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.

RENESAS

M16C/60 Series

Clock-Asynchronous Serial I/O (UART)

1. Abstract

The following article introduces Clock-Asynchronous Serial I/O of M16C/60 series.

2. Introduction

The explanation of this issue is applied of the following condition.

Applicable MCU: M16C/60 Series

3.0 Contents

3.1 Outline of Clock-Asynchronous Serial I/O

Clock-Asynchronous Serial I/O requires only one data line. In Clock-Asynchronous Serial I/O, a START bit and STOP bits are added to the beginning and end of the data, and the communication is synchronized character by character, as opposed to bit by bit. (A parity bit can also be added for increased transfer reliability.) The transfer time for each bit is fixed. Transfer speed is expressed in "bps" (bits per second) units, in other words, the number of bits that can be transferred in one second.

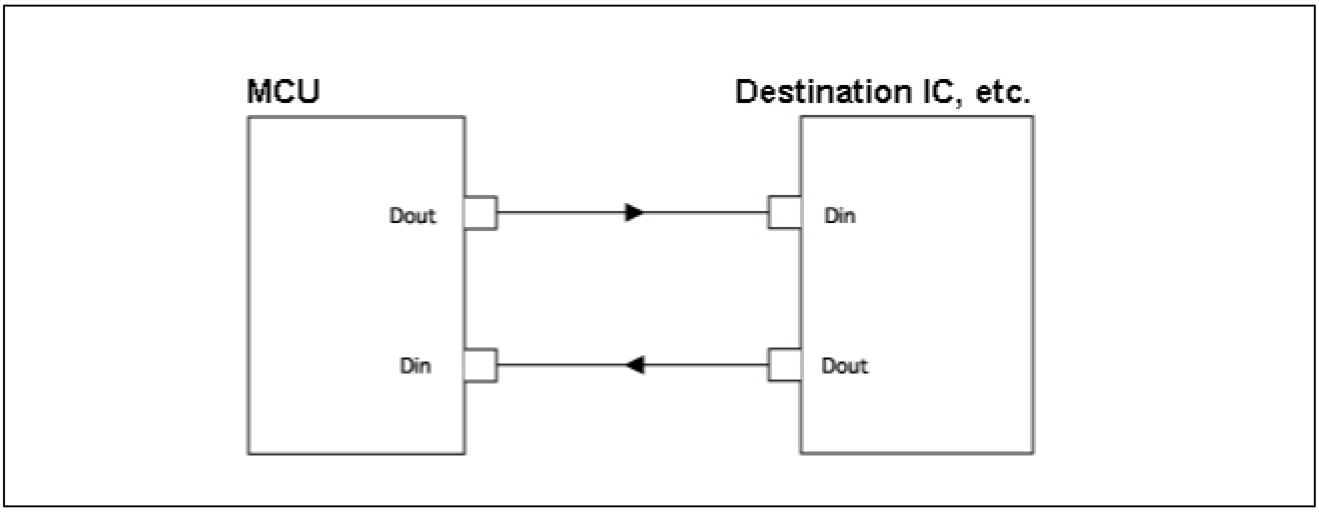


Figure1. Connection Example of Clock-Asynchronous Serial I/O

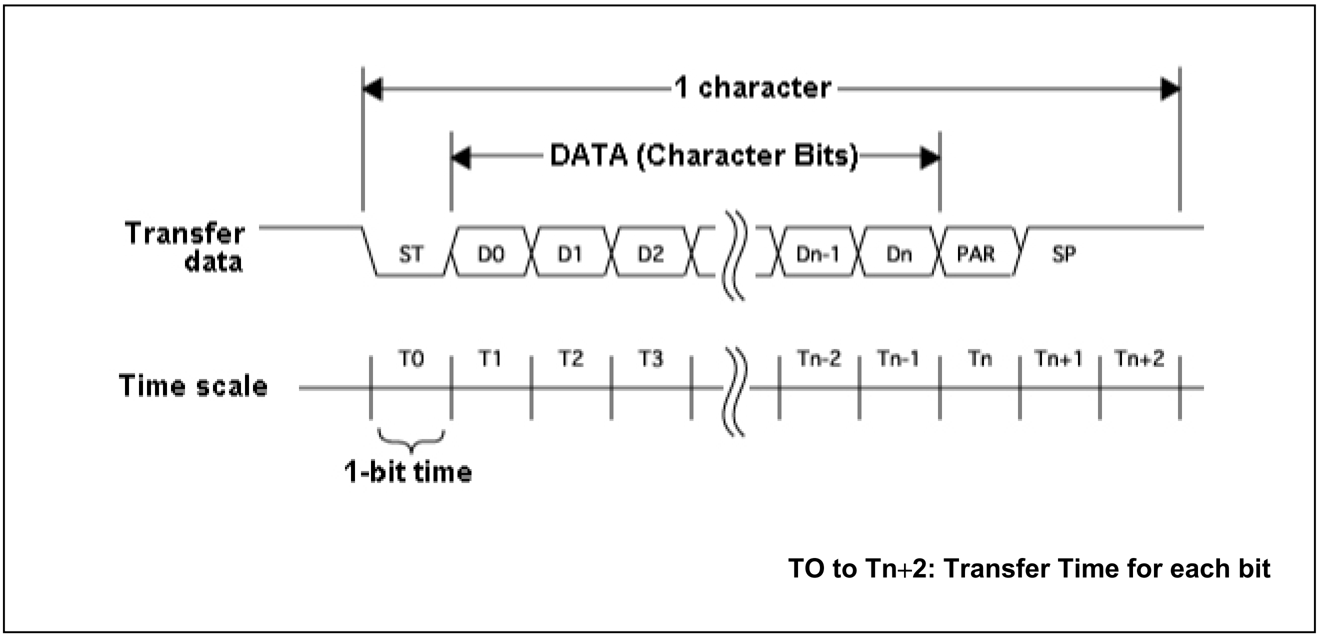


Figure2. Clock-Asynchronous Serial Interface Example

Table1. Transfer Data: Names and Functions

ST START Bit	1 bit of "L" signal added to the head of character bits. Indicates the start of data transfer.
DATA Character Bits	Data to be transferred.
PAR Parity Bit	Parity signal added at the end of character bits. Not added when "parity disable" is specified.
ST STOP Bits	1 or 2 bits of "H" signal added to the end of character bits (or after the parity bit, when enabled). Indicates the end of data transfer.

3.2 Features of M16C/60 Series Clock-Asynchronous Serial I/O

M16C/60 Series Clock-Asynchronous Serial I/O offers sophisticated functions to communicate with various external devices as make software development easy.

- **Selectable Transfer Data Format**
  The following can be selected to determine the transfer data format: character bit length (7 bits/8 bits/9 bits), parity bit (odd/even/none), STOP bit length (1 bit/2 bits).

- **Reversible Logic for Input/Output Data (for applicable serial I/O only)**
  When writing the transmit data and when reading the received data, the data can be reversed.

- **Reversible Polarity for Data I/O Pins (for applicable serial I/O only)**
  The level of the transmit data output pin and the receive data input pin can be reversed.

- **Selectable Transmit Interrupt Factors**
  The generation timing for the transmit interrupt request can be selected from either when the transmit buffer register becomes empty or when the transmit is completed.

- **CTS/RTS Function**
  The CTS function can be used during data transmit or the RTS function can be used during data receive.

- **Sleep Mode**
  The sleep mode can be used to enable the receive operation only when MSB (the highest bit) of the received data is "1".
- **Receive Error Detect Function**
  When the serial I/O fails to receive the data, it sets error flags, allowing the user to recognize errors (overrun, framing, and parity errors).

- **Supports SIM Interface**
  The serial I/O can also be setup to support SIM interface simply by adding the settings.

3.3 Clock-Asynchronous Serial I/O Operation

In M16C/60 Series clock-asynchronous serial I/O, you can set the mode and transfer speed and enable transfers by setting each control register. In transmit operation, transmit can be initiated automatically by writing transmit data to the transmit buffer register. In receive operation, the received data is automatically written to the receive buffer register. Completion of a receive or transmit can be determined by the transmit register empty flag or by the receive complete flag in the control registers.

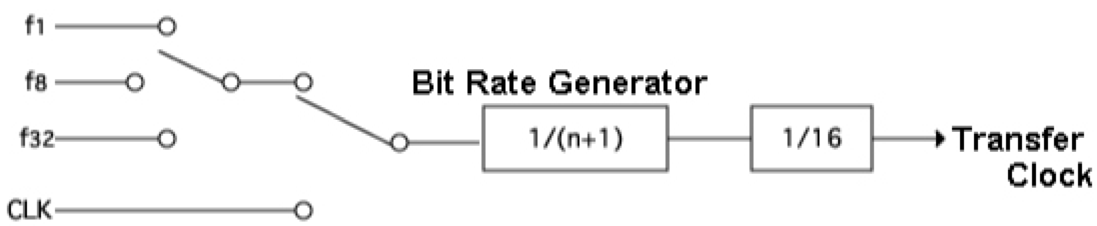
Transfer Data Format Selection

The transfer data format is determined by the settings of the transmit/receive mode register as shown below.

- Character bit length (7 bits/8 bits/9 bits)
- Parity bit (odd/even/none)
- STOP bit length (1 bit/2 bits)

Setting the Transfer Speed

The transfer speed is determined by the count source (f1, f8, f32, or external clock) and the set value (0 to 255) of the bit rate generator. (The output from the bit rate generator is divided by 16 again and becomes the transfer clock.)



Note: Clocks f1, f8 and f32 are the results of the main clock divided into 1/1, 1/8 and 1/32, respectively. n is the value set in the bit rate generator.

Figure3. Bit Rate Generator

Transfer Speed (Baud Rate) Setting Example

Table2. Transfer Speed (Baud Rate) Setting Example

Baud Rate (bps)	Count Source of Bit Rate Generator	System Clock 16MHz		System Clock 7.3728MHz	
		Set Value of Bit Rate Generator: n	Actual Speed (bps)	Set Value of Bit Rate Generator: n	Actual Speed (bps)
9600	f1	103 (67h)	9615	47 (2Fh)	9600
19200	f1	51 (32h)	19231	23 (17h)	19200

Transfer Initiation

To initiate the transmit, set the transmit enable bit then write the transmit data to the transmit buffer register. To receive, set the receive enable bit.

Note: In the M16C/62 Group, TxD2 is N-channel open-drain output and must be pulled up.

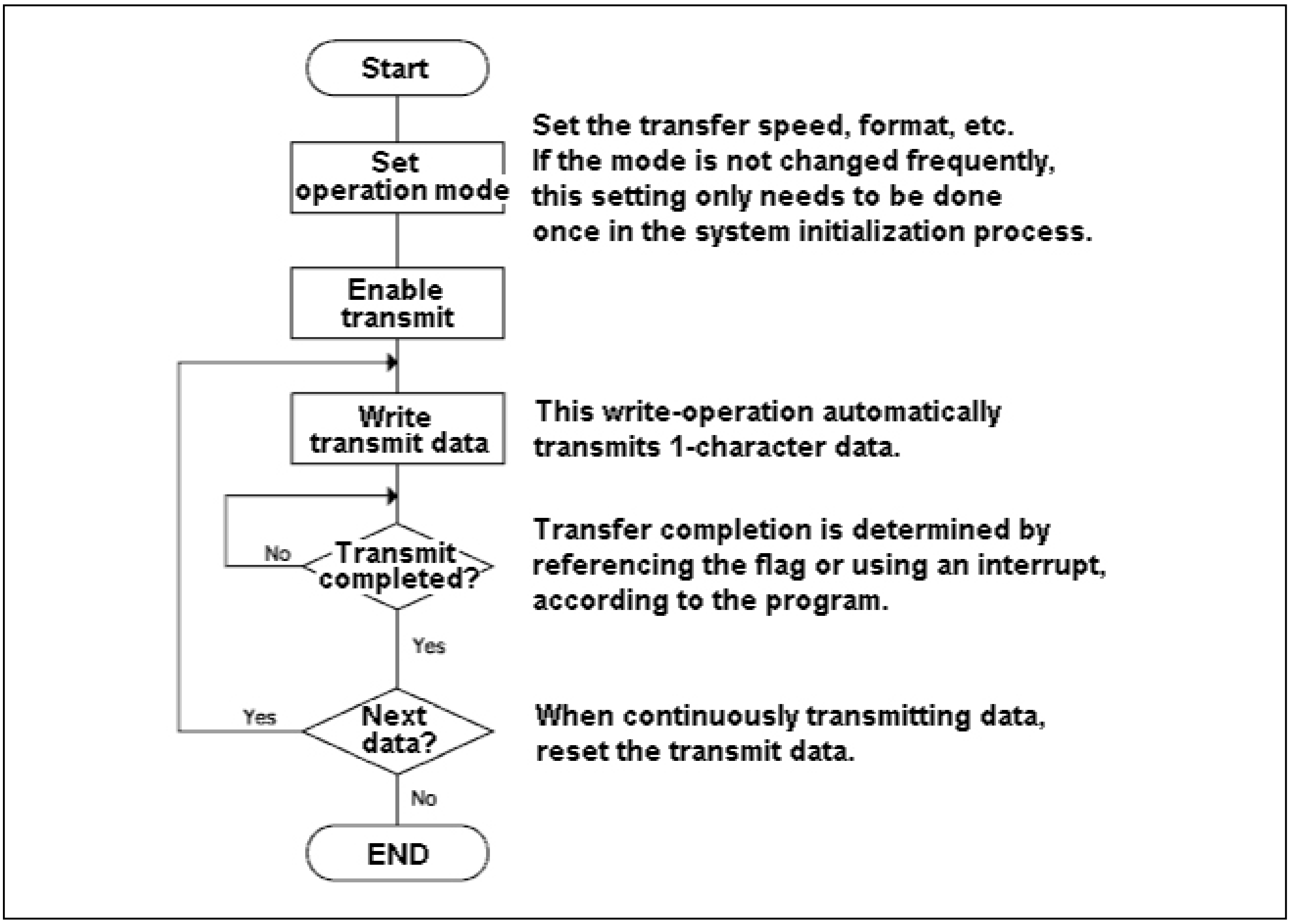


Figure4. Flow Chart (Transmit)

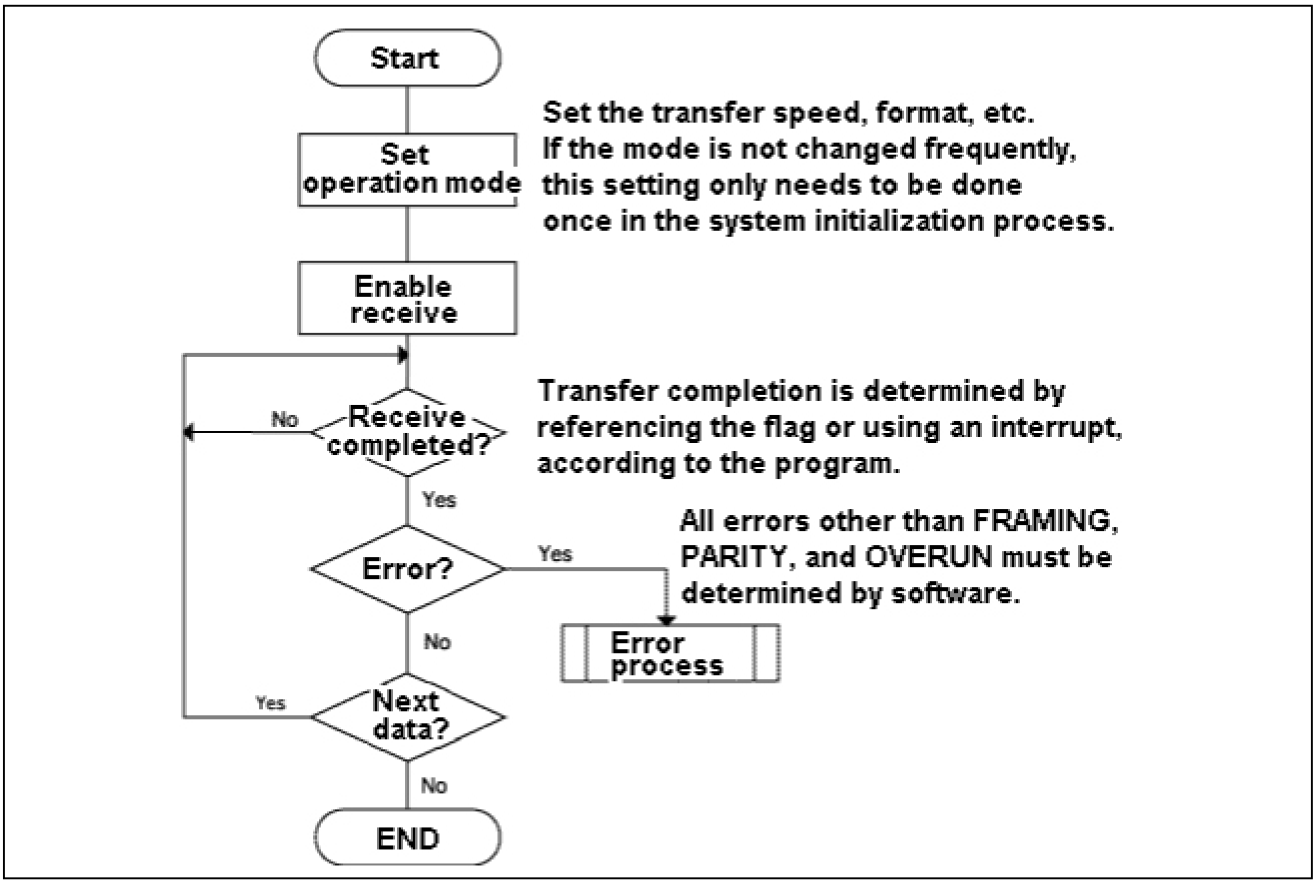


Figure5. Flowchart (Receive)

3.4 Continuous Data Transmit Using an Interrupt

When setting and transmitting continuous data in an interrupt process, you can minimize the void between each data by setting the transmit interrupt factor to "transmit buffer empty".

The data written to the transmit buffer register will be output from the TxD pin after being transferred to the transmit register. When the interrupt factor is set to "transmit completed" by the transmit interrupt cause select bit, the interrupt request is generated when the output of 1-character data is completed. On the other hand, when the factor is set to "transmit buffer empty", the interrupt request will be generated when data is transferred from the transmit buffer register to the transmit register, and the next data can be set in the transmit buffer register before the current transmit is complete.

Operation Timing of Interrupt Factor Setting "Transmit Buffer Empty"

(8 character bits, 2 STOP bits, and parity bit enabled)

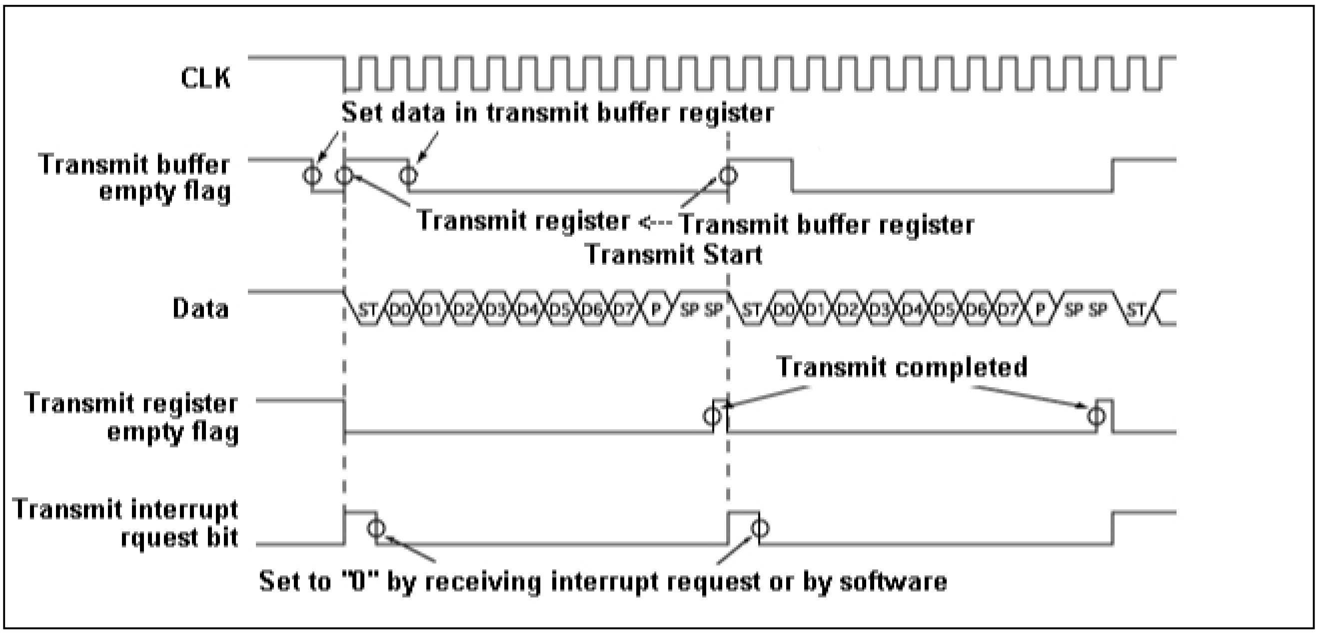


Figure6. Operation Timing Diagram for Interrupt Factor Setting "Transmit Buffer Empty"

Operation Timing of Interrupt Factor Setting "Transmit Completed"

(8 character bits, 2 STOP bits, and parity bit enabled)

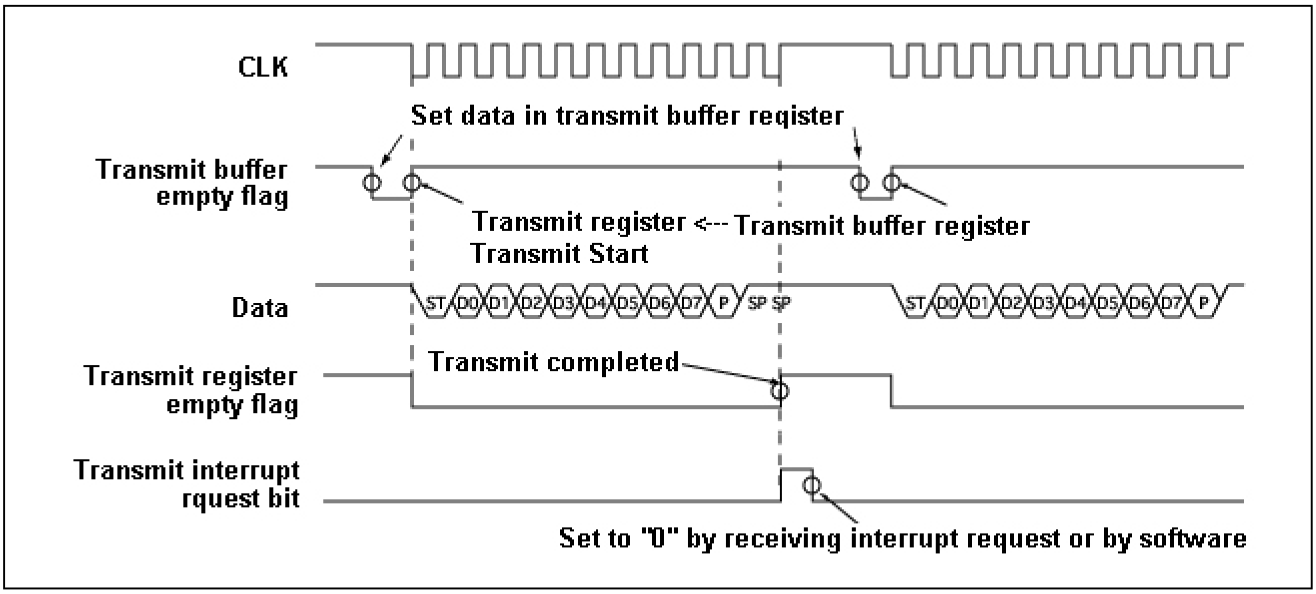


Figure7. Operation Timing Diagram for Interrupt Factor Setting "Transmit Completed"

Transmit Process Example for Using an Interrupt

In a main routine, enable the transmit. In an interrupt process, transmit data.

[Outline]

1. Set each clock-asynchronous serial I/O function.
2. Clear the transmit interrupt request bit to be used, set the interrupt enable flag.
3. To start the transmit, set the transmit enable bit and write the first transmit data to the transmit buffer register.
4. Transmit 1-character data at a time whenever transmit interrupt is generated.
5. Disable the transmit interrupt after writing the last data to the transmit buffer register

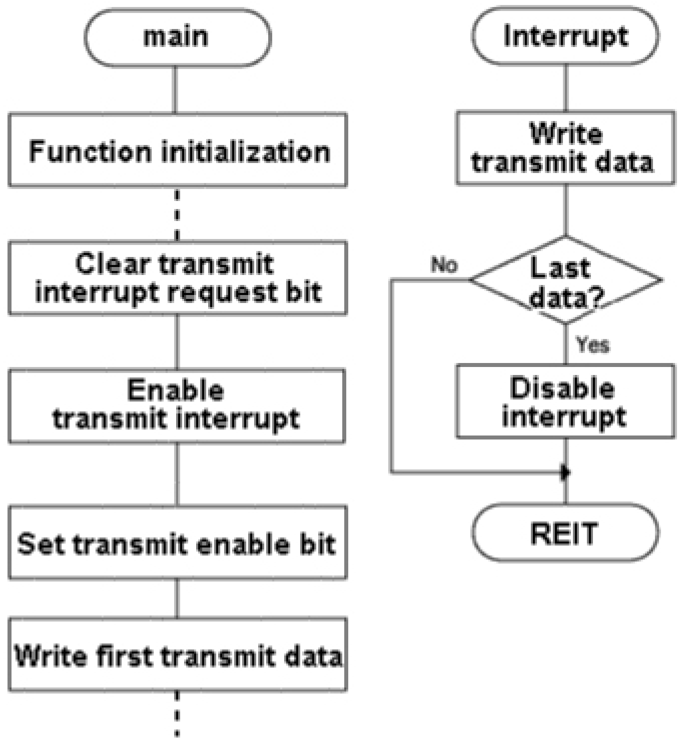


Figure8. General Flowchart

Sample Program

Transmitting data contained in addresses 500h to 503h.

```
                              .
                              .
                              .
;===================== Main Program ====================
MAIN:
   JSR      INIT              ; Go to UART0 initial settings
   FSET     I                 ; Enable interrupt
   MOV.B    #1H,U0C1          ; Enable transmit
   MOV.B    [A0],U0TB         ; Write transmit data
                              .
                              .
                              .
;============== UART0 Function Initial Settings =============
INIT:
   MOV.B      #05H,U0MR       ; Select internal clock, set data length to 8 bits,
                              ; select 1 stop bit, disable parity
   MOV.B    #10H,U0C0         ; Disable CTS/RTS function, select count source f1
   MOV.B    #00H,UCON         ; UART0 transmit interrupt cause = transmit buffer empty
   MOV.B    #67H,U0BRG        ; Transfer speed = f1/104 x 1/16
   MOV.W    #500H,DATA        ; Set position for reading transmit data
   MOV.W    #500H,A0
   MOV.B    #3,CNT            ; Set number of transfers (4-1)
   MOV.B    #1H,S0TIC         ; Set UART0 transmit interrupt priority level
INT_END:
   RTS
                              .
                              .
                              .
;===========    UART0 Transmit Interrupt Process    ===========
S0T_INT:
   PUSH.W A0                  ; Save A0
   INC.B    DATA              ; Set position for reading next transmit data
   MOV.W    DATA,A0           ;
   MOV.B    [A0],U0TB         ; Write next transmit data
   DEC.B    CNT               ; Count down number of transfers
   JNZ      S0T_INT_1         ; Keep interrupt enabled if more data to transmit
                              ;
   MOV.B      #00H,S0TIC      ; If last data, disable transmit interrupt
S0T_INT_1:
   POP.W    A0                ; Restore A0
   REIT
                              .
                              .
                              .
```

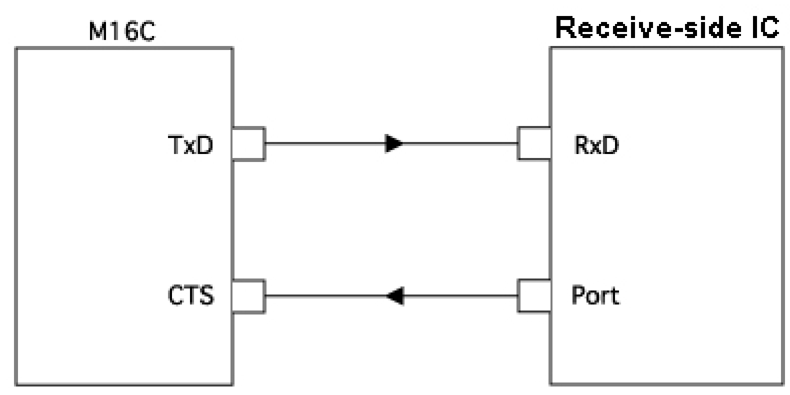
3.5 CTS/RTS Functions

The CTS (Clear To Send) function detects the status of the external IC and controls data transmit accordingly. When the CTS function is selected and the input level of the CTS pin goes to "L", the transmit is started. Because the level of the CTS pin is confirmed at the start of the transmit, if the level goes to "H" during the transmit, the transmit will be stopped after the current transmit is completed.

The RTS (Request To Send) function informs the external IC that the MCU is receive-ready. With the RTS function, the RTS pin will automatically output an "L" level when the MCU is receive-ready. The RTS pin goes to the "H" level at the first falling edge of the transfer clock.

There are three selections available for the CTS/RTS functions: CTS function only, RTS function only, both functions disabled.

*: If both functions are disabled, the CTS/RTS pin can be used as a programmable input/output port.



Note: In the M16C/62 Group, TxD2 is N-channel open-drain output and must be pulled up.

Figure9. Connection Example for CTS Function

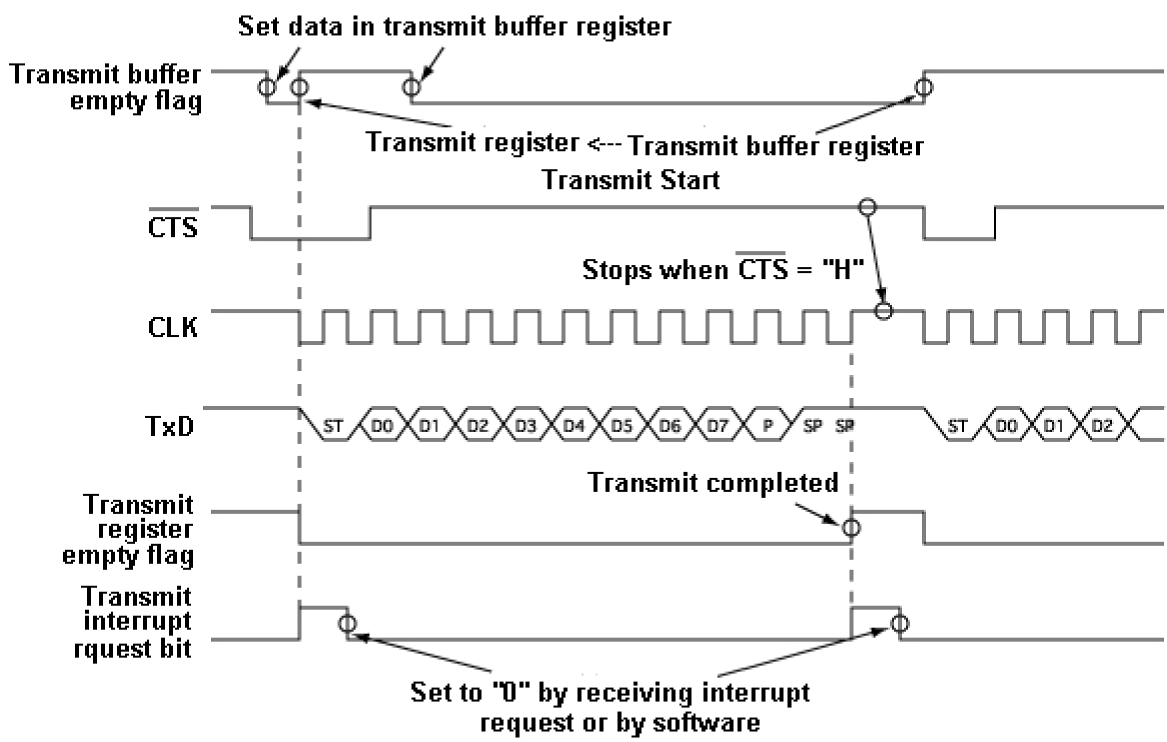
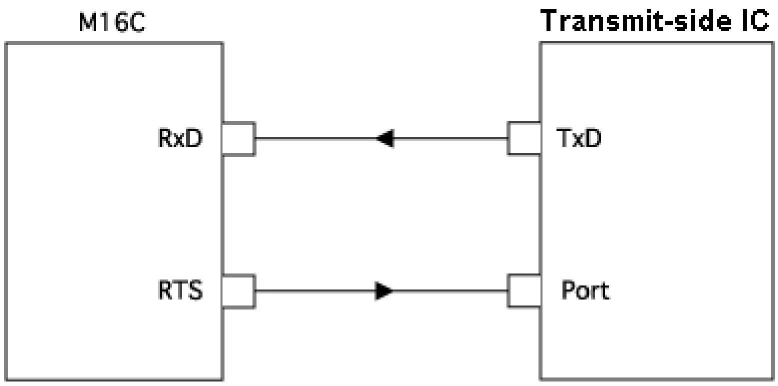
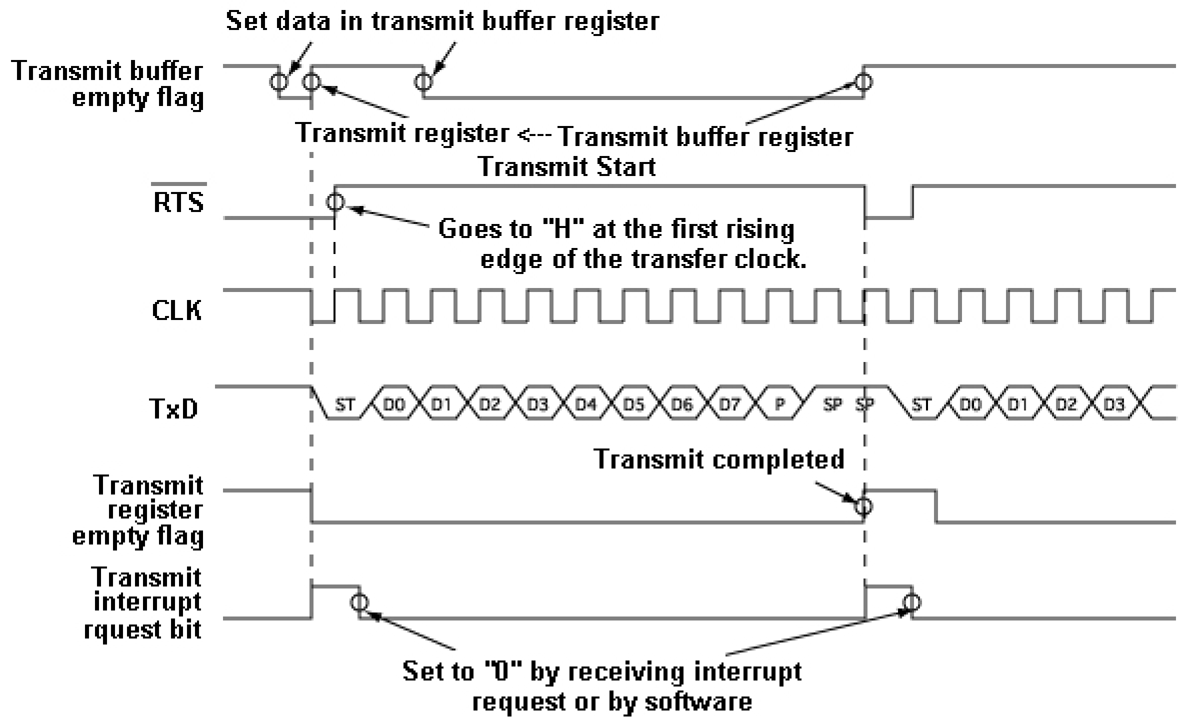


Figure10. Operation Timing Diagram



Note: In the M16C/62 Group, TxD2 is N-channel open-drain output and must be pulled up.

Figure11. Connection Example for RTS Function



Note: The RTS signal goes to "L" when the receive is completed (after detection of the STOP bit), not when the received data is read from the receive buffer register. Therefore, the next data receive may start upon completion of the current data receive. In this case, read the received data in the receive buffer register by the time the next data receive completes (in order to avoid an overrun error).

Figure12. Operation Timing Diagram

Process Example for RTS Function

In a main routine, enable the receive. In an interrupt process, read out data.

[Outline]

1. Select the RTS function and set each clock-asynchronous serial I/O function.
2. Clear the receive interrupt request bit to be used, set the interrupt enable flag.
3. Set the receive enable bit. The RTS pin output level will then go to "L" to inform the transmit-side that the MCU is in the receive-ready status.
4. Read out the received data from the receive buffer register whenever receive interrupt is generated.
5. Disable the receive interrupt after reading the last data from the receive buffer register.

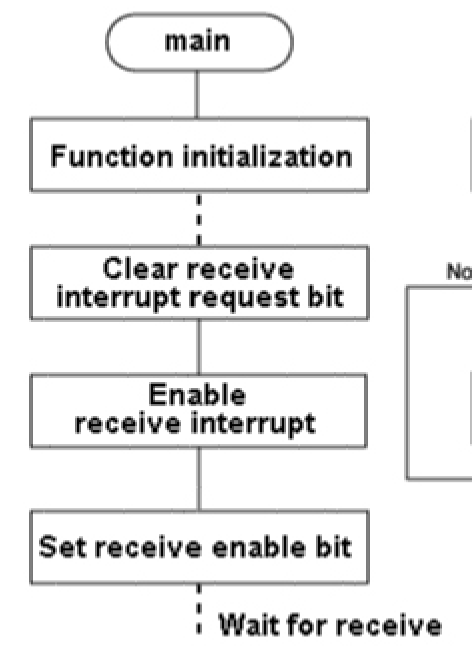


Figure13. General Flowchart

Sample Program

Store received data in addresses 500h to 503h.

```
                                    .
                                    .
                                    .
;===================== Main Program ===================
MAIN:
        JSR         INIT                ; Go to UART0 initial settings
        FSET        I                   ; Enable interrupt
        MOV.B       #4H,U0C1            ; Enable receive
                                    .
                                    .
                                    .
;============== UART0 Function Initial Settings =============
INIT:
        MOV.B       #05H,U0MR           ; Select internal clock, set data length to 8 bits,
                                        ; select 1 stop bit, disable parity
        MOV.B       #04H,U0C0           ; Enable RTS, select count source f1
        MOV.B       #67H,U0BRG          ; Transfer speed = f1/104 x 1/16
        MOV.B       #00H,UCON           ;
        MOV.W       #500H,DATA          ; Set position for storing received data
        MOV.B       #4,CNT              ; Set number of transfers
        MOV.B       #1H,S0RIC           ; Set UART0 receive interrupt priority level
INT_END:
        RTS
                                    .
                                    .
                                    .
;============  UART0 Receive Interrupt Process    ===========
S0R_INT:
        PUSH.W      A0                  ; Save A0
        MOV.W       DATA,A0             ;
        MOV.W       U0RB,[A0]           ; Read received data
        INC.B       DATA                ; Set position for storing next receive data
        DEC.B       CNT                 ; Count down number of transfers
        JNZ         S0R_INT_1           ; Keep interrupt enabled if more data to receive
                                        ;
        MOV.B       #00H,S0RIC          ; If last data, disable receive interrupt
S0R_INT_1:
        POP.W       A0                  ; Restore A0
        REIT
                                    .
                                    .
                                    .
```

3.6 Sleep Function

This function is used to enable transfers to a specified MCU among the multiple MCUs connected using clock-asynchronous serial I/O. Once this function is selected, data with "1" as the highest bit (MSB) will be received, data with "0" as MSB will not be received. (The contents of the receive register will not be transferred to the receive buffer register.)

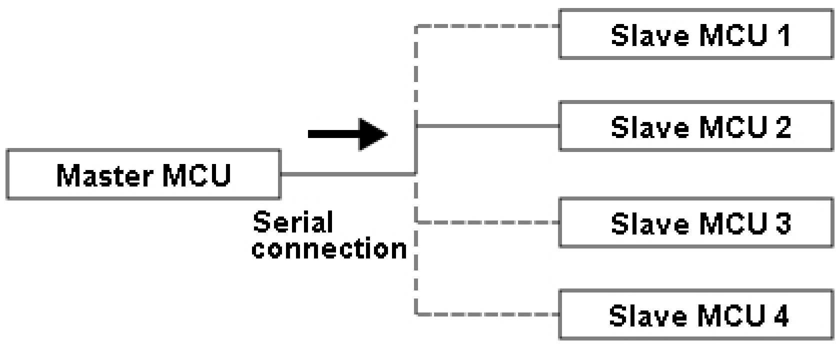


Figure14. Transferring to a specified MCU

3.7 How to Recover from an Error

Recovery procedures, such as initialization, must be executed if an error occurs during clock-asynchronous serial I/O operations. In addition, after recovering from the error, it may be necessary to execute the resend process or resend request process with software.

Detection of Receive Error

In M16C/60 Series MCUs, overrun, framing, and parity errors can be detected during clock-asynchronous serial I/O. Each corresponding error flag goes "1", as well as the error sum flag.

Overrun Error

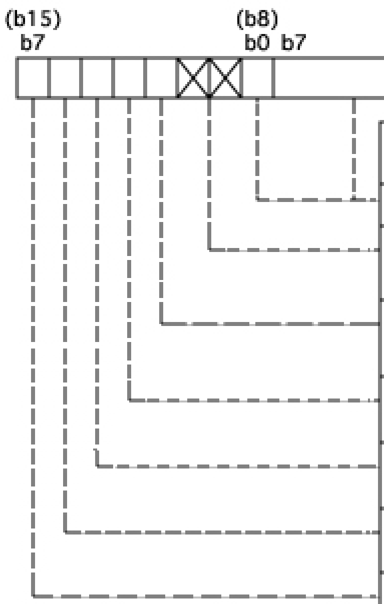
An overrun error occurs when new data is lined up to the receive register before the previously received data is read out from the receive buffer register. When this error occurs, the last received data is stored in the receive buffer register. Also, the receive interrupt request bit is not set to "1". An overrun error causes the overrun error flag to go to "1".

Framing Error

A framing error occurs when the specified number of STOP bits could not be received. A framing error causes the framing error flag to go to "1".

Parity Error

By enabling parity to increase data transfer reliability, 1 bit of parity signal is added to the character bits during the transmit, so that the total of 1s in the character bits and parity bit turns into even number or odd number. During the receive operation, if the number of 1s does not match the specified number (even/odd), a parity error occurs. A parity error causes the parity error flag to go to "1".



Bit Symbol	Bit Name	Function
–	–	Received Data
Nothing is assigned. Write "0" when writing to this bit, "0" is read out when reading from this bit.		
Invalid in UART. Write "0" when writing to this bit, "0" is read out when reading from this bit.		
OER	Overrun Error Flag	0: No error 1: Error detected
FER	Framing Error Flag	0: No error 1: Error detected
PER	Parity Error Flag	0: No error 1: Error detected
SUM	Error Sum Flag	0: No error 1: Error detected

Note: The framing error flag and parity error flag are set to "0" when the lower byte of the receive buffer register is read out. Always read out the receive buffer register in 16-bit units.

Figure15. Error Flag Configuration

Recovery from Error

Execute recovery process according to the following flowchart. Any changes in this setup order may result in MCU malfunction.

All errors other than overrun, framing, and parity errors must be detected through software.

Table3. Recovery from Error

Cause of Error	Error Determination	Recovery Method
Overrun Error Framing Error Parity Error	Reference error flag	Initialization Resend request process
Other (abnormal data, etc)	Data determined by software	Resend process/Resend request process

Recovery Process (Initialization, etc.): Receive

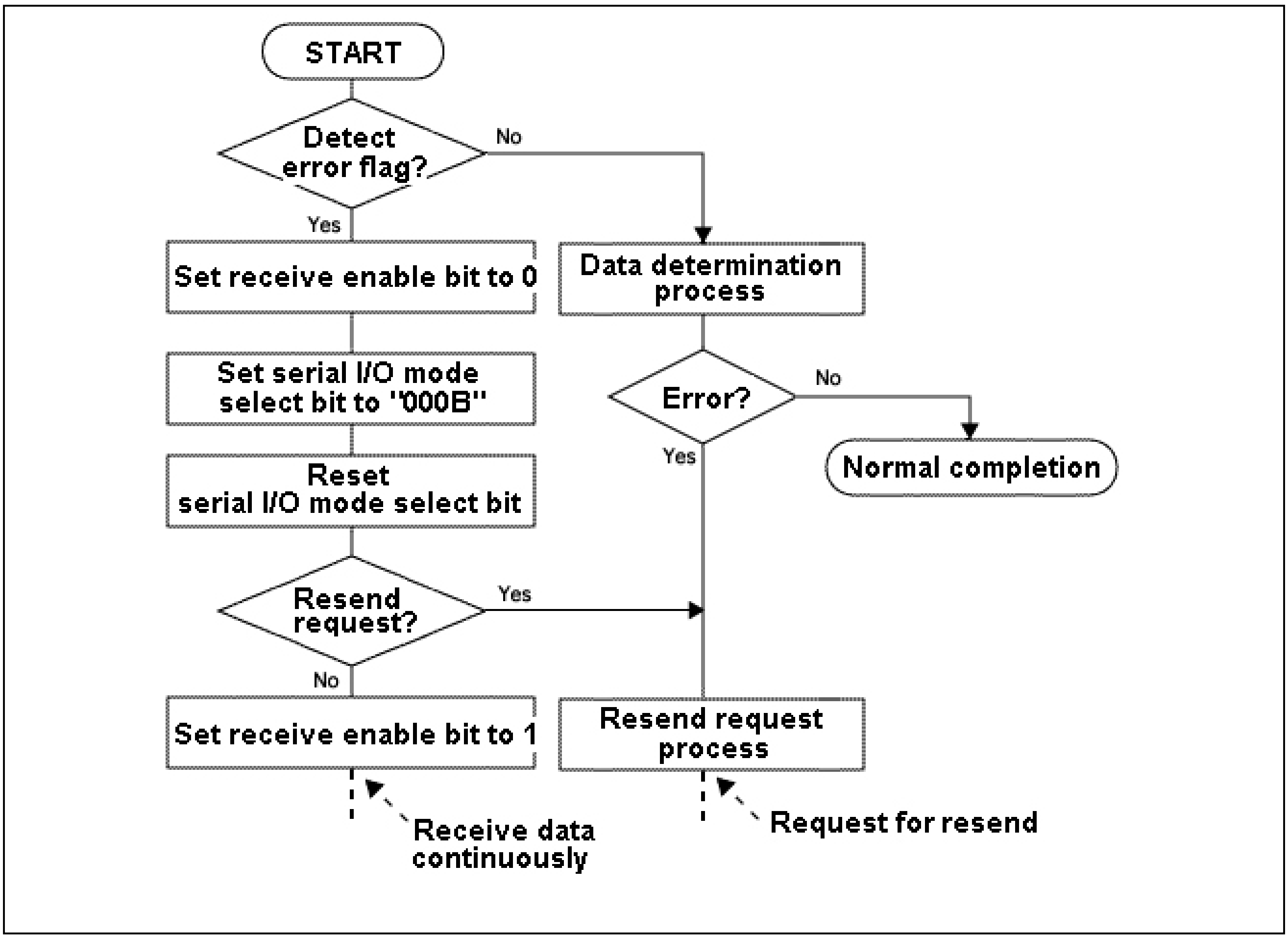


Figure16. Recovery Process (Initialization, etc.) Flowchart (Receive)

Recovery Process (Initialization, etc.): Transmit

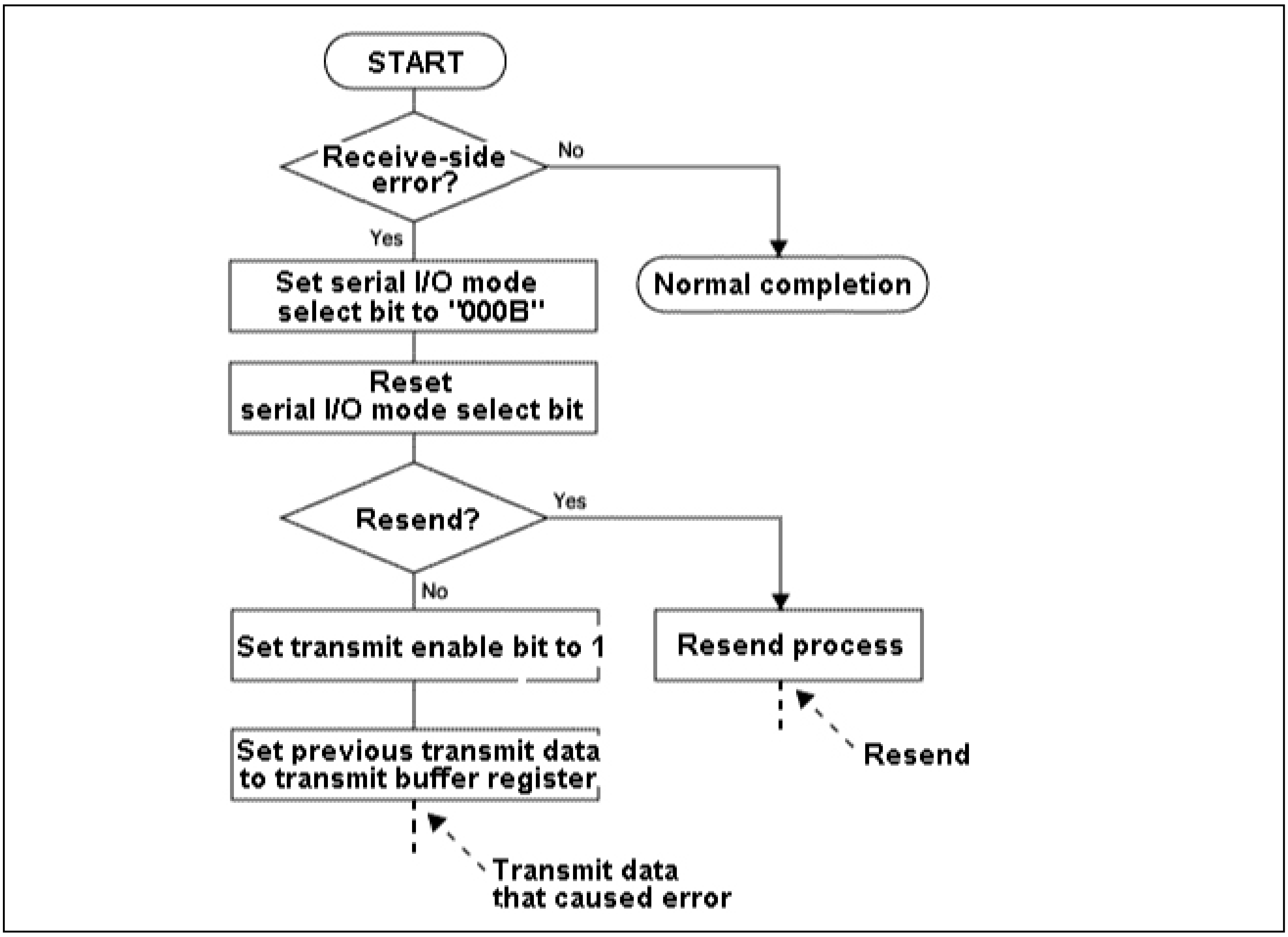


Figure17. Recovery Process (Initialization, etc.) Flowchart (Transmit)

3.8 PC Communications using RS-232C

RS-232C is a standard interface approved by the Electronic Industries Association (EIA) for serial data transfers between Data Communications Equipment (DCE) and Data Terminal Equipment (DTE). This interface enables data transfers between the M16C/60 Series clock-asynchronous serial I/O and the user's PC.

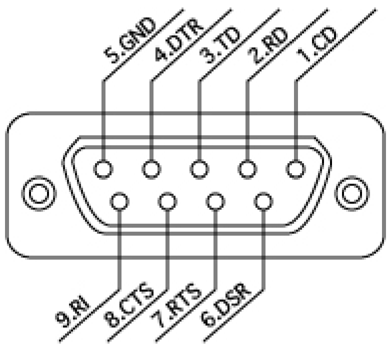


Figure18. PC-side Serial Port Pin Assignments (9-pin version)

Table4. PC-side Serial Port Pin Assignments (9-pin version)

Pin No.	Symbol	Name
1	CD	(Data Channel Receive) Carrier Detect
2	RD	Receive Data
3	TD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indication

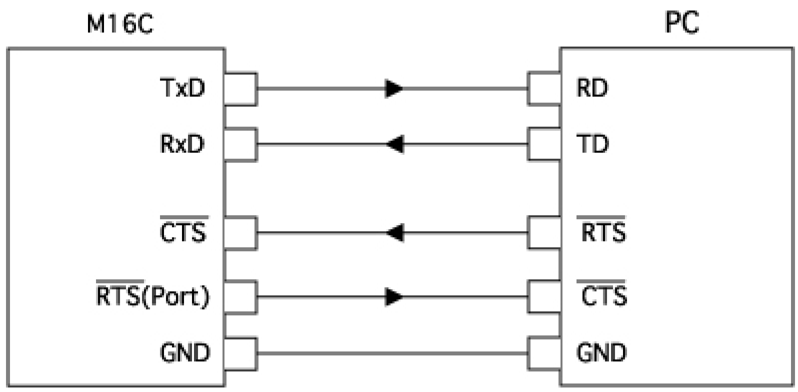


Figure19. M16C - PC Interface Example

Application Example: Downloading a program using RS-232C

1. Send program data from PC to M16C using RS-232C.
2. Save program data in M16C RAM.
3. Execute program after receiving all data.

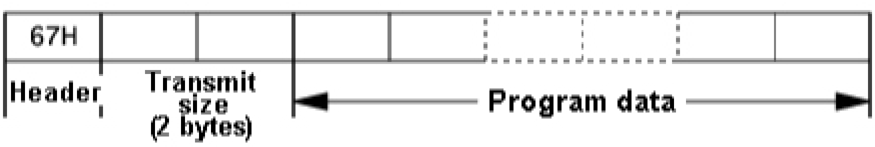


Figure20. Transfer format for data to be transmitted from PC

[Outline]

1. Set each clock-asynchronous serial I/O function.
2. Clear the receive interrupt request bit to be used, set the interrupt enable flag.
3. Set the receive enable bit and wait for the data sent from the PC.
4. Confirm that the first received data is the header (67h). If it is not the header, ignore the received data and wait for the header.
5. After receiving the header, receive the data-size of the program (2 bytes).
6. After receiving the data-size, save the actual program data sent from the PC in the RAM area, in the correct order.
7. After receiving all program data indicated by the data-size, execute the program.

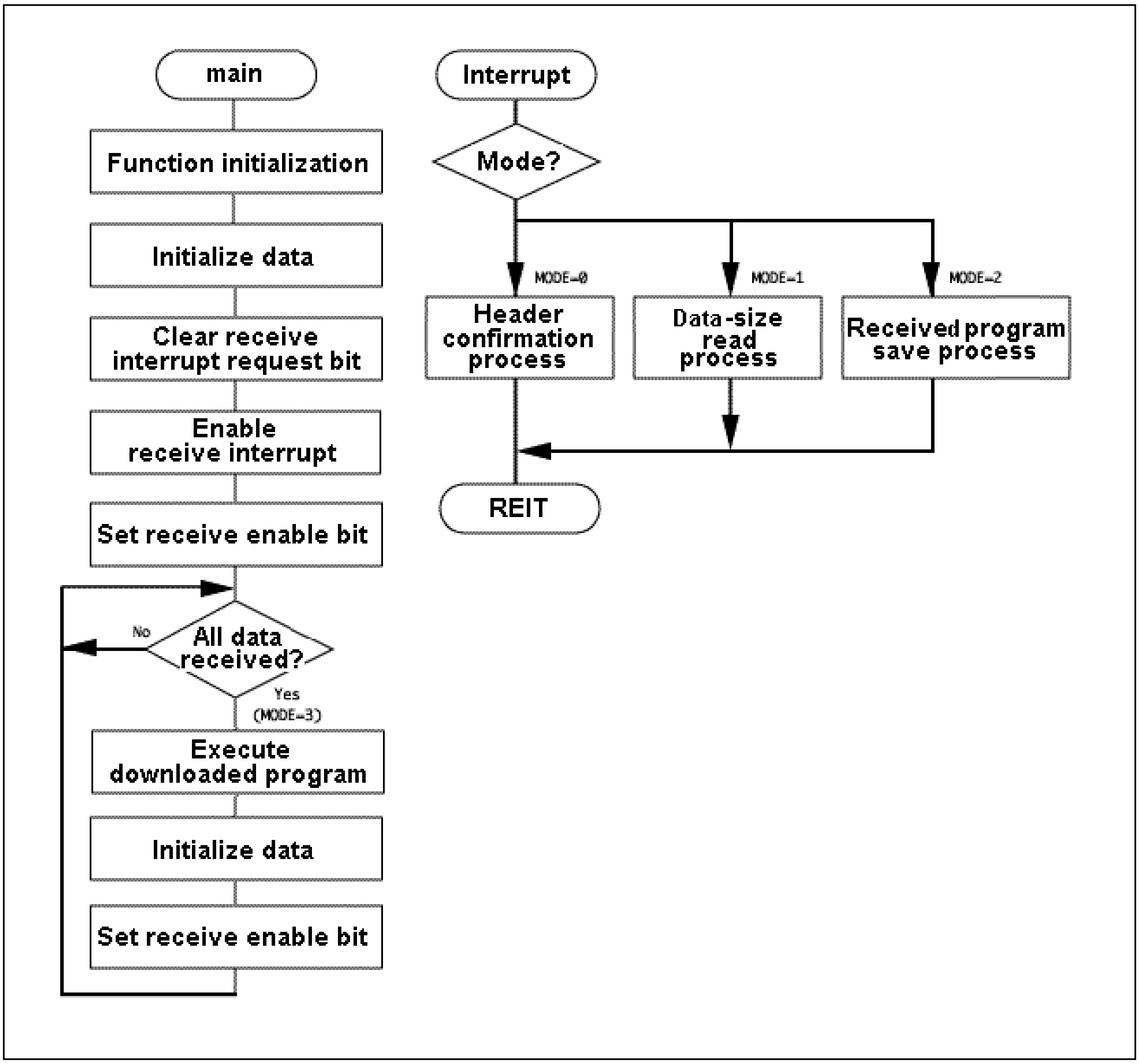


Figure21. Flowchart: Downloading a program using RS-232C

4. Programming Code

```
;*****************************************************************************
;                    Work RAM Area Reservation
;
; Copyright 2003 Renesas Technology America, Inc.
; AND Renesas Solutions Corporation
; All rights reserved
;*****************************************************************************
              .SECTION      RAM,DATA
              .ORG 400H
WORKRAM_TOP:
MODE:         .BLKB   1              ; Operation Mode 0 = Wait for header.
                                     ; 1 = Wait for program size.
                                     ; 2 = Receive program data.
                                     ; 3 = Execute received program.
CNT:          .BLKB   1              ; For counting number of program size reception.
PRG_SIZE:     .BLKB   2              ; For storing program size.
WRITE_POINTER:       .BLKB   2       ; Position pointer for storing program data.
REC_DATA:     .BLKB   2              ; For storing received data.
WORKRAM_END:

PRGRAM_DATA:                         ;
;=========================== Main Program ===================================
;
MAIN:
       JSR    SFR_INIT               ; Function Initialization.
       JSR    DATA_INIT              ; RAM Data Initialization.
       BSET   RE_U0C1                ; Enable reception.
       FSET   I                      ; Set interrupt enable flag.

MAIN_LOOP:
       CMP.B  #3,MODE                ; Loop until reception completion of program data.
       JNZ    MAIN_LOOP              ;
       JSR    PRGRAM_DATA            ; Execute downloaded program.
       JSR    DATA_INIT              ; RAM Data Initialization.
       BSET   RE_U0C1                ; Enable reception.
       JMP    MAIN_LOOP
;*****************************************************************************
;                    Function Initialization
;*****************************************************************************
;
SFR_INIT:
       MOV.B  #05H,U0MR       ; Set clock-asynchronous serial I/O,
                              ; select 8-bit data length and 1 STOP bit, disable parity.
       MOV.B  #10H,U0C0       ; Disable CTS/RTS functions.
       MOV.B  #103,U0BRG      ; Transfer speed: 9600bps
       MOV.B  #1H,S0RIC       ; Enable receive interrupt (interrupt prioriry level =
1).
SFR_INIT_END:
       RTS
```

```
;*****************************************************************************
;                     RAM Data Initialization
;*****************************************************************************
DATA_INIT:
       MOV.B  #0,MODE              ; Initialize mode.
       MOV.B  #0,CNT               ; Initialize number of program size reception.
       MOV.W  #0,WRITE_POINTER     ; Initialize position pointer for storing program
data.
DATA_INIT_END:
       RTS

;*****************************************************************************
;                     Interrupt Process
;*****************************************************************************
S0R_INT:
       PUSH.W A0                   ; Save A0.
       PUSH.W A1                   ; Save A1.
       MOV.W  U0RB,REC_DATA        ; Read received data.
       BTST   15,REC_DATA          ; Check errors (error sum flag).
       JC     ERR_PROC             ; Go to Error Process when an error is detected.
       CMP.B  #0,MODE
       JZ     HEDDER_CHK           ; Check header when mode 0 is selected.
       CMP.B  #1,MODE
       JZ     SIZE_CHK             ; Receive program size when mode 1 is selected.
       CMP.B  #2,MODE
       JZ     PRG_WRITE            ; Store received data to RAM when mode 2 is
selected.
       JMP    S0R_INT_END

;*****************************************************************************
;                     Header Check
;*****************************************************************************
HEDDER_CHK:
    CMP.B     #67H,REC_DATA        ; If header is 67H,
    JNZ       S0R_INT_END          ; change mode to 1.
    MOV.B     #1,MODE
    JMP       S0R_INT_END

;*****************************************************************************
;                     Program Size Reception
;*****************************************************************************
SIZE_CHK:
    MOV.B     CNT,A1                   ; Set position for storing program size.
    MOV.B     REC_DATA,PRG_SIZE[A1]    ; Store program size.
    INC.B     CNT                      ; Increment number of program size reception.
    CMP.B     #2,CNT           ; When reception of 2-byte program size is completed,
    JNZ       S0R_INT_END          ; change mode to 2.
    MOV.B     #2,MODE              ;
    MOV.B     #0,CNT               ; Initialize number of program size reception.
    JMP       S0R_INT_END          ;
```

```
;*****************************************************************************
;                     Received Program Data Storing
;*****************************************************************************
PRG_WRITE:
    MOV.W    WRITE_POINTER,A0            ; Set position for storing program data.
    MOV.B    REC_DATA,PRGRAM_DATA[A0]    ; Store received data to RAM area.
    INC.W    A0               ; Increment position pointer for storing program data.
    MOV.W    A0,WRITE_POINTER
    CMP.W    PRG_SIZE,A0                 ; When all data reception is completed,
    JNZ      S0R_INT_END                 ; change mode to 3.
    BCLR     RE_U0C1                     ; Disable reception.
    MOV.W    #0,WRITE_POINTER            ; Initialize position pointer for storing
program data.
    MOV.B    #3,MODE
;*****************************************************************************
;                     End of Interrupt
;*****************************************************************************
S0R_INT_END:
       POP.W  A1                   ; Restore A1
       POP.W  A0                   ; Restore A0
       REIT

;*****************************************************************************
;                     Error Process
;*****************************************************************************
ERR_PROC:

       ;**************************************************
       ;*    Describe your error process program here.   *
       ;**************************************************

       JMP    S0R_INT_END
```

5. Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com/

E-mail Support

support_apl@renesas.com

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	Feb 25, 2004	-	First edition issued