Renesas Electronics America Inc - UPD78F9456GK-9ET-A Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	-
Core Size	-
Speed	-
Connectivity	-
Peripherals	-
Number of I/O	-
Program Memory Size	-
Program Memory Type	-
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	-
Data Converters	-
Oscillator Type	-
Operating Temperature	-
Mounting Type	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/upd78f9456gk-9et-a

Email: info@E-XFL.COM Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

LIST OF TABLES (2/2)

1.4.2 Pin configuration of µPD789446, 789456 Subseries (Top view)

★ • 64-pin plastic TQFP (fine pitch) (12 × 12)

µPD789445GK-xxx-9ET
µPD789446GK-xxx-9ET
µPD789455GK-xxx-9ET
µPD789456GK-xxx-9ET

µPD789445GK-xxx-9ET-A
µPD789446GK-xxx-9ET-A
µPD789455GK-xxx-9ET-A
µPD789456GK-xxx-9ET-A

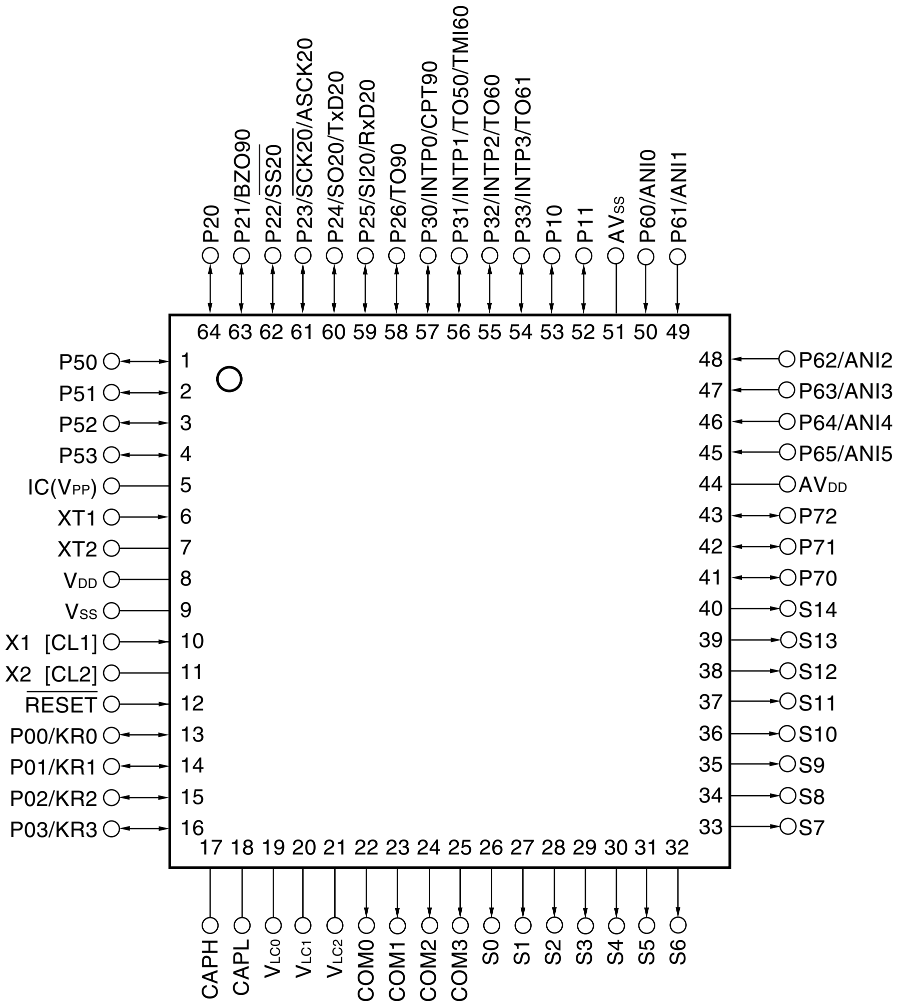
µPD78F9456GK-9ET
µPD78F9456GK-9ET-A

★ • 64-pin plastic LQFP (fine pitch) (10 × 10)

µPD789445GB-xxx-8EU
µPD789446GB-xxx-8EU
µPD789455GB-xxx-8EU
µPD789456GB-xxx-8EU

µPD789445GB-xxx-8EU-A
µPD789446GB-xxx-8EU-A
µPD789455GB-xxx-8EU-A
µPD789455GB-xxx-8EU-A

µPD78F9456GB-8EU
µPD78F9456GB-8EU-A



Cautions 1. Connect the IC (Internally Connected) pin directly to V_{SS}.

2. Connect the AV_{DD} pin to V_{DD}.

3. Connect the AV_{SS} pin to V_{SS}.

Remarks 1. The items in parentheses apply to the µPD78F9456.

2. The items in brackets apply when RC oscillation is selected (mask option).

Table 3-4. Special Function Register List (2/2)

Address	Special Function Register (SFR) Name	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 Bit	8 Bits	16 Bits	
FF27H	Port mode register 7	PM7	R/W	√	√	–	FFH
FF28H	Port mode register 8[Note]	PM8		√	√	–	
FF29H	Port mode register 9[Note]	PM9		√	√	–	
FF32H	Pull-up resistor option register B2	PUB2		√	√	–	00H
FF33H	Pull-up resistor option register B3	PUB3		√	√	–	
FF37H	Pull-up resistor option register B7	PUB7		√	√	–	
FF38H	Pull-up resistor option register B8[Note]	PUB8		√	√	–	
FF39H	Pull-up resistor option register B9[Note]	PUB9		√	√	–	
FF42H	Watchdog timer clock select register	WDCS		–	√	–	
FF48H	16-bit timer mode control register 90	TMC90		√	√	–	
FF49H	Buzzer output control register 90	BZC90		√	√	–	
FF4AH	Watch timer mode control register	WTM		√	√	–	
FF4CH	8-bit compare register H60	CRH60	W	–	√	–	Undefined
FF4DH	8-bit timer mode control register 50	TMC50	R/W	√	√	–	00H
FF4EH	8-bit timer mode control register 60	TMC60		√	√	–	
FF4FH	Carrier generator output control register 60	TCA60	W	–	√	–	
FF70H	Asynchronous serial interface mode register 20	ASIM20	R/W	√	√	–	
FF71H	Asynchronous serial interface status register 20	ASIS20	R	√	√	–	
FF72H	Serial operation mode register 20	CSIM20	R/W	√	√	–	
FF73H	Baud rate generator control register 20	BRGC20		–	√	–	
FF80H	A/D converter mode register 0	ADM0		√	√	–	
FF84H	Analog input channel specification register 0	ADS0		√	√	–	
FFB0H	LCD display mode register 0	LCDM0		√	√	–	
FFB2H	LCD clock control register 0	LCDC0		√	√	–	
FFB3H	LCD voltage amplification control register 0	LCDVA0		√	√	–	
FFE0H	Interrupt request flag register 0	IF0		√	√	–	
FFE1H	Interrupt request flag register 1	IF1		√	√	–	
FFE4H	Interrupt mask flag register 0	MK0		√	√	–	FFH
FFE5H	Interrupt mask flag register 1	MK1		√	√	–	
FFECH	External interrupt mode register 0	INTM0		–	√	–	00H
FFEDH	External interrupt mode register 1	INTM1		–	√	–	
FFF0H	Suboscillation mode register	SCKM		√	√	–	
FFF2H	Subclock control register	CSS		√	√	–	
FFF5H	Key return mode register 00	KRM00		√	√	–	
FFF7H	Pull-up resistor option register 0	PU0		√	√	–	
FFF9H	Watchdog timer mode register	WDTM		√	√	–	
FFFAH	Oscillation stabilization time select register	OSTS		–	√	–	04H
FFFBH	Processor clock control register	PCC		√	√	–	02H

Note μPD789426 and 789436 Subseries only.

3.4.3 Special function register (SFR) addressing

[Function]

The memory-mapped special function registers (SFRs) are addressed with 8-bit immediate data in an instruction word.

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

[Operand format]

Identifier	Description
sfr	Special function register name

[Description example]

MOV PM0, A; When selecting PM0 for sfr

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

[Illustration]

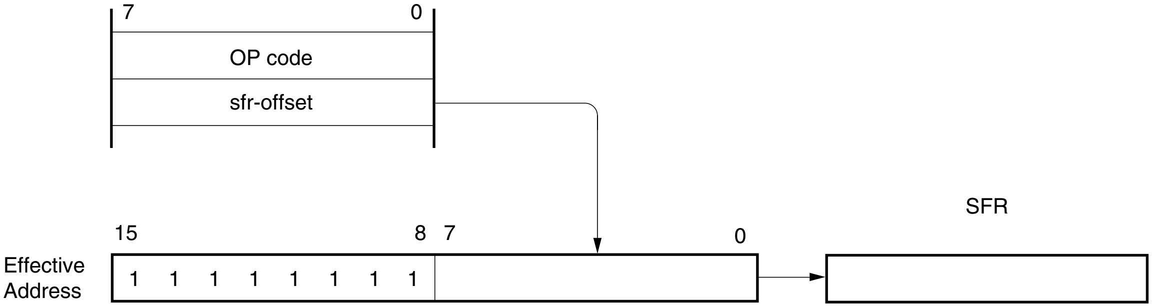
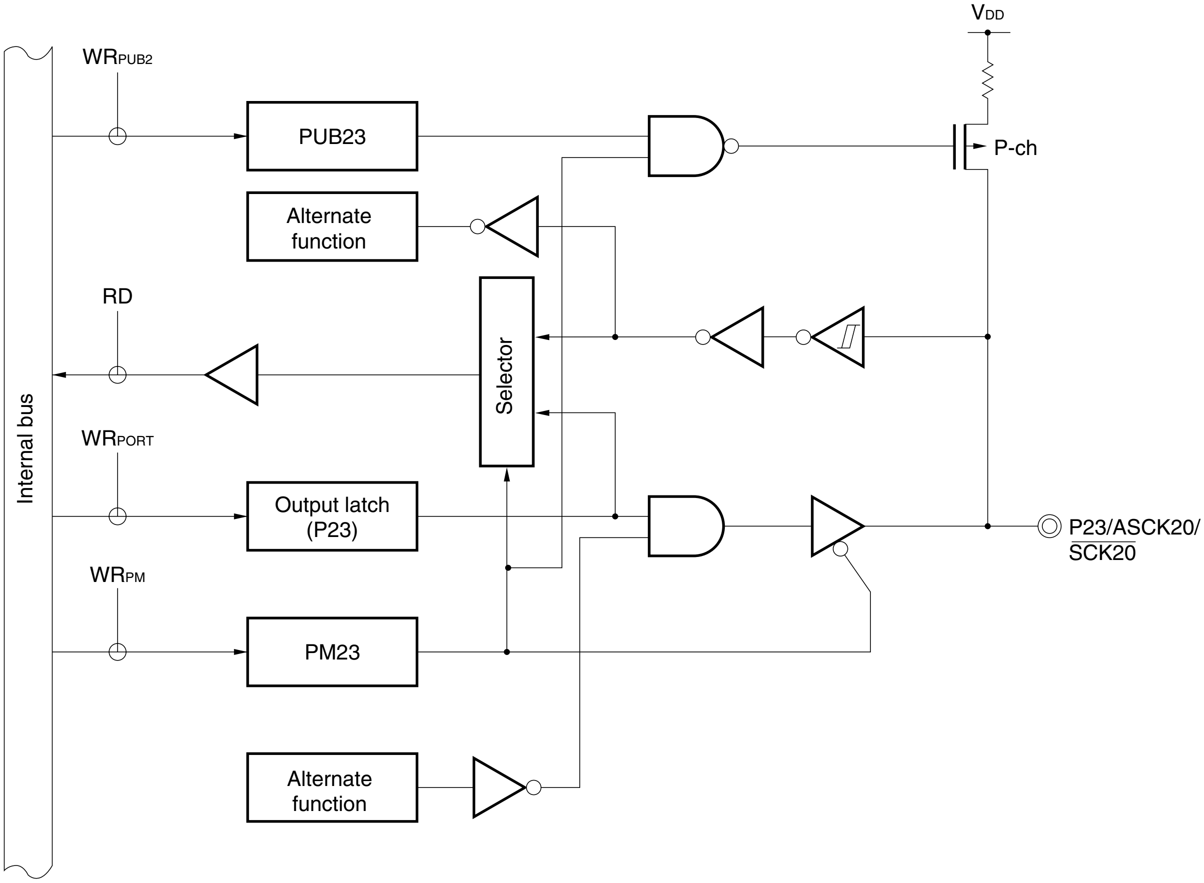


Figure 4-8. Block Diagram of P23



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

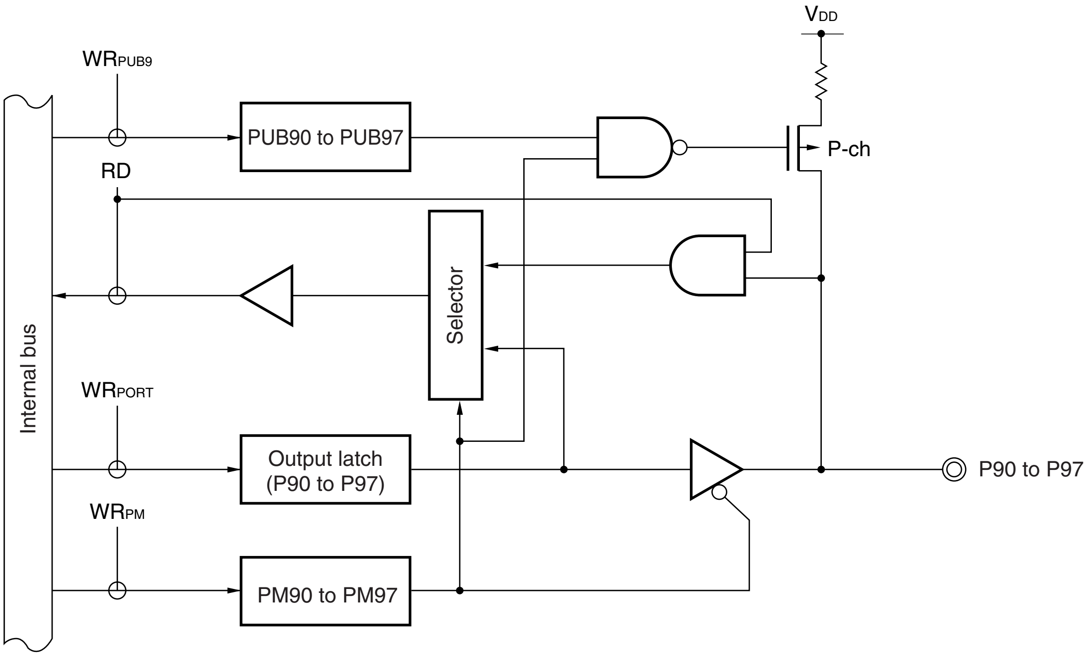
4.2.9 Port 9 (μPD789426, 789436 Subseries only)

This is an 8-bit I/O port with an output latch. Port 9 can be specified in the input or output mode in 1-bit units by using port mode register 9 (PM9). When using the pins of this port as input port pins, on-chip pull-up resistors can be connected in 1-bit units by setting pull-up resistor option register B9 (PUB9).

This port is set in the input mode when the $\overline{\text{RESET}}$ signal is input.

Figure 4-17 shows a block diagram of port 9.

Figure 4-17. Block Diagram of P90 to P97



PUB9: Pull-up resistor option register B9
PM: Port mode register
RD: Port 9 read signal
WR: Port 9 write signal

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

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

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

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

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

2. To overwrite CR90 during a count operation, it is necessary to disable interrupts in advance, using interrupt mask flag register 1 (MK1). It is also necessary to disable inversion of the timer output data, using 16-bit timer mode control register 90 (TMC90). If the value in CR90 is rewritten in the interrupt-enabled state, an interrupt request may occur at the moment of rewrite.

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

TM90 is used to count the number of pulses.

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

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

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

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

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

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

TCP90 captures the contents of TM90.

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

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

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

(4) 16-bit counter read buffer 90

This buffer is used to latch and hold the count value for TM90.

6.4 16-Bit Timer 90 Operation

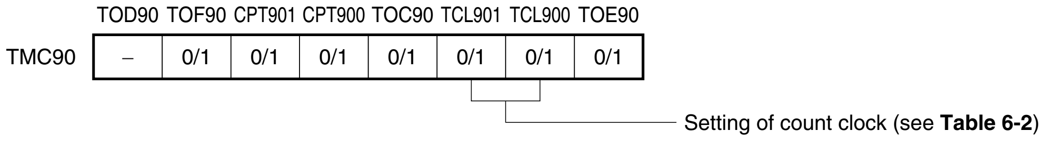
6.4.1 Operation as timer interrupt

★ 16-bit timer 90 can generate interrupts repeatedly each time the free-running counter value reaches the value set to CR90. Since this counter is not cleared and holds the count even after an interrupt is generated, the interval time is equal to one cycle of the count clock set in TCL901 and TCL900.

To operate 16-bit timer 90 as a timer interrupt, the following settings are required.

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

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



	TOD90	TOF90	CPT901	CPT900	TOC90	TCL901	TCL900	TOE90
TMC90	–	0/1	0/1	0/1	0/1	0/1	0/1	0/1

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

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

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

Caution When rewriting the value in CR90 during a count operation, be sure to execute the following processing.

<1> Set interrupt disabled (set TMMK90 (bit 1 of interrupt mask flag register 1 (MK1)) to 1).
<2> Disable inversion control of timer output data (set TOC90 to 0)

If the value in CR90 is rewritten in the interrupt-enabled state, an interrupt request may occur at the moment of rewrite.

Table 6-2. Interval Time of 16-Bit Timer 90

TCL901	TCL900	Count Clock	Interval Time
0	0	$2^{2}/f_X$ (0.8 μs)	$2^{18}/f_X$ (52.4 ms)
0	1	$2^{6}/f_X$ (12.8 μs)	$2^{22}/f_X$ (838.9 ms)
1	0	$2^{7}/f_X$ (25.6 μs)	$2^{23}/f_X$ (1.68 s)
1	1	$1/f_{XT}$ (30.5 μs)	$2^{16}/f_{XT}$ (2.0 s)

Remarks 1. f_X: Main system clock oscillation frequency

2. f_{XT}: Subsystem clock oscillation frequency

3. The parenthesized values apply to operation at f_X = 5.0 MHz or f_{XT} = 32.768 kHz.

(b) Timer 60: Pulse generator mode

The timer output status inverts repeatedly due to the settings of TM60, CR60, and CRH60, and pulses of any duty ratio are output (either P32/INTP2/TO60 or P33/INTP3/TO61 can be selected as the timer output pin using software).

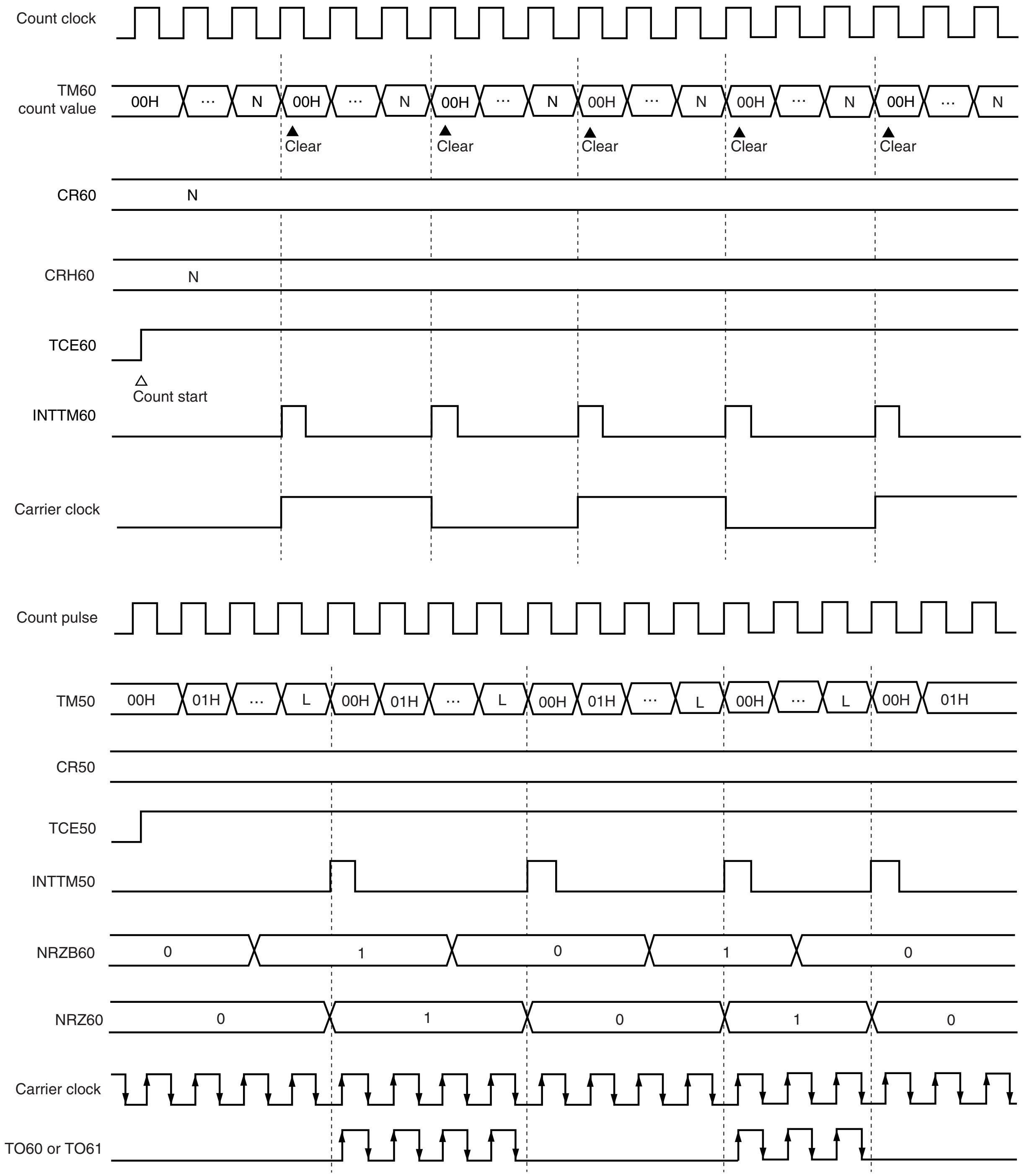
7.2 8-Bit Timers 50, 60 Configuration

8-bit timers 50 and 60 include the following hardware.

Table 7-2. 8-Bit Timer Configuration

Item	Configuration
Timer counters	8 bits × 2 (TM50, TM60)
Registers	Compare registers: 8 bits × 3 (CR50, CR60, CRH60)
Timer outputs	3 (TO50, TO60, TO61)
Control registers	8-bit timer mode control register 50 (TMC50) 8-bit timer mode control register 60 (TMC60) Carrier generator output control register 60 (TCA60) Port mode register 3 (PM3) Port 3 (P3)

Figure 7-21. Timing of Carrier Generator Operation (When CR60 = CRH60 = N)



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

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

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

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

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

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

Caution Bits 0 and 1 must be set to 0.

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

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

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

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

f_X: Main system clock oscillation frequency
n: Values determined by the settings of TPS200 to TPS203 as shown in the above table ($2 \leq n \leq 8$)

Table 12-5. Example of Relationships Between System Clock and Baud Rate

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

★ **Caution Do not select n = 1 during operation at fx > 2.5 MHz because the resulting baud rate exceeds the rated range.**

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

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

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

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

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

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

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

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

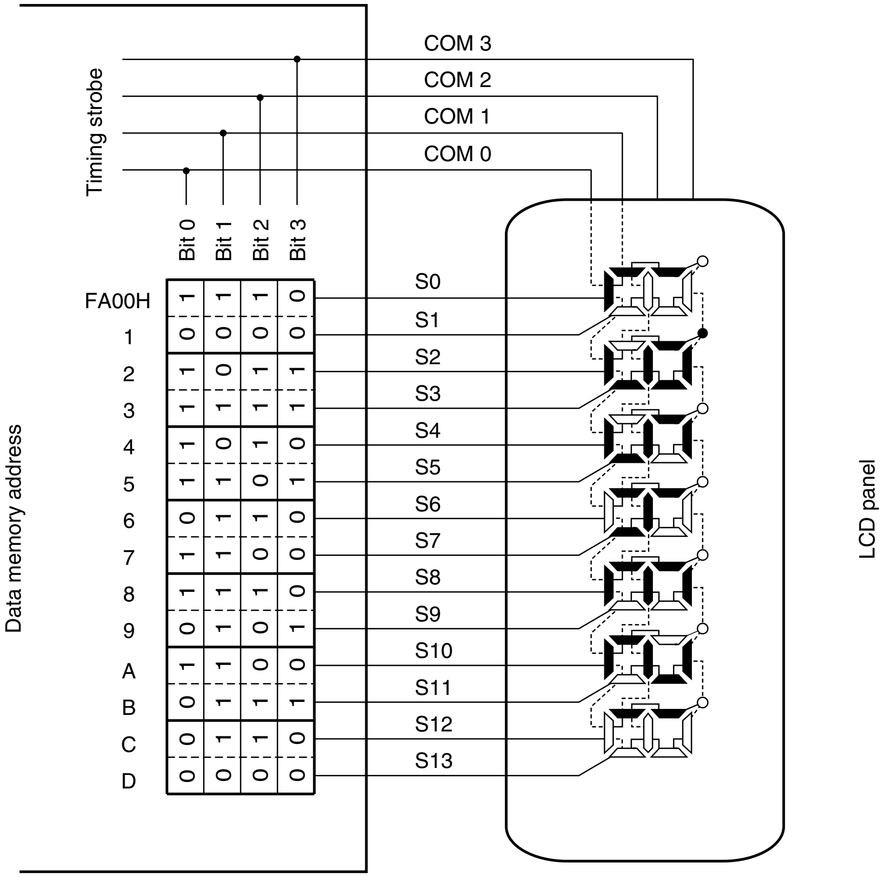
CL20	Transmit data character length specification
0	7 bits
1	8 bits

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

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

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

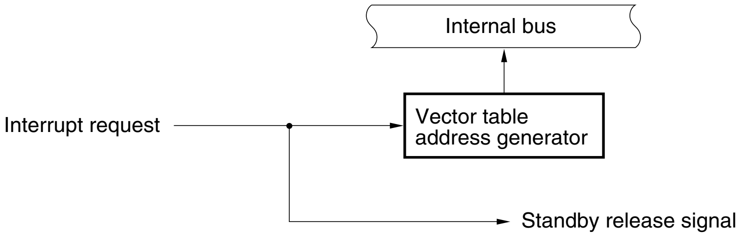
Figure 13-12. Example of Connecting Four-Time Slot LCD Panel



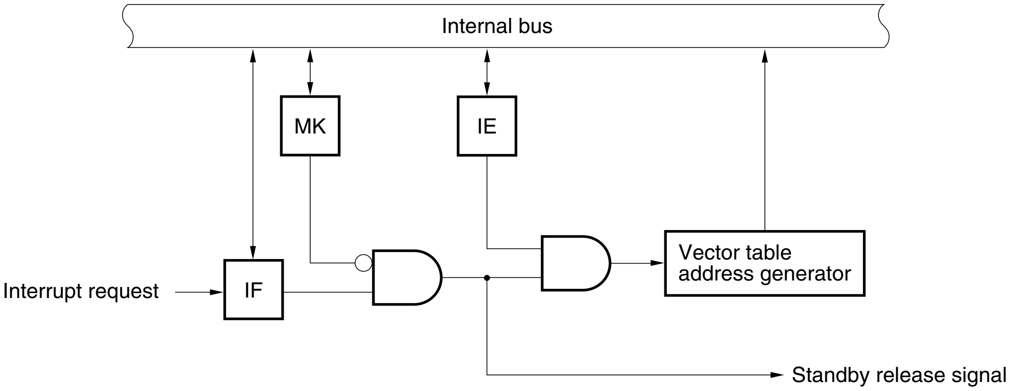
	Bit 0	Bit 1	Bit 2	Bit 3
FA00H	1	1	1	0
1	0	0	0	0
2	1	0	1	1
3	1	1	1	1
4	1	0	1	0
5	1	1	0	1
6	0	1	1	0
7	1	1	0	0
8	1	1	1	0
9	0	1	0	1
A	1	1	0	0
B	0	1	1	1
C	0	1	1	0
D	0	0	0	0

Figure 14-1. Basic Configuration of Interrupt Function

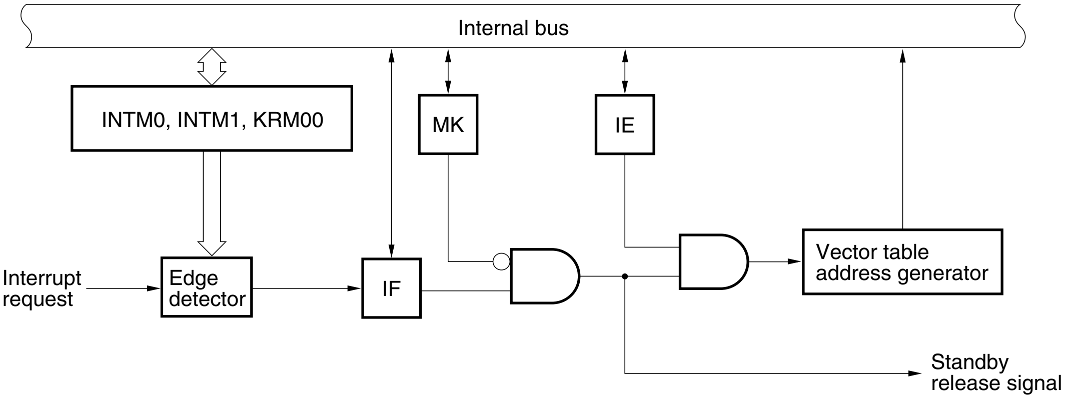
(A) Internal non-maskable interrupt



(B) Internal maskable interrupt

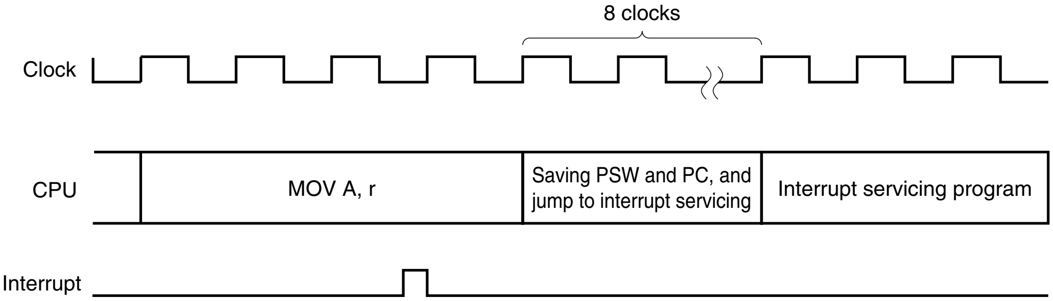


(C) External maskable interrupt



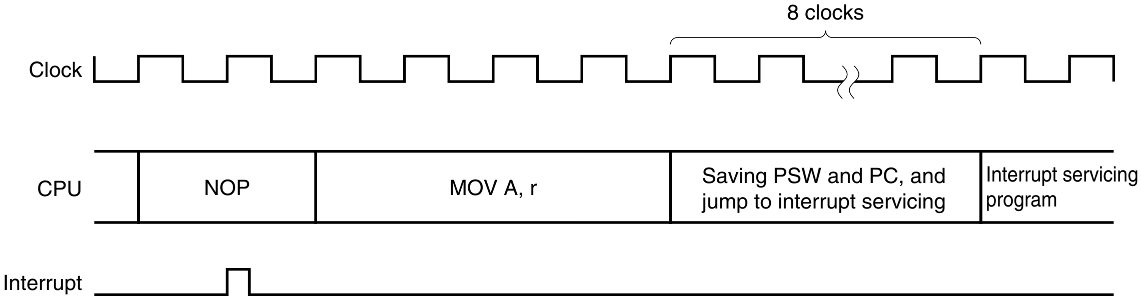
INTM0:	External interrupt mode register 0
INTM1:	External interrupt mode register 1
KRM00:	Key return mode register 00
IF:	Interrupt request flag
IE:	Interrupt enable flag
MK:	Interrupt mask flag

Figure 14-13. Interrupt Request Acknowledgment Timing (Example: MOV A, r)


8 clocks
Clock
CPU
MOV A, r
Saving PSW and PC, and jump to interrupt servicing
Interrupt servicing program
Interrupt

If the interrupt request has generated an interrupt request flag (XXIF) by the time the instruction clocks under execution, n clocks (n = 4 to 10), are n – 1, interrupt request acknowledgment processing will start following the completion of the instruction under execution. Figure 14-13 shows an example using the 8-bit data transfer instruction MOV A, r. Because this instruction is executed in 4 clocks, if an interrupt request is generated between the start of execution and the 3rd clock, interrupt request acknowledgment processing will take place following the completion of MOV A, r.

Figure 14-14. Interrupt Request Acknowledgment Timing
(When Interrupt Request Flag Is Generated in Final Clock Under Execution)


8 clocks
Clock
CPU
NOP
MOV A, r
Saving PSW and PC, and jump to interrupt servicing
Interrupt servicing program
Interrupt

If the interrupt request flag (XXIF) is generated in the final clock of the instruction, interrupt request acknowledgment processing will begin after execution of the next instruction is complete.

Figure 14-14 shows an example whereby an interrupt request was generated in the 2nd clock of NOP (a 2-clock instruction). In this case, the interrupt request will be processed after execution of MOV A, r, which follows NOP, is complete.

Caution When interrupt request flag registers 0 and 1 (IF0 and IF1), or interrupt mask flag registers 0 and 1 (MK0 and MK1) are being accessed, interrupt requests will be held pending.

14.4.3 Multiple interrupt servicing

Multiple interrupts, in which another interrupt request is acknowledged while an interrupt request being serviced, can be serviced using the priority order. If multiple interrupts are generated at the same time, they are serviced in the order according to the priority assigned to each interrupt request in advance (refer to **Table 14-1**).

CHAPTER 15 STANDBY FUNCTION

15.1 Standby Function and Configuration

15.1.1 Standby function

The standby function is to reduce the power consumption of the system and can be effected in the following two modes:

(1) HALT mode

This mode is set when the HALT instruction is executed. The HALT mode stops the operation clock of the CPU. The system clock oscillator continues oscillating. This mode does not reduce the power consumption as much as the STOP mode, but is useful for resuming processing immediately when an interrupt request is generated, or for intermittent operations.

(2) STOP mode

This mode is set when the STOP instruction is executed. The STOP mode stops the main system clock oscillator and stops the entire system. The power consumption of the CPU can be substantially reduced in this mode.

The data memory can be retained at the low voltage (V_{DD} = 1.8 V). Therefore, this mode is useful for retaining the contents of the data memory at an extremely low power consumption.

The STOP mode can be released by an interrupt request, so that this mode can be used for intermittent operation. However, some time is required until the system clock oscillator stabilizes after the STOP mode has been released. If processing must be resumed immediately by using an interrupt request, therefore, use the HALT mode.

In both modes, the previous contents of the registers, flags, and data memory before setting the standby mode are all retained. In addition, the statuses of the output latch of the I/O ports and output buffer are also retained.

Caution To set the STOP mode, be sure to stop the operations of the peripheral hardware, and then execute the STOP instruction.

Mnemonic	Operands	Byte	Clock	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	x	x	x
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte	x	x	x
	A, r	2	4	A, CY ← A + r	x	x	x
	A, saddr	2	4	A, CY ← A + (saddr)	x	x	x
	A, !addr16	3	8	A, CY ← A + (addr16)	x	x	x
	A, [HL]	1	6	A, CY ← A + (HL)	x	x	x
	A, [HL+byte]	2	6	A, CY ← A + (HL + byte)	x	x	x
ADDC	A, #byte	2	4	A, CY ← A + byte + CY	x	x	x
	saddr, #byte	3	6	(saddr), CY ← (saddr) + byte + CY	x	x	x
	A, r	2	4	A, CY ← A + r + CY	x	x	x
	A, saddr	2	4	A, CY ← A + (saddr) + CY	x	x	x
	A, !addr16	3	8	A, CY ← A + (addr16) + CY	x	x	x
	A, [HL]	1	6	A, CY ← A + (HL) + CY	x	x	x
	A, [HL+byte]	2	6	A, CY ← A + (HL + byte) + CY	x	x	x
SUB	A, #byte	2	4	A, CY ← A – byte	x	x	x
	saddr, #byte	3	6	(saddr), CY ← (saddr) – byte	x	x	x
	A, r	2	4	A, CY ← A – r	x	x	x
	A, saddr	2	4	A, CY ← A – (saddr)	x	x	x
	A, !addr16	3	8	A, CY ← A – (addr16)	x	x	x
	A, [HL]	1	6	A, CY ← A – (HL)	x	x	x
	A, [HL+byte]	2	6	A, CY ← A – (HL + byte)	x	x	x

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).

A.5 Debugging Tools (Hardware)

IE-78K0S-NS In-circuit emulator	In-circuit emulator for debugging hardware and software of application system using the 78K/0S Series. Can be used with an integrated debugger (ID78K0S-NS). Used in combination with an AC adapter, emulation probe, and interface adapter for connecting the host machine.
IE-78K0S-NS-A In-circuit emulator	In-circuit emulator with enhanced functions of the IE-78K0S-NS. The debug function is further enhanced by adding a coverage function and enhancing the tracer and timer functions.
IE-70000-MC-PS-B AC adapter	Adapter for supplying power from a 100 to 240 VAC outlet.
IE-70000-98-IF-C Interface adapter	Adapter required when using a PC-9800 series (except notebook type) as the host machine (C bus supported).
IE-70000-CD-IF-A PC card interface	PC card and interface cable required when using a notebook type PC as the host machine (PCMICA socket supported).
IE-70000-PC-IF-C Interface adapter	Adapter required when using an IBM PC/AT or compatible as the host machine (ISA bus supported).
IE-70000-PCI-IF-A Interface adapter	Adapter required when using a personal computer incorporating the PCI bus as the host machine.
IE-789456-NS-EM1 Emulation board	Emulation board for emulating the peripheral hardware inherent to the device. Used in combination with an in-circuit emulator.
NP-64GK NP-H64GK-TQ Emulation probe	Probe for connecting the in-circuit emulator and target system. Used in combination with TGK-064SBW.
TGK-064SBW Conversion adapter	Conversion adapter used to connect a target system board designed to allow mounting a 64-pin plastic TQFP (GK-9ET type) and the NP-64GK/NP-H64GK-TQ.
NP-64GB-TQ NP-H64GB-TQ Emulation probe	Probe for connecting the in-circuit emulator and target system. Used in combination with TGB-064SDP.
TGB-064SDP Conversion adapter	Conversion adapter used to connect a target system board designed to allow mounting a 64-pin plastic LQFP (GB-8EU type) and the NP-64GB-TQ/NP-H64GB-TQ.

Remarks 1. NP-64GK, NP-H64GK-TQ, NP-64GB-TQ, and NP-H64GB-TQ are products of Naito Densei Machida Mfg. Co., Ltd.

For further information, contact: Naito Densei Machida Mfg. Co., Ltd. (+81-45-475-4191)

2. TGK-064SBW and TGB-064SDP are products made by TOKYO ELETECH CORPORATION.

For further information, contact: Daimaru Kogyo, Ltd.
Tokyo Electronics Department (TEL +81-3-3820-7112)
Osaka Electronics Department (TEL +81-6-6244-6672)

C.2 Register Index (Alphabetic Order of Register Symbol)