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.

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.

H8/300H Tiny Series

Example of Remote Control Signal Reception from Power-Down Mode

Introduction

Using an interrupt input signal received by an infrared receiver for remote control, the H8/36014 in standby mode is caused to enter an active mode. The H8/36014 then receives data from the infrared remote controller.

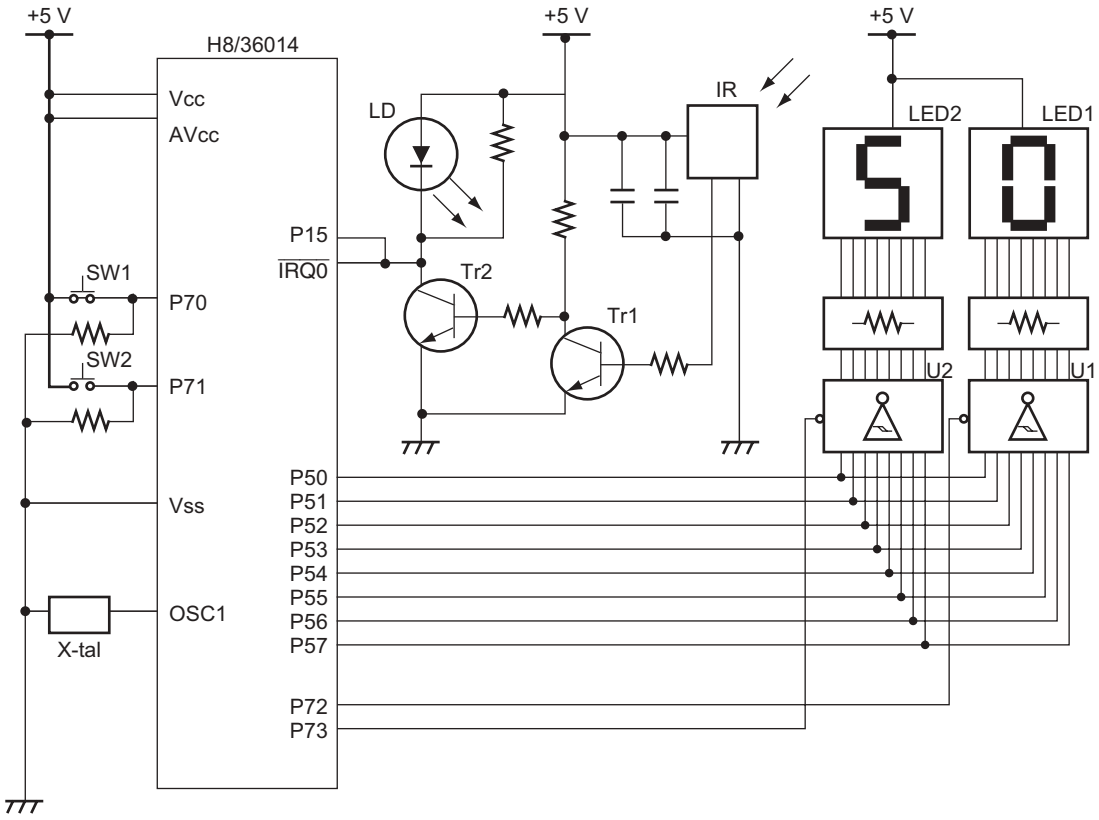
Target Device

H8/300H Tiny Series H8/36014 CPU

Contents

1. Specifications

1. Figure 1 shows the hardware configuration for infrared remote control data reception using an infrared receiver for remote control.
2. In this sample task, an infrared remote control signal causes the microcomputer to make a transition from standby mode, a power-down mode of the microcomputer, to active mode. The microcomputer then receives data signals.
3. The received data is displayed in two hexadecimal digits (one byte) on seven-segment LEDs. Each time a push-button switch (SW1) is pressed, the display is shifted one byte in succession.
4. The microcomputer can be put into standby mode again by pressing another push-button switch (SW2).
5. In this sample task, the operating voltage (Vcc) and analog power supply voltage (AVcc) of the H8/36014 are 5 V. An external crystal oscillator is used as a clock source to obtain the OSC clock frequency of 10 MHz.



[Legend]

IR:	IR receiver for remote control (GP1UD281XK, Sharp Corporation)
X-tal:	Oscillator (10 MHz)
LED1, LED2:	Anode-common 7-segment LED
LD:	Light-emitting diode
U1, U2:	3-state output inverter driver (HD74LS240)
SW1, SW2:	Push-button switch
Tr1, Tr2:	Transistor (2SC1815, Toshiba Corporation)

Figure 1 Hardware Configuration

6. The infrared receiver used in this task is an infrared detecting unit for remote control, model GP1UD281XK, from Sharp Corporation. Its specifications are as follows.
   A. Figure 2 shows the block diagram of the infrared receiver for remote control.

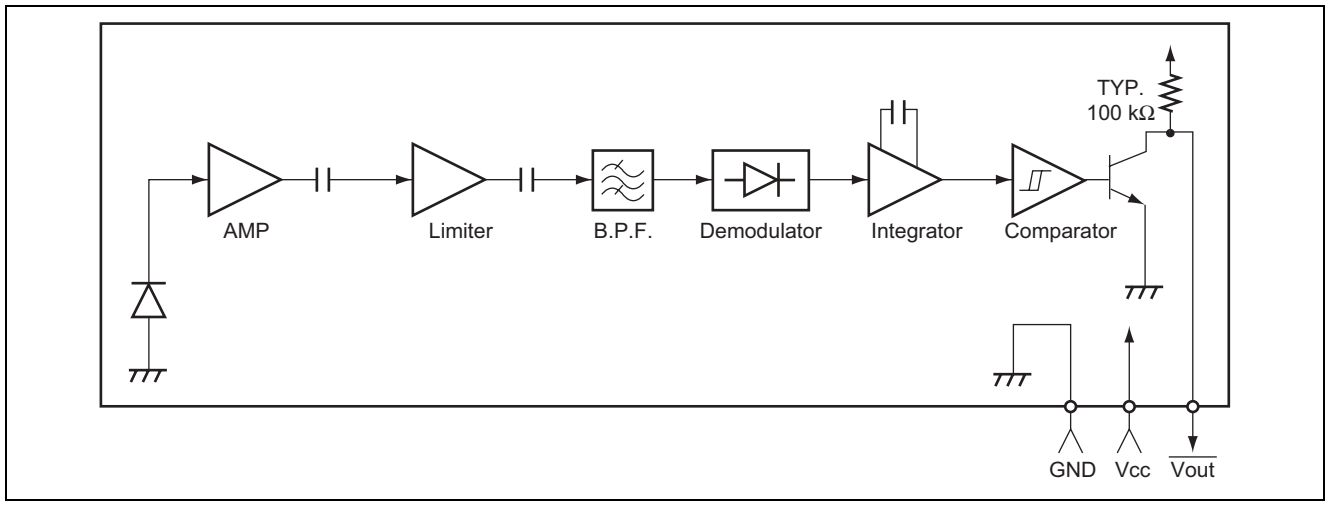


Figure 2 Block Diagram of the Infrared Receiver

   B. Features of the GP1UD281XK are as follows.
      a. Power supply voltage range is from 2.7 to 5.5 V.
      b. Carrier frequency is 38 kHz.
      c. Built-in demodulation circuit supporting PPM (Pulse Position Modulation)

7. The operation of this sample task is as follows.
   A. Operation proceeds in the order (standby mode) → (active mode) → (standby mode).
   B. The infrared signal transmitted from an infrared remote controller is received (optical reception) and demodulated by the infrared receiver, and the leading part of the signal initiates acceptance of an interrupt by the microcomputer.
   C. When the interrupt request is generated, standby mode is canceled and interrupt exception processing is started to make transition to active mode after a "wait time" (16 to 13,072 clock cycles) has elapsed, without waiting for the "oscillation stabilization time" specified in the “AC characteristics” to elapse. This is because an external clock signal is supplied.

      Oscillation stabilization wait time = Oscillation stabilization time + Wait time

   D. After accepting the interrupt, the microcomputer makes transition from standby mode to active mode when the "oscillation stabilization wait time" has elapsed.
   E. After the transition to active mode, the microcomputer receives the succeeding signals.
   F. The received data is displayed on two seven-segment LEDs. Two hexadecimal digits (one byte) are displayed; and each time a push-button switch (SW1) is pressed, the received data is shifted two digits (one byte) to display the next data byte.
   G. The infrared remote controller data used in this sample task is 4 bytes (= 32 bits). The data are displayed as follows on two seven-segment LEDs.

      "50" → "AF" → "17" → "E8" After the last two digits are displayed, "--" is displayed

(Specifications of the remote control codes are not disclosed by manufacturers. The above display is the results of byte-unit operation conforming to the LSB-first format, which is the remote control signal format generally used.)

H. After displaying of the received data is finished, standby mode can be entered again by pressing another push-button switch (SW2). The microcomputer thus goes into a state waiting for the reception of remote control signals.

I. In binary form, the received data is as follows: the first and second bytes, as well as the third and fourth bytes, are bitwise inversions of each other.

H'50 (= 0101 0000), H'AF (= 1010 1111), H'17 (= 0001 0111), H'E8 (= 1110 1000)

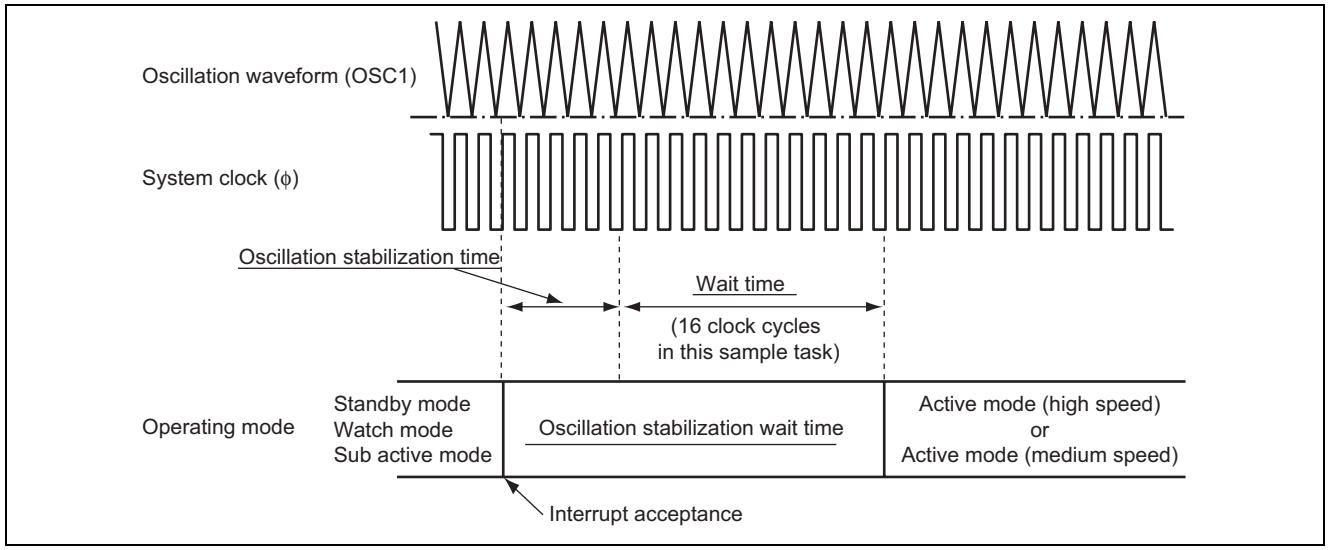


Figure 3 Oscillation Stabilization Wait Time

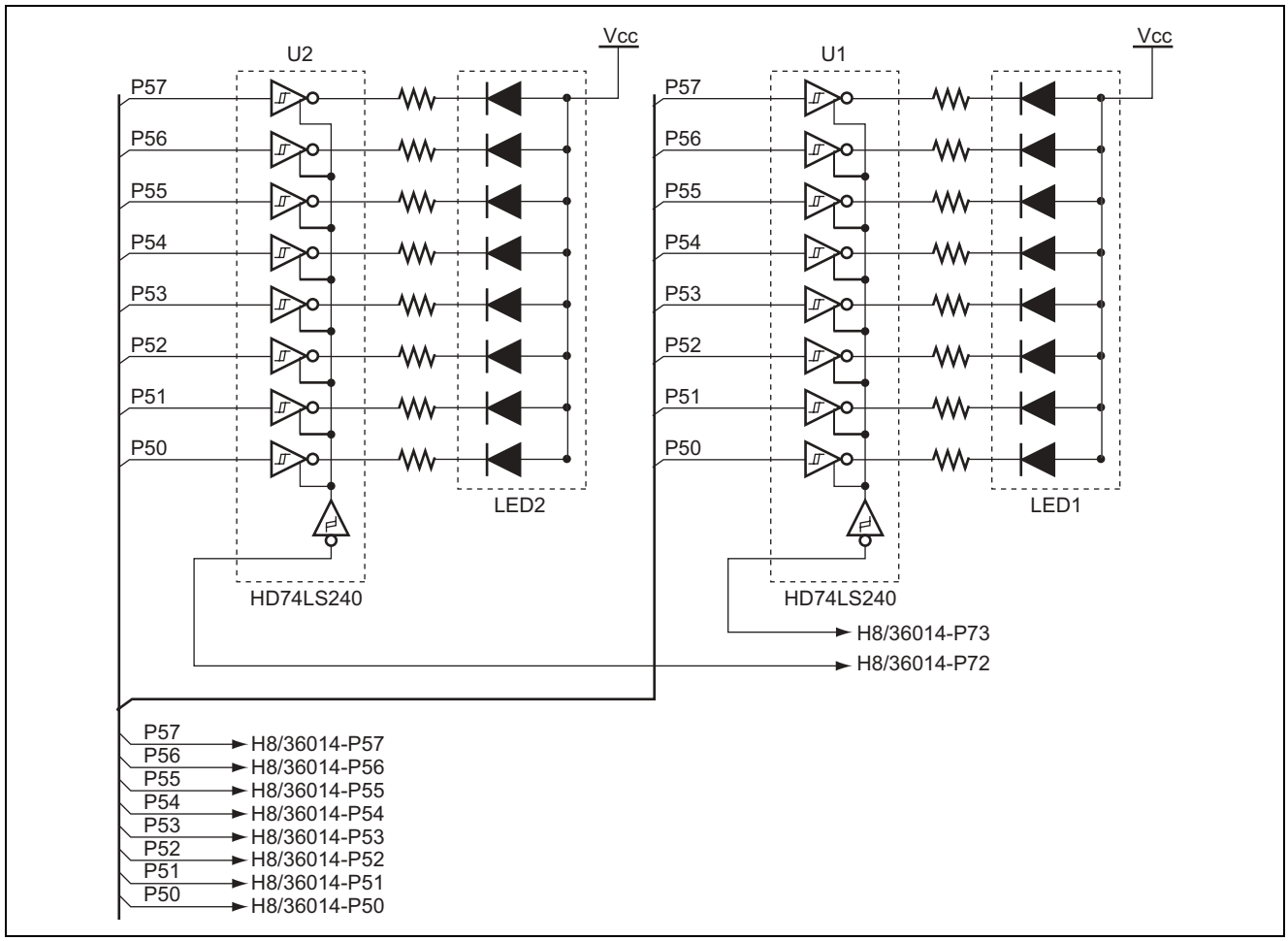


Figure 4 7-Segment LED Control

8. In this sample task, the results of remote control input are displayed in hexadecimal on seven-segment LEDs (H'FF to H'00). Figure 5 illustrates how the results of remote control input are displayed on the LEDs.

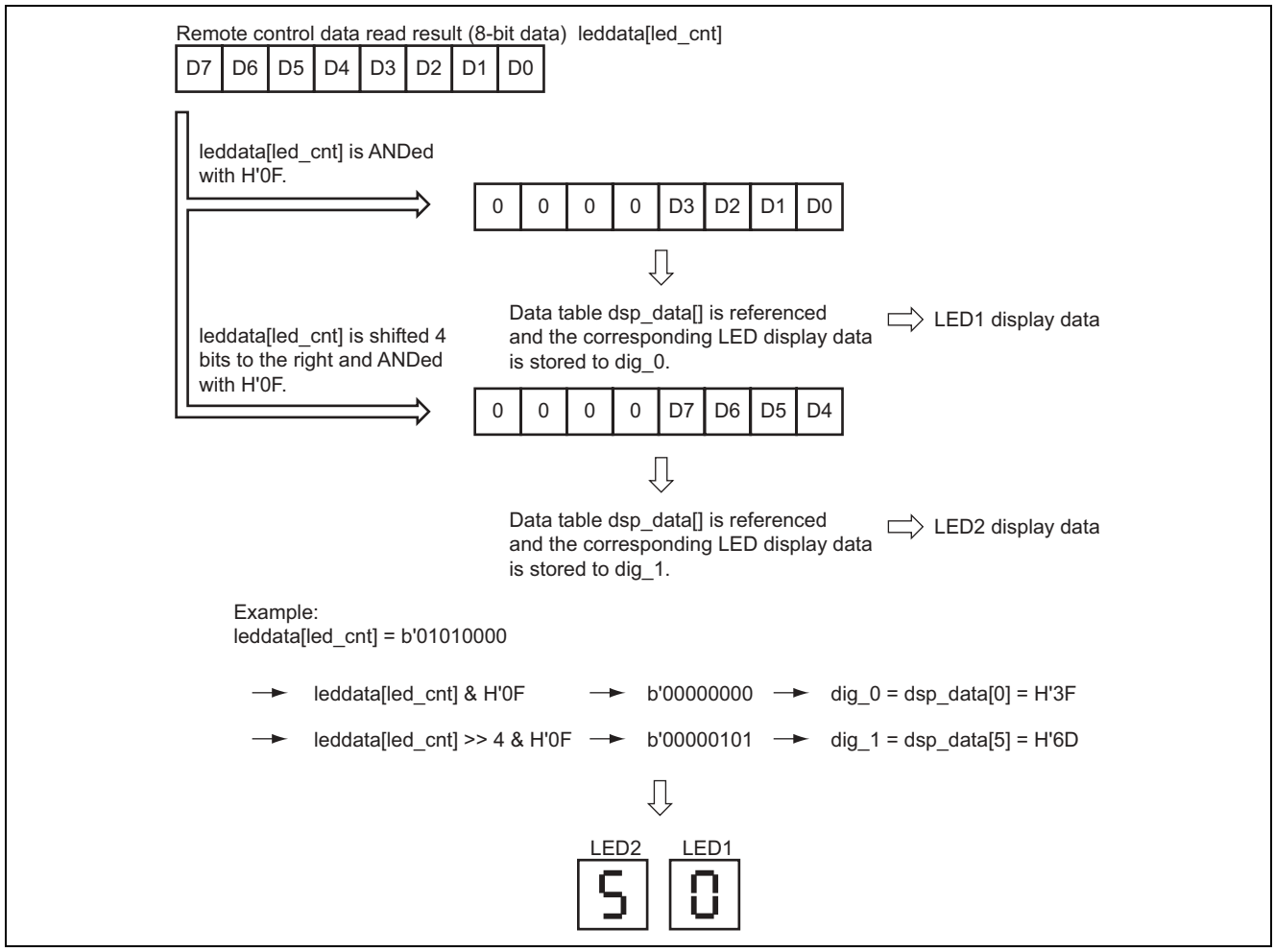


Figure 5 How Remote Control Input Data are Displayed on the LEDs

2. Description of Functions

Figure 6 shows a block diagram of the H8/36014 functions used in this sample task, while table 1 indicates the assignment of functions.

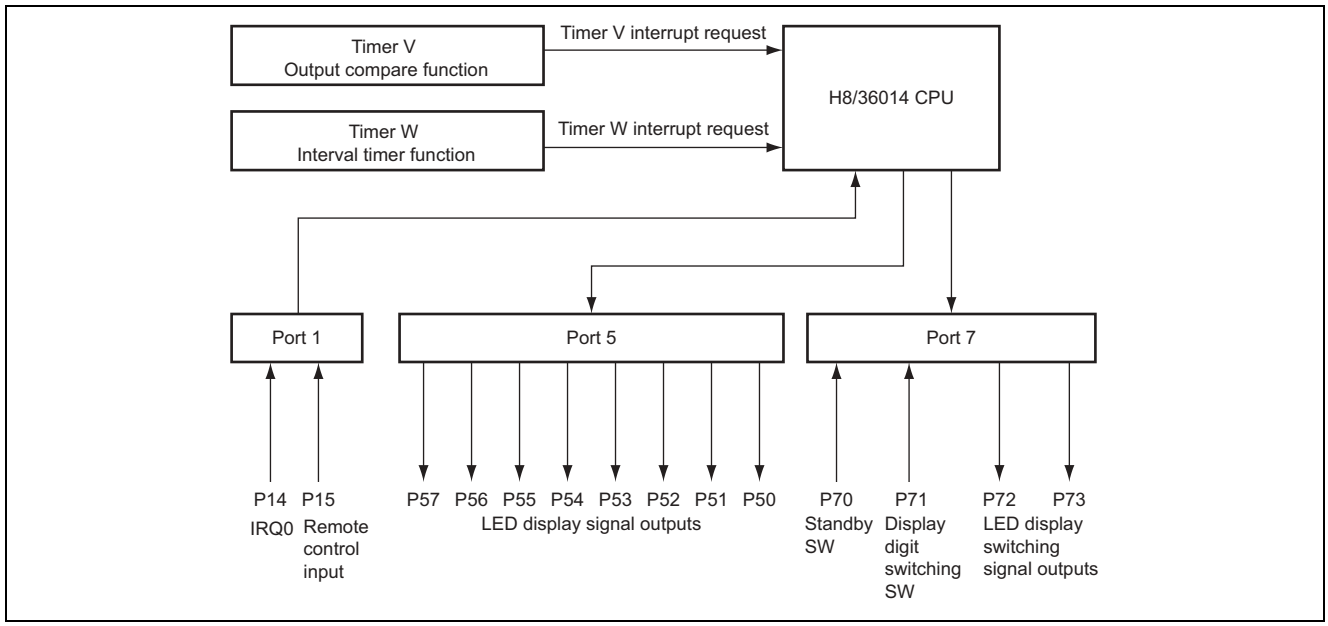


Figure 6 Block Diagram of Functions Used

Table 1 Assignment of Functions

Element	Description
Timer V	The output compare function of timer V is used to implement the periodic input of infrared data signal from a remote controller via input pin P15 at 0.1 ms intervals.
Timer W	The interval timer function of timer W is used to control the switching of seven-segment LED display. Dynamic lighting is performed by lighting the two seven-segment LEDs in order every 6.5536 ms, which is the overflow period of the timer W.
Port 1	Remote control infrared data is received from the input pin P15, and the IRQ0 interrupt of P14 causes a transition from standby mode to active (high-speed) mode.
Port 5	Data is displayed on the seven-segment LEDs using the P50 to P57 output pins. The remote control data from pin P15 is converted into two digits of hexadecimal display data and output to the LEDs.
Port 7	By alternately turning P73 and P72 on and off, the two seven-segment LEDs are lit alternately. By pressing the display switch on P71, multiple bytes of input remote control codes are displayed in sequence. By pressing the standby switch on P70, a transition from active (high-speed) mode to standby mode is made.

Figure 7 shows the connections of the seven-segment LEDs. As shown in figure 7, the corresponding LED segments are lit by outputting a high level from port 5. The relation between the port 5 output and the LED display data is given in table 2.

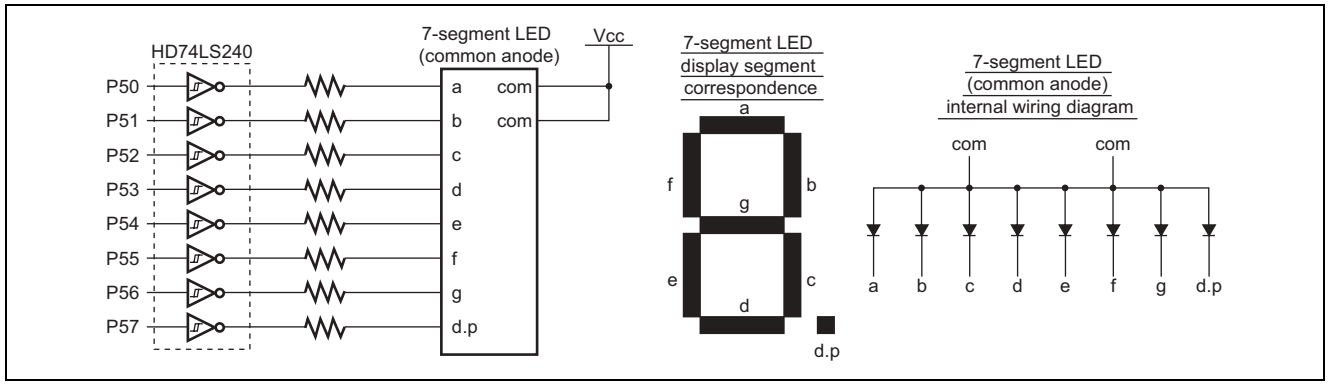
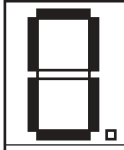
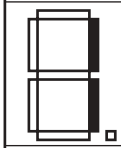
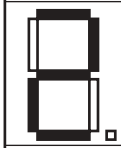
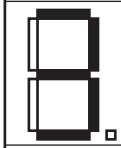
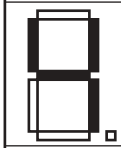
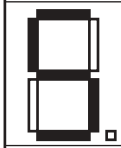
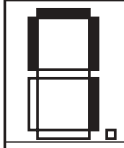
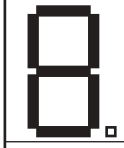
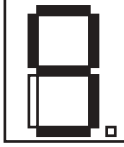


Figure 7 7-Segment LED Connection and Internal Wiring

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

LED Display	Port 5 Output Data							
	P57	P56	P55	P54	P53	P52	P51	P50
	0	0	1	1	1	1	1	1
	0	0	0	0	0	1	1	0
	0	1	0	1	1	0	1	1
	0	1	0	0	1	1	1	1
	0	1	1	0	0	1	1	0
	0	1	1	0	1	1	0	1
	0	1	1	1	1	1	0	1
	0	0	1	0	0	1	1	1
	0	1	1	1	1	1	1	1
	0	1	1	0	1	1	1	1

LED Display	Port 5 Output Data							
	P57	P56	P55	P54	P53	P52	P51	P50
	0	1	1	1	0	1	1	1
	0	1	1	1	1	1	0	0
	0	0	1	1	1	0	0	1
	0	1	0	1	1	1	1	0
	0	1	1	1	1	0	0	1
	0	1	1	1	0	0	0	1

3. Principles of Operation

1. Figure 8 shows the principles of operation that receives remote control signals using timer V. The operating mode is first set to standby mode, a remote control infrared signal is input from P15, and the resulting IRQ0 interrupt causes the CPU to make a transition to active (high-speed) mode. The state of the infrared signal is then read on timer V interrupts occurring every 0.1 ms.

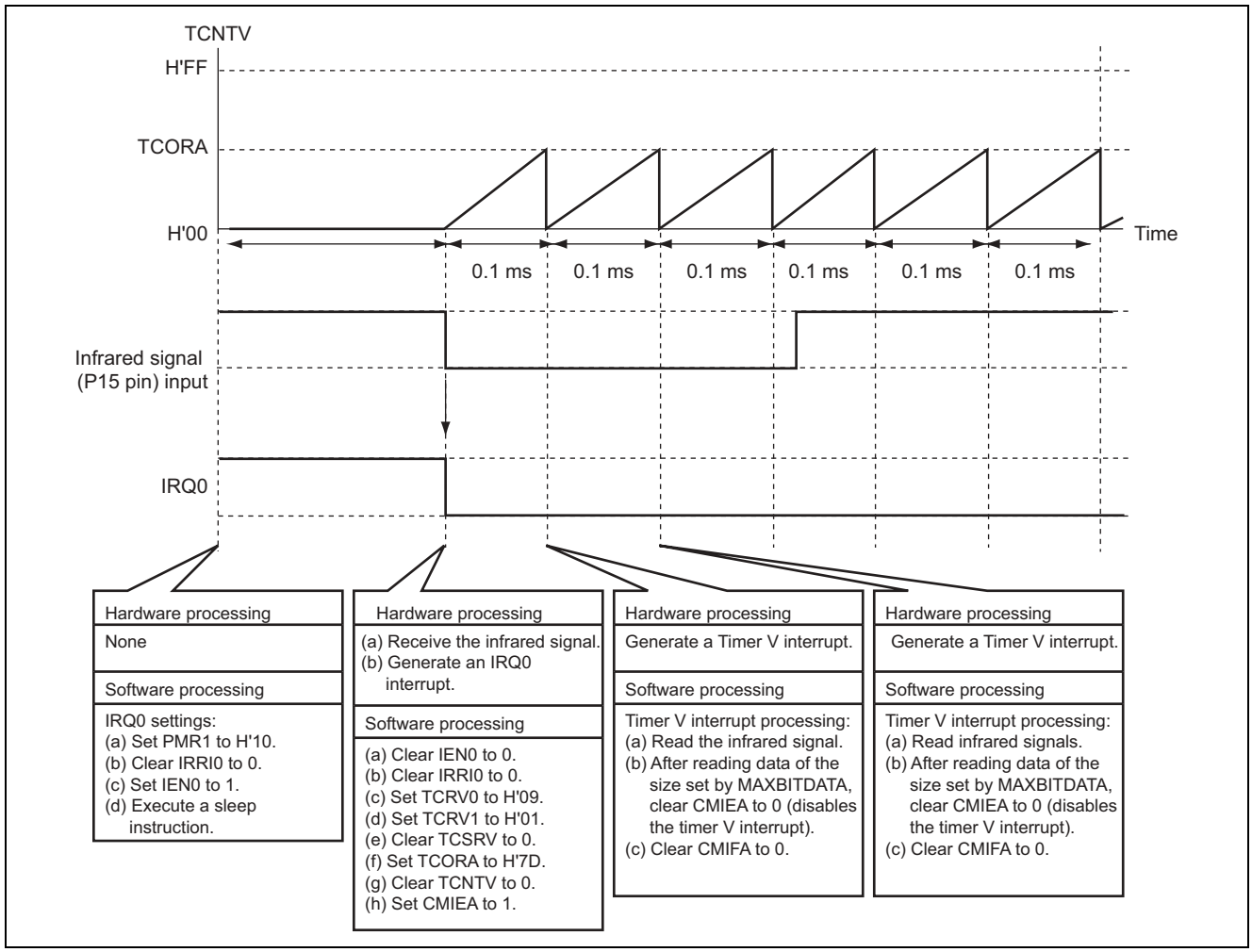


Figure 8 Principles of Operation: Remote Control Signal Reception Using Timer V

2. The principle of operation to control display of the seven-segment LEDs is explained below. Figure 9 illustrates the case when displaying "50" on LED2 and LED1. As shown in figure 9, data is displayed dynamically on the seven-segment LEDs by outputting a display on LED1 and LED2 by turns at each overflow period of timer W.

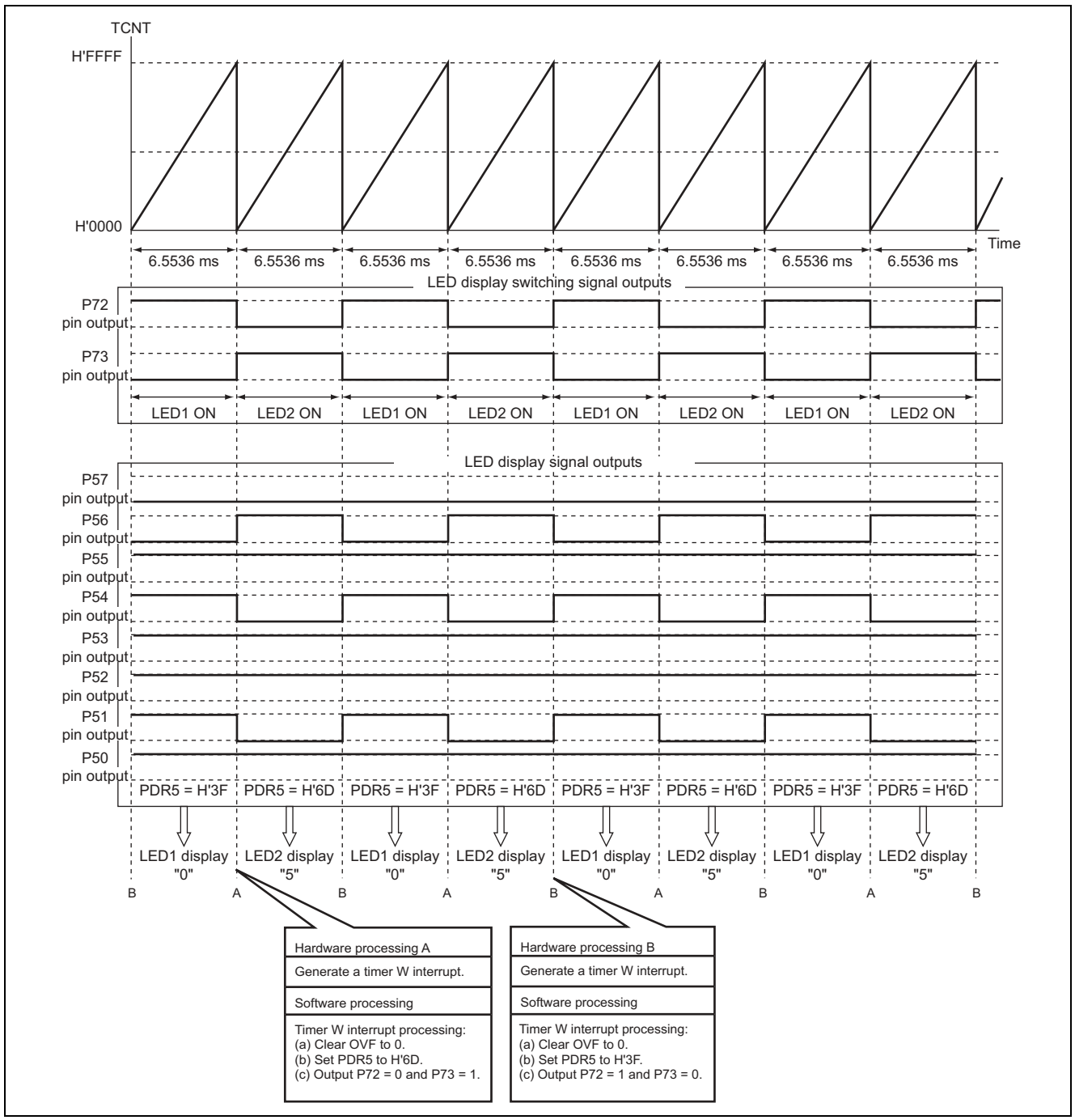


Figure 9 Principles of Operation: 7-Segment LED Display Control

4. Description of Software

4.1 Modules

Table 3 describes the modules used in this sample task.

Table 3 Description of Modules

Module Name	Label Name	Function
Main routine	main	After initialization, makes a transition to standby mode, waits for the end of data capture, performs code decision processing, and repeats the LED display processing.
Code decision processing	code_decision	Extracts codes from the data input from the remote controller.
LED display processing	led_disp	When the display switching switch is turned on, displays the multiple bytes of input remote control codes in sequence. When the standby switch is turned on, makes a transition from active (high-speed) mode to standby mode.
Software delay processing	delay	Used as an approx. 300 ms software timer.
IRQ0 interrupt processing	irq0	Disables the IRQ0 interrupt.
Timer W interrupt processing	tmrw	Clears the interrupt flag and controls the LED display data output and LED display switching.
Timer V interrupt processing	tmrv	Clears the interrupt flag and reads the state of the infrared signal.

4.2 Arguments

This sample task uses no arguments.

4.3 Internal Registers

The internal registers used in this sample task are described in table 4.

Table 4 Description of Internal Registers

Register Name		Function	Address	Setting
TCRV0		Timer control register V0 Selects TCNTV input clock, sets TCNTV clearing conditions and controls interrupt requests.	H'FFA0	H'09
	CMIEB	Compare match interrupt enable B When CMIEB = 0, disables interrupt requests by CMFB in TCSRV.	Bit 7	0
	CMIEA	Compare match interrupt enable A When CMIEA = 0, disables interrupt requests by CMFA in TCSRV. When CMIEA = 1, enables interrupt requests by CMFA in TCSRV.	Bit 6	0/1
	OVIE	Timer overflow interrupt enable When OVIE = 0, disables interrupt requests by OVF in TCSRV.	Bit 5	0
	CCLR1	Counter clear 1, 0	Bit 4	0
	CCLR0	When CCLR1 = 0 and CCLR0 = 1, clears the counter on compare match A.	Bit 3	1
	CKS2	Clock select 2 to 0	Bit 2	0
	CKS1	When CKS2 = 0, CKS1 = 0, CKS0 = 1, and ICKS0 = 1,	Bit 1	0
	CKS0	TCNTV counts on the falling edge of internal clock ϕ/8.	Bit 0	1
TCRV1		Timer control register V1 Selects TRGV pin edge, enables TRGV input, and selects TCNTV input clock.	H'FFA5	H'01
	TVEG1	TRGV input edge select 1, 0	Bit 4	0
	TVEG0	When TVEG1 = 0 and TVEG0 = 0, these bits disable trigger input from TRGV pin.	Bit 3	0
	TRGE	TRGV input enable If TRGE = 0, starting of TCNTV count-up operation by TRGV pin input and stopping of TCNTV count-up operation when TCNTV is cleared on compare-match are disabled.	Bit 2	0
	ICKS0	Internal clock select 0 When CKS2 = 0, CKS1 = 0, CKS0 = 1 and ICKS0 = 1, TCNTV counts on the falling edge of internal clock ϕ/8.	Bit 0	1

Register Name		Function	Address	Setting
TCSRV		Timer control/status register V Indicates status flag and controls output on compare-match.	H'FFA1	H'00
	CMFB	Compare-match flag B Set to 1 when the values of TCNTV and TCORB match.	Bit 7	0
	CMFA	Compare match flag A Set to 1 when the values of TCNTV and TCORA match. Cleared to 0 by writing 0 to CMFA after CMFA is read while CMFA = 1.	Bit 6	0
	OVF	Timer overflow flag Set to 1 when TCNTV value overflows.	Bit 5	0
	OS3	Output select 3, 2	Bit 3	0
	OS2	These bits select the TMOV pin output on compare-match B. When set to OS3 = 0 and OS2 = 0, the output does not change.	Bit 2	0
	OS1	Output select 1, 0	Bit 1	0
	OS0	These bits select the TMOV pin output on compare-match A. When set to OS1 = 0 and OS0 = 0, the output does not change.	Bit 0	0
TCORA		Time constant register A Used to generate an interrupt on compare-match with TCNTV.	H'FFA2	H'7D
TCNTV		Timer counter V 8-bit readable/writable up counter.	H'FFA4	0
TMRW		Timer mode register W Selects general register functions and timer output mode.	H'FF80	H'80
	CTS	Counter start CTS = 1 indicates that TCNT has started counting.	Bit 7	1
TSRW		Timer status register W Indicates interrupt request statuses.	H'FF83	H'00
	OVF	Timer overflow When OVF = 0, indicates that TCNT has not overflowed. When OVF = 1, indicates that TCNT has overflowed. Cleared to 0 by writing 0 to OVF after OVF is read while OVF = 1.	Bit 7	0
TIERW		Timer interrupt enable register W Controls timer W interrupt requests.	H'FF82	H'00
	OVIE	Timer overflow interrupt enable When OVIE = 1, enables the interrupt request by the OVF flag (FOVI).	Bit 7	1/0

Register Name		Function	Address	Setting
SYSCR1		System control register 1 Controls power-down mode.	H'FFF0	H'F0 (at initial setting)
	SSBY	Software standby When SSBY = 1, a transition to standby mode is made after a SLEEP instruction is executed in active mode.	Bit 7	1
	STS2	Standby timer select 2 to 0 When STS2 = 1 STS1 = 1, and STS 0 = 1, wait time of 16 clock cycles is set.	Bit 6	1
	STS1		Bit 5	1
	STS0		Bit 4	1
IENR1		Interrupt enable register 1	H'FFF4	H'01
	IEN0	When IEN0 = 0, disables $\overline{\text{IRQ0}}$ pin interrupt request. When IEN0 = 1, enables $\overline{\text{IRQ0}}$ pin interrupt request.	Bit 0	1/0
IRR1		Interrupt flag register 1	H'FFF6	H'00
	IRRI0	Cleared by writing 0 when IRRI0 = 1. Set to 1 when the $\overline{\text{IRQ0}}$ pin is set up to be used for interrupt input, and the edge specified for this pin is input.	Bit 0	0
PMR1		Port mode register 1 When PMR1 = H'10, P14 functions as $\overline{\text{IRQ0}}$ input pin.	H'FFE0	H'10
PDR1		Port data register 1 Port 1 general I/O port data register	H'FFD4	H'00
PCR1		Port control register 1 When PCR1 = H'00, the P17 to P 10 pins function as general input pins.	H'FFE4	H'00
PDR5		Port data register 5 Port 5 general I/O port data register	H'FFD8	H'00
PCR5		Port control register 5 When PCR5 = H'FF, the P57 to P50 pins function as general output pins.	H'FFE8	H'FF
PDR7		Port data register 7 Port 7 general I/O port data register When PDR7 = H'00, the LED is lit. When PDR7 = H'0C, the LED is not lit.	H'FFDA	H'0C/H'00
PCR7		Port control register 7 When PCR7 = H'0C, the P73 and P72 pins function as general output pins and other pins function as general input pins.	H'FFEA	H'0C

4.4 RAM Usage

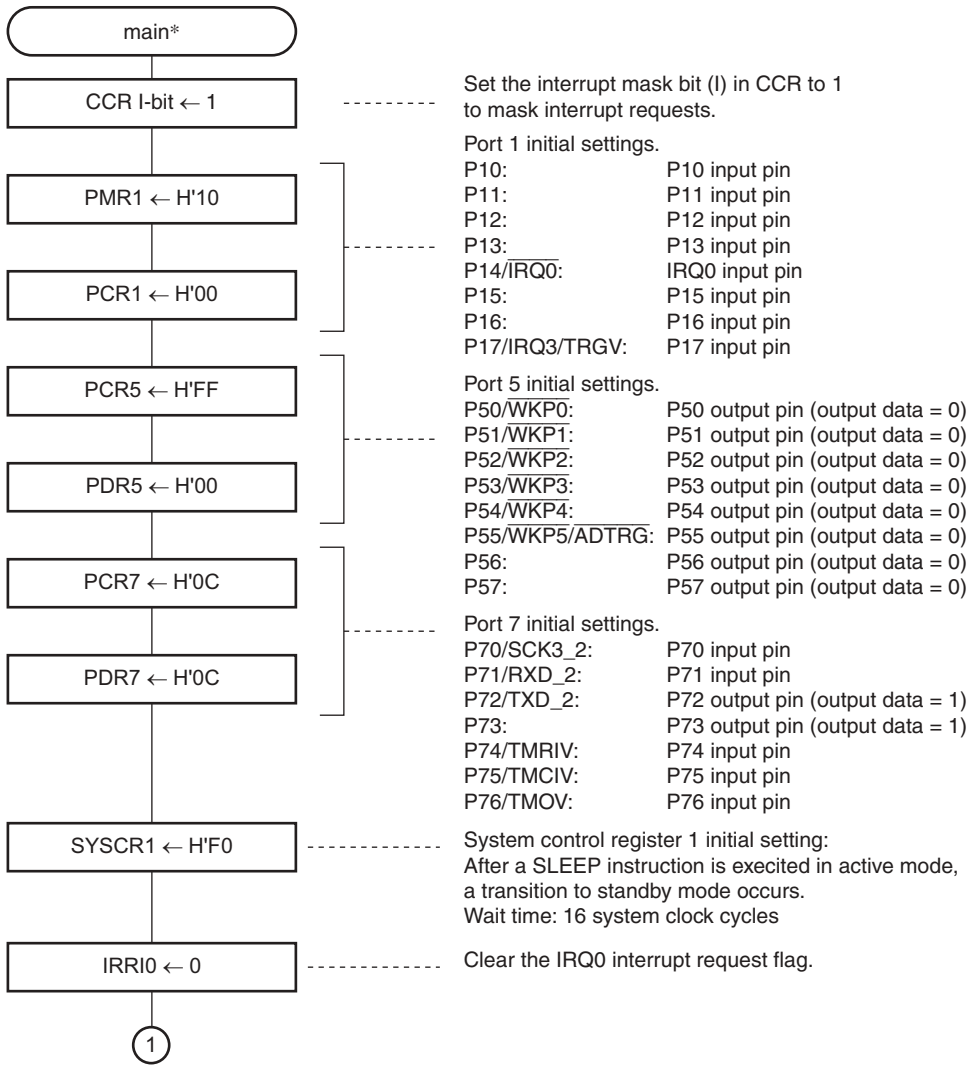
Table 5 describes the RAM usage in this sample task.

Table 5 Description of RAM

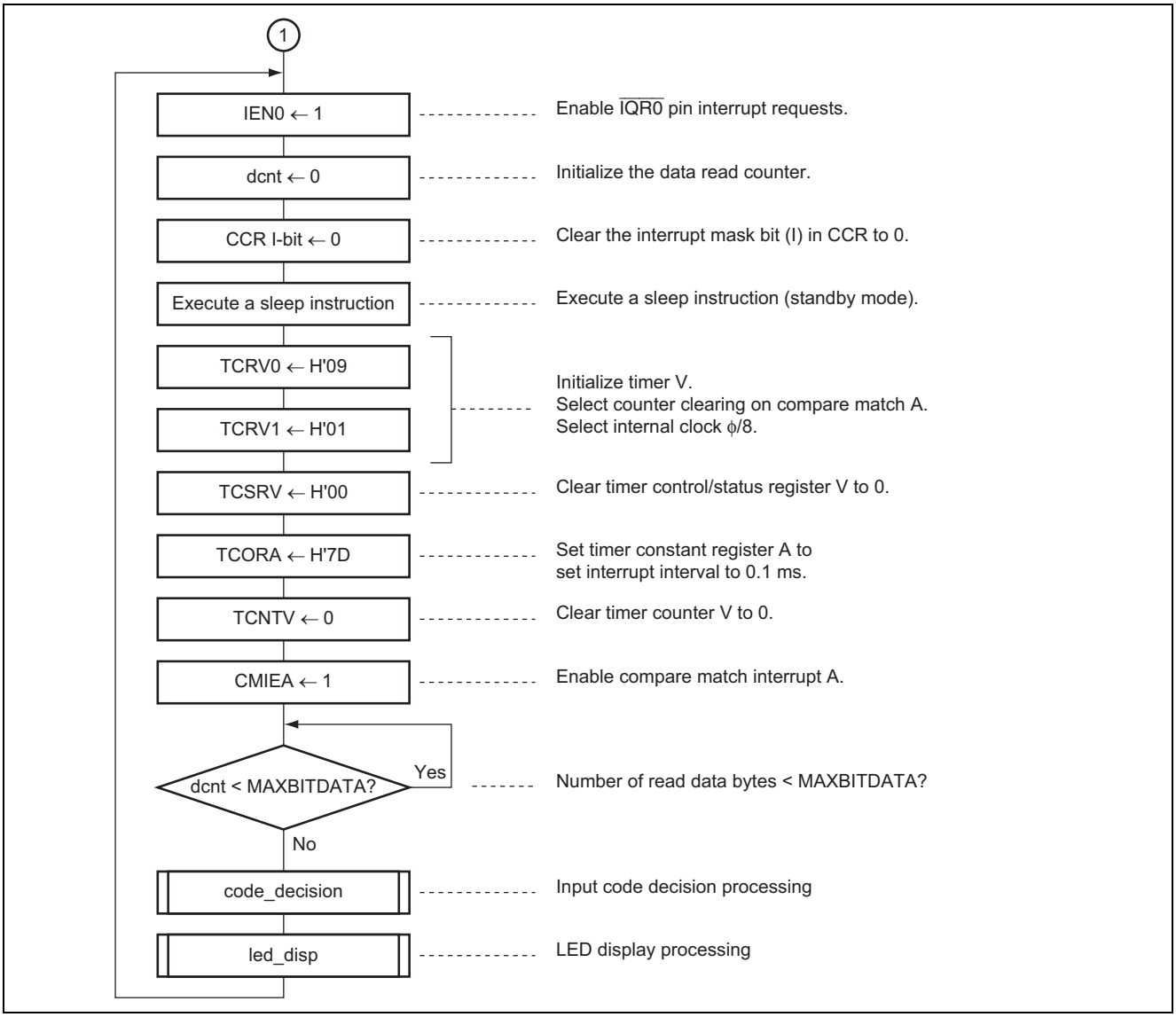
Label Name	Function	Address	Used in
dig_0	Stores LED1 display data (1 byte)	H'FB8A	led_disp,tmrw
dig_1	Stores LED2 display data (1 byte)	H'FB8B	led_disp,tmrw
cnt	8-bit counter used for switching the display on LED1 and LED2 (1 byte)	H'FB8C	tmrw
i	Stores loop count value (2 bytes)	H'FB80	code_decision
li	Stores loop count value (4 bytes)	H'FB82	delay
ptr	Pointer used for switching the display on LED1 and LED2 (2 bytes)	H'FB86	tmrw
dcnt	Bit data counter used in signal reception (2 bytes)	H'FB88	main, tmrv
data	Stores bit data in signal reception (700 bytes)	H'FB8D	code_decision, tmrv
leddata	Stores codes extracted from the bit data (100 bytes)	H'FE49	code_decision, led_disp
led_cnt	Stores leddata display digit (1 byte)	H'FEAD	led_disp
bit_cnt	Stores the position of the bit to be turned on/off (1 byte)	H'FEAE	code_decision
pulse_cnt	Counts high levels in one pulse (1 byte)	H'FEAF	code_decision
byte_cnt	Counts the leddata bytes (1 byte)	H'FEB0	code_decision, led_disp

5. Flowchart

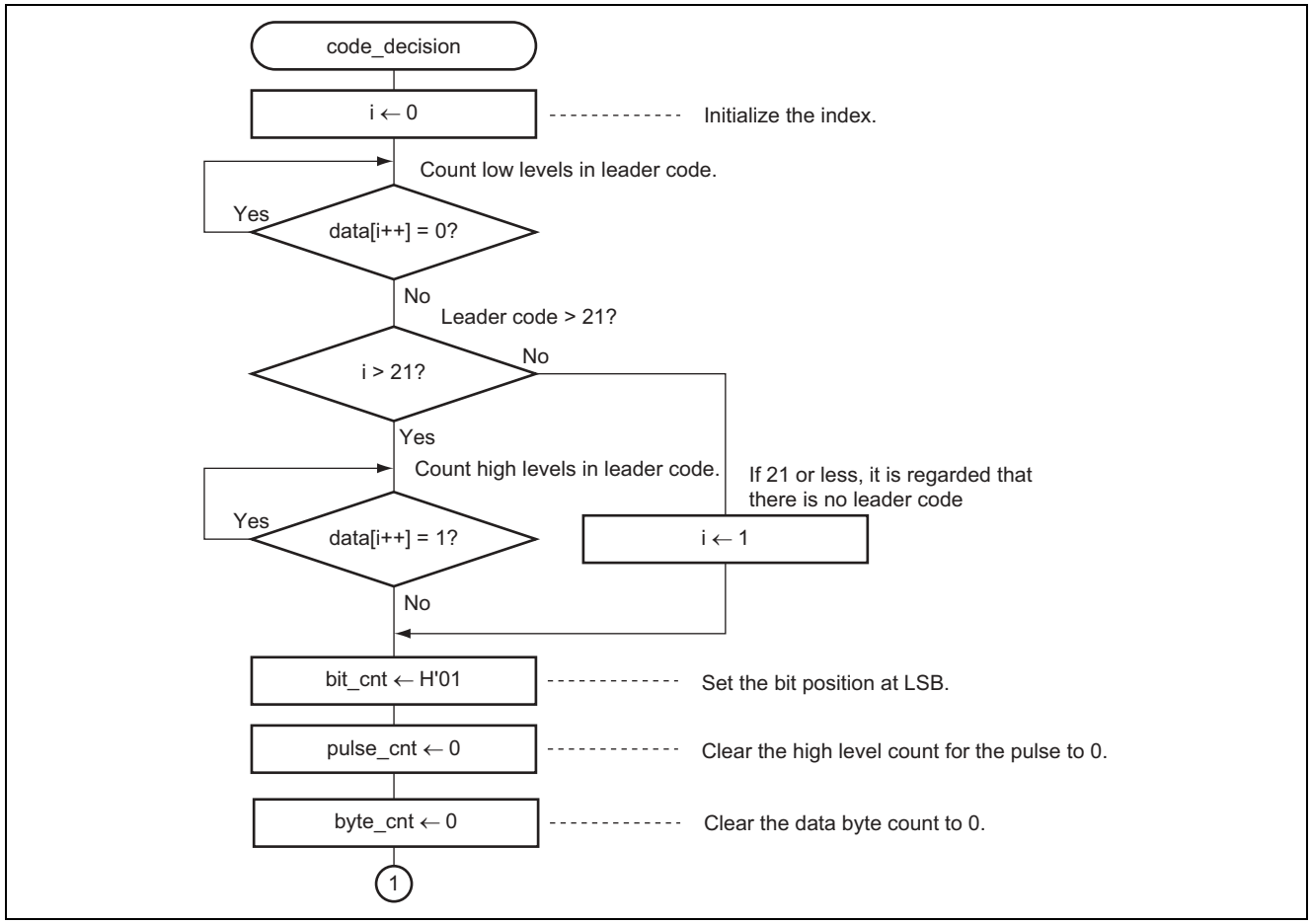
5.1 Main Routine (main)

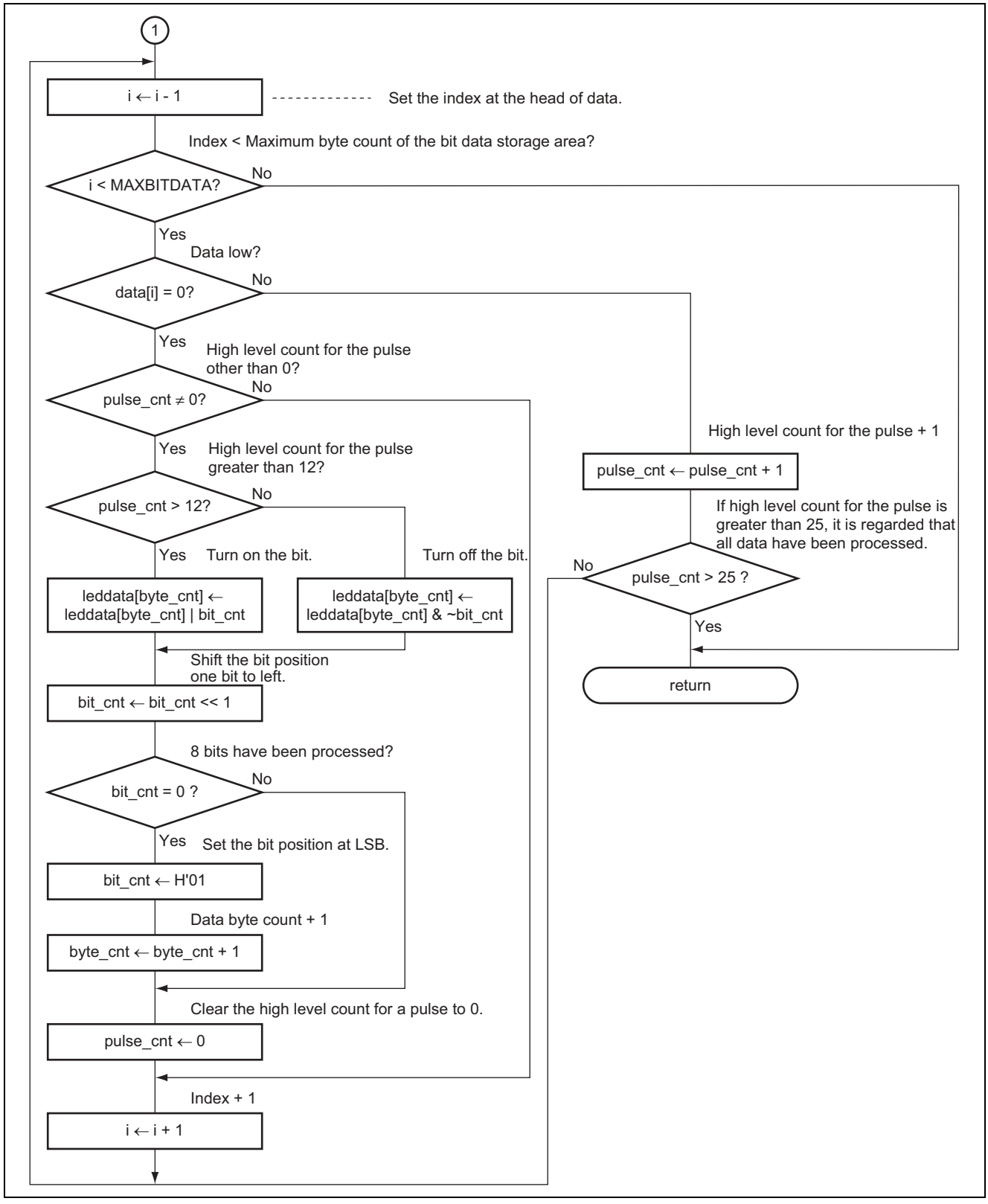


Note: * In this sample task, the stack pointer is set by INIT.SRC (assembly language).

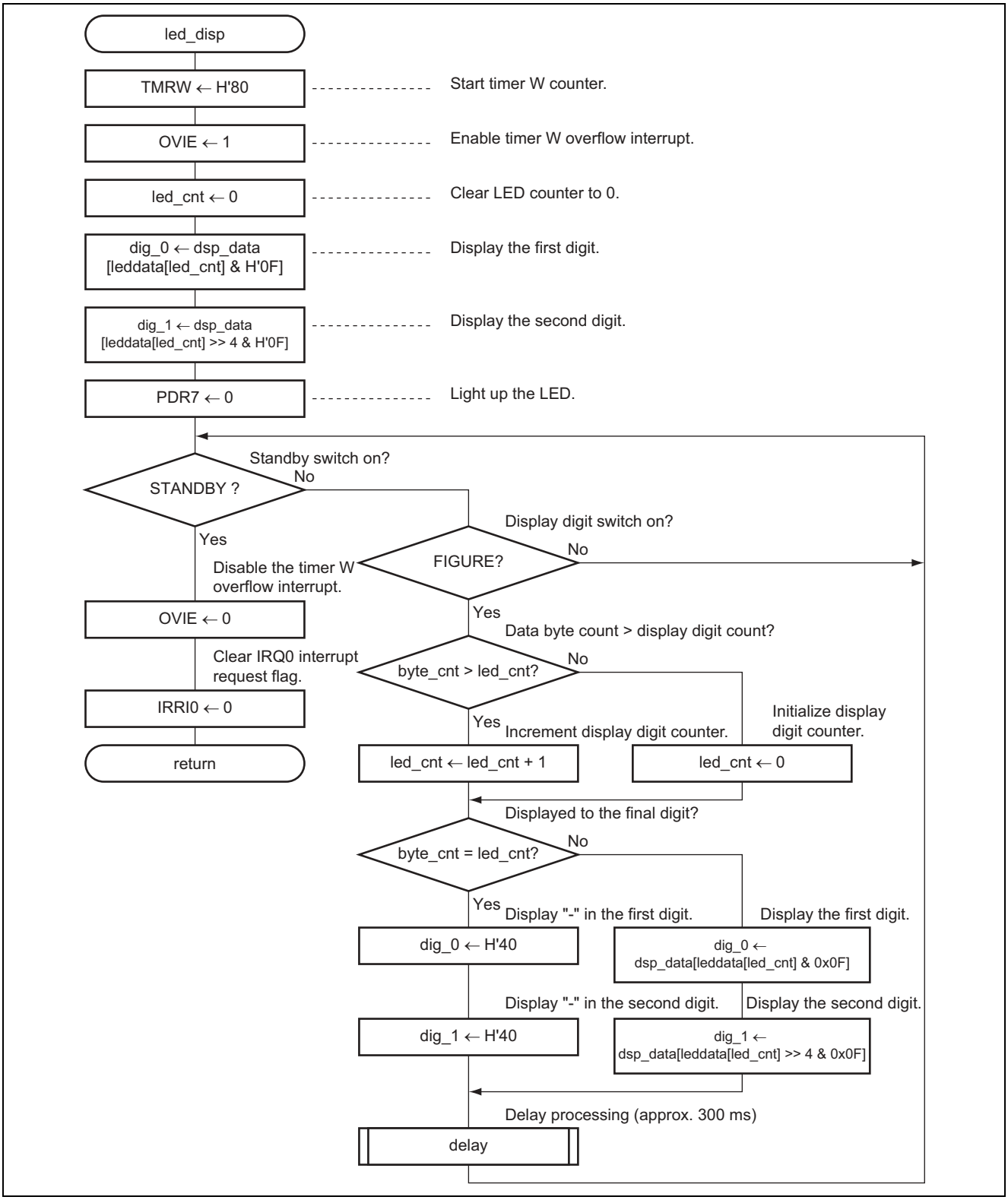

1
IEN0 ← 1
Enable IQR0 pin interrupt requests.
dcnt ← 0
Initialize the data read counter.
CCR I-bit ← 0
Clear the interrupt mask bit (I) in CCR to 0.
Execute a sleep instruction
Execute a sleep instruction (standby mode).
TCRV0 ← H'09
TCRV1 ← H'01
Initialize timer V.
Select counter clearing on compare match A.
Select internal clock φ/8.
TCSRV ← H'00
Clear timer control/status register V to 0.
TCORA ← H'7D
Set timer constant register A to
set interrupt interval to 0.1 ms.
TCNTV ← 0
Clear timer counter V to 0.
CMIEA ← 1
Enable compare match interrupt A.
dcnt < MAXBITDATA?
Yes
Number of read data bytes < MAXBITDATA?
No
code_decision
Input code decision processing
led_disp
LED display processing

5.2 Code Decision Processing (code_decision)

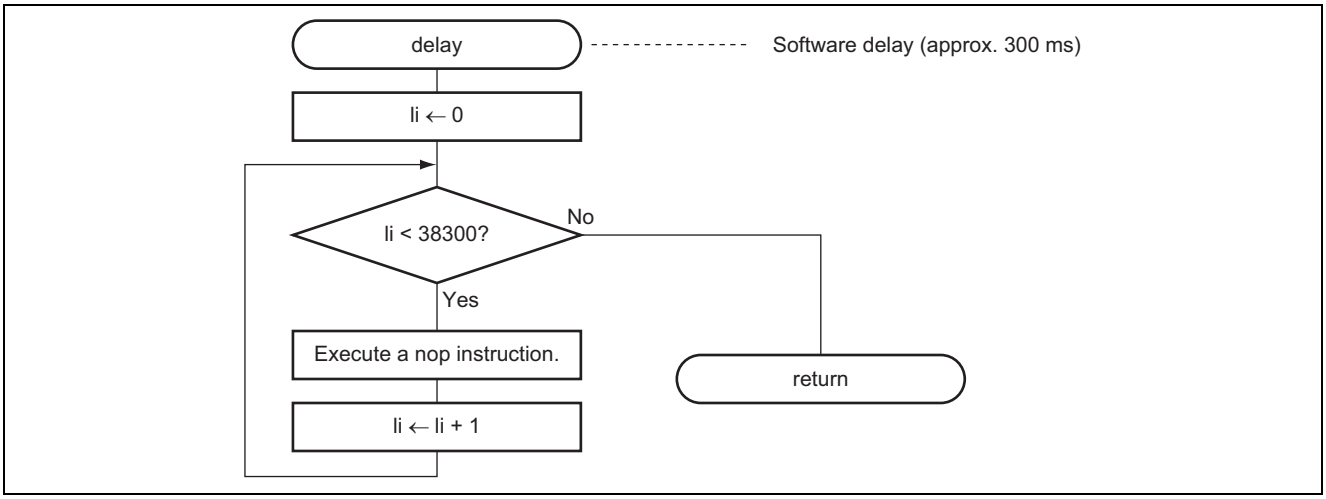



1
i ← i - 1
Set the index at the head of data.
Index < Maximum byte count of the bit data storage area?
i < MAXBITDATA?
No
Yes
Data low?
data[i] = 0?
No
Yes
High level count for the pulse other than 0?
pulse_cnt ≠ 0?
No
Yes
High level count for the pulse greater than 12?
pulse_cnt > 12?
No
Yes
Turn on the bit.
Turn off the bit.
leddata[byte_cnt] ← leddata[byte_cnt] | bit_cnt
leddata[byte_cnt] ← leddata[byte_cnt] & ~bit_cnt
Shift the bit position one bit to left.
bit_cnt ← bit_cnt << 1
8 bits have been processed?
bit_cnt = 0 ?
No
Yes
Set the bit position at LSB.
bit_cnt ← H'01
Data byte count + 1
byte_cnt ← byte_cnt + 1
Clear the high level count for a pulse to 0.
pulse_cnt ← 0
Index + 1
i ← i + 1
High level count for the pulse + 1
pulse_cnt ← pulse_cnt + 1
If high level count for the pulse is greater than 25, it is regarded that all data have been processed.
pulse_cnt > 25 ?
No
Yes
return

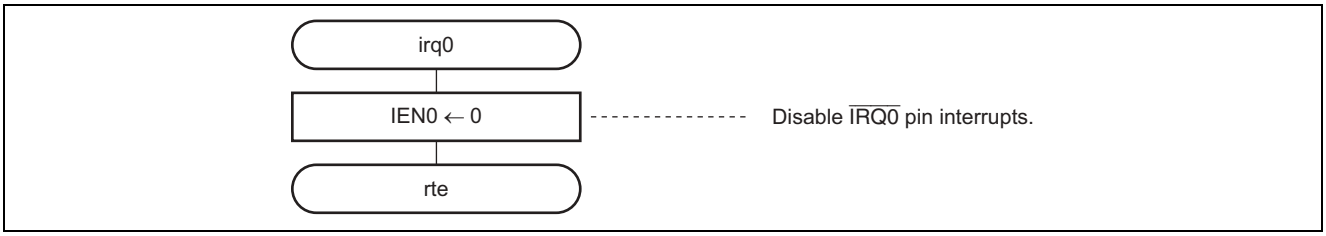
5.3 LED Display Processing (led_disp)



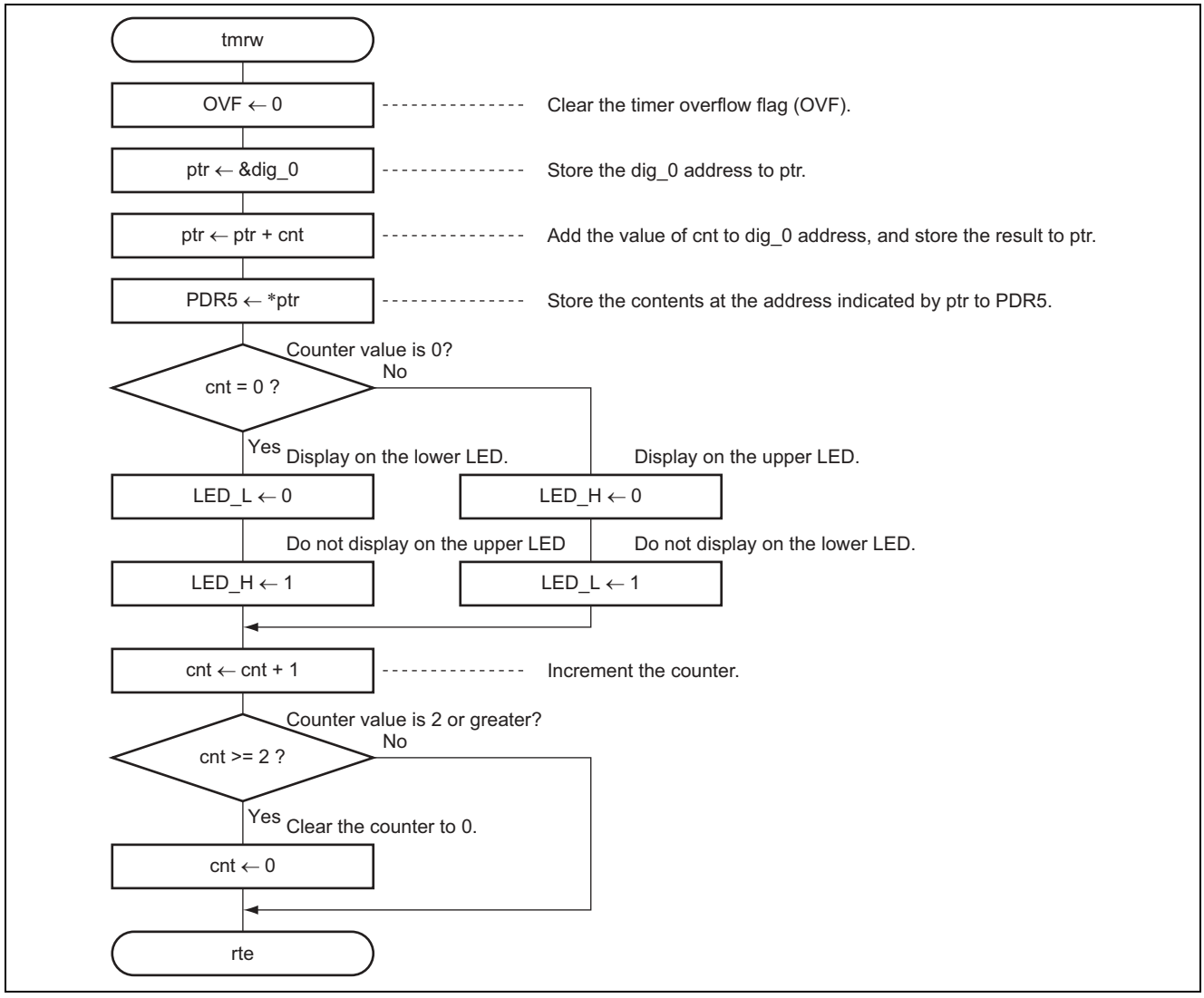
5.4 Software Display Processing (delay)



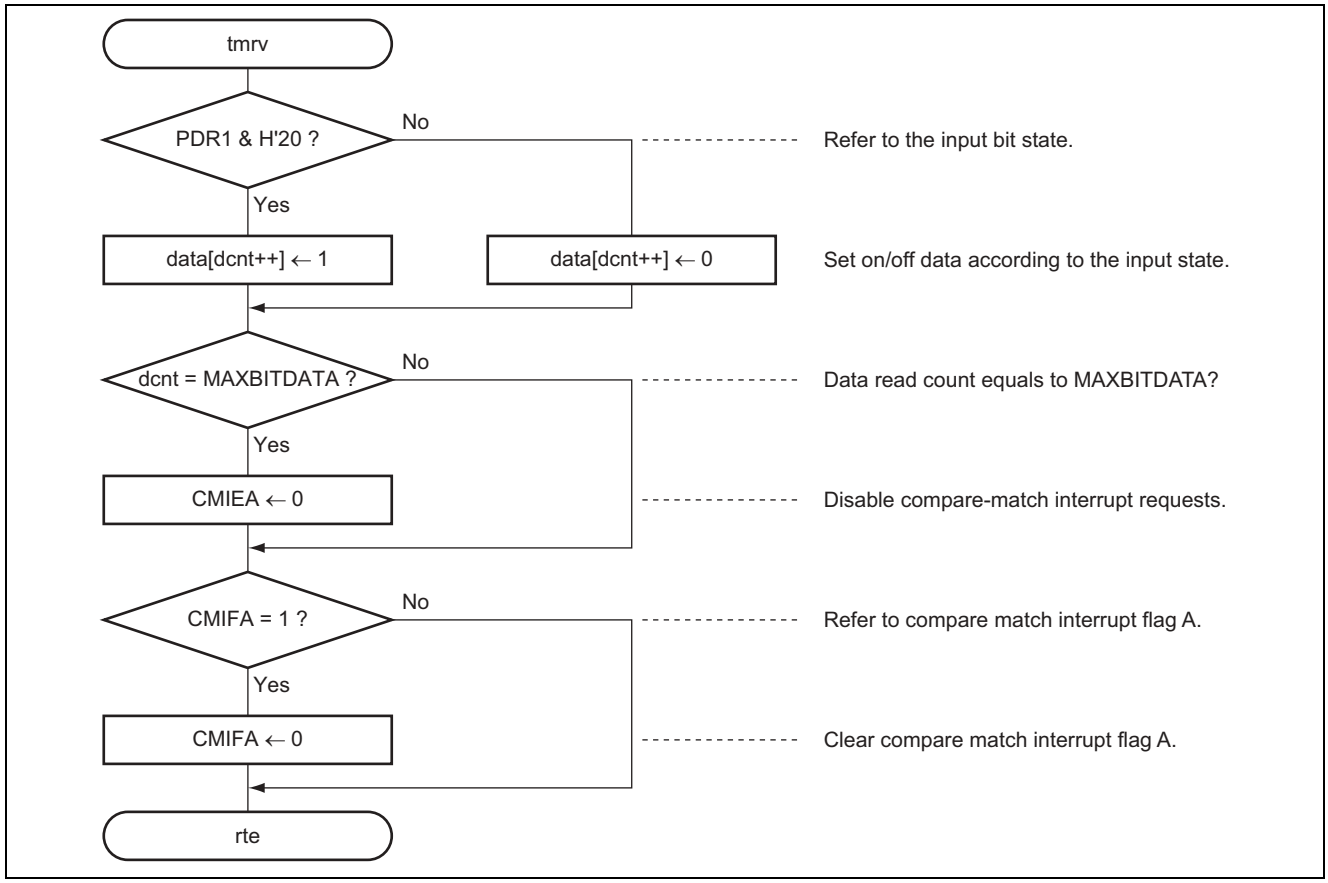
5.5 IRQ0 Interrupt Processing (irq0)



5.6 Timer W Interrupt Processing (tmrw)



5.7 Timer V Interrupt Processing (tmrv)



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
```

```
/*****************************************************************************************************/
/*                                                                                                   */
/* H8/300H Tiny Series -H8/36014- Application Note                                                   */
/*                                                                                                   */
/* Application                                                                                       */
/* Remote controller                                                                                 */
/*                                                                                                   */
/*****************************************************************************************************/

#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 PMR1       *(volatile unsigned char *)0xFFE0     /* Port mode register 1               */
#define PDR1       *(volatile unsigned char *)0xFFD4     /* Port data register 1               */
#define PCR1       *(volatile unsigned char *)0xFFE4     /* Port control register 1            */

#define PDR2       *(volatile unsigned char *)0xFFD5     /* Port data register 2               */
#define PCR2       *(volatile unsigned char *)0xFFE5     /* Port control register 2            */

#define PMR5       *(volatile unsigned char *)0xFFE1     /* Port mode register 5               */
#define PUCR5      *(volatile unsigned char *)0xFFD1     /* 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 PDR7       *(volatile unsigned char *)0xFFDA     /* Port data register 7               */
#define PCR7       *(volatile unsigned char *)0xFFEA     /* Port control register 7            */
#define PDR7_BIT   (*(struct BIT *)0xFFDA)
#define STANDBY    PDR7_BIT.b0                           /* Standby switch                     */
#define FIGURE     PDR7_BIT.b1                           /* LED figure switch                  */
#define LED_H      PDR7_BIT.b2                           /* LED(HIGH) ON                       */
#define LED_L      PDR7_BIT.b3                           /* LED(LOW) ON                        */
```

```
#define TMRW        *(volatile unsigned char *)0xFF80     /* Timer mode register W              */
#define TIERW       *(volatile unsigned char *)0xFF82     /* Timer interrupt enable register W  */
#define TIERW_BIT   (*(struct BIT *)0xFF82)               /* Timer interrupt enable register W  */
#define OVIE        TIERW_BIT.b7                          /* Overflow interrupt enable          */
#define TSRW        *(volatile unsigned char *)0xFF83     /* Timer status register W            */
#define TSRW_BIT    (*(struct BIT *)0xFF83)               /* Timer status register W            */
#define OVF         TSRW_BIT.b7                           /* Timer overflow                     */

#define TCRV0       *(volatile unsigned char *)0xFFA0     /* Timer control register V0          */
#define TCRV0_BIT   (*(struct BIT *)0xFFA0)               /* Timer control register V0          */
#define CMIEA       TCRV0_BIT.b6                          /* Compare match interrupt enable A   */
#define TCSRV       *(volatile unsigned char *)0xFFA1     /* Timer control/status register V    */
#define TCSRV_BIT   (*(struct BIT *)0xFFA1)               /* Timer control/status register V    */
#define CMIFA       TCSRV_BIT.b6                          /* Compare match flag A               */
#define TCORA       *(volatile unsigned char *)0xFFA2     /* Time constant register A           */
#define TCRV1       *(volatile unsigned char *)0xFFA5     /* Timer control register V1          */
#define TCNTV       *(volatile unsigned char *)0xFFA4     /* Timer counter V                    */

#define SYSCR1      *(volatile unsigned char *)0xFFF0     /* System control register 1          */
#define IENR1       *(volatile unsigned char *)0xFFF4     /* Interrupt enable register 1        */
#define IENR1_BIT   (*(struct BIT *)0xFFF4)
#define IEN0        IENR1_BIT.b0                          /* IRQ0 interrupt request enable      */

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

#pragma interrupt (irq0)
#pragma interrupt (tmrw)
#pragma interrupt (tmrv)

/* 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 tmrv(void);                                          /* Timer V interrupt routine          */
void tmrw(void);                                          /* Timer W 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                       */
int li;                                                 /* 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'001c IRQ0 vector                 */
};
#pragma section V3                                      /* Vector section set                 */
void (*const VEC_TBL3[])(void) = {
    tmrw                                                /* H'002a Timer W interrupt vector    */
};
#pragma section V4                                      /* Vector section set                 */
void (*const VEC_TBL4[])(void) = {
    tmrv                                                /* H'002c Timer V interrupt vector    */
};
#pragma section                                         /* P                                  */

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

PMR1 = 0x10;                                            /* Use IRQ0                           */
PCR1 = 0x00;                                            /* Input Infrared Rays                */
PCR5 = 0xFF;                                            /* Initialize: output LED             */
PDR5 = 0x00;                                            /* Clear LED                          */
PCR7 = 0x0C;                                            /* Initialize: input SW & LED control */
PDR7 = 0x0C;                                            /* LED OFF                            */

SYSCR1 = 0xF0;                                          /* Standby mode, use external clock   */

IRRI0 = 0;                                              /* Clear IRQ0 interrupt request flag  */

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

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

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

```
        TCRV0 = 0x09;                                   /* Initialize Timer V                */
        TCRV1 = 0x01;                                   /* Select internal clock             */
        TCSRV = 0x00;                                   /* Clear flag, output: don't change  */
        TCORA = 0x7D;                                   /* Set interrupt interval to 0.1msec */
        TCNTV = 0;                                      /* Clear timer counter V             */
        CMIEA = 1;                                      /* Enable compare match Interrupt    */
        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)
{
TMRW = 0x80;                                          /* Start timer W                       */
OVIE = 1;                                             /* Enable timer W OVF interrupt        */

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          */
PDR7 = 0x00;                                          /* LED ON                              */
    while(1){
        if(STANDBY){                                  /* Standby switch ON ?                 */
        OVIE = 0;                                     /* Disable timer W OVF interrupt       */
        IRRI0 = 0;                                    /* clear IRQ0 interrupt request flag   */
        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)
{
for(li = 0;li < 38300;li++)
nop();
}

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

```
/*****************************************************************************************************/
/*  Timer W Interrupt (in order to light LED in turn)                                                */
/*****************************************************************************************************/
void tmrw(void)
{
    OVF     = 0;                                        /* Clear OVF 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 V Interrupt(every 0.1msec)                                                                 */
/*****************************************************************************************************/
void tmrv(void)
{
    if(PDR1 & 0x20){
        data[dcnt++] = 1;                               /* Bit ON                             */
    }else{
        data[dcnt++] = 0;                               /* Bit OFF                            */
    }
    if(dcnt == MAXBITDATA){                             /* End ?                              */
        CMIEA = 0;                                      /* Disable compare match Interrupt    */
    }
    if ( CMIFA == 1 ) {
        CMIFA = 0;                                      /* Clear Compare match flag A         */
    }
}
```

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Jul.28.04	—	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.