Hitachi America, Ltd.

Application Note
H8/300 SCI

Cristian Tomescu

Asynchronous Serial Communication

INTRODUCTION

Binary information can be transmitted either in parallel or serially. Although data transmission can be accomplished much quicker using a parallel communication scheme, it obviously requires more lines between the source and the destination. So for the sake of simplicity, especially for long distance communications, a serial, 2-wire channel for transmitting and receiving data is quite adequate. Such a typical setup is between some kind of Data Terminal Equipment (such as a PC or an embedded system) and a Data Communication Equipment device (such as a modem) via a standard RS-232C serial interface.

Hitachi's H8/300 family microcontrollers support an on-chip Serial Communication Interface (SCI), providing a dual channel for low to moderately high speed data transfers. Each SCI channel is configurable to either asynchronous or clocked synchronous formats and can be driven either from the on-chip baud rate generator or from an external source. This note details the SCI operation for the H8/330 and H8/320 microcontroller devices in the asynchronous mode where communication speed is not a prime factor, but asynchronous operation is.

THE ASYNCHRONOUS DATA FORMAT

Asynchronous operation offers the advantage of a simple and inexpensive communication protocol in applications that do not require clock synchronized receive-transmit operations and where the transfer speed is not critical. Because of relatively low transfer rates, the potential for communication errors is low and better error checking methods are provided. For example, the H8/300 family supports byte synchronous transfers up to 2.5 million bits per second but only 305K bits per second can be achieved in the asynchronous mode. However, if the application involves a human operator entering ASCII characters on a keyboard to a terminal, the data transfer speed will be obviously limited by the operator's typing dexterity.

Data transfers can occur at any time if the communication channels are enabled, and one character at a time. Since the SCI has independent transmit and receive channels, simultaneous data transfers are possible (full duplex). Transfers are initiated by a high-to-low transition at one of the 2 communication links, TxD (transmit data output pin) or RxD (receive data input pin), between the idle state and the low level of the start bit. The character data or frame is composed of a low level start bit, 7 or 8 bits of actual data, an optional parity bit, and 1 or 2 high level stop bits indicating end of the frame. The least significant data bit (LSB) is transferred first. Addition of a parity bit determines if the total number of 1's in the frame is even or odd. For instance, if the frame had an even number of ones already, a low parity bit is sent in order to generate even parity; if odd parity is desired, a high bit is transmitted instead. The stop bit is high, and is indistinguishable from the high signal that is sent when no frame is being transmitted. In other words, if the frame has "n" stop bit(s), it means that the next frame must wait at least that long after the last frame bit or parity bit of the previous message has been sent before it can begin sending its start bit again. Figure 1 shows the asynchronous data format supported by the H8/300 SCI:

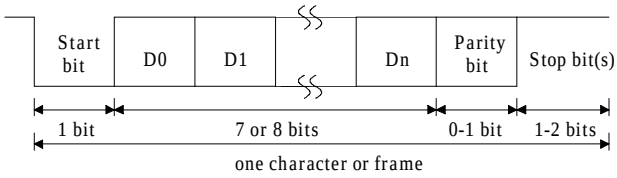

Figure 1. Asynchronous Data Format.

ASYNCHRONOUS MODE INITIALIZATION

The asynchronous mode is entered when bit 7, the C/A bit in the Serial Mode Register (SMR), is cleared to 0. The communication protocol is established by setting bits 3 through 6 as desired. Bit 6 determines the frame size, bit 5 decides whether or not a parity bit is added, bit 4 selects either odd or even parity, and bit 3 determines the number of stop bits indicating end-of-frame. Table 1 shows all possible data transfer formats as determined by

bits 3-6. The Error Detection section on page 3 discusses the function of the parity bits in more detail.

7	6	5	4	3	2	1	0
C/A	CHR	PE	O/E	STOP	----	CKS1	CKS0
0	↕	↕	↕	↕	X	↕	↕

Figure 2. Serial Mode Register (SMR).

CHR	PE	O/E	STOP	Data Size	Parity bit	Stop bits
0	0	X	0	8bits	none	1
0	0	X	1	8bits	none	2
0	1	0	0	8bits	1(even)	1
0	1	0	1	8bits	1(even)	2
0	1	1	0	8bits	1(odd)	1
0	1	1	1	8bits	1(odd)	2
1	0	X	0	7bits	none	1
1	0	X	1	7bits	none	2
1	1	0	0	7bits	1(even)	1
1	1	0	1	7bits	1(even)	2
1	1	1	0	7bits	1(odd)	1
1	1	1	1	7bits	1(odd)	2

Table 1. Data formats.

The data transfer speed, or bit rate, is determined by the baud-rate generator clock, which in turn is based on the external oscillator frequency, the pre-scaled internal CPU clock frequency, and the value programmed into the Bit Rate Register (BRR). Each bit in the data frame is 16 internal clocks wide, and is sampled midway on the rising edge of the baud generator clock. Figure 3 below further exemplifies these relationships:

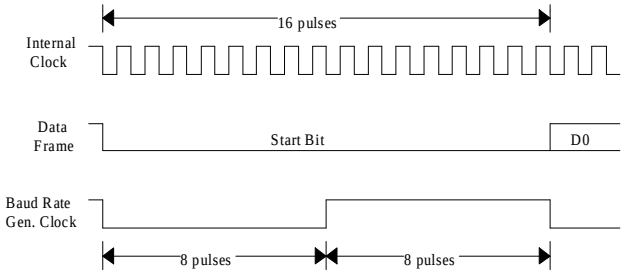

Figure 3. Communication Data vs. Clock

Bits 1 and 0 (CKS1 and 0) of the SMR determine the internal clock frequency as shown below in table 2:

n	CKS1	CKS0	Internal Clock
0	0	0	Φ
1	0	1	Φ/4
2	1	0	Φ/16
3	1	1	Φ/64

Table2. Clock Options.

The baud generator clock rate can be output at the ASCK pin of the controller by setting bit 0 (CKE0) of the Serial Control Register (SCR) to 1. Tables in the SCI section of the hardware manuals give BRR values for common bit rates and oscillator frequencies. When using the tables, it is important to note that the oscillator frequency shown refers to the speed of the external (XTAL) crystal, and not to the system clock (CPU clock). Due to the integer requirement of the BRR, not all internal clock frequencies are available for the full range of XTAL frequencies. In most cases, however, a setting is available to produce an internal clock rate very close to the desired frequency. The bit rate is given by the equation below:

$$\text{Bit rate} = \text{osc freq (sec)} / [64 \times 2^{2n} \times (N+1)] \qquad \text{eq.1}$$

where:
osc = crystal oscillator frequency (XTAL)
N = the value loaded into BRR
n = the internal clock source, according to table 2.

Alternatively, custom bit rates can be produced by determining the BRR value for a specific rate. Equation 1 converts to:

$$N = \text{osc freq (sec)} / [64 \times 2^{2n} \times \text{bit rate}] - 1 \qquad \text{eq.2}$$

For example, for a bit rate = 4800 and using a XTAL freq = 10MHz = 10×10^6 (n =0), the BRR value will be:

$N = (10 \times 10^6) / [64 \times 1 \times 4800] - 1 = 33$

Decimal BRR values should be rounded to the nearest integer. This integer value can be used to calculate the exact frequency from equation 1.

Alternatively, the SCI can be also driven by an external clock through the ASCK pin of the controller. This condition is selected by setting bit 1 (CKE1) of the Serial Control Register to 1. In this case, the clock frequency should equal the desired bit rate x 16.

TRANSMIT OPERATION

The transmitter is enabled by setting the transmit enable (TE) bit in the Serial Control Register (SCR) to 1. The I/O line to which ATxD is multiplexed automatically

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

switches to output and the pin is pulled high regardless of the corresponding bit values in the port data and data direction registers. Initially, the default contents of the Transmit Data Register (TDR), a frame of all 1's, will be sent first through the serial channel. Effective data transmission begins right after, when the first data byte is loaded into TDR and its contents automatically moved into the Transmit Shift Register (TSR). Starting with the least-significant bit, each bit is shifted from the TSR onto the ATxD line on each rising edge of the serial clock. When the most significant bit has been shifted out, the TSR is automatically re-loaded with the next data byte stored in the TDR, and transmission starts again. To end transmission, the TE bit in the SCR register must be cleared. In order to insure proper operation, the programmer must be aware of the following details:

1. The transmit data register empty (TDRE) bit of the Serial Status Register (SSR) must be cleared right after the transmit data byte has been loaded into the TDR in order to avoid data corruption. Since this bit is set when the TDR contents are transferred to the TSR, the programmer can easily control the transmission data flow by inserting a conditional loop that waits for the current data to be sent out to the TSR before the next data byte is loaded into the TDR. Figure 4 illustrates a standard transmission process.
2. If the TDR is not re-loaded before the TSR is emptied, the ATxD line will be held at the logic level of the last bit transmitted.

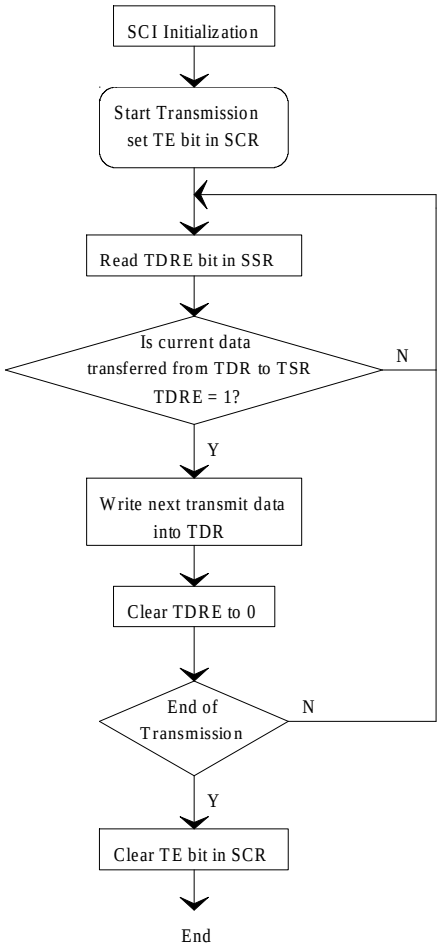

Figure 4. Transmit data flowchart.

RECEIVE OPERATION

The receiver is enabled by setting the receive-enable (RE) bit in the SCR to 1. The I/O line to which ARxD is multiplexed automatically switches to input regardless of the contents of the corresponding bit values in the port data and data direction registers. Receiving begins upon detection of the start bit at the ARxD pin, and the data bits are latched and shifted into the Receive Shift Register (RSR), starting with the most significant bit, at the bit rate of the external transmitter device. When 8 bits have been received, the byte is automatically transferred from the RSR into the Receive Data Register (RDR). The user software will normally read out the received data from the RDR, and the receive process continues. To end receiving, the RE bit in the SCR must be cleared. For proper operation, the programmer must be aware of the following details:

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

1. The ARxD input pin must be initialized high by the software before the receive function is enabled so that the low-level start bit can be detected at the pin during the high-to-low transition.
2. The receive data register full (RDRF) bit of the SSR must be cleared after the receive data byte has been read from the RDR by the user software in order to avoid reading incorrect data. Since this bit is set when the contents of the RSR are transferred into the RDR, the programmer can easily insure proper receive data flow by inserting a conditional loop that waits for the current incoming serial data to be shifted into the RSR and moved into the RDR. Figures 5 and 6 describe the standard receive process.

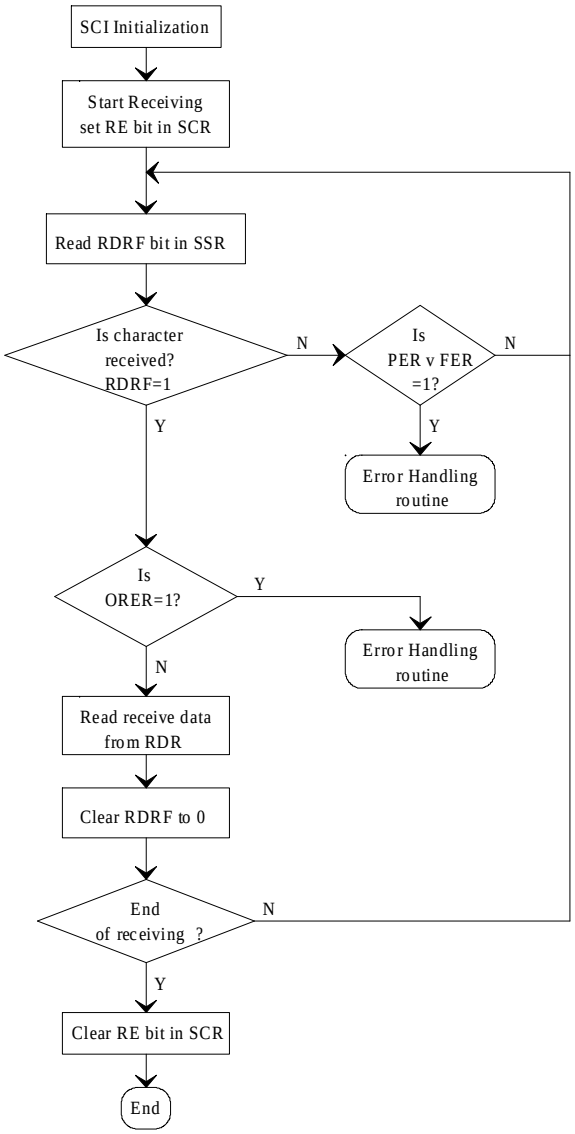

Figure 5. Receive data flowchart.

ERROR DETECTION

The SCI module can avoid receiving erroneous information by providing automatic error checks embedded in the asynchronous data stream. Each data byte is compared to the pre-defined communication protocol set in the Serial Mode Register (SMR) and scrutinized based upon the 3 following error condition flags in the SSR:

1. A **receive overrun error** occurs if the RDR is full and the RDRF bit is not cleared before 8 more bits have been shifted into the RSR. This condition is indicated when the serial control register bit (ORER) is set to 1. In this situation, the last byte of data is not transferred from the RSR to the RDR, and data reception stops until the ORER bit is cleared. If the receive-end interrupt is enabled, a RxI interrupt is issued to the CPU and user software can be executed to allow the SCI to catch up with the incoming data stream.
2. A **parity error** occurs if the total number of data bits of value 1 in the frame is different than the defined condition set in the SMR during SCI initialization by the parity mode (O/E) bit. For example, if the parity bit is supposed to indicate an even number of 1's, and a frame with an odd number of 1's is received, the parity error flag (PER) is set to 1. Usually, this condition indicates that one of the frame bits or the parity bit was changed due to noise. The contents of the RSR are transferred to the RDR, but the RDRF bit is not set to 1 until the PER bit is cleared by software, indicating that the error condition has been corrected. If the receive-end interrupt is enabled, a RxI interrupt is issued to the CPU and user software can be executed to remove the error condition. If the parity enable (PE) bit was cleared during the SCI initialization, this error type will not be recognized.
3. A **framing error** occurs when a stop bit is expected, but a low signal is received instead. In this case, the framing error receive (FER) bit is set to 1. This condition usually indicates that the receiver is using a wrong clock rate, either because the user selected the wrong value in SMR and BRR, or because the receiver oscillator is out of calibration. In addition, if the external transmitter is faulty and sending frames before the stop bits have been timed out, the FER flag will be set as well. The contents of the RSR are transferred to the RDR, but the RDRF bit is not set to 1 until the FER bit is cleared by software, indicating that the error condition has been removed. If the receive-end interrupt is enabled, the RxI

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

interrupt is issued and user software can be executed to remove the error condition.

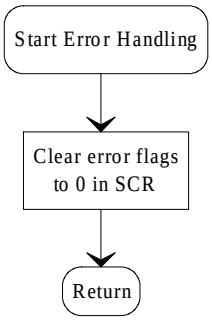

Figure 6. Error handling flowchart.

INTERRUPT CONTROL

The SCI module can be set to request 3 types of interrupts based upon the following conditions:

1. **Transmit-end interrupt (TxI)** is enabled by setting the transmit interrupt enable (TIE) bit of the SCR to 1, and is requested when the transmit data register empty (TDRE) bit in the Serial Status Register (SSR) is set to 1. This signifies that the Transmit Data Register (TDR) is empty and it is ready to receive the next byte.
2. **Receive-end Interrupt (RxI)** is enabled by setting the receive interrupt enable (RIE) bit of the SCR to 1, and is requested when the receive data register full (RDRF) bit in the SSR is set to 1. This signifies that the Receive Data Register (RDR) has just been loaded with one character byte.
3. **Receive-error Interrupt (ERI)** is enabled by setting the RIE bit to 1 as well. It is requested if any of the 3 error flags, ORER, FER, and PER are set to 1.

If both receiver and transmitter are operating independently by interrupt control and simultaneous interrupts occur, the receiver interrupt takes priority over the transmitter. If any error flags are set, the ERI interrupt request will be serviced first. Within the H8/300 architecture, the internal interrupts generated from the SCI take a low priority. In order to guarantee proper data transfers, the CPU must be available and able to update the SCI data registers, status bits, and pointers before the next byte has been transmitted or received, as explained in the previous 2 sections. If external, timer, or Dual-port RAM-generated interrupts are enabled, the designer must ensure this timing requirement can be met.

APPLICATION EXAMPLE

The most efficient way of illustrating the SCI operation is, as always, by program example. The H8/330 eval board is used as a target system, and the chip will operate in mode 3 (single-chip mode). Please refer to the H8/330 Evaluation Board User's Manual for a complete description of its features.

A sequence of ASCII character bytes located in a certain area of the on-chip ROM will be serially transmitted via the RS-232 channel to a data terminal which will display the character string. After all initializations are performed, the CPU will enter the sleep mode, waiting for interrupts to occur. Transmission will be triggered by a falling edge at the external interrupt line IRQ1 caused by pushing an external manual switch. The Start Transmission Routine will send out the first character, and will also enable the transmit-end interrupt (TxI) and disable the receive-end interrupt (RxI). Hence, ther rest of the characters will be transmitted through the successive execution of the Transmit Interrupt Service Routine. The end of the transmission will be detected by a null character at the end of the data string, at which time the TxI will be disabled and the start transmission interrupt (IRQ1) and RxI will be enabled. The CPU will re-enter the sleep mode, waiting either for an IRQ1 or a RxI. If both requests happen to occur at the same time, IRQ1 will take precedence because of higher priority. Receiving will be initiated by actuating a key on a keyboard connected to the data terminal which will transmit the pressed character ASCII code into the H8/330 receiver via the RS-232 channel. The pressed character will be echoed and displayed on a data terminal as well. The receive data will be stored into a pre-defined area of the internal ROM memory until a period character is detected, upon which the CPU will return to the sleep mode.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

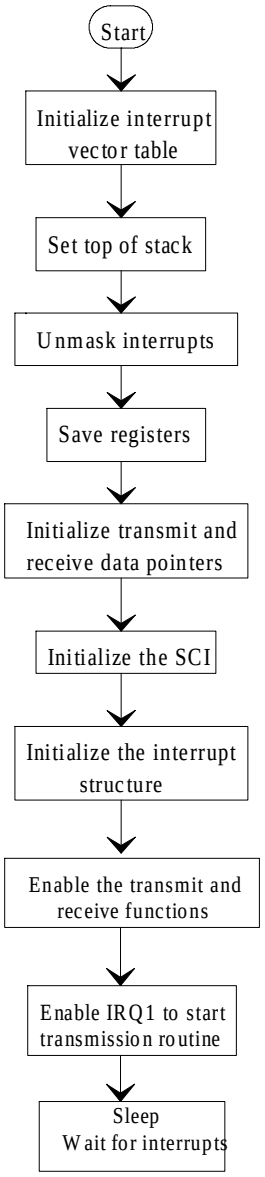

Figure 7. Main Program.

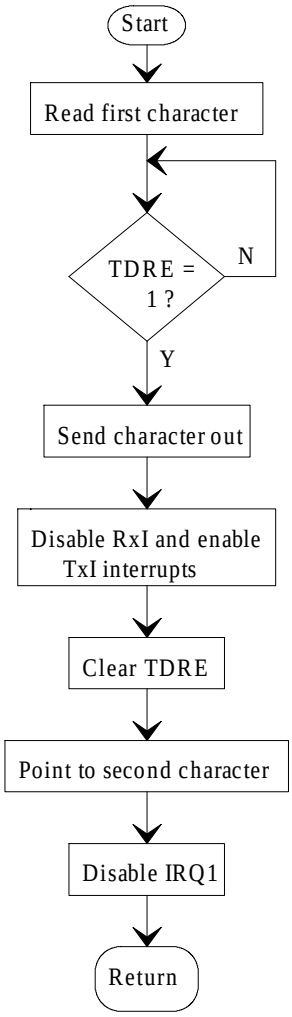

Figure 8. Start Transmission Routine.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

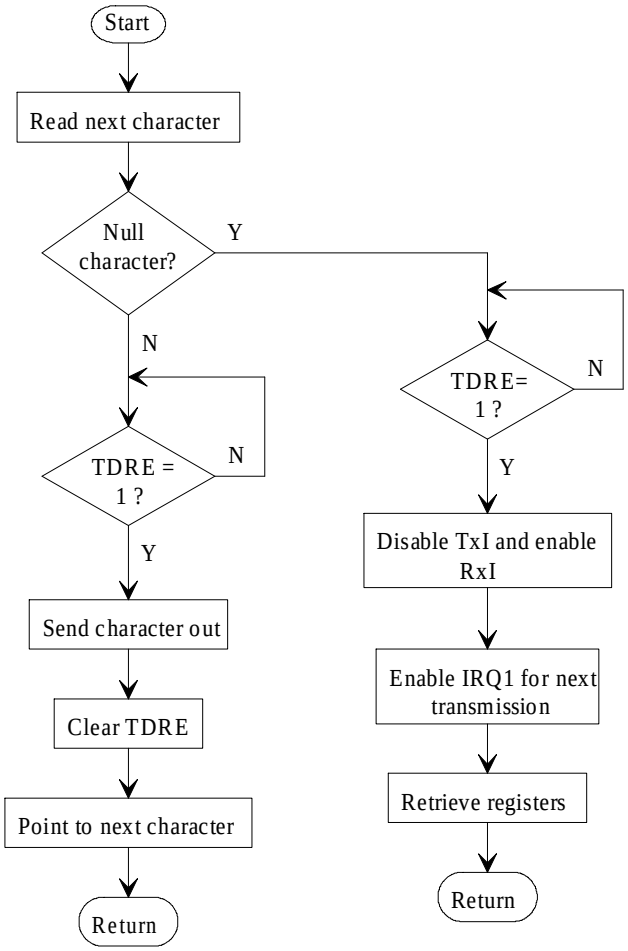

Figure 9. Transmit Interrupt Service Routine.

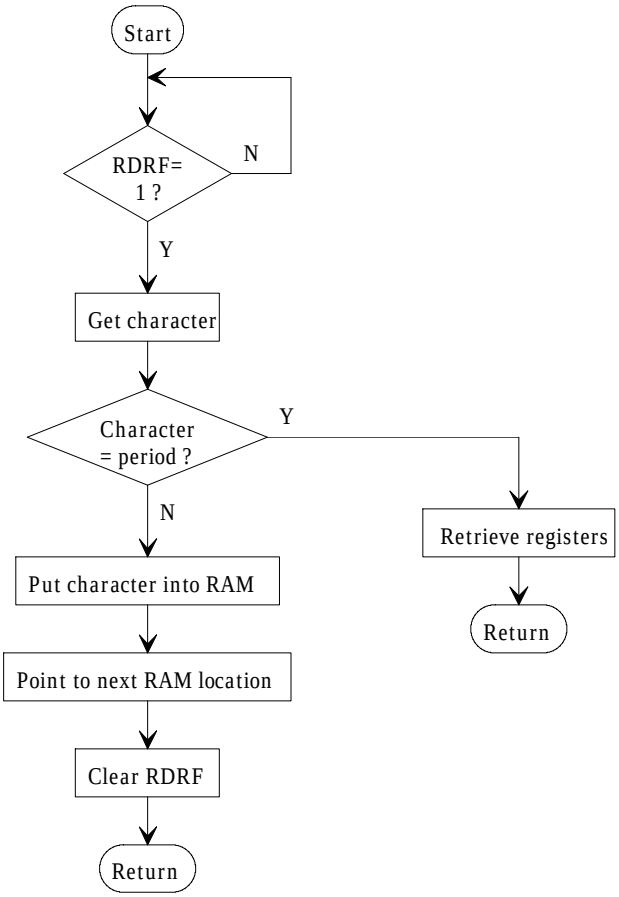

Figure 10. Receive Interrupt Service Routine.

HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

H8330.EQU LISTING

```
        .list off
;H8330 definitions
;I/O Port Addresses
P1_DDR  .EQU    H'FFB0
P2_DDR  .EQU    H'FFB1
P3_DDR  .EQU    H'FFB4
P4_DDR  .EQU    H'FFB5
P5_DDR  .EQU    H'FFB8
P6_DDR  .EQU    H'FFB9
;P7_DDR .EQU    Input only!!!
P8_DDR  .EQU    H'FFBD
P9_DDR  .EQU    H'FFC0

P1_DR   .EQU    H'FFB2
P2_DR   .EQU    H'FFB3
P3_DR   .EQU    H'FFB6
P4_DR   .EQU    H'FFB7
P5_DR   .EQU    H'FFBA
P6_DR   .EQU    H'FFBB
P7_DR   .EQU    H'FFBE
P8_DR   .EQU    H'FFBF
P9_DR   .EQU    H'FFC1

;Serial Communication Interface

SCI0_SMR        .EQU    H'FFD8
SCI0_BRR        .EQU    H'FFD9
SCI0_SCR        .EQU    H'FFDA
SCI0_TDR        .EQU    H'FFDB
SCI0_SSR        .EQU    H'FFDC
SCI0_RDR        .EQU    H'FFDD

;RAM Locations Definitions

TOP_RAM         .EQU    H'FF80
REC_DATA        .EQU    H'0D80
DATA            .EQU    H'0400

;Interrupt Controller Registers

ISCR    .EQU    H'FFC6
IER     .EQU    H'FFC7

        .list on
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

```
Microtec Research ASMH83    Version 1.0A    Jan 13 11:21:18 1993     Page 1


Command line:  C:\MRI\ASMH83\ASMH83.EXE -l apnote1.src
Line        Addr
1                                   ;       Include the H8330 register definitions file
2
3                                           .include "c:\mri\asmh83\H8300.equ"
4
5                                   ;       Specify interrupt vector addresses
6
7                                           .org    H'0A
8           000A 0544                       .data.w START_TRANS             ;use IRQ1 to start transmission routine
9                                           .org    H'38
10          0038 0596                       .data.w START_REC               ;use RxI (receive-end) to receive next character
11                                          .org    H'3A
12          003A 0566                       .data.w TRANS                   ;use TXI (transmit-end) to transmit next character
13
14                                  ;       Specify location of ASCII character data string
15
16                                          .org    H'400
17          0400 4252 4156                  .data.b H'42,H'52,H'41,H'56     ;BRAV
18          0404 4F20 2620                  .data.b H'4F,H'20,H'26,H'20     ;O_&_
19          0408 4645 4C49                  .data.b H'46,H'45,H'4C,H'49     ;FELI
20          040C 4349 5441                  .data.b H'43,H'49,H'54,H'41     ;CITA
21          0410 5449 4F4E                  .data.b H'54,H'49,H'4F,H'4E     ;TION
22          0414 5320 2020                  .data.b H'53,H'20,H'20,H'20     ;S___
23          0418 00                         .data.b H'00                    ;end-of-string (null character)
24
25                                  ;       Specify program start address
26
27                                          .org    H'500
28
29                                  ;       Perform MCU initializations and save registers
30
31          0500 7907 FF80                  mov.w   #TOP_RAM,R7             ;set stackpointer to H'FF80
32          0504 0700                       ldc     #0,CCR                  ;unmask all interrupts
33          0506 6DF0                       push    R0                      ;save registers
34          0508 6DF5                       push    R5
35          050A 6DF6                       push    R6
36          050C 7906 0400                  mov.w   #DATA,R6                ;point to ASCII string start address in RAM
37          0510 7905 0D80                  mov.w   #REC_DATA,R5            ;point to start RAM location where the receive data
                                                                            ;will be stored
38
39                                  ;       Initialize the SCI - set baud rate, communication format, and serial clock rate
40
41          0514 7FDA 7240                  bclr    #4,@SCI0_SCR            ;disable the receive function (clear RE bit)
42          0518 7FDA 7250                  bclr    #5,@SCI0_SCR            ;disable the transmit function (clear TE bit)
43          051C F81F                       mov.b   #31,R0L                 ;set the BRR value corresponding to an internal
44          051E 38D9                       mov.b   R0L,@SCI0_BRR           ;clock of 10MHz and 9600 baud
45          0520 F804                       mov.b   #H'04,R0L               ;set communication format and clock rate
46          0522 38D8                       mov.b   R0L,@SCI0_SMR
47
48                                  ;       Initialize IRQ1 - it will be used to start the transmission routine
49
50          0524 7FC0 7210                  bclr    #1,@P9_DDR              ;P9(1) used as IRQ1 input
51          0528 7FC1 7010                  bset    #1,@P9_DR
52          052C 7FC6 7010                  bset    #1,@ISCR                ;IRQ1 will be sensed on falling edge
53
54                                  ;       Enable transmit and receive
55
56          0530 7FDA 7050                  bset    #5,@SCI0_SCR            ;enable the transmit function (set TE bit)
57          0534 7FDA 7040                  bset    #4,@SCI0_SCR            ;enable the receive function (set RE bit)
58          0538 7FC7 7010                  bset    #1,@IER                 ;enable IRQ1
59
60                                  ;       Wait for start transmission interrupt to occur
61
62          053C 0180                   WAIT:   sleep
63          053E 0000                           nop
64          0540 0000                           nop
65          0542 40F8                           bra     WAIT
66
67                                  ;       Start transmission routine
68
69                                  START_TRANS:
70          0544 6868                           mov.b   @R6,R0L                 ;get first character ASCII code
71          0546 7EDC 7370              TEST1:  btst    #7,@SCI0_SSR            ;is the TDRE bit set?
72          054A 47FA                           beq     TEST1                   ;if not, test again
73          054C 38DB                           mov.b   R0L,@SCI0_TDR           ;send the first character data out
74          054E 7FDA 7260                      bclr    #6,@SCI0_SCR            ;disable receive-end interrupt
75          0552 7FDA 7070                      bset    #7,@SCI0_SCR            ;enable transmit-end interrupt
76          0556 7FDC 7270                      bclr    #7,@SCI0_SSR            ;clear the TDRE bit
77          055A 0B06                           adds    #1,R6                   ;point to second character
78          055C 7FC7 7210                      bclr    #1,@IER                 ;disable IRQ1
79          0560 5670                           rte
80          0562 0000                           nop
81          0564 0000                           nop
82
83                                  ;       Begin transmit interrupt service routine
84
85          0566 6868                   TRANS:  mov.b   @R6,R0L                 ;get next character ASCII code
86          0568 4710                           beq     END_TRANS               ;if null character, go to END_TRANS
87          056A 7EDC 7370              TEST2:  btst    #7,@SCI0_SSR            ;is the TDRE bit set?
88          056E 47FA                           beq     TEST2                   ;if not, test again
89          0570 38DB                           mov.b   R0L,@SCI0_TDR           ;send the next character out
90          0572 7FDC 7270                      bclr    #7,@SCI0_SSR            ;clear the TDRE bit
91          0576 0B06                           adds    #1,R6                   ;point to next character
92          0578 5670                           rte
93                                  END_TRANS:
94          057A 7EDC 7370                      btst    #7,@SCI0_SSR            ;is last character transmitted?
95          057E 47FA                           beq     END_TRANS               ;if not, wait until finished
96          0580 7FDA 7270                      bclr    #7,@SCI0_SCR            ;disable transmit-end interrupts
97          0584 7FC7 7010                      bset    #1,@IER                 ;enable IRQ1 for start transmission
98          0588 7FDA 7060                      bset    #6,@SCI0_SCR            ;enable receive-end interrupts
```


HITACHI
Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300

```
Microtec Research ASMH83    Version 1.0A    Jan 13 11:21:18 1993     Page 2


Line        Addr
99          058C 6D76                          pop     R6                  ;retrieve previous register contents
100         058E 6D70                          pop     R0
101         0590 5670                          rte
102         0592 0000                          nop
103         0594 0000                          nop
104
105                                    ;       Start receive routine
106
107                                    START_REC:
108         0596 7EDC 7360                     btst    #6,@SCI0_SSR        ;is the RDRF bit set?
109         059A 47FA                          beq     START_REC           ;if not, test again
110         059C 20DD                          mov.b   @SCI0_RDR,R0H       ;get character
111         059E A02E                          cmp.b   #H'2E,R0H           ;is character a period
112         05A0 470A                          beq     END_REC             ;if so, end receiving
113         05A2 68D0                          mov.b   R0H,@R5             ;put character ASCII code into RAM
114         05A4 0B05                          adds    #1,R5               ;point to next RAM address
115         05A6 7FDC 7260                     bclr    #6,@SCI0_SSR        ;clear the RDRF bit
116         05AA 5670                          rte
117                                    END_REC:
118         05AC 6D75                          pop     R5                  ;retrieve previous register contents
119         05AE 6D70                          pop     R0
120         05B0 5670                          rte
121                                            .end
```


HITACHI

Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300