To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

RENESAS

User's Manual

μPD753304

4-Bit Single-chip Microcontroller

μPD753304

Document No. U12020EJ2V0UM00 (2nd edition)
Date Published October 1997 N

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

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

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

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

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

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated “quality assurance program“ for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

- Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
- Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
- Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

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

Anti-radioactive design is not implemented in this product.

M7 96.5

Regional Information

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

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

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

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

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

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

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

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

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

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

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

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

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

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

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Cumbica-Guarulhos-SP, Brasil
Tel: 011-6465-6810
Fax: 011-6465-6829

J97. 8

Major Revisions in This Edition

Page	Contents
Throughout	Status of μPD753304 is changed from “under development” to “developed”.
	Method of supplying mass-produced devices changed to pellet/wafer.
	Main system clock is changed from 4.0 MHz to 3.6 MHz (Typ.) and subsystem clock is changed from 32.768 MHz to 47 kHz (Typ.).
p.27	**Table 2-2 Status at Reset by Port 3 Mask Option** is added.
p.139	**5.6.8 Supply of LCD drive power supply V_{LC0}, V_{LC1}, and V_{LC2}** Mask option of LCD display mode is added. LCD drive split resistor 10kΩ (typ.) is deleted.
p.140	**Figure 5-39 LCD Drive Power Connection Example** is changed
p.177	**Chapter 7 STANDBY FUNCTION** Interrupt request signal is added to STOP mode of release signal of subsystem clock.
p.275	**APPENDIX F REVISION HISTORY** is added.

The mark ★shows major revised points.

INTRODUCTION

Readers: This manual is intended for user engineers who understand the functions of the μPD753304 4-bit single-chip microcontroller, and wish to design application systems using this microcontroller.

Purpose: This manual describes the hardware functions of the μPD753304 in the organization described below.

Organization: This manual contains the following information:

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

How to Read This Manual:

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

- If you have experience of using the μPD75308B and 753108,
  → Read **APPENDIX A μPD75308B, 753108, AND 753304 FUNCTIONAL LIST** to check differences between the μPD75308B and the microcontroller described in this manual.

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

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

- To understand the overall functions of the μPD753304,
  → Read this manual in the order of Contents.

Legend

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

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

Documents related to devices

Document name	Document number	
	Japanese	English
μPD753304 Data Sheet	U11874J	U11874E
μPD753304 User's Manual	U12020J	U12020E (This manual)
75XL Series Selection Guide	U10453J	U10453E

Documents related to development tools

Document name			Document number	
			Japanese	English
Hardware	IE-75000-R/IE-75001-R User's Manual		EEU-846	EEU-1416
	IE-75300-R-EM User's Manual		U11354J	U11354E
	EP-753304DU-R User's Manual		U12173J	U12173E
Software	RA75X Assembler Package	Operation	U12622J	EEU-1346
	User's Manual	Language	U12385J	EEU-1363

Other documents

Document name	Document number	
	Japanese	English
IC Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535J	C10535E
Quality Grades on NEC Semiconductor Devices	C11531J	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983J	C10983E
Electrostatic Discharge (ESD) Test	C11892J	C11892E
Guide to Quality Assurance for Semiconductor Devices	C11893E	MEI-1202
Microcomputer Product Series Guide	U11416J	—

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

[MEMO]

CONTENTS

LIST OF FIGURES (1/3)

LIST OF FIGURES (2/3)

LIST OF FIGURES (3/3)

[MEMO]

LIST OF TABLES (1/2)

LIST OF TABLES (2/2)

CHAPTER 1 GENERAL

The μPD753304 is a 4-bit single-chip microcontroller in the NEC's 75XL Series, a successor to the 75X Series that boasts a wealth of variations. The previous 75X Series that incorporates an LCD controller/driver comes in a 80-pin package, however, the μPD753304 is sold as a pellet/wafer in order to be incorporated in LCD-equipped mobile appliances.

The features of the μPD753304 are as follows:

- ○ On-chip RC oscillator
- ★ • Main system clock : f_{CC} = 3.6 MHz
  (typ. value at external resistor 6.8 kΩ, on-chip 10-pF (typ.) capacitor)
  • Subsystem clock : f_{CT} = 47 kHz (typ.)
  (on-chip resistor and capacitor)
- ○ Immediate processing can be started after standby mode release
- ○ Subsystem clock oscillation can be stopped during STOP mode
- ★ ○ Sold as pellet/wafer for incorporation in LCD-equipped mobile appliances
- ○ Power supply voltage: V_{DD} = 2.5 to 5.5 V
- ○ On-chip programmable LCD controller/driver

Application Fields

- Small-size LCD displays, etc.

1.1 Functional Outline

	Parameter		Function	
★	Instruction execution time		• 1.1, 2.2, 4.4, 17.8 μs (@ 3.6 MHz with main system clock) • 85.1 μs (@ 47 kHz with subsystem clock)	
	On-chip memory	ROM	4096 x 8 bits	
		RAM	256 x 4 bits	
	General-purpose register		• 4-bit operation: 8 x 4 banks • 8-bit operation: 4 x 4 banks	
	Input/output port	CMOS input/output	12 pins	Four have software-specifiable on-chip pull-up resistors, and four can also be used as segment outputs
	LCD controller/driver		• Segment selection: 20/24 segments (can be changed to CMOS input/output port in 4-pin units; max. 4) • Display mode selection: Static, 1/2 duty (1/2 bias), 1/3 duty (1/2 bias), 1/3 duty (1/3 bias), 1/4 duty (1/3 bias)	
★			• LCD display mode can be selected by mask option	
	Timer		3 channels • 8-bit timer counter: 1 channel (subclock source input function provided) • Basic interval timer/watchdog timer: 1 channel • Watch timer: 1 channel	
★	Clock output (PCL)		• Φ, 3.6 MHz, 450 kHz, 225 kHz (@ 3.6 MHz with main system clock)	
★	Buzzer output (BUZ)		• 2.94, 5.88, 47 kHz (@ 47 kHz with subsystem clock) • 1.76, 3.52, 28.13 kHz (@ 3.6 MHz with main system clock)	
	Vectored interrupts		External: 1, Internal: 2	
	Test input		Internal: 1	
	System clock oscillator		• RC oscillator for main system clock oscillation (external resistor, on-chip 10-pF (typ.) capacitor) • RC oscillator for subsystem clock oscillation (on-chip resistor and capacitor)	
	Standby function		STOP/HALT mode	
★	Ambient operating temperature		T_A = –10 to +60°C	
	Power supply voltage		V_{DD} = 2.5 to 5.5 V	
★	Package		• Mass production type: pellet/wafer • ES product (for evaluation): 42-pin ceramic shrink DIP (600 mil)	

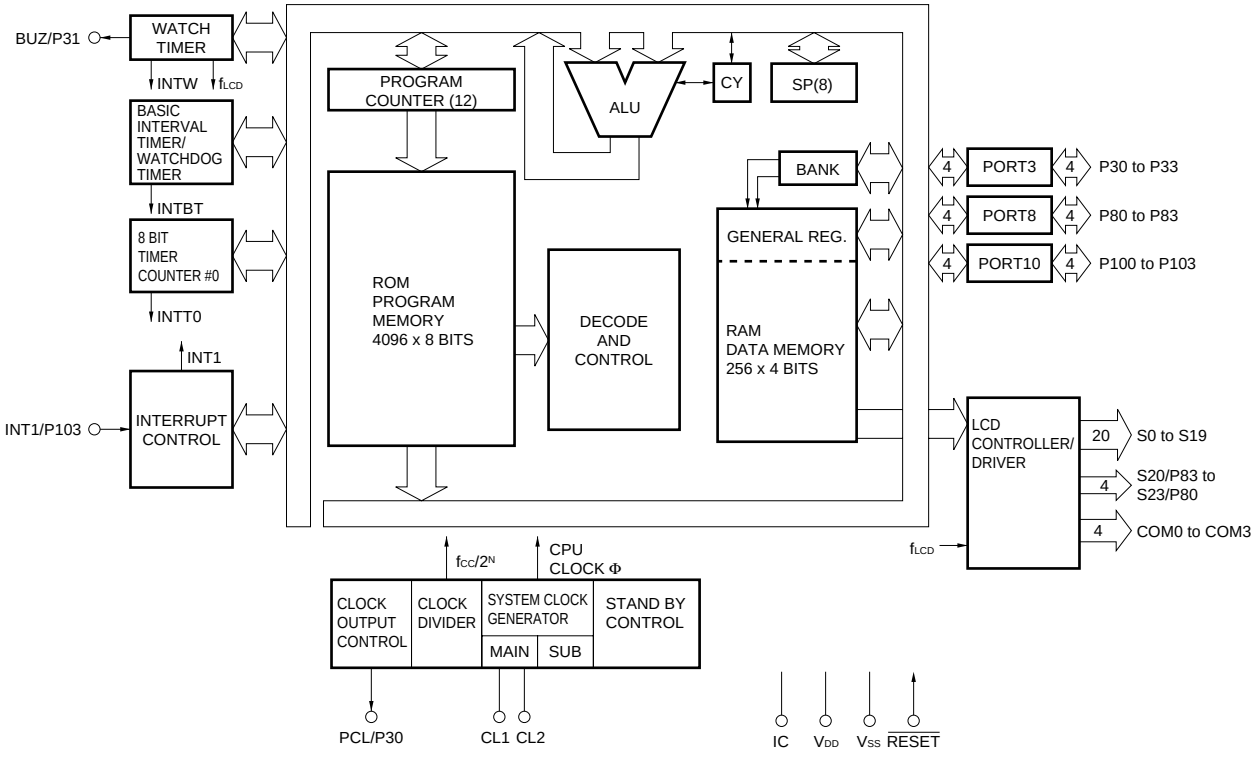
1.2 Ordering Information

Part number	Method of Supplying
★ μPD753304P-xxx	Pellet
★ μPD753304W-xxx	Wafer

Caution The μPD753304 is sold as pellet/wafer. In addition, 42-pin ceramic shrink DIP is available for ES product.

Remark xxx indicates a ROM code suffix.

1.3 Block Diagram



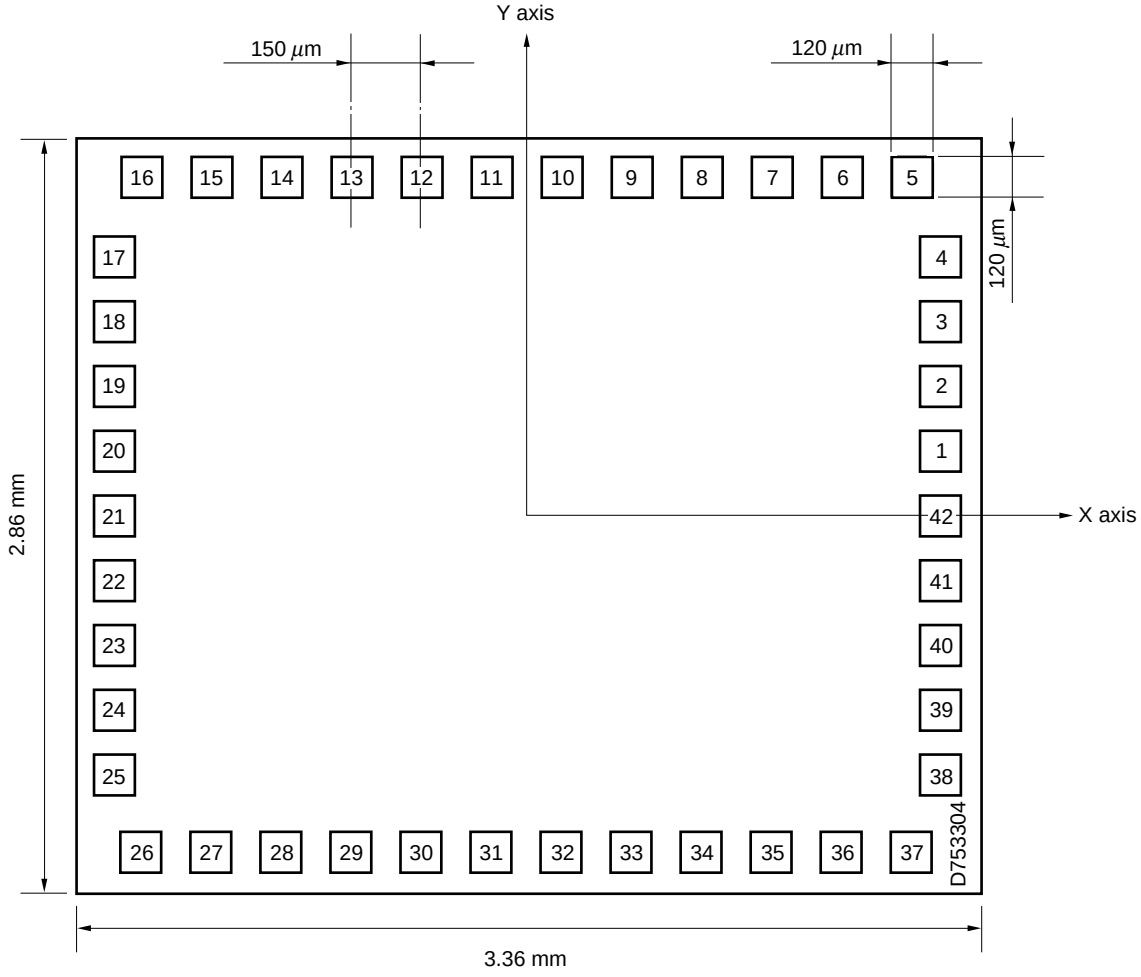
1.4 Pin Configuration

● **Pin configuration of mass production type (pad layout)**

• Pellet

μPD753304P-xxx

Chip size : 3.36 x 2.86 mm^2
Pad distance : 150 μm
Pad size : 120 x 120 μm



Pad coordinates (unit : μm : pad center coordinates)

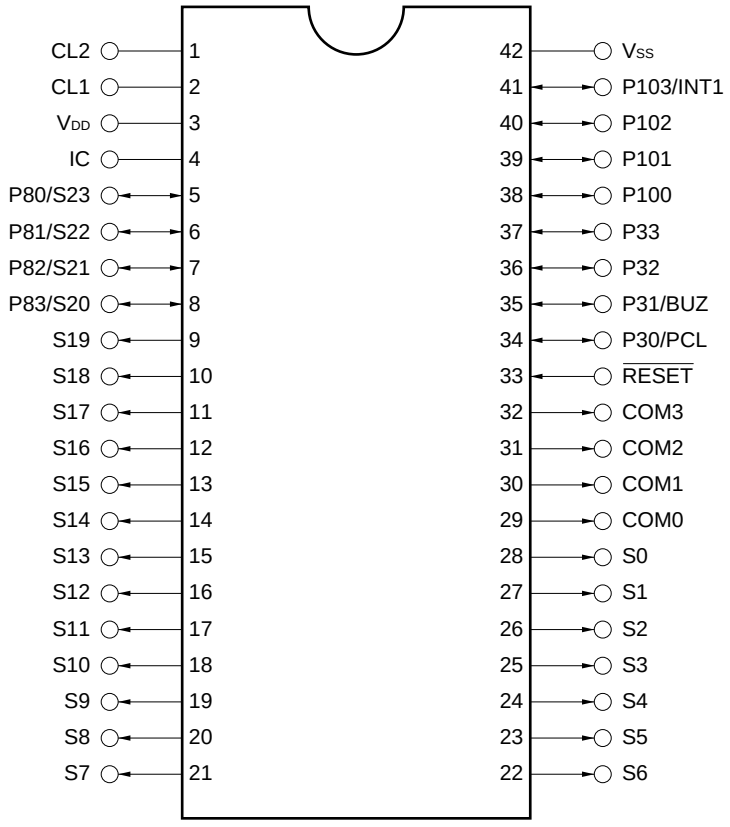
No.	Pin name	X axis	Y axis
1	CL2	1549	311
2	CL1	1549	540
3	V_{DD}	1549	769
4	IC	1549	998
5	P80/S23	1422.5	1299
6	P81/S22	1169.5	1299
7	P82/S21	916.5	1299
8	P83/S20	663.5	1299
9	S19	410.5	1299
10	S18	157.5	1299
11	S17	–216.5	1299
12	S16	–469.5	1299
13	S15	–715.5	1299
14	S14	–961.5	1299
15	S13	–1207.5	1299
16	S12	–1453.5	1299
17	S11	–1549	992.5
18	S10	–1549	746.5
19	S9	–1549	500.5
20	S8	–1549	254.5
21	S7	–1549	–105.5

No.	Pin name	X axis	Y axis
22	S6	–1549	–351.5
23	S5	–1549	–597.5
24	S4	–1549	–843.5
25	S3	–1549	–1089.5
26	S2	–1301	–1299
27	S1	–1055	–1299
28	S0	–809	–1299
29	COM0	–563	–1299
30	COM1	–317	–1299
31	COM2	–71	–1299
32	COM3	289	–1299
33	$\overline{\text{RESET}}$	518	–1299
34	P30/PCL	747	–1299
35	P31/BUZ	976	–1299
36	P32	1205	–1299
37	P33	1434	–1299
38	P100	1549	–997
39	P101	1549	–768
40	P102	1549	–539
41	P103/INT1	1549	–310
42	V_{SS}	1549	0.5

Caution Connect pellet back side to GND.

● **Pin configuration of ES product (Top View)**

• 42-pin ceramic shrink DIP (600 mil)



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

Caution The μPD753304 is sold as pellet/wafer. The above pin configuration applies to ES product.

Pin Identification

BUZ	: Buzzer Clock	P100 to P103	: Port 10
CL1, CL2	: RC Oscillator	PCL	: Programmable Clock
COM0 to COM3	: Common Output 0 to 3	$\overline{\text{RESET}}$	: Reset
IC	: Internally Connected	S0 to S23	: Segment Output 0 to 23
INT1	: External Vectored Interrupt 1	V_{DD}	: Positive Power Supply
P30 to P33	: Port 3	V_{SS}	: Ground
P80 to P83	: Port 8		

CHAPTER 2 PIN FUNCTION

2.1 Pin Functions of μPD753304

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

Pin name	Input/output	Alternate function	Function	8-bit I/O	After reset	I/O circuit type[Note 1]
★ P30	Input/output	PCL	Programmable 4-bit input/output port (PORT3). This port can be specified input/output bit-wise. Input/output mode can be specified at reset for this port (Mask option [Note 2]).	No	Input[Note 2]	E
P31		BUZ				
P32		—				
P33		—				
P80	Input/output	S23	4-bit input/output port (PORT8).	No	Input	H
P81		S22				
P82		S21				
P83		S20				
P100	Input/output	—	Programmable 4-bit input/output port (PORT10). Input/output can be specified bit-wise for this port. On-chip pull-up resistor connection can be specified by software in 4-bit units.	No	Input with pull-up resistor	E-B
P101		—				
P102		—				
P103		INT1				<F>-A

Notes 1. Circuit types enclosed in brackets indicate Schmitt trigger input.

★ **2.** Input/Output mode after reset can be specified with mask option. Refer to Table 2-2 for details.

★ **Table 2-2. Status at Reset by Port 3 Mask Option**

Pin name	Status after reset		
	Mask option <1>	Mask option <2>	Mask option <3>
P30/PCL	Input	Low-level output	Low-level output
P31/BUZ			
P32			
P33			High-level output

Table 2-3. Pin Function of Non-port Pins

	Pin name	Input/output	Alternate function	Function		After reset	I/O circuit type[Note 1]
★	PCL	Output	P30	Clock output		Input[Note 2]	E
	BUZ		P31	Optional frequency output (for buzzer or system clock trimming)			
	INT1	Input	P103	Edge detection vectored interrupt input (detection edge selection enable)	Asynchronous	Input with pull-up resistor	<F>-A
★	S0 to S19	Output	—	Segment signal output		High impedance	G-B
	S20 to S23	Output	P83 to P80	Segment signal output		Input	H
★	COM0 to COM3	Output	—	Common signal output		High impedance	G-B
	CL1	—	—	Resistor (R) connection pin for main system clock oscillation. External clock cannot be input.		—	—
	CL2	—					
	$\overline{\text{RESET}}$	Input	—	System reset input (low-level active). Pull-up resistor can be incorporated (mask option).		—	<B>-A
	IC	—	—	Internally Connected. Connect to V_{DD} directly.		—	—
	V_{DD}	—	—	Positive power supply		—	—
	V_{SS}	—	—	Ground potential		—	—

Notes 1. Circuit types enclosed in brackets indicate Schmitt trigger input.

★ **2.** Input/Output mode after reset can be specified with mask option. Refer to Table 2-2 for details.

2.2 Pin Functions

2.2.1 P30 to P33 (PORT3) ... input/output shared with PCL and BUZ
P80 to P83 (PORT8) ... input/output shared with S23 to S20
P100 to P103 (PORT10) ... input/output shared with INT1

These pins are 4-bit input/output port pins with output latch. In addition to the input/output port functions, the following functions are provided.

- Port 3 : Clock output (PCL)
  Optional frequency output (BUZ)
- Port 8 : Segment signal output (S23 to S20)
- Port 10 : Vectored interrupt input (INT1)

Input/output mode selection of ports are set by a port mode register. Ports 3 and 10 are specified bit-wise, and port 8 is specified in 4-bit units.

In addition, on-chip pull-up resistor connection can be specified for port 10 by software in 4-bit units. The specification is performed by manipulating pull-up resistor specification register group B (POGB).

$\overline{\text{RESET}}$ signal generation sets the ports as follows.

★ Port 3 (P30 to P33) : Input/Output mode after reset can be specified with mask option.
Refer to Table 2-2 for details.

Port 10 (P100 to P103) : Input mode (with pull-up resistor)

2.2.2 PCL ⋯ output shared with port 3

This is a programmable clock output pin. This pin is used to supply the clock to a peripheral LSI (such as a slave microcontroller), and is shared with the P30 pin. When the $\overline{\text{RESET}}$ signal is generated, the contents of the clock output mode register (CLOM) are cleared to “0”, disabling the output of the clock. In this case, the PCL pin can be used as an ordinary port pin.

Refer to **5.2.4 Clock output circuit** for details.

2.2.3 BUZ ⋯ output shared with port 3

This is a frequency output pin used to issue a buzzer sound or trim the system clock oscillation frequency by outputting a specified frequency (2.94, 5.88, or 47 kHz: @ 47 kHz with subsystem clock). This pin is shared with the P31 pin, and is valid only when bit 7 (WM7) of the watch mode register (WM) is set to “1”.

When the $\overline{\text{RESET}}$ signal is generated, WM7 is cleared to “0”, so that the BUZ pin is used as an ordinary port pin.

Refer to **5.4.2 Watch mode register** for details.

2.2.4 INT1 ··· input shared with port 10

This pin is an edge detection vectored interrupt input pin. The edge to be detected can be selected by using the edge detection mode register (IM1).

- **INT1 (bit 0 of IM1)**
  (a) Active at rising edge
  (b) Active at falling edge

INT1 is an asynchronous input pin. The signal input to this pin is acknowledged as long as the signal has a specific high-level width, regardless of the operating clock of the CPU. This pin is shared with the P103 pin.

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

★ INT1 can be used to release the STOP mode and HALT mode.

INT1 is Schmitt trigger input pin.

2.2.5 S0 to S23 ··· outputs

These are segment signal output pins that can directly drive the segment pins (front panel electrodes) of an LCD and perform static and 2- or 3-time division drive of the 1/2 bias method or 3- or 4-time division drive of the 1/3 bias method.

S0 through S23 are shared with port 8, and the modes of these pins can be selected by using display mode register (LCDM).

2.2.6 COM0 to COM3 ··· outputs

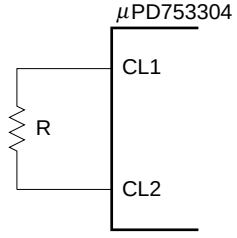
These are common signal output pins that can directly drive the common pins (rear panel electrodes) of an LCD. They output common signals at static (COM0, 1, 2, and 3 outputs), 2-time division drive of the 1/2 bias method (COM0 and 1 outputs) or 3-time division drive (COM0, 1, and 2 outputs), or 3-time division drive of the 1/3 bias method (COM0, 1, and 2 outputs) or 4-time division drive (COM0, 1, 2, and 3 outputs).

2.2.7 CL1, CL2

These pins connect a resistor (R) for main system clock oscillation. These pins incorporate 10-pF (typ.) capacitor (C).

An external clock cannot be input to these pins.

RC oscillation



2.2.8 $\overline{\text{RESET}}$

This pin inputs a low-active reset signal.

The $\overline{\text{RESET}}$ signal is an asynchronous input signal and is generated 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 cannot only be used to initialize and start the CPU, but also to release the STOP and HALT modes.

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

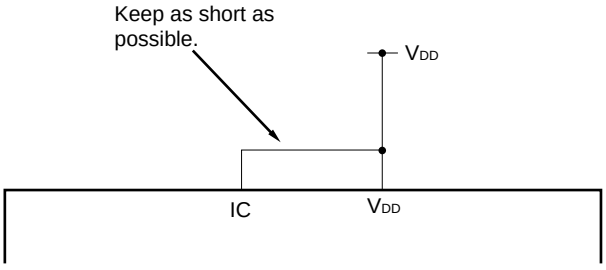
This pin can specify 60-kΩ (typ.) on-chip pull-up resistor by using mask option.

2.2.9 IC

The IC (Internally Connected) pin sets a test mode in which the μPD753304 is tested before shipment. Usually connect the IC pin directly to the V_{DD} pin with as short a wiring length as possible.

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

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



2.2.10 V_{DD}

Positive power supply pin.

2.2.11 V_{SS}

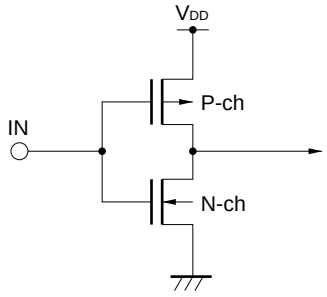
GND potential.

2.3 Pin Input/Output Circuits

The μPD753304 pin input/output circuits are shown schematically.

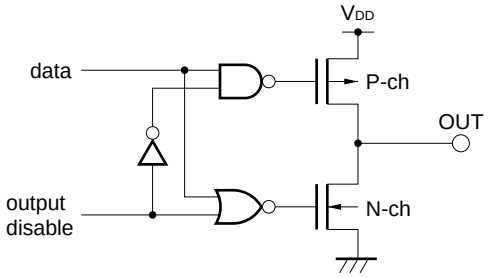
(1/2)

TYPE A



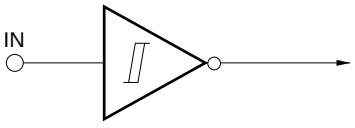
CMOS standard input buffer

TYPE D



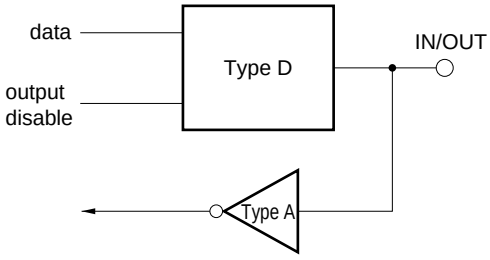
Push-pull output that can be placed in output high-impedance (both P-ch and N-ch off).

TYPE B

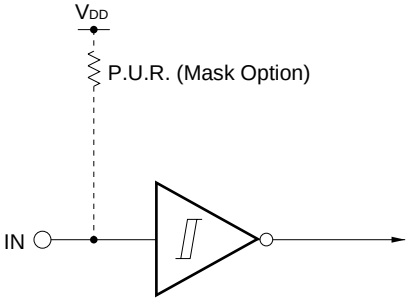


Schmitt trigger input with hysteresis characteristics

TYPE E



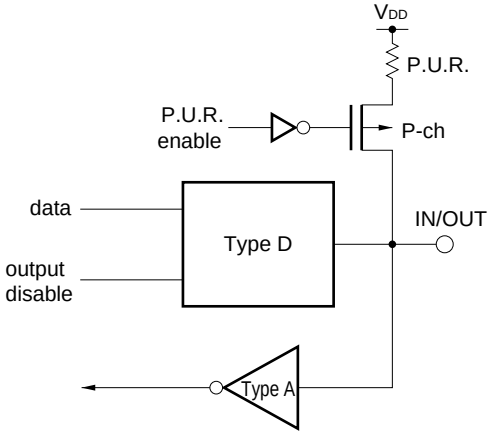
TYPE B-A



P.U.R. : Pull-Up Resistor

Schmitt trigger input with hysteresis characteristics

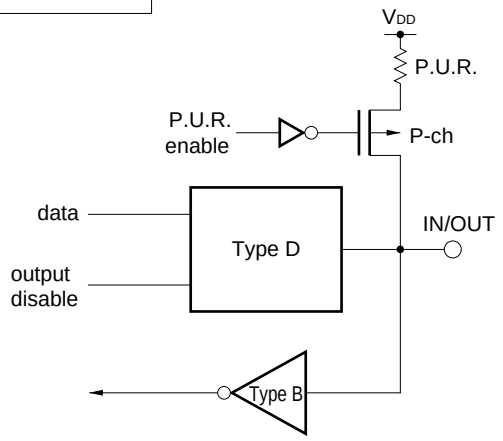
TYPE E-B



P.U.R. : Pull-Up Resistor

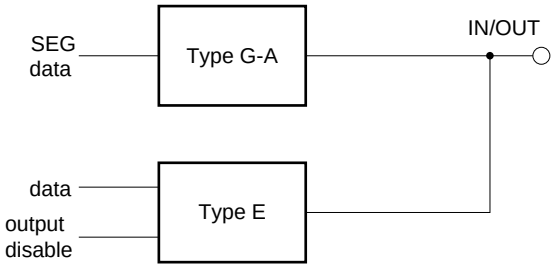
(2/2)

TYPE F-A

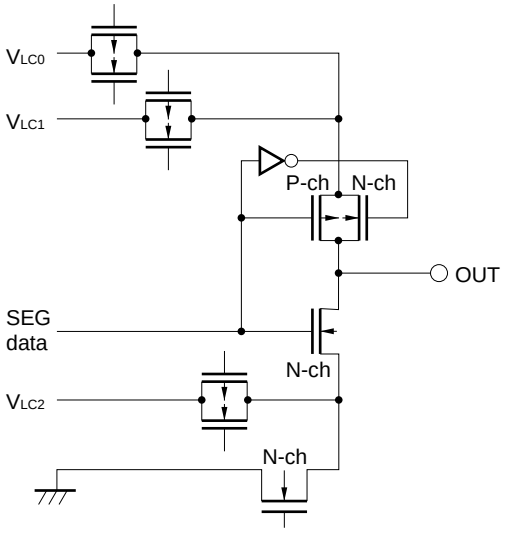


P.U.R. : Pull-Up Resistor

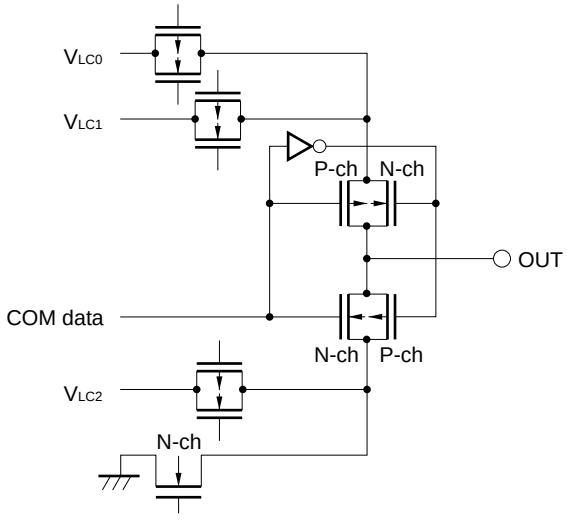
TYPE H



TYPE G-A



TYPE G-B



2.4 Recommended Connections for Unused Pins

★ **Table 2-4. List of Recommended Connections for Unused Pins**

Pin	Recommended connection
P30/PCL	Input mode: Individually connect to V_{SS} or V_{DD} via a resistor. Output mode: Leave open.
P31/BUZ	
P32	
P33	
P100	
P101	
P102	
P103/INT1	
S0 to S19	Leave open.
COM0 to COM3	
S20/P83 to S23/P80	Input mode: Individually connect to V_{SS} or V_{DD} via a resistor. Output mode: Leave open.
IC	Connect to V_{DD} directly.

CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP

The 75XL architecture employed for the μPD753304 has the following features:

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

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

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

This chapter describes each of these features.

3.1 Bank Configuration of Data Memory and Addressing Mode

3.1.1 Bank configuration of data memory

The μPD753304 is provided with static RAM at the addresses 000H through 0FFH of the memory bank 0 data memory space. Writable display data memories (24 x 4 bits) are allocated to addresses 1E0H through 1F7H of the memory bank 1. Peripheral hardware units (such as I/O ports and timers) are allocated to addresses F80H through FFFH of the memory bank 15.

The μPD753304 employs memory bank configuration that directly or indirectly specifies the lower 8 bits of an address by an instruction and the higher 4 bits of the address by a memory bank when the data memory space of 12-bit address (4 Kwords x 4 bits) is addressed.

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

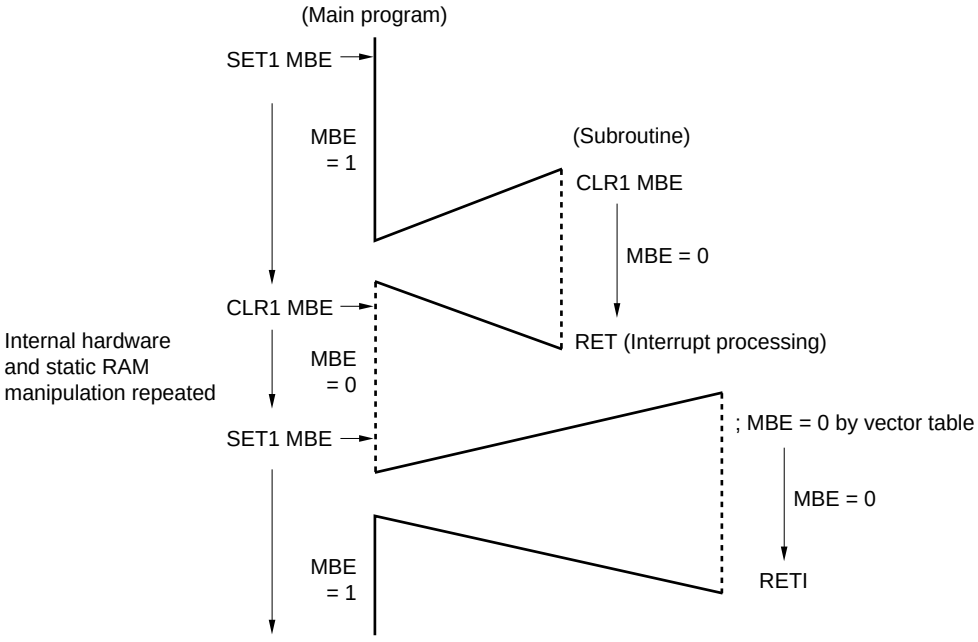
- Memory bank enable flag (MBE)
- Memory bank select register (MBS)

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

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

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

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



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

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

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

Examples 1. To clear MBE and fix memory bank

```
CLR1    MBE     ; MBE ← 0
```

2. To select memory bank 1

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

3.1.2 Addressing mode of data memory

The 75XL architecture employed for the μPD753304 provides the seven types of addressing modes as shown in Table 3-1, so that the data memory space can be efficiently addressed by the bit length of the data to be processed, and so that programming can be carried out efficiently.

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

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

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

In the mode of MBE = 1, MB = MBS.

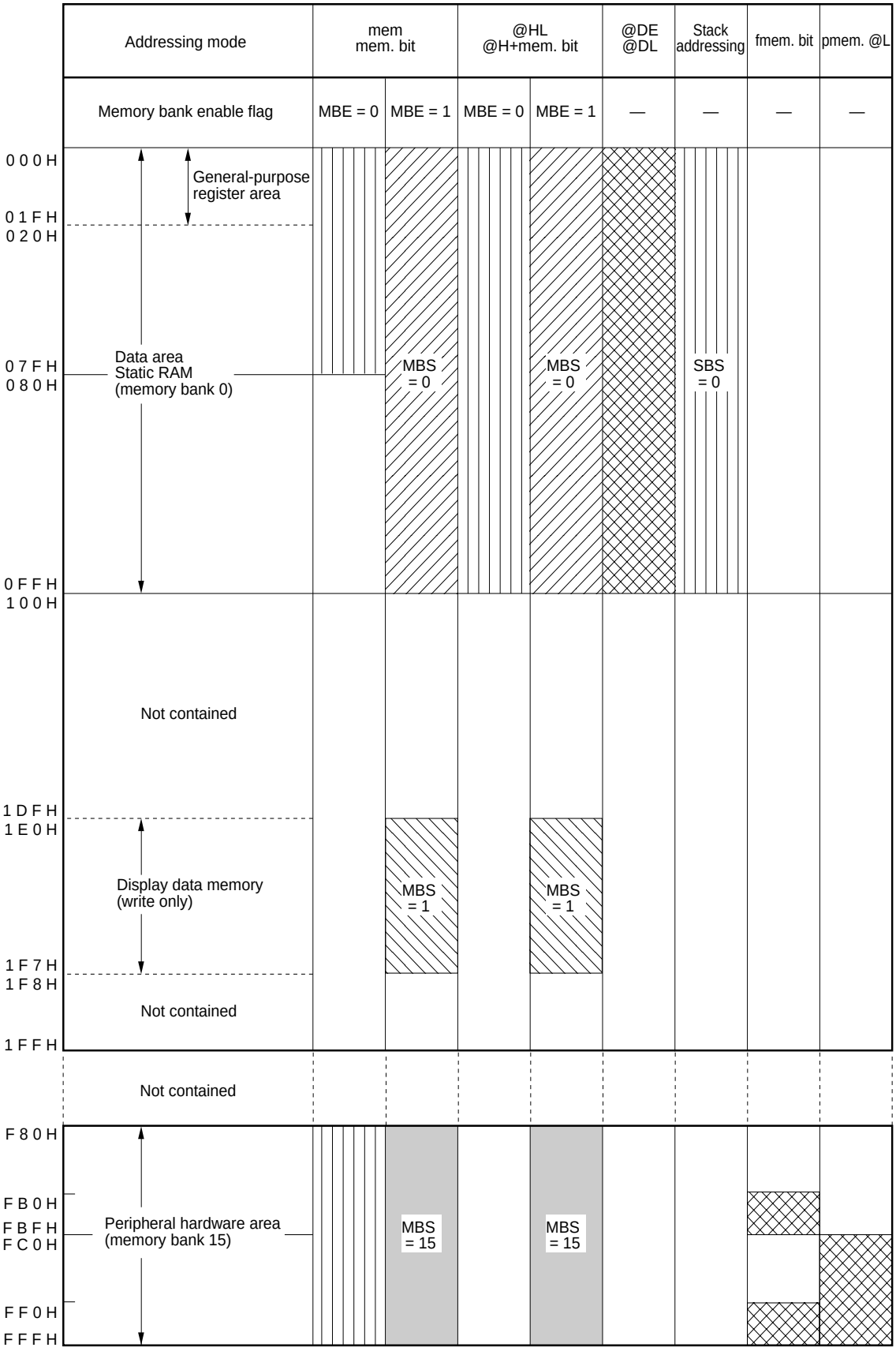
This addressing mode can be used with four instructions: bit set and reset (SET1 and CLR1) instructions, and bit test instructions (SKT and SKF).

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

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

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

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



Remark –: don't care

Table 3-1. Addressing Mode

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

(2) 4-bit direct addressing (mem)

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

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

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

(3) 8-bit direct addressing (mem)

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

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

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

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

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

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

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

When the HL register pair is specified, the autoincrement/autodecrement mode, which increments (+1) or decrements (–1) the L register upon instruction execution can be used to reduce the number of program steps.

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

```
DATA1   EQU   57H
DATA2   EQU   67H
        SET1  MBE                 ; MBE ← 1
        SEL   MB0                 ; MBS ← 0
        MOV   D, #DATA1 SHR 4     ; D ← 5
        MOV   HL, #DATA2 AND 0FFH ; HL ← 17H
LOOP:   MOV   A, @DL              ; A ← (DL)
        XCH   A, @HL–             ; A ↔ (HL), L ← L – 1
        BR    LOOP
```

The addressing mode that uses HL register pair as the data pointer is widely used to transfer, operate, compare, and input/output data. The addressing mode using DE or DL register pair 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 space can be updated as shown in Figure 3-3.

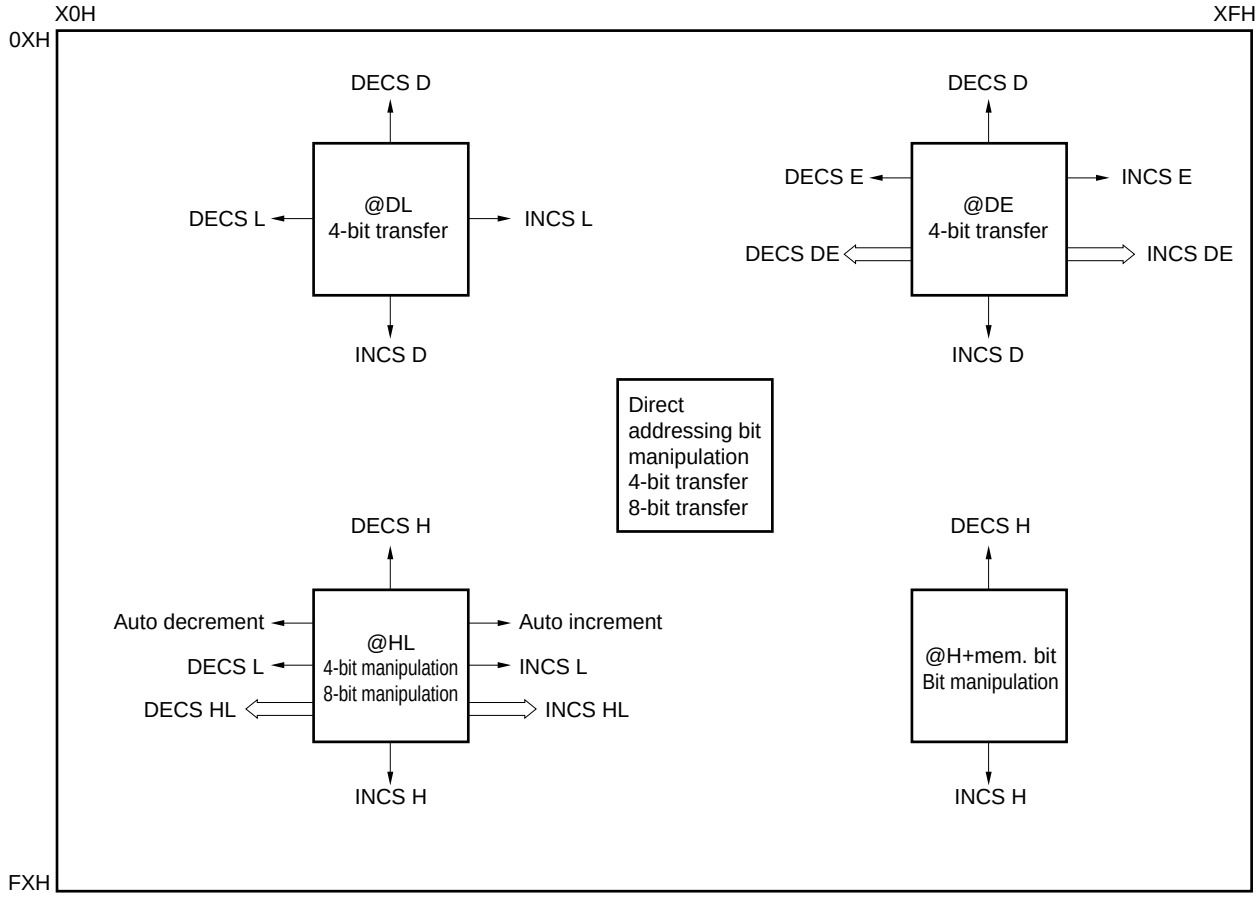
Examples 1. To compare data 50H through 57H with data 60H through 67H

```
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 00H through FFH

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

Figure 3-3. Static RAM Address Update Method



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

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

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

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

Examples 1. To compare whether the count register (T0) value of timer 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 00H through FFH

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

(6) Bit manipulation addressing

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

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

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

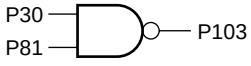
(a) Specific address bit direct addressing (fmem.bit)

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

Examples 1. To test timer 0 interrupt request flag (IRQT0) and, if it is set, clear the request flag and reset P100

```
SKTCLR    IRQT0       ; IRQT0 = 1?
BR        NO          ; NO
CLR1      PORT10.0    ; YES
```

2. To reset P103 if both P30 and P81 pins are 1




```
MOV1      CY, PORT3.0 ; CY ← P30
AND1      CY, PORT8.1 ; CY ∧ P81
NOT1      CY          ; CY ← CY̅
MOV1      PORT10.3, CY ; P103 ← CY
```

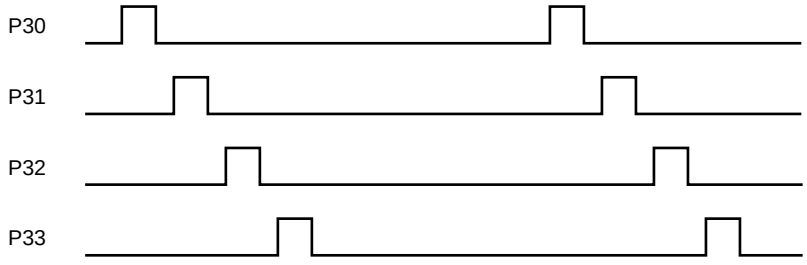
(b) Specific address bit register indirect addressing (pmem.@L)

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

This addressing mode specifies the higher 10 bits of 12-bit data memory address directly by using an operand, and the lower 2 bits and the bit address 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 3



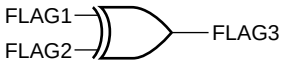

```
LOOP2:  MOV     L, #0
LOOP1:  SET1    PORT3.@L      ; Bits of port 3 (L1-0) ← 1
        CLR1    PORT3.@L      ; Bits of port 3 (L1-0) ← 0
        INCS    L
        SKE     L, #4H
        BR      LOOP1
        BR      LOOP2
```

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

This addressing mode enables bit manipulation in the entire data 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
        MOV1    CY, @H+FLAG1        ; CY ← FLAG1
        XOR1    CY, @H+FLAG2        ; CY ← CY ∀ FLAG2
        MOV1    @H+FLAG3, CY        ; FLAG3 ← CY
```

(7) Stack addressing

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

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

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

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

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

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

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

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

3.2 Bank Configuration of General-Purpose Registers

The μPD753304 is provided with four register banks with each bank consisting of eight general-purpose registers: X, A, B, C, D, E, H, and L. The general-purpose register area consisting of these registers is mapped to the addresses 00H through 1FH of memory bank 0 (refer to **Figure 3-5**). 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	x	x	Fixed to bank 0
1	0	0	0	0	Bank 0 selection
			0	1	Bank 1 selection
			1	0	Bank 2 selection
			1	1	Bank 3 selection

Fixed to 0

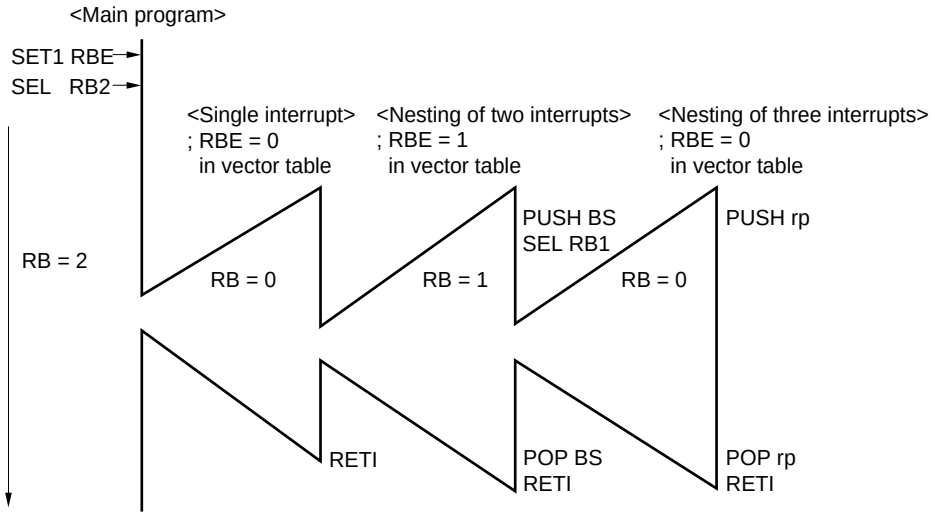
Remark x = don't care

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

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

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

Figure 3-4. Example of Using Register Banks



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

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

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

The general-purpose register area provided to the μPD753304 can be used not only as 4-bit registers, but also as 8-bit register pairs. This feature allows the μPD753304 to provide transfer, operation, comparison, and increment/decrement instructions comparable to those of 8-bit microcontrollers and allows you to program mainly with general-purpose registers.

(1) To use as 4-bit registers

When the general-purpose register area is used as a 4-bit register area, a total of eight general-purpose registers, X, A, B, C, D, E, H, and L, specified by RBE and RBS can be used as shown in Figure 3-5. Of these registers, A register 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 register pair serves as an 8-bit accumulator, playing the central role in transferring, operating, and comparing 8-bit data. The other register pairs can transfer, compare, and increment or decrement data with the accumulator. The HL register pair is mainly used as a data pointer. The DE and DL register pairs are also used as auxiliary data pointers.

Examples 1.

```
INCS    HL          ; Skips if HL ← HL+1, HL = 00H
ADDS    XA, BC      ; Skips if XA ← XA+BC, carry
SUBC    DE', XA     ; DE' ← DE' – XA – CY
MOV     XA, XA'     ; XA ← XA'
MOVT    XA, @PCDE   ; XA ← (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 BC' register pair and, if not, 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
```

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

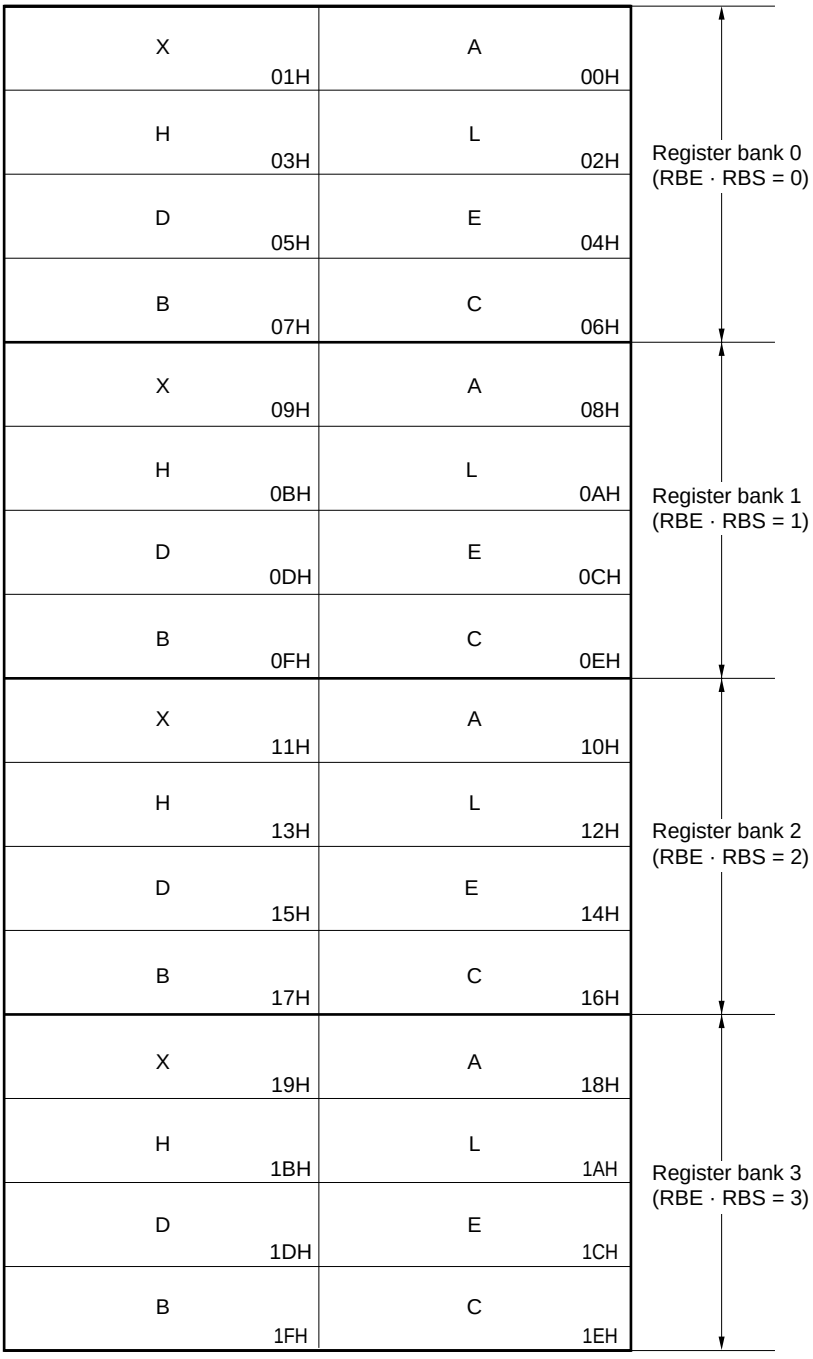
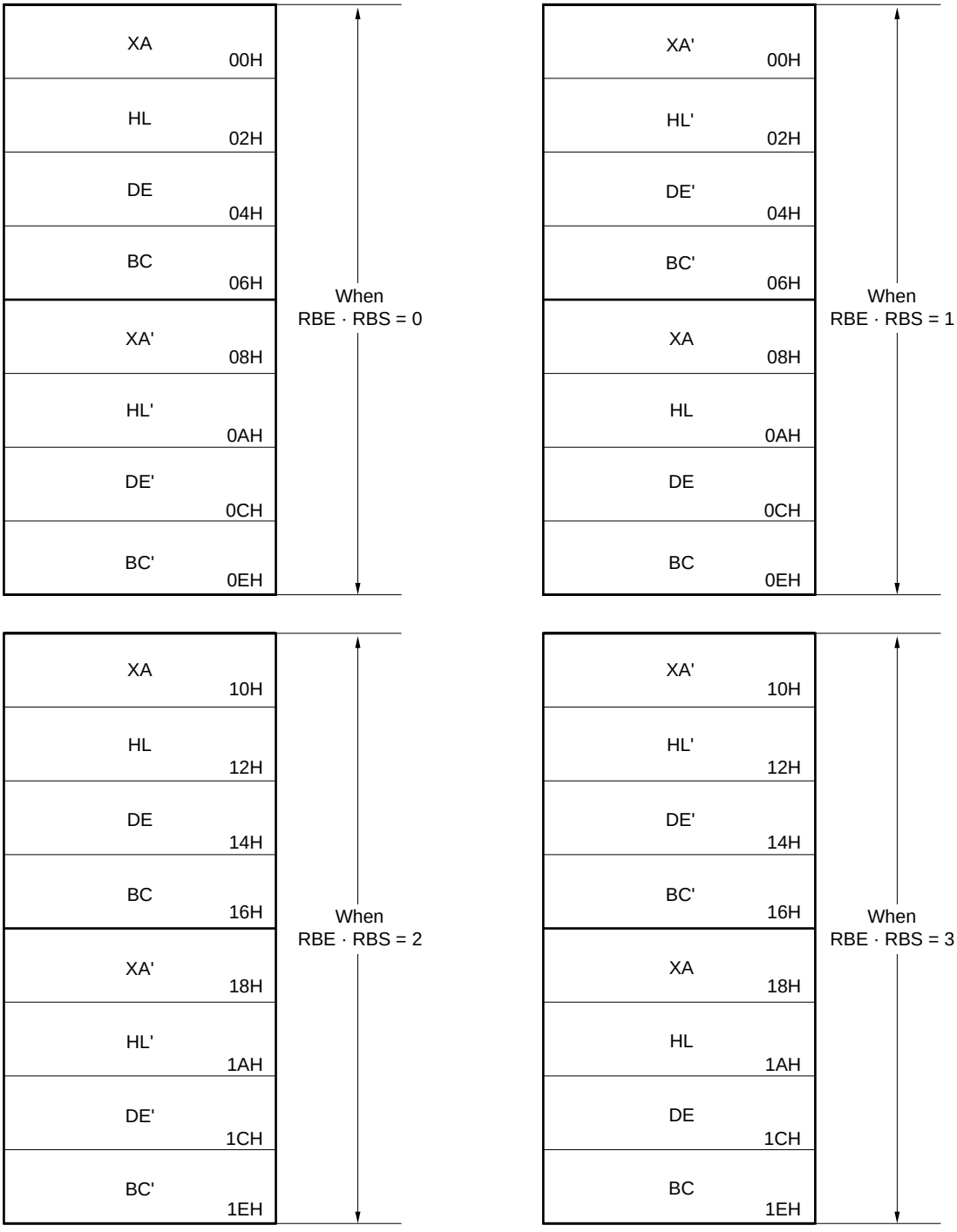


Figure 3-6. General-Purpose Register Configuration (for 8-bit operation)



3.3 Memory-Mapped I/O

The μPD753304 employs memory-mapped I/O where peripheral hardware such as the input/output ports and timers are mapped in data memory space addresses F80H to FFFH, as shown in Figure 3-2. Thus, special instructions to control the peripheral hardware are not provided and memory manipulation instructions are all used to control the peripheral hardware (Some hardware control mnemonics are provided for easy understanding of programs).

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

The display data memory (write only) mapped in addresses 1E0H to 1F7H is manipulated by specifying memory bank 1.

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

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

Example
```
CLR1    MBE       ; MBE = 0
SET1    TM0. 3    ; Starts timer 0
EI      IEBT      ; Enables INTBT
DI      IE1       ; Disables INT1
SKTCLR  IRQT0     ; Tests and clears INTT0 request flag
SET1    PORT8, @L ; Sets port 8
```

The I/O map of the μPD753304 is shown in Figure 3-7.
The meanings of the items in Figure 3-7 are as follows.

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

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

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

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

Figure 3-7. μPD753304 I/O Map (1/3)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
F80H	Stack pointer (SP)				R/W	–	–	Yes	–	Bit 0 is fixed to 0.
F82H	Register bank selection register (RBS)				R	–	Yes	Yes	–	**Note 1**
	Bank selection register (BS)									
F83H	Memory bank selection register (MBS)					–	Yes			
F84H	Stack bank selection register (SBS)				R/W	–	Yes	–	–	
F85H	Basic interval timer mode register (BTM)				W	Δ	Yes	–	mem.bit	Bit manipulation can be performed only on bit 3.
F86H	Basic interval timer (BT)				R	–	–	Yes	–	
F8BH	WDTM Note 2				W	Δ	–	–	mem.bit	Bit manipulation can be performed only on bit 3.
F8CH	Display mode register (LCDM)				R/W	Δ (W)	–	Yes	mem.bit	Bit manipulation can be performed only on bit 3.
						–	–			
F8EH	Display control register (LCDC)				R/W	–	Yes	–	–	
F8FH	LCD/port selection register (LPS)				R/W	–	Yes	–	–	

F98H	Watch mode register (WM)	R/W	–	–	Yes	–	

FA0H	Timer counter 0 mode register (TM0)	R/W	Δ (W)	–	Yes	mem.bit	Bit manipulation can be performed only on bit 3.
			–	–		–	
FA4H	Timer counter 0 count register (T0)	R	–	–	Yes	–	
FA6H	Timer counter 0 modulo register (TMOD0)	R/W	–	–	Yes	–	

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

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

Figure 3-7. μPD753304 I/O Map (2/3)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FB0H	IST1	IST0	MBE	RBE	R/W	Yes(R/W)	Yes(R/W)	Yes	fmem.bit	8-bit manipulation is read only.
	Program status word (PSW)									
	(CY)Note 1	(SK2)Note 1	(SK1)Note 1	(SK0)Note 1		–	–	(R)		
FB2H	Interrupt priority selection register (IPS) IME	–	–	–	R/W	–	Yes	–		**Note 2**
FB3H	Processor clock control register (PCC)				R/W	–	Yes	–		**Note 3**
FB5H	INT1 edge detection mode register (IM1)				R/W	–	Yes	–	–	
FB7H	System clock control register (SCC)				R	Δ (R/W)	Yes	–	–	Bit manipulation can be performed only on bits 0, 3.
FB8H	INTA register (INTA) –	–	IEBT	IRQBT	R/W	Yes	Yes	–	fmem.bit	**Note 4**
FBAH	INTC register (INTC) –	–	IEW	IRQW	R/W	Yes	Yes			**Note 4**
FBCH	INTE register (INTE) –	–	IET0	IRQT0	R/W	Yes	Yes	–		**Note 4**
FBEH	INTG register (INTG) IE1	IRQ1	–	–	R/W	Yes	Yes	–		**Note 4**

FCFH	Sub-oscillator control register (SOS)	R/W	–	Yes	–	–	

Notes 1. Not registered as a reserved word.

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

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

4. 1-bit manipulation can be performed only on the reserved word for 1-bit manipulation.

Remarks 1. IEXXX : Interrupt enable flag

2. IRQXXX: Interrupt request flag

Figure 3-7. μPD753304 I/O Map (3/3)

Address	Hardware name (symbol) b3	b2	b1	b0	R/W	Manipulation unit 1-bit	4-bit	8-bit	Bit manipulation addressing	Remarks
FD0H	Clock output mode register (CLOM)				R/W	–	Yes	–	–	
FDEH	–	–	–	(PO10)Note	R/W	–	–	Yes	–	
	Pull-up resistor specification register group B (POGB)									
	–	–	–	–						
FE8H	(PM33)Note	(PM32)Note	(PM31)Note	(PM30)Note	R/W	–	–	Yes	–	
	Port mode register group A (PMGA)									
	–	–	–	–						
FEAH	(PM103)Note	(PM102)Note	(PM101)Note	(PM100)Note	R/W	–	–	Yes	–	
	Port mode register group D (PMGD)									
	–	–	–	–						
FEEH	–	–	–	(PM8)Note	R/W	–	–	Yes	–	
	Port mode register group C (PMGC)									
	–	–	–	–						
FF3H	Port 3 (PORT3)				R/W	Yes	Yes	–	–	
FF8H	Port 8 (PORT8)				R/W	Yes	Yes	–		
FFAH	Port 10 (PORT10)				R/W	Yes	Yes	–		

Note Not registered as a reserved word.

CHAPTER 4 INTERNAL CPU FUNCTIONS

4.1 Switching Function between Mk I Mode and Mk II Mode

4.1.1 Difference between Mk I mode and Mk II mode

The CPU of the μPD753304 has the following two modes: Mk I and Mk II, either of which can be selected. The mode can be switched by the bit 3 of the stack bank selection register (SBS).

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

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

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

Caution The Mk II mode supports a program area exceeding 16 Kbytes for the 75X and 75XL Series. Therefore, this mode is effective for enhancing software compatibility with products exceeding 16 Kbytes.
When the Mk II mode is selected, the number of stack bytes used during execution of subroutine call instructions (used area) increases by one byte per stack compared to the Mk I mode. When the CALL !addr and CALLF !faddr instructions are used, the machine cycle becomes longer by one machine cycle. Therefore, use the Mk I mode if the RAM efficiency and processing performance are more important than software compatibility.

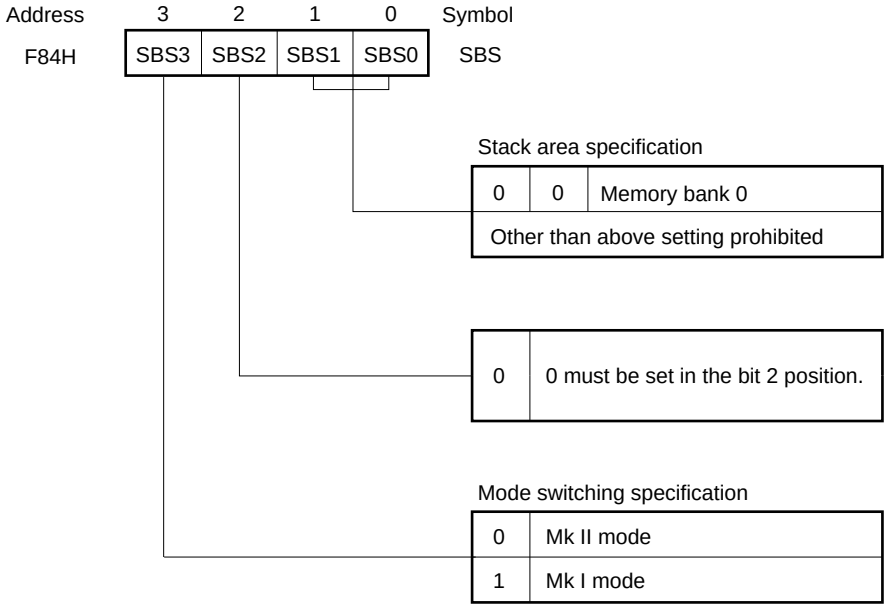
4.1.2 Setting method of stack bank selection register (SBS)

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

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

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

Figure 4-1. Stack Bank Selection Register Format



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

4.2 Program Counter (PC) ····· 12-bit

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

Figure 4-2. Program Counter Configuration

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

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

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

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

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

When the $\overline{\text{RESET}}$ signal is generated, the program counter (PC) is loaded with the contents of addresses 0000H and 0001H in the program memory. The program can start from any address according to the conents of the 0000H and 0001H addresses.

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

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

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

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

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

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

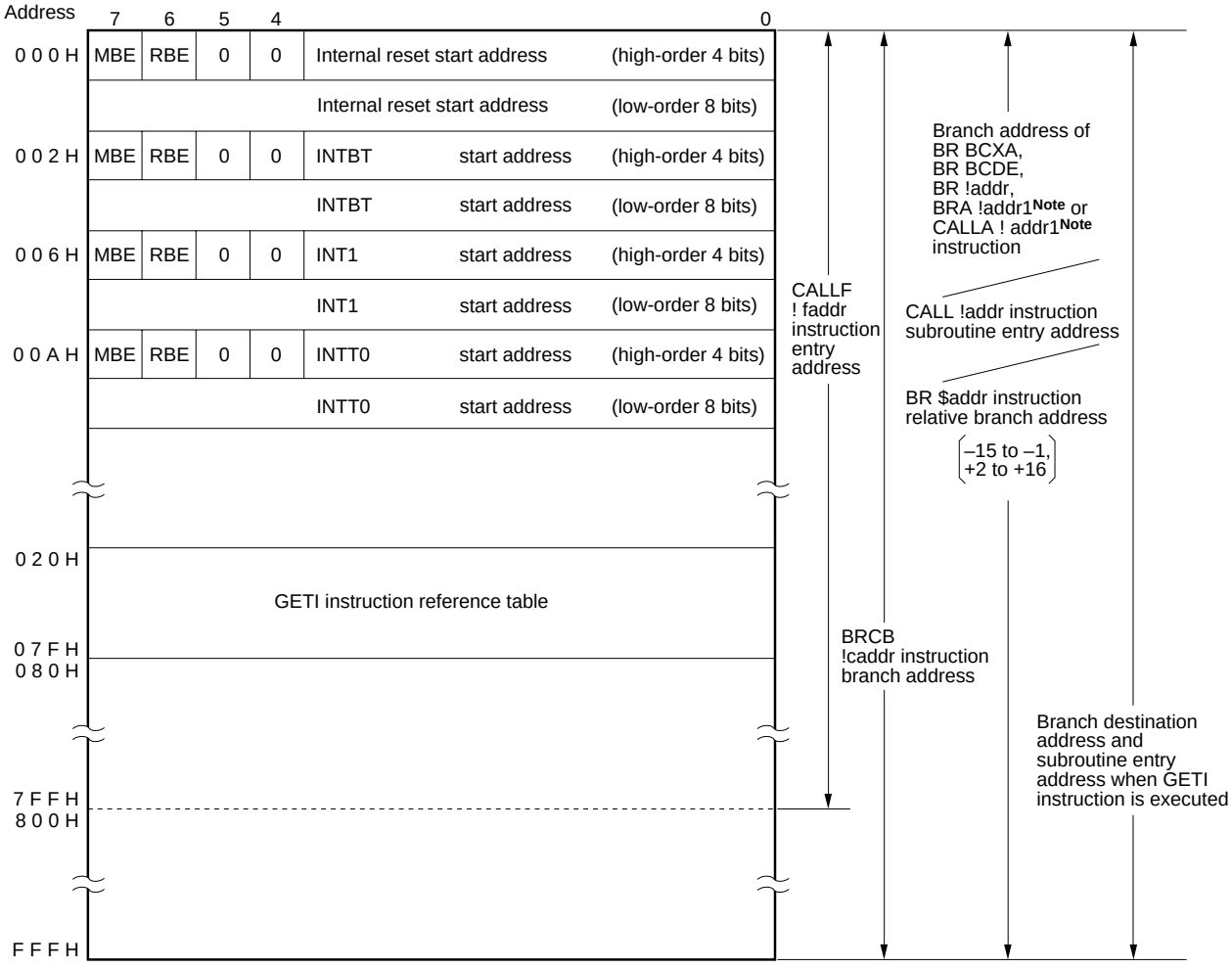
- Addresses 0000H and 0001H
  Vector table wherein the program start address and the values set for the RBE and MBE at the time a $\overline{\text{RESET}}$ signal is generated are written. Reset and start are possible from any address.

- Addresses 0002H to 000BH
  Vector table wherein the program start address and values set for the RBE and MBE by the vectored interrupts are written. Interrupt processing can be started from any address.

- Addresses 0020H to 007FH
  Table area referenced by the GETI instruction[Note].

Note The GETI instruction realizes a 1-byte instruction on behalf of any 2-byte instruction, 3-byte instruction, or two 1-byte instructions. It is used to decrease the program steps (See **10.1.1 GETI instruction**).

Figure 4-3. Program Memory Map



Note Can be used only in the Mk II mode.

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

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

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

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

4.4.1 Configuration of data memory

(1) Data area

A data area consists of static RAM, and is used to store data and as a stack memory when a subroutine or interrupt is executed. The contents of this area can be backed up for a long time by batteries even when the CPU is stopped in the standby mode. The data area is manipulated by using memory manipulation instructions.
Static RAM is mapped to memory bank 0 in units of 256 words x 4 bits. Although bank 0 is mapped as a data area, it can also be used as a general-purpose register area (000H through 01FH) and as a stack area (000H through 0FFH). Addresses 1E0H through 1F7H of bank 1 are used as a display data memory[Note 1].
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[Note 2]. To use an 8-bit manipulation instruction, specify an even address.

Notes 1. Write only
2. The display data memory cannot be manipulated in 8-bit units.

- **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 (Refer to **4.5 General-Purpose Registers**).

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

- **Display data memory (write only)**
  The display data of an LCD are written to this area. The data written to this display data memory are automatically read and displayed by hardware when the LCD is driven. The addresses of this area not used for display cannot be used as data area.

(2) Peripheral hardware area

The peripheral hardware area is mapped to addresses F80H through FFFH of memory bank 15.
This area is manipulated by using a memory manipulation instruction, in the same manner as the static area. Note, however, that the bit units in which the peripheral hardware units can be manipulated differ depending on the address. The addresses to which no peripheral hardware unit is allocated cannot be accessed because these addresses are not provided to the data memory.

4.4.2 Specifying bank of data memory

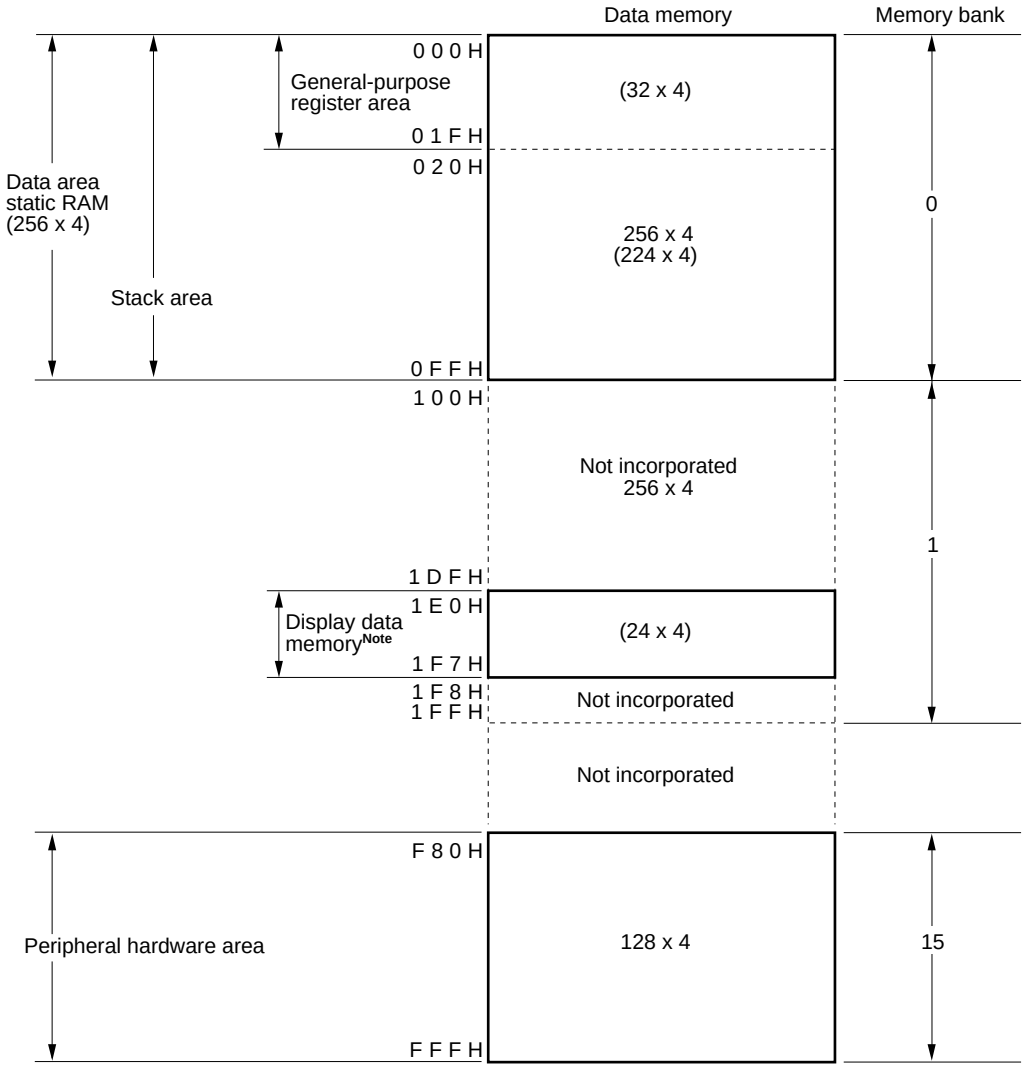
A memory bank is specified by setting a 4-bit memory bank selection register (MBS) to 0, 1, or 15 when bank specification is enabled by setting a memory bank enable flag (MBE) to 1. When bank specification is disabled (MBE = 0), bank 0 or 15 is automatically specified depending on the addressing mode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

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

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

- General-purpose register area..... See **4.5 General-Purpose Registers**
- Stack area ..... See **4.7 Stack Pointer (SP) and Stack Bank Selection Register (SBS)**
- Display data memory ..... See **5.6.6 Display data memory**
- Peripheral hardware area ..... See **CHAPTER 5 PERIPHERAL HARDWARE FUNCTION**

Figure 4-4. Data Memory Map



Note Write only

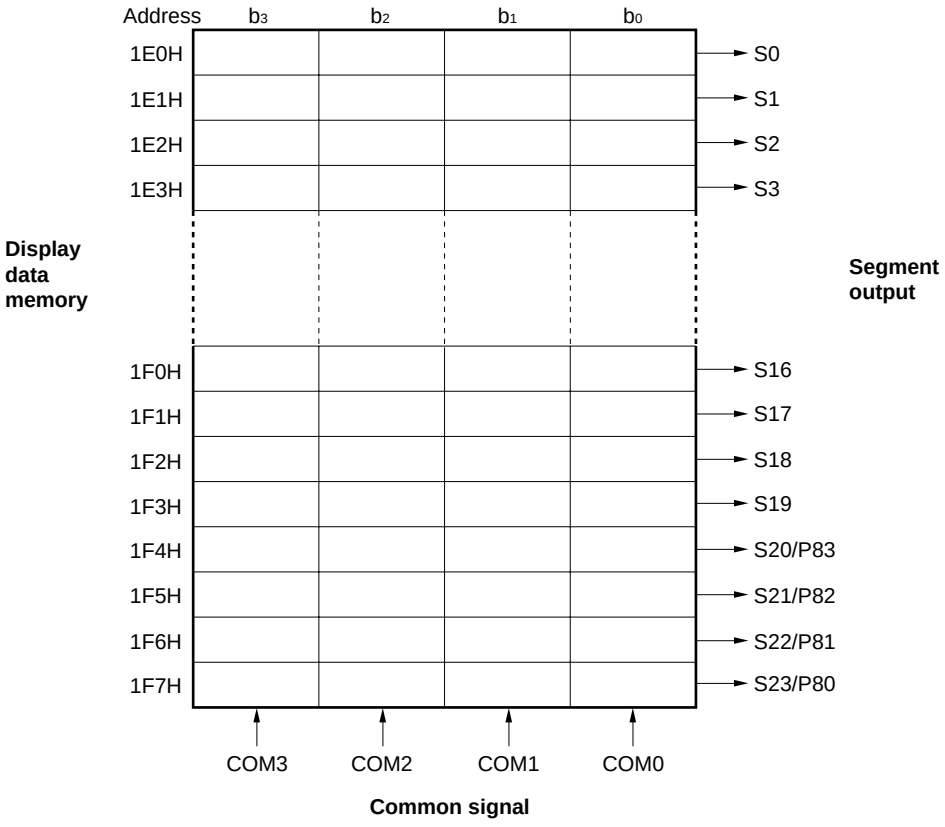
The contents of the data memory are undefined at reset. Therefore, they must be initialized at the beginning of program execution (RAM clear). Otherwise, unexpected bugs may occur.

Example To clear RAM at addresses 000H through 0FFH, 1E0H through 1F7H

```
        SET1    MBE
        SEL     MB0
        MOV     A, #0H
        MOV     HL, #04H
RAMC0:  MOV     @HL, A      ; Clears 04H to FFH[Note]
        INCS    HL          ; HL ← HL+1
        BR      RAMC0
        SEL     MB1
        MOV     HL, #0E0H
RAMC1:  MOV     @HL, A      ; Clears 1E0H to 1F7H
        INCS    HL          ; HL ← HL+1
        SKE     H, #0EH
        SKE     L, #8H
        BR      RAMC1
```

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

Figure 4-5. Configuration of Display Data Memory



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

Caution The display data memory cannot be manipulated in 8-bit units and is write only.

Example To clear display data memory at addresses 1E0H to 1F7H

```
        SET1    MBE
        SEL     MB1
        MOV     HL, #0E0H
        MOV     A, #0H
LOOP:   MOV     @HL, A      ; Clears display data memory in 4-bit units at once
        INCS    HL
        SKE     H, #0EH
        SKE     L, #8H
        BR      LOOP
```

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

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

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

RB = RBE•RBS (RBS = 0 to 3)

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

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

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

Figure 4-6. General-Purpose Register Configuration

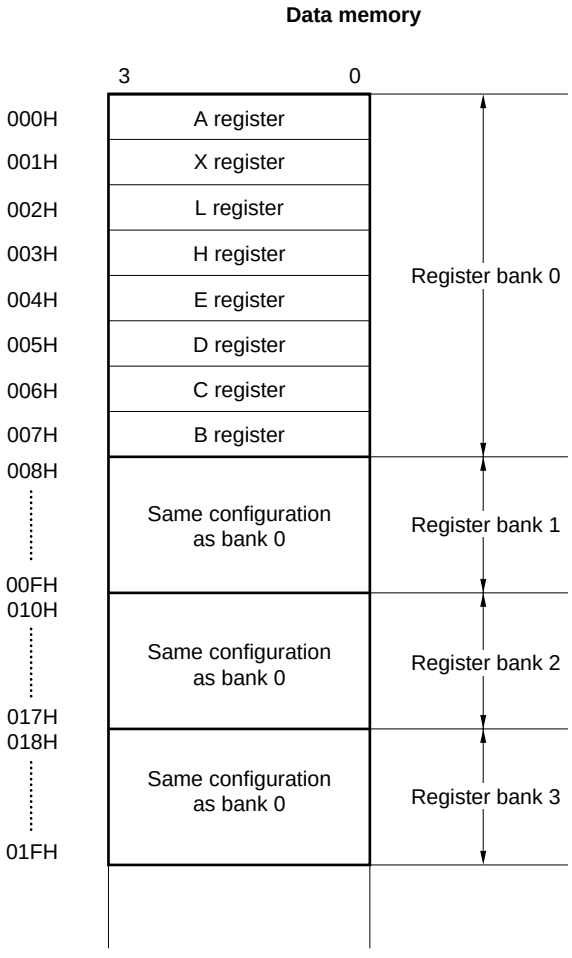
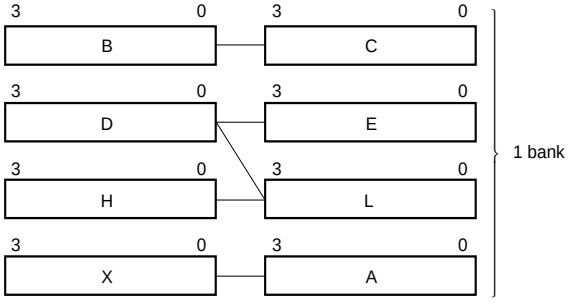


Figure 4-7. Register Pair Configuration

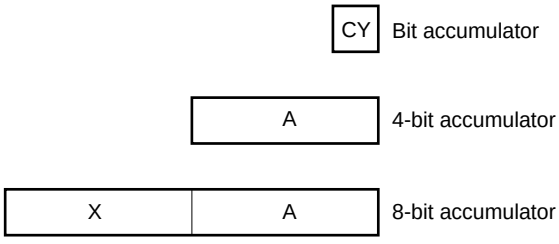


4.6 Accumulators

The μPD753304 uses the A register and XA register pair as accumulators. The A register is used as the main register during execution of 4-bit data processing instructions; the XA register pair is used as the main register pair during execution of 8-bit data processing instructions.

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

Figure 4-8. Accumulators



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

The μPD753304 uses static RAM for stack memory (LIFO). The stack pointer (SP) is an 8-bit register which holds top address information of the stack area.

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

Table 4-2. Stack Area Selected by SBS

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

The SP decrements before a write (save) operation in the stack memory and increments after a read (restore) operation from the stack memory.

Figures 4-10 to 4-13 show the data saved and restored by the stack operations.

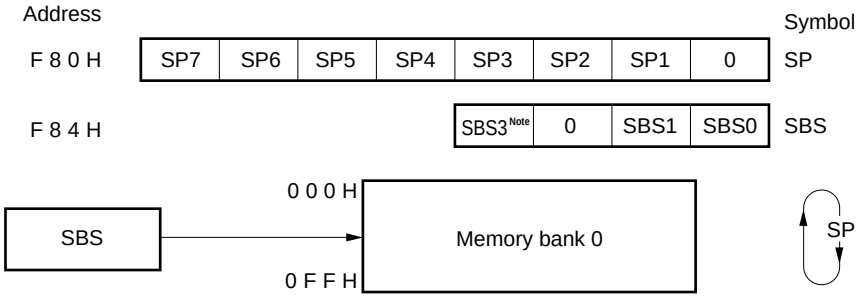
The initial value of the SP is set by an 8-bit memory manipulation instruction and the initial value of the SBS is set by a 4-bit memory manipulation instruction to determine a stack area. Its contents can also be read.

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

The stack area is limited to the memory bank 0, and data is returned to 0FFH when further stacking operation is performed in addresses starting with 000H.

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

Figure 4-9. Stack Pointer and Stack Bank Selection Register Configuration



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

Example SP Initialization

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

```
SEL     MB15        ; or CLR1 MBE
MOV     A, #8H
MOV     SBS, A      ; Assign memory bank 0 to the stack area (Mk I mode)
MOV     XA, #00H
MOV     SP, XA      ; SP ← 00H (Stack operation from 0FFH)
```

Figure 4-10. Data Saved in Stack Memory (Mk I mode)

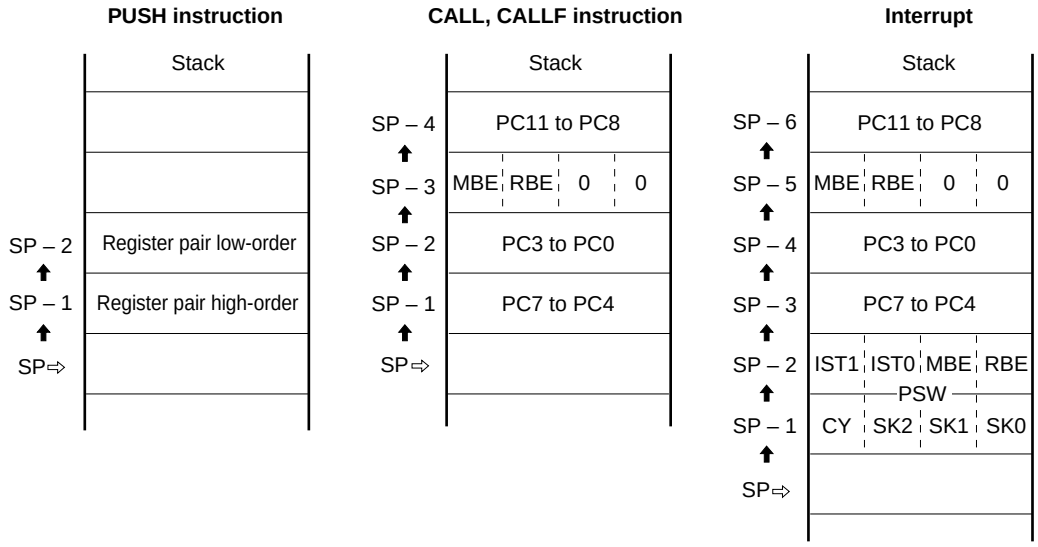


Figure 4-11. Data Restored from Stack Memory (Mk I mode)

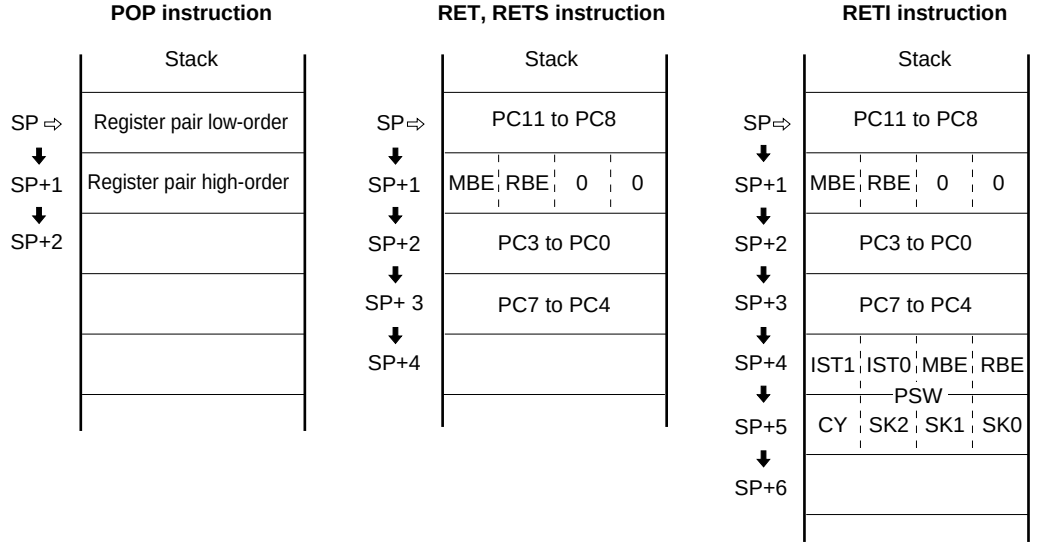


Figure 4-12. Data Saved in Stack Memory (Mk II mode)

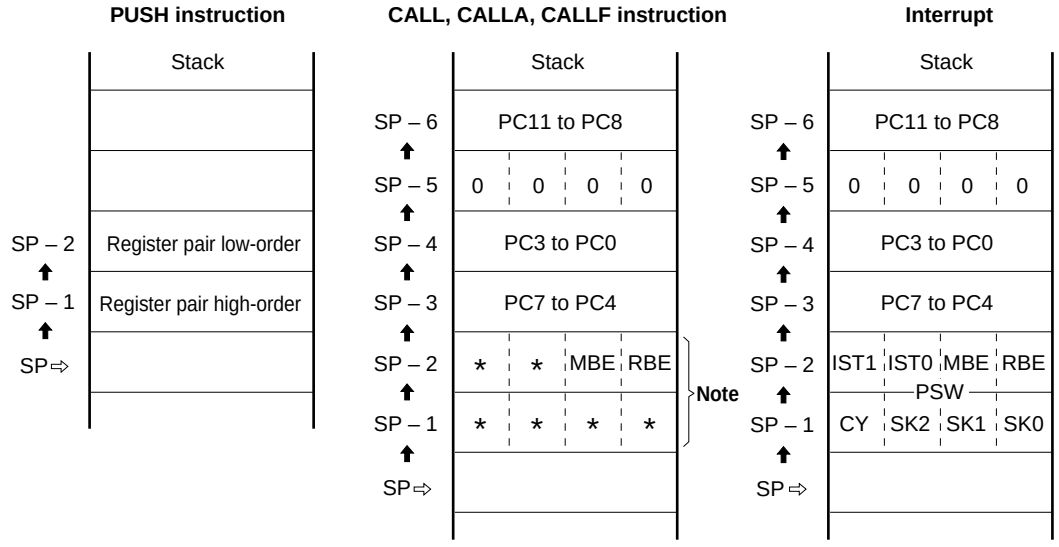
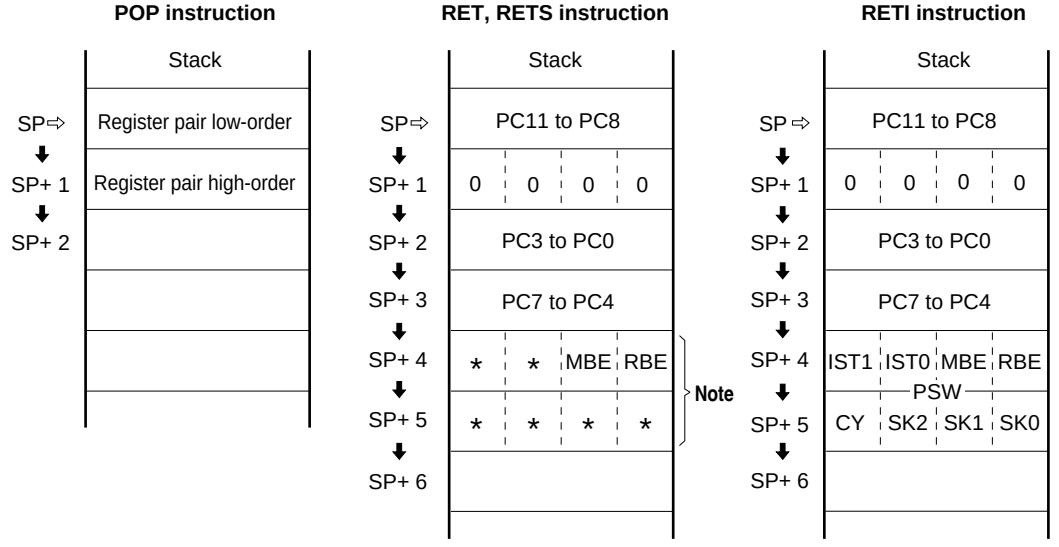


Figure 4-13. Data Restored from Stack Memory (Mk II mode)



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

Remark A star mark (*) in the above illustrations means undefined.

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

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

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

Figure 4-14. Program Status Word Format

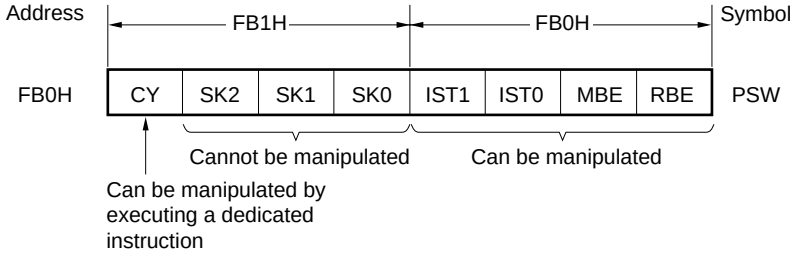


Table 4-3. PSW Flags Saved and Restored during Stack Operation

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

(1) Carry flag (CY)

The carry flag (CY) is a 1-bit flag which stores overflow or underflow occurrence information when an operation instruction with carry (ADDC or SUBC) is executed. The carry flag also serves as a bit accumulator.

Boolean algebra operation is performed between the bit accumulator and the data memory specified by bit address, and the result can be stored in the accumulator.

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

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

Table 4-4. Carry Flag Manipulation Instructions

	Instruction (mnemonics)		Carry flag operation and processing
Carry flag manipulation dedicated instruction	SET1	CY	Set CY to 1
	CLR1	CY	Clear CY to 0
	NOT1	CY	Reverses the CY status
	SKT	CY	Skip if CY contains 1
Bit transfer instruction	MOV1	mem*.bit, CY	Transfer CY contents to the specified bit
	MOV1	CY, mem*.bit	Transfer the specified bit contents to CY
Bit Boolean instruction	AND1	CY, mem*.bit	AND, OR, and XOR in the specified bit contents and CY contents and set the result in CY
	OR1	CY, mem*.bit	
	XOR1	CY, mem*.bit	
Interrupt processing	When interrupt is executed		Save CY and other PSW bits in stack memory in 8-bit parallel
	RETI		Restore CY and other PSW bits in parallel from stack memory

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

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

Example AND address 3FH bit 3 and P33 and output the result to P80.

```
MOV     H, #3H          ; Set high-order 4-bit address in H register
MOV1    CY, @H+0FH.3    ; CY ← 3FH bit 3
AND1    CY, PORT3.3     ; CY ← CY ^ P33
MOV1    PORT8.0, CY     ; P80 ← CY
```

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

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

(3) Interrupt status flag (IST1, IST0)

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

Table 4-5. Interrupt Status Flag Indication

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

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

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

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

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

(4) Memory bank enable flag (MBE)

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

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

When MBE is set to 1, the data memory address space depends on the MBS setting.

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

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

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

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

(5) Register bank enable flag (RBE)

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

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

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

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

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

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

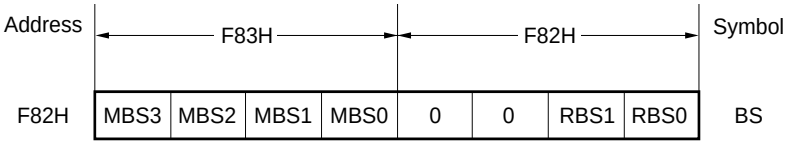
4.9 Bank Selection Register (BS)

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

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

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

Figure 4-15. Bank Selection Register Format



(1) Memory bank selection register (MBS)

The memory bank selection register (MBS) is a 4-bit register which stores high-order 4-bit address information of a 12-bit data memory address. The memory bank to be accessed is specified by the register contents (For the μPD753304, only banks 0, 1, and 15 can be specified).

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

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

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

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

MBE	MBS				Memory bank
	3	2	1	0	
0	x	x	x	x	Fixed to memory bank 0
1	0	0	0	0	Memory bank 0 selection
	0	0	0	1	Memory bank 1 selection
	1	1	1	1	Memory bank 15 selection
Other than above					Setting prohibited

x : don't care

(2) Register bank selection register (RBS)

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

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

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

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

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

Fixed to 0

x : don't care

[MEMO]

CHAPTER 5 PERIPHERAL HARDWARE FUNCTION

5.1 Digital I/O Port

Memory mapped I/O is employed for the μPD753304. All the I/O ports are mapped in the data memory space.

Figure 5-1. Digital Ports Data Memory Addresses

Address	3	2	1	0	
F F 3 H	P33	P32	P31	P30	PORT3
F F 8 H	P83	P82	P81	P80	PORT8
F F A H	P103	P102	P101	P100	PORT10

Table 5-2 lists the input/output ports manipulation instructions. For PORT3 and PORT10, in addition to 4-bit input/output, bit manipulation can be performed.

Examples 1. The status shown on P33 is tested and the values depending on the results of test are output to port 8.

```
SKT    PORT3.3      ; Skip if bit 3 of port 3 is 1.
MOV    A, #8H       ; A ← 8H
MOV    A, #4H       ; A ← 4H      String effect
SEL    MB15         ; or CLR1 MBE
OUT    PORT8, A     ; Port 8 ← A
```

2. SET1 PORT10.@L ; The bit (in port 10) specified by the L register is set to 1.

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

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

The configurations of the ports are shown in Figures 5-2 to 5-7.

Table 5-1. 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 bit-wise	Also used for PCL and BUZ
PORT8		Can be set to input or output mode in 4-bit units	Also used for S20 to S23
PORT10		Can be set to input or output mode bit-wise	Also used for INT1

Figure 5-2. P30 Configuration

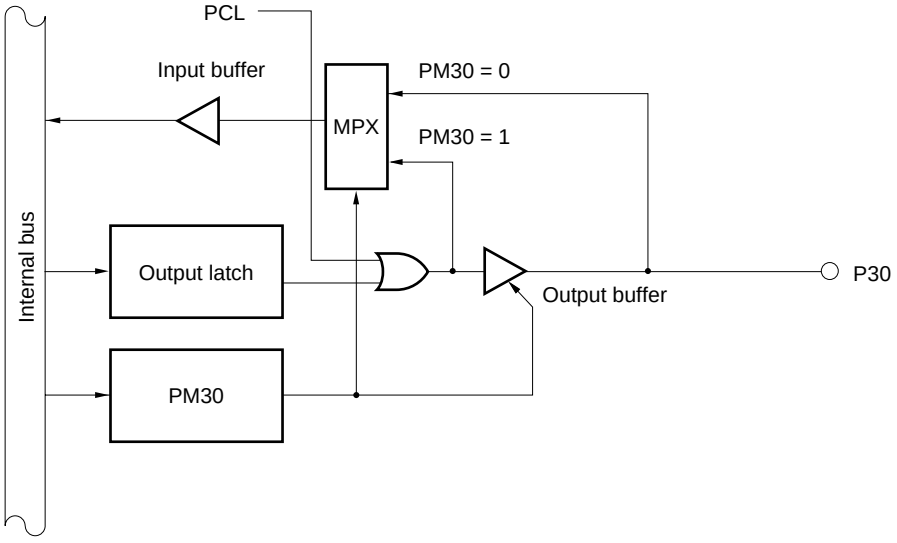


Figure 5-3. P31 Configuration

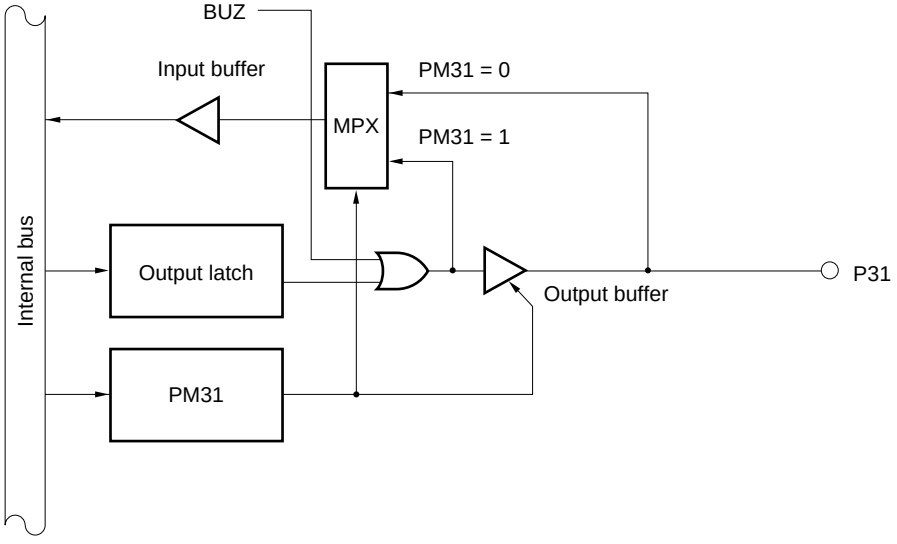


Figure 5-4. P3n Configuration (n = 2, 3)

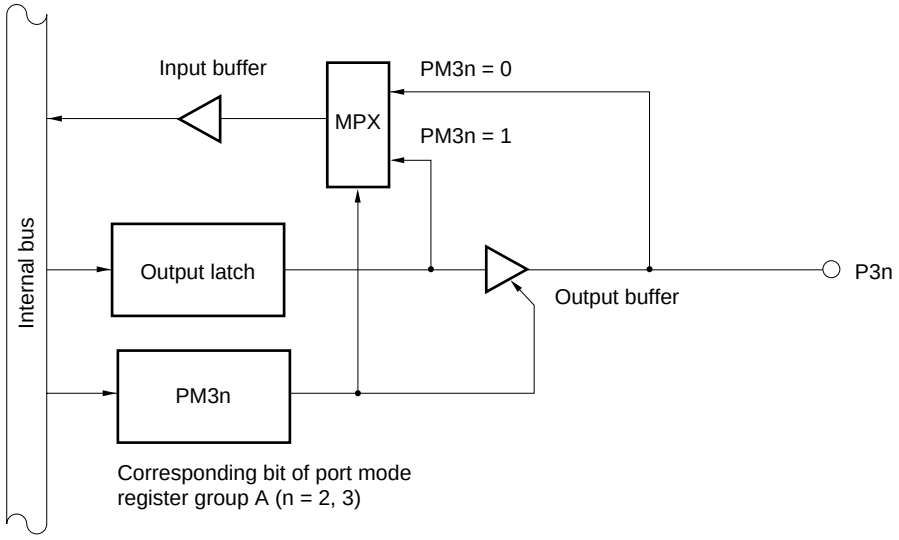


Figure 5-5. Port 8 Configuration

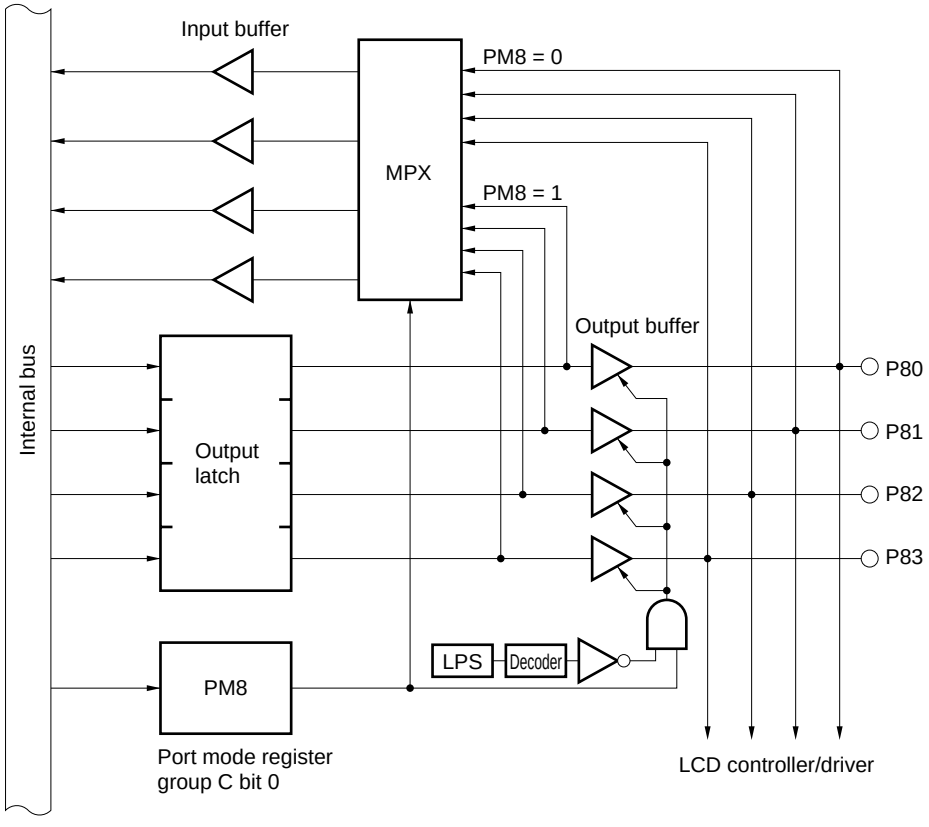


Figure 5-6. P10n Configuration (n = 0 to 2)

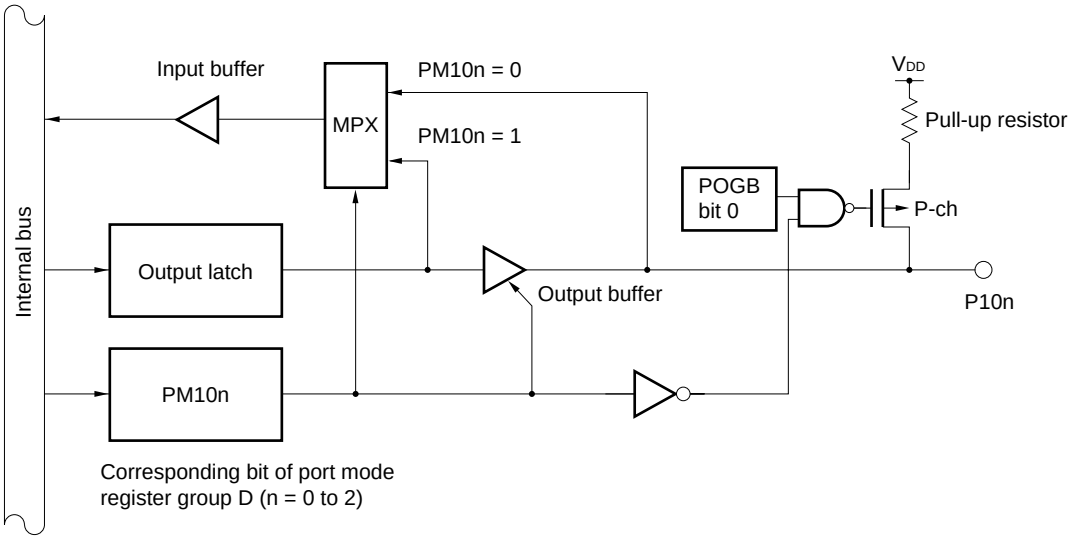
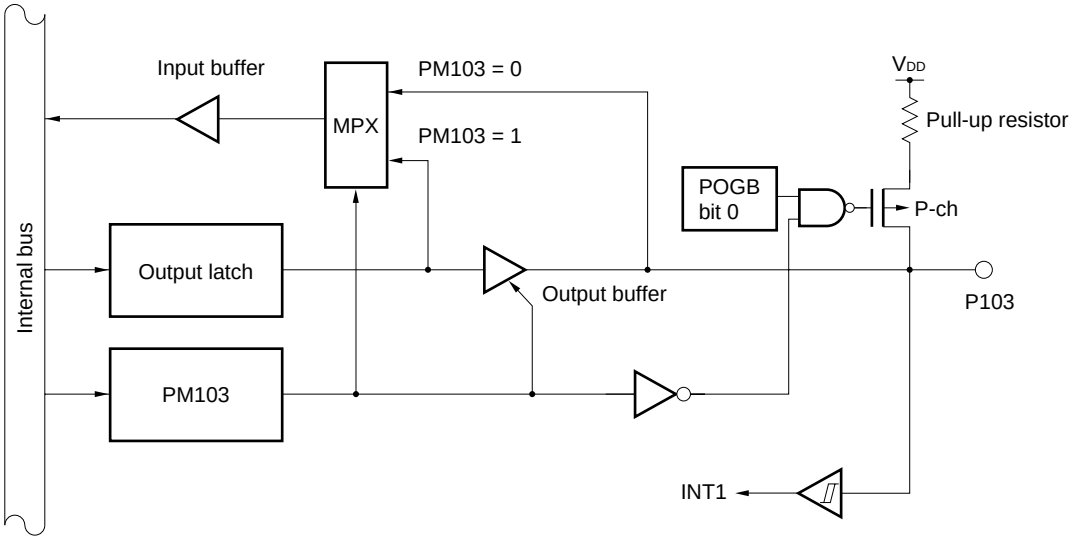


Figure 5-7. P103 Configuration



5.1.2 Setting I/O mode

The input or output mode of each I/O port is set by the corresponding port mode register as shown in Figure 5-8. The input or output mode can be specified bit-wise for port 3 by using the port mode register group A (PMGA), for port 10 by using the port mode register group D (PMGD). Port mode register group C (PMGC) is used to specify the input or output mode of port 8 in 4-bit units.

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

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

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

When the $\overline{\text{RESET}}$ signal is generated, each port mode register and each port are set as follows.

Port mode register group A (PMGA): 0FH
Port mode register group C (PMGC): 00H
Port mode register group D (PMGD): 00H
★ Port 3 (P30 to P33): Input/output mode after reset can be specified with mask option. Refer to Table 2-2 for details.
Port 8 (P80 to P83): Input mode
Port 10 (P100 to P103): Input mode (with pull-up resistor)

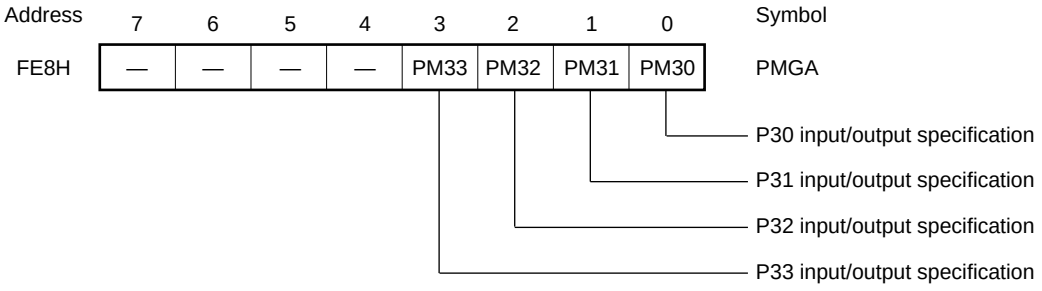
Example To use P30, 31 as input pins and P32, 33 as output pins

```
CLR1    MBE         ; or SEL MB15
MOV     XA, #0CH
MOV     PMGA, XA
```

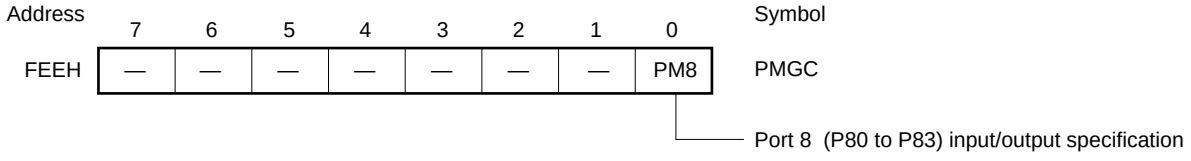
Figure 5-8. Port Mode Register Formats

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

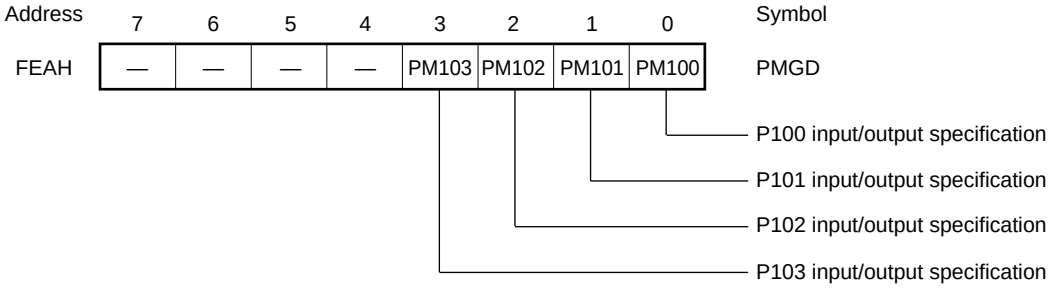
Port mode register group A



Port mode register group C



Port mode register group D



5.1.3 Digital I/O port manipulation instruction

Because all the I/O ports of the μPD753304 are mapped to the data memory space, they can be manipulated by using data memory manipulation instructions. Of these data memory manipulation instructions, those considered to be especially useful for manipulating the I/O pins and their range of applications are shown in Table 5-2.

(1) Bit manipulation instruction

Because the specific address bit direct addressing (fmem.bit) and specific address bit register indirect addressing (pmem.@L) are applicable to digital I/O ports 3, 8, and 10, the bits of these ports can be manipulated regardless of the specifications by MBE and MBS.

Example To OR P30 and P31 and set P81 in output mode

```
MOV1    CY, PORT3.0     ; CY ← P30
OR1     CY, PORT3.1     ; CY ← CY V P31
MOV1    PORT8.1, CY     ; P81 ← 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 8

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT8
ADDS    A, @HL          ; A ← A+PORT8
NOP
MOV     @HL, A          ; PORT8 ← A
```

3. To test whether the data of port 10 is greater than the value of the accumulator

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT10
SUBS    A, @HL          ; A < PORT10
BR      NO              ; NO
                        ; YES
```

Table 5-2. I/O Pin Manipulation Instructions

Instruction \ PORT			PORT 3	PORT 8	PORT 10
IN	A, PORTn	Note 1	√		
IN	XA, PORTn		—		
OUT	PORTn, A	Note 1	√		
OUT	PORTn, XA		—		
MOV	A, PORTn	Note 1	√		
MOV	PORTn, A	Note 1	√		
XCH	A, PORTn	Note 1	√		
MOV1	CY, PORTn.bit		√		
MOV1	CY, PORTn.@L	Note 2	√		
MOV1	PORTn.bit, CY		√		
MOV1	PORTn.@L, CY	Note 2	√		
INCS	PORTn	Note 1	√		
SET1	PORTn.bit		√		
SET1	PORTn.@L	Note 2	√		
CLR1	PORTn.bit		√		
CLR1	PORTn.@L	Note 2	√		
SKT	PORTn.bit		√		
SKT	PORTn.@L	Note 2	√		
SKF	PORTn.bit		√		
SKF	PORTn.@L	Note 2	√		
SKTCLR	PORTn.bit		√		
SKTCLR	PORTn.@L	Note 2	√		
AND1	CY, PORTn.bit		√		
AND1	CY, PORTn.@L	Note 2	√		
OR1	CY, PORTn.bit		√		
OR1	CY, PORTn.@L	Note 2	√		
XOR1	CY, PORTn.bit		√		
XOR1	CY, PORTn.@L	Note 2	√		

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

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

5.1.4 Operation of digital I/O port

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

(1) Operation in input mode

When a test instruction such as SKT, a bit input instruction such as MOV1, or an instruction that loads port data to the internal bus in 4-bit units, such as IN, MOV, an operation, or a comparison instruction, is executed, the data of each pin is manipulated.

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

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

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

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

(2) Operation in output mode

When a test instruction, bit input instruction, or an instruction that loads port data to the internal bus in 4-bit units 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, and the contents of the accumulator are latched to the output latches of the specified port and output from the port pins.

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

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

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

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

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

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

5.1.5 Connecting pull-up resistors

Port 10 of the μPD753304 can be connected to an on-chip pull-up resistor. Port 10 can be connected with a pull-up resistor via software.

Table 5-4 shows how to specify connection of the pull-up resistor. The on-chip pull-up resistor is connected via software in the format shown in **Figure 5-9**.

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

$\overline{\text{RESET}}$ signal initializes POGB to 01H.

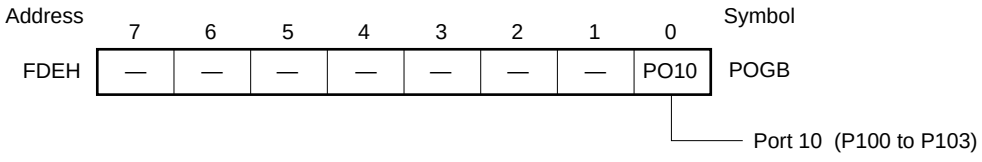
Table 5-4. On-Chip Pull-Up Resistor Specification Method

Port (Pin name)	Pull-up resistor specification method	Specified bit
Port 10 (P100 to P103)	Specifies connection in 4-bit units via software.	POGB.0

Figure 5-9. Pull-Up Resistor Specify Register Format

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

Pull-up resistor specify register group B



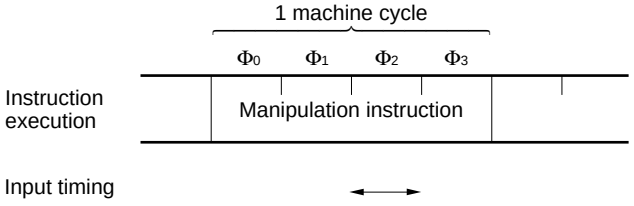
5.1.6 I/O timing of digital I/O port

Figure 5-10 shows the timing by which data is output to the output latch and the timing by which the pin data or the data of the output latch is loaded to the internal bus.

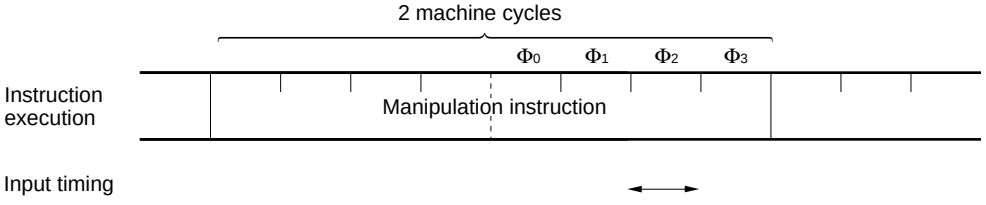
Figure 5-11 shows the ON timing when an on-chip pull-up resistor is connected to a port pin via software.

Figure 5-10. I/O Timing of Digital I/O Port

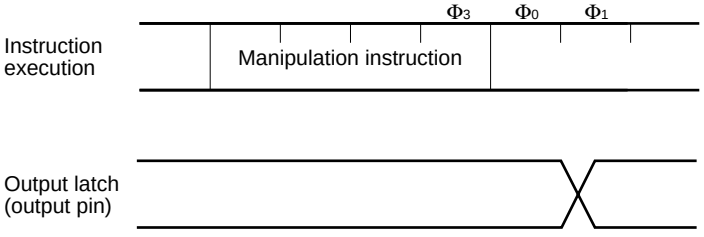
(a) When data is loaded by 1-machine cycle instruction



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



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



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

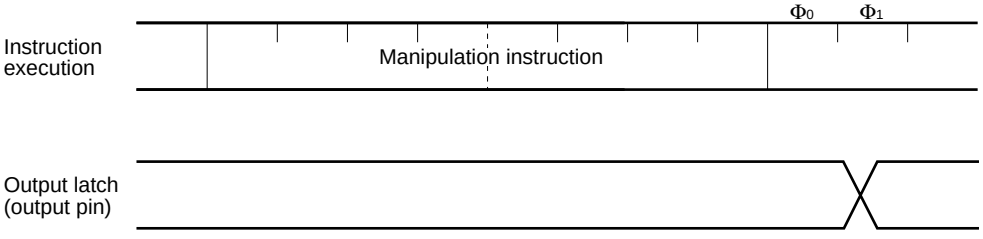
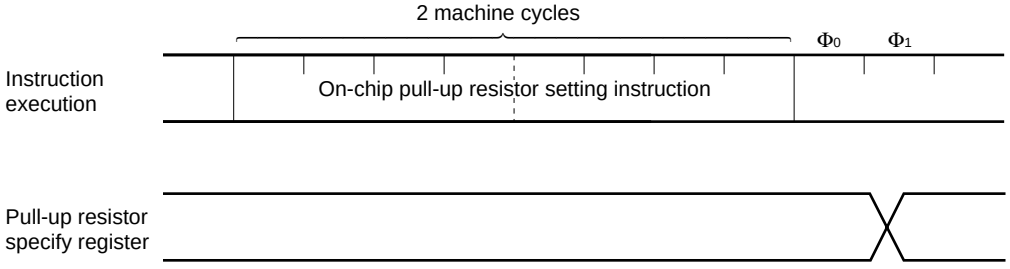


Figure 5-11. ON Timing of On-Chip Pull-up Resistor Connected via Software



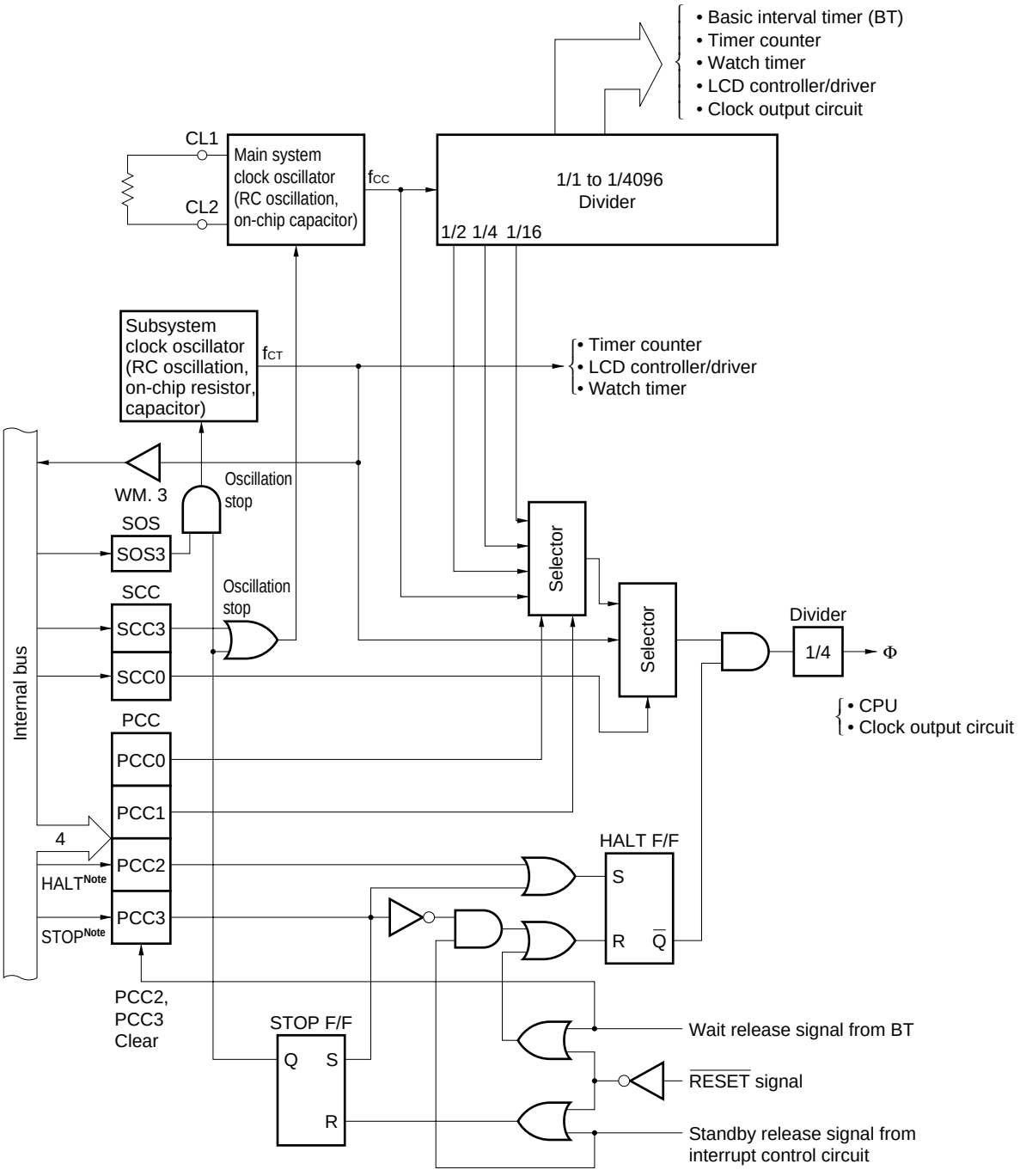
5.2 Clock Generator

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

5.2.1 Clock generator configuration

The configuration of the clock generator is shown in Figure 5-12.

Figure 5-12. Clock Generator Block Diagram



Note Instruction execution

Remarks 1. f_{CC} = Main system clock frequency

2. f_{CT} = Subsystem clock frequency

3. Φ = CPU clock

4. PCC: Processor clock control register

5. SCC: System clock control register

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

5.2.2 Clock generator function and operation

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

- Main system clock f_{CC}
- Subsystem clock f_{CT}
- CPU clock Φ
- Clock to peripheral hardware

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

(a) When the $\overline{\text{RESET}}$ signal is generated, the minimum speed mode of the main system clock (17.8 μs at 3.6-MHz operation) is selected (PCC = 0 and SCC = 0).

(b) When the main system clock is selected, one of four CPU clock frequencies can be selected (1.1 μs, 2.2 μs, 4.4 μs, and 17.8 μs at 3.6-MHz operation) by setting PCC.

(c) The standby mode (STOP or HALT mode) can be used both in the main system clock and subsystem clock.

(d) The subsystem clock is selected by setting SCC, and operation can be performed at a very low-speed and low current consumption (85.1 μs at 47-kHz operation). In this case, the PCC setup value does not affect the CPU clock.

(e) When the subsystem clock is selected, main system clock oscillation can be stopped by setting SCC. Both the HALT mode and STOP mode can also be used. When the STOP instruction is executed after bit 3 of SOS is set to 1, subsystem clock oscillation can be stopped.

(f) The main system clock is divided to generate a clock supplied to peripheral hardware. The subsystem clock can be supplied directly only to the watch timer. Thus, the watch function and the timer counter, LCD controller/driver, and buzzer output function which operate using the watch timer clocks can also continue operation even in the oscillation stop state of the main system clock.

(g) When the subsystem clock is selected, the timer counter, LCD controller/driver, and watch timer can continue normal operation. However, the rest of the hardware operates with the main system clock, therefore it cannot be used when the main system clock stops.

(1) Processor clock control register (PCC)

The PCC is a 4-bit register whereof the low-order 2 bits select the CPU clock Φ and high-order 2 bits control the CPU operating mode (see **Figure 5-13**).

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

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

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

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

The CPU clock can be selected only when the PCC operates with the main system clock. When it operates with a subsystem clock, its low-order 2 bits are invalidated and the frequency is fixed to $f_{CT}/4$. The STOP instruction can be executed both with the main system clock and subsystem clock.

Examples 1. The machine cycle is set to the fastest mode (1.1 μs at 3.6-MHz operation).

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

2. The machine cycle is set to 2.2 μs at 3.6-MHz operation.

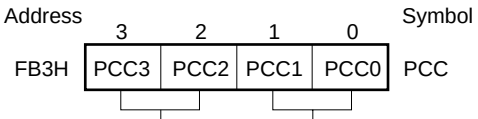
```
SEL     MB15
MOV     A, #0010B
MOV     PCC, A
```

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

```
STOP
NOP
```

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

Figure 5-13. Processor Clock Control Register Format



CPU clock selection bit

★ **(When f_{CC} = 3.6 MHz)**

		SCC3, SCC0 = 00 Values in parentheses are applied when f_{CC} = 3.6 MHz		SCC3, SCC0 = 01 or 11 Values in parentheses are applied when f_{CT} = 47 kHz	
		CPU clock frequency	1 machine cycle	CPU clock frequency	1 machine cycle
0	0	$\Phi = f_{CC}/64$ (56.25 kHz)	17.8 μs	$\Phi = f_{CT}/4$ (11.75 kHz)	85.1 μs
0	1	$\Phi = f_{CC}/16$ (225 kHz)	4.4 μs		
1	0	$\Phi = f_{CC}/8$ (450 kHz)	2.2 μs		
1	1	$\Phi = f_{CC}/4$ (900 kHz)	1.1 μs		

Remarks 1. f_{CC} : Main system clock oscillator output frequency

2. f_{CT} : Subsystem clock oscillator output frequency

CPU operating mode control bits

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

(2) System clock control register (SCC)

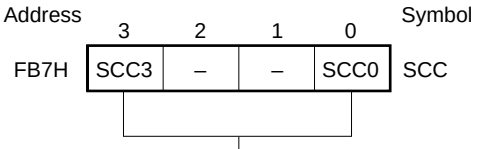
The system clock control register (SCC) is a 4-bit register whose least significant bit is used to select CPU clock Φ, and the most significant bit is used to control (stop) main system clock oscillation (see **Figure 5-14**).

SCC bit 0 and bit 3 exist at the same data memory address, but cannot be changed at the same time. Thus, SCC bit 0 and bit 3 are set by using a bit manipulation instruction. SCC bit 0 and bit 3 can always be operated independently of the MBE contents.

Main system clock oscillation can be stopped by setting SCC bit 3 only during subsystem clock operation. Main system clock oscillation is stopped by using the STOP instruction during main system clock operation.

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

Figure 5-14. System Clock Control Register Format



SCC3	SCC0	CPU clock selection	Main system clock oscillation
0	0	Main system clock	Oscillation is enabled.
0	1	Subsystem clock	
1	0	Setting prohibited	
1	1	Subsystem clock	Oscillation is stopped.

Cautions 1. Changing the system clock requires a maximum of $1/f_{CT}$ time. To stop main system clock oscillation, after changing the subsystem clock, set SCC.3 to 1 after the machine cycle or cycles listed in Table 5-5 had elapsed.

2. Even if oscillation is stopped by setting SCC.3 during main system clock operation, normal STOP mode is not entered.

(3) System clock oscillator

The main system clock oscillator oscillates with the resistor (R) connected to the CL1 and CL2 pins as shown in Figure 5-15. No external clock can be input.

The relation between the main system clock oscillator output frequency (f_{CC}), the resistance (R), and the capacitance (C)[Note] is as follows.

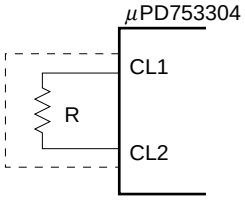
Note 10-pF (typ.) capacitor (C) is incorporated.

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

Caution In some cases, f_{CC} may deviate because of the fluctuation of power supply voltage and temperature.

Figure 5-15. Main System Clock Oscillator External Circuit

RC oscillation



Because the subsystem clock oscillator does not have external pins, the resistor (R) and capacitor (C) cannot be connected (both resistor and capacitor are incorporated). In addition, no external clock can be input.

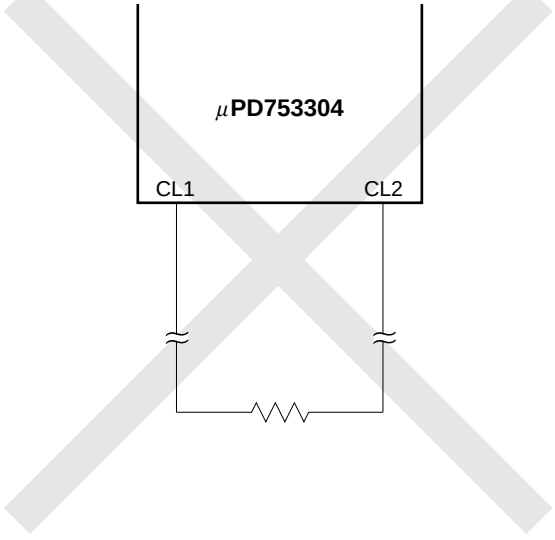
Caution Wire the portion enclosed by broken lines in Figure 5-15 as follows to prevent adverse influence by wiring capacitance when using the main system clock oscillator.

- **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 flows.**
- **Do not extract signals from the oscillator.**

Figure 5-16 shows an example of connecting the oscillator incorrectly.

Figure 5-16. Incorrect Example of Connection (1/2)

(a) Wiring length too long



(b) Crossed signal line

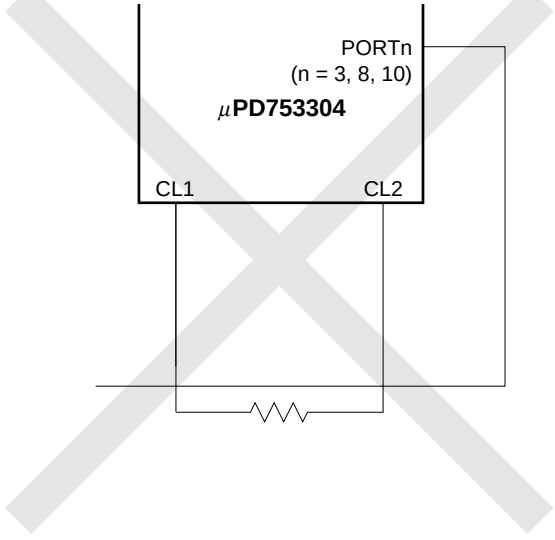
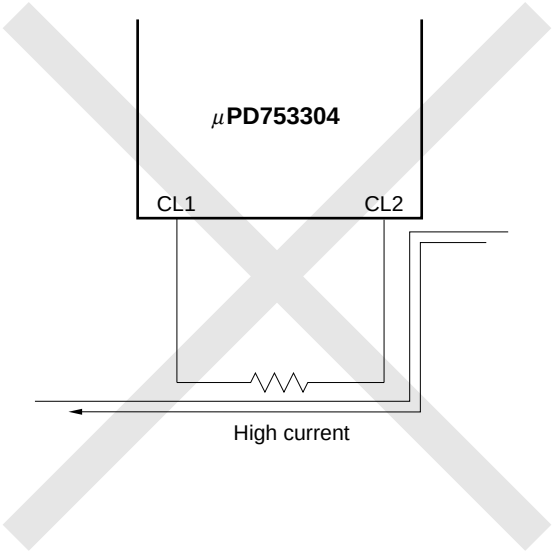
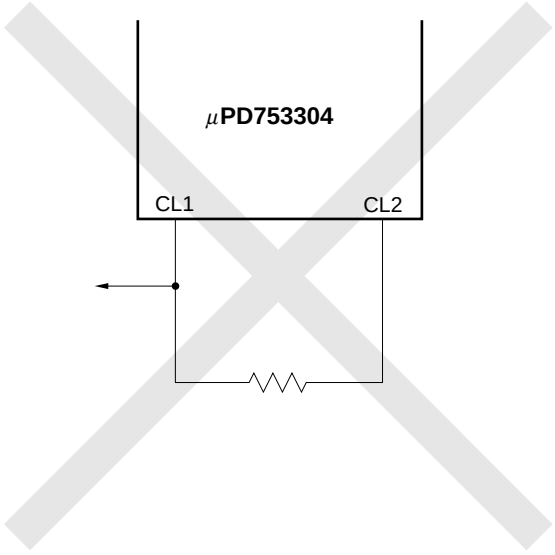


Figure 5-16. Incorrect Example of Connection (2/2)

(c) High alternating current close to signal line



(d) Signal extracted



(4) Divider

The divider divides the main system clock oscillator output (f_{CC}) to generate various clocks.

(5) Sub-oscillator control register (SOS)

SOS is a 4-bit register that controls the subsystem clock oscillation stop using the most significant bit (see Figure 5-17). After setting the SOS bit 3 to 1, the subsystem clock oscillation stops upon the execution of STOP instruction. The STOP instruction can be executed both with the main system clock and subsystem clock. When the $\overline{\text{RESET}}$ signal is generated, SOS is cleared to 0.

Figure 5-17. Sub-oscillator Control Register (SOS) Format

Address	3	2	1	0	Symbol
FCFH	SOS3	—	—	—	SOS

Sub-oscillator stop enable flag

0	Stop disabled
1	Stop by STOP instruction

5.2.3 Setting of system clock and CPU clock

(1) Time required to switch system clock to/from CPU clocks

The system and CPU clocks can be switched by using the low-order two bits of PCC and the least significant bit of SCC. However, this clock switching is not immediately made after the registers are rewritten, and the clock before the clock switching is made is used for operation during given machine cycles. Thus, to stop main system clock oscillation, execute a STOP instruction after the switching time elapses.

★ **Table 5-5. Maximum Time Required to Switch System to/from CPU Clocks**

Setup value before switching			Setup value after switching														
SCC0	PCC1	PCC0	SCC0	PCC1	PCC0	SCC0	PCC1	PCC0	SCC0	PCC1	PCC0	SCC0	PCC1	PCC0	SCC0	PCC1	PCC0
			0	0	0	0	0	1	0	1	0	0	1	1	1	x	x
0	0	0				1 machine cycle			1 machine cycle			1 machine cycle			$\frac{f_{CC}}{64f_{CT}}$ machine cycles (2 machine cycles)		
	0	1	4 machine cycles						4 machine cycles			4 machine cycles			$\frac{f_{CC}}{16f_{CT}}$ machine cycles[Note] (5 machine cycles)		
	1	0	8 machine cycles			8 machine cycles						8 machine cycles			$\frac{f_{CC}}{8f_{CT}}$ machine cycles (10 machine cycles)		
	1	1	16 machine cycles			16 machine cycles			16 machine cycles						$\frac{f_{CC}}{4f_{CT}}$ machine cycles (20 machine cycles)		
1	x	x	1 machine cycle			1 machine cycle[Note]			1 machine cycle			1 machine cycle					

Note Emulation cannot be performed by tools.

Caution The values of f_{CC} and f_{CT} change depending on the environmental temperature of the resonator and the variance of load capacitance characteristics. When f_{CC} is higher than its nominal value or f_{CT} is lower than its nominal value, the machine cycles obtained by the formulae $f_{CC}/64f_{CT}$, $f_{CC}/16f_{CT}$, $f_{CC}/8f_{CT}$, and $f_{CC}/4f_{CT}$ given in the table are larger than those obtained by the nominal values of f_{CC} and f_{CT}.

Thus, when setting a wait time necessary for switching the system clock to/from CPU clock, it must be longer than the machine cycle obtained by the nominal values of f_{CC} and f_{CT}.

Remarks 1. ( ): f_{CC} = 3.6 MHz, f_{CT} = 47 kHz

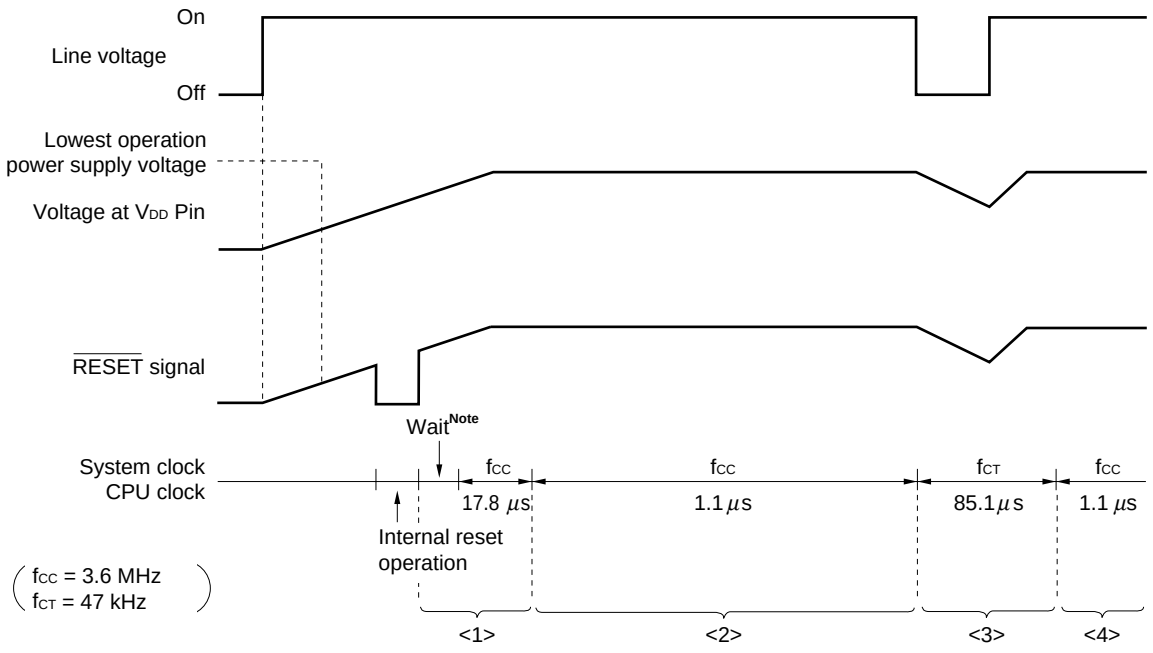
2. x: don't care

3. The CPU clock Φ is supplied to the internal CPU and its inverse (defined to be 1 machine cycle in this manual) is the minimum instruction execution time.

(2) Switching procedure between system clock and CPU clock

The switching procedure between the system clock and CPU clock is explained according to Figure 5-18.

Figure 5-18. Switching between System Clock and CPU Clock



<1> After the wait time[Note] has elapsed for stable oscillation by the $\overline{\text{RESET}}$ signal, the CPU starts operation with the slowest speed (17.8 μs: 3.6-MHz operation) of the main system clock.

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

<3> The failure of the line voltage is detected by an interrupt input, etc. to set bit 0 of the SCC to “1”, and then the CPU starts operation with the subsystem clock. At this time, the subsystem clock must have started oscillation. After the time necessary to switch to the subsystem clock (20 machine cycles) has elapsed, bit 3 of the SCC is set to “1” and then the main system clock stops oscillation.

<4> The recovery of the line voltage is detected by an interrupt, and then bit 3 of the SCC is cleared to “0” to make the main system clock start oscillation. After the time necessary for stable oscillation has elapsed, bit 0 of the SCC is cleared to “0” and the CPU operates in the highest speed.

Note The wait time is fixed to 56/f_{CC} (15.6 at 3.6-MHz operation).

5.2.4 Clock output circuit

(1) Clock output circuit configuration

The configuration of the clock output circuit is shown in Figure 5-19.

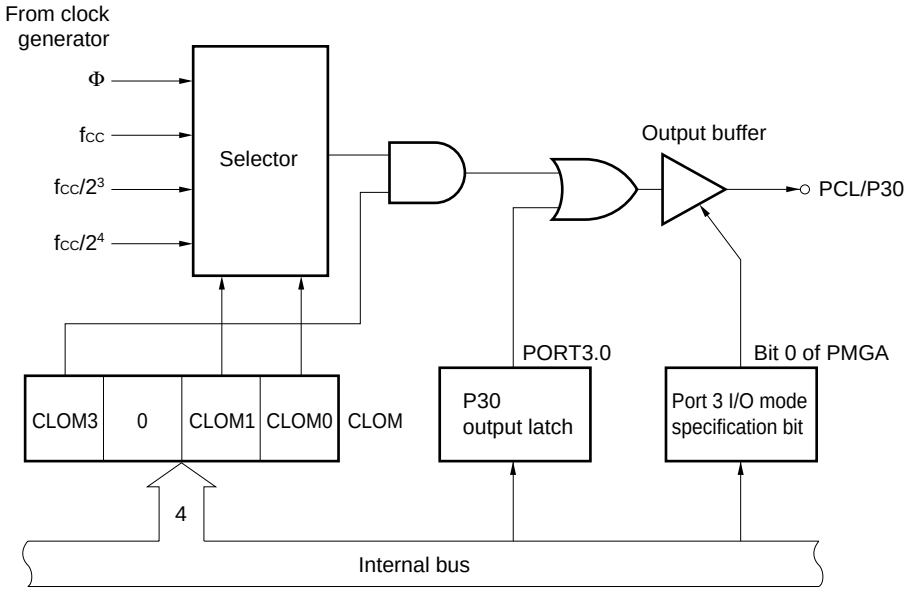
(2) Clock output circuit function

The clock output circuit is provided to output the clock pulses from the PCL/P30 pin to the remote control waveform output applications and peripheral LSIs.

The clock pulses must be output in the following steps.

(a) Select a clock output frequency. Disable the clock output.
(b) Write “0” in the output latch at P30.
(c) Set the I/O mode of the port 3 to output.
(d) Enable the clock output.

Figure 5-19. Clock Output Circuit Block Diagram



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

(3) Clock output mode register (CLOM)

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

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

Example The CPU clock Φ is output from the PCL/P30 pin.

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

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

Figure 5-20. Clock Output Mode Register Format

Address FD0H

3	2	1	0	Symbol
CLOM3	0	CLOM1	CLOM0	CLOM

Clock output frequency select bit

★ **When f_{CC} = 3.6 MHz**

0	0	Φ output[Note] (900 kHz, 450 kHz, 225 kHz, 56.25 kHz)
0	1	$f_{CC}/2^3$ (450 kHz)
1	0	$f_{CC}/2^4$ (225 kHz)
1	1	f_{CC} (3.6 MHz)

Note Φ is the CPU clock selected by the PCC.

Clock output enable/disable bit

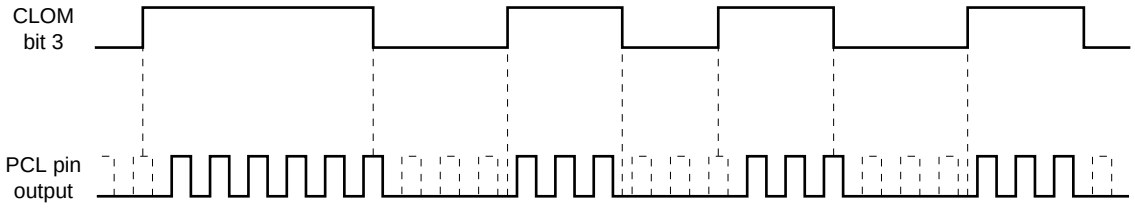
0	Output disabled.
1	Output enabled.

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

(4) Application example of remote control waveform output

The μPD753304 clock output function can be used for remote control waveform output. The carrier frequency of remote control waveform output is selected by the clock frequency select bit of the clock output mode register. The pulse output enable/disable is selected by controlling the clock output enable/disable bit by software. Special attention is paid not to output small-width pulses when switching clock output enable/disable.

Figure 5-21. Application Example of Remote Control Waveform Output



5.3 Basic Interval Timer/Watchdog Timer

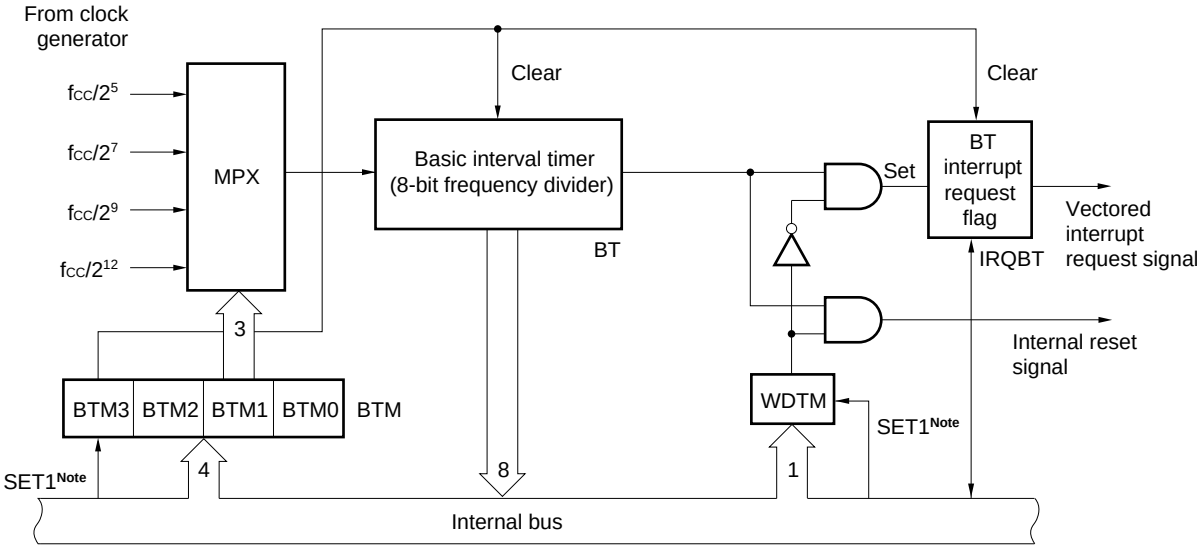
The μPD753304 is provided with the 8-bit basic interval timer/watchdog timer and has the following functions.

(a) Interval timer operation to generate a reference time interrupt
(b) Watchdog timer operation to detect a runaway of program and reset the CPU
(c) Reads the contents of counting

5.3.1 Basic interval timer/watchdog timer configuration

The configuration of the basic interval timer/watchdog timer is shown in Figure 5-22.

Figure 5-22. Basic Interval Timer/Watchdog Timer Block Diagram



Note Instruction execution

5.3.2 Basic interval timer mode register (BTM)

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

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

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

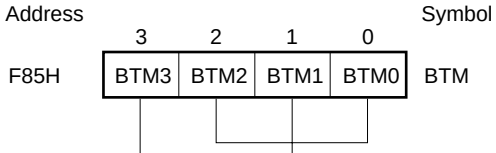
Example The interrupt generation interval is set to 2.28 ms (3.6-MHz operation).

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

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

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

Figure 5-23. Basic Interval Timer Mode Register Format



★ **When f_{CC} = 3.6 MHz**

			Clock input specification	Interrupt interval time
0	0	0	$f_{CC}/2^{12}$ (879 Hz)	$2^{20}/f_{CC}$ (291 ms)
0	1	1	$f_{CC}/2^{9}$ (7.03 kHz)	$2^{17}/f_{CC}$ (36.4 ms)
1	0	1	$f_{CC}/2^{7}$ (28.1 kHz)	$2^{15}/f_{CC}$ (9.10 ms)
1	1	1	$f_{CC}/2^{5}$ (112.5 kHz)	$2^{13}/f_{CC}$ (2.28 ms)
Other than above			Setting prohibited	–

Start control bit of basic interval timer/watchdog timer

The basic interval timer/watchdog timer starts by writing “1” (the counter and interrupt request flag are cleared).

Reset to “0” automatically when the operation starts.

5.3.3 Watchdog timer enable flag (WDTM)

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

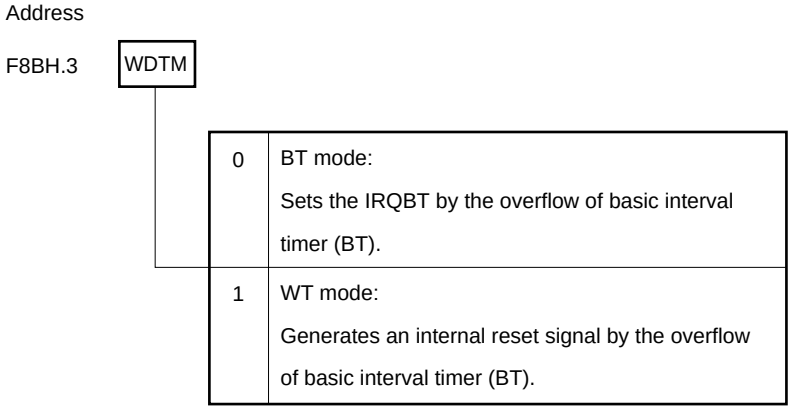
The WDTM is set by a bit manipulation instruction. Once it is set, it cannot be cleared by an instruction.

Example Setting of watchdog timer function

```
SEL     MB15        ; or CLR1 MBE
SET1    WDTM
  ⋮
SET1    BTM.3       ; Bit 3 of BTM is set to “1”.
```

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

Figure 5-24. Watchdog Timer Enable Flag (WDTM) Format



5.3.4 Basic interval timer operations

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

Four interrupt generation intervals can be set by BTM (see **Figure 5-23**).

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

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

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

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

Example Interrupts are generated every 2.28 ms (during 3.6-MHz operation).

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

5.3.5 Watchdog timer operations

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

In the watchdog timer mode, program runaway is detected by utilizing the interval timer wherein the BT overflows. Four intervals can be selected by bits 0-2 of the BTM (see **Figure 5-23**). Select one of them suitable for user’s system. Set an interval and divide the program so that it can be executed in the interval and execute the instruction which clears the BT at the ends of the divided program. If the instruction which clears the BT is not reached within the time set (that is, the program is not executed normally = runaway), the BT overflows and an internal reset signal is generated to forcibly terminate the program. Namely, it indicates a program runaway has occurred and been detected.

Set the watchdog timer with the following procedure (setting of <1> and <2> can be set at the same time).

<1> Set the interval in the BTM.
<2> Set bit 3 of the BTM to “1”. } Initialization
<3> Set WDTM to “1”.
<4> Then, set bit 3 of the BTM to “1” within the interval.

Example The basic interval timer/watchdog timer is used as the watchdog timer with 9.10 ms (during 3.6-MHz operation).

The program is divided into several modules which end within the time set for the BTM (9.10 ms) and the BT is cleared at the ends of the modules. In case a runaway occurs, the BT is not cleared within the time set, therefore it overflows and an internal reset signal is generated.

Initialization :

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

(Then, the bit 3 of the BTM is set to “1” every 9.10 ms.)

Module 1 :

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

Processing is completed within 9.10 ms.

Module 2 :

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

Processing is completed within 9.10 ms

5.3.6 Other functions

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

- **Reads the counting operation**

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

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

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

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

2. The synchronization of the pulses which are input to an INT1 interrupt (detection edge can be selected) is set. The pulse cycle is assumed not to exceed the value set for the BT. The value set for the BTM is assumed to be 9.10 ms or more (during 3.6-MHz operation).
Specify the INT1 detection edge before entering the INT1 interrupt routine.

<INT1 interrupt routine (MBE = 0)>

```
LOOP :  MOV     XA, BT        ; First reading
        MOV     BC, XA        ; Stores data
        MOV     XA, BT        ; Second reading
        SKE     A, C
        BR      LOOP
        MOV     A, X
        SKE     A, B
        BR      LOOP
        SKT     PORT10.3      ; P103 = 1?
        BR      AA            ; NO
        MOV     XA, BC        ; Stores data in the data memory
        MOV     BUFF, XA
        CLR1    FLAG          ; Data exists. Clears the flag.
        DI      IE1
        CLR1    IM1.0
        CLR1    IRQ1
        EI      IE1
        RETI
AA :    MOV     HL, #BUFF
        MOV     A, C
        SUBC    A, @HL
        INCS    L
        MOV     C, A
        MOV     A, B
        SUBC    A, @HL
        MOV     B, A
        MOV     XA, BC
        MOV     BUFF, XA      ; Stores data
        SET1    FLAG          ; Data exists. Sets the flag.
        RETI
```

5.4 Watch Timer

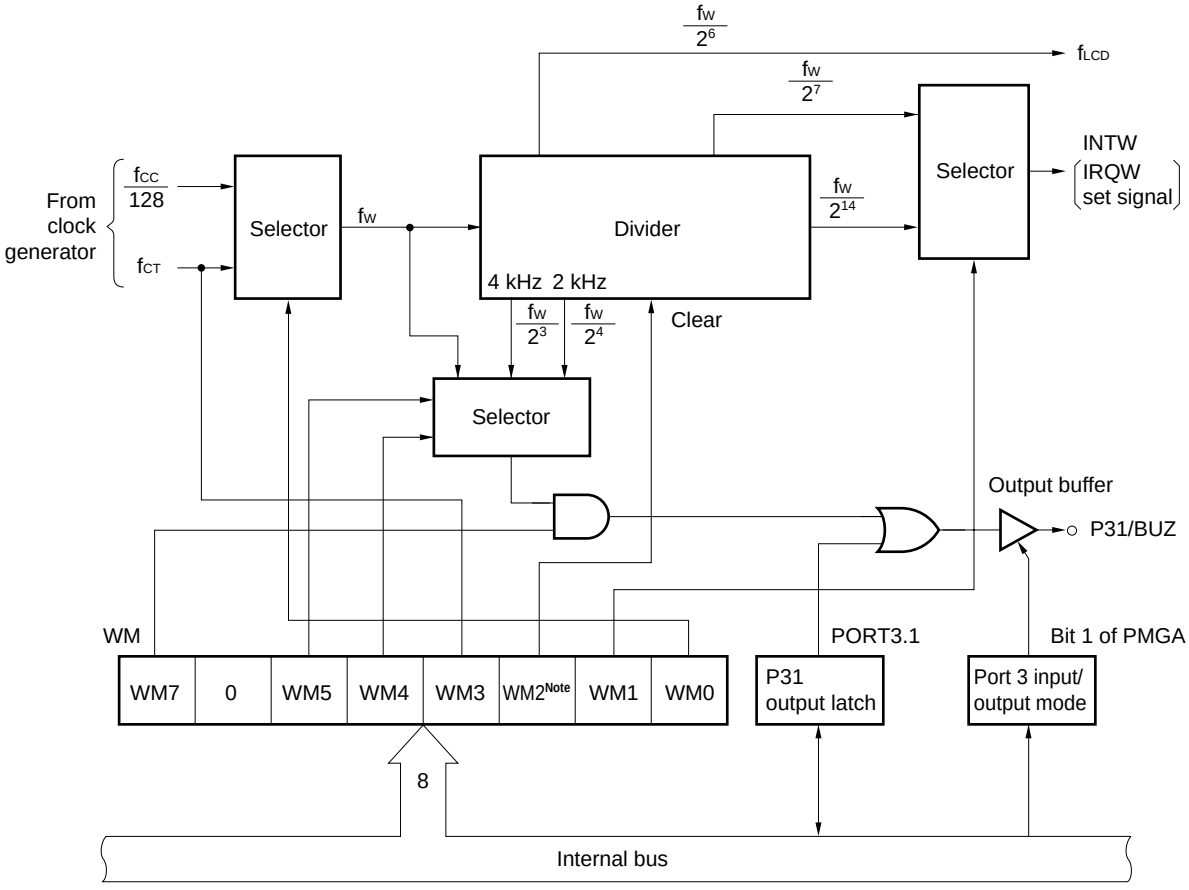
The μPD753304 is provided with a 1-channel of watch timer. This watch timer has the following functions:

(a) Sets the test flag (IRQW) with $f_W/2^{14}$ interval.
The standby mode can be released by the IRQW.
(b) Convenient for program debugging and checking as interval becomes 128 times longer ($f_W/2^7$) with the fast feed mode.
(c) Outputs the frequencies ($f_W/2^4$, $f_W/2^3$, f_W) to the P31/BUZ pin, usable for buzzer and trimming of system clock oscillation frequencies.
(d) Clears the frequency divider to make the clock start with zero seconds.

5.4.1 Configuration of watch timer

Figure 5-25 shows the configuration of the watch timer.

Figure 5-25. Watch Timer Block Diagram



Note Set to 1 when LCD controller driver is used.

5.4.2 Watch mode register

The watch mode register (WM) is an 8-bit register which controls the watch timer. Its format is shown in Figure 5-26.

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

All the bits are cleared to “0” by a $\overline{\text{RESET}}$ signal generation.

Figure 5-26. Watch Mode Register Format

Address	7	6	5	4	3	2	1	0	Symbol
F98H	WM7	0	WM5	WM4	WM3	WM2[Note]	WM1	WM0	WM

BUZ output enable/disable bit

WM7	0	Disables BUZ output.
	1	Enables BUZ output.

BUZ output frequency select bit

WM5	WM4	BUZ output frequency
0	0	$\frac{f_W}{2^4}$
0	1	$\frac{f_W}{2^3}$
1	0	Setting prohibited
1	1	f_W

WM3	Undefined when byte read

Watch operation enable/disable bit

WM2[Note]	0	Stops watch operation (clears the frequency divider, LCD clock supply disable).
	1	Watch operation possible (LCD clock supply enable).

Note Set to 1 when LCD controller/driver is used.

Operation mode select bit

WM1	0	Normal watch mode (Set IRQW with $\frac{f_W}{2^{14}}$).
	1	Fast watch mode (Set IRQW with $\frac{f_W}{2^7}$).

Count clock (f_W) select bit

WM0	0	Frequency divided output of system clock: selects $\frac{f_{CC}}{128}$
	1	Subsystem clock: selects f_{CT}

5.5 Timer Counter

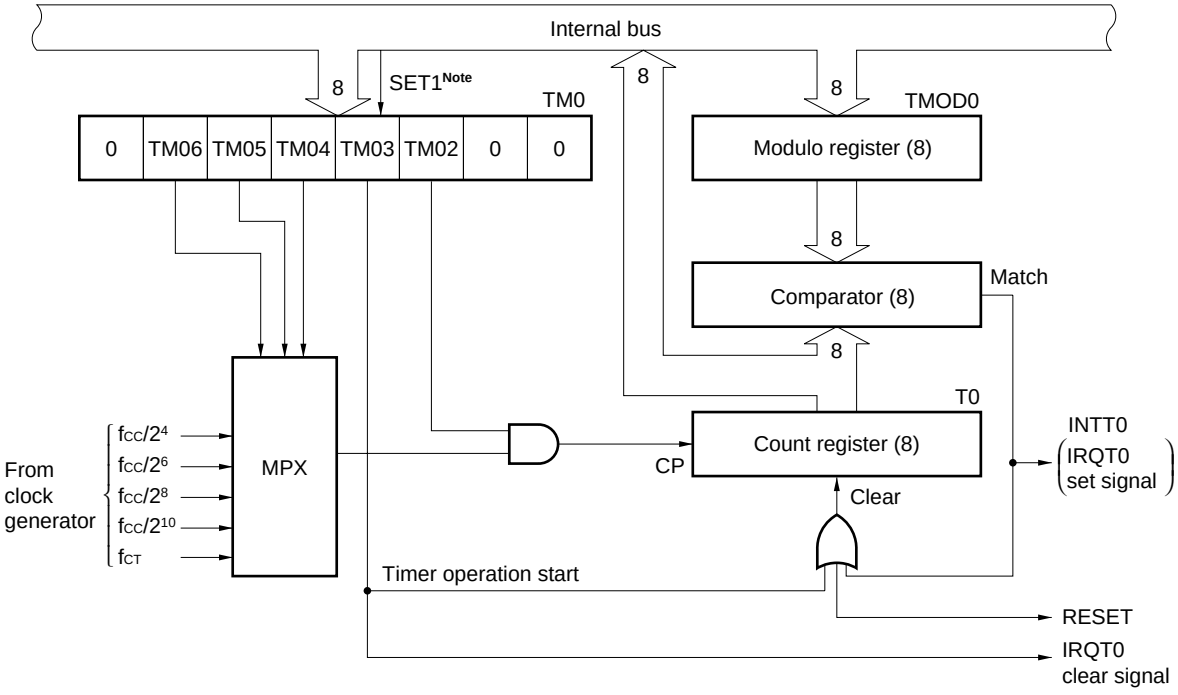
The μPD753304 has one channel of timer counter. The timer counter has the following functions.

(a) Programmable interval timer operation
(b) Reads the count value.

5.5.1 Configuration of timer counter

Figure 5-27 shows the configuration of the timer counter.

Figure 5-27. Timer Counter Block Diagram



Note Instruction execution

Caution When setting data to TM0, be sure to set bits 0, 1, and 7 to 0.

- **Timer counter mode register (TM0)**

  The timer counter mode register (TM0) is an 8-bit register which controls the timer counter.
  Its format is shown in Figure 5-28.
  The timer counter mode register is set by an 8-bit memory manipulation instruction.
  Bit 3 is a timer start indication bit and can be operated bit-wise. It is automatically reset to "0" when the timer operation starts.
  All the bits of the timer counter mode register are cleared to "0" by a $\overline{\text{RESET}}$ signal generation.

Examples 1. Start the timer in the interval timer mode of CP = 3.25 kHz (during 3.6-MHz operation).

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

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

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

Figure 5-28. Timer Counter Mode Register Format

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

Count pulse (CP) selection bit

★ (When f_{CC} = 3.6-MHz operation)

TM06	TM05	TM04	Count pulse (CP)
0	1	0	f_{CT} (47 kHz)
1	0	0	$f_{CC}/2^{10}$ (3.52 kHz)
1	0	1	$f_{CC}/2^{8}$ (14.1 kHz)
1	1	0	$f_{CC}/2^{6}$ (56.25 kHz)
1	1	1	$f_{CC}/2^{4}$ (225 kHz)
Other than above			Setting prohibited

Timer start indication bit

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

Operation mode

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

Caution When setting data to TM0, be sure to set bits 0, 1, and 7 to 0.

5.5.2 8-bit timer counter mode operation

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

(1) Register setting

The following three registers are used in the 8-bit timer counter mode.

- Timer counter mode register (TM0)
- Timer counter count register (T0)
- Timer counter modulo register (TMOD0)

(a) Timer counter mode register (TM0)

When the 8-bit timer counter mode is used, TM0 must be set as follows (for the format of the TM0, see **Figure 5-28**).

The TM0 is manipulated by an 8-bit manipulation instruction. Bit 3 is a timer start indication bit and can be manipulated bit-wise and is automatically cleared to “0” when the timer starts.

The TM0 is cleared to 00H when an internal reset signal is generated.

(2) Timer counter time setting

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

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

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

Once the timer is set, an interrupt request signal (IRQT0) is generated at the intervals set in the timer.
Table 5-6 lists the resolution and maximum allowable setting time (that is, time when FFH is set in the modulo register) for each count pulse to the timer counter.

★ **Table 5-6. Resolution and Maximum Allowable Setting Time**

Mode register			f_{CC} = 3.6-MHz operation	
TM06	TM05	TM04	Resolution	Max. setting time
0	1	0	21.3 μs[Note]	5.45 ms[Note]
1	0	0	284 μs	72.8 ms
1	0	1	71.1 μs	18.2 ms
1	1	0	17.8 μs	4.55 ms
1	1	1	4.4 μs	1.14 ms

Note f_{CT} = 47-kHz operation

(3) 8-bit timer counter mode application

● Use as an interval timer which causes an interrupt to occur at 50 ms intervals (@f_{CC} = 3.6 MHz).

- Set the high-order four bits of the mode register (TM0) to 0100B to select the maximum allowable setting time 72.8 ms.
- Set the low-order four bits of the TM0 to 1100B.
- The value set in the modulo register (TMOD0) is as follows:

$$\frac{50 \text{ ms}}{284\ \mu\text{s}} = 176,\ 176 - 1 = \text{AFH}$$

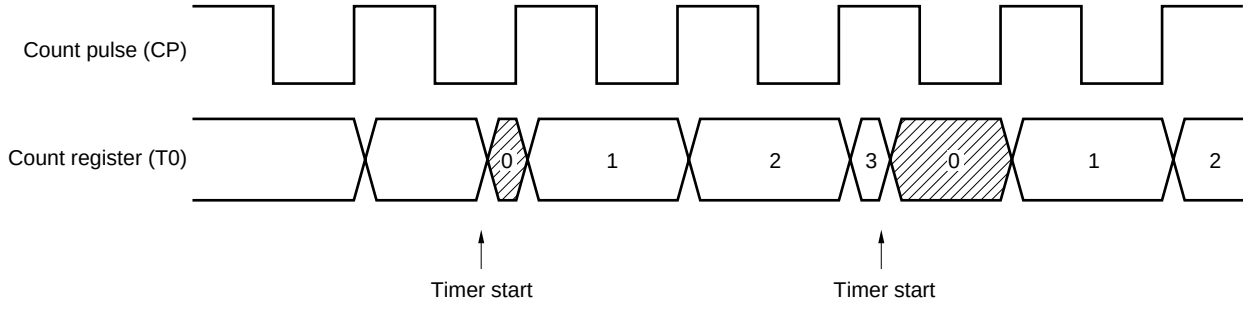
<Program example>

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

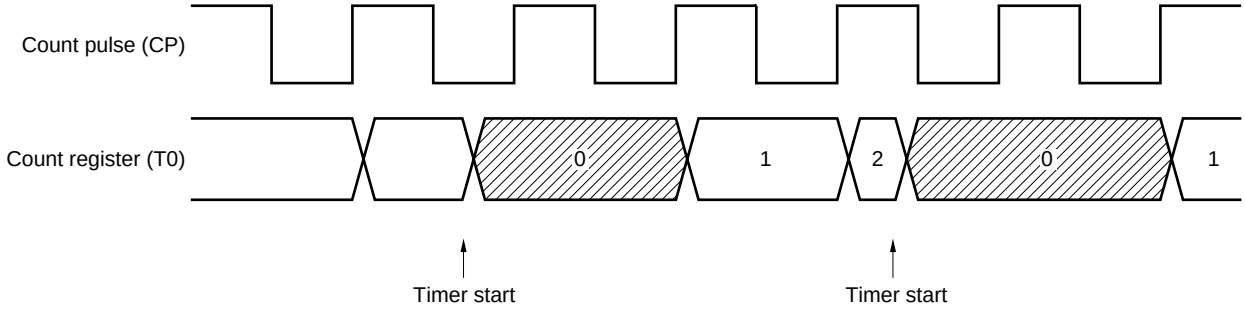
5.5.3 Notes on using timer counter

(1) Error when starting the timer

During the time from the timer start (bit 3 of the TM0 is set to “1”) to the match signal generation, an error of one count pulse (CP) at maximum is produced with respect to the value obtained by the formula: (value set in modulo register + 1) x resolution. This is because the count register T0 is cleared asynchronously with the CP as shown below.



When the frequency of CP is one machine cycle or more, the time from the timer start (bit 3 of the TM0 is set to “1”) to the match signal generation has a discrepancy of two clock pulses at maximum to the value obtained by the formula: (value set in modulo register + 1) x resolution. This is because the T0 is cleared asynchronously with the CP based on the CPU clock as shown below.



(2) Caution on starting the timer

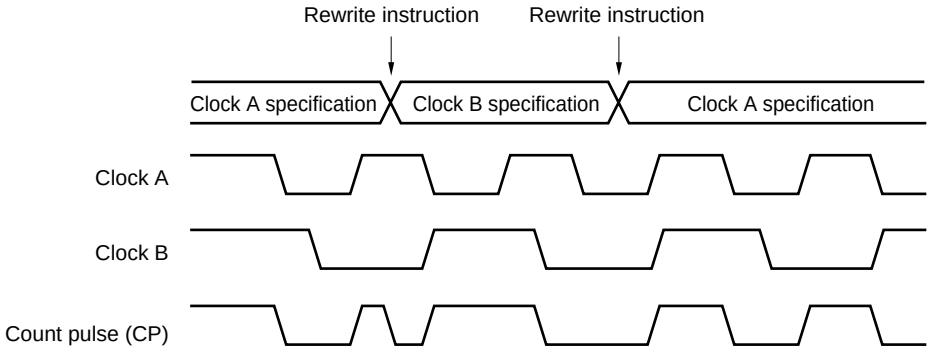
The count register T0 and interrupt request flag IRQT0 are always cleared when the timer starts (bit 3 of the TM0 is set to “1”). On the other hand, when the timer is operating and the IRQT0 is set and the timer starts at the same timing, the IRQT0 may not be able to be cleared. This does not cause trouble when the IRQT0 is used as a vectored interrupt. However, when the IRQT0 is tested, it appears to be set although the timer has started. Therefore, when the timer starts at the timing when the IRQT0 may be set high, the timer must stop (bit 2 of the TM0 is set to “0”) and then restart or the timer start operation must be done twice.

Example Timer start at the timing when the IRQT0 may be set high

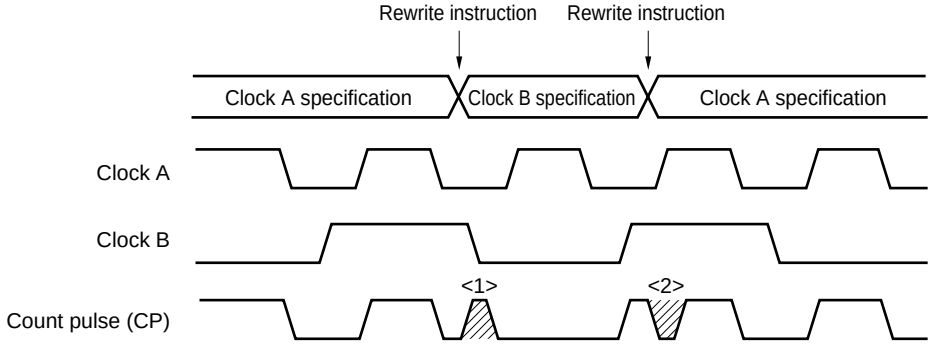
```
SEL     MB15
MOV     XA, #0
MOV     TM0, XA     ; Timer stop
MOV     XA, #4CH
MOV     TM0, XA     ; Restart
or
SEL     MB15
SET1    TM0.3
SET1    TM0.3       ; Restart
```

(3) Caution on changing the count pulse

When the count pulse (CP) is changed by rewriting the timer counter mode register (TM0), the specification for the change is valid immediately after the instruction is executed.

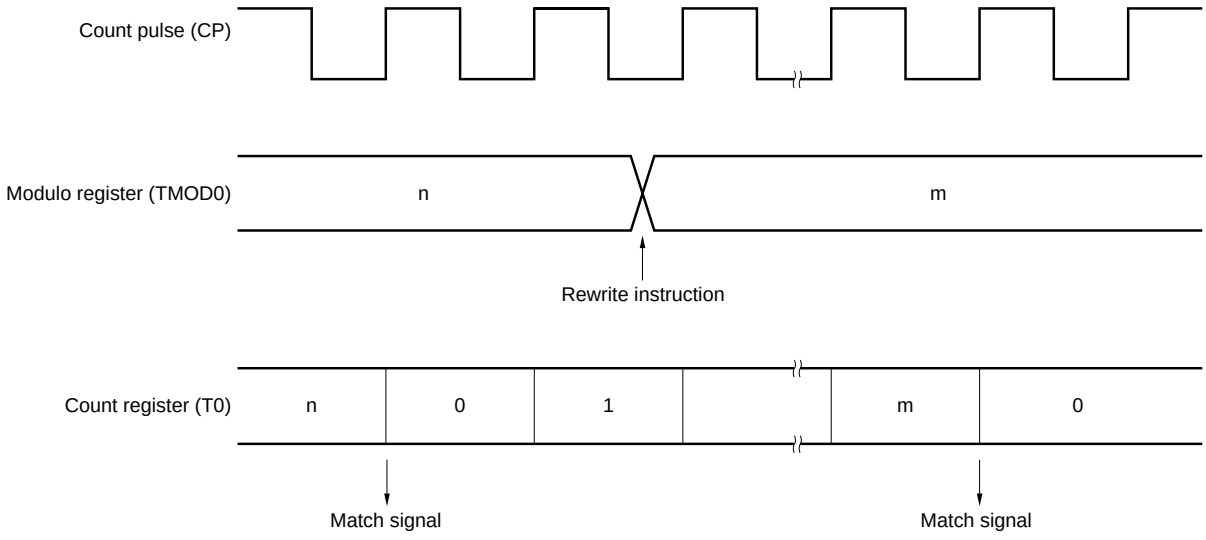


Depending on the combination of clock pulses at the time the CP is changed, whisker-like clock pulses (<1> or <2> in the illustration below) may be produced. In this case, incorrect counting may occur or the contents of count register (T0) may be disrupted. Therefore, when changing the CP, set bit 3 of the TM0 to “1” and restart the timer simultaneously.

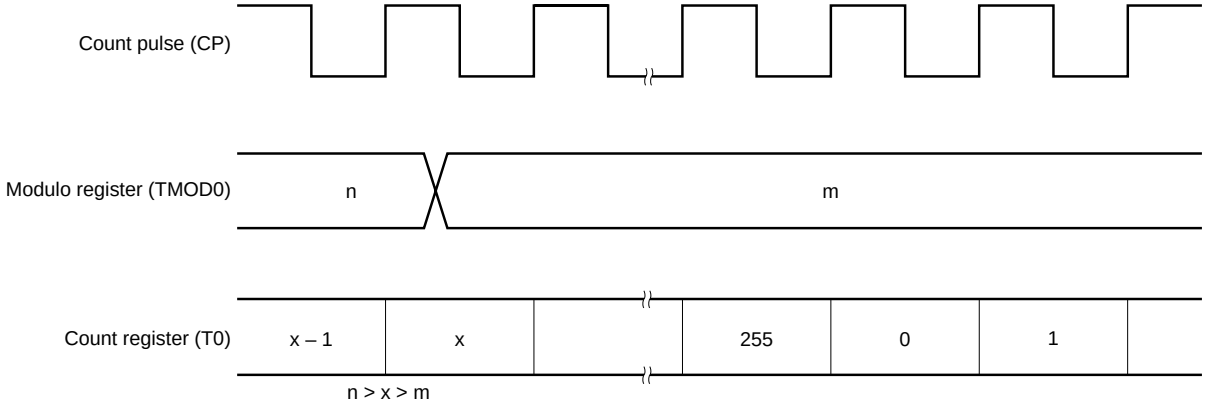


(4) Operation after changing the modulo register

The contents of the modulo register (TMOD0) are rewritten when an 8-bit data memory manipulation instruction is executed.



If the value of the TMOD0 after a change is smaller than the value of the count register (T0), the T0 continues counting and overflows to restart counting from 0. Therefore, if the value (m) after the TMOD0 is changed is smaller than the value (n) before the TMOD0 is changed, the timer must restart after the TMOD0 is changed.



5.6 LCD Controller/Driver

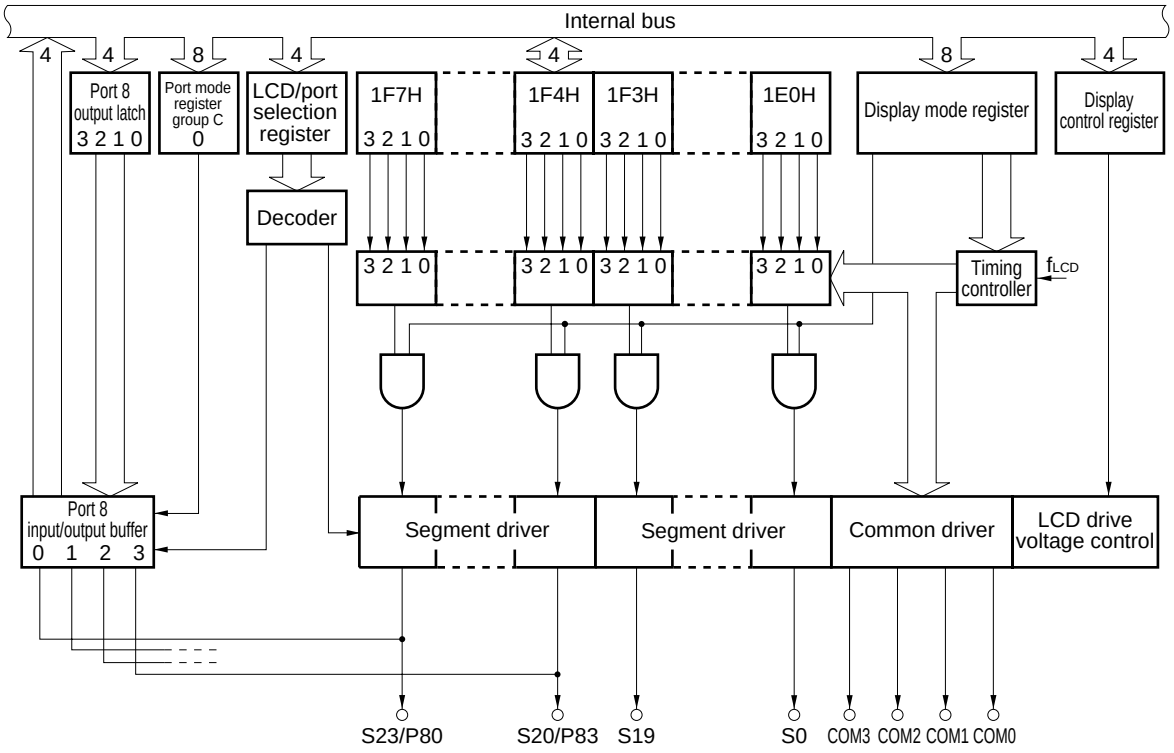
5.6.1 LCD controller/driver configuration

The μPD753304 incorporates a display controller which generates segment and common signals according to the display data memory contents, and incorporates segment and common drivers which can drive the LCD panel directly.

Figure 5-29 shows the LCD controller/driver configuration.

Caution When using the LCD controller/driver, set the bit 2 of the watch mode register WM to "1".

Figure 5-29. LCD Controller/Driver Block Diagram



5.6.2 LCD controller/driver functions

The μPD753304 LCD controller/driver functions are as follows:

(a) Display data memory is read automatically by DMA operation and segment and common signals are generated.
(b) Display mode can be selected from among the following five:
<1> Static
<2> 1/2 duty (time multiplexing by 2), 1/2 bias
<3> 1/3 duty (time multiplexing by 3), 1/2 bias
<4> 1/3 duty (time multiplexing by 3), 1/3 bias
<5> 1/4 duty (time multiplexing by 4), 1/3 bias
(c) A frame frequency can be selected from among four in each display mode.
(d) A maximum of 24 segment signal output pins (S0 to S23) and four common signal output pins (COM0 to COM3).
(e) The segment signal output pins (S20 to S23) can be changed to the I/O ports (PORT8).
(f) Split resistor to supply LCD drive power supply (mask option) is incorporated.
• Various bias methods and LCD drive voltages can be applicable.
• When display is off, current flowing through the split resistor is cut.
(g) It can also operate by using the subsystem clock.
★ (h) LCD display mode can be selected.

Table 5-7 lists the maximum number of picture elements that can be displayed in each display mode.

Table 5-7. Maximum Number of Displayed Picture Elements

Bias method	Time division	Common signals used	Maximum number of picture elements
–	Static	COM0 (COM1, 2, 3)	24 (segment 24 x common 1)Note 1
1/2	2	COM0, 1	48 (segment 24 x common 2)Note 2
	3	COM0, 1, 2	72 (segment 24 x common 3)Note 3
1/3	3		
	4	COM0, 1, 2, 3	96 (segment 24 x common 4)Note 4

Notes 1. 3 digits (8 segment signals/digit) on LCD panel (display).
2. 6 digits (4 segment signals/digit) on LDC panel (display).
3. 8 digits (3 segment signals/digit) on LCD panel (display).
4. 12 digits (2 segment signals/digit) on LCD panel (display).

5.6.3 Display mode register (LCDM)

The display mode register (LCDM) consists of eight bits to specify the display mode, LCD clock, frame frequency, and display output on/off control.

LCDM is set by using an 8-bit memory manipulation instruction. Only bit 3 (LCDM3) can be set and cleared by using bit manipulation instructions.

When the $\overline{\text{RESET}}$ signal is generated, all the bits are cleared to “0”.

Figure 5-30. Display Mode Register Format

Address	7	6	5	4	3	2	1	0	Symbol
F8CH	0	0	LCDM5	LCDM4	LCDM3	LCDM2	LCDM1	LCDM0	LCDM

★ **LCD Clock Selection**

LCDM5	LCDM4	LCDCL[Note]
0	0	$f_W/2^9$ (92 Hz)
0	1	$f_W/2^8$ (184 Hz)
1	0	$f_W/2^7$ (367 Hz)
1	1	$f_W/2^6$ (734 Hz)

Note LCDCL is supplied only when the watch timer operates. To use the LCD controller driver, bit 2 of watch mode register WM should be set to 1.

Display Mode Selection

LCDM3	LCDM2	LCDM1	LCDM0	Time division value	Bias method
0	x	x	x	Display off[Note]	
1	0	0	0	4	1/3
1	0	0	1	3	1/3
1	0	1	0	2	1/2
1	0	1	1	3	1/2
1	1	0	0	Static	
Other than above				Setting prohibited	

Note All segment signals are unselected.

★ **Frame Frequency (Hz)**

LCDCL / Display duty	$f_W/2^9$ (92 Hz)	$f_W/2^8$ (184 Hz)	$f_W/2^7$ (367 Hz)	$f_W/2^6$ (734 Hz)
Static	92	184	367	734
1/2	46	92	184	367
1/3	31	61	122	245
1/4	23	46	92	184

When f_W = 47 kHz

f_W : Input clock to watch timer ($f_{CC}/128$ or f_{CT})

5.6.4 Display control register (LCDC)

The display control register controls LCD drive as follows.

- Enables/disables the common and segment outputs.
- Cuts the current flowing through the split resistors for the LCD driving power supply.

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

When the $\overline{\text{RESET}}$ signal is generated, all the bits of the display control register are cleared to “0”.

Figure 5-31. Display Control Register Format

Address	3	2	1	0	Symbol
F8EH	0	0	0	LCDC0	LCDC

Display output status by LCDC0 and LCDM3

LCDC0	0	1	
LCDM3	x	0	1
COM0 to COM3	Outputs “L” (display off).	Outputs a common signal corresponding to the display mode.	Outputs a common signal corresponding to the display mode.
S0 to S19	Outputs “L” (display off).	Outputs a segment signal corresponding to the display mode (outputs an unselected level, display off).	Outputs a segment signal corresponding to the display mode (display on).
S20 to S23 segments specification pin			
S20 to S23 ports specification pin	I/O ports. Whether the port is used as an input or output port depends on the specification of the port mode register group C (PMGC).		
Power supply for the split resistors	Off	On	On

5.6.5 LCD/port selection register (LPS)

The LCD/port selection register (LPS) switches the segment signal outputs (S20 to S23) to the input/output port.

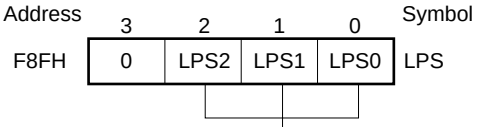
Of segment signal outputs, S20 to S23 are also used as PORT8. Four outputs each are switched together as one unit.

By setting the LPS value to “0000”, S20 to S23 can be switched to input/output port (PORT8).

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

All LPS bits are cleared to “0” by generating the $\overline{\text{RESET}}$ signal.

Figure 5-32. LCD/Port Selection Register Format



			Usable pin	
			Segment output pin	Port pin
0	0	0	—	P83 to P80
0	1	0	S20 to S23	—
Other than above			Setting prohibited	

Cautions 1. All LPS bits are cleared to “0” during resetting, and the LCD cannot be used. Be sure to set the LPS value to 0010 when using the LCD.

2. Be sure to set bit 3 of LPS to “0”.

3. If the port pins are specified as segment signal outputs by using LPS, they do not enter the floating state even if they are set to the input mode through the port mode register group C. When an input instruction is executed for the port specified as a segment signal output, the content of the output latch will be input.

5.6.6 Display data memory

The display data memory is mapped in 1E0H to 1F7H and is write only.

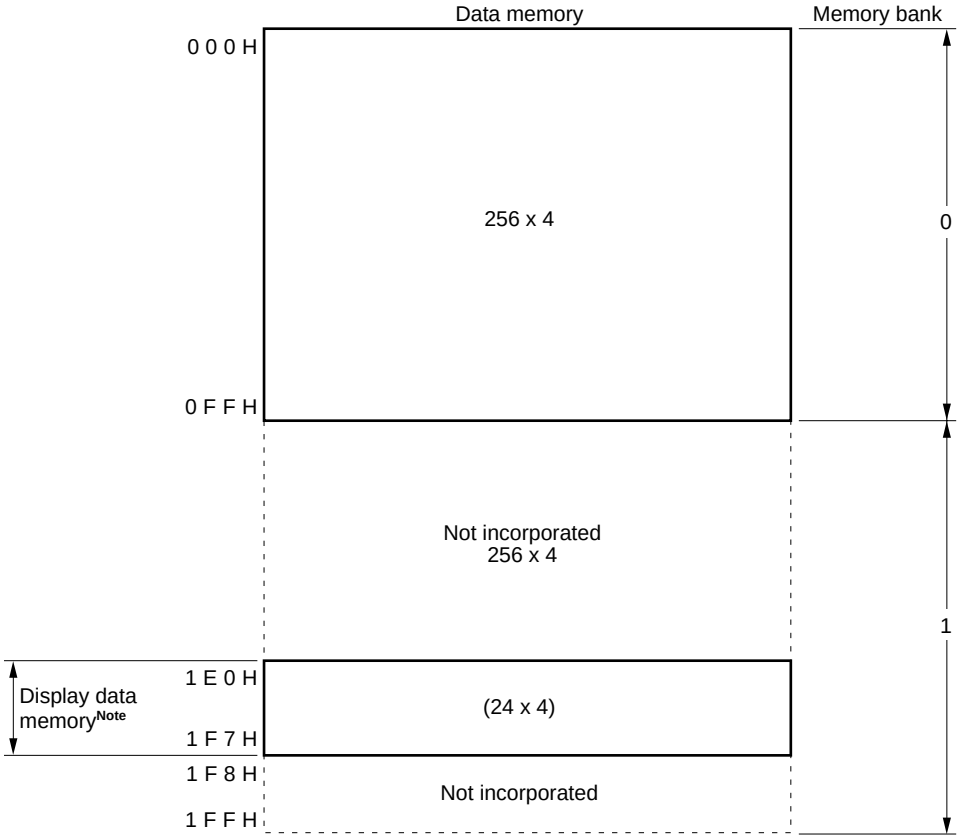
The display data memory is a memory area read by the LCD controller/driver, which performs DMA operation independently of CPU operation. The LCD controller controls the segment signals according to data in the display data memory.

The area not used for LCD display or port cannot be used for normal data memory.

The display data memory is manipulated in 1- or 4-bit units. It cannot be manipulated in 8-bit units.

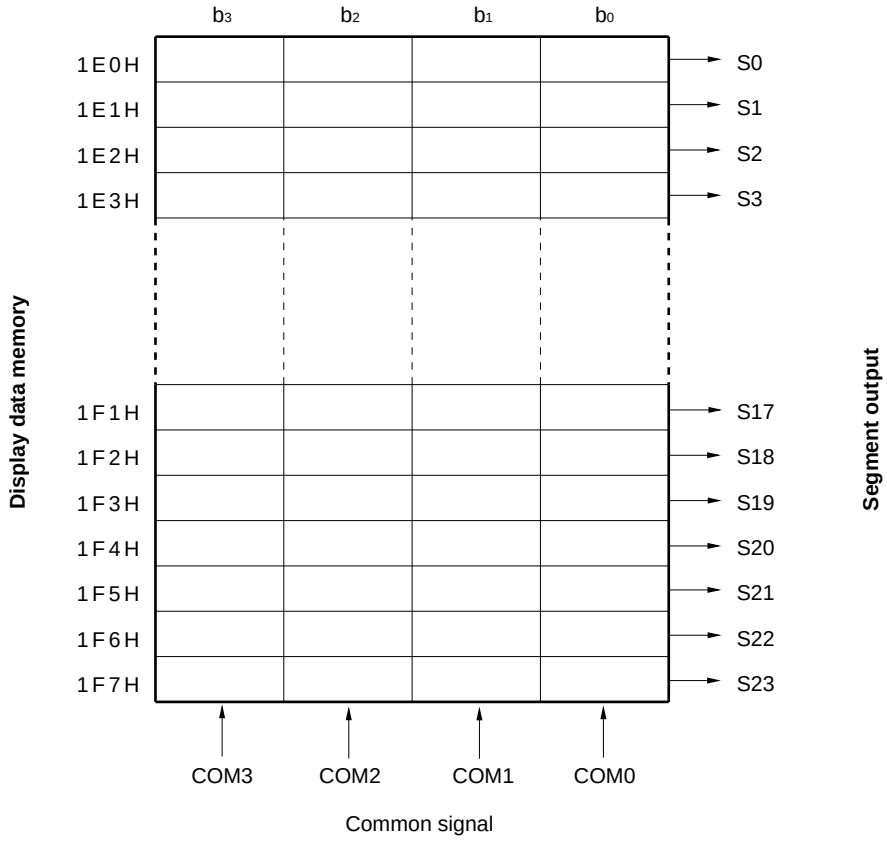
Figure 5-34 shows the relationship between the display data memory bits and segment output.

Figure 5-33. Data Memory Map



Note Write only

Figure 5-34. Relationship between Display Data Memory and Common Segments



5.6.7 Common signal and segment signal

The individual picture elements of the LCD panel light up when the potential difference between the corresponding common signal and segment signal is greater than a predetermined voltage (LCD driving voltage V_{LCD}). When the potential difference goes lower than the V_{LCD} or is zero, the individual picture elements go out.

The LCD panel is degraded when a DC voltage is applied for the common signal and segment signal, therefore it is driven by an AC voltage.

(1) Common signal

The common signal is a selection timing in a sequence shown in Table 5-8 in accordance with the time division number which is set and repeats with that cycle. In the static mode, the same signal is output to the COM0 to COM3. When the time division number is 2, COM2 pin and COM3 pin must be open. When the time division number is 3, the COM3 pin must be open.

Table 5-8. COM Signal

COM signal / Time division number	COM0	COM1	COM2	COM3
Static				
2			Open	Open
3				Open
4				

(2) Segment signal

There are 24 segment signals corresponding to the 24 locations in the display data memory (1E0H-1F7H) of the data memory. Each location's bits 0, 1, 2, and 3 are automatically read out in synchronization with the selection timings of COM0, COM1, COM2, and COM3, respectively. When the contents of each bit is 1, it is converted to the selection voltage; and if they are 0, it is converted to the non-selection voltage and then output via the segment pins (S0 to S23).

As stated above, what combination of LCD panel's front panel electrodes (corresponding to the segment signals) and rear panel electrodes (corresponding to the common signals) forms a display pattern on the display data memory must be checked and then the bit data which corresponds one-to-one to the pattern to be displayed must be written.

Though bits 1/2/3 in the display data memory in the static method, bits 2/3 in the division number 2 method, and bit 3 in the division number 3 are not accessed, they cannot be used for purposes other than display.

(3) Common and segment signal output waveforms

Tables 5-9 to 5-11 list voltages output to the common and segment signals. $+V_{LCD}/-V_{LCD}$ on voltage is applied only when both signals become selection voltages; otherwise, the off voltage is applied.

Table 5-9. LCD Drive Voltage (Static)

Segment signal Sn / Common signal COM0	Selection	Non-selection
	V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
V_{SS}/V_{LC0}	$+V_{LCD}/-V_{LCD}$	0 V/0 V

Table 5-10. LCD Drive Voltage (1/2 Bias Method)

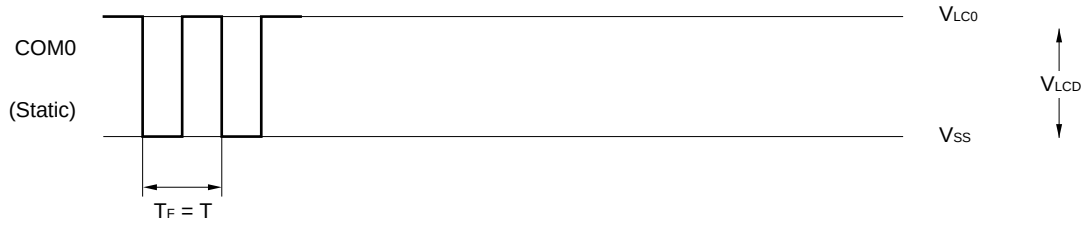
Segment signal Sn / Common signal COMm		Selection	Non-selection
		V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
Selection	V_{SS}/V_{LC0}	$\boxed{+V_{LCD}/-V_{LCD}}$	0 V/0 V
Non-selection	$V_{LC1} = V_{LC2}$	$+\frac{1}{2}V_{LCD}/-\frac{1}{2}V_{LCD}$	$-\frac{1}{2}V_{LCD}/+\frac{1}{2}V_{LCD}$

Table 5-11. LCD Drive Voltage (1/3 Bias Method)

Segment signal Sn / Common signal COMm		Selection	Non-selection
		V_{LC0}/V_{SS}	V_{LC2}/V_{LC1}
Selection	V_{SS}/V_{LC0}	$\boxed{+V_{LCD}/-V_{LCD}}$	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$
Non-selection	V_{LC1}/V_{LC2}	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$

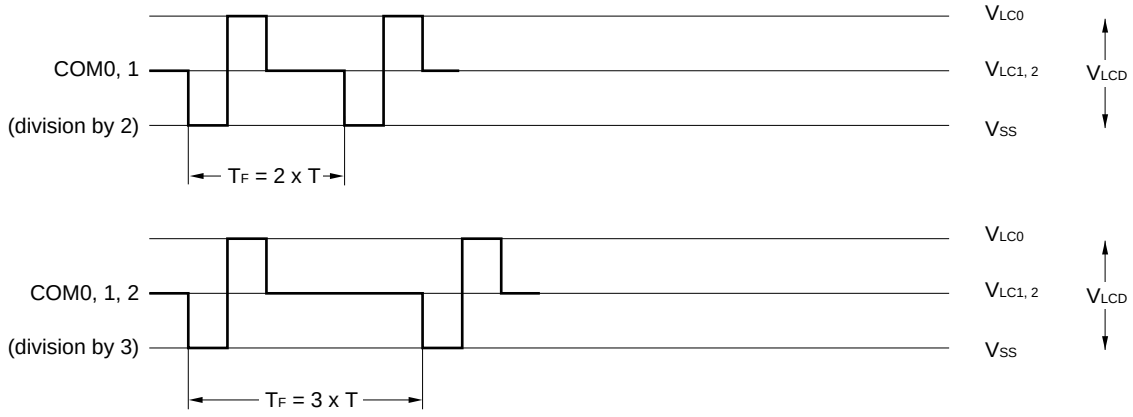
Figures 5-35 to 5-37 show the common signal waveforms. Figure 5-38 shows the common and segment signal electric potentials and phases.

Figure 5-35. Common Signal Waveform (Static)



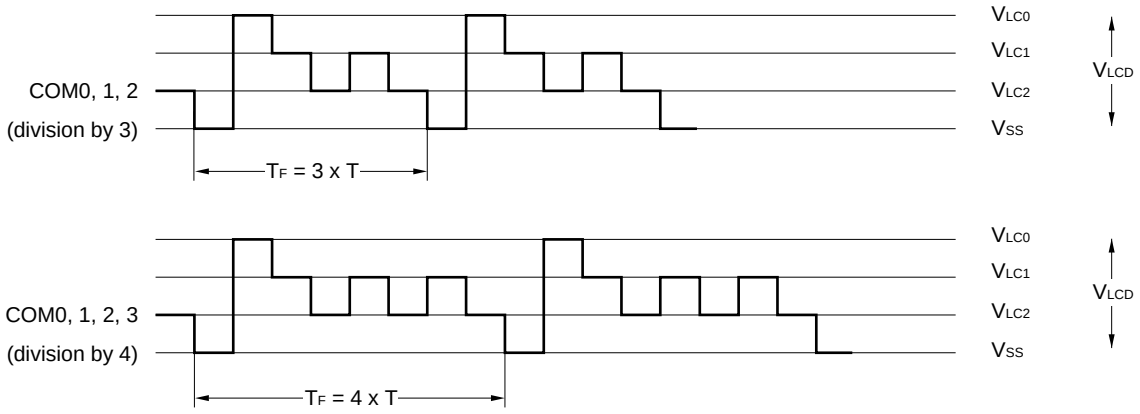
T : One cycle of LCDCL
T_F : Frame cycle

Figure 5-36. Common Signal Waveform (1/2 Bias Method)



T : One cycle of LCDCL
T_F : Frame cycle

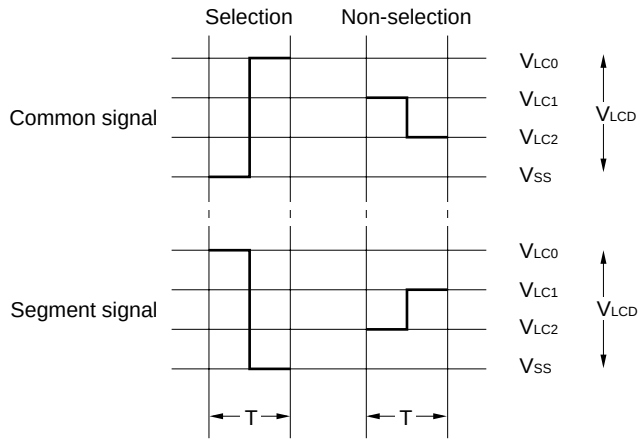
Figure 5-37. Common Signal Waveform (1/3 Bias Method)



T : One cycle of LCDCL
T_F : Frame cycle

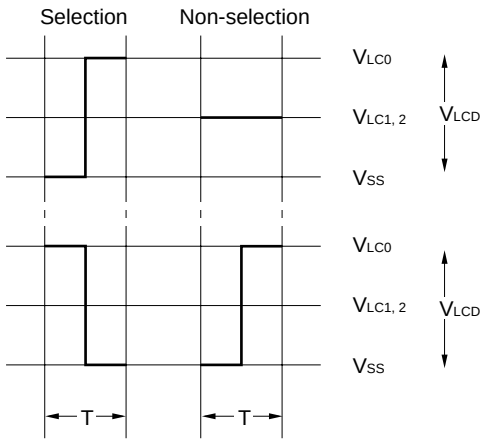
Figure 5-38. Common Signal and Segment Signal Electric Potentials and Phases

(a) 1/3 bias method

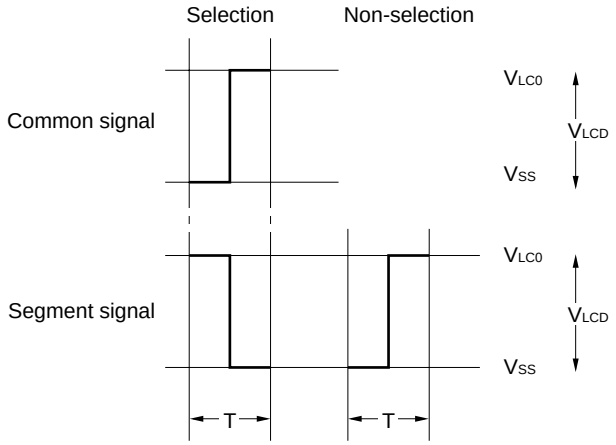


T : One cycle of LCDCL

(b) 1/2 bias method



(c) Static display mode



5.6.8 Supply of LCD drive power supply V_{LC0}, V_{LC1}, and V_{LC2}

Though the μPD753304 incorporates a split resistor for the LCD drive power supply, the V_{LC0} to V_{LC2} pins (LCD drive power supply) and BIAS pin (external split resistor cut pin) are not incorporated as external pins. The following three mask options can be selected to support each display mode.

★ (1) Static mode (Short between BIAS and V_{LC0}, Open between V_{LC0} and V_{LC1})

(2) 1/2 bias mode (Short between BIAS and V_{LC0}, Short between V_{LC0} and V_{LC1})

(3) 1/3 bias mode (Short between BIAS and V_{LC0})

Table 5-12 lists proper LCD drive power supply values based on the static, 1/2, and 1/3 bias methods.

Table 5-12. LCD Drive Power Supply Values

Bias method / LCD drive power	No bias (static mode)	1/2	1/3
V_{LC0}	V_{LCD} (= V_{DD})	V_{LCD}	V_{LCD}
V_{LC1}	0 V	$1/2V_{LCD}$	$2/3V_{LCD}$
V_{LC2}			$1/3V_{LCD}$
V_{SS}		0 V	0 V

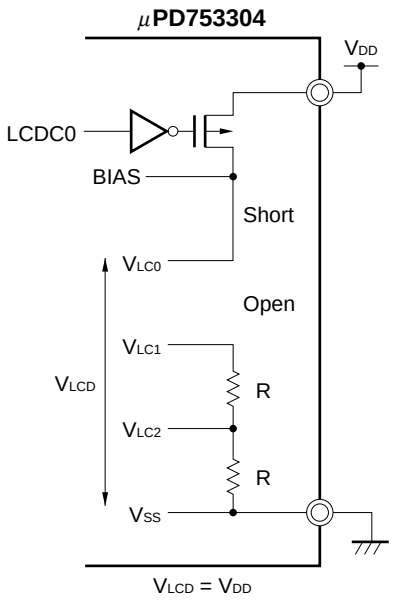
Figure 5-39, (a) to (c) show LCD drive power supply examples according to Table 5-12.

The current flow through the split resistor can also be cut by clearing display control register bit 0 (LCDC0) to 0.

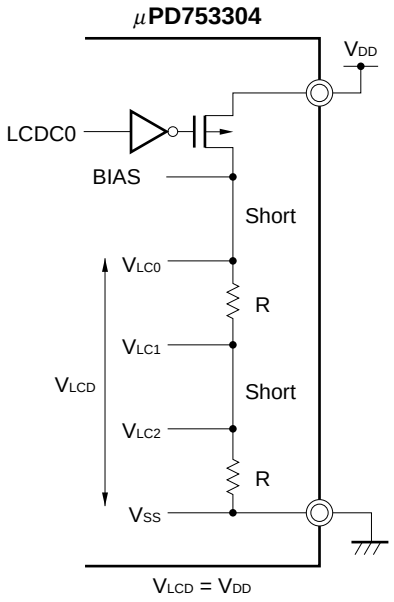
This LCD power on/off control is also useful to prevent DC voltage from being applied to the LCD when the LCD clock (if the system clock is selected) is stopped by execution of a STOP instruction, when the watch timer operates using the main system clock. That is, the display control register bit 0 (LCDC0) is cleared to 0 and all LCD drive power sources are placed in the same potential V_{SS} immediately before the STOP instruction is executed, thereby suppressing the potential difference between the LCD electrodes even if the LCD clock is stopped. When the watch timer operates by using the subsystem clock, LCD display can be continued.

★ **Figure 5-39. LCD Drive Power Supply Connection Examples**

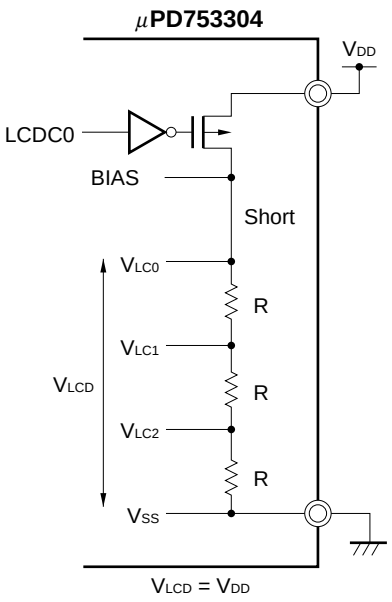
(a) Static display mode
(Example of V_{DD} = 5 V, V_{LCD} = 5 V)



(b) 1/2 bias method
(Example of V_{DD} = 5 V, V_{LCD} = 5 V)



(c) 1/3 bias method
(Example of V_{DD} = 5 V, V_{LCD} = 5 V)



5.6.9 Display mode

(1) Static display example

Figure 5-41 shows connection of a static 3-digit LCD panel having the display pattern shown in Figure 5-40, the μPD753304 segment signals (S0 to S23), and the common signal (COM0). In this example, "12.3" is displayed. The contents of the display data memory (addresses 1E0H to 1F7H) correspond to the display pattern.

Here, "2." at the second digit position is taken as an example. It is necessary to output the selection and non-selection voltages as shown in Table 5-13 to the S8 to S15 pins at the COM0 common signal timing according to the display pattern shown in Figure 5-40.

Table 5-13. S16 to S23 Pin Selection and Non-selection Voltage (Static Display Example)

Segment / Common	S8	S9	S10	S11	S12	S13	S14	S15
COM0	Selection	Non-selection	Selection	Selection	Non-selection	Selection	Selection	Selection

This shows that the bit 0 of display data memory addresses 1E8H to 1EFH corresponding to S8 to S15 needs to be set to 10110111.

Figure 5-42 shows the S11, S12, and COM0 LCD drive waveforms. This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level being generated when S11 is the selection voltage at the COM0 selection timing.

Since the same waveform as COM0 is output to COM1 to COM3, the drive capability can be increased by connecting COM0, COM1, COM2, and COM3.

Figure 5-40. Static Mode LCD Display Pattern and Electrode Connection

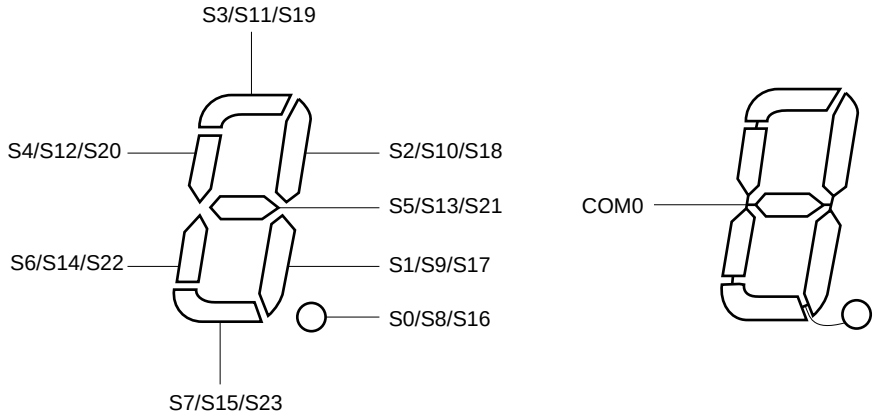


Figure 5-41. Static LCD Panel Connection Example

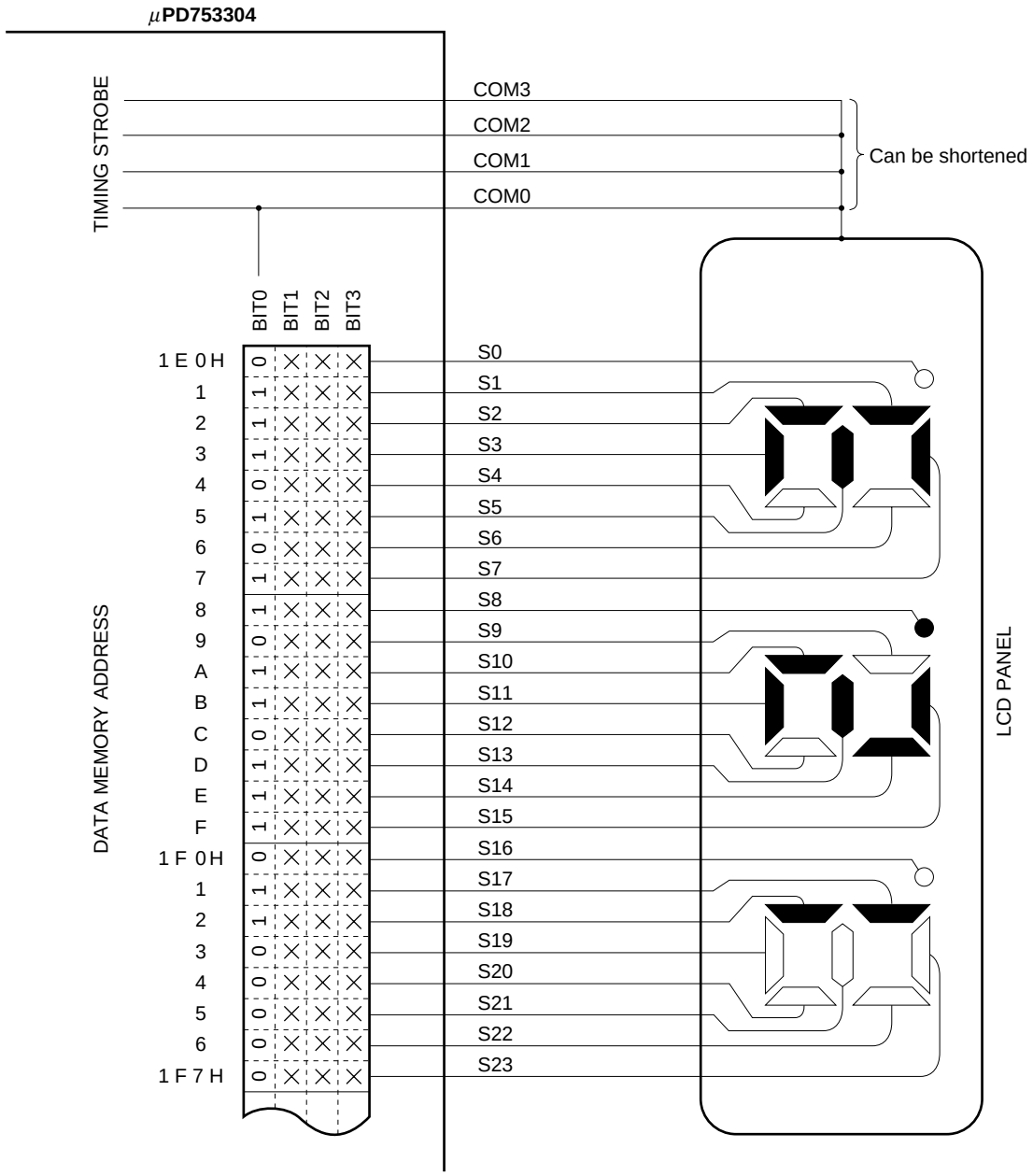
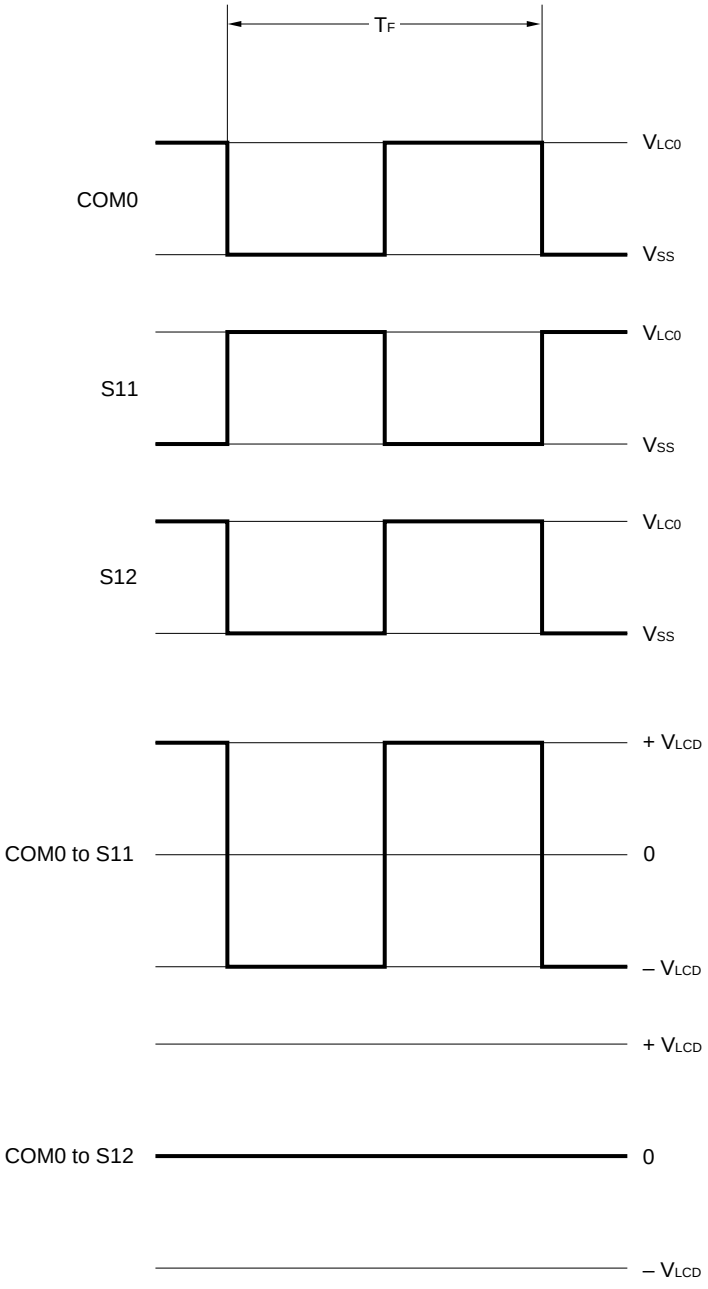


Figure 5-42. Static LCD Drive Waveform Example



(2) Division by 2 display example

Figure 5-44 shows connection of a division by 2 mode 6-digit LCD panel having the display pattern shown in Figure 5-43, the μPD753304 segment signals (S0 to S23), and the common signals (COM0 and COM1). In the example, "1234.56" is displayed. The contents of the display data memory (addresses 1E0H to 1F7H) correspond to the display pattern.

Here, "3." at the fourth digit position is taken as an example. It is necessary to output the selection and non-selection voltages as shown in Table 5-14 to the S12 to S15 pins at the COM0 and COM1 common signal timings according to the display pattern shown in Figure 5-43.

Table 5-14. S12 to S15 Pin Selection and Non-selection Voltage (Division by 2 Display Example)

Segment / Common	S12	S13	S14	S15
COM0	Selection	Selection	Non-selection	Non-selection
COM1	Non-selection	Selection	Selection	Selection

This shows that for example, the display data memory address 1EFH corresponding to S15 needs to be set to xx10.

Figure 5-45 shows an LCD drive waveform example among S15 and common signals. This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level being generated when S15 is the selection voltage at the COM1 selection timing.

Figure 5-43. Division by 2 Mode LCD Display Pattern and Electrode Connection

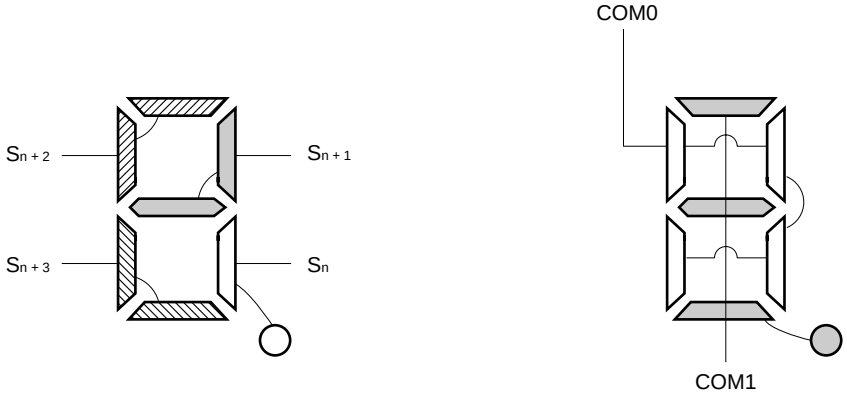


Figure 5-44. Division by 2 LCD Panel Connection Example

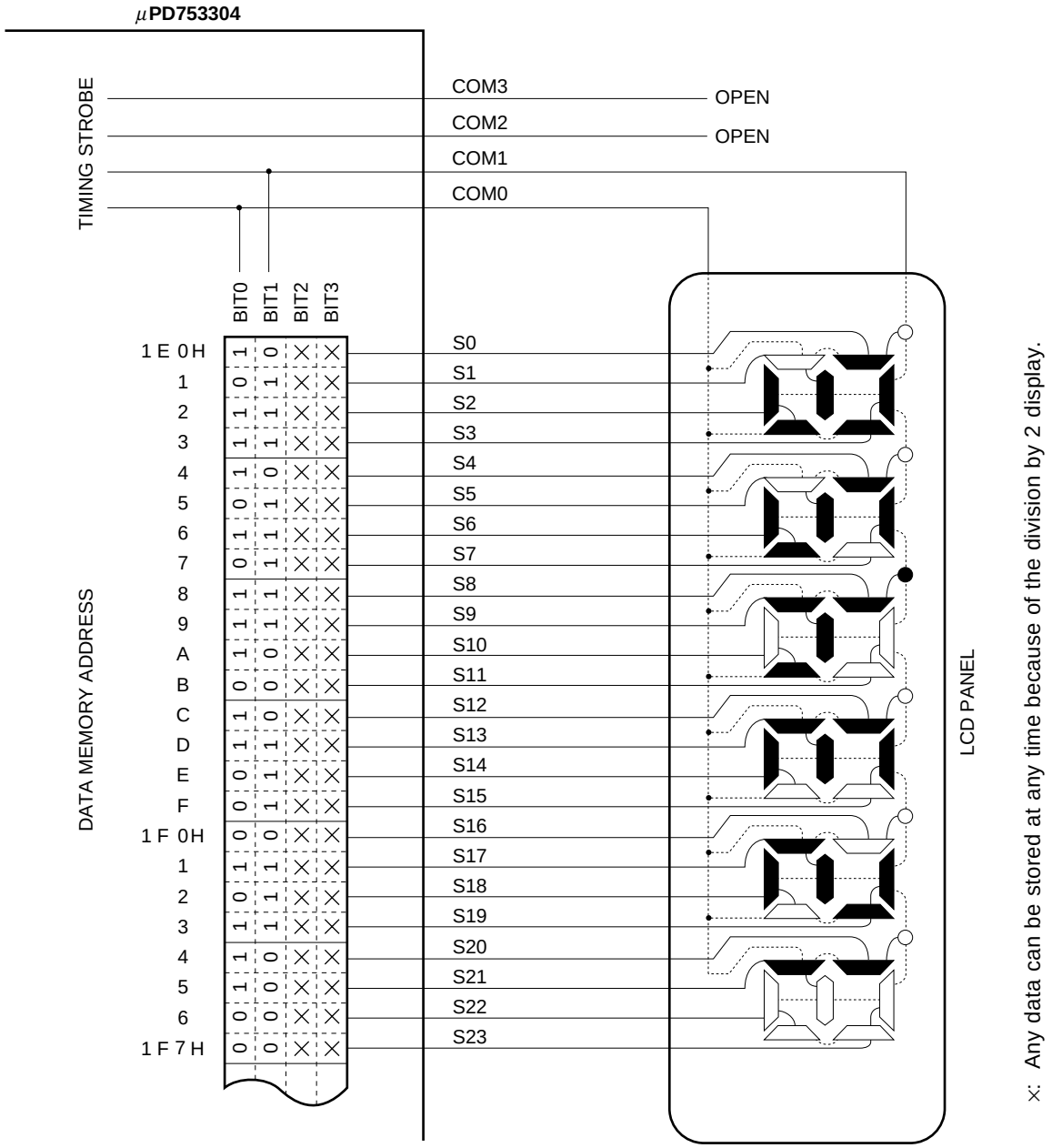
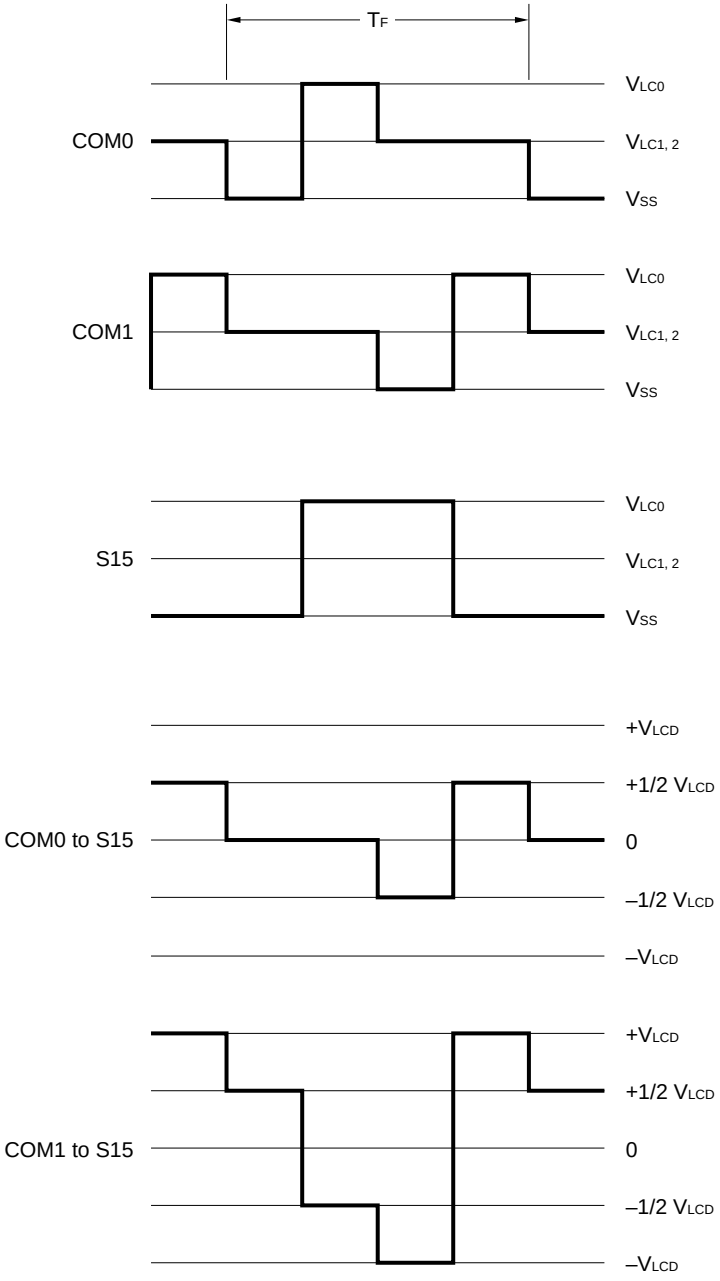


Figure 5-45. Division by 2 LCD Drive Waveform Example (1/2 Bias Method)



(3) Division by 3 display example

Figure 5-47 shows connection of a division by 3 mode 8-digit LCD panel having the display pattern shown in Figure 5-46, the μPD753304 segment signals (S0 to S23), and the common signals (COM0 to COM2). In this example, "123456.78" is displayed. The contents of the display data memory (addresses 1E0H to 1F7H) correspond to the display pattern.

Here, "6." at the third digit position is taken as an example. It is necessary to output the selection and non-selection voltages as shown in Table 5-15 to the S6 to S8 pins at the COM0 to COM2 common signal timings according to the display pattern shown in Figure 5-46.

Table 5-15. S6 to S8 Pin Selection and Non-selection Voltage (Division by 3 Display Example)

Segment / Common	S6	S7	S8
COM0	Non-selection	Selection	Selection
COM1	Selection	Selection	Selection
COM2	Selection	Selection	—

This shows that the display data memory address 1E6H corresponding to S6 needs to be set to x110.

Figures 5-48 (1/2 bias method) and 5-49 (1/3 bias method) show LCD drive waveforms among S6 and common signals. These show an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level being generated when S6 is the selection voltage at the COM1 selection timing and S6 is the selection voltage at the COM2 selection timing.

Figure 5-46. Division by 3 Mode LCD Display Pattern and Electrode Connection

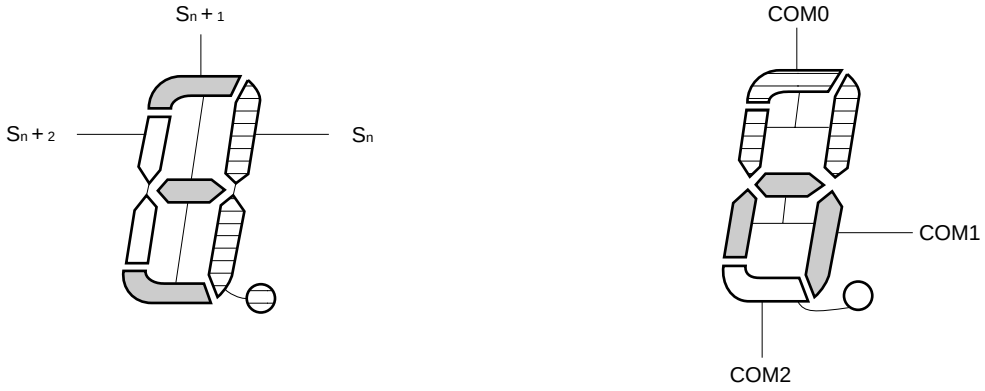


Figure 5-47. Division by 3 LCD Panel Connection Example

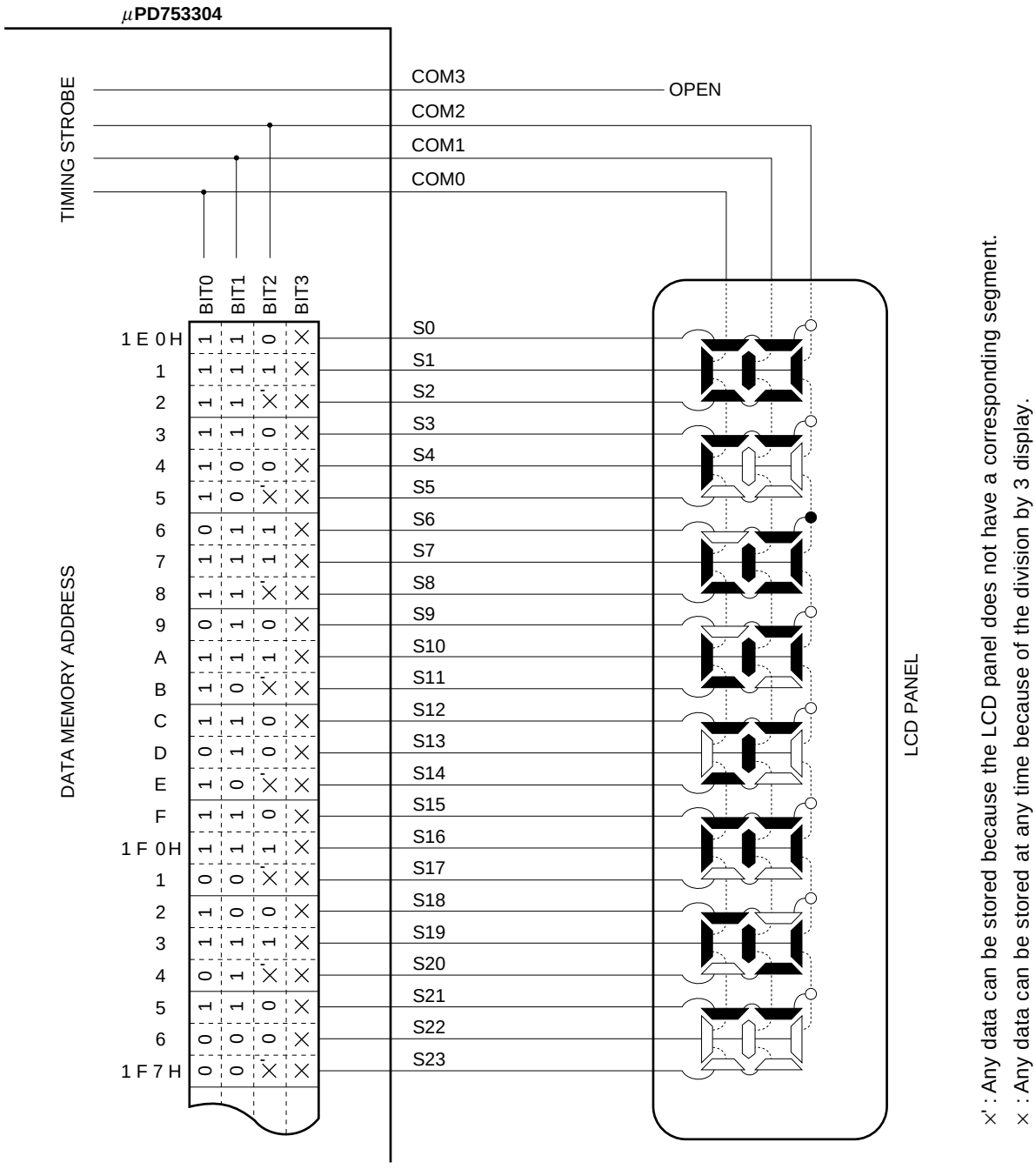


Figure 5-48. Division by 3 LCD Drive Waveform Example (1/2 Bias Method)

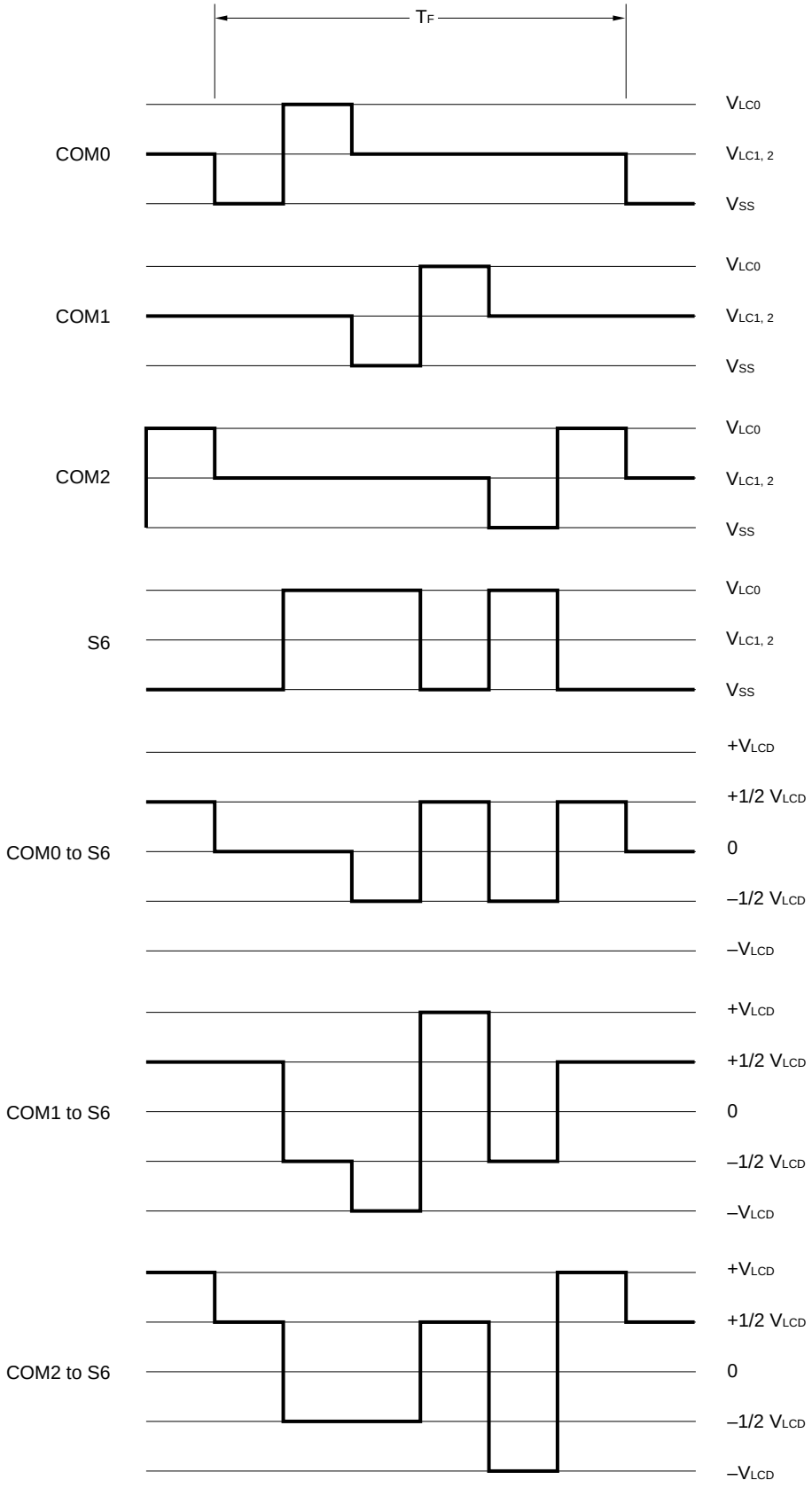
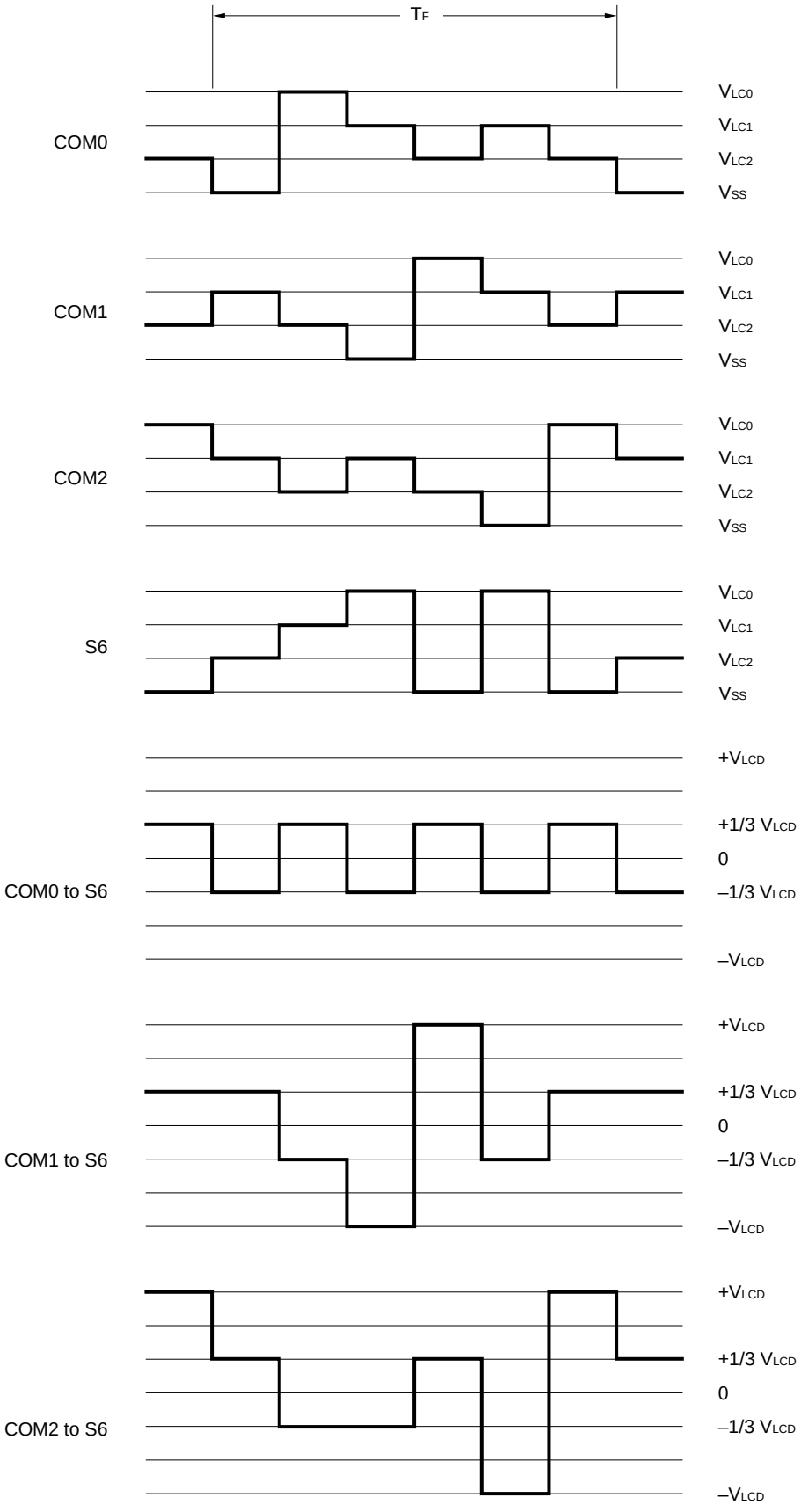


Figure 5-49. Division by 3 LCD Drive Waveform Example (1/3 Bias Method)



(4) Division by 4 display example

Figure 5-51 shows connection of a division by 4 mode 12-digit LCD panel having the display pattern shown in Figure 5-50, the μPD753304 segment signals (S0 to S23), and the common signals (COM0 to COM3). In this example, "123456.789012" is displayed. The contents of the display data memory (addresses 1E0H to 1F7H) correspond to the display pattern.

Here, "6." at the 7th digit position is taken as an example. It is necessary to output the selection and non-selection voltages as shown in Table 5-16 to the S12 and S13 pins at the COM0 to COM3 common signal timings according to the display pattern shown in Figure 5-50.

Table 5-16. S12, S13 Pin Selection and Non-selection Voltage (Division by 4 Display Example)

Segment / Common	S12	S13
COM0	Selection	Selection
COM1	Non-selection	Selection
COM2	Selection	Selection
COM3	Selection	Selection

This shows that the display data memory address 1ECH corresponding to S12 needs to be set to 1101.

Figure 5-52 shows LCD drive waveforms for S12, COM0, and COM1 signals (waveforms for COM2 and COM3 are omitted because of space limitation). This shows an alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level being generated when S12 is the selection voltage at the COM0 selection timing.

Figure 5-50. Division by 4 Mode LCD Display Pattern and Electrode Connection

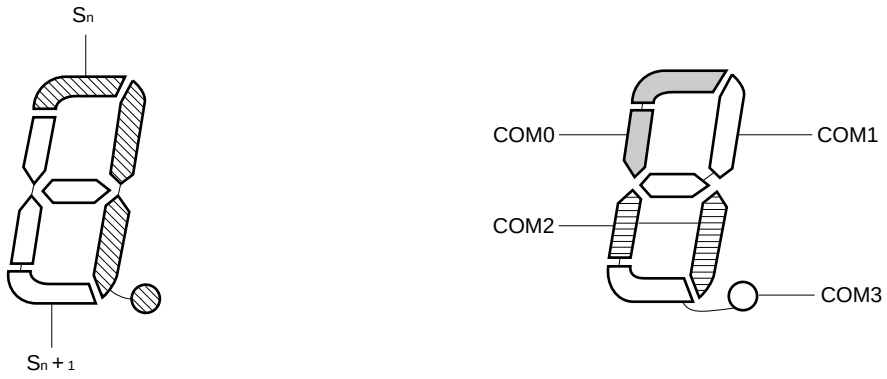


Figure 5-51. Division by 4 LCD Panel Connection Example

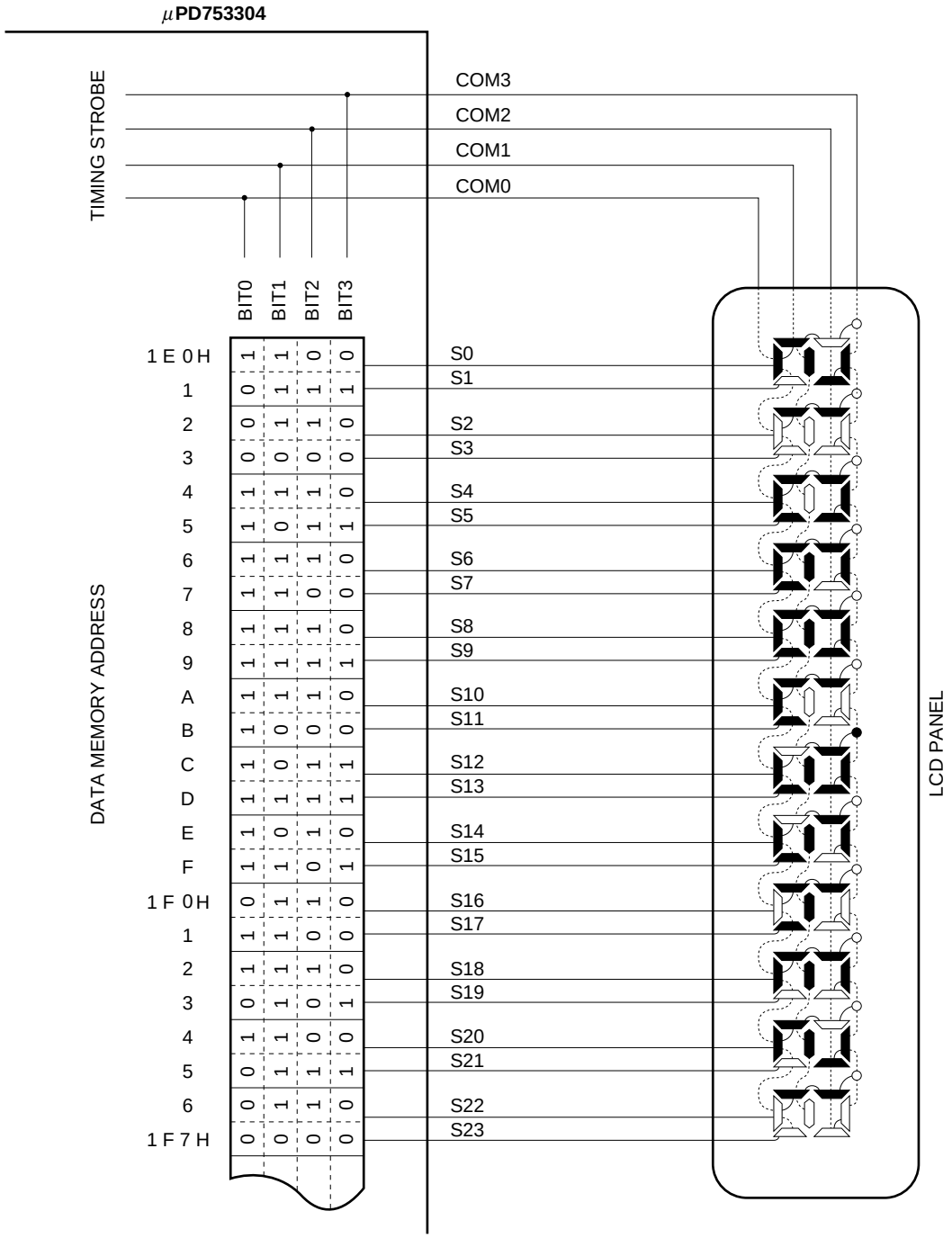
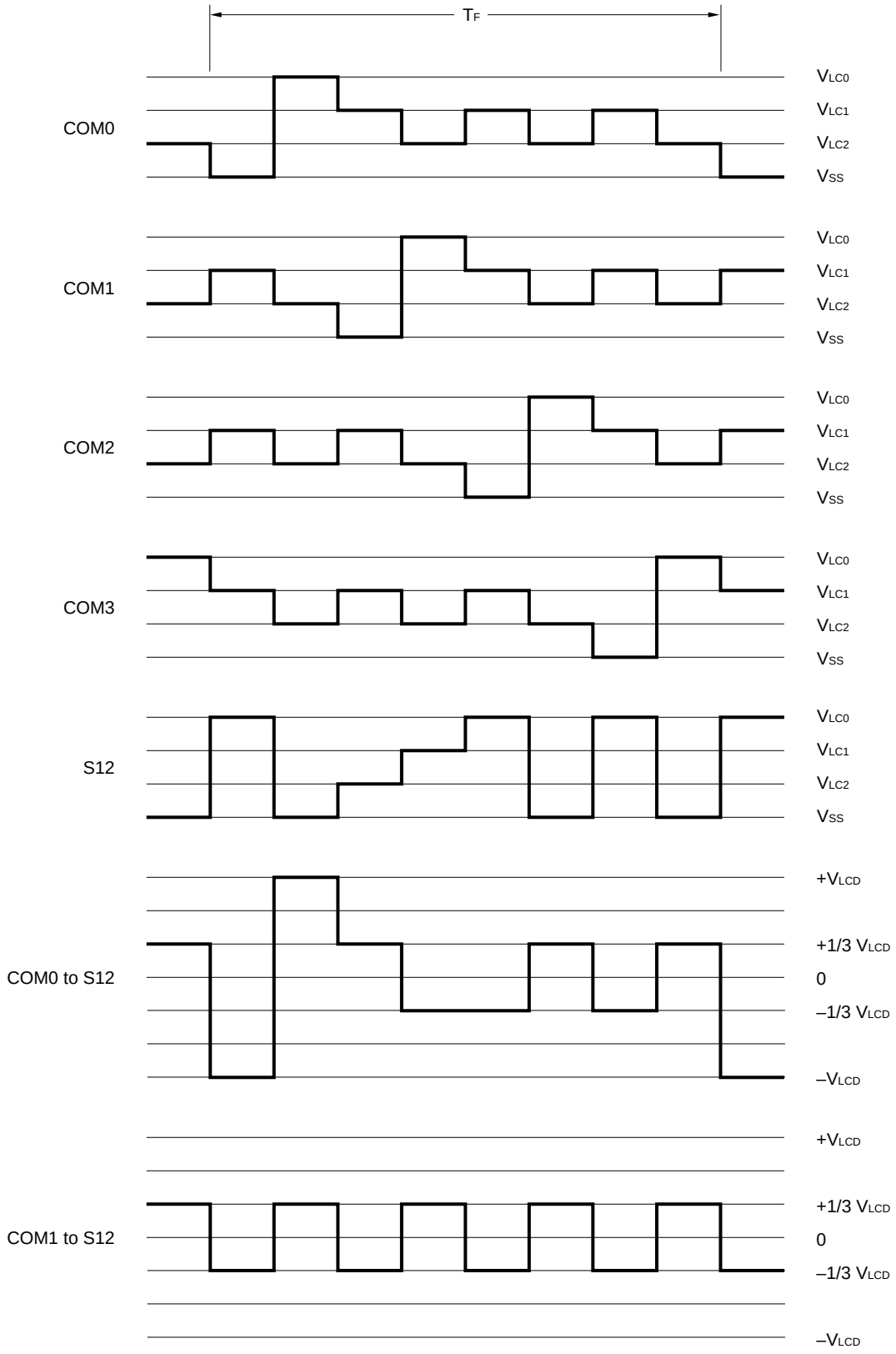


Figure 5-52. Division by 4 LCD Drive Waveform Example (1/3 Bias Method)



[MEMO]

CHAPTER 6 INTERRUPT FUNCTION AND TEST FUNCTION

The μPD753304 has three vectored interrupt sources and one test input that can be used for various applications.

The interrupt control circuit of the μPD753304 has unique features and can process interrupts at extremely high speed.

(1) Interrupt function

(a) Vectored interrupt function for hardware control, enabling/disabling the interrupt acceptance by the interrupt enable flag (IExxx) and interrupt master enable flag (IME).

(b) Can set any interrupt start address.

(c) Multiple interrupts wherein the order of priority can be specified by the interrupt priority selection register (IPS).

(d) Test function of interrupt request flag (IRQxxx). An interrupt generated can be checked by software.

(e) Release the standby mode. The interrupt to be released can be selected by the interrupt enable flag.

(2) Test function

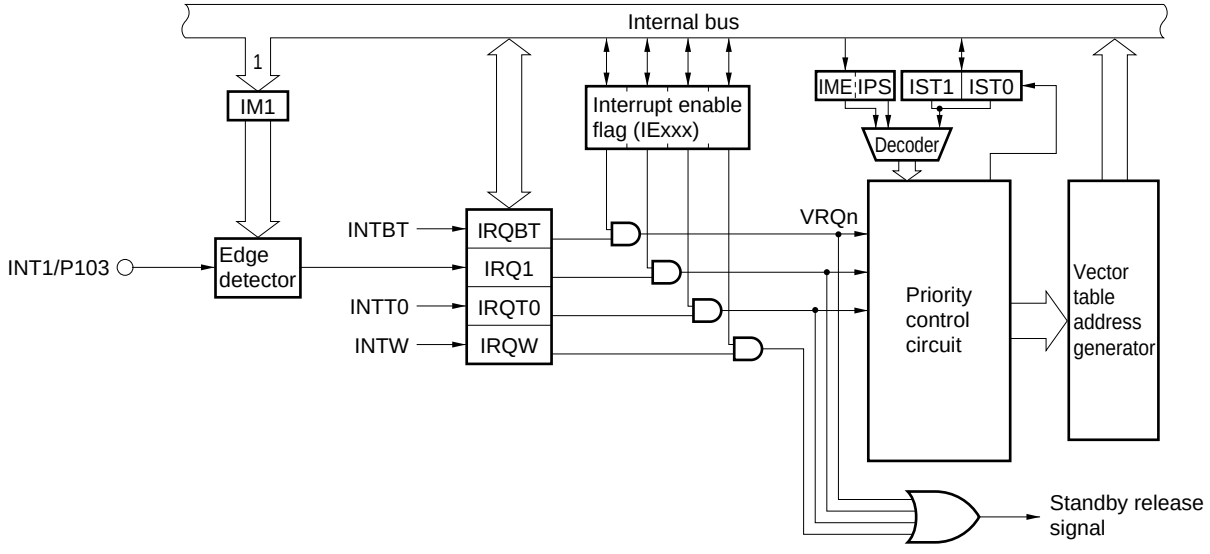
(a) Test request flag (IRQxxx) generation can be checked by software.

(b) Release the standby mode. The test source to be released can be selected by the test enable flag.

6.1 Configuration of Interrupt Control Circuit

The interrupt control circuit is configured as shown in Figure 6-1, and each hardware unit is mapped to the data memory space.

Figure 6-1. Interrupt Control Circuit Block Diagram



6.2 Types of Interrupt Sources and Vector Tables

The μPD753304 has the following three types of interrupt sources, and multiple interrupts by software control are allowed.

Table 6-1. Types of Interrupt Sources

Interrupt source		Internal/ external	Interrupt priority[Note]	Vectored interrupt request signal (vector table address)
INTBT	(Reference interval signal sent from the basic interval timer/watchdog timer)	Internal	1	VRQ1 (0002H)
INT1	(Selects rising edge or falling edge)	External	2	VRQ3 (0006H)
INTT0	(Match signal between the count register and modulo register of the timer counter 0)	Internal	3	VRQ5 (000AH)

Note The priority of interrupts is applied when several interrupt requests are generated simultaneously.

Figure 6-2. Interrupt Vector Table

Address				
0002H	MBE	RBE	INTBT start address (high-order 6 bits)	
	INTBT start address (low-order 8 bits)			
0006H	MBE	RBE	INT1 start address (high-order 6 bits)	
	INT1 start address (low-order 8 bits)			
000AH	MBE	RBE	INTT0 start address (high-order 6 bits)	
	INTT0 start address (low-order 8 bits)			

The interrupt priority in Table 6-1 indicates the priority according to which interrupts are executed if two or more interrupts occur at the same time, or if two or more interrupt requests are kept pending.

To the vector table, write the start address of interrupt processing, and the set values of MBE and RBE during interrupt processing. The vector table is set by using an assembler pseudo-instruction (VENTn: n = 1, 3, 5).

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 processing routine
<3> Setting of RBE in interrupt processing routine
<4> Symbol indicating start address of interrupt processing routine

Caution The contents (MBE, RBE, and start address) described to the operand of VENTn (n = 1, 3, 5) instruction is stored in the vector table address of 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
```

6.3 Hardware Controlling Interrupt Function

(1) Interrupt request flag and interrupt enable flag

The μPD753304 has the following three interrupt request flags (IRQxxx) corresponding to the respective interrupt sources:

INT1 interrupt request flag (IRQ1)
Timer counter 0 interrupt request flag (IRQT0)
BT interrupt request flag (IRQBT)

Each interrupt request flag is set to "1" when the corresponding interrupt request is generated, and is automatically cleared to "0" when the interrupt processing is executed.
The μPD753304 also has three interrupt enable flags (IExxx) corresponding to the respective interrupt request flags.

INT1 interrupt enable flag (IE1)
Timer counter 0 interrupt enable flag (IET0)
BT interrupt enable flag (IEBT)

When the interrupt enable flag is "1", the interrupt is enabled; and when it is "0", the interrupt is disabled.
When the interrupt request flag is set and interrupt enable flag enables the interrupt, a vectored interrupt request (VRQn: n = 1, 3, 5) is generated. This signal is also used to release the standby mode.
The interrupt request flag and interrupt enable flag are operated by a bit manipulation instruction and 4-bit memory manipulation instruction. When the bit manipulation instruction is used, they can be directly manipulated at any time regardless of MBE setting. The interrupt enable flag is manipulated by an EI IExxx instruction and DI IExxx instruction. A SKTCLR instruction is normally used to test the interrupt request flag.

```
Example  EI      IEBT    ; Enables INTBT.
         DI      IE1     ; Disables INT1.
         SKTCLR  IRQT0   ; Skips and clears when IRQT0 is 1.
```

When the interrupt request flag is set by an instruction, a vectored interrupt is executed as if an interrupt were generated.
The interrupt request flag and interrupt enable flag are cleared to "0" when a $\overline{\text{RESET}}$ signal is generated and all the interrupts are disabled.

Table 6-2. Set Signals for Interrupt Request Flags

Interrupt request flag	Set signal for interrupt request flag	Interrupt enable flag
IRQBT	Set by the reference interval signal by the basic interval timer/watchdog timer.	IEBT
IRQ1	Reset by the edge detection of an INT1/P103 pin input signal. The detection edge is selected by the INT1 edge detection mode register (IM1).	IE1
IRQT0	Set by a match signal sent from the timer counter 0.	IET0

(2) Interrupt priority selection register (IPS)

The interrupt priority selection register selects the higher-order priority interrupts in a system wherein multiple interrupts are allowed. Its low-order 3 bits are used for specification.

Bit 3 is the interrupt master enable flag (IME) which specifies whether all the interrupts are disabled or not.

The IPS is set by a 4-bit memory manipulation instruction. Bit 3 is set and reset by an EI/DI instruction.

To change the contents of the low-order 3 bits of IPS, the interrupt must be disabled (IME = 0).

```
Example  DI              ; Disables interrupt
         CLR1    MBE
         MOV     A, #1011
         MOV     IPS, A  ; Gives high-order priority to INT1 and enables interrupt
```

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

Figure 6-3. Interrupt Priority Selection Register

Address	3	2	1	0	Symbol
FB2H	IPS3	IPS2	IPS1	IPS0	IPS

Selection of higher-order priority interrupts

0	0	0	No interrupts are handled as higher-order priority interrupts.	
0	0	1	VRQ1 (INTBT)	The above vectored interrupts are regarded as higher-order priority interrupts.
0	1	1	VRQ3 (INT1)	
1	0	1	VRQ5 (INTT0)	
Other than above			Setting prohibited	

Interrupt master enable flag (IME)

0	Disables all the interrupts and no vectored interrupt is started.
1	Controls interrupt enable/disable by the corresponding interrupt enable flag.

(3) Hardware of INT1

- Figure 6-4 shows the configuration of the INT1. An external interrupt is input to select the detection edge, rising edge or falling edge.
  The detection edge is selected by the INT1 edge detection mode register (IM1).
  Figure 6-5 shows the format of the IM1. IM1 is set by a bit manipulation instruction. All the bits are cleared to “0” when the $\overline{\text{RESET}}$ signal is generated and the rising edge specification is adopted.

Figure 6-4. Configuration of INT1

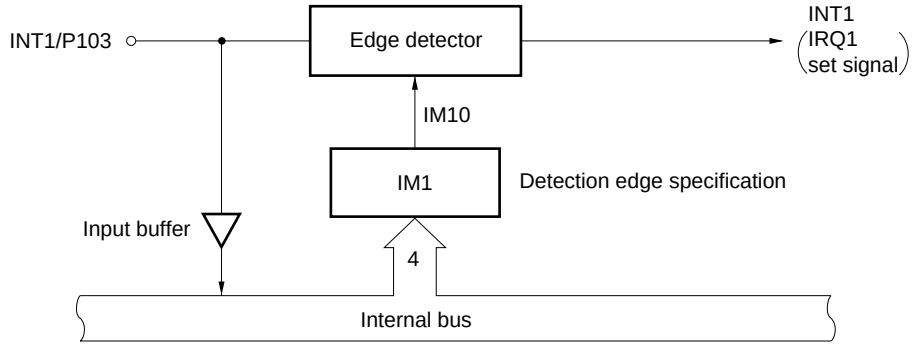


Figure 6-5. INT1 Edge Detection Mode Register Format

Address: FB5H

3	2	1	0
0	0	0	IM10

Symbol: IM1

IM10	Detection edge specification
0	Rising edge specification
1	Falling edge specification

Caution When the edge detection mode register is changed, the interrupt request flag may be set in some case, therefore the interrupts must be disabled beforehand to change the mode register and the interrupt request flag must be cleared by a CLR1 instruction, and then the interrupts must be enabled.

(4) Interrupt status flag

The interrupt status flags (IST0 and IST1) indicate the status of the processing currently executed by the CPU and are included in PSW.

The interrupt priority control circuit controls nesting of interrupts according to the contents of these flags as shown in Table 6-3.

IST0 and IST1 can be changed by using a 4-bit or bit manipulation instruction, and 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 the interrupt. Execute the EI instruction after manipulating the flags to enable the interrupt.

IST1 and IST0 are saved to the stack memory along with the other flags of PSW when an interrupt is acknowledged, and their statuses are automatically changed higher by one. When the RETI instruction is executed, the original values of IST1 and IST0 are restored.

The contents of these flags are cleared to "0" when the $\overline{\text{RESET}}$ signal is generated.

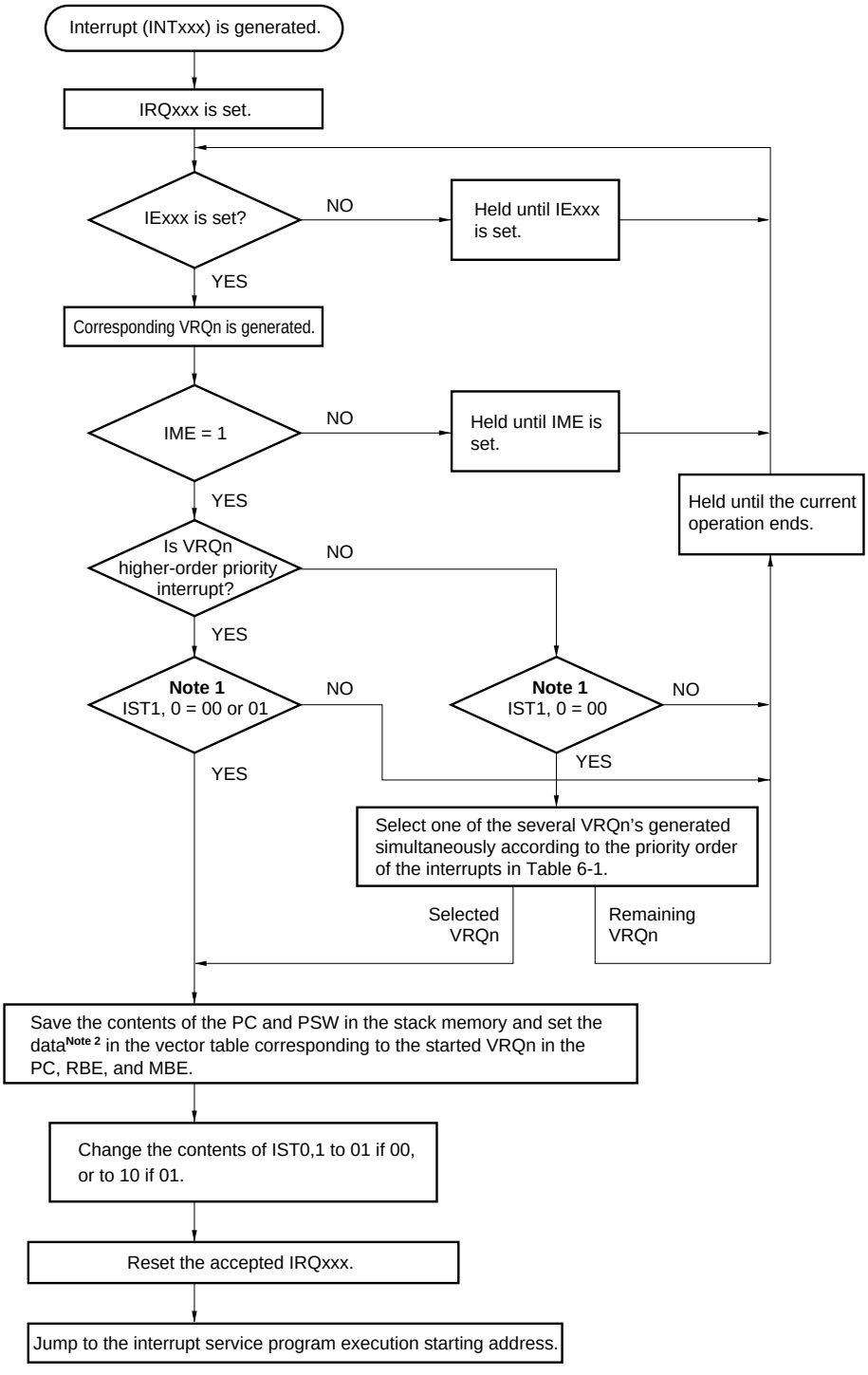
Table 6-3. IST1, IST0 and Interrupt Processing Status

IST1	IST0	Status of processing under execution	Processing by CPU	Interrupt request that can be acknowledged	After interrupt acknowledged	
					IST1	IST0
0	0	Status 0	Executes normal program	All interrupts can be acknowledged	0	1
0	1	Status 1	Processes interrupt with low or high priority	Interrupt with high priority can be acknowledged	1	0
1	0	Status 2	Processes interrupt with high priority	Acknowledging all interrupts is disabled	–	–
1	1	Setting prohibited				

6.4 Interrupt Sequence

When an interrupt is generated, it is executed in the following procedure.

Figure 6-6. Interrupt Processing Sequence



Notes 1. IST1, 0: interrupt status flag (bits 3, 2 of PSW; see **Table 6-3**)

2. Each vector table stores the interrupt service program starting address and values set for the MBE and RBE at the time the interrupt service starts.

6.5 Multiple Interrupt Service Control

The μPD753304 accepts multiple interrupts in the following two methods.

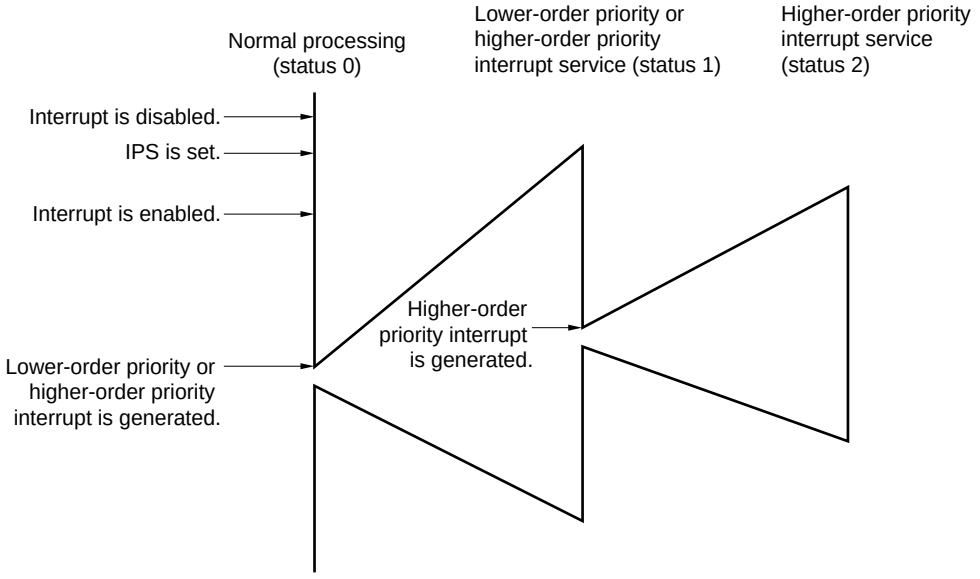
(1) Multiple interrupts wherein higher-order priority interrupts are specified

This is a standard multiple interrupts method of the μPD753304, wherein one of the interrupt sources is selected to enable the multiple interrupts (double interrupts) of the interrupt.

That is, the higher-order priority interrupts specified by the interrupt priority selection register (IPS) can be accepted when the status of the current operation is 0 and 1, and the other lower-order priority interrupts can be serviced when the status is 0 (see **Figure 6-7** and **Table 6-3**).

Therefore, if this method is used when you wish to nest only one interrupt, operations such as enabling and disabling interrupts while the interrupt is processed need not to be performed, and the nesting level can be kept to 2.

Figure 6-7. Multiple Interrupts by Higher-Order Priority Interrupts



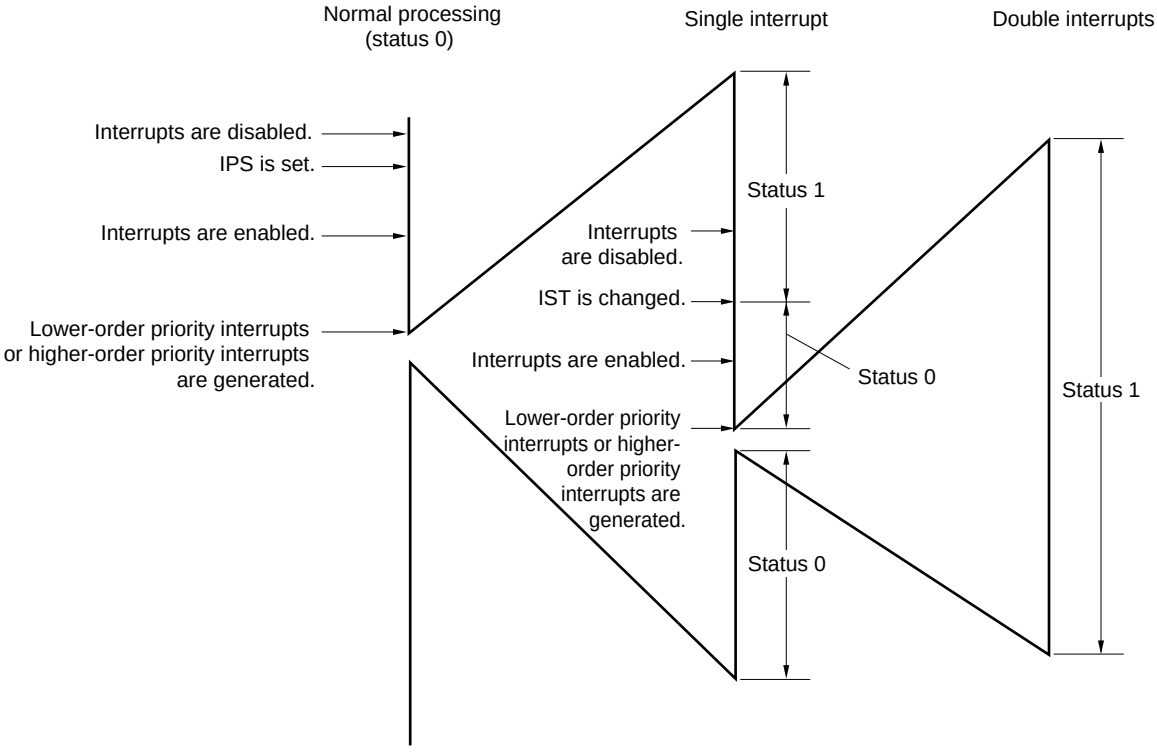
(2) Multiple interrupts changing the interrupt status flag

If the interrupt status flag is changed by the program, multiple interrupts can be accepted. That is, if IST1 and IST0 are changed to “0, 0” (status 0), multiple interrupts can be serviced.

This method is used when multiple interrupts (two or more interrupts) are to be accepted.

The IST1 and IST0 must be changed beforehand in a status in which the interrupts are disabled by a DI instruction.

Figure 6-8. Multiple Interrupts by Changing Interrupt Status Flag

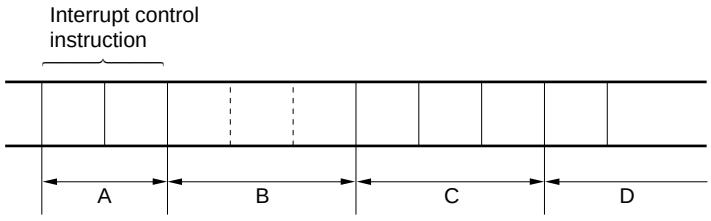


6.6 Machine Cycles until Interrupt Processing

The number of machine cycles required since 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 executed

If IRQxxx is set while an interrupt control instruction is executed, the next one instruction is executed. Then three machine cycles of interrupt processing is performed and the interrupt routine is executed.



A: Sets IRQxxx
B: Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)
C: Interrupt processing (3 machine cycles)
D: Executes interrupt routine

Cautions 1. If two or more interrupt control instructions are described in a row, these control instructions will be all executed consecutively. After having executed the one instruction following the interrupt control instruction executed last, interrupt processing of three machine cycles will be performed, followed by the interrupt routine.

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 kept pending until the EI instruction is executed next time.

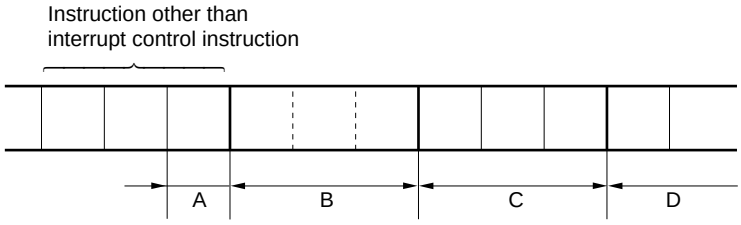
Remarks 1. An interrupt control instruction manipulates the hardware units related to interrupt (address FBxH of the data memory). The DI and EI instructions are interrupt control instructions.

2. The three machine cycles of interrupt processing are the time required to manipulate the stack which is manipulated when an interrupt is acknowledged.

(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 processing are performed, and finally the interrupt routine is executed.

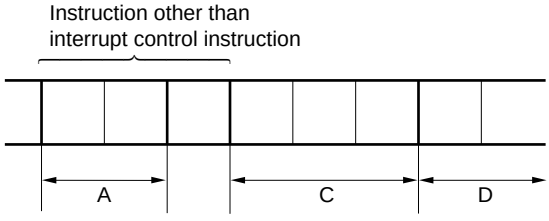


A: Sets IRQxxx
B: Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)
C: Interrupt processing (3 machine cycles)
D: Executes interrupt routine

Caution If the next instruction is an interrupt control instruction, the one instruction following the interrupt control instruction executed last is executed, three machine cycles of interrupt processing are performed, and finally the interrupt routine is executed. If the DI instruction is executed after IRQxxx has been set, the interrupt request corresponding to the set IRQxxx is kept pending.

(b) If IRQxxx is set before the last machine cycle of the instruction under execution

In this case, three machine cycles of interrupt processing are performed after execution of the current instruction, and then the interrupt routine is executed.



A: Sets IRQxxx
C: Interrupt processing (3 machine cycles)
D: Executes interrupt routine

6.7 Effective Usage of Interrupt

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. If the interrupt routine is for one nested interrupt, use register bank 0 with RBE = 0, so that you do not have to save or restore the registers. When two interrupts are nested, set RBE to 1, save the register bank by using the PUSH BS instruction, and use register bank 1 with RBS = 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 enhanced by setting the interrupt request flag by an instruction.

6.8 Application of Interrupt

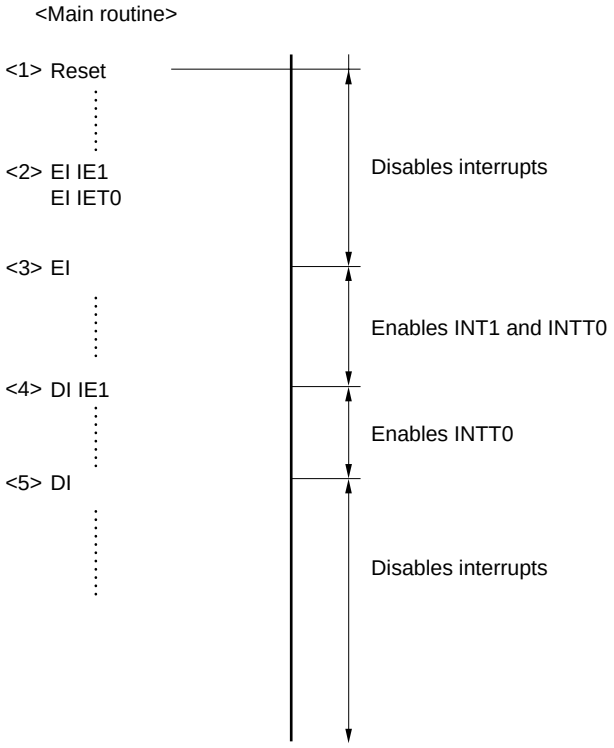
To use the interrupt function, first set as follows by the main routine:

(a) Set the interrupt enable flag of the interrupt used (by using the EI IExxx instruction).
(b) To use INT1, select the active edge (set IM1).
(c) To use nesting (of an interrupt with the higher priority), set IPS (IME can be set at the same time).
(d) Set the interrupt master enable flag (by using the EI instruction).

In the interrupt routine, MBE and RBE are set by the vector table. However, when the interrupt specified as having the higher priority is processed, the register bank must be saved and set.

To return from the interrupt routine, use the RETI instruction.

(1) Enabling or disabling interrupt



<1> All the interrupts are disabled by the $\overline{\text{RESET}}$ signal.

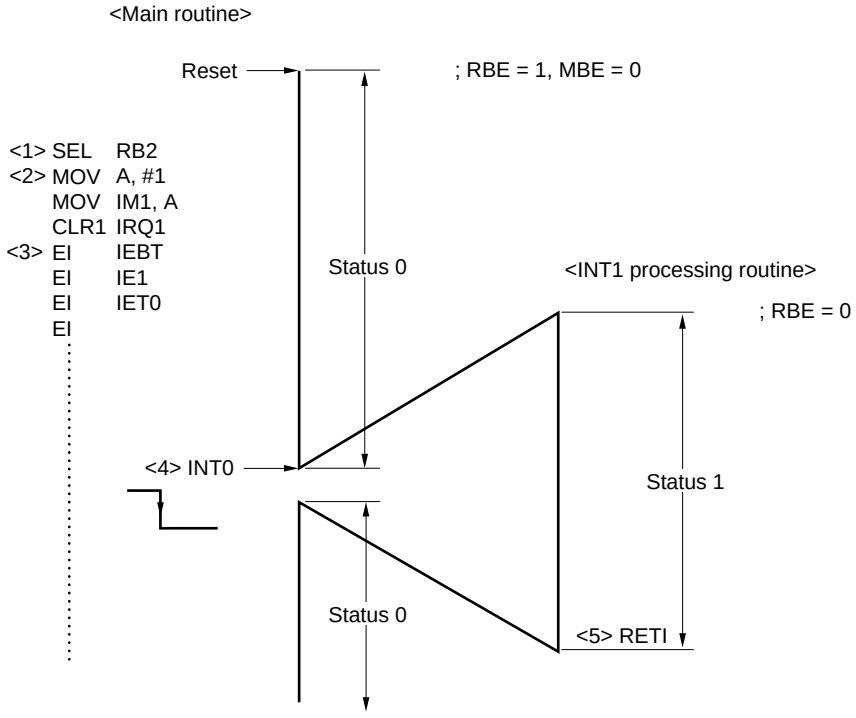
<2> An interrupt enable flag is set by the EI IExxx instruction.
At this stage, the interrupts are still disabled.

<3> The interrupt master enable flag is set by the EI instruction.
At this stage, INT1 and INTT0 are enabled.

<4> The interrupt enable flag is cleared by the DI IExxx instruction, and INT1 is disabled.

<5> All the interrupts are disabled by the DI instruction.

(2) Example of using INTBT, INT1 (falling edge active), INTT0, no multiple interrupt (all interrupts have lower priority)



<1> All the interrupts are disabled by the $\overline{\text{RESET}}$ signal and status 0 is set.
RBE = 1 is specified by the reset vector table. The SEL RB2 instruction uses register banks 2 and 3.

<2> INT1 is specified to be active at the falling edge.

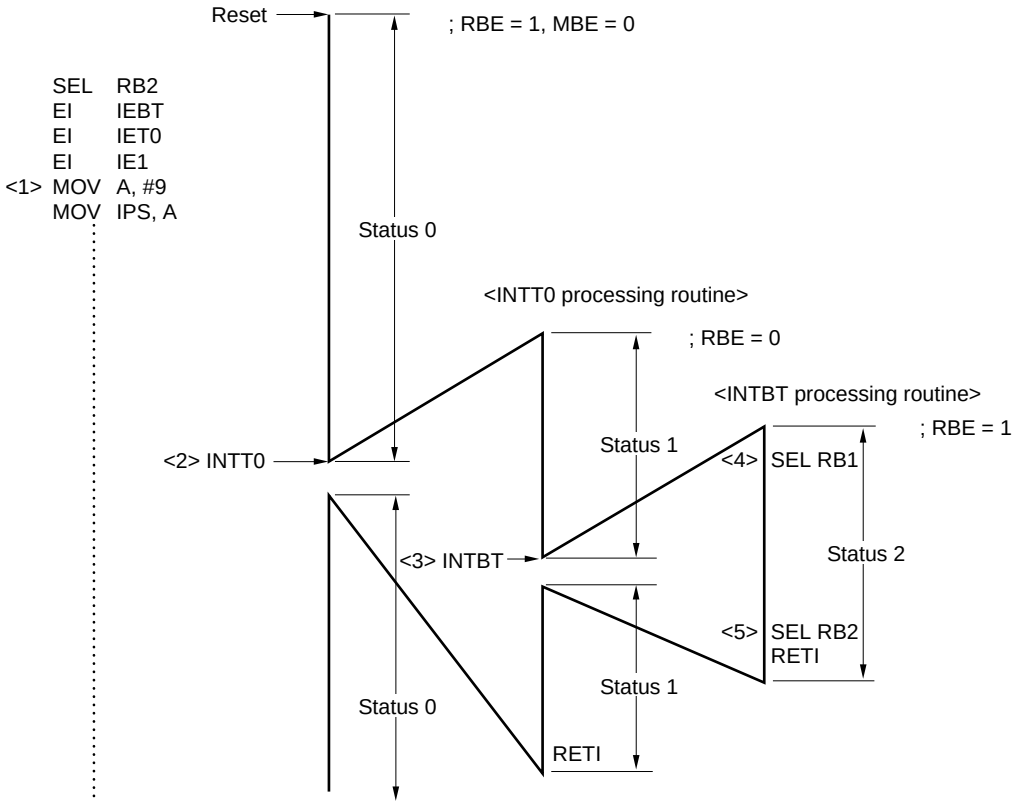
<3> The interrupt is enabled by the EI, EI IExxx instruction.

<4> The INT1 interrupt routine is started at the falling edge of INT1. The status is changed to 1, and all the interrupts are disabled.
RBE = 0, and register banks 0 and 1 are used.

<5> Execution returns from the interrupt routine when the RETI instruction is executed. The status is returned to 0 and the interrupt is enabled.

Remark If all the interrupts are used as having the lower priority as shown in this example, saving or restoring the register bank is not necessary if RBE = 1 and RBS = 2 for the main routine and register banks 2 and 3 are used, and RBE = 0 for the interrupt routine and register banks 0 and 1 are used.

(3) Nesting of interrupts with higher priority
(INTBT has higher priority and INTT0 and INT1 have lower priority)



<1> INTBT is specified as having the higher priority by setting the IPS, and the interrupt is enabled at the same time.

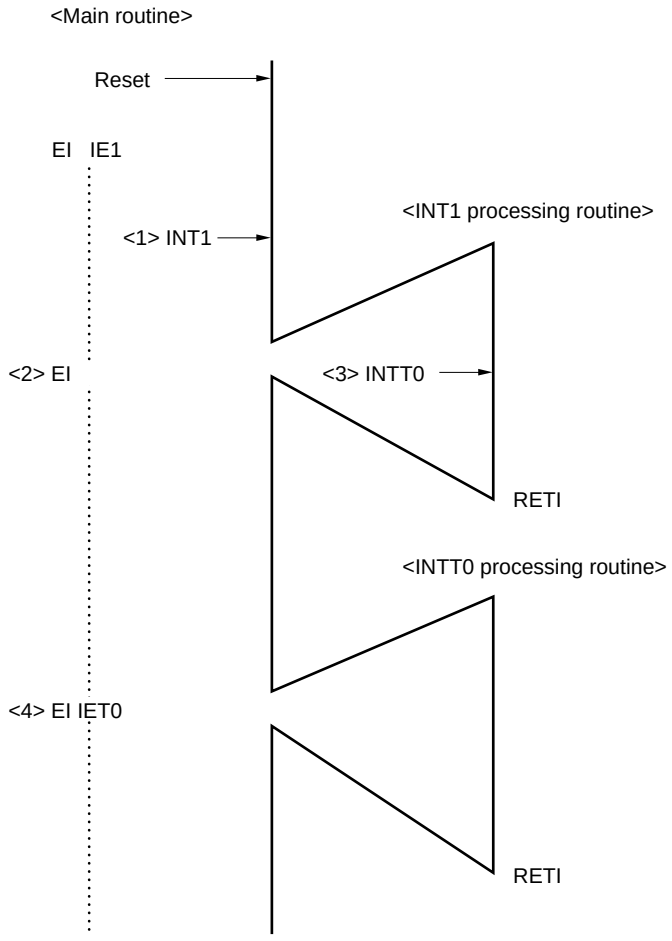
<2> INTT0 processing routine is started when INTT0 with the lower priority occurs. Status 1 is set and the other interrupts with the lower priority are disabled. RBE = 0 to select register bank 0.

<3> INTBT with the higher priority occurs. The interrupts are nested. The status is changed to 0 and all the interrupts are disabled.

<4> RBE = 1 and RBS = 1 to select register bank 1 (only the registers used may be saved by the PUSH instruction).

<5> RBS is returned to 2, and execution returns to the main routine. The status is returned to 1.

(4) Executing pending interrupt – interrupt input while interrupts are disabled –



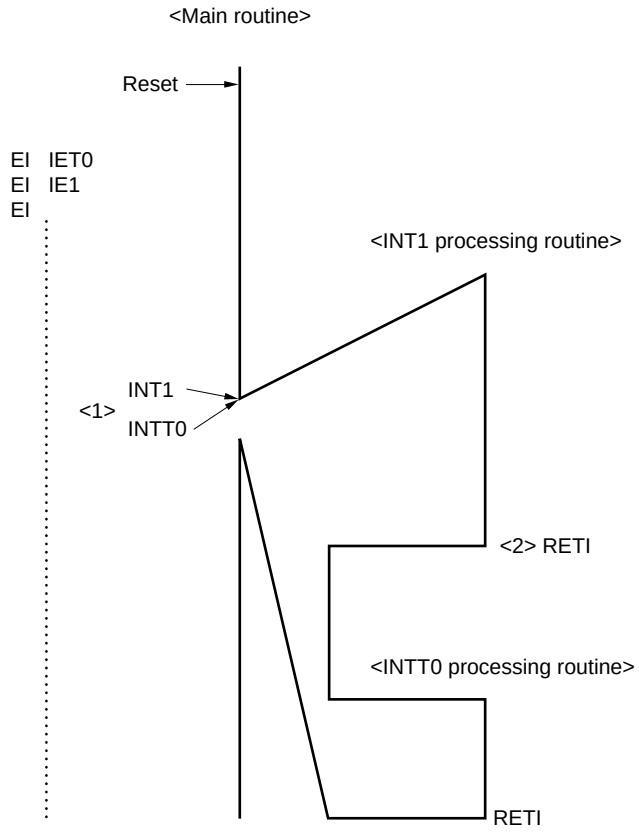
<1> The request flag is kept pending even if INT1 is set while the interrupts are disabled.

<2> INT1 processing routine is started when the interrupts are enabled by the EI instruction.

<3> Same as <1>.

<4> INTT0 processing routine is started when the pending INTT0 is enabled.

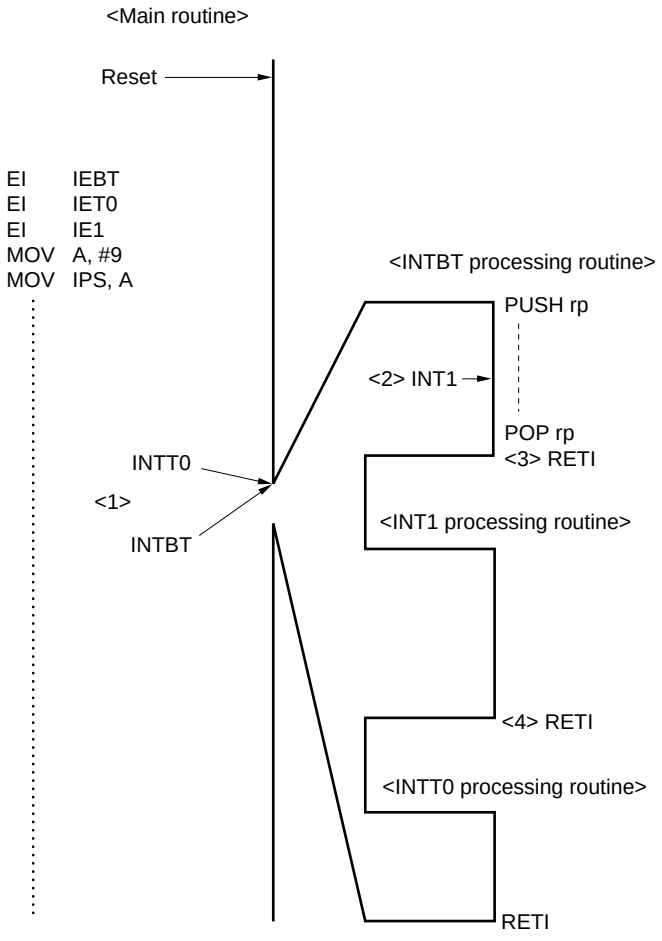
(5) Executing pending interrupt – two interrupts with lower priority occur simultaneously –



<1> If INT1 and INTT0 with the lower priority occur at the same time (while the same instruction is executed), INT1 with the higher priority is executed first (INTT0 is kept pending).

<2> When the INT1 processing routine is terminated by the RETI instruction, the pending INTT0 processing routine is started.

(6) Executing pending interrupt – interrupt occurs during interrupt processing (INTBT has higher priority and INTT0 and INT1 have lower priority) –



<1> If INTBT with the higher priority and INTT0 with the lower priority occur at the same time, the processing of the interrupt with the higher priority is started (if there is no possibility that an interrupt with the higher priority occurs while another interrupt with the higher priority is processed, DI IExx is not necessary).

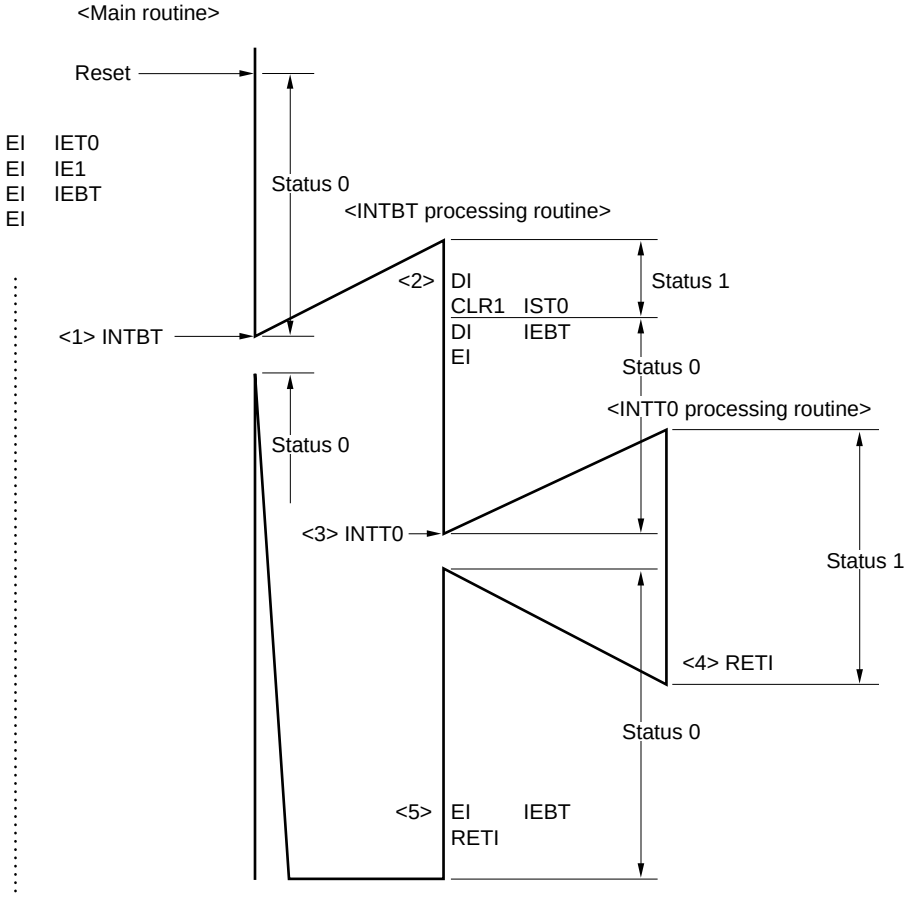
<2> If an interrupt with the lower priority occurs while the interrupt with the higher priority is executed, the interrupt with the lower priority is kept pending.

<3> When the interrupt with the higher priority has been processed, INT1 with the higher priority of the pending interrupts is executed.

<4> When the processing of INT1 has been terminated, the pending INTT0 is processed.

(7) Enabling two nesting of interrupts

– INTT0 and INT1 are nested doubly and INTBT is nested singly –



<1> When INTBT that does not enable nesting occurs, the INTBT processing routine is started. The status is 1.

<2> The status is changed to 0 by clearing IST0. INTBT that does not enable nesting is disabled.

<3> When INTT0 that enables nesting occurs, nesting is executed. The status is changed to 1, and all the interrupts are disabled.

<4> The status is returned to 0 when INTT0 processing is terminated.

<5> The disabled INTBT is enabled, and execution returns to the main routine.

6.9 Test Function

6.9.1 Types of test sources

The μPD753304 has one type of test source.

Table 6-4. Type of Test Source

Test source	Internal/external
INTW (signal from watch timer)	Internal

6.9.2 Hardware devices controlling test function

(1) Test request flag, test enable flag

The test request flag (IRQW) is set to “1” when a test request (INTW) is generated. When the test processing is completed, it must be cleared to “0” by software.

The test enable flag (IEW) is annexed to test request flag. When it is “1”, a standby release signal is enabled; and when it is “0”, the signal is disabled.

When both the test request flag and test enable flag are set to “1”, a standby release signal is generated.

Table 6-5 lists the set signal for the test request flag.

Table 6-5. Set Signal for Test Request Flag

Test request flag	Set signal for test request flag	Test enable flag
IRQW	Set by a signal sent from the watch timer.	IEW

CHAPTER 7 STANDBY FUNCTION

The μPD753304 has a standby function that reduces the power dissipation of the system. This standby function can be implemented in the following two modes:

★ Both modes can be canceled by generating an interrupt request that is enabled by an interrupt enable flag, or $\overline{\text{RESET}}$ signal.

- STOP mode
- HALT mode

The functions of the STOP and HALT modes are as follows:

(1) STOP mode

In this mode, the main system clock oscillation circuit and the subsystem clock oscillation circuit are stopped and therefore, the entire system is stopped. As a result, the power dissipation of the CPU is substantially reduced. Moreover, the contents of the data memory can be retained at a low voltage (V_{DD} = 2.5 V MIN.). This mode is therefore useful for retaining the data memory contents with an extremely low current dissipation.

512/f_{CC} or no wait can be selected for a wait time in STOP mode release by an interrupt request. Therefore, select no wait when processing must be started immediately by an interrupt request.

When processing must be started immediately by an interrupt request with 512/f_{CC} selected, select the HALT mode.

Subsystem clock oscillation can be stopped upon STOP instruction execution after setting bit 3 of SOS to 1.

(2) HALT mode

In this mode, the operating clock of the CPU is stopped. Oscillation of the system clock oscillation circuit continues. This mode does not reduce the current dissipation as much as the STOP mode, but it is useful when processing must be resumed immediately when an interrupt request is issued, or for an intermittent operation such as a watch operation.

In either mode, all the contents of the registers, flags, and data memory immediately before the standby mode is set are retained. Moreover, the contents of the output latches and output buffers of the I/O ports are also retained; therefore, the status of the I/O ports are processed in advance so that the current dissipation of the overall system can be minimized.

The following page describes the points to be noted in using the standby mode.

Cautions 1. The STOP mode and HALT mode can be used regardless of whether the system operates with the main system clock or subsystem clock.

★ **Both modes can be canceled by generating an interrupt request that is enabled by an interrupt enable flag, or reset signal.**

2. If the main system clock is set to STOP mode when the LCD controller/driver, watch timer, and timer counter operate with main system clock f_{CC}, the operations of the LCD controller/driver, watch timer, and timer counter are stopped.

To continue the operations, therefore, change the operating clock to subsystem clock f_{CT} before setting the main system clock to the STOP mode.

3. Efficient operation with a low current dissipation at a low voltage can be performed by selecting the standby mode, CPU clock, and system clock. In any case, however, the time described in 5.2.3 Setting of system clock and CPU clock is required until the operation is started with the new clock when the clock has been changed by manipulating the control register. To use the clock switching function and standby mode in combination, therefore, set the standby mode after the time required for switching has elapsed.

4. To use the standby mode, process so that the current dissipation of the I/O ports is minimized. Especially, do not open the input port, and be sure to input low or high level to the input port.

7.1 Standby Mode Setting and Operation Status

Table 7-1. Operation Status in Standby Mode

Item \ Mode		STOP mode	HALT mode
Set instruction		STOP instruction	HALT instruction
System clock when set		Settable both by the main system clock and subsystem clock.	
Operation status	Clock generator	The main system clock stops oscillation. The subsystem clock stops oscillation when the sub-oscillator stop enable flag (SOS.3) is set to 1.	Only the CPU clock Φ halts (oscillation continues).
	Basic interval timer/ watchdog timer	Operation stops.	Operable only when the main system clock is oscillated BT mode : Set IRQBT with reference time interval WT mode : Reset generation with BT overflow
	Timer counter	Operable only when f_{CT} is selected as the count clock, and SOS.3 is set to 0.	Cannot operate only when the division of the main system clock is selected as the count clock during the main system clock stop.
	LCD controller/driver	Operable only when f_{CT} is selected as the LCDCL, and SOS.3 is set to 0.	Operable.
	Watch timer	Operable only when f_{CT} is selected as the count clock, and SOS.3 is set to 0.	Operable.
	External interrupt	Operable only when SOS.3 is set to 0.	
	CPU	The operation stops.	
Release signal		Interrupt request signal sent from the operable hardware enabled by the interrupt enable flag or $\overline{\text{RESET}}$ signal generation.	

The STOP mode is set by a STOP instruction and the HALT mode is set by a HALT instruction. The STOP instruction and HALT instruction set bit 3 and bit 2, respectively, of the PCC.

A NOP instruction must be placed following the STOP instruction and HALT instruction.

When changing the CPU clock by the low-order 2 bits of the PCC, there may be a time difference between PCC updating and CPU clock change as shown in **Table 5-5 Maximum Time Required to Switch System to/from CPU Clocks**. Consequently, when the operating clock before the standby mode and the CPU clock after the standby mode is released are to be changed, the PCC must be updated and then the standby mode must be set after the machine cycle necessary to change the CPU clock has elapsed.

In the standby mode, the data items stored in all the registers and data memory whose operations are stopped in the standby mode, such as the general-purpose registers, flags, mode registers, and output latch, are held.

Caution Before setting the standby mode, reset all the interrupt request flags.
If there is an interrupt source in which both the interrupt request flag and interrupt enable flag are set, the standby mode is released at the moment the system enters it (See Figure 6-1 Interrupt Control Circuit Block Diagram). However, when the STOP mode is set, the system enters the HALT mode immediately after a STOP instruction is executed and then returns to the operating mode after a wait[Note].

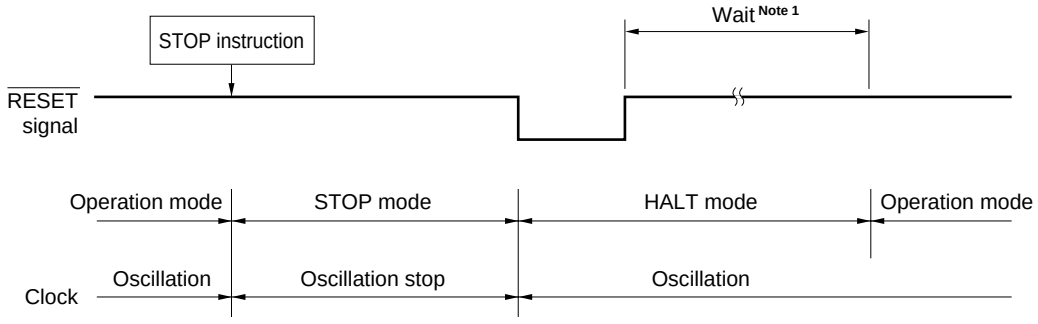
Note Either of the following can be selected by the mask option.
512/f_{CC} (142 μs: 3.6-MHz operation)
No wait

7.2 Standby Mode Release

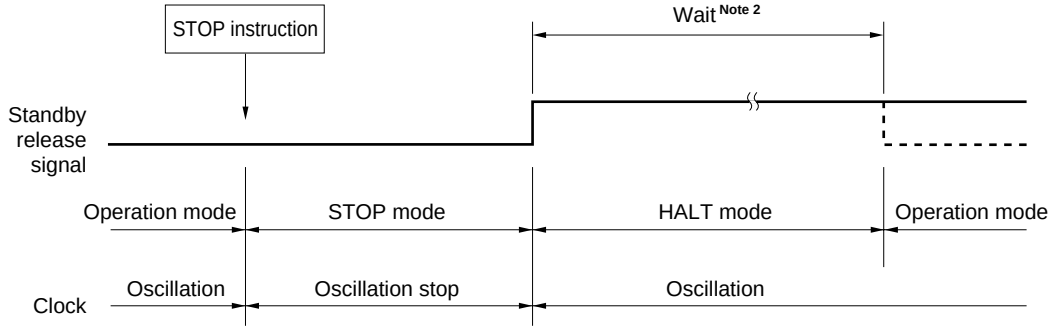
The standby mode (STOP or HALT) is released when an interrupt request signal enabled with an interrupt enable flag occurs or a $\overline{\text{RESET}}$ signal is generated. Figure 7-1 shows the standby mode release operation.

Figure 7-1. Standby Mode Release Operation (1/2)

(a) STOP mode release when a $\overline{\text{RESET}}$ signal is generated



(b) STOP mode release when an interrupt occurs



Notes 1. 56/fCC (15.6 μs: 3.6-MHz operation)

2. Either of the following can be selected by the mask option.

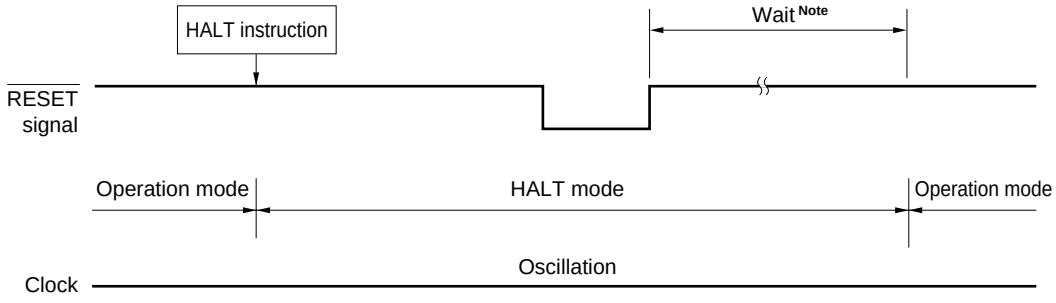
512/fCC (142 μs: 3.6-MHz operation)

No wait

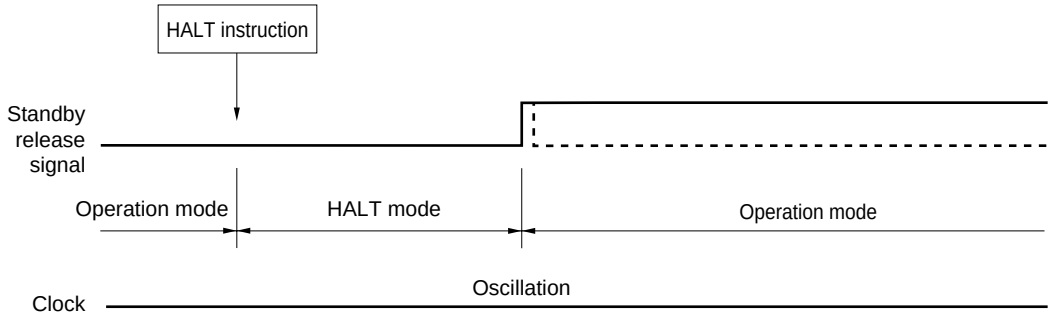
Remark Broken line: When the interrupt request to release the standby is acknowledged.

Figure 7-1. Standby Mode Release Operation (2/2)

(c) HALT mode release when a $\overline{\text{RESET}}$ signal is generated



(d) HALT mode release when an interrupt occurs



Note 56/fcc (15.6 μs: 3.6-MHz operation)

Remark Broken line: When the interrupt request to release the standby is acknowledged.

7.3 Operation After Releasing Standby Mode

(1) When the standby mode is released by a $\overline{\text{RESET}}$ signal generation, the normal reset operation is performed.

(2) When the standby mode is released by generation of an interrupt, whether a vectored interrupt is to be serviced or not at the time the CPU resumes instruction execution is determined by the contents of the interrupt master enable flag (IME).

(a) When IME = 0

Following the release of the standby mode, the instruction execution resumes starting with the instruction subsequent to the standby mode setting instruction. The interrupt request flag is held.

(b) When IME = 1

After the standby mode is released, two instructions are executed and then a vectored interrupt is executed. However, if the standby mode is released by the INTW, a vectored interrupt is not generated; therefore the operations identical to (a) above are performed.

7.4 Selection of Mask Option

In the μPD753304 standby function, the wait time after the STOP mode is released by an interrupt generation can be selected from the follwoing two types by the mask option.

<1> 512/f_{CC} (142 μs: 3.6-MHz operation)

<2> No wait

7.5 Application of Standby Mode

Use the standby mode in the following procedure:

<1> Detect the cause that sets the standby mode, such as power failure by interrupt input or port input.
<2> Process the I/O ports (process so that the current dissipation is minimized).
It is important not to open the input port. Be sure to input a low or high level to the input port.
<3> Specify an 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 processing is performed or not).
<5> Specify the CPU clock to be used after the standby mode has been released (to change the clock, make sure that the necessary machine cycles elapse before the standby mode is set).
<6> Select the wait time after the standby mode has been released (specify by mask option).
<7> Set the standby mode (by using the STOP or HALT instruction).

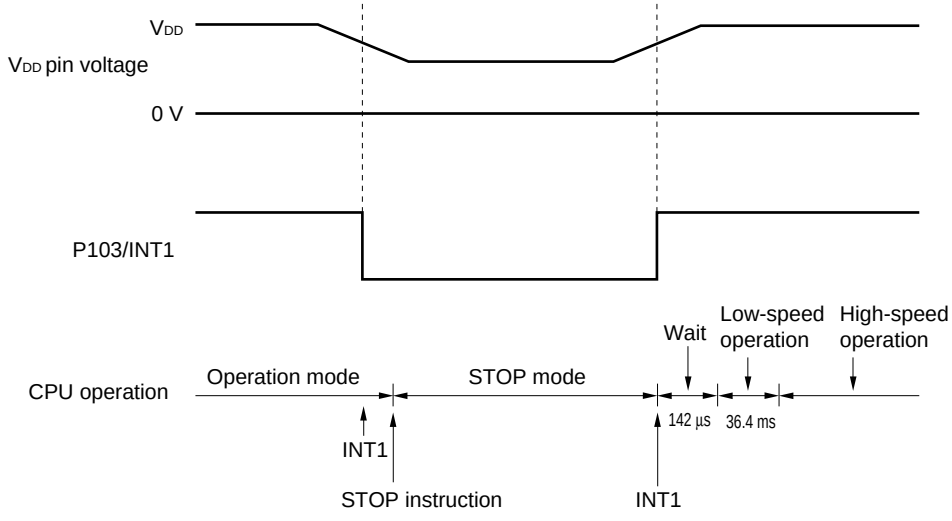
By using the standby mode in combination with the system clock switching function, low current dissipation and low-voltage operation can be realized.

(1) Application example of STOP mode (f_{CC} = 3.6 MHz)

<When using the STOP mode under the following conditions>

- The STOP mode is set at the falling edge of INT1 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 dissipation 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 after the STOP mode has been released should be selected 142 μs by the mask option.
- A wait time of 36.4 ms elapses until the power supply stabilizes after the STOP mode has been released. The P103/INT1 pin is checked two times to eliminate chattering.

<Timing chart>



<Program example>

(INT1 processing program, MBE = 0)

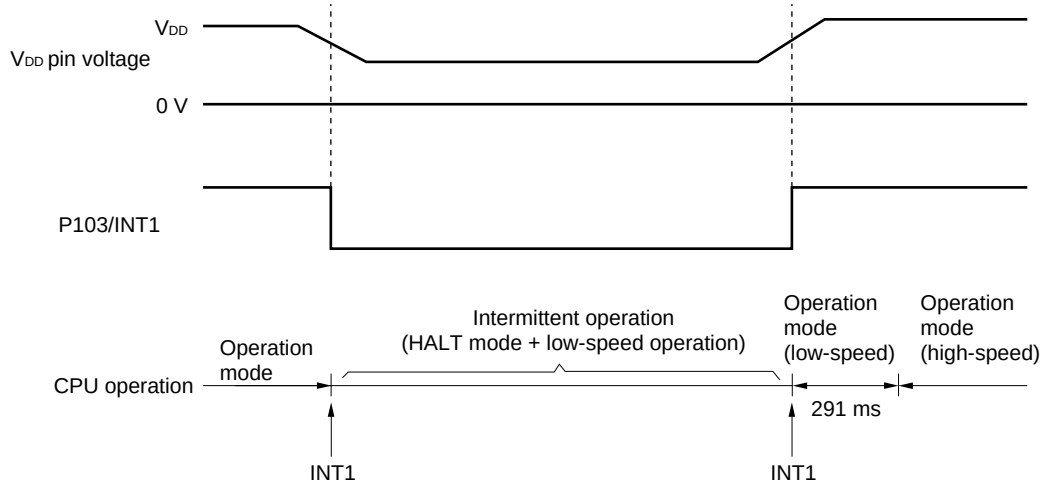
```
VSUB1:  SKT     PORT10.3        ;  P103 = 1 ?
        BR      PDOWN           ;  Power down
        SET1    BTM.3           ;  Power on
WAIT:   SKT     IRQBT           ;  Waits for 36.4 ms
        BR      WAIT
        SKT     PORT10.3        ;  Checks chattering
        BR      PDOWN
        MOV     A, #0011B
        MOV     PCC, A          ;  Sets high-speed mode
      ( MOV     XA, #xxH        )  ;  Sets port mode register
      ( MOV     PMGm, XA        )
        SET1    IM1.0
        EI      IEBT
        EI      IET0
        RETI
PDOWN:  MOV     A, #0           ;  Lowest-speed mode
        MOV     PCC, A
        MOV     XA, #00H
        MOV     LCDM, XA        ;  LCD display off
        MOV     LCDC, A
        MOV     PMGD, XA        ;  I/O port in high-impedance state
        DI      IEBT            ;  Disables INTBT and INTT0
        DI      IET0
        MOV     A, #1011B
        MOV     BTM, A          ;  Wait time ≒ 36.4 ms
        CLR1    IM1.0
        NOP
        STOP                    ;  Sets STOP mode
        NOP
        RETI
```

(2) Application example of HALT mode (f_{CC} = 3.6 MHz)

<To perform intermittent operation under the following conditions>

- The system clock (f_{CT} = 47 kHz) is switched to the subsystem clock at the falling edge of INT1.
- Main system clock oscillation is stopped, and HALT mode is set.
- In the standby mode, an intermittent operation is performed at 0.35-sec interval.
- The system clock is switched to the main system clock again at the rising edge of INT1.
- INTBT is not used.
- After the HALT mode is released, wait 291 ms for safety of power supply.
  Moreover, check the P103/INT1 pin two times to eliminate chattering.

<Timing chart>



<Program example>

```
(Initial setting)
            MOV     A, #0011B
            MOV     PCC, A          ;  High-speed mode
            MOV     XA, #05
            MOV     WM, XA          ;  Subsystem clock
            EI      IE1
            EI      IEW
            EI                      ;  Enables interrupt
(Main routine)
            SKT     PORT10.3        ;  Power supply OK?
            HALT                    ;  Power down mode
            NOP                     ;  Power supply OK?
            SKTCLR  IRQW            ;  0.35-sec flag?
            BR      MAIN            ;  NO
            CALL    WATCH           ;  Watch subroutine
MAIN:       .
            .
            .
            .
            .

(INT1 processing routine)
VINT1:      SKT     PORT10.3        ;  Power supply OK?, MBE = 0
            BR      PDOWN
            CLR1    SCC.3           ;  Main system clock starts oscillation
            MOV     A, #1000B
            MOV     BTM, A
WAIT1:      SKT     IRQBT           ;  Waits for 291 ms
            BR      WAIT1
            SKT     PORT10.3        ;  Checks chattering
            BR      PDOWN
            CLR1    SCC.0           ;  Switches to main system clock
            SET1    IM1.0
            RETI
PDOWN:      MOV     XA, #00H
            MOV     LCDM, XA        ;  LCD display off
            MOV     LCDC, A
            SET1    SCC.0           ;  Switches to subsystem clock
            MOV     A, #6
```

```
WAIT2:  INCS   A        ;  Waits for 20 machine cycles or more[Note]
        BR     WAIT2
        CLR1   IM1.0
        SET1   SCC.3    ;  Main system clock stops oscillation
        RETI
```

Note For how to switch the system clock and CPU clock, refer to **5.2.3 Setting of system clock and CPU clock**.

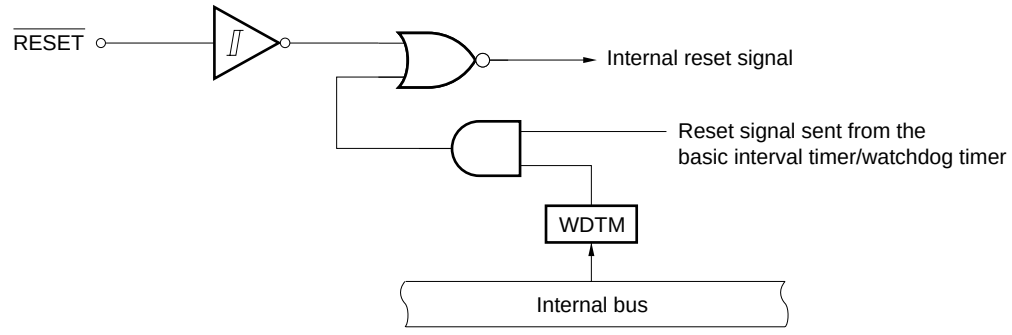
Caution To switch the system clock from the main system clock to the subsystem clock, wait until the oscillation of the subsystem clock is stabilized.

[MEMO]

CHAPTER 8 RESET FUNCTION

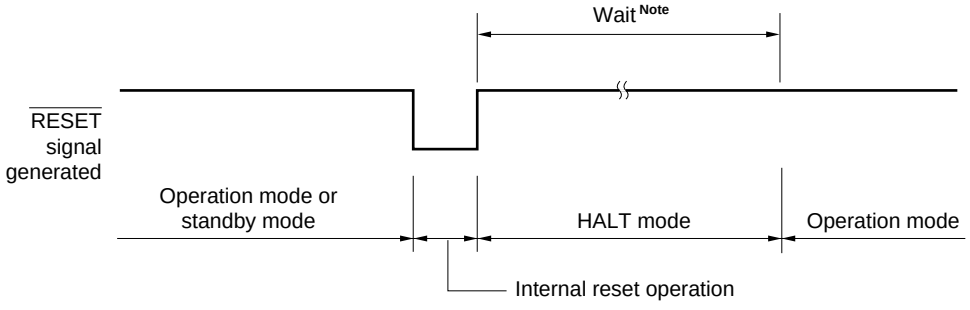
There are two reset inputs: external reset signal ($\overline{\text{RESET}}$) and reset signal sent from the basic interval timer/watchdog timer. When either one of the reset signals are input, an internal reset signal is generated. Figure 8-1 shows the circuit diagram of the above two inputs.

Figure 8-1. Configuration of Reset Function



Generation of the $\overline{\text{RESET}}$ signal causes each hardware to be initialized as listed in Table 8-1. Figure 8-2 shows the timing chart of the reset operation.

Figure 8-2. Reset Operation by $\overline{\text{RESET}}$ Signal Generation



Note 56/fcc (15.6 μs: 3.6-MHz operation)

Table 8-1. Status of Each Hardware After Reset (1/2)

Hardware		$\overline{\text{RESET}}$ signal generation in the standby mode	$\overline{\text{RESET}}$ signal generation in operation
Program counter (PC)		Sets the low-order 4 bits of program memory's address 0000H to the PC11 through PC8 and the contents of address 0001H to the PC7 through PC0.	Sets the low-order 4 bits of program memory's address 0000H to the PC11 through PC8 and the contents of address 0001H to the PC7 through PC0.
PSW	Carry flag (CY)	Held	Undefined
	Skip flag (SK0 to SK2)	0	0
	Interrupt status flag (IST0)	0	0
	Bank enable flag (MBE, RBE)	Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.	Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.
Stack pointer (SP)		Undefined	Undefined
Stack bank selection register (SBS)		1000B	1000B
Data memory (RAM)		Held	Undefined
General-purpose register (X, A, H, L, D, E, B, C)		Held	Undefined
Bank selection register (MBS, RBS)		0, 0	0, 0
Basic interval timer/watchdog timer	Counter (BT)	Undefined	Undefined
	Mode register (BTM)	0	0
	Watchdog timer enable flag (WDTM)	0	0
Timer counter (T0)	Counter (T0)	0	0
	Modulo register (TMOD0)	FFH	FFH
	Mode register (TM0)	0	0
Watch timer	Mode register (WM)	0	0
Clock generator, clock output circuit	Processor clock control register (PCC)	0	0
	System clock control register (SCC)	0	0
	Clock output mode register (CLOM)	0	0
Sub-oscillator control register (SOS)		0	0
LCD controller/ driver	Display mode register (LCDM)	0	0
	Display control register (LCDC)	0	0
	LCD/port selection register (LPS)	0	0
Interrupt function	Interrupt request flag (IRQxxx)	Reset (0)	Reset (0)
	Interrupt enable flag (IExxx)	0	0
	Interrupt priority selection register (IPS)	0	0
	INT1 mode register (IM1)	0	0

Table 8-1. Status of Each Hardware After Reset (2/2)

Hardware		$\overline{\text{RESET}}$ signal generation in the standby mode	$\overline{\text{RESET}}$ signal generation in operation
Digital port	Output buffer (P30 to P33)	On	On
	Output buffer (P80 to P83, P100 to P103)	Off	Off
	Output latch (P30 to P32, P80 to P83, P100 to P103)	Clear (0)	Clear (0)
	Output latch (P33)	Set (1)	Set (1)
	I/O mode register (PMGA)	0FH	0FH
	I/O mode registers (PMGC, D)	00H	00H
	Pull-up resistor specification register (POGB)	01H	01H

[MEMO]

CHAPTER 9 MASK OPTION

9.1 Mask Option of $\overline{\text{RESET}}$ Pin

The $\overline{\text{RESET}}$ pin can specify incorporation of a 60-kΩ (typ.) pull-up resistor by the mask option.

★ 9.2 Mask Option of LCD Display Mode

Though the μPD753304 incorporates a split resistor R for the LCD drive power supply, the V_{LC0} to V_{LC2} pins (LCD drive power supply) and BIAS pin (external split resistor cut pin) are not incorporated as external pins. The following three mask options can be selected to support each display mode.

<1> Static mode (Short between BIAS and V_{LC0} , Open between V_{LC0} and V_{LC1})
<2> 1/2 bias mode (Short between BIAS and V_{LC0}, Short between V_{LC1} and V_{LC2})
<3> 1/3 bias mode (Short between BIAS and V_{LC0})

9.3 Mask Option of Standby Function

In the standby function of the μPD753304, a wait time is selected by the mask option. The wait time means the required time for the CPU to return to the normal operation mode after the STOP mode is released by an interrupt generation (refer to **7.2 Standby Mode Release**).

Either of the following can be selected for the wait time.

<1> 512/f_{CC} (142 μs: 3.6-MHz operation)
<2> No wait

★ 9.4 Mask Option of Port 3

Input/output mode after reset can be set with mask options of Port 3. The option can be selected among three types shown in table 9-1.

Table 9-1. Status at Reset by Port 3 Mask Option

Pin name	Status after reset		
	Mask option <1>	Mask option <2>	Mask option <3>
P30/PCL	Input	Low-level output	Low-level output
P31/BUZ			
P32			
P33			High-level output

[MEMO]

CHAPTER 10 INSTRUCTION SET

The instruction set of the μPD753304 is based on the instruction set of the 75X Series and therefore, maintains compatibility with the 75X Series, but has some improved features. They are:

(1) Bit manipulation instructions for various applications
(2) Efficient 4-bit manipulation instructions
(3) 8-bit manipulation instructions comparable to those of 8-bit microcontrollers
(4) GETI instruction for reducing program size
(5) String-effect and base number adjustment instructions enhancing program efficiency
(6) Table reference instructions ideal for successive reference
(7) 1-byte relative branch instruction
(8) Easy-to-understand, well-organized NEC's standard mnemonics

For the addressing modes applicable to data memory manipulation and the register banks valid for instruction execution, refer to **3.2 Bank Configuration of General-Purpose Registers**.

10.1 Unique Instructions

This section describes the unique instructions of the μPD753304's instruction set.

10.1.1 GETI instruction

The GETI instruction converts the following instructions into 1-byte instructions:

(a) Subroutine call instruction to 4 Kbytes space (0000H-0FFFH)
(b) Branch instruction to 4 Kbytes space (0000H-0FFFH)
(c) Any 2-byte, 2-machine cycle instruction (except BRCB and CALLF instructions)
(d) Combination of two 1-byte instructions

The GETI instruction references a table at addresses 0020H through 007FH of the program memory and executes the referenced 2-byte data as an instruction of (a) to (d). Therefore, 48 types of instructions can be converted into 1-byte instructions.

If instructions that are frequently used are converted into 1-byte instructions by using this GETI instruction, the number of bytes of the program can be substantially decreased.

10.1.2 Bit manipulation instruction

The μPD753304 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. Three types of bit manipulation addressing can be used. The bits manipulated in each addressing are shown in Table 10-1.

Table 10-1. Types of Bit Manipulation Addressing and Specification Range

Addressing	Peripheral hardware that can be manipulated	Addressing range of bit that can be manipulated
fmem.bit	RBE, MBE, IST1, IST0, SCC, IExxx, IRQxxx	FB0H through FBFH
	PORT3, 8, 10	FF0H through FFFH
pmem.@L	PORT3, 8, 10	FC0H through FFFH
@H + mem.bit	All peripheral hardware units that can be manipulated bit-wise	All bits of memory bank specified by MB that can be manipulated bit-wise

Remarks 1. xxx: 1, BT, T0, W

2. MB = MBE·MBS

10.1.3 String-effect instruction

The μPD753304 has the following two types of string-effect instructions:

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

“String effect” means locating these two types of instructions at contiguous addresses.

Example
```
A0  : MOV  A, #0
A1  : MOV  A, #1
XA7 : MOV  XA, #07
```

When string-effect instructions are arranged as shown in this example, and if the address executed first is A0, the two instructions following this address are replaced with the NOP instructions. If the address executed first is A1, the following one instruction is replaced with the NOP instruction. In other words, only the instruction that is executed first is valid, and all the string-effect instructions that follow are processed as NOP instructions.

By using these string-effect instructions, constants can be efficiently set to the accumulator (A register or register pair XA) and data pointer (register pair HL).

10.1.4 Base number adjustment instruction

Some application requires that the result of addition or subtraction of 4-bit data (which is carried out in binary number) be converted into a decimal number or into a number with a base of 6, such as time.

Therefore, the μPD753304 is provided with base number adjustment instructions that adjusts the result of addition or subtraction of 4-bit data into a number with any base.

(a) Base adjustment of result of addition

Where the base number to which the result of addition executed is to be adjusted is m, the contents of the accumulator and memory (HL) are added in the following combination, and the result of addition 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 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.

Example To add accumulator and memory in decimal

```
ADDS   A, #6
ADDC   A, @HL    ; A, CY ← A + (HL) + CY
ADDS   A, #10
       :
```

(b) Base adjustment of result of subtraction

Where the base number to 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, however, 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.

10.1.5 Skip instruction and number of machine cycles required for skipping

The instruction set of the μPD753304 configures a program where instructions may be or may not be skipped if a given condition is satisfied.

If a skip condition is satisfied when a skip instruction is executed, the instruction next to the skip instruction is skipped and the instruction after the next is executed.

When a skip occurs, the number of machine cycles required for skipping is:

(a) If the instruction that follows the skip instruction (i.e., the instruction to be skipped) is a 3-byte instruction (BR !addr, BRA !addr1, CALL !addr, or CALLA !addr1 instruction): 2 machine cycles

(b) Instruction other than (a): 1 machine cycle

10.2 Instruction Sets and their Operations

(1) Expression formats and description methods of operands

The operand is described in the operand column of each instruction in accordance with the description method for the operand expression format of the instruction. For details, refer to **"RA75X Assembler Package Users' Manual-Language (U12385E)"**. If there are several elements, one of them is selected. Capital letters and the + and – symbols are key words and are described as they are.

For immediate data, appropriate numbers and labels are described.

Instead of the labels such as mem, fmem, pmem, and bit, the symbols of the register flags shown in Figure 3-7 can be described. However, there are restrictions in the labels that can be described for fmem and pmem. For details, see **Table 3-1 Addressing Mode** and **Figure 3-7 μPD753304 I/O Map**.

Expression format	Description method
reg reg1	X, A, B, C, D, E, H, L X, B, C, D, E, H, L
rp rp1 rp2 rp' rp'1	XA, BC, DE, HL BC, DE, HL BC, DE XA, BC, DE, HL, XA', BC', DE', HL' BC, DE, HL, XA', BC', DE', HL'
rpa rpa1	HL, HL+, HL–, DE, DL DE, DL
n4 n8	4-bit immediate data or label 8-bit immediate data or label
mem bit	8-bit immediate data or label**Note** 2-bit immediate data or label
fmem pmem	FB0H-FBFH, FF0H-FFFH immediate data or label FC0H-FFFH immediate data or label
addr, addr1 caddr faddr	000H-FFFH immediate data or label 12-bit immediate data or label 11-bit immediate data or label
taddr	20H-7FH immediate data (where bit0 = 0) or label
PORTn IExxx RBn MBn	PORT3, PORT8, PORT10 IEBT, IET0, IE1, IEW RB0-RB3 MB0, MB1, MB15

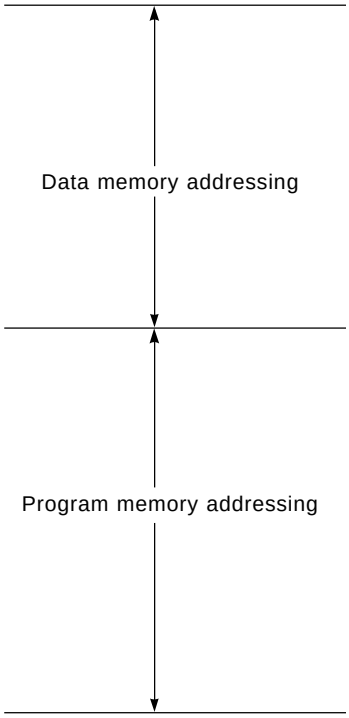
Note mem can be only used for even address in 8-bit data processing.

(2) Legend in explanation of operation

A	: A register; 4-bit accumulator
B	: B register
C	: C register
D	: D register
E	: E register
H	: H register
L	: L register
X	: X register
XA	: XA register pair; 8-bit accumulator
BC	: BC register pair
DE	: DE register pair
HL	: HL register pair
XA’	: XA’ expanded register pair
BC’	: BC’ expanded register pair
DE’	: DE’ expanded register pair
HL’	: HL’ expanded register pair
PC	: Program counter
SP	: Stack pointer
CY	: Carry flag; bit accumulator
PSW	: Program status word
MBE	: Memory bank enable flag
RBE	: Register bank enable flag
PORTn	: Port n (n = 3, 8, 10)
IME	: Interrupt master enable flag
IPS	: Interrupt priority selection register
IExxx	: Interrupt enable flag
RBS	: Register bank selection register
MBS	: Memory bank selection register
PCC	: Processor clock control register
.	: Separation between address and bit
(xx)	: The contents addressed by xx
xxH	: Hexadecimal data

(3) Explanation of symbols under addressing area column

*1	MB = MBE • MBS (MBS = 0, 1, 15)
*2	MB = 0
*3	MBE = 0 : MB = 0 (000H-07FH) MB = 15 (F80H-FFFH) MBE = 1 : MB = MBS (MBS = 0, 1, 15)
*4	MB = 15, fmem = FB0H-FBFH, FF0H-FFFH
*5	MB = 15, pmem = FC0H-FFFH
*6	addr = 000H-FFFH
*7	addr = (Current PC) – 15 to (Current PC) – 1 (Current PC) + 2 to (Current PC) + 16
	addr1 = (Current PC) – 15 to (Current PC) – 1 (Current PC) + 2 to (Current PC) + 16
*8	caddr = 000H-FFFH
*9	faddr = 0000H-07FFH
*10	taddr = 0020H-007FH
*11	addr1 = 000H-FFFH



Remarks 1. MB indicates memory bank that can be accessed.

2. In *2, MB = 0 independently of how MBE and MBS are set.

3. In *4 and *5, MB = 15 independently of how MBE and MBS are set.

4. *6 to *11 indicate the areas that can be addressed.

(4) Explanation of number of machine cycles column

S denotes the number of machine cycles required by skip operation when a skip instruction is executed. The value of S varies as follows.

- When no skip is made: S = 0
- When the skipped instruction is a 1- or 2-byte instruction: S = 1
- When the skipped instruction is a 3-byte instruction[Note] : S = 2

Note 3-byte instruction: BR !addr, BRA !addr1, CALL !addr or CALLA !addr1 instruction

Caution The GETI instruction is skipped in one machine cycle.

One machine cycle is equal to one cycle (= t_{CY}) of CPU clock Φ; time can be selected from among four types by setting PCC (see **Figure 5-13 Processor Clock Control Register Format**).

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	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'		

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	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'1, CY ← rp'1+XA+CY		
	SUBS	A, @HL	1	1+S	A ← A–(HL)	*1	borrow
		XA, rp'	2	2+S	XA ← XA–rp'		borrow
		rp'1, XA	2	2+S	rp'1 ← rp'1–XA		borrow
	SUBC	A, @HL	1	1	A, CY ← A–(HL)–CY	*1	
		XA, rp'	2	2	XA, CY ← XA–rp'–CY		
		rp'1, XA	2	2	rp'1, CY ← rp'1–XA–CY		
	AND	A, #n4	2	2	A ← A ∧ n4		
		A, @HL	1	1	A ← A ∧ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∧ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∧ XA		
	OR	A, #n4	2	2	A ← A ∨ n4		
		A, @HL	1	1	A ← A ∨ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∨ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∨ XA		
	XOR	A, #n4	2	2	A ← A ∀ n4		
		A, @HL	1	1	A ← A ∀ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∀ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∀ XA		

Note “0” must be set to B register.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Accumulator manipulation	RORC	A	1	1	$CY \leftarrow A_0$, $A_3 \leftarrow CY$, $A_{n-1} \leftarrow A_n$		
	NOT	A	2	2	$A \leftarrow \overline{A}$		
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	$CY \leftarrow \overline{CY}$		
Memory bit manipulation	SET1	mem.bit	2	2	(mem.bit)← 1	*3	
		fmem.bit	2	2	(fmem.bit) ← 1	*4	
		pmem.@L	2	2	$(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) \leftarrow 1$	*5	
		@H+mem.bit	2	2	$(H+mem_{3-0}.bit) \leftarrow 1$	*1	
	CLR1	mem.bit	2	2	(mem.bit)← 0	*3	
		fmem.bit	2	2	(fmem.bit) ← 0	*4	
		pmem.@L	2	2	$(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) \leftarrow 0$	*5	
		@H+mem.bit	2	2	$(H+mem_{3-0}.bit) \leftarrow 0$	*1	
	SKT	mem.bit	2	2+S	Skip if (mem.bit) = 1	*3	(mem.bit) = 1
		fmem.bit	2	2+S	Skip if (fmem.bit) = 1	*4	(fmem.bit) = 1
		pmem.@L	2	2+S	Skip if $(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) = 1$	*5	(pmem.@L) = 1
		@H+mem.bit	2	2+S	Skip if $(H+mem_{3-0}.bit) = 1$	*1	(@H+mem.bit) = 1
	SKF	mem.bit	2	2+S	Skip if (mem.bit) = 0	*3	(mem.bit) = 0
		fmem.bit	2	2+S	Skip if (fmem.bit) = 0	*4	(fmem.bit) = 0
		pmem.@L	2	2+S	Skip if $(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) = 0$	*5	(pmem.@L) = 0
		@H+mem.bit	2	2+S	Skip if $(H+mem_{3-0}.bit) = 0$	*1	(@H+mem.bit) = 0
	SKTCLR	fmem.bit	2	2+S	Skip if (fmem.bit) = 1 and clear	*4	(fmem.bit) = 1
		pmem.@L	2	2+S	Skip if $(pmem_{7-2}+L_{3-2}.bit(L_{1-0})) = 1$ and clear	*5	(pmem.@L) = 1
		@H+mem.bit	2	2+S	Skip if $(H+mem_{3-0}.bit) = 1$ and clear	*1	(@H+mem.bit) = 1

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Memory bit manipulation	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[Note 1]	addr	–	–	PC$_{11-0}$ ← addr Select appropriate instruction from among BR !addr, BRCB !caddr, and BR $addr according to the assembler being used.	*6	
		addr1	–	–	PC$_{11-0}$ ← addr1 Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr, and BR $addr1 according to the assembler being used.	*11	
		!addr	3	3	PC$_{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[Note 2]	*6	
		BCXA	2	3	PC$_{11-0}$ ← BCXA[Note 2]	*6	
	BRA[Note 1]	!addr1	3	3	PC$_{11-0}$ ← addr1	*11	
	BRCB	!caddr	2	2	PC$_{11-0}$ ← caddr$_{11-0}$	*8	

Notes 1. The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

2. “0” must be set to B register.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Subroutine stack control	CALLA[Note]	!addr1	3	3	(SP–2) ← x, x, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← addr1, SP ← SP–6	*11	
	CALL[Note]	!addr	3	3	(SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) ← PC_{11-0} PC_{11-0} ← addr, SP ← SP–4	*6	
				4	(SP–2) ← x, x, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← addr, SP ← SP–6		
	CALLF[Note]	!faddr	2	2	(SP–3) ← MBE, RBE, 0, 0 (SP–4) (SP–1) (SP–2) ← PC_{11-0} PC_{11-0} ← 0+faddr, SP ← SP–4	*9	
				3	(SP–2) ← x, x, MBE, RBE (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 PC_{11-0} ← 0+faddr, SP ← SP–6		
	RET[Note]		1	3	PC_{11-0} ← (SP) (SP+3) (SP+2) MBE, RBE, 0, 0 ← (SP+1), SP ← SP+4		
					x, x, 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
					0, 0, 0, 0 ← (SP+1) PC_{11-0} ← (SP) (SP+3) (SP+2) x, x, MBE, RBE ← (SP+4) 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		
	PUSH	rp	1	1	(SP–1) (SP–2) ← rp, SP ← SP–2		
		BS	2	2	(SP–1) ← MBS, (SP–2) ← RBS, SP ← SP–2		
	POP	rp	1	1	rp ← (SP+1) (SP), SP ← SP+2		
		BS	2	2	MBS ← (SP+1), RBS ← (SP), SP ← SP+2		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction group	Mnemonic	Operand	Number of bytes	Number of machine cycles	Operation	Addressing area	Skip condition
Interrupt control	EI		2	2	IME (IPS.3) ← 1		
		IExxx	2	2	IExxx ← 1		
	DI		2	2	IME (IPS.3) ← 0		
		IExxx	2	2	IExxx ← 0		
Input/output	IN[Note 1]	A, PORTn	2	2	A ← PORTn (n = 3, 8, 10)		
	OUT[Note 1]	PORTn, A	2	2	PORTn ← A (n = 3, 8, 10)		
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, 1, 15)		
	GETI[Note 2, 3]	taddr	1	3	• When TBR instruction PC_{11-0} ← (taddr) $_{3-0}$ + (taddr+1)	*10	
					• When 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		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed		Depending on the reference instruction
				3	• When TBR instruction PC_{11-0} ← (taddr) $_{3-0}$ + (taddr+1)	*10	
				4	• When TCALL instruction (SP–6) (SP–3) (SP–4) ← PC_{11-0} (SP–5) ← 0, 0, 0, 0 (SP–2) ← x, x, MBE, RBE PC_{11-0} ← (taddr) $_{3-0}$ + (taddr+1) SP ← SP–6		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr+1) instruction is executed.		Depending on the reference instruction

Notes 1. While the IN instruction and OUT instruction are being executed, the MBE must be set to 0 or 1, and MBS must be set to 15.

2. The TBR and TCALL instructions are the table definition assembler pseudo-instructions of the GETI instruction.

3. The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

10.3 Op Code of Each Instruction

(1) Description of symbol of op code

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

reg, reg1

P_2	P_1	P_0	reg-pair
0	0	0	XA
0	0	1	XA’
0	1	0	HL
0	1	1	HL’
1	0	0	DE
1	0	1	DE’
1	1	0	BC
1	1	1	BC’

rp’, rp’1

Q_2	Q_1	Q_0	addressing
0	0	0	@HL
0	1	0	@HL+
0	1	1	@HL–
1	0	0	@DE
1	0	1	@DL

@rpa, @rpa1

P_2	P_1	reg-pair
0	0	XA
0	1	HL
1	0	DE
1	1	BC

rp2, rp1, rp

N_5	N_2	N_1	N_0	IExxx
0	0	0	0	IEBT
0	0	1	0	IEW
0	1	0	0	IET0
1	1	1	0	IE1

I_n : immediate data for n4 or n8
D_n : immediate data for mem
B_n : immediate data for bit
N_n : immediate data for n or IExxx
T_n : immediate data for taddr x 1/2
A_n : immediate data for [relative address distance from branch destination address (2 – 16)] – 1
S_n : immediate data for 1’s complement of [relative address distance from branch destination address (15 – 1)]

(2) Op code for bit manipulation addressing

∗1 in the operand column indicates the following three types:

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

The second byte ∗2 of the op code corresponding to the above addressing is as follows:

∗1	2nd byte of op code								Accessible bit
fmem.bit	1	0	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FB0H-FBFH that can be manipulated
	1	1	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FF0H-FFFH that can be manipulated
pmem.@L	0	1	0	0	G_3	G_2	G_1	G_0	Bit of FC0H-FFFH that can be manipulated
@H+mem.bit	0	0	B_1	B_0	D_3	D_2	D_1	D_0	Bit of accessible memory bank that can be manipulated

B_n : immediate data for bit

F_n : immediate data for fmem (indicates low-order 4 bits of address)

G_n : immediate data for pmem (indicates bits 5-2 of address)

D_n : immediate data for mem (indicates low-order 4 bits of address)

Instruction group	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Transfer	MOV	A, #n4	0 1 1 1 I_3 I_2 I_1 I_0		
		reg1, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 1 R_2 R_1 R_0	
		rp, #n8	1 0 0 0 1 P_2 P_1 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @rpa1	1 1 1 0 0 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 0	
		@HL, A	1 1 1 0 1 0 0 0		
		@HL, XA	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 0	
		A, mem	1 0 1 0 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		mem, A	1 0 0 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		mem, XA	1 0 0 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg	1 0 0 1 1 0 0 1	0 1 1 1 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 1 1 P_2 P_1 P_0	
		reg1, A	1 0 0 1 1 0 0 1	0 1 1 1 0 R_2 R_1 R_0	
		rp'1, XA	1 0 1 0 1 0 1 0	0 1 0 1 0 P_2 P_1 P_0	
	XCH	A, @rpa1	1 1 1 0 1 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 1	
		A, mem	1 0 1 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg1	1 1 0 1 1 R_2 R_1 R_0		
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 0 P_2 P_1 P_0	
Table reference	MOVT	XA, @PCDE	1 1 0 1 0 1 0 0		
		XA, @PCXA	1 1 0 1 0 0 0 0		
		XA, @BCDE	1 1 0 1 0 1 0 1		
		XA, @BCXA	1 1 0 1 0 0 0 1		
Bit transfer	MOV1	CY, ∗1	1 0 1 1 1 1 0 1	*2	
		∗1, CY	1 0 0 1 1 0 1 1	*2	

Instruction group	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Operation	ADDS	A, #n4	0 1 1 0 I_3 I_2 I_1 I_0		
		XA, #n8	1 0 1 1 1 0 0 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @HL	1 1 0 1 0 0 1 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 1 0 0 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 1 0 0 0 P_2 P_1 P_0	
	ADDC	A, @HL	1 0 1 0 1 0 0 1		
		XA, rp’	1 0 1 0 1 0 1 0	1 1 0 1 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 1 0 1 0 P_2 P_1 P_0	
	SUBS	A, @HL	1 0 1 0 1 0 0 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 1 1 0 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 1 1 0 0 P_2 P_1 P_0	
	SUBC	A, @HL	1 0 1 1 1 0 0 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 1 1 1 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 1 1 1 0 P_2 P_1 P_0	
	AND	A, #n4	1 0 0 1 1 0 0 1	0 0 1 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 1 0 0 0 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 0 0 1 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 0 0 1 0 P_2 P_1 P_0	
	OR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 0 0 0 0 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 0 1 0 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 0 1 0 0 P_2 P_1 P_0	
	XOR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 1 0 0 0 0		
		XA, rp’	1 0 1 0 1 0 1 0	1 0 1 1 1 P_2 P_1 P_0	
		rp’1, XA	1 0 1 0 1 0 1 0	1 0 1 1 0 P_2 P_1 P_0	
Accumulator manipulation	RORC	A	1 0 0 1 1 0 0 0		
	NOT	A	1 0 0 1 1 0 0 1	0 1 0 1 1 1 1 1	

Instruction group	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Increment/ decrement	INCS	reg	1 1 0 0 0 R_2 R_1 R_0		
		rp1	1 0 0 0 1 P_2 P_1 0		
		@HL	1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0	
		mem	1 0 0 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
	DECS	reg	1 1 0 0 1 R_2 R_1 R_0		
		rp'	1 0 1 0 1 0 1 0	0 1 1 0 1 P_2 P_1 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 manipulation	SET1	CY	1 1 1 0 0 1 1 1		
	CLR1	CY	1 1 1 0 0 1 1 0		
	SKT	CY	1 1 0 1 0 1 1 1		
	NOT1	CY	1 1 0 1 0 1 1 0		
Memory bit manipulation	SET1	mem.bit	1 0 B_1 B_0 0 1 0 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 0 1 1 1 0 1	*2	
	CLR1	mem.bit	1 0 B_1 B_0 0 1 0 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 0 1 1 1 0 0	*2	
	SKT	mem.bit	1 0 B_1 B_0 0 1 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 1 1 1 1 1 1	*2	
	SKF	mem.bit	1 0 B_1 B_0 0 1 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		[*1]	1 0 1 1 1 1 1 0	*2	
	SKTCLR	[*1]	1 0 0 1 1 1 1 1	*2	
	AND1	CY, [*1]	1 0 1 0 1 1 0 0	*2	
	OR1	CY, [*1]	1 0 1 0 1 1 1 0	*2	
	XOR1	CY, [*1]	1 0 1 1 1 1 0 0	*2	

Instruction group	Mnemonic	Operand	Op code		
			B_1	B_2	B_3
Branch	BR	!addr	1 0 1 0 1 0 1 1	0 0 ← addr →	
		$addr (+16) to (+2)	0 0 0 0 A_3 A_2 A_1 A_0		
		$addr (–1) to (–15)	1 1 1 1 S_3 S_2 S_1 S_0		
		PCDE	1 0 0 1 1 0 0 1	0 0 0 0 0 1 0 0	
		PCXA	1 0 0 1 1 0 0 1	0 0 0 0 0 0 0 0	
		BCDE	0 0 0 0 0 1 0 1		
		BCXA	1 0 0 1 1 0 0 1	0 0 0 0 0 0 0 1	
	BRA	!addr1	1 0 1 1 1 0 1 0	0 ← addr1 →	
	BRCB	!caddr	0 1 0 1 ← caddr →		
Subroutine stack control	CALLA	!addr1	1 0 1 1 1 0 1 1	0 ← addr →	
	CALL	!addr	1 0 1 0 1 0 1 1	0 1 ← addr →	
	CALLF	!faddr	0 1 0 0 0 ← faddr →		
	RET		1 1 1 0 1 1 1 0		
	RETS		1 1 1 0 0 0 0 0		
	RETI		1 1 1 0 1 1 1 1		
	PUSH	rp	0 1 0 0 1 P_2 P_1 1		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 1	
	POP	rp	0 1 0 0 1 P_2 P_1 0		
		BS	1 0 0 1 1 0 0 1	0 0 0 0 0 1 1 0	
Input/output	IN	A, PORTn	1 0 1 0 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
	OUT	PORTn, A	1 0 0 1 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
Interrupt control	EI		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 0	
		IExxx	1 0 0 1 1 1 0 1	1 0 N_5 1 1 N_2 N_1 N_0	
	DI		1 0 0 1 1 1 0 0	1 0 1 1 0 0 1 0	
		IExxx	1 0 0 1 1 1 0 0	1 0 N_5 1 1 N_2 N_1 N_0	
CPU control	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		

10.4 Instruction Function and Application

This section describes the functions and applications of the respective instructions. The instructions that can be used and the functions of the instructions differ between the Mk I and Mk II modes of the μPD753304. Read the descriptions on the following pages according to the following guidance:

How to read

( ) : This instruction can be used commonly to all the following:
In Mk I and Mk II modes of the μPD753304

(I) : This instruction can be used only in the Mk I mode of the μPD753304.

(II) : This instruction can be used only in the Mk II mode of the μPD753304.

(I/II) : This instruction can be used commonly in the Mk I and Mk II modes of the μPD753304, but the function may differ between the Mk I and Mk II modes.
In the Mk I mode, read the description under the heading [Mk I mode]. In the Mk II mode, read the description under the heading [Mk II mode].

10.4.1 Transfer instructions

MOV A, #n4

Function: A ← n4 n4 = I_{3-0}: 0-FH

Transfers 4-bit immediate data n4 to the A register (4-bit accumulator). This instruction has a string effect (group A). Therefore, if this instruction is followed by another MOV A, #n4 or MOV XA, #n8 instruction, the instruction following this instruction is treated 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_{3-0} 0-FH

Transfers 4-bit immediate data n4 to A register reg1 (X, H, L, D, E, B, or C).

MOV XA, #n8

Function: XA ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair XA. This instruction has a string effect. Therefore, if this instruction is followed by another MOV XA, #n8 or MOV A, #n4 instruction, the instruction following this instruction is treated as NOP.

MOV HL, #n8

Function: HL ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair HL. This instruction has a string effect. Therefore, if this instruction is followed by another MOV HL, #n8 instruction, the instruction following this instruction is treated as NOP.

MOV rp2, #n8

Function: rp2 ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair rp2 (BC, DE).

MOV A, @HL

MOV A, @HL+

MOV A, @HL–

MOV A, @rpa1

Function: A ← (Register pair specified by operand)
when register pair = HL+ : skip if L = 0
when register pair = HL– : skip if L = FH

Transfers the contents of the data memory addressed by the specified register pair (HL, HL+, HL–, DE, or DL) to the A register.

If autoincrement (HL+) is specified as a register pair, the contents of the L register are automatically incremented by one after the data has been transferred. If the contents of the L register become 0 as a result, the next one instruction is skipped.

If autodecrement (HL–) is specified as a register pair, the contents of the L register are automatically decremented by one after the data has been transferred. If the contents of the L register become FH as a result, the next one instruction is skipped.

MOV XA, @HL

Function: A ← (HL), X ← (HL+1)

Transfers the contents of the data memory addressed by register pair HL to the A register, and the contents of the next memory address to the X register.

If the contents of the L register are a odd number, an address whose least significant bit is ignored is transferred.

Application example

To transfer the data at addresses 3EH and 3FH to register pair XA

```
MOV HL, #3EH
MOV XA, @HL
```

MOV @HL, A

Function: (HL) ← A

Transfers the contents of the A register to the data memory addressed by register pair HL.

MOV @HL, XA

Function: (HL) ← A, (HL+1) ← X

Transfers the contents of the A register to the data memory addressed by register pair HL, and the contents of the X register to the next memory address.

However, if the contents of the L register are a odd number, an address whose least significant bit is ignored is transferred.

MOV A, mem

Function: A ← (mem) mem = D_{7-0}: 00H-FFH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register.

MOV XA, mem

Function: A ← (mem), X ← (mem+1) mem = D_{7-0}: 00H-FEH

Transfers the contents of the data memory addressed by 8-bit immediate data mem to the A register and the contents of the next address to the X register.

The address that can be specified by mem is an even address.

Application example

To transfer the data at addresses 40H and 41H to register pair XA

```
MOV XA, 40H
```

MOV mem, A

Function: (mem) ← A mem = D_{7-0}: 00H-FFH

Transfers the contents of the A register to the data memory addressed by 8-bit immediate data mem.

MOV mem, XA

Function: (mem) ← A, (mem+1) ← X mem = D_{7-0}: 00H-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').

XCH A, @HL

XCH A, @HL+

XCH A, @HL–

XCH A, @rpa1

Function: A ↔ (Register pair specified by operand)
when register pair = HL+ : skip if L = 0
when register pair = HL– : skip if L = FH

Exchanges the contents of the A register with the contents of the data memory addressed by the specified register pair (HL, HL+, HL–, DE, or DL).

If autoincrement (HL+) is specified as a register pair, the contents of the L register are automatically incremented by one after the data have been exchanged. If the increment result is 0, the next one instruction is skipped.

If autodecrement (HL–) is specified as a register pair , the contents of the L register are automatically decremented by one after the data have been exchanged. If the decrement result is FH, the next one instruction is skipped.

Application example

To exchange the data at data memory addresses 20H through 2FH with the data at addresses 30H through 3FH

```
        SEL   MB0
        MOV   D, #2
        MOV   HL, #30H
LOOP:   XCH   A, @HL    ; A ↔ (3x)
        XCH   A, @DL    ; A ↔ (2x)
        XCH   A, @HL+   ; A ↔ (3x)
        BR    LOOP
```

XCH XA, @HL

Function: A ↔ (HL), X ↔ (HL+1)

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL, and the contents of the X register with the contents of the next address.

If the contents of the L register are odd numbers, however, an address whose least significant bit is ignored is specified.

XCH A, mem

Function: A ↔ (mem) mem = D_{7-0}: 00H-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-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').

10.4.2 Table reference instructions

MOVT XA, @PCDE

Function: XA ← ROM (PC_{11-8}+DE)

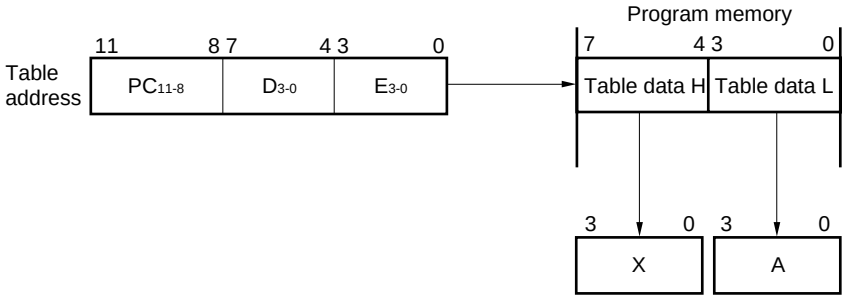
Transfers the low-order 4 bits of the table data in the program memory addressed when the low-order 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair DE to the A register, and the high-order 4 bits to the X register.

The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction).

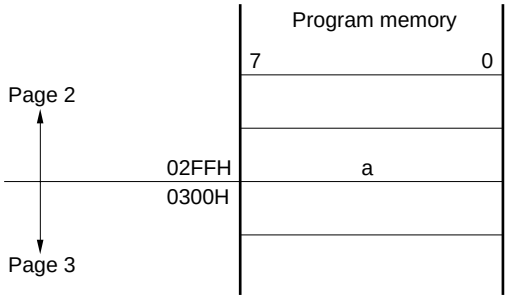
The program counter is not affected by execution of this instruction.

This instruction is useful for successively referencing table data.



Caution

The MOVT XA, @PCDE instruction usually references the table data in page where the instruction exists. If the instruction is at address xxFFH, however, the table data in the page where the instruction exists is not referenced, but the table data in the next page is referenced.



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 xxF0H through xxFFH to data memory addresses 30H through 4FH

```
SUB:   SEL    MB0
       MOV    HL, #30H      ; HL ← 30H
       MOV    DE, #0F0H     ; DE ← F0H
LOOP:  MOVT   XA, @PCDE     ; XA ← table data
       MOV    @HL, XA       ; (HL) ← XA
       INCS   HL            ; HL ← HL+2
       INCS   HL
       INCS   E             ; E ← E+1
       BR     LOOP
       RET
       ORG    xxF0H
       DB     xxH, xxH, ···  ; table data
```

MOVT XA, @PCXA

Function: XA ← ROM (PC_{11-8}+XA)

Transfers the low-order 4 bits of the table data in the program memory addressed when the low-order 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 high-order 4 bits to the X register.

The table address is determined by the contents of the PC when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction).

The PC is not affected by execution of this instruction.

Caution

If an instruction exists at address xxFFH, the table data of the next page is transferred, in the same manner as MOVT XA, @PCDE.

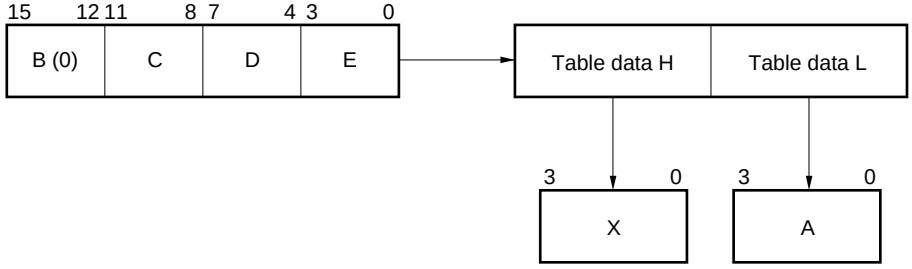
MOVT XA, @BCDE

Function: XA ← ROM (BCDE)

Transfers the low-order 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 high-order 4 bits to the X register.

In the μPD753304, register B is invalid. Set 0000B to register B.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction). The PC is not affected by execution of this instruction.



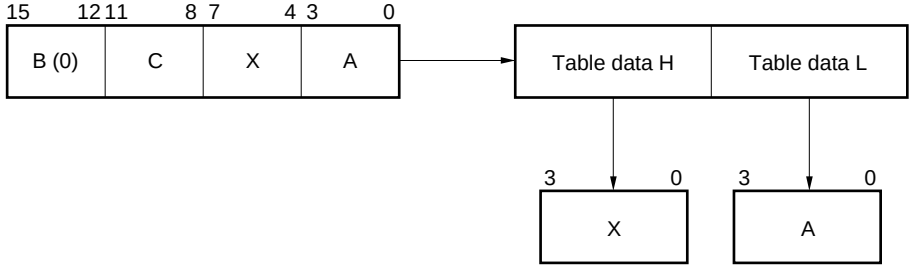
MOVT XA, @BCXA

Function: XA ← ROM (BCXA)

Transfers the low-order 4 bits of the table data (8-bit) in the program memory addressed by the register B and the contents of registers C, X, and A, to the A register, and the high-order 4 bits to the X register.

In the μPD753304, register B is invalid. Set 0000B to register B.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction). The PC is not affected by execution of this instruction.



10.4.3 Bit transfer instructions

- **MOV1 CY, fmem.bit**
- **MOV1 CY, pmem.@L**
- **MOV1 CY, @H+mem.bit**

Function: CY ← (Bit specified by operand)

Transfers the contents of the data memory bit specified by the bit manipulation addressing (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 specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit).

Application example

To output the flag of bit 3 at data memory address 3FH to the bit 2 of port 3

```
FLAG   EQU 3FH.3
SEL    MB0
MOV    H, #FLAG SHR6 ; H ← high-order 4 bits of FLAG
MOV1   CY, @H+FLAG   ; CY ← FLAG
MOV1   PORT3.2, CY   ; P32 ← CY
```

10.4.4 Operation instructions

ADDS A, #n4

Function: A ← A + n4; Skip if carry. n4 = I_{3-0}: 0-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 **10.1.4 Base number adjustment instruction**).

ADDS XA, #n8

Function: XA ← XA + n8; Skip if carry. n8 = I_{7-0}: 00H-FFH

Adds 8-bit immediate data n8 to the contents of register pair XA. 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
```

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 following this instruction, and if a carry occurs as a result of executing this instruction, the ADDS A, #n4 instruction is skipped. If a carry does not occur, the ADDS A, #n4 instruction is executed, and a function that disables the skip function of the ADDS A, #n4 instruction is effected. Therefore, these instructions can be used in combination for base number adjustment (refer to **10.1.4 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 flag. 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.

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'1 ← rp'1 + XA; Skip if borrow.

Subtracts the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the 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 from the contents of the A register, including the carry flag, and sets the result to the A register. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

If an ADDS A, #n4 instruction is placed following this instruction, and if a borrow does not occur as a result of executing this instruction, the ADDS A, #n4 instruction is skipped. If a borrow occurs, the ADDS A, #n4 instruction is executed, and a function that disables the skip function of the ADDS A, #n4 instruction is effected. Therefore, these instructions can be used in combination for base number adjustment (refer to **10.1.4 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 flag, and sets the result to register pair XA. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

SUBC rp'1, XA

Function: rp'1, CY ← rp'1 – XA – CY

Subtracts the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), including the carry flag, and sets the result to the specified register pair rp'1. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

AND A, #n4

Function: A ← A ∧ n4 n4 = I_{3-0}: 0-FH

ANDs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To clear the high-order 2 bits of the accumulator to 0

```
AND A, #0011B
```

AND A, @HL

Function: A ← A ∧ (HL)

ANDs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

AND XA, rp'

Function: XA ← XA ∧ rp'

ANDs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

AND rp'1, XA

Function: rp' 1 ← rp'1 ∧ XA

ANDs the contents of register pair XA with the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair.

OR A, #n4

Function: A ← A ∨ n4 n4 = I_{3-0}: 0-FH

ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To set the low-order 3 bits of the accumulator to 1

```
OR A, #0111B
```

OR A, @HL

Function: A ← A ∨ (HL)

ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

OR XA, rp'

Function: XA ← XA ∨ rp'

ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

OR rp'1, XA

Function: rp' 1 ← rp'1 ∨ XA

ORs the contents of register pair XA with the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to register pair rp'1.

XOR A, #n4

Function: A ← A ⊽ n4 n4 = I_{3-0}: 0-FH

Exclusive-ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To invert the high-order 4 bits of the accumulator

```
XOR A, #1000B
```

XOR A, @HL

Function: A ← A ⊽ (HL)

Exclusive-ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

XOR XA, rp'

Function: XA ← XA ⊽ rp'

Exclusive-ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

XOR rp'1, XA

Function: rp' 1 ← rp'1 ⊽ XA

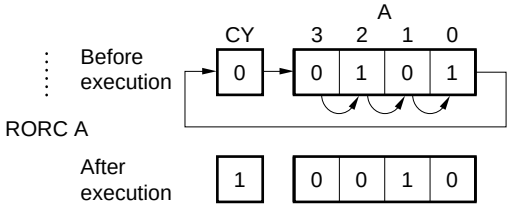
Exclusive-ORs the contents of register pair XA with the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to register pair rp'1.

10.4.5 Accumulator manipulation instructions

RORC A

Function: CY ← A_0, A_{n-1} ← A_n, A_3 ← CY (n = 1-3)

Rotates the contents of the A register (4-bit accumulator) 1 bit to the right with the carry flag.



NOT A

Function: A ← $\overline{A}$

Takes 1's complement of the A register (4-bit accumulator) (inverts the bits of the accumulator).

10.4.6 Increment/decrement instructions

INCS reg

Function: reg ← reg + 1; Skip if reg = 0

Increments the contents of register reg (X, A, H, L, D, E, B, 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 register pair HL. If the contents of the data memory become 0 as a result, the next instruction is skipped.

INCS mem

Function: (mem) ← (mem) + 1; Skip if (mem) = 0, mem = D_{7-0}: 00H-FFH

Increments the contents of the data memory addressed by 8-bit immediate data mem. If the contents of the data memory become 0 as a result, the next instruction is skipped.

DECS reg

Function: reg ← reg – 1; Skip if reg = FH

Decrements the contents of register reg (X, A, H, L, D, E, B, or C). If reg = FH as a result, the next instruction is skipped.

DECS rp'

Function: rp' ← rp' – 1; Skip if rp' = 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.

10.4.7 Comparison instructions

SKE reg, #n4

Function: Skip if reg = n4 n4 = I_{3-0}: 0-FH

Skips the next instruction if the contents of register reg (X, A, H, L, D, E, B, or C) are equal to 4-bit immediate data n4.

SKE @HL, #n4

Function: Skip if (HL) = n4 n4 = I_{3-0}: 0-FH

Skips the next instruction if the contents of the data memory addressed by register pair HL are equal to 4-bit immediate data n4.

SKE A, @HL

Function: Skip if A = (HL)

Skips the next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL.

SKE XA, @HL

Function: Skip if A = (HL) and X = (HL + 1)

Skips the next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL and if the contents of the X register are equal to the contents of the next memory address.

However, if the contents of the L register are an odd number, the address whose least significant bit 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 the contents of 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').

10.4.8 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.

10.4.9 Memory bit manipulation instructions

SET1 mem.bit

Function: (mem.bit) ← 1 mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Sets the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

SET1 fmem.bit

SET1 pmem.@L

SET1 @H+mem.bit

Function: (Bit specified by operand) ← 1

Sets the bit of the data memory specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit).

CLR1 mem.bit

Function: (mem.bit) ← 0 mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Clears the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

CLR1 fmem.bit

CLR1 pmem.@L

CLR1 @H+mem.bit

Function: (Bit specified by operand) ← 0

Clears the bit of the data memory specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit).

SKT mem.bit

Function: Skip if (mem.bit) = 1
mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Skips the next instruction if the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem is 1.

SKT fmem.bit

SKT pmem.@L

SKT @H+mem.bit

Function: Skip if (Bit specified by operand) = 1

Skips the next instruction if the bit of the data memory specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit) is 1.

SKF mem.bit

Function: Skip if (mem.bit) = 0
mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Skips the next instruction if the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem is 0.

SKF fmem.bit

SKF pmem.@L

SKF @H+mem.bit

Function: Skip if (Bit specified by operand) = 0

Skips the next instruction if the bit of the data memory specified by the bit manipulation addressing (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 specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit) is 1, and then clears the bit to “0”.

AND1 CY, fmem.bit

AND1 CY, pmem.@L

AND1 CY, @H+mem.bit

Function: CY $\leftarrow$ CY $\wedge$ (Bit specified by operand)

ANDs the content of the carry flag with the contents of the data memory specified by the bit manipulation addressing (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 specified by the bit manipulation addressing (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 content of the carry flag with the contents of the data memory specified by the bit manipulation addressing (fmem.bit, pmem.@L, or @H+mem.bit), and sets the result to the carry flag.

10.4.10 Branch instructions

BR addr

Function: $PC_{11-0} \leftarrow$ addr
addr = 0000H-0FFFH

Branches to an address addressed by immediate data addr.

This instruction is an assembler pseudo-instruction 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-0FFFH

Branches to an address addressed by immediate data addr1.

This instruction is an assembler pseudo-instruction and is replaced by the assembler at assembly time with the optimum instruction from the BRA !addr1, BR !addr, BRCB !caddr, and BR $addr1 instructions.

II BRA !addr1

Function: $PC_{11-0} \leftarrow$ addr1

BR !addr

Function: $PC_{11-0} \leftarrow$ addr
addr = 0000H-0FFFH

Transfers immediate data addr to the program counter (PC) and branches to the address addressed 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.

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.

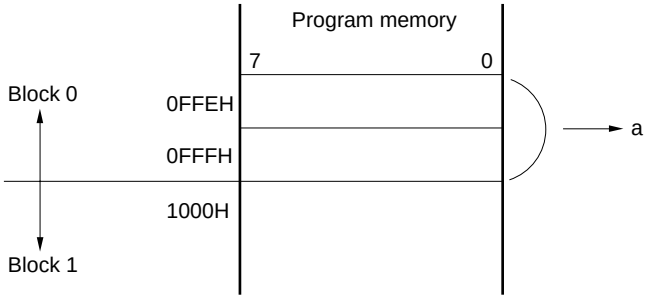
BRCB !caddr

Function: $PC_{11-0} \leftarrow caddr_{11-0}$
caddr = 0000H-0FFFH

Branches to an address specified by the low-order 12 bits of the program counter (PC_{11-0}) replaced with 12-bit immediate data caddr.

Caution

The BRCB !caddr instruction usually branches execution within the block where the instruction exists. If the first byte of this instruction is at address 0FFEH or 0FFFH, however, execution does not branch to block 0, but to block 1.



If the BRCB !caddr instruction is at position a in the figure above, execution branches to block 1 (unmounted), not block 0. Do not use BRCB !caddr instruction at the address 0FFEH.

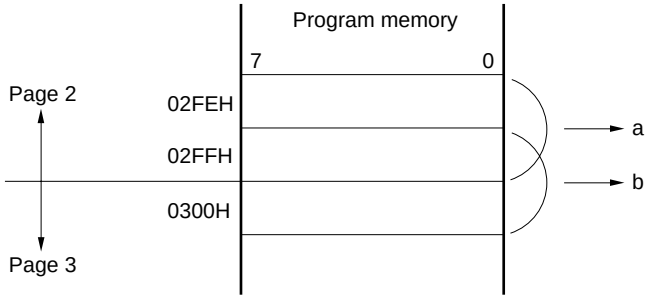
BR PCDE

Function: $PC_{11-0} \leftarrow PC_{11-8} + DE$
$PC_{7-4} \leftarrow D$, $PC_{3-0} \leftarrow E$

Branches to the address specified by the low-order 8 bits of the program counter (PC_{7-0}) replaced with the contents of register pair DE. The high-order bits of the program counter are not affected.

Caution

The BR PCDE instruction usually branches execution within the page where the instruction exists. If the first byte of the op code is at address xxFE or xxFFH, however, execution does not branch in that page, but to the next page.



For example, if the BR PCDE instruction is at position a or b in the above figure, execution branches to the low-order 8-bit address specified by the contents of register pair DE in page 3, not in page 2.

BR PCXA

Function: $PC_{11-0} \leftarrow PC_{11-8} + XA$
$PC_{7-4} \leftarrow X$, $PC_{3-0} \leftarrow A$

Branches to the address specified by the low-order 8 bits of the program counter (PC_{7-0}) replaced with the contents of register pair XA. The high-order bits of the program counter are not affected.

Caution

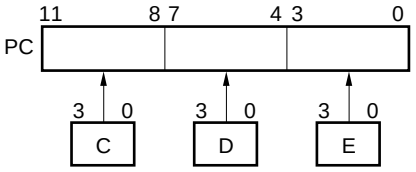
This instruction branches execution to the next page, not to the same page, if the first byte of the op code is at address 0xFEH or 0xFFH, in the same manner as the BR PCDE instruction.

BR BCDE

Function: $PC_{11-0} \leftarrow$ BCDE

Branches to the address specified by the program counter replaced with the contents of registers B, C, D, and E.

However, the PC of the μPD753304 is 12 bits. The PC is replaced with the contents of registers C, D, and E. Be sure to set 0000B to the register B.

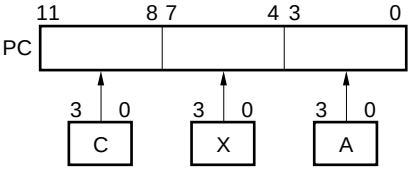


BR BCXA

Function: $PC_{11-0} \leftarrow$ BCXA

Branches to the address specified by the program counter replaced with the contents of registers B, C, X, and A.

However, the PC of the μPD753304 is 12 bits. The PC is replaced with the contents of registers C, X, and A. Be sure to set 0000B to the register B.



TBR addr

Function:

This is an assembler pseudo-instruction for table definition by the GETI instruction. It is used to replace a 3-byte BR !addr instruction with a 1-byte GETI instruction. Code the 12-bit address data as addr. For details, refer to the **RA75X Assembler Package User's Manual-Language (U12385E)**.

10.4.11 Subroutine stack control instructions

(II) CALLA !addr1

Function: **(SP–2) ← x, x, 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: **[Mk I mode]**
(SP–1) ← PC_{7-4}, (SP–2) ← PC_{3-0}
(SP–3) ← MBE, RBE, 0, 0
(SP–4) ← PC_{11-8}, PC_{11-0} ← addr, SP ← SP–4

addr = 0000H-0FFFH

[Mk II mode]
(SP–2) ← x, x, 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

addr = 0000H-0FFFH

Saves the program counter (return address), and the contents of MBE and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address addressed by 12-bit immediate data addr.

I/II CALLF !faddr

Function: **[Mk I mode]**

(SP–1) ← PC_{7-4}, (SP–2) ← PC_{3-0}
(SP–3) ← MBE, RBE, 0, 0
(SP–4) ← PC_{11-8}, SP ← SP–4
PC_{11-0} ← 0+faddr

faddr = 0000H-07FFH

[Mk II mode]
(SP–2) ← x, x, 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-07FFH

Saves the program counter (PC; return address), and the contents of MBE and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address addressed 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 pseudo-instruction for table definition by the GETI instruction. It is used to replace a 3-byte CALL !addr instruction with a 1-byte GETI instruction. Code 12-bit address data as addr. For details, refer to the **RA75X Assembler Package User's Manual-Language (U12385E)**.

RET (I/II)

Function: [Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, RBE, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3), SP \leftarrow SP+4$
[Mk II mode] $PC_{11-8} \leftarrow (SP), 0, 0, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3)$
$x, x, MBE, RBE \leftarrow (SP+4)$
$SP \leftarrow SP+6$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), and then increments the contents of the SP.

Caution

None of the flags of the program status word (PSW), except MBE and RBE, are restored.

RETS (I/II)

Function: [Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, RBE, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3), SP \leftarrow SP+4$
Then skip unconditionally
[Mk II mode] $PC_{11-8} \leftarrow (SP), 0, 0, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3)$
$x, x, MBE, RBE \leftarrow (SP+4), SP \leftarrow SP+6$
Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), increments the contents of the SP, and then skips unconditionally.

Caution

None of the flags of the program status word (PSW), except MBE and RBE, are restored.

RETI (I/II)

Function: [Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, RBE, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3)$
$PSW_L \leftarrow (SP+4), PSW_H \leftarrow (SP+5)$
$SP \leftarrow SP+6$
[Mk II mode] $PC_{11-8} \leftarrow (SP), 0, 0, 0, 0 \leftarrow (SP+1)$
$PC_{3-0} \leftarrow (SP+2), PC_{7-4} \leftarrow (SP+3)$
$PSW_L \leftarrow (SP+4), PSW_H \leftarrow (SP+5)$
$SP \leftarrow SP+6$

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.

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 high-order bits of the register pair (rp_H: X, H, D, or B) are saved to the stack addressed by (SP–1), and the low-order 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 selection register (MBS) and register bank selection 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 (stack) addressed by the stack pointer (SP) to register pair rp (XA, HL, DE, or BC), and then increments the contents of the SP.

The contents of (SP) are restored to the low-order bits of the register pair (rp_L: A, L, E, or C), and the contents of (SP+1) are restored to the high-order bits (rp_H: X, H, D, or B).

POP BS

Function: RBS ← (SP), MBS ← (SP+1), SP ← SP+2

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the register bank selection register (RBS) and memory bank selection register (MBS), and then increments the contents of the SP.

10.4.12 Interrupt control instructions

EI

Function: IME (IPS.3) ← 1

Sets the interrupt master enable flag (bit 3 of the interrupt priority selection register) to “1” to enable interrupts. Acknowledging an interrupt is controlled by an interrupt enable flag corresponding to the interrupt.

EI IExxx

Function: IExxx ← 1 xxx = N_5, $N_{2\text{-}0}$

Sets the interrupt enable flag (IExxx) to “1” to enable acknowledging the interrupt (xxx = BT, T0, W, 1).

DI

Function: IME (IPS.3) ← 0

Resets the interrupt master enable flag (bit 3 of the interrupt priority selection register) to “0” to disable all interrupts, regardless of the contents of the respective interrupt enable flags.

DI IExxx

Function: IExxx ← 0 xxx = N_5, $N_{2\text{-}0}$

Resets the interrupt enable flag (IExxx) to “0” to disable acknowledging the interrupt (xxx = BT, T0, W, 1).

10.4.13 Input/output instructions

IN A, PORTn

Function: A ← PORTn n = N_{3-0}: 3, 8, or 10

Transfers the contents of a port specified by PORTn (n = 3, 8, or 10) to the A register.

Caution

When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 3, 8, or 10 can be specified to n.

The data of the output latch is loaded to the A register in the output mode, and the data of the port pins are loaded to the register in the input mode by specifying the input/output mode.

OUT PORTn, A

Function: PORTn ← A n = N_{3-0}: 3, 8, or 10

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 3, 8, or 10).

Caution

When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15).

Only 3, 8, or 10 can be specified to n.

10.4.14 CPU control instructions

HALT

Function: PCC.2 ← 1

Sets the HALT mode (this instruction sets bit 2 of the processor clock control register).

Caution

Make sure that a NOP instruction follows the HALT instruction.

STOP

Function: PCC.3 ← 1

Sets the STOP mode (this instruction sets bit 3 of the processor clock control register).

Caution

Make sure that a NOP instruction follows the STOP instruction.

NOP

Function: Does nothing but consumes 1 machine cycle.

10.4.15 Special instructions

SEL RBn

Function: RBS ← n n = N_{1-0}: 0-3

Sets 2-bit immediate data n to the register bank selection register (RBS).

SEL MBn

Function: MBS ← n n = N_{3-0}: 0, 1, 15

Transfers 4-bit immediate data n to the memory bank selection register (MBS).

I/II GETI taddr

Function: taddr = T_{5-0}, 0: 20H-7FH

[Mk I mode]

- **When a table defined by the TBR instruction is referenced**
  PC_{11-0} ← $(taddr)_{3-0}$ + (taddr+1)
- **When a table defined by the 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 a table defined by an instruction other than TBR and TCALL is referenced**
  Executes instruction with (taddr) (taddr+1) as op code

[Mk II mode]

- **When a table defined by the TBR instruction is referenced**[Note]
  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$
- **When a table defined by the TCALL instruction is referenced**[Note]
  $(SP-2) \leftarrow x, x, MBE, RBE$
  $(SP-3) \leftarrow PC_{7-4}, (SP-4) \leftarrow PC_{3-0}$
  $(SP-5) \leftarrow 0, 0, 0, 0, (SP-6) \leftarrow PC_{11-8}$
  $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr+1)$
  $SP \leftarrow SP-6$
- **When a table defined by an instruction other than TBR and TCALL is referenced**
  Executes instruction with (taddr) (taddr+1) as op code

Note The addresses specified by the TBR and TCALL instructions are limited to 0000H to 0FFFH.

References the 2-byte data at the program memory address specified by (taddr), (taddr+1) and executes it as an instruction.

The area of the reference table consists of addresses 0020H through 007FH. Data must be written to this area in advance. When the data to be written is 1-byte or 2-byte instructions, code the mnemonics directly.

When a 3-byte call instruction or 3-byte branch instruction is used, data is written by using an assembler pseudo-instruction (TCALL or TBR).

Only an even address can be specified as taddr.

Caution

Only the 2-machine cycle instructions can be placed in the reference table as a 2-byte instructions (except the BRCB and CALLF instructions). Two 1-byte instructions can be used only in the following combinations:

Instruction of 1st byte	Instruction of 2nd byte

MOV A, @HL
MOV @HL, A
XCH A, @HL
	INCS L
	DECS L
	INCS H
	DECS H
	INCS HL

MOV A, @DE XCH A, @DE	INCS E DECS E INCS D DECS D INCS DE
MOV A, @DL XCH A, @DL	INCS L DECS L INCS D DECS D

The contents of the PC are not incremented while the GETI instruction is executed. Therefore, after the referenced instruction has been executed, processing continues from the address following that of the GETI instruction.

If the instruction preceding the GETI instruction has a skip function, the GETI instruction is skipped in the same manner as other 1-byte instructions. If the instruction referenced by the GETI instruction has a skip function, the instruction that follows the GETI instruction is skipped.

If an instruction having a string effect is referenced by the GETI instruction, it is executed as follows:

- If the instruction preceding the GETI instruction has the string effect in the same group as the referenced instruction, the string effect is lost and the referenced instruction is not skipped when GETI instruction is executed.
- If the instruction next to GETI instruction has the string effect in the same group as the referenced instruction, the string effect by the referenced instruction is valid, and the instruction following that instruction is skipped.

Application example

```
MOV     HL, #00H
MOV     XA, #FFH
CALL    SUB1
BR      SUB2
```

Replaced by GETI instruction

```
        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
```

[MEMO]

APPENDIX A μPD75308B, 753108 AND 753304 FUNCTIONAL LIST

(1/3)

Parameter		μPD75308B	μPD753108	μPD753304
Program memory		Mask ROM 0000H-1F7FH (8064 x 8 bits)	Mask ROM 0000H-1FFFH (8192 x 8 bits)	Mask ROM 0000H-0FFFH (4096 x 8 bits)
Data memory		000H-1FFH (512 x 4 bits)		000H-0FFH (256 x 4 bits)
CPU		75X Standard	75XL CPU	
Main system clock oscillator		Crystal/ceramic oscillator		RC oscillator
Subsystem clock oscillator		Crystal oscillator		RC oscillator
Wait time upon release by $\overline{\text{RESET}}$ signal		$2^{17}/f_X$	$2^{17}/f_X$, $2^{15}/f_X$ (Select by mask option)	$56/f_{CC}$
Wait time upon STOP mode release by interrupt generation		$2^{20}/f_X$, $2^{17}/f_X$, $2^{15}/f_X$, $2^{13}/f_X$ (Select by BTM setting)		$512/f_{CC}$, no wait (Select by mask option)
Clock oscillator that can execute STOP instruction		Main system clock oscillator		Main system clock and subsystem clock oscillators
★ Instruction execution time	When main system clock is selected	0.95, 1.91, 15.3 μs (during 4.19-MHz operation)	• 0.95, 1.91, 3.81, 15.3 μs (during 4.19-MHz operation) • 0.67, 1.33, 2.67, 10.7 μs (during 6.0-MHz operation)	1.1, 2.2, 4.4, 17.8 μs (during 3.6-MHz operation)
★	When subsystem clock is selected	122 μs (during 32.768-kHz operation)		85.1 μs (at 47-kHz operation)
Stack	SBS register	None	SBS.3 = 1: Mk I mode selection SBS.3 = 0: Mk II mode selection	
	Stack area	000H-0FFH	000H-1FFH	000H-0FFH
	Subroutine call instruction stack operation	2-byte stack	When MK I mode: 2-byte stack When MK II mode: 3-byte stack	
Instruction	BRA !addr1 CALLA !addr1	Unavailable	When MK I mode: unavailable When MK II mode: available	

	MOVT XA, @BCDE
	MOVT XA, @BCXA
	BR BCDE
	BR BCXA
			Available	

	CALL !addr	3 machine cycles	Mk I mode: 3 machine cycles, Mk II mode: 4 machine cycles	
	CALLF !faddr	2 machine cycles	Mk I mode: 2 machine cycles, Mk II mode: 3 machine cycles	
I/O port	CMOS input	8	8	0
	CMOS input/output	16	20	12
	Bit port output	8	0	0
	N-ch open-drain input/output	8	4	0
	Total	40	32	12

(2/3)

	Parameter		μPD75308B	μPD753108	μPD753304
	LCD controller/driver		Segment selection: 24/28/32 segments (can be changed to CMOS input/output port in 4 time-unit; max. 8)	Segment selection: 16/20/24 segments (can be changed to CMOS input/output port in 4 time-unit; max. 8)	Segment selection: 20/24 segments (can be changed to CMOS input/output port in 4 time-unit; max. 4)
			Display mode selection: static, 1/2 duty (1/2 bias), 1/3 duty (1/2 bias), 1/3 duty (1/3 bias), 1/4 duty (1/3 bias)		
★			On-chip split resistor for LCD driver can be specified by using mask option.		On-chip split resistor for LCD
★			LCD drive voltage cannot be selected.		
	Timer		3 channels • Basic interval timer: 1 channel • 8-bit timer/event counter: 1 channel • Watch timer: 1 channel	5 channels • Basic interval timer/ watchdog timer: 1 channel • 8-bit timer/event counter: 3 channels (can be used as 16-bit timer/event counter) • Watch timer: 1 channel	3 channels • Basic interval timer/ watchdog timer: 1 channel • 8-bit timer counter: 1 channel (subclock source input function provided) • Watch timer: 1 channel
★	Clock output (PCL)		• Φ, 524, 262, 65.5 kHz (Main system clock: during 4.19-MHz operation)	• Φ, 524, 262, 65.5 kHz (Main system clock: during 4.19-MHz operation) • Φ, 750, 375, 93.8 kHz (Main system clock: during 6.0-MHz operation)	• Φ, 3.6 MHz, 450 kHz, 225 kHz (Main system clock: during 3.6-MHz operation)
★	BUZ output (BUZ)		• 2 kHz (Main system clock: during 4.19-MHz operation)	• 2, 4, 32 kHz (Main system clock: during 4.19-MHz operation or subsystem clock: during 32.768-kHz operation) • 2.93, 5.86, 46.9 kHz (Main system clock: during 6.0-MHz operation)	• 2.94, 5.88, 47 kHz (Subsystem clock: during 47-kHz operation) • 1.76, 3.52, 28.13 kHz (Main system clock: during 36-MHz operation)
	Serial interface		3 modes are available • 3-wire serial I/O mode ... MSB/LSB can be selected for transfer first bit • 2-wire serial I/O mode • SBI mode		None
	SOS register	Feedback resistor cut flag (SOS.0)	None	Incorporated	None
		Sub-oscillator circuit current cut flag (SOS.1)	None	Incorporated	None
		Sub-oscillator circuit stop enable flag (SOS.3)	None		Yes
	Register bank selection register (RBS)		None	Yes	

(3/3)

Parameter	μPD75308B	μPD753108	μPD753304
Vectored interrupt	External: 3, internal: 3	External: 3, internal: 5	External: 1, internal: 2
Power supply voltage	V_{DD} = 2.0 to 6.0 V	V_{DD} = 1.8 to 5.5 V	V_{DD} = 2.5 to 5.5 V
★ Operating ambient temperature	T_A = –40 to +85 ˚C		T_A = –10 to +60°C
★ Package	• 80-pin plastic QFP (14 x 20 mm) • 80-pin plastic QFP (14 x 14 mm) • 80-pin plastic TQFP (Fine pitch) (12 x 12 mm)	• 64-pin plastic QFP (14 x 14 mm) • 64-pin plastic QFP (12 x 12 mm)	• Mass product: pellet/wafer • ES product: 42-pin ceramic shrink DIP (600 mil), (for evaluation)

[MEMO]

APPENDIX B DEVELOPMENT TOOLS

The following development tools are provided for system development using the μPD753304.

In 75XL Series, the relocatable assembler which is common to the series is used in combination with the device file of each product.

Language processor

RA75X relocatable assembler	Host machine	OS	Supply media	Part number (product name)
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD	μS5A13RA75X
			5-inch 2HD	μS5A10RA75X
	IBM PC/AT™ and compatibles	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13RA75X
			5-inch 2HC	μS7B10RA75X

Device file	Host machine	OS	Supply media	Part number (product name)
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD	μS5A13DF753304
			5-inch 2HD	μS5A10DF753304
	IBM PC/AT and compatibles	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13DF753304
			5-inch 2HC	μS7B10DF753304

Note Although Ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operations of the assembler and device file are guaranteed only on the above host machines and OSs.

Debugging tool

The in-circuit emulators (IE-75000-R and IE-75001-R) are available as the program debugging tool for the μPD753304.

The system configurations are described as follows.

<table>
<tr><td rowspan="5">Hardware</td><td>IE-75000-R^{Note 1}</td><td colspan="4">In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X Series and 75XL Series. When developing the μPD753304, the emulation board IE-75300-R-EM and emulation probe EP-753304DU-R that are sold separately must be used with the IE-75000-R.
It can debug the system efficiently by connecting the host machine.
It contains the emulation board IE-75000-R-EM which is connected.</td></tr>
<tr><td>IE-75001-R</td><td colspan="4">In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X Series and 75XL Series. When developing the μPD753304, the emulation board IE-75300-R-EM and emulation probe EP-753304DU-R that are sold separately must be used with the IE-75001-R.
It can debug the system efficiently by connecting the host machine.</td></tr>
<tr><td>IE-75300-R-EM</td><td colspan="4">Emulation board for evaluating the application systems that use the μPD753304.
It must be used with the IE-75000-R or IE-75001-R.</td></tr>
<tr><td>EP-753304DU-R</td><td colspan="4">Emulation probe for ES products.
It must be connected to IE-75000-R (or IE-75001-R) and IE-75300-R-EM.</td></tr>
<tr><td rowspan="5">Software</td><td rowspan="5">IE control program</td><td colspan="4">Connects the IE-75000-R or IE-75001-R to a host machine via RS-232-C and Centronics interface and controls the IE-75000-R or IE-75001-R on a host machine.</td></tr>
<tr><td rowspan="2">Host machine</td><td colspan="2" rowspan="2">OS</td><td rowspan="2">Supply media</td><td rowspan="2">Part number (product name)</td></tr>
<tr></tr>
<tr><td>PC-9800 series</td><td colspan="2">MS-DOS
(Ver. 3.30 to Ver. 6.2^{Note 2})</td><td>3.5-inch 2HD</td><td>μS5A13IE75X</td></tr>
<tr><td>PC-9800 series</td><td colspan="2">MS-DOS
(Ver. 3.30 to Ver. 6.2^{Note 2})</td><td>5-inch 2HD</td><td>μS5A10IE75X</td></tr>
<tr><td>IBM PC/AT and compatibles</td><td colspan="2">Refer to “OS for IBM PC”</td><td>3.5-inch 2HC</td><td>μS7B13IE75X</td></tr>
<tr><td>IBM PC/AT and compatibles</td><td colspan="2">Refer to “OS for IBM PC”</td><td>5-inch 2HC</td><td>μS7B10IE75X</td></tr>
</table>

Notes 1. Maintenance product

2. Although Ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operation of the IE control program is guaranteed only on the above host machines and OSs.

OS for IBM PC

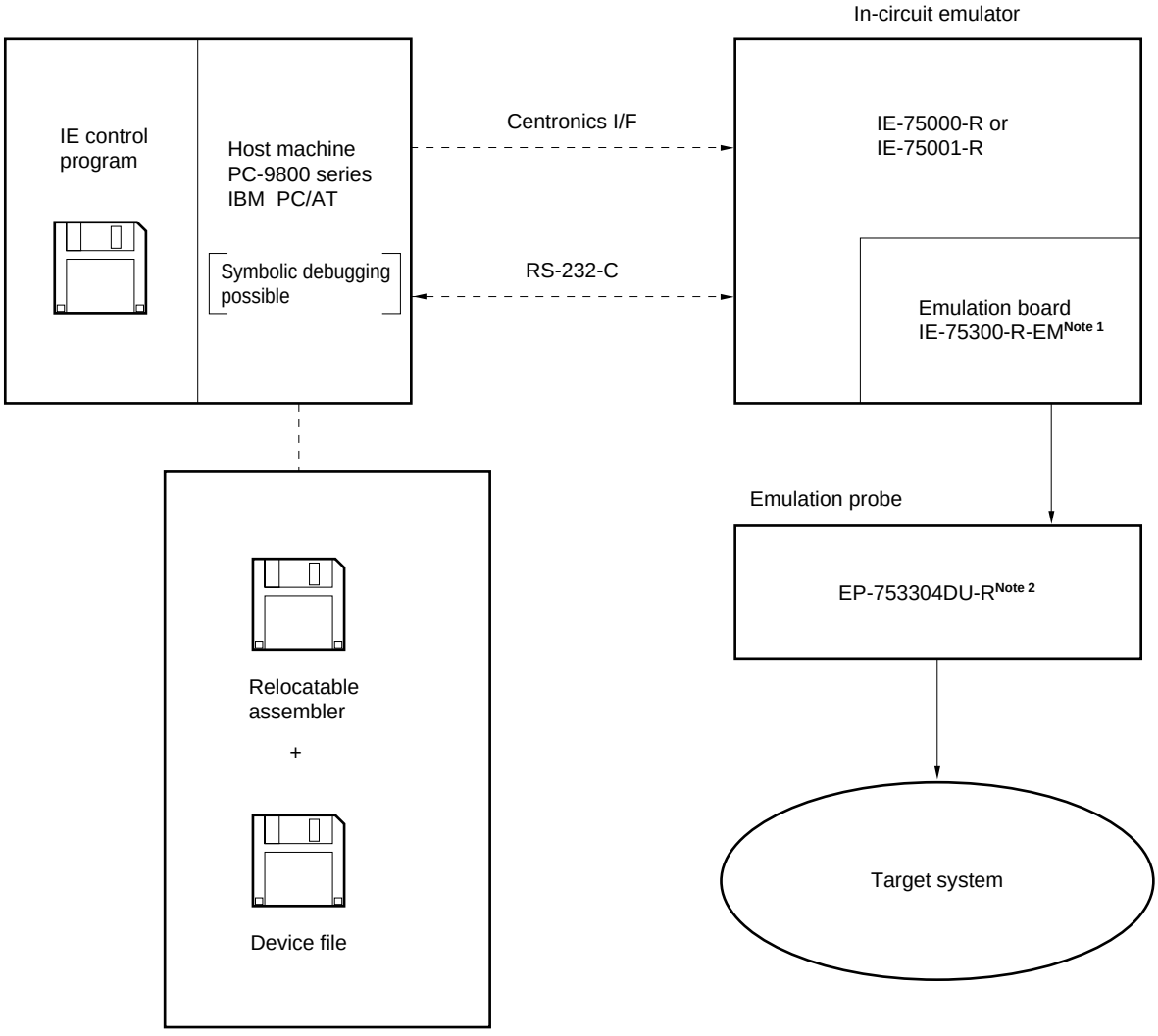
The following IBM PC OSs are supported.

OS	Version
PC DOS	Ver. 5.02 to Ver. 6.3 J6.1/V[Note] to J6.3/V[Note]
MS-DOS	Ver. 5.0 to Ver. 6.22 5.0/V[Note] to 6.2/V[Note]
IBM DOS™	J5.02/V[Note]

Note Only English version is supported.

Caution Although Ver. 5.0 and later versions have the task swap function, the function cannot be used with this software.

Development Tool Configuration



Notes 1. The in-circuit emulator does not include IE-75300-R-EM (sold separately).

2. Emulation probe for the ES product.

APPENDIX C ORDERING MASK ROMS

After completion of program development, place an order for a mask ROM using the following procedure:

<1> Reservation for ordering mask ROM

Inform NEC of your schedule for ordering mask ROM (delivery may be delayed if NEC is not informed in advance).

<2> Preparation of ordering media

The mask ROM ordering media can be selected from the following three types.

- UV-EPROM[Note]
- 3.5-inch IBM-format floppy disk (outside Japan only)
- 5-inch IBM-format floppy disk (outside Japan only)

Note When ordering by UV-EPROM, make three copies of UV-EPROM.
Mask option data should be written on the mask option information sheet.

<3> Preparation of necessary documents

Fill out the following documents when ordering the mask ROM:

A. Mask ROM Ordering Sheet
B. Mask ROM Ordering Checksheet
C. Mask Option Information Sheet (necessary for product with mask option)

<4> Ordering

Submit the media prepared in <2> and documents prepared in <3> to NEC by the planned ordering date.

[MEMO]

APPENDIX D INSTRUCTION INDEX

D.1 Instruction Index (by function)

D.2 Instruction Index (alphabetical order)

APPENDIX E HARDWARE INDEX

[MEMO]

★

APPENDIX F REVISION HISTORY

The revision history and the applicable chapters are shown in the following table.

Edition	Revised contents	Applicable chapters
2nd edition	Status of μPD753304 is changed from “under development” to “developed”.	Throughout
	Method of supplying mass-produced devices changed to pellet/wafer.	
	Main system clock is changed from 4.0 MHz to 3.6 MHz (typ.) and subsystem clock is changed from 32.768 MHz to 47 kHz (typ.).	
	Status at reset by Port 3 mask option is added.	CHAPTER 2 PIN FUNCTION
	Mask options of LCD display mode is added.	CHAPTER 5 PERIPHERAL
	10 kΩ (typ.) split resistor for LCD drive is deleted.	HARDWARE FUNCTION
	Interrupt request signal to STOP mode release signal of subsystem clock is added.	CHAPTER 7 STANDBY FUNCTION

[MEMO]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

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

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑