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.

APPLICATION NOTE

H8/300H Tiny Series

Sending and Receiving Characters Using an Infrared Transceiver

Introduction

This document describes how to send and receive characters over the physical layer between an infrared transceiver connected to the H8/36014 and the IrDA communication port of a personal computer.

Target Device

H8/300H Tiny Series H8/36014 CPU

Contents

1. Specifications

- Figure 1 shows the hardware configuration for sending and receiving data using the infrared transceiver connected to the H8/36014.
- The infrared transceiver receives the characters sent from an IrDA communication port of a personal computer, the H8/36014 increments the ASCII codes of the received characters, and returns the new codes to the personal computer. The display of the personal computer shows the characters corresponding to the incremented ASCII codes.
- The operating voltage (Vcc) and the analog power supply voltage (AVcc) of the H8/36014 are 3.3 V and the oscillator clock frequency is 10 MHz.

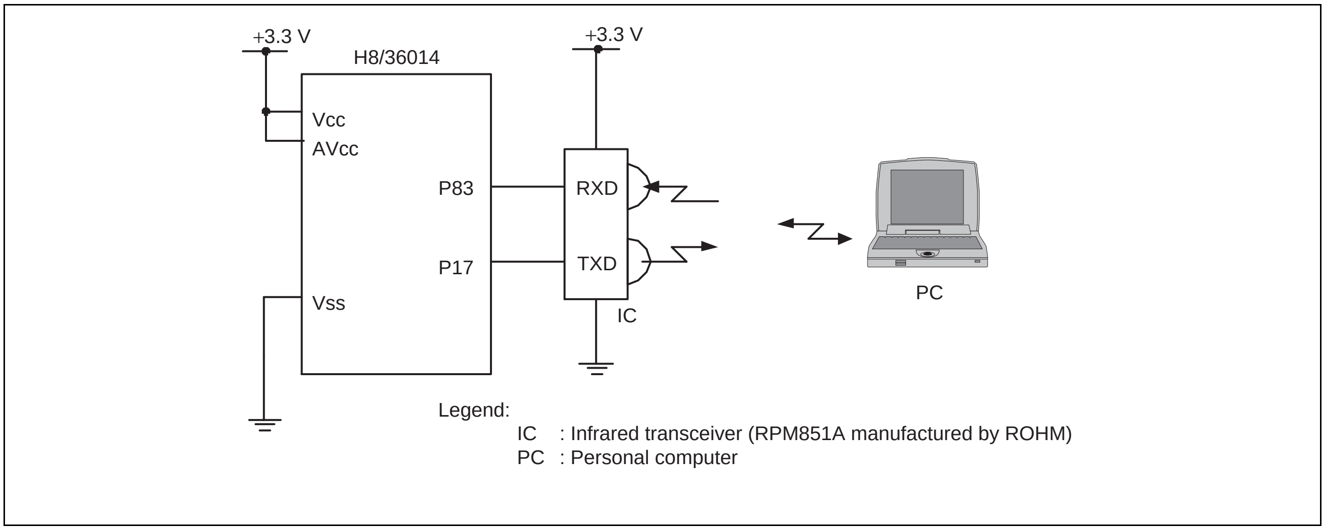


Figure 1 Hardware Configuration

- The infrared transceiver manufactured by ROHM (model: RPM851A) is used.

Figure 2 shows the operation timing chart for the infrared transceiver.

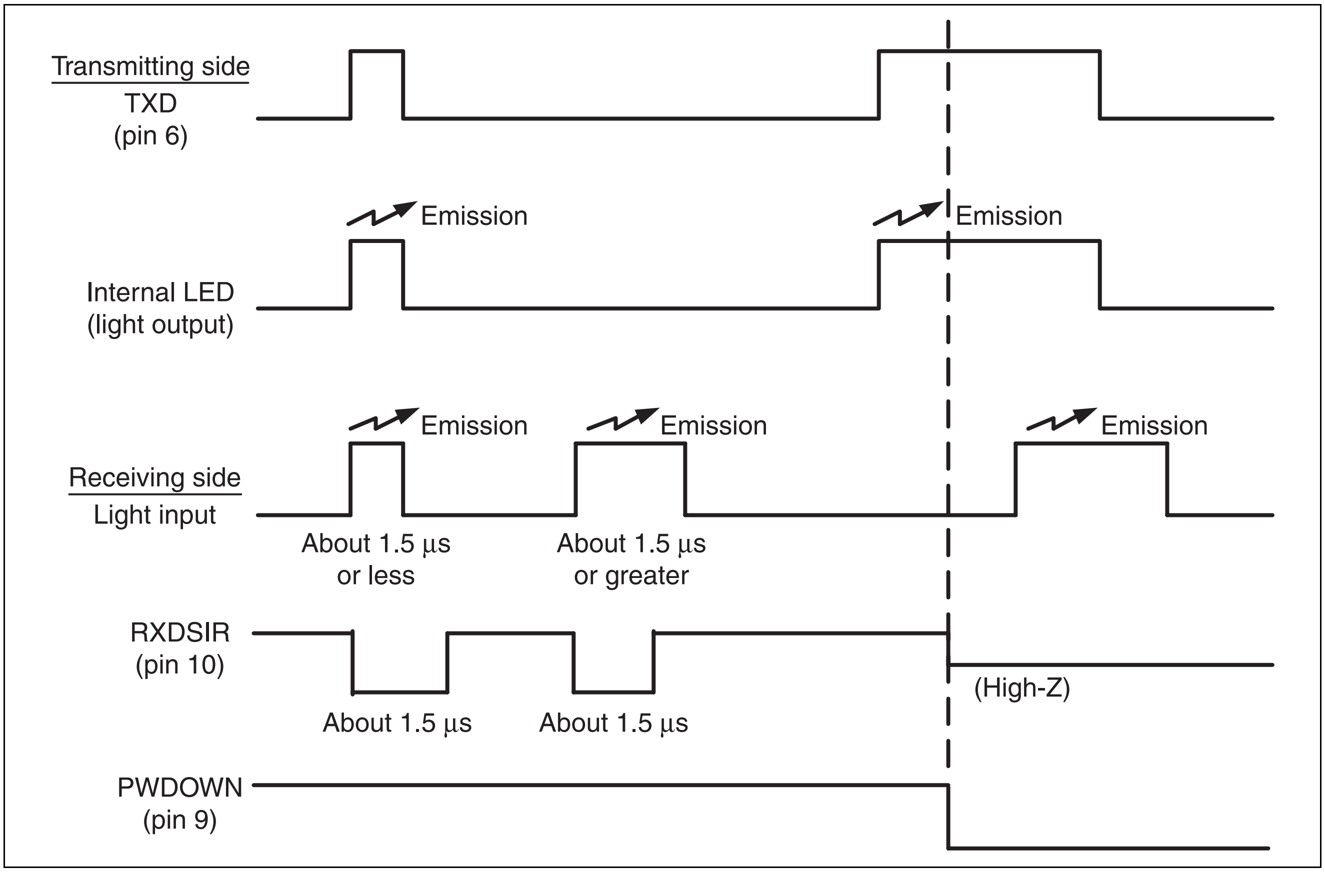


Figure 2 Timing Chart of an Infrared Transceiver Operation

The features of the RPM851A are as follows.

- Conforms to IrDA Version 1.0.
- Consumes small current during standby (normally 220 µA).
- Employs the power-down control function suitable for applications driven by a battery.
- Power-supply voltage ranges from 2.7 V to 5.5 V.
- Comes in surface-mount packages suitable for installation on both the top and side.

Figure 3 shows the block diagram and the application circuit of the infrared transceiver.

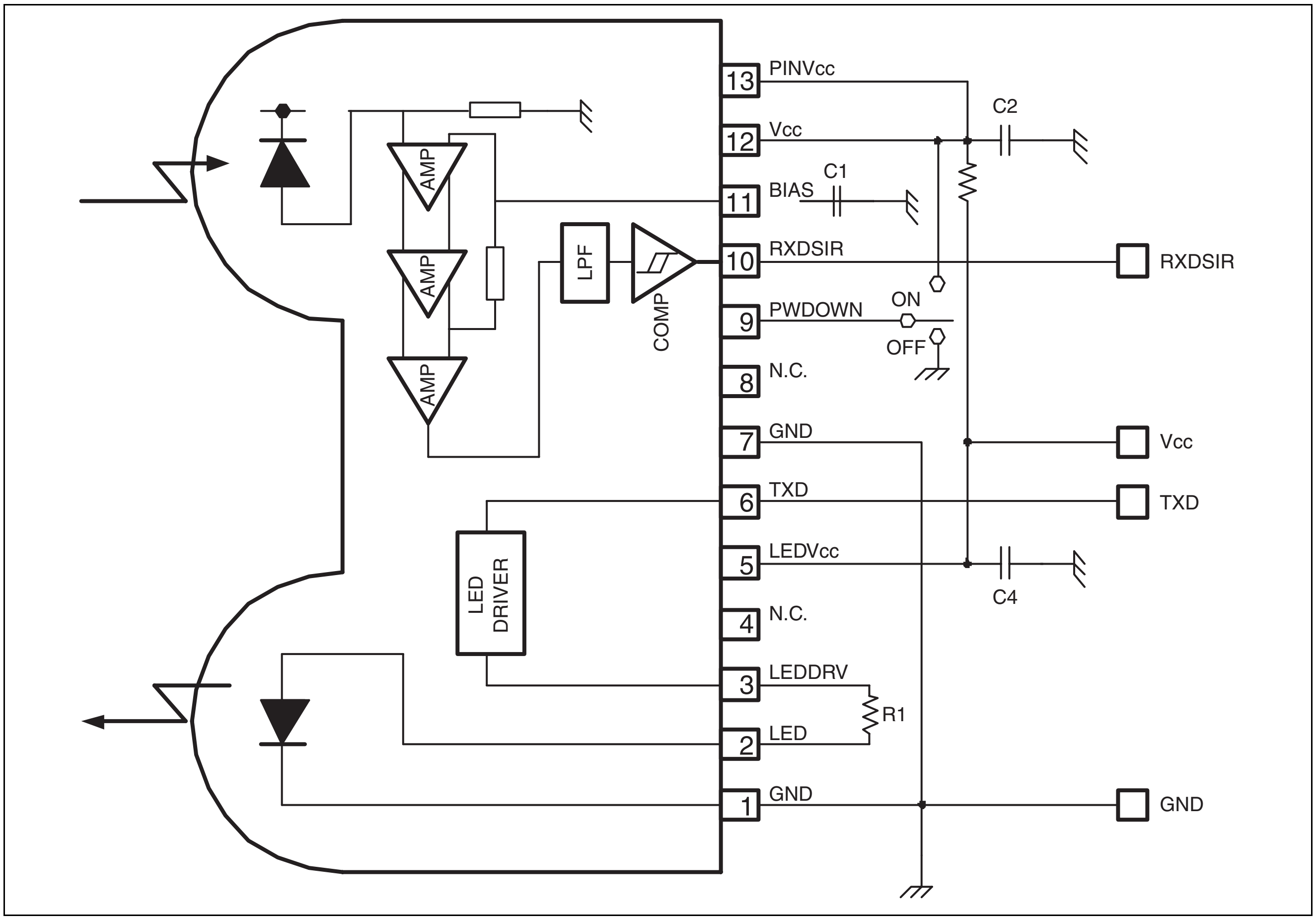


Figure 3 Block Diagram and Application Circuit of the Infrared Transceiver

The procedure for sending and receiving data using the infrared transceiver is as follows.

1. The application program in the personal computer is used to communicate over the physical layer using the waveform conforming to IrDA version 1.0.
2. Enter a character, i.e. "1" which corresponds to 31H of the ASCII code, from the keyboard of the personal computer.
3. The modulated signals are sent from the IrDA communication port of the personal computer starting from the LSB.
4. The infrared transceiver connected to the H8/36014 receives the signals, demodulates them, and obtains 31H.
5. The H8/36014 increments the received signals to 32H, modulates them, and immediately sends them back to the personal computer via the infrared transceiver.
6. The personal computer receives the signals via the IrDA communication port, demodulates them, and obtains 32H. Since ASCII code 32H is equivalent to decimal 2, the display shows "2".

2. Description of Functions

The block diagram of the functions in the H8/36014 used is given in Figure 4.

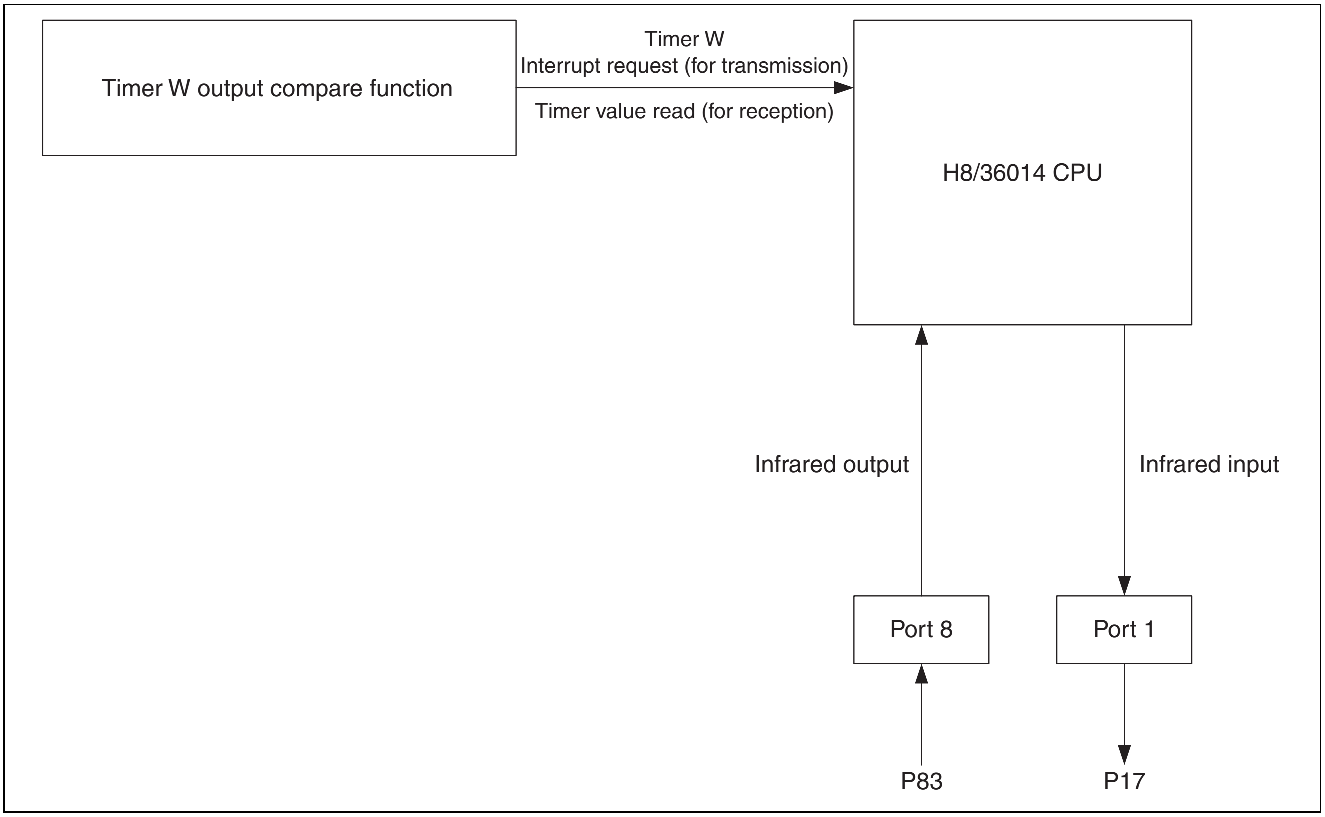


Figure 4 Diagram of the Used Functional Blocks

Table 1 shows the details about each function.

Table 1 Details of the Functions

Function	Description
Timer W	Employs the compare match function to output a toggle signal. Set a desired value in general register B (GRB) to change the output frequency.
Port 1	The P17 output pin of port 1 is used to send infrared signals to the personal computer.
Port 8	The P83 input pin of port 8 is used to receive infrared signals from the personal computer.

3. Description of Operation

Figure 5 describes the operation of infrared communications using timer W. As shown in Figure 5, the H8/36014 outputs a toggle signal in every compare match cycle of timer W and sends infrared signals to the personal computer.

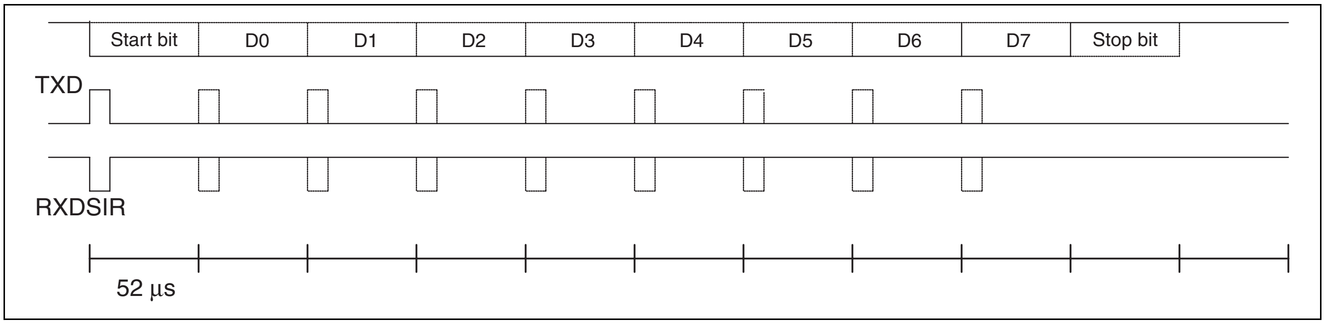


Figure 5 Operation of Infrared Communications Using Timer W

In this task, signals are transmitted at 19.2 kHz. Therefore, one bit is equal to about 52 µs.

On transmission: Each bit is cleared to 0 for about 4.5 µs. The negative logic (0: High, 1: Low) is used.

On reception: Note that signals change for 1.5 µs due to the restrictions of the RPM851A. The positive logic (1: High, 0: Low) is used.

The H8/36014 references the received bits 20 times each time it receives data.

4. Description of Software

4.1 Modules

Table 2 describes the modules used.

Table 2 Description about the Modules

Module	Label	Description
Main routine	main	Defines the initial settings, and calls the infrared signal reception routine and the infrared signal transmission routine alternatively.
Infrared signal reception routine	irda_rcv	Receives signals using infrared.
Infrared signal transmission routine	irda_snd	Sends signals using infrared.
Timer W interrupt routine	tmrw	Used as a timer that expires in 52 μs.

4.2 Parameters

No arguments are used in this task.

4.3 Registers

Table 3 is a list of registers of the H8/36014 used in this task.

Table 3 Registers

Register		Description	Address	Set value
TMRW		Timer mode register W Selects the function of general registers and the output mode of the timer.	H'FF80	H'80
	CTS	Counter start bit When CTS = 1, TCNT starts counting. When CTS = 0, TCNT stops counting.	Bit 7	1
TCRW		Timer control register W Selects the clock signal for the count register. Sets the condition to clear the count register and selects the output level of the timer.	H'FF81	H'82
	CCLR	Count clear bit When CCLR = 1, TCNT is cleared by compare match flag A.	Bit 7	1
	CKS2	Clock signal selection bits When CKS2 = 0, CKS1 = 0 and CKS0 = 0, system clock signals are used as the input clock signals for TCNT without dividing them.	Bit 6	0
	CKS1		Bit 5	0
	CKS0		Bit 4	0
TIERW		Timer interrupt enable register W Controls the interrupt requests to timer W.	H'FF82	H'00 (initial setting)
	IMIEA	Input capture/compare match interrupt enable A When IMIEA = 1 and IMFA of TSRW is set, interrupt request signal IMIA is enabled.	Bit 0	1
TSRW		Indicates the interrupt request status.	H'FF83	H'00
	IMFA	Input capture/compare match flag A	Bit 0	0
TCNT		Count register 16-bit up counter using every eighth system clock cycle for input clock signals.	H'FF86	H'00
GRA		General-purpose register A	H'FF88	H'FF
PCR1		Port control register 1 Selects the I/O of the pins used as general I/O pins of port 1 bit by bit. When PCR1 = H'A0, pins 17 and 15 function as general output pins. Otherwise, pins 17 and 15 are used as general input pins.	H'FFE4	H'A0
PDR1		Port data register 1 General-purpose I/O port data register for port 1	H'FFD4	H'00
PCR8		Port control register 7 Selects the I/O of the pins used as general I/O pins of port 8 bit by bit. When PCR8 = H'F7, pin 83 functions as a general input pin. Otherwise, pin 83 functions as a general output pin.	H'FFEB	H'F7
PDR8		Port data register 8 General-purpose I/O port data register for port 8	H'FFDB	H'00 (initial setting)

4.4 RAM

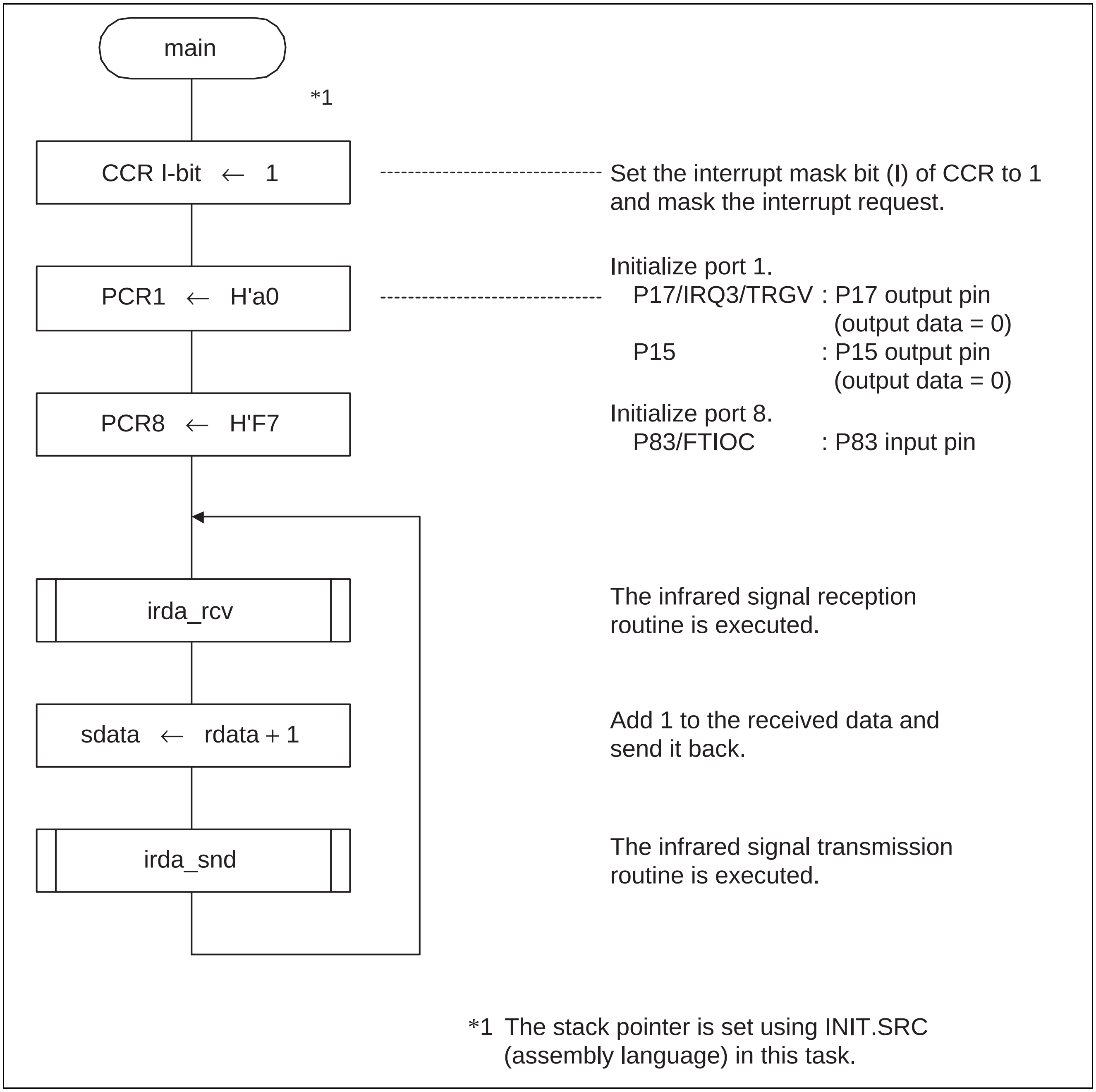
Table 4 shows how the on-chip RAM is used .

Table 4 Description about RAM

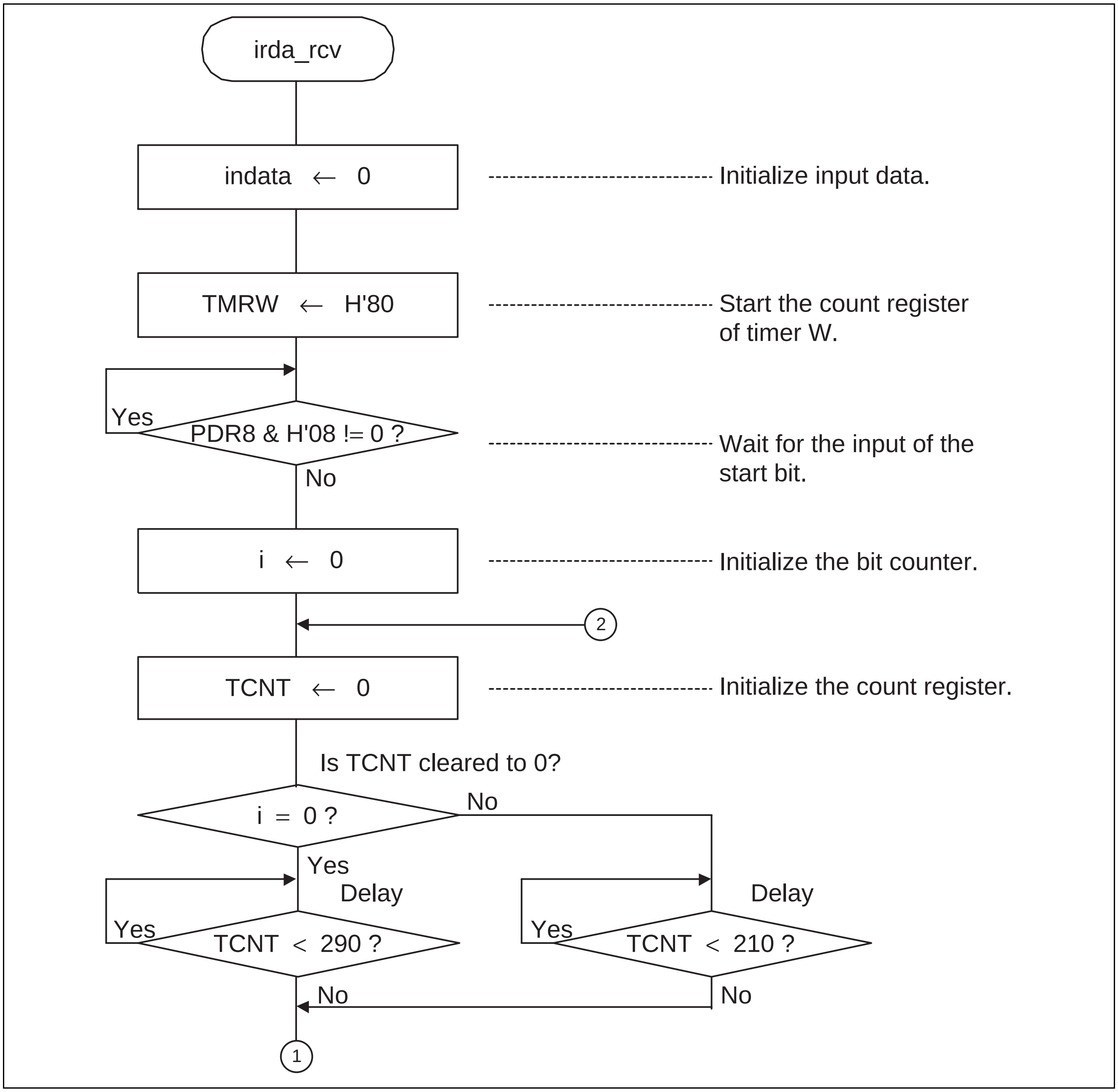
Label	Description	Address	Used by:
sdata	Stores the data to be sent (one byte).	H'FB86	main, tmrw
rdata	Stores the received data (one byte).	H'FB87	main, tmrw
bitdata	Stores the 10-bit data to be sent (ten bytes).	H'FB88	main, tmrw
i	Stores a loop counter (two bytes).	H'FB80	input_key
j	Stores a loop counter (two bytes).	H'FB82	input_key
bit	Stores bit data at reception (one byte).	H'FB92	main
indata	Stores input data (one byte).	H'FB93	input_key
Wtimeup	Flag for indicating the timer W expiration (two bytes)	H'FB84	input_key
bitpos	Used to determine the on and off states of bits (one byte).	H'FB94	input_key
dummy	Stores a dummy byte (one byte).	H'FB95	

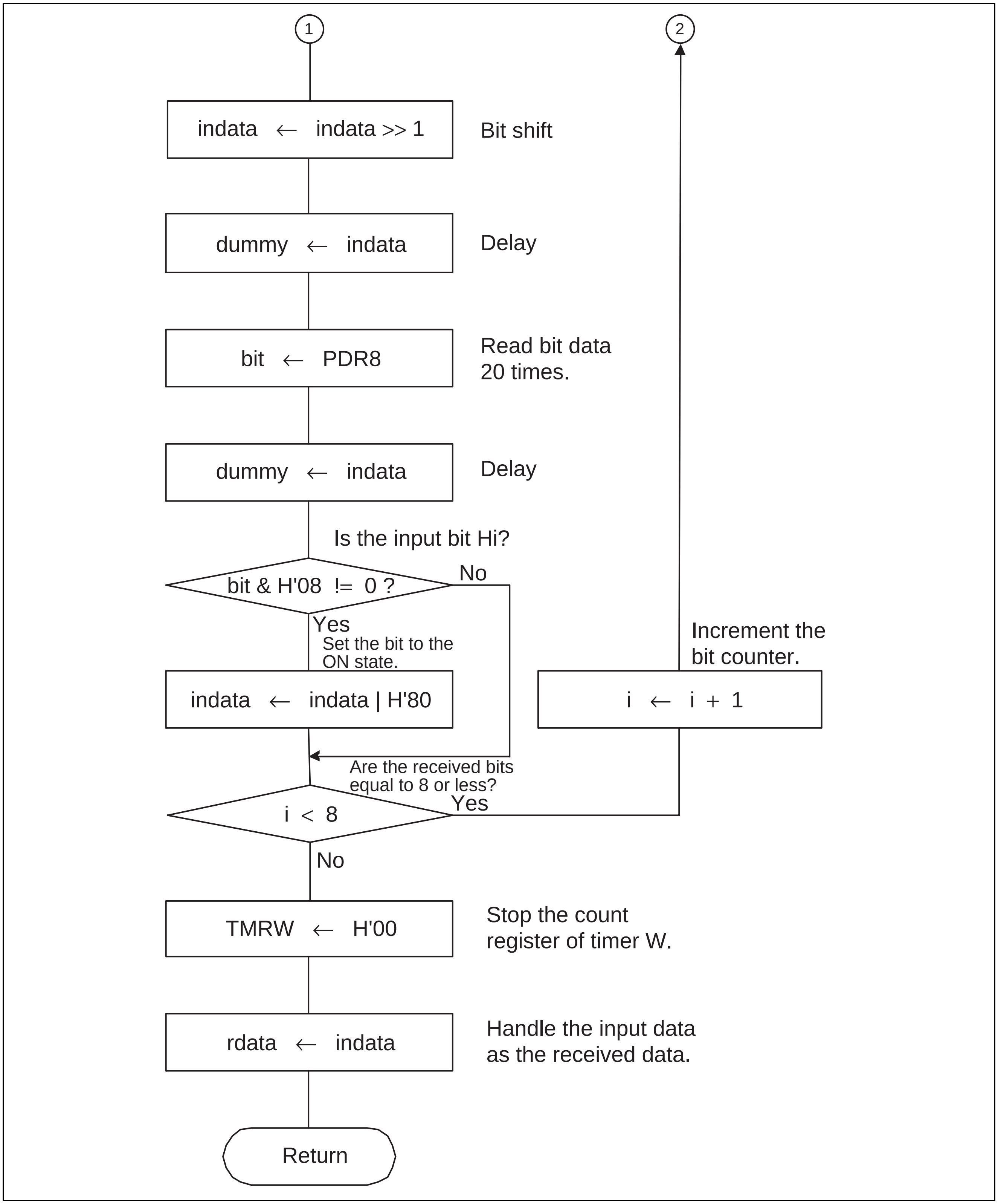
5. Flowchart

5.1 Main Routine (main)

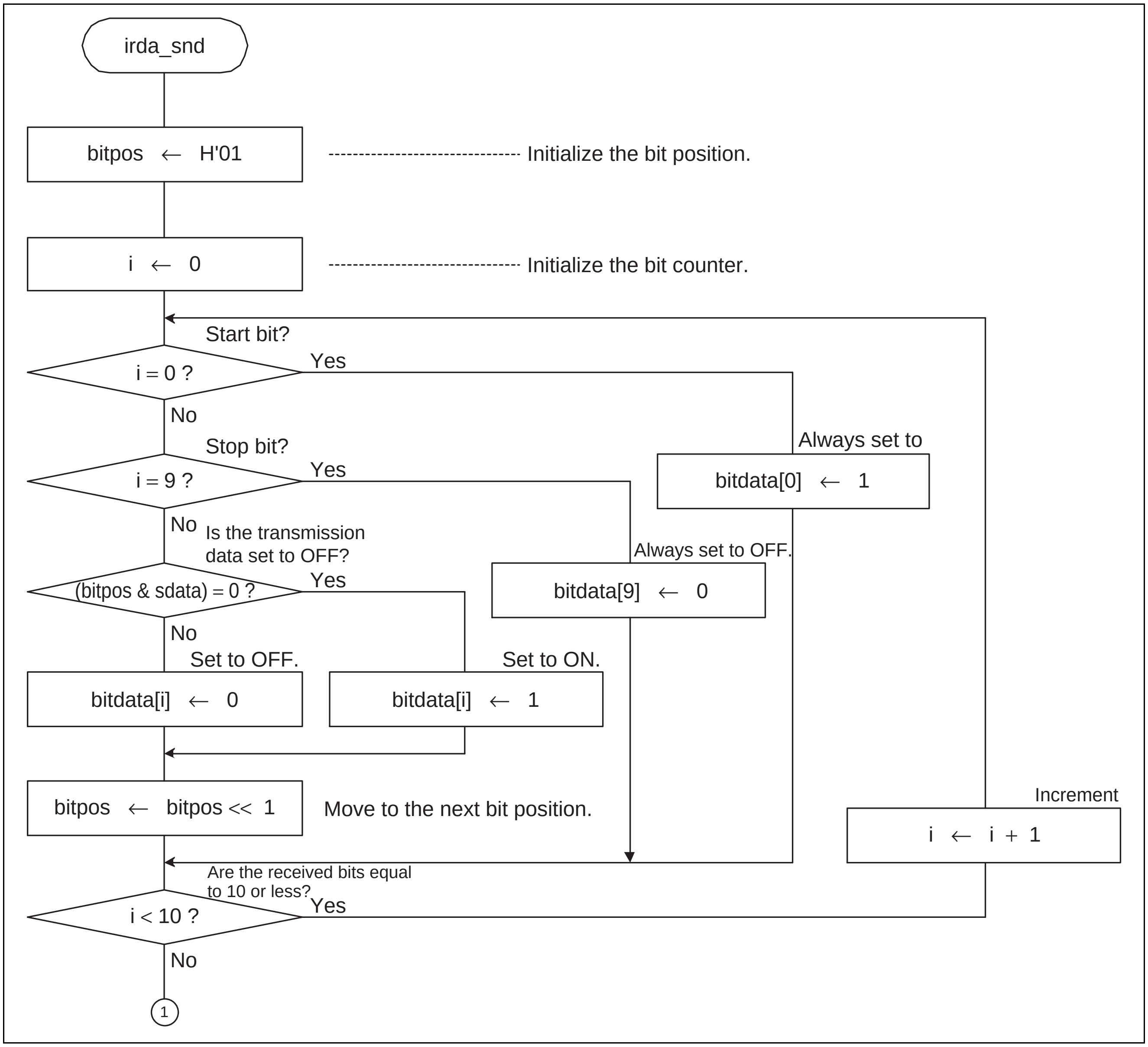


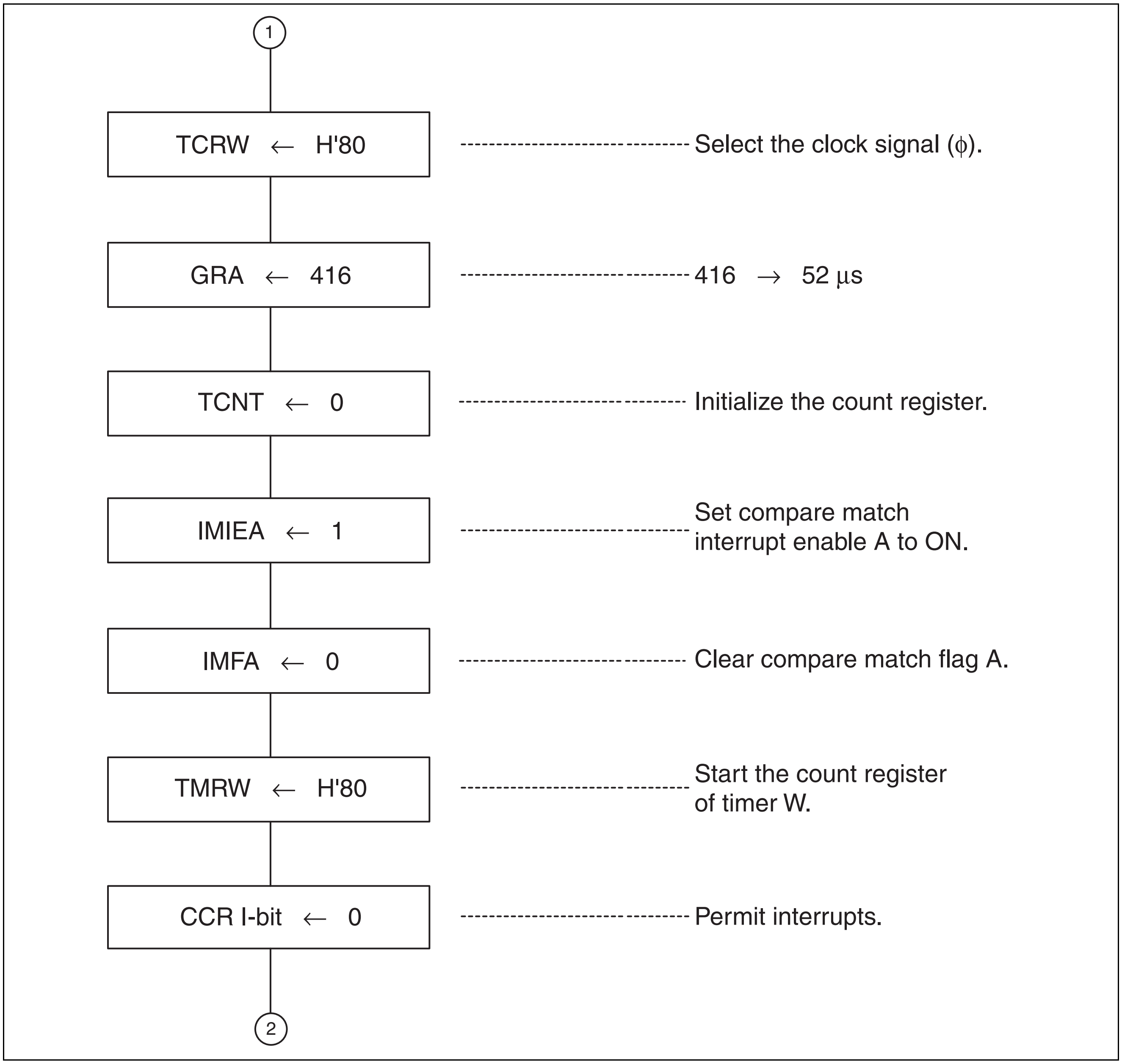
5.2 Infrared Signal Reception Routine (irda_rcv)

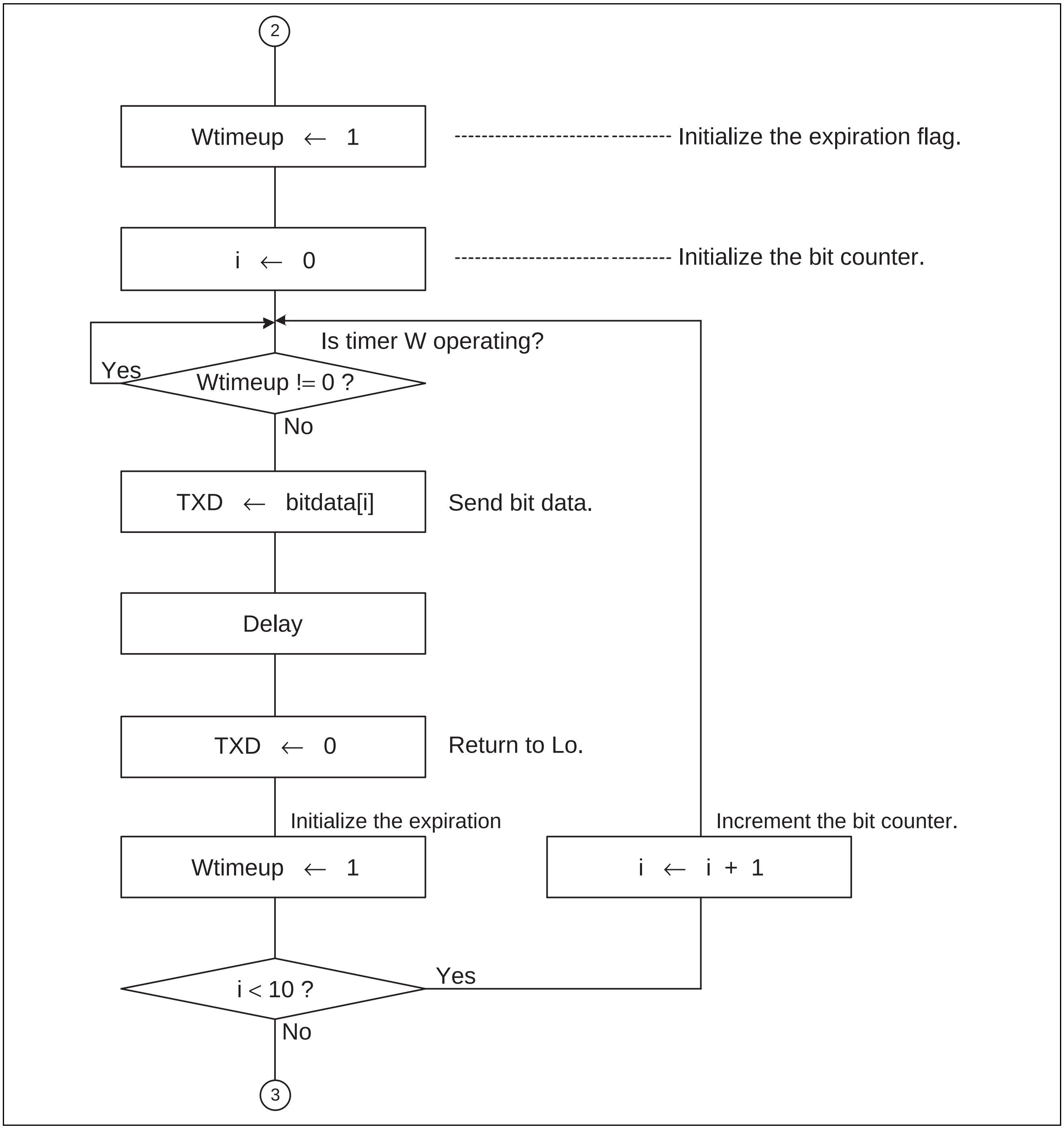


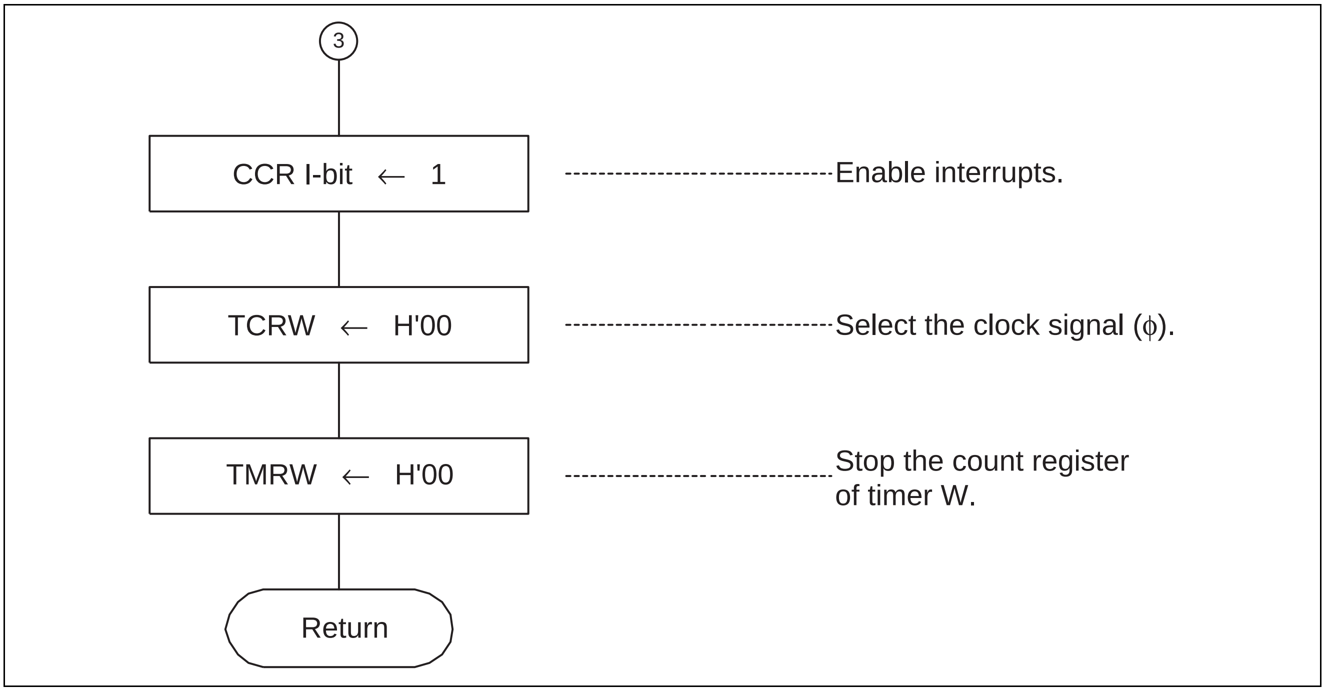

1
2
indata ← indata >> 1
Bit shift
dummy ← indata
Delay
bit ← PDR8
Read bit data
20 times.
dummy ← indata
Delay
Is the input bit Hi?
bit & H'08 != 0 ?
No
Yes
Set the bit to the
ON state.
indata ← indata | H'80
Increment the
bit counter.
i ← i + 1
Are the received bits
equal to 8 or less?
i < 8
Yes
No
TMRW ← H'00
Stop the count
register of timer W.
rdata ← indata
Handle the input data
as the received data.
Return

5.3 Infrared Signal Transmission Routine (irda_snd)

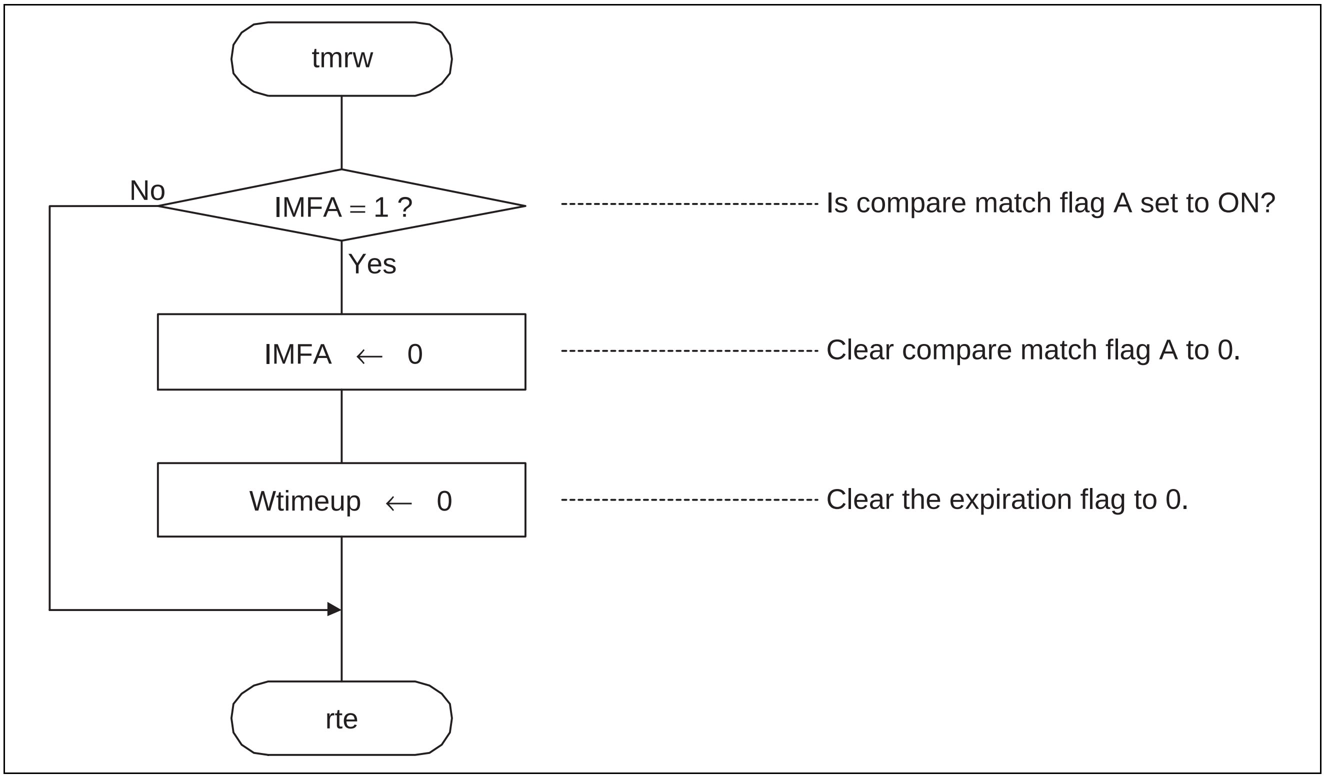



1
TCRW ← H'80
Select the clock signal (φ).
GRA ← 416
416 → 52 µs
TCNT ← 0
Initialize the count register.
IMIEA ← 1
Set compare match interrupt enable A to ON.
IMFA ← 0
Clear compare match flag A.
TMRW ← H'80
Start the count register of timer W.
CCR I-bit ← 0
Permit interrupts.
2


2
Wtimeup ← 1
Initialize the expiration flag.
i ← 0
Initialize the bit counter.
Is timer W operating?
Yes
Wtimeup != 0 ?
No
TXD ← bitdata[i]
Send bit data.
Delay
TXD ← 0
Return to Lo.
Initialize the expiration
Increment the bit counter.
Wtimeup ← 1
i ← i + 1
i < 10 ?
Yes
No
3


3
CCR I-bit ← 1
Enable interrupts.
TCRW ← H'00
Select the clock signal (φ).
TMRW ← H'00
Stop the count register of timer W.
Return

4. Timer W interrupt routine (tmrw)



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 */
/* Infrared communication example */



#include    <machine.h>

/* Symbol defnition */
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 PDR1  *(volatile unsigned char *)0xFFD4        /* Port data register 1      */
#define PDR1_BIT (*(struct BIT *)0xFFD4)               /* Port data register 1      */
#define TXD          PDR1_BIT.b7                       /* Transmit Data             */
#define TST          PDR1_BIT.b5                       /* Receive Data          */
#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 PDR8  *(volatile unsigned char *)0xFFDB        /* Port data register 8      */
#define PCR8  *(volatile unsigned char *)0xFFEB        /* Port control register 8   */

#define SMR    *(volatile unsigned char *)0xFFA8       /* Serial mode register */
#define BRR    *(volatile unsigned char *)0xFFA9       /* Bit rate register    */
#define SCR3 *(volatile unsigned char *)0xFFAA         /* Serial control register 3 */
#define TDR  *(volatile unsigned char *)0xFFAB         /* Transmit data register       */
```

```
#define SSR  *(volatile unsigned char *)0xFFAC        /* Serial status register       */
#define RDR  *(volatile unsigned char *)0xFFAD        /* Receive data register        */
#define PMR1 *(volatile unsigned char *)0xFFE0        /* Port mode register 1 */

#define TMRW  *(volatile unsigned char *)0xFF80       /* Timer mode register W        */
#define TCRW  *(volatile unsigned char *)0xFF81       /* Timer control register W     */
#define TCRW_BIT (*(struct BIT *)0xFF81)              /* Timer control register W     */
#define TIERW *(volatile unsigned char *)0xFF82       /* Timer interrupt enable register W */
#define TIERW_BIT (*(struct BIT *)0xFF82)             /* Timer interrupt enable register  */
#define IMIEA  TIERW_BIT.b0                           /* Compare match interrupt enable A  */
#define TSRW   *(volatile unsigned char *)0xFF83      /* Timer status register W      */
#define TSRW_BIT (*(struct BIT *)0xFF83)              /* Timer status register W      */
#define IMFA  TSRW_BIT.b0                             /* Compare match flag A  */
#define TCNT  *(volatile unsigned short *)0xFF86      /* Timer counter       */
#define GRA       *(volatile unsigned short *)0xFF88  /* General-purpose register A   */

#pragma interrupt (tmrw)
/* Function definition */
extern void INIT(void);                               /* Set stack pointer    */
void irda_rcv( void );                                /* Reception routine    */
void irda_snd( void );                                /* Transmission routine        */
void tmrw(void);                                      /* Timer W interrupt routine */
void main(void);                                      /* Main routine                */


/* RAM definition */
volatile unsigned char   sdata;                       /* Transmit data           */
volatile unsigned char   rdata;                       /* Receive data        */
unsigned char bitdata[10];                            /* Bit data (send)            */
int i,j;                                              /* Loop counter        */
unsigned char bit, indata;                            /* Input data          */
volatile int  Wtimeup;                                /* Timer W timer expiration   */
unsigned char bitpos;                                 /* Bit position        */
char dummy;

/* Vector address */
#pragma section V1                                    /* Set vector section    */
void (*const VEC_TBL1[])(void) = {
      INIT                                            /* H'0000 Reset vector   */
};

#pragma section V2                                    /* Set vector section    */
void (*const VEC_TBL2[])(void) = {
      tmrw                                            /* H'002a Timer W interrupt vector  */
};
#pragma section                                       /* P                     */



/************************************************************/
/* Main program                                             */
/************************************************************/
void main(void)
{
```

```
      set_imask_ccr(1);                             /* CCR I-bit = 1              */

      PCR1 = 0xA0;                                  /* Port 2 bit 7, bit 5 output  */
      PCR8 = 0xF7;                                  /* Port 8 bit 3 input     */

      while(1){
              irda_rcv();                           /* Reception routine          */
              sdata = rdata + 1;
              irda_snd();                           /* Transmission routine   */
      }

}


/************************************************************/
/*    Reception routine                                     */
/************************************************************/
void irda_rcv( void )
{
      indata = 0;                                   /* Initialize input data        */

      TMRW = 0x80;                                  /* Start timer W count register  */
      while(PDR8 & 0x08);                           /* Wait for start bit     */

      for(i=0;i<8;i++){
              TCNT = 0;                             /* Clear timer count register to zero */
              if(i == 0)
                      while(TCNT < 290);            /* 36.25 us               */
              else
                      while(TCNT < 210);            /* 26.25 us               */
              indata >>= 1;                         /* Shift input data       */
              dummy = indata;                       /* Dummy wait (8 cycles)  */
              bit = PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8
                      & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8 & PDR8;
              dummy = indata;                       /* Dummy wait (8 cycles)  */
              if(bit & 0x08) {                      /* If input is H-level    */
                      indata |= 0x80;               /* then set data              */
              }
      }
      TMRW = 0x00;                                  /* Stop timer W count register   */

      rdata = indata;
}


/*************************************************************/
/*    Transmission routine                                   */
/*************************************************************/
void irda_snd( void )
{
      /* Set transmit         */
      bitpos = 0x01;                                /* Set bit position     (0 bit) */
      for(i = 0; i < 10; i++) {
              if(i == 0) {
```

```
                bitdata[0] = 1;                 /* Set start bit                */
        } else if(i == 9) {
                bitdata[9] = 0;                 /* Set stop bit         */
        } else {
                if((bitpos & sdata) == 0) {     /* Set data bit         */
                        bitdata[i] = 1;
                } else {
                        bitdata[i] = 0;
                }
                bitpos <<= 1;                   /* Shift bit position   */
        }
    }

    TCRW = 0x80;                                /* Select clock signal(φ)      */
    GRA = 416;                                  /* 52 us                */
    TCNT = 0;                                   /* Clear timer count register to zero */
    IMIEA = 1;                                  /* Compare match interrupt enable A */
    IMFA = 0;                                   /* Compare match flag A */
    TMRW = 0x80;                                /* Start timer W count register     */
    set_imask_ccr(0);                           /* CCR I-bit = 0        */

    Wtimeup = 1;
    for(i = 0; i < 10; i++) {
            while(Wtimeup);                     /* Wait for 52 us       */
            TXD = bitdata[i];                   /* TXD to ON                    */
            for(j = 0; j < 1; j++);
            TXD = 0;                            /* TXD to OFF           */
            Wtimeup = 1;
    }

    set_imask_ccr(1);                           /* CCR I-bit = 1                */
    TCRW = 0x00;                                /* Select clock signal(φ)     */
    TMRW = 0x00;                                /* Stop timer W count register */

}


/*************************************************************/
/* Timer W interrupt(every 52 µs)                           */
/*************************************************************/
void tmrw(void)
{
    if ( IMFA == 1 ) {
            IMFA = 0;                           /* Clear compare match flag A   */
            Wtimeup = 0;                        /* set time-up          */
    }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.20.03	—	First edition issued