NEC

User's Manual

μPD753108

4-bit Single-chip Microcontrollers

μPD753104
μPD753106
μPD753108
μPD75P3116

Document No. U10890EJ3V0UM00 (3rd edition)
Date Published March 1998 N CP(K)

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

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

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

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: 65-253-8311
Fax: 65-250-3583

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

J98. 2

Major Revisions in this Edition

Page	Description
Throughout	Data bus pins (D0 to D7) are added.
p. 41	**Table 2-3 List of Recommended Connections for Unused Pins** is changed.
p. 71	Caution in **Table 4-1 Differences between Mk I Mode and Mk II Mode** is changed.
p. 122	Description on feedback resistor mask option in **5.2.2 (6) Subsystem clock oscillator control register (SOS)** is deleted.
p. 124	**Table 5-5 Maximum Time Required to Switch System to/from CPU Clocks** is changed.
p. 125	**Figure 5-19 Switching between System Clock and CPU Clock** is changed.
p. 238	Cautions are added to **5.6.7 (a) Bus release signal (REL)** and **5.6.7 (b) Command signal (CMD)**.
p. 340	**7.4 Mask Option Selection** is added.
p. 353	**9.2 Writing Program Memory** is changed.
p. 354	**9.3 Reading Program Memory** is changed.
p. 358	**10.3 Mask Option of Feedback Resistor for Subsystem Clock** is deleted.
p. 449	Ordering media in **APPENDIX C ORDERING MASK ROMS** are changed.

The mark ★ shows major revised points.

INTRODUCTION

Readers: This manual is intended for user engineers who understand the functions of the μPD753104, 753106, 753108, and 75P3116 4-bit single-chip microcontrollers, and wish to design application systems using any of these microcontrollers.

Purpose: This manual describes the hardware functions of the μPD753104, 753106, 753108, and 75P3116 in the organization described below.

Organization: This manual contains the following information:

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

How to Read This Manual:

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

- If you have experience of using the μPD75308B,
  → Read **APPENDIX A μPD75308B, 753108, AND 75P3116 FUNCTIONAL LIST** to check differences between the μPD75308B and the microcontrollers described in this manual.

- If you use this manual as the manual for the μPD753104, 753106, and 75P3116,
  → Unless otherwise specified, the μPD753108 is regarded as the representative model, and description throughout this manual is focused on this model. Refer to section **1.3 Differences among μPD753108 Subseries Products** to check the differences among the respective models.

- 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 μPD753104, 753106, 753108, and 75P3116,
  → Read this manual in the order of Contents.

Legend

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

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

Documents related to devices

Document Name	Document Number	
	English	Japanese
μPD753108 User's Manual	U10890E (This manual)	U10890J
μPD753104, 753106, 753108 Data Sheet	U10086E	U10086J
μPD75P3116 Data Sheet	U11369E	U11369J
μPD753108 Instruction Table	—	IEM-5600
75XL Series Selection Guide	U10453E	U10453J

Documents related to development tools

Document Name			Document Number	
			English	Japanese
Hardware	IE-75000-R/IE-75001-R User's Manual		EEU-1416	EEU-846
	IE-75300-R-EM User's Manual		U11354E	U11354J
	EP-753108GC/GK-R User's Manual		EEU-1495	EEU-968
	PG-1500 User's Manual		U11940E	U11940J
Software	RA75X Assembler Package User's Manual	Operation	U12622E	U12622J
		Language	U12385E	U12385J
	PG-1500 Controller User's Manual	PC-9800 series (MS-DOS™-based)	EEU-1291	EEU-704
		IBM PC series (PC DOS™-based)	U10540E	EEU-5008

Other documents

Document Name	Document Number	
	English	Japanese
IC Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535E	C10535J
Quality Grades on NEC Semiconductor Devices	C11531E	C11531J
NEC Semiconductor Device Reliability/Quality Control System	C10983E	C10983J
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E	C11892J
Guide to Quality Assurance for Semiconductor Devices	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 (1/4)

LIST OF FIGURES (2/4)

LIST OF FIGURES (3/4)

LIST OF FIGURES (4/4)

LIST OF TABLES (1/2)

LIST OF TABLES (2/2)

CHAPTER 1 GENERAL

The μPD753104, 753106,753108, and 75P3116 are 4-bit single-chip microcontrollers in the NEC's 75XL Series, a successor to the 75X Series that boasts a wealth of variations. These four devices are called the μPD753108 Subseries as a general term.

The μPD753108 is based on the existing μPD75308B, but has a higher ROM capacity and more sophisticated CPU functions, and can operate at high speeds at a voltage of as low as 1.8 V. In addition, the μPD753108 is also provided with an LCD controller/driver.

This model is available in a small plastic QFP (12 × 12 mm) and is ideal for small application set that uses an LCD panel.

The features of the μPD753108 are as follows:

- Low-voltage operation: V_{DD} = 1.8 to 5.5 V
- Variable instruction execution time useful for high-speed operation and power saving
  0.95 μs, 1.91 μs, 3.81 μs, 15.3 μs (at 4.19 MHz)
  0.67 μs, 1.33 μs, 2.67 μs, 10.7 μs (at 6.0 MHz)
  122 μs (at 32.768 kHz)
- Five timer channels
- Programmable LCD controller/driver
- Small package (64-pin plastic QFP (12 × 12 mm))

The μPD75P3116 is provided with a one-time PROM that can be electrically written and is pin-compatible with the μPD753104, 753106, and 753108. This one-time PROM model is convenient for experimental development or small-scale production of an application system.

Application Fields

- Remote controllers
- Camera-contained VCRs (camcoders)
- Cameras
- Gas meters, etc.

Remark Unless otherwise specified, the μPD753108 is regarded as the representative model, and description throughout this manual is focused on this model.

1.1 Functional Outline

Parameter		Function	
Instruction execution time		• 0.95, 1.91, 3.81, 15.3 μs (@ 4.19 MHz with main system clock) • 0.67, 1.33, 2.67, 10.7 μs (@ 6.0 MHz with main system clock) • 122 μs (@ 32.768 kHz with subsystem clock)	
On-chip memory	ROM	4096 × 8 bits (μPD753104)	
		6144 × 8 bits (μPD753106)	
		8192 × 8 bits (μPD753108)	
		16384 × 8 bits (μPD75P3116)	
	RAM	512 × 4 bits	
General-purpose register		• 4-bit operation: 8 × 4 banks • 8-bit operation: 4 × 4 banks	
Input/output port	CMOS input	8 pins	Of these, seven have software-specifiable on-chip pull-up resistors
	CMOS input/output	20 pins	Twelve have software-specifiable on-chip pull-up resistors, and eight can also be used as segment outputs
	N-ch open-drain input/output	4 pins	Each withstands up to 13 V and has a mask-option pull-up resistor[Note 1]
	Total	32 pins	
LCD controller/driver		• Segment selection: 16/20/24 segments (can be changed to CMOS input/output port in 4-pin units; max. 8) • 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 drive can be specified by mask option[Note 2]	
Timer		5 channels • 8-bit timer/event counter: 3 channels (Can also be used as a 16-bit timer/event counter, a carrier generator, or a gated timer) • Basic interval timer/watchdog timer: 1 channel • Watch timer: 1 channel	
Serial interface		• 3-wire serial I/O mode ... MSB or LSB can be selected for transferring first bit • 2-wire serial I/O mode • SBI mode	
Bit sequential buffer (BSB)		16 bits	
Clock output (PCL)		• Φ, 524, 262, 65.5 kHz (@ 4.19 MHz with main system clock) • Φ, 750, 375, 93.8 kHz (@ 6.0 MHz with main system clock)	
Buzzer output (BUZ)		• 2, 4, 32 kHz (@ 4.19 MHz with main system clock or @ 32.768 kHz with subsystem clock) • 2.93, 5.86, 46.9 kHz (@ 6.0 MHz with main system clock)	
Vectored interrupt		External: 3, Internal: 5	
Test input		External: 1, Internal: 1	
System clock oscillator		• Ceramic or crystal oscillator for main system clock oscillation • Crystal oscillator for subsystem clock oscillation	
Standby function		STOP/HALT mode	
Power supply voltage		V_{DD} = 1.8 to 5.5 V	
Package		• 64-pin plastic QFP (14 × 14 mm) • 64-pin plastic QFP (12 × 12 mm)	

Notes 1. The μPD75P3116 is not provided with the function to connect pull-up resistors by mask option.

2. The μPD75P3116 is not provided with a split resistor by mask option.

1.2 Ordering Information

Part Number	Package	Internal ROM
μPD753104GC-×××-AB8	64-pin plastic QFP (14 × 14 mm, 0.8-mm pitch)	Mask ROM
μPD753104GK-×××-8A8	64-pin plastic QFP (12 × 12 mm, 0.65-mm pitch)	Mask ROM
μPD753106GC-×××-AB8	64-pin plastic QFP (14 × 14 mm, 0.8-mm pitch)	Mask ROM
μPD753106GK-×××-8A8	64-pin plastic QFP (12 × 12 mm, 0.65-mm pitch)	Mask ROM
μPD753108GC-×××-AB8	64-pin plastic QFP (14 × 14 mm, 0.8-mm pitch)	Mask ROM
μPD753108GK-×××-8A8	64-pin plastic QFP (12 × 12 mm, 0.65-mm pitch)	Mask ROM
μPD75P3116GC-AB8	64-pin plastic QFP (14 × 14 mm, 0.8-mm pitch)	One-time PROM
μPD75P3116GK-8A8	64-pin plastic QFP (12 × 12 mm, 0.65-mm pitch)	One-time PROM

Remark ××× indicates ROM code suffix.

1.3 Differences among μPD753108 Subseries Products

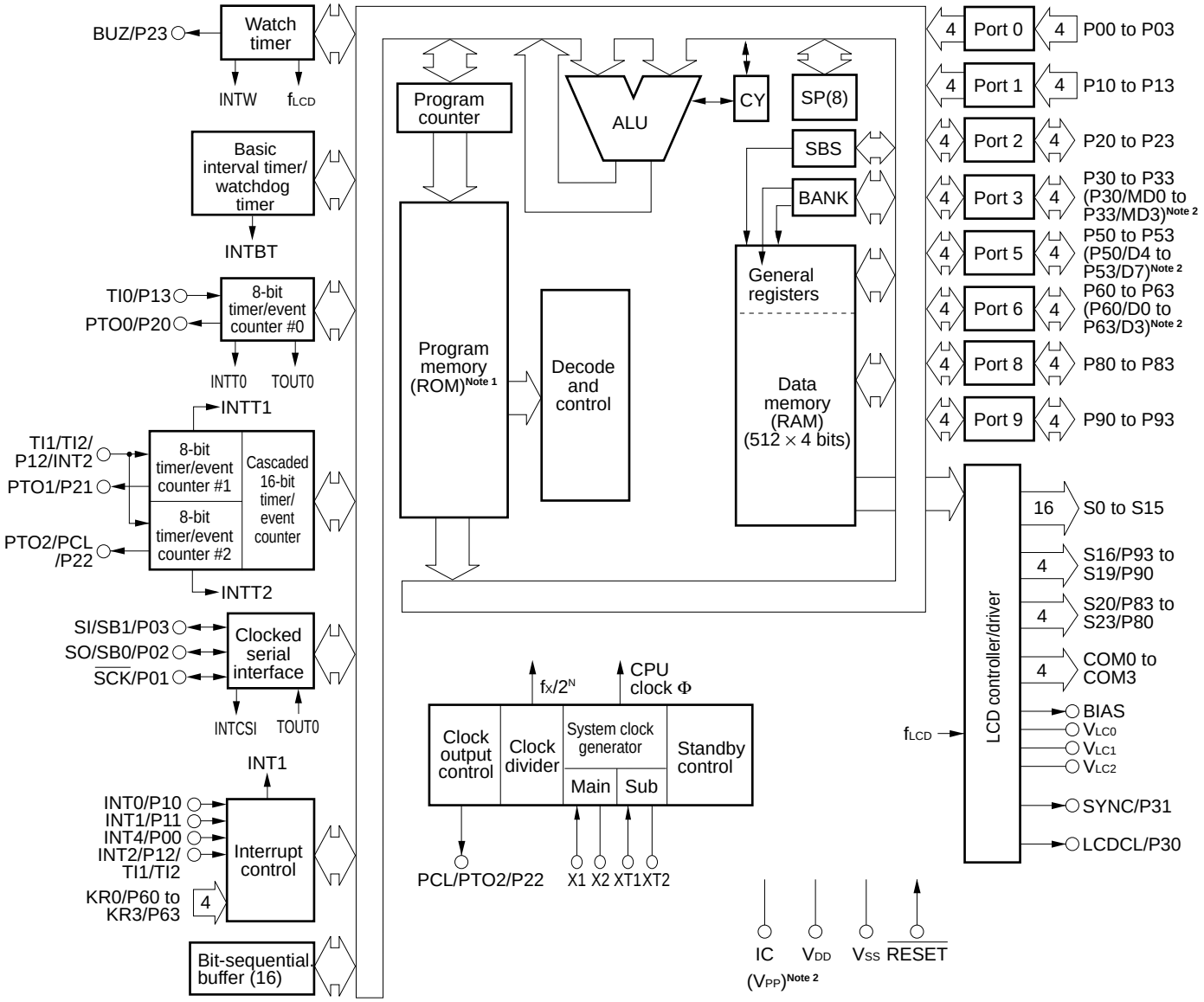
★

Item		μPD753104	μPD753106	μPD753108	μPD75P3116
Program counter		12 bits	13 bits		14 bits
Program memory (byte)		Mask ROM 4096	Mask ROM 6144	Mask ROM 8192	One-time PROM 16384
Data memory (× 4 bits)		512			
Mask option	Pull-up resistor of port 5	Available (specifiable to incorporate or not)			None (Cannot incorporate)
	Wait time after $\overline{\text{RESET}}$	Available (selectable from $2^{17}/f_X$ and $2^{15}/f_X$)**Note**			None (Fixed to $2^{15}/f_X$)**Note**
	Split resistor for LCD driving	Available (specifiable to incorporate or not)			None (Cannot incorporate)
Pin connection	Pins 5 to 8	P30 to P33			P30/MD0 to P33/MD3
	Pins 10 to 13	P50 to P53			P50/D4 to P53/D7
	Pins 14 to 17	P60/KR0 to P63/KR3			P60/KR0/D0 to P63/KR3/D3
	Pin 21	IC			V_{PP}
Other		Noise immunity and noise radiation differ because circuit scale and mask layout differ.			

Note The wait time after $\overline{\text{RESET}}$ becomes 21.8 ms at 6.0 MHz and 31.3 ms at 4.19 MHz if $2^{17}/f_X$ is selected; it becomes 5.46 ms at 6.0 MHz and 7.81 ms at 4.19 MHz if $2^{15}/f_X$ is selected.

Caution There are differences in noise immunity and noise radiation between the PROM and mask ROM versions. When pre-producing an application set with the PROM version and then mass-producing it with the mask ROM version, be sure to conduct sufficient evaluations for the commercial samples (not engineering samples) of the mask ROM version.

1.4 Block Diagram

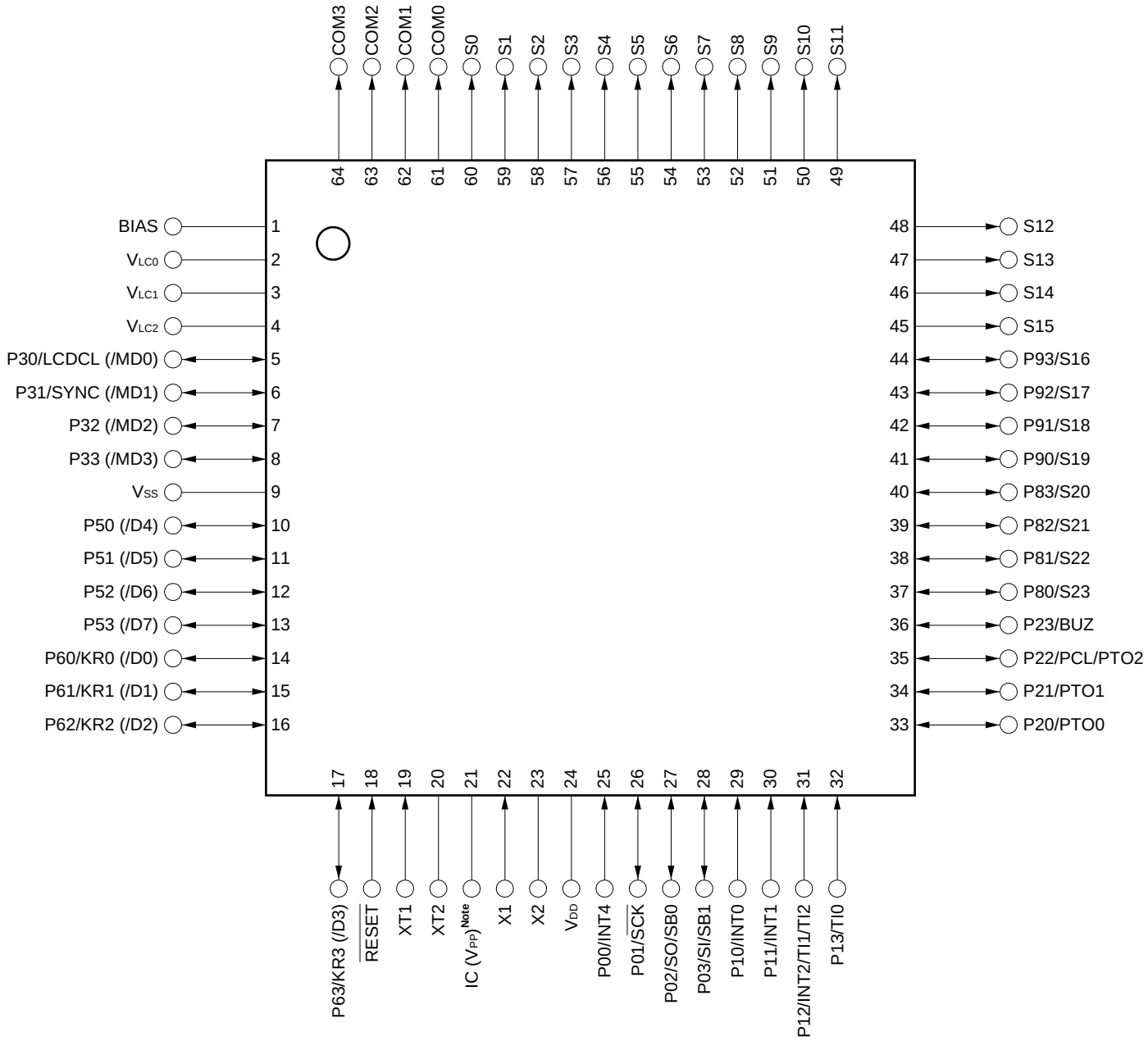


Notes 1. Capacity of the ROM depends on the product.

2. The pin names or functions in parentheses applies with the μPD75P3116.

1.5 Pin Configuration (Top View)

- **64-pin plastic QFP (14 × 14 mm)**
  μPD753104GC-×××-AB8, μPD753106GC-×××-AB8
  μPD753108GC-×××-AB8, μPD75P3116GC-AB8
- **64-pin plastic QFP (12 × 12 mm)**
  μPD753104GK-×××-8A8, μPD753106GK-×××-8A8
  μPD753108GK-×××-8A8, μPD75P3116GK-8A8



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

Remark The pin names or functions in parentheses applies with the μPD75P3116.

Pin Identification

BIAS	: LCD Power Supply Bias Control	P90 to P93	: Port 9
BUZ	: Buzzer Clock	PCL	: Programmable Clock
COM0 to COM3	: Common Output 0 to 3	PTO0 to PTO2	: Programmable Timer Output 0 to 2
D0 to D7	: Data Bus 0 to 7	$\overline{\text{RESET}}$	: Reset
IC	: Internally Connected	S0 to S23	: Segment Output 0 to 23
INT0, INT1, INT4	: External Vectored Interrupt 0, 1, 4	SB0, SB1	: Serial Data Bus 0, 1
INT2	: External Test Input 2	$\overline{\text{SCK}}$	: Serial Clock
KR0 to KR3	: Key Return 0 to 3	SI	: Serial Input
LCDCL	: LCD Clock	SO	: Serial Output
MD0 to MD3	: Mode Selection 0 to 3	SYNC	: LCD Synchronization
P00 to P03	: Port 0	TI0 to TI2	: Timer Input 0 to 2
P10 to P13	: Port 1	V_{DD}	: Positive Power Supply
P20 to P23	: Port 2	V_{LC0} to V_{LC2}	: LCD Power Supply 0 to 2
P30 to P33	: Port 3	V_{PP}	: Programming Power Supply
P50 to P53	: Port 5	V_{SS}	: Ground
P60 to P63	: Port 6	X1, X2	: Main System Clock Oscillation 1, 2
P80 to P83	: Port 8	XT1, XT2	: Subsystem Clock Oscillation 1, 2

CHAPTER 2 PIN FUNCTION

2.1 Pin Functions of μPD753108

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

Pin Name	Input/Output	Alternate Function	Function	8-bit I/O	After Reset	I/O Circuit Type[Note 1]
P00	Input	INT4	4-bit input port (PORT0). For P01 to P03, on-chip pull-up resistors can be specified by software in 3-bit units.	No	Input	[B]
P01	Input/Output	$\overline{SCK}$				[F]-A
P02	Input/Output	SO/SB0				[F]-B
P03	Input/Output	SI/SB1				[M]-C
P10	Input	INT0	4-bit input port (PORT1). On-chip pull-up resistors can be specified by software in 4-bit units. Noise eliminator can be selected in P10/INT0.	No	Input	[B]-C
P11		INT1				
P12		TI1/TI2/INT2				
P13		TI0				
P20	Input/Output	PTO0	4-bit input/output port (PORT2). On-chip pull-up resistors can be specified by software in 4-bit units.	No	Input	E-B
P21		PTO1				
P22		PCL/PTO2				
P23		BUZ				
P30	Input/Output	LCDCL (/MD0)[Note 2]	Programmable 4-bit input/output port (PORT3). This port can be specified input/output bit-wise. On-chip pull-up resistor can be specified by software in 4-bit units.	No	Input	E-B
P31		SYNC (/MD1)[Note 2]				
P32		(MD2)[Note 2]				
P33		(MD3)[Note 2]				
★ P50[Note 3]	Input/Output	(D4)[Note 2]	N-ch open-drain 4-bit input/output port (PORT5). Each pin withstands up to 13 V in open-drain mode. A pull-up resistor can be contained bit-wise (mask option).[Note 4]	No	High level (when pull-up resistors are provided) or high-impedance	M-D (M-E)[Note 2]
P51[Note 3]		(D5)[Note 2]				
P52[Note 3]		(D6)[Note 2]				
P53[Note 3]		(D7)[Note 2]				

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

2. The pin names or functions in parentheses are available with the μPD75P3116.

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

4. The μPD75P3116 is not provided with the function to connect pull-up resistors by mask option.

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

Pin Name	Input/Output	Alternate Function	Function	8-bit I/O	After Reset	I/O Circuit Type[Note 1]
★ P60	Input/Output	KR0 (/D0)[Note 2]	Programmable 4-bit input/output port (PORT6). This port can be specified for input/output bit-wise. On-chip pull-up resistors can be specified by software in 4-bit units.	No	Input	[F]-A
P61		KR1 (/D1)[Note 2]				
P62		KR2 (/D2)[Note 2]				
P63		KR3 (/D3)[Note 2]				
P80	Input/Output	S23	4-bit input/output port (PORT8). On-chip pull-up resistors can be specified by software in 4-bit units.[Note 3]	Yes	Input	H
P81		S22				
P82		S21				
P83		S20				
P90	Input/Output	S19	4-bit input/output port (PORT9). On-chip pull-up resistors can be specified by software in 4-bit units.[Note 3]		Input	H
P91		S18				
P92		S17				
P93		S16				

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

2. The pin names or functions in parentheses are available with the μPD75P3116.

3. If these pins are used as segment outputs, do not enable the on-chip pull-up resistors for these pins by software.

Table 2-2. Pin Function of Pins Other Than Port Pins (1/2)

Pin Name	Input/Output	Alternate Function	Function		After Reset	I/O Circuit Type[Note 1]
TI0	Input	P13	Inputs external event pulses to the timer/ event counter.		Input	[B]-C
TI1		P12/INT2/TI2				
TI2		P12/INT2/TI1				
PTO0	Output	P20	Timer/event counter output		Input	E-B
PTO1		P21				
PTO2		P22/PCL				
PCL		P22/PTO2	Clock output			
BUZ		P23	Optional frequency output (for buzzer output or system clock trimming)			
$\overline{SCK}$	Input/Output	P01	Serial clock input/output		Input	[F]-A
SO/SB0		P02	Serial data output Serial data bus input/output			[F]-B
SI/SB1		P03	Serial data input Serial data bus input/output			[M]-C
INT4	Input	P00	Edge detection vectored interrupt input (both rising edge and falling edge detection)		Input	[B]
INT0	Input	P10	Edge detection vectored interrupt input (detection edge can be selected). Noise eliminator can be selected in INT0/P10.	Can select with noise eliminator or asynchronous	Input	[B]-C
INT1		P11		Asynchronous		
INT2	Input	P12/TI1/TI2	Rising edge detection testable input	Asynchronous	Input	[B]-C
KR0 to KR3	Input	P60 to P63	Falling edge detection testable input		Input	[F]-A
S0 to S15	Output	—	Segment signal output		**Note 2**	G-A
S16 to S19	Output	P93 to P90	Segment signal output		Input	H
S20 to S23	Output	P83 to P80	Segment signal output		Input	H
COM0 to COM3	Output	—	Common signal output		**Note 2**	G-B
V_{LC0} to V_{LC2}	—	—	LCD drive power On-chip split resistor can be connected by mask option.[Note 3]		—	—

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

2. Each output selects the following V_{LCX} as input source.
S0 to S15: V_{LC1}, COM0 to COM2: V_{LC2}, COM3: V_{LC0}

3. The μPD75P3116 does not contain the mask options and a split resistor is not on-chip.

Table 2-2. Pin Function of Pins Other Than Port Pins (2/2)

Pin Name	Input/Output	Alternate Function	Function	After Reset	I/O Circuit Type[Note 1]
BIAS	Output	—	Output for external split resistor disconnect	**Note 2**	—
LCDCL[Note 3]	Output	P30	Clock output for externally expanded driver	Input	E-B
SYNC[Note 3]	Output	P31	Clock output for externally expanded driver sync	Input	E-B
X1	Input	—	Crystal/ceramic connection pin for the main system clock oscillator. When inputting the external clock, input the external clock to pin X1, and the reverse phase of the external clock to pin X2.	—	—
X2	—				
XT1	Input	—	Crystal connection pin for the subsystem clock oscillator. When the external clock is used, input the external clock to pin XT1. In this case, input the reverse phase of clock to pin XT2. Pin XT1 can be used as a 1-bit input (test) pin.	—	—
★ XT2	—				
$\overline{\text{RESET}}$	Input	—	System reset input (low-level active)	—	[B]
MD0 to MD3	Input	P30 to P33	Provided only to the μPD75P3116. Select modes for writing/verifying program memory (PROM).	Input	E-B
★ D0 to D3	Input/Output	P60/KR0 to P63/KR3	Provided only to the μPD75P3116. Data bus pins during program memory (PROM) write/verify.	Input	[F]-A
★ D4 to D7		P50 to P53		High-impedance	M-E
IC	—	—	Internally connected. Connect this pin directly to V_{DD}.	—	—
V_{PP}	—	—	Provided only to the μPD75P3116. Supplies voltage necessary for writing/verifying program memory (PROM). Directly connect this pin to V_{DD} for normal operation. Apply +12.5 V to this pin to write/verify PROM.	—	—
V_{DD}	—	—	Positive power supply	—	—
V_{SS}	—	—	Ground potential	—	—

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

2. When a split resistor is contained ..... Low level
When no split resistor is contained ..... High-impedance

3. These pins are provided for future system expansion. At present, these pins are used only as pins P30 and P31.

2.2 Pin Functions

2.2.1 P00 to P03 (PORT0) ··· input shared with INT4, $\overline{\text{SCK}}$, SO/SB0, and SI/SB1
P10 to P13 (PORT1) ··· input shared with INT0, INT1, INT2/TI1/TI2, and TI0

These pins are 4-bit input ports. These pins have the following functions, in addition to the input port function.

- Port 0 : Vectored interrupt input (INT4)
  Serial interface I/Os ($\overline{\text{SCK}}$, SO/SB0, SI/SB1)
- Port 1 : Vectored interrupt inputs (INT0, INT1)
  Edge detection test input (INT2)
  External event pulse input to timer/event counter (TI0 to TI2)

The alternate function pin of port 0 is provided with the output function depending on the operation mode when port 0 uses serial interface function.

Each pin of port 0 and port 1 is Schmitt trigger input pins to prevent malfunctioning due to noise. In addition, the P10 pin can select a noise eliminator (for details, refer to **6.3 (3) Hardware of INT0, INT1, and INT4**).

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

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

2.2.2 P20 to P23 (PORT2) ··· I/O shared with PTO0 to PTO2, PCL, and BUZ

P30 to P33 (PORT3) ··· I/O shared with LCDCL, SYNC, and MD0 to MD3[Note]

★ **P50 to P53 (PORT5) ··· N-ch open-drain, medium voltage (13 V), I/O shared with D4 to D7[Note]**

★ **P60 to P63 (PORT6) ··· I/O shared with KR0 to KR3 and D0 to D3[Note]**

P80 to P83 (PORT8) and P90 to P93 (PORT9) ··· I/O shared with S23 to S20 and S19 to S16

These pins are 4-bit I/O ports with output latch.

In addition to the I/O port function, ports share the following functions.

- Port 2 : Timer/event counter outputs (PTO0 to PTO2)
  Clock output (PCL)
  Any frequency output (BUZ)
- Port 3 : Clock for driving external expansion LCD driver (LCDCL)
  Clock for synchronizing external expansion LCD driver (SYNC)
  ★ Mode selection (MD0 to MD3)[Note] of program memory (PROM) write/verify
- ★ Port 5 : Data bus (D4 to D7)[Note] of program memory (PROM) write/verify
- Port 6 : Key interrupt input (KR0 to KR3)
  ★ Data bus (D0 to D3)[Note] of program memory (PROM) write/verify
- Ports 8 and 9 : Segment signal outputs (S23 to S20 and S19 to S16)

Note Alternate function pin only in the μPD75P3116.

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

Input/output mode selection of each port is set by the port mode register. Ports 2, 5, 8, and 9 can be set in input or output mode in 4-bit units, and ports 3 and 6 can be set in input or output mode in 1-bit units.

Ports 2, 3, 6, 8, and 9 can be connected with on-chip pull-up resistors in 4-bit units via software, by manipulating the pull-up resistor specification register group A and B (POGA and POGB). Port 5 can be connected with a pull-up resistor in 1-bit units by mask option. However, the μPD75P3116 is not provided with a pull-up resistor by mask option.

Ports 8 and 9 can be set in input or output mode in pairs in 8-bit units. When the $\overline{\text{RESET}}$ signal is asserted, ports 2, 3, 6, 8, and 9 are set in input mode (high impedance), and port 5 is set at the high-level (when the pull-up resistor of mask option is connected) or high-impedance state.

2.2.3 TI0 to TI2 ··· inputs shared with port 1

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

These pins become effective by selecting the external event pulse input for the count pulse (CP) with the timer/event counter mode register (TM0, TM1, TM2).

TI0 through TI2 are Schmitt trigger input pins.

Refer to **5.5.1 (1) Timer/event counter mode register (TM0, TM1, TM2)**.

2.2.4 PTO0 to PTO2 ··· outputs shared with port 2

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

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

Refer to **5.5.2 (3) Timer/event counter operation**.

2.2.5 PCL ··· output shared with port 2

This is a programmable clock output pin and is used to supply the clock to a peripheral LSI (such as a slave microcomputer). When the $\overline{\text{RESET}}$ signal is asserted, the contents of the clock 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**.

2.2.6 BUZ ··· output shared with port 2

This is a frequency output pin and is used to issue a buzzer sound or trim the system clock frequency by outputting a specified frequency (2, 4, or 32 kHz: 4.19 MHz of main system clock or 32.768 kHz of subsystem clock). This bit is valid only when the bit 7 (WM7) of the watch mode register (WM) is set to "1".

When the $\overline{\text{RESET}}$ signal is asserted, 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**.

2.2.7 $\overline{\text{SCK}}$, SO/SB0, and SI/SB1 ··· 3-state I/Os shared with port 0

These are serial interface I/O pins and operate in accordance with the setting of the serial operation mode register (CSIM). When 3-wire serial I/O mode is selected, $\overline{\text{SCK}}$ functions as the CMOS I/O, SO functions as the CMOS output, and SI functions as the CMOS input. When 2-wire serial I/O mode or SBI mode is selected, $\overline{\text{SCK}}$ functions as the CMOS I/O, and SB1 (SB0) functions as the N-ch open-drain I/O.

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

All these pins are Schmitt trigger input pins.

Refer to **5.6 Serial Interface**.

2.2.8 INT4 ··· input shared with port 0

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

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

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

2.2.9 INT0 and INT1 ··· inputs shared with port 1

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

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

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

(2) INT1 (bit 0 of IM1)

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

INT0 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. Beside that, a noise eliminator can be added with software. Two levels of the sampling clock for noise elimination are provided. The acknowledged signal width differs depending on the CPU operating clock.

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

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

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

INT0 and INT1 are Schmitt trigger input pins.

2.2.10 INT2 ··· input shared with port 1

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

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

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

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

2.2.11 KR0 to KR3 ··· inputs shared with port 6

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

The interrupt source can be selected from “KR2 and KR3” and “KR0 to KR3” by using the edge detection mode register (IM2).

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

2.2.12 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.

S16 through S19 and S20 through S23 are shared with ports 9 and 8, respectively, and the modes of these pins can be selected by using display mode register (LCDM).

2.2.13 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).

2.2.14 V_{LC0} to V_{LC2}

These are power supply pins to drive an LCD. With the μPD753108, a split resistor can be internally connected to the V_{LC0} through V_{LC2} pins by mask option, so that power to drive the LCD in accordance with each bias method can be supplied without an external split resistor. The μPD75P3116 does not have a mask option and is not provided with a split resistor.

2.2.15 BIAS

This is an output pin for dividing resistor cutting. It is connected to the V_{LC0} pin to supply various types of LCD driving voltages and is used to change a resistance division ratio, connect an external resistor along with the V_{CL0} through V_{CL2} pins and V_{SS} pin, and fine-tune the LCD driving supply voltage.

2.2.16 LCDCL

This is a clock output pin for driving an external LCD expansion driver.

2.2.17 SYNC

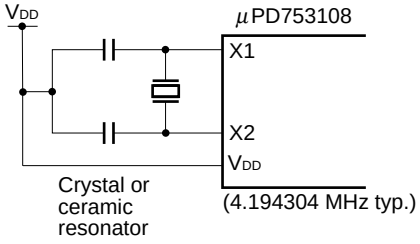
This is a clock output pin to synchronize an external LCD expansion driver.

2.2.18 X1 and X2

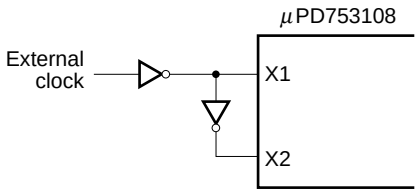
These pins connect a crystal/ceramic oscillator for main system clock oscillation.

An external clock can also be input to these pins.

(a) Crystal/ceramic oscillation



(b) External clock

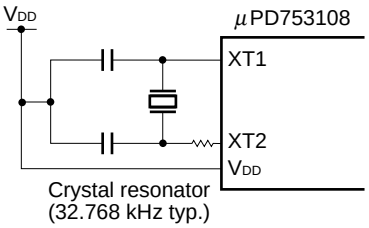


2.2.19 XT1 and XT2

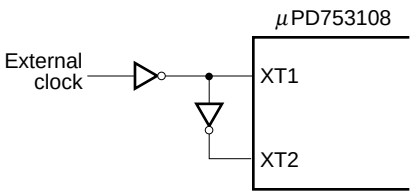
These pins connect a crystal oscillator for subsystem clock oscillation.

An external clock can also be input to these pins.

(a) Crystal oscillation



★ **(b) External clock**



Remark Refer to paragraph **5.2.2 (6)** when the subsystem clock is not used.

2.2.20 $\overline{\text{RESET}}$

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

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

This pin can be used to initialize and start the CPU, but also to release the STOP and HALT modes.

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

2.2.21 MD0 to MD3 (μPD75P3116 only) ⋯ inputs/outputs shared with port 3

These pins are provided to the μPD75P3116 only, and are used to select a mode when the program memory (one-time PROM) is written or verified.

★ **2.2.22 D0 to D7 (μPD75P3116 only) ⋯ inputs/outputs shared with ports 5 and 6**

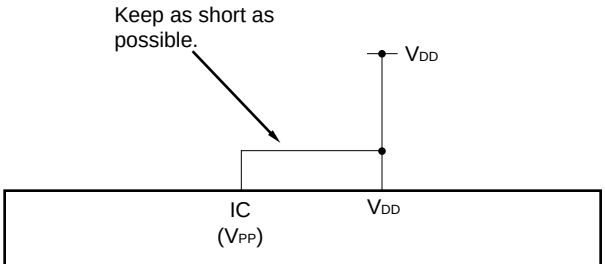
These pins are provided to the μPD75P3116 only, and are used as data bus pins when the program memory (one-time PROM) is written or verified.

2.2.23 IC (μPD753104, 753106, and 753108 only)

The IC (Internally Connected) pin sets a test mode in which the μPD753108 is tested before shipment. It is usually best to connect the IC pin directly to the V_{DD} pin with as short a wiring length as possible.

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

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



2.2.24 V_{PP} (μPD75P3116 only)

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

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

2.2.25 V_{DD}

Positive power supply pin.

2.2.26 V_{SS}

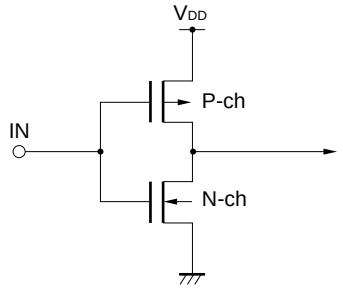
GND.

2.3 Pin Input/Output Circuits

The μPD753108 pin input/output circuits are shown schematically.

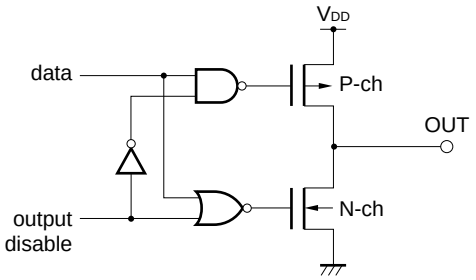
(1/3)

TYPE A



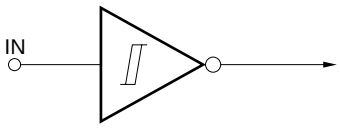
CMOS specification input buffer.

TYPE D



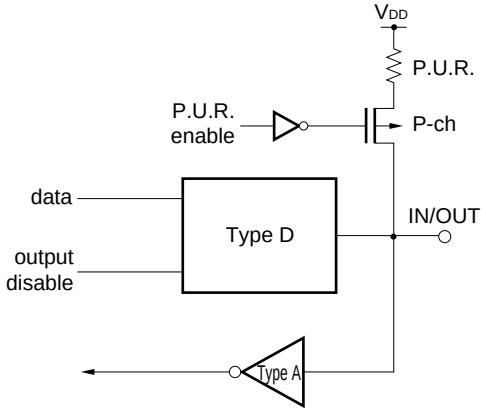
Push-pull output that can be placed in output high-impedance (both P-ch, N-ch off).

TYPE B



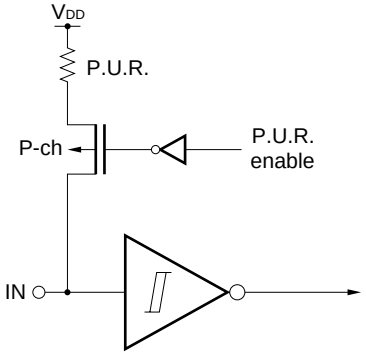
Schmitt trigger input having hysteresis characteristic.

TYPE E-B



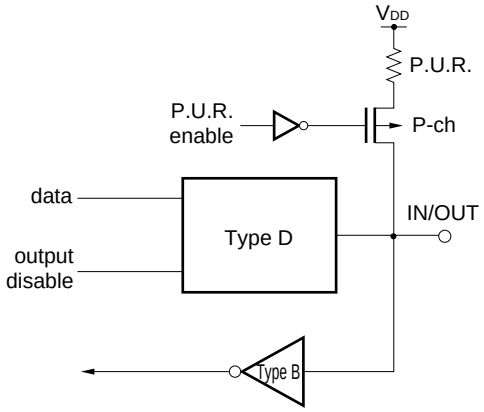
P.U.R. : Pull-Up Resistor

TYPE B-C



P.U.R. : Pull-Up Resistor

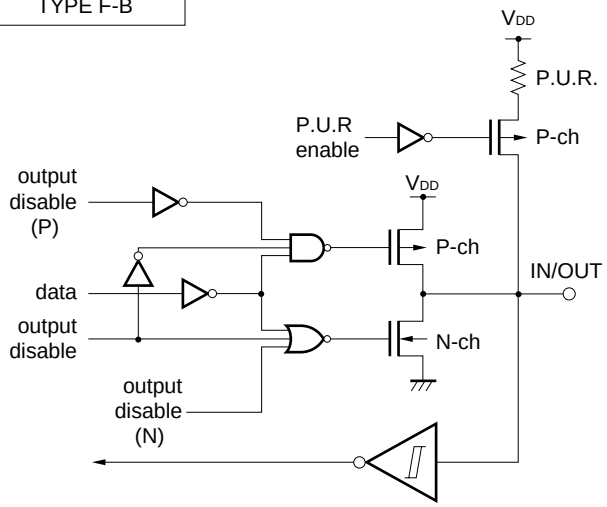
TYPE F-A



P.U.R. : Pull-Up Resistor

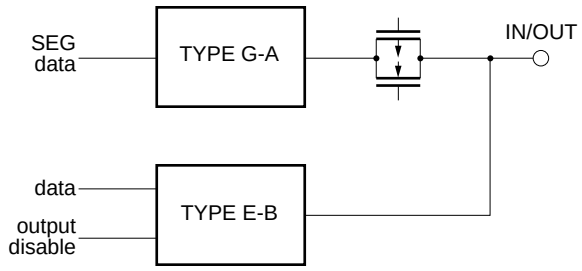
(2/3)

TYPE F-B

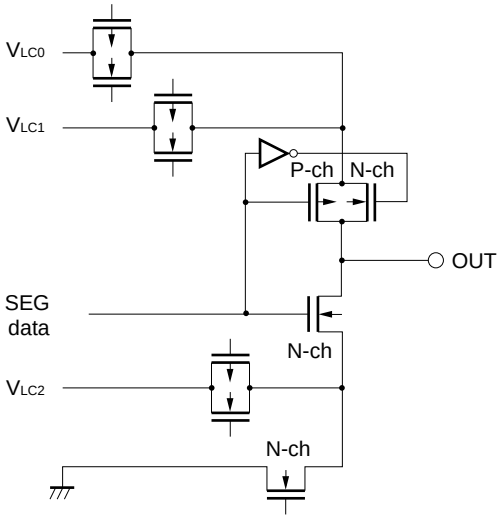


P.U.R. : Pull-Up Resistor

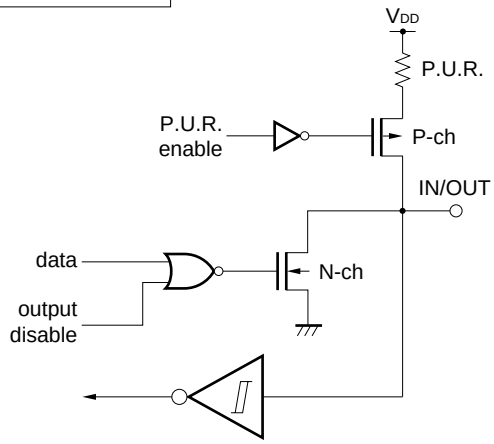
TYPE H



★ TYPE G-A

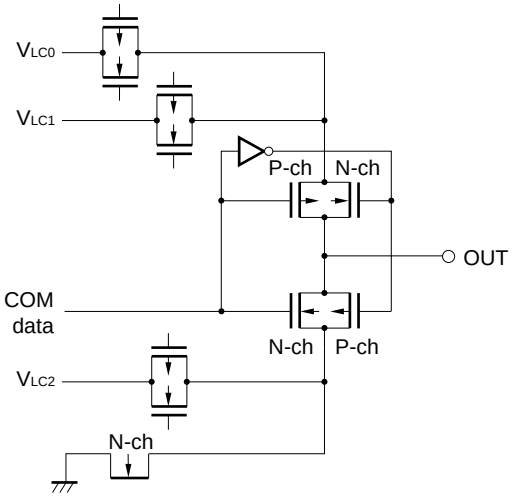


TYPE M-C

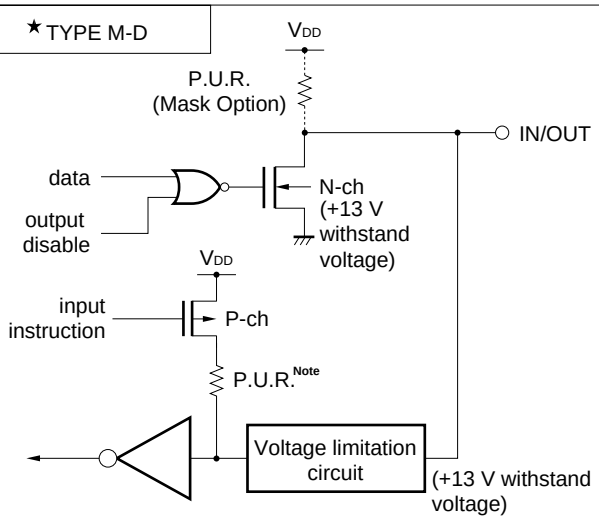


P.U.R. : Pull-Up Resistor

★ TYPE G-B



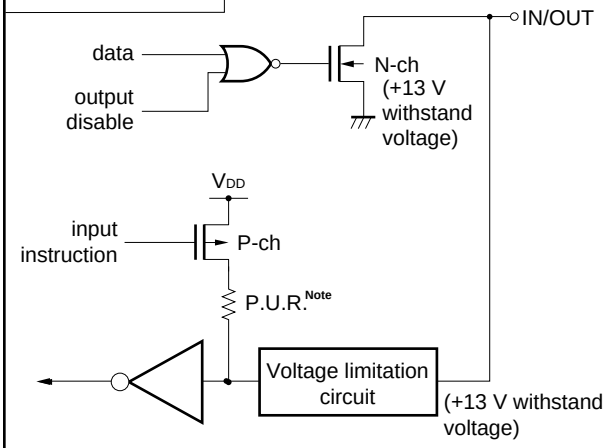
★ TYPE M-D



Note Pull-up resistor that operates only when an input instruction is executed if on-chip pull-up resistors are not specified by mask option (current flows from V_{DD} to the pin when the pin is low).

(3/3)

★ TYPE M-E



Note Pull-up resistor that operates only when an input instruction is executed (current flows from V_{DD} to the pin when the pin is low).

2.4 Recommended Connections for Unused Pins

★ **Table 2-3. List of Recommended Connections for Unused Pins**

Pin	Recommended Connection
P00/INT4	Connect to V_{SS} or V_{DD}.
P01/$\overline{SCK}$	Connect to V_{SS} or V_{DD} individually via resistor.
P02/SO/SB0	
P03/SI/SB1	Connect to V_{SS}.
P10/INT0	Connect to V_{SS} or V_{DD}.
P11/INT1	
P12/TI1/TI2/INT2	
P13/TI0	
P20/PTO0	Input state : Connect to V_{SS} or V_{DD} individually via resistor. Output state: Leave open.
P21/PTO1	
P22/PCL/PTO2	
P23/BUZ	
P30/LCDCL (/MD0)**Note 1**	
P31/SYNC (/MD1)**Note 1**	
P32 (/MD2)**Note 1**	
P33 (/MD3)**Note 1**	
P50 (/D4)**Note 1**	Connect to V_{SS} (Do not connect pull-up resistor by mask option).
P51 (/D5)**Note 1**	
P52 (/D6)**Note 1**	
P53 (/D7)**Note 1**	
P60/KR0 (/D0)**Note 1**	Input state : Connect to V_{SS} or V_{DD} individually via resistor. Output state: Leave open.
P61/KR1 (/D1)**Note 1**	
P62/KR2 (/D2)**Note 1**	
P63/KR3 (/D3)**Note 1**	
S0 to S15	Leave open.
COM0 to COM3	
S16/P93 to S19/P90	Input state : Connect to V_{SS} or V_{DD} individually via resistor. Output state: Leave open.
S20/P83 to S23/P80	
V_{LC0} to V_{LC2}	Connect to V_{SS}.
BIAS	Only if all of V_{LC0} to V_{LC2} are unused, connect to V_{SS}. In other cases, leave open.
XT1**Note 2**	Connect to V_{SS} or V_{DD}.
XT2**Note 2**	Leave open.
IC (V_{PP})**Note 1**	Connect directly to V_{DD}.

Notes 1. ( ): μPD75P3116 only

2. When not using the subsystem clock, select SOS.0 = 1 (the on-chip feedback resistor is not used).

[MEMO]

CHAPTER 3 FEATURES OF ARCHITECTURE AND MEMORY MAP

The 75XL architecture employed for the μPD753108 has the following features:

- Internal RAM: 4K words × 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 μPD753108 is provided with static RAM at the addresses 000H through 1FFH of the data memory space, of which a 24 × 4 bit area of addresses 1E0H through 1F7H can also be used as a display data memory. Peripheral hardware units (such as I/O ports and timers) are allocated to addresses F80H through FFFH.

The μPD753108 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 (4K words × 4 bits) is addressed.

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

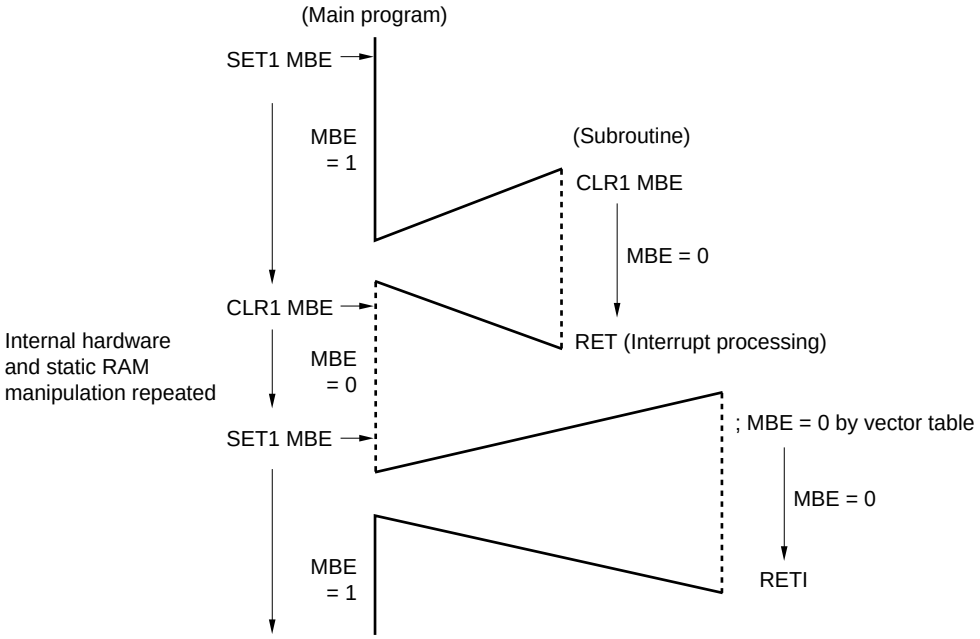
- Memory bank enable flag (MBE)
- Memory bank selection register (MBS)

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, so that the data memory space can be expanded.

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

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

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



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 μPD753108 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.

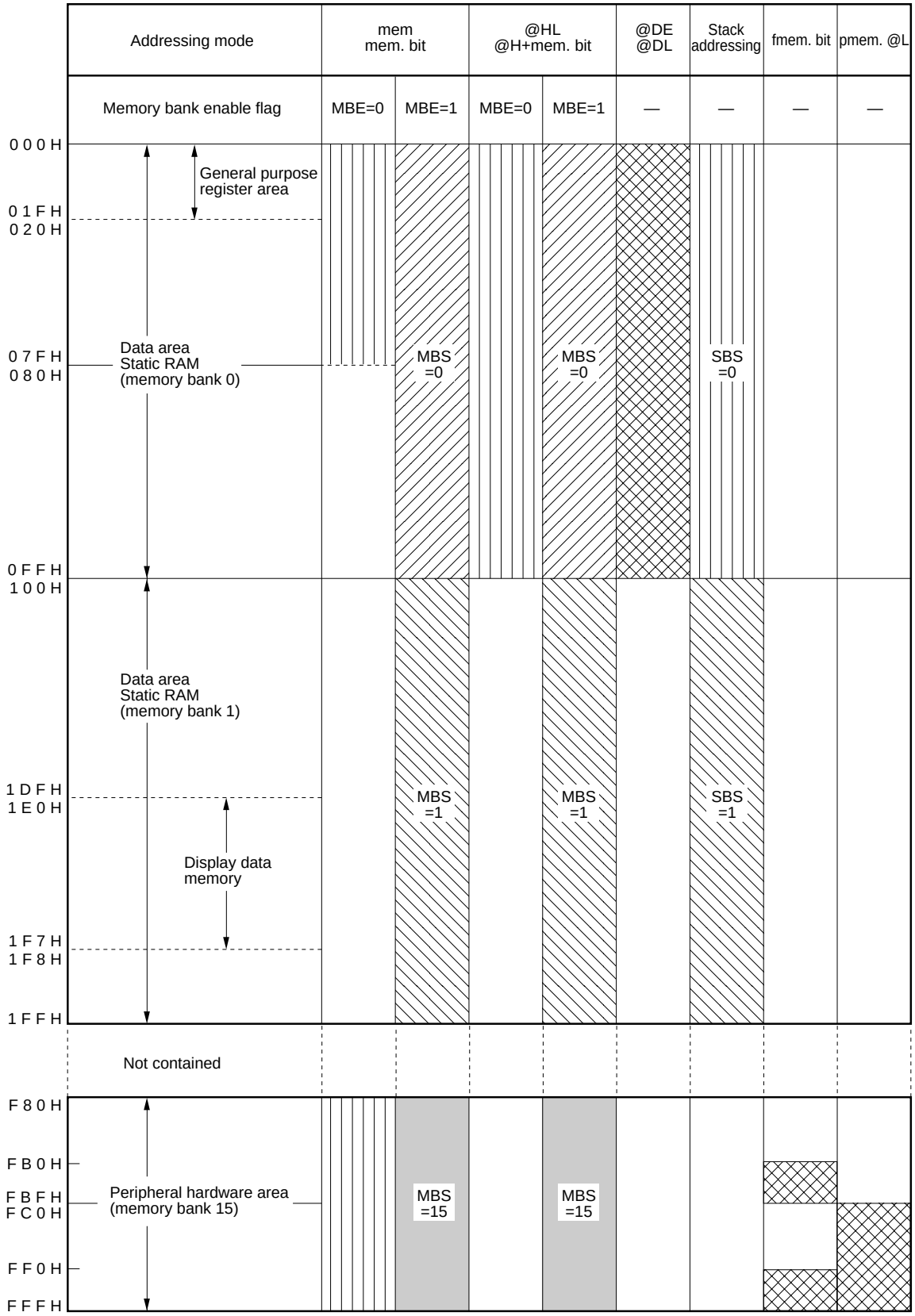
In the mode of MBE = 1, MB = MBS; therefore, the entire data memory space can be addressed.

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

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

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

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

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



Remark –: don't care

Table 3-1. Addressing Mode

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

(2) 4-bit direct addressing (mem)

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

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

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

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

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

```
BUFF    EQU     11AH        ; “BUFF” is at address 11AH
        SET1    MBE         ; MBE ← 1
        SEL     MB1         ; MBS ← 1
        MOV     A, BUFF     ; A ← (BUFF)
        SEL     MB15        ; MBS ← 15
        OUT     PORT5, A    ; PORT5 ← A
```

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

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

(3) 8-bit direct addressing (mem)

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

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

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

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

Examples 1. To transfer the 8-bit data of ports 8 and 9 to addresses 20H and 21

```
DATA    EQU     020H
        CLR1    MBE             ; MBE ← 0
        IN      XA, PORT8       ; X ← port 9, A ← port 8
        MOV     DATA, XA        ; (21H) ← X, (20H) ← A
```

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

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

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

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

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

Example To transfer data 50H through 57H to addresses 110H through 117H

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

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

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

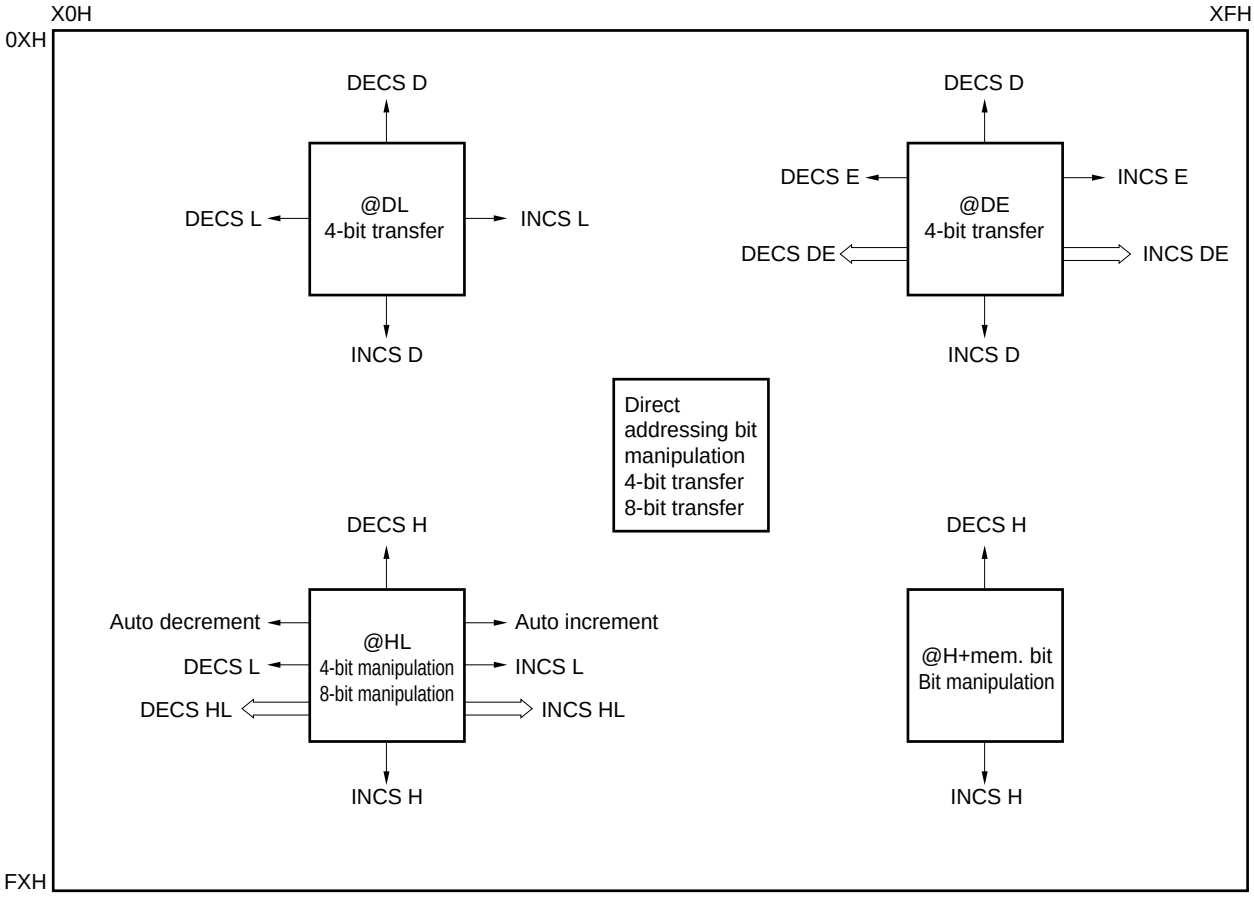
Examples 1. To compare data 50H through 57H with data 110H through 117H

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

2. 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  L             ; L ← L+1
        BR    LOOP
        INCS  H             ; H ← H+1
        BR    LOOP
```

Figure 3-3. Static RAM Address Update Method



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

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

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

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

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

```
DATA    EQU    30H
        CLR1   MBE
        MOV    HL, #DATA
        MOV    XA, T0          ; XA ← count register 0
        SKE    A, @HL          ; A = (HL)?
        BR     NO
        INCS   L
        MOV    A, X            ; A ← X
        SKE    A, @HL          ; A = (HL)?
```

2. To clear data memory at 00H through FFH

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

(6) Bit manipulation addressing

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

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

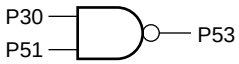
(a) Specific address bit direct addressing (fmem.bit)

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 FF9H, to which the I/O ports are mapped, and FB0H through FBH, 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 flag and reset P63

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

Examples 2. To reset P53 if both P30 and P51 pins are 1




```
(i)          SET1    CY              ; CY ← 1
             AND1    CY, PORT3.0     ; CY ∧ P30
             AND1    CY, PORT5.1     ; CY ∧ P51
             SKT     CY              ; CY = 1?
             BR      SETP
             CLR1    PORT5.3         ; P53 ← 0
             ⋮
SETP:        SET1    PORT5.3         ; P53 ← 1
             ⋮

(ii)         SKT     PORT3.0         ; P30 = 1?
             BR      SETP
             SKT     PORT5.1         ; P51 = 1?
             BR      SETP
             CLR1    PORT5.3         ; P53 ← 0
             ⋮
SETP:        SET1    PORT5.3         ; P53 ← 1
```

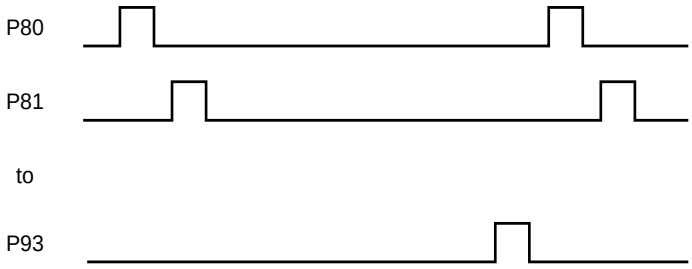
(b) Specific address bit register indirect addressing (pmem. @L)

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

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

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

Example To output pulses to the respective bits of ports 8 and 9




```
LOOP2:  MOV     L, #0
LOOP1:  SET1    PORT8.@L        ; Bits of ports 8 and 9 (L1-0) ← 1
        CLR1    PORT8.@L        ; Bits of ports 8 and 9 (L1-0) ← 0
        INCS    L
        NOP
        SKE     L, #08H
        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
        CLR1    CY              ; CY ← 0
        OR1     CY, @H+FLAG1    ; CY ← CY ∨ FLAG1
        XOR1    CY, @H+FLAG2    ; CY ← CY ∀ FLAG2
        SET1    @H+FLAG3        ; FLAG3 ← 1
        SKT     CY              ; CY = 1?
        CLR1    @H+FLAG3        ; FLAG3 ← 0
```

(7) Stack addressing

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

The address of data memory bank 0 pointed to by the stack pointer (8 bits) is specified in this addressing mode.

This addressing is also used to save or restore register contents by using the PUSH or POP instruction, in addition to during interrupt processing or subroutine processing.

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

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

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

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

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

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

3.2 Bank Configuration of General-Purpose Registers

The μPD753108 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	×	×	Fixed to bank 0
1	0	0	0	0	Bank 0 selection
			0	1	Bank 1 selection
			1	0	Bank 2 selection
			1	1	Bank 3 selection

Fixed to 0

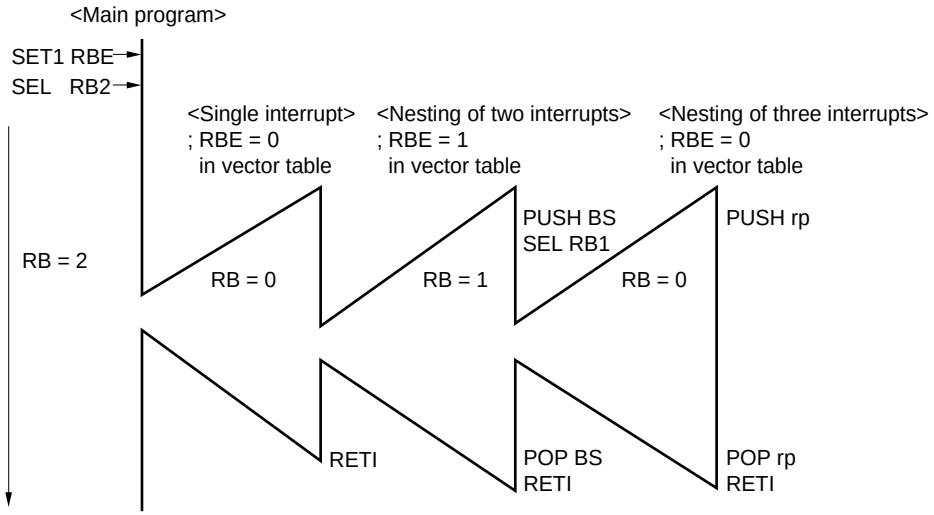
Remark × = don't care

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

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

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

Figure 3-4. Example of Using Register Banks



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

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

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

The general-purpose register area provided to the μPD753108 can be used not only as 4-bit registers, but also as 8-bit register pairs. This feature allows the μPD753108 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 plays a central role in transferring, operating, and comparing 4-bit data as a 4-bit accumulator. The other registers can transfer, compare, and increment or decrement data with the accumulator.

(2) To use as 8-bit registers

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

Examples 1.

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

2. To test whether the value of the count register (T0) of timer/event counter 0 is greater than the value of register pair BC' 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)

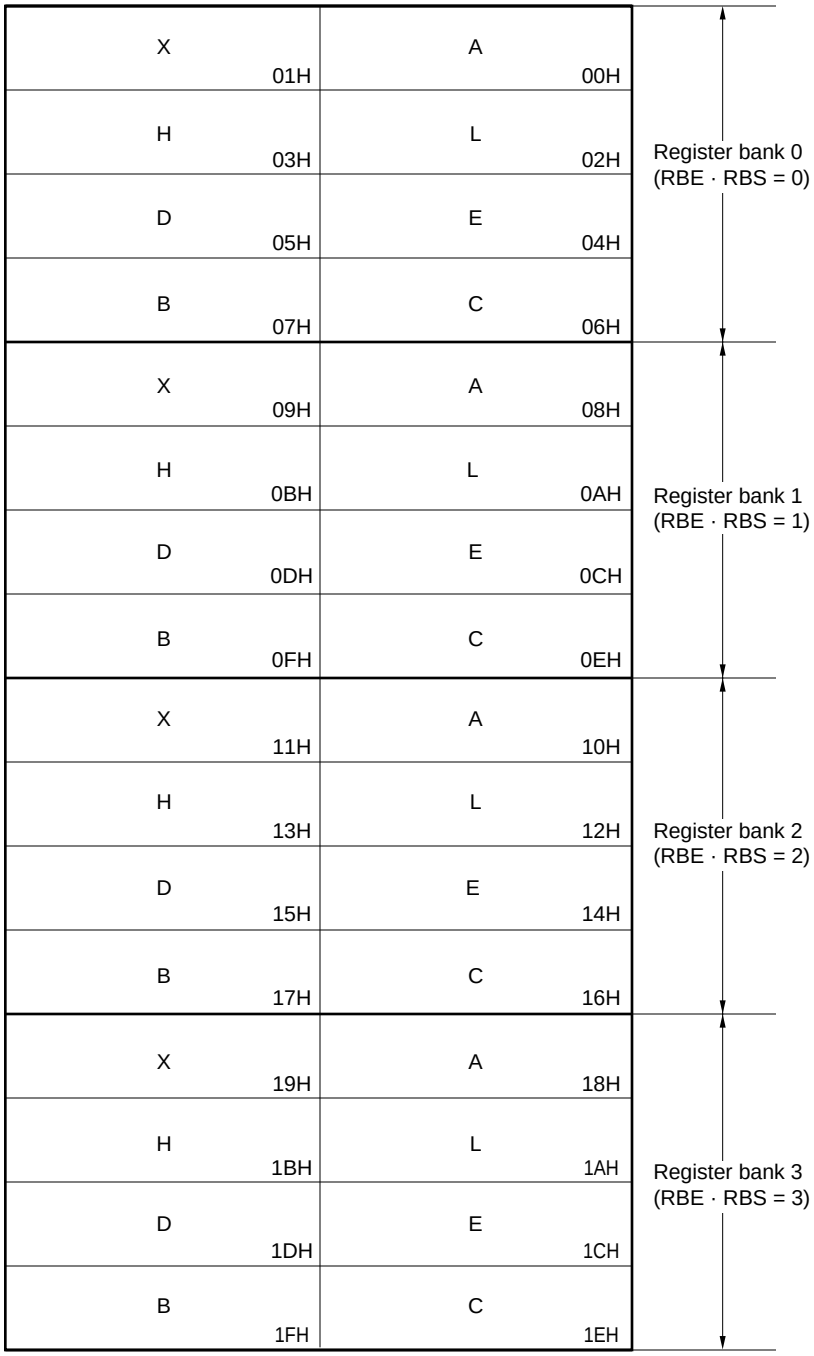
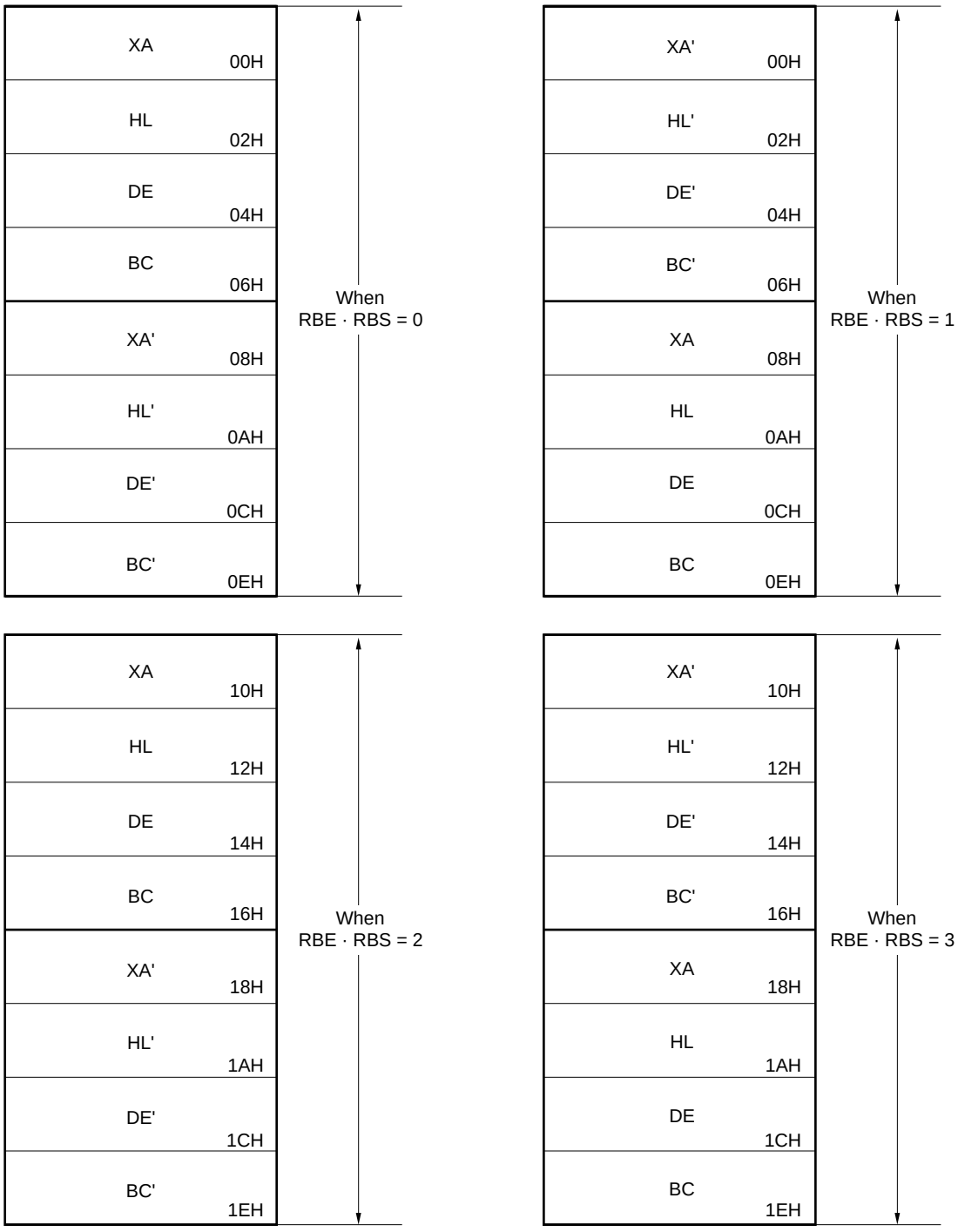


Figure 3-6. General-Purpose Register Configuration (for 8-bit operation)



3.3 Memory-Mapped I/O

The μPD753108 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 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.	BSBn.X PORTn.X
4-bit manipulation	Specified by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15).	All the hardware for which 4-bit manipulation is possible
	Specified by register indirect addressing @HL with (MBE = 1, MBS = 15).	
8-bit manipulation	Specified by direct addressing mem with MBE = 0 or (MBE = 1, MBS = 15). Note that mem must be an even-number address.	All the hardware for which 8-bit manipulation is possible
	Specified by register indirect addressing @HL with MBE = 1, MBS = 15. Note that the contents of the L register are an even number.	

Example

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

The I/O map of the μPD753108 is shown in Figure 3-7.
The meanings of the items in Figure 3-7 are as follows.

- Hardware name (symbol) ..... A name 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 ..... Indicates the usable bit manipulation addressing when bit manipulation is performed addressing on the hardware.

Figure 3-7. μPD753108 I/O Map (1/5)

Address	Hardware Name (symbol) b3	b2	b1	b0	R/W	Manipulation Unit 1-bit	4-bit	8-bit	Bit Manipulation Addressing	Remarks
F80H	Stack pointer (SP)				R/W	—	—	Yes	—	Bit 0 is fixed to 0.
F82H	Register bank selection register (RBS)				R	—	Yes	Yes	—	**Note 1**
F83H	Bank selection register (BS) / 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	—	
F88H	Modulo register for setting timer/ event counter high level (TMOD2H)				R/W	—	—	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	—	—	

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

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

Figure 3-7. μPD753108 I/O Map (2/5)

Address	Hardware Name (symbol) b3	b2	b1	b0	R/W	Manipulation Unit 1-bit	4-bit	8-bit	Bit Manipulation Addressing	Remarks
F90H	Timer/event counter 2 mode register (TM2)				R/W	Δ (W)	—	Yes	—	Bit manipulation can be performed only on bit 3.
						—	—		—	
F92H	TOE2	REMC	NRZB	NRZ	R/W	Yes	Yes	Yes	—	Bit 3 can be written only.
	Timer/event counter 2 control register (TC2)					Δ	—			Bit manipulation can be performed only on bit 3.
	TGCE	—	—	—						
F94H	Timer/event counter 2 count register (T2)				R	—	—	Yes	—	
F96H	Timer/event counter 2 modulo register (TMOD2)				R/W	—	—	Yes	—	
F98H	Watch mode register (WM)				R/W	Δ (R)	—	Yes	—	Bit manipulation can be performed only on bit 3.
						—	—			
FA0H	Timer/event counter 0 mode register (TM0)				R/W	Δ (W)	—	Yes	mem.bit	Bit manipulation can be performed only on bit 3.
						—	—		—	
FA2H	TOE0 Note 1				W	Yes	—	—	mem.bit	
FA4H	Timer/event counter 0 count register (T0)				R	—	—	Yes	—	
FA6H	Timer/event counter 0 modulo register (TMOD0)				R/W	—	—	Yes	—	
FA8H	Timer/event counter 1 mode register (TM1)				R/W	Δ (W)	—	Yes	mem.bit	Bit manipulation can be performed only on bit 3.
						—	—		—	
FAAH	TOE1 Note 2				W	Yes	—	—	mem.bit	
FACH	Timer/event counter 1 count register (T1)				R	—	—	Yes	—	
FAEH	Timer/event counter 1 modulo register (TMOD1)				R/W	—	—	Yes	—	

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

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

Figure 3-7. μPD753108 I/O Map (3/5)

Address	Hardware Name (symbol) b3	b2	b1	b0	R/W	Manipulation Unit 1-bit	4-bit	8-bit	Bit Manipulation Addressing	Remarks
FB0H	IST1	IST0	MBE	RBE	R/W	Yes (R/W)	Yes (R/W)	Yes (R)	fmem.bit	8-bit manipulation is read only.
	Program status word (PSW)									
	CY Note 1	SK2 Note 1	SK1 Note 1	SK0 Note 1		Δ Note 2	—			
FB2H	Interrupt priority selection register (IPS)				R/W	—	Yes	—		**Note 3**
FB3H	Processor clock control register (PCC)				R/W	—	Yes	—		**Note 4**
FB4H	INT0 edge detection mode register (IM0)				R/W	—	Yes	—	—	
FB5H	INT1 edge detection mode register (IM1)				R/W	—	Yes			
FB6H	INT2 edge detection mode register (IM2)				R/W	—	Yes			
FB7H	System clock control register (SCC)				R/W	Δ	Yes	—	—	Bit manipulation can be performed only on bits 0, 3.
FB8H	INTA register (INTA) IE4	IRQ4	IEBT	IRQBT	R/W	Yes	Yes	—	fmem.bit	
FBAH	INTC register (INTC)		IEW	IRQW	R/W	Yes	Yes			
FBCH	INTE register (INTE) IET1	IRQT1	IET0	IRQT0	R/W	Yes	Yes	—		
FBDH	INTF register (INTF) IET2	IRQT2	IECSI	IRQCSI	R/W	Yes	Yes			
FBEH	INTG register (INTG) IE1	IRQ1	IE0	IRQ0	R/W	Yes	Yes	—		
FBFH	INTH register (INTH)		IE2	IRQ2	R/W	Yes	Yes			

FC0H	Bit sequential buffer 0 (BSB0)	R/W	Yes	Yes	Yes	mem.bit pmem.@L		
FC1H	Bit sequential buffer 1 (BSB1)	R/W	Yes	Yes				
FC2H	Bit sequential buffer 2 (BSB2)	R/W	Yes	Yes	Yes			
FC3H	Bit sequential buffer 3 (BSB3)	R/W	Yes	Yes				
FCFH	Subsystem clock oscillator control register (SOS)	R/W	—	Yes	—	—		

Remarks 1. IEXXX : Interrupt enable flag

2. IRQXXX: Interrupt request flag

Notes 1. Not registered as a reserved word.

2. Write into CY with CY manipulation instruction.

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

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

Figure 3-7. μPD753108 I/O Map (4/5)

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

FE0H	Serial operation mode register (CSIM)				R/W	—	—	Yes	—	**Note 1**
	CSIE	COI	WUP			Δ (R) (W)	—		mem.bit	
FE2H	CMDD	RELD	CMDT	RELT	R/W	Yes	—	—	mem.bit	R/W depends on the bit number.
	SBI control register (SBIC)									
	BSYE	ACKD	ACKE	ACKT						
FE4H	Serial I/O shift register (SIO)				R/W	—	—	Yes	—	
FE6H	Slave address register (SVA)				R/W	—	—	Yes	—	
FE8H	PM33 Note 2	PM32 Note 2	PM31 Note 2	PM30 Note 2	R/W	—	—	Yes	—	
	Port mode register group A (PMGA)									
	PM63 Note 2	PM62 Note 2	PM61 Note 2	PM60 Note 2						
FECH	—	PM2 Note 2	—	—	R/W	—	—	Yes	—	
	Port mode register group B (PMGB)									
	—	—	PM5 Note 2	—						
FEEH	—	—	PM9 Note 2	PM8 Note 2	R/W	—	—	Yes	—	
	Port mode register group C (PMGC)									
	—	—	—	—						

Notes 1. For the 1-bit manipulation, R/W depends on the bit number.

2. Not registered as a reserved word.

Figure 3-7. μPD753108 I/O Map (5/5)

Address	Hardware Name (symbol) b3	b2	b1	b0	R/W	Manipulation Unit 1-bit	4-bit	8-bit	Bit Manipulation Addressing	Remarks
FF0H	Port 0			(PORT0)	R	Yes	Yes	—	—	
FF1H	Port 1			(PORT1)	R	Yes	Yes			
FF2H	Port 2			(PORT2)	R/W	Yes	Yes	—		
FF3H	Port 3			(PORT3)	R/W	Yes	Yes			
FF5H	Port 5			(PORT5)	R/W	Yes	Yes	—		
FF6H	$\overline{KR3}$ Port 6	$\overline{KR2}$	$\overline{KR1}$	$\overline{KR0}$ ($\overline{PORT6}$)	R/W	Yes	Yes	—		
FF8H	Port 8			(PORT8)	R/W	Yes	Yes	Yes	fmem.bit	
FF9H	Port 9			(PORT9)	R/W	Yes	Yes		pmem.@L	

CHAPTER 4 INTERNAL CPU FUNCTIONS

4.1 Switching Function between Mk I Mode and Mk II Mode

4.1.1 Difference between Mk I and Mk II modes

The CPU of the μPD753108 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 the μPD75308B.
  Can be used in the 75XL CPU with a ROM capacity of up to 16 Kbytes.
- Mk II mode : Incompatible with the μPD75308B.
  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 Mk II mode supports the program area that exceeds 16 Kbytes in 75X and 75XL Series. A software compatibility with devices whose ROM capacity is more than 16 Kbytes can be improved with this mode.**
In Mk II mode, the stack byte number (use area) during subroutine call instruction execution increases one byte per one stack compared to that of Mk I mode. Moreover, the machine cycle is lengthened for one cycle at the execution of CALL !addr or CALLF !faddr instruction.
Therefore, Mk I mode is recommended for applications with a focus on the RAM efficiency or processing performance rather than the software compatibility.

4.1.2 Setting method of stack bank selection register (SBS)

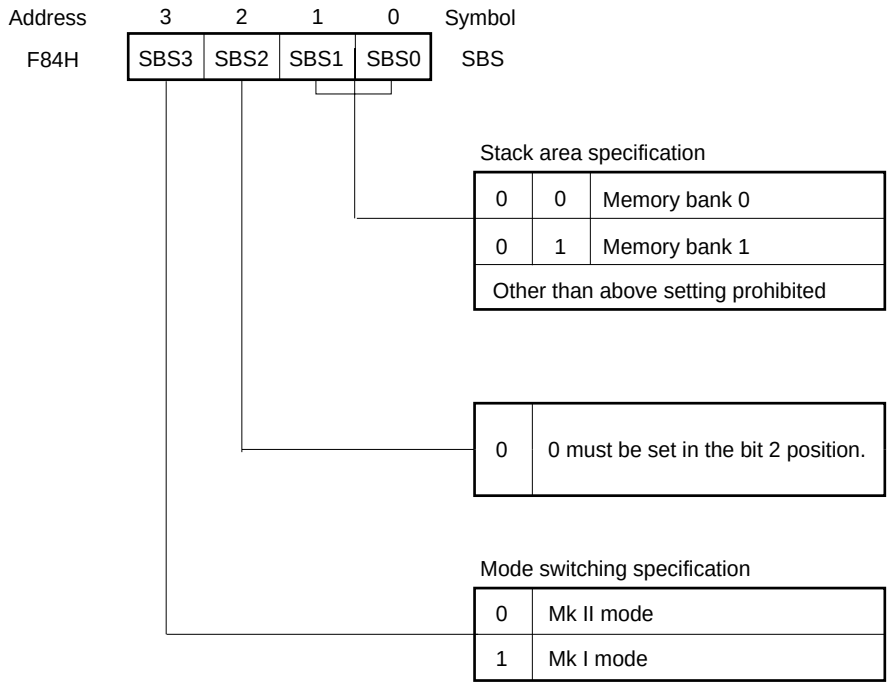
Switching between the Mk I mode and Mk II mode can be done by the SBS. Figure 4-1 shows the format.

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

When using the Mk I mode, the SBS must be initialized to 100XB[Note] at the beginning of a program. When using the Mk II mode, it must be initialized to 000XB[Note].

Note Set a desired value to X.

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 (μPD753104)
13-bit (μPD753106, 753108)
14-bit (μPD75P3116)

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

Figure 4-2. Program Counter Configuration

(a) μPD753104

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

(b) μPD753106, 753108

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

(c) μPD75P3116

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

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

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

When a subroutine call instruction (CALL, CALLA, or CALLF) is executed or when a 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 the return instruction (RET, RETS, or RETI) is executed, the contents of the stack memory are set to the PC.

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

μPD753104	: $PC_{11-8} \leftarrow (0000H)_{3-0}$,	$PC_{7-0} \leftarrow (0001H)_{7-0}$
μPD753106, 753108	: $PC_{12-8} \leftarrow (0000H)_{4-0}$,	$PC_{7-0} \leftarrow (0001H)_{7-0}$
μPD75P3116	: $PC_{13-8} \leftarrow (0000H)_{5-0}$,	$PC_{7-0} \leftarrow (0001H)_{7-0}$

4.3 Program Memory (ROM) ·····4096× 8 bits (μPD753104)
6144 × 8 bits (μPD753106)
8192 × 8 bits (μPD753108)
16384 × 8 bits (μPD75P3116)

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

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

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

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

- 000H to FFFH : μPD753104
- 0000H to 17FFH: μPD753106
- 0000H to 1FFFH: μPD753108
- 0000H to 3FFFH: μPD75P3116

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 at any address.

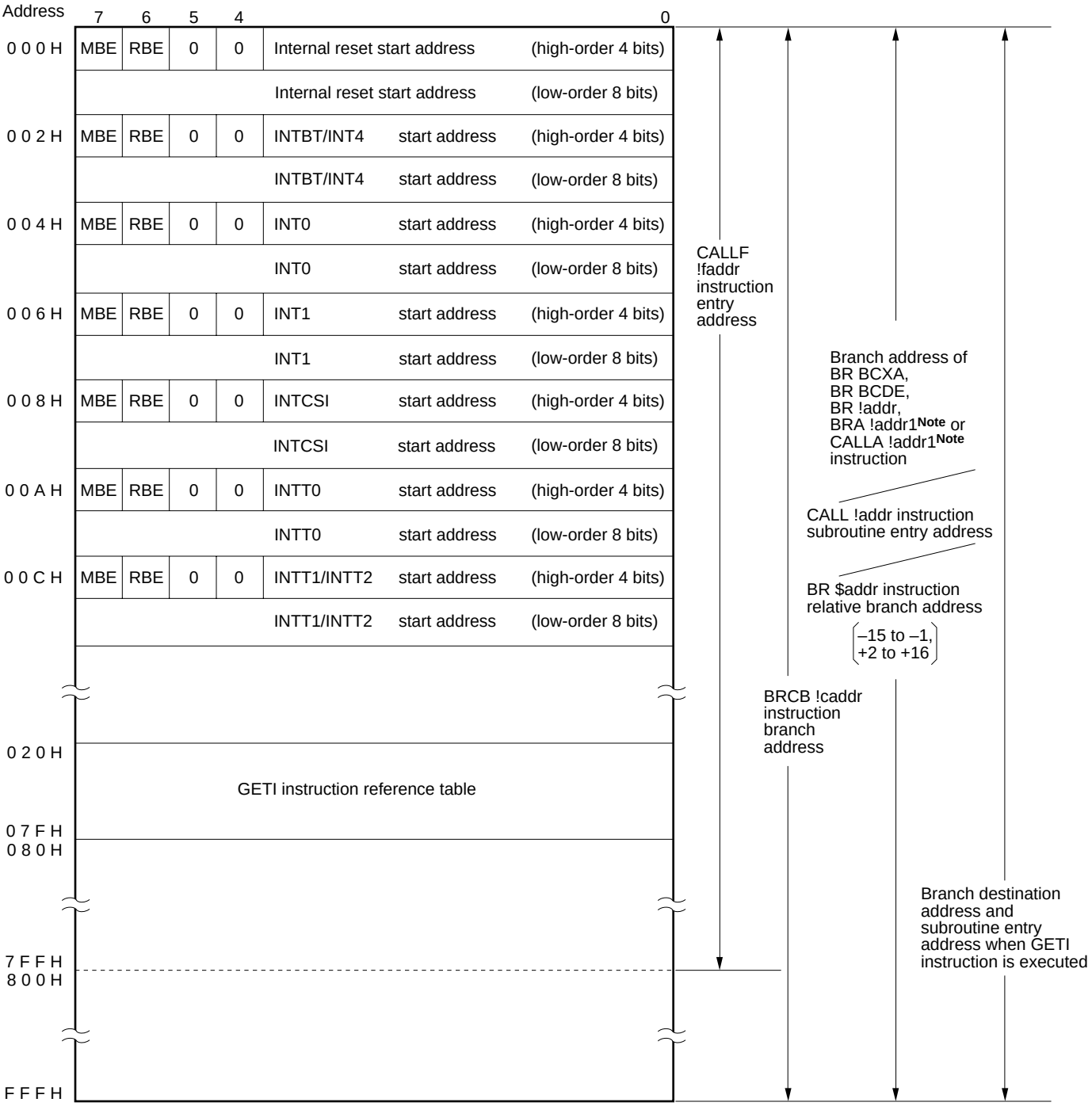
- Addresses 0002H to 000DH
  Vector table wherein the program start address and values set for the RBE and MBE by the vectored interrupts are written. Interrupt execution can be started at any address.

- 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 section **11.1.1 GETI instruction**).

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

★ **(a) μPD753104**

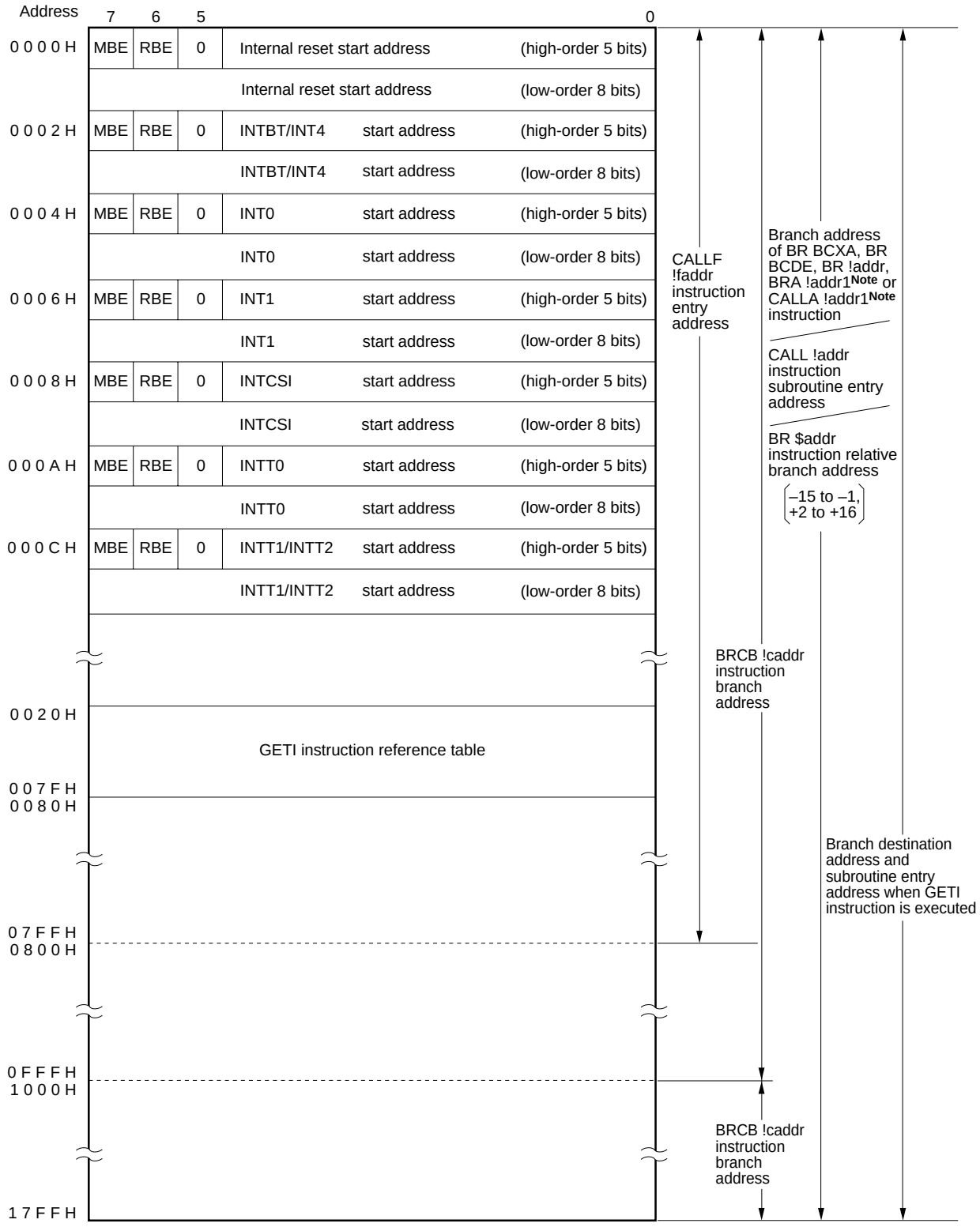


Note Can be used only in the Mk II mode.

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.

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

(b) μPD753106

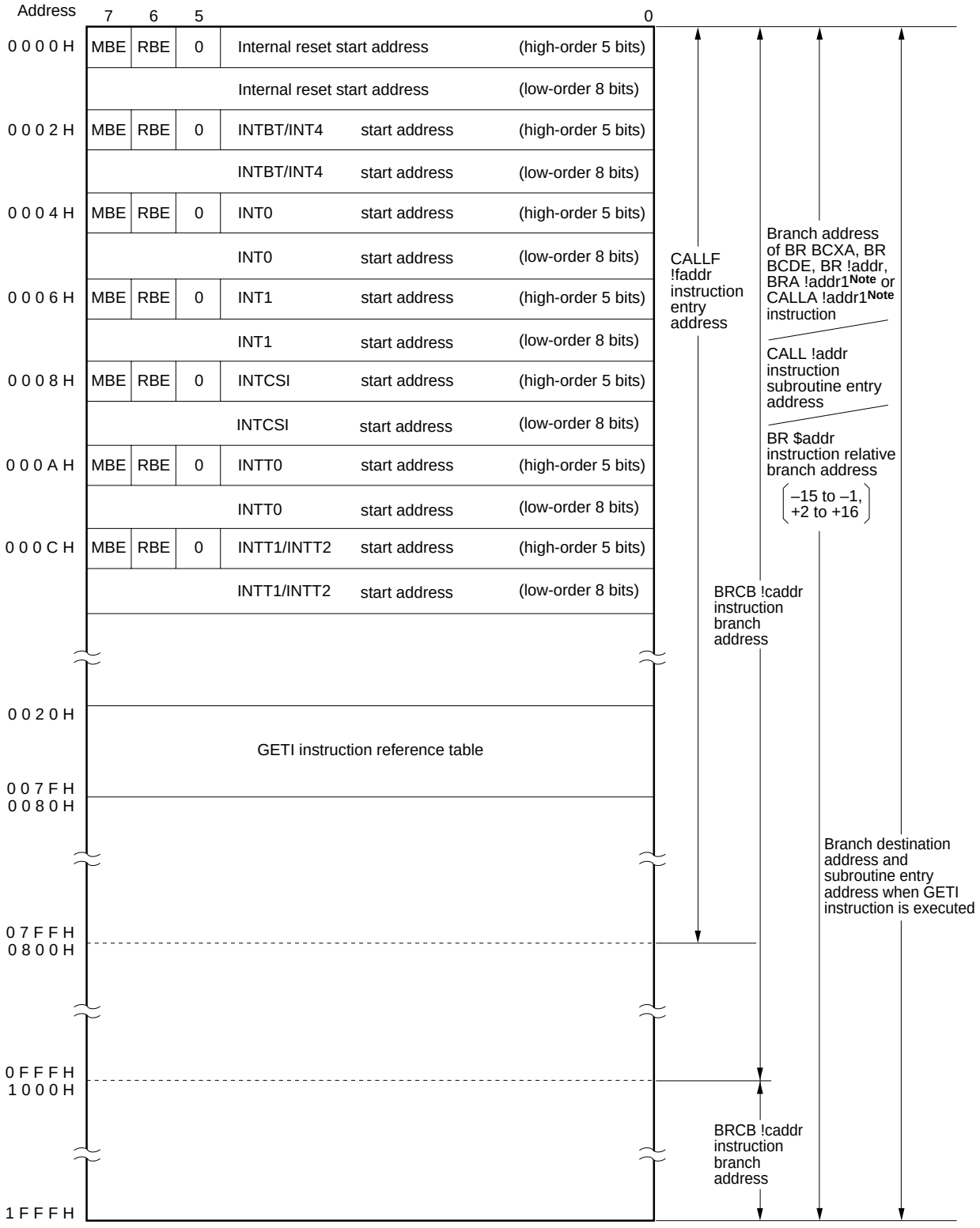


Note Can be used only in the Mk II mode.

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.

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

(c) μPD753108

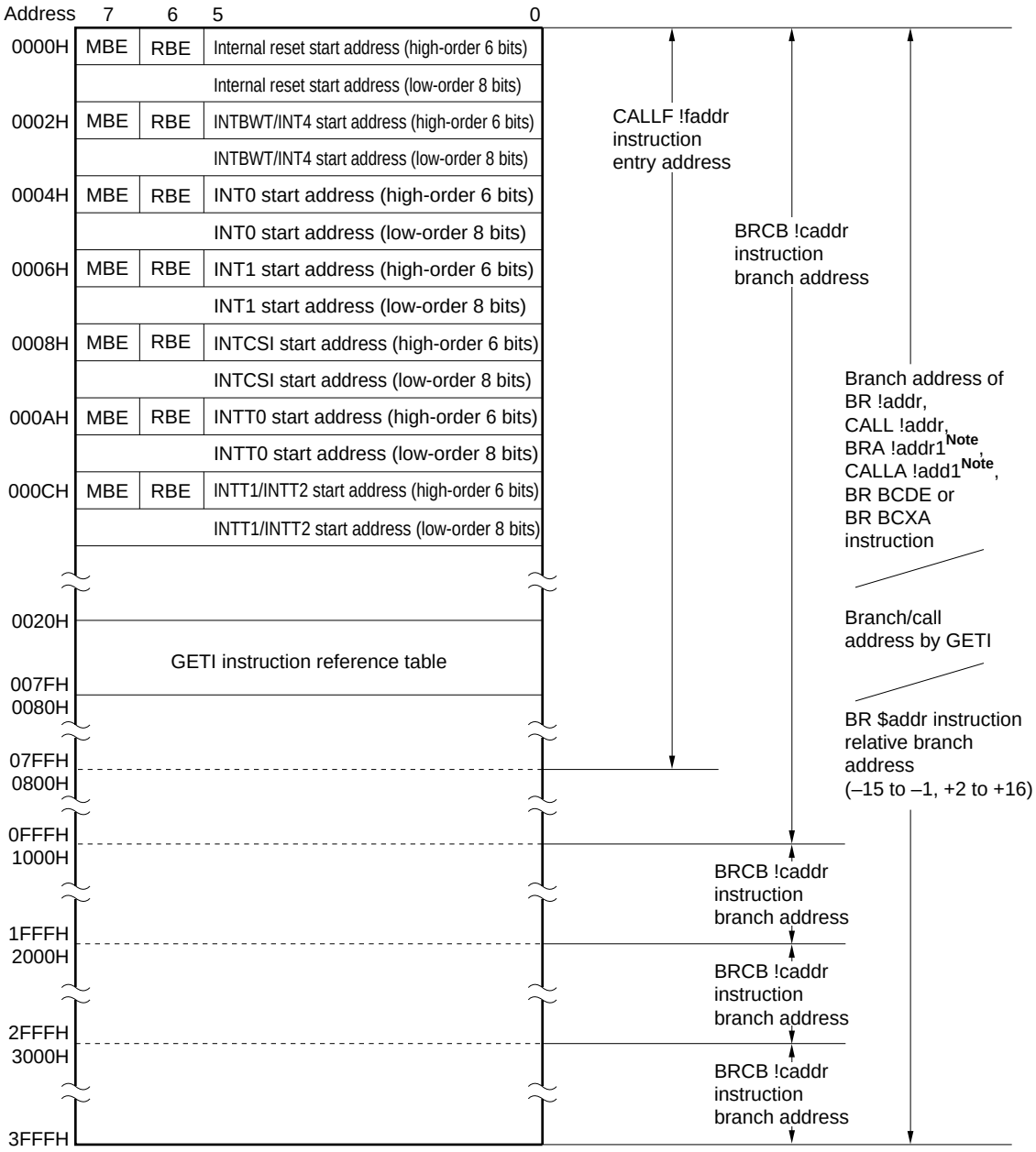


Note Can be used only in the Mk II mode.

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.

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

(d) μPD75P3116



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) ··· 512 words × 4 bits

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

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

4.4.1 Configuration of data memory

(1) Data area

A data area consists of static RAM, and is used to store data and as a stack memory when a subroutine or interrupt is executed. The contents of this area can be backed up for a long time by batteries even when the CPU is stopped in the standby mode. The data area is manipulated by using memory manipulation instructions.
Static RAM is mapped to memory banks 0 and 1 in units of 256 × 4 bits each. 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[Note 1] (000H through 1FFH). Bank 1 can be used as a display data memory (1E0H through 1F7H).
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. One stack area can be selected from memory bank 0 or 1.
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.

- **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.

- **Display data memory**
  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 can be used as data area addresses.

(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

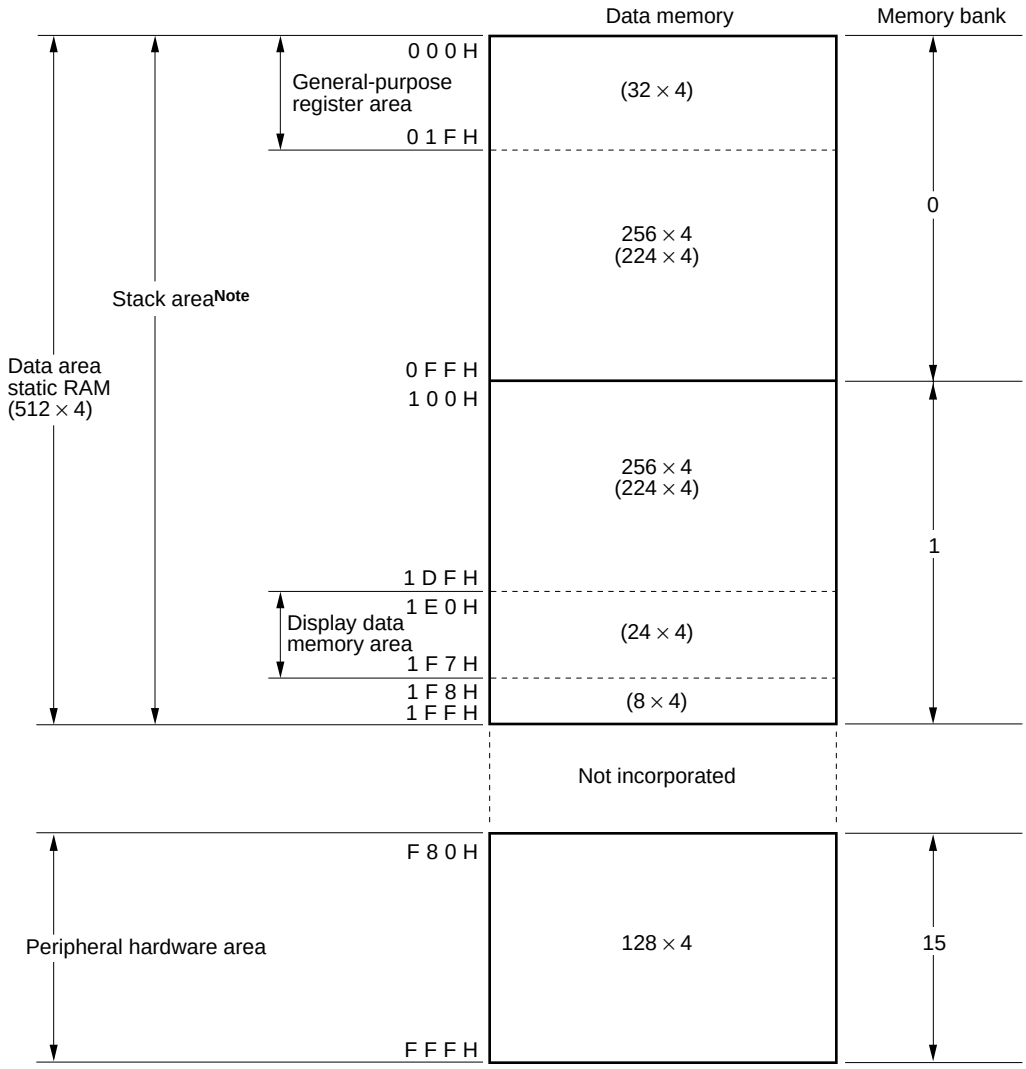
A memory bank is specified by setting a 4-bit memory bank selection register (MBS) to 0, 1, or 15 when bank specification is enabled by setting a memory bank enable flag (MBE) to 1. When bank specification is disabled (MBE = 0), bank 0 or 15 is automatically specified depending on the addressing mode 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 section **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 section **4.5 General-Purpose Registers**.
- Stack area..... See section **4.7 Stack Pointer (SP) and Stack Bank Selection Register (SBS)**.
- Display data memory..... See section **5.7.6 Display data memory**.
- Peripheral hardware area..... See **CHAPTER 5 PERIPHERAL HARDWARE FUNCTION**.

Figure 4-4. Data Memory Map



Note Either memory bank 0 or 1 can be assigned to the stack area.

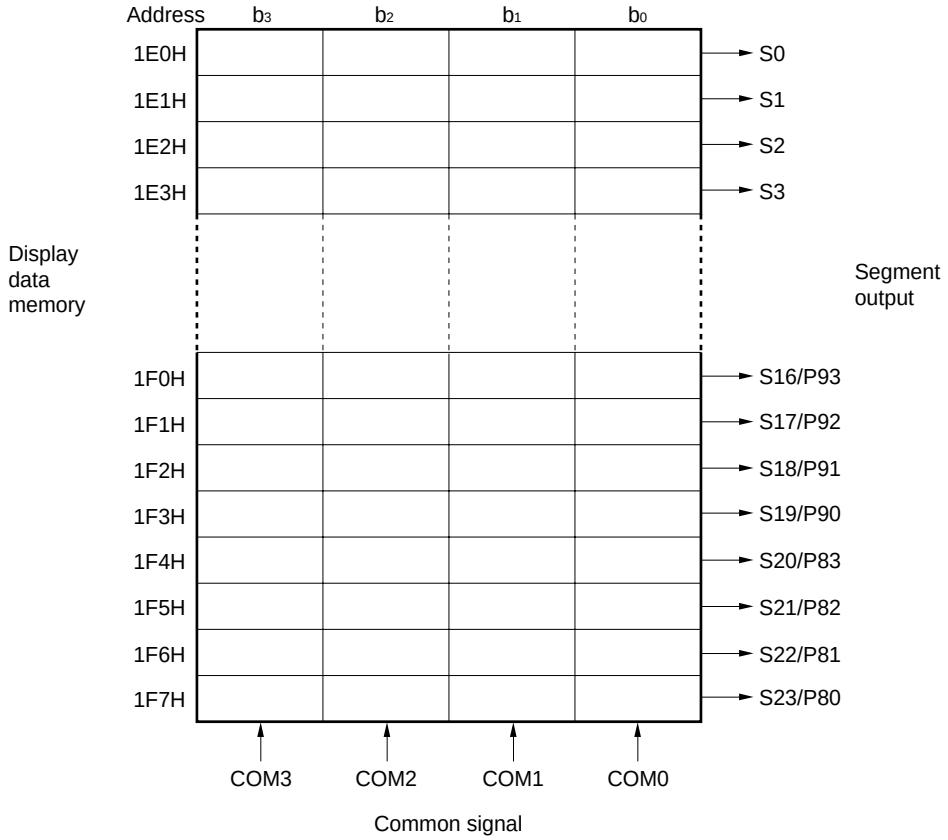
The contents of the data memory are undefined at reset. Therefore, they must be initialized at the beginning of program execution (RAM clear). Otherwise, unexpected bugs may occur.

Example To clear RAM at addresses 000H through 1FFH

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

★ **Example** To clear display data memory at addresses 1E0H through 1F7H

```
        SET1    MBE
        SEL     MB1
        MOV     HL, #0E0H
        MOV     A, #00H
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× 4 bits × 4 banks

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

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

RB = RBE • RBS (RBS = 0 to 3)

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

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

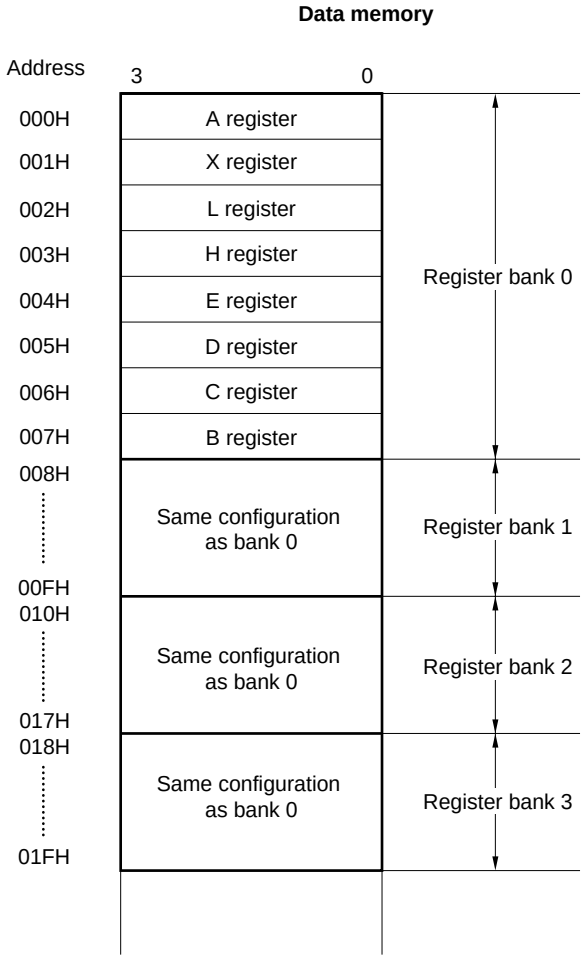
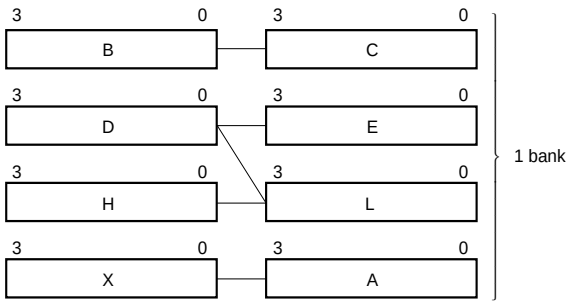


Figure 4-7. Register Pair Configuration

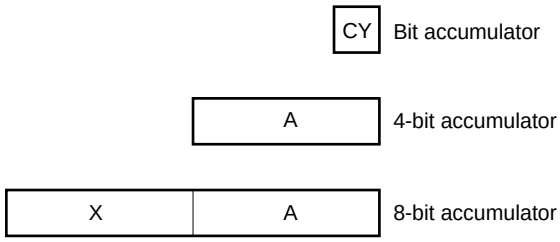


4.6 Accumulators

The μPD753108 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 μPD753108 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 to 1FFH of memory bank 0 or 1. Specify one memory bank using 2-bit SBS (See **Table 4-2**).

Table 4-2. Stack Area Selected by SBS

SBS		Stack Area
SBS1	SBS0	
0	0	Memory bank 0
0	1	Memory bank 1
Other than above		Setting prohibited

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

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

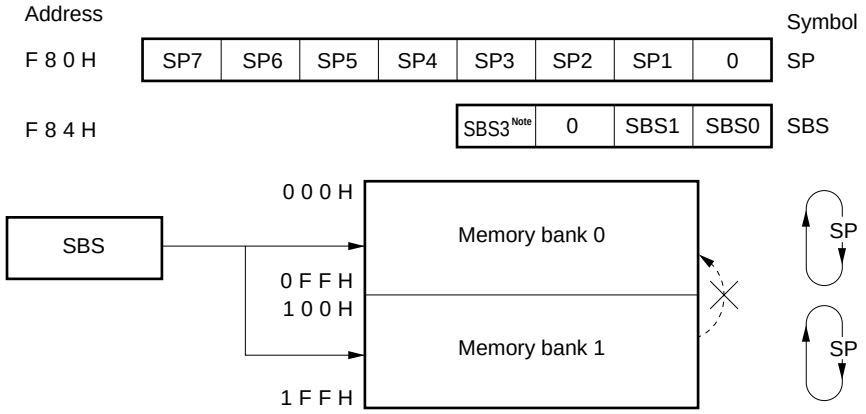
The initial value of the SP is set by an 8-bit memory manipulation instruction and the initial value of the SBS is set by a 4-bit memory manipulation instruction to determine a stack area. Its contents can also be read.

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

The stack area is limited to the memory bank specified by the SBS, and data is returned to nFFH in the same bank when further stacking operation is performed in addresses starting with n00H. Data cannot be stacked over the boundary of memory bank without rewriting the SBS.

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

Figure 4-9. 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 1 is assigned to the stack area and data is stacked in addresses starting with 1FFH.

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

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

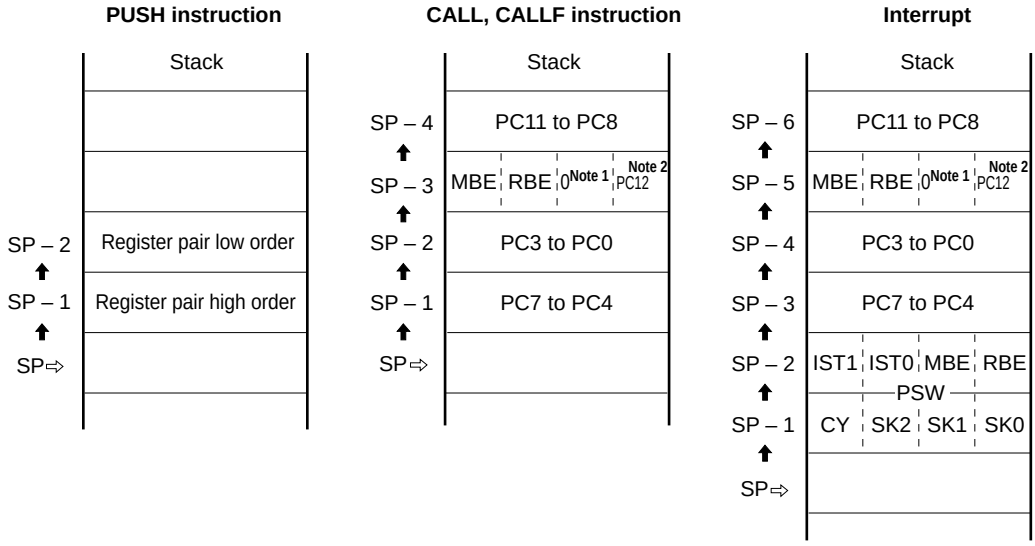
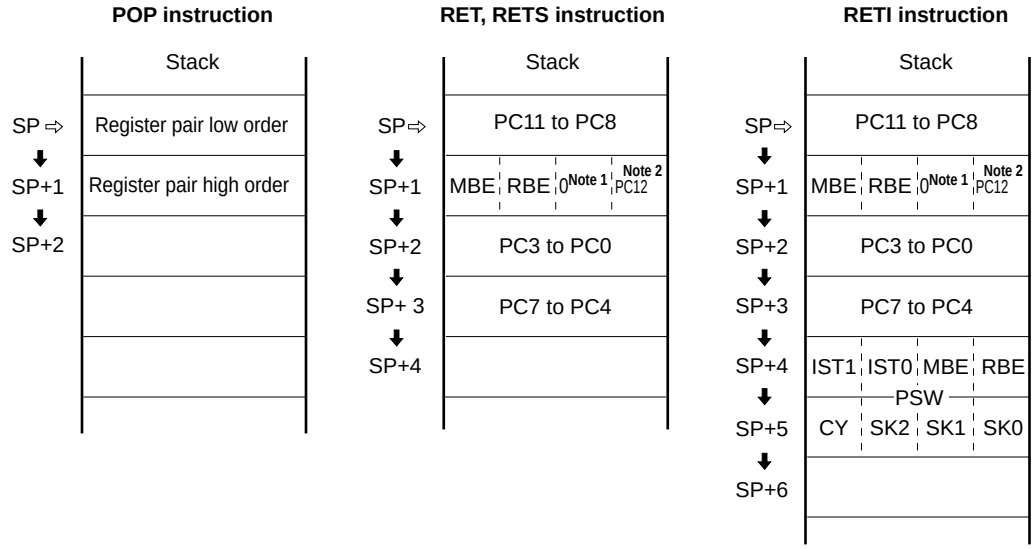


Figure 4-11. Data Restored from Stack Memory (Mk I mode)



Notes 1. For the μPD75P3116, PC13 is saved to this position.

2. For the μPD753104, PC12 is set to 0.

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

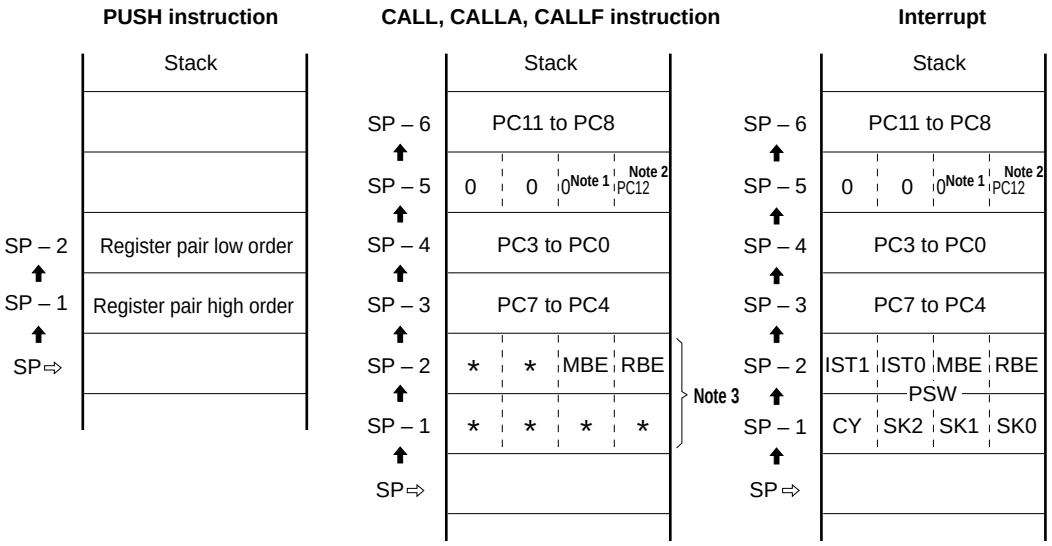
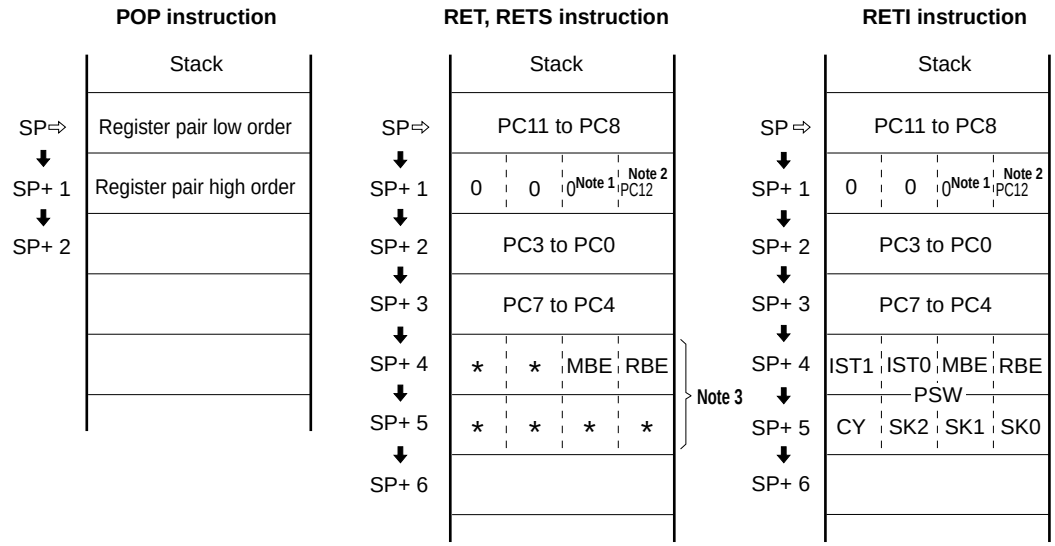


Figure 4-13. Data Restored from Stack Memory (Mk II mode)



Notes 1. For the μPD75P3116, PC13 is saved to this position.

2. For the μPD753104, PC12 is set to 0.

3. 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 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 Configuration

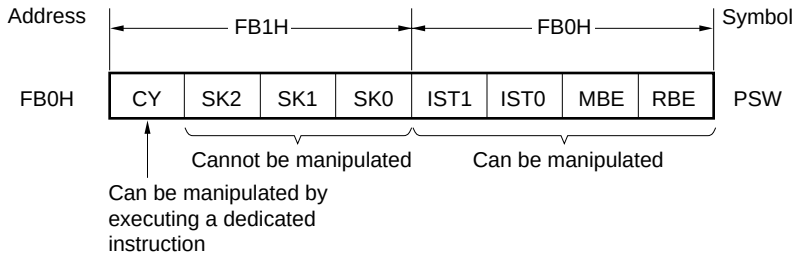


Table 4-3. PSW Flags Saved and Restored during Stack Operation

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

(1) Carry flag (CY)

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

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

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

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

Table 4-4. Carry Flag Manipulation Instructions

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

Remark mem*.bit indicates following three bit manipulation addressing

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

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

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

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

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

(3) Interrupt status flag (IST1, IST0)

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

Table 4-5. Interrupt Status Flag Indication

IST1	IST0	Status of Processing Being Performed	Processing Indication and Interrupt Control
0	0	Status 0	During normal program processing. Acknowledgment of all interrupts is enabled.
0	1	Status 1	During low-priority or high-priority interrupt processing. High-priority interrupt acknowledgment is enabled.
1	0	Status 2	During high-priority interrupt processing. Acknowledgment of all interrupt is disabled.
1	1	—	Setting prohibited

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

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

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

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

(4) Memory bank enable flag (MBE)

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

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

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

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

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

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

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

(5) Register bank enable flag (RBE)

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

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

When RBE is set to 1, general-purpose registers of one bank can be selected among register banks 0 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 bank 0 and 1 for 8-bit operation.

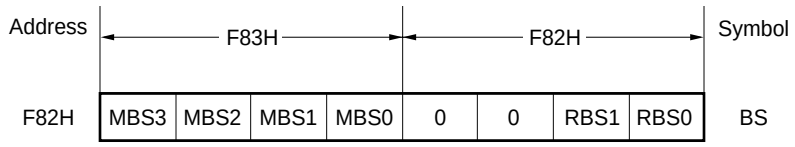
4.9 Bank Selection Register (BS)

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

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

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

Figure 4-15. Bank Selection Register Configuration



(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 μPD753108, 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.

(2) Register bank selection register (RBS)

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

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

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

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

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

Fixed to 0

× : Don't care

[MEMO]

CHAPTER 5 PERIPHERAL HARDWARE FUNCTION

5.1 Digital I/O Port

Memory mapped I/O is employed for the μPD753108. All the I/O ports are mapped in the data memory space.

Figure 5-1. Digital Ports Data Memory Addresses

Address	3	2	1	0	
F F 0 H	P03	P02	P01	P00	PORT0
F F 1 H	P13	P12	P11	P10	PORT1
F F 2 H	P23	P22	P21	P20	PORT2
F F 3 H	P33	P32	P31	P30	PORT3
F F 5 H	P53	P52	P51	P50	PORT5
F F 6 H	P63	P62	P61	P60	PORT6
F F 8 H	P83	P82	P81	P80	PORT8
F F 9 H	P93	P92	P91	P90	PORT9

Table 5-2 lists the input/output ports manipulation instructions for port 8 and port 9 in addition to 4-bit input/output, 8-bit input/output and bit manipulation can be performed. These enable many types of control.

Examples 1. The status shown on P13 is tested and the values depending on the results of test are output to ports 8 and 9.

```
SKT     PORT1.3      ; Skip if bit 3 of port 1 is 1.
MOV     XA, #18H     ; XA ← 18H
MOV     XA, #14H     ; XA ← 14H  String effect
SEL     MB15         ; or CLR1 MBE
OUT     PORT8, XA    ; ports 9, 8 ← XA
```

2. SET1 PORT8. @L ; The bit (in ports 8 and 9) specified by the L register is set to 1.

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

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

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

Table 5-1. Types and Features of Digital Ports

Port (Pin Name)	Function	Operation and Features	Remarks
PORT0 (P00 to P03)	4-bit input	The alternate function pins function as outputs depending on the operation mode when the serial interface function is used.	Also used for the INT4, $\overline{SCK}$, SO/SB0, and SI/SB1 pins.
PORT1 (P10 to P13)		Dedicated 4-bit input port.	Also used for the INT0 to INT2 and TI0 to TI2 pins.
PORT2 (P20 to P23)	4-bit I/O	Can be set to input mode or output mode in 4-bit units.	Also used for the PTO0 to PTO2, PCL, and BUZ pins.
PORT3 (P30 to P33)		Can be set to input mode or output mode in 1-bit units.	Also used for the LCDCL, SYNC, and MD0 to MD3**Note 1** pins.
PORT5 (P50 to P53)	4-bit I/O (N-channel open-drain, 13 V DC rating)	Can be set to input mode or output mode in 4-bit units. Pull-up resistor can be connected in 1-bit units by mask option**Note 2**.	Also used for the D4 to D7**Note 1** pins.
PORT6 (P60 to P63)	4-bit I/O	Can be set to input mode or output mode in 1-bit units.	Also used for the KR0 to KR3 and D0 to D3**Note 1** pins.
PORT8 (P80 to P83)		Can be set to input mode or output mode in 4-bit units. By combining, data can be input or output in 8-bit units.	Also used for the S20 to S23 pins.
PORT9 (P90 to P93)			Also used for the S16 to S19 pins.

Notes 1. Alternate function pins only in the μPD75P3116.

2. Pull-up resistor cannot be connected by mask option in the μPD75P3116.

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

Figure 5-2. Ports 0, 1 Configuration

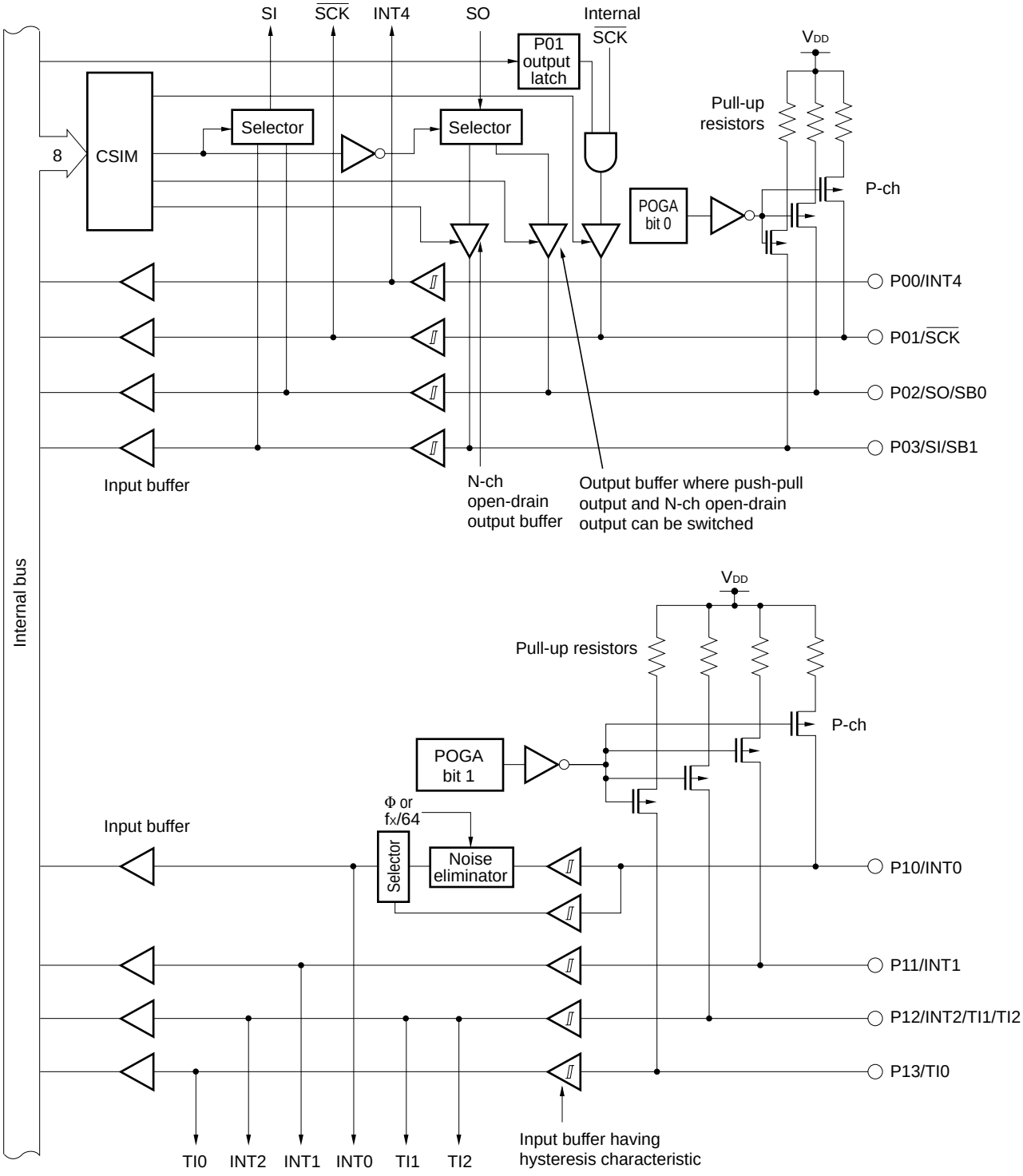


Figure 5-3. Ports 3, 6 Configuration

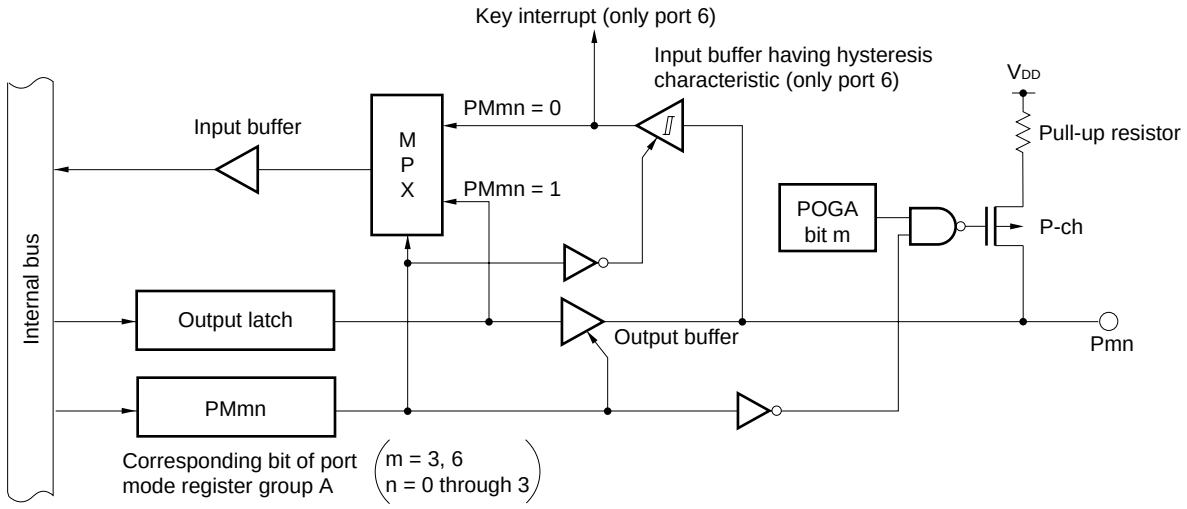


Figure 5-4. Port 2 Configuration

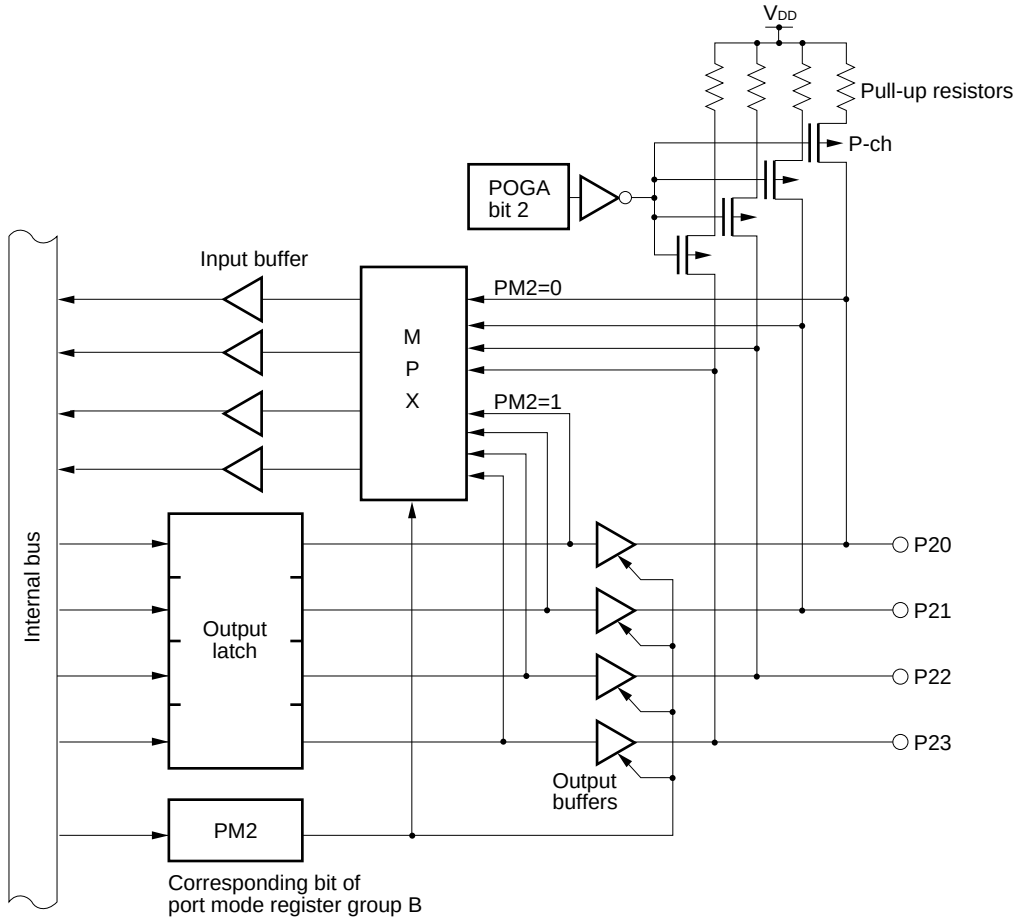


Figure 5-5. Port 5 Configuration

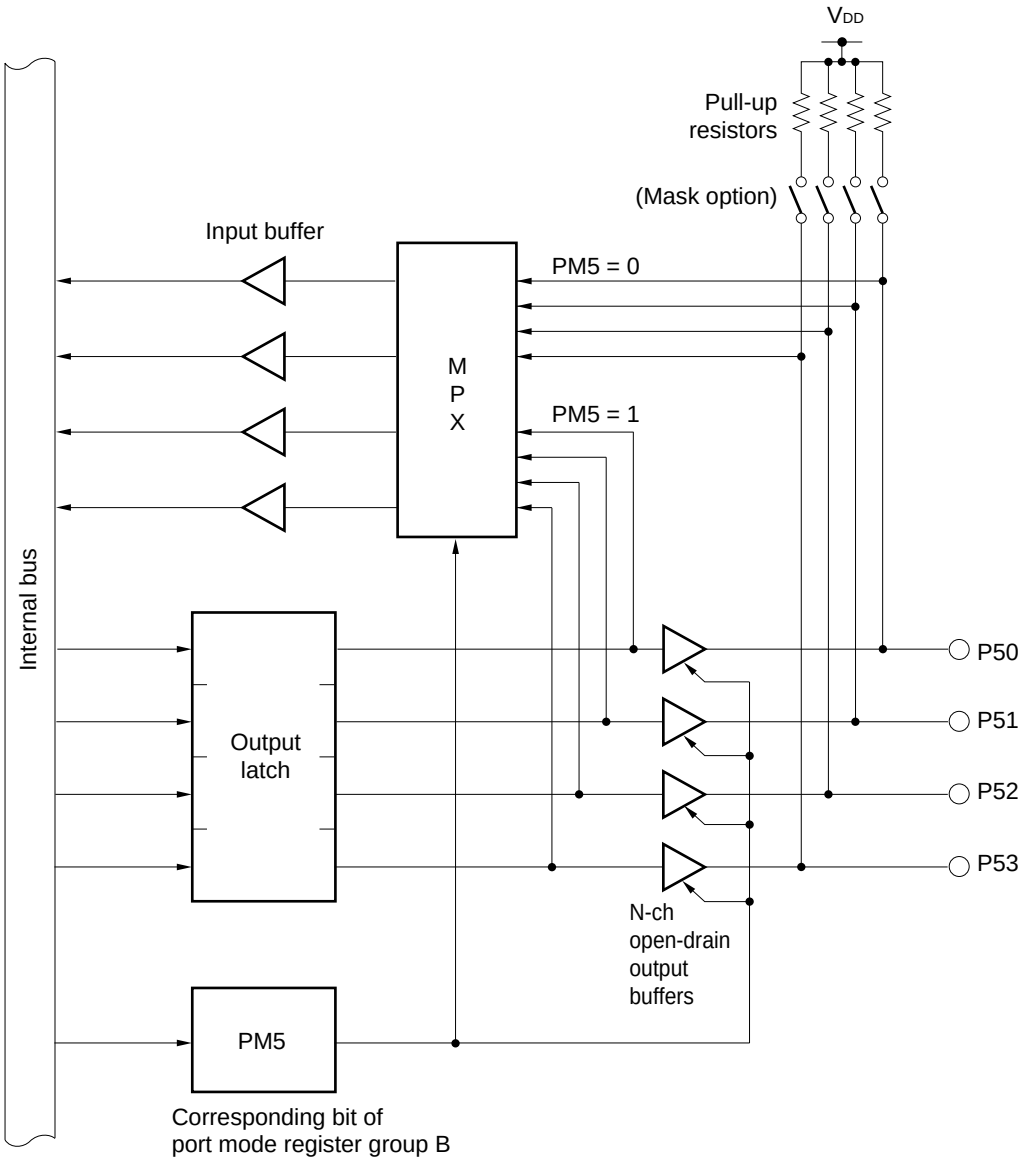
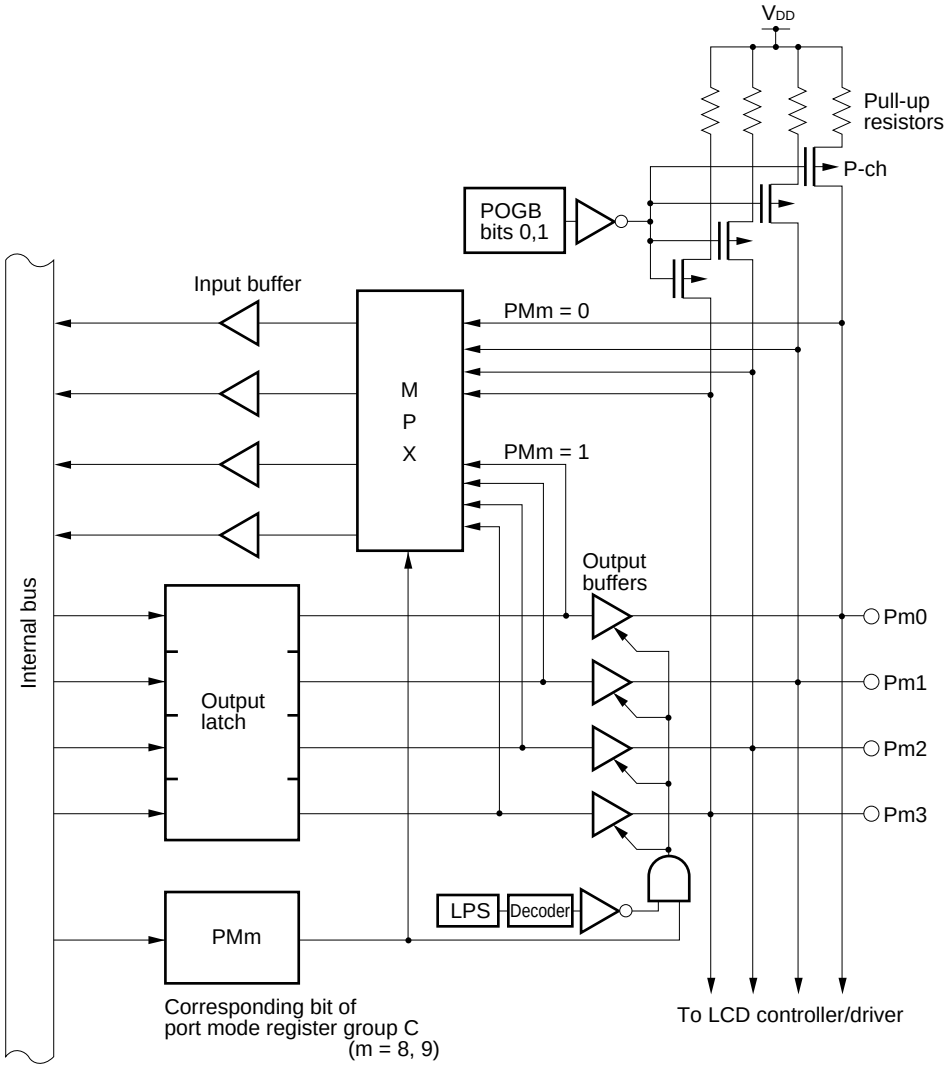


Figure 5-6. Ports 8, 9 Configuration



5.1.2 Setting I/O mode

The input or output mode of each I/O port is set by the corresponding port mode register as shown in Figure 5-7. Ports 3 and 6 can be set in the input or output mode in 1-bit units by using port mode register group A (PMGA). Ports 2 and 5 are set in the input or output mode in 4-bit units by using port mode register group B (PMGB). Port mode group register group C (PMGC) is used to set the input or output mode of ports 8 and 9 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, B, and C are set by using an 8-bit memory manipulation instruction.

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

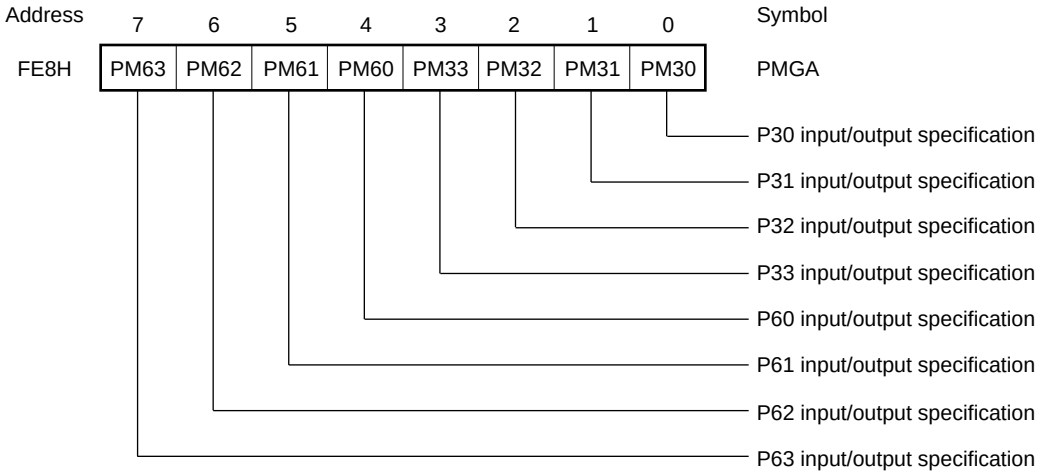
Example To use P30, 31, 62, and 63 as input pins and P32, 33, 60, and 61 as output pins

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

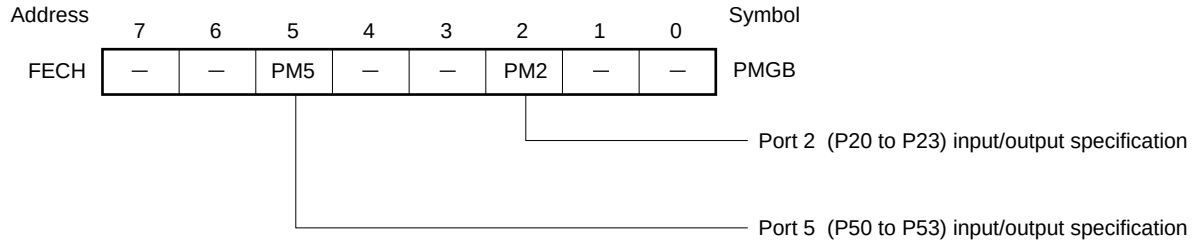
Figure 5-7. Port Mode Register Formats

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

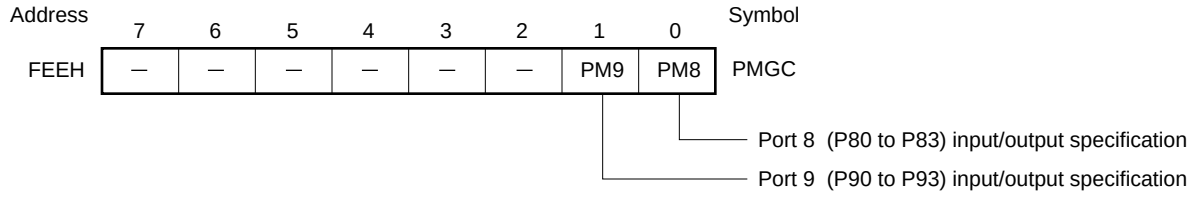
Port mode register group A



Port mode register group B



Port mode register group C



5.1.3 Digital I/O port manipulation instruction

Because all the I/O ports of the μPD753108 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 0 through 3, 5, 6, 8, and 9, the bits of these ports can be manipulated regardless of the specifications by MBE and MBS.

Example To OR P50 and P51 and set P61 in output mode

```
MOV1    CY, PORT5.0     ; CY ← P50
OR1     CY, PORT5.1     ; CY ← CY V P51
MOV1    PORT6.1, CY     ; P61 ← 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 5

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

3. To test whether the data of port 5 is greater than the value of the accumulator

```
SET1    MBE
SEL     MB15
MOV     HL, #PORT5
SUBS    A, @HL          ; A < PORT5
BR      NO              ; NO
                        ; YES
```

(3) 8-bit manipulation instruction

In addition to the IN and OUT instructions, the MOV, XCH, and SKE instructions can be used to manipulate ports 8 and 9, which can be manipulated in 8-bit units. In this case also, memory bank 15 must be selected, just as when 4-bit manipulation instructions are used to manipulate ports.

Example To output the data of register pair BC to the output port specified by the 8-bit data input from ports 8 and 9

```
SET1    MBE
SEL     MB15
IN      XA, PORT8       ; XA ← ports 9 and 8
MOV     HL, XA          ; HL ← XA
MOV     XA, BC          ; XA ← BC
MOV     @HL, XA         ; Port (L) ← XA
```

★ **Table 5-2. I/O Pin Manipulation Instructions**

Instruction \ PORT	PORT 0	PORT 1	PORT 2	PORT 3	PORT 5	PORT 6	PORT 8	PORT 9
IN A, PORTn **Note 1**				√				
IN XA, PORTn **Note 1**	—		—		—		√	
OUT PORTn, A **Note 1**	—		√					
OUT PORTn, XA **Note 1**	—		—		—		√	
MOV A, PORTn **Note 1**				√				
MOV XA, PORTn **Note 1**	—		—		—		√	
MOV PORTn, A **Note 1**				√				
MOV PORTn, XA **Note 1**	—		—		—		√	
XCH A, PORTn **Note 1**				√				
XCH XA, 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, 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- or 8-bit units, such as OUT or MOV, is executed, the data of the accumulator is latched to the output latch. The output buffer remains off.

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

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

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

(2) Operation in output mode

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

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

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

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

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

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

Instruction Executed	Operation of Port and Pins	
	Input Mode	Output Mode
SKT ① SKF ①	Tests pin data	Tests output latch data
MOV1 CY, ①	Transfers pin data to CY	Transfers output latch data to CY
AND1 CY, ① OR1 CY, ① XOR1 CY, ①	Performs operation between pin data and CY	Performs operation between output latch data and CY

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

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

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

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

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

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

5.1.5 Connecting pull-up resistors

Each port pin of the μPD753108 can be connected to an internal pull-up resistor (except the P00 pin). Some pins can be connected with a pull-up resistor by software and the others can be connected by mask option.

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

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

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

Port (Pin Name)	Pull-up Resistor Specification Method	Specified Bit
PORT0 (P01 to P03)Note 1	Specifies to connect in 3-bit units by software.	POGA.0
PORT1 (P10 to P13)	Specifies to connect in 4-bit units by software.	POGA.1
PORT2 (P20 to P23)		POGA.2
PORT3 (P30 to P33)		POGA.3
PORT5 (P50 to P53)	Specifiable by mask option in 1-bit units.	—
PORT6 (P60 to P63)	Specifies to connect in 4-bit units by software.	POGA.6
PORT8 (P80 to P83)Note 2		POGB.0
PORT9 (P90 to P93)Note 2		POGB.1

Notes 1. The P00 pin cannot be specified to connect an on-chip pull-up resistor.

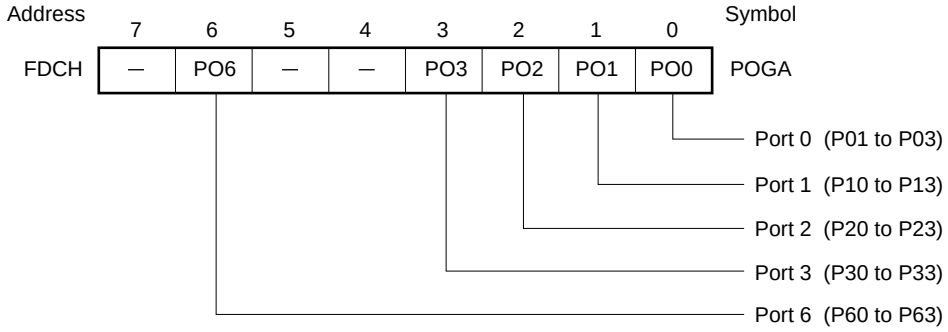
2. If these pins are used as segment outputs, do not specify to connect the on-chip pull-up resistor by software.

Remark Pull-up resistor cannot be connected by mask option in the μPD75P3116.

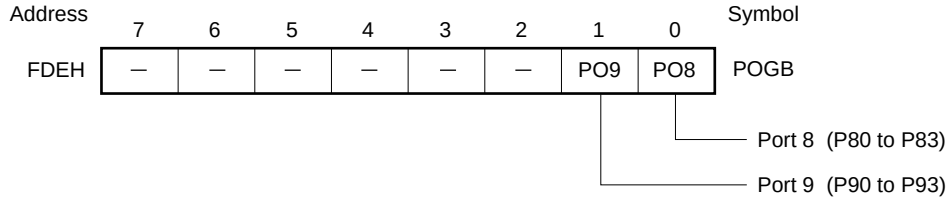
Figure 5-8. Pull-Up Resistor Specify Register Formats

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

Pull-up resistor specify register group A



Pull-up resistor specify register group B



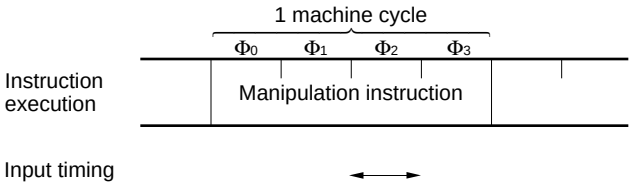
5.1.6 I/O timing of digital I/O port

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

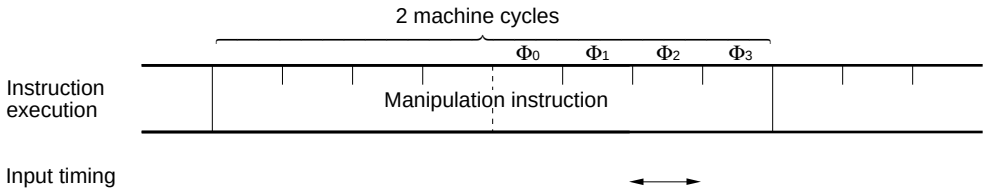
Figure 5-10 shows the ON timing when an on-chip pull-up resistor is connected to a port pin via software.

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

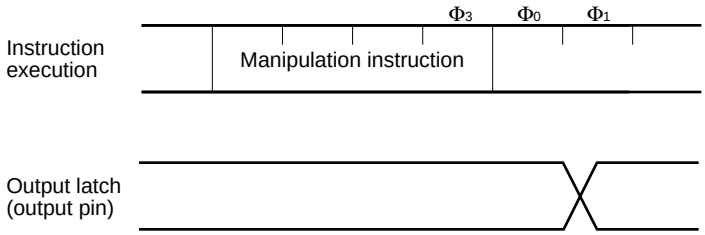
★ **(a) When data is loaded by 1-machine cycle instruction**



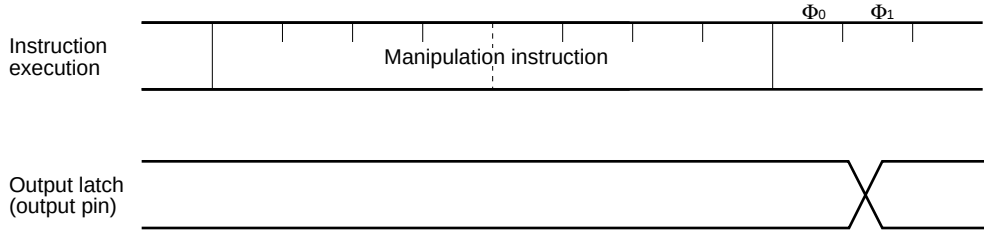
★ **(b) When data is loaded by 2-machine cycle instruction**



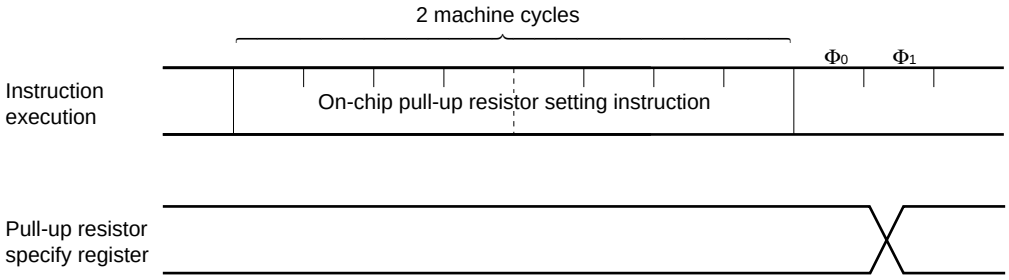
(c) When data is latched by 1-machine cycle instruction



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



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



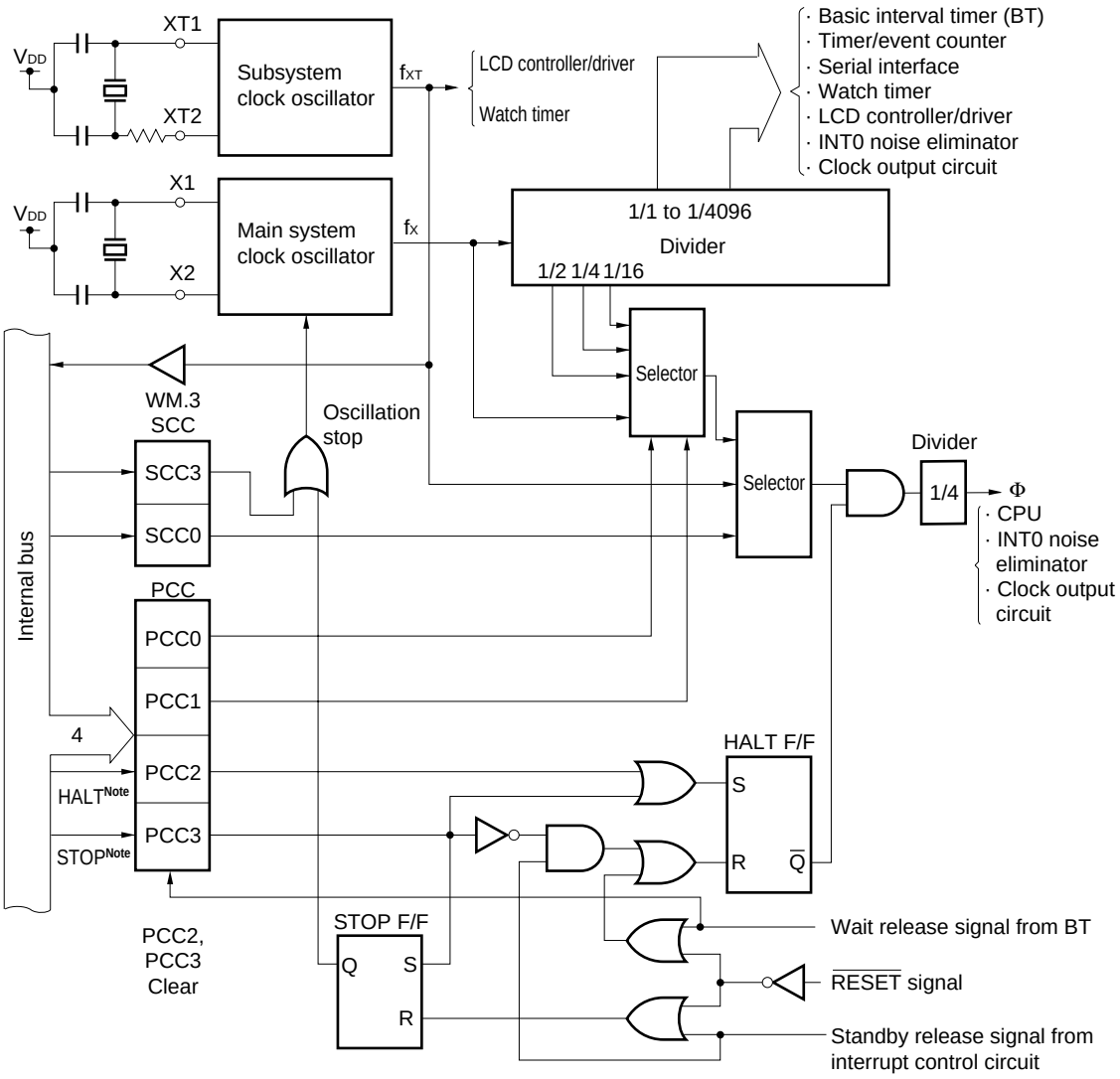
5.2 Clock Generator

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

5.2.1 Clock generator configuration

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

Figure 5-11. Clock Generator Block Diagram



Note Instruction execution

Remarks 1. f_X = Main system clock frequency

2. f_{XT} = Subsystem clock frequency

3. Φ = CPU clock

4. PCC: Processor Clock Control Register

5. SCC: System Clock Control Register

6. 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 mode of the CPU such as standby mode.

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

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

(a) When the $\overline{\text{RESET}}$ signal is generated, the minimum speed mode of the main system clock (10.7 μs at 6.00-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 (0.67 μs, 1.33 μs, 2.67 μs, and 10.7 μs at 6.00-MHz operation) by setting PCC.

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

(d) Subsystem clock is selected by setting SCC, and operation can be performed at a very low-speed and low current consumption (122 μs at 32.768-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. The HALT mode can also be used, but the STOP mode cannot be used (Subsystem clock oscillation cannot 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 LCD controller/driver and buzzer output function which operate using the watch timer clocks can also continue operation in the standby mode.

(g) When the subsystem clock is selected, the watch timer and LCD controller/driver can continue normal operation. The serial interface and timer/event counter can continue the operation when they have selected an external clock as a clock. However, the other hardware operate with the 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 5-12**).

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

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

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

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

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 $f_{XT}/4$. The STOP instruction can be executed only when the PCC operates with the main system clock.

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

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

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

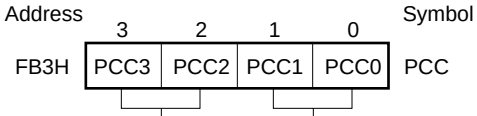
```
SEL     MB15
MOV     A, #0010B
MOV     PCC, A
```

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

```
STOP
NOP
```

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

Figure 5-12. Processor Clock Control Register Format



CPU clock selection bit

(When f_X = 6.0 MHz)

		SCC3, SCC0 = 00 Values in Parentheses Are Applied When f_X = 6.0 MHz		SCC3, SCC0 = 01 or 11 Values in Parentheses Are Applied When f_{XT} = 32.768 kHz	
		CPU Clock Frequency	1 Machine Cycle	CPU Clock Frequency	1 Machine Cycle
0	0	$\Phi = f_X/64$ (93.8 kHz)	10.7 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
0	1	$\Phi = f_X/16$ (375 kHz)	2.67 μs		
1	0	$\Phi = f_X/8$ (750 kHz)	1.33 μs		
1	1	$\Phi = f_X/4$ (1.5 MHz)	0.67 μs		

(When f_X = 4.19 MHz)

		SCC3, SCC0 = 00 Values in Parentheses Are Applied When f_X = 4.19 MHz		SCC3, SCC0 = 01 or 11 Values in Parentheses Are Applied When f_{XT} = 32.768 kHz	
		CPU Clock Frequency	1 Machine Cycle	CPU Clock Frequency	1 Machine Cycle
0	0	$\Phi = f_X/64$ (65.5 kHz)	15.3 μs	$\Phi = f_{XT}/4$ (8.192 kHz)	122 μs
0	1	$\Phi = f_X/16$ (262 kHz)	3.81 μs		
1	0	$\Phi = f_X/8$ (524 kHz)	1.91 μs		
1	1	$\Phi = f_X/4$ (1.05 MHz)	0.95 μs		

Remarks 1. f_X : Main system clock oscillator output frequency

2. f_{XT} : 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)

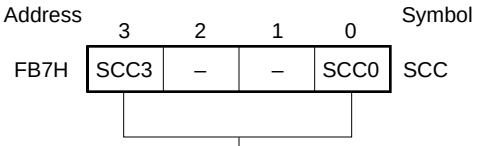
The system clock register (SCC) is a 4-bit register whose least significant bit is used to select CPU clock Φ, and the most significant bit is used to control (stop) main system clock oscillation (See **Figure 5-13**).

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

Main system clock oscillation can be stopped by setting SCC.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 5-13. 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_{XT} time. To stop main system clock 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.

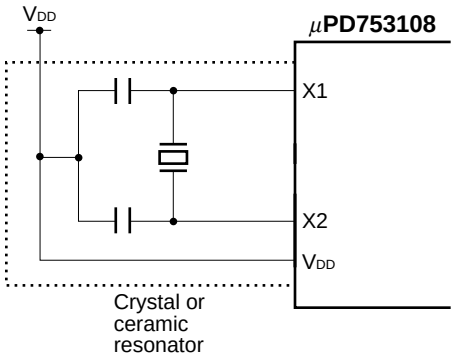
★ **3. When "1" is set to SCC.3, the X2 pin is internally pulled up to V_{DD} with the resistor of 50 kΩ (TYP.).**

(3) System clock oscillators

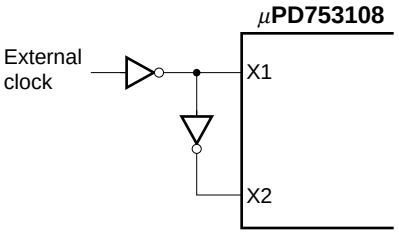
The main system clock oscillator oscillates with a crystal resonator or ceramic resonator connected to the X1 and X2 pins (4.194304 MHz TYP.). External clock can also be input.

Figure 5-14. Main System Clock Oscillator External Circuit

(a) Crystal or ceramic oscillation



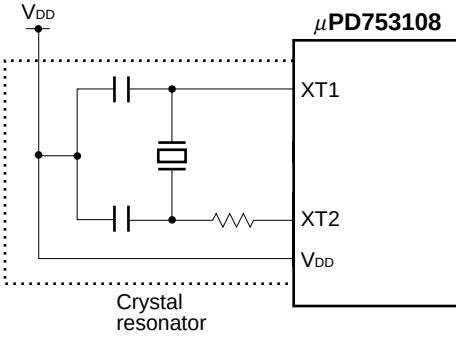
(b) External clock



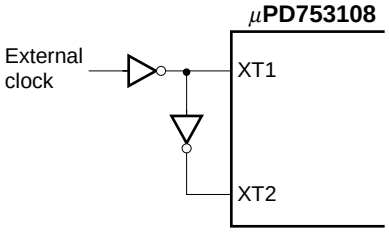
The subsystem clock oscillator oscillates with a crystal resonator connected to the XT1 and XT2 pins (32.768 kHz TYP.). External clock can also be input.

Figure 5-15. Subsystem Clock Oscillator External Circuit

(a) Crystal oscillation



★(b) External clock



★ **Cautions 1. When the STOP mode is set, the X2 pin is internally pulled up to V_{DD} with the resistor of 50 kΩ (TYP.).**

2. Wire the portion enclosed by broken lines in Figures 5-14 and 5-15 as follows to prevent adverse influence by wiring capacitance when using the main system clock and subsystem clock oscillation circuits.

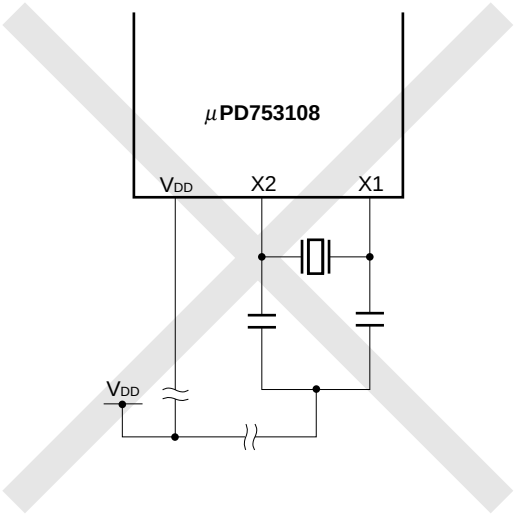
- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any other signal lines.**
- **Do not route the wiring in the vicinity of a line through which a high alternating current flows.**
- **Always keep the potential at the connecting point of the capacitor of the oscillation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

The amplification factor of the subsystem clock oscillation circuit is kept low to reduce the power dissipation, and therefore it is more susceptible to noise than the main system clock oscillation circuit. To use the subsystem clock oscillation circuit, therefore, exercise care in wiring.

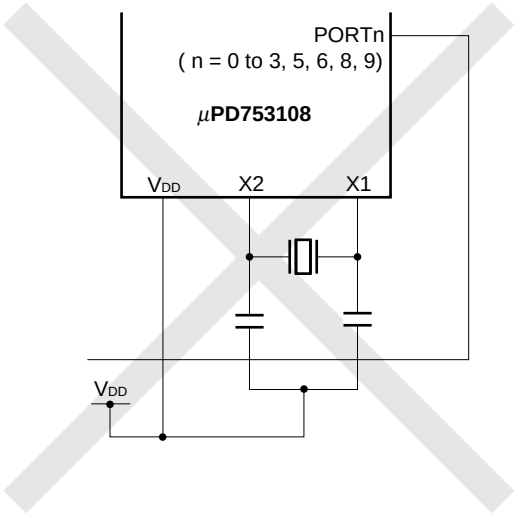
Figure 5-16 shows examples of connecting the resonator incorrectly.

Figure 5-16. Example of Connecting Resonator Incorrectly (1/2)

(a) Wiring length too long



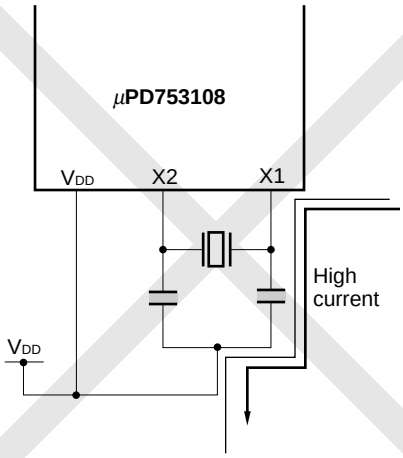
(b) Crossed signal line



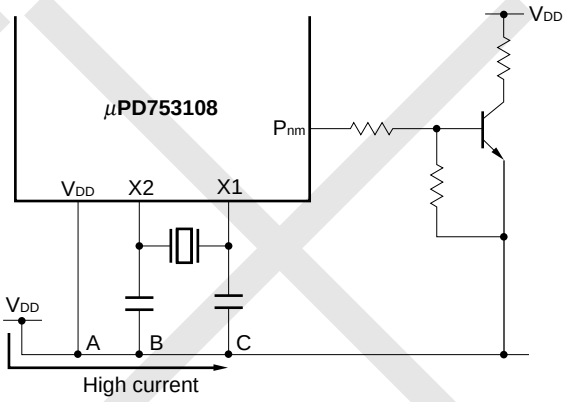
Remark When using the subsystem clock, take X1 and X2 in the above figures as XT1 and XT2. Also, connect a resistor in series with XT2.

Figure 5-16. Example of Connecting Resonator Incorrectly (2/2)

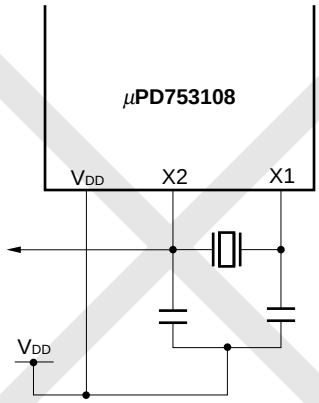
(c) High alternating current close to signal line



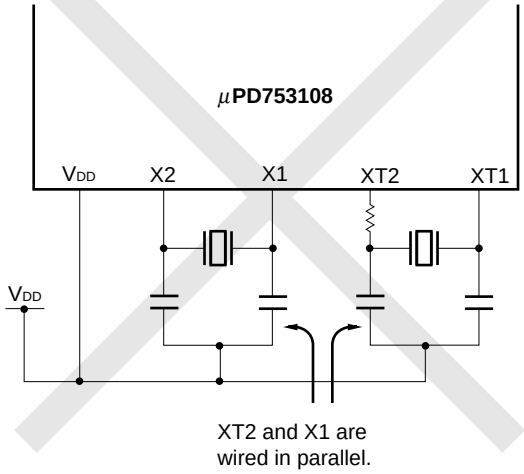
(d) Current flowing through power line of oscillation circuit (potential at points A, B, and C changes)



(e) Signal extracted

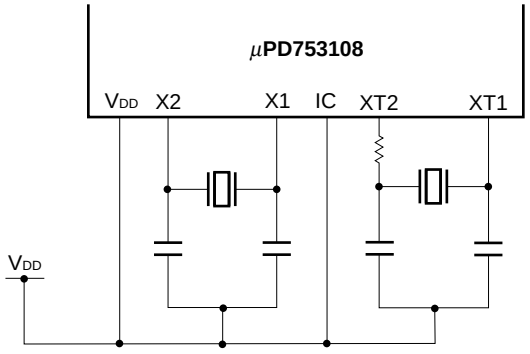


(f) Main system clock and subsystem clock signal lines close and parallel to each other



Remark When using the subsystem clock, take X1 and X2 in the above figures as XT1 and XT2. Also, connect a resistor in series with XT2.

Cautions 3. In Figure 5-16 (f), XT2 and X1 are wired in parallel. In consequence, the crosstalk noise of X1 may be superimposed on XT2, causing malfunctioning.
To avoid the malfunctioning, do not wire XT2 and X1 in parallel, and connect the IC pin between XT2 and X1 pins to V_{DD}.



(4) Divider circuit

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

(5) Subsystem clock oscillator control functions

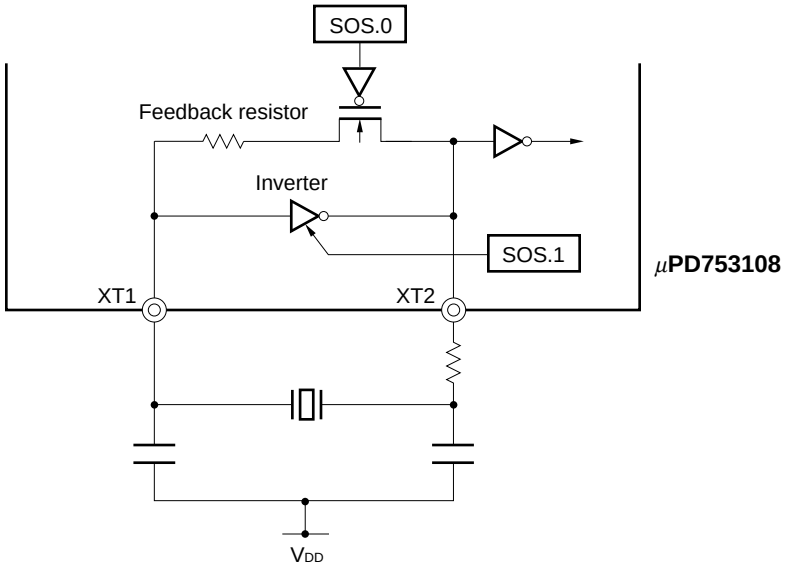
The μPD753108 subsystem clock oscillator has the following two control functions.

- Selects by software whether an on-chip feedback resistor is to be used or not[Note].
- Reduces current consumption by decreasing the drive current of the on-chip inverter when the supply voltage is high ($V_{DD} \geq 2.7$ V).

★ **Note** When not using the subsystem clock, set SOS.0 to 1 (disable the on-chip feedback resistor) via software, connect XT1 to V_{SS} or V_{DD}, and leave XT2 open. This reduces the current consumption in the subsystem clock oscillator.

The above functions can be used by switching the bits 0 and 1 of the subsystem clock oscillator control register (SOS) (See **Figure 5-17**).

Figure 5-17. Subsystem Clock Oscillator



(6) Subsystem clock oscillator control register (SOS)

The SOS selects whether to use the on-chip feedback resistor or not and controls the drive current of the on-chip inverter (See **Figure 5-18**).

When the $\overline{\text{RESET}}$ signal is asserted, all the bits of this register are cleared to 0. The function of each flag of the SOS register is described below.

★ **(a) SOS.0 (feedback resistor cut flag)**

Whether or not to use the on-chip feedback resistor can be selected by software by setting SOS.0 in the μPD753108**.**

If no resonator is used, the current consumption can be reduced by setting SOS.0 to "1" to turn off the feedback circuit.

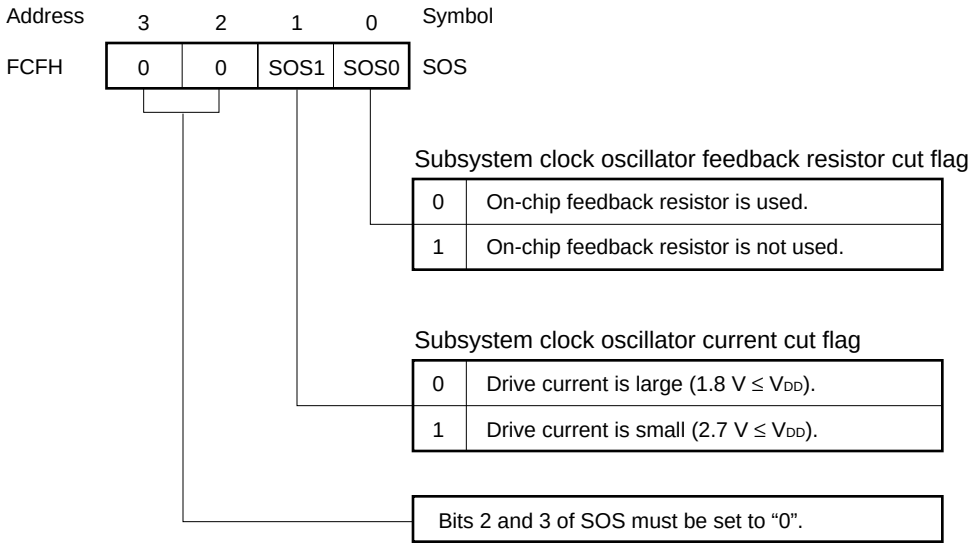
If a resonator is used, be sure to set "0" (feedback circuit turned on).

(b) SOS.1 (drive capability switching flag)

The on-chip inverter of the μPD753108's subsystem clock oscillator is designed to operate with a low supply voltage (V_{DD} = 1.8 V possible) and so its drive current is large. When it is used with a high supply voltage ($V_{DD} \geq$ 2.7 V), its supply current becomes too large. In this case, the supply current can be decreased by reducing the drive current of the inverter by setting the SOS.1 to "1".

Note if it is set to "1" at the time V_{DD} is less than 2.7 V, the oscillator may stop oscillation due to too small a drive current. When V_{DD} of less than 2.7 V is used, SOS.1 must be set to "0".

Figure 5-18. Subsystem Clock Oscillator Control Register (SOS) Format



Remark When the subsystem clock is not necessary, set the XT1 and XT2 pins and SOS register as follows:

XT1 : Connect to V_{SS} or V_{DD}
XT2 : Open
SOS : 00X1B (X: don’t care)

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 or set bit 3 of SCC after switching the 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	×	×
0	0	0				1 machine cycle			1 machine cycle			1 machine cycle			$\frac{f_X}{64f_{XT}}$ machine cycles (3 machine cycles)		
	0	1	4 machine cycles						4 machine cycles			4 machine cycles			$\frac{f_X}{16f_{XT}}$ machine cycles (12 machine cycles)		
	1	0	8 machine cycles			8 machine cycles						8 machine cycles			$\frac{f_X}{8f_{XT}}$ machine cycles (23 machine cycles)		
	1	1	16 machine cycles			16 machine cycles			16 machine cycles						$\frac{f_X}{4f_{XT}}$ machine cycles (46 machine cycles)		
1	×	×	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_X and f_{XT} change depending on the environmental temperature of the resonator and the variance of load capacitance characteristics. When f_X is higher than its nominal value or f_{XT} is lower than its nominal value, the machine cycles obtained by the formulae $f_X/64f_{XT}$, $f_X/16f_{XT}$, $f_X/8f_{XT}$, and $f_X/4f_{XT}$ given in the table are larger than those obtained by the nominal values of f_X and f_{XT}.

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_X and f_{XT}.

Remarks 1. ( ): f_X = 6.0 MHz, f_{XT} = 32.768 kHz

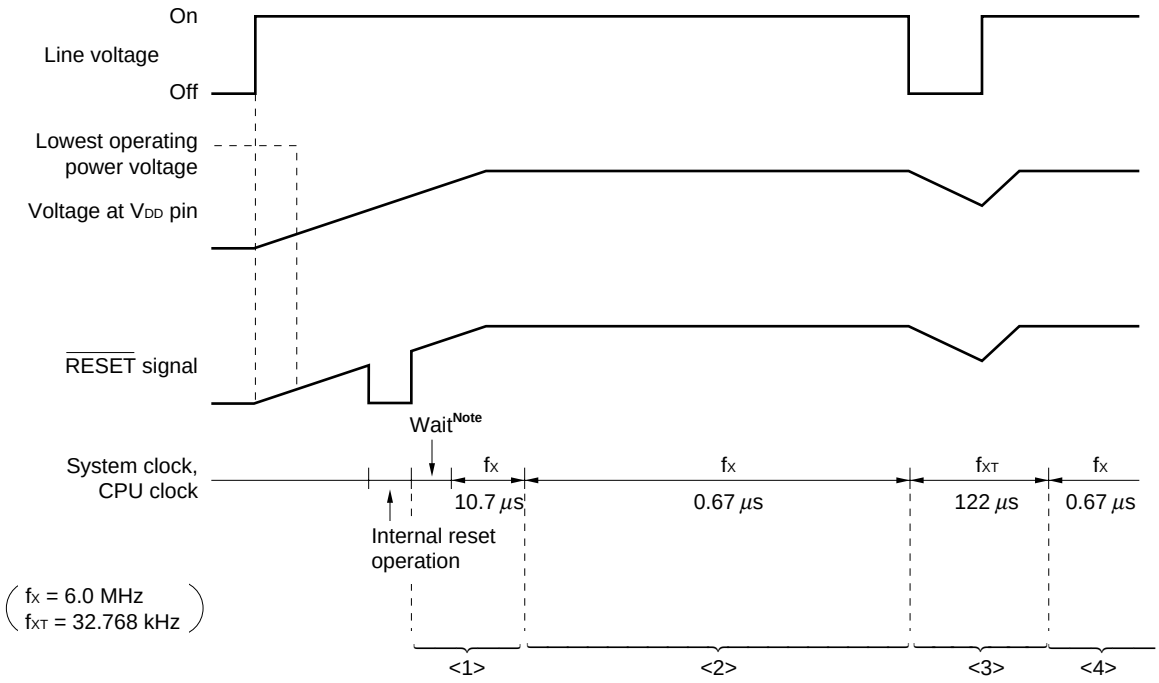
2. ×: don't care

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 5-19.

★ **Figure 5-19. 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 (10.7 μs: 6.0-MHz operation, 15.3 μs: 4.19-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 (INT4) 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 (46 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 following two wait times can be selected by mask option.

$2^{17}/f_X$ (21.8 ms: 6.0-MHz operation, 31.3 ms: 4.19-MHz operation)

$2^{15}/f_X$ (5.46 ms: 6.0-MHz operation, 7.81 ms: 4.19-MHz operation)

The μPD75P3116 provides no mask options and the wait time is fixed to $2^{15}/f_X$.

5.2.4 Clock output circuit

(1) Clock output circuit configuration

The configuration of the clock output circuit is shown in Figure 5-20.

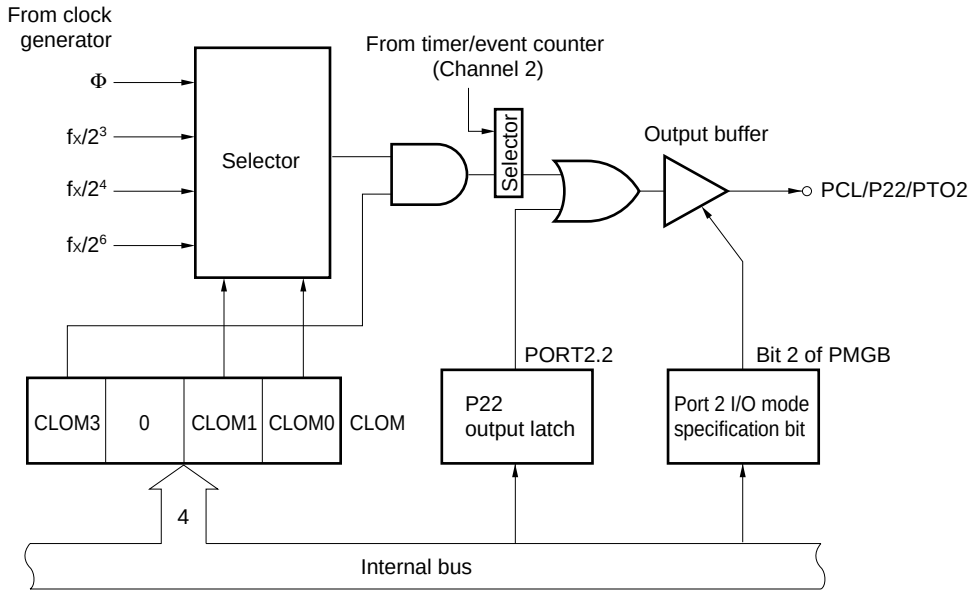
(2) Clock output circuit function

The clock output circuit is provided to output the clock pulses from the PCL/P22/PTO2 pin to the remote control waveform output application and peripheral LSI's.

The clock pulses must be output in the following steps.

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

Figure 5-20. Clock Output Circuit Block Diagram



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

(3) Clock output mode register (CLOM)

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

It must be set by a 4-bit memory manipulation instruction.

Example The CPU clock Φ is output from the PCL/P22/PTO2 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 5-21. Clock Output Mode Register Format

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

Clock output frequency select bit

When f_X = 6.00 MHz

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

When f_X = 4.19 MHz

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

Note Φ is the CPU clock selected by the PCC.

Clock output enable/disable bit

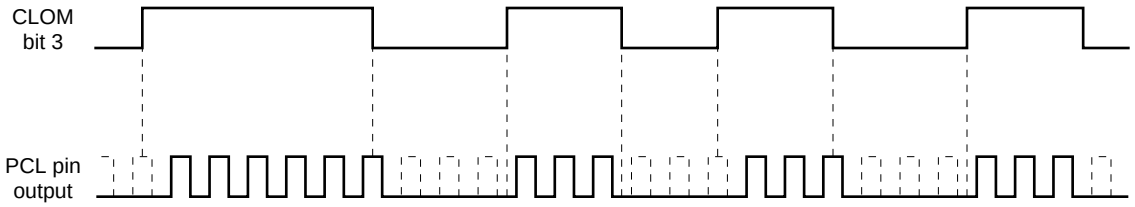
0	Output disabled.
1	Output enabled.

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

(4) Application example of remote control waveform output

The μPD753108 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 5-22. Application Example of Remote Control Waveform Output



5.3 Basic Interval Timer/Watchdog Timer

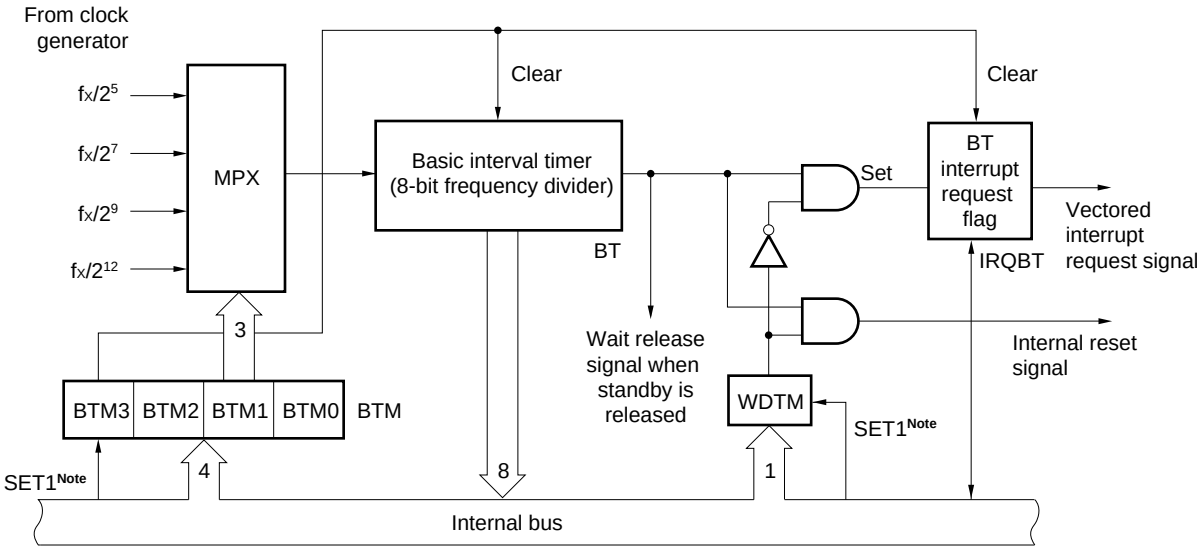
The μPD753108 is provided with the 8-bit basic interval timer/watchdog timer and has the following functions.

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

5.3.1 Basic interval timer/watchdog timer configuration

The configuration of the basic interval timer/watchdog timer is shown in Figure 5-23.

Figure 5-23. Basic Interval Timer/Watchdog Timer Block Diagram



Note Instruction execution

5.3.2 Basic interval timer mode register (BTM)

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

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

Bit 3 can be set by a bit manipulation instruction.

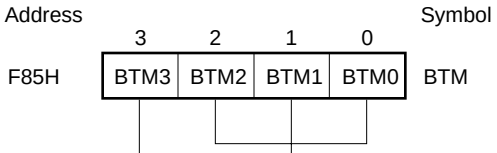
Example The interrupt generation interval is set to 1.37 ms (6.00 MHz).

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

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

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

Figure 5-24. Basic Interval Timer Mode Register Format



When f_X = 6.00 MHz

			Clock Input Specification	Interrupt Interval Time (Wait Time When Standby Is Released)
0	0	0	$f_X/2^{12}$ (1.46 kHz)	$2^{20}/f_X$ (175 ms)
0	1	1	$f_X/2^9$ (11.7 kHz)	$2^{17}/f_X$ (21.8 ms)
1	0	1	$f_X/2^7$ (46.9 kHz)	$2^{15}/f_X$ (5.46 ms)
1	1	1	$f_X/2^5$ (188 kHz)	$2^{13}/f_X$ (1.37 ms)
Other than the above			Setting prohibited	–

When f_X = 4.19 MHz

			Clock Input Specification	Interrupt Interval Time (Wait Time When Standby Is Released)
0	0	0	$f_X/2^{12}$ (1.02 kHz)	$2^{20}/f_X$ (250 ms)
0	1	1	$f_X/2^9$ (8.19 kHz)	$2^{17}/f_X$ (31.3 ms)
1	0	1	$f_X/2^7$ (32.768 kHz)	$2^{15}/f_X$ (7.81 ms)
1	1	1	$f_X/2^5$ (131 kHz)	$2^{13}/f_X$ (1.95 ms)
Other than the above			Setting prohibited	–

Start control bit of basic interval timer/watchdog timer

The basic interval timer/watchdog timer starts by writing “1”
(The counter and interrupt request flag are cleared).
Reset to "0" 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.

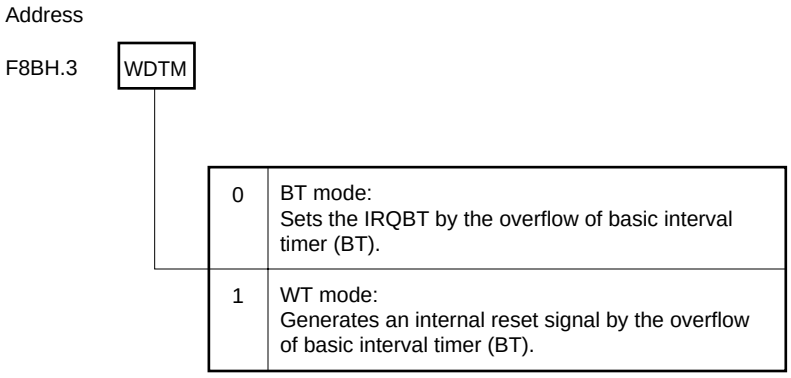
It is set by a bit manipulation instruction. Once it is set, it cannot be cleared by an instruction.

Example Setting of watchdog timer

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

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

Figure 5-25. 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 basic interval timer (BT) always increments by the clock sent from the clock generator and the counting operation cannot be stopped.

Four interrupt generation intervals can be set by BTM (See **Figure 5-24**).

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

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

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

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

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

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

5.3.5 Watchdog timer operations

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

In the watchdog timer mode, program runaway is detected by utilizing the interval time wherein the BT overflows. Four intervals can be selected by bits 0 to 2 of the BTM (See **Figure 5-24**). Select one of them suitable for user’s system. Set an interval and divide the program so that it can be executed in the interval and execute the instruction which clears the BT at the ends of the divided program. If the instruction which clears the BT is not reached within the time set (that is, the program is not executed normally = runaway), the BT overflows and an internal reset signal is generated to forcibly terminate the program. Namely, it indicates a program runaway has occurred and been detected.

Set the watchdog timer with the following procedure (<1> 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 5.46 ms (during 6.00-MHz operation).

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

Initialization :

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

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

Module 1 :

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

Processing is completed within 5.46 ms.

Module 2 :

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

Processing is completed within 5.46 ms.

5.3.6 Other functions

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

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

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

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

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

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

(2) Reads the counting operation

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

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

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

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

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

<INT4 interrupt routine (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     PORT0.0     ; P00 = 1?
        BR      AA          ; NO
        MOV     XA, BC      ; Stores data in the data memory
        MOV     BUFF, XA
        CLR1    FLAG        ; Data exists. Clears the flag.
        RETI
AA :    MOV     HL, #BUFF
        MOV     A, C
        SUBC    A, @HL
        INCS    L
        MOV     C, A
        MOV     A, B
        SUBC    A, @HL
        MOV     B, A
        MOV     XA, BC
        MOV     BUFF, XA    ; Stores data
        SET1    FLAG        ; Data exists. Sets the flag.
        RETI
```

5.4 Watch Timer

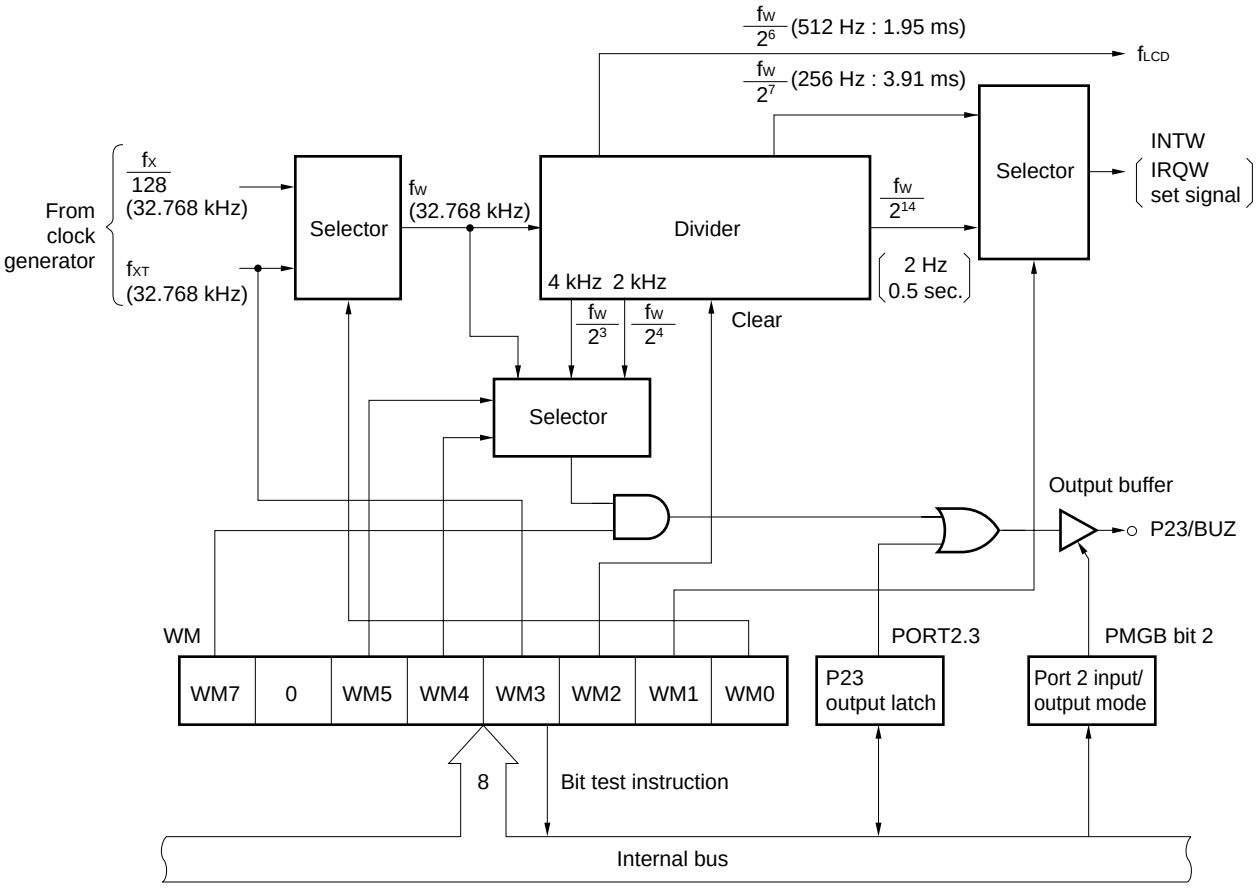
The μPD753108 is provided with a 1-channel of watch timer. This watch timer has the following functions:

(a) Sets the test flag (IRQW) with 0.5 sec interval.
The standby mode can be released by the IRQW.
(b) 0.5 sec interval can be created by both the main system clock and subsystem clock. Take f_X = 4.194304 MHz for the main system clock frequency and f_{XT} = 32.768 kHz for the subsystem clock.
(c) Convenient for program debugging and checking as interval becomes 128 times longer (3.91 ms) with the fast feed mode.
(d) Outputs the frequencies (2.048, 4.096, 32.768 kHz) to the P23/BUZ pin, usable for buzzer and trimming of system clock frequencies.
(e) Clears the frequency divider to make the clock start with zero seconds.

5.4.1 Configuration of watch timer

Figure 5-26 shows the configuration of the watch timer.

Figure 5-26. Watch Timer Block Diagram



Remark The values enclosed in parentheses are applied when f_X = 4.194304 MHz and f_{XT} = 32.768 kHz.

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 5-27.

The watch mode register (WM) is set by an 8-bit manipulation instruction except for bit 3. It is provided for testing the input level at the XT1 pin. Data cannot be entered via the pin.

All the bits except for bit 3 are cleared to “0” by a $\overline{\text{RESET}}$ signal generation.

Example Make a time by the main system clock (4.19 MHz). Enable buzzer output.

```
CLR1    MBE
MOV     XA, #84H
MOV     WM, XA     ; Sets the WM
```

Figure 5-27. Format of Watch Mode Register

Address	7	6	5	4	3	2	1	0	Symbol
F98H	WM7	0	WM5	WM4	WM3	WM2	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	$\frac{f_W}{2^4}$ (2.048 kHz)
0	1	$\frac{f_W}{2^3}$ (4.096 kHz)
1	0	Setting prohibited.
1	1	f_W (32.768 kHz)

Input level to XT1 pin (only bit test is possible)

WM3	0	Input to XT1 pin is low level.
	1	Input to XT1 pin is high level.

Watch operation enable/disable bit

WM2	0	Stops watch operation (clears the frequency divider).
	1	Watch operation possible.

Operation mode select bit

WM1	0	Normal watch mode ($\frac{f_W}{2^{14}}$: sets the IRQW in 0.5 sec).
	1	Fast watch mode ($\frac{f_W}{2^7}$: sets the IRQW in 3.91 ms).

Count clock (f_W) select bit

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

Remark The function in parentheses is available when f_W = 32.768 kHz.

5.5 Timer/Event Counter

The μPD753108 has three channels of timer/event counters. The timer/event counter has the following functions.

(a) Programmable interval timer operation
(b) Square wave output of any frequency to the PTOn pin.
(c) Event counter operation
(d) Divides the frequency of signal input via the TIn pin to 1-Nth of the original signal and outputs the divided frequency to the PTOn pin (frequency divider operation).
(e) Supplies the serial shift clock to the serial interface circuit (for channel 0 only).
(f) Calls the counting status.

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

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

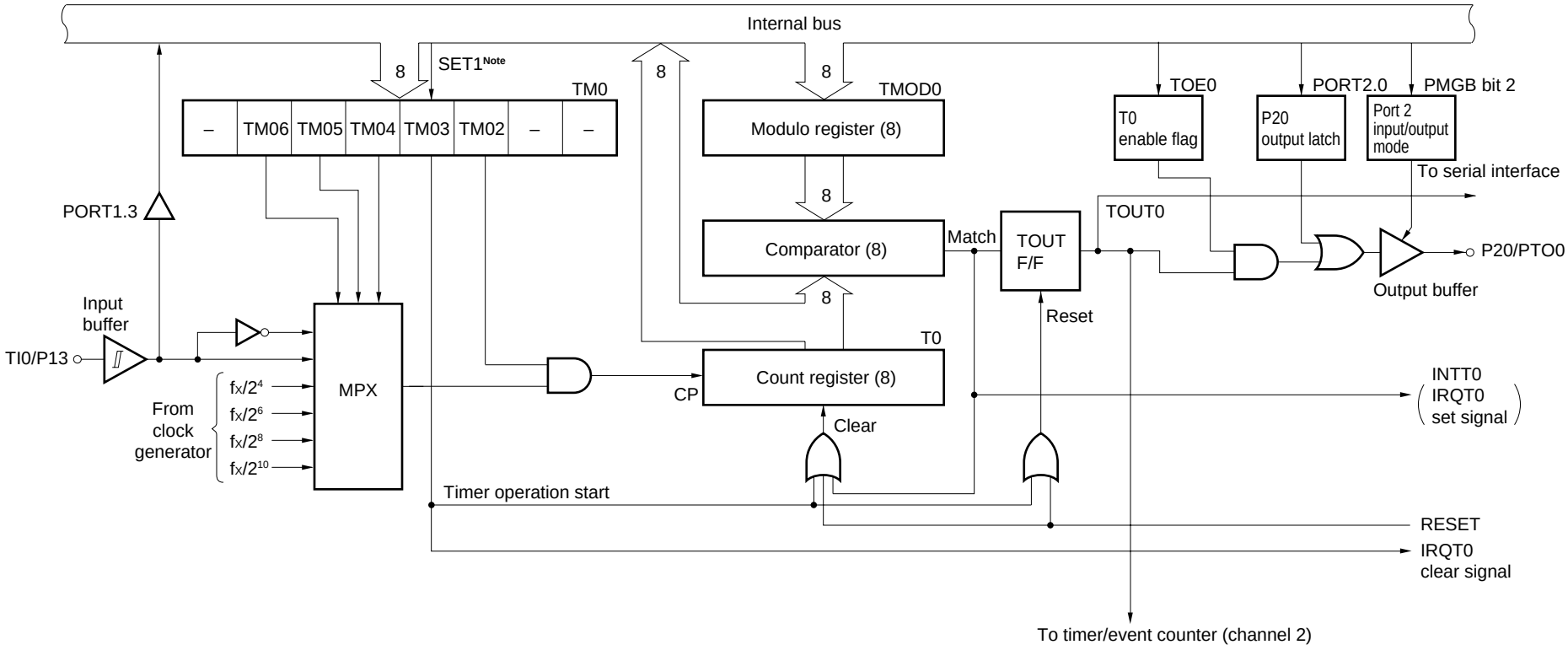
Mode \ Channel		Channel 0	Channel 1	Channel 2
8-bit timer/event counter mode		Usable	Usable	Usable
	Gate control function	n/a[Note]	n/a	Usable
PWM pulse generator mode		n/a	n/a	Usable
16-bit timer/event counter mode		n/a	Usable	
	Gate control function	n/a[Note]	Usable	
Carrier generator mode		n/a	Usable	

Note Used for gate control signal generation

5.5.1 Configuration of timer/event counter

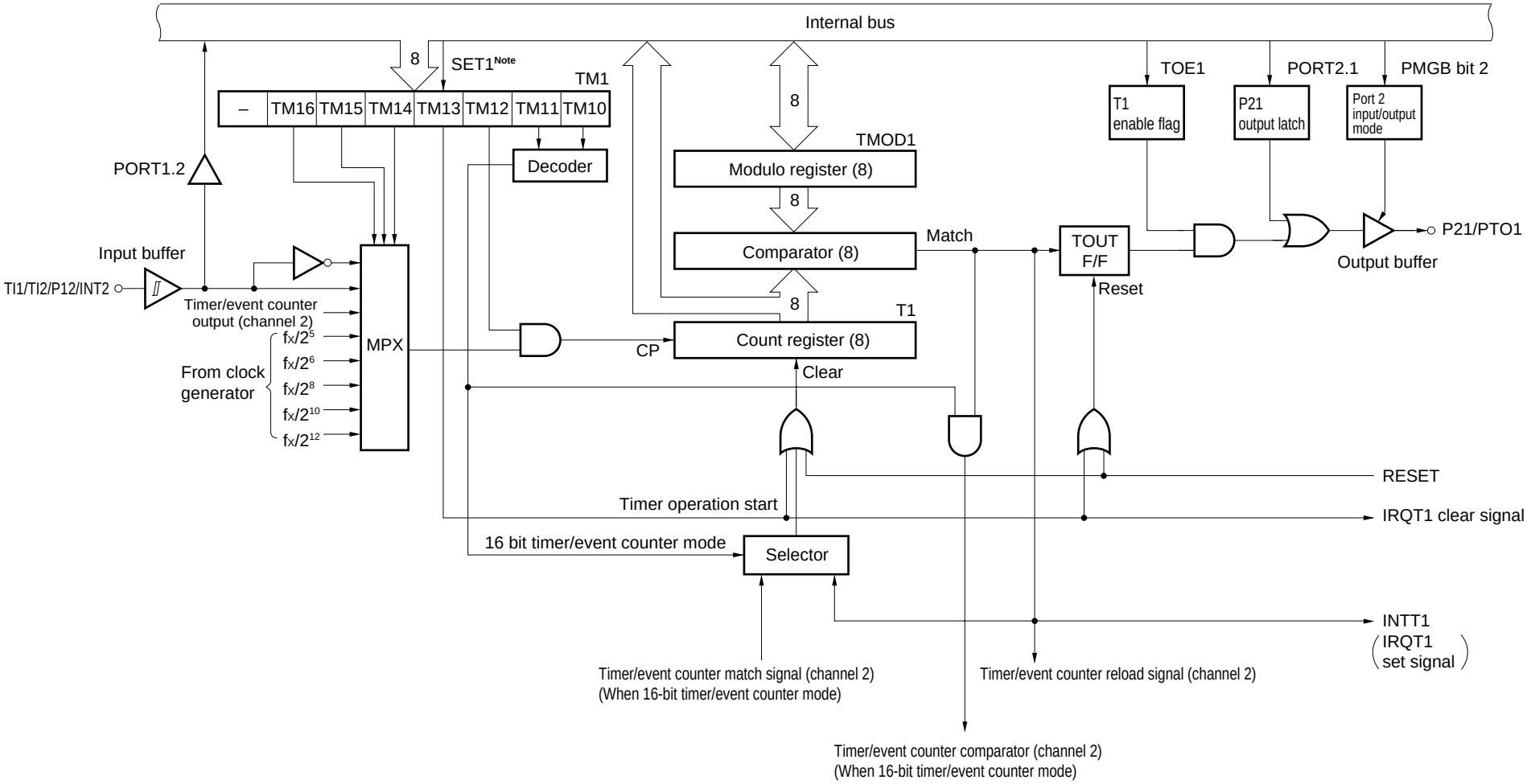
Figures 5-28 to 5-30 show the configuration of the timer/event counter.

Figure 5-28. Timer/Event Counter (Channel 0) Block Diagram



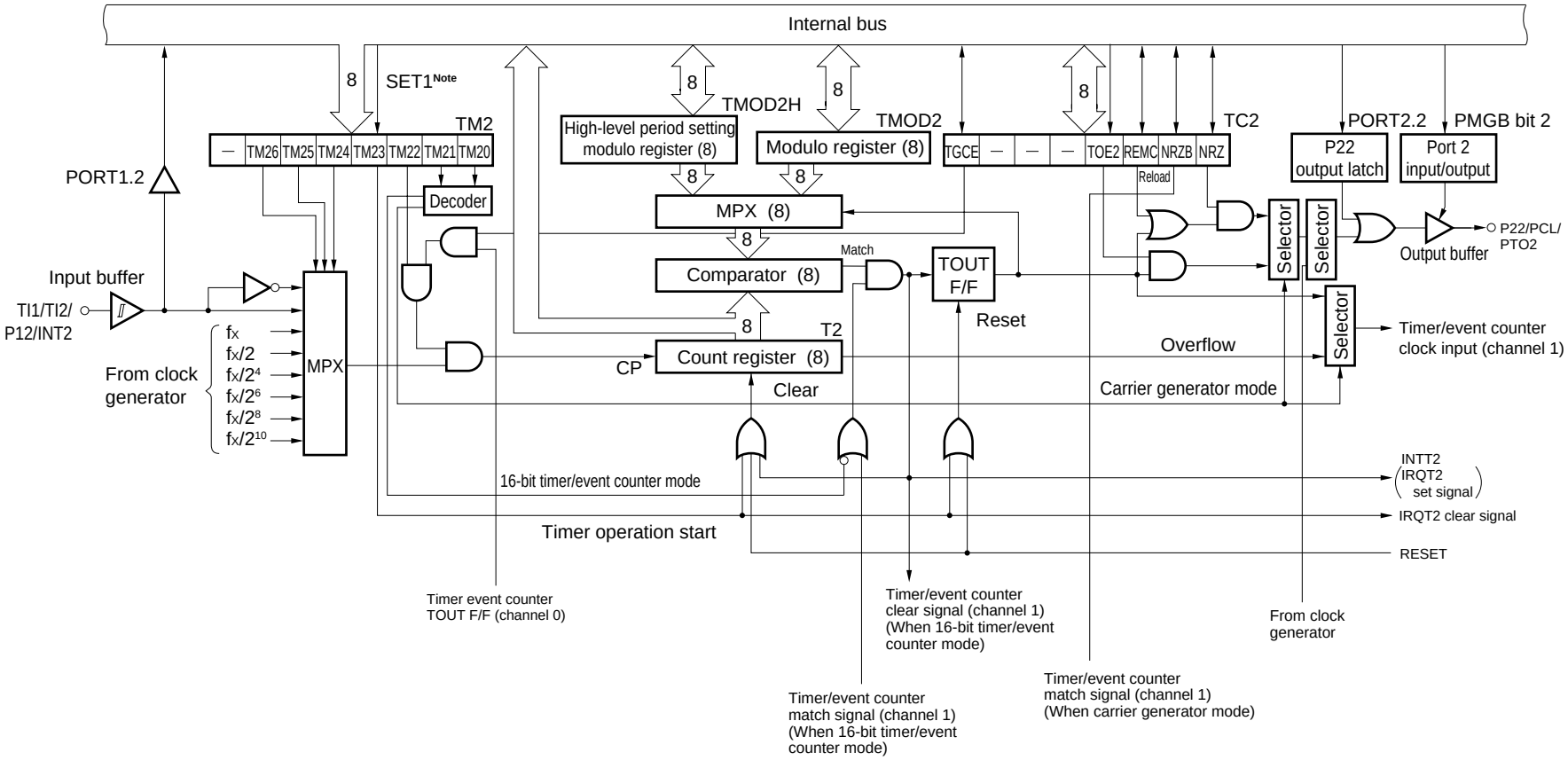
Note Instruction execution

Figure 5-29. Timer/Event Counter (Channel 1) Block Diagram



Note Instruction execution

Figure 5-30. Timer/Event Counter (Channel 2) Block Diagram



Note Instruction execution

(1) Timer/event counter mode register (TM0, TM1, TM2)

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

Its format is shown in Figures 5-31 to 5-33.

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

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

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

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

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

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

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

Figure 5-31. Timer/Event Counter Mode Register (Channel 0) Format

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

Count pulse (CP) selection bit

When f_X = 6.00 MHz

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

When f_X = 4.19 MHz

TM06	TM05	TM04	Count Pulse (CP)
0	0	0	TI0 rising edge
0	0	1	TI0 falling edge
1	0	0	$f_X/2^{10}$ (4.10 kHz)
1	0	1	$f_X/2^8$ (16.4 kHz)
1	1	0	$f_X/2^6$ (65.5 kHz)
1	1	1	$f_X/2^4$ (262 kHz)
Other 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

Figure 5-32. Timer/Event Counter Mode Register (Channel 1) Format (1/2)

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

Count pulse (CP) select bit

When f_X = 6.00 MHz

TM16	TM15	TM14	Count Pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Overflow of timer/event counter channel 2
0	1	1	$f_X/2^5$ (188 kHz)
1	0	0	$f_X/2^{12}$ (1.46 kHz)
1	0	1	$f_X/2^{10}$ (5.86 kHz)
1	1	0	$f_X/2^8$ (23.4 kHz)
1	1	1	$f_X/2^6$ (93.8 kHz)

When f_X = 4.19 MHz

TM16	TM15	TM14	Count Pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Overflow of timer/event counter channel 2
0	1	1	$f_X/2^5$ (131 kHz)
1	0	0	$f_X/2^{12}$ (1.02 kHz)
1	0	1	$f_X/2^{10}$ (4.10 kHz)
1	1	0	$f_X/2^8$ (16.4 kHz)
1	1	1	$f_X/2^6$ (65.5 kHz)

Timer start indication bit

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

Operation mode

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

Figure 5-32. Timer/Event Counter Mode Register (Channel 1) Format (2/2)

Operation mode select bit

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

Note When it is used in combination with the TM20 and TM21 (= 11) of the timer/event counter mode register (channel 2), it enters the carrier generator mode.

Figure 5-33. Timer/Event Counter Mode Register (Channel 2) Format (1/2)

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

Count pulse (CP) select bit

When f_X = 6.00 MHz

TM26	TM25	TM24	Count Pulse (CP)
0	0	0	TI2 rising edge
0	0	1	TI2 falling edge
0	1	0	$f_X/2$ (3.00 MHz)
0	1	1	f_X (6.00 MHz)
1	0	0	$f_X/2^{10}$ (5.86 kHz)
1	0	1	$f_X/2^8$ (23.4 kHz)
1	1	0	$f_X/2^6$ (93.8 kHz)
1	1	1	$f_X/2^4$ (375 kHz)

When f_X = 4.19 MHz

TM26	TM25	TM24	Count Pulse (CP)
0	0	0	TI2 rising edge
0	0	1	TI2 falling edge
0	1	0	$f_X/2$ (2.10 MHz)
0	1	1	f_X (4.19 MHz)
1	0	0	$f_X/2^{10}$ (4.10 kHz)
1	0	1	$f_X/2^8$ (16.4 kHz)
1	1	0	$f_X/2^6$ (65.5 kHz)
1	1	1	$f_X/2^4$ (262 kHz)

Figure 5-33. Timer/Event Counter Mode Register (Channel 2) Format (2/2)

Timer start indication bit

TM23	When 1 is written into the bit, the counter and IRQT2 flag are cleared. If bit 2 is set to 1, count operation is started.

Operation mode

TM22	Count Operation
0	Stop (retention of count contents)
1	Count operation

Operation mode select bit

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

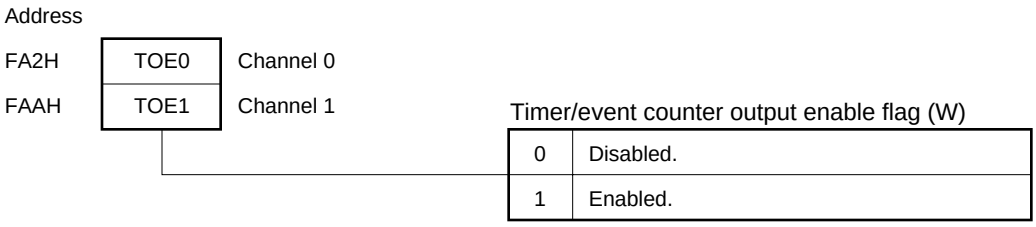
(2) Timer/event counter output enable flag (TOE0, TOE1)

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

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

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

Figure 5-34. Timer/Event Counter Output Enable Flag Format



(3) Timer/event counter control register (TC2)

The timer/event counter control register (TC2) is an 8-bit register which controls the timer/event counter. Its format is shown in Figure 5-35.

The timer/event counter control register (TC2) is set by an 8-bit/4-bit memory manipulation instruction.

All the bits of the timer/event counter control register (TC2) are cleared to “0” by the internal reset signal generation.

Figure 5-35. Timer/Event Counter Control Register Format

Address	7	6	5	4	3	2	1	0	Symbol
F92H	TGCE	—	—	—	TOE2	REMC	NRZB	NRZ	TC2

Gate control enable flag

TGCE	Gate Control
0	Disabled. (If bit 2 of the TM2 is set to “1”, the count operation is performed regardless of the sampling clock status.)
1	Enabled. (If bit 2 of the TM2 is set to “1”, the count operation is performed when the sampling clock is high, and is stopped when the sampling clock is low)

Timer output enable flag

TOE2	Timer Output
0	Disabled (outputs the low level).
1	Enabled.

Remote control output control flag

REMC	Remote Control Output
0	Outputs the carrier pulse when NRZ = 1.
1	Output a high-level signal when NRZ = 1.

No return zero buffer flag

NRZB	No return zero data to be output next. Transferred to the NRZ when a timer/event counter (channel 1) interrupt is generated.

No return zero flag

NRZ	No Return Zero Data
0	Outputs a low-level signal.
1	Outputs the carrier pulse or high-level signal.

5.5.2 8-bit timer/event counter mode operation

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

(1) Register setting

The following four registers are used in the 8-bit timer/event counter mode.

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

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

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

When the 8-bit timer/event counter mode is used, TMn must be set as follows (For the format of the TMn, see **Figures 5-31** to **5-33**).

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

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

Figure 5-36. Timer/Event Counter Mode Register Setup (1/3)

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

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

Count pulse (CP) selection bit

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

Timer start indication bit

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

Operation mode

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

Figure 5-36. Timer/Event Counter Mode Register Setup (2/3)

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

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

Count pulse (CP) selection bit

TM16	TM15	TM14	Count Pulse (CP)
0	0	0	TI1 rising edge
0	0	1	TI1 falling edge
0	1	0	Timer/event counter channel 2 overflow
0	1	1	$f_X/2^5$
1	0	0	$f_X/2^{12}$
1	0	1	$f_X/2^{10}$
1	1	0	$f_X/2^8$
1	1	1	$f_X/2^6$

Timer start indication bit

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

Operation mode

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

Operation mode selection bit

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

Figure 5-36. Timer/Event Counter Mode Register Setup (3/3)

(c) In the case of timer/event counter (channel 2)

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

Count pulse (CP) selection bit

TM26	TM25	TM24	Count Pulse (CP)
0	0	0	TI2 rising edge
0	0	1	TI2 falling edge
0	1	0	$f_X/2$
0	1	1	f_X
1	0	0	$f_X/2^{10}$
1	0	1	$f_X/2^8$
1	1	0	$f_X/2^6$
1	1	1	$f_X/2^4$

Timer start indication bit

TM23	When “1” is written to the bit, the counter and IRQT2 flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM22	Count Operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM21	TM20	Mode
0	0	8-bit timer/event counter mode

(b) Timer/event counter control register (TC2)

Figure 5-37 shows the setting of the TC2 when it is used in an 8-bit timer/event counter mode (See **Figure 5-35 Timer/Event Counter Control Register Format**).

The TC2 is manipulated by an 8-bit/4-bit manipulation instruction and bit manipulation instruction.

The TC2 is cleared to 00H by an internal reset signal generation.

The flag indicated by the full lines indicates a flag which is used in the 8-bit timer/event counter mode.

The flag indicated by the broken lines must not be used in the 8-bit timer/event counter mode (Set 0).

Figure 5-37. Timer/Event Counter Control Register Setup

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

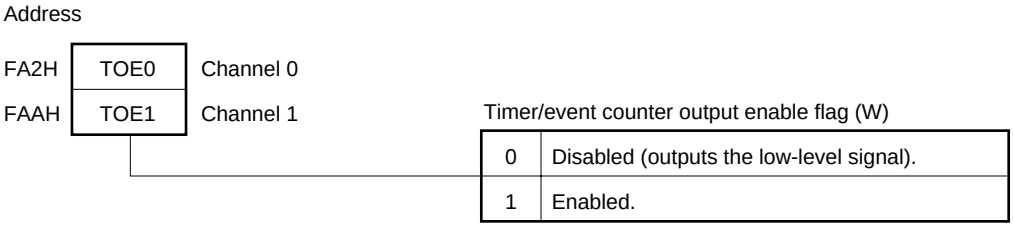
Gate control enable flag

TGCE	Gate Control
0	Disabled. (If bit 2 of the TM2 is set to “1”, the count operation is performed regardless of the status of the sampling clock.)
1	Enabled. (If bit 2 of the TM2 is set to “1”, the count operation is performed when the sampling clock is high, and is stopped when the sampling clock is low.)

Timer output enable flag

TOE2	Timer Output
0	Disabled (outputs the low-level signal).
1	Enabled.

Figure 5-38. Timer/Event Counter Output Enable Flag Setup



(2) Timer/event counter time setting

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

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

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

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

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

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

Mode Register			During 6.00-MHz Operation		During 4.19-MHz Operation	
TM06	TM05	TM04	Resolution	Max. Time Setting	Resolution	Max. Time Setting
1	0	0	171 μs	43.7 ms	244 μs	62.5 ms
1	0	1	42.7 μs	10.9 ms	61.0 μs	15.6 ms
1	1	0	10.7 μs	2.73 ms	15.3 μs	3.91 ms
1	1	1	2.67 μs	683 μs	3.81 μs	977 μs

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

Mode Register			During 6.00-MHz Operation		During 4.19-MHz Operation	
TM16	TM15	TM14	Resolution	Max. Time Setting	Resolution	Max. Time Setting
0	1	1	5.33 μs	1.37 ms	7.63 μs	1.95 ms
1	0	0	683 μs	175 ms	977 μs	250 ms
1	0	1	171 μs	43.7 ms	244 μs	62.5 ms
1	1	0	42.7 μs	10.9 ms	61.0 μs	15.6 ms
1	1	1	10.7 μs	2.73 ms	15.3 μs	3.91 ms

(c) When timer/event counter (channel 2)

Mode Register			During 6.00-MHz Operation		During 4.19-MHz Operation	
TM26	TM25	TM24	Resolution	Max. Time Setting	Resolution	Max. Time Setting
0	1	0	333 ns	85.3 μs	477 ns	122 μs
0	1	1	167 ns	42.7 μs	238 ns	61.0 μs
1	0	0	171 μs	43.7 ms	244 μs	62.5 ms
1	0	1	42.7 μs	10.9 ms	61.0 μs	15.6 ms
1	1	0	10.7 μs	2.73 ms	15.3 μs	3.91 ms
1	1	1	2.67 μs	683 μs	3.81 μs	977 μs

(3) Timer/event counter operation

The timer/event counter operates as follows. In the operation, the gate control enable flag (TGCE) of the timer/event counter control register (TC2) must be set to 0.

Figure 5-39 shows the configuration of the timer/event counter.

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

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

Figure 5-40 is a timing chart of the timer/event counter.

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

<1> Set a count in the TMODn.

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

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

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

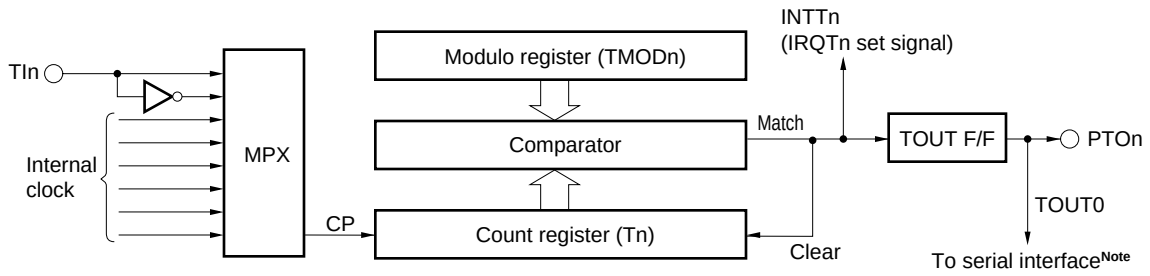
<1> Clear the output latch of P2n.

<2> Set port 2 to the output mode.

<3> Make a status wherein the pull-up resistor is not incorporated in port 2 (When outputting PTO2, disable output of PCL).

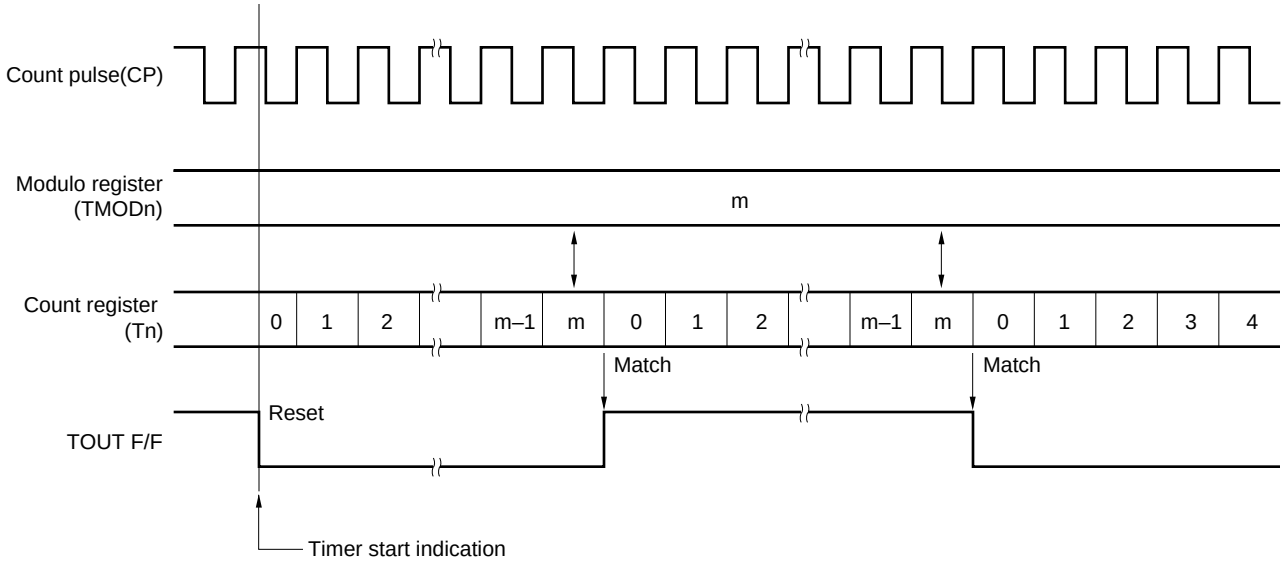
<4> Set the timer/event counter output enable flag (TOEn) to 1.

Figure 5-39. Configuration of Timer/Event Counter



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

Figure 5-40. Count Operation Timing



Remark m: Modulo register setup value
n = 0 to 2

(4) Event counter operation with gate control function (8-bit)

The timer/event counter (channel 2) can be used as an event counter with a gate control function. Set the gate control enable flag (TGCE) of the timer/event counter control register to 1 when using this function.
When timer/event counter channel 0 counts to the specified number, the gate signal is generated.
When the gate signal (output of TOUT F/F of T0) is high, the count pulses of timer/event counter (channel 2) can be counted as shown in Figure 5-42 (for details, refer to **(3) Timer/event counter operation**).

<1> The count pulse (CP) is selected by setting the mode register (TM2), and the CP is input to the count register (T2) when the gate signal is high.
<2> Interrupt is generated at the rising edge and falling edge of the gate signal. Normally, the contents of the T2 are read out by an interrupt subroutine at the falling edge and the T2 is cleared for the subsequent count operation.

Figure 5-42 shows the timing chart of the event counter operation.
The event counter normally starts operation by the following procedure.

<1> Set the operation mode, count pulse, and counter clear indication in the TM2.
<2> Set the number of count in the TMOD0.
<3> Set the operation mode and start indication in the TM0.

Caution A value other than 00H must be set in the modulo register (TMOD0, TMOD2).

Figure 5-41. Configuration of Event Count

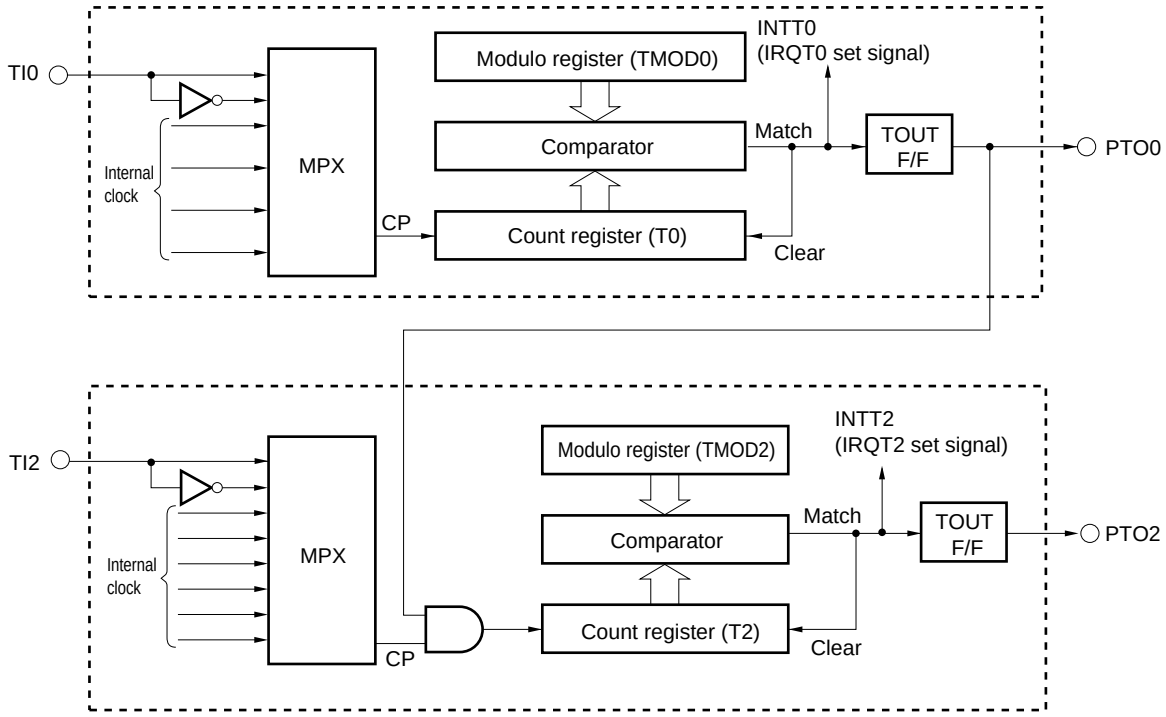
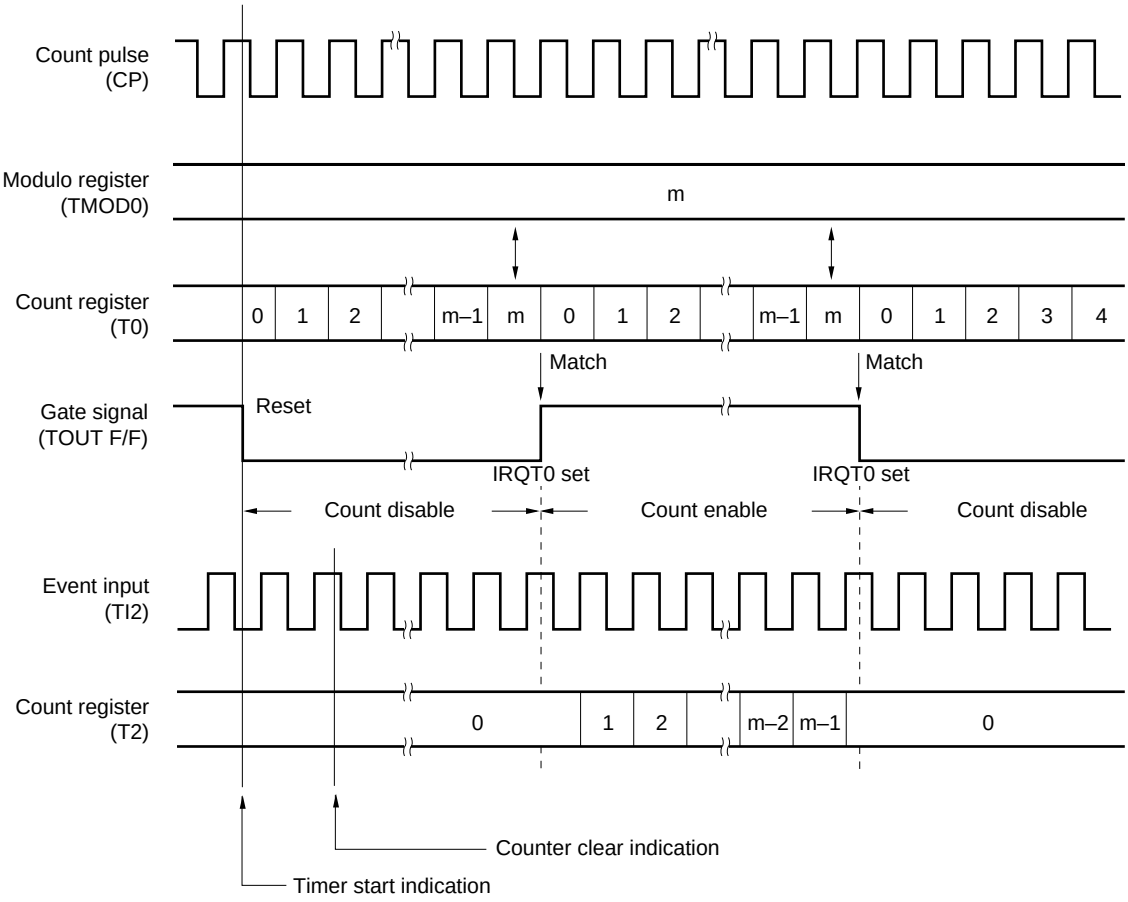


Figure 5-42. Event Count Operation Timing



Remark m: Modulo register setup value

(5) 8-bit timer/event counter mode application

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

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

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

<Program example>

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

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

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

- Set the high-order four bits of the mode register (TMn) to 0000 to select rising edge.
- Set the low-order four bits of the TMn to 1100B.
- The value set in the modulo register (TMODn) is 99 = 100 – 1.

<Program example>

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

(c) Application used as an event counter with sampling time (15 ms) and hold time (2 ms) after 121 μs count disabled time (@f_X = 4.19 MHz).
Set the timer/event counter (channel 0) as follows.

- Set the high-order 4 bits of the mode register (TM0) to 0101B to select the longest time 15.6 ms.
- Set the low-order 4 bits of the TM0 to 1100B to select the 8-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo register (TMOD0) to 01H (121 μs) initially and then 20H (15.03 ms) and F5H (2.02 ms).

Set the timer/event counter (channel 2) as follows.

- Set the high-order 4 bits of the TM2 to 0000B in order to select the TI2 rising edge.
- Set the low-order 4 bits of the TM2 to 1100B in order to select the 8-bit timer/event counter mode and count operation and indicate counter clear.
- Set the TGCE to “1” and enable gate control.
- Set the TMOD2 to the greatest value FFH.
- Designate the memory MEM which stores the contents of the count register (T2).

<Program example>

```
MAIN:   SEL     MB15            ; or CLR1 MBE
        SET1    TGCE            ; Enables gate control.
        MOV     XA, #00001100B
        MOV     TM2, XA         ; Sets the mode and clears the counter.
        MOV     XA, #001H
        MOV     TMOD0, XA       ; Sets the modulo (during the initial counting disabled time).
        MOV     XA, #01011100B
        MOV     TM0, XA         ; Sets the mode and indicates timer start.
        MOV     B, #00H         ; Initialization
        EI                      ; Enables interrupt.
        EI      IET0            ; Enables timer (channel 0) interrupt.
```

```
; <Subroutine>
            INCS    B
            SKE     B, #02H
            BR      SAMP
  HOLD:     MOV     XA, #020H
            MOV     TMOD0, XA       ; Rewrite modulo (2 ms)
            MOV     XA, T2
            MOV     MEM, XA         ; Read the counter
            SET1    TM2. 3          ; Clear the counter
            MOV     B, #00H
            BR      END
  SAMP:     MOV     XA, #0F5H
            MOV     TMOD0, XA       ; Rewrite modulo (15 ms)
  END :     RETI
```

Remark In this example, TI0 and TI1 can be used as input pins.

When the sampling clock goes to high, the count operation starts and, at the same time, the initial interrupt is generated. The TMOD0 is updated to F5H and then the count operation continues for 15 ms.
When sampling clock goes to low, the count operation stops and, at the same time, the second interrupt is generated. The TMOD0 is updated to 20H and then the count operation stops for 2 ms. The contents of the T2 are read out and it is cleared for the next count operation.
Then, repeat the above operation.

5.5.3 PWM pulse generator mode (PWM mode) operation

It performs as an 8-bit PWM pulse generator in this mode.

(1) Register setting

The following five registers are used in the PWM mode.

- Timer/event counter mode register (TM2)
- Timer/event counter control register (TC2)
- Timer/event counter count register (T2)
- Timer/event counter high-level period setting modulo register (TMOD2H)
- Timer/event counter modulo register (TMOD2)

(a) Timer/event counter mode register (TM2)

When using the PWM mode, set the TM2 as shown in Figure 5-43. For the format of the TM2, see **Figure 5-33 Timer/Event Counter Mode Register (Channel 2) Format**.

The TM2 is manipulated by an 8-bit manipulation instruction. Bit 3 is a timer start indication bit. It can be manipulated bit-wise and is automatically cleared to 0 when the timer starts.

The TM2 is cleared to 00H when an internal reset signal is generated.

Figure 5-43. Timer/Event Counter Mode Register Setup

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

Count pulse (CP) selection bit

TM26	TM25	TM24	Count Pulse (CP)
0	0	0	TI2 rising edge
0	0	1	TI2 falling edge
0	1	0	$f_X/2$
0	1	1	f_X
1	0	0	$f_X/2^{10}$
1	0	1	$f_X/2^8$
1	1	0	$f_X/2^6$
1	1	1	$f_X/2^4$

Timer start indication bit

TM23	When “1” is written to the bit, the counter and IRQT2 flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM22	Count Operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM21	TM20	Mode
0	1	PWM pulse generator mode

(b) Timer/event counter control register (TC2)

When using the PWM mode, set the TC2 as shown in Figure 5-44 (For the format of TC2, see **Figure 5-35 Timer/Event Counter Control Register Format**).

The TC2 is manipulated by an 8-bit/4-bit manipulation instruction and bit manipulation instruction.

The TC2 is cleared to 00H by an internal reset signal generation.

The flag indicated by the full lines is used in the PWM mode.

The flag indicated by the broken lines must not be used for the PWM mode (Set 0).

Figure 5-44. Timer/Event Counter Control Register Setup

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

Timer output enable flag

TOE2	Timer Output
0	Disabled (output the low-level signal).
1	Enabled.

(2) PWM pulse generator operation

The PWM pulse generator operates as follows. Figure 5-45 shows its configuration.

<1> When the mode register (TM2) is set, the count pulse (CP) is selected and input to the count register (T2).
<2> The contents of the T2 are compared with those of the high-level period setting modulo register (TMOD2H), and if they are equal, a match signal is generated and the timer out flip-flop (TOUT F/F) flips.
<3> The contents of the T2 are compared with those of the modulo register (TMOD2), and if they are equal, a match signal is generated and an interrupt request flag (IRQT2) is set. At the same time, the TOUT F/F flips.
<4> The above operations <2> and <3> repeat alternatively.

Figure 5-46 shows the timing chart of the PWM pulse generator.
The PWM pulse generator normally starts operation in the following procedure.

<1> Set the number of count in the TMOD2H.
<2> Set the number of low-level count in the TMOD2.
<3> Set the operating mode, count pulse, and start indication in the TM2.

Caution Set values other than 00H in the modulo register (TMOD2) and high-level period setting modulo register (TMOD2H).

When using the timer/event counter output pin (PTO2), set the alternate function pins P22 and PCL as follows.

<1> Clear the output latch of P22.
<2> Set port 2 to output mode.
<3> Make a status wherein port 2's on-chip pull-up resistor is not incorporated, and disable the PCL output.
<4> Set the timer/event counter output enable flag (TOE2) to 1.

Figure 5-45. Configuration of PWM Pulse Generator

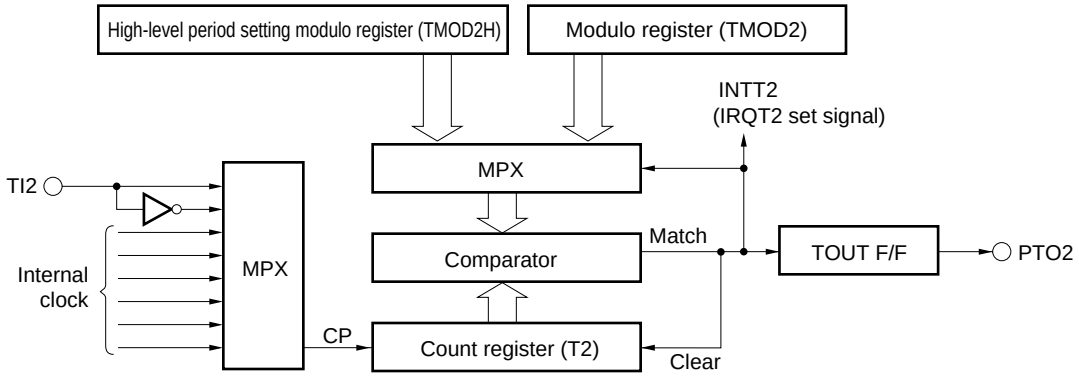
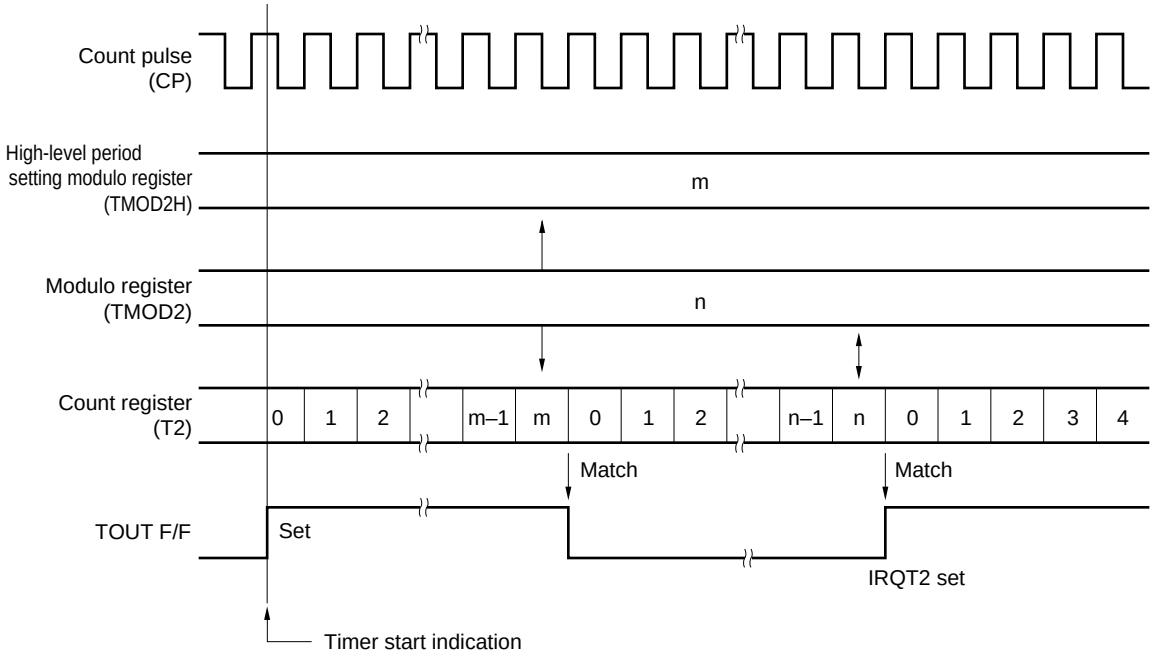


Figure 5-46. PWM Pulse Generator Operation Timing



Remark m : Setup value of high-level period setting modulo register
n : Modulo register setup value

(3) PWM mode application

The pulses (frequency is 38.0 kHz (cycle is 26.3 μs) and duty ratio is 1/3) are output to the PTO2 pin (@f_X = 4.19 MHz).

- Set the high-order 4 bits of the mode register (TM2) to 0011B and select the longest setup time 61.1 μs.
- Set the low-order 4-bits of the TM2 to 1101B and select the PWM mode and count operation and indicate timer start.
- Set the timer output enable flag (TOE2) to “1” and enable the timer output.
- The high-level period setting modulo register (TMOD2H) is set as follows.

$$\frac{1}{3} \times \frac{26.3\ \mu s}{239\ ns} - 1 = 36.7 - 1 \cong 36 = 24H$$

- The modulo register (TMOD2) is set as follows.

$$\frac{2}{3} \times \frac{26.3\ \mu s}{239\ ns} - 1 = 73.4 - 1 \cong 72 = 48H$$

<Program example>

```
SEL     MB15          ; or CLR1 MBE
SET1    TOE2          ; Enables timer output.
MOV     XA, #024H
MOV     TMOD2H, XA    ; Sets the modulo (high-level period).
MOV     XA, #48H
MOV     TMOD2, XA     ; Sets the modulo (low-level period).
MOV     XA, #00111101B
MOV     TM2, XA       ; Sets the mode and timer start.
```

Remark In this example, TI0, TI1, and TI2 can be used as input pins.

5.5.4 16-bit timer/event counter mode operation

Used as a 16-bit timer/event counter in this mode. It performs 16-bit programmable interval timer and event counter. When it is used in the 16-bit timer/event counter mode, the channel 1 and channel 2 of the timer/event counter are used in combination.

(1) Register setting

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

- Timer/event counter mode registers (TM1, TM2)
- Timer/event counter control register (TC2)**Note**
- Timer/event counter count registers (T1, T2)
- Timer/event counter modulo registers (TMOD1, TMOD2)

Note The timer/event counter (channel 1) uses the timer/event counter output enable flag (TOE1).

(a) Timer/event counter mode registers (TM1, TM2)

When using the 16-bit timer/event counter mode, set the TM1 and TM2 as shown in Figure 5-47. For the formats of the TM1 and TM2, see **Figure 5-32 Timer/Event Counter Mode Register (Channel 1) Format** and **Figure 5-33 Timer/Event Counter Mode Register (Channel 2) Format**, respectively.

The TM1 and TM2 are manipulated by 8-bit manipulation instructions. 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 TM1 and TM2 are cleared to 00H by an internal reset signal generation.

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

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

Figure 5-47. Timer/Event Counter Mode Register Setup

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	—	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1
F90H	—	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) selection bit (n = 1, 2)

TMn6	TMn5	TMn4	TM1	TM2
0	0	0	TI1 rising edge	TI2 rising edge
0	0	1	TI1 falling edge	TI2 falling edge
0	1	0	Count register (T2) overflow	$f_X/2$
0	1	1	$f_X/2^5$	f_X
1	0	0	$f_X/2^{12}$	$f_X/2^{10}$
1	0	1	$f_X/2^{10}$	$f_X/2^8$
1	1	0	$f_X/2^8$	$f_X/2^6$
1	1	1	$f_X/2^6$	$f_X/2^4$

Timer start indication bit

TM23	When “1” is written to the bit, the counter and IRQTn flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TM22	Count Operation
0	Stop (retention of count contents)
1	Count operation

Operation mode selection bit

TM21	TM20	TM11	TM10	Mode
1	0	1	0	16-bit timer/event counter mode

(b) Timer/event counter control register (TC2)

When using the 16-bit timer/event counter mode, set the TC2 as shown in Figure 5-48. For the format of the TC2, see **Figure 5-35 Timer/Event Counter Control Register Format**.

The TC2 is manipulated by an 8-bit/4-bit manipulation instruction and bit manipulation instruction.

The TC2 is cleared to 00H by an internal reset signal generation.

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

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

Figure 5-48. Timer/Event Counter Control Register Setup

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

Gate control enable flag

TGCE	Gate Control
0	Disabled. (If the bit 2 of the TM2 is set to “1”, the count opration is performed regardless of the status of sampling clock.)
1	Enabled. (If the bit 2 of the TM2 is set to “1”, the count opration is performed when the sampling clock is high and is stopped when the sampling clock is low.)

Timer output enable flag

TOE2	Timer Output
0	Disabled (outputs the low level signal).
1	Enabled.

(2) Timer/event counter time setting

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

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

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

Once the timer is set, an interrupt request signal (IRQT2) is generated at the intervals set in the timer. Table 5-8 lists the resolution and maximum allowable time setting (that is, time when FFH is set in the modulo register) for each count pulse to the timer/event counter.

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

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

Mode Register			During 6.00-MHz Operation		During 4.19-MHz Operation	
TM16	TM15	TM14	Resolution	Max. Time Setting	Resolution	Max. Time Setting
0	1	1	5.33 μs	350 ms	7.63 μs	500 ms
1	0	0	683 μs	44.7 s	977 μs	64.0 s
1	0	1	171 μs	11.2 s	244 μs	16.0 s
1	1	0	42.7 μs	2.80 s	61.0 μs	4.00 s
1	1	1	10.7 μs	699 ms	15.3 μs	1.00 s

(b) When timer/event counter (channel 2)

Mode Register			During 6.00-MHz Operation		During 4.19-MHz Operation	
TM26	TM25	TM24	Resolution	Max. Time Setting	Resolution	Max. Time Setting
0	1	0	333 ns	21.8 ms	477 ns	31.3 ms
0	1	1	167 ns	10.9 ms	238 ns	15.6 ms
1	0	0	171 μs	11.2 s	244 μs	16.0 s
1	0	1	42.7 μs	2.80 s	61.0 μs	4.00 s
1	1	0	10.7 μs	699 ms	15.3 μs	1.00 s
1	1	1	2.67 μs	175 ms	3.81 μs	250 ms

(3) Timer/event counter operation

The timer/event counter operates as follows. In this operation mode, set the gate control enable flag (TGCE) of the timer/event counter control register (TC2) to 0.

Figure 5-49 shows the configuration of the timer/event counter.

<1> The count pulse (CP) is selected by setting the mode registers (TM1 and TM2), and is input to the count register (T2). The overflow of the T2 is input to the count register (T1).

<2> The contents of the T1 and those of the modulo register (TMOD1) are compared, and if they are equal, a match signal is generated.

<3> The contents of the T2 are compared with those of the modulo register (TMOD2), and if they are equal, a match signal is generated.

<4> If the match signals of <2> and <3> above are the same, an interrupt request flag (IRQT2) is set. At the same time, the timer out flip-flop (TOUT F/F) flips.

Figure 5-50 shows the timing chart of the timer/event counter operation.

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

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

<2> Set the low-order 8-bits of the count expressed by a 16-bit width in the TMOD2.

<3> Set the operating mode and count pulse in the TM1.

<4> Set the operating mode, count pulse, and start indication in the TM2.

Caution Set a value other than 00H to the modulo register (TMOD2).

When using the timer/event counter output pin (PTO2), set the alternate function pins P22 and PCL as follows.

<1> Clear the output latch of P22.

<2> Set port 2 to the output mode.

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

<4> Set the timer/event counter output enable flag (TOE2) to 1.

Figure 5-49. Timer/Event Counter Operation Configuration

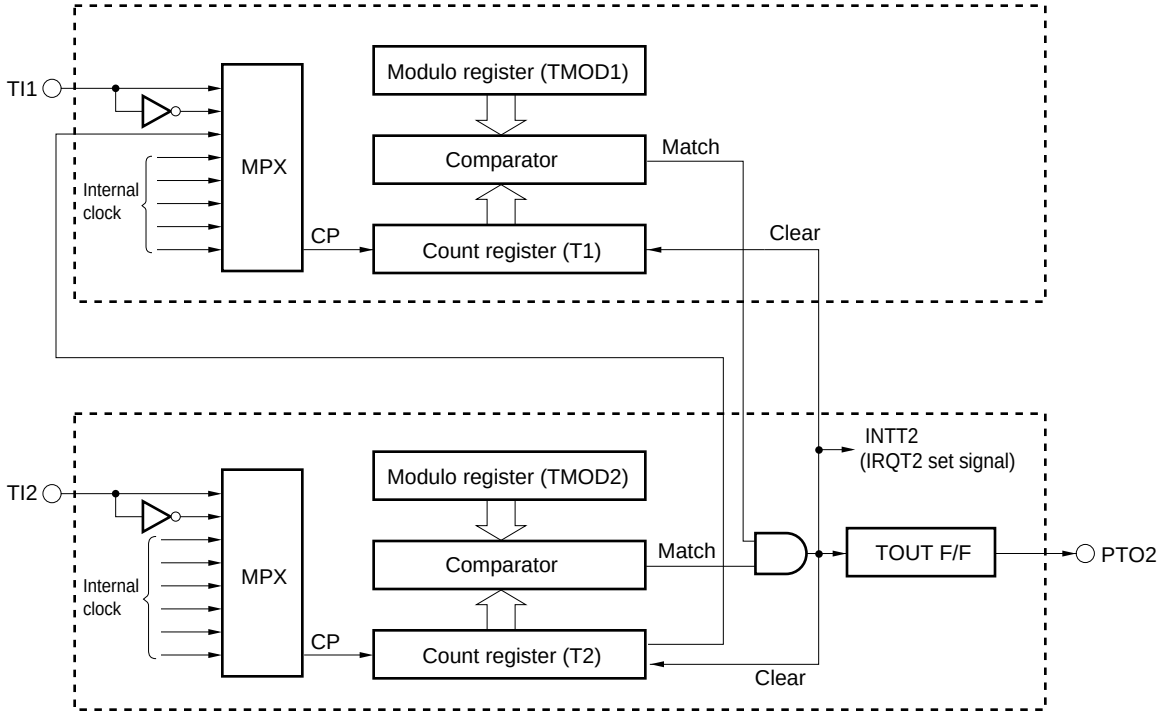
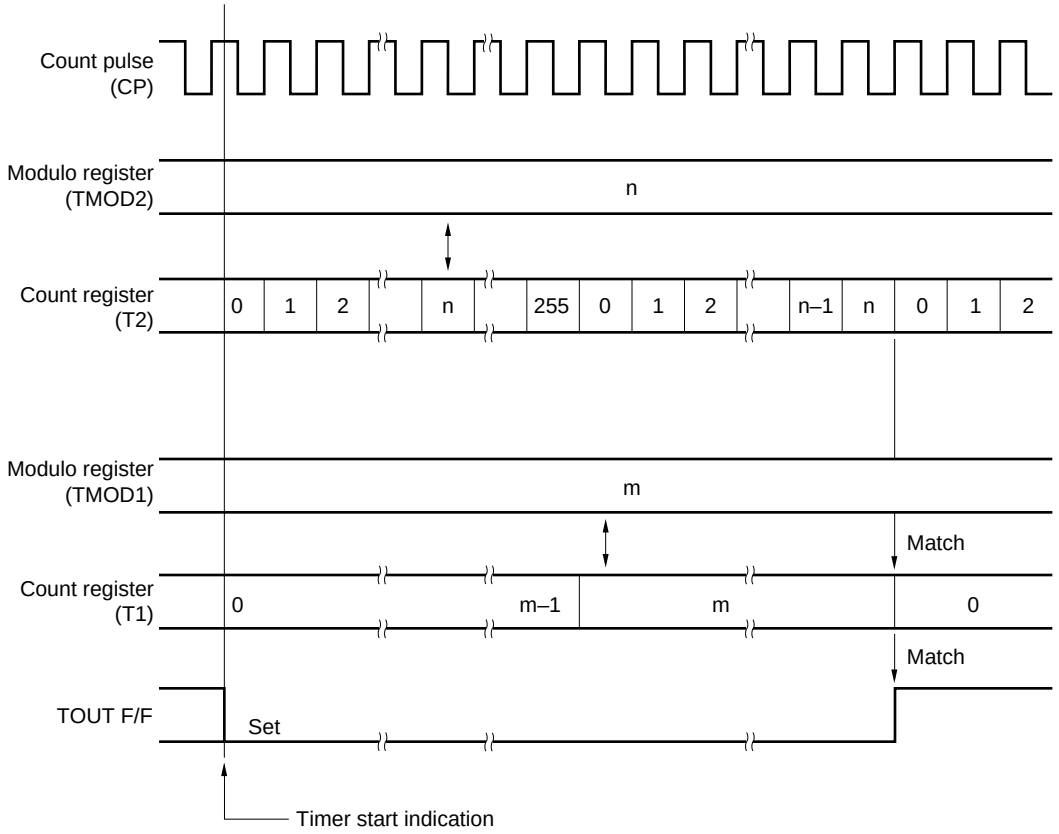


Figure 5-50. Count Operation Timing



Remark m : Modulo register (TMOD1) setup value
n : Modulo register (TMOD2) setup value

(4) Event counter operation with gate control function (16 bits)

The timer/event counter (channel 1) and timer/event counter (channel 2) can be used as an event counter with gate control function. Set the gate control enable flag (TGCE) of the timer/event counter control register to 1 when using this function.

When timer/event counter (channel 0) counts to the specified number, the gate signal is generated.

When the gate signal (output of TOUT F/F of T0) is high, the count pulses of timer/event counters (channel 1 and channel 2) can be counted as shown in Figure 5-52 (for details, refer to **(3) Timer/event counter operation**).

<1> When the mode registers (TM1 and TM2) are set, the count pulse (CP) is selected. When the gate signal is high, the CP is input to the count register (T2). The overflow of the T2 is input to the count register (T1).

<2> Interrupts are generated at the rising edge and falling edge of the gate signal. Normally, the contents of the T1 and T2 are read out by an interrupt subroutine of the falling edge and they are then cleared for the next count operation.

Figure 5-52 shows the timing chart of the event counter operation.

The event counter normally starts operation in the following procedure.

<1> Set the operation mode and count pulse in the TM1.

<2> Set the operation mode, count pulse, and counter clear indication in the TM2.

<3> Set the number of count in the TMOD0.

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

Cautions 1. Set a value other than 00H in the modulo registers (TMOD0, TMOD1, TMOD2).

2. Do not set 1 to the timer/event counter interrupt enable flag (IET1).

Figure 5-51. Event Count Operation Configuration

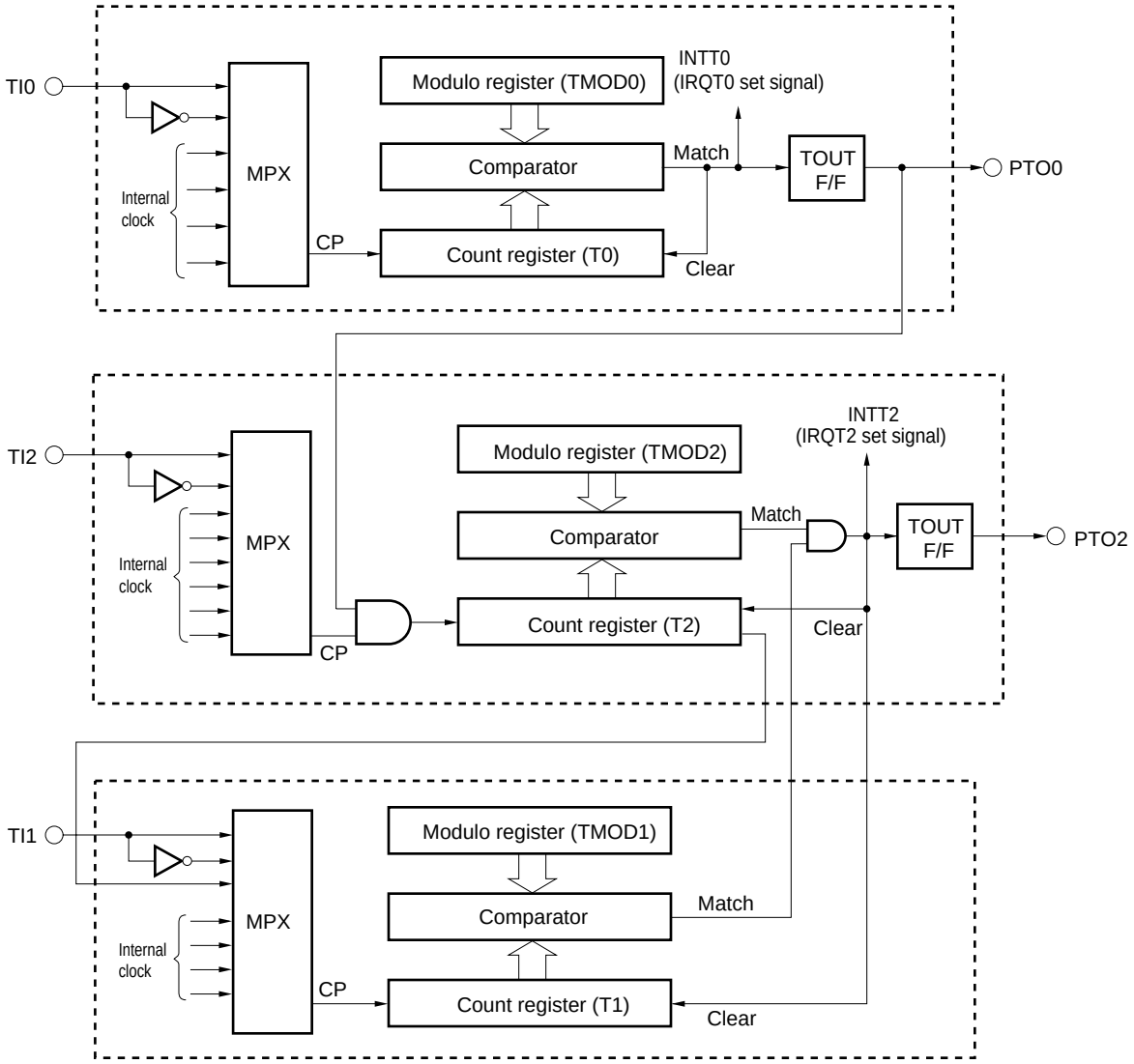
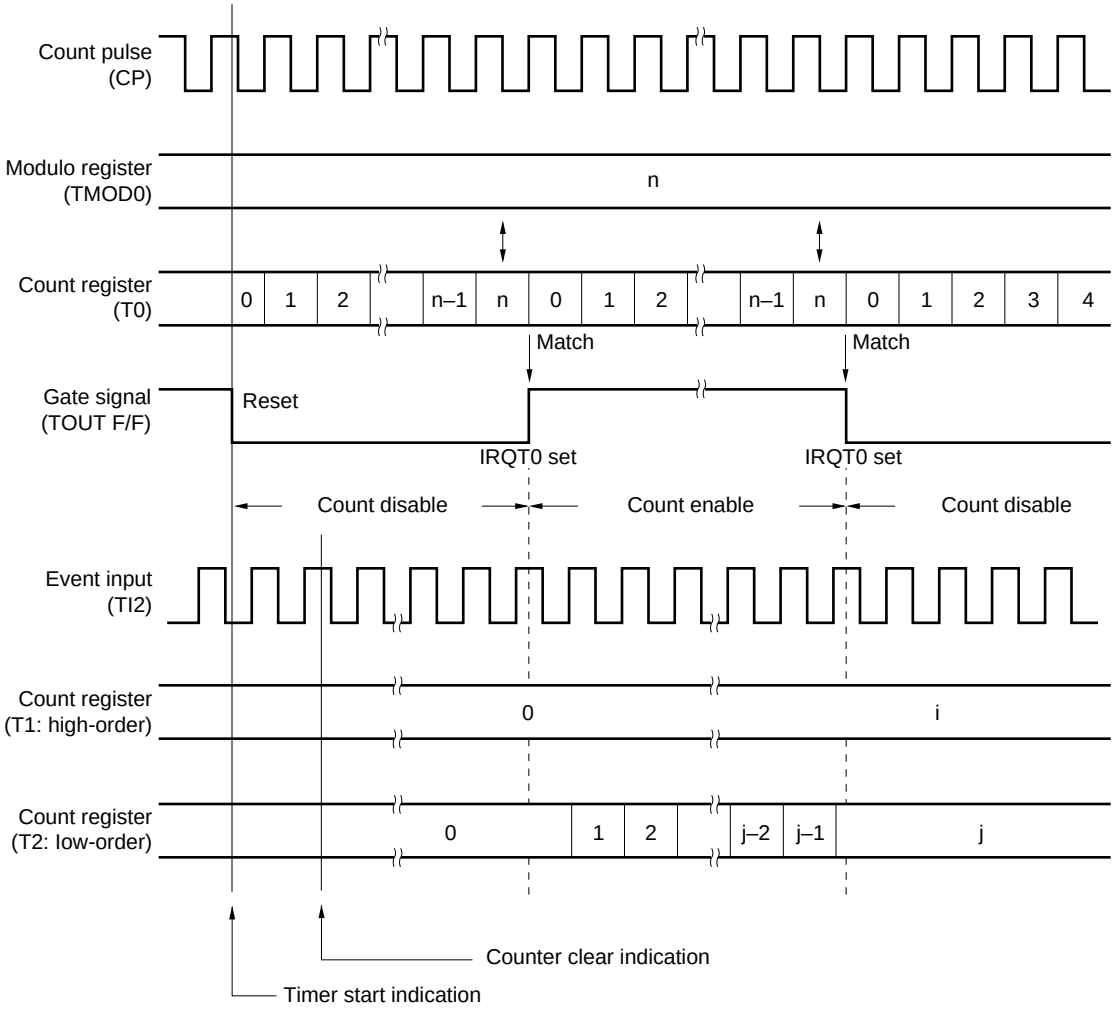


Figure 5-52. Event Count Operation Timing



Remark i : Count register (T1: high-order) setup value
j : Count register (T2: low-order) setup value
n : Modulo register (TMOD0) setup value

(5) 16-bit timer/event counter mode application

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

- Set the high-order 4-bits of the mode register (TM1) to 0010B and select the overflow of the count register (T2).
- Set the high-order 4-bits of the TM2 to 0100B and select the longest setup time 16.0 sec.
- Set the low-order 4-bits of the TM1 to 0010B and select the 16-bit timer/event counter mode.
- Set the low-order 4-bits of the TM2 to 1110B and select the 16-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo registers (TMOD1, TMOD2) as follows.

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

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #050H
MOV     TMOD1, XA       ; Sets the modulo (for high-order 8 bits).
MOV     XA, #00BH
MOV     TMOD2, XA       ; Sets the modulo (for low-order 8 bits).
MOV     XA, #00100010B
MOV     TM1, XA         ; Sets the mode.
MOV     XA, #01001110B
MOV     TM2, XA         ; Sets the mode and starts the timer.
DI      IET1            ; Disables the timer (channel 1) interrupts.
EI                      ; Enables the interrupts.
EI      IET2            ; Enables the timer (channel 2) interrupts.
```

Remark In this example, TI0, TI1, and TI2 can be used as the input pins.

(b) When the pulse (input from the TI2 pin) count reaches 1000, the interrupts are generated (The pulses are active high).

- Set the high-order 4 bits of the mode register (TM1) to 0010B and select the overflow of the count register (T2).
- Set the high-order 4 bits of the TM2 to 0000B and select the rising edge of the TI2 input.
- Set the low-order 4 bits of the TM1 to 0010B and select the 16-bit timer/event counter mode.
- Set the low-order 4 bits of the TM2 to 1110B and select the 16-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo registers (TMOD1, TMOD2) to 1000 – 1 = 999 = 03E7H. Set the TMOD1 to 03H. Set the TMOD2 to E7H.

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #003H
MOV     TMOD1, XA       ; Sets the modulo (for high-order 8 bits).
MOV     XA, #0E7H
MOV     TMOD2, XA       ; Sets the modulo (for low-order 8 bits).
MOV     XA, #00100010B
MOV     TM1, XA         ; Sets the mode.
MOV     XA, #00001110B
MOV     TM2, XA         ; Sets the mode and starts the timer.
DI      IET1            ; Disables the timer (channel 1) interrupts.
EI
EI      IET2            ; Enables the timer (channel 2) interrupts.
```

Remark In this example, TI1 and TI2 can be used as the input pins.

(c) Following the 121 μs count disabled period, it can be used as an event counter with the sampling time (15 ms) and hold time (2 ms) (@ f_X = 4.19 MHz).
The timer/event counter (channel 0) can be set as follows.

- Set the high-order 4 bits of the mode register (TM0) to 0101B and select the longest setup time 15.6 ms.
- Set the low-order 4 bits of the TM0 to 1100B and select the 8-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo register (TMOD0) to 01H (121 μs) initially and then 20H (15.03 ms) and F5H (2.02 ms).

The timer/event counter (channel 1) is set as follows:

- Set the high-order 4 bits of the TM1 to 0010B and select the overflow of the count register (T2).
- Set the low-order 4 bits of the TM0 to 0010B and select the 16-bit timer/event counter mode and count operation and indicate timer start.
- Set the TMOD1 to the longest setup value FFH.
- Designate the memory MEM1 that stores the contents of the count register (T1).

The timer/event counter (channel 2) is set as follows.

- Set the high-order 4 bits of the TM2 to 0000B and select the rising edge of the TI2.
- Set the low-order 4 bits of the TM2 to 1110B and select the 16-bit timer/event counter mode and count operation and indicate counter clear.
- Set the TGCE to “1” and enable the gate control.
- Set the TMOD2 to the longest setup value FFH.
- Designate the memory MEM2 that stores the contents of the count register (T2).

<Program example>

```
MAIN:   SEL     MB15            ; or CLR1 MBE
        SET1    TGCE            ; Enables the gate control.
        MOV     XA, #00100010B
        MOV     TM1, XA         ; Sets the mode.
        MOV     XA, #00001110B
        MOV     TM2, XA         ; Sets the mode and clears the counter.
        MOV     XA, #001H
        MOV     TMOD0, XA       ; Sets the modulo (the initial count disabled time).
        MOV     XA, #01011100B
        MOV     TM0, XA         ; Sets the mode and indicates timer start.
        MOV     B, #00H         ; Initialization
        EI                      ; Enables the interrupts.
        EI      IET0            ; Enables the timer (channel 0) interrupts.
```

```
; <Subroutine>
            INCS    B
            SKE     B, #02H
            BR      SAMP
  HOLD:     MOV     XA, #020H
            MOV     TMOD0, XA     ; Updates the modulo (2 ms).
            MOV     XA, T1
            MOV     MEM1, XA      ; Reads the counter (T1).
            MOV     XA, T2
            MOV     MEM2, XA      ; Reads the counter (T2).
            SET1    TM2.3         ; Clears the counter.
            MOV     B, #00H
            BR      END
  SAMP:     MOV     XA, #0F5H
            MOV     TMOD0, XA     ; Updates the modulo (15 ms).
  END:      RETI
```

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

When the sampling clock goes high, the count operation starts and, at the same time, the initial interrupt is generated. The TMOD0 is updated to F5H and then the count operation continues for 15 ms.
When the sampling clock goes low, the count operation stops and, at the same time, the second interrupt is generated. The TMOD0 is updated to 20H and then the count operation is disabled for 2 ms. The contents of the T1 and T2 are read out and then they are cleared for the next count operation.
Then, the above operation is repeated.

5.5.5 Carrier generator mode (CG mode) operation

It is used as an 8-bit carrier generator in this mode.

When using this mode, it is used in combination with channel 1 and channel 2 of the timer/event counter.

The timer/event counter (channel 1) generates remote control signals.

The timer/event counter (channel 2) generates carrier clocks.

(1) Register setting

In the CG mode, the following eight registers are used.

- Timer/event counter mode registers (TM1, TM2)
- Timer/event counter control register (TC2)**Note**
- Timer/event counter count registers (T1, T2)
- Timer/event counter modulo registers (TMOD1, TMOD2)
- Timer/event counter high-level period setting modulo register (TMOD2H)

Note The channel 1 of the timer/event counter uses the timer/event counter output enable flag (TOE1).

(a) Timer/event counter mode register (TM1, TM2)

When using the CG mode, set the TM1 and TM2 as shown in Figure 5-53 (For the format of the TM1, see **Figure 5-32 Timer/Event Counter Mode Register (Channel 1) Format**. For the format of TM2, see **Figure 5-33 Timer/Event Counter Mode Register (Channel 2) Format**).

The TM1 and TM2 are manipulated by the 8-bit manipulation instructions. Bit 3 is a timer start indication bit and can be manipulated bit-wise and is automatically cleared to 0 when the timer starts operation.

The TM1 and TM2 are cleared to 00H when the internal reset signals are generated.

Figure 5-53. Timer/Event Counter Mode Register Setup (n = 1, 2)

Address	7	6	5	4	3	2	1	0	Symbol
FA8H	—	TM16	TM15	TM14	TM13	TM12	TM11	TM10	TM1
F90H	—	TM26	TM25	TM24	TM23	TM22	TM21	TM20	TM2

Count pulse (CP) selection bit

TMn6	TMn5	TMn4	TM1	TM2
0	0	0	TI1 rising edge	TI2 rising edge
0	0	1	TI1 falling edge	TI2 falling edge
0	1	0	Carrier clock input	$f_X/2$
0	1	1	$f_X/2^5$	f_X
1	0	0	$f_X/2^{12}$	$f_X/2^{10}$
1	0	1	$f_X/2^{10}$	$f_X/2^8$
1	1	0	$f_X/2^8$	$f_X/2^6$
1	1	1	$f_X/2^6$	$f_X/2^4$

Timer start indication bit

TMn3	When “1” is written into the bit, the counter and IRQTn flag are cleared. If bit 2 is set to “1”, count operation is started.

Operation mode

TMn2	Count Operation
0	Stop (retention of count contents)
1	Count operation

Opetation mode selection bit

TM21	TM20	TM11	TM10	Mode
1	1	0	0	Carrier generator mode

(b) Timer/event counter control register (TC2)

When using the CG mode, set the timer output enable flag (TOE1) and TC2 as shown in Figures 5-54 and 5-55 (For the format of the TC2, see **Figure 5-35 Timer/Event Counter Control Register Format)**.

The TOE1 is manipulated by a bit manipulation instruction. The TC2 is manipulated by an 8-bit/4-bit manipulation instruction and bit manipulation instruction.

The TOE1 and TC2 are cleared to 00H by the internal reset signal generation.

The flag indicated by the full lines expresses a flag used in the CG mode.

The flag indicated by the broken lines must not be used in the CG mode (Set “0”).

Figure 5-54. Timer/Event Counter Output Enable Flag Setup

Address

FAAH | TOE1 |

Timer/event counter output enable flag (W)

0	Disabled.
1	Enabled.

Figure 5-55. Timer/Event Counter Control Register Setup

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

Remote control output control flag

REMC	Remote Control Output
0	Outputs the carrier pulse when NRZ = 1.
1	Outputs the high-level signal when NRZ = 1.

No return zero buffer flag

NRZB	No return zero data to be output next. Transferred to the NRZ when a timer/event counter (channel 1) interrupt is generated.

No return zero flag

NRZ	No Return Zero Data
0	Outputs the low-level signal.
1	Outputs the carrier pulse or high-level signal.

(2) Carrier generator operation

The carrier generator operates as follows. Figure 5-56 shows its configuration.

(a) Timer/event counter (channel 1) operation

The timer/event counter (channel 1) determines the reloading interval from the no return zero buffer flag (NRZB) to the no return zero flag (NRZ). The timer/event counter (channel 1) operates as follows (For details, see **5.5.2 8-bit timer/event counter mode operation**).

<1> When the mode register (TM1) is set, the count pulse (CP) is set and is input to the count register (T1).

<2> The contents of the T1 and those of the modulo register (TMOD1) are compared, and if they are equal, a match signal is generated and the interrupt request flag (IRQT1) is set. At the same time, the timer out flip-flop (TOUT F/F) flips.

(b) Timer/event counter (channel 2) operation

The timer/event counter (channel 2) generates the carrier clock and outputs the carrier signal according to the no return zero data. The timer/event counter (channel 2) operates as follows (For details, see **5.5.3 PWM pulse generator mode (PWM mode) operation**).

<1> When the mode register (TM2) is set, the count pulse (CP) is selected and is input to the count register (T2).

<2> The contents of the T2 and those of the high-level period setting modulo register (TMOD2H) are compared, and if they are equal, a match signal is generated and the timer out flip/flop (TOUT F/F) flips.

<3> The contents of the T2 and those of the modulo register (TMOD2) are compared, and if they are equal, a match signal is generated and the interrupt request flag (IRQT2) is set. At the same time, the TOUT F/F flips.

<4> Repeat the above operations <2> and <3>.

<5> The no return zero data is reloaded from the NRZB to the NRZ when an interrupt is generated in the timer/event counter (channel 1).

<6> When the remote control output control flag (REMC) is set and NRZ = 1, the carrier clock signal or high-level signal is output. When NRZ = 0, a low-level signal is output.

Figure 5-57 shows the timing chart of the carrier generator.

The carrier generator normally operates in the following procedure.

<1> Set the number of count of the carrier clock's high-level signals in the TMOD2H.
<2> Set the number of count of the carrier clock's low-level signals in the TMOD2.
<3> Set the style of the output waveform in the REMC.
<4> Set the operation mode, count pulse, and start indication in the TM2.
<5> Set the number of count in the TMOD1.
<6> Set the operation mode, count pulse, and start indication in the TM1.
<7> Set the next no return zero data in the NRZB at any time before an interrupt is generated in the timer/event counter (channel 1).

Caution Set the values other than 00H in the modulo registers (TMOD1, TMOD2, TMOD2H).

When using the timer/event counter output pin (PTO1), set the alternate function pin P21 as follows.

<1> Clear the output latch of P21.
<2> Set port 2 to the output mode.
<3> Make a status wherein the on-chip pull-up resistor in port 2 is not incorporated.
<4> Set the timer/event counter output enable flag (TOE1) to 1.

Figure 5-56. Carrier Generator Operation Configuration

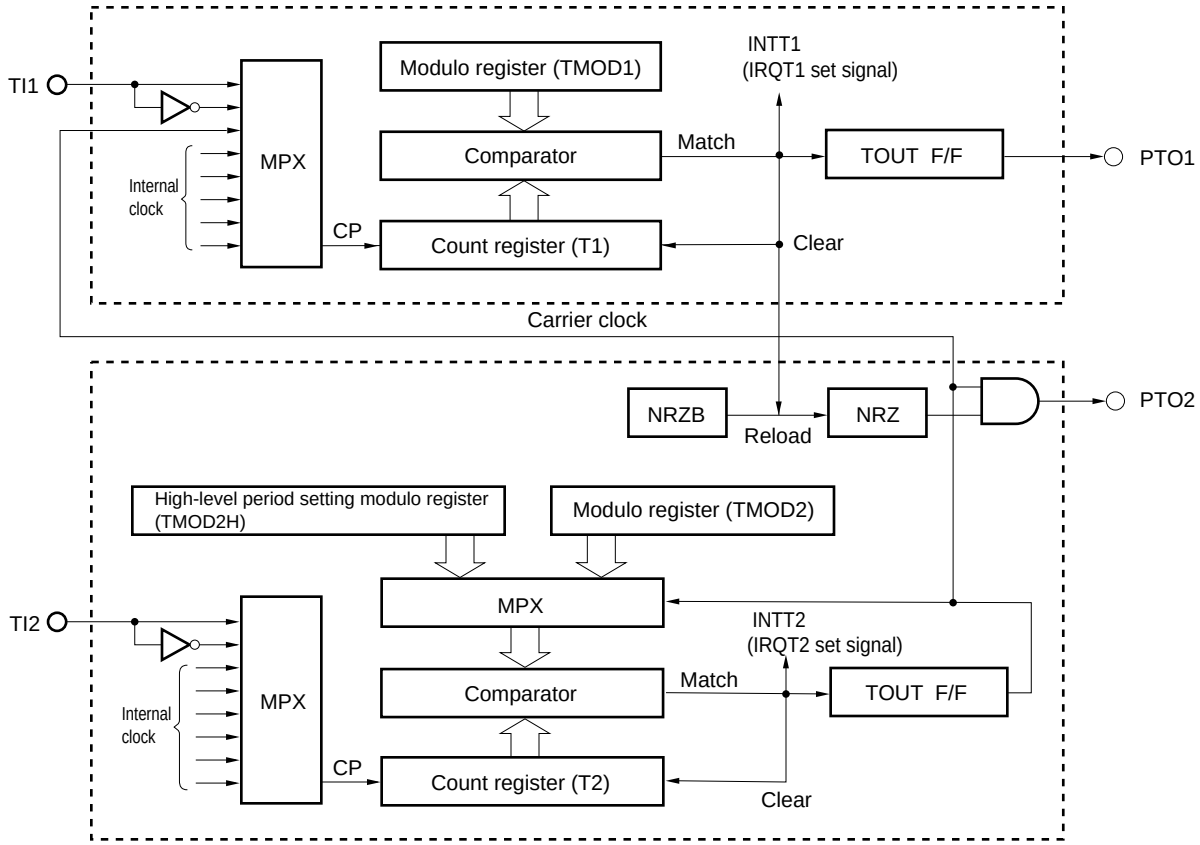
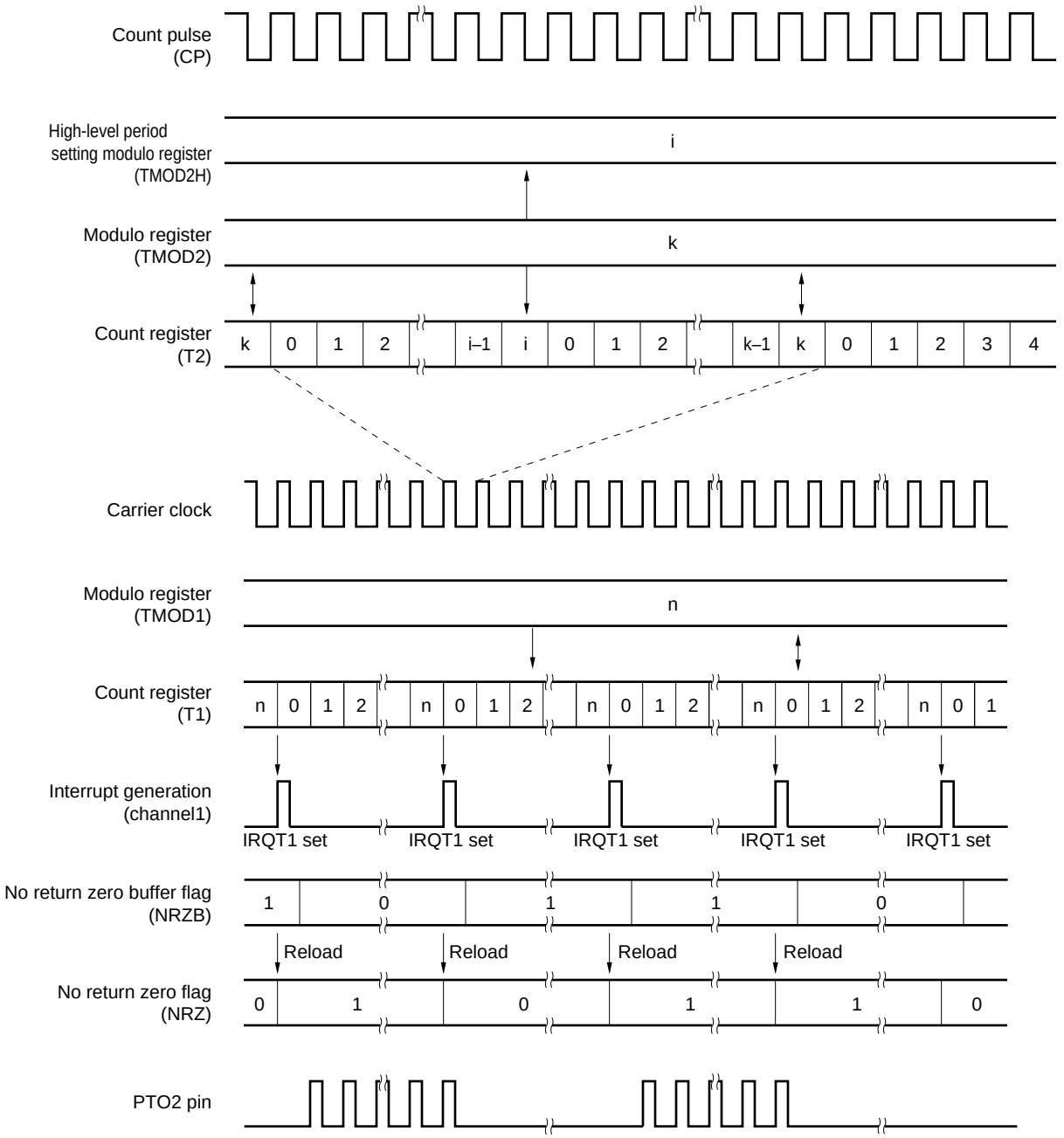
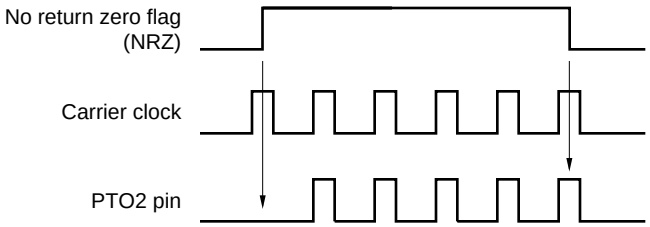


Figure 5-57. Carrier Generator Operation Timing



Remark i : Setup value of high-level period setting modulo register (TMOD2H)
k : Modulo register (TMOD2) setup value
n : Modulo register (TMOD1) setup value

Remark When the PTO2 pin is high (the no return zero flag (NRZ) is "0" and the carrier clock signal is high) and a timer/event counter (channel 1) interrupt is generated, the PTO2 pin output does not change according to the updated contents of NRZ until the carrier clock signal goes high. When the PTO2 pin is high (the NRZ is "1" and the carrier clock is high) and a timer/event counter (channel 1) interrupt is generated, the PTO2 pin output does not change according to the updated contents of NRZ until the carrier clock signal goes low.
This is to keep a certain high-level pulse width of the output carrier (see the figure below).



Even if the NRZ is set to "1", the PTO2 pin output does not go high until the next carrier clock pulse goes high.

Even if the NRZ is set to "0", the PTO2 pin output does not go low until the next carrier clock pulse goes low.

(3) CG mode applications

It can be used as a carrier generator for remote control transmission.

(a) A carrier clock signal (frequency is 38.0 kHz (cycle: 26.3 μs) and duty ratio is 1/3) is generated (@ f_X = 4.19 MHz).

- Set the high-order 4-bits of the mode register (TM2) to 0011B and select the longest setup time 61.1 μs.
- Set the low-order 4-bits of the TM2 to 1111B and select the CG mode and count operation and indicate timer start.
- Set the timer output enable flag (TOE2) to "1" and enable timer output.
- Set the high-level period setting modulo register (TMOD2H) to the following value.

$$\frac{1}{3}\times\frac{26.3\ \mu s}{239\ ns}-1=36.7-1\cong 36=24H$$

- Set the modulo register (TMOD2) to the following value.

$$\frac{2}{3}\times\frac{26.3\ \mu s}{239\ ns}-1=73.4-1\cong 72=48H$$

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #024H
MOV     TMOD2H, XA      ; Sets the modulo (high-level period).
MOV     XA, #48H
MOV     TMOD2, XA       ; Sets the modulo (low-level period).
MOV     XA, #00111111B
MOV     TM2, XA         ; Sets the mode and starts the timer.
```

(b) A reader code (carrier clock output period is 9 ms and low-level output period is 4.5 ms) is output. See the illustration below (@ f_X = 4.19 MHz).

- Set the high-order 4 bits of the mode register (TM1) to 0110B and select the longest setup time 15.6 ms.
- Set the low-order 4 bits of the TM1 to 1100B and select the 8-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo register (TMOD1) to the following initial value.

$$\frac{9\ \text{ms}}{61\ \mu\text{s}} - 1 = 147.5 - 1 \cong 146 = 92\text{H}$$

- Set the TMOD1 as follows when update it.

$$\frac{4.5\ \text{ms}}{61\ \mu\text{s}} - 1 = 73.7 - 1 \cong 73 = 49\text{H}$$

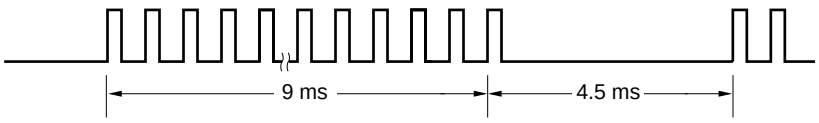
- Set the high-order 4 bits of the TC2 to 0000B and disable the gate control.
- Set the low-order 4 bits of the TC2 to 0000B and output the carrier clock signal when no return zero data is “1” and set the next no return zero data to “0”.

<Program example>

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #092H
MOV     TMOD1, XA       ; Sets the modulo (carrier clock output period).
MOV     XA, #00000000B
MOV     TC2, XA
SET1    NRZ             ; Sets the no return zero data to “1”.
MOV     XA, #01101100B
MOV     TM1, XA         ; Sets the mode and starts the timer.
EI                      ; Enables the interrupts.
EI      IET1            ; Enables the timer (channel 1) interrupts.
```

; <Subroutine>

```
MOV     XA, #049H
MOV     TMOD1, XA       ; Updates the modulo (low-level output period).
RETI
```

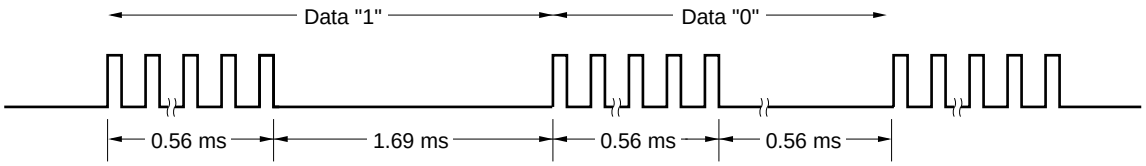


(c) A custom code (the carrier clock output period is 0.56 ms and low-level output period is 1.69 ms when data is “1”, and carrier clock output period is 0.56 ms and low-level output period is 0.56 ms when data is “0”) is output. See the illustration below (@ f_X = 4.19 MHz).

- Set the high-order 4 bits of the mode register (TM1) to 0011B and select the longest setup time 1.95 ms.
- Set the low-order 4 bits of the TM1 to 1100B and select the 8-bit timer/event counter mode and count operation and indicate timer start.
- Set the modulo register (TMOD1) to the following initial value.

$$\frac{0.56\ \text{ms}}{7.64\ \mu\text{s}} - 1 = 73.3 - 1 \cong 72 = 48\text{H}$$

- When data is “0” during the period in which the carrier output is not specified by TMOD1, the processing is performed for the same duration as the output period. When data is “1”, the processing time is three times longer than the output period.
- Set the high-order 4 bits of the TC2 to 0000B and disable the gate control.
- Set the low-order 4 bits of the TC2 to 0000B, and if no return zero data is “1”, output the carrier clock and set the next no return zero data to “0”.
- Set the transmit data (“0” or “1”) in the bit sequential buffer.



<Program example>

In this example, it is assumed that the output latch of the PTO2 pin is fixed to "0" and the pin is set in the output mode. It is also assumed that the carrier clock is being generated by the program in <2> above.

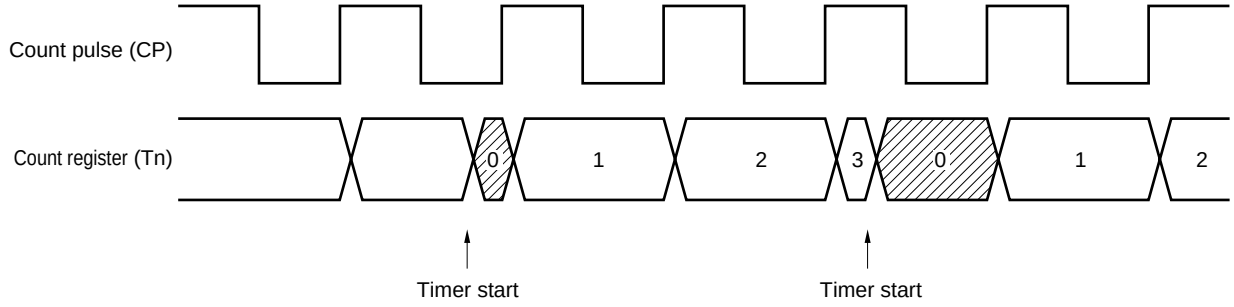
```
; SEND_CARIER_DATA_PRO
            SEL     MB15            ; or CLR1 MBE
            MOV     HL, #00H        ; Sets pointer of BSB (bit sequential buffer) to L. H is
                                      used to save bit data of BSB temporarily.

; CG_Init & Send_1st_Data
            MOV     XA, #48H
            MOV     TMOD1, XA       ; Sets modulo register (carrier clock output period).
            MOV     XA, #00000000B  ; Disables gate control, enables output of carrier clock,
                                      and initializes NRZB and NRZ to 0.
            MOV     TC2, XA
            SET1    NRZ             ; Sets "1" to no return zero flag.
            MOV     XA, #01101100B  ; Selects count pulse, and sets 8-bit timer/event counter
                                      mode.
            MOV     TM1, XA         ; Enables timer/event count operation, and starts timer.

; Send_1st_Data
            CALL    !GET_DAT        ; Gets data from BSB
            CALL    !SEND_D_0       ; Outputs carrier with data 0 and 1, and sets low-level
                                      output period once.
            SKE     H, #1H          ; If bit 0 is 1, adds low-level output period twice.
            BR      SEND_1_F        ; If bit 0 is 0, searches next data with low-level output.
            CALL    !SEND_D_1       ; Adds two low-level output periods.
                                      Transmits data of bits 0 to F of BSB with PTO2 pin
                                      outputting low-level.

SEND_1_F:                           ; Transmits data of bits 0 to F of BSB.
            SET1    NRZB            ; Sets NRZB to 1 during low-level output period of
                                      previous data so that carrier of data transmitted next
                                      is output on next generation of IRQT1.
            INCS    L               ; Counts transmit data. If L changes from 0FH to 0H,
                                      ends data transmission.
            BR      LOOP_C_0
            BR      SEND_END
```

```
LOOP_C_0: SKTCLR  IRQT1          ; Waits low-level output of previous data (acknowledges
                                   end of previous data).
          BR      LOOP_C_0
                                 ; Starts carrier output.
          CLR1    NRZB           ; Clears NRZB to 0 in advance so that first low-level
                                   output is performed on next generation of IRQT1.
          CALL    !GET_DAT
          CALL    !SEND_D_0
          SKE     H, #1H         ; If data gotten is 1, adds low-level output period twice
                                   (SEND_D_1).
          BR      SEND_1_F       ; If data is 0, transmits next data with PTO2 pin output-
                                   ting low-level.
          CALL    !SEND_D_1
          BR      SEND_1_F
SEND_END:                        ; End of transmitting 16 bits of data.

; <Subroutine>
GET_DAT:                         ; Searches data of BSB indicated by @L.  Sets value to
                                   H register.
          SKT     BSB0, @L
          MOV     A, #0
          MOV     A, #1
          MOV     H, A
          RET

SEND_D_0:                        ; Outputs carrier with data 0 and 1 and sets low-level
                                   output once.
LOOP_1st: SKTCLR IRQT1
          BR      LOOP_1st       ; Waits carrier output.
          RET                    ; Starts first low-level output.
SEND_D_1:
          CLR1    NRZB           ; If data is 1, sets second low-level output.
LOOP_2nd: SKTCLR IRQT1
          BR      LOOP_2nd       ; Waits first low-level output.
                                 ; Starts second low-level output.
          CLR1    NRZB           ; Sets third low-level output.
LOOP_3rd: SKTCLR  IRQT1
          BR      LOOP_3rd       ; Waits second low-level output.
                                 ; Starts third low-level output.
          RET
```

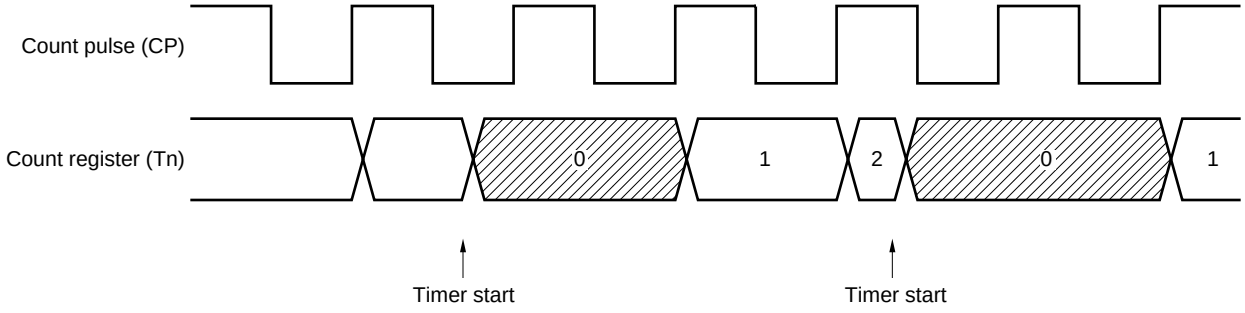
5.5.6 Notes on using timer/event counter

(1) Error when starting the timer

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



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



(2) Caution on starting the timer

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

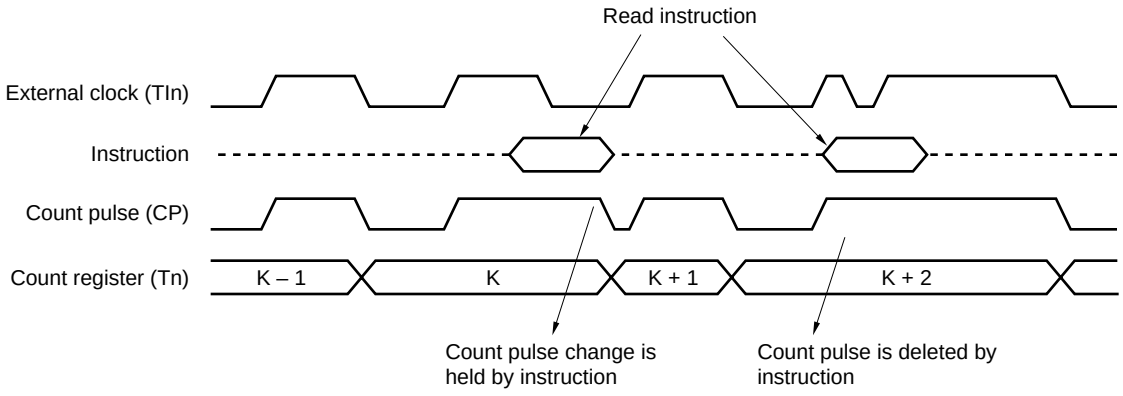
Example Timer start at the timing when the IRQTn may be set high

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

(3) Error when reading the count register

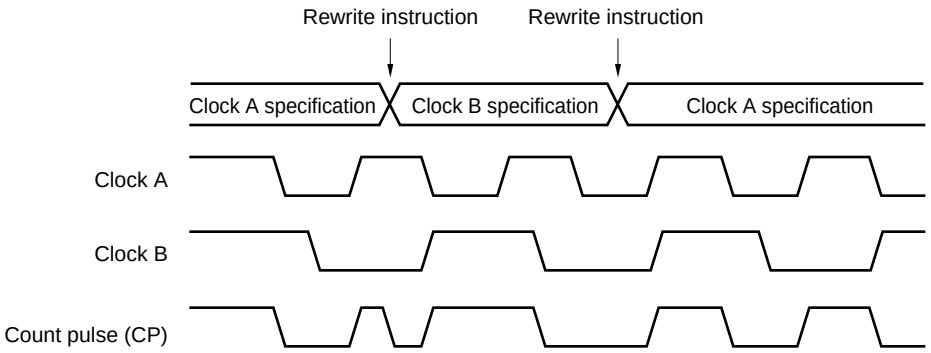
The count register (Tn) can be read any time by an 8-bit data memory manipulation instruction. When the instruction is being executed, the count pulse (CP) does not change and the contents of the Tn are kept unchanged. When the power supply for the CP is input from the TIn, the CP is cut during the instruction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

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

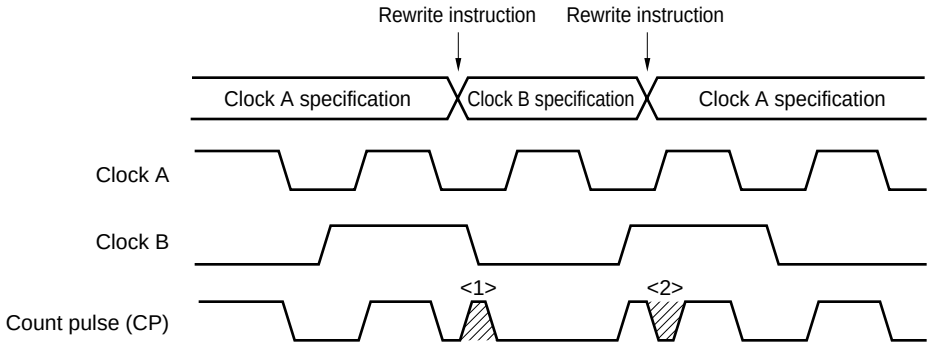


(4) Caution on changing the count pulse

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

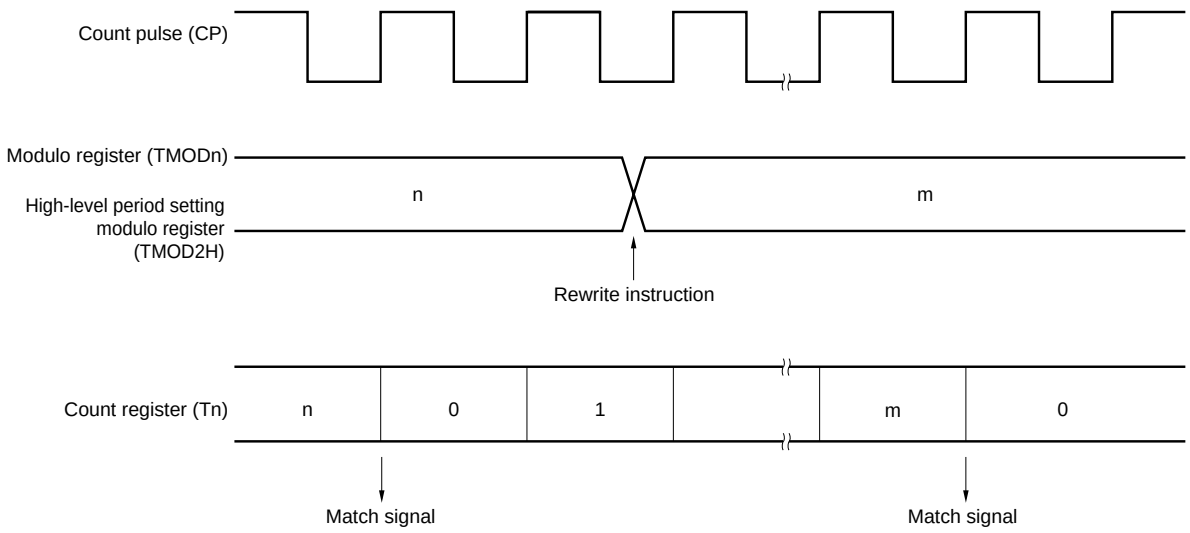


Depending on the combination of clock pulses at the time the CP is changed, whisker-like clock pulses (<1> or <2> in the illustration below) may be produced. In this case, incorrect counting may occur and the count register (Tn) may be disrupted. Therefore, when changing the CP, set bit 3 of the TMn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

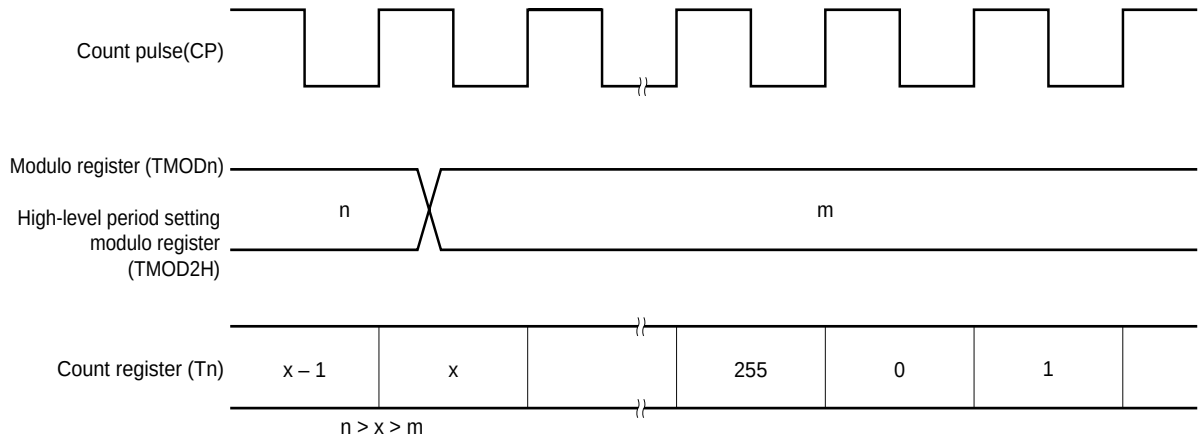


(5) Operation after changing the modulo register

The contents of the modulo register (TMODn) and high-level period setting modulo register (TMOD2H) are rewritten by an 8-bit data memory manipulation instruction.



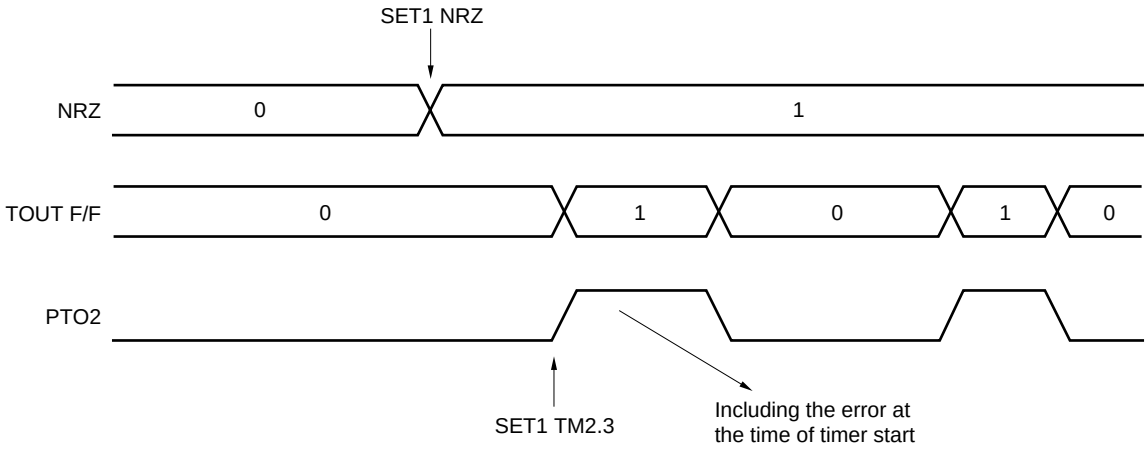
If the value of the TMODn after a change is smaller than the value of the count register (Tn), the Tn continues counting and overflows to restart counting from 0. Therefore, if the value (m) after the TMODn and TMOD2H are changed is smaller than the value (n) before they are changed, the timer must restart after they are changed.



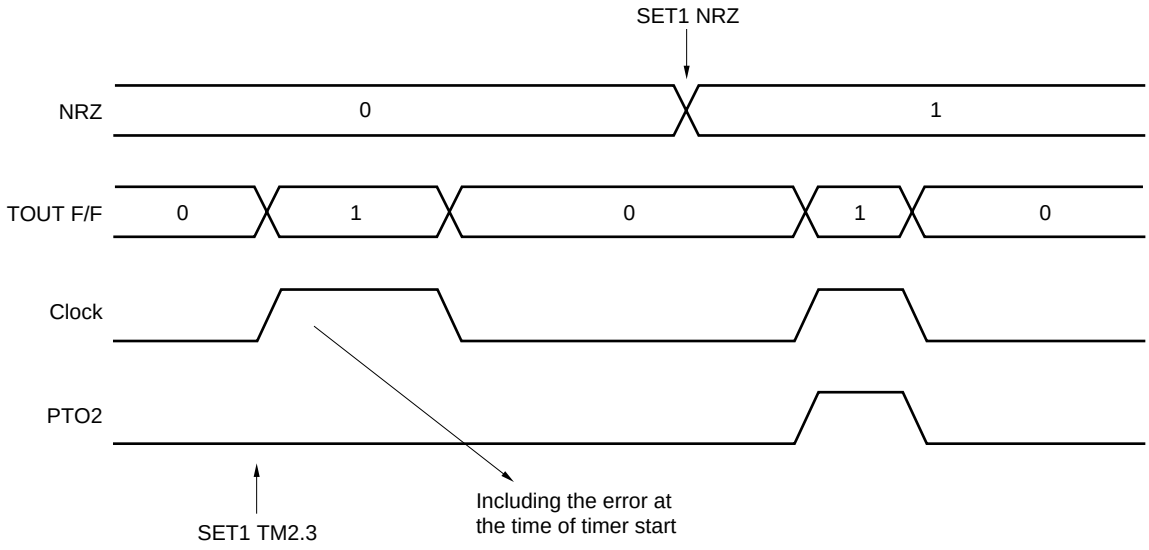
(6) Caution on using the carrier generator (at the start)

When a carrier clock is generated, an error of one count pulse (CP) at maximum (two clock pulses at maximum when the CP frequency is one machine cycle or more) is produced with respect to the value obtained by the formula ((modulo register value + 1) × resolution) during the high-level period of the initial carrier clock after the timer starts (the bit 3 of the TM2 is set to "1"). For details, refer to paragraph **(1) Error when starting the timer**.

When no return zero flag (NRZ) is set to "1" and then the timer starts (the bit 3 of the TM2 is set to "1") in case a carrier is output as the initial code, the error at the time of timer start is contained during the time the initial carrier clock is high.



Therefore, when a carrier is to be output as the initial code, start the timer (set bit 3 of the TM2 to "1") and then set NRZ to "1".

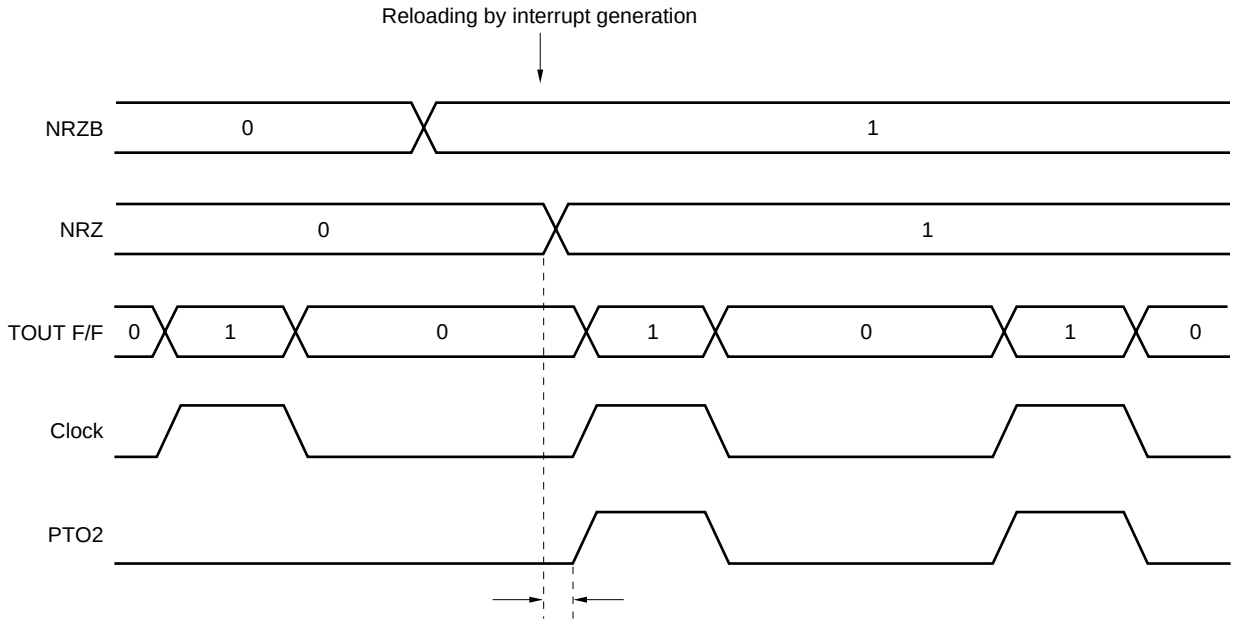


(7) Caution on using the carrier generator (when reloading)

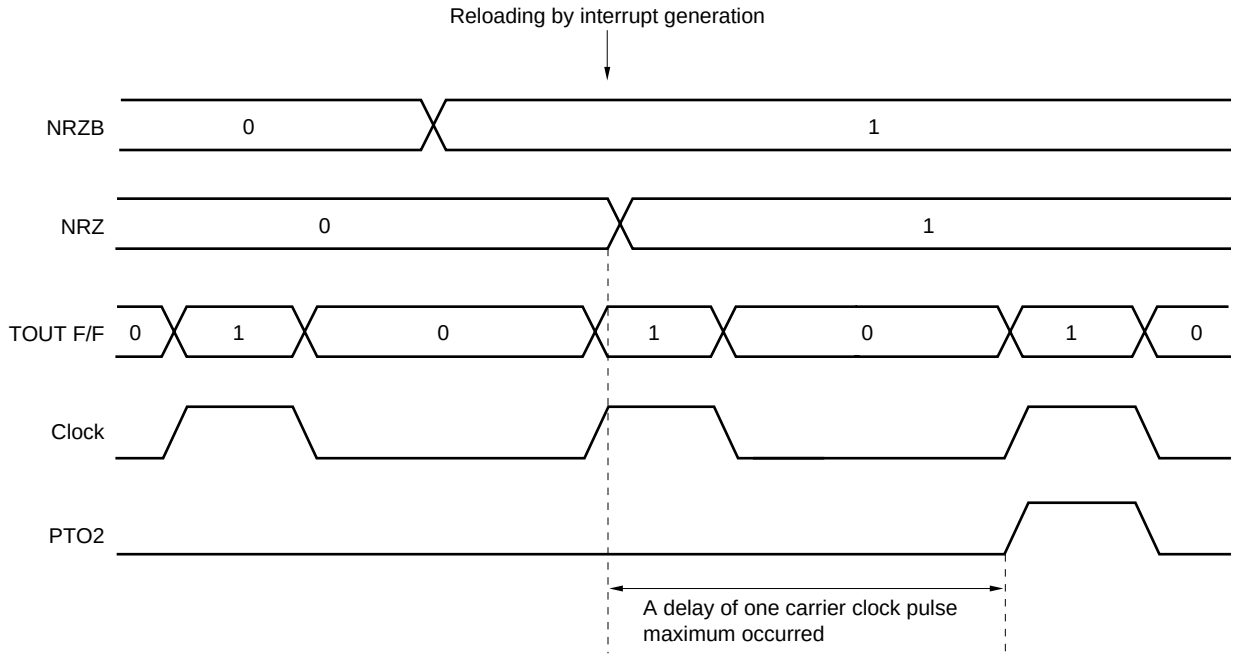
When a carrier is output to the PTO2 pin, a delay of one carrier clock pulse at maximum occurs during the time from the reloading (the contents of the no return zero buffer flag (NRZB) are transferred to the no return zero flag (NRZ) by a timer/event counter (channel 1) interrupt generation and the contents of the NRZ are updated to “1”) to the initial carrier generation.

This is because reloading is done asynchronously with the carrier clock and for keeping the carrier at the stable high level.

<Delay after reloading is at minimum>

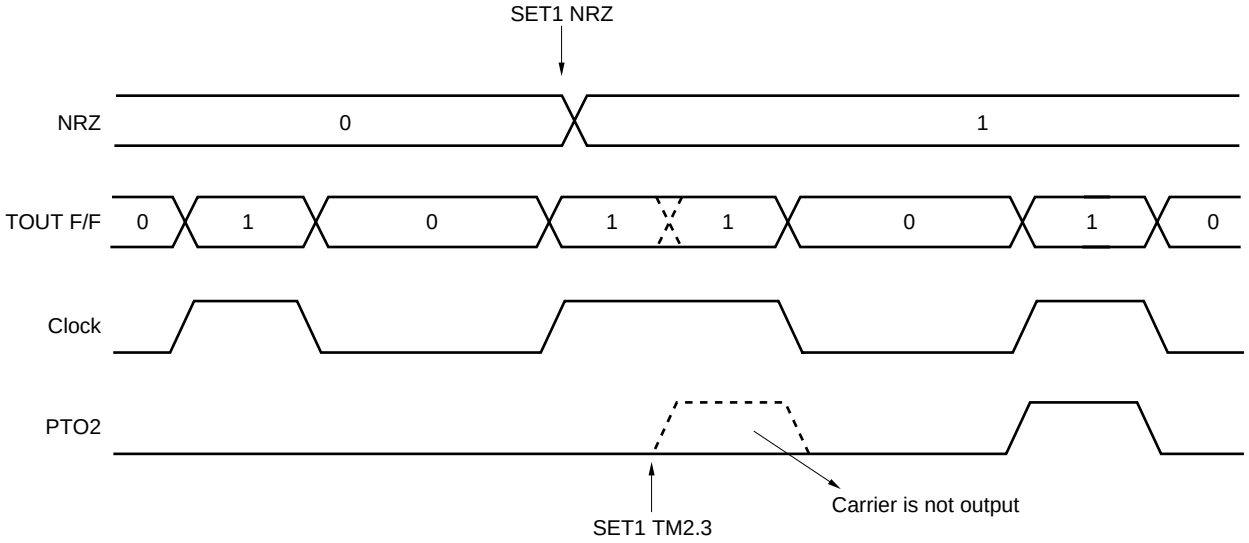


<Delay after reloading is at maximum>

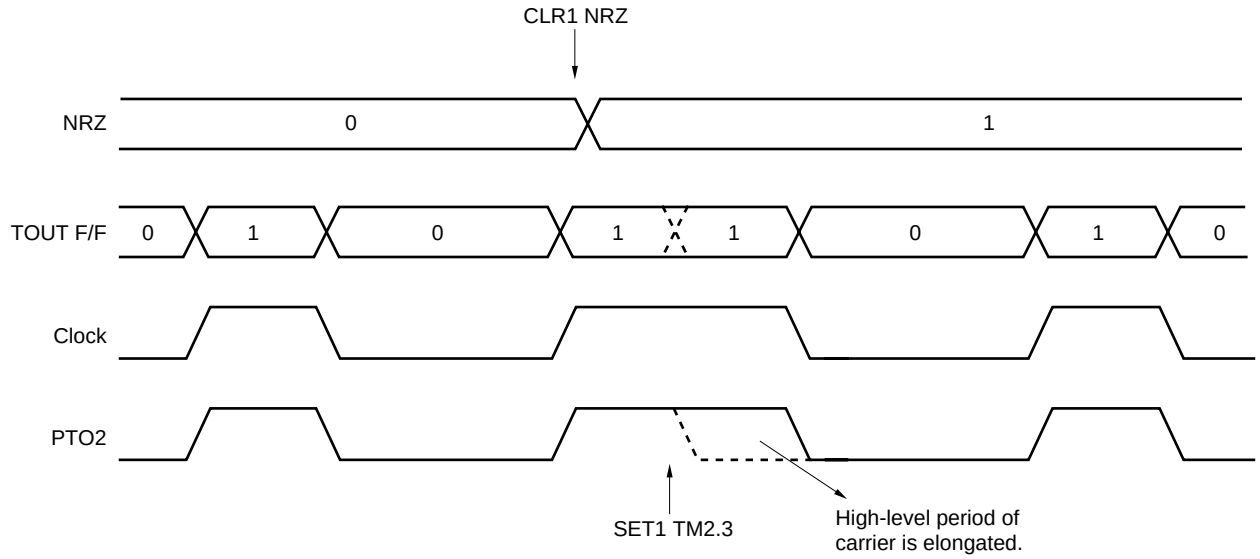


(8) Caution on using the carrier generator (when restarting)

When the carrier clock is high (TOUT F/F is holding “1”) and reloading is to be done forcibly by directly rewriting the contents of no return zero flag (NRZ) and the timer is to restart (by setting bit 3 of TM2 to “1”), the carrier may not be output to the PTO2 pin as shown below.



Similarly to the above, when carrier clock is high (TOUT F/F is holding “1”) and reloading is to be done forcibly by directly rewriting the contents of NRZ and the timer is to restart (by setting bit 3 of TM2 to “1”), the carrier output to the PTO2 pin may keep its high level for a longer time as shown below.



5.6 Serial Interface

5.6.1 Serial interface function

The μPD753108 incorporates a clock-synchronous 8-bit serial interface consisting of the following four modes.

(1) Operation stop mode

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

(2) 3-wire serial I/O mode

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

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

Because the first bit of 8-bit data for serial transfer is switchable to MSB or LSB, the μPD753108 can be connected to any device regardless of whether its first bit is MSB or LSB.

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

(3) 2-wire serial I/O mode

This mode performs data transfer in 8-bit units with two lines: serial clock ($\overline{SCK}$), and serial data bus (SB0 or SB1).

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

The levels of output to $\overline{SCK}$ and SB0 (or SB1) can be controlled by software so that they can accept any data transfer.

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

(4) SBI mode (serial bus interface mode)

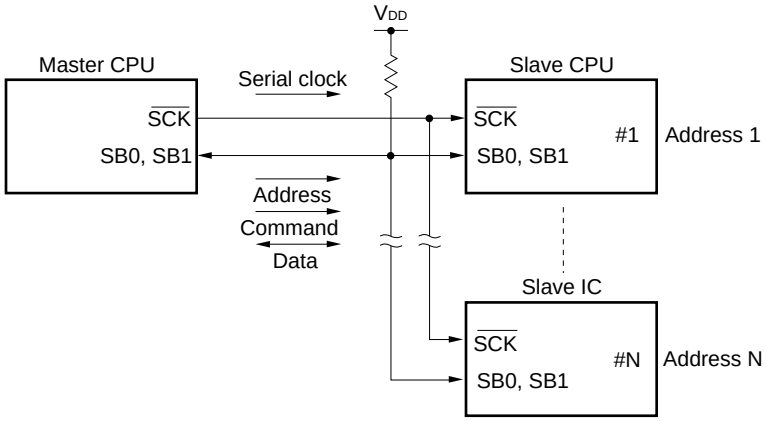
This mode enables communication to a number of devices with two lines (serial clock ($\overline{SCK}$) and serial data bus (SB0 or SB1)) and is compliant with the NEC serial bus format.

The transmitter can output an “address” for selecting the target device for serial communication, a “command” to direct to the target device, and actual “data” to the serial data bus.

The receiver can identify the data as an “address”, “command”, and “data” by hardware.

This function allows efficient use of the I/O port as in 2-wire serial I/O mode, and moreover, enables simpler serial interface controller for application programs.

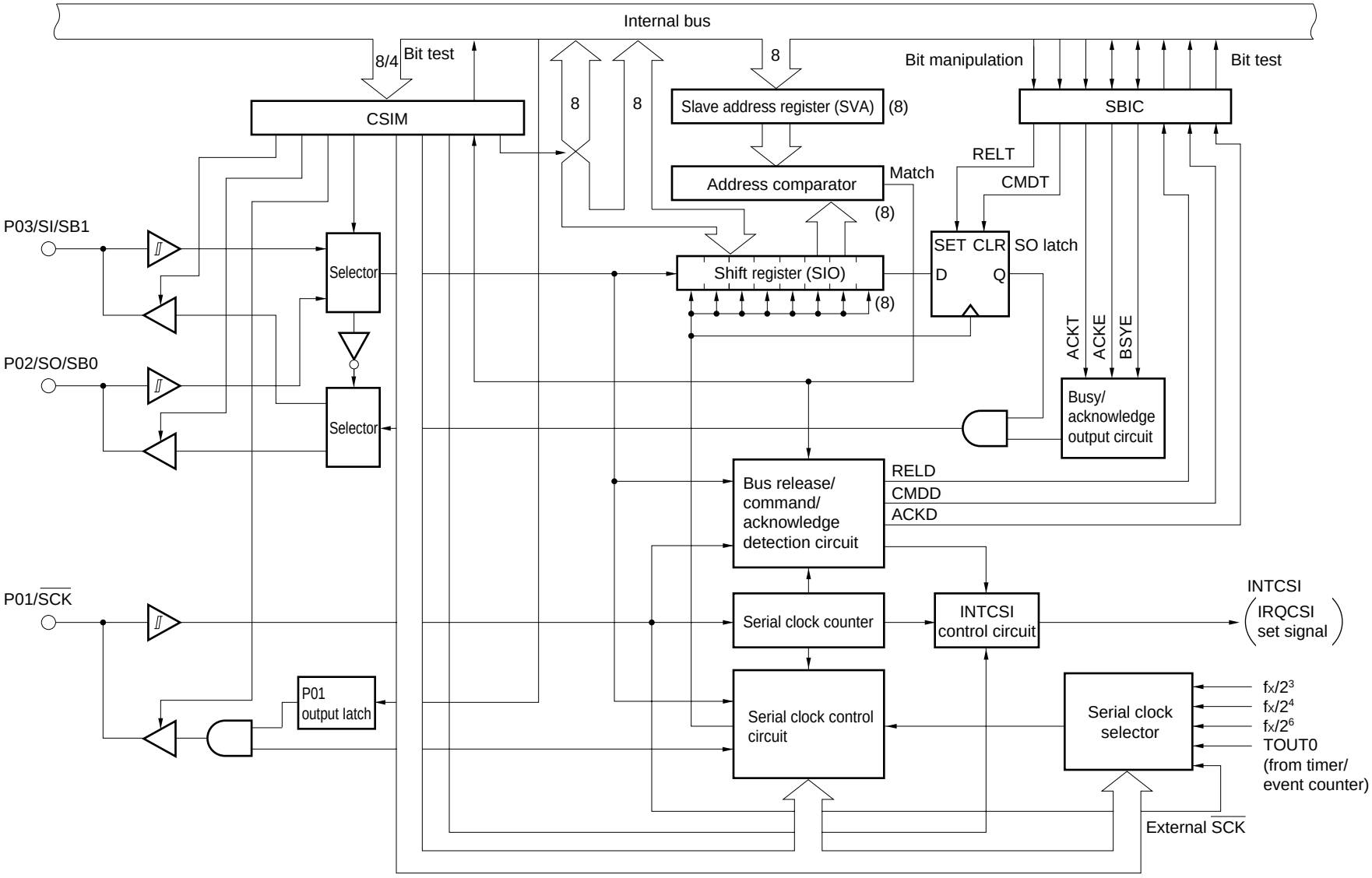
Figure 5-58. Example of SBI System Configuration



5.6.2 Configuration of serial interface

Figure 5-59 shows a block diagram of the serial interface.

Figure 5-59. Serial Interface Block Diagram



(1) Serial operation mode register (CSIM)

This 8-bit register specifies the operation mode and serial clock wake-up function of the serial interface (For details, refer to **5.6.3 (1) Serial operation mode register (CSIM)**).

(2) Serial bus interface control register (SBIC)

This 8-bit register consists of bits that control the status of the serial bus and flags that indicate the various statuses of the data input from the serial bus. It is mainly used in the SBI mode (For details, refer to **5.6.3 (2) Serial bus interface control register (SBIC)**).

(3) Shift register (SIO)

This register converts 8-bit serial data into parallel data or 8-bit parallel data into serial data. It performs transmission or reception (shift operation) in synchronization with the serial clock. Actual transmission or reception is controlled by writing data to the SIO (For details, refer to **5.6.3 (3) Shift register (SIO)**).

(4) SO latch

This latch holds the levels of the SO/SB0 and SI/SB1 pins. It can also be controlled directly via software. In the SBI mode, this latch is set when $\overline{\text{SCK}}$ has been asserted eight times (For details, refer to **5.6.3 (2) Serial bus interface control register (SBIC)**).

(5) Serial clock selector

This selects the serial clock to be used.

(6) Serial clock counter

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

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

- **In SBI mode**

  This register and comparator are used when the μPD753108 is used as a slave device. The slave places its specification number (slave address value) in the SVA. The master outputs a slave address to select a specific slave.

  The address comparator of the slave compares the slave address the slave has received from the master with the value in the SVA. When the address coincides with the SVA value, the slave is selected.

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

  When the μPD753108 is used as a slave or master, this register and comparator detects an error (For details, refer to **5.6.3 (4) Slave address register (SVA)**).

(8) INTCSI control circuit

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

- **In 3-wire and 2-wire serial I/O modes**
  An interrupt request is generated each time eight serial clocks have been counted.

- **In SBI mode**
  When WUP[Note] = "0" ..... An interrupt request is generated each time eight serial clocks have been counted.
  When WUP = "1" ..... An interrupt request is generated when the value of SVA and that of SIO coincide after an address has been received.

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

(9) Serial clock control circuit

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

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

These circuits output and detect control signals in the SBI mode.
They do not operate in the three-wire and two-wire serial I/O modes.

(11) P01 output latch

This latch generates the serial clock via software after eight serial clocks have been generated.
It is set to "1" when the reset signal is input.
To select the internal system clock as the serial clock, set the P01 output latch to "1".

5.6.3 Register function

(1) Serial operation mode register (CSIM)

Figure 5-60 shows the format of the serial operation mode register (CSIM).

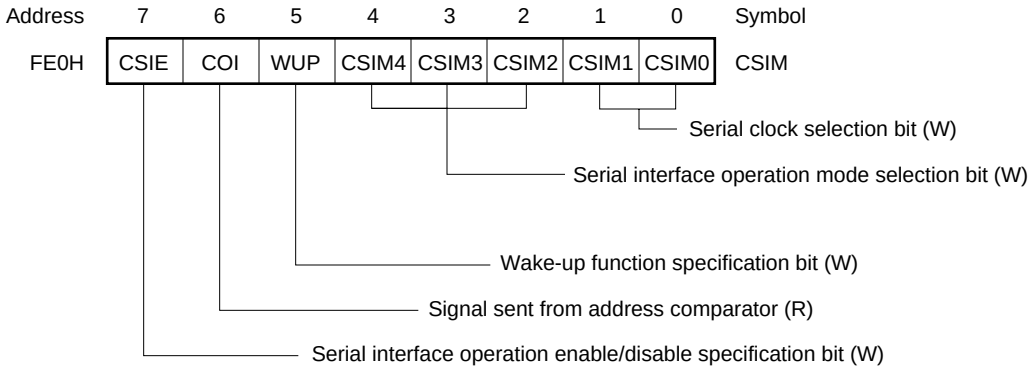
The CSIM is an 8-bit register which specifies the serial interface operation mode, serial clock, and wake-up function.

The register is manipulated by an 8-bit memory manipulation instruction. The high-order 3 bits of the register can be operated bit-wise. In this case, the name of each bit is used to manipulate.

Whether the read/write operation can be performed or not depends on the bit (See **Figure 5-60**). Bit 6 can be used only as a bit test and data written in the bit position is invalid.

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

Figure 5-60. Serial Operation Mode Register (CSIM) Format (1/4)



Remarks 1. (R) Read only.

2. (W) Write only.

Figure 5-60. Serial Operation Mode Register (CSIM) Format (2/4)

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0, SI/SB1 Pins
CSIE	0	Disable the shift operation.	Clear	Hold	Dedicated to port 0 function
	1	Enables the shift operation.	Count operation	Enable setting	Both for the function in each mode and port 0

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

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

2. The P01/$\overline{\text{SCK}}$ pin enters the following status by setting the CSIE, CSIM1, and CSIM0.

CSIE	CSIM1	CSIM0	Status of P01/$\overline{\text{SCK}}$ Pin
0	0	0	Input port (P01)
1	0	0	High-impedance ($\overline{\text{SCK}}$ input)
0	0	1	High-level output
0	1	0	
0	1	1	
1	0	1	Serial clock output (high-level output : at the end of serial data transfer)
1	1	0	
1	1	1	

3. During serial data transfer, follow the procedure (<1> to <3>) to clear the CSIE.

<1> Clear the interrupt enable flag (IECSI) and disable the interrupts.

<2> Clear the CSIE.

<3> Clear the interrupt request flag (IRQCSI).

Examples 1. $f_X/2^4$ is selected for the serial clock. A serial interrupt IRQCSI is generated at the end of serial data transfer. Serial data transfer is done in the SBI mode with the SB0 pin as the serial data bus.

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

2. Serial data transfer complying with the contents of the CSIM is enabled.

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

Figure 5-60. Serial Operation Mode Register (CSIM) Format (3/4)

Signal from address comparator (R)

COI[Note]	Condition To Be Cleared (COI = 0)	Condition To Be Set (COI = 1)
	The contents of the slave address register (SVA) are not the same as those of the shift register.	The contents of the slave address register (SVA) are the same as those of the shift register.

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

Wake-up function specification bit (W)

WUP	0	Sets the IRQCSI at each time serial data transfer ends in each mode.
	1	Used only in the SBI mode. Sets the IRQCSI only when an address received after a bus release is equal to the data of the slave address register (wake-up status). SB0 and SB1 are high-impedance buses.

Caution If WUP = 1 while a $\overline{\text{BUSY}}$ signal is output, $\overline{\text{BUSY}}$ is not released. The $\overline{\text{BUSY}}$ signal is output during the time from the release of $\overline{\text{BUSY}}$ to the falling edge of serial clock ($\overline{\text{SCK}}$) in the SBI. When making WUP = 1, be sure to release the $\overline{\text{BUSY}}$ status and then assure the SB0 (or SB1) pin is high.

Serial interface operation mode selection bit (W)

CSIM4	CSIM3	CSIM2	Operation Mode	Shift Register Bit Order	SO/SB0/P02 Pin Function	SI/SB1/P03 Pin Function
×	0	0	3-wire serial I/O mode	$SIO_{7-0} \leftrightarrow XA$ (transferred at MSB first)	SO (CMOS output)	SI (CMOS input)
		1		$SIO_{0-7} \leftrightarrow XA$ (transferred at LSB first)		
0	1	0	SBI mode	$SIO_{7-0} \leftrightarrow XA$ (transferred at MSB first)	SB0 (N-channel open-drain I/O)	P03 (CMOS input)
1					P02 (CMOS input)	SB1 (N-channel open-drain I/O)
0	1	1	2-wire serial I/O mode	$SIO_{7-0} \leftrightarrow XA$ (transferred at MSB first)	SB0 (N-channel open-drain I/O)	P03 (CMOS input)
1					P02 (CMOS input)	SB1 (N-channel open-drain I/O)

Remark × : don't care

Figure 5-60. Serial Operation Mode Register (CSIM) Format (4/4)

Serial clock selection bit (W)

CSIM1	CSIM0	Serial Clock			$\overline{SCK}$ Pin Mode
		3-wire Serial I/O Mode	SBI Mode	2-wire Serial I/O Mode	
0	0	Clock input from outside to $\overline{SCK}$ pin			Input
0	1	Timer/event counter output (TOUT0)			Output
1	0	$f_X/2^4$ (375 kHz: 6.00-MHz operation, 262 kHz: 4.19-MHz operation)		$f_X/2^6$ (93.8 kHz: 6.00-MHz operation, 65.5 kHz: 4.19-MHz operation)	
1	1	$f_X/2^3$ (750 kHz: 6.00-MHz operation, 524 kHz: 4.19-MHz operation)			

(2) Serial bus interface control register (SBIC)

Figure 5-61 shows the format of the serial bus interface control register (SBIC).

The SBIC is an 8-bit register which is composed of the bits controlling the serial bus and the flags indicating the status of input data. It is used mainly in the SBI mode.

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

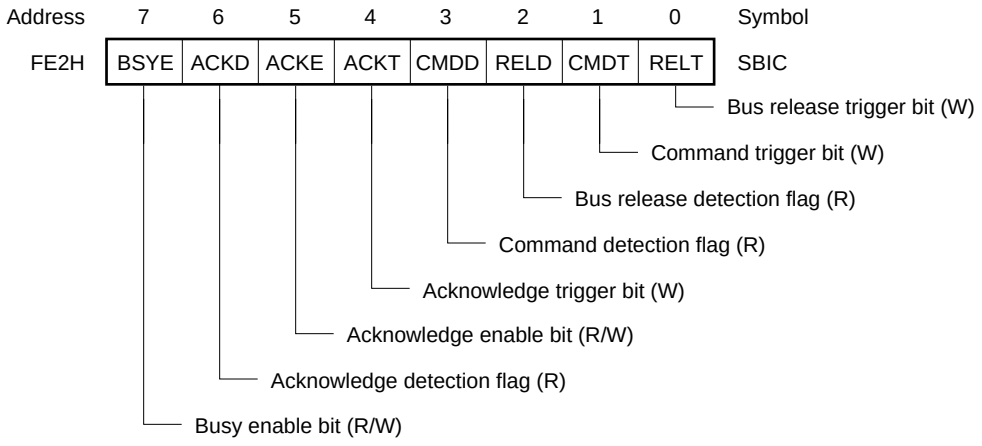
Whether the read/write operation can be performed or not depends on the bit (See **Figure 5-61**).

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

Caution Only the following bits can be used in the 3-wire and 2-wire serial I/O modes.

- **Bus release trigger bit (RELT) ..... sets the SO latch.**
- **Command trigger bit (CMDT) ..... clears the SO latch.**

Figure 5-61. Serial Bus Interface Control Register (SBIC) Format (1/3)



Remarks 1. (R) : Read only.

2. (W) : Write only.

3. (R/W) : Both read/write.

Figure 5-61. Serial Bus Interface Control Register (SBIC) Format (2/3)

Busy enable bit (R/W)

BSYE	0	<1> Disables the automatic output of a busy signal. <2> Stops the output of a busy signal in synchronization with the falling edge of the $\overline{SCK}$ immediately after a clear instruction is executed.
	1	Outputs a busy signal in synchronization with the falling edge of the $\overline{SCK}$ following an acknowledge signal.

Examples 1. A command signal is output.

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

2. The RELD and CMDD are tested to identify the received data for an appropriate operation. The interrupt routine sets the WUP to 1 and is executed only when the address is matched.

```
SEL     MB15
SKF     RELD     ; Tests the RELD.
BR      !ADRS
SKT     CMDD     ; Tests the CMDD.
BR      !DATA
BR      !CMD
CMD  :  ..... ; Interprets the command.
DATA :  ..... ; Processes the data.
ADRS :  ..... ; Decodes the address.
```

Acknowledge detection flag (R)

ACKD	Condition To Be Cleared (ACKD = 0)	Condition To Be Set (ACKD = 1)
	<1> At the start of data transfer <2> When the $\overline{RESET}$ signal is generated.	When acknowledge signal ($\overline{ACK}$) is detected (in synchronization with the rising edge of the $\overline{SCK}$).

Acknowledge enable bit (R/W)

ACKE	0	Disables the automatic output of the acknowledge signal ($\overline{ACK}$). Output by the ACKT is enabled.	
	1	When it is set before data transfer terminates	The $\overline{ACK}$ is output in synchronization with the 9th clock pulse of $\overline{SCK}$.
		When it is set after data transfer terminates	The $\overline{ACK}$ is output in synchronization with the $\overline{SCK}$ immediately after a set instruction is executed.

Acknowledge trigger bit (W)

ACKT	When it is set at the end of data transfer, the $\overline{ACK}$ is output in synchronization with the next $\overline{SCK}$. It is automatically cleared to 0 after the $\overline{ACK}$ is output.

Cautions 1. Do not set this bit to 1 before the end of serial transfer or during transfer.

2. The ACKT cannot be cleared by software.

3. When setting the ACKT, set the ACKE to 0.

Figure 5-61. Serial Bus Interface Control Register (SBIC) Format (3/3)

Command detection flag (R)

CMDD	Condition To Be Cleared (CMDD = 0)	Condition To Be Set (CMDD = 1)
	<1> When transfer start instruction is being executed. <2> When bus release signal (REL) is detected. <3> When the $\overline{\text{RESET}}$ signal is generated. <4> CSIE = 0 (see **Figure 5-60**)	When command signal (CMD) is detected.

Bus release detection flag (R)

RELD	Condition To Be Cleared (RELD = 0)	Condition To Be Set (RELD = 1)
	<1> When transfer start instruction is being executed. <2> When the $\overline{\text{RESET}}$ signal is generated. <3> CSIE = 0 (see **Figure 5-60**) <4> SVA is not equal to SIO when an address is received.	When bus release signal (REL) is detected.

Command trigger bit (W)

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

Caution The SB0 (or SB1) must not be cleared during serial data transfer, but before or after it.

Bus release trigger bit (W)

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

Caution The SB0 (or SB1) must not be cleared during serial data transfer, but before or after it.

(3) Shift register (SIO)

Figure 5-62 shows the configuration of the system comprising the shift register and peripheral devices. The SIO is an 8-bit register which performs parallel-to-serial conversion and shift operation in synchronization with the serial clock.

Serial data transfer starts when data is entered into the SIO.

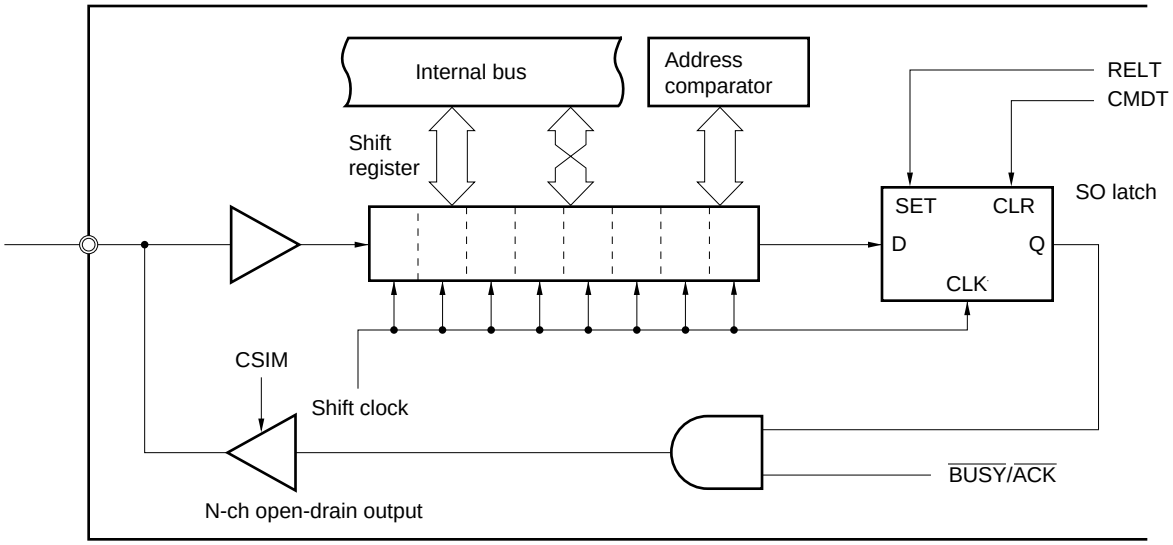
During data transmission, the data written in the SIO is output to the serial output (SO) or serial data bus (SB0 or SB1). During receiving, data is read from the serial input (SI) or SB0 (or SB1) to the SIO.

Data can be read and written by an 8-bit manipulation instruction.

When a $\overline{\text{RESET}}$ signal is generated during an operation, the contents of the SIO are uncertain. When the $\overline{\text{RESET}}$ signal is generated in the standby mode, the contents of the SIO are held.

The shift operation stops after data is transmitted or received in an 8-bit unit.

Figure 5-62. System Comprising Shift Register and Peripheral Devices Configuration



Data can be written and read to/from the SIO in the following timings.

- The serial interface operation enable/disable bit (CSIE) is set to 1 except when the CSIE is set to 1 after data is written in the shift register.
- The serial clock is masked after the 8-bit serial data transfer ends.
- The $\overline{\text{SCK}}$ is high.

Be sure to write or read data to or from the SIO when $\overline{\text{SCK}}$ is high.

The input pin of the data bus is shared with the output pin in the two-wire serial I/O mode and SBI mode. The output pin is of the N-ch open-drain configuration. Therefore, set FFH in the SIO of the device that is to receive data.

(4) Slave address register (SVA)

SVA is an 8-bit register that sets a slave address (specification number).

It is operated by an 8-bit manipulation instruction.

The contents of the SVA becomes uncertain when a $\overline{\text{RESET}}$ signal is generated. However, when the $\overline{\text{RESET}}$ signal is generated in the standby mode, the contents of the SVA are held.

(a) Detection of slave address (in SBI mode)

When the μPD753108 is connected to the serial bus as a slave device, SVA is used to set the slave address (specification number) of the μPD753108. The master outputs a slave address to the slaves connected to the bus, to select a specific slave. The slave address output from the master is compared with the value of the SVA of the slave by the address comparator of the slave. When the two addresses coincide, the slave is selected.

At this time, the bit 6 (COI) of the serial operation mode register (CSIM) is set to "1". When an address is received from the master and coincidence between the received address and the address set to the SVA is not detected, the bus release detection flag (RELD) is cleared to 0. IRQCSI is set only when coincidence is detected when WUP = 1. This interrupt function allows the slave (μPD753108) to learn that the master has issued a request for communication.

(b) Detection of errors (in 2-wire serial I/O mode and SBI mode)

The SVA detects an error in the following cases:

- When the μPD753108 operates as the master and transmits addresses, commands, and data
- When the μPD753108 transmits data as a slave device

For details, refer to **5.6.6 (6) Error detection** or **5.6.7 (8) Error detection**.

5.6.4 Operation stop mode

The operation stop mode is used when serial transfer is not performed, to reduce the power dissipation.

In this mode, the shift register does not perform shift operations. Therefore, it can be used as an ordinary 8-bit register.

When the reset signal is input, operation stop mode is set. The P02/SO/SB0 and P03/SI/SB1 pins are set to the input port mode. The P01/$\overline{\text{SCK}}$ pin can be used as an input port pin if so specified by the serial operation mode register.

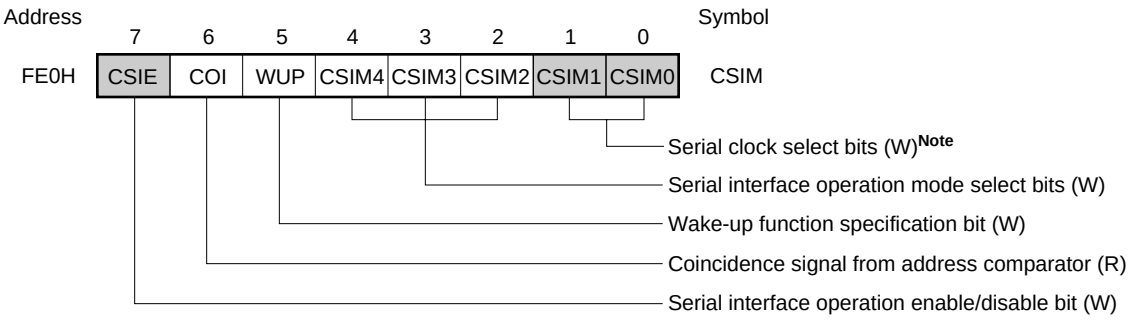
[Register setting]

Operation stop mode is set by using the serial operation mode register (CSIM) (For the format of the CSIM, refer to **5.6.3 (1) Serial operation mode register (CSIM)**).

The CSIM is manipulated by an 8-bit manipulation instruction. However, the CSIE bit of this register can be manipulated in 1-bit units. The name of the bit can be used for manipulation.

The CSIM is set to 00H at reset.

The shaded portions in the figure below indicate the bits used in operation stop mode.



Note This bit can select the status of the P01/$\overline{\text{SCK}}$ pin.

Remark (R) : read only
(W) : write only

Serial interface operation enable/disable bit (W)

		Operation of Shift Register	Serial Clock Counter	IRQCSI Flag	SO/SB0 and SI/SB1 Pins
CSIE	0	Shift operation disabled	Cleared	Retained	Dedicated to port 0 function

Serial clock select bit (W)

The P01/$\overline{SCK}$ pin is set to the following status according to the setting of the CSIM0 and CSIM1 bits.

CSIM1	CSIM0	Status of P01/$\overline{SCK}$ Pin
0	0	High impedance
0	1	High level
1	0	
1	1	

Clear the CSIE bit using the following procedure during serial transfer:

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

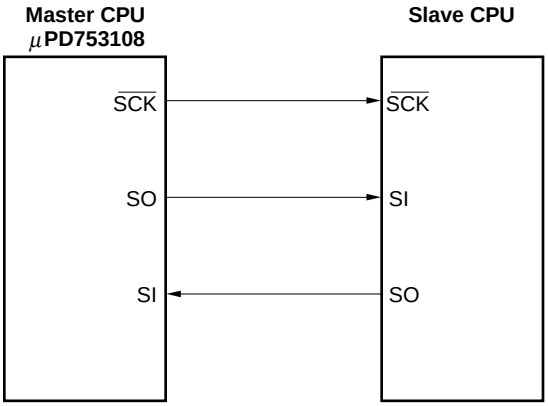
5.6.5 Operation in 3-wire serial I/O mode

In the 3-wire operation mode, the μPD753108 can be connected to microcontrollers in the 75XL Series, 75X Series, and 78K Series, and to various peripheral I/O devices.

In this mode, communication is established by using three lines: serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

Figure 5-63. Example of System Configuration in 3-wire Serial I/O Mode

3-wire serial I/O ↔ 3-wire serial I/O



Remark The μPD753108 can be also used as a slave CPU.

(1) Register setting

When 3-wire serial I/O mode is used, the following two registers must be set:

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

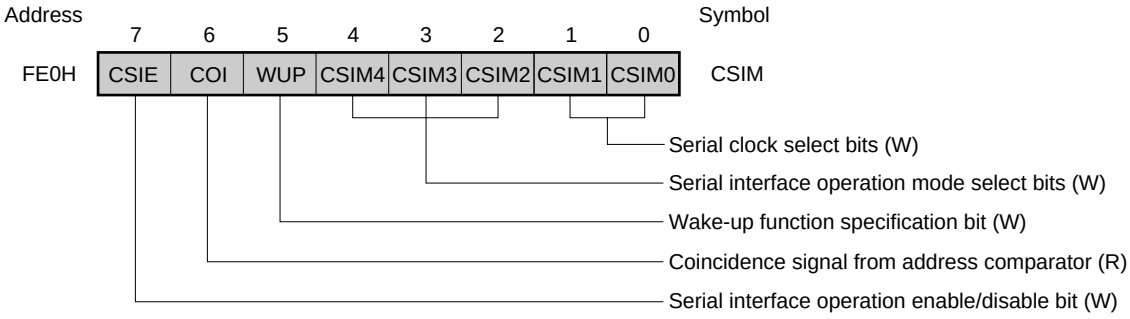
(a) Serial operation mode register (CSIM)

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

CSIM is manipulated by using 8-bit manipulation instructions. Bits 7, 6, and 5 can also be manipulated in 1-bit units.

The contents of the CSIM are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in 3-wire serial I/O mode.



Remark (R) : read only
(W) : write only

Serial interface operation enable/disable bit (W)

		Operation of Shift Register	Serial Clock Counter	IRQCSI Flag	SO/SB0 and SI/SB1 Pins
CSIE	1	Shift operation enabled	Count operation	Can be set	Function in each mode and port 0 function shared

Signal from address comparator (R)

COI[Note]	Clear Condition (COI = 0)	Set Condition (COI = 1)
	When the slave address register (SVA) data and shift register data do not coincide	When slave address register (SVA) data and shift register data coincide

Note COI can be read before the start of serial transfer and after completion of the serial transfer. An undefined value is obtained if this bit is read during transfer. Data written to COI by an 8-bit manipulation instruction is ignored.

Wake-up function specification bit (W)

WUP	0	Sets IRQCSI each time serial transfer is completed

Serial interface operation mode select bit (W)

CSIM4	CSIM3	CSIM2	Bit Order of Shift Register	SO Pin Function	SI Pin Function
×	0	0	$SIO_{7-0} \leftrightarrow XA$ (MSB first)	SO (CMOS output)	SI (CMOS input)
		1	$SIO_{0-7} \leftrightarrow XA$ (LSB first)		

Remark ×: don't care

Serial clock select bit (W)

CSIM1	CSIM0	Serial Clock	$\overline{SCK}$ Pin Mode
0	0	External clock input to $\overline{SCK}$ pin	Input
0	1	Timer/event counter output (TOUT0)	Output
1	0	$f_X/2^4$ (262 kHz)**Note**	
1	1	$f_X/2^3$ (524 kHz)**Note**	

Note The frequency in parentheses applies when f_X = 4.19-MHz operation.

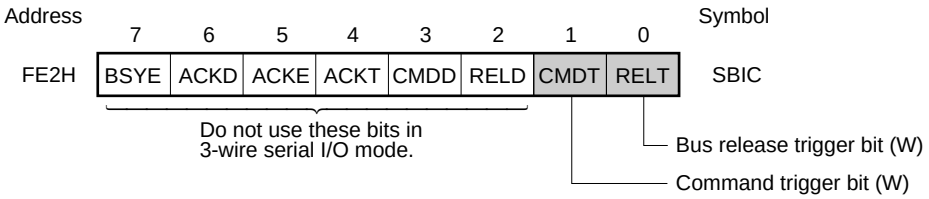
(b) Serial bus interface control register (SBIC)

When the three-wire serial I/O mode is used, set SBIC as shown below (For the format of SBIC, refer to **5.6.3 (2) Serial bus interface control register (SBIC)**).

This register is manipulated by using bit manipulation instructions.

The contents of SBIC are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the three-wire serial I/O mode.



Remark (W): write only

Command trigger bit

CMDT	This bit controls the output trigger of a command signal (CMD). When this bit is set to 1, the SO latch is cleared to 0. After that, the CMDT bit is automatically cleared to 0.

Bus release trigger bit (W)

RELT	This bit controls the output trigger of a bus release signal (REL). When this bit is set to 1, the SO latch is set to 1. After that, the RELT bit is automatically cleared to 0.

Caution Do not use bits in the SBIC register other than CMDT and RELT in the 3-wire serial I/O mode.

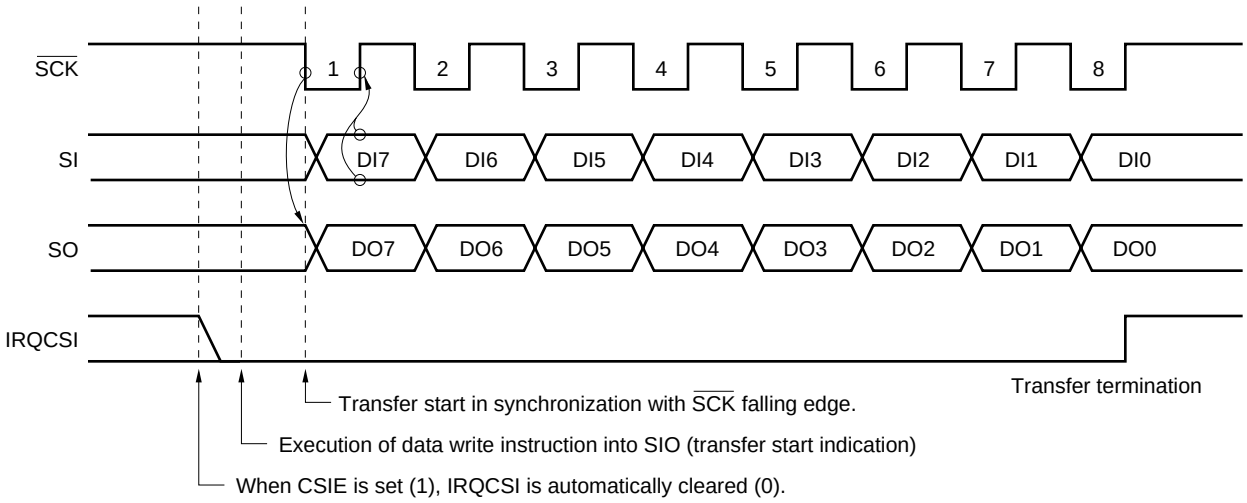
(2) Communication operation

The 3-wire serial I/O mode transmit/receive data in 8-bit units. Data is transferred one bit at a time in synchronization with a given serial clock.

Shift register shift operation is performed in synchronization with the serial clock ($\overline{SCK}$) falling edge. Transmit data is retained in the SO latch and output from the SO pin. On the $\overline{SCK}$ rising edge, receive data input to the SI pin is latched in the shift register.

When 8-bit transfer terminates, shift register operation automatically stops and the interrupt request flag (IRQCSI) is set.

Figure 5-64. 3-wire Serial I/O Mode Timing



Because the SO pin is a CMOS output pin and outputs the status of the SO latch, the output status of the SO pin can be manipulated by setting the RELT and CMDT bits.

However, do not perform this manipulation during serial transfer.

The output status of the $\overline{SCK}$ pin can be controlled by manipulating the P01 latch in the output mode (mode of the internal system clock) (Refer to **5.6.8 $\overline{SCK}$ pin output manipulation**).

(3) Selecting the serial clock

The serial clock is selected by using bits 0 and 1 of the serial operation mode register (CSIM). The following four types of serial clocks can be selected:

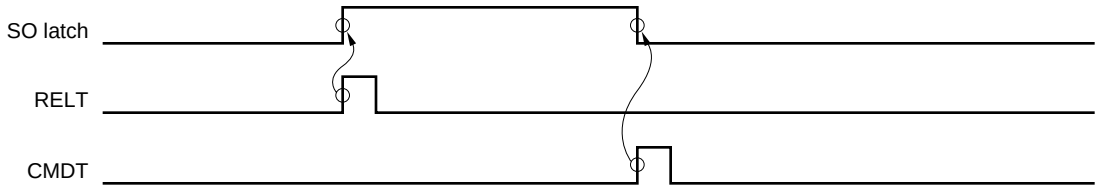
Table 5-9. Selection of Serial Clock and Applications (in 3-wire serial I/O mode)

Mode Register		Serial Clock		Timing at Which Shift Register Can Be Read/ Written and Serial Transfer Can Be Started	Application
CSIM1	CSIM0	Source	Masking Serial Clock		
★ 0	0	External $\overline{SCK}$	Automatically masked at end of transfer of 8-bit data	<1> In operation enable mode (CSIE = 1) <2> If serial clock is masked after 8-bit serial transfer <3> When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Half duplex start-stop synchronization transfer (software control)
1	0	$f_X/2^4$			Medium-speed serial transfer
1	1	$f_X/2^3$			High-speed serial transfer

(4) Signals

Figure 5-65 illustrates the operation of RELT and CMDT.

Figure 5-65. Operation of RELT and CMDT



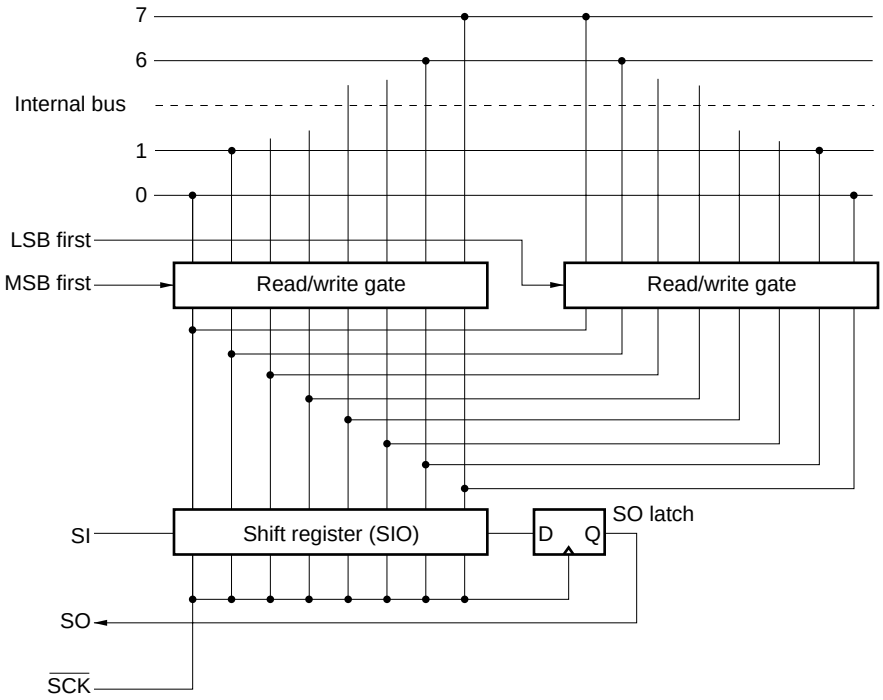
(5) Transfer first bit change between MSB and LSB

The 3-wire serial I/O mode enables selection of the most significant bit (MSB) or least significant bit (LSB) for the transfer first bit.

Figure 5-66 shows the shift register (SIO) and internal bus configuration. As shown in Figure 5-66, MSB and LSB can be reversed for read/write.

MSB or LSB can be specified as the transfer first bit by setting serial operation mode register (CSIM) bit 2.

Figure 5-66. Transfer Bit Change Circuit



The transfer first bit is switched by changing the bit order of data write into the shift register (SIO). The SIO shift order is always the same.

Change the transfer first bit between MSB and LSB before writing data into the shift register.

(6) Starting transfer

Serial transfer is started when the transfer data is placed in the shift register (SIO), if the following two conditions are satisfied:

- Serial interface operation enable/disable bit (CSIE) = 1
- The internal serial clock is stopped after 8-bit serial transfer or $\overline{SCK}$ is high

Caution Transfer is not started even if CSIE is set to "1" after the data has been written to the shift register.

When an 8-bit transfer has been completed, the serial transfer is automatically stopped, and an interrupt request flag (IRQCSI) is set.

Example To transfer the RAM data specified by the HL register to SIO and, at the same time, load the data in SIO to the accumulator and start serial transfer

```
MOV     XA, @HL         ; Fetches out transfer data from RAM
SEL     MB15            ; or CLR1 MBE
XCH     XA, SIO         ; Exchanges transmit data and receive data, and starts transfer
```

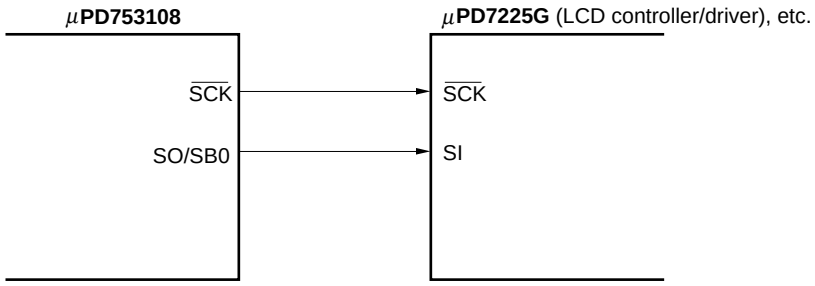
(7) Applications using 3-wire serial I/O mode

Example 1. To transfer data, MSB first, with 262-kHz transfer clock (at 4.19 MHz) (master operation)

<Program example>

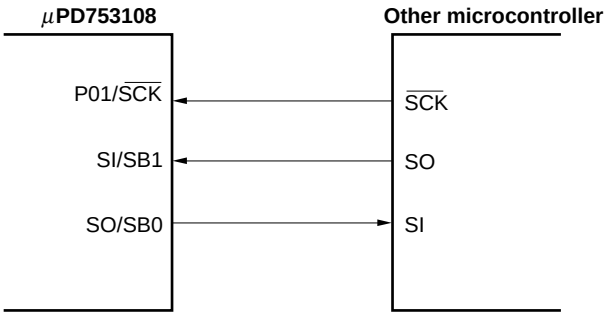
```
CLR1    MBE
MOV     XA, #10000010B
MOV     CSIM, XA        ; Sets transfer mode
MOV     XA, TDATA       ; TDATA is address storing transfer data
MOV     SIO, XA         ; Sets transfer data and starts transfer
```

Caution After transfer has been started for the first time, transfer can be started by writing data to SIO (by using MOV SIO, XA or XCH XA, SIO) the second and later times.



In this example, the SI/SB1 pin of the μPD753108 can be used as an input pin.

Example 2. To transmit or receive data, LSB first, with an external clock (slave operation)
(In this example, the shift register is read/written using a function that reverses the MSB-LSB order.)



<Program example>

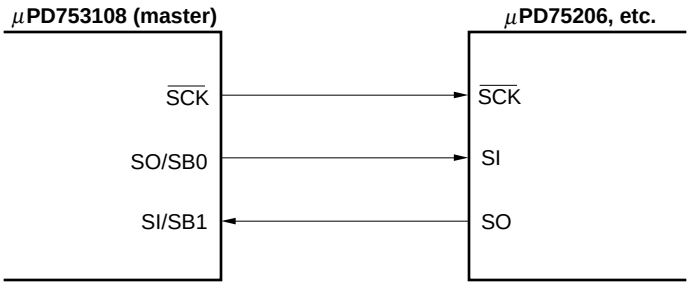
Main routine

```
CLR1   MBE
MOV    XA, #84H
MOV    CSIM, XA        ; Stops serial operation, MSB/LSB inverse mode, external clock
MOV    XA, TDATA
MOV    SIO, XA         ; Sets transfer data and starts transfer
EI     IECSI
EI
```

Interrupt routine (MBE = 0)

```
MOV    XA, TDATA
XCH    XA, SIO         ; Receive data ↔ transmit data, starts transfer
MOV    RDATA, XA       ; Saves receive data
RETI
```

Example 3. To transmit or receive data at high speeds using a 524-kHz (at 4.19 MHz) transfer clock



<Program example> ··· Master

```
        CLR1    MBE
        MOV     XA, #10000011B
        MOV     CSIM, XA        ; Sets transfer mode
        MOV     XA, TDATA
        MOV     SIO, XA         ; Sets transfer data and starts transfer
          :
LOOP:   SKTCLR  IRQCSI          ; Test IRQCSI
        BR      LOOP
        MOV     XA, SIO         ; Receives data
```

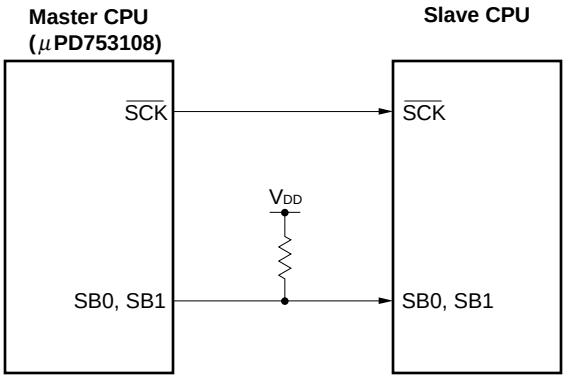
5.6.6 Operation in 2-wire serial I/O mode

The 2-wire serial I/O mode can be used in any communication format if so specified by program.

Basically, communication is established by using two lines: serial clock ($\overline{SCK}$) and serial data input/output (SB0 or SB1).

Figure 5-67. Example of System Configuration in 2-wire Serial I/O Mode

2-wire serial I/O ↔ 2-wire serial I/O



Remark The μPD753108 can be also used as a slave CPU.

(1) Register setting

When the 2-wire serial I/O mode is used, the following two registers must be set:

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

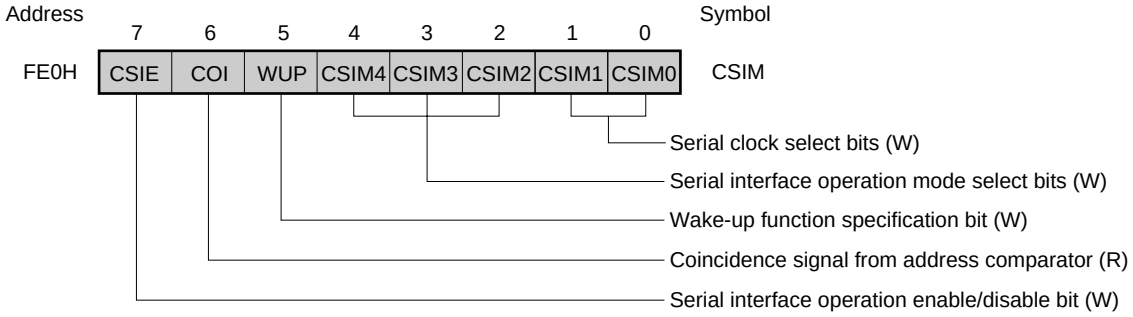
(a) Serial operation mode register (CSIM)

When the 2-wire serial I/O mode is used, set CSIM as shown below (For the format of CSIM, refer to **5.6.3 (1) Serial operation mode register (CSIM)**).

CSIM is manipulated by using an 8-bit manipulation instructions. Bits 7, 6, and 5 can also be manipulated in 1-bit units.

The contents of the CSIM are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the 2-wire serial I/O mode.



Remark (R) : read only
(W): write only

Serial interface operation enable/disable bit (W)

		Operation of Shift Register	Serial Clock Counter	IRQCSI Flag	SO/SB0 or SI/SB1 Pin
CSIE	1	Shift operation enabled	Count operation	Can be set	Function in each mode and port 0 function shared

Signal from address comparator (R)

COI[Note]	Clear Condition (COI = 0)	Set Condition (COI = 1)
	When slave address register (SVA) data and shift register data do not coincide	When slave address register (SVA) data and shift register data coincide

Note COI can be read before the start of a serial transfer or after completion of a serial transfer. An undefined value is read if this bit is read during transfer. Data written to COI by an 8-bit manipulation instruction is ignored.

Wake-up function specification bit (W)

WUP	0	Sets IRQCSI each time serial transfer is completed

Serial interface operation mode select bit (W)

CSIM4	CSIM3	CSIM2	Bit Order of Shift Register	SB0/P02 Pin Function	SB1/P03 Pin Function
0	1	1	$SIO_{7-0} \leftrightarrow XA$ (MSB first)	SB0 (N-ch open-drain I/O)	P03 (CMOS input)
1				P02 (CMOS input)	SB1 (N-ch open-drain I/O)

Serial clock select bit (W)

CSIM1	CSIM0	Serial Clock	$\overline{SCK}$ Pin Mode
0	0	External clock input to $\overline{SCK}$ pin	Input
0	1	Timer/event counter output (TOUT0)	Output
1	0	$f_X/2^6$ (65.5 kHz)**Note**	
1	1		

Note The frequency in parentheses applies when f_X = 4.19-MHz operation.

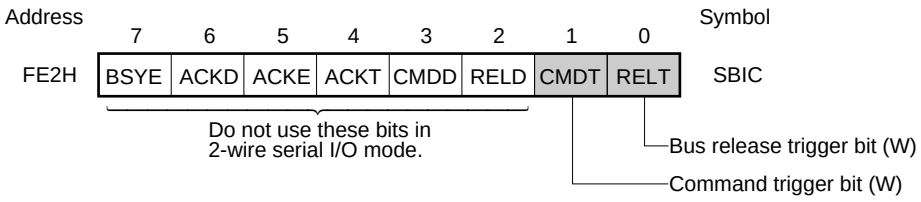
(b) Serial bus interface control register (SBIC)

When the 2-wire serial I/O mode is used, set SBIC as shown below (For the format of SBIC, refer to **5.6.3 (2) Serial bus interface control register (SBIC)**).

This register is manipulated by using bit manipulation instructions.

The contents of SBIC are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in the 2-wire serial I/O mode.



Remark (W): write only

Command trigger bit

CMDT	This bit controls the output trigger of a command signal (CMD). When this bit is set to 1, the SO latch is cleared to 0. After that, the CMDT bit is automatically cleared to 0.

Bus release trigger bit (W)

RELT	This bit controls the output trigger of a bus release signal (REL). When this bit is set to 1, the SO latch is set to 1. After that, the RELT bit is automatically cleared to 0.

Caution Do not use bits of the SBIC register other than CMDT and RELT in 2-wire serial I/O mode.

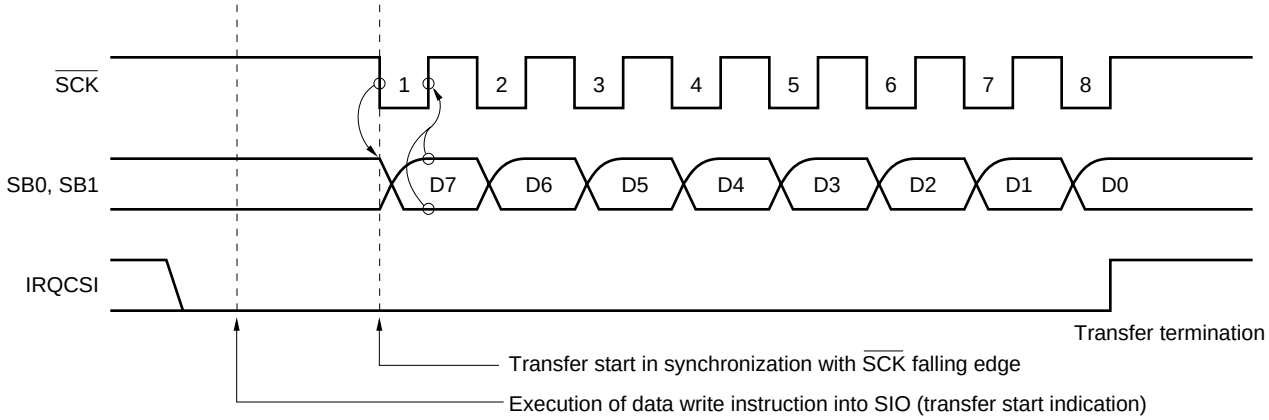
(2) Communication operation

The 2-wire serial I/O mode transmits/receives data in 8-bit units. Data is transferred one bit at a time in synchronization with a given serial clock.

Shift register shift operation is performed in synchronization with the serial clock ($\overline{SCK}$) falling edge. Transmit data is retained in the SO latch and output starting at the MSB from the SB0/P02 (or SB1/P03) pin. On the $\overline{SCK}$ rising edge, receive data input from the SB0 (or SB1) pin is latched in the shift register.

When an 8-bit transfer terminates, shift register operation automatically stops and the interrupt request flag (IRQCSI) is set.

Figure 5-68. 2-wire Serial I/O Mode Timing



The SB0 (or SB1) pin specified for the serial data bus becomes N-ch open-drain input/output, thus must be pulled high with an external pull-up resistor. Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance.

Since the SB0 (or SB1) pin outputs the SO latch state, the SB0 (or SB1) pin output state can be manipulated by setting the RELT and CMDT bits.

However, do not perform this manipulation during serial transfer.

In the output mode (internal system clock mode), the $\overline{SCK}$ pin output state can be controlled if the P01 output latch is manipulated (See **5.6.8 $\overline{SCK}$ pin output manipulation**).

(3) Selecting the serial clock

The serial clock is selected by using the bits 0 and 1 of the serial operation mode register (CSIM). Three types of serial clocks can be selected:

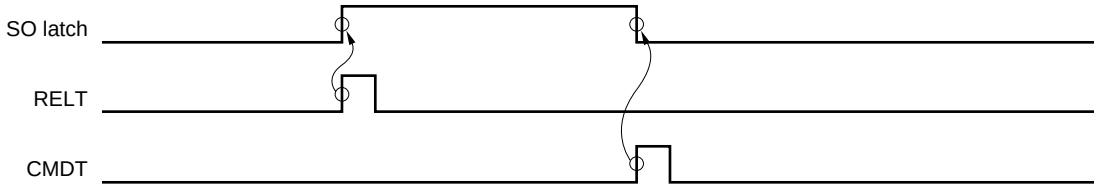
Table 5-10. Selection of Serial Clock and Applications (in 2-wire serial I/O mode)

Mode Register		Serial Clock		Timing at Which Shift Register Can Be Read/ Written and Serial Transfer Can Be Started	Application
CSIM1	CSIM0	Source	Masking Serial Clock		
★ 0	0	External $\overline{SCK}$	Automatically masked at end of transfer of 8-bit data	<1> In operation enable mode (CSIE = 1) <2> If serial clock is masked after 8-bit serial transfer <3> When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Serial transfer at any speed
1	0	$f_X/2^6$			Low-speed serial transfer
1	1				

(4) Signals

Figure 5-69 illustrates the operation of RELT and CMDT.

Figure 5-69. Operation of RELT and CMDT



(5) Starting transfer

Serial transfer is started when the transfer data is placed in the shift register (SIO), if the following two conditions are satisfied:

- Serial interface operation enable/disable bit (CSIE) = 1
- The internal serial clock is stopped after 8-bit serial transfer, or $\overline{SCK}$ is high

Cautions 1. Transfer is not started even if CSIE is set to "1" after the data has been written to the shift register.

2. Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance.

When an 8-bit transfer has been completed, serial transfer is automatically stopped, and an interrupt request flag (IRQCSI) is set.

(6) Error detection

In 2-wire serial I/O mode, because the status of the serial bus SB0 or SB1 during transmission is also loaded to the shift register SIO of the device transmitting data, an error can be detected by the following methods:

(a) By comparing SIO data before and after transmission

If the two data differ from each other, it can be assumed that a transmission error has occurred.

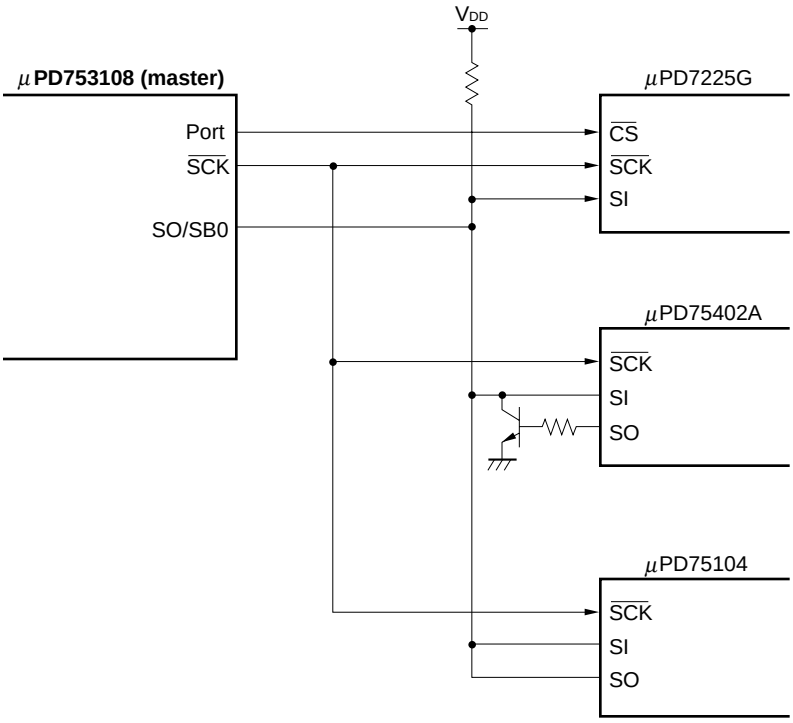
(b) By using the slave address register (SVA)

The transmit data is placed in SIO and SVA and transmission is executed. After transmission, the COI bit (coincidence signal from the address comparator) of the serial operation mode register (CSIM) is tested. If this bit is "1", the transmission has been completed normally. If it is "0", it can be assumed that a transmission error has occurred.

(7) Application using 2-wire serial I/O mode

2-wire serial I/O mode can be used to connect multiple devices by configuring a serial bus.

Example To configure a system by connecting the μPD753108 as the master and μPD75104, μPD75402A, and μPD7225G as slaves



The SI and SO pins of the μPD75104 are connected together. When serial data is not output, the serial operation mode register is manipulated so that the output buffer is turned off to release the bus.

Because the SO pin of the μPD75402A cannot go into a high-impedance state, a transistor is connected to the SO pin as shown in the figure, so that the SO pin can be used as an open-collector output pin. When data is input to the μPD75402A, the transistor is turned off by writing 00H to the shift register in advance.

The timing of when each microcontroller outputs data is determined in advance.

The serial clock is output by the μPD753108, which is the master. All the slave microcontrollers operate on an external clock.

5.6.7 SBI mode operation

The SBI (serial bus interface) is a high-speed serial interface system which is compliant with the NEC serial bus format.

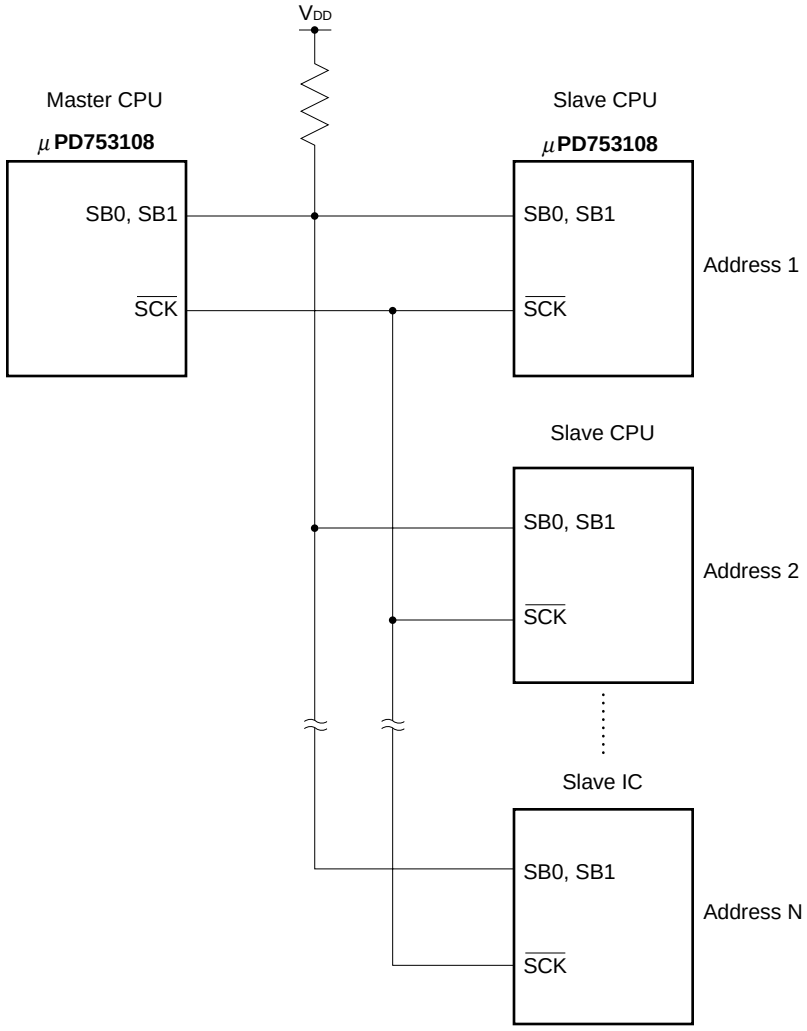
The SBI is a high-speed serial bus of a single master; bus configuration function is added to the clocked serial I/O system so that the master can communicate with a number of devices by using two signal lines. When microcontrollers and peripheral ICs make up a serial bus, the number of ports and wiring on the printed circuit boards can be reduced.

The master can output an “address” to select the slave device with which it is to communicate, a “command” to tell the slave which operation to perform, and actual “data”, via the serial data bus. The slave identifies the data it has received from the master as an “address”, “command”, or “data” by using hardware. This SBI function simplifies the portion of the application program that controls the serial interface.

The SBI function is provided in several devices such as the “75XL Series”, “75X Series”, and 8- and 16-bit single-chip microcontrollers in the “78K Series”.

Figure 5-70 shows an example of the configuration of the serial bus with CPUs and peripheral ICs having a serial interface conforming to SBI.

Figure 5-70. SBI System Configuration Example



Cautions 1. Since the serial data bus pin SB0 (or SB1) is open-drain output in the SBI, the serial data bus line is in the wired-OR status. The serial data bus line needs a pull-up resistor.

2. When master-slave exchange is to be performed, the I/O switching of the serial clock line ($\overline{SCK}$) is done asynchronously between the master and slave, therefore the pull-up resistor is also necessary for the $\overline{SCK}$.

(1) Function of SBI

If two or more devices are connected to configure a serial bus with the existing serial I/O method, many ports and much wiring are necessary to distinguish among the chip select signal, command, and data, and to identify the busy status, because the existing serial I/O method only provides a data transfer function. Moreover, if software performs these manipulations, the workload of the software increases.

In the SBI mode, the serial bus can be configured by using only two lines: serial clock $\overline{SCK}$ and serial data bus SB0 or SB1. Therefore, the number of ports can be reduced and the wiring on the printed circuit board can be shortened.

The functions of the SBI mode are described below.

(a) Address/command/data identification function

Serial data is identified as an address, command, or data.

(b) Chip select function by using address

The master transmits an address to a slave to select the slave (chip select).

(c) Wake-up function

The slave can judge whether it has received an address (whether the slave has received the chip select signal from the master) by using the wake-up function (which can be set or cleared via software).

When the wake-up function is set, an interrupt (IRQCSI) is generated when the slave has received an address coinciding with its own address. Therefore, even when the master communicates with two or more slaves, the slaves other than that selected by the master can operate independently of the serial communication between the master and selected slave.

(d) Acknowledge signal ($\overline{ACK}$) control function

The acknowledge signal is controlled so that confirmation can be made that serial data has been received.

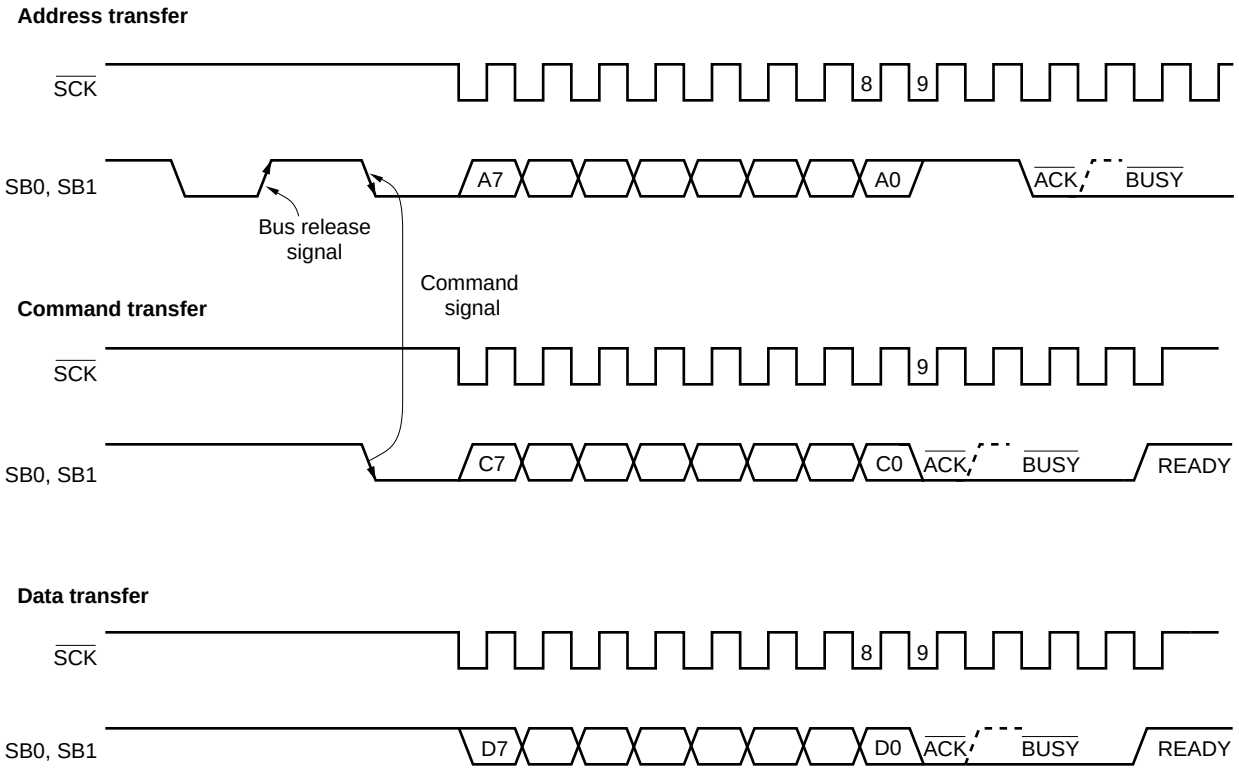
(e) Busy signal ($\overline{BUSY}$) control function

The busy signal is controlled so that the master is notified of the busy status of a slave.

(2) Definition of SBI

This paragraph describes the format of the serial data in the SBI mode and the meanings of the signals used. The serial data transferred in the SBI mode are classified into “address”, “command”, and “data”.

Figure 5-71. SBI Transfer Timings



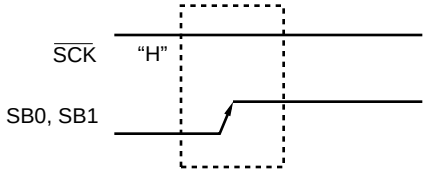
The bus release and command signals are output by the master. $\overline{\text{BUSY}}$ is output by the slave. $\overline{\text{ACK}}$ can be output by both the master and slave (usually, this signal is output by the receiver of 8-bit data).

The master continues outputting the serial clock since the start of 8-bit data transfer until the $\overline{\text{BUSY}}$ signal is deasserted.

(a) Bus release signal (REL)

The bus release signal is asserted when the SB0 or SB1 line goes high while the $\overline{SCK}$ line is high (i.e., when the serial clock is not output). This signal is output by the master.

Figure 5-72. Bus Release Signal



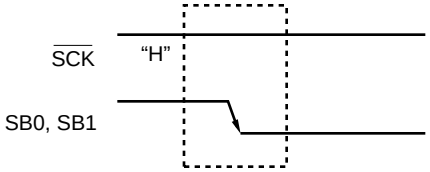
The bus release signal indicates that the master is to transmit an address to a slave. The slave has hardware that detects the bus release signal.

★ **Caution The transition of the SB0 or SB1 line from low level to high level when the $\overline{SCK}$ line is high is interpreted as a bus release signal. Therefore, if the transition timing of the bus is shifted due to the influence of the board capacitance, this may be interpreted as a bus release signal regardless of whether or not data are transmitted. Take precautions in wiring so that noise is not applied to the signal line.**

(b) Command signal (CMD)

The command signal is asserted when the SB0 or SB1 line goes low while the $\overline{SCK}$ line is high (i.e., when the serial clock is not output). This signal is output by the master.

Figure 5-73. Command Signal



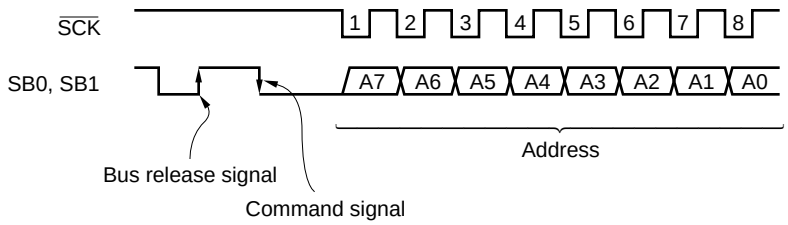
The slave has hardware that detects the command signal.

★ **Caution The transition of the SB0 or SB1 line from high level to low level when the $\overline{SCK}$ line is high is interpreted as a command signal. Therefore, if the transition timing of the bus is shifted due to the influence of the board capacitance, this may be interpreted as a command signal regardless of whether or not data are transmitted. Take precautions in wiring so that noise is not applied to the signal line.**

(c) Address

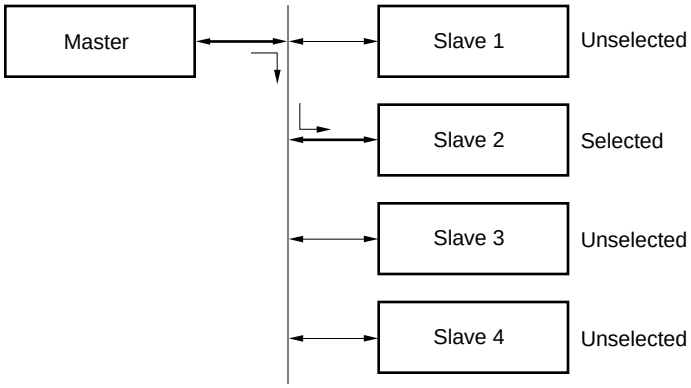
An address is 8-bit data output by the master to select a specific slave from the slaves connected to the bus line.

Figure 5-74. Address



The 8-bit data following the bus release signal and command signal is defined as an address. The slave detects an address by using hardware, and checks whether the 8-bit data coincides with its own specification number (slave address). If the 8-bit data coincides with the slave address, the slave is selected. After that, the slave communicates with the master, until the master later unselects the slave.

Figure 5-75. Selecting Slave by Address



(d) Command and data

The master transmits commands to or transmits data to or receives data from the slave it has selected by transmitting an address.

Figure 5-76. Command

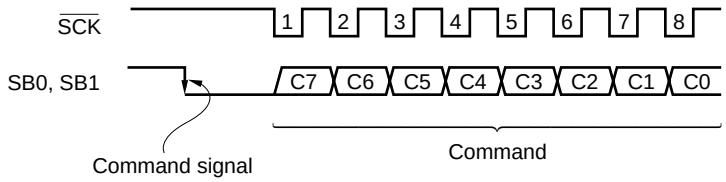
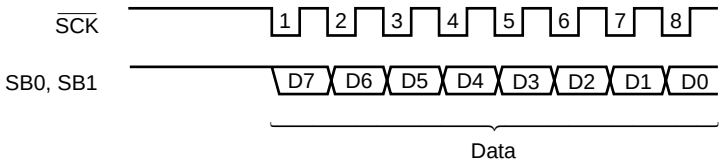


Figure 5-77. Data



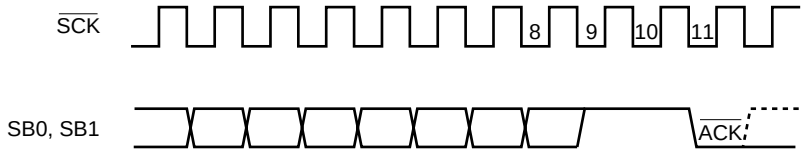
8-bit data following a command signal is defined as a command. 8-bit data that does not follow a command signal is defined as data. How to use commands and data can be determined arbitrarily, depending on the communication specifications.

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

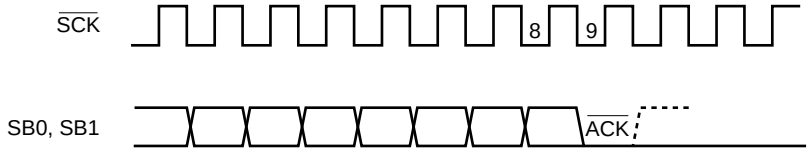
The acknowledge signal is used for confirmation of data reception between the transmitter and receiver.

Figure 5-78. Acknowledge Signal

(When output in synchronization with 11th $\overline{SCK}$)



(When output in synchronization with 9th $\overline{SCK}$)



The acknowledge signal is a one-shot pulse synchronized with the falling edge of $\overline{SCK}$ after 8-bit data has been transferred, and can be synchronized with arbitrary assertion of $\overline{SCK}$.

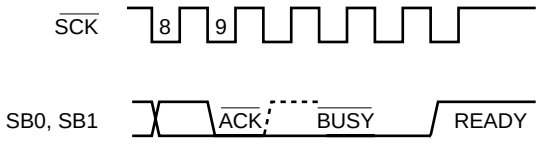
The transmitter side checks, after it has transmitted 8-bit data, whether the receiver side returns an acknowledge signal. If the acknowledge signal is not returned in a fixed time period after the data has been transmitted, it is judged that the data has not been received correctly.

(f) Busy ($\overline{\text{BUSY}}$) and ready (READY) signals

A busy signal is output by a slave to inform the master that the slave is preparing for transmission or reception.

A ready signal is also output by a slave to inform the master that the slave is now ready for transmission or reception.

Figure 5-79. Busy and Ready Signals



In the SBI mode, the slave makes the SB0 (or SB1) line low to inform the master of the busy status.

The busy signal is output following the acknowledge signal output by the master or slave. The busy signal is asserted or deasserted in synchronization with the falling edge of $\overline{\text{SCK}}$. The master automatically ends output of serial clock $\overline{\text{SCK}}$ when the busy signal is deasserted.

The master can start the next transfer when the busy signal has been deasserted and the ready signal is asserted.

(3) Register setting

When the SBI mode is used, the following two registers must be set:

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

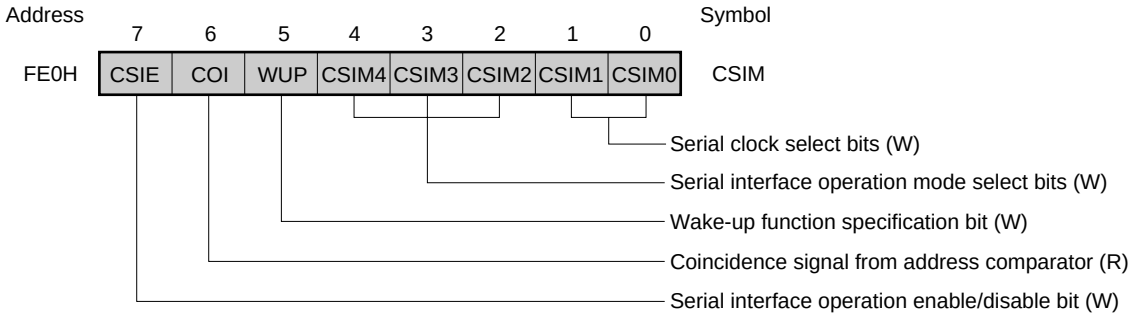
(a) Serial operation mode register (CSIM)

When the SBI mode is used, set the CSIM as shown below (For the format of the CSIM, refer to **5.6.3 (1) Serial operation mode register (CSIM)**).

The CSIM is manipulated by using 8-bit manipulation instructions. Bits 7, 6, and 5 can also be manipulated in 1-bit units.

The contents of the CSIM are cleared to 00H at reset.

The shaded portion in the figure indicates the bits used in SBI mode.



Remark (R) : read only
(W): write only

Serial interface operation enable/disable bit (W)

		Operation of Shift Register	Serial Clock Counter	IRQCSI Flag	SO/SB0 or SI/SB1 Pin
CSIE	1	Shift operation range	Count operation	Can be set	Function in each mode and port 0 function shared

Signal from address comparator (R)

COI[Note]	Clear Condition (COI = 0)	Set Condition (COI = 1)
	When slave address register (SVA) data and shift register data do not coincide	When slave address register (SVA) data and shift register data coincide

Note COI can be read before the start of serial transfer and after the completion of serial transfer. An undefined value is read if this bit is read during transfer. Data written to COI by an 8-bit manipulation instruction is ignored.

Wake-up function specification bit (W)

WUP	0	Sets IRQCSI each time a serial transfer is completed with SBI mode masked
	1	Used only by the slave in SBI mode. Only when the address received by the slave after the bus has been released coincides with the data in the slave register of the slave (wake-up status), IRQCSI is set. SB0 or SB1 goes into a high-impedance state.

Caution $\overline{\text{BUSY}}$ is not deasserted if WUP is set to 1 while the $\overline{\text{BUSY}}$ signal is output. In the SBI mode, after a command to deassert the $\overline{\text{BUSY}}$ signal has been issued, the $\overline{\text{BUSY}}$ signal is output until the next serial clock ($\overline{\text{SCK}}$) falls. Before setting WUP to 1, be sure to deassert the $\overline{\text{BUSY}}$ signal and confirm that the SB0 (or SB1) pin has gone high.

Serial interface operation mode select bit (W)

CSIM4	CSIM3	CSIM2	Bit Order of Shift Register	SB0/P02 Pin Function	SB1/P03 Pin Function
0	1	0	$SIO_{7-0} \leftrightarrow XA$ (MSB first)	SB0 (N-ch open-drain I/O)	P03 (CMOS input)
1				P02 (CMOS input)	SB1 (N-ch open-drain I/O)

Serial clock select bit (W)

CSIM1	CSIM0	Serial Clock	$\overline{\text{SCK}}$ Pin Mode
0	0	External clock input to $\overline{\text{SCK}}$ pin	Input
0	1	Timer/event counter output (TOUT0)	Output
1	0	$f_X/2^4$ (262 kHz)**Note**	
1	1	$f_X/2^3$ (524 kHz)**Note**	

Note The frequency in parentheses applies when f_X = 4.19-MHz operation.

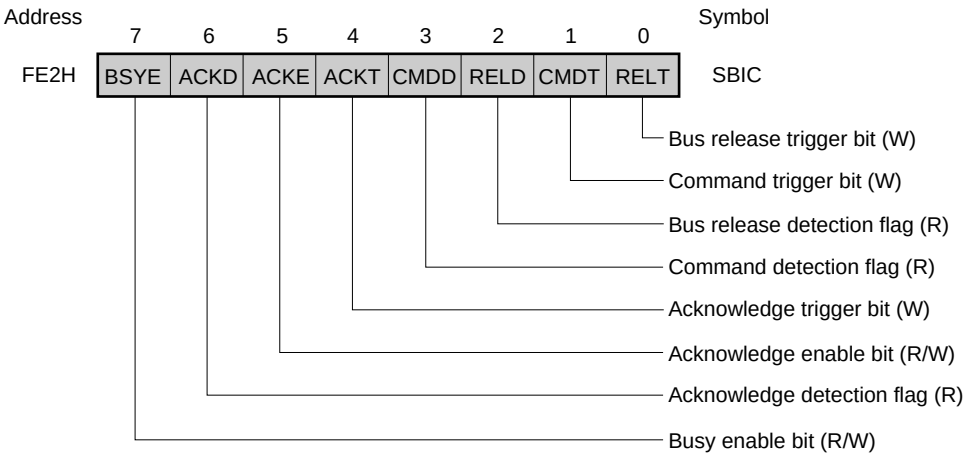
(b) Serial bus interface control register (SBIC)

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

This register is manipulated by using bit manipulation instructions.

The contents of SBIC are cleared to 00H at reset.

The shaded portion in the figure indicates the used bits in the SBI mode.



Remark (R) : read only
(W) : write only
(RW) : read/write

Busy enable bit (R/W)

BSYE	0	<1> Disables automatic output of busy signal <2> Stops output of busy signal in synchronization with falling edge of $\overline{SCK}$ immediately after clear instruction has been executed
	1	Following acknowledge signal, busy signal is output in synchronization with falling edge of $\overline{SCK}$

Acknowledge detection flag (R)

ACKD	Clear Condition (ACKD = 0)	Set Condition (ACKD = 1)
	<1> At start of transfer <2> At reset input	When acknowledge signal ($\overline{ACK}$) is detected (synchronized with rising edge of $\overline{SCK}$)

Acknowledge enable bit (R/W)

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

Acknowledge trigger bit (W)

ACKT	If this bit is set after end of transfer, $\overline{ACK}$ is output in synchronization with next $\overline{SCK}$. This bit is automatically cleared to 0 after $\overline{ACK}$ signal has been output.

Cautions 1. Do not set this bit to 1 before the end of serial transfer or during transfer.

2. ACKT cannot be cleared by software.

3. To set ACKT, clear ACKE to 0.

Command detection flag (R)

CMDD	Clear Condition (CMDD = 0)	Set Condition (CMDD = 1)
	<1> When transfer start instruction is executed <2> When bus release signal (REL) is detected <3> When reset signal is input <4> CSIE = 0 (Refer to **Figure 5-60**.)	When command signal (CMD) is detected

Bus release detection flag (R)

RELD	Clear Condition (RELD = 0)	Set Condition (RELD = 1)
	<1> When transfer start instruction is executed <2> When reset signal is input <3> CSIE = 0 (Refer to **Figure 5-60**.) <4> When SVA and SIO do not coincide at time address is received	When bus release signal (REL) is detected

Command trigger bit (W)

CMDT	This bit controls output trigger of command signal (CMD). When this bit is set to 1, SO latch is cleared to 0. After that, the CMDT bit is automatically cleared to 0.

Caution Do not set SB0 (or SB1) during serial transfer. Be sure to set it before the start of or after the end of transfer.

Bus release trigger bit (W)

RELT	This bit controls output trigger of bus release signal (REL). When this bit is set to 1, SO latch is set to 1. After that, the RELT bit is automatically cleared to 0.

Caution Do not set SB0 (or SB1) during serial transfer. Be sure to set it before the start of or after the end of transfer.

(4) Selecting the serial clock

The serial clock is selected by using the bits 0 and 1 of the serial operation mode register (CSIM). The following four types of serial clocks can be selected:

Table 5-11. Selection of Serial Clock and Applications (in SBI mode)

Mode Register		Serial Clock		Timing at Which Shift Register Can Be Read/ Written and Serial Transfer Can Be Started	Application
CSIM1	CSIM0	Source	Masking Serial Clock		
0	0	External $\overline{SCK}$	Automatically masked at end of transfer of 8-bit data	<1> In operation enable mode (CSIE = 1) <2> If serial clock is masked after 8-bit serial transfer <3> When $\overline{SCK}$ is high	Slave CPU
0	1	TOUT F/F			Serial transfer at any speed
1	0	$f_X/2^4$			Medium-speed serial transfer
1	1	$f_X/2^3$			High-speed serial transfer

When the internal system clock is selected, $\overline{SCK}$ is internally stopped when $\overline{SCK}$ has been asserted and deasserted eight times. Externally, however, counting $\overline{SCK}$ continues until the slave enters the ready status.

(5) Signals

Figures 5-80 through 5-85 illustrate the operation of the signals in the SBI mode. Table 5-12 lists the signals used in the SBI mode.

Figure 5-80. RELT, CMDT, RELD, CMDD Operation (master)

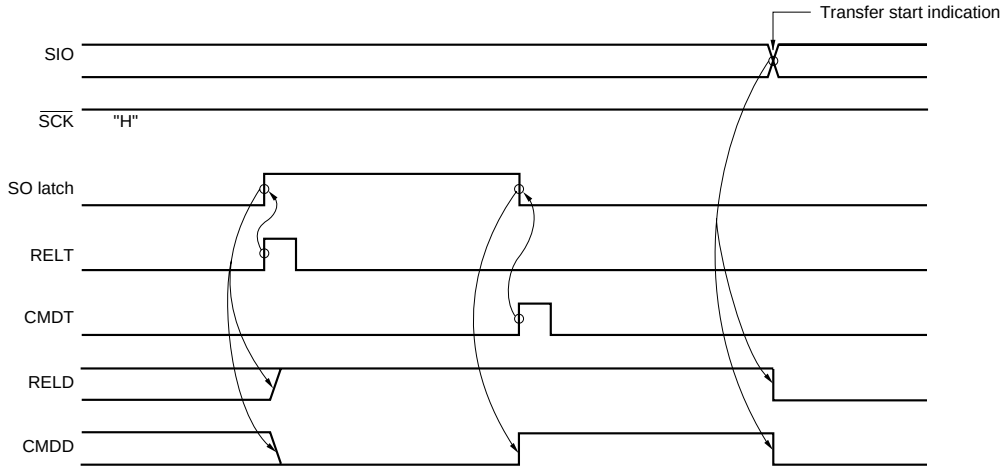


Figure 5-81. RELT, CMDT, RELD, CMDD Operation (slave)

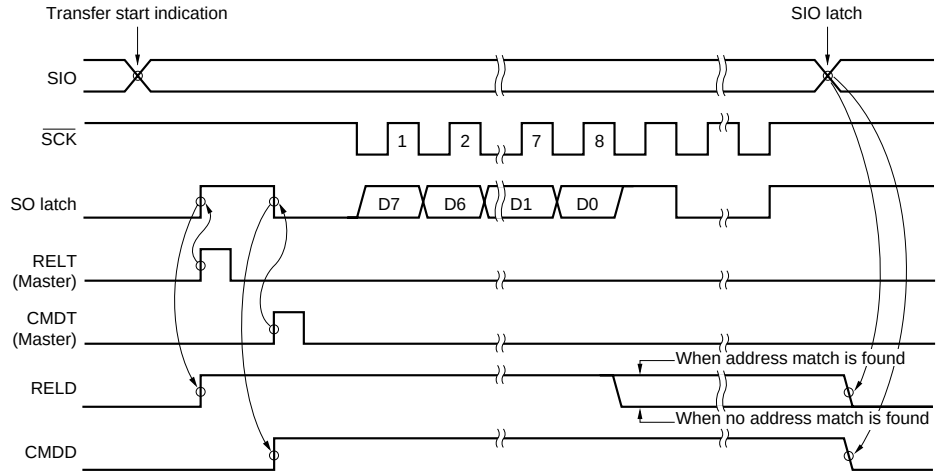
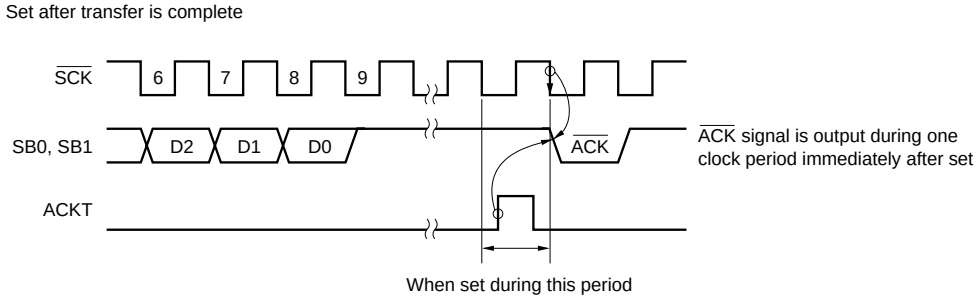


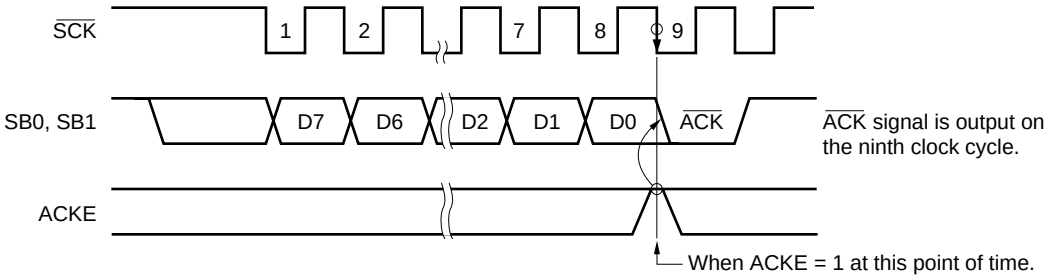
Figure 5-82. ACKT Operation



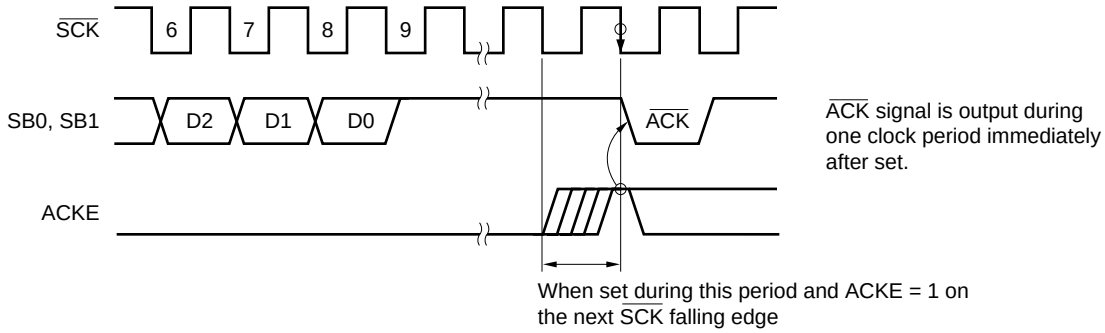
Caution Do not set ACKT before transfer terminates.

Figure 5-83. ACKE Operation

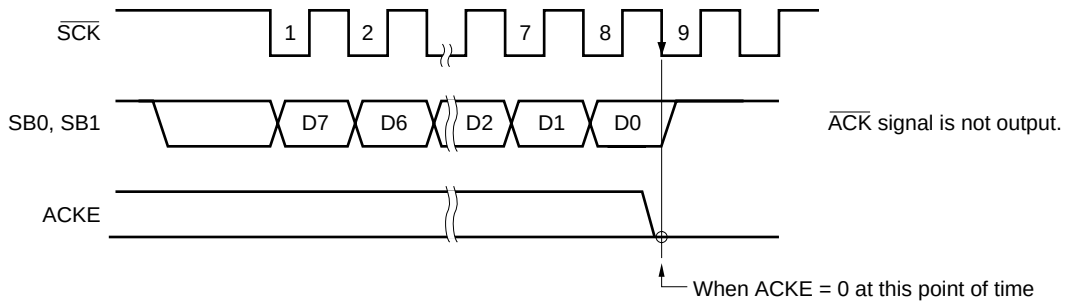
(a) When ACKE = 1 when transfer is complete



(b) When set after transfer is complete



(c) When ACKE = 0 when transfer is complete



(d) When the period during which ACKE = 1 is short

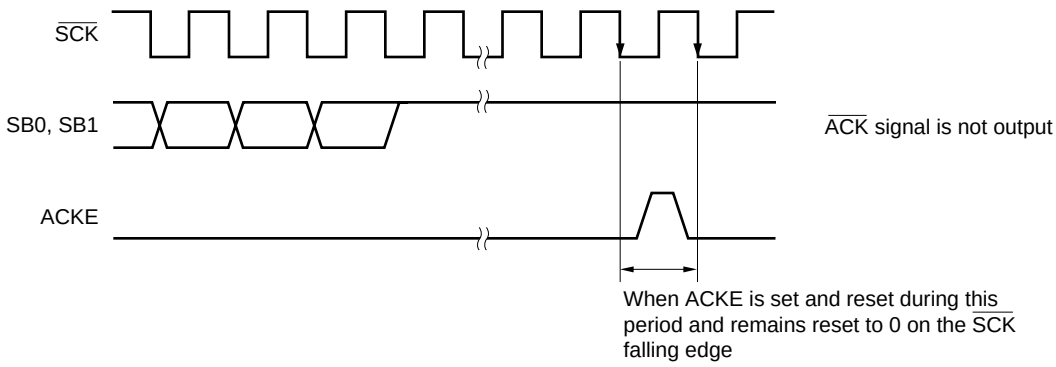
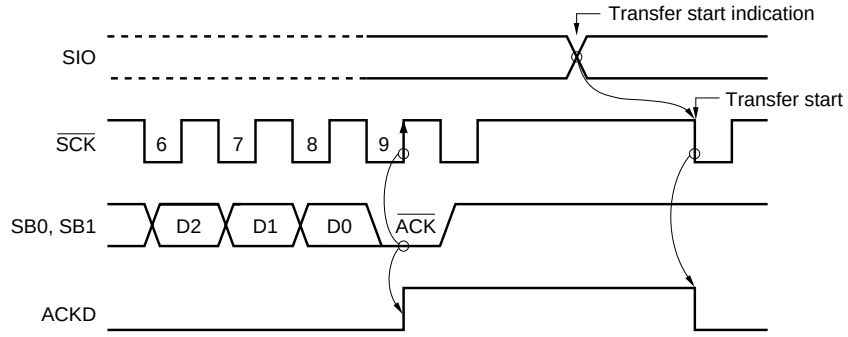
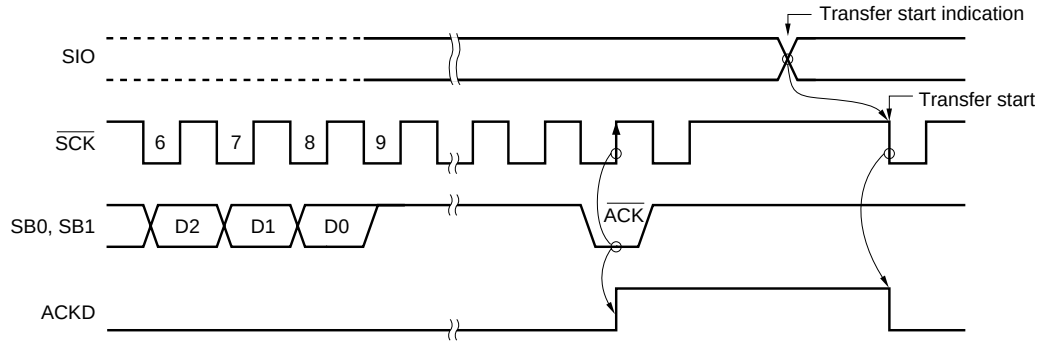


Figure 5-84. ACKD Operation

(a) When $\overline{ACK}$ signal is output during the period of the ninth $\overline{SCK}$ clock



(b) When $\overline{ACK}$ signal is output after the ninth $\overline{SCK}$ clock



(c) Reset timing when transfer start indication is given during BUSY

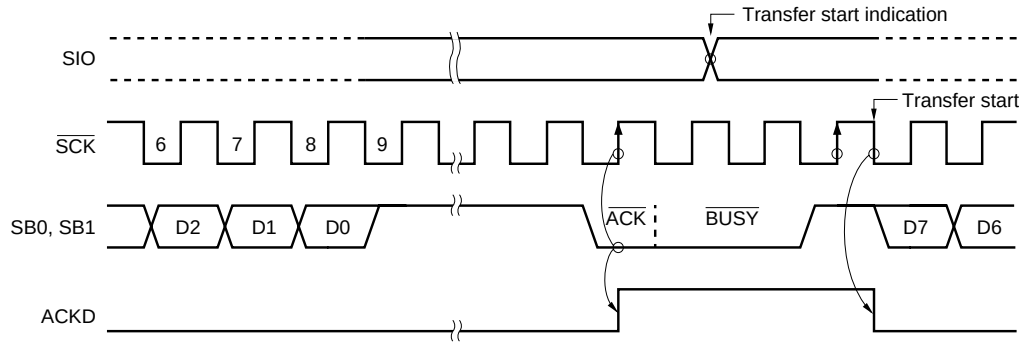


Figure 5-85. BSYE Operation

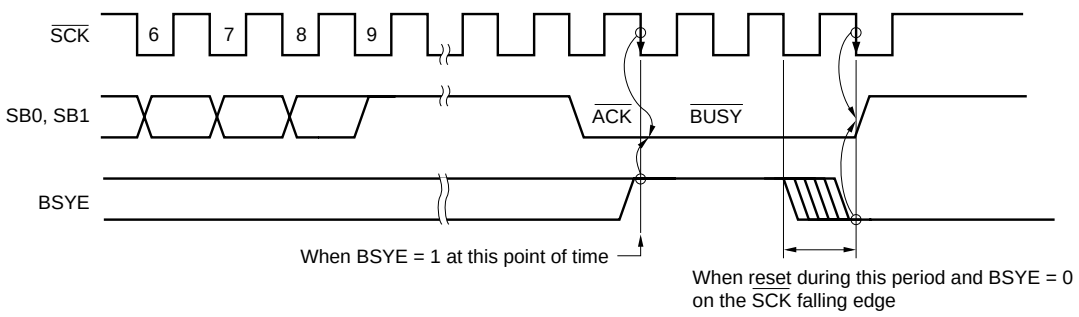


Table 5-12. Signal in SBI Mode (1/2)

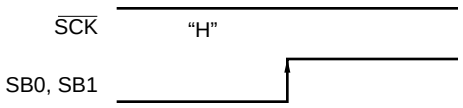
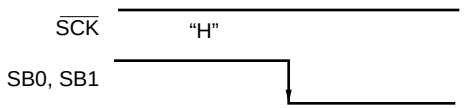
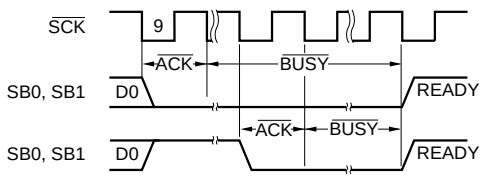
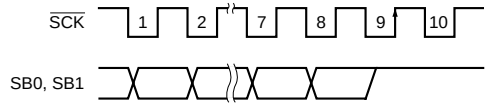
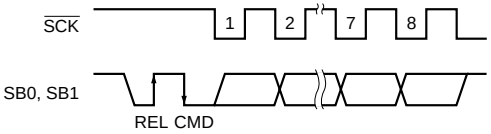
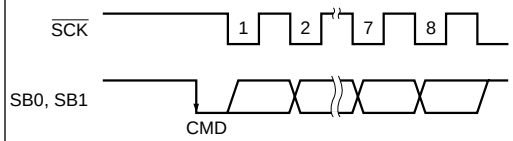
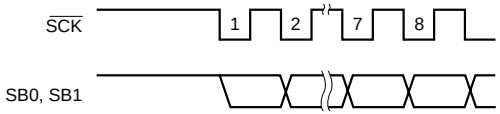
Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Flag Influence	Explanation
Bus release signal (REL)	Master	SB0, SB1 rising edge when $\overline{SCK}$ = 1		• Set RELT	• Set RELD • Clear CMDD	When this signal is followed by CMD signal output, it indicates that transmit data is address.
Command signal (CMD)	Master	SB0, SB1 falling edge when $\overline{SCK}$ = 1		• Set CMDT	• Set CMDD	(i) After REL signal is output, transmit data is address. (ii) REL signal is not output. Transmit data is command.
Acknowledge signal ($\overline{ACK}$)	Master/ slave	Low signal output to SB0, SB1 during the period of one clock $\overline{SCK}$ after completion of serial reception	[Synchronous busy signal] 	<1> ACKE = 1 <2> ACKT set	• Set ACKD	Completion of reception.
Busy signal ($\overline{BUSY}$)	Slave	[Synchronous busy signal] Low-level output to SB0, SB1 following acknowledge signal		• BSYE = 1	—	Serial reception cannot be done because processing is being performed.
Ready signal (READY)	Slave	High-level output to SB0, SB1 before start or after completion of serial transfer		<1> BSYE = 0 <2> Execution of SIO data write instruction (transfer start indication)	—	Serial reception is enabled.

Table 5-12. Signal in SBI Mode (2/2)

Signal Name	Output Device	Definition	Timing Chart	Output Conditions	Flag Influence	Explanation
Serial clock ($\overline{SCK}$)	Master	Synchronizing clock for output of address, command, data, $\overline{ACK}$ signal, synchronous $\overline{BUSY}$ signal, etc. Address, command, or data is transferred on the first eight.		Execution of data write instruction into SIO when CSIE = 1 (serial transfer start indication)**Note 2**	IRQCSI is set (on the rising edge of ninth clock)**Note 1**	Signal output timing to serial data bus
Address (A7 to A0)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after REL and CMD signals are output.				Slave device address value on serial bus
Command (C7 to C0)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after only CMD signal is output without REL signal output.				Indication, message to slave device
Data (D7 to D0)	Master or slave	8-bit data transferred in synchronization with $\overline{SCK}$ when neither REL nor CMD signal is output.				Data processed by slave or master

Notes 1. When WUP = 0, IRQCSI is always set on the rising edge of the ninth $\overline{SCK}$ clock.
When WUP = 1, IRQCSI is set on the rising edge of the ninth $\overline{SCK}$ clock only when address is received and matches the value in the slave address register (SVA).

2. In the $\overline{BUSY}$ state for the data transmission and reception (transfer), transfer is started after the READY state is entered.

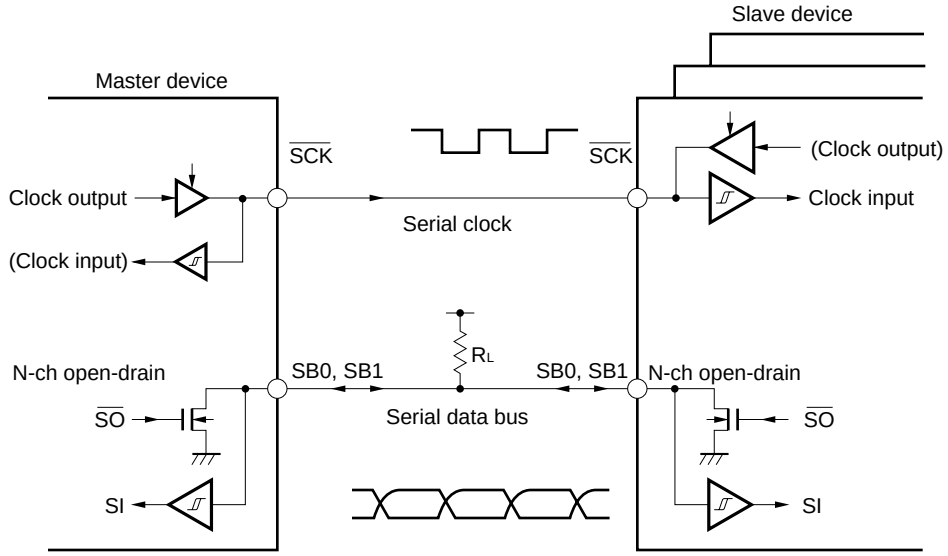
(6) Pin configuration

The configurations of the serial clock pin ($\overline{SCK}$) and serial data bus pin (SB0 or SB1) are as follows:

(a) $\overline{SCK}$ ..... Inputs or outputs serial clock
  <1> Master ..... CMOS, push-pull output
  <2> Slave ..... Schmitt input
(b) SB0, SB1 ..... Serial data I/O pin
  N-ch open-drain output and Schmitt input for both master and slave

Because the serial data bus line is of the N-ch open-drain output configuration, an external pull-up resistor must be connected to it.

Figure 5-86. Pin Configuration



Caution Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance. The transistor can be always turned off during transfer. However, if the wake-up function specification bit (WUP) = 1, the N-ch transistor is always off. Therefore, it is not necessary to write FFH to SIO before reception.

(7) Detection of address coincidence

In the SBI mode, the master transmits an address to select a specific slave and then starts communicating with the selected slave.

Whether the address transmitted to a slave coincides with the address of the slave is detected by the hardware of the slave. For this purpose, the slave is provided with a slave address register (SVA). In the wake-up status (WUP = 1), the slave sets IRQCSI only when the address transmitted from the master coincides with the value set in the SVA of the slave.

Cautions 1. Whether a slave is selected or not is detected by observing if there is a coincidence between the address transmitted from the master and the slave address of the slave after the bus has been released (RELD = 1).

For this coincidence detection, an address coincidence interrupt (IRQCSI) that is generated when WUP = 1 is usually used. Therefore, detect whether a slave is selected or not when WUP = 1.

2. To detect whether the slave is selected when WUP = 0 and without using an interrupt, do not detect the address coincidence, but transmit and receive a command determined by the program in advance.

(8) Error detection

In the SBI mode, because the status of the serial bus SB0 or SB1 during transmission is also loaded to the shift register SIO of the device transmitting data, an error can be detected by the following methods:

(a) By comparing SIO data before and after transmission

If the two data differ from each other, it can be assumed that a transmission error has occurred.

(b) By using slave address register (SVA)

The transmit data is placed in SIO and SVA and transmission is executed. After transmission, the COI bit (coincidence signal from the address comparator) of the serial operation mode register (CSIM) is tested. If this bit is “1”, the transmission has been completed normally. If it is “0”, it can be assumed that a transmission error has occurred.

(9) Communication operation

In the SBI mode, the master usually selects one of the slave devices with which it is to communicate, by outputting an “address” onto the serial bus.

After the master has selected the slave device, commands and data are transmitted and received between the master and slave. In this way, serial communication is implemented.

Figures 5-87 through 5-90 show the timing charts illustrating each kind of data communication.

In the SBI mode, the shift register performs shift operations in synchronization with the falling edge of the serial clock ($\overline{SCK}$), and the transmit data is latched to the SO latch and output from the SB0/P02 or SB1/P03 pin with the MSB first. The receive data input to the SB0 (or SB1) pin at the rising edge of $\overline{SCK}$ is latched to the shift register.

Figure 5-87. Address Transmission Operation from Master Device to Slave Device (when WUP = 1)

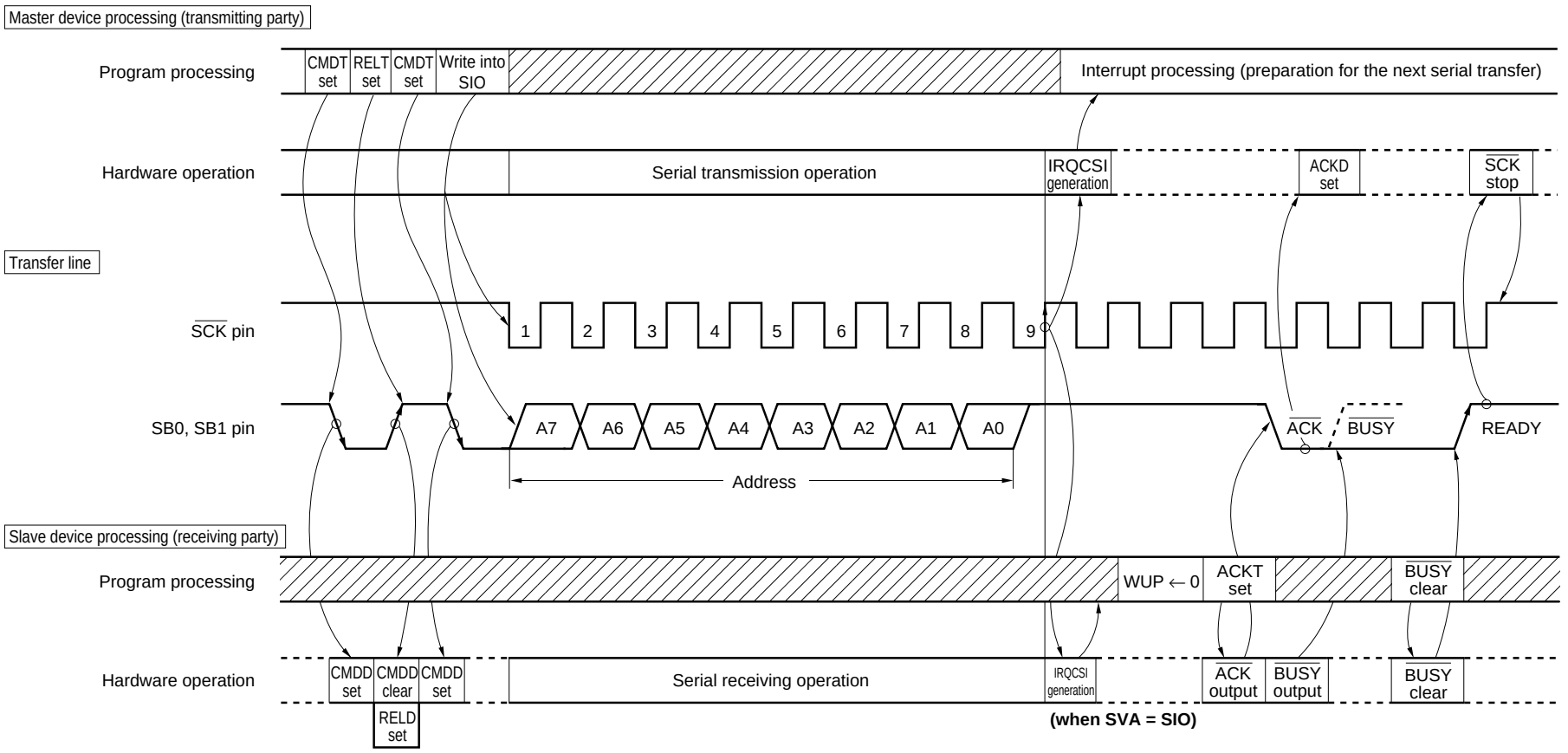


Figure 5-88. Command Transmission Operation from Master Device to Slave Device

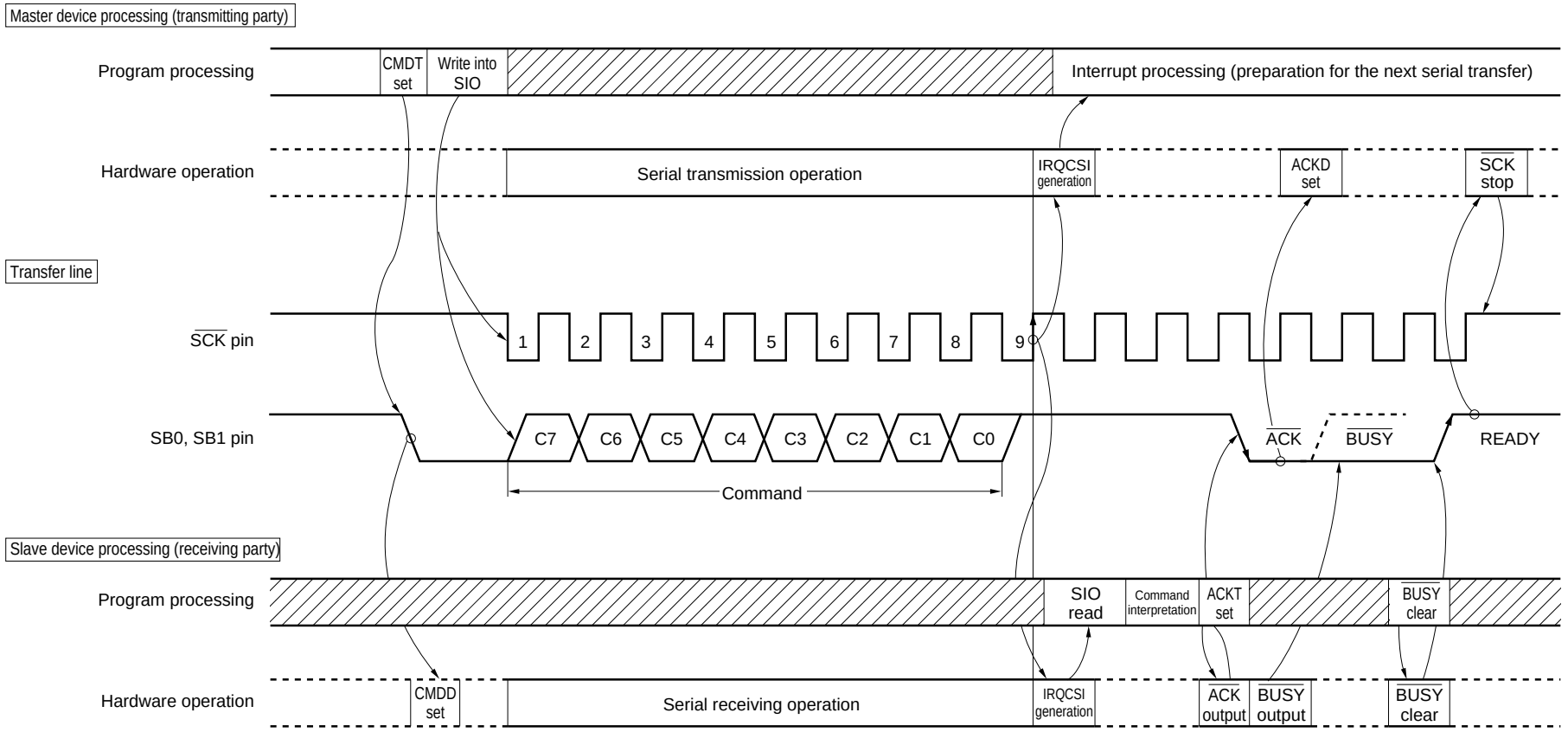


Figure 5-89. Data Transmission Operation from Master Device to Slave Device

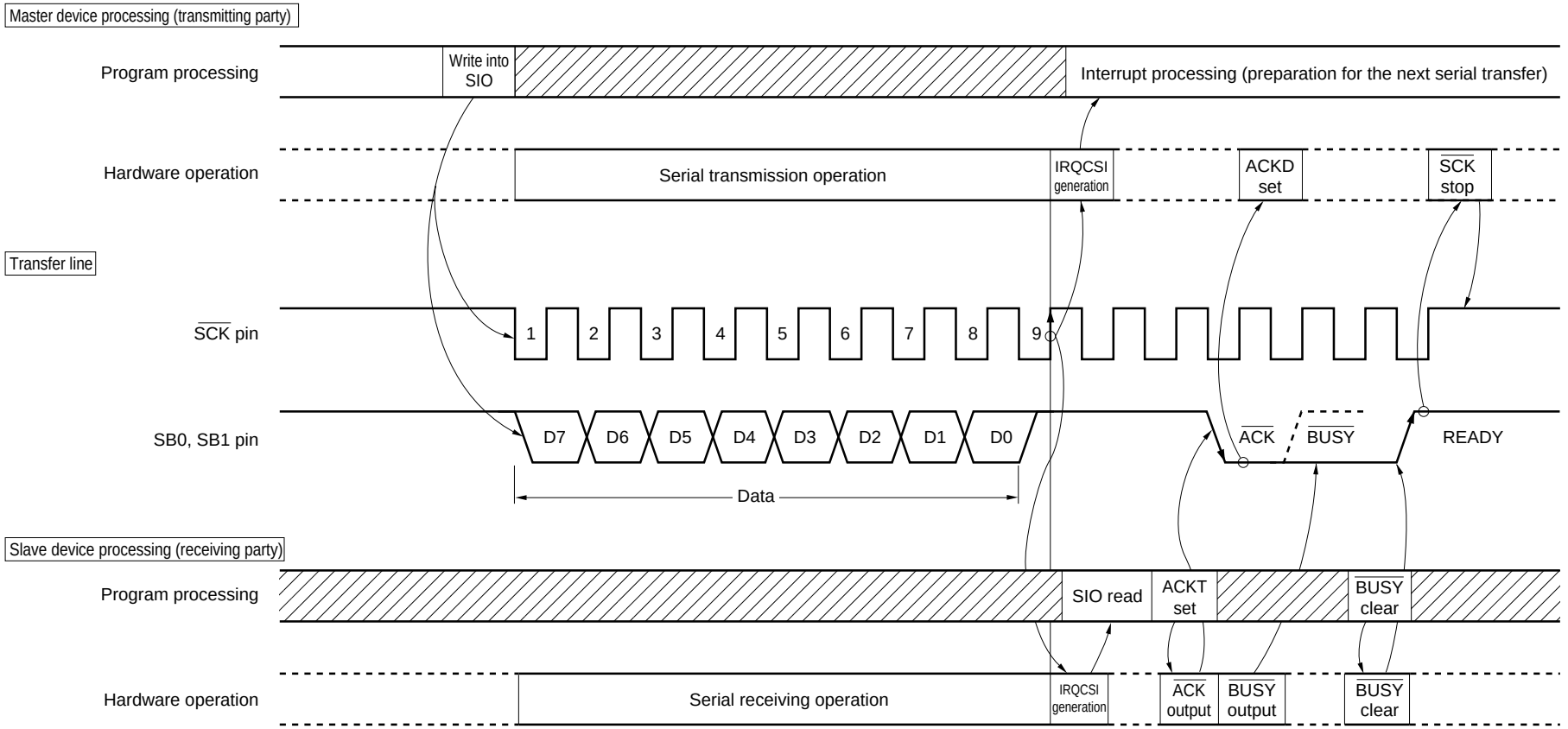
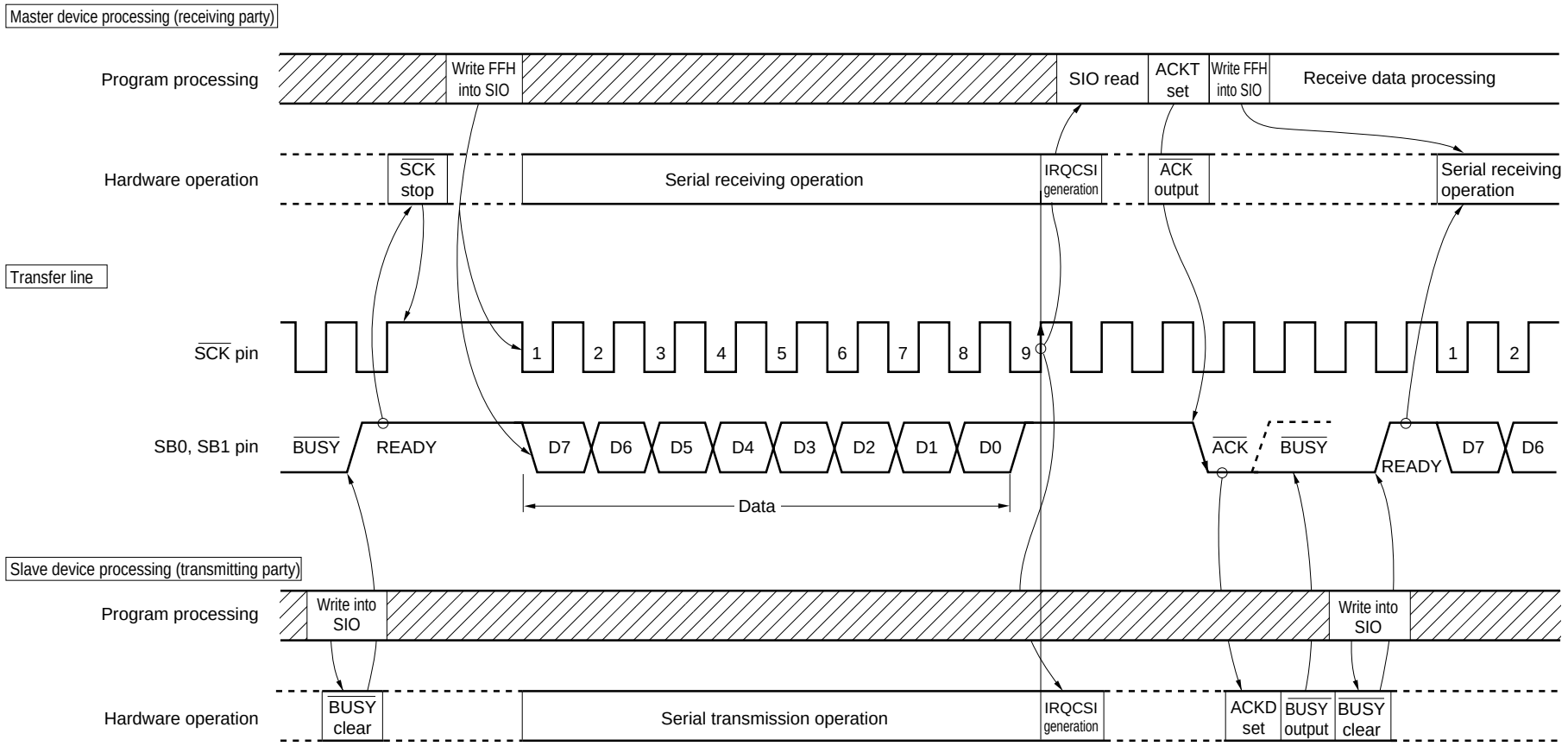


Figure 5-90. Data Transmission Operation from Slave Device to Master Device



(10) Starting transfer

Serial transfer is started when the transfer data is placed in the shift register (SIO), if the following two conditions are satisfied:

- Serial interface operation enable/disable bit (CSIE) = 1
- The internal serial clock is stopped after 8-bit serial transfer or $\overline{SCK}$ is high

Cautions 1. Transfer is not started even if CSIE is set to "1" after the data has been written to the shift register.

2. Because it is necessary to turn off the N-ch transistor when data is received, write FFH to SIO in advance.
When the wake-up function specification bit (WUP) = 1, however, it is not necessary to write FFH to SIO, because the N-ch transistor is always off.

3. If data is written to SIO while the slave is busy, the data is not lost.
When the SB0 (or SB1) input goes high and the slave becomes ready after the slave has been released from the busy status, transfer is started.

When an 8-bit transfer has been completed, the serial transfer is automatically stopped, and an interrupt request flag (IRQCSI) is set.

Example To transfer the RAM data addressed by the HL register and at the same time, load the SIO data in the accumulator, and start serial transfer

```
MOV   XA, @HL    ; Takes out transmit data from RAM
SEL   MB15       ; or CLR1 MBE
XCH   XA, SIO    ; Exchanges transmit data and receive data, and starts transfer
```

(11) Notes on SBI mode

(a) Whether a slave is selected or not is detected by observing if there is a coincidence between an address transmitted from the master after the bus has been released (RELD = 1) and the slave address of the slave. For this coincidence detection, an address coincidence interrupt (IRQCSI) that is generated when WUP = 1 is usually used. Therefore, detect whether a slave is selected or not by using the slave address when WUP = 1.

(b) To detect whether a slave is selected or not when WUP = 0 without using the interrupt, do not detect address coincidence, but transmit or receive a command set in advance by program.

(c) If WUP is set to 1 while the $\overline{\text{BUSY}}$ signal is output, the $\overline{\text{BUSY}}$ signal is not deasserted. In the SBI mode, the $\overline{\text{BUSY}}$ signal is output until the next serial clock ($\overline{\text{SCK}}$) falls after a command that deassert the $\overline{\text{BUSY}}$ signal has been issued. Before setting WUP to 1, be sure to deassert the $\overline{\text{BUSY}}$ signal and confirm that the SB0 (or SB1) pin has gone high.

(12) Application of SBI mode

This paragraph introduces an application example in which serial data communication is executed in the SBI mode. In this example, the μPD753108 can operate as both the master and slave CPU on the serial bus. Moreover, the master can be changed by a command.

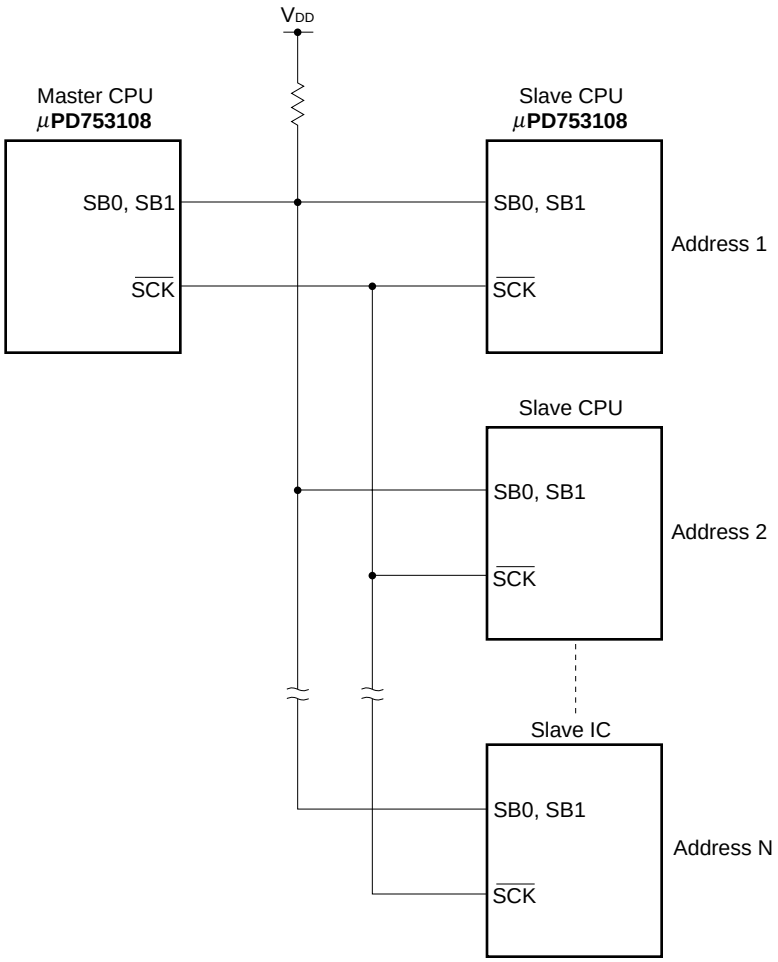
(a) Serial bus configuration

In the application example presented below, it is assumed that the μPD753108 is connected to the bus line as one of the devices in the serial bus.

The μPD753108 uses the serial data bus SB0 (or SB1) and serial clock $\overline{SCK}$ pins.

Figure 5-91 shows an example of serial bus configuration.

Figure 5-91. Example of Serial Bus Configuration



(b) Command description

<Types of commands>

In this application example, the following commands are used:

<1> READ : Transfers data from slave to master
<2> WRITE : Transfers data from master to slave
<3> END : Notifies slave of end of WRITE command
<4> STOP : Notifies slave that WRITE command has been aborted
<5> STATUS : Reads status of slave
<6> RESET : Unselects slave currently selected
<7> CHGMST : Relinquishes mastership to slave

<Communication procedure>

Communication between the master and a slave is carried out in the following procedure:

<1> The master transmits the address of a slave with which the master is to communicate, to select the slave (chip select).
The slave that has received the address returns $\overline{\text{ACK}}$ to start communication with the master (the slave is selected).

<2> Commands and data are transmitted between the master and the slave selected in <1>.
Note that the other slaves must be unselected, because commands and data are transmitted between the master and a slave on a one-to-one basis.

<3> Communication ends when the slave is unselected. The slave is unselected in the following cases:
- When the master transmits the RESET command, the selected slave is unselected.
- When the master is changed to a slave by the CHGMST command, the device changed to a slave is unselected.

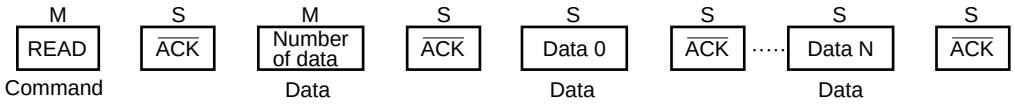
<Command format>

Here is the transfer format of each command:

<1> READ command

This command reads data from a slave. The number of data to be read varies from 1 to 256 bytes. The master specifies the number of data as a parameter. If 00H is specified as the number of data, 256 bytes of data is transferred.

Figure 5-92. Transfer Format of READ Command



Remark M : output by master
S : output by slave

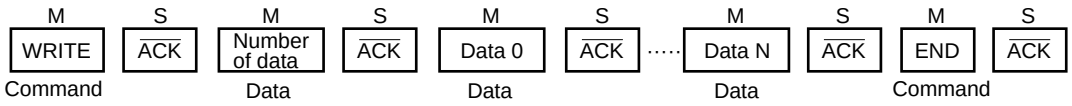
If the slave has more data than the number of data it has received, the slave returns $\overline{ACK}$; if not, the slave does not return $\overline{ACK}$, and an error occurs.

The master sends $\overline{ACK}$ to the slave each time it receives 1 byte.

<2> WRITE, END, and STOP commands

The WRITE command writes data to a slave. The number of data to be written is variable from 1 to 256 bytes. The master specifies the number of data as a parameter. If 00H is specified as the number of data, data of 256 bytes is transferred.

Figure 5-93. Transfer Formats of WRITE and END Commands



Remark M : output by master
S : output by slave

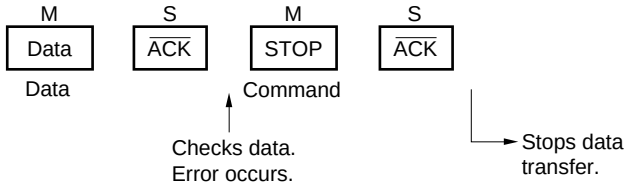
The slave returns $\overline{\text{ACK}}$ after it has received the number of data, if it has an enough area to store the received data. If the area runs short, the slave does not return $\overline{\text{ACK}}$, and an error occurs.

The master sends the END command after it has transferred all the data. This command notifies the slave that all the data have been correctly transferred.

The slave receives the END command even before all the data have been received. In this case, the data which has been received immediately before receiving the END command is valid.

The master compares the contents of SIO before and after transfer to check to see if the data have been correctly output onto the bus. If the contents of SIO before and after transfer differ, the master sends the STOP command to stop data transfer.

Figure 5-94. Transfer Format of STOP Command



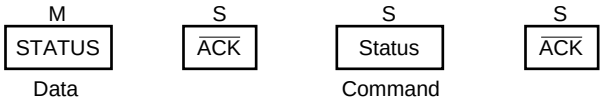
Remark M : output by master
S : output by slave

When the slave receives the STOP command, it invalidates 1-byte of the data received immediately before reception of the STOP command.

<3> STATUS command

This command reads the status of the slave currently selected.

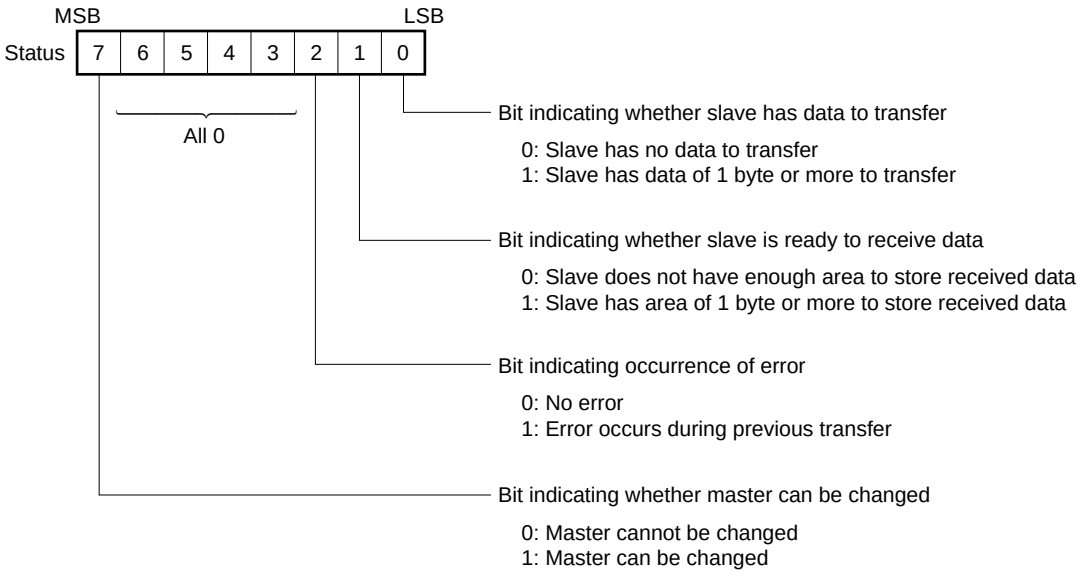
Figure 5-95. Transfer Format of STATUS Command



Remark M : output by master
S : output by slave

The format of the status returned by the slave is as follows:

Figure 5-96. Status Format of STATUS Command



The master returns $\overline{ACK}$ to slave when it has received the data of status.

<4> RESET command

This command unselects the slave currently selected. By sending the RESET command, the master can unselect all the slaves.

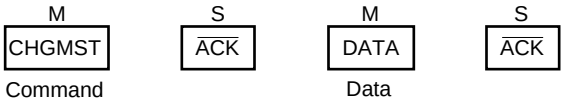
Figure 5-97. Transfer Format of RESET Command

Remark M : output by master
S : output by slave

<5> CHGMST command

This command gives the mastership to the slave currently selected.

Figure 5-98. Transfer Format of CHGMST Command



Remark M : output by master
S : output by slave

When the slave has received the CHGMST command, it decides whether it can receive mastership, and returns the following data to the master:

- FFH: Master can be changed
- 00H: Master cannot be changed

The slave compares the contents of SIO before and after transfer of data. If the SIO contents do not coincide, the slave does not return $\overline{ACK}$, and an error occurs.
The master returns $\overline{ACK}$ when it has received data. If the received data is FFH, the master starts operating as a slave. After the slave has sent data FFH and received $\overline{ACK}$ from the master, it starts operating as a master.

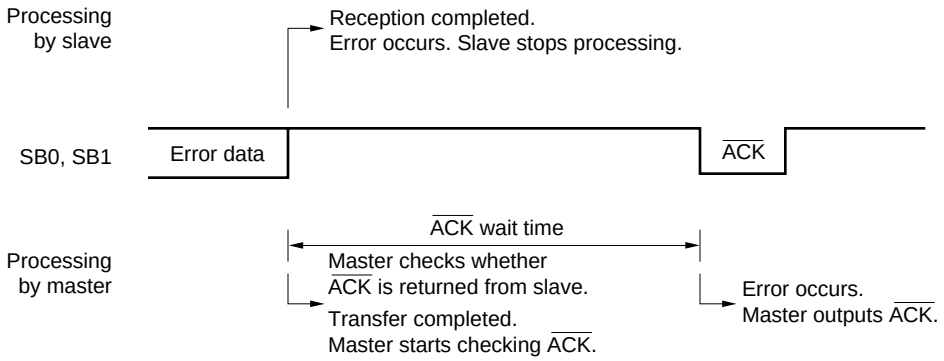
<Occurrence of error>

An error may occur during communication between the master and a slave.

If an error occurs, the slave notifies the master of error occurrence by not returning $\overline{ACK}$ to the master. If an error occurs only when the slave receives data, the slave sets the bit of the status that indicates occurrence of an error and cancels the processing of all the commands under execution.

The master checks whether the slave has returned $\overline{ACK}$ after it has completed transfer of 1 byte. If the slave does not return $\overline{ACK}$ in specific time after the master has completed transfer, the master judges that an error has occurred, and outputs a dummy $\overline{ACK}$ signal.

Figure 5-99. Operations of Master and Slave in Case of Error



The following types of errors may occur:

- Error occurs in slave
  - <1> If the transfer format of a command is wrong
  - <2> If an undefined command is received
  - <3> If the number of data to be transferred by the slave runs short when READ command is executed
  - <4> If the slave does not have an enough area to store data when the WRITE command is executed
  - <5> If the data transferred by the READ, STATUS, or CHGMST command changes

If any of the above occurs, the slave does not return $\overline{ACK}$.

- Error occurs in master
  When the data to be transferred by the master changes when the WRITE command is executed, the master sends the STOP command to the slave.

5.6.8 $\overline{\text{SCK}}$ pin output manipulation

The $\overline{\text{SCK}}$/P01 pin, which incorporates an output latch, can also produce static output by software control in addition to a normal serial clock.

The number of $\overline{\text{SCK}}$s can be set as desired by using software to control the P01 output latch (the SO/SB0/P02, and SI/SB1/P03 pins are controlled by setting the SBIC RELT and CMDT bits).

The $\overline{\text{SCK}}$/P01 pin output control method is described below:

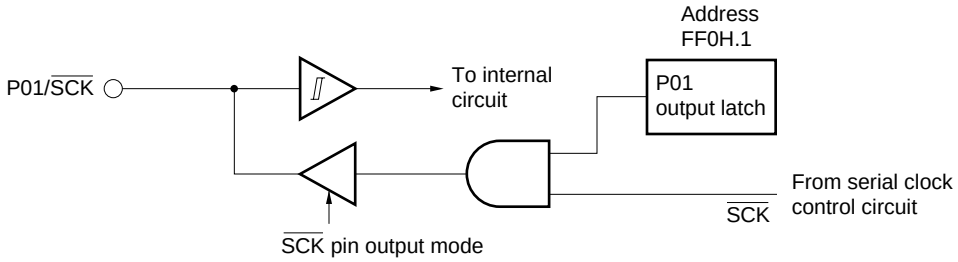
<1> The serial operation mode register (CSIM) is set ($\overline{\text{SCK}}$ pin: output mode). $\overline{\text{SCK}}$ from serial clock control circuit is set to 1 during serial transfer stop.

<2> The P01 output latch is controlled by using bit manipulation instructions.

Example To output one clock pulse to $\overline{\text{SCK}}$/P01 pin by using software.

```
SEL     MB15            ; or CLR1 MBE
MOV     XA, #10000011B  ; SCK (fx/2^3), output mode
MOV     CSIM, XA
CLR1    0FF0H.1         ; SCK/P01 ← 0
SET1    0FF0H.1         ; SCK/P01 ← 1
```

Figure 5-100. $\overline{\text{SCK}}$/P01 Pin Configuration



The P01 output latch is mapped in bit 1 of address FF0H. When the $\overline{\text{RESET}}$ signal is generated, the P01 output latch is set to 1.

Cautions 1. The P01 output latch must be set to 1 during normal serial transfer.

2. Do not use "PORT0.1" to specify the P01 output latch address. Write directly the address (0FF0H.1) in operand or specify SCKP. At that time, set MBE to 0, or set MBE to 1 and MBS to 15 in advance.

Do not use		Use	
CLR1	PORT0.1	CLR1	0FF0H.1
SET1	PORT0.1	SET1	0FF0H.1
		CLR1	SCKP
		SET1	SCKP

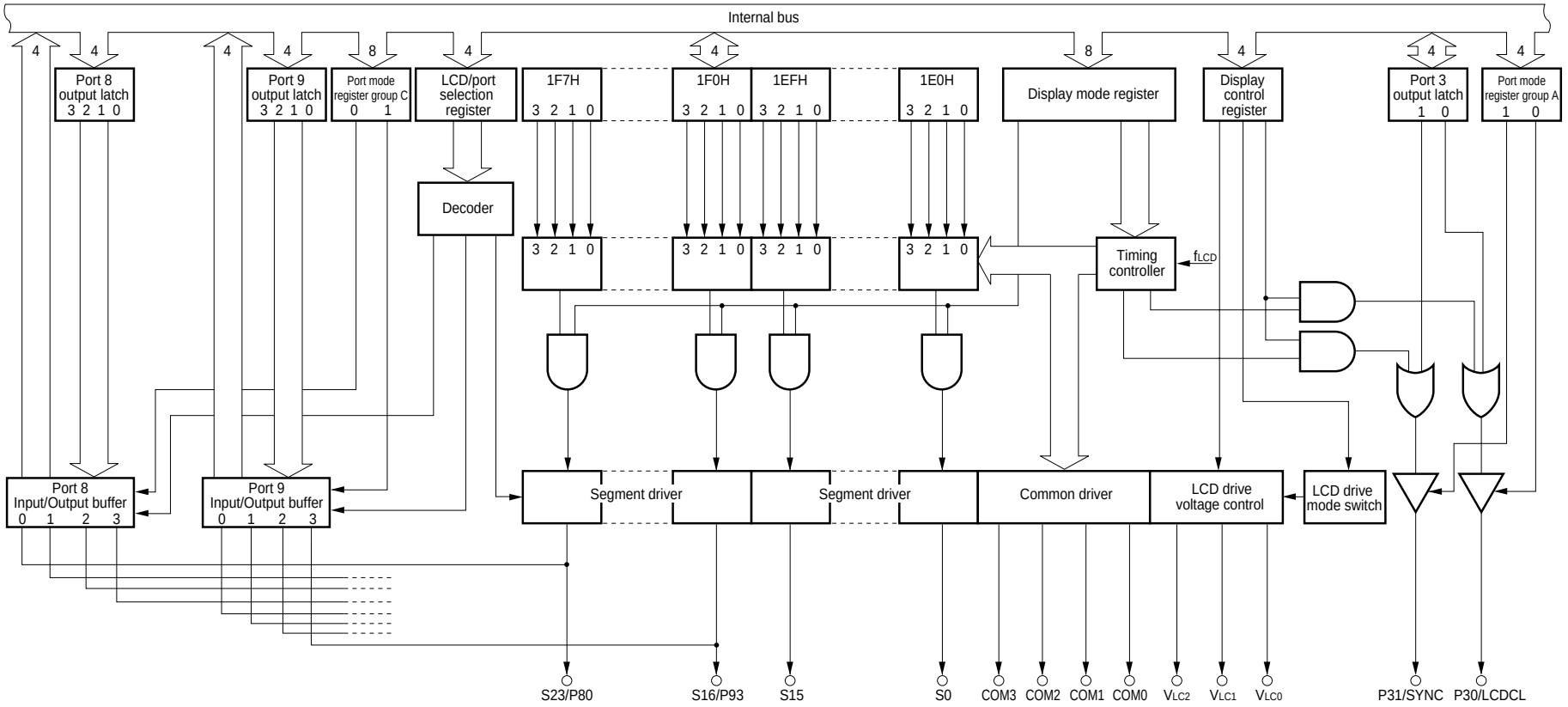
5.7 LCD Controller/Driver

5.7.1 LCD controller/driver configuration

The μPD753108 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 5-101 shows the LCD controller/driver configuration.

Figure 5-101. LCD Controller/Driver Block Diagram



5.7.2 LCD controller/driver functions

The μPD753108 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 (S16 to S23) can be changed to the I/O ports (PORT8 and PORT9).
(f) Split resistor can be incorporated to supply LCD drive power (mask option).
- Various bias methods and LCD drive voltages can be applicable.
- When display is off, current flowing through the split resistor is cut.

(g) Display data memory not used for display can be used for normal data memory.
(h) It can also operate by using the subsystem clock.

Table 5-13 lists the maximum number of picture elements that can be displayed in each display mode.

Table 5-13. 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 × common 1)Note 1
1/2	2	COM0, 1	48 (segment 24 × common 2)Note 2
	3	COM0, 1, 2	72 (segment 24 × common 3)Note 3
1/3	3		
	4	COM0, 1, 2, 3	96 (segment 24 × 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).

5.7.3 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 5-102. 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	$f_W/2^9$ (64 Hz)
0	1	$f_W/2^8$ (128 Hz)
1	0	$f_W/2^7$ (256 Hz)
1	1	$f_W/2^6$ (512 Hz)

Note LCDCL is supplied only when the watch timer operates. To use the LCD controller, 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	×	×	×	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 the above				Setting prohibited	

Note All segment signals are unselected.

Frame frequency (Hz)

LCDCL / Display Duty	$f_W/2^9$ (64 Hz)	$f_W/2^8$ (128 Hz)	$f_W/2^7$ (256 Hz)	$f_W/2^6$ (512 Hz)
Static	64	128	256	512
1/2	32	64	128	256
1/3	21	43	85	171
1/4	16	32	64	128

When f_W = 32.768 kHz

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

5.7.4 Display control register (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.
- Enables/disables synchronization clock (LCDCL) and synchronization signal (SYNC) outputs to the external segment signal extending controller/driver.
- Switching of LCD drive modes (normal mode and low-voltage mode) by supply voltage
  Normal mode ..... Low current consumption.
  Low-voltage mode ..... For low voltage operation.

Caution To drive LCD at $V_{DD} \leq 2.2$ V, be sure to set the low-voltage mode.

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 5-103. Format of Display Control Register

Address	3	2	1	0	Symbol
F8EH	0	LCDC2	VAC0	LCDC0	LCDC

Output enable/disable specification bit for LCDCL and SYNC signal

LCDC2	0	Disables the output of LCDCL and SYNC signal.
	1	Enables the output of LCDCL and SYNC signal.

Caution The LCDCL and SYNC signal are provided for system extension in the future. Currently, disable the signal output.

LCD drive mode select bit

VAC0	0	Normal mode ($2.2\ V \leq V_{DD} \leq 5.5\ V$)
	1	Low-voltage mode ($1.8\ V \leq V_{DD} \leq 5.5\ V$)

Display output status by LCDC0 and LCDM3

LCDC0	0	1	
LCDM3	×	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 S15 S16 to S23 segments specification pin	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).
S16 to S23 bit 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 (BIAS pin output)	Off (high-impedance)**Note**	On (high-level)**Note**	On (high-level)**Note**

Note The descriptions in the parentheses apply to the case where the split resistors are not used.

5.7.5 LCD/port selection register (LPS)

The LCD/port selection register (LPS) switches the segment signal output (S16 to S23) to the input/output port.

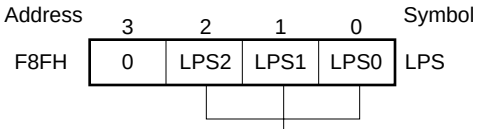
Of segment signal outputs, S16 to S19 are also used as PORT9 and S20 to S23, as PORT8. Four outputs each are switched together as one unit.

By setting the LPS value to “0000”, S16 to S23 can be switched to input/output ports (PORT9 and PORT8).

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

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

Figure 5-104. LCD/Port Selection Register Format



			Usable Pin	
			Segment Output Pin	Port Pin
0	0	0	—	P93 to P90, P83 to P80
0	0	1	S16 to S19	P83 to P80
0	1	0	S16 to S19, S20 to S23	—
Other than the 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 0001 or 0010 when using the LCD.

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

3. If the S16/P93 through S19/P90 and S20/P83 through S23/P80 pins are used as segment signal outputs, do not enable the on-chip pull-up resistors for these pins by software. 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. This is why specifying on-chip pull-up resistor connection by software is unnecessary. When an input instruction is executed for the port specified as a segment signal output, the content of the output latch will be input.

5.7.6 Display data memory

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

The display data memory is an 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 can 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 5-106 shows the relationship between the display data memory bits and segment output.

Figure 5-105. Data Memory Map

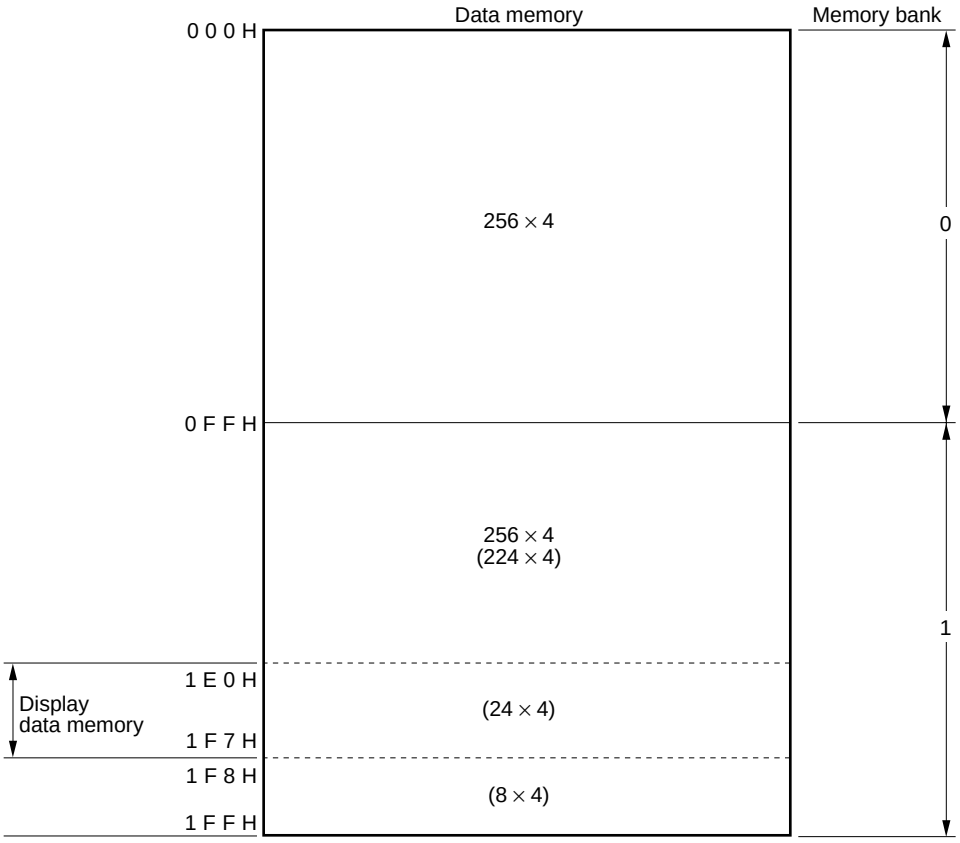
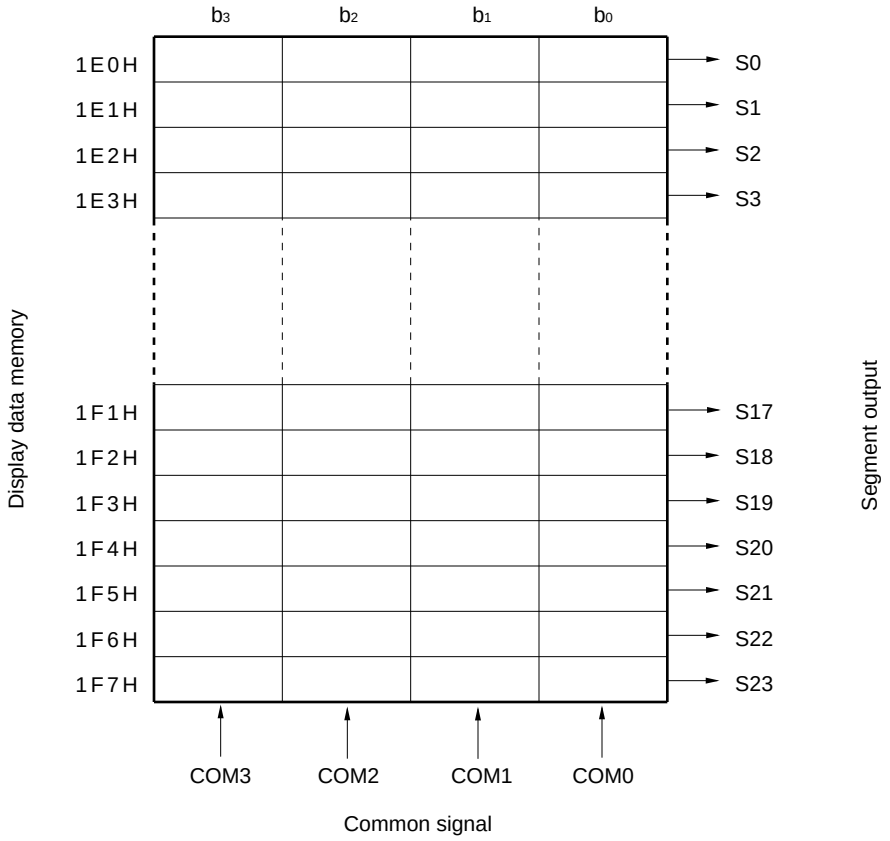


Figure 5-106. Relationship between Display Data Memory and Common Segments



5.7.7 Common signal and segment signal

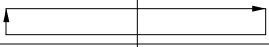
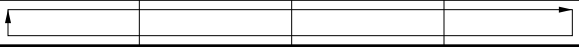
The individual picture elements of the LCD panel light up when the potential difference between the corresponding common signal and segment signal is greater than a predetermined value (LCD driving voltage V_{LCD}). When it goes lower than the V_{LCD} or is zero, they go out.

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 5-14 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 5-14. Common Signal

Common 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 to 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.

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, therefore they can be used for purposes other than display.

(3) Common and segment signal output waveforms

Tables 5-15 to 5-17 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 5-15. 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 5-16. LCD Drive Voltage (1/2 Bias method)

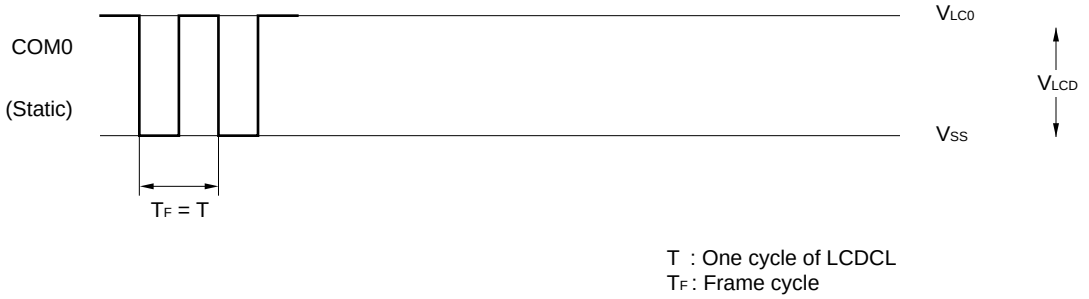
Segment Signal Sn / Common Signal COMm		Selection	Non-selection
		V_{LC0}/V_{SS}	V_{SS}/V_{LC0}
Selection	V_{SS}/V_{LC0}	$\boxed{+V_{LCD}/-V_{LCD}}$	0 V/0 V
Non-selection	$V_{LC1} = V_{LC2}$	$+\frac{1}{2}V_{LCD}/-\frac{1}{2}V_{LCD}$	$-\frac{1}{2}V_{LCD}/+\frac{1}{2}V_{LCD}$

Table 5-17. 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}	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$	$+\frac{1}{3}V_{LCD}/-\frac{1}{3}V_{LCD}$

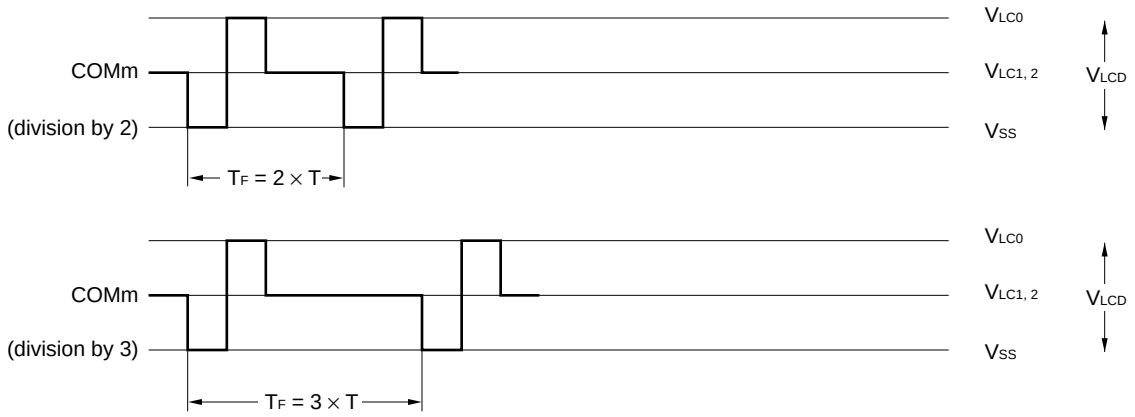
Figures 5-107 to 5-109 show the common signal waveforms. Figure 5-110 shows the common and segment signal electric potentials and phases.

Figure 5-107. Common Signal Waveform (Static)



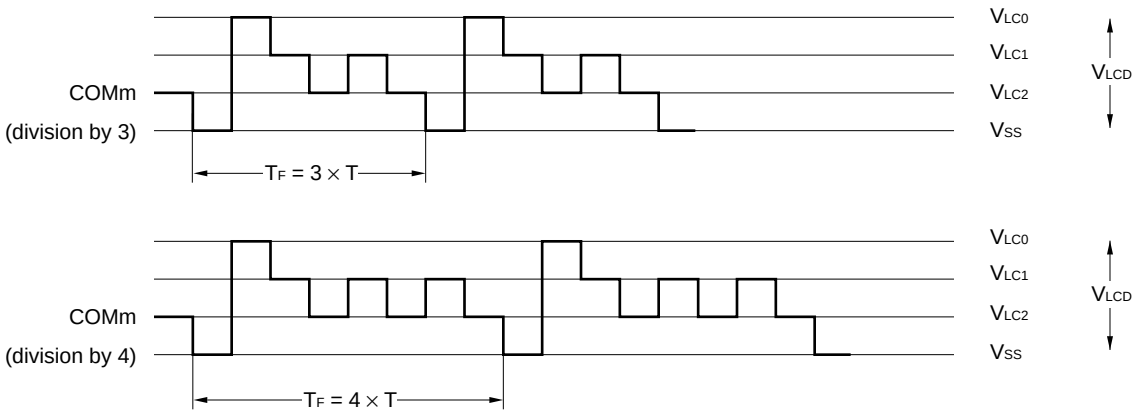
T : One cycle of LCDCL
T_F : Frame cycle

Figure 5-108. Common Signal Waveform (1/2 Bias method)



T : One cycle of LCDCL
T_F : Frame cycle

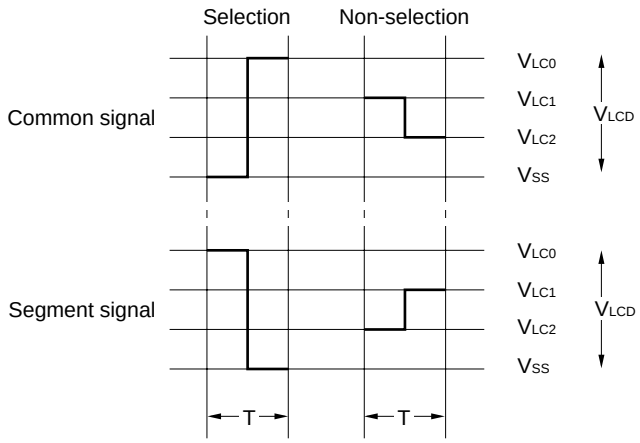
Figure 5-109. Common Signal Waveform (1/3 Bias method)



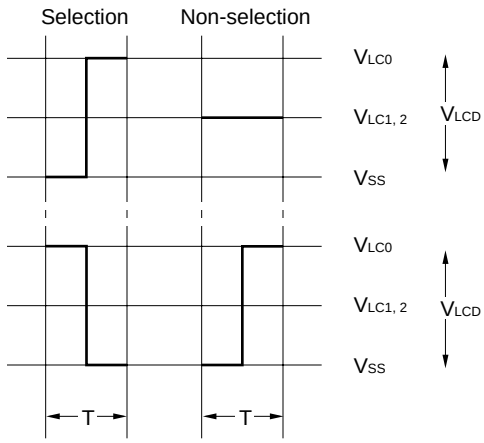
T : One cycle of LCDCL
T_F : Frame cycle

Figure 5-110. Common and Segment Signal Electric Potentials and Phases

(a) 1/3 bias method

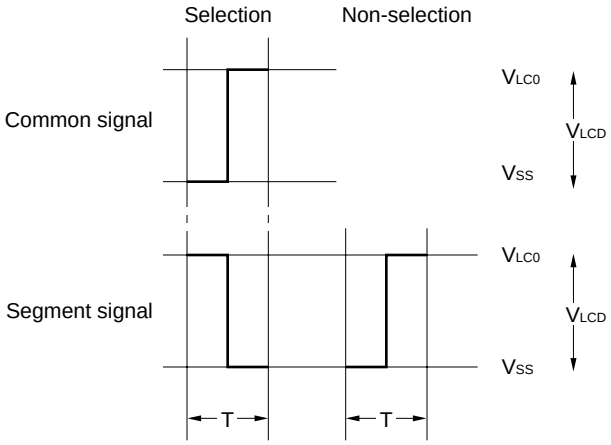


(b) 1/2 bias method



T : One cycle of LCDCL

(c) Static display mode



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

In the μPD753108, a split resistor can be incorporated in the V_{LC0} to V_{LC2} pins for the LCD drive power supply. According to the bias method, the LCD drive power can be supplied without an external split resistor. The μPD753108 also includes the BIAS pin to deal with various LCD drive voltages. The BIAS and V_{LC0} pins are connected externally.

Table 5-18 lists proper LCD drive power supply values based on the static, 1/2, and 1/3 bias methods.

Table 5-18. LCD Drive Power Supply Values

Bias Method / LCD Drive Power	No Bias (Static Mode)	1/2	1/3
V_{LC0}	V_{LCD}	V_{LCD}	V_{LCD}
V_{LC1}	$2/3V_{LCD}$	$1/2V_{LCD}$ **Note**	$2/3V_{LCD}$
V_{LC2}	$1/3V_{LCD}$		$1/3V_{LCD}$
V_{SS}	0 V	0 V	0 V

Note When 1/2 bias is used, the V_{LC1} and V_{LC2} pins must be connected externally.

Remark When the BIAS and V_{LC0} pins are not connected, $V_{LCD} = 3/5V_{DD}$ (internal split resistor must be specified by using a mask option).
When the BIAS and V_{LC0} pins are connected, $V_{LCD} = V_{DD}$.

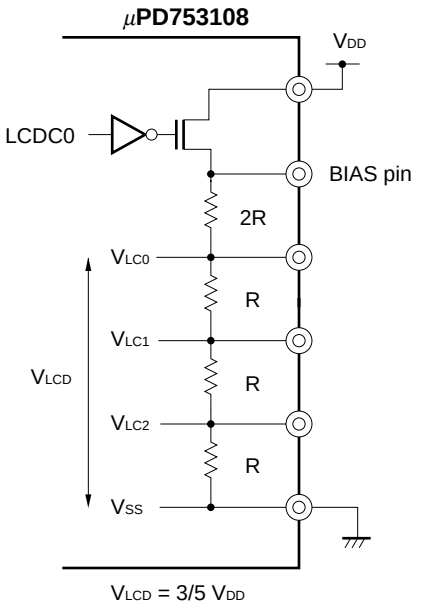
Figure 5-111, (a) to (c) show LCD drive power supply examples according to Table 5-18.

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

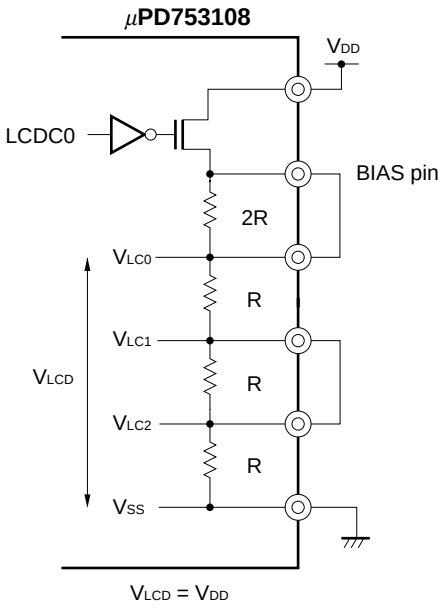
This LCD power on/off control is also useful to prevent DC voltage from being applied to LCD (if the system clock subsystem is selected) when the LCD clock is stopped by execution of a STOP instruction, when the watch timer operates using the main system clock. That is, display control register bit 0 (LCDC0) is cleared 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, the LCD display can be connected.

Figure 5-111. LCD Drive Power Connection Examples (when split resistor is incorporated)

(a) 1/3 bias method and static display mode (In example V_{DD} = 5 V, V_{LCD} = 3 V)



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



(c) 1/3 bias method and static display mode (In example V_{DD} = 5 V, V_{LCD} = 5 V)

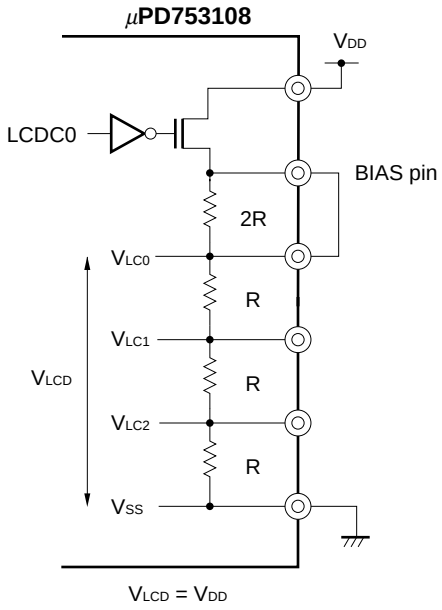
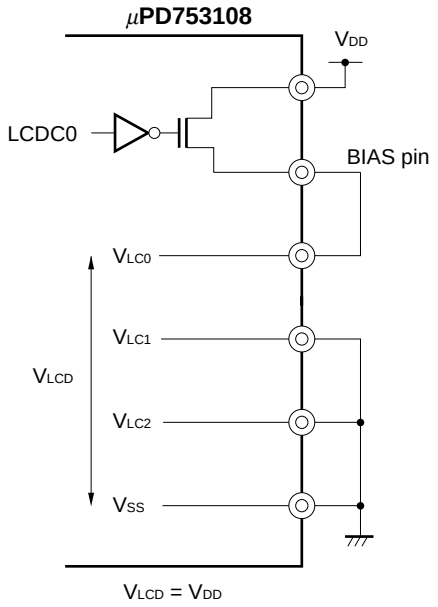
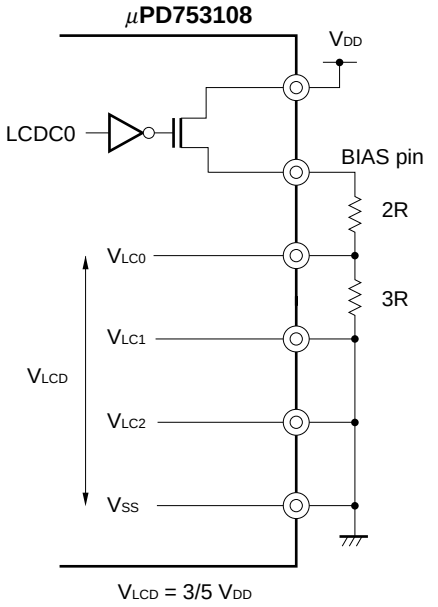


Figure 5-112. LCD Drive Power Connection Examples (when split resistor is connected externally)

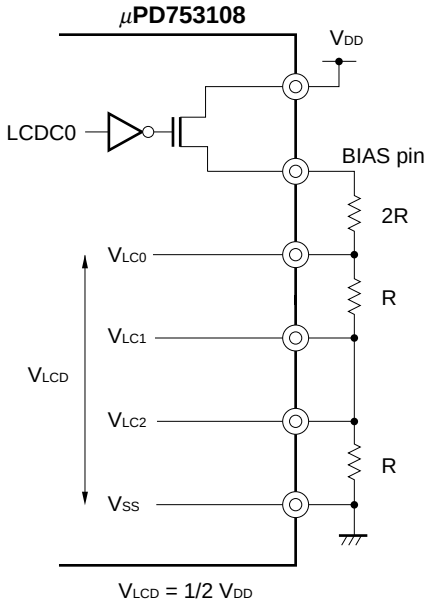
(a) Static display mode[Note]
(In example V_{DD} = 5 V, V_{LCD} = 5 V)



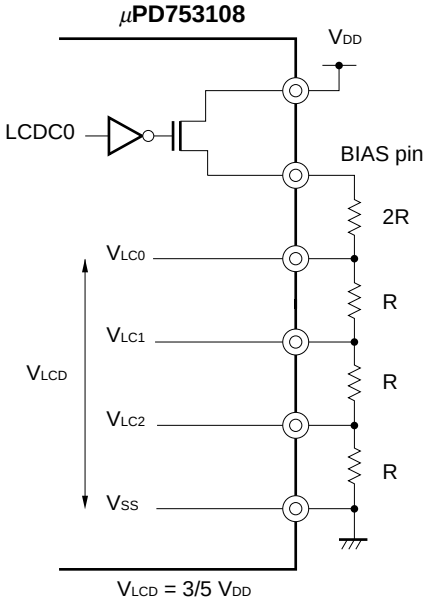
(b) Static display mode
(In example V_{DD} = 5 V, V_{LCD} = 3 V)



(c) 1/2 bias method
(In example V_{DD} = 5 V, V_{LCD} = 2.5 V)



(d) 1/3 bias method
(In example V_{DD} = 5 V, V_{LCD} = 3 V)



Note Set LCDC0 always to 1 (including the case of standby mode).

5.7.9 Display mode

(1) Static display example

Figure 5-114 shows connection of a static 3-digit LCD panel having the display pattern shown in Figure 5-113, the μPD753108 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 selection and non-selection voltages as shown in Table 5-19 to the S8 to S15 pins at the COM0 common signal timing according to the display pattern shown in Figure 5-113.

Table 5-19. 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 string of display data memory addresses 1E8H to 1EFH corresponding to S8 to S15 needs to be set to 10110111.

Figure 5-115 shows the S11, S12, and COM0 LCD drive waveforms. This shows that alternating current square wave of $+V_{LCD}/-V_{LCD}$ that is LCD on level is generated when S11 becomes the selection voltage at the selection timing of COM0.

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 5-113. Static Mode LCD Display Pattern and Electrode Connection

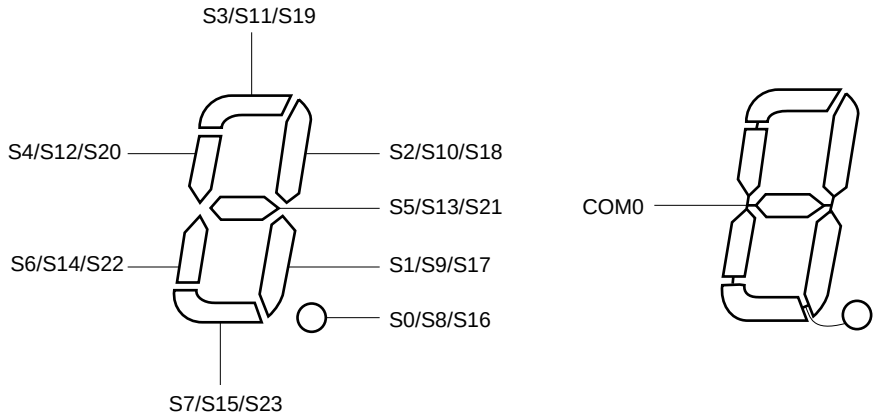


Figure 5-114. Static LCD Panel Connection Example

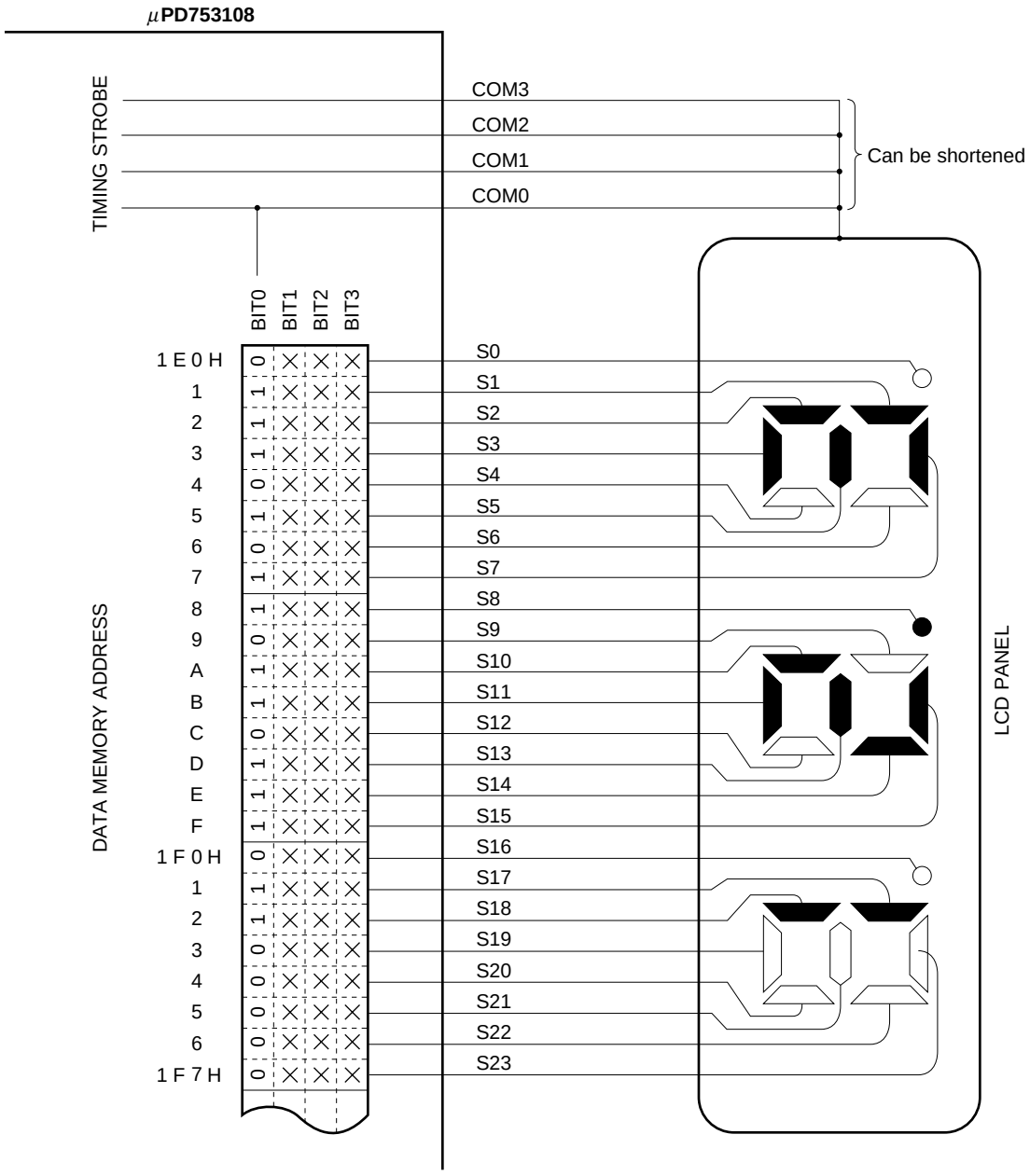
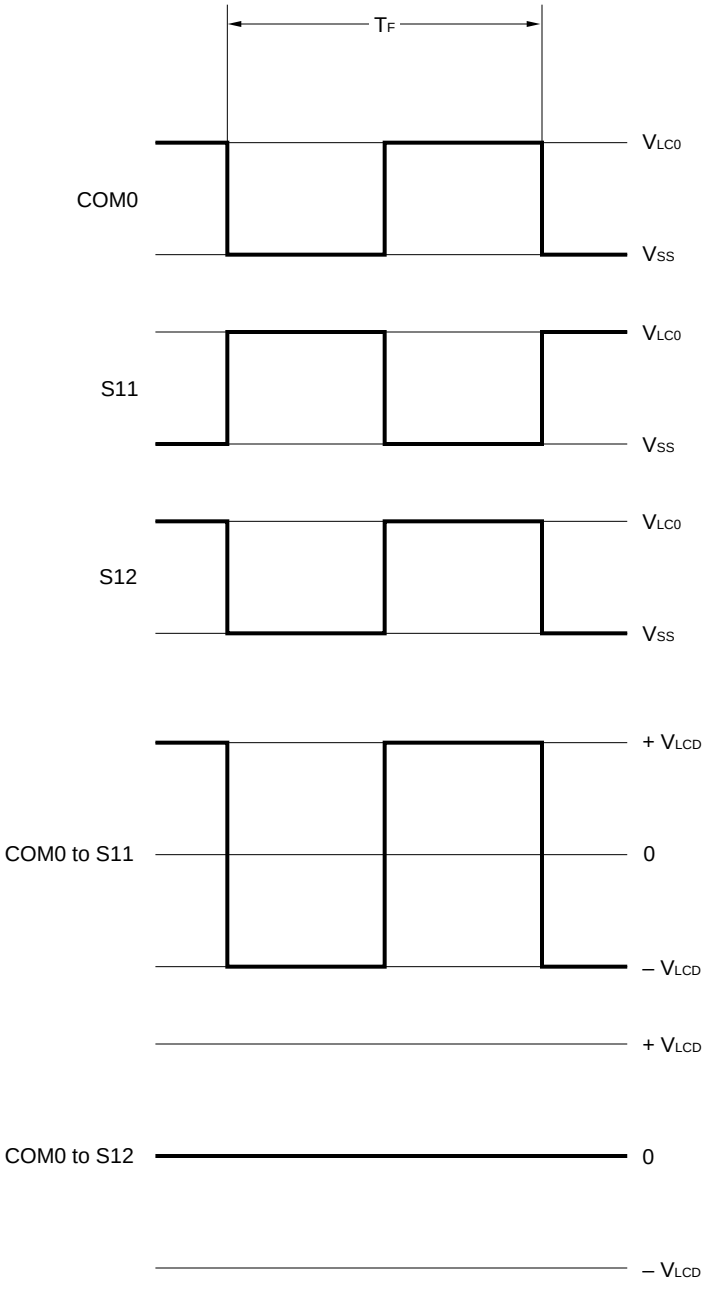


Figure 5-115. Static LCD Drive Waveform Example



(2) Division by 2 display example

Figure 5-117 shows connection of the division by 2 mode 6-digit LCD panel having the display pattern shown in Figure 5-116, the μPD753108 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 third digit position is taken as an example. It is necessary to output the selection and non-selection voltages as shown in Table 5-20 to the S12 to S15 pins at the timing of the COM0 and COM1 common signal according to the display pattern shown in Figure 5-116.

Table 5-20. 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 bits corresponding to S15 need to be set to ××10.

Figure 5-118 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 the LCD on level being generated when S15 is the selection voltage at the COM1 selection timing.

Figure 5-116. Division by 2 Mode LCD Display Pattern and Electrode Connection

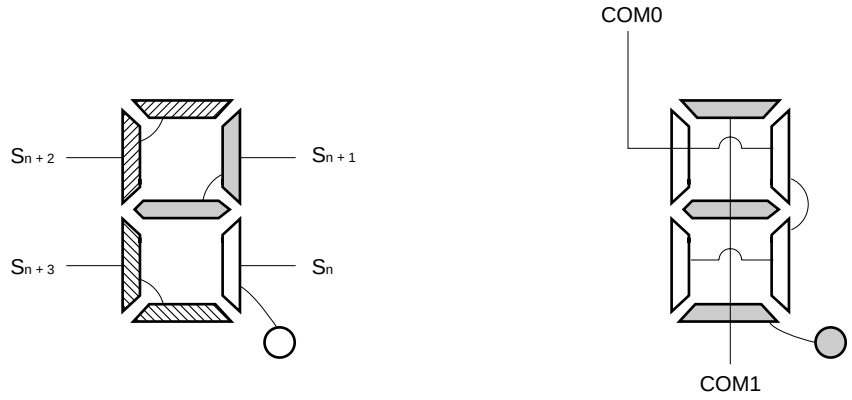


Figure 5-117. Division by 2 LCD Panel Connection Example

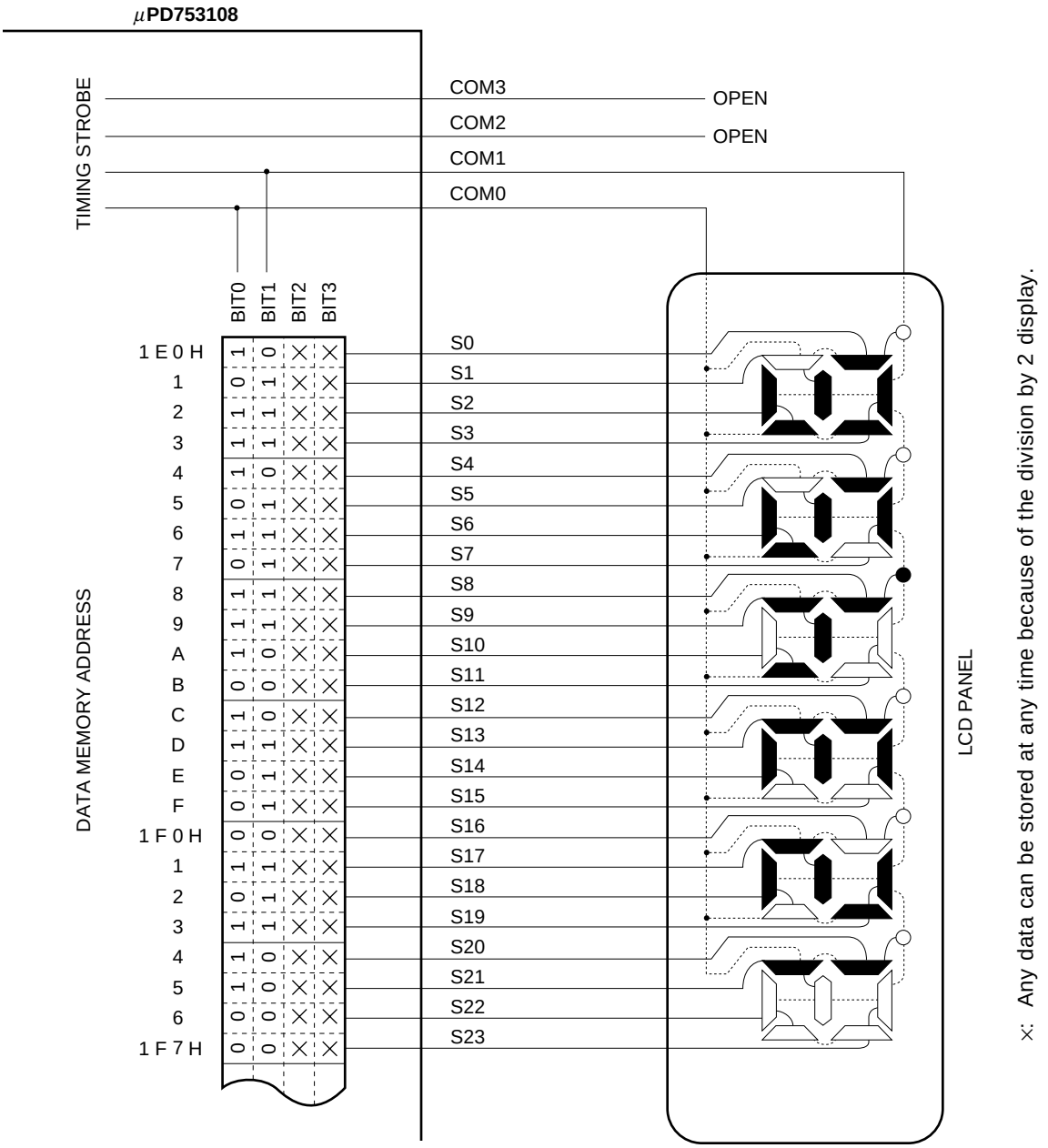
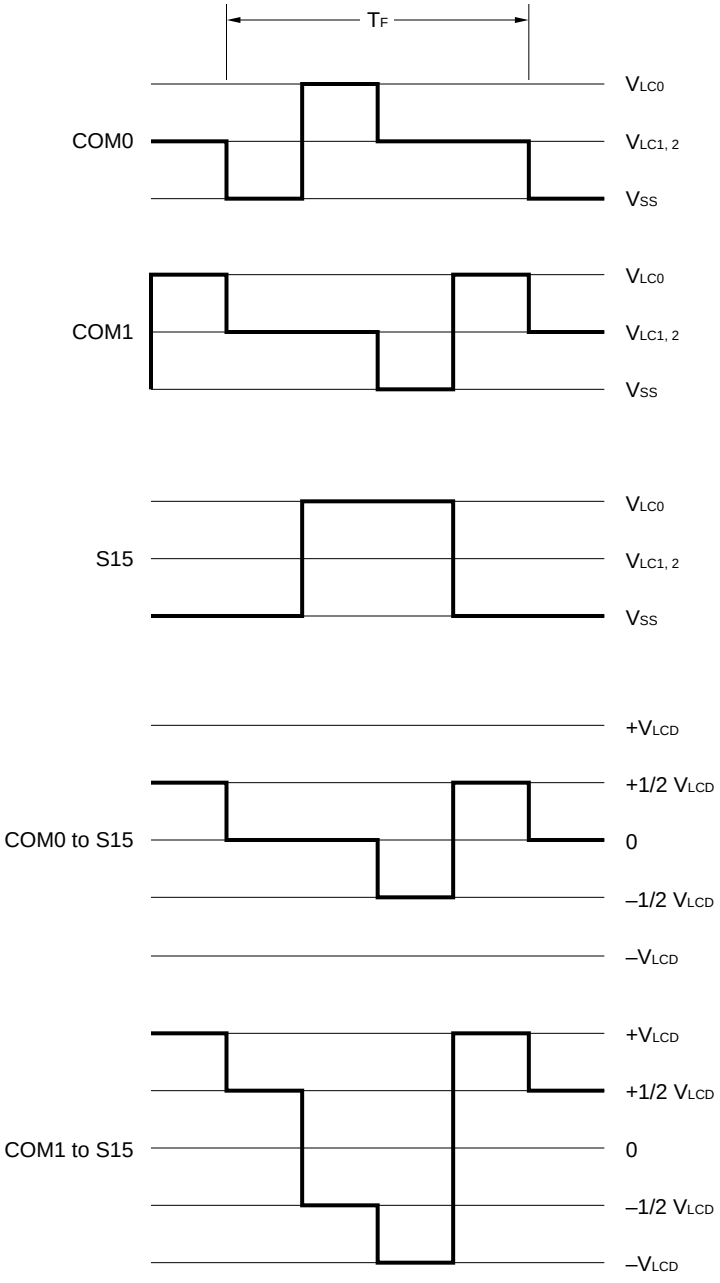


Figure 5-118. Division by 2 LCD Drive Waveform Example (1/2 Bias method)



(3) Division by 3 display example

Figure 5-120 shows connection of a division by 3 mode 8-digit LCD panel having the display pattern shown in Figure 5-119, the μPD753108 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 5-21 to the S6 to S8 pins at the COM0 to COM2 common signal timings according to the display pattern shown in Figure 5-119.

Table 5-21. 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 bits at display data memory address 1E6H corresponding to S6 need to be set to ×110. Figure 5-121 (1/2 bias method) and 5-122 (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 5-119. Division by 3 Mode LCD Display Pattern and Electrode Connection

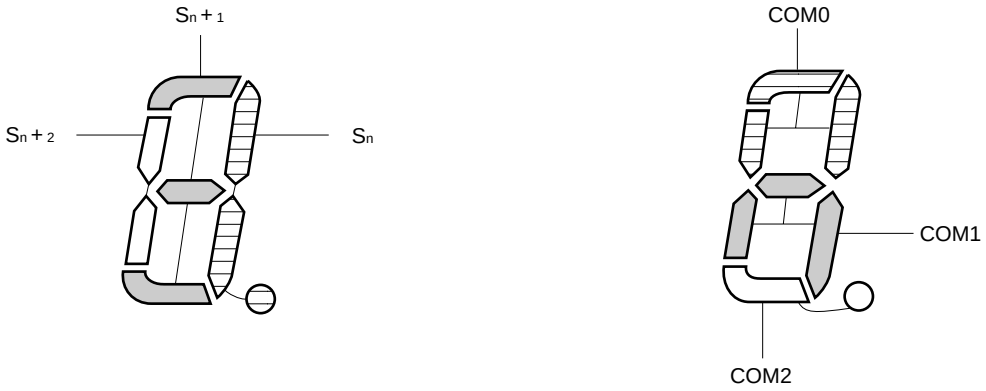


Figure 5-120. Division by 3 LCD Panel Connection Example

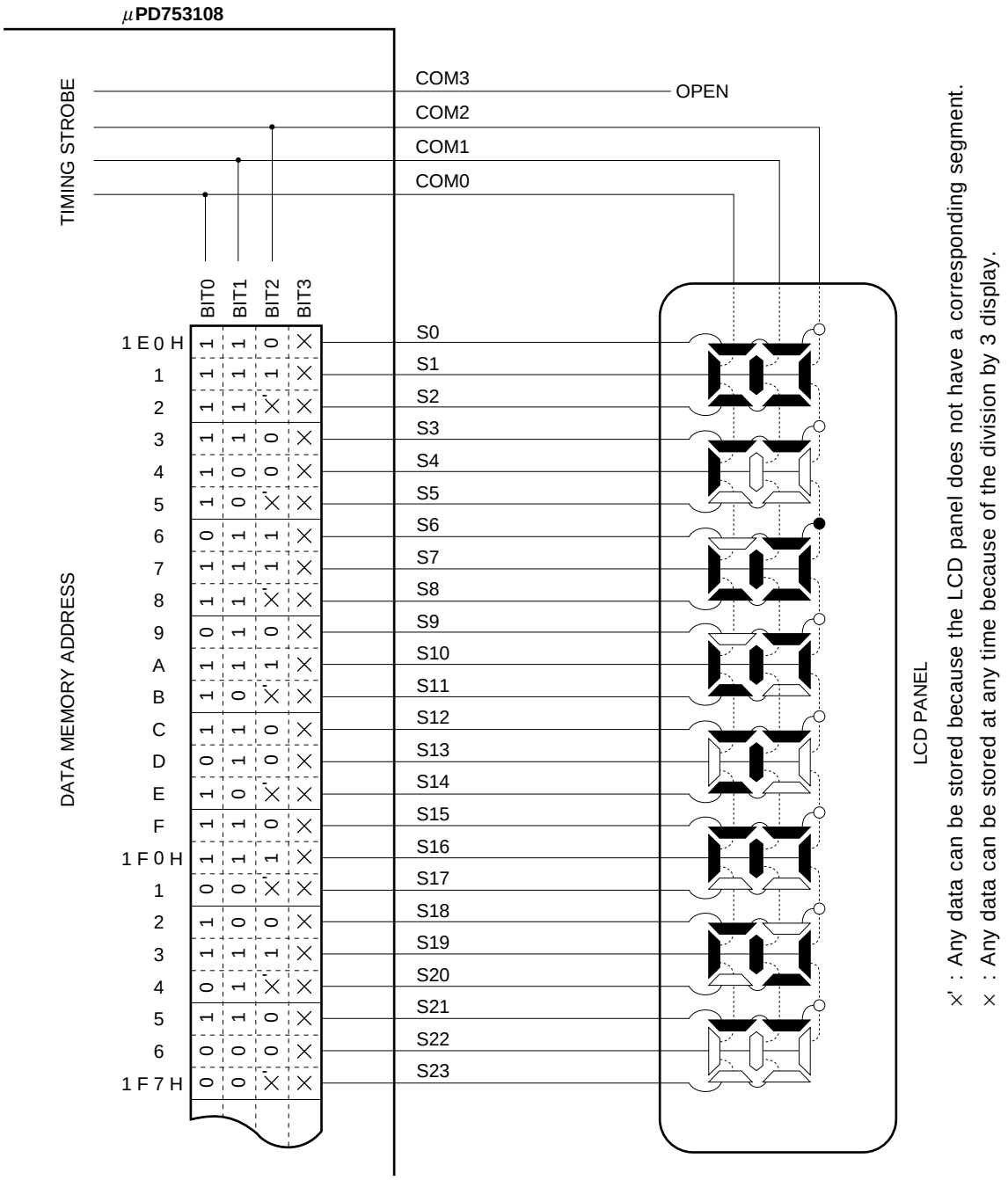


Figure 5-121. Division by 3 LCD Drive Waveform Example (1/2 Bias method)

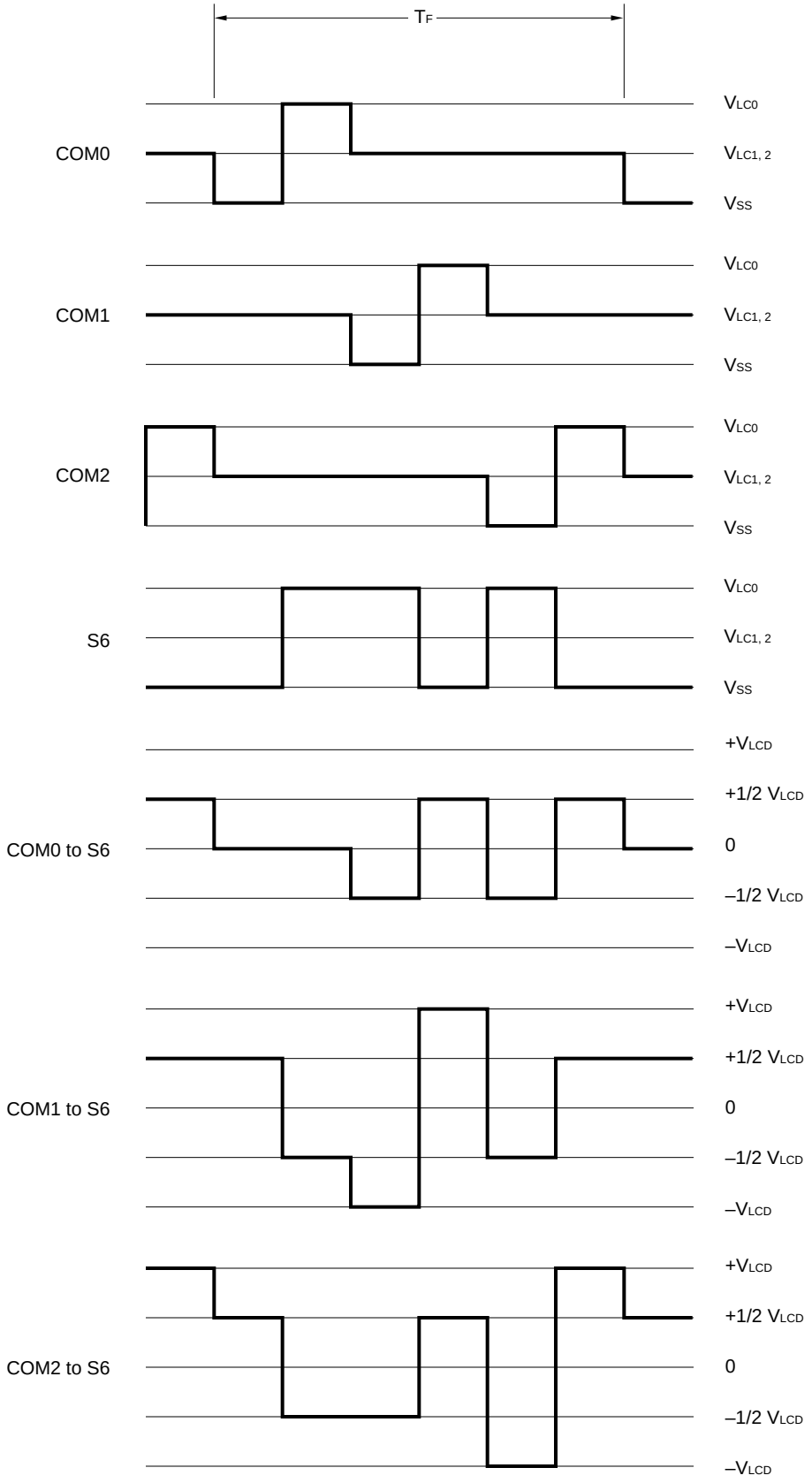
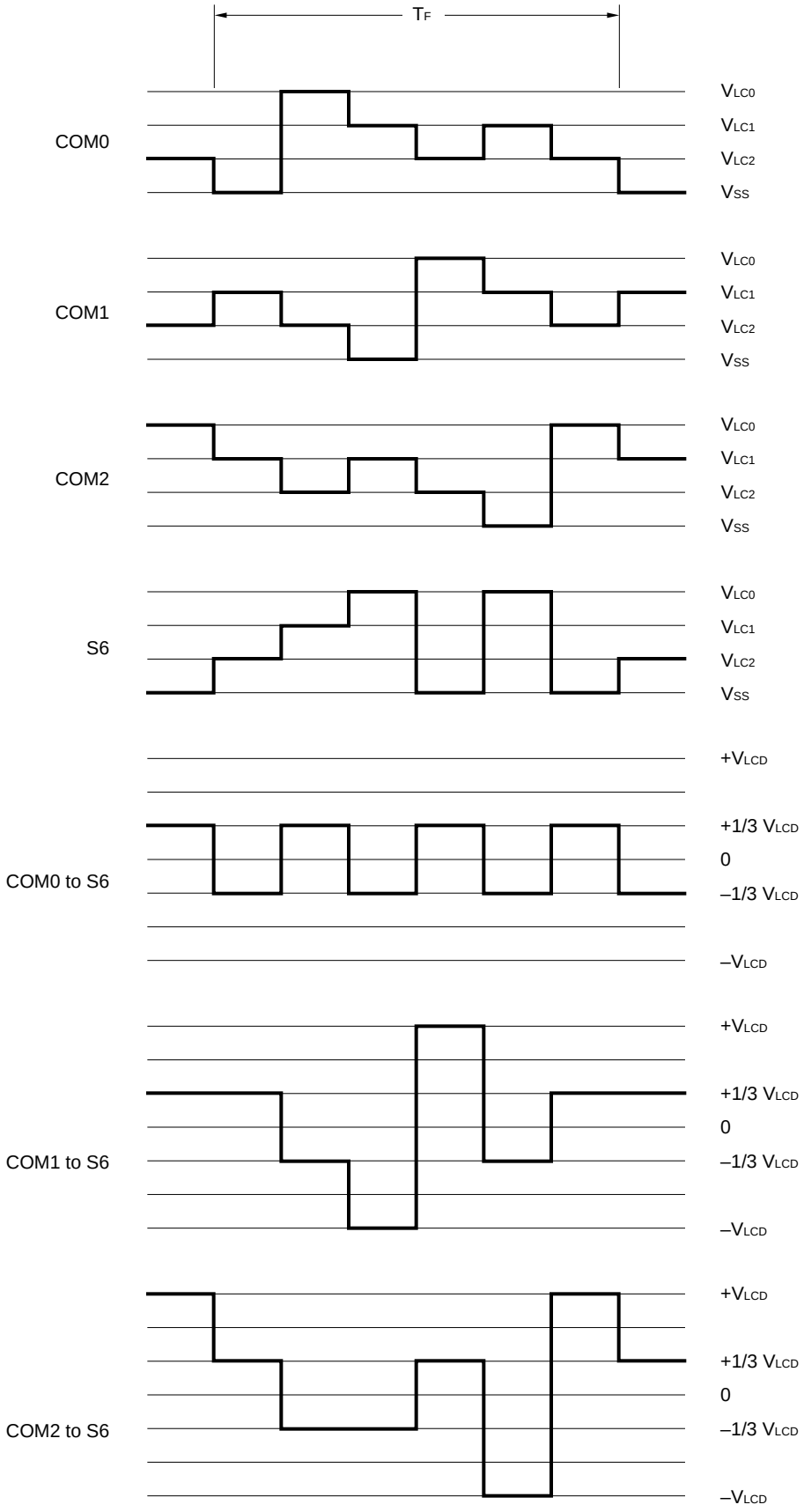


Figure 5-122. Division by 3 LCD Drive Waveform Example (1/3 Bias method)



(4) Division by 4 display example

Figure 5-124 shows connection of the division by 4 mode 12-digit LCD panel having the display pattern shown in Figure 5-123, the μPD753108 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 5-22 to the S12 and S13 pins at the COM0 to COM3 common signal timing according to the display pattern shown in Figure 5-123.

Table 5-22. 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 bits at display data memory address 1ECH corresponding to S12 need to be set to 1101. Figure 5-125 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 the LCD on level being generated when S12 becomes the selection voltage at the COM0 selection timing.

Figure 5-123. Division by 4 Mode LCD Display Pattern and Electrode Connection

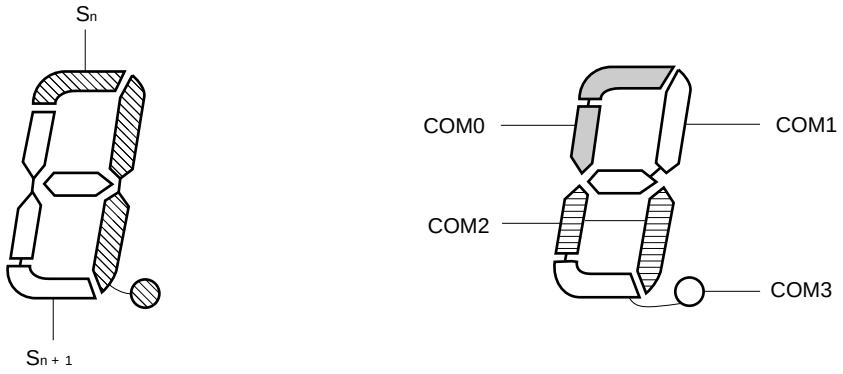


Figure 5-124. Division by 4 LCD Panel Connection Example

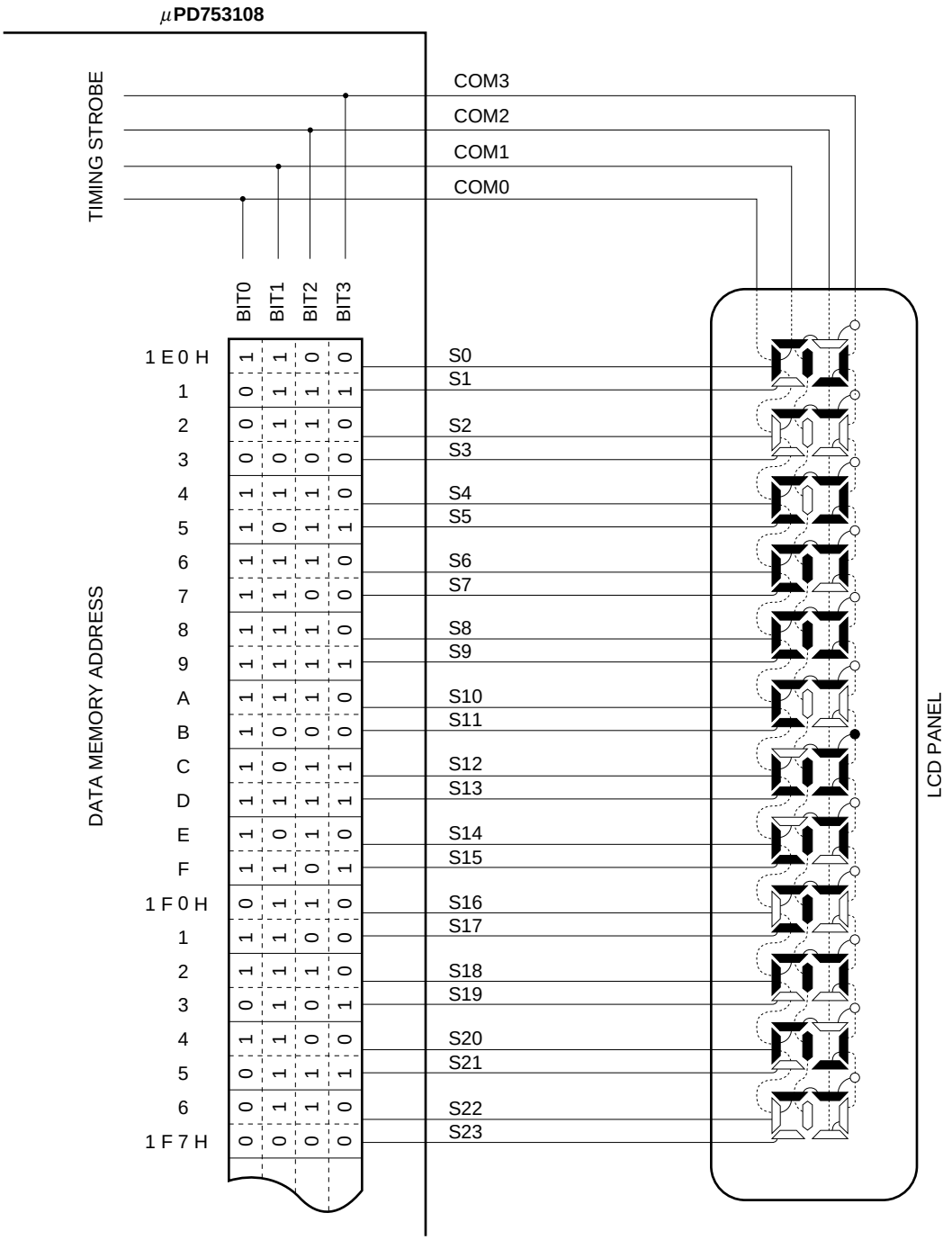
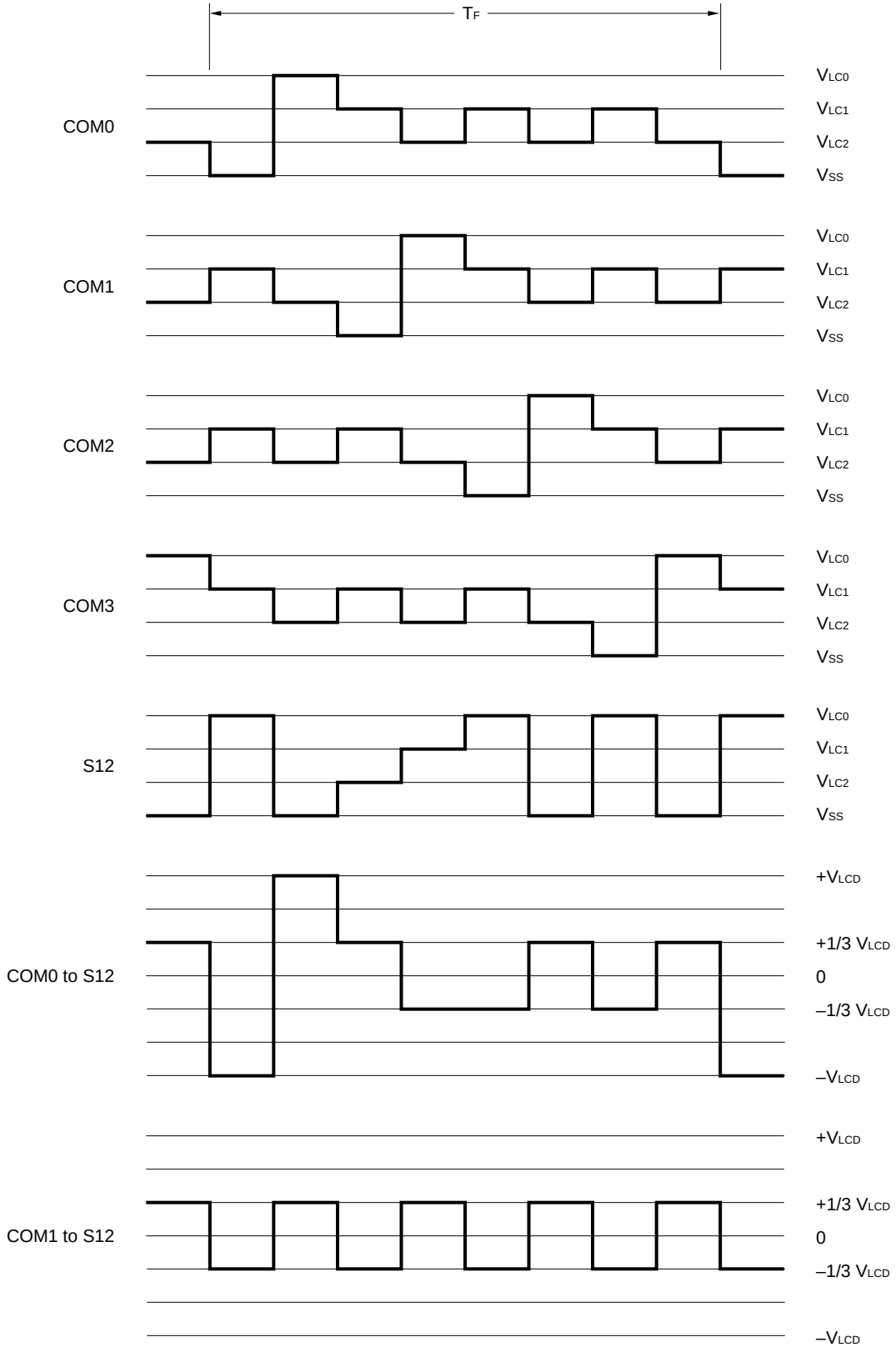


Figure 5-125. Division by 4 LCD Drive Waveform Example (1/3 Bias method)

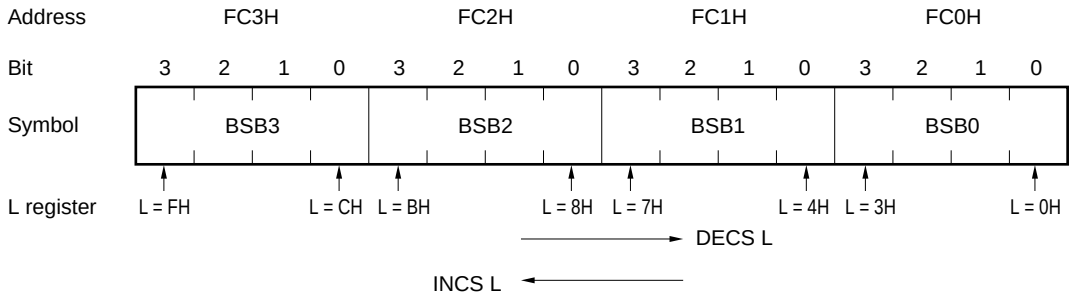


5.8 Bit Sequential Buffer ··· 16 Bits

The bit sequential buffer (BSB) is a special data memory for bit manipulation and the bit manipulation can be easily performed by changing the address specification and bit specification in sequence, therefore it is useful when processing a long data bit-wise.

The data memory is composed of 16 bits and the pmem.@L addressing of a bit manipulation instruction is possible. The bit can be specified indirectly by the L register. In this case, processing can be done by moving the specified bit in sequence by incrementing and decrementing the L register in the program loop.

Figure 5-126. Bit Sequential Buffer Format



Remarks 1. In the pmem.@L addressing, the specified bit moves corresponding to the L register.

2. In the pmem.@L addressing, the BSB can be manipulated regardless of MBE/MBS specification.

Data can be operated by direct addressing. It can be used for continuous input and output of 1-bit data together with the 1-bit/4-bit/8-bit direct addressing and pmem.@L addressing. For 8-bit manipulation, the BSB0 and BSB2 must be specified to manipulate the high-order 8-bits and low-order 8 bits.

Example The 16-bit data of BUFF 1, 2 is output serially starting with the bit 0 of port 3.

```
        CLR1    MBE
        MOV     XA, BUFF1
        MOV     BSB0, XA    ; Sets BSB0, 1.
        MOV     XA, BUFF2
        MOV     BSB2, XA    ; Sets BSB2, 3.
        MOV     L, #0
LOOP0 : SKT     BSB0, @L    ; Tests the specification bit of BSB.
        BR      LOOP1
        NOP                 ; Dummy (timing adjustment)
        SET1    PORT3.0     ; Sets the bit 0 of port 3.
        BR      LOOP2
LOOP1 : CLR1    PORT3.0     ; Clears the bit 0 of port 3.
        NOP                 ; Dummy (timing adjustment)
        NOP
LOOP2 : INCS    L           ; L ← L+1
        BR      LOOP0
        RET
```

[MEMO]

CHAPTER 6 INTERRUPT FUNCTION AND TEST FUNCTION

The μPD753108 has eight vectored interrupt sources and two test inputs that can be used for various applications.

The interrupt control circuit of the μPD753108 has unique features and can process interrupts at extremely high speed.

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.

(1) Interrupt function

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

(b) Can set any interrupt start address.

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

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

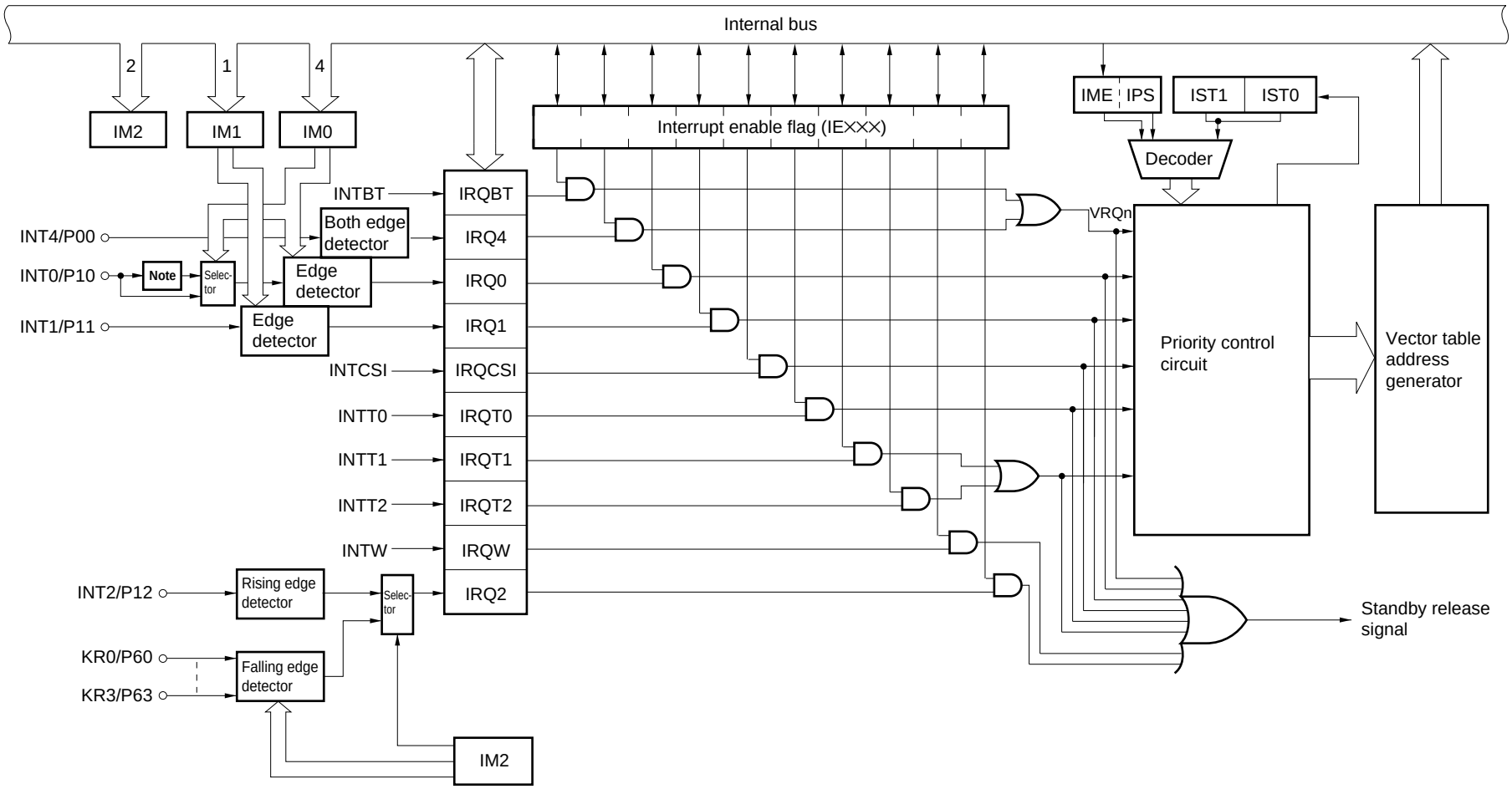
(e) Release the standby mode. An interrupt to be released can be selected by the interrupt enable flag.

(2) Test function

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

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

Figure 6-1. Interrupt Control Circuit Block Diagram



Note Noise eliminator (standby release is disabled when noise eliminator is selected)

6.2 Types of Interrupt Sources and Vector Tables

The μPD753108 has the following eight types of interrupt sources, and multiple interrupts by software control are allowed.

Table 6-1. Types of Interrupt Sources

Interrupt Source		Internal/ External	Interrupt Priority[Note]	Vectored Interrupt Request Signal (Vector Table Address)
INTBT	(Reference interval signal sent from the basic interval timer/watchdog timer)	Internal	1	VRQ1 (0002H)
INT4	(Detection by both rising edge and falling edge is valid.)	External		
INT0	(Selects rising edge or falling edge.)	External	2	VRQ2 (0004H)
INT1		External	3	VRQ3 (0006H)
INTCSI	(Serial data transfer end signal)	Internal	4	VRQ4 (0008H)
INTT0	(Match signal between the count register and modulo register of the timer/event counter 0)	Internal	5	VRQ5 (000AH)
INTT1	(Match signal between the count register and modulo register of the timer/event counter 1)	Internal	6	VRQ6 (000CH)
INTT2	(Match signal between the count register and modulo register of the timer/event counter 2)	Internal		

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/INT4 start address (high-order 6 bits)
			INTBT/INT4 start address (low-order 8 bits)
0004H	MBE	RBE	INT0 start address (high-order 6 bits)
			INT0 start address (low-order 8 bits)
0006H	MBE	RBE	INT1 start address (high-order 6 bits)
			INT1 start address (low-order 8 bits)
0008H	MBE	RBE	INTCSI start address (high-order 6 bits)
			INTCSI start address (low-order 8 bits)
000AH	MBE	RBE	INTT0 start address (high-order 6 bits)
			INTT0 start address (low-order 8 bits)
000CH	MBE	RBE	INTT1, INTT2 start address (high-order 6 bits)
			INTT1, INTT2 start address (low-order 8 bits)

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

To the vector table, write the start address of interrupt processing, and the set values of MBE and RBE during interrupt processing. The vector table is set by using an assembler pseudo-instruction (VENTn: n = 1 to 6).

Example Setting of vector table of INTBT/INT4

```
VENT1  MBE = 0, RBE = 0, GOTOBT
  ↑      ↑      ↑      ↑
 <1>    <2>    <3>    <4>
```

<1> Vector table of address 0002
<2> Setting of MBE in interrupt processing routine
<3> Setting of RBE in interrupt processing routine
<4> Symbol indicating start address of interrupt processing routine

★ **Caution The contents that are written into the operand of VENTn (n = 1 to 6) instruction (MBE, RBE, start address) are stored into the vector table address of 2n.**

Example Setting of vector tables of INTBT/INT4 and INTT0

```
VENT1   MBE = 0, RBE = 0, GOTOBT ; INTBT/INT4 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 μPD753108 has the following eight interrupt request flags (IRQ×××) corresponding to the respective interrupt sources:

INT0 interrupt request flag (IRQ0)	Serial interface interrupt request flag (IRQCSI)
INT1 interrupt request flag (IRQ1)	Timer/event counter 0 interrupt request flag (IRQT0)
INT4 interrupt request flag (IRQ4)	Timer/event counter 1 interrupt request flag (IRQT1)
BT interrupt request flag (IRQBT)	Timer/event counter 2 interrupt request flag (IRQT2)

Each interrupt request flag is set to "1" when the corresponding interrupt request is generated, and is automatically cleared to "0" when the interrupt processing is executed. However, because IRQBT and IRQ4 (also IRQT1 and IRQT2) share the vector address, these flags are cleared differently from the other flags (refer to **6.6 Vector Address Share Interrupt Service**).

The μPD753108 also has eight interrupt enable flags (IE×××) corresponding to the respective interrupt request flags.

INT0 interrupt enable flag (IE0)	Serial interface interrupt enable flag (IECSI)
INT1 interrupt enable flag (IE1)	Timer/event counter 0 interrupt enable flag (IET0)
INT4 interrupt enable flag (IE4)	Timer/event counter 1 interrupt enable flag (IET1)
BT interrupt enable flag (IEBT)	Timer/event counter 2 interrupt enable flag (IET2)

When the interrupt enable flag is "1", the interrupt is enabled; and when it is "0", the interrupt is disabled.

When the interrupt request flag is set and interrupt enable flag enables the interrupt, a vectored interrupt request (VRQn) is generated. It is also used to release the standby mode.

The interrupt request flag and interrupt enable flag are operated by a bit manipulation instruction and 4-bit memory manipulation instruction. When the bit 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 IE××× instruction and DI IE××× instruction. A SKTCLR instruction is normally used to test the interrupt request flag.

```
Example  EI       IE0     ; Enables INT0.
         DI       IE1     ; Disables INT1.
         SKTCLR   IRQCSI  ; Skips and clears when IRQCSI is 1.
```

When the interrupt request flag is set by an instruction, a vectored interrupt is executed as if an interrupt were generated.

The interrupt request flag and interrupt enable flag are cleared to "0" when a $\overline{\text{RESET}}$ signal is generated and all the interrupts are 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
IRQ4	Set by the detection of both rising edge and falling edge of an INT4/P00 pin.	IE4
IRQ0	Reset by the edge detection of an INT0/P10 pin input signal. The detection edge is selected by the INT0 edge detection mode register (IM0).	IE0
IRQ1	Reset by the edge detection of an INT1/P11 pin input signal. The detection edge is selected by the INT1 edge detection mode register (IM1).	IE1
IRQCSI	Set by a serial data transfer end signal of the serial interface.	IECSI
IRQT0	Set by a match signal sent from the timer/event counter 0.	IET0
IRQT1	Set by a match signal sent from the timer/event counter 1.	IET1
IRQT2	Set by a match signal sent from the timer/event counter 2.	IET2

(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 lower 3 bits of IPS, the interrupt must be disabled (IME = 0).

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

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

Figure 6-3. Interrupt Priority Selection Register

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

Selection of higher-order-priority interrupts

0	0	0	No interrupts are handled as higher-order-priority interrupts.	
0	0	1	VRQ1 (INTBT/INT4)	The above vectored interrupts are regarded as higher-order priority interrupts.
0	1	0	VRQ2 (INT0)	
0	1	1	VRQ3 (INT1)	
1	0	0	VRQ4 (INTCSI)	
1	0	1	VRQ5 (INTT0)	
1	1	0	VRQ6 (INTT1/INTT2)	
1	1	1	Setting prohibited	

Interrupt master enable flag (IME)

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

(3) Hardware of INT0, INT1, and INT4

(a) Figure 6-4 (a) shows the configuration of the INT0. An external interrupt is input to select the detection edge, that is, rising edge or falling edge.

The INT0 has a noise eliminator by means of the sampling clock (See **Figure 6-5 Noise Eliminator Input/ Output Timing**). The noise eliminator removes the pulses which are narrower than the two cycles of the sampling clock[Note]. However, a pulse which is wider than the one cycle of the sampling clock may be accepted in some cases as an interrupting signal depending on a sampling timing. The pulses which are wider than the two cycles of sampling clock are accepted all the time as interrupting signals (See **Figure 6-5 <2> (a)**).

The INT0 has the two sampling clocks: Φ and f_X/64, either of which can be selected. One of them is selected by the bit 3 (IM03) of the INT0 edge detection mode register (IM0) (See **Figure 6-6 (a)**).

The detection edge is selected by the bits 0/1 (IM00/IM01) of the INT0 edge detection mode register (IM0).

Figure 6-6 (a) shows the format of the IM0. It is set by a 4-bit manipulation instruction. All the bits are cleared to "0" when the $\overline{\text{RESET}}$ signal is generated and the rising edge specification is adopted.

Note $2t_{CY}$ when the sampling clock is Φ.
$128/f_X$ when the sampling clock is f_X/64.

Cautions 1. Pulses are input to the INT0/P10 pin via a noise eliminator if it is used as a port, therefore pulses longer than the two cycles of sampling clock must be input.

2. When the noise eliminator is selected, that is IM02 is set to 0, the INT0 performs sampling by a clock, therefore it does not operate in the standby mode (the noise eliminator does not operate when the CPU clock Φ is not supplied). Consequently, if the standby mode is to be released by the INT0, the noise eliminator must not be selected, that is the IM02 must be set to 1.

(b) Figure 6-4 (b) shows the configuration of the INT1. An external interrupt is input to select the detection edge, rising edge or falling edge.

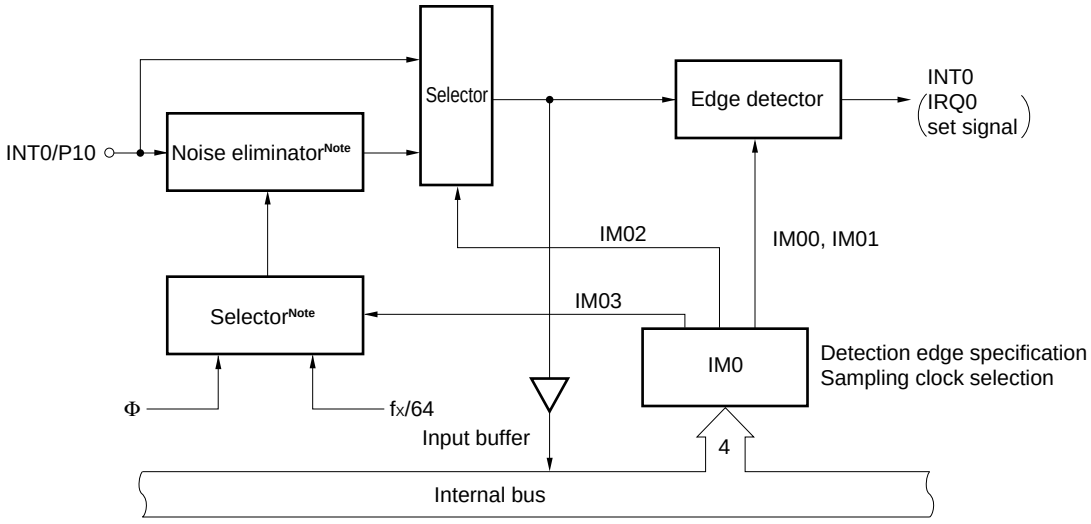
The detection edge is selected by the INT1 edge detection mode register (IM1).

Figure 6-6 (b) shows the format of the IM1. It is set by a bit manipulation instruction. All the bits are cleared to "0" when the $\overline{\text{RESET}}$ signal is generated and the rising edge specification is adopted.

(c) Figure 6-4 (c) shows the configuration of the INT4. An external interrupt is input so that both the rising edge and falling edge can be detected.

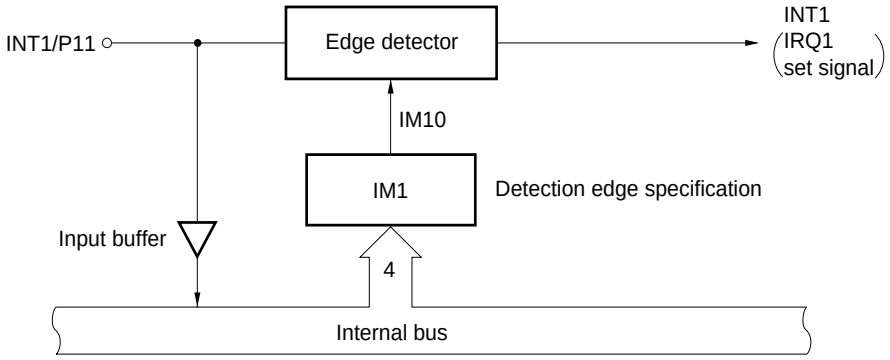
Figure 6-4. Configurations of INT0, INT1, and INT4

(a) INT0 hardware



★ **Note** Even though fx/64 is selected, the HALT mode cannot be released by INT0.

(b) INT1 hardware



(c) INT4 hardware

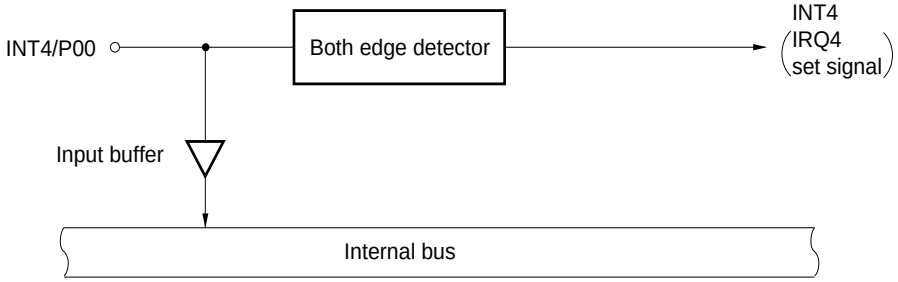
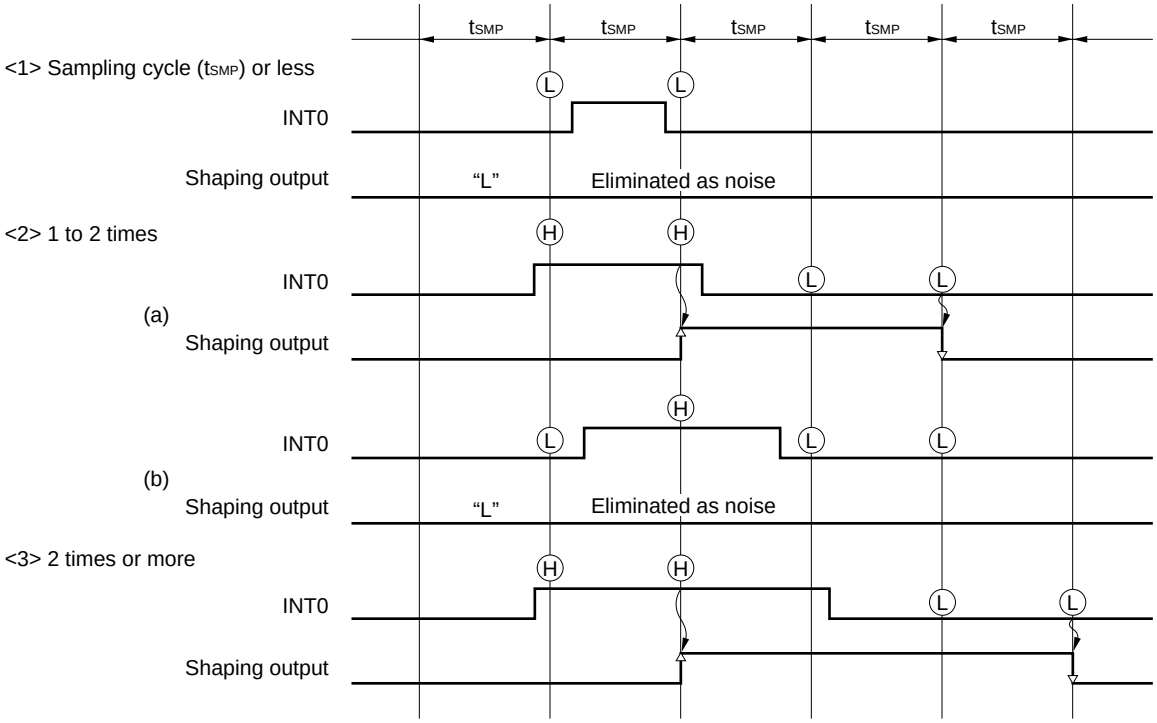


Figure 6-5. Noise Eliminator Input/Output Timing



Remark $t_{SMP} = t_{CY}$ or $64/f_X$

Figure 6-6. Edge Detection Mode Register Format

(a) INT0 edge detection mode register (IM0)

Address	3	2	1	0	Symbol
FB4H	IM03	IM02	IM01	IM00	IM0

IM01	IM00	Detection Edge Specification
0	0	Rising edge specification
0	1	Falling edge specification
1	0	Rising edge/falling edge specification
1	1	Ignored (The interrupt request flag is not set).

IM02	Noise Eliminator Select Bit	Sampling	Standby Release
0	Selects the noise eliminator.	Enabled	Impossible
1	Does not select the noise eliminator.	Disabled	Possible

IM03	Sampling Clock
0	Φ (0.67 μs, 1.33 μs, 2.67 μs, 10.7 μs: 6.00-MHz operation)
1	f_X/64 (10.7 μs : 6.00-MHz operation)

(b) INT1 edge detection mode register (IM1)

Address	3	2	1	0	Symbol
FB5H	0	0	0	IM10	IM1

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

Caution When the edge detection mode register is changed, the interrupt request flag may be set in some case, therefore the interrupts must be disabled beforehand to select the mode register and the interrupt request flag must be cleared by a CLR1 instruction, and then the interrupts must be enabled. When f_X/64 is selected as the sampling clock by changing the IM0, the interrupt request flag must be cleared when 16 machine cycles have elapsed since the mode register was changed.

(4) Interrupt status flag

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

The interrupt priority control circuit controls 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 asserted.

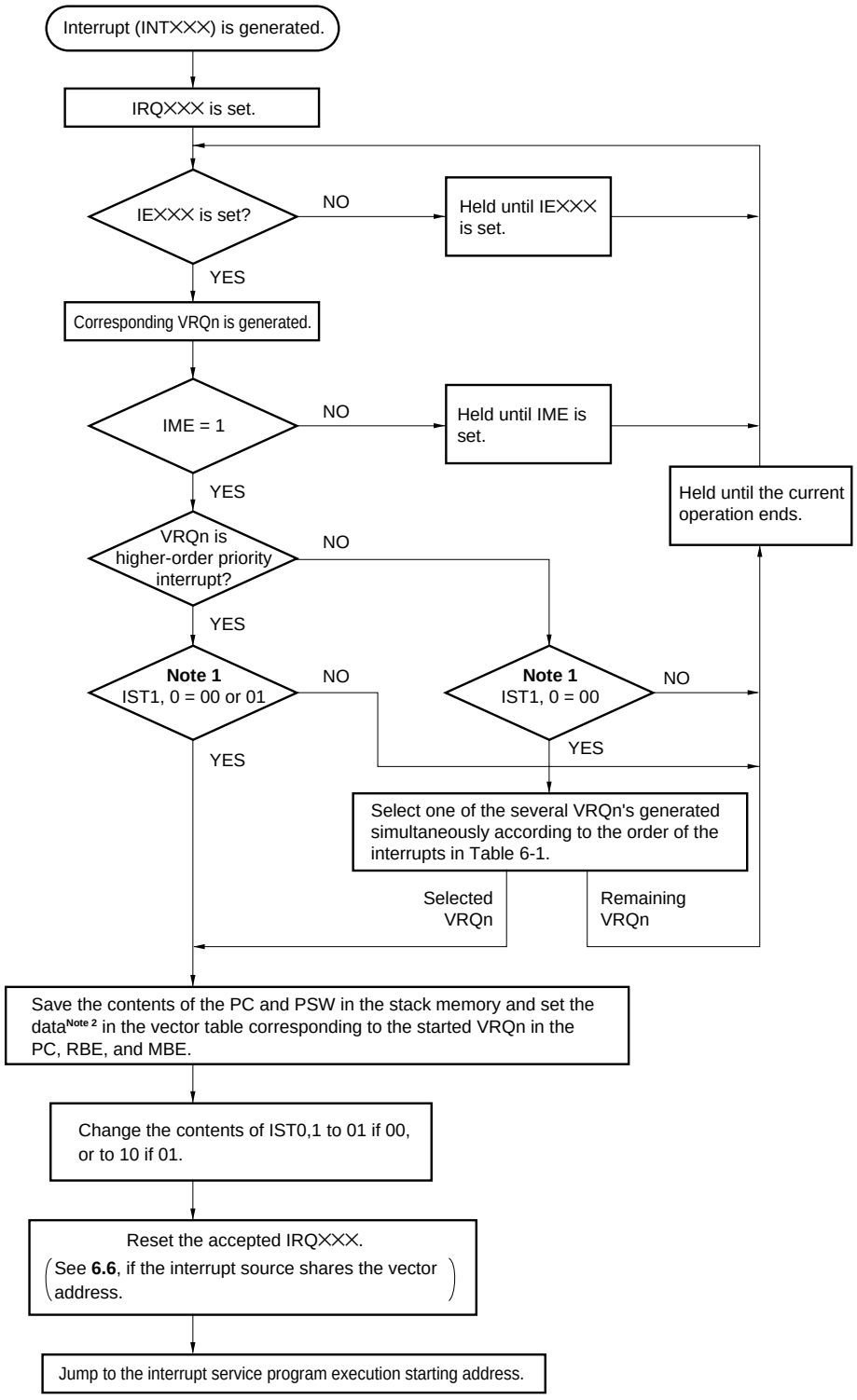
Table 6-3. IST1 and IST0 and Interrupt Processing Status

IST1	IST0	Status of Processing under Execution	Processing by CPU	Interrupt Request That Can Be Acknowledged	After Interrupt Acknowledged	
					IST1	IST0
0	0	Status 0	Executes normal program	All interrupts can be acknowledged	0	1
0	1	Status 1	Processes interrupt with low or high priority	Interrupt with high priority can be acknowledged	1	0
1	0	Status 2	Processes interrupt with high priority	Acknowledging all interrupts is disabled	—	—
1	1	Setting prohibited				

6.4 Interrupt Sequence

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

Figure 6-7. Interrupt Processing Sequence



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

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

6.5 Multiple Interrupt Service Control

The μPD753108 accepts multiple interrupts in the following two methods.

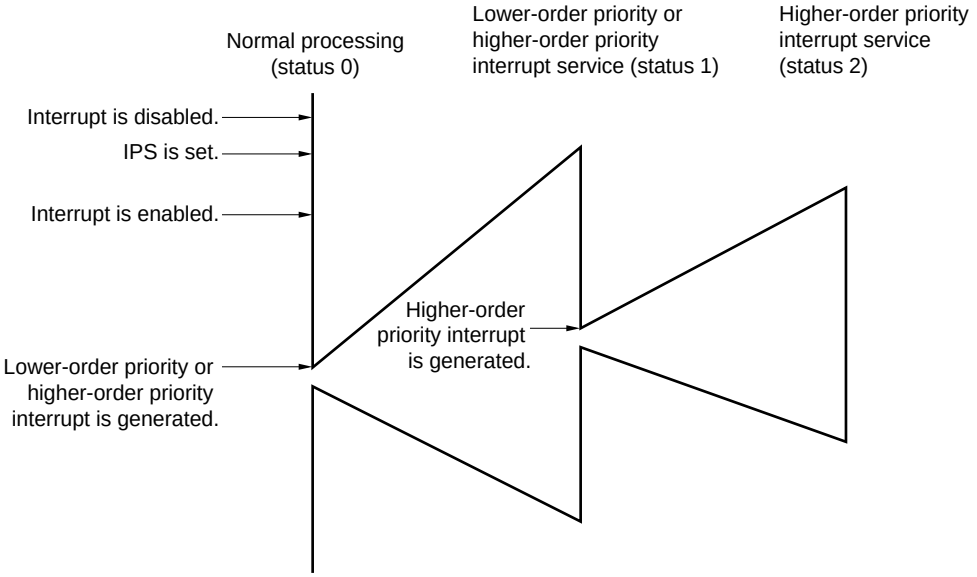
(1) Multiple interrupts wherein higher-order priority interrupts are specified

This is a standard multiple interrupts method of the μPD753108, 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 6-8** and **Table 6-3**).

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

Figure 6-8. Multiple Interrupts by Higher-Order Priority Interrupts



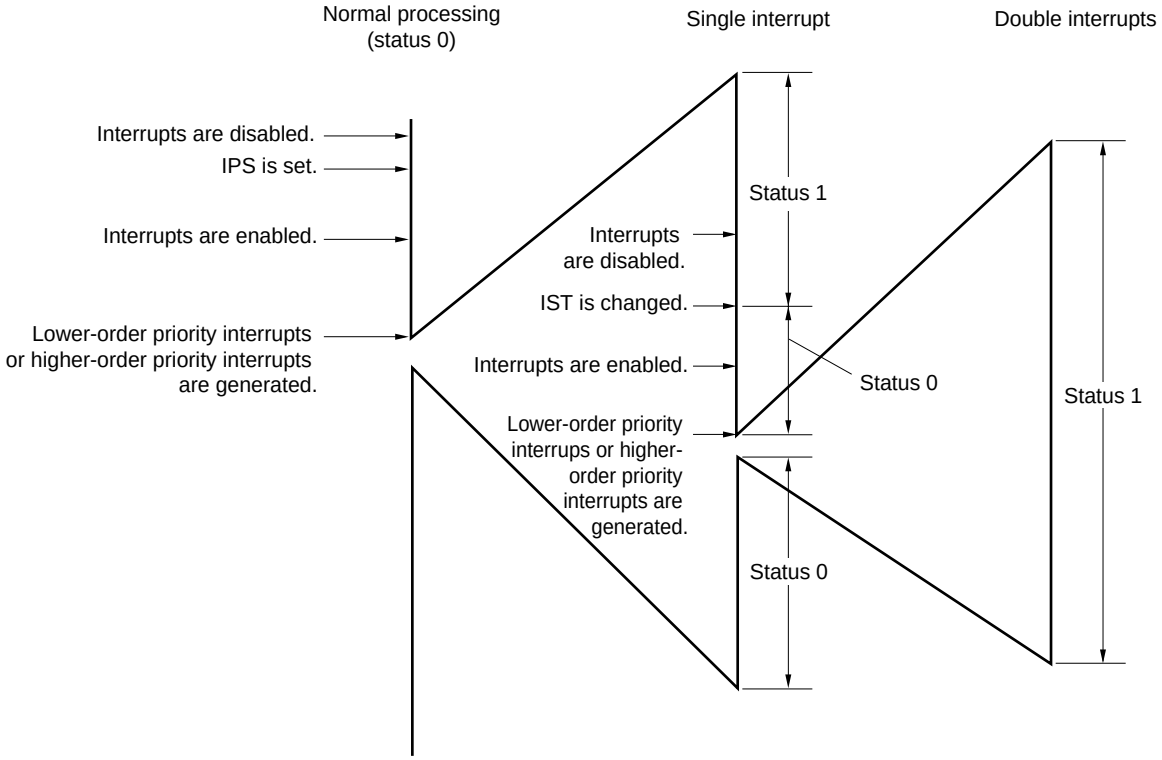
(2) Multiple interrupts changing the interrupt status flag

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

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

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

Figure 6-9. Multiple Interrupts by Changing Interrupt Status Flag



6.6 Vector Address Share Interrupt Service

Because INTBT, INT4 and INTT1, INTT2 interrupt sources share the vector table addresses, interrupt source selection is made as described below:

(1) To use one interrupt only

Set the interrupt enable flag to 1 for the required one of the two interrupt sources sharing the vector table addresses, and clear the interrupt enable flag of the other interrupt source. In this case, an interrupt request is generated from the interrupt source corresponding to the interrupt enable flag that is set to 1 (IE×××= 1). When the interrupt request is acknowledged, the interrupt request flag is cleared.

(2) To use both interrupts

Set both the interrupt enable flags of the two interrupt sources to 1. In this case, an interrupt request is made by ORing the interrupt request flags of the two interrupt sources.

Even if an interrupt request is acknowledged when either or both of the interrupt request flags are set to 1, the interrupt request flags are not reset.

Therefore, the interrupt service routine must decide which interrupt source the interrupt is generated from. This is accomplished by executing the SKTCLR instruction at the beginning of interrupt service routine to check the interrupt request flags.

If both the request flags are set when this request flag is tested and cleared, the interrupt request remains even if one of the request flags is cleared. If this interrupt is selected as having the higher priority, nesting processing is started by the remaining interrupt request.

Consequently, the interrupt request not tested is processed first. If the selected interrupt has the lower priority, the remaining interrupt is kept pending and therefore, the interrupt request tested is processed first. Therefore, an interrupt sharing a vector address with another interrupt is identified differently, depending whether it has the higher priority, as shown in Table 6-4.

Table 6-4. Identifying Interrupt Sharing Vector Table Address

With higher priority	Interrupt is disabled and interrupt request flag of interrupt that takes precedence is tested
With lower priority	Interrupt request flag of interrupt source that takes precedence is tested

Examples 1. To use both INTBT and INT4 as having the higher priority and give priority to INT4

```
        DI
        SKTCLR   IRQ4      ;  IRQ4 = 1 ?
        BR       VSUBBT
          :                    Processing routine
        EI                     of INT4
        RETI
          :
VSUBBT: CLR1     IRQBT
          :
          :                    Processing routine
          :                    of INTBT
        EI
        RETI
```

2. To use both INTBT and INT4 as having the lower priority and give priority to INT4

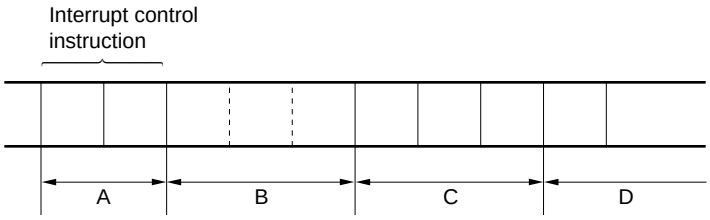
```
        SKTCLR   IRQ4      ;  IRQ4 = 1 ?
        BR       VSUBBT
          :
          :                    Processing routine
          :                    of INT4
        RETI
          :
VSUBBT: CLR1     IRQBT
          :
          :                    Processing routine
          :                    of INTBT
        RETI
```

6.7 Machine Cycles until Interrupt Processing

The number of machine cycles required since an interrupt request flag (IRQ×××) has been set until the interrupt routine is executed is as follows:

(1) If IRQ××× is set while interrupt control instruction is executed

If IRQ××× 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 IRQ×××

B: Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)

C: Interrupt processing (3 machine cycles)

D: Executes interrupt routine

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

2. If the DI instruction is executed when or after IRQ××× is set (A in the above figure), the interrupt request corresponding to IRQ××× that has been set is kept pending until the EI instruction is executed next time.

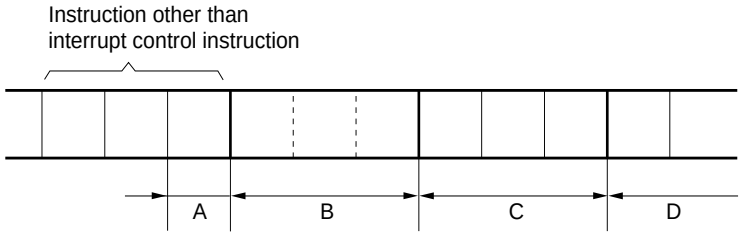
Remarks 1. An interrupt control instruction manipulates the hardware units related to interrupt (address FB×H of the data memory). The DI and EI instructions are interrupt control instructions.

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

(2) If IRQ×××× is set while instruction other than (1) is executed

(a) If IRQ×××× 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 IRQ××××

B: Executes next one instruction (1 to 3 machine cycles; differs depending on instruction)

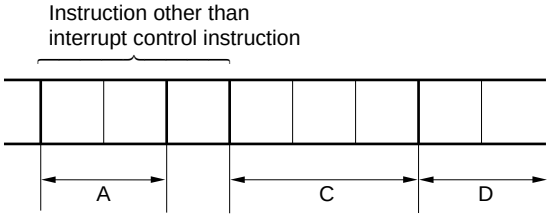
C: Interrupt processing (3 machine cycles)

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 IRQ×××× has been set, the interrupt request corresponding to the set IRQ×××× is kept pending.

(b) If IRQ×××× 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 IRQ××××

C: Interrupt processing (3 machine cycles)

D: Executes interrupt routine

6.8 Effective Usage of Interrupt

Use the interrupt function effectively as follows:

(1) Set MBE to 0 in interrupt processing routine.

If the memory used in the interrupt routine is allocated to addresses 00H through 7FH, and MBE is cleared to 0 by the interrupt vector table, you can program without having to consider the memory bank.

If it is necessary to use memory bank 1, save the memory bank selection register by using the PUSH BS instruction and then select memory bank 1.

(2) Use different register banks for the normal routine and interrupt routine.

The normal routine uses register banks 2 and 3 with RBE = 1 and RBS = 2. If the interrupt routine is for one nested interrupt, use register bank 0 with RBE = 0, so that you do not have to save or restore the registers. When two or more interrupts are nested, set RBE to 1, save the register bank by using the PUSH BS instruction, and set RBS to 1 to select register bank 1.

(3) Use the software interrupt for debugging.

Even if an interrupt request flag is set by an instruction, the same operation as when an interrupt occurs is performed. For debugging of an irregular interrupt or debugging when two or more interrupts occur at the same time, the efficiency can be enhanced by setting the interrupt flag by an instruction.

6.9 Application of Interrupt

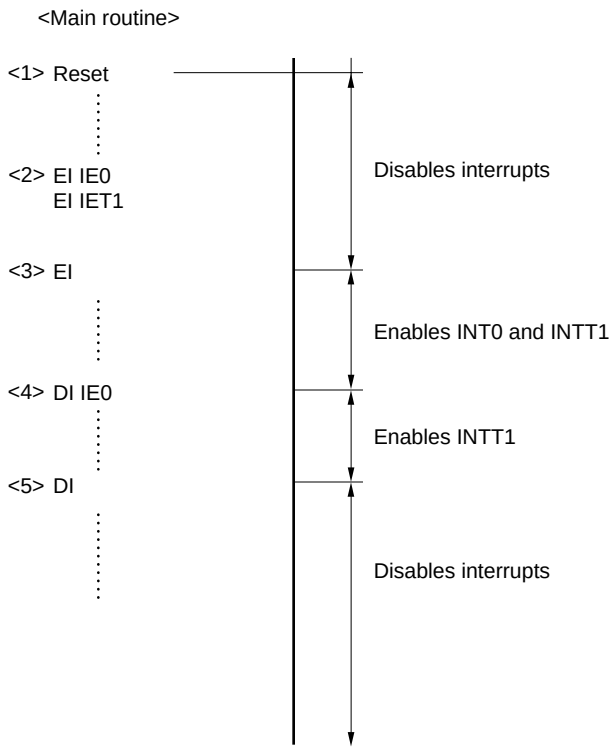
To use the interrupt function, first set as follows by the main routine:

(a) Set the interrupt enable flag of the interrupt used (by using the EI IE××× instruction).
(b) To use INT0 or INT1, select the active edge (set IM0 or IM1).
(c) To use nesting (of an interrupt with the higher priority), set IPS (IME can be set at the same time).
(d) Set the interrupt master enable flag (by using the EI instruction).

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

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

(1) Enabling or disabling interrupt



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

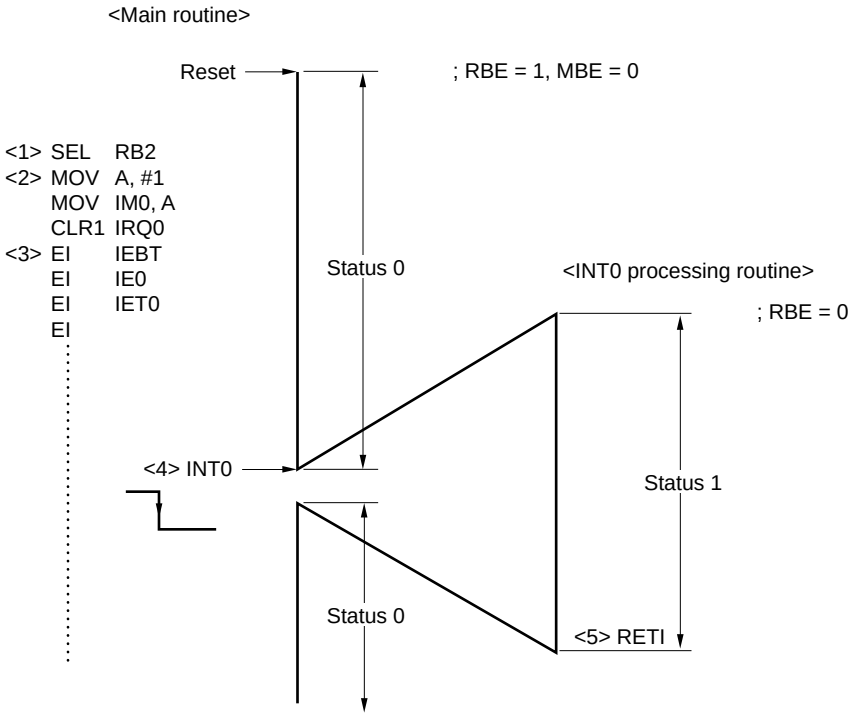
<2> An interrupt enable flag is set by the EI IE×××× instruction.
At this stage, the interrupts are still disabled.

<3> The interrupt master enable flag is set by the EI instruction.
INT0 and INTT1 are enabled at this time.

<4> The interrupt enable flag is cleared by the DI IE×××× instruction, and INT0 is disabled.

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

(2) Example of using INTBT, INT0 (falling edge active), and INTT0. No multiple interrupt (all interrupts have lower priority)



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

<2> INT0 is specified to be active at the falling edge.

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

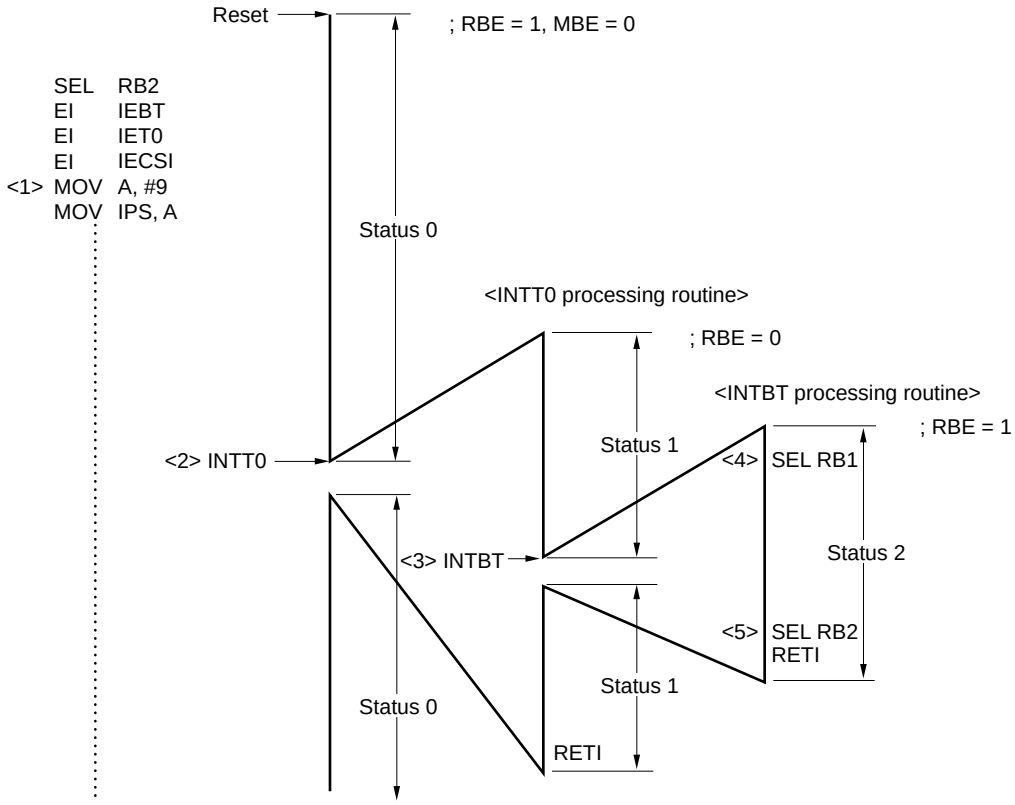
<4> The INT0 interrupt routine is started at the falling edge of INT0. The status is changed to 1, and all the interrupts are disabled.
RBE = 0, and register banks 0 and 1 are used.

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

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

(3) Nesting of interrupts with higher priority

(INTBT has higher priority and INTT0 and INTCSI have lower priority)



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

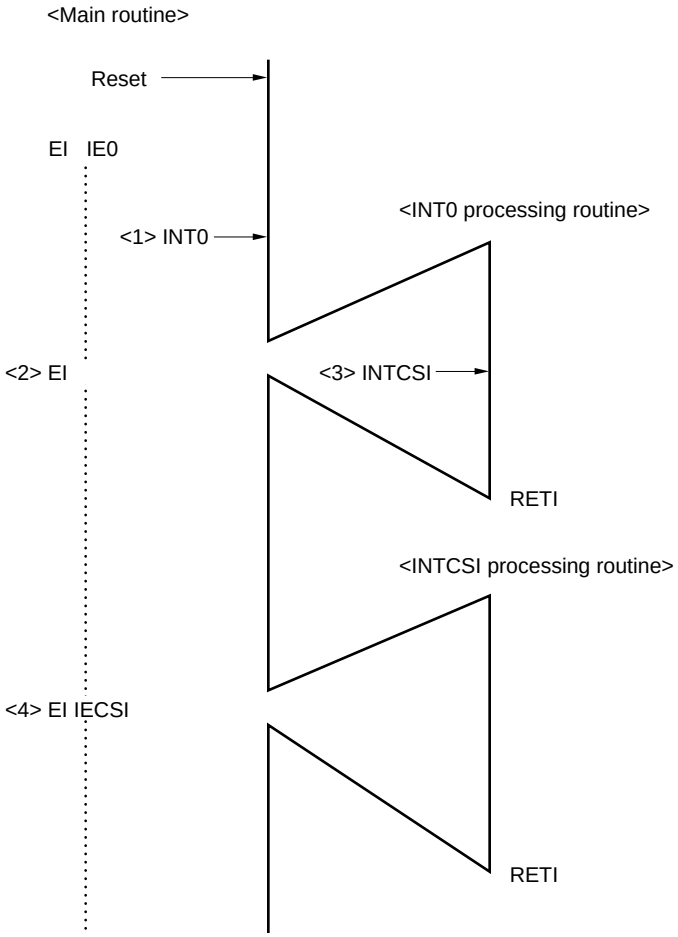
<2> INTT0 processing routine is started when INTT0 with the lower priority occurs. Status 1 is set and the other interrupts with the lower priority are disabled. RBE = 0 to select register bank 0.

<3> INTBT with the higher priority occurs. 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 –



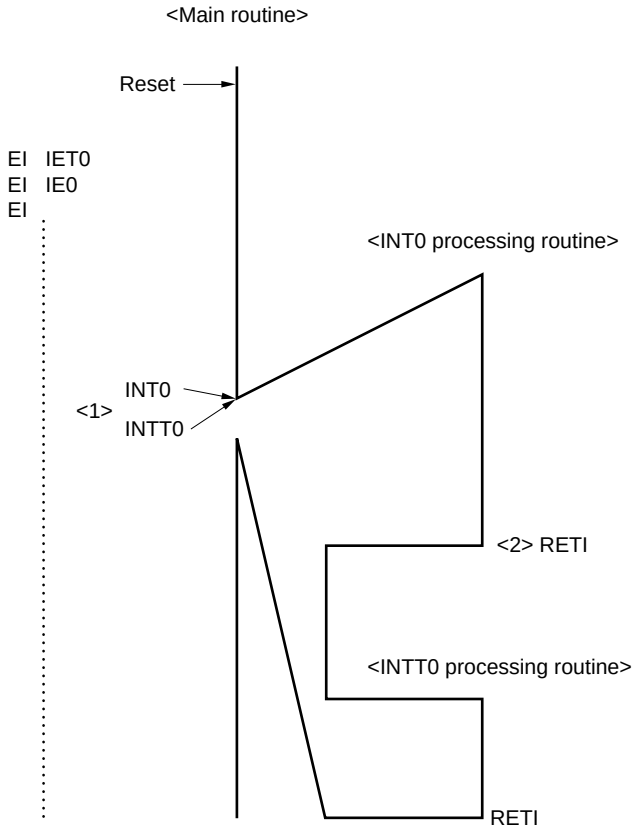
<1> The request flag is kept pending even if INT0 is set while the interrupts are disabled.

<2> INT0 processing routine is started when the interrupts are enabled by the EI instruction.

<3> Same as <1>.

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

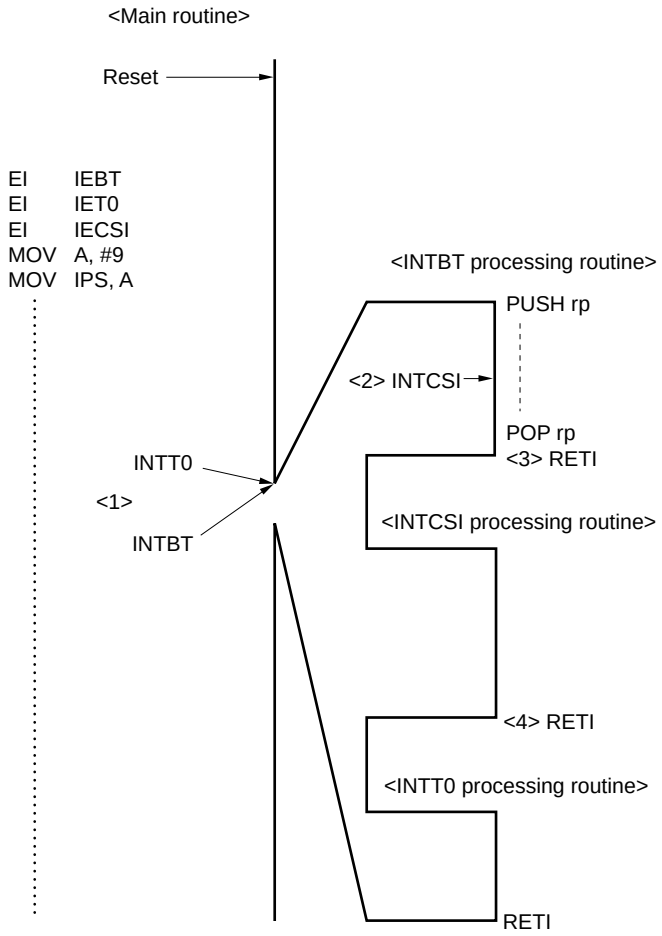
(5) Executing pending interrupt – two interrupts with lower priority occur simultaneously –



<1> If INT0 and INTT0 with the lower priority occur at the same time (while the same instruction is executed), INT0 with the higher priority is executed first (INTT0 is kept pending).

<2> When the INT0 processing routine is terminated by the RETI instruction, the pending INTT0 processing routine is started.

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



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

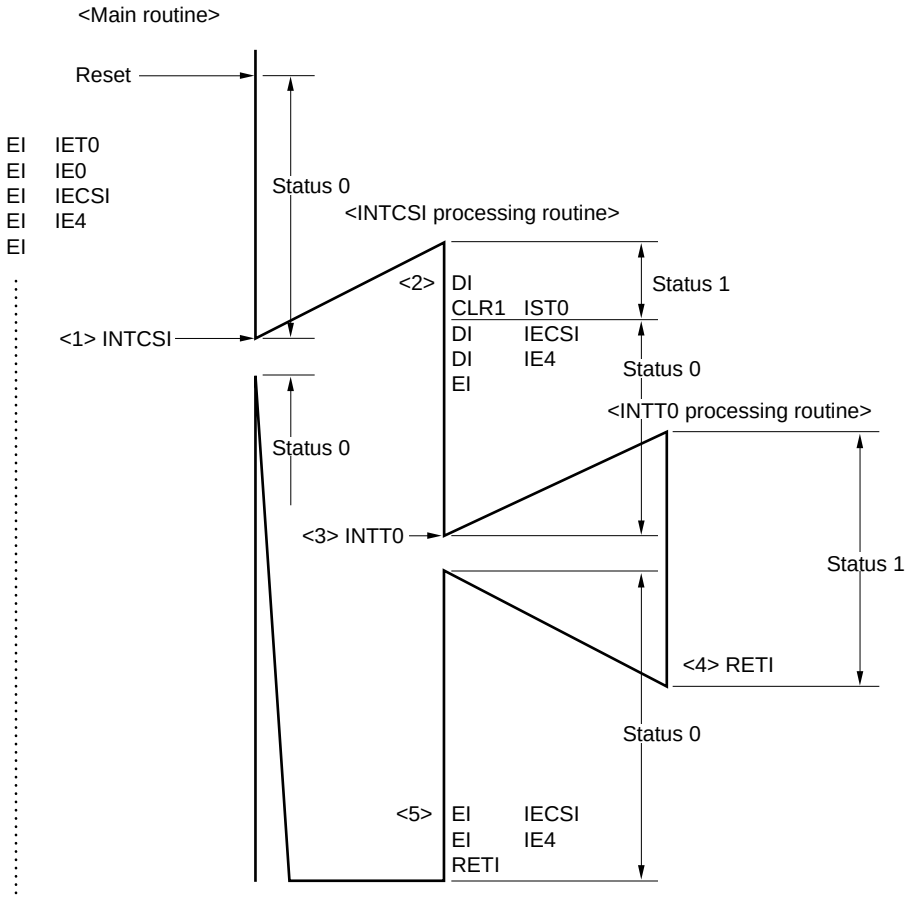
<2> If an interrupt with the lower priority occurs while the interrupt with the higher priority is executed, the interrupt with the lower priority is kept pending.

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

<4> When the processing of INTCSI has been completed, the pending INTT0 is processed.

(7) Enabling two nesting of interrupts

– INTT0 and INT0 are nested doubly and INTCSI and INT4 are nested singly –



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

<2> The status is changed to 0 by clearing IST0. INTCSI and INT4 that do not enable nesting are 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 completed.

<5> The disabled INTCSI and INT4 are enabled, and execution returns to the main routine.

6.10 Test Function

6.10.1 Types of test sources

The μPD753108 has two types of test sources. Of these, INT2 is provided with two types of edge-detection testable inputs.

Table 6-5. Types of Test Sources

Test Source	Internal/External
INT2 (detects rising edge input to INT2 or falling edge input to KR0 to KR3)	External
INTW (signal from watch timer)	Internal

6.10.2 Hardware devices controlling test function

(1) Test request flag, test enable flag

The test request flag (IRQ×××) is set to "1" when a test request (INT×××) is generated. When the test processing is completed, it must be cleared to "0" by software.

The test enable flag (IE×××) is annexed to each test request flag. When it is "1", a standby release signal is enabled; and when it is "0", the signal is disabled.

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

Table 6-6 lists the set signals for the test request flags.

Table 6-6. Set Signal for Test Request Flag

Test Request Flag	Set Signal for Test Request Flag	Test Enable Flag
IRQW	Set by a signal sent from the watch timer.	IEW
IRQ2	Set by either the rising edge detection of a signal input to the INT2/P12 pin or the falling edge detection of a signal input to the KR0/P60 to KR3/P63 pins. The detection edge is selected by the INT2 edge detection mode register (IM2).	IE2

(2) INT2, key interrupt (KR0 to KR3) hardware

Figure 6-10 shows the configuration of INT2 and KR0 to KR3.

The IRQ2 set signal is output by the edge detection at the following two series of pins. The pin is selected by the INT2 edge detection mode register (IM2).

(a) Rising edge detection of input to INT2 pin

The IRQ2 is set when the rising edge of a signal input to the INT2 pin is detected.

(b) Falling edge detection of a signal input to KR0 to KR3 pins (key interrupt)

A pin which is used for the interrupt input is selected among the KR0 to KR3 pins by the INT2 edge detection mode register (IM2). The IRQ2 is set by the falling edge detection of a signal input to a selected pin.

Figure 6-11 shows the format of the IM2. The IM2 is set by a 4-bit manipulation instruction. All the bits are cleared to “0” by a $\overline{\text{RESET}}$ signal and the rising edge specification by the INT2 is adopted.

Figure 6-10. INT2 and KR0 to KR3 Block Diagram

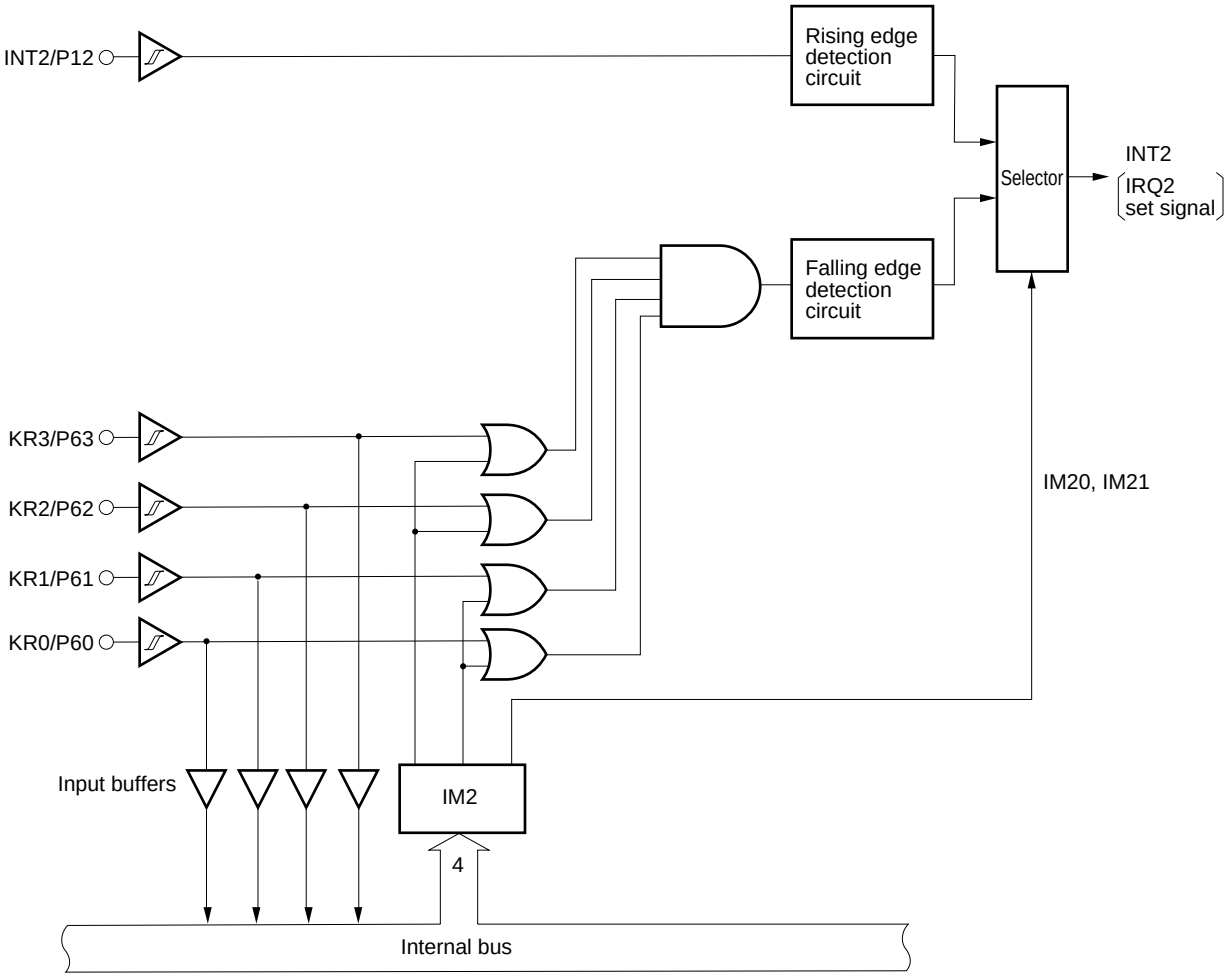


Figure 6-11. Format of INT2 Edge Detection Mode Register (IM2)

Address	3	2	1	0	Symbol
FB6H	0	0	IM21	IM20	IM2

IM21	IM20	INT2 Test Source	Interrupt Input Pin
0	0	Rising edge specification of INT2 pin input	INT2 (1)
0	1	Setting prohibited	—
1	0	Falling edge specification of any KRX pin input	KR2, KR3 (2)
1	1		KR0 to KR3 (4)

Cautions 1. If the edge detection mode register is changed, the test request flag may be set in some cases; therefore the test input must be disabled beforehand to change the mode register and the test request flag must be cleared by a CLR1 instruction, and then a test input must be enabled.

2. When a low-level signal is input to a pin among those pins selected for falling edge detection, the IRQ2 is not set even if falling edges are input to the other pins.

[MEMO]

CHAPTER 7 STANDBY FUNCTION

The μPD753108 has a standby function that reduces the power dissipation of the system. This standby function can be implemented in the following two modes:

- STOP mode
- HALT mode

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

(1) STOP mode

In this mode, the main system clock oscillation circuit is stopped and therefore, the entire system is stopped. The current dissipation of the CPU is substantially reduced.

Moreover, the contents of the data memory can be retained at a low voltage (V_{DD} = 1.8 V MIN.). This mode is therefore useful for retaining the data memory contents with an extremely low current dissipation.

The STOP mode of the μPD753108 can be released by an interrupt request; therefore, the microcontroller can operate intermittently. However, because a wait time is required for stabilizing the oscillation of the clock oscillation circuit when the STOP mode has been released, use the HALT mode if processing must be started immediately after the standby mode has been released by an interrupt request.

(2) HALT mode

In this mode, the operating clock of the CPU is stopped. Oscillation of the system clock oscillation circuit continues. This mode does not reduce the 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 statuses of the I/O ports are processed in advance so that the current dissipation of the overall system can be minimized.

The following page describes the points to be noted in using the standby mode.

Cautions 1. The STOP mode can be used only when the system operates with the main system clock (oscillation of the subsystem clock cannot be stopped). The HALT mode can be used regardless of whether the system operates with the main system clock or subsystem clock.

2. If the STOP mode is set when the LCD controller/driver and watch timer operate with main system clock f_X, the operations of the LCD controller/driver and watch timer are stopped. To continue the operations of these, therefore, change the operating clock to subsystem clock f_{XT} before setting 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 selecting function and standby mode in combination, therefore, set the standby mode after the time required for selection 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 it.

7.1 Standby Mode Setting and Operation Status

Table 7-1. Operation Status in Standby Mode

Item \ Mode		STOP Mode	HALT Mode
Set instruction		STOP instruction	HALT instruction
System clock when set		Settable only when the main system clock is used.	Settable both by the main system clock and subsystem clock.
Operation status	Clock generator	Only the main system clock stops oscillation.	Only the CPU clock Φ halts (oscillation continues).
	Basic interval timer/ Watchdog timer	Operation stops.	Operable only when the main system clock is oscillated (The IRQBT is set in the reference interval).
	Serial interface	Operable only when an external $\overline{SCK}$ input is selected as the serial clock.	Operable only when an external $\overline{SCK}$ input is selected as the serial clock or when the main system clock is oscillated.
	Timer/event counter	Operable only when a signal input to the TI0 to TI2 pins is specified as the count clock.	Operable only when a signal input to the TI0 to TI2 pins is specified as the count clock or when the main system clock is oscillated.
	Watch timer	Operable when f_{XT} is selected as the count clock.	Operable.
	LCD controller/driver	Operable only when f_{XT} is selected as the LCDCL.	Operable.
	External interrupt	The INT1, 2, and 4 are operable. Only the INT0 is not operated**Note**.	
	CPU	The operation stops.	
Release signal		Interrupt request signal sent from the operable hardware enabled by the interrupt enable flag or $\overline{RESET}$ signal input.	

Note Can operate only when the noise eliminator is not used (IM02 = 1) by bit 2 of the edge detection mode register (IM0).

The STOP mode is set by a STOP instruction and the HALT mode is set by a HALT instruction. The STOP instruction and HALT instruction set bit 3 and bit 2, respectively, of the PCC.

A NOP instruction must be placed following the STOP instruction and HALT instruction.

When changing the CPU clock by the low-order 2 bits of the PCC, there may be a time difference between PCC updating and CPU clock change as shown in **Table 5-5 Maximum Time Required to Switch System to/from CPU Clocks**. Consequently, when the operating clock before the standby mode and the CPU clock after the standby mode is released are to be changed, the PCC must be updated and then the standby mode must be set after the machine cycle necessary to change the CPU clock has elapsed.

In the standby mode, the data items stored in all the registers and data memory such as the general-purpose register, flags, mode registers, and output latch which stop operation during the standby mode are held.

★ **Cautions 1. When the system is set in the STOP mode, the X2 pin is pulled up internally to V_{DD} with the 50-kΩ (TYP.) resistor.**

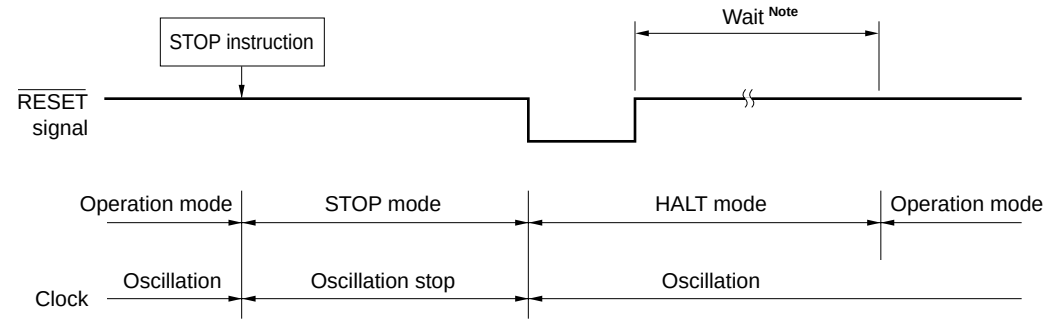
2. Before setting the standby mode, reset all the interrupt request flags.
If there is an interrupt source in which both the test request flag and test enable flag are set, the standby mode is released at the moment the system enters it (See Figure 6-1 Interrupt Control Circuit Block Diagram). However, when the STOP mode is set, the system enters the HALT mode immediately after a STOP instruction is executed and then returns to the operating mode after waiting for a time which is set in the BTM register.

7.2 Standby Mode Release

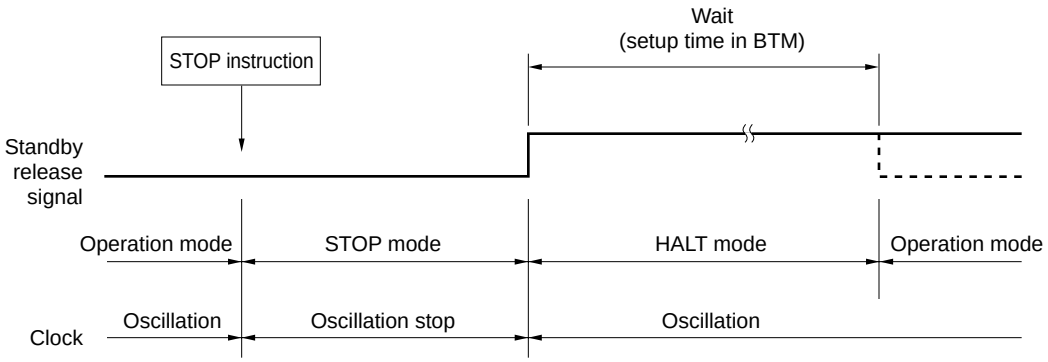
The standby mode (STOP or HALT) is released when an interrupt request signal enabled with an interrupt enable flag occurs or a $\overline{\text{RESET}}$ signal is generated. Figure 7-1 shows the standby mode release operation.

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



Note The following two wait times can be specified by the mask option.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz, 31.3 ms at 4.19 MHz)

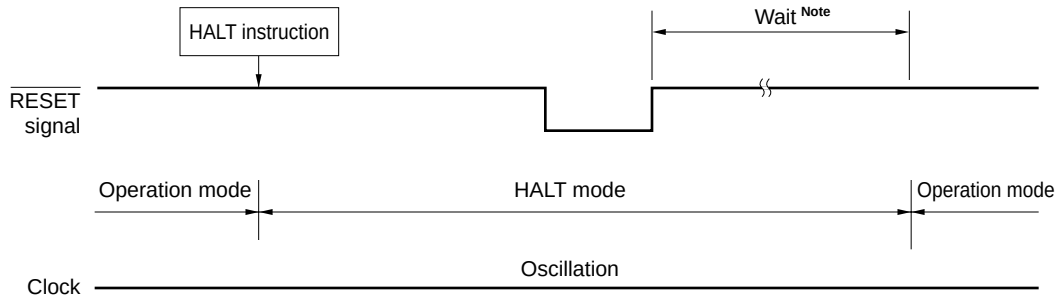
$2^{15}/f_X$ (5.46 ms at 6.00 MHz, 7.81 ms at 4.19 MHz)

However, the μPD75P3116 has no mask options and it is fixed to $2^{15}/f_X$.

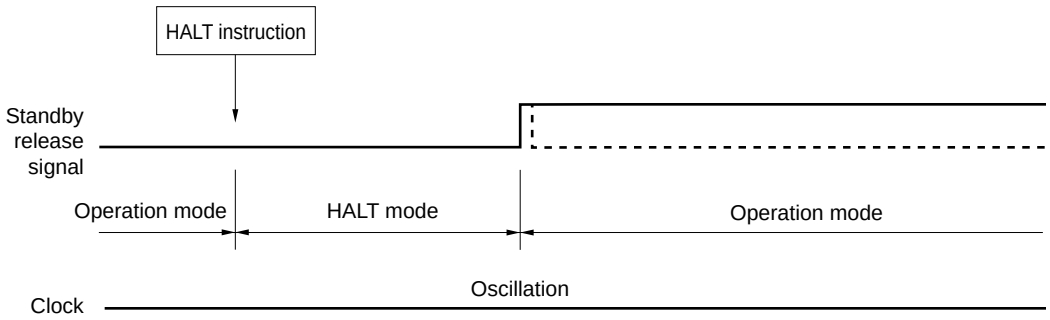
Remark Broken line: When the interrupt request to release the standby mode is acknowledged.

Figure 7-1. Standby Mode Release Operation (2/2)

(c) HALT mode release when a $\overline{\text{RESET}}$ signal is generated



(d) HALT mode release when an interrupt occurs



Note The following two wait times can be specified by the mask option.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz, 31.3 ms at 4.19 MHz)

$2^{15}/f_X$ (5.46 ms at 6.00 MHz, 7.81 ms at 4.19 MHz)

However, the μPD75P3116 has no mask options and it is fixed to $2^{15}/f_X$.

Remark Broken line: When the interrupt request to release the standby mode is acknowledged.

When the STOP mode has been released by an interrupt, the wait time is determined by the setting of BTM (refer to **Table 7-2**).

The time required for the oscillation to stabilize varies depending on the type of the resonator used and the supply voltage when the STOP mode has been released. Therefore, select the appropriate wait time depending on a given condition, and set BTM before setting the STOP mode.

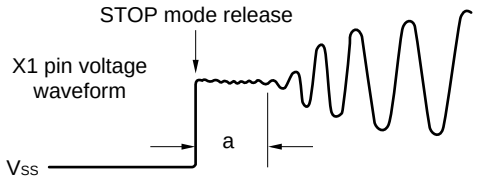
Table 7-2. Wait Time Selection by Using BTM

BTM3	BTM2	BTM1	BTM0	Wait Time[Note]	
				When f_X = 6.00 MHz	When f_X = 4.19 MHz
–	0	0	0	About $2^{20}/f_X$ (about 175 ms)	About $2^{20}/f_X$ (about 250 ms)
–	0	1	1	About $2^{17}/f_X$ (about 21.8 ms)	About $2^{17}/f_X$ (about 31.3 ms)
–	1	0	1	About $2^{15}/f_X$ (about 5.46 ms)	About $2^{15}/f_X$ (about 7.81 ms)
–	1	1	1	About $2^{13}/f_X$ (about 1.37 ms)	About $2^{13}/f_X$ (about 1.95 ms)
Other than the above				Setting prohibited	

Note This time does not include the time until oscillation is started after the STOP mode is released.

Caution The wait time that elapses when the STOP mode has been released does not include the time that elapses until the clock oscillation is started after the STOP mode has been released (a in Figure 7-2), regardless of whether the STOP mode has been released by the $\overline{\text{RESET}}$ signal or occurrence of an interrupt.

Figure 7-2. Wait Time when STOP Mode is Released



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 and INT2 (testable inputs), a vectored interrupt is not generated; therefore the operations identical to (a) above are performed.

★ 7.4 Mask Option Selection

In the standby function of the μPD753108, the wait time after the standby function release by the $\overline{\text{RESET}}$ signal generation can be selected from the following two types of times with the mask option.

<1> $2^{17}/f_X$ (21.8 ms: 6.0-MHz operation, 31.3 ms: 4.19-MHz operation)
<2> $2^{15}/f_X$ (5.46 ms: 6.0-MHz operation, 7.81 ms: 4.19-MHz operation)

The μPD75P3116, however, does not have mask option and its wait time is fixed to $2^{15}/f_X$.

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 an interrupt input or power failure by port input (use of INT4 to detect a power failure is recommended).
<2> Process the I/O ports (process so that the current dissipation is minimized).
It is important not to open the input port. Be sure to input a low or high level to it.
<3> Specify an interrupt that releases the standby mode (use of INT4 is effective. Clear the interrupt enable flags of the interrupts that do not release the standby mode).
<4> Specify the operation to be performed after the standby mode has been released (manipulate IME depending on whether interrupt processing is performed or not).
<5> Specify the CPU clock to be used after the standby mode has been released (to change the clock, make sure that the necessary machine cycles elapse before the standby mode is set).
<6> Select the wait time to elapse after the standby mode has been released.
<7> Set the standby mode (by using the STOP or HALT instruction).

By using the standby mode in combination with the system clock selecting function, low current dissipation and low-voltage operation can be realized.

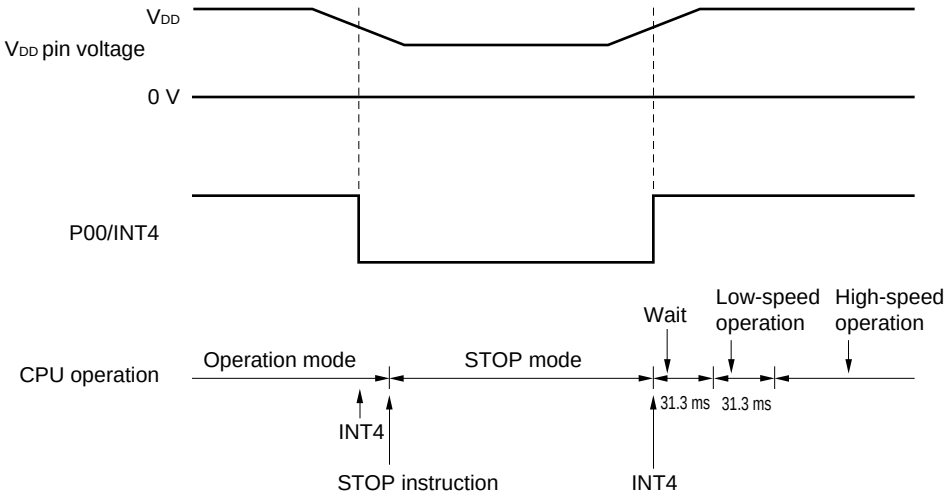
(1) Application example of STOP mode (f_X = 4.19 MHz)

<When using the STOP mode under the following conditions>

- The STOP mode is set at the falling edge of INT4 and released at the rising edge (INTBT is not used).
- All the I/O ports go into a high-impedance state (if the pins are externally processed so that the current dissipation is reduced in a high-impedance state).
- Interrupts INT0 and INTT0 are used in the program. However, these interrupts are not used to release the STOP mode.
- The interrupts are enabled even after the STOP mode has been released.
- After the STOP mode has been released, operation is started with the slowest CPU clock.
- The wait time that elapses after the mode has been released is about 31.3 ms.
- A wait time of 31.3 ms elapses until the power supply stabilizes after the mode has been released. The P00/INT4 pin is checked two times to eliminate chattering.

<Timing chart>

★



<Program example>

(INT4 processing program, MBE = 0)

```
VSUB4:  SKT     PORT0.0     ; P00 = 1 ?
        BR      PDOWN       ; Power down
        SET1    BTM.3       ; Power on
WAIT:   SKT     IRQBT       ; Waits for 31.3 ms
        BR      WAIT
        SKT     PORT0.0     ; Checks chattering
        BR      PDOWN
        MOV     A, #0011B
        MOV     PCC, A      ; Sets high-speed mode
        MOV     XA, #XXH    ; Sets port mode register
        MOV     PMGm, XA
        EI      IE0
        EI      IET0
        RETI
PDOWN:  MOV     A, #0       ; Lowest-speed mode
        MOV     PCC, A
        MOV     XA, #00H
        MOV     LCDM, XA    ; LCD display off
        MOV     LCDC, A
        MOV     PMGA, XA    ; I/O port in high-impedance state
        MOV     PMGB, XA
        DI      IE0         ; Disables INT0 and INTT0
        DI      IET0
        MOV     A, #1011B
        MOV     BTM, A      ; Wait time ≅ 31.3 ms
        NOP
        STOP                ; Sets STOP mode
        NOP
        RETI
```

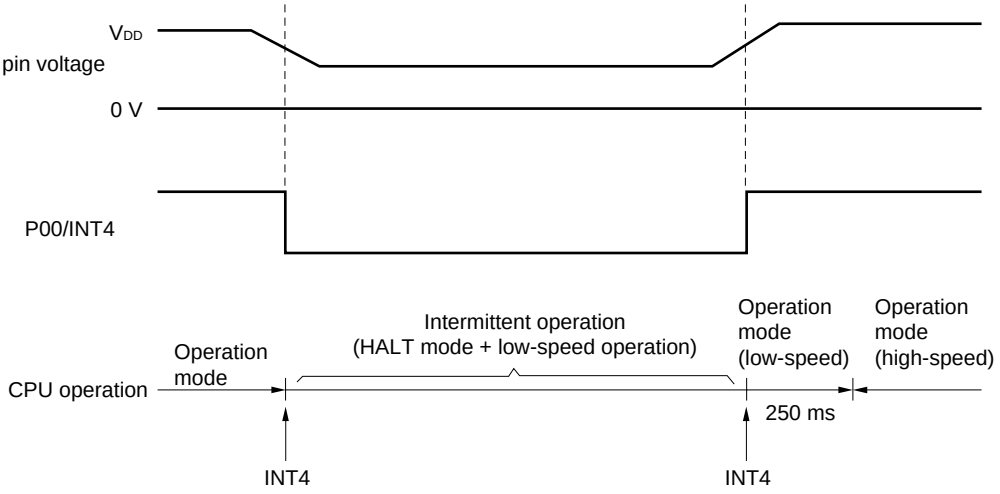
(2) Application of HALT mode (f_X = 4.19 MHz)

<To perform intermittent operation under the following conditions>

- The standby mode is set at the falling edge of INT4 and released at the rising edge.
- In the standby mode, an intermittent operation is performed at intervals of 250 ms (INTBT).
- INT4 and INTBT are assigned with the lower priority.
- The slowest CPU clock is selected in the standby mode.

<Timing chart>

★



<Program example>

```
(Initial setting)
            MOV     A, #0011B
            MOV     PCC, A          ; High-speed mode
            MOV     XA, #05
            MOV     WM, XA          ; Subsystem clock
            EI      IE4
            EI      IEW
            EI                      ; Enables interrupt
(Main routine)
            SKT     PORT0.0         ; Power supply OK?
            HALT                    ; Power down mode
            NOP                     ; Power supply OK?
            SKTCLR  IRQW            ; 0.5-sec flag?
            BR      MAIN            ; NO
            CALL    WATCH           ; Watch subroutine
MAIN:       .
            .
            .
            .
            .
(INT4 processing routine)
VINT4:      SKT     PORT0.0         ; Power supply OK?, MBE = 0
            BR      PDOWN
            CLR1    SCC.3           ; Main system clock starts oscillating
            MOV     A, #1000B
            MOV     BTM, A
WAIT1:      SKT     IRQBT           ; Waits for 250 ms
            BR      WAIT1
            SKT     PORT0.0         ; Checks chattering
            BR      PDOWN
            CLR1    SCC.0           ; Selects main system clock
            RETI
PDOWN:      MOV     XA, #00H
            MOV     LCDM, XA        ; LCD display off
            MOV     LCDC, A
            SET1    SCC.0           ; Selects subsystem clock
            MOV     A, #6
```

```
WAIT2:  INCS    A           ; Waits for 32 machine cycles or more[Note]
        BR      WAIT2
        SET1    SCC.3       ; Main system clock oscillation stops
        RETI
```

Note For how to select the system clock and CPU clock, refer to **5.2.3 Setting of system clock and CPU clock**.

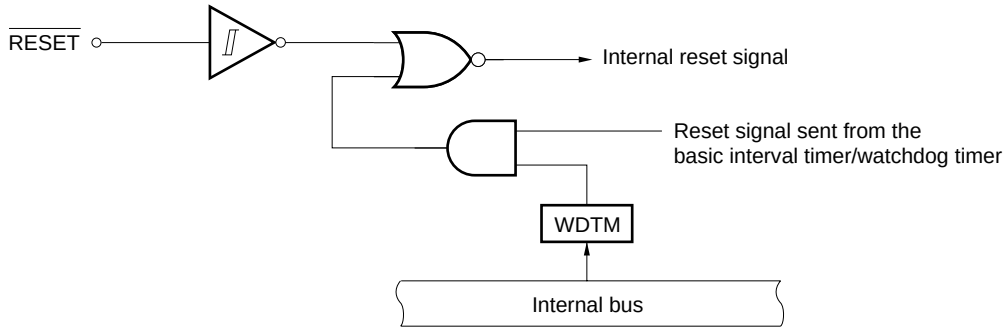
Caution To change the system clock from the main system clock to the subsystem clock, wait until the oscillation of the subsystem clock is stabilized.

[MEMO]

CHAPTER 8 RESET FUNCTION

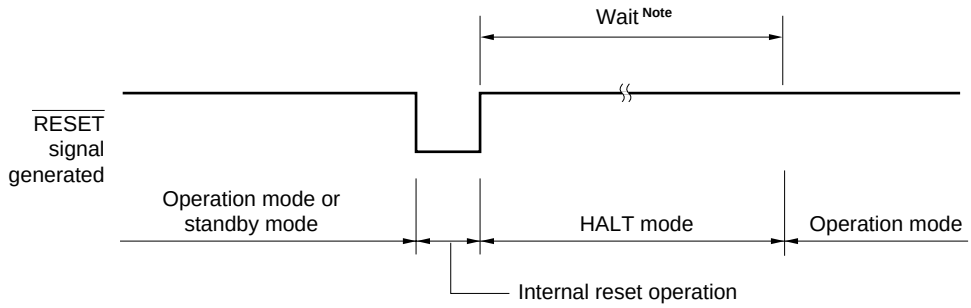
There are two reset inputs: external $\overline{\text{RESET}}$ signal and reset signal sent from the basic interval timer/watchdog timer. When either one of the reset signals are input, an internal reset signal is generated. Figure 8-1 shows the circuit diagram of the above two inputs.

Figure 8-1. Configuration of Reset Function



Generation of the $\overline{\text{RESET}}$ signal causes each device to be initialized as listed in Table 8-1. Figure 8-2 shows the timing chart of the reset operation.

Figure 8-2. Reset Operation by $\overline{\text{RESET}}$ Signal Generation



Note The following two times can be selected by the mask option.

$2^{17}/f_X$ (21.8 ms at 6.00 MHz or 31.3 ms at 4.19 MHz)

$2^{15}/f_X$ (5.46 ms at 6.00 MHz or 7.81 ms at 4.19 MHz)

However, the μPD75P3116 has no mask options and it is fixed to $2^{15}/f_X$.

Table 8-1. Status of Each Device After Reset (1/2)

Hardware			$\overline{\text{RESET}}$ Signal Generation in Standby Mode	$\overline{\text{RESET}}$ Signal Generation in Operation
Program counter (PC)		μPD753104	Sets the low-order 4 bits of program memory's address 0000H to the PC11 to PC8 and the contents of address 0001H to the PC7 to PC0.	Sets the low-order 4 bits of program memory's address 0000H to the PC11 to PC8 and the contents of address 0001H to the PC7 to PC0.
		μPD753106, μPD753108	Sets the low-order 5 bits of program memory's address 0000H to the PC12 to PC8 and the contents of address 0001H to the PC7 to PC0.	Sets the low-order 5 bits of program memory's address 0000H to the PC12 to PC8 and the contents of address 0001H to the PC7 to PC0.
		μPD75P3116	Sets the low-order 6 bits of program memory's address 0000H to the PC13 to PC8 and the contents of address 0001H to the PC7 to PC0.	Sets the low-order 6 bits of program memory's address 0000H to the PC13 to PC8 and the contents of address 0001H to the PC7 to PC0.
PSW	Carry flag (CY)		Held	Undefined
	Skip flag (SK0 to SK2)		0	0
	Interrupt status flag (IST0, IST1)		0	0
	Bank enable flag (MBE, RBE)		Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.	Sets the bit 6 of program memory's address 0000H to the RBE and bit 7 to the MBE.
Stack pointer (SP)			Undefined	Undefined
Stack bank 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/event counter (T0)	Counter (T0)		0	0
	Modulo register (TMOD0)		FFH	FFH
	Mode register (TM0)		0	0
	TOE0, TOUT F/F		0, 0	0, 0
Timer/event counter (T1)	Counter (T1)		0	0
	Modulo register (TMOD1)		FFH	FFH
	Mode register (TM1)		0	0
	TOE1, TOUT F/F		0, 0	0, 0

Table 8-1. Status of Each Device After Reset (2/2)

Hardware		$\overline{\text{RESET}}$ Signal Generation in Standby Mode	$\overline{\text{RESET}}$ Signal Generation in Operation
Timer/event counter (T2)	Counter (T2)	0	0
	Modulo register (TMOD2)	FFH	FFH
	High-level period setting modulo register (TMOD2H)	FFH	FFH
	Mode register (TM2)	0	0
	TOE2, TOUT F/F	0, 0	0, 0
	REMC, NRZ, NRZB	0, 0, 0	0, 0, 0
	TGCE	0	0
Watch timer	Mode register (WM)	0	0
Serial interface	Shift register (SIO)	Held	Undefined
	Operation mode register (CSIM)	0	0
	SBI control register (SBIC)	0	0
	Slave address register (SVA)	Held	Undefined
Clock generator, clock output circuit	Processor clock control register (PCC)	0	0
	System clock control register (SCC)	0	0
	Clock output mode register (CLOM)	0	0
Subsystem clock 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 (IRQ×××)	Reset (0)	Reset (0)
	Interrupt enable flag (IE×××)	0	0
	Interrupt priority selection register (IPS)	0	0
	INT0, 1, 2 mode registers (IM0, IM1, IM2)	0, 0, 0	0, 0, 0
Digital port	Output buffer	Off	Off
	Output latch	Cleared (0)	Cleared (0)
	I/O mode registers (PMGA, B, C)	0	0
	Pull-up resistor setting register (POGA, B)	0	0
Bit sequential buffer (BSB0 to BSB3)		Held	Undefined

[MEMO]

CHAPTER 9 WRITING AND VERIFYING PROM (PROGRAM MEMORY)

The program memory of the μPD75P3116 is a one-time PROM. The memory capacity is as follows:

μPD75P3116: 16384 words × 8 bits

To write or verify this one-time PROM, the pins shown in Table 9-1 are used. Note that no address input pins are used and that the address is updated by inputting a clock from the X1 pin.

Table 9-1. Pins Used to Write or Verify Program Memory

Pin Name	Function
V_{PP}	Applies program voltage for writing or verifying program memory (usually, V_{DD}).
X1, X2	Inputs clock to update address when program memory is written or verified. Inverted signal of X1 pin is input to X2 pin.
MD0/P30 to MD3/P33	Selects operation mode when program memory is written or verified.
D0/P60 to D3/P63 (low-order 4) D4/P50 to D7/P53 (high-order 4)	Inputs or outputs 8-bit data when program memory is written or verified.
V_{DD}	Supplies power supply voltage. Supplies 1.8 to 5.5 V for normal operation and +6 V when program memory is written or verified.

Cautions 1. The program memory contents of the μPD75P3116 cannot be erased by ultraviolet rays because the μPD75P3116 is not provided with a window for erasure.

2. Connect the pins not used for writing or verifying the program memory to V_{SS}.

9.1 Operation Mode for Writing/Verifying Program Memory

When +6 V is applied to the V_{DD} pin and +12.5 V is applied to the V_{PP} pin of the μPD75P3116, the program memory write/verify mode is set. In this mode, the following operation modes can be selected by using the MD0 through MD3 pins.

Table 9-2. Operation Mode

Specifies Operation Mode						Operation Mode
V_{PP}	V_{DD}	MD0	MD1	MD2	MD3	
+12.5 V	+6 V	H	L	H	L	Clears program memory address to 0
		L	H	H	H	Write mode
		L	L	H	H	Verify mode
		H	×	H	H	Program inhibit mode

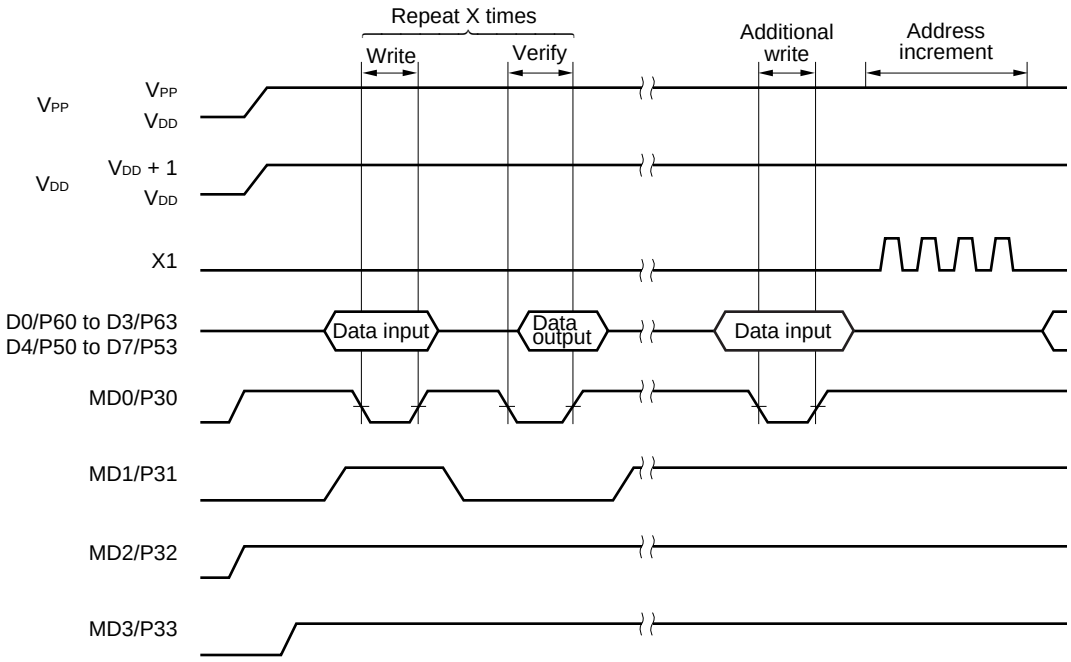
×: L or H

★ 9.2 Writing Program Memory

The program memory can be written in the following procedure at high speed:

(1) Pull down the unused pins to V_{SS} via a resistor. The X1 pin is low level.
(2) Supply 5 V to the V_{DD} and V_{PP} pins.
(3) Wait for 10 μs.
(4) Set the program memory address 0 clear mode.
(5) Supply 6 V to V_{DD} and 12.5 V to V_{PP}.
(6) Write data in the 1-ms write mode.
(7) Set the verify mode. If the data have been correctly written, proceed to (8). If not, repeat (6) and (7).
(8) Additional writing of (number of times data have been written in (6) and (7): X) $\times$ 1 ms
(9) Input a pulse four times to the X1 pin to update the program memory address (by one).
(10) Repeat (6) through (9) until the last address is written.
(11) Set the program memory address 0 clear mode.
(12) Change the voltage applied to the V_{DD} and V_{PP} pins to 5 V.
(13) Turn off the power supply.

Steps (2) through (9) above are illustrated below.

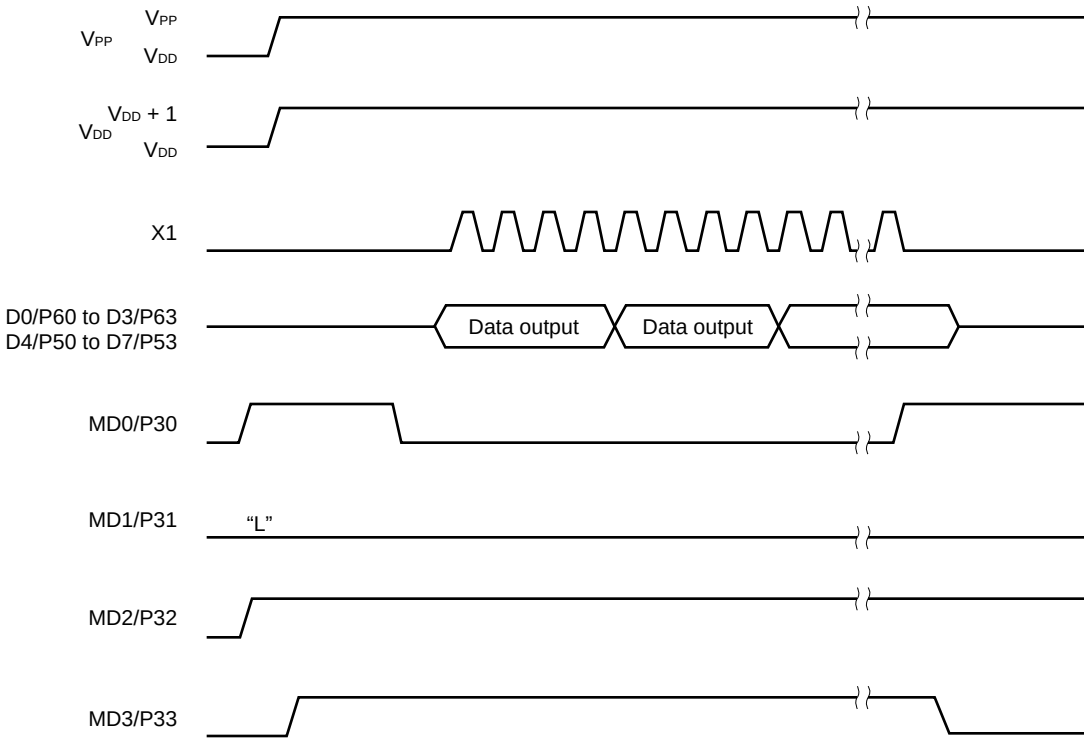


★ 9.3 Reading Program Memory

The contents of the program memory can be read in the following procedure. Reading is performed in the verify mode.

(1) Pull down the unused pins to V_{SS} via a resistor. The X1 pin is low level.
(2) Supply 5 V to the V_{DD} and V_{PP} pins.
(3) Wait for 10 μs.
(4) Set the program memory address 0 clear mode.
(5) Supply 6 V to V_{DD} and 12.5 V to V_{PP}.
(6) Verify mode. Data at each address is sequentially output while four clock pulses are input to the X1 pin.
(7) Set the program memory address 0 clear mode.
(8) Change the voltage applied to the V_{DD} and V_{PP} pins to 5 V.
(9) Turn off the power supply.

Steps (2) through (7) above are illustrated below.



9.4 Screening of One-Time PROM

Because of its structure, it is difficult for NEC to completely test the one-time PROM product before shipment. It is therefore recommended that screening be performed to verify the PROM contents after the necessary data has been written to the PROM and the product has been stored under the following conditions.

Storage Temperature	Storage Time
125°C	24 hours

★ NEC offers for a fee one-time PROM writing, marking, screening, and verifying service called QTOP™ microcontroller. For details, consult an NEC sales representative.

[MEMO]

CHAPTER 10 MASK OPTION

10.1 Pins

The pins of the μPD753108 have the following mask options:

Table 10-1. Selecting Mask Option of Pin

Pin	Mask Option
P50 to P53	Pull-up resistor can be connected in 1-bit units.
V_{LC0} to V_{LC2}, BIAS	LCD drive power supplying split resistors can be connected to four pins at once.

10.1.1 Mask option of P50 through P53

P50 through P53 (port 5) can be connected with pull-up resistors by mask option. The mask option can be specified in 1-bit units.

If the pull-up resistor is connected by mask option, port 5 goes high on reset. If the pull-up resistor is not connected, the port goes into a high-impedance state on reset.

Pull-up resistor cannot be connected by mask option in the μPD75P3116.

10.1.2 Mask option of V_{LC0} through V_{LC2}

Split resistors can be connected to the V_{LC0} through V_{LC2} pins (LCD drive power supply) and BIAS pin (external split resistor cutting pin) by mask option. Therefore, LCD drive power can be supplied without an external split resistor according to each bias (for details, refer to **5.7.8 Supply of LCD drive power V_{LC0}, V_{LC1}, and V_{LC2}**).

The following three mask options can be selected.

<1> No split resistor is connected.
<2> A 10-kΩ (typ.) split resistor is connected.
<3> A 100-kΩ (typ.) split resistor is connected.

The mask option is specified for the V_{LC0} through V_{LC2} and BIAS pins at once and cannot be specified in 1-pin units.

The BIAS pin goes low on reset when the split resistor is connected to this pin by mask option. When the split resistor is not connected, the BIAS pin goes into a high-impedance state on reset.

The μPD75P3116 does not have a mask option, and cannot be connected with a split resistor. Connect an external split resistor to the μPD75P3116, if necessary.

10.2 Mask Option of Standby Function

The standby function of the μPD753108 allows you to select wait time by using a mask option. The wait time is required for the CPU to return to the normal operation mode after the standby function has been released by the $\overline{\text{RESET}}$ signal (for details, refer to **7.2 Standby Mode Release**).

The following two wait times can be selected:

<1> $2^{17}/f_X$ (21.8 ms when f_X = 6.00 MHz; 31.3 ms when f_X = 4.19 MHz)
<2> $2^{15}/f_X$ (5.46 ms when f_X = 6.00 MHz; 7.81 ms when f_X = 4.19 MHz)

The μPD75P3116 does not have a mask option and its wait time is fixed to $2^{15}/f_X$.

CHAPTER 11 INSTRUCTION SET

The instruction set of the μPD753108 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 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 section **3.2 Bank Configuration of General-Purpose Registers**.

11.1 Unique Instructions

This section describes the unique instructions of the μPD753108's instruction set.

11.1.1 GETI instruction

The GETI instruction converts the following instructions into 1-byte instructions:

(a) Subroutine call instruction to 16 Kbytes space (0000H to 3FFFH)
(b) Branch instruction to 16 Kbytes space (0000H to 3FFFH)
(c) Any 2-byte, 2-machine cycle instruction (except BRCB and CALLF instructions)
(d) Combination of two 1-byte instructions

The GETI instruction references a table at addresses 0020H through 007FH of the program memory and executes the referenced 2-byte data as an instruction of (a) to (d). Therefore, 48 types of instructions can be converted into 1-byte instructions.

If instructions that are frequently used are converted into 1-byte instructions by using this GETI instruction, the number of bytes of the program can be substantially decreased.

11.1.2 Bit manipulation instruction

The μPD753108 has reinforced bit test, bit transfer, and bit Boolean (AND, OR, and XOR) instructions, in addition to the ordinary bit manipulation (set and clear) instructions.

The bit to be manipulated is specified in the bit manipulation addressing mode. Three types of bit manipulation addressing modes can be used. The bits manipulated in each addressing mode are shown in Table 11-1.

Table 11-1. Types of Bit Manipulation Addressing Modes and Specification Range

Addressing	Peripheral Hardware That Can Be Manipulated	Addressing Range of Bit That Can Be Manipulated
fmem.bit	RBE, MBE, IST1, IST0, SCC, IE×××, IRQ×××	FB0H through FBFH
	PORT0 to 3, 5, 6, 8, 9	FF0H through FFFH
pmem.@L	BSB0 to 3, PORT0 to 3, 5, 6, 8, 9	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. ×××: 0, 1, 2, 4, BT, T0, T1, T2, W, CSI

2. MB = MBE•MBS

11.1.3 String-effect instruction

The μPD753108 has the following two types of string-effect instructions:

(a) MOV A, #n4 or MOV XA, #n8
(b) MOV HL, #n8

"String effect" means locating these two types of instructions at contiguous addresses.

```
Example  A0    : MOV   A, #0
         A1    : MOV   A, #1
         XA7   : MOV   XA, #07
```

When string-effect instructions are arranged as shown in this example, and if the address executed first is A0, the two instructions following this address are replaced with the NOP instructions. If the address executed first is A1, the following one instruction is replaced with the NOP instruction. In other words, only the instruction that is executed first is valid, and all the string-effect instructions that follow are processed as NOP instructions.

By using these string-effect instructions, constants can be efficiently set to the accumulator (A register or register pair XA) and data pointer (register pair HL).

11.1.4 Base number adjustment instruction

Some application requires that the result of addition or subtraction of 4-bit data (which is carried out in binary number) be converted into a decimal number or into a number with a base of 6, such as time.

Therefore, the μPD753108 is provided with base number adjustment instructions that adjusts the result of addition or subtraction of 4-bit data into a number with any base.

(a) Base adjustment of result of addition

Where the base number to which the result of addition executed is to be adjusted is m, the contents of the accumulator and memory (HL) are added in the following combination, and the result is adjusted to a number with a base of m:

```
ADDS   A, #16–m
ADDC   A, @HL   ; A, CY ← A + (HL) + CY
ADDS   A, #m
```

Occurrence of an overflow is indicated by the carry flag.

If a carry occurs as a result of executing the ADDC A, @HL instruction, the ADDS A, #n4 instruction is skipped. If a carry does not occur, the ADDS A, #n4 instruction is executed. At this time, however, the skip function of the instruction is disabled, and the following instruction is not skipped even if a carry occurs as a result of addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

Example To add accumulator and memory in decimal

```
ADDS  A, #6
ADDC  A, @HL    ; A, CY ← A + (HL) + CY
ADDS  A, #10
        ⋮
```

(b) Base adjustment of result of subtraction

Where the base number into which the result of subtraction executed is to be adjusted is m, the contents of memory (HL) are subtracted from those of the accumulator in the following combination, and the result of subtraction is adjusted to a number with a base of m:

```
SUBC A, @HL
ADDS A, #m
```

Occurrence of an underflow is indicated by the carry flag.

If a borrow does not occur as a result of executing the SUBC A, @HL instruction, the following ADDS A, #n4 instruction is skipped. If a borrow occurs, the ADDS A, #n4 instruction is executed. At this time, the skip function of this instruction is disabled, and the following instruction is not skipped, even if a carry occurs as a result of addition. Therefore, a program can be written after the ADDS A, #n4 instruction.

11.1.5 Skip instruction and number of machine cycles required for skipping

The instruction set of the μPD753108 configures a program where instructions may be or may not be skipped if a given condition is satisfied.

If a skip condition is satisfied when a skip instruction is executed, the instruction next to the skip instruction is skipped and the instruction after the next is executed.

When a skip occurs, the number of machine cycles required for skipping is:

(a) If the instruction that follows the skip instruction (i.e., the instruction to be skipped) is a 3-byte instruction (BR !addr, BRA !addr1, CALL !addr, or CALLA !addr1 instruction): 2 machine cycles

(b) Instruction other than (a): 1 machine cycle

11.2 Instruction Sets and their Operations

(1) Expression formats and description methods of operands

The operand is described in the operand column of each instruction in accordance with the description method for the operand expression format of the instruction. For details, refer to **RA75X ASSEMBLER PACKAGE USER'S 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 registers shown in Figure 3-7 can be described. However, there are restrictions in the labels that can be described for fmem and pmem. For details, see **Table 3-1 Addressing Mode** and **Figure 3-7 μPD753108 I/O Map**.

Expression Format	Description Method
reg	X, A, B, C, D, E, H, L
reg1	X, B, C, D, E, H, L
rp	XA, BC, DE, HL
rp1	BC, DE, HL
rp2	BC, DE
rp'	XA, BC, DE, HL, XA', BC', DE', HL'
rp'1	BC, DE, HL, XA', BC', DE', HL'
rpa	HL, HL+, HL–, DE, DL
rpa1	DE, DL
n4	4-bit immediate data or label
n8	8-bit immediate data or label
mem	8-bit immediate data or label[Note]
bit	2-bit immediate data or label
fmem	FB0H-FBFH, FF0H-FFFH immediate data or label
pmem	FC0H-FFFH immediate data or label
addr	000H-FFFH immediate data or label (μPD753104) 0000H-17FFH immediate data or label (μPD753106) 0000H-1FFFH immediate data or label (μPD753108) 0000H-3FFFH immediate data or label (μPD7P3116)
addr1	000H-FFFH immediate data or label (μPD753104) 0000H-17FFH immediate data or label (μPD753106) 0000H-1FFFH immediate data or label (μPD753108) 0000H-3FFFH immediate data or label (μPD75P3116)
caddr	12-bit immediate data or label
faddr	11-bit immediate data or label
taddr	20H-7FH immediate data (where bit 0 = 0) or label
PORTn	PORT0-PORT3, PORT5, PORT6, PORT8, PORT9
IE×××	IEBT, IET0-IET2, IE0-IE2, IE4, IECSI, IEW
RBn	RB0-RB3
MBn	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 = 0 to 3, 5, 6, 8, 9)
IME	: Interrupt master enable flag
IPS	: Interrupt priority selection register
IE×××	: Interrupt enable flag
RBS	: Register bank selection register
MBS	: Memory bank selection register
PCC	: Processor clock control register
.	: Separation between address and bit
(××)	: The contents addressed by ××
××H	: Hexadecimal data

(3) Explanation of symbols under addressing area column

*1	MB = MBE • MBS (MBS = 0, 1, 15)		Data memory addressing
*2	MB = 0		
*3	MBE = 0 : MB = 0 (00H-7FH) 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	μPD753104	addr = 000H-FFFH	Program memory addressing
	μPD753106	addr = 0000H-17FFH	
	μPD753108	addr = 0000H-1FFFH	
	μPD75P3116	addr = 0000H-3FFFH	
*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	μPD753104	caddr = 000H-FFFH	
	μPD753106	caddr = 0000H-0FFFH (PC_{12} = 0) or 1000H-17FFH (PC_{12} = 1)	
	μPD753108	caddr = 0000H-0FFFH (PC_{12} = 0) or 1000H-1FFFH (PC_{12} = 1)	
	μPD75P3116	caddr = 0000H-0FFFH ($PC_{13, 12}$ = 00B) or 1000H-1FFFH ($PC_{13, 12}$ = 01B) or 2000H-2FFFH ($PC_{13, 12}$ = 10B) or 3000H-3FFFH ($PC_{13, 12}$ = 11B)	
*9	faddr = 0000H-07FFH		
*10	taddr = 0020H-007FH		
*11	μPD753104	addr1 = 000H-FFFH (Only in Mk II mode)	
	μPD753106	addr1 = 0000H-17FFH (Only in Mk II mode)	
	μPD753108	addr1 = 0000H-1FFFH (Only in Mk II mode)	
	μPD75P3116	addr1 = 0000H-3FFFH (Only in Mk II mode)	

Remarks 1. MB indicates memory bank that can be accessed.

2. In *2, MB = 0 independently of how MBE and MBS are set.

3. In *4 and *5, MB = 15 independently of how MBE and MBS are set.

4. *6 to *11 indicate the areas that can be addressed.

(4) Explanation of number of machine cycles column

S denotes the number of machine cycles required by skip operation when a skip instruction is executed. The value of S varies as follows.

- When no skip is made ..... S = 0
- When the skipped instruction is a 1- or 2-byte instruction ..... S = 1
- When the skipped instruction is a 3-byte instruction**Note** ..... S = 2

Note 3-byte instruction: BR !addr, BRA !addr1, CALL !addr or CALLA !addr1 instruction

Caution The GETI instruction is skipped in one machine cycle.

One machine cycle is equal to one cycle of CPU clock (= t_{CY}); time can be selected from among four types by setting PCC (See **Figure 5-12 Processor Clock Control Register Format**).

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Transfer	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	• μPD753104 XA ← $(PC_{11-8} + DE)_{ROM}$		
					• μPD753106, 753108 XA ← $(PC_{12-8} + DE)_{ROM}$		
					• μPD75P3116 XA ← $(PC_{13-8} + DE)_{ROM}$		
		XA, @PCXA	1	3	• μPD753104 XA ← $(PC_{11-8} + XA)_{ROM}$		
					• μPD753106, 753108 XA ← $(PC_{12-8} + XA)_{ROM}$		
					• μPD75P3116 XA ← $(PC_{13-8} + XA)_{ROM}$		
		XA, @BCDE	1	3	XA ← $(BCDE)_{ROM}$**Note**	*6	
		XA, @BCXA	1	3	XA ← $(BCXA)_{ROM}$**Note**	*6	
Bit transfer	MOV1	CY, fmem.bit	2	2	CY ← (fmem.bit)	*4	
		CY, pmem.@L	2	2	CY ← $(pmem_{7-2} + L_{3-2}.bit(L_{1-0}))$	*5	
		CY, @H+mem.bit	2	2	CY ← $(H + mem_{3-0}.bit)$	*1	
		fmem.bit, CY	2	2	(fmem.bit) ← CY	*4	
		pmem.@L, CY	2	2	$(pmem_{7-2} + L_{3-2}.bit(L_{1-0}))$ ← CY	*5	
		@H+mem.bit, CY	2	2	$(H + mem_{3-0}.bit)$ ← CY	*1	
Operation	ADDS	A, #n4	1	1 + S	A ← A + n4		carry
		XA, #n8	2	2 + S	XA ← XA + n8		carry
		A, @HL	1	1 + S	A ← A + (HL)	*1	carry
		XA, rp'	2	2 + S	XA ← XA + rp'		carry
		rp'1, XA	2	2 + S	rp'1 ← rp'1 + XA		carry
	ADDC	A, @HL	1	1	A, CY ← A + (HL) + CY	*1	
		XA, rp'	2	2	XA, CY ← XA + rp' + CY		
		rp'1, XA	2	2	rp'1, CY ← rp'1 + XA + CY		

Note When the μPD753104 is used, set 0 to the B register.
When the μPD753106 and 753108 are used, only low-order 1 bit is valid in B register.
When the μPD75P3116 is used, only low-order 2 bits are valid.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Operation	SUBS	A, @HL	1	1 + S	A ← A – (HL)	*1	borrow
		XA, rp'	2	2 + S	XA ← XA – rp'		borrow
		rp'1, XA	2	2 + S	rp'1 ← rp'1 – XA		borrow
	SUBC	A, @HL	1	1	A, CY ← A – (HL) – CY	*1	
		XA, rp'	2	2	XA, CY ← XA – rp' – CY		
		rp'1, XA	2	2	rp'1, CY ← rp'1 – XA – CY		
	AND	A, #n4	2	2	A ← A ∧ n4		
		A, @HL	1	1	A ← A ∧ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∧ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∧ XA		
	OR	A, #n4	2	2	A ← A ∨ n4		
		A, @HL	1	1	A ← A ∨ (HL)	*1	
		XA, rp'	2	2	XA ← XA ∨ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ∨ XA		
	XOR	A, #n4	2	2	A ← A ⊻ n4		
		A, @HL	1	1	A ← A ⊻ (HL)	*1	
		XA, rp'	2	2	XA ← XA ⊻ rp'		
		rp'1, XA	2	2	rp'1 ← rp'1 ⊻ XA		
Accumulator manipulation	RORC	A	1	1	CY ← A_0, A_3 ← CY, A_{n-1} ← A_n		
	NOT	A	2	2	A ← $\overline{A}$		
Increment and decrement	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'

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Carry flag manipulation	SET1	CY	1	1	CY ← 1		
	CLR1	CY	1	1	CY ← 0		
	SKT	CY	1	1 + S	Skip if CY = 1		CY = 1
	NOT1	CY	1	1	CY ← $\overline{CY}$		
Memory bit manipulation	SET1	mem. bit	2	2	(mem.bit) ← 1	*3	
		fmem. bit	2	2	(fmem.bit) ← 1	*4	
		pmem. @L	2	2	($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0})) ← 1	*5	
		@H+mem. bit	2	2	(H + mem_{3-0}.bit) ← 1	*1	
	CLR1	mem. bit	2	2	(mem.bit) ← 0	*3	
		fmem. bit	2	2	(fmem.bit) ← 0	*4	
		pmem. @L	2	2	($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0})) ← 0	*5	
		@H+mem. bit	2	2	(H + mem_{3-0}.bit) ← 0	*1	
	SKT	mem. bit	2	2 + S	Skip if (mem.bit) = 1	*3	(mem.bit) = 1
		fmem. bit	2	2 + S	Skip if (fmem.bit) = 1	*4	(fmem.bit) = 1
		pmem. @L	2	2 + S	Skip if ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0})) = 1	*5	(pmem.@L) = 1
		@H+mem. bit	2	2 + S	Skip if (H + mem_{3-0}.bit) = 1	*1	(@H+mem.bit) = 1
	SKF	mem. bit	2	2 + S	Skip if (mem.bit) = 0	*3	(mem.bit) = 0
		fmem. bit	2	2 + S	Skip if (fmem.bit) = 0	*4	(fmem.bit) = 0
		pmem. @L	2	2 + S	Skip if ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0})) = 0	*5	(pmem.@L) = 0
		@H+mem. bit	2	2 + S	Skip if (H + mem_{3-0}.bit) = 0	*1	(@H+mem.bit) = 0
	SKTCLR	fmem. bit	2	2 + S	Skip if (fmem.bit) = 1 and clear	*4	(fmem.bit) = 1
		pmem. @L	2	2 + S	Skip if ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0})) = 1 and clear	*5	(pmem.@L) = 1
		@H+mem. bit	2	2 + S	Skip if (H + mem_{3-0}.bit) = 1 and clear	*1	(@H+mem.bit) = 1
	AND1	CY, fmem. bit	2	2	CY ← CY ∧ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ∧ ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ∧ (H + mem_{3-0}.bit)	*1	
	OR1	CY, fmem. bit	2	2	CY ← CY ∨ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ∨ ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ∨ (H + mem_{3-0}.bit)	*1	
	XOR1	CY, fmem. bit	2	2	CY ← CY ⊻ (fmem.bit)	*4	
		CY, pmem. @L	2	2	CY ← CY ⊻ ($pmem_{7-2}$ + L_{3-2}.bit(L_{1-0}))	*5	
		CY, @H+mem. bit	2	2	CY ← CY ⊻ (H + mem_{3-0}.bit)	*1	

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Branch	BR[Note]	addr	—	—	• μPD753104 $PC_{11-0} \leftarrow$ addr {Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.}	*6	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr {Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.}		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr {Select appropriate instruction from among BR !addr, BRCB !caddr and BR $addr according to the assembler being used.}		

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Branch	BR[Note]	addr1	—	—	• μPD753104 $PC_{11-0} \leftarrow$ addr (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)	*11	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr1 (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr1 (Select appropriate instruction from among BR !addr, BRA !addr1, BRCB !caddr and BR $addr1 according to the assembler being used.)		
		!addr	3	3	• μPD753104 $PC_{11-0} \leftarrow$ addr	*6	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr		
		$addr	1	2	• μPD753104 $PC_{11-0} \leftarrow$ addr	*7	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr		

Note The above operations in the shaded boxes can be performed only in the Mk II mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Branch	BR[Note 1]	$addr1	1	2	• μPD753104 $PC_{11-0} \leftarrow$ addr1	*7	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr1		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr1		
		PCDE	2	3	• μPD753104 $PC_{11-0} \leftarrow PC_{11-8}$ + DE		
					• μPD753106, 753108 $PC_{12-0} \leftarrow PC_{12-8}$ + DE		
					• μPD75P3116 $PC_{13-0} \leftarrow PC_{13-8}$ + DE		
		PCXA	2	3	• μPD753104 $PC_{11-0} \leftarrow PC_{11-8}$ + XA		
					• μPD753106, 753108 $PC_{12-0} \leftarrow PC_{12-8}$ + XA		
					• μPD75P3116 $PC_{13-0} \leftarrow PC_{13-8}$ + XA		
		BCDE	2	3	• μPD753104 $PC_{11-0} \leftarrow$ BCDE[Note 2]	*6	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ BCDE[Note 3]		
					• μPD75P3116 $PC_{13-0} \leftarrow$ BCDE[Note 4]		
		BCXA	2	3	• μPD753104 $PC_{11-0} \leftarrow$ BCXA[Note 2]	*6	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ BCXA[Note 3]		
					• μPD75P3116 $PC_{13-0} \leftarrow$ BCXA[Note 4]		

Notes 1. The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

2. “0” must be set to B register.

3. Only low-order one bit is valid in B register.

4. Only low-order two bits are valid in B register.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Branch	BRA[Note]	!addr1	3	3	• μPD753104 $PC_{11-0} \leftarrow$ addr1	*11	
					• μPD753106, 753108 $PC_{12-0} \leftarrow$ addr1		
					• μPD75P3116 $PC_{13-0} \leftarrow$ addr1		
	BRCB	!caddr	2	2	• μPD753104 $PC_{11-0} \leftarrow caddr_{11-0}$	*8	
					• μPD753106, 753108 $PC_{12-0} \leftarrow PC_{12} + caddr_{11-0}$		
					• μPD75P3116 $PC_{13-0} \leftarrow PC_{13, 12} + caddr_{11-0}$		
Subroutine stack control	CALLA[Note]	!addr1	3	3	• μPD753104 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) $\leftarrow PC_{11-0}$ (SP – 5) ← 0, 0, 0, 0 PC_{11-0} ← addr1, SP ← SP – 6	*11	
					• μPD753106, 753108 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) $\leftarrow PC_{11-0}$ (SP – 5) $\leftarrow$ 0, 0, 0, PC_{12} PC_{12-0} ← addr1, SP ← SP – 6		
					• μPD75P3116 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) $\leftarrow PC_{11-0}$ (SP – 5) $\leftarrow$ 0, 0, $PC_{13, 12}$ PC_{13-0} ← addr1, SP ← SP – 6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Subroutine stack control	CALL[Note]	!addr	3	3	• μPD753104 (SP – 3) ← MBE, RBE, 0, 0 (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{11-0} ← addr, SP ← SP – 4	*6	
					• μPD753106, 753108 (SP – 3) ← MBE, RBE, 0, PC_{12} (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{12-0} ← addr, SP ← SP – 4		
					• μPD75P3116 (SP – 3) ← MBE, RBE, $PC_{13, 12}$ (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{13-0} ← addr, SP ← SP – 4		
				4	• μPD753104 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, 0, 0 PC_{11-0} ← addr, SP ← SP – 6		
					• μPD753106, 753108 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, 0, PC_{12} PC_{12-0} ← addr, SP ← SP – 6		
					• μPD75P3116 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, $PC_{13, 12}$ PC_{13-0} ← addr, SP ← SP – 6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Subroutine stack control	CALLF[Note]	!faddr	2	2	• μPD753104 (SP – 3) ← MBE, RBE, 0, 0 (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{11-0} ← 0 + faddr, SP ← SP – 4	*9	
					• μPD753106, 753108 (SP – 3) ← MBE, RBE, 0, PC_{12} (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{12-0} ← 00 + faddr, SP ← SP – 4		
					• μPD75P3116 (SP – 3) ← MBE, RBE, $PC_{13, 12}$ (SP – 4) (SP – 1) (SP – 2) ← PC_{11-0} PC_{13-0} ← 000 + faddr, SP ← SP – 4		
				3	• μPD753104 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, 0, 0 PC_{11-0} ← 0 + faddr, SP ← SP – 6		
					• μPD753106, 753108 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, 0, PC_{12} PC_{12-0} ← 00 + faddr, SP ← SP – 6		
					• μPD75P3116 (SP – 2) ← ×, ×, MBE, RBE (SP – 6) (SP – 3) (SP – 4) ← PC_{11-0} (SP – 5) ← 0, 0, $PC_{13, 12}$ PC_{13-0} ← 000 + faddr, SP ← SP – 6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Subroutine stack control	RET[Note]		1	3	• μPD753104 $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ MBE, RBE, 0, 0 $\leftarrow$ (SP + 1) SP $\leftarrow$ SP + 4		
					• μPD753106, 753108 $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ MBE, RBE, 0, $PC_{12} \leftarrow$ (SP + 1) SP $\leftarrow$ SP + 4		
					• μPD75P3116 $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ MBE, RBE, 0, $PC_{13, 12} \leftarrow$ (SP + 1) SP $\leftarrow$ SP + 4		
					• μPD753104 ×, ×, MBE, RBE $\leftarrow$ (SP + 4) 0, 0, 0, 0, $\leftarrow$ (SP + 1) $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ SP $\leftarrow$ SP + 6		
					• μPD753106, 753108 ×, ×, MBE, RBE $\leftarrow$ (SP + 4) MBE, 0, 0, $PC_{12} \leftarrow$ (SP + 1) $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ SP $\leftarrow$ SP + 6		
					• μPD75P3116 ×, ×, MBE, RBE $\leftarrow$ (SP + 4) $PC_{11-0} \leftarrow (SP)\ (SP+3)\ (SP+2)$ MBE, 0, $PC_{13, 12} \leftarrow$ (SP + 1) SP $\leftarrow$ SP + 6		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Subroutine stack control	RETS[Note]		1	3 + S	• μPD753104 MBE, RBE, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) SP ← SP + 4 then skip unconditionally		Unconditional
					• μPD753106, 753108 MBE, 0, 0, PC_{12} ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) SP ← SP + 4 then skip unconditionally		
					• μPD75P3116 MBE, 0, $PC_{13, 12}$ ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) SP ← SP + 4 then skip unconditionally		
					• μPD753104 0, 0, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) ×, ×, MBE, RBE ← (SP + 4) SP ← SP + 6 then skip unconditionally		
					• μPD753106, 753108 0, 0, 0, PC_{12} ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) ×, ×, MBE, RBE ← (SP + 4) SP ← SP + 6 then skip unconditionally		
					• μPD75P3116 ×, ×, MBE, RBE ← (SP + 4) PC_{11-0} ← (SP) (SP + 3) (SP + 2) 0, 0, $PC_{13, 12}$ ← (SP + 1) SP ← SP + 6 then skip unconditionally		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Subroutine stack control	RETI[Note]		1	3	• μPD753104 MBE, RBE, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
					• μPD753106, 753108 MBE, RBE, 0, PC_{12} ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
					• μPD75P3116 MBE, RBE, $PC_{13, 12}$ ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
					• μPD753104 0, 0, 0, 0 ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
					• μPD753106, 753108 0, 0, 0, PC_{12} ← (SP + 1) PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
					• μPD75P3116 0, 0, $PC_{13, 12}$ ← SP + 1 PC_{11-0} ← (SP) (SP + 3) (SP + 2) PSW ← (SP + 4) (SP + 5) SP ← SP + 6		
	PUSH	rp	1	1	(SP – 1) (SP – 2) ← rp SP ← SP – 2		
		BS	2	2	(SP – 1) ← MBS, (SP – 2) ← RBS SP ← SP – 2		
	POP	rp	1	1	rp ← (SP + 1) (SP) SP ← SP + 2		
		BS	2	2	MBS ← (SP + 1), RBS ← (SP) SP ← SP + 2		
Interrupt control	EI		2	2	IME (IPS.3) ← 1		
		IE×××	2	2	IE××× ← 1		
	DI		2	2	IME (IPS.3) ← 0		
		IE×××	2	2	IE××× ← 0		

Note The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Input/output	IN[Note]	A, PORTn	2	2	A ← PORTn (n = 0-3, 5, 6, 8, 9)		
		XA, PORTn	2	2	XA ← PORTn + 1, PORTn (n = 8)		
	OUT[Note]	PORTn, A	2	2	PORTn ← A (n = 2, 3, 5, 6, 8, 9)		
		PORTn, XA	2	2	PORTn + 1, PORTn ← XA (n = 8)		
CPU control	HALT		2	2	Set HALT Mode (PCC.2 ← 1)		
	STOP		2	2	Set STOP Mode (PCC.3 ← 1)		
	NOP		1	1	No Operation		
Special	SEL	RBn	2	2	RBS ← n (n = 0-3)		
		MBn	2	2	MBS ← n (n = 0, 1, 15)		

Note 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.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Special	GETI[Note]	taddr	1	3	• μPD753104 • When TBR instruction PC11–0 ← (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
					• μPD753106, 753108 • When TBR instruction PC$_{12-0}$ ← (taddr)$_{4-0}$ + (taddr + 1)		
					• When TCALL instruction (SP – 4) (SP – 1) (SP – 2) ← PC$_{11-0}$ (SP – 3) ← MBE, RBE, 0, PC$_{12}$ PC$_{12-0}$ ← (taddr)$_{4-0}$ + (taddr + 1) SP ← SP – 4		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr + 1) instruction is executed.		Depending on the reference instruction
					• μPD75P3116 • When TBR instruction PC$_{13-0}$ ← (taddr)$_{5-0}$ + (taddr + 1)		
					• When TCALL instruction (SP – 4) (SP – 1) (SP – 2) ← PC$_{11-0}$ (SP + 1) ← MBE, RBE, 0, PC$_{13, 12}$ PC$_{13-0}$ ← (taddr)$_{5-0}$ + (taddr + 1) SP ← SP – 4		
					• When instruction other than TBR and TCALL instructions (taddr) (taddr + 1) instruction is executed.		Depending on the reference instruction

Note The TBR and TCALL instructions are the table definition assembler pseudo-instructions of the GETI instruction.

Instruction Group	Mnemonic	Operand	Number of Bytes	Number of Machine Cycles	Operation	Addressing Area	Skip Condition
Special	GETI Notes 1, 2	taddr	1	3	• μPD753104 • When TBR instruction $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr + 1)$	*10	
				4	• When TCALL instruction $(SP - 6)$ $(SP - 3)$ $(SP - 4) \leftarrow PC_{11-0}$ $(SP - 5) \leftarrow 0, 0, 0, 0$ $(SP - 2) \leftarrow \times, \times$, MBE, RBE $PC_{11-0} \leftarrow (taddr)_{3-0} + (taddr + 1)$ $SP \leftarrow SP - 6$		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr + 1) instruction is executed.		Depending on the reference instruction
					• μPD753106, 753108 • When TBR instruction $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$		
				4	• When TCALL instruction $(SP - 6)$ $(SP - 3)$ $(SP - 4) \leftarrow PC_{11-0}$ $(SP - 5) \leftarrow 0, 0, 0, PC_{12}$ $(SP - 2) \leftarrow \times, \times$, MBE, RBE $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$ $SP \leftarrow SP - 6$		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr + 1) instruction is executed.		Depending on the reference instruction
					• μPD75P3116 • When TBR instruction $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr + 1)$		
				4	• When TCALL instruction $(SP - 6)$ $(SP - 3)$ $(SP - 4) \leftarrow PC_{11-0}$ $(SP - 5) \leftarrow$ MBE, RBE, $PC_{13, 12}$ $(SP - 2) \leftarrow \times, \times$, MBE, RBE $PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr + 1)$ $SP \leftarrow SP - 6$		
				3	• When instruction other than TBR and TCALL instructions (taddr) (taddr + 1) instruction is executed.		Depending on the reference instruction

Notes 1. The TBR and TCALL instructions are the table definition assembler pseudo-instructions of the GETI instruction.

2. The above operations in the shaded boxes can be performed only in the Mk II mode. The other operations can be performed only in the Mk I mode.

11.3 Op Code of Each Instruction

(1) Description of symbol of op code

R_2 R_1 R_0	reg
0 0 0	A
0 0 1	X
0 1 0	L
0 1 1	H
1 0 0	E
1 0 1	D
1 1 0	C
1 1 1	B

reg

reg1

P_2 P_1 P_0	reg-pair
0 0 0	XA
0 0 1	XA’
0 1 0	HL
0 1 1	HL’
1 0 0	DE
1 0 1	DE’
1 1 0	BC
1 1 1	BC’

rp’

rp’1

Q_2 Q_1 Q_0	addressing
0 0 0	@HL
0 1 0	@HL+
0 1 1	@HL–
1 0 0	@DE
1 0 1	@DL

@rpa

@rpa1

P_2 P_1	reg-pair
0 0	XA
0 1	HL
1 0	DE
1 1	BC

rp2

rp1

rp

N_5 N_2 N_1 N_0	IE×××
0 0 0 0	IEBT
0 0 1 0	IEW
0 1 0 0	IET0
0 1 0 1	IECSI
0 1 1 0	IE0
0 1 1 1	IE2
1 0 0 0	IE4
1 1 0 0	IET1
1 1 0 1	IET2
1 1 1 0	IE1

I_n : immediate data for n4 or n8
D_n : immediate data for mem
B_n : immediate data for bit
N_n : immediate data for n or IE×××
T_n : immediate data for taddr × 1/2
A_n : immediate data for [relative address distance from branch destination address (2 – 16)] – 1
S_n : immediate data for 1’s complement of [relative address distance from branch destination address (15 – 1)]

(2) Op code for bit manipulation addressing

∗1 in the operand field indicates the following three types:

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

The second byte ∗2 of the op code corresponding to the above addressing is as follows:

∗1	2nd Byte of Op Code								Accessible Bit
fmem. bit	1	0	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FB0H to FBFH that can be manipulated
	1	1	B_1	B_0	F_3	F_2	F_1	F_0	Bit of FF0H to FFFH that can be manipulated
pmem. @L	0	1	0	0	G_3	G_2	G_1	G_0	Bit of FC0H to FFFH that can be manipulated
@H+mem. bit	0	0	B_1	B_0	D_3	D_2	D_1	D_0	Bit of accessible memory bank that can be manipulated

B_n : immediate data for bit

F_n : immediate data for fmem (indicates low-order 4 bits of address)

G_n : immediate data for pmem (indicates bits 5 to 2 of address)

D_n : immediate data for mem (indicates low-order 4 bits of address)

Instruction	Mnemonic	Operand	Op Code		
			B_1	B_2	B_3
Transfer	MOV	A, #n4	0 1 1 1 I_3 I_2 I_1 I_0		
		reg1, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 1 R_2 R_1 R_0	
		rp, #n8	1 0 0 0 1 P_2 P_1 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @rpa1	1 1 1 0 0 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 0	
		@HL, A	1 1 1 0 1 0 0 0		
		@HL, XA	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 0	
		A, mem	1 0 1 0 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		mem, A	1 0 0 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		mem, XA	1 0 0 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg	1 0 0 1 1 0 0 1	0 1 1 1 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 1 1 P_2 P_1 P_0	
		reg1, A	1 0 0 1 1 0 0 1	0 1 1 1 0 R_2 R_1 R_0	
		rp'1, XA	1 0 1 0 1 0 1 0	0 1 0 1 0 P_2 P_1 P_0	
	XCH	A, @rpa1	1 1 1 0 1 Q_2 Q_1 Q_0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 0 0 0 1	
		A, mem	1 0 1 1 0 0 1 1	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
		XA, mem	1 0 1 1 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 0	
		A, reg1	1 1 0 1 1 R_2 R_1 R_0		
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 0 P_2 P_1 P_0	
Table 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	Mnemonic	Operand	Op Code		
			B_1	B_2	B_3
Operation	ADDS	A, #n4	0 1 1 0 I_3 I_2 I_1 I_0		
		XA, #n8	1 0 1 1 1 0 0 1	I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	
		A, @HL	1 1 0 1 0 0 1 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 0 0 P_2 P_1 P_0	
	ADDC	A, @HL	1 0 1 0 1 0 0 1		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 0 1 0 P_2 P_1 P_0	
	SUBS	A, @HL	1 0 1 0 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 0 0 P_2 P_1 P_0	
	SUBC	A, @HL	1 0 1 1 1 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 1 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 1 1 1 0 P_2 P_1 P_0	
	AND	A, #n4	1 0 0 1 1 0 0 1	0 0 1 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 0 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 0 1 0 P_2 P_1 P_0	
	OR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 0 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 0 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 0 0 P_2 P_1 P_0	
	XOR	A, #n4	1 0 0 1 1 0 0 1	0 1 0 1 I_3 I_2 I_1 I_0	
		A, @HL	1 0 1 1 0 0 0 0		
		XA, rp'	1 0 1 0 1 0 1 0	1 0 1 1 1 P_2 P_1 P_0	
		rp'1, XA	1 0 1 0 1 0 1 0	1 0 1 1 0 P_2 P_1 P_0	
Accumulator manipula-tion	RORC	A	1 0 0 1 1 0 0 0		
	NOT	A	1 0 0 1 1 0 0 1	0 1 0 1 1 1 1 1	

Instruction	Mnemonic	Operand	Op Code		
			B_1	B_2	B_3
Increment/ decrement	INCS	reg	1 1 0 0 0 R_2 R_1 R_0		
		rp1	1 0 0 0 1 P_2 P_1 0		
		@HL	1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0	
		mem	1 0 0 0 0 0 1 0	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
	DECS	reg	1 1 0 0 1 R_2 R_1 R_0		
		rp'	1 0 1 0 1 0 1 0	0 1 1 0 1 P_2 P_1 P0	
Comparison	SKE	reg, #n4	1 0 0 1 1 0 1 0	I_3 I_2 I_1 I_0 0 R_2 R_1 R_0	
		@HL, #n4	1 0 0 1 1 0 0 1	0 1 1 0 I_3 I_2 I_1 I_0	
		A, @HL	1 0 0 0 0 0 0 0		
		XA, @HL	1 0 1 0 1 0 1 0	0 0 0 1 1 0 0 1	
		A, reg	1 0 0 1 1 0 0 1	0 0 0 0 1 R_2 R_1 R_0	
		XA, rp'	1 0 1 0 1 0 1 0	0 1 0 0 1 P_2 P_1 P_0	
Carry flag manipulation	SET1	CY	1 1 1 0 0 1 1 1		
	CLR1	CY	1 1 1 0 0 1 1 0		
	SKT	CY	1 1 0 1 0 1 1 1		
	NOT1	CY	1 1 0 1 0 1 1 0		
Memory bit 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	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	
I/O	IN	A, PORTn	1 0 1 0 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
		XA, PORTn	1 0 1 0 0 0 1 0	1 1 1 1 N_3 N_2 N_1 N_0	
	OUT	PORTn, A	1 0 0 1 0 0 1 1	1 1 1 1 N_3 N_2 N_1 N_0	
		PORTn, XA	1 0 0 1 0 0 1 0	1 1 1 1 N_3 N_2 N_1 N_0	
Interrupt control	EI		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 1	1 0 N_5 1 1 N_2 N_1 N_0	
	DI		1 0 0 1 1 1 0 0	1 0 1 1 0 0 1 0	
		IE×××	1 0 0 1 1 1 0 0	1 0 N_5 1 1 N_2 N_1 N_0	
CPU control	HALT		1 0 0 1 1 1 0 1	1 0 1 0 0 0 1 1	
	STOP		1 0 0 1 1 1 0 1	1 0 1 1 0 0 1 1	
	NOP		0 1 1 0 0 0 0 0		
Special	SEL	RBn	1 0 0 1 1 0 0 1	0 0 1 0 0 0 N_1 N_0	
		MBn	1 0 0 1 1 0 0 1	0 0 0 1 N_3 N_2 N_1 N_0	
	GETI	taddr	0 0 T_5 T_4 T_3 T_2 T_1 T_0		

11.4 Instruction Function and Application

This section describes the functions and applications of the respective instructions. The instructions that can be used and the functions of the instructions differ between the Mk I and Mk II modes of the μPD753104, 753106, 753108, and 75P3116. 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:

- μPD753104
- μPD753106
- μPD753108
- μPD75P3116

} In Mk I and Mk II modes

(I) : This instruction can be used only in the Mk I mode of the μPD753104, 753106, 753108, and 75P3116.

(II) : This instruction can be used only in the Mk II mode of the μPD753104, 753106, 753108, and 75P3116.

(I/II) : This instruction can be used commonly in the Mk I and Mk II modes of the μPD753104, 753106, 753108, and 75P3116, 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].

Remark In this section, it is assumed that the 13-bit program counter of the μPD753106 and μPD753108 is used. Note that the program counter of the μPD753104 is 12 bits wide, and that of the μPD75P3116 is 14 bits wide.

11.4.1 Transfer instructions

MOV A, #n4

Function: A ← n4 n4 = I_{3-0}: 0-FH

Transfers 4-bit immediate data n4 to the A register (4-bit accumulator). This instruction has a string effect (group A). Therefore, if this instruction is followed by another MOV A, #n4 or MOV XA, #n8, the following instruction will be processed as a NOP instruction.

Application example

(1) To set 0BH to the accumulator

```
MOV A, #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, and if two or more of this instruction are executed in succession or if this instruction is followed by the MOV A, #n4 instruction, the instruction following this instruction is treated as NOP.

MOV HL, #n8

Function: HL ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair HL. This instruction has a string effect. If two or more of this instructions are executed in succession, those that follow the first instruction are treated as NOP.

MOV rp2, #n8

Function: rp2 ← n8 n8 = I_{7-0}: 00H-FFH

Transfers 8-bit immediate data n8 to register pair rp2 (BC, DE).

MOV A, @HL

MOV A, @HL+

MOV A, @HL–

MOV A, @rpa1

Function: A ← (Register pair specified with an 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 a 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 an 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 an 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 with an 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 a 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 ↔ (3×)
        XCH   A, @DL     ; A ↔ (2×)
        XCH   A, @HL+    ; A ↔ (3×)
        BR    LOOP
```

XCH XA, @HL

Function: A ↔ (HL), X ↔ (HL + 1)

Exchanges the contents of the A register with the contents of the data memory addressed by register pair HL, and the contents of the X register with the contents of the next address.

If the contents of the L register are odd numbers, however, an address whose least significant bit is ignored is specified.

XCH A, mem

Function: A ↔ (mem) mem = D_{7-0}: 00H-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').

11.4.2 Table reference instructions

MOVT XA, @PCDE

Function: μPD753106 and μPD753108 XA ← ROM (PC_{12-8} + DE)

Transfers the low-order 4 bits of the table data in the program memory addressed to the A register when the low-order 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair DE, and the high-order 4 bits to the X register.

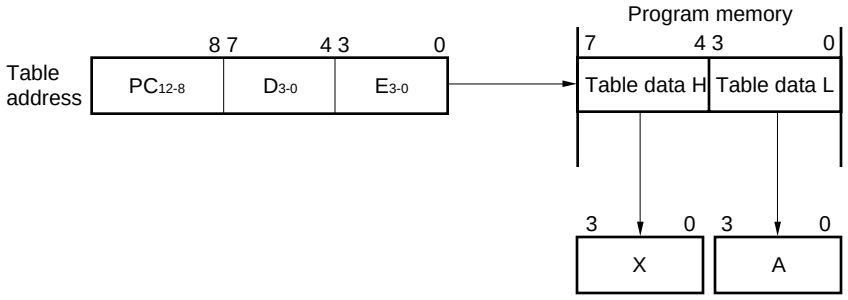
The table address is determined by the contents of the program counter (PC) when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction).

The program counter is not affected by execution of this instruction.

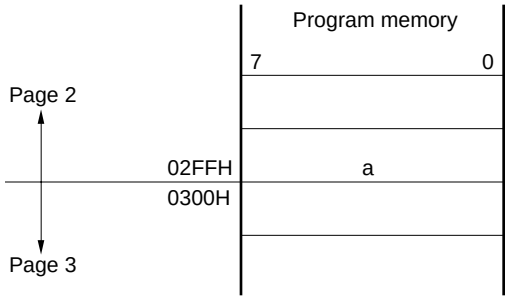
This instruction is useful for successively referencing table data.

Example In the case of μPD753106 or 753108



Remark The function described here applies to the μPD753106 and 753108 that has a 13-bit program counter. Note that the program counter of the μPD753104 is 12 bits wide and that of the μPD75P3116 is 14 bits wide.

Caution The MOVT XA, @PCDE instruction usually references the table data in page where the instruction exists. If the instruction is at address ××FFH, however, the table data in the page where the instruction exists is not referenced, but the table data in the next page is referenced.



For example, if the MOV XA, @PCDE instruction is located at position a in the above figure, the table data in page 3, not page 2, specified by the contents of register pair DE is transferred to register pair XA.

Application example

To transfer the 16-byte data at program memory addresses ××F0H through ××FFH to data memory addresses 30H through 4FH

```
SUB:   SEL    MB0
       MOV    HL, #30H      ; HL ← 30H
       MOV    DE, #0F0H     ; DE ← F0H
LOOP:  MOVT   XA, @PCDE     ; XA ← table data
       MOV    @HL, XA       ; (HL) ← XA
       INCS   HL            ; HL ← HL + 2
       INCS   HL
       INCS   E             ; E ← E + 1
       BR     LOOP
       RET
       ORG    ××F0H
       DB     ××H, ××H, ···  ; table data
```

MOVT XA, @PCXA

Function: μPD753106 and μPD753108 XA ← ROM (PC_{12-8} + XA)

Transfers the low-order 4 bits of the table data in the program memory addressed to the A register when the low-order 8 bits (PC_{7-0}) of the program counter (PC) are replaced with the contents of register pair XA, and the high-order 4 bits to the X register.

The table address is determined by the contents of the PC when this instruction is executed.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction). The PC is not affected by execution of this instruction.

Caution If an instruction exists at address ××FFH, the table data of the next page is transferred, in the same manner as MOVT XA, @PCDE.

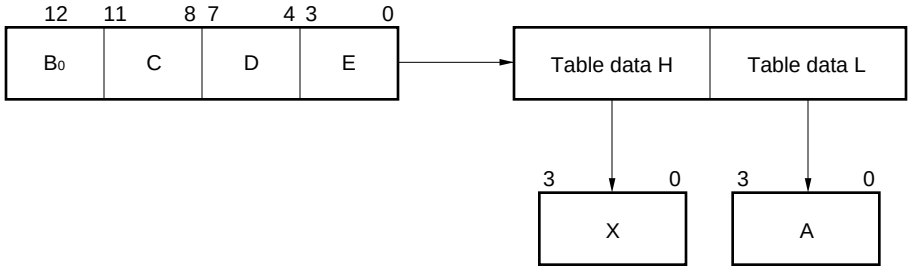
Remark The function described here applies to the μPD753106 and 753108 that has a 13-bit program counter. Note that the program counter of the μPD753104 is 12 bits wide and that of the μPD75P3116 is 14 bits wide.

MOVT XA, @BCDE

Function: μPD75106 and 753108 XA ← ROM (B_0 + CDE)

Transfers the low-order 4 bits of the table data (8-bit) in the program memory addressed by the least significant bit of register B and the contents of registers C, D, and E, to the A register, and the high-order 4 bits to the X register.

The necessary data must be programmed to the table area in advance by using an assembler pseudo-instruction (DB instruction). The PC is not affected by execution of this instruction.

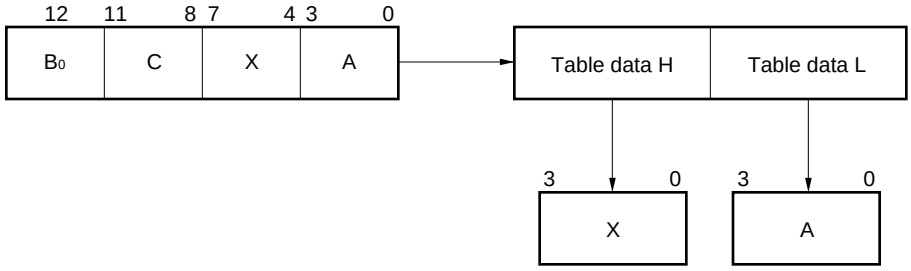


MOVT XA, @BCXA

Function: μPD753106 and 753108 XA ← ROM (B_0 + CXA)

Transfers the low-order 4 bits of the table data (8-bit) in the program memory addressed by the least significant bit of register B and the contents of registers C, X, and A, to the A register, and the high-order 4 bits to the X register.

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.



Remark The function described here applies to the μPD753106 and 753108 that has a 13-bit program counter. Note that the program counter of the μPD753104 is 12 bits wide and that of the μPD75P3116 is 14 bits wide.

11.4.3 Bit transfer instructions

MOV1 CY, fmem. bit

MOV1 CY, pmem. @L

MOV1 CY, @H+mem. bit

Function: CY ← (bit specified by operand)

Transfers the contents of the data memory addressed in the bit manipulating addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) to the carry flag (CY).

MOV1 fmem. bit, CY

MOV1 pmem. @L, CY

MOV1 @H+mem. bit, CY

Function: (bit specified by operand) ← CY

Transfers the contents of the carry flag (CY) to the data memory bit addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit).

Application example

To output the flag of bit 3 at data memory address 3FH to the bit 2 of port 3

```
FLAG   EQU 3FH.3
SEL    MB0
MOV    H, #FLAG SHR 6 ; H ← high-order 4 bits of FLAG
MOV1   CY, @H+FLAG    ; CY ← FLAG
MOV1   PORT3. 2, CY   ; P32 ← CY
```

11.4.4 Operation instructions

ADDS A, #n4

Function: A ← A + n4; Skip if carry. n4 = I_{3-0}: 0-FH

Adds 4-bit immediate data n4 to the contents of the A register. 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 section **11.1.4 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 section **11.1.4 Base number adjustment instruction**).

ADDC XA, rp'

Function: XA, CY ← XA + rp' + CY

Adds the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') to the contents of register pair XA, including the carry. If a carry occurs as a result, the carry flag is set; if not, the carry flag is reset.

ADDC rp'1, XA

Function: rp'1, CY ← rp'1 + XA + CY

Adds the contents of register pair XA to the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), including the carry flag. If a carry occurs as a result, the carry flag is set; if not, the carry flag is reset.

SUBS A, @HL

Function: A ← A – (HL); Skip if borrow

Subtracts the contents of the data memory addressed by register pair HL from the contents of the A register, and sets the result to the A register. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

SUBS XA, rp'

Function: XA ← XA – rp'; Skip if borrow

Subtracts the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') from the contents of register pair XA, and sets the result to register pair XA. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

Application example

To compare specified data memory contents with the contents of a register pair

```
MOV     XA, mem
SUBS    XA, rp'
                ; (mem) ≥ rp'
                ; (mem) < rp'
```

SUBS rp'1, XA

Function: rp' ← rp'1 + XA; Skip if borrow

Subtracts the contents of register pair XA from register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to specified register pair rp'1. If a borrow occurs as a result, the next instruction is skipped.

The carry flag is not affected.

SUBC A, @HL

Function: A, CY ← A – (HL) – CY

Subtracts the contents of the data memory addressed by register pair HL to the contents from the A register, including the carry flag, and sets the result to the A register. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

If an ADDS A, #n4 instruction is placed following this instruction, and if a borrow does not occur as a result of executing this instruction, the ADDS A, #n4 instruction is skipped. If a borrow occurs, the ADDS A, #n4 instruction is executed, and a function that disables the skip function of the ADDS A, #n4 instruction is effected. Therefore, these instructions can be used in combination for base number adjustment (refer to section **11.1.4 Base number adjustment instruction**).

SUBC XA, rp'

Function: XA, CY ← XA – rp' – CY

Subtracts the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') from the contents of register pair XA, including the carry, and sets the result to register pair XA. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

SUBC rp'1, XA

Function: rp'1, CY ← rp'1 – XA – CY

Subtracts the contents of register pair XA from the contents of register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), including the carry flag, and sets the result to specified register pair rp'1. If a borrow occurs as a result, the carry flag is set; if not, the carry flag is reset.

AND A, #n4

Function: A ← A ∧ n4 n4 = I_{3-0}: 0-FH

ANDs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To clear the high-order 2 bits of the accumulator to 0

```
AND A, #0011B
```

AND A, @HL

Function: A ← A ∧ (HL)

ANDs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

AND XA, rp'

Function: XA ← XA ∧ rp'

ANDs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

AND rp'1, XA

Function: rp' 1 ← rp'1 ∧ XA

ANDs the contents of register pair XA with register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair.

OR A, #n4

Function: A ← A ∨ n4 n4 = I_{3-0}: 0-FH

ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To set the low-order 3 bits of the accumulator to 1

```
OR A, #0111B
```

OR A, @HL

Function: A ← A ∨ (HL)

ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

OR XA, rp'

Function: XA ← XA ∨ rp'

ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

OR rp'1, XA

Function: rp' 1 ← rp'1 ∨ XA

ORs the contents of register pair XA with register pair rp'1 (HL, DE, BC, XA', HL', DE', or BC'), and sets the result to the specified register pair rp'1.

XOR A, #n4

Function: A ← A ⊻ n4 n4 = I_{3-0}: 0-FH

Exclusive-ORs 4-bit immediate data n4 with the contents of the A register, and sets the result to the A register.

Application example

To invert the high-order 4 bits of the accumulator

```
XOR A, #1000B
```

XOR A, @HL

Function: A ← A ⊻ (HL)

Exclusive-ORs the contents of the data memory addressed by register pair HL with the contents of the A register, and sets the result to the A register.

XOR XA, rp'

Function: XA ← XA ⊻ rp'

Exclusive-ORs the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC') with the contents of register pair XA, and sets the result to register pair XA.

XOR rp’1, XA

Function: rp’ 1 ← rp’1 ⩛ XA

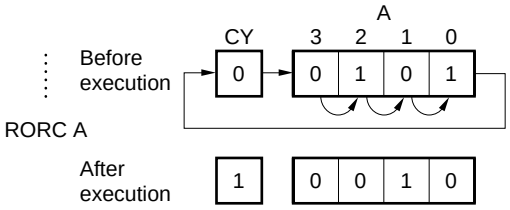
Exclusive-ORs the contents of register pair XA with register pair rp’1 (HL, DE, BC, XA’, HL’, DE’, or BC’), and sets the result to the specified register pair rp’1.

11.4.5 Accumulator manipulation instructions

RORC A

Function: CY ← A_0, A_{n-1} ← A_n, A_3 ← CY (n = 1-3)

Rotates the contents of the A register (4-bit accumulator) 1 bit to the right with the carry flag.



NOT A

Function: A ← $\overline{A}$

Takes 1's complement of the A register (4-bit accumulator) (inverts the bits of the accumulator).

11.4.6 Increment/decrement 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.

11.4.7 Compare instructions

SKE reg, #n4

Function: Skip if reg = n4 n4 = I_{3-0}: 0-FH

Skips the next instruction if the contents of register reg (X, A, H, L, D, E, B, or C) are equal to 4-bit immediate data n4.

SKE @HL, #n4

Function: Skip if (HL) = n4 n4 = I_{3-0}: 0-FH

Skips the next instruction if the contents of the data memory addressed by register pair HL are equal to 4-bit immediate data n4.

SKE A, @HL

Function: Skip if A = (HL)

Skips the next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL.

SKE XA, @HL

Function: Skip if A = (HL) and X = (HL + 1)

Skips the next instruction if the contents of the A register are equal to the contents of the data memory addressed by register pair HL and if the contents of the X register are equal to the contents of the next memory address.

However, if the contents of the L register are an odd number, an address whose least significant bit is ignored is specified.

SKE A, reg

Function: Skip if A = reg

Skips the next one instruction if the contents of the A register are equal to register reg (X, A, H, L, D, E, B, or C).

SKE XA, rp'

Function: Skip if XA = rp'

Skips the next one instruction if the contents of register pair XA are equal to the contents of register pair rp' (XA, HL, DE, BC, XA', HL', DE', or BC').

11.4.8 Carry flag manipulation instructions

SET1 CY

Function: CY ← 1

Sets the carry flag.

CLR1 CY

Function: CY ← 0

Clears the carry flag.

SKT CY

Function: Skip if CY = 1

Skips the next one instruction if the carry flag is 1.

NOT1 CY

Function: CY ← $\overline{\text{CY}}$

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.

11.4.9 Memory bit manipulation instructions

SET1 mem. bit

Function: (mem. bit) ← 1 mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Sets the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

SET1 fmem. bit

SET1 pmem. @L

SET1 @H+mem. bit

Function: (bit specified by operand) ← 1

Sets the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit).

CLR1 mem. bit

Function: (mem. bit) ← 0 mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Clears the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem.

CLR1 fmem. bit

CLR1 pmem. @L

CLR1 @H+mem. bit

Function: (bit specified by operand) ← 0

Clears the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit).

SKT mem. bit

Function: Skip if (mem. bit) = 1
mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Skips the next instruction if the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem is 1.

SKT fmem. bit

SKT pmem. @L

SKT @H+mem. bit

Function: Skip if (bit specified by operand) = 1

Skips the next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) is 1.

SKF mem. bit

Function: Skip if (mem. bit) = 0
mem = D_{7-0}: 00H-FFH, bit = B_{1-0}: 0-3

Skips the next instruction if the bit specified by 2-bit immediate data bit at the address specified by 8-bit immediate data mem is 0.

SKF fmem. bit

SKF pmem. @L

SKF @H+mem. bit

Function: Skip if (bit specified by operand) = 0

Skips the next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) is 0.

SKTCLR fmem. bit

SKTCLR pmem. @L

SKTCLR @H+mem. bit

Function: Skip if (bit specified by operand) = 1 then clear

Skips the next instruction if the bit of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit) is 1, and then clears the bit to “0”.

AND1 CY, fmem. bit

AND1 CY, pmem. @L

AND1 CY, @H+mem. bit

Function: CY $\leftarrow$ CY $\wedge$ (bit specified by operand)

ANDs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

OR1 CY, fmem. bit

OR1 CY, pmem. @L

OR1 CY, @H+mem. bit

Function: CY $\leftarrow$ CY $\vee$ (bit specified by operand)

ORs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

XOR1 CY, fmem. bit

XOR1 CY, pmem. @L

XOR1 CY, @H+mem. bit

Function: CY ← CY ⩛ (bit specified by operand)

Exclusive-ORs the content of the carry flag with the contents of the data memory addressed in the bit manipulation addressing mode (fmem. bit, pmem. @L, or @H+mem. bit), and sets the result to the carry flag.

11.4.10 Branch instructions

BR addr

Function: μ**PD753108** $PC_{12-0} \leftarrow$ **addr**
addr = 0000H-1FFFH

Branches to an address specified 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.

BR addr1

Function: μ**PD753108** $PC_{12-0} \leftarrow$ **addr1**
addr1 = 0000H-1FFFH

Branches to an address specified 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.

BRA !addr1

Function: μ**PD753108** $PC_{12-0} \leftarrow$ **addr1**

BR !addr

Function: μ**PD753108** $PC_{12-0} \leftarrow$ **addr**
addr = 0000H-1FFFH

Transfers immediate data addr to the program counter (PC) and branches to the address specified by the PC.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

BR $addr

Function: μPD753108 $PC_{12-0} \leftarrow$ 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.

BR $addr1

Function: μPD753108 $PC_{12-0} \leftarrow$ 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.

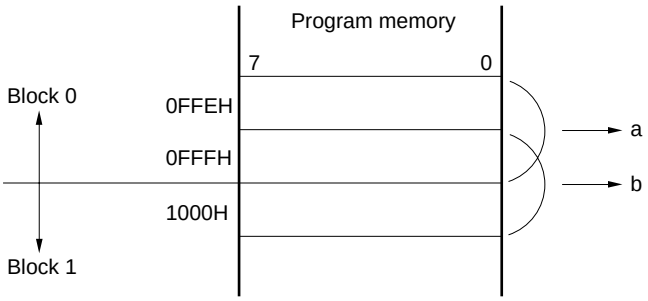
Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

BRCB !caddr

Function: μPD753108 $PC_{12-0} \leftarrow PC_{12} + caddr_{11-0}$
caddr = n000H-nFFFH
n = PC_{12} = 0, 1

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 or b in the figure above, execution branches to block 1, not block 0.

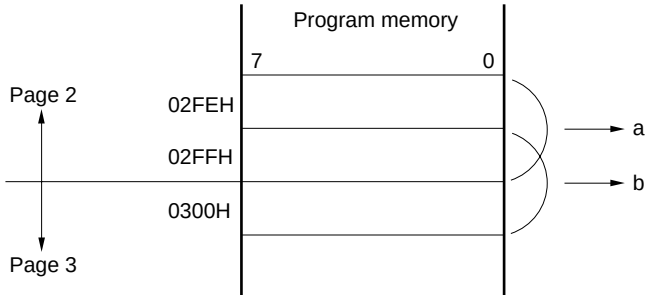
Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

BR PCDE

Function: μPD753108 $PC_{12-0} \leftarrow PC_{12-8} + DE$
$PC_{7-4} \leftarrow D$, $PC_{3-0} \leftarrow E$

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 ××FE or ××FFH, however, execution does not branch in that page, but to the next page.



For example, if the BR PCDE instruction is at position a or b in the above figure, execution branches to the low-order 8-bit address specified by the contents of register pair DE in page 3, not in page 2.

BR PCXA

Function: μPD753108 $PC_{12-0} \leftarrow PC_{12-8} + XA$
$PC_{7-4} \leftarrow X$, $PC_{3-0} \leftarrow A$

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.

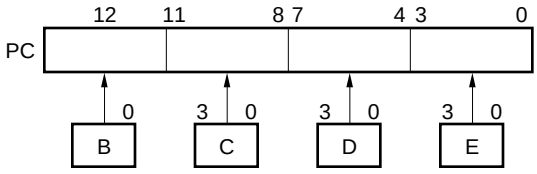
Caution This instruction branches execution to the next page, not to the same page, if the first byte of the op code is at address ××FEH or ××FFH, in the same manner as the BR PCDE instruction.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

BR BCDE

Function: μPD753108 $PC_{12-0} \leftarrow B_0 + CDE$

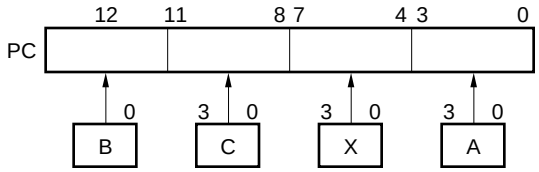
Branches to the address specified by the contents of the program counter replaced with the contents of registers B_0, C, D, and E.



BR BCXA

Function: μPD753108 $PC_{12-0} \leftarrow B_0 + CXA$

Branches to the address specified by the contents of the program counter replaced with the contents of registers B_0, C, X, and A.



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**.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

11.4.11 Subroutine stack control instructions

(II) CALLA !addr1

Function: μ**PD753108**

(SP – 2) ← ×, ×, MBE, RBE, (SP – 3) ← PC$_{7-4}$
(SP – 4) ← PC$_{3-0}$, (SP – 5) ← 0, 0, 0, PC$_{12}$
(SP – 6) ← PC$_{11-8}$
PC$_{12-0}$ ← addr1, SP ← SP – 6

(I/II) CALL !addr

Function: μ**PD753108**

[Mk I mode]
(SP – 1) ← PC$_{7-4}$, (SP – 2) ← PC$_{3-0}$
(SP – 3) ← MBE, RBE, 0, PC$_{12}$
(SP – 4) ← PC$_{11-8}$, PC$_{12-0}$ ← addr, SP ← SP – 4

addr = 0000H-1FFFH

[Mk II mode]
(SP – 2) ← ×, ×, MBE, RBE
(SP – 3) ← PC$_{7-4}$, (SP – 4) ← PC$_{3-0}$
(SP – 5) ← 0, 0, 0, PC$_{12}$, (SP – 6) ← PC$_{11-8}$
PC$_{12-0}$ ← addr, SP ← SP – 6

addr = 0000H-1FFFH

Saves the contents of the program counter (return address), MBE, and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by 14-bit immediate data addr.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

(I/II) CALLF !faddr

Function: **μPD753108**

[Mk I mode]

(SP – 1) ← PC_{7-4}, (SP – 2) ← PC_{3-0}

(SP – 3) ← MBE, RBE, 0, PC_{12}

(SP – 4) ← PC_{11-8}, SP ← SP – 4

PC_{12-0} ← 00 + faddr

faddr = 0000H-07FFH

[Mk II mode]

(SP – 2) ← ×, ×, MBE, RBE

(SP – 3) ← PC_{7-4}, (SP – 4) ← PC_{3-0}

(SP – 5) ← 0, 0, 0, PC_{12}, (SP – 6) ← PC_{11-8}

SP ← SP – 6

PC12–0 ← 00 + faddr

faddr = 0000H-07FFH

Saves the contents of the program counter (return address), MBE, and RBE to the data memory (stack) addressed by the stack pointer (SP), decrements the SP, and then branches to the address specified by 11-bit immediate data faddr. The address range from which a subroutine can be called is limited to 0000H to 07FFH (0 to 2047).

TCALL !addr

Function

This is an assembler 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**.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

RET (I/II)

Function: μPD753108

[Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, RBE, 0, PC_{12} \leftarrow (SP + 1)$
$PC_{3-0} \leftarrow (SP + 2), PC_{7-4} \leftarrow (SP + 3), SP \leftarrow SP + 4$

[Mk II mode] $PC_{11-8} \leftarrow (SP), MBE, 0, 0, PC_{12} \leftarrow (SP + 1)$
$PC_{3-0} \leftarrow (SP + 2), PC_{7-4} \leftarrow (SP + 3)$
$\times, \times$, MBE, RBE $\leftarrow (SP + 4)$
$SP \leftarrow SP + 6$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), and then increments the contents of the SP.

Caution None of the flags of the program status word (PSW), except MBE and RBE, are restored.

RETS (I/II)

Function: μPD753108

[Mk I mode] $PC_{11-8} \leftarrow (SP), MBE, 0, 0, PC_{12} \leftarrow (SP + 1)$
$PC_{3-0} \leftarrow (SP + 2), PC_{7-4} \leftarrow (SP + 3), SP \leftarrow SP + 4$
Then skip unconditionally

[Mk II mode] $PC_{11-8} \leftarrow (SP), 0, 0, 0, PC_{12} \leftarrow (SP + 1)$
$PC_{3-0} \leftarrow (SP + 2), PC_{7-4} \leftarrow (SP + 3)$
$\times, \times$, MBE, RBE $\leftarrow (SP + 4), SP \leftarrow SP + 6$
Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), memory bank enable flag (MBE), and register bank enable flag (RBE), increments the contents of the SP, and then skips unconditionally.

Caution None of the flags of the program status word (PSW), except MBE and RBE, are restored.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

I/II RETI

Function: μPD753108

[Mk I mode] **$PC_{11-8} \leftarrow (SP)$, MBE, RBE, 0, $PC_{12} \leftarrow (SP + 1)$**
$PC_{3-0} \leftarrow (SP + 2)$, $PC_{7-4} \leftarrow (SP + 3)$
$PSW_L \leftarrow (SP + 4)$, $PSW_H \leftarrow (SP + 5)$
$SP \leftarrow SP + 6$

[Mk II mode] **$PC_{11-8} \leftarrow (SP)$, 0, 0, 0, $PC_{12} \leftarrow (SP + 1)$**
$PC_{3-0} \leftarrow (SP + 2)$, $PC_{7-4} \leftarrow (SP + 3)$
$PSW_L \leftarrow (SP + 4)$, $PSW_H \leftarrow (SP + 5)$
$SP \leftarrow SP + 6$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) 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.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

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 4 bits of the register pair (rp_H: X, H, D, or B) are saved to the stack addressed by (SP – 1), and the low-order 4 bits (rp_L: A, L, E, or C) are saved to the stack addressed by (SP – 2).

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 addressed by the stack pointer (SP) to register pair rp (XA, HL, DE, or BC), and then increments the contents of the stack pointer.

The contents of (SP) are restored to the low-order 4 bits of the register pair (rp_L: A, L, E, or C), and the contents of (SP + 1) are restored to the high-order 4 bits (rp_H: X, H, D, or B).

POP BS

Function: RBS ← (SP), MBS ← (SP + 1), SP ← SP + 2

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the register bank selection register (RBS) and memory bank selection register (MBS), and then increments the contents of the SP.

11.4.12 Interrupt control instructions

EI

Function: IME (IPS. 3) ← 1

Sets the interrupt mask enable flag (bit 3 of the interrupt priority selection register) to “1” to enable interrupts. Acknowledging an interrupt is controlled by an interrupt enable flag corresponding to the interrupt.

EI IE×××

Function: IE××× ← 1 ××× = N_5, $N_{2\text{-}0}$

Sets a specified interrupt enable flag (IE×××) to “1” to enable acknowledging the corresponding interrupt (××× = BT, CSI, T0, T, T2, W, 0, 1, 2, or 4).

DI

Function: IME (IPS. 3) ← 0

Resets the interrupt mask 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 IE×××

Function: IE××× ← 0 ××× = N_5, $N_{2\text{-}0}$

Resets a specified interrupt enable flag (IE×××) to “0” to disable acknowledging the corresponding interrupt (××× = BT, CSI, T0, T1, T2, W, 0, 1, 2, or 4).

11.4.13 Input/output instructions

IN A, PORTn

Function: A ← PORTn n = N_{3-0}: 0-3, 5, 6, 8, or 9

Transfers the contents of a port specified by PORTn (n = 0-3, 5, 6, 8, or 9) to the A register.

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 0 to 3, 5, 6, 8 or 9 can be specified to n.
The data of the output latch is loaded to the A register in the output mode, and the data of the port pins are loaded to the register in the input mode by specifying input/output mode.

IN XA, PORTn

Function: A ← PORTn, X ← PORTn+1 n = N_{3-0}: 8

Transfers the contents of the port specified by PORTn (n = 8) to the A register, and transfers the contents of the next port to the X register.

Caution Only 8 can be specified as n. When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15).
The data of the output latch is loaded to the A and X registers in the output mode, and the data of the port pins are loaded to the registers in the input mode by specifying input/output mode.

OUT PORTn, A

Function: PORTn ← A n = N_{3-0}: 2, 3, 5, 6, 8, or 9

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 2, 3, 5, 6, 8, or 9).

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 2, 3, 5, 6, 8, or 9 can be specified as n.

OUT PORTn, XA

Function: PORTn ← A, PORTn+1 ← X n = N_{3-0}: 8

Transfers the contents of the A register to the output latch of a port specified by PORTn (n = 8), and the contents of the X register to the output latch of the next port.

Caution When this instruction is executed, it is necessary to set MBE = 0 or (MBE = 1, MBS = 15). Only 8 can be specified as n.

11.4.14 CPU control instructions

HALT

Function: PCC. 2 ← 1

Sets the HALT mode (this instruction sets bit 2 of the processor clock control register).

Caution Make sure that a NOP instruction follows the HALT instruction.

STOP

Function: PCC. 3 ← 1

Sets the STOP mode (this instruction sets bit 3 of the processor clock control register).

Caution Make sure that a NOP instruction follows the STOP instruction.

NOP

Function: Does nothing but consumes 1 machine cycle.

11.4.15 Special instructions

SEL RBn

Function: RBS ← n n = N_{1-0}: 0-3

Sets 2-bit immediate data n to the register bank 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: μPD753108
taddr = T_{5-0}, 0: 20H-7FH
[Mk I mode]

- **When a table defined by the TBR instruction is referenced**
  PC_{12-0} ← $(taddr)_{4-0}$ + (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, PC_{12}
  (SP – 4) ← PC_{11-8}
  PC_{12-0} ← $(taddr)_{4-0}$ + (taddr + 1)
  SP ← SP – 4
- **When a table defined by an instruction other than TBR or TCALL is referenced**
  Executes instruction with (taddr) (taddr + 1) as op code

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

[Mk II mode]

- **When a table defined by the TBR instruction is referenced[Note]**
  $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$
- **When a table defined by the TCALL instruction is referenced[Note]**
  (SP – 2) ← ×, ×, MBE, RBE
  $(SP - 3) \leftarrow PC_{7-4}$, $(SP - 4) \leftarrow PC_{3-0}$
  $(SP - 5) \leftarrow 0, 0, 0, PC_{12}$, $(SP - 6) \leftarrow PC_{11-8}$
  $PC_{12-0} \leftarrow (taddr)_{4-0} + (taddr + 1)$
  SP ← SP – 6
- **When a table defined by an instruction other than TBR or TCALL is referenced**
  Executes instruction with (taddr) (taddr + 1) as op code

Note The address specified by the TBR and TCALL instructions is limited to 0000H to 3FFFH.

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 by taddr.

Remark The function described in this section applies to the μPD753108, which has a 13-bit program counter and addr = 0000H-1FFFH. Interpret the above explanation according to the case where the μPD753104, μPD753106, and μPD75P3116 have the program counters of 12 bits and addr = 000H-FFFH, 13 bits and addr = 0000H-17FFH, and 14 bits and addr = 0000H-3FFFH, respectively.

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 )   Replaced by GETI
( CALL  SUB1     )
( BR    SUB2     )
          ORG     20H
HL00 :    MOV     HL, #00H
XAFF :    MOV     XA, #FFH
CSUB1 :   TCALL   SUB1
BSUB2 :   TBR     SUB2
            :
          GETI    HL00      ; MOV HL, #00H
            :
          GETI    BSUB2     ; BR SUB2
            :
          GETI    CSUB1     ; CALL SUB1
            :
          GETI    XAFF      ; MOV XA, #FFH
```

APPENDIX A μPD75308B, 753108 AND 75P3116 FUNCTIONAL LIST

Parameter		μPD75308B	**μPD753108**	**μPD75P3116**
Program memory		Mask ROM 0000H-1F7FH (8064 × 8 bits)	Mask ROM 0000H-1FFFH (8192 × 8 bits)	One-time PROM 0000H-3FFFH (16384 × 8 bits)
Data memory		000H-1FFH (512 × 4 bits)		
CPU		75X Standard	75XL CPU	
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)	
	When subsystem clock is selected	122 μs (32.768-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	
	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	
	CMOS input/output	16	20	
	Bit port output	8	0	
	N-ch open-drain input/output	8	4	
	Total	40	32	

Parameter		μPD75308B	**μPD753108**	**μPD75P3116**
LCD controller/driver		Segment selection: 24/28/32 (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)	
		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.		No on-chip split resistor for LCD driver
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 a 16-bit timer/event counter, carrier generator, gated timer) • 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)	
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)	
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		
SOS register	Feedback resistor cut flag (SOS.0)	None	Contained	
	Subsystem clock oscillator current cut flag (SOS.1)	None	Contained	
Register bank selection register (RBS)		None	Yes	
Standby release by INT0		No	Yes	
Interrupt priority selection register (IPS)		None	Yes	

Parameter	μPD75308B	**μPD753108**	**μPD75P3116**
Vectored interrupt	External: 3, internal: 3	External: 3, internal: 5	
Supply voltage	V_{DD} = 2.0 to 6.0 V	V_{DD} = 1.8 to 5.5 V	
Operating ambient temperature	T_A = –40 to +85°C		
Package	• 80-pin plastic QFP (14 × 20 mm) • 80-pin plastic QFP (14 × 14 mm) • 80-pin plastic TQFP (Fine pitch) (12 × 12 mm)	• 64-pin plastic QFP (14 × 14 mm) • 64-pin plastic QFP (12 × 12 mm)	

[MEMO]

APPENDIX B DEVELOPMENT TOOLS

The following development tools are provided for system development using the μPD753108.

In 75XL Series, the relocatable assembler which is common to the series is used in combination with the device file of each product.

Language processor

RA75X relocatable assembler	Host Machine	OS	Distribution Media	Part Number (Product Name)
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD	μS5A13RA75X
			5-inch 2HD	μS5A10RA75X
	IBM PC/AT™ and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13RA75X
			5-inch 2HC	μS7B10RA75X

Device file	Host Machine	OS	Distribution Media	Part Number (Product Name)
	PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note])	3.5-inch 2HD	μS5A13DF753108
			5-inch 2HD	μS5A10DF753108
	IBM PC/AT and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13DF753108
			5-inch 2HC	μS7B10DF753108

Note Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operations of the assembler and device file are guaranteed only on the above host machines and OSs.

PROM write tools

<table>
<tr><td>Hardware</td><td>PG-1500</td><td colspan="4">PG-1500 is a PROM programmer which enables you to program single-chip microcontrollers including PROM by stand-alone or host machine operation by connecting an attached board and optional programmer adapter to PG-1500. It also enables you to program typical PROM devices of 256K bits to 4M bits.</td></tr>
<tr><td></td><td>PA-75P3116GC</td><td colspan="4">PROM programmer adapter for the μPD75P3116GC. Connect the programmer adapter to PG-1500 for use.</td></tr>
<tr><td></td><td>PA-75P3116GK</td><td colspan="4">PROM programmer adapter for the μPD75P3116GK. Connect the programmer adapter to PG-1500 for use.</td></tr>
<tr><td>Software</td><td>PG-1500 controller</td><td colspan="4">PG-1500 and a host machine are connected by serial and parallel interface and PG-1500 is controlled on the host machine.</td></tr>
<tr><td></td><td></td><td rowspan="2">Host Machine</td><td colspan="2"></td><td rowspan="2">Part Number (Product Name)</td></tr>
<tr><td></td><td></td><td>OS</td><td>Distribution Media</td></tr>
<tr><td></td><td></td><td>PC-9800 series</td><td>MS-DOS (Ver. 3.30 to Ver. 6.2^{Note})</td><td>3.5-inch 2HD</td><td>μS5A13PG1500</td></tr>
<tr><td></td><td></td><td></td><td></td><td>5-inch 2HD</td><td>μS5A10PG1500</td></tr>
<tr><td></td><td></td><td>IBM PC/AT and compatible machines</td><td>Refer to “OS for IBM PC”</td><td>3.5-inch 2HC</td><td>μS7B13PG1500</td></tr>
<tr><td></td><td></td><td></td><td></td><td>5-inch 2HC</td><td>μS7B10PG1500</td></tr>
</table>

Note Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remark Operation of the PG-1500 controller is guaranteed only on the above host machines and OSs.

Debugging tool

The in-circuit emulators (IE-75000-R and IE-75001-R) are available as the program debugging tool for the μPD753108.

The system configurations are described as follows.

Hardware	IE-75000-R[Note 1]	In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X Series and 75XL Series. When developing a μPD753108 Subseries, the emulation board IE-75300-R-EM and emulation probe that are sold separately must be used with the IE-75000-R. By connecting with the host machine and the PROM programmer, efficient debugging can be made. It contains the emulation board IE-75000-R-EM which is connected.
	IE-75001-R	In-circuit emulator for debugging the hardware and software when developing the application systems that use the 75X Series and 75XL Series. When developing a μPD753108 Subseries, the emulation board IE-75300-R-EM and emulation probe that are sold separately must be used with the IE-75001-R. By connecting with the host machine and the PROM programmer, efficient debugging can be made.
	IE-75300-R-EM	Emulation board for evaluating the application systems that use a μPD753108 Subseries. It must be used with the IE-75000-R or IE-75001-R.
	EP-753108GC-R EV-9200GC-64	Emulation probe for the μPD753108GC. It must be connected to the IE-75000-R (or IE-75001-R) and IE-75300-R-EM. It is supplied with the 64-pin conversion socket EV-9200GC-64 which facilitates connection to a target system.
	EP-753108GK-R ★ TGK-064SBW[Note 2]	Emulation probe for the μPD753108GK. It must be connected to the IE-75000-R (or IE-75001-R) and IE-75300-R-EM. It is supplied with the 64-pin conversion adapter TGK-064SBW which facilitates connection to a target system.
Software	IE control program	Connects the IE-75000-R or IE-75001-R to a host machine via RS-232-C and Centronics interface and controls the above hardware on a host machine.

Host Machine	OS	Distribution Media	Part Number (Product Name)
PC-9800 series	MS-DOS (Ver. 3.30 to Ver. 6.2[Note 3])	3.5-inch 2HD	μS5A13IE75X
		5-inch 2HD	μS5A10IE75X
IBM PC/AT and compatible machines	Refer to "**OS for IBM PC**"	3.5-inch 2HC	μS7B13IE75X
		5-inch 2HC	μS7B10IE75X

Notes 1. Maintenance parts

★ **2.** Made by TOKYO ELETECH CORPORATION (TEL: 03-5295-1661). Consult an NEC sales representative for purchasing.

3. Although MS-DOS ver.5.00 and later versions have the task swap function, the function cannot be used with this software.

Remarks 1. Operation of the IE control program is guaranteed only on the above host machines and OSs.

2. The μPD753104, 753106, 753108, and 75P3116 are collectively called μPD753108 Subseries.

OS for IBM PC

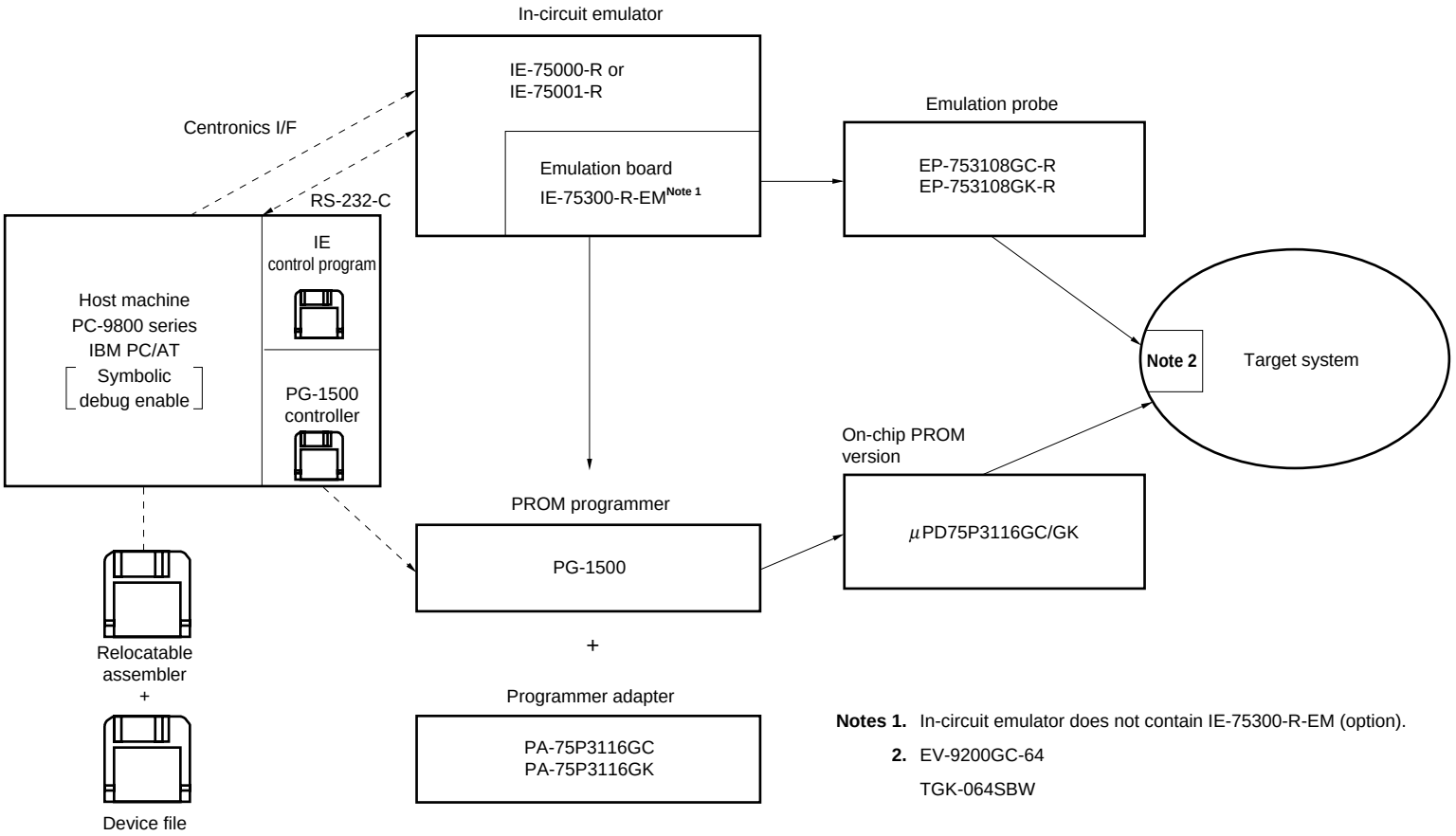
The following IBM PC OSs are supported.

OS	Version
PC DOS	Ver. 3.1 to Ver. 6.3 J6.1/V[Note] to J6.3/V[Note]
MS-DOS	Ver. 5.00 to Ver. 6.22 5.0/V[Note] to 6.2/V[Note]
IBM DOS™	J5.02/V[Note]

Note Only English version is supported.

Caution Ver. 5.0 and later versions have the task swap function, but it cannot be used with this software.

Development Tool Configuration



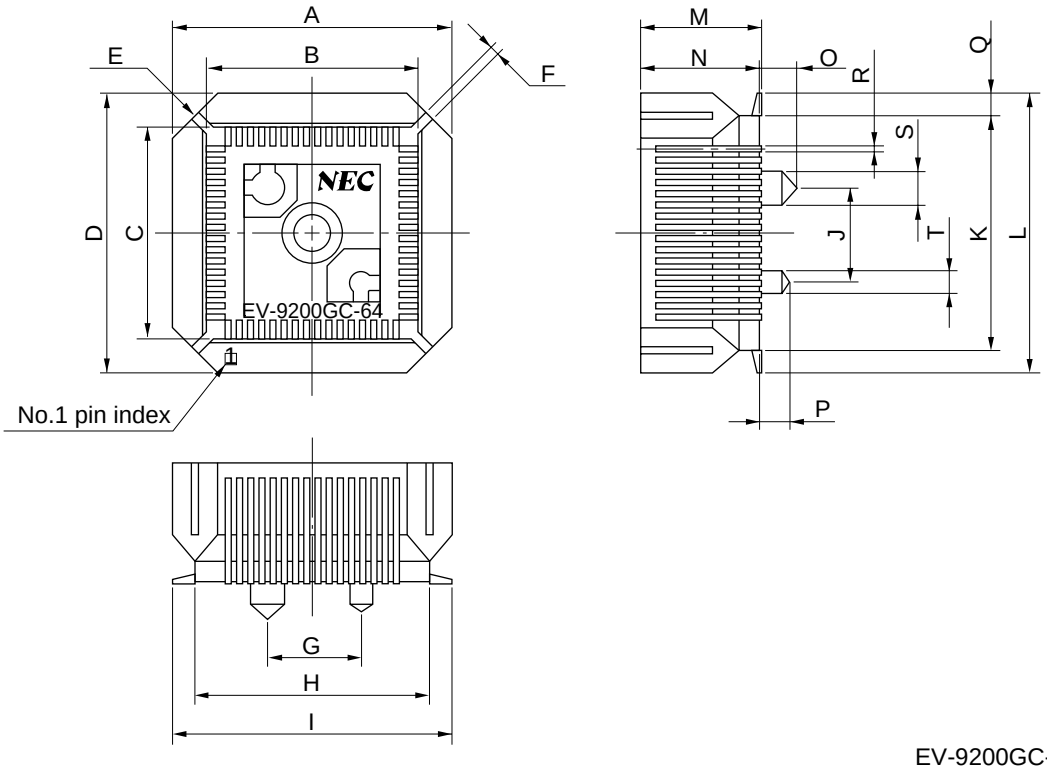
Notes 1. In-circuit emulator does not contain IE-75300-R-EM (option).

2. EV-9200GC-64

TGK-064SBW

Package Drawing and Recommended Footprint of Conversion Socket (EV-9200GC-64)

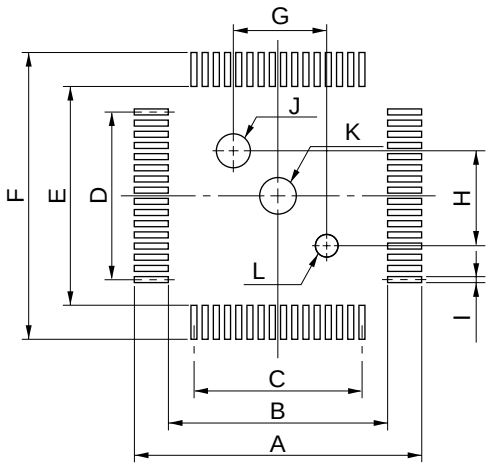
Figure B-1. Package Drawing of EV-9200GC-64 (for reference only)



EV-9200GC-64-G0

ITEM	MILLIMETERS	INCHES
A	18.8	0.74
B	14.1	0.555
C	14.1	0.555
D	18.8	0.74
E	4-C 3.0	4-C 0.118
F	0.8	0.031
G	6.0	0.236
H	15.8	0.622
I	18.5	0.728
J	6.0	0.236
K	15.8	0.622
L	18.5	0.728
M	8.0	0.315
N	7.8	0.307
O	2.5	0.098
P	2.0	0.079
Q	1.35	0.053
R	0.35 ± 0.1	$0.014^{+0.004}_{-0.005}$
S	$\phi 2.3$	$\phi 0.091$
T	$\phi 1.5$	$\phi 0.059$

Figure B-2. Recommended Footprint of EV-9200GC-64 (for reference only)



EV-9200GC-64-P1E

ITEM	MILLIMETERS	INCHES
A	19.5	0.768
B	14.8	0.583
C	$0.8\pm0.02\times15=12.0\pm0.05$	$0.031^{+0.002}_{-0.001}\times0.591=0.472^{+0.003}_{-0.002}$
D	$0.8\pm0.02\times15=12.0\pm0.05$	$0.031^{+0.002}_{-0.001}\times0.591=0.472^{+0.003}_{-0.002}$
E	14.8	0.583
F	19.5	0.768
G	6.00 ± 0.08	$0.236^{+0.004}_{-0.003}$
H	6.00 ± 0.08	$0.236^{+0.004}_{-0.003}$
I	0.5 ± 0.02	$0.197^{+0.001}_{-0.002}$
J	$\phi2.36\pm0.03$	$\phi0.093^{+0.001}_{-0.002}$
K	$\phi2.2\pm0.1$	$\phi0.087^{+0.004}_{-0.005}$
L	$\phi1.57\pm0.03$	$\phi0.062^{+0.001}_{-0.002}$

Caution Dimensions of mount pad for EV-9200 and that for target device (QFP) may be different in some parts. For the recommended mount pad dimensions for QFP, refer to "SEMICONDUCTOR DEVICE MOUNTING TECHNOLOGY MANUAL" (C10535E).

[MEMO]

APPENDIX C ORDERING MASK ROMS

After your program has been developed, place an order for a mask ROM using the following procedure:

<1> Reservation for ordering mask ROM

Inform NEC of your schedule to place an order for the mask ROM (NEC's response may be delayed if it is not informed in advance).

★ **<2> Preparation of ordering media**

The following three media for ordering the mask ROM are available:

- UV-EPROM[Note]
- 3.5-inch IBM-format floppy disk (outside Japan only)
- 5-inch IBM-format floppy disk (outside Japan only)

Note Prepare three UV-EPROMs having the same contents when ordering with UV-EPROM.
For the product with mask option, write down the mask option data on the mask option information sheet.

<3> Preparation of necessary documents

Fill out the following documents when ordering the mask ROM:

A. Mask ROM Ordering Sheet
B. Mask ROM Ordering Check Sheet
C. Mask Option Information Sheet (necessary for product with mask option)

<4> Ordering

Submit the media prepared in <2> and documents prepared in <3> to NEC by the reserved date.

Caution Refer to the information document, ROM Code Ordering Method (C10302J: Japanese-version) for details.

[MEMO]

APPENDIX D INSTRUCTION INDEX

D.1 Instruction Index (by function)

D.2 Instruction Index (alphabetical order)

APPENDIX E HARDWARE INDEX

APPENDIX F REVISION HISTORY

The table shown below lists the major revised points in each edition of this manual. Note that, in the "Including Chapter" column, the chapter numbers and names of each edition are shown, not necessarily this edition.

Edition	Description	Including Chapter
Second	The development status of the μPD753104, 753106, 753108, and 75P3116 is changed from “Under development” to “Development completed.”	Throughout
	The input withstand voltage of the port 5’s pins in the N-ch open-drain mode is changed from 12 V to 13 V.	
	A caution description is added for the S16/P93 to S19/P90 and S20/P83 to S23/P80 pins used as segment signal outputs.	
	An explanation for the feedback resistor mask options is added in paragraph “Subsystem clock oscillator control register (SOS).”	CHAPTER 5 PERIPHERAL HARDWARE FUNCTION
	Values for 6.00-MHz operation are added in the “Resolution and Maximum Allowable Time Setting” tables.	
	Of the LCD drive modes, the lower LCD drive voltage limit in the normal mode is changed from 2.7 V to 2.2 V.	
	The explanation about mask options is added.	CHAPTER 10 MASK OPTION
	Entries in the instruction function description sections are rearranged according to those in the “Instruction Sets and their Operations” section.	CHAPTER 11 INSTRUCTION SET
	The versions of the supported operating systems are upgraded.	APPENDIX B DEVELOPMENT TOOLS
Third	Data bus pins (D0 to D7) are added.	Throughout
	“List of Recommended Connections for Unused Pins” is changed.	CHAPTER 2 PIN FUNCTION
	Caution in “Differences between Mk I Mode and Mk II Mode” is changed.	CHAPTER 4 INTERNAL CPU FUNCTIONS
	Description on feedback resistor mask option in the section of the “Subsystem clock oscillator control register (SOS)” is deleted.	CHAPTER 5 PERIPHERAL HARDWARE FUNCTION
	“Maximum Time Required to Switch System to/from CPU Clocks” is changed.	
	“Switching between System Clock and CPU Clock” is changed.	
	Cautions are added to “Bus release signal (REL)” and “Command signal (CMD).”	
	“Mask Option Selection” is added.	CHAPTER 7 STANDBY FUNCTION
	“Writing Program Memory” is changed.	CHAPTER 9 WRITING AND VERIFYING PROM (PROGRAM MEMORY)
	“Reading Program Memory” is changed.	
	“Mask Option of Feedback Resistor for Subsystem Clock” is deleted.	CHAPTER 10 MASK OPTION
	Ordering media in “Ordering Mask ROMs” are changed.	APPENDIX C ORDERING MASK ROMS

[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 Semiconductor Technical Hotline
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑