

How to Get a Second Asynchronous Interface on a 80C51 Microcontroller Family

1. Introduction

The goal of this application note is to provide a low cost solution for implementing a second asynchronous serial interface.

The standard microcontrollers from the 80C51 family implement only one asynchronous serial interface. However, developers may like to have a second one for debug purpose or for using it in the final application. One efficient solution is described in this application note.

2. Description

When using an 80C51 standard UART in asynchronous mode, the baud rate clock is generated by the Timer 1 programmed to generate a 16x oversampling clock with SMOD bit set to 0. For example a 2400 bauds rate leads to a Timer 1 overflow frequency of 38400 Hz.

The idea used in this application note is to take advantage of this clock by getting usage of the Timer 1 interrupt while keeping other timers free for the application. Timer 1 interrupt is normally disabled when used as baud rate generator.

2.1. Features

- No addition of external hardware.
- No use of extra timer.
- Full Duplex mode.
- Different baud rate in reception and transmission capability.

3. Implementation

3.1. Overview

Character transfer is done in 3 steps:

1. Transmission of a start bit (logic 0).
2. Transmission of 8 or 9 bits depending on the format.
3. Transmission of at least one stop bit (logic 1).

Figure 1 shows the serial transfer of the ASCII character 'A', 0x41 on the bus.

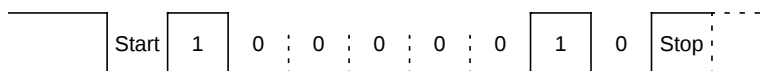


Figure 1. Serial Transfer of ASCII Character 0x41

3.2. Resources

3.2.1. Hardware

Only two I/O pins are needed to implement the second interface: one for the reception input (RxD1) and one for the transmission output (TxD1).

3.2.2. Software

Seven bytes and four bits of data memory are used for configuring and controlling the serial interface.

3.3. Receiver

On each Timer 1 interrupt, RxD1 input is sampled for detecting the 1 to 0 transition indicating the beginning of a start bit. A second sampling is done half a bit later to valid the start bit recognition. Then, sampling is made in the middle of the received bits for the data bits and the stop bit. Figure 2 shows the sampling points position during serial data reception.

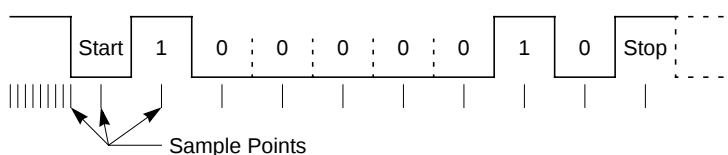


Figure 2. Reception Sampling Points

3.4. Transmitter

The operation of transmission is simpler than reception, it consist only on writing the bit value to the TxD1 line at the transmission rate starting from start bit up to stop bit.

3.5. CPU Load

Table 1 shows the CPU loads for various modes and speeds using a C51 clocked at 11.059 MHz and the hardware UART at 1200 bauds. This CPU loads is divided by 2 while X2 featured microcontroller is used.

The baud rate for the hardware serial interface is limited to 1200 bauds in order to limit the Timer 1 overflow frequency and so the CPU load.

Table 1. CPU Load Versus Traffic

Traffic	Load
No traffic: Idle Mode	41.7%
1200 bauds in Tx or Rx: Half Duplex Mode	50%
1200 bauds in Tx and Rx: Full Duplex Mode	57.4%
9600 bauds in Tx and Rx: Full Duplex Mode	68.5%

In Idle Mode the CPU load is due to the start bit detection in the Timer1 interrupt (polling mode). The number of machine cycles spent in this interrupt may vary from 10 cycles in Idle Mode to 49 cycles in Full Duplex Mode.

3.6. Development tools

Software has been written using standard C51 Intel assembler.

3.7. Demonstration software

The demonstration software simply waits the reception of a character from a host and sends it back to this host. The software is based on three user's routines and an interrupt service routine.

3.7.1. si_install

This is the serial interface installation routine.

It initializes the Timer 1 and enables its interrupt. It also initializes flags that set the receiver (sr_ready= 1) and transmitter (st_ready= 1) as ready.

3.7.2. si_txd

This is the character transmission routine.

Character to send is passed through the accumulator. Format is 8 bits: 7-bit data with parity bit. It is easy not to transfer the parity for an 8-bit data transmission by removing the following code lines from the routine:

```
mov    C,P
mov    ACC.7,C                ; set parity
```

The routine waits for the transmitter becomes ready (st_ready= 1) before sending the new character.

3.7.3. si_rxd

This is the character reception routine.

It loops until a character is received. Character received is returned in the accumulator.

3.7.4. si_interrupt

This is the Timer 1 interrupt service routine.

Reception checks the validity of a character. Flag sr_incom is set when a valid character is available, flag sr_error is set instead if start bit or stop bit is invalid.

3.8. Files

The software is divided into three files:

- MAIN.A51

This file contains the demonstration routine.

```
main                main demonstration routine.
```

- SERIAL.A51

This file contains the serial driver routines.

```
si_install          serial interface installation routine.
```

```
si_txd              character transmission routine.
```

```
si_rxd              character reception routine.
```

```
si_interrupt        Timer 1 interrupt service routine.
```

- SERIAL.INC

This File is the serial interface definition file. It contains the serial pin and speed definitions.

4. Enhancement

The advantage of the proposed implementation is that RxD1 and TxD1 lines can be connected to any spare I/O lines. Its drawback is that in idle mode (no traffic in reception nor in transmission), the CPU load is very significant. One way to avoid this load (due to start bit detection) is to detect the start bit not by polling but by interrupt. The drawback is that RxD1 input must then be connected to either INT0# or INT1# input programmed in low level interrupt for detecting the start bit falling edge.

5. Appendix A

5.1. MAIN.A51

```

; -----
; MAIN.A51
; -----
; Company: Atmel Wireless & Microcontrollers
; -----
; Comments: This file contains the demonstration software for the
;           serial software interface
; -----

$TITLE (Software serial interface)          ; Software serial interface
                                           ; with programmable speed

$NOMOD51
$INCLUDE (reg51.inc)
        RSEG    PROG
NAME    UART_SOFT

; Segment definition
PROG        SEGMENT CODE
BYTE_VAR    SEGMENT DATA
BIT_VAR     SEGMENT BIT
STACK       SEGMENT IDATA

; stack definition
        RSEG    STACK
        DS 10h                                ; 16 bytes Stack

; vectors definition
        CSEG    AT 0000h                      ; Reset vector
        jmp     main
        CSEG    AT 001Bh                      ; Timer 1 vector
        jmp     si_interrupt

; software serial interface demonstration program
; characters received on P1.0 are transmitted on P1.1
; speed is 1200 bauds in reception and transmission
; oscillator frequency = 11.059 MHz

        RSEG    PROG
; Main routine
main:     mov     SP,#STACK-1
        lcall    si_install                    ; serial interface installation
loop:     lcall    si_rxd                      ; wait for a character reception
        lcall    si_txd                      ; send character received
        sjmp     loop

        END

```

5.2. SERIAL.A51

```

;-----
; SERIAL.A51
;-----
; Company: Atmel Wireless & Microcontrollers
;-----
; Comments: This file contains serial software interface routines
;-----

; BITS DEFINITION
        RSEG    BIT_VAR
st_ready: DBIT 1           ; 1 if transmitter ready
sr_ready: DBIT 1           ; 1 if receiver ready
sr_error: DBIT 1           ; 1 if receiver error
sr_incom: DBIT 1           ; 1 if character received

; VARS DEFINITION
        RSEG    BYTE_VAR
        ; Receiver
sr_ch:    DS 1              ; character in reception
sr_count: DS 1              ; internal counter
sr_status: DS 1             ; receiver status
sr_char:  DS 1              ; character received
        ; Transmitter
st_char:  DS 1              ; character in transmission
st_count: DS 1              ; internal counter
st_status: DS 1             ; transmitter status

; SERIAL INTERFACE INSTALLATION ROUTINE
si_install:
        mov     TCON,#40H    ; Timer 1 enabled
        mov     TMOD,#20H    ; 8-bit auto-reload
        mov     TH1,#0E8H    ; 1200 bauds
        mov     SCON,#52H    ; serial port mode 1
        setb    st_ready     ; transmitter ready
        setb    sr_ready     ; receiver ready
        clr     sr_error     ; no error
        setb    ET1          ; Timer 1 interrupt enabled
        setb    EA
        ret

; TRANSMISSION OF A CHARACTER ON TXD1
si_txd:   jnb     st_ready,si_txd
        mov     C,P
        mov     ACC.7,C      ; set parity
        mov     st_char,A    ; character to send
        mov     st_count,#(ST_SPEED/2) ; set counter at 1/2 bit duration
        mov     st_status,#0 ; status initialization
        clr     st_ready     ; start transmission
        ret

; WAIT FOR A RECEIVED CHARACTER ON RXD1
si_rxd:   jnb     sr_incom,si_rxd
        mov     A,sr_char    ; character received
        clr     sr_incom     ; character readed
        ret

; INTERRUPT ROUTINE
si_interrupt:

```

```

        jnb     sr_ready,sr_int1
        ; receiver not busy
        jb      RxD1,st_int0           ; start bit ?
        clr     sr_ready
        mov     sr_count,#(SR_SPEED/2) ; load counter at 1/2 bit duration
        mov     sr_status,#0           ; status initialization
        sjmp    st_int0
sr_int1: djnz    sr_count,st_int0       ; sample point ?
        push    ACC
        push    PSW
        mov     A,sr_status
        jnz     sr_int3
        jb      RxD1,sr_frame          ; start bit OK (0) ?
sr_int2: inc     sr_status
        mov     sr_count,#SR_SPEED
        sjmp    sr_int5
sr_frame: setb   sr_ready
        setb    sr_error               ; receiver error
        sjmp    sr_int5
sr_int3: cjne    A,#9,$+3              ; 8 bits + stop bit
        jnc     sr_int4
        mov     C,RxD1                 ; bit sampling
        mov     A,sr_ch
        rrc     A
        mov     sr_ch,A
        sjmp    sr_int2
sr_int4: jnb     RxD1,sr_frame          ; stop bit OK (1) ?
        mov     sr_char,sr_ch          ; save character received
        setb    sr_ready
        setb    sr_incom               ; 1 char. received
sr_int5: pop     PSW
        pop     ACC

st_int0: ; transmission part
        jb      st_ready,st_int5       ; exit if no character to send
        djnz    st_count,st_int5       ; sample point ?
        push    ACC
        push    PSW
        mov     A,st_status
        jnz     st_int1                ; start ?
        clr     TxD1                   ; set start bit
        mov     st_count,#ST_SPEED
        inc     st_status
        sjmp    st_int4
st_int1: cjne    A,#9,$+3              ; 8 bits + stop bit
        jnc     st_int2
        mov     A,st_char
        rrc     A                       ; bit to send in carry
        mov     st_char,A
        mov     TxD1,C                 ; transmission of bit
        mov     st_count,#ST_SPEED     ; counter initialization
        inc     st_status
        sjmp    st_int4
st_int2: cjne    A,#10,st_int3         ; end of character ?
        setb    st_ready
        sjmp    st_int4
st_int3: setb    TxD1                  ; set stop bit
        mov     st_count,#ST_SPEED
        inc     st_status

```

```
st_int4:  pop    PSW
          pop    ACC
st_int5:  reti
```

END

5.2.1. SERIAL.INC

```
; -----
; SERIAL.INC
; -----
; Company: Atmel Wireless & Microcontrollers
; -----
; Comments: This file contains the serial software interface definitions
; -----

; CONSTANT DEFINITION

; pin definition
RxD1      EQU    P1.0                ; reception pin
TxD1      EQU    P1.1                ; transmission pin

; serial baud rate depends on the hardware baud rate:
; for example with hardware baud rate equal to 1200 bauds,
; 32 value for SX_SPEED constant give 1200 bauds
; 16 value for SX_SPEED constant give 2400 bauds
; 8 value for SX_SPEED constant give 48000 bauds
; 4 value for SX_SPEED constant give 9600 bauds

SR_SPEED   EQU    32                  ; reception at 1200 bauds
ST_SPEED   EQU    32                  ; transmission at 1200 bauds
```