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.

M16C/62P Group

Procedure for Successive Serial I/O Transmission/Reception Using the DMAC

1. Abstract

This application note presents the procedure for successive serial I/O transmission/reception using the DMAC and an example on how to use it.

2. Introduction

The explanation of this issue is applied to the following condition:
Applicable MCU: M16C/62P Group

This program can also be used when operating other microcomputers within the M16C family, provided they have the same SFR (Special Function Registers) as the M16C/62P microcomputers. However, some functions may have been modified.
Refer to the User's Manual for details. Use functions covered in this Application Note only after careful evaluation.

3. Explanation of the Example Procedure

The example procedure selects serial I/O transmission (or reception) for the cause of request to the DMAC, and writes the next data to the transmit buffer (or reads from the receive buffer) at high speed in synchronism with the I/O transmission. This operation is performed successively as many times as the number of DMAC transfers needed.

3.1. Example Connection

Figure 1 shows an example device connection for successive transmission/reception.

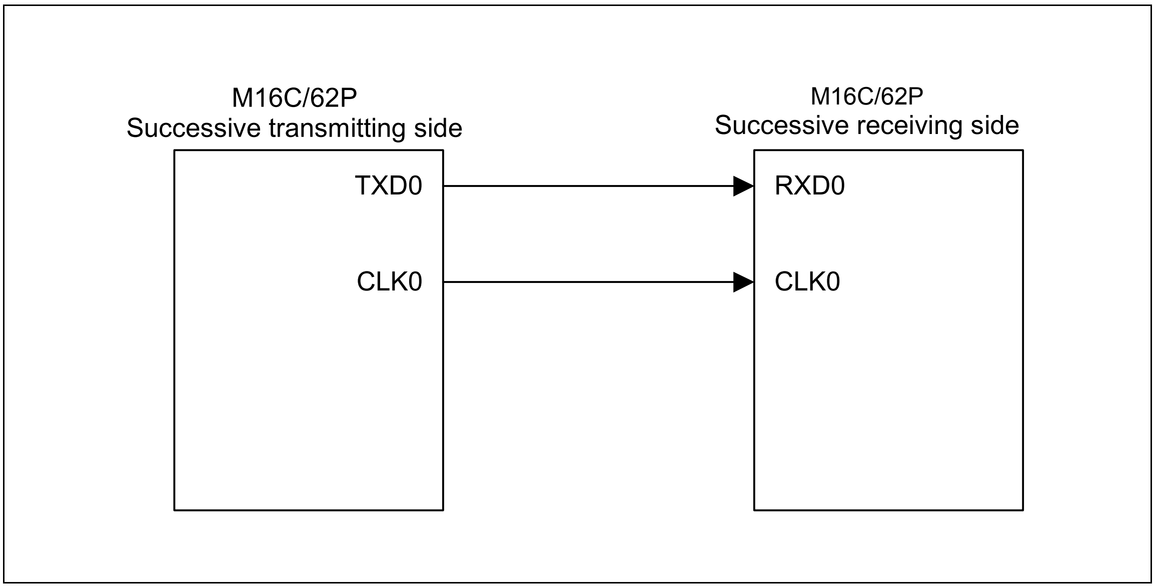


Figure 1. Example Connection for Successive Transmission/Reception

3.2. Setting Up Successive Transmission

The following shows how to set up the device for the case where 8 bytes of data are successively transmitted.

Usage Example:

- System
  VCC1=VCC2=5.0V, XIN=16MHz
- DMAC Setting
  DMA Request Factors=UART0 transfer, Single transfer, Transfer unit = 8 bits, Transfer source address direction=Forward direction, Transfer destination address direction=fixed (U0TB register)
- Serial I/O Setting
  Clock synchronous serial I/O mode, BRG count source = f1SIO, Bit Rates=62500bps (BRG=127), Transmit Interrupt Cause=Transmit buffer empty

Operation:
Specify UART0 transmission for the cause of request to the DMAC and after writing the first byte to the UART0 transmit buffer, transmit the remaining 7 bytes of data successively using a UART0 transmit interrupt request as a trigger. Figure 2 shows successive transmission/reception timing.

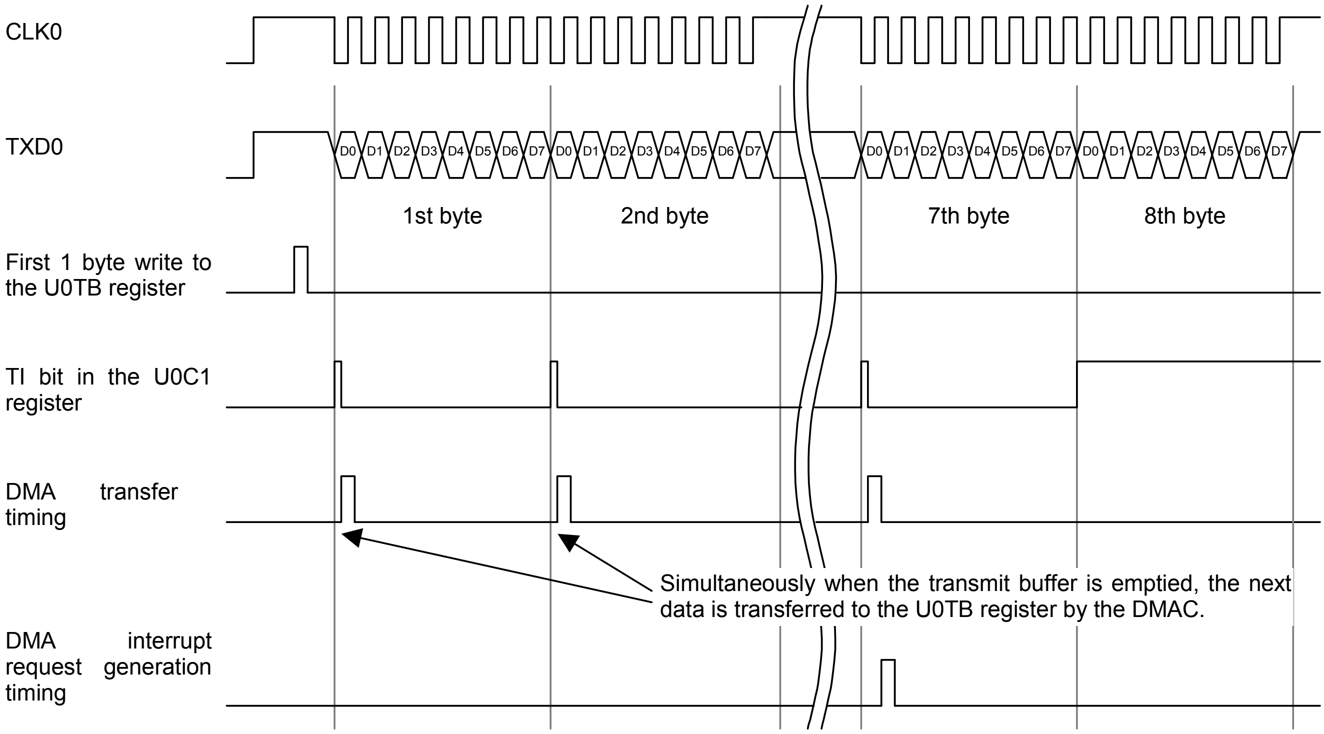
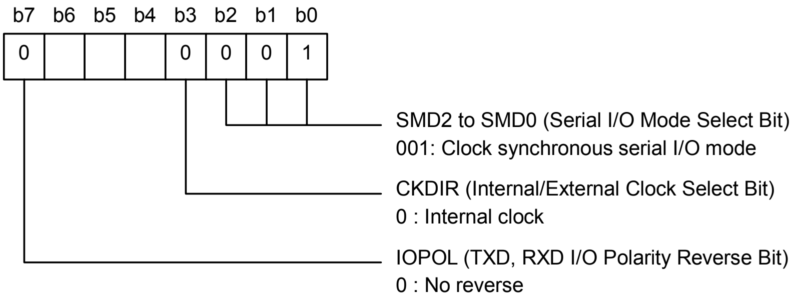

CLK0
TXD0
D0
D1
D2
D3
D4
D5
D6
D7
1st byte
2nd byte
7th byte
8th byte
First 1 byte write to the U0TB register
TI bit in the U0C1 register
DMA transfer timing
Simultaneously when the transmit buffer is emptied, the next data is transferred to the U0TB register by the DMAC.
DMA interrupt request generation timing

Figure 2. Successive Transmission/reception Timing

(1) Setting up the serial I/O

- Set up the U0MR register (UART0 transmit/receive mode register).



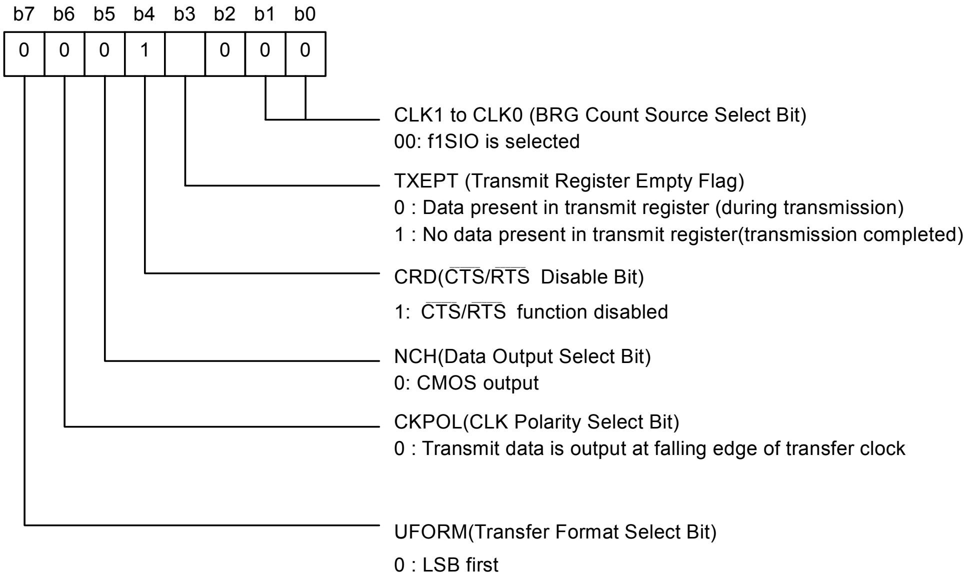
b7	b6	b5	b4	b3	b2	b1	b0
0				0	0	0	1

SMD2 to SMD0 (Serial I/O Mode Select Bit)
001: Clock synchronous serial I/O mode

CKDIR (Internal/External Clock Select Bit)
0 : Internal clock

IOPOL (TXD, RXD I/O Polarity Reverse Bit)
0 : No reverse

- Set up the U0C0 register (UART0 transmit/receive control register 0)



b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1		0	0	0

CLK1 to CLK0 (BRG Count Source Select Bit)
00: f1SIO is selected

TXEPT (Transmit Register Empty Flag)
0 : Data present in transmit register (during transmission)
1 : No data present in transmit register(transmission completed)

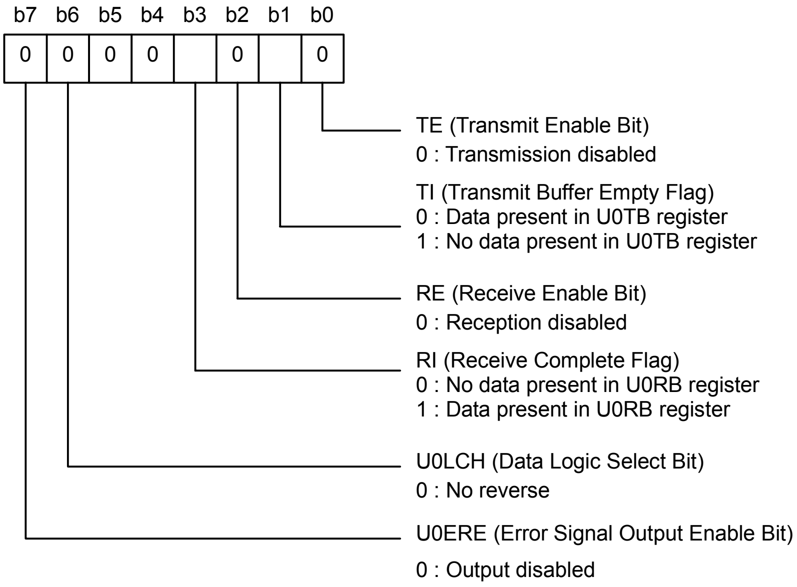
CRD($\overline{\text{CTS}}$/$\overline{\text{RTS}}$ Disable Bit)
1: $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ function disabled

NCH(Data Output Select Bit)
0: CMOS output

CKPOL(CLK Polarity Select Bit)
0 : Transmit data is output at falling edge of transfer clock

UFORM(Transfer Format Select Bit)
0 : LSB first

- Set up the U0C1 register (UART0 transmit/receive control register 1)



b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0		0		0

TE (Transmit Enable Bit)
0 : Transmission disabled

TI (Transmit Buffer Empty Flag)
0 : Data present in U0TB register
1 : No data present in U0TB register

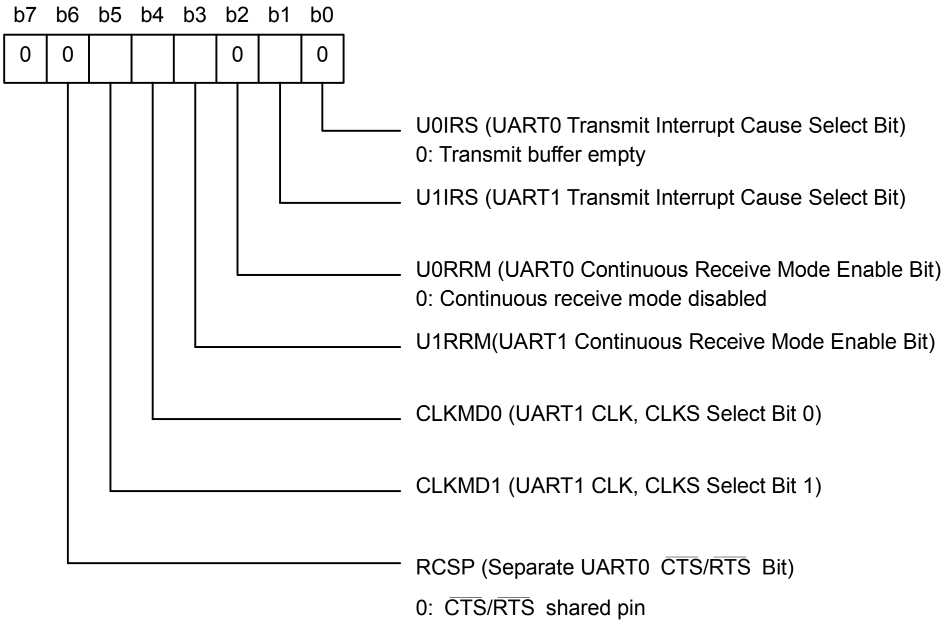
RE (Receive Enable Bit)
0 : Reception disabled

RI (Receive Complete Flag)
0 : No data present in U0RB register
1 : Data present in U0RB register

U0LCH (Data Logic Select Bit)
0 : No reverse

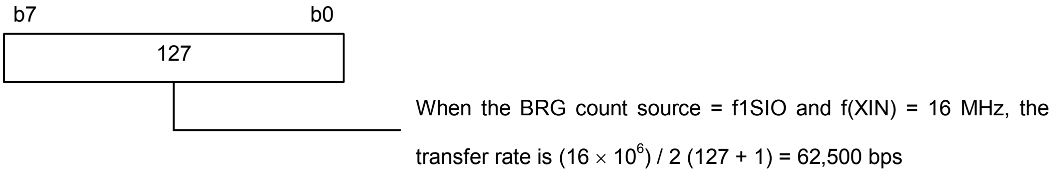
U0ERE (Error Signal Output Enable Bit)
0 : Output disabled

- Set up the UCON register (UART transmit/receive control register 2)

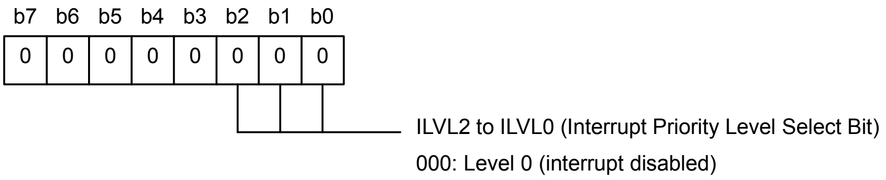

b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 0
U0IRS (UART0 Transmit Interrupt Cause Select Bit)
0: Transmit buffer empty
U1IRS (UART1 Transmit Interrupt Cause Select Bit)
U0RRM (UART0 Continuous Receive Mode Enable Bit)
0: Continuous receive mode disabled
U1RRM(UART1 Continuous Receive Mode Enable Bit)
CLKMD0 (UART1 CLK, CLKS Select Bit 0)
CLKMD1 (UART1 CLK, CLKS Select Bit 1)
RCSP (Separate UART0 $\overline{CTS}$/$\overline{RTS}$ Bit)
0: $\overline{CTS}$/$\overline{RTS}$ shared pin

- Set the U0SMR register (UART0 special mode register), U0SMR2 register (UART0 special mode register 2), U0SMR3 register (UART0 special mode register 3), and U0SMR4 register (UART0 special mode register 4) to “00h”.

- Set up the U0BRG register (UART0 bit rate generation register)

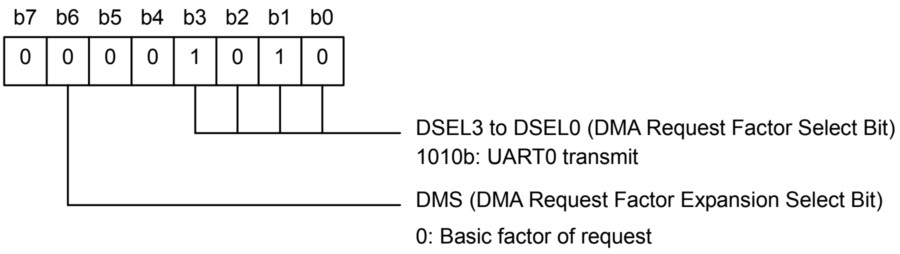

b7 b0
127
When the BRG count source = f1SIO and f(XIN) = 16 MHz, the
transfer rate is (16 × 10^6) / 2 (127 + 1) = 62,500 bps

- Set up the S0TIC register (UART0 transmit interrupt control register)


b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 0 0 0 0 0
ILVL2 to ILVL0 (Interrupt Priority Level Select Bit)
000: Level 0 (interrupt disabled)

(2) Setting up the DMAC

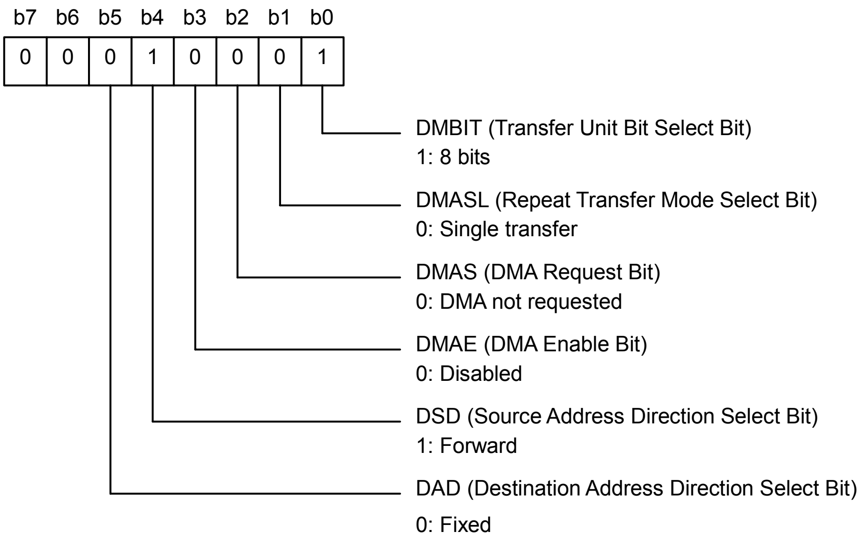
- Set up the DM0SL register (DMA0 request cause select register)



DSEL3 to DSEL0 (DMA Request Factor Select Bit)
1010b: UART0 transmit

DMS (DMA Request Factor Expansion Select Bit)
0: Basic factor of request

- Set up the DM0CON register (DMA0 control register)



DMBIT (Transfer Unit Bit Select Bit)
1: 8 bits

DMASL (Repeat Transfer Mode Select Bit)
0: Single transfer

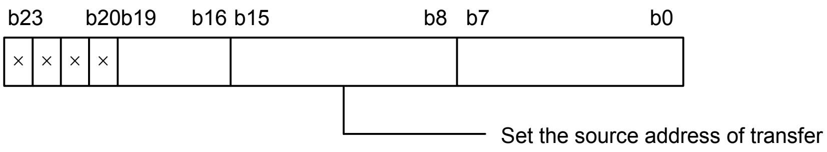
DMAS (DMA Request Bit)
0: DMA not requested

DMAE (DMA Enable Bit)
0: Disabled

DSD (Source Address Direction Select Bit)
1: Forward

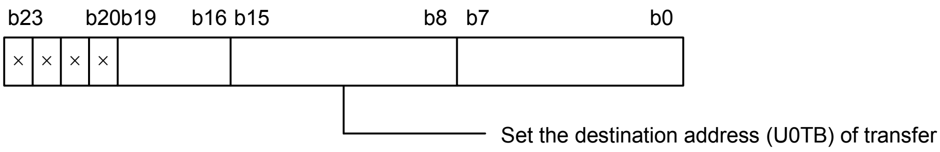
DAD (Destination Address Direction Select Bit)
0: Fixed

- Set up the SAR0 register (DMA0 source pointer)



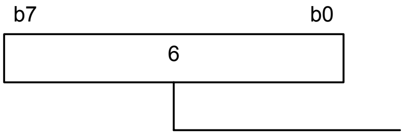
Set the source address of transfer

- Set up the DAR0 register (DMA0 destination pointer)



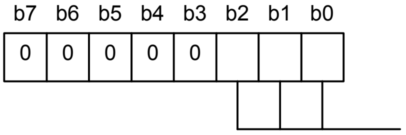
Set the destination address (U0TB) of transfer

- Set up the TCR0 register (DMA0 transfer counter)



Since the first byte of 8-byte successive transmission is written and then transferred to the U0TB register directly (not transferred by the DMAC), set the value “6” here so that 7 bytes will be transferred by DMA.

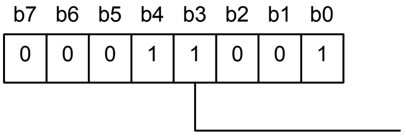
- Set up the DM0IC register (DMA0 interrupt control register)



ILVL2 to ILVL0 (Interrupt Priority Level Select Bit)
Set the interrupt priority level

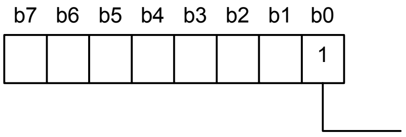
(3) Enables interrupt (I flag =“1”)

(4) Set up the DM0CON register (DMA0 control register) back again (to enable DMA)



DMAE (DMA Enable Bit)
1: Enable

(5) Enables transmit
Set the TE bit in the U0C1 register to “1” (transmit enable)



TE (Transmit Enable Bit)
1: Transmission enabled

(6) Starting successive transmissions
Write the first byte of successive transmit data to the U0TB register. Thereafter, the other bytes of data are successively transmitted by means of the DMAC transfer initiated by a UART0 transmit interrupt request until the count set in the DMA transfer counter expires.

(7) DMAC transfer complete interrupt processing
Set the DMAC transfer complete flag.

3.3. Setting Up Successive Reception

The following shows how to set up the device for the case where 8 bytes of data are successively received.

Usage Example:

- System
  VCC1=VCC2=5.0V, XIN=16MHz
- DMAC Setting
  DMA Request Factors=UART0 reception, Single transfer, Transfer unit = 16 bits (including an error flag), Transfer source address direction=fixed (U0RB register), Transfer destination address direction=Forward direction
- Serial I/O Setting
  Clock synchronous serial I/O mode, External clock (Note 1), Continuous receive mode enabled

Operation:
Specify UART0 reception for the cause of request to the DMAC and after a dummy read of the UART0 receive buffer, receive the data successively using a UART0 receive interrupt as a trigger. Figure 3 shows successive reception timing.

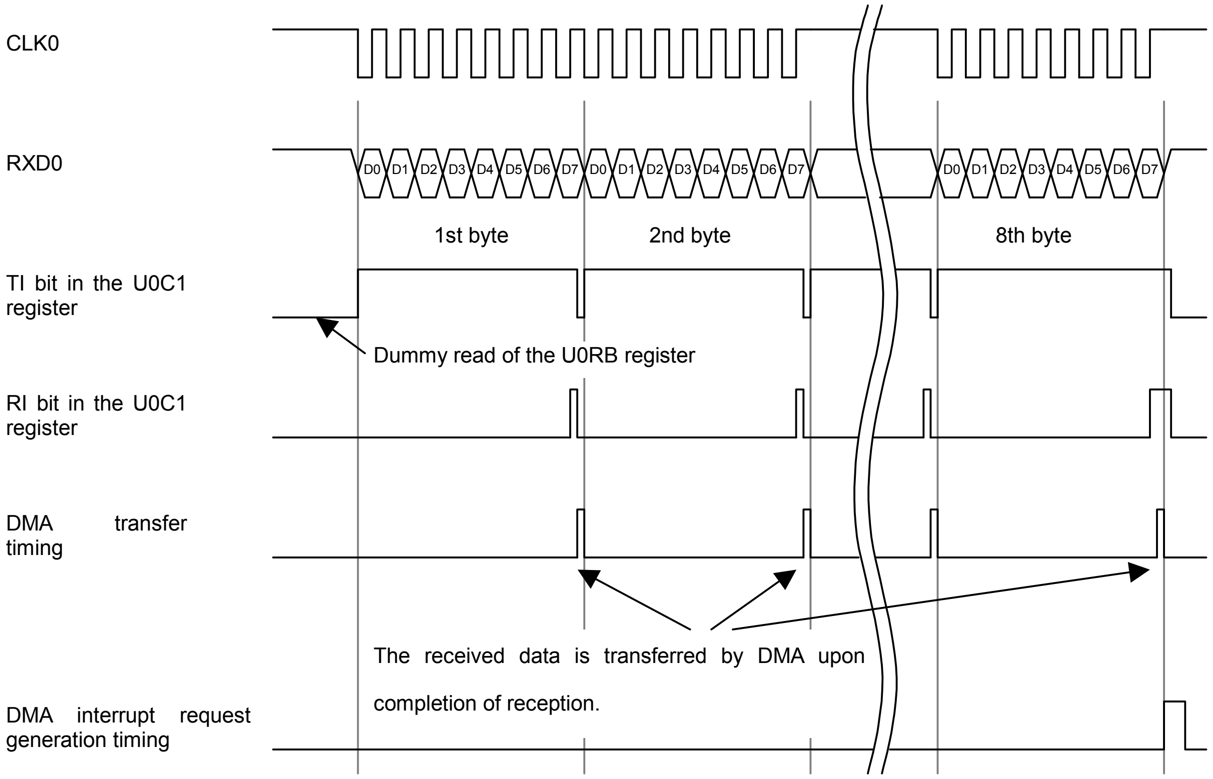
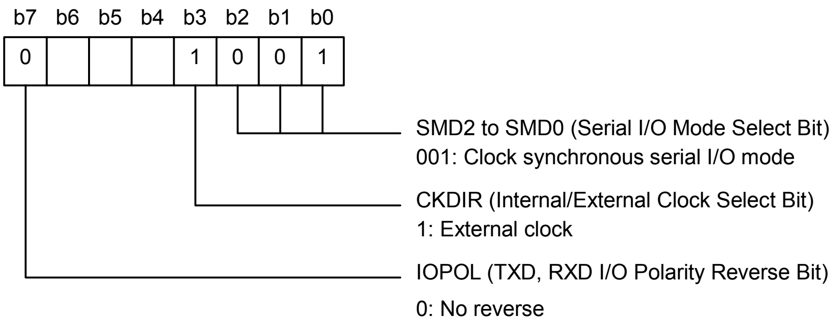

CLK0
RXD0
D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4 D5 D6 D7
1st byte
2nd byte
8th byte
TI bit in the U0C1 register
Dummy read of the U0RB register
RI bit in the U0C1 register
DMA transfer timing
The received data is transferred by DMA upon completion of reception.
DMA interrupt request generation timing

Figure 3. Successive Reception Timing

Note 1:
When the input at the CLK0 pin before data reception is high (or low if the CKPOL bit in the U0C0 register = 1), the conditions described below must be met:
・TE bit in the U0C1 register = 1 (transmission enabled)
・RE bit in the U0C1 register = 1 (reception enabled)
・U0RB register is read

(1) Setting up Serial I/O

- Set up the U0MR register (UART0 transmit/receive mode register



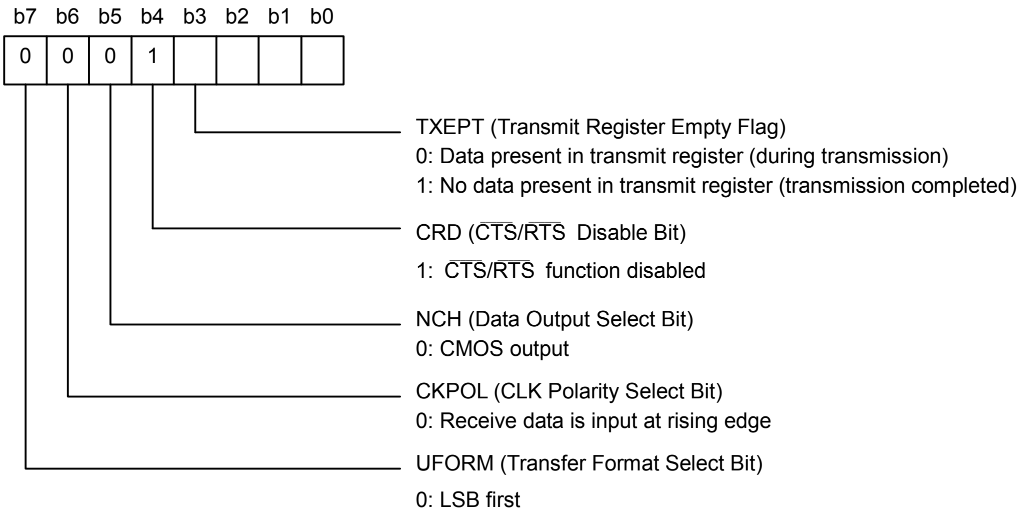
b7	b6	b5	b4	b3	b2	b1	b0
0				1	0	0	1

SMD2 to SMD0 (Serial I/O Mode Select Bit)
001: Clock synchronous serial I/O mode

CKDIR (Internal/External Clock Select Bit)
1: External clock

IOPOL (TXD, RXD I/O Polarity Reverse Bit)
0: No reverse

- Set up the U0C0 register ((UART0 transmit/receive control register 0)



b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1				

TXEPT (Transmit Register Empty Flag)
0: Data present in transmit register (during transmission)
1: No data present in transmit register (transmission completed)

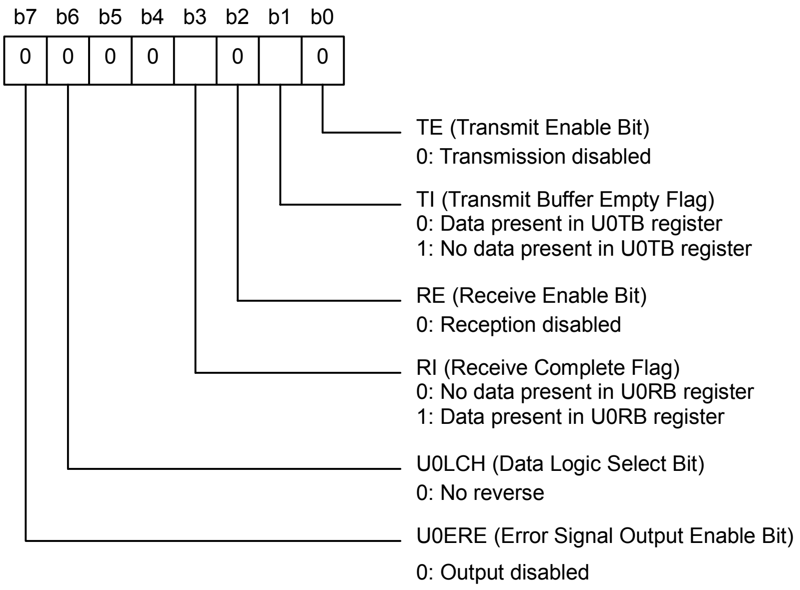
CRD ($\overline{\text{CTS}}$/$\overline{\text{RTS}}$ Disable Bit)
1: $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ function disabled

NCH (Data Output Select Bit)
0: CMOS output

CKPOL (CLK Polarity Select Bit)
0: Receive data is input at rising edge

UFORM (Transfer Format Select Bit)
0: LSB first

- Set up the U0C1 register (UART0 transmit/receive control register 1)



b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0		0		0

TE (Transmit Enable Bit)
0: Transmission disabled

TI (Transmit Buffer Empty Flag)
0: Data present in U0TB register
1: No data present in U0TB register

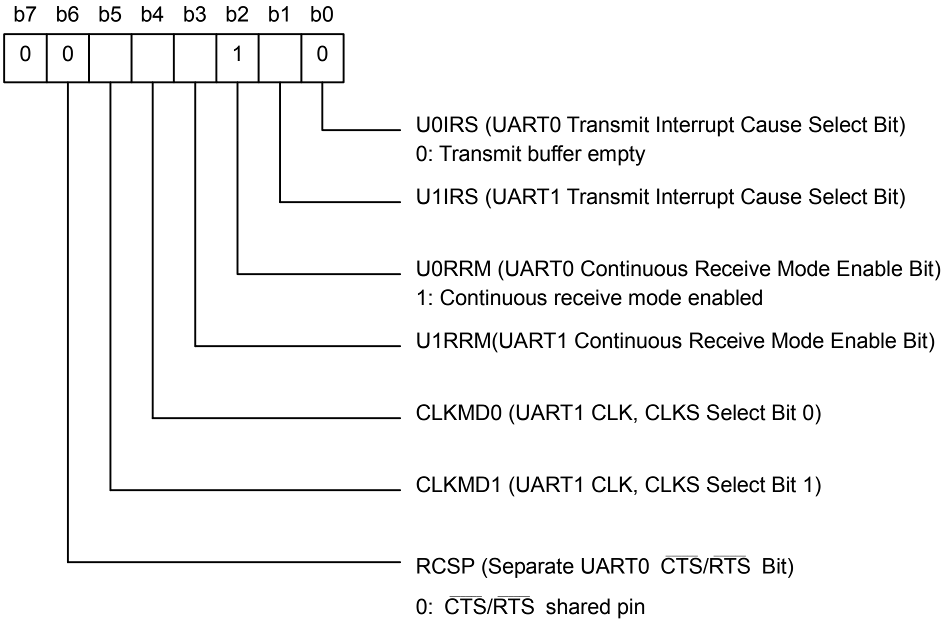
RE (Receive Enable Bit)
0: Reception disabled

RI (Receive Complete Flag)
0: No data present in U0RB register
1: Data present in U0RB register

U0LCH (Data Logic Select Bit)
0: No reverse

U0ERE (Error Signal Output Enable Bit)
0: Output disabled

- Set up the UCON register (UART transmit/receive control register 2)



U0IRS (UART0 Transmit Interrupt Cause Select Bit)
0: Transmit buffer empty

U1IRS (UART1 Transmit Interrupt Cause Select Bit)

U0RRM (UART0 Continuous Receive Mode Enable Bit)
1: Continuous receive mode enabled

U1RRM(UART1 Continuous Receive Mode Enable Bit)

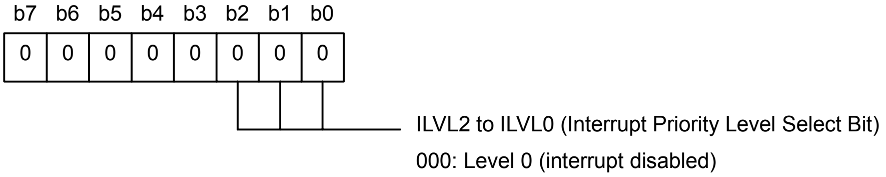
CLKMD0 (UART1 CLK, CLKS Select Bit 0)

CLKMD1 (UART1 CLK, CLKS Select Bit 1)

RCSP (Separate UART0 $\overline{CTS}$/$\overline{RTS}$ Bit)
0: $\overline{CTS}$/$\overline{RTS}$ shared pin

- Set the U0SMR register (UART0 special mode register), U0SMR2 register (UART0 special mode register 2), U0SMR3 register (UART0 special mode register 3), and U0SMR4 register (UART0 special mode register 4) to "00h".

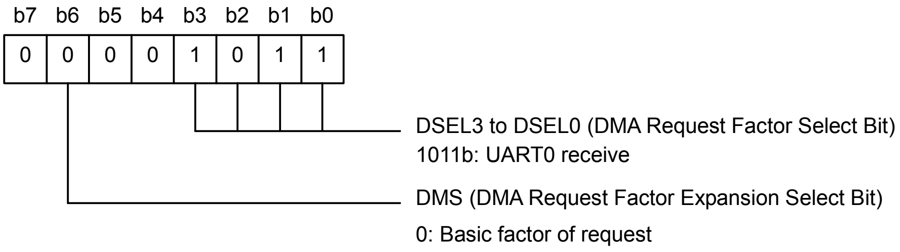
- Set up the S0RIC register (UART0 receive interrupt control register)



ILVL2 to ILVL0 (Interrupt Priority Level Select Bit)
000: Level 0 (interrupt disabled)

(2) Setting up the DMAC

- Set up the DM0SL register (DMA0 request cause select register)

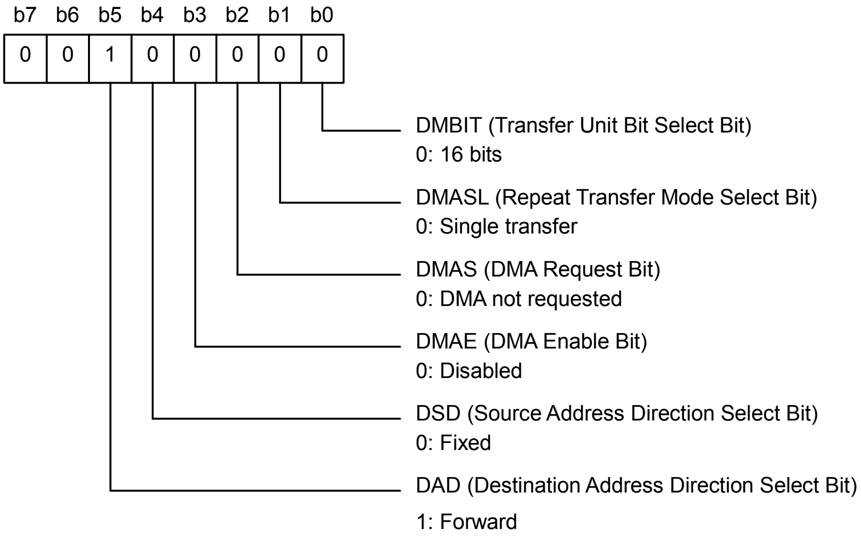


b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	1	0	1	1

DSEL3 to DSEL0 (DMA Request Factor Select Bit)
1011b: UART0 receive

DMS (DMA Request Factor Expansion Select Bit)
0: Basic factor of request

- Set up the DM0CON register (DMA0 control register)



b7	b6	b5	b4	b3	b2	b1	b0
0	0	1	0	0	0	0	0

DMBIT (Transfer Unit Bit Select Bit)
0: 16 bits

DMASL (Repeat Transfer Mode Select Bit)
0: Single transfer

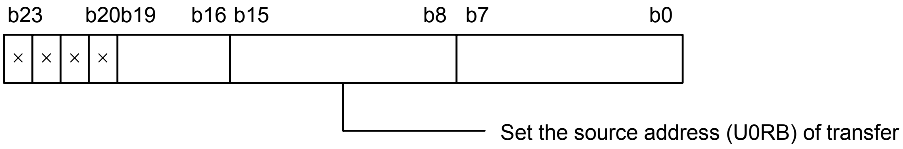
DMAS (DMA Request Bit)
0: DMA not requested

DMAE (DMA Enable Bit)
0: Disabled

DSD (Source Address Direction Select Bit)
0: Fixed

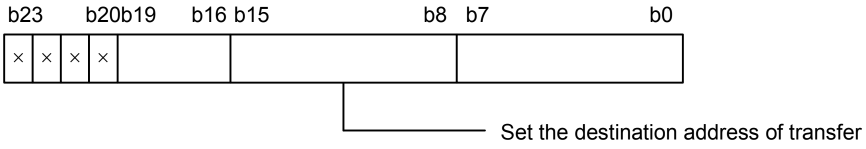
DAD (Destination Address Direction Select Bit)
1: Forward

- Set up the SAR0 register (DMA0 source pointer)



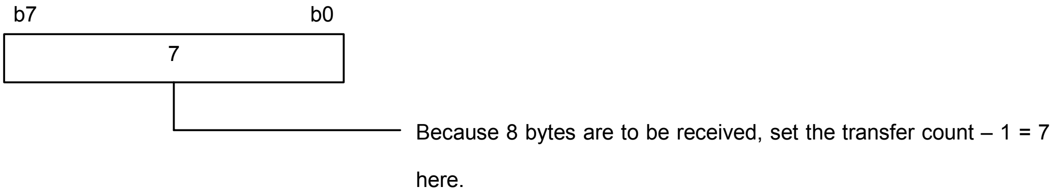
Set the source address (U0RB) of transfer

- Set up the DAR0 register (DMA0 destination pointer)



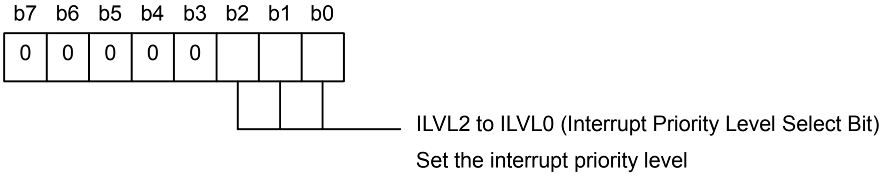
Set the destination address of transfer

- Set up the TCR0 register (DMA0 transfer counter)



Because 8 bytes are to be received, set the transfer count – 1 = 7 here.

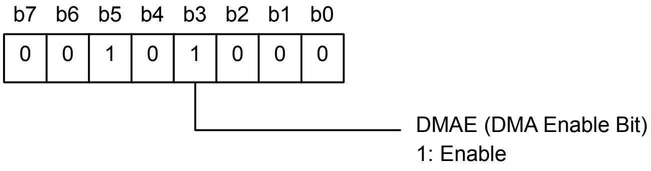
- Set up the DM0IC register (DMA0 interrupt control register)



ILVL2 to ILVL0 (Interrupt Priority Level Select Bit)
Set the interrupt priority level

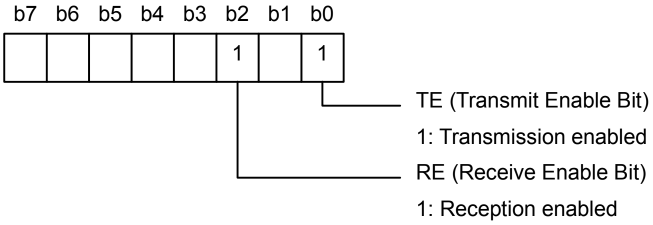
(3) Enables interrupt (I flag =“1”)

(4) Set up the DM0CON register (DMA0 control register) back again (to enable DMA)



DMAE (DMA Enable Bit)
1: Enable

(5) Enables transmit/receive
Set the TE and RE bits in the U0C1 register both to “1”, to enable transmission and reception.



TE (Transmit Enable Bit)
1: Transmission enabled
RE (Receive Enable Bit)
1: Reception enabled

(6) Starting successive reception
Access the U0RB register for dummy read to initiate successive reception.

(7) DMAC transfer complete interrupt processing
Check the received data for errors and, if necessary, reinitialize the serial I/O as error processing.

4. Example of a Sample Program

4.1. Example of a Successive Transmission Program

The following shows an example program for successively transmitting 8 bytes of data using the DMAC. Here, settings are made assuming the DMAC and serial I/O specifications shown below.

- DMAC Specification
  DMA Request Factors=UART0 transfer, Single transfer, Transfer unit = 8 bits, Transfer source address direction=Forward direction, Transfer destination address direction=fixed (U0TB register)
- Serial I/O Specification
  Clock synchronous serial I/O mode, CTS/RTS function disable, BRG count source = f1SIO, Bit Rates=62500bps (XIN=16MHz), Transmit Interrupt Cause=Transmit buffer empty

```
/***********************************************************************/
/*                                                          */
/*  M16C/62 Group Program Collection                             */
/*                                                          */
/*  FILE NAME : rjj05b0543_snd.c                                 */
/*  CPU       : M16C/62P Group                                 */
/*  FUNCTION  : The sample program of the serial I/O continuation     */
/*             transmission using DMAC.                         */
/*  HISTORY   : 2004.11.01 Ver 1.00                             */
/*                                                          */
/*  Copyright (C) 2004. Renesas Technology Corp.                    */
/*  Copyright (C) 2004. Renesas Solutions Corp.                     */
/*  All right reserved.                                        */
/*                                                          */
/***********************************************************************/

/****************************************/
/*    include file                     */
/****************************************/
#include "sfr62p.h"        // Special Function Register Header File

/****************************************/
/*    Function declaration              */
/****************************************/
void sio_init(void);       // Serial-I/O initialize routine
void dma_init(void);       // DMAC initialize routine
void dma0_int(void);       // DMA0 interrupt routine

/****************************************/
/*    Global variable declaration       */
/****************************************/
                        // Transfer data area.
const unsigned char    snd_data[8] =
       {0x01, 0x03, 0x07, 0x0f, 0x1f, 0x3f, 0x7f, 0xff};
unsigned short  dma_flg;   // DMA transmit complate flag. 1=complate.

/****************************************/
/*    Main Program                     */
/****************************************/
void main(void)
{

   sio_init();           // Serial-I/O initialization.

   dma_init();           // DMAC initialization.

   asm("fset i");         // Interrupt enabled

   dm0con |= 0x08;        // Set DM0CON register.
                        //  <DMAE>  : DMA enable
   te_u0c1 = 1;          // U0C1 register re-setup.
                       //  <TE>     : transmit enabled

   u0tb = snd_data[0];

   while(1);

}

/*****************************************/
/*  DMAC initialize routine           */
/*****************************************/
void dma_init(void)
{
   dm0sl = 0x0a;         // Set DM0SL register.
                       //  <DSEL3-0> : UART0 transmit
                       //  <DMS>     : Basic cause

   dm0con = 0x11;        // Set DM0CON register.
                       //  <DMBIT> : 8bit
                       //  <DMASL> : Single transfer
                       //  <DMAE>  : DMA disable
                       //  <DSD>   : Src  address direction=Forward
                       //  <DAD>   : Dest address direction=Fix

                       // Set DMA0 Source pointer address.
   sar0_addr.byte.low  = (char)(&snd_data[1]);
   sar0_addr.byte.mid  = (char)((unsigned long)(&snd_data[1]) >> 8);
   sar0_addr.byte.high = (char)((unsigned long)(&snd_data[1]) >> 16);

                       // Set DMA0 Destination pointer address.
   dar0_addr.byte.low  = (char)(&u0tb);
   dar0_addr.byte.mid  = (char)((unsigned long)(&u0tb) >> 8);
   dar0_addr.byte.high = (char)((unsigned long)(&u0tb) >> 16);

                       // Set DMA0 transfer counter.
   tcr0 = 6;

   dm0ic = 4;           // Set DMA0 interrupt priority-level = 4.

}

/*****************************************/
/*  Serial-I/O initialize routine      */
/*****************************************/
void sio_init(void)
{
   u0mr = 0x01;          // Set U0MR register.
                       //  <SMOD2-0> : Clock-synchronous
                       //  <CKDIR>   : Internal-clock
                       //  <IOPOL>   : No reverse

   u0c0 = 0x18;          // Set U0C0 register.
                       //  <CLK1-0>  : f1SIO
                       //  <CRD>     : CTS/RTS function disabled
                       //  <NCH>     :
                       //  <CKPOL>   : transfer data is output at falling edge
                       //  <UFORM>   : LSB first

   u0c1 = 0x00;          // Set U0C1 register.
                       //  <TE>     : transmit disabled
                       //  <RE>     : receive disabled
                       //  <U0LCH>   : No reverse
                       //  <U0ERE>   : Error signal output disable

   ucon = 0x00;          // Set UCON register.
                       //  <U0IRS> : Transmit interrupt cause = Buffer empty
                       //  <RCSP>  : CTS/RTS shared pin
```

```
    u0smr  = 0x00;          // Set U0SMR register.
    u0smr2 = 0x00;          // Set U0SMR2 register.
    u0smr3 = 0x00;          // Set U0SMR3 register.
                         //  <NODC> : CLK0 is CMOS output
    u0smr4 = 0x00;          // Set U0SMR4 register.

    u0brg = 127;            // Set U0BRG register.
                         //   62500bps(XIN=16MHz)

    s0tic = 0;              // Set UART0 transmit interrupt priority-level = 0.

}

/*****************************************/
/*  DMA0 interrupt routine             */
/*****************************************/
#pragma INTERRUPT/B dma0_int
// "/B" = Instead of saving the registers to the stack,
//       you can switch to the alternate registers.

void dma0_int(void)
{
    dma_flg = 1;            // DMA transmit complate set.
    p10  = 0xff;            // Transmit complate display.
    pd10 = 0xff;

}
```

4.2. Example of a Successive Reception Program

The following shows an example program for successively receiving 8 bytes of data using the DMAC. Here, settings are made assuming the DMAC and serial I/O specifications shown below.

- DMAC Specification
  DMA Request Factors=UART0 reception, Single transfer, Transfer unit = 16 bits (including an error flag), Transfer source address direction=fixed (U0RB register), Transfer destination address direction=Forward direction
- Serial I/O Specification
  Clock synchronous serial I/O mode, External clock, CTS/RTS function disabled, Continuous receive mode enabled

```
/***********************************************************************/
/*                                                          */
/*  M16C/62 Group Program Collection                             */
/*                                                          */
/*  FILE NAME : rjj05b0543_rcv.c                                */
/*  CPU      : M16C/62P Group                                 */
/*  FUNCTION  : The sample program of the serial I/O continuation     */
/*            reception using DMAC.                            */
/*  HISTORY   : 2004.11.01 Ver 1.00                             */
/*                                                          */
/*  Copyright (C) 2004. Renesas Technology Corp.                   */
/*  Copyright (C) 2004. Renesas Solutions Corp.                    */
/*  All right reserved.                                       */
/*                                                          */
/***********************************************************************/

/****************************************/
/*    include file                 */
/****************************************/
#include "sfr62p.h"           // Special Function Register Header File

/****************************************/
/*    Function declaration           */
/****************************************/
void sio_init(void);          // Serial-I/O initialize routine
void dma_init(void);          // DMAC initialize routine
void dma0_int(void);          // DMA0 interrupt routine

/****************************************/
/*    Global variable declaration      */
/****************************************/
unsigned short  rcv_data[8];    // Repeat receive data area
unsigned short  dummy_buf;     // Dummy receive data area

/****************************************/
/*    Main Program                 */
/****************************************/
void main(void)
{

    sio_init();           // Serial-I/O initialization.

    dma_init();           // DMAC initialization.

    asm("fset i");         // Interrupt enabled

    dm0con |= 0x08;        // Set DM0CON register.
                       //  <DMAE>  : DMA enable

    u0c1 |= 0x05;          // U0C1 register re-setup.
                       //  <TE>      : transmit disabled
                       //  <RE>     : receive enabled
   dummy_buf = u0rb;      // Data reception is started after dummy reception..

   while(1);

}

/****************************************/
/*  DMAC initialize routine      */
/****************************************/
void dma_init(void)
{
   dm0sl = 0x0b;         // Set DM0SL register.
                       //  <DSEL3-0> : UART0 receive
                       //  <DMS>    : Basic cause

   dm0con = 0x20;         // Set DM0CON register.
                       //  <DMBIT> : 16bit
                       //  <DMASL> : SIngle transfer
                       //  <DMAE>  : DMA disable
                       //  <DSD>   : Src  address direction=Fix
                       //  <DAD>   : Dest address direction=Forward

                       // Set DMA0 Source pointer address.
   sar0_addr.byte.low  = (char)(&u0rb);
   sar0_addr.byte.mid  = (char)((unsigned long)(&u0rb) >> 8);
   sar0_addr.byte.high = (char)((unsigned long)(&u0rb) >> 16);

                       // Set DMA0 Destination pointer address.
   dar0_addr.byte.low  = (char)(&rcv_data);
   dar0_addr.byte.mid  = (char)((unsigned long)(&rcv_data) >> 8);
   dar0_addr.byte.high = (char)((unsigned long)(&rcv_data) >> 16);

                       // Set DMA0 transfer counter.
   tcr0 = 7;

   dm0ic = 4;            // Set DMA0 interrupt priority-level = 4.

}

/****************************************/
/*  Serial-I/O initialize routine      */
/****************************************/
void sio_init(void)
{
   u0mr = 0x09;          // Set U0MR register.
                       //  <SMOD2-0> : Clock-synchronous
                       //  <CKDIR>   : External-clock
                       //  <IOPOL>   : No reverse

   u0c0 = 0x1c;          // Set U0C0 register.
                       //  <CLK1-0>  :
                       //  <CRD>    : CTS/RTS function disabled
                       //  <NCH>    :
                       //  <CKPOL>   : receive data is input at rising edge
                       //  <UFORM>   : LSB first

   u0c1 = 0x00;          // Set U0C1 register.
                       //  <TE>     : transmit disabled
                       //  <RE>     : receive disabled
                       //  <U0LCH>   : No reverse
                       //  <U0ERE>   : Error signal output disable

   ucon = 0x04;          // Set UCON register.
                       //  <U0RRM> : Continuous receive mode enabled
                       //  <RCSP>  : CTS/RTS shared pin

   u0smr  = 0x00;        // Set U0SMR register.
   u0smr2 = 0x00;         // Set U0SMR2 register.
   u0smr3 = 0x00;         // Set U0SMR3 register.
```

```
    u0smr4 = 0x00;          // Set U0SMR4 register.

    s0ric = 0;             // Set UART0 receive interrupt priority-level = 0.

}

/****************************************/
/*  DMA0 interrupt routine            */
/****************************************/
#pragma INTERRUPT/B dma0_int
// "/B" = Instead of saving the registers to the stack,
//       you can switch to the alternate registers.

void dma0_int(void)
{
                          // Receive data display.
    pd0 = 0xff;             // P0 is an output port.
    pd1 = 0xff;             // P1 is an output port.
    pd2 = 0xff;             // P2 is an output port.
    pd3 = 0xff;             // P3 is an output port.
    pd4 = 0xff;             // P4 is an output port.
    pd5 = 0xff;             // P5 is an output port.
    pd6 = 0xff;             // P6 is an output port.
    pd7 = 0xff;             // P7 is an output port.
    p0  = rcv_data[0];        // Receice data 1st byte display.
    p1  = rcv_data[1];        // Receice data 2ndt byte display.
    p2  = rcv_data[2];        // Receice data 3rd byte display.
    p3  = rcv_data[3];        // Receice data 4th byte display.
    p4  = rcv_data[4];        // Receice data 5th byte display.
    p5  = rcv_data[5];        // Receice data 6th byte display.
    p6  = rcv_data[6];        // Receice data 7th byte display.
    p7  = rcv_data[7];        // Receice data 8th byte display.

    prc2 = 1;
    pd9  = 0xff;                // P9 is an output port.
    pd10 = 0xff;                // p10 is an output port.
    p9   = rcv_data[7];          // Receive data 8th byte display.
    p10  = (char)(rcv_data[7] >> 8);// Error flag display.

}
```

5. Reference

Renesas Technology Corporation Home Page

http://www.renesas.com/

E-mail Support

E-mail: csc@renesas.com

Hardware Manual

M16C/62P Group Hardware Manual Rev.2.30
(Use the latest version on the home page: http://www.renesas.com)

TECHNICAL UPDATE/TECHNICAL NEWS

(Use the latest information on the home page: http://www.renesas.com)

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.01	2005.03.25	-	First edition issued

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.