Hitachi America, Ltd.

Application Note

H8/300

Cristian Tomescu

A H8/300 Family Software UART implementation

INTRODUCTION

The Universal Asynchronous Receiver-Transmitter (UART) is a standard peripheral feature on all of the H8/300 Family microcontrollers, with the exception of the H8/310 and H8/3101. This feature is implemented as part of the Serial Communication Interface (SCI) channel , which supports not only asynchronous communication but also synchronous communication. The UART on-chip implementation consists of a serial, 2-wire channel for transmitting and receiving data at speeds up to 365 Kbits/second (at a system clock of 10MHz), in half-duplex as well as in full-duplex, and driven from either the SCI baud-rate generator or from an external serial clock source. Although sufficient in most applications, one asynchronous channel may not be enough in applications that require intensive, multi-channel communications with various external devices such as modems or some other kind of Data Terminal devices via a standard RS-232 interface. Hence, the solution lies in implementing such an asynchronous communication channel entirely in software. A simple, one-channel receive-transmit link could be implemented utilizing 2 unused I/O lines and approximately 2K of assembly code, if the user can spare that much space either in the on-chip ROM area or in an external ROM.

The stadard UART protocol is graphically illustrated in Figure 1, and involves the following parameters:

1. **Start bit:** indicates that transmission or receiving has begun, and must always be low.
2. **Data bits:** form the actual serial information, and follow immediately the start bit. The number of data bits can either be 7 or 8, low or high.
3. **Parity bit:** this is an optional control bit. It determines if the total number of 1's in the data information is even or odd, thus creating either even or odd parity check. For instance, if the data frame had an even number of 1's already, a low parity bit would be sent to generate even parity; if odd parity is desired, a high parity bit should be present instead. If no parity check is desired, this bit is not communicated.
4. **Stop bit(s):** either one or two high bits after the data or the parity bit indicates the end of a data frame (or character). This insures that the frame is properly transmitted or received, and that the next frame must wait at least that long before a new character is transmitted (or received) again.
5. **Baud rate:** indicates the speed at which the serial bits are travelling through the channel. The time each bit must be present at the channel is equal to 1/baudrate.

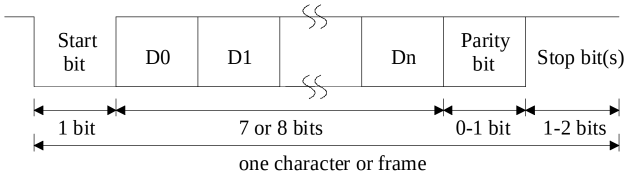

Figure 1.

OPERATION OVERVIEW

This application note shows how a UART with the same functionality as the one in the H8/300 SCI module can be implemented strictly using assembly code. The only difference from the H8/300 SCI module is that this implementation allows only for half-duplex communication, and not for simultaneous transmit and receive. The H8/330 microcontroller device was used since it is representative of the whole H8/300 family. 2 I/O lines were used to act as a serial transmitter and a serial receiver. Specifically, line 0 of I/O Port 9 is acting as the transmit channel, and line 2 of the same I/O Port is used to receive external serial data. The P9(0) line is initialized high, and transmission starts upon the first high-to-low transition at the line **after** the line has been high for at least 2 bit times. The time length of each bit (T) is directly determined from the baudrate (=1/baudrate), and can be implemented in software using either software delay loops or the 16-bit timer counter. Figure 2 shows graphically how the transmit operation is achieved. At the end of each bit time, P9(0) is set to its next level, and maintained there for another bit time period, and so on until the final stop bit time.

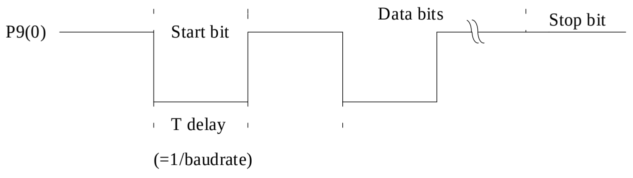

Figure 2.

The P9(2) line is configured as an input with its MOS pull-up high. This I/O pin is multiplexed with the IRQ0 activation line. Before any data is received, the IRQ0 interrupt is enabled for edge triggering. Upon a high-to-low transition at this line, the IRQ0 interrupt service routine starts to be executed, and the receive operation begins. The receive pin is sampled by the CPU at the middle of each bit time, and the value read is transferred into a designated receive data register. In order to sample the start bit level, the software has to wait for half the bit time (that is T/2=1/[2xbaudrate]) from the time IRQ0 kicks in. After the start bit level is detected, the software has to wait for T=1/baudrate to sample each subsequent frame bits. This process is illustrated in Figure 3.

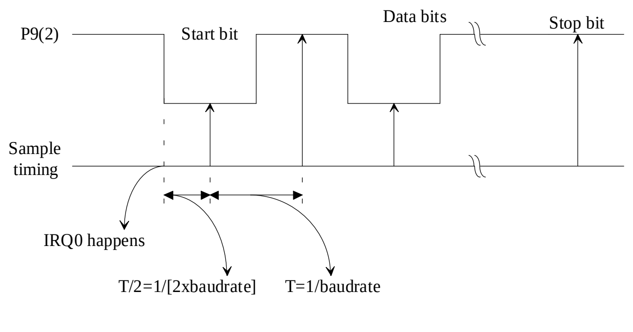

Figure 3.

This particular software UART implementation allows serial communication at 7 speeds: 300, 600, 1200, 2400, 4800, 9600, and 19,200 baud. The bit times corresponding to each baud rate and half baudrate sample points are given in the table below:

Baud rate	bit time	1/2 bit time
300	3330µs	1667µs
600	1667µs	833µs
1200	833µs	417µs
2400	417µs	208µs
4800	208µs	104µs
9600	104µs	52µs
19200	52µs	26µs

Another important factor in the design of a software-driven UART is the speed of the CPU clock in relation to the bit rate. This fact imposes some limitations in the design. Specifically, for a receive operation triggered by an external interrupt (IRQ0), it takes at least 20 states before the interrupt service routine starts executing. In addition, the bit times are obtained through software delays by idle instructions, and each instruction execution takes a certain amount of system clock cycles. At a system clock of 0.5MHz, it means at least 40us before the first instruction in the receive interrupt routine is executed. Therefore, the UART cannot run at 19,200 baud at this frequency since the start bit must be sampled 26us after IRQ0 kicks in. So, a minimum CPU frequency of 1MHz is necessary to run a software UART at 19,200 baud.

HARDWARE SETUP

The hardware platform used to run the software UART implementation consisted of the H8/330 evaluation board, the ASE development system and the H8/330 buffer box, an external RS-232 level converter board, a VT-320 "dummy" data terminal, and a keyboard. The H8/330 was emulated in the ASE system. The receive and transmit designated I/O lines are driven through a MX232 Dual Channel driver/receiver (on the RS-232 converter board) in order to provide the necessary voltage levels for proper RS-232 conversation. Then, the serial lines are linked via a 25-pin RS-232 female connector to the VT-320 "dummy" terminal. The keyboard is connected to the terminal as well, and is used to send out ASCII-coded characters to the H8/330. The VT-320 terminal will receive and display ASCII data from the H8/330. Figure 4 shows the schematic of the RS-232 converter board. For complete description of the H8/330 evaluation board features, refer to the H8/330 Evaluation Board User's manual.

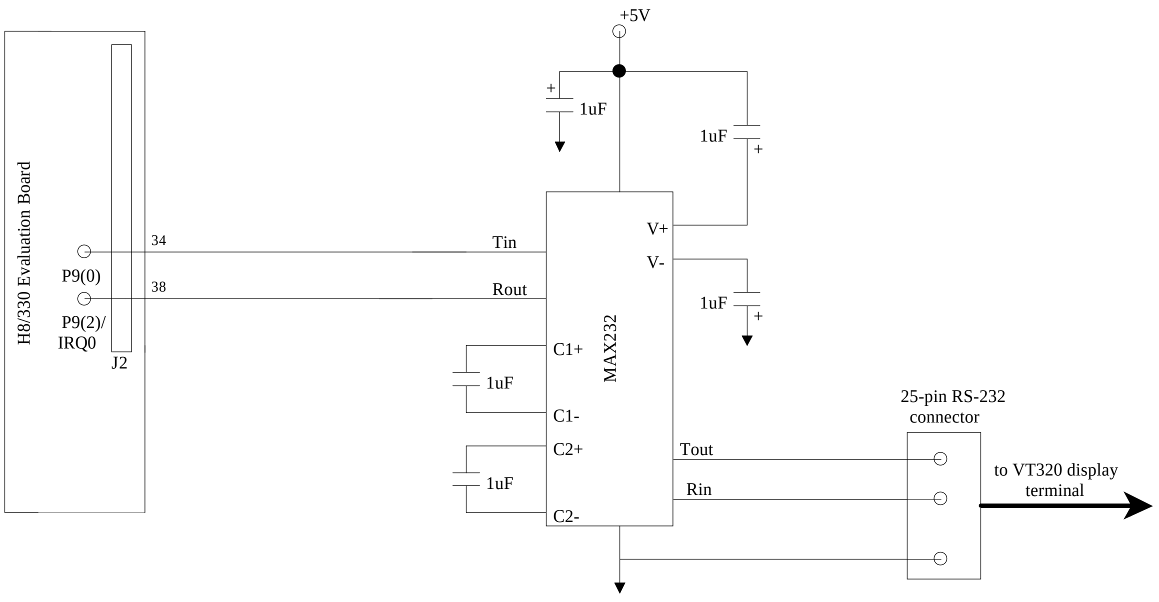

Figure 4.

SOFTWARE DESCRIPTION

This section will attempt to give a complete and detailed explanation of the software UART implementation. Note that this is only one example of such an implementation, and a multitude of approaches are possible. The program flowcharts are illustrated in figures 5 - 12, and are followed by the program listing.

This program allows for serial communication at the 7 different baudrates mentioned in the Operation Overview section. Also, it includes the parity/non-parity, even/odd parity, 7/8 data bits, and 1/2 stop bits options. The various baudrates as well as the half baudrate bit times for sampling the receive start bit are located in memory starting at locations H'0500 and H'0510 respectively. The number of data bits in the frame are placed at locations H'0520 and H'0521. The parity/non-parity option is flagged at locations H'0540 and H'0541. Likewise, the even/odd parity and 1/2 stop bits options are indicated at consecutive memory locations starting at H'0550 and H'0530 respectively. The program performs basically 3 functions:

1. Transmits an initial "greeting" message to the VT-320 terminal screen.
2. Continously runs in a wait loop waiting for an ASCII character to be pressed on the keyboard, whereupon it sends this character to the receive buffer memory located between H'FD80 - H'FF7F in the on-chip RAM.
3. Upon successful receiving of each character, the same character is promptly transmitted back and displayed on the VT-320 terminal screen.

The start message is transmitted to the terminal at 9600 baud, 8-bit data frame, no parity, and 1 stop bit. The user must make sure that the receiving device (in this case the VT-320 terminal) is properly setup for the above-mentioned serial protocol. The I/O port lines used for transmission and receiving are properly configured for output and input in the P9DDR register. The P9(2) input line has also its MOS pullup high. Transmission starts when the level at P9(0) is pulled low. This pin will stay low for the bit time corresponding to the baudrate. A loop comprised of 4 instructions lasting 10 clock states is used to provide the necessary bit time delay. In this example, the system clock runs at 10MHz, meaning that one clock state lasts 0.1us. Since the execution of the loop takes 10 states, meaning 1us, the bit time delay counter value is thus equal to the bit time value corresponding to the baudrate. This value is loaded into a general register and used as the loop counter. When the count expires, P9(0) is set to the level of the least significant bit (LSB) of the first data byte, and the loop process repeats. Upon the end of the loop count, the data is shifted one bit to the right, and the next bit is transmitted as explained above. When all data bits have been transmitted, the software transmits a high stop bit in the same way as the other bits. Then, the software repeats the process and transmits the next data byte until a null character is detected. Then, it enters a wait loop and waits for the IRQ0 interrupt to exit from it.

The second part of the program will receive ASCII characters entered on the keyboard and store them in the on-chip RAM area starting at location H'FD80. Before starting the receive routine, the user must make sure that the transmitting device (in this case the VT-320 terminal) is set up with the serial protocol conditions specified in the receive portion of the program. In the provided example, data is received in 8-bit format with even parity and 2 stop bits at 9600 bits/second. These conditions could be easily changed by changing the symbol in the appropriate line of code. To switch to a different baudrate, simply change the symbol at line 253 of the code. Each baudrate change should be accompanied by an appropriate change in the half baudrate count value at line 239. Likewise, the data format can be changed at line 238, parity/non-parity is selected at line 240, even/odd parity at lines 305 and 325, and 1/2 stop bits at line 279. The receive process starts when a high-to-low transition is detected at the P9(2)/IRQ0 input pin. This occurs as an ASCII key is pressed on the keyboard. The program then enters the IRQ0 interrupt service routine. Since the receive data input line is multiplexed with the IRQ0 line, the first instruction of the interrupt service routine must disable the IRQ0 interrupt. Then, the software loops for half a bit time count, and then samples the level at the pin. If it is high, the program stops (indicative of a receive error). If it is low, the software loops for the bit time corresponding to the chosen baudrate, samples the level at the pin, and writes the most significant bit of the receive data register (R4L) with that value. If the data bit is high, a previously zero-initialized register (R6H - High data bit copunter) is incremented. Also, the data bit register counter (R1H - containing the number of data bits flag) is decremented. Then, the contents of R4L are shifted one bit to the right, and the next data bit is sampled. When the last data bit has been sampled, the software checks for parity/non-parity by reading the flag in the parity or non-parity memory location. If parity is chosen, then the software performs another check to determine if even or odd parity is desired by reading the flag in the even or odd parity memory location. Then the parity bit is sampled in the same way as the data bits. If the expected level of the parity bit is not correct, the program stops. If the parity check is as expected, the received data byte is loaded into the memory buffer. Then, the software checks to determine if 1 stop bit or 2 stop bits are to be received by reading the flag in the 1_STOP or 2_STOP memory location. The stop bit(s) is then sampled in the same way as the other bits. Upon successful receiving of one data byte, the software proceeds on to transmit the just-received ASCII character back to the VT-320 terminal for display.

The re-transmit portion of the program is similar in approach to the initial message transmit routine. The re-transmit protocol is kept the same as the receiving protocol. Since the data bits counter has been decremented in the receive part of the program, the data format must be re-initialized at line 361. The start and data bits are transmitted as described previously in this section. When all the data bits have been sent out, the software performs a parity/non-parity check. If parity is chosen, it checks for even or odd parity, and proceeds on transmitting the appropriate parity level. Finally, the software checks for the number of stop bits that need to be transmitted, and proceeds on transmitting them. At the end of the ASCII character re-transmission, the IRQ0 interrupt is enabled again and the program returns from the interrupt service routine to the wait loop until a new ASCII character is entered on the keyboard and the whole process starts again.

FLOWCHARTS

Figures 5-11 show the program flowcharts. Figure 5 illustrates the main program flowchart. Figure 6 shows the Initial Message Transmit portion and figure 7 shows the interrupt service routine. Figure 8 illustrates the receiving portion of the interrupt routine, and figure 9 shows the receive parity check program segment. Finally, figure 10 shows the re-transmit portion of the interrupt service routine, and figure 11 shows the transmit parity check program segment.

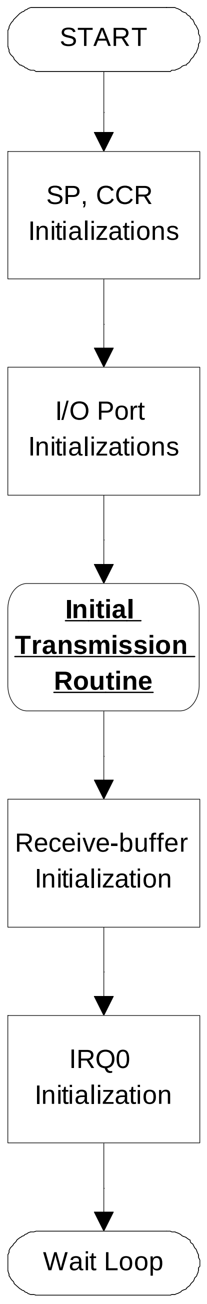

Figure 5. Main program.

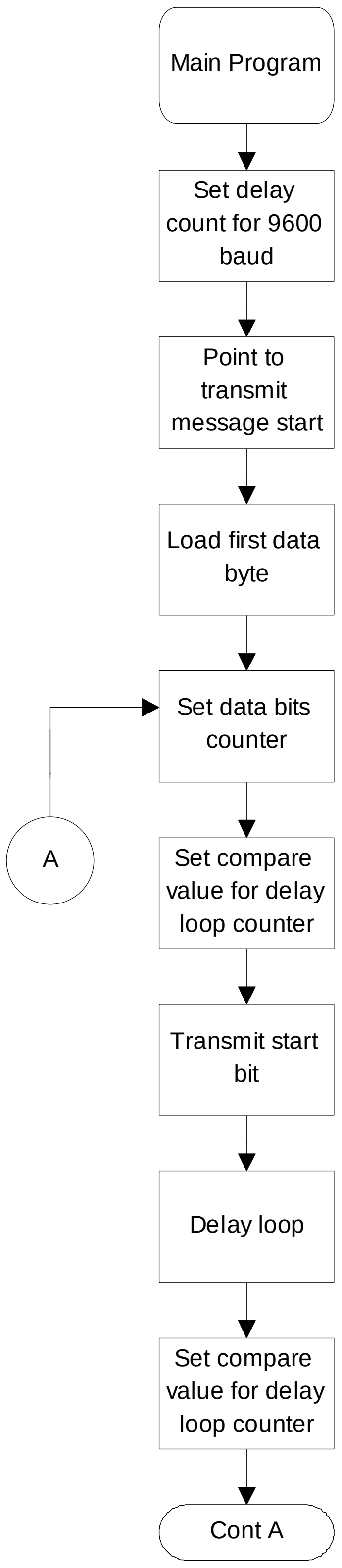
Main Program
Set delay count for 9600 baud
Point to transmit message start
Load first data byte
Set data bits counter
A
Set compare value for delay loop counter
Transmit start bit
Delay loop
Set compare value for delay loop counter
Cont A

Figure 6. Initial Transmission.

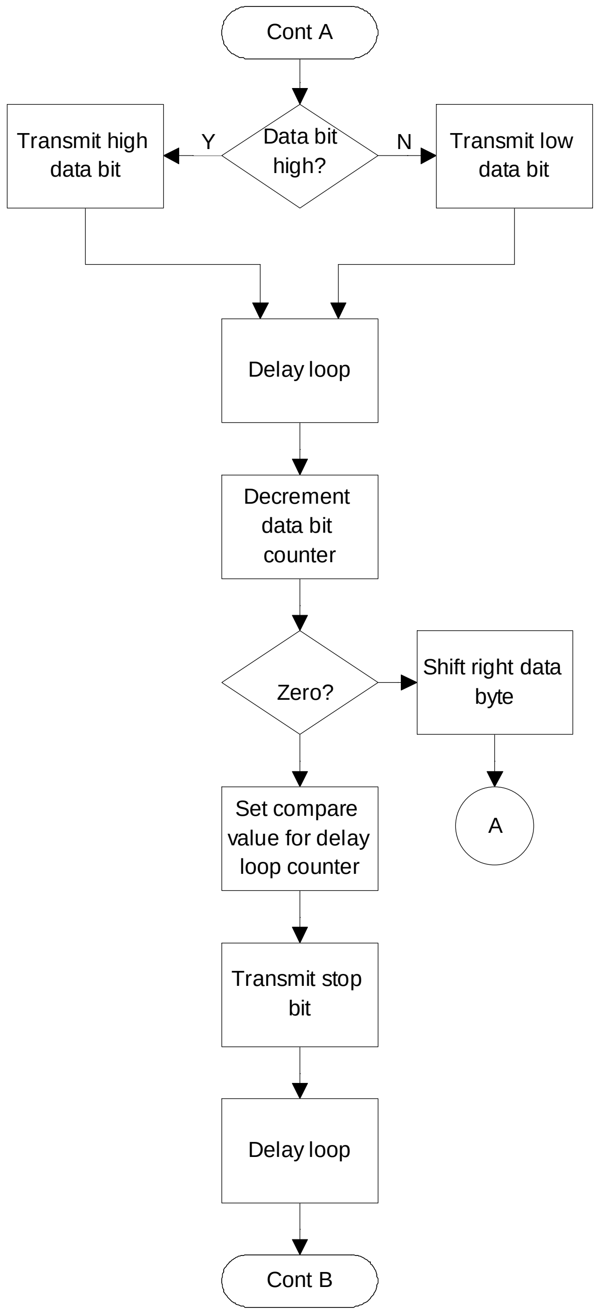

Figure 6. Continuation.

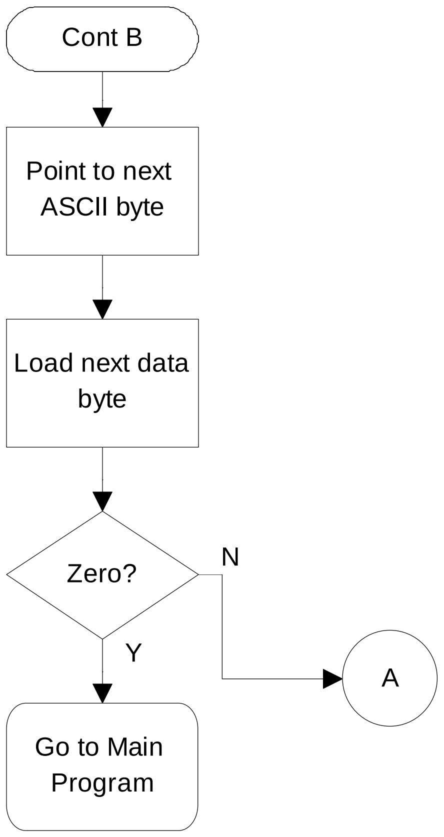

Figure 6. Continuation.

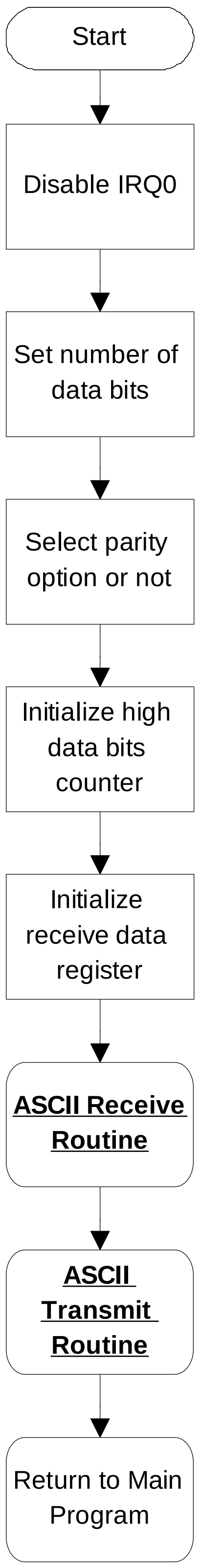

Figure 7. Interrupt Service Routine.

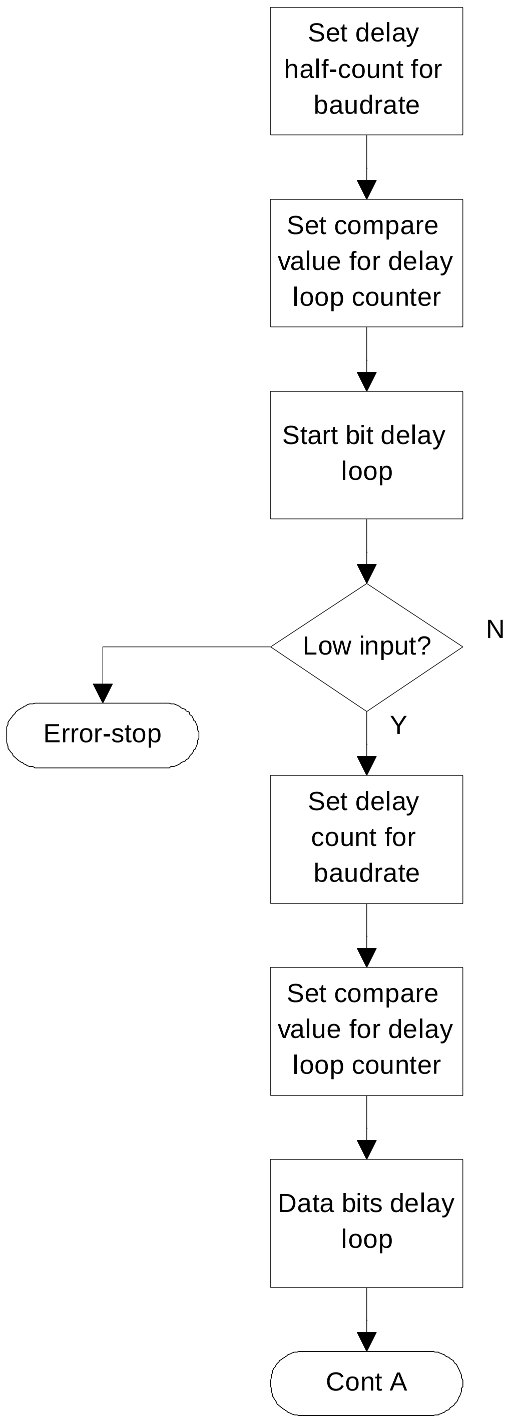
Set delay half-count for baudrate
Set compare value for delay loop counter
Start bit delay loop
Low input?
N
Y
Error-stop
Set delay count for baudrate
Set compare value for delay loop counter
Data bits delay loop
Cont A

Figure 8. Receive ASCII character.

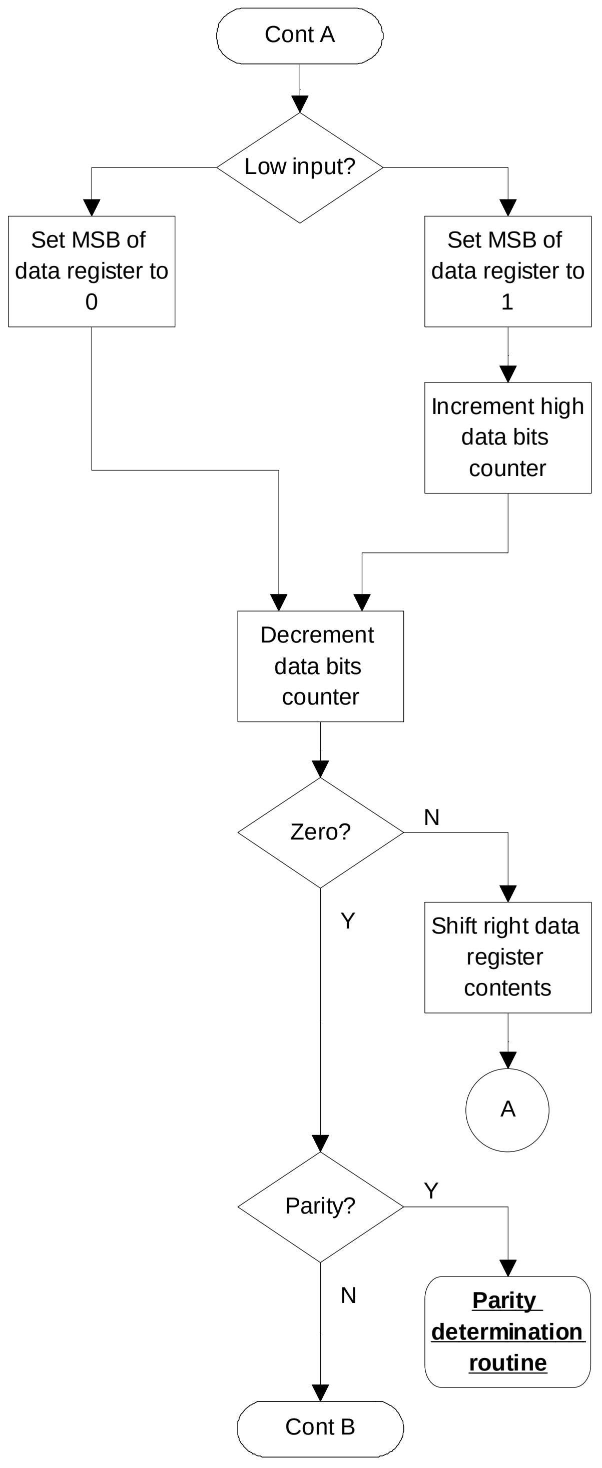
Cont A
Low input?
Set MSB of data register to 0
Set MSB of data register to 1
Increment high data bits counter
Decrement data bits counter
Zero?
N
Y
Shift right data register contents
A
Parity?
Y
N
Parity determination routine
Cont B

Figure 8. Continuation.

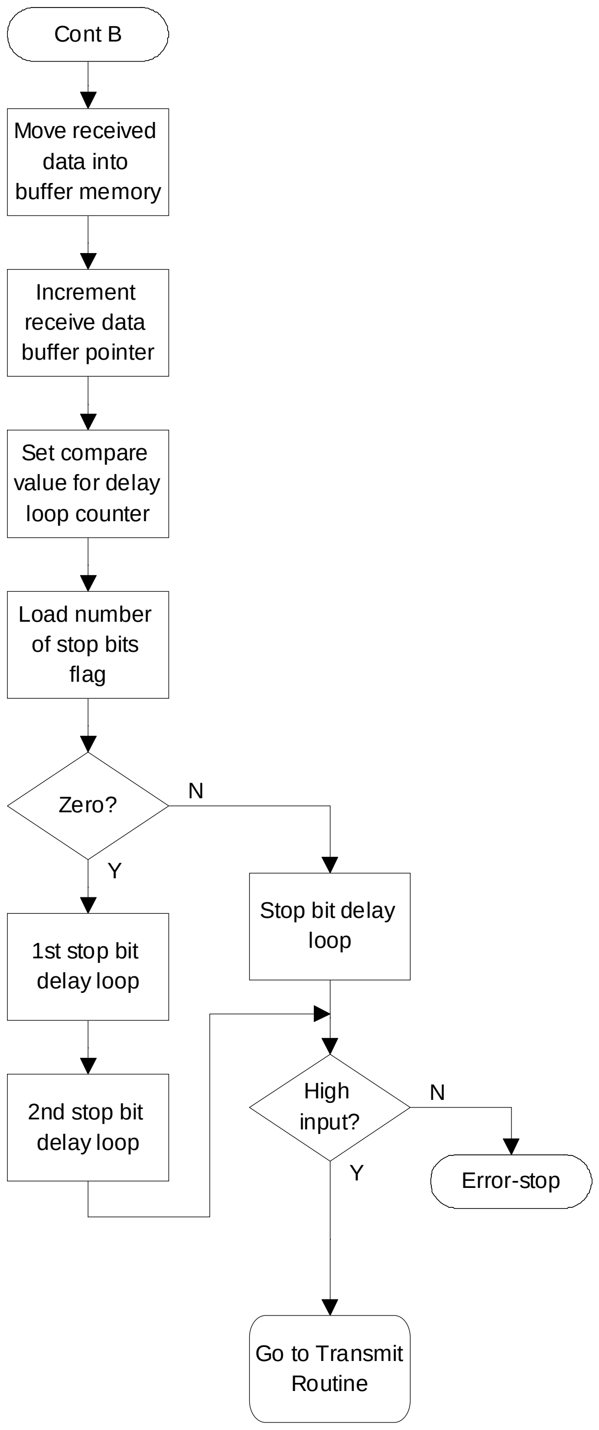
Cont B
Move received data into buffer memory
Increment receive data buffer pointer
Set compare value for delay loop counter
Load number of stop bits flag
Zero?
N
Y
1st stop bit delay loop
Stop bit delay loop
2nd stop bit delay loop
High input?
N
Y
Error-stop
Go to Transmit Routine

Figure 8. Continuation.

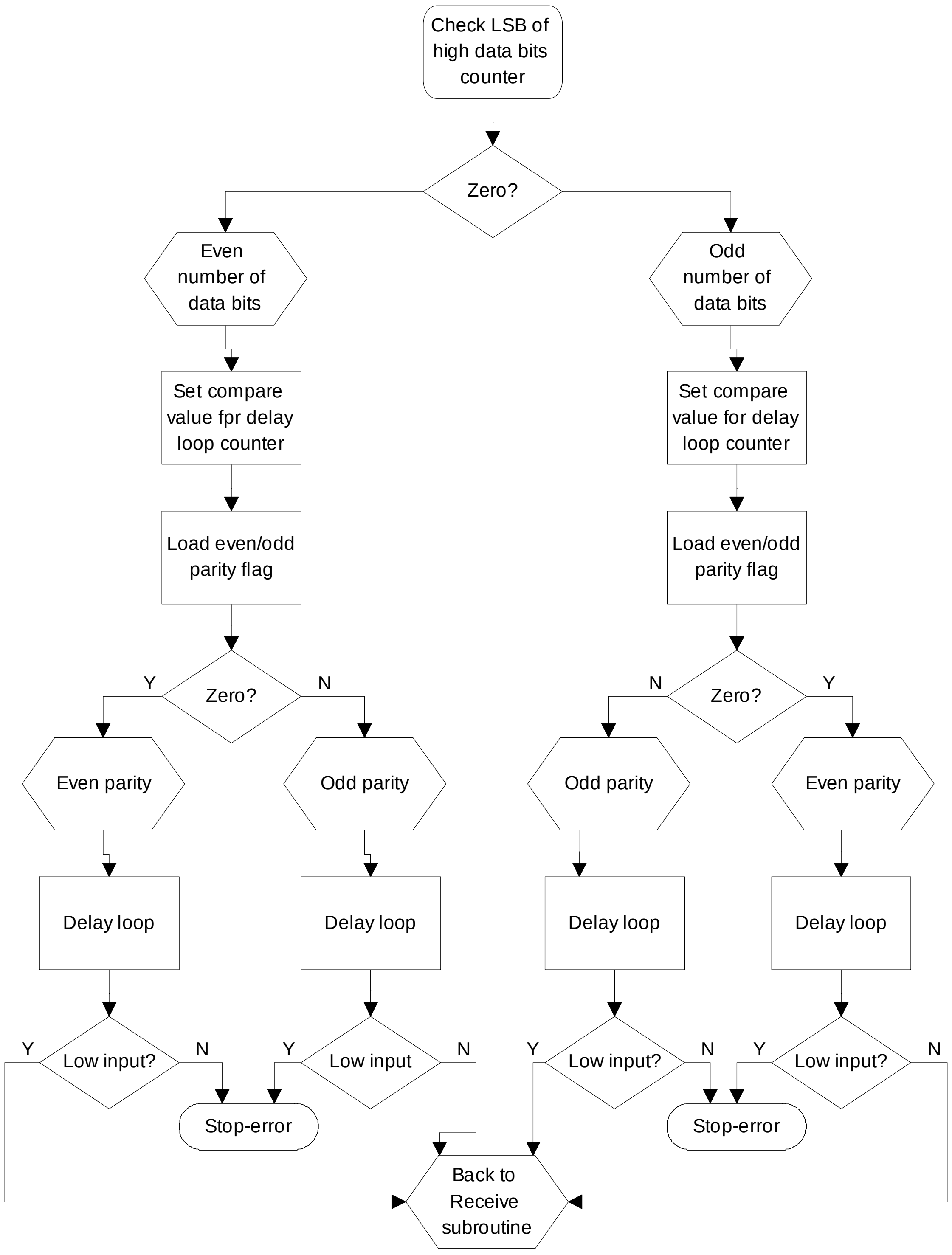

Figure 9. Receive Parity Check.

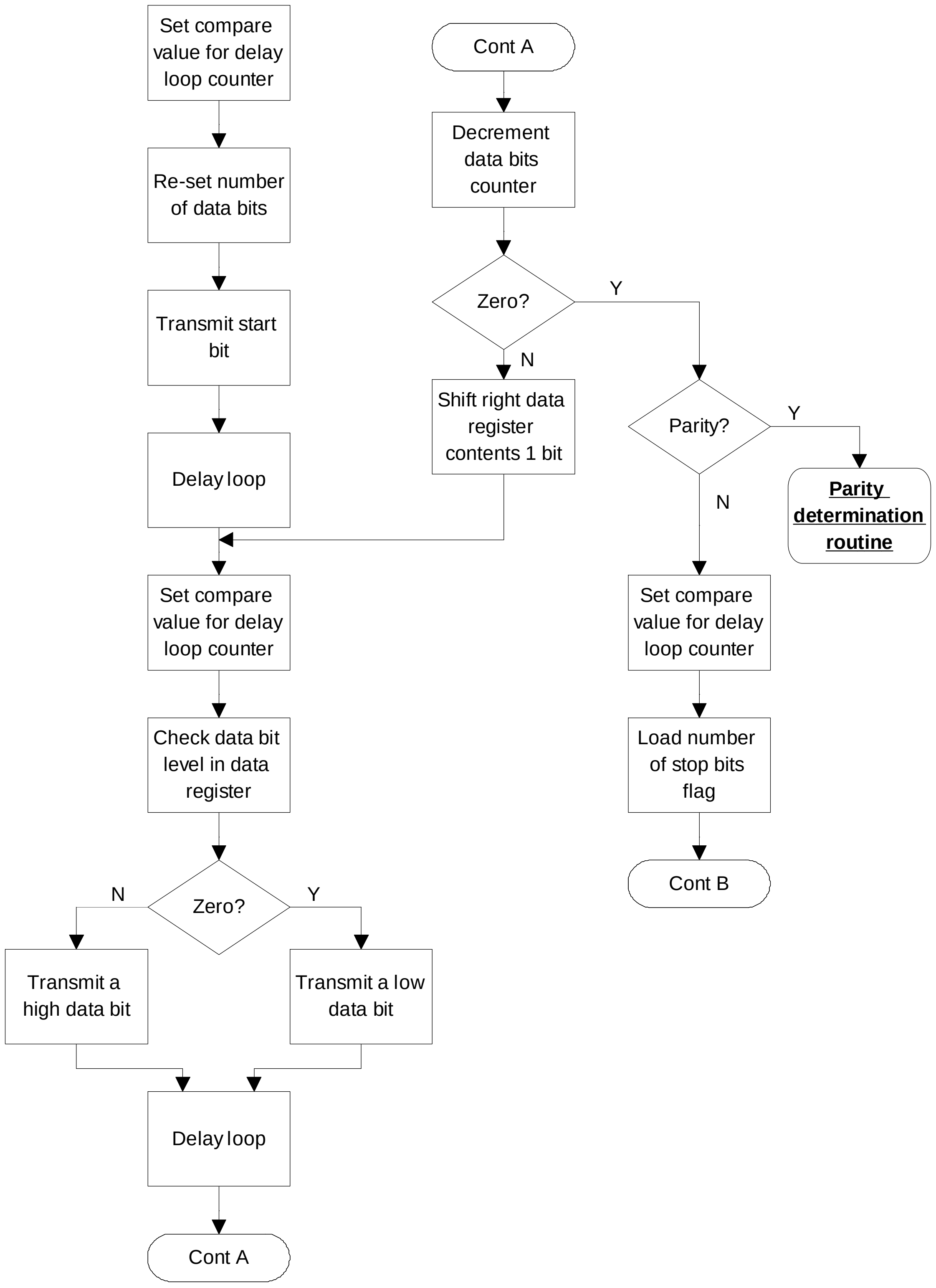
Set compare value for delay loop counter
Re-set number of data bits
Transmit start bit
Delay loop
Set compare value for delay loop counter
Check data bit level in data register
Zero?
N
Y
Transmit a high data bit
Transmit a low data bit
Delay loop
Cont A
Cont A
Decrement data bits counter
Zero?
Y
N
Shift right data register contents 1 bit
Parity?
Y
N
Parity determination routine
Set compare value for delay loop counter
Load number of stop bits flag
Cont B

Figure 10. Re-transmit ASCII character.

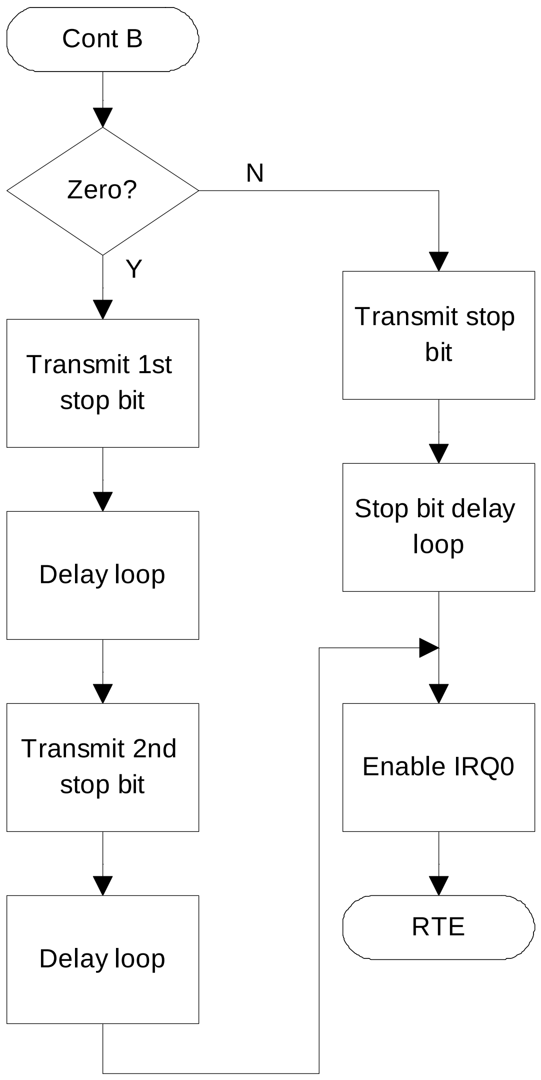

Figure 10. Continuation.

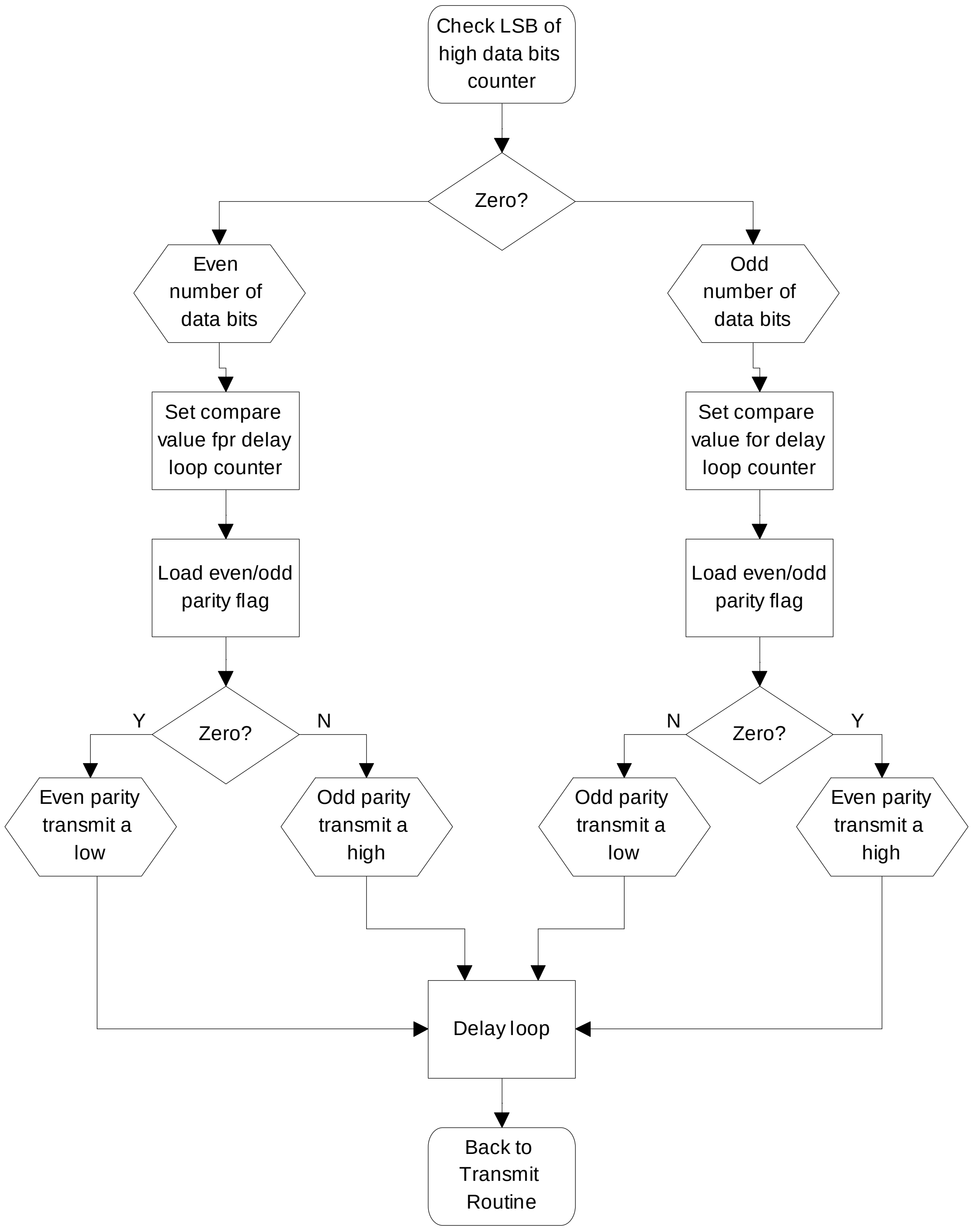
Check LSB of high data bits counter
Zero?
Even number of data bits
Odd number of data bits
Set compare value fpr delay loop counter
Set compare value for delay loop counter
Load even/odd parity flag
Load even/odd parity flag
Y
Zero?
N
N
Zero?
Y
Even parity transmit a low
Odd parity transmit a high
Odd parity transmit a low
Even parity transmit a high
Delay loop
Back to Transmit Routine

Figure 11. Re-transmit Parity Check.

```
Command line:  C:\MRI\ASMH83\ASMH83.EXE -l uart1.src
Line        Addr
1                                  ;**************A SOFTWARE UART IMPLEMENTATION*************
2
3                                  ; H8/330 register definitions
4
5           FFC0                   P9_DDR          .equ    H'FFC0
6           FFC1                   P9_DR           .equ    H'FFC1
7           FFC6                   ISCR            .equ    H'FFC6
8           FFC7                   IER             .equ    H'FFC7
9
10                                 ; Baud rate addresses
11
12          0500                   B_300           .equ    H'0500
13          0502                   B_600           .equ    H'0502
14          0504                   B_1200          .equ    H'0504
15          0506                   B_2400          .equ    H'0506
16          0507                   B_4800          .equ    H'0507
17          0508                   B_9600          .equ    H'0508
18          0509                   B_19200         .equ    H'0509
19
20                                 ; Half baudrate count addresses
21
22          0510                   HB_300          .equ    H'0510
23          0512                   HB_600          .equ    H'0512
24          0514                   HB_1200         .equ    H'0514
25          0516                   HB_2400         .equ    H'0516
26          0517                   HB_4800         .equ    H'0517
27          0518                   HB_9600         .equ    H'0518
28          0519                   HB_19200        .equ    H'0519
29
30                                 ; Data size indicator addresses
31
32          0520                   DATA_8          .equ    H'0520
33          0521                   DATA_7          .equ    H'0521
34
35                                 ; Stop bit(s) indicator address
36
37          0530                   STOP_1          .equ    H'0530
38          0531                   STOP_2          .equ    H'0531
39
40                                 ; Parity/No parity indicator addresses
41
42          0540                   NO_PAR          .equ    H'540
43          0541                   PAR             .equ    H'541
44
45                                 ; Even/Odd parity indicator addresses
46
47          0550                   EVEN_PAR        .equ    H'550
48          0551                   ODD_PAR         .equ    H'551
49
50                                 ; Transmit and receive start memory locations
51
52          0100                   TR_DATA         .equ    H'0100
53          FD80                   REC_DATA        .equ    H'FD80
54
55                                 ; Specify reset address
56
57                                         .org    H'0
58          0000 0600                      .data.w MAIN
59
60                                 ; Specify IRQ0 interrupt vector address
61
62                                         .org    H'08
63          0008 0690                      .data.w RECEIVE
64
65                                 ; Initial transmission data space allocated in ROM between H'100
66                                 ; and H'4FF.  This example transmits "ASYNCHRONOUS TRANSMISSION"
67                                 ; in ASCII coding to a VT-320 data terminal.
68
69                                         .org    H'100
70          0100 2020 2020 2020 2020       .data.b H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20
                 2020
71          010A 2020 2020 2020 2020       .data.b H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20
                 2020
72          0114 20                        .data.b H'20
73          0115 2A2A 2A2A 2A20 4153       .data.b H'2A,H'2A,H'2A,H'2A,H'2A,H'20,H'41,H'53,H'59,H'4E
                 594E
74          011F 4348 524F 4E4F 5553       .data.b H'43,H'48,H'52,H'4F,H'4E,H'4F,H'55,H'53,H'20,H'54
                 2054
75          0129 5241 4E53 4D49 5353       .data.b H'52,H'41,H'4E,H'53,H'4D,H'49,H'53,H'53,H'49,H'4F
                 494F
76          0133 4E2A 2A2A 2A2A            .data.b H'4E,H'2A,H'2A,H'2A,H'2A,H'2A
```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 2

Addr
0139 2020 2020 2020 2020            .data.b H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20
     2020
0143 2020 2020 2020 2020            .data.b H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20,H'20
     2020
014D 2020 2000                      .data.b H'20,H'20,H'20,H'00


                            ; Baud rates counter table starts at H'500

                                    .org    H'500
0500 0CB2                           .data.w H'CB2   ;count corresponding to 300 baud
0502 0659                           .data.w H'659   ;count corresponding to 600 baud
0504 0341                           .data.w H'341   ;count corresponding to 1200 baud
0506 01A1                           .data.w H'1A1   ;count corresponding to 2400 baud
0508 D0                             .data.b H'D0    ;count corresponding to 4800 baud
0509 68                             .data.b H'68    ;count corresponding to 9600 baud
050A 34                             .data.b H'34    ;count corresponding to 19200 baud

                            ; Baud rates half-count table starts at H'510

                                    .org    H'510
0510 0659                           .data.w H'659   ;half-count corresponding to 300 baud
0512 0341                           .data.w H'341   ;half-count corresponding to 600 baud
0514 01A1                           .data.w H'1A1   ;half-count corresponding to 1200 baud
0516 D0                             .data.b H'D0    ;half-count corresponding to 2400 baud
0517 68                             .data.b H'68    ;half-count corresponding to 4800 baud
0518 34                             .data.b H'34    ;half-count corresponding to 9600 baud
0519 1A                             .data.b H'1A    ;half-count corresponding to 19200 baud

                            ; Data size indicators

                                    .org    H'520
0520 08                             .data.b H'08            ;8 data bits
0521 07                             .data.b H'07            ;7 data bits

                            ; Stop bits indicators

                                    .org    H'530
0530 01                             .data.b H'01            ;1 stop bit indicator
0531 00                             .data.b H'0             ;2 stop bits indicator

                            ; Parity indicators

                                    .org    H'540
0540 00                             .data.b H'0             ;no parity
0541 01                             .data.b H'01            ;parity

                            ; Even/Odd parity indicators

                                    .org    H'550
0550 00                             .data.b H'0             ;even parity indicator
0551 01                             .data.b H'01            ;odd parity indicator

                            ; Main program starts at H'600

                                    .org    H'600
0600 7907 FF80              MAIN:   mov.w   #H'FF80,R7      ;initialize stackpointer
0604 0700                           ldc     #0,CCR          ;un-mask all interrupts

                            ; Initialize the I/O port lines for transmit and receive

0606 F801                           mov.b   #01,R0L         ;P9(0) is transmit output
0608 38C0                           mov.b   R0L,@P9_DDR     ;and P9(2) is receive input
060A F805                           mov.b   #H'05,R0L       ;turn MOS pullup for P9(2)
060C 38C1                           mov.b   R0L,@P9_DR      ;high and set P9(0) high

                            ; The initial message "ASYNCHRONOUS TRANSMISSION" is transmitted
                            ; to the VT-320 terminal.  Transmission protocol is:  8 data bits,
                            ; no parity, 1 stop bit, 9600 baud

                            ; Register usage:  R2L - contains baud rate (delay loop counter)
                            ;                  R3L - contains compare value for delay loop
                            ;                  R1L - contains ASCII data byte
                            ;                  R0L - general usage register
                            ;                  R6  - transmit data buffer address pointer
                            ;                  R1H - data bit counter

060E 6A0A 0508                      mov.b   @B_9600,R2L     ;set delay counter for 9600 bits/sec.
0612 7906 0100                      mov.w   #TR_DATA,R6     ;point to start address of data string
0616 6869                           mov.b   @R6,R1L         ;grab data byte
```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 3


Addr
0618 FB01                   START:  mov.b   #1,R3L          ;set compare value for delay counter loop
061A F108                           mov.b   #8,R1H          ;set data bit counter
061C F804                           mov.b   #H'04,R0L       ;start transmission
061E 38C1                           mov.b   R0L,@P9_DR
0620 0A0B                   LOOP1:  inc     R3L             ;start bit delay loop
0622 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
0624 0000                           nop
0626 46F8                           bne     LOOP1           ;if not, loop again
0628 41F6                           brn     LOOP1           ;delay
062A FB02                   DATA:   mov.b   #2,R3L          ;update compare value for delay counter loop
062C 7309                           btst.b  #0,R1L          ;check for data bit level
062E 4718                           beq     LOW
0630 F805                           mov.b   #H'05,R0L       ;data bit is high
0632 38C1                           mov.b   R0L,@P9_DR
0634 0000                           nop
0636 0A0B                   LOOP2:  inc     R3L             ;high data bits delay loop
0638 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
063A 0000                           nop
063C 46F8                           bne     LOOP2           ;if not, loop again
063E 0000                           nop                     ;delay
0640 1A01                           dec     R1H             ;if yes, update the data bit counter
0642 471C                           beq     STOP            ;last data bit?
0644 1109                           shlr    R1L             ;if not, get next data bit
0646 40E2                           bra     DATA
0648 F804                   LOW:    mov.b   #H'04,R0L       ;data bit is low
064A 38C1                           mov.b   R0L,@P9_DR
064C 0000                           nop                     ;delay
064E 0A0B                   LOOP3:  inc     R3L             ;low data bits delay loop
0650 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
0652 0000                           nop
0654 46F8                           bne     LOOP3           ;if not, loop again
0656 0000                           nop                     ;delay
0658 1A01                           dec     R1H             ;if yes, update the data bit counter
065A 4704                           beq     STOP            ;last data bit?
065C 1109                           shlr    R1L             ;if not, get next data bit
065E 40CA                           bra     DATA
0660 FB00                   STOP:   mov.b   #0,R3L          ;update compare value for delay counter loop
0662 41FC                           brn     STOP            ;delay
0664 41FA                           brn     STOP
0666 41F8                           brn     STOP
0668 7FC1 7000                      bset.b  #0,@P9_DR       ;set stop bit level
066C 0A0B                   LOOP4:  inc     R3L             ;stop bit delay loop
066E 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
0670 0000                           nop
0672 46F8                           bne     LOOP4           ;if not, loop again
0674 0B06                           adds    #1,R6           ;if yes, point to next ASCII byte
0676 6869                           mov.b   @R6,R1L         ;grab next byte
0678 4702                           beq     STRT_REC        ;if null character, stop initial transmission
067A 409C                           bra     START           ;transmit new data byte

                            ; ASCII byte data entered by pressing keys on a keyboard are received
                            ; and stored in the eval board's off-chip RAM starting at H'FD80

                            ; Register usage:  R5  - contains the memory buffer address pointer
                            ;                  R2L - contains delay loop counters
                            ;                  R3L - contains compare value for delay loop
                            ;                  R4L - contains received ASCII data byte
                            ;                  R4H - contains stop bit(s) indicator
                            ;                  R6L - contains parity indicator
                            ;                  R6H - contains the number of high data bits
                            ;                  R1H - contains the number of data bits
                            ;                  R0H - contains the even/odd parity indicator

                            ; Enable IRQ0 receive interrupt

                            STRT_REC:
067C 7FC6 7000                      bset.b  #0,@ISCR        ;IRQ0 sensed on the falling edge
0680 7905 FD80                      mov.w   #REC_DATA,R5    ;point to start address of receive buffer
                            REC_AGAIN:
0684 7FC7 7000                      bset.b  #0,@IER         ;enable IRQ0

                            ; Wait for receive interrupt

0688 0000                   WAIT:   nop
068A 40FC                           bra     WAIT
068C 0000                           nop
068E 0000                           nop

```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 4

Addr                                        ; Start of receive process

                            RECEIVE:
0690 7FC7 7200                      bclr.b  #0,@IER         ;disable IRQ0 - use I/O pin as input pin
0694 FB04                           mov.b   #4,R3L          ;set compare value for delay counter loop
0696 6A01 0520                      mov.b   @DATA_8,R1H     ;set data bit counter for 8-bit data
069A 6A0A 0518                      mov.b   @HB_9600,R2L    ;set delay half-count for 9600 bits/sec
069E 6A0E 0541                      mov.b   @PAR,R6L        ;select parity
06A2 F600                           mov.b   #0,R6H          ;initialize high data bits counter for parity check
06A4 0A0B                   LOOP5:  inc     R3L             ;start bit half-count delay loop
06A6 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
06A8 0000                           nop
06AA 46F8                           bne     LOOP5           ;if not, loop again
06AC 41F6                           brn     LOOP5           ;delays
06AE 41F4                           brn     LOOP5
06B0 FC00                           mov.b   #0,R4L          ;clear receive data register
06B2 7EC1 7320                      btst.b  #2,@P9_DR       ;check input pin level
06B6 4702                           beq     DATA_REC        ;if low, proceed on to receive data
06B8 0180                           sleep                   ;start bit cannot be high - stop
                            DATA_REC:
06BA 6A0A 0508                      mov.b   @B_9600,R2L     ;set delay counter for 9600 bits/sec
06BE FB02                   AGAIN:  mov.b   #2,R3L          ;set compare value for delay counter loop
06C0 41FC                           brn     AGAIN           ;delay
06C2 0A0B                   LOOP6:  inc     R3L             ;data bits delay loop
06C4 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
06C6 0000                           nop
06C8 46F8                           bne     LOOP6           ;if not, loop again
06CA 7EC1 7320                      btst.b  #2,@P9_DR       ;check level at receive pin
06CE 470C                           beq     ZERO
06D0 707C                           bset.b  #7,R4L          ;data bit is high
06D2 0A06                           inc     R6H             ;increment high data bit counter for parity check
06D4 1A01                           dec     R1H             ;decrement data bit counter
06D6 470E                           beq     PAR_CHK1        ;if MSB, determine if parity check
06D8 138C                           rotr    R4L             ;shift down data bit
06DA 40E2                           bra     AGAIN           ;delay for next data bit
06DC 727C                   ZERO:   bclr.b  #7,R4L          ;data bit is low
06DE 1A01                           dec     R1H             ;decrement data bit counter
06E0 4704                           beq     PAR_CHK1        ;if MSB, determine if parity check
06E2 138C                           rotr    R4L             ;shift down data bit
06E4 40D8                           bra     AGAIN           ;delay for next data bit
                            PAR_CHK1:
06E6 AE00                           cmp.b   #0,R6L          ;parity or not?
06E8 4630                           bne     PARITY1         ;if non-zero, receive parity bit too
06EA 68DC                   CONT:   mov.b   R4L,@R5         ;no parity, move data into receive buffer
06EC 0B05                           adds    #1,R5           ;point to next buffer location
06EE FB02                           mov.b   #2,R3L          ;set compare value for stop bit loop
06F0 6A04 0530                      mov.b   @STOP_1,R4H     ;determine number of stop bits
06F4 4710                           beq     TWO_STOP        ;if zero flag, 2 stop bits
                            ONE_STOP:
06F6 0A0B                           inc     R3L             ;stop bit delay loop
06F8 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
06FA 0000                           nop
06FC 46F8                           bne     ONE_STOP        ;if not, loop again
06FE 7EC1 7320              TEST:   btst.b  #2,@P9_DR       ;verify stop bit level
0702 466C                           bne     TRANSMIT        ;if high, transmit back the ASCII data
0704 0180                   ERROR:  sleep                   ;if low, receive failure
                            TWO_STOP:
0706 0A0B                           inc     R3L             ;1st stop bit delay loop
0708 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
070A 0000                           nop
070C 46F8                           bne     TWO_STOP        ;if not, loop again
070E FB01                           mov.b   #1,R3L          ;update compare value for 2nd stop bit delay
0710 0A0B                   LOOP7:  inc     R3L             ;2nd stop bit delay loop
0712 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
0714 0000                           nop
0716 46F8                           bne     LOOP7           ;if not, loop again
0718 40E4                           bra     TEST            ;continue
                            PARITY1:
071A 7306                           btst.b  #0,R6H          ;determine number of high data bits
071C 4728                           beq     EVEN_DAT        ;if even number, go to EVEN_DAT
                            ODD_DAT:
071E FB04                           mov.b   #4,R3L          ;set compare value for delay
0720 6A00 0550                      mov.b   @EVEN_PAR,R0H   ;select even (or odd) parity
0724 4710                           beq     EV_PAR1         ;if zero, even parity
                            OD_PAR1:
0726 0A0B                           inc     R3L             ;parity bit delay loop
0728 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
072A 0000                           nop
072C 46F8                           bne     OD_PAR1         ;if not, loop again
072E 7EC1 7320                      btst.b  #2,@P9_DR       ;check receive pin level
```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 5


Addr
0732 463A                   bne     PAR_ERR         ;if high, parity error
0734 40B4                   bra     CONT            ;if low, continue
                    EV_PAR1:
0736 0A0B                   inc     R3L             ;parity bit delay loop
0738 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
073A 0000                   nop
073C 46F8                   bne     EV_PAR1         ;if not, loop again
073E 7EC1 7320              btst.b  #2,@P9_DR       ;check receive pin level
0742 472A                   beq     PAR_ERR         ;if low, parity error
0744 40A4                   bra     CONT            ;if high, continue
                    EVEN_DAT:
0746 FB04                   mov.b   #4,R3L          ;set compare value for delay
0748 6A00 0550              mov.b   @EVEN_PAR,R0H   ;select even (or odd) parity
074C 4710                   beq     EV_PAR2         ;if zero, even parity
                    OD_PAR2:
074E 0A0B                   inc     R3L             ;parity bit delay loop
0750 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
0752 0000                   nop
0754 46F8                   bne     OD_PAR2         ;if not, loop again
0756 7EC1 7320              btst.b  #2,@P9_DR       ;check receive pin level
075A 4712                   beq     PAR_ERR         ;if low, parity error
075C 408C           CONT1:  bra     CONT            ;if high, continue
                    EV_PAR2:
075E 0A0B                   inc     R3L             ;parity bit delay loop
0760 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
0762 0000                   nop
0764 46F8                   bne     EV_PAR2         ;if not, loop again
0766 7EC1 7320              btst.b  #2,@P9_DR       ;check receive pin level
076A 4602                   bne     PAR_ERR         ;if high, parity error
076C 40EE                   bra     CONT1           ;if low, continue
                    PAR_ERR:
076E 0180                   sleep                   ;stop, parity error

                    ; Each received ASCII character is transmitted back and displayed
                    ; on the VT-320 data terminal

                    ; Register usage:  R2L - contains baud rate delay loop counter
                    ;                  R3L - contains compare value for delay loop
                    ;                  R4L - contains the ASCII data byte
                    ;                  R4H - contains stop bit(s) indicator
                    ;                  R6L - contains parity indicator
                    ;                  R6H - contains the number of high data bits
                    ;                  R1H - contains the number of data bits
                    ;                  R0H - contains the even/odd parity indicator
                    ;                  R0L - general purpose register

                    TRANSMIT:
0770 FB01                   mov.b   #1,R3L          ;set compare value for delay
0772 6A01 0520              mov.b   @DATA_8,R1H     ;set data bit counter for 8-bit
0776 F802                   mov.b   #H'02,R0L       ;start transmission
0778 38C1                   mov.b   R0L,@P9_DR
077A 0A0B           LOOP8:  inc     R3L             ;start bit delay loop
077C 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
077E 0000                   nop
0780 46F8                   bne     LOOP8           ;if not, loop again
0782 41F6                   brn     LOOP8           ;delay
0784 0000                   nop
                    DATA_NXT:
0786 FB02                   mov.b   #2,R3L          ;update compare value for delay counter loop
0788 730C                   btst.b  #0,R4L          ;check for data bit level
078A 4718                   beq     DAT_LOW
078C F803                   mov.b   #H'03,R0L       ;data bit is high
078E 38C1                   mov.b   R0L,@P9_DR
0790 0000                   nop
0792 0A0B           LOOP9:  inc     R3L             ;high data bits delay loop
0794 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
0796 0000                   nop
0798 46F8                   bne     LOOP9           ;if not, loop again
079A 0000                   nop                     ;delay
079C 1A01                   dec     R1H             ;if yes, update the data bit counter
079E 471C                   beq     PAR_CHK2        ;if last data bit, determine if parity check
07A0 110C                   shlr    R4L             ;if not, get next data bit
07A2 40E2                   bra     DATA_NXT
                    DAT_LOW:
07A4 F804                   mov.b   #H'04,R0L       ;data bit is low
07A6 38C1                   mov.b   R0L,@P9_DR
07A8 0000                   nop                     ;delay
07AA 0A0B           LOOP10: inc     R3L             ;low data bits delay loop
07AC 1CBA                   cmp.b   R3L,R2L         ;is delay count complete?
```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 5


Line        Addr392        07AE 0000                       nop
393         07B0 46F8                     bne     LOOP10          ;if not, loop again
394         07B2 0000                     nop                     ;delay
395         07B4 1A01                     dec     R1H             ;if yes, update the data bit counter
396         07B6 4704                     beq     PAR_CHK2        ;if last data bit, determine if parity check
397         07B8 110C                     shlr    R4L             ;if not, get next data bit
398         07BA 40CA                     bra     DATA_NXT
399                               PAR_CHK2:
400         07BC AE00                     cmp.b   #0,R6L          ;parity or not?
401         07BE 4638                     bne     PARITY2         ;if non-zero, transmit parity bit
402                               STOP_NXT:
403         07C0 FB00                     mov.b   #0,R3L          ;update compare value for delay counter loop
404         07C2 A400                     cmp.b   #0,R4H          ;determine the number of stop bits
405         07C4 4716                     beq     TWO_STP         ;if zero, 2 stop bits
406         07C6 41F8                     brn     STOP_NXT        ;delay
407         07C8 0000                     nop
408         07CA 7FC1 7000                bset.b  #0,@P9_DR       ;set stop bit level
409                               ONE_STP:
410         07CE 0A0B                     inc     R3L             ;stop bit delay loop
411         07D0 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
412         07D2 0000                     nop
413         07D4 46F8                     bne     ONE_STP         ;if not, loop again
414         07D6 7FC7 7000        RETURN: bset.b  #0,@IER         ;enable IRQ0 again
415         07DA 5670                     rte                     ;return to receive next character
416                               TWO_STP:
417         07DC 41E2                     brn     STOP_NXT        ;delay
418         07DE 0000                     nop
419         07E0 7FC1 7000                bset.b  #0,@P9_DR       ;set 1st stop bit level
420         07E4 0A0B             LOOP11: inc     R3L             ;1st stop bit delay loop
421         07E6 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
422         07E8 0000                     nop
423         07EA 46F8                     bne     LOOP11          ;if not, loop again
424         07EC FB00                     mov.b   #0,R3L          ;update the compare value for 2nd delay
425         07EE 0A0B             LOOP12: inc     R3L             ;2nd stop bit delay loop
426         07F0 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
427         07F2 0000                     nop
428         07F4 46F8                     bne     LOOP12          ;if not, loop again
429         07F6 40DE                     bra     RETURN
430                               PARITY2:
431         07F8 7306                     btst.b  #0,R6H          ;determine number of high data bits
432         07FA 4726                     beq     EV_DAT          ;if even number, go to EV_DAT
433                               OD_DAT:
434         07FC FB02                     mov.b   #2,R3L          ;set compare value for delay
435         07FE A000                     cmp.b   #0,R0H          ;even or odd parity?
436         0800 4710                     beq     EV_PAR3         ;if zero, even parity
437                               OD_PAR3:
438         0802 41F8                     brn     OD_DAT          ;delay
439         0804 F804                     mov.b   #H'04,R0L       ;parity bit is low
440         0806 38C1                     mov.b   R0L,@P9_DR
441         0808 0A0B             LOOP13: inc     R3L             ;parity bit delay loop
442         080A 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
443         080C 0000                     nop
444         080E 46F8                     bne     LOOP13          ;if not, loop again
445         0810 40AE                     bra     STOP_NXT        ;continue
446                               EV_PAR3:
447         0812 41E8                     brn     OD_DAT          ;delay
448         0814 F805                     mov.b   #H'05,R0L       ;parity bit is high
449         0816 38C1                     mov.b   R0L,@P9_DR
450         0818 0A0B             LOOP14: inc     R3L             ;parity bit delay loop
451         081A 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
452         081C 0000                     nop
453         081E 46F8                     bne     LOOP14          ;if not, loop again
454         0820 409E             BRANCH: bra     STOP_NXT        ;continue
455         0822 FB02             EV_DAT: mov.b   #2,R3L          ;set compare value for delay
456         0824 A000                     cmp.b   #0,R0H          ;even or odd parity?
457         0826 4710                     beq     EV_PAR4         ;if zero, even parity
458                               OD_PAR4:
459         0828 41F8                     brn     EV_DAT          ;delay
460         082A F805                     mov.b   #H'05,R0L       ;parity bit is high
461         082C 38C1                     mov.b   R0L,@P9_DR
462         082E 0A0B             LOOP15: inc     R3L             ;parity bit delay loop
463         0830 1CBA                     cmp.b   R3L,R2L         ;is delay count complete?
464         0832 0000                     nop
465         0834 46F8                     bne     LOOP15          ;if not, loop again
466         0836 4088                     bra     STOP_NXT        ;continue
467                               EV_PAR4:
468         0838 41E8                     brn     EV_DAT          ;delay
469         083A F804                     mov.b   #H'04,R0L       ;parity bit is low
470         083C 38C1                     mov.b   R0L,@P9_DR
471         083E 0A0B             LOOP16: inc     R3L             ;parity bit delay loop
```

```
Microtec Research ASMH83    Version 1.0A    Mar 10 15:02:32 1994     Page 6


Line        Addr
472         0840 1CBA                           cmp.b   R3L,R2L         ;is delay count complete?
473         0842 0000                           nop
474         0844 46F8                           bne     LOOP16          ;if not, loop again
475         0846 40D8                           bra     BRANCH          ;continue
476                                             .end
```