M16C/26A,29 Group

CRC Calculation Circuit

1. Abstract

This application note describes how to use CRC calculation circuit of the M16C/26A and M16C/29 Group microcomputers, and shows a method on how to detect a communication error.

2. Introduction

The explanation of this issue is applied to the following condition:

- MCU: M16C/26A Group
  M16C/29 Group

This program can be operated under the condition of M16C family products with the same SFR (Special Function Register) as M16C/26A, 29 group products. Because some functions may be modified of the M16C family products, see the user's manual. When using the functions shown in this application note, evaluate them carefully for an operation.

3. Specifications

CRC (Cyclic Redundancy Check) is a block check based on CRC code, which is used for detecting a communication error. The M16C/26A and M16C/29 group microcomputers can generate CRC code using a CRC calculation circuit.

3.1 CRC calculation and CRC code

In CRC calculations, communication data is handled as a numerical value, which is called a message polynomial.

CRC calculations are a calculation method that divides this message polynomial by a constant called the generating polynomial (modulo-2 division).

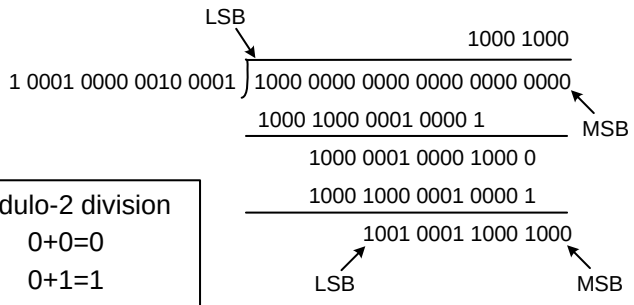
One of the following two expressions can be used for the generating polynomial.

- CRC-CCITT:
  $X^{16}+X^{12}+X^{5}+1$
- CRC-16:
  $X^{16}+X^{15}+X^{2}+1$

The resulting "remainder" of this calculation is called CRC code.

Generating Polynomial: 1 0001 0000 0010 0001 ($X^{16}+X^{12}+X^5+1$)
Message Polynomial: 1000 0000 (01H is transmitted and received LSB first)

CRC Code: 1001 0001 1000 1000 (Because LSB first, = 1189h)



Modulo-2 division
0+0=0
0+1=1
1+0=1
1+1=0
-1=1

Generating Polynomial: 1 1000 0000 0000 0101 ($X^{16}+X^{15}+X^2+1$)
Message Polynomial: 1000 0000 (01H is transmitted and received LSB first)

CRC Code: 1000 0011 0000 0011 (Because LSB first, = C0C1h)

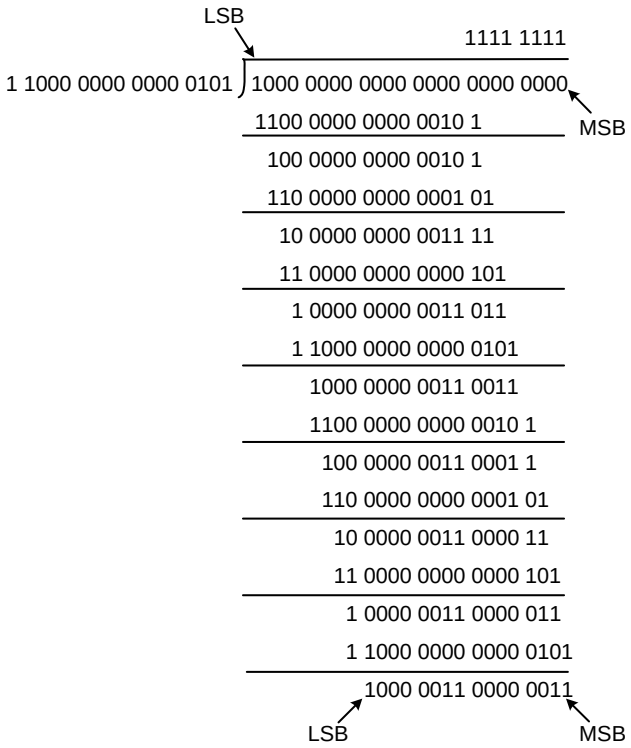


Figure 1. CRC Calculation Example

3.2 Hardware configuration of the CRC calculation circuit

Although the principle of CRC calculations is a division, a hardware implementation of CRC calculations uses a shift register and exclusive OR to generate CRC code.

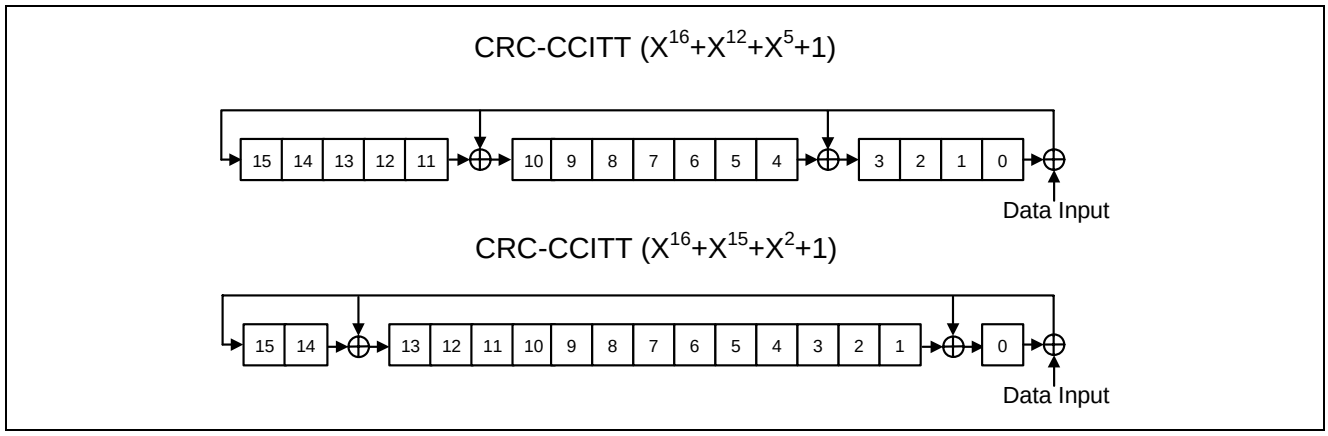


Figure 2. CRC Calculation Circuit Example

3.3 Internal operation during CRC calculation

The M16C/26A and M16C/29 group microcomputers operate in such a way that when one byte of data is written to the CRC input register, CRC code is generated based on that data and the content of the CRC data register, then the generated CRC code is stored in the CRC data register. Therefore, when transmitting or receiving multiple bytes of data, the transmit/receive data can be written to the CRC input register successively, and the content eventually stored in the CRC data register can be used in block check as CRC code.

Furthermore, the CRC circuit includes the ability to snoop reads and writes to certain SFR addresses. This can be used to accumulate the CRC value on a stream of data without using extra bandwidth to explicitly write data into the CRCIN register. For example, it may be useful to snoops the writes to a UART TX buffer, or the reads from a UART RX buffer.

To snoops an SFR address, the target address is written to the CRC Snoop Address Register (CRCSAR). The two most signification bits of this register enable snooping on reads or writes to the target address. If the target SFR is written to by the CPU or DMA, and the CRC snoop write bit is set (CRCSW=1), the CRC will latch the data into the CRCIN register. The new CRC code will be set in the CRCD register.

The following shows how CRC calculations are performed when 12345678h (001 0010 0011 0100 0101 0110 0111 1000b) is transmitted/received for LSB first mode.

3.3.1 For CRC-CCITT

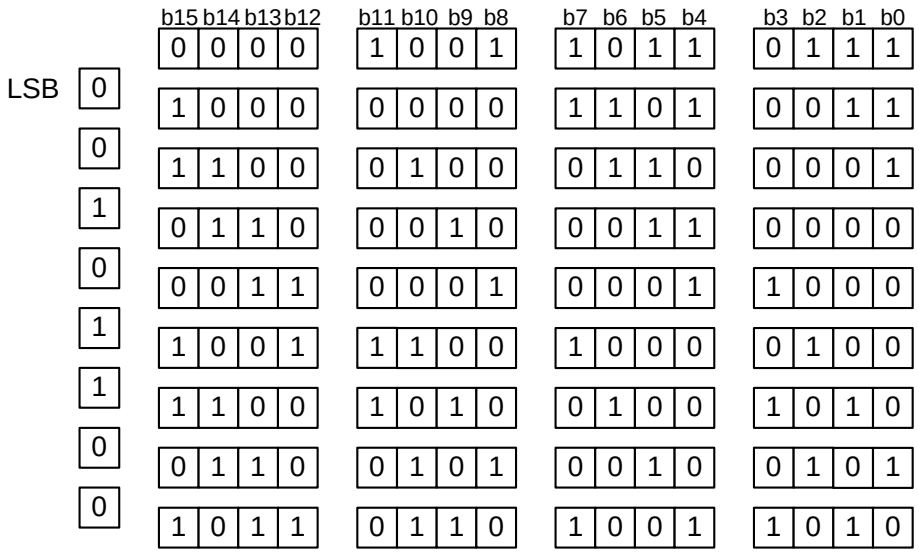
1. Write the initial value “0000h” to the CRC data register.
2. Write the first transmitted/received data “78h” to the CRC input register.
3. The data in the CRC input register is fed into the CRC data register one bit at a time beginning with the LSB, and the arithmetic operation shown below is performed.
   This operation results in the content of the CRC data register being changed to “FFCFh”.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LSB 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
1	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1	0
1	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1

4. Hold the content of the CRC data register intact while writing the next transmitted/received data “56h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “09B7h”.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1
LSB 0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
1	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1
0	1	0	0	1	1	0	1	1	0	1	1	1	0	1	0	1
1	0	1	0	0	1	1	0	1	1	0	1	1	1	0	1	0
0	1	0	1	0	0	1	1	0	1	1	0	1	1	1	0	1
1	0	0	0	1	0	0	1	1	0	1	1	0	1	1	1	0
0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	1	1

5. Hold the content of the CRC data register intact while writing the next transmitted/received data “34h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “B69Ah”.



6. Hold the content of the CRC data register intact while writing the next transmitted/received data “12h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “08F6h”.

3.3.2 For CRC-16

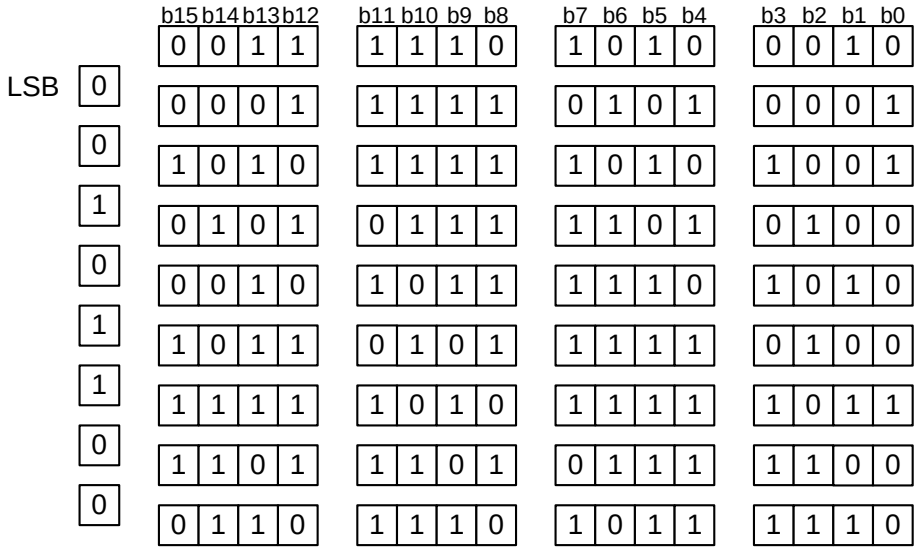
1. Write the initial value “0000h” to the CRC data register.
2. Write the first transmitted/received data “78h” to the CRC input register.
3. The data in the CRC input register is fed into the CRC data register one bit at a time beginning with the LSB, and the arithmetic operation shown below is performed.
   This operation results in the content of the CRC data register being changed to “2200h.”

		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0

4. Hold the content of the CRC data register intact while writing the next transmitted/received data “56h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “3EA2h”.

		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
		0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
LSB	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	1
	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0
	0	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0
	1	0	1	1	0	1	0	1	0	0	0	0	1	0	0	0	1
	0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0	1
	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
	0	0	1	1	1	1	1	1	0	1	0	1	0	0	0	1	0

5. Hold the content of the CRC data register intact while writing the next transmitted/received data “34h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “6EBEh”.

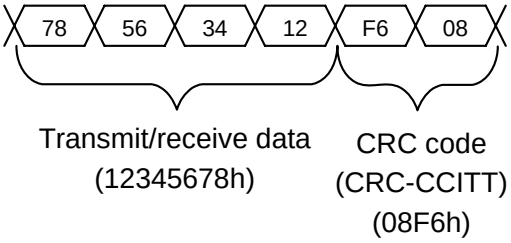


6. Hold the content of the CRC data register intact while writing the next transmitted/received data “12h” to the CRC input register.
   This operation results in the content of the CRC data register being changed to “7D6Eh”.

3.4 How to use the CRC

When transmitting/receiving multiple bytes of data, detection of a communication error can be facilitated by transmitting/receiving CRC code along with the transmitted/received data.

Transmit/receive format for the case where 12345678h
(001 0010 0011 0100 0101 0110 0111 1000b) is transmitted/received LSB first



3.4.1 Setup procedure on transmit Side

(1) Write the initial value “0000h” to the CRC data register. Set a CRC polynomial and CRC mode in the CRC mode register.
(2) Set the address of the transmit buffer register in the SFR monitor address register and at the same time, set the write monitor enable bit to “1”.
(3) Write bytes of transmit data to the transmit buffer successively.
(4) When all bytes of data have been transmitted, read out the content (CRC code) of the CRC data register and write this CRC code to the transmit buffer register to transmit it.

3.4.2 Setup procedure on receive side

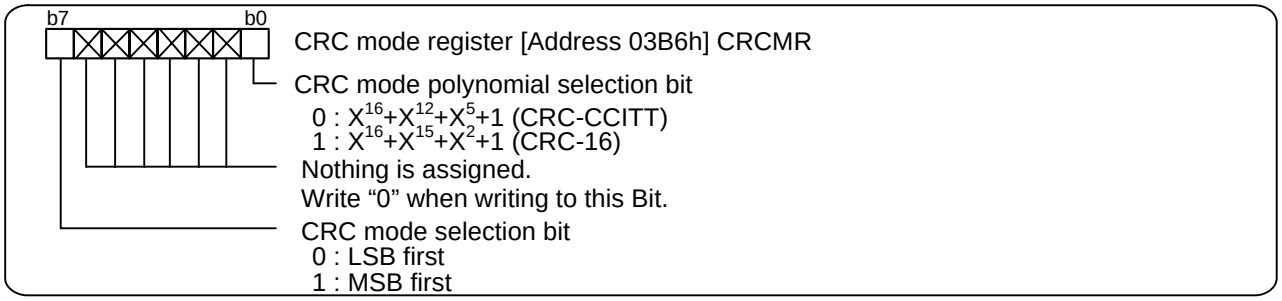
(1) Write the initial value “0000h” to the CRC data register. Set a CRC polynomial and CRC mode in the CRC mode register.
(2) Set the address of the receive buffer register in the SFR monitor address register and at the same time, set the read monitor enable bit to “1”.
(3) Each time data reception is completed, read the received data from the receive buffer register. When all bytes of received data have been read out, the content of the CRC data register on the receive side and that of the transmit side should be the same.
(4) Since CRC code is transmitted from the transmit side, read it out from the receive buffer register in the same way like the received data at each time data reception is completed.
(5) When all bytes of data including CRC code have been received, the content of the CRC data register should be “0000h”. It means that the reception has finished normally. If the content of the CRC data register is not “0000h”, it means that a communication error occurred. In this case, please request a re-transmission from the transmit side.

3.5 Register setting

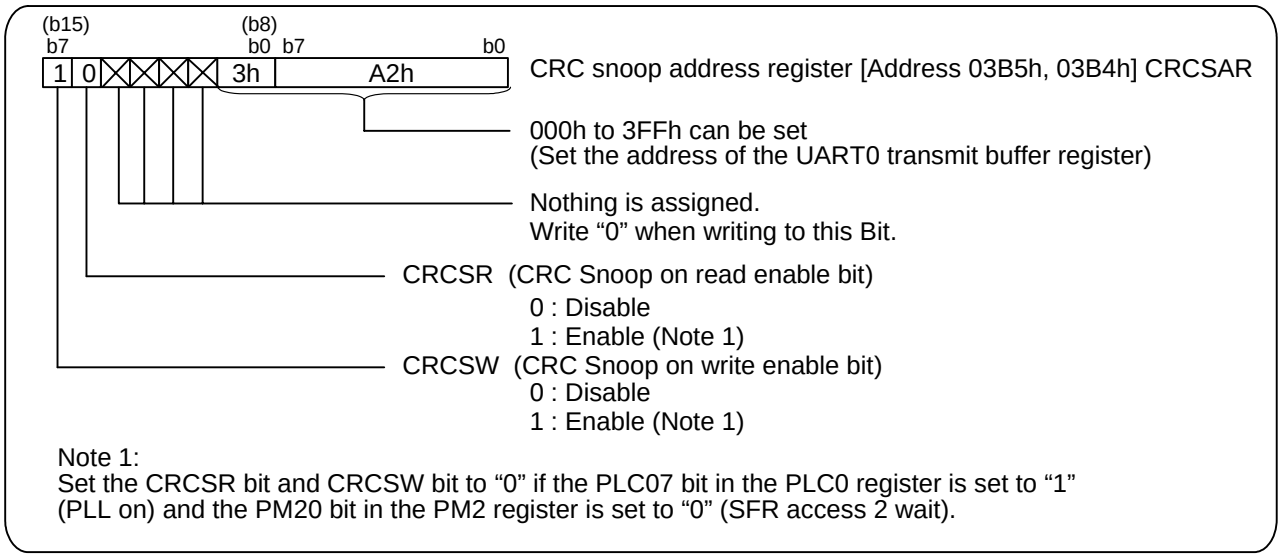
The following procedure in this application note is based on M16C/29 group products, M16C/26A group's setup procedure please refer to the hardware user's manual.

3.5.1 Transmit side register setting

(1) Setting CRC mode register

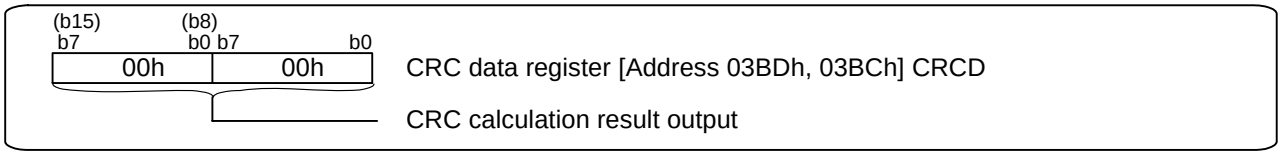


(2) Setting CRC snoop address register



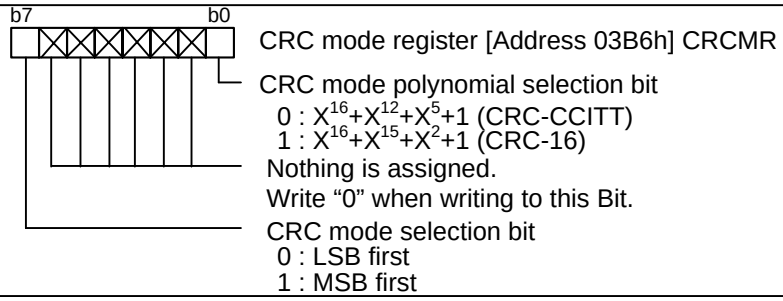
Note 1:
Set the CRCSR bit and CRCSW bit to "0" if the PLC07 bit in the PLC0 register is set to "1" (PLL on) and the PM20 bit in the PM2 register is set to "0" (SFR access 2 wait).

(3) Setting CRC data register

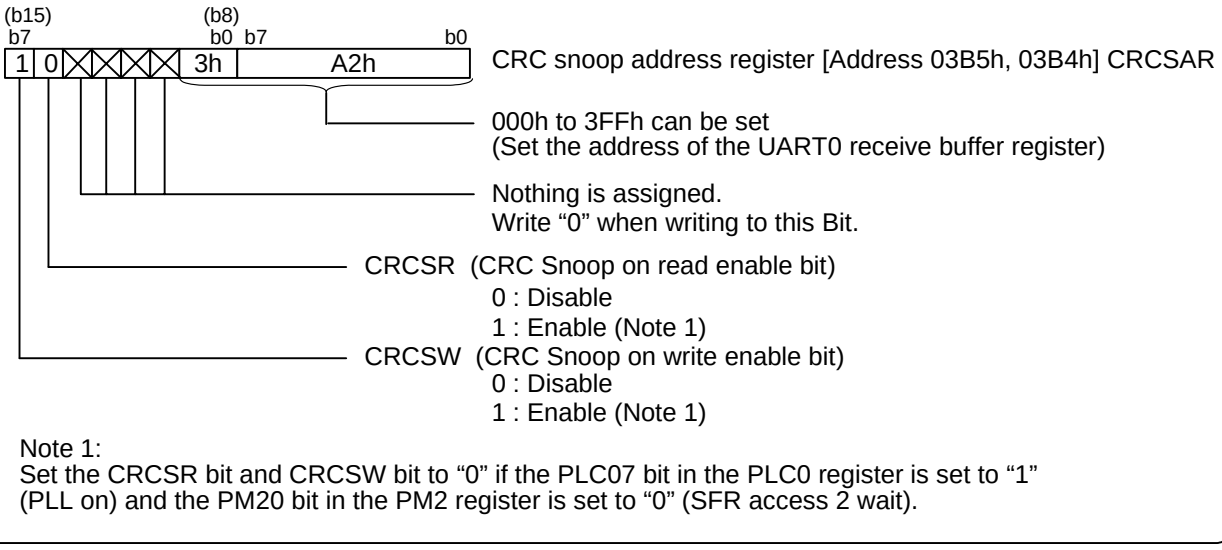


3.5.2 Receive side register setting

(1) Setting CRC mode register

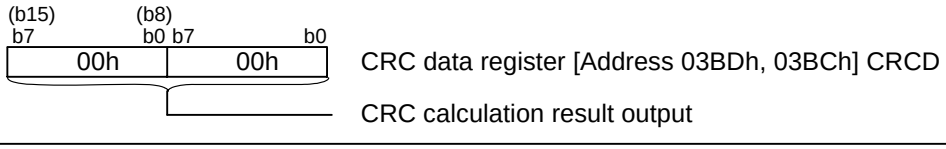


(2) Setting CRC snoop address register



Note 1:
Set the CRCSR bit and CRCSW bit to “0” if the PLC07 bit in the PLC0 register is set to “1” (PLL on) and the PM20 bit in the PM2 register is set to “0” (SFR access 2 wait).

(3) Setting CRC data register



4. The example of reference program

The program shown below is an example where 4 bytes of data are transmitted/received in clock asynchronous serial I/O mode and then CRC code of communication data is transferred as figure3.

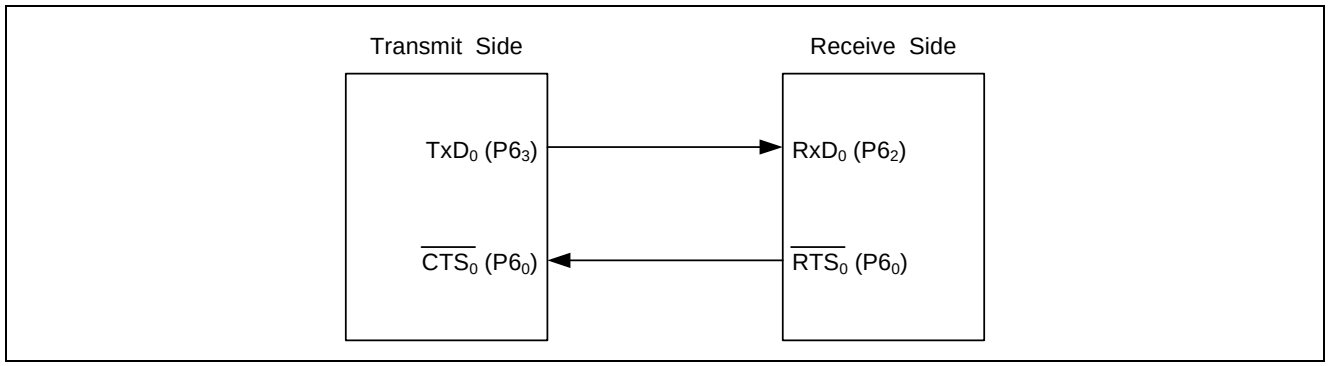


Figure 3. Example of CRC

4.1 Transmit side program

```
/************************************************************/
/*                                                          */
/* M16C/Tiny Series Program Collection                      */
/*                                                          */
/* FILE NAME   : rec05b0005-0102_snd.c                      */
/* CPU         : M16C/29 Group                              */
/* FUNCTION    : CRC Calculation Circuit                    */
/*             (Clock asynchronous serial I/O transfer)     */
/* HISTORY     : 2006.04.13 Ver 1.02                        */
/*                                                          */
/* Copyright (C) 2006. Renesas Technology Corp.             */
/* All right reserved.                                      */
/*                                                          */
/************************************************************/

/************************************************************/
/*    Include File                                          */
/************************************************************/
#include "sfr29.h"    // Special function register header file

/************************************************************/
/*    Function Definition                                   */
/************************************************************/
void init_mcu(void);  // MCU initialize
void init_uart(void); // UART initialize
void init_crc(void);  // CRC initialize
void crc_snd(void);
void wait_10ms(void); // Main clock oscillation stable wait routine

/************************************************************/
/*    Define Label                                          */
/************************************************************/
```

```
#define PRODUCT_TYPE 0    // 29 group: 0   26A group: 1
#define PIN_TYPE 0        // 80 pin: 0     64 pin: 1 (29 group)
                          // 48 pin: 0     42 pin: 1 (26A group)
#define PLL_ON 0          // 0: not use PLL or PM20 is 1
                          // 1: use PLL and PM20 is 0

/*****************************************************/
/*    Define Const                                   */
/*****************************************************/
/* port2_0 - port2_7 data : 0  1  2  3  4  5  6  7  8  9  9 */
unsigned char count_data[11] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xD8,0x80,0x90,0x90};

/*****************************************************/
/*    Define Variable                                */
/*****************************************************/
unsigned short trans_data[4];
unsigned int i;
unsigned char crc_data_l;
unsigned char crc_data_h;

/*****************************************************/
/*    Main Program                                   */
/*****************************************************/
void main(void)
{
   init_mcu();
   init_uart();
   init_crc();
   crc_snd();
}

/*****************************************************/
/*    MCU Initialize Program                         */
/*****************************************************/
void init_mcu(void)
{
   prcr = 0x03;   // Protect register off

   pm20 = 1;      // Specifying wait when accessing SFR (Note)
                  // (Note) This bit can only be rewritten while the PLC07 bit is
                  // "0" (PLL Off). Also, to select a 16MHz or higher PLL clock,
                  // set this bit to "0" (2 wait).

   wait_10ms();   // Waiting for main clock oscillation stable

   cm2 = 0x00;    // System register2 Initialize
   cm1 = 0x20;    // System register1 Xcin-Xcout: high
   cm0 = 0x08;    // System register0 Xcin-Xcout: high

   prcr = 0x00;   // Protect register on

   #if PRODUCT_TYPE      // Product selection: 26A group
```

```
      ifsr2a = 1;         // Interrupt request cause select register2 IFSR2A
                          // <IFSR20>   : Reserved bit (Must be set to "1")
      prcr = 0x04;        // Protect register off
      #if PIN_TYPE        // Port setting
         pacr = 0x01;     // 42pin type
      #else
         pacr = 0x04;     // 48pin type
      #endif
      prcr = 0x00;        // Protect register on
   #else                  // Product selection: 29 group
      ifsr2a = 0;         // Interrupt request cause select register2 IFSR2A
                          // <IFSR20>   : Reserved bit (Must be set to "0")
      prcr = 0x04;        // Protect register off
      #if PIN_TYPE        // Port setting
         pacr = 0x02;     // 64pin type
      #else
         pacr = 0x03;     // 80pin type
      #endif
      prcr = 0x00;        // Protect register on
   #endif
}

/*****************************************************/
/*    Main Clock Oscillation Stable Wait 10ms Routine   */
/*****************************************************/
void wait_10ms(void)
{
   ta0mr = 0x00;     // Set Timer A0 mode register (Timer mode, count source: f1)

   ta0 = 20000-1;    // Setting counter value (10msec @4MHz/2, f1)

   ta0ic = 0x00;     // Clear interrupt request bit

   tabsr = 0x01;     // Timer A0 start counting

   while (ir_ta0ic == 0){   }

   ir_ta0ic = 0;     // Clear interrupt request bit

   tabsr = 0x00;     // Timer A0 stops counting
}

/*****************************************************/
/*    UART Initialize Program                        */
/*****************************************************/
void init_uart(void)
{
   u0mr = 0x45;   // UART0 transmit/receive mode register setting
                  // UART mode transfer data 8 bits long
                  // Internal clock select one stop bit
                  // Parity enabled (odd parity)

   u0c0 = 0x00;   // UART0 transmit/receive control register 0 setting
```

```
                    // CTS function select
                    // CTS/RTS function enabled
                    // TxD0 pin is CMOS output
                    // Transmission data is output at falling edge of transfer clock
                    // and receive data is input at rising edge LSB first

    ucon = 0x01;    // UART transmit/receive control register 2 setting
                    // UART0 transmit interrupt cause is selected to "Transmission
                    // completed (TXEPT=1)"
                    // CTS/RTS shared pin

    u0brg = 129;    // Setting UART0 bit rate generator (Approx 9600bps @20MHz f1)
    u0c1 = 0x01;    // UART transmit/receive control register 1 setting transmit
                    // enabled

    trans_data[0] = 0x78;
    trans_data[1] = 0x56;
    trans_data[2] = 0x34;
    trans_data[3] = 0x12;
}

/*******************************************************/
/*    CRC Initialize Program                           */
/*******************************************************/
void init_crc(void)
{
    #if PLL_ON
        crcsw = 0;              // CRC snoop on write is disabled
        crcsr = 0;              // CRC snoop on read is disabled

        #else
                                // CRC calculation circuit
        crcps = 0;              // CRC mode polynomial selection bit is selected
                                // to "0" (CRC-CCITT)
        crcms = 0;              // CRC mode selection bit is selected to "0" (LSB
                                // first)

        crcsarl = (char)(&u0tb);  // Setting SFR snoop address
        crcsarh = (char)((unsigned long)(&u0tb) >> 8 );

        crcsw = 1;              // CRC snoop on write is enabled
        crcsr = 0;              // CRC snoop on read is disabled

        crcd = 0;               // CRC data register set to "0"

    #endif
}

/*******************************************************/
/*    CRC Send Program                                 */
/*******************************************************/
void crc_snd(void)
{
```

```
    for (i=0;i<4;i++)
    {
       u0tb = trans_data[i];  // Writing transmit data
       while (!ti_u0c1)
       {
                              // Check & wait the status of UART0 transmit buffer
                              // empty flag
       }
    }
    crc_data_l = crcdl;
    crc_data_h = crcdh;
    u0tb = crc_data_l;        // Writing transmit CRC data
    while (!ti_u0c1)
    {
                              // Check & wait the status of UART0 transmit buffer
                              // empty flag
    }
    u0tb = crc_data_h;        // Writing transmit CRC data
    while (!ti_u0c1)
    {
                              // Check & wait the status of UART0 transmit buffer
                              // empty flag
    }
}
```

4.2 Receive side program

```
/************************************************************/
/*                                                          */
/* M16C/Tiny Series Program Collection                      */
/*                                                          */
/* FILE NAME   : rec05b0005-0102_rcv.c                      */
/* CPU         : M16C/29 Group                              */
/* FUNCTION    : CRC Calculation Circuit                    */
/*              (Clock asynchronous serial I/O receive)     */
/* HISTORY     : 2006.04.13 Ver 1.02                        */
/*                                                          */
/* Copyright (C) 2006. Renesas Technology Corp.             */
/* All right reserved.                                      */
/*                                                          */
/************************************************************/

/************************************************************/
/*    Include File                                          */
/************************************************************/
#include "sfr29.h"    // Special function register header file

/************************************************************/
/*    Function Definition                                   */
/************************************************************/
void init_mcu(void);  // MCU initialize
void init_uart(void); // UART initialize
void init_crc(void);  // CRC initialize
void crc_rcv(void);
void wait_10ms(void); // Main clock oscillation stable wait routine

/************************************************************/
/*    Define Label                                          */
/************************************************************/
#define PRODUCT_TYPE 0    // 29 group: 0   26A group: 1
#define PIN_TYPE 0        // 80 pin: 0     64 pin: 1 (29 group)
                          // 48 pin: 0     42 pin: 1 (26A group)
#define PLL_ON 0          // 0: not use PLL or PM20 is 1
                          // 1: use PLL and PM20 is 0

/************************************************************/
/*    Define Const                                          */
/************************************************************/
/* port2_0 - port2_7 data : 0  1  2  3  4  5  6  7  8  9  9 */
unsigned char count_data[11] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xD8,0x80,0x90,0x90};

unsigned short recv_data[6];
unsigned int i;

/************************************************************/
/*    Main Program                                          */
/************************************************************/
```

```
void main(void)
{
   init_mcu();
   init_uart();
   init_crc();
   crc_rcv();
}

/*****************************************************/
/*    MCU Initialize Program                         */
/*****************************************************/
void init_mcu(void)
{
   prcr = 0x03;   // Protect register off

   pm20 = 1;      // Specifying wait when accessing SFR (Note)
                  // (Note) This bit can only be rewritten while the PLC07 bit is
                  // "0" (PLL Off). Also, to select a 16MHz or higher PLL clock,
                  // set this bit to "0" (2 wait).

   wait_10ms();   // For waiting 10ms, initialize Time A0 and start counting.

   cm2 = 0x00;    // System register2 Initialize
   cm0 = 0x08;    // System register0 Xcin-Xcout: high
   cm1 = 0x20;    // System register1 Xcin-Xcout: high

   prcr = 0x00;   // Protect register on

   #if PRODUCT_TYPE       // Product selection: 26A group
      ifsr2a = 1;         // Interrupt request cause select register2 IFSR2A
                          // <IFSR20>   : Reserved bit (Must be set to "1")
      prcr = 0x04;        // Protect register off
      #if PIN_TYPE        // Port setting
         pacr = 0x01;     // 42pin type
      #else
         pacr = 0x04;     // 48pin type
      #endif
      prcr = 0x00;        // Protect register on
   #else                  // Product selection: 29 group
      ifsr2a = 0;         // Interrupt request cause select register2 IFSR2A
                          // <IFSR20>   : Reserved bit (Must be set to "0")
      prcr = 0x04;        // Protect register off
      #if PIN_TYPE        // Port setting
         pacr = 0x02;     // 64pin type
      #else
         pacr = 0x03;     // 80pin type
      #endif
      prcr = 0x00;        // Protect register on
   #endif
}
```

```
/*******************************************************/
/*    Main Clock Oscillation Stable Wait 10ms Routine   */
/*******************************************************/
void wait_10ms(void)
{
   ta0mr = 0x00;     // Set Timer A0 mode register (Timer mode, count source: f1)

   ta0 = 20000-1;    // Setting counter value (10msec @4MHz/2, f1)

   ta0ic = 0x00;     // Clear interrupt request bit

   tabsr = 0x01;     // Timer A0 start counting

   while (ir_ta0ic == 0){   }

   ir_ta0ic = 0;     // Clear interrupt request bit

   tabsr = 0x00;     // Timer A0 stops counting
}

/*******************************************************/
/*    UART Initialize Program                          */
/*******************************************************/
void init_uart(void)
{
   u0mr = 0x45;   // UART0 transmit/receive mode register setting
                  // UART mode transfer data 8 bits long
                  // Internal clock select one stop bit
                  // Parity enabled (odd parity)

   u0c0 = 0x04;   // UART0 transmit/receive control register 0 setting
                  // RTS function select
                  // CTS/RTS function enabled
                  // TxD0 pin is CMOS output
                  // Transmission data is output at falling edge of transfer clock
                  // and reception data is input at rising edge LSB first

   ucon = 0x01;   // UART transmit/receive control register 2 setting
                  // UART0 transmit interrupt cause is selected to "Transmission
                  // completed (TXEPT=1)"
                  // CTS/RTS shared pin

   u0brg = 129;   // Setting UART0 bit rate generator (Approx 9600bps @20MHz f1)

   u0c1 = 0x04;   // UART transmit/receive control register 1 setting Receive
                  // enabled
}

/*******************************************************/
/*    CRC Initialize Program                           */
/*******************************************************/
void init_crc(void)
{
```

```
    #if PLL_ON
       crcsw = 0;             // CRC snoop on write is disabled
       crcsr = 0;             // CRC snoop on read is disabled

    #else
                              // CRC calculation circuit
       crcps = 0;             // CRC mode polynomial selection bit is selected
                              // to "0" (CRC-CCITT)
       crcms = 0;             // CRC mode selection bit is selected to "0" (LSB
                              // first)

       crcsarl = (char)(&u0rb);  // Setting SFR snoop address
       crcsarh = (char)((unsigned long)(&u0rb) >> 8 );

       crcsw = 0;             // CRC snoop on write is disabled
       crcsr = 1;             // CRC snoop on read is enabled

       crcd = 0;              // CRC data register set to "0"

    #endif
}

/*****************************************************/
/*    CRC Receive Program                            */
/*****************************************************/
void crc_rcv(void)
{
    for (i=0;i<6;i++)
    {
       while (!ri_u0c1)
       {
                              // Check & wait the status of UART0 receive complete
                              // flag
       }
       recv_data[i] = u0rbl;  // Reading receive data
    }

    if ( crcd == 0 )
    {
                              // CRC data check CRCD == 0x00?
       while (1)
       {
                              // Normal end loop
       }
    }
    else
    {
       while (1)
       {
                              // Error end loop
       }
    }
}
```

5. Reference

Renesas web-site

http://www.renesas.com/

Inquiry

http://www.renesas.com/inquiry
csc@renesas.com

Hardware manual

M16C/26A Group (M16C/26A, M16C/26T) Hardware Manual Rev.1.00
M16C/29 Group Hardware Manual Rev.1.00
(Use the latest version on the web-site: http://www.renesas.com)

Technical update/Technical news

(Use the latest version on the web-site: http://www.renesas.com)

Revision

Rev.	Issue data	Description	
		Page	Summary
1.00	Dec.16.05	-	First edition issued
1.01	Jan.25.06	-	Sample program modified: define PRODUCT_TYPE
1.02	Apr.14.06	-	Modified function “wait_10ms” in sample program

Keep safety first in your circuit designs!

1. Renesas Technology Corporation 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 Corporation 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 Corporation or a third party.
2. Renesas Technology Corporation 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation for further details on these materials or the products contained therein.