NEC

Preliminary User's Manual

μPD789074 Subseries

8-Bit Single-Chip Microcontrollers

μPD789071
μPD789072
μPD789074
μPD78F9076

Document No. U14801EJ1V0UMJ1 (1st edition)
Date Published October 2000 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 these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

License not needed: μPD78F9076

The customer must judge the need for license: μPD789071, 789072, 789074

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

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

M5D 98. 12

Regional Information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NEC do Brasil S.A.
Electron Devices Division
Guarulhos-SP Brasil
Tel: 55-11-6462-6810
Fax: 55-11-6462-6829

J00.7

[MEMO]

INTRODUCTION

Readers

This manual is intended for user engineers to gain an understanding of the functions of the μPD789074 Subseries in order to design and develop its application systems and programs.

Purpose

This manual is intended for users to understand the functions described in the **Organization** below.

Organization

Two manuals are available for the μPD789074 Subseries: this manual and the Instruction Manual (common to the 78K/0S Series).

μPD789074 Subseries User's Manual	78K/0S Series User's Manual Instructions
• Pin functions • Internal block functions • Interrupts • Other internal peripheral functions	• CPU function • Instruction set • Instruction description

How to Read This Manual

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

◊ To understand the overall functions of the μPD789074 Subseries
→ Read this manual in the order of the **CONTENTS**.
◊ How to read register formats
→ The name of a bit whose number is enclosed with <> is reserved in the assembler and is defined in the C compiler by the header file sfrbit.h.
◊ To learn the detailed functions of a register whose register name is known
→ See **APPENDIX C REGISTER INDEX**.
◊ To learn the details of the instruction functions of the 78K/0S Series
→ Refer to **78K/0S Series User's Manual Instructions (U11047E)** separately available.

Conversions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{\times\times\times}$ (Overscore over pin or signal name)
Note:	Footnote for item marked **Note** in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	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 No.	
	English	Japanese
μPD789071, 789072, 789074 Data Sheet	To be prepared	To be prepared
μPD78F9076 Preliminary Product Information	U14708E	U14708J
μPD789074 Subseries User's Manual	This manual	U14801J
78K/0S Series User's Manual Instructions	U11047E	U11047J
78K/0, 78K/0S Series Application Note Flash Memory Write	U14458E	U14458J

Documents related to development tools (user's manuals)

Document Name		Document No.	
		English	Japanese
RA78K0S Assembler Package	Operation	U11622E	U11622J
	Assembly Language	U11599E	U11599J
	Structured Assembly Language	U11623E	U11623J
CC78K/0S C Compiler	Operation	U11816E	U11816J
	Language	U11817E	U11817J
SM78K0S System Simulator Windows™ Based	Reference	U11489E	U11489J
SM78K Series System Simulator	External Parts User Open Interface Specifications	U10092E	U10092J
ID78K0S-NS Integrated Debugger Windows Based	Reference	U12901E	U12901J
IE-78K0S-NS		U13549E	U13549J
IE-789046-NS-EM1		U14433E	U14433J

Documents related to embedded software (user's manuals)

Document Name		Document No.	
		English	Japanese
78K/0S Series OS MX78K0S	Basics	U12938E	U12938J

Other related documents

Document Name	Document No.	
	English	Japanese
SEMICONDUCTORS SELECTION GUIDE Products & Package	X13769X	
Semiconductor Device Mounting Technology Manual	C10535E	C10535J
Quality Grades on NEC Semiconductor Device	C11531E	C11531J
NEC Semiconductor Device Reliability/Quality Control System	C10983E	C10983J
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E	C11892J
Guide to Microcomputer-Related Products by Third Party	–	U11416J

Caution The related documents listed above are subject to change without notice. Be sure to use the latest documents for designing, etc.

CONTENTS

CHAPTER 1 GENERAL 19

1.1 Features 19
1.2 Applications 19
1.3 Ordering Information 19
1.4 Pin Configuration (Top View) 20
1.5 78K/0S Series Lineup 21
1.6 Block Diagram 23
1.7 Overview of Functions 24

CHAPTER 2 PIN FUNCTIONS 25

2.1 Pin Function List 25
2.2 Description of Pin Functions 27
2.2.1 P00 to P07 (Port 0) 27
2.2.2 P10 to P15 (Port 1) 27
2.2.3 P20 to P27 (Port 2) 27
2.2.4 P30, P31 (Port 3) 28
2.2.5 $\overline{\text{RESET}}$ 28
2.2.6 X1, X2 28
2.2.7 XT1, XT2 28
2.2.8 V_{DD} 28
2.2.9 V_{SS} 28
2.2.10 V_{PP} (μPD78F9076 only) 28
2.2.11 IC (mask ROM version only) 29
2.3 Pin I/O Circuits and Recommended Connection of Unused Pins 30

CHAPTER 3 CPU ARCHITECTURE 33

3.1 Memory Space 33
3.1.1 Internal program memory space 37
3.1.2 Internal data memory (internal high-speed RAM) space 37
3.1.3 Special function register (SFR) area 37
3.1.4 Data memory addressing 38
3.2 Processor Registers 42
3.2.1 Control registers 42
3.2.2 General-purpose registers 45
3.2.3 Special function registers (SFRs) 46
3.3 Instruction Address Addressing 49
3.3.1 Relative addressing 49
3.3.2 Immediate addressing 50
3.3.3 Table indirect addressing 51
3.3.4 Register addressing 51

3.4 Operand Address Addressing 52
3.4.1 Direct addressing 52
3.4.2 Short direct addressing 53
3.4.3 Special function register (SFR) addressing 54
3.4.4 Register addressing 55
3.4.5 Register indirect addressing 56
3.4.6 Based addressing 57
3.4.7 Stack addressing 57

CHAPTER 4 PORT FUNCTIONS 59

4.1 Port Functions 59
4.2 Port Configuration 61
4.2.1 Port 0 61
4.2.2 Port 1 62
4.2.3 Port 2 63
4.2.4 Port 3 67
4.3 Port Function Control Registers 68
4.4 Operation of Port Functions 71
4.4.1 Writing to I/O port 71
4.4.2 Reading from I/O port 71
4.4.3 Arithmetic operation of I/O port 71

CHAPTER 5 CLOCK GENERATOR 73

5.1 Clock Generator Functions 73
5.2 Clock Generator Configuration 73
5.3 Clock Generator Control Register 74
5.4 System Clock Oscillators 75
5.4.1 System clock oscillator 75
5.4.2 Frequency divider 77
5.5 Clock Generator Operation 78
5.6 Changing Setting of CPU Clock 79
5.6.1 Time required for switching CPU clock 79
5.6.2 Switching CPU clock 79

CHAPTER 6 16-BIT TIMER 81

6.1 16-Bit Timer Functions 81
6.2 16-Bit Timer Configuration 81
6.3 16-Bit Timer Control Registers 84
6.4 16-Bit Timer Operation 88
6.4.1 Operation as timer interrupt 88
6.4.2 Operation as timer output 90
6.4.3 Capture operation 91
6.4.4 16-bit timer counter 90 readout 92
6.4.5 Buzzer output operation 93

CHAPTER 7 8-BIT TIMER/EVENT COUNTER 80 95

7.1 Functions of 8-Bit Timer/Event Counter 80 95
7.2 8-Bit Timer/Event Counter 80 Configuration 96
7.3 8-Bit Timer/Event Counter 80 Control Registers 97
7.4 Operation of 8-Bit Timer/Event Counter 80 99
7.4.1 Operation as interval timer 99
7.4.2 Operation as external event counter 101
7.4.3 Operation as square wave output 102
7.4.4 Operation as PWM output 104
7.5 Notes on Using 8-Bit Timer/Event Counter 80 105

CHAPTER 8 WATCHDOG TIMER 107

8.1 Watchdog Timer Functions 107
8.2 Watchdog Timer Configuration 108
8.3 Watchdog Timer Control Registers 109
8.4 Watchdog Timer Operation 111
8.4.1 Operation as watchdog timer 111
8.4.2 Operation as interval timer 112

CHAPTER 9 SERIAL INTERFACE 20 113

9.1 Serial Interface 20 Functions 113
9.2 Serial Interface 20 Configuration 113
9.3 Serial Interface 20 Control Registers 117
9.4 Serial Interface 20 Operation 124
9.4.1 Operation stop mode 124
9.4.2 Asynchronous serial interface (UART) mode 125
9.4.3 3-wire serial I/O mode 137

CHAPTER 10 INTERRUPT FUNCTIONS 147

10.1 Interrupt Function Types 147
10.2 Interrupt Sources and Configuration 147
10.3 Interrupt Function Control Registers 150
10.4 Interrupt Processing Operation 155
10.4.1 Non-maskable interrupt request acknowledgement operation 155
10.4.2 Maskable interrupt request acknowledgement operation 157
10.4.3 Multiple interrupt servicing 159
10.4.4 Interrupt request reserve 161

CHAPTER 11 STANDBY FUNCTION 163

11.1 Standby Function and Configuration 163
11.1.1 Standby function 163
11.1.2 Standby function control register 164
11.2 Operation of Standby Function 165
11.2.1 HALT mode 165

11.2.2 STOP mode 167

CHAPTER 12 RESET FUNCTION 169

CHAPTER 13 μPD78F9076 173

13.1 Flash Memory Programming 174

13.1.1 Selecting communication mode 174
13.1.2 Function of flash memory programming 175
13.1.3 Flashpro III connection example 175
13.1.4 Setting Example with Flashpro III (PG-FP3) 177

CHAPTER 14 INSTRUCTION SET OVERVIEW 179

14.1 Operation 179

14.1.1 Operand identifiers and description methods 179
14.1.2 Description of "Operation" column 180
14.1.3 Description of "Flag" column 180

14.2 Operation List 181
14.3 Instructions Listed by Addressing Type 186

APPENDIX A DEVELOPMENT TOOLS 189

A.1 Language Processing Software 191
A.2 Flash Memory Writing Tools 192
A.3 Debugging Tools 192

A.3.1 Hardware 192
A.3.2 Software 193

APPENDIX B EMBEDDED SOFTWARE 195

APPENDIX C REGISTER INDEX 197

C.1 Register Name Index (Alphabetic Order) 197
C.2 Register Symbol Index (Alphabetic Order) 199

LIST OF FIGURES (1/3)

Figure No.	Title	Page
2-1	Pin Input/Output Circuits	31
3-1	Memory Map (μPD789071)	33
3-2	Memory Map (μPD789072)	34
3-3	Memory Map (μPD789074)	35
3-4	Memory Map (μPD78F9076)	36
3-5	Data Memory Addressing (μPD789071)	38
3-6	Data Memory Addressing (μPD789072)	39
3-7	Data Memory Addressing (μPD789074)	40
3-8	Data Memory Addressing (μPD78F9076)	41
3-9	Program Counter Configuration	42
3-10	Program Status Word Configuration	42
3-11	Stack Pointer Configuration	44
3-12	Data to Be Saved to Stack Memory	44
3-13	Data to Be Restored from Stack Memory	44
3-14	General-Purpose Register Configuration	45
4-1	Port Types	59
4-2	Block Diagram of P00 to P07	61
4-3	Block Diagram of P10 to P15	62
4-4	Block Diagram of P20	63
4-5	Block Diagram of P21	64
4-6	Block Diagram of P22 to P26	65
4-7	Block Diagram of P27	66
4-8	Block Diagram of P30 and P31	67
4-9	Format of Port Mode Register	68
4-10	Format of Pull-up Resistor Option Register 0	69
4-11	Format of Pull-up Resistor Option Register B2	70
5-1	Block Diagram of Clock Generator	73
5-2	Format of Processor Clock Control Register	74
5-3	External Circuit of System Clock Oscillator	75
5-4	Example of Incorrect Resonator Connection	76
5-5	Switching Between System Clock and CPU Clock	79
6-1	Block Diagram of 16-Bit Timer	82
6-2	Format of 16-Bit Timer Mode Control Register 90	85
6-3	Format of Buzzer Output Control Register 90	86
6-4	Format of Port Mode Register 3	87
6-5	Settings of 16-Bit Timer Mode Control Register 90 for Timer Interrupt Operation	88
6-6	Timing of Timer Interrupt Operation	89
6-7	Settings of 16-Bit Timer Mode Control Register 90 for Timer Output Operation	90

LIST OF FIGURES (2/3)

Figure No.	Title	Page
6-8	Timer Output Timing	90
6-9	Settings of 16-Bit Timer Mode Control Register 90 for Capture Operation	91
6-10	Capture Operation Timing (Both Edges of CPT90 Pin Are Specified)	91
6-11	16-Bit Timer Counter 90 Readout Timing	92
6-12	Settings of Buzzer Output Control Register 90 for Buzzer Output Operation	93
7-1	Block Diagram of 8-Bit Timer/Event Counter 80	96
7-2	Format of 8-Bit Timer Mode Control Register 80	97
7-3	Format of Port Mode Register 2	98
7-4	Interval Timer Operation Timing	100
7-5	External Event Counter Operation Timing (with Rising Edge Specified)	101
7-6	Square Wave Output Timing	103
7-7	PWM Output Timing	104
7-8	Start Timing of 8-Bit Timer Counter 80	105
7-9	External Event Counter Operation Timing	105
8-1	Block Diagram of Watchdog Timer	108
8-2	Format of Watchdog Timer Clock Selection Register	109
8-3	Format of Watchdog Timer Mode Register	110
9-1	Block Diagram of Serial Interface 20	114
9-2	Block Diagram of Baud Rate Generator 20	115
9-3	Format of Serial Operation Mode Register 20	117
9-4	Format of Asynchronous Serial Interface Mode Register 20	118
9-5	Format of Asynchronous Serial Interface Status Register 20	120
9-6	Format of Baud Rate Generator Control Register 20	121
9-7	Format of Asynchronous Serial Interface Transmit/Receive Data	131
9-8	Asynchronous Serial Interface Transmission Completion Interrupt Timing	133
9-9	Asynchronous Serial Interface Reception Completion Interrupt Timing	134
9-10	Receive Error Timing	135
9-11	3-Wire Serial I/O Mode Timing	140
10-1	Basic Configuration of Interrupt Function	149
10-2	Format of Interrupt Request Flag Register	151
10-3	Format of Interrupt Mask Flag Register	152
10-4	Format of External Interrupt Mode Register 0	153
10-5	Program Status Word Configuration	154
10-6	Flowchart from Non-Maskable Interrupt Request Generation to Acknowledgement	156
10-7	Timing of Non-Maskable Interrupt Request Acknowledgement	156
10-8	Acknowledgement of Non-Maskable Interrupt Request	156
10-9	Interrupt Request Acknowledgement Processing Algorithm	158
10-10	Interrupt Request Acknowledgement Timing (Example of MOV A,r)	159

LIST OF FIGURES (3/3)

Figure No.	Title	Page
10-11	Interrupt Request Acknowledgement Timing (When Interrupt Request Flag Is Set at the Last Clock During Instruction Execution)	159
10-12	Example of Multiple Interrupts	160
11-1	Format of Oscillation Stabilization Time Selection Register	164
11-2	Releasing HALT Mode by Interrupt	165
11-3	Releasing HALT Mode by $\overline{\text{RESET}}$ Input	166
11-4	Releasing STOP Mode by Interrupt	168
11-5	Releasing STOP Mode by $\overline{\text{RESET}}$ Input	168
12-1	Block Diagram of Reset Function	169
12-2	Reset Timing by $\overline{\text{RESET}}$ Input	170
12-3	Reset Timing by Watchdog Timer Overflow	170
12-4	Reset Timing by $\overline{\text{RESET}}$ Input in STOP Mode	170
13-1	Format of Communication Mode Selection	174
13-2	Flashpro III Connection Example in 3-Wire Serial I/O Mode	175
13-3	Flashpro III Connection Example in UART Mode	176
13-4	Flashpro III Connection Example in Pseudo 3-Wire Mode (When P0 Is Used)	176
A-1	Development Tools	190

LIST OF TABLES (1/2)

Table No.	Title	Page
2-1	Types of Pin Input/Output Circuits and Recommended Connection of Unused Pins	30
3-1	Internal ROM Capacity	37
3-2	Vector Table	37
3-3	Special Function Registers	47
4-1	Port Functions	60
4-2	Configuration of Port	61
4-3	Port Mode Register and Output Latch Settings for Using Alternate Functions	69
5-1	Configuration of Clock Generator	73
5-2	Maximum Time Required for Switching CPU Clock	79
6-1	Configuration of 16-Bit Timer	81
6-2	Interval Time of 16-Bit Timer	88
6-3	Settings of Capture Edge	91
6-4	Buzzer Frequency of 16-Bit Timer	93
7-1	Interval Time of 8-Bit Timer/Event Counter 80	95
7-2	Square Wave Output Range of 8-Bit Timer/Event Counter 80	95
7-3	Configuration of 8-Bit Timer/Event Counter 80	96
7-4	Interval Time of 8-Bit Timer/Event Counter 80	99
7-5	Square Wave Output Range of 8-Bit Timer/Event Counter	102
8-1	Inadvertent Loop Detection Time of Watchdog Timer	107
8-2	Interval Time	107
8-3	Configuration of Watchdog Timer	108
8-4	Inadvertent Loop Detection Time of Watchdog Timer	111
8-5	Interval Generated Using Interval Timer	112
9-1	Configuration of Serial Interface 20	113
9-2	Serial Interface 20 Operating Mode Settings	119
9-3	Example of Relationship Between System Clock and Baud Rate	122
9-4	Relationship Between ASCK20 Pin Input Frequency and Baud Rate (When BRGC20 Is Set to 80H)	123
9-5	Example of Relationship Between System Clock and Baud Rate	130
9-6	Relationship Between ASCK20 Pin Input Frequency and Baud Rate (When BRGC20 Is Set to 80H)	130
9-7	Receive Error Causes	135
10-1	Interrupt Sources	148
10-2	Interrupt Request Signals and Corresponding Flags	150
10-3	Time from Generation of Maskable Interrupt Request to Servicing	157

LIST OF TABLES (2/2)

Table No.	Title	Page
11-1	Operation Statuses in HALT Mode	165
11-2	Operation After Releasing HALT Mode	166
11-3	Operation Statuses in STOP Mode	167
11-4	Operation After Releasing STOP Mode	168
12-1	Status of Hardware After Reset	171
13-1	Differences Between Flash Memory and Mask ROM Versions	173
13-2	Communication Mode	174
13-3	Major Functions of Flash Memory Programming	175
13-4	Setting Example with PG-FP3	177
14-1	Operand Identifiers and Description Methods	179

[MEMO]

CHAPTER 1 GENERAL

1.1 Features

• ROM and RAM capacity

Item / Product Name	Program Memory		Data Memory (Internal High-Speed RAM)
μPD789071	Mask ROM	2 KB	256 KB
μPD789072		4 KB	
μPD789074		8 KB	
μPD78F9076	Flash memory	16 KB	

- Minimum instruction execution time can be changed from high-speed (0.4 μs) to ultra-low speed (122 μs) at 5.0 MHz operation with system clock.
- I/O ports: 34
- Serial interface: 1 channel
  3-wire serial I/O mode/UART mode: 1 channel
- Timer: 3 channels
  - 16-bit timer: 1 channel
  - 8-bit timer/event counter: 1 channel
  - Watchdog timer: 1 channel
- Vectored interrupt sources: 8
- Supply voltage: V_{DD} = 1.8 to 5.5 V
- Operating ambient temperature: T_A = –40 to +85°C

1.2 Applications

Small, general home electrical appliances, telephones, etc.

1.3 Ordering Information

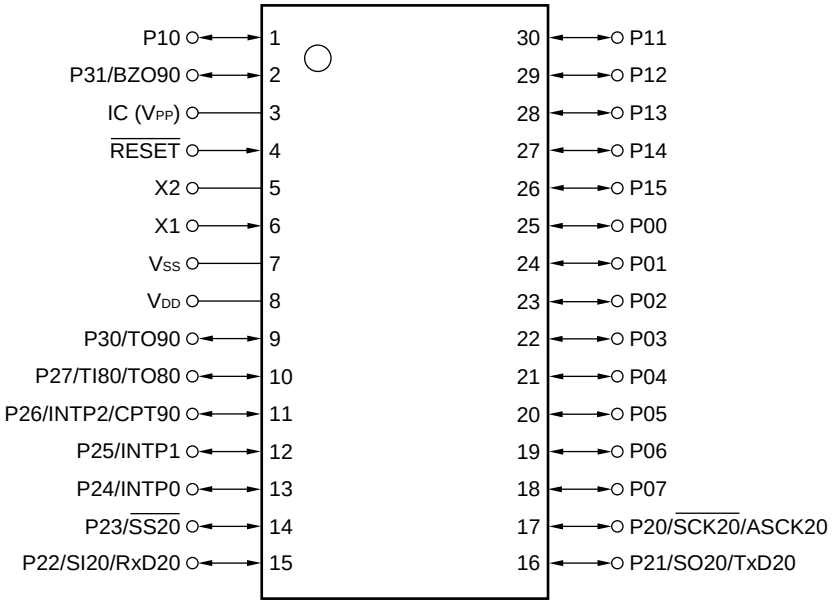
Part Number	Package	Internal ROM
μPD789071MC-×××-5A4	30-pin plastic SSOP (7.62 mm (300))	Mask ROM
μPD789072MC-×××-5A4	30-pin plastic SSOP (7.62 mm (300))	Mask ROM
μPD789074MC-×××-5A4	30-pin plastic SSOP (7.62 mm (300))	Mask ROM
μPD78F9076MC-5A4	30-pin plastic SSOP (7.62 mm (300))	Flash memory

Remark ××× indicates ROM code suffix.

1.4 Pin Configuration (Top View)

30-pin plastic SSOP (7.62 mm (300))

μPD789071MC-×××-5A4
μPD789072MC-×××-5A4
μPD789074MC-×××-5A4
μPD78F9076MC-5A4



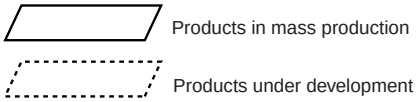
Caution Connect the IC (Internally Connected) pin directly to V_{SS}.

Remark Pin connections in parentheses are intended for the μPD78F9076.

ASCK20:	Asynchronous Serial Input	SCK20:	Serial Clock
BZO90:	Buzzer Output	SI20:	Serial Input
CPT90:	Capture Trigger Input	SO20:	Serial Output
IC:	Internally Connected	SS20:	Chip Select Input
INTP0 to INTP2:	External interrupt input	TI80:	Timer Input
P00 to P07:	Port 0	TO80, TO90:	Timer Output
P10 to P15:	Port 1	TxD20:	Transmit Data
P20 to P27:	Port 2	V_{DD}:	Power Supply
P30, P31:	Port 3	V_{PP}:	Programming Power Supply
RESET:	Reset	V_{SS}:	Ground
RxD20:	Receive Data	X1, X2:	Crystal/ceramic Oscillator

1.5 78K/0S Series Lineup

The products in the 78K/0S Series are listed below. The names enclosed in boxes are subseries names.



Y subseries products support SMB.

Small-scale package, general-purpose applications

44-pin	μPD789046		μPD789074 with added subsystem clock
42-/44-pin	μPD789026		μPD789014 with enhanced timer, expanded ROM, RAM capacity
30-pin	μPD789074		μPD789026 with enhanced timer
28-pin	μPD789014		On-chip UART, capable of operating at low-voltage (1.8 V)

Small-scale package, general-purpose applications and A/D function

44-pin	μPD789177	μPD789177Y	μPD789167 with enhanced A/D converter
44-pin	μPD789167	μPD789167Y	μPD789104A with enhanced timer
30-pin	μPD789156		μPD789146 with enhanced A/D converter
30-pin	μPD789146		μPD789104A with added EEPROM
30-pin	μPD789134A		μPD789124A with enhanced A/D converter
30-pin	μPD789124A		RC oscillation version of the μPD789104A
30-pin	μPD789114A		μPD789104A with enhanced A/D converter
30-pin	μPD789104A		μPD789026 with added A/D converter and multiplier

Inverter control

44-pin	μPD789842	On-chip inverter controller and UART

LCD drive

80-pin	μPD789417A	μPD789407A with enhanced A/D converter
80-pin	μPD789407A	μPD789456 with enhanced I/O
64-pin	μPD789456	μPD789446 with enhanced A/D converter
64-pin	μPD789446	μPD789426 with enhanced display output
64-pin	μPD789436	μPD789426 with enhanced A/D converter
64-pin	μPD789426	μPD789306 with added A/D converter
64-pin	μPD789316	RC oscillation version of the μPD789306
64-pin	μPD789306	Basic subseries for driving LCDs

Dot LCD drive

144-pin	μPD789835	Segment/common outputs: 96
88-pin	μPD789830	Segments: 40, commons: 16

ASSP

52-pin	μPD789467	μPD789327 with added A/D converter
52-pin	μPD789327	For remote controller, on-chip LCD controller/driver
44-pin	μPD789800	For PC keyboard, on-chip USB function
44-pin	μPD789840	For keypad, on-chip POC
20-pin	μPD789861	RC oscillation version of the μPD789860
20-pin	μPD789860	For keyless entry, on-chip POC and key return circuit

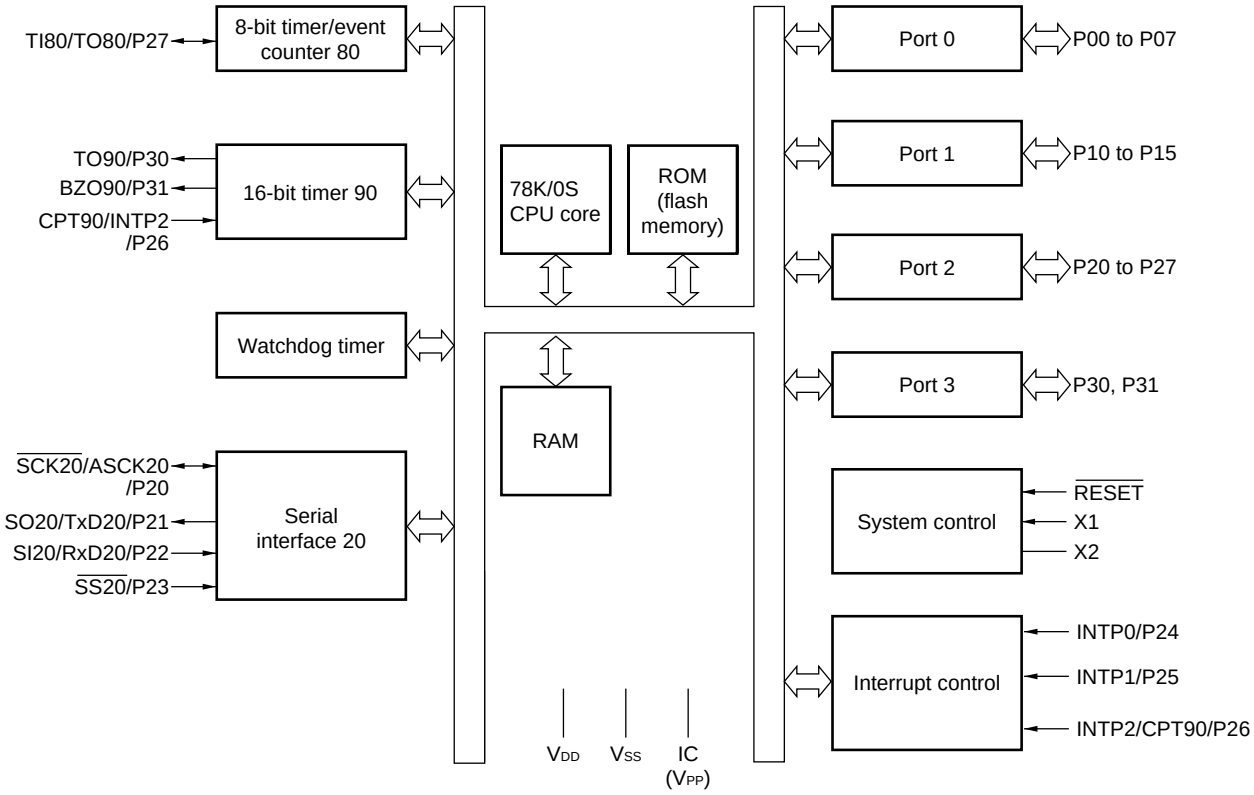
78K/0S Series

The major differences among the subseries are listed below.

Subseries \ Function		ROM Size	Timer				8-Bit A/D	10-Bit A/D	Serial Interface	I/O	Minimum V_{DD} Value	Remarks
			8-Bit	16-Bit	Watch	WDT						
Small-scale, general-purpose applications	μPD789046	16 K	1 ch	1 ch	1 ch	1 ch	–	–	1 ch (UART: 1 ch)	34	1.8 V	–
	μPD789026	4 K to 16 K			–							
	μPD789074	2 K to 8 K								24		
	μPD789014	2 K to 4 K	2 ch	–						22		
Small-scale, general-purpose applications and A/D function	μPD789177	16 K to 24 K	3 ch	1 ch	1 ch		–	8 ch	1 ch (UART: 1 ch)	31	1.8 V	–
	μPD789167						8 ch	–				
	μPD789156	8 K to 16 K	1 ch		–		–	4 ch		20		On-chip EEPROM
	μPD789146						4 ch	–				
	μPD789134A	2 K to 8 K					–	4 ch				RC-oscillator version
	μPD789124A						4 ch	–				
	μPD789114A						–	4 ch				–
	μPD789104A						4 ch	–				
Inverter control	μPD789842	8 K to 16 K	3 ch	**Note**	1 ch	1 ch	8 ch	–	1 ch (UART: 1 ch)	30	4.0 V	–
LCD drive	μPD789417A	12 K to 24 K	3 ch	1 ch	1 ch	1 ch	–	7 ch	1 ch (UART: 1 ch)	43	1.8 V	–
	μPD789407A						7 ch	–				
	μPD789456	12 K to 16 K	2 ch				–	6 ch		30		
	μPD789446						6 ch	–				
	μPD789436						–	6 ch		40		
	μPD789426						6 ch	–				
	μPD789316	8 K to 16 K					–		2 ch (UART: 1 ch)	23		RC-oscillator version
	μPD789306											–
Dot LCD drive	μPD789835	24 K to 60 K	6 ch	–	1 ch	1 ch	3 ch	–	1 ch (UART: 1 ch)	28	1.8 V	–
	μPD789830	24 K	1 ch	1 ch			–			30	2.7 V	
ASSP	μPD789467	4 K to 24 K	2 ch	–	1 ch	1 ch	1 ch	–	–	18	1.8 V	On-chip LCD
	μPD789327						–		1 ch	21		
	μPD789800	8 K	2 ch		–	1 ch	–		2 ch (USB: 1 ch)	31	4.0 V	–
	μPD789840						4 ch		1 ch	29	2.8 V	
	μPD789861	4 K					–		–	14	1.8 V	RC-oscillator version, on-chip EEPROM
	μPD789860											On-chip EEPROM

Note 10-bit timer: 1 channel

1.6 Block Diagram



Remarks 1. Internal ROM capacity varies depending on the product.

2. Pin connections in parentheses are intended for the μPD78F9076.

1.7 Overview of Functions

Item \ Part Number		μPD789071	μPD789072	μPD789074	μPD78F9076
Internal memory	ROM	Mask ROM			Flash memory
		2 KB	4 KB	8 KB	16 KB
	High-speed RAM	256 bytes			
Minimum instruction execution time		0.4/1.6 μs (@5.0 MHz operation with system clock)			
General-purpose registers		8 bits × 8 registers			
Instruction set		• 16-bit operations • Bit manipulations (such as set, reset, and test)			
I/O ports		CMOS I/O: 24			
Serial interface		Switchable between 3-wire serial I/O and UART modes: 1 channel			
Timers		• 16-bit timer: 1 channel • 8-bit timer/event counter: 1 channel • Watchdog timer: 1 channel			
Timer outputs		2			
Vectored interrupt sources	Maskable	Internal: 4, external: 3			
	Non-maskable	Internal: 1			
Power supply voltage		V_{DD} = 1.8 to 5.5 V			
Operating ambient temperature		T_A = –40 to +85°C			
Package		33-pin plastic SSOP (7.62 mm (300))			

The outline of the timer is as follows.

		16-Bit Timer 90	8-Bit Timer/Event Counter 80	Watchdog Timer
Operating mode	Interval timer	–	1 channel	1 channel[Note]
	External event counter	–	1 channel	–
Function	Timer outputs	1	1	–
	PWM outputs	–	1	–
	Square-wave outputs	–	1	–
	Buzzer outputs	1	–	–
	Capture	1 input	–	–
	Interrupt sources	1	1	1

Note The watchdog timer provides the watchdog timer function and interval timer function. Use either of the functions.

2.1 Pin Function List

(1) Port pins

Pin Name	I/O	Function	After Reset	Alternate Function
P00 to P07	I/O	Port 0 8-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	–
P10 to P15	I/O	Port 1 6-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	–
P20	I/O	Port 2 8-bit I/O port Input/output can be specified in 1-bit units. An on-chip pull-up resistor can be specified by means of the pull-up resistor option register B2 (PUB2).	Input	$\overline{\text{SCK20}}$/ASCK20
P21				SO20/TxD20
P22				SI20/RxD20
P23				$\overline{\text{SS20}}$
P24				INTP0
P25				INTP1
P26				INTP2/CPT90
P27				TI80/TO80
P30	I/O	Port 3 2-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	TO90
P31				BZO90

(2) Non-port pins

Pin Name	I/O	Function	After Reset	Alternate Function
INTP0	Input	External interrupt input for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified	Input	P24
INTP1				P25
INTP2				P26/CPT90
$\overline{SCK20}$	I/O	Serial interface (SIO10) serial clock input	Input	P20/ASCK20
SI20	Input	Serial interface (SIO20) serial data input	Input	P22/RxD20
SO20	Output	Serial interface (SIO20) serial data output	Input	P21/TxD20
$\overline{SS20}$	Input	Serial interface chip select input	Input	P23
ASCK20	Input	Serial clock input for asynchronous serial interface	Input	P20/$\overline{SCK20}$
RxD20	Input	Serial data input for asynchronous serial interface	Input	P22/SI20
TxD20	Output	Serial data output for asynchronous serial interface	Input	P21/SO20
TO90	Output	16-bit timer (TM90) output	Input	P30
BZO90	Output	Buzzer output	Input	P31
CPT90	Input	Capture edge input	Input	P26/INTP2
TO80	Output	8-bit timer (TM80) output	Input	P27/TI80
TI80	Input	External count clock input to 8-bit timer (TM80)	Input	P27/TO80
X1	Input	Connecting crystal resonator for system clock oscillation	–	–
X2	–		–	–
$\overline{RESET}$	Input	System reset input	Input	–
V_{DD}	–	Positive supply voltage	–	–
V_{SS}	–	Ground potential	–	–
IC	–	Internally connected. Connect directly to V_{SS}.	–	–
V_{PP}	–	This pin is used to set the flash memory programming mode and applies a high voltage when a program is written or verified. In normal operation mode, connect directly to V_{SS}.	–	–

2.2 Description of Pin Functions

2.2.1 P00 to P07 (Port 0)

These pins constitute an 8-bit I/O port and can be set to input or output port mode in 1-bit units by using port mode register 0 (PM0). When these pins are used as an input port, an on-chip pull-up resistor can be used by setting pull-up resistor option register 0 (PU0).

2.2.2 P10 to P15 (Port 1)

These pins constitute a 6-bit I/O port and can be set to input or output port mode in 1-bit units by using port mode register 1 (PM1). When these pins are used as an input port, an on-chip pull-up resistor can be used by setting pull-up resistor option register 0 (PU0).

2.2.3 P20 to P27 (Port 2)

These pins constitute an 8-bit I/O port. In addition, these pins provide a function to perform input/output to/from the timer, to input/output the data and clock of the serial interface, and to input the external interrupt.

Port 2 can be set to the following operation modes in 1-bit units.

(1) Port mode

In port mode, P20 to P27 function as an 8-bit I/O port. Port 2 can be set to input or output mode in 1-bit units by using port mode register 2 (PM2). For P20 to P27, whether to use on-chip pull-up resistors can be specified in 1-bit units by using pull-up resistor option register B2 (PUB2), regardless of the setting of port mode register 2 (PM2).

(2) Control mode

In this mode, P20 to P27 function as the timer input/output and the serial interface data and clock input/output.

(a) TI80

This is the external clock input pin for the 8-bit timer/event counter 80.

(b) TO80

This is the timer output pin of the 8-bit timer/event counter 80.

(c) INTP0 to INTP2

These are external interrupt input pins for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified.

(d) CPT90

This is the capture edge input pin of the 16-bit timer counter 90.

(e) SI20, SO20

This is the serial data I/O pin of the serial interface.

(f) $\overline{\text{SCK20}}$

This is the serial clock I/O pin of the serial interface.

(g) $\overline{\text{SS20}}$

This is the chip select input pin of the serial interface.

(h) RxD20, TxD20

These are the serial data I/O pins of the asynchronous serial interface.

(i) ASCK20

This is the serial clock input pin of the asynchronous serial interface.

Caution When using P20 to P27 as serial interface pins, the input/output mode and output latch must be set according to the functions to be used. For details of the setting, see Table 9-2 Serial Interface 20 Operation Mode Settings.

2.2.4 P30, P31 (Port 3)

These pins constitute a 2-bit I/O port. In addition, these pins function as the timer output and the buzzer output.

Port 3 can be set to the following operation modes in 1-bit units.

(1) Port mode

When this port is used as an input port, an on-chip pull-up resistor can be used by setting pull-up resistor option register 0 (PU0).

(2) Control mode

In this mode, P30 and P31 function as the timer output and the buzzer output.

(a) TO90

This is the output pin of the 16-bit timer 90.

(b) BZO90

This is the buzzer output pin of the 16-bit timer 90.

2.2.5 $\overline{\text{RESET}}$

An active-low system reset signal is input to this pin.

2.2.6 X1, X2

These pins are used to connect a crystal resonator for system clock oscillation.

To supply an external clock, input the clock to X1 and input the inverted signal to X2.

2.2.7 XT1, XT2

These pins are used to connect a crystal resonator for subsystem clock oscillation.

To supply an external clock, input the clock to XT1 and input the inverted signal to XT2.

2.2.8 V_{DD}

This pin supplies positive power.

2.2.9 V_{SS}

This pin is the ground potential pin.

2.2.10 V_{PP} (μPD78F9076 only)

A high voltage should be applied to this pin when the flash memory programming mode is set and when the program is written or verified.

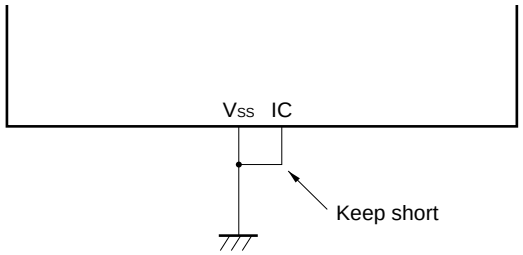
Directly connect this pin to V_{SS} in normal operation mode.

2.2.11 IC (mask ROM version only)

The IC (Internally Connected) pin is used to set the μPD789071, 789072, and 789074 to test mode before shipment. In normal operation mode, directly connect this pin to the V_{SS} pin with as short a wiring length as possible.

If a potential difference is generated between the IC pin and the V_{SS} pin due to a long wiring length between these pins or an external noise superimposed on the IC pin, the user program may not run correctly.

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



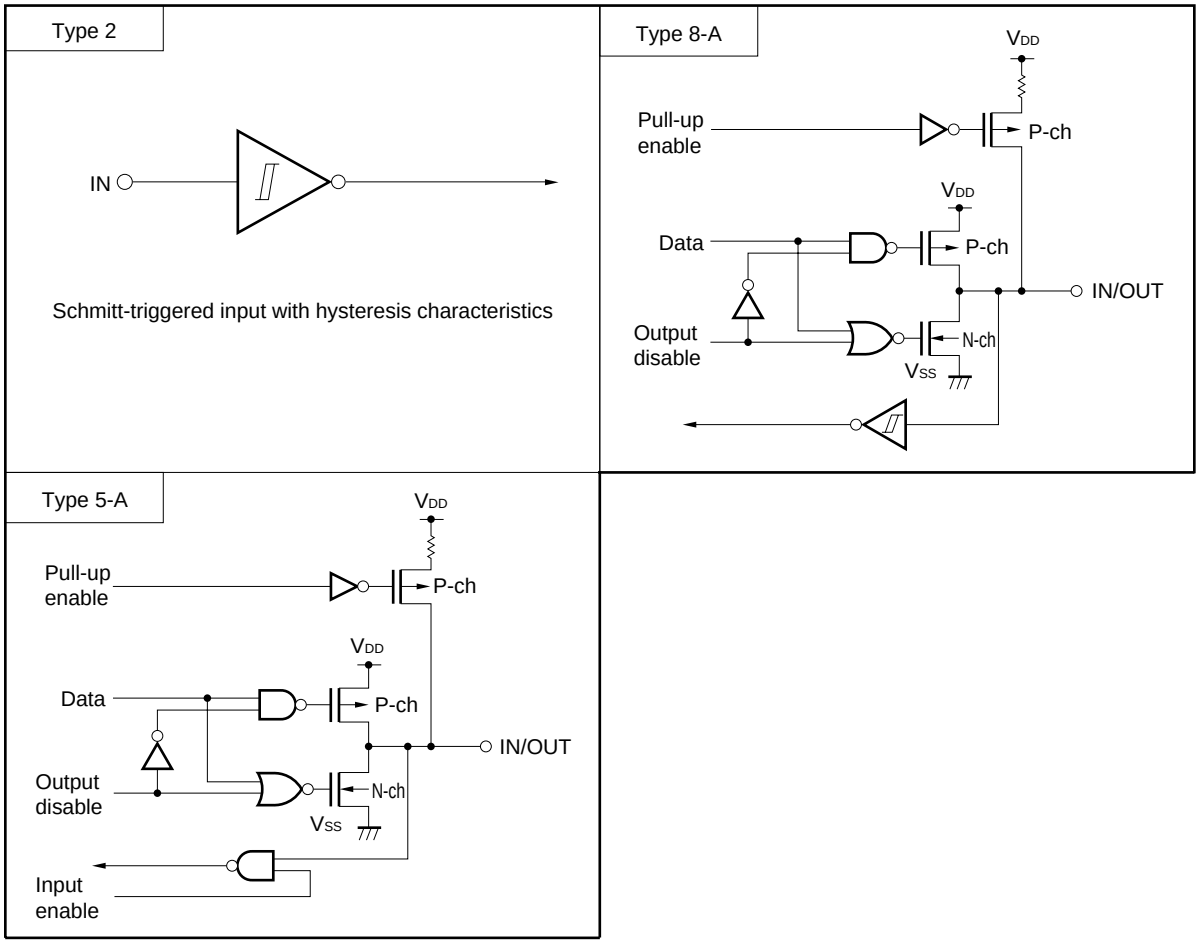
2.3 Pin I/O Circuits and Recommended Connection of Unused Pins

The input/output circuit type of each pin and recommended connection of unused pins are shown in Table 2-1. For the input/output circuit configuration of each type, refer to Figure 3-1.

Table 2-1. Types of Pin Input/Output Circuits and Recommended Connection of Unused Pins

Pin Name	I/O Circuit Type	I/O	Recommended Connection of Unused Pins
P00 to P07	5-A	I/O	Input: Connect to V_{DD} or V_{SS} via a resistor. Output: Leave open.
P10 to P15			
P20/$\overline{SCK20}$/ASCK20	8-A		
P21/SO20/TxD20			
P22/SI20/RxD20			
P23/$\overline{SS20}$			
P24/INTP0			Input: Connect to V_{SS} via a resistor. Output: Leave open.
P25/INTP1			
P26/INTP2/CPT90			
P27/TI80/TO80			Input: Connect to V_{DD} or V_{SS} via a resistor. Output: Leave open.
P30/TO90	5-A		
P31/BZO90			
$\overline{RESET}$	2	Input	–
IC	–	–	Connect directly to V_{SS}.
V_{PP}			

Figure 2-1. Pin Input/Output Circuits



[MEMO]

3.1 Memory Space

The μPD789074 Subseries can each access up to 64 KB of memory space. Figures 3-1 through 3-4 show the memory maps.

Figure 3-1. Memory Map (μPD789071)

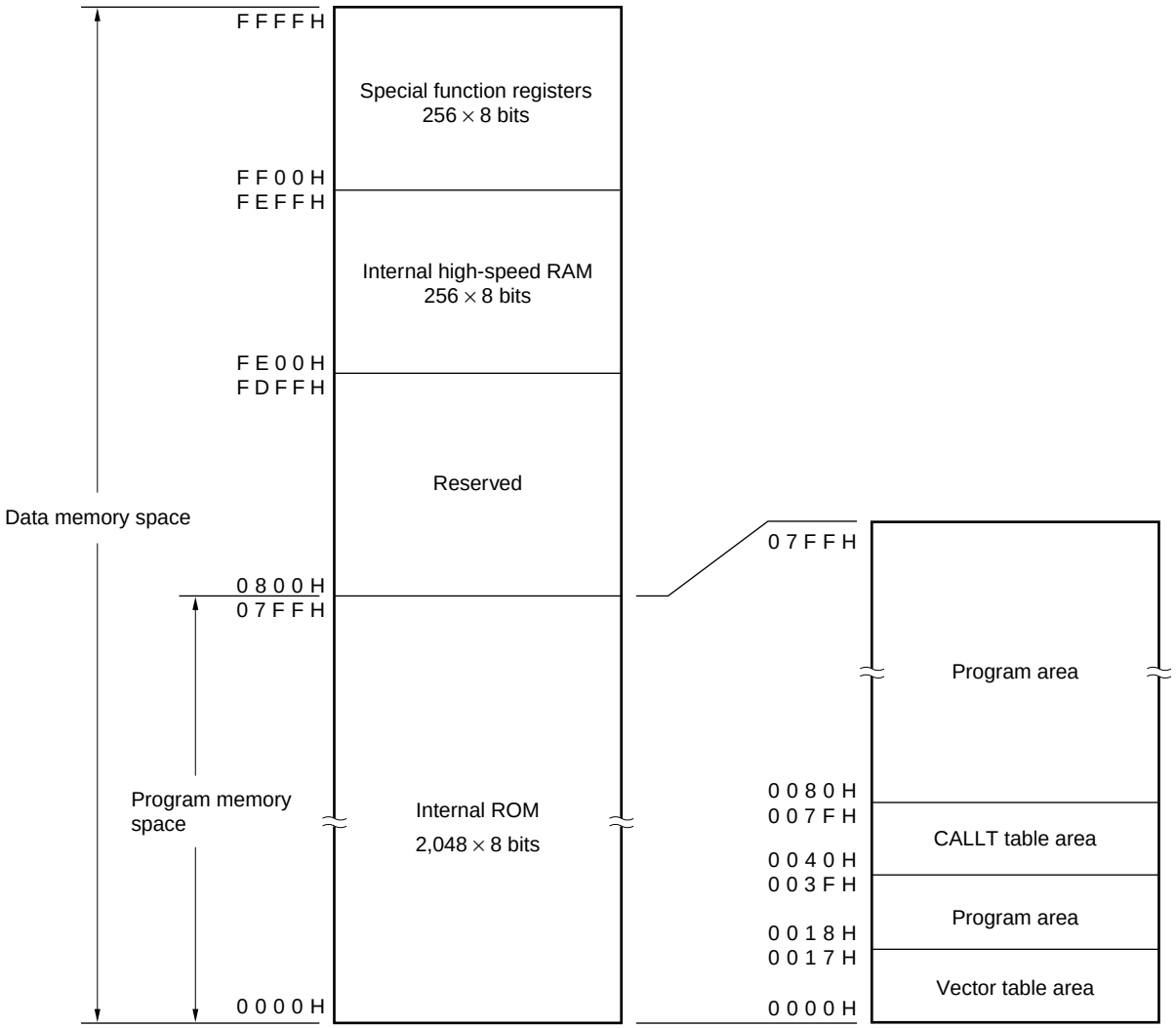


Figure 3-2. Memory Map (μPD789072)

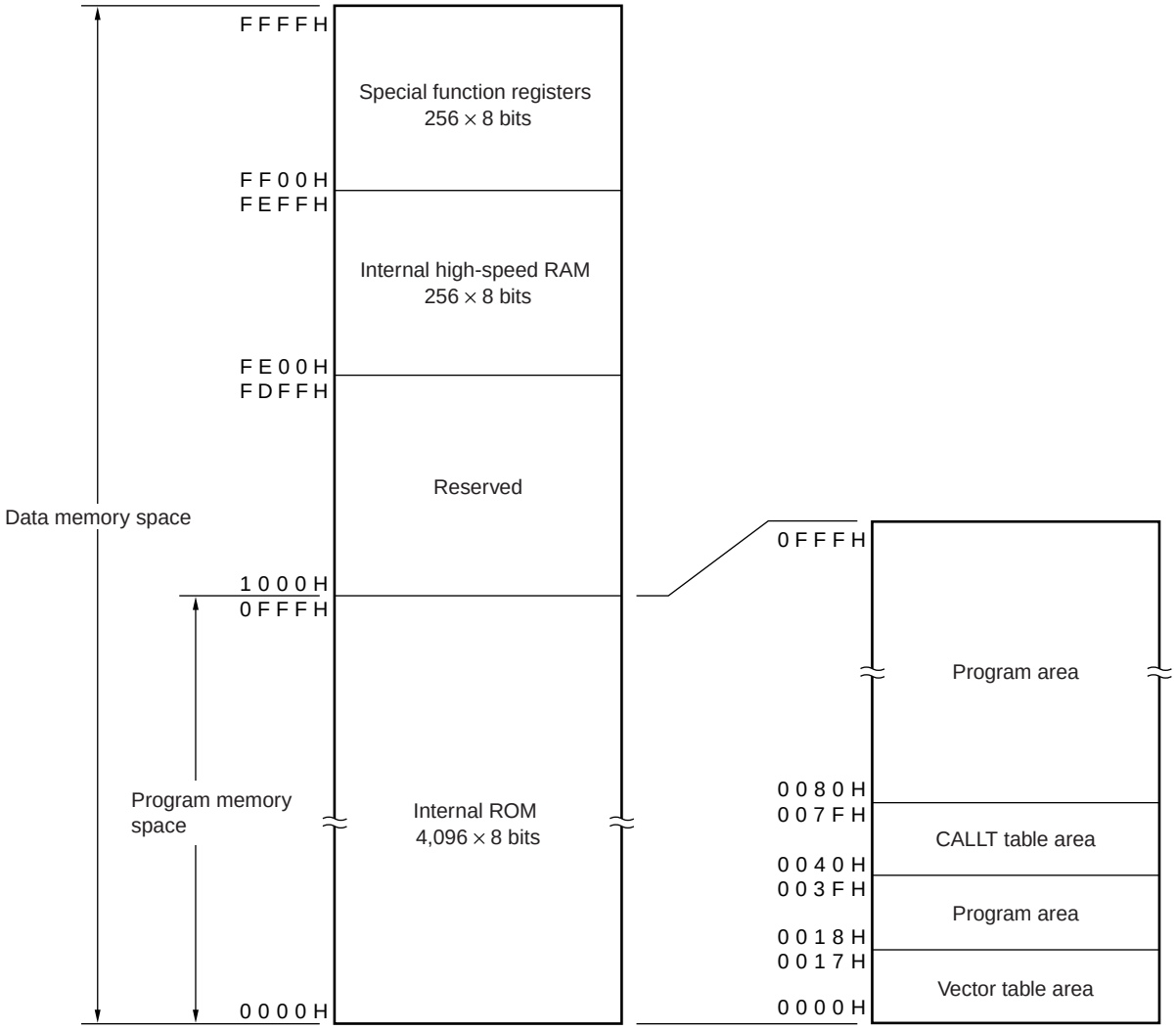


Figure 3-3. Memory Map (μPD789074)

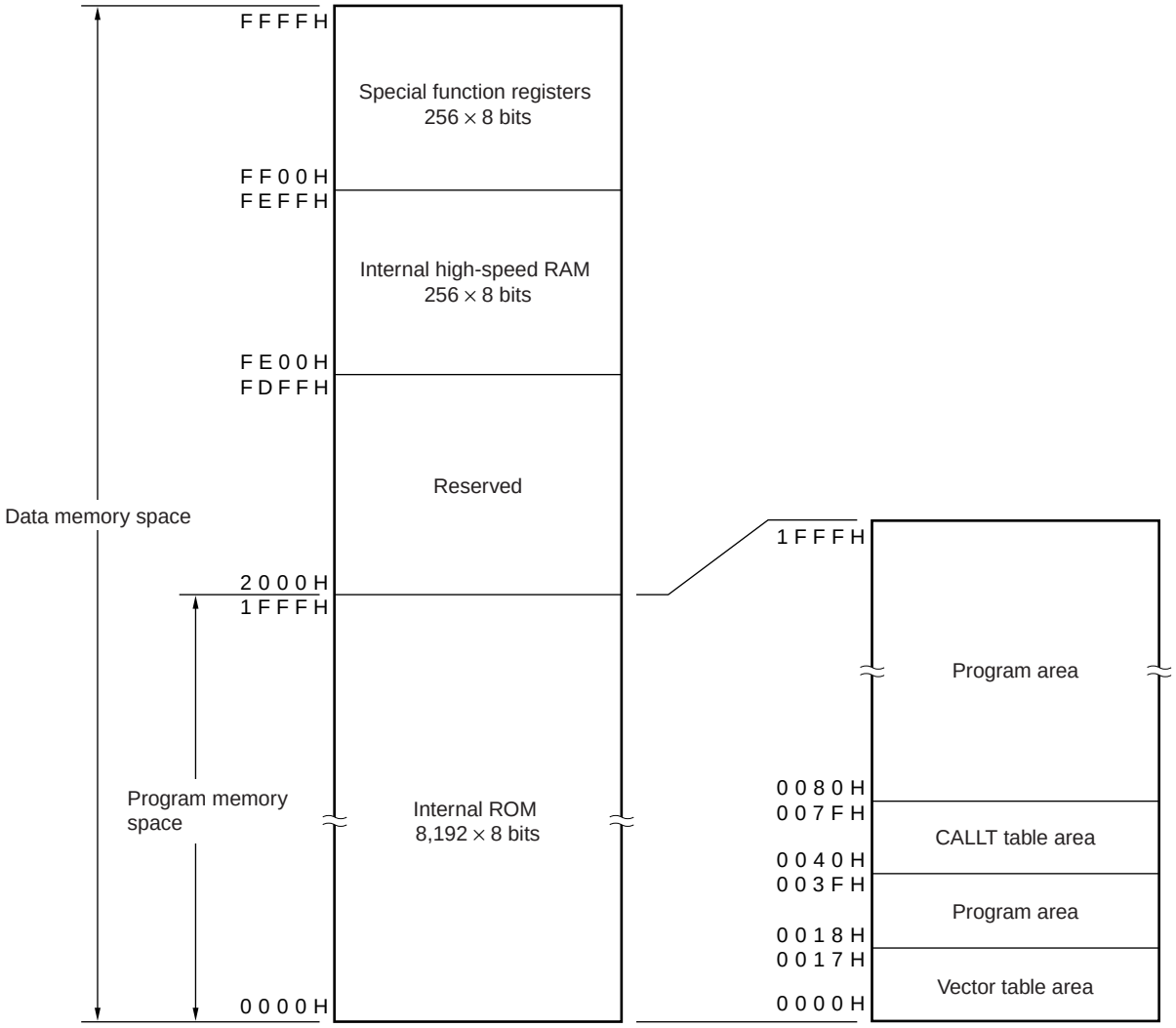
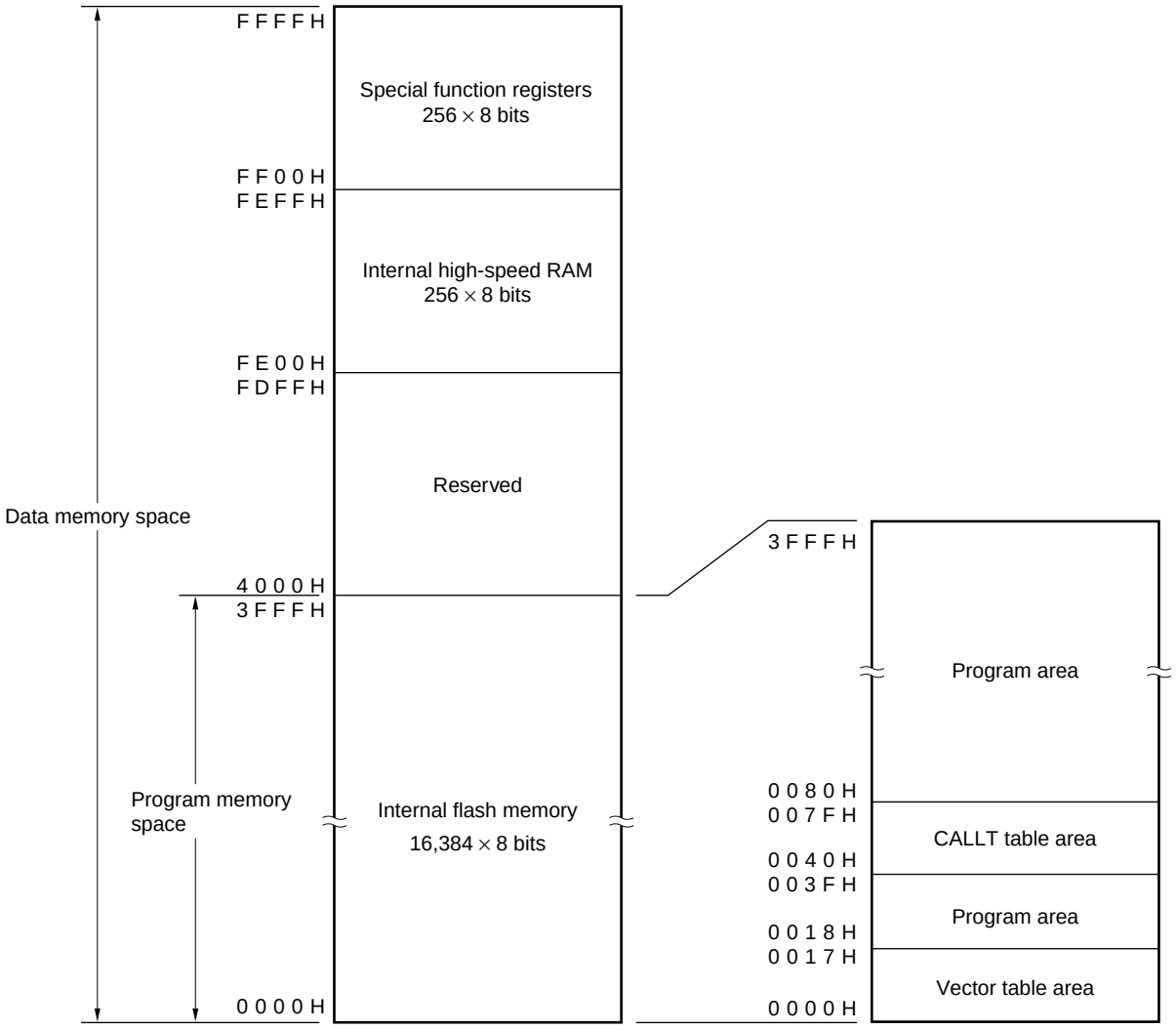


Figure 3-4. Memory Map (μPD78F9076)



3.1.1 Internal program memory space

The internal program memory space stores programs and table data. This space is usually addressed by the program counter (PC).

The μPD789046 Subseries provide the following internal ROMs (or flash memory) containing the following capacities.

Table 3-1. Internal ROM Capacity

Part Number	Internal ROM	
	Structure	Capacity
μPD789071	Mask ROM	2,048 × 8 bits
μPD789072		4,096 × 8 bits
μPD789074		8,192 × 8 bits
μPD78F9076	Flash memory	16,384 × 8 bits

The following areas are allocated to the internal program memory space:

(1) Vector table area

The 24-byte area of addresses 0000H to 0017H is reserved as a vector table area. This area stores program start addresses to be used when branching by $\overline{\text{RESET}}$ input or interrupt request generation. Of a 16-bit program address, the lower 8 bits are stored in an even address, and the higher 8 bits are stored in an odd address.

Table 3-2. Vector Table

Vector Table Address	Interrupt Request	Vector Table Address	Interrupt Request
0000H	$\overline{\text{RESET}}$ input	000CH	INTSR20/INTCSI20
0004H	INTWDT	000EH	INTST20
0006H	INTP0	0014H	INTTM80
0008H	INTP1	0016H	INTTM90
000AH	INTP2		

(2) CALLT instruction table area

The subroutine entry address of a 1-byte call instruction (CALLT) can be stored in the 64-byte area of addresses 0040H to 007FH.

3.1.2 Internal data memory (internal high-speed RAM) space

The μPD789074 Subseries provide 256-byte internal high-speed RAM.

The internal high-speed RAM can also be used as a stack memory.

3.1.3 Special function register (SFR) area

Special function registers (SFRs) of on-chip peripheral hardware are allocated to the area of FF00H to FFFFH (see **Table 3-3**).

3.1.4 Data memory addressing

Each of the μPD789074 Subseries is provided with a wide range of addressing modes to make memory manipulation as efficient as possible. The data memory area (FE00H to FFFFH) can be accessed using a unique addressing mode according to its use, such as a special function register (SFR). Figures 3-5 through 3-8 illustrate the data memory addressing.

Figure 3-5. Data Memory Addressing (μPD789071)

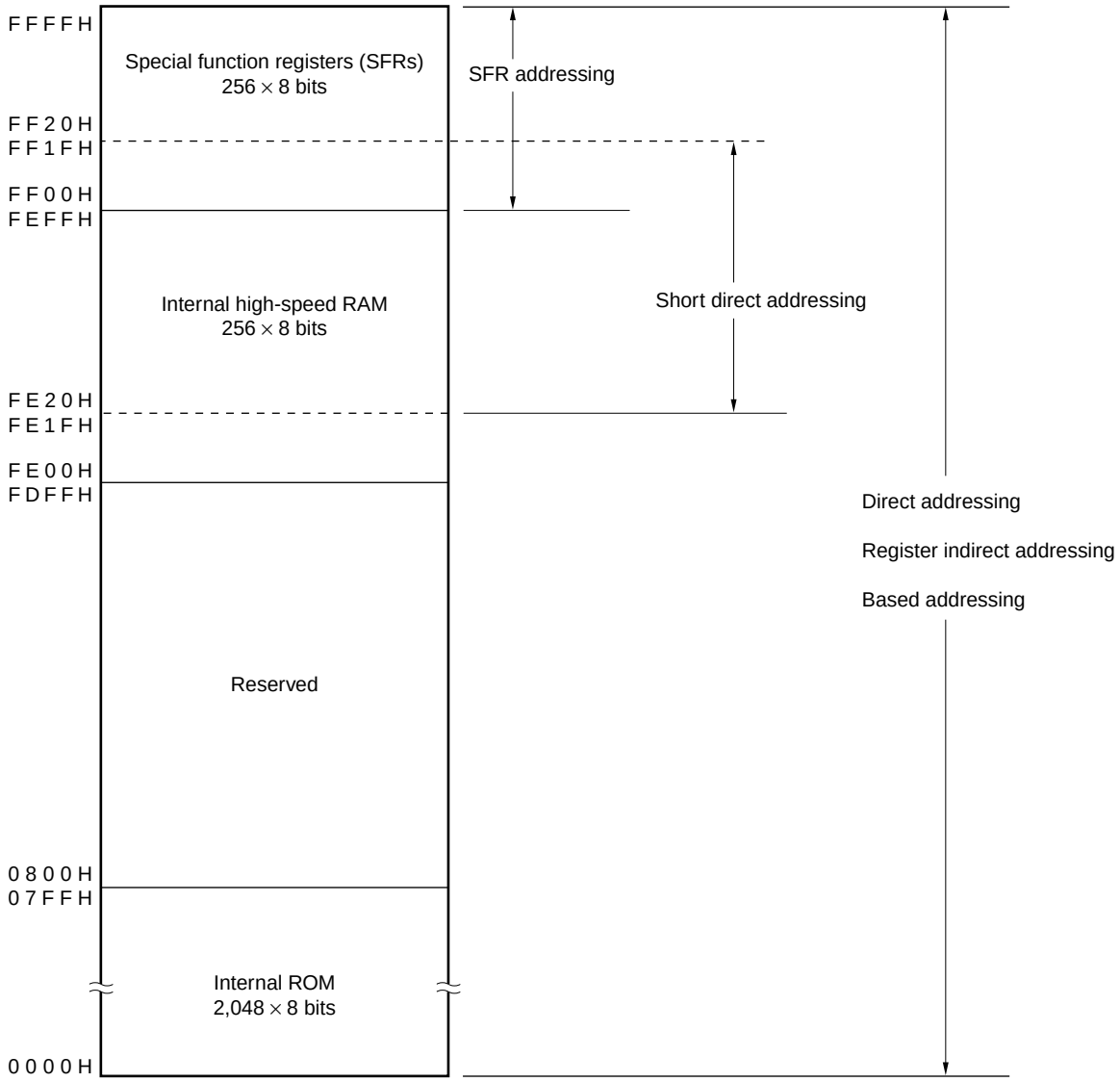


Figure 3-6. Data Memory Addressing (μPD789072)

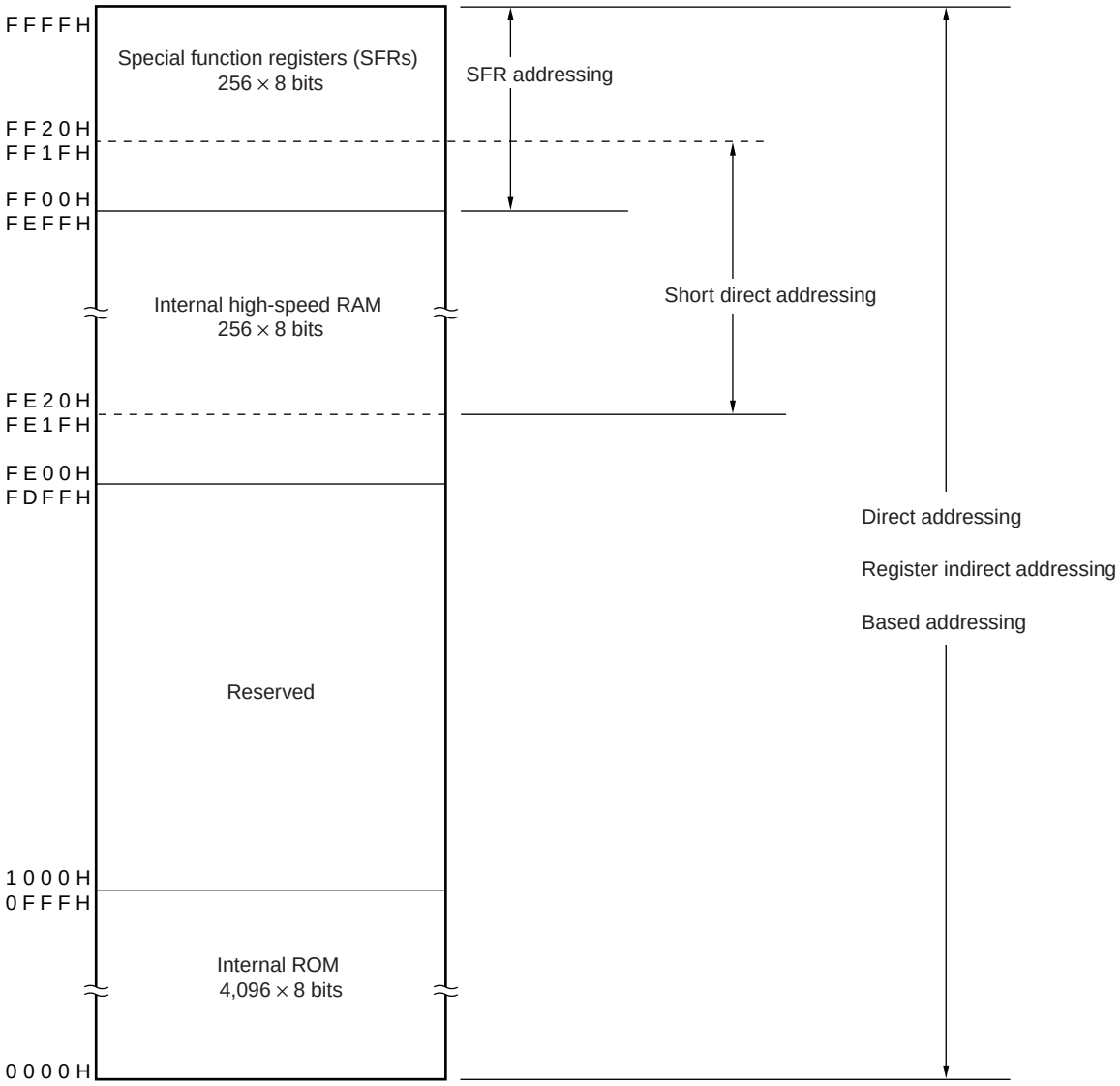


Figure 3-7. Data Memory Addressing (μPD789074)

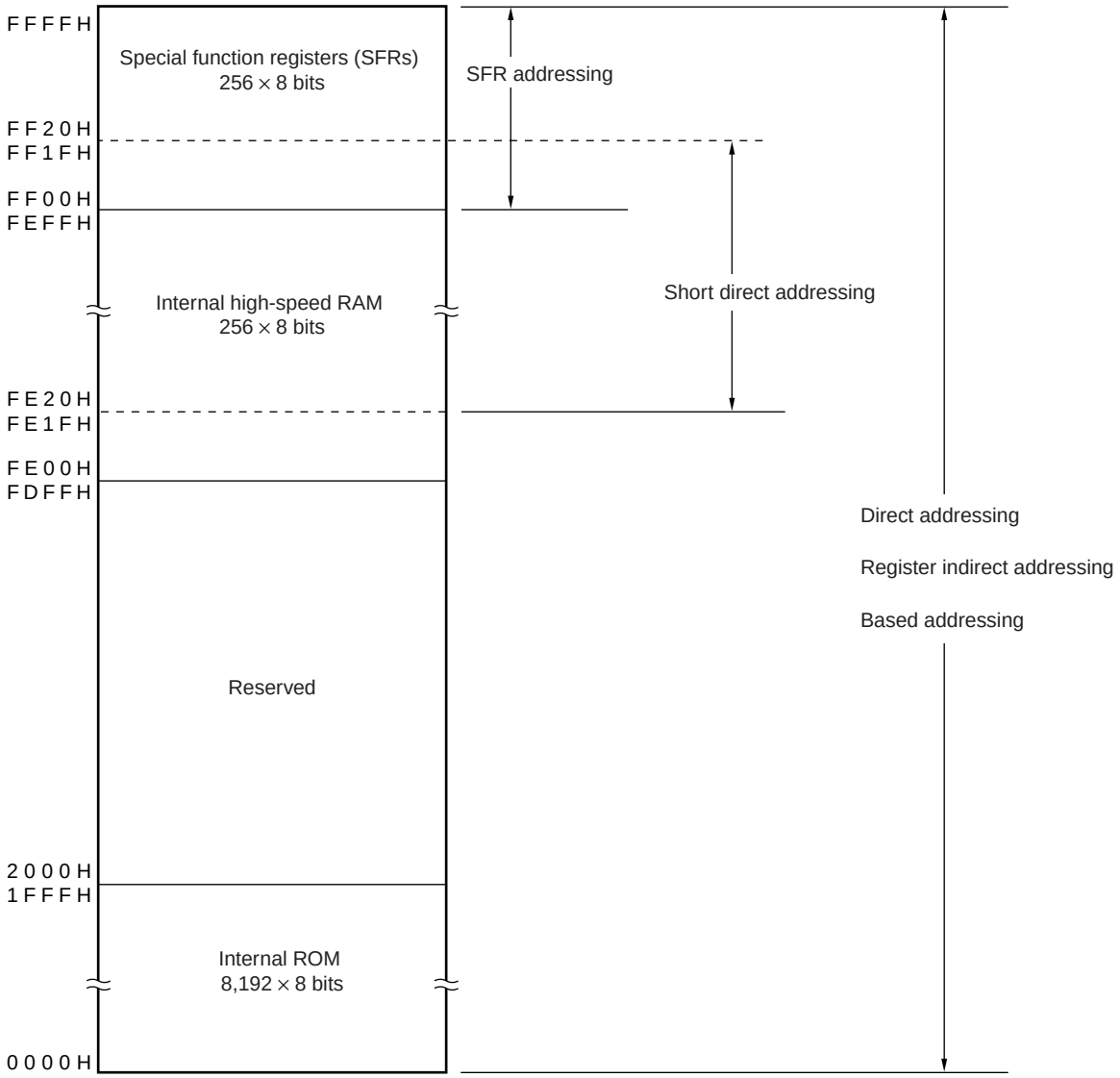
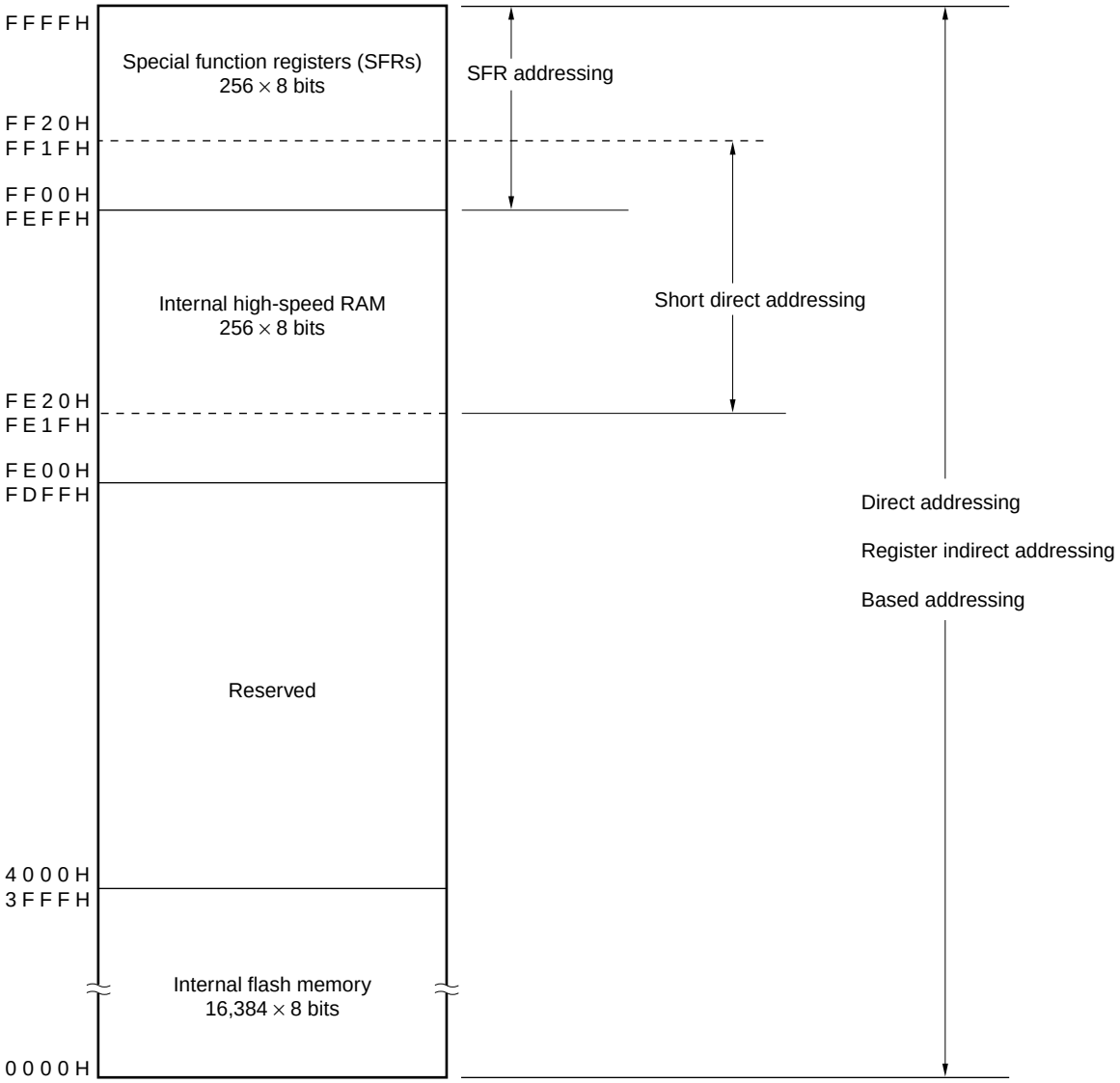


Figure 3-8. Data Memory Addressing (μPD78F9076)



3.2 Processor Registers

The μPD789074 Subseries provide the following on-chip processor registers:

3.2.1 Control registers

The control registers have special functions to control the program sequence statuses and stack memory. The control registers include a program counter, a program status word, and a stack pointer.

(1) Program counter (PC)

The program counter is a 16-bit register which holds the address information of the next program to be executed.

In normal operation, the PC is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data or register contents are set.

$\overline{\text{RESET}}$ input sets the reset vector table values at addresses 0000H and 0001H to the program counter.

Figure 3-9. Program Counter Configuration

	15															0
PC	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

(2) Program status word (PSW)

The program status word is an 8-bit register consisting of various flags to be set/reset by instruction execution.

Program status word contents are automatically stacked upon interrupt request generation or PUSH PSW instruction execution and are automatically restored upon execution of the RETI and POP PSW instructions.

$\overline{\text{RESET}}$ input sets PSW to 02H.

Figure 3-10. Program Status Word Configuration

	7							0
PSW	IE	Z	0	AC	0	0	1	CY

(a) Interrupt enable flag (IE)

This flag controls interrupt request acknowledge operations of the CPU.

When IE = 0, the interrupt disabled (DI) status is set. All interrupt requests except non-maskable interrupt are disabled.

When IE = 1, the interrupt enabled (EI) status is set. Interrupt request acknowledgment is controlled with an interrupt mask flag for various interrupt sources.

This flag is reset to 0 upon DI instruction execution or interrupt acknowledgment and is set to 1 upon EI instruction execution.

(b) Zero flag (Z)

When the operation result is zero, this flag is set to 1. It is reset to 0 in all other cases.

(c) Auxiliary carry flag (AC)

If the operation result has a carry from bit 3 or a borrow at bit 3, this flag is set to 1. It is reset to 0 in all other cases.

(d) Carry flag (CY)

This flag stores overflow and underflow that have occurred upon add/subtract instruction execution. It stores the shift-out value upon rotate instruction execution and functions as a bit accumulator during bit operation instruction execution.

(3) Stack pointer (SP)

This is a 16-bit register to hold the start address of the memory stack area. Only the internal high-speed RAM area can be set as the stack area.

Figure 3-11. Stack Pointer Configuration

	15															0
SP	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

The SP is decremented before writing (saving) to the stack memory and is incremented after reading (restoring) from the stack memory.

Each stack operation saves/restores data as shown in Figures 3-12 and 3-13.

Caution Since $\overline{\text{RESET}}$ input makes SP contents undefined, be sure to initialize the SP before instruction execution.

Figure 3-12. Data to Be Saved to Stack Memory

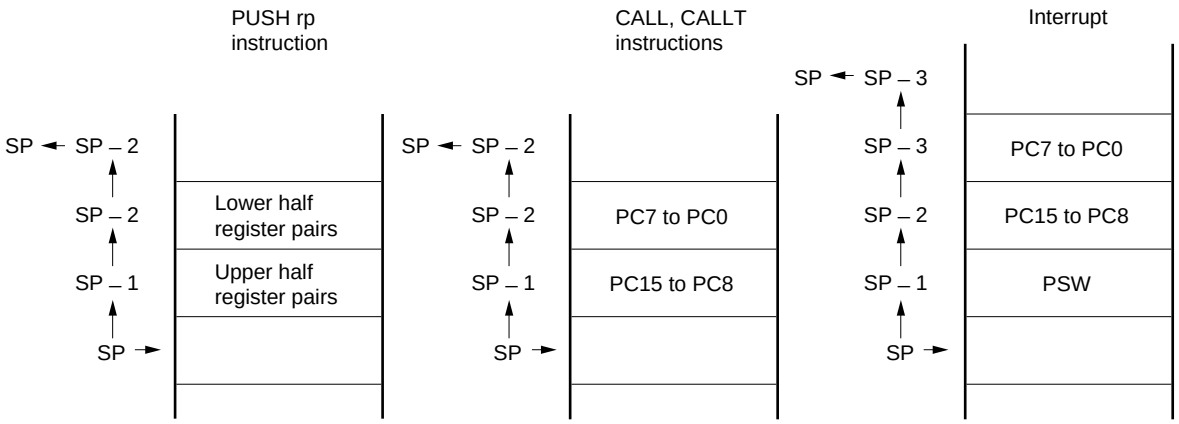
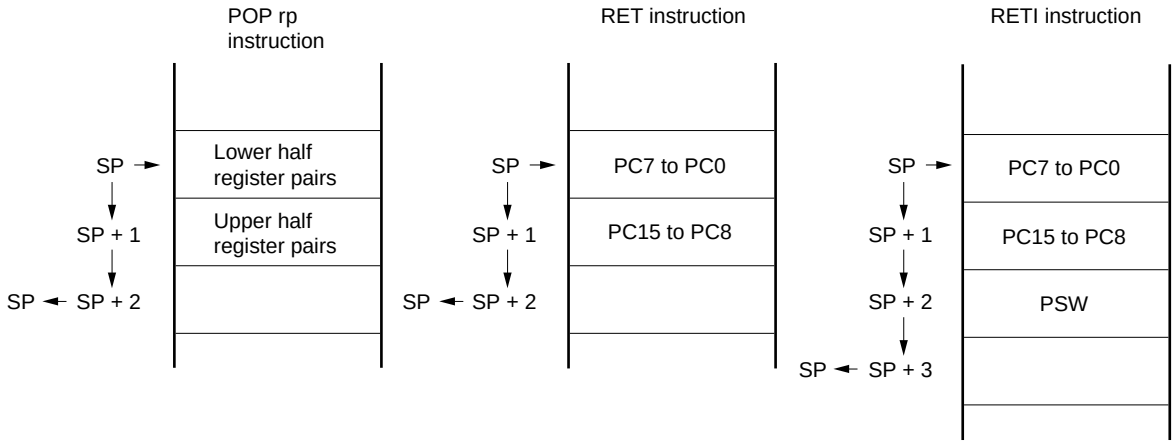


Figure 3-13. Data to Be Restored from Stack Memory



3.2.2 General-purpose registers

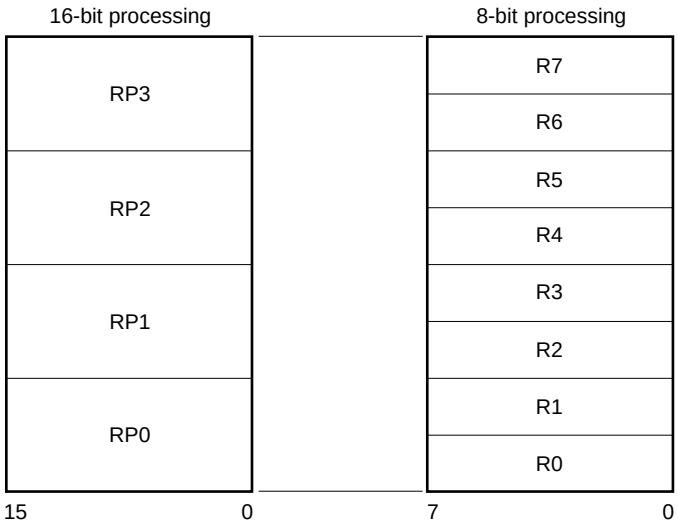
A general-purpose register consists of eight 8-bit registers (X, A, C, B, E, D, L, and H).

In addition each register being used as an 8-bit register, two 8-bit registers in pairs can be used as a 16-bit register (AX, BC, DE, and HL).

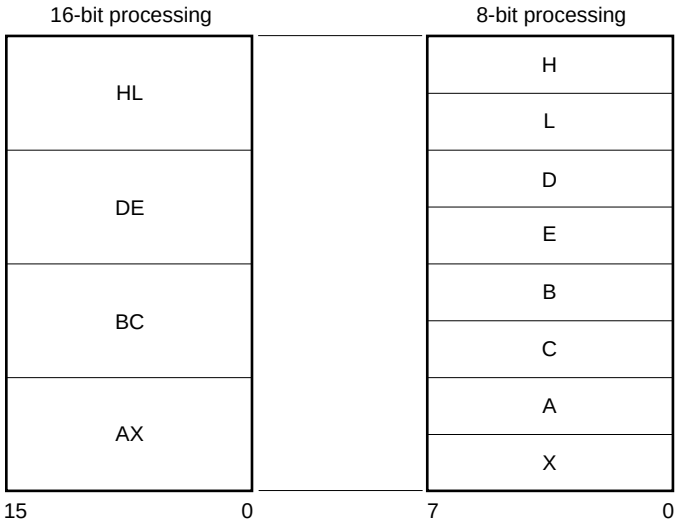
Registers can be described in terms of function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL) and absolute names (R0 to R7 and RP0 to RP3).

Figure 3-14. General-Purpose Register Configuration

(a) Absolute names



(b) Function names



3.2.3 Special function registers (SFRs)

Unlike the general-purpose registers, each special function register has a special function.

The special function registers are allocated to the 256-byte area FF00H to FFFFH.

The special function registers can be manipulated, like the general-purpose registers, with operation, transfer, and bit manipulation instructions. Manipulatable bit units (1, 8, and 16) differ depending on the special function register type.

Each manipulation bit unit can be specified as follows.

- 1-bit manipulation
  Describes a symbol reserved by the assembler for the 1-bit manipulation instruction operand (sfr.bit). This manipulation can also be specified with an address.

- 8-bit manipulation
  Describes a symbol reserved by the assembler for the 8-bit manipulation instruction operand (sfr). This manipulation can also be specified with an address.

- 16-bit manipulation
  Describes a symbol reserved by the assembler for the 16-bit manipulation instruction operand. When specifying an address, describe an even address.

Table 3-3 lists the special function registers. The meanings of the symbols in this table are as follows:

- Symbol
  Indicates the addresses of the implemented special function registers. The symbols shown in this column are reserved words in the assembler, and have already been defined in a header file called "sfrbit.h" in the C compiler. Therefore, these symbols can be used as instruction operands if an assembler or integrated debugger is used.

- R/W
  Indicates whether the special function register can be read or written.
  R/W: Read/write
  R: Read only
  W: Write only

- Number of bits manipulated simultaneously
  Indicates the bit units (1, 8, and 16) in which the special function register can be manipulated.

- After reset
  Indicates the status of the special function register when the $\overline{\text{RESET}}$ signal is input.

Table 3-3. Special Function Registers (1/2)

Address	Special Function Register (SFR) Name	Symbol	R/W	Number of Bits Manipulated Simultaneously			After Reset
				1 Bit	8 Bits	16 Bits	
FF00H	Port 0	P0	R/W	√	√	–	00H
FF01H	Port 1	P1		√	√	–	
FF02H	Port 2	P2		√	√	–	
FF03H	Port 3	P3		√	√	–	
FF16H	16-bit compare register 90	CR90Note 1	W	–	√Note 2	√Note 3	FFFFH
FF17H							
FF18H	16-bit timer counter 90	TM90Note 1	R	–	√Note 2	√Note 3	0000H
FF19H							
FF1AH	16-bit capture register 90	TCP90Note 1		–	√Note 2	√Note 3	Undefined
FF1BH							
FF20H	Port mode register 0	PM0	R/W	√	√	–	FFH
FF21H	Port mode register 1	PM1		√	√	–	
FF22H	Port mode register 2	PM2		√	√	–	
FF23H	Port mode register 3	PM3		√	√	–	
FF32H	Pull-up resistor option register B2	PUB2		√	√	–	00H
FF42H	Watchdog timer clock selection register 2	WDCS		–	√	–	
FF48H	16-bit timer mode control register 90	TMC90		√	√	–	
FF49H	Buzzer output control register 90	BZC90		√	√	–	
FF50H	8-bit compare register 80	CR80	W	–	√	–	Undefined
FF51H	8-bit timer counter 80	TM80	R	–	√	–	00H
FF53H	8-bit timer mode control register 80	TMC80	R/W	√	√	–	
FF70H	Asynchronous serial interface mode register 20	ASIM20		√	√	–	
FF71H	Asynchronous serial interface status register 20	ASIS20	R	√	√	–	
FF72H	Serial operation mode register 20	CSIM20	R/W	√	√	–	
FF73H	Baud rate generator control register 20	BRGC20		–	√	–	

Notes 1. These SFRs are for 16-bit access only.

2. CR90, TM90, and TCP90 are designed only for 16-bit access. In direct addressing, however, 8-bit access can also be performed.

3. 16-bit access is allowed only in short direct addressing.

Table 3-3. Special Function Registers (2/2)

Address	Special Function Register (SFR) Name	Symbol	R/W	Number of Bits Manipulated Simultaneously			After Reset
				1 Bit	8 Bits	16 Bits	
FF74H	Transmission shift register 20	TXS20 SIO20	W	–	√	–	FFH
	Receive buffer register 20	RXB20	R	–	√	–	Undefined
FFE0H	Interrupt request flag register 0	IF0	R/W	√	√	–	00H
FFE1H	Interrupt request flag register 1	IF1		√	√	–	
FFE4H	Interrupt mask flag register 0	MK0		√	√	–	FFH
FFE5H	Interrupt mask flag register 1	MK1		√	√	–	
FFECH	External interrupt mode register 0	INTM0		–	√	–	00H
FFF7H	Pull-up resistor option register 0	PU0		√	√	–	
FFF9H	Watchdog timer mode register	WDTM		√	√	–	
FFFAH	Oscillation stabilization time selection register	OSTS		–	√	–	04H
FFFBH	Processor clock control register	PCC		√	√	–	02H

3.3 Instruction Address Addressing

An instruction address is determined by the program counter (PC) contents. The PC contents are normally incremented (+1 for each byte) automatically according to the number of bytes of an instruction to be fetched each time another instruction is executed. When a branch instruction is executed, the branch destination address information is set to the PC to branch by the following addressing (for details of each instruction, refer to **78K/0S Series User's Manual Instructions (U11047E)**).

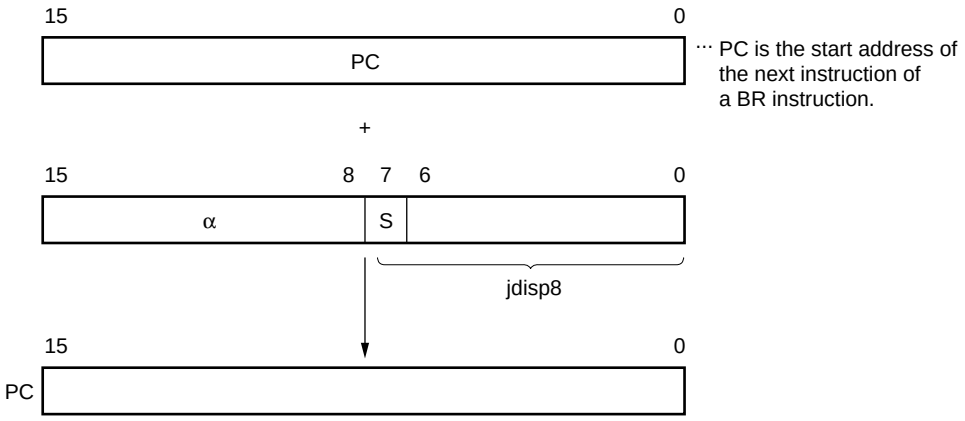
3.3.1 Relative addressing

[Function]

The value obtained by adding 8-bit immediate data (displacement value: jdisp8) of an instruction code to the start address of the following instruction is transferred to the program counter (PC) to branch. The displacement value is treated as signed two's complement data (–128 to +127) and bit 7 becomes the sign bit. In other words, the range of branch in relative addressing is between –128 and +127 of the start address of the following instruction.

This function is carried out when the BR $addr16 instruction or a conditional branch instruction is executed.

[Illustration]



When S = 0, α indicates that all bits are "0".
When S = 1, α indicates that all bits are "1".

3.3.2 Immediate addressing

[Function]

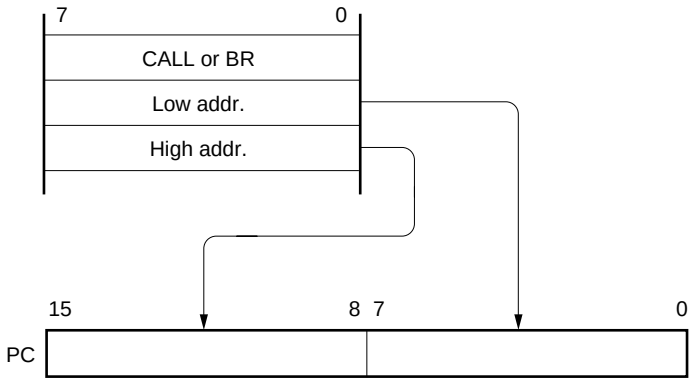
Immediate data in the instruction word is transferred to the program counter (PC) to branch.

This function is carried out when the CALL !addr16 and BR !addr16 instructions are executed.

CALL !addr16 and BR !addr16 instructions can be used to branch to all the memory spaces.

[Illustration]

In case of CALL !addr16 and BR !addr16 instructions



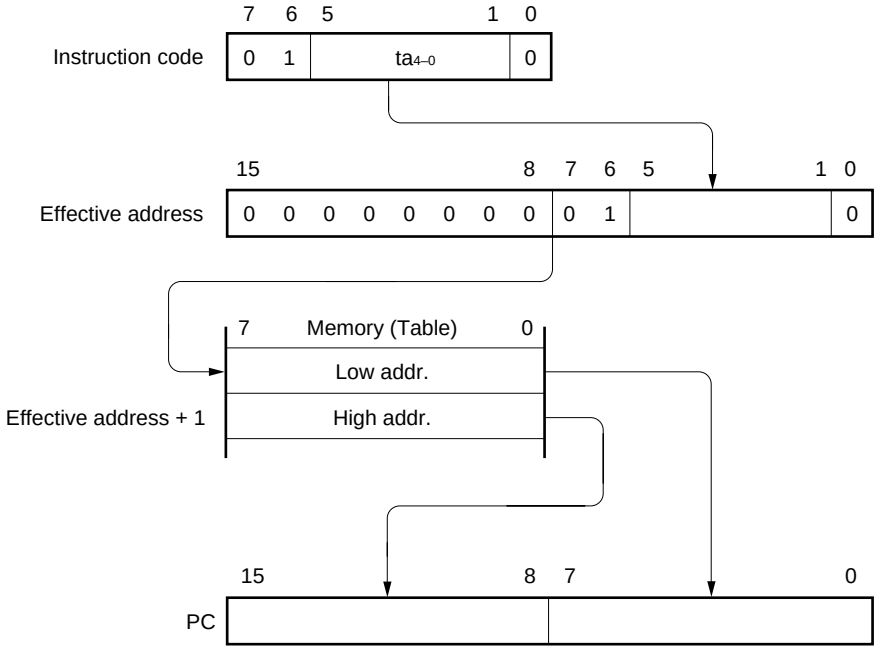
3.3.3 Table indirect addressing

[Function]

The table contents (branch destination address) of the particular location to be addressed by the immediate data of an instruction code from bit 1 to bit 5 are transferred to the program counter (PC) to branch.

Table indirect addressing is carried out when the CALLT [addr5] instruction is executed. This instruction can be used to branch to all the memory spaces according to the address stored in the memory table 40H to 7FH.

[Illustration]



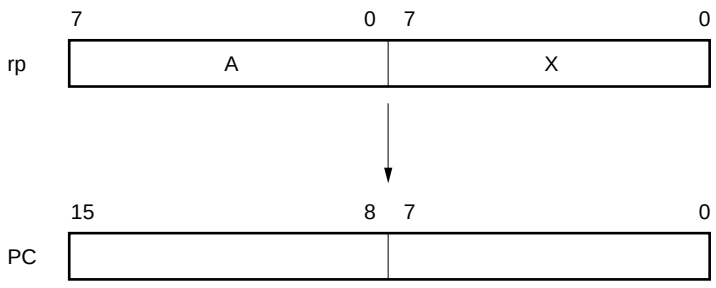
3.3.4 Register addressing

[Function]

The register pair (AX) contents to be specified with an instruction word are transferred to the program counter (PC) to branch.

This function is carried out when the BR AX instruction is executed.

[Illustration]



3.4 Operand Address Addressing

The following methods (addressing) are available to specify the register and memory to undergo manipulation during instruction execution.

3.4.1 Direct addressing

[Function]

The memory indicated by immediate data in an instruction word is directly addressed.

[Operand format]

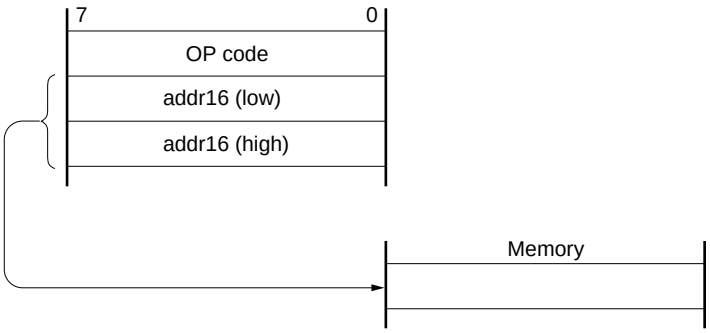
Identifier	Description
addr16	Label or 16-bit immediate data

[Description example]

MOV A, !FE00H; When setting !addr16 to FE00H

Instruction code	0	0	1	0	1	0	0	1	OP Code
	0	0	0	0	0	0	0	0	00H
	1	1	1	1	1	1	1	0	FEH

[Illustration]



3.4.2 Short direct addressing

[Function]

The memory to be manipulated in the fixed space is directly addressed with the 8-bit data in an instruction word.

The fixed space where this addressing is applied is the 256-byte space FE20H to FF1FH. An internal high-speed RAM is mapped at FE20H to FEFFH and the special function registers (SFR) are mapped at FF00H to FF1FH.

The SFR area where short direct addressing is applied (FF00H to FF1FH) is a part of the total SFR area. In this area, ports which are frequently accessed in a program and a compare register of the timer counter are mapped, and these SFRs can be manipulated with a small number of bytes and clocks.

When 8-bit immediate data is at 20H to FFH, bit 8 of an effective address is set to 0. When it is at 00H to 1FH, bit 8 is set to 1. See [Illustration] below.

[Operand format]

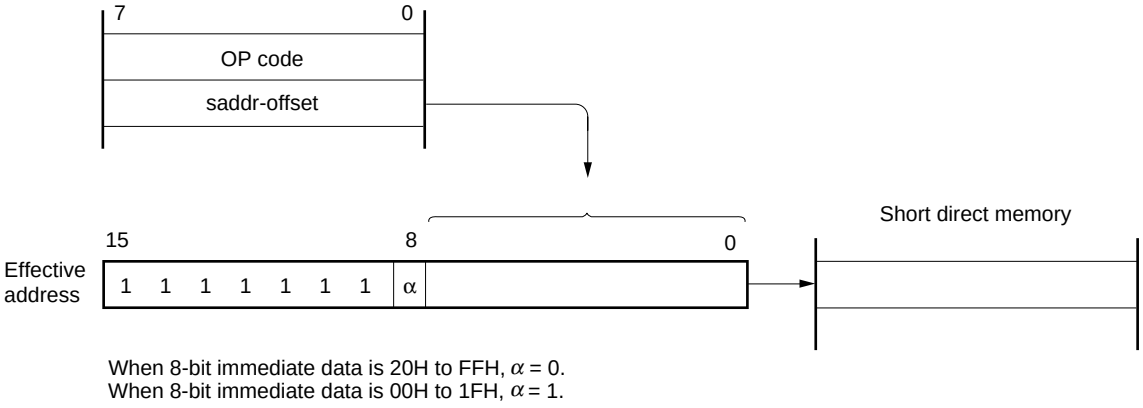
Identifier	Description
saddr	Label or FE20H to FF1FH immediate data
saddrp	Label or FE20H to FF1FH immediate data (even address only)

[Description example]

MOV FE90H, #50H; When setting saddr to FE90H and the immediate data to 50H

Instruction code	1	1	1	1	0	1	0	1	OP code
	1	0	0	1	0	0	0	0	90H (saddr-offset)
	0	1	0	1	0	0	0	0	50H (immediate data)

[Illustration]



When 8-bit immediate data is 20H to FFH, $\alpha = 0$.
When 8-bit immediate data is 00H to 1FH, $\alpha = 1$.

3.4.3 Special function register (SFR) addressing

[Function]

A memory-mapped special function register (SFR) is addressed with the 8-bit immediate data in an instruction word.

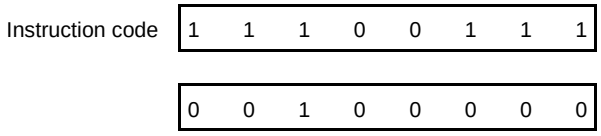
This addressing is applied to the 256-byte spaces FF00H to FFFFH and FFE0H to FFFFH. However, SFRs mapped at FF00H to FF1FH can also be accessed with short direct addressing.

[Operand format]

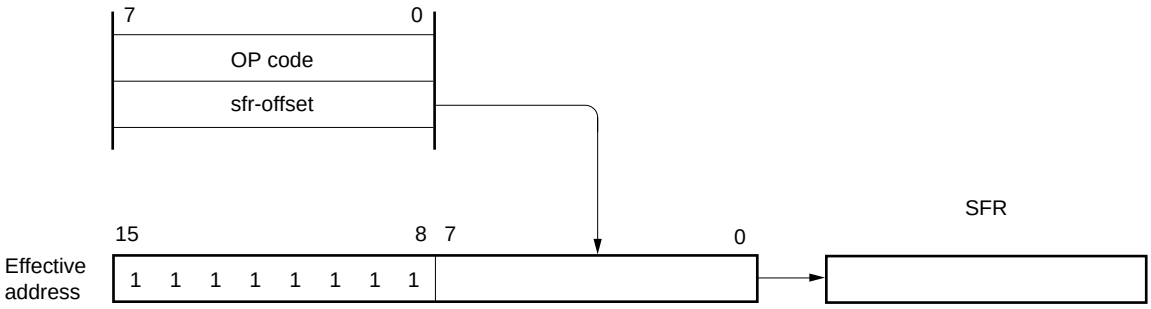
Identifier	Description
sfr	Special function register name

[Description example]

MOV PM0, A; When selecting PM0 for sfr



[Illustration]



3.4.4 Register addressing

[Function]

A general-purpose register is accessed as an operand.

The general-purpose register to be accessed is specified with the register specify code and functional name in the instruction code.

Register addressing is carried out when an instruction with the following operand format is executed. When an 8-bit register is specified, one of the eight registers is specified with 3 bits in the instruction code.

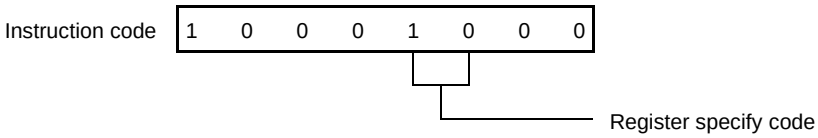
[Operand format]

Identifier	Description
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

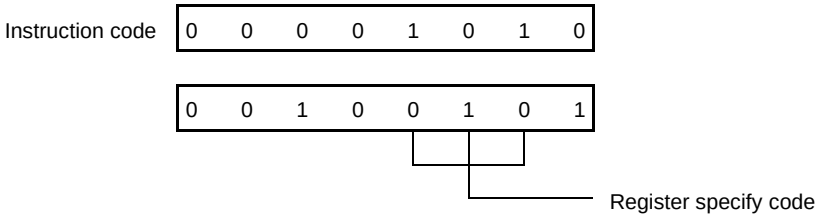
'r' and 'rp' can be described with absolute names (R0 to R7 and RP0 to RP3) as well as function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL).

[Description example]

MOV A, C; When selecting the C register for r



INCW DE; When selecting the DE register pair for rp



3.4.5 Register indirect addressing

[Function]

The memory is addressed with the contents of the register pair specified as an operand. The register pair to be accessed is specified with the register pair specify code in the instruction code. This addressing can be carried out for all the memory spaces.

[Operand format]

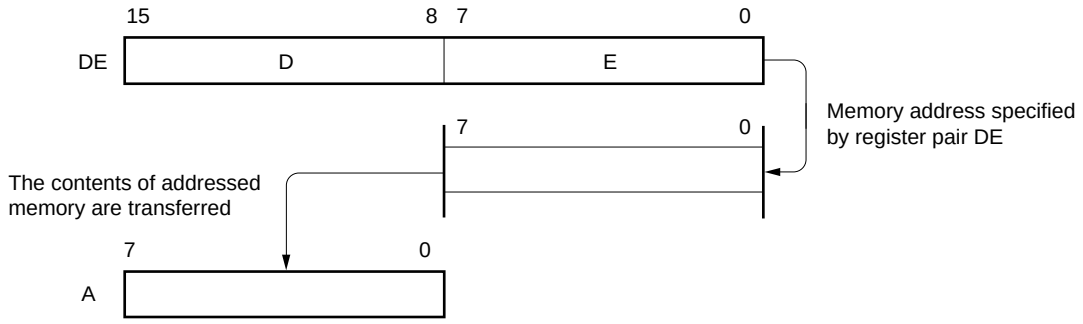
Identifier	Description
–	[DE], [HL]

[Description example]

MOV A, [DE]; When selecting register pair [DE]

Instruction code	0	0	1	0	1	0	1	1

[Illustration]



3.4.6 Based addressing

[Function]

8-bit immediate data is added to the contents of the base register, that is, the HL register pair, and the sum is used to address the memory. Addition is performed by expanding the offset data as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

[Operand format]

Identifier	Description
–	[HL+byte]

[Description example]

MOV A, [HL+10H]; When setting byte to 10H

Instruction code	0	0	1	0	1	1	0	1
	0	0	0	1	0	0	0	0

3.4.7 Stack addressing

[Function]

The stack area is indirectly addressed with the stack pointer (SP) contents.

This addressing method is automatically employed when the PUSH, POP, subroutine call, and RETURN instructions are executed or the register is saved/restored upon interrupt request generation.

Stack addressing can be used to access the internal high-speed RAM area only.

[Description example]

In the case of PUSH DE

Instruction code	1	0	1	0	1	0	1	0

[MEMO]

CHAPTER 4 PORT FUNCTIONS

4.1 Port Functions

The μPD789074 Subseries is provided with the ports shown in Figure 4-1. These ports enable several types of control. Table 4-1 lists the functions of each port.

These ports, while originally designed as digital input/output ports, have alternate functions. For the alternate functions, refer to **2.1 Pin Function List**.

Figure 4-1. Port Types

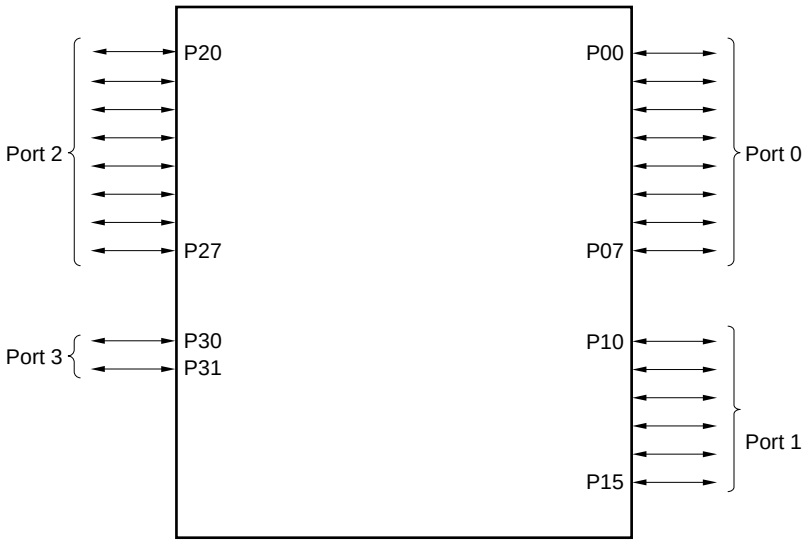


Table 4-1. Port Functions

Pin Name	I/O	Function	After Reset	Alternate Function
P00 to P07	I/O	Port 0 8-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	–
P10 to P15	I/O	Port 1 6-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	–
P20	I/O	Port 2 8-bit I/O port Input/output can be specified in 1-bit units. An on-chip pull-up resistor can be specified by means of pull-up resistor option register B2 (PUB2).	Input	$\overline{\text{SCK20}}$/ASCK20
P21				SO20/TxD20
P22				SI20/RxD20
P23				$\overline{\text{SS20}}$
P24				INTP0
P25				INTP1
P26				INTP2/CPT90
P27				TI80/TO80
P30	I/O	Port 3 2-bit I/O port Input/output can be specified in 1-bit units. When used as an input port, an on-chip pull-up resistor can be specified by means of the pull-up resistor option register 0 (PU0).	Input	TO90
P31				BZO90

4.2 Port Configuration

Ports include the following hardware.

Table 4-2. Configuration of Port

Parameter	Configuration
Control registers	Port mode registers (PMm: m = 0 to 3) Pull-up resistor option register 0 (PU0) Pull-up resistor option register B2 (PUB2)
Ports	CMOS I/O: 24
Pull-up resistors	Software control: 24

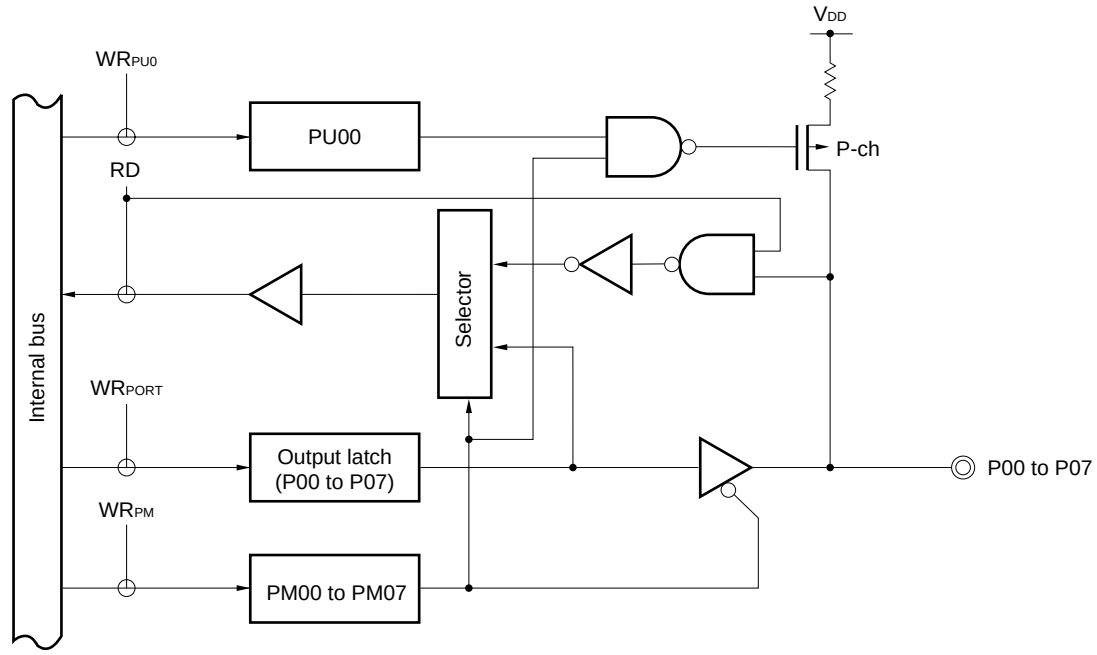
4.2.1 Port 0

This is an 8-bit I/O port with an output latch. Port 0 can be set to input or output mode in 1-bit units by using port mode register 0 (PM0). When pins P00 to P07 are used as input port pins, on-chip pull-up resistors can be connected in 8-bit units by using pull-up resistor option register 0 (PU0).

$\overline{\text{RESET}}$ input sets port 0 to input mode.

Figure 4-2 shows a block diagram of port 0.

Figure 4-2. Block Diagram of P00 to P07



PU0: Pull-up resistor option register 0
PM: Port mode register
RD: Port 0 read signal
WR: Port 0 write signal

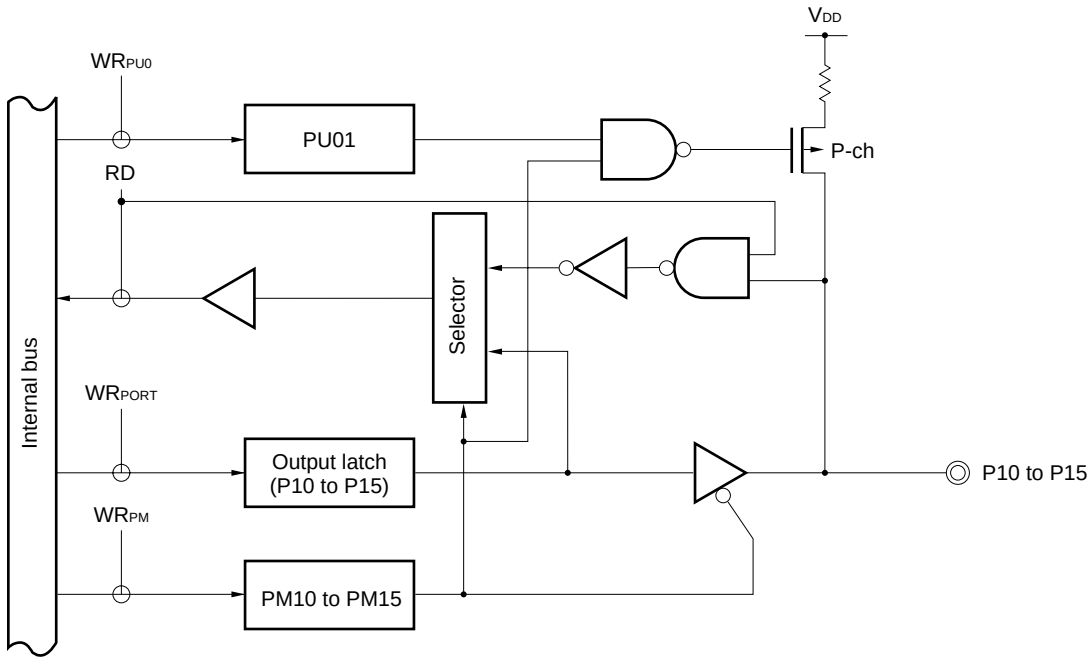
4.2.2 Port 1

This is a 6-bit I/O port with an output latch. Port 1 can be set to input or output mode in 1-bit units by using port mode register 1 (PM1). When the P10 to P15 pins are used as input port pins, on-chip pull-up resistors can be connected in 6-bit units by using pull-up resistor option register 0 (PU0).

$\overline{\text{RESET}}$ input sets port 1 to input mode.

Figure 4-3 shows a block diagram of port 1.

Figure 4-3. Block Diagram of P10 to P15



PU0: Pull-up resistor option register 0
PM: Port mode register
RD: Port 1 read signal
WR: Port 1 write signal

4.2.3 Port 2

This is an 8-bit I/O port with output latches. Port 2 can be set to input or output mode in 1-bit units by using port mode register 2 (PM2). For pins P20 to P27, on-chip pull-up resistors can be connected in 1-bit units by using pull-up resistor option register B2 (PUB2).

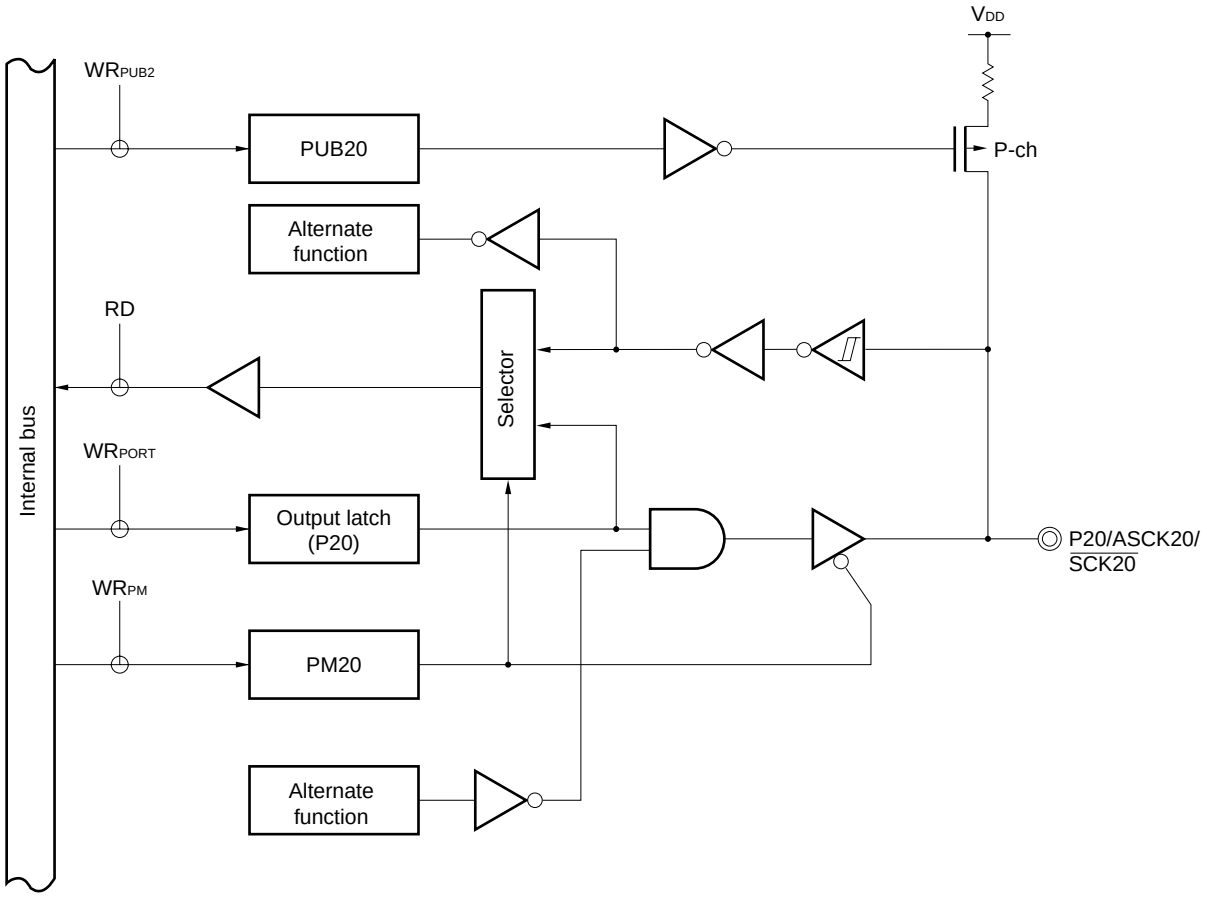
The port is also used as external interrupt input, serial interface I/O, and timer I/O.

$\overline{\text{RESET}}$ input sets port 2 to input mode.

Figures 4-4 through 4-7 show block diagrams of port 2.

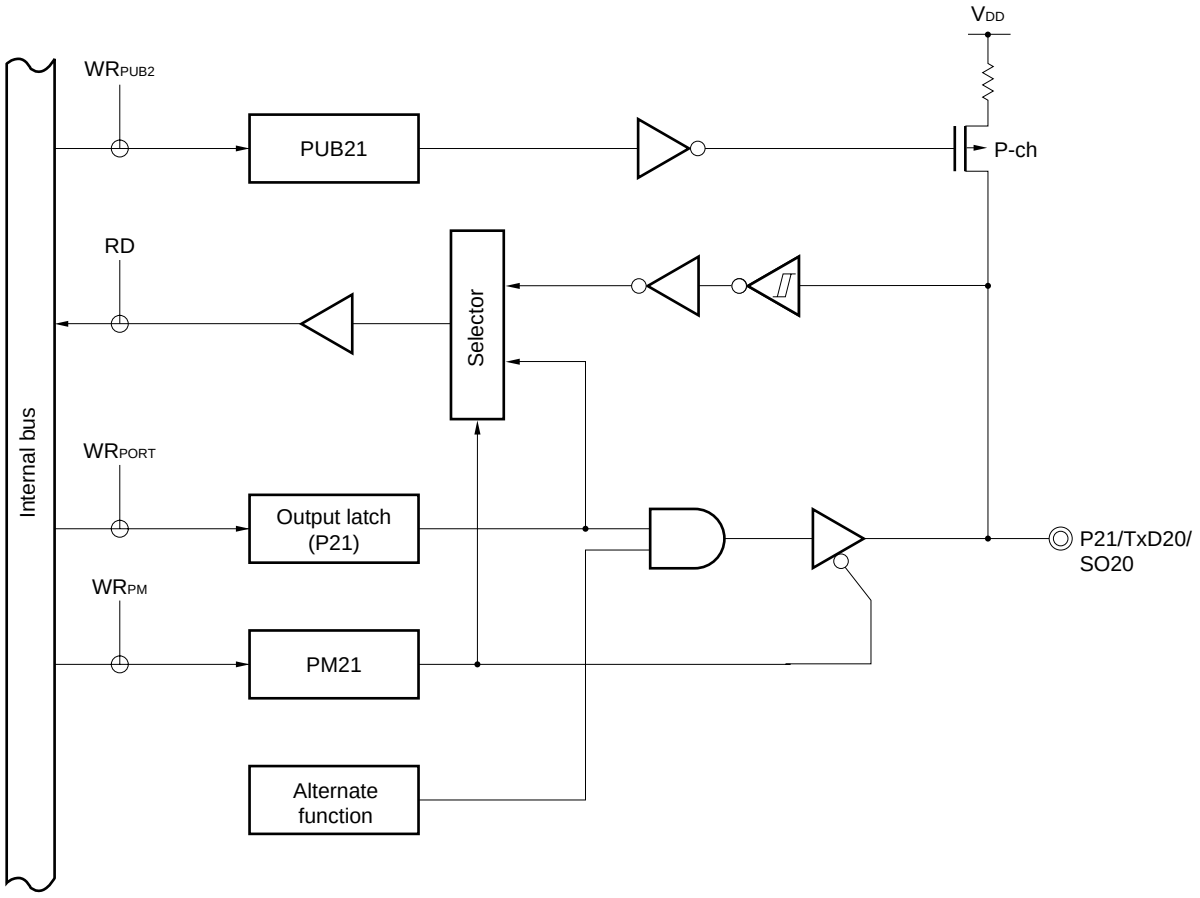
Caution When using the pins of port 2 for the serial interface, the I/O and output latches must be set according to the function to be used. For details of the settings, see Table 9-2 Serial Interface 20 Operation Mode Settings.

Figure 4-4. Block Diagram of P20



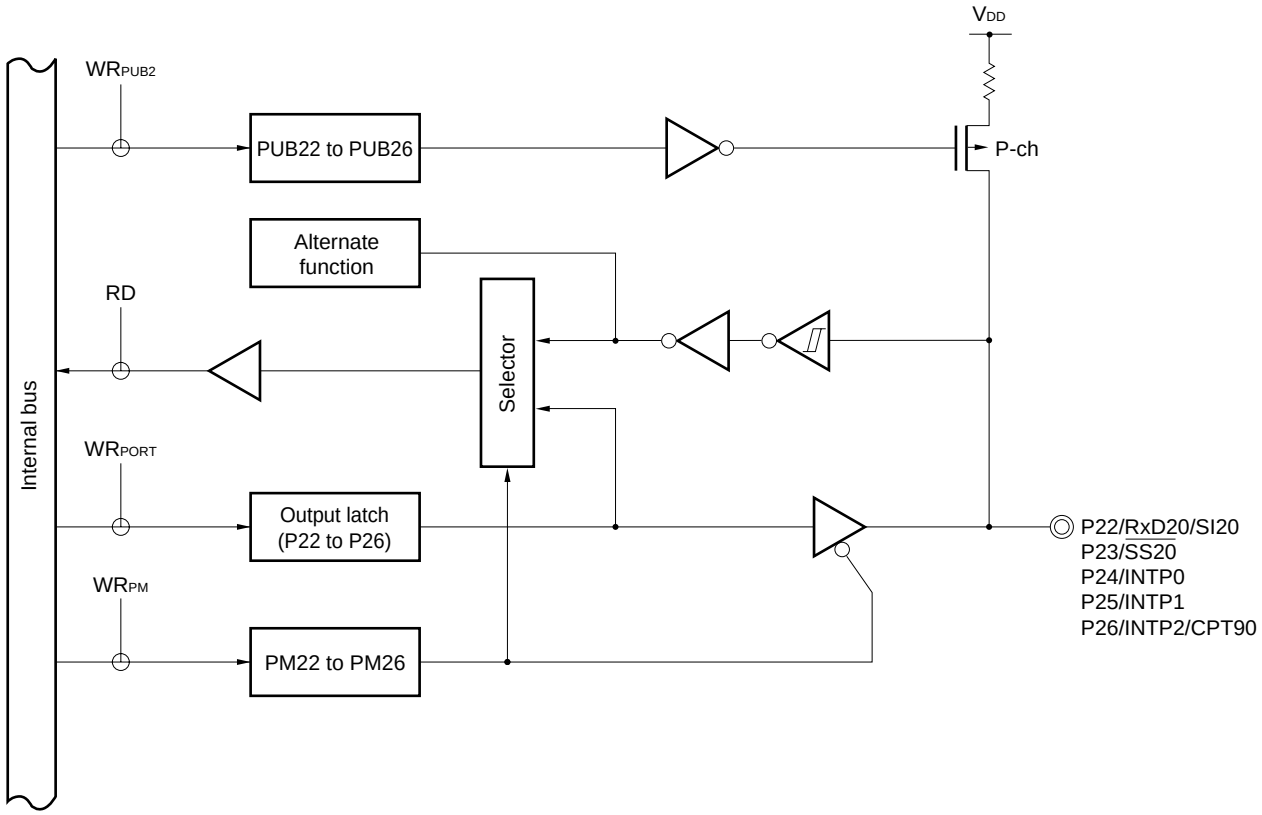
PUB2: Pull-up resistor option register B2
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

Figure 4-5. Block Diagram of P21



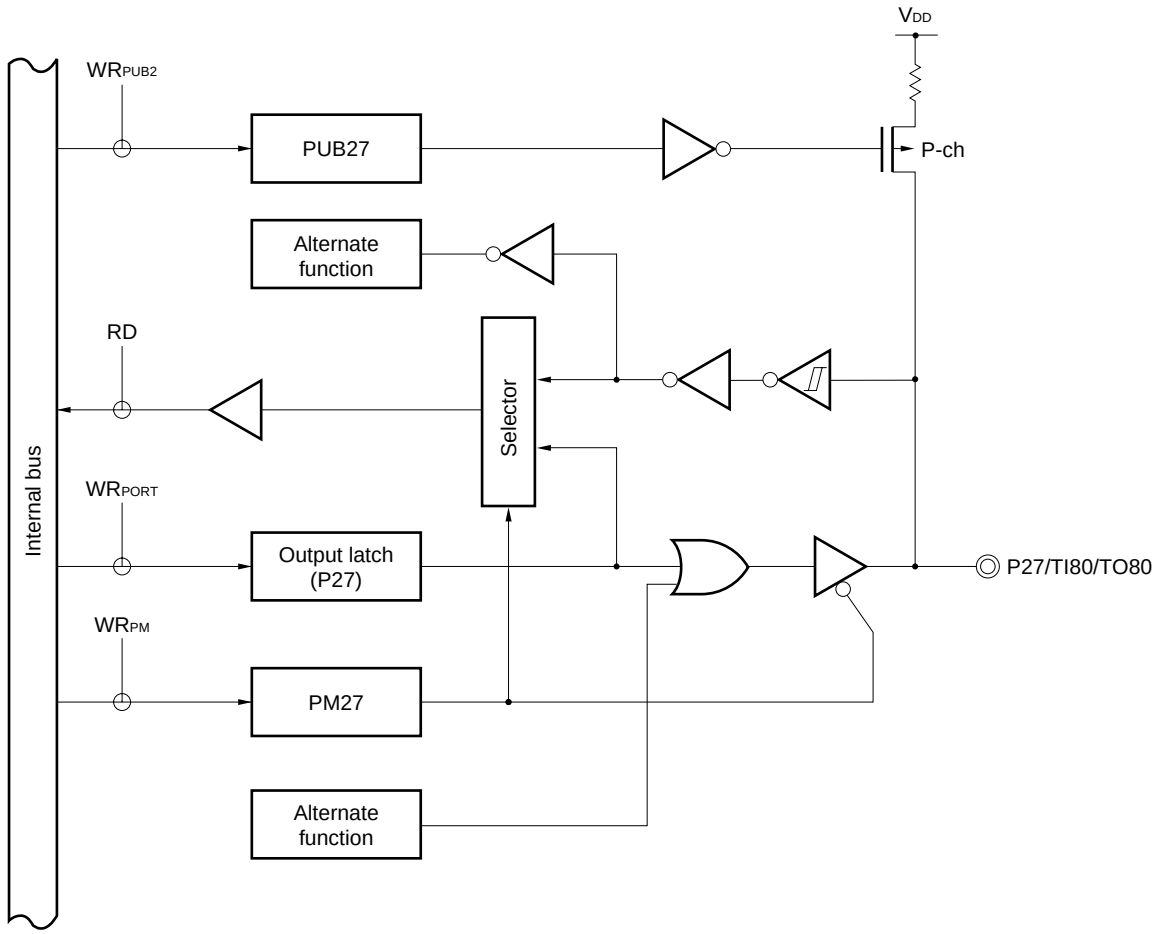
PUB2: Pull-up resistor option register B2
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

Figure 4-6. Block Diagram of P22 to P26



PUB2: Pull-up resistor option register B2
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

Figure 4-7. Block Diagram of P27



PUB2: Pull-up resistor option register B2
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

4.2.4 Port 3

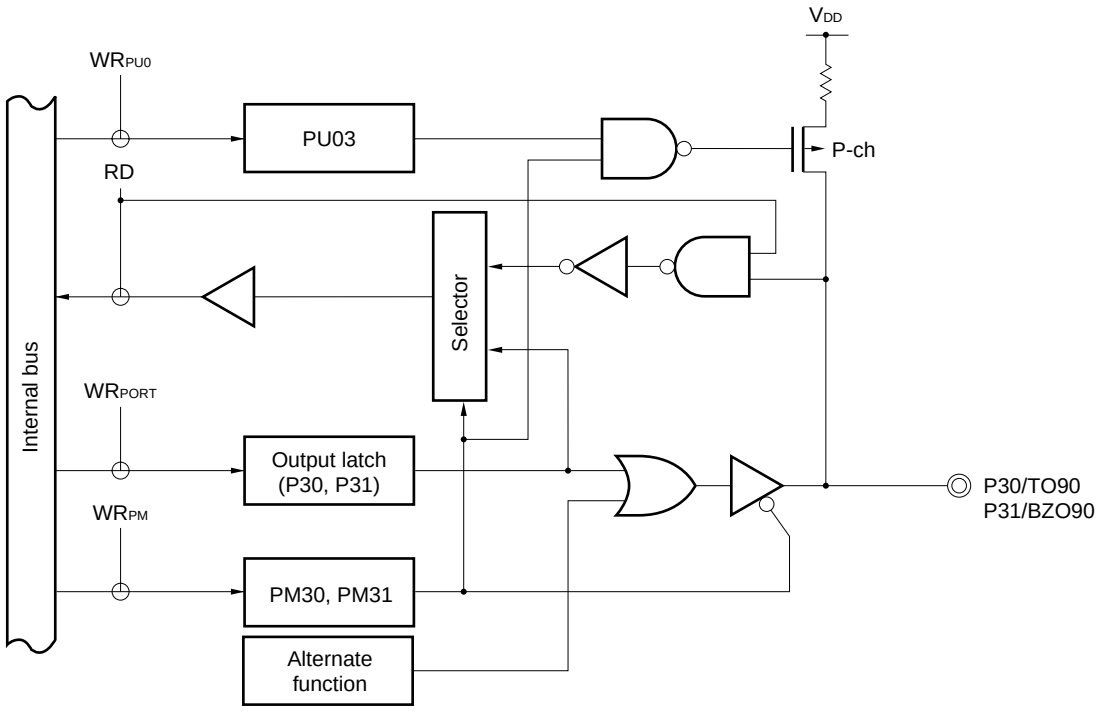
This is a 2-bit I/O port with an output latch. Port 3 can be set to input or output mode in 1-bit units by using port mode register 3 (PM3). When P30 and P31 are used as input port pins, on-chip pull-up resistors can be connected in 2-bit units by using pull-up resistor option register 0 (PU0).

The port is also used as timer output and buzzer output.

$\overline{\text{RESET}}$ input sets port 3 to input mode.

Figure 4-9 shows a block diagram of port 3.

Figure 4-8. Block Diagram of P30 and P31



PU0: Pull-up resistor option register 0

PM: Port mode register

RD: Port 3 read signal

WR: Port 3 write signal

4.3 Port Function Control Registers

The following two types of registers are used to control the ports.

- Port mode registers (PM0 to PM3)
- Pull-up resistor option registers (PU0 and PUB2)

(1) Port mode registers (PM0 to PM3)

The port mode registers separately set each port bit to either input or output.

Each port mode register is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets the port mode registers to FFH.

When port pins are used for alternate functions, the corresponding port mode register and output latch must be set or reset as described in Table 4-3.

Caution When port 2 is acting as an output port, and its output level is changed, an interrupt request flag is set, because this port is also used as the input for an external interrupt. To use port 2 in output mode, therefore, the interrupt mask flag must be set to 1 in advance.

Figure 4-9. Format of Port Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00	FF20H	FFH	R/W
PM1	1	1	PM15	PM14	PM13	PM12	PM11	PM10	FF21H	FFH	R/W
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20	FF22H	FFH	R/W
PM3	1	1	1	1	1	1	PM31	PM30	FF23H	FFH	R/W

PMmn	Pmn pin input/output mode selection (m = 0 to 3, n = 0 to 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Table 4-3. Port Mode Register and Output Latch Settings for Using Alternate Functions

Pin Name	Alternate Function		PM××	P××
	Name	Input/Output		
P24	INTP0	Input	1	×
P25	INTP1	Input	1	×
P26	INTP2	Input	1	×
	CPT90	Input	1	×
P27	TI80	Input	1	×
	TO80	Output	0	0
P30	TO90	Output	0	0
P31	BZO90	Output	0	0

Caution When using the pins of port 2 for the serial interface, the I/O or output latch must be set according to the function to be used. For details of the settings, see Table 9-2 Serial Interface 20 Operation Mode Settings.

Remark ×: don't care
PM××: Port mode register
P××: Port output latch

(2) Pull-up resistor option register 0 (PU0)

Pull-up resistor option register 0 (PU0) sets whether an on-chip pull-up resistor is used on port 0, 1, or 3. For ports specified by PU0 to use on-chip pull-up resistors, pull-up resistors can be internally used only for the bits set to input mode. No on-chip pull-up resistors can be used for the bits set to output mode regardless of the setting of PU0. On-chip pull-up resistors cannot be used even when the pins are used as the alternate-function output pins.

PU0 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears PU0 to 00H.

Figure 4-10. Format of Pull-up Resistor Option Register 0

Symbol	7	6	5	4	<3>	2	<1>	<0>	Address	After reset	R/W
PU0	0	0	0	0	PU03	0	PU01	PU00	FFF7H	00H	R/W

PU0m	Pm on-chip pull-up resistor selection (m = 0, 1, 3)
0	On-chip pull-up resistor is not used.
1	On-chip pull-up resistor is used.

Caution Bits 2 and 4 to 7 must all be set to 0.

(3) Pull-up resistor option register B2 (PUB2)

This register specifies whether an on-chip pull-up resistor connected to each pin of port 2 is used. The pins for which use of an on-chip pull-up resistor is specified by PUB2 can use a pull-up register internally, regardless of the setting of the port mode register.

PUB2 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears PUB2 to 00H.

Figure 4-11. Format of Pull-up Resistor Option Register B2

Symbol	<7>	<6>	<5>	<4>	<3>	<2>	<1>	<0>	Address	After reset	R/W
PUB2	PUB27	PUB26	PUB25	PUB24	PUB23	PUB22	PUB21	PUB20	FF32H	00H	R/W

PUB2n	P2n on-chip pull-up resistor selection (n = 0 to 7)
0	On-chip pull-up resistor is not used.
1	On-chip pull-up resistor is used.

4.4 Operation of Port Functions

The operation of a port differs depending on whether the port is set to input or output mode, as described below.

4.4.1 Writing to I/O port

(1) In output mode

A value can be written to the output latch of a port by using a transfer instruction. The contents of the output latch can be output from the pins of the port.

The data once written to the output latch is retained until new data is written to the output latch.

(2) In input mode

A value can be written to the output latch by using a transfer instruction. However, the status of the port pin is not changed because the output buffer is OFF.

The data once written to the output latch is retained until new data is written to the output latch.

Caution A 1-bit memory manipulation instruction is executed to manipulate one bit of a port. However, this instruction accesses the port in 8-bit units. When this instruction is executed to manipulate a bit of a port consisting both of inputs and outputs, therefore, the contents of the output latch of the pin that is set to input mode and not subject to manipulation become undefined.

4.4.2 Reading from I/O port

(1) In output mode

The contents of the output latch can be read by using a transfer instruction. The contents of the output latch are not changed.

(2) In input mode

The status of a pin can be read by using a transfer instruction. The contents of the output latch are not changed.

4.4.3 Arithmetic operation of I/O port

(1) In output mode

An arithmetic operation can be performed with the contents of the output latch. The result of the operation is written to the output latch. The contents of the output latch are output from the port pins.

The data once written to the output latch is retained until new data is written to the output latch.

(2) In input mode

The contents of the output latch become undefined. However, the status of the pin is not changed because the output buffer is OFF.

Caution A 1-bit memory manipulation instruction is executed to manipulate one bit of a port. However, this instruction accesses the port in 8-bit units. When this instruction is executed to manipulate a bit of a port consisting both of inputs and outputs, therefore, the contents of the output latch of the pin that is set to input mode and not subject to manipulation become undefined.

[MEMO]

CHAPTER 5 CLOCK GENERATOR

5.1 Clock Generator Functions

The clock generator generates the clock to be supplied to the CPU and peripheral hardware.
The following type of system clock oscillator is used.

- **System clock oscillator**
  This circuit oscillates at 1.0 to 5.0 MHz. Oscillation can be stopped by executing the STOP instruction.

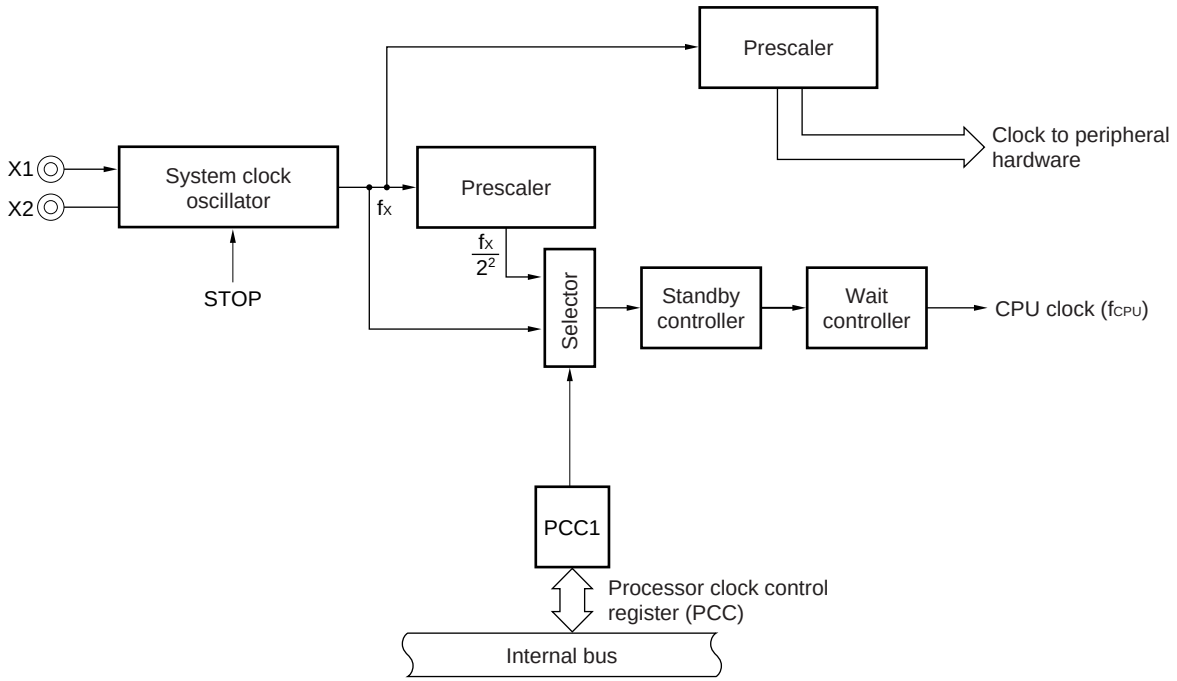
5.2 Clock Generator Configuration

The clock generator includes the following hardware.

Table 5-1. Configuration of Clock Generator

Item	Configuration
Control register	Processor clock control register (PCC)
Oscillator	Crystal/ceramic oscillator

Figure 5-1. Block Diagram of Clock Generator



5.3 Clock Generator Control Register

The clock generator is controlled by the following register:

- Processor clock control register (PCC)

(1) Processor clock control register (PCC)

PCC selects the CPU clock and the division ratio.

PCC is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PCC to 02H.

Figure 5-2. Format of Processor Clock Control Register

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PCC	0	0	0	0	0	0	PCC1	0	FFFBH	02H	R/W

PCC1	CPU clock (f_{CPU}) selection
0	f_X (0.2 μs)
1	$f_X/2^2$ (0.8 μs)

Caution Bits 0 and 2 to 7 must all be set to 0.

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

3. Minimum instruction execution time: $2f_{CPU}$

- f_{CPU} = 0.2 μs: 0.4 μs
- f_{CPU} = 0.8 μs: 1.6 μs

5.4 System Clock Oscillators

5.4.1 System clock oscillator

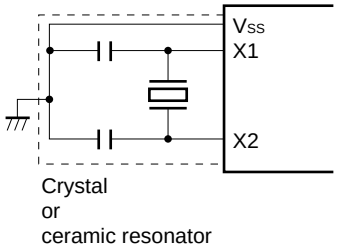
The system clock oscillator is oscillated by the crystal or ceramic resonator (5.0 MHz TYP.) connected across the X1 and X2 pins.

An external clock can also be input to the circuit. In this case, input the clock signal to the X1 pin, and input the inverted signal to the X2 pin.

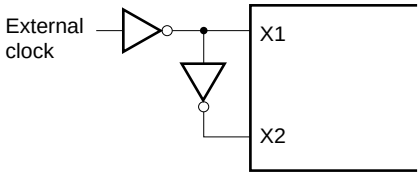
Figure 5-3 shows the external circuit of the system clock oscillator.

Figure 5-3. External Circuit of System Clock Oscillator

(a) Crystal or ceramic oscillation



(b) External clock



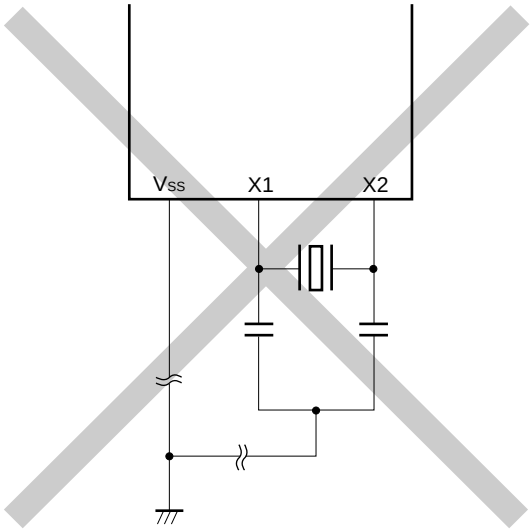
Cautions 1. When using the system clock oscillator, wire as follows in the area enclosed by the broken lines in Figure 5-3 to avoid an adverse effect from wiring capacitance.

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines. Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always make the ground point of the oscillator capacitor the same potential as V_{SS}. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator.**

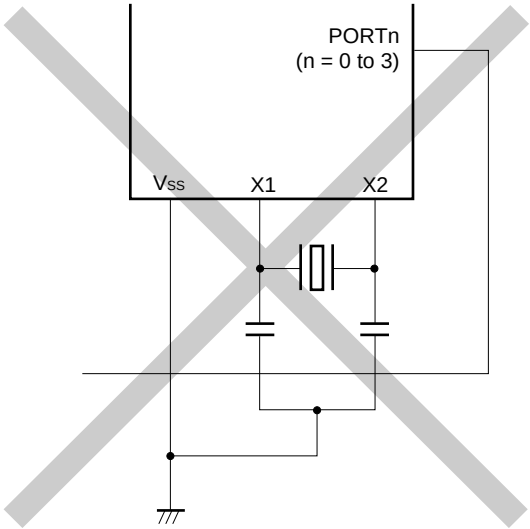
Figure 5-4 shows an example of incorrect resonator connections.

Figure 5-4. Example of Incorrect Resonator Connection (1/2)

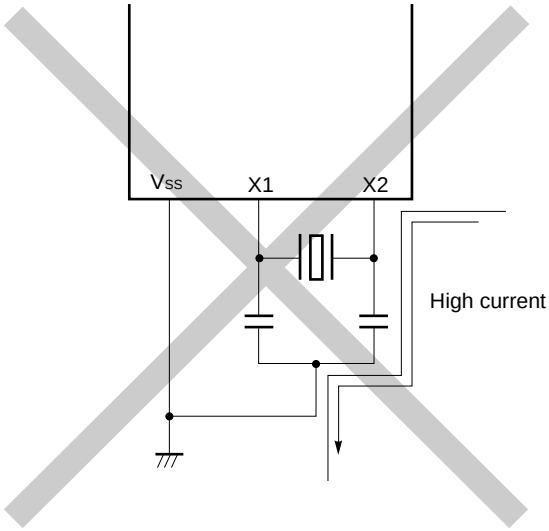
(a) Wiring too long



(b) Crossed signal line



(c) Wiring near high fluctuating current



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

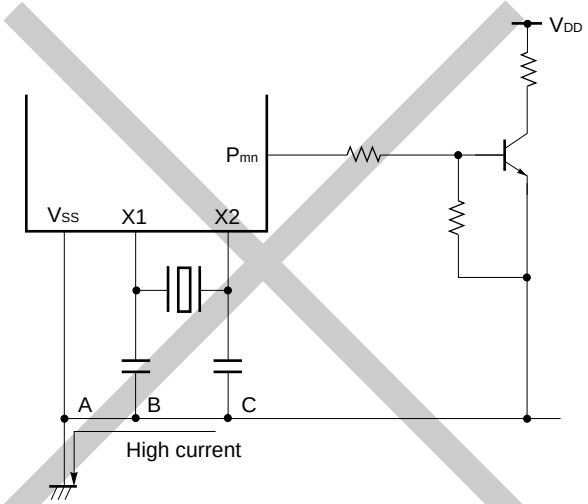
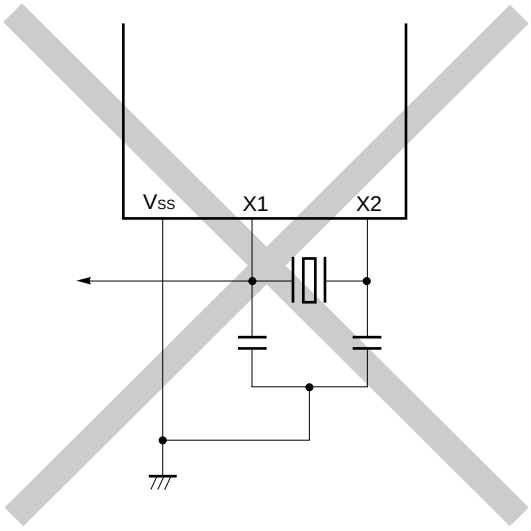


Figure 5-4. Example of Incorrect Resonator Connection (2/2)

(e) Signal is fetched



5.4.2 Frequency divider

The frequency divider divides the system clock oscillator output (fx) and generates clocks.

5.5 Clock Generator Operation

The clock generator generates the following clocks and controls the operation modes of the CPU, such as standby mode:

- System clock f_X
- CPU clock f_{CPU}
- Clock to peripheral hardware

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

(a) The slow mode $2f_{CPU}$ (1.6 μs: at 5.0 MHz operation) of the system clock is selected when the $\overline{\text{RESET}}$ signal is generated (PCC = 02H). While a low level is input to the $\overline{\text{RESET}}$ pin, oscillation of the system clock is stopped.

(b) Two types of CPU clocks (f_{CPU}) (0.2 μs, 0.8 μs: at 5.0 MHz operation) can be selected by the PCC setting.

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

(d) The clock for the peripheral hardware is generated by dividing the frequency of the system clock. Therefore, the peripheral hardware stops when the system clock stops (except for an external input clock).

5.6 Changing Setting of CPU Clock

5.6.1 Time required for switching CPU clock

The CPU clock can be selected by using bit 1 (PCC1) of the processor clock control register (PCC).

Actually, the specified clock is not selected immediately after the setting of PCC has been changed, and the old clock is used for the duration of several instructions after that (see **Table 5-2**).

Table 5-2. Maximum Time Required for Switching CPU Clock

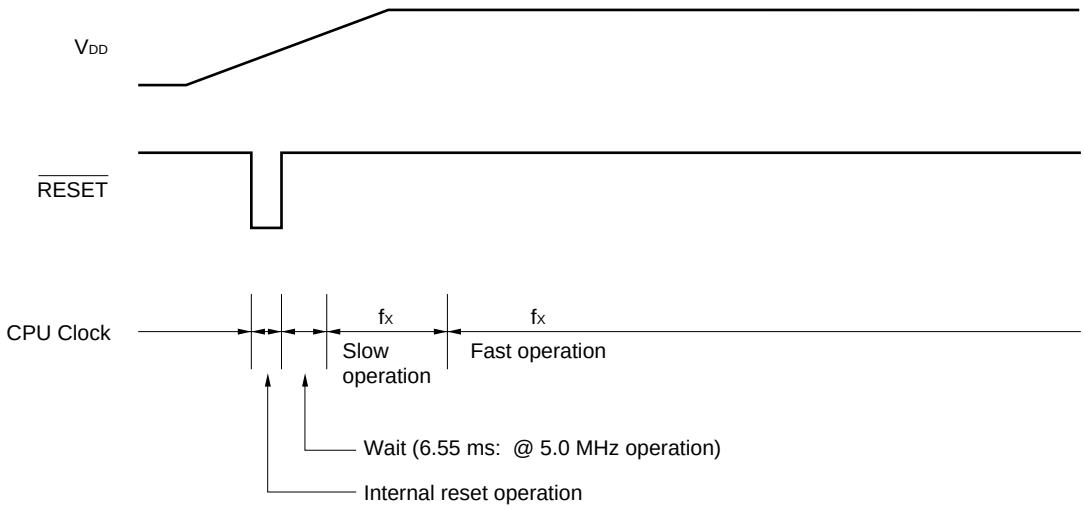
Set Value Before Switching	Set Value After Switching	
PCC1	PCC1	PCC1
	0	1
0		4 clocks
1	2 clocks	

Remark Two clocks are the minimum instruction execution time of the CPU clock before switching.

5.6.2 Switching CPU clock

The following figure illustrates how the CPU clock is switched.

Figure 5-5. Switching Between System Clock and CPU Clock



<1> The CPU is reset when the $\overline{\text{RESET}}$ pin is made low on power application. The effect of resetting is released when the $\overline{\text{RESET}}$ pin is later made high, and the system clock starts oscillating. At this time, the oscillation stabilization time ($2^{15}/f_X$) is automatically secured.

After that, the CPU starts instruction execution at the slow speed of the system clock (1.6 μs: @ 5.0 MHz operation).

<2> After the time required for the V_{DD} voltage to rise to the level at which the CPU can operate at the high speed has elapsed, the processor clock control register (PCC) is rewritten so that the high-speed operation can be selected.

[MEMO]

CHAPTER 6 16-BIT TIMER

6.1 16-Bit Timer Functions

The 16-bit timer has the following functions.

- Timer interrupt
- Timer output
- Buzzer output
- Count value capture

(1) Timer interrupt

An interrupt is generated when a count value and compare value match.

(2) Timer output

Timer output can be controlled when a count value and compare value match.

(3) Buzzer output

Buzzer output can be controlled by software.

(4) Count value capture

A count value of 16-bit timer counter 90 (TM90) is latched into a capture register synchronizing with the capture trigger and retained.

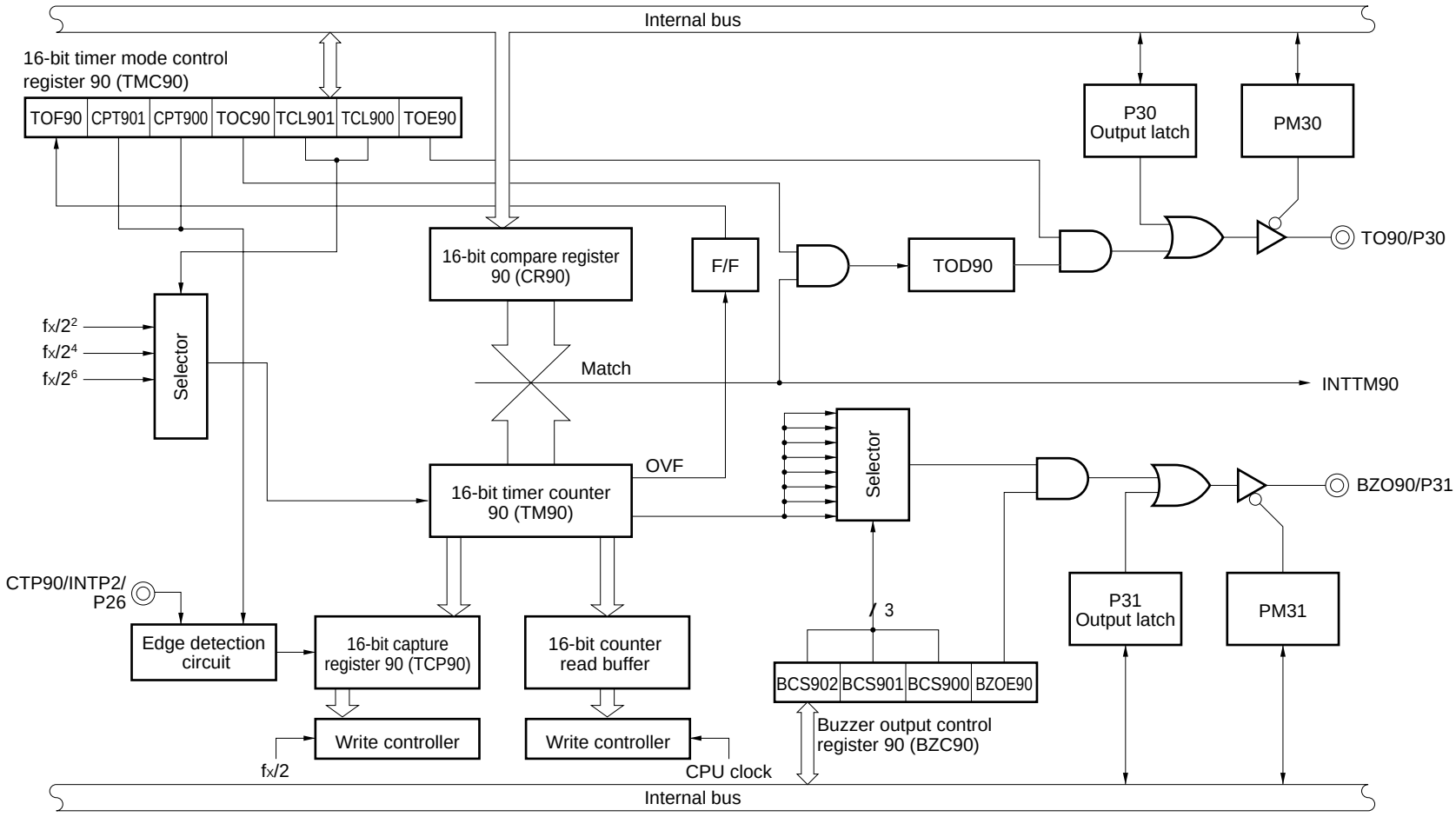
6.2 16-Bit Timer Configuration

The 16-bit timer includes the following hardware.

Table 6-1. Configuration of 16-Bit Timer

Item	Configuration
Timer counter	16 bits × 1 (TM90)
Registers	Compare register: 16 bits × 1 (CR90) Capture register: 16 bits × 1 (TCP90)
Timer outputs	1 (TO90)
Control registers	16-bit timer mode control register 90 (TMC90) Buzzer output control register 90 (BZC90) Port mode register 3 (PM3)

Figure 6-1. Block Diagram of 16-Bit Timer



(1) 16-bit compare register 90 (CR90)

A value specified in CR90 is compared with the count in 16-bit timer counter 90 (TM90). If they match, an interrupt request (INTTM90) is issued by CR90.

CR90 is set with an 8-bit or 16-bit memory manipulation instruction. Any value from 0000H to FFFFH can be set.

$\overline{\text{RESET}}$ input sets CR90 to FFFFH.

Cautions 1. CR90 is designed to be manipulated with a 16-bit memory manipulation instruction. It can also be manipulated with 8-bit memory manipulation instructions, however. When an 8-bit memory manipulation instruction is used to set CR90, it must be accessed by direct addressing.

2. To re-set CR90 during a count operation, it is necessary to disable interrupts in advance, using interrupt mask flag register 1 (MK1). It is also necessary to disable inversion of the timer output data, using 16-bit timer mode control register 90 (TMC90). If CR90 is rewritten with interrupts enabled, an interrupt request may be issued immediately.

(2) 16-bit timer counter 90 (TM90)

TM90 is used to count the number of pulses.

The contents of TM90 are read with an 8-bit or 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears TM90 to 0000H.

Cautions 1. The count becomes undefined when STOP mode is released, because the count operation is performed before oscillation stabilizes.

2. TM90 is designed to be manipulated with a 16-bit memory manipulation instruction. It can also be manipulated with 8-bit memory manipulation instructions, however. When an 8-bit memory instruction is used to manipulate TM90, it must be accessed by direct addressing.

3. When an 8-bit memory manipulation instruction is used to manipulate TM90, the lower and higher bytes must be read as a pair, in that order.

(3) 16-bit capture register 90 (TCP90)

TCP90 captures the contents of 16-bit timer counter 90 (TM90).

This register is set with an 8-bit or 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes TCP90 undefined.

Caution TCP90 is designed to be manipulated with a 16-bit memory manipulation instruction. It can also be manipulated with 8-bit memory manipulation instructions, however. When an 8-bit memory manipulation instruction is used to manipulate TCP90, it must be accessed by direct addressing.

(4) 16-bit counter read buffer 90

This buffer is used to latch and hold the count for 16-bit timer counter 90 (TM90).

6.3 16-Bit Timer Control Registers

The following three registers are used to control the 16-bit timer.

- 16-bit timer mode control register 90 (TMC90)
- Buzzer output control register 90 (BZC90)
- Port mode register 3 (PM3)

(1) 16-bit timer mode control register 90 (TMC90)

16-bit timer mode control register 90 (TMC90) controls the setting of the count clock, capture edge, etc.

TMC90 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears TMC90 to 00H.

Figure 6-2. Format of 16-Bit Timer Mode Control Register 90

Symbol	7	<6>	5	4	3	2	1	<0>	Address	After reset	R/W
TMC90	TOD90	TOF90	CPT901	CPT900	TOC90	TCL901	TCL900	TOE90	FF48H	00H	R/W[Note]

TOD90	Timer output data
0	Timer output of 0
1	Timer output of 1

TOF90	Overflow flag setting
0	Reset or cleared by software
1	Set when the 16-bit timer overflows

CPT901	CPT900	Capture edge selection
0	0	Capture operation disabled
0	1	Captured at the rising edge at the CPT90 pin
1	0	Captured at the falling edge at the CPT90 pin
1	1	Captured at both the rising and falling edges at the CPT90 pin

TOC90	Timer output data inversion control
0	Inversion disabled
1	Inversion enabled

TCL901	TCL900	16-bit timer counter 90 count clock (fcl) selection
0	0	$f_X/2^2$ (1.25 MHz)
0	1	$f_X/2^6$ (78.125 kHz)
1	0	$f_X/2^4$ (31.1 kHz)
1	1	f_{XT} (32.768 kHz)

TOE90	16-bit timer counter output control
0	Output disabled (port mode)
1	Output enabled

Note Bit 7 is read-only.

Caution Disable interrupts in advance using interrupt mask flag register 1 (MK1) when changing the data of TCL901 and TCL900. Also, prevent the timer output data from being inverted by setting TOE90 to 1.

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

(2) Buzzer output control register 90 (BZC90)

This register selects a buzzer frequency based on fcl selected with the count clock select bits (TCL901 and TCL900), and controls the output of a square wave.

BZC90 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears BZC90 to 00H.

Figure 6-3. Format of Buzzer Output Control Register 90

Symbol	7	6	5	4	3	2	1	<0>	Address	After reset	R/W
BZC90	0	0	0	0	BCS902	BCS901	BCS900	BZOE90	FF49H	00H	R/W

BCS902	BCS901	BCS900	Buzzer frequency		
			fcl = $f_X/2^2$	fcl = $f_X/2^6$	fcl = $f_X/2^4$
0	0	0	fcl/2^4 (78.125 kHz)	fcl/2^4 (4.88 kHz)	fcl/2^4 (19.5 kHz)
0	0	1	fcl/2^5 (39.063 kHz)	fcl/2^5 (2.44 kHz)	fcl/2^5 (9.77 kHz)
0	1	0	fcl/2^8 (4.88 kHz)	fcl/2^8 (305 Hz)	fcl/2^8 (1.22 kHz)
0	1	1	fcl/2^9 (2.44 kHz)	fcl/2^9 (153 Hz)	fcl/2^9 (610 Hz)
1	0	0	fcl/2^{10} (1.22 kHz)	fcl/2^{10} (76 Hz)	fcl/2^{10} (305 Hz)
1	0	1	fcl/2^{11} (610 Hz)	fcl/2^{11} (38 Hz)	fcl/2^{11} (153 Hz)
1	1	0	fcl/2^{12} (305 Hz)	fcl/2^{12} (19 Hz)	fcl/2^{12} (76.3 Hz)
1	1	1	fcl/2^{13} (153 Hz)	fcl/2^{13} (10 Hz)	fcl/2^{13} (38.1 Hz)

BZOE90	Buzzer port output control
0	Disables buzzer port output.
1	Enables buzzer port output.

Caution Bits 4 to 7 must all be set to 0.

Remarks 1. f_X: System clock oscillation frequency

2. fcl: Count clock frequency of 16-bit timer 90.

3. The parenthesized values apply to operation at f_X = 5.0 MHz.

(3) Port mode register 3 (PM3)

PM3 is used to set each bit of port 3 to input or output.

When pin P30/TO90 is used for timer output, reset the output latch of P30 and PM30 to 0; when pin P31/BZO90 is used for buzzer output, reset the output latch of P31 and PM31 to 0.

PM3 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM3 to FFH.

Figure 6-4. Format of Port Mode Register 3

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PM3	1	1	1	1	1	1	PM31	PM30	FF23H	FFH	R/W

PM3n	P3n pin I/O mode (n = 0, 1)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

6.4 16-Bit Timer Operation

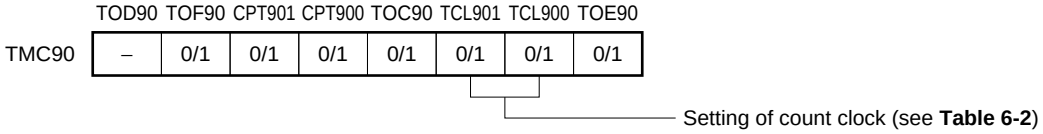
6.4.1 Operation as timer interrupt

In the timer interrupt function, interrupts are repeatedly generated at the count value preset in 16-bit compare register 90 (CR90) taking the value set in TCL901 and TCL900 as the interval.

To operate the 16-bit timer as a timer interrupt, the following settings are required.

- Set count values in CR90
- Set 16-bit timer mode control register 90 (TMC90) as shown in Figure 6-5.

Figure 6-5. Settings of 16-Bit Timer Mode Control Register 90 for Timer Interrupt Operation



Caution If 0 is set for both the CPT901 and CPT900 flags, the capture operation is disabled.

When the count value of 16-bit timer counter 90 (TM90) matches the value set in CR90, counting of TM90 continues and an interrupt request signal (INTTM90) is generated.

Table 6-2 shows the interval time, and Figure 6-6 shows the timing of the timer interrupt operation.

Caution Perform the following processing when rewriting CR90 during a count operation.
<1> Disable interrupts (TMMK90 (bit 4 of the interrupt mask flag register 1 (MK1)) = 1).
<2> Disable inversion control of timer output data (TOC90 = 0).
If CR90 is rewritten with interrupts enabled, an interrupt request may be issued immediately.

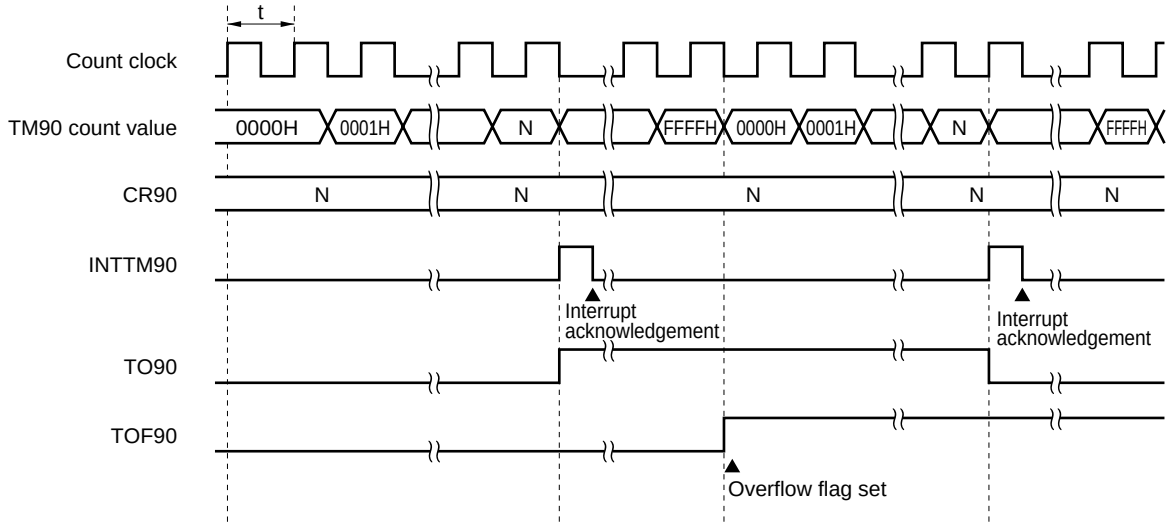
Table 6-2. Interval Time of 16-Bit Timer

TCL901	TCL900	Count Clock	Interval Time
0	0	$2^2/f_X$ (0.8 μs)	$2^{18}/f_X$ (52.4 ms)
0	1	$2^6/f_X$ (12.8 μs)	$2^{22}/f_X$ (838.9 ms)
1	0	$2^4/f_X$ (3.2 μs)	$2^{20}/f_X$ (210.0 ms)
1	1	Setting prohibited	

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

Figure 6-6. Timing of Timer Interrupt Operation



Remark N = 0000H to FFFFH

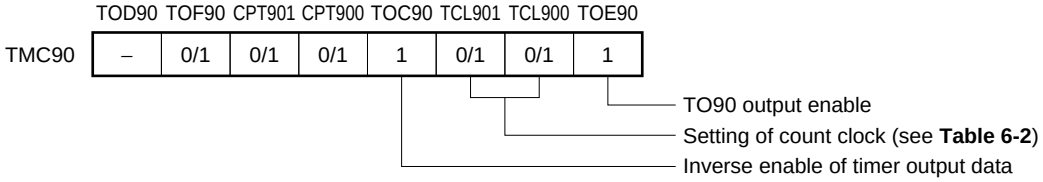
6.4.2 Operation as timer output

Timer outputs are repeatedly generated at the count value preset in 16-bit compare register 90 (CR90) taking the value set in TCL901 and TCL900 as the interval.

To operate the 16-bit timer as a timer output, the following settings are required.

- Set P30 to output mode (PM30 = 0).
- Reset the output latch of P30 to 0.
- Set the count value in CR90.
- Set 16-bit timer mode control register 90 (TMC90) as shown in Figure 6-7.

Figure 6-7. Settings of 16-Bit Timer Mode Control Register 90 for Timer Output Operation

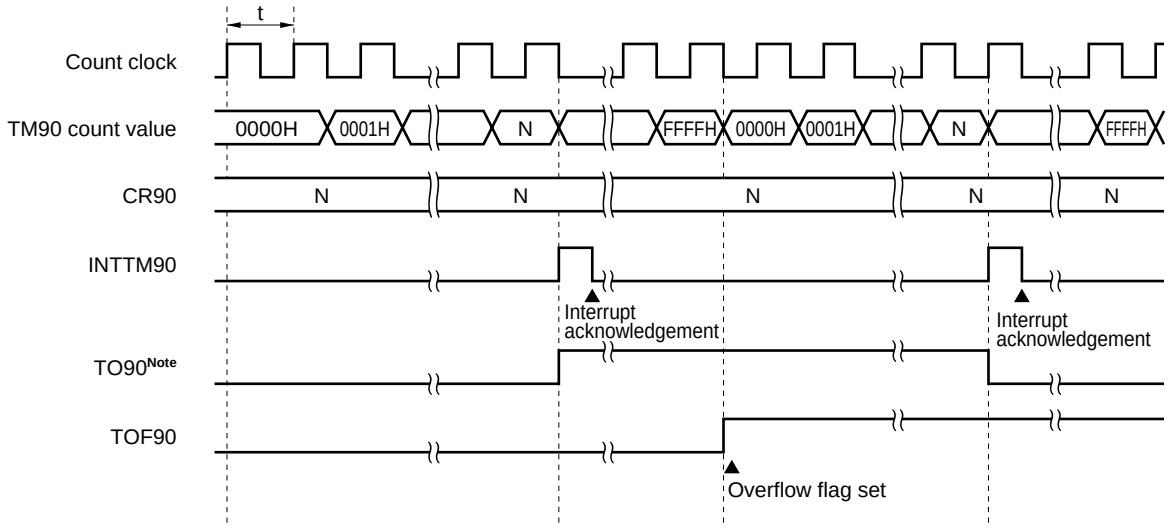


Caution If both the CPT901 flag and CPT900 flag are set to 0, the capture operation is disabled.

When the count value of 16-bit timer counter 90 (TM90) matches the value set in CR90, the output status of the TO90/P30 pin is inverted. This enables timer output. At that time, the TM90 count continues and an interrupt request signal (INTTM90) is generated.

Figure 6-8 shows the timing of timer output (see Table 6-2 for the interval time of the 16-bit timer).

Figure 6-8. Timer Output Timing



Note The TO90 initial value becomes low level during output enable (TOE90 = 1).

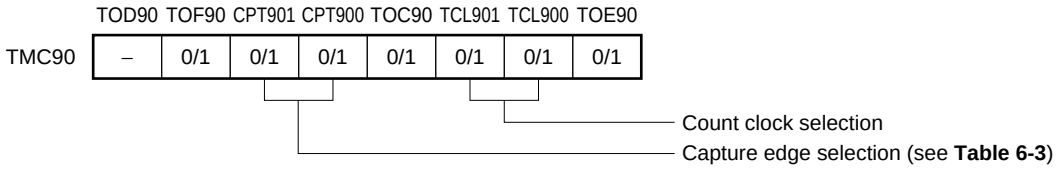
Remark N = 0000H to FFFFH

6.4.3 Capture operation

The capture operation consists of latching the count value of 16-bit timer counter 90 (TM90) into a capture register in synchronization with a capture trigger, and retaining the count value.

Set TMC90 as shown in Figure 6-9 to allow 16-bit timer to start the capture operation.

Figure 6-9. Settings of 16-Bit Timer Mode Control Register 90 for Capture Operation



16-bit capture register 90 (TCP90) starts a capture operation after the CPT90 capture trigger edge is detected, and latches and retains the count value of 16-bit timer counter 90. TCP90 fetches the count value within 2 clocks and retains the count value until the next capture edge detection.

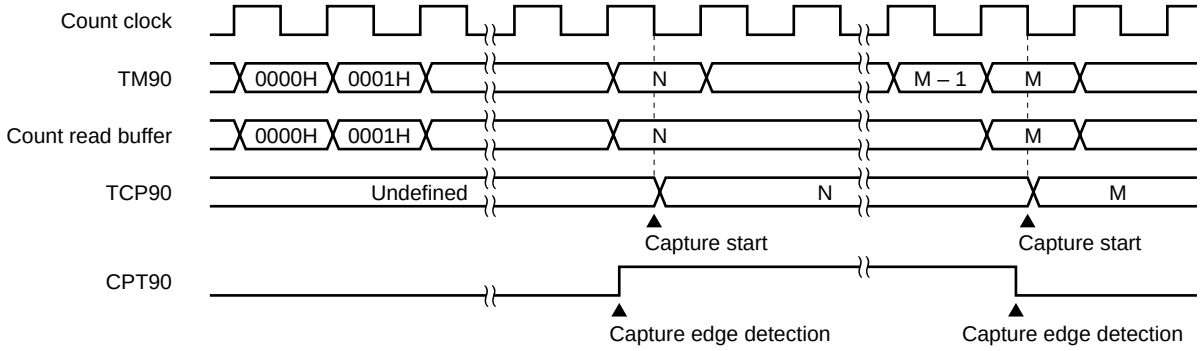
Table 6-3 and Figure 6-10 show the settings of the capture edge and capture operation timing, respectively.

Table 6-3. Settings of Capture Edge

CPT901	CPT900	Capture Edge Selection
0	0	Capture operation disabled
0	1	CPT90 pin rising edge
1	0	CPT90 pin falling edge
1	1	CPT90 pin both edges

Caution Because TCP90 is rewritten when a capture trigger edge is detected during a TCP90 read, disable the capture trigger edge detection during a TCP90 read.

Figure 6-10. Capture Operation Timing (Both Edges of CPT90 Pin Are Specified)



Remark N = 0000H to FFFFH
M = 0000H to FFFFH

6.4.4 16-bit timer counter 90 readout

The count value of 16-bit timer counter 90 (TM90) is read out with a 16-bit manipulation instruction.

TM90 readout is performed through a counter read buffer. The counter read buffer latches the TM90 count value. The buffer operation is then held pending at the CPU clock falling edge after the read signal of the TM90 lower byte rises and the count value is retained. The counter read buffer value at the retention state can be read out as the count value.

Cancellation of the pending state is performed at the CPU clock falling edge after the read signal of the TM90 higher byte falls.

$\overline{\text{RESET}}$ input clears TM90 to 0000H and TM90 resumes counting in the freerunning mode.

Figure 6-11 shows the timing of 16-bit timer counter 90 readout.

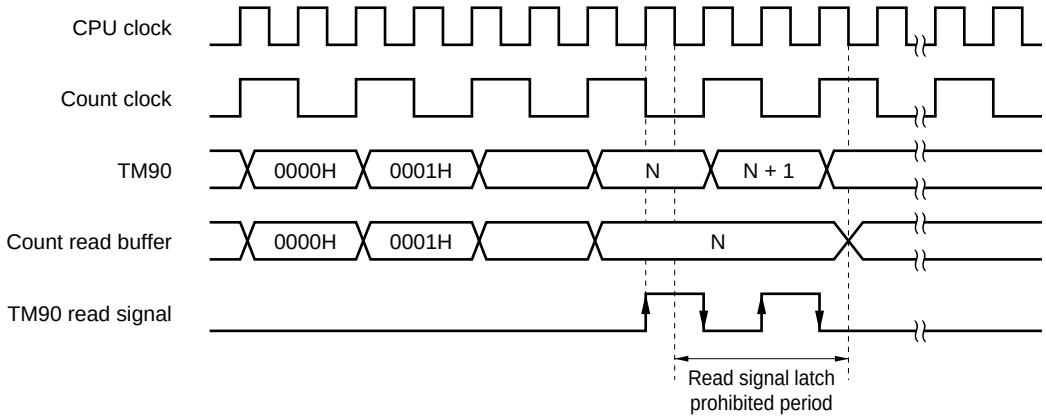
Cautions 1. The count value after releasing the stop mode becomes undefined because the count operation is executed during the oscillation stabilization time.

2. Though TM90 is designed for a 16-bit transfer instruction, an 8-bit transfer instruction can also be used.

When using the 8-bit transfer instruction, execute it by direct addressing.

3. When using the 8-bit transfer instruction, execute in the order from the lower byte to the higher byte in pairs. If only the lower byte is read, the pending state of the counter read buffer is not canceled, and if only the higher byte is read, an undefined count value is read.

Figure 6-11. 16-Bit Timer Counter 90 Readout Timing



Remark N = 0000H to FFFFH

6.4.5 Buzzer output operation

The buzzer frequency is set using buzzer output control register 90 (BZC90) based on the count clock selected with TCL901 and TCL900 of TMC90 (source clock). A square wave of the set buzzer frequency is output.

Table 6-4 shows the buzzer frequency.

Set the 16-bit timer counter as follows to use it for buzzer output:

- Set P31 to output mode (PM31 = 0).
- Reset output latch of P31 to 0.
- Set a count clock by using TCL901 and TCL900.
- Set BZC90 as shown in Figure 6-12.

Figure 6-12. Settings of Buzzer Output Control Register 90 for Buzzer Output Operation

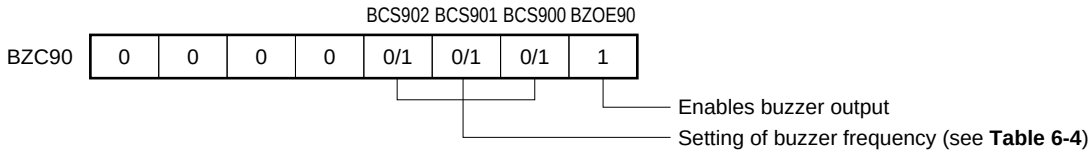


Table 6-4. Buzzer Frequency of 16-Bit Timer

BCS902	BCS901	BCS900	Buzzer Frequency		
			$fcl = f_X/2^2$	$fcl = f_X/2^6$	$fcl = f_X/2^4$
0	0	0	$fcl/2^4$ (78.1 kHz)	$fcl/2^4$ (4.88 kHz)	$fcl/2^4$ (19.5 kHz)
0	0	1	$fcl/2^5$ (39.1 kHz)	$fcl/2^5$ (2.44 kHz)	$fcl/2^5$ (9.77 kHz)
0	1	0	$fcl/2^8$ (4.88 kHz)	$fcl/2^8$ (305 Hz)	$fcl/2^8$ (1.22 kHz)
0	1	1	$fcl/2^9$ (2.44 kHz)	$fcl/2^9$ (153 Hz)	$fcl/2^9$ (610 Hz)
1	0	0	$fcl/2^{10}$ (1.22 kHz)	$fcl/2^{10}$ (76 Hz)	$fcl/2^{10}$ (305 Hz)
1	0	1	$fcl/2^{11}$ (610 Hz)	$fcl/2^{11}$ (38 Hz)	$fcl/2^{11}$ (153 Hz)
1	1	0	$fcl/2^{12}$ (305 Hz)	$fcl/2^{12}$ (19 Hz)	$fcl/2^{12}$ (76.3 Hz)
1	1	1	$fcl/2^{13}$ (153 Hz)	$fcl/2^{13}$ (10 Hz)	$fcl/2^{13}$ (38.1 Hz)

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

[MEMO]

CHAPTER 7 8-BIT TIMER/EVENT COUNTER 80

7.1 Functions of 8-Bit Timer/Event Counter 80

8-bit timer/event counter 80 has the following functions:

- Interval timer
- External event counter
- Square wave output
- PWM output

(1) 8-bit interval timer

When 8-bit timer/event counter 80 is used as an interval timer, it generates an interrupt at a time interval set in advance.

Table 7-1. Interval Time of 8-Bit Timer/Event Counter 80

Minimum Interval Time	Maximum Interval Time	Resolution
$1/f_X$ (200 ns)	$2^8/f_X$ (51.2 μs)	$1/f_X$ (200 ns)
$2^8/f_X$ (51.2 μs)	$2^{16}/f_X$ (13.1 ms)	$2^8/f_X$ (51.2 μs)

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

(2) External event counter

The number of pulses of an externally input signal can be counted.

(3) Square wave output

A square wave of arbitrary frequency can be output.

Table 7-2. Square Wave Output Range of 8-Bit Timer/Event Counter 80

Minimum Pulse Width	Maximum Pulse Width	Resolution
$1/f_X$ (200 ns)	$2^9/f_X$ (51.2 μs)	$1/f_X$ (200 ns)
$2^8/f_X$ (51.2 μs)	$2^{16}/f_X$ (13.1 ms)	$2^8/f_X$ (51.2 μs)

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

(4) PWM output

8-bit resolution PWM output can be produced.

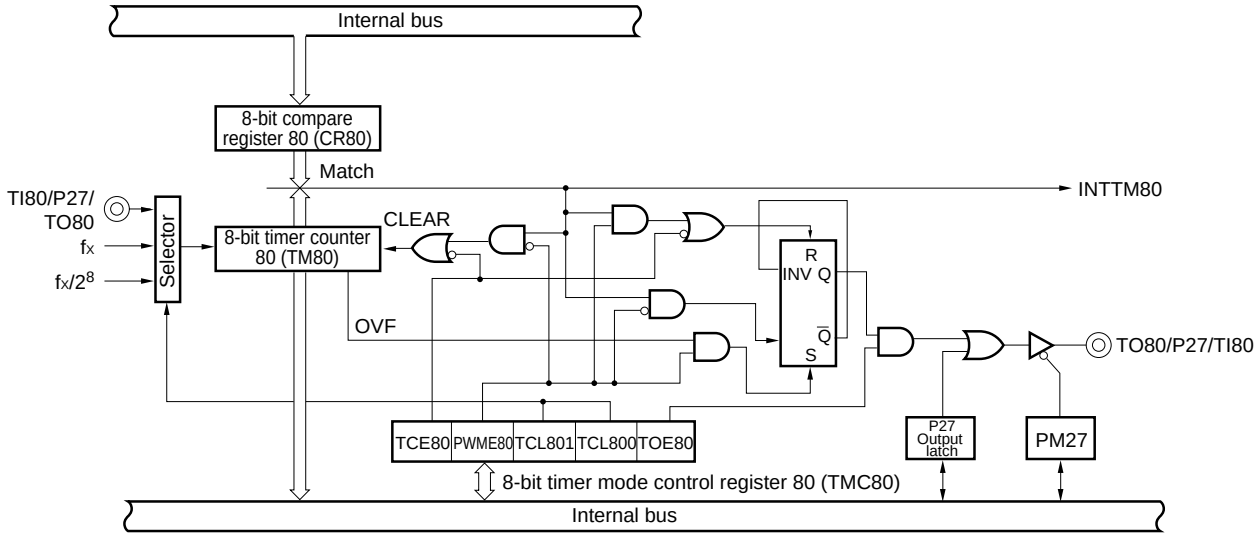
7.2 8-Bit Timer/Event Counter 80 Configuration

8-bit timer/event counter 80 includes the following hardware.

Table 7-3. Configuration of 8-Bit Timer/Event Counter 80

Item	Configuration
Timer counter	8 bits × 1 (TM80)
Register	Compare register: 8 bits × 1 (CR80)
Timer outputs	1 (TO80)
Control registers	8-bit timer mode control register 80 (TMC80) Port mode register 2 (PM2)

Figure 7-1. Block Diagram of 8-Bit Timer/Event Counter 80



(1) 8-bit compare register 80 (CR80)

A value specified in CR80 is compared with the count in 8-bit timer counter 80 (TM80). If they match, an interrupt request (INTTM80) is issued.

CR80 is set with an 8-bit memory manipulation instruction. Any value from 00H to FFH can be set.

$\overline{\text{RESET}}$ input makes CR80 undefined.

Cautions 1. Before rewriting CR80, stop the timer operation. If CR80 is rewritten while the timer operation is enabled, the match interrupt request signal may be generated immediately.

2. Do not clear CR80 to 00H in PWM output mode (when PWME80 = 1: bit 6 of 8-bit timer mode control register 80 (TMC80)); otherwise, PWM output may not be produced normally.

(2) 8-bit timer counter 80 (TM80)

TM80 is used to count the number of pulses.

Its contents are read with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears TM80 to 00H.

7.3 8-Bit Timer/Event Counter 80 Control Registers

The following two registers are used to control 8-bit timer/event counter 80.

- 8-bit timer mode control register 80 (TMC80)
- Port mode register 2 (PM2)

(1) 8-bit timer mode control register 80 (TMC80)

TMC80 determines whether to enable or disable 8-bit timer counter 80 (TM80), specifies the count clock for TM80, and controls the operation of the output control circuit of 8-bit timer/event counter.

TMC80 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears TMC80 to 00H.

Figure 7-2. Format of 8-Bit Timer Mode Control Register 80

Symbol	<7>	<6>	5	4	3	2	1	<0>	Address	After reset	R/W
TMC80	TCE80	PWME80	0	0	0	TCL801	TCL800	TOE80	FF53H	00H	R/W

TCE80	8-bit timer counter 80 operation control
0	Operation disabled (TM80 is cleared to 0.)
1	Operation enabled

PWME80	Operation mode selection
0	Timer counter operation mode
1	PWM output mode

TCL801	TCL800	8-bit timer counter 80 count clock selection
0	0	f_X (5.0 MHz)
0	1	$f_X/2^8$ (19.5 kHz)
1	0	Rising edge of TI80[Note]
1	1	Falling edge of TI80[Note]

TOE80	8-bit timer/event counter output control
0	Output disabled (port mode)
1	Output enabled

Note When inputting a clock signal externally, timer output cannot be used.

Caution Always stop the timer before setting TMC80.

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

(2) Port mode register 2 (PM2)

PM2 specifies whether each bit of port 2 is used for input or output.

To use the TO80/P27/TI80 pin for timer output, the PM27 and P27 output latch must be reset to 0.

PM2 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM2 to FFH.

Figure 7-3. Format of Port Mode Register 2

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20	FF22H	FFH	R/W

PM2n	P2n pin input/output mode selection (n = 0 to 5)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

7.4 Operation of 8-Bit Timer/Event Counter 80

7.4.1 Operation as interval timer

The interval timer repeatedly generates an interrupt at a time interval specified by the count value preset in 8-bit compare register 80 (CR80).

To operate 8-bit timer/event counter 80 as an interval timer, settings must be made in the following sequence.

<1> Disable operation of 8-bit timer counter 80 (TM80) (TCE80 (bit 7 of 8-bit timer mode control register 80 (TMC80)) = 0).
<2> Set the count clock of 8-bit timer/event counter 80 (see **Table 7-4**).
<3> Set a count value in CR80.
<4> Enable the operation of TM80 (TCE80 = 1).

When the count value of 8-bit timer counter 80 (TM80) matches the value set in CR80, TM80 is cleared to 0 and continues counting. At the same time, an interrupt request signal (INTTM80) is generated.

Table 7-4 shows interval time, and Figure 7-4 shows the timing of interval timer operation.

Cautions 1. Stop the timer operation before rewriting CR80. If CR80 is rewritten while the timer operation is enabled, a match signal may be generated immediately (an interrupt request will be generated if interrupts are enabled).

2. If setting the count clock to TMC80 and enabling the operation of TM80 are performed at the same time with an 8-bit memory manipulation instruction, the error one cycle after the timer has been started may exceed one clock. To use the 8-bit timer/event counter as an interval timer, therefore, make the settings in the above sequence.

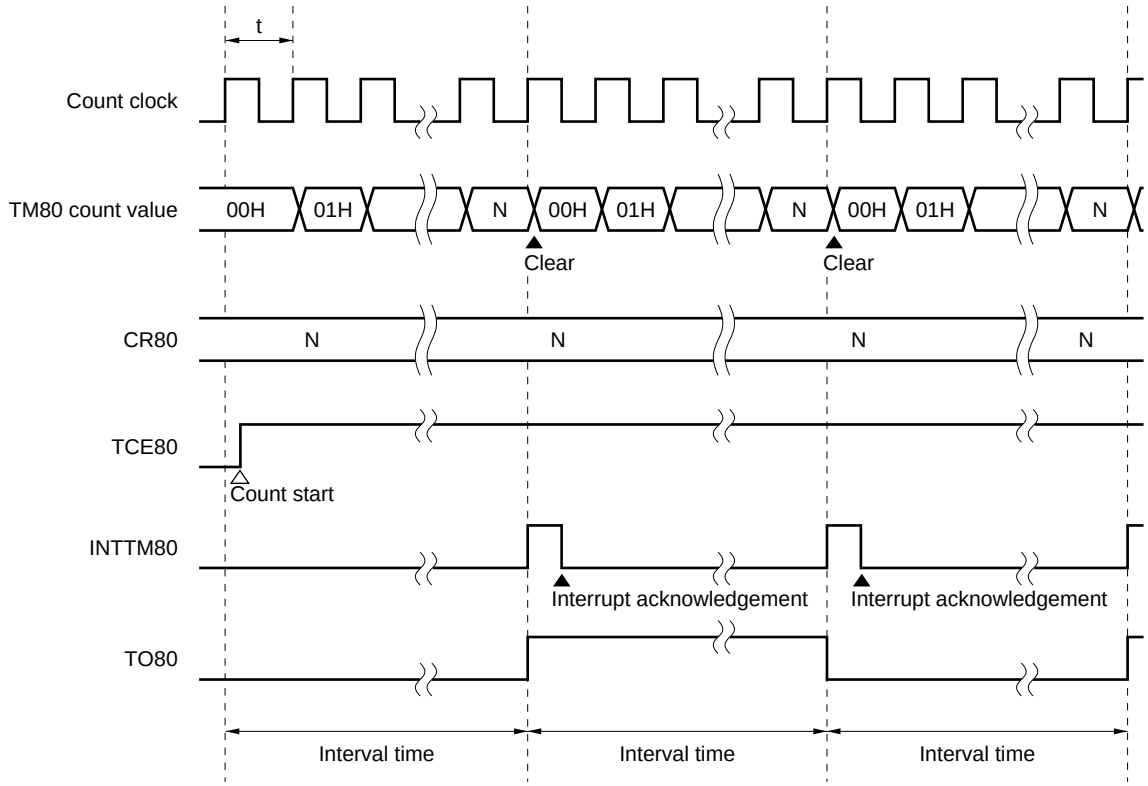
Table 7-4. Interval Time of 8-Bit Timer/Event Counter 80

TCL801	TCL800	Minimum Interval Time	Maximum Interval Time	Resolution
0	0	$1/f_X$ (200 ns)	$2^8/f_X$ (51.2 μs)	$1/f_X$ (200 ns)
0	1	$2^8/f_X$ (51.2 μs)	$2^{16}/f_X$ (13.1 ms)	$2^8/f_X$ (51.2 μs)
1	0	TI80 input cycle	$2^8 \times$ TI80 input cycle	TI80 input edge cycle
1	1	TI80 input cycle	$2^8 \times$ TI80 input cycle	TI80 input edge cycle

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

Figure 7-4. Interval Timer Operation Timing



Remark Interval time = (N + 1) × t
N = 00H to FFH

7.4.2 Operation as external event counter

The external event counter counts the number of external clock pulses input to the TI80/P27/TO80 pin by using 8-bit timer counter 80 (TM80).

To operate 8-bit timer/event counter 80 as an external event counter, settings must be made in the following sequence.

<1> Set P27 to input mode (PM27 = 1).
<2> Disable operation of 8-bit timer counter 80 (TM80) (TCE80 (bit 7 of 8-bit timer mode control register 80 (TMC80)) = 0).
<3> Specify the rising or falling edge of TI80 (see **Table 7-4**). Disable output of TO80 (TOE80 (bit 0 of TMC80) = 0) and PWM output (PWME80 (bit 6 of TMC80) = 0).
<4> Set a count value in CR80.
<5> Enable the operation of TM80 (TCE80 = 1).

Each time the valid edge specified by bit 1 (TCL800) of TMC80 is input, the value of 8-bit timer counter 80 (TM80) is incremented.

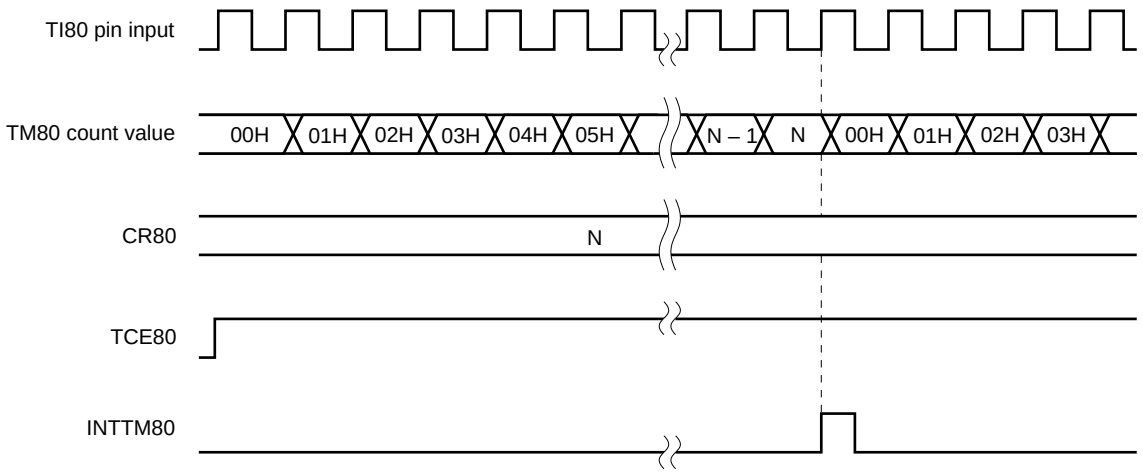
When the count value of TM80 matches the value set in CR80, TM80 is cleared to 0 and continues counting. At the same time, an interrupt request signal (INTTM80) is generated.

Figure 7-5 shows the timing of the external event counter operation (with rising edge specified).

Cautions 1. Before rewriting CR80, stop the timer operation. If CR80 is rewritten while the timer operation is enabled, a match interrupt request signal may be generated immediately.

2. If setting the count clock to TMC80 and enabling the operation of TM80 are performed at the same time with an 8-bit memory manipulation instruction, the error one cycle after the timer has been started may exceed one clock. To use the 8-bit timer/event counter as an external event counter, therefore, make the settings in the above sequence.

Figure 7-5. External Event Counter Operation Timing (with Rising Edge Specified)



Remark N = 00H to FFH

7.4.3 Operation as square wave output

The 8-bit timer/event counter can generate output square waves of an arbitrary frequency at an interval specified by the count value preset in 8-bit compare register 80 (CR80).

To operate 8-bit timer/event counter 80 for square wave output, settings must be made in the following sequence.

<1> Set P27 to output mode (PM27 = 0). Set the output latch of P27 to 0.
<2> Disable operation of 8-bit timer counter 80 (TM80) (TCE80 = 0).
<3> Set a count clock for 8-bit timer/event counter 80 (see **Table 7-5**), enable output of TO80 (TOE80 = 1), and disable PWM output (PWME80 = 0).
<4> Set a count value in CR80.
<5> Enable the operation of TM80 (TCE80 = 1).

When the count value of 8-bit timer counter 80 (TM80) matches the value set in CR80, the TO80 pin output will be inverted. Through application of this mechanism, square waves of any frequency can be output. As soon as a match occurs, TM80 is cleared to 0 and continues counting, generating an interrupt request signal (INTTM80).

Setting bit 7 (TCE80) of TMC80 to 0 clears the square-wave output to 0.

Table 7-5 shows the square wave output range, and Figure 7-6 shows timing of square wave output.

Cautions 1. Stop the timer operation before rewriting CR80. If CR80 is rewritten while the timer operation is enabled, a match interrupt request signal may be generated immediately.

2. If setting the count clock to TMC80 and enabling the operation of TM80 are performed at the same time with an 8-bit memory manipulation instruction, the error one cycle after the timer has been started may exceed one clock. To use the 8-bit timer/event counter as a square wave output, therefore, make the settings in the above sequence.

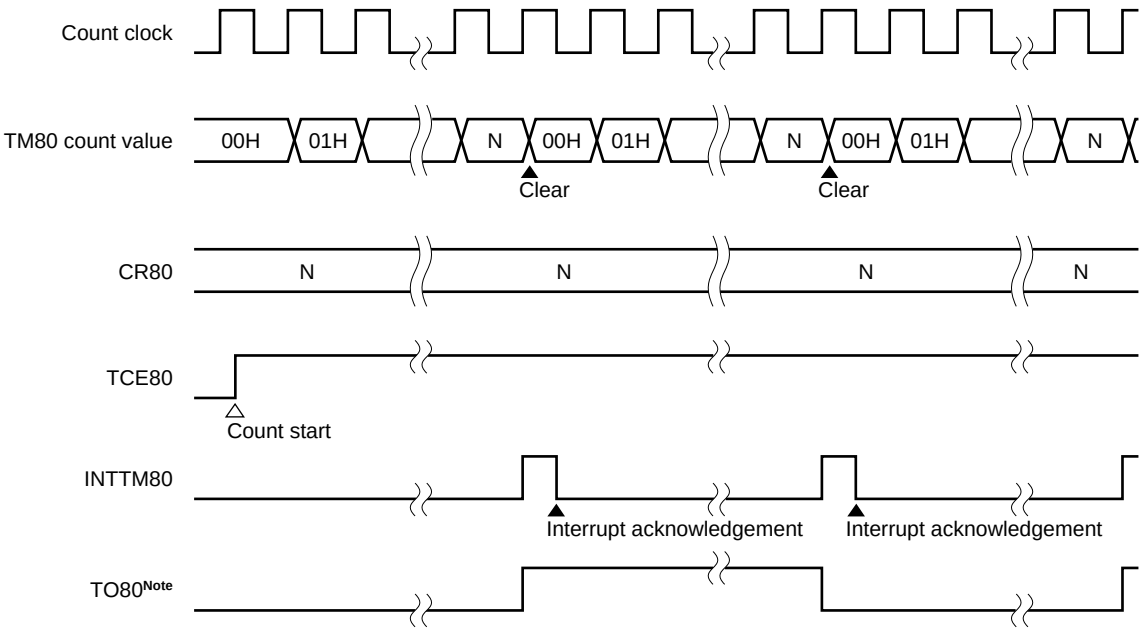
Table 7-5. Square Wave Output Range of 8-Bit Timer/Event Counter

TCL801	TCL800	Minimum Pulse Width	Maximum Pulse Width	Resolution
0	0	$1/f_X$ (200 ns)	$2^8/f_X$ (51.2 μs)	$1/f_X$ (200 ns)
0	1	$2^8/f_X$ (51.2 μs)	$2^{16}/f_X$ (13.1 ms)	$2^8/f_X$ (51.2 μs)

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

Figure 7-6. Square Wave Output Timing



Note The initial value of TO80 is low for output enable (TOE80 = 1).

7.4.4 Operation as PWM output

PWM output enables an interrupt to be generated repeatedly at an interval specified by the count value preset in 8-bit compare register 80 (CR80).

To use 8-bit timer/event counter 80 for PWM output, the following settings are required.

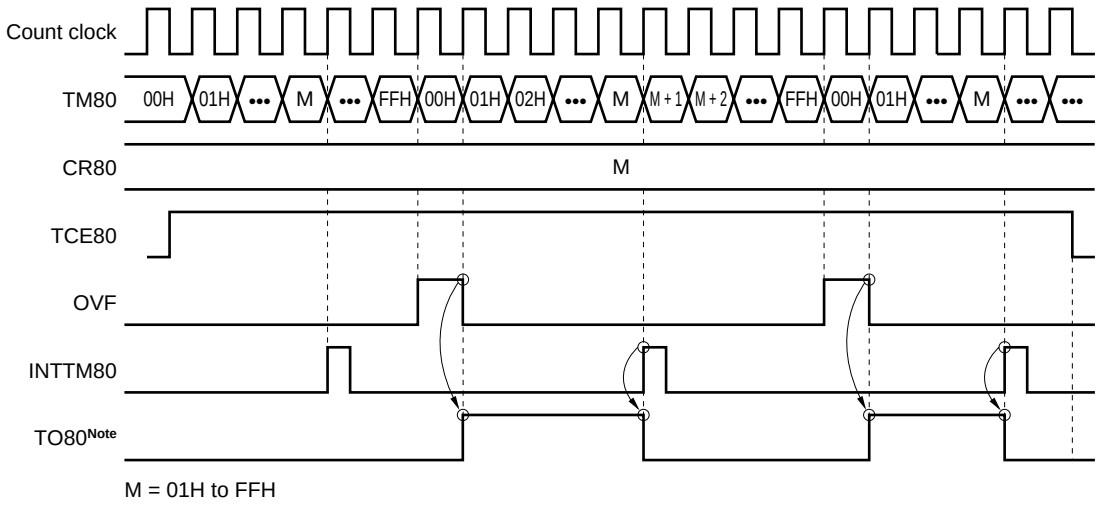
<1> Set P27 to output mode (PM27 = 0). Set the output latch of P27 to 0.
<2> Disable the operation of 8-bit timer counter 80 (TM80) (TCE80 = 0).
<3> Set a count clock for 8-bit timer/event counter (see **Table 7-4**), and enable output of TO80 (TOE80 = 1) and PWM output (PWME80 = 1).
<4> Set a count value in CR80.
<5> Enable the operation of TM80 (TCE80 = 1).

When the count value of 8-bit timer counter 80 (TM80) matches the value set in CR80, TM80 continues counting, and an interrupt request signal (INTTM80) is generated.

Cautions 1. If CR80 is rewritten during timer operation, a high level may be output during the next cycle (see 7.5 (2) Setting of 8-bit compare register 80).

2. If setting the count clock to TMC80 and enabling the operation of TM80 are performed at the same time with an 8-bit memory manipulation instruction, the error one cycle after the timer has been started may exceed one clock. To use the 8-bit timer/event counter as a PWM output, therefore, make the settings in the above sequence.

Figure 7-7. PWM Output Timing



Note The initial value of TO80 is low for output enable (TOE80 = 1).

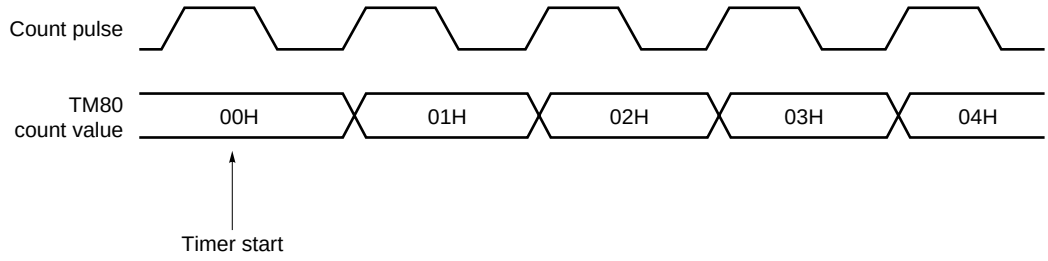
Caution Do not set CR80 to 00H in PWM output mode, otherwise, PWM may not be output normally.

7.5 Notes on Using 8-Bit Timer/Event Counter 80

(1) Error on starting timer

An error of up to 1 clock is included in the time between the timer being started and a match signal being generated. This is because 8-bit timer counter 80 (TM80) is started asynchronously to the count pulse.

Figure 7-8. Start Timing of 8-Bit Timer Counter 80

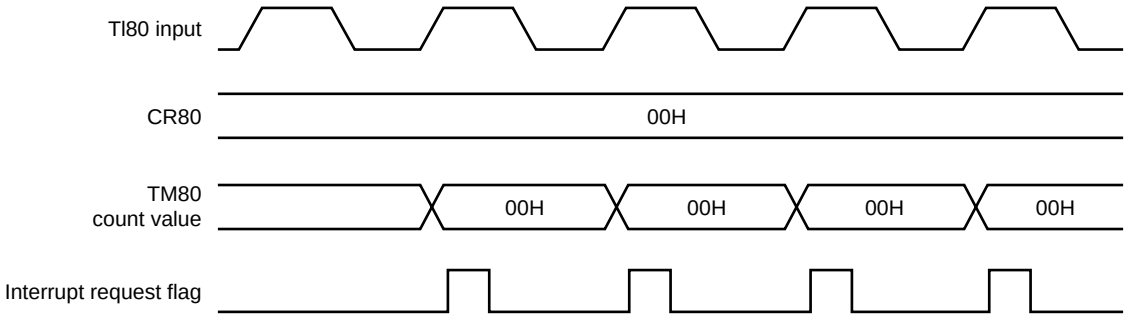


(2) Setting of 8-bit compare register 80

8-bit compare register 80 (CR80) can be set to 00H.

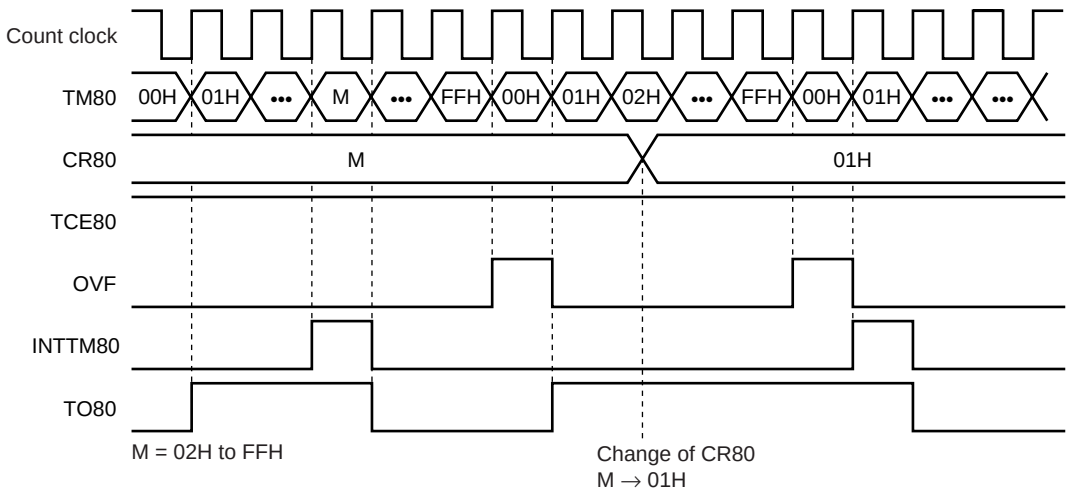
Therefore, one pulse can be counted when 8-bit timer/event counter 80 operates as an event counter.

Figure 7-9. External Event Counter Operation Timing



Cautions 1. Before rewriting CR80 in timer counter operation mode (PWME80 (bit 6 of 8-bit timer mode control register 80 (TMC80) = 0), stop the timer operation. If CR80 is rewritten while the timer operation is enabled, a match interrupt request signal may be generated immediately.

2. If CR80 is rewritten during timer operation in PWM output operation mode (PWME80 = 1), a high level may be output during the next cycle (count pulse × 256). This occurs when CR80 is set to a value smaller than the TM80 value at the time CR80 is rewritten.



3. Do not set CR80 to 00H in PWM operation mode (when PWME80 = 1) otherwise, PWM may not be output normally.

CHAPTER 8 WATCHDOG TIMER

8.1 Watchdog Timer Functions

The watchdog timer has the following functions:

- Watchdog timer
- Interval timer

Caution Select the watchdog timer mode or interval timer mode by using the watchdog timer mode register (WDTM).

(1) Watchdog timer

The watchdog timer is used to detect inadvertent program loops. When an inadvertent loop is detected, a non-maskable interrupt or a $\overline{\text{RESET}}$ signal can be generated.

Table 8-1. Inadvertent Loop Detection Time of Watchdog Timer

Inadvertent Loop Detection Time	At f_X = 5.0 MHz
$2^{11} \times 1/f_X$	410 μs
$2^{13} \times 1/f_X$	1.64 ms
$2^{15} \times 1/f_X$	6.55 ms
$2^{17} \times 1/f_X$	26.2 ms

f_X: System clock oscillation frequency

(2) Interval timer

The interval timer generates an interrupt at an arbitrary preset interval.

Table 8-2. Interval Time

Interval	At f_X = 5.0 MHz
$2^{11} \times 1/f_X$	410 μs
$2^{13} \times 1/f_X$	1.64 ms
$2^{15} \times 1/f_X$	6.55 ms
$2^{17} \times 1/f_X$	26.2 ms

f_X: System clock oscillation frequency

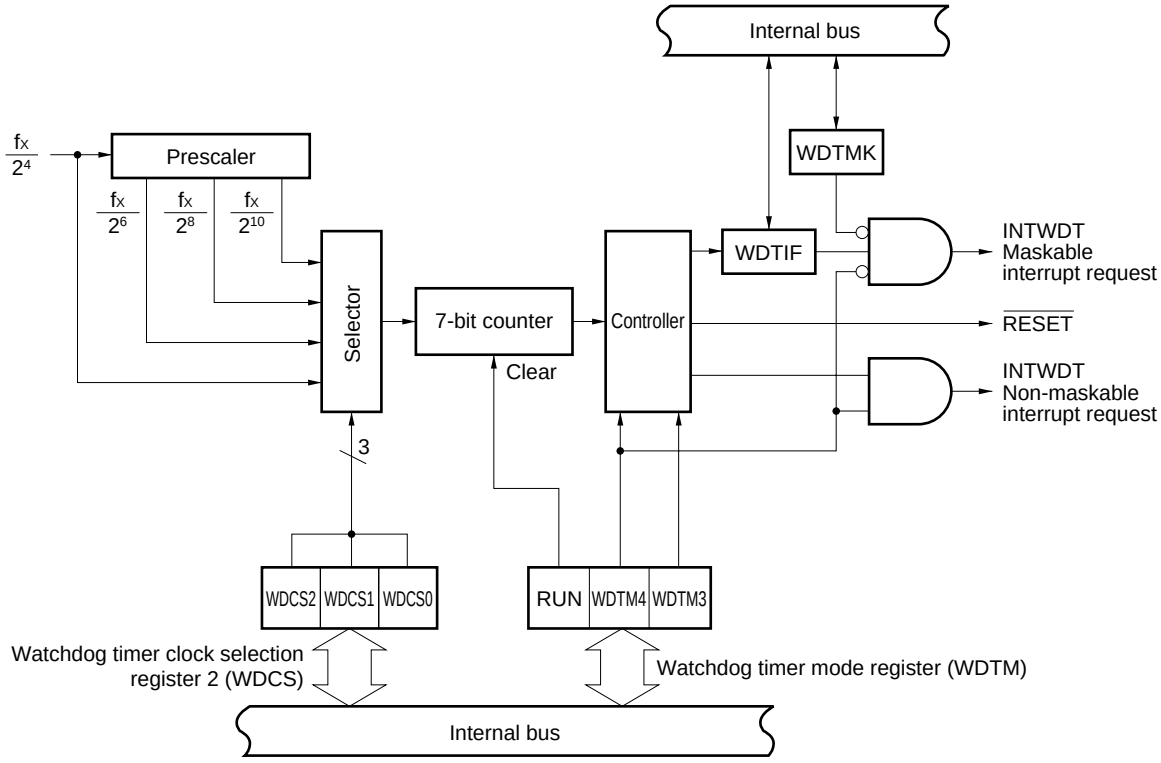
8.2 Watchdog Timer Configuration

The watchdog timer includes the following hardware.

Table 8-3. Configuration of Watchdog Timer

Item	Configuration
Control registers	Watchdog timer clock selection register (WDCS) Watchdog timer mode register (WDTM)

Figure 8-1. Block Diagram of Watchdog Timer



8.3 Watchdog Timer Control Registers

The following two registers are used to control the watchdog timer.

- Watchdog timer clock selection register (WDCS)
- Watchdog timer mode register (WDTM)

(1) Watchdog timer clock selection register (WDCS)

This register sets the watchdog timer count clock.

WDCS is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears WDCS to 00H.

Figure 8-2. Format of Watchdog Timer Clock Selection Register

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
WDCS	0	0	0	0	0	WDCS2	WDCS1	WDCS0	FF42H	00H	R/W

WDCS2	WDCS1	WDCS0	Count clock selection	Interval time
0	0	0	$f_X/2^4$ (313 kHz)	$2^{11}/f_X$ (410 μs)
0	1	0	$f_X/2^6$ (78.1 kHz)	$2^{13}/f_X$ (1.64 ms)
1	0	0	$f_X/2^8$ (19.5 kHz)	$2^{15}/f_X$ (6.55 ms)
1	1	0	$f_X/2^{10}$ (4.88 kHz)	$2^{17}/f_X$ (26.2 ms)
Other than above			Setting prohibited	

Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

(2) Watchdog timer mode register (WDTM)

This register sets the operation mode of the watchdog timer, and enables/disables counting of the watchdog timer.

WDTM is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears WDTM to 00H.

Figure 8-3. Format of Watchdog Timer Mode Register

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
WDTM	RUN	0	0	WDTM4	WDTM3	0	0	0	FFF9H	00H	R/W

RUN	Watchdog timer operation selection[Note 1]
0	Stops counting.
1	Clears counter and starts counting.

WDTM4	WDTM3	Watchdog timer operation mode selection[Note 2]
0	0	Operation stop
0	1	Interval timer mode (Generates a maskable interrupt upon overflow occurrence.)[Note 3]
1	0	Watchdog timer mode 1 (Generates a non-maskable interrupt upon overflow occurrence.)
1	1	Watchdog timer mode 2 (Starts a reset operation upon overflow occurrence.)

Notes 1. Once RUN has been set to 1, it cannot be cleared to 0 by software. Therefore, when counting is started, it cannot be stopped by any means other than $\overline{\text{RESET}}$ input.

2. Once WDTM3 and WDTM4 have been set to 1, they cannot be cleared to 0 by software.

3. The watchdog timer starts operation as an interval timer when RUN is set to 1.

Cautions 1. When the watchdog timer is cleared by setting RUN to 1, the actual overflow time is up to 0.8% shorter than the time set by the watchdog timer clock selection register (WDCS).

2. To set watchdog timer mode 1 or 2, set WDTM4 to 1 after confirming WDTIF (bit 0 of interrupt request flag register 0 (IF0)) being set to 0. When watchdog timer mode 1 or 2 is selected with WDTIF set to 1, a non-maskable interrupt is generated upon the completion of rewriting WDTM.

8.4 Watchdog Timer Operation

8.4.1 Operation as watchdog timer

The watchdog timer detects an inadvertent program loop when bit 4 (WDTM4) of the watchdog timer mode register (WDTM) is set to 1.

The count clock (inadvertent loop detection time interval) of the watchdog timer can be selected by bits 0 to 2 (WDCS0 to WDCS2) of the watchdog timer clock selection register (WDCS). By setting bit 7 (RUN) of WDTM to 1, the watchdog timer is started. Set RUN to 1 within the set inadvertent loop detection time interval after the watchdog timer has been started. By setting RUN to 1, the watchdog timer can be cleared and start counting. If RUN is not set to 1, and the inadvertent loop detection time is exceeded, a system reset signal or a non-maskable interrupt is generated, depending on the value of bit 3 (WDTM3) of WDTM.

The watchdog timer continues operation in HALT mode, but stops in STOP mode. Therefore, first set RUN to 1 to clear the watchdog timer before executing the STOP instruction.

Caution The actual inadvertent loop detection time may be up to 0.8% shorter than the set time.

Table 8-4. Inadvertent Loop Detection Time of Watchdog Timer

WDCS2	WDCS1	WDCS0	Inadvertent Loop Detection Time	At f_X = 5.0 MHz
0	0	0	$2^{11} \times 1/f_X$	410 μs
0	1	0	$2^{13} \times 1/f_X$	1.64 ms
1	0	0	$2^{15} \times 1/f_X$	6.55 ms
1	1	0	$2^{17} \times 1/f_X$	26.2 ms

f_X: System clock oscillation frequency

8.4.2 Operation as interval timer

When bits 4 and 3 (WDTM4, WDTM3) of the watchdog timer mode register (WDTM) are set to 0 and 1, respectively, the watchdog timer operates as an interval timer that repeatedly generates an interrupt at an interval specified by a preset count value.

Select a count clock (or interval time) by setting bits 0 to 2 (WDCS0 to WDCS2) of the watchdog timer clock selection register (WDCS). The watchdog timer starts operation as an interval timer when the RUN bit (bit 7 of WDTM) is set to 1.

In interval timer mode, the interrupt mask flag (WDTMK) is valid, and a maskable interrupt (INTWDT) can be generated. The priority of INTWDT is set as the highest of all the maskable interrupts.

The interval timer continues operation in HALT mode, but stops in STOP mode. Therefore, first set RUN to 1 to clear the interval timer before executing the STOP instruction.

Cautions 1. Once bit 4 (WDTM4) of WDTM is set to 1 (when watchdog timer mode is selected), interval timer mode is not set unless a $\overline{\text{RESET}}$ signal is input.

2. The interval time may be up to 0.8% shorter than the set time when WDTM has just been set.

Table 8-5. Interval Generated Using Interval Timer

WDCS2	WDCS1	WDCS0	Interval Time	At f_X = 5.0 MHz
0	0	0	$2^{11} \times 1/f_X$	410 μs
0	1	0	$2^{13} \times 1/f_X$	1.64 ms
1	0	0	$2^{15} \times 1/f_X$	6.55 ms
1	1	0	$2^{17} \times 1/f_X$	26.2 ms

f_X: System clock oscillation frequency

CHAPTER 9 SERIAL INTERFACE 20

9.1 Serial Interface 20 Functions

Serial interface 20 has the following three modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode
- 3-wire serial I/O mode

(1) Operation stop mode

This mode is used when serial transfer is not performed. Power consumption is minimized in this mode.

(2) Asynchronous serial interface (UART) mode

This mode is used to send and receive the one byte of data that follows a start bit. It supports full-duplex communication.

Serial interface 20 contains a UART-dedicated baud rate generator, enabling communication over a wide range of baud rates. It is also possible to define baud rates by dividing the frequency of the clock input to the ASCK20 pin.

(3) 3-wire serial I/O mode (switchable between MSB-first and LSB-first transmission)

This mode is used to transmit 8-bit data, using three lines: a serial clock ($\overline{\text{SCK20}}$) line and two serial data lines (SI20 and SO20).

As it supports simultaneous transmission and reception, 3-wire serial I/O mode requires less processing time for data transmission than asynchronous serial interface mode.

Because, in 3-wire serial I/O mode, it is possible to select whether 8-bit data transmission begins with the MSB or LSB, serial interface 20 can be connected to any device regardless of whether that device is designed for MSB-first or LSB-first transmission.

3-wire serial I/O mode is useful for connecting peripheral I/O circuits and display controllers having conventional synchronous serial interfaces, such as those of the 75X/XL, 78K, and 17K Series devices.

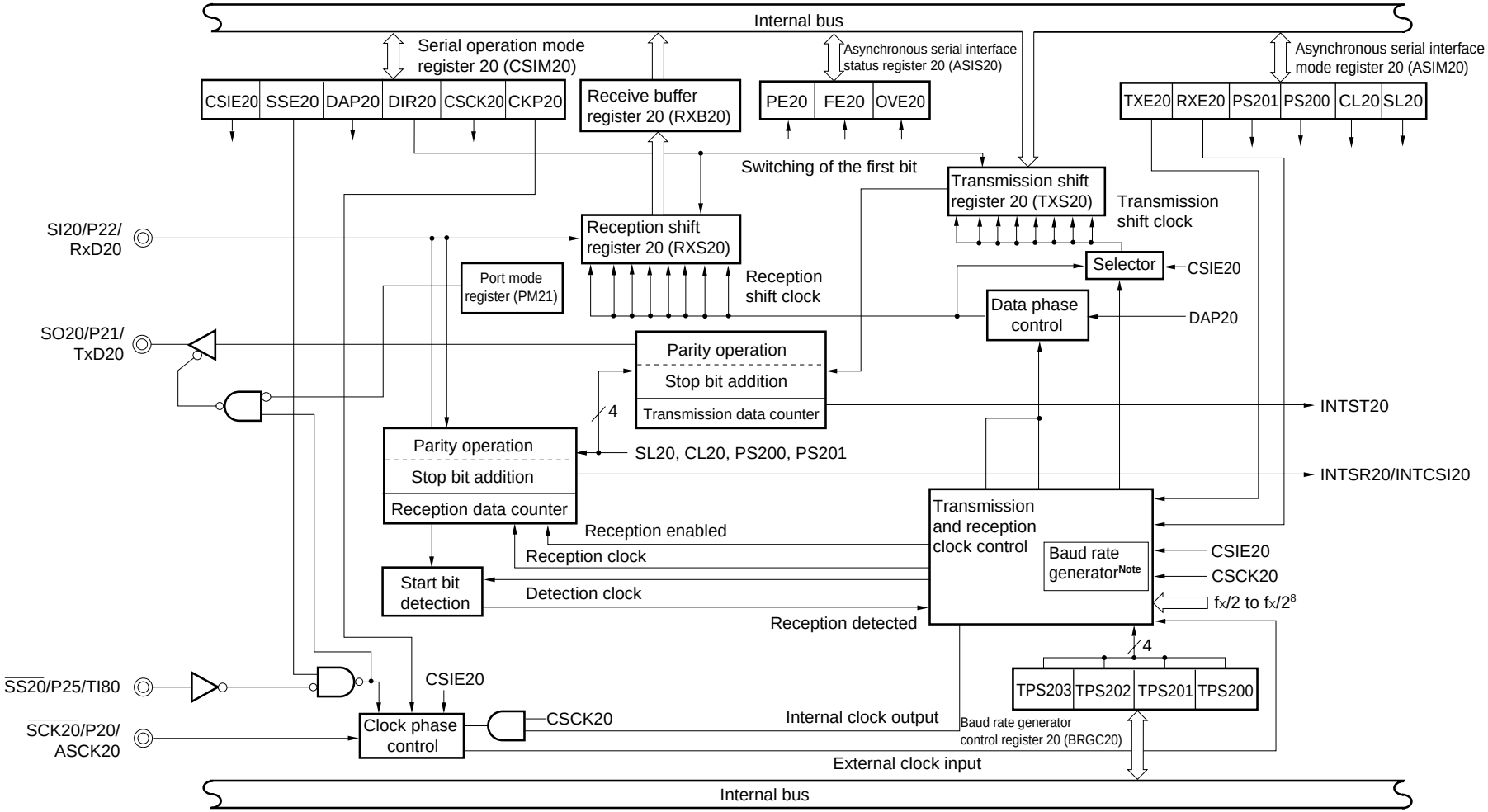
9.2 Serial Interface 20 Configuration

Serial interface 20 includes the following hardware.

Table 9-1. Configuration of Serial Interface 20

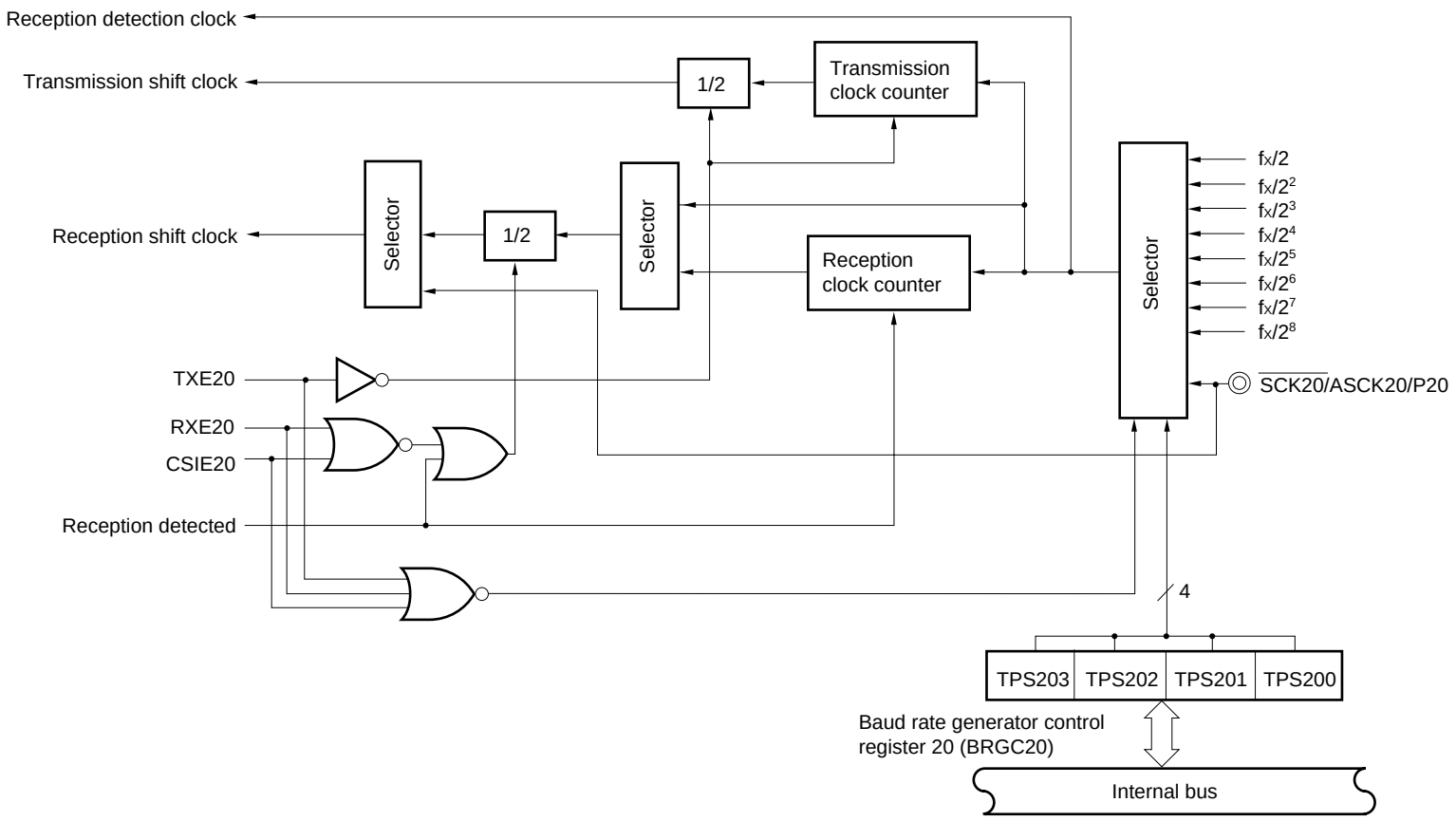
Item	Configuration
Registers	Transmission shift register 20 (TXS20) Reception shift register 20 (RXS20) Receive buffer register 20 (RXB20)
Control registers	Serial operation mode register 20 (CSIM20) Asynchronous serial interface mode register 20 (ASIM20) Asynchronous serial interface status register 20 (ASIS20) Baud rate generator control register 20 (BRGC20)

Figure 9-1. Block Diagram of Serial Interface 20



Note See **Figure 9-2** for the configuration of the baud rate generator.

Figure 9-2. Block Diagram of Baud Rate Generator 20



(1) Transmission shift register 20 (TXS20)

TXS20 is a register in which transmission data is prepared. The transmission data is output from TXS20 bit-serially.

When the data length is seven bits, bits 0 to 6 of the data in TXS20 will be transmission data. Writing data to TXS20 triggers transmission.

TXS20 can be written with an 8-bit memory manipulation instruction, but cannot be read.

$\overline{\text{RESET}}$ input sets TXS20 to FFH.

Caution Do not write to TXS20 during transmission.

TXS20 and receive buffer register 20 (RXB20) are mapped at the same address, so that any attempt to read from TXS20 results in a value being read from RXB20.

(2) Reception shift register 20 (RXS20)

RXS20 is a register in which serial data, received at the RxD20 pin, is converted to parallel data. Once one entire byte has been received, RXS20 feeds the reception data to receive buffer register 20 (RXB20).

RXS20 cannot be manipulated directly by a program.

(3) Receive buffer register 20 (RXB20)

RXB20 holds a reception data. A new reception data is transferred from reception shift register 20 (RXS20) every 1-byte data reception.

When the data length is seven bits, the reception data is sent to bits 0 to 6 of RXB20, in which the MSB is always fixed to 0.

RXB20 can be read with an 8-bit memory manipulation instruction, but cannot be written.

$\overline{\text{RESET}}$ input makes RXB20 undefined.

Caution RXB20 and transmission shift register 20 (TXS20) are mapped at the same address, so that any attempt to write to RXB20 results in a value being written to TXS20.

(4) Transmission controller

The transmission controller controls transmission. For example, it adds start, parity, and stop bits to the data in transmission shift register 20 (TXS20), according to the setting of asynchronous serial interface mode register 20 (ASIM20).

(5) Reception controller

The reception controller controls reception according to the setting of asynchronous serial interface mode register 20 (ASIM20). It also checks for errors, such as parity errors, during reception. If an error is detected, asynchronous serial interface status register 20 (ASIS20) is set according to the status of the error.

9.3 Serial Interface 20 Control Registers

Serial interface 20 is controlled by the following registers.

- Serial operation mode register 20 (CSIM20)
- Asynchronous serial interface mode register 20 (ASIM20)
- Asynchronous serial interface status register 20 (ASIS20)
- Baud rate generator control register 20 (BRGC20)

(1) Serial operation mode register 20 (CSIM20)

CSIM20 is set when serial interface 20 is used in 3-wire serial I/O mode.

CSIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears CSIM20 to 00H.

Figure 9-3. Format of Serial Operation Mode Register 20

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM20	CSIE20	SSE20	0	0	DAP20	DIR20	CSCK20	CKP20	FF72H	00H	R/W

CSIE20	3-wire serial I/O mode operation control
0	Operation disabled
1	Operation enabled

SSE20	$\overline{\text{SS20}}$ pin selection	Function of $\overline{\text{SS20}}$/P23 pin	Communication status
0	Not used	Port function	Communication enabled
1	Used	0	Communication enabled
		1	Communication disabled

DAP20	3-wire serial I/O mode data phase selection
0	Outputs at the falling edge of $\overline{\text{SCK20}}$.
1	Outputs at the rising edge of $\overline{\text{SCK20}}$.

DIR20	First-bit specification
0	MSB
1	LSB

CSCK20	3-wire serial I/O mode clock selection
0	External clock input to the $\overline{\text{SCK20}}$ pin
1	Output of the dedicated baud rate generator

CKP20	3-wire serial I/O mode clock phase selection
0	Clock is active low, and $\overline{\text{SCK20}}$ is at high level in the idle state.
1	Clock is active high, and $\overline{\text{SCK20}}$ is at low level in the idle state.

Cautions 1. Bits 4 and 5 must both be set to 0.

2. CSIM20 must be cleared to 00H, if UART mode is selected.

(2) Asynchronous serial interface mode register 20 (ASIM20)

ASIM20 is set when serial interface 20 is used in asynchronous serial interface mode.

ASIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ASIM20 to 00H.

Figure 9-4. Format of Asynchronous Serial Interface Mode Register 20

Symbol	<7>	<6>	5	4	3	2	1	0	Address	After reset	R/W
ASIM20	TXE20	RXE20	PS201	PS200	CL20	SL20	0	0	FF70H	00H	R/W

TXE20	Transmit operation control
0	Transmit operation stop
1	Transmit operation enable

RXE20	Receive operation control
0	Receive operation stop
1	Receive operation enable

PS201	PS200	Parity bit specification
0	0	No parity
0	1	Always add 0 parity at transmission. Parity check is not performed at reception (No parity error is generated).
1	0	Odd parity
1	1	Even parity

CL20	Transmit data character length specification
0	7 bits
1	8 bits

SL20	Transmit data stop bit length
0	1 bit
1	2 bits

Cautions 1. Bits 0 and 1 must both be set to 0.

2. If 3-wire serial I/O mode is selected, ASIM20 must be cleared to 00H.

3. Switch operating modes after halting the serial transmit/receive operation.

Table 9-2. Serial Interface 20 Operating Mode Settings

(1) Operation stop mode

ASIM20		CSIM20			PM22	P22	PM21	P21	PM20	P20	First Bit	Shift Clock	P22/SI20/ RxD20 Pin Function	P21/SO20/ TxD20 Pin Function	P20/$\overline{\text{SCK20}}$/ ASCK20 Pin Function
TXE20	RXE20	CSIE20	DIR20	CSCK20											
0	0	0	×	×	×Note 1	×Note 1	×Note 1	×Note 1	×Note 1	×Note 1	–	–	P22	P21	P20
Other than above											Setting prohibited				

(2) 3-wire serial I/O mode

ASIM20		CSIM20			PM22	P22	PM21	P21	PM20	P20	First Bit	Shift Clock	P22/SI20/ RxD20 Pin Function	P21/SO20/ TxD20 Pin Function	P20/$\overline{\text{SCK20}}$/ ASCK20 Pin Function
TXE20	RXE20	CSIE20	DIR20	CSCK20											
0	0	1	0	0	×Note 2	×Note 2	0	1	1	×	MSB	External clock	SI20Note 2	SO20 (CMOS output)	$\overline{\text{SCK20}}$ input
				1					0	1		Internal clock			$\overline{\text{SCK20}}$ output
		1	1	0					1	×	LSB	External clock			$\overline{\text{SCK20}}$ input
				1					0	1		Internal clock			$\overline{\text{SCK20}}$ output
Other than above											Setting prohibited				

(3) Asynchronous serial interface mode

ASIM20		CSIM20			PM22	P22	PM21	P21	PM20	P20	First Bit	Shift Clock	P22/SI20/ RxD20 Pin Function	P21/SO20/ TxD20 Pin Function	P20/$\overline{\text{SCK20}}$/ ASCK20 Pin Function
TXE20	RXE20	CSIE20	DIR20	CSCK20											
1	0	0	0	0	×Note 1	×Note 1	0	1	1	×	LSB	External clock	P22	TxD20 (CMOS output)	ASCK20 input
									×Note 1	×Note 1		Internal clock			P20
0	1	0	0	0	1	×	×Note 1	×Note 1	1	×		External clock	RxD20	P21	ASCK20 input
									×Note 1	×Note 1		Internal clock			P20
1	1	0	0	0	1	×	0	1	1	×		External clock		TxD20 (CMOS output)	ASCK20 input
									×Note 1	×Note 1		Internal clock			P20
Other than above											Setting prohibited				

Notes 1. These pins can be used for port functions.

2. When only transmission is used, this pin can be used as P22 (CMOS input/output).

Remark ×: don't care.

(3) Asynchronous serial interface status register 20 (ASIS20)

ASIS20 indicates the type of a reception error, if it occurs while asynchronous serial interface mode is set.

ASIS20 is read with a 1-bit or 8-bit memory manipulation instruction.

The contents of ASIS20 are undefined in 3-wire serial I/O mode.

$\overline{\text{RESET}}$ input clears ASIS20 to 00H.

Figure 9-5. Format of Asynchronous Serial Interface Status Register 20

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
ASIS20	0	0	0	0	0	PE20	FE20	OVE20	FF71H	00H	R

PE20	Parity error flag
0	No parity error has occurred.
1	A parity error has occurred (parity mismatch in transmission data).

FE20	Flaming error flag
0	No framing error has occurred.
1	A framing error has occurred (no stop bit detected).Note 1

OVE20	Overrun error flag
0	No overrun error has occurred.
1	An overrun error has occurredNote 2. (Before data was read from the receive buffer register, the subsequent receive operation was completed.)

Notes 1. Even when the stop bit length is set to 2 bits by setting bit 2 (SL20) of asynchronous serial interface mode register 20 (ASIM20), the stop bit detection at reception is performed with 1 bit.

2. Be sure to read receive buffer register 20 (RXB20) when an overrun error occurs. If not, every time the data is received an overrun error is generated.

(4) Baud rate generator control register 20 (BRGC20)

BRGC20 is used to specify the serial clock for serial interface 20.

BRGC20 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears BRGC20 to 00H.

Figure 9-6. Format of Baud Rate Generator Control Register 20

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
BRGC20	TPS203	TPS202	TPS201	TPS200	0	0	0	0	FF73H	00H	R/W

TPS203	TPS202	TPS201	TPS200	3-bit counter source clock selection	n
0	0	0	0	$f_X/2$ (2.5 MHz)	1
0	0	0	1	$f_X/2^2$ (1.25 MHz)	2
0	0	1	0	$f_X/2^3$ (625 kHz)	3
0	0	1	1	$f_X/2^4$ (313 kHz)	4
0	1	0	0	$f_X/2^5$ (156 kHz)	5
0	1	0	1	$f_X/2^6$ (78.1 kHz)	6
0	1	1	0	$f_X/2^7$ (39.1 kHz)	7
0	1	1	1	$f_X/2^8$ (19.5 kHz)	8
1	0	0	0	External clock input to the ASCK20 pin[Note]	—
Other than above				Setting prohibited	

Note An external clock can be used only in UART mode.

Cautions 1. When writing to BRGC20 is performed during a communication operation, the output of the baud rate generator is disrupted and communication cannot be performed normally. Be sure not to write to BRGC20 during a communication operation.

2. Be sure not to select n = 1 during operation at f_X = 5.0 MHz because the resulting baud rate exceeds the rated range.

3. When the external input clock is selected, set port mode register 2 (PM2) to input mode.

Remarks 1. f_X: System clock oscillation frequency

2. n: Value determined by setting TPS200 through TPS203 ($1 \leq n \leq 8$)

3. The parenthesized values apply to operation at f_X = 5.0 MHz.

The baud rate transmit/receive clock to be generated is either a signal scaled from the system clock, or a signal scaled from the clock input from the ASCK20 pin.

(a) Generation of baud rate transmit/receive clock form system clock

The transmit/receive clock is generated by scaling the system clock. The baud rate of a clock generated from the system clock is estimated by using the following expression.

$$[\text{Baud rate}] = \frac{f_X}{2^{n+1} \times 8} \text{ [Hz]}$$

f_X: System clock oscillation frequency

n: Value determined by values of TPS200 through TPS203 as shown in Figure 9-6 ($2 \leq n \leq 8$)

Table 9-3. Example of Relationship Between System Clock and Baud Rate

Baud Rate (bps)	n	BRGC20 Set Value	Error (%)	
			f_X = 5.0 MHz	f_X = 4.9152 MHz
1,200	8	70H	1.73	0
2,400	7	60H		
4,800	6	50H		
9,600	5	40H		
19,200	4	30H		
38,400	3	20H		
76,800	2	10H		

Caution Do not select n = 1 during operation at f_X = 5.0 MHz because the resulting baud rate exceeds the rated range.

(b) Generation of baud rate transmit/receive clock from external clock input from ASCK20 pin

The transmit/receive clock is generated by scaling the clock input from the ASCK20 pin. The baud rate of a clock generated from the clock input from the ASCK20 pin is estimated by using the following expression.

$$[\text{Baud rate}] = \frac{f_{ASCK}}{16} [\text{Hz}]$$

f_{ASCK}: Frequency of clock input from the ASCK20 pin

Table 9-4. Relationship Between ASCK20 Pin Input Frequency and Baud Rate (When BRGC20 Is Set to 80H)

Baud Rate (bps)	ASCK20 Pin Input Frequency (kHz)
75	1.2
150	2.4
300	4.8
600	9.6
1,200	19.2
2,400	38.4
4,800	76.8
9,600	153.6
19,200	307.2
31,250	500.0
38,400	614.4

9.4 Serial Interface 20 Operation

Serial interface 20 provides the following three types of modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode
- 3-wire serial I/O mode

9.4.1 Operation stop mode

In operation stop mode, serial transfer is not executed; therefore, the power consumption can be reduced. The P20/$\overline{\text{SCK20}}$/ASCK20, P21/SO20/TxD20, and P22/SI20/RxD20 pins can be used as normal I/O ports.

(1) Register setting

Operation stop mode is set by serial operation mode register 20 (CSIM20) and asynchronous serial interface mode register 20 (ASIM20).

(a) Serial operation mode register 20 (CSIM20)

CSIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears CSIM20 to 00H.

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM20	CSIE20	SSE20	0	0	DAP20	DIR20	CSCK20	CKP20	FF72H	00H	R/W

CSIE20	3-wire serial I/O mode operation control
0	Operation disabled
1	Operation enabled

Caution Bits 4 and 5 must both be set to 0.

(b) Asynchronous serial interface mode register 20 (ASIM20)

ASIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ASIM20 to 00H.

Symbol	<7>	<6>	5	4	3	2	1	0	Address	After reset	R/W
ASIM20	TXE20	RXE20	PS201	PS200	CL20	SL20	0	0	FF70H	00H	R/W

TXE20	Transmit operation control
0	Transmit operation stop
1	Transmit operation enable

RXE20	Receive operation control
0	Receive operation stop
1	Receive operation enable

Caution Bits 0 and 1 must both be set to 0.

9.4.2 Asynchronous serial interface (UART) mode

In this mode, the one-byte data following the start bit is transmitted/received and thus full-duplex communication is possible.

This device incorporates a UART-dedicated baud rate generator that enables communication at a desired baud rate from many options. In addition, the baud rate can also be defined by dividing the clock input to the ASCK20 pin.

The UART-dedicated baud rate generator also can output the 31.25 kbps baud rate that complies with the MIDI standard.

(1) Register setting

UART mode is set by serial operation mode register 20 (CSIM20), asynchronous serial interface mode register 20 (ASIM20), asynchronous serial interface status register 20 (ASIS20), and baud rate generator control register 20 (BRGC20).

(a) Serial operation mode register 20 (CSIM20)

CSIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears CSIM20 to 00H.

Set CSIM20 to 00H when UART mode is selected.

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM20	CSIE20	SSE20	0	0	DAP20	DIR20	CSCK20	CKP20	FF72H	00H	R/W

CSIE20	3-wire serial I/O mode operation control
0	Operation disabled
1	Operation enabled

SSE20	$\overline{\text{SS20}}$ pin selection	Function of $\overline{\text{SS20}}$/P23 pin	Communication status
0	Not used	Port function	Communication enabled
1	Used	0	Communication enabled
		1	Communication disabled

DAP20	3-wire serial I/O mode data phase selection
0	Outputs at the falling edge of $\overline{\text{SCK20}}$.
1	Outputs at the rising edge of $\overline{\text{SCK20}}$.

DIR20	First-bit specification
0	MSB
1	LSB

CSCK20	3-wire serial I/O mode clock selection
0	External clock input to the $\overline{\text{SCK20}}$ pin
1	Output of the dedicated baud rate generator

CKP20	3-wire serial I/O mode clock phase selection
0	Clock is active low, and $\overline{\text{SCK20}}$ is high level in the idle state.
1	Clock is active high, and $\overline{\text{SCK20}}$ is low level in the idle state.

Caution Bits 4 and 5 must both be set to 0.

(b) Asynchronous serial interface mode register 20 (ASIM20)

ASIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ASIM20 to 00H.

Symbol	<7>	<6>	5	4	3	2	1	0	Address	After reset	R/W
ASIM20	TXE20	RXE20	PS201	PS200	CL20	SL20	0	0	FF70H	00H	R/W

TXE20	Transmit operation control
0	Transmit operation stopped
1	Transmit operation enabled

RXE20	Receive operation control
0	Receive operation stopped
1	Receive operation enabled

PS201	PS200	Parity bit specification
0	0	No parity
0	1	Always add 0 parity at transmission. Parity check is not performed at reception. (No parity error is generated.)
1	0	Odd parity
1	1	Even parity

CL20	Character length specification
0	7 bits
1	8 bits

SL20	Transmit data stop bit length specification
0	1 bit
1	2 bits

Cautions 1. Bits 0 and 1 must both be set to 0.

2. Switch operating modes after halting the serial transmit/receive operation.

(c) Asynchronous serial interface status register 20 (ASIS20)

ASIS20 is read with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ASIS20 to 00H.

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
ASIS20	0	0	0	0	0	PE20	FE20	OVE20	FF71H	00H	R

PE20	Parity error flag
0	Parity error not generated
1	Parity error generated (when the parity of transmit data does not match)

FE20	Flaming error flag
0	Framing error not generated
1	Framing error generated (when stop bit is not detected)[Note 1]

OVE20	Overrun error flag
0	Overrun error not generated
1	Overrun error generated[Note 2] (when the next receive operation is completed before the data is read from the receive buffer register)

Notes 1. Even when the stop bit length is set to 2 bits by setting bit 2 (SL20) of asynchronous serial interface mode register 20 (ASIM20), the stop bit detection at reception is performed with 1 bit.

2. Be sure to read receive buffer register 20 (RXB20) when an overrun error occurs. If not, every time the data is received an overrun error is generated.

(d) Baud rate generator control register 20 (BRGC20)

BRGC20 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears BRGC20 to 00H.

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
BRGC20	TPS203	TPS202	TPS201	TPS200	0	0	0	0	FF73H	00H	R/W

TPS203	TPS202	TPS201	TPS200	3-bit counter source clock selection	n
0	0	0	0	$f_X/2$ (2.5 MHz)	1
0	0	0	1	$f_X/2^2$ (1.25 MHz)	2
0	0	1	0	$f_X/2^3$ (625 kHz)	3
0	0	1	1	$f_X/2^4$ (313 kHz)	4
0	1	0	0	$f_X/2^5$ (156 kHz)	5
0	1	0	1	$f_X/2^6$ (78.1 kHz)	6
0	1	1	0	$f_X/2^7$ (39.1 kHz)	7
0	1	1	1	$f_X/2^8$ (19.5 kHz)	8
1	0	0	0	External clock input to ASCK20 pin	—
Other than above				Setting prohibited	

Cautions 1. When writing to BRGC20 is performed during a communication operation, the output of the baud rate generator is disrupted and communication cannot be performed normally. Be sure not to write to BRGC20 during a communication operation.

2. Be sure not to select n = 1 during operation at f_X = 5.0 MHz because the resulting baud rate exceeds the rated range.

3. When the external input clock is selected, set port mode register 2 (PM2) to input mode.

Remarks 1. f_X: System clock oscillation frequency

2. n: Value determined by setting TPS200 through TPS203 ($1 \leq n \leq 8$)

3. The parenthesized values apply to operation at f_X = 5.0 MHz.

The baud rate transmit/receive clock to be generated is either a signal scaled from the system clock, or a signal scaled from the clock input from the ASCK20 pin.

(i) Generation of baud rate transmit/receive clock from system clock

The transmit/receive clock is generated by scaling the system clock. The baud rate of the clock generated from the system clock is estimated by using the following expression.

$$[\text{Baud rate}] = \frac{f_X}{2^{n+1} \times 8} \text{ [Hz]}$$

f_X: System clock oscillation frequency

n: Value determined by setting TPS200 through TPS203 as shown in the above table ($2 \leq n \leq 8$)

Table 9-5. Example of Relationship Between System Clock and Baud Rate

Baud Rate (bps)	n	BRGC20 Set Value	Error (%)	
			f_X = 5.0 MHz	f_X = 4.9152 MHz
1,200	8	70H	1.73	0
2,400	7	60H		
4,800	6	50H		
9,600	5	40H		
19,200	4	30H		
38,400	3	20H		
76,800	2	10H		

Caution Do not select n = 1 during operation at f_X = 5.0 MHz because the resulting baud rate exceeds the rated range.

(ii) Generation of baud rate transmit/receive clock from external clock input from ASCK20 pin

The transmit/receive clock is generated by scaling the clock input from the ASCK20 pin. The baud rate of the clock generated from the clock input from the ASCK20 pin is estimated by using the following expression.

$$[\text{Baud rate}] = \frac{f_{ASCK}}{16} \text{ [Hz]}$$

f_{ASCK}: Frequency of clock input from the ASCK20 pin

Table 9-6. Relationship Between ASCK20 Pin Input Frequency and Baud Rate (When BRGC20 Is Set to 80H)

Baud Rate (bps)	ASCK20 Pin Input Frequency (kHz)
75	1.2
150	2.4
300	4.8
600	9.6
1,200	19.2
2,400	38.4
4,800	76.8
9,600	153.6
19,200	307.2
31,250	500.0
38,400	614.4

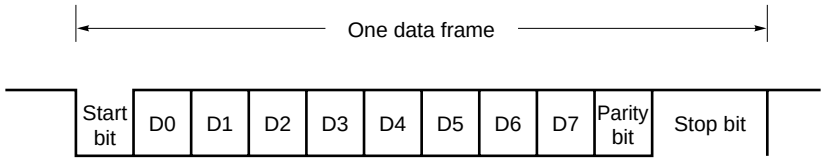
(2) Communication operation

(a) Data format

The transmit/receive data format is as shown in Figure 9-7. One data frame consists of a start bit, character bits, a parity bit, and stop bit(s).

The specification of the character bit length in one data frame, parity selection, and specification of the stop bit length is carried out with asynchronous serial interface mode register 20 (ASIM20).

Figure 9-7. Format of Asynchronous Serial Interface Transmit/Receive Data



- Start bits ..... 1 bit
- Character bits ..... 7 bits/8 bits
- Parity bits..... Even parity/odd parity/0 parity/no parity
- Stop bit(s) ..... 1 bit/2 bits

When 7 bits are selected as the number of character bits, only the lower 7 bits (bits 0 to 6) are valid; in transmission the most significant bit (bit 7) is ignored, and in reception the most significant bit (bit 7) is always "0".

The serial transfer rate is selected by ASIM20 and baud rate generator control register 20 (BRGC20).

If a serial data receive error is generated, the receive error contents can be determined by reading the status of asynchronous serial interface status register 20 (ASIS20).

(b) Parity types and operation

The parity bit is used to detect a bit error in the communication data. Normally, the same kind of parity bit is used on the transmitting side and the receiving side. With even parity and odd parity, a one-bit (odd number) error can be detected. With 0 parity and no parity, an error cannot be detected.

(i) Even parity

- **At transmission**

  The parity bit is determined so that the number of bits with a value of "1" in the transmit data including the parity bit may be even. The parity bit value should be as follows.

  The number of bits with a value of "1" is an odd number in transmit data: 1
  The number of bits with a value of "1" is an even number in transmit data: 0

- **At reception**

  The number of bits with a value of "1" in the receive data including the parity bit is counted, and if the number is odd, a parity error is generated.

(ii) Odd parity

- **At transmission**

  Conversely to even parity, the parity bit is determined so that the number of bits with a value of "1" in the transmit data including the parity bit may be odd. The parity bit value should be as follows.

  The number of bits with a value of "1" is an odd number in transmit data: 0
  The number of bits with a value of "1" is an even number in transmit data: 1

- **At reception**

  The number of bits with a value of "1" in the receive data including the parity bit is counted, and if the number is even, a parity error is generated.

(iii) 0 parity

When transmitting, the parity bit is set to "0" irrespective of the transmit data.

At reception, a parity bit check is not performed. Therefore, a parity error is not generated, irrespective of whether the parity bit is set to "0" or "1".

(iv) No parity

A parity bit is not added to the transmit data.

At reception, data is received assuming that there is no parity bit. Since there is no parity bit, a parity error is not generated.

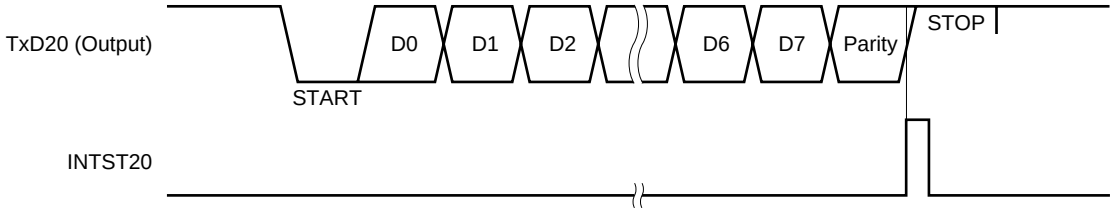
(c) Transmission

A transmit operation is started by writing transmit data to transmission shift register 20 (TXS20). The start bit, parity bit, and stop bit(s) are added automatically.

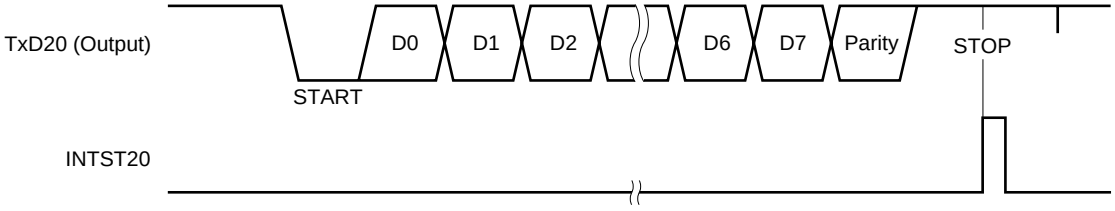
When the transmit operation starts, the data in TXS20 is shifted out, and when TXS20 is empty, a transmission completion interrupt (INTST20) is generated.

Figure 9-8. Asynchronous Serial Interface Transmission Completion Interrupt Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



Caution Do not rewrite asynchronous serial interface mode register 20 (ASIM20) during a transmit operation. If the ASIM20 register is rewritten during transmission, subsequent transmission may not be performed (the normal state is restored by $\overline{\text{RESET}}$ input).

It is possible to determine whether transmission is in progress by software by using a transmission completion interrupt (INTST20) or the interrupt request flag (STIF20) set by INTST20.

(d) Reception

When bit 6 (RXE20) of asynchronous serial interface mode register 20 (ASIM20) is set to 1, a receive operation is enabled and sampling of the RxD20 pin input is performed.

RxD20 pin input sampling is performed using the serial clock specified by ASIM20.

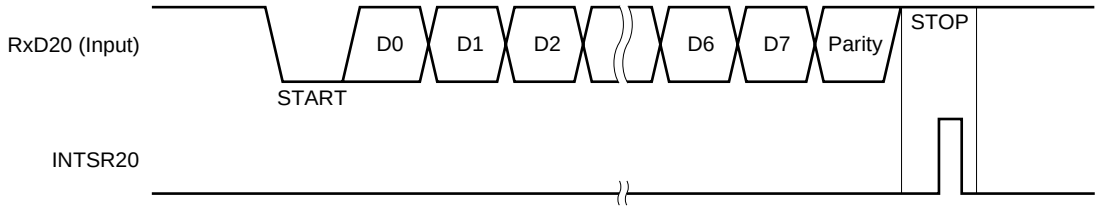
When the RxD20 pin input becomes low, the 3-bit counter starts counting, and at the time when half the time determined by the specified baud rate has passed, the data sampling start timing signal is output. If the RxD20 pin input sampled again as a result of this start timing signal is low, it is identified as a start bit, the 3-bit counter is initialized and starts counting, and data sampling is performed. When character data, a parity bit, and one stop bit are detected after the start bit, reception of one frame of data ends.

When one frame of data has been received, the receive data in the shift register is transferred to receive buffer register 20 (RXB20), and a reception completion interrupt (INTSR20) is generated.

If an error is generated, the receive data in which the error was generated is still transferred to RXB20, and INTSR20 is generated.

If the RXE20 bit is reset to 0 during the receive operation, the receive operation is stopped immediately. In this case, the contents of RXB20 and asynchronous serial interface status register 20 (ASIS20) are not changed, and INTSR20 is not generated.

Figure 9-9. Asynchronous Serial Interface Reception Completion Interrupt Timing



Caution Be sure to read receive buffer register 20 (RXB20) even if a receive error occurs. If RXB20 is not read, an overrun error will be generated when the next data is received, and the receive error state will continue indefinitely.

(e) Receive errors

The following three errors may occur during a receive operation: a parity error, a framing error, and an overrun error. After data reception, an error flag is set in asynchronous serial interface status register 20 (ASIS20). Receive error causes are shown in Table 9-7.

It is possible to determine what kind of error was generated during reception by reading the contents of ASIS20 in the reception error interrupt servicing (see **Figures 9-9** and **9-10**).

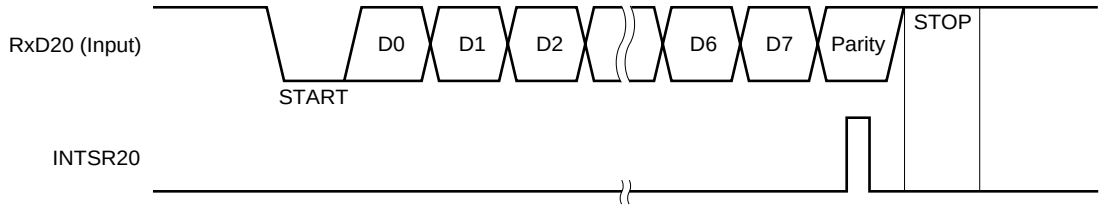
The contents of ASIS20 are reset to 0 by reading receive buffer register 20 (RXB20) or receiving the next data (if there is an error in the next data, the corresponding error flag is set).

Table 9-7. Receive Error Causes

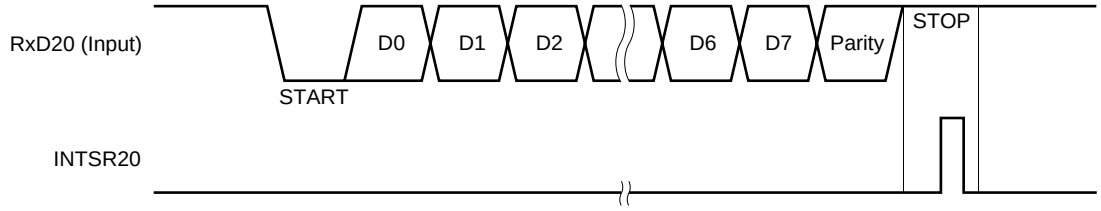
Receive Errors	Cause
Parity error	Transmission-time parity and reception data parity do not match.
Framing error	Stop bit not detected
Overrun error	Reception of next data is completed before data is read from receive buffer register.

Figure 9-10. Receive Error Timing

(a) Parity error generated



(b) Framing error or overrun error generated



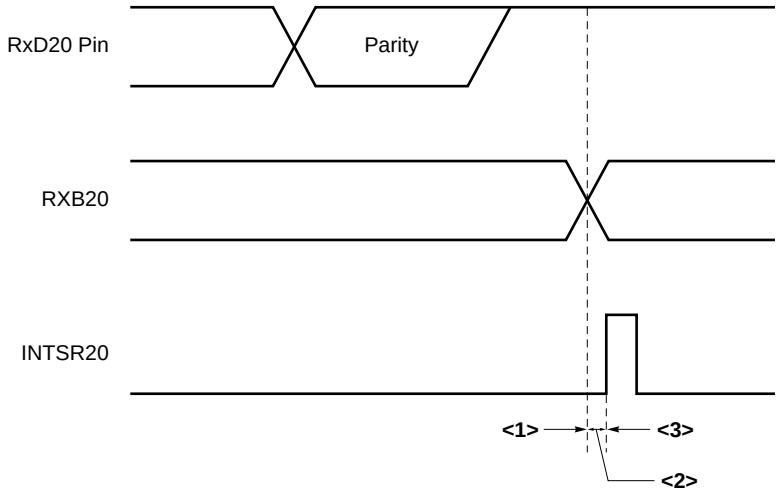
Cautions 1. The contents of the ASIS20 register are reset to 0 by reading receive buffer register 20 (RXB20) or receiving the next data. To ascertain the error contents, read ASIS20 before reading RXB20.

2. Be sure to read receive buffer register 20 (RXB20) even if a receive error is generated. If RXB20 is not read, an overrun error will be generated when the next data is received, and the receive error state will continue indefinitely.

(3) Cautions related to UART mode

(a) When bit 7 (TXE20) of asynchronous serial interface mode register 20 (ASIM20) is cleared during transmission, be sure to set transmission shift register 20 (TXS20) to FFH, then set TXE20 to 1 before executing the next transmission.

(b) When bit 6 (RXE20) of asynchronous serial interface mode register 20 (ASIM20) is cleared during reception, receive buffer register 20 (RXB20) and the receive completion interrupt (INTSR20) are as follows.



When RXE20 is set to 0 at a time indicated by **<1>**, RXB20 holds the previous data and INTSR20 is not generated.
When RXE20 is set to 0 at a time indicated by **<2>**, RXB20 renews the data and INTSR20 is not generated.
When RXE20 is set to 0 at a time indicated by **<3>**, RXB20 renews the data and INTSR20 is generated.

9.4.3 3-wire serial I/O mode

The 3-wire serial I/O mode is useful for connection of peripheral I/Os and display controllers, etc., that incorporate a conventional synchronous serial interface, such as the 75XL Series, 78K Series, 17K Series, etc.

Communication is performed using three lines: the serial clock ($\overline{SCK20}$), serial output (SO20), and serial input (SI20).

(1) Register setting

3-wire serial I/O mode settings are performed using serial operation mode register 20 (CSIM20), asynchronous serial interface mode register 20 (ASIM20), and baud rate generator control register 20 (BRGC20).

(a) Serial operation mode register 20 (CSIM20)

CSIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$ input clears CSIM20 to 00H.

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM20	CSIE20	SSE20	0	0	DAP20	DIR20	CSCK20	CKP20	FF72H	00H	R/W

CSIE20	3-wire serial I/O mode operation control
0	Operation disabled
1	Operation enabled

SSE20	$\overline{SS20}$ pin selection	Function of $\overline{SS20}$/P23 pin	Communication status
0	Not used	Port function	Communication enabled
1	Used	0	Communication enabled
		1	Communication disabled

DAP20	3-wire serial I/O mode data phase selection
0	Outputs at the falling edge of $\overline{SCK20}$.
1	Outputs at the rising edge of $\overline{SCK20}$.

DIR20	First-bit specification
0	MSB
1	LSB

CSCK20	3-wire serial I/O mode clock selection
0	External clock input to the $\overline{SCK20}$ pin
1	Output of the dedicated baud rate generator

CKP20	3-wire serial I/O mode clock phase selection
0	Clock is active low, and $\overline{SCK20}$ is at high level in the idle state.
1	Clock is active high, and $\overline{SCK20}$ is at low level in the idle state.

Caution Bits 4 and 5 must both be set to 0.

(b) Asynchronous serial interface mode register 20 (ASIM20)

ASIM20 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ASIM20 to 00H.

When 3-wire serial I/O mode is selected, ASIM20 must be set to 00H.

Symbol	<7>	<6>	5	4	3	2	1	0	Address	After reset	R/W
ASIM20	TXE20	RXE20	PS201	PS200	CL20	SL20	0	0	FF70H	00H	R/W

TXE20	Transmit operation control
0	Transmit operation stop
1	Transmit operation enable

RXE20	Receive operation control
0	Receive operation stop
1	Receive operation enable

PS201	PS200	Parity Bit specification
0	0	No parity
0	1	Always add 0 parity at transmission. Parity check is not performed at reception. (No parity error is generated.)
1	0	Odd parity
1	1	Even parity

CL20	Character length specification
0	7 bits
1	8 bits

SL20	Transmit data sop bit length specification
0	1 bit
1	2 bits

Cautions 1. Bits 0 and 1 must both be set to 0.

2. Switch operating modes after halting the serial transmit/receive operation.

(c) Baud rate generator control register 20 (BRGC20)

BRGC20 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears BRGC20 to 00H.

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
BRGC20	TPS203	TPS202	TPS201	TPS200	0	0	0	0	FF73H	00H	R/W

TPS203	TPS202	TPS201	TPS200	3-bit counter source clock selection	n
0	0	0	0	$f_X/2$ (2.5 MHz)	1
0	0	0	1	$f_X/2^2$ (1.25 MHz)	2
0	0	1	0	$f_X/2^3$ (625 kHz)	3
0	0	1	1	$f_X/2^4$ (313 kHz)	4
0	1	0	0	$f_X/2^5$ (156 kHz)	5
0	1	0	1	$f_X/2^6$ (78.1 kHz)	6
0	1	1	0	$f_X/2^7$ (39.1 kHz)	7
0	1	1	1	$f_X/2^8$ (19.5 kHz)	8
Other than above				Setting prohibited	

Cautions 1. When writing to BRGC20 is performed during a communication operation, the baud rate generator output is disrupted and communication cannot be performed normally. Be sure not to write to BRGC20 during a communication operation.

2. Be sure not to select n = 1 during operation at f_X = 5.0 MHz because the resulting baud rate exceeds the rated range.

3. When the external input clock is selected, set port mode register 2 (PM2) to input mode.

Remarks 1. f_X: System clock oscillation frequency

2. n: Value determined by setting TPS200 through TPS203 ($1 \leq n \leq 8$)

3. The parenthesized values apply to operation at f_X = 5.0 MHz.

If the internal clock is used as the serial clock for 3-wire serial I/O mode, set bits TPS200 to TPS203 to set the frequency of the serial clock. To obtain the frequency to be set, use the following expression. When an external serial clock is used, setting BRGC20 is not necessary.

$$\text{Serial clock frequency} = \frac{f_X}{2^{n+1}} \text{ [Hz]}$$

f_X: System clock oscillation frequency

n: Value determined by setting TPS200 to TPS203 as shown in the above table ($1 \leq n \leq 8$)

(2) Communication operation

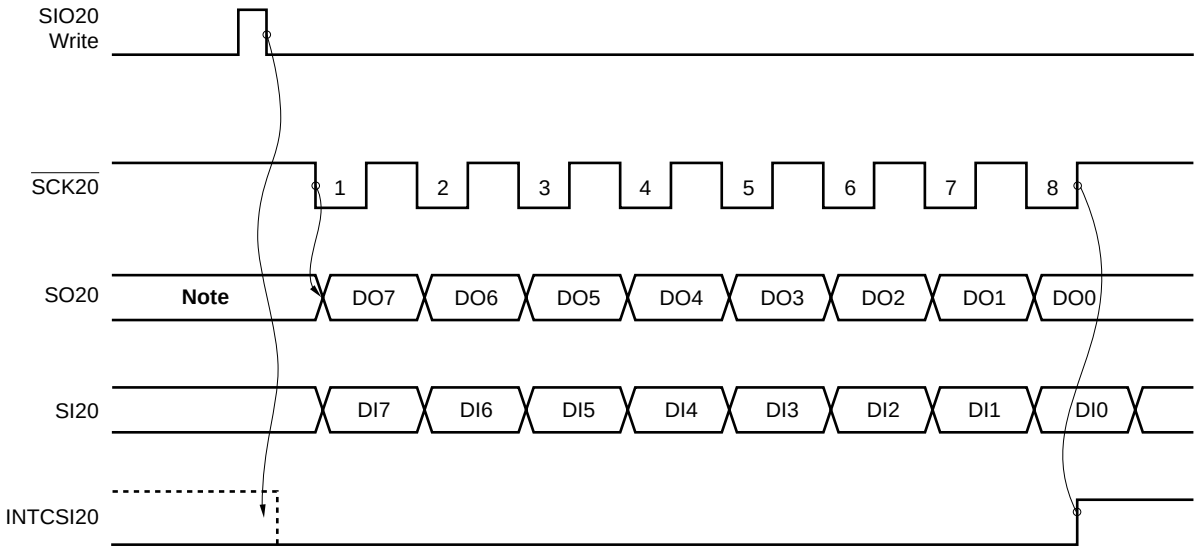
In 3-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in synchronization with the serial clock.

Transmission shift register 20 (TXS20/SIO20) and reception shift register 20 (RXS20) shift operations are performed in synchronization with the fall of the serial clock ($\overline{SCK20}$). Then transmit data is held in the SO20 latch and output from the SO20 pin. Also, receive data input to the SI20 pin is latched in receive buffer register 20 (RXB20/SIO20) on the rise of $\overline{SCK20}$.

At the end of an 8-bit transfer, the operation of TXS20/SIO20 and RXS20 stops automatically, and the interrupt request signal (INTCSI20) is generated.

Figure 9-11. 3-Wire Serial I/O Mode Timing (1/7)

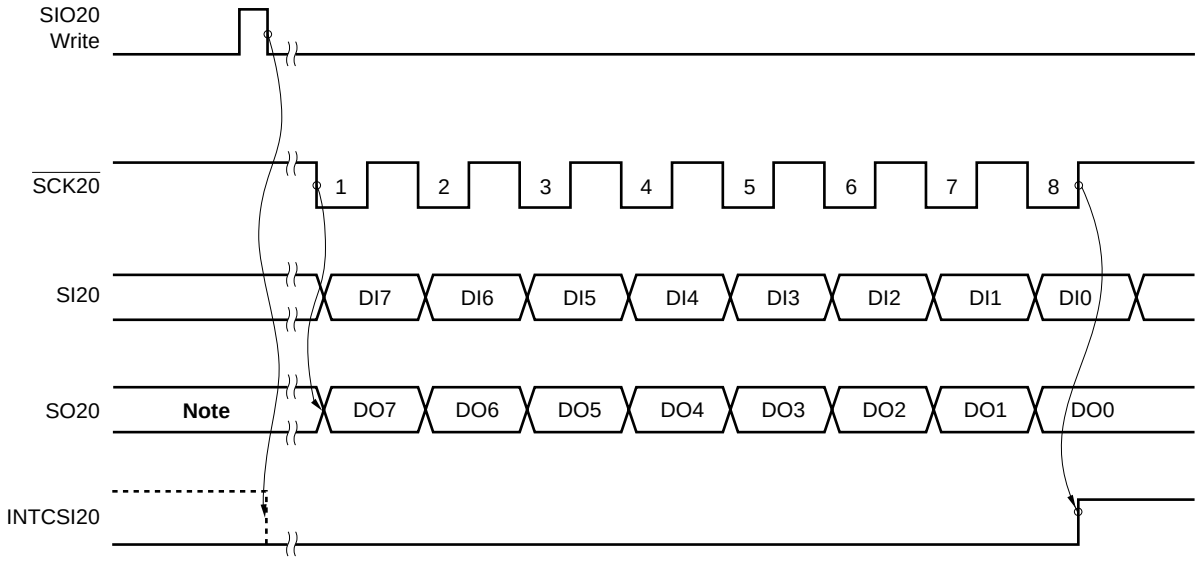
(i) Master operation timing (when DAP20 = 0, CKP20 = 0, SSE20 = 0)



Note The value of the last bit previously output is output.

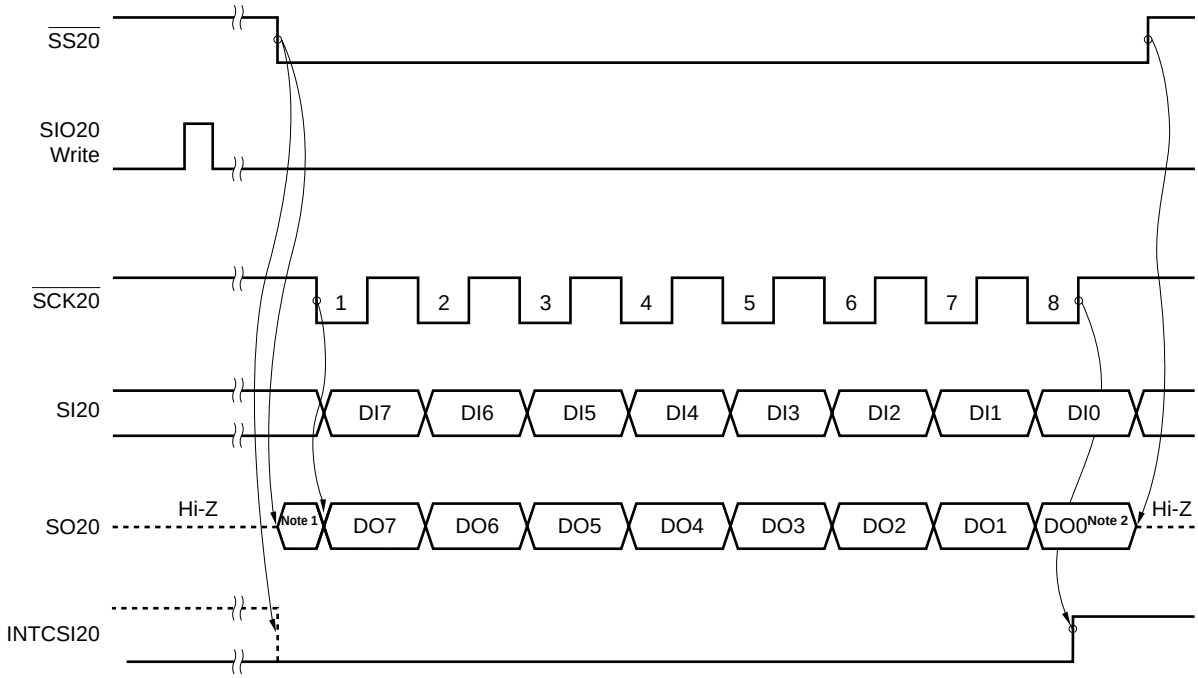
Figure 9-11. 3-Wire Serial I/O Mode Timing (2/7)

(ii) Slave operation timing (when DAP20 = 0, CKP20 = 0, SSE20 = 0)



Note The value of the last bit previously output is output.

(iii) Slave operation (when DAP20 = 0, CKP20 = 0, SSE20 = 1)

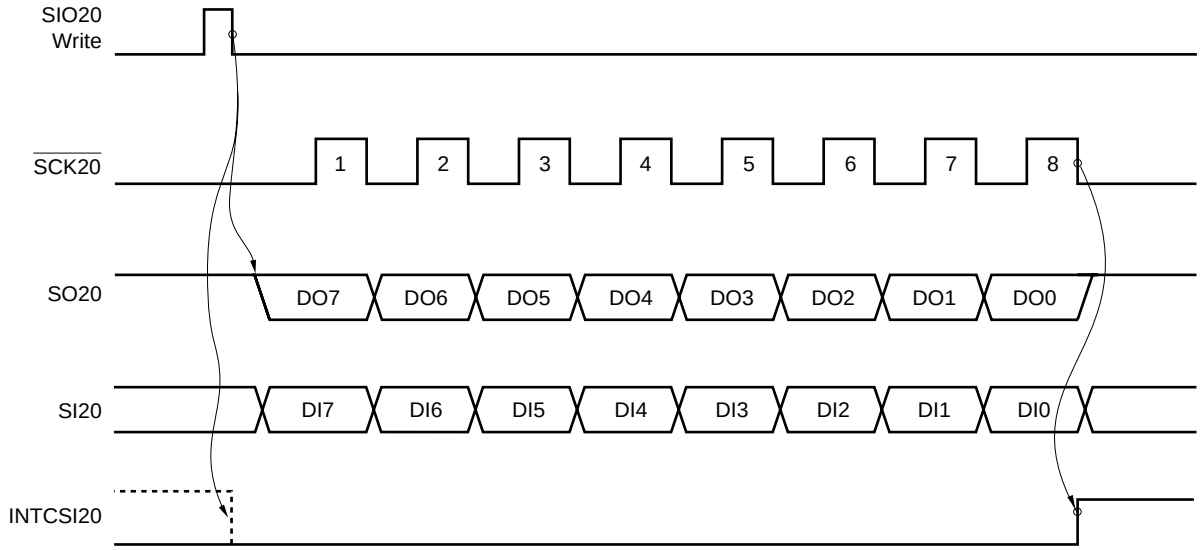


Notes 1. The value of the last bit previously output is output.

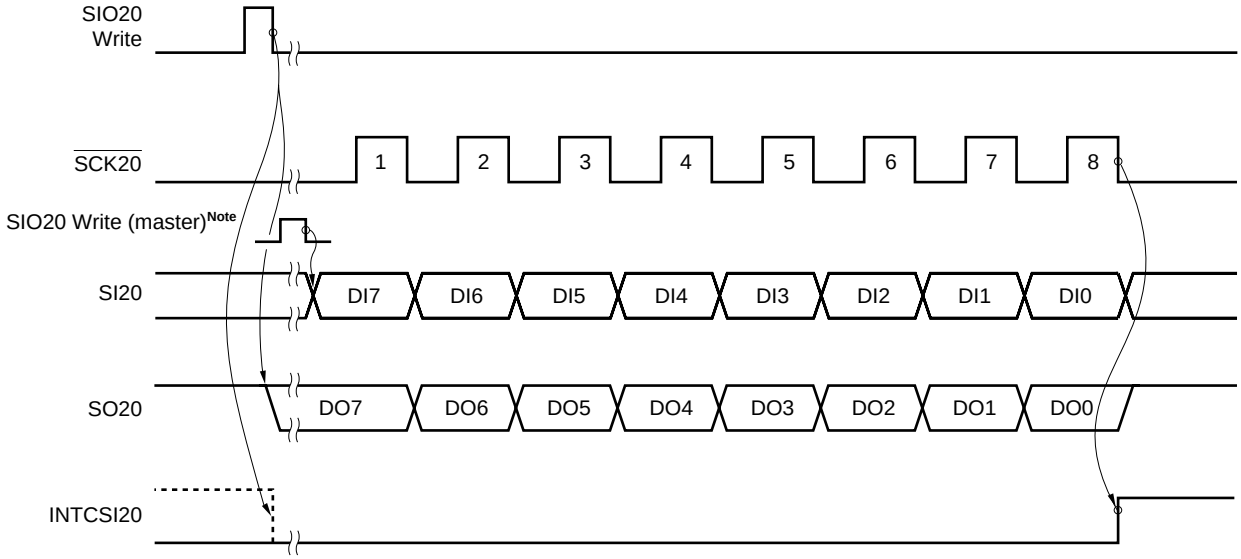
2. DO0 is output until $\overline{SS20}$ rises.
When $\overline{SS20}$ is high, SO20 is in a high-impedance state.

Figure 9-11. 3-Wire Serial I/O Mode Timing (3/7)

(iv) Master operation (when DAP20 = 0, CKP20 = 1, SSE20 = 0)



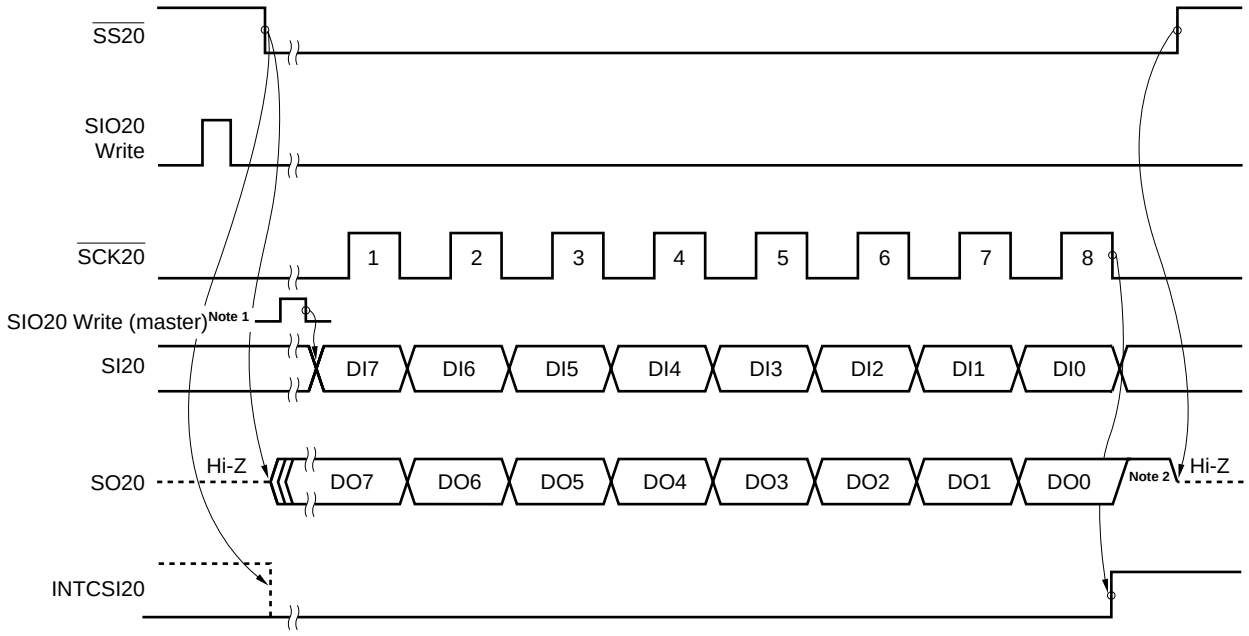
(v) Slave operation (when DAP20 = 0, CKP20 = 1, SSE20 = 0)



Note The data of SI20 is loaded at the first rising edge of $\overline{\text{SCK20}}$. Make sure that the master outputs the first bit before the first rising of $\overline{\text{SCK20}}$.

Figure 9-11. 3-Wire Serial I/O Mode Timing (4/7)

(vi) Slave operation (when DAP20 = 0, CKP20 = 1, SSE20 = 1)



Notes 1. The data of SI20 is loaded at the first rising edge of $\overline{\text{SCK20}}$. Make sure that the master outputs the first bit before the first rising of $\overline{\text{SCK20}}$.

2. SO20 is high until $\overline{\text{SS20}}$ rises after completion of DO0 output. When $\overline{\text{SS20}}$ is high, SO20 is in a high-impedance state.

(vii) Master operation (when DAP20 = 1, CKP20 = 0, SSE20 = 0)

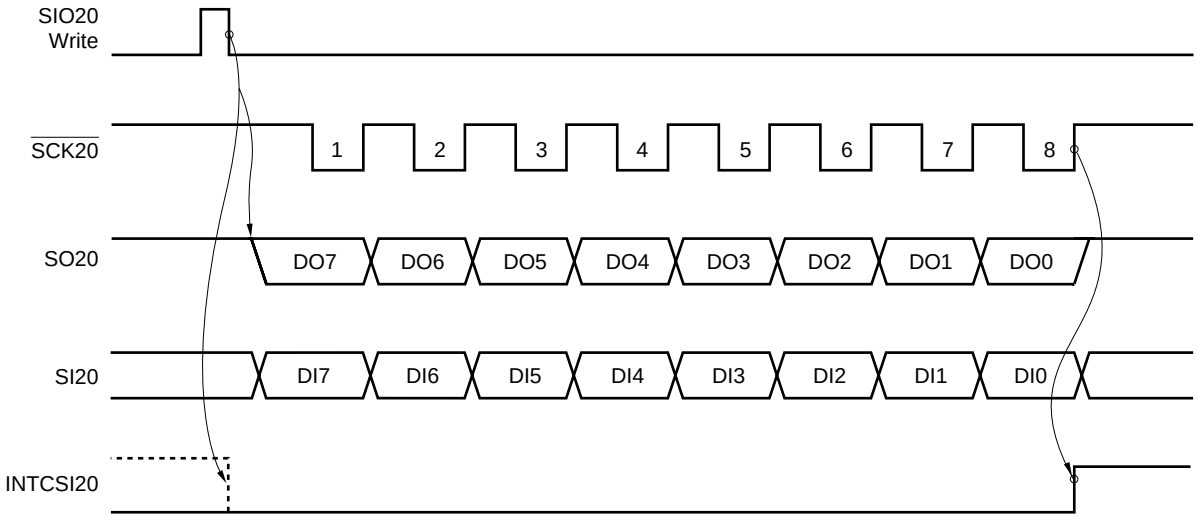
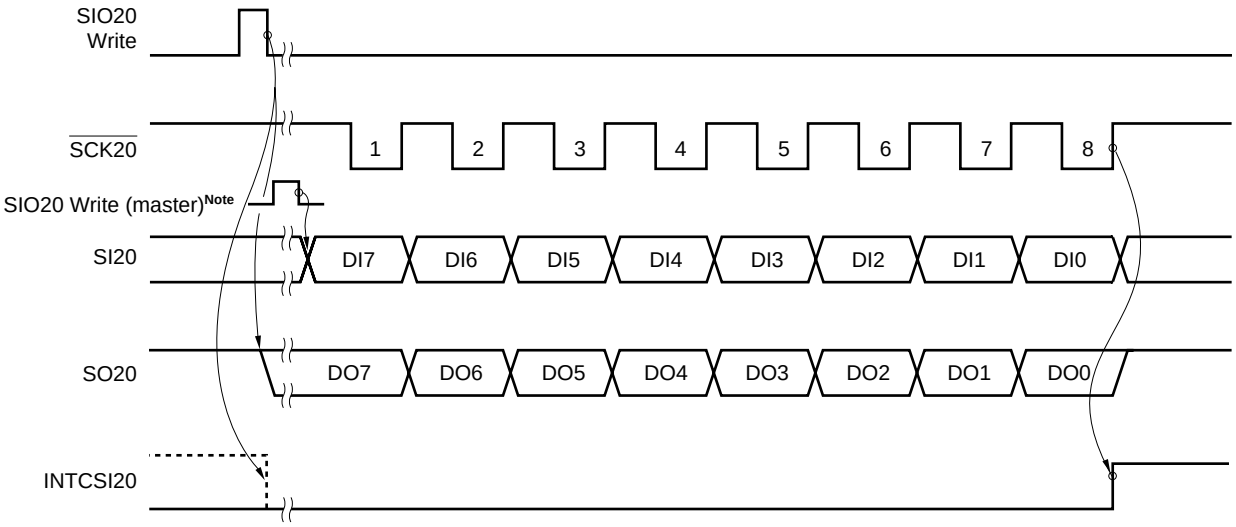


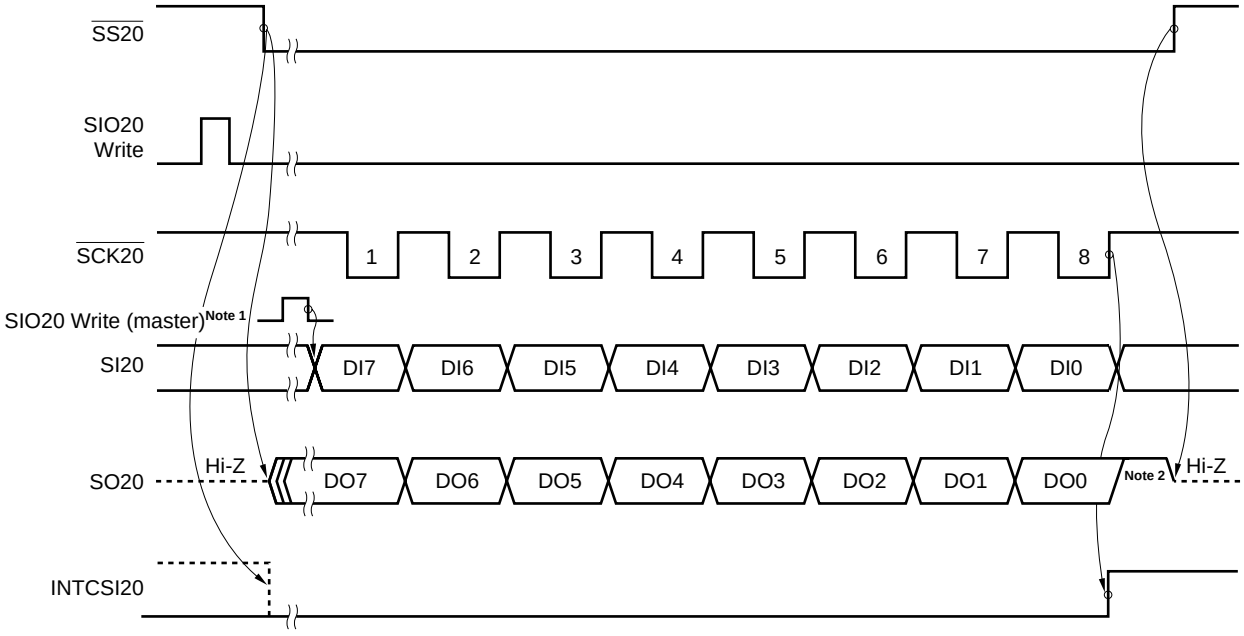
Figure 9-11. 3-Wire Serial I/O Mode Timing (5/7)

(viii) Slave operation (when DAP20 = 1, CKP20 = 0, SSE20 = 0)



Note The data of SI20 is loaded at the first falling edge of $\overline{\text{SCK20}}$. Make sure that the master outputs the first bit before the first falling of $\overline{\text{SCK20}}$.

(ix) Slave operation (when DAP20 = 1, CKP20 = 0, SSE20 = 1)

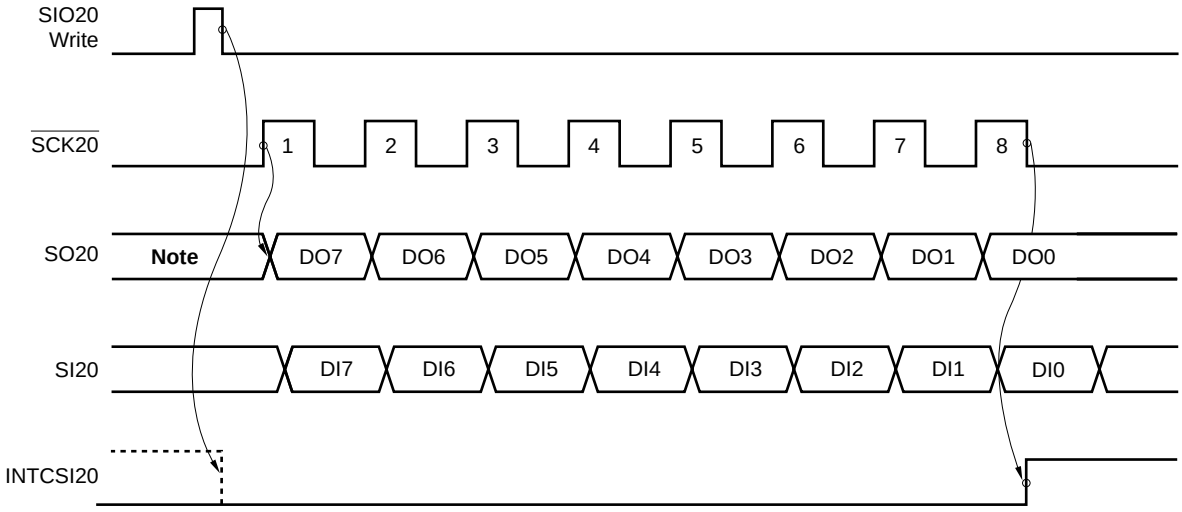


Notes 1. The data of SI20 is loaded at the first falling edge of $\overline{\text{SCK20}}$. Make sure that the master outputs the first bit before the first falling of $\overline{\text{SCK20}}$.

2. SO20 is high until $\overline{\text{SS20}}$ rises after completion of DO0 output. When $\overline{\text{SS20}}$ is high, SO20 is in a high-impedance state.

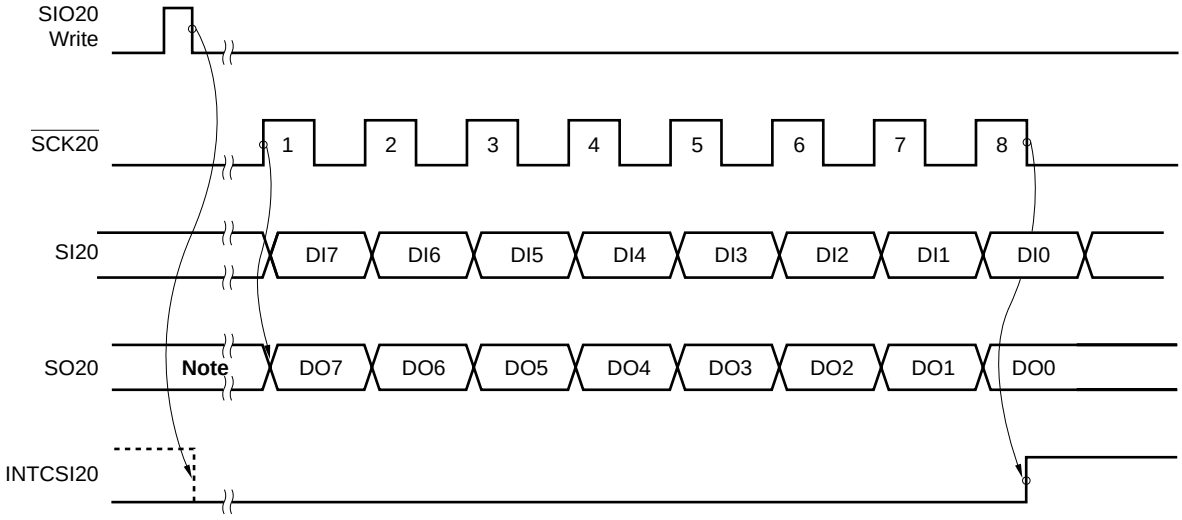
Figure 9-11. 3-Wire Serial I/O Mode Timing (6/7)

(x) Master operation (when DAP20 = 1, CKP20 = 1, SSE20 = 0)



Note The value of the last bit previously output is output.

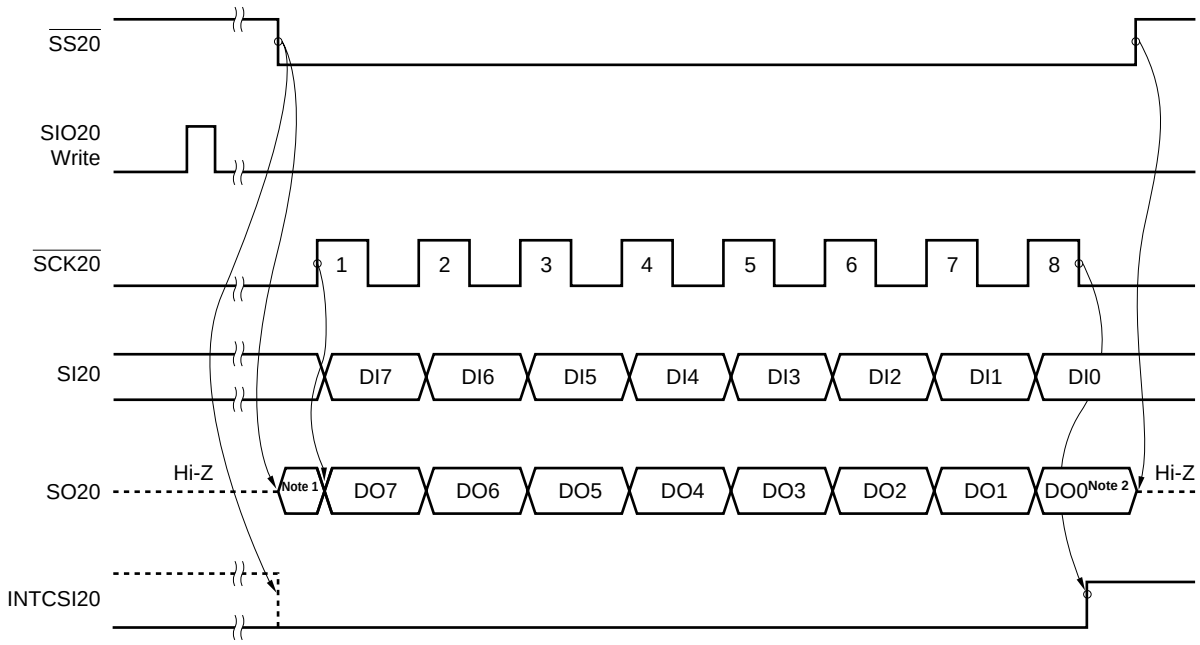
(xi) Slave operation (when DAP20 = 1, CKP20 = 1, SSE20 = 0)



Note The value of the last bit previously output is output.

Figure 9-11. 3-Wire Serial I/O Mode Timing (7/7)

(xii) Slave operation (when DAP20 = 1, CKP20 = 1, SSE20 = 1)



Notes 1. The value of the last bit previously output is output.

2. DO0 is output until $\overline{SS20}$ rises.
When $\overline{SS20}$ is high, SO20 is in a high-impedance state.

(3) Transfer start

Serial transfer is started by setting transfer data to the transmission shift register (TXS20/SIO20) when the following two conditions are satisfied.

- Serial operation mode register 20 (CSIM20) bit 7 (CSIE20) = 1
- Internal serial clock is stopped or $\overline{SCK20}$ is high after 8-bit serial transfer.

Caution If CSIE20 is set to "1" after data is written to TXS20/SIO20, transfer does not start.

The termination of 8-bit transfer stops the serial transfer automatically and generates the interrupt request signal (INTCSI20).

CHAPTER 10 INTERRUPT FUNCTIONS

10.1 Interrupt Function Types

The following two types of interrupt functions are used.

(1) Non-maskable interrupt

This interrupt is acknowledged unconditionally even if interrupts are disabled. It does not undergo interrupt priority control and is given top priority over all other interrupt requests.
A standby release signal is generated.
An interrupt from the watchdog timer is the only non-maskable interrupt source.

(2) Maskable interrupt

These interrupts undergo mask control. If two or more interrupts are simultaneously generated, each interrupt has a predetermined priority as shown in Table 10-1.
A standby release signal is generated.
There are three external sources and four internal sources of maskable interrupts.

10.2 Interrupt Sources and Configuration

There are a total of 8 non-maskable and maskable interrupt sources (see **Table 10-1**).

Table 10-1. Interrupt Sources

Interrupt Type	Priority[Note 1]	Interrupt Source		Internal/External	Vector Table Address	Basic Configuration Type[Note 2]
		Name	Trigger			
Non-maskable interrupt	–	INTWDT	Watchdog timer overflow (when watchdog timer mode 1 is selected)	Internal	0004H	(A)
Maskable interrupt	0	INTWDT	Watchdog timer overflow (when interval timer mode is selected)			(B)
	1	INTP0	Pin input edge detection	External	0006H	(C)
	2	INTP1			0008H	
	3	INTP2			000AH	
	4	INTSR20	End of UART reception on serial interface 20	Internal	000CH	(B)
		INTCSI20	End of 3-wire SIO transfer reception on serial interface 20			
	5	INTST20	End of UART transmission on serial interface 20		000EH	
	6	INTTM80	Generation of match signal for 8-bit timer/event counter 80		0014H	
	7	INTTM90	Generation of match signal for 16-bit timer 90		0016H	

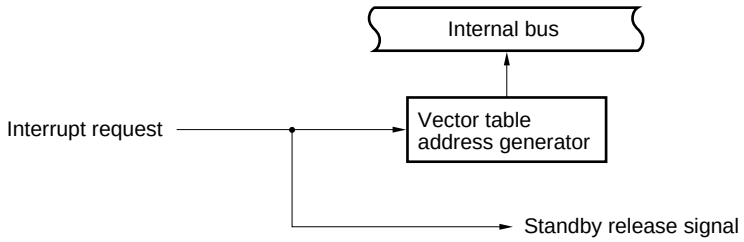
Notes 1. Priority is the priority order when several maskable interrupt requests are generated at the same time. 0 is the highest and 7 is the lowest.

2. Basic configuration types (A), (B), and (C) correspond to (A), (B), and (C) in Figure 10-1.

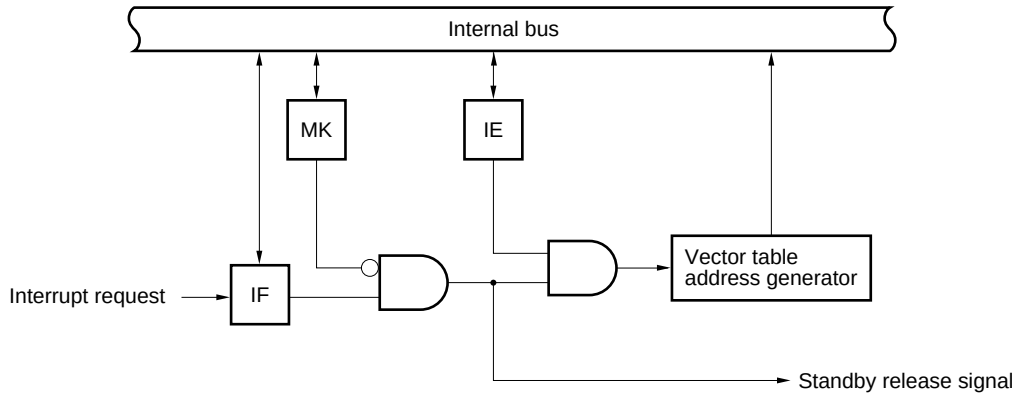
Remark There are two interrupt sources for the watchdog timer (INTWDT): non-maskable interrupts and maskable interrupts. Either one (but not both) should be selected for actual use.

Figure 10-1. Basic Configuration of Interrupt Function

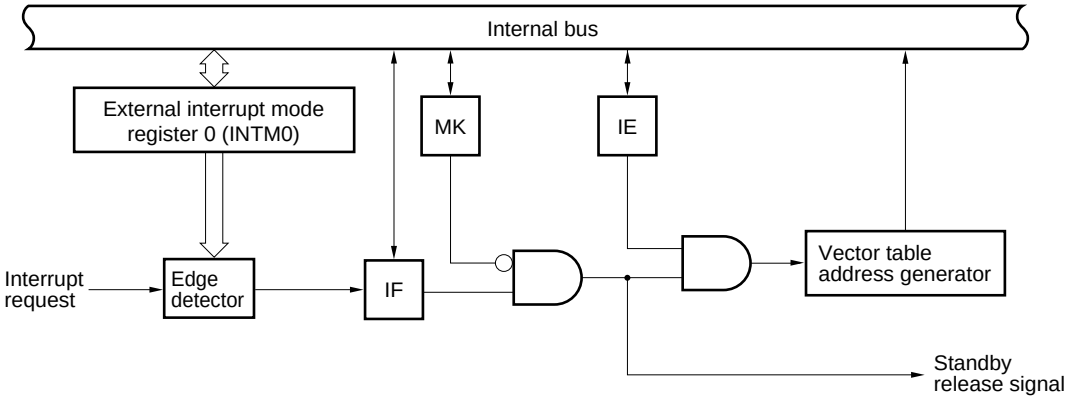
(A) Internal non-maskable interrupt



(B) Internal maskable interrupt



(C) External maskable interrupt



IF: Interrupt request flag
IE: Interrupt enable flag
MK: Interrupt mask flag

10.3 Interrupt Function Control Registers

The interrupt functions are controlled by the following four types of registers:

- Interrupt request flag registers 0 and 1 (IF0 and IF1)
- Interrupt mask flag registers 0 and 1 (MK0 and MK1)
- External interrupt mode register 0 (INTM0)
- Program status word (PSW)

Table 10-2 lists interrupt requests, the corresponding interrupt request flags, and interrupt mask flags.

Table 10-2. Interrupt Request Signals and Corresponding Flags

Interrupt Request Signal	Interrupt Request Flag	Interrupt Mask Flag
INTWDT	WDTIF	WDTMK
INTP0	PIF0	PMK0
INTP1	PIF1	PMK1
INTP2	PIF2	PMK2
INTSR20/INTCSI20	SRIF20	SRMK20
INTST20	STIF20	STMK20
INTTM80	TMIF80	TMMK80
INTTM90	TMIF90	TMMK90

(1) Interrupt request flag registers 0 and 1 (IF0 and IF1)

An interrupt request flag is set to 1 when the corresponding interrupt request is issued, or when the related instruction is executed. It is cleared to 0 when the interrupt request is acknowledged, when a $\overline{\text{RESET}}$ signal is input, or when a related instruction is executed.

IF0 and IF1 are set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears IF0 and IF1 to 00H.

Figure 10-2. Format of Interrupt Request Flag Register

Symbol	7	6	<5>	<4>	<3>	<2>	<1>	<0>	Address	After reset	R/W
IF0	0	0	STIF20	SRIF20	PIF2	PIF1	PIF0	WDTIF	FFE0H	00H	R/W

	7	6	5	4	3	2	<1>	<0>			
IF1	0	0	0	0	0	0	TMIF90	TMIF80	FFE1H	00H	R/W

××IF×	Interrupt request flag
0	No interrupt request signal has been issued.
1	An interrupt request signal has been issued; an interrupt request has been made.

Cautions 1. Bits 6 and 7 of IF0 and bits 2 to 7 of IF1 must all be set to 0.

2. The WDTIF flag can be read- and write-accessed only when the watchdog timer is being used as an interval timer. It must be cleared to 0 if the watchdog timer is used in watchdog timer mode 1 or 2.

3. When port 2 is being used as an output port, and its output level is changed, an interrupt request flag is set, because this port is also used as an external interrupt input. To use port 2 in output mode, therefore, the interrupt mask flag must be preset to 1.

(2) Interrupt mask flag registers 0 and 1 (MK0 and MK1)

The interrupt mask flags are used to enable and disable the corresponding maskable interrupts.

MK0 and MK1 are set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets MK0 and MK1 to FFH.

Figure 10-3. Format of Interrupt Mask Flag Register

Symbol	7	6	<5>	<4>	<3>	<2>	<1>	<0>	Address	After reset	R/W
MK0	1	1	STMK20	SRMK20	PMK2	PMK1	PMK0	WDTMK	FFE4H	FFH	R/W
	7	6	5	4	3	2	<1>	<0>			
MK1	1	1	1	1	1	1	TMMK90	TMMK80	FFE5H	FFH	R/W

××MK	Interrupt handling control
0	Enables interrupt handling.
1	Disables interrupt handling.

Cautions 1. Bits 6 and 7 of MK0 and bits 2 to 7 of MK1 must all be set to 1.

2. When the watchdog timer is being used in watchdog timer mode 1 or 2, any attempt to read the WDTMK flag results in an undefined value being detected.

3. When port 2 is being used as an output port, and its output level is changed, an interrupt request flag is set, because this port is also used as an external interrupt input. To use port 2 in output mode, therefore, the interrupt mask flag must be preset to 1.

(3) External interrupt mode register 0 (INTM0)

INTM0 is used to specify a valid edge for INTP0 to INTP2.

INTM0 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears INTM0 to 00H.

Figure 10-4. Format of External Interrupt Mode Register 0

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
INTM0	ES21	ES20	ES11	ES10	ES01	ES00	0	0	FFECH	00H	R/W

ES21	ES20	INTP2 valid edge selection
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES11	ES10	INTP1 valid edge selection
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES01	ES00	INTP0 valid edge selection
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

Cautions 1. Bits 0 and 1 must both be set to 0.

2. Before setting INTM0, set the corresponding interrupt mask flag to 1 to disable interrupts.

To enable interrupts, clear to 0 the corresponding interrupt request flag, then the corresponding interrupt mask flag.

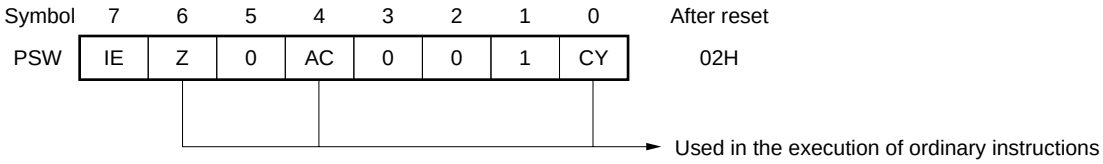
(4) Program status word (PSW)

The program status word is used to hold the instruction execution result and the current status of the interrupt requests. The IE flag, used to enable and disable maskable interrupts, is mapped to PSW.

PSW can be read- and write-accessed in 8-bit units, as well as using bit manipulation instructions and dedicated instructions (EI and DI). When a vector interrupt is acknowledged, the PSW is automatically saved to a stack, and the IE flag is reset to 0.

$\overline{\text{RESET}}$ input sets PSW to 02H.

Figure 10-5. Program Status Word Configuration



IE	Whether to enable/disable interrupt acknowledgement
0	Disabled
1	Enabled

10.4 Interrupt Processing Operation

10.4.1 Non-maskable interrupt request acknowledgement operation

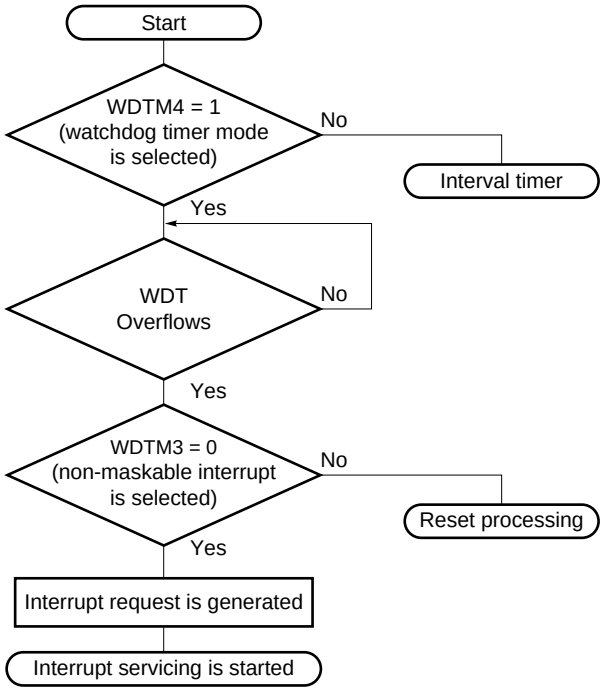
The non-maskable interrupt request is unconditionally acknowledged even when interrupts are disabled. It is not subject to interrupt priority control and takes precedence over all other interrupts.

When the non-maskable interrupt request is acknowledged, PSW and PC are saved to the stack in that order, the IE flag is reset to 0, the contents of the vector table are loaded to the PC, and then program execution branches.

Figure 10-6 shows the flowchart from non-maskable interrupt request generation to acknowledgement. Figure 10-7 shows the timing of non-maskable interrupt request acknowledgement. Figure 10-8 shows the acknowledgement operation if multiple non-maskable interrupts are generated.

Caution During a non-maskable interrupt service program execution, do not input another non-maskable interrupt request; if it is input, the service program will be interrupted and the new interrupt request will be acknowledged.

Figure 10-6. Flowchart from Non-Maskable Interrupt Request Generation to Acknowledgement



WDTM: Watchdog timer mode register

WDT: Watchdog timer

Figure 10-7. Timing of Non-Maskable Interrupt Request Acknowledgement

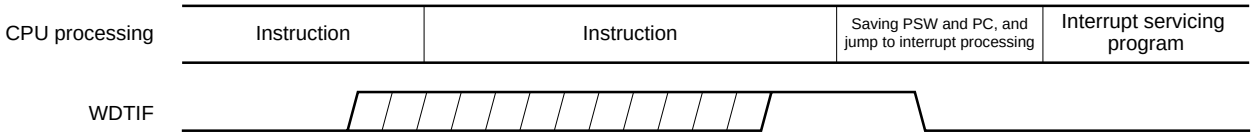
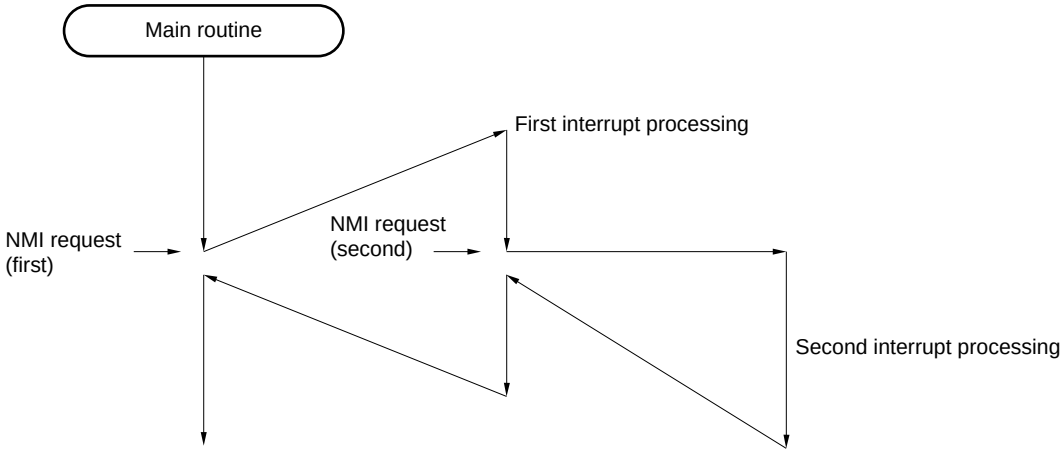


Figure 10-8. Acknowledgement of Non-Maskable Interrupt Request



10.4.2 Maskable interrupt request acknowledgement operation

A maskable interrupt request can be acknowledged when the interrupt request flag is set to 1 and the corresponding interrupt mask flag is cleared to 0. A vectored interrupt request is acknowledged in the interrupt enabled status (when the IE flag is set to 1).

The time required to start the interrupt servicing after a maskable interrupt request has been generated is shown in Table 10-3.

See Figures 10-10 and 10-11 for the interrupt request acknowledgement timing.

Table 10-3. Time from Generation of Maskable Interrupt Request to Servicing

Minimum Time	Maximum Time[Note]
9 clocks	19 clocks

Note The wait time is maximum when an interrupt request is generated immediately before BT and BF instruction.

Remark 1 clock: $\frac{1}{f_{CPU}}$ (f_{CPU}: CPU clock)

When two or more maskable interrupt requests are generated at the same time, they are acknowledged starting from the interrupt request assigned the highest priority.

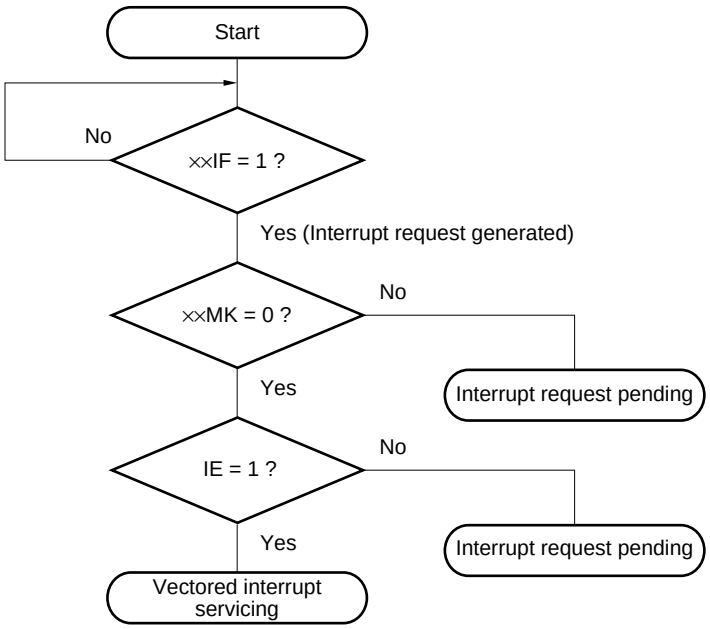
A pending interrupt is acknowledged when a status in which it can be acknowledged is set.

Figure 10-9 shows the algorithm of interrupt requests acknowledgement.

When a maskable interrupt request is acknowledged, the contents of the PSW and PC are saved to the stack in that order, the IE flag is reset to 0, and the data in the vector table determined for each interrupt request is loaded to the PC, and execution branches.

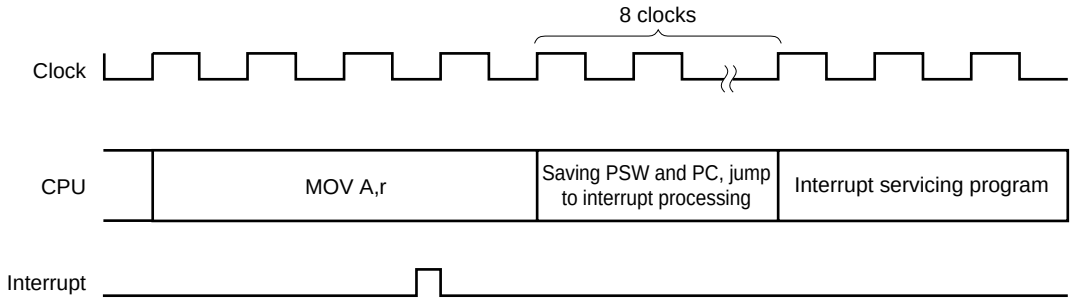
To return from interrupt servicing, use the RETI instruction.

Figure 10-9. Interrupt Request Acknowledgement Processing Algorithm



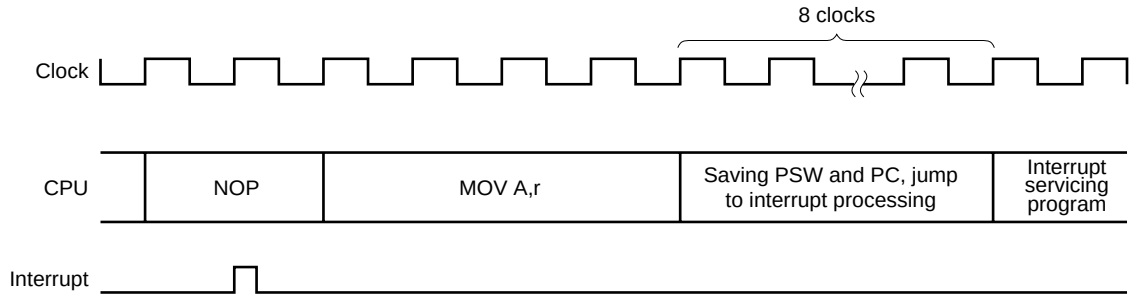
××IF: Interrupt request flag
××MK: Interrupt mask flag
IE: Flag to control maskable interrupt request acknowledgement (1 = enable, 0 = disable)

Figure 10-10. Interrupt Request Acknowledgement Timing (Example of MOV A,r)



If an interrupt request flag (××IF) is set before an instruction clock n (n = 4 to 10) under execution becomes n – 1, the interrupt is acknowledged after the instruction under execution is complete. Figure 10-10 shows an example of the interrupt request acknowledgement timing for an 8-bit data transfer instruction MOV A,r. Since this instruction is executed for 4 clocks, if an interrupt occurs for 3 clocks after the execution starts, the interrupt acknowledgement processing is performed after the MOV A,r instruction is executed.

Figure 10-11. Interrupt Request Acknowledgement Timing (When Interrupt Request Flag Is Set at the Last Clock During Instruction Execution)



If an interrupt request flag (××IF) is set at the last clock of the instruction, the interrupt acknowledgement processing starts after the next instruction is executed. Figure 10-11 shows an example of the interrupt acknowledgement timing for an interrupt request flag that is set at the second clock of NOP (2-clock instruction). In this case, the MOV A,r instruction after the NOP instruction is executed, and then the interrupt acknowledgement processing is performed.

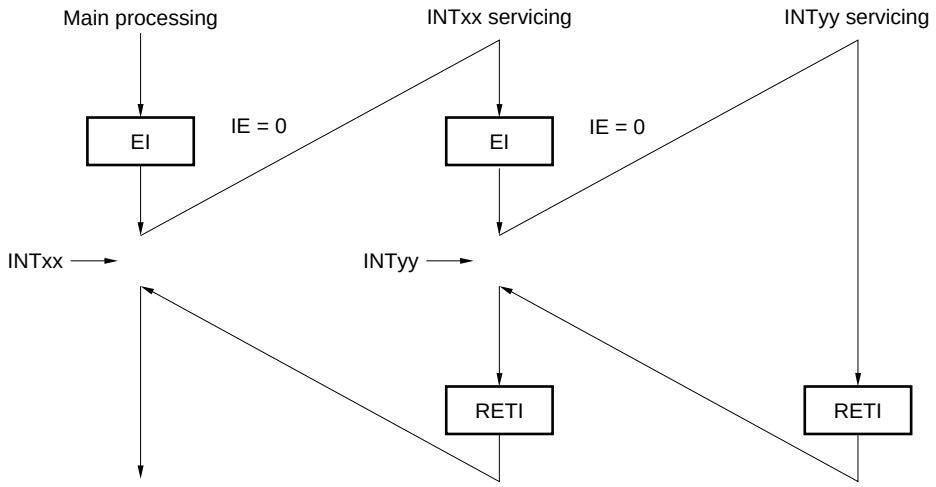
Caution Interrupt requests are reserved while interrupt request flag register 0 or 1 (IF0 or IF1) or interrupt mask flag register 0 or 1 (MK0 or MK1) is being accessed.

10.4.3 Multiple interrupt servicing

Multiple interrupt servicing in which another interrupt is acknowledged while an interrupt is being processed can be performed using a priority order system. When two or more interrupts are generated at once, interrupt servicing is performed according to the priority assigned to each interrupt request in advance (see **Table 10-1**).

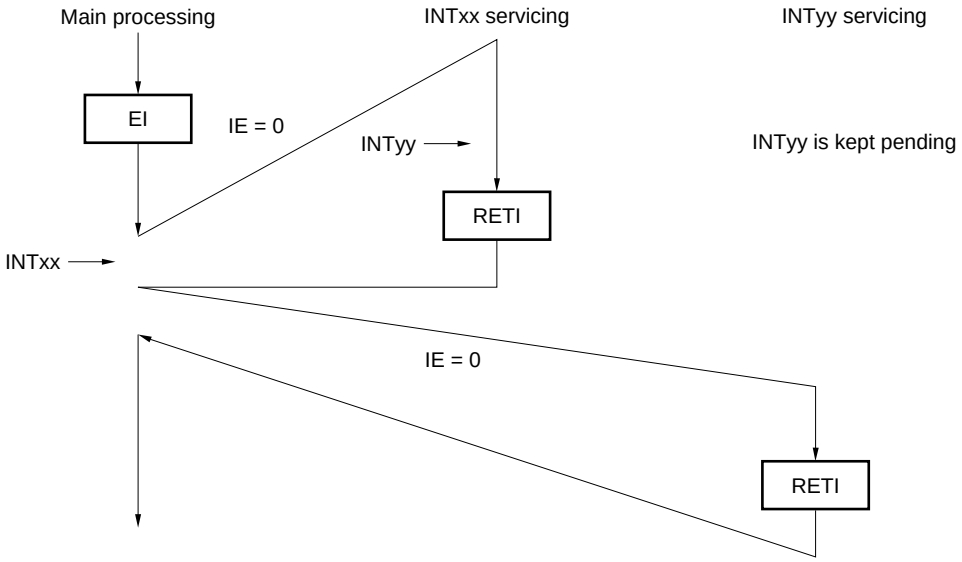
Figure 10-12. Example of Multiple Interrupts

Example 1. A multiple interrupt is acknowledged



During interrupt INTxx servicing, interrupt request INTyy is acknowledged, and multiple interrupts are generated. The EI instruction is issued before each interrupt request acknowledgement, and the interrupt request acknowledgement enable state is set.

Example 2. Multiple interrupts are not generated because interrupts are not enabled



Because interrupts are not enabled in interrupt INTxx servicing (the EI instruction is not issued), interrupt request INTyy is not acknowledged, and multiple interrupts are not generated. The INTyy request is reserved and acknowledged after the INTxx servicing is performed.

IE = 0: Interrupt request acknowledgement disabled

10.4.4 Interrupt request reserve

Some instructions may reserve the acknowledgement of an instruction request until the completion of the execution of the next instruction even if the interrupt request (maskable interrupt, non-maskable interrupt, and external interrupt) is generated during the execution. The following shows such instructions (interrupt request reserve instruction).

- Manipulation instruction for interrupt request flag registers 0 and 1 (IF0 and IF1)
- Manipulation instruction for interrupt mask flag registers 0 and 1 (MK0 and MK1)

[MEMO]

CHAPTER 11 STANDBY FUNCTION

11.1 Standby Function and Configuration

11.1.1 Standby function

The standby function is used to reduce the power consumption of the system and can be effected in the following two modes:

(1) HALT mode

This mode is set when the HALT instruction is executed. HALT mode stops the operation clock of the CPU. The system clock oscillator continues oscillating. This mode does not reduce the current drain as much as STOP mode, but is useful for resuming processing immediately when an interrupt request is generated, or for intermittent operations.

(2) STOP mode

This mode is set when the STOP instruction is executed. The STOP mode stops the main system clock oscillator and stops the entire system. The current consumption of the CPU can be substantially reduced in this mode.

The low voltage (V_{DD} = 1.8 V max.) of the data memory can be retained. Therefore, this mode is useful for retaining the contents of the data memory at an extremely low current consumption.

STOP mode can be released by an interrupt request, so that this mode can be used for intermittent operation. However, some time is required until the system clock oscillator stabilizes after STOP mode has been released. If processing must be resumed immediately by using an interrupt request, therefore, use the HALT mode.

In both modes, the previous contents of the registers, flags, and data memory before setting standby mode are all retained. In addition, the statuses of the output latches of the I/O ports and output buffers are also retained.

Caution To set STOP mode, be sure to stop the operations of the peripheral hardware, and then execute the STOP instruction.

11.1.2 Standby function control register

The wait time after STOP mode is released upon interrupt request until the oscillation stabilizes is controlled with the oscillation stabilization time selection register (OSTS).

OSTS is set with an 8-bit memory manipulation instruction.

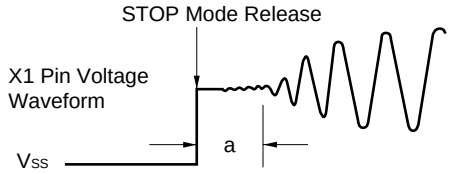
$\overline{\text{RESET}}$ input sets OSTS to 04H. However, the oscillation stabilization time after $\overline{\text{RESET}}$ input is $2^{15}/f_X$, instead of $2^{17}/f_X$.

Figure 11-1. Format of Oscillation Stabilization Time Selection Register

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0	FFFAH	04H	R/W

OSTS2	OSTS1	OSTS0	Oscillation stabilization time selection
0	0	0	$2^{12}/f_X$ (819 μs)
0	1	0	$2^{15}/f_X$ (6.55 ms)
1	0	0	$2^{17}/f_X$ (26.2 ms)
Other than above			Setting prohibited

Caution The wait time after STOP mode is released does not include the time from STOP mode release to clock oscillation start ("a" in the figure below), regardless of release by $\overline{\text{RESET}}$ input or by interrupt generation.



Remarks 1. f_X: System clock oscillation frequency

2. The parenthesized values apply to operation at f_X = 5.0 MHz.

11.2 Operation of Standby Function

11.2.1 HALT mode

(1) HALT mode

HALT mode is set by executing the HALT instruction.

The operation statuses in HALT mode are shown in the following table.

Table 11-1. Operation Statuses in HALT Mode

Item	HALT Mode Operation Status
System clock generator	System clock oscillation enabled Clock supply to CPU stopped
CPU	Operation disabled
Port (output latch)	Remains in the state existing before the selection of HALT mode
16-bit timer 90	Operation enabled
8-bit timer/event counter 80	Operation enabled
Watchdog timer	Operation enabled
Serial interface 20	Operation enabled
External interrupt	Operation enabled[Note]

Note Maskable interrupt that is not masked

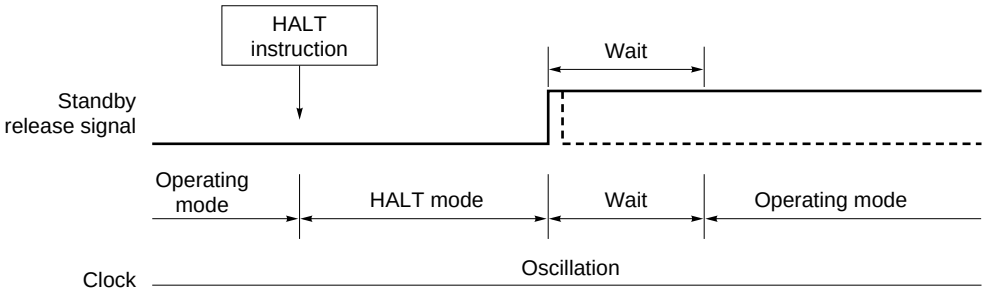
(2) Releasing HALT mode

HALT mode can be released by the following three sources:

(a) Releasing by unmasked interrupt request

HALT mode is released by an unmasked interrupt request. In this case, if interrupt request acknowledgement is enabled, vectored interrupt processing is performed. If interrupt acknowledgement is disabled, the instruction at the next address is executed.

Figure 11-2. Releasing HALT Mode by Interrupt



Remarks 1. The broken lines indicate the case where the interrupt request that has released standby mode is acknowledged.

2. The wait time is as follows:

- When vectored interrupt processing is performed: 9 to 10 clocks
- When vectored interrupt processing is not performed: 1 to 2 clocks

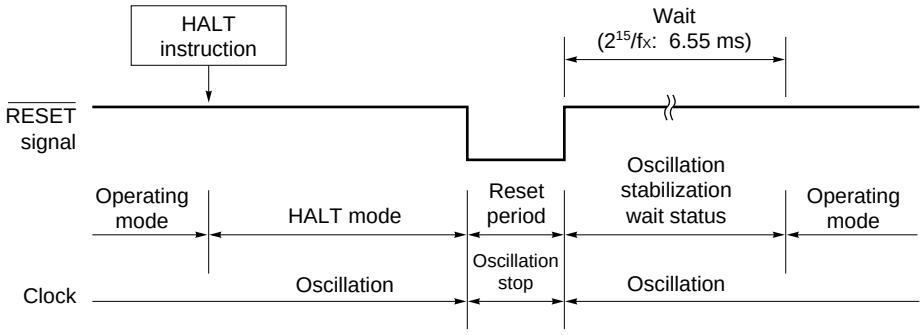
(b) Releasing by non-maskable interrupt request

HALT mode is released regardless of whether interrupts are enabled or disabled, and vectored interrupt processing is performed.

(c) Releasing by $\overline{\text{RESET}}$ input

When HALT mode is released by the $\overline{\text{RESET}}$ signal, execution branches to the reset vector address in the same manner as the ordinary reset operation, and program execution starts.

Figure 11-3. Releasing HALT Mode by $\overline{\text{RESET}}$ Input



Remarks 1. fx: System clock oscillation frequency

2. The parenthesized values apply to operation at fx = 5.0 MHz.

Table 11-2. Operation After Releasing HALT Mode

Releasing Source	MK××	IE	Operation
Maskable interrupt request	0	0	Executes next address instruction.
	0	1	Executes interrupt servicing.
	1	×	Retains HALT mode.
Non-maskable interrupt request	–	×	Executes interrupt servicing.
$\overline{\text{RESET}}$ input	–	–	Reset processing

×: don't care

11.2.2 STOP mode

(1) Setting and operation status of STOP mode

STOP mode is set by executing the STOP instruction.

Caution Because standby mode can be released by an interrupt request signal, standby mode is released as soon as it is set if there is an interrupt source whose interrupt request flag is set and interrupt mask flag is reset. When STOP mode is set, therefore, HALT mode is set immediately after the STOP instruction has been executed, the wait time set by the oscillation stabilization time selection register (OSTS) elapses, and then the operation mode is set.

The operation statuses in STOP mode are shown in the following table.

Table 11-3. Operation Statuses in STOP Mode

Item	STOP Mode Operation Status
Clock generator	System clock oscillation stopped
CPU	Operation stopped
Port (output latch)	Remains in the state existing before STOP mode has been set
16-bit timer 90	Operation stopped
8-bit timer/event counter 80	Operation enabled only when TI80 is selected for count clock
Watchdog timer	Operation stopped
Serial interface 20	Operation enabled only when external clock is input to serial clock
External interrupt	Operation enabled[Note]

Note Maskable interrupt that is not masked

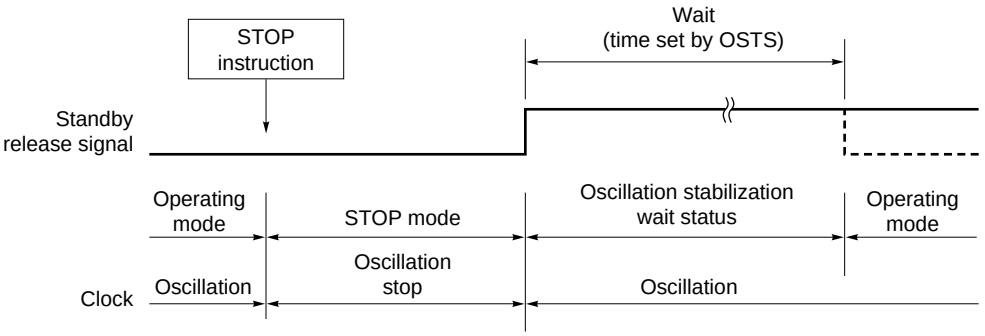
(2) Releasing STOP mode

STOP mode can be released by the following two sources:

(a) Releasing by unmasked interrupt request

STOP mode can be released by an unmasked interrupt request. In this case, vectored interrupt processing is performed if interrupt acknowledgement is enabled after the oscillation stabilization time has elapsed. If interrupt acknowledgement is disabled, the instruction at the next address is executed.

Figure 11-4. Releasing STOP Mode by Interrupt

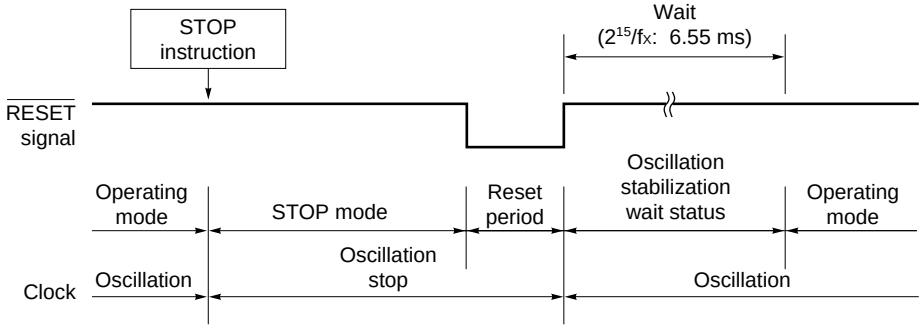


Remark The broken lines indicate the case where the interrupt request that has released standby mode is acknowledged.

(b) Releasing by $\overline{\text{RESET}}$ input

When STOP mode is released by the $\overline{\text{RESET}}$ signal, the reset operation is performed after the oscillation stabilization time has elapsed.

Figure 11-5. Releasing STOP Mode by $\overline{\text{RESET}}$ Input



Remarks 1. fx: System clock oscillation frequency

2. The parenthesized values apply to operation at fx = 5.0 MHz.

Table 11-4. Operation After Releasing STOP Mode

Releasing Source	MK××	IE	Operation
Maskable interrupt request	0	0	Executes next address instruction.
	0	1	Executes interrupt servicing.
	1	×	Retains STOP mode.
$\overline{\text{RESET}}$ input	–	–	Reset processing

×: don't care

CHAPTER 12 RESET FUNCTION

The following two operations are available to generate reset signals.

(1) External reset input by $\overline{\text{RESET}}$ signal input
(2) Internal reset by watchdog timer runaway time detection

External and internal reset have no functional differences. In both cases, program execution starts at the address at 0000H and 0001H by reset signal input.

When a low level is input to the $\overline{\text{RESET}}$ pin or the watchdog timer overflows, a reset is applied and each hardware is set to the status shown in Table 12-1. Each pin is high impedance during reset input or during the oscillation stabilization time just after reset clear.

When a high level is input to the $\overline{\text{RESET}}$ pin, the reset is cleared and program execution is started after the oscillation stabilization time has elapsed. The reset applied by the watchdog timer overflow is automatically cleared after reset, and program execution is started after the oscillation stabilization time has elapsed (see **Figures 12-2** through **12-4**).

Cautions 1. For an external reset, input a low level of 10 μs or more to the $\overline{\text{RESET}}$ pin.

2. When STOP mode is cleared by reset, the STOP mode contents are held during reset input. However, the port pins become high impedance.

Figure 12-1. Block Diagram of Reset Function

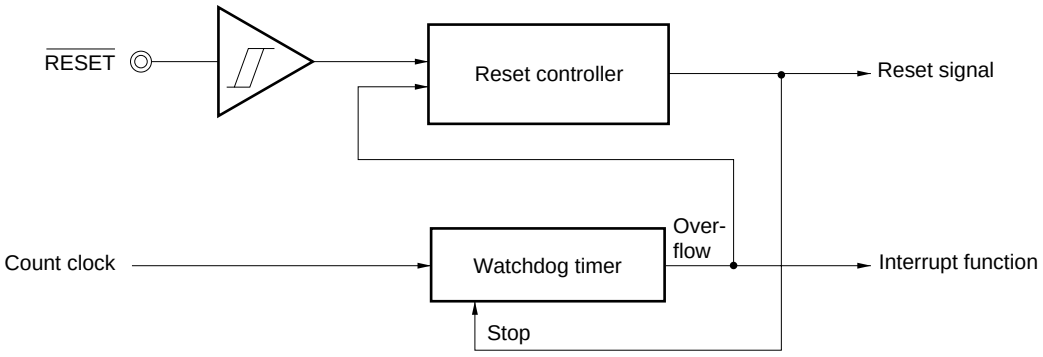


Figure 12-2. Reset Timing by $\overline{\text{RESET}}$ Input

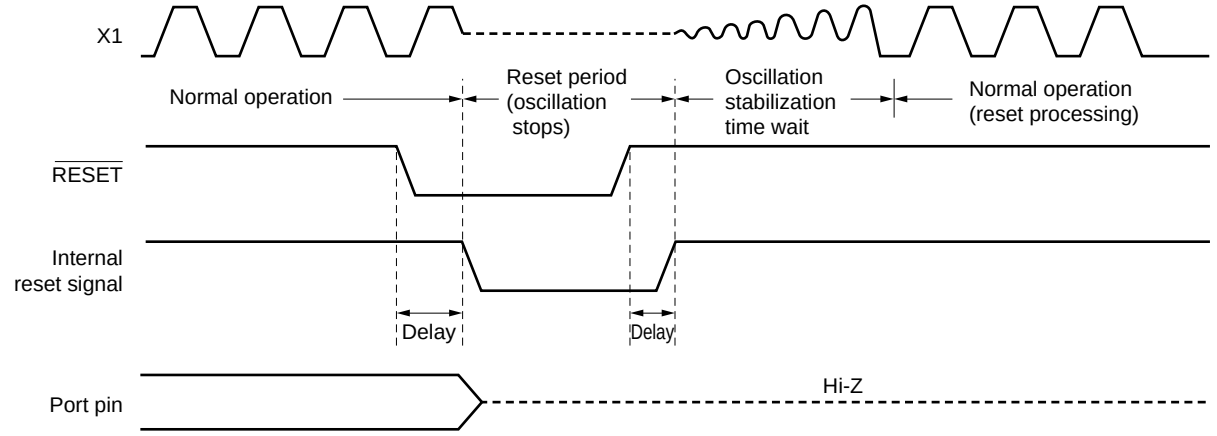


Figure 12-3. Reset Timing by Watchdog Timer Overflow

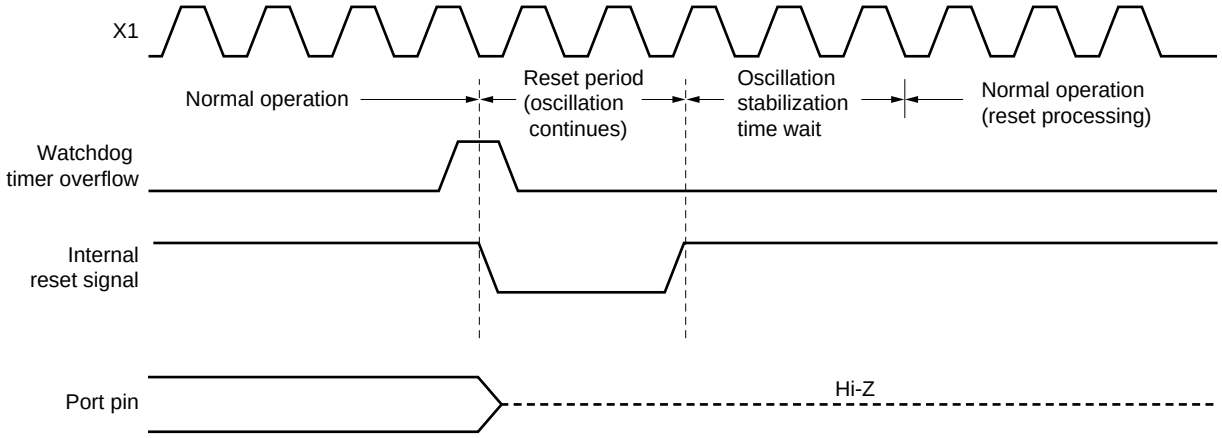


Figure 12-4. Reset Timing by $\overline{\text{RESET}}$ Input in STOP Mode

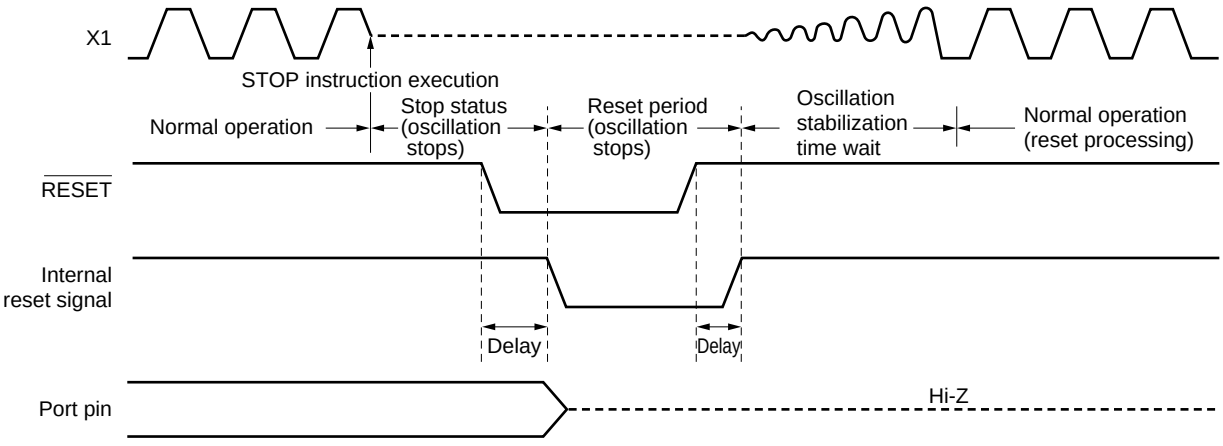


Table 12-1. Status of Hardware After Reset

Hardware		Status After Reset
Program counter (PC)[Note 1]		Loaded with the contents of the reset vector table (0000H, 0001H)
Stack pointer (SP)		Undefined
Program status word (PSW)		02H
RAM	Data memory	Undefined[Note 2]
	General-purpose registers	Undefined[Note 2]
Ports (P0 to P3) (output latch)		00H
Port mode registers (PM0 to PM3)		FFH
Pull-up resistor option registers (PU0, PUB2)		00H
Oscillation stabilization time selection register (OSTS)		04H
16-bit timer	Timer counter (TM90)	0000H
	Compare register (CR90)	FFFFH
	Control register (TMC90)	00H
	Capture register (TCP90)	Undefined
8-bit timer/event counter	Timer counter (TM80)	00H
	Compare register (CR80)	Undefined
	Mode control register (TMC80)	00H
Watchdog timer	Clock selection register (WDCS)	00H
	Mode register (WDTM)	00H
Serial interface	Serial operation mode register (CSIM20)	00H
	Asynchronous serial interface mode register (ASIM20)	00H
	Asynchronous serial interface status register (ASIS20)	00H
	Baud rate generator control register (BRGC20)	00H
	Transmission shift register (TXS20)	FFH
	Receive buffer register (RXB20)	Undefined
Interrupts	Request flag registers (IF0, IF1)	00H
	Mask flag registers (MK0, MK1)	FFH
	External interrupt mode register (INTM0)	00H

Notes 1. While a reset signal is being input, and during the oscillation stabilization period, the contents of the PC will be undefined, while the remainder of the hardware will be the same as after the reset.

2. In standby mode, the RAM enters the hold state after a reset.

[MEMO]

CHAPTER 13 μPD78F9076

The μPD78F9076 replaces the internal ROM of the μPD789071, 789072, and 789074 with flash memory. The differences between the flash memory and the mask ROM versions are shown in Table 13-1.

Table 13-1. Differences Between Flash Memory and Mask ROM Versions

Item		Flash Memory Version	Mask ROM Version		
		μPD78F9076	μPD789071	μPD789072	μPD789074
Internal memory	ROM structure	Flash memory	Mask ROM		
	ROM capacity	16 KB	2 KB	4 KB	8 KB
	High-speed RAM	256 bytes			
V_{PP} pin		Provided	Not provided		
Electrical characteristics		Varies depending on flash memory or mask ROM version.			

Caution The flash memory and mask ROM versions have different noise immunity and noise radiation characteristics. Do not use ES versions for evaluation when considering switching from flash memory versions to those using mask ROM upon the transition from preproduction to mass-production. CS versions (mask ROM versions) should be used in this case.

13.1 Flash Memory Programming

The on-chip program memory in the μPD78F9076 is flash memory.

The flash memory can be written with the μPD78F9076 mounted on the target system (on-board write). Connect the dedicated flash writer (Flashpro III (part number: FL-PR3, PG-FP3)) to the host machine and target system to write the flash memory.

Remark FL-PR3 is a product of Naito Densei Machida Mfg. Co., Ltd.

13.1.1 Selecting communication mode

The flash memory is written by using Flashpro III and by means of serial communication. Select a communication mode from those listed in Table 13-2. To select a communication mode, the format shown in Figure 13-1 is used. Each communication mode is selected by the number of V_{PP} pulses shown in Table 13-2.

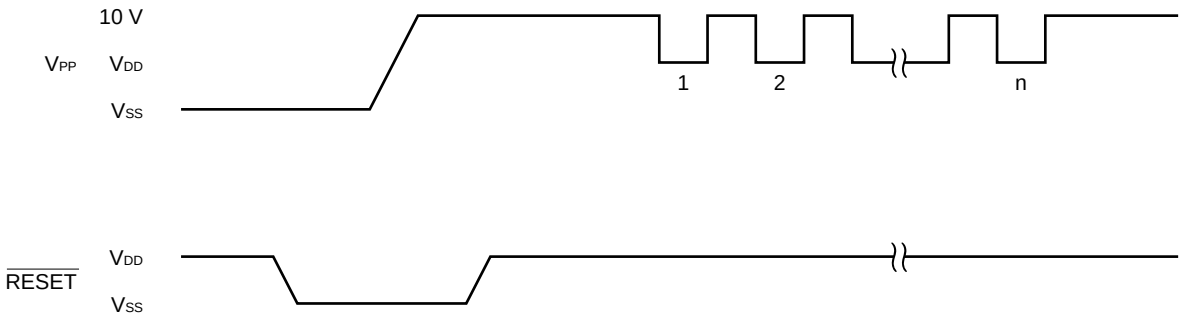
Table 13-2. Communication Mode

Communication Mode	Pins Used	Number of V_{PP} Pulses
3-wire serial I/O	$\overline{SCK20}$/ASCK20/P20 SO20/TxD20/P21 SI20/RxD20/P22	0
UART	TxD20/SO20/P21 RxD20/SI20/P22	8
Pseudo 3-wire mode[Note]	P00 (Serial clock input) P01 (Serial data output) P02 (Serial data input)	12

Note Serial transfer is performed by controlling a port by software.

Caution Be sure to select a communication mode depending on the number of V_{PP} pulses shown in Table 13-2.

Figure 13-1. Format of Communication Mode Selection



13.1.2 Function of flash memory programming

By transmitting/receiving commands and data in the selected communication mode, operations such as writing to the flash memory are performed. Table 13-3 shows the major functions of flash memory programming.

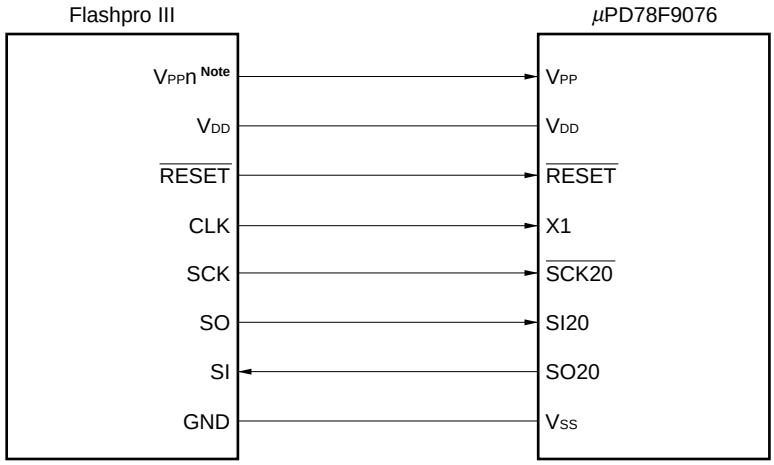
Table 13-3. Major Functions of Flash Memory Programming

Function	Description
Batch erase	Erases all contents of memory.
Batch blank check	Checks erased state of entire memory.
Data write	Writes to flash memory based on write start address and number of data written (number of bytes).
Batch verify	Compares all contents of memory with input data.

13.1.3 Flashpro III connection example

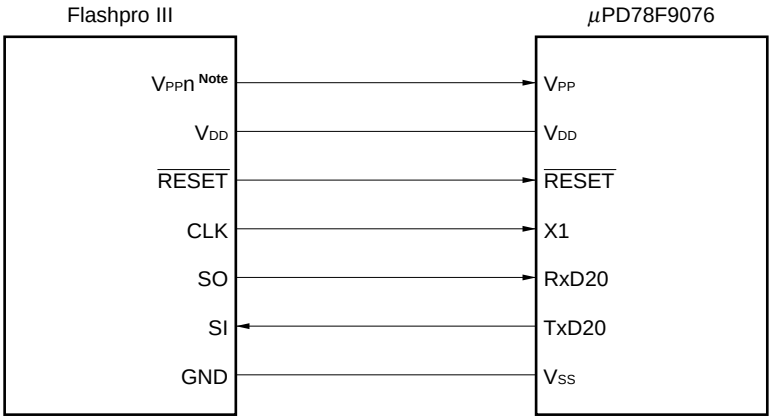
The connection between the Flashpro III and the μPD78F9076 differs depending on the communication mode (3-wire serial I/O, UART, or pseudo 3-wire mode). Figures 13-2 to 13-4 show the connection in the respective modes.

Figure 13-2. Flashpro III Connection Example in 3-Wire Serial I/O Mode



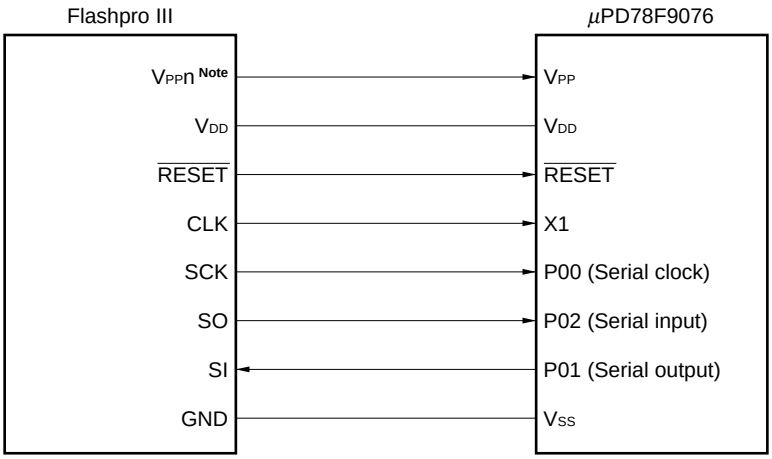
Note n = 1, 2

Figure 13-3. Flashpro III Connection Example in UART Mode



Note n = 1, 2

Figure 13-4. Flashpro III Connection Example in Pseudo 3-Wire Mode (When P0 Is Used)



Note n = 1, 2

13.1.4 Setting Example with Flashpro III (PG-FP3)

When writing data to the flash memory by using the Flashpro III (PG-FP3), set as follows.

<1> Load the parameter file.

<2> Select a serial mode and serial clock by using the type command.

<3> An example of setting PG-FP3 is shown below.

Table 13-4. Setting Example with PG-FP3

Communication Mode	Setting Example with PG-FP3		Number of V_{PP} Pulses[Note 1]
3-wire serial I/O	COMM PORT	SIO-ch0	0
	CPU CLK	On Target Board	
		In Flashpro	
	On Target Board	4.1943 MHz	
	SIO CLK	1.0 MHz	
	In Flashpro	4.0 MHz	
	SIO CLK	1.0 MHz	
UART	COMM PORT	UART-ch0	8
	CPU CLK	On Target Board	
	On Target Board	4.1943 MHz	
	UART BPS	9,600 bps[Note 2]	
Pseudo 3-wire mode	COMM PORT	Port A	12
	CPU CLK	On Target Board	
		In Flashpro	
	On Target Board	4.1943 MHz	
	SIO CLK	1 kHz	
	In Flashpro	4.0 MHz	
	SIO CLK	1 kHz	

Notes 1. The number of V_{PP} pulses supplied from the Flashpro III when serial communication is initialized. These pulse counts determine the pins used for communication.

2. Select 9,600 bps, 19,200 bps, 38,400 bps, or 76,800 bps.

Remark COMM PORT: Selects serial port.
SIO CLK: Selects serial clock frequency.
CPU CLK: Selects source of CPU clock to be input.

[MEMO]

CHAPTER 14 INSTRUCTION SET OVERVIEW

This chapter lists the instruction set of the μPD789074 Subseries. For details of the operation and machine language (instruction code) of each instruction, refer to **78K/0S Series User's Manual Instructions (U11047E)**.

14.1 Operation

14.1.1 Operand identifiers and description methods

Operands are described in "Operand" column of each instruction in accordance with the description method of the instruction operand identifier (refer to the assembler specifications for details). When there are two or more description methods, select one of them. Uppercase letters and the symbols #, !, $, and [ ] are key words and are described as they are. Each symbol has the following meaning.

- #: Immediate data specification
- !: Absolute address specification
- $: Relative address specification
- [ ]: Indirect address specification

In the case of immediate data, describe an appropriate numeric value or a label. When using a label, be sure to describe the #, !, $ and [ ] symbols.

For operand register identifiers, r and rp, either function names (X, A, C, etc.) or absolute names (names in parentheses in the table below, R0, R1, R2, etc.) can be used for description.

Table 14-1. Operand Identifiers and Description Methods

Identifier	Description Method
r rp sfr	X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7) AX (RP0), BC (RP1), DE (RP2), HL (RP3) Special-function register symbol
saddr saddrp	FE20H to FF1FH Immediate data or labels FE20H to FF1FH Immediate data or labels (even addresses only)
addr16 addr5	0000H to FFFFH Immediate data or labels (only even addresses for 16-bit data transfer instructions) 0040H to 007FH Immediate data or labels (even addresses only)
word byte bit	16-bit immediate data or label 8-bit immediate data or label 3-bit immediate data or label

Remark For symbols of special function registers, see **Table 3-3 Special Function Registers**.

14.1.2 Description of "Operation" column

A:	A register; 8-bit accumulator
X:	X register
B:	B register
C:	C register
D:	D register
E:	E register
H:	H register
L:	L register
AX:	AX register pair; 16-bit accumulator
BC:	BC register pair
DE:	DE register pair
HL:	HL register pair
PC:	Program counter
SP:	Stack pointer
PSW:	Program status word
CY:	Carry flag
AC:	Auxiliary carry flag
Z:	Zero flag
IE:	Interrupt request enable flag
NMIS:	Flag indicating non-maskable interrupt servicing in progress
( ):	Memory contents indicated by address or register contents in parentheses
$\times_H$, $\times_L$:	Higher 8 bits and lower 8 bits of 16-bit register
$\wedge$:	Logical product (AND)
$\vee$:	Logical sum (OR)
$\veebar$:	Exclusive logical sum (exclusive OR)
$\overline{\quad}$:	Inverted data
addr16:	16-bit immediate data or label
jdisp8:	Signed 8-bit data (displacement value)

14.1.3 Description of "Flag" column

(Blank):	Unchanged
0:	Cleared to 0
1:	Set to 1
$\times$:	Set/cleared according to the result
R:	Previously saved value is stored

14.2 Operation List

Mnemonic	Operand	Bytes	Clocks	Operation	Flag
					Z AC CY
MOV	r, #byte	3	6	r ← byte	
	saddr, #byte	3	6	(saddr) ← byte	
	sfr, #byte	3	6	sfr ← byte	
	A, r Note 1	2	4	A ← r	
	r, A Note 1	2	4	r ← A	
	A, saddr	2	4	A ← (saddr)	
	saddr, A	2	4	(saddr) ← A	
	A, sfr	2	4	A ← sfr	
	sfr, A	2	4	sfr ← A	
	A, !addr16	3	8	A ← (addr16)	
	!addr16, A	3	8	(addr16) ← A	
	PSW, #byte	3	6	PSW ← byte	× × ×
	A, PSW	2	4	A ← PSW	
	PSW, A	2	4	PSW ← A	× × ×
	A, [DE]	1	6	A ← (DE)	
	[DE], A	1	6	(DE) ← A	
	A, [HL]	1	6	A ← (HL)	
	[HL], A	1	6	(HL) ← A	
	A, [HL + byte]	2	6	A ← (HL + byte)	
	[HL + byte], A	2	6	(HL + byte) ← A	
XCH	A, X	1	4	A ↔ X	
	A, r Note 2	2	6	A ↔ r	
	A, saddr	2	6	A ↔ (saddr)	
	A, sfr	2	6	A ↔ sfr	
	A, [DE]	1	8	A ↔ (DE)	
	A, [HL]	1	8	A ↔ (HL)	
	A, [HL, byte]	2	8	A ↔ (HL + byte)	

Notes 1. Except r = A.

2. Except r = A, X.

Remark One instruction clock cycle is one CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Mnemonic	Operand	Bytes	Clocks	Operation	Flag		
					Z	AC	CY
MOVW	rp, #word	3	6	rp ← word			
	AX, saddrp	2	6	AX ← (saddrp)			
	saddrp, AX	2	8	(saddrp) ← AX			
	AX, rp Note	1	4	AX ← rp			
	rp, AX Note	1	4	rp ← AX			
XCHW	AX, rp Note	1	8	AX ↔ rp			
ADD	A, #byte	2	4	A, CY ← A + byte	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte	×	×	×
	A, r	2	4	A, CY ← A + r	×	×	×
	A, saddr	2	4	A, CY ← A + (saddr)	×	×	×
	A, !addr16	3	8	A, CY ← A + (addr16)	×	×	×
	A, [HL]	1	6	A, CY ← A + (HL)	×	×	×
	A, [HL + byte]	2	6	A, CY ← A + (HL + byte)	×	×	×
ADDC	A, #byte	2	4	A, CY ← A + byte + CY	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte + CY	×	×	×
	A, r	2	4	A, CY ← A + r + CY	×	×	×
	A, saddr	2	4	A, CY ← A + (saddr) + CY	×	×	×
	A, !addr16	3	8	A, CY ← A + (addr16) + CY	×	×	×
	A, [HL]	1	6	A, CY ← A + (HL) + CY	×	×	×
	A, [HL + byte]	2	6	A, CY ← A + (HL + byte) + CY	×	×	×
SUB	A, #byte	2	4	A, CY ← A – byte	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) – byte	×	×	×
	A, r	2	4	A, CY ← A – r	×	×	×
	A, saddr	2	4	A, CY ← A – (saddr)	×	×	×
	A, !addr16	3	8	A, CY ← A – (addr16)	×	×	×
	A, [HL]	1	6	A, CY ← A – (HL)	×	×	×
	A, [HL + byte]	2	6	A, CY ← A – (HL + byte)	×	×	×

Note Only when rp = BC, DE, or HL.

Remark One instruction clock cycle is one CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Mnemonic	Operand	Bytes	Clocks	Operation	Flag		
					Z	AC	CY
SUBC	A, #byte	2	4	A, CY ← A – byte – CY	×	×	×
	saddr, #byte	3	6	(saddr), CY ← (saddr) – byte – CY	×	×	×
	A, r	2	4	A, CY ← A – r – CY	×	×	×
	A, saddr	2	4	A, CY ← A – (saddr) – CY	×	×	×
	A, !addr16	3	8	A, CY ← A – (addr16) – CY	×	×	×
	A, [HL]	1	6	A, CY ← A – (HL) – CY	×	×	×
	A, [HL + byte]	2	6	A, CY ← A – (HL + byte) – CY	×	×	×
AND	A, #byte	2	4	A ← A ∧ byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr) ∧ byte	×		
	A, r	2	4	A ← A ∧ r	×		
	A, saddr	2	4	A ← A ∧ (saddr)	×		
	A, !addr16	3	8	A ← A ∧ (addr16)	×		
	A, [HL]	1	6	A ← A ∧ (HL)	×		
	A, [HL + byte]	2	6	A ← A ∧ (HL + byte)	×		
OR	A, #byte	2	4	A ← A ∨ byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr) ∨ byte	×		
	A, r	2	4	A ← A ∨ r	×		
	A, saddr	2	4	A ← A ∨ (saddr)	×		
	A, !addr16	3	8	A ← A ∨ (addr16)	×		
	A, [HL]	1	6	A ← A ∨ (HL)	×		
	A, [HL + byte]	2	6	A ← A ∨ (HL + byte)	×		
XOR	A, #byte	2	4	A ← A ⊻ byte	×		
	saddr, #byte	3	6	(saddr) ← (saddr) ⊻ byte	×		
	A, r	2	4	A ← A ⊻ r	×		
	A, saddr	2	4	A ← A ⊻ (saddr)	×		
	A, !addr16	3	8	A ← A ⊻ (addr16)	×		
	A, [HL]	1	6	A ← A ⊻ (HL)	×		
	A, [HL + byte]	2	6	A ← A ⊻ (HL + byte)	×		

Remark One instruction clock cycle is one CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Mnemonic	Operand	Bytes	Clocks	Operation	Flag		
					Z	AC	CY
CMP	A, #byte	2	4	A – byte	×	×	×
	saddr, #byte	3	6	(saddr) – byte	×	×	×
	A, r	2	4	A – r	×	×	×
	A, saddr	2	4	A – (saddr)	×	×	×
	A, !addr16	3	8	A – (addr16)	×	×	×
	A, [HL]	1	6	A – (HL)	×	×	×
	A, [HL + byte]	2	6	A – (HL + byte)	×	×	×
ADDW	AX, #word	3	6	AX, CY ← AX + word	×	×	×
SUBW	AX, #word	3	6	AX, CY ← AX – word	×	×	×
CMPW	AX, #word	3	6	AX – word	×	×	×
INC	r	2	4	r ← r + 1	×	×	
	saddr	2	4	(saddr) ← (saddr) + 1	×	×	
DEC	r	2	4	r ← r + 1	×	×	
	saddr	2	4	(saddr) ← (saddr) − 1	×	×	
INCW	rp	1	4	rp ← rp + 1			
DECW	rp	1	4	rp ← rp − 1			
ROR	A, 1	1	2	(CY, $A_7 \leftarrow A_0$, $A_{m-1} \leftarrow A_m$) × 1			×
ROL	A, 1	1	2	(CY, $A_0 \leftarrow A_7$, $A_{m+1} \leftarrow A_m$) × 1			×
RORC	A, 1	1	2	(CY ← A_0, $A_7 \leftarrow$ CY, $A_{m-1} \leftarrow A_m$) × 1			×
ROLC	A, 1	1	2	(CY ← A_7, $A_0 \leftarrow$ CY, $A_{m+1} \leftarrow A_m$) × 1			×
SET1	saddr.bit	3	6	(saddr.bit) ← 1			
	sfr.bit	3	6	sfr.bit ← 1			
	A.bit	2	4	A.bit ← 1			
	PSW.bit	3	6	PSW.bit ← 1	×	×	×
	[HL].bit	2	10	(HL).bit ← 1			
CLR1	saddr.bit	3	6	(saddr.bit) ← 0			
	sfr.bit	3	6	sfr.bit ← 0			
	A.bit	2	4	A.bit ← 0			
	PSW.bit	3	6	PSW.bit ← 0	×	×	×
	[HL].bit	2	10	(HL).bit ← 0			
SET1	CY	1	2	CY ← 1			1
CLR1	CY	1	2	CY ← 0			0
NOT1	CY	1	2	CY ← $\overline{CY}$			×

Remark One instruction clock cycle is one CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

Mnemonic	Operand	Bytes	Clocks	Operation	Flag		
					Z	AC	CY
CALL	!addr16	3	6	$(SP - 1) \leftarrow (PC + 3)_H$, $(SP - 2) \leftarrow (PC + 3)_L$, $PC \leftarrow$ addr16, $SP \leftarrow SP - 2$			
CALLT	[addr5]	1	8	$(SP - 1) \leftarrow (PC + 1)_H$, $(SP - 2) \leftarrow (PC + 1)_L$, $PC_H \leftarrow$ (00000000, addr5 + 1), $PC_L \leftarrow$ (00000000, addr5), $SP \leftarrow SP - 2$			
RET		1	6	$PC_H \leftarrow (SP + 1)$, $PC_L \leftarrow (SP)$, $SP \leftarrow SP + 2$			
RETI		1	8	$PC_H \leftarrow (SP + 1)$, $PC_L \leftarrow (SP)$, $PSW \leftarrow (SP + 2)$, $SP \leftarrow SP + 3$, NMIS $\leftarrow 0$	R	R	R
PUSH	PSW	1	2	$(SP - 1) \leftarrow PSW$, $SP \leftarrow SP - 1$			
	rp	1	4	$(SP - 1) \leftarrow rp_H$, $(SP - 2) \leftarrow rp_L$, $SP \leftarrow SP - 2$			
POP	PSW	1	4	$PSW \leftarrow (SP)$, $SP \leftarrow SP + 1$	R	R	R
	rp	1	6	$rp_H \leftarrow (SP + 1)$, $rp_L \leftarrow (SP)$, $SP \leftarrow SP + 2$			
MOVW	SP, AX	2	8	$SP \leftarrow AX$			
	AX, SP	2	6	$AX \leftarrow SP$			
BR	!addr16	3	6	$PC \leftarrow$ addr16			
	$addr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8			
	AX	1	6	$PC_H \leftarrow A$, $PC_L \leftarrow X$			
BC	$saddr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 if CY = 1			
BNC	$saddr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 if CY = 0			
BZ	$saddr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 if Z = 1			
BNZ	$saddr16	2	6	$PC \leftarrow PC + 2 +$ jdisp8 if Z = 0			
BT	saddr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if (saddr.bit) = 1			
	sfr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if sfr.bit = 1			
	A.bit, $addr16	3	8	$PC \leftarrow PC + 3 +$ jdisp8 if A.bit = 1			
	PSW.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if PSW.bit = 1			
BF	saddr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if (saddr.bit) = 0			
	sfr.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if sfr.bit = 0			
	A.bit, $addr16	3	8	$PC \leftarrow PC + 3 +$ jdisp8 if A.bit = 0			
	PSW.bit, $addr16	4	10	$PC \leftarrow PC + 4 +$ jdisp8 if PSW.bit = 0			
DBNZ	B, $addr16	2	6	$B \leftarrow B - 1$, then $PC \leftarrow PC + 2 +$ jdisp8 if $B \neq 0$			
	C, $addr16	2	6	$C \leftarrow C - 1$, then $PC \leftarrow PC + 2 +$ jdisp8 if $C \neq 0$			
	saddr, $addr16	3	8	$(saddr) \leftarrow (saddr) - 1$, then $PC \leftarrow PC + 3 +$ jdisp8 if $(saddr) \neq 0$			
NOP		1	2	No Operation			
EI		3	6	IE $\leftarrow$ 1 (Enable Interrupt)			
DI		3	6	IE $\leftarrow$ 0 (Disable Interrupt)			
HALT		1	2	Set HALT Mode			
STOP		1	2	Set STOP Mode			

Remark One instruction clock cycle is one CPU clock cycle (f_{CPU}) selected by the processor clock control register (PCC).

14.3 Instructions Listed by Addressing Type

(1) 8-bit instructions

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, INC, DEC, ROR, ROL, RORC, ROLC, PUSH, POP, DBNZ

1st Operand \ 2nd Operand	#byte	A	r	sfr	saddr	!addr16	PSW	[DE]	[HL]	[HL + byte]	$addr16	1	None
A	ADD ADDC SUB SUBC AND OR XOR CMP		MOV[Note] XCH[Note] ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP		ROR ROL RORC ROLC	
r	MOV	MOV											INC DEC
B, C											DBNZ		
sfr	MOV	MOV											
saddr	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV									DBNZ		INC DEC
!addr16		MOV											
PSW	MOV	MOV											PUSH POP
[DE]		MOV											
[HL]		MOV											
[HL + byte]		MOV											

Note Except r = A.

(2) 16-bit instructions

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

1st Operand \ 2nd Operand	#word	AX	rp[Note]	saddrp	SP	None
AX	ADDW SUBW CMPW		MOVW XCHW	MOVW	MOVW	
rp	MOVW	MOVW[Note]				INCW DECW PUSH POP
saddrp		MOVW				
sp		MOVW				

Note Only when rp = BC, DE, or HL.

(3) Bit manipulation instructions

SET1, CLR1, NOT1, BT, BF

1st Operand \ 2nd Operand	$addr16	None
A.bit	BT BF	SET1 CLR1
sfr.bit	BT BF	SET1 CLR1
saddr.bit	BT BF	SET1 CLR1
PSW.bit	BT BF	SET1 CLR1
[HL].bit		SET1 CLR1
CY		SET1 CLR1 NOT1

(4) Call instructions/branch instructions

CALL, CALLT, BR, BC, BNC, BZ, BNZ, DBNZ

1st Operand \ 2nd Operand	AX	!addr16	[addr5]	$addr16
Basic instructions	BR	CALL BR	CALLT	BR BC BNC BZ BNZ
Compound instructions				DBNZ

(5) Other instructions

RET, RETI, NOP, EI, DI, HALT, STOP

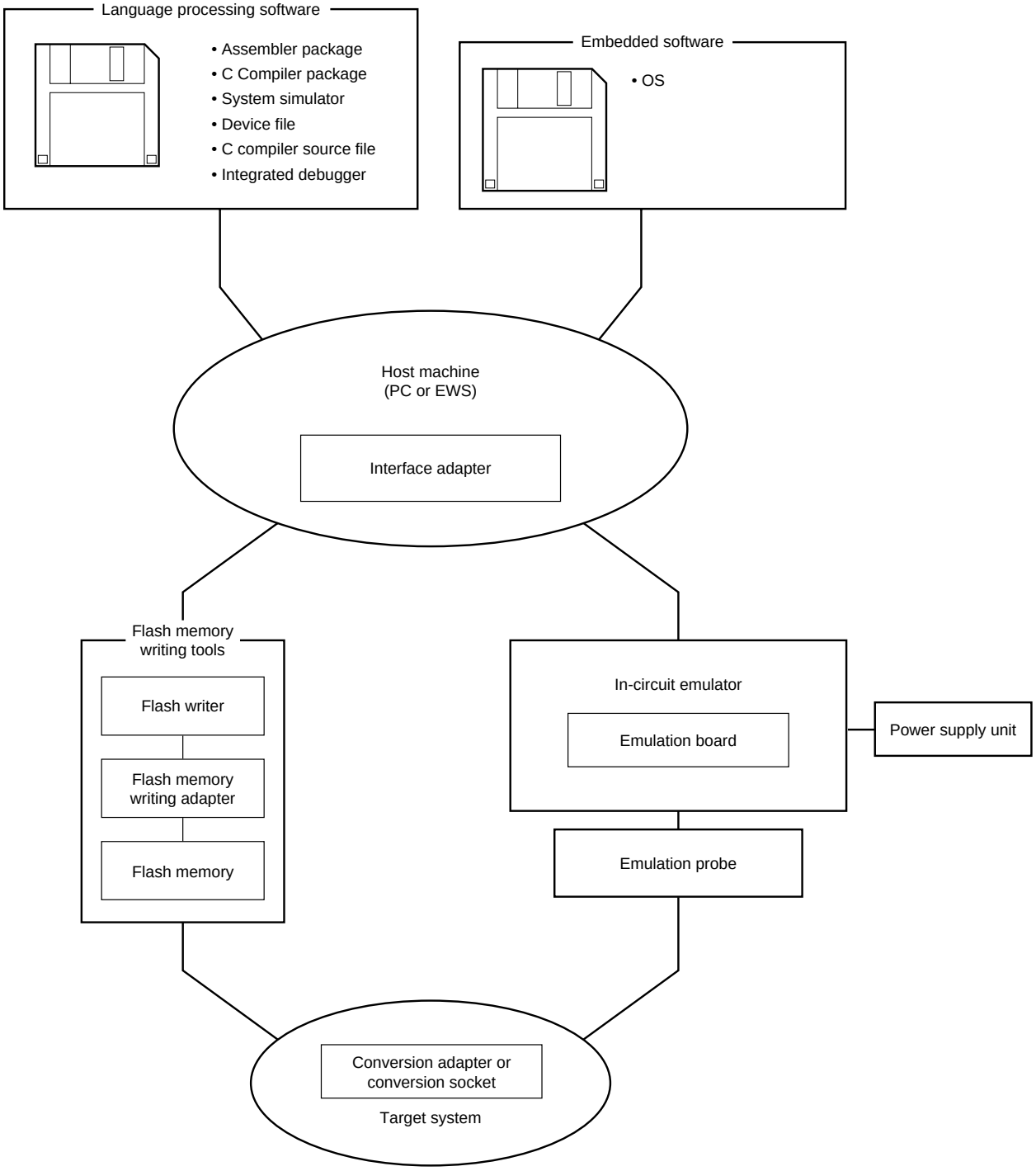
APPENDIX A DEVELOPMENT TOOLS

The following development tools are available for development of systems using the μPD789074 Subseries. Figure A-1 shows development tools.

- Compatibility with PC98-NX Series
  Unless stated otherwise, products which are supported for IBM PC/AT™ and compatibles can also be used with the PC98-NX Series. When using the PC98-NX Series, therefore, refer to the explanations for IBM PC/AT and compatibles.

- Windows
  Unless stated otherwise, "Windows" refers to the following operating systems.
  - Windows 3.1
  - Windows 95
  - Windows NT™ Ver. 4.0

Figure A-1. Development Tools



A.1 Language Processing Software

RA78K0S Assembler package	Program that converts program written in mnemonic into object code that can be executed by microcontroller. In addition, automatic functions to generate symbol table and optimize branch instructions are also provided. Used in combination with optional device file (DF789076). **<Caution when used under PC environment>** The assembler package is a DOS-based application but may be used under the Windows environment by using Project Manager of Windows (included in the package).
	Part number: μS××××RA78K0S
CC78K0S C compiler package	Program that converts program written in C language into object codes that can be executed by microcontroller. Used in combination with optional assembler package (RA78K0S) and device file (DF789076). **<Caution when used under PC environment>** The C compiler package is a DOS-based application but may be used under the Windows environment by using Project Manager of Windows (included in the assembler package).
	Part number: μS××××CC78K0S
DF789076[Note] Device file	File containing the information inherent to the device. Used in combination with other optional tools (RA78K0S, CC78K0S, SM78K0S).
	Part number: μS××××DF789076
CC78K0S-L C compiler source file	Source file of functions constituting object library included in C compiler package. Necessary for changing object library included in C compiler package according to customer's specifications. Since this is the source file, its working environment does not depend on any particular operating system.
	Part number: μS××××CC78K0S-L

Note DF789076 is a common file that can be used with RA78K0S, CC78K0S, and SM78K0S.

Remark ×××× in the part number differs depending on the host machines and operating systems to be used.

μS××××RA78K0S
μS××××CC78K0S
μS××××DF789076
μS××××CC78K0S-L

××××	Host Machine	OS	Supply Media
AA13	PC-9800 series	Japanese Windows[Note]	3.5" 2HD FD
AB13	IBM PC/AT and compatibles	Japanese Windows[Note]	3.5" 2HC FD
BB13		English Windows[Note]	
3P16	HP9000 series 700™	HP-UX™ (Rel.10.10)	DAT (DDS)
3K13	SPARCstation™	SunOS™ (Rel.4.1.4), Solaris™ (Rel.2.5.1)	3.5" 2HC FD
3K15			1/4" CGMT
3R13	NEWS™ (RISC)	NEWS-OS™ (Rel.6.1)	3.5" 2HC FD

Note Also operates under the DOS environment.

A.2 Flash Memory Writing Tools

Flashpro III (part number: FL-PR3, PG-FP3) Flash writer	Flash writer dedicated to the microcontrollers incorporating a flash memory.
FA-30MC Flash memory writing adapter	Flash memory writing adapter. Used in connection with Flashpro III. 30-pin plastic SSOP (MC-5A4 type)

Remark FL-PR3 and FA-30MC are products of Naito Densei Machida Mfg. Co., Ltd.
For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (+81-44-822-3813)

A.3 Debugging Tools

A.3.1 Hardware

IE-78K0S-NS In-circuit emulator		In-circuit emulator for debugging hardware and software of application system using 78K/0S Series. Supports integrated debugger (ID78K0S-NS). Used in combination with AC adapter, emulation probe, and interface adapter for connecting the host machine.
IE-70000-MC-PS-B AC adapter		Adapter for supplying power from 100 to 240 VAC outlet.
IE-70000-98-IF-C Interface adapter		Adapter required when using a PC-9800 series (except notebook type) as the host machine of IE-78K0S-NS (C bus supported).
IE-70000-CD-IF-A PC card interface		PC card and interface cable required when using a notebook type PC as the host machine of IE-78K0S-NS (PCMICA socket supported).
IE-70000-PC-IF-C Interface adapter		Adapter required when using IBM PC/AT and compatibles as the host machine of IE-78K0S-NS (ISA bus supported).
IE-70000-PCI-IF Interface adapter		Adapter required when using a personal computer incorporating the PCI bus is used as the host machine of IE-78K0S-NS.
IE-789046-NS-EM1 + NP-K907 Emulation board		Emulation board for emulating the peripheral hardware inherent to the device. Used in combination with in-circuit emulator.
NP-44GB Emulation probe		Cable for connecting the in-circuit emulator and target system. Used in combination with the NGS-30 when supporting 30-pin plastic SSOP (MC-5A4 type).
	NGS-30 Conversion socket	Conversion socket used to connect a target system board designed to allow mounting of the 30-pin plastic SSOP and the NP-36GS.

Remark NP-36GS, NGS-30, and NP-K907 are products of Naito Densei Machida Mfg. Co., Ltd.
For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (+81-44-822-3813)

A.3.2 Software

ID78K0S-NS Integrated debugger (Supports in-circuit emulator IE-78K0S-NS)	Control program for debugging the 78K/0S Series. This program provides a graphical user interface. It runs on Windows for personal computer users and on OSF/Motif™ for engineering work station users, and has visual designs and operability that comply with these operating systems. In addition, it has a powerful debug function that supports C language. Therefore, trace results can be displayed at a C language level by the window integration function that links source program, disassembled display, and memory display, to the trace result. This software also allows users to add other function extension modules such as task debugger and system performance analyzer to improve the debug efficiency for programs using a real-time operating system. Used in combination with optional device file (DF789076).
	Part number: μS××××ID78K0S-NS

Remark ×××× in the part number differs depending on the host machines and operating system to be used.

μS××××ID78K0S-NS

××××	Host Machine	OS	Supply Media
AA13	PC-9800 series	Japanese Windows[Note]	3.5" 2HD FD
AB13	IBM PC/AT and compatibles	Japanese Windows[Note]	3.5" 2HC FD
BB13		English Windows[Note]	

Note Also operates under the DOS environment.

SM78K0S System simulator	Debugs program at C source level or assembler level while simulating operation of target system on host machine. SM78K0S runs on Windows. By using SM78K0S, the logic and performance of an application can be verified independently of hardware development even when the in-circuit emulator is not used. This enhances development efficiency and improves software quality. Used in combination with optional device file (DF789076).
	Part number: μS××××SM78K0S
DF789076[Note] Device file	File containing the information inherent to the device. Used in combination with other optional tools (RA78K0S, CC78K0S, SM78K0S).
	Part number: μS××××DF789076

Note DF789076 is a common file that can be used with RA78K0S, CC78K0S, and SM78K0S.

Remark ×××× in the part number differs depending on the host machines and operating system to be used.

μS××××SM78K0S

××××	Host Machine	OS	Supply Media
AA13	PC-9800 series	Japanese Windows[Note]	3.5" 2HD FD
AB13	IBM PC/AT and compatibles	Japanese Windows[Note]	3.5" 2HC FD
BB13		English Windows[Note]	

Note Also operates under the DOS environment.

[MEMO]

APPENDIX B EMBEDDED SOFTWARE

The following embedded software products are available for efficient program development and maintenance of the μPD789074 Subseries.

MX78K0S OS	MX78K0S is a subset OS that is based on the μITRON specification and is supplied with the MX78K0S nucleus. The MX78K0S OS controls tasks, events, and time. In task control, the MX78K0S OS controls task execution order, and switches processing to the task to be executed next. **<Caution when used under the PC environment>** MX78K0S is a DOS-based application. Use this software from the DOS prompt when running it on Windows.
	Part number: μS××××MX78K0S

Remark ××× in the part number differs depending on the host machines and operating system to be used.

μS××××MX78K0S

××××	Host Machine	OS	Supply Media
AA13	PC-9800 series	Japanese Windows[Note]	3.5" 2HD FD
AB13	IBM PC/AT compatibles	Japanese Windows[Note]	3.5" 2HC FD
BB13		English Windows[Note]	

Note Also operates under the DOS environment.

[MEMO]

APPENDIX C REGISTER INDEX

C.1 Register Name Index (Alphabetic Order)

16-bit capture register 90 (TCP90) 83
16-bit compare register 90 (CR90) 83
16-bit timer counter 90 (TM90) 83
16-bit timer mode control register 90 (TMC90) 84
8-bit compare register 80 (CR80) 96
8-bit timer counter 80 (TM80) 96
8-bit timer mode control register 80 (TMC80) 97

[A]

Asynchronous serial interface mode register 20 (ASIM20) 118, 124, 127, 138
Asynchronous serial interface status register 20 (ASIS20) 120, 128

[B]

Baud rate generator control register 20 (BRGC20) 121, 129, 139
Buzzer output control register 90 (BZC90) 86

[E]

External interrupt mode register 0 (INTM0) 153

[I]

Interrupt mask flag register 0, 1 (MK0, MK1) 152
Interrupt request flag register 0, 1 (IF0, IF1) 151

[O]

Oscillation stabilization time selection register (OSTS) 164

[P]

Port 0 (P0) 61
Port 1 (P1) 62
Port 2 (P2) 63
Port 3 (P3) 67
Port mode register 0 (PM0) 68
Port mode register 1 (PM1) 68
Port mode register 2 (PM2) 68, 98
Port mode register 3 (PM3) 68, 87
Processor clock control register (PCC) 74
Pull-up resistor option register 0 (PU0) 69
Pull-up resistor option register B2 (PUB2) 70

[R]

Receive buffer register 20 (RXB20) 116

[S]

Serial operation mode register 20 (CSIM20)....117, 124, 126, 137

[T]

Transmission shift register 20 (TXS20)....116

[W]

Watchdog timer clock selection register (WDCS)....109

Watchdog timer mode register (WDTM)....110

C.2 Register Symbol Index (Alphabetic Order)

[A]

ASIM20: Asynchronous serial interface mode register 20 ..... 118, 124, 127, 128
ASIS20 : Asynchronous serial interface status register 20 ..... 120, 128

[B]

BRGC20: Baud rate generator control register 20 ..... 121, 129, 139
BZC90: Buzzer output control register 90 86

[C]

CR80: 8-bit compare register 80 96
CR90: 16-bit compare register 90 83
CSIM20: Serial operation mode register 20 ..... 117, 124, 126, 137

[I]

IF0: Interrupt request flag register 0 151
IF1: Interrupt request flag register 1 151
INTM0: External interrupt mode register 0 153

[M]

MK0: Interrupt mask flag register 0 152
MK1: Interrupt mask flag register 1 152

[O]

OSTS: Oscillation stabilization time selection register 164

[P]

P0: Port 0 61
P1: Port 1 62
P2: Port 2 63
P3: Port 3 67
PCC: Processor clock control register 74
PM0: Port mode register 0 68
PM1: Port mode register 1 68
PM2: Port mode register 2 ..... 68, 98
PM3: Port mode register 3 ..... 68, 87
PU0: Pull-up resistor option register 0 69
PUB2: Pull-up resistor option register B2 70

[R]

RXB20: Receive buffer register 20 116

[T]

TCP90: 16-bit capture register 90 83
TM80: 8-bit timer counter 80 96
TM90: 16-bit timer counter 90 83
TMC80: 8-bit timer mode control register 80 97

TMC90: 16-bit timer mode control register 90 84
TXS20: Transmission shift register 20 116

[W]

WDCS: Watchdog timer clock selection register 109
WDTM: Watchdog timer mode register 110

NEC

Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

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

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-435-9608

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	❑	❑	❑	❑