

# AN4002/D

## Using the 16-bit timer of an HC05 for an interrupt driven software SCI

by Johann Holzmann  
Field Application Engineering, Munich, Germany

### Introduction

In many applications an oversized microcontroller has to be chosen because of the necessity of having an asynchronous serial link to the outside world. Since most of the smaller microcontrollers do not have an SCI they cannot be used even though they could do the rest of the job easily. A software SCI that does not eat up too much of the CPU performance would be a feasible way to fulfil the requirements of most of the applications in question.

### General

The solution discussed within this application note is working in half duplex mode, that means either transmitting or receiving, not both at the same time. This is sufficient for the vast majority of applications and much easier to implement. Looking at the timing (Figure 1.) of a byte transfer using the standard NRZ asynchronous transmission protocol of an RS232 type serial communication link makes it fairly easy to understand the requirements for a software SCI.

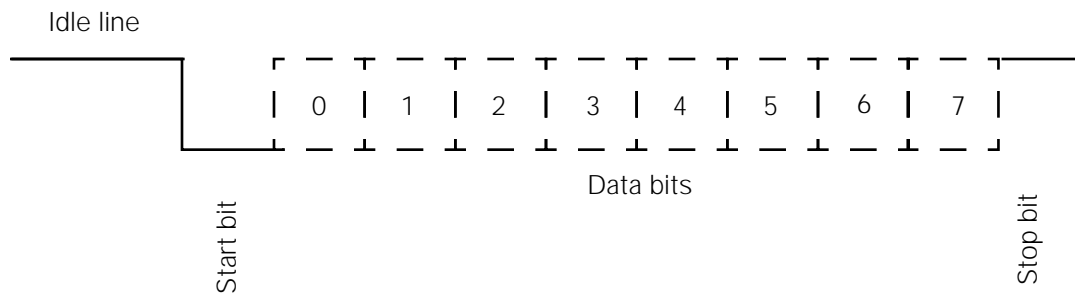


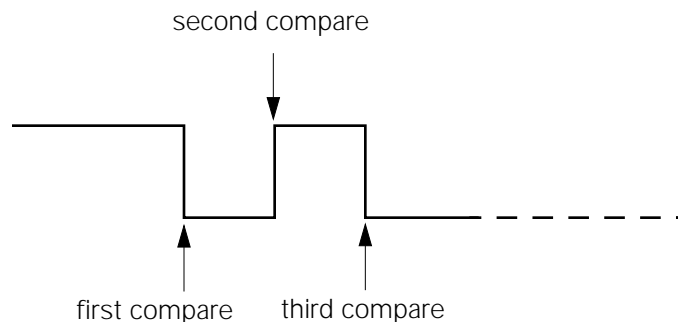
Figure 1. SCI Timing

A complete byte transfer takes 10 bit times for transmission since one byte of data is preceded by a start bit and succeeded by a stop bit. Coming from idle line, which is a logic one, the first falling edge indicates the beginning of the start bit, which is always a logic zero and therefore also the beginning of a byte transfer. All timing of that particular byte transfer is referenced to this falling edge. After one bit time the first data bit begins and so forth until after nine bit times the stop bit begins which is always a logic one.

The 16-bit freerunning timer with one input capture, one output compare and the associated interrupts of an HC05 provide a way to emulate an SCI with fairly little effort. In addition to the timer, a portpin that can be sampled using BRSET or BRCLR commands is required. On some HC05s this can directly be done on the input capture pin, as on the HC05E6; on others the input capture pin has to be connected to a separate input pin. This version of the software SCI uses two bytes of RAM, one for the data register for receiving and transmitting and the other byte for flags. It performs a standard receive/transmit with LSB first.

## Transmitting

For the operation of the software SCI the transmission of a byte will be considered first. The transmission starts with setting the I bit in the condition code register to ensure proper timing and to read the contents of the freerunning timer. Then an offset is added to that value and the result is stored into the output compare register to have a defined time to begin. The OLVL bit is set to zero to produce the required falling edge for the start bit upon the coming compare, and the associated interrupt is enabled (Figure 2.). When running through the interrupt service routine it must distinguish between an input capture and an output compare interrupt to allow for differentiation between the beginning of a byte reception or between receiving/transmitting in progress.

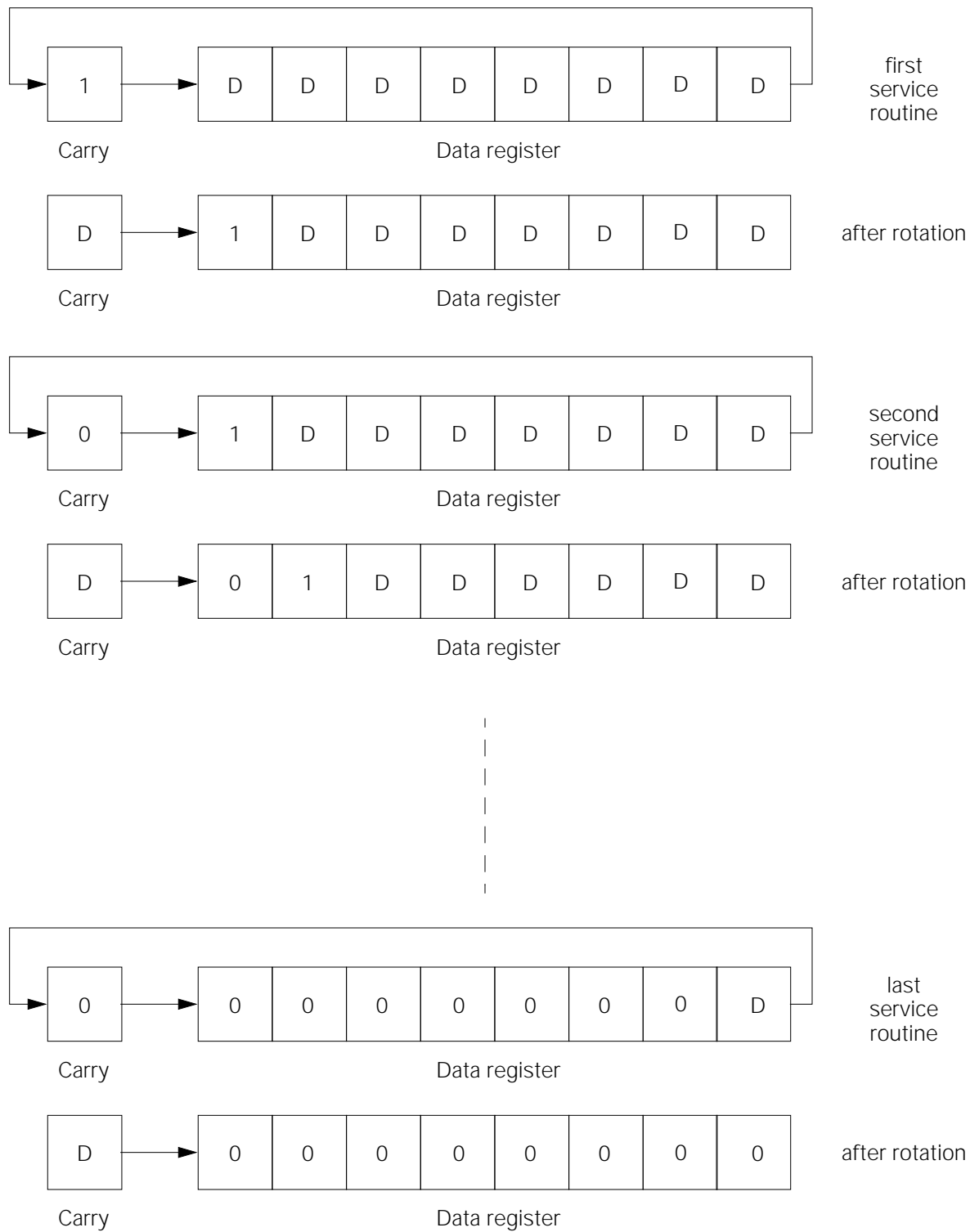


**Figure 2.** Output Compares for transmitting

Since receiving and transmitting both use the output compare, an additional flag that will be set later on in the service routine is checked to distinguish between receiving and transmitting. Then the TX-flag is checked using a BRCLR instruction; on the first service routine entry for a byte transmission, this bit has to be a one. This is achieved during initialisation or at the end of a byte transfer. The BRCLR instruction copies the bit value into the carry. This is necessary to have a one for the stop bit. This bit is then cleared for all the subsequent entries. The next data bit is rotated into the carry and the OLVL bit is set or cleared according to the carry set or clear. The rotation also brings the carry into the data register, so the value for the stop bit is appended to the databyte. One bit time is added to the content of the output compare register, so that the output compare action will produce the data bit on the RXD line.

On all the subsequent entries the carry will be a zero, so that after setting the OLVL bit to the previously mentioned stop bit, the data register is cleared. Since the stop bit is always a one, the data register is not cleared until after the stop bit is rotated into the carry. This acts as a bit counter and therefore additional software to check for the end of a byte is not required.

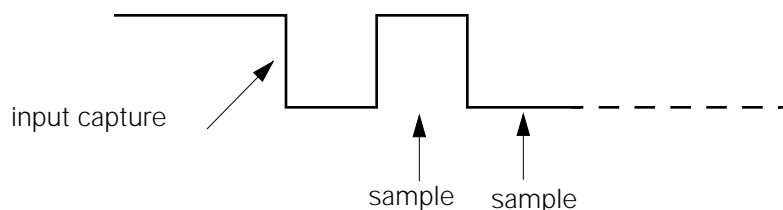
As soon as the OLVL bit is set to a one to produce the stop bit, the transmission complete flag is set. Note that this happens in the service routine at the beginning of the eighth data bit, so the rest of this bit as well as the stop bit still have to be transmitted.



**Figure 3.** Data register and Carry during transmission

## Receiving

The reception of a byte is initiated by an input capture interrupt; the contents of the capture register represent the time of detecting the falling edge of the start bit and therefore the beginning of a byte reception. As before, the interrupt service routine has to distinguish between input capture and output compare. 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 compare register and interrupt enabling is switched accordingly. The data register is cleared and bit seven of the data register set; as before, this acts as a bit counter. In the following compare service routine, the data at the pin (either input capture or port pin) is once sampled using a BRSET instruction; as before, this brings the data bit into the carry. It is rotated into the data register and one bit time is added to the compare register (Figure 4.).



**Figure 4.** Interrupt times for receiving

Since the data register was previously cleared except for bit seven, a zero is always rotated into the carry until the eighth data bit is received (Figure 5.). After reception of the last data bit, the receive data full flag is set and the interrupt is switched back to capture.

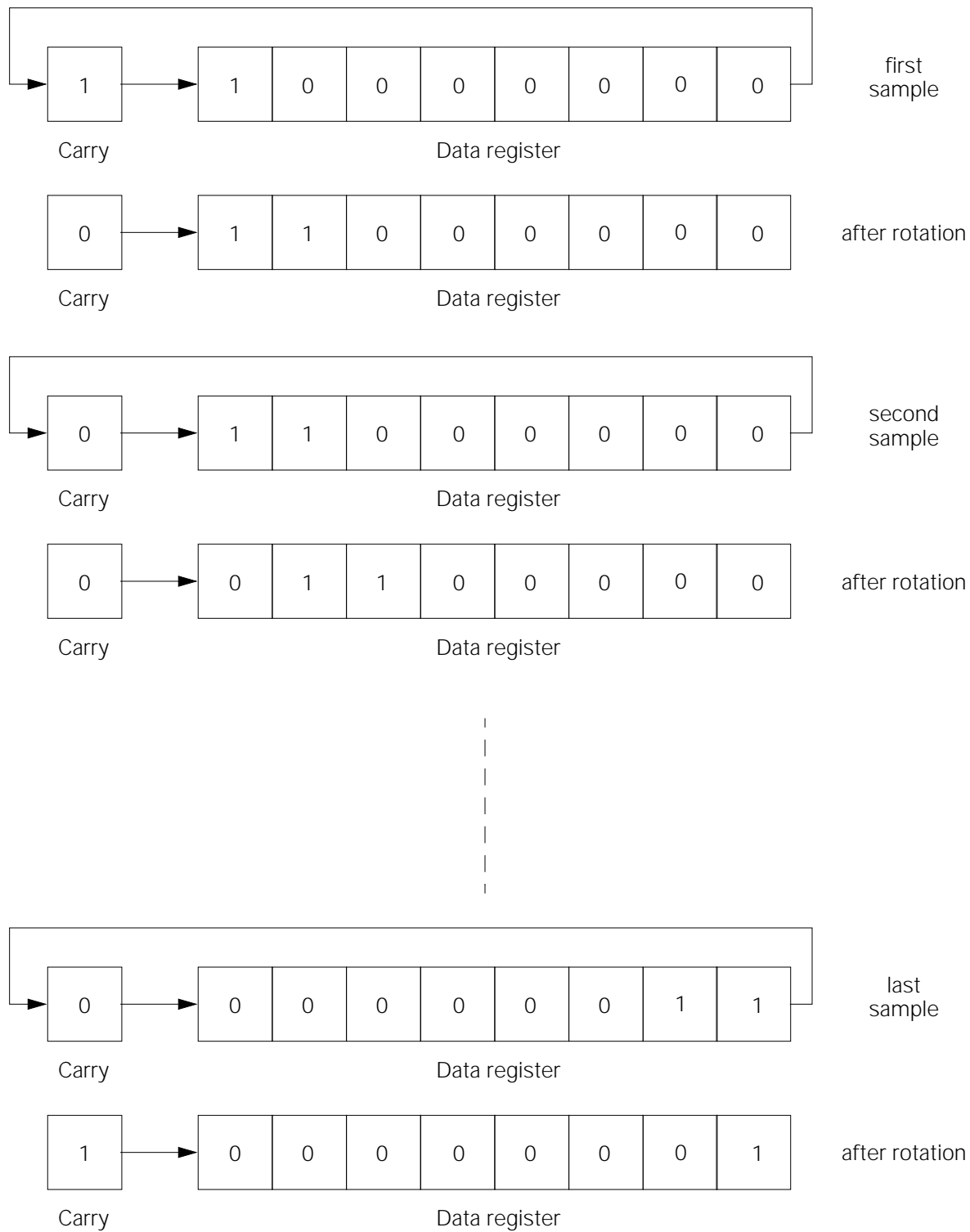
## Considerations

To send a byte of data, this byte has to be written into the data register and the SCI called using a BSR or JSR instruction.

At a bus speed of two megahertz, one run through the interrupt service routine takes around 35 microseconds; this is roughly 35% of the CPU performance at a transmission rate of 9600 baud. This in turn determines the maximum baud rate.

To change the baudrate, just the values for BITHI/BITLO and BIT1 HI/BIT1 LO have to be changed accordingly.

Some applications might need to have an external interrupt serviced while the SCI is working. This can be achieved within the external interrupt service routine, by enabling the timer interrupt. Care must be taken in such a case that the SCI interrupt really gets priority, otherwise the timing will be disrupted.



**Figure 5.** Data register and Carry during reception

## Listing

```

*****
*
*       Software SCI for HC05
*
*****

* The below used init routine is only valid for the HC05E6 and has to
* be adopted accordingly for any other HC05.

PORTC    EQU    $02
DDRC     EQU    $06
PCSR     EQU    $0A
ICLHI    EQU    $14
ICLLO    EQU    $15
TCR      EQU    $12
TSR      EQU    $13
OClHI    EQU    $16
OClLO    EQU    $17
TCNTHI   EQU    $18
TCNTLO   EQU    $19
BITHI    EQU    $00
BITLO    EQU    $34
BITlHI   EQU    $00
BITlLO   EQU    $48
RAM      EQU    $80
ROM      EQU    $800
ICF      EQU    7
OCF      EQU    6
ICIE     EQU    7
OCIE     EQU    6
IEDG     EQU    1
OLVL     EQU    0
RX       EQU    7
TX       EQU    6
RDRF     EQU    5
TDRE     EQU    4

        ORG     RAM

SCIFLAG  RMB    1
SCIDREG  RMB    1

        ORG     ROM

BEGIN    LDA     #$02           ;set the OC pin to high ==> idle line
        STA     PORTC
        LDA     #$FE
        STA     DDRC
        CLR     SCIFLAG       ;clear SCI status register
        CLR     SCIDREG       ;clear SCI data register
        LDA     TSR           ;clear possibly set OC and IC ...
        LDA     IClLO        ;...interrupt flags
        LDA     OClLO
        LDA     #$03         ;connect timer system to ports...
        STA     PCSR         ;...(only for HC05E6)
        LDA     #$81         ;init timer system to OC-level = high...
        STA     TCR          ;...(idle line), IC to falling edge...
                               ;...(detect start bit), disable...
                               ;...OC interrupt, enable IC interrupt...
                               ;...(SCI ready to receive)
        BSET    TX,SCIFLAG   ;"clear" first-entry-to-transmit flag
        CLI

MAIN     BRA     MAIN

SCI      SEI                ;disable interrupts to ensure proper
                               ;timing
        LDX     TCNTHI       ;read current timer value...
        LDA     TCNTLO       ;...and add offset...
        ADD     #$15         ;...store into...
        STA     OClLO        ;...output compare registers
        TXA                ;...to have a defined start
        ADC     #$00         ;...of transmission upon
        STA     OClHI        ;...the following compare.
        LDA     TSR
        LDA     OClLO

```

```

STA    OC1LO
LDA    #01000000          ;generate start bit by setting the...
                          ;...OLVL bit to falling edge,  disable ...
STA    TCR                ;...IC-interrupt, enable OC-interrupt
CLI
RTS

T_INT  LDA    TSR          ;Timer interrupt = SCI interrupt
                          ;read status register to allow...
                          ;...clearance of flags
BRSET  ICIE,TCR,RECEIVE  ;IC-interrupt ?, yes -->received start bit
BRSET  RX,SCIFLAG,RX1    ;otherwise OC-interrupt:
                          ;is SCI receiving?, yes -->
                          ;no, transmitting:

BRCLR  TX,SCIFLAG,TX1    ;First TX entry ? no --> TX1
                          ;yes...
BCLR   TX,SCIFLAG        ;..."set" flag for next entry
                          ;(this is necessary to definitely. .
                          ;...have the Carry clear for each. .
                          ;...transmit entry, because the...
                          ;...Carry clears the databyte...
                          ;...but have it set for the first. .
                          ;...entry to generate the stop bit...
                          ;...and start the "bit-counter")

TX1    ROR    SCIDREG      ;shift next data bit into carry
BCC    TX2              ;if low --> TX2
BSET   OLVL,TCR        ;if high, next OC level to high
BEQ    TX_END          ;if stop bit, --> 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, that means next...
                          ;...data bit is low, therefore...
LDA
ADD    OC1LO           ;...next OC level to low
TAX    #BITLO         ;add bit time to OC
LDA
ADC    OC1HI
STA    #BITHI
STX    OC1HI
RTI    OC1LO

TX_END LDA    OC1LO      ;add last bit time to OC...
ADD    #BITLO         ;...(for the stop bit)
TAX
LDA    OC1HI
ADC    #BITHI
STA    OC1HI
STX    OC1LO
LDA    TSR            ;clear possibly set IC-flag
LDA    IC1LO
LDA    #81            ;disable OC-interrupt,...
STA    TCR            ;...enable IC-interrupt
LDA    #50            ;"clear" the first TX entry flag..
STA    SCIFLAG        ;...again and set the TDRE bit.
                          ;note that even so the TDRE bit...
                          ;...is set, the transmission of...
                          ;...the databyte is not yet completed...
                          ;...there are still the rest of the...
                          ;...last data bit and the stop bit ...
                          ;...to be transmitted.

RTI

RECEIVE LDA    IC1LO    ;start bit has been received...
ADD    #BIT1LO        ;...now add 1 1/2 bit times...
TAX
LDA    IC1HI          ;...to OC for the first....
ADC    #BIT1HI        ;...data bit sampling
STA    OC1HI
LDA    TSR            ;clear possibly set OC flag
STX    OC1LO
BSET   RX,SCIFLAG    ;set receive-in-progress flag
LDA    #41            ;disable IC interrupt, enable...

```

```

        STA     TCR                ;...OC interrupt
        LDA     #$80              ;clear data register and set bit 7...
        STA     SCIDREG           ;...as "bit-counter"
        RTI


RX1     BRSET   0,PORTC,RX2       ;get data bit high or low and...
RX2     ROR     SCIDREG           ;...put it into data register
        BCS     RX_END           ;last bit ? yes --> end
        LDA     OClLO           ;no, add bit time...
        ADD     #BITLO           ;...for next sample
        TAX
        LDA     OClHI
        ADC     #BITHI
        STA     OClHI
        STX     OClLO
        RTI

RX_END  LDA     TSR                ;byte received, clear possibly set...
        LDA     IC1LO           ;...IC flag...
        LDA     #$81           ;...disable OC interrupt...
        STA     TCR                ;...enable IC interrupt
        BSET   RDRF,SCIFLAG       ;set receive register full flag
                                        ;(note that even so the RDRF flag...
                                        ;...is set the received byte is...
                                        ;...not yet completed, the rest of. .
                                        ;...the last data bit and the...
                                        ;...stop bit are still on their way)
        BCLR   RX,SCIFLAG         ;clear receive-in-progress flag
        RTI

        END

```

All products are sold on Motorola's Terms & Conditions of Supply. In ordering a product covered by this document the Customer agrees to be bound by those Terms & Conditions and nothing contained in this document constitutes or forms part of a contract (with the exception of the contents of this Notice). A copy of Motorola's Terms & Conditions of Supply is available on request.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

The Customer should ensure that it has the most up to date version of the document by contacting its local Motorola office. This document supersedes any earlier documentation relating to the products referred to herein. The information contained in this document is current at the date of publication. It may subsequently be updated, revised or withdrawn.

Literature Distribution Centres:

EUROPE: Motorola Ltd., European Literature Centre, 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

ASIA PACIFIC: Motorola Semiconductors (H.K.) Ltd., Silicon Harbour Center,

No. 2, Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

JAPAN: Nippon Motorola Ltd., 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.

USA: Motorola Literature Distribution, P.O. Box 20912, Phoenix, Arizona 85036.

