To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

Notice

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

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

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

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

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

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

H8/300L SLP Series

Example of Receiving Remote Control Signals in Power-Saving Modes

Introduction

Signals received by an infrared remote control receiver are used as interrupt input to shift the mode of the microcomputer from standby to active, then it receives data from the infrared remote controller.

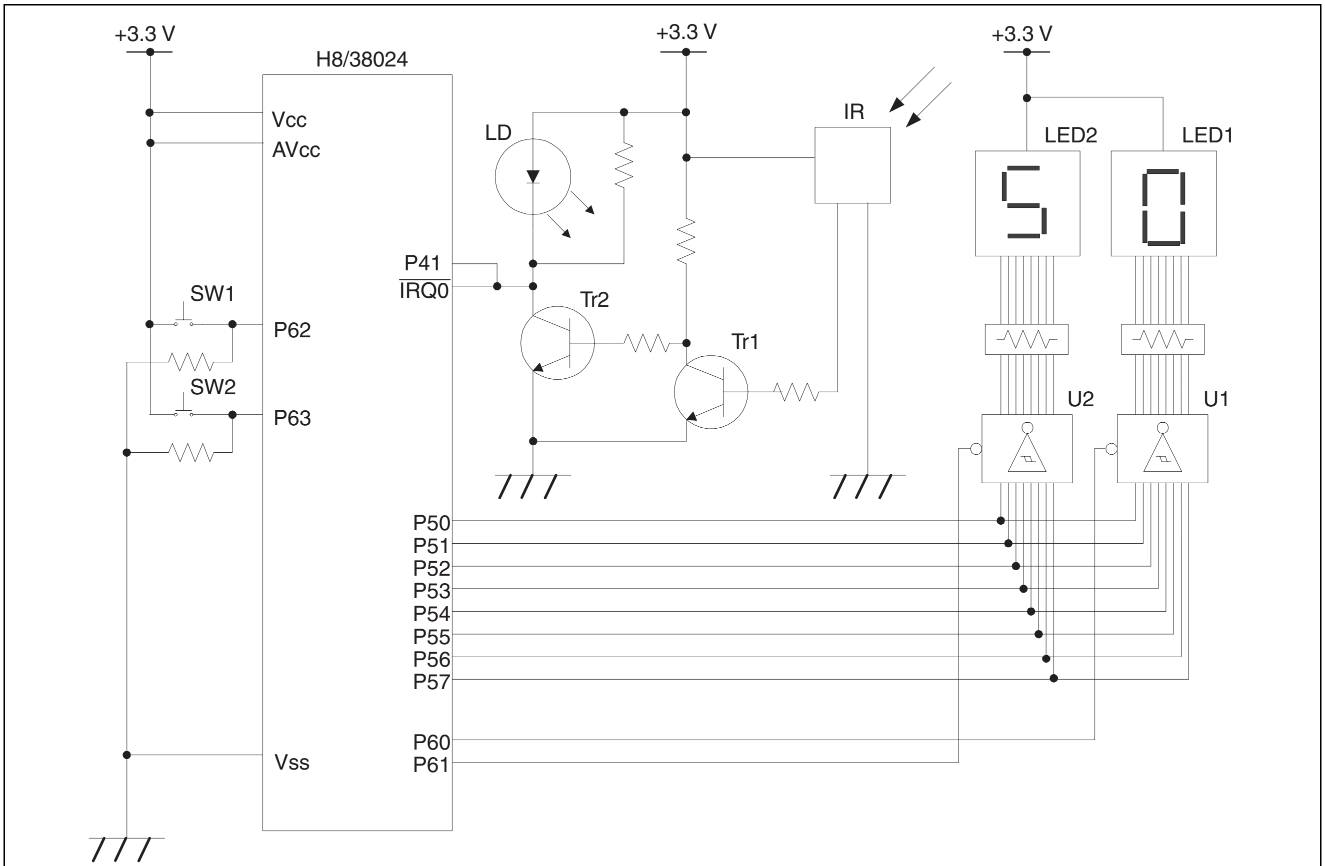
Target Device

H8/38024

Contents

1. Specifications

1. Figure 1 shows a hardware configuration required to receive infrared remote control data using an infrared remote control receiver.
2. In this sample task, infrared remote control signals are used to shift the mode of the microcomputer from standby, one of the power-saving modes of the microcomputer to the active mode, in which the microcomputer can receive signals.
3. Received data is displayed on an array of 7-segment LEDs as a 2-digit hexadecimal number (one byte). Pressing the pushbutton switch (SW1) shifts the display by one byte.
4. To return the microcomputer to the standby mode again, press the pushbutton switch (SW2).
5. In this sample task, the operating voltage (Vcc) and analog power supply voltage (AVcc) of the H8/38024 are both 3.3 V, the OSC clock frequency is 10 MHz when a ceramic resonator is used, and the watch clock frequency is 32.768 kHz.



Legend:
- IR: Infrared remote control receiver (GP1UD281XK manufactured by Sharp)
- LED1 and LED2: Common-anode 7-segment LEDs
- LD: Light-emitting diode
- U1 and U2: Tri-state output inverter drivers (HD74LS240)
- SW1 and SW2: Pushbutton switches
- Tr1 and Tr2: Transistors (2SC1815 manufactured by Toshiba)

Figure 1 Hardware Configuration

6. The infrared remote control receiver used in this task is a remote control receiver (model GP1UD281XK) manufactured by Sharp. The specifications are as follows:
   A. Figure 2 is an internal block diagram of the infrared remote control receiver.

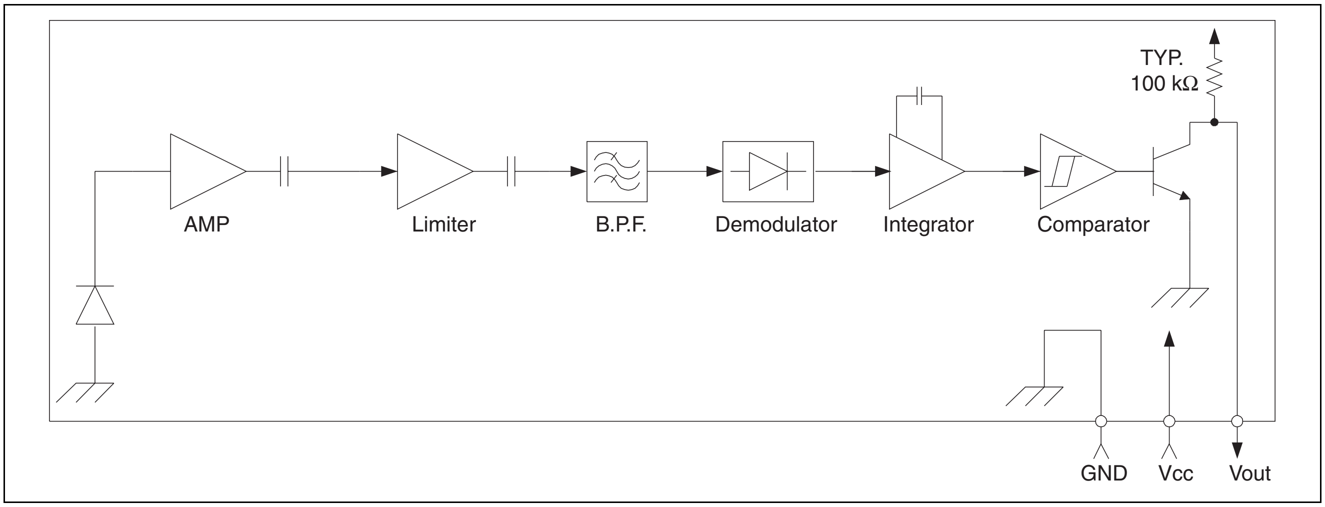


Figure 2 Internal Block Diagram

   B. Characteristics of the GP1UD281XK are listed below.
      a. Power supply voltage range: 2.7 to 5.5 V
      b. Carrier frequency: 38 kHz
      c. Internal demodulator compatible with pulse position modulation (PPM)
7. The circuit in this sample task operates as follows.
   A. The microcomputer first operates in the standby mode, then in the active mode, and finally in the standby mode.
   B. Infrared signals transmitted from the infrared remote controller are received and demodulated by the infrared remote control receiver. The first several signals are used to start the acceptance of an interrupt on the microcomputer.
   C. When the interrupt is accepted, the oscillation waveform of the microcomputer starts changing. When the time specified for the "oscillation stabilization time" of AC characteristics has elapsed, the system clock starts operating.
   D. After that, when the "waiting time" (8 to 16,384 states) has elapsed, the standby mode is released. Interrupt exception handling starts and the microcomputer shifts to the active mode.
   E. When the "oscillation stabilization waiting time" has elapsed after the interrupt is accepted, the microcomputer shifts from the standby mode to the active mode.
   F. The operation of the circuit in this sample task was checked with the waiting time set to 16 states. Figure 3 shows the definition of the waiting time.
      oscillation-stabilization-waiting-time = oscillation-stabilization-time + waiting-time
   G. The microcomputer shifting to the active mode receives subsequent signals.
   H. Received data is displayed using two 7-segment LEDs. The data is displayed as a 2-digit hexadecimal number (one byte). Each time the pushbutton switch (SW1) is pressed, fetched data is shifted by two digits (one byte) and the next data is displayed.
   I. The data of the infrared remote controller used in this sample task consists of 4 bytes (= 32 bits) and is displayed as follows:

      ['H50] → ['HAF] → ['H17] → ['HE8] → [– –] (=End)

      (Each manufacturer does not release remote control codes. The above data shows the result of handling data byte by byte (8 bits) with conforming to the LSB first format, which is a general remote control signal format.)

J. After received data has been displayed, pressing the pushbutton switch (SW2) as required places the microcomputer in the standby mode, that is, the status waiting for a remote control signal.
K. For your information, the received data is represented in binary as follows. The bit-inverted value of byte 1 is byte 2 and the bit-inverted value of byte 3 is byte 4.

['H50] (=0101 0000), ['HAF] (=0101 1111), ['H17] (=0001 0111), ['HE8] (=1110 1000)

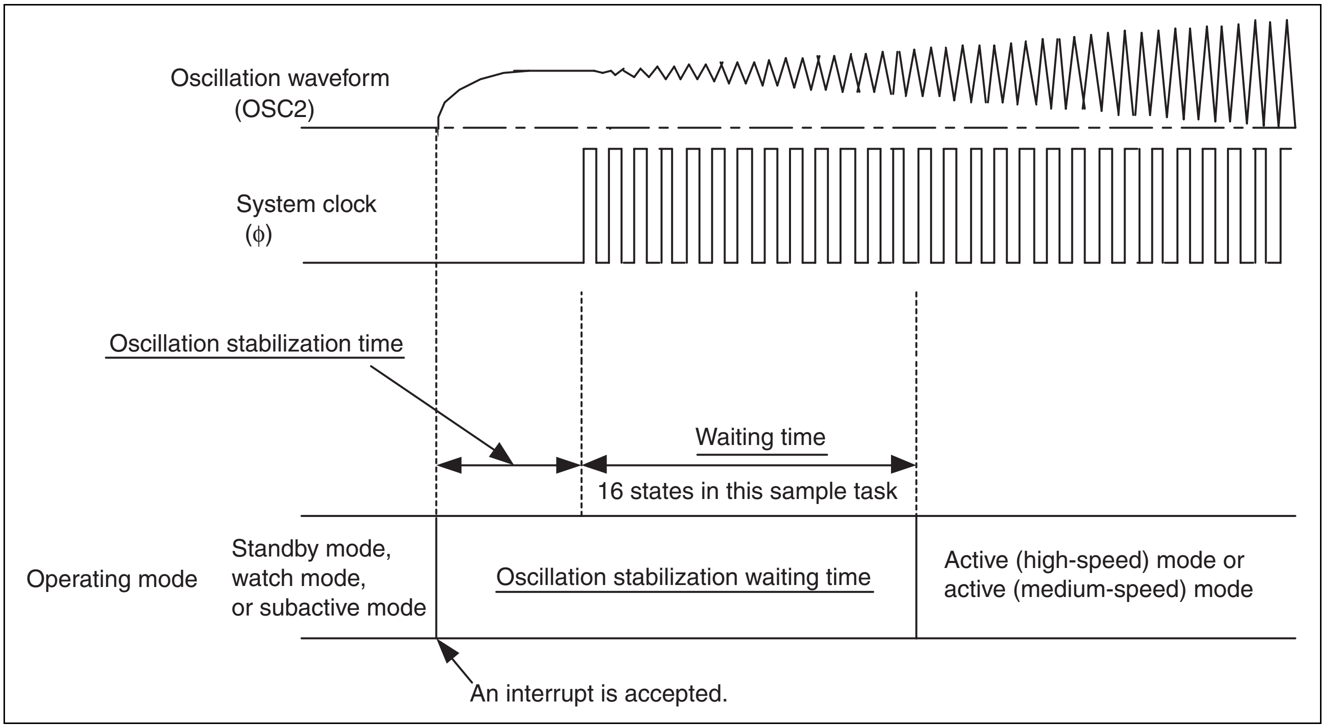


Figure 3 Oscillation Stabilization Waiting Time

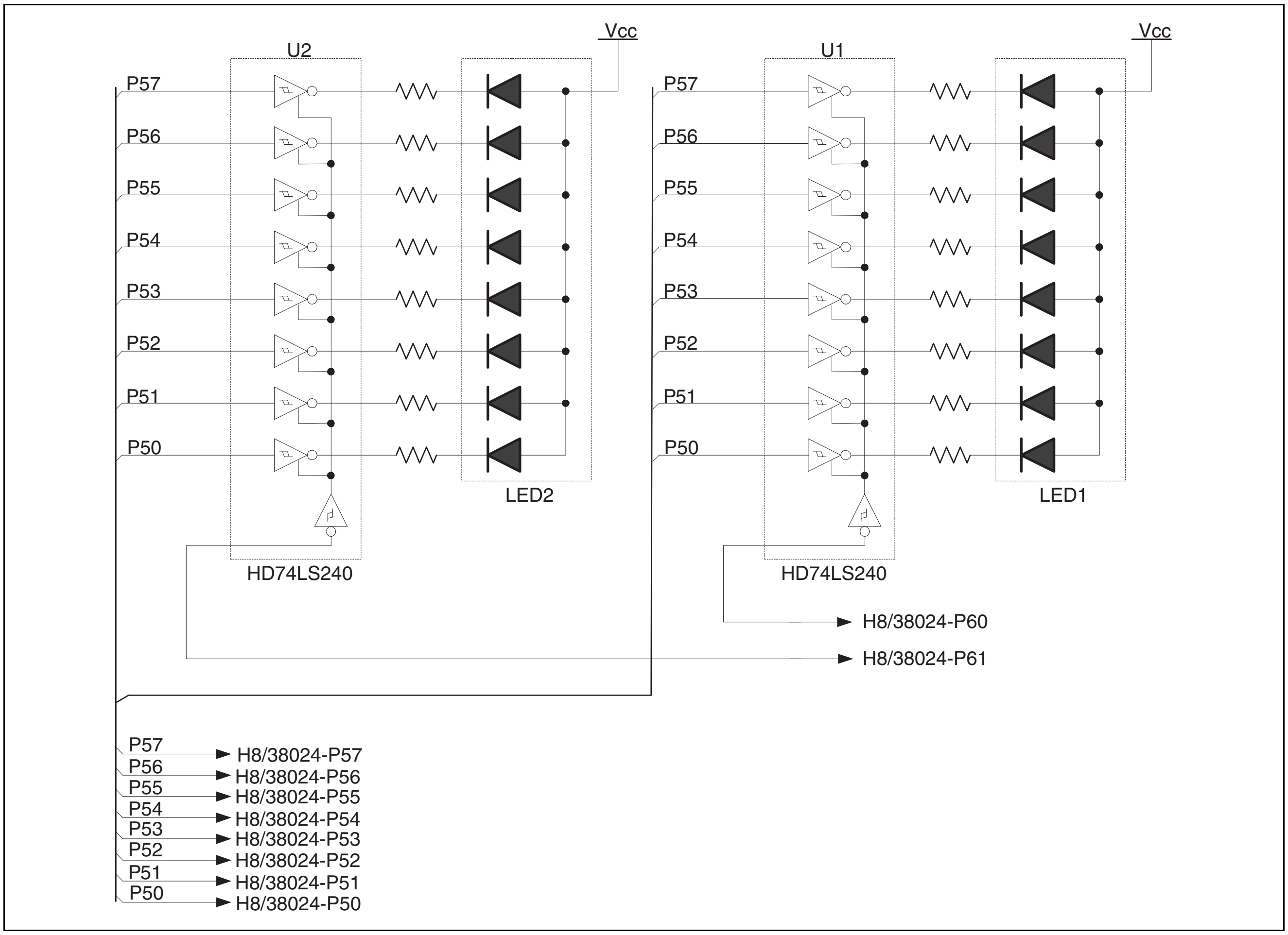


Figure 4 Method of Controlling 7-Segment LEDs

8. In this sample task, the remote control input result is displayed on the 7-segment LED array as a hexadecimal number (H'FF to H'00). Figure 5 shows how the remote control input result is displayed on the LEDs.

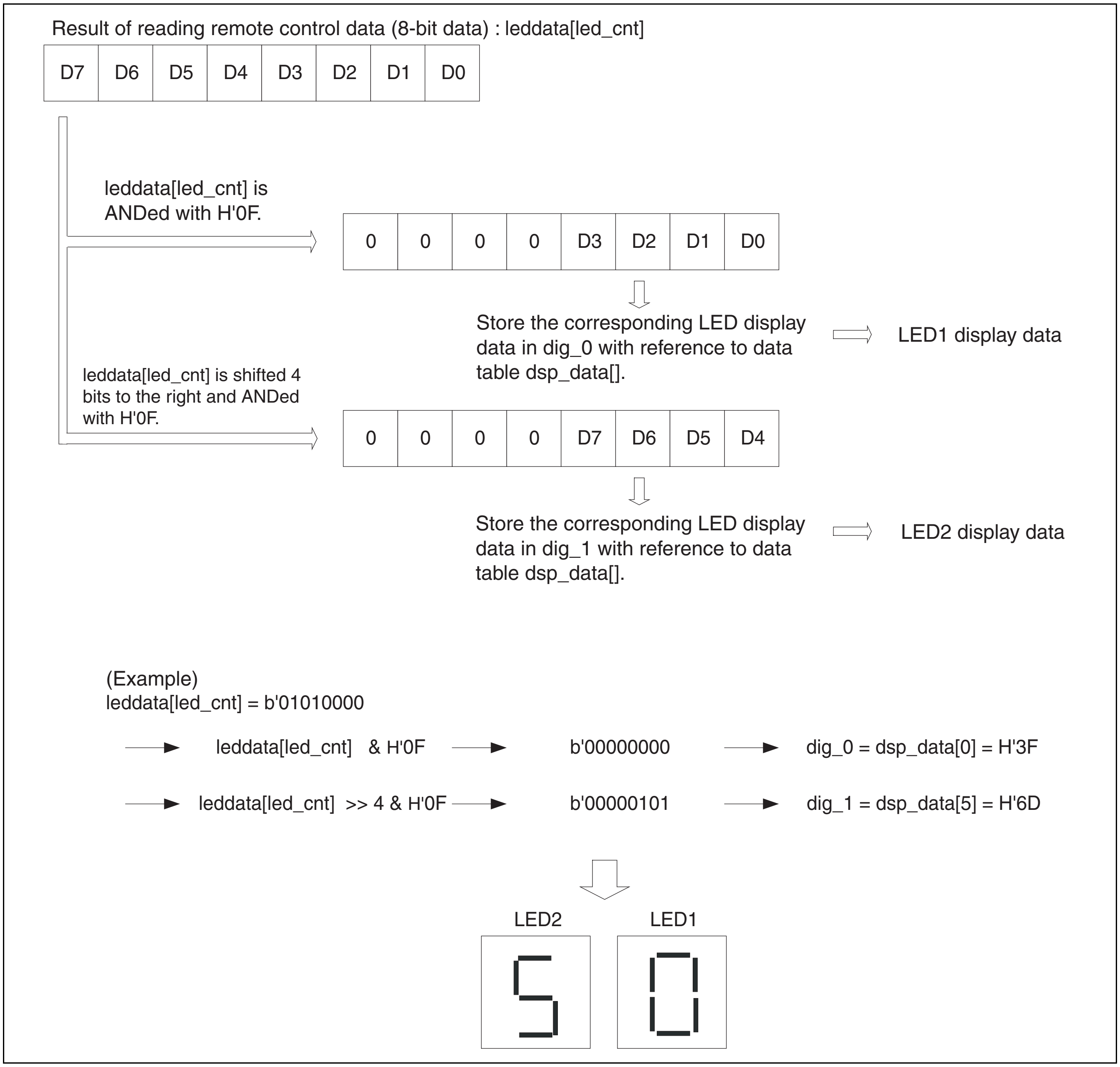


Figure 5 Method of Displaying the Remote Control Input Result on the LEDs

2. Description of Functions

1. Figure 6 is a block diagram of the H8/38024 functions used in this sample task. Table 1 is a list of the function assignments.

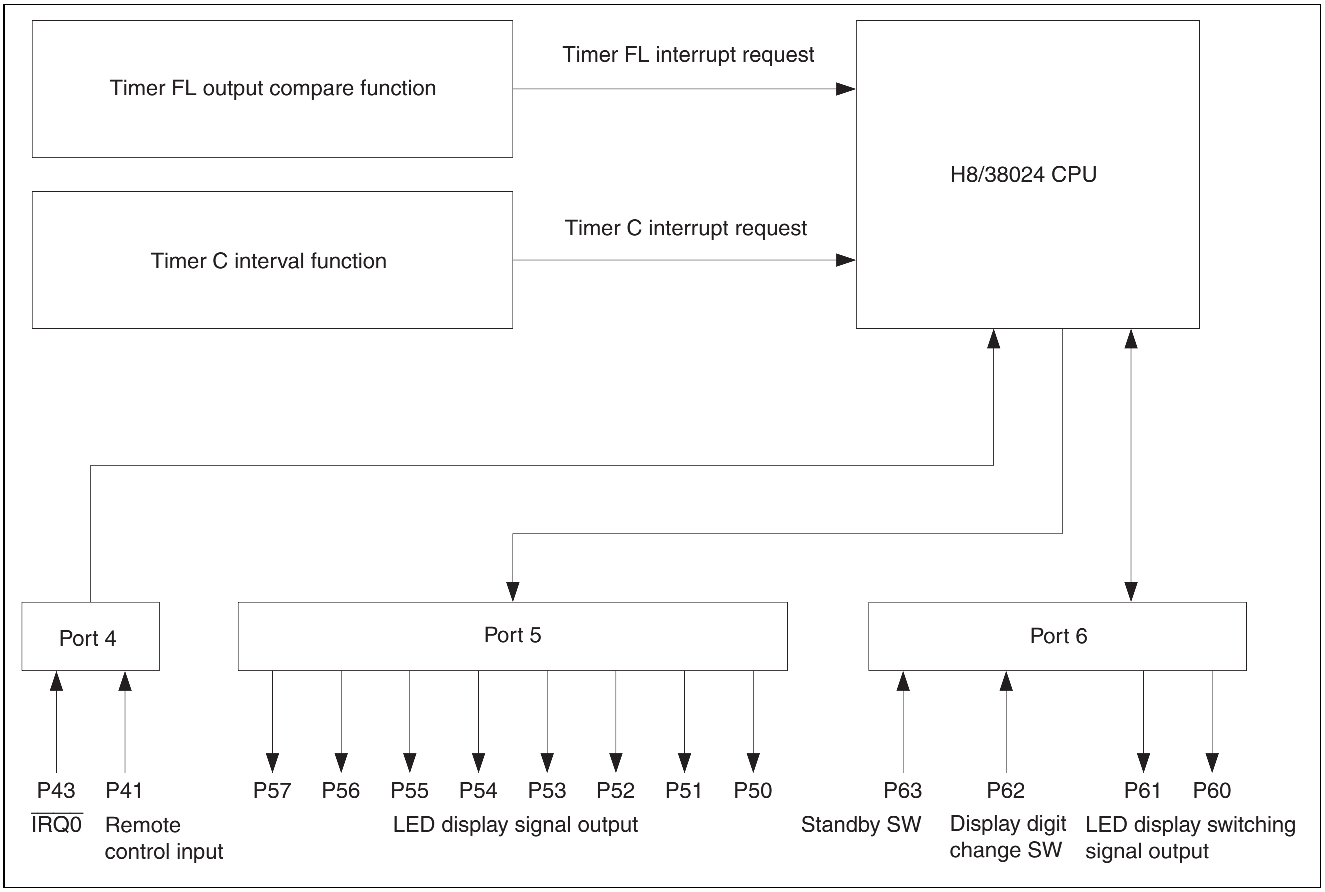


Figure 6 Block Diagram of Functions Used

Table 1 Assignment of Functions

Function	Description
Timer FL	The output compare function is used to input the infrared remote control data from the P41 input pin at intervals of 0.1 ms.
Timer C	The interval function is used to control the switching between the 7-segment LEDs. Each of the two 7-segment LEDs is lit in sequence at intervals of 3.2768 ms (the time at which timer C overflows), enabling dynamic illumination of the LED.
Port 4	Infrared remote control data is received at the P41 input pin and P43 ($\overline{IRQ0}$ interrupt) is used to shift the mode from standby to active (high-speed).
Port 5	The P50 to P57 output pins are used to display data on the currently active 7-segment LED. The remote control data from the P41 pin is converted to 2-digit hexadecimal display data and output to the LEDs.
Port 6	P61 and P60 are turned on and off alternately to light the two 7-segment LEDs respectively. Pressing the display digit change SW (P62) displays input remote control codes consisting of multiple bytes in sequence. Pressing the standby SW (P63) shifts the mode from active (high-speed) to standby.

2. Figure 7 is a connection diagram of a 7-segment LED used. Outputting a high level from port 5 lights the corresponding segments of the LED, as shown in the figure. Table 2 is a list of the relationships between port 5 outputs and data on the LED display.

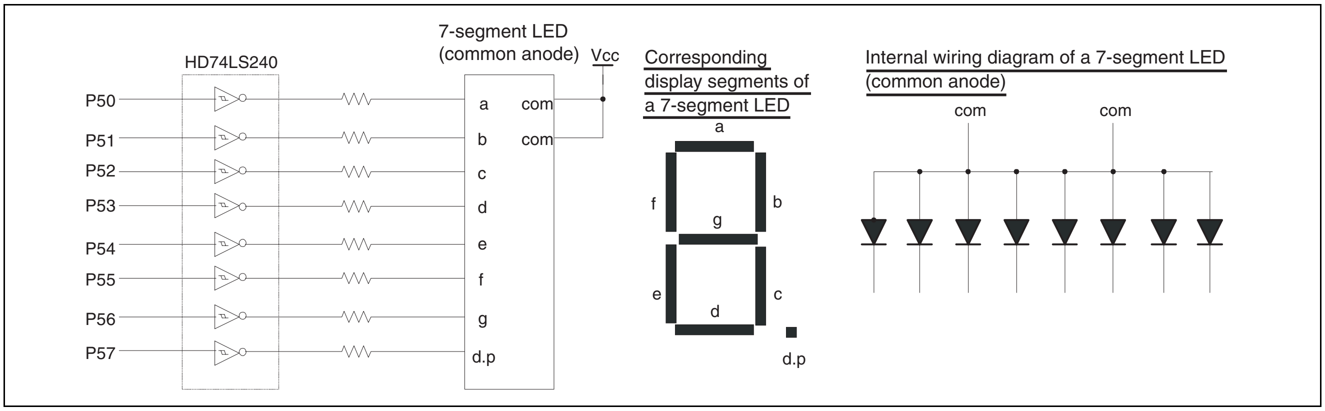
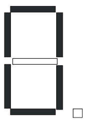
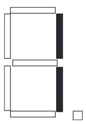
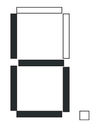
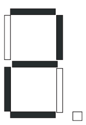
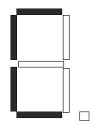
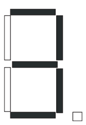
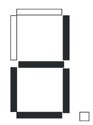
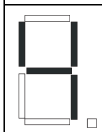
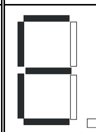
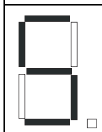
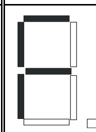
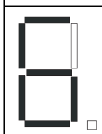
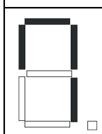
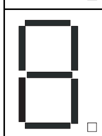
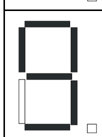


Figure 7 Connection Diagram and Internal Wiring of a 7-Segment LED

Table 2 Relationship between Port 5 Outputs and 7-Segment LED Display Data

LED display	Port 5 output data								LED display	Port 5 output data							
	P57	P56	P55	P54	P53	P52	P51	P50		P57	P56	P55	P54	P53	P52	P51	P50
	1	1	1	1	1	1	0	0		1	1	1	0	1	1	1	0
	0	1	1	0	0	0	0	0		0	0	1	1	1	1	1	0
	1	1	0	1	1	0	1	0		1	0	0	1	1	1	0	0
	1	1	1	1	0	0	1	0		0	1	1	1	1	0	1	0
	0	1	1	0	0	1	1	0		1	0	0	1	1	1	1	0
	1	0	1	1	0	1	1	0		1	0	0	0	1	1	1	0
	1	0	1	1	1	1	1	0									
	1	1	1	0	0	1	0	0									
	1	1	1	1	1	1	1	0									
	1	1	1	1	0	1	1	0									

3. Principles of Operation

1. Figure 8 shows how remote control signals are received using timer FL. The operating mode is set to standby, and infrared remote control signals are input from P41. An $\overline{\text{IRQ0}}$ interrupt generated upon receiving the signals shifts the MCU mode to active (high-speed) and timer FL interrupts generated at intervals of 0.1 ms are used to input the infrared signal state.

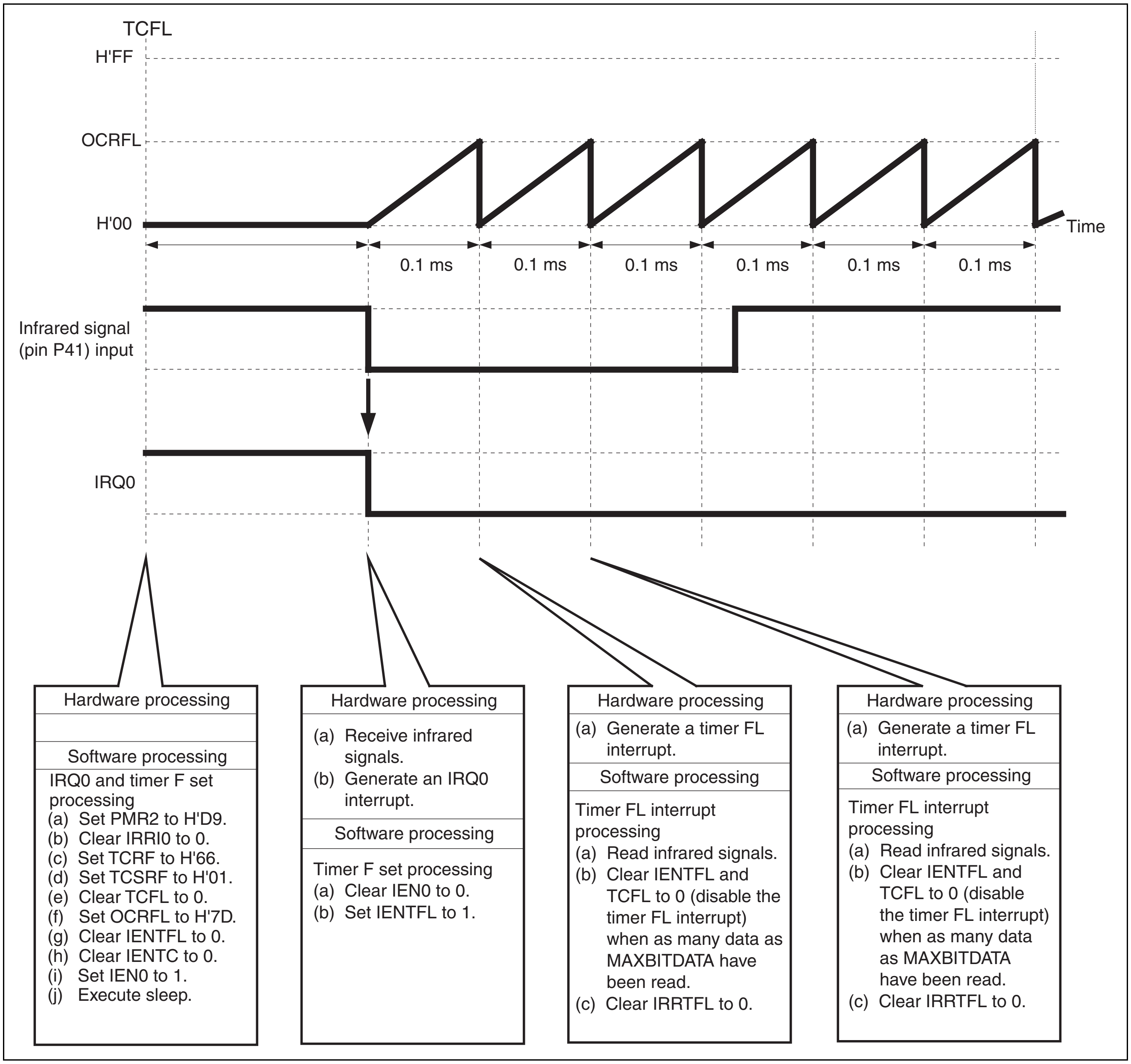


Figure 8 Description of Receiving Remote Control Signals Using Timer FL

2. The following describes the operation of 7-segment LED display control. Figure 9 shows how a value of "50" is displayed on LED2 and LED1. As shown in the figure, each of LED1 and LED2 is lit in sequence at timer C overflow intervals, enabling dynamic display on the 7-segment LED.

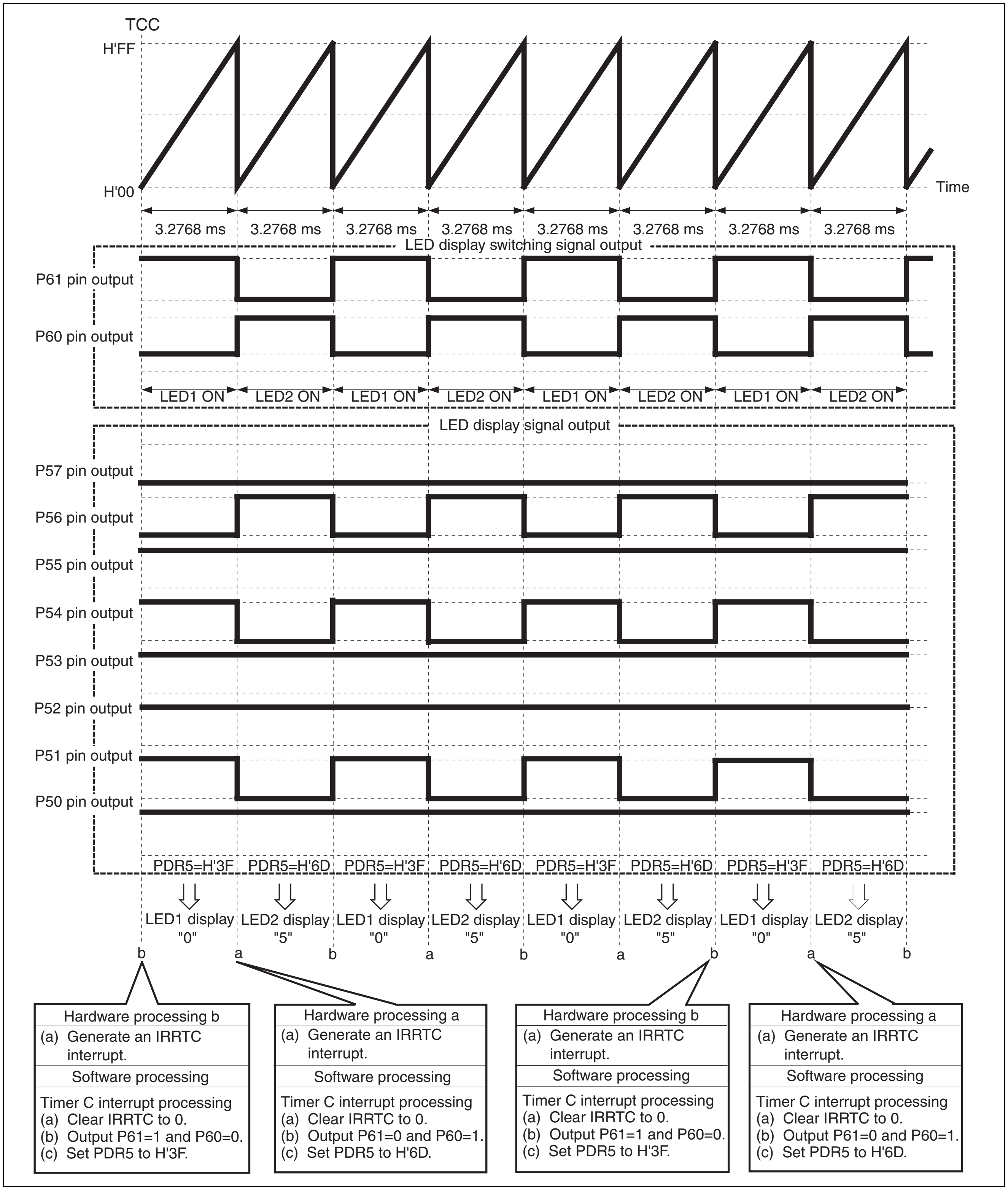


Figure 9 Description of 7-Segment LED Display Control

4. Description of Software

1. Description of modules
   Table 3 is a list of the modules used in this sample task.

Table 3 Description of Modules

Module Name	Label Name	Function
Main routine	main	After making the initial settings, shifts the mode to standby, waits for the completion of data fetching, and performs code decision processing and LED display processing repeatedly.
Code decision processing	code_decision	Extracts the code section from data input from the remote controller.
LED display processing	led_disp	Displays input remote control codes consisting of multiple bytes in sequence when the display switching SW is on. Shifts the mode from active (high-speed) to standby when the standby SW is on.
Software delay processing	delay	Used as a software timer which measures about 300 ms.
$\overline{IRQ0}$ interrupt processing	irq0	Disables an $\overline{IRQ0}$ interrupt.
Timer C interrupt processing	tmrc	Clears the interrupt flag, outputs the LED display data, and controls LED display switching.
Timer F interrupt processing	tmrf	Clears the interrupt flag and inputs the infrared signal state.

2. Description of arguments
   No arguments are used in this sample task.
3. Description of internal registers
   Table 4 is a list of the internal registers used in this sample task.

Table 4 Description of Internal Registers

Register Name		Description	Address	Setting
TCRF		Timer control register F: Switches between the 16- and 8-bit modes, selects among four types of internal clocks and an external event, and sets the output levels of the TMOFH and TMOFL pins.	H′FFB6	H′66
	TOLH	Toggle output level H: Sets the output level of the TMOFH pin. When TOLH = 0, the level is low.	Bit 7	0
	CKSH2	Clock select H: When CKSH2 = 1, CKSH1 = 1, and CKSH0 = 0, counting is performed using an internal clock of ϕ/4.	Bit 6	1
	CKSH1		Bit 5	1
	CKSH0		Bit 4	0
	TOLL	Toggle output level L: Sets the output level of the TMOFL pin. When TOLL = 0, the level is low.	Bit 3	0
	CKSL2	Clock select L: When CKSL2 = 1, CKSL1 = 1, and CKSL0 = 0, counting is performed using an internal clock of ϕ/4.	Bit 2	1
	CKSL1		Bit 1	1
	CKSL0		Bit 0	0
TMC		Timer mode register C: Selects the automatic reload function, controls count-up or count-down of the counter, and selects an input clock.	H′FFB4	H′1B
	TMC7	Automatic reload function selection: Selects the automatic reload function of timer C. When TMC7 = 0, the interval function is selected.	Bit 7	0
	TMC6	Counter-up/down control: Selects the up-counter or down-counter function. When TMC6 = 0 and TMC5 = 1, the TCC functions as an up-counter.	Bit 6	0
	TMC5		Bit 5	0
	TMC2	Clock select: Selects a clock to be input to the TCC. When TMC2 = 0, TMC1 = 1, and TMC0 = 1, counting is performed using an internal clock of ϕ/64.	Bit 2	0
	TMC1		Bit 1	1
	TMC0		Bit 0	1
TLC		Timer load register C: Sets the TCC reload value.	H′FFB5	H′00

Table 4 Description of Internal Registers

Register Name		Description	Address	Setting
TCSRF		Timer control/status register F: Specifies whether to enable or disable counter clearing, sets the overflow flag and compare match flag, and specifies whether to enable or disable an interrupt request due to an overflow.	H′FFB7	H′01
	OVFH	Timer overflow flag H: Status flag indicating that TCFH overflow has occurred (H'FF → H'00)	Bit 7	0
	CMFH	Compare match flag H: Status flag indicating that the values of TCFH and OCRFH match	Bit 6	0
	OVIEH	Timer overflow interrupt enable H: When OVIEH = 0, an interrupt request due to a TCFH overflow is disabled.	Bit 5	0
	CCLRH	Counter clear H: When CCLRH = 1, TCF clearing due to a compare match is enabled.	Bit 4	0
	OVFL	Timer overflow flag L: Status flag indicating that TCFL overflow has occurred (H'FF → H'00)	Bit 3	0
	CMFL	Compare match flag L: Status flag indicating that the values of TCFL and OCRFL match	Bit 2	0
	OVIEL	Timer overflow interrupt enable L: When OVIEL = 0, an interrupt request due to a TCFL overflow is disabled.	Bit 1	0
	CCLRL	Counter clear L: When CCLRL = 1, TCFL clearing due to a compare match is enabled.	Bit 0	1
TCFL		8-bit timer counter: 8-bit read/write up-counter	H′FFB9	H′00
OCRFL		8-bit output compare register: Generates an interrupt due to a TCFL compare match.	H′FFBB	H′7D

Table 4 Description of Internal Registers

Register Name		Description	Address	Setting
IENR1		Interrupt enable register 1	H'FFF3	H'01 (At initial setting)
	IEN0	When IEN0 = 1, an $\overline{IRQ0}$ pin interrupt request is enabled.	Bit 0	1/0
IENR2		Interrupt enable register 2	H'FFF4	H'00 (At initial setting)
	IENTFL	When IENTFL = 1, a timer FL interrupt request is enabled.	Bit 2	1/0
	IENTC	When IENTC = 1, a timer C interrupt request is enabled.	Bit 1	1/0
IRR1		Interrupt request register 1	H'FFF6	H'00 (At initial setting)
	IRRI0	When IRRI0 = 1, this bit can be cleared by writing 0 in it. This bit is set to 1 when the $\overline{IRQ0}$ pin is set to the interrupt input and the designated signal edge is input.	Bit 0	1/0
IRR2		Interrupt request register 2	H'FFF7	H'00 (At initial setting)
	IRRTFL	When IRRTFL = 1, this bit can be cleared by writing 0 in it. This bit is set to 1 when the values of TCFL and OCRFL match in the 8-bit timer mode.	Bit 2	1/0
	IRRTC	When IRRTC = 1, this bit can be cleared by writing 0 in it. This bit is set to 1 when the counter value of timer C overflows (H'FF → H'00) or underflows (H'00 → H'FF).	Bit 1	1/0
SYSCR1		System control register 1: Controls the power down mode.	H'FFF0	H'F7
	SSBY	Software standby: 1: Executes the SLEEP instruction in the active mode and shifts the mode to standby or watch.	Bit 7	1
	STS2	Standby timer select 2 to 0: 7: Sets the waiting time to 16 states.	Bit 6	1
	STS1		Bit 5	1
	STS0		Bit 4	1
	LSON	Low-speed on flag: 0: Uses the system clock (φ) as the CPU operating clock.	Bit 3	0
	MA1	Active (medium-speed) mode clock select: 3: φosc/128	Bit 1	1
	MA0		Bit 0	1

Table 4 Description of Internal Registers

Register Name	Description	Address	Setting
PDR4	Port data register 4: General I/O port data register for port 4	H′FFD7	H′00
PCR4	Port control register 4: Specifies, bit by bit, whether to use each of the port 4 pins to be used as general I/O ports as an input pin or output pin. When PCR4 = H'00, the P47 to P40 pins function as general input pins.	H′FFE7	H′00
PDR5	Port data register 5: General I/O port data register for port 5	H′FFD8	H′00
PCR5	Port control register 5: Specifies, bit by bit, whether to use each of the port 5 pins to be used as general I/O ports as an input pin or output pin. When PCR5 = H'FF, the P57 to P50 pins function as general output pins.	H′FFE8	H′FF
PDR6	Port data register 6: General I/O port data register for port 6	H′FFD9	H′03
PCR6	Port control register 6: Specifies, bit by bit, whether to use each of the port 6 pins to be used as general I/O ports as an input pin or output pin. When PCR6 = H'03, the P61 and P60 pins function as general output pins and the P67 to P62 pins function as general input pins.	H′FFE9	H′03

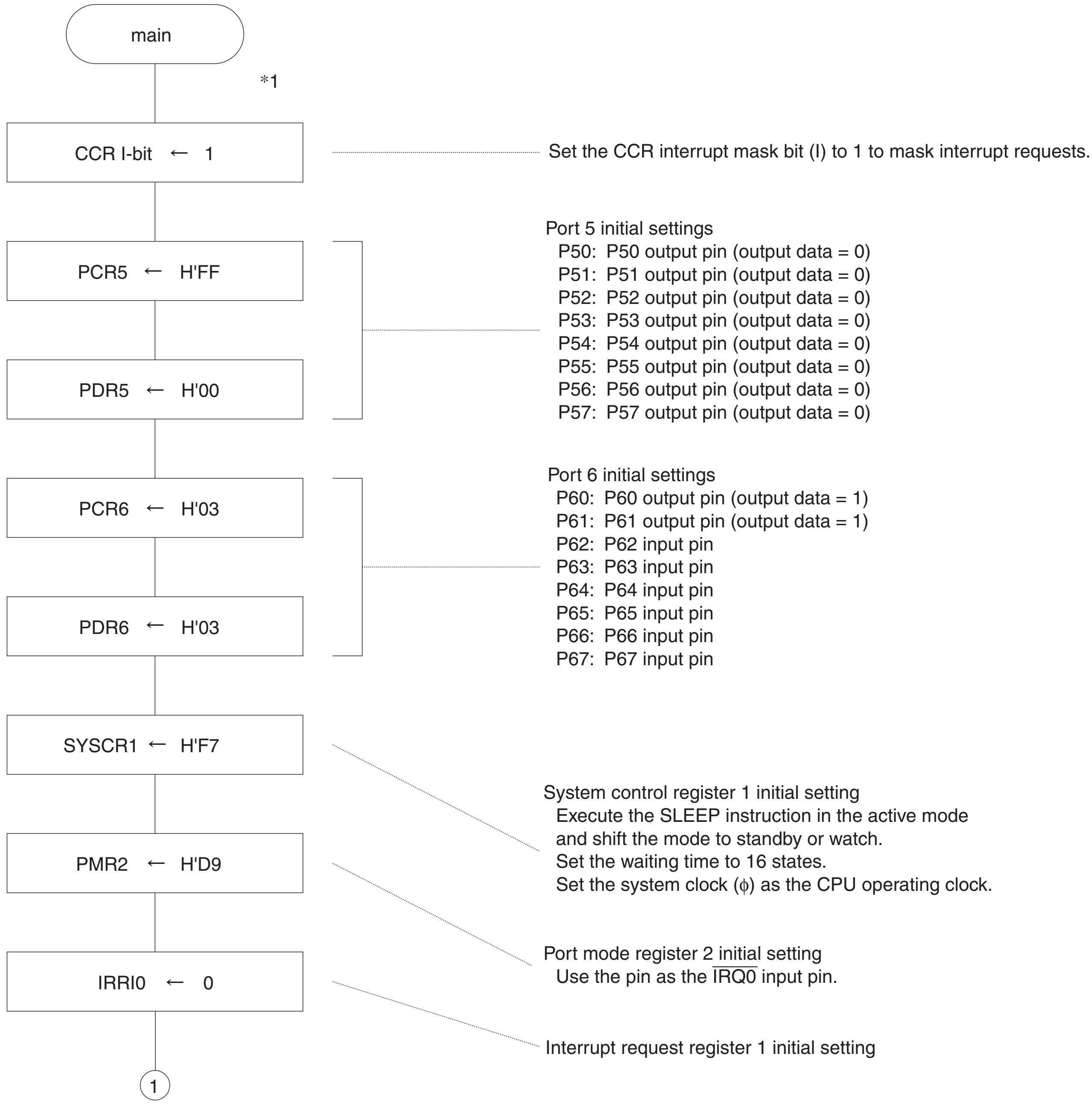
4. RAM usage
   Table 5 describes RAM usage in this sample task.

Table 5 Description of RAM

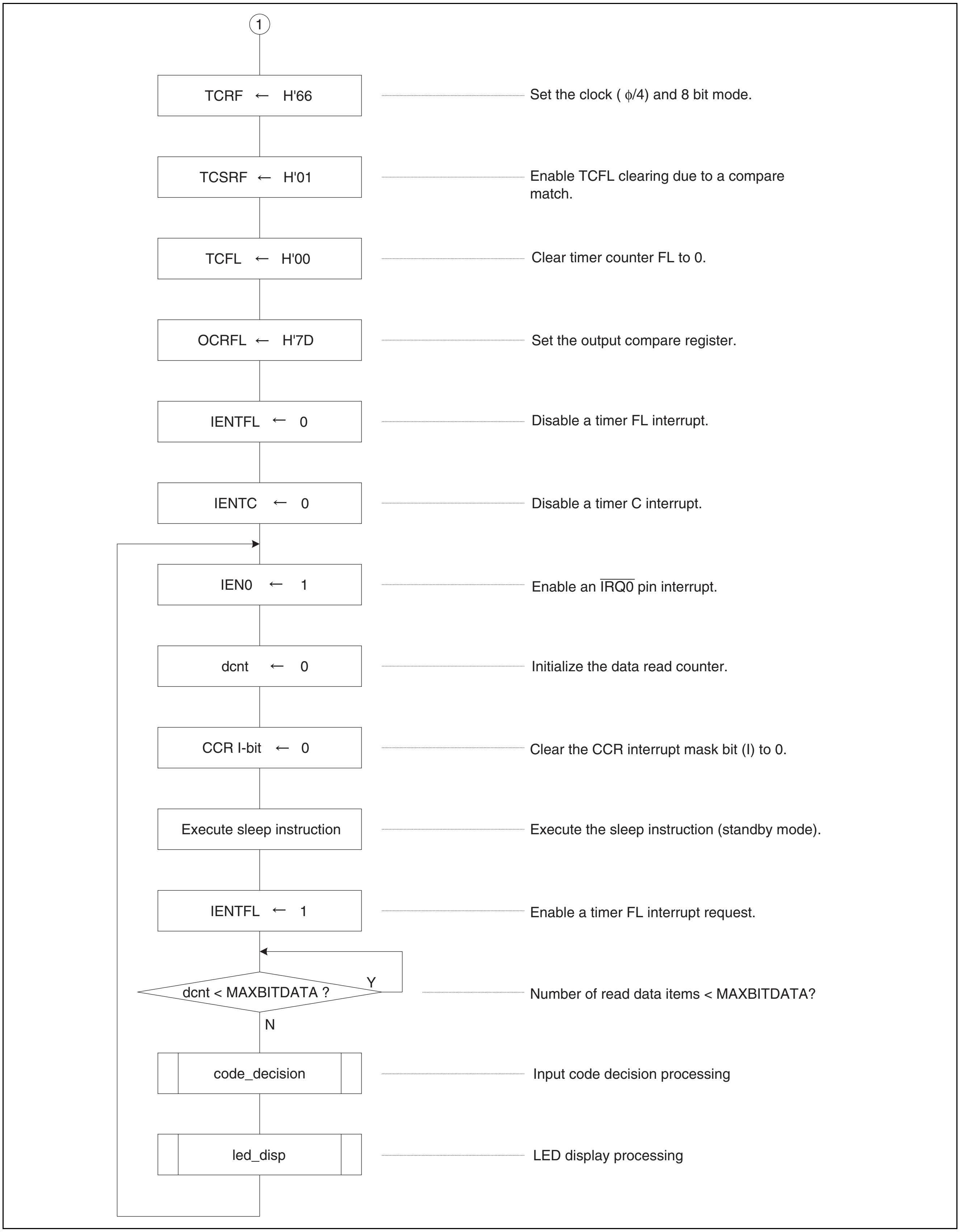
Label name	Description	Address	Module label name
dig_0	Stores LED1 display data (1 byte).	H′FB86	led_disp, tmrc
dig_1	Stores LED2 display data (1 byte).	H′FB87	led_disp, tmrc
cnt	8-bit counter for switching display LED1 and LED2 (1 byte)	H′FB88	tmrc
i	Stores the loop counter value (2 bytes).	H′FB80	code_decision, delay
ptr	Pointer used for switching display LED1 and LED2 (2 bytes)	H′FB82	tmrc
dcnt	Receive bit data counter (2 bytes)	H′FB84	main, tmrf
data	Stores bit data at reception (700 bytes).	H′FB89	code_decision, tmrf
leddata	Stores the code section extracted from the bit data (100 bytes).	H′FE45	code_decision, led_disp
led_cnt	Stores the display digit of leddata (1 byte).	H′FEAA	led_disp
bit_cnt	Stores the on/off bit position (1 byte).	H′FEAB	code_decision
pulse_cnt	Counts the high level of each pulse (1 byte).	H′FEAC	code_decision
byte_cnt	Counts the number of leddata items (1 byte).	H′FEAD	code_decision, led_disp

5. Flowchart

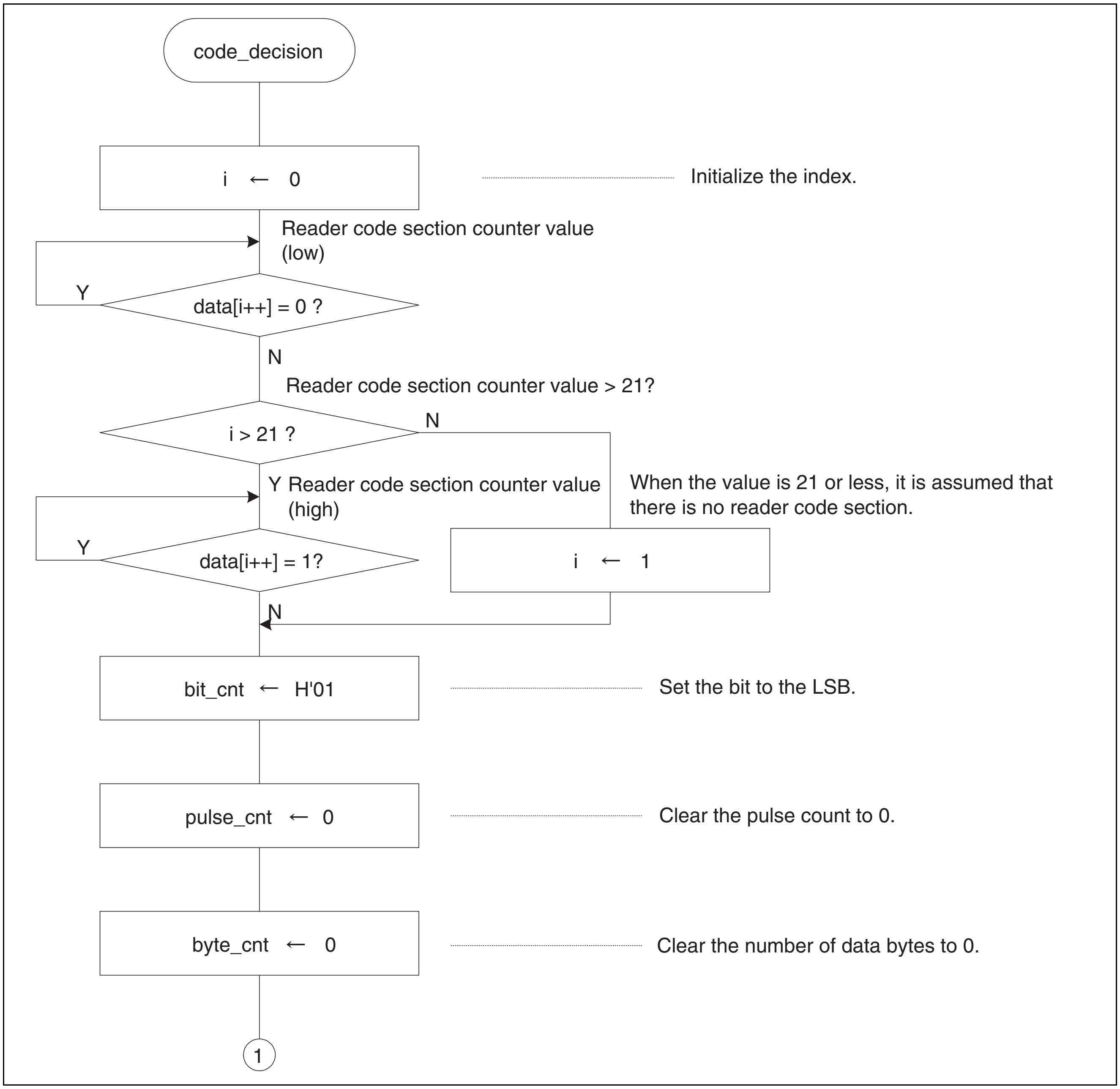
1. Main routine (main)

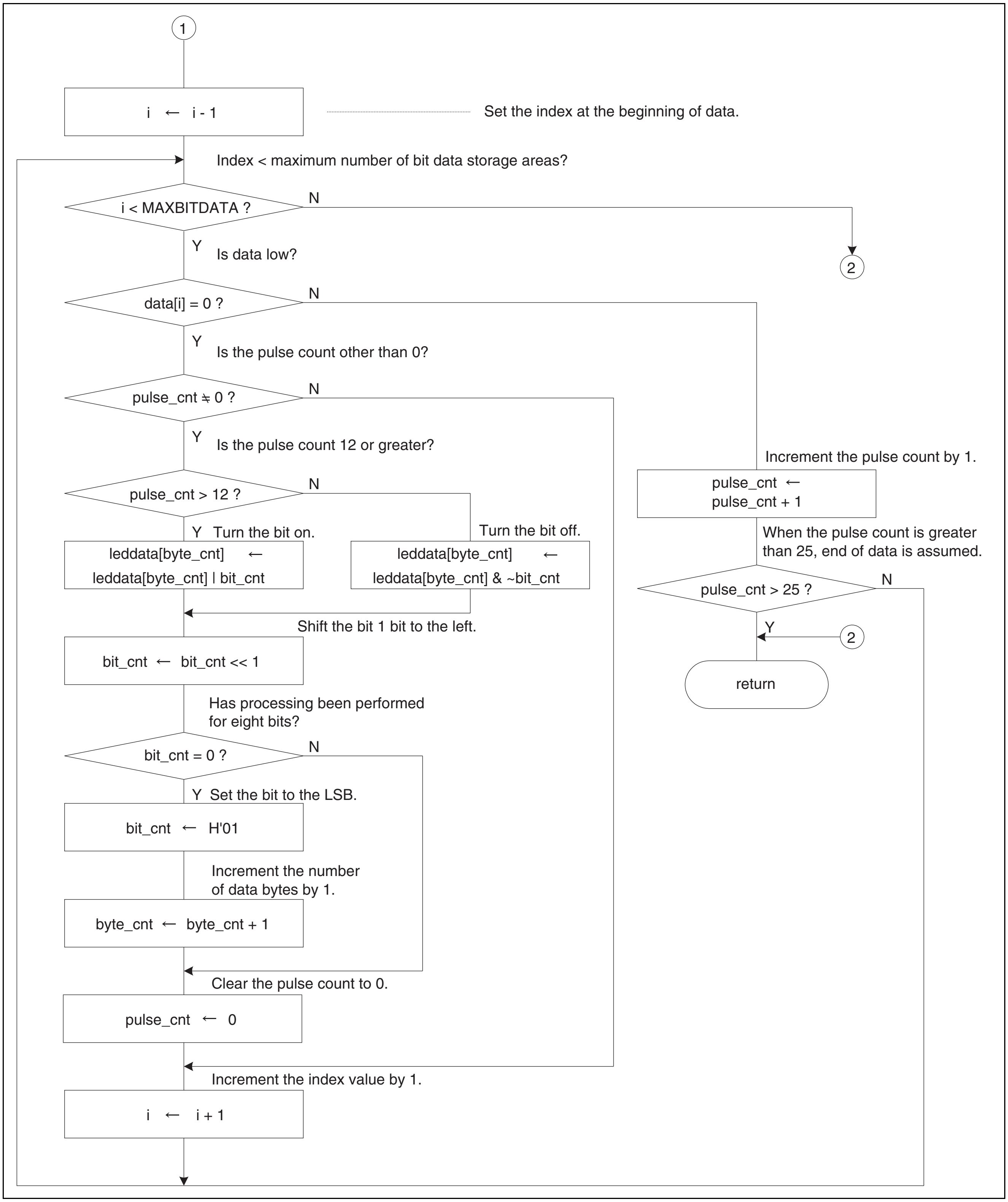


[Note] *1 In this sample task, INIT.SRC (assembly language) is used to set the stack pointers.

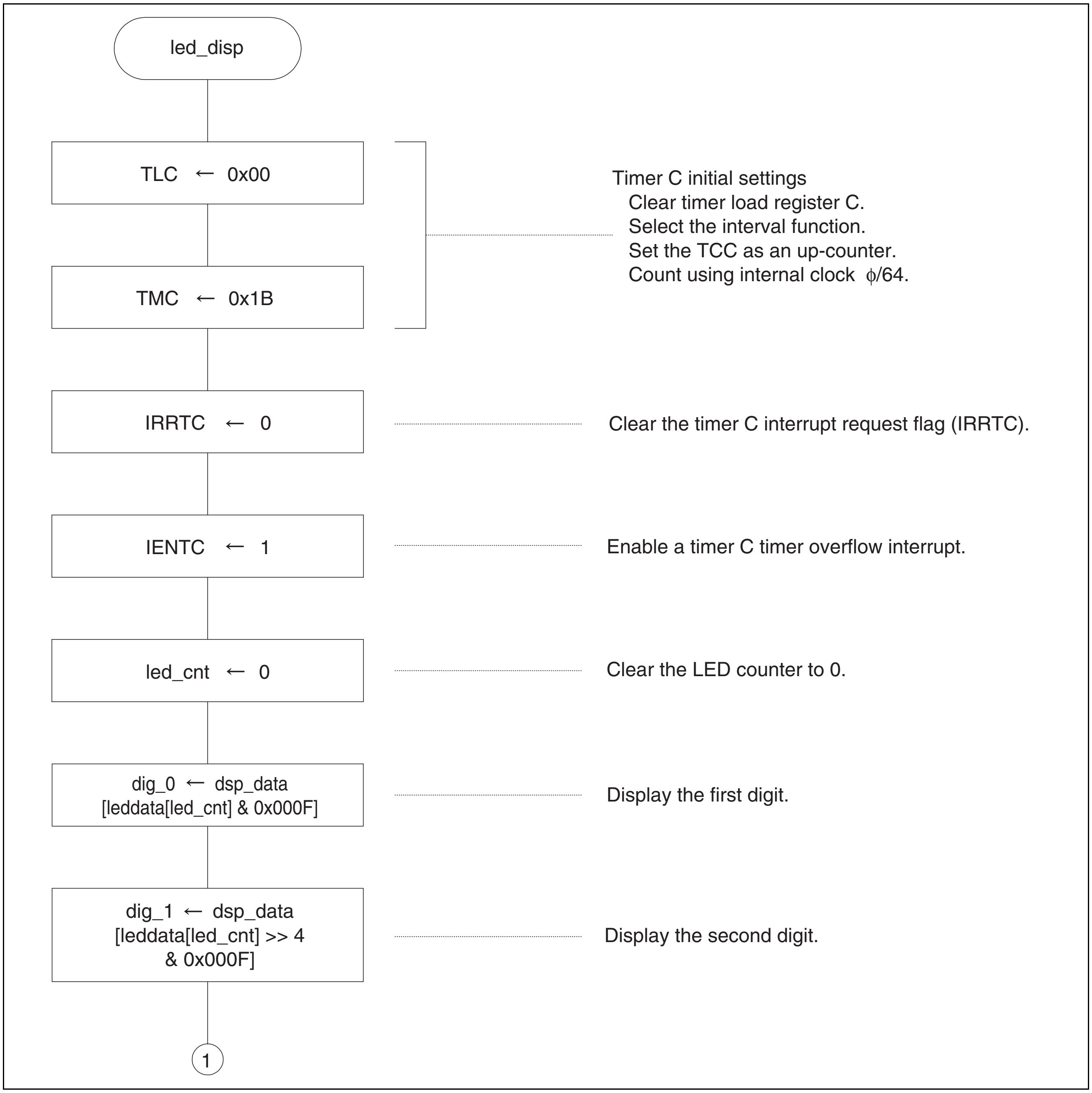

1
TCRF ← H'66
Set the clock (φ/4) and 8 bit mode.
TCSRF ← H'01
Enable TCFL clearing due to a compare match.
TCFL ← H'00
Clear timer counter FL to 0.
OCRFL ← H'7D
Set the output compare register.
IENTFL ← 0
Disable a timer FL interrupt.
IENTC ← 0
Disable a timer C interrupt.
IEN0 ← 1
Enable an IRQ0 pin interrupt.
dcnt ← 0
Initialize the data read counter.
CCR I-bit ← 0
Clear the CCR interrupt mask bit (I) to 0.
Execute sleep instruction
Execute the sleep instruction (standby mode).
IENTFL ← 1
Enable a timer FL interrupt request.
dcnt < MAXBITDATA ?
Y
N
Number of read data items < MAXBITDATA?
code_decision
Input code decision processing
led_disp
LED display processing

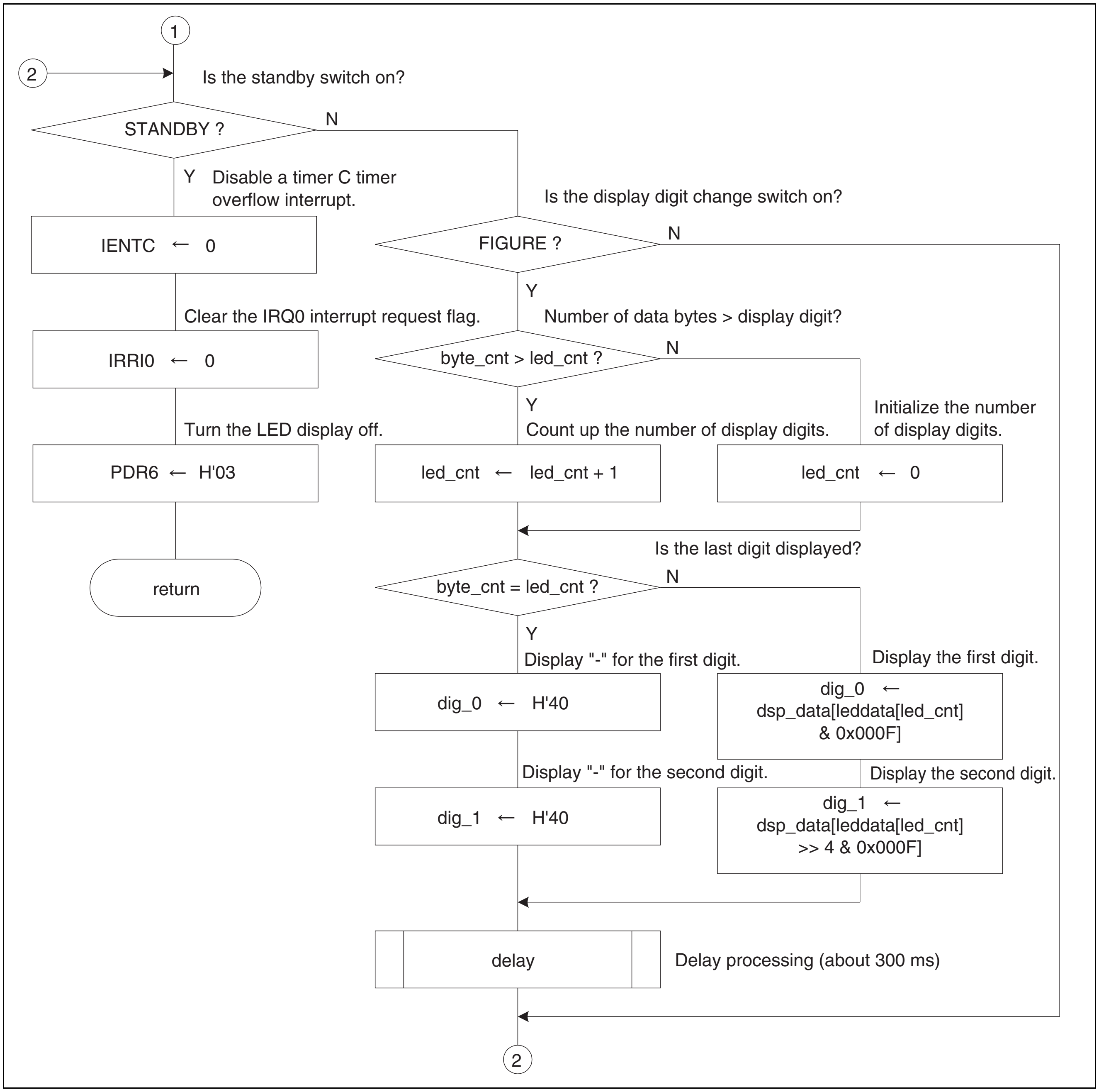
2. Code decision processing (code_decision)



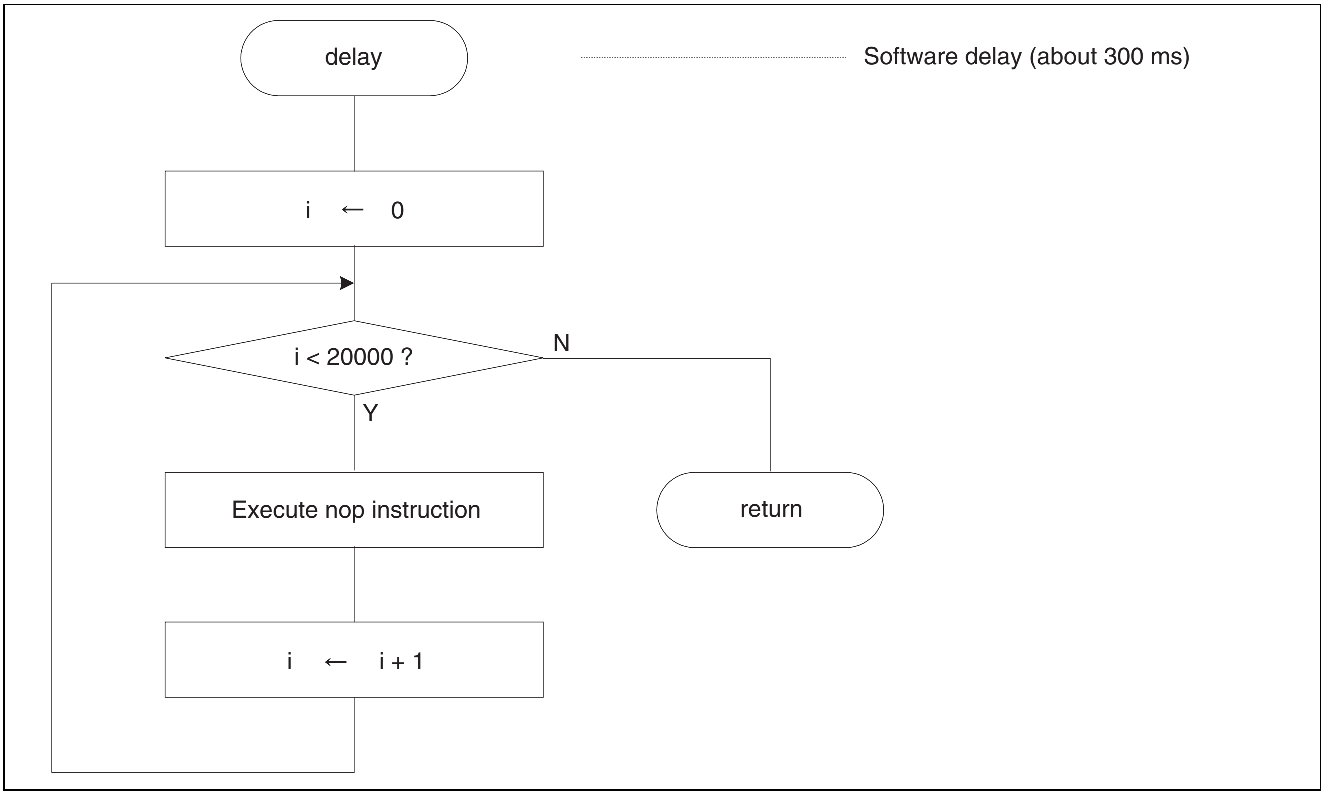

1
i ← i - 1
Set the index at the beginning of data.
Index < maximum number of bit data storage areas?
i < MAXBITDATA ?
N
Y
Is data low?
2
data[i] = 0 ?
N
Y
Is the pulse count other than 0?
pulse_cnt ≠ 0 ?
N
Y
Is the pulse count 12 or greater?
Increment the pulse count by 1.
pulse_cnt ← pulse_cnt + 1
pulse_cnt > 12 ?
N
Y Turn the bit on.
Turn the bit off.
When the pulse count is greater than 25, end of data is assumed.
leddata[byte_cnt] ← leddata[byte_cnt] | bit_cnt
leddata[byte_cnt] ← leddata[byte_cnt] & ~bit_cnt
pulse_cnt > 25 ?
N
Y
2
Shift the bit 1 bit to the left.
bit_cnt ← bit_cnt << 1
return
Has processing been performed for eight bits?
bit_cnt = 0 ?
N
Y Set the bit to the LSB.
bit_cnt ← H'01
Increment the number of data bytes by 1.
byte_cnt ← byte_cnt + 1
Clear the pulse count to 0.
pulse_cnt ← 0
Increment the index value by 1.
i ← i + 1

3. LED display processing (led_disp)

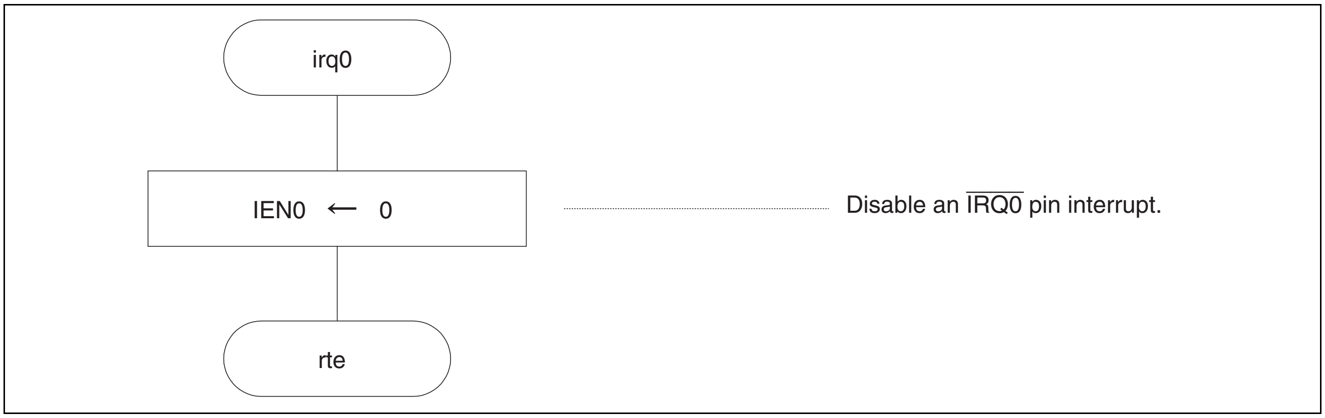



1
2
Is the standby switch on?
STANDBY ?
N
Y
Disable a timer C timer overflow interrupt.
IENTC ← 0
Clear the IRQ0 interrupt request flag.
IRRI0 ← 0
Turn the LED display off.
PDR6 ← H'03
return
Is the display digit change switch on?
FIGURE ?
N
Y
Number of data bytes > display digit?
byte_cnt > led_cnt ?
N
Y
Count up the number of display digits.
led_cnt ← led_cnt + 1
Initialize the number of display digits.
led_cnt ← 0
Is the last digit displayed?
byte_cnt = led_cnt ?
N
Y
Display "-" for the first digit.
dig_0 ← H'40
Display the first digit.
dig_0 ← dsp_data[leddata[led_cnt] & 0x000F]
Display "-" for the second digit.
dig_1 ← H'40
Display the second digit.
dig_1 ← dsp_data[leddata[led_cnt] >> 4 & 0x000F]
delay
Delay processing (about 300 ms)
2

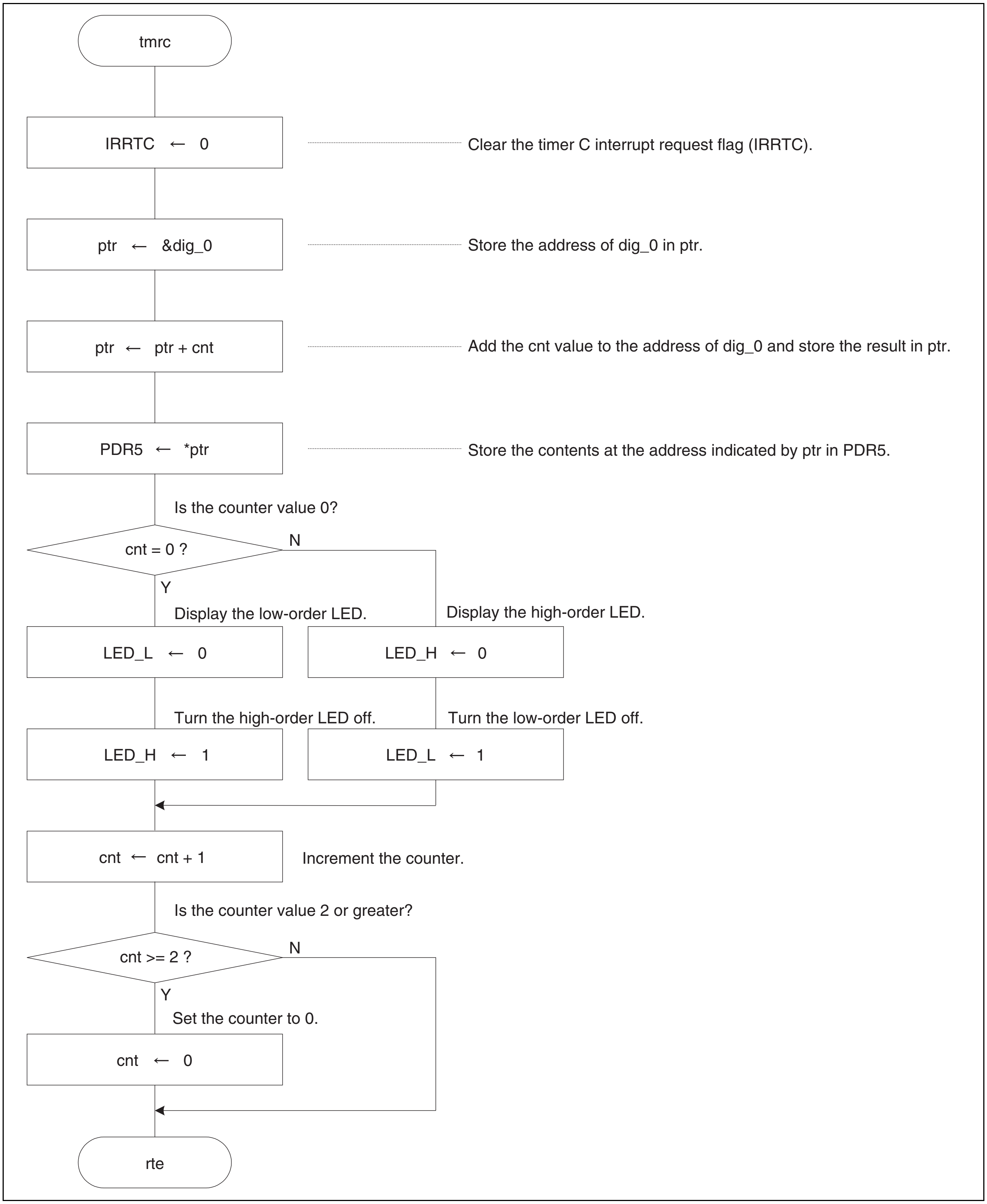
4. Software delay processing (delay)



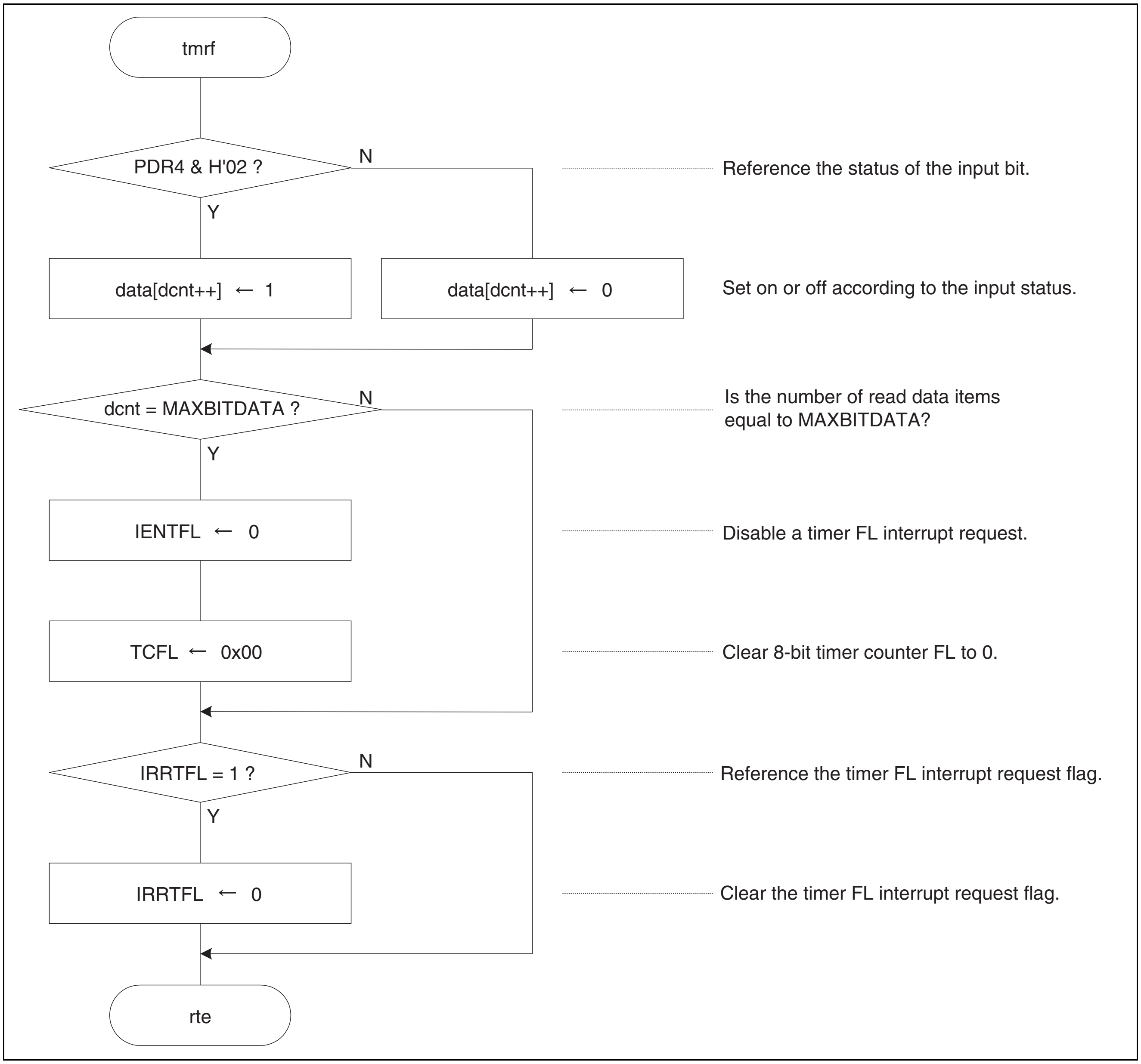
5. $\overline{\text{IRQ0}}$ interrupt processing (irq0)



6. Timer C interrupt processing (tmrc)



7. Timer F interrupt processing (tmrf)



6. Program Listing

INIT.SRC (program listing)

```
   .export  _INIT
   .import  _main
;
   .section P,CODE
_INIT:
   mov.w #h'ff80,r7
   ldc.b #b'10000000,ccr
   jmp   @_main
;
   .end
```

```
/* Super Low Power Series -H8/38024- Application note */
/* Application example */
/* Remote control */

#include <machine.h>

/* Symbol Definition */
struct BIT {
   unsigned char b7:1;      /* bit 7 */
   unsigned char b6:1;      /* bit 6 */
   unsigned char b5:1;      /* bit 5 */
   unsigned char b4:1;      /* bit 4 */
   unsigned char b3:1;      /* bit 3 */
   unsigned char b2:1;      /* bit 2 */
   unsigned char b1:1;      /* bit 1 */
   unsigned char b0:1;      /* bit 0 */
};

#define H   1                                         /* High Level */
#define L   0                                         /* Low Level */
#define MAXBITDATA   700                              /* max bit data size */
#define MAXLEDDATA   100                              /* max led data size */

#define PDR4   *(volatile unsigned char *)0xFFD7    /* Port data register 4 */
#define PCR4   *(volatile unsigned char *)0xFFE7    /* Port control register 4
*/
#define PMR2   *(volatile unsigned char *)0xFFC9    /* Port mode register 2 */

#define PMR5   *(volatile unsigned char *)0xFFCC    /* Port mode register 5 */
#define PUCR5  *(volatile unsigned char *)0xFFE2    /* Port pull-up control
register 5 */
#define PDR5   *(volatile unsigned char *)0xFFD8    /* Port data register 5 */
#define PCR5   *(volatile unsigned char *)0xFFE8    /* Port control register 5
*/

#define PDR6   *(volatile unsigned char *)0xFFD9    /* Port data register 6 */
#define PDR6_BIT (*(struct BIT *)0xFFD9)
#define STANDBY   PDR6_BIT.b3                         /* standby switch */
```

```
#define FIGURE PDR6_BIT.b2                          /* LED figure switch */
#define LED_H  PDR6_BIT.b1                          /* LED(HIGH) ON/OFF */
#define LED_L  PDR6_BIT.b0                          /* LED(LOW) ON/OFF */
#define PCR6   *(volatile unsigned char *)0xFFE9    /* Port control register 6
*/

#define TCRF   *(volatile unsigned char *)0xFFB6    /* timer control register
F */
#define TCSRF  *(volatile unsigned char *)0xFFB7    /* timer control status
register F */
#define TCSRF_BIT (*(struct BIT *)0xFFB7)
#define CMFL   TCSRF_BIT.b2                         /* Compare-Match Flag L */
#define TCFL   *(volatile unsigned char *)0xFFB9    /* 8 bit timer counter
F(LOW)  */
#define OCRFL  *(volatile unsigned char *)0xFFBB    /* 8 bit output compare
register F(LOW)  */

#define TMC    *(volatile unsigned char *)0xFFB4    /* Timer mode register C
*/
#define TLC    *(volatile unsigned char *)0xFFB5    /* Timer Load register C
*/

#define IENR1  *(volatile unsigned char *)0xFFF3    /* Interrupt enable
register 1 */
#define IENR1_BIT (*(struct BIT *)0xFFF3)
#define IEN0   IENR1_BIT.b0                         /* IRQ0 interrupt enable
*/
#define IRR1   *(volatile unsigned char *)0xFFF6    /* Interrupt request
register 1 */
#define IRR1_BIT (*(struct BIT *)0xFFF6)
#define IRRI0  IRR1_BIT.b0                          /* IRQ0 interrupt request
flag */

#define IENR2  *(volatile unsigned char *)0xFFF4    /* interrupt enable
register 2 */
#define IENR2_BIT (*(struct BIT *)0xFFF4)
#define IENTFL IENR2_BIT.b2                         /* Timer FL interrupt
enable */
#define IENTC  IENR2_BIT.b1                         /* Timer C interrupt
enable */
#define IRR2   *(volatile unsigned char *)0xFFF7    /* interrupt request
register 2 */
#define IRR2_BIT (*(struct BIT *)0xFFF7)
#define IRRTFL IRR2_BIT.b2                          /* Timer FL interrupt
enable */
#define IRRTC  IRR2_BIT.b1                          /* Timer C interrupt
request flag */

#define SYSCR1 *(volatile unsigned char *)0xFFF0    /* system control register
1 */
#define SYSCR2 *(volatile unsigned char *)0xFFF1    /* system control register
2 */
```

```
#pragma interrupt (irq0)
#pragma interrupt (tmrc)
#pragma interrupt (tmrf)

/* Function define */
extern void INIT(void);                     /* Stack pointer set */
void main(void);                            /* main routine */
void code_decision(void);                   /* code decision routine */
void led_disp(void);                        /* LED display routine */
void delay(void);                           /* delay routine */
void irq0(void);                            /* IRQ0 routine */
void tmrc(void);                            /* Timer C interrupt routine */
void tmrf(void);                            /* Timer F interrupt routine */

/* Data table */
const unsigned char dsp_data[16] =
{
   0x3f,                                    /* LED display data = "0" */
    0x06,                                   /* LED display data = "1" */
    0x5b,                                   /* LED display data = "2" */
    0x4f,                                   /* LED display data = "3" */
    0x66,                                   /* LED display data = "4" */
    0x6d,                                   /* LED display data = "5" */
    0x7d,                                   /* LED display data = "6" */
    0x27,                                   /* LED display data = "7" */
    0x7f,                                   /* LED display data = "8" */
    0x6f,                                   /* LED display data = "9" */
    0x77,                                   /* LED display data = "A" */
    0x7c,                                   /* LED display data = "B" */
    0x39,                                   /* LED display data = "C" */
    0x5e,                                   /* LED display data = "D" */
    0x79,                                   /* LED display data = "E" */
    0x71                                    /* LED display data = "F" */
};

/* RAM define */
unsigned char dig_0;                        /* Dig-0 LED display data store
*/
unsigned char dig_1;                        /* Dig-1 LED display data store
*/
unsigned char cnt;                          /* LED enable counter */
int i;                                      /* loop counter   */
unsigned char *ptr;                         /* Pointer set */
int dcnt;                                   /* read data counter */
unsigned char data[MAXBITDATA];             /* read data(bit data) */
unsigned char leddata[MAXLEDDATA];          /* read data(led data) */
unsigned char led_cnt;                      /* led display counter */
unsigned char bit_cnt;                      /* 8bit counter */
unsigned char pulse_cnt;                    /* pulse counter */
unsigned char byte_cnt;                     /* byte counter */

/* Vector address */
#pragma section V1                          /* Vector section set */
void (*const VEC_TBL1[])(void) = {
```

```
   INIT                                          /* H'0000 Reset vector */
};
#pragma section V2                               /* Vector section set */
void (*const VEC_TBL2[])(void) = {
   irq0                                          /* H'0008 IRQ0 vector */
};
#pragma section V3                               /* Vector section set */
void (*const VEC_TBL3[])(void) = {
   tmrc                                          /* H'001a Timer C interrupt
vector */
};
#pragma section V4                               /* Vector section set */
void (*const VEC_TBL4[])(void) = {
   tmrf                                          /* H'001c Timer F interrupt
vector */
};
#pragma section                                  /* P */

/**************************************************************/
/* Main program                                             */
/**************************************************************/
void main(void)
{
   set_imask_ccr(1);                             /* CCR I-bit = 1  */


   PCR5 = 0xFF;                                  /* Initialize : output LED */
   PDR5 = 0x00;                                  /* LED clear */
   PCR6 = 0x03;                                  /* Initialize : input SW & LED
control*/
   PDR6 = 0x03;                                  /* LED OFF */

   SYSCR1 = 0xF7;                                /* standby mode, 16 state */

   PMR2 = 0xD9;                                  /* Initialize : use IRQ0 */
   IRRI0 = 0;                                    /* clear IRQ0 interrupt request
flag */

   TCRF = 0x66;                                  /* Set intrernal clock : ƒÓ/4 */
   TCSRF = 0x01;                                 /* Enable TCFL clear */
   TCFL = 0x00;                                  /* clear Timer Counter FL to 0
      */
   OCRFL = 0x7D;                                 /* set interrupt interval to
0.1msec  */

   IENTFL = 0;                                   /* Timer FL interrupt disable */
   IENTC = 0;                                    /* Timer C interrupt disable */

   while(1){
      IEN0 = 1;                                  /* IRQ0 enable */

      dcnt = 0;                                  /* clear read data count */

      set_imask_ccr(0);                          /* CCR I-bit = 0  */
```

```
      sleep();                              /* standby */

      IENTFL = 1;                           /* enable interrupt request FL
*/

      while(dcnt < MAXBITDATA);

      code_decision();                      /* code decision routine */

      led_disp();                           /* LED display routine */
   }
}

/************************************************************/
/*    code decision routine                                 */
/************************************************************/
void code_decision(void)
{
   i = 0;
   while(data[i++] == 0);                   /* Leader Code */
   if(i > 21)                               /* then leader cord follows */
      while(data[i++]);                     /* Leader Code */
   else                                     /* else data code */
      i = 1;                                /* initialize index */

   bit_cnt = 0x01;                          /* set bit counter to LSB */
   pulse_cnt = 0;                           /* pulse counter to zero clear
*/
   byte_cnt = 0;                            /* byte counter to zero clear */
   for(i=i-1;i<MAXBITDATA;i++){
      if(data[i] == L){                     /* Low ? */
         if(pulse_cnt){
            if(pulse_cnt > 12)
               leddata[byte_cnt] |= bit_cnt;   /* bit on */
            else
               leddata[byte_cnt] &= ~bit_cnt;  /* bit off */
            bit_cnt <<= 1;                  /* bit shift */
            if(!bit_cnt){                   /* next byte ? */
               bit_cnt = 0x01;              /* set bit counter to LSB */
               byte_cnt++;                  /* count up */
            }
            pulse_cnt = 0;                  /* pulse counter to zero clear
*/
         }
      } else {                              /* High */
         pulse_cnt++;                       /* count up */
         if(pulse_cnt > 25)
            break;
      }
   }
}

/************************************************************/
/*    LED display routine                                   */
```

```
/************************************************************/
void led_disp(void)
{
   TLC   = 0x00;                                                /* Clear Timer
Load register C to 0 */
    TMC   = 0x1B;                                               /* Timer C
initialize */
    IRRTC = 0;                                                  /* Clear IRRTC
to 0 */
   IENTC = 1;                                                   /* Timer C
interrupt enable */

   led_cnt = 0;
   dig_0 = dsp_data[leddata[led_cnt]  & 0x0F];                  /* Dig-0 LED
display data set */
    dig_1 = dsp_data[leddata[led_cnt]  >> 4 & 0x0F];            /* Dig-1 LED
display data set */
   while(1){
      if(STANDBY){                                              /* standby
switch ON ? */
         delay();
         IENTC = 0;                                             /* Timer C
interrupt disable */
         IRRI0 = 0;                                             /* clear IRQ0
interrupt request flag */
         PDR6 = 0x03;                                           /* LED OFF */
         break;
      } else if(FIGURE){                                        /* LED figure
switch ON ? */
         if(byte_cnt > led_cnt)
            led_cnt++;                                          /* next byte
*/
         else
            led_cnt = 0;                                        /* start byte
*/
         if(byte_cnt == led_cnt){
            dig_0 = 0x40;                                       /* Dig-0 LED
display data set(-) */
            dig_1 = 0x40;                                       /* Dig-1 LED
display data set(-) */
         } else {
            dig_0 = dsp_data[leddata[led_cnt]  & 0x0F];         /* Dig-0 LED
display data set */
            dig_1 = dsp_data[leddata[led_cnt]  >> 4 & 0x0F];    /* Dig-1 LED
display data set */
         }
         delay();
      }
   }
}

/************************************************************/
/*    delay routine(about 300msec)                          */
/************************************************************/
```

```
void delay(void)
{
   long i;

   for(i = 0;i < 20000;i++)
      nop();
}

/************************************************************/
/* Interrupt Request 0                                      */
/************************************************************/
void irq0(void)
{
   IEN0 = 0;                                  /* IRQ0 disable */
}

/************************************************************/
/* Timer C Interrupt(in order to light LED in turn)         */
/************************************************************/
void tmrc(void)
{
    IRRTC = 0;                                /* Clear IRRTC to 0 */

   ptr = &dig_0;                              /* LED display data store
address set */
    ptr += cnt;                               /* LED display data read */
    PDR5 = *ptr;                              /* LED display data output */
   if(!cnt){
      LED_L = 0;                              /* LED(LOW) ON */
      LED_H = 1;                              /* LED(HIGH) OFF */
   } else {
      LED_H = 0;                              /* LED(HIGH) ON */
      LED_L = 1;                              /* LED(LOW) OFF */
   }
    cnt++;                                    /* "cnt" increment */
    if (cnt >= 2){                            /* 2 times end ? */
      cnt = 0;                                /* "cnt" initialize */
   }
}

/************************************************************/
/* Timer F Interrupt(every 0.1msec)                         */
/************************************************************/
void tmrf(void)
{
   if(PDR4 & 0x02)
      data[dcnt++] = 1;                       /* bit ON */
   else
      data[dcnt++] = 0;                       /* bit OFF */

   if(dcnt == MAXBITDATA){                    /* end ? */
      IENTFL = 0;                             /* disable interrupt request FL
*/
```

```
      TCFL = 0x00;                         /* clear Timer Counter FL to 0
      */
   }
   if ( IRRTFL == 1 ) {
      IRRTFL = 0;                          /* Clear Compare match flag A */
   }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jul.16.04	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.