Freescale Semiconductor
Application Note

AN1818
Rev. 1, 11/2004

Software SCI Routines with the 16-Bit Timer Module

by: Brad Bierschenk
MMD Applications Engineering
Austin, Texas

Introduction

Many applications that communicate to off-board devices require an asynchronous serial link. A Freescale microcontroller unit (MCU) with a serial communications interface (SCI) module can provide this communications functionality.

However, in many applications, an MCU that does not have an SCI module must be used. If asynchronous communications capability is needed, it must be provided through software control of existing modules. A bit-banged approach, as documented in HC05 MCU Software-Driven Asynchronous Serial Communication Techniques Using the MC68HC705J1A (Freescale document order number AN1240), is convenient, but requires dedicated software overhead while transmitting and receiving data.

Through the use of the 16-bit free-running counter, the HC05 and other MCU families can provide an interrupt-driven software SCI with minimal software overhead.

General Information

The solution discussed here works in half-duplex mode. This means it can transmit or receive serial data, but cannot simultaneously transmit and receive. This is enough for most applications and is much easier to implement than a full-duplex solution.

The timing in Figure 1 shows the standard non-return-to-zero (NRZ) asynchronous transmission protocol of an RS-232 serial transfer.

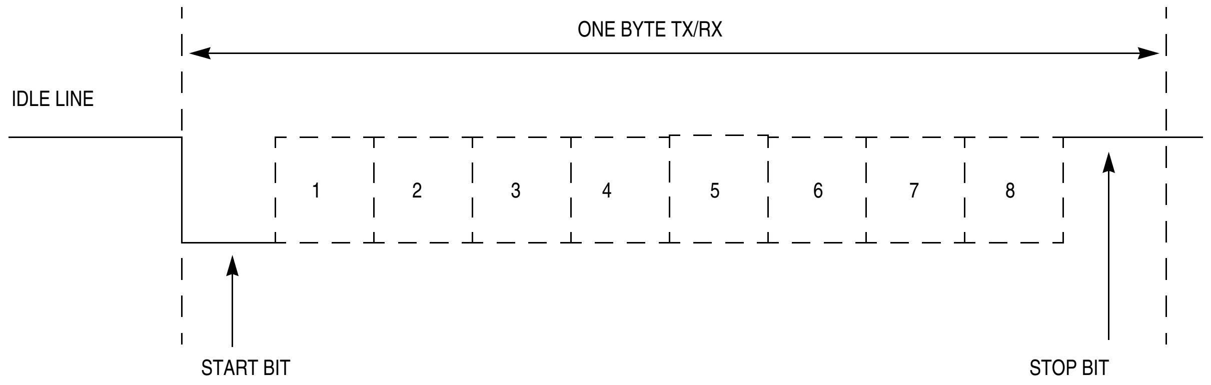

Figure 1. Serial RS-232 Timing

A complete byte transfer takes 10 bit times due to the start and stop bits. The first falling edge indicates the beginning of the start bit, and thus the beginning of a byte transmission. After the start bit, data is sent in eight bits. The logic-high stop bit signals the end of the byte transmission.

A 16-bit free-running timer counter with one input capture (IC), one output compare (OC), and the associated interrupts, allows software emulation of an SCI module with only a small amount of processor overhead (see Figure 2). In addition to the timer module, one digital input pin that can be sampled using BRSET or BRCLR instructions is needed.

On some MCUs, including the MC68HC705P6A, the input capture pin can be read directly as a digital input. On other MCUs, the input capture pin also should be connected to a digital input pin to allow digital polling.

A byte variable in RAM can be used to simulate the flags of an SCI status and control register; likewise, a RAM variable can function as a data register where transmitted and received bytes are stored (see Figure 3).

RX	TX	RDRF	TDRE	X	X	X	X
7	6	5	4	3	2	1	0

Figure 2. Simulated Status Register in RAM Variable

RX — Receive In-Progress Flag

A 1 here signifies that a receive is in progress.

TX — Transmit In-Progress Flag

A 0 here indicates a transmit is in progress.

RDRF — Receive Data Register Full

A 1 here indicates that a byte has been received.

TDRE — Transmit Data Register Empty

A 1 here indicates that a byte has been transmitted.

D7	D6	D5	D4	D3	D2	D1	D0
7	6	5	4	3	2	1	0

Figure 3. Simulated Data Register in RAM Variable

Receiving Serial Data

In this application, if data is not being transmitted, the input capture (IC) function of the timer is enabled. In this way, the user can wait for the start bit of an incoming transmission without any software overhead. When the start bit is received, the IC interrupt is triggered. This provides both a wakeup to start receiving and the start of a timing reference via the value in the IC registers (see Figure 4).

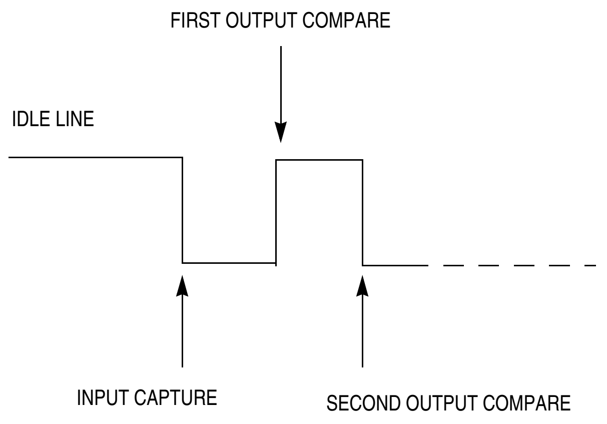

Figure 4. Receiving with the Timer Functions

The contents of the IC registers show the time of the falling edge of the start bit. The resulting timer interrupt routine has to determine which event (IC or OC) triggered the interrupt. In the first entry, one and a half bit times are added to the content of the capture register. The result is stored in the OC registers and interrupts are switched from the IC to the OC. The delay of one and a half bit times will cause an output compare event approximately in the middle of the first data bits reception.

Next, the data register is cleared, and bit 7 of the data register is set. This most significant bit (MSB) of the data variable acts as a bit counter. In the next output compare, the data at the pin (either TCAP or port pin) is sampled using a BRSET instruction. This brings the value of the data received into the carry bit of the condition code register (CCR). The rotate right through carry (ROR) instruction rotates the new data bit into the data register. It is rotated into the data register and one bit time is added to the OC register.

Because the data register was cleared prior to reception, and bit 7 was set, a 0 is always rotated into the carry bit until the eighth data bit is received. The setting of the carry bit after a rotate indicates that the eighth bit has been received. When this happens, the receive data full flag is set and the interrupt capability is switched back to input capture.

Transmitting

To transmit a byte, a mechanism is needed that can trigger at a given rate and allow changing of the bit level of an output. The OC function of the 16-bit timer module allows this (see Figure 5).

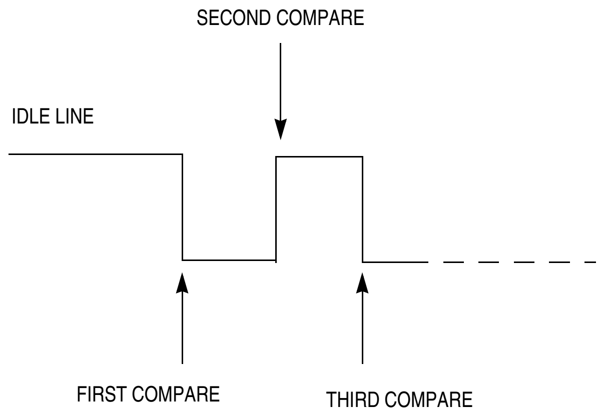

Figure 5. Transmitting with the Output Compare Function

The routine SCISend in the software listing provides the transmit function. Before calling SCISend, the user places the byte to be transmitted into the SCIData location. Transmission starts by setting the I bit in the condition code register (CCR) to ensure proper timing, and reading the contents of the free-running counter. An offset is then added to that value, and the result is stored into the output compare registers. This defines the time the transmission will begin. The OLVL bit is set to 0 to produce the required falling edge for the start bit at the time of the next compare. The OC interrupt is enabled, and the user can now wait for the predefined OC event to drive the TCMP pin low to start the transmission.

When running through the timer interrupt service routine, distinguishing between an IC or an OC event (they both use the same interrupt) is a must. In this way, the user can arbitrate between the beginning of a byte reception and a reception/transmission in progress.

Just as with the receiving code, the transmission of a byte uses the propagation of a logic 1 from the carry to provide a bit counter. When all bits have been transmitted, a logic 1 will be rotated into the carry bit, and OC can be set up to transmit the logic high stop bit.

Baud Rates

To change the baud rate, adjust the values of BITHI and BITLO to represent one bit time at the frequency of the timer module. Likewise, BIT1HI and BIT1LO should be changed to represent one and a half bit times at the frequency of the timer module.

The internal frequency of operation and the latency of the timer interrupt define the maximum baud rate that can be achieved. The rate of the timer interrupts should not be programmed to be faster than the latency of the interrupt service routine. If this happens, one might miss OC or IC events (see Figure 6).

The frequency of the 16-bit timer counter is four times slower than the internal operating frequency. The formula to determine what number to add to the timer value to cause a specific delay is:

$$f_{Bus} \div [(\text{baud rate}) \times 4]$$

For example:

Internal Frequency	Timer Frequency	9600 Baud	4800 Baud	2400 Baud	1200 Baud
2 MHz	500 kHz	$0034	$0068	$00D0	$01A0

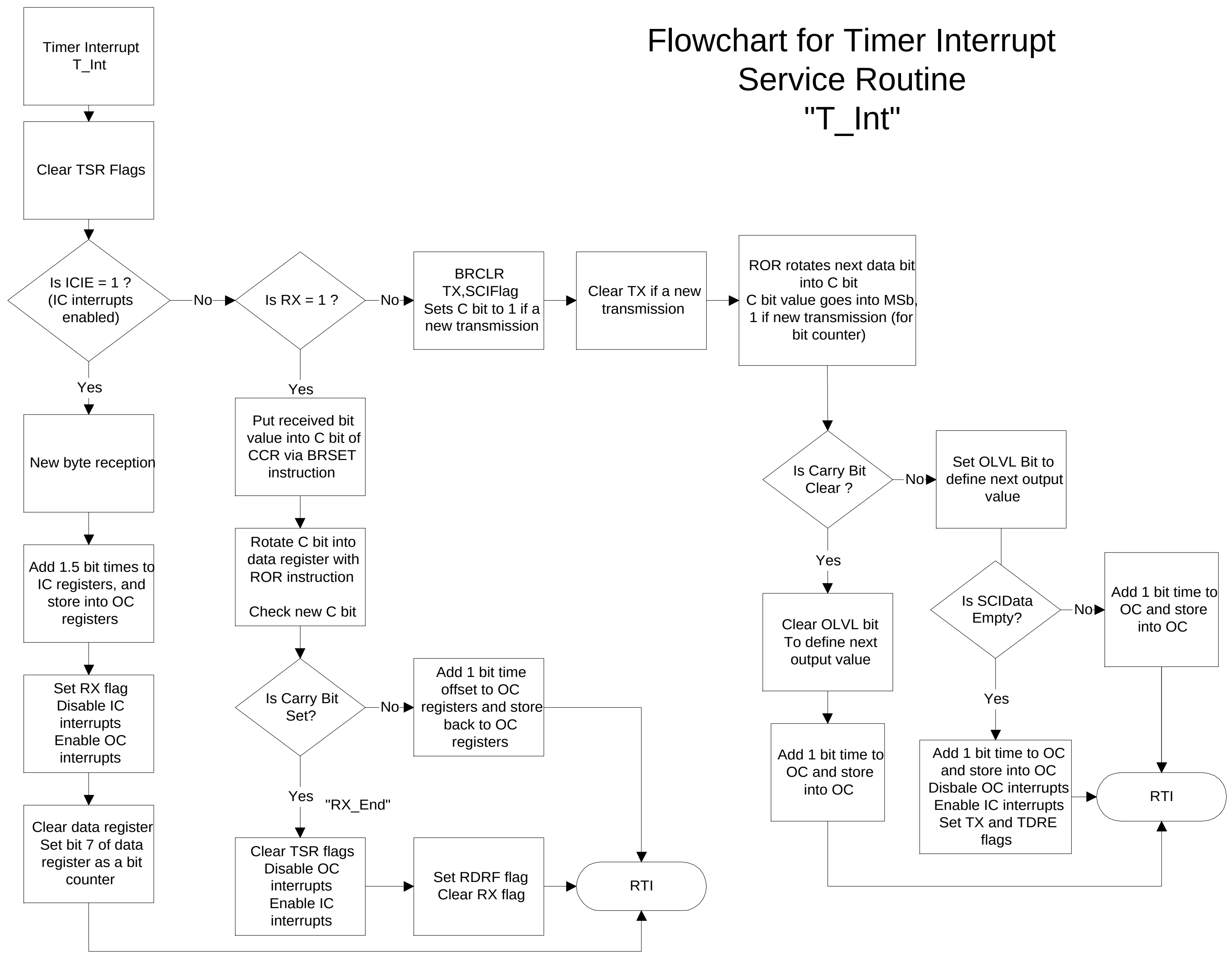
Flowchart for Timer Interrupt Service Routine "T_Int"
Timer Interrupt T_Int
Clear TSR Flags
Is ICIE = 1 ? (IC interrupts enabled)
No
Is RX = 1 ?
No
BRCLR TX,SCIFlag Sets C bit to 1 if a new transmission
Clear TX if a new transmission
ROR rotates next data bit into C bit C bit value goes into MSb, 1 if new transmission (for bit counter)
Yes
New byte reception
Add 1.5 bit times to IC registers, and store into OC registers
Set RX flag Disable IC interrupts Enable OC interrupts
Clear data register Set bit 7 of data register as a bit counter
Yes
Put received bit value into C bit of CCR via BRSET instruction
Rotate C bit into data register with ROR instruction
Check new C bit
Is Carry Bit Set?
No
Add 1 bit time offset to OC registers and store back to OC registers
Yes
"RX_End"
Clear TSR flags Disable OC interrupts Enable IC interrupts
Set RDRF flag Clear RX flag
RTI
Is Carry Bit Clear ?
No
Set OLVL Bit to define next output value
Yes
Clear OLVL bit To define next output value
Add 1 bit time to OC and store into OC
Is SCIData Empty?
No
Add 1 bit time to OC and store into OC
Yes
Add 1 bit time to OC and store into OC Disbale OC interrupts Enable IC interrupts Set TX and TDRE flags
RTI

Figure 6. Flowchart for Timer Interrupt Service

Software Example

The code listing that follows illustrates reading and writing serial data through the timer interface. This simple software loop waits for data to be received and echoes the value back to the sending device.

Code Listing

```
* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
* SWSCI.ASM
* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
* A software-driven SCI simulation for the 705P6A MCU,
* using the timer's input capture and output compare
* functions.
*
* Brad Bierschenk, MMD Applications Engineering
* Oak Hill, Austin, Texas
* 08/06/99
* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
* NOTES:
* a) The "SCI" subroutine sets up the transmit routine
*    so to send a byte, you have to load it into SCI data
*    variable, and JSR to SCI
* b) The "simulated" SCI status and data register are held
*    in RAM, and the "simulated" SCI interrupt is really the
*    timer interrupt.
* c) Limitation is half-duplex only.
* d) To transmit, use the SCI routine. But you will not
*    be able to receive until the transmission is complete.
* e) This requires a part that can digitally read its
*    TCAP pin (P6A). Otherwise, a separate input pin should
*    be tied to the TCAP pin for polling.
* 4) The P6A REQUIRES a pullup on TCAP to VDD for this
*    application.
* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
* -----------------------------------------------------------------
* Needed P6A bits and bytes
* -----------------------------------------------------------------
RAMSPACE   EQU   $0050
ROMSPACE   EQU   $0100

PORTB      EQU   $01
PORTC      EQU   $02
PORTD      EQU   $03
DDRB       EQU   $05
DDRC       EQU   $06
DDRD       EQU   $07
TCR        EQU   $12
TSR        EQU   $13
IC1HI      EQU   $14
IC1LO      EQU   $15
OC1HI      EQU   $16
OC1LO      EQU   $17
TCNTHI     EQU   $18
TCNTLO     EQU   $19
OLVL       EQU   0
```

```
IEDG        EQU   1
OCF         EQU   6
ICF         EQU   7
OCIE        EQU   6
ICIE        EQU   7

* Software SCI equates for RAM variable SCIFlag
TDRE        EQU   4
RDRF        EQU   5
TX          EQU   6
RX          EQU   7

;BIT1HI+BIT1LO define the timer delay for 1.5 bit times at given
;baud rate.
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
;9600       baud
BITHI       EQU   $00
BITLO       EQU   $34
BIT1HI      EQU   $00
BIT1LO      EQU   $48

;4800       baud
;BITHI      EQU   $00
;BITLO      EQU   $68
;BIT1HI     EQU   $00
;BIT1LO     EQU   $9C

;2400       baud
;BITHI      EQU   $00
;BITLO      EQU   $D0
;BIT1HI     EQU   $01
;BIT1LO     EQU   $38

;1200       baud
;BITHI      EQU   $01
;BITLO      EQU   $A0
;BIT1HI     EQU   $02
;BIT1LO     EQU   $70

* ------------------------------------------------------------
* RAM Variables
* ------------------------------------------------------------
           ORG    RAMSPACE
SCIFlag    RMB    1                ;Simulated Status register
SCIData    RMB    1                ;Simulated Data register
* -------------------------------------------------------------
* Start of program code
* -------------------------------------------------------------
           ORG    ROMSPACE
Begin      LDA    #$10
           STA    PORTB            ;Set OC pin to high ==> idle line
           LDA    #$F7
           STA    DDRB

           CLR    SCIFlag          ;Clear SCI status register
           CLR    SCIData          ;Clear SCI data register
           LDA    TSR              ;Clear possibly set OC & IC flags
           LDA    IC1LO
           LDA    OC1LO
```

```
          ;Initialize timer system to OCLevel High (idle)
          ;IC falling edge (detect start bit), disable OCI
          ;enable ICI (SCI ready to receive)
          LDA    #$81
          STA    TCR

          BSET TX,SCIFlag        ;Clear first-entry-to-transmit
                                 ;flag
          CLI                    ;Globally enable interrupts

Main      BRCLR  RDRF,SCIFlag,*  ;Wait for a byte to be received

          ;Allow ~2 bit times for rest of last bit and stop bit
                                 ;~210 µs ~= 55 cycles
          LDA    #$09            ;2
DelayLoop DECA                   ;3
          BNE     DelayLoop      ;3

                                 ;Echo back the received byte...
          BCLR   RDRF,SCIFlag
          JSR    SCISend

                                 ;Wait for next received byte
          BRA    Main

* ---------------------------------------------------------------
* SCISend sets up the timer module to transmit a byte.
* Uses the OC function to transmit data. Can't receive
* while transmitting (limitation is half-duplex)
* --------------------------------------------------------------
SCISend   SEI                    ;Disable interrupts to ensure
                                 ;timing
          LDX    TCNTHI          ;Read current timer value
          LDA    TCNTLO
          ADD    #$15            ;Add offset
          STA    OC1LO           ;Store new value
          TXA
          ADC    #$00            ;Accommodate carry if needed
          STA    OC1HI
          LDA    TSR
          LDA    OC1LO
          STA    OC1LO
          LDA    #%01000000      ;Generate start bit by setting OLVL
          STA    TCR             ;bit to falling edge, disable ICI,
                                 ;enable OCI
          CLI                    ;Globally enable interrupts again
          RTS

* ---------------------------------------------------
* T_Int is the timer interrupt service routine.
* Must arbitrate whether an IC or OC caused the interrupt,
* to determine whether receiving or transmitting a byte.
* (Timer interrupt ~= SCI Interrupt)
* OC event is either 1) byte transmitting or 2) sampling
* byte being received.
* IC event is the start bit of a received byte
* -------------------------------------------------------------
T_Int    LDA     TSR         ;Clear any flags
```

```
   ;If IC interrupts are enabled, we are in receive mode
   ;and have received start bit on TCAP BRSET ICIE,TCR,Receive

   ;If OC interrupts are enabled, we are either
   ;transmitting a byte, or are sampling a byte coming in
   BRSET        RX,SCIFlag,RX1
   ;Is SCI receiving?

   ;Is this a byte transmitalready-in-progress?
   ;The BRCLR instruction sets the carry bit to the value
   ;of the bit being tested.
   BRCLR        TX,SCIFlag,TX1

   ;New transmission
   ;Carry bit gets set, clear the flag to indicate
   ;transmit-in-progress.
   ;C = 1 will be rotated into bit 7 of data register
   ;for use as a bit counter.
   BCLR         TX,SCIFlag

         ;Transmitting
TX1      ROR     SCIData         ;Shift next data bit into carry
         BCC     TX2             ;If low, go to TX2
         BSET    OLVL,TCR        ;If high, next OC level to high
         ;If Data register is zero, and Carry is set, we have
         ;just rotated out the last bit, and need to send the
         ;stop bit.
         BEQ     TX_End          ;If stop bit, go to TX_End
         LDA     OC1LO           ;Otherwise, add bit time to OC
         ADD     #BITLO          ;for the next bit
         TAX
         LDA     OC1HI
         ADC     #BITHI
         STA     OC1HI
         STX     OC1LO
         RTI

TX2      BCLR    OLVL,TCR        ;Carry was low means next data bit
                                 ;low
                                 ;so next OC level to low
         LDA     OC1LO           ;Add bit time to OC
         ADD     #BITLO
         TAX
         LDA     OC1HI
         ADC     #BITHI
         STA     OC1HI
         STX     OC1LO
         RTI

TX_End   LDA        OC1LO        ;Add last bit time to OC for the
                                 ;stop bit
ADD      #BITLO
         TAX
         LDA        OC1HI
         ADC        #BITHI
         STA        OC1HI
         STX        OC1LO
         LDA        TSR
         LDA        IC1LO
         LDA        #$81
         STA        TCR          ;Disable OCI, enable ICI
         ;Clear first TX entry flag again,
         ;and set the TDRE bit. NOTE that even though
         ;the TDRE bit is set, the TX of the data byte
         ;is not complete, with the rest of the last bit
         ;and the stop bit to be transmitted
         LDA         #$50
         STA         SCIFlag
         RTI

Receive  LDA         IC1LO        ;Start bit has been received
         ADD         #BIT1LO      ;add 1+1/2 bit times
         TAX                      ;to OC for the first bit sampling
         LDA         IC1HI
         ADC         #BIT1HI
         STA         OC1HI
         LDA         TSR
         STX         OC1LO
         BSET        RX,SCIFlag   ;Set receive-in-progress flag
         LDA         #$41         ;disable ICI, enable OCI
         STA         TCR
         LDA         #$80         ;Clear data register, set bit 7 as
         STA         SCIData      ;a bit counter
         RTI

RX1      BRSET       7,PORTD,RX2  ;get bit level from TCAP pin and
RX2      ROR         SCIData      ;put it into data variable
         BCS         RX_End       ;End if it is the last bit
         LDA         OC1LO        ;If not add bit time
         ADD         #BITLO       ;for next sample
         TAX
         LDA         OC1HI
         ADC         #BITHI
         STA         OC1HI
         STX         OC1LO
         RTI

RX_End   LDA          TSR         ;Byte received, clear possibly set
                                  ;IC flag
         LDA          IC1LO
         LDA          #$81        ;Disable OCI, enable ICI
         STA          TCR
         ;Set receive register full flag in RAM
         ;NOTE that even so, the RX byte is not complete
         ;the rest of the data bit and the stop bit are
         ;still on their way.
         BSET         RDRF,SCIFlag
         BCLR         RX,SCIFlag  ;Clear receive-in-progress flag
         RTI

* -------------------------------------------------------------
* P6A Vector definitions
* -------------------------------------------------------------
         ORG          $1FF8       ;Timer vector
         FDB          T_Int

         ORG          $1FFE       ;Reset vector
         FDB          Begin
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com