NEC

Preliminary User's Manual

μPD78F9500, 78F9501, 78F9502

8-Bit Single-Chip Microcontrollers

μPD78F9500
μPD78F9501
μPD78F9502

Document No. U18681EJ1V0UD00 (1st edition)
Date Published June 2007 NS

[MEMO]

NOTES FOR CMOS DEVICES

① **VOLTAGE APPLICATION WAVEFORM AT INPUT PIN**

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (MAX) and V_{IH} (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (MAX) and V_{IH} (MIN).

② **HANDLING OF UNUSED INPUT PINS**

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ **PRECAUTION AGAINST ESD**

A 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 when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up 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 benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ **STATUS BEFORE INITIALIZATION**

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

⑤ **POWER ON/OFF SEQUENCE**

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current.

The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

⑥ **INPUT OF SIGNAL DURING POWER OFF STATE**

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements.

Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
PC/AT is a trademark of International Business Machines Corporation.
HP9000 series 700 and HP-UX are trademarks of Hewlett-Packard Company.
SPARCstation is a trademark of SPARC International, Inc.
Solaris and SunOS are trademarks of Sun Microsystems, Inc.
SuperFlash is a registered trademark of Silicon Storage Technology, Inc. in several countries including the United States and Japan.

Caution: This product uses SuperFlash® technology licensed from Silicon Storage Technology, inc.

INTRODUCTION

Target Readers

This manual is intended for user engineers who wish to understand the functions of the μPD78F9500, 78F9501, 78F9502 in order to design and develop its application systems and programs.

Purpose

This manual is intended to give users on understanding of the functions described in the **Organization** below.

Organization

Two manuals are available for the μPD78F9500, 78F9501, 78F9502: this manual and the Instruction Manual (common to the 78K/0S Series).

μPD78F9500, 78F9501, 78F9502 User's Manual	78K/0S Series Instructions User's Manual
• Pin functions • Internal block functions • Interrupts • Other internal peripheral functions • Electrical specifications	• CPU function • Instruction set • Instruction description

How to Use 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 μPD78F9500, 78F9501, 78F9502
→ Read this manual in the order of the **CONTENTS**.
◊ How to read register formats
→ For a bit number enclosed in angle brackets (<>), the bit name is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S.
◊ To learn the detailed functions of a register whose register name is known
→ See **APPENDIX B REGISTER INDEX**.
◊ To learn the details of the instruction functions of the 78K/0S Series
→ Refer to **78K/0S Series Instructions User's Manual (U11047E)** separately available.
◊ To learn the electrical specifications of the μPD78F9500, 78F9501, 78F9502
→ See **CHAPTER 16 ELECTRICAL SPECIFICATIONS (TARGET)**.

Conventions

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 with **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.
μPD78F9500, 78F9501, 78F9502 User's Manual	This manual
78K/0S Series Instructions User's Manual	U11047E

Documents Related to Development Software Tools (User's Manuals)

Document Name		Document No.
RA78K0S Assembler Package	Operation	U16656E
	Language	U14877E
	Structured Assembly Language	U11623E
CC78K0S C Compiler	Operation	U16654E
	Language	U14872E
ID78K0S-QB Ver. 2.81 Integrated Debugger	Operation	U17287E
PM plus Ver.5.20		U16934E

Documents Related to Development Hardware Tools (User's Manuals)

Document Name	Document No.
QB-78K0SKX1 In-Circuit Emulator	U18219E
QB-MINI2 On-chip debug emulator with programming function	U18371E

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

Documents Related to Flash Memory Writing

Document Name	Document No.
PG-FP4 Flash Memory Programmer User's Manual	U15260E

Other Related Documents

Document Name	Document No.
SEMICONDUCTOR SELECTION GUIDE - Products and Packages -	X13769X
Semiconductor Device Mount Manual	**Note**
Quality Grades on NEC Semiconductor Devices	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983E
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E

Note See the "Semiconductor Device Mount Manual" website (http://www.necel.com/pkg/en/mount/index.html).

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

CONTENTS

CHAPTER 1 OVERVIEW

1.1 Features

O 78K0S CPU core

O ROM and RAM capacities

Item / Part number	Program Memory (Flash Memory)	Memory (Internal High-Speed RAM)
μPD78F9500	1 KB	128 bytes
μPD78F9501	2 KB	
μPD78F9502	4 KB	

O Minimum instruction execution time: 0.2 μs (with 10 MHz@4.0 to 5.5 V operation)

O Clock

- High-speed system clock … Selected from the following two sources
  - External clock: 1 to 10 MHz
  - High-speed internal oscillator: 8 MHz ±2%
- Low-speed internal oscillator 240 kHz (TYP.) … Watchdog timer, timer clock in intermittent operation

O I/O ports: 8 (CMOS I/O: 7, CMOS input: 1)

O Timer: 2 channels

- 8-bit timer: 1 channel … PWM output × 1
- Watchdog timer: 1 channel … Operable with low-speed internal oscillation clock

O On-chip power-on-clear (POC) circuit (A reset is automatically generated when the voltage drops to 2.1 V ±0.1 V or below)

O On-chip low voltage detector (LVI) circuit (An interrupt/reset (selectable) is generated when the detection voltage is reached)

- Detection voltage: Selectable from ten levels between 2.35 and 4.3 V

O Single-power-supply flash memory

- Flash self programming enabled
- Software protection function: Protected from outside party copying (no flash reading command)
- Time required for writing by dedicated flash memory programmer: Approximately 3 seconds (4 KB)
  - ∗ Write operations on mass production lines supported

O Safety function

- Watchdog timer operated by clock independent from CPU
  … A hang-up can be detected even if the system clock stops
- Supply voltage drop detectable by LVI
  … Appropriate processing can be executed before the supply voltage drops below the operation voltage
- Equipped with option byte function
  … Important system operation settings set in hardware

O Assembly and C language supported

O Enhanced development environment

- Support for full-function emulator (IECUBE), simplified emulator (MINICUBE2), and simulator

O Supply voltage: V_{DD} = 2.0 to 5.5 V

∗ Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

O Operating temperature range: T_A = −40 to +85°C

1.2 Ordering Information

Part Number

μPD78F9 ××× - ×× (×) - ××× -A

	Semiconductor component
Blank	Conventional
-A	Lead-free

	Quality Grades
Blank	Standard (General management)

	Package type
MA-CAC	Plastic SSOP

	Number of pins	High-speed RAM	Flash memory
500	10 pins	128 bytes	1 K bytes
501	10 pins	128 bytes	2 K bytes
502	10 pins	128 bytes	4 K bytes

	Product type
F	Flash memory versions

[Part number list]

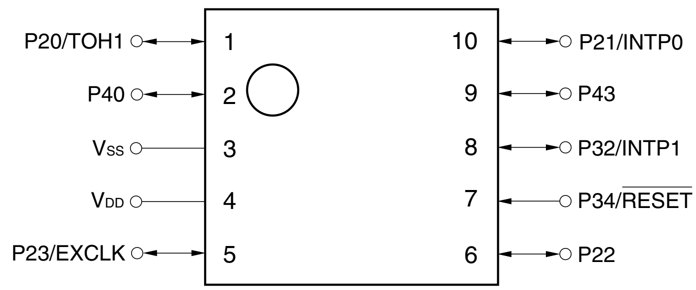
μ PD78F9500MA-CAC-A

μ PD78F9501MA-CAC-A

μ PD78F9502MA-CAC-A

1.3 Pin Configuration (Top View)

10-pin plastic SSOP



INTP0, INTP1:	External interrupt input	TOH1:	Timer output
P20 to P23:	Port 2	V_{DD}:	Power supply
P30, P34:	Port 3	V_{SS}:	Ground
P40, P43:	Port 4	EXCLK:	External Clock Input
$\overline{\text{RESET}}$:	Reset		(Main System Clock)

1.4 78K0S/Kx1+ Product Lineup

The following table shows the product lineup of the 78K0S/Kx1+.

<table>
<tr><th colspan="2">Item \ Part Number</th><th>78K0S/KU1+</th><th>78K0S/KY1+</th><th colspan="2">78K0S/KA1+</th><th>78K0S/KB1+</th></tr>
<tr><td colspan="2">Number of pins</td><td>10 pins</td><td>16 pins</td><td colspan="2">20 pins</td><td>30 pins</td></tr>
<tr><td rowspan="2">Internal memory</td><td>Flash memory</td><td colspan="2">1 KB, 2 KB, 4 KB</td><td>2 KB</td><td>4 KB</td><td>4 KB, 8 KB</td></tr>
<tr><td>RAM</td><td colspan="2">128 bytes</td><td>128 bytes</td><td>256 bytes</td><td>256 bytes</td></tr>
<tr><td colspan="2">Supply voltage</td><td colspan="5">V_{DD} = 2.0 to 5.5 V[Note 1]</td></tr>
<tr><td colspan="2">Minimum instruction execution time</td><td colspan="5">0.20 μs (10 MHz, V_{DD} = 4.0 to 5.5 V)
0.33 μs (6 MHz, V_{DD} = 3.0 to 5.5 V)
0.40 μs (5 MHz, V_{DD} = 2.7 to 5.5 V)
1.0 μs (2 MHz, V_{DD} = 2.0 to 5.5 V)</td></tr>
<tr><td colspan="2">System clock (oscillation frequency)</td><td colspan="5">High-speed internal oscillation (8 MHz (TYP.))
Crystal/ceramic oscillation (1 to 10 MHz)[Note 2]
External clock input oscillation (1 to 10 MHz)</td></tr>
<tr><td colspan="2">Clock for TMH1 and WDT (oscillation frequency)</td><td colspan="5">Low-speed internal oscillation (240 kHz (TYP.))</td></tr>
<tr><td rowspan="3">Port</td><td>CMOS I/O</td><td>7</td><td>13</td><td colspan="2">15</td><td>24</td></tr>
<tr><td>CMOS input</td><td>1</td><td>1</td><td colspan="2">1</td><td>1</td></tr>
<tr><td>CMOS output</td><td>–</td><td>–</td><td colspan="2">1</td><td>1</td></tr>
<tr><td rowspan="4">Timer</td><td>16-bit (TM0)</td><td colspan="5">1 ch[Note 3]</td></tr>
<tr><td>8-bit (TMH)</td><td colspan="5">1 ch</td></tr>
<tr><td>8-bit (TM8)</td><td colspan="2">–</td><td colspan="3">1 ch</td></tr>
<tr><td>WDT</td><td colspan="5">1 ch</td></tr>
<tr><td colspan="2">Serial interface</td><td colspan="2">–</td><td colspan="3">LIN-Bus-supporting UART: 1 ch</td></tr>
<tr><td colspan="2">A/D converter[Note 4]</td><td colspan="5">10 bits: 4 ch (2.7 to 5.5 V)[Note 4]</td></tr>
<tr><td colspan="2">Multiplier (8 bits × 8 bits)</td><td colspan="4">–</td><td>Provided</td></tr>
<tr><td rowspan="2">Interrupts</td><td>External</td><td colspan="2">2</td><td colspan="3">4</td></tr>
<tr><td>Internal</td><td colspan="2">5[Note 5]</td><td colspan="3">9</td></tr>
<tr><td rowspan="4">Reset</td><td>$\overline{\text{RESET}}$ pin</td><td colspan="5">Provided</td></tr>
<tr><td>POC</td><td colspan="5">2.1 V ±0.1 V</td></tr>
<tr><td>LVI</td><td colspan="5">Provided (selectable by software)</td></tr>
<tr><td>WDT</td><td colspan="5">Provided</td></tr>
<tr><td colspan="2">Operating temperature range</td><td>Standard products: –40 to +85°C</td><td colspan="4">Standard products, (A) grade products: –40 to +85°C
(A2) grade products: –40 to +125°C</td></tr>
</table>

Note 1. Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

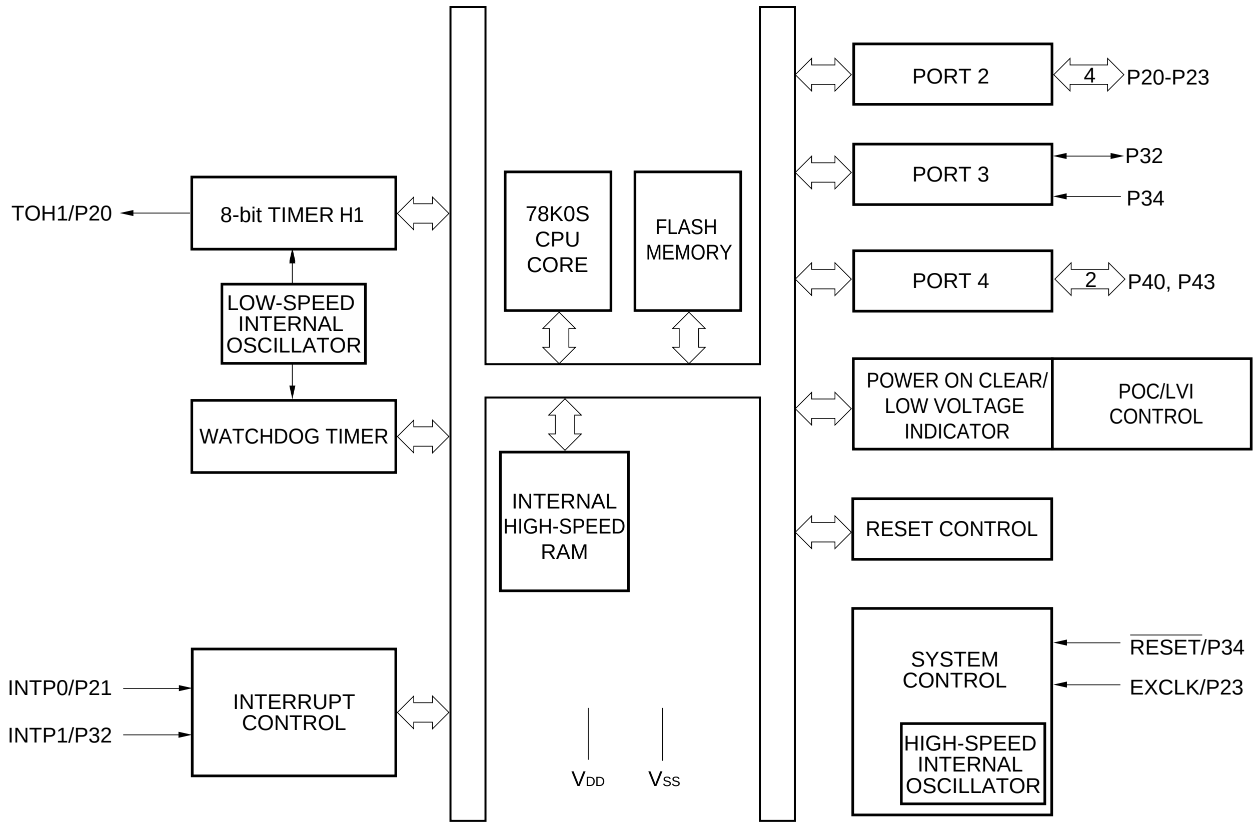
2. This product (μPD78F950x) does not support crystal/ceramic oscillation

3. The product without A/D converter (μPD78F950x) in the 78K0S/KU1+ has no 16 bit timer (TM0).

4. The product without A/D converter (μPD78F95xx) is provided for the 78K0S/KU1+ and 78K0S/KY1+ respectively. This product has no A/D converter.

5. The product without A/D converter in the 78K0S/KU1+ has 2 factors, while the products without A/D converter in the 78K0S/KY1+ has 4 factors.

1.5 Block Diagram



1.6 Functional Outline

Item			μPD78F9500	μPD78F9501	μPD78F9502
Internal memory	Flash memory		1 KB	2 KB	4 KB
	High-speed RAM		128 bytes		
Memory space			64 KB		
Clock	Main	High-speed system clock	External clock input: 10 MHz (V_{DD} = 2.7 to 5.5 V)		
		Internal high-speed oscillation clock	8 MHz (TYP.)		
	Internal low-speed oscillation clock		240 kHz (TYP.)		
General-purpose registers			8 bits × 8 registers		
Instruction execution time			0.2 μs/0.4 μs/0.8 μs/1.6 μs/3.2 μs (high-speed system clock: f_X = 10 MHz)		
I/O port			Total: 8 pins CMOS I/O: 7 pins CMOS input: 1 pin		
Timer			• 8-bit timer (timer H1): 1 channel • Watchdog timer: 1 channel		
	Timer output		2 pins (PWM: 1 pin)		
Vectored interrupt sources	External		2		
	Internal		2		
Reset			• Reset by $\overline{\text{RESET}}$ pin • Internal reset by watchdog timer • Internal reset by power-on clear • Internal reset by low-voltage detector		
Supply voltage			V_{DD} = 2.0 to 5.5 V[Note]		
Operating temperature range			T_A = –40 to +85°C		
Package			10-pin plastic SSOP		

Note Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on- clear (POC) circuit is 2.1 V ±0.1 V.

2.1 Pin Function List

(1) Port pins

<table>
<tr><th>Pin Name</th><th>I/O</th><th colspan="2">Function</th><th>After Reset</th><th>Alternate-Function Pin</th></tr>
<tr><td>P20</td><td rowspan="4">I/O</td><td colspan="2" rowspan="4">Port 2.
4-bit I/O port.
Can be set to input or output mode in 1-bit units.
An on-chip pull-up resistor can be connected by setting software.</td><td rowspan="4">Input port</td><td>TOH1</td></tr>
<tr><td>P21</td><td>INTP0</td></tr>
<tr><td>P22</td><td>–</td></tr>
<tr><td>P23[Note]</td><td>EXCLK</td></tr>
<tr><td>P32</td><td>I/O</td><td rowspan="2">Port 3
An on-chip pull-up resistor can be connected by setting software.</td><td>Can be set to input or output mode in 1-bit units.</td><td>Input port</td><td>INTP1</td></tr>
<tr><td>P34[Note]</td><td>Input</td><td>Input only</td><td>Input port</td><td>$\overline{\text{RESET}}$[Note]</td></tr>
<tr><td>P40, P43</td><td>I/O</td><td colspan="2">Port 4.
2-bit I/O port.
Can be set to input or output mode in 1-bit units.
An on-chip pull-up resistor can be connected by setting software.</td><td>Input port</td><td>–</td></tr>
</table>

Note For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

Caution The P22 and P23/EXCLK pins are pulled down during reset. The P34/$\overline{\text{RESET}}$ pin is pulled up during reset by the reset pin function/power-on clear circuit.

(2) Non-port pins

<table>
<tr><th>Pin Name</th><th>I/O</th><th>Function</th><th>After Reset</th><th>Alternate-Function Pin</th></tr>
<tr><td>INTP0</td><td rowspan="2">Input</td><td rowspan="2">External interrupt input for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified</td><td rowspan="2">Input port</td><td>P21</td></tr>
<tr><td>INTP1</td><td>P32</td></tr>
<tr><td>TOH1</td><td>Output</td><td>8-bit timer H1 output</td><td>Input port</td><td>P20</td></tr>
<tr><td>$\overline{\text{RESET}}$ [Note]</td><td>Input</td><td>System reset input</td><td>Input port</td><td>P34[Note]</td></tr>
<tr><td>EXCLK[Note]</td><td>Input</td><td>External clock input for main system clock</td><td>Input port</td><td>P23[Note]</td></tr>
<tr><td>V_{DD}</td><td>–</td><td>Positive power supply</td><td>–</td><td>–</td></tr>
<tr><td>V_{SS}</td><td>–</td><td>Ground potential</td><td>–</td><td>–</td></tr>
</table>

Note For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

Caution The P22 and P23/EXCLK pins are pulled down during reset. The P34/$\overline{\text{RESET}}$ pin is pulled up during reset by the reset pin function/power-on clear circuit.

2.2 Pin Functions

2.2.1 P20 to P23 (Port 2)

P20 to P23 constitute a 4-bit I/O port. In addition to the function as I/O port pins, these pins also have a function to output a timer signal, input an external interrupt request signal, and input an external clock for the main system clock.

P23 also functions as the EXCLK. For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

These pins can be set to the following operation modes in 1-bit units.

(1) Port mode

P20 to P23 function as a 4-bit I/O port. Each bit of this port can be set to the input or output mode by using port mode register 2 (PM2). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 2 (PU2).

(2) Control mode

P20 to P23 function to output a timer signal, and input an external interrupt request signal.

(a) TOH1

This pin outputs a signal from 8-bit timer H1.

(b) INTP0

This is an external interrupt request input pin for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified.

(c) EXCLK

This pin inputs and external clock input for the main system clock.

Caution The P22 and P23/EXCLK pins are pulled down during reset.

2.2.2 P32 and P34 (Port 3)

P32 is a 1-bit I/O port. In addition to the function as an I/O port pin, this pin also has a function to input an external interrupt request signal.

P34 is a 1-bit input-only port. This pin is also used as a $\overline{\text{RESET}}$ pin, and when the power is turned on, this is the reset function.

For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

When P34 is used as an input port pin, connect the pull-up resistor.

P32 and P34 can be set to the following operation modes in 1-bit units.

(1) Port mode

P32 functions as a 1-bit I/O port. This pin can be set to the input or output mode by using port mode register 3 (PM3). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 3 (PU3).

P34 functions as a 1-bit input-only port.

(2) Control mode

P32 functions as an external interrupt request input pin (INTP1) for which the valid edge (rising edge, falling edge, or both rising and falling edges) can be specified. .

Caution The P34/$\overline{\text{RESET}}$ pin is pulled up during reset by the reset pin function/power-on clear circuit.

2.2.3 P40 and P43 (Port 4)

P40 and P43 constitute a 2-bit I/O port. Each bit of this port can be set to the input or output mode by using port mode register 4 (PM4). In addition, an on-chip pull-up resistor can be connected to the port by using pull-up resistor option register 4 (PU4).

2.2.4 $\overline{\text{RESET}}$

This pin inputs an active-low system reset signal. When the power is turned on, this is the reset function, regardless of the option byte setting.

Caution The P34/$\overline{\text{RESET}}$ pin is pulled up during reset by the reset pin function/power-on clear circuit.

2.2.5 V_{DD}

This is the positive power supply pin.

2.2.6 V_{SS}

This is the ground pin.

Be sure to connect V_{SS} to a stabilized GND (= 0 V).

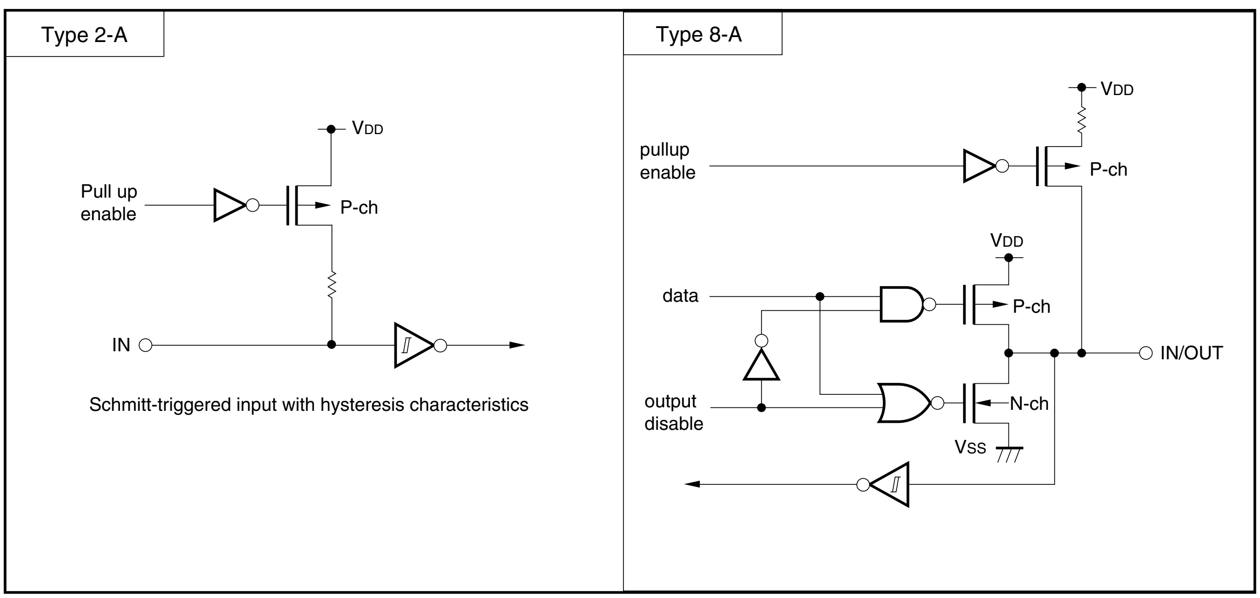
2.3 Pin I/O Circuits and Connection of Unused Pins

Table 2-1 shows I/O circuit type of each pin and the connections of unused pins.
For the configuration of the I/O circuit of each type, refer to **Figure 2-1**.

Table 2-1. Types of Pin I/O Circuits and Connection of Unused Pins

Pin Name	I/O Circuit Type	I/O	Recommended Connection of Unused Pin
P20/TOH1	8-A	I/O	Input: Individually connect to V_{DD} or V_{SS} via resistor. Output: Leave open.
P21/INTP0			
P22			
P23/EXCLK			
P32/INTP1			
P34/$\overline{RESET}$	2-A	Input	Set ENPU34 to "1" on the option byte, and leave the pin open.
P40 and P43	8-A	I/O	Input: Individually connect to V_{DD} or V_{SS} via resistor. Output: Leave open.

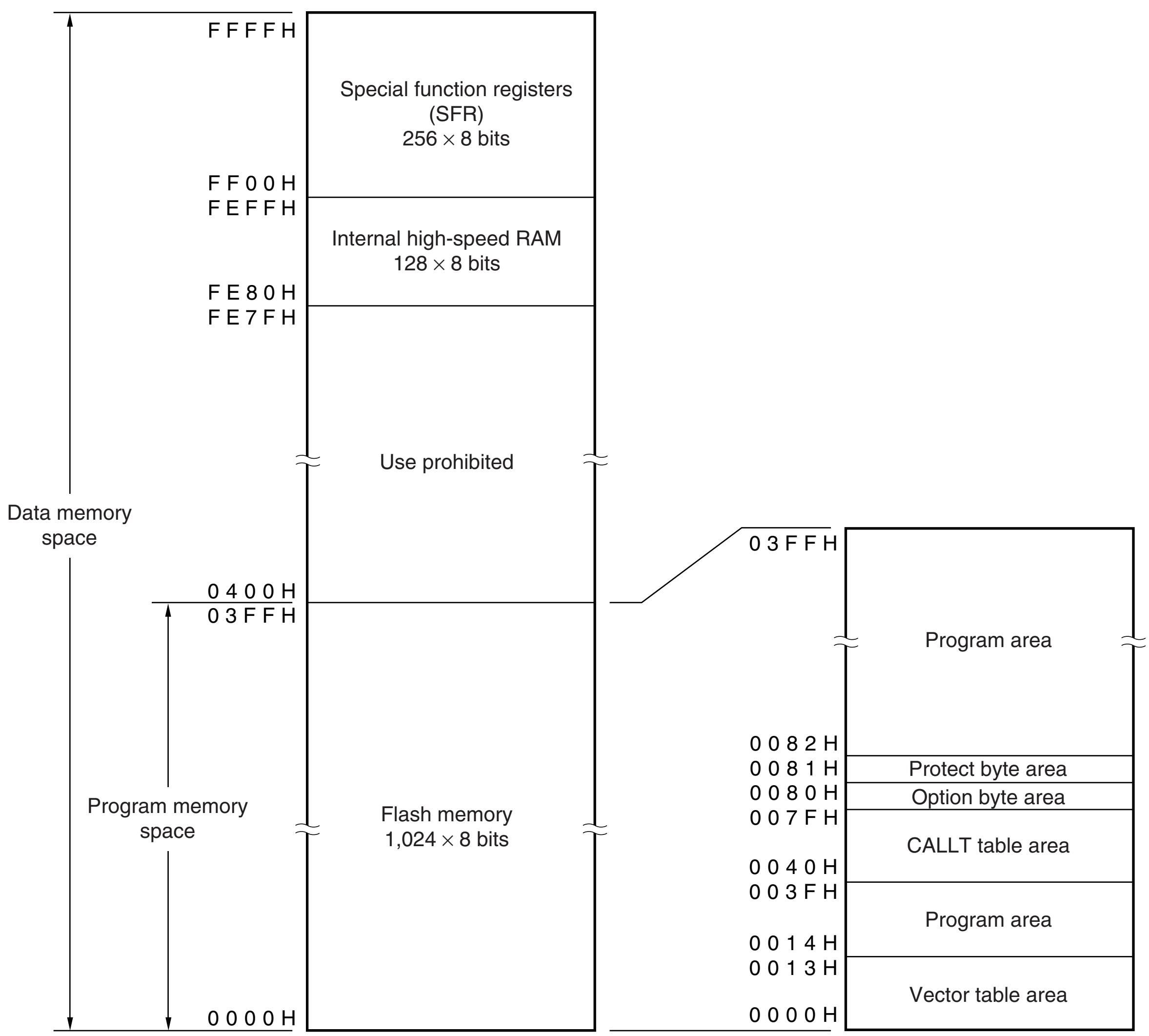
Figure 2-1. Pin I/O Circuits



3.1 Memory Space

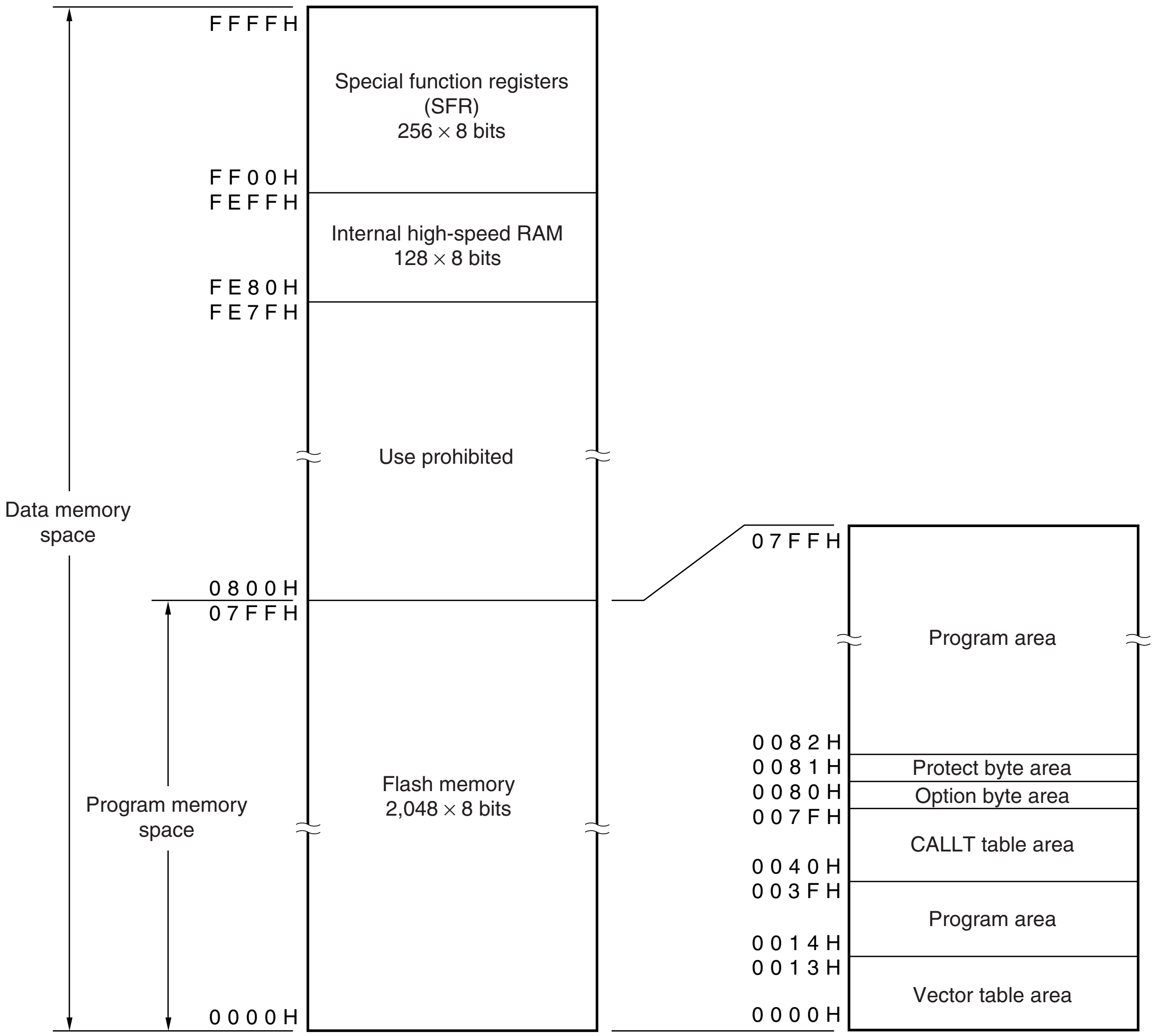
The μPD78F9500, 78F9501, 78F9502 can access up to 64 KB of memory space. Figures 3-1 to 3-3 show the memory maps.

Figure 3-1. Memory Map (μPD78F9500)



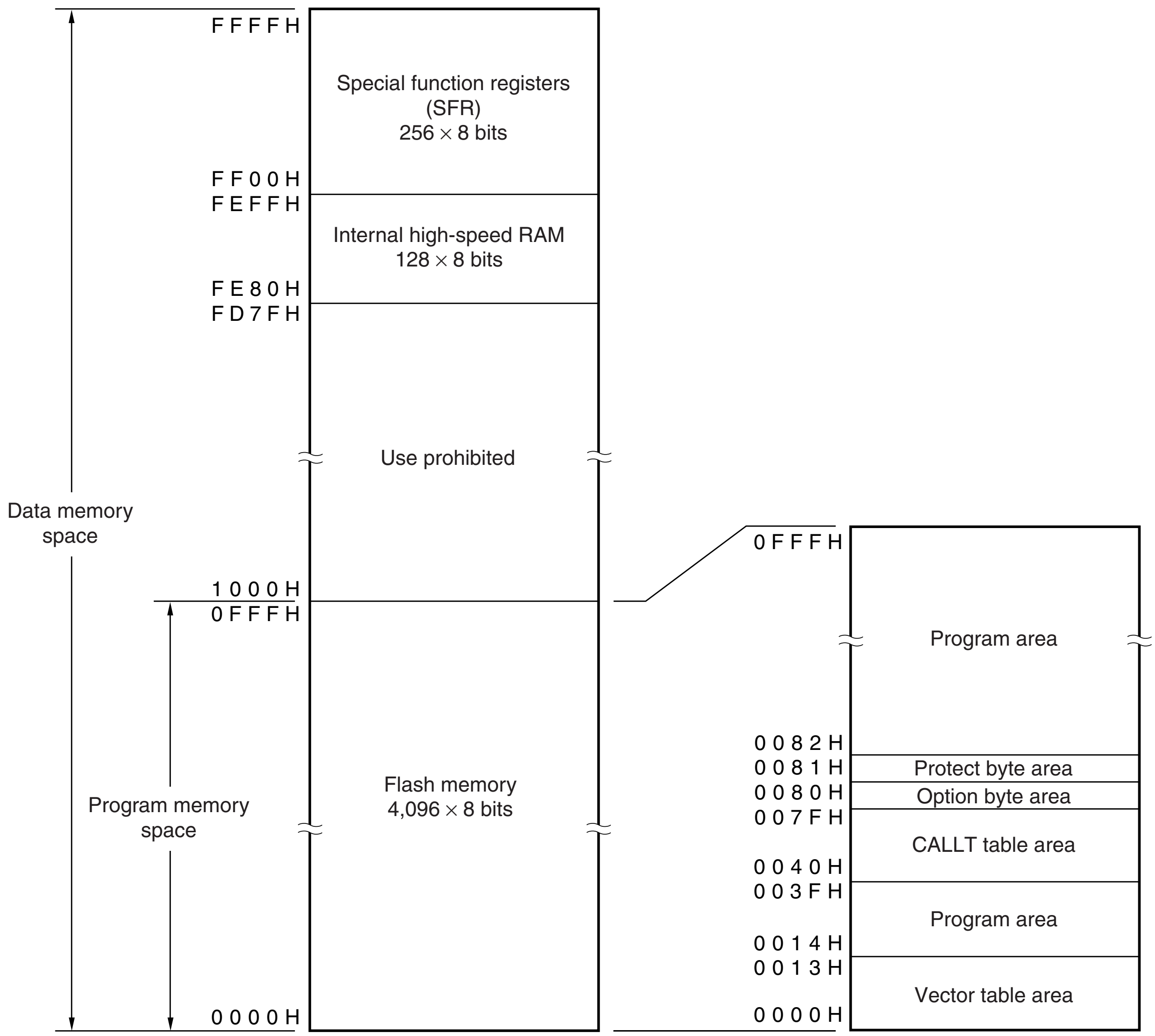
Remark The option byte and protect byte are 1 byte each.

Figure 3-2. Memory Map (μPD78F9501)



Remark The option byte and protect byte are 1 byte each.

Figure 3-3. Memory Map (μPD78F9502)



Remark The option byte and protect byte are 1 byte each.

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 μPD78F9500, 78F9501, 78F9502 provide the following internal ROMs (or flash memory) containing the following capacities.

Table 3-1. Internal ROM Capacity

Part Number	Internal ROM	
	Structure	Capacity
μPD78F9500	Flash memory	1,024 × 8 bits
μPD78F9501		2,048 × 8 bits
μPD78F9502		4,096 × 8 bits

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

(1) Vector table area

The 20-byte area of addresses 0000H to 0013H is reserved as a vector table area. This area stores program start addresses to be used when branching by RESET or interrupt request generation. Of a 16-bit 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
0000H	Reset
0006H	INTLVI
0008H	INTP0
000AH	INTP1
000CH	INTTMH1

(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) Option byte area

The option byte area is the 1-byte area of address 0080H. For details, refer to **CHAPTER 13 OPTION BYTE**.

(4) Protect byte area

The protect byte area is the 1-byte area of address 0081H. For details, refer to **CHAPTER 14 FLASH MEMORY**.

3.1.2 Internal data memory space

128-byte internal high-speed RAM is provided in the μPD78F9500, 78F9501, 78F9502.

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

The μPD78F9500, 78F9501, 78F9502 are provided with a wide range of addressing modes to make memory manipulation as efficient as possible. The area (FE80H to FEFFH) which contains a data memory and the special function register (SFR) area can be accessed using a unique addressing mode in accordance with each function. Figures 3-4 to 3-6 illustrate the data memory addressing.

Figure 3-4. Data Memory Addressing (μPD78F9500)

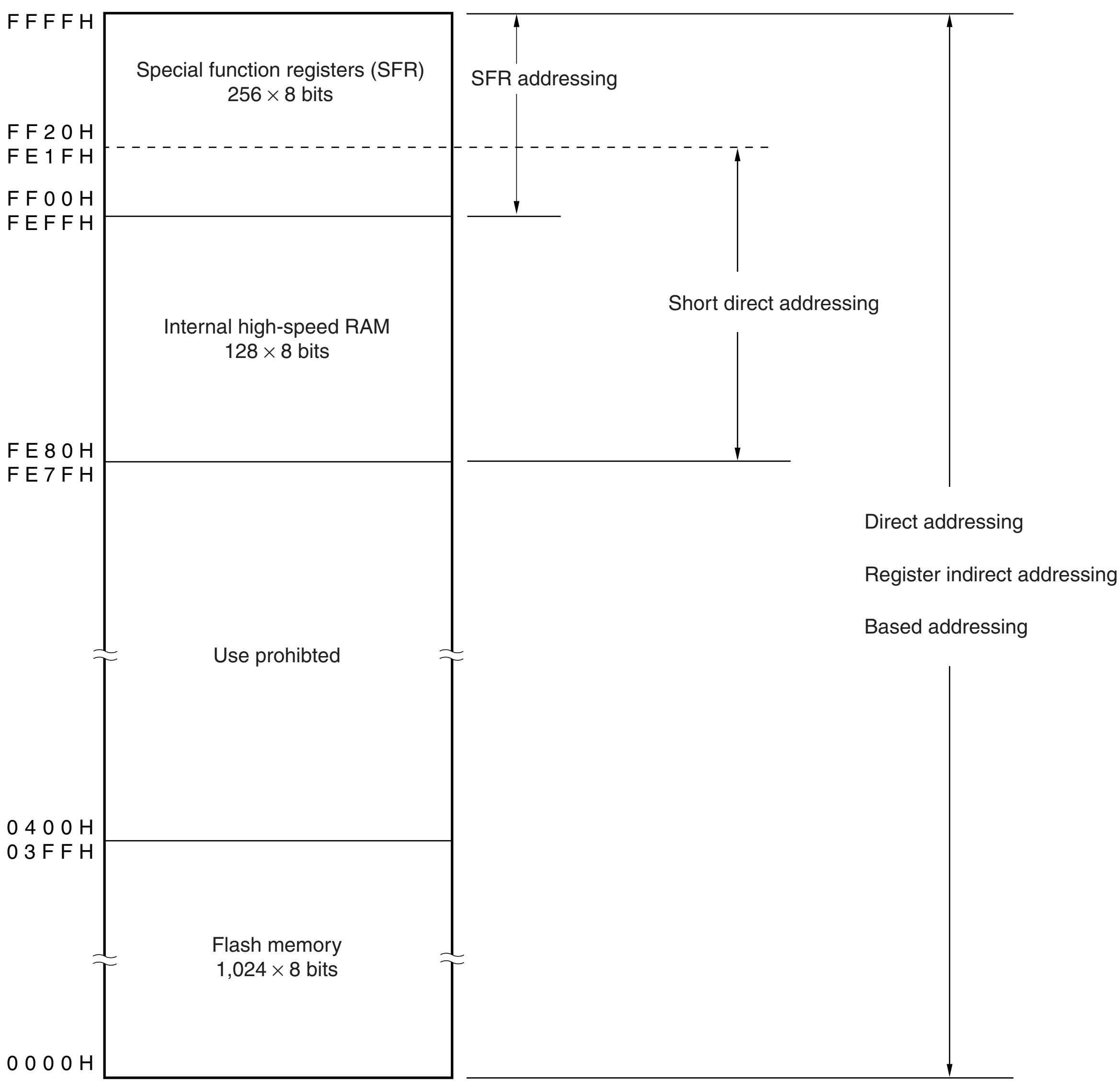


Figure 3-5. Data Memory Addressing (μPD78F9501)

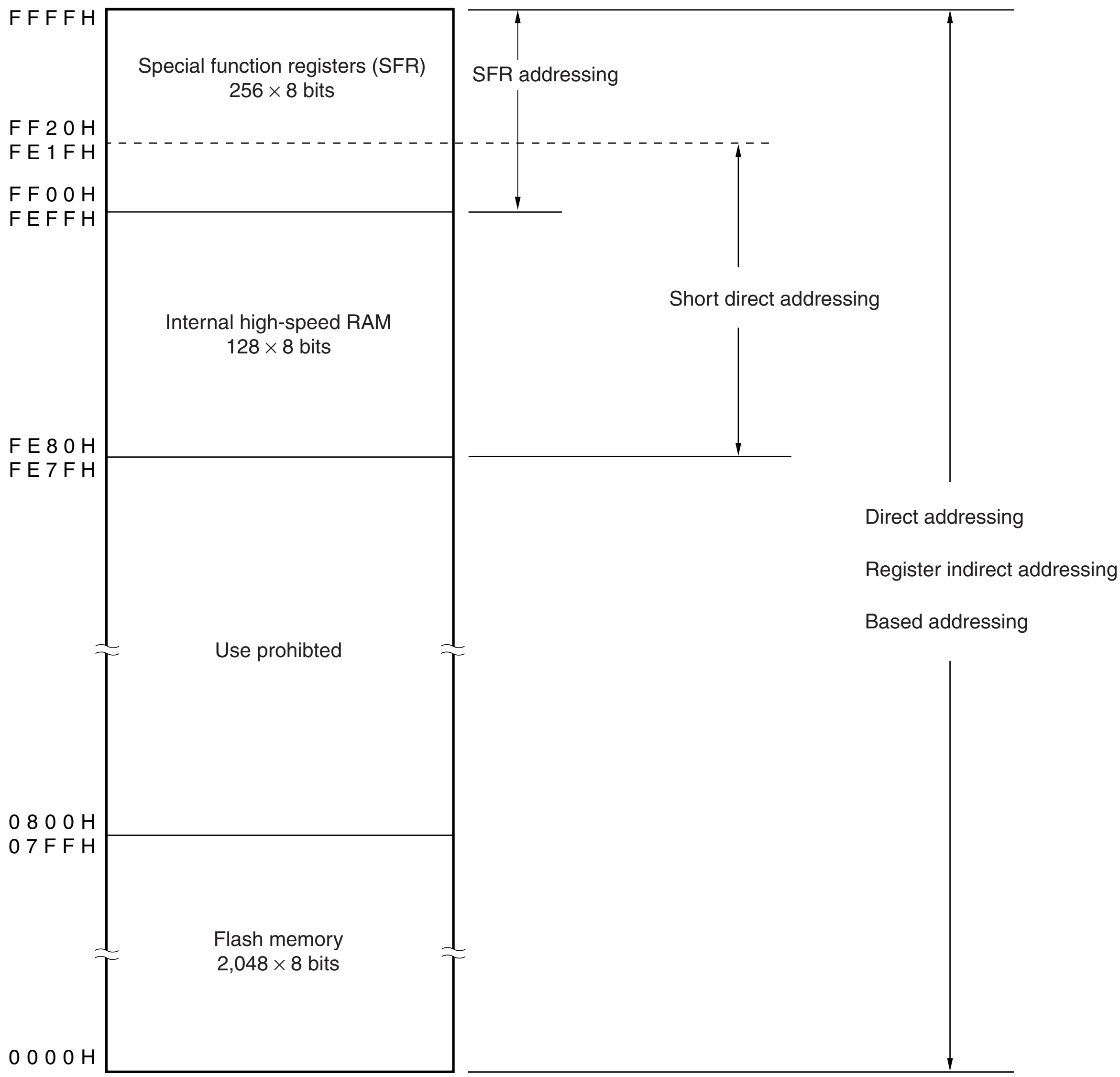
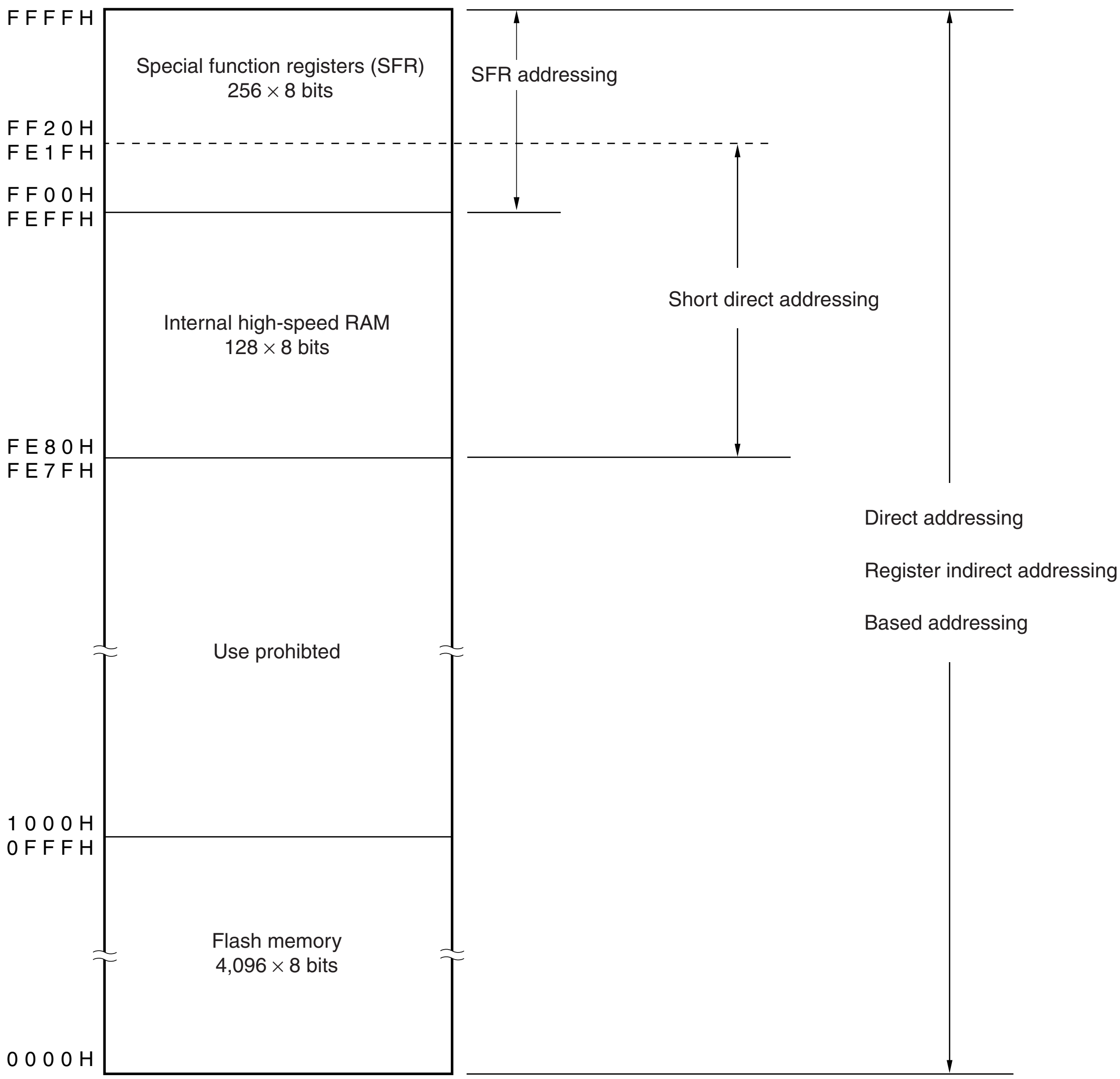


Figure 3-6. Data Memory Addressing (μPD78F9502)



3.2 Processor Registers

The μPD78F9500, 78F9501, 78F9502 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.

Reset signal generation sets the reset vector table values at addresses 0000H and 0001H to the program counter.

Figure 3-7. 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 stored in stack area upon interrupt request generation or PUSH PSW instruction execution and are restored upon execution of the RETI and POP PSW instructions.

Reset signal generation sets PSW to 02H.

Figure 3-8. 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 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 (Other than the internal high-speed RAM area cannot be set as the stack area).

Figure 3-9. 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 3-10 and 3-11.

Caution 1. Since reset signal generation makes the SP contents undefined, be sure to initialize the SP before using the stack memory.

2. Stack pointers can be set only to the high-speed RAM area, and only the lower 10 bits can be actually set.
0FF00H is in the SFR area, not the high-speed RAM area, so it was converted to 0FB00H that is in the high-speed RAM area.
When the value is actually pushed onto the stack, 1 is subtracted from 0FB00H to become 0FAFFH, but that value is not in the high-speed RAM area, so it is converted to 0FEFFH, which is the same value as when 0FF00H is set to the stack pointer.

Figure 3-10. Data to Be Saved to Stack Memory

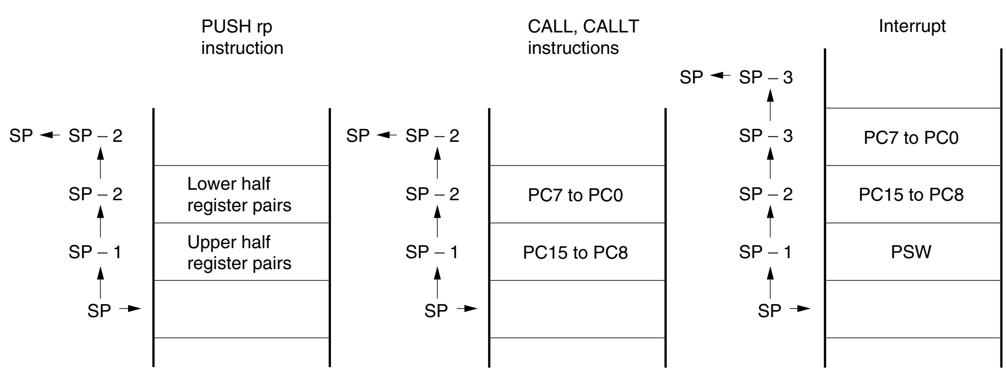
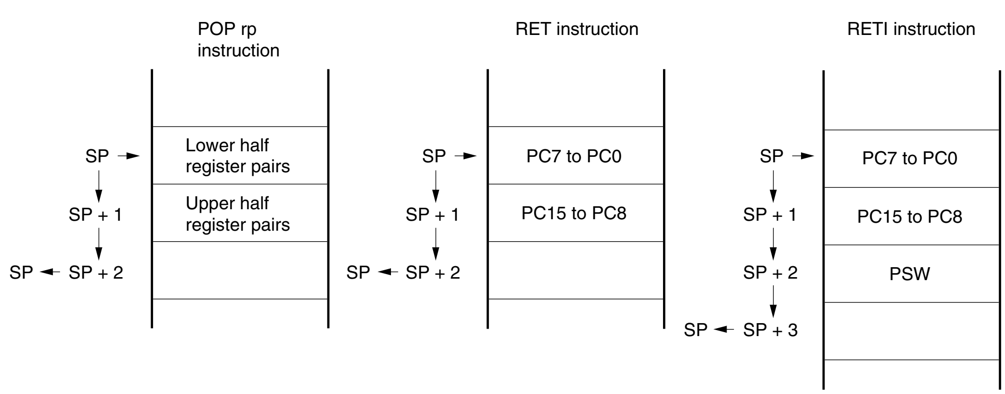


Figure 3-11. Data to Be Restored from Stack Memory



3.2.2 General-purpose registers

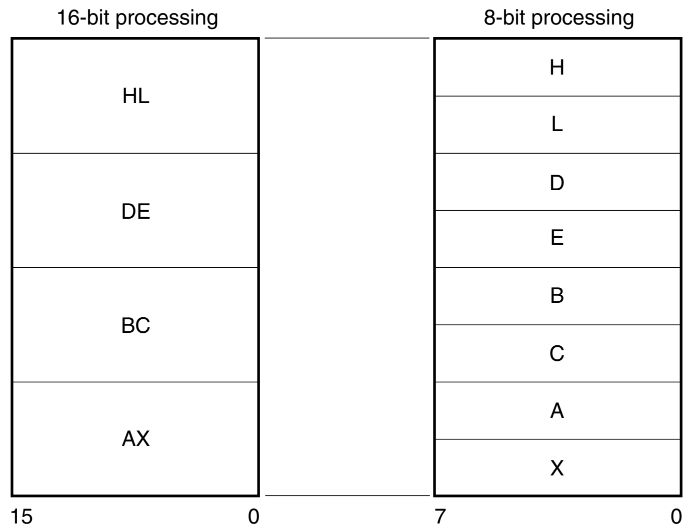
A general-purpose register consists of eight 8-bit registers (X, A, C, B, E, D, L, and H).

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).

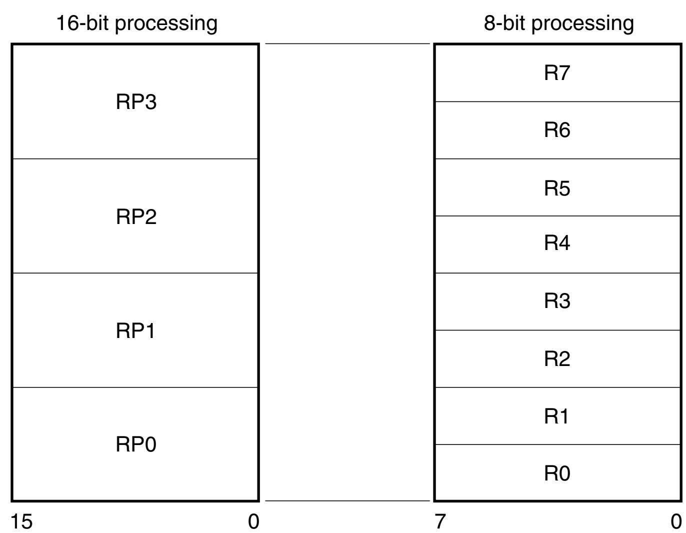
Registers can be described in terms of function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL) and absolute names (R0 to R7 and RP0 to RP3).

Figure 3-12. General-Purpose Register Configuration

(a) Function names



(b) Absolute 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 and bit.

- 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. It is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S. 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 a reset is input.

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

Address	Symbol	Special Function Register (SFR) Name								R/W	Number of Bits Manipulated Simultaneously			After Reset	Reference page
		7	6	5	4	3	2	1	0		1	8	16		
FF00H, FF01H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF02H	P2	0	0	0	0	P23	P22	P21	P20	R/W Note 1	√	√	–	00H	57
FF03H	P3	0	0	0	P34	0	P32	0	0		√	√	–	00H	57
FF04H	P4	P47	P46	P45	P44	P43	P42	P41	P40		√	√	–	00H	57
FF05H to FF0DH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF0EH	CMP01	–	–	–	–	–	–	–	–	R/W	–	√	–	00H	74
FF0FH	CMP11	–	–	–	–	–	–	–	–		–	√	–	00H	74
FF10H, FF11H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF22H	PM2	1	1	1	1	PM23	PM22	PM21	PM20	R/W	√	√	–	FFH	56
FF23H	PM3	1	1	1	1	1	PM32	1	1		√	√	–	FFH	56
FF24H	PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40		√	√	–	FFH	56
FF25H to FF31H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF32H	PU2	0	0	0	0	PU23	PU22	PU21	PU20	R/W	√	√	–	00H	58
FF33H	PU3	0	0	0	0	0	PU32	0	0		√	√	–	00H	58
FF34H	PU4	PU47	PU46	PU45	PU44	PU43	PU42	PU41	PU40		√	√	–	00H	58
FF35H to FF47H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF48H	WDTM	0	1	1	WDCS4	WDCS3	WDCS2	WDCS1	WDCS0	R/W	–	√	–	67H	90
FF49H	WDTE	–	–	–	–	–	–	–	–		–	√	–	9AH	91

Note Only P34 is an input-only port.

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

Address	Symbol	Special Function Register (SFR) Name								R/W	Number of Bits Manipulated Simultaneously			After Reset	Reference page
		7	6	5	4	3	2	1	0		1	8	16		
FF50H	LVIM	<LVION>	0	0	0	0	0	<LVIMD>	<LVIF>	R/W	√	√	–	00H[Note 1]	124
FF51H	LVIS	0	0	0	0	LVIS3	LVIS2	LVIS1	LVIS0		–	√	–	00H[Note 1]	125
FF52H, FF53H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF54H	RESF	0	0	0	WDTRF	0	0	0	LVIRF	R	–	√	–	00H[Note 2]	118
FF55H to FF57H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF58H	LSRCM	0	0	0	0	0	0	0	<LSRSTOP>	R/W	√	√	–	00H	64
FF59H to FF6FH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FF70H	TMHMD1	<TMHE1>	CKS12	CKS11	CKS10	TMMD11	TMMD10	<TOLEV1>	<TOEN1>	R/W	√	√	–	00H	78
FF71H to FF9FH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFA0H	PFCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0	W	–	√	–	Undefined	149
FFA1H	PFS	0	0	0	0	0	WEPRERR	VCERR	FPRERR	R/W	√	√	–	00H	149
FFA2H	FLPMC	0	PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	0	FLSPM		–	√	–	Undefined	148
FFA3H	FLCMD	0	0	0	0	0	FLCMD2	FLCMD1	FLCMD0		√	√	–	00H	151
FFA4H	FLAPL	FLAP7	FLAP6	FLAP5	FLAP4	FLAP3	FLAP2	FLAP1	FLAP0		√	√	–	Undefined	152

Notes 1. Retained only after a reset by LVI.

2. Varies depending on the reset cause.

Remark For a bit name enclosed in angle brackets (<>), the bit name is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S.

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

Address	Symbol	Special Function Register (SFR) Name								R/W	Number of Bits Manipulated Simultaneously			After Reset	Reference page
		7	6	5	4	3	2	1	0		1	8	16		
FFA5H	FLAPH	0	0	0	0	FLA P11	FLA P10	FLA P9	FLA P8	R/W	√	√	–	Undefined	152
FFA6H	FLAPH C	0	0	0	0	FLAP C11	FLAP C10	FLAP C9	FLAP C8		√	√	–	00H	152
FFA7H	FLAPLC	FLAP C7	FLAP C6	FLAP C5	FLAP C4	FLAP C3	FLAP C2	FLAP C1	FLAP C0		√	√	–	00H	152
FFA8H	FLW	FLW7	FLW6	FLW5	FLW4	FLW3	FLW2	FLW1	FLW0		–	√	–	00H	153
FFA9H to FFDFH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFE0H	IF0	0	<TMIF 010>	<TMIF 000>	<TMIF H1>	<PIF 1>	<PIF 0>	<LVI IF>	0	R/W	√	√	–	00H	100
FFE1H to FFE3H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFE4H	MK0	1	<TMM K010>	<TMM K000>	<TMM KH1>	<PMK 1>	<PMK 0>	<LVIM K>	1	R/W	√	√	–	FFH	101
FFE5H to FFEBH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFECH	INTM0	0	0	ES11	ES10	ES01	ES00	0	0	R/W	–	√	–	00H	101
FFEDH to FFF2H	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFF3H	PPCC	0	0	0	0	0	0	PPCC 1	PPCC 0	R/W	√	√	–	02H	63
FFF4H to FFFAH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FFFBH	PCC	0	0	0	0	0	0	PCC1	0	R/W	√	√	–	02H	63

Remark For a bit name enclosed in angle brackets (<>), the bit name is defined as a reserved word in the RA78K0S, and is defined as an sfr variable using the #pragma sfr directive in the CC78K0S.

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 Instructions User's Manual (U11047E)**).

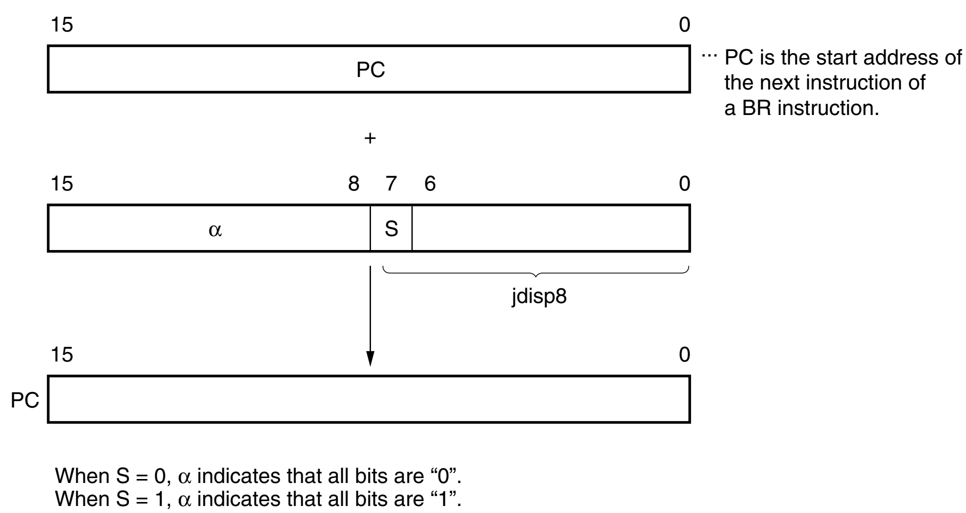
3.3.1 Relative addressing

[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]



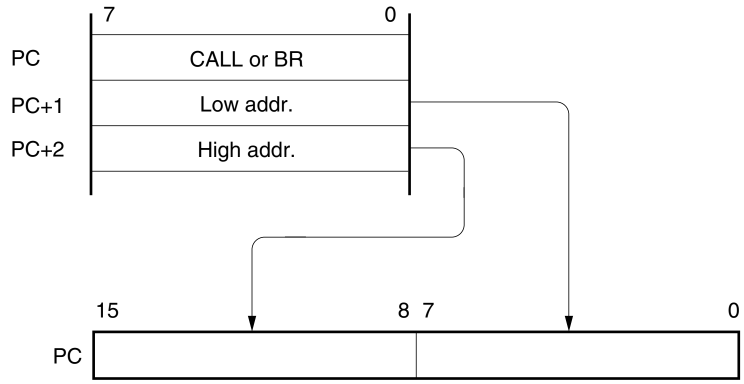
3.3.2 Immediate addressing

[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

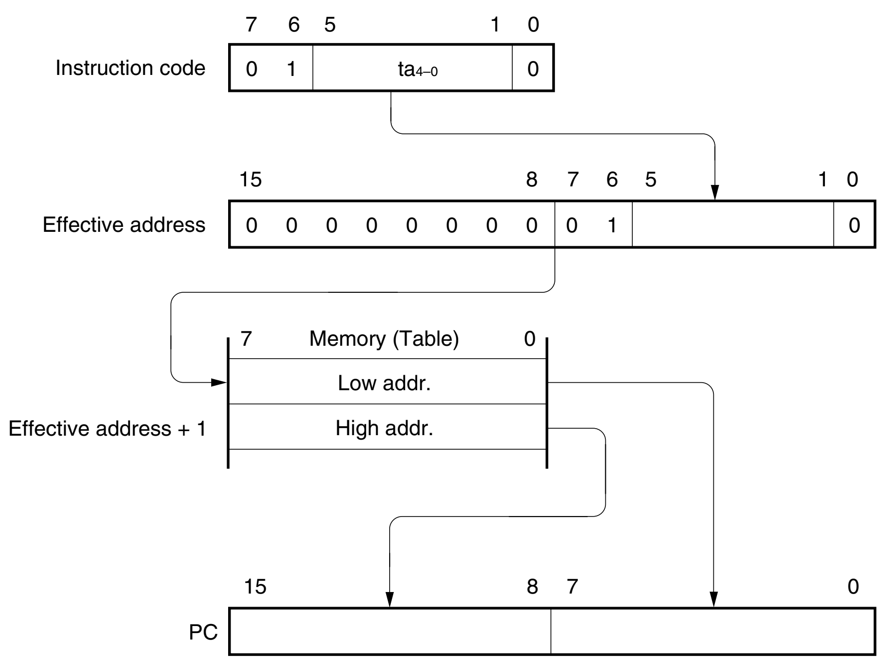


3.3.3 Table indirect addressing

[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]



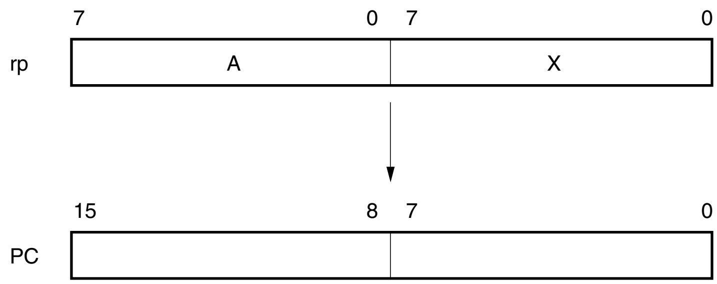
3.3.4 Register addressing

[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]

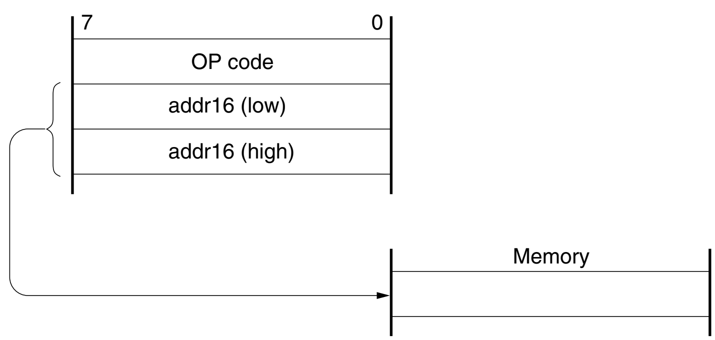
Identifier	Description
addr16	Label or 16-bit immediate data

[Description example]

MOV A, !FE80H; When setting !addr16 to FE80H

Instruction code	0	0	1	0	1	0	0	1	OP Code
	1	0	0	0	0	0	0	0	80H
	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 160-byte space FE80H to FF1FH (FE80H to FEFFH (internal high-speed RAM) + FF00H to FF1FH (special function registers)).

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 80H to FFH, bit 8 of an effective address is cleared to 0. When it is at 00H to 1FH, bit 8 is set to 1. See **[Illustration]** below.

Identifier	Description
saddr	Label or FE80H to FF1FH immediate data
saddrp	Label or FE80H to FF1FH immediate data (even address only)

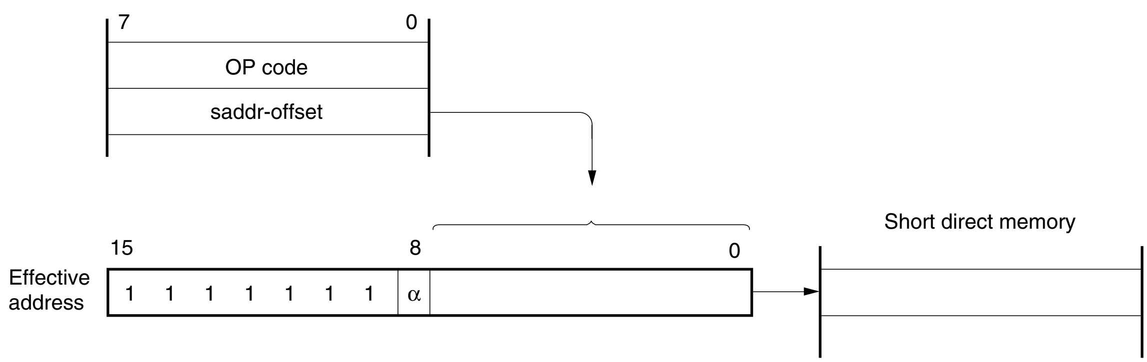
[Description example]

EQU DATA1 0FE90H ; DATA1 shows FE90H of a saddr area,

MOV DATA1, #50H ; When setting 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 space FF00H to FFFFH. However, SFRs mapped at FF00H to FF1FH are accessed with short direct addressing.

[Operand format]

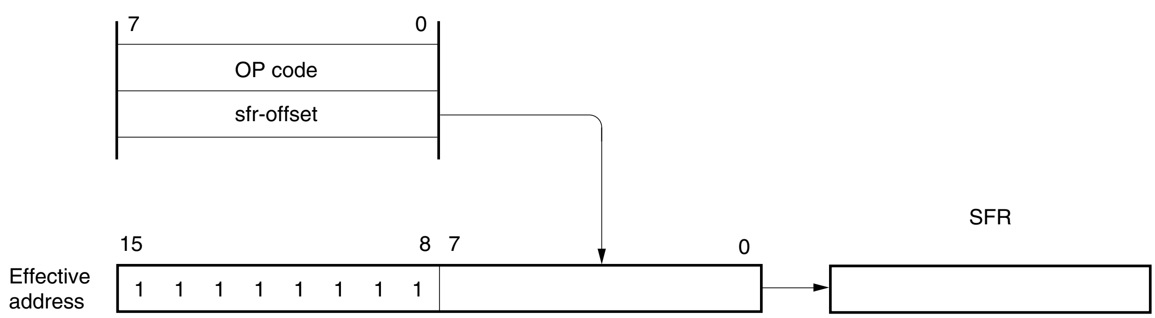
Identifier	Description
sfr	Special function register name

[Description example]

MOV PM0, A; When selecting PM0 for sfr

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

[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.

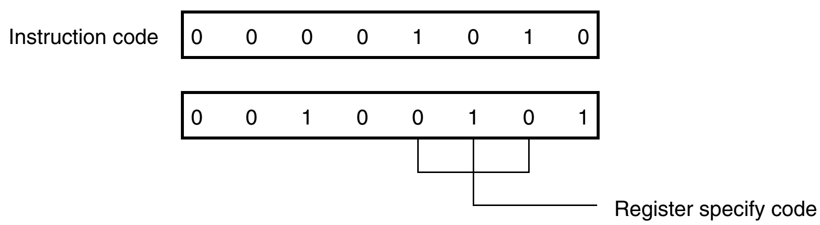
[Operand format]

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

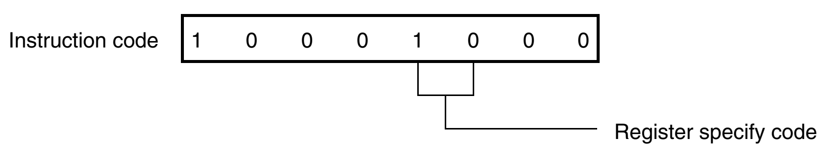
'r' and 'rp' can be described with absolute names (R0 to R7 and RP0 to RP3) as well as function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL).

[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]

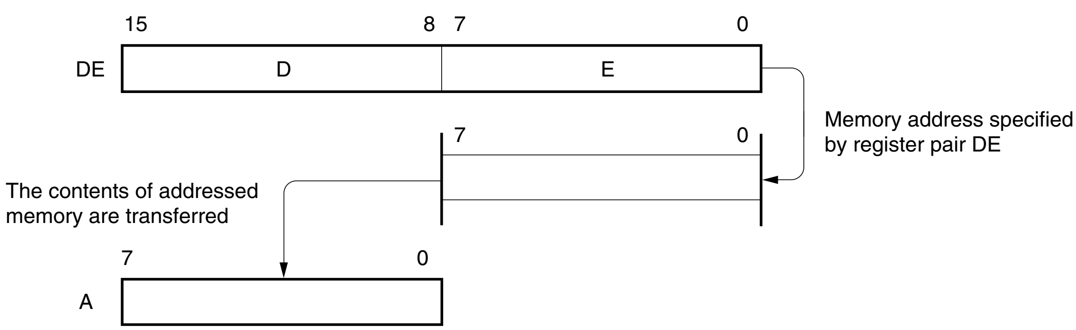
Identifier	Description
–	[DE], [HL]

[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]

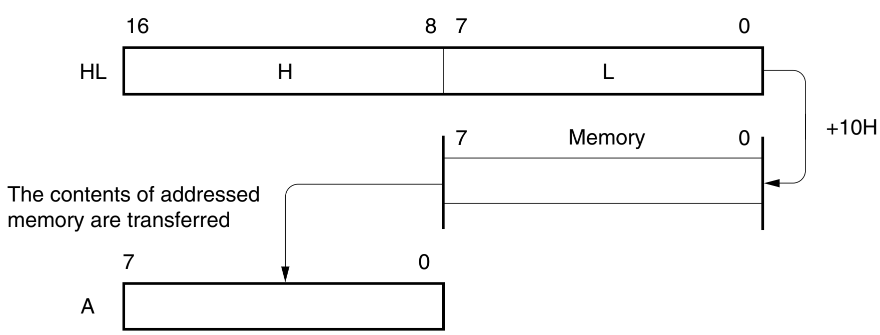
Identifier	Description
–	[HL+byte]

[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

[Illustration]



3.4.7 Stack addressing

[Function]

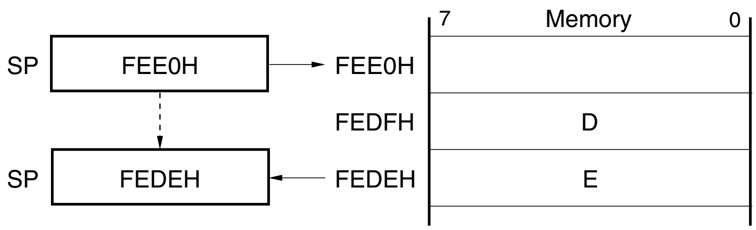
The stack area is indirectly addressed with the stack pointer (SP) contents.
This addressing method is automatically employed when the PUSH, POP, subroutine call, and return instructions are executed or the register is saved/restored upon interrupt request generation.
Stack addressing can be used to 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

[Illustration]



4.1 Functions of Ports

The μPD78F9500, 78F9501, 78F9502 have the ports shown in Figure 4-1, which can be used for various control operations. Table 4-1 shows the functions of each port.

In addition to digital I/O port functions, each of these ports has an alternate function. For details, refer to **CHAPTER 2 PIN FUNCTIONS**.

Figure 4-1. Port Functions

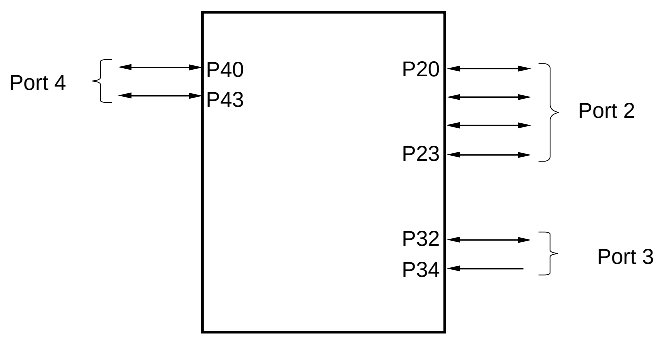


Table 4-1. Port Functions

Pin Name	I/O	Function		After Reset	Alternate-Function Pin
P20	I/O	Port 2. 4-bit I/O port. Can be set to input or output mode in 1-bit units. On-chip pull-up resistor can be connected by setting software.		Input	TOH1
P21					INTP0
P22					-
P23[Note]					EXCLK[Note]
P32	I/O	Port 3 On-chip pull-up resistor can be connected by setting software.	Can be set to input or output mode in 1-bit units.	Input	INTP1
P34[Note]	Input		Input only	Input	$\overline{RESET}$[Note]
P40 and P43	I/O	Port 4. 2-bit I/O port. Can be set to input or output mode in 1-bit units. On-chip pull-up resistor can be connected setting software.		Input	–

Note For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

Caution The P22 and P23/EXCLK pins are pulled down during reset. The P34/$\overline{RESET}$ pin is pulled up during reset by the reset pin function/power-on clear circuit.

Remark P23 can be allocated when the high-speed internal oscillation is selected as the system clock.

4.2 Port Configuration

Ports consist of the following hardware units.

Table 4-2. Configuration of Ports

Item	Configuration
Control registers	Port mode registers (PM2 to PM4) Port registers (P2 to P4) Pull-up resistor option registers (PU2 to PU4)
Ports	Total: 8 (CMOS I/O: 7, CMOS input: 1)
Pull-up resistor	Total: 7

4.2.1 Port 2

Port 2 is a 4-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 2 (PM2). When the P20 to P23 pins are used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 2 (PU2).

This port can also be used for timer I/O, and external interrupt request input.

The P23 pin is also used as the EXCLK pin of the system clock oscillator. The functions of the EXCLK pin differs, therefore, depending on the selected system clock oscillator. The following two system clock oscillators can be used.

(1) High-speed internal oscillator

The P23 pin can be used as I/O port pin.

(2) External clock input

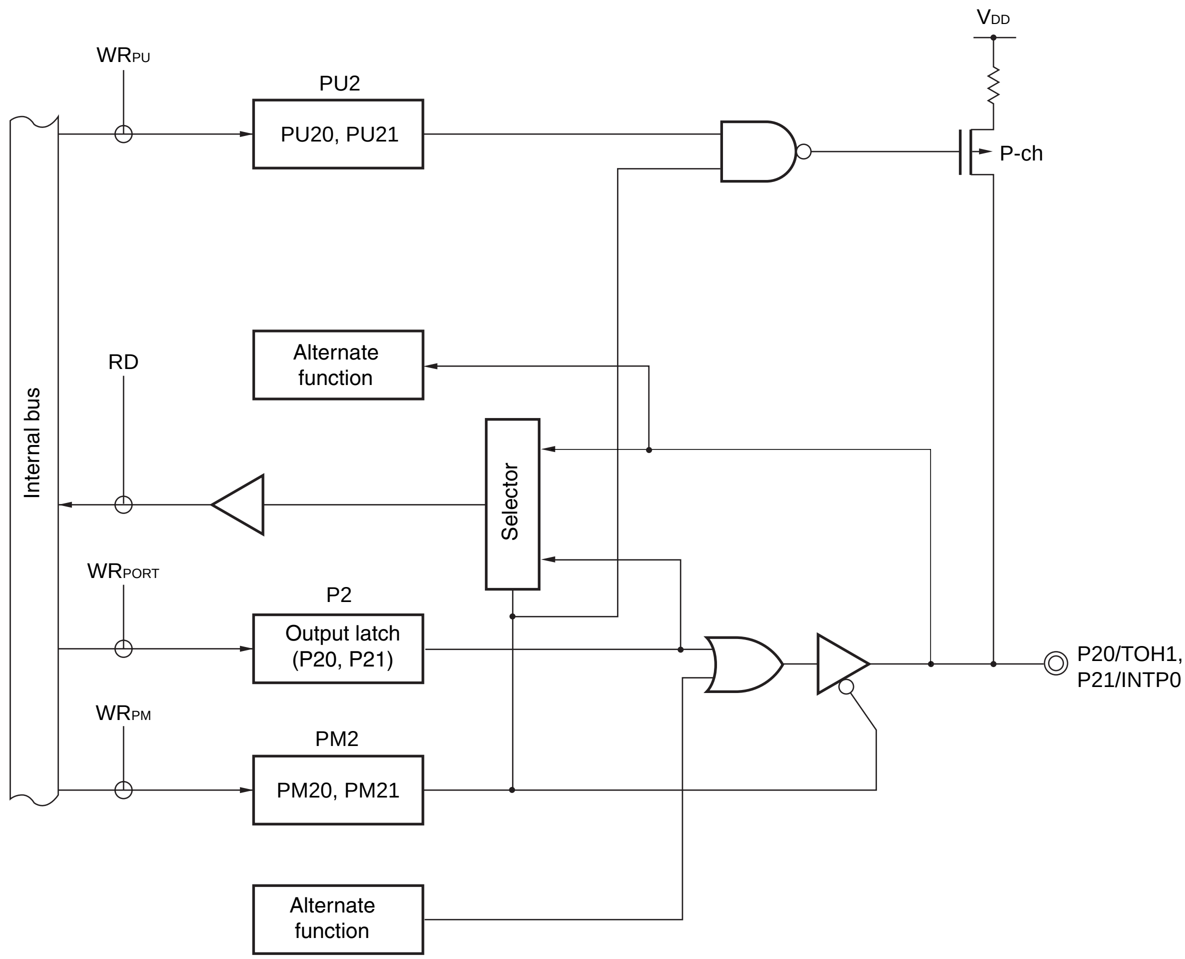
The P23 pin is used as the EXCLK pin to input an external clock, and therefore it cannot be used as an I/O port pin.

The system clock oscillation is selected by the option byte. For details, refer to **CHAPTER 13 OPTION BYTE**.

Reset signal generation sets port 2 to the input mode.

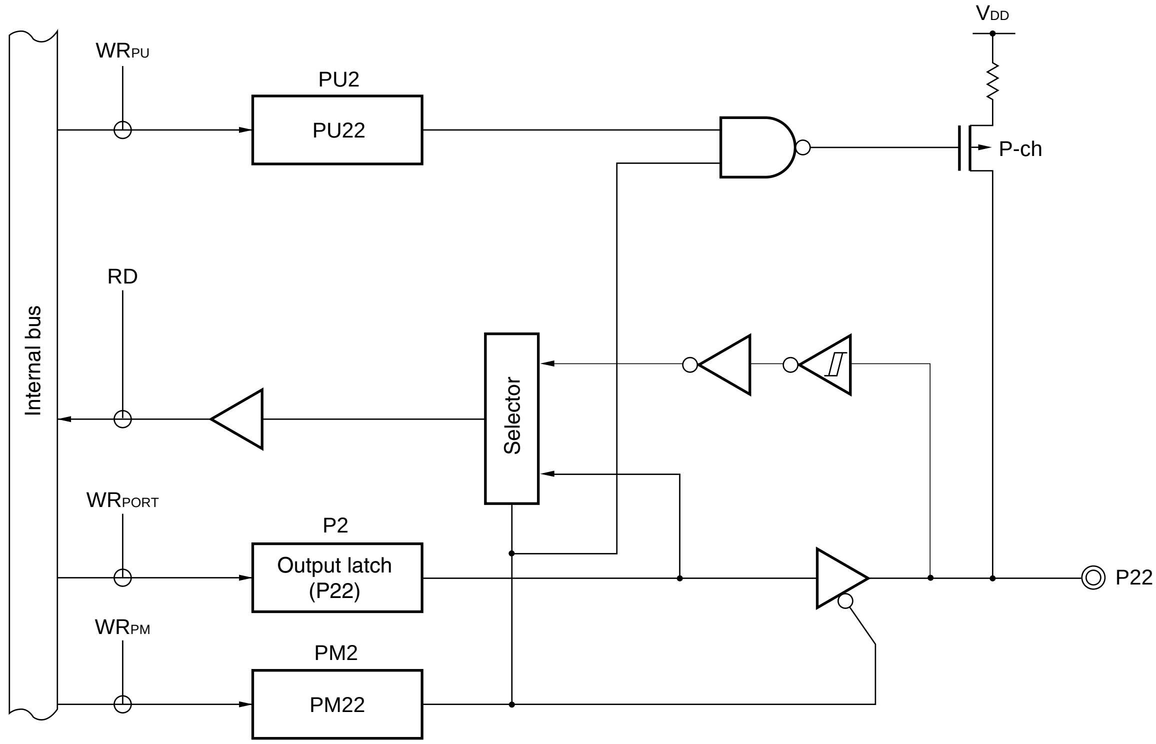
Figures 4-2 to 4-4 show the block diagrams of port 2.

Figure 4-2. Block Diagram of P20 and P21



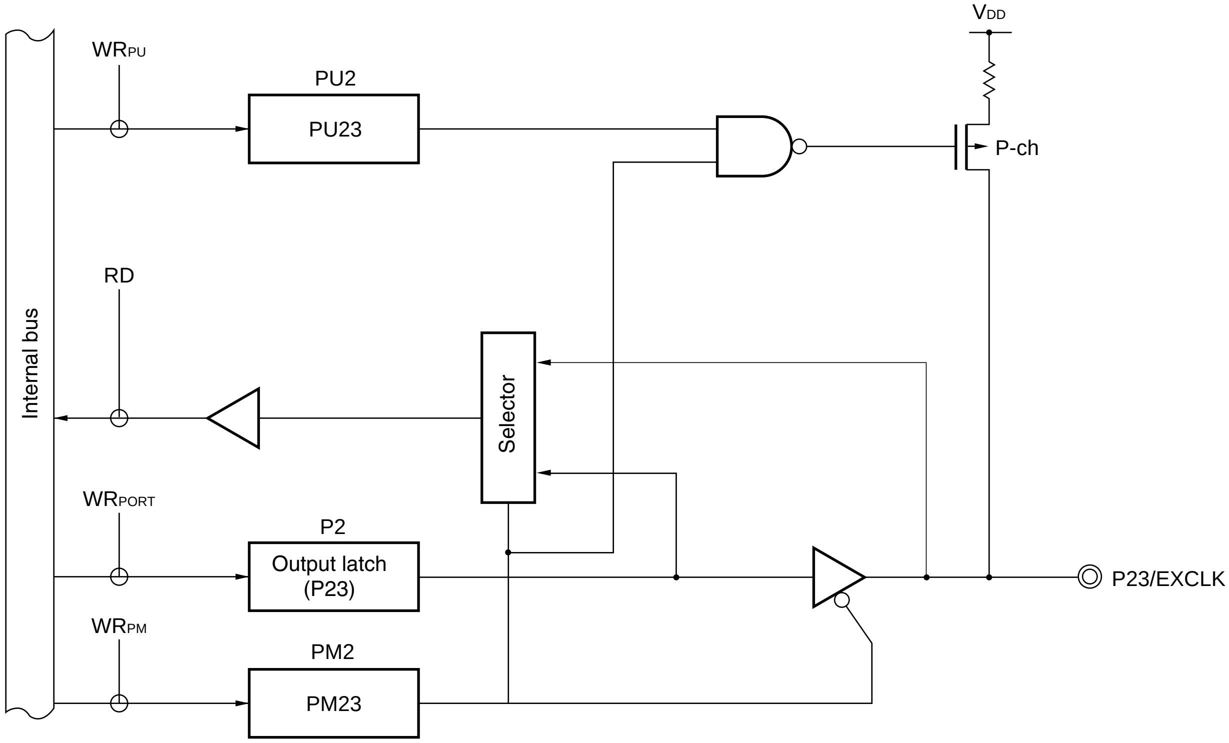
P2: Port register 2
PU2: Pull-up resistor option register 2
PM2: Port mode register 2
RD: Read signal
WR××: Write signal

Figure 4-3. Block Diagram of P22



P2: Port register 2
PU2: Pull-up resistor option register 2
PM2: Port mode register 2
RD: Read signal
WR××: Write signal

Figure 4-4. Block Diagram of P23



P2: Port register 2
PU2: Pull-up resistor option register 2
PM2: Port mode register 2
RD: Read signal
WR××: Write signal

4.2.2 Port 3

The P32 pin is a 1-bit I/O port with an output latch. This pin can be set to the input or output mode by using port mode register 3 (PM3). When this pin is used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 3 (PU3). This pin can also be used for external interrupt request input.

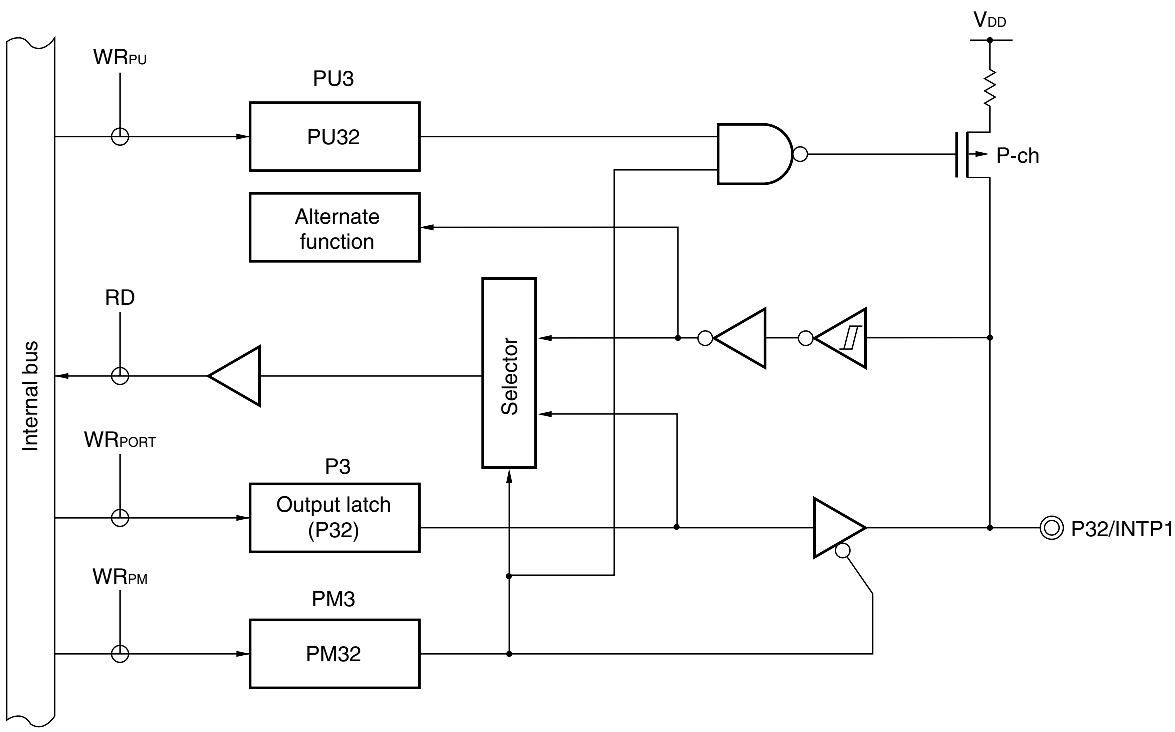
The P32 pin is a Reset signal generation sets port 3 to the input mode.

The P34 pin is a 1-bit input-only port. This pin is also used as a $\overline{\text{RESET}}$ pin, and when the power is turned on, this is the reset function. For the setting method for pin functions, see **CHAPTER 13 OPTION BYTE**.

When P32 and P34 are used as an input port pins, connect the pull-up resistor.

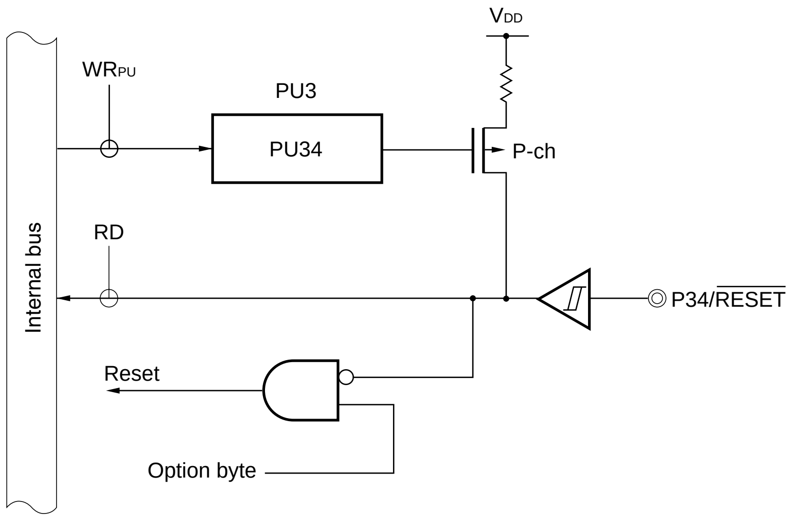
Figures 4-5 and 4-6 show the block diagrams of port 3.

Figure 4-5. Block Diagram of P32



P3: Port register 3
PU3: Pull-up resistor option register 3
PM3: Port mode register 3
RD: Read signal
WR××: Write signal

Figure 4-6. Block Diagram of P34



RD: Read signal

Caution Because the P34 pin functions alternately as the RESET pin, if it is used as an input port pin, the function to input an external reset signal to the RESET pin cannot be used. The function of the port is selected by the option byte. For details, refer to CHAPTER 13 OPTION BYTE.
Also, since the option byte is referenced after the reset release, if low level is input to the RESET pin before the referencing, then the reset state is not released. When it is used as an input port pin, connect the pull-up resistor.

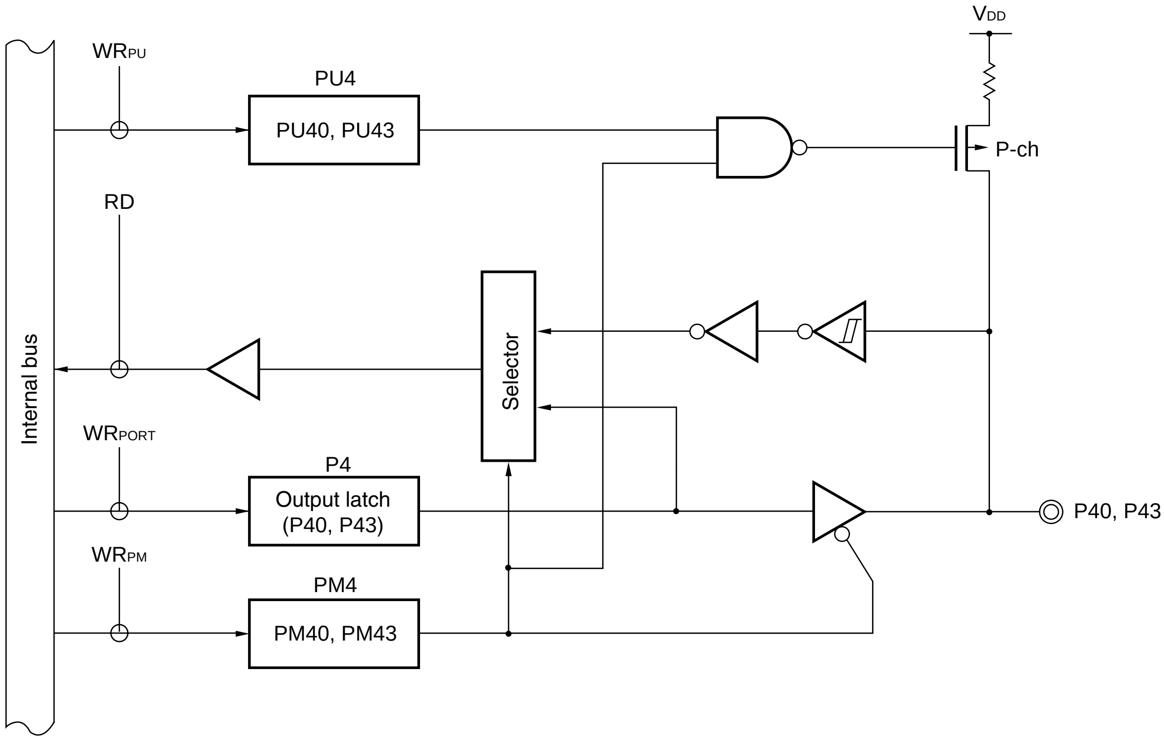
4.2.3 Port 4

Port 4 is a 2-bit I/O port with an output latch. Each bit of this port can be set to the input or output mode by using port mode register 4 (PM4). When the P40 and P43 pins are used as an input port, an on-chip pull-up resistor can be connected in 1-bit units by using pull-up resistor option register 4 (PU4).

Reset signal generation sets port 4 to the input mode.

Figures 4-7 shows the block diagram of port 4.

Figure 4-7. Block Diagram of P40 and P43



P4: Port register 4
PU4: Pull-up resistor option register 4
PM4: Port mode register 4
RD: Read signal
WR××: Write signal

4.3 Registers Controlling Port Functions

The ports are controlled by the following four types of registers.

- Port mode registers (PM2 to PM4)
- Port registers (P2 to P4)
- Pull-up resistor option registers (PU2 to PU4)

(1) Port mode registers (PM2 to PM4)

These registers are used to set the corresponding port to the input or output mode in 1-bit units.

Each port mode register can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation sets these registers to FFH.

When a port pin is used as an alternate-function pin, set its port mode register and output latch as shown in Table 4-3.

Caution Because P21 and P32 are also used as external interrupt pins, the corresponding interrupt request flag is set if each of these pins is set to the output mode and its output level is changed. To use the port pin in the output mode, therefore, set the corresponding interrupt mask flag to 1 in advance.

Figure 4-8. Format of Port Mode Register

Address: FF22H, After reset: FFH, R/W

Symbol	7	6	5	4	3	2	1	0
PM2	1	1	1	1	PM23	PM22	PM21	PM20

Address: FF23H, After reset: FFH, R/W

Symbol	7	6	5	4	3	2	1	0
PM3	1	1	1	1	1	PM32	1	1

Address: FF24H, After reset: FFH, R/W

Symbol	7	6	5	4	3	2	1	0
PM4	1	1	1	1	PM43	1	1	PM40

PMmn	Selection of I/O mode of Pmn pin (m = 2 to 4; n = 0 to 3)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

(2) Port registers (P2 to P4)

These registers are used to write data to be output from the corresponding port pin to an external device connected to the chip.

When a port register is read, the pin level is read in the input mode, and the value of the output latch of the port is read in the output mode.

P20 to P23, P32, P40 and P43 are set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation sets these registers to 00H.

Figure 4-9. Format of Port Register

Address: FF02H, After reset: 00H (Output latch) R/W

Symbol	7	6	5	4	3	2	1	0
P2	0	0	0	0	P23	P22	P21	P20

Address: FF03H, After reset: 00H[Note] (Output latch) R/W[Note]

Symbol	7	6	5	4	3	2	1	0
P3	0	0	0	P34	0	P32	0	0

Address: FF04H, After reset: 00H (Output latch) R/W

Symbol	7	6	5	4	3	2	1	0
P4	0	0	0	0	P43	0	0	P40

Pmn	m = 2 to 4; n = 0 to 4	
	Controls of output data (in output mode)	Input data read (in input mode)
0	Output 0	Input low level
1	Output 1	Input high level

Note Because P34 is read-only, its reset value is undefined.

(3) Pull-up resistor option registers (PU2 to PU4)

These registers are used to specify whether an on-chip pull-up resistor is connected to P20 to P23, P32, P34, P40 and P43. By setting PU2 to PU4, an on-chip pull-up resistor can be connected to the port pin corresponding to the bit of PU2 to PU4.

PU2 to PU4 are set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation set these registers to 00H.

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

Address: FF32H, After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PU2	0	0	0	0	PU23	PU22	PU21	PU20

Address: FF33H, After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PU3	0	0	0	PU34	0	PU32	0	0

Address: FF34H, After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PU4	0	0	0	0	PU43	0	0	PU40

PUmn	Selection of connection of on-chip pull-up resistor of Pmn (m = 2 to 4; n = 0 to 4)
0	Does not connect on-chip pull-up resistor
1	Connects on-chip pull-up resistor

4.4 Operation of Port Function

The operation of a port differs, as follows, depending on the setting of the I/O mode.

Caution Although a 1-bit memory manipulation instruction manipulates 1 bit, it accesses a port in 8-bit units. Therefore, the contents of the output latch of a pin in the input mode, even if it is not subject to manipulation by the instruction, are undefined in a port with a mixture of inputs and outputs.

4.4.1 Writing to I/O port

(1) In output mode

A value can be written to the output latch by a transfer instruction. In addition, the contents of the output latch are output from the pin. Once data is written to the output latch, it is retained until new data is written to the output latch.

When a reset signal is generated, cleans the data in the output latch.

(2) In input mode

A value can be written to the output latch by a transfer instruction. Because the output buffer is off, however, the pin status remains unchanged.

Once data is written to the output latch, it is retained until new data is written to the output latch.

When a reset signal is generated, cleans the data in the output latch.

4.4.2 Reading from I/O port

(1) In output mode

The contents of the output latch can be read by a transfer instruction. The contents of the output latch remain unchanged.

(2) In input mode

The pin status can be read by a transfer instruction. The contents of the output latch remain unchanged.

4.4.3 Operations on I/O port

(1) In output mode

An operation is performed on the contents of the output latch and the result is written to the output latch. The contents of the output latch are output from the pin.

Once data is written to the output latch, it is retained until new data is written to the output latch.

Reset signal generation clears the data in the output latch.

(2) In input mode

The pin level is read and an operation is performed on its contents. The operation result is written to the output latch. However, the pin status remains unchanged because the output buffer is off.

When a reset signal is generated, cleans the data in the output latch.

CHAPTER 5 CLOCK GENERATORS

5.1 Functions of Clock Generators

The clock generators include a circuit that generates a clock (system clock) to be supplied to the CPU and peripheral hardware, and a circuit that generates a clock (interval time generation clock) to be supplied to the watchdog timer and 8-bit timer H1 (TMH1).

5.1.1 System clock oscillators

The following three types of system clock oscillators are used.

- High-speed internal oscillator
  This circuit internally oscillates a clock of 8 MHz (TYP.). Its oscillation can be stopped by execution of the STOP instruction.
  If the High-speed internal oscillator is selected to supply the system clock, the EXCLK pin can be used as I/O port pins.

- External clock input circuit
  This circuit supplies a clock from an external IC to the EXCLK pin. A clock of 1 MHz to 10 MHz can be supplied. Internal clock supply can be stopped by execution of the STOP instruction.

The system clock source is selected by using the option byte. For details, refer to **CHAPTER 13 OPTION BYTE**.
When using the EXCLK pin as I/O port pins, refer to **CHAPTER 4 PORT FUNCTIONS** for details.

5.1.2 Clock oscillator for interval time generation

The following circuit is used as a clock oscillator for interval time generation.

- Low-speed internal oscillator
  This circuit oscillates a clock of 240 kHz (TYP.). Its oscillation can be stopped by using the low-speed internal oscillation mode register (LSRCM) when it is specified by the option byte that its oscillation can be stopped by software.

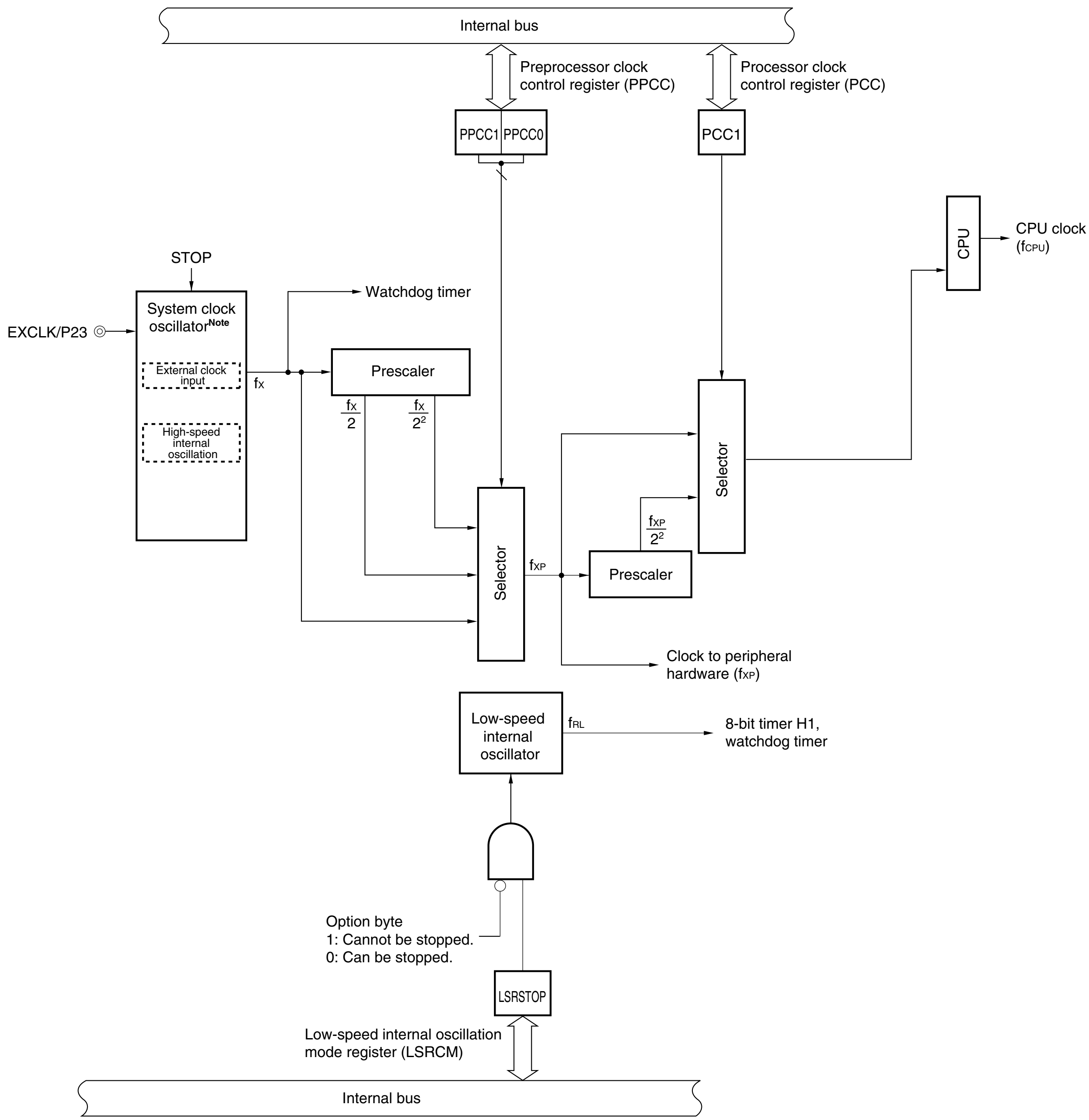
5.2 Configuration of Clock Generators

The clock generators consist of the following hardware.

Table 5-1. Configuration of Clock Generators

Item	Configuration
Control registers	Processor clock control register (PCC) Preprocessor clock control register (PPCC) Low-speed internal oscillation mode register (LSRCM)
Oscillators	High-speed internal oscillator External clock input circuit Low-speed internal oscillator

Figure 5-1. Block Diagram of Clock Generators



Note Select the high-speed internal oscillator or external clock input circuit as the system clock source by using the option byte.

5.3 Registers Controlling Clock Generators

The clock generators are controlled by the following three registers.

- Processor clock control register (PCC)
- Preprocessor clock control register (PPCC)
- Low-speed internal oscillation mode register (LSRCM)

(1) Processor clock control register (PCC) and preprocessor clock control register (PPCC)

These registers are used to specify the division ratio of the system clock.

PCC and PPCC are set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation sets PCC and PPCC to 02H.

Figure 5-2. Format of Processor Clock Control Register (PCC)

Address: FFFBH, After reset: 02H, R/W

Symbol	7	6	5	4	3	2	1	0
PCC	0	0	0	0	0	0	PCC1	0

Figure 5-3. Format of Preprocessor Clock Control Register (PPCC)

Address: FFF3H, After reset: 02H, R/W

Symbol	7	6	5	4	3	2	1	0
PPCC	0	0	0	0	0	0	PPCC1	PPCC0

PPCC1	PPCC0	PCC1	Selection of CPU clock (f_{CPU}) Note 1
0	0	0	f_X
0	1	0	$f_X/2$ Note 2
0	0	1	$f_X/2^2$
1	0	0	$f_X/2^2$ Note 3
0	1	1	$f_X/2^3$ Note 2
1	0	1	$f_X/2^4$ Note 3
Other than above			Setting prohibited

Notes 1. The setting range of the CPU clock differs depending on the supply voltage to be used. Be sure to refer to CPU clock and peripheral clock frequencies described in **AC Characteristics** in **CHAPTER 16 ELECTRICAL SPECIFICATIONS (TARGET)**.

2. If PPCC = 01H, the clock (f_{XP}) supplied to the peripheral hardware is $f_X/2$.

3. If PPCC = 02H, the clock (f_{XP}) supplied to the peripheral hardware is $f_X/2^2$.

The fastest instruction of the μPD78F9500, 78F9501, 78F9502 is executed in two CPU clocks. Therefore, the relationship between the CPU clock (f_{CPU}) and the minimum instruction execution time is as shown in Table 5-2.

Table 5-2. Relationship between CPU Clock and Minimum Instruction Execution Time

CPU Clock (f_{CPU}) Note	Minimum Instruction Execution Time: 2/f_{CPU}	
	High-speed internal oscillation clock (at 8.0 MHz (TYP.))	External clock input (at 10.0 MHz)
f_X	0.25 μs	0.2 μs
$f_X/2$	0.5 μs	0.4 μs
$f_X/2^2$	1.0 μs	0.8 μs
$f_X/2^3$	2.0 μs	1.6 μs
$f_X/2^4$	4.0 μs	3.2 μs

Note The CPU clock (high-speed internal oscillation clock, or external clock input) is selected by the option byte.

(2) Low-speed internal oscillation mode register (LSRCM)

This register is used to select the operation mode of the low-speed internal oscillator (240 kHz (TYP.)).

This register is valid when it is specified by the option byte that the low-speed internal oscillator can be stopped by software. If it is specified by the option byte that the low-speed internal oscillator cannot be stopped by software, setting of this register is invalid, and the low-speed internal oscillator continues oscillating. In addition, the source clock of WDT is fixed to the low-speed internal oscillator. For details, refer to **CHAPTER 7 WATCHDOG TIMER**.

LSRCM can be set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation sets LSRCM to 00H.

Figure 5-4. Format of Low-Speed internal oscillation Mode Register (LSRCM)

Address: FF58H, After reset: 00H, R/W

Symbol	7	6	5	4	3	2	1	<0>
LSRCM	0	0	0	0	0	0	0	LSRSTOP

LSRSTOP	Oscillation/stop of low-speed internal oscillator
0	Low-speed internal oscillates
1	Low-speed internal oscillator stops

5.4 System Clock Oscillators

The following three types of system clock oscillators are available.

- High-speed internal oscillator: Internally oscillates a clock of 8 MHz (TYP.).
- External clock input circuit: Supplies a clock of 1 MHz to 10 MHz to the EXCLK pin.

5.4.1 High-speed internal oscillator

The μPD78F9500, 78F9501, 78F9502 include a high-speed internal oscillator (8 MHz (TYP.)).

If the high-speed internal oscillation is selected by the option byte as the clock source, the EXCLK pin can be used as an I/O port pin.

For details of the option byte, refer to **CHAPTER 13 OPTION BYTE**. For details of I/O ports, refer to **CHAPTER 4 PORT FUNCTIONS**.

5.4.2 External clock input circuit

This circuit supplies a clock from an external IC to the EXCLK pin.

5.4.3 Prescaler

The prescaler divides the clock (f_X) output by the system clock oscillator to generate a clock (f_{XP}) to be supplied to the peripheral hardware. It also divides the clock to peripheral hardware (f_{XP}) to generate a clock to be supplied to the CPU.

Remark The clock output by the oscillator selected by the option byte (high-speed internal oscillator, or external clock input circuit) is divided. For details of the option byte, refer to **CHAPTER 13 OPTION BYTE**.

5.5 Operation of CPU Clock Generator

A clock (f_{CPU}) is supplied to the CPU from the system clock (f_X) oscillated by one of the following three types of oscillators.

- High-speed internal oscillator: Internally oscillates a clock of 8 MHz (TYP.).
- External clock input circuit: Supplies a clock of 1 MHz to 10 MHz to EXCLK pin.

The system clock oscillator is selected by the option byte. For details of the option byte, refer to **CHAPTER 13 OPTION BYTE**.

(1) High-speed internal oscillator

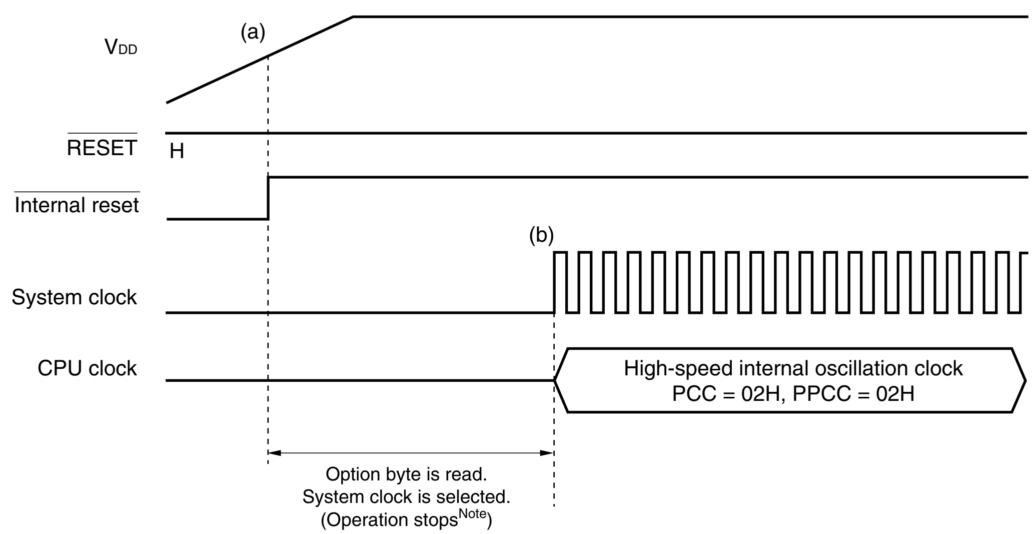
When the high-speed internal oscillation is selected by the option byte, the following is possible.

- Shortening of start time
  If the high-speed internal oscillator is selected as the oscillator, the CPU can be started without having to wait for the oscillation stabilization time of the system clock. Therefore, the start time can be shortened.
- Improvement of expandability
  If the high-speed internal oscillator is selected as the oscillator, the EXCLK pin can be used as I/O port pins. For details, refer to **CHAPTER 4 PORT FUNCTIONS**.

Figures 5-6 and 5-7 show the timing chart and status transition diagram of the default start by the high-speed internal oscillation.

Remark When the high-speed internal oscillation is used, the clock accuracy is ±5%.

Figure 5-6. Timing Chart of Default Start by High-Speed Internal Oscillation

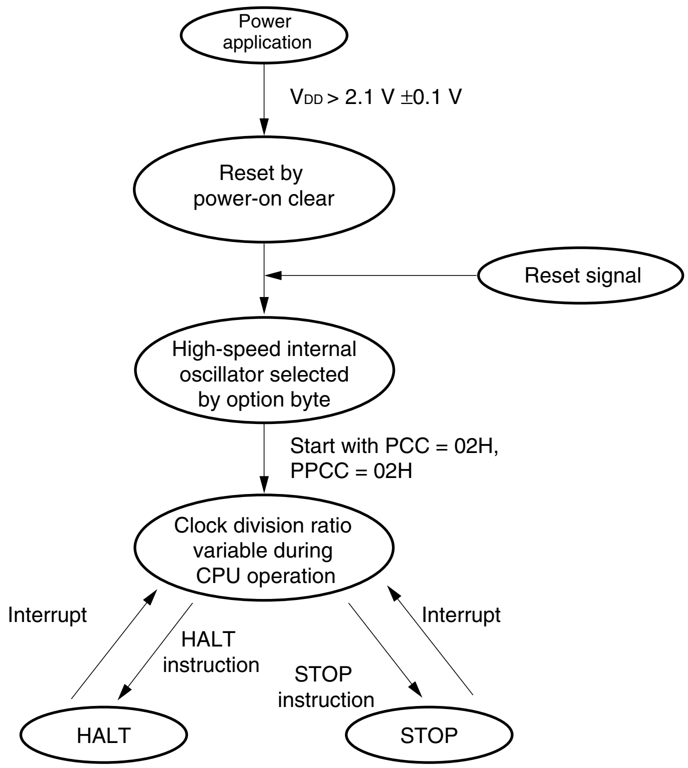


Note Operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

(a) The internal reset signal is generated by the power-on clear function on power application, the option byte is referenced after reset, and the system clock is selected.

(b) The option byte is referenced and the system clock is selected. Then the high-speed internal oscillation clock operates as the system clock.

Figure 5-7. Status Transition of Default Start by High-Speed internal oscillation



Remark PCC: Processor clock control register
PPCC: Preprocessor clock control register

(2) External clock input circuit

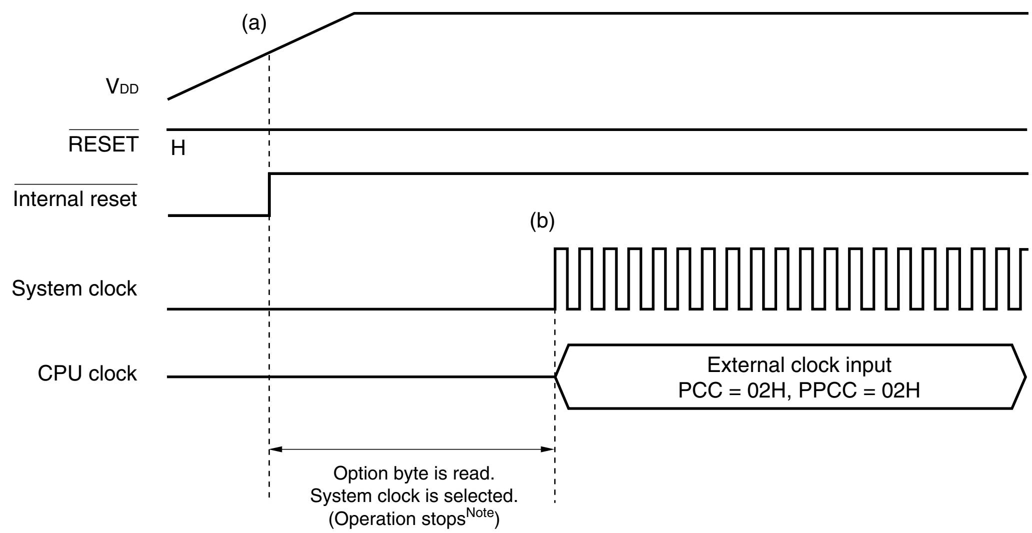
If external clock input is selected by the option byte, the following is possible.

- High-speed operation

  The accuracy of processing is improved as compared with high-speed internal oscillation (8 MHz (TYP.)) because an oscillation frequency of 1 MHz to 10 MHz can be selected and an external clock with a small frequency deviation can be supplied.

Figures 5-8 and 5-9 show the timing chart and status transition diagram of default start by external clock input.

Figure 5-8. Timing of Default Start by External Clock Input

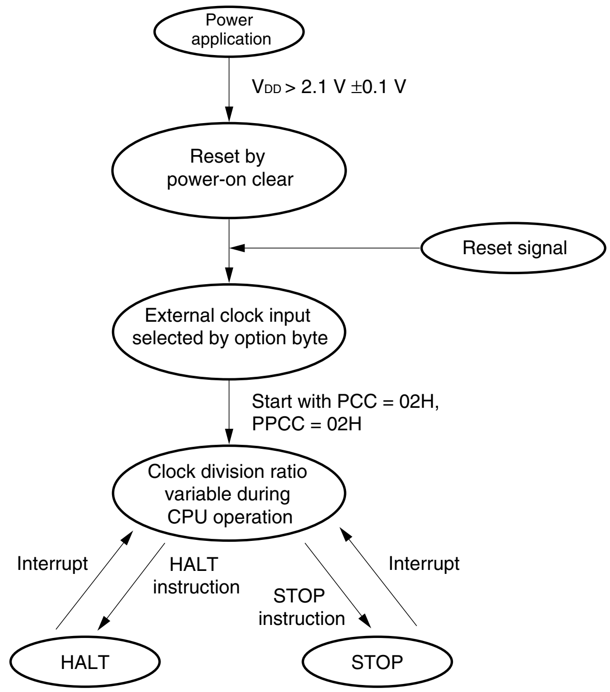


Note Operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

(a) The internal reset signal is generated by the power-on clear function on power application, the option byte is referenced after reset, and the system clock is selected.

(b) The option byte is referenced and the system clock is selected. Then the external clock operates as the system clock.

Figure 5-9. Status Transition of Default Start by External Clock Input



Remark PCC: Processor clock control register
PPCC: Preprocessor clock control register

5.6 Operation of Clock Generator Supplying Clock to Peripheral Hardware

The following two types of clocks are supplied to the peripheral hardware.

- Clock to peripheral hardware (f_{XP})
- Low-speed internal oscillation clock (f_{RL})

(1) Clock to peripheral hardware

The clock to the peripheral hardware is supplied by dividing the system clock (f_X). The division ratio is selected by the pre-processor clock control register (PPCC).

Three types of frequencies are selectable: "f_X", "$f_X/2$", and "$f_X/2^2$". Table 5-3 lists the clocks supplied to the peripheral hardware.

Table 5-3. Clocks to Peripheral Hardware

PPCC1	PPCC0	Selection of clock to peripheral hardware (f_{XP})
0	0	f_X
0	1	$f_X/2$
1	0	$f_X/2^2$
1	1	Setting prohibited

(2) Low-speed internal oscillation clock

The low-speed internal oscillator of the clock oscillator for interval time generation is always started after release of reset, and oscillates at 240 kHz (TYP.).

It can be specified by the option byte whether the low-speed internal oscillator can or cannot be stopped by software. If it is specified that the low-speed internal oscillator can be stopped by software, oscillation can be started or stopped by using the low-speed internal oscillation mode register (LSRCM). If it is specified that it cannot be stopped by software, the clock source of WDT is fixed to the low-speed internal oscillation clock (f_{RL}).

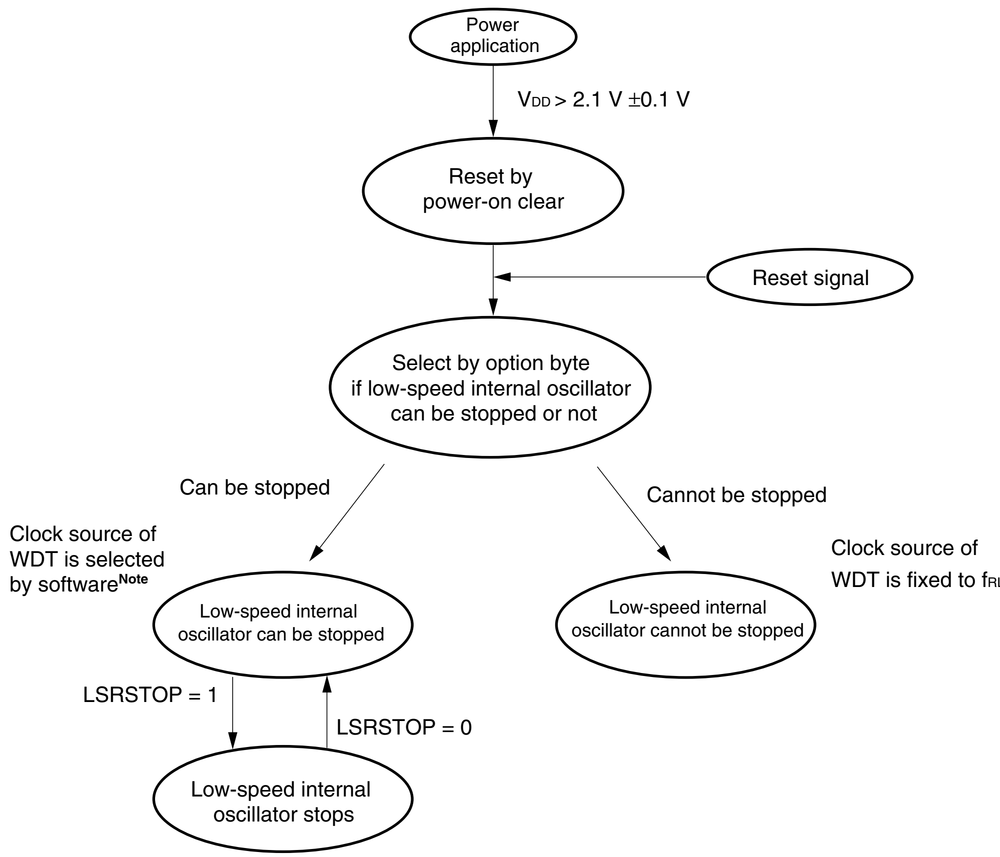
The low-speed internal oscillator is independent of the CPU clock. If it is used as the source clock of WDT, therefore, a hang-up can be detected even if the CPU clock is stopped. If the low-speed internal oscillator is used as a count clock source of 8-bit timer H1, 8-bit timer H1 can operate even in the standby status.

Table 5-4 shows the operation status of the low-speed internal oscillator when it is selected as the source clock of WDT and the count clock of 8-bit timer H1. Figure 5-10 shows the status transition of the low-speed internal oscillator.

Table 5-4. Operation Status of Low-Speed Internal Oscillator

Option Byte Setting		CPU Status	WDT Status	TMH1 Status
Can be stopped by software	LSRSTOP = 1	Operation mode	Stopped	Stopped
	LSRSTOP = 0		Operates	Operates
	LSRSTOP = 1	Standby	Stopped	Stopped
	LSRSTOP = 0		Stopped	Operates
Cannot be stopped		Operation mode	Operates	
		Standby		

Figure 5-10. Status Transition of Low-Speed Internal Oscillator



Note The clock source of the watchdog timer (WDT) is selected from f_X or f_{RL}, or it may be stopped. For details, refer to **CHAPTER 7 WATCHDOG TIMER**.

6.1 Functions of 8-Bit Timer H1

8-bit timer H1 has the following functions.

- Interval timer
- PWM output mode
- Square-wave output

6.2 Configuration of 8-Bit Timer H1

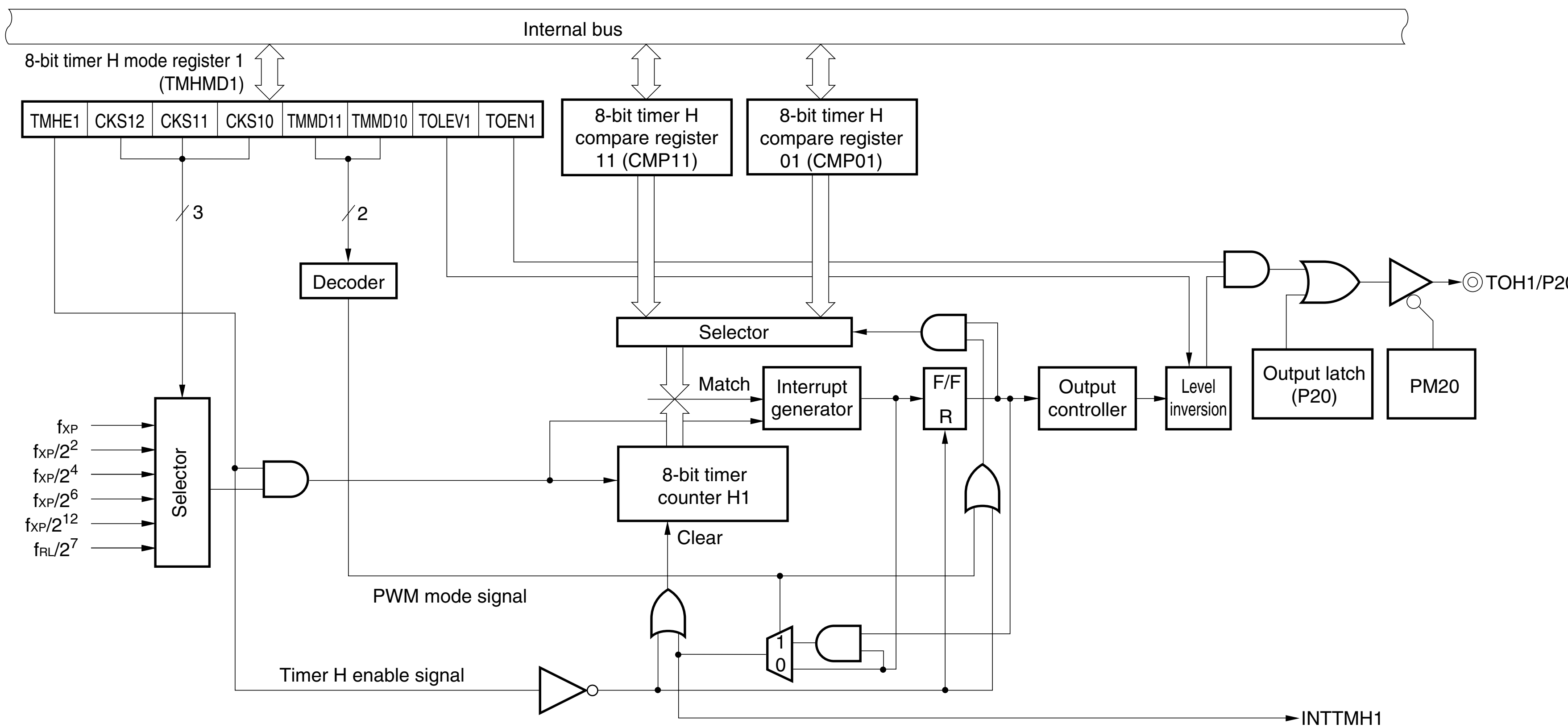
8-bit timer H1 consists of the following hardware.

Table 6-1. Configuration of 8-Bit Timer H1

Item	Configuration
Timer register	8-bit timer counter H1
Registers	8-bit timer H compare register 01 (CMP01) 8-bit timer H compare register 11 (CMP11)
Timer output	TOH1
Control registers	8-bit timer H mode register 1 (TMHMD1) Port mode register 2 (PM2) Port register 2 (P2)

Figure 6-1 shows a block diagram.

Figure 6-1. Block Diagram of 8-Bit Timer H1



(1) 8-bit timer H compare register 01 (CMP01)

This register can be read or written by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

Figure 6-2. Format of 8-Bit Timer H Compare Register 01 (CMP01)

Address: FF0EH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CMP01								

Caution CMP01 cannot be rewritten during timer count operation.

(2) 8-bit timer H compare register 11 (CMP11)

This register can be read or written by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

Figure 6-3. Format of 8-Bit Timer H Compare Register 11 (CMP11)

Address: FF0FH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CMP11								

CMP11 can be rewritten during timer count operation.

If the CMP11 value is rewritten during timer operation, the compare value after the rewrite takes effect at the timing at which the count value and the compare value before the rewrite match. If the timing at which the count value and compare value match conflicts with the timing of the writing from the CPU to CMP11, the compare value after the rewrite takes effect at the timing at which the next count value and the compare value before the rewrite match.

Caution In the PWM output mode, be sure to set CMP11 when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to CMP11).

6.3 Registers Controlling 8-Bit Timer H1

The following three registers are used to control 8-Bit Timer H1.

- 8-bit timer H mode register 1 (TMHMD1)
- Port mode register 2 (PM2)
- Port register 2 (P2)

(1) 8-bit timer H mode register 1 (TMHMD1)

This register controls the mode of timer H.

This register can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

Figure 6-4. Format of 8-Bit Timer H Mode Register 1 (TMHMD1)

Address: FF70H After reset: 00H R/W

Symbol	<7>	6	5	4	3	2	<1>	<0>
TMHMD1	TMHE1	CKS12	CKS11	CKS10	TMMD11	TMMD10	TOLEV1	TOEN1

TMHE1	Timer operation enable
0	Stop timer count operation (counter is cleared to 0)
1	Enable timer count operation (count operation started by inputting clock)

CKS12	CKS11	CKS10	Count clock (f_{CNT}) selection
0	0	0	f_{XP} (10 MHz)
0	0	1	$f_{XP}/2^2$ (2.5 MHz)
0	1	0	$f_{XP}/2^4$ (625 kHz)
0	1	1	$f_{XP}/2^6$ (156.25 kHz)
1	0	0	$f_{XP}/2^{12}$ (2.44 kHz)
1	0	1	$f_{RL}/2^7$ (1.88 kHz (TYP.))
Other than above			Setting prohibited

TMMD11	TMMD10	Timer operation mode
0	0	Interval timer mode
1	0	PWM output mode
Other than above		Setting prohibited

TOLEV1	Timer output level control (in default mode)
0	Low level
1	High level

TOEN1	Timer output control
0	Disable output
1	Enable output

Cautions 1. When TMHE1 = 1, setting the other bits of the TMHMD1 register is prohibited.

2. In the PWM output mode, be sure to set 8-bit timer H compare register 11 (CMP11) when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).

Remarks 1. f_{XP}: Oscillation frequency of clock to peripheral hardware

2. f_{RL}: Low-speed internal oscillation clock oscillation frequency

3. Figures in parentheses apply to operation at f_{XP} = 10 MHz, f_{RL} = 240 kHz (TYP.).

(2) Port mode register 2 (PM2)

When using the P20/TOH1 pin for timer output, clear PM20, the output latch of P20 to 0.
PM2 can be set by a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation sets PM2 to FFH.

Figure 6-5. Format of Port Mode Register 2 (PM2)

Address: FF22H After reset: FFH R/W

Symbol	7	6	5	4	3	2	1	0
PM2	1	1	1	1	PM23	PM22	PM21	PM20

PM2n	P2n pin I/O mode selection (n = 0 to 3)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

6.4 Operation of 8-Bit Timer H1

6.4.1 Operation as interval timer/square-wave output

When 8-bit timer counter H1 and compare register 01 (CMP01) match, an interrupt request signal (INTTMH1) is generated and 8-bit timer counter H1 is cleared to 00H.

Compare register 11 (CMP11) is not used in interval timer mode. Since a match of 8-bit timer counter H1 and the CMP11 register is not detected even if the CMP11 register is set, timer output is not affected.

By setting bit 0 (TOEN1) of timer H mode register 1 (TMHMD1) to 1, a square wave of any frequency (duty = 50%) is output from TOH1.

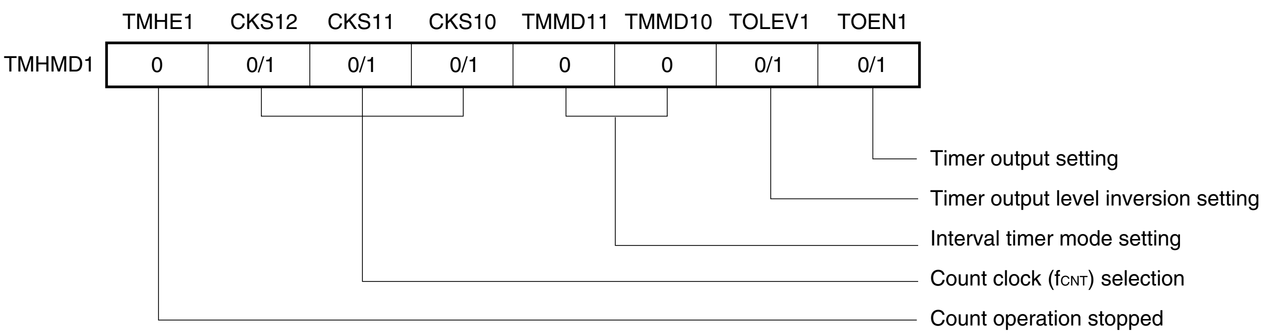
(1) Usage

Generates the INTTMH1 signal repeatedly at the same interval.

<1> Set each register.

Figure 6-6. Register Setting During Interval Timer/Square-Wave Output Operation

(i) Setting timer H mode register 1 (TMHMD1)



(ii) CMP01 register setting

- Compare value (N)

<2> Count operation starts when TMHE1 = 1.

<3> When the values of 8-bit timer counter H1 and the CMP01 register match, the INTTMH1 signal is generated and 8-bit timer counter H1 is cleared to 00H.

$$\text{Interval time} = (N+1)/f_{CNT}$$

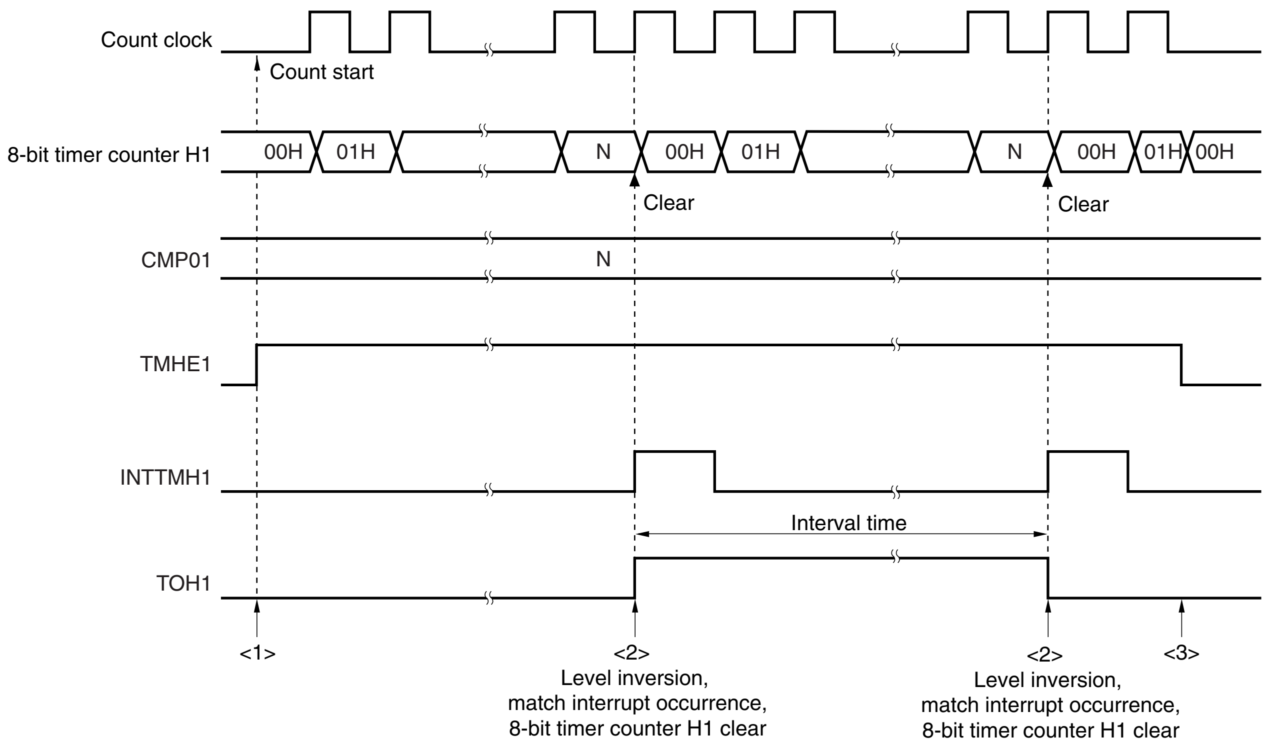
<4> Subsequently, the INTTMH1 signal is generated at the same interval. To stop the count operation, clear TMHE1 to 0.

(2) Timing chart

The timing of the interval timer/square-wave output operation is shown below.

Figure 6-7. Timing of Interval Timer/Square-Wave Output Operation (1/2)

(a) Basic operation (01H ≤ CMP01 ≤ FEH)



<1> The count operation is enabled by setting the TMHE1 bit to 1. The count clock starts counting no more than 1 clock after the operation is enabled.

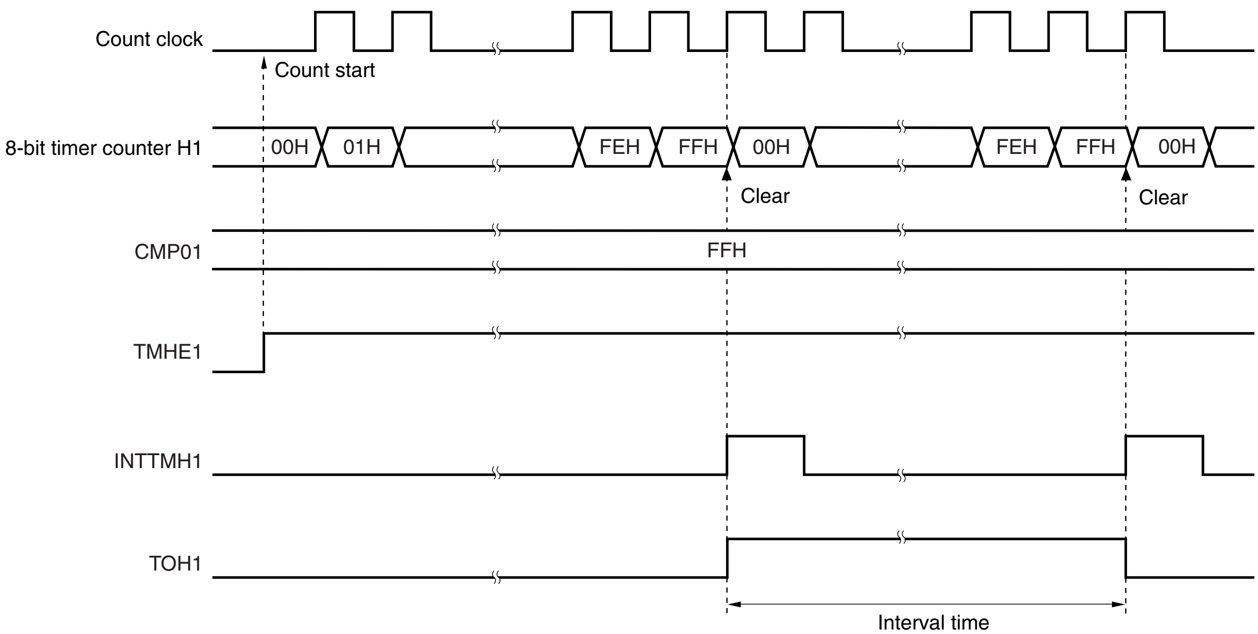
<2> When the values of 8-bit timer counter H1 and the CMP01 register match, the value of 8-bit timer counter H1 is cleared, the TOH1 output level is inverted, and the INTTMH1 signal is output.

<3> The INTTMH1 signal and TOH1 output become inactive by clearing the TMHE1 bit to 0 during timer H1 operation. If these are inactive from the first, the level is retained.

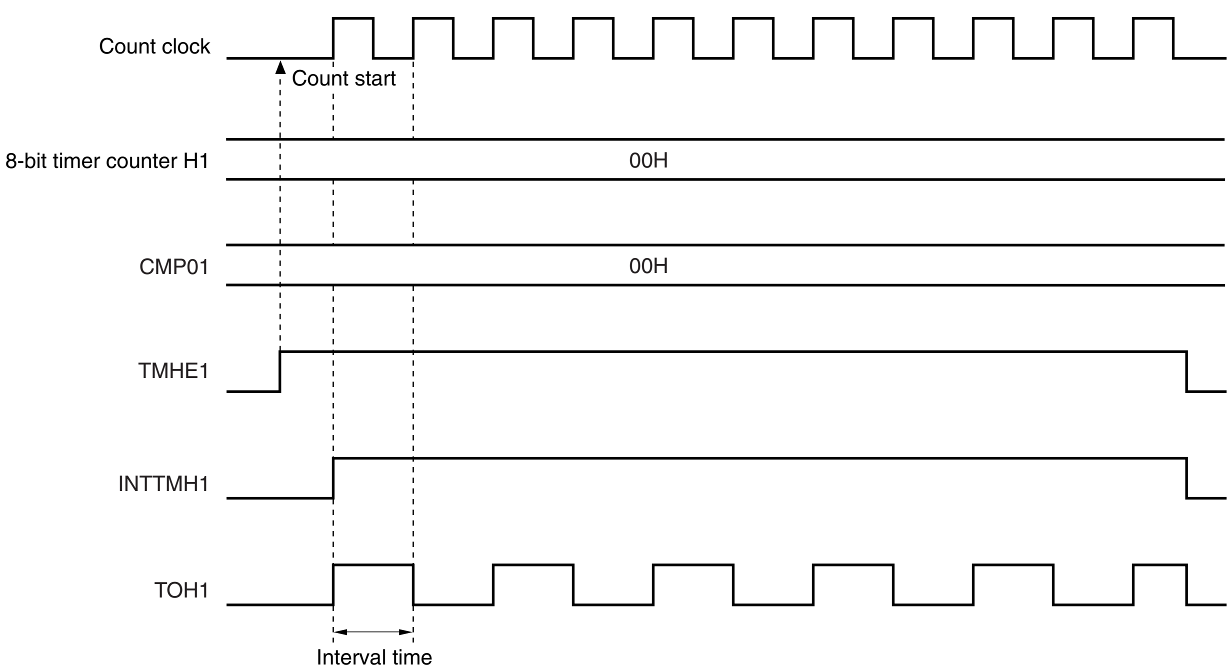
Remark 01H ≤ N ≤ FEH

Figure 6-7. Timing of Interval Timer/Square-Wave Output Operation (2/2)

(b) Operation when CMP01 = FFH



(c) Operation when CMP01 = 00H



6.4.2 Operation as PWM output mode

In PWM output mode, a pulse with an arbitrary duty and arbitrary cycle can be output.

8-bit timer compare register 01 (CMP01) controls the cycle of timer output (TOH1). Rewriting the CMP01 register during timer operation is prohibited.

8-bit timer compare register 11 (CMP11) controls the duty of timer output (TOH1). Rewriting the CMP11 register during timer operation is possible.

The operation in PWM output mode is as follows.

TOH1 output becomes active and 8-bit timer counter H1 is cleared to 0 when 8-bit timer counter H1 and the CMP01 register match after the timer count is started. TOH1 output becomes inactive when 8-bit timer counter H1 and the CMP11 register match.

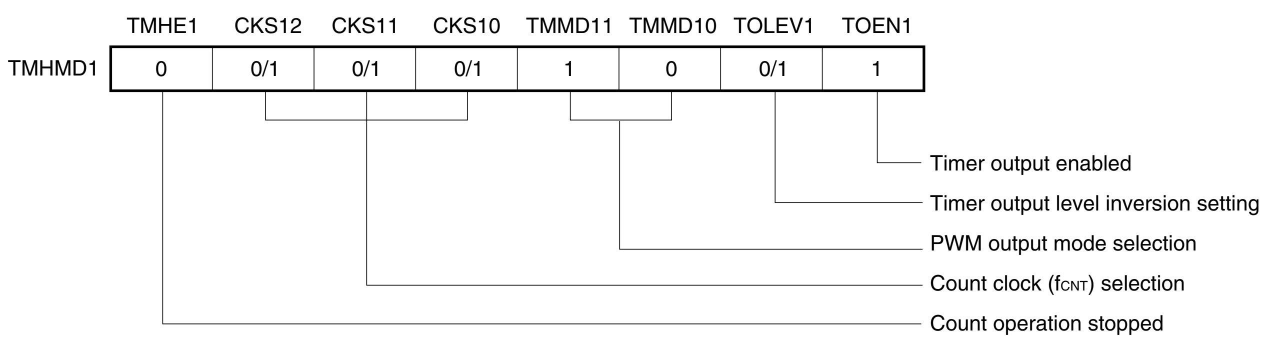
(1) Usage

In PWM output mode, a pulse for which an arbitrary duty and arbitrary cycle can be set is output.

<1> Set each register.

Figure 6-8. Register Setting in PWM Output Mode

(i) Setting timer H mode register 1 (TMHMD1)



(ii) Setting CMP01 register

- Compare value (N): Cycle setting

(iii) Setting CMP11 register

- Compare value (M): Duty setting

Remark 00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH

<2> The count operation starts when TMHE1 = 1.

<3> The CMP01 register is the compare register that is to be compared first after count operation is enabled. When the values of 8-bit timer counter H1 and the CMP01 register match, 8-bit timer counter H1 is cleared, an interrupt request signal (INTTMH1) is generated, and TOH1 output becomes active. At the same time, the compare register to be compared with 8-bit timer counter H1 is changed from the CMP01 register to the CMP11 register.

<4> When 8-bit timer counter H1 and the CMP11 register match, TOH1 output becomes inactive and the compare register to be compared with 8-bit timer counter H1 is changed from the CMP11 register to the CMP01 register. At this time, 8-bit timer counter H1 is not cleared and the INTTMH1 signal is not generated.

<5> By performing procedures <3> and <4> repeatedly, a pulse with an arbitrary duty can be obtained.

<6> To stop the count operation, set TMHE1 = 0.

If the setting value of the CMP01 register is N, the setting value of the CMP11 register is M, and the count clock frequency is f_{CNT}, the PWM pulse output cycle and duty are as follows.

PWM pulse output cycle = $(N+1)/f_{CNT}$ Duty = Active width : Total width of PWM = (M + 1) : (N + 1)

Cautions 1. In PWM output mode, the setting value for the CMP11 register can be changed during timer count operation. However, three operation clocks (signal selected using the CKS12 to CKS10 bits of the TMHMD1 register) or more are required to transfer the register value after rewriting the CMP11 register value.

2. Be sure to set the CMP11 register when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).

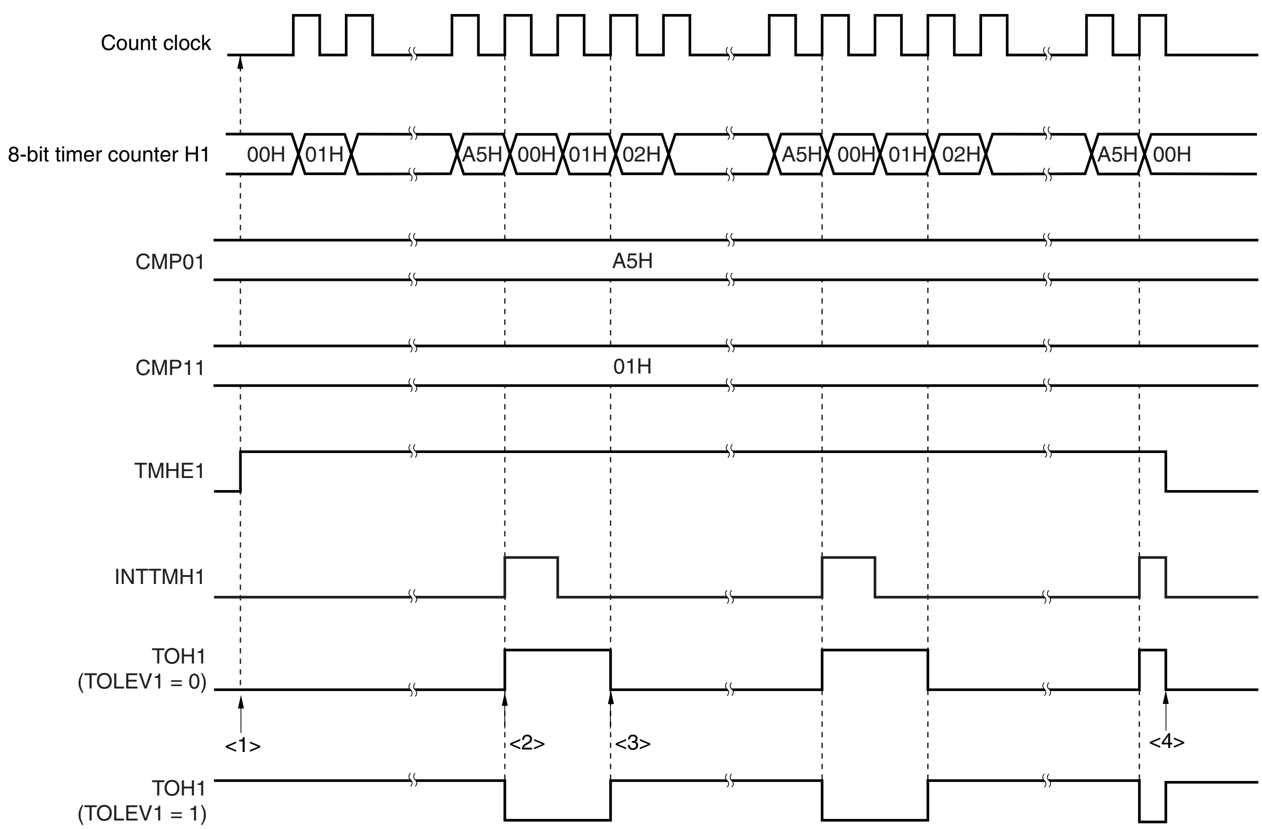
(2) Timing chart

The operation timing in PWM output mode is shown below.

Caution Make sure that the CMP11 register setting value (M) and CMP01 register setting value (N) are within the following range.
00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH

Figure 6-9. Operation Timing in PWM Output Mode (1/4)

(a) Basic operation (00H < CMP11 < CMP01 < FFH)



<1> The count operation is enabled by setting the TMHE1 bit to 1. Start 8-bit timer counter H1 by masking one count clock to count up. At this time, TOH1 output remains inactive (when TOLEV1 = 0).

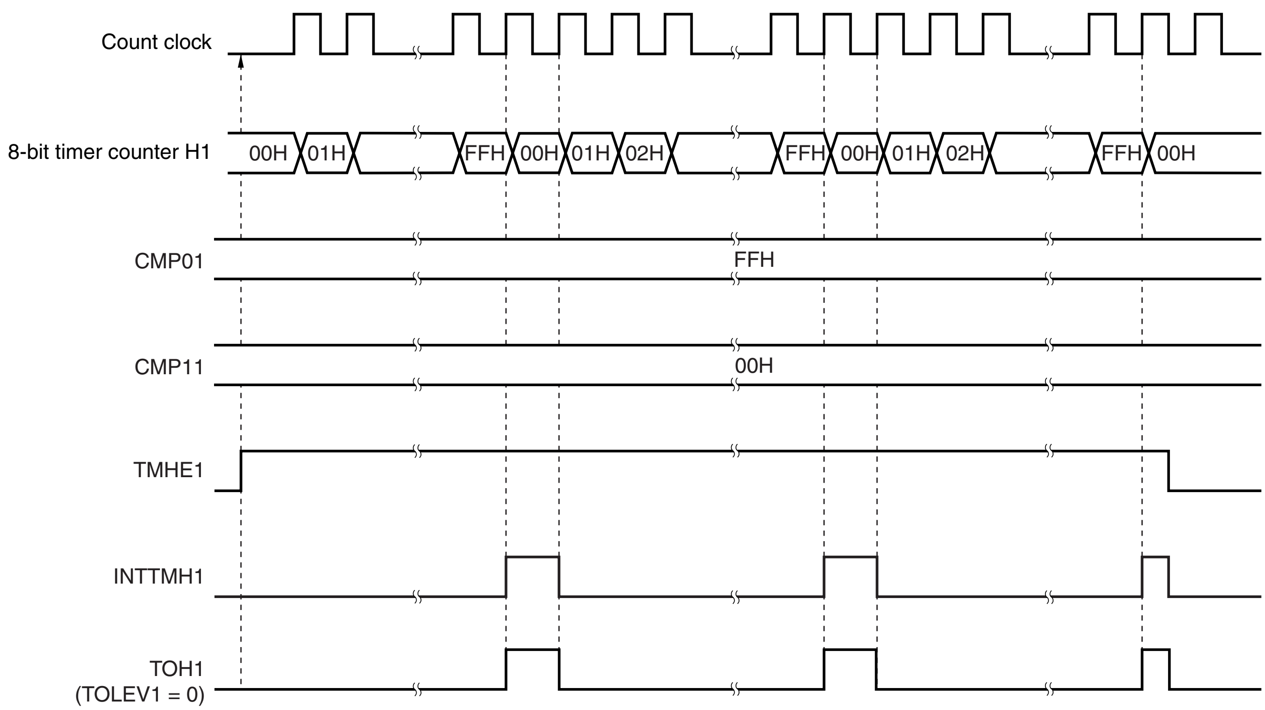
<2> When the values of 8-bit timer counter H1 and the CMP01 register match, the TOH1 output level is inverted, the value of 8-bit timer counter H1 is cleared, and the INTTMH1 signal is output.

<3> When the values of 8-bit timer counter H1 and the CMP11 register match, the level of the TOH1 output is returned. At this time, the 8-bit timer counter value is not cleared and the INTTMH1 signal is not output.

<4> Clearing the TMHE1 bit to 0 during timer H1 operation makes the INTTMH1 signal and TOH1 output inactive.

Figure 6-9. Operation Timing in PWM Output Mode (2/4)

(b) Operation when CMP01 = FFH, CMP11 = 00H



(c) Operation when CMP01 = FFH, CMP11 = FEH

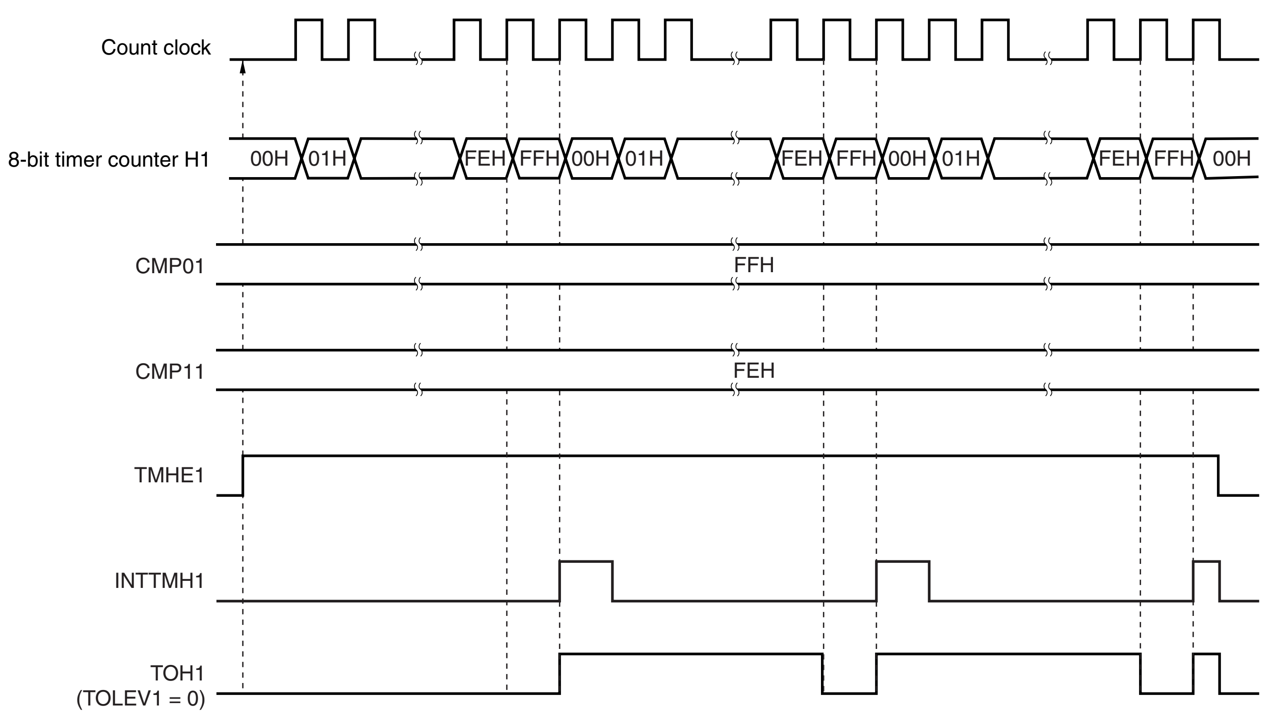


Figure 6-9. Operation Timing in PWM Output Mode (3/4)

(d) Operation when CMP01 = 01H, CMP11 = 00H

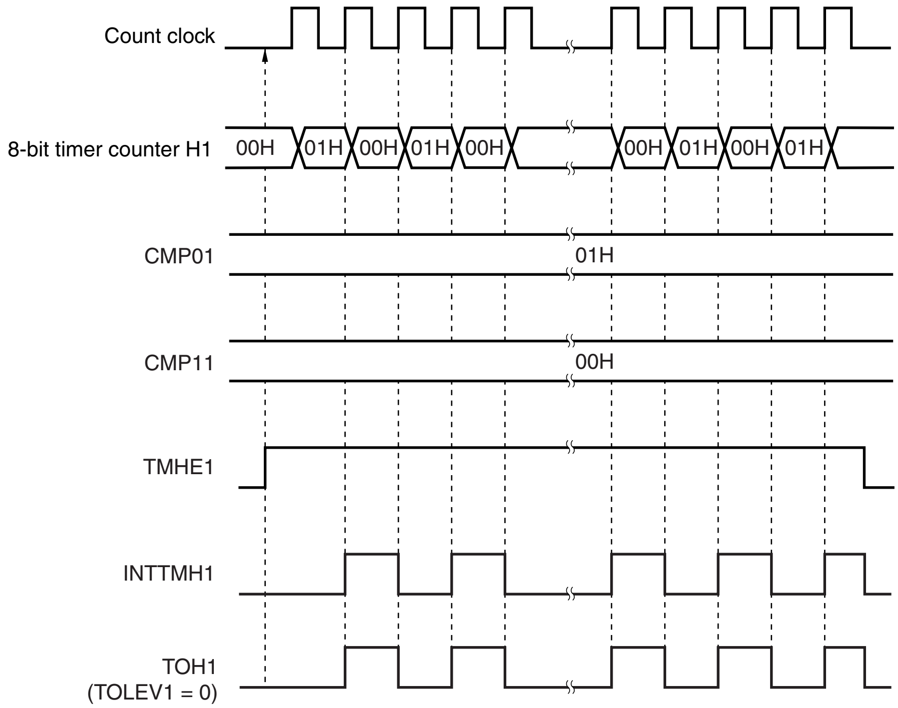
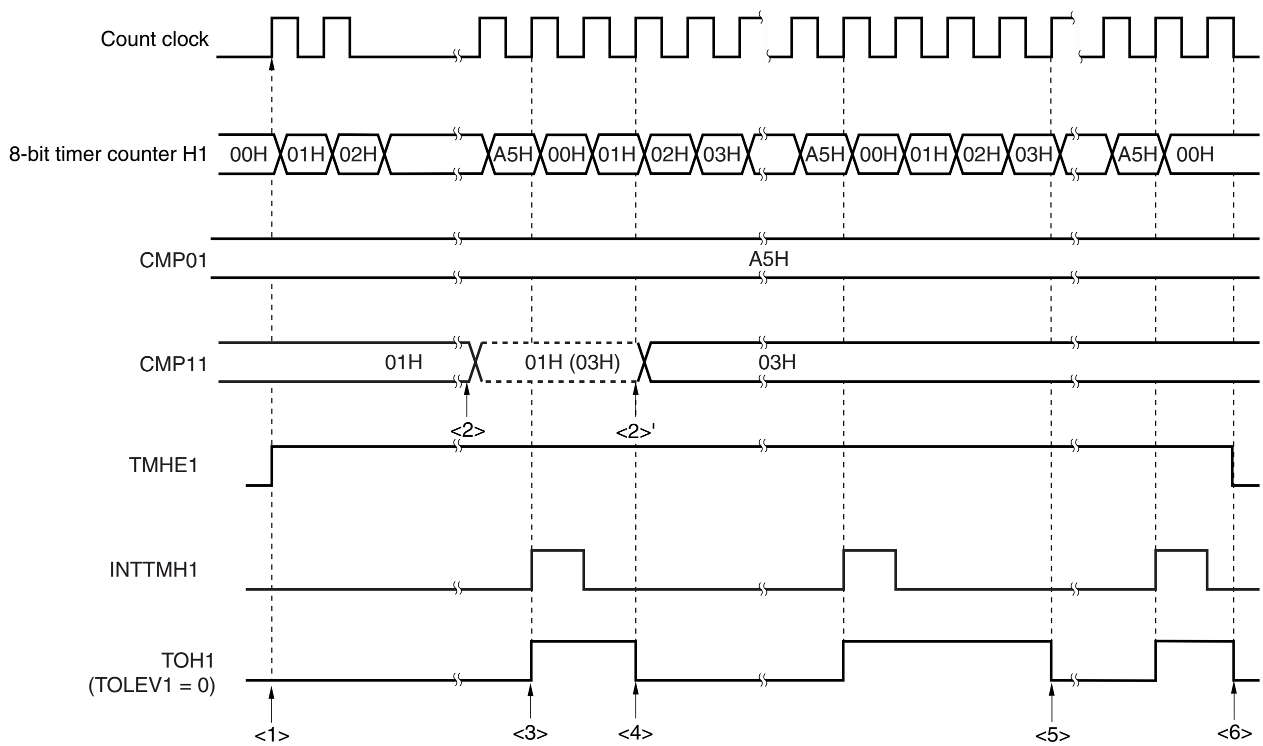


Figure 6-9. Operation Timing in PWM Output Mode (4/4)

(e) Operation by changing CMP11 (CMP11 = 02H → 03H, CMP01 = A5H)



<1> The count operation is enabled by setting TMHE1 = 1. Start 8-bit timer counter H1 by masking one count clock to count up. At this time, the TOH1 output remains inactive (when TOLEV1 = 0).

<2> The CMP11 register value can be changed during timer counter operation. This operation is asynchronous to the count clock.

<3> When the values of 8-bit timer counter H1 and the CMP01 register match, the value of 8-bit timer counter H1 is cleared, the TOH1 output becomes active, and the INTTMH1 signal is output.

<4> If the CMP11 register value is changed, the value is latched and not transferred to the register. When the values of 8-bit timer counter H1 and the CMP11 register before the change match, the value is transferred to the CMP11 register and the CMP11 register value is changed (<2>').
However, three count clocks or more are required from when the CMP11 register value is changed to when the value is transferred to the register. If a match signal is generated within three count clocks, the changed value cannot be transferred to the register.

<5> When the values of 8-bit timer counter H1 and the CMP11 register after the change match, the TOH1 output becomes inactive. 8-bit timer counter H1 is not cleared and the INTTMH1 signal is not generated.

<6> Clearing the TMHE1 bit to 0 during timer H1 operation makes the INTTMH1 signal and TOH1 output inactive.

CHAPTER 7 WATCHDOG TIMER

7.1 Functions of Watchdog Timer

The watchdog timer is used to detect an inadvertent program loop. If a program loop is detected, an internal reset signal is generated.

When a reset occurs due to the watchdog timer, bit 4 (WDTRF) of the reset control flag register (RESF) is set to 1. For details of RESF, see **CHAPTER 10 RESET FUNCTION**.

Table 7-1. Loop Detection Time of Watchdog Timer

Loop Detection Time	
During Low-Speed Internal oscillation Clock Operation	During System Clock Operation
$2^{11}/f_{RL}$ (4.27 ms)	$2^{13}/f_X$ (819.2 μs)
$2^{12}/f_{RL}$ (8.53 ms)	$2^{14}/f_X$ (1.64 ms)
$2^{13}/f_{RL}$ (17.07 ms)	$2^{15}/f_X$ (3.28 ms)
$2^{14}/f_{RL}$ (34.13 ms)	$2^{16}/f_X$ (6.55 ms)
$2^{15}/f_{RL}$ (68.27 ms)	$2^{17}/f_X$ (13.11 ms)
$2^{16}/f_{RL}$ (136.53 ms)	$2^{18}/f_X$ (26.21 ms)
$2^{17}/f_{RL}$ (273.07 ms)	$2^{19}/f_X$ (52.43 ms)
$2^{18}/f_{RL}$ (546.13 ms)	$2^{20}/f_X$ (104.86 ms)

Remarks 1. f_{RL}: Low-speed internal oscillation clock oscillation frequency

2. f_X: System clock oscillation frequency

3. Figures in parentheses apply to operation at f_{RL} = 480 kHz (MAX.), f_X = 10 MHz.

The operation mode of the watchdog timer (WDT) is switched according to the option byte setting of the on-chip low-speed internal oscillator as shown in Table 7-2.

Table 7-2. Option Byte Setting and Watchdog Timer Operation Mode

	Option Byte Setting	
	Low-Speed Internal Oscillator Cannot Be Stopped	Low-Speed Internal Oscillator Can Be Stopped by Software
Watchdog timer clock source	Fixed to f_{RL}[Note 1].	• Selectable by software (f_X, f_{RL} or stopped) • When reset is released: f_{RL}
Operation after reset	Operation starts with the maximum interval ($2^{18}/f_{RL}$).	Operation starts with the maximum interval ($2^{18}/f_{RL}$).
Operation mode selection	The interval can be changed only once.	The clock selection/interval can be changed only once.
Features	The watchdog timer cannot be stopped.	The watchdog timer can be stopped[Note 2].

Notes 1. As long as power is being supplied, low-speed internal oscillator cannot be stopped (except in the reset period).

2. The conditions under which clock supply to the watchdog timer is stopped differ depending on the clock source of the watchdog timer.

<1> If the clock source is f_X, clock supply to the watchdog timer is stopped under the following conditions.
- When f_X is stopped
- In HALT/STOP mode
- During oscillation stabilization time

<2> If the clock source is f_{RL}, clock supply to the watchdog timer is stopped under the following conditions.
- If the CPU clock is f_X and if f_{RL} is stopped by software before execution of the STOP instruction
- In HALT/STOP mode

Remarks 1. f_{RL}: Low-speed internal oscillation clock oscillation frequency

2. f_X: System clock oscillation frequency

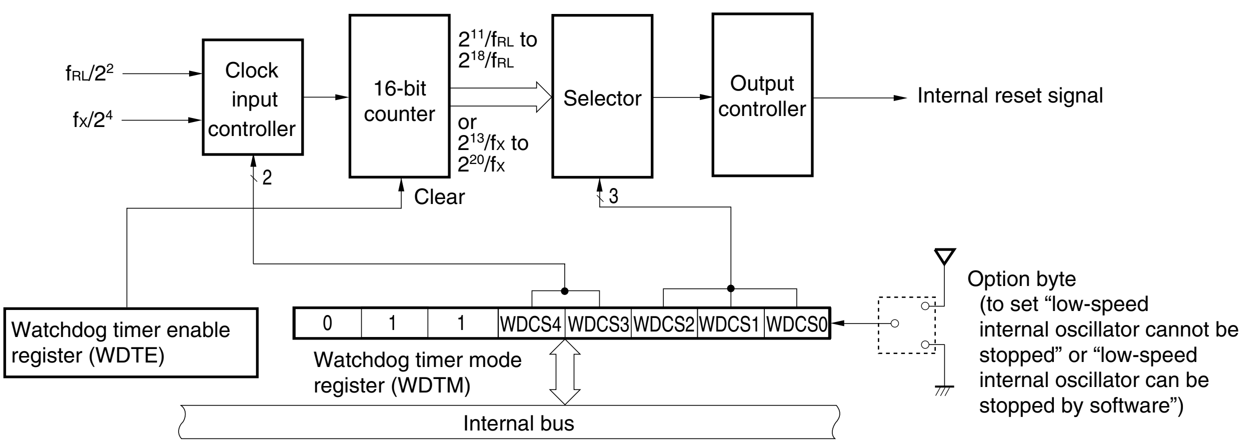
7.2 Configuration of Watchdog Timer

The watchdog timer consists of the following hardware.

Table 7-3. Configuration of Watchdog Timer

Item	Configuration
Control registers	Watchdog timer mode register (WDTM) Watchdog timer enable register (WDTE)

Figure 7-1. Block Diagram of Watchdog Timer



Remarks 1. f_{RL}: Low-speed internal oscillation clock oscillation frequency

2. f_X: System clock oscillation frequency

7.3 Registers Controlling Watchdog Timer

The watchdog timer is controlled by the following two registers.

- Watchdog timer mode register (WDTM)
- Watchdog timer enable register (WDTE)

(1) Watchdog timer mode register (WDTM)

This register sets the overflow time and operation clock of the watchdog timer.

This register can be set by an 7-bit memory manipulation instruction and can be read many times, but can be written only once after reset is released.

Reset signal generation sets this register to 67H.

Figure 7-2. Format of Watchdog Timer Mode Register (WDTM)

Address: FF48H After reset: 67H R/W

Symbol	7	6	5	4	3	2	1	0
WDTM	0	1	1	WDCS4	WDCS3	WDCS2	WDCS1	WDCS0

WDCS4 Note 1	WDCS3 Note 1	Operation clock selection
0	0	Low-speed internal oscillation clock (f_{RL})
0	1	System Clock (f_X)
1	×	Watchdog timer operation stopped

WDCS2 Note 2	WDCS1 Note 2	WDCS0 Note 2	Overflow time setting	
			During low-speed internal oscillation clock operation	During system clock operation
0	0	0	$2^{11}/f_{RL}$ (4.27 ms)	$2^{13}/f_X$ (819.2 μs)
0	0	1	$2^{12}/f_{RL}$ (8.53 ms)	$2^{14}/f_X$ (1.64 ms)
0	1	0	$2^{13}/f_{RL}$ (17.07 ms)	$2^{15}/f_X$ (3.28 ms)
0	1	1	$2^{14}/f_{RL}$ (34.13 ms)	$2^{16}/f_X$ (6.55 ms)
1	0	0	$2^{15}/f_{RL}$ (68.27 ms)	$2^{17}/f_X$ (13.11 ms)
1	0	1	$2^{16}/f_{RL}$ (136.53 ms)	$2^{18}/f_X$ (26.21 ms)
1	1	0	$2^{17}/f_{RL}$ (273.07 ms)	$2^{19}/f_X$ (52.43 ms)
1	1	1	$2^{18}/f_{RL}$ (546.13 ms)	$2^{20}/f_X$ (104.86 ms)

Notes 1. If “low-speed internal oscillator cannot be stopped” is specified by the option byte, this cannot be set. The low-speed internal oscillation clock will be selected no matter what value is written.

2. Reset is released at the maximum cycle (WDCS2, 1, 0 = 1, 1, 1).

Cautions 1. Set bits 7, 6, and 5 to 0, 1, and 1, respectively. Do not set the other values.

Cautions 2. After reset is released, WDTM can be written only once by an 8-bit memory manipulation instruction. If writing is attempted a second time, an internal reset signal is generated. However, at the first write, if “1” and “x” are set for WDCS4 and WDCS3 respectively and the watchdog timer is stopped, then the internal reset signal does not occur even if the following are executed.

- **Second write to WDTM**
- **1-bit memory manipulation instruction to WDTE**
- **Writing of a value other than “ACH” to WDTE**

3. WDTM cannot be set by a 1-bit memory manipulation instruction.

4. When using the flash memory self programming by self writing, set the overflow time for the watchdog timer so that enough everflow time is secured (Example 1-byte writing: 200 μs MIN., 1-block deletion: 10 ms MIN.).

Remarks 1. f_{RL}: Low-speed internal oscillation clock oscillation frequency

2. f_X: System clock oscillation frequency

3. ×: Don’t care

4. Figures in parentheses apply to operation at f_{RL} = 480 kHz (MAX.), f_X = 10 MHz.

(2) Watchdog timer enable register (WDTE)

Writing ACH to WDTE clears the watchdog timer counter and starts counting again.
This register can be set by an 8-bit memory manipulation instruction.
Reset signal generation sets this register to 9AH.

Figure 7-3. Format of Watchdog Timer Enable Register (WDTE)

Address: FF49H After reset: 9AH R/W

Symbol	7	6	5	4	3	2	1	0
WDTE								

Cautions 1. If a value other than ACH is written to WDTE, an internal reset signal is generated.

2. If a 1-bit memory manipulation instruction is executed for WDTE, an internal reset signal is generated.

3. The value read from WDTE is 9AH (this differs from the written value (ACH)).

7.4 Operation of Watchdog Timer

7.4.1 Watchdog timer operation when "low-speed internal oscillator cannot be stopped" is selected by option byte

The operation clock of watchdog timer is fixed to low-speed internal oscillation clock.

After reset is released, operation is started at the maximum cycle (bits 2, 1, and 0 (WDCS2, WDCS1, WDCS0) of the watchdog timer mode register (WDTM) = 1, 1, 1). The watchdog timer operation cannot be stopped.

The following shows the watchdog timer operation after reset release.

1. The status after reset release is as follows.
   - Operation clock: Low-speed internal oscillation clock
   - Cycle: $2^{18}/f_{RL}$ (546.13 ms: At operation with f_{RL} = 480 kHz (MAX.))
   - Counting starts
2. The following should be set in the watchdog timer mode register (WDTM) by an 8-bit memory manipulation instruction[Notes 1, 2].
   - Cycle: Set using bits 2 to 0 (WDCS2 to WDCS0)
3. After the above procedures are executed, writing ACH to WDTE clears the count to 0, enabling recounting.

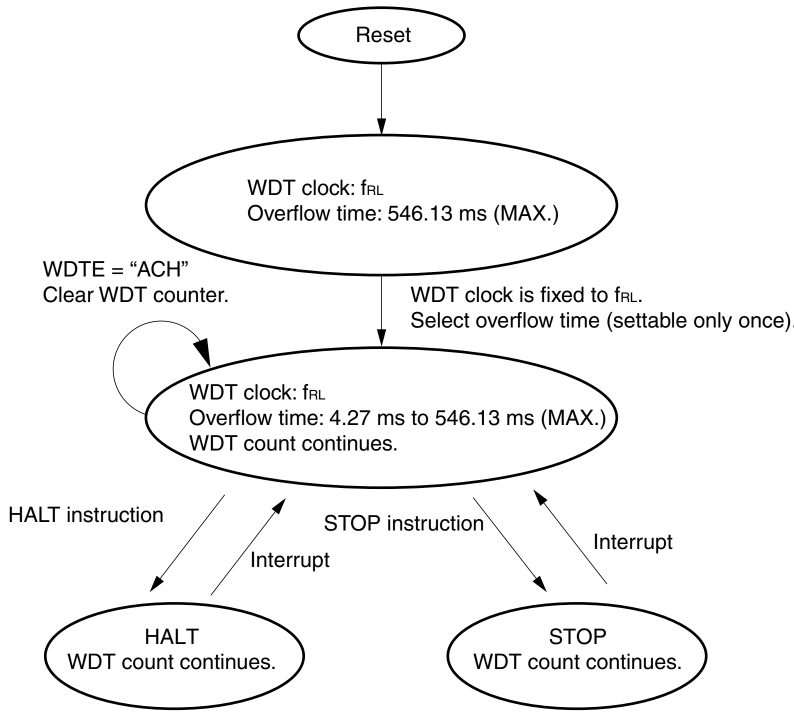
Notes 1. The operation clock (low-speed internal oscillation clock) cannot be changed. If any value is written to bits 3 and 4 (WDCS3, WDCS4) of WDTM, it is ignored.

2. As soon as WDTM is written, the counter of the watchdog timer is cleared.

Caution In this mode, operation of the watchdog timer cannot be stopped even during STOP instruction execution. For 8-bit timer H1 (TMH1), a division of the low-speed internal oscillation clock can be selected as the count source, so clear the watchdog timer using the interrupt request of TMH1 before the watchdog timer overflows after STOP instruction execution. If this processing is not performed, an internal reset signal is generated when the watchdog timer overflows after STOP instruction execution.

A status transition diagram is shown below

Figure 7-4. Status Transition Diagram When “Low-Speed Internal Oscillator Cannot Be Stopped” Is Selected by Option Byte



7.4.2 Watchdog timer operation when “low-speed internal oscillator can be stopped by software” is selected by option byte

The operation clock of the watchdog timer can be selected as either the low-speed internal oscillation clock or system clock.

After reset is released, operation is started at the maximum cycle of the low-speed internal oscillation clock (bits 2, 1, and 0 (WDCS2, WDCS1, WDCS0) of the watchdog timer mode register (WDTM) = 1, 1, 1).

The following shows the watchdog timer operation after reset release.

1. The status after reset release is as follows.
   - Operation clock: Low-speed internal oscillation clock
   - Cycle: $2^{18}/f_{RL}$ (546.13 ms: At operation with f_{RL} = 480 kHz (MAX.))
   - Counting starts
2. The following should be set in the watchdog timer mode register (WDTM) by an 8-bit memory manipulation instruction[Notes 1, 2, 3].
   - Operation clock: Any of the following can be selected using bits 3 and 4 (WDCS3 and WDCS4).
     Low-speed internal oscillation clock
     Syatem clock (f_X)
     Watchdog timer operation stopped
   - Cycle: Set using bits 2 to 0 (WDCS2 to WDCS0)
3. After the above procedures are executed, writing ACH to WDTE clears the count to 0, enabling recounting.

Notes 1. As soon as WDTM is written, the counter of the watchdog timer is cleared.

2. Set bits 7, 6, and 5 to 0, 1, 1, respectively. Do not set the other values.

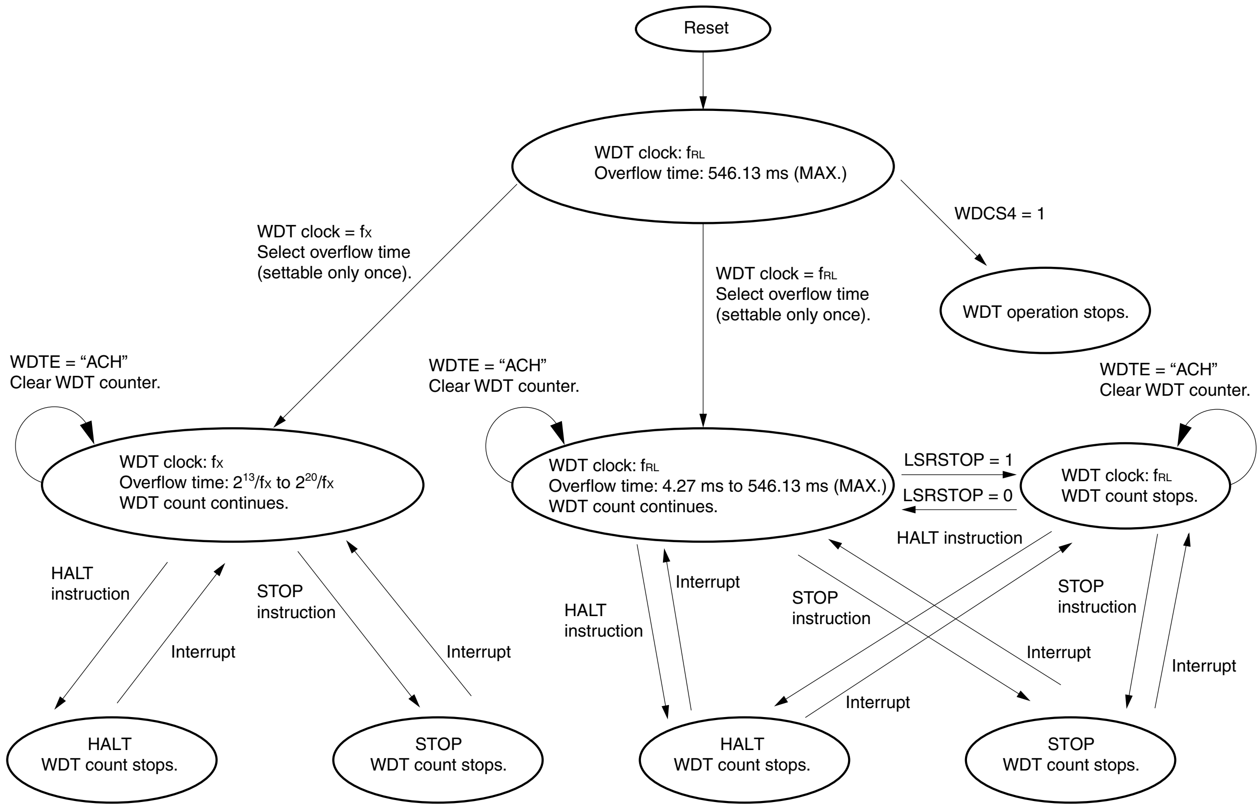
3. At the first write, If the watchdog timer is stopped by setting WDCS4 and WDCS3 to 1 and ×, respectively, an internal reset signal is not generated even if the following processing is performed.
- WDTM is written a second time.
- A 1-bit memory manipulation instruction is executed to WDTE.
- A value other than ACH is written to WDTE.

Caution In this mode, watchdog timer operation is stopped during HALT/STOP instruction execution. After HALT/STOP mode is released, counting is started again using the operation clock of the watchdog timer set before HALT/STOP instruction execution by WDTM. At this time, the counter is not cleared to 0 but holds its value.

For the watchdog timer operation during STOP mode and HALT mode in each status, see **7.4.3 Watchdog timer operation in STOP mode** and **7.4.4 Watchdog timer operation in HALT mode**.

A status transition diagram is shown below.

Figure 7-5. Status Transition Diagram When “Low-Speed Internal Oscillator Can Be Stopped by Software” Is Selected by Option Byte



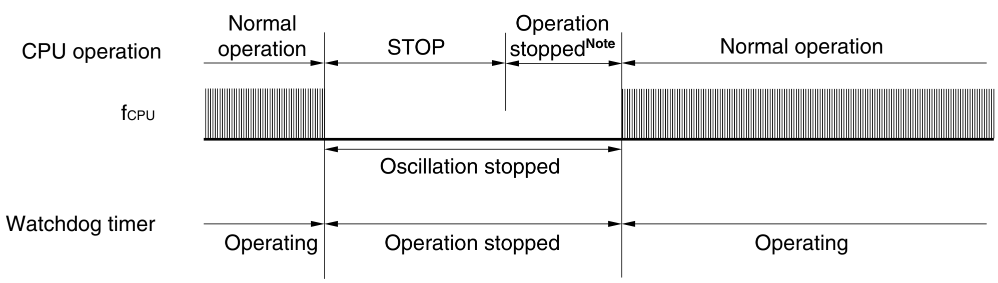
7.4.3 Watchdog timer operation in STOP mode (when "low-speed internal oscillator can be stopped by software" is selected by option byte)

The watchdog timer stops counting during STOP instruction execution regardless of whether the system clock or low-speed internal oscillation clock is being used.

(1) When the watchdog timer operation clock is the system clock (f_X) when the STOP instruction is executed

When STOP instruction is executed, operation of the watchdog timer is stopped. After STOP mode is released, operation stops for 34 μs (TYP.) and then counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

Figure 7-6. Operation in STOP Mode (WDT Operation Clock: Clock to Peripheral Hardware)

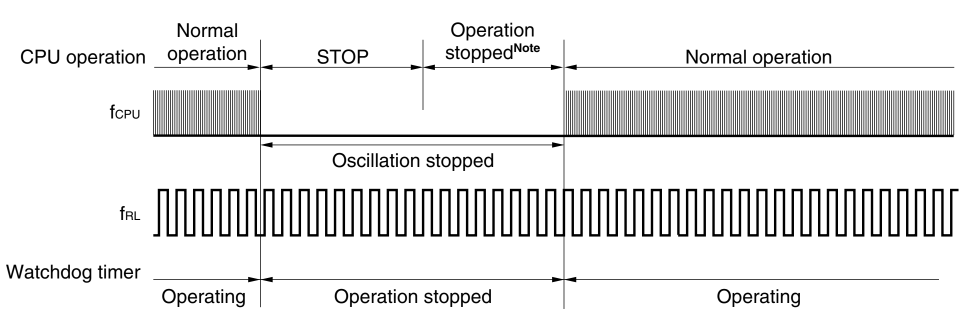


Note The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

(2) When the watchdog timer operation clock is the low-speed internal oscillation clock (f_{RL}) when the STOP instruction is executed

When the STOP instruction is executed, operation of the watchdog timer is stopped. After STOP mode is released, operation stops for 34 μs (TYP.) and then counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

Figure 7-7. Operation in STOP Mode (WDT Operation Clock: Low-Speed Internal Oscillation Clock)

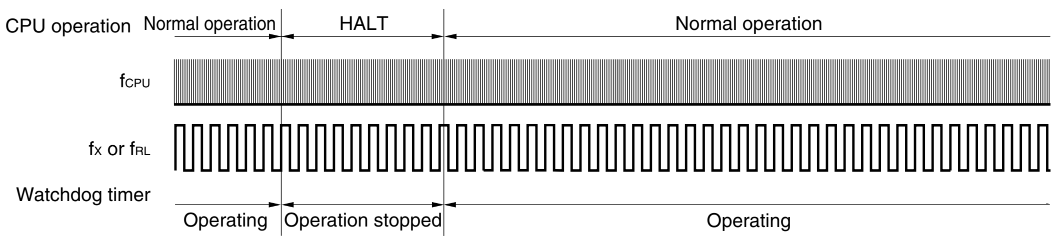


Note The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

7.4.4 Watchdog timer operation in HALT mode (when "low-speed internal oscillator can be stopped by software" is selected by option byte)

The watchdog timer stops counting during HALT instruction execution regardless of whether the operation clock of the watchdog timer is the system clock (f_X) or low-speed internal oscillation clock (f_{RL}). After HALT mode is released, counting is started again using the operation clock before the operation was stopped. At this time, the counter is not cleared to 0 but holds its value.

Figure 7-8. Operation in HALT Mode



CHAPTER 8 INTERRUPT FUNCTIONS

8.1 Interrupt Function Types

There are two types of interrupts: maskable interrupts and resets.

- **Maskable interrupts**
  These interrupts undergo mask control. When an interrupt request occurs, the standby release signal occurs, and if an interrupt can be acknowledged then the program corresponding to the address written in the vector table address is executed (vector interrupt servicing). When several interrupt requests are generated at the same time, processing takes place in the priority order of the vector interrupt servicing. For details on the priority order, see Table 8-1.
  There are two internal sources and two external sources of maskable interrupts.

- **Reset**
  The CPU and SFR are returned to their initial states by the reset signal. The causes for reset signal occurrences are shown in Table 8-1.
  When a reset signal occurs, program execution starts from the programs at the addresses written in addresses 0000H and 0001H.

8.2 Interrupt Sources and Configuration

There are a total of 4 maskable interrupt sources, and up to four reset sources (see **Table 8-1**).

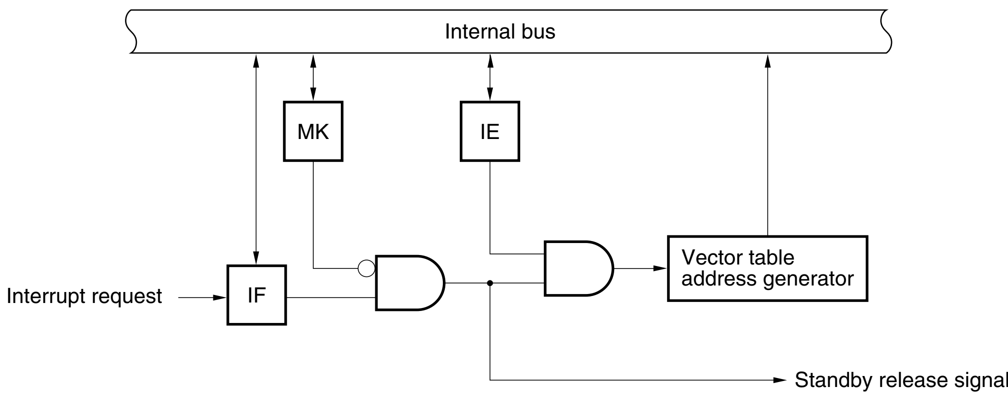
Table 8-1. Interrupt Sources

Interrupt Type	Priority[Note 1]	Interrupt Source		Internal/External	Vector Table Address	Basic Configuration Type[Note 2]
		Name	Trigger			
Maskable	1	INTLVI	Low-voltage detection[Note 3]	Internal	0006H	(A)
	2	INTP0	Pin input edge detection	External	0008H	(B)
	3	INTP1			000AH	
	4	INTTMH1	Match between TMH1 and CMP01 (when compare register is specified)	Internal	000CH	(A)
Reset	–	RESET	Reset input	–	0000H	–
		POC	Power-on-clear			
		LVI	Low-voltage detection[Note 4]			
		WDT	WDT overflow			

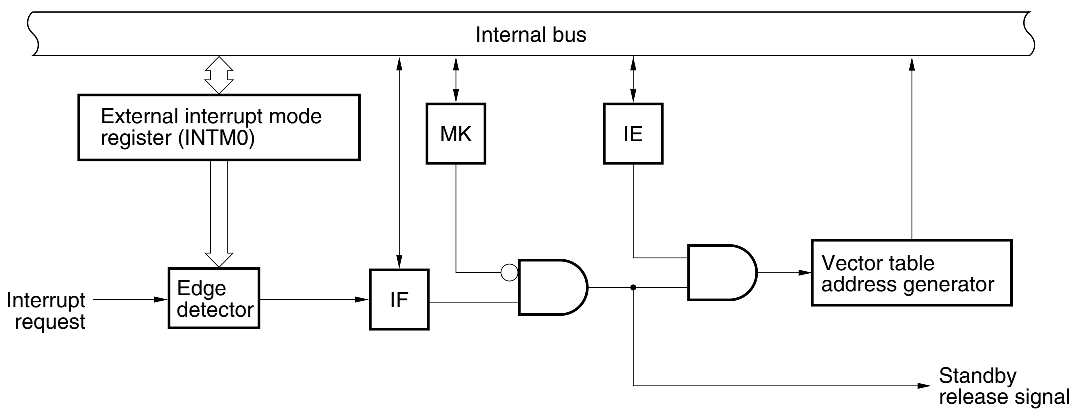
Notes 1. Priority is the vector interrupt servicing priority order when several maskable interrupt requests are generated at the same time. 1 is the highest and 4 is the lowest.
2. Basic configuration types (A) and (B) correspond to (A) and (B) in Figure 8-1.
3. When bit 1 (LVIMD) of low-voltage detection register (LVIM) = 0 is selected.
4. When bit 1 (LVIMD) of low-voltage detection register (LVIM) = 1 is selected.

Figure 8-1. Basic Configuration of Interrupt Function

(A) Internal maskable interrupt



(B) External maskable interrupt



IF: Interrupt request flag

IE: Interrupt enable flag

MK: Interrupt mask flag

8.3 Interrupt Function Control Registers

The interrupt functions are controlled by the following four types of registers.

- Interrupt request flag register 0 (IF0)
- Interrupt mask flag register 0 (MK0)
- External interrupt mode register 0 (INTM0)
- Program status word (PSW)

Table 8-2 lists interrupt requests, the corresponding interrupt request flags, and interrupt mask flags.

Table 8-2. Interrupt Request Signals and Corresponding Flags

Interrupt Request Signal	Interrupt Request Flag	Interrupt Mask Flag
INTLVI	LVIIF	LVIMK
INTP0	PIF0	PMK0
INTP1	PIF1	PMK1
INTTMH1	TMIFH1	TMMKH1

(1) Interrupt request flag register 0 (IF0)

An interrupt request flag is set to 1 when the corresponding interrupt request is issued, or when the instruction is executed. It is cleared to 0 by executing an instruction when the interrupt request is acknowledged or when a reset signal is input.
IF0 is set with a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation clears IF0 to 00H.

Figure 8-2. Format of Interrupt Request Flag Register 0 (IF0)

Address: FFE0H After reset: 00H R/W

Symbol	7	6	5	<4>	<3>	<2>	<1>	0
IF0	0	0	0	TMIFH1	PIF1	PIF0	LVIIF	0

××IF×	Interrupt request flag
0	No interrupt request signal has been issued.
1	An interrupt request signal has been issued; an interrupt request status.

Caution Because P21 and P32 have an alternate function as external interrupt inputs, when the output level is changed by specifying the output mode of the port function, an interrupt request flag is set. Therefore, the interrupt mask flag should be set to 1 before using the output mode.

(2) Interrupt mask flag register 0 (MK0)

The interrupt mask flag is used to enable and disable the corresponding maskable interrupts.
MK0 is set with a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation sets MK0 to FFH.

Figure 8-3. Format of Interrupt Mask Flag Register 0 (MK0)

Address: FFE4H After reset: FFH R/W

Symbol	7	6	5	<4>	<3>	<2>	<1>	0
MK0	1	1	1	TMMKH1	PMK1	PMK0	LVIMK	1

××MK×	Interrupt servicing control
0	Enables interrupt servicing.
1	Disables interrupt servicing.

Caution Because P21 and P32 have an alternate function as external interrupt inputs, when the output level is changed by specifying the output mode of the port function, an interrupt request flag is set. Therefore, the interrupt mask flag should be set to 1 before using the output mode.

(3) External interrupt mode register 0 (INTM0)

This register is used to set the valid edge of INTP0 and INTP1.
INTM0 is set with an 8-bit memory manipulation instruction.
Reset signal generation clears INTM0 to 00H.

Figure 8-4. Format of External Interrupt Mode Register 0 (INTM0)

Address: FFECH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
INTM0	0	0	ES11	ES10	ES01	ES00	0	0

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. Be sure to clear bits 0, 1, 6, and 7 to 0.

Cautions 2. Before setting the INTM0 register, be sure to set the corresponding interrupt mask flag (××MK× = 1) to disable interrupts. After setting the INTM0 register, clear the interrupt request flag (××IF× = 0), then clear the interrupt mask flag (××MK× = 0), which will enable interrupts.

(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 vectored interrupt is acknowledged, the PSW is automatically saved to a stack, and the IE flag is reset to 0.

Reset signal generation sets PSW to 02H.

Figure 8-5. Program Status Word (PSW) Configuration

Symbol	7	6	5	4	3	2	1	0	After reset
PSW	IE	Z	0	AC	0	0	1	CY	02H

Z, AC, CY → Used in the execution of ordinary instructions

IE	Whether to enable/disable interrupt acknowledgment
0	Disabled
1	Enabled

8.4 Interrupt Servicing Operation

8.4.1 Maskable interrupt request acknowledgment 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. If the interrupt enabled status is in effect (when the IE flag is set to 1), then the request is acknowledged as a vector interrupt.

The time required to start the vectored interrupt servicing after a maskable interrupt request has been generated is shown in Table 8-3.

See Figures 8-7 and 8-8 for the interrupt request acknowledgment timing.

Table 8-3. 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 instructions.

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.

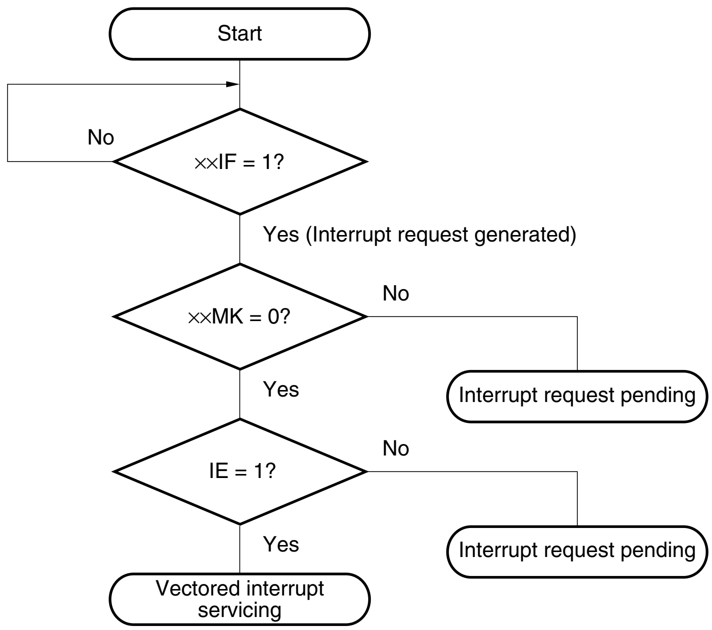
A pending interrupt is acknowledged when a status in which it can be acknowledged is set.

Figure 8-6 shows the algorithm of interrupt request acknowledgment.

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.

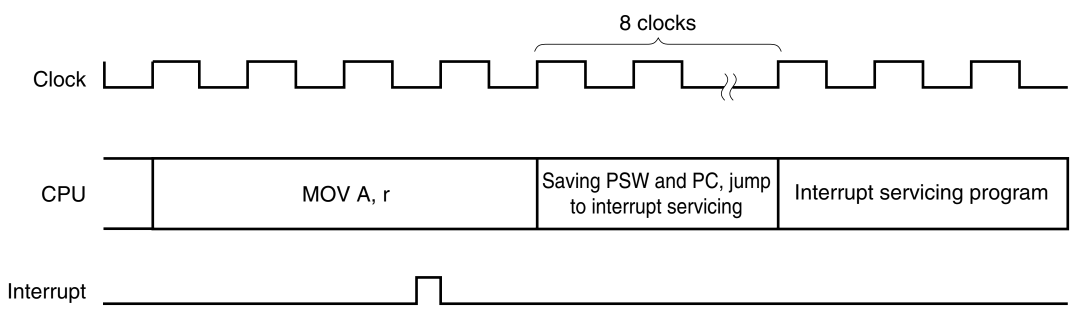
To return from interrupt servicing, use the RETI instruction.

Figure 8-6. Interrupt Request Acknowledgment Processing Algorithm



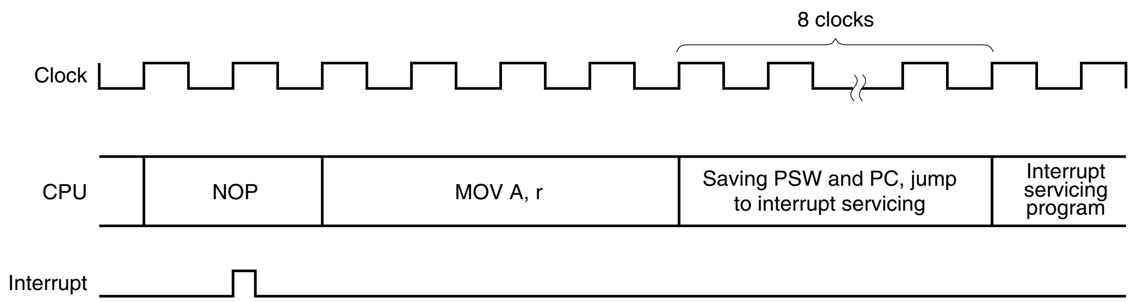
××IF: Interrupt request flag
××MK: Interrupt mask flag
IE: Flag to control maskable interrupt request acknowledgment (1 = enable, 0 = disable)

Figure 8-7. Interrupt Request Acknowledgment 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 8-7 shows an example of the interrupt request acknowledgment 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 instruction fetch starts, the interrupt acknowledgment processing is performed after the MOV A, r instruction is executed.

Figure 8-8. Interrupt Request Acknowledgment Timing (When Interrupt Request Flag Is Set at Last Clock During Instruction Execution)



If an interrupt request flag (××IF) is set at the last clock of the instruction, the interrupt acknowledgment processing starts after the next instruction is executed.

Figure 8-8 shows an example of the interrupt request acknowledgment 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 acknowledgment processing is performed.

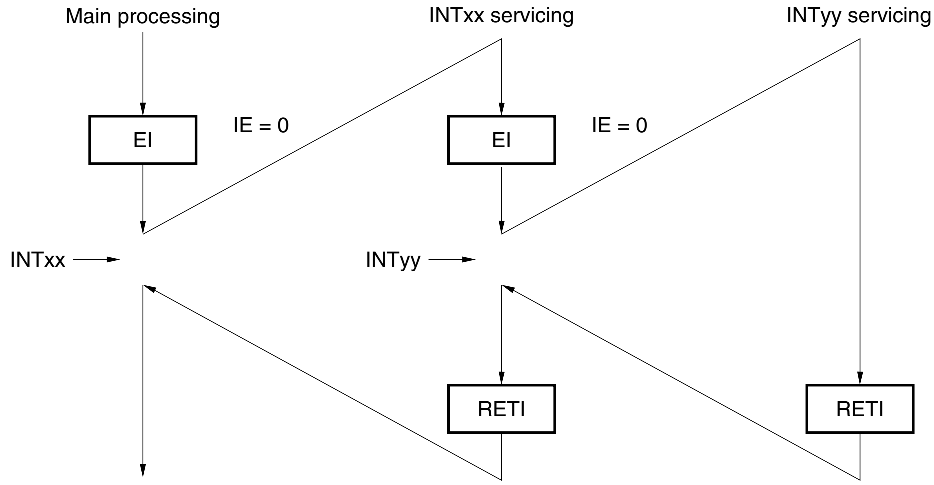
Caution Interrupt requests will be held pending while the interrupt request flag register 0 (IF0) or interrupt mask flag register 0 (MK0) are being accessed.

8.4.2 Multiple interrupt servicing

In order to perform multiple interrupt servicing in which another interrupt is acknowledged while an interrupt is being serviced, the interrupt mask function must be used to mask interrupts for which a low priority is to be set.

Figure 8-9. Example of Multiple Interrupts (1/2)

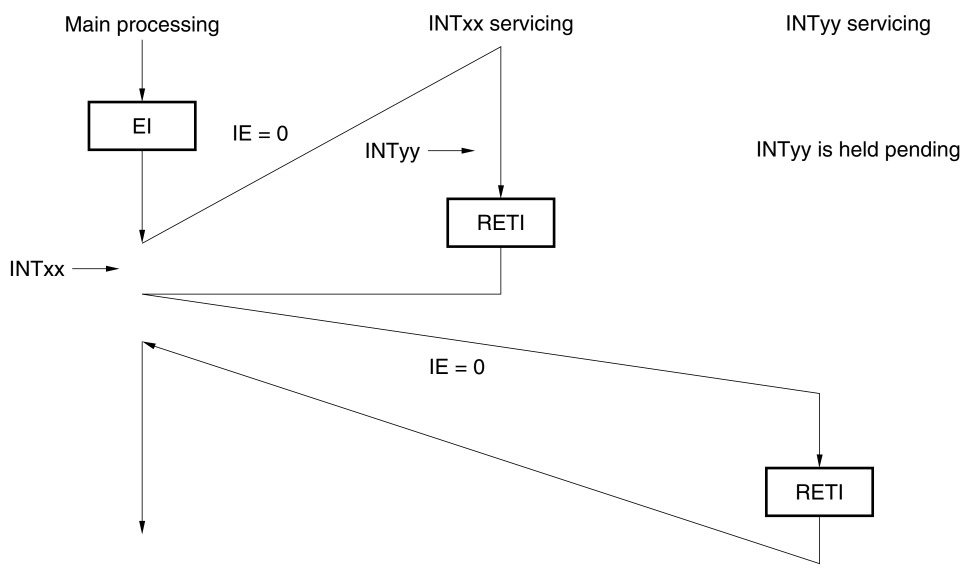
Example 1. Multiple interrupts are acknowledged



During interrupt INTxx servicing, interrupt request INTyy is acknowledged, and multiple interrupts are generated. Before each interrupt request acknowledgement, the EI instruction is issued, the interrupt mask is released, and the interrupt request acknowledgement enable state is set.

Caution Multiple interrupts can be acknowledged even for low-priority interrupts.

Example 2. Multiple interrupts are not generated because interrupts are not enabled



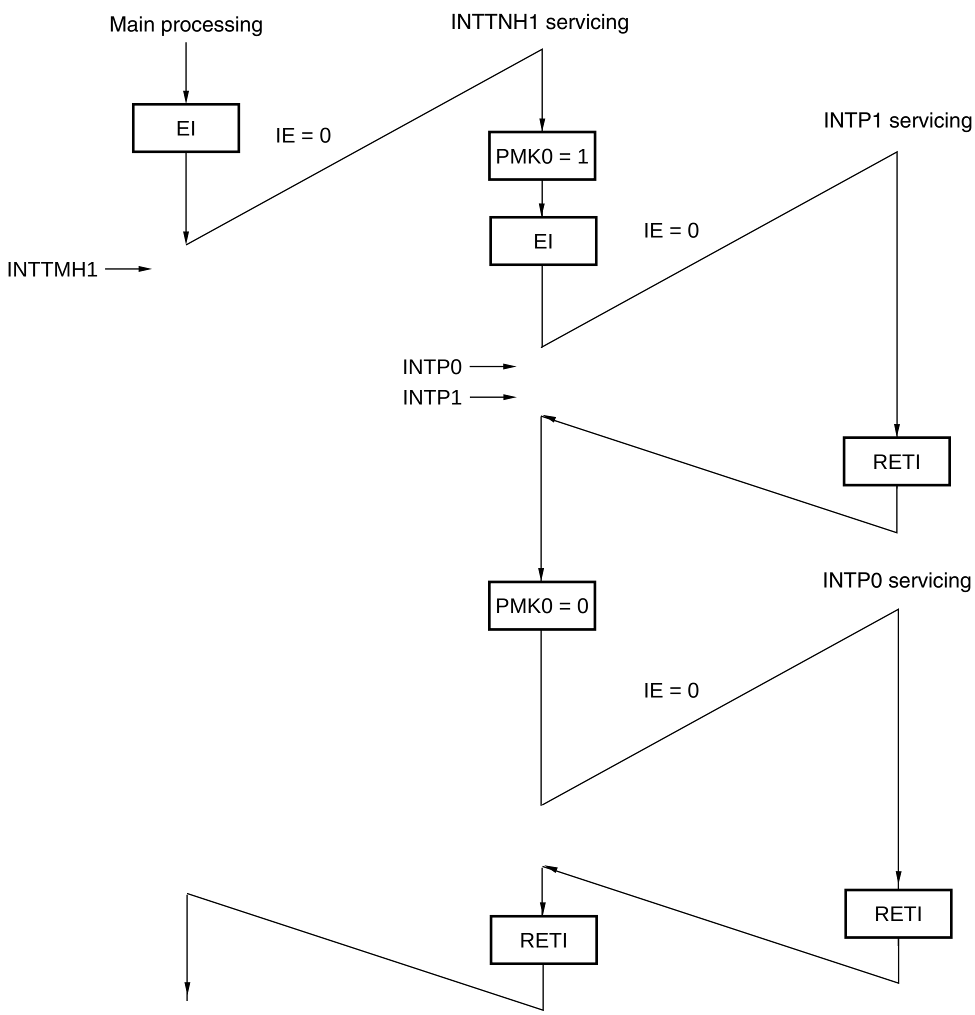
Because interrupts are not enabled in interrupt INTxx servicing (the EI instruction is not issued), interrupt request INTyy is not acknowledged, and multiple interrupts are not generated. The INTyy request is held pending and acknowledged after the INTxx servicing is performed.

IE = 0: Interrupt request acknowledgment disabled

Figure 8-9. Example of Multiple Interrupts (2/2)

Example 3. A priority is controlled by the Multiple interrupts

The vector interrupt enable state is set for INTP0, INTP1, and INTTMH1.
(Interruption priority INTP0 > INTP1 > INTTMH1 (refer to **Table8-1**))



In the interrupt INTTMH1 servicing, servicing is performed such that the INTP1 interrupt is given priority, since the INTP0 interrupt was first masked.

Afterwards, once the interrupt mask for INTP0 is released, INTP0 processing through multiple interrupts is performed.

IE = 0: Interrupt request acknowledgment disabled

8.4.3 Interrupt request pending

Some instructions may keep pending the acknowledgment of an instruction request until the completion of the execution of the next instruction even if the interrupt request (maskable interrupt and external interrupt) is generated during the execution. The following shows such instructions (interrupt request pending instruction).

- Manipulation instruction for interrupt request flag register 0 (IF0)
- Manipulation instruction for interrupt mask flag register 0 (MK0)

CHAPTER 9 STANDBY FUNCTION

9.1 Standby Function and Configuration

9.1.1 Standby function

Table 9-1. Relationship Between Operation Clocks in Each Operation Status

Status / Operation Mode	Low-Speed Internal Oscillator			System Clock	Clock Supplied to Peripheral Hardware
	Note 1	**Note 2**			
		LSRSTOP = 0	LSRSTOP = 1		
Reset	Stopped			Stopped	Stopped
STOP	Oscillating	Oscillating[Note 3]	Stopped		
HALT				Oscillating	Oscillating

Notes 1. When “Cannot be stopped” is selected for low-speed internal oscillator by the option byte.

2. When it is selected that the low-speed internal oscillator “can be stopped by software”, oscillation of the low-speed internal oscillator can be stopped by LSRSTOP.

3. If the operating clock of the watchdog timer is the low-speed internal oscillation clock, the watchdog timer is stopped.

Caution The LSRSTOP setting is valid only when “Can be stopped by software” is set for the low-speed internal oscillator by the option byte.

Remark LSRSTOP: Bit 0 of the low-speed internal oscillation mode register (LSRCM)

The standby function is designed to reduce the operating current of the system. The following two modes are available.

(1) HALT mode

HALT instruction execution sets the HALT mode. In the HALT mode, the CPU operation clock is stopped. Oscillation of the system clock oscillator continues. If the low-speed internal oscillator is operating before the HALT mode is set, oscillation of the clock of the low-speed internal oscillator continues (refer to **Table 9-1**. Oscillation of the low-speed internal oscillation clock (whether it cannot be stopped or can be stopped by software) is set by the option byte). In this mode, the operating current is not decreased as much as in the STOP mode, but the HALT mode is effective for restarting operation immediately upon interrupt request generation and frequently carrying out intermittent operations.

(2) STOP mode

STOP instruction execution sets the STOP mode. In the STOP mode, the system clock oscillator stops, stopping the whole system, thereby considerably reducing the CPU operating current.

Because this mode can be cleared by an interrupt request, it enables intermittent operations to be carried out. However, select the HALT mode if processing must be immediately started by an interrupt request when the operation stop time[Note] is generated after the STOP mode is released.

Note The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

In either of these two modes, all the contents of registers, flags and data memory just before the standby mode is set are held. The I/O port output latches and output buffer statuses are also held.

Cautions 1. When shifting to the STOP mode, be sure to stop the peripheral hardware operation before executing STOP instruction (except the peripheral hardware that operates on the low-speed internal oscillation clock).

2. If the low-speed internal oscillator is operating before the STOP mode is set, oscillation of the low-speed internal oscillation clock cannot be stopped in the STOP mode (refer to Table 9-1).

9.2 Standby Function Operation

9.2.1 HALT mode

(1) HALT mode

The HALT mode is set by executing the HALT instruction.

The operating statuses in the HALT mode are shown below.

Caution Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag clear, the standby mode is immediately cleared if set.

Table 9-2. Operating Statuses in HALT Mode

Item \ Setting of HALT Mode		Low-Speed Internal Oscillator cannot be stopped[Note].	Low-Speed Internal Oscillator can be stopped[Note].	
			When Low-Speed Internal Oscillation Continues	When Low-Speed Internal Oscillation Stops
System clock		Clock supply to CPU is stopped.		
CPU		Operation stops.		
Port (latch)		Holds status before HALT mode was set.		
8-bit timer H1	Sets count clock to f_{XP} to $f_{XP}/2^{12}$	Operable		
	Sets count clock to $f_{RL}/2^7$	Operable	Operable	Operation stops.
Watchdog timer	“System clock” selected as operating clock	Setting disabled.	Operation stops.	
	“Low-speed internal oscillation clock” selected as operating clock	Operable (Operation continues)	Operation stops.	
Power-on-clear circuit		Always operates.		
Low-voltage detector		Operable		
External interrupt		Operable		

Note “Cannot be stopped” or “Stopped by software” is selected for low-speed internal oscillator by the option byte (for the option byte, see **CHAPTER 13 OPTION BYTE**).

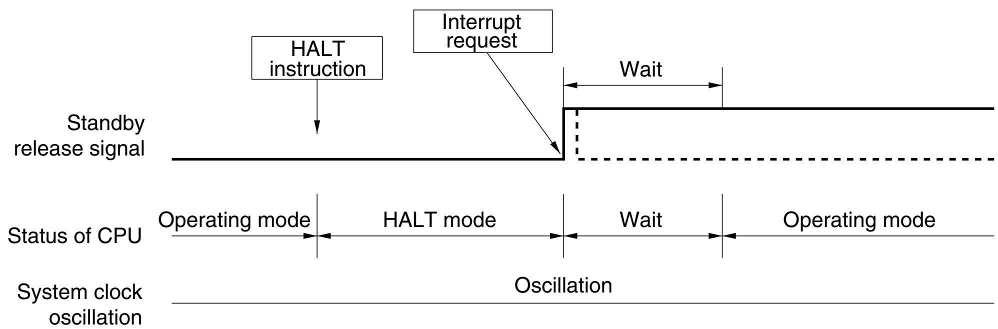
(2) HALT mode release

The HALT mode can be released by the following two sources.

(a) Release by unmasked interrupt request

When an unmasked interrupt request is generated, the HALT mode is released. If interrupt acknowledgement is enabled, vectored interrupt servicing is carried out. If interrupt acknowledgement is disabled, the next address instruction is executed.

Figure 9-1. HALT Mode Release by Interrupt Request Generation



Remarks 1. The broken lines indicate the case when the interrupt request which has released the standby mode is acknowledged.

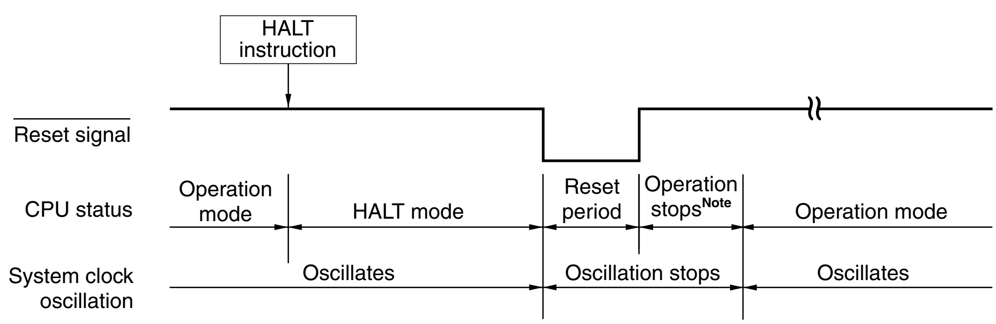
2. The wait time is as follows:

- When vectored interrupt servicing is carried out: 11 to 13 clocks
- When vectored interrupt servicing is not carried out: 3 to 5 clocks

(b) Release by reset signal generation

When the reset signal is input, HALT mode is released, and then, as in the case with a normal reset operation, the program is executed after branching to the reset vector address.

Figure 9-2. HALT Mode Release by Reset Signal Generation



Note Operation is stopped (277 μs (MIN.), 544 μs (TYP.), 1.075 ms (MAX.)) because the option byte is referenced.

Table 9-3. Operation in Response to Interrupt Request in HALT Mode

Release Source	MK××	IE	Operation
Maskable interrupt request	0	0	Next address instruction execution
	0	1	Interrupt servicing execution
	1	×	HALT mode held
Reset signal generation	–	×	Reset processing

×: don't care

9.2.2 STOP mode

(1) STOP mode setting and operating statuses

The STOP mode is set by executing the STOP instruction.

Caution Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. Thus, in the STOP mode, the normal operation mode is restored after the STOP instruction is executed and then the operation is stopped for the duration of 34 μs (TYP.).

The operating statuses in the STOP mode are shown below.

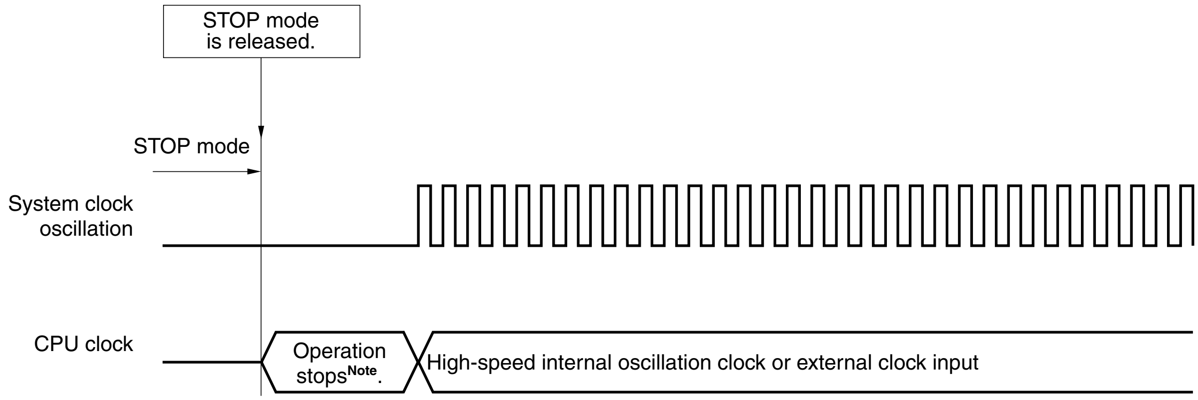
Table 9-4. Operating Statuses in STOP Mode

Item \ Setting of STOP Mode		Low-Speed Internal Oscillator cannot be stopped[Note].	Low-Speed Internal Oscillator can be stopped[Note].	
			When Low-Speed Internal Oscillation Continues	When Low-Speed Internal Oscillation Stops
System clock		Oscillation stops.		
CPU		Operation stops.		
Port (latch)		Holds status before STOP mode was set.		
8-bit timer H1	Sets count clock to f_{XP} to $f_{XP}/2^{12}$	Operation stops.		
	Sets count clock to $f_{RL}/2^7$	Operable	Operable	Operation stops.
Watchdog timer	"System clock" selected as operating clock	Setting disabled.	Operation stops.	
	"Low-speed internal oscillation clock" selected as operating clock	Operable (Operation continues)	Operation stops.	
Power-on-clear circuit		Always operates.		
Low-voltage detector		Operable		
External interrupt		Operable		

Note "Cannot be stopped" or "Stopped by software" is selected for low-speed internal oscillator by the option byte (for the option byte, see **CHAPTER 13 OPTION BYTE**).

(2) STOP mode release

Figure 9-3. Operation Timing When STOP Mode Is Released



Note The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

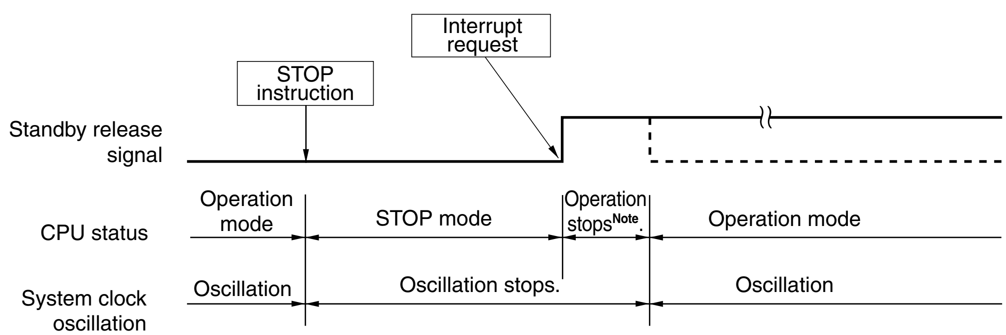
The STOP mode can be released by the following two sources.

(a) Release by unmasked interrupt request

When an unmasked interrupt request (8-bit timer H1, low-voltage detector, external interrupt request) is generated, the STOP mode is released. After the oscillation stabilization time has elapsed, if interrupt acknowledgment is enabled, vectored interrupt servicing is carried out. If interrupt acknowledgment is disabled, the next address instruction is executed.

Note Only when sets count clock to $f_{RL}/2^7$

Figure 9-4. STOP Mode Release by Interrupt Request Generation



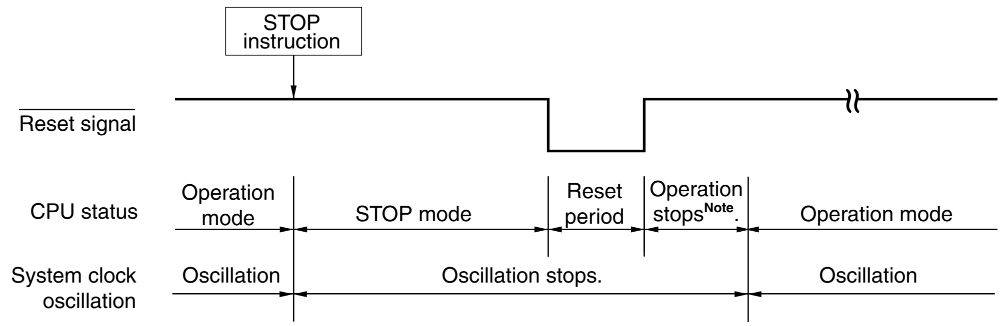
Note The operation stop time is 17 μs (MIN.), 34 μs (TYP.), and 67 μs (MAX.).

Remark The broken lines indicate the case when the interrupt request that has released the standby mode is acknowledged.

(b) Release by reset signal generation

When the reset signal is input, STOP mode is released and a reset operation is performed after the oscillation stabilization time has elapsed.

Figure 9-5. STOP Mode Release by Reset signal generation



Note Operation is stopped (277 μs (MIN.), 544 μs (TYP.), 1.075 ms (MAX.)) because the option byte is referenced.

Table 9-5. Operation in Response to Interrupt Request in STOP Mode

Release Source	MK××	IE	Operation
Maskable interrupt request	0	0	Next address instruction execution
	0	1	Interrupt servicing execution
	1	×	STOP mode held
Reset signal generation	–	×	Reset processing

×: don't care

CHAPTER 10 RESET FUNCTION

The following four operations are available to generate a reset signal.

(1) External reset input via $\overline{\text{RESET}}$ pin
(2) Internal reset by watchdog timer overflows
(3) Internal reset by comparison of supply voltage and detection voltage of power-on-clear (POC) circuit
(4) Internal reset by comparison of supply voltage and detection voltage of low-power-supply detector (LVI)

External and internal resets have no functional differences. In both cases, program execution starts from the programs at the address written in addresses 0000H and 0001H when the reset signal is generated.

A reset is applied when a low level is input to the $\overline{\text{RESET}}$ pin, the watchdog timer overflows, or by POC and LVI circuit voltage detection, and each item of hardware is set to the status shown in Table 10-1. Each pin is high impedance during reset signal generation or during the oscillation stabilization time just after reset release, except for P130, which is low-level output.

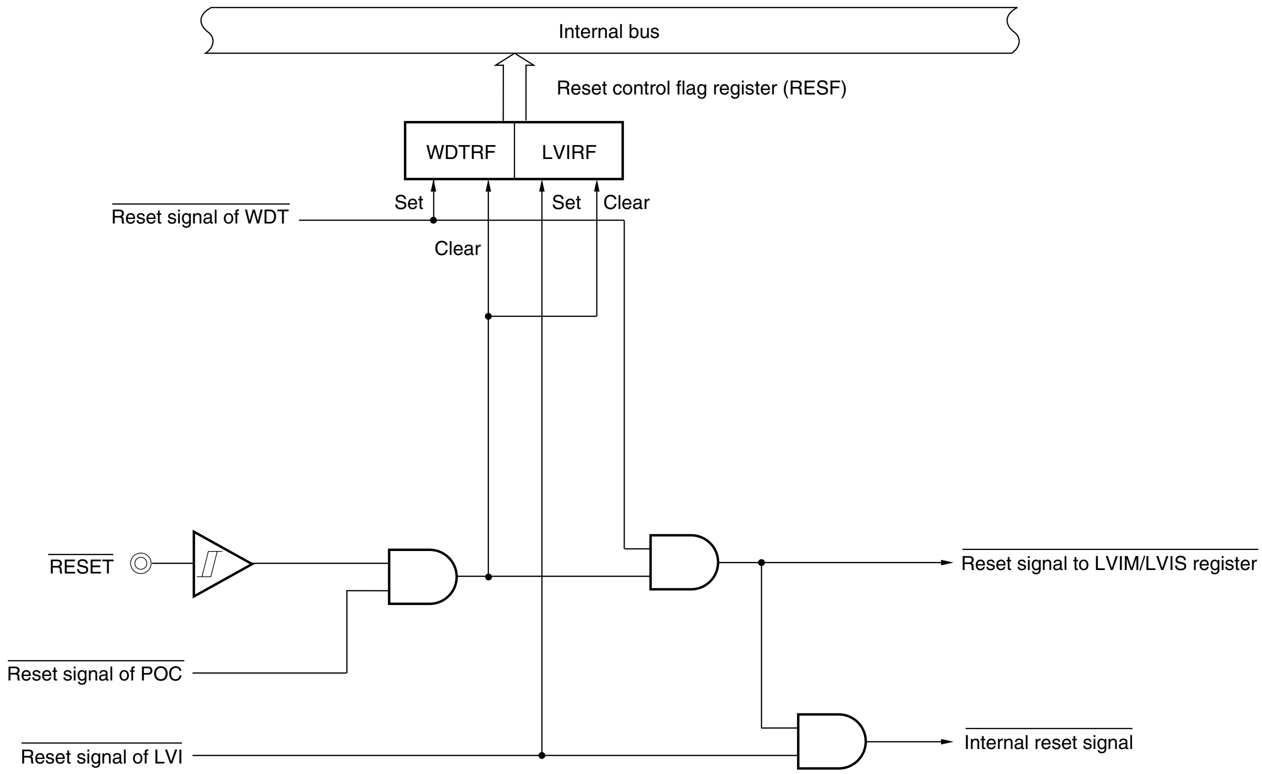
When a low level is input to the $\overline{\text{RESET}}$ pin, a reset occurs, and when a high level is input to the $\overline{\text{RESET}}$ pin, the reset is released and the CPU starts program execution after referencing the option byte. A reset generated by the watchdog timer source is automatically released after the reset, and the CPU starts program execution after referencing the option byte. (see **Figures 10-2** to **10-4**). Reset by POC and LVI circuit power supply detection is automatically released when $V_{DD} > V_{POC}$ or $V_{DD} > V_{LVI}$ after the reset, and the CPU starts program execution after referencing the option byte (see **CHAPTER 11 POWER-ON-CLEAR CIRCUIT** and **CHAPTER 12 LOW-VOLTAGE DETECTOR**).

Cautions 1. For an external reset, input a low level for 2 μs or more to the $\overline{\text{RESET}}$ pin.

2. During reset signal generation, the system clock and low-speed internal oscillation clock stop oscillating.

3. When the $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34), the μPD78F9500, 78F9501, 78F9502 are reset if a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the POC circuit and before the option byte is referenced again. The reset status is retained until a high level is input to the $\overline{\text{RESET}}$ pin.

Figure 10-1. Block Diagram of Reset Function

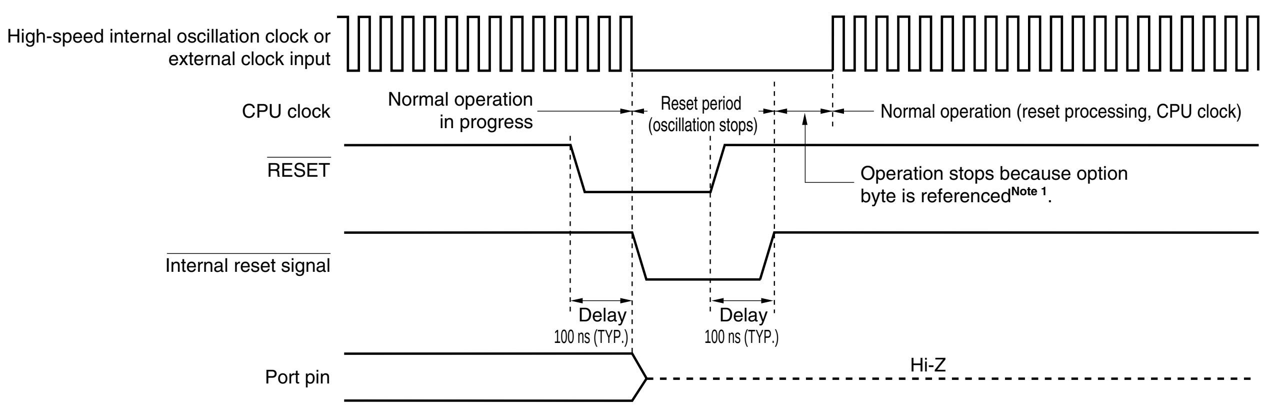


Caution The LVI circuit is not reset by the internal reset signal of the LVI circuit.

Remarks 1. LVIM: Low-voltage detect register

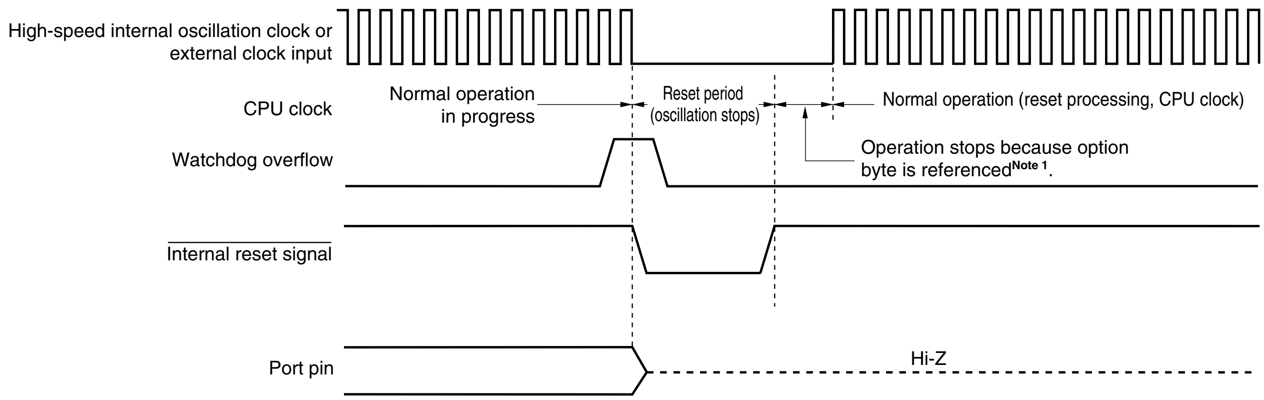
2. LVIS: Low-voltage detection level select register

Figure 10-2. Timing of Reset by RESET Input



Note The operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

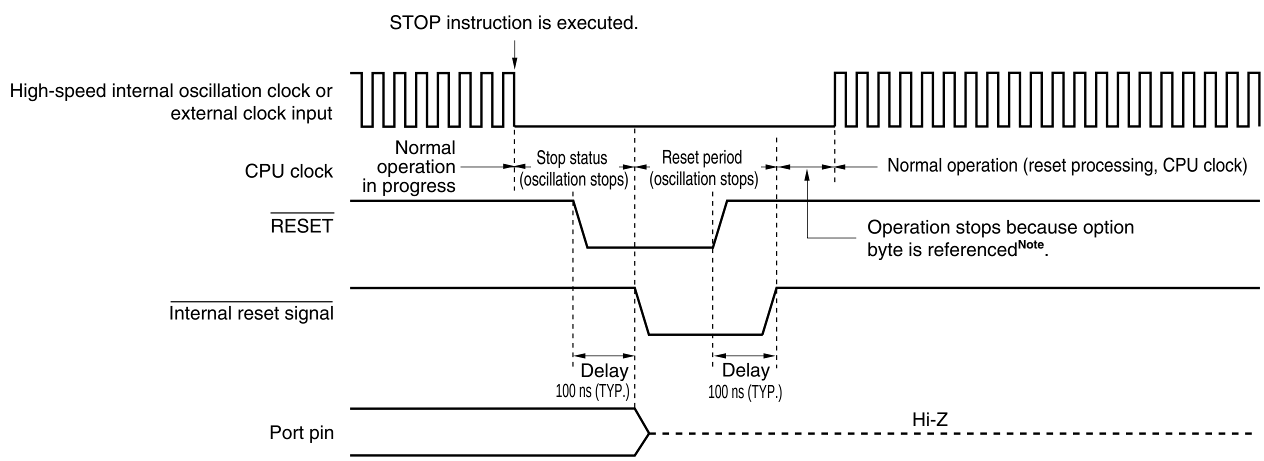
Figure 10-3. Timing of Reset by Overflow of Watchdog Timer



Note The operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

Caution The watchdog timer is also reset in the case of an internal reset of the watchdog timer.

Figure 10-4. Reset Timing by RESET Input in STOP Mode



Note The operation stop time is 277 μs (MIN.), 544 μs (TYP.), and 1.075 ms (MAX.).

Remark For the reset timing of the power-on-clear circuit and low-voltage detector, refer to **CHAPTER 11 POWER-ON-CLEAR CIRCUIT** and **CHAPTER 12 LOW-VOLTAGE DETECTOR**.

Table 10-1. Hardware Statuses After Reset Acknowledgment

Hardware		Status After Reset
Program counter (PC) Note 1		Contents of reset vector table (0000H and 0001H) are set.
Stack pointer (SP)		Undefined
Program status word (PSW)		02H
RAM	Data memory	Undefined Note 2
	General-purpose registers	Undefined Note 2
Ports (P2 to P4) (output latches)		00H
Port mode registers (PM2 to PM4)		FFH
Pull-up resistor option registers (PU2 to PU4)		00H
Processor clock control register (PCC)		02H
Preprocessor clock control register (PPCC)		02H
Low-speed internal oscillation mode register (LSRCM)		00H
8-bit timer H1	Compare registers (CMP01, CMP11)	00H
	Mode register 1 (TMHMD1)	00H
Watchdog timer	Mode register (WDTM)	67H
	Enable register (WDTE)	9AH
Reset function	Reset control flag register (RESF)	00H Note3
Low-voltage detector	Low-voltage detection register (LVIM)	00H Note3
	Low-voltage detection level select register (LVIS)	00H Note3
Interrupt	Request flag registers (IF0)	00H
	Mask flag registers (MK0)	FFH
	External interrupt mode registers (INTM0)	00H
Flash memory	Flash protect command register (PFCMD)	Undefined
	Flash status register (PFS)	00H
	Flash programming mode control register (FLPMC)	Undefined
	Flash programming command register (FLCMD)	00H
	Flash address pointer L (FLAPL)	Undefined
	Flash address pointer H (FLAPH)	
	Flash address pointer H compare register (FLAPHC)	00H
	Flash address pointer L compare register (FLAPLC)	00H
	Flash write buffer register (FLW)	00H

Notes 1. Only the contents of PC are undefined while reset signal generation and while the oscillation stabilization time elapses. The statuses of the other hardware units remain unchanged.

2. The status after reset is held in the standby mode.

Notes 3. These values change as follows depending on the reset source.

Register \ Reset Source		$\overline{\text{RESET}}$ Input	Reset by POC	Reset by WDT	Reset by LVI
RESF	WDTRF	Cleared (0)	Cleared (0)	Set (1)	Held
	LVIRF			Held	Set (1)
LVIM		Cleared (00H)	Cleared (00H)	Cleared (00H)	Held
LVIS					

10.1 Register for Confirming Reset Source

Many internal reset generation sources exist in the μPD78F9500, 78F9501, 78F9502. The reset control flag register (RESF) is used to store which source has generated the reset request.

RESF can be read by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input, reset signal generation by power-on-clear (POC) circuit, and reading RESF clear RESF to 00H.

Figure 10-5. Format of Reset Control Flag Register (RESF)

Address: FF54H After reset: 00H[Note] R

Symbol	7	6	5	4	3	2	1	0
RESF	0	0	0	WDTRF	0	0	0	LVIRF

WDTRF	Internal reset request by watchdog timer (WDT)
0	Internal reset request is not generated, or RESF is cleared.
1	Internal reset request is generated.

LVIRF	Internal reset request by low-voltage detector (LVI)
0	Internal reset request is not generated, or RESF is cleared.
1	Internal reset request is generated.

Note The value after reset varies depending on the reset source.

Caution Do not read data by a 1-bit memory manipulation instruction.

The status of RESF when a reset request is generated is shown in Table 10-2.

Table 10-2. RESF Status When Reset Request Is Generated

Flag \ Reset Source	$\overline{\text{RESET}}$ Input	Reset by POC	Reset by WDT	Reset by LVI
WDTRF	Cleared (0)	Cleared (0)	Set (1)	Held
LVIRF			Held	Set (1)

CHAPTER 11 POWER-ON-CLEAR CIRCUIT

11.1 Functions of Power-on-Clear Circuit

The power-on-clear circuit (POC) has the following functions.

- Generates internal reset signal at power on.
- Compares supply voltage (V_{DD}) and detection voltage (V_{POC} = 2.1 V ±0.1 V), and generates internal reset signal when $V_{DD} < V_{POC}$.
- Compares supply voltage (V_{DD}) and detection voltage (V_{POC} = 2.1 V ±0.1 V), and releases internal reset signal when $V_{DD} \geq V_{POC}$.

Cautions 1. If an internal reset signal is generated in the POC circuit, the reset control flag register (RESF) is cleared to 00H.

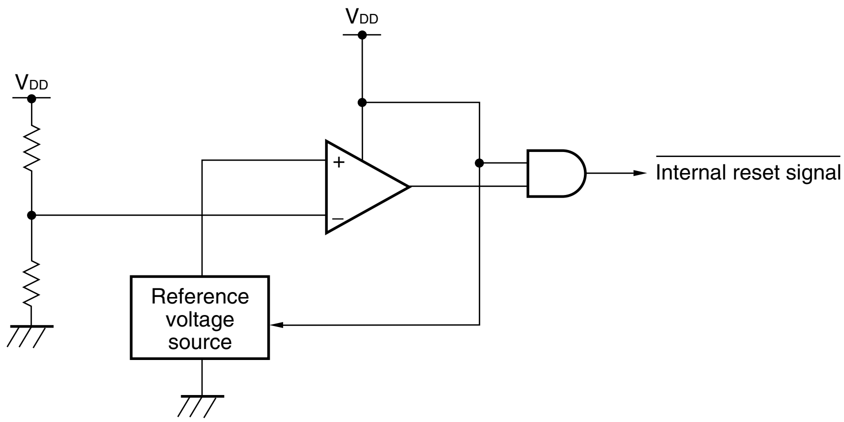
2. Because the detection voltage (V_{POC}) of the POC circuit is in a range of 2.1 V ±0.1 V, use a voltage in the range of 2.2 to 5.5 V.

Remark This product incorporates multiple hardware functions that generate an internal reset signal. A flag that indicates the reset cause is located in the reset control flag register (RESF) for when an internal reset signal is generated by the watchdog timer (WDT) or low-voltage-detection (LVI) circuit. RESF is not cleared to 00H and the flag is set to 1 when an internal reset signal is generated by WDT or LVI.
For details of RESF, see **CHAPTER 10 RESET FUNCTION**.

11.2 Configuration of Power-on-Clear Circuit

The block diagram of the power-on-clear circuit is shown in Figure 11-1.

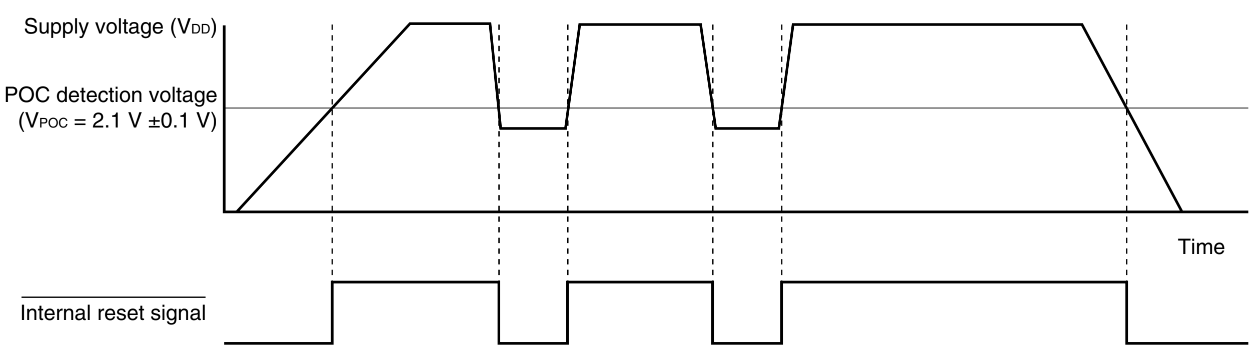
Figure 11-1. Block Diagram of Power-on-Clear Circuit



11.3 Operation of Power-on-Clear Circuit

In the power-on-clear circuit, the supply voltage (V_{DD}) and detection voltage (V_{POC} = 2.1 V ±0.1 V) are compared, and an internal reset signal is generated when $V_{DD} < V_{POC}$, and an internal reset is released when $V_{DD} \geq V_{POC}$.

Figure 11-2. Timing of Internal Reset Signal Generation in Power-on-Clear Circuit



11.4 Cautions for Power-on-Clear Circuit

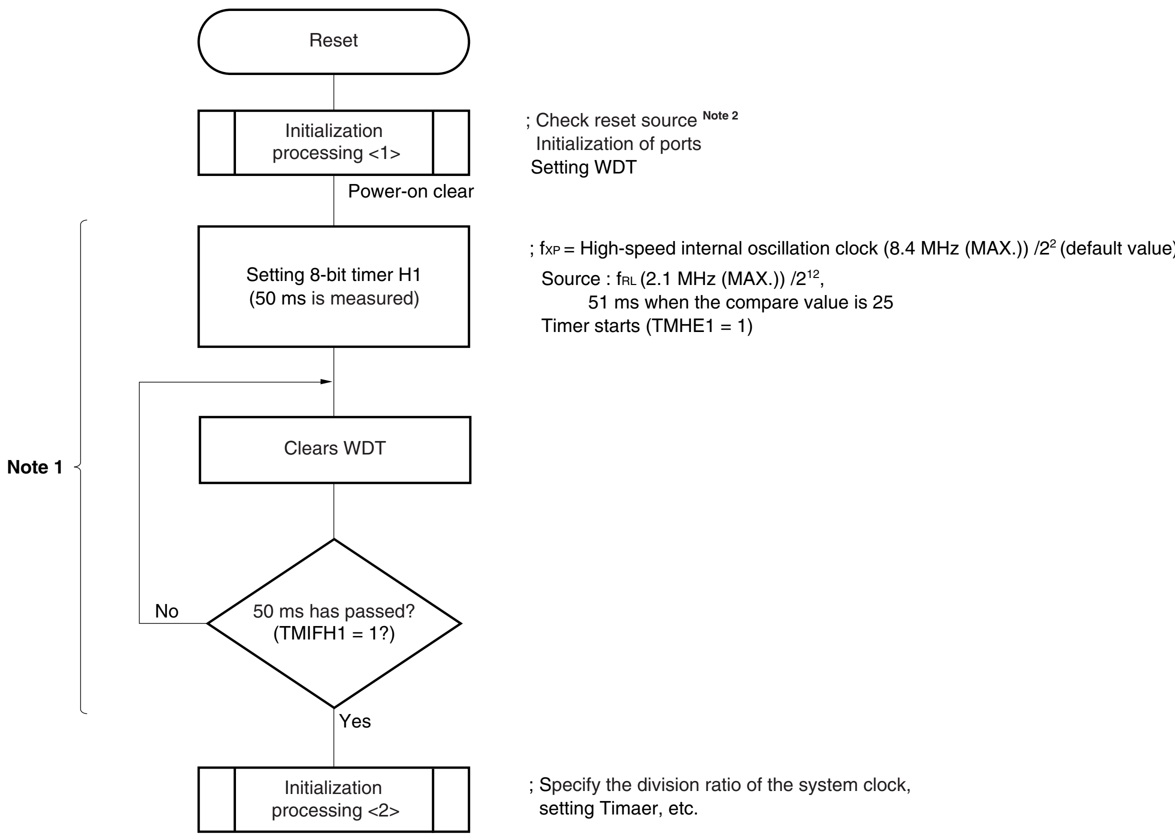
In a system where the supply voltage (V_{DD}) fluctuates for a certain period in the vicinity of the POC detection voltage (V_{POC}), the system may be repeatedly reset and released from the reset status. In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking the following action.

<Action>

After releasing the reset signal, wait for the supply voltage fluctuation period of each system by means of a software counter that uses a timer, and then initialize the ports.

Figure 11-3. Example of Software Processing After Release of Reset (1/2)

• If supply voltage fluctuation is 50 ms or less in vicinity of POC detection voltage

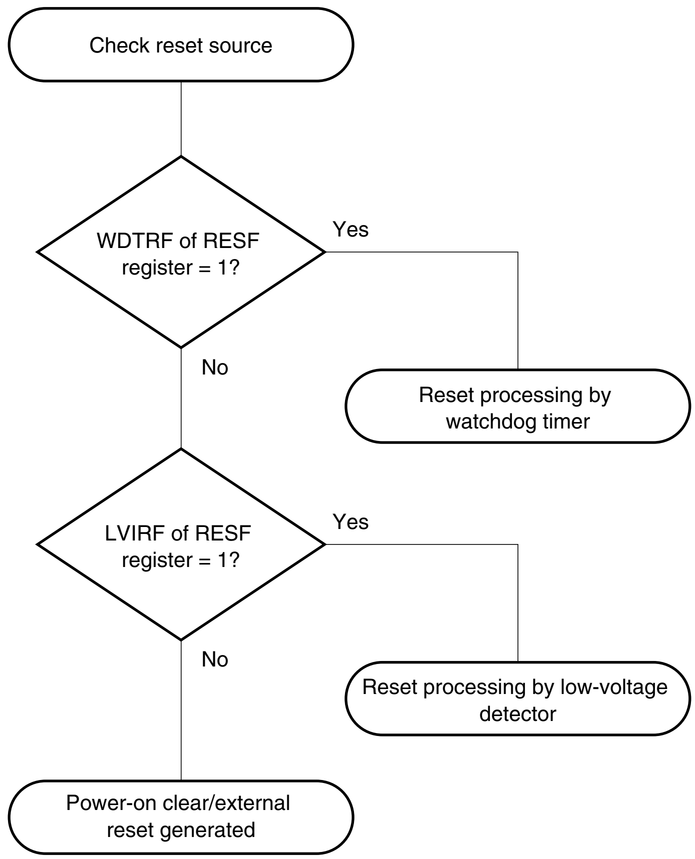


Notes 1. If reset is generated again during this period, initialization processing <2> is not started.

2. A flowchart is shown on the next page.

Figure 11-3. Example of Software Processing After Release of Reset (2/2)

• Checking reset cause



CHAPTER 12 LOW-VOLTAGE DETECTOR

12.1 Functions of Low-Voltage Detector

The low-voltage detector (LVI) has following functions.

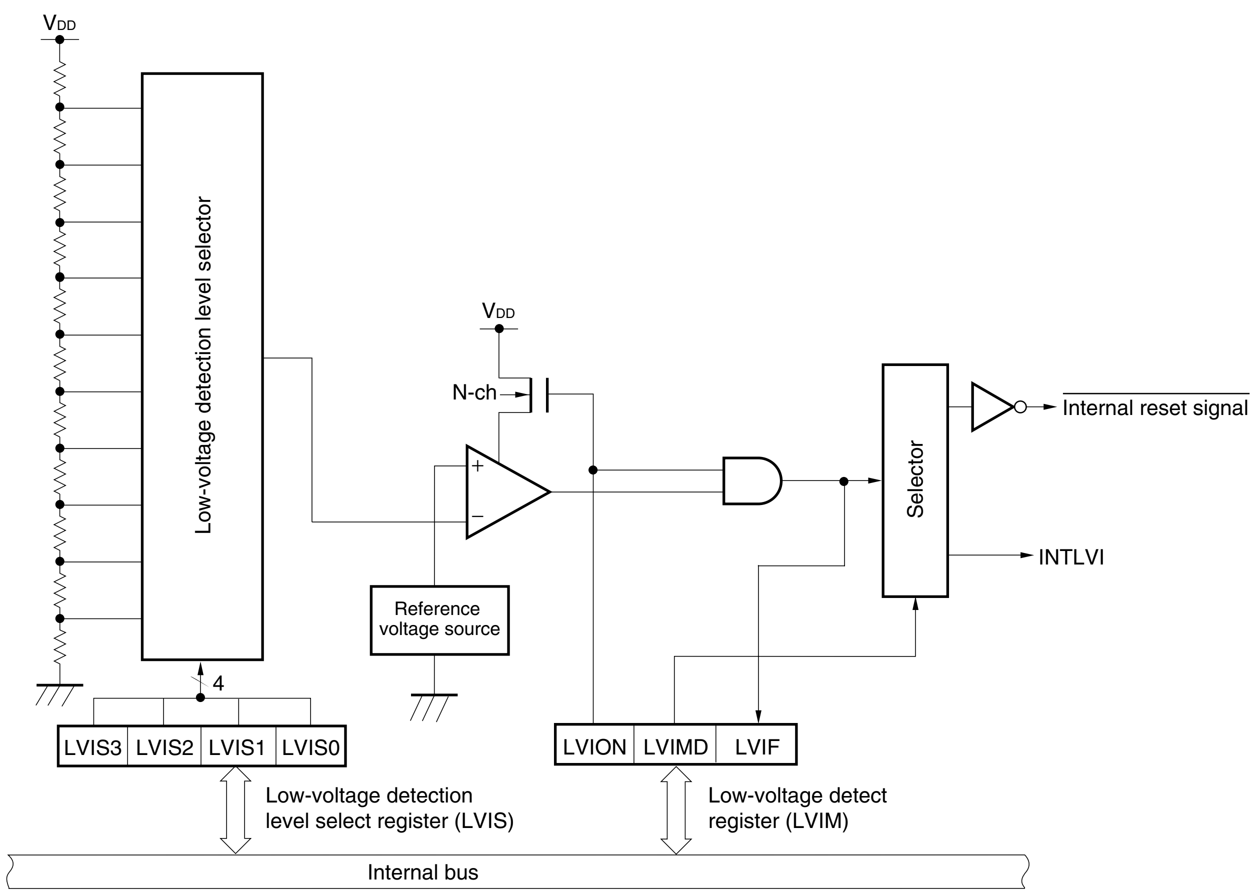
- Compares supply voltage (V_{DD}) and detection voltage (V_{LVI}), and generates an internal interrupt signal or internal reset signal when $V_{DD} < V_{LVI}$.
- Detection levels (ten levels) of supply voltage can be changed by software.
- Interrupt or reset function can be selected by software.
- Operable in STOP mode.

When the low-voltage detector is used to reset, bit 0 (LVIRF) of the reset control flag register (RESF) is set to 1 if reset occurs. For details of RESF, refer to **CHAPTER 10 RESET FUNCTION**.

12.2 Configuration of Low-Voltage Detector

The block diagram of the low-voltage detector is shown in Figure 12-1.

Figure 12-1. Block Diagram of Low-Voltage Detector



12.3 Registers Controlling Low-Voltage Detector

The low-voltage detector is controlled by the following registers.

- Low-voltage detect register (LVIM)
- Low-voltage detection level select register (LVIS)

(1) Low-voltage detect register (LVIM)

This register sets low-voltage detection and the operation mode.
This register can be set by a 1-bit or 8-bit memory manipulation instruction.
Reset signal generation clears this register to 00H[Note 1].

Figure 12-2. Format of Low-Voltage Detect Register (LVIM)

Address: FF50H After reset: 00H[Note 1] R/W[Note 2]

Symbol	<7>	6	5	4	3	2	<1>	<0>
LVIM	LVION	0	0	0	0	0	LVIMD	LVIF

LVION[Note 3]	Enabling low-voltage detection operation
0	Disable operation
1	Enable operation

LVIMD	Low-voltage detection operation mode selection
0	Generate interrupt signal when supply voltage (V_{DD}) < detection voltage (V_{LVI})
1	Generate internal reset signal when supply voltage (V_{DD}) < detection voltage (V_{LVI})

LVIF[Note 4]	Low-voltage detection flag
0	Supply voltage (V_{DD}) ≥ detection voltage (V_{LVI}), or when operation is disabled
1	Supply voltage (V_{DD}) < detection voltage (V_{LVI})

Notes 1. For a reset by LVI, the value of LVIM is not initialized.

2. Bit 0 is a read-only bit.

3. When LVION is set to 1, operation of the comparator in the LVI circuit is started. Use software to instigate a wait of at least 0.2 ms from when LVION is set to 1 until the voltage is confirmed at LVIF.

4. The value of LVIF is output as the interrupt request signal INTLVI when LVION = 1 and LVIMD = 0.

Cautions 1. To stop LVI, follow either of the procedures below.

- **When using 8-bit manipulation instruction: Write 00H to LVIM.**
- **When using 1-bit memory manipulation instruction: Clear LVION to 0.**

2. Be sure to set bits 2 to 6 to 0.

(2) Low-voltage detection level select register (LVIS)

This register selects the low-voltage detection level.

This register can be set by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H[Note].

Figure 12-3. Format of Low-Voltage Detection Level Select Register (LVIS)

Address: FF51H, After reset: 00H[Note] R/W

Symbol	7	6	5	4	3	2	1	0
LVIS	0	0	0	0	LVIS3	LVIS2	LVIS1	LVIS0

LVIS3	LVIS2	LVIS1	LVIS0	Detection level
0	0	0	0	V_{LVI0} (4.3 V ±0.2 V)
0	0	0	1	V_{LVI1} (4.1 V ±0.2 V)
0	0	1	0	V_{LVI2} (3.9 V ±0.2 V)
0	0	1	1	V_{LVI3} (3.7 V ±0.2 V)
0	1	0	0	V_{LVI4} (3.5 V ±0.2 V)
0	1	0	1	V_{LVI5} (3.3 V ±0.15 V)
0	1	1	0	V_{LVI6} (3.1 V ±0.15 V)
0	1	1	1	V_{LVI7} (2.85 V ±0.15 V)
1	0	0	0	V_{LVI8} (2.6 V ±0.1 V)
1	0	0	1	V_{LVI9} (2.35 V ±0.1 V)
Other than above				Setting prohibited

Note For a reset by LVI, the value of LVIS is not initialized.

Caution 1. Bits 4 to 7 must be set to 0.

2. If a value other than the above is written during LVI operation, the value becomes undefined at the very moment it is written, and thus be sure to stop LVI (bit 7(LVION) = 0 on the LVIM register) before writing.

12.4 Operation of Low-Voltage Detector

The low-voltage detector can be used in the following two modes.

- Used as reset
  Compares the supply voltage (V_{DD}) and detection voltage (V_{LVI}), and generates an internal reset signal when $V_{DD} < V_{LVI}$, and releases internal reset when $V_{DD} \geq V_{LVI}$.

- Used as interrupt
  Compares the supply voltage (V_{DD}) and detection voltage (V_{LVI}), and generates an interrupt signal (INTLVI) when $V_{DD} < V_{LVI}$.

The operation is set as follows.

(1) When used as reset

- When starting operation

<1> Mask the LVI interrupt (LVIMK = 1).

<2> Set the detection voltage using bits 3 to 0 (LVIS3 to LVIS0) of the low-voltage detection level select register (LVIS).

<3> Set bit 7 (LVION) of LVIM to 1 (enables LVI operation).

<4> Use software to instigate a wait of at least 0.2 ms.

<5> Wait until “supply voltage (V_{DD}) ≥ detection voltage (V_{LVI})” at bit 0 (LVIF) of LVIM is confirmed.

<6> Set bit 1 (LVIMD) of LVIM to 1 (generates internal reset signal when supply voltage (V_{DD}) < detection voltage (V_{LVI})).

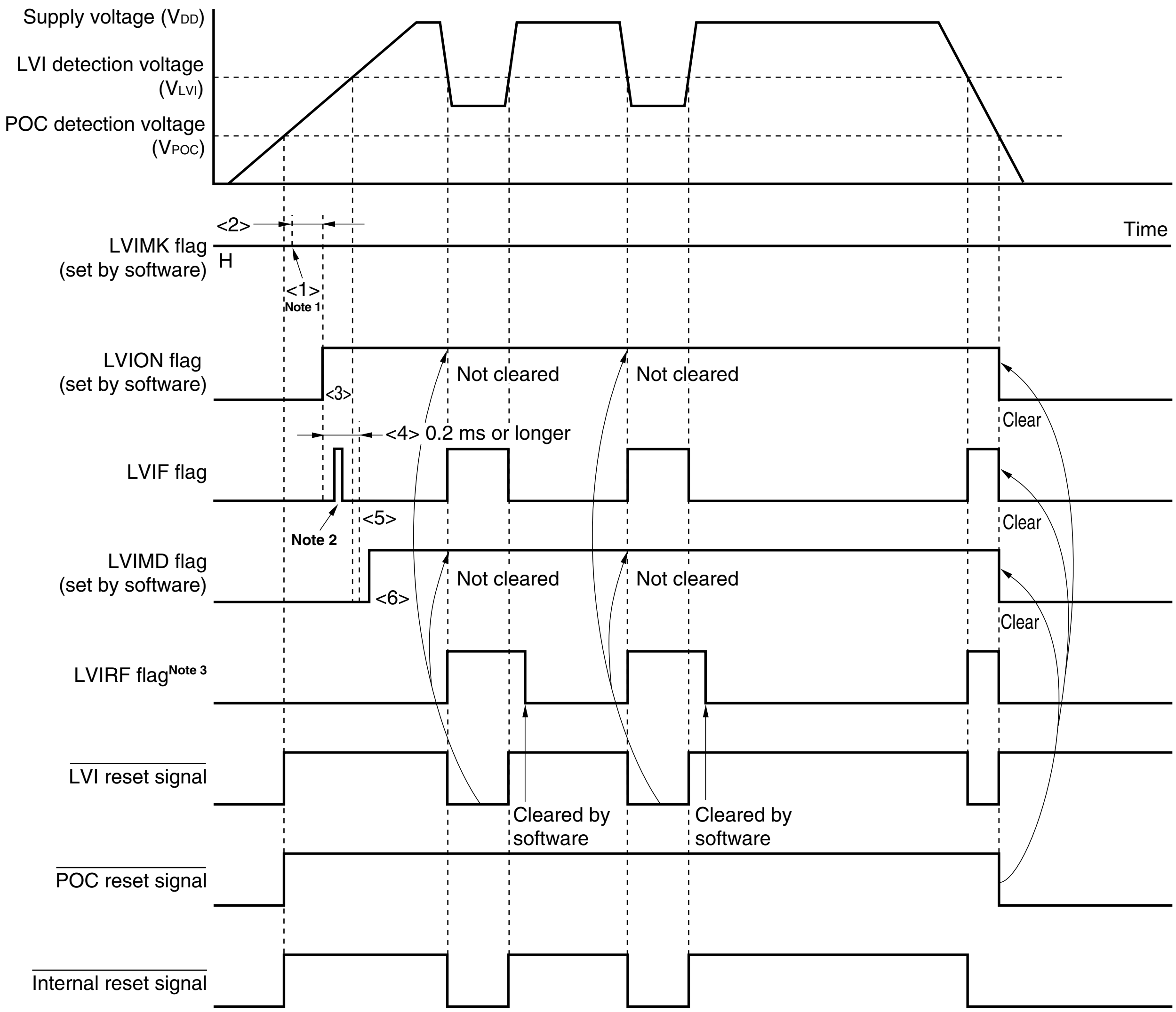
Figure 12-4 shows the timing of generating the internal reset signal of the low-voltage detector. Numbers <1> to <6> in this figure correspond to <1> to <6> above.

Cautions 1. <1> must always be executed. When LVIMK = 0, an interrupt may occur immediately after the processing in <3>.

2. If supply voltage (V_{DD}) ≥ detection voltage (V_{LVI}) when LVIMD is set to 1, an internal reset signal is not generated.

- When stopping operation
  Either of the following procedures must be executed.
  - When using 8-bit memory manipulation instruction: Write 00H to LVIM.
  - When using 1-bit memory manipulation instruction: Clear LVIMD to 0 and LVION to 0 in that order.

Figure 12-4. Timing of Low-Voltage Detector Internal Reset Signal Generation



Notes 1. The LVIMK flag is set to “1” by reset signal generation.

2. The LVIF flag may be set (1).

3. LVIRF is bit 0 of the reset control flag register (RESF). For details of RESF, refer to **CHAPTER 10 RESET FUNCTION**.

Remark <1> to <6> in Figure 12-4 above correspond to <1> to <6> in the description of “when starting operation” in **12.4 (1) When used as reset**.

(2) When used as interrupt

- When starting operation

<1> Mask the LVI interrupt (LVIMK = 1).

<2> Set the detection voltage using bits 3 to 0 (LVIS3 to LVIS0) of the low-voltage detection level select register (LVIS).

<3> Set bit 7 (LVION) of LVIM to 1 (enables LVI operation).

<4> Use software to instigate a wait of at least 0.2 ms.

<5> Wait until “supply voltage (V_{DD}) ≥ detection voltage (V_{LVI})” at bit 0 (LVIF) of LVIM is confirmed.

<6> Clear the interrupt request flag of LVI (LVIIF) to 0.

<7> Release the interrupt mask flag of LVI (LVIMK).

<8> Execute the EI instruction (when vector interrupts are used).

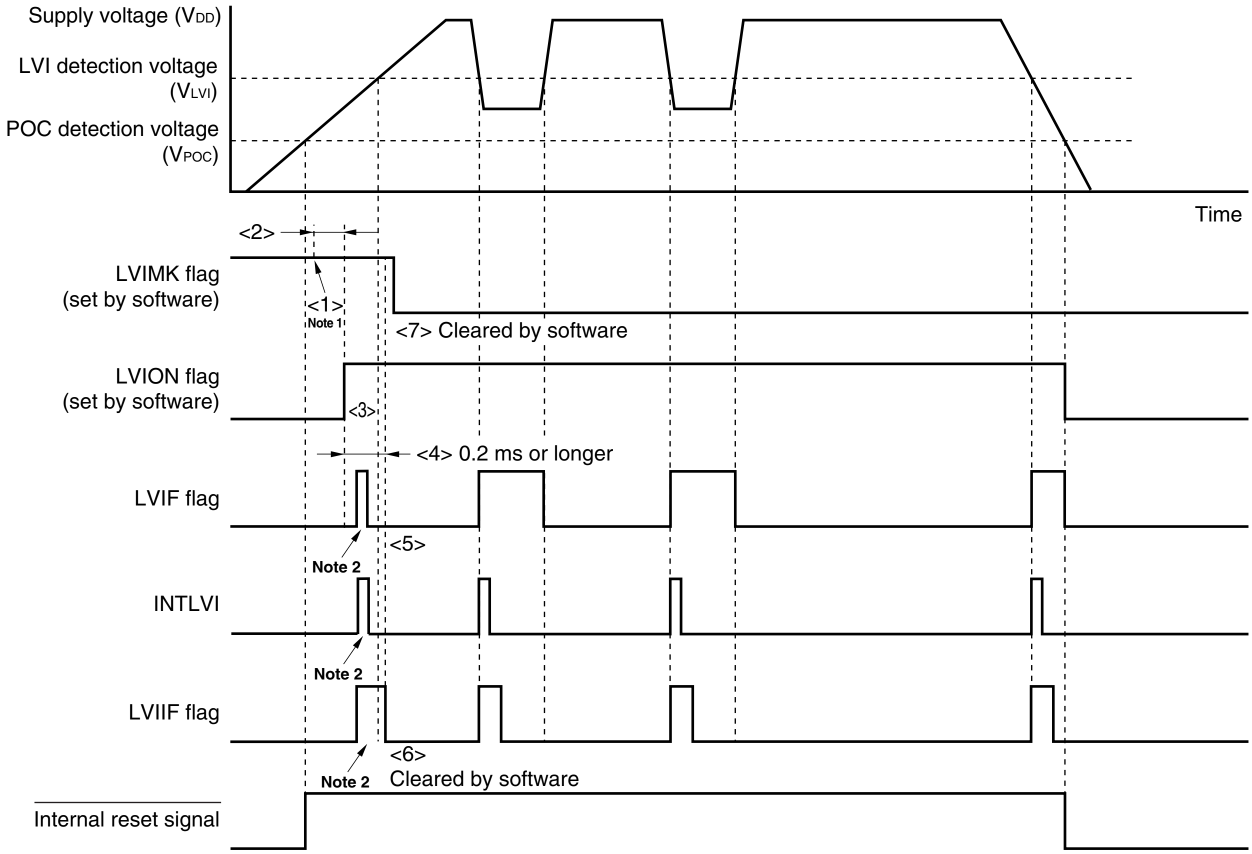
Figure 12-5 shows the timing of generating the interrupt signal of the low-voltage detector. Numbers <1> to <7> in this figure correspond to <1> to <7> above.

- When stopping operation

  Either of the following procedures must be executed.

  - When using 8-bit memory manipulation instruction: Write 00H to LVIM.
  - When using 1-bit memory manipulation instruction: Clear LVION to 0.

Figure 12-5. Timing of Low-Voltage Detector Interrupt Signal Generation



Notes 1. The LVIMK flag is set to “1” by reset signal generation.

2. An interrupt request signal (INTLVI) may be generated, and the LVIF and LVIIF flags may be set to 1.

Remark <1> to <7> in Figure 12-5 above correspond to <1> to <7> in the description of “when starting operation” in **12.4 (2) When used as interrupt**.

12.5 Cautions for Low-Voltage Detector

In a system where the supply voltage (V_{DD}) fluctuates for a certain period in the vicinity of the LVI detection voltage (V_{LVI}), the operation is as follows depending on how the low-voltage detector is used.

<1> When used as reset

The system may be repeatedly reset and released from the reset status.

In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking action (1) below.

<2> When used as interrupt

Interrupt requests may be frequently generated. Take (b) of action (2) below.

In this system, take the following actions.

<Action>

(1) When used as reset

After releasing the reset signal, wait for the supply voltage fluctuation period of each system by means of a software counter that uses a timer, and then initialize the ports (see **Figure 12-6**).

(2) When used as interrupt

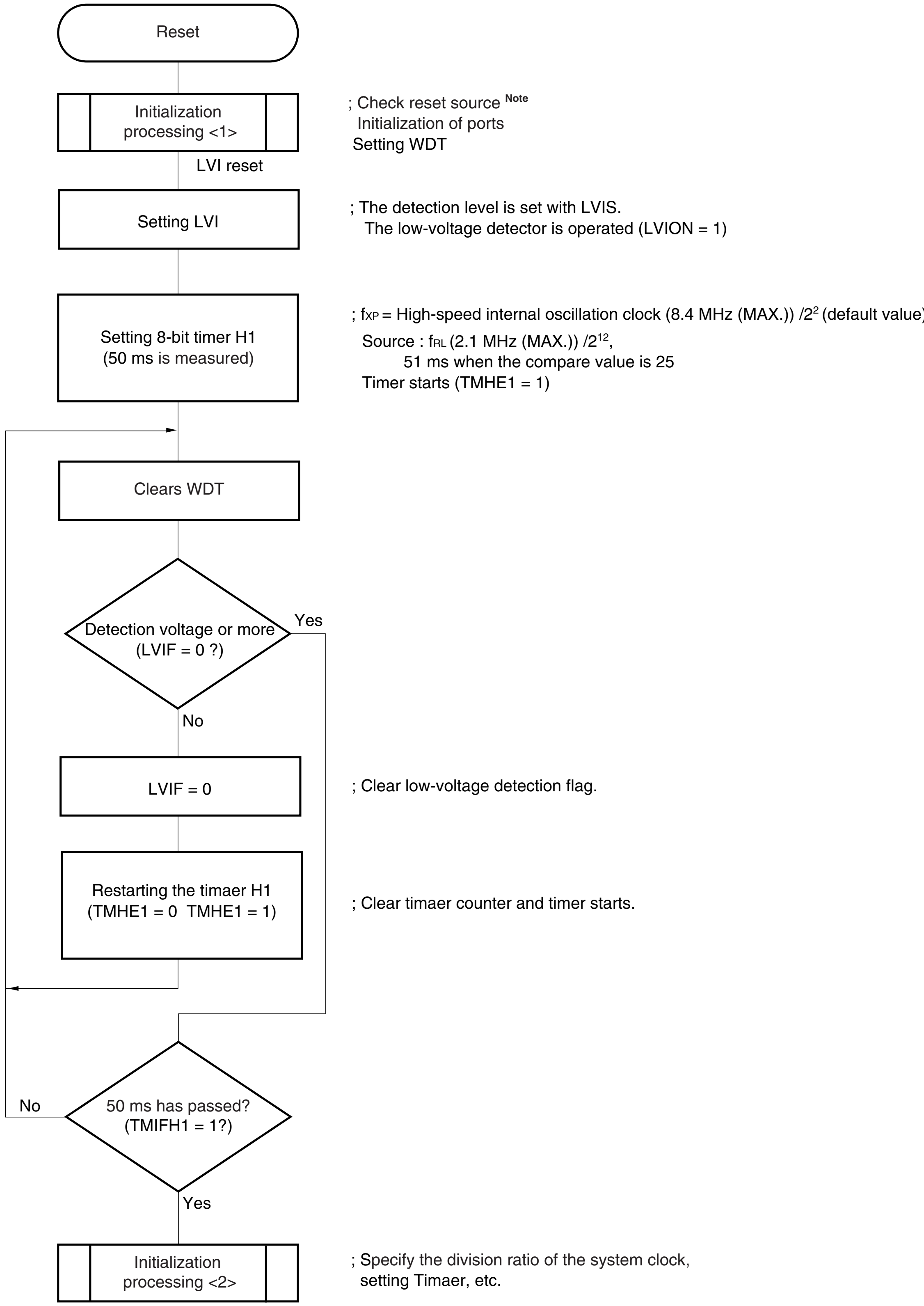
(a) Perform the processing[Note] for low voltage detection. Check that “supply voltage (V_{DD}) ≥ detection voltage (V_{LVI})” in the servicing routine of the LVI interrupt by using bit 0 (LVIF) of the low-voltage detection register (LVIM). Clear bit 1 (LVIIF) of interrupt request flag register 0 (IF0) to 0.

(b) In a system where the supply voltage fluctuation period is long in the vicinity of the LVI detection voltage, wait for the supply voltage fluctuation period, check that “supply voltage (V_{DD}) ≥ detection voltage (V_{LVI})” using the LVIF flag and clear LVIIF flag to 0.

Note For low voltage detection processing, the CPU clock speed is switched to slow speed, etc.

Figure 12-6. Example of Software Processing After Release of Reset (1/2)

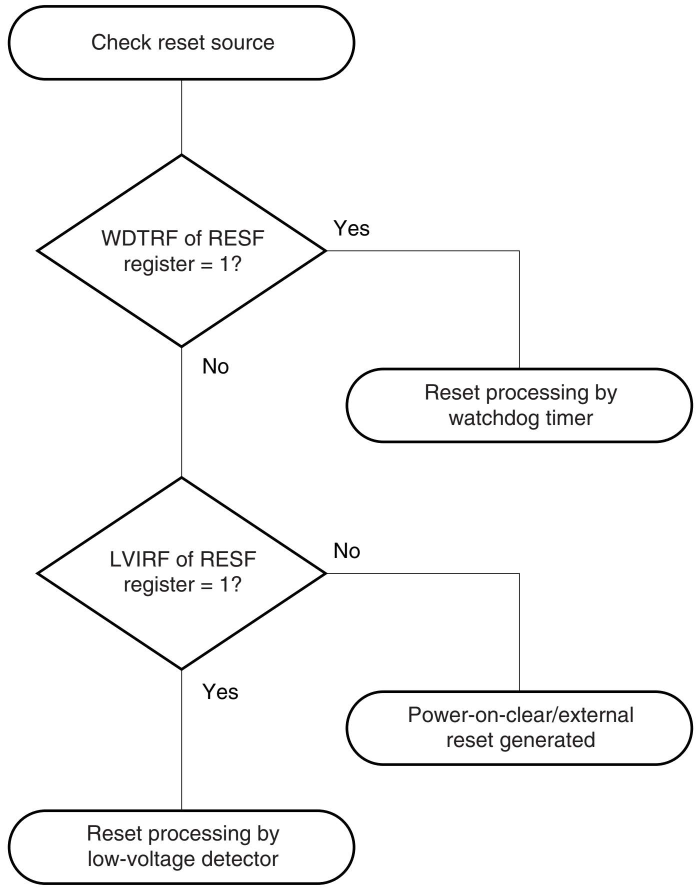
- If supply voltage fluctuation is 50 ms or less in vicinity of LVI detection voltage



Note A flowchart is shown on the next page.

Figure 12-6. Example of Software Processing After Release of Reset (2/2)

- Checking reset source



CHAPTER 13 OPTION BYTE

13.1 Functions of Option Byte

The address 0080H of the flash memory of the μPD78F9500, 78F9501, 78F9502 is an option byte area. When power is supplied or when starting after a reset, the option byte is automatically referenced, and settings for the specified functions are performed. When using the product, be sure to set the following functions by using the option byte.

(1) Selection of system clock source

- High-speed internal oscillation clock
- External clock input

(2) Low-speed internal oscillation clock oscillation

- Cannot be stopped.
- Can be stopped by software.

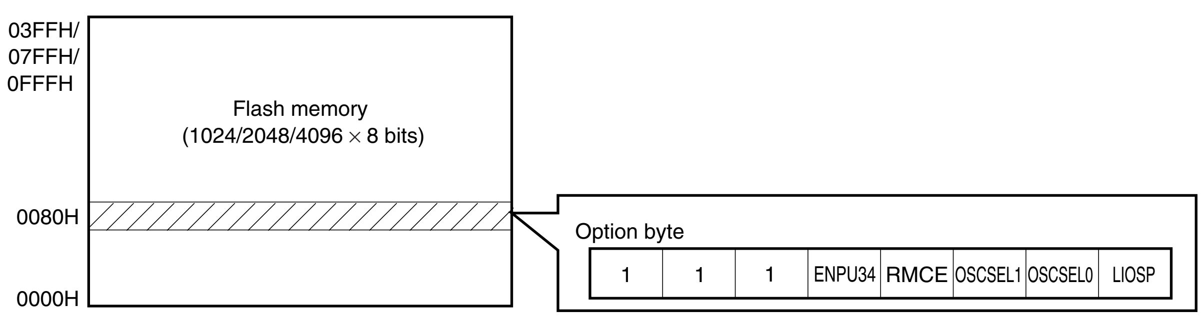
(3) Control of $\overline{\text{RESET}}$ pin

- Used as $\overline{\text{RESET}}$ pin
- $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34) (see **13.3 Caution When the RESET Pin Is Used as an Inport-Only Port Pin (P34)**).
- The on-chip pull-up resistor on $\overline{\text{RESET}}$ pin is selected, or $\overline{\text{RESET}}$ pin is set open.

(4) Oscillation stabilization time on power application or after reset release

- 2^{10}/fx
- 2^{12}/fx
- 2^{15}/fx
- 2^{17}/fx

Figure 13-1. Positioning of Option Byte



13.2 Format of Option Byte

Format of option bytes is shown below.

Figure 13-2. Format of Option Byte (1/2)

Address: 0080H

7	6	5	4	3	2	1	0
1	1	1	ENPU34	RMCE	OSCSEL1	OSCSEL0	LIOCP

ENPU34	Selection of on-chip pull-up resistor on $\overline{\text{RESET}}$ pin
1	On-chip pull-up resistor on $\overline{\text{RESET}}$ pin is selected.
0	On-chip pull-up resistor on $\overline{\text{RESET}}$ pin is not selected.

Remark When used as $\overline{\text{RESET}}$ pin, the pin can be left open by setting ENPU34 to "1" while the pin is not in use.

RMCE	Control of $\overline{\text{RESET}}$ pin
1	$\overline{\text{RESET}}$ pin is used as is.
0	$\overline{\text{RESET}}$ pin is used as input port pin (P34).

Caution Because the option byte is referenced after reset release, if a low level is input to the $\overline{\text{RESET}}$ pin before the option byte is referenced, then the reset state is not released.
When used as an input-only port (P34), the setting of the on-chip pull-up resistor can be done by PU34 on PU3 register.

OSCSEL1	OSCSEL0	Selection of system clock source
0	0	Setting prohibited
0	1	External clock input
1	×	High-speed internal oscillation clock

Caution Because the EXCLK pin is also used as the P23 pin, the condition under which the EXCLK pin can be used differ depending on the selected system clock source.
(1) External clock input is selected
Because the pin is used as an external clock input pin, P23 cannot be used as an I/O port pin.
(2) High-speed internal oscillation clock is selected
P23 pin can be used as an I/O port pin.

Remark × : don't care

Figure 13-2. Format of Option Byte (2/2)

LIOCP	Low-speed internal oscillates
1	Cannot be stopped (oscillation does not stop even if 1 is written to the LSRSTOP bit)
0	Can be stopped by software (oscillation stops when 1 is written to the LSRSTOP bit)

Cautions 1. If it is selected that low-speed internal oscillator cannot be stopped, the count clock to the watchdog timer (WDT) is fixed to low-speed internal oscillation clock.

2. If it is selected that low-speed internal oscillator can be stopped by software, supply of the count clock to WDT is stopped in the HALT/STOP mode, regardless of the setting of bit 0 (LSRSTOP) of the low-speed internal oscillation mode register (LSRCM). Similarly, clock supply is also stopped when a clock other than the low-speed internal oscillation clock is selected as a count clock to WDT.
While the low-speed internal oscillator is operating (LSRSTOP = 0), the clock can be supplied to the 8-bit timer H1 even in the STOP mode.

Remarks 1. ( ): f_X = 10 MHz

2. For the oscillation stabilization time of the resonator, refer to the characteristics of the resonator to be used.

3. An example of software coding for setting the option bytes is shown below.

```
OPB CSEG AT 0080H
DB 10010001B        ; Set to option byte
                    ; Low-speed internal oscillator cannot be stopped
                    ; The RESET pin is used as an input-only port pin (P34).
                    ; Minimum oscillation stabilization time (2^10/fx)
```

4. For details on the timing at which the option byte is referenced, see **CHAPTER 10 RESET FUNCTION**.

13.3 Caution When the $\overline{\text{RESET}}$ Pin Is Used as an Import-Only Port Pin (P34)

Be aware of the following when erasing/writing by on-board programming using a dedicated flash memory programmer once again on the already-written device which has been set as "The $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34)" by the option byte function.

Before supplying power to the target system, connect a dedicated flash memory programmer and turn its power on.

If the power is supplied to the target system beforehand, it cannot be switched to the flash memory programming mode.

CHAPTER 14 FLASH MEMORY

14.1 Features

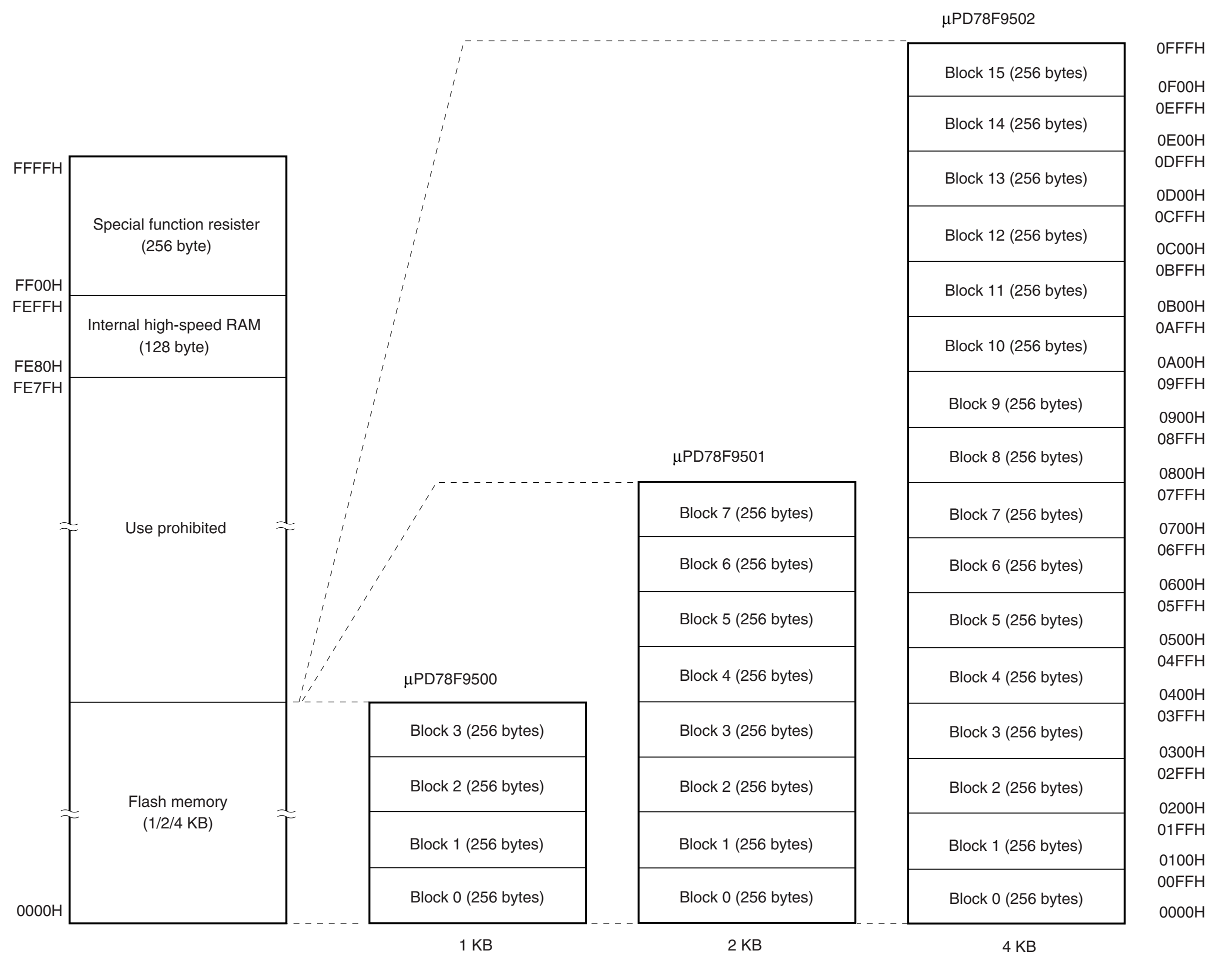
The internal flash memory of the μPD78F9500, 78F9501, 78F9502 has the following features.

- Erase/write even without preparing a separate dedicated power supply
- Capacity: 1/2/4 KB
  - Erase unit: 1 block (256 bytes)
  - Write unit: 1 block (at onboard/offboard programming time), 1 byte (at self programming time)
- Rewriting method
  - Rewriting by communication with dedicated flash memory programmer (on-board/off-board programming)
  - Rewriting flash memory by user program (self programming)
- Supports rewriting of the flash memory at onboard/offboard programming time through security functions
- Supports security functions in block units at self programming time through protect bytes

14.2 Memory Configuration

The 1/2/4 KB internal flash memory area is divided into 4/8/16 blocks and can be programmed/erased in block units. All the blocks can also be erased at once, by using a dedicated flash memory programmer.

Figure 14-1. Flash Memory Mapping



14.3 Functional Outline

The internal flash memory of the μPD78F9500, 78F9501, 78F9502 can be rewritten by using the rewrite function of the dedicated flash memory programmer, regardless of whether the μPD78F9500, 78F9501, 78F9502 have already been mounted on the target system or not (on-board/off-board programming).

The function for rewriting a program with the user program (self programming), which is ideal for an application when it is assumed that the program is changed after production/shipment of the target system, is provided.

Refer to Table 14-1 for the flash memory writing control function.

In addition, a security function that prohibits rewriting the user program written to the internal flash memory is also supported, so that the program cannot be changed by an unauthorized person.

Refer to **14.7.3 Security settings** for details on the security function.

Table 14-1. Rewrite Method

Rewrite Method	Functional Outline	Operation Mode
On-board programming	Flash memory can be rewritten after the device is mounted on the target system, by using a dedicated flash memory programmer.	Flash memory programming mode
Off-board programming	Flash memory can be rewritten before the device is mounted on the target system, by using a dedicated flash memory programmer and a dedicated program adapter board (FA series).	
Self programming	Flash memory can be rewritten by executing a user program that has been written to the flash memory in advance by means of on-board/off-board programming.	Self programming mode

Remarks 1. The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

2. Refer to the following sections for details on the flash memory writing control function.

- **14.7 On-Board and Off-Board Flash Memory Programming**
- **14.8 Flash Memory Programming by Self Writing**

14.4 Writing with Flash Memory Programmer

The following two types of dedicated flash memory programmers can be used for writing data to the internal flash memory of the μPD78F9500, 78F9501, 78F9502.

- FlashPro4 (PG-FP4, FL-PR4)

Data can be written to the flash memory on-board or off-board, by using a dedicated flash memory programmer.

(1) On-board programming

The contents of the flash memory can be rewritten after the μPD78F9500, 78F9501, 78F9502 have been mounted on the target system. The connectors that connect the dedicated flash memory programmer must be mounted on the target system.

(2) Off-board programming

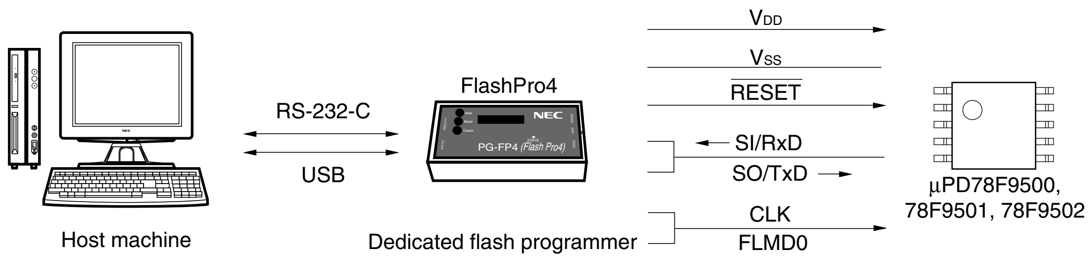
Data can be written to the flash memory with a dedicated program adapter (FA series) before the μPD78F9500, 78F9501, 78F9502 are mounted on the target system.

Remark The FL-PR4 and FA series are products of Naito Densei Machida Mfg. Co., Ltd.

14.5 Programming Environment

The environment required for writing a program to the flash memory is illustrated below.

Figure 14-2. Environment for Writing Program to Flash Memory (FlashPro4)



Note DGCLK is a clock for communication, while DGDATA is a transmit/receive signal for communication data.

A host machine that controls the dedicated flash memory programmer is necessary. When using the PG-FP4 or FL-PR4, data can be written with just the dedicated flash memory programmer after downloading the program from the host machine.

UART is used for manipulation such as writing and erasing when interfacing between the dedicated flash memory programmer and the µPD78F9500, 78F9501, 78F9502. To write the flash memory off-board, a dedicated program adapter (FA series) is necessary.

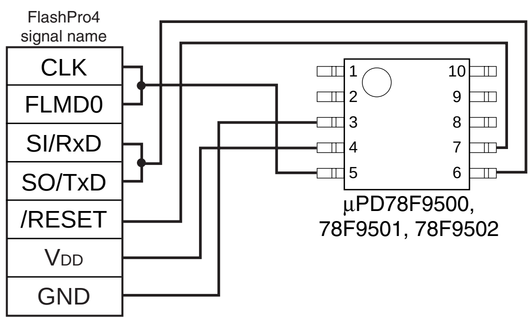
Download the latest programmer firmware, GUI, and parameter file from the download site for development tools (http://www.necel.com/micro/ods/jpn/index.html).

Table 14-2. Wiring Between μPD78F9500, 78F9501, 78F9502, and FlashPro4

FlashPro4 Connection Pin			μPD78F9500, 78F9501, 78F9502 Connection Pin	
Pin Name	I/O	Pin Function	Pin Name	Pin No.
CLK[Note]	Output	Clock to μPD78F9500, 78F9501, 78F9502	EXCLK/P23	5
FLMD0[Note]	Output	On-board mode signal		
SI/RxD[Note]	Input	Receive signal	P22	6
SO/TxD[Note]	Output	Receive signal/on-board mode signal		
/RESET	Output	Reset signal	$\overline{\text{RESET}}$/P34	7
V_{DD}	–	V_{DD} voltage generation/voltage monitor	V_{DD}	4
GND	–	Ground	V_{SS}	3

Note In the μPD78F9500, 78F9501, 78F9502, the CLK and FLMD0 signals are connected to the EXCLK pin; therefore, these signals need to be directly connected.

Figure 14-3. Wiring diagram with FlashPro4



14.6 Processing of Pins on Board

To write the flash memory on-board, connectors that connect the dedicated flash memory programmer must be provided on the target system. First provide a function that selects the normal operation mode or flash memory programming mode on the board.

When the flash memory programming mode is set, all the pins not used for programming the flash memory are in the same status as immediately after reset. Therefore, if the external device does not recognize the state immediately after reset, the pins must be processed as described below.

The state of the pins in the self programming mode is the same as that in the HALT mode.

14.6.1 EXCLK pin

The EXCLK pin is used as the serial interface of flash memory programming. Therefore, if the EXCLK pin is connected to an external device, a signal conflict occurs. To prevent the conflict of signals, isolate the connection with the external device.

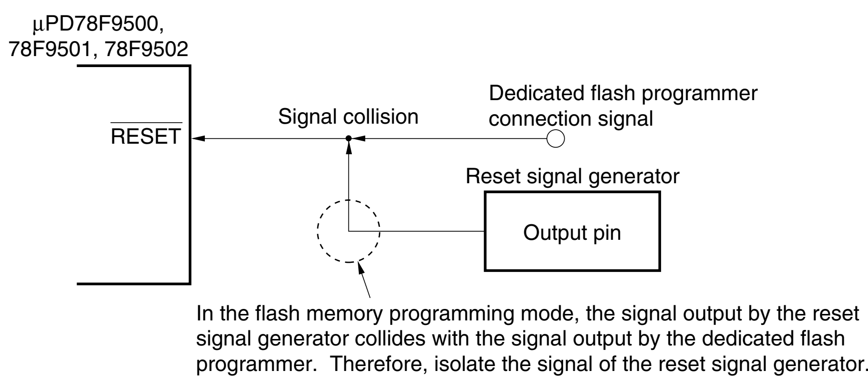
When connected a capacitor to the EXCLK pin, waveform at the time of communication is changed. Therefore there is a possibility that cannot communicate depending on capacitor capacitance. When perform flash memory programming, isolate connection with a condenser.

14.6.2 $\overline{\text{RESET}}$ pin

If the reset signal of the dedicated flash memory programmer is connected to the $\overline{\text{RESET}}$ pin that is connected to the reset signal generator on the board, signal collision takes place. To prevent this collision, isolate the connection with the reset signal generator.

If the reset signal is input from the user system while the flash memory programming mode is set, the flash memory will not be correctly programmed. Do not input any signal other than the reset signal of the dedicated flash memory programmer.

Figure 14-4. Signal Collision ($\overline{\text{RESET}}$ Pin)



14.6.3 Port pins

When the flash memory programming mode is set, all the pins not used for flash memory programming enter the same status as that immediately after reset. If external devices connected to the ports do not recognize the port status immediately after reset, the port pin must be connected to V_{DD} or V_{SS} via a resistor.

The state of the pins in the self programming mode is the same as that in the HALT mode.

14.6.4 Power supply

Connect the V_{DD} pin to V_{DD} of the flash memory programmer, and the V_{SS} pin to V_{SS} of the flash memory programmer.

14.7 On-Board and Off-Board Flash Memory Programming

14.7.1 Flash memory programming mode

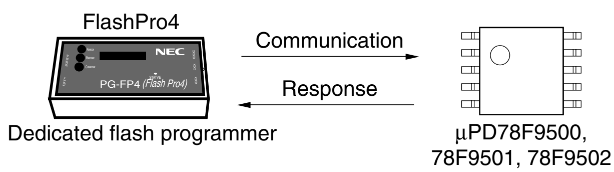
To rewrite the contents of the flash memory by using the dedicated flash memory programmer, set the μPD78F9500, 78F9501, 78F9502 in the flash memory programming mode. When the μPD78F9500, 78F9501, 78F9502 are connected to the flash memory programmer and a communication command is transmitted to the microcontroller, the microcontroller is set in the flash memory programming mode.

Change the mode by using a jumper when writing the flash memory on-board.

14.7.2 Communication commands

The dedicated flash memory programmer controls the μPD78F9500, 78F9501, 78F9502 by using commands. The signals sent from the flash memory programmer to the μPD78F9500, 78F9501, 78F9502 are called communication commands, and the commands sent from the μPD78F9500, 78F9501, 78F9502 to the dedicated flash memory programmer are called response.

Figure 14-5. Communication Commands



Communication commands are listed in the table below. All these communication commands are issued from the programmer and the μPD78F9500, 78F9501, 78F9502 perform processing corresponding to the respective communication commands.

Table 14-3. Communication Commands

Classification	Communication Command Name	Function
Erase	Batch erase (chip erase) command	Erases the contents of the entire memory
	Block erase command	Erases the contents of the memory of the specified block
Write	Write command	Writes to the specified address range and executes a verify check of the contents.
Checksum	Checksum command	Reads the checksum of the specified address range and compares with the written data.
Blank check	Blank check command	Confirms the erasure status of the entire memory.
Security	Security set command	Prohibits batch erase (chip erase) command, block erase command, and write command to prevent operation by third parties.

The μPD78F9500, 78F9501, 78F9502 return a response for the communication command issued by the dedicated flash memory programmer. The response names sent from the μPD78F9500, 78F9501, 78F9502 are listed below.

Table 14-4. Response Name

Command Name	Function
ACK	Acknowledges command/data.
NAK	Acknowledges illegal command/data.

14.7.3 Security settings

The operations shown below can be prohibited using the security setting command.

- Batch erase (chip erase) is prohibited
  Execution of the block erase and batch erase (chip erase) commands for entire blocks in the flash memory is prohibited. Once execution of the batch erase (chip erase) command is prohibited, all the prohibition settings can no longer be cancelled.

Caution After the security setting of the batch erase is set, erasure cannot be performed for the device. In addition, even if a write command is executed, data different from that which has already been written to the flash memory cannot be written because the erase command is disabled.

- Block erase is prohibited
  Execution of the block erase command in the flash memory is prohibited. This prohibition setting can be cancelled using the batch erase (chip erase) command.

- Write is prohibited
  Execution of the write and block erase commands for entire blocks in the flash memory is prohibited. This prohibition setting can be cancelled using the batch erase (chip erase) command.

Remark The security setting is valid when the programming mode is set next time.

The batch erase (chip erase), block erase, and write commands are enabled by the default setting when the flash memory is shipped. The above security settings are possible only for on-board/off-board programming. Each security setting can be used in combination.

Table 14-5 shows the relationship between the erase and write commands when the μPD78F9500, 78F9501, 78F9502 security function is enabled.

Table 14-5. Relationship Between Commands When Security Function Is Enabled

<table>
<tr><th>Command / Security</th><th>Batch Erase (Chip Erase) Command</th><th>Block Erase Command</th><th>Write Command</th></tr>
<tr><td>When batch erase (chip erase) security operation is enabled</td><td>Disabled</td><td rowspan="3">Disabled</td><td>Enabled[Note]</td></tr>
<tr><td>When block erase security operation is enabled</td><td rowspan="2">Enabled</td><td>Enabled</td></tr>
<tr><td>When write security operation is enabled</td><td>Disabled</td></tr>
</table>

Note Since the erase command is disabled, data different from that which has already been written to the flash memory cannot be written.

Table 14-6 shows the relationship between the security setting and the operation in each programming mode.

Table 14-6. Relationship Between Security Setting and Operation In Each Programming Mode

Programming Mode / Security Setting	On-Board/Off-Board Programming		Self Programming	
	Security Setting	Security Operation	Security Setting	Security Operation
Batch erase (chip erase)	Possible	Valid[Note 1]	Impossible	Invalid[Note 2]
Block erase				
Write				

Notes 1. Execution of each command is prohibited by the security setting.
2. Execution of self programming command is possible regardless of the security setting.

14.8 Flash Memory Programming by Self Writing

The μPD78F9500, 78F9501, 78F9502 support a self programming function that can be used to rewrite the flash memory via a user program, making it possible to upgrade programs in the field.

Caution Self programming processing must be included in the program before performing self writing.

Remarks 1. For usages of self programming, refer to use example mentioned in after 14.8.4.
2. To use the internal flash memory of the μPD78F9500, 78F9501, 78F9502 as the external EEPROM for storing data, refer to "78K0S/Kx1+ EEPROM Emulation Application Note" (U17379E).

14.8.1 Outline of self programming

To execute self programming, shift the mode from the normal operation of the user program (normal mode) to the self programming mode. Write/erase processing for the flash memory, which has been set to the register in advance, is performed by executing the HALT instruction during self programming mode. The HALT state is automatically released when processing is completed.

To shift to the self programming mode, execute a specific sequence for a specific register. Refer to **14.8.4 Example of shifting normal mode to self programming** for details.

Remark Data written by self programming can be referenced with the MOV instruction.

Table 14-7. Self Programming Mode

Mode	User Program Execution	Execution of Write/erase for Flash Memory with HALT Instruction
Normal mode	Enabled	–
Self programming mode	Enabled[Note]	Enabled

Note Maskable interrupt servicing is disabled during self programming mode.

Figure 14-6 shows a block diagram for self programming, Figure 14-7 shows the self programming state transition diagram, Table 14-8 lists the commands for controlling self programming.

Figure 14-6. Block Diagram of Self Programming

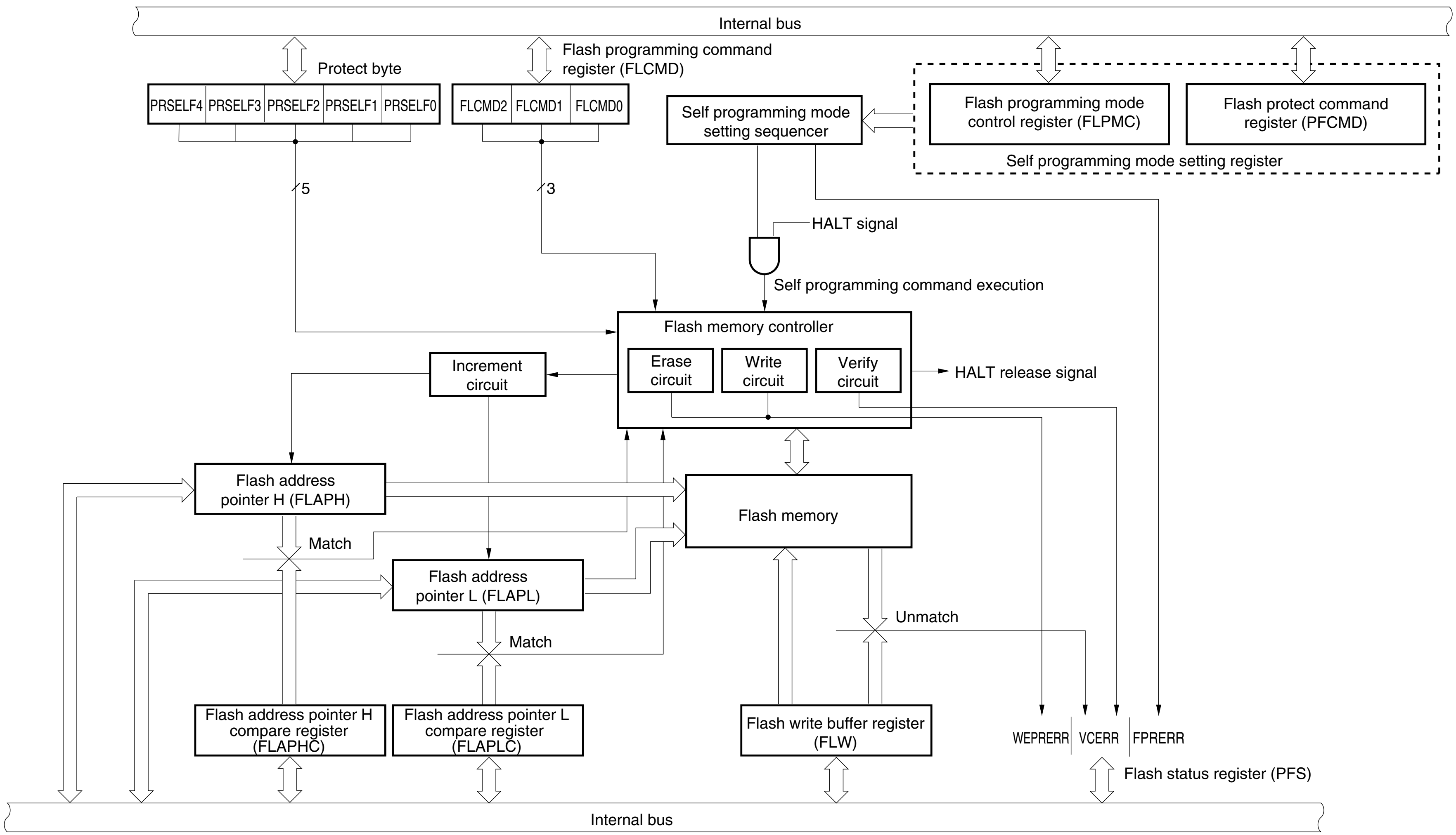


Figure 14-7. Self Programming State Transition Diagram

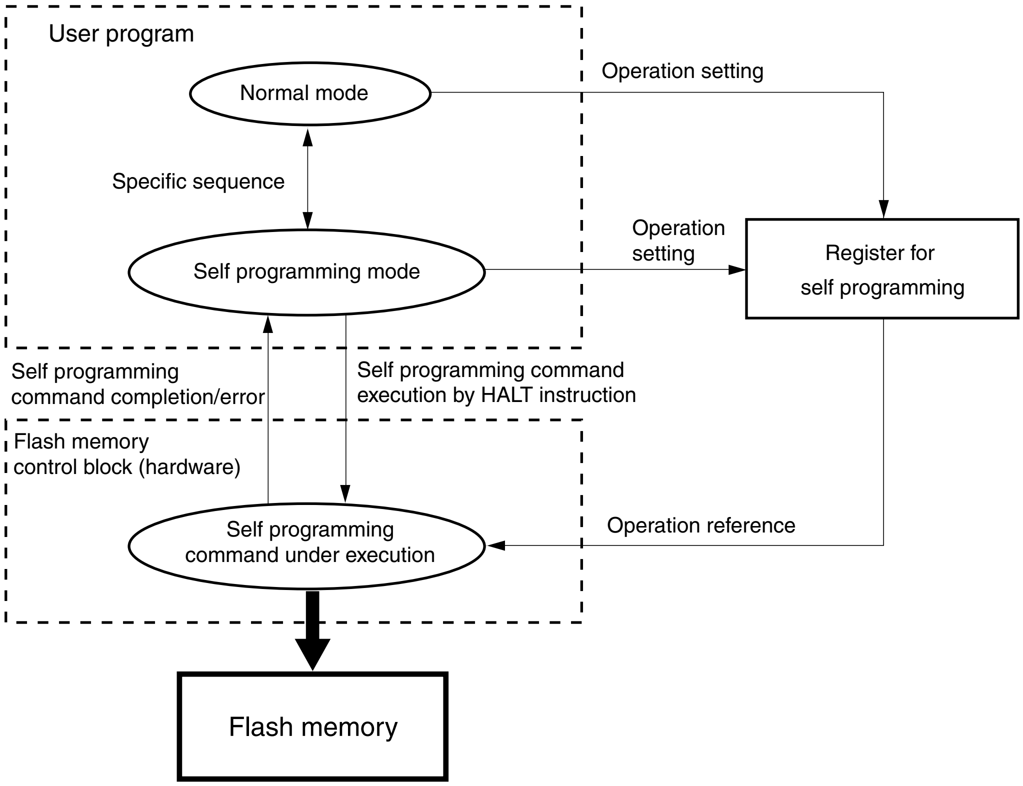


Table 14-8. Self Programming Controlling Commands

Command Name	Function	Time Taken from HALT Instruction Execution to Command Execution End
Internal verify 1	This command is used to check if data has been correctly written to the flash memory. It is used to check whether data has been written to an entire block.	Internal verify for 1 block (internal verify command executed once): 6.8 ms
Internal verify 2	This command is used to check if data has been correctly written to the flash memory. It is used to check whether data has been written in the same block.	Internal verify for 1 byte: 27 μs
Block erasure	This command is used to erase a specified block. Specify the block number before execution.	8.5 ms
Block blank check	This command is used to check if data in a specified block has been erased. Specify the block number, then execute this command.	480 μs
Byte write	This command is used to write 1-byte data to the specified address in the flash memory. Specify the write address and write data, then execute this command.	150 μs

Remark The command internal verify 1 can be executed by specifying an address in the same block but internal verify 2 is recommended if data is written to two or more addresses in the same block.

14.8.2 Cautions on self programming function

- No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 14-8 for the time taken for the execution of self programming.
- Interrupts that occur during self programming can be acknowledged after self programming mode ends. To avoid this operation, disable interrupt servicing (by setting MK0 to FFH, and executing the DI instruction) before a mode is shifted from the normal mode to the self programming mode with a specific sequence.
- RAM is not used while a self programming command is being executed.
- If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed.
- The value of the blank data set during block erasure is FFH.
- Set the CPU clock so that it is 1 MHz or more during self programming.
- Execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, then execute self programming. At this time, the HALT instruction is automatically released after 10 μs (MAX.) + 2 CPU clocks (f_{CPU}).
- If the clock of the oscillator or an external clock is selected as the system clock, execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, wait for 8 μs after releasing the HALT status, and then execute self programming.
- Check FPRERR using a 1-bit memory manipulation instruction.
- The state of the pins in self programming mode is the same as that in HALT mode.
- Since the security function set via on-board/off-board programming is disabled in self programming mode, the self programming command can be executed regardless of the security function setting. To disable write or erase processing during self programming, set the protect byte.
- Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command, there is a possibility that device does not operate normally.
- Clear the value of the FLCMD register to 00H immediately before setting self-programming mode and normal operation mode.

14.8.3 Registers used for self-programming function

The following registers are used for the self-programming function.

- Flash programming mode control register (FLPMC)
- Flash protect command register (PFCMD)
- Flash status register (PFS)
- Flash programming command register (FLCMD)
- Flash address pointers H and L (FLAPH and FLAPL)
- Flash address pointer H compare register and flash address pointer L compare register (FLAPHC and FLAPLC)
- Flash write buffer register (FLW)

The μPD78F9500, 78F9501, 78F9502 have an area called a protect byte at address 0081H of the flash memory.

(1) Flash programming mode control register (FLPMC)

This register is used to set the operation mode when data is written to the flash memory in the self-programming mode, and to read the set value of the protect byte.

Data can be written to FLPMC only in a specific sequence (refer to **14.8.3 (2) Flash protect command register (PFCMD)**) so that the application system does not stop by accident because of malfunction due to noise or program hang-up.

This register is set with an 8-bit memory manipulation instruction.
Reset signal generation makes the contents of this register undefined.

Figure 14-8. Format of Flash Programming Mode Control Register (FLPMC)

Address: FFA2H After reset: Undefined[Note 1] R/W[Note 2]

Symbol	7	6	5	4	3	2	1	0
FLPMC	0	PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	0	FLSPM

FLSPM	Selection of operation mode during self-programming mode
0	Normal mode This is the normal operation status. Executing the HALT instruction sets standby status.
1	Self-programming mode Self programming commands can be executed by executing the specific sequence to change modes while in normal mode. Set a command, an address, and data to be written, then execute the HALT instruction to execute self programming.

PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	The set value of the protect byte is read to these bits.

Notes 1. Bit 0 (FLSPM) is cleared to 0 when reset is released. The set value of the protect byte is read to bits 2 to 6 (PRSELF0 to PRSELF4) after reset is released.

2. Bits 2 to 6 (PRSELF0 to PRSELF4) are read-only.

Cautions 1. Cautions in the case of setting the self programming mode, refer to 14.8.2 Cautions on self programming function.

2. Set the CPU clock so that it is 1 MHz or more during self programming.

3. Execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, then execute self programming. At this time, the HALT instruction is automatically released after 10 μs (MAX.) + 2 CPU clocks (f_{CPU}).

4. If the clock of the oscillator or an external clock is selected as the system clock, execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, wait for 8 μs after releasing the HALT status, and then execute self programming.

5. Clear the value of the FLCMD register to 00H immediately before setting self-programming mode and normal operation mode.

(2) Flash protect command register (PFCMD)

If the application system stops inadvertently due to malfunction caused by noise or program hang-up, an operation to write the flash programming mode control register (FLPMC) may have a serious effect on the system. PFCMD is used to protect FLPMC from being written, so that the application system does not stop inadvertently.

Writing FLPMC is enabled only when a write operation is performed in the following specific sequence.

<1> Write a specific value to PFCMD (A5H)

<2> Write the value to be set to bit 0 (FLSPM) of the FLPMC (writing in this step is invalid)

<3> Write the inverted value of the value to be set to bit 0 (FLSPM) of the FLPMC (writing in this step is invalid)

<4> Write the value to be set to bit 0 (FLSPM) of the FLPMC (writing in this step is valid)

Caution Interrupt servicing cannot be executed in self-programming mode. Disable interrupt servicing (by executing the DI instruction while MK0 = FFH) before executing the specific sequence that sets self-programming mode and after executing the specific sequence that changes the mode to the normal mode.

This rewrites the value of the register, so that the register cannot be written illegally.

Occurrence of an illegal store operation can be checked by bit 0 (FPRERR) of the flash status register (PFS). Check FPRERR using a 1-bit memory manipulation instruction.

A5H must be written to PFCMD each time the value of FLPMC is changed.

PFCMD can be set by an 8-bit memory manipulation instruction.

Reset signal generation makes PFCMD undefined.

Figure 14-9. Format of Flash Protect Command Register (PFCMD)

Address: FFA0H After reset: Undefined W

Symbol	7	6	5	4	3	2	1	0
PFCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0

(3) Flash status register (PFS)

If data is not written to the flash programming mode control register (FLPMC), which is protected, in the correct sequence (writing the flash protect command register (PFCMD)), FLPMC is not written and a protection error occurs. If this happens, bit 0 of PFS (FPRERR) is set to 1.

When FPRERR is 1, it can be cleared to 0 by writing 0 to it.

Errors that may occur during self-programming are reflected in bit 1 (VCERR) and bit 2 (WEPRERR) of PFS. VCERR or WEPRERR can be cleared by writing 0 to them.

All the flags of the PFS register must be pre-cleared to 0 to check if the operation is performed correctly.

PFS can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears PFS to 00H.

Caution Check FPRERR using a 1-bit memory manipulation instruction.

Figure 14-10. Format of Flash Status Register (PFS)

Address: FFA1H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PFS	0	0	0	0	0	WEPRERR	VCERR	FPRERR

1. Operating conditions of FPRERR flag

<Setting conditions>

- If PFCMD is written when the store instruction operation recently performed on a peripheral register is not to write a specific value (A5H) to FLPMC
- If the first store instruction operation after <1> is on a peripheral register other than FLPMC
- If the first store instruction operation after <2> is on a peripheral register other than FLPMC

- If a value other than the inverted value of the value to be set to FLPMC is written by the first store instruction after <2>
- If the first store instruction operation after <3> is on a peripheral register other than FLPMC
- If a value other than the value to be set to FLPMC (value written in <2>) is written by the first store instruction after <3>

Remark The numbers in angle brackets above correspond to the those in **(2) Flash protect command register (PFCMD)**.

<Reset conditions>
- If 0 is written to the FPRERR flag
- If the reset signal is generation

2. Operating conditions of VCERR flag

<Setting conditions>
- Erasure verification error
- Internal writing verification error

If VCERR is set, it means that the flash memory has not been erased or written correctly. Erase or write the memory again in the specified procedure.

Remark The VCERR flag may also be set if an erase or write protect error occurs.

<Reset conditions>
- When 0 is written to the VCERR flag
- When the reset signal generation

3. Operating conditions of WEPRERR flag

<Setting conditions>
- If the area specified by the protect byte to be protected from erasing or writing is specified by the flash address pointer H (FLAPH) and a command is executed to this area
- If 1 is written to a bit that has not been erased (a bit for which the data is 0).

<Reset conditions>
- When 0 is written to the WEPRERR flag
- When the reset signal generation

(4) Flash programming command register (FLCMD)

This register is used to specify whether the flash memory is erased, written, or verified in the self-programming mode.

This register is set by using a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

Figure 14-11. Format of Flash Programming Command Register (FLCMD)

Address: FFA3H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
FLCMD	0	0	0	0	0	FLCMD2	FLCMD1	FLCMD0

FLCMD2	FLCMD1	FLCMD0	Command Name	Function
0	0	1	Internal verify 1	This command is used to check if data has been correctly written to the flash memory. It is used to check whether data has been written to an entire block. If an error occurs, bit 1 (VCERR) or bit 2 (WEPRERR) of the flash status register (PFS) is set to 1.
			Internal verify 2	This command is used to check if data has been correctly written to the flash memory. It is used to check whether data has been written in the same block. If an error occurs, bit 1 (VCERR) or bit 2 (WEPRERR) of the flash status register (PFS) is set to 1.
0	1	1	Block erase	This command is used to erase specified block. It is used both in the on-board mode and self-programming mode.
1	0	0	Block blank check	This command is used to check if the specified block has been erased.
1	0	1	Byte write	This command is used to write 1-byte data to the specified address in the flash memory. Specify the write address and write data, then execute this command. If 1 is written to a bit that has not been erased (a bit for which the data is 0), then bit 2 (WEPRERR) of the flash status register (PFS) becomes 1.
Other than above[Note]			Setting prohibited	

Note If any command other than those above is executed, command execution may immediately be terminated, and bit 1 or 2 (WEPRERR or VCERR) of the flash status register (PFS) may be set to 1.

(5) Flash address pointers H and L (FLAPH and FLAPL)

These registers are used to specify the start address of the flash memory when the memory is erased, written, or verified in the self-programming mode.

FLAPH and FLAPL consist of counters, and they are incremented until the values match with those of FLAPHC and FLAPLC when the programming command is not executed. When the programming command is executed, therefore, set the value again.

These registers are set with a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation makes these registers undefined.

Figure 14-12. Format of Flash Address Pointer H/L (FLAPH/FLAPL)

Address: FFA4H, FFA5H After reset: 00H R/W

FLAPH (FFA5H)								FLAPL (FFA4H)							
0	0	0	0	FLAP11	FLAP10	FLAP9	FLAP8	FLAP7	FLAP6	FLAP5	FLAP4	FLAP3	FLAP2	FLAP1	FLAP0

Caution Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.

(6) Flash address pointer H compare register and flash address pointer L compare register (FLAPHC and FLAPLC)

These registers are used to specify the address range in which the internal sequencer operates when the flash memory is verified in the self-programming mode.

Set FLAPHC to the same value as that of FLAPH. Set the last address of the range in which verification is to be executed to FLAPLC.

These registers are set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears these registers to 00H.

Figure 14-13. Format of Flash Address Pointer H/L Compare Registers (FLAPHC/FLAPLC)

Address: FFA6H, FFA7H After reset: 00H R/W

FLAPHC (FFA6H)								FLAPLC (FFA7H)							
0	0	0	0	FLAPC11	FLAPC10	FLAPC9	FLAPC8	FLAPC7	FLAPC6	FLAPC5	FLAPC4	FLAPC3	FLAPC2	FLAPC1	FLAPC0

Cautions 1. Be sure to clear bits 4 to 7 of FLAPH and FLAPHC to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.

2. Set the number of the block subject to a block erase, verify, or blank check (same value as FLAPH) to FLAPHC.

3. Clear FLAPLC to 00H when a block erase is performed, and set this register to FFH when a blank check is performed.

(7) Flash write buffer register (FLW)

This register is used to store the data to be written to the flash memory.

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

Reset signal generation clears these registers to 00H.

Figure 14-14. Format of Flash Write Buffer Register (FLW)

Address: FFA8H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
FLW	FLW7	FLW6	FLW5	FLW4	FLW3	FLW2	FLW1	FLW0

(8) Protect byte

This protect byte is used to specify the area that is to be protected from writing or erasing. The specified area is valid only in the self-programming mode. Because self-programming of the protected area is invalid, the data written to the protected area is guaranteed.

Figure 14-15. Format of Protect Byte (1/2)

Address: 0081H

7	6	5	4	3	2	1	0
1	PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	1	1

- μPD78F9500

PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	Status
0	1	1	1	0	Blocks 3 to 0 are protected.
0	1	1	1	1	Blocks 1 and 0 are protected. Blocks 2 and 3 can be written or erased.
1	1	1	1	1	All blocks can be written or erased.
Other than above					Setting prohibited

- μPD78F9501

PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	Status
0	1	1	0	0	Blocks 7 to 0 are protected.
0	1	1	0	1	Blocks 5 to 0 are protected. Blocks 6 and 7 can be written or erased.
0	1	1	1	0	Blocks 3 to 0 are protected. Blocks 4 to 7 can be written or erased.
0	1	1	1	1	Blocks 1 and 0 are protected. Blocks 2 to 7 can be written or erased.
1	1	1	1	1	All blocks can be written or erased.
Other than above					Setting prohibited

Figure 14-19. Format of Protect Byte (2/2)

• μPD78F9502

PRSELF4	PRSELF3	PRSELF2	PRSELF1	PRSELF0	Status
0	1	0	0	0	Blocks 15 to 0 are protected.
0	1	0	0	1	Blocks 13 to 0 are protected. Blocks 14 and 15 can be written or erased.
0	1	0	1	0	Blocks 11 to 0 are protected. Blocks 12 to 15 can be written or erased.
0	1	0	1	1	Blocks 9 to 0 are protected. Blocks 10 to 15 can be written or erased.
0	1	1	0	0	Blocks 7 to 0 are protected. Blocks 8 to 15 can be written or erased.
0	1	1	0	1	Blocks 5 to 0 are protected. Blocks 6 to 15 can be written or erased.
0	1	1	1	0	Blocks 3 to 0 are protected. Blocks 4 to 15 can be written or erased.
0	1	1	1	1	Blocks 1 and 0 are protected. Blocks 2 to 15 can be written or erased.
1	1	1	1	1	All blocks can be written or erased.
Other than above					Setting prohibited

14.8.4 Example of shifting normal mode to self programming mode

The operating mode must be shifted from normal mode to self programming mode before performing self programming.

An example of shifting to self programming mode is explained below.

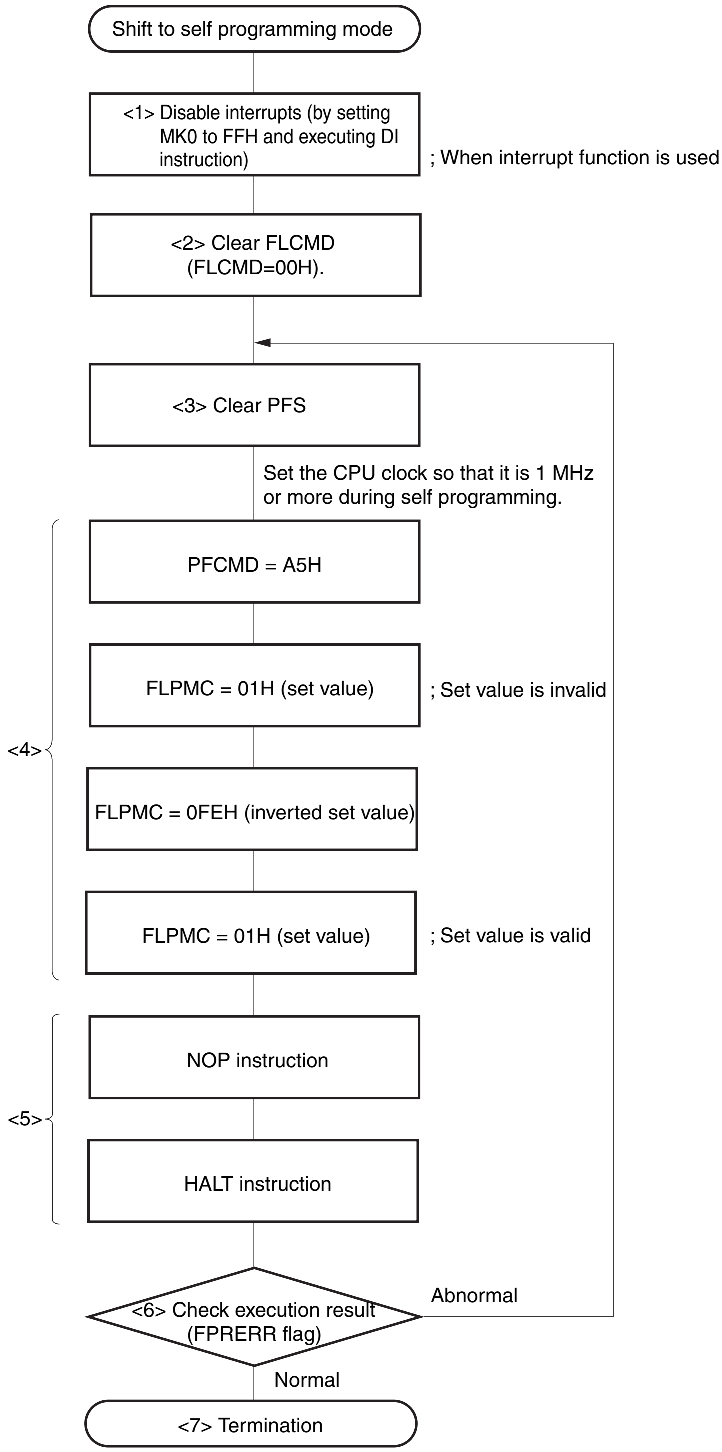
<1> Disable interrupts if the interrupt function is used (by setting the interrupt mask flag registers (MK0) to FFH and executing the DI instruction).
<2> Clear FLCMD (FLCMD=00H).
<3> Clear the flash status register (PFS).
<4> Set self programming mode using a specific sequence.[Note]
- Write a specific value (A5H) to PFCMD.
- Write 01H to FLPMC (writing in this step is invalid).
- Write 0FEH (inverted value of 01H) to FLPMC (writing in this step is invalid).
- Write 01H to FLPMC (writing in this step is valid).

<7> Execute NOP instruction and HALT instruction.
<6> Check the execution result of the specific sequence using bit 0 (FPRERR) of PFS.
Abnormal → <3>, normal → <7>
<7> Mode shift is completed.

Note Set the CPU clock so that it is 1 MHz or more during self programming.

Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.

Figure 14-16. Example of Shifting to Self Programming Mode



Caution **Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

Remark <1> to <7> in Figure 14-16 correspond to <1> to <7> in **14.8.4** (previous page).

An example of the program that shifts the mode to self programming mode is shown below.

```
;----------------------------
;START
;----------------------------
       MOV      MK0,#11111111B     ; Masks all interrupts
       MOV      FLCMD,#00H         ; Clear FLCMD register

       DI

ModeOnLoop:                        ; Configure settings so that the CPU clock ≥ 1 MHz
       MOV      PFS,#00H           ; Clears flash status register
       MOV      PFCMD,#0A5H        ; PFCMD register control
       MOV      FLPMC,#01H         ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH        ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H         ; Sets self programming mode with FLPMC register
                                   ; control (sets value)

       NOP
       HALT
       BT PFS.0,$ModeOnLoop        ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs.

;----------------------------
;END
;----------------------------
```

14.8.5 Example of shifting self programming mode to normal mode

The operating mode must be returned from self programming mode to normal mode after performing self programming.

An example of shifting to normal mode is explained below.

<1> Clear FLCMD (FLCMD=00H).

<2> Clear the flash status register (PFS).

<3> Set normal mode using a specific sequence.

- Write the specific value (A5H) to PFCMD.
- Write 00H to FLPMC (writing in this step is invalid)
- Write 0FFH (inverted value of 00H) to FLPMC (writing in this step is invalid)
- Write 00H to FLPMC (writing in this step is valid)

<4> Check the execution result of the specific sequence using bit 0 (FPRERR) of PFS.
Abnormal → <2>, normal → <5>

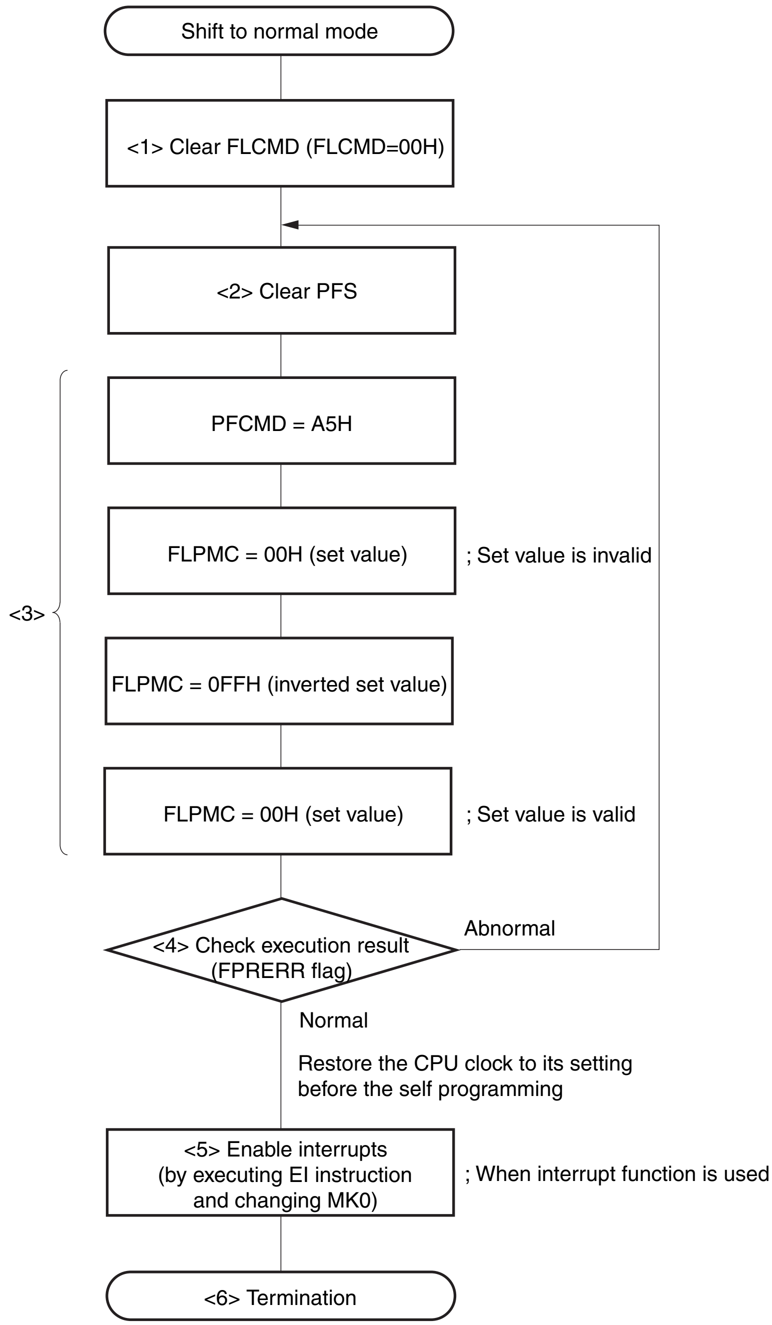
<5> Enable interrupt servicing (by executing the EI instruction and changing MK0) to restore the original state.

<6> Mode shift is completed

Note After the specific sequence is correctly executed, restore the CPU clock to its setting before the self programming.

Caution Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.

Figure 14-17. Example of Shifting to Normal Mode



Caution **Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.**

Remark <1> to <6> in Figure 14-17 correspond to <1> to <6> in **14.8.5** (previous page).

An example of a program that shifts the mode to normal mode is shown below.

```
;----------------------------
;START
;----------------------------

       MOV      FLCMD,#00H         ; Clear FLCMD register
ModeOffLoop:
       MOV      PFS,#00H           ; Clears flash status register
       MOV      PFCMD,#0A5H        ; PFCMD register control
       MOV      FLPMC,#00H         ; FLPMC register control (sets value)
       MOV      FLPMC,#0FFH        ; FLPMC register control (inverts set value)
       MOV      FLPMC,#00H         ; Sets normal mode via FLPMC register control (sets value)

       BT PFS.0,$ModeOffLoop       ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs
                                   ; Restore the CPU clock to its setting before the self
                                   ; programming

       MOV      MK0,#INT_MK0       ; Restores interrupt mask flag

       EI

;----------------------------
;END
;----------------------------
```

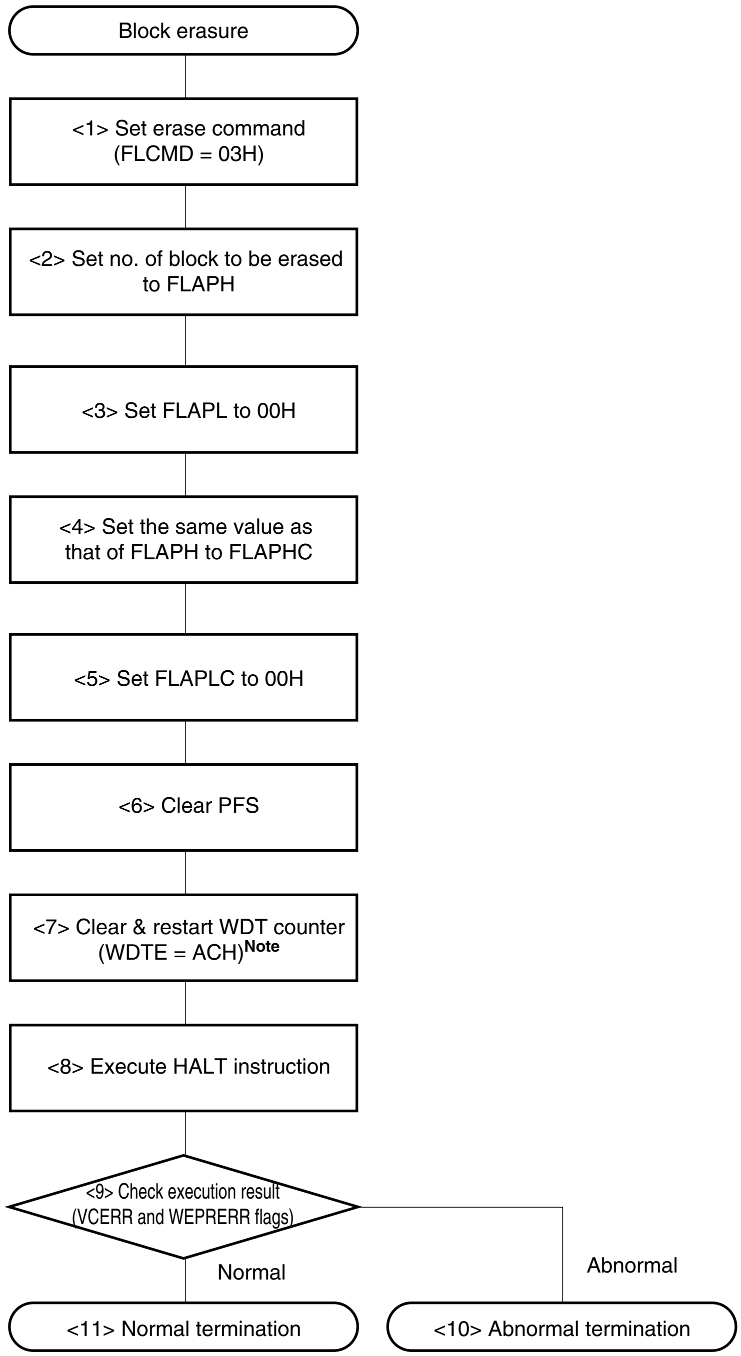
14.8.6 Example of block erase operation in self programming mode

An example of the block erase operation in self programming mode is explained below.

<1> Set 03H (block erase) to the flash program command register (FLCMD).
<2> Set the block number to be erased, to flash address pointer H (FLAPH).
<3> Set flash address pointer L (FLAPL) to 00H.
<4> Write the same value as FLAPH to the flash address pointer H compare register (FLAPHC).
<5> Set the flash address pointer L compare register (FLAPLC) to 00H.
<6> Clear the flash status register (PFS).
<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>
<10> Block erase processing is abnormally terminated.
<11> Block erase processing is normally terminated.

Note This setting is not required when the watchdog timer is not used.

Figure 14-18. Example of Block Erase Operation in Self Programming Mode



Note This setting is not required when the watchdog timer is not used.

Remark <1> to <11> in Figure 14-18 correspond to <1> to <11> in **14.8.6** (previous page).

An example of a program that performs a block erase in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------

FlashBlockErase:
       MOV      FLCMD,#03H     ; Sets flash control command (block erase)
       MOV      FLAPH,#07H     ; Sets number of block to be erased (block 7 is specified here)
       MOV      FLAPL,#00H     ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H    ; Sets erase block compare number (same value as that of FLAPH)
       MOV      FLAPLC,#00H    ; Fixes FLAPLC to "00H"

       MOV      PFS,#00H       ; Clears flash status register
       MOV      WDTE,#0ACH     ; Clears & restarts WDT
       HALT                    ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A    ; Execution result is stored in variable
                               ; (CmdStatus = 0: normal termination, other than 0: abnormal
                               ; termination)

;----------------------------
;END
;----------------------------
```

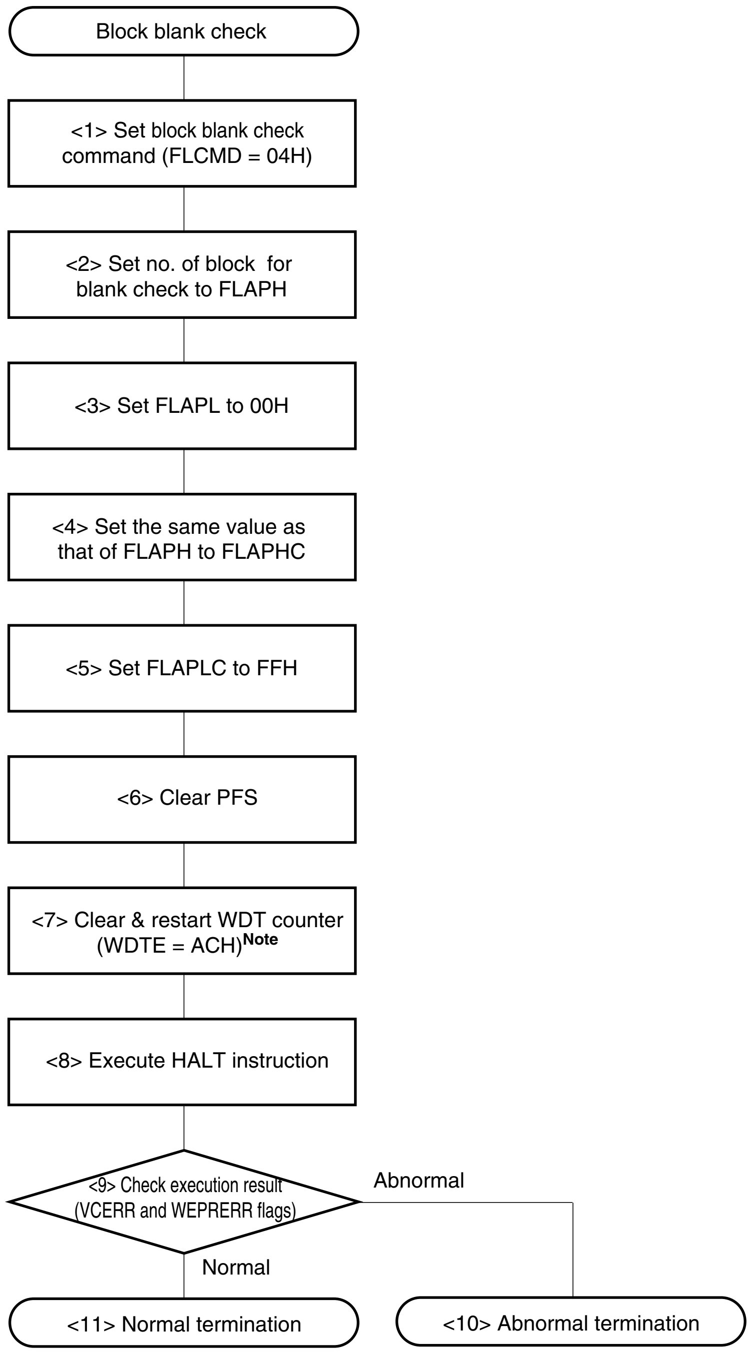
14.8.7 Example of block blank check operation in self programming mode

An example of the block blank check operation in self programming mode is explained below.

<1> Set 04H (block blank check) to the flash program command register (FLCMD).
<2> Set the number of block for which a blank check is performed, to flash address pointer H (FLAPH).
<3> Set flash address pointer L (FLAPL) to 00H.
<4> Write the same value as FLAPH to the flash address pointer H compare register (FLAPHC).
<5> Set the flash address pointer L compare register (FLAPLC) to FFH.
<6> Clear the flash status register (PFS).
<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>
<10> Block blank check is abnormally terminated.
<11> Block blank check is normally terminated.

Note This setting is not required when the watchdog timer is not used.

Figure 14-19. Example of Block Blank Check Operation in Self Programming Mode



Note This setting is not required when the watchdog timer is not used.

Remark <1> to <11>in Figure 14-19 correspond to <1> to <11> in **14.8.7** (previous page).

An example of a program that performs a block blank check in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------

FlashBlockBlankCheck:
       MOV      FLCMD,#04H      ; Sets flash control command (block blank check)
       MOV      FLAPH,#07H      ; Sets number of block for blank check (block 7 is specified
                                ; here)
       MOV      FLAPL,#00H      ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H     ; Sets blank check block compare number (same value as that of
                                ; FLAPH)
       MOV      FLAPLC,#0FFH    ; Fixes FLAPLC to "FFH"

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

14.8.8 Example of byte write operation in self programming mode

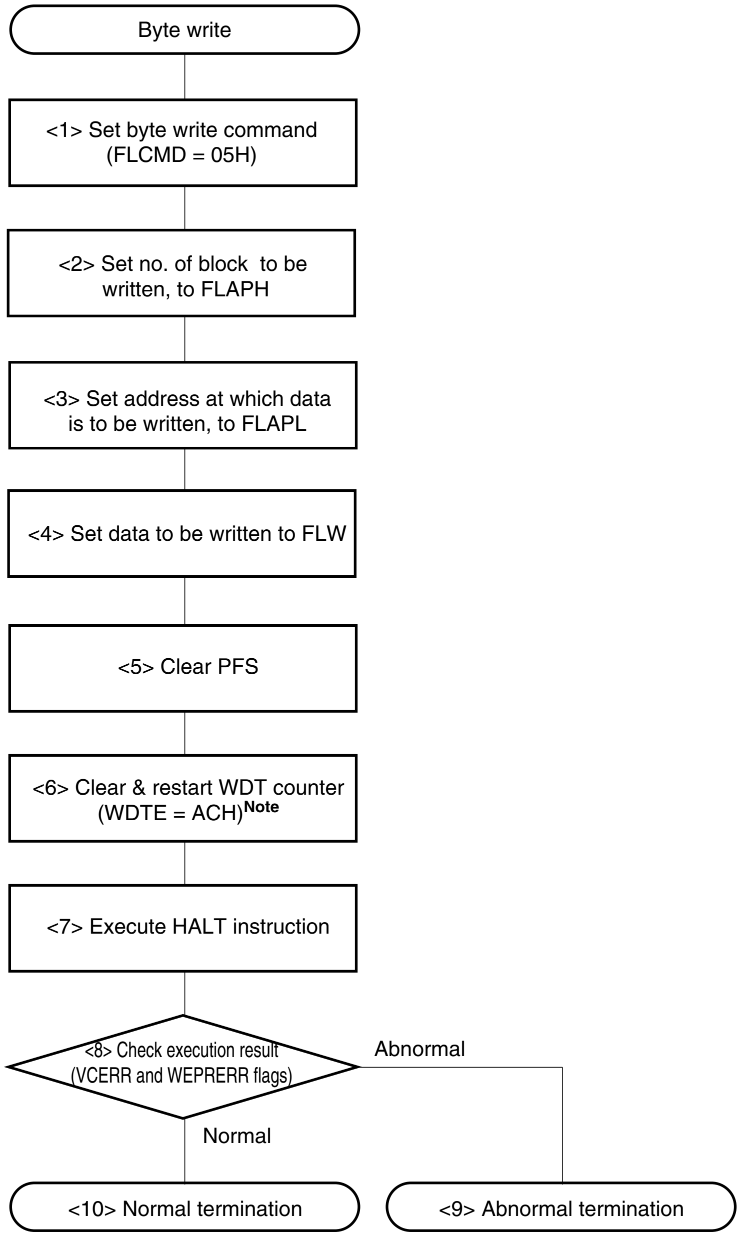
An example of the byte write operation in self programming mode is explained below.

<1> Set 05H (byte write) to the flash program command register (FLCMD).
<2> Set the number of block to which data is to be written, to flash address pointer H (FLAPH).
<3> Set the address at which data is to be written, to flash address pointer L (FLAPL).
<4> Set the data to be written, to the flash write buffer register (FLW).
<5> Clear the flash status register (PFS).
<6> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<7> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<8> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <9>
Normal → <10>
<9> Byte write processing is abnormally terminated.
<10> Byte write processing is normally terminated.

Note This setting is not required when the watchdog timer is not used.

Caution If a write results in failure, erase the block once and write to it again.

Figure 14-20. Example of Byte Write Operation in Self Programming Mode



Note This setting is not required when the watchdog timer is not used.

Remark <1> to <10> in Figure 14-20 correspond to <1> to <10> in **14.8.8** (previous page).

An example of a program that performs a byte write in self programming mode is shown below.

```
;----------------------------
;START
;----------------------------
FlashWrite:
       MOV      FLCMD,#05H      ; Sets flash control command (byte write)
       MOV      FLAPH,#07H      ; Sets address to which data is to be written, with
                                ; FLAPH (block 7 is specified here)
       MOV      FLAPL,#20H      ; Sets address to which data is to be written, with
                                ; FLAPL (address 20H is specified here)
       MOV      FLW,#10H        ; Sets data to be written (10H is specified here)

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

14.8.9 Example of internal verify operation in self programming mode

An example of the internal verify operation in self programming mode is explained below.

- **Internal verify 1**

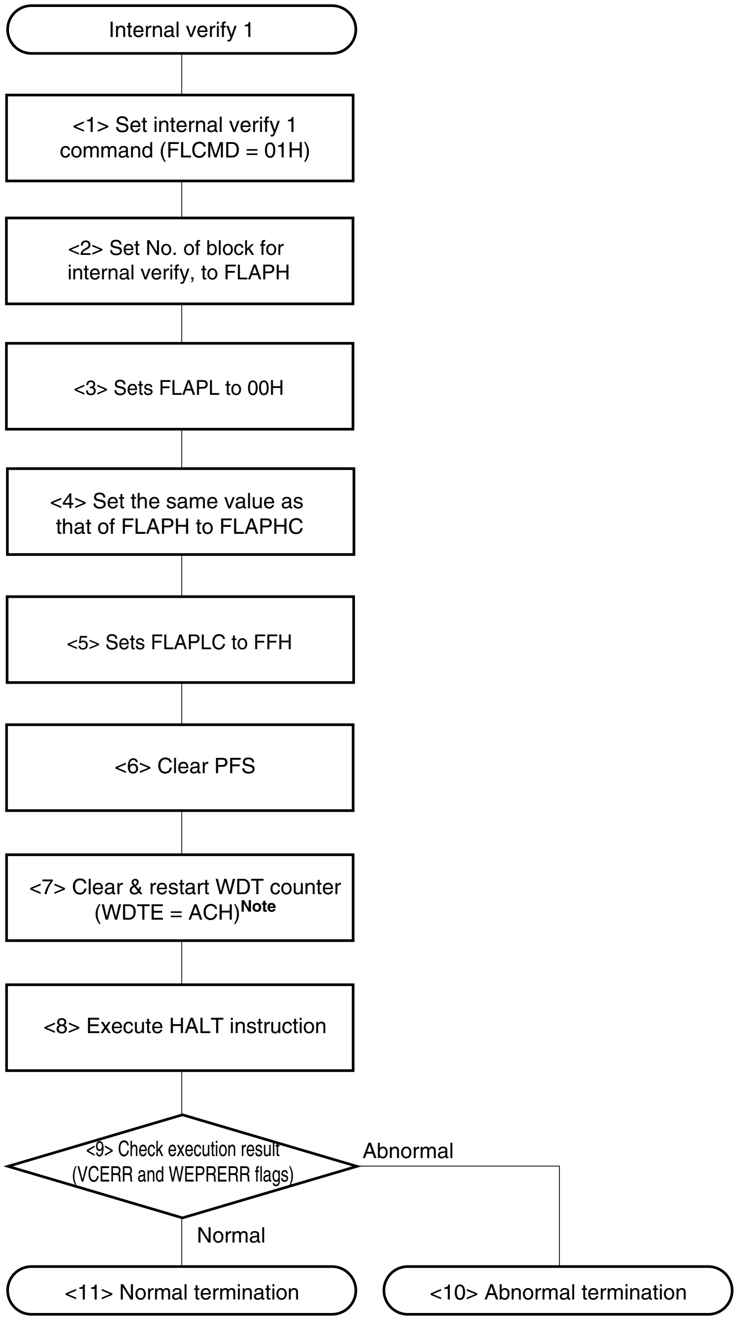
<1> Set 01H (internal verify 1) to the flash program command register (FLCMD).
<2> Set the number of block for which internal verify is performed, to flash address pointer H (FLAPH).
<3> Sets the flash address pointer L (FLAPL) to 00H.
<4> Write the same value as that of FLAPH to the flash address pointer H compare register (FLAPHC).
<5> Sets the flash address pointer L compare register (FLAPLC) to FFH.
<6> Clear the flash status register (PFS).
<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>
<10> Internal verify processing is abnormally terminated.
<11> Internal verify processing is normally terminated.

- **Internal verify 2**

<1> Set 02H (internal verify 2) to the flash program command register (FLCMD).
<2> Set the number of block for which internal verify is performed, to flash address pointer H (FLAPH).
<3> Sets flash address pointer L (FLAPL) to the start address.
<4> Write the same value as that of FLAPH to the flash address pointer H compare register (FLAPHC).
<5> Sets flash address pointer L compare register (FLAPLC) to the end address.
<6> Clear the flash status register (PFS).
<7> Write ACH to the watchdog timer enable register (WDTE) (clear and restart the watchdog timer counter)[Note].
<8> Execute the HALT instruction then start self programming. (Execute an instruction immediately after the HALT instruction if self programming has been executed.)
<9> Check if a self programming error has occurred using bit 1 (VCERR) and bit 2 (WEPRERR) of PFS.
Abnormal → <10>
Normal → <11>
<10> Internal verify processing is abnormally terminated.
<11> Internal verify processing is normally terminated.

Note This setting is not required when the watchdog timer is not used.

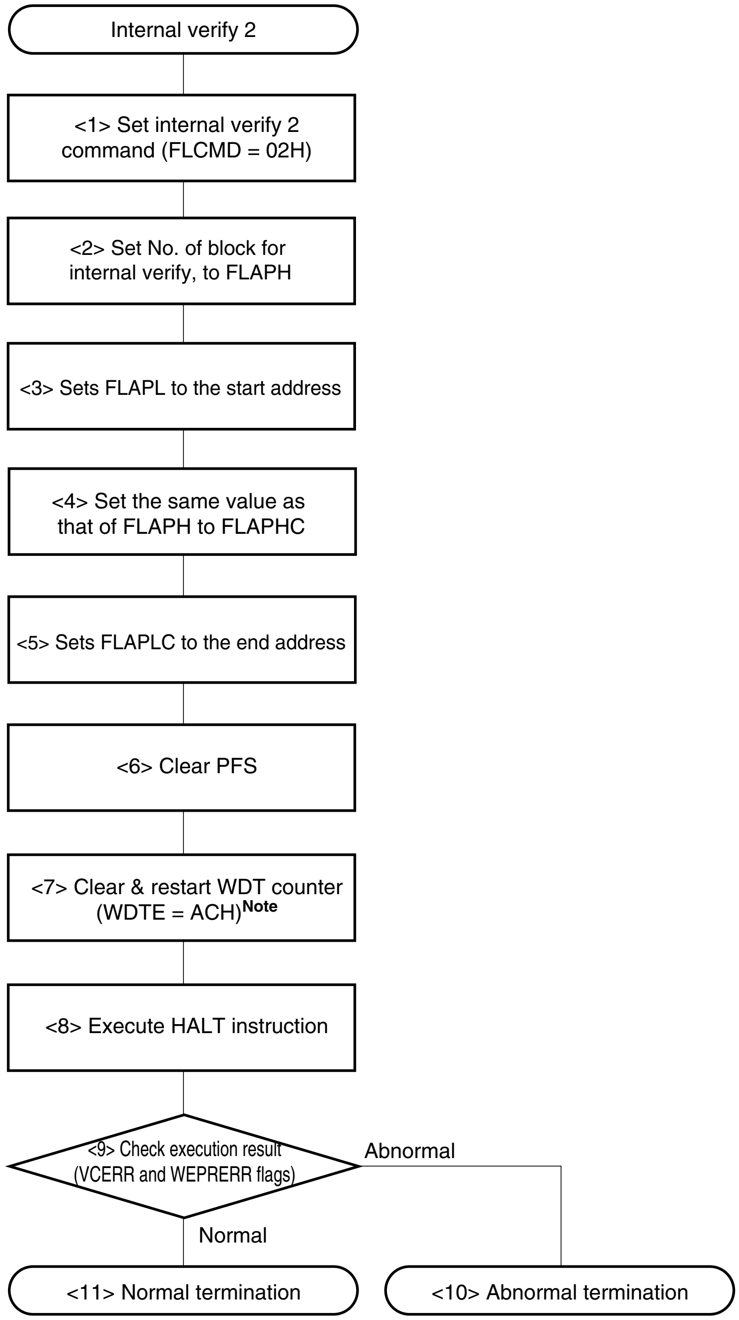
Figure 14-21. Example of Internal Verify Operation in Self Programming Mode



Note This setting is not required when the watchdog timer is not used.

Remark <1> to <11> in Figure 14-21 correspond to Internal verify 1 <1> to <11> in **14.8.9** (previous page).

Figure 14-22. Example of Internal Verify Operation in Self Programming Mode



Note This setting is not required when the watchdog timer is not used.

Remark <1> to <11> in Figure 14-22 correspond to Internal verify 2 <1> to <11> in **14.8.9** (the page before last).

An example of a program that performs an internal verify in self programming mode is shown below.

- **Internal verify 1**

```
;----------------------------
;START
;----------------------------
FlashVerify:
       MOV      FLCMD,#01H      ; Sets flash control command (internal verify 1)
       MOV      FLAPH,#07H      ; Set the number of block for which internal verify is
                                ; performed, to FLAPH (Example: Block 7 is specified here)
       MOV      FLAPL,#00H      ; Sets FLAPL to 00H
       MOV      FLAPHC,#07H
       MOV      FLAPLC,#FFH     ; Sets FLAPLC to FFH

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

- **Internal verify 2**

```
;----------------------------
;START
;----------------------------
FlashVerify:
       MOV      FLCMD,#02H      ; Sets flash control command (internal verify 2)
       MOV      FLAPH,#07H      ; Set the number of block for which internal verify is
                                ; performed, to FLAPH (Example: Block 7 is specified here)
       MOV      FLAPL,#00H      ; Sets FLAPL to the start address for verify (Example: Address
                                ; 00H is specified here)
       MOV      FLAPHC,#07H
       MOV      FLAPLC,#20H     ; Sets FLAPLC to the end address for verify (Example: Address
                                ; 20H is specified here)

       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       MOV      CmdStatus,A     ; Execution result is stored in variable
                                ; (CmdStatus = 0: normal termination, other than 0: abnormal
                                ; termination)

;----------------------------
;END
;----------------------------
```

14.8.10 Examples of operation when command execution time should be minimized in self programming mode

Examples of operation when the command execution time should be minimized in self programming mode are explained below.

(1) Erasure to blank check

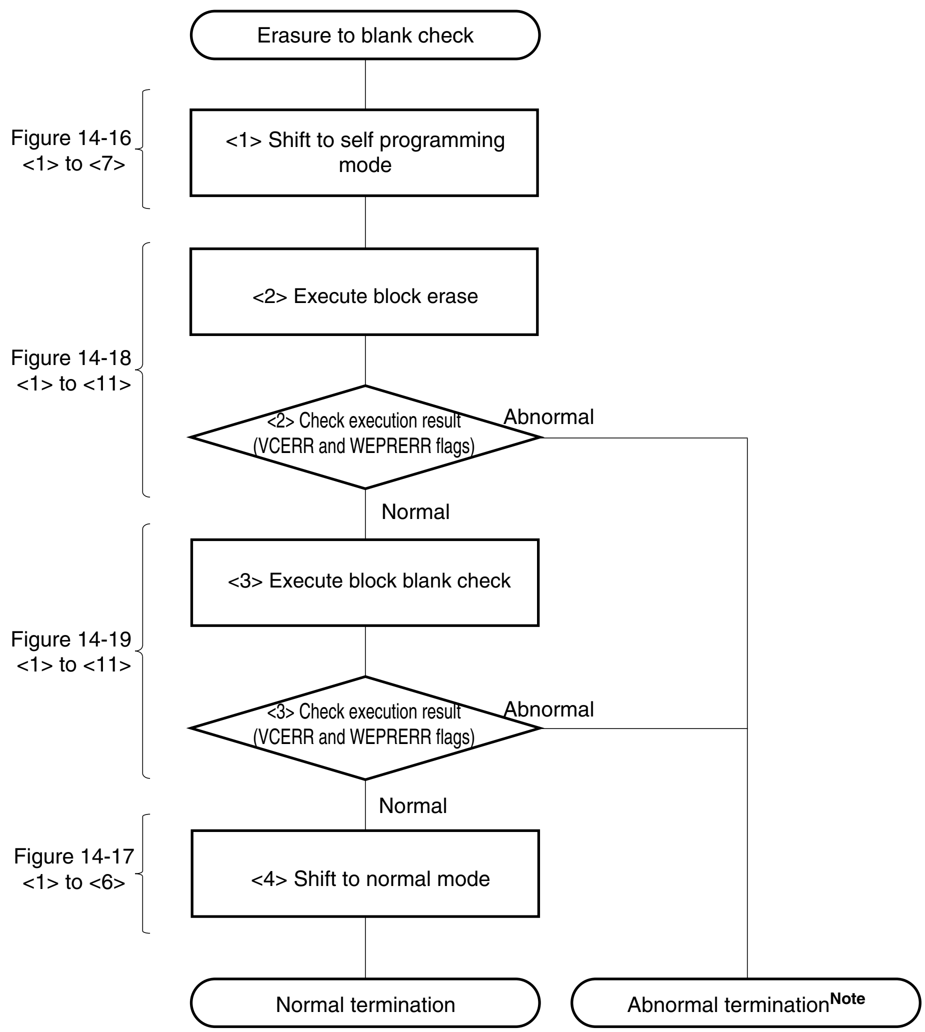
<1> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)

<2> Execution of block erase → Error check (<1> to <11> in **14.8.6**)

<3> Execution of block blank check → Error check (<1> to <11> in **14.8.7**)

<4> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)

Figure 14-23. Example of Operation When Command Execution Time Should Be Minimized (from Erasure to Blank Check)



Note Perform processing to shift to normal mode in order to return to normal processing.

Remark <1> to <4> in Figure 14-23 correspond to <1> to <4> in **14.8.10 (1)** above.

An example of a program when the command execution time (from erasure to black check) should be minimized in self programming mode is shown below.

```
;---------------------------------------------------------------------
;START
;---------------------------------------------------------------------
       MOV      MK0,#11111111B   ; Masks all interrupts
       MOV      FLCMD,#00H       ; Clears FLCMD register

       DI

ModeOnLoop:                      ; Configure settings so that the CPU clock · 1 MHz
       MOV      PFS,#00H         ; Clears flash status register
       MOV      PFCMD,#0A5H      ; PFCMD register control
       MOV      FLPMC,#01H       ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH      ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H       ; Sets self programming mode with FLPMC register control (sets
                                 ; value)

       NOP
       HALT
       BT PFS.0,$ModeOnLoop      ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs.
FlashBlockErase:
       MOV      FLCMD,#03H       ; Sets flash control command (block erase)
       MOV      FLAPH,#07H       ; Sets number of block to be erased (block 7 is specified
                                 ; here)
       MOV      FLAPL,#00H       ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H      ; Sets erase block compare number (same value as that of
                                 ; FLAPH)
       MOV      FLAPLC,#00H      ; Fixes FLAPLC to "00H"

       MOV      WDTE,#0ACH       ; Clears & restarts WDT
       HALT                      ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError     ; Checks erase error
                                 ; Performs abnormal termination processing when an error
                                 ; occurs.

FlashBlockBlankCheck:
       MOV      FLCMD,#04H       ; Sets flash control command (block blank check)
       MOV      FLAPH,#07H       ; Sets number of block for blank check (block 7 is specified
                                 ; here)
       MOV      FLAPL,#00H       ; Fixes FLAPL to "00H"

       MOV      FLAPHC,#07H      ; Sets blank check block compare number (same value as of
                                 ; FLAPH)
```

```
        MOV      FLAPLC,#0FFH      ; Fixes FLAPLC to "FFH"
        MOV      WDTE,#0ACH        ; Clears & restarts WDT
        HALT                       ; Self programming is started
        MOV      A,PFS
        CMP      A,#00H
        BNZ      $StatusError      ; Checks blank check error
                                   ; Performs abnormal termination processing when an error
                                   ; occurs.

        MOV      FLCMD,#00H        ; Clears FLCMD register
ModeOffLoop:
        MOV      PFS,#00H          ; Clears flash status register
        MOV      PFCMD,#0A5H       ; PFCMD register control
        MOV      FLPMC,#00H        ; FLPMC register control (sets value)
        MOV      FLPMC,#0FFH       ; FLPMC register control (inverts set value)
        MOV      FLPMC,#00H        ; Sets normal mode via FLPMC register control (sets value)

        BT PFS.0,$ModeOffLoop      ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs.
                                   ; After the specific sequence is correctly executed, restore
                                   ; the CPU clock to its setting before the self programming

        MOV      MK0,#INT_MK0      ; Restores interrupt mask flag

        EI

        BR       StatusNormal

;----------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;----------------------------------------------------------------------
StatusError:

;----------------------------------------------------------------------
;END (normal termination processing)
;----------------------------------------------------------------------
StatusNormal:
```

(2) Write to internal verify

<1> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)

<2> Specification of source data for write

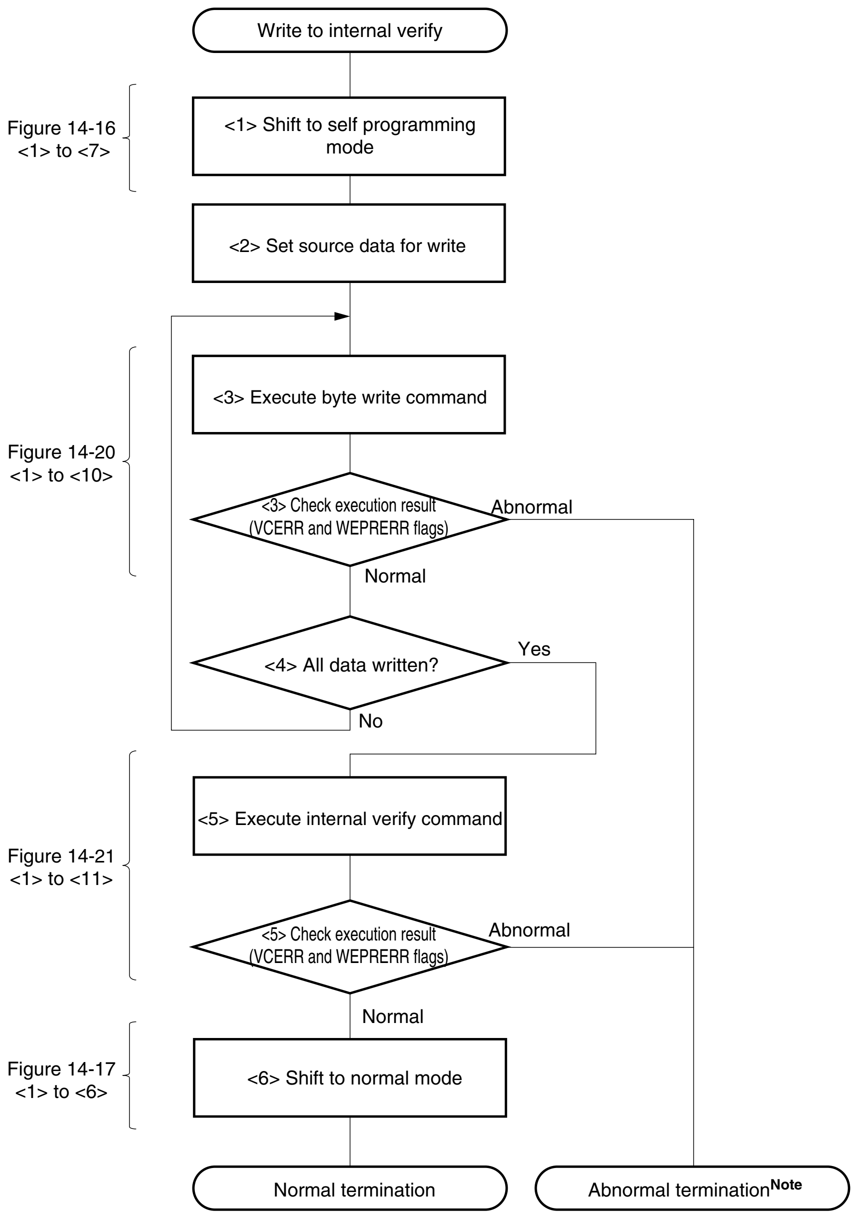
<3> Execution of byte write → Error check (<1> to <10> in **14.8.8**)

<4> <3> is repeated until all data are written.

<5> Execution of internal verify → Error check (<1> to <11> in **14.8.9**)

<6> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)

Figure 14-24. Example of Operation When Command Execution Time Should Be Minimized (from Write to Internal Verify)



Note Perform processing to shift to normal mode in order to return to normal processing.

Remark <1> to <6> in Figure 14-24 correspond to <1> to <6> in **14.8.10 (2)** above.

An example of a program when the command execution time (from write to internal verify) should be minimized in self programming mode is shown below.

```
;----------------------------------------------------------------------
;START
;----------------------------------------------------------------------
       MOV      MK0,#11111111B   ; Masks all interrupts
       MOV      FLCMD,#00H       ; Clears FLCMD register
       DI

ModeOnLoop:                      ; Configure settings so that the CPU clock ≥ 1 MHz
       MOV      PFS,#00H         ; Clears flash status register
       MOV      PFCMD,#0A5H      ; PFCMD register control
       MOV      FLPMC,#01H       ; FLPMC register control (sets value)
       MOV      FLPMC,#0FEH      ; FLPMC register control (inverts set value)
       MOV      FLPMC,#01H       ; Sets self programming mode with FLPMC register control
                                 ; (sets value)

       NOP
       HALT
       BT PFS.0,$ModeOnLoop      ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs.

FlashWrite:
       MOVW     HL,#DataAdrTop   ; Sets address at which data to be written is located
       MOVW     DE,#WriteAdr     ; Sets address at which data is to be written
FlashWriteLoop:
       MOV      FLCMD,#05H       ; Sets flash control command (byte write)
       MOV      A,D
       MOV      FLAPH,A          ; Sets address at which data is to be written
       MOV      A,E
       MOV      FLAPL,A          ; Sets address at which data is to be written
       MOV      A,[HL]
       MOV      FLW,A            ; Sets data to be written

       MOV      WDTE,#0ACH       ; Clears & restarts WDT
       HALT                      ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError     ; Checks write error
                                 ; Performs abnormal termination processing when an error
                                 ; occurs.

       INCW     HL               ; address at which data to be written is located + 1
       MOVW     AX,HL
       CMPW     AX,#DataAdrBtm   ; Performs internal verify processing
       BNC      $FlashVerify     ; if write of all data is completed
```

```
        INCW     DE                ; Address at which data is to be written + 1
        BR       FlashWriteLoop

FlashVerify:
        MOVW     HL,#WriteAdr      ; Sets verify address

        MOV      FLCMD,#02H        ; Sets flash control command (internal verify 2)
        MOV      A,H
        MOV      FLAPH,A           ; Sets verify start address
        MOV      A,L
        MOV      FLAPL,A           ; Sets verify start address
        MOV      A,D
        MOV      FLAPHC,A          ; Sets verify end address
        MOV      A,E
        MOV      FLAPLC,A          ; Sets verify end address

        MOV      WDTE,#0ACH        ; Clears & restarts WDT
        HALT                       ; Self programming is started
        MOV      A,PFS
        CMP      A,#00H
        BNZ      $StatusError      ; Checks internal verify error
                                   ; Performs abnormal termination processing when an error
                                   ; occurs.

        MOV      FLCMD,#00H        ; Clears FLCMD register
ModeOffLoop:
        MOV      PFS,#00H          ; Clears flash status register
        MOV      PFCMD,#0A5H       ; PFCMD register control
        MOV      FLPMC,#00H        ; FLPMC register control (sets value)
        MOV      FLPMC,#0FFH       ; FLPMC register control (inverts set value)
        MOV      FLPMC,#00H        ; Sets normal mode via FLPMC register control (sets value)

        BT PFS.0,$ModeOffLoop      ; Checks completion of write to specific registers
                                   ; Repeats the same processing when an error occurs.
                                   ; After the specific sequence is correctly executed, restore
                                   ; the CPU clock to its setting before the self programming
        MOV      MK0,#INT_MK0      ; Restores interrupt mask flag

        EI

        BR       StatusNormal

;---------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;---------------------------------------------------------------------
StatusError:
```

```
;--------------------------------------------------------------------
;END (normal termination processing)
;--------------------------------------------------------------------
StatusNormal:



;--------------------------------------------------------------------
; Data to be written
;--------------------------------------------------------------------
DataAdrTop:
        DB      XXH
        DB      XXH
        DB      XXH
        DB      XXH

         :
         :

        DB      XXH
DataAdrBtm:
;--------------------------------------------------------------------
```

Remark Internal verify 2 is used in the above program example. Use internal verify 1 to verify s whole block.

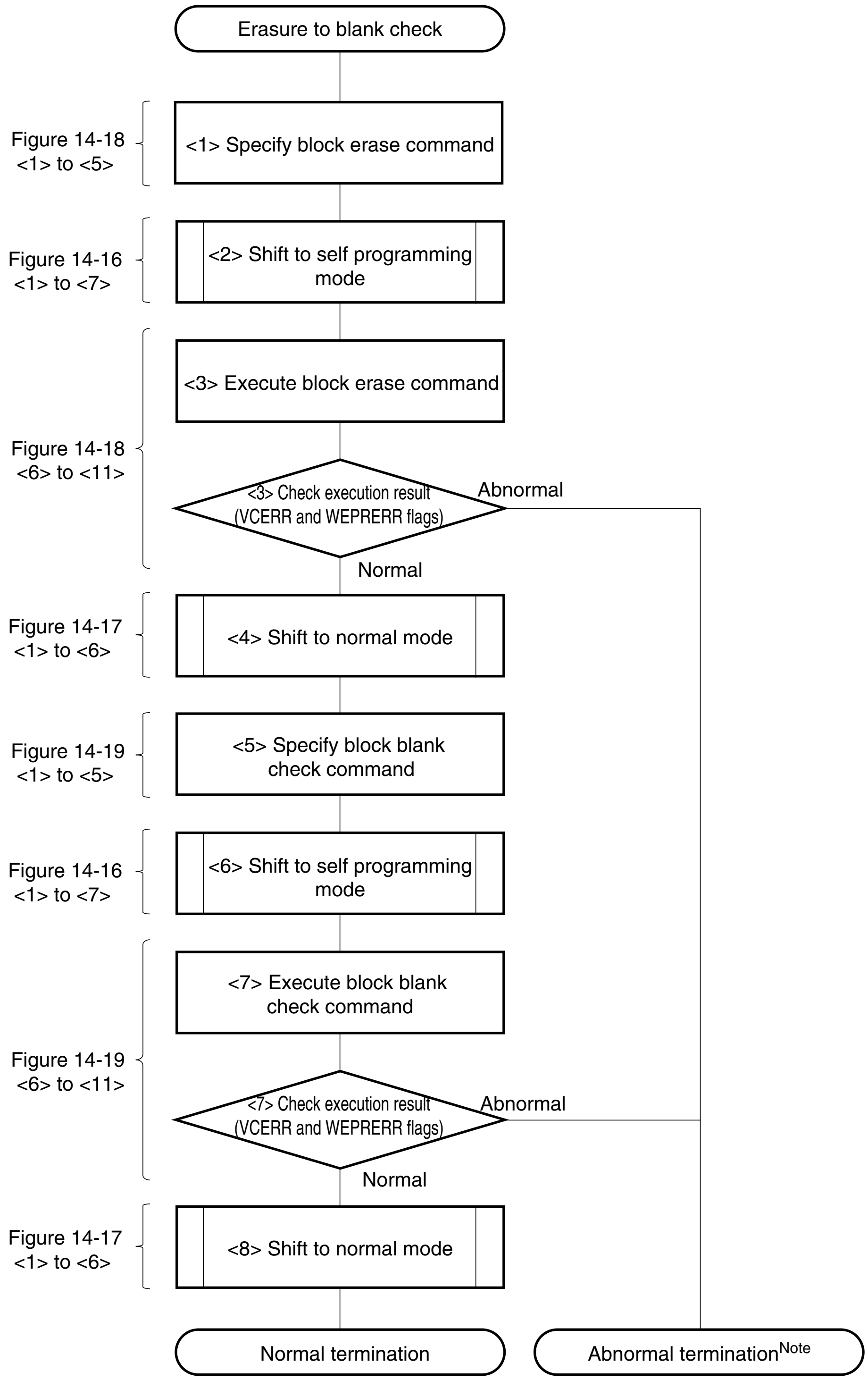
14.8.11 Examples of operation when interrupt-disabled time should be minimized in self programming mode

Examples of operation when the interrupt-disabled time should be minimized in self programming mode are explained below.

(1) Erasure to blank check

<1> Specification of block erase command (<1> to <5> in **14.8.6**)
<2> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)
<3> Execution of block erase command → Error check (<6> to <11> in **14.8.6**)
<4> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)
<5> Specification of block blank check command (<1> to <5> in **14.8.7**)
<6> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)
<7> Execution of block blank check command → Error check (<6> to <11> in **14.8.7**)
<8> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)

Figure 14-25. Example of Operation When Interrupt-Disabled Time Should Be Minimized (from Erasure to Blank Check)



Note Perform processing to shift to normal mode in order to return to normal processing.

Remark <1> to <8> in Figure 14-25 correspond to <1> to <8> in **14.8.11 (1)** (previous page).

An example of a program when the interrupt-disabled time (from erasure to blank check) should be minimized in self programming mode is shown below.

```
;---------------------------------------------------------------------
;START
;---------------------------------------------------------------------
FlashBlockErase:
       ; Sets erase command
       MOV      FLCMD,#03H      ; Sets flash control command (block erase)
       MOV      FLAPH,#07H      ; Sets number of block to be erased (block 7 is specified here)
       MOV      FLAPL,#00H      ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H     ; Sets erase block compare number (same value as that of FLAPH)
       MOV      FLAPLC,#00H     ; Fixes FLAPLC to "00H"

       CALL     !ModeOn         ; Shift to self programming mode

       ; Execution of erase command
       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError    ; Checks erase error
                                ; Performs abnormal termination processing when an error
                                ; occurs.

       CALL     !ModeOff        ; Shift to normal mode


       ; Sets blank check command
       MOV      FLCMD,#04H      ; Sets flash control command (block blank check)
       MOV      FLAPH,#07H      ; Sets block number for blank check (block 7 is specified here)
       MOV      FLAPL,#00H      ; Fixes FLAPL to "00H"
       MOV      FLAPHC,#07H     ; Sets blank check block compare number (same value as that of
                                ; FLAPH)
       MOV      FLAPLC,#0FFH    ; Fixes FLAPLC to "FFH"

       CALL     !ModeOn         ; Shift to self programming mode

       ; Execution of blank check command
       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError    ; Checks blank check error
                                ; Performs abnormal termination processing when an error occurs
```

```
        CALL     !ModeOff        ; Shift to normal mode

        BR       StatusNormal

;-------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;-------------------------------------------------------------------
StatusError:


;-------------------------------------------------------------------
;END (normal termination processing)
;-------------------------------------------------------------------
StatusNormal:


;-------------------------------------------------------------------
;Processing to shift to self programming mode
;-------------------------------------------------------------------
ModeOn:
        MOV      MK0,#11111111B   ; Masks all interrupts
        MOV      FLCMD,#00H       ; Clears FLCMD register

        DI

ModeOnLoop:                      ; Configure settings so that the CPU clock ≥ 1 MHz
        MOV      PFS,#00H        ; Clears flash status register
        MOV      PFCMD,#0A5H     ; PFCMD register control
        MOV      FLPMC,#01H      ; FLPMC register control (sets value)
        MOV      FLPMC,#0FEH     ; FLPMC register control (inverts set value)
        MOV      FLPMC,#01H      ; Sets self programming mode via FLPMC register control (sets
                                 ; value)

        NOP
        HALT
        BT PFS.0,$ModeOnLoop     ; Checks completion of write to specific registers
                                 ; Repeats the same processing when an error occurs.

        RET

;-------------------------------------------------------------------
; Processing to shift to normal mode
;-------------------------------------------------------------------
ModeOff:
        MOV      FLCMD,#00H      ; Clears FLCMD register
        MOV      PFS,#00H        ; Clears flash status register
        MOV      PFCMD,#0A5H     ; PFCMD register control
```

```
MOV      FLPMC,#00H       ; FLPMC register control (sets value)
MOV      FLPMC,#0FFH      ; FLPMC register control (inverts set value)
MOV      FLPMC,#00H       ; Sets normal mode via FLPMC register control (sets value)

BT PFS.0,$ModeOffLoop     ; Checks completion of write to specific registers
                          ; Repeats the same processing when an error occurs.
                          ; After the specific sequence is correctly executed, restore
                          ; the CPU clock to its setting before the self programming
MOV      MK0,#INT_MK0     ; Restores interrupt mask flag

EI

RET
```

(2) Write to internal verify

<1> Specification of source data for write

<2> Specification of byte write command (<1> to <4> in **14.8.8**)

<3> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)

<4> Execution of byte write command → Error check (<5> to <10> in **14.8.8**)

<5> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)

<6> <2> to <5> is repeated until all data are written.

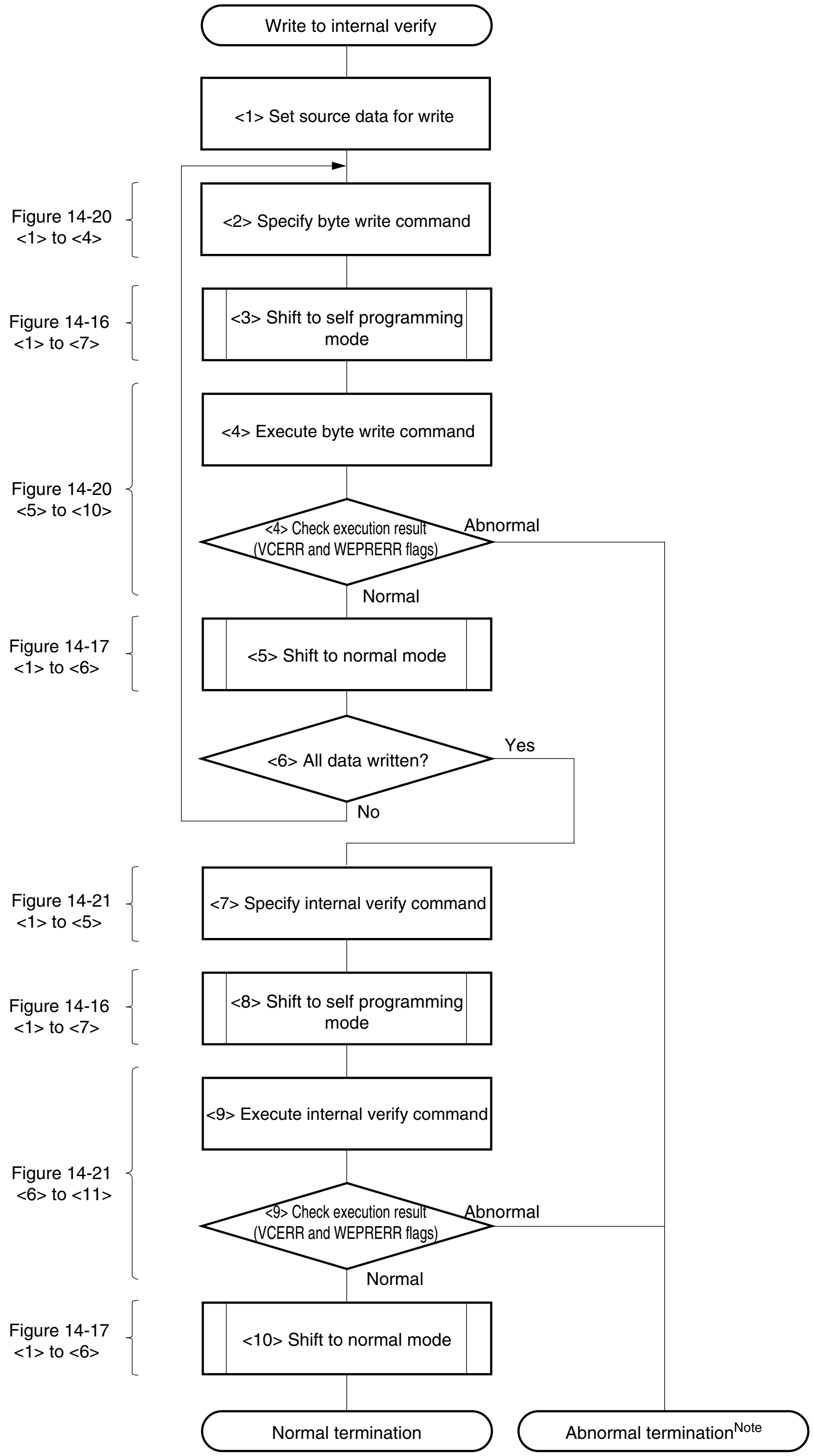
<7> The internal verify command is specified (<1> to <5> in **14.8.9**)

<8> Mode is shifted from normal mode to self programming mode (<1> to <7> in **14.8.4**)

<9> Execution of internal verify command → Error check (<6> to <11> in **14.8.9**)

<10> Mode is shifted from self programming mode to normal mode (<1> to <6> in **14.8.5**)

Figure 14-26. Example of Operation When Interrupt-Disabled Time Should Be Minimized (from Write to Internal Verify)



Note Perform processing to shift to normal mode in order to return to normal processing.

Remark <1> to <10> in Figure 14-28 correspond to <1> to <10> in **14.8.11 (2)** (previous page).

An example of a program when the interrupt-disabled time (from write to internal verify) should be minimized in self programming mode is shown below.

```
;----------------------------------------------------------------------
;START
;----------------------------------------------------------------------
       ; Sets write command
FlashWrite:
       MOVW     HL,#DataAdrTop ; Sets address at which data to be written is located
       MOVW     DE,#WriteAdr   ; Sets address at which data is to be written

FlashWriteLoop:
       MOV      FLCMD,#05H     ; Sets flash control command (byte write)
       MOV      A,D
       MOV      FLAPH,A        ; Sets address at which data is to be written
       MOV      A,E
       MOV      FLAPL,A        ; Sets address at which data is to be written
       MOV      A,[HL]
       MOV      FLW,A          ; Sets data to be written

       CALL     !ModeOn        ; Shift to self programming mode

       ; Execution of write command
       MOV      PFS,#00H       ; Clears flash status register
       MOV      WDTE,#0ACH     ; Clears & restarts WDT
       HALT                    ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError   ; Checks write error
                               ; Performs abnormal termination processing when an error
                               ; occurs.

       CALL     !ModeOff       ; Shift to normal mode

       MOV      MK0,#INT_MK0   ; Restores interrupt mask flag

       EI

       ; Judgment of writing all data
       INCW     HL             ; Address at which data to be written is located + 1
       MOVW     AX,HL
       CMPW     AX,#DataAdrBtm ; Performs internal verify processing
       BNC      $FlashVerify   ; if write of all data is completed

       INCW     DE             ; Address at which data is to be written + 1
       BR       FlashWriteLoop

       ; Setting internal verify command
```

```
FlashVerify:
       MOVW     HL,#WriteAdr    ; Sets verify address

       MOV      FLCMD,#02H      ; Sets flash control command (internal verify 2)
       MOV      A,H
       MOV      FLAPH,A         ; Sets verify start address
       MOV      A,L
       MOV      FLAPL,A         ; Sets verify start address
       MOV      A,D
       MOV      FLAPHC,A        ; Sets verify end address
       MOV      A,E
       MOV      FLAPLC,A        ; Sets verify end address

       CALL     !ModeOn         ; Shift to self programming mode

       ; Execution of internal verify command
       MOV      PFS,#00H        ; Clears flash status register
       MOV      WDTE,#0ACH      ; Clears & restarts WDT
       HALT                     ; Self programming is started
       MOV      A,PFS
       CMP      A,#00H
       BNZ      $StatusError    ; Checks internal verify error
                                ; Performs abnormal termination processing when an error occurs

       CALL     !ModeOff        ; Shift to normal mode

       BR       StatusNormal

;----------------------------------------------------------------------
;END (abnormal termination processing); Perform processing to shift to
      normal mode in order to return to normal processing
;----------------------------------------------------------------------
StatusError:



;----------------------------------------------------------------------
;END (normal termination processing)
;----------------------------------------------------------------------
StatusNormal:



;----------------------------------------------------------------------
;Processing to shift to self programming mode
;----------------------------------------------------------------------
ModeOn:
       MOV      MK0,#11111111B  ; Masks all interrupts
       MOV      FLCMD,#00H      ; Clears FLCMD register
```

```
        DI

ModeOnLoop:                     ; Configure settings so that the CPU clock ≥ 1 MHz
        MOV     PFS,#00H        ; Clears flash status register
        MOV     PFCMD,#0A5H     ; PFCMD register control
        MOV     FLPMC,#01H      ; FLPMC register control (sets value)
        MOV     FLPMC,#0FEH     ; FLPMC register control (inverts set value)
        MOV     FLPMC,#01H      ; Sets self programming mode via FLPMC register control (sets
                                ; value)

        NOP
        HALT
        BT PFS.0,$ModeOnLoop    ; Checks completion of write to specific registers
                                ; Repeats the same processing when an error occurs.

        RET


;-------------------------------------------------------------------
; Processing to shift to normal mode
;-------------------------------------------------------------------
ModeOff:
        MOV     FLCMD,#00H      ; Clears FLCMD register
        MOV     PFS,#00H        ; Clears flash status register
        MOV     PFCMD,#0A5H     ; PFCMD register control
        MOV     FLPMC,#00H      ; FLPMC register control (sets value)
        MOV     FLPMC,#0FFH     ; FLPMC register control (inverts set value)
        MOV     FLPMC,#00H      ; Sets normal mode via FLPMC register control (sets value)

        BT PFS.0,$ModeOffLoop   ; Checks completion of write to specific registers
                                ; Repeats the same processing when an error occurs.
                                ; After the specific sequence is correctly executed, restore
                                ; the CPU clock to its setting before the self programming

        MOV     MK0,#INT_MK0    ; Restores interrupt mask flag

        EI

        RET


;-------------------------------------------------------------------
;Data to be written
;-------------------------------------------------------------------
DataAdrTop:
        DB      XXH
        DB      XXH
        DB      XXH
```

```
        DB        XXH

        :
        :

        DB        XXH
DataAdrBtm:
;-----------------------------------------------------------------------
```

Remark Internal verify 2 is used in the above program example. Use internal verify 1 to verify s whole block.

CHAPTER 15 INSTRUCTION SET OVERVIEW

This chapter lists the instruction set of the μPD78F9500, 78F9501, 78F9502. For details of the operation and machine language (instruction code) of each instruction, refer to **78K/0S Series Instructions User's Manual (U11047E)**.

15.1 Operation

15.1.1 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 15-1. Operand Identifiers and Description Methods

Identifier	Description Method
r	X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7)
rp	AX (RP0), BC (RP1), DE (RP2), HL (RP3)
sfr	Special function register symbol
saddr	FE20H to FF1FH Immediate data or labels
saddrp	FE20H to FF1FH Immediate data or labels (even addresses only)
addr16	0000H to FFFFH Immediate data or labels (only even addresses for 16-bit data transfer instructions)
addr5	0040H to 007FH Immediate data or labels (even addresses only)
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label

Remark For symbols of special function registers, see **Table 3-3 Special Function Registers**.

15.1.2 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
( ):	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)
$\overline{}$:	Inverted data
addr16:	16-bit immediate data or label
jdisp8:	Signed 8-bit data (displacement value)

15.1.3 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

15.2 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	Flag AC	Flag 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 ← A_0, A_{m-1} ← A_m) × 1			×
ROL	A, 1	1	2	(CY, A_0 ← A_7, A_{m+1} ← A_m) × 1			×
RORC	A, 1	1	2	(CY ← A_0, A_7 ← CY, A_{m-1} ← A_m) × 1			×
ROLC	A, 1	1	2	(CY ← A_7, A_0 ← CY, A_{m+1} ← 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) ← $(PC + 3)_H$, (SP – 2) ← $(PC + 3)_L$, PC ← addr16, SP ← SP – 2			
CALLT	[addr5]	1	8	(SP – 1) ← $(PC + 1)_H$, (SP – 2) ← $(PC + 1)_L$, PC_H ← (00000000, addr5 + 1), PC_L ← (00000000, addr5), SP ← SP – 2			
RET		1	6	PC_H ← (SP + 1), PC_L ← (SP), SP ← SP + 2			
RETI		1	8	PC_H ← (SP + 1), PC_L ← (SP), PSW ← (SP + 2), SP ← SP + 3, NMIS ← 0	R	R	R
PUSH	PSW	1	2	(SP – 1) ← PSW, SP ← SP – 1			
	rp	1	4	(SP – 1) ← rp_H, (SP – 2) ← rp_L, SP ← SP – 2			
POP	PSW	1	4	PSW ← (SP), SP ← SP + 1	R	R	R
	rp	1	6	rp_H ← (SP + 1), rp_L ← (SP), SP ← SP + 2			
MOVW	SP, AX	2	8	SP ← AX			
	AX, SP	2	6	AX ← SP			
BR	!addr16	3	6	PC ← addr16			
	$addr16	2	6	PC ← PC + 2 + jdisp8			
	AX	1	6	PC_H ← A, PC_L ← X			
BC	$saddr16	2	6	PC ← PC + 2 + jdisp8 if CY = 1			
BNC	$saddr16	2	6	PC ← PC + 2 + jdisp8 if CY = 0			
BZ	$saddr16	2	6	PC ← PC + 2 + jdisp8 if Z = 1			
BNZ	$saddr16	2	6	PC ← PC + 2 + jdisp8 if Z = 0			
BT	saddr.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if (saddr.bit) = 1			
	sfr.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if sfr.bit = 1			
	A.bit, $addr16	3	8	PC ← PC + 3 + jdisp8 if A.bit = 1			
	PSW.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if PSW.bit = 1			
BF	saddr.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if (saddr.bit) = 0			
	sfr.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if sfr.bit = 0			
	A.bit, $addr16	3	8	PC ← PC + 3 + jdisp8 if A.bit = 0			
	PSW.bit, $addr16	4	10	PC ← PC + 4 + jdisp8 if PSW.bit = 0			
DBNZ	B, $addr16	2	6	B ← B – 1, then PC ← PC + 2 + jdisp8 if B ≠ 0			
	C, $addr16	2	6	C ← C – 1, then PC ← PC + 2 + jdisp8 if C ≠ 0			
	saddr, $addr16	3	8	(saddr) ← (saddr) – 1, then PC ← PC + 3 + jdisp8 if (saddr) ≠ 0			
NOP		1	2	No Operation			
EI		3	6	IE ← 1 (Enable Interrupt)			
DI		3	6	IE ← 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).

15.3 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

2nd Operand / 1st 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

2nd Operand / 1st 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

2nd Operand / 1st 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

2nd Operand / 1st 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

CHAPTER 16 ELECTRICAL SPECIFICATIONS (TARGET)

Absolute Maximum Ratings (T_A = 25°C)

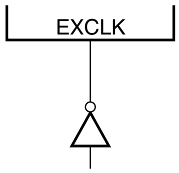
Parameter	Symbol	Conditions	Ratings	Unit
Supply voltage	V_{DD}		–0.3 to +6.5	V
	V_{SS}		–0.3 to +0.3	V
Input voltage	V_I	P20 to P23, P32, P34, P40, P43	–0.3 to V_{DD} + 0.3 Note	V
Output voltage	V_O		–0.3 to V_{DD} + 0.3 Note	V
Analog input voltage	V_{AN}		–0.3 to V_{DD} + 0.3 Note	V
Output current, high	I_{OH}	Per pin	–10.0	mA
		Total of P20 to P23, P32, P40, P43	–44.0	mA
Output current, low	I_{OL}	Per pin	20.0	mA
		Total of P20 to P23, P32, P40, P43	44.0	mA
Operating ambient temperature	T_A	In normal operation mode	–40 to +85	°C
		During flash memory programming		°C
Storage temperature	T_{stg}	Flash memory blank status	–65 to +150	°C
		Flash memory programming already performed	–40 to +125	°C

Note Must be 6.5 V or lower

Caution Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.

Remark Unless specified otherwise, the characteristics of alternate-function pins are the same as those of port pins.

Oscillator Characteristics (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V[Note 1], V_{SS} = 0 V)

Resonator	Recommended Circuit	Parameter	Conditions	MIN.	TYP.	MAX.	Unit
External clock		External main system clock frequency (f_{EXCLK})[Note 2]	2.7 V ≤ V_{DD} ≤ 5.5 V	2.0		10.0	MHz
			2.0 V ≤ V_{DD} < 2.7 V	2.0		5.0	
		External main system clock input high-/low-level width (t_{EXCLKH}, t_{EXCLKL})	2.7 V ≤ V_{DD} ≤ 5.5 V	0.045		0.25	μs
			2.0 V ≤ V_{DD} < 2.7 V	0.09		0.25	

Notes 1. Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

2. Indicates only oscillator characteristics. Refer to **AC Characteristics** for instruction execution time.

Remark For the resonator selection and oscillator constant, users are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

High-Speed Internal Oscillator Characteristics (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V[Note 1], V_{SS} = 0 V)

Resonator	Parameter	Conditions		MIN.	TYP.	MAX.	Unit
High-speed internal oscillator	Oscillation frequency (f_X)[Note 2]	2.7 V ≤ V_{DD} ≤ 5.5 V	T_A = –10 to +70°C	7.84	8.00	8.16	MHz
		2.0 V ≤ V_{DD} < 2.7 V	T_A = –40 to +85°C		T.B.D		MHz
		2.0 V ≤ V_{DD} < 2.7 V		5.5			MHz

Notes 1. Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

2. Indicates only oscillator characteristics. Refer to **AC Characteristics** for instruction execution time.

Low-Speed Internal Oscillator Characteristics (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V[Note], V_{SS} = 0 V)

Resonator	Parameter	Conditions	MIN.	TYP.	MAX.	Unit
Low-speed internal oscillator	Oscillation frequency (f_{RL})		120	240	480	kHz

Note Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

DC Characteristics (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V Note, V_{SS} = 0 V) (1/2)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Output current, high	I_{OH}	Per pin	2.0 V ≤ V_{DD} ≤ 5.5 V			–5	mA
		Total of all pins	4.0 V ≤ V_{DD} ≤ 5.5 V			–25	mA
			2.0 V ≤ V_{DD} < 4.0 V			–15	mA
Output current, low	I_{OL}	Per pin	2.0 V ≤ V_{DD} ≤ 5.5 V			10	mA
		Total of all pins	4.0 V ≤ V_{DD} ≤ 5.5 V			30	mA
			2.0 V ≤ V_{DD} < 4.0 V			15	mA
Input voltage, high	V_{IH1}	P23 in external clock mode and pins other than P20 and P21		0.8V_{DD}		V_{DD}	V
	V_{IH2}	P23 in other than external clock mode, P20 and P21		0.7V_{DD}		V_{DD}	V
Input voltage, low	V_{IL1}	P23 in external clock mode and pins other than P20 and P21		0		0.2V_{DD}	V
	V_{IL2}	P23 in other than external clock mode, P20 and P21		0		0.3V_{DD}	V
Output voltage, high	V_{OH}	Total of output pins I_{OH} = –15 mA	4.0 V ≤ V_{DD} ≤ 5.5 V I_{OH} = –5 mA	V_{DD} – 1.0			V
		I_{OH} = –100 μA	2.0 V ≤ V_{DD} < 4.0 V	V_{DD} – 0.5			V
Output voltage, low	V_{OL}	Total of output pins I_{OL} = 30 mA	4.0 V ≤ V_{DD} ≤ 5.5 V I_{OL} = 10 mA			1.3	V
		2.0 V ≤ V_{DD} < 4.0 V I_{OL} = 400 μA				0.4	V
Input leakage current, high	I_{LIH}	V_I = V_{DD}	Pins other than EXCLK			1	μA
Input leakage current, low	I_{LIL}	V_I = 0 V	Pins other than EXCLK			–1	μA
Output leakage current, high	I_{LOH}	V_O = V_{DD}	Pins other than EXCLK			1	μA
Output leakage current, low	I_{LOL}	V_O = 0 V	Pins other than EXCLK			–1	μA
Pull-up resistance value	R_{PU}	V_I = 0 V		10	30	100	kΩ
		V_I = 0 V (P34, reset status)		10	30	100	kΩ

Note Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

Remark Unless specified otherwise, the characteristics of alternate-function pins are the same as those of port pins.

DC Characteristics (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V[Note 1], V_{SS} = 0 V) (2/2)

Parameter	Symbol	Conditions			MIN.	TYP.	MAX.	Unit
Supply current[Note 2]	I_{DD1}[Note 3]	External clock input oscillation operating mode[Note 6]	f_X = 10 MHz, V_{DD} = 5.0 V ±10%[Note 4]			6.1	12.2	mA
			f_X = 6 MHz, V_{DD} = 5.0 V ±10%[Note 4]			5.5	11.0	mA
			f_X = 5 MHz, V_{DD} = 3.0 V ±10%[Note 5]			3.0	6.0	mA
	I_{DD2}	External clock input HALT mode[Note 6]	f_X = 10 MHz, V_{DD} = 5.0 V ±10%[Note 4]	When peripheral functions are stopped		1.7	3.8	mA
				When peripheral functions are operating			6.7	
			f_X = 6 MHz, V_{DD} = 5.0 V ±10%[Note 4]	When peripheral functions are stopped		1.3	3.0	mA
				When peripheral functions are operating			6.0	
			f_X = 5 MHz, V_{DD} = 3.0 V ±10%[Note 5]	When peripheral functions are stopped		0.48	1	mA
				When peripheral functions are operating			2.1	
	I_{DD3}[Note 3]	High-speed internal oscillation operating mode[Note 7]	f_X = 8 MHz, V_{DD} = 5.0 V ±10%[Note 4]			5.0	10.0	mA
	I_{DD4}	High-speed internal oscillation HALT mode[Note 7]	f_X = 8 MHz, V_{DD} = 5.0 V ±10%[Note 4]	When peripheral functions are stopped		1.4	3.2	mA
				When peripheral functions are operating			5.9	
	I_{DD5}	STOP mode	V_{DD} = 5.0 V ±10%	When low-speed internal oscillation is stopped		3.5	20.0	μA
				When low-speed internal oscillation is operating		17.5	32.0	
			V_{DD} = 3.0 V ±10%	When low-speed internal oscillation is stopped		3.5	15.5	μA
				When low-speed internal oscillation is operating		11.0	26.0	

Notes 1. Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

2. Total current flowing through the internal power supply (V_{DD}). However, the current that flows through the pull-up resistors of ports is not included.

3. I_{DD1} and I_{DD3} includ peripheral operation current.

4. When the processor clock control register (PCC) is set to 00H.

5. When the processor clock control register (PCC) is set to 02H.

6. When external clock input is selected as the system clock source using the option byte.

7. When high-speed internal oscillation clock is selected as the system clock source using the option byte.

AC Characteristics

Basic operation (T_A = –40 to +85°C, V_{DD} = 2.0 to 5.5 V[Note], V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Cycle time (minimum instruction execution time)	T_{CY}	External clock input	4.0 V ≤ V_{DD} ≤ 5.5 V	0.2		16	μs
			3.0 V ≤ V_{DD} < 4.0 V	0.33		16	μs
			2.7 V ≤ V_{DD} < 3.0 V	0.4		16	μs
			2.0 V ≤ V_{DD} < 2.7 V	1		16	μs
		High-speed internal oscillation clock	4.0 V ≤ V_{DD} ≤ 5.5 V	0.23		4.22	μs
			2.7 V ≤ V_{DD} < 4.0 V	0.47		4.22	μs
			2.0 V ≤ V_{DD} < 2.7 V	0.95		4.22	μs
Interrupt input high-level width, low-level width	t_{INTH}, t_{INTL}			1			μs
$\overline{\text{RESET}}$ input low-level width	t_{RSL}			2			μs

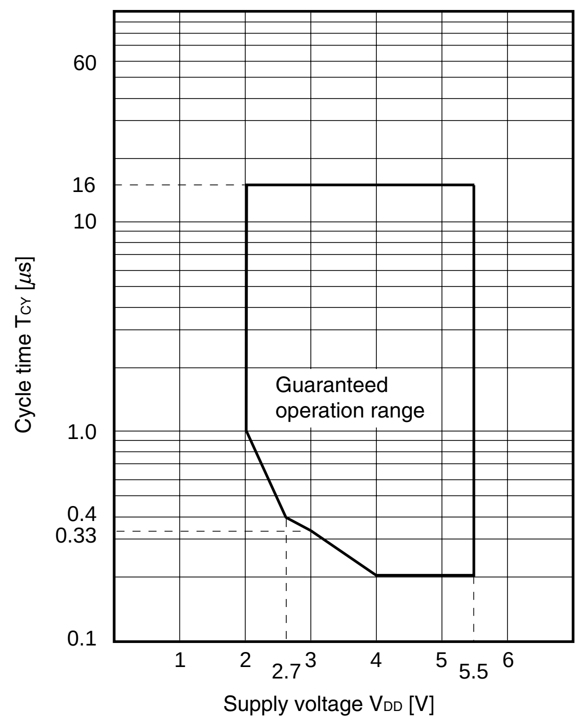
Note Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on clear (POC) circuit is 2.1 V ±0.1 V.

CPU Clock Frequency, Peripheral Clock Frequency

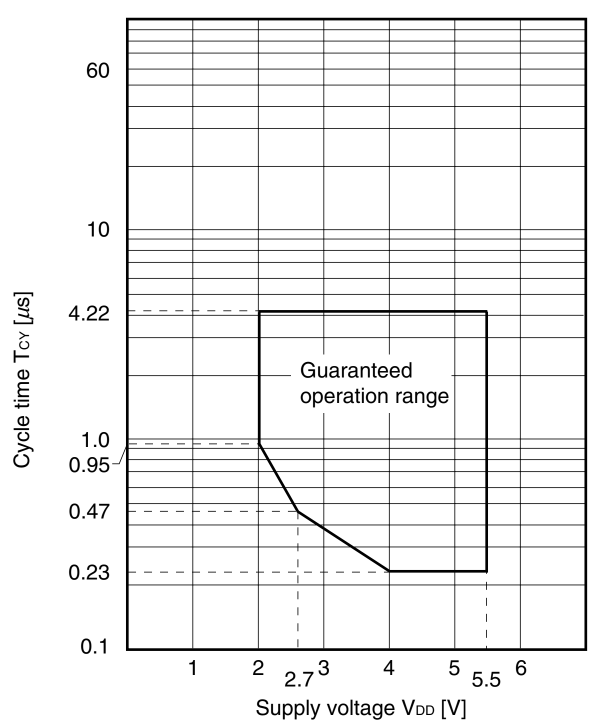
Parameter	Conditions	CPU Clock (f_{CPU})	Peripheral Clock (f_{XP})
External clock	4.0 V ≤ V_{DD} ≤ 5.5 V	125 kHz ≤ f_{CPU} ≤ 10 MHz	500 kHz ≤ f_{XP} ≤ 10 MHz
	3.0 V ≤ V_{DD} < 4.0 V	125 kHz ≤ f_{CPU} ≤ 6 MHz	
	2.7 V ≤ V_{DD} < 3.0 V	125 kHz ≤ f_{CPU} ≤ 5 MHz	
	2.0 V ≤ V_{DD} < 2.7 V[Note]	125 kHz ≤ f_{CPU} ≤ 2 MHz	500 kHz ≤ f_{XP} ≤ 5 MHz
High-speed internal oscillator	4.0 V ≤ V_{DD} ≤ 5.5 V	500 kHz (Typ.) ≤ f_{CPU} ≤ 8 MHz (Typ.)	2 MHz (Typ.) ≤ f_{XP} ≤ 8 MHz (Typ.)
	2.7 V ≤ V_{DD} < 4.0 V	500 kHz (Typ.) ≤ f_{CPU} ≤ 4 MHz (Typ.)	
	2.0 V ≤ V_{DD} < 2.7 V[Note]	500 kHz (Typ.) ≤ f_{CPU} ≤ 2 MHz (Typ.)	2 MHz (Typ.) ≤ f_{XP} ≤ 4 MHz (Typ.)

Note Use this product in a voltage range of 2.2 to 5.5 V because the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

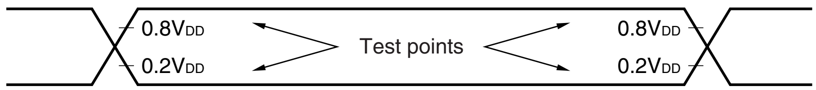
T_{CY} vs. V_{DD} (External Clock Input)



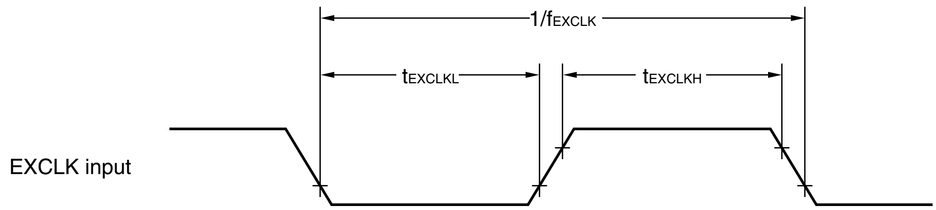
T_{CY} vs. V_{DD} (High-speed internal oscillator Clock)



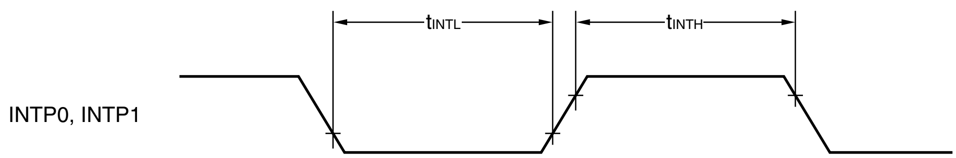
AC Timing Test Points (Excluding EXCLK Input)



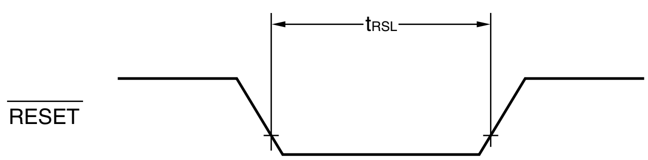
Clock Timing



Interrupt Input Timing



$\overline{\text{RESET}}$ Input Timing



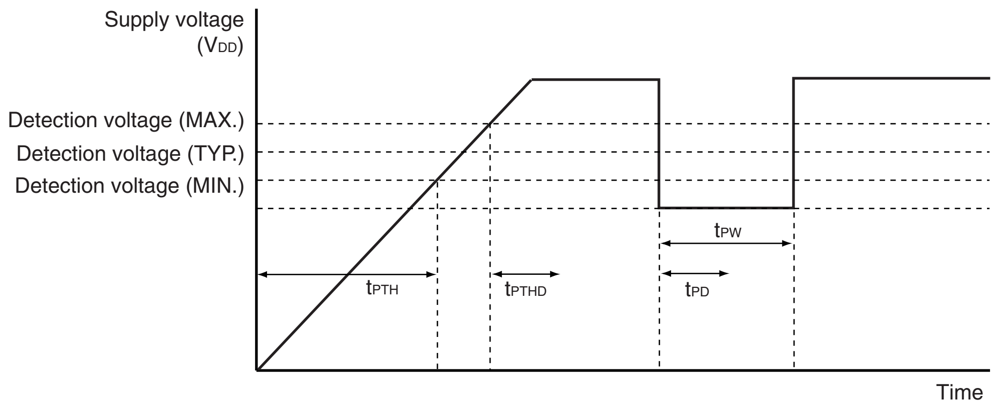
POC Circuit Characteristics (T_A = –40 to +85°C)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Detection voltage	V_{POC}		2.0	2.1	2.2	V
Power supply rise time	t_{PTH}	V_{DD}: 0 V → 2.1 V	1.5			μs
Response delay time 1[Note 1]	t_{PTHD}	When power supply rises, after reaching detection voltage (MAX.)			3.0	ms
Response delay time 2[Note 2]	t_{PD}	When power supply falls			1.0	ms
Minimum pulse width	t_{PW}		0.2			ms

Notes 1. Time required from voltage detection to internal reset release.

2. Time required from voltage detection to internal reset signal generation.

POC Circuit Timing



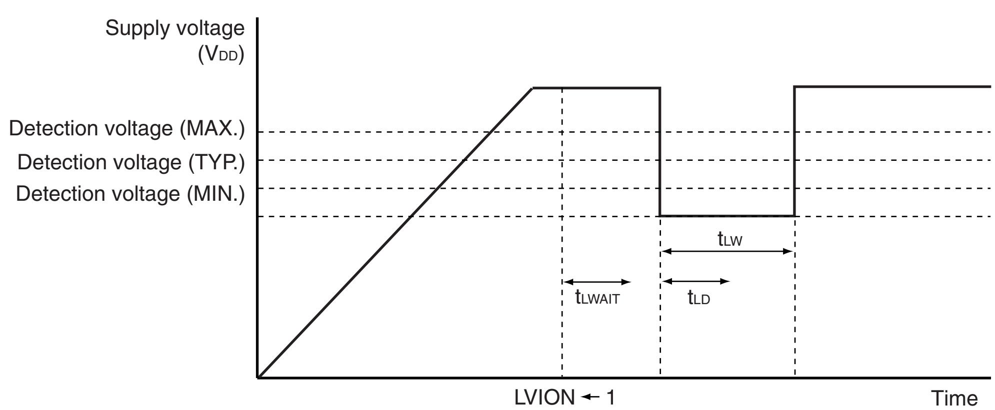
LVI Circuit Characteristics (T_A = –40 to +85°C)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Detection voltage	V_{LVI0}		4.1	4.3	4.5	V
	V_{LVI1}		3.9	4.1	4.3	V
	V_{LVI2}		3.7	3.9	4.1	V
	V_{LVI3}		3.5	3.7	3.9	V
	V_{LVI4}		3.3	3.5	3.7	V
	V_{LVI5}		3.15	3.3	3.45	V
	V_{LVI6}		2.95	3.1	3.25	V
	V_{LVI7}		2.7	2.85	3.0	V
	V_{LVI8}		2.5	2.6	2.7	V
	V_{LVI9}		2.25	2.35	2.45	V
Response time[Note 1]	t_{LD}			0.2	2.0	ms
Minimum pulse width	t_{LW}		0.2			ms
Operation stabilization wait time[Note 2]	t_{LWAIT}			0.1	0.2	ms

Notes 1. Time required from voltage detection to interrupt output or internal reset signal generation.
2. Time required from setting LVION to 1 to operation stabilization.

Remarks 1. $V_{LVI0} > V_{LVI1} > V_{LVI2} > V_{LVI3} > V_{LVI4} > V_{LVI5} > V_{LVI6} > V_{LVI7} > V_{LVI8} > V_{LVI9}$
2. $V_{POC} < V_{LVIm}$ (m = 0 to 9)

LVI Circuit Timing



Data Memory STOP Mode Low Supply Voltage Data Retention Characteristics (T_A = –40 to +85°C)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Data retention supply voltage	V_{DDDR}		2.0		5.5	V
Release signal set time	t_{SREL}		0			μs

Flash Memory Programming Characteristics (T_A = –40 to +85°C, 2.7 V ≤ V_{DD} ≤ 5.5 V, V_{SS} = 0 V)

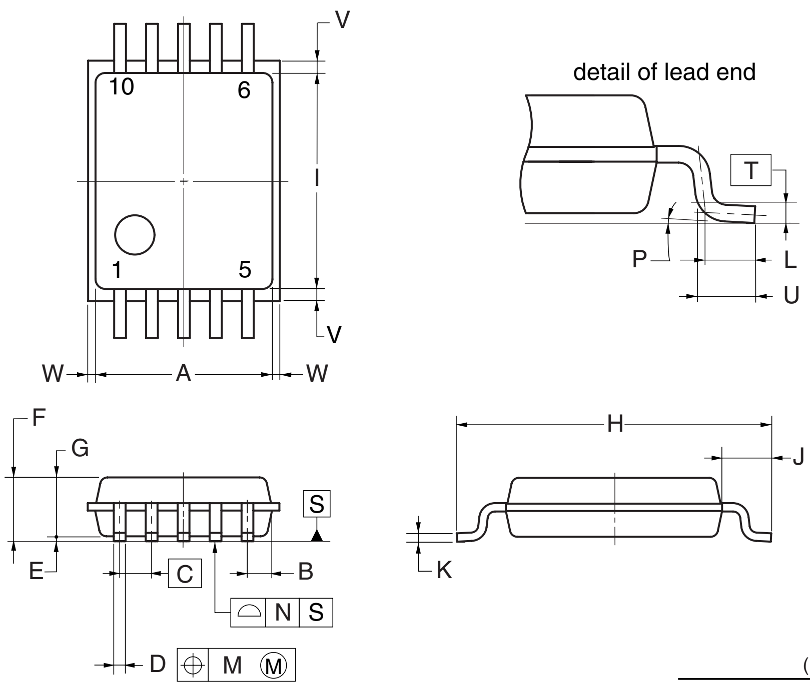
Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Supply current	I_{DD}	V_{DD} = 5.5 V				7.0	mA
Erasure count[Note] (per 1 block)	N_{ERASE}	T_A = –40 to +85°C		1000			Times
Chip erase time	T_{CERASE}	T_A = –10 to +85°C, N_{ERASE} ≤ 100	4.5 V ≤ V_{DD} ≤ 5.5 V			0.8	s
			3.5 V ≤ V_{DD} < 4.5 V			1.0	s
			2.7 V ≤ V_{DD} < 3.5 V			1.2	s
		T_A = –10 to +85°C, N_{ERASE} ≤ 1000	4.5 V ≤ V_{DD} ≤ 5.5 V			4.8	s
			3.5 V ≤ V_{DD} < 4.5 V			5.2	s
			2.7 V ≤ V_{DD} < 3.5 V			6.1	s
		T_A = –40 to +85°C, N_{ERASE} ≤ 100	4.5 V ≤ V_{DD} ≤ 5.5 V			1.6	s
			3.5 V ≤ V_{DD} < 4.5 V			1.8	s
			2.7 V ≤ V_{DD} < 3.5 V			2.0	s
		T_A = –40 to +85°C, N_{ERASE} ≤ 1000	4.5 V ≤ V_{DD} ≤ 5.5 V			9.1	s
			3.5 V ≤ V_{DD} < 4.5 V			10.1	s
			2.7 V ≤ V_{DD} < 3.5 V			12.3	s
Block erase time	T_{BERASE}	T_A = –10 to +85°C, N_{ERASE} ≤ 100	4.5 V ≤ V_{DD} ≤ 5.5 V			0.4	s
			3.5 V ≤ V_{DD} < 4.5 V			0.5	s
			2.7 V ≤ V_{DD} < 3.5 V			0.6	s
		T_A = –10 to +85°C, N_{ERASE} ≤ 1000	4.5 V ≤ V_{DD} ≤ 5.5 V			2.6	s
			3.5 V ≤ V_{DD} < 4.5 V			2.8	s
			2.7 V ≤ V_{DD} < 3.5 V			3.3	s
		T_A = –40 to +85°C, N_{ERASE} ≤ 100	4.5 V ≤ V_{DD} ≤ 5.5 V			0.9	s
			3.5 V ≤ V_{DD} < 4.5 V			1.0	s
			2.7 V ≤ V_{DD} < 3.5 V			1.1	s
		T_A = –40 to +85°C, N_{ERASE} ≤ 1000	4.5 V ≤ V_{DD} ≤ 5.5 V			4.9	s
			3.5 V ≤ V_{DD} < 4.5 V			5.4	s
			2.7 V ≤ V_{DD} < 3.5 V			6.6	s
Byte write time	T_{WRITE}	T_A = –40 to +85°C, N_{ERASE} ≤ 1000				150	μs
Internal verify	T_{VERIFY}	Per 1 block				6.8	ms
		Per 1 byte				27	μs
Blank check	T_{BLKCHK}	Per 1 block				480	μs
Retention years		T_A = 85°C[Note 2], N_{ERASE} ≤ 1000		10			Years

Notes 1. Depending on the erasure count (N_{ERASE}), the erase time varies. Refer to the chip erase time and block erase time parameters.

2. When the average temperature when operating and not operating is 85°C.

Remark When a product is first written after shipment, "erase → write" and "write only" are both taken as one rewrite.

10-PIN PLASTIC SSOP (5.72 mm (225))



(UNIT:mm)

ITEM	DIMENSIONS
A	3.60±0.10
B	0.50
C	0.65 (T.P.)
D	0.24±0.08
E	0.10±0.05
F	1.45 MAX.
G	1.20±0.10
H	6.40±0.20
I	4.40±0.10
J	1.00±0.20
K	$0.17^{+0.08}_{-0.07}$
L	0.50
M	0.13
N	0.10
P	$3^{\circ}{}^{+5^{\circ}}_{-3^{\circ}}$
T	0.25 (T.P.)
U	0.60±0.15
V	0.25 MAX.
W	0.15 MAX.

P10MA-65-CAC

NOTE

Each lead centerline is located within 0.13 mm of its true position (T.P.) at maximum material condition.

CHAPTER 18 RECOMMENDED SOLDERING CONDITIONS

These products should be soldered and mounted under the following recommended conditions.
For technical information, see the following website.

Semiconductor Device Mount Manual (http://www.necel.com/pkg/en/mount/index.html)

Cautions 1. Products whis –A at the end of the part number are lead-free products.
2. For soldering methods and conditions other than those recommended below, contact an NEC Electronics sales representative.

Table 18-1. Surface Mounting Type Soldering Conditions

10-pin plastic SSOP (lead-free products)
μPD78F9500MA-CAC-A, 78F9501MA-CAC-A, 78F9502MA-CAC-A

Soldering Method	Soldering Conditions	Recommended Condition Symbol
Infrared reflow	Package peak temperature: 260°C, Time: 30 seconds max. (at 210°C or higher), Count: 3 times or less, Exposure limit: 7 days[Note] (after that, prebake at 125°C for 10 to 72 hours)	IR60-107-3
Wave soldering	For details, contact an NEC Electronics sales representative.	–
Partial heating	Pin temperature: 350°C max., Time: 3 seconds max. (per pin row)	–

Note After opening the dry pack, store it at 25°C or less and 65% RH or less for the allowable storage period.

Caution Do not use different soldering methods together (except for partial heating).

APPENDIX A DEVELOPMENT TOOLS

The following development tools are available for development of systems using the μPD78F9500, 78F9501, 78F9502. Figure A-1 shows development tools.

- Compatibility with PC98-NX series

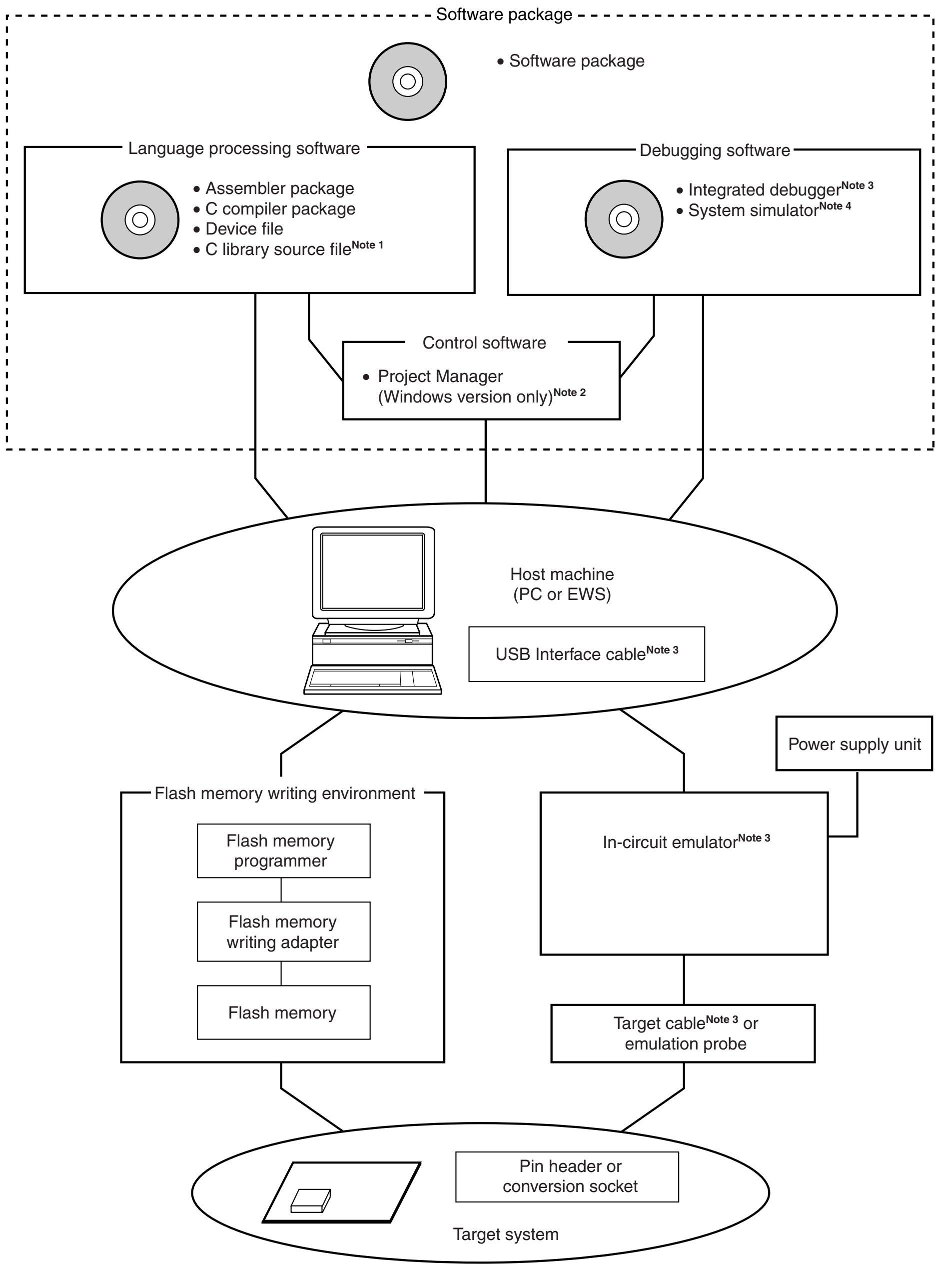
  Unless stated otherwise, products which are supported by IBM PC/AT™ and compatibles can also be used with the PC98-NX series. When using the PC98-NX series, therefore, refer to the explanations for IBM PC/AT and compatibles.

- Windows™

  Unless stated otherwise, "Windows" refers to the following operating systems.
  - Windows 98
  - Windows NT™ Ver. 4.0
  - Windows 2000
  - Windows XP™

Figure A-1. Development Tools (1/2)

(1) When using the in-circuit emulator QB-78K0SKX1



Notes 1. The C library source file is not included in the software package.

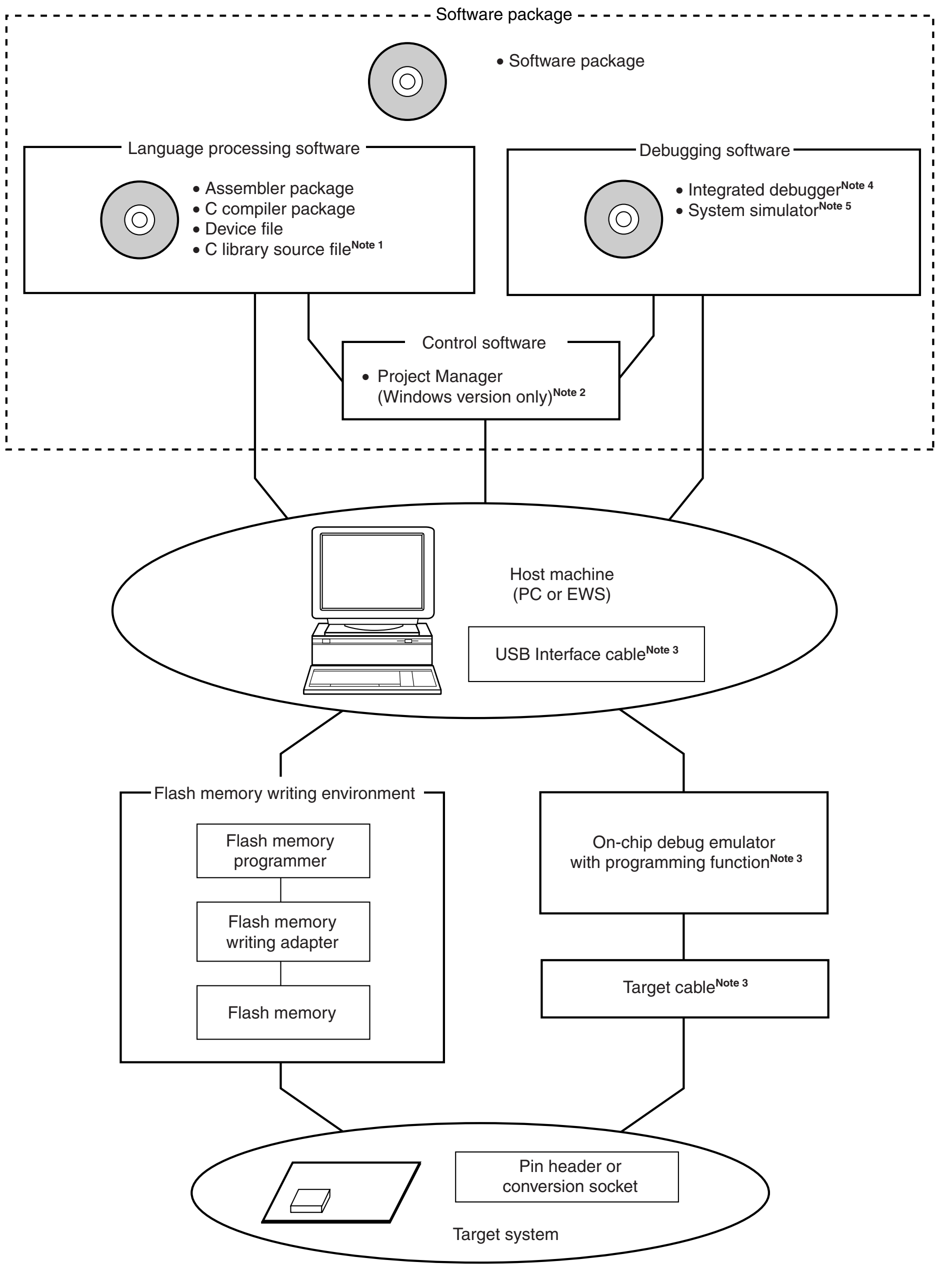
2. The Project Manager PM+ is included in the assembler package.
PM+ is used only in the Windows environment.

3. The in-circuit emulator QB-78K0SKX1 is provided with the integrated debugger ID78K0S-QB, the on-chip debug emulator with programming function QB-MINI2, a USB interface cable, a power supply unit, and a target cable. Other products are optional.

4. It is planned to use SM+ for 78K0S/Kx1+ as a system simulator.

Figure A-1. Development Tools (2/2)

(2) When using the on-chip debug emulator with programming function QB-MINI2



Notes 1. The C library source file is not included in the software package.

2. The Project Manager PM+ is included in the assembler package.
PM+ is used only in the Windows environment.

3. The on-chip debug emulator with programming function QB-MINI2 is provided with a USB interface cable, and a target cable.

4. The integrated debugger ID78K0S-QB is not included with the QB-MINI2.
The integrated debugger ID78K0S-QB is available on the following website.
http://www.necel.com/micro/ods/eng/

5. It is planned to use SM+ for 78K0S/Kx1+ as a system simulator.

A.1 Software Package

SP78K0S Software package	This is a package that bundles the software tools required for development of the 78K/0S Series. The following tools are included. RA78K0S, CC78K0S, ID78K0S-NS, SM+ for 78K0S[Note 1], SM78K0S[Notes 2], and device files
	Part number: μS××××SP78K0S

Notes 1. SM+ for 78K0S is not included in SP78K0S Ver. 2.00 or earlier.

2. The SM78K0S does not support the 78K0S/Kx1+.

3. The DF789234 is not included in SP78K0S Ver. 2.00 or earlier. The DF789234 is available on the following website.

http://www.necel.com/micro/ods/eng/

Remark ×××× in the part number differs depending on the operating system to be used.

μS××××SP78K0S

××××	Host Machine	OS	Supply Medium
AB17	PC-9800 series, IBM PC/AT and compatibles	Japanese Windows	CD-ROM
BB17		English Windows	

A.2 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 device file (DF789234) (sold separately). **<Caution when used in PC environment>** The assembler package is a DOS-based application but may be used under the Windows environment by using PM+ of Windows (included in the assembler package).
	Part number: μS××××RA78K0S
CC78K0S C library package	Program that converts program written in C language into object codes that can be executed by microcontroller. Used in combination with assembler package (RA78K0S) and device file (DF789234) (both sold separately). **<Caution when used in PC environment>** The C compiler package is a DOS-based application but may be used under the Windows environment by using PM+ of Windows (included in the assembler package).
	Part number: μS××××CC78K0S
DF789234[Note 1] Device file	File containing the information inherent to the device. Used in combination with other tools (RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, or SM+ for 78K0S) (all sold separately).
	Part number: μS××××DF789234
CC78K0S-L[Note 2] C library 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

Notes 1. DF789234 is a common file that can be used with RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB and SM+ for 78K0S.

2. CC78K0S-L is not included in the software package (SP78K0S).

Remark ×××× in the part number differs depending on the host machine and operating system to be used.

μS××××RA78K0S
μS××××CC78K0S
μS××××CC78K0S-L

××××	Host Machine	OS	Supply Media
AB17	PC-9800 series, IBM PC/AT and compatibles	Japanese Windows	CD-ROM
BB17		English Windows	
3P17	HP9000 series 700™	HP-UX™ (Rel.10.10)	
3K17	SPARCstation™	SunOS™ (Rel.4.1.4), Solaris™ (Rel.2.5.1)	

μS××××DF789234

××××	Host Machine	OS	Supply Media
AB13	PC-9800 series, IBM PC/AT and compatibles	Japanese Windows	3.5" 2HD FD
BB13		English Windows	

A.3 Control Software

PM+ Project manager	This is control software designed so that the user program can be efficiently developed in the Windows environment. With this software, a series of user program development operations, including starting the editor, build, and starting the debugger, can be executed on the PM+. **<Caution>** The PM+ is included in the assembler package (RA78K0S). It can be used only in the Windows environment.

A.4 Flash Memory Writing Tools

FlashPro4 (FL-PR4, PG-FP4) Flash memory programmer	Flash memory programmer dedicated to the microcontrollers incorporating a flash memory
QB-MINI2 On-chip debug emulator with programming function	This is a flash memory programmer dedicated to microcontrollers incorporating a flash memory. It is available also as an on-chip debug emulator which serves to debug hardware and software when developing application systems using all flash microcontrollers (including the 78K0S/Kx1+).
FA-78F9202MA-CAC-MX (under development) Flash memory writing adapter	Flash memory writing adapter. Used in connection with the flash memory programmer.

Remark FL-PR4 and FA-78F9202MA-CAC-MX are products of Naito Densei Machida Mfg. Co., Ltd.
For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (TEL +81-42-750-4172)

A.5 Debugging Tools (Hardware)

A.5.1 When using in-circuit emulator QB-78K0SKX1

QB-78K0SKX1 In-circuit emulator	This in-circuit emulator serves to debug hardware and software when developing application systems using the 78K0S/Kx1+. It supports the integrated debugger (ID78K0S-QB). It is connected to the included AC adapter, target cable, and host machine via a USB interface cable.
QB-50-EP-01T Emulation probe	This emulation probe is a flexible type and is used to connect the in-circuit emulator and target system.
QB-10MA-EA-01T Exchange adapter	This exchange adapter is used to perform pin conversion from the in-circuit emulator to target connector.
QB-10MA-NQ-01T Target connector	This target connector is used to mount on the target system.
Specifications of pin header on target system	0.635 mm × 0.635 mm (height: 6 mm)

Remark The QB-78K0SKX1 is provided with a AC adapter, a USB interface cable, a target cable, the integrated debugger ID78K0S-QB, and the on-chip debug emulator with programming function QB-MINI2.
The QB-78K0SKX1 is also provided with an emulation probe, an exchange adapter, and a target connector depending on the ordering code.

A.5.2 When using in-circuit emulator QB-MINI2

QB-MINI2 On-chip debug emulator with programming function	This is an on-chip debug emulator which serves to debug hardware and software when developing application systems using all flash microcontrollers (including the 78K0S/Kx1+). It is available also as a flash memory programmer dedicated to microcontrollers incorporating a flash memory.
Specifications of pin header on target system	16-pin general-purpose connector (2.54 mm pitch)

Remark The QB-MINI2 is provided with a USB interface cable, and a target cable. In addition, the integrated debugger (including ID78K0S-QB) is used as control software. The integrated debugger is available on the following website.
http://www.necel.com/micro/ods/eng/

A.6 Debugging Tools (Software)

ID78K0S-QB (supporting in-circuit emulator QB-78K0SKX1/ QB-78K0SKX1MINI, and on-chip debug emulator with programming function QB-MINI2)	This debugger supports the in-circuit emulators for the 78K0S/Kx1+ Series. ID78K0S-QB is Windows-based software. Provided with the debug function supporting C language, source programming, disassemble display, and memory display are possible. This is used with the device file (DF789234) (sold separately). It is provided with the in-circuit emulator QB-78K0SKX1/QB-78K0SKX1MINI.
	Ordering number: μS××××ID78K0S-QB (not for sale)
SM+ for 78K0S/Kx1+[Note 1] System simulator	This is a system simulator for the 78K/0S series. SM+ for 78K0S/Kx1+ is Windows-based software. This simulator can execute C-source-level or assembler-level debugging while simulating the operations of the target system on the host machine. By using SM+ for 78K0S/Kx1+, the logic and performance of the application can be verified independently of hardware development. Therefore, the development efficiency can be enhanced and the software quality can be improved. This simulator is used with a device file (DF789234) (sold separately).
	Part number: μS××××SM789234-B
DF789234[Note 2] Device file	This is a file that has device-specific information. It is used with the RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, and SM+ for 78K0S (all sold separately).
	Part number: μS××××DF789234

Note 1. SM+ for 78K0S/Kx1+ is currently being considered as the system simulator for the μPD78F9500, 78F9501, 78F9502.

2. DF789234 is a common file that can be used with the RA78K0S, CC78K0S, ID78K0S-NS, ID78K0S-QB, and SM+ for 78K0S.

APPENDIX B REGISTER INDEX

B.1 Register Index (Register Name)

B.2 Register Index (Symbol)

APPENDIX C LIST OF CAUTIONS

This appendix lists cautions described in this document.
"Classification (hard/soft)" in table is as follows.

Hard: Cautions for microcontroller internal/external hardware
Soft: Cautions for software such as register settings or programs

(1/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 2	Hard	Pin functions	P22, P23/X1/ANI3, P34/RESET	The P22 and P23/EXCLK pins are pulled down during reset. The P34/RESET pin is pulled up during reset by the reset pin function/power-on clear circuit.	pp. 19-21 ☐
Chapter 3	Soft	Memory space	SP: stack pointer	Since reset signal generation makes the SP contents undefined, be sure to initialize the SP before using the stack memory.	p. 32 ☐
				Stack pointers can be set only to the high-speed RAM area, and only the lower 10 bits can be actually set. 0FF00H is in the SFR area, not the high-speed RAM area, so it was converted to 0FB00H that is in the high-speed RAM area. When the value is actually pushed onto the stack, 1 is subtracted from 0FB00H to become 0FAFFH, but that value is not in the high-speed RAM area, so it is converted to 0FEFFH, which is the same value as when 0FF00H is set to the stack pointer.	p. 32 ☐
Chapter 4	Hard	Port functions	P22, P23/X1/ANI3, P34/RESET	The P22 and P23/EXCLK pins are pulled down during reset. The P34/RESET pin is pulled up during reset by the reset pin function/power-on clear circuit.	p. 48 ☐
			P34	Because the P34 pin functions alternately as the RESET pin, if it is used as an input port pin, the function to input an external reset signal to the RESET pin cannot be used. The function of the port is selected by the option byte. For details, refer to CHAPTER 13 OPTION BYTE. Also, since the option byte is referenced after the reset release, if low level is input to the RESET pin before the referencing, then the reset state is not released. When it is used as an input port pin, connect the pull-up resistor.	p. 54 ☐
			P21, P32	Because P21 and P32 are also used as external interrupt pins, the corresponding interrupt request flag is set if each of these pins is set to the output mode and its output level is changed. To use the port pin in the output mode, therefore, set the corresponding interrupt mask flag to 1 in advance.	p. 56 ☐
			–	Although a 1-bit memory manipulation instruction manipulates 1 bit, it accesses a port in 8-bit units. Therefore, the contents of the output latch of a pin in the input mode, even if it is not subject to manipulation by the instruction, are undefined in a port with a mixture of inputs and outputs.	p. 59 ☐
Chapter 6	Soft	8-bit timer H1	CMP01: 8-bit timer H compare register 01	CMP01 cannot be rewritten during timer count operation.	p. 74 ☐

(2/7)

Chapter	Classification	Function	Details of Function	Cautions	Page	
Chapter 6	Soft	8-bit timer H1	CMP11: 8-bit timer H compare register 11	In the PWM output mode, be sure to set CMP11 when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to CMP11).	p. 74	□
			TMHMD1: 8-bit timer H mode register 1	When TMHE1 = 1, setting the other bits of the TMHMD1 register is prohibited.	p. 76	□
				In the PWM output mode, be sure to set 8-bit timer H compare register 11 (CMP11) when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).	p. 76	□
	Hard		PWM output	In PWM output mode, the setting value for the CMP11 register can be changed during timer count operation. However, three operation clocks (signal selected using the CKS12 to CKS10 bits of the TMHMD1 register) or more are required to transfer the register value after rewriting the CMP11 register value.	p. 82	□
	Soft			Be sure to set the CMP11 register when starting the timer count operation (TMHE1 = 1) after the timer count operation was stopped (TMHE1 = 0) (be sure to set again even if setting the same value to the CMP11 register).	p. 82	□
				Make sure that the CMP11 register setting value (M) and CMP01 register setting value (N) are within the following range. 00H ≤ CMP11 (M) < CMP01 (N) ≤ FFH	p. 82	□
Chapter 7	Soft	Watchdog timer	WDTM: Watchdog timer mode register	Set bits 7, 6, and 5 to 0, 1, and 1, respectively. Do not set the other values.	p. 90	□
				After reset is released, WDTM can be written only once by an 8-bit memory manipulation instruction. If writing is attempted a second time, an internal reset signal is generated. However, at the first write, if “1” and “x” are set for WDCS4 and WDCS3 respectively and the watchdog timer is stopped, then the internal reset signal does not occur even if the following are executed. • Second write to WDTM • 1-bit memory manipulation instruction to WDTE • Writing of a value other than “ACH” to WDTE	p. 91	□
				WDTM cannot be set by a 1-bit memory manipulation instruction.	p. 91	□
				When using the flash memory self programming by self writing, set the overflow time for the watchdog timer so that enough everflow time is secured (Example 1-byte writing: 200 μs MIN., 1-block deletion: 10 ms MIN.).	p. 91	□
			WDTE: Watchdog timer enable register	If a value other than ACH is written to WDTE, an internal reset signal is generated.	p. 91	□
				If a 1-bit memory manipulation instruction is executed for WDTE, an internal reset signal is generated.	p. 91	□
				The value read from WDTE is 9AH (this differs from the written value (ACH)).	p. 91	□
	Hard		When “low-speed internal oscillator cannot be stopped” is selected by option byte	In this mode, operation of the watchdog timer cannot be stopped even during STOP instruction execution. For 8-bit timer H1 (TMH1), a division of the low-speed internal oscillation clock can be selected as the count source, so clear the watchdog timer using the interrupt request of TMH1 before the watchdog timer overflows after STOP instruction execution. If this processing is not performed, an internal reset signal is generated when the watchdog timer overflows after STOP instruction execution.	p. 92	□
			when “low-speed internal oscillator can be stopped by software” is selected by option byte	In this mode, watchdog timer operation is stopped during HALT/STOP instruction execution. After HALT/STOP mode is released, counting is started again using the operation clock of the watchdog timer set before HALT/STOP instruction execution by WDTM. At this time, the counter is not cleared to 0 but holds its value.	p. 94	□

(3/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 8	Soft	Interrupt functions	IF0: Interrupt request flag registers, MK0: Interrupt mask flag registers	Because P21 and P32 have an alternate function as external interrupt inputs, when the output level is changed by specifying the output mode of the port function, an interrupt request flag is set. Therefore, the interrupt mask flag should be set to 1 before using the output mode.	pp. 100, 101 ☐
			INTM0: External interrupt mode register 0	Be sure to clear bits 0, 1, 6, and 7 to 0.	p. 101 ☐
				Before setting the INTM0 register, be sure to set the corresponding interrupt mask flag (××MK× = 1) to disable interrupts. After setting the INTM0 register, clear the interrupt request flag (××IF× = 0), then clear the interrupt mask flag (××MK× = 0), which will enable interrupts.	p. 102 ☐
			Interrupt requests are held pending	Interrupt requests will be held pending while the interrupt request flag registers (IF0) or interrupt mask flag registers (MK0) are being accessed.	p. 104 ☐
			Interrupt request pending	Multiple interrupts can be acknowledged even for low-priority interrupts.	p. 105 ☐
Chapter 9	Soft	Standby Function	–	The LSRSTOP setting is valid only when “Can be stopped by software” is set for the low-speed internal oscillator by the option byte.	p. 107 ☐
	Hard		STOP mode	When shifting to the STOP mode, be sure to stop the peripheral hardware operation before executing STOP instruction (except the peripheral hardware that operates on the low-speed internal oscillation clock).	p. 108 ☐
			STOP mode	If the low-speed internal oscillator is operating before the STOP mode is set, oscillation of the low-speed internal oscillation clock cannot be stopped in the STOP mode (refer to Table 9-1).	p. 108 ☐
	Soft		HALT mode setting and operating statuses	Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag clear, the standby mode is immediately cleared if set.	p. 109 ☐
			STOP mode setting and operating statuses	Because an interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. Thus, in the STOP mode, the normal operation mode is restored after the STOP instruction is executed and then the operation is stopped for 34 μs (TYP.).	p. 111 ☐
Chapter 10	Hard	Reset function	–	For an external reset, input a low level for 2 μs or more to the $\overline{\text{RESET}}$ pin.	p. 114 ☐
				During reset signal generation, the system clock and low-speed internal oscillation clock stop oscillating.	p. 114 ☐
				When the $\overline{\text{RESET}}$ pin is used as an input-only port pin (P34), the μPD78F9500, 78F9501, 78F9502 is reset if a low level is input to the $\overline{\text{RESET}}$ pin after reset is released by the POC circuit and before the option byte is referenced again. The reset status is retained until a high level is input to the $\overline{\text{RESET}}$ pin.	p. 114 ☐
				The LVI circuit is not reset by the internal reset signal of the LVI circuit.	p. 115 ☐
			Timing of reset by overflow of watchdog timer	The watchdog timer is also reset in the case of an internal reset of the watchdog timer.	p. 116 ☐
			RESF: Reset control flag register	Do not read data by a 1-bit memory manipulation instruction.	p. 118 ☐

(4/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 11	Soft	Power-on-clear circuit	Functions of power-on-clear circuit	If an internal reset signal is generated in the POC circuit, the reset control flag register (RESF) is cleared to 00H.	p. 119 □
	Hard			Because the detection voltage (V_{POC}) of the POC circuit is in a range of 2.1 V ±0.1 V, use a voltage in the range of 2.2 to 5.5 V.	p. 119 □
	Soft		Cautions for power-on-clear circuit	In a system where the supply voltage (V_{DD}) fluctuates for a certain period in the vicinity of the POC detection voltage (V_{POC}), the system may be repeatedly reset and released from the reset status. In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking the following action.	p. 121 □
Chapter 12	Soft	Low-voltage detector	LVIM: Low-voltage detect register	To stop LVI, follow either of the procedures below. • When using 8-bit manipulation instruction: Write 00H to LVIM. • When using 1-bit memory manipulation instruction: Clear LVION to 0.	p. 124 □
				Be sure to set bits 2 to 6 to 0.	p. 124 □
			LVIS: Low-voltage detection level select register	Bits 4 to 7 must be set to 0.	p. 125 □
				If a value other than the above is written during LVI operation, the value becomes undefined at the very moment it is written, and thus be sure to stop LVI (bit 7(LVION) = 0 on the LVIM register) before writing.	p. 125 □
			When used as reset	<1> must always be executed. When LVIMK = 0, an interrupt may occur immediately after the processing in <3>.	p. 126 □
				If supply voltage (V_{DD}) ≥ detection voltage (V_{LVI}) when LVIM is set to 1, an internal reset signal is not generated.	p. 126 □
			Cautions for low-voltage detector	In a system where the supply voltage (V_{DD}) fluctuates for a certain period in the vicinity of the LVI detection voltage (V_{LVI}), the operation is as follows depending on how the low-voltage detector is used. <1> When used as reset The system may be repeatedly reset and released from the reset status. In this case, the time from release of reset to the start of the operation of the microcontroller can be arbitrarily set by taking action (1) below. <2> When used as interrupt Interrupt requests may be frequently generated. Take (b) of action (2) below.	p. 130 □
Chapter 13	Hard	Option byte	Control of $\overline{RESET}$ pin	Because the option byte is referenced after reset release, if a low level is input to the $\overline{RESET}$ pin before the option byte is referenced, then the reset state is not released. When used as an input-only port (P34), the setting of the on-chip pull-up resistor can be done by PU34 on PU3 register.	p. 134 □
			Selection of system clock source	Because the EXCLK pin is also used as the P23 pin, the condition under which the EXCLK pin can be used differ depending on the selected system clock source. (1) External clock input is selected Because the pin is used as an external clock input pin, P23 cannot be used as an I/O port pin. (2) High-speed internal oscillation clock is selected P23 pin can be used as an I/O port pin.	p. 134 □

(5/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 13	Hard	Option byte	Low-speed internal oscillates	If it is selected that low-speed internal oscillator cannot be stopped, the count clock to the watchdog timer (WDT) is fixed to low-speed internal oscillation clock.	p. 135 ☐
				If it is selected that low-speed internal oscillator can be stopped by software, supply of the count clock to WDT is stopped in the HALT/STOP mode, regardless of the setting of bit 0 (LSRSTOP) of the low-speed internal oscillation mode register (LSRCM). Similarly, clock supply is also stopped when a clock other than the low-speed internal oscillation clock is selected as a count clock to WDT. While the low-speed internal oscillator is operating (LSRSTOP = 0), the clock can be supplied to the 8-bit timer H1 even in the STOP mode.	p. 135 ☐
			Caution When the $\overline{RESET}$ Pin Is Used as an Import-Only Port Pin (P34)	Be aware of the following when erasing/writing by on-board programming using a dedicated flash memory programmer once again on the already-written device which has been set as "The $\overline{RESET}$ pin is used as an input-only port pin (P34)" by the option byte function. Before supplying power to the target system, connect a dedicated flash memory programmer and turn its power on. If the power is supplied to the target system beforehand, it cannot be switched to the flash memory programming mode.	p. 135 ☐
Chapter 14	Soft	Flash memory	Security settings	After the security setting of the batch erase is set, erasure cannot be performed for the device. In addition, even if a write command is executed, data different from that which has already been written to the flash memory cannot be written because the erase command is disabled.	p. 143 ☐
			Self programming function	Self programming processing must be included in the program before performing self writing.	p. 144 ☐
				No instructions can be executed while a self programming command is being executed. Therefore, clear and restart the watchdog timer counter in advance so that the watchdog timer does not overflow during self programming. Refer to Table 14-8 for the time taken for the execution of self programming.	p. 147 ☐
				Interrupts that occur during self programming can be acknowledged after self programming mode ends. To avoid this operation, disable interrupt servicing (by setting MK0 to FFH, and executing the DI instruction) before a mode is shifted from the normal mode to the self programming mode with a specific sequence.	p. 147 ☐
				RAM is not used while a self programming command is being executed.	p. 147 ☐
				If the supply voltage drops or the reset signal is input while the flash memory is being written or erased, writing/erasing is not guaranteed.	p. 147 ☐
				The value of the blank data set during block erasure is FFH.	p. 147 ☐
				Set the CPU clock so that it is 1 MHz or more during self programming.	p. 147 ☐
				Execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, then execute self programming. At this time, the HALT instruction is automatically released after 10 μs (MAX.) + 2 CPU clocks (f_{CPU}).	p. 147 ☐
				If the clock of the oscillator or an external clock is selected as the system clock, execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, wait for 8 μs after releasing the HALT status, and then execute self programming.	p. 147 ☐
				Check FPRERR using a 1-bit memory manipulation instruction.	p. 147 ☐
				The state of the pins in self programming mode is the same as that in HALT mode.	p. 147 ☐

(6/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 14	Soft	Flash memory	Self programming function	Since the security function set via on-board/off-board programming is disabled in self programming mode, the self programming command can be executed regardless of the security function setting. To disable write or erase processing during self programming, set the protect byte.	p. 147 ☐
				Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.	p. 147 ☐
				Clear the value of the FLCMD register to 00H immediately before setting self-programming mode and normal operation mode.	p. 147 ☐
			FLPMC: Flash programming mode control register	Cautions in the case of setting the self programming mode, refer to 14.8.2 Cautions on self programming function.	p. 148 ☐
				Set the CPU clock so that it is 1 MHz or more during self programming.	p. 148 ☐
				Execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, then execute self programming. At this time, the HALT instruction is automatically released after 10 μs (MAX.) + 2 CPU clocks (f_{CPU}).	p. 148 ☐
				If the clock of the oscillator or an external clock is selected as the system clock, execute the NOP and HALT instructions immediately after executing a specific sequence to set self-programming mode, wait for 8 μs after releasing the HALT status, and then execute self programming.	p. 148 ☐
				Clear the value of the FLCMD register to 00H immediately before setting selfprogramming mode and normal operation mode.	p. 148 ☐
			PFCMD: Flash protect command register	Interrupt servicing cannot be executed in self-programming mode. Disable interrupt servicing (by executing the DI instruction while MK0 and MK1 = FFH) before executing the specific sequence that sets self-programming mode and after executing the specific sequence that changes the mode to the normal mode.	p. 149 ☐
			PFS: Flash status register	Check FPRERR using a 1-bit memory manipulation instruction.	p. 149 ☐
			FLAPH, FLAPL: Flash address pointers H and L	Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.	p. 152 ☐
			FLAPHC, FLAPLC: Flash address pointer H/L compare registers	Be sure to clear bits 4 to 7 of flash address pointer H (FLAPH) and flash address pointer H compare register (FLAPHC) to 0 before executing the self programming command. If the value of these bits is 1 when executing the self programming command.	p. 152 ☐
				Set the number of the block subject to a block erase, verify, or blank check (same value as FLAPH) to FLAPHC.	p. 152 ☐
				Clear FLAPLC to 00H when a block erase is performed, and FFH when a blank check is performed.	p. 152 ☐

(7/7)

Chapter	Classification	Function	Details of Function	Cautions	Page
Chapter 14	Soft	Flash memory	Shifting to self programming mode Shifting to normal mode	Be sure to perform the series of operations described above using the user program at an address where data is not erased nor written.	pp. 154, 155, 157, 158 ☐
			Byte write	If a write results in failure, erase the block once and write to it again.	p. 166 ☐
Chapter 16	Hard	Electrical specifications	Absolute maximum ratings	Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.	p. 200 ☐
Chapter 18	Hard	Recom-mended soldering conditions	Lead-free products	Products whis -A at the end of the part number are lead-free products.	p. 211 ☐
			–	For soldering methods and conditions other than those recommended below, contact an NEC Electronics sales representative.	p. 211 ☐
				Do not use different soldering methods together (except for partial heating).	p. 211 ☐

For further information, please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706