APPLICATION NOTE

SCI SOFTWARE COMMUNICATION WITH A PC USING ST72251 16-BIT TIMER

by 8-Bit Micro Application Team

1 INTRODUCTION

This document presents a standard communication between a ST7 microcontroller and a PC. It shows how to emulate a SCI communication in half-duplex mode by software, using the ST7 timer resource.

2 SCI COMMUNICATION

The main features of a SCI communications interface are summarized below.

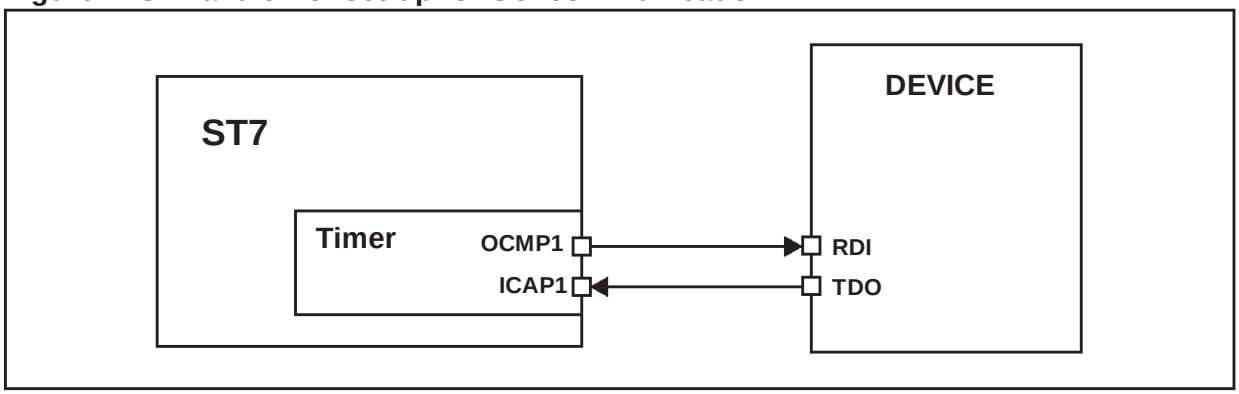
2.1 MAIN FEATURES

The Serial Communication Interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format.

The SCI allows a very wide range of baud rates and different baud rates for transmission and reception.

In SCI communication, only two signals are needed, one for transmission and the other for reception. No clock signal is needed as it works in asynchronous mode. Each device has a Transmit Data Output pin (OCMP1 pin for the ST7) and a Receive Data Input pin (ICAP1 pin for the ST7). See Figure 1.

Figure 1. ST7 and timer set-up for SCI communication



The user must be very careful in identifying the use of each pin. This can easily be done by putting the device in transmission and checking with an oscilloscope if a transmission frame is present or not.

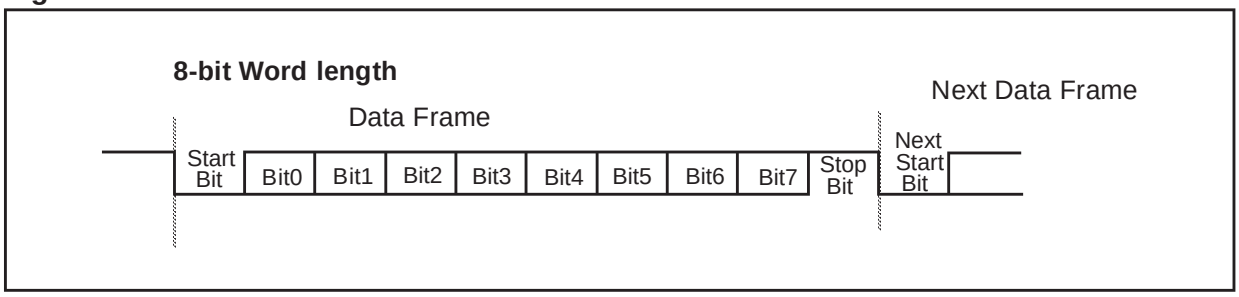
2.2 BAUD RATES

Transmission and reception can be driven by their own baud rate generator. However be aware that for correct communication, the receiver must have a reception baud rate strictly equal to the transmission baud rate of the transmitter. **If not, the communication will be corrupted**. As soon as this condition is met, a wide range of baud rates is possible.

2.3 FRAMES

Any transmission is Least Significant Bit first. A data word is usually 8 bits long. A data frame begins with a «start bit», which is a '0' bit and ends with a «stop bit», which is a '1' bit. See Figure 2.

Figure 2. Frames



In some cases, a 9th bit can be used, as a parity bit or as a second stop bit.

3 RS232 COMMUNICATION WITH A PC

3.1 MAIN FEATURES

Electrical and protocol characteristics of RS232 are different from those used by the SCI. In RS232 communication, high level is typically +7V and low level is typically -7V, while the SCI peripheral works at TTL levels (0, +5V).

Furthermore, the polarities are different. A '1' bit coming from the SCI corresponds to a '0' bit in RS232, and a '0' bit to a '1' bit. **It is true for all bits including the start and stop bits**.

So it is necessary to implement a conversion between the PC and the ST7. In the application, a MAX232 is used for this purpose.

3.2 PC CONFIGURATION

The PC will be used as a terminal. The description below refers to the Windows 3.11 environment. Under Windows 95, the terminal application is called hyperterminal.

Under Windows, open the «terminal» application. To configure it, go to the communication in parameters menu. The options of this window must be the same as the ones defined for your ST7.

After selecting the right serial communication port, select the same baud rate as the one set for the ST7. As the PC accepts only one baud rate, transmission and reception baud rates will have the same value. Data word must be 8 bits, but you can choose to use 1 or 2 stop bits. «Flow control» can be either X_{on}/X_{off} or none.

The PC is then correctly configured.

4 ST72251 CONFIGURATION

This application was implemented with a ST72251. This microcontroller uses a 16 MHz external clock. A description of the baud rate selection is given later in this application note.

4.1 GENERAL INITIALIZATION

ST72251 internal clock works at 16MHz/2 = 8MHz.

Two pins of the ST7 are used:

- Input capture 1 pin of timer A (ICAP1 pin).
- Output compare 1 pin of timer A (OCMP1 pin).

The ICAP1 pin is the alternate function of pin 0 of Port B. During the initialization it is configured as an input.

The OCMP1 pin is the alternate function of pin 1 of Port B. During the initialization it is configured as an output with high level. This level is the default level for SCI communication.

Please, refer to the datasheet for the description of the initialization of these pins as input or output.

4.2 TIMER CONFIGURATION

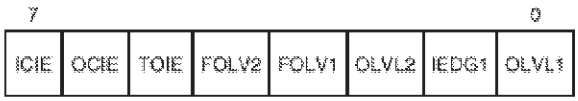
The timer consists of a 16-bit free-running counter driven by a programmable prescaler. It includes a time overflow, two input captures and two output compares.

4.2.1 General description

The timer is associated with two control registers, one status register, and six pairs (16-bits) of data registers for to the input captures, the output compares, the counter and the alternate counter.

The bits used in the application are described below.

Timer A control register 1 (TACR1):



- ICIE: Input Capture Interrupt Enable

If set, a timer interrupt occurs when the ICF1 or ICF2 bit of the TASR register is set.

- OCIE: Output Compare Interrupt Enable

If set, a timer interrupt occurs when the OCF1 or OCF2 bit of the TASR register is set.

- FOLV1: Forced Output Level 1

Forces the OLVL1 bit to be copied to the OCMP1 pin when set.

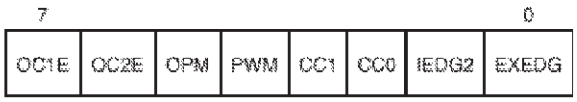
- IEDG1: Input Edge 1

When reset, a falling edge triggers the capture. When set, a rising edge triggers the capture.

- OLVL1: Output Level 1

This bit is copied to the OCMP1 pin if an output compare 1 event occurs while the OC1E bit of TACR2 register is set.

Timer A control register 2 (TACR2):



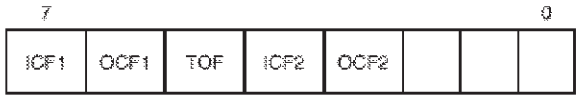
- OC1E: Output Compare 1 Enable

When set, the OCMP1 pin is assigned to the output compare 1 capability.

- CC1-CC0: Clock Control

The value of the timer clock is given by these bits.

Timer A status register (TASR):



This register is used in the application to recognize which event has generated a timer interrupt.

When set, the ICF1 bit (Input Capture Flag 1) indicates that an input capture 1 occurred, and the OCF1 bit (Output Compare Flag 1) indicates that an output compare 1 occurred.

The other flags are not used in this application.

4.2.2 Timer initialization

First the value of the prescaler must be chosen. This is done using the CC1&CC0 bits of the TACR2 register. In the application $f_{CPU}/4$ is selected, resetting these two bits.

The OCMP1 pin is dedicated to the output compare 1 capability by setting the OC1E bit of the TACR2 register.

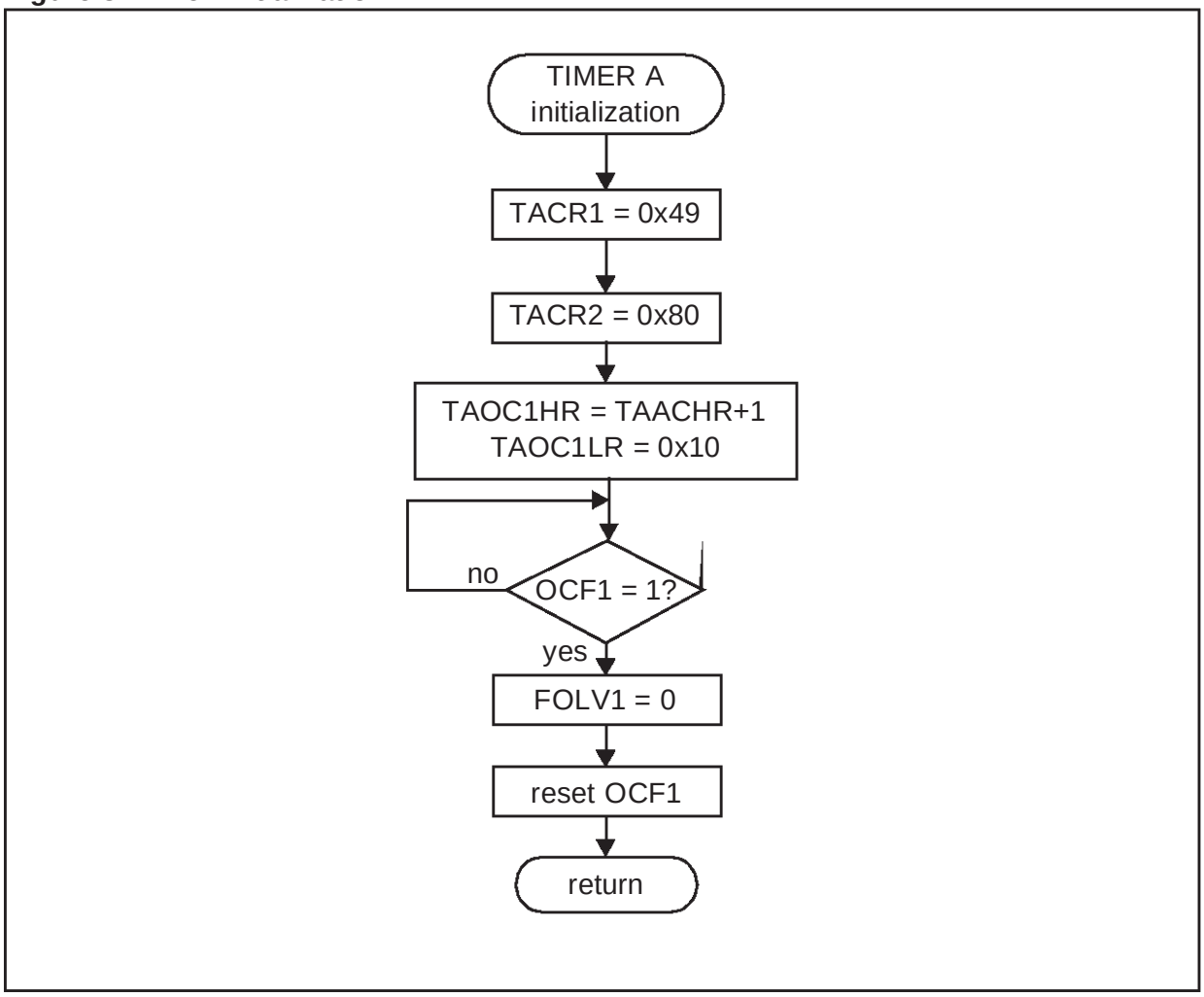
In the application a new byte is recognized by the falling edge by means of the start bit. So the falling edge is selected for input capture 1, resetting the IEDG1 bit of the TACR1 register.

The default value for SCI communication (‘1‘) is put into the OLVL1 bit of the TACR1 register, and this value is forced to the OCMP1 pin setting the FOLV1 bit of the TACR1 register. So the line is put high.

The output compares are enabled by setting the OCIE bit of the TACR1 register.

A first output compare 1 is generated so the OLVL1 value is put on the OCMP1 pin. Then we can stop forcing this value by resetting the FOLV1 bit of the TACR1 register.

Figure 3. Timer initialization



5 SCI COMMUNICATION BETWEEN ST7 AND PC THROUGH RS232

The software at the end of this application note is the driver for emulating the SCI with a timer. The complete software can be found on the ST internet website. It is of course only an example. It is up to the user to adapt it to his specific application.

5.1 GENERAL DESCRIPTION

In this application, a ST72251 is connected to a PC. The communication is performed using the «terminal» application of Windows 3.11.

A ASCII character is sent by the PC to the ST7. As the ST7 sends the same character as received, it is very easy to check if the communication is correct. **Note that this program does not manage communication errors.**

The communication described is a half-duplex communication. The software sends a first character, then it enters reception mode. When a character is received, it enters transmission mode, sends it and so on.

The software uses interrupt strategy which allows to have other applications to work at the same time.

The user must select its transmission speed by defining variables del_1bh, del_1bl, del_sampl, del_samph.

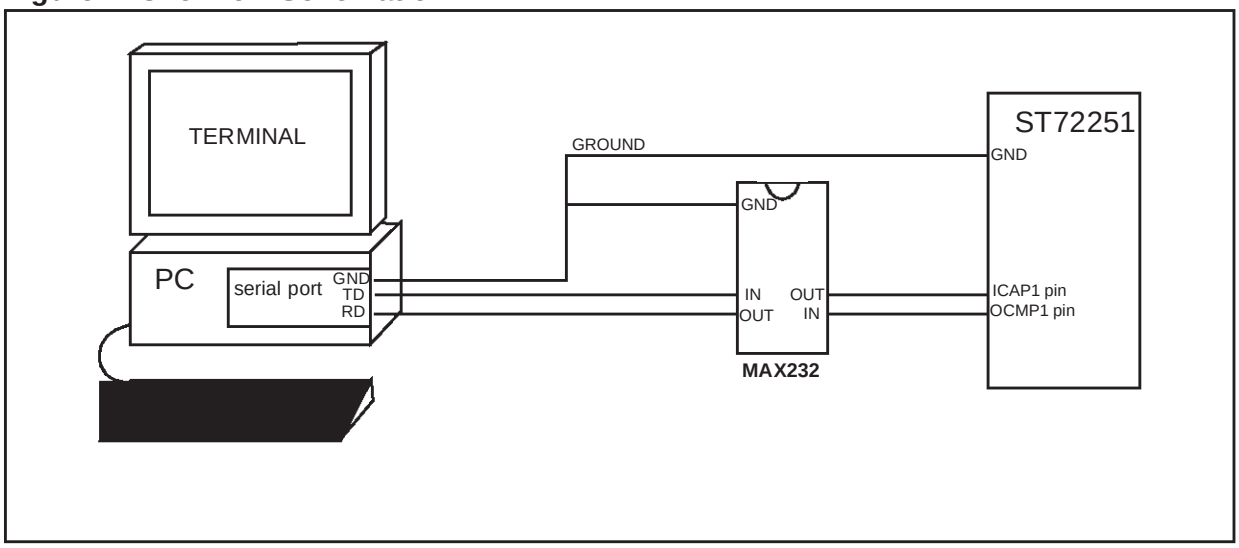
The output compare 1 is used to generate the bit clock. So an output compare 1 interrupt occurs each time a bit must be sent or received.

When the ST7 is in reception mode, it detects the start of a new byte (falling edge of the start bit). An input capture 1 interrupt occurs. Then each data bit is sampled three times in the middle of the bit and a decision is made. If there are more ‘1‘s than ‘0‘s, the bit received is a ‘1‘, else it is a ‘0‘.

5.2 HARDWARE

The ST7 emulated MCU cannot be directly connected to a PC, as it uses the RS232 protocol. The conversion between SCI and RS232 can be done using a MAX232. An overview schematic is given below (Figure 4.).

Figure 4. Overview Schematic



Be sure that the three main devices (PC, ST7, MAX232) have the same electrical reference (GND).

The Receive Data pin (RD) of the serial port of the PC must correspond to the OCMP1 pin of the ST7, and the Transmit Data pin (TD) to the ICAP1 pin.

5.3 FLOWCHARTS

The flowcharts of the application are presented below. The main flowchart is given in Figure 5. The timer A interrupt flowchart is given in Figure 6. and the timer A initialization is given in Figure 3.

The software uses software register: sci_status register.

			TE	BS	RE	BR	SP

SP: Sampling Phase of reception mode.
RE: Reception Enable.
TE: Transmission Enable.
BR: Byte Received flag.
BS: Byte Sent flag.

Figure 5. Main Flowchart

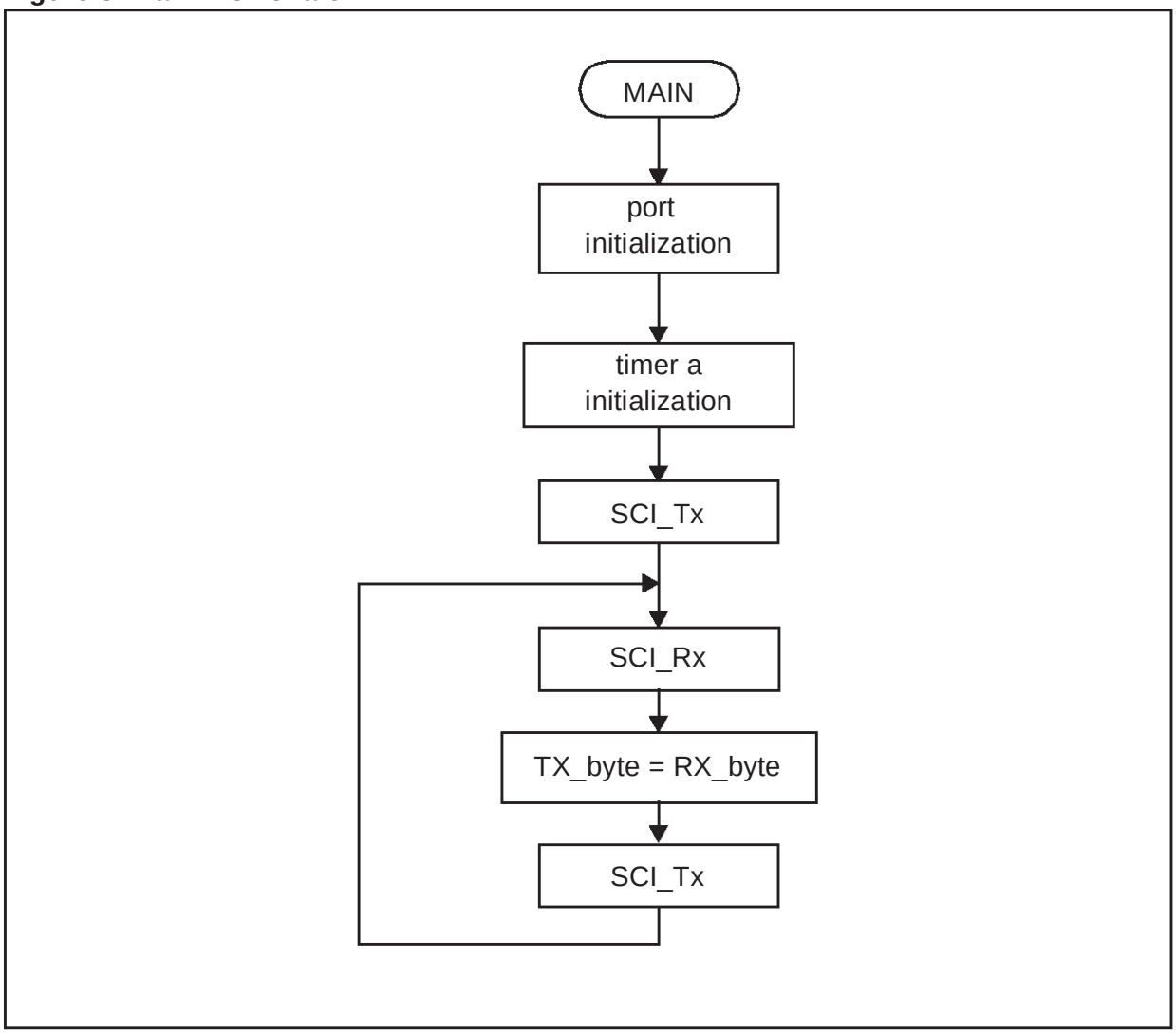


Figure 6. Timer A Interrupt Flowchart

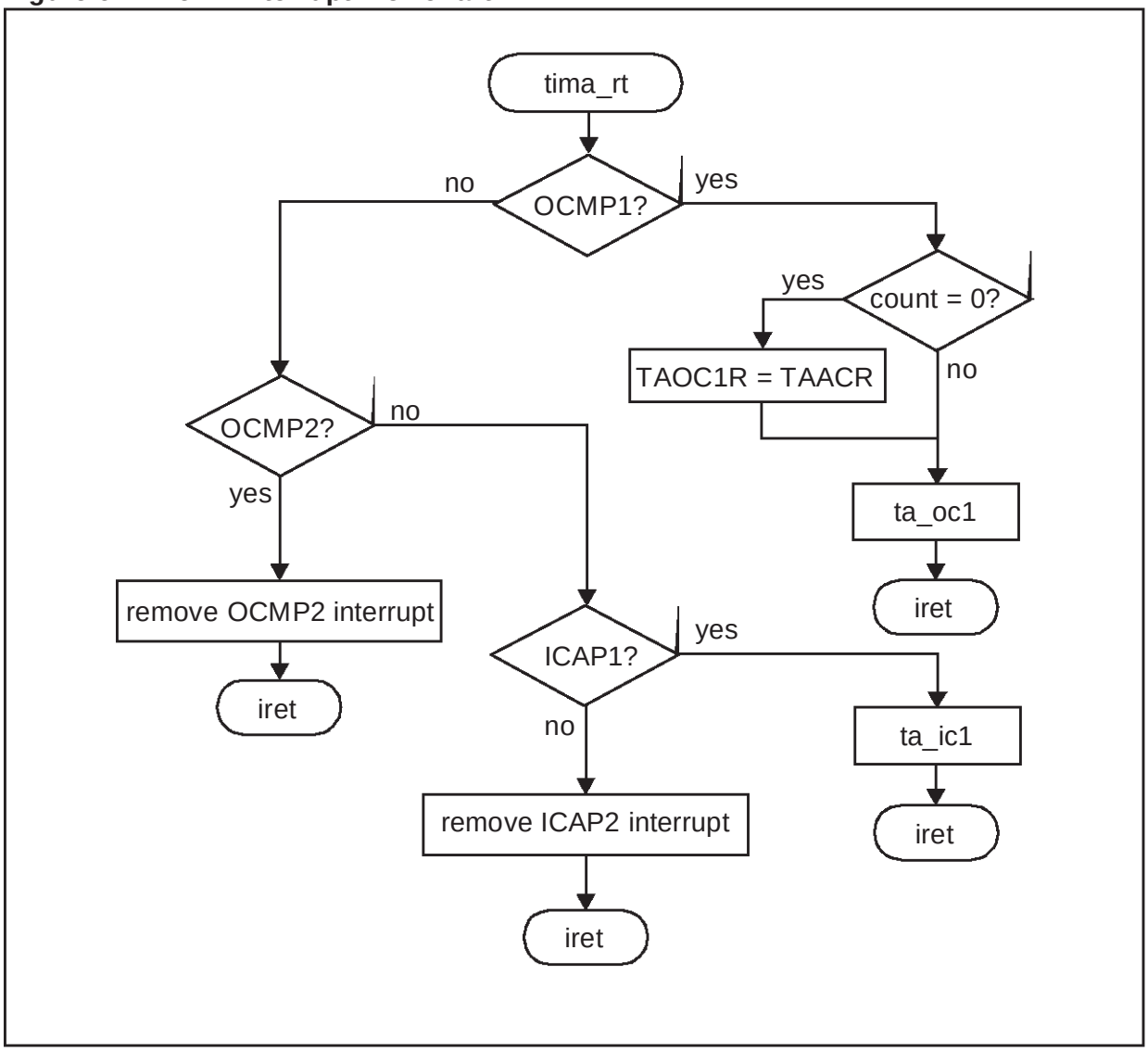


Figure 7. Transmission and Reception Flowcharts

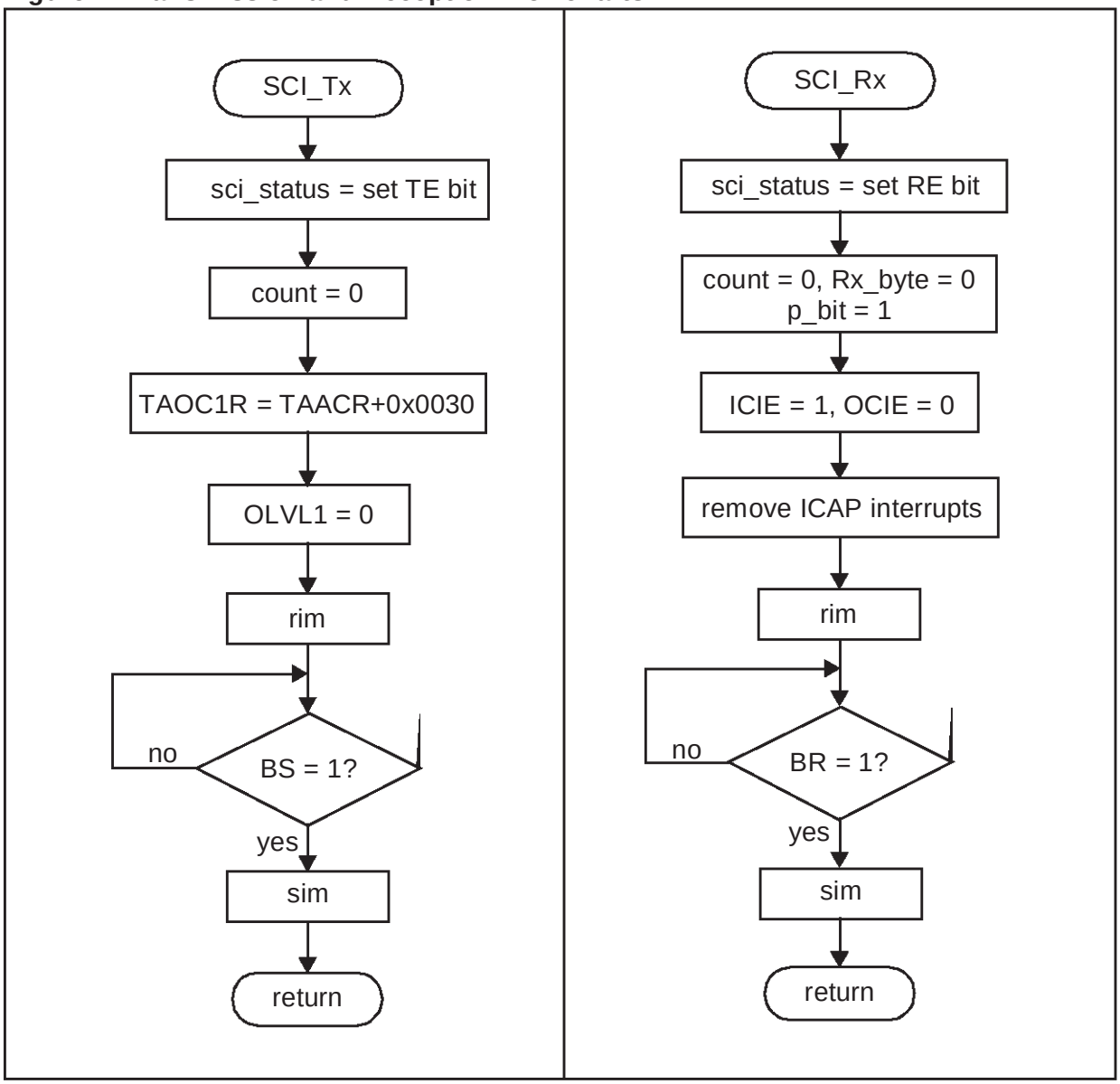


Figure 8. Input Capture 1 Flowchart

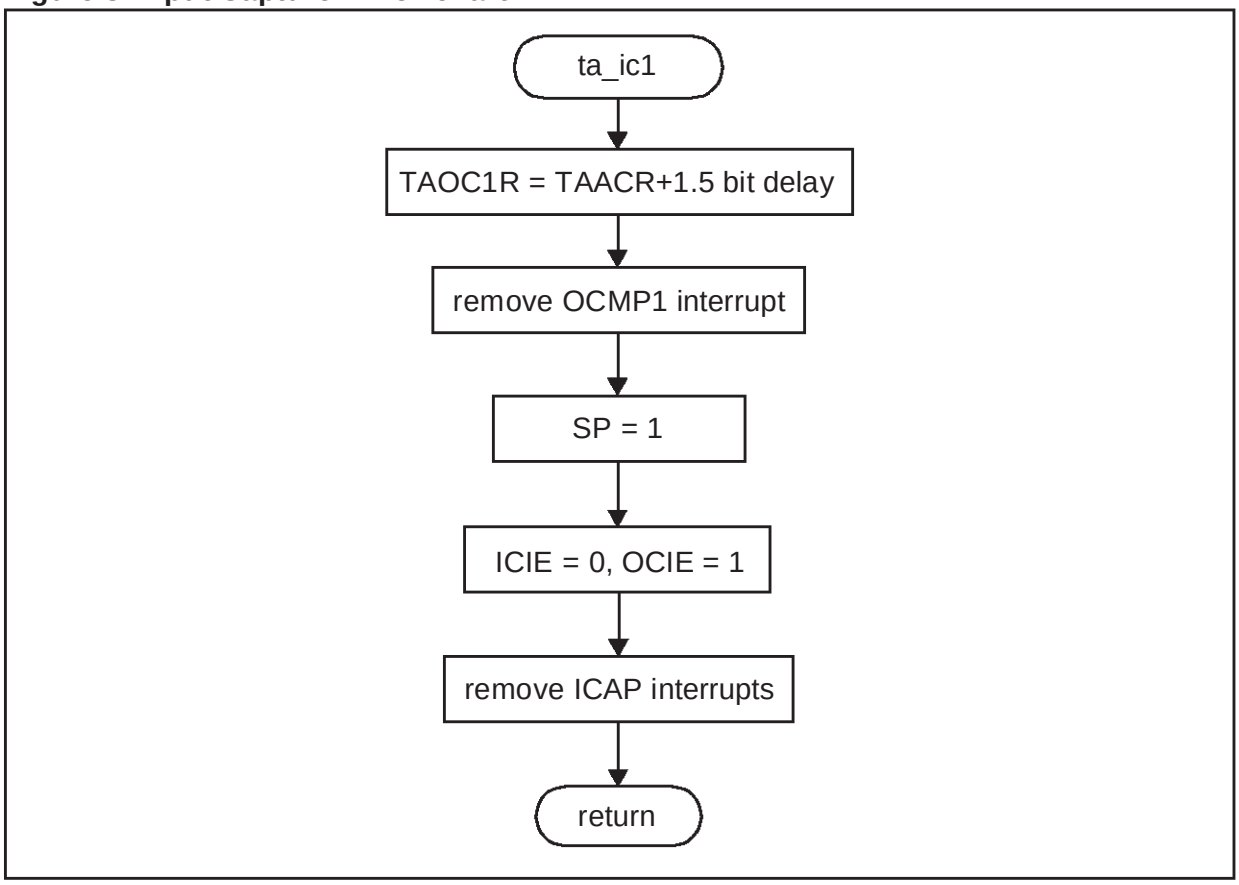
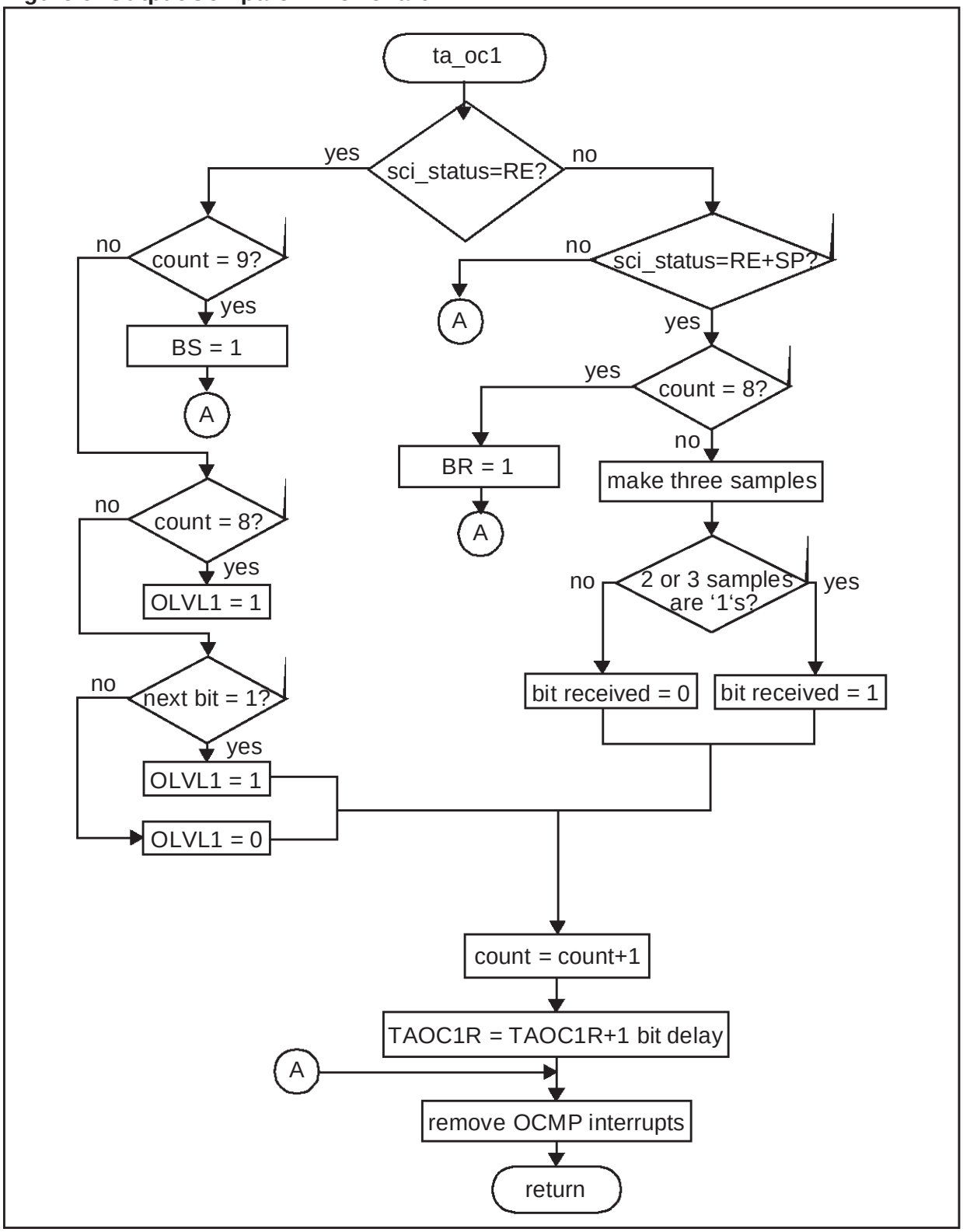


Figure 9. Output Compare 1 Flowchart



5.4 SOFTWARE

The assembly code given below is guidance only. The file cannot be used alone. The complete software can be found in the ST internet website.

```
st7/

;*************** (c) 1997 STMicroelectronics **********************
;
;PROJECT:   EVALUATION BOARD - ST7 SCI DEMO SYSTEM
;COMPILER:  ST7 ASSEMBLY CHAIN
;MODULE:    sci_io.asm
;CREATION DATE: 06/03/98
;AUTHOR:    PPG Micro Application / STMicroelectronics Rousset
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;  THE SOFTWARE INCLUDED IN THIS FILE IS FOR GUIDANCE ONLY. STMicroelectronics
;  SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL
;  DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THIS SOFTWARE.
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;DESCRIPTION:  ST7 emulation of a SCI communication with a timer software
;           driver for a RS232 communication. Communication with a
;           terminal in half-duplex mode. Validated on ST72251
;           using a 16MHz clock.
;           When a byte is sent, the program waits for the reception of
;           a new byte. It uses timer A interrupts.
;           Six basic routines are defined:
;               - a timer initialization routine
;               - a transmission routine which sends the new byte to
;                the terminal.
;               - a reception routine which puts the received byte
;                from the terminal to RX_byte.
;               - the timer A interrupts routine.
;               - an input capture 1 routine.
;               - an output compare 1 routine.
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;MODIFICATIONS:
;
; 06/03/98 - V2.0 - New assembly version
;
```

```
;******************************************************************************

    TITLE  "sci.ASM"
                ; This title will appear on each
                ; page of the listing file.
    MOTOROLA    ; This directive forces the Motorola
                ; format for the assembly (default).
    #INCLUDE "st72251.inc" ; Include st7285 registers and
                ; memory mapping file.


;******************************************************************************
;    Macro definitions
;******************************************************************************

; bit definition of TACR1 register ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 #define    OLVL1 $0   ; Output level 1.
 #define    FOLV1 $3   ; Force Output Level 1.

; bit definition of TASR register ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 #define    OCF2  $3   ; Output Compare Flag 2.
 #define    ICF2  $4   ; Input Capture Flag 2.
 #define    OCF1  $6   ; Output Compare Flag 1.
 #define    ICF1  $7   ; Input Capture Flag 1.

; bit definition of the sci_status register ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

sci_status register

			TE	BS	RE	BR	SP

```
#define     SP    $0   ; Sampling Phase of reception mode.
 #define    BR    $1   ; Byte Received flag.
 #define    RE    $2   ; Reception Enable.
 #define    BS    $3   ; Byte Sent flag.
 #define    TE    $4   ; Transmission Enable.

#define     Mask_ICAP1$01;Mask for the PB0 pin (ICAP1)
```

```
;*******************************************************************************
;    RAM SEGMENT
;*******************************************************************************
  BYTES           ; following addresses are 8 bit length.

  segment byte at 80-FF 'ram0'

; Variable definitions ************************************************

.TX_byte    DC.B 0; Byte to transmit.
.RX_byte    DC.B 0; Reception byte.
.count      DC.B 0; Indicate the bit number.
.sci_status DC.B 0; 5 bits software register that indicates the
                  ; communication mode.
                  ; See above for the bits definition.
.p_bit      DC.B 0 ; Indicate the position of the bit to receive.
.tmp        DC.B 0


;*******************************************************************************
;    Public routines (defined here)
;*******************************************************************************
; routines

;*******************************************************************************
;    External routines  (defined elsewhere)
;*******************************************************************************
; routines

;*******************************************************************************
;     MACROs SUB-ROUTINES LIBRARY SECTION
;*******************************************************************************

;    (must be placed here and not at the end of the file)


    WORDS

  segment 'rom'

; Constants definitions ***********************************************

; Baud rate selection for f_CPU = 8MHz ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes of the number of timer cycles to create a delay of
; one bit for a transmission speed of 19200bd.
;.del_1bl   DC.B $67
;.del_1bh   DC.B $00
```

```
; Baud rate selection for f_CPU = 8MHz ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes of the number of timer cycles to create a delay of
; one bit for a transmission speed of 9600bd.
;.del_1bl   DC.B $D1
;.del_1bh   DC.B $00
; Baud rate selection for f_CPU = 8MHz ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes of the number of timer cycles to create a delay of
; one bit for a transmission speed of 4800bd.
.del_1bl    DC.B $A1
.del_1bh    DC.B $01
; Baud rate selection for f_CPU = 8MHz ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes of the number of timer cycles to create a delay of
; one bit for a transmission speed of 2400bd.
;.del_1bl   DC.B $41
;.del_1bh   DC.B $03
; Baud rate selection for f_CPU = 8MHz ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes of the number of timer cycles to create a delay of
; one bit for a transmission speed of 1200bd.
;.del_1bl   DC.B $82
;.del_1bh   DC.B $06

; Delay to sample in the middle of the first bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes to sample in the middle of the first bit for
; a transmission speed of 19200bd.
;.del_sampl DC.B $4E
;.del_samph DC.B $00
; Delay to sample in the middle of the first bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes to sample in the middle of the first bit for
; a transmission speed of 9600bd.
;.del_sampl DC.B $FD
;.del_samph DC.B $00
; Delay to sample in the middle of the first bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes to sample in the middle of the first bit for
; a transmission speed of 4800bd.
.del_sampl  DC.B $35
.del_samph  DC.B $02
; Delay to sample in the middle of the first bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes to sample in the middle of the first bit for
; a transmission speed of 2400bd.
;.del_sampl DC.B $A5
;.del_samph DC.B $04
; Delay to sample in the middle of the first bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; Low and high bytes to sample in the middle of the first bit for
; a transmission speed of 1200bd.
;.del_sampl DC.B $88
;.del_samph DC.B $09
```

```
; Program code ********************************************************************

;    ****************************************
;    *                      *
;    *    SUB-ROUTINES LIBRARY SECTION *
;    *                      *
;    ****************************************

;-------------------------------------------------------------------------------
;ROUTINE NAME: port_init
;INPUT/OUTPUT: None.
;DESCRIPTION: Port initialization routine.
;COMMENTS:
;-------------------------------------------------------------------------------
.port_init

       bset PBDDR,#1 ; Pin 1 of port B in output mode,
       bset PBOR,#1 ; will be OCMP1 pin in alternate function.
       bres PBOR,#0 ; Pin 0 of port B in input mode,
       bres PBDDR,#0 ; will be ICAP1 pin in alternate function.
       bres PBDR, #0;Reset of Pins 0 & 1 of the PBDR register
       bres PBDR, #1
       ret

;-------------------------------------------------------------------------------
;ROUTINE NAME: ta_oc1
;INPUT/OUTPUT: TAOC1R, sci_status, TX_byte / TAOC1R, sci_status, RX_byte.
;DESCRIPTION: Send the next bit in transmission mode. Sample and determine
;           the value of the next bit in reception mode.
;COMMENTS:  Least significant bit first.
;-------------------------------------------------------------------------------
.ta_oc1

       ld   A,sci_status ; Put SCI mode in A.
       cp   A,#16 ; Transmission mode?
       jrne RX    ; Yes continue, else go to RX.

; Transmission process ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A, count
       cp   A,#9       ; Stop bit sent?
       jrne TX_S       ; Yes continue, else go to TX_S.

; Byte sent ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       bset sci_status,#BS ; Byte sent mode.
       jra  end_it

TX_S   cp   A,#8       ; All data sent?
       jrne TX        ; Yes continue, else go to TX.
```

```
; Send stop bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       bset TACR1,#OLVL1   ; Put OCMP1 pin in high level for stop bit.
       jra  end_b

; Send next data bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TX     btjf TX_byte,#0,TX_0 ; Is the next bit to send equal to 1?
       bset TACR1,#OLVL1   ; Put OCMP1 pin in high level for 1 bit.
       jra  end_b

TX_0   bres TACR1,#OLVL1   ; Put OCMP1 pin in low level for 1 bit.

; Create a delay of 1 bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
end_b  srl  TX_byte     ; Put the next bit to send as bit 0 of TX_byte.
end_rx inc  count      ; Increment bit number.
       ld   X,TAOC1HR     ; Check the last OCMP1 value.
       ld   A,TAOC1LR
       add  A,del_1bl    ; Add a delay of one bit to this value.
       ld   Y,A
       ld   A,X
       adc  A,del_1bh
       ld   TAOC1HR,A    ; An OCMP1 will occur one bit after
       ld   A,Y       ; the current one.
       ld   TAOC1LR,A

; End of the output compare management ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
end_it ld   A,TASR      ; Remove pending output compare interrupts.
       ld   A,TAOC1LR
       ld   A,TAOC2LR

       ret           ; End.

RX     cp   A,#5  ; Reception mode?
       jrne end_it      ; Yes continue, else go to end_it.

; Reception mode ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,count
       cp   A,#8       ; Byte completely received?
       jreq RX_e       ; No continue, else go to RX_e.

; Check the new bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,PBDR
       and  A,#Mask_ICAP1 ; Check the first sample of ICAP1 pin.
       ld   tmp,A
       ld   A,PBDR
       and  A,#Mask_ICAP1 ; Check the second sample of ICAP1 pin.
       add  A,tmp      ; Add it to the first value.
       ld   tmp,A
       ld   A,PBDR
```

```
       and  A,#Mask_ICAP1 ; Check the third sample of ICAP1 pin.
       add  A,tmp      ; Add it to the former value.
       cp   A,#$02 ; Two or three '1' received?
       jrmi RX_0       ; Yes continue, else go to RX_0.
       ld   A,RX_byte
       add  A,p_bit
       ld   RX_byte,A    ; Set bit with number count of RX_byte.

; Create a delay of 1 bit ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RX_0   sll  p_bit
       jra  end_rx

; New byte received ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RX_e   bset sci_status,#BR ; Byte received mode.
       jra  end_it

;-------------------------------------------------------------------------------
;ROUTINE NAME : ta_ic1
;INPUT/OUTPUT: None / sci_status.
;DESCRIPTION  : Enters in sampling mode.
;COMMENTS : Done when an input capture 1 event occurs
;           while in reception mode.
;-------------------------------------------------------------------------------
.ta_ic1

; Create a delay of one bit and a half ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   X,TAACHR     ; Check the current value of timer A.
       ld   A,TAACLR
       add  A,del_sampl    ; Add a delay to have a OCMP1, 1.5 bit
       ld   Y,A       ; after the ICAP1.
       ld   A,X
       adc  A,del_samph
       ld   TAOC1HR,A     ; An output compare will occur in 1.5 bit.
       ld   A,TASR      ; Remove pending OCMP1 interrupt.
       ld   A,Y
       ld   TAOC1LR,A

; Enter in sampling mode ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       bset sci_status,#SP ; Enter sampling mode.
       ld   A,#$41      ; Enable OCMP interrupts,
       ld   TACR1,A     ; disable ICAP interrupts.
       ld   A,TAIC1LR    ; Remove pending ICAP interrupts.
       ld   A,TAIC2LR

ret
```

```
;-------------------------------------------------------------------------
;ROUTINE NAME:TA_init
;INPUT/OUTPUT: None.
;DESCRIPTION: Initialization routine of timer A.
;COMMENTS:
;-------------------------------------------------------------------------
.TA_init

; Initialization of control registers ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,#$49      ; OCMP enabled, force the OLVL1 value to
       ld   TACR1,A     ; OCMP1 pin when OC1E is set.
       ld   A,#$80
       ld   TACR2,A     ; Set OC1E: OCMP1 pin in alternate mode.

; Generate a first OCMP1 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,TAACHR
       inc  A
       ld   TAOC1HR,A
       ld   A,#$10
       ld   TAOC1LR,A
att    btjf TASR,#OCF1,att    ; Waits until a OCMP1 occurs.
       bres TACR1,#FOLV1 ; Stop forcing OLVL1 to OCMP1 pin.
       ld   A,TAOC1LR    ; Clear OCMP1 flag.

       ret

;-------------------------------------------------------------------------
;ROUTINE NAME: SCI_Tx
;INPUT/OUTPUT: Byte to transmit (TX_byte) / None.
;DESCRIPTION: Transmit one byte via output_compare_1 pin of timer A.
;COMMENTS:  Least significant bit first.
;-------------------------------------------------------------------------
.SCI_Tx

; Enter transmission mode ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,#$10
       ld   sci_status,A   ; Enter transmission mode.
       clr  count      ; Clear bit number (LSB first).

; Generate a delay before the first OCMP1 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   X,TAACHR     ; Check the current value of timer A.
       ld   A,TAACLR
       add  A,#$30     ; Add a delay to this value.
       ld   Y,A
       ld   A,X
       adc  A,#$00
       ld   TAOC1HR,A    ; An output compare will occur after the delay.
       ld   A,Y
       ld   TAOC1LR,A
```

```
       bres TACR1,#OLVL1   ; Put OCMP1 pin in low level for start bit.
       rim          ; Remove interrupt mask.

; Loop for sending byte ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
waitT  btjf sci_status,#BS,waitT ; Byte sent mode?
       sim             ; Set interrupt mask.
       ret

;-------------------------------------------------------------------------------
;ROUTINE NAME : SCI_Rx
;INPUT/OUTPUT: None / byte received (RX_byte).
;DESCRIPTION  : Receive new byte via input_capture_1 pin of timer A.
;COMMENTS :  Least significant bit first.
;-------------------------------------------------------------------------------
.SCI_Rx

; Enter reception mode ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       ld   A,#$04
       ld   sci_status,A   ; Reception mode.
       ld   A,#0
       ld   count,A     ; Clear bit number (LSB first).
       ld   RX_byte,A    ; Clear reception byte.
       ld   A,#1
       ld   p_bit,A     ; Initialize p_bit.
       ld   A,#$80
       ld   TACR1,A     ; ICAP enabled, OCMP disabled.
       ld   A,TASR      ; Remove ICAP interrupts.
       ld   A,TAIC1LR
       ld   A,TAIC2LR
       rim           ; Remove interrupt mask.

; Loop for receiving byte ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
waitR  btjf sci_status,#BR,waitR   ; Byte received mode?
       sim      ; Set interrupt mask.
       ret
```

```
;    ****************************************
;    *                      *
;    *     MAIN-ROUTINES SECTION *
;    *                      *
;    ****************************************

.main

; Initializations ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       call  port_init
       call  TA_init
       ld    A,#'t'
       ld    TX_byte,A    ; First byte to send.
       call  SCI_Tx

; Main loop, transmission and reception ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
loop   call  SCI_Rx
       ld    A,RX_byte
       ld    TX_byte,A    ; Send the received byte.
       call  SCI_Tx
       jra   loop


;    ****************************************
;    *                      *
;    * INTERRUPT SUB-ROUTINES LIBRARY SECTION*
;    *                      *
;    ****************************************

.dummy_rt    iret   ; Empty subroutine. Go back to main (iret instruction).

; Timer interrupts routine ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.tima_rt

       btjf  TASR,#OCF1,oc2   ; Is it an OCMP1? Yes continue, else go to oc2.
       ld    A,count
       jrne  ts       ; count = 0? Yes continue, else jump to ts.
       ld    A,TAACHR     ; Check the current value of timer A.
       ld    TAOC1HR,A    ; Save it in the OCMP1.
       ld    A,TAACLR
       ld    TAOC1LR,A
ts     call  ta_oc1
       iret

oc2    btjf  TASR,#OCF2,ic1   ; Is it an OCMP2? Yes continue, else go to ic1.
       ld    A,TAOC2LR    ; Remove OCMP2 interrupt.
       iret
```

```
ic1     btjf  TASR,#ICF1,ic2   ; Is it an ICAP1? Yes continue, else go to ic2.
        call  ta_ic1
        iret

ic2     ld    A,TAIC2LR    ; Remove ICAP2 interrupt.
        iret



    segment 'vectit'

            DC.W not used;FFE0-FFE1h location
            DC.W not used;FFE2-FFE3h location
.i2c_it     DC.W dummy_rt ;FFE4-FFE5h location
            DC.W not used ;FFE6-FFE7h location
            DC.W not used ;FFE8-FFE9h location
            DC.W not used ;FFEA-FFEBh location
            DC.W not used ;FFEC-FFEDh location
.timb_it    DC.W dummy_rt ;FFEE-FFEFh location
            DC.W not used;FFF0-FFF1h location
.tima_it    DC.W tima_rt ;FFF2-FFF3h location
.spi_rt     DC.W dummy_rt ;FFF4-FFF5h location
            DC.W not used;FFF6-FFF7h location
.ext1_it    DC.W dummy_rt ;FFF8-FFF9h location
.ext0_it    DC.W dummy_rt ;FFFA-FFFBh location
.softit     DC.W dummy_rt ;FFFC-FFFDh location
.reset      DC.W main  ;FFFE-FFFFh location

        END

;*** (c) 1997 STMicroelectronics ****************** END OF FILE ****
```

THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS.