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.

H8/300L SLP Series

Emulating SCI using I/O Port (portSCI)

Introduction

Multi-channel communications with various external devices may be required in some applications. A simple means of communication is to use the serial port. However, due to the limited serial ports available, there may be a need to implement the communication using I/O port.

In this document, an asynchronous communication channel using two I/O lines is implemented. The transmission and reception links have been established with the PC at 9600 bps.

Target Device

H8/38024

Contents

1. Theory

1.1 Overview

The software for simulated serial communication interface is written in C for easy portability. The H8/38024 microcontroller is used as the target in this application note.

The UART protocol used for this application note is 1 start bit, 8 data bits, no parity bit, and 1 stop bit as shown below.

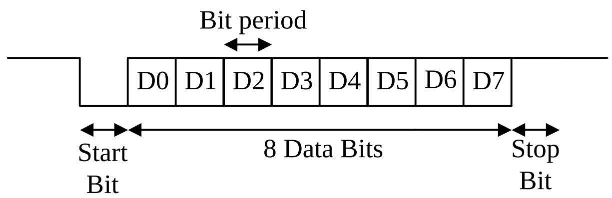

Figure 1 UART Protocol

In order to transmit and receive data correctly, the bit period must be accurate. Slight variations would result in accumulation of timing errors and hence data will be decoded wrongly. The bit period is calculated as follows:

Bit period = 1/baud rate

As for the baud rate of 9600 bps, the bit period would be:

Bit period = 1/9600 = 104.167 µs

In order to generate this bit period, either a timer function or a *for loop* could be used. An accurate value has to be used for the timer register value or the *for loop*.

For transmission, the output port pin is pulled high. A '0' is sent for the start bit followed by 8 data bits, in which the least significant bit (D0) is sent first, and finally a '1' is sent for the stop bit.

The receiving port pin also has to be pulled high. When the signal level goes low, either a start bit is received or unwanted noise causes a drop in voltage level. Hence, a delay of half a bit period is carried out before sampling to verify a '0' for the start bit.

1.2 Implementation

The 38024F CPU board is used in this application note. Pin 6 of Port 1 (P16) is used as the transmit channel and pin 4 of Port 1 (P14) is used to receive the external serial data. The crystal frequency used is 9.8304 MHz and the baud rate is 9600bps. These can be easily changed to the user's requirement in the C program.

For transmission, P16 is configured as an output with its MOS pulled high. The data is transmitted by calling the *transmit* subroutine.

P14 is configured as an input with its MOS also pulled high. This I/O pin is multiplexed with the IRQ4 activation.

Before receiving any data, P14 is set to accept an external interrupt. IRQ4 interrupt is initialized for high-to-low edge triggering, hence the high-to-low transition at the line after receiving the start bit would generate an interrupt to start the IRQ4 interrupt service routine to perform the receive operation.

In the interrupt service routine, P14 is set to function as an input port pin to receive the data stream. The software waits for half a bit period as soon as the IRQ4 interrupt occurs to sample the start bit. After detecting the start bit, the *receive* subroutine waits for a bit period to sample each of the 8 data bits. This process is illustrated in figure 2.

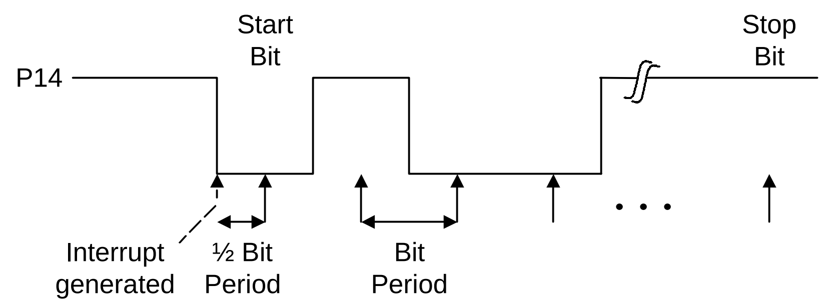

Figure 2 Sampling Periods

Timer F is used to implement the delay for the bit period. Timer F is initialized as a 16-bit timer and generates an interrupt when a compare match occurs. The calculation for the output compare register value is shown in the following section 1.3.

1.3 Output Compare Register Value Calculation

Timer F is a free running counter with a built-in output compare function. It is initialized to generate an interrupt when a compare match occurs. That is, Timer Control Register F (TCRF) starts incrementing and an interrupt is generated once its value matches that of the value in Output Compare Register FH (OCRFH).

The internal clock is set to $\phi/4$, by setting bits 2 to 1 of Timer Control Register F (TCRF) as indicated in bold in table 1.

Table 1 Clock Selection for Timer F

Bit 2 CKSL2	Bit 1 CKSL1	Bit 0 CKSL0	Description
0	0	0	Counting on external event (TMIF) rising/falling edge
0	0	1	
0	1	0	
0	1	1	Use prohibited
1	0	0	Internal clock: counting on $\phi/32$
1	0	1	Internal clock: counting on $\phi/16$
1	**1**	**0**	**Internal clock: counting on $\phi/4$**
1	1	1	Internal clock: counting on $\phi w/4$

The output compare register value is calculated as follows:

For: Bit period = 1/baud rate

ϕ = crystal frequency/2

Internal clock = $\phi/4$

d = output compare register value

Therefore: d × internal clock period = bit period required

$$d \times \frac{1}{\frac{\text{crystal frequency}}{2 \times 4}} = \frac{1}{\text{bit period}}$$

$$d = \frac{\text{crystal frequency}}{8 \times \text{baud rate}}$$

However, Timer F needs a time of around 25µs before it is initialized and starts incrementing its 16-bit timer counter TCF. This would result in a longer bit period, hence the value needs to be offset. It is found that the offset of 49 is required in this application note when the crystal oscillator (9.3204 MHz) is used.

Therefore, the value to be loaded into OCRFH is d – 49.

Note: This delay of Timer F for initialization depends on the value of crystal oscillator used. User has to change the offset value if a different crystal oscillator is used.

1.4 Alternative Solution

- Polling for start bit
  Instead of using an interrupt to detect the start bit, the user can use a polling method. The user can continuously read the receive pin, P14, to wait for the start bit. An example code is shown below.

```
while(1)
{
  if (RX == 0)
  receive();
}
```

  However, this method could only be used when the main program has nothing else to perform as the user needs to continuously check the receive pin.

- Using *for loop* for delay
  A *for loop* can be used in place of Timer F to implement the delay required for the bit period. The user would be required to find out the suitable delay value for the *for loop*. An example for such a delay subroutine is shown as follows.

```
void delay (unsigned short d)
{
  for (i = 0; i < d; i ++)
}
```

2. Operation

2.1 Hardware Setup

In order to communicate with an external device, the RS-232C connection has to be setup. A simple serial driver has to be built to condition the signal level between the MCU and external target device. The I/O pins have to be pulled high via 10-K pull-up resistors.

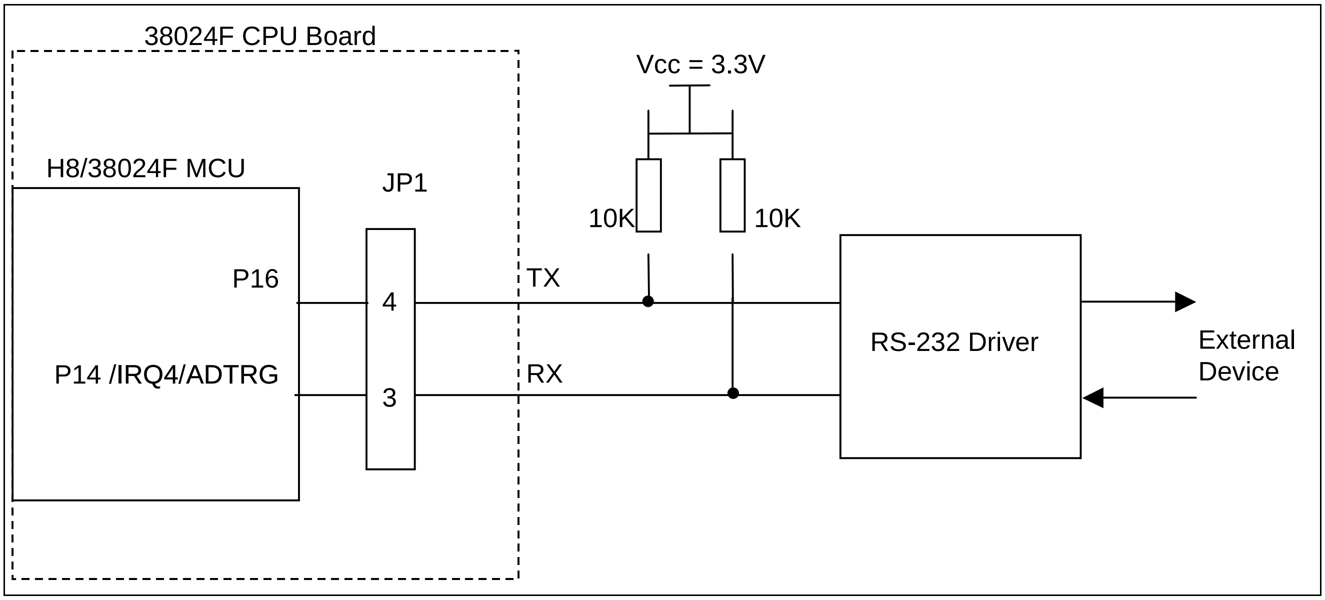

Figure 3 The MCU Setup with the RS-232C Driver

A schematic diagram for the RS-232C driver is given in figure 4.

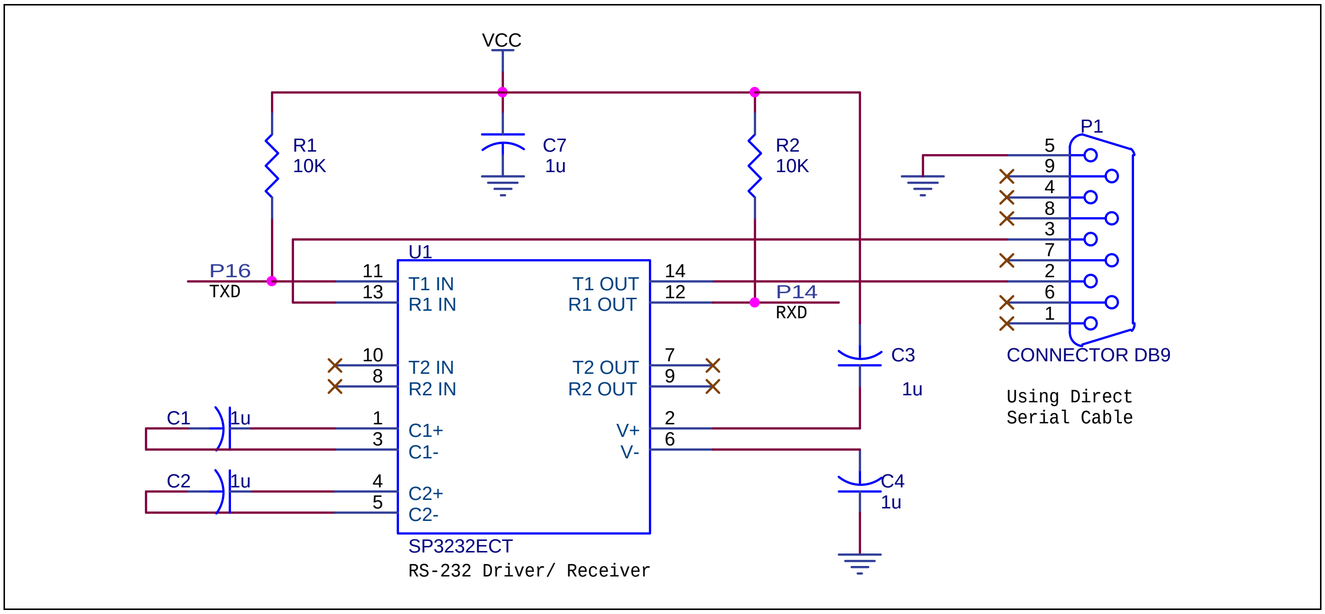

Figure 4 RS-232C Driver

2.2 Hyper Terminal Setting

When the user sets up communication between the MCU and PC using HyperTerminal, the COM port settings have to be made in accordance with the UART protocol and baud rate used in the program.

For the baud rate of 9600 bps, the following settings have to be made to the COM port connected to the RS-232C driver. Select *Properties* in the *File* menu of HyperTerminal window and click on *Configure…* to change the Port Settings.

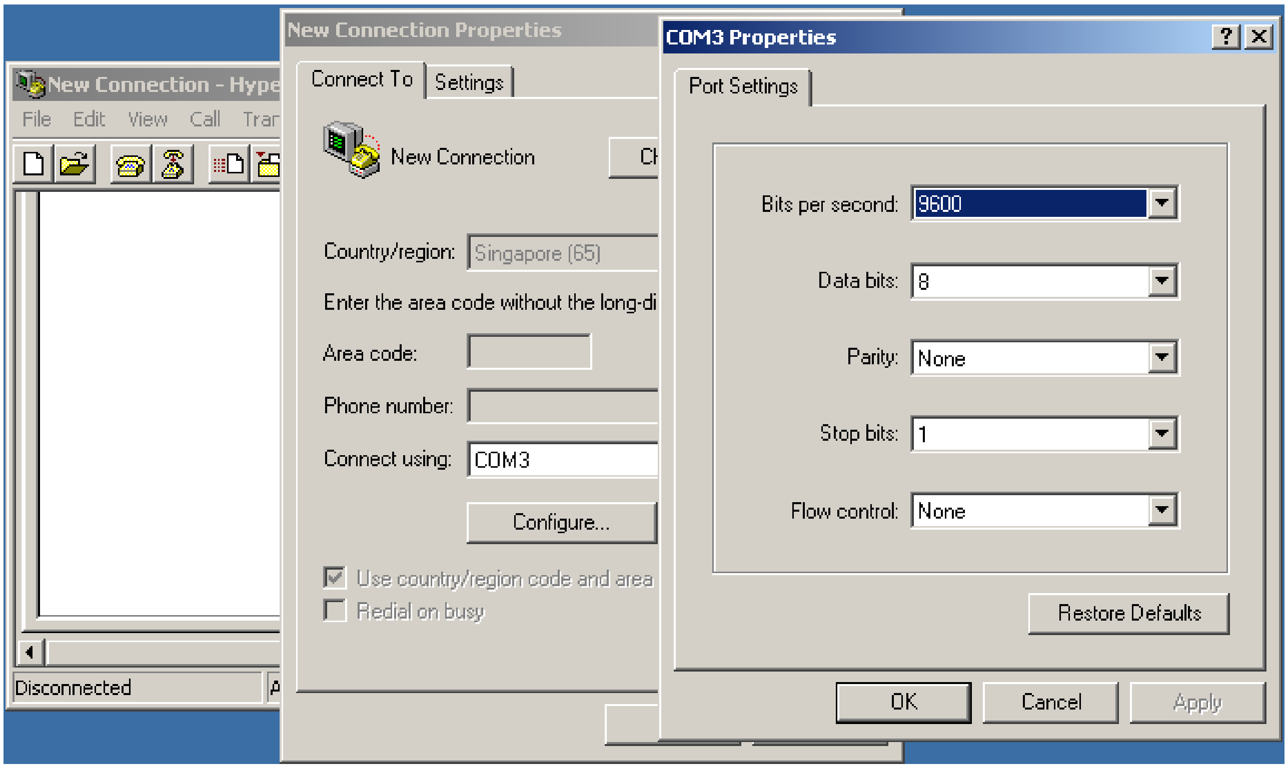

Figure 5 PC HyperTerminal Settings

Next, setup the HyperTerminal so that the character keyed in by the user will be shown in the HyperTerminal window. Click on *ASCII Setup…* in the *Settings* tab and click in the check box of *Echo typed characters locally* as shown in figure 6.

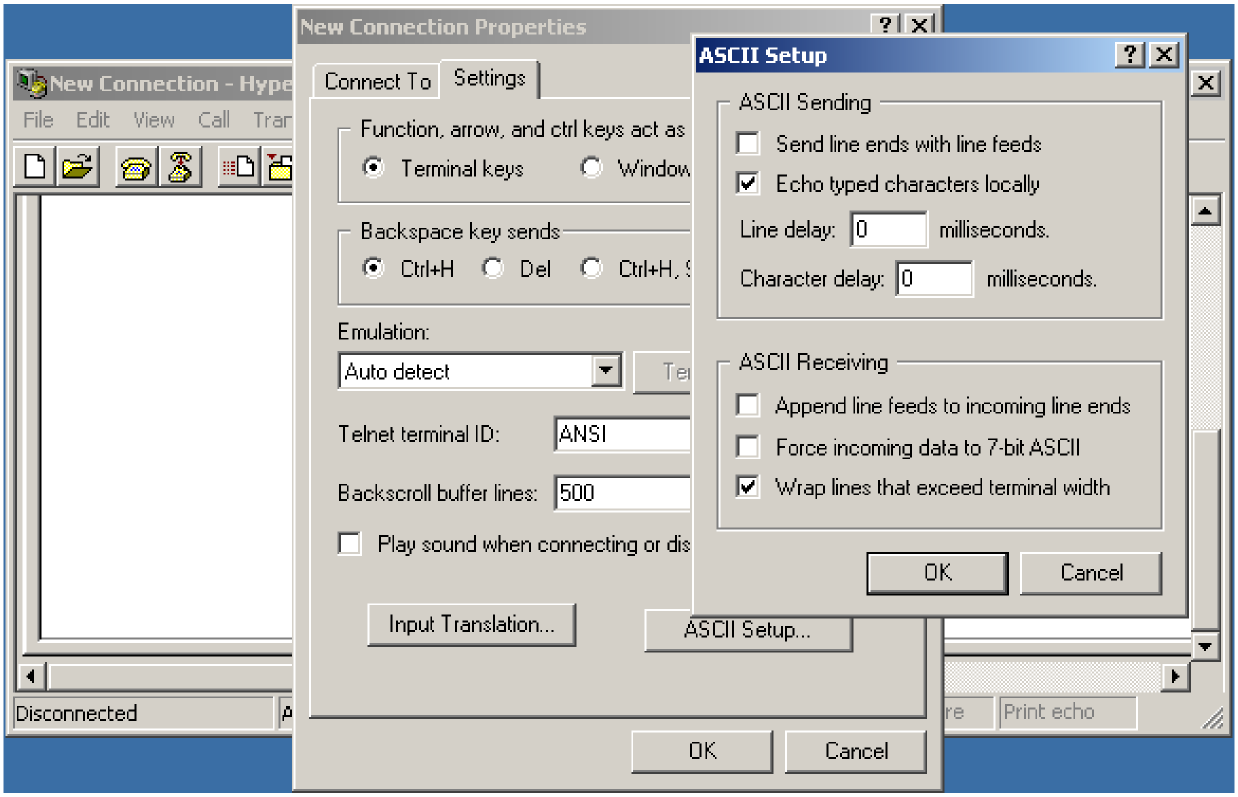

Figure 6 ASCII Setup

By setting the correct baud rate, the user will be able to see the characters **Test** displayed in the HyperTerminal. The user can enter any characters in the Hyper Terminal and the decoded character will be re-transmitted onto the hyper terminal window for verification. Figure 7 shows an example of the result on the Hyper Terminal window. If the character is decoded wrongly (when the stop bit is not detected), **Er** is transmitted and will be displayed on the Hyper Terminal window.

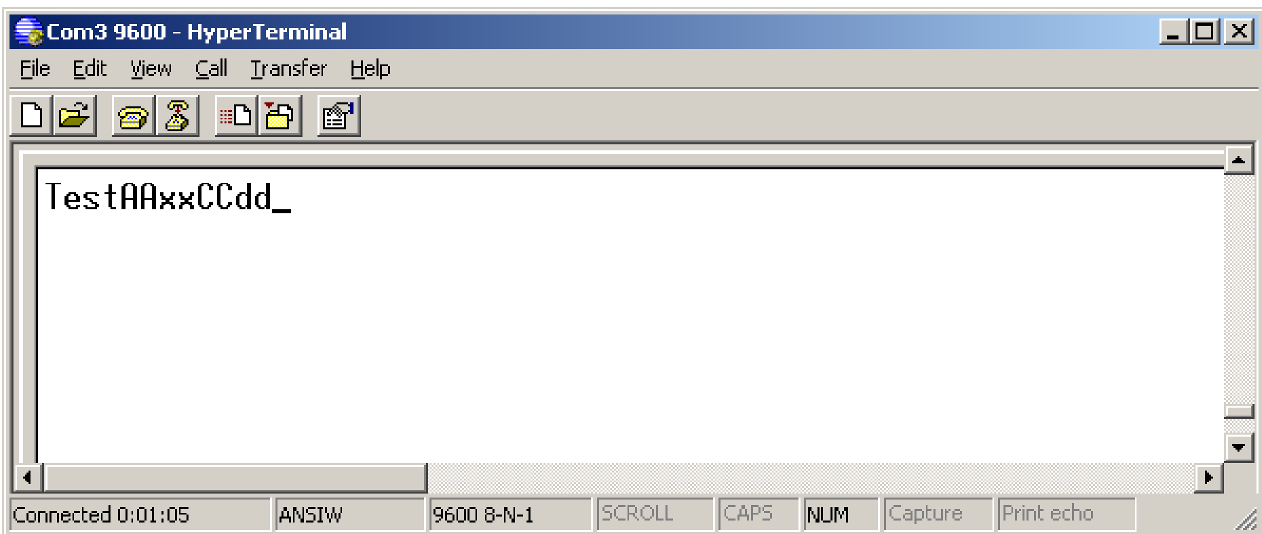

Figure 7 Results Displayed on PC HyperTerminal

3. Code Listing

The following attached code for this application note is generated using the HEW project generator targeting at the H8/38024 MCU. The tool chain used is the SLP/TINY tool chain.

Flowcharts are included to illustrate the main functionality and to give a better understanding for user.

Note: Optimization must be turned off for the program to work.

```
/**********************************************************************/
/*                                                                    */
/*  FILE        :Emulate_SCI.c                                           */
/*  DATE        :Mon, Aug 25, 2003                                    */
/*  DESCRIPTION :Main Program                                         */
/*  CPU TYPE    :H8/38024F                                            */
/*                                                                    */
/*  This file is generated by Renesas Project Generator (Ver.2.1).    */
/*                                                                    */
/**********************************************************************/

//include 38024F IO define header file
#include "iodefine.h"

#include <machine.h>
#include <_h_c_lib.h>

//include additional flag define header file
#include "flagdefine.h"

void initialize (void);
void transmit (unsigned char);
void receive (void);
void transmit_string (void);
void delay (unsigned short);

char *buff_ptr;
static const char TX_buffer[] = "Test";       // Transmit buffer
char RX_buffer;                               // Receive buffer
unsigned int i =0;

//--------------- Main Program ----------------//
void main(void)
{
   initialize();

   transmit_string();

   while(1);
}

//--------- Initialization of Port 1, Timer F & IRQ4 ---------//
void initialize (void)
{
// Initialize Port 1
   P_IO.PUCR1.BIT.PUCR16 = 1;      // P16 MOS pull-up
   TX = 1;
   P_IO.PCR1.BIT.PCR16 = 1;        // P16 as output (TX)

// Initialize IRQ4
   P_SYSCR.IEGR.BIT.IEG4 = 0;      // Interrupt generated at falling edge of
                                   //IRQ4
   P_SYSCR.IENR1.BIT.IEN4 = 1;     // Enables IRQ4
   P_IO.PMR1.BIT.IRQ4 = 1;         // P14 used as IRQ4
```

```
// Initialize Timer F
   P_TMRF.TCRF.BYTE = 0x8E;        // Set TMOFH pin output level to HIGH and
                                   //internal clock of o/4
   P_TMRF.TCSRF.BIT.CCLRH = 1;     //TCF cleared when TCF and OCRF match

}

//--------------- Transmit a character ----------------//

void transmit (unsigned char a)
{
   int i;
   MON_RAM.TX_CHAR.BYTE = a;

// start bit
   TX = 0;
   delay(bit_period);

// 8 data bits
   for (i=0; i<8; i++)
   {
      if (MON_RAM.TX_CHAR.BIT.bit0 == 0)
         TX = 0;

      else
         TX = 1;

      delay(bit_period);
      MON_RAM.TX_CHAR.BYTE = MON_RAM.TX_CHAR.BYTE >> 1;
   }

// stop bit
   TX = 1;
   delay(bit_period);
}

//--------------- Transmit characters in Transmit Buffer ----------------//
void transmit_string (void)
{
   buff_ptr = (char *)&TX_buffer;

   while (  *buff_ptr != 0)
   {
      MON_RAM.TX_CHAR.BYTE = (*buff_ptr++);
      transmit(MON_RAM.TX_CHAR.BYTE);    // call transmit subroutine to tansmit
                                         //each character
   }
}

//--------------- Store characters in RX_CHAR ----------------//
void receive (void)
{
   int j;
```

```
    RX_buffer = 0;

// Receive data bits
    for (j=0; j<8; j++)
    {
        delay(bit_period);
        if (RX == 1)
            MON_RAM.RX_CHAR.BYTE = MON_RAM.RX_CHAR.BYTE | (0x01 << j);

        else
            MON_RAM.RX_CHAR.BYTE = MON_RAM.RX_CHAR.BYTE & rotlc(j,0xFE);
    }

// Receive stop bit
    delay(bit_period);
    if (RX != 1)
    {
        transmit('E');          // error if sampled stop bit='0', transmit 'Er'
        transmit('r');
    }
    else
    {
        RX_buffer = MON_RAM.RX_CHAR.BYTE ;       // save character in receive
                                                 //buffer
        transmit(RX_buffer);                     // transmit character in receive
                                                 //buffer
    }
    P_IO.PMR1.BIT.IRQ4 = 1;               // P14 used as IRQ4
}

//--------------- Delay function using Timer F ----------------//
void delay (unsigned short d)
{
    d = d-49;           // decrease d to offset for delay during setup of timer

    P_TMRF.OCRF.BYTE.H = d<<8;      // save count (d) in output compare register
    P_TMRF.OCRF.BYTE.L = d;

    P_TMRF.TCSRF.BIT.CMFH = 0;      // Clear compare match flag
    P_TMRF.TCF.BYTE.H = 0;          // Clear counter and start the timer F
    P_TMRF.TCF.BYTE.L = 0;

    while (P_TMRF.TCSRF.BIT.CMFH == 0);   // Loop until a compare match occurs

    P_TMRF.TCSRF.BIT.CMFH = 0;      // Clear compare match flag
}
```

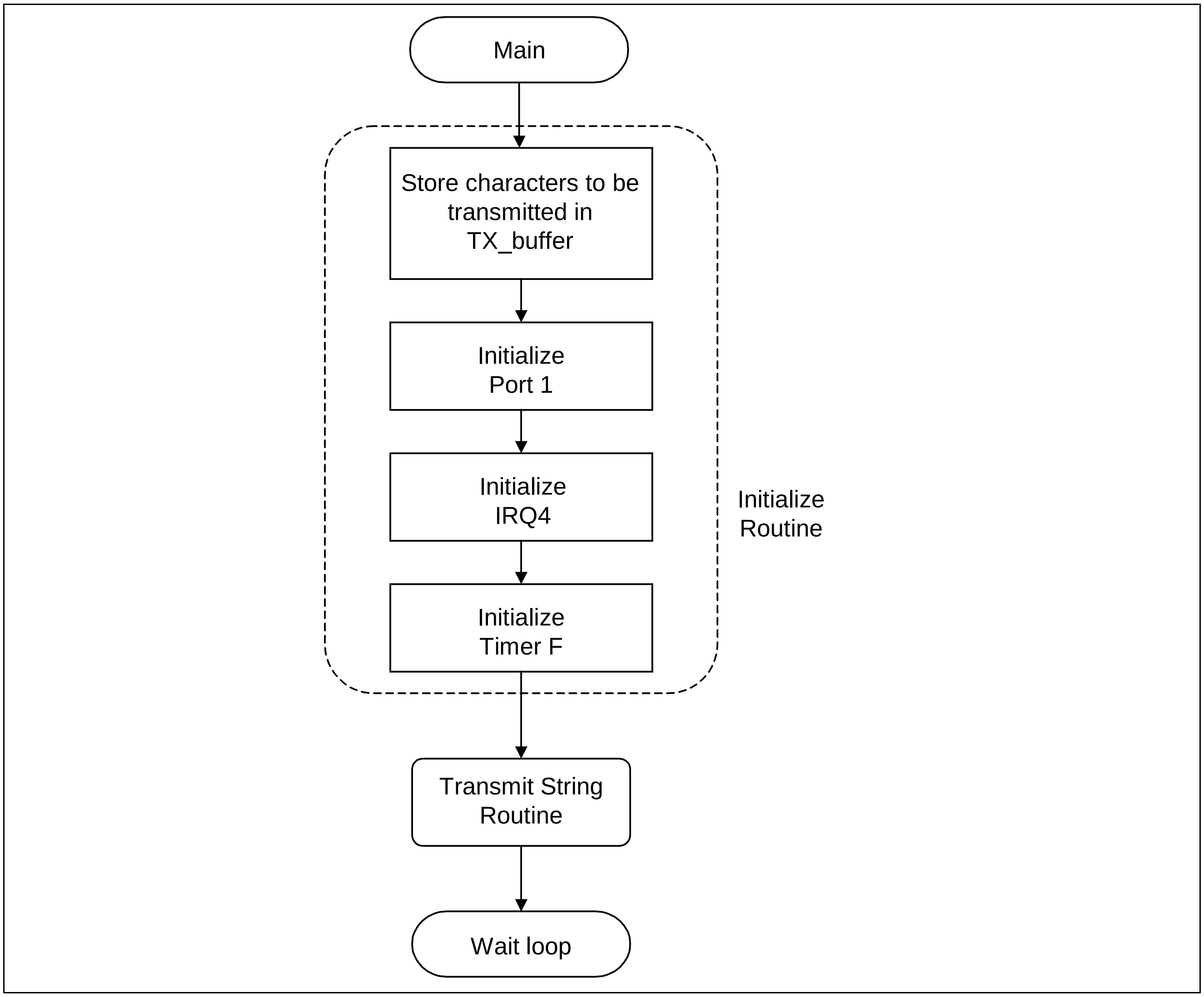

Figure 8 Main Program

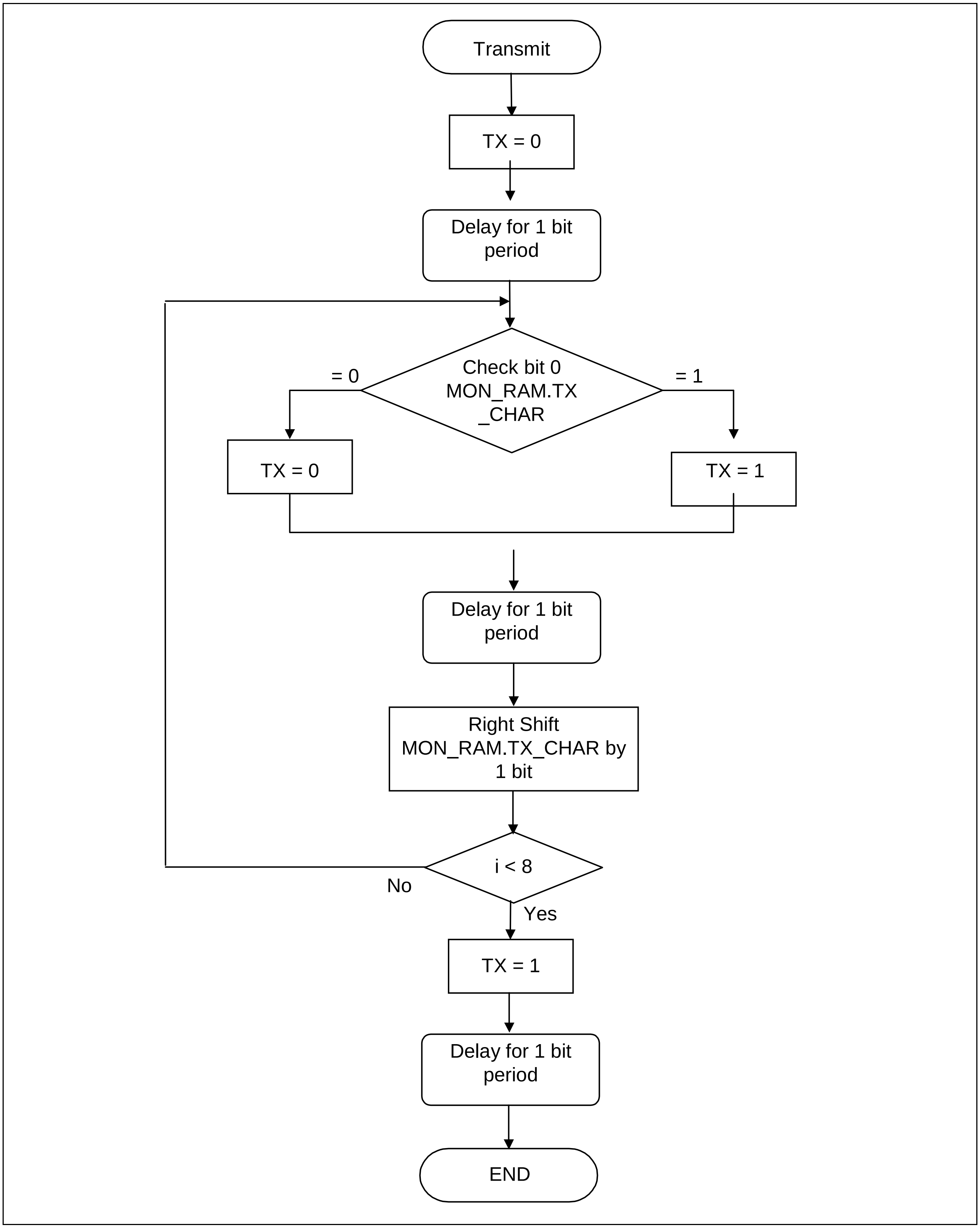

Figure 9 Transmit Routine

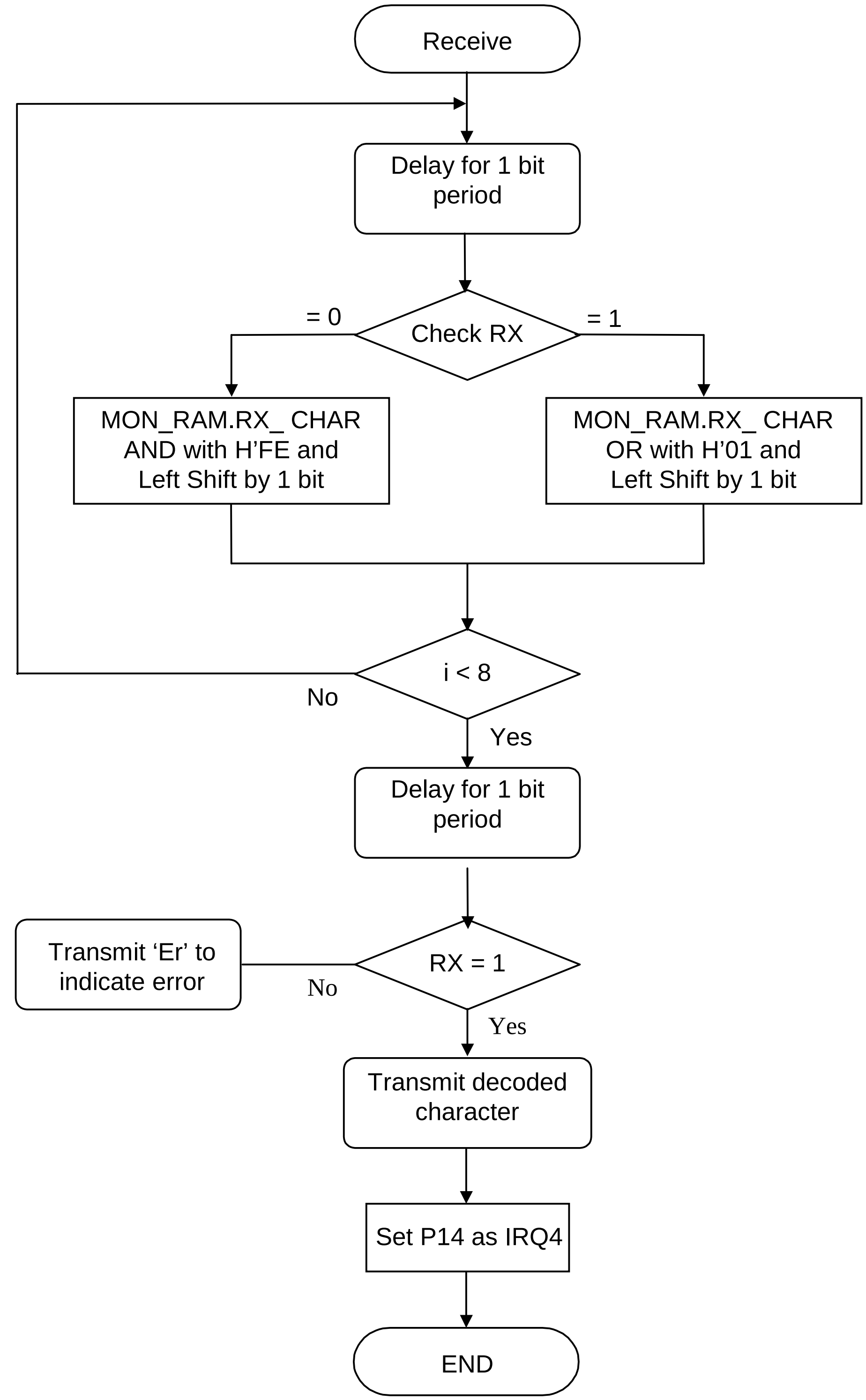

Figure 10 Receive Routine

```
/***********************************************************************/
/*  FILE        :flagdefine.h                                          */
/*  DATE        :Tue, Aug 19, 2003                                     */
/*  DESCRIPTION :Definition of flag                                   */
/*  CPU TYPE    :H8/38024                                              */
/*  Additional header file                                            */
/*                                                                     */
/***********************************************************************/

#define XTAL         9830400L           // for crystal frequency of 9.83204Mhz

#define  BAUD        9600L              // for baud rate of 9600

#define  bit_period  (XTAL / (8*BAUD))  // for internal clk = Ø /4
                                        // NOTE: Ø = XTAL/2
#define  sample      ((bit_period) / 2L)

#define  TX          P_IO.PDR1.BIT.P16  // Port 1 pin 6 as transmit pin
#define  RX          P_IO.PDR1.BIT.P14  // Port 1 pin 4 as receive pin

/**************************************************************************/
/*      H8/38024 Flag Definition File                        Ver 1.0      */
/**************************************************************************/
struct MON                                            /*struct MON_RAM*/
{                                                     /*              */
            union {                                   /*              */
                  unsigned char BYTE;                 /*  Byte Access */
                  struct {                            /*  Bit  Access */
                         unsigned char bit7:1;        /*              */
                         unsigned char bit6:1;        /*              */
                         unsigned char bit5:1;        /*              */
                         unsigned char bit4:1;        /*              */
                         unsigned char bit3:1;        /*              */
                   unsigned char bit2:1;              /*              */
                   unsigned char bit1:1;              /*              */
                         unsigned char bit0:1;        /*              */
                         }      BIT;                  /*              */
                  } TX_CHAR;                          /*              */
            union {                                   /*              */
                  unsigned char BYTE;                 /*  Byte Access */
                  struct {                            /*  Bit  Access */
                         unsigned char bit7:1;        /*              */
                         unsigned char bit6:1;        /*              */
                         unsigned char bit5:1;        /*              */
                         unsigned char bit4:1;        /*              */
                         unsigned char bit3:1;        /*              */
                   unsigned char bit2:1;              /*              */
                   unsigned char bit1:1;              /*              */
                         unsigned char bit0:1;        /*              */
                         }      BIT;                  /*              */
                  } RX_CHAR;                          /*              */
};                                                    /*              */
#define MON_RAM    (*(volatile struct MON    *)0xFD20)     /* MON_RAM Addr */
```

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :intprg.c                                              */
/*  DATE        :Mon, Aug 25, 2003                                     */
/*  DESCRIPTION :Interrupt Program                                     */
/*  CPU TYPE    :H8/38024F                                             */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.2.1).     */
/*                                                                     */
/***********************************************************************/

// include 38024F IO define header file
#include "iodefine.h"

#include <machine.h>

// include additional flag define  header file
#include "flagdefine.h"

extern void delay (unsigned short);
extern void receive (void);
```

NOTE: Add the following in the IRQ4 vector

```
void INT_IRQ4(void)
{
   set_imask_ccr(1);              // Disable interrupts

   P_IO.PMR1.BIT.IRQ4 = 0;        // P14 used as i/o pin

   P_IO.PCR1.BIT.PCR14 = 0;       // P14 as Input (RX)
   P_IO.PUCR1.BIT.PUCR14 = 1;     // P14 MOS pull-up

// start bit
    delay(sample);           // delay half a bit period to sample at the middle
                             // of each bit

   if (RX == 0)      // Start receive sequence if sampled start bit equals '0'
      receive();

   P_SYSCR.IRR1.BIT.IRRI4 = 0;    // clear interrupt request flag
}
```

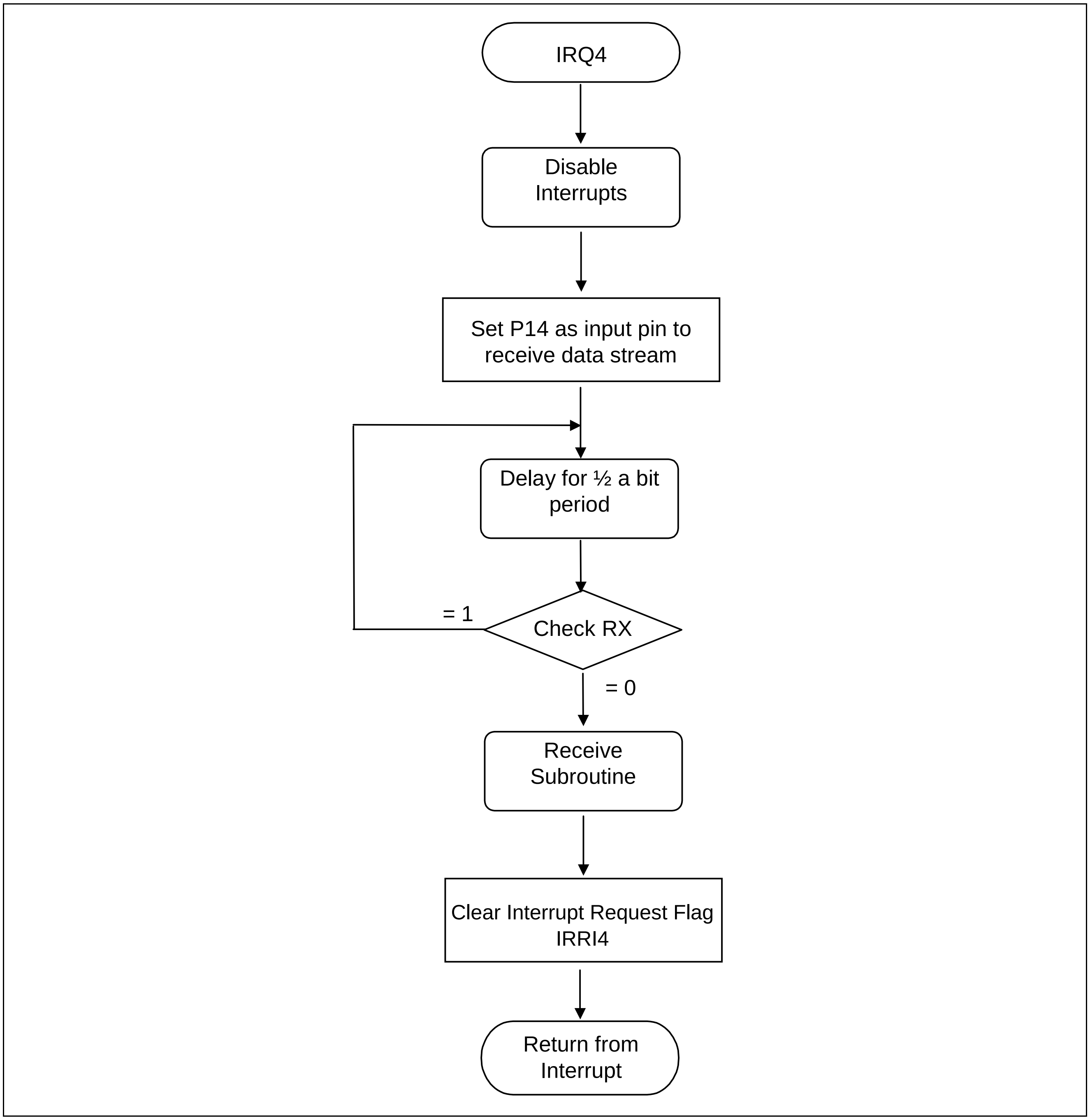

Figure 11 Interrupt Service Routine 4 (IRQ4)

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.10.04	—	First edition issued