Application Note

AN2502/D
2/2004

Using Two Channels of the HC08 TIM to Achieve a Full-Duplex Software SCI

By Jorge Zambada
Motorola SPS
Mexico Applications Laboratory
Guadalajara, Mexico

Introduction

This document describes how to use the HC08 Family's 16-bit free-running timer and the timer interface module (TIM) to establish a full-duplex interrupt-driven software SCI module.

Many applications require an asynchronous serial link with other devices, but some MCUs do not have a hardware-implemented SCI module. Other applications require more than one SCI module, which is difficult to find in a low-cost microcontroller unit (MCU).

If a hardware SCI module is unavailable, a software-implemented SCI is necessary to provide the vital asynchronous serial link between an MCU and other devices. Other application notes (see **References**) describe the implementation of software SCI modules on HC05 MCUs. AN1240/D describes a "bit-banged" approach that requires dedicated software overhead while transmitting and receiving data. AN1818/D uses the 16-bit free-running counter to reduce software overhead, but this implementation on the HC05 Family can function only in half-duplex mode.

Although no software SCI can fully replace a hardware SCI's very fast baud rates, the sophisticated full-duplex implementation described in this document is a practical solution where the TIM features and some CPU time and memory can be dedicated to implementing the software SCI.

Overview

By using two channels of the TIM in an HC08 MCU, a software interrupt driven SCI module can be implemented with full-duplex operation and reduced software overhead. Each of the channels used is dedicated to a single operation; one for receiving and one for transmitting.

This product incorporates SuperFlash® technology licensed from SST.

The time required for sending a byte is 10 bit-times. A bit-time is equal to 1/baud rate. For example, if the baud rate is 9600 bps, the entire frame takes 1.0416 ms.

The transmission begins with a high-to-low transition as soon as the bus is in idle state; that is, in logic high for more than 10 bit-times. Next, the desired byte is transmitted with its least significant bit first. A byte transmission is ended with a logic level high known as the stop bit, which indicates that a transmission is finished. See **Figure 1**.

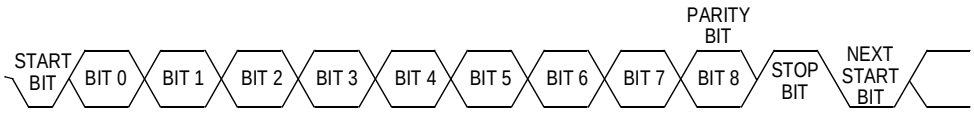

Figure 1. Non-Return-to-Zero Mark/Space Data Format

The channel used for reception is configured as an input capture on the falling edge to detect a high-to-low transition known as the start bit. When this event occurs, the input capture operation will “capture” the time at which the event occurred. Because this pin logic is hardware implemented in the HC08 MCU, the exact time of the beginning of the reception is captured.

Modes

Two operation modes of the software-implemented SCI are presented in this document: normal mode and enhanced mode.

Normal Mode

Features of normal mode:

- Full-duplex operation
- Minimal code size
- Minimal software overhead
- Easily configured for different baud rates
- Interrupt request by each reception bit
- Uses less CPU time and memory space than enhanced mode
- No error detection available
- No receiver full or transmitter empty subroutine available
- One fixed data format: eight data bits and one stop bit

Enhanced Mode

Features of enhanced mode:

- Full-duplex operation
- Minimal software overhead
- Easily configured for different baud rates
- Interrupt request by each reception bit
- Two different subroutines available for receiver full and for transmitter empty, each with an enable/disable bit
- Three flags for error detection:
  - ORE — Overrun error
  - FE — Framing error
  - PE — Parity error
- 12 easily configured data frame formats:
  - Eight or nine data bits
  - Even parity, odd parity, or no parity bit
  - One or two stop bits
- Two enable/disable bits for reception and/or transmission

Normal Mode Registers

In normal mode, five RAM variables are declared: one for status flags and four for data (receiving and transmitting).

NOTE: *All registers that have an "r" before the register name are pseudo registers that have been created in RAM. Therefore, these registers do not appear in the device data sheet. The rSCSR register in* ***Figure 2*** *is an example of a pseudo register. All other registers are normal physical registers as described in the device data sheet.*

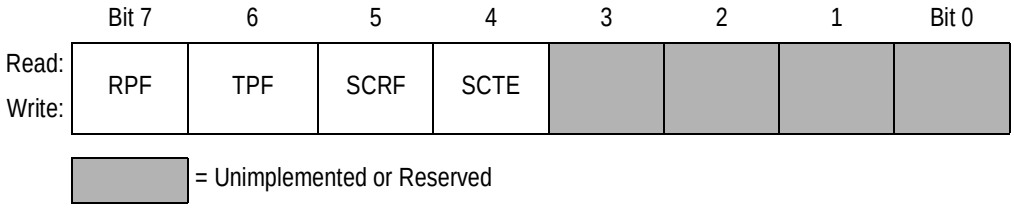

Figure 2. SCI Status Register (rSCSR)

RPF — Reception in Progress Flag
1 = Reception is in progress
0 = Reception is not in progress

TPF — Transmit in Progress Flag
1 = Transmission is in progress
0 = Transmission not in progress

SCRF — SCI Receiver Full
1 = Data available on the receive data register
0 = No data available

SCTE — SCI Transmitter Empty
1 = Transmit data register is empty
0 = Transmit data register is not empty

Because this SCI performs full-duplex operation, it is necessary to have four separate data registers; two for reception and two for transmission. Each SCI operation uses two data registers; one is used in the interrupt service routine to shift in data for reception and shift out data for transmission. These registers are shown in **Figure 3**, **Figure 4**, **Figure 5**, and **Figure 6**.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
Write:								

Figure 3. SCI Receive Data Register (rSCRDR)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0
Write:								

Figure 4. SCI Receive Shift Register (rSCRSR)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TD7	TD6	TD5	TD4	TD3	TD2	TD1	TD0
Write:								

Figure 5. SCI Transmit Data Register (rSCTDR)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0
Write:								

Figure 6. SCI Transmit Shift Register (rSCTSR)

Enhanced Mode Registers

In enhanced mode, eleven RAM variables are declared. One is the configuration register (rSCCR), two are the status registers (rSCSR1 and rSCSR2), and the other eight are data registers for transmission and reception.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TIEN	RIEN	SB	M	TEN	REN	PEN	PTY

Figure 7. SCI Configuration Register (rSCCR)

TIEN — Transmission Completed Subroutine Enable Bit
- 1 = Subroutine enabled for end of transmission
- 0 = Subroutine disabled for end of transmission

RIEN — Reception Completed Subroutine Enable Bit
- 1 = Subroutine enabled for end of reception
- 0 = Subroutine disabled for end of reception

SB — Stop Bit Selection
- 1 = Two stop bits
- 0 = One stop bits

M — Character Length Selection Bit
- 1 = Nine data bits
- 0 = Eight data bits

TEN — Transmit Enable Bit
- 1 = Transmit enabled
- 0 = Transmit disabled

REN — Receive Enable Bit
- 1 = Receive enabled
- 0 = Receive disabled

PEN — Parity Enable Bit
- 1 = Parity bit enabled
- 0 = Parity bit disabled

PTY — Parity Bit
- 1 = Odd parity
- 0 = Even parity

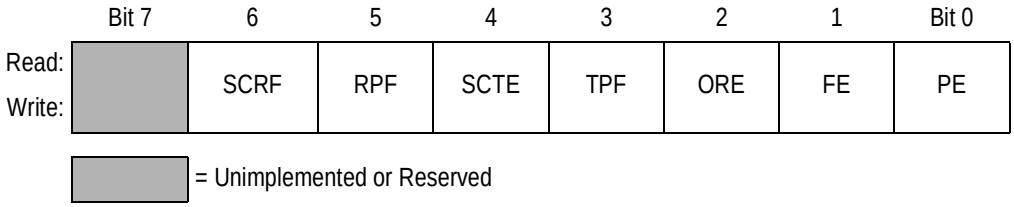

Figure 8. SCI Status Register 1 (rSCSR1)

SCRF — SCI Receiver Full Bit
- 1 = Data available on rSCRDRH:rSCRDRL
- 0 = Data not available on rSCRDRH:rSCRDRHL

RPF — Receive in Progress Flag
- 1 = Reception in progress
- 0 = Reception not in progress

SCTE — SCI Transmitter Empty
- 1 = rSCTDRH:rSCTDRL empty
- 0 = rSCTDRH:rSCTDRL not empty

TPF — Transmit in Progress Flag
- 1 = Transmission in progress
- 0 = Transmission not in progress

ORE — Overrun Error Flag
- 1 = Overrun error
- 0 = No overrun error

FE — Framing Error Flag
- 1 = Framing error
- 0 = No framing error

PE — Parity Error Flag
- 1 = Parity error
- 0 = No parity error

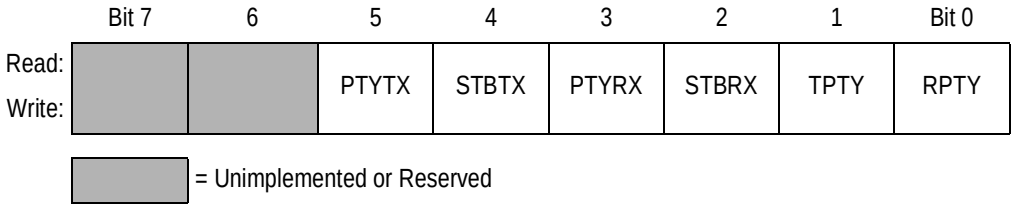

Figure 9. SCI Status Register 2 (rSCSR2)

PTYTX:STBTX — Transmitting Flags
- 00 = Transmitting data bits
- 1X = Transmitting parity bit
- 01 = Transmitting stop bits

PTYRX:STBRX — Receiving Flags
- 00 = Receiving data bits
- 10 = Receiving parity bit

01 = Receiving first stop bit
11 = Receiving second stop bit

TPTY — Transmit Temporal Parity Bit
1 = Temporal odd transmit parity
0 = Temporal even transmit parity

RPTY — Receive Temporal Parity Bit
1 = Temporal odd receive parity
0 = Temporal even receive parity

Figure 10. SCI Receive Data Registers (rSCRDRH:rSCRDRL)

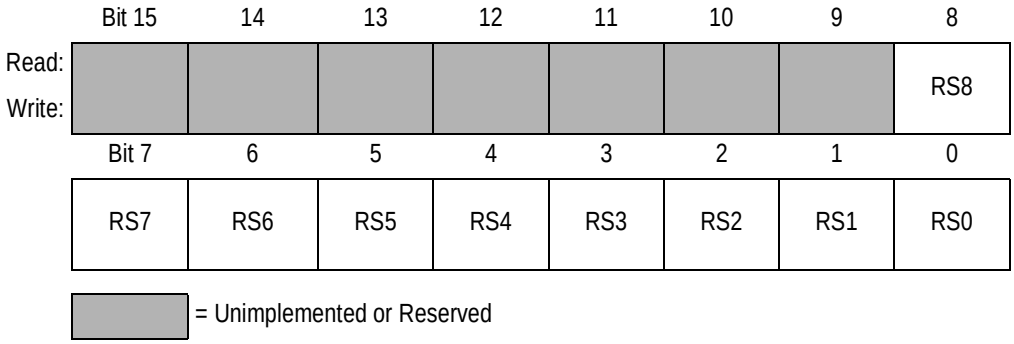

Figure 11. SCI Receive Data Registers (rSCRSRH:rSCRSRL)

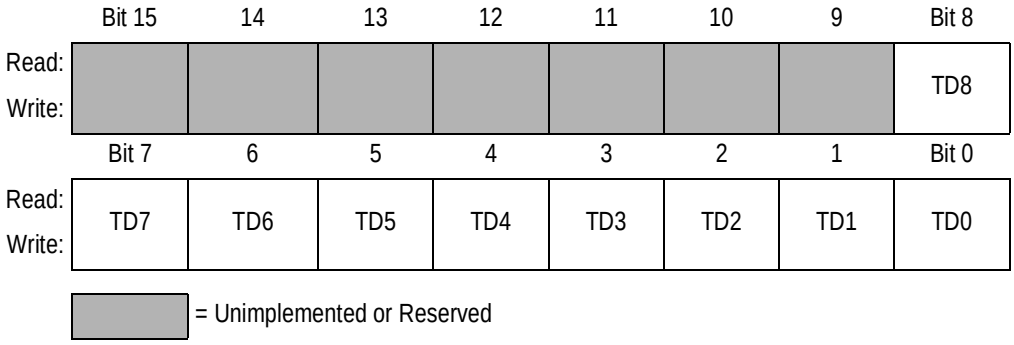

Figure 12. SCI Transmit Data Registers (rSCTDRH:rSCTDRL)

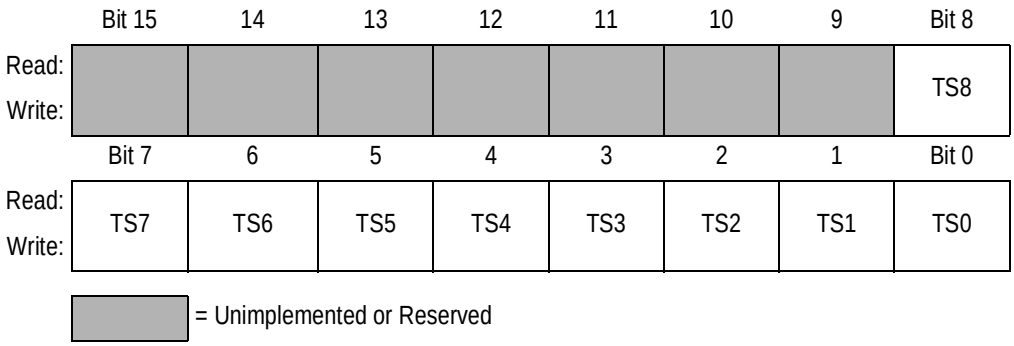

Figure 13. SCI Transmit Data Registers (rSCTSRH:rSCTSRL)

The number of bits used in the data registers is configured in the character length selection bit (M) in rSCCR.

Reception in Normal Mode

It is possible to implement full-duplex operation because two independent channels of the TIM are used for each of the operations with independent data registers.

Figure 14 is a timing diagram of the start bit reception.

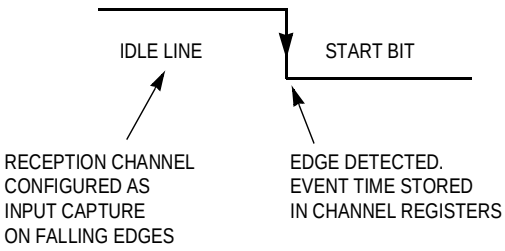

Figure 14. Receiving the Start Bit

First the channel is configured for input capture on falling edges. When this edge is detected, an interrupt service routine is asserted and serviced. Ideally, a UART (universal asynchronous receiver/transmitter) performs an over-sampling technique to ensure data integrity. In software-implemented SCIs, only one pin check per received bit is performed. The time at which this event occurs is stored in the reception channel registers by the internal hardware of the MCU. This value will be used for subsequent data bit receptions.

In the interrupt service routine (ISR) for the input capture, the time for the first received bit is set. At this point, the reception channel is configured as output compare and used as a time base for data bit receptions. Because the input capture interrupt gives a time located in a bit boundary, more than 1 bit-time

must be added to the reception channel to check the pin at the correct time. **Figure 15** shows the first data bit reception.

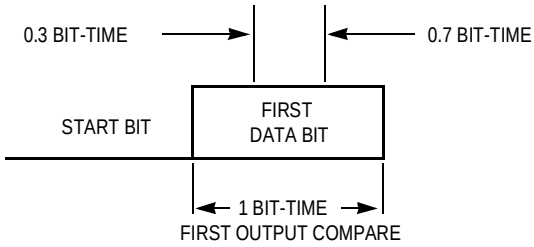

Figure 15. First Data Bit Reception

For proper operation, the instruction that checks the pin state must be executed after 0.3 of the bit-time and before 0.7 of the bit-time. This is accomplished by adding 1.3 bit-times to the reception channel as soon as the start bit has been detected. In the output compare interrupt for the reception channel, several instructions are executed before the pin check instruction. These instructions are collectively called the pin check latency in this application note.

Figure 16 illustrates how to calculate the correct time added after the start bit.

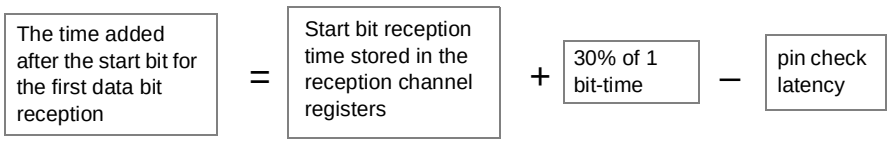

Figure 16. Time Added After the Start Bit

Figure 17 shows that the time added to the reception channel between the first to the last data bit receptions is 1 bit-time.

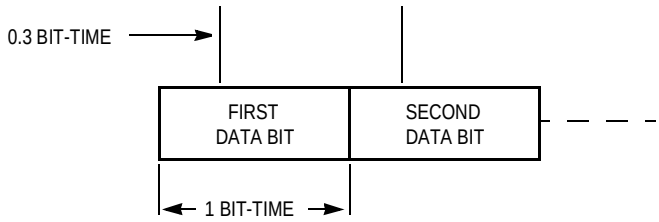

Figure 17. Data Bit Reception

As soon as the last bit is received, the received data message is moved from the reception register that is being shifted in each bit reception to a user-readable reception register. To complete a data reception, the reception channel is configured as input capture on falling edges to detect the start bit of the next data message to be received. Last bit reception is shown in **Figure 18**.

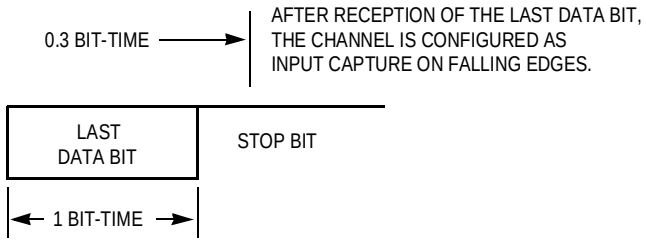

Figure 18. Last Bit Reception

Transmission in Normal Mode

For transmitting data, one of the channels of the TIM is used. Using the output compare function of the channel, it is possible to transmit bits without losing time accuracy within bit transmissions. This is because the pin logic of the output compare function is hardware implemented on the HC08 MCU.

As soon as the transmission is initiated, the SCISend subroutine is called. In this subroutine, the free-running counter of the timer is read and 1 bit-time is added and stored in the transmission channel registers.

As shown in **Figure 19**, after the time has been stored in the transmission channel registers, the transmission channel is configured as clear on output compare. This is done to send the start bit of the data to be transmitted.

Transmission channel registers = Current value of the free running timer + 1 bit-time.

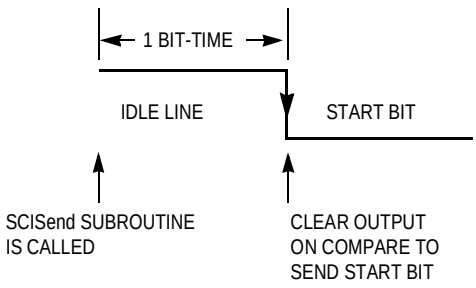

Figure 19. Start Bit Transmission

The transmission of data bits is performed by the channel interrupt service routine, which generates output compares depending on the data bit value to be sent.

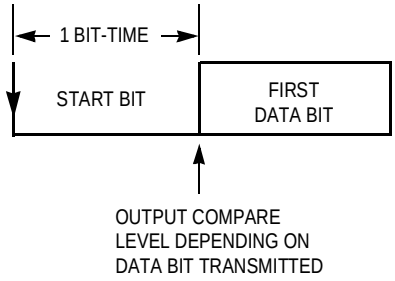

Figure 20. First Data Bit Transmission

In the interrupt service routine of the transmission channel, the channel registers are read and 1 bit-time is added to generate the next output compare for the next data bit transmission.

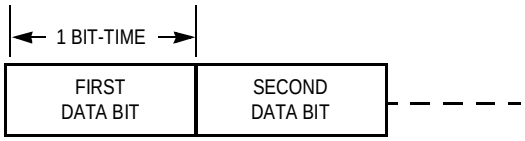

Figure 21. Data Bits Transmission

As shown in **Figure 21**, each bit to be transmitted is shifted out of the transmission shift register, and the output compare level is configured depending on the value of each bit to be transmitted. The process is repeated until the last data bit is transmitted.

In the last bit transmission interrupt, the channel is configured as set on output compare to send the stop bit. This is shown in **Figure 22**. As shown in this figure, the SCTE flag is checked before turning off the channel, so if a transmission is pending, the channel is configured as clear on output compare to send the next start bit after 1 bit-time.

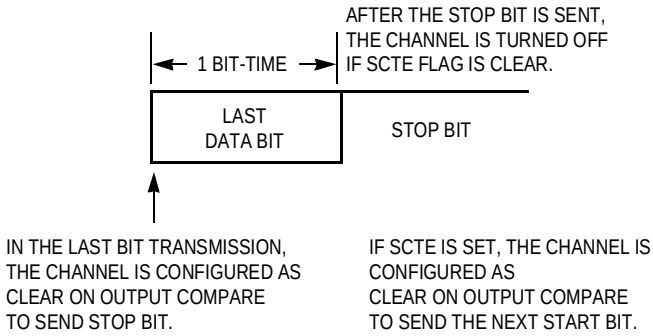

Figure 22. Stop Bit Transmission

Reception in Enhanced Mode

Steps 1 and 2 are the same in normal mode and enhanced mode:

1. Reception of the start bit
2. Data bits reception process

Steps 3 through 6 are specific to enhanced mode:

3. Number of data bits to be received
4. Parity bit computation after each bit reception
5. One or two stop bit receptions
6. Subroutine for end of reception is executed if enabled in the rSCCR

The number of data bits received is dependent on the M configuration bits in rSCCR.

Table 1. M — Data Length Configuration Bit

M	Data Bits
0	8
1	9

After a bit is shifted into the reception shift register, the received bit is considered for the parity bit computation. **Figure 23** shows the time at which the parity bit is sent, if PEN is enabled (PEN = 1).

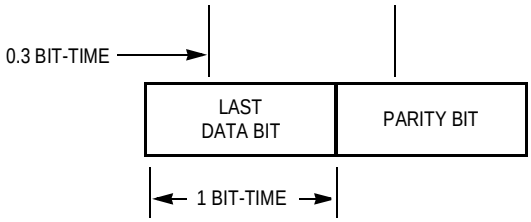

Figure 23. Parity Bit Reception

If the parity bit is enabled (PEN = 1), an extra output compare is done after the last data bit reception. The parity bit is received and compared with the value computed from the data bits previously received. The parity error PE flag is set if the computed value from the data bits differs from the one being received. The polarity of the parity bit is configured in the PTY configuration bit in the rSCCR. **Table 2** shows the parity bit configurations.

Table 2. Parity Bit Configuration Bits

PEN	PTY	Parity Configuration
0	X	No parity
1	0	Odd parity
1	1	Even parity

RPTY is the bit in which the parity for the received data is calculated. This flag is used for the calculation of the parity error. The formula for computing the parity error is:

RPTY = Data bit (0) ⊕ Data bit (1) ⊕ … ⊕ Data bit (N-1)

PE = RPTY ⊕ PTY

Table 3 summarizes the parity error combinations.

Table 3. Parity Error Truth Table

PEN	PTY	Parity Error
0	0	No error
0	1	Error
1	0	No error
1	1	Error

In enhanced mode software SCI, one or two more output compare interrupts are performed to check whether the stop bits are at a proper level. Depending on the SB bit in the configuration register rSCCR, the reception process adds either one or two output compares. If a low level is detected in the reception of the stop bits, a framing error is set in the FE flag.

Table 4. Stop Bits

SB	Stop Bits
1	Two stop bits
0	One stop bit

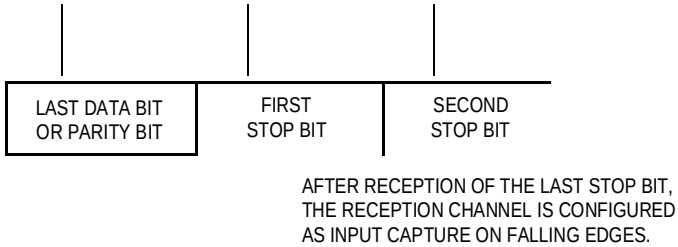

Figure 24. Stop Bits Reception

As soon as the last stop bit is received, the receiver full bit (SCRF) is checked to move the received data from the receiver shift register to the receiver data register. If the SCRF flag is set, which indicates that a valid data previously received has not been read, an overrun error is set in the ORE flag and the new received data is moved to the receiver data register.

If the reception completed subroutine is enabled (RIEN = 1) in rSCCR, the user is able to place code for executing after data is received. The following code listing shows the name of this subroutine for end of reception:

```
*********************************************************************
* Program Goes here after a Reception if RIEN = 1 in the rSCCR      *
*********************************************************************

SCIRXFULL:
; ENTER YOUR SCIRXFULL CODE HERE
            RTI
```

Table 5. Reception Completed Subroutine Enable Bit

RIEN	Configuration
1	Subroutine enabled
0	Subroutine disabled

The user must be aware that any code executed in this subroutine will affect the MCU performance because the code will be executed in the interrupt service routine.

Transmission in Enhanced Mode

Steps 1 and 2 are the same in normal mode and enhanced mode:

1. Transmission of the start bit
2. Data bits transmission process

Steps 3 through 5 are specific to enhanced mode:

3. Number of data bits to be transmitted
4. Parity bit computation
5. Second stop bit is sent when enabled in the rSCCR

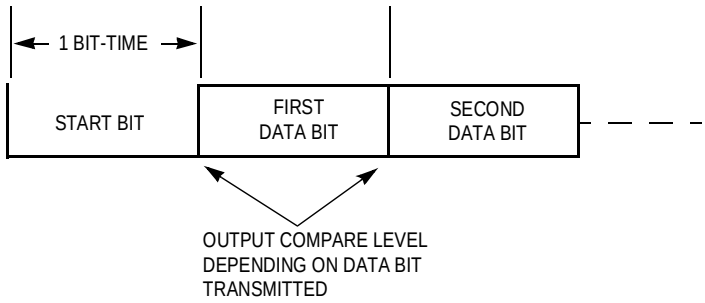

Figure 25. Data Bit Transmission in Enhanced Mode

If the parity bit is enabled (PEN = 1), the parity bit is sent after the last data bit transmission. The following formula shows the computation of the parity bit to be sent:

TPTY = Data bit (0) ⊕ Data bit (1) ⊕ … ⊕ Data bit (N-1)

Parity bit to be transmitted = (TPTY ⊕ PTY)

Table 6. Parity Bit to be Transmitted

TPTY(1)	PTY	Parity Bit to be Sent
0	0	1
0	1	0
1	0	0
1	1	1

1. TPTY bit is used for temporal computation of the parity bit.

Figure 25 shows the timing for parity bit transmission.

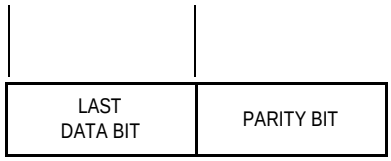

Figure 26. Parity Bit Transmission

After the parity bit is transmitted, one or two stop bits are transmitted depending on the SB bit in rSCSR1.

Table 7. Stop Bits to be Transmitted

SB	Stop Bits to be Sent
1	Two stop bits
0	One stop bit

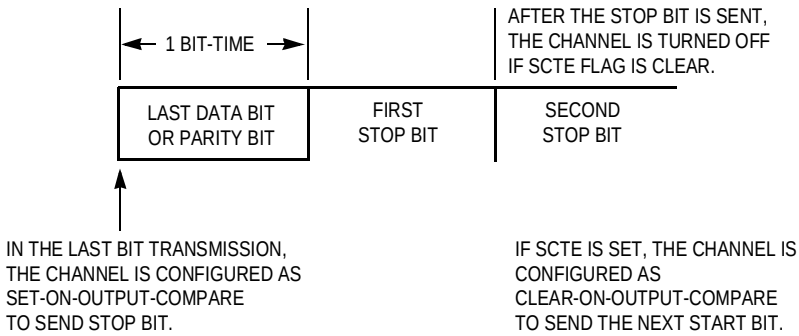

Figure 27. Transmission of Stop Bits

As soon as the full data is transmitted (including start bit, data bits, either no or one parity bit, and either one or two stop bits), an end of transmission subroutine is fetched if enabled (**TIEN** = 1) in **rSCCR**.

Table 8. Transmission Completed Subroutine Enable Bit

TIEN	Configuration
1	Subroutine enabled
0	Subroutine disabled

The following code listing shows the name of the subroutine to be fetched after a transmission is completed.

```
*********************************************************************
* Program Goes here after a transmission if TIEN = 1 in the rSCCR    *
*********************************************************************

SCITXEMPTY:
; ENTER YOUR SCITXEMPTY CODE HERE
            RTI
```

The user must be aware that any code executed in this subroutine will affect MCU performance because the code will be executed in the interrupt service routine.

Baud Rates

The baud rate is determined by the frequency of the timer module. The BITHI:BITLO is 1 bit-time, and the BIT1HI:BIT1LO is 1.3 bit-times. BITHI:BITLO = (TIM Freq.) ÷ (Baud Rate) BIT1HI:BIT1LO = (BITHI:BITLO) × 1.3 – pin check latency. The pin check latency is the number of cycles from the beginning of the reception interrupt service routine to the instruction that checks the state of the pin. From the following code listing of normal mode, the number of cycles of the interrupt entrance and the instructions are shown.

```
GetByte:  BRCLR   SCR,rSCSR          ;[.r...]
                      :
                      :
                      :
RX_isr:                                ;[9 due to interrupt entry]
            PSHH                       ;[2]
            BCLR    CH0F,TSC0          ;[4]
            BRSET   RPF,rSCSR,rxinprog ;[5]

                      :
                      :
                      :
rxinprog:
            CLC                        ;[1]
            BRCLR   RPIN,PTD,nocarry   ;[.r...]
```

This code shows that the pin check latency is: 3 + 9 + 2 + 4 + 5 + 1 + 2 = 26 + (0/+5) CPU bus cycles. In the code listings at the end of the document, the user can also find the duration of the instructions preformed prior to the pin check instruction for the enhanced mode.

Example Typical Baud Rates

Using an external crystal oscillator of 9.8304 MHz on the MC68HC908JK3, no prescaler selection for the TIM, and a pin check latency of 26 timer cycles for the normal operation of the SCI, the values of typical baud rates would be as shown in **Table 9**.

Table 9. Baud Rates for Normal Mode

	Baud Rate			
	9600	**4800**	**2400**	**1200**
BITHI:BITLO	0x0100	0x0200	0x0400	0x0800
BIT1HI:BIT1LO	0x0133	0x0280	0x0519	0x0A4C

Using the same crystal frequency and considering that the pin check latency in the enhanced mode is 28 timer cycles with no prescaler, the values of the baud rates are shown in **Table 10**.

Table 10. Baud Rates for Enhanced Mode

	Baud Rate			
	9600	**4800**	**2400**	**1200**
BITHI:BITLO	0x0100	0x0200	0x0400	0x0800
BIT1HI:BIT1LO	0x012E	0x027B	0x0514	0x0A47

The maximum baud rate is dependent on the TIM frequency and the maximum number of cycles that each of the interrupts uses. A summary of the maximum baud rates depending on SCI configuration is depicted in **Table 11**. These maximum baud rates are calculated from the maximum number of cycles that each interrupt takes during transmission and reception.

Table 11. Maximum Baud Rates for Enhanced and Normal Modes

SCI Implementation					Maximum Baud Rates			
Normal Mode					Half-Duplex		Full-Duplex	
Data Bits	Parity	Stop Bits	Max Cycles Rx	Max Cycles Tx	f_{Bus} 2.4576 MHz	f_{Bus} 8 MHz	f_{Bus} 2.4576 MHz	f_{Bus} 8 MHz
8	None	1	73	88	27927.27	90909.09	11170.91	36363.64
Enhanced Mode					Half-Duplex		Full-Duplex	
Data Bits	Parity	Stop Bits	Max Cycles Rx	Max Cycles Tx	f_{Bus} 2.4576 MHz	f_{Bus} 8 MHz	f_{Bus} 2.4576 MHz	f_{Bus} 8 MHz
8	None	1	168	137	14628.57	47619.05	7175.47	23357.66
9	None	1	163	142	15077.30	49079.75	6922.82	22535.21
8	Even	1	167	137	14716.17	47904.19	7175.47	23357.66
9	Even	1	162	142	15170.37	49382.72	6922.82	22535.21
8	Odd	1	167	137	14716.17	47904.19	7175.47	23357.66
9	Odd	1	162	142	15170.37	49382.72	6922.82	22535.21
8	None	2	151	151	16275.50	52980.13	6510.20	21192.05
9	None	2	146	156	15753.85	51282.05	6301.54	20512.82
8	Even	2	151	151	16275.50	52980.13	6510.20	21192.05
9	Even	2	146	156	15753.85	51282.05	6301.54	20512.82
8	Odd	2	151	151	16275.50	52980.13	6510.20	21192.05
9	Odd	2	146	156	15753.85	51282.05	6301.54	20512.82

NOTE:
Table 11 does not consider latency for other interrupts, so the user must compute the final performance required for the MCU to implement a specific application.

Half-Duplex Maximum Baud Rate

When receiving data in half-duplex operation, the reception line must be checked within 30% and 70% of 1 bit-time. When adding 1.3 of 1 bit-time for the reception of the first bit as shown in **Figure 15**, the number of timer counts to be added to the reception channel register is 1 bit-time multiplied by 1.3 minus pin check latency. The resulting number of counts to be added must be greater than 1 bit-time for the half-duplex operation. This result gives the first boundary condition for half-duplex operation.

When receiving or transmitting data in half-duplex, the interrupt service routines must be performed in fewer CPU cycles than the bit-time calculated for the desired baud rate, which is another boundary condition. The resulting formula for the maximum baud rate in half-duplex is:

Maximum baud rate = f_{Bus}/(MAX(Latency/0.3, Max Cycles Rx, Max Cycles Tx)

Full-Duplex Maximum Baud Rate

When receiving data in the full-duplex operation based on **Figure 15**, a read of the input pin for the bit being received must be performed within 30% and 70% of 1 bit-time to check the state of the pin in the center. For full-duplex operation, the value of the bit-time multiplied by 0.4 (70% to 30%) in timer counts must be greater that the number of cycles for the transmission interrupt. Therefore, if the transmission interrupt arrives before the reception interrupt, as may happen as a worst case, the reception line will be checked in the 70% of the bit-time.

When transmitting data in the full-duplex operation, both reception and transmission interrupts must be completed in fewer cycles than the bit-time in timer counts. Both boundaries are taken into account to calculate the maximum baud rate (BR) in which the SCI will fit. The following formula shows this boundary:

Max. BR = f_{Bus} /(MAX(max. cycles Tx/0.4, max. cycles Rx + max. cycles Tx)

Hardware

Figure 28 shows the connection between the HC08 MCU and a MAX232 physical interface.

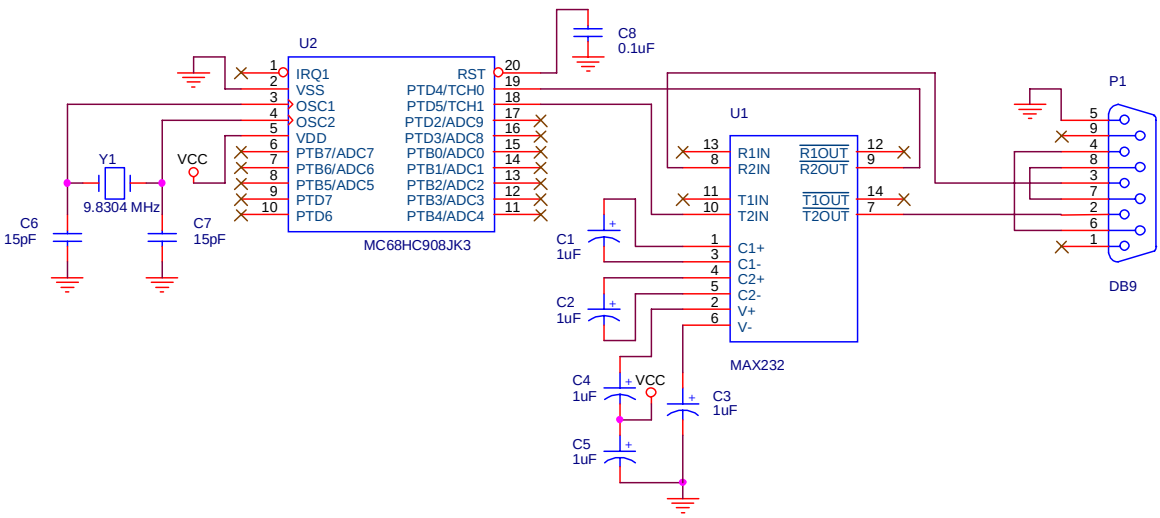

Figure 28. HC08 to MAX232 Interface Connection

Software

The two modes of operation in this application note are presented in this document with flowcharts and assembly source code.

Normal Mode Flowcharts

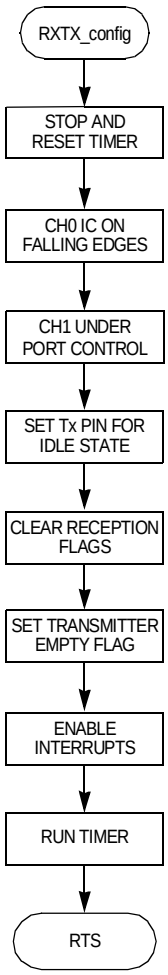

Figure 29. Initial Configuration — Normal Mode

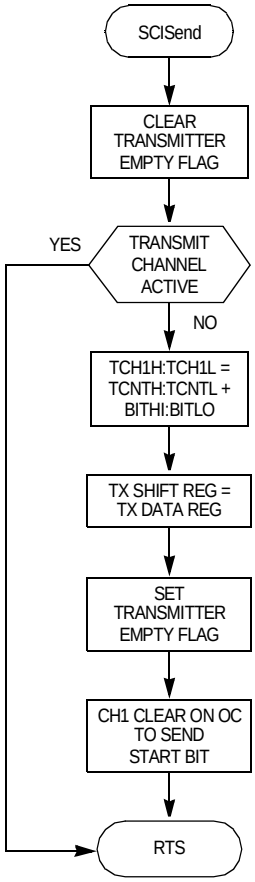

Figure 30. SCI Send — Normal Mode

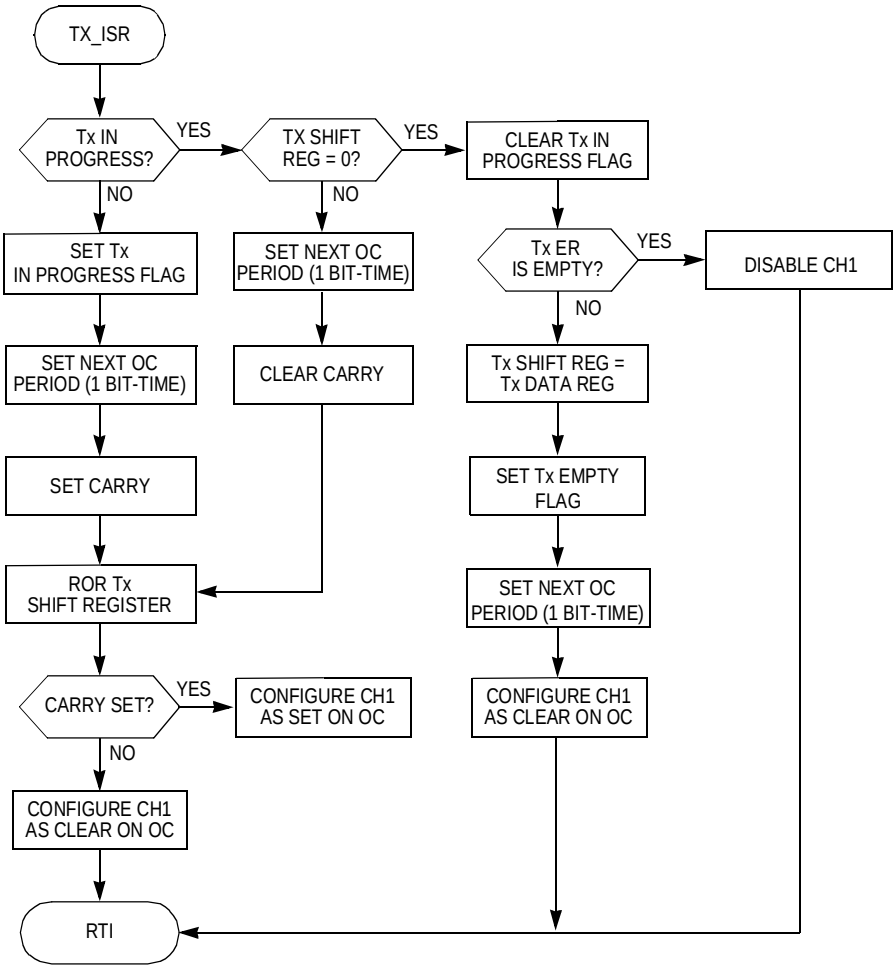

Figure 31. Transmit ISR — Normal Mode

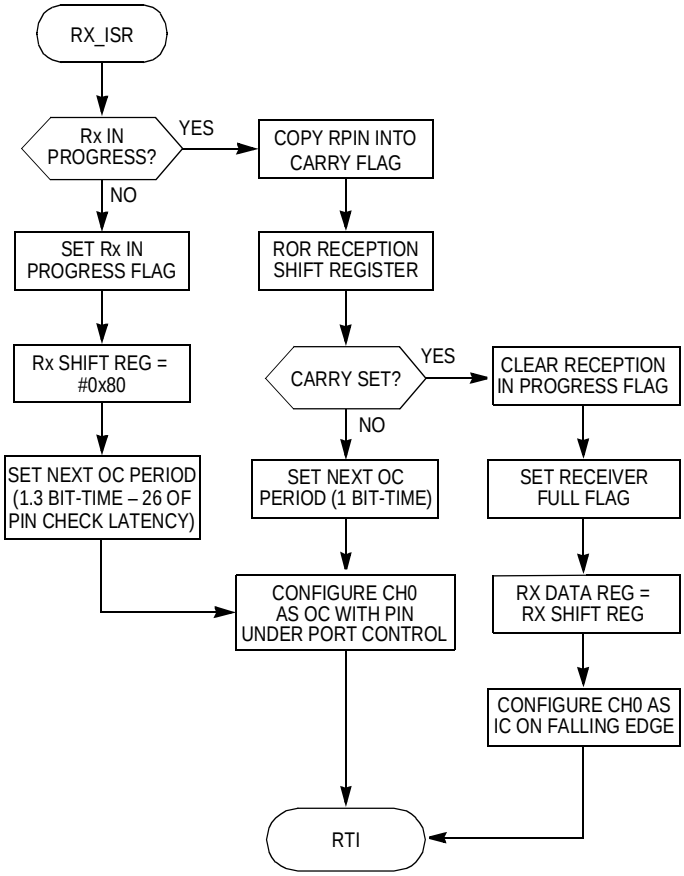

Figure 32. Receive ISR — Normal Mode

Normal Mode Code Listing

```
;*********************************************************************
;* SCI NORMAL.asm                                                    *
;*********************************************************************
;* A software interrupt driven SCI module using two TIM channels for  *
;* the HC08 MCU. Normal mode.                                        *
;*                                                                   *
;* By Jorge Zambada                                                  *
;* Motorola SPS                                                      *
;* Mexico Applications Laboratory                                    *
;* Guadalajara, Mexico                                               *
;* 2003                                                              *
;*********************************************************************
;* NOTES:                                                            *
;*********************************************************************
;* 1) In this code listing, one of the low cost family of the HC08,the*
;* JK3 MCU is used. This MCU has two TIM channels, one used for       *
;* transmission (CH1), and the other for reception (CH0).            *
;* 2) In the SCISend, a transmission is started for the byte stored in*
;* the transmission data register rSCTDR. In RXTX_Config are         *
;* configured the two channels for the two processes: reception and   *
;* transmission.                                                     *
;* 3) Two different ISR, one for reception and one for transmission,  *
;* makes possible the full duplex operation of this software SCI      *
;* module.                                                           *
;* 4) Bit times are calculated with a free running timer (TIM).      *
;* 5) A list of standard baud rates is shown below. Remember that the *
;* maximum baud rate is proportional to the input clock frequency of  *
;* the TIM.                                                          *
;*********************************************************************

VectorStart EQU   $FFDE

; For TIM Prescaler of 1
; (Bus Freq)/(Baud Rate)=BITHI:BITLO ->1 bit time
; BITHI:BITLO x 1.3-26=BIT1HI:BIT1LO->1.3 bit time - pin check latency

; Example of 9600 bps, XTAL = 9.8304 MHz -> Bus Freq = 9.8304 MHz/4
; BITHI:BITLO = 2457600/9600 = $0100
; BIT1HI:BIT1LO = $0100*1.3 - 26 = $0133

;9600 baud
BITHI       EQU   $01
BITLO       EQU   $00
BIT1HI      EQU   $01
BIT1LO      EQU   $33

;4800 baud
;BITHI       EQU   $02
;BITLO       EQU   $00
;BIT1HI      EQU   $02
;BIT1LO      EQU   $80

;2400 baud
```

```
;BITHI       EQU   $04
;BITLO       EQU   $00
;BIT1HI      EQU   $05
;BIT1LO      EQU   $19

;1200 baud
;BITHI       EQU   $08
;BITLO       EQU   $00
;BIT1HI      EQU   $0A
;BIT1LO      EQU   $4C

; MISC Flags
TPIN        EQU   5           ; TX PIN PORT D
RPIN        EQU   4           ; RX PIN PORT D

; Bit positions on the SCIFlag register
RPF         EQU   7           ; Receive in progress
TPF         EQU   6           ; Transmit in progress
SCRF        EQU   5           ; SCI Receiver Full
SCTE        EQU   4           ; SCI Transmitter Empty

; Include file for the 68HC908JL3, 68HC908JK3, 68HC908JK1
        Include 'jk3_registers.inc'

            ORG   RamStart

rSCSR       RMB   1           ; SCI Status Register
rSCRDR      RMB   1           ; SCI Receive Data Register
rSCRSR      RMB   1           ; SCI Receive Shift Register
rSCTDR      RMB   1           ; SCI Transmit Data Register
rSCTSR      RMB   1           ; SCI Transmit Shift Register

            ORG   RomStart

;**********************************************************************
;* Program goes here after reset                                      *
;**********************************************************************

Start:

            RSP               ; Reset Stack Pointer
            CLRX              ; Initialize MCU registers
            CLRH
            CLRA
            BSET    0,CONFIG1 ; Disable Watch Dog Timer

; Call this subroutine for enable Tx and Rx
            JSR     RXTX_config
                              ; configure TX and RX. Call this
                              ; function with the desired config.
                              ; before any software SCI operation

; ENTER YOUR MAIN CODE HERE. This is an example of receiving a byte
; which is stored in the accumulator and then sending the same byte.
```

```
Again:
            JSR     GetByte   ; Wait for a byte reception and store
                              ; the received data in the accumulator
            JSR     PutByte   ; Store the accumulator content to the
                              ; transmission register and wait for
                              ; the transmission to be completed
            BRA     Again

;*********************************************************************
;* This Subroutine configures the two channels:                      *
;* RXTX_config:                                                      *
;* -> Stop and reset timer.                                          *
;* -> Input capture on falling edges with interrupts enabled, so that *
;* we can detect the start bit.                                      *
;* -> Disable channel and set initial value of that pin              *
;* (IDLE STATE)                                                      *
;* -> Set transmitter empty flag, to be prepared for first           *
;* transmission                                                      *
;* -> Port configuration                                             *
;* -> Enable interrupts                                              *
;* -> Run timer                                                      *
;*********************************************************************

RXTX_config:
            LDA     #$30      ; stop and reset timer
            STA     TSC
            MOV     #$48,TSC0 ; IC, Falling edge, with interrupts
            MOV     #$00,TSC1 ; OC, output preset HIGH
            MOV     #$10,rSCSR
                              ; Set Tx Empty Flag
            BSET    TPIN,PTD  ; Set pin for Idle State
            BSET    TPIN,DDRD
            BCLR    RPIN,DDRD
            CLI               ; Enable all interrupts
            BCLR    TSTOP,TSC ; enable timer counter
            RTS

;*********************************************************************
;* GetByte subroutine is a friendly usage of the SCI Reception       *
;* features, which provides Flags check and storage of the received  *
;* byte in the accumulator. The User would modify this subroutine if *
;* the received byte needs to be stored in another register or memory *
;* location.                                                         *
;*********************************************************************

GetByte:
            BRCLR   SCRF,rSCSR,*
                              ; 1........ [3 CYCLES of instruction
                              ;            prior to RX interrupt]
                              ; Wait for byte to be received. This
                              ; flag is set when the received byte is
                              ; moved from the reception shift
                              ; register to the reception data
                              ; register.
            LDA     rSCRDR    ; Store received byte in the accumulator
```

```
            BCLR    SCRF,rSCSR
                            ; Clear receiver full flag to allow
                            ; more receptions
            RTS

;*********************************************************************
;* PutByte subroutine is a friendly usage of the SCI Transmission     *
;* features, which provides Flags check and storage from the          *
;* accumulator to the transmission register. User would modify this   *
;* subroutine if the byte to be sent is stored in another register or *
;* location.                                                          *
;*********************************************************************

PutByte:
            STA     rSCTDR    ; Store the accumulator content in the
                              ; transmission shift register
            JSR     SCIsend   ; Start byte transmission
            BRCLR   SCTE,rSCSR,*
                              ; Wait for the byte to be transferred
                              ; to the transmission shift register
            RTS

;*********************************************************************
;* SCIsend.                                                           *
;* PutByte calls this subroutine.                                     *
;* if no transmission is in progress, this subroutine configures the  *
;* transmit channel and calculates the time to send the start bit.    *
;* The value stored in TCH1H:TCH1L represent a time dependent to the  *
;* baud rate referenced as BITHI:BITLO. If a transmission is in       *
;* progress, the byte to be transmitted is queued for transmission.   *
;*********************************************************************

SCIsend:
            BCLR    SCTE,rSCSR
                            ; Clear transmitter empty flag
            BRSET   CH1IE,TSC1,SCIsend_end
                            ; If transmission channel
                            ; is active, the byte to be transmitter
                            ; is queued and exits SCISend subroutine
            BCLR    CH1F,TSC1 ; Program goes here if there is no
                            ; transmission in progress
            LDHX    TCNTH     ; Read current count
            TXA
            ADD     #BITLO    ; Add 1 bit time for
                              ; next compare to send start bit.
            TAX
            PSHH
            PULA
            ADC     #BITHI
            PSHA
            PULH
            STHX    TCH1H     ; Store calculated value in transmission
                              ; channel.
            BSET    SCTE,rSCSR
                              ; Set transmitter empty flag,
```

```
                        ; indicating that a new data to be
                        ; transmitted can be queued.
            MOV     rSCTDR,rSCTSR
                        ; Move data from Tx Register to
                        ; Tx Shift Register
            MOV     #$58,TSC1 ; Config. channel 1 as OC with
                        ; interrupts enabled.
                        ; Clear on output compare for start bit
SCIsend_end:
            RTS

;*********************************************************************
;* This ISR is dedicated only for transmission. each transmitted bit  *
;* including start bit and stop bit generate this interrupt.          *
;* Each transmission bit configures the polarity of output compare:   *
;* for example, if the next transmission bit is a logic '1', the      *
;* channel is configured as set on output compare after a 1 bit time. *
;* If there is a transmission in progress, this routine stores a $80  *
;* in the transmit shift register for 8 transmission bits plus a Stop *
;* Bit. If there is no byte queued to be transmitted, the             *
;* transmission channels are disabled for Idle State. If there is a   *
;* byte to be transmitted in the queue, the time for next start bit   *
;* is calculated and the new transmission is started.                 *
;*********************************************************************

TX_isr:

            PSHH
            BCLR    CH1F,TSC1
            BRSET   TPF,rSCSR,txinprog
                        ; Check if there's a
                        ; transmission in progress
            BSET    TPF,rSCSR ; Program goes here if no transmission
                        ; is in progress
            LDHX    TCH1H     ; Calculate the time for next bit to
                        ; be transmitted. Start bit was already
                        ; sent by the SCISend subroutine
            TXA
            ADD     #BITLO    ; Set time for next OC
            TAX               ; after 1 bit time
            PSHH
            PULA
            ADC     #BITHI
            PSHA
            PULH
            STHX    TCH1H     ; Store the calculated time in the
                        ; channel register for next output
                        ; compare
            SEC
            ROR     rSCTSR    ; With this operation, the bit to be
                        ; transmitted is copied in the carry
                        ; flag
oc_highorlow:
            BCS     oc_high   ; if Carry is High, the transmission
                        ; channel is configured to set on
```

```
                        ; output compare
oc_low:
            MOV     #$58,TSC1 ; If carry cleared, config.
                        ; as clear on output compare
            PULH
            RTI
oc_high:
            MOV     #$5C,TSC1 ; If carry set, config.
                        ; as set on output compare
            PULH
            RTI
txinprog:
            LDA     rSCTSR    ; Program goes here if a transmission is
                        ; in progress
            CBEQA   #$00,txfinished
                        ; If SCIDatatx is cleared,
                        ; means that all data bits were
                        ; sent, including the stop bit
            LDHX    TCH1H     ; if transmission is in progress and
                        ; has no finished transmitting, the time
                        ; for the next compare to send next bit
                        ; is added to the transmission channel
                        ; register
            TXA
            ADD     #BITLO    ; Add 1 bit time for next compare
            TAX
            PSHH
            PULA
            ADC     #BITHI
            PSHA
            PULH
            STHX    TCH1H
            CLC
            ROR     rSCTSR    ; Once the time for the next compare is
                        ; calculated, the polarity of the next
                        ; compare is taken from the transmission
                        ; shift register
            BRA     oc_highorlow
txfinished:
            BCLR    TPF,rSCSR ; Indicate that No Tx is in progress
            BRSET   SCTE,rSCSR,NoTxPending
                        ; Check if another byte
                        ; is pending for transmission.
            BSET    SCTE,rSCSR
                        ; If a byte is pending for transmission
                        ; the transmitter empty flag is empty
                        ; to allow other byte to be queued.
            MOV     rSCTDR,rSCTSR
                        ; Move data from Tx Register to
                        ; Tx Shift Register
            LDHX    TCH1H     ; One bit time is added to the
                        ; transmission channel to send start
                        ; bit of the next byte to be sent
            TXA
            ADD     #BITLO    ; Add 1 bit time for next compare
```

```
            TAX
            PSHH
            PULA
            ADC     #BITHI
            PSHA
            PULH
            STHX    TCH1H
            BRA     oc_low    ; Send start bit for pending Tx
NoTxPending:
            CLR     TSC1      ; Disable transmission Channel
            PULH
            RTI

;*********************************************************************
;* This ISR is dedicated for reception.                              *
;* When the receiving line is in idle state, this channel is         *
;* configured as input capture on falling edge, waiting for a start  *
;* bit. When the start bit is received, the channel is configured as *
;* output compare with pins under port control, since the output     *
;* compare will be used only as a timing reference for the bits      *
;* reception. In the first data bit reception, the time added to the *
;* reception channel is 1.3 bit time minus 26 of pin check latency,  *
;* so the instruction that check the pin state does not check the    *
;* state in the bit time boundary. This latency is measured from the *
;* beginning of the ISR to the instruction that checks the pin state.*
;* In each consecutive output compare the pin state is read and      *
;* shifted into the reception shift register. When all the data bits *
;* are received, the stop bit is ignored and the channel is          *
;* again configured as input capture on falling edges to detect the  *
;* next start bit                                                    *
;*********************************************************************

RX_isr:                       ; 2........ [9 CYCLES Interrupt Entrance]

            PSHH              ; 3........ [2 CYCLES]
            BCLR    CH0F,TSC0 ; 4........ [4 CYCLES]
            BRSET   RPF,rSCSR,rxinprog
                              ; 5........ [5 Cycles]
                              ; check if Rx in progress
            BSET    RPF,rSCSR ; set receive in progress flag. Program
                              ; goes here if no transmission is in
                              ; progress.
            MOV     #$80,rSCRSR
                              ; Store $80 which represents a stop bit
                              ; after 8 transmission bits.
            LDHX    TCH0H     ; Since this is the first time we enter
                              ; the ISR for the reception, a 1.3 bit
                              ; time minus pin check latency must be
                              ; added to the channel register.
            TXA
            ADD     #BIT1LO   ; Add 1.3 bit time minus pin check lat.
            TAX
            PSHH
            PULA
            ADC     #BIT1HI
```

```
            PSHA
            PULH
            STHX    TCH0H
            MOV     #$50,TSC0 ; config. channel 0 as output
                              ; compare with int enabled
                              ; Pin under port control to use the
                              ; output compare function just as a
                              ; timing interrupt.
            PULH
            RTI
rxinprog:
            CLC               ; 6........ [1 CYCLES]
                              ; program goes here if there is a
                              ; reception in progress. The carry
                              ; flag is cleared.
            BRCLR   RPIN,PTD,nocarry
                              ; 7........ [2 CYCLES from instruction
                              ;            fetch to the pin reading]
                              ; PIN CHECK LATENCY IS:
                              ; 3 + 9 + 2 + 4 + 5 + 1 + 2 = 26 Cycles
                              ; of pin check latency
            SEC               ; Copy pin state into carry flag
nocarry:
            ROR     rSCRSR    ; Rotate memory to save the received
                              ; bit into the receive shift register.
            BCS     rxfinished
                              ; If carry=1 after rotation, means
                              ; that next Tx bit is the Stop Bit.
            LDHX    TCH0H     ; Program goes here if there are still
                              ; bits to receive.
            TXA
            ADD     #BITLO    ; Add 1 bit time
            TAX
            PSHH
            PULA
            ADC     #BITHI
            PSHA
            PULH
            STHX    TCH0H
            MOV     #$50,TSC0 ; config. channel 0 as OC with
                              ; interrupts. Pin under port control
            PULH
            RTI
rxfinished:
                              ; In this version of the SCI, the
                              ; stop bit is not checked, since
                              ; there is no error detection.
            BCLR    RPF,rSCSR ; Clear reception in progress flag
            BSET    SCRF,rSCSR
                              ; Indicate that there's a valid data
                              ; on SCIDatarx
            MOV     rSCRSR,rSCRDR
            MOV     #$48,TSC0 ; IC, Falling edge, with ints. enabled
            PULH
            RTI
```

```
;*********************************************************************
;* Dummy ISR                                                         *
;*********************************************************************

dummy_isr:
            BRA     dummy_isr
            RTI               ; return

;*********************************************************************
;* Vector definitions                                                *
;*********************************************************************

            ORG     VectorStart

            FDB     dummy_isr ; ADC Conversion Complete Vector
            FDB     dummy_isr ; Keyboard Vector
            FDB     dummy_isr ; (No Vector Assigned $FFE2-$FFE3)
            FDB     dummy_isr ; (No Vector Assigned $FFE4-$FFE5)
            FDB     dummy_isr ; (No Vector Assigned $FFE6-$FFE7)
            FDB     dummy_isr ; (No Vector Assigned $FFE8-$FFE9)
            FDB     dummy_isr ; (No Vector Assigned $FFEA-$FFEB)
            FDB     dummy_isr ; (No Vector Assigned $FFEC-$FFED)
            FDB     dummy_isr ; (No Vector Assigned $FFEE-$FFEF)
            FDB     dummy_isr ; (No Vector Assigned $FFF0-$FFF1)
            FDB     dummy_isr ; TIM1 Overflow Vector
            FDB     TX_isr    ; TIM1 Channel 1 Vector
            FDB     RX_isr    ; TIM1 Channel 0 Vector
            FDB     dummy_isr ; (No Vector Assigned $FFF8-$FFF9)
            FDB     dummy_isr ; ~IRQ1
            FDB     dummy_isr ; SWI Vector
            FDB     Start     ; Reset Vector
```

Enhanced Mode Flowcharts

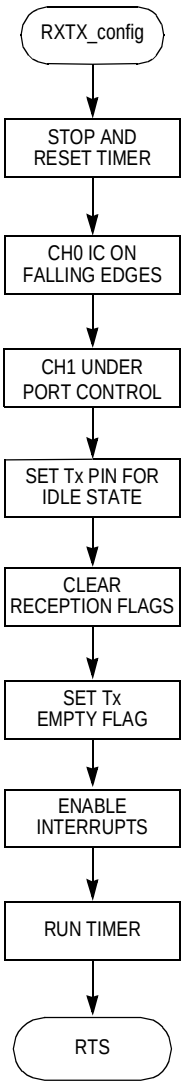

Figure 33. Initial Configuration — Enhanced Mode

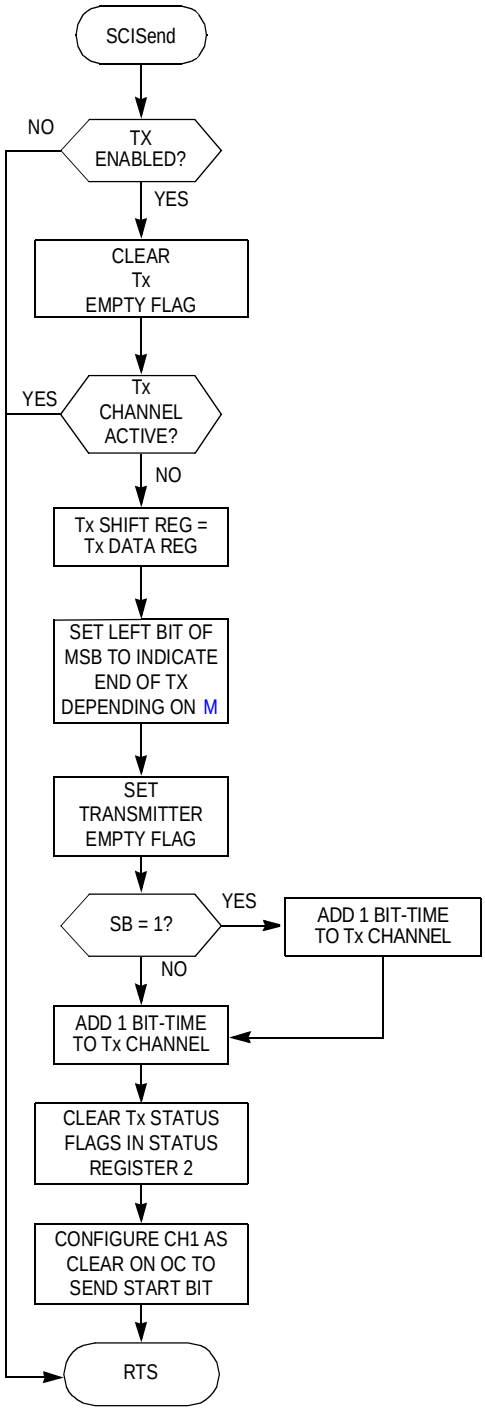

Figure 34. SCI Send — Enhanced Mode

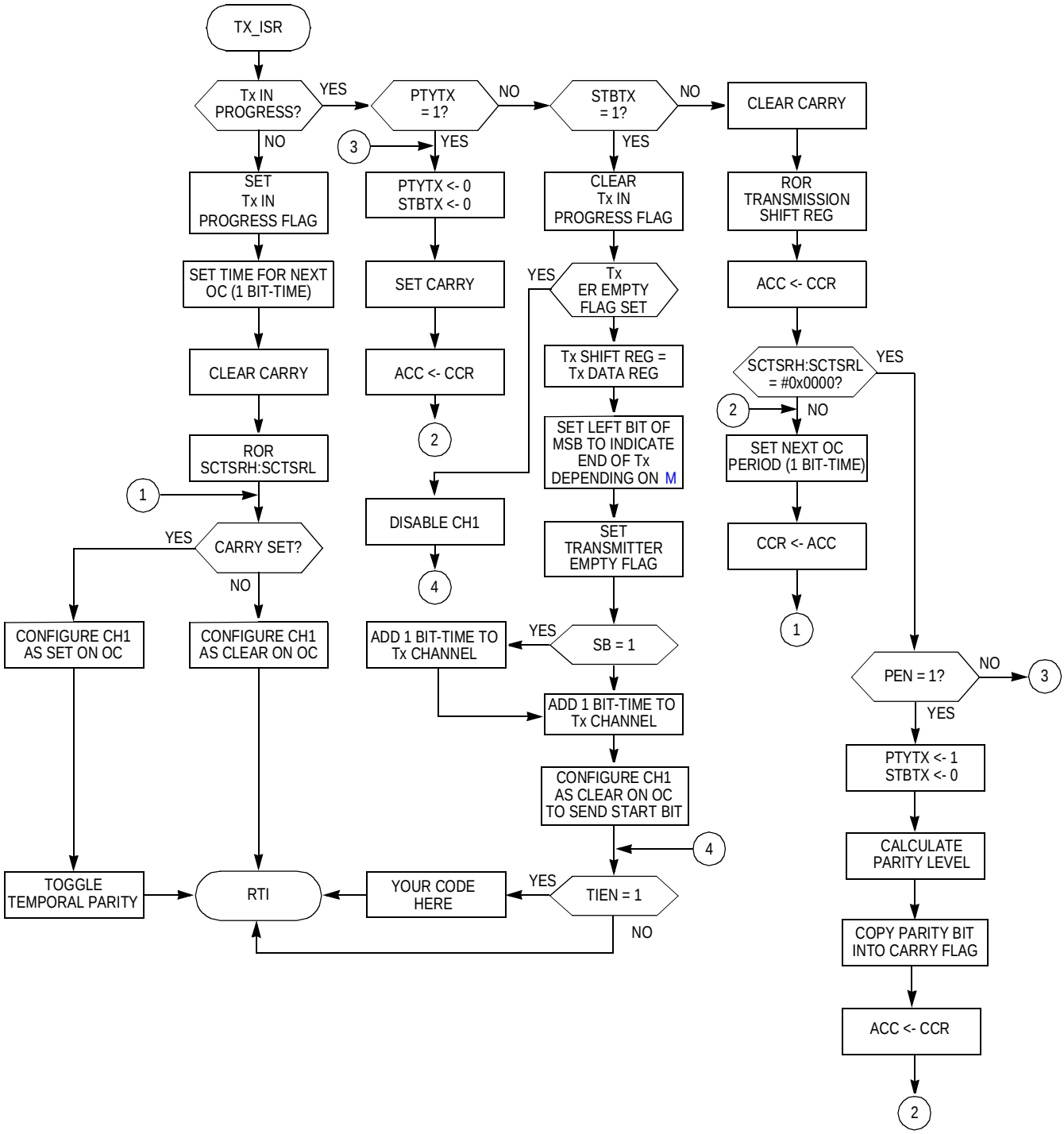

Figure 35. Transmit ISR — Enhanced Mode

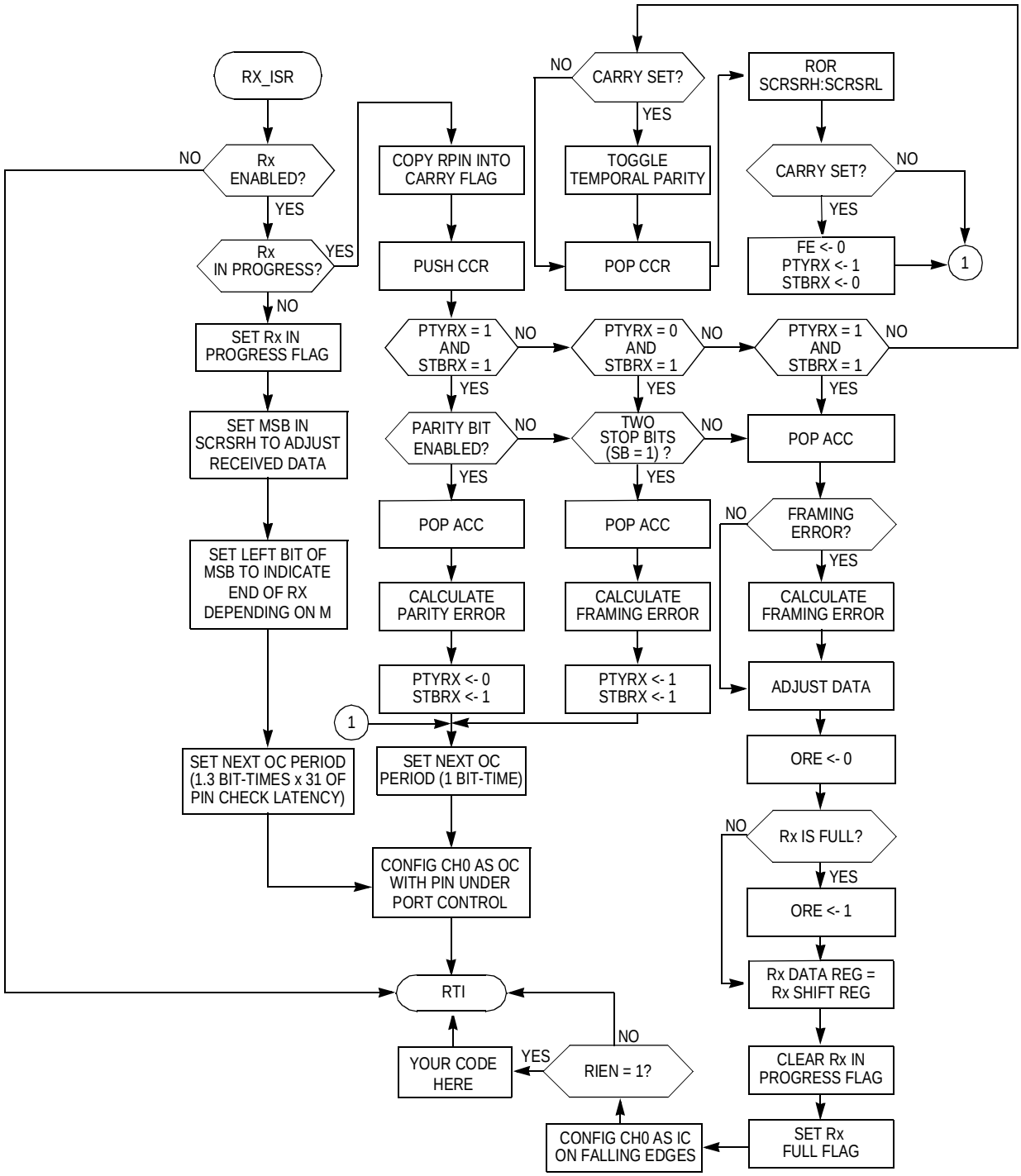

Figure 36. Receive ISR — Enhanced Mode

Enhanced Mode Code Listing

```
;*************************************************************************
;* SCI ENHANCED.ASM                                                      *
;*************************************************************************
;* A software interrupt driven SCI module using two TIM channels for  *
;* the HC08 MCU. Enhanced mode.                                          *
;*                                                                       *
;* By Jorge Zambada                                                      *
;* Motorola SPS                                                          *
;* Mexico Applications Laboratory                                        *
;* Guadalajara, Mexico                                                   *
;* 2003                                                                  *
;*************************************************************************
;* NOTES:                                                                *
;*************************************************************************
;* 1) In this code listing we use one of the low cost family of the   *
;* HC08, the JK3 MCU. This MCU has two TIM channels, we use one for   *
;* transmission, and the other for reception. In this example we use  *
;* CH1 for transmission and CH0 for reception.                           *
;* 2) In the SCISend,we start a transmission of the data stored in the*
;* transmission register rSCTDRL for 8 bit transmissions, and in      *
;* rSCTDRH:rSCTDRL for 9 bit transmissions. In RXTX_config are        *
;* configured the two channels for the two processes: reception and   *
;* transmission.                                                         *
;* 3) Two different ISR (one for reception and one for transmission)  *
;* makes possible the full duplex operation of this software SCI      *
;* module.                                                               *
;* 4) Bit times are calculated with a free running timer (TIM).       *
;* 5) A list of standard baud rates is shown below. Remember that the *
;* maximum baud rate is proportional to the input clock frequency of  *
;* the TIM.                                                              *
;* 6) Two subroutines are provided for Transmitter empty and Receiver *
;* full if enabled in the configuration register, rSCCR.              *
;* 7) Three flags in the rSCSR make possible error detection:         *
;* ORE -> Receive overrun error. Is set when SCI Receiver Full SCRF   *
;* flag is set and a new reception is beginning.                         *
;* PE -> Receive parity error. Is set when a received parity bit      *
;* is not equal to the parity calculation.                               *
;* FE -> Framing error. Is set when receiving a logic '0' in the      *
;* stop bit reception.                                                   *
;*************************************************************************

VectorStart EQU   $FFDE

; For TIM Prescaler of 1
; (Bus Freq)/(Baud Rate)=BITHI:BITLO ->1 bit time
; BITHI:BITLO * 1.3-31=BIT1HI:BIT1LO->1.3 bit time - pin check latency

; Example of 9600 bps, XTAL = 9.8304 MHz -> Bus Freq = 9.8304 MHz/4
; BITHI:BITLO = 2457600/9600 = $0100
; BIT1HI:BIT1LO = $0100*1.3 - 31 = $012E

;9600 baud
BITHI       EQU   $01
```

```
BITLO       EQU   $00
BIT1HI      EQU   $01
BIT1LO      EQU   $2E

;4800 baud
;BITHI       EQU   $02
;BITLO       EQU   $00
;BIT1HI      EQU   $02
;BIT1LO      EQU   $7B

;2400 baud
;BITHI       EQU   $04
;BITLO       EQU   $00
;BIT1HI      EQU   $05
;BIT1LO      EQU   $14

;1200 baud
;BITHI       EQU   $08
;BITLO       EQU   $00
;BIT1HI      EQU   $0A
;BIT1LO      EQU   $47

; MISC Flags
TPIN        EQU   5           ; TX PIN PORT D
RPIN        EQU   4           ; RX PIN PORT D

; Bit positions on the SCI Control Register (rSCCR)
TIEN        EQU   7           ; Tx empty interrupt enable bit
RIEN        EQU   6           ; Rx full interrupt enable bit
SB          EQU   5           ; Stop Bit Selection
M           EQU   4           ; Character Length Selection
TEN         EQU   3           ; Transmit Enable
REN         EQU   2           ; Receive Enable
PEN         EQU   1           ; Parity Enable
PTY         EQU   0           ; Parity Bit

; Bit positions on the SCI Status Register 1 (rSCSR1)
SCRF        EQU   6           ; Receive Data Register Full
RPF         EQU   5           ; Receive in progress
SCTE        EQU   4           ; Transmit Data Register Empty
TPF         EQU   3           ; Transmit in progress
ORE         EQU   2           ; Overrun Error
FE          EQU   1           ; Framing Error
PE          EQU   0           ; Parity Error

; Bit positions on the SCI Status Register 2 (rSCSR2)
PTYTX       EQU   5           ; Sending PTY
STBTX       EQU   4           ; sending STB
PTYRX       EQU   3           ; Receiving PTY
STBRX       EQU   2           ; Receiving STB
TPTY        EQU   1           ; Temporal Tx Parity bit
RPTY        EQU   0           ; Temporal Rx Parity bit
```

```
; Include file for the 68HC908JL3, 68HC908JK3, 68HC908JK1
        Include 'jk3_registers.inc'

            ORG   RamStart

rSCCR       RMB   1           ; SCI Control Register
rSCSR1      RMB   1           ; SCI Status Register 1
rSCSR2      RMB   1           ; SCI Status Register 2
rSCRDRH     RMB   1           ; SCI Receive Data Register High
rSCRDRL     RMB   1           ; SCI Receive Data Register Low
rSCRSRH     RMB   1           ; SCI Receive Shift Register High
rSCRSRL     RMB   1           ; SCI Receive Shift Register Low
rSCTDRH     RMB   1           ; SCI Transmit Data Register High
rSCTDRL     RMB   1           ; SCI Transmit Data Register Low
rSCTSRH     RMB   1           ; SCI Transmit Shift Register High
rSCTSRL     RMB   1           ; SCI Transmit Shift Register Low

            ORG   RomStart

;*********************************************************************
;* Program goes here after reset                                     *
;*********************************************************************

Start:

            RSP               ; Reset Stack Pointer
            CLRX              ; Initialize MCU registers
            CLRH
            CLRA
            BSET  0,CONFIG1   ; Disable Watchdog Timer

; Example of configuration:
            MOV   #$0C,rSCCR  ; 8 bits, No PTY, 1 SB
                              ; Disable RX / TX byte subroutine

; Call this subroutine for SCI initial configuration
            JSR   RXTX_config ; Configure Channels for reception and
                              ; transmission. Call this function
                              ; any SCI operation.

; ENTER YOUR MAIN CODE HERE. This is an example of receiving 8 or
; 9 bits which is stored in the Index Registers H:X and then sending the
; same bits back

Again:
            JSR     GetByte   ; Wait for data reception and store
                              ; the received data in H:X registers
            JSR     PutByte   ; Store the H:X registers content to the
                              ; transmission register and wait for
                              ; the transmission to be completed
            BRA     Again
```

```
;*********************************************************************
;* This Subroutine configures the two channels:                      *
;* -> Stop and Reset the timer.                                      *
;* -> Input capture on falling edges with interrupts enabled, so that *
;* we can detect the start bit.                                      *
;* -> Disable channel and set initial value of that pin              *
;* (IDLE STATE)                                                      *
;* -> Set transmitter empty flag, to be prepared for first           *
;* transmission                                                      *
;* -> Port configuration                                             *
;* -> Clear status flags                                             *
;* -> Enable interrupts                                              *
;* -> Run timer                                                      *
;*********************************************************************

RXTX_config:
            LDA   #$30        ; stop and reset timer
            STA   TSC
            MOV   #$48,TSC0   ; IC, Falling edge, with interrupts
            MOV   #$00,TSC1   ; OC, output preset HIGH
            MOV   #$10,rSCSR1 ; Set Tx Empty Flag
            BSET  TPIN,PTD    ; Set pin for Idle State
            BSET  TPIN,DDRD
            BCLR  RPIN,DDRD
            CLR   rSCSR2
            CLI               ; Enable all interrupts
            BCLR  TSTOP,TSC   ; Enable timer counter
            RTS

;*********************************************************************
;* GetByte subroutine is a friendly usage of the SCI Reception       *
;* features, which provides Flags check and storage of the received  *
;* byte in the H:X registers. The User would modify this subroutine  *
;* if the received data needs to be stored in another register or    *
;* memory location.                                                  *
;*********************************************************************

GetByte:
            BRCLR   SCRF,rSCSR1,*
                              ; 1........ [3 CYCLES of instruction
                              ;            prior to interrupt]
                              ; Wait for byte to be received. This
                              ; flag is set when the received byte is
                              ; moved from the reception shift
                              ; register to the reception data
                              ; register.
            LDHX    rSCRDRH   ; Store received byte in the H:X
                              ; registers
            BCLR    SCRF,rSCSR1
                              ; Clear receiver full flag to allow
                              ; more receptions
            RTS
```

```
;*********************************************************************
;* PutByte subroutine is a friendly usage of the SCI Transmission     *
;* features, which provides Flags check and storage from the H:X      *
;* registers to the transmission register. User would modify this     *
;* subroutine if the byte to be sent is stored in another register or *
;* location.                                                          *
;*********************************************************************

PutByte:
            STHX    rSCTDRH   ; Store the H:X registers content in the
                              ; transmission shift register
            JSR     SCISend   ; Start byte transmission
            BRCLR   SCTE,rSCSR1,*
                              ; Wait for the byte to be transferred
                              ; to the transmission shift register
            RTS

;*********************************************************************
;* Program Goes here after a transmission if TIEN = 1 in the rSCCR    *
;*********************************************************************

SCITXEMPTY:
; ENTER YOUR SCITXEMPTY CODE HERE
            RTI

;*********************************************************************
;* Program Goes here after a Reception if RIEN = 1 in the rSCCR       *
;*********************************************************************

SCIRXFULL:
; ENTER YOUR SCIRXFULL CODE HERE
            RTI

;*********************************************************************
;* SCIsend.                                                          *
;* PutByte calls this subroutine.                                    *
;* If TEN = 0 (transmissions disabled) in the rSCCR this subroutine   *
;* returns with no modifications to the transmit channel.            *
;* if no transmission is in progress, this subroutine configures the  *
;* transmit channel and calculates the time to send the start bit.    *
;* The value stored in TCH1H:TCH1L represents a time dependent to the *
;* baud rate referenced as BITHI:BITLO.                              *
;*********************************************************************

SCISend:
            BRCLR TEN,rSCCR,SCIsend_end
                              ; Return from subroutine if TX is not
                              ; enabled
            BCLR  SCTE,rSCSR1 ; Clear transmitter empty flag
            BRSET CH1IE,TSC1,SCIsend_end
                              ; If transmission channel is active,
                              ; the byte to be transmitter is queued
                              ; and exits SCISend subroutine
            LDHX  rSCTDRH     ; Store data to be transmitted from
                              ; the transmission data register to
```

```
                        ; the transmission shift register
            STHX  rSCTSRH
            LDA   rSCCR       ; Mask the configuration flags in
                              ; the accumulator
            AND   #$12        ; Check M & PEN configuration bits to
                              ; see the data length
            CBEQA #$02,_07BIT ; If the user selected 8 data bits
                              ; with parity bit, the final number
                              ; of data bits is 7
            CBEQA #$10,_09BIT ; If the user selected 9 data bits
                              ; with no parity, the final number of
                              ; of bits is 9

; Set the bit at the left of the MSB depending
; on the Character Length Selection and Parity Enable bit (M & PEN)
; to know the end of the transmission.
; For example: for 8 data bits
; rSCTSRH = #%00000001
;                     \______ Will indicate end of Tx
; rSCTSRL = #%databits

            MOV   #$01,rSCTSRH
                              ; If the user selected 9 data bits
                              ; with parity, the final number of
                              ; data bits is 8
                              ; 08 BIT SELECTION
            BRA   conf_OC
_07BIT:
            CLR   rSCTSRH     ; 07 BIT SELECTION
            BSET  7,rSCTSRL   ; This bit is set to indicate the end
                              ; of data bits reception to the ISR.
                              ; This bit position depends on the
                              ; data bits length
            BRA   conf_OC
_09BIT:
            LDA   rSCTSRH     ; 09 BIT SELECTION
            AND   #$01        ; Set bit to indicate end of data
                              ; bits reception
            ORA   #$02
            STA   rSCTSRH
conf_OC:
            BCLR  CH1F,TSC1   ; Clear Channel Flag
            BSET  SCTE,rSCSR1 ; Set transmitter empty flag,
                              ; indicating that a new data to be
                              ; transmitted can be queued.
            LDHX  TCNTH       ; Read current count
            TXA
            ADD   #BITLO      ; Add 1 bit time for next compare
                              ; to send start bit
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
```

```
            BRCLR SB,rSCCR,storeinCH1
                              ; if there is a 2 stop bit
                              ; configuration, another bit time is
                              ; added to the current calculation
            TXA
            ADD   #BITLO      ; Add 1 more bit time
                              ; for next compare
                              ; to handle 2 stop bits
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
storeinCH1:
            STHX  TCH1H       ; Store the calculated time in the
                              ; channel timer register to generate
                              ; the output compare.
            LDA   rSCSR2
            AND   #$0D        ; Clear Transmitter Flags in rSCSR2
                              ; to start a new transmission. This is
                              ; done to reset the parity calculation
            STA   rSCSR2
            MOV   #$58,TSC1   ; config. channel 1 as OC
                              ; with interrupt enabled
                              ; Clear on Output compare to send the
                              ; start bit
SCIsend_end:
            RTS

;*********************************************************************
;* This ISR is dedicated only for transmission. each transmitted bit  *
;* generate this interrupt, excluding the second stop bit when        *
;* configued.                                                         *
;* Each transmission bit configures the polarity of output compare:   *
;* for example, if the next transmission bit is a logic '1', the      *
;* channel is configured as set on output compare after 1 bit time.   *
;* If the bit to be transmitted is a logic '1', the temporal parity is*
;* toggled,so at the end of the transmission if the temporal parity is*
;* 1 means that an odd number of logic '1' has been transmitted.      *
;* 0 means that an even number of logic '1' has been transmitted.     *
;*********************************************************************

TX_isr:

            PSHH
            BCLR  CH1F,TSC1
            BRSET TPF,rSCSR1,txinprog
                              ; Check if there's a transmission in
                              ; progress
            BSET  TPF,rSCSR1  ; Program goes here if no transmission
                              ; is in progress
            LDHX  TCH1H       ; Calculate the time for next bit to
                              ; be transmitted. Start bit was already
                              ; sent by the SCISend subroutine
```

```
            TXA
            ADD   #BITLO      ; Set time for next OC
                              ; after 1 bit time
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
            STHX  TCH1H       ; Store the calculated time in the
                              ; channel register for next output
                              ; compare
            CLC
            ROR   rSCTSRH     ; Copy next transmission
            ROR   rSCTSRL     ; bit into carry flag using the two
                              ; shift registers
oc_highorlow:
            BCS   oc_high
            MOV   #$58,TSC1   ; If carry cleared, configure the
                              ; channel as clear output on next
                              ; compare
            PULH
            RTI               ; return from interrupt
oc_high:
            LDA   rSCSR2      ; Program goes here if the bit to be
                              ; transmitted is a logic "1", and the
                              ; parity is recalculated in the
                              ; transmit temporal parity bit TPTY
            EOR   #$02
            STA   rSCSR2
            MOV   #$5C,TSC1   ; If carry set, configure channel as
                              ; set output on next compare
            PULH
            RTI               ; return from interrupt
txinprog:
            BRSET PTYTX,rSCSR2,sendingPTY
                              ; Check if sending parity
            BRSET STBTX,rSCSR2,sendingSTB
                              ; Check if sending stop bits
            CLC               ; transmitting data bits
            ROR   rSCTSRH
            ROR   rSCTSRL
            TPA               ; Copy CCR into A
                              ; If transmit data register
                              ; zero, data transmission
                              ; is done
            LDHX  rSCTSRH
            CPHX  #$0000
            BEQ   txfinished  ; Exit from interrupt
nextbittime:
            PSHA              ; send next bit depending on carry
            LDHX  TCH1H
            TXA
            ADD   #BITLO      ; Add 1 bit time for next output
                              ; compare
```

```
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
            STHX  TCH1H
            PULA
            TAP
            BRA   oc_highorlow
                              ; Branch to calc. next OC polarity
txfinished:
            BRCLR PEN,rSCCR,sendingPTY
                              ; If Pty. disabled, send stop bit
            BSET  PTYTX,rSCSR2
                              ; Indicate Parity Transfer
            BCLR  STBTX,rSCSR2
            LDA   rSCCR
            AND   #$01
            LSLA
            EOR   rSCSR2      ; Calculate PTY level depending on the
                              ; configuration of the parity: Even or
                              ; Odd, and the temporal parity
                              ; calculated from the data being sent
            COMA
            LSRA
            LSRA              ; Store Parity level
                              ; into carry flag
            TPA
            BRA   nextbittime ; Calc. next OC time
sendingPTY:
            BSET  STBTX,rSCSR2
                              ; Indicate sending stop bits
            BCLR  PTYTX,rSCSR2
            SEC
            TPA
            BRA   nextbittime ; Calculate next output compare time
sendingSTB:
            BCLR  TPF,rSCSR1  ; Indicate that transmission is done
            BRSET SCTE,rSCSR1,NoTxPending
                              ; If there is a transmission pending
                              ; this interrupt performs the same
                              ; operation as the SCIsend subroutine
            LDHX  rSCTDRH     ; The data to be transmitted is moved
                              ; from the transmit data register to
                              ; the transmit shift register
            STHX  rSCTSRH
            LDA   rSCCR       ; The number of data bits depends on
                              ; the number of bits configured and
                              ; the parity bit configuration:
                              ; M=0 & PEN=0 -> 8 data bits. No Parity
                              ; M=0 & PEN=1 -> 7 data bits. Parity
                              ; M=1 & PEN=0 -> 9 data bits. No Parity
                              ; M=1 & PEN=1 -> 8 data bits. Parity
            AND   #$12        ; Check M & PEN configuration bits
```

```
            CBEQA #$02,__07BIT
            CBEQA #$10,__09BIT

; Set the bit at the left of the MSB depending
; on the Character Length Selection and Parity Enable bit (M & PEN)
; to know the end of the transmission.
; For example for 8 data bits and no parity enabled:
; rSCTSRH = #%00000001
;                     \______ Will indicate end of Tx
; rSCTSRL = #%databits

            MOV   #$01,rSCTSRH
                              ; 08 BIT SELECTION
            BRA   _conf_OC
__07BIT:
            CLR   rSCTSRH     ; 07 BIT SELECTION
            BSET  7,rSCTSRL
            BRA   _conf_OC
__09BIT:
            LDA   rSCTSRH     ; 09 BIT SELECTION
            AND   #$01
            ORA   #$02
            STA   rSCTSRH
_conf_OC:
            BCLR  CH1F,TSC1   ; Clear Channel Flag
            BSET  SCTE,rSCSR1 ; Set transmitter empty flag to allow
                              ; another data to be queued
            LDHX  TCH1H       ; Read current channel value
            TXA
            ADD   #BITLO      ; Add 1 bit time for next compare
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
            BRCLR SB,rSCCR,_storeinCH1
            TXA
            ADD   #BITLO      ; Add 1 more bit time
                              ; for next compare
                              ; to handle 2 stop bits of the current
                              ; data being sent
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
_storeinCH1:
            STHX  TCH1H       ; Store calculated data into the
                              ; channel register
            LDA   rSCSR2      ; Reset Flags for next transmission
            AND   #$0D        ; Clear TX Flags in rSCSR2
            STA   rSCSR2
            MOV   #$58,TSC1   ; config. channel 1 as OC
```

```
                            ; with interrupt enabled
                            ; Clear pin on output compare to send
                            ; the start bit
            BRA   TxPending ; If a transmission is pending, the
                            ; channel is not disabled

NoTxPending:
            CLR   TSC1      ; Disable tx channel
TxPending:
            PULH
            BRCLR TIEN,rSCCR,EXITTX
                            ; If enabled, jump to tx done
            JMP   SCITXEMPTY ; subroutine
EXITTX:
            RTI             ; Else exit

;*********************************************************************
;* This ISR is dedicated for reception.                              *
;* When the receiving line is in idle state, this channel is         *
;* configured as input capture on falling edge, waiting for a start  *
;* bit. When the start bit is received, the channel is configured as *
;* output compare with pins under port control, since the output     *
;* compare will be used only as a timing reference for the bits      *
;* reception. In the first data bit reception, the time added to the *
;* reception channel is 1.3 bit time minus 31 of pin check latency,  *
;* so the instruction that check the pin state does not check the    *
;* state in the bit time boundary. This latency is measured from the *
;* beginning of the ISR to the instruction that checks the pin state.*
;* In each consecutive output compare the pin state is read and      *
;* shifted into the reception shift register. When all the data bits *
;* are received, the parity is received if enabled and one or two    *
;* stop bits are received. The channel is again configured as input  *
;* capture on falling edges to detect the next start bit             *
;*********************************************************************

RX_isr:                     ; 2........ [9 CYCLES Interrupt Entrance]

            BCLR  CH0F,TSC0 ; 3........ [4 CYCLES]
                            ; acknowledge interrupt
            BRSET REN,rSCCR,RxNoExit
                            ; 4........ [5 CYCLES]
            JMP   RXEXIT    ; Exits from interrupt if receptions
                            ; are not enabled
RxNoExit:
            PSHH            ; 5........ [2 CYCLES]
                            ; Program goes here if receptions are
                            ; enabled
            BRSET RPF,rSCSR1,rxinprog
                            ; 6........ [5 CYCLES]
                            ; check if Rx in progress
            BSET  RPF,rSCSR1 ; set reception in progress flag
                            ; if there is no reception in progress
            LDA   rSCSR2
            AND   #$32      ; Clear RX Flags in rSCSR2 for the new
                            ; reception
```

```
            STA   rSCSR2
            LDA   rSCCR       ; The number of data bits depends on
                              ; the number of bits configured and
                              ; the parity bit configuration:
                              ; M=0 & PEN=0 -> 8 data bits. No Parity
                              ; M=0 & PEN=1 -> 7 data bits. Parity
                              ; M=1 & PEN=0 -> 9 data bits. No Parity
                              ; M=1 & PEN=1 -> 8 data bits. Parity
            AND   #$12        ; Check M & PEN configuration bits
            CBEQA #$02,___07BIT
            CBEQA #$10,___09BIT

; For each character length are set two bits:
; one will indicate the end of the reception
; and the other is going to be used to accommodate the
; data received.
; For example, for 9 bit selection in M:
; rSCRDRH:rSCRDRL -> 10000001 00000000
;                     \      \________ Will indicate Rx done
;                      \_______________ Used to adjust Rx data

            LDHX  #$8080      ; 08 BIT SELECTION
            BRA   confCH0
___07BIT:
            LDHX  #$8040      ; 07 BIT SELECTION
            BRA   confCH0
___09BIT:
            LDHX  #$8100      ; 09 BIT SELECTION
confCH0:
            STHX  rSCRSRH
            LDHX  TCH0H       ; Here is the time at which the start
            TXA               ; bit was received
            ADD   #BIT1LO     ; Add 1.3 bit time - 31 for next OC
                              ; If no other interrupt is present on
                              ; the user software, each data bit is
                              ; going to be received in 30 % of the
                              ; bit time
            TAX
            PSHH
            PULA
            ADC   #BIT1HI
            PSHA
            PULH
            STHX  TCH0H
            MOV   #$50,TSC0   ; config. channel 0 as output
                              ; compare with int enabled
                              ; Pin under port control. Each output
                              ; compare interrupt will give the time
                              ; to check the pin state, that is why
                              ; the channel is configured as pin
                              ; under port control
            PULH
            RTI               ; Return from interrupt
rxinprog:
            CLC               ; 7........ [1 CYCLES]
```

```
            BRCLR RPIN,PTD,nocarry
                              ; 8........ [2 CYCLES from instruction
                              ;            fetch to the pin reading]
                              ; PIN CHECK LATENCY IS:
                              ; 3 + 9 + 4 + 5 + 2 + 5 + 1 + 2 = 31
                              ; Cycles of pin check latency
                              ; Copy pin state into carry. The pin
                              ; check latency is measured from the
                              ; beginning of the interrupt to this
                              ; instruction
            SEC
nocarry:
            TPA               ; Some operations are going to be done
                              ; with the accumulator for fast
                              ; execution, thus the CCR register is
                              ; pushed onto the stack
            PSHA
            LDA   rSCSR2      ; The software branches depending on
                              ; PTYRX and STBRX status flags. These
                              ; flags indicate the state in which the
                              ; program is in.
            AND   #%00001100
            CBEQA #%00001000,ptyrec
                              ; Branch if receiving parity
                              ; This bit is received to check the
                              ; parity error condition
            CBEQA #%00000100,sb1rec
                              ; Branch if receiving 1st SB
                              ; Both stop bits are received to check
                              ; the overrun error condition
            CBEQA #%00001100,sb2rec
                              ; Branch if receiving 2nd SB

; Program goes here if receiving data bits

            BCC   notogPTY
            LDA   rSCSR2
            EOR   #$01        ; Toggle temporal Rx Parity if
                              ; the received bit is a logic '1'
            STA   rSCSR2      ; Store new temp Rx parity RPTY flag
notogPTY:
            PULA
            TAP               ; Pop flags. Rotate reception shift
                              ; register with carry. Carry has the
                              ; logic value of the received bit
            ROR   rSCRSRH     ; With carry set or cleared, rotate
            ROR   rSCRSRL     ; memory if carry set after rotation
            BCC   nextRXtime  ; means that next Rx bit is the Pty
            BCLR  FE,rSCSR1   ; Init Framing error flag
            BSET  PTYRX,rSCSR2
                              ; Indicate next reception is the
                              ; parity bit
            BCLR  STBRX,rSCSR2
nextRXtime:
            LDHX  TCH0H       ; Load current channel value
```

```
            TXA
            ADD   #BITLO      ; Add 1 bit time
            TAX
            PSHH
            PULA
            ADC   #BITHI
            PSHA
            PULH
            STHX  TCH0H       ; Store new calculated time into
                              ; channel registers
            MOV   #$50,TSC0   ; config. channel 0 as output
            PULH              ; compare with int enabled
                              ; Pin under port control.
            RTI               ; Return from interrupt
ptyrec:
            BRCLR PEN,rSCCR,sb1rec
                              ; If Parity disabled, branch
                              ; to 1st SB reception
            PULA              ; Program goes here if the parity bit
                              ; is being received. CCR is poped from
                              ; stack and stored in the accumulator
            EOR   rSCSR2      ; Calculate Parity Error
            EOR   rSCCR
            AND   #$01
            BCLR  PE,rSCSR1
            ORA   rSCSR1      ; (A.0 = 1)? -> PE=1
            STA   rSCSR1      ; ELSE -> PE=0
            BCLR  PTYRX,rSCSR2
                              ; Indicate next reception
            BSET  STBRX,rSCSR2
                              ; to be 1st Stop Bit
            BRA   nextRXtime  ; Branch to calculate next output
                              ; compare time
sb1rec:
            BRCLR SB,rSCCR,sb2rec
                              ; Branch to 2nd Stop Bit reception if
                              ; SB=0
            PULA              ; Pop CCR and store it in the acc.
            LSLA
            EOR   #$02        ; Calculate Framing Error
            AND   #$02        ; flag
            BCLR  FE,rSCSR1
            ORA   rSCSR1      ; (A.1 = 1)? -> FE=1
            STA   rSCSR1      ; ELSE -> FE=0
            BSET  PTYRX,rSCSR2
            BSET  STBRX,rSCSR2
                              ; Indicate that the next received bit
                              ; is the second stop bit
            BRA   nextRXtime  ; Calculate next output compare time
sb2rec:
            PULA
            BRSET FE,rSCSR1,rxfinished
                              ; If FE=1, rx done
            LSLA
            EOR   #$02        ; Calculate Framing Error
```

```
            AND   #$02        ; flag
            BCLR  FE,rSCSR1
            ORA   rSCSR1      ; (A.1 = 1)? -> FE=1
            STA   rSCSR1      ; ELSE -> FE=0
                              ; The same formula for framing error is
                              ; computed for the second stop bit
rxfinished:
            LDA   rSCRSRH     ; Prepare received data for the adjust
                              ; operation
            LDX   rSCRSRL
            CLC
adjustdata:
            RORA
            RORX              ; Accommodate data. Rotate right until
                              ; the rotated bit is a logic '1', which
                              ; is the bit that was set at the
                              ; beginning of the reception
            BCC   adjustdata
            BCLR  ORE,rSCSR1
            BRCLR SCRF,rSCSR1,noORerror
            BSET  ORE,rSCSR1  ; If the previously received data has
                              ; not been read, the Overrun Error ORE
                              ; is set
noORerror:
            STA   rSCRDRH     ; Store adjusted data into
            STX   rSCRDRL     ; Receive Data Registers
            BCLR  RPF,rSCSR1  ; Indicate that there's a
            BSET  SCRF,rSCSR1 ; valid data on rSCRDRH:rSCRDRL
            MOV   #$48,TSC0   ; IC, Falling edge, with
            PULH              ; interrupt enabled
            BRCLR RIEN,rSCCR,RXEXIT
                              ; If enabled, jump to Rx done
            JMP   SCIRXFULL   ; Subroutine
RXEXIT:
            RTI               ; Else Exit
```

```
;*********************************************************************
;* Dummy ISR                                                         *
;*********************************************************************

dummy_isr:
            BRA   dummy_isr
            RTI               ; return

;*********************************************************************
;* Vector definitions                                                *
;*********************************************************************

            ORG   VectorStart
            FDB   dummy_isr   ; ADC Conversion Complete Vector
            FDB   dummy_isr   ; Keyboard Vector
            FDB   dummy_isr   ; (No Vector Assigned $FFE2-$FFE3)
            FDB   dummy_isr   ; (No Vector Assigned $FFE4-$FFE5)
            FDB   dummy_isr   ; (No Vector Assigned $FFE6-$FFE7)
            FDB   dummy_isr   ; (No Vector Assigned $FFE8-$FFE9)
            FDB   dummy_isr   ; (No Vector Assigned $FFEA-$FFEB)
            FDB   dummy_isr   ; (No Vector Assigned $FFEC-$FFED)
            FDB   dummy_isr   ; (No Vector Assigned $FFEE-$FFEF)
            FDB   dummy_isr   ; (No Vector Assigned $FFF0-$FFF1)
            FDB   dummy_isr   ; TIM1 Overflow Vector
            FDB   TX_isr      ; TIM1 Channel 1 Vector
            FDB   RX_isr      ; TIM1 Channel 0 Vector
            FDB   dummy_isr   ; (No Vector Assigned $FFF8-$FFF9)
            FDB   dummy_isr   ; ~IRQ1
            FDB   dummy_isr   ; SWI Vector
            FDB   Start       ; Reset Vector
```

Conclusion

In this application note, two modes of software SCI are described and coded. One is for a simple asynchronous communication protocol, where no data length selection is available and no errors are detected. The other implementation of the SCI is for a multiple format with error detection asynchronous protocol. The user has two optimized implementation of an SCI without adding any external UART.

NOTE: *With the exception of mask set errata documents, if any other Motorola document contains information that conflicts with the information in the device data sheet, the data sheet should be considered to have the most current and correct data.*

References

MC68HC908QY4/D: *MC68HC908QY4, MC68HC908QT4, MC68HC908QY2, MC68HC908QT2, MC68HC908QY1, MC68HC908QT1, Data Sheet*

MC68HC908JL3/H: *MC68HC908JK1, MC68HC908JK3, MC68HC908JL3 Data Sheet*

HC908JL3AD/D: *Addendum to MC68HC908JL3/H*

AN1240/D: *HC05 MCU Software-Driven Asynchronous Serial Communication Techniques Using the MC68HC705J1A*

AN1818/D: *Software SCI Routines with the 16-Bit Timer Module*

AN2637/D: *Software SCI for the MC68HC908QT/QY MCU*

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:
Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

JAPAN:
Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

ASIA/PACIFIC:
Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

HOME PAGE:
http://motorola.com/semiconductors

Freescale Semiconductor, Inc.