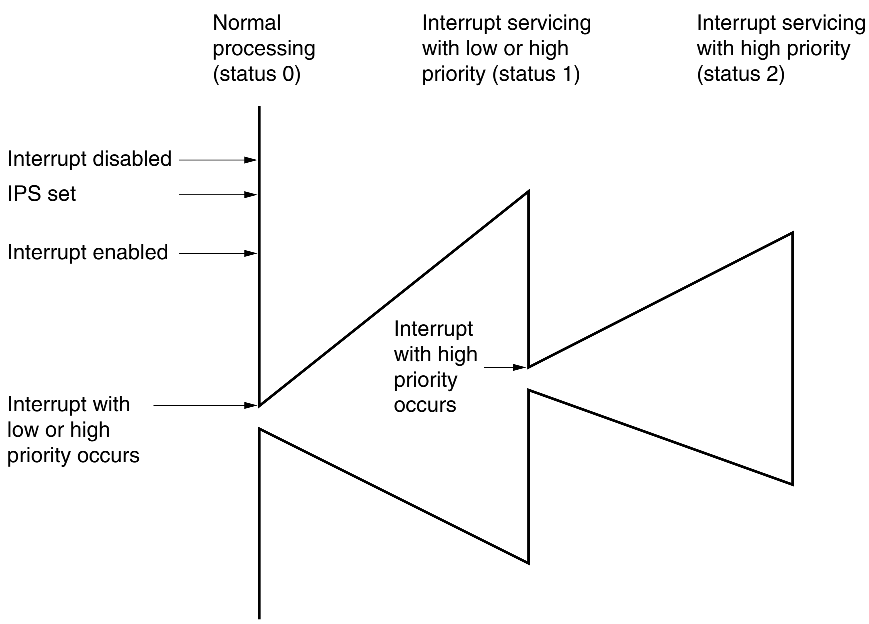
The μPD754244 can nest interrupts by the following two methods.

(1) Nesting with interrupt having high priority specified

This method is the standard nesting method of the μPD754244. One interrupt source is selected and nested. An interrupt with a higher priority specified by the interrupt priority select register (IPS) can occur when the status of the processing under execution is 0 or 1, and the other interrupts (interrupts with a lower priority) can occur when the status is 0 (refer to **Figure 7-8** and **Table 7-3**).

Therefore, if you use this method when you wish to nest only one interrupt, operations such as enabling and disabling interrupts, that is, changing the interrupt status flag, while the interrupt is serviced do not need to be performed, and the nesting level can be kept to 2.

Figure 7-8. Nesting of Interrupt with High Priority



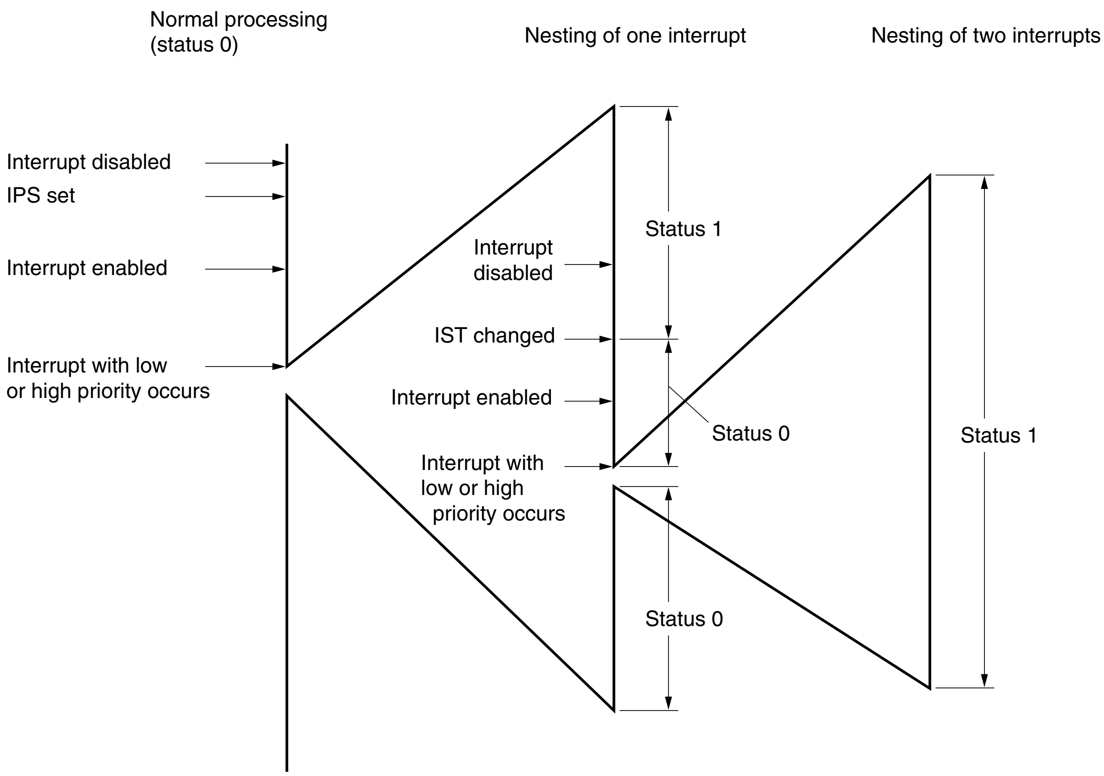
Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Nesting by changing interrupt status flags

Nesting can be implemented if the interrupt status flags are changed by program. In other words, nesting is enabled when IST1 and IST0 are cleared to “0, 0” by an interrupt servicing program, and status 0 is set. This method is used to nest two or more interrupts, or to implement nesting level 3 or higher.

Before changing IST1 and IST0, disable interrupts by using the DI instruction.

Figure 7-9. Interrupt Nesting by Changing Interrupt Status Flag



Download from Www.Somanuals.com. All Manuals Search And Download.

7.6 Servicing of Interrupts Sharing Vector Address

Because interrupt sources INTT1 and INTT2 share vector tables, you should select one or both of the interrupt sources in the following way.

(1) To use one interrupt

Of the two interrupt sources sharing a vector table, set the interrupt enable flag of the necessary interrupt source to “1”, and clear the interrupt enable flag of the other interrupt source to “0”. In this case, an interrupt request is generated by the interrupt source that is enabled (IE××× = 1). When the interrupt is acknowledged, the interrupt request flag is reset.

(2) To use both interrupts

Set the interrupt enable flags of both the interrupt sources to “1”. In this case, the interrupt request flags of the two interrupt sources are ORed.

In this case, if an interrupt request is acknowledged when one or both the interrupt flags are set, the interrupt request flags of both the interrupt sources are not reset. This is in order to ascertain which interrupt was generated during interrupt servicing.

Therefore, it is necessary to identify which interrupt source has generated the interrupt by using an interrupt service routine. This can be done by checking the interrupt request flags by executing the SKTCLR instruction at the beginning of the interrupt service routine.

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 servicing is started by the remaining interrupt request.

Consequently, the interrupt request not tested is serviced first. If the selected interrupt has a lower priority, the remaining interrupt is held pending and therefore, the interrupt request tested is serviced first. Therefore, an interrupt sharing a vector address with the other interrupt is identified differently, depending whether it has a higher priority, as shown in Table 7-4.

Table 7-4. Identifying Interrupt Sharing Vector 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

Download from Www.Somanuals.com. All Manuals Search And Download.

Examples **1.** To use both INTT1 and INTT2 as having higher priority, and give priority to INTT2

```
            DI
            SKTCLR      IRQT2      ;  IRQT2=1?
            BR          VSUBBT     ⎫
              :                    ⎪
                                   ⎬ Service routine of INTT2
            EI                     ⎪
            RETI                   ⎪
              :                    ⎭

VSUBBT:     CLR1        IRQT1      ⎫
              :                    ⎬ Service routine of INTT1
            EI                     ⎪
            RETI                   ⎭
```

2. To use both INTT1 and INTT2 as having lower priority, and give priority to INTT2

```
            SKTCLR      IRQT2      ;  IRQT2 =1?
            BR          VSUBBT     ⎫
              :                    ⎪
                                   ⎬ Service routine of INTT2
            RETI                   ⎪
              :                    ⎭

VSUBBT:     CLR1        IRQT1      ⎫
              :                    ⎬ Service routine of INTT1
            RETI                   ⎭
```

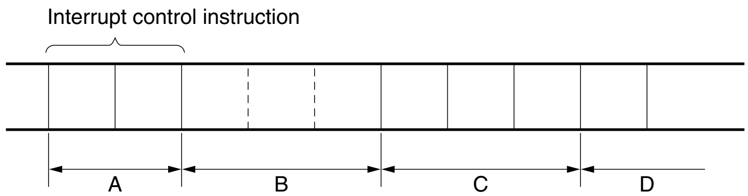

Download from Www.Somanuals.com. All Manuals Search And Download.

7.7 Machine Cycles Until Interrupt Servicing

The number of machine cycles required from when an interrupt request flag (IRQxxx) has been set until the interrupt routine is executed is as follows.

(1) If IRQxxx is set while interrupt control instruction is being executed

If IRQxxx is set while an interrupt control instruction is being executed, the next instruction is executed. Then three machine cycles of interrupt servicing is performed and the interrupt routine is executed.



A: Sets IRQxxx

B: Executes next instruction (1 to 3 machine cycles; differs depending on instruction)

C: Interrupt servicing (3 machine cycles)

D: Executes interrupt routine

Cautions 1. If two or more interrupt control instructions are successively executed, the instruction following the interrupt control instruction executed last is executed, three machine cycles of interrupt servicing is performed, and then the interrupt routine is executed.

2. If the DI instruction is executed when or after IRQxxx is set (A in the above figure), the interrupt request corresponding to IRQxxx that has been set is held pending until the EI instruction is executed next time.

Remarks 1. An interrupt control instruction manipulates the hardware units related to interrupts (address FB×H of the data memory). The EI and DI instructions are interrupt control instructions.

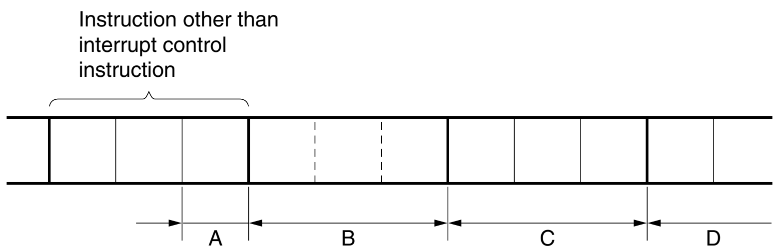
2. The three machine cycles of interrupt servicing is the time required to manipulate the stack that is manipulated when an interrupt is acknowledged.

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) If IRQxxx is set while instruction other than (1) is executed

(a) If IRQxxx 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 servicing is performed, and then the interrupt routine is executed.



A: Sets IRQxxx

B: Executes next instruction (1 to 3 machine cycles; differs depending on instruction)

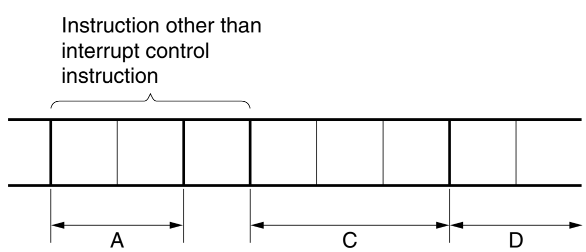
C: Interrupt servicing (3 machine cycles)

D: Executes interrupt routine

Caution If the next instruction is an interrupt control instruction, the instruction following the interrupt control instruction executed last is executed, three machine cycles of interrupt servicing is performed, and then the interrupt routine is executed. If the DI instruction is executed after IRQxxx has been set, the interrupt request corresponding to the set IRQxxx is held pending.

(b) If IRQxxx is set before the last machine cycle of the instruction under execution

In this case, three machine cycles of servicing is performed after execution of the current instruction, and then the interrupt routine is executed.



A: Sets IRQxxx

B: Interrupt servicing (3 machine cycles)

C: Executes interrupt routine

Download from Www.Somanuals.com. All Manuals Search And Download.

7.8 Effective Usage of Interrupts

Use the interrupt function effectively as follows.

(1) 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. For the interrupt service routine 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 BR instruction, and set RBS to 1 to select register bank 1.

(2) 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 increased by using an instruction to set the interrupt flag.

7.9 Application of Interrupt

To use the interrupt function, first set as follows using the main program.

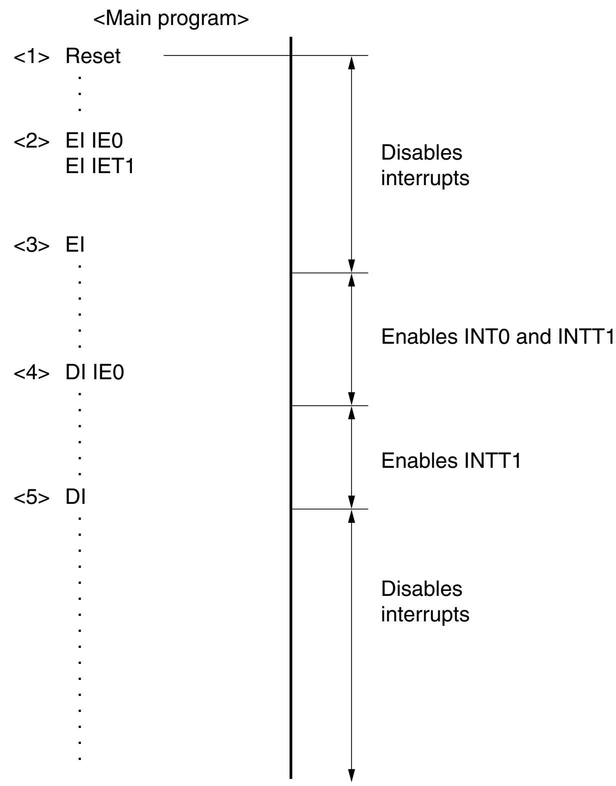
(a) Set the interrupt enable flag of the interrupt used (by using the EI IE××× instruction).
(b) To use INT0, select the active edge (set IM0).
(c) To use nesting (of an interrupt with a higher priority), set IPS (IME can be set at the same time).
(d) Set the interrupt master enable flag (by using the EI instruction).

In the interrupt service program, MBE and RBE are set by the vector table. However, when the interrupt specified as having a higher priority is serviced, the register bank must be saved and set.

To return from the interrupt service routine, use the RETI instruction.

Download from Www.Somanuals.com. All Manuals Search And Download.

(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.

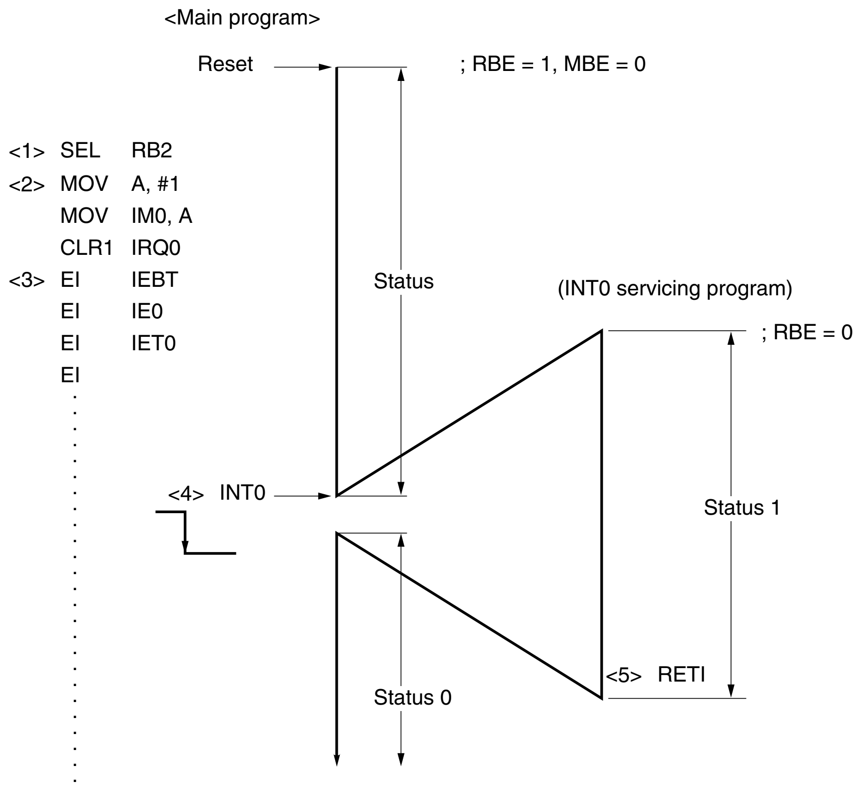
<3> The interrupt master enable flag is set by the EI instruction. INT0 and INTT1 are enabled at this time.

<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.

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Example of using INTBT and INT0 (falling edge active): not nested (all interrupts have higher 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> Interrupts are enabled by the EI, EI IE××× instruction.

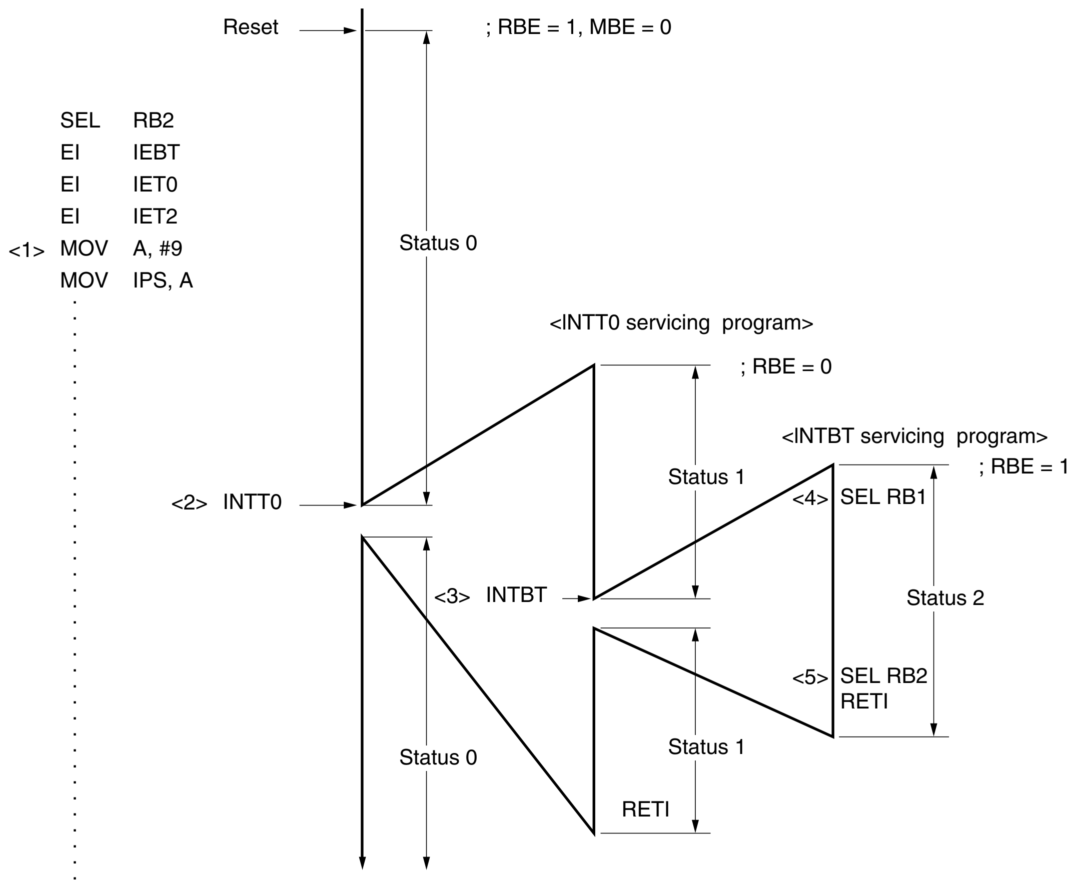
<4> The INT0 interrupt servicing program is started at the falling edge of INT0. The status is changed to 1, and all interrupts are disabled.
RBE = 0, and register banks 0 and 1 are used.

<5> Execution returns from the interrupt routine when the RETI instruction is executed. The status is returned to 0 and interrupts are enabled.

Remark If all the interrupts are used with 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 program and register banks 2 and 3 are used, and RBE = 0 for the interrupt routine and register banks 0 and 1 are used.

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Nesting of interrupts with higher priority (INTBT has higher priority and INTT0 and INTT2 have lower priority)



<1> INTBT is specified as having a higher priority by setting of IPS, and interrupts are enabled at the same time.

<2> INTT0 servicing program is started when INTT0 with a lower priority occurs. Status 1 is set and the other interrupts with a lower priority are disabled. RBE = 0 to select register bank 0.

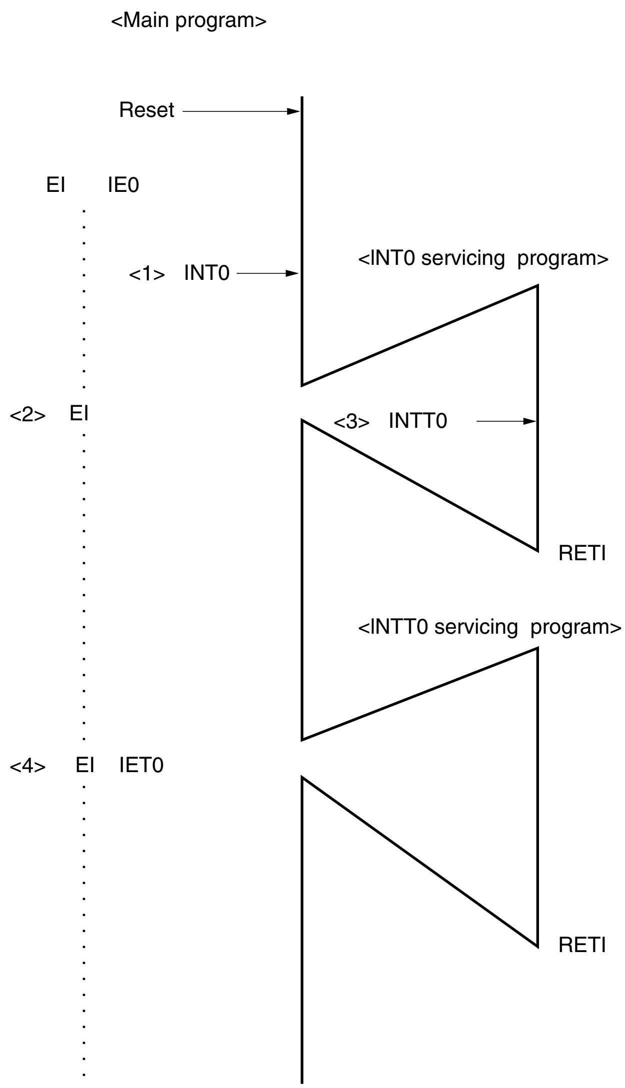
<3> INTBT with a higher priority occurs. The interrupts are nested. The status is changed to 0 and all interrupts are disabled.

<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.

Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Executing pending interrupt - interrupt input while interrupts are disabled -



<1> The request flag is held pending even if INT0 is set while the interrupts are disabled.

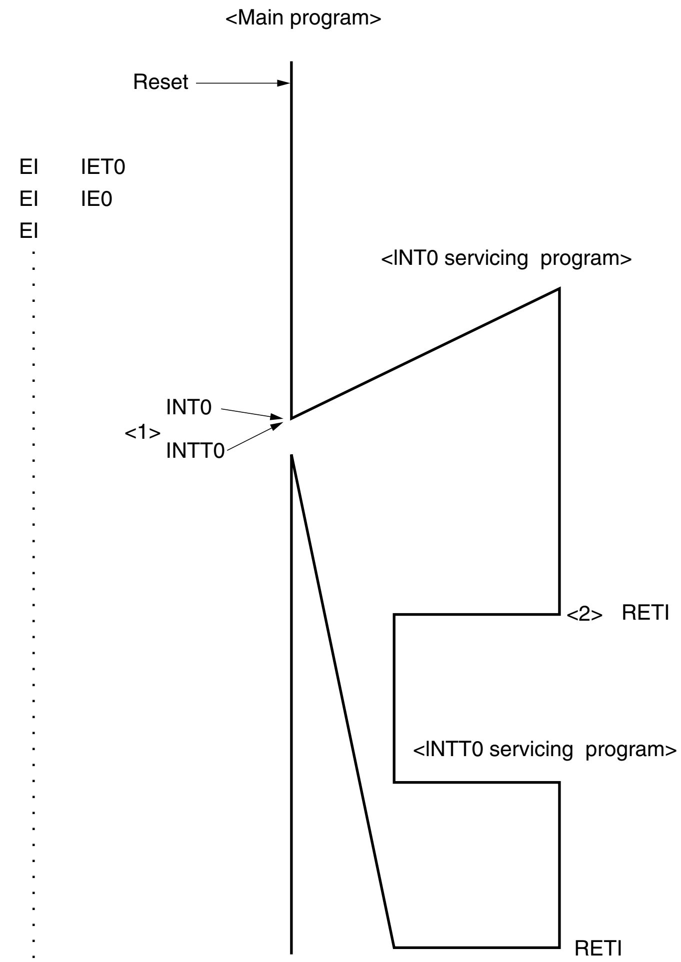
<2> INT0 servicing program is started when the interrupts are enabled by the EI instruction.

<3> Same as <1>.

<4> INTT0 servicing program is started when the pending INTT0 is enabled.

Download from Www.Somanuals.com. All Manuals Search And Download.

(5) Executing pending interrupt - two interrupts with lower priority occur simultaneously -

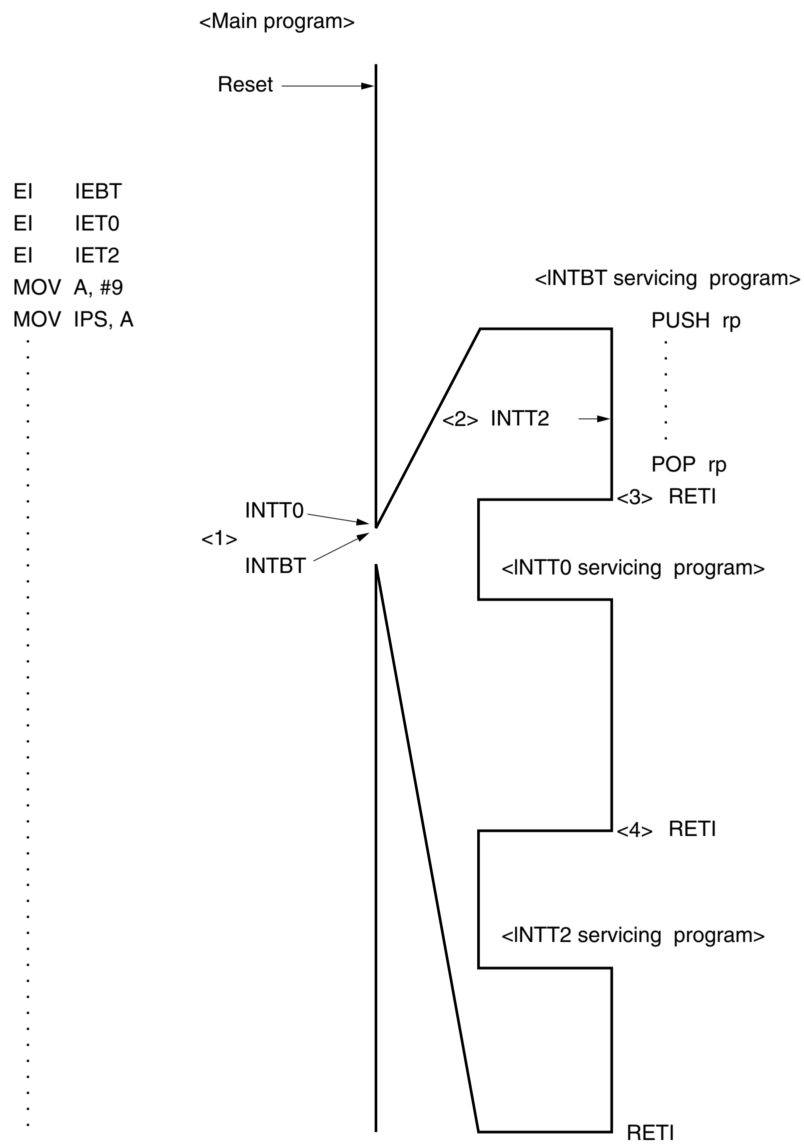


<1> If INT0 and INTT0 with a lower priority occur at the same time (while the same instruction is being executed), INT0 with a higher priority is executed first (INTT0 is held pending).

<2> When the INT0 servicing routine is terminated by the RETI instruction, the pending INTT0 servicing program is started.

Download from Www.Somanuals.com. All Manuals Search And Download.

★ **(6) Executing pending interrupt - interrupt occurs during interrupt service (INTBT has higher priority and INTT0 and INTT2 have lower priority) -**



<1> If INTBT with a higher priority and INTT0 with a lower priority occur at the same time, the servicing of the interrupt with the higher priority is started. (If there is no possibility that an interrupt with a higher priority will occur while another interrupt with a higher priority is being serviced, DI IE×× is not necessary.)

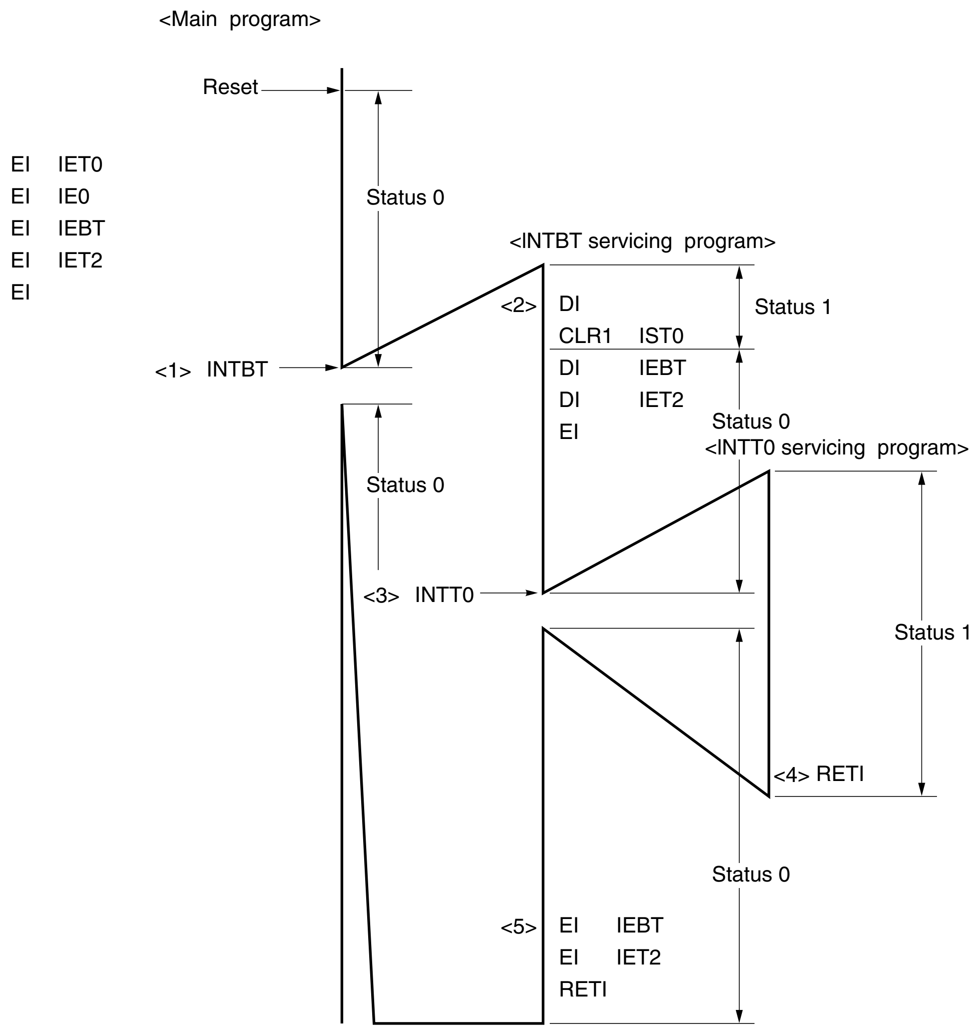
<2> If an interrupt with a lower priority occurs while an interrupt with a higher priority is being executed, the interrupt with the lower priority is held pending.

<3> When the interrupt with the higher priority has been serviced, INTT0 with the highest priority of the pending interrupts is executed.

<4> When the servicing of INTT0 has been completed, the pending INTT2 is serviced.

Download from Www.Somanuals.com. All Manuals Search And Download.

(7) Enabling nesting of two interrupts - INTT0 and INT0 are nested doubly and INTBT and INTT2 are nested singly -



<1> When an INTBT that does not enable nesting occurs, the INTBT servicing routine is started. The status is 1.

<2> The status is changed to 0 by clearing IST0. INTBT and INTT2 that do not enable nesting are disabled.

<3> When an INTT0 that enables nesting occurs, nesting is executed. The status is changed to 1, and all interrupts are disabled.

<4> The status is returned to 1 when INTT0 servicing is completed.

<5> The disabled INTBT and INTT2 are enabled, and execution returns to the main routine.

Download from Www.Somanuals.com. All Manuals Search And Download.

7.10 Test Function

7.10.1 Types of test sources

The μPD754244 has a test source, INT2. INT2 is an edge-detection testable input.

Table 7-5. Types of Test Sources

Test Source	Internal/External
INT2 (detects falling edge of input to KR4 to KR7 pins)	External

7.10.2 Hardware controlling test function

(1) Test request and test enable flags

The test request flag (IRQ2) is set to "1" when a test request is generated (INT2). Clear this flag to "0" by software after the test processing has been executed.

A test enable flag (IE2) is provided for the test request flag. When this flag is "1", the standby release signal is enabled; when it is "0", the signal is disabled.

If both the test request flag and test enable flag are set to "1", the standby release signal is generated.

Table 7-6 shows the signals that set the test request flags.

Table 7-6. Test Request Flag Setting Signals

Test Request Flag	Test Request Flag Setting Signal	Test Enable Flag
IRQ2	Detection of falling edge of any input to KR4/P70-KR7/P73 pins. Edge to be detected is selected by INT2 edge detection mode register (IM2)	IE2

(2) Hardware of key interrupts (KR4 to KR7)

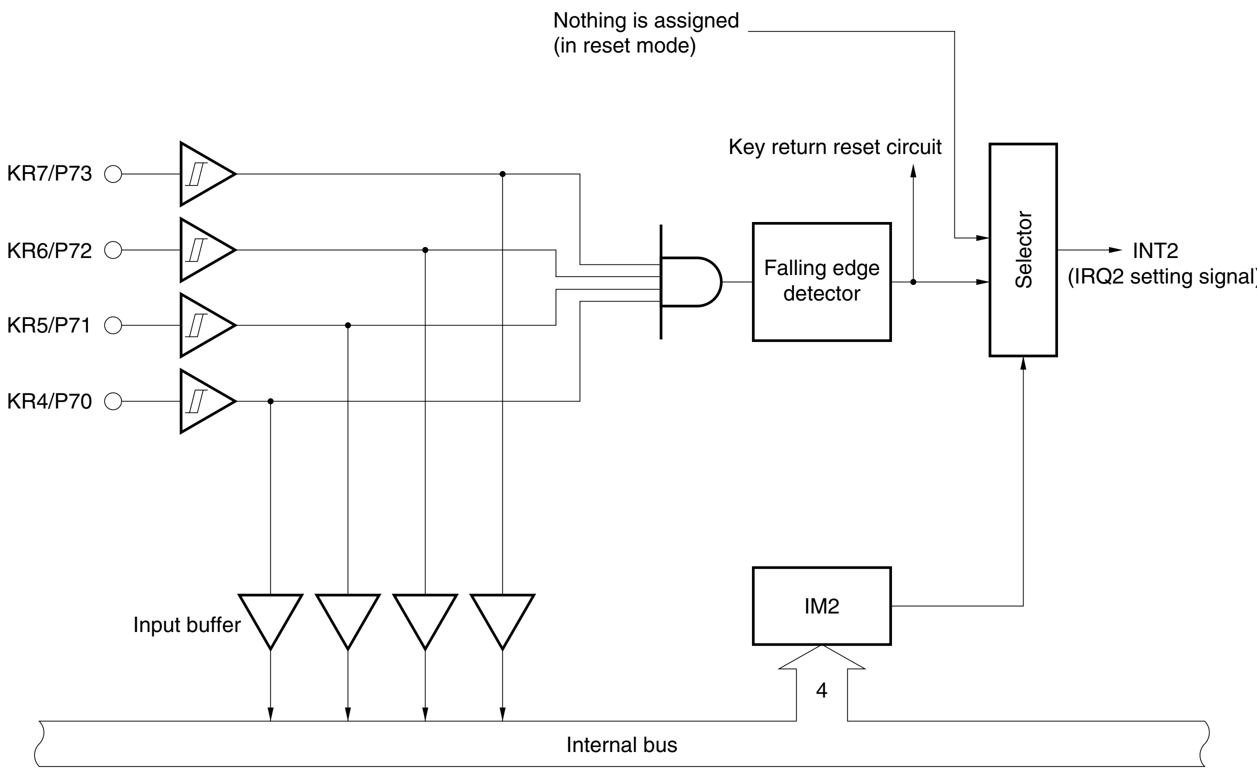
Figure 7-10 shows the configuration of KR4 to KR7.

The IRQ2 setting signal is output when a specified edge is detected on either of the key interrupts. Which pin's falling input is selected is specified by using the INT2 edge detection mode register (IM2).

Figure 7-11 shows the format of IM2. IM2 is set by a 4-bit manipulation instruction. When the reset signal is asserted, all the bits of this register are cleared to "0".

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-10. Block Diagram of KR4 to KR7



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 7-11. 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	Test input pin
0	0	Assigned nothing	–
0	1	Inputs falling edge of any of KR4/P70 to KR7/P73 pin	KR4-KR7
Other		Setting prohibited	

Cautions 1. If the contents of the edge detection mode register are changed, the test request flag may be set. Disable the test input before changing the contents of the mode register. Then, clear the test request flag by the CLR1 instruction and enable the test input.

2. If a low level is input to even one of the KR× pins, IRQ2 is not set even if the falling edge is input to the other pins.

3. On reset, all bits of IM2 become 0. For this reason, nothing is assigned as the N2 test source. When performing an interrupt using a falling edge input on any of pins KR4 to KR7, IM2 must be set to 0001B.

(3) KRREN pin functions

In the STOP mode, when the KRREN pin is high, and a falling edge input is generated any of pins KR4 to KR7, a system reset occurs.

Table 7-7. KR4 to KR7 Pins, KRREN Pin and Test Function

Pins KR4-KR7	Operating Mode	KRREN Pin	Test Function
Falling edge signal generated	Normal operating and HALT mode	Low	Set IRQ2
		High	
	STOP mode	Low	
		High	Disabled for system reset

Furthermore, STOP mode can be released without altering the interrupt enable flag when the KRREN pin is high, by using falling edge input (key return reset) at pin KRn (n = 4 to 7).

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 8 STANDBY FUNCTION

The μPD754244 possesses a standby function that reduces the power consumption 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 system clock oscillator is stopped and therefore the entire system is stopped. The power consumption 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 consumption.

The STOP mode of the μPD754244 can be released by an interrupt request; therefore, the microcontroller can operate intermittently. However, because a certain wait time is required for stabilizing the oscillation of the clock oscillator 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 oscillator continues. This mode does not reduce the power consumption 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 consumption of the overall system can be minimized.

The following page describes the points to be noted in using the standby mode.

Cautions 1. You can operate the μPD754144 efficiently with a low current consumption at a low voltage by selecting the standby mode and CPU clock. In any case, however, the time described in 6.2.3 Setting 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 switching function and standby mode in combination, therefore, set the standby mode after the time required for switching has elapsed.

2. To use the standby mode, process so that the current consumption of the I/O ports is minimized.

Especially, do not leave input ports open; be sure to input either a low or high level.

Download from Www.Somanuals.com. All Manuals Search And Download.

8.1 Settings and Operating Statuses of Standby Mode

Table 8-1. Operating Statuses in Standby Mode

<table>
<tr><th colspan="2"></th><th>STOP Mode</th><th>HALT Mode</th></tr>
<tr><td colspan="2">Instruction to be set</td><td>STOP instruction</td><td>HALT instruction</td></tr>
<tr><td rowspan="5">Operating status</td><td>Clock generator</td><td>Operation stopped</td><td>Only CPU clock Φ is stopped
(oscillation continues)</td></tr>
<tr><td>Basic interval timer/
watchdog timer</td><td>Operation stopped</td><td>Operation possible
(BT mode: Sets IRQBT at reference time intervals
WT mode: Generates reset signal when BT overflows)</td></tr>
<tr><td>Timer counter</td><td>Operation stopped</td><td>Operation possible</td></tr>
<tr><td>External interrupt</td><td colspan="2">INT0 cannot operate[Note]
INT2 can only operate at KRn fall.</td></tr>
<tr><td>CPU</td><td colspan="2">Operation stopped</td></tr>
<tr><td>Release signal</td><td></td><td>• Reset signal
• Interrupt request signal from hardware in which interrupt is enabled
• System reset signal (key return reset) generated by KRn fall when KRREN pin is 1</td><td>• Reset signal
• Interrupt request signal from hardware in which interrupt is enabled</td></tr>
</table>

Note Operation is possible only when the noise eliminator is not selected (when IM02 = 1) by bit 2 of the edge detection mode register (IM0).

Download from Www.Somanuals.com. All Manuals Search And Download.

The STOP mode is set by the STOP instruction, and the HALT mode is set by the HALT instruction (the STOP and HALT instructions respectively set bits 3 and 2 of PCC).

Be sure to write a NOP instruction after the STOP and HALT instructions.

When changing the CPU operating clock by using the lower 2 bits of PCC, a certain time elapses after the bits of PCC have been rewritten until the CPU clock is actually changed, as indicated in **Table 6-5 Maximum Time Required for Changing CPU Clock**. To change the operating clock before the standby mode is set and the CPU clock after the standby mode has been released, set the standby mode after the lapse of the machine cycles necessary for changing the CPU clock, after rewriting the contents of PCC.

In the standby mode, the data is retained for all the registers and data memory that stop in the standby mode, such as general-purpose registers, flags, mode registers, and output latches.

Cautions 1. When the μPD754244 is set in the STOP mode, the X2 pin is internally pulled up to V_{DD} by a resistor of 50 kΩ (typ.).

2. Reset all the interrupt request flags before setting the standby mode.
If there is an interrupt source whose interrupt request flag and interrupt enable flag are both set, the standby mode is released immediately after it has been set (refer to Figure 7-1 Block Diagram of Interrupt Controller).
If the STOP mode is set in the μPD754244, however, the HALT mode is set immediately after the STOP instruction has been executed, and the time set by the BTM register elapses. Then, the normal operation mode is restored.
Also, in the μPD754144, HALT mode is entered immediately after the STOP instruction has been executed, and after a wait of $2^9/f_{CC}$ (512 μs at 1.0 MHz), normal mode operation is restored.

Download from Www.Somanuals.com. All Manuals Search And Download.

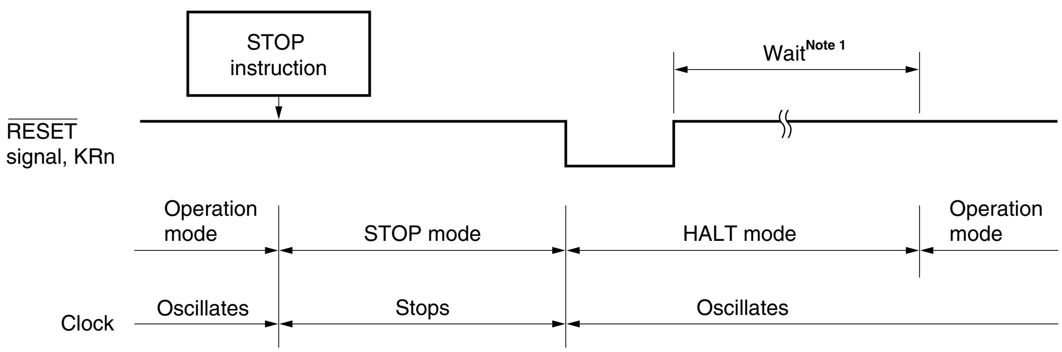
8.2 Releasing Standby Mode

Both the STOP and HALT modes can be released when an interrupt request signal occurs that is enabled by the corresponding interrupt enable flag, or when the $\overline{\text{RESET}}$ signal is asserted. Furthermore, STOP mode can be released without altering the interrupt enable flag, when the KRREN pin is high, by using a falling edge input (key return reset) at the KRn pin.

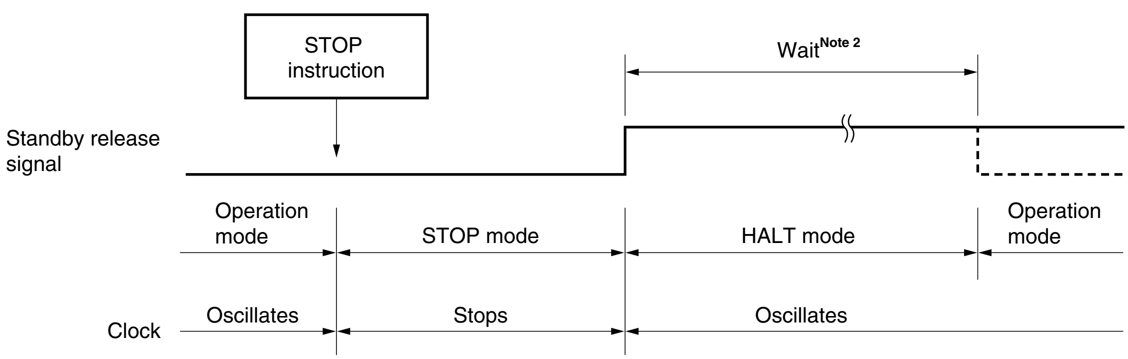
Figure 8-1 illustrates how each mode is released.

Figure 8-1. Releasing Standby Mode (1/2)

(a) Releasing STOP mode by $\overline{\text{RESET}}$ signal, or by key return reset



(b) Releasing STOP mode by interrupt (except for releasing by key return reset)



Notes 1. μPD754244: The following two times can be selected by mask option.

2^{17}/fx (21.8 ms at 6.0 MHz, 31.3 ms at 4.19 MHz)

2^{15}/fx (5.46 ms at 6.0 MHz, 7.81 ms at 4.19 MHz)

μPD754144: The wait time is fixed to 56/fcc (56 μs at 1.0 MHz)

2. μPD754244: The time is set by BTM.

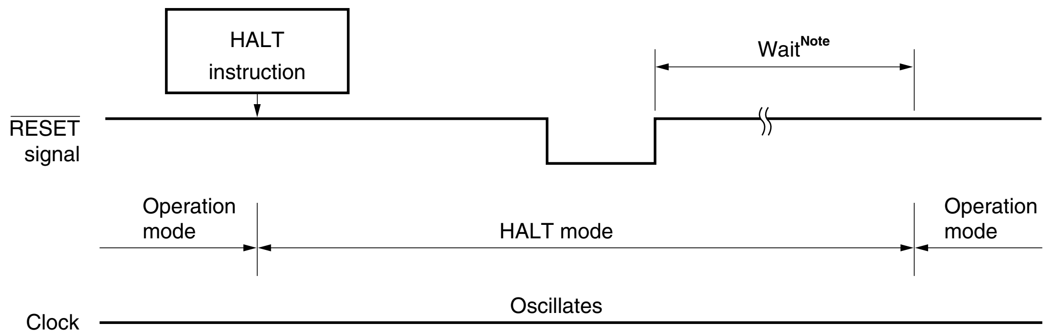
μPD754144: This time is fixed to 2^9/fcc (512 μs at 1.0 MHz).

Remark The broken lines indicate acknowledgment of the interrupt request that releases the standby mode.

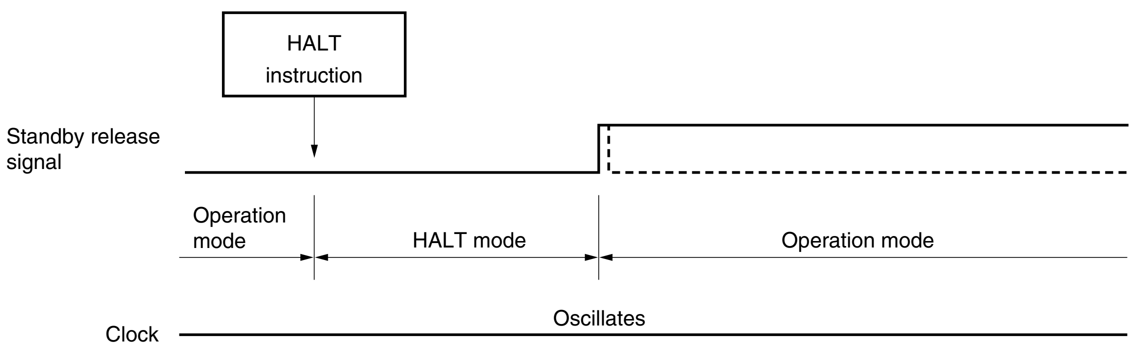
Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 8-1. Releasing Standby Mode (2/2)

(c) Releasing HALT mode by $\overline{\text{RESET}}$ signal



(d) Releasing HALT mode by interrupt



Note μPD754244: The following two times can be selected by mask option.

2^{17}/fx (21.8 ms at 6.0 MHz, 31.3 ms at 4.19 MHz)

2^{15}/fx (5.46 ms at 6.0 MHz, 7.81 ms at 4.19 MHz)

μPD754144: The wait time is fixed to 56/fcc (56 μs at 1.0 MHz)

Remark The broken lines indicate acknowledgment of the interrupt request that releases the standby mode.

The interrupt used to release STOP mode is selected by setting (1) the corresponding interrupt enable flag (IE). STOP mode is released when the interrupt request flag (IRQ) selected in STOP mode is set (1).

In this case, be sure to clear (0) all IEs and IRQs before setting (1) the corresponding interrupt IE. This is because STOP mode is not released by any interrupt other than that selected.

The procedure to release STOP mode by interrupt generation is shown below.

<1> Clear all IEs and IRQs

<2> Set IE for the interrupt used to release STOP mode.

Download from Www.Somanuals.com. All Manuals Search And Download.

<3> Clear again the IRQ used to release STOP mode to enter STOP mode.
In this STOP mode, IRQ of the selected interrupt is set and HALT mode is entered.
Then, after a wait time, the system returns to the normal operating mode.

In the case of the μPD754244, when the STOP mode has been released by an interrupt, the wait time is determined by the setting of BTM (refer to **Table 8-2**).

The time required for the oscillation to stabilize varies depending on the type of oscillator used and the supply voltage when the STOP mode has been released. Therefore, you should select the appropriate wait time depending on the given conditions, and set BTM before setting the STOP mode.

μPD754144: The wait time is fixed to $2^9/f_{CC}$ (512 μs at 1.0 MHz)

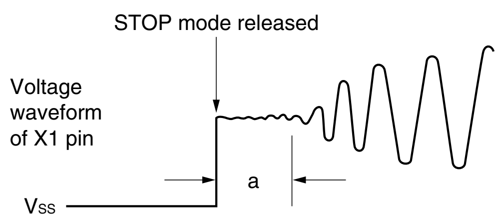
Table 8-2. Selecting Wait Time by BTM

BTM3	BTM2	BTM1	BTM0	Wait Time[Note]	
				f_X = 6.0 MHz	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 above				Setting prohibited	

Note This time does not include the time required to start oscillation after the STOP mode has been released.

Caution In the case of the μPD754244, 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 8-2), regardless of whether the STOP mode has been released by the RESET signal or occurrence of an interrupt.

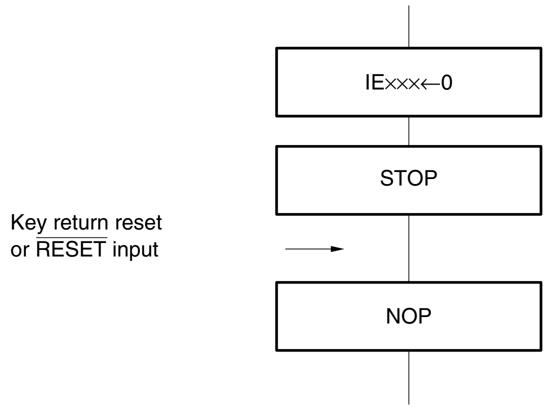
Figure 8-2. Wait Time After Releasing STOP Mode



Before releasing STOP mode by a key return reset or RESET input rather than interrupt input, be sure to clear all interrupt enable flags (including IE2) as shown in Figure 8-3.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 8-3. STOP Mode Release by Key Return Reset or RESET Input



The differences between release by a key return reset and release by RESET input are as follows.

	RESET Input	Key Return Reset
Key return flag (KRF)	0	1
Watchdog flag (WDF)	0	Retained

Download from Www.Somanuals.com. All Manuals Search And Download.

8.3 Operation After Release of Standby Mode

(1) When the standby mode has been released by the $\overline{\text{RESET}}$ signal, the normal reset operation is performed.

(2) When the standby mode has been released by an interrupt, whether or not a vectored interrupt is executed when the CPU has resumed instruction execution is determined by the contents of the interrupt master enable flag (IME).

(a) When IME = 0

Execution is started from the instruction next to the one that set the standby mode after the standby mode has been released. The interrupt request flag is retained.

(b) When IME = 1

A vector interrupt is executed after the standby mode has been released and then two instructions have been executed. However, if the standby mode has been released by INT2 (testable input), servicing same as (a) is performed because no vectored interrupt is generated in this case.

8.4 Application of Standby Mode

Use the standby mode according to the following procedure.

This example applies to the operation of the μPD754244 at f_X = 4.19 MHz. With f_X = 6.0 MHz operation of the μPD754244 and f_{CC} = 1.0 operation of the μPD754144, the CPU clock and the wait time are different even if the settings are the same.

<1> Detect the cause that sets the standby mode such as an interrupt input or power failure by port input.

<2> Process the I/O ports (process so that the current consumption is minimized).
Especially, do not leave the input ports open. Be sure to input a low or high level.

<3> Specify the interrupt that releases the standby mode.

<4> Specify the operation to be performed after the standby mode has been released (manipulate IME depending on whether interrupt servicing 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).

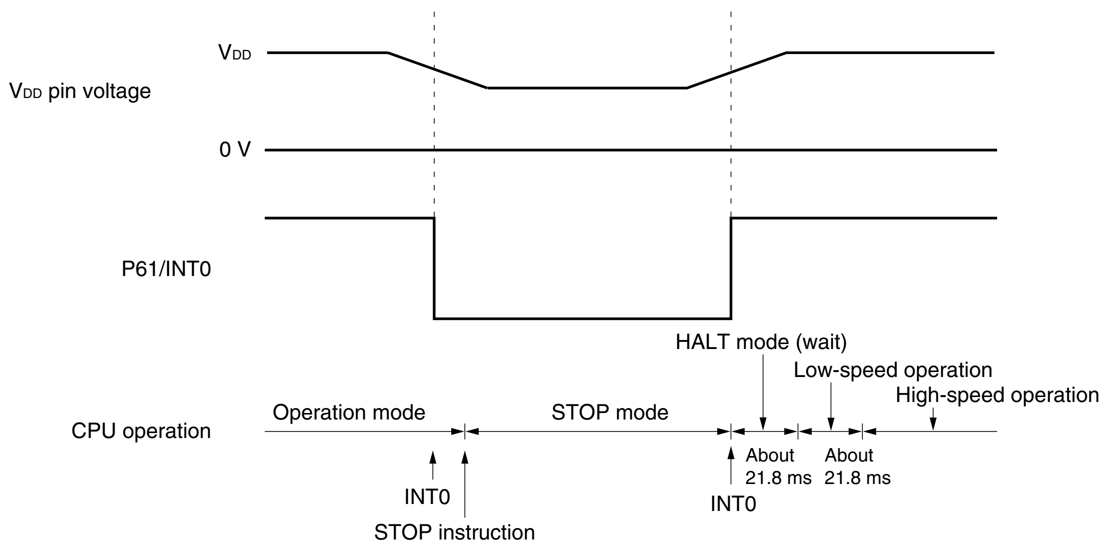
Download from Www.Somanuals.com. All Manuals Search And Download.

(1) Application example of STOP mode (when using the μPD754244 at fX = 6.0 MHz)

<When using the STOP mode under the following conditions>

- The STOP mode is set at the falling edge of INT0 and released at the rising edge.
- All the I/O ports go into a high-impedance state (if the pins are externally processed so that the current consumption is reduced in a high-impedance state).
- Interrupts INTBT 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.
- The wait time that elapses after the mode has been released is about 21.8 ms.
- A wait time of 21.8 ms elapses until the power supply stabilizes after the mode has been released. The P61/INT0 pin is checked twice to prevent chattering.

<Timing chart>



Download from Www.Somanuals.com. All Manuals Search And Download.

<Program example>

(INT0 servicing program, MBE = 0)

```
VSUB0:    SKT   PORT6.1        ; P61 = 1?
          BR    PDOWN          ; Power down
          SET1  BTM.3          ; Power on
WAIT:     SKT   IRQBT          ; Waits for 21.8 ms
          BR    WAIT
          SKT   PORT6.1        ;  Checks chattering
          BR    PDOWN
          MOV   A, #0011B
          MOV   PCC, A         ; Sets high-speed mode
         (MOV   XA.#××H   )    ; Sets port mode register
         (MOV   PMGm, XA  )
          EI    IEBT
          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
          DI    IEBT           ; Disables INTBT and INTT0
          DI    IET0
          MOV   A, #1011B
          MOV   BTM, A         ; Wait time ≒ 21.8 ms
          NOP
          STOP                 ; Sets STOP mode
          NOP
          RETI
```

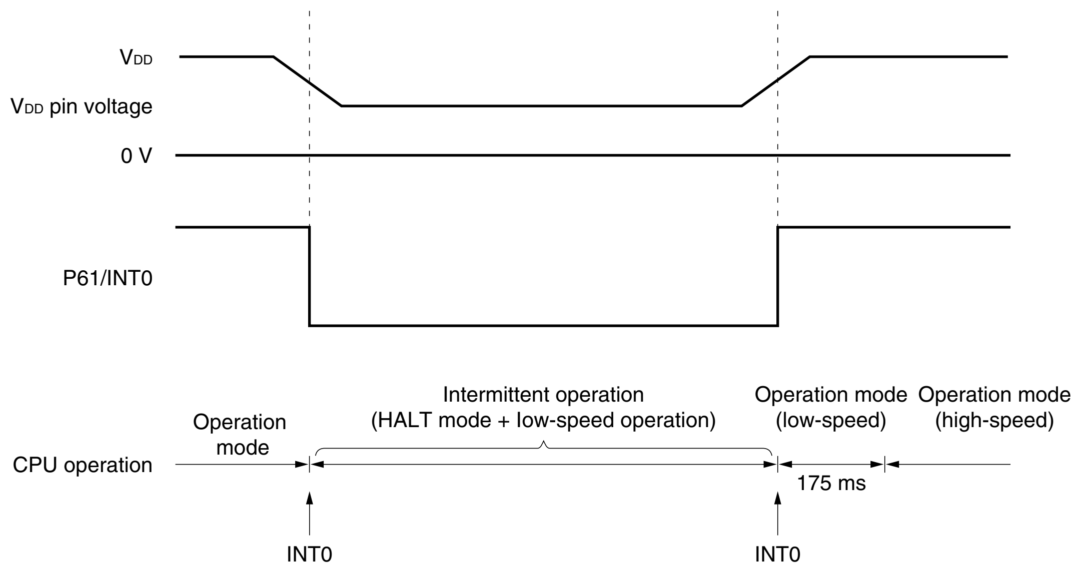

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Application example of HALT mode (when using the μPD754244 at f_X = 6.0 MHz)

<To perform intermittent operation under the following conditions>

- The standby mode is set at the falling edge of INT0 and released at the rising edge.
- In the standby mode, an intermittent operation is performed at intervals of 175 ms (INTBT).
- INT0 and INTBT are assigned a lower priority.
- The slowest CPU clock is selected in the standby mode.

<Timing chart>



Download from Www.Somanuals.com. All Manuals Search And Download.

<Program example>

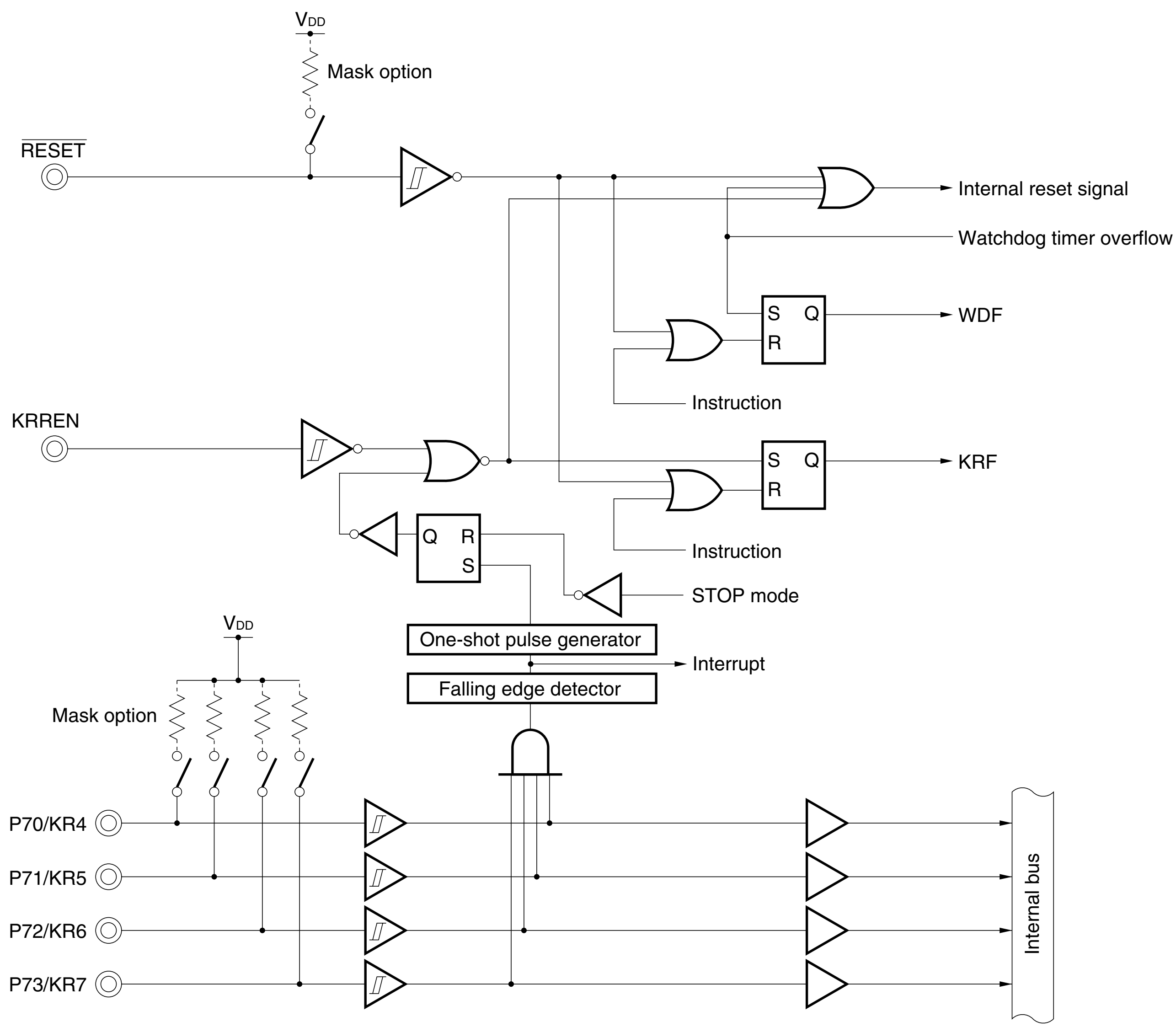
```
BTAND4:   SKTCLR   IRQ0        ; INT0 = 1?
          BR       VSUBBT      ; NO
          SKT      PORT6.1     ; P61 = 1?
          BR       PDOWN       ; Power down
          SET1     BTM.3       ; Starts BT
WAIT:     SKT      IRQBT       ; Waits for 175 ms
          BR       WAIT
          SKT      PORT6.1
          BR       PDOWN
          MOV      A, #0011B   ; High-speed mode
          MOV      PCC, A
          [EI      IEn]        ; IEn ← 1
          RETI
PDOWN:    MOV      A, #0       ; Lowest-speed mode
          MOV      PCC, A    }
          [DI      IEn]      } ; IEn ← 0
                             }   Keeps 46 machine cycles
SETHLT:   HALT                 ; HALT mode
          NOP
          RETI
VSUBBT:   CLR1     IRQBT     }
          :                  }
          :                  }   Processing during intermittent operation
          :                  }
          BR       SETHLT    }
```


Download from Www.Somanuals.com. All Manuals Search And Download.

9.1 Configuration and Operation of Reset Function

Three types of reset signals are used: the external reset signal ($\overline{RESET}$), a reset signal from the basic interval timer/watchdog timer, and a key return reset. When any one of these reset signals is input, the internal reset signal is asserted. Figure 9-1 shows the configuration of the reset circuit.

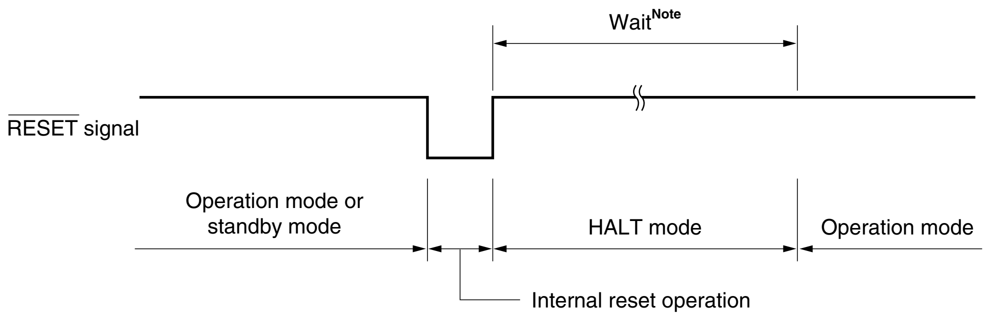
Figure 9-1. Configuration of Reset Circuit



When the $\overline{RESET}$ signal is generated, each hardware unit is initialized as shown in Table 9-1. Figure 9-2 shows the timing of the reset operation.

Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 9-2. Reset Operation by RESET Signal



Note μPD754244: The following two times can be selected by the mask option.

2^{17}/fx (21.8 ms at 6.0 MHz, 31.3 ms at 4.19 MHz)

2^{15}/fx (5.46 ms at 6.0 MHz, 7.81 ms at 4.19 MHz)

μPD754144: The wait time is fixed to 56/fcc (56 μs at 1.0 MHz).

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 9-1. Status of Each Hardware Unit After Reset (1/3)

Hardware		When $\overline{RESET}$ Signal Asserted in Standby Mode	When $\overline{RESET}$ Signal Asserted During Operation
Program counter (PC)		Sets lower 4 bits of program memory address 0000H to PC11-PC8, and contents of address 0001H to PC7-PC0	Same as left
PSW	Carry flag (CY)	Retained	Undefined
	Skip flags (SK0-SK2)	0	0
	Interrupt status flags (IST0, IST1)	0	0
	Bank enable flags (MBE, RBE)	Sets bit 6 of program memory address 0000H to RBE and bit 7 to MBE	Same as left
Stack pointer (SP)		Undefined	Undefined
Stack bank select register (SBS)		1000B	1000B
Data memory (RAM)		Retained	Undefined
Data memory (EEPROM)		Retained**Note 1**	Retained**Note 2**
EEPROM write control register (EWC)		0	0
General-purpose registers (X, A, H, L, D, E, B, C)		Retained	Undefined
Bank select registers (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 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 counter (T1)	Counter (T1)	0	0
	Modulo register (TMOD1)	FFH	FFH
	Mode register (TM1)	0	0
	TOE1, TOUT F/F	0, 0	0, 0

Notes
1. If STOP mode is entered during an EEPROM write operation or if HALT mode is entered during a write operation and then the $\overline{RESET}$ signal is input, this becomes undefined.
2. If the $\overline{RESET}$ signal is input during an EEPROM write operation, the address data becomes undefined.

Download from Www.Somanuals.com. All Manuals Search And Download.

Table 9-1. Status of Each Hardware Unit After Reset (2/3)

Hardware		When $\overline{RESET}$ Signal Asserted in Standby Mode	When $\overline{RESET}$ Signal Asserted During Operation
Timer counter (T2)	Counter (T2)	0	0
	Modulo register (TMOD2)	FFH	FFH
	High-level period setting modulo register (TMOD2H)	FFH	FFH
	Mode register (TM2)	0	0
	TOE2, TOUT F/F	0, 0	0, 0
	REMC, NRZ, NRZB	0, 0, 0	0, 0, 0
Programmable threshold port mode register (PTHM)		00H	00H
Clock generation circuit	Processor clock control register (PCC)	0	0
Interrupt function	Interrupt request flag (IRQ×××)	Reset (0)	Reset (0)
	Interrupt enable flag (IE×××)	0	0
	Interrupt master enable flag (IME)	0	0
	Interrupt priority select register (IPS)	0	0
	INT0, 2 mode registers (IM0, IM2)	0, 0, 0	0, 0, 0
Digital ports	Output buffer	Off	Off
	Output latch	Cleared (0)	Cleared (0)
	I/O mode registers (PMGA, PMGC)	0	0
	Pull-up resistor specification register (POGA, POGB)	0	0
Bit sequential buffers (BSB0 to BSB3)		Retained	Undefined

Table 9-1. Hardware Status After Reset (3/3)

Hardware	Generation of $\overline{RESET}$ Signal by Key Return Reset	Generation of $\overline{RESET}$ Signal in Standby Mode	Generation of $\overline{RESET}$ Signal by WDT in Operation	Generation of $\overline{RESET}$ Signal in Operation
Watchdog flag (WDF)	Retains the previous state.	0	1	0
Key return flag (KRF)	1	0	Retains the previous state.	0

Download from Www.Somanuals.com. All Manuals Search And Download.

9.2 Watchdog Flag (WDF), Key Return Flag (KRF)

WDF and KRF are mapped to bit 2 and 3 of address FC6H respectively.

The contents of WDF and KRF are undefined initially, but they are initialized to “0” by external $\overline{\text{RESET}}$ signal generation.

WDF is cleared by a watchdog timer overflow signal, and KRF is set by a reset signal generated by the KRn pin. As a result, by checking the contents of WDF and KRF, it is possible to know what kind of reset signal is generated.

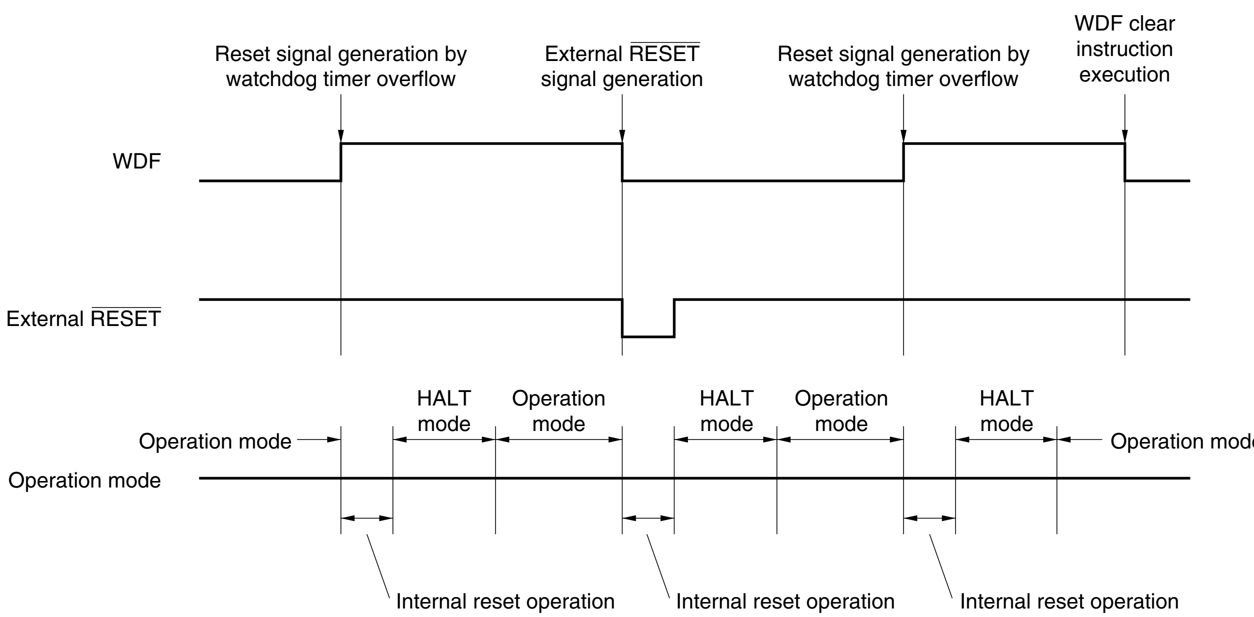
As WDF and KRF are cleared only by an external signal or instruction execution, once these flags are set, they are not cleared until an external signal is generated or a clear instruction is executed. Check and clear the contents of WDF and KRF after the reset start operation by executing the SKTCLR instruction, etc.

Table 9-2 lists the contents of WDF and KRF corresponding to each signal. Figure 9-3 shows the WDF operation in generating each signal, and Figure 9-4 shows the KRF operation in generating each signal.

Table 9-2. WDF and KRF Contents Corresponding to Each Signal

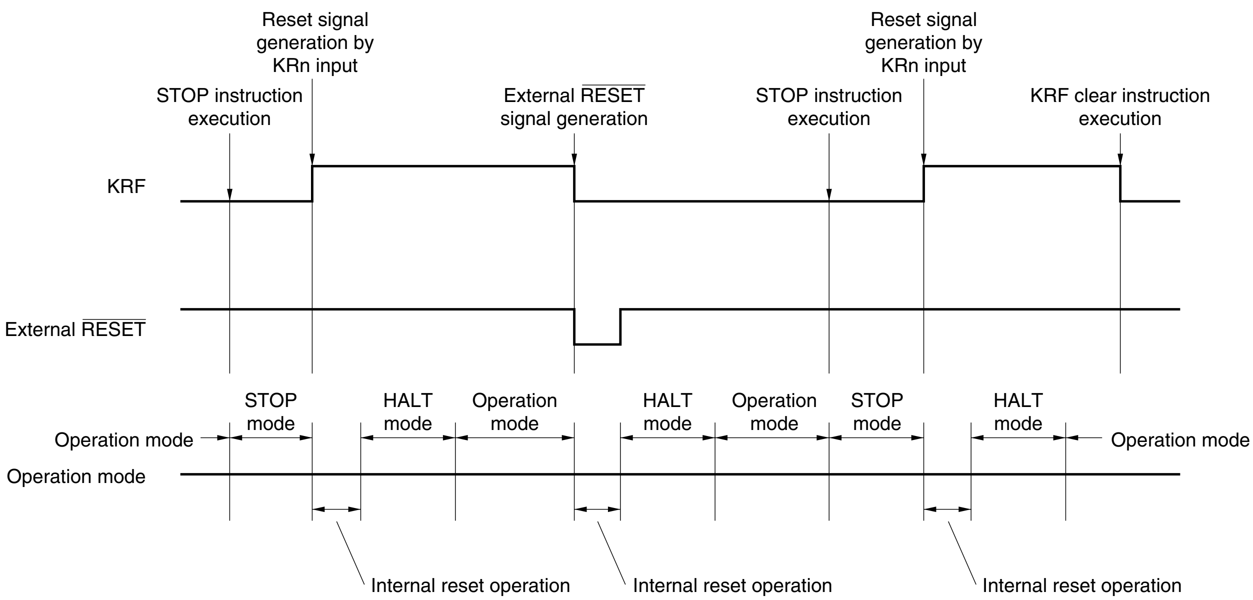
Hardware	External $\overline{\text{RESET}}$ Signal Generation	Reset Signal Generation by Watch-dog Timer Overflow	Reset Signal Generation by KRn Input	WDF Clear Instruction Execution	KRF Clear Instruction Execution
Watchdog flag (WDF)	0	1	Hold	0	Hold
Key return flag (KRF)	0	Hold	1	Hold	0

Figure 9-3. WDF Operation in Generating Each Signal



Download from Www.Somanuals.com. All Manuals Search And Download.

Figure 9-4. KRF Operation in Generating Each Signal



Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 10 MASK OPTIONS

The μPD754144 and 754244 have the following mask options.

Table 10-1. Selection of Mask Options

Item	μPD754144	μPD754244
P70/KR4 to P73/KR7	On-chip pull-up resistors specifiable in 1-bit units by mask option	
$\overline{\text{RESET}}$ pin	On-chip pull-up resistors specifiable by mask option	
Oscillation stabilization wait time	Fixed to $56/f_{CC}$	Selectable from $2^{17}/f_X$ and $2^{15}/f_X$

10.1 Pin Mask Options

10.1.1 Mask option of P70/KR4 to P73/KR7

On-chip 100 kΩ (typ.) pull-up resistors can be specified by mask option for P70/KR4 to P73/KR7 (port 7).
The mask option can be specified in 1-bit units.

10.1.2 $\overline{\text{RESET}}$ pin mask option

On-chip 100 kΩ (typ.) pull-up resistors can be specified by mask option for the $\overline{\text{RESET}}$ pin.

10.2 Oscillation Stabilization Wait Time Mask Option

The oscillation stabilization wait time mask option differs between the μPD754144 and 754244.

In the μPD754244, it is possible to select a wait time by mask option. This wait time refers to the time after the standby function is released by the $\overline{\text{RESET}}$ signal until the system returns to the normal operation mode (refer to **8.2 Releasing Standby Mode**, for details.)

The wait time can be selected from the following two times.

(1) $2^{17}/f_X$ (21.8 ms: at f_X = 6.0 MHz, 31.3 ms: at f_X = 4.19 MHz)
(2) $2^{15}/f_X$ (5.46 ms: at f_X = 6.0 MHz, 7.81 ms: at f_X = 4.19 MHz)

The μPD754144 has no mask option and the wait time is fixed to $56/f_{CC}$ (56 μs at 1.0 MHz).

Download from Www.Somanuals.com. All Manuals Search And Download.

CHAPTER 11 INSTRUCTION SET

The instruction set of the μPD754244 is based on the instruction set of the 75X Series and therefore maintains compatibility with the 75X Series, but with the following improved features.

(1) Bit manipulation instructions for various applications
(2) Efficient 4-bit manipulation instructions
(3) 8-bit manipulation instructions comparable to those of 8-bit microcontrollers
(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 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 instructions unique to the μPD754244's instruction set.

11.1.1 GETI instruction

The GETI instruction converts the following instructions into 1-byte instructions.

(a) Subroutine call instruction to 4 KB space (0000H to 0FFFH)
(b) Branch instruction to 4 KB space (0000H to 0FFFH)
(c) Any 2-byte, 2-machine-cycle instruction (except BRCB and CALLF instructions)
(c) Combination of two 1-byte instructions

The GETI instruction references a table at addresses 0020H to 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.1.2 Bit manipulation instruction

The μPD754244 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, IE×××, IRQ×××	FB0H to FBFH
	PORT3, 6, 7, 8	FF0H to FFFH
pmem. @L	BSB0-3, PORT8	FC0H to 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, 2, T0, T1, T2, EE
2. MB = MBE · MBS

11.1.3 String-effect instruction

The μPD754244 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 NOP instructions. If the address executed first is A1, the following instruction is replaced with a 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 (register A or register pair XA) and data pointer (register pair HL).

Download from Www.Somanuals.com. All Manuals Search And Download.

11.1.4 Base number adjustment instruction

Some applications require that the result of addition or subtraction of 4-bit data (which is carried out in binary) be converted into a decimal number or into a number with a base of 6, such as time.

Therefore, the μPD754244 is provided with base number adjustment instructions that adjust the result of addition or subtraction of 4-bit data into a number with any base.

(1) 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 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
          ⋮
```

(2) 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.1.5 Skip instruction and number of machine cycles required for skipping

The instruction set of the μPD754244 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 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 Set and Operation

(1) Operand representation and description

Describe an operand in the operand field of each instruction according to the operand description method of the instruction (for details, refer to **RA75X Assembler Package Language User's Manual**. If two or more operands are shown, select one of them. Uppercase letters, +, and - are keywords and must be described as is.

The symbols of register flags can be described as labels, instead of mem, fmem, pmem, and bit. (However, the number of labels described for fmem and pmem are limited. For details, refer to **Table 3-1 Addressing Modes** and **Figure 3-7 μPD754244 I/O Map**).

Download from Www.Somanuals.com. All Manuals Search And Download.

Representation	Description
reg	X, A, B, C, D, E, H, L
reg1	X, B, C, D, E, H, L
rp	XA, BC, DE, HL
rp1	BC, DE, HL
rp2	BC, DE
rp'	XA, BC, DE, HL, XA', BC', DE', HL'
rp'1	BC, DE, HL, XA', BC', DE', HL'
rpa	HL, HL+, HL–, DE, DL
rpa1	DE, DL
n4	4-bit immediate data or label
n8	8-bit immediate data or label
mem	8-bit immediate data or label[Note]
bit	2-bit immediate data or label
fmem	Immediate data FB0H to FBFH, FF0H to FFFH or label
pmem	Immediate data FC0H to FFFH or label
addr	Immediate data 0000H to 0FFFH or label
addr1	Immediate data 0000H to 0FFFH or label
caddr	12-bit immediate data or label
faddr	11-bit immediate data or label
taddr	Immediate data 20H to 7FH (where bit0 = 0) or label
PORTn	PORT3, 6, 7, 8
IE×××	IEBT, IET0 to IET2, IE0, IE2, IEEE
RBn	RB0 to RB3
MBn	MB0, MB4, MB15

Note mem can be described only for an even address for 8-bit data processing.

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Conventions for 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:	Register pair (XA); 8-bit accumulator
BC:	Register pair (BC)
DE:	Register pair (DE)
HL:	Register pair (HL)
XA':	Expansion register pair (XA')
BC':	Expansion register pair (BC')
DE':	Expansion register pair (DE')
HL':	Expansion register pair (HL')
PC:	Program counter
SP:	Stack pointer
CY:	Carry flag; bit accumulator
PSW:	Program status word
MBE:	Memory bank enable flag
RBE:	Register bank enable flag
PORTn:	Port n (n = 3, 6, 7, 8)
IME:	Interrupt master enable flag
IPS:	Interrupt priority select register
IE×××:	Interrupt enable flag
RBS:	Register bank select flag
MBS:	Memory bank select flag
PCC:	Processor clock control register
. :	Address or bit delimiter
(××):	Contents addressed by ××
××H:	Hexadecimal data

Download from Www.Somanuals.com. All Manuals Search And Download.

(3) Symbols in addressing area field

*1	MB = MBE MBS (MBS = 0, 4, 15)	Data memory addressing
*2	MB = 0	
*3	MBE = 0: MB = 0 (000H to 07FH) MB = 15 (F80H to FFFH) MBE = 1: MB = MBS (MBS = 0, 4, 15)	
*4	MB = 15, fmem = FB0H to FBFH, FF0H to FFFH	
*5	MB = 15, pmem = FC0H to FFFH	
*6	addr = 0000H to 0FFFH	Program memory addressing
*7	addr, addr1 = (Current PC) – 15 to (Current PC) –1 (Current PC) + 2 to (Current PC) +16	
*8	caddr = 0000H to 0FFFH	
*9	taddr = 000H to 07FFH	
*10	taddr = 0020H to 007FH	
*11	addr1 = 0000H to 0FFFH	

Remarks 1. MB indicates a memory bank that can be accessed.

2. In *2, MB = 0 regardless of MBE and MBS.

3. In *4 and *5, MB = 15 regardless of MBE and MBS.

4. *6 to *11 indicate areas that can be addressed.

Download from Www.Somanuals.com. All Manuals Search And Download.

(4) Explanation of machine cycle field

S indicates the number of machine cycles required for an instruction with skip to execute the skip operation. The value of S varies as follows.

- When skip is executed .... S = 0
- When 1- or 2-byte instruction is skipped .... S = 1
- When 3-byte instruction[Note] is skipped .... S = 2

Note 3-byte instructions: BR !addr, BRA !addr1, CALL !addr, CALLA !addr1

Caution The GETI instruction is skipped in one machine cycle.

One machine cycle is equal to one cycle of CPU clock Φ (= t_{CY}), and four times can be set by PCC (refer to **Figure 6-15 Processor Clock Control Register Format**).

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Transfer	**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 ↔ (rpa1)	*2	
		XA, @HL	2	2	XA ↔ (HL)	*1	
		A, mem	2	2	A ↔ (mem)	*3	
		XA, mem	2	2	XA ↔ (mem)	*3	
		A, reg1	1	1	A ↔ reg1		
		XA, rp'	2	2	XA ↔ rp'		

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Table reference	**MOVT**	XA, @PCDE	1	3	XA ← $(PC_{11-8} + DE)_{ROM}$		
		XA, @PCXA	1	3	XA ← $(PC_{11-8} + XA)_{ROM}$		
		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', CY ← rp'1 + XA + CY		
	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		
Operation	**AND**	A, #n4	2	2	A ← A ∧ n4		
		A, @HL	1	1	A ← A ∧ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∧ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∧ XA		
	OR	A, #n4	2	2	A ← A ∨ n4		
		A, @HL	1	1	A ← A ∨ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∨ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∨ XA		
	XOR	A, #n4	2	2	A ← A ∀ n4		
		A, @HL	1	1	A ← A ∀ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∀ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∀ XA		

Note Set 0 to the B register.

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Accumulator manipulation	**RORC**	A	1	1	CY ← A_0, A_3 ← CY, A_{n-1} ← A_n		
	NOT	A	2	2	A ← $\overline{A}$		
Increment/ decrement	**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	**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'
Carry flag manipulation	**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	**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(mem.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

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Memory bit manipula-tion	**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	
Branch	**BR**[Note1]	addr	–	–	PC$_{11-0}$ ← addr Optimum instruction is selected by assembler from following: BR !addr BRCB !caddr BR $addr1	*6	
		addr1	–	–	PC$_{11-0}$ ← addr1 Optimum instruction is selected by assembler from following: BR !addr BRA !addr1 BRCB !caddr BR $addr1	*11	
		!addr	3	3	PC$_{11-0}$ ← addr	*6	
		$addr	1	2	PC$_{11-0}$ ← addr	*7	
		$addr1	1	2	PC$_{11-0}$ ← addr1	*7	
		PCDE	2	3	PC$_{11-0}$ ← PC$_{11-8}$ + DE		
		PCXA	2	3	PC$_{11-0}$ ← PC$_{11-8}$ + XA		
		BCDE	2	3	PC$_{11-0}$ ← BCDE[Note2]	*6	
		BCXA	2	3	PC$_{11-0}$ ← BCXA[Note2]	*6	
	BRA[Note1]	!addr1	3	3	PC$_{11-0}$ ← addr1	*11	
	BRCB	!caddr	2	2	PC$_{11-0}$ ← caddr$_{11-0}$	*8	

Notes **1.** The shaded portion is supported only in the MkII mode. All others are supported only in the MkI mode.
2. Set 0 to the B register

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Subroutine/stack control	**CALLA**[Note]	!addr1	3	3	(SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 (SP–2) ← ×, ×, MBE, RBE PC_{11-0} ← addr1, SP ← SP – 6	*11	
	CALL[Note]	!addr	3	3	(SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP–3) ← MBE, RBE, 0 PC_{11-0} ← addr, SP ← SP – 4	*6	
				4	(SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 (SP–2) ← ×, ×, MBE, RBE PC_{11-0} ← addr, SP ← SP – 6		
	CALLF[Note]	!faddr	2	2	(SP–4) (SP–1) (SP–2) ← PC_{11-0} (SP – 3) ← MBE, RBE, 0, 0 PC_{11-0} ← 0 + faddr, SP ← SP – 4	*9	
				3	(SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 (SP – 2) ← ×, ×, MBE, RBE PC_{11-0} ← 0 + faddr, SP ← SP – 6		
	RET[Note]		1	3	MBE, RBE, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) SP ← SP + 4		
					×, ×, MBE, RBE ← (SP + 4) 0, 0, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2), SP ← SP + 6		
	RETS[Note]		1	3 + S	MBE, RBE, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) SP ← SP + 4 then skip unconditionally		Unconditional
					×, ×, MBE, RBE ← (SP + 4) 0, 0, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2), SP ← SP + 6 then skip unconditionally		
	RETI[Note]		1	3	MBE, RBE, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5), SP ← SP + 6		
					0, 0, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5), SP ← SP + 6		

Note The shaded portion is supported only in the MkII mode. All others are supported only in the MkI mode.

Download from Www.Somanuals.com. All Manuals Search And Download.

Instructions	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Subroutine/stack control	**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	**EI**		2	2	IME (IPS.3) ← 1		
		IE×××	2	2	IE××× ← 1		
	DI		2	2	IME (IPS.3) ← 0		
		IE×××	2	2	IE××× ← 0		
I/O	**IN**[Note1]	A, PORT$_n$	2	2	A ← PORT$_n$ (n = 3, 6, 7, 8)		
	OUT[Note1]	PORT$_n$, A	2	2	PORT$_n$ ← A (n = 3, 6, 8)		
CPU control	**HALT**		2	2	Set HALT Mode (PCC.2 ← 1)		
	STOP		2	2	Set STOP Mode (PCC.3 ← 1)		
	NOP		1	1	No Operation		
Special	**SEL**	RBn	2	2	RBS ← n (n = 0-3)		
		MBn	2	2	MBS ← n (n = 0, 4, 15)		
	GETI[Note2, 3]	taddr	1	3	. TBR instruction PC$_{11-0}$ ← (taddr)$_{3-0}$ + (taddr+1)	*10	
					. TCALL instruction (SP–4) (SP–1) (SP–2) ← PC$_{11-0}$ (SP–3) ← MBE, RBE, 0, 0 PC$_{11-0}$ ← (taddr)$_{3-0}$ + (taddr+1) SP ← SP–4		
					. Other than TBR and TCALL instructions Executes instruction of (taddr) (taddr+1)		Depends on referenced instruction
			1	3	. TBR instruction PC$_{11-0}$ ← (taddr)$_{3-0}$ + (taddr+1)		
				4	. TCALL instruction (SP–6) (SP–3) (SP–4) ← PC$_{11-0}$ (SP–5) ← 0, 0, 0, 0 (SP–2) ← ×, ×, MBE, RBE PC$_{11-0}$ ← (taddr)$_{3-0}$ + (taddr+1) SP ← SP-6		
				3	. Other than TBR and TCALL instructions Executes instruction of (taddr) (taddr+1)		Depends on referenced instruction

Notes
1. To execute an IN/OUT instruction, it is necessary that MBE = 0 or MBE = 1, MBS = 15.
2. The shaded portion is supported only in the MkII mode. All others are supported only in the MkI mode.
3. The TBR and TCALL instructions are the assembler directives for table definition.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.3 Opcode of Each Instruction

(1) Description of symbol of opcode

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	1	0	0	IET0
0	1	1	0	IE0
0	1	1	1	IE2
1	0	0	1	IEEE
1	1	0	0	IET1
1	1	0	1	IET2

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)]

Download from Www.Somanuals.com. All Manuals Search And Download.

(2) Opcode 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 opcode corresponding to the above addressing is as follows.

*1	2nd Byte of Opcode	Accessible Bit
fmem. bit	1 0 B_1 B_0 F_3 F_2 F_1 F_0	Bit of FB0H to FBFH that can be manipulated
	1 1 B_1 B_0 F_3 F_2 F_1 F_0	Bit of FF0H to FFFH that can be manipulated
pmem. @L	0 1 0 0 G_3 G_2 G_1 G_0	Bit of FC0H to 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 lower 4 bits of address)

G_n: immediate data for pmem
(indicates bits 5-2 of address)

D_n: immediate data for mem
(indicates lower 4 bits of address)

Download from Www.Somanuals.com. All Manuals Search And Download.

Instruction	Mnemonic	Operand	Opcode		
			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 reference	**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	

Download from Www.Somanuals.com. All Manuals Search And Download.

Instruction	Mnemonic	Operand	Opcode		
			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	

Download from Www.Somanuals.com. All Manuals Search And Download.

Instruction	Mnemonic	Operand	Opcode		
			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 P_0		
		@HL	1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0	
		mem	1 0 0 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
	DECS	reg	1 1 0 0 1 R_2 R_1 R_0		
		rp'	1 0 1 0 1 0 1 0	0 1 1 0 1 P_2 P_1 P_0	
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 manipula- tion	**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	

Download from Www.Somanuals.com. All Manuals Search And Download.

Instruction	Mnemonic	Operand	Opcode		
			B_1	B_2	B_3
Branch	**BR**	!addr	1 0 1 0 1 0 1 1	0 0 ←——	addr ——→
		$addr1 (+16) ~ (+2)	0 0 0 0 A_3 A_2 A_1 A_0		
		(−1) ~ (−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 ——→	
Subrou-tine/stack control	**CALLA**	!addr1	1 0 1 1 1 0 1 1	0 ←——	addr1 ——→
	CALL	!addr	1 0 1 0 1 0 1 1	0 1 ←——	addr ——→
	CALLF	!faddr	0 1 0 0 0 ←——	faddr ——→	
	RET		1 1 1 0 1 1 1 0		
	RETS		1 1 1 0 0 0 0 0		
	RETI		1 1 1 0 1 1 1 1		
	PUSH	rp	0 1 0 0 1 P_2 P_1 1		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 1	
	POP	rp	0 1 0 0 1 P_2 P_1 0		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 0	
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	
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	
	OUT	PORTn, A	1 0 0 1 0 0 1 1	1 1 1 1 N_3 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		

Download from Www.Somanuals.com. All Manuals Search And Download.

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 MkI and MkII modes of the μPD754144, and 754244. Read the descriptions on the following pages according to the following guidance.

How to read

◯ : This instruction can be used commonly in the MkI and MkII modes of the μPD754144 and 754244.

(I) : This instruction can be used only in the MkI mode of the μPD754144 and 754244.

(II) : This instruction can be used only in the MkII mode of the μPD754144 and 754244.

(I/II) : This instruction can be used commonly in the MkI and MkII modes of the μPD754144, and 754244, but the function may differ between the MkI and MkII modes.
In the MkI mode, refer to the description under the heading [MkI mode]. In the MkII mode, read the description under the heading [MkII mode].

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.1 Transfer instructions

◯ MOV A, #n4

Function: A ← n4 n4 = $I_{3\text{-}0}$: 0-FH

Transfers 4-bit immediate data n4 to the A register (4-bit accumulator). This instruction has a string effect (group A), and if MOV A, #n4 or MOV XA, #n8 follows this instruction, the string-effect instruction following the instruction executed is processed as NOP.

Application example

(1) To set 0BH to the accumulator

```
        MOV A, #0BH
```

(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 = $I_{3\text{-}0}$ 0 to 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 = $I_{7\text{-}0}$: 00H to FFH

Transfers 8-bit immediate data n8 to register pair XA. This instruction has a string effect, and if the same instruction or an MOV A, #n4 instruction follows this instruction, the string-effect instruction following the instruction executed is processed as NOP.

◯ MOV HL, #n8

Function: HL ← n8 n8 = $I_{7\text{-}0}$: 00H-FFH

Transfers 8-bit immediate data n8 to register pair HL. This instruction has a string effect, and if the same instruction follows this instruction, the string-effect instructions following the instruction executed is processed as NOP.

◯ MOV rp2, #n8

Function: rp2 ← n8 n8 = $I_{7\text{-}0}$: 00H to FFH

Transfers 8-bit immediate data n8 to register pair rp2 (BC, DE).

Download from Www.Somanuals.com. All Manuals Search And Download.

○ MOV A, @HL

Function: A ← (HL)

Transfers the contents of the data memory content addressed by register pair HL is transferred to the A register.

○ MOV A, @HL+

Function: A ← (HL), L ← L+1
skip if L = 0H

Transfers the contents of the data memory addressed by register pair HL to the A register. Then, the contents of the L register are automatically incremented by one, and if the contents of the L register become 0H as a result, the next instruction is skipped.

○ MOV A, @HL–

Function: A ← (HL), L ← L–1
skip if L = FH

Transfers the contents of the data memory addressed by register pair HL to the A register. Then, the contents of the L register are automatically decremented by one, and if the contents of the L register become FH as a result the next instruction is skipped.

○ MOV A, @rpa1

Function: A ← (rpa)
Where rpa = HL+: skip if L = 0
where rpa = HL–: skip if L = FH

Transfers the contents of the data memory addressed by register pair rpa (HL, HL+, HL–, DE, or DL) to the A register.

If autoincrement (HL+) is specified as rpa, 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 instruction is skipped.

If autodecrement (HL–) is specified as rpa, 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 instruction is skipped.

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 = $D_{7\text{-}0}$: 00H to FFH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register.

◯ MOV XA, mem

Function: A ← (mem), X ← (mem+1) mem = $D_{7\text{-}0}$: 00H to FEH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register and the contents of the next address to the X register.

The address 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
```


Download from Www.Somanuals.com. All Manuals Search And Download.

◯ MOV mem, A

Function: (mem) ← A mem = D_{7-0}: 00H to FFH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem.

◯ MOV mem, XA

Function: (mem) ← A, (mem+1) ← X mem = D_{7-0}: 00H to FEH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem 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').

Download from Www.Somanuals.com. All Manuals Search And Download.

⬭ XCH A, @HL

Function: A ↔ (HL)

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL.

⬭ XCH A, @HL+

Function: A ↔ (HL), L ← L+1
skip if L = 0H

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL. Then, the contents of the L register are automatically incremented by one, and if the contents of the L register become 0H as a result, the next 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 register pair HL. Then, the contents of the L register are automatically decremented by one, and if the contents of the L register become FH as a result, the next instruction is skipped.

⬭ XCH A, @rpa1

Function: A ↔ (rpa)
Where rpa = HL+: skip if L = 0
Where rpa = HL–: sKIP if L = FH

Exchanges the contents of the A register with the contents of the data memory addressed by register pair rpa (HL, HL+, HL–, DE, or DL). If autoincrement (HL+) or autodecrement (HL–) is specified as rpa, the contents of the L register are automatically incremented or decremented by one after the data have been exchanged. If the result is 0 in the case of HL+ and FH in the case of HL–, the next instruction is skipped.

Application example

To exchange the data at data memory addresses 20H to 2FH with the data at addresses 30H to 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
```


Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 an odd number, however, an address whose least significant bit is ignored is specified.

◯ XCH A, mem

Function: A ↔ (mem) mem = D_{7-0}: 00H to FEH

Exchanges the contents of the A register with the contents of the data memory addressed by 8-bit immediate data mem.

◯ XCH XA, mem

Function: A ↔ (mem), X ↔ (mem+1) mem = D_{7-0}: 00H to 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').

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.2 Table reference instructions

◯ MOV XA, @PCDE

Function: XA ← ROM (PC_{11-8}+DE)

Transfers the lower 4 bits of the table data in the program memory addressed when the lower 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair DE, to the A register, and the higher 4 bits to the X register.

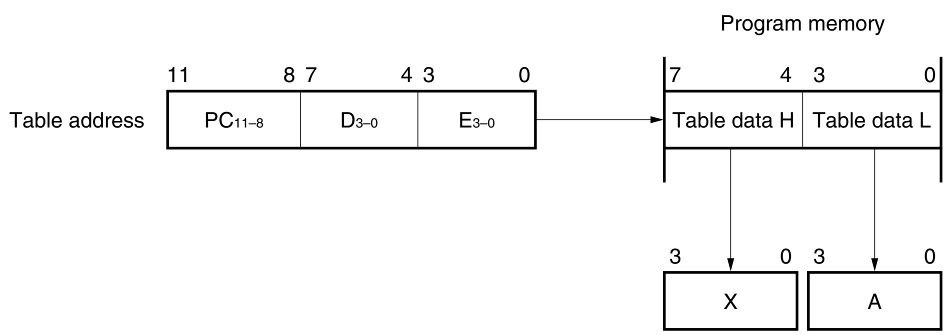
The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler directive (DB instruction).

The program counter is not affected by execution of this instruction.

This instruction is useful for successively referencing table data.

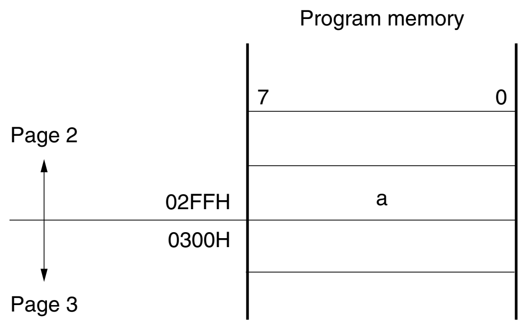
Example



Download from Www.Somanuals.com. All Manuals Search And Download.

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 next page is referenced instead of the table data in the page where the instruction exists.



For example, if the MOVT 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 0×F0H to 0×FFH to data memory addresses 30H to 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   0×F0H
        DB    ××H, ××H, ... ; table data
```


Download from Www.Somanuals.com. All Manuals Search And Download.

◯ MOVT XA, @PCXA

Function: XA ← ROM (PC_{11-8}+XA)

Transfers the lower 4 bits of the table data in the program memory addressed when the lower 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair XA, to the A register, and the higher 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 directive (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.

◯ MOVT XA, @BCDE

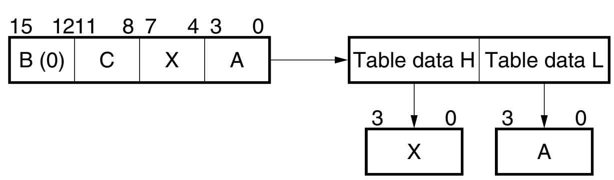
Function: XA ← ROM (BCDE)

Transfers the lower 4 bits of the table data (8-bit) in the program memory addressed by the register B and the contents of registers C, D, and E, to the A register, and the higher 4 bits to the X register.

However, in the μPD754244, register B is invalid. Be sure to set register B to 0000B.

The necessary data must be programmed to the table area in advance by using an assembler directive (DB instruction). The PC is not affected by execution of this instruction.

Example



Download from Www.Somanuals.com. All Manuals Search And Download.

○ MOVT XA, @BCXA

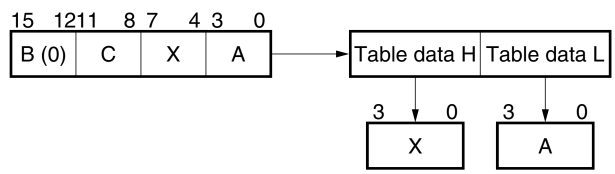
Function: XA ← ROM (BCXA)

Transfers the lower 4 bits of the table data (8-bit) in the program memory addressed by the B register and the contents of registers C, X, and A, to the A register, and the higher 4 bits to the X register.

However, on the μPD754244, register B is invalid. Be sure to set register B to 0000B.

The necessary data must be programmed to the table area in advance by using an assembler directive (DB instruction). The PC is not affected by execution of this instruction.

Example



Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.3 Bit transfer instructions

◯ MOV1 CY, fmem.bit

◯ MOV1 CY, pmem.@L

◯ MOV1 CY, @H+mem.bit

Function: CY ← (bit specified by operand)

Transfers the contents of the data memory 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 SHR 6 ; H ← higher 4 bits of FLAG
MOV1    CY, @H+FLAG    ; CY ← FLAG
MOV1    PORT3.2, CY    ; P32 ← CY
```


Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.4 Operation instructions

◯ ADDS A, #n4

Function: A ← A+n4; Skip if carry. n4 = I_{3-0}: 0 to FH

Adds 4-bit immediate data n4 to the contents of the A register. 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 **11.1.4 Base number adjustment instruction**).

◯ ADDS XA, #n8

Function: XA ← XA+n8; Skip if carry. n8 = I_{7-0}: 00H to FFH

Adds 8-bit immediate data n8 to the contents of register pair XA. 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. 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. 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'). 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
```


Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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, 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 next to 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 to **11.1.4 Base number adjustment instruction**).

◯ 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, 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’), 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 the ADDS A, #n4 instruction is placed next to 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 **11.1.4 Base number adjustment instruction**).

◯ 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

○ 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 higher 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 a specified register pair.

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 lower 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 a specified register pair.

◯ 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 higher 4 bits of the accumulator

XOR A, #1000B

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 a specified register pair.

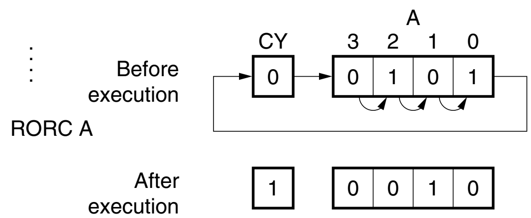
Download from Www.Somanuals.com. All Manuals Search And Download.

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 left 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).

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.6 Increment/decrement instructions

◯ INCS reg

Function: reg ← reg+1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L, D, E, B, 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 to 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’ = FFH

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.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.7 Compare instructions

◯ SKE reg, #n4

Function: Skip if reg = n4 n4 = $I_{3\text{-}0}$: 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 = $I_{3\text{-}0}$: 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, an address whose least significant address is ignored is specified.

◯ SKE A, reg

Function: Skip if A = reg

Skips the next 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 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’).

Download from Www.Somanuals.com. All Manuals Search And Download.

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 instruction if the carry flag is 1.

◯ NOT1 CY

Function: CY ← $\overline{\text{CY}}$

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.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.9 Memory bit manipulation instructions

◯ SET1 mem.bit

Function: (mem.bit) ← 1 mem = D_{7-0}: 00H to 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 to 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 to 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ 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 to 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 clears the bit to “0”.

Download from Www.Somanuals.com. All Manuals Search And Download.

○ AND1 CY, fmem.bit

○ AND1 CY, pmem.@L

○ AND1 CY, @H+mem.bit

Function: CY $\leftarrow$ CY $\wedge$ (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 $\leftarrow$ CY $\vee$ (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 $\leftarrow$ CY $\veebar$ (bit specified by operand)

Exclusive-ORs the contents 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.10 Branch instructions

(I) BR addr

Function: $PC_{11-0} \leftarrow$ addr
addr = 0000H to 0FFFH

Branches to an address specified by immediate data addr.

This instruction is an assembler directive 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: $PC_{11-0} \leftarrow$ addr1
addr1 = 0000H to 0FFFH

Branches to an address specified by immediate data addr1.

This instruction is an assembler directive and is replaced by the assembler at assembly time with the optimum instruction from the BRA !addr1, BR !addr, BRCB !caddr, and BR $addr instructions.

(II) BRA !addr1

Function: $PC_{11-0} \leftarrow$ addr1

BR !addr

Function: $PC_{11-0} \leftarrow$ addr
addr = 0000H to 0FFFH

Transfers immediate data addr to the program counter (PC) and branches to an address specified by the PC.

BR $addr

Function: $PC_{11-0} \leftarrow$ 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

(II) BR $addr1

Function: $PC_{11-0} \leftarrow$ addr1
addr1 = (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.

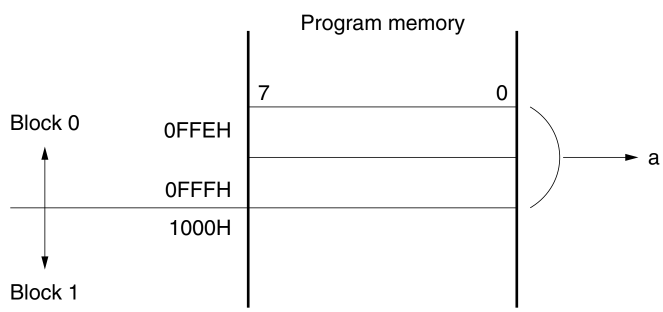
◯ BRCB !caddr

Function: $PC_{11-0} \leftarrow caddr_{11-0}$
caddr = 0000H to 0FFFH

Branches to an address specified by the program counter (PC_{11-0}) replaced with 12-bit immediate data caddr.

Caution

The BRCB !caddr instruction usually branches execution in a block where the instruction exists. If the first byte of this instruction is at address 0FFEH, however, execution does not branch to block 0 but to block 1.



If the BRCB !caddr instruction is at position a in the figure above, execution branches to block 1 (unmounted), not block 0.

Do not use the BRC !caddr instruction at the address 0FFEH.

Download from Www.Somanuals.com. All Manuals Search And Download.

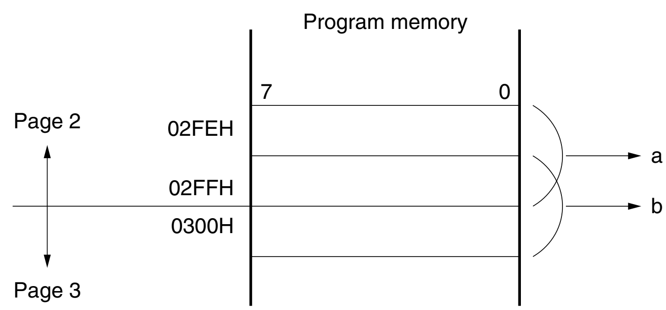
◯ BR PCDE

Function: $PC_{11\text{-}0} \leftarrow PC_{11\text{-}8} + DE$
$PC_{7\text{-}4} \leftarrow D$, $PC_{3\text{-}0} \leftarrow E$

Branches to an address specified by the lower 8 bits of the program counter ($PC_{7\text{-}0}$) 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 to the page where the instruction exists. If the first byte of the op code is at address ××FE 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 lower 8-bit address specified by the contents of register pair DE in page 3, not in page 2.

◯ BR PCXA

Function: $PC_{11\text{-}0} \leftarrow PC_{11\text{-}8} + XA$
$PC_{7\text{-}4} \leftarrow X$, $PC_{3\text{-}0} \leftarrow A$

Branches to an address specified by the lower 8 bits of the program counter ($PC_{7\text{-}0}$) 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

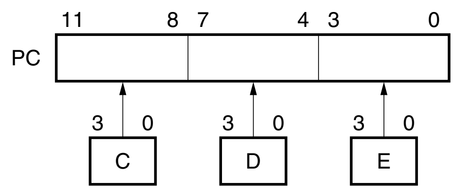
◯ BR BCDE

Function: PC_{11-0} ← BCDE

Example

To branch to an address specified by the contents of the program counter replaced by the contents of registers B, C, D, and E

However, the PC of the μPD754244 is 12 bits. The contents of PC are replaced by the contents of registers C, D and E. Always set register B to 0000B.



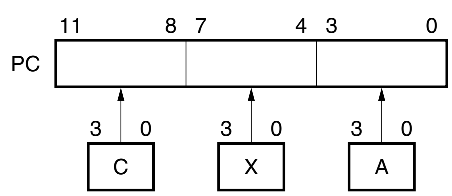
◯ BR BCXA

Function: PC_{11-0} ← BCXA

Example

To branch to an address specified by the contents of the program counter replaced by the contents of registers B, C, X, and A

However, the PC of the μPD754244 is 12 bits. The contents of PC are replaced by the contents of registers C, X and A. Always set register B to 0000B.



◯ TBR addr

Function:

This is an assembler directive for table definition by the GETI instruction. It is used to replace a 3-byte BR !addr instruction with a 1-byte GETI instruction. Describe 12-bit address data as addr. For details, refer to **RA75X Assembler Package Language User's Manual**.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.11 Subroutine/stack control instructions

(II) CALLA !addr1

Function: (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_{11-0} ← addr1, SP ← SP – 6

(I/II) CALL !addr

Function: [MkI 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 – 4

addr = 0000H to 0FFFH

[MkII 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_{11-0} ← addr, SP ← SP–6

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 an address specified by 12-bit immediate data addr.

Download from Www.Somanuals.com. All Manuals Search And Download.

I/II CALLF !faddr

Function: [MkI 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_{11-0} ← 0+faddr

faddr = 0000H to 07FFH

[MkII 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} ← 0+faddr

faddr = 0000H to 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 an 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 directive for table definition by the GETI instruction. It is used to replace a 3-byte CALL !addr instruction with a 1-byte GETI instruction. Describe 12-bit address data as addr. For details, refer to **RA75X Assembler Package Language User's Manual**.

Download from Www.Somanuals.com. All Manuals Search And Download.

(I/II) RET

Function: [MkI 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
[MkII mode] $PC_{11-8} \leftarrow$ (SP), 0, 0, 0, 0 $\leftarrow$ (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

All the flags of the program status word (PSW) other than MBE and RBE are not restored.

(I/II) RETS

Function: [MkI 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
[MkII mode] $PC_{11-8} \leftarrow$ (SP), 0, 0, 0, 0 $\leftarrow$ (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

All the flags of the program status word (PSW) other than MBE and RBE are not restored.

(I/II) RETI

Function: [MkI 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

[MkII mode] $PC_{11-8} \leftarrow$ (SP), 0, 0, 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

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.

Download from Www.Somanuals.com. All Manuals Search And Download.

◯ PUSH rp

Function: (SP–1) ←rp_H, (SP–2) ← rp_L, SP ← 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 higher 4 bits of the register pair (rp_H, X, H, D, or B) are saved to the stack addressed by (SP–1), and the lower 4 bits (rp_L: A, L, E, or C) are saved to the stack addressed by (SP–2).

◯ 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.

◯ 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 decrements the contents of the stack pointer.

The contents of (SP) are restored to the higher 4 bits of the register pair (rp_H, X, H, D, or B), and the contents of (SP+1) are restored to the lower 4 bits (rp_L: A, L, E, or C).

◯ 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.

Download from Www.Somanuals.com. All Manuals Search And Download.

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, T0, T1, T2, 0, 2, or EE).

◯ 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, T0, T1, T2, 0, 2, or EE).

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.13 Input/output instructions

◯ IN A, PORTn

Function: A ← PORTn n = $N_{3\text{-}0}$: 3, 6, 7, 8

Transfers the contents of a port specified by PORTn (n = 3, 6, 7, 8) to the A register.

Caution

When this instruction is executed, it is necessary that MBE = 0 or (MBE = 1, MBS = 15). n can be 3, 6, 7, 8.

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.

◯ OUT PORTn, A

Function: PORTn ← A n = $N_{3\text{-}0}$: 3, 6, 8

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 3, 6, 8).

Caution

When this instruction is executed, it is necessary that MBE = 0 or (MBE = 1, MBS = 15).

Only 3, 6, and 8 can be specified as n.

Download from Www.Somanuals.com. All Manuals Search And Download.

11.4.14 CPU control instruction

◯ HALT

Function: PCC.2 ← 1

Sets the HALT mode (this instruction sets the 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 the bit 3 of the processor clock control register).

Caution
Make sure that a NOP instruction follows the STOP instruction.

◯ NOP

Function: Executes nothing but consumes 1 machine cycle.

Download from Www.Somanuals.com. All Manuals Search And Download.

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, 4, 15

Transfers 4-bit immediate data n to the memory bank select register (MBS).

(I/II) GETI taddr

Function: taddr = T_{5-0}, 0: 20H to 7FH

[MkI mode]

- **When table defined by TBR instruction is referenced**
  PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1)
- **When table defined by TCALL instruction is referenced**
  (SP–1) ← PC_{7-4}, (SP–2) ← PC_{3-0}
  (SP–3) ← MBE, RBE, 0, 0
  (SP–4) ← PC_{11-8}
  PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1)
  SP ← SP–4
- **When table defined by instruction other than TBR and TCALL is referenced**
  Executes instruction with (taddr) (taddr+1) as op code

[MkII mode]

- **When table defined by TBR instruction is referenced**[Note]
  PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1)
- **When table defined by TCALL instruction is referenced**[Note]
  (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_{11-0} ← $(taddr)_{3-0}$ + (taddr+1), SP ← SP–6
- **When table defined by instruction other than TBR and TCALL is referenced**
  Executes instruction with (taddr) (taddr+1) as op code

Note The address specified by the TBR and TCALL instructions is limited to 0000H to 0FFFH.

Download from Www.Somanuals.com. All Manuals Search And Download.

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 to 007FH. Data must be written to this area in advance. Write the mnemonic of a 1-byte or 2-byte instruction as the data as is.

When a 3-byte call instruction and 3-byte branch instruction is used, data is written by using an assembler directive (TCALL or TBR).

Only an even address can be specified by taddr.

Caution

Only a 2-machine-cycle instruction can be set to the reference table as a 2-byte instruction (except the BRCB and CALLF instructions). Two 1-byte instructions can be set 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 reference instruction has been executed, processing continues from the address next to 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 the 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 of 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 of the same group as the referenced instruction, the string effect by the referenced instruction is valid, and the instruction following that instruction is skipped.

Download from Www.Somanuals.com. All Manuals Search And Download.

Application example

```
 ┌ MOV   HL, #00H  ┐
 │ MOV   XA, #FFH  │  Replaced by GETI
 │ CALL  SUB1      │
 └ BR    SUB2      ┘

         ORG     20H
HL00:    MOV     HL, #00H
XAFF:    MOV     XA, #FFH
CSUB1:   TCALL   SUB1
BSUB2:   TBR     SUB2
           :
         GETI    HL00        ; MOV HL, #00H
           :
         GETI    BSUB2       ; BR SUB2
           :
         GETI    CSUB1       ; CALL SUB1
           :
         GETI    XAFF        ; MOV XA, #FFH
```


Download from Www.Somanuals.com. All Manuals Search And Download.

APPENDIX A DEVELOPMENT TOOLS

The following development tools are available to support development of systems using the μPD754244. With the 75XL Series, a relocatable assembler that can be used in common with any model in the series is used in combination with a device file dedicated to the model being used.

Language Processor

RA75X relocatable assembler	Host machine	OS	Supply medium	Order code
	PC-9800 series	MS-DOS™ (Ver.3.30 ~ Ver.6.2 Note)	3.5"2HD	μS5A13RA75X
			5"2HD	μS5A10RA75X
	IBM PC/AT™ or compatible machine	Refer to **OS of IBM PC**.	3.5" 2HC	μS7B13RA75X
			5"2HC	μS7B10RA75X

Device file	Host machine	OS	Supply medium	Order code
	PC-9800 series	MS-DOS (Ver.3.30 ~ Ver.6.2 Note)	3.5"2HD	μS5A13DF754244
			5"2HD	μS5A10DF754244
	IBM PC/AT or compatible machine	Refer to **OS of IBM PC**.	3.5" 2HC	μS7B13DF754244
			5"2HC	μS7B10DF754244

Note Although Ver.5.00 or above has a task swap function, this function cannot be used with this software.

Remark The operations of the assembler and device file are guaranteed only on the above host machines and OSs.

Download from Www.Somanuals.com. All Manuals Search And Download.

Debugging Tools

In-circuit emulators (IE-75000-R and IE-75001-R) are available as the debugging tools for the μPD754244. The following table shows the system configuration of the in-circuit emulators.

<table>
<tr><td rowspan="5">Hardware</td><td colspan="2">IE-75000-R^{Note1}</td><td colspan="4">The IE-75000-R is an in-circuit emulator that debugs the hardware and software of an application system using the 75X Series or 75XL Series. To develop the μPD754244, use this in-circuit emulator with an emulation board IE-75300-R-EM and emulation probe EP-754144GS-R (both sold separately).
The in-circuit emulator is connected to a host machine for efficient debugging.
The IE-75000-R contains the emulation board IE-75000-R-EM.</td></tr>
<tr><td colspan="2">IE-75001-R</td><td colspan="4">The IE-75001-R is an in-circuit emulator that debugs the hardware and software of an application system using the 75X Series or 75XL Series. To develop the μPD754244, use this in-circuit emulator with an emulation board IE-75300-R-EM and emulation probe EP-754144GS-R (both sold separately).
The in-circuit emulator is connected to a host machine to provide efficient debugging.</td></tr>
<tr><td colspan="2">IE-75300-R-EM</td><td colspan="4">This is an emulation board to evaluate an application system using the μPD754244. It is used with the IE-75000-R or IE-75001-R.</td></tr>
<tr><td colspan="2">EP-754144GS-R</td><td colspan="4" rowspan="2">This is an emulation probe for the μPD754244GS.
It is connected to the IE-75000-R or IE-75001-R and IE-75300-R-EM.
Flexible board EV-9500GS-20 (for 20-pin plastic SSOP) and EV-9501GS-20 (for 20-pin plastic SOP) facilitating connection with target board are supplied.</td></tr>
<tr><td></td><td>EV-9500GS-20
EV-9501GS-20</td></tr>
<tr><td rowspan="7">Software</td><td colspan="2" rowspan="7">IE control program</td><td colspan="4">This program connects the IE-75000-R or IE-75001-R and a host machine with an RS-232C or Centronics interface to control the IE-75000-R or IE-75001-R on the host machine.</td></tr>
<tr><td colspan="3">Host machine</td><td rowspan="2">Order code</td></tr>
<tr><td></td><td>OS</td><td>Supply medium</td></tr>
<tr><td rowspan="2">PC-9800 series</td><td rowspan="2">MS-DOS
(Ver.3.30
~
Ver.6.2^{Note2})</td><td>3.5"2HD</td><td>μS5A13IE75X</td></tr>
<tr><td>5"2HD</td><td>μS5A10IE75X</td></tr>
<tr><td rowspan="2">IBM PC/AT or compatible machine</td><td rowspan="2">Refer to OS of IBM PC.</td><td>3.5" 2HC</td><td>μS7B13IE75X</td></tr>
<tr><td>5"2HC</td><td>μS7B10IE75X</td></tr>
</table>

Notes 1. This is a maintenance part.

2. Although Ver.5.00 or above has a task swap function, this function cannot be used with this software.

Remark The operation of the IE control program is guaranteed only on the above host machines and OSs.

Download from Www.Somanuals.com. All Manuals Search And Download.

OS of IBM PC

The following OSs are supported as the OS for IBM PCs.

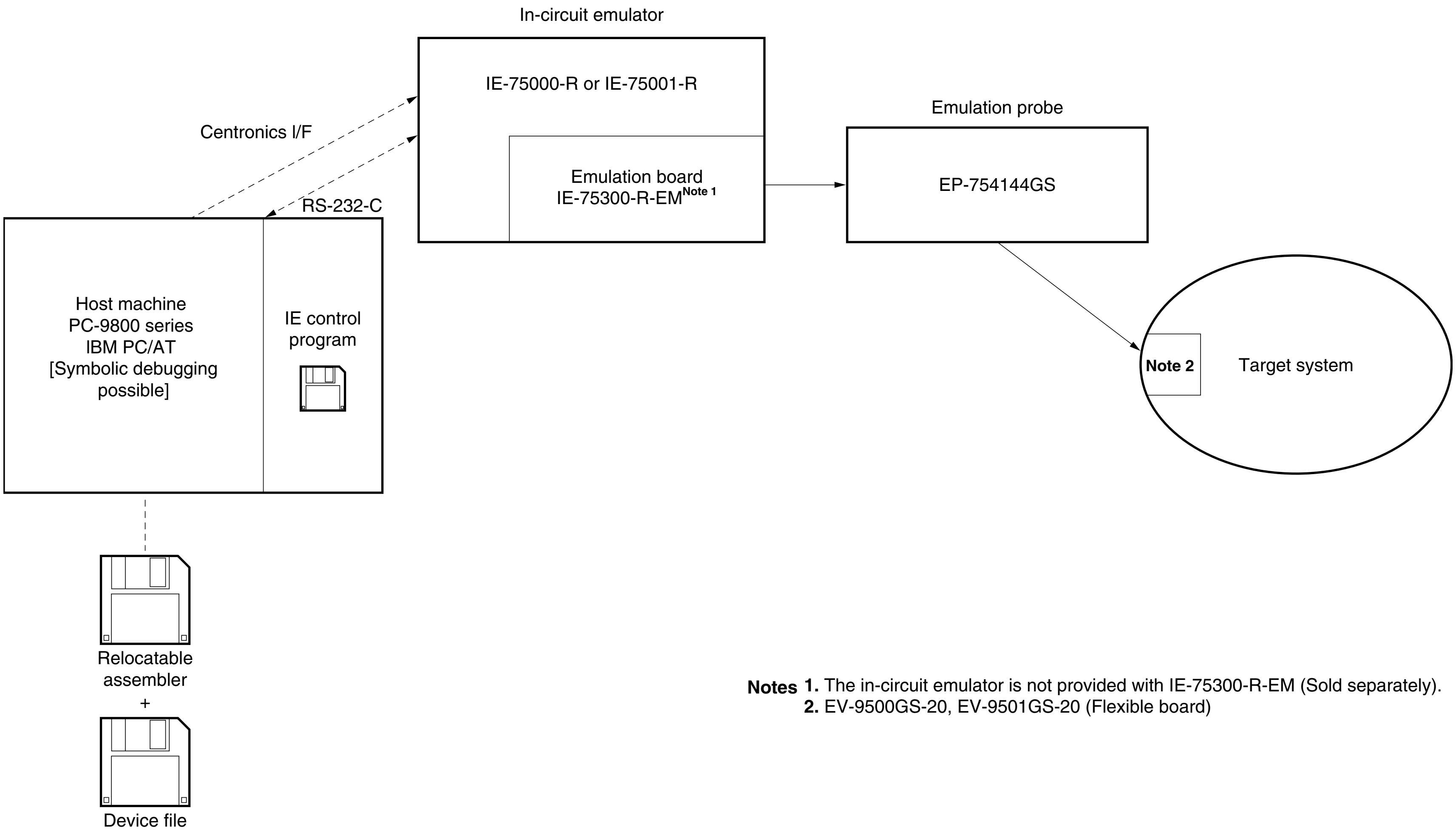
OS	Version
PC DOS™	Ver.5.02 to Ver.6.3 J6.1/V Note to J6.3/V Note
MS-DOS	Ver.5.0 to Ver.6.22 5.0/V Note to 6.2/V Note
IBM DOS™	J5.02/V Note

Note Only the English mode is supported.

Caution Although Ver.5.00 or above has a task swap function, this function cannot be used with this software.

Download from Www.Somanuals.com. All Manuals Search And Download.

★ Development Tool Configuration



Notes 1. The in-circuit emulator is not provided with IE-75300-R-EM (Sold separately).

2. EV-9500GS-20, EV-9501GS-20 (Flexible board)

Download from Www.Somanuals.com. All Manuals Search And Download.

APPENDIX B ORDERING MASK ROM

After your program has been developed, you can place an order for mask ROM using the following procedure.

<1> Reservation for mask ROM ordering

Inform NEC Electronics of when you intend to place an order for the mask ROM. (NEC's response may be delayed if we are not informed in advance.)

<2> Preparation of ordering media

The following 3 media are available for ordering mask ROM:

- UV-EPROM**Note**
- 3"5 IBM-format floppy disk (outside Japan)

Note Prepare three UV-EPROMs with the same contents. For products with mask options, 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:

- Mask ROM Ordering Sheet
- Mask ROM Ordering Check Sheet
- Mask Option Information Sheet (necessary for products with mask options)

<4> Ordering

Submit the media prepared in <2> and documents prepared in <3> to NEC by the order reservation date.

Download from Www.Somanuals.com. All Manuals Search And Download.

APPENDIX C INSTRUCTION INDEX

C.1 Instruction Index (By Function)

Download from Www.Somanuals.com. All Manuals Search And Download.

[Accumulator instruction]

[Increment/decrement instruction]

[Compare instruction]

[Carry flag manipulation instruction]

[Memory bit manipulation instruction]

[Branch instruction]

Download from Www.Somanuals.com. All Manuals Search And Download.

[Subroutine/stack control instruction]

[Interrupt control instruction]

[Input/output instruction]

[CPU control instruction]

[Special instruction]

Download from Www.Somanuals.com. All Manuals Search And Download.

C.2 Instruction Index (Alphabetical Order)

Download from Www.Somanuals.com. All Manuals Search And Download.

Download from Www.Somanuals.com. All Manuals Search And Download.

Download from Www.Somanuals.com. All Manuals Search And Download.

APPENDIX D HARDWARE INDEX

Download from Www.Somanuals.com. All Manuals Search And Download.

[S]

[T]

[W]

Download from Www.Somanuals.com. All Manuals Search And Download.

APPENDIX E REVISION HISTORY

The revision history is shown below. “Location” indicates the corresponding chapters in the preceding edition.

Edition	Description	Location
2nd edition	Change of representative model from μPD754144 to μPD754244	Throughout
	Change of EEPROM write time and number of write operations	**CHAPTER 5 EEPROM**
	Addition of **Note** when development tool is used	
	Addition of list of functions of μPD754144, 754244, and 75F4264	**APPENDIX A LIST OF FUNCTIONS OF μPD754144, 754244, AND 75F4264**
	Change of device file name	**APPENDIX B DEVELOPMENT TOOLS**
	Upgrading of version of OS supported by development tools	
	Change of media for ordering mask ROM	**APPENDIX C ORDERING MASK ROM**
3rd edition	Correction of description in figure in **7.9 Application of Interrupt (6) Executing pending interrupt - interrupt occurs during interrupt service (INTBT has higher priority and INTT0 and INTT2 have lower priority)**	**CHAPTER 7 INTERRUPT AND TEST FUNCTIONS**
	Correction of instruction code of “BR BCDE”	**CHAPTER 11 INSTRUCTION SET**
	Deletion of flash-related products in configuration diagram	**APPENDIX A DEVELOPMENT TOOLS**
	Deletion of **APPENDIX A LIST OF FUNCTIONS OF μPD754144, 754244, AND 75F4264**	–

Download from Www.Somanuals.com. All Manuals Search And Download.

Free Manuals Download Website
http://myh66.com
http://usermanuals.us
http://www.somanuals.com
http://www.4manuals.cc
http://www.manual-lib.com
http://www.404manual.com
http://www.luxmanual.com
http://aubethermostatmanual.com
Golf course search by state
http://golfingnear.com
Email search by domain
http://emailbydomain.com
Auto manuals search
http://auto.somanuals.com
TV manuals search
http://tv.somanuals.com