To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

H8/300H Tiny Series

Monitoring Software

Introduction

The personal computer and H8/3664 are connected using an RS-232C driver, and memory dumps and memory (RAM) editing are performed via a terminal window (Hyperterminal) on the personal computer.

Target Device

H8/3664

Contents

1. Specifications

1. As shown in figure 1.1, the personal computer and H8/3664 are connected using an RS-232C driver, and memory dumps and memory (RAM) editing are performed via a terminal window (Hyperterminal) on the personal computer.
2. As the communication format for transmitted data, transmission and reception are performed with a data length of 8 bits, with no parity bit, one stop bit, at a bit rate of 9600 bits/s.

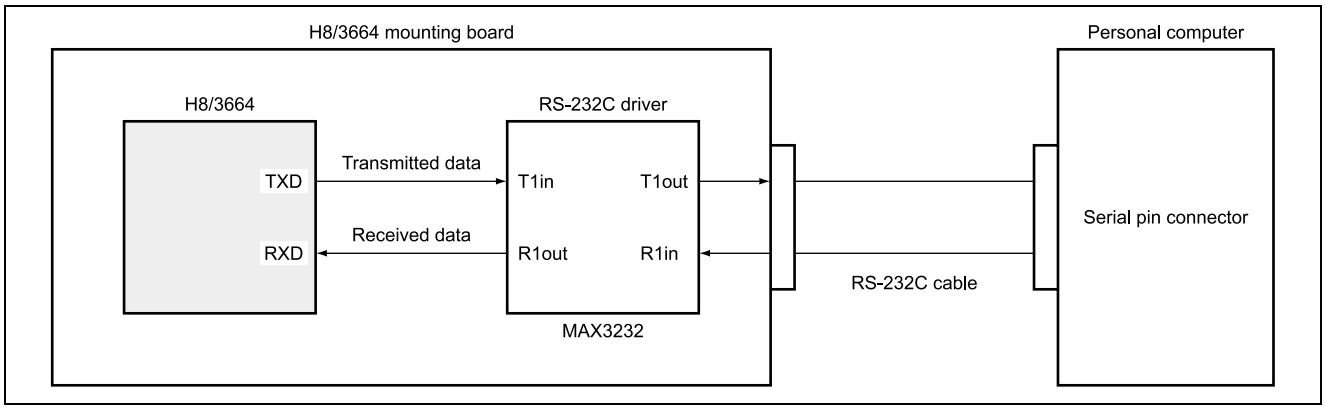
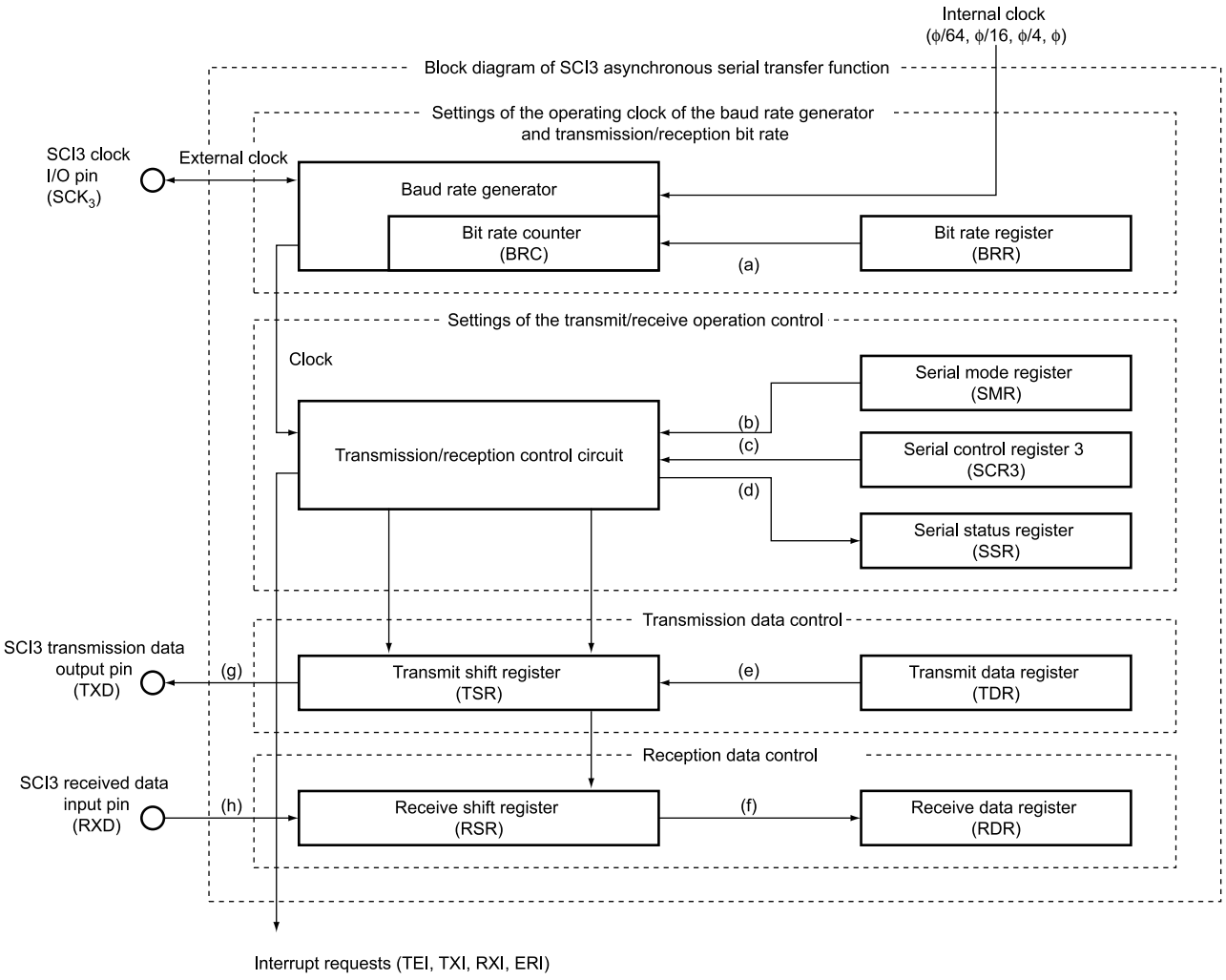


Figure 1.1 Diagram of connection of H8/3664 mounting board to personal computer

2. Description of Functions

1. In this sample task, the serial communication interface (SCI) is used to simultaneously transmit and receive asynchronous serial data. Figure 2.1 shows a block diagram of simultaneous asynchronous serial data transmission and reception. Below, this block diagram of simultaneous asynchronous serial data transmission and reception is described.

- In the asynchronous mode, serial data transmission is performed asynchronously, with synchronization in character units.
- Serial data transmission can be performed with standard asynchronous communication LSI devices, such as a Universal Asynchronous Receiver/Transmitter (UART) and an Asynchronous Communication Interface Adapter (ACIA).
- A multiprocessor communication function is provided, enabling serial communication with multiple processors.
- Any of twelve formats can be selected as the communication format.
- Independent transmission and reception units are provided, so that transmission and reception can be performed simultaneously. And because both the transmission unit and the reception unit have a double-buffer structure, continuous transmission and continuous reception are possible.
- An internal baud rate generator enables selection of an arbitrary baud rate.
- The transmission/reception clock source can be selected from among the internal clock and an external clock.
- There are six interrupt factors: transmission ended, transmission data empty, reception data full, overrun error, framing error, and parity error.
- The receive shift register (RSR) is a register to receive serial data. Serial data input from the RXD pin is set in the RSR in order received from the LSB (bit 0), and converted into parallel data. When one byte of data is received, the data is automatically transferred to the RDR. The RSR cannot be directly read or written from the CPU.
- The receive data register (RDR) is an 8-bit register which stores received serial data. When reception of one byte of data ends, the received data is transferred from RSR to RDR, and the reception operation is ended. Then, the RSR can again receive data. The RSR and RDR have a double-buffer structure, so that continuous reception operation is possible. The RDR is a dedicated reception register, and so cannot be written from the CPU.
- The transmit shift register (TSR) is a register to transmit serial data. Transmission data from the TDR is transmitted to the TSR, and by sending this to the TXD pin in order from the LSB (bit 0), serial data transmission is performed. When one byte of data is transmitted, the next transmission data is automatically transferred from TDR to TSR, and transmission is begun. However, if data is not written to TDR ("1" is set in TDRE), then data is not transferred from TDR to TSR. Reading and writing of TSR directly from the CPU is not possible.
- The transmit data register (TDR) is an 8-bit register which stores transmission data. When TSR "empty" is detected, transmission data written to the TDR is transferred to the TSR, and serial data transmission is begun. During TSR serial data transmission, if the next transmission data is written to the TDR, continuous transmission becomes possible. The TDR can always be read and written by the CPU.
- The serial mode register (SMR) is an 8-bit register used to set the serial data communication format and to select the clock source for the baud rate generator. The SMR can always be read and written by the CPU.
- The serial control register 3 (SCR3) is an 8-bit register which selects transmit/receive operations, asynchronous mode clock output, interrupt request permission/prohibition, and the transmit/receive clock source. The SCR3 can always be read and written by the CPU.



Notes: (a) Together with the operating clock of the baud rate generator selected by SMR, sets the transmission/reception bit rate. In this sample task, transmission bit rate is set to 9600 bit/s.
(b) Sets the serial data communication format and selects the clock source for the baud rate generator. Regarding the serial data communication format of this sample task, the operating mode is set to asynchronous mode, the data length is set to 8 bits, parity to none, stop bit length to 1 bit, and the clock source for the on-chip baud rate generator to φ clock.
(c) Selects transmit/receive operations, asynchronous mode clock output, and interrupt request permission/prohibition. Regarding clock output in the asynchronous mode of this sample task, the communication mode is set to asynchronous, and clock source is set to the on-chip baud rate generator. Regarding the interrupt request permission/prohibition, a transmit data empty interrupt request, a receive data full interrupt request, and a receive error interrupt request are all prohibited.
(d) Indicates the operating state of the SCI3 by a status flag (transmit data register empty, receive data register, overrun error, framing error, parity error, or transmit end).
(e) When TSR "empty" is detected, transmission data written to TDR is transferred to TSR.
(f) When reception of one byte of data ends, the received data is transferred from RSR to RDR.
(g) Transmission data.
(h) Received data.

Figure 2.1 A Synchronous simulations transmission/reception of serial data

- The serial status register (SSR) is an 8-bit register which includes a status flag indicating the operating state of the SCI3, and multiprocessor bits. The SSR can always be read and written by the CPU. However, "1" cannot be written to TDRE, RDRF, OER, PER, or FER. Also, in order to write "0" to clear these bits, "1" must first be read. TEND and MPBR are read-only, and cannot be written.
- The bit rate register (BRR) is an 8-bit register which, together with the operating clock of the baud rate generator selected by CSK1 and CSK0 of SMR, sets the transmission/reception bit rate. BRR can always be read and written by the CPU.
- Table 2.1 shows an example of BRR settings in asynchronous mode. Table 2.1 shows values in active mode when OSC is 16 MHz.

Table 2.1 Example of BRR settings in asynchronous mode

R bit rate (bit/s)	110	150	300	600	1200	2400	4800	9600	19200	31250	38400
n	3	2	2	1	1	0	0	0	0	0	0
N	70	207	103	207	103	207	103	51	25	15	12
Error (%)	0.03	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.00	0.16

Notes: 1. The error boundaries should be set so as to fall within 1%.

2. The BRR settings can be found with the following expression.

$$N = \frac{OSC}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

[Legend]

B: Bit rate (bit/s)

N: BRR setting of the baud rate generator ($0 \le N \le 255$)

OSC: ϕ_{osc} (MHz) = 16 MHz

n: Settings of CKS1 and CKS2 in SMR ($0 \le n \le 3$) (See table 2.2 for the relationship between n and clock.)

Table 2.2 Relationship between n and clock

		SMR setting	
n	**Clock**	**CKS1**	**CKS0**
0	ϕ	0	0
1	$\phi/4$	0	1
2	$\phi/16$	1	0
3	$\phi/64$	1	1

3. Error shown in table 2.1 is indicated by rounding off the third decimal place of the value found with the following expression.

$$\text{Error (\%)} = \left\{ \frac{\phi \times 10^6}{(N + 1) \times B \times 64 \times 2^{2n-1}} \right\} - 1 \times 100$$

4. The maximum bit rate (asynchronous mode) when OSC is 16 MHz is 500000 (bit/s). Note that the settings are: n = 0, and N = 0.

- Asynchronous mode is a mode in which a start bit indicating the beginning of communication and a stop bit indicating the end of communication are added to the data to form characters for transmission and reception, and serial communication is performed with synchronization in single character units.
- Within SCI3, the transmission and reception units are independent, so that full-duplex communication is possible. And, both the transmission and the reception units have a double-buffer structure, so that data can be written during transmission and data can be read during reception, and continuous transmission and reception are possible.
- Figure 2.2 shows the asynchronous communication data format. In asynchronous communication, the communication line is normally held in the mark state ("high" level). The SCI3 monitors the communication line, and regards the occurrence of a space ("low" level) as a start bit, and begins serial communication.
- One character of communication data is configured to begin with a start bit ("low" level), then the transmitted/received data (LSB first, from the lowermost bit), then a parity bit ("high" or "low" level), and finally a stop bit ("high" level).
- In asynchronous mode, synchronization is on the falling edge of the start bit at the time received. Further, data is sampled on the eighth cycle of a clock at a frequency 16 times the interval of one bit, so that communication data is captured at the center of each bit.

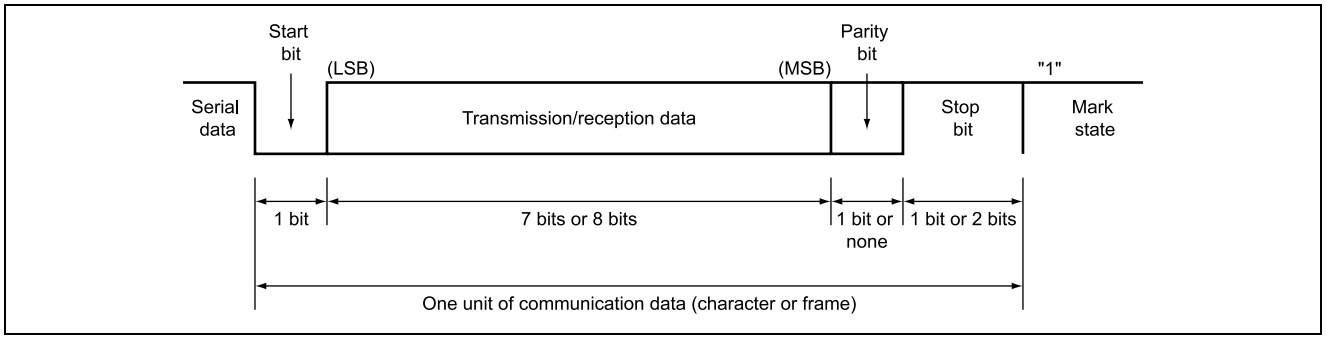


Figure 2.2 Asynchronous communication data format

- The SCI3 clock (SCK_3) is the SCI3 clock I/O pin.
- The SCI3 receive data input (RXD) is the SCI3 received data input pin.
- The SCI3 transmit data output (TXD) is the SCI3 transmission data output pin.
- There are six types of SCI3 interrupt requests: transmission end, transmission data empty, received data full, and three types of reception error (overrun error, framing error, parity error). A common vector address is assigned.
- Each interrupt request is permitted or prohibited by the SCR3 TIE and RIE bits.
- When bit TDRE is set to 1 in SSR, a TXI interrupt is requested. When bit TEND is set to 1 in SSR, a TEI interrupt is requested. These two interrupts are generated during transmission.
- The initial value of the TDRE of SSR is "1". Hence before transferring transmission data to TDR, if the TIE of SCR3 is set to "1" and a transmission data empty interrupt request (TXI) is permitted, TXI is generated even if transmission data is not prepared.
- The initial value of the TEND of SSR is "1". Hence before transferring transmission data to TDR, if the TEIE of SCR3 is set to "1" and a transmission end interrupt request (TEI) is permitted, TEI is generated even if transmission data has not been transmitted.
- By performing processing to transfer transmission data to TDR during an interrupt processing routine, the interrupts can be utilized efficiently. Also, in order to prevent the generation of these interrupts (TXI, TEI), after transferring transmission data to TDR, the permission bits corresponding to these interrupt requests (TIE, TEIE) are set to "1".
- If the RDRF of SSR is set to "1", RXI is generated. If any of OER, PER, or FER is set to "1", ERI is generated. These two interrupt requests occur during reception.

2. Table 2.3 indicates function allocations in this sample task. The functions are allocated as shown in table 2.3 to perform asynchronous serial data transmission and reception simultaneously.

Table 2.3 Function allocation

Function	Function Allocation
RSR	Register to receive serial data
RDR	Register to store received data
SMR	Sets serial data communication format and clock source for the baud rate generator
SSR	Status flag indicating SCI3 operating status
BRR	Sets transmission/reception bit rate
PMR1	Sets TXD output pin
SCK3	SCI3 clock output pin
TXD	SCI3 transmission data input pin
RXD	SCI3 received data input pin

3. Principles of Operation

1. Figure 3.1 illustrates the principle of operation. As figure 3.1 indicates, simultaneous asynchronous serial data transmission and reception are performed through both hardware and software processing.

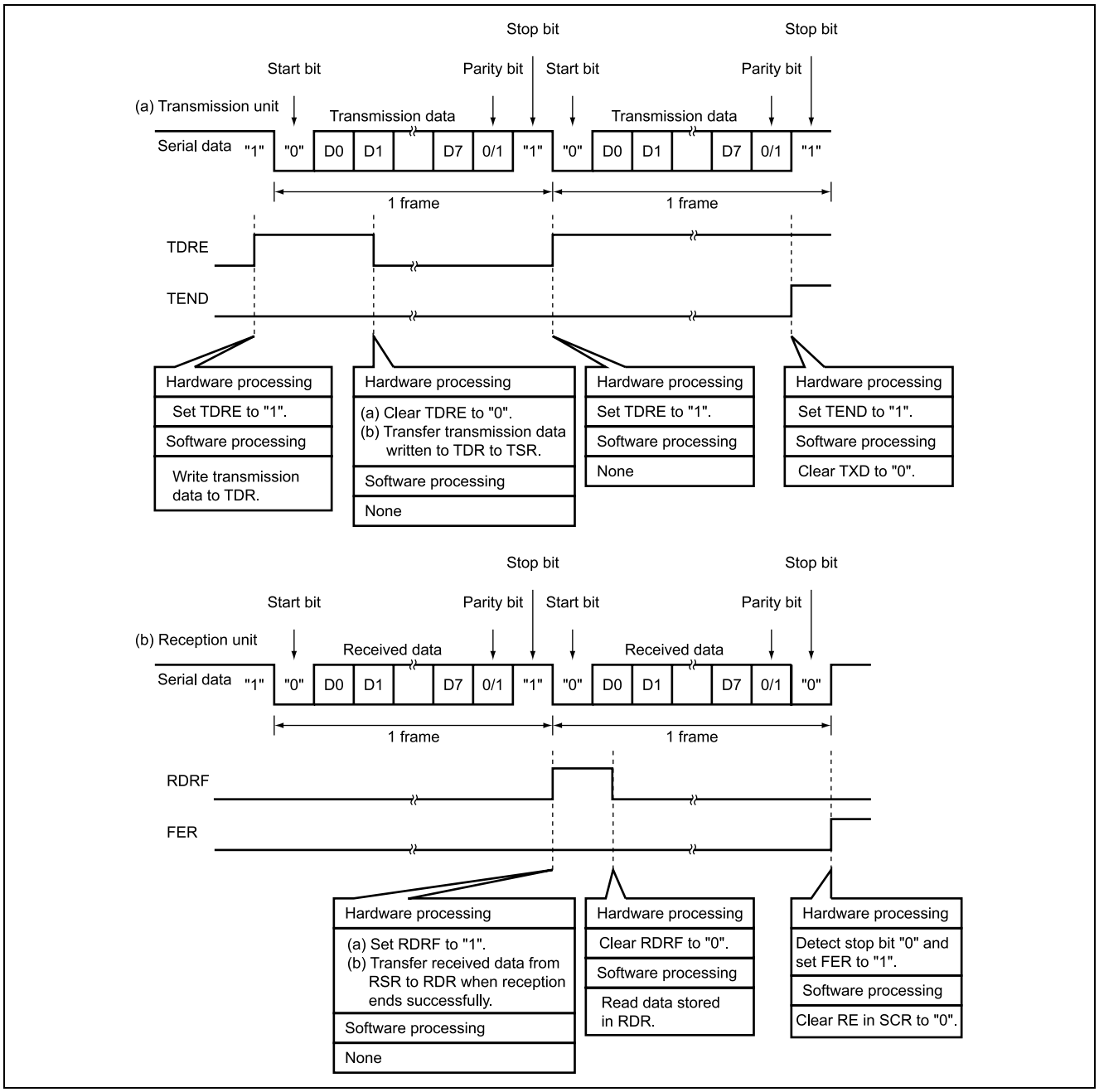


Figure 3.1 Principle of operation of simultaneous asynchronous serial data transmission and reception

4. Description of Software

4.1 Description of modules

Table 4.1 describes the modules used in this sample task (giving arguments and return values).

Table 4.1 Modules used

Module (Function) Name	Argument	Return value	Function
INIT (assembly)	None	None	Sets stack pointer (sets R7 to H'FF80) Sets CCR (prohibits interrupt requests) Jumps to main
main	None	None	Main module
Init_sci	None	None	Clears the serial status register to 0 Sets for serial communications (asynchronous mode, data length: 8 bits, no parity, 1 stop bit, bit rate: 9600 bps) Permits TXI, RXI, ERI interrupt processing and transmission/reception
Dsp_Init	None	None	Sets data on the terminal window of the personal computer for displaying the program version
Int_SCI3	None	None	Serial interrupt processing Determines whether the serial register is in error status, transmission status, or reception status
Int_s3err	None	None	Clears the receive counter and serial status register to 0
Int_s3rx	None	None	Serial reception processing
Int_s3tx	None	None	Serial transmission processing
Ans_send	None	None	Response data set processing
Dump_send	None	None	Memory dump data set processing
DSP_mem	None	None	Pre-memory edit data set processing
DSP_err	None	None	Data set processing for error
Set_send1	work (transmission data)	None	Transmission data set processing
DSP_idle	None	None	Data set processing for line feed
Dump_start	adrs (dump start address)	None	Variable set for memory dump processing
Cnvt_AsctoAddress	None	None	Address data conversion (4-byte ASCII -> 2-byte Hex)
Cnvt_AxctoHex	*ptr (converted data storage address)	data (1-byte Hex data)	Converts 2-byte ASCII data to 1-byte Hex data
Cnvt1_AxctoHex	*ptr (converted data storage address)	data (4-bit Hex data)	Converts 1-byte ASCII data to 4-bit Hex data
Cnvt1_HextoAsc	data (1-byte Hex data)	*ptr (converted data storage address)	Converts 1-byte Hex data to 2-byte ASCII data
Int_trap	None	None	Trap interrupt processing
NMI	None	None	NMI interrupt processing (no processing performed)
Int_s3te	None	None	SCI1 TEI interrupt processing (no processing performed)

4.2 Component files

Table 4.2 lists the files used in this task.

Table 4.2 Files

File Name	Processing
DBSCT.C	Initialization of non-initialized area (HEW automatic generation)
INIT.SRC	Stack pointer and CCR settings (at resetting)
KMONI.C	Main program, serial interrupt processing, communication subroutine processing, and vector table definition
ASCDEFINE.H	ASCII code definition (header file)

4.3 Section definitions

Table 4.3 lists section definitions for this task.

Table 4.3 Section definition

Address	Section	Description
H'0000	CV1	Reset vector address
H'0010	CV2	Trap vector address
H'002E	CV3	SCI vector address
H'0100	P	Program area
	C$DSEC	Initialized data area (defined by DBSCT.C)
	C$BSEC	Non-initialized data area (defined by DBSCT.C)
	D	Initialized data area
	C	Constant area
H'FB80	B	Non-initialized data area
	R	Initialized data area

4.4 Global variables used

Global variables used in this sample task are described in table 4.4.

Table 4.4 Global variables used

Variable Name	Type	Size	Application
s3rx_buf	unsigned char	16	Reception data buffer
s3tx_buf	unsigned char	60	Transmission data buffer
s3rx_cnt	unsigned char	1	Received data bit length
s3tx_cnt	unsigned char	1	Transmitted data bit length
s3tx_cnt	unsigned char	1	Transmission data bit length
dummy	unsigned char	1	Serial data register dummy write area
trap_flag	unsigned char	1	Trap interrupt processing flag
moni_code	unsigned char	1	Selects re-reception operation after data has been received
ans_code	unsigned char	1	Selects operation upon generating response data
dump_ycnt	unsigned char	1	Dump line count upon memory dump
*cr_adrs	unsigned char	1	Address for memory dump/memory edit
*dump_adrs	unsigned char	1	Address for memory dump/memory edit
CMem.data	unsigned char	2	Address for memory dump/memory edit (union with *CMem.adrs)
CMem.adrs	unsigned char	1	Address for memory dump/memory edit (union with *CMem.data)

4.5 Internal registers used

Table 4.5 describes internal registers used in this sample task.

Table 4.5 Internal registers used

Register Name		Function	Operation	Setting
SMR	COM	Sets communication mode to asynchronous mode	Setting	0
	CHR	Sets data length in asynchronous mode to 8 bits	Setting	0
	PE	In asynchronous mode, does not perform addition and checks of parity bit upon transmission	Setting	0
	PM	Invalid since PE in SMR = 0	Setting	0
	STOP	Sets stop bit length in asynchronous mode to 1 bit	Setting	0
	MP	Prohibits multiprocessor communication function	Setting	0
	CKS1 CKS0	Set the clock source for the on-chip baud rate generator to ϕ clock	Setting	CKS1 = 0 CKS0 = 0
BRR		Sets transmission bit rate to 9600 (bit/s) together with the operation clock for the baud rate generator selected by CKS1 and CKS0 in SMR	Setting	H'51
SCR3	TIE	When TIE = 1, enables TXI interrupt requests	Setting	0/1
	RIE	Permits RXI and ERI interrupt requests	Setting	1
	TE	Permits transmit operation	Setting	1
	RE	Permits receive operation	Setting	1
	MPIE	Invalid since MP in SMR = 0	Setting	0
	TEIE	Refuses TEI interrupt requests	Setting	0
	CKE1 CKE0	Set a clock source to baud rate generator	Setting	CKE1 = 0 CKE0 = 0
TDR		8-bit register to store transmission data	Storage	—
RDR		8-bit register to store received data	Storage/ reference	—
SSR	TDRE	0: Transmission data is written to TDR, or 0 is written after reading 1 1: Data is transferred from TDR to TSR	Setting/ reference	0/1
	RDRF	0: Data in RDR is read, or 0 is written after reading 1 1: Data is transferred from RSR to RDR after reception has ended successfully	Setting/ reference	0/1
	OER	0: 0 is written after reading 1 1: Overrun error occurs during reception	Setting/ reference	0/1
	FER	0: 0 is written after reading 1 1: Framing error occurs during reception	Setting/ reference	0/1
	PER	0: 0 is written after reading 1 1: Parity error occurs during reception	Setting/ reference	0/1
	TEND	0: Transmission data is written to TDR, or 0 is written after reading 1 1: TDRE = 1 upon transmission of lowermost bit of the transmission character	Setting/ reference	0/1
	MPBR	Stores the multiprocessor bit in the received character. No change when RE in SCR = 0	Reference	—
	MPBT	Sets multiprocessor bit value in the transmission character	Reference	—
PMR1		Sets $P2_2$/TXD pin function as TXD pin function	Setting	0x02

4.6 Monitor Control

The method of H8/3664 monitor control is described below.

All input characters should be single-byte alphanumeric characters.

If an error is made in character input, [Backspace] can be used to delete the preceding character.

The initial letter "D" and alphabet characters (A to F) in addresses during memory dump control, and the initial character "E" and alphabet characters (A to F) in addresses during memory editing, can be input as lower-case characters.

1. Performing a memory dump
   After input of "D", the leading address for the data to be dumped is input as a 4-digit hexadecimal number, and [Enter] is pressed to cause 128 bits of dump data beginning from the specified address to be displayed.

   Example: To display dump data starting from address 0xFE80 → DFE80 (Figure 4.1)

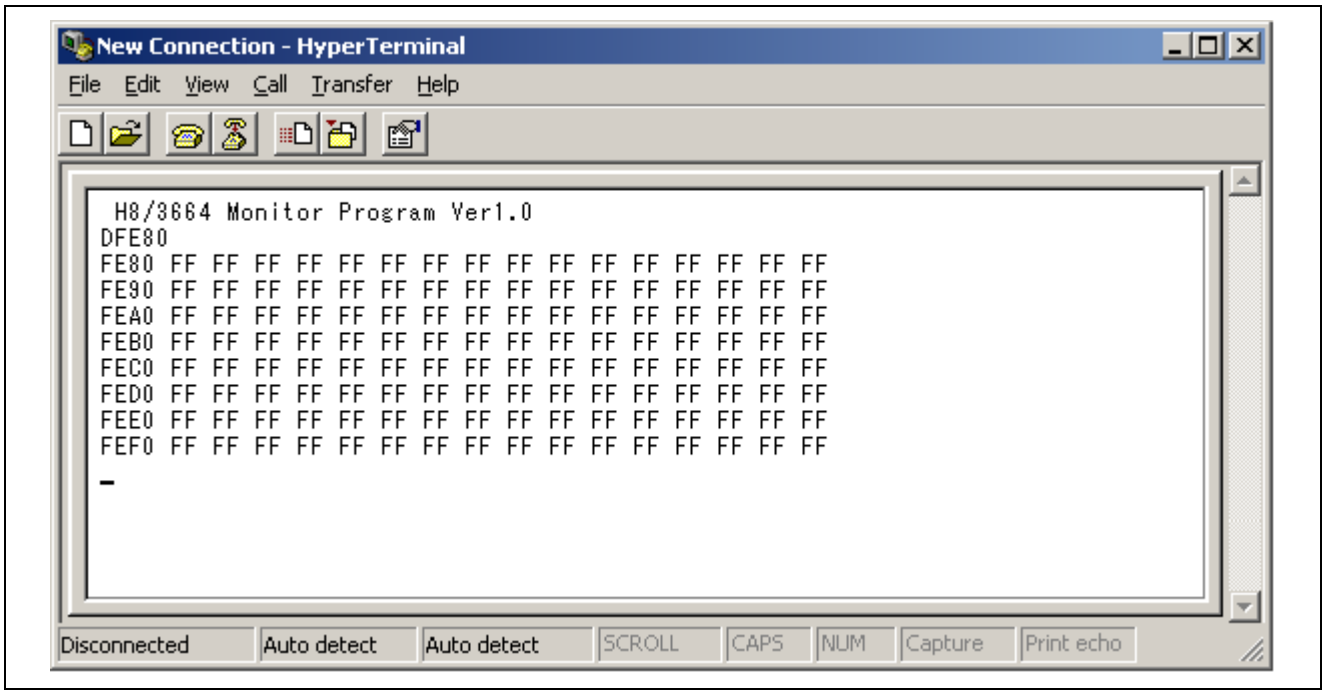


Figure 4.1 Memory data dump (address 0xFE80)

After the dump data is displayed, if [Enter] is pressed without input of any characters, the next 128 bits of dump data is displayed.

Example: In the state shown in figure 4.1 (dump data up to address 0xFEFF displayed), press [Enter]
→128 bits of dump data from address 0xFF00 is displayed (Figure 4.2)

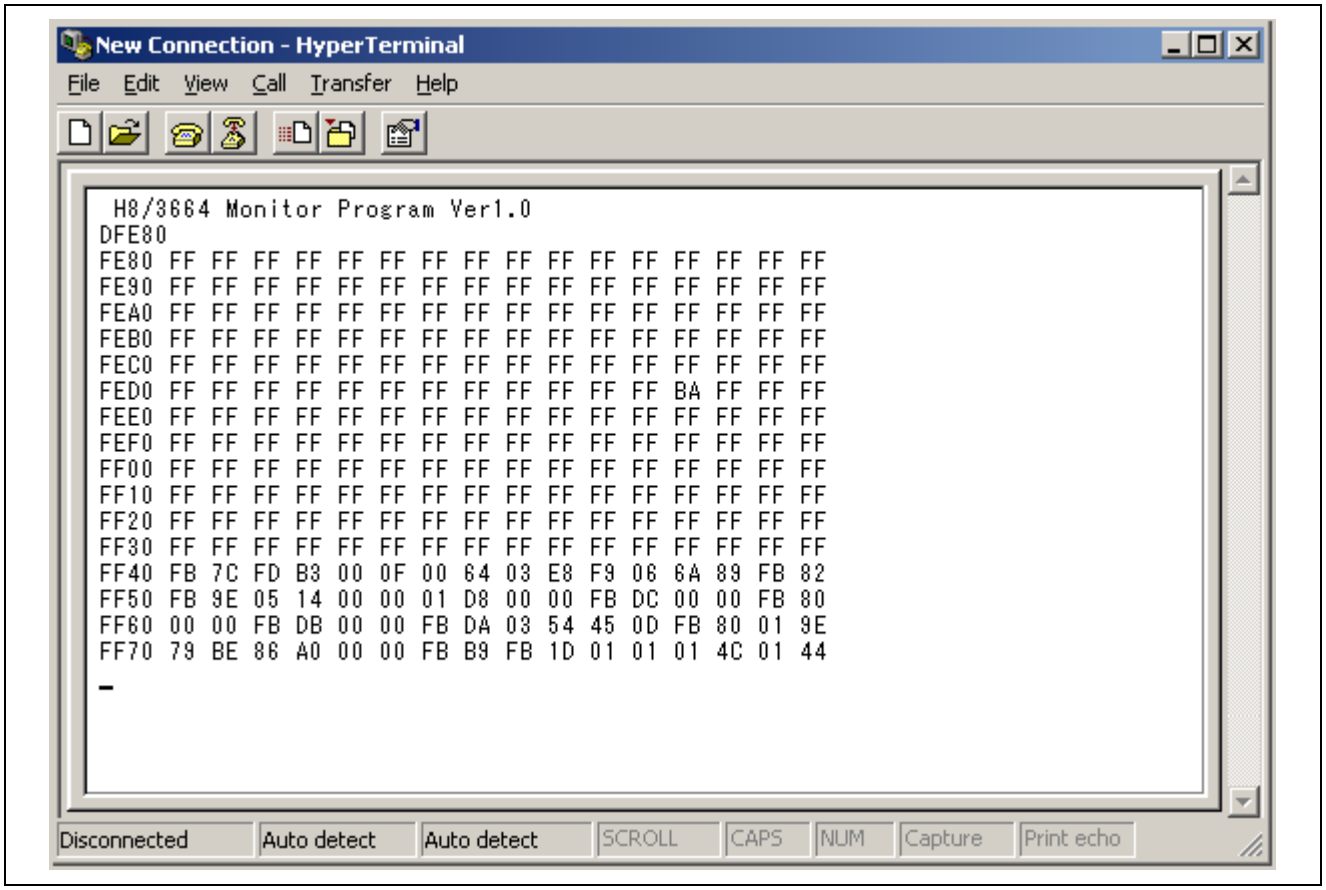


Figure 4.2 Memory data dump (sequential dump)

2. Performing memory edits
   After input of "E", the address for editing is input as a 4-digit hexadecimal number, and [Enter] is pressed. On input of an address, the specified address and the current value at the specified address are displayed.

   Example: To edit data at address 0xFE00 → EFE00 (Figure 4.3)

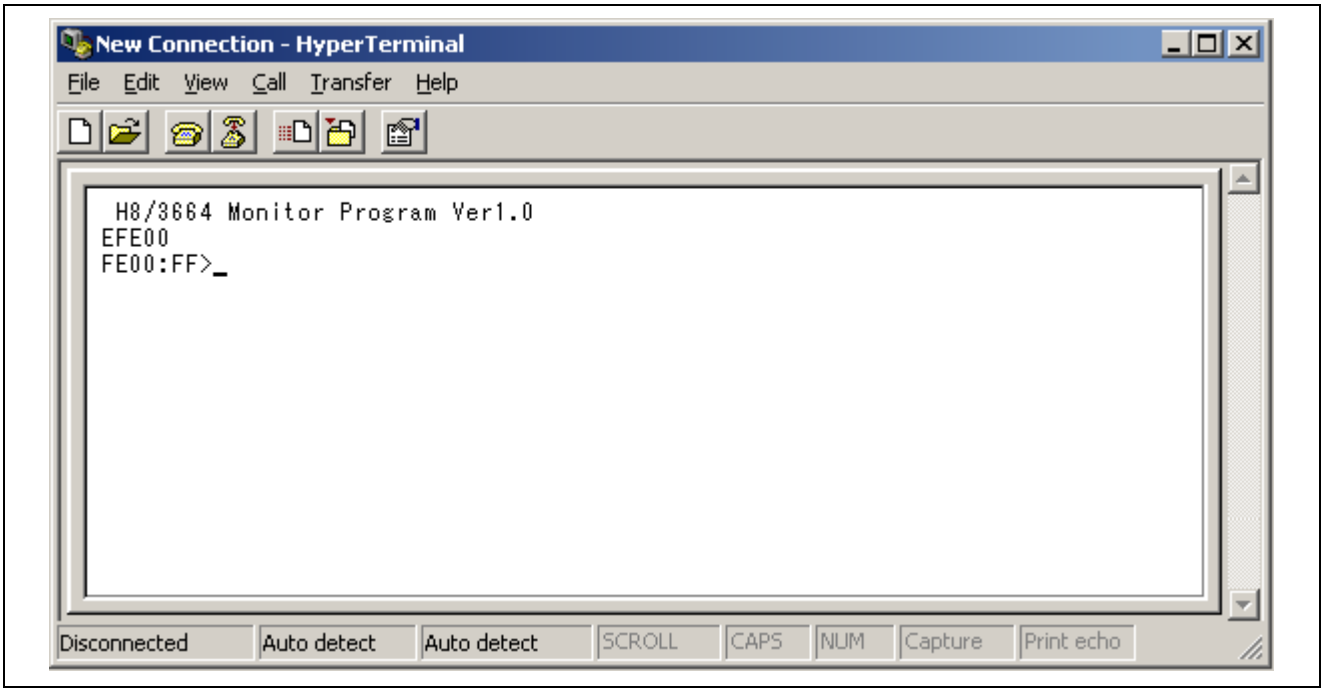


Figure 4.3 Memory data edit (address specification)

In a state in which an address has been specified and the data at the specified address is displayed, input the value to which the data is to be changed, and press [Enter].

Example: In the state of figure 4.3, to change the data at address 0xFE00 from FF to 00
→ FE00:FF>00 (Figure 4.4)

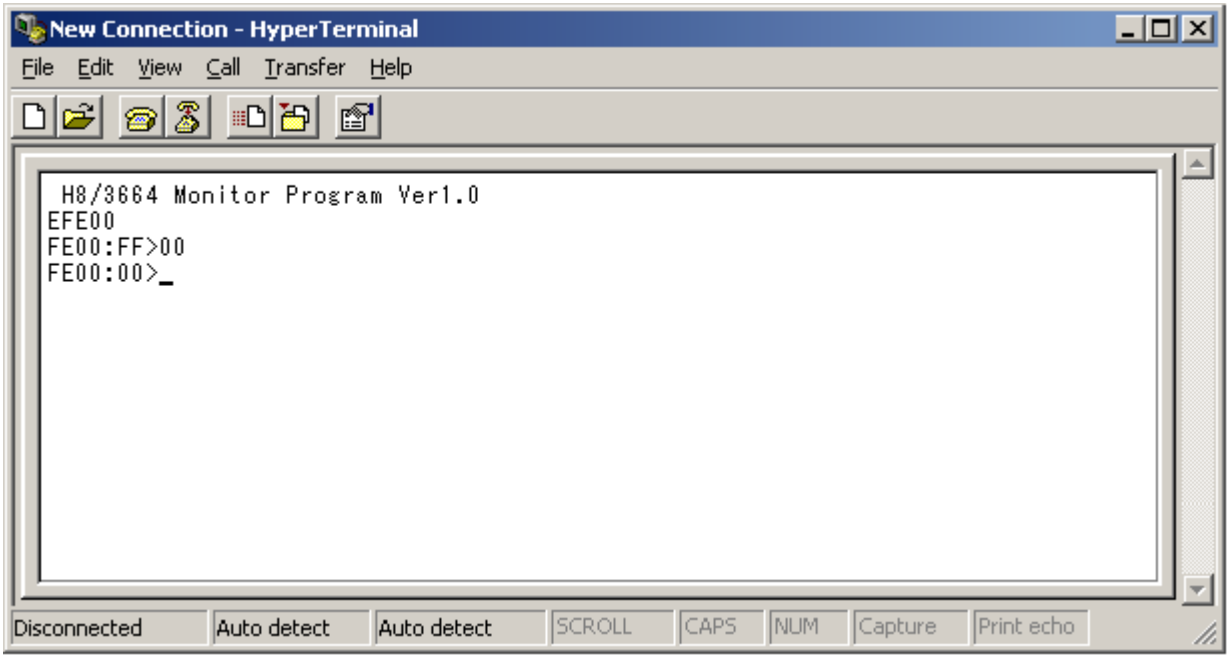


Figure 4.4 Memory data edit (data change)

In a state in which an address and the current value are displayed, if [Enter] is pressed without input of any characters, the data at the next address can be edited (Figure 4.5).

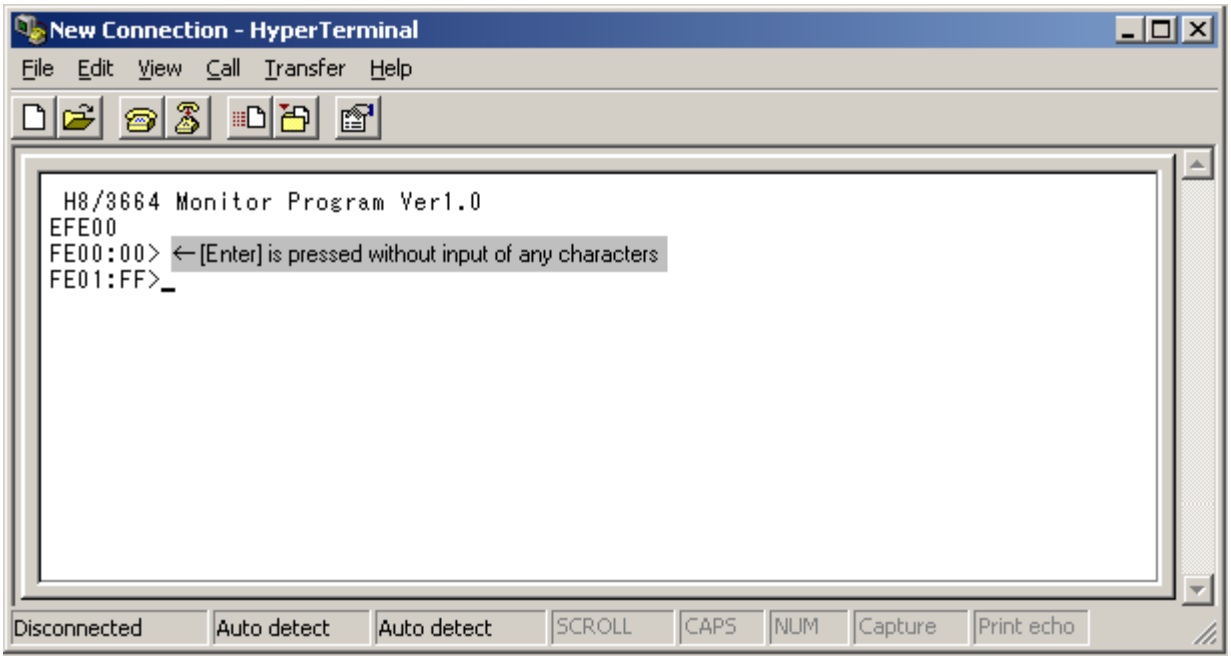


Figure 4.5 Memory data edit (address feed)

In a state in which an address and the current value are displayed, if " ^ " is pressed without input of any characters, the data at the previous address can be edited (Figure 4.6).

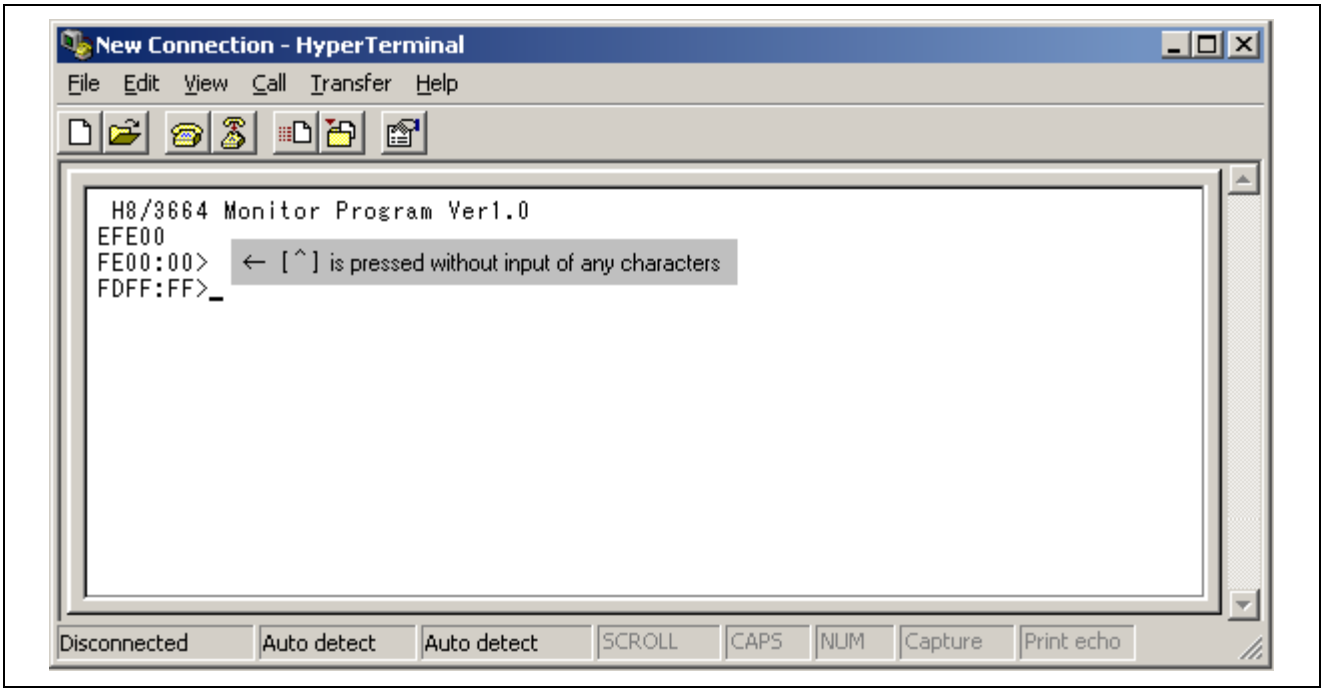


Figure 4.6 Memory data edit (address return)

To end memory editing, press " . " without input of any characters; memory editing is concluded. (Figure 4.7)

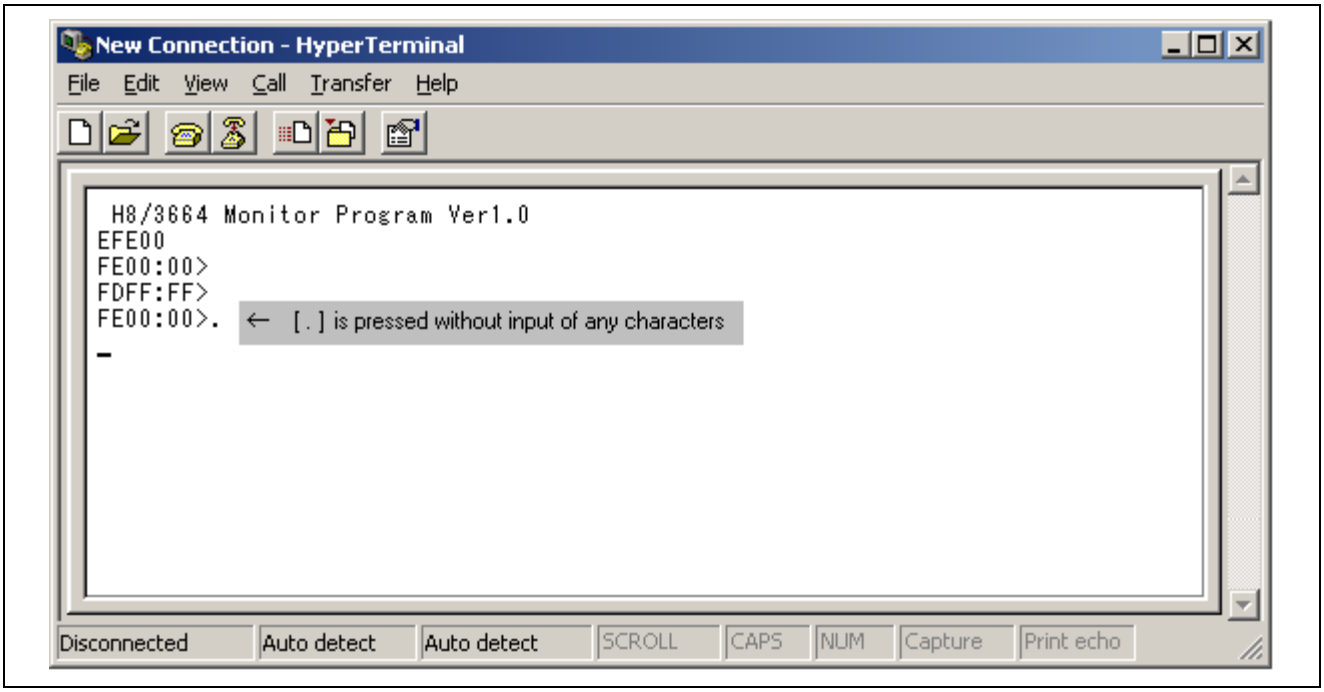


Figure 4.7 Memory data edit (edit end)

4.7 Hierarchical diagram of modules

A hierarchical diagram of modules is shown in figure 4.8.

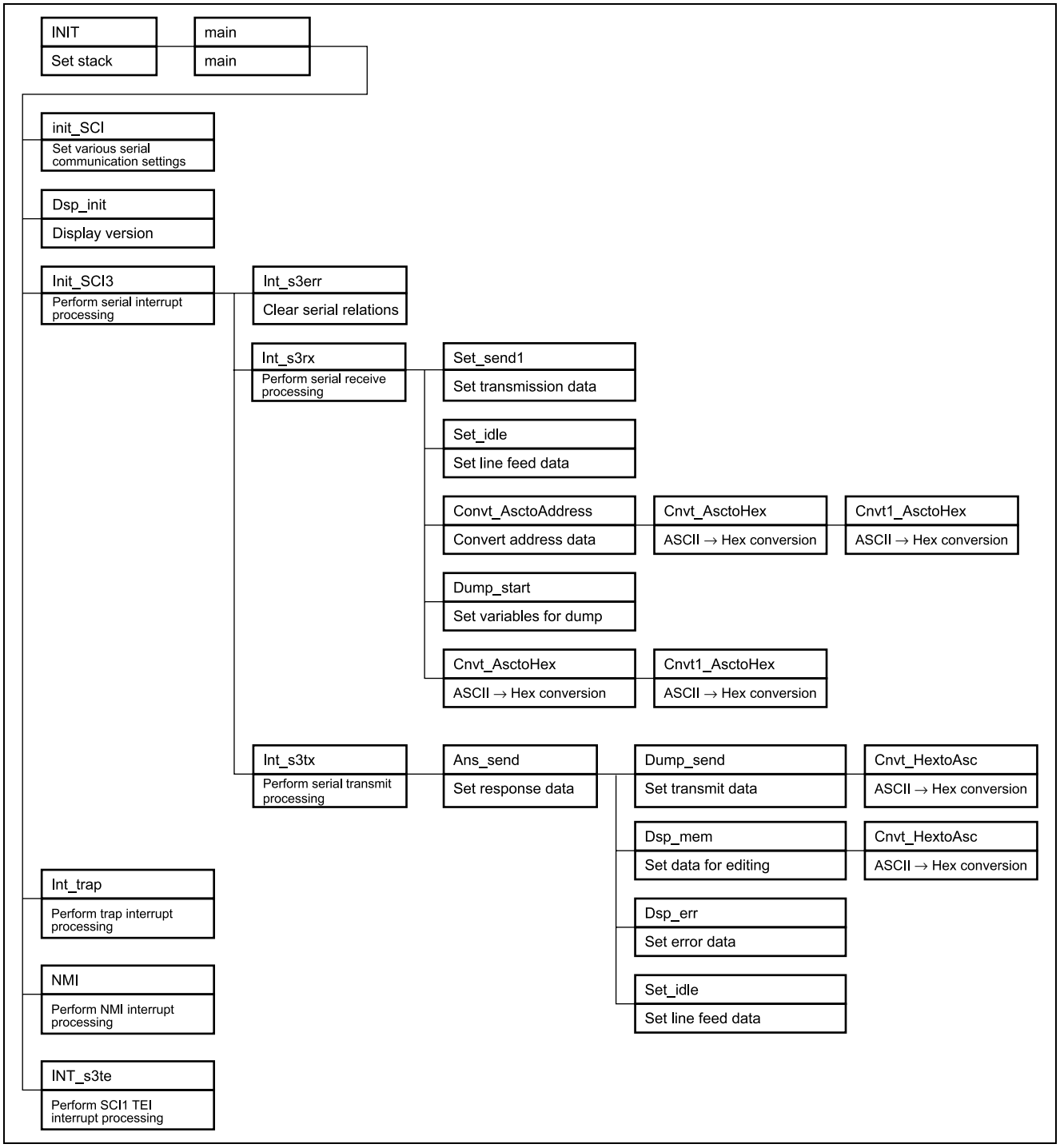
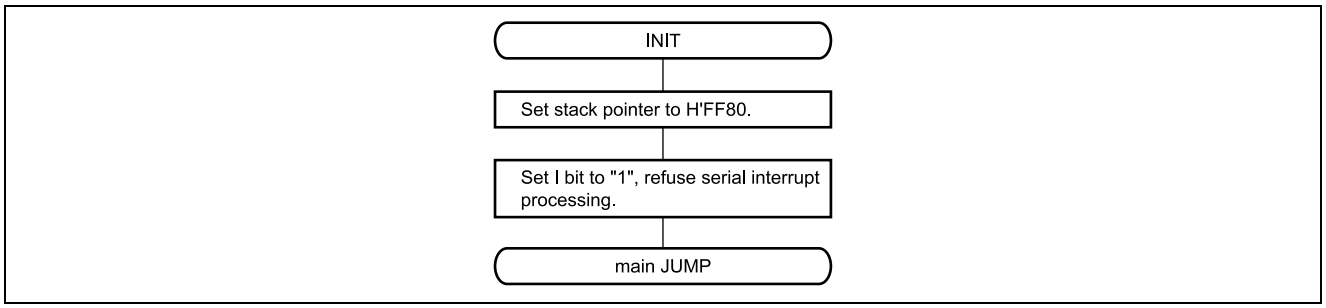
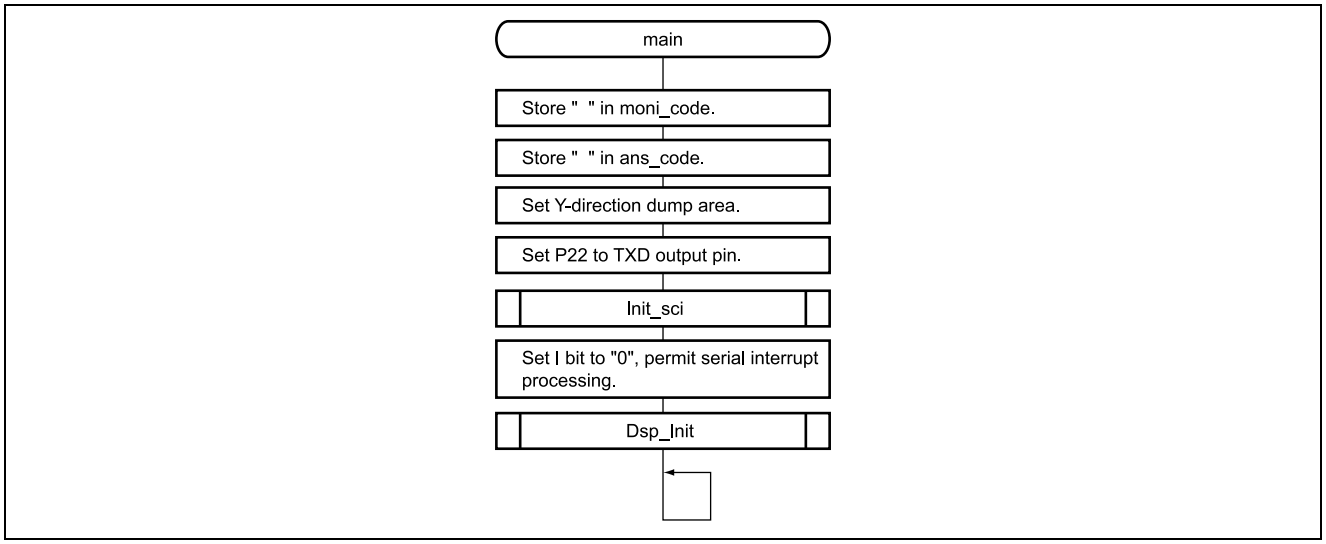
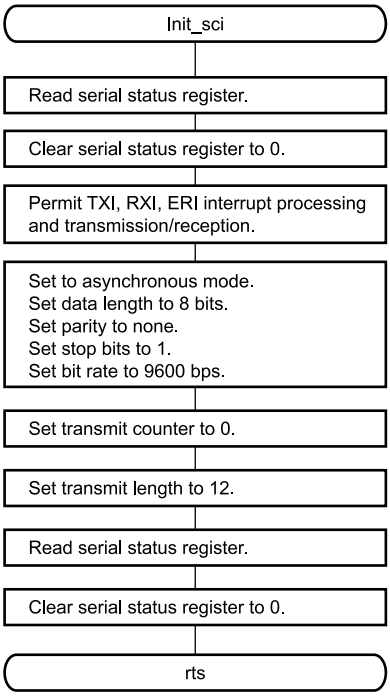


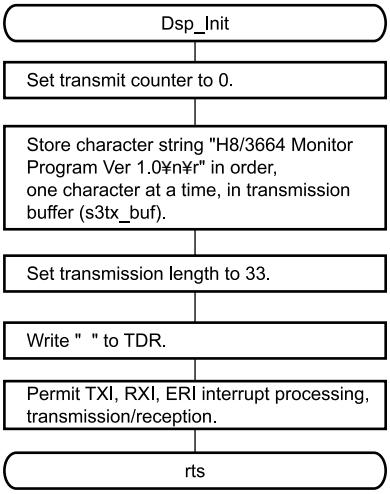
Figure 4.8 Hierarchical diagram of modules

5. Flowcharts

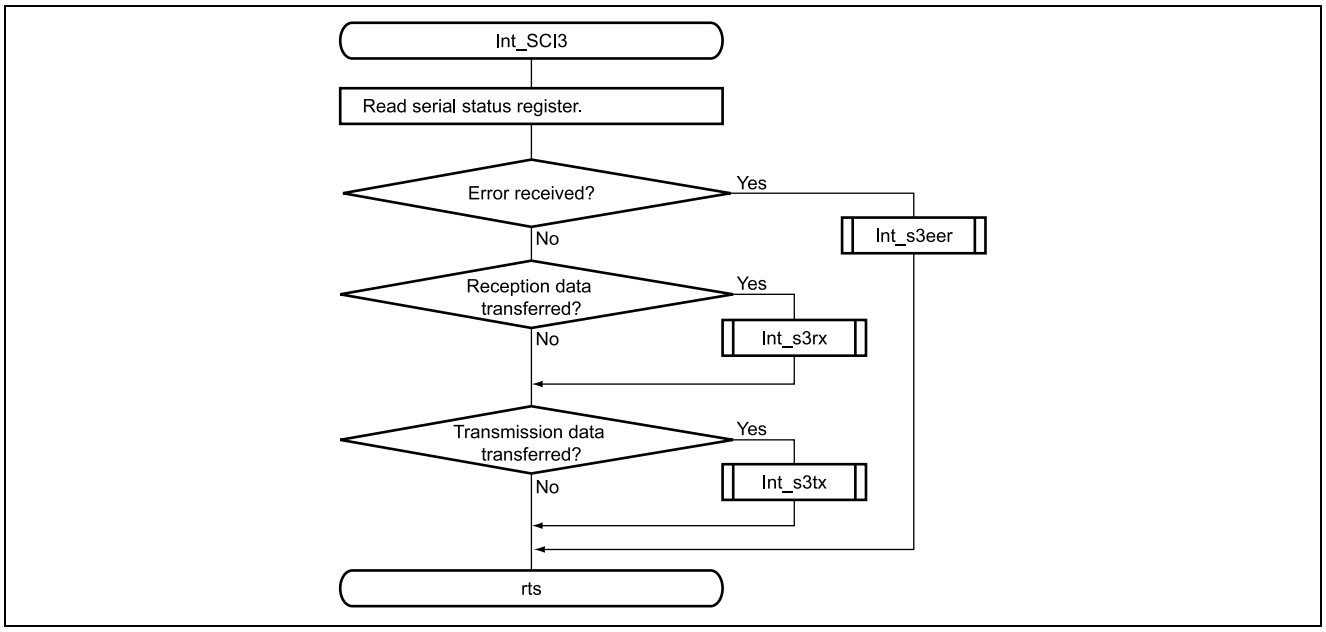


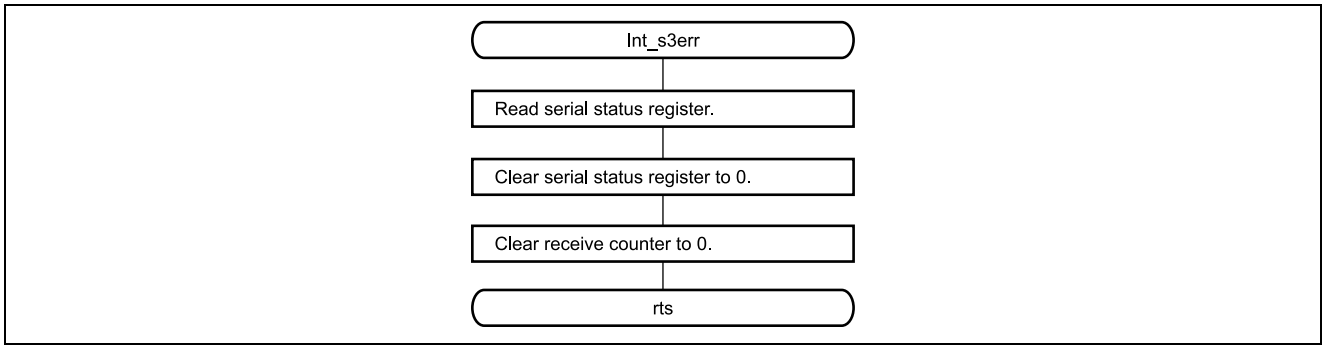


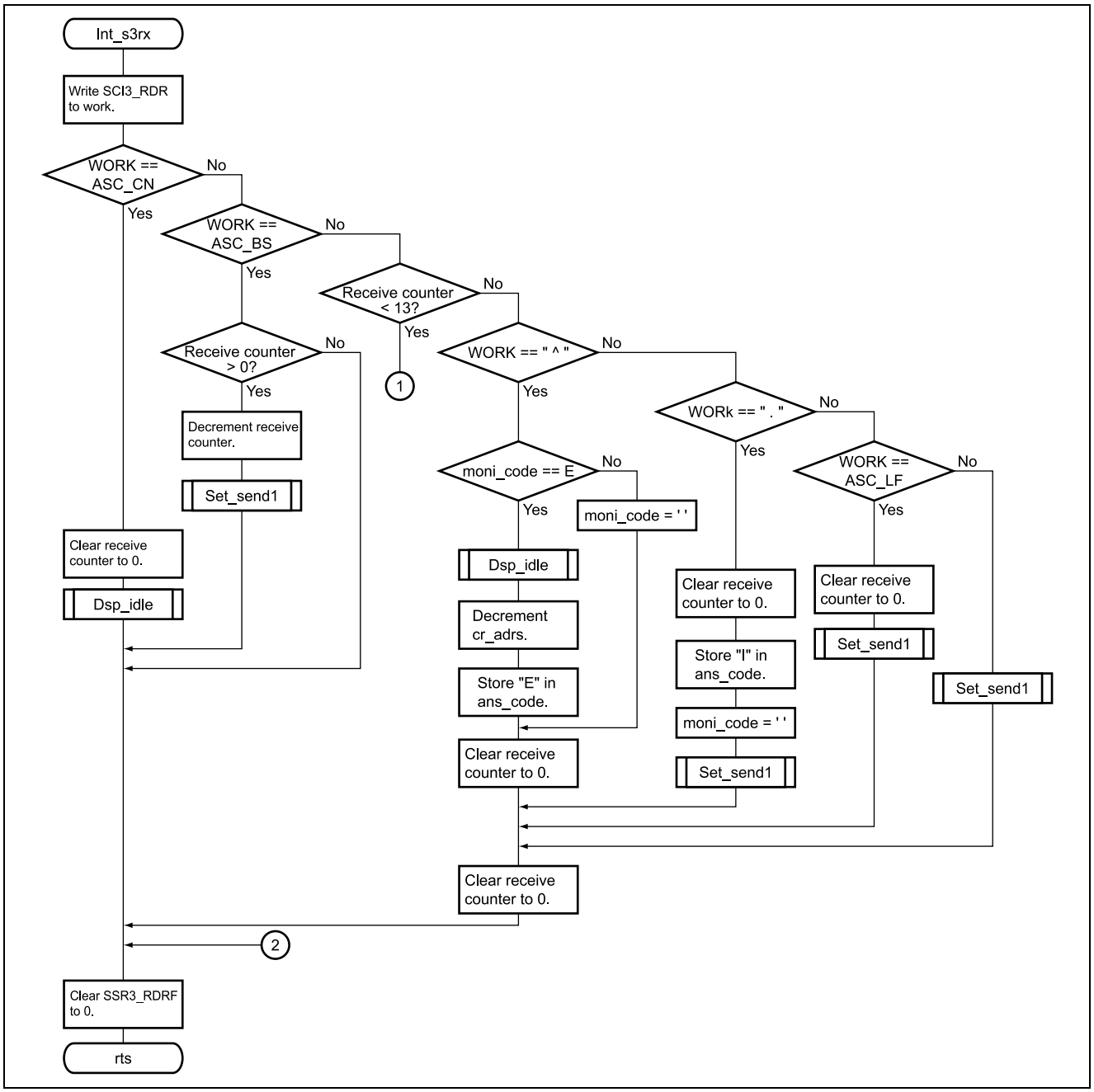

Init_sci
Read serial status register.
Clear serial status register to 0.
Permit TXI, RXI, ERI interrupt processing and transmission/reception.
Set to asynchronous mode.
Set data length to 8 bits.
Set parity to none.
Set stop bits to 1.
Set bit rate to 9600 bps.
Set transmit counter to 0.
Set transmit length to 12.
Read serial status register.
Clear serial status register to 0.
rts

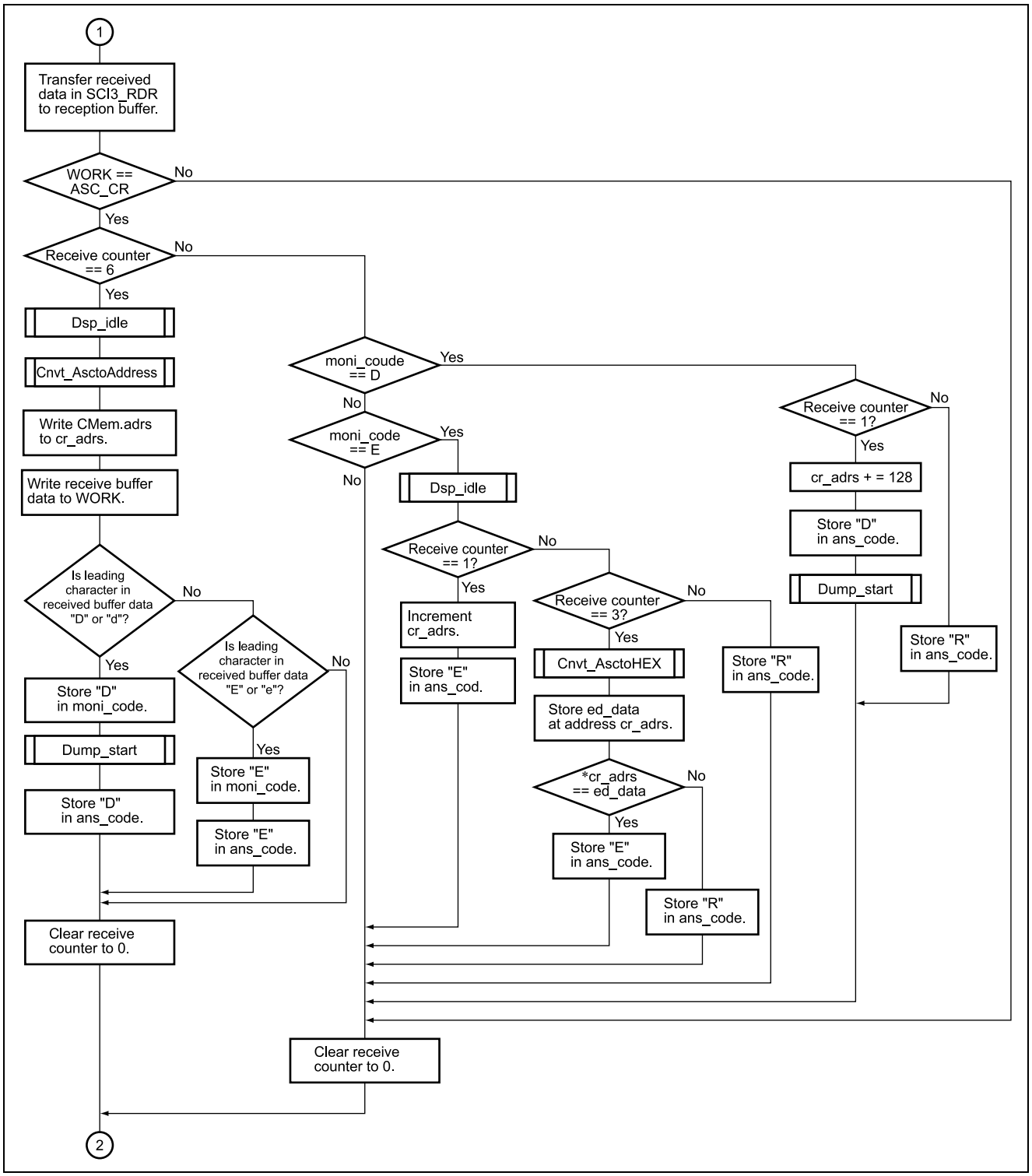

Dsp_Init
Set transmit counter to 0.
Store character string "H8/3664 Monitor Program Ver 1.0¥n¥r" in order, one character at a time, in transmission buffer (s3tx_buf).
Set transmission length to 33.
Write " " to TDR.
Permit TXI, RXI, ERI interrupt processing, transmission/reception.
rts

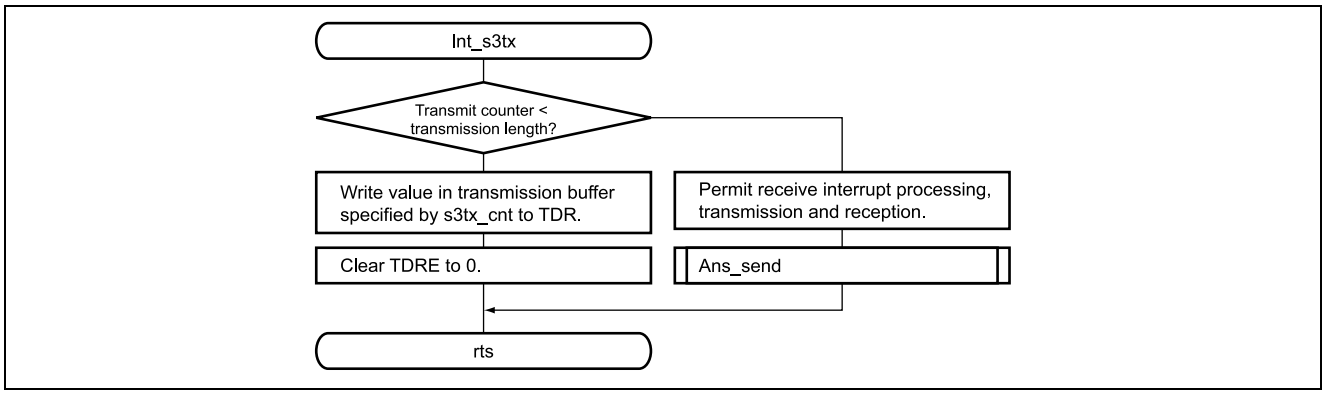
Serial interrupt processing routine

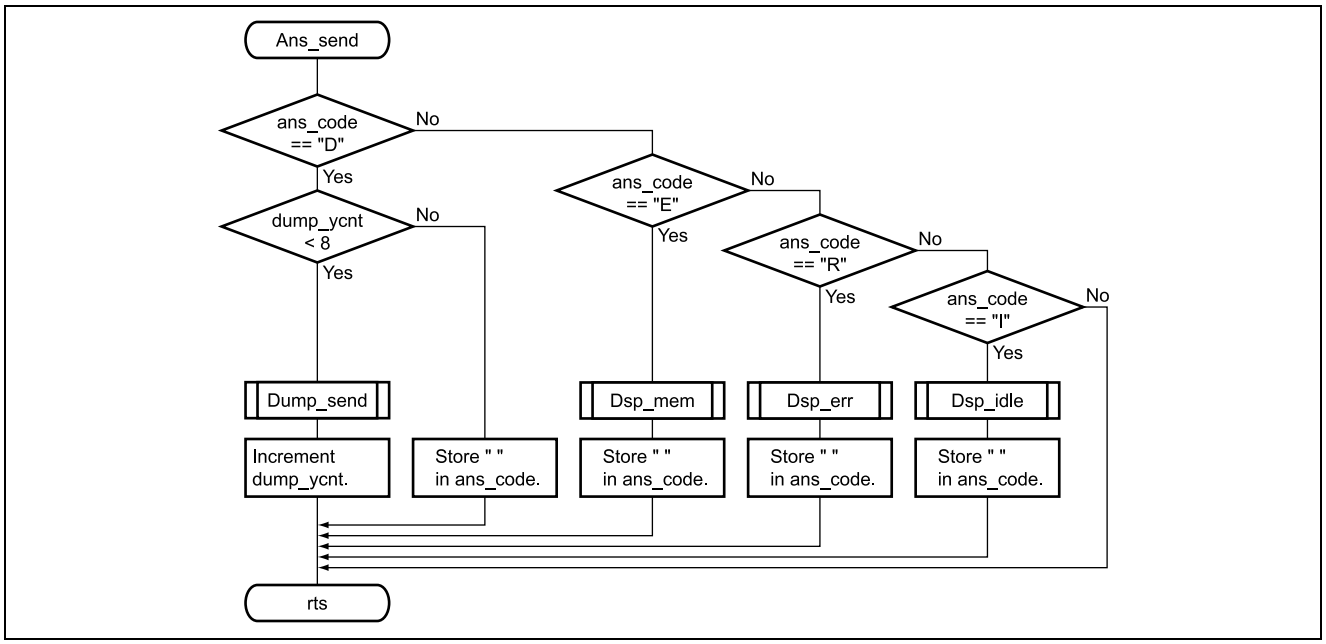


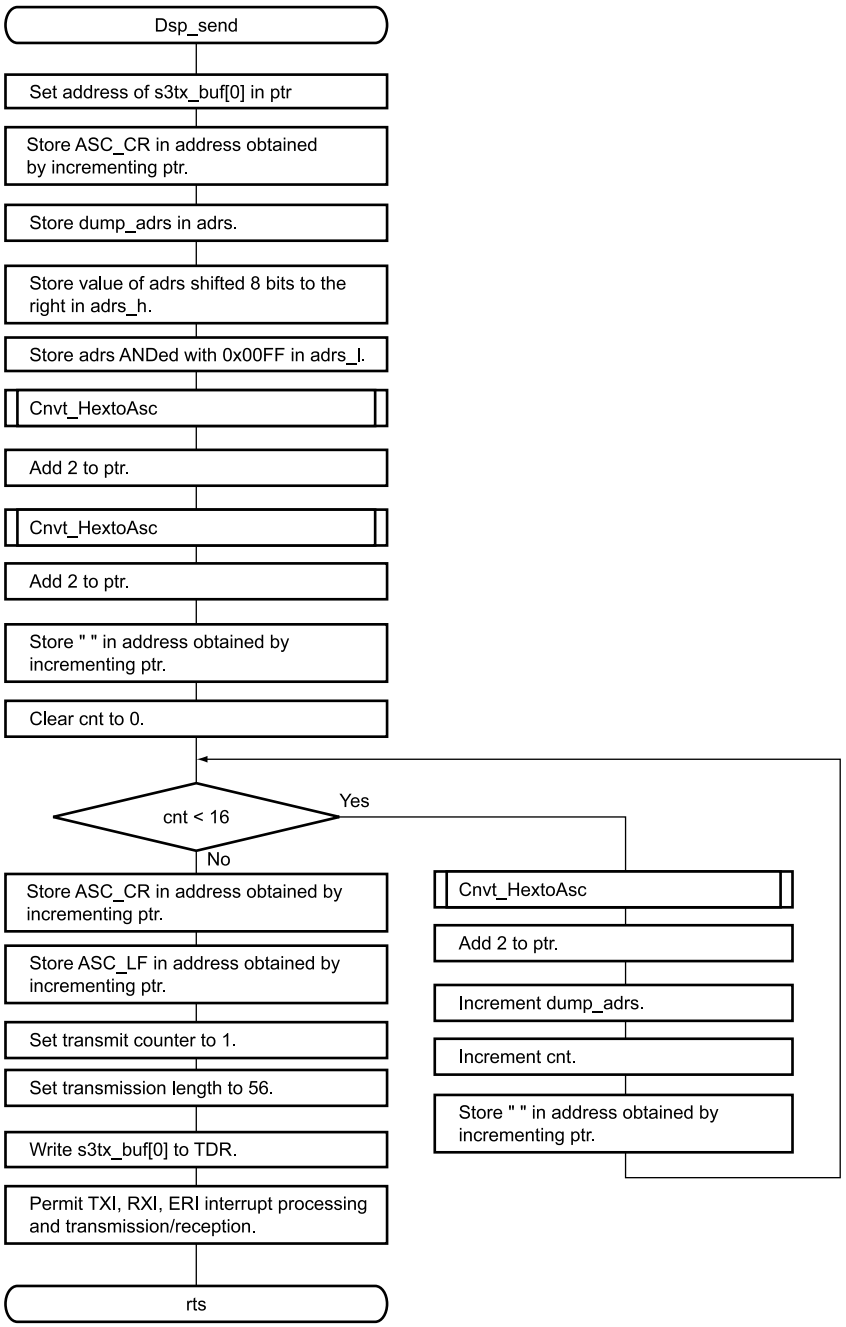


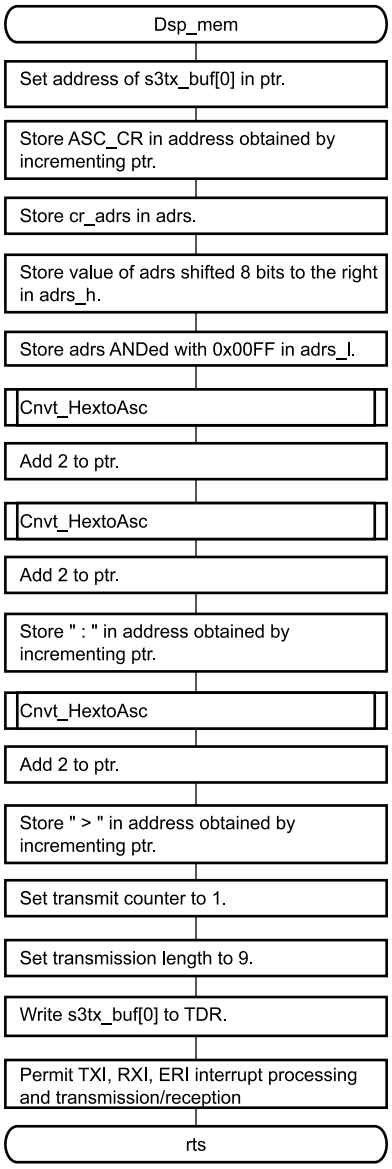

Int_s3rx
Write SCI3_RDR to work.
WORK == ASC_CN
No
Yes
WORK == ASC_BS
No
Yes
Receive counter < 13?
No
Yes
1
Receive counter > 0?
No
Yes
Decrement receive counter.
Set_send1
Clear receive counter to 0.
Dsp_idle
WORK == " ^ "
No
Yes
moni_code == E
No
Yes
moni_code = ' '
Dsp_idle
Decrement cr_adrs.
Store "E" in ans_code.
Clear receive counter to 0.
WORk == " . "
No
Yes
Clear receive counter to 0.
Store "I" in ans_code.
moni_code = ' '
Set_send1
WORK == ASC_LF
No
Yes
Clear receive counter to 0.
Set_send1
Set_send1
Clear receive counter to 0.
2
Clear SSR3_RDRF to 0.
rts

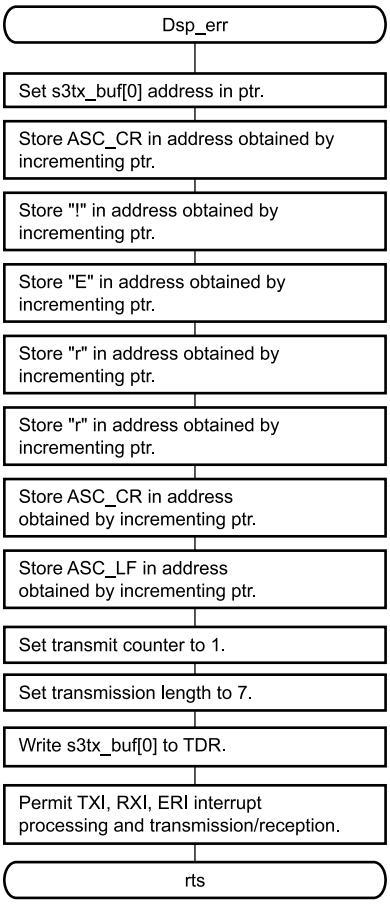

1
Transfer received data in SCI3_RDR to reception buffer.
WORK == ASC_CR
No
Yes
Receive counter == 6
No
Yes
Dsp_idle
Cnvt_AsctoAddress
Write CMem.adrs to cr_adrs.
Write receive buffer data to WORK.
Is leading character in received buffer data "D" or "d"?
No
Yes
Store "D" in moni_code.
Dump_start
Store "D" in ans_code.
Is leading character in received buffer data "E" or "e"?
No
Yes
Store "E" in moni_code.
Store "E" in ans_code.
Clear receive counter to 0.
moni_coude == D
Yes
No
moni_code == E
Yes
No
Dsp_idle
Receive counter == 1?
No
Yes
Increment cr_adrs.
Store "E" in ans_cod.
Receive counter == 3?
No
Yes
Cnvt_AsctoHEX
Store ed_data at address cr_adrs.
*cr_adrs == ed_data
No
Yes
Store "E" in ans_code.
Store "R" in ans_code.
Store "R" in ans_code.
Receive counter == 1?
No
Yes
cr_adrs + = 128
Store "D" in ans_code.
Dump_start
Store "R" in ans_code.
Clear receive counter to 0.
2

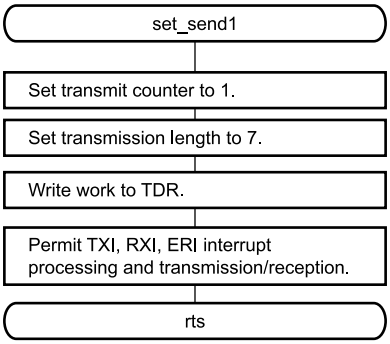

Int_s3tx
Transmit counter < transmission length?
Write value in transmission buffer specified by s3tx_cnt to TDR.
Clear TDRE to 0.
Permit receive interrupt processing, transmission and reception.
Ans_send
rts

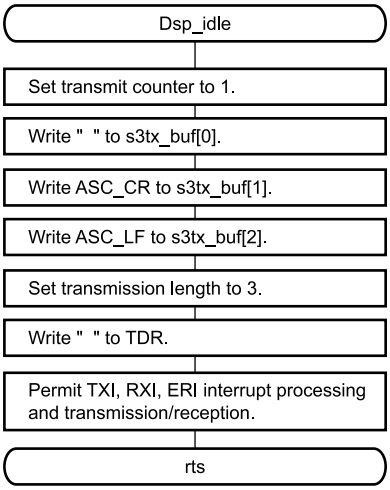

Ans_send
ans_code == "D"
No
Yes
dump_ycnt < 8
No
Yes
ans_code == "E"
No
Yes
ans_code == "R"
No
Yes
ans_code == "I"
No
Yes
Dump_send
Increment dump_ycnt.
Store " " in ans_code.
Dsp_mem
Store " " in ans_code.
Dsp_err
Store " " in ans_code.
Dsp_idle
Store " " in ans_code.
rts

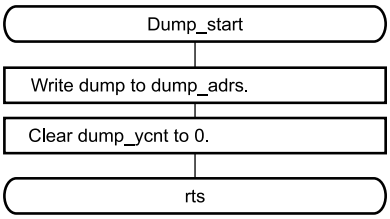

Dsp_send
Set address of s3tx_buf[0] in ptr
Store ASC_CR in address obtained by incrementing ptr.
Store dump_adrs in adrs.
Store value of adrs shifted 8 bits to the right in adrs_h.
Store adrs ANDed with 0x00FF in adrs_l.
Cnvt_HextoAsc
Add 2 to ptr.
Cnvt_HextoAsc
Add 2 to ptr.
Store " " in address obtained by incrementing ptr.
Clear cnt to 0.
cnt < 16
Yes
No
Store ASC_CR in address obtained by incrementing ptr.
Store ASC_LF in address obtained by incrementing ptr.
Set transmit counter to 1.
Set transmission length to 56.
Write s3tx_buf[0] to TDR.
Permit TXI, RXI, ERI interrupt processing and transmission/reception.
rts
Cnvt_HextoAsc
Add 2 to ptr.
Increment dump_adrs.
Increment cnt.
Store " " in address obtained by incrementing ptr.

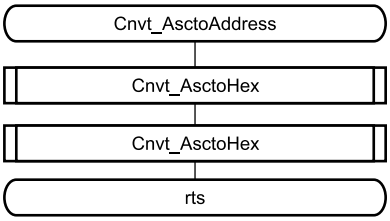

Dsp_mem
Set address of s3tx_buf[0] in ptr.
Store ASC_CR in address obtained by incrementing ptr.
Store cr_adrs in adrs.
Store value of adrs shifted 8 bits to the right in adrs_h.
Store adrs ANDed with 0x00FF in adrs_l.
Cnvt_HextoAsc
Add 2 to ptr.
Cnvt_HextoAsc
Add 2 to ptr.
Store " : " in address obtained by incrementing ptr.
Cnvt_HextoAsc
Add 2 to ptr.
Store " > " in address obtained by incrementing ptr.
Set transmit counter to 1.
Set transmission length to 9.
Write s3tx_buf[0] to TDR.
Permit TXI, RXI, ERI interrupt processing and transmission/reception
rts

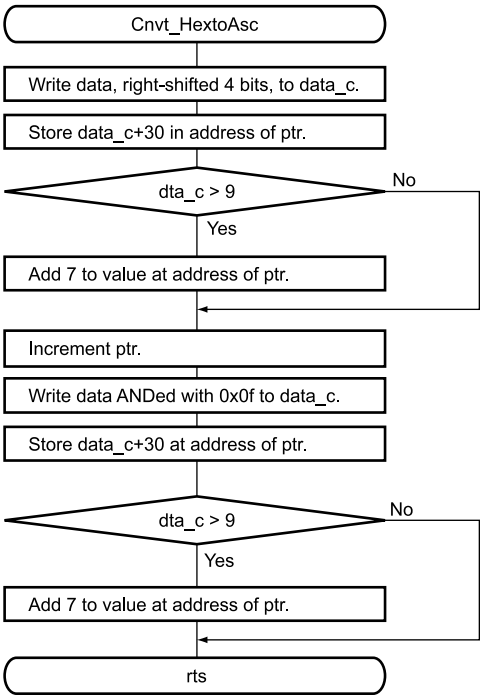

Dsp_err
Set s3tx_buf[0] address in ptr.
Store ASC_CR in address obtained by incrementing ptr.
Store "!" in address obtained by incrementing ptr.
Store "E" in address obtained by incrementing ptr.
Store "r" in address obtained by incrementing ptr.
Store "r" in address obtained by incrementing ptr.
Store ASC_CR in address obtained by incrementing ptr.
Store ASC_LF in address obtained by incrementing ptr.
Set transmit counter to 1.
Set transmission length to 7.
Write s3tx_buf[0] to TDR.
Permit TXI, RXI, ERI interrupt processing and transmission/reception.
rts

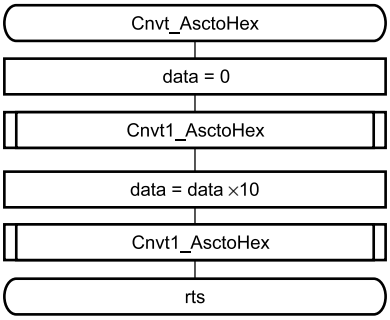

set_send1
Set transmit counter to 1.
Set transmission length to 7.
Write work to TDR.
Permit TXI, RXI, ERI interrupt processing and transmission/reception.
rts

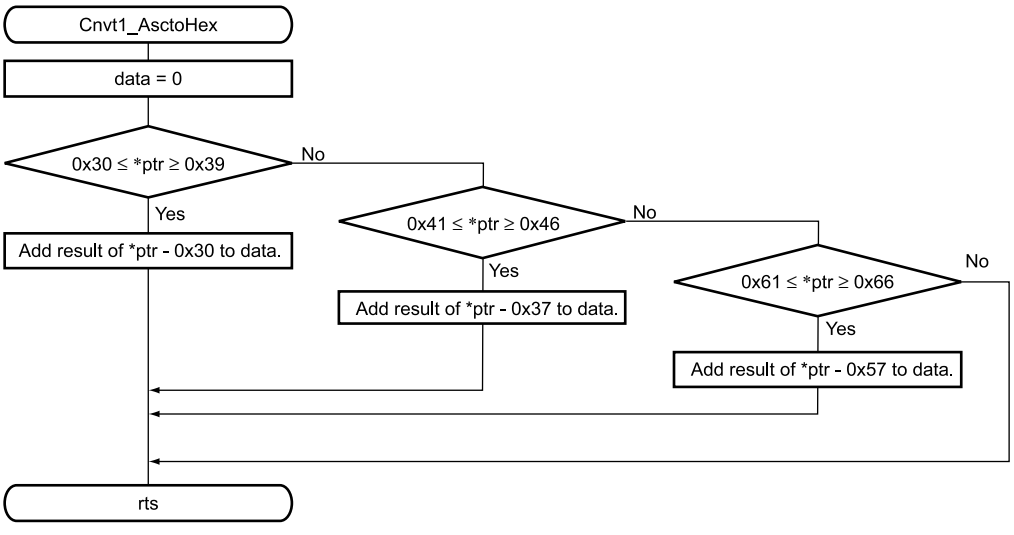

Dsp_idle
Set transmit counter to 1.
Write " " to s3tx_buf[0].
Write ASC_CR to s3tx_buf[1].
Write ASC_LF to s3tx_buf[2].
Set transmission length to 3.
Write " " to TDR.
Permit TXI, RXI, ERI interrupt processing and transmission/reception.
rts


Dump_start
Write dump to dump_adrs.
Clear dump_ycnt to 0.
rts


Cnvt_AsctoAddress
Cnvt_AsctoHex
Cnvt_AsctoHex
rts


Cnvt_HextoAsc
Write data, right-shifted 4 bits, to data_c.
Store data_c+30 in address of ptr.
dta_c > 9
No
Yes
Add 7 to value at address of ptr.
Increment ptr.
Write data ANDed with 0x0f to data_c.
Store data_c+30 at address of ptr.
dta_c > 9
No
Yes
Add 7 to value at address of ptr.
rts


Cnvt_AsctoHex
data = 0
Cnvt1_AsctoHex
data = data ×10
Cnvt1_AsctoHex
rts


Cnvt1_AsctoHex
data = 0
0x30 ≤ *ptr ≥ 0x39
No
Yes
Add result of *ptr - 0x30 to data.
0x41 ≤ *ptr ≥ 0x46
No
Yes
Add result of *ptr - 0x37 to data.
0x61 ≤ *ptr ≥ 0x66
No
Yes
Add result of *ptr - 0x57 to data.
rts

6. Program Listing

```
     .EXPORT   _INIT
     .IMPORT   _main
;
     .SECTION    P,CODE
_INIT:
     MOV.W     #H'FF80,R7
     LDC.B     #B'10000000,CCR
     JMP       @_main
;
     .END
```

```
/************************************************************************************************************************/
/*                                                                                                                      */
/*   FILE           :kmoni.c                                                                                            */
/*   DATE           :Man, Jul 11, 2001                                                                                  */
/*   DESCRIPTION    :Main Program                                                                                       */
/*   CPU TYPE       :H8/3664F                                                                                           */
/*                                                                                                                      */
/*   This file is generated by Renesas Project Generator (Ver.1.2).                                                     */
/*                                                                                                                      */
/************************************************************************************************************************/
     #include  "machine.h"
     #include  "ascdefine.h"
     #include  "string.h"

#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif

struct BIT {
     unsigned char  b7:1;       /* bit 7 */
     unsigned char  b6:1;       /* bit 6 */
     unsigned char  b5:1;       /* bit 5 */
     unsigned char  b4:1;       /* bit 4 */
     unsigned char  b3:1;       /* bit 3 */
     unsigned char  b2:1;       /* bit 2 */
     unsigned char  b1:1;       /* bit 1 */
     unsigned char  b0:1;       /* bit 0 */
};

#define SCI3_SMR    (*(volatile unsigned char *)0xFFA8)                    /* SCI3  Address                    */
#define SCI3_BRR    (*(volatile unsigned char *)0xFFA9)                    /* SCI3  Address                    */
#define SCI3_SCR    (*(volatile unsigned char *)0xFFAA)                    /* SCI3  Address                    */
#define SCI3_TDR    (*(volatile unsigned char *)0xFFAB)                    /* SCI3  Address                    */
#define SCI3_SSR    (*(volatile unsigned char *)0xFFAC)                    /* SCI3  Address                    */
#define SCI3_RDR    (*(volatile unsigned char *)0xFFAD)                    /* SCI3  Address                    */
#define ssr3        (*(struct BIT *)0xFFAC)
#define SSR3_RDRF   ssr3.b6
#define SSR3_TDRE        ssr3.b7

#define PMR1        *(volatile unsigned char *)0xFFE0

#pragma        interrupt(Int_SCI3)
#pragma        interrupt(Int_trap)
#pragma        interrupt(NMI)
```

```
unsigned char  s3rx_buf[16];
unsigned char  s3tx_buf[60];
unsigned char  s3rx_cnt;
unsigned char  s3tx_cnt;
unsigned char  s3tx_len;
unsigned char  dummy;
unsigned char  trap_flag;

unsigned char  moni_code;
unsigned char  ans_code;
unsigned char  dump_ycnt;
unsigned char  *cr_adrs;
unsigned char  *dump_adrs;
union     {
     unsigned char  data[2];
     unsigned char  *adrs;
          } CMem;

/*;*************************************************************************************************************************************/
/*;  Function definition                                                                                                               */
/*;*************************************************************************************************************************************/
extern         void INIT(void);
void           main ( void );
void           Init_sci ( void );
void           Int_SCI3( void );
void           Int_s3tx ( void );
void           Int_s3rx ( void );
void           Int_s3err( void );
void           Int_trap( void );
void           Cnvt_HextoAsc(unsigned char data ,unsigned char *ptr);
unsigned char  Cnvt_AsctoHex ( unsigned char *ptr );
unsigned char  Cnvt1_AsctoHex(unsigned char *ptr);
void           Cnvt_AsctoAddress( void );

void           Ans_send( void );
void           Set_send1 ( unsigned char work );
void           Dsp_mem( void );
void           Dsp_err( void );
void           Dsp_idle( void );
void           Dsp_init( void );
void           Dump_start( unsigned char *adrs );
void           Dump_send( void );

/*;*************************************************************************************************************************************/
/*;  Vector Address                                                                                                                    */
/*;*************************************************************************************************************************************/
#pragma   section        V1                                                  /* VECTOR SECTOIN SET                    */
void (*const VEC_TBL1[])(void) = {
/*   0x00 - 0x0f                                                                                                                       */
     INIT,                                                                 /* 00 reset                              */
     INIT
};
#pragma   section        V2                                                  /* VECTOR SECTOIN SET                    */
void (*const VEC_TBL2[])(void) = {
     Int_trap,                                                             /* 10  trap#0                            */
     Int_trap,                                                             /* 12  trap#1                            */
     Int_trap,                                                             /* 14  trap#2                            */
     Int_trap                                                              /* 16  trap#3                            */
};
#pragma   section        V3                                                  /* VECTOR SECTOIN SET                    */
void (*const VEC_TBL3[])(void) = {
     Int_SCI3,                                                             /* 2E SCI3_RX  Interrupt                 */
     Int_SCI3                                                              /* 2E SCI3_RX  Interrupt                 */
};
```

```
#pragma   section                                                                    /* P                                    */
/*;*************************************************************************************************************************/
/*;  Main Program                                                                                                          */
/*;*************************************************************************************************************************/
void main ( void )  {

     moni_code = ' ';
     ans_code = ' ';
     dump_ycnt = 8;

     PMR1 = 0x02;
     Init_sci();
     set_imask_ccr(0);                                                               /* Condition code setting               */
     Dsp_init();

     while (1) {
          ;
     }
}

void abort(void)
{

}
/*-------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Init_sci                                                                                          */
/*   <Function task>     Serial setting                                                                                    */
/*   <Argument>          None                                                                                              */
/*   <Return value>      None                                                                                              */
/*   <External variable> None                                                                                              */
/*   <Function used>     None                                                                                              */
/*-------------------------------------------------------------------------------------------------------------------------*/
void Init_sci ( void )   {

     unsigned long *plPtr_0;
     unsigned long *plPtr_1;
     unsigned char  speed;


/*-------------------------------------------------------------------------------------------------------------------------*/
/*   <Communication with UNIT>                                                                                             */
/*   SC register                 7    6    5    4    3    2    1    0                                                     */
/*   Serial status       (SSR): TDRE RDRF ORER FER  PER  ---  ---  ---                                                     */
/*   Serial mode         (SMR): C/A  8/7  PE   O/E  STP  MP   CKS1 CKS0                                                    */
/*                              0    0    0    0    0    0    0    0                                                     */
/*   Bit rate            (BRR):                                                                                            */
/* Serial control        (SCR): TIE  RIE  TE   RE   MPI  TEI  CKE1 CKE0                                                    */
/*                              1    1    1    1    0    0    0    0                                                     */
/*-------------------------------------------------------------------------------------------------------------------------*/

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
     SCI3_SCR = 0xf0;
     SCI3_BRR = 51;                                                                  /*  9600bps CLK:16MHz                   */

     SCI3_SMR = 0x00;                                                                /* PE:0                                 */

     s3rx_cnt = 0;

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Int_SCI3                                                                                               */
/*   <Function task>     Serial interrupt processing                                                                            */
/*   <External variable>                                                                                                        */
/*   <Function used>     Int_s3rx,Int_s3tx,int_s3err                                                                            */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Int_SCI3( void )    {
 unsigned char work;

     work = SCI3_SSR;
     if ((work & 0x38) != 0)  {
          Int_s3err();
     }
     else {
          if ((work & 0x40) != 0)  {
               Int_s3rx();
          }
          if ((work & 0x80) != 0)  {
               Int_s3tx();
          }
     }
}

/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>          Int_s3tx                                                                                          */
/*   <Function task>     Serial transmission processing                                                                         */
/*   <External variable> s1tx_cnt,s1tx_len                                                                                      */
/*   <Function used>     Ans_send                                                                                               */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Int_s3tx( void )    {

     if (s3tx_cnt < s3tx_len) {
          SCI3_TDR = s3tx_buf[s3tx_cnt++];
          SSR3_TDRE = 0;
     }
     else {
          SCI3_SCR = 0x70;                                                    /* TEIE:0                                    */
          Ans_send();
     }
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>          Int_s3rx                                                                                          */
/*   <Function task>     Serial receive processing                                                                              */
/*   <External variable> s3rx_buf,s3rx_cnt,                                                                                     */
/*   <Function used>     Dsp_idle,Set_send1,Cnvt_AsctoAddress,Dump_start,                                                       */
/*                  Cnvt_AsctoHex                                                                                               */
/*------------------------------------------------------------------------------------------------------------------------------*/
/*------------------------------------------------------------------------------------------------------------------------------*/
/*                        1   2   3   4   5   6   7   8   9   10  11  12                                                        */
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   memory Dump    : D | x   x   x   x   CR                                                                                   */
/*                  : CR                                                                                                        */
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   memory Edit    : M | x   x   x   x   CR                                                                                   */
/*                  : CR                                                                                                        */
/*                  : ^                                                                                                         */
/*                  : x   x     CR                                                                                              */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Int_s3rx( void )    {
 unsigned char work,ed_data;

     work = SCI3_RDR;
```

```
if (work == ASC_CN) {
     s3rx_cnt = 0;
     Dsp_idle();
}
else if (work == ASC_BS) {
     if (s3rx_cnt > 0)   {
          s3rx_cnt--;
          Set_send1(work);
     }
}
else if (s3rx_cnt < 13) {
     s3rx_buf[s3rx_cnt++] = work;

     if (work == ASC_CR) {
          if (s3rx_cnt == 6)  {
               Dsp_idle();
               Cnvt_AsctoAddress();
               cr_adrs = (unsigned char *)CMem.adrs;
               work = s3rx_buf[0];
               if (work == 'D' || work == 'd') {
                    moni_code = 'D';
                    Dump_start(cr_adrs);
                    ans_code = 'D';
               }
               else if (work == 'E' || work == 'e') {
                    moni_code = 'E';
                    ans_code = 'E';
               }
          s3rx_cnt = 0;
          }
          else {
               switch (moni_code) {
                    case 'D':                                          /*--- Dump---*/
                         if (s3rx_cnt == 1) {
                              cr_adrs+=128;
                              ans_code = 'D';
                              Dump_start(cr_adrs);
                         }
                         else {
                              ans_code = 'R';
                         }
                         break;
                    case 'E':                                          /*--- Edit ---*/
                         Dsp_idle();
                         if (s3rx_cnt == 1) {
                              cr_adrs++;
                                  ans_code = 'E';
                         }
                         else if (s3rx_cnt == 3) {
                              ed_data = Cnvt_AsctoHex(s3rx_buf);
                              *cr_adrs = ed_data;
                              if (*cr_adrs == ed_data) {
                                  ans_code = 'E';
                              }
                              else {
                                  ans_code = 'R';
                              }
                         }
                         else {
                                  ans_code = 'R';
                         }
                         break;
                    default:                                           /*--- idle ---*/
                         break;
               }
```

```
                    s3rx_cnt = 0;
               }
          }
          else if (work == '^')    {
               if (moni_code == 'E') {
                    Dsp_idle();
                    cr_adrs--;
                    ans_code = 'E';
               }
               else {
                    moni_code = ' ';
               }
               s3rx_cnt = 0;
          }
          else if (work == '.')    {
               s3rx_cnt = 0;
               ans_code = 'I';
               moni_code = ' ';
               Set_send1(work);
          }
          else if (work == ASC_LF) {
               s3rx_cnt = 0;
               Set_send1(work);
          }
          else {
               Set_send1(work);
          }
     }
     else {
          s3rx_cnt = 0;
     }

     SSR3_RDRF = 0;
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   Set_send                                                                                                                   */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Set_send1 ( unsigned char work ) {

     s3tx_cnt = 1;
     s3tx_len = 1;
     SCI3_TDR = work;
     SCI3_SCR = 0xf0;
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Ans_send                                                                                               */
/*   <Function task>     Answer-back data transmission                                                                          */
/*   <Argument>          None                                                                                                   */
/*   <Return value>      None                                                                                                   */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                             */
/*   <Function used>     Dump_send,Dsp_mem,Dsp_mcng,Dsp_err,Dsp_idle                                                             */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Ans_send ( void )   {

     switch (ans_code) {
          case 'D':                                                    /*--- Dump ---*/
               if (dump_ycnt < 8)  {
                    Dump_send();
                    dump_ycnt++;
               }
               else {
                    ans_code = ' ';
               }
```

```
                break;
            case 'E':                                                   /*--- Edit ---*/
                Dsp_mem();
                ans_code = ' ';
                break;
            case 'R':                                                   /*--- Err ---*/
                Dsp_err();
                ans_code = ' ';
                break;
            case 'I':                                                   /*--- Idle ---*/
                Dsp_idle();
                ans_code = ' ';
                break;
            default:                                                    /*---  ---*/
                break;
        }
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dsp_mem                                                                                                */
/*   <Function task>     Memory                                                                                                 */
/*   <Argument>          None                                                                                                   */
/*   <Return value>      None                                                                                                   */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                             */
/*   <Function used>     Cnvt_HextoAsc                                                                                          */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Dsp_mem( void )     {
 unsigned char *ptr,adrs_h,adrs_l;
 unsigned short     adrs;

     ptr = &s3tx_buf[0];
     *ptr++ = ASC_CR;
     adrs = (unsigned short)cr_adrs;
     adrs_h = (unsigned char)(adrs >> 8);
     adrs_l = (unsigned char)(adrs & 0x00ff);
     Cnvt_HextoAsc(adrs_h,ptr);
     ptr+=2;
     Cnvt_HextoAsc(adrs_l,ptr);
     ptr+=2;
     *ptr++ = ':';
     Cnvt_HextoAsc(*cr_adrs,ptr);
     ptr+=2;
     *ptr++ = '>';

     s3tx_cnt = 1;
     s3tx_len = 9;
     SCI3_TDR = s3tx_buf[0];
     SCI3_SCR = 0xf0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dsp_idle                                                                                         */
/*   <Function task>     Error display                                                                                    */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                       */
/* <Function used>       None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_idle( void )    {

          s3tx_cnt = 1;
          s3tx_buf[0] = ' ';
          s3tx_buf[1] = ASC_CR;
          s3tx_buf[2] = ASC_LF;
          s3tx_len = 3;
          SCI3_TDR = ' ';
          SCI3_SCR = 0xf0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dsp_init                                                                                         */
/*   <Function task>     Initial window display                                                                           */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                       */
/*   <Function used>     None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_init( void )    {

     s3tx_cnt = 0;
     strcpy((char *)s3tx_buf,"H8/3664 Monitor Program Ver1.0\n\r");
     s3tx_len = 33;
     SCI3_TDR = ' ';
     SCI3_SCR = 0xf0;
}

/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dsp_err                                                                                          */
/*   <Function task>     Error display                                                                                    */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                       */
/*   <Function used>     None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_err( void )     {
 unsigned char *ptr;

     ptr = &s3tx_buf[0];
     *ptr++ = ASC_CR;
     *ptr++ = '!';
     *ptr++ = 'E';
     *ptr++ = 'r';
     *ptr++ = 'r';
     *ptr++ = ASC_CR;
     *ptr++ = ASC_LF;

     s3rx_cnt = 0;
     s3tx_cnt = 1;
     s3tx_len = 7;
     SCI3_TDR = s3tx_buf[0];
     SCI3_SCR = 0xf0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dump_start                                                                                             */
/*   <Function task>     Memory dump start                                                                                      */
/*   <Argument>          None                                                                                                   */
/*   <Return value>      None                                                                                                   */
/*   <External variable> s0tx_cnt,s0tx_len,s0tx_buf                                                                             */
/*   <Function used>     None                                                                                                   */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Dump_start( unsigned char *adrs ) {
     dump_adrs = adrs;
     dump_ycnt = 0;
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Dump_send                                                                                              */
/*   <Function task>     Memory dump data transmission                                                                          */
/*   <Argument>          None                                                                                                   */
/*   <Return value>      None                                                                                                   */
/*   <External variable> s3tx_cnt,s3x_len,s3tx_buf                                                                              */
/*   <Function used>     Cnvt_HextoAsc                                                                                          */
/*------------------------------------------------------------------------------------------------------------------------------*/
/*------------------------------------------------------------------------------------------------------------------------------*/
/*           1         2         3         4         5                                                                        */
/*   12345678901234567890123456789012345678901234567890123                                                                     */
/*   0000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F                                                                      */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Dump_send( void )   {
 unsigned char *ptr,adrs_h,adrs_l,cnt;
 unsigned short     adrs;

          ptr = &s3tx_buf[0];
          *ptr++ = ASC_CR;
          adrs = (unsigned short)dump_adrs;
          adrs_h = (unsigned char)(adrs >> 8);
          adrs_l = (unsigned char)(adrs & 0x00ff);
          Cnvt_HextoAsc(adrs_h,ptr);
          ptr+=2;
          Cnvt_HextoAsc(adrs_l,ptr);
          ptr+=2;
          *ptr++ = ' ';
          cnt = 0;
          while (cnt <16)     {
               Cnvt_HextoAsc(*dump_adrs,ptr);
               ptr+=2;
               dump_adrs++;
               cnt++;
               *ptr++ = ' ';
          }
          *ptr++ = ASC_CR;
          *ptr++ = ASC_LF;

          s3tx_cnt = 1;
          s3tx_len = 56;
          SCI3_TDR = s3tx_buf[0];
          SCI3_SCR = 0xf0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Cnvt_HextoAsc(0x30, 0x39,0x41, 0x46)                                                                   */
/*   <Function task>     hexdata(1byte) >ASCIIdata(2byte) Processing for converting                                             */
/*   <Argument>          hexdata,(unsigned char *)ascii_data                                                                    */
/*   <Return value>      None                                                                                                   */
/*   <External variable> None                                                                                                   */
/*   <Function used>     None                                                                                                   */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Cnvt_HextoAsc(unsigned char data ,unsigned char *ptr) {
 unsigned char data_c;

     data_c = data >> 4;
     *ptr = data_c + 0x30;
     if (data_c > 9)
          *ptr += 7;

     ptr++;
     data_c = data & 0x0f;
     *ptr = data_c + 0x30;
     if (data_c > 9)
          *ptr += 7;
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Int_s3err                                                                                              */
/*   <Function task>     Serial 0 receive error processing                                                                      */
/*   <External variable> s3rx_cnt                                                                                               */
/*   <Function used>     None                                                                                                   */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Int_s3err( void )   {

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
     s3rx_cnt = 0;
}
/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>          Int_trap                                                                                          */
/*   <Function task>     Trap interrupt processing                                                                              */
/*   <External variable> trap_flag                                                                                              */
/*   <Function used>     None                                                                                                   */
/*------------------------------------------------------------------------------------------------------------------------------*/
void Int_trap( void )    {

     while (trap_flag == 0)
          ;
     trap_flag = 0;
}

/*------------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Cnvt_AsctoHex                                                                                          */
/*   <Function task>     2byte ascii code => 1byte (7-4bit,3-0bit)                                                              */
/*   <Argument>          Pointer of ASCII code character string                                                                 */
/*   <Return value>      1byte(Hex)                                                                                             */
/*   <External variable> None                                                                                                   */
/*   <Function used>     Cnvt1_AsctoHex                                                                                         */
/*------------------------------------------------------------------------------------------------------------------------------*/
unsigned char Cnvt_AsctoHex(unsigned char *ptr) {
     unsigned char  data = 0;

     data = Cnvt1_AsctoHex(ptr++);
     data *= 0x10;
     data += Cnvt1_AsctoHex(ptr);

     return(data);
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Cnvt1_AsctoHex                                                                                   */
/*   <Function task>     1byte ascii code => 1byte(bit3-0)                                                                */
/*   <Argument>          Pointer of ASCII code character string                                                           */
/*   <Return value>      1byte(bit3-0)                                                                                    */
/*   <External variable> None                                                                                             */
/*   <Function used>     None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
/*   ascii     0   -   9       A   B   C   D   E   F                                                                    */
/*   hex       30  -  39      41  42  43  44  45  46                                                                    */
/*------------------------------------------------------------------------------------------------------------------------*/
unsigned char Cnvt1_AsctoHex(unsigned char *ptr) {
     unsigned char  data = 0;

     if ((0x30 <= *ptr) && (*ptr <= 0x39)) {
          data += *ptr - 0x30;
     }
     else if ((0x41 <= *ptr) && (*ptr <= 0x46)) {
          data += *ptr - 0x37;
     }
     else if ((0x61 <= *ptr) && (*ptr <= 0x66)) {
          data += *ptr - 0x57;
     }
return(data);
}

/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Cnvt_AsctoAddress(0x30□0x39,0x41□0x46)                                                           */
/*   <Function task>     ASCIIdata(4byte)->Processing for converting the reference address(2bytes)                        */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> CMemAddress,s3rx_buf                                                                             */
/*   <Function used>     Cnvt_AsctoHex                                                                                    */
/*------------------------------------------------------------------------------------------------------------------------*/
void Cnvt_AsctoAddress( void )
{
                    /* Return address conversion(for every two bytes)                                                   */
                    /* buf[1][2][3][4]                                                                                  */
                    /*     -  -  -  -   x  x  x  x                                                                      */
     CMem.data[0] = Cnvt_AsctoHex(&s3rx_buf[1]);
     CMem.data[1] = Cnvt_AsctoHex(&s3rx_buf[3]);
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     NMI                                                                                              */
/*   <Function task>     NMI interrupt                                                                                    */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> None                                                                                             */
/*   <Function used>     None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void NMI(void) {

}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <Function name>     Int_s3te                                                                                         */
/*   <Function task>     SCI1TEI interrupt                                                                                */
/*   <Argument>          None                                                                                             */
/*   <Return value>      None                                                                                             */
/*   <External variable> None                                                                                             */
/*   <Function used>     None                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void Int_s3te(void) {

}
```

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Sep.29.03	—	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.