MICROPROCESSOR BASED SYSTEMS

CMP 410

LECTURE 7

https://AssemSite8.wix.com/site8
**The Bit Rate & Baud Rate**

**Bit Rate** = \((\text{No. of sending or receiving Bits})/\text{Sec}\)

**Baud Rate** = \((\text{No. of frames or Patterns or Symbols})/\text{Sec}\)

*frame includes specific No. of bits*
Calculate the baud rate, if TH1 = F4h and the bit SMOD (PCON.7) has its initial value, beside $f_{osc} = 11.0592 \text{ MHz}$.

SMOD is MSB of the register PCON and initially = 0

Baud Rate_{serial mode 1} = 2^{SMOD} \times \frac{f_{osc}}{(32 \times 12)[256 - TH1]}

Example T1 mode 2

Baud Rate_{serial mode 1} = 2^0 \times \frac{11.0592 \text{ MHz}}{384[256-244]}

= 2400 \text{ [symbols/sec]}
Generating the baud rate’s clock pulses

\[ Baud \text{ Rate}_{mode \ 1} = 2^{SMOD} \times \frac{f_{osc}}{\left(12 \times 32\right)[256 - TH1]} \]

- Clock divider (/2)
- Clock divider (/16)
- Clock divider (/256 − TH1)
- Clock divider (/12)
- Internal oscillator

\( SMOD = 0 \)

\( SMOD = 1 \)
The Serial Control register of 8051

<table>
<thead>
<tr>
<th>Modes</th>
<th>SM2</th>
<th>REN</th>
<th>TB8</th>
<th>RB8</th>
<th>TI</th>
<th>RI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SM0, SM1** for controlling the serial mode according to the following table:

<table>
<thead>
<tr>
<th>Mode</th>
<th>SM0</th>
<th>SM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **SM2** ... It uses for serial communication as master and slaves microcontrollers via serial ports.
<table>
<thead>
<tr>
<th>Modes</th>
<th>Control bits</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM0</td>
<td>SM2</td>
<td>TB8</td>
</tr>
<tr>
<td></td>
<td>REN</td>
<td>RB8</td>
</tr>
<tr>
<td>SM1</td>
<td></td>
<td>TI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RI</td>
</tr>
</tbody>
</table>

- **RB8** ... It is the 9\(^{th}\) received data bit in serial mode 2,3
- **TB8** ... It is the 9\(^{th}\) transmitted data bit in serial mode 2,3
- **REN** ... It is the receiving enable bit.
- **RI** ... "Receive Interrupt" flag. It automatically sets to '1' by HW when the SBUF completely received all data bit.

- **TI** ... "Transmit Interrupt" flag. It automatically sets to '1' by HW when the SBUF completely transmitted all data bit.
The Serial UART Transmitting in MCS-51

```assembly
MOV SBUF, A
JNB TI, $1
CLR TI
```

Wait until all data-bits passed to TXD

```
SBUF
TXD-pin
```

The Serial UART Receiving in MCS-51

```assembly
JNB RI, $
MOV A, SBUF
CLR RI
```

Wait until all data-bits passed via RXD

```
SBUF
RXD-pin
```

Baud rate clocks
Example

Draw block diagram and write assembly code (without ISR) for receiving characters from a computer to AT89C51 via serial port with baud rate 2400 (TH1= F4h) and drive them to P0. Write a comment for each instruction.

![Block Diagram](image)

<table>
<thead>
<tr>
<th>Serial Control (SCON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modes: SM0, SM1</td>
</tr>
<tr>
<td>SM2, REN, TB8, RB8, TI, RI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modes</th>
<th>SM2</th>
<th>REN</th>
<th>TB8</th>
<th>RB8</th>
<th>TI</th>
<th>RI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SM1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
INCLUDE 89C51.mc
ORG 0H
SJMP MAIN

ORG 30H
MAIN:

MOV TH1, #F4H ; Adjusting baud rate 2400
MOV TMOD, #20H ; Timer 1 in Mode 2
MOV SCON, #50H ; Serial Mode 1
SETB TR1 ; Run Timer 1

;------------
Wait_Ch:;

JNB RI, Wait_Ch ; Wait for complete receiving character
MOV P0, SBUF ; Load received character into Port 0
CLR RI ; Clear Receive Interrupt Flag
SJMP Wait_Ch ; Loop

END
MANY DEVICES AND KITS ARE SUPPLEMENTED WITH SPI

CONTROL AREA NETWORK (CAN)

TRANSCEIVER

Microprocessor 2017-18 11
## UART vs SPI (serial-peripheral-interface)

<table>
<thead>
<tr>
<th>UART</th>
<th>SPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-wire serial bus (Tx &amp; RX)</td>
<td>four-wire serial bus (SCK, MISO, MOSI, SS)</td>
</tr>
<tr>
<td>Used in Asynchronous serial communication.</td>
<td>Used in Synchronous serial comm.</td>
</tr>
<tr>
<td>It Built in AT89C51</td>
<td>it doesn’t built in AT89C51 (but built in other members of family)</td>
</tr>
<tr>
<td>Interfacing with single unit at a time</td>
<td>Interfacing with multiple units at a time</td>
</tr>
<tr>
<td>Used in Half and Full duplex</td>
<td>Used in Half and Full duplex</td>
</tr>
</tbody>
</table>
The SPI is called the *four-wire* serial bus because it has four terminals

1) SCK …Serial clock from master
2) MOSI …Master out Slave in
3) MISO …Master in Slave out
4) SS …Slave Selection
The connection between Master and multi Slaves modules via SPI
Bit stream from master to slave

Master to Slave = (11001010)
Bit stream from slave to master

Slave to Master = (01100010)
CREATING SPI PORT TO AT89C51
A SPI-wireless module connected with 8051 as following block diagram. It receives data frames from remote control and send them to AT89c51 via SPI to open/close a gate according to attached flowchart. Write assembly program to do this design.
Initialization

Enable slave

Read one frame

Open gate

= A0h

No

Close gate

= F0h

No

Nothing
org 0
SJMP main
org 30h
main:
SCK_out EQU P2.0
MISO_in EQU P2.1
SS_out EQU P2.3
Gate EQU P2.4

;---------------------
CLR SCK_out ;(SCK = 0)
CLR Gate ;

SETB MISO_in ;to accept inputs

;---------------------
again:
The loop of main program

```
again:
CLR SS_out ; enable Slave (SS = 0)
MOV R7, #8
LOOP:
ACALL READ_BIT
DJNZ R7, LOOP
ACALL Compare
SJMP again
;-------------------------
Compare:
```
The subroutines

```assembly
Compare:
CJNE A, # A0h, next_1
SETB Gate ; open gate
RET
next_1:
CJNE A, # F0h, next_2
CLR Gate ; Close gate
next_2: RET
;-----------------------------

READ_BIT:
SETB SCK_out
CLR SCK_out
;-----------------------------

MOV C, MISO_in ; receive bit in CY
RLC A ; insert received bit into A
RET
```

Positive edge trigger
If the previous example is modified as following block diagram and flowchart, write its program code.
Initialization

Enable slave

Read one pattern

Send AAh via MOSI

Send FFh via MOSI

Open gate

Close gate

= A0h

= F0h

No

No

Microprocessor 2017-18
The Interrupt in AT89c51

- **program interrupt** is a command signal (internal or external) to a CPU to leave its main program (after execute the current instruction) and perform subprogram called interrupt service routine (ISR).
- When CPU is finished the ISR, it returns back to the main program(*return address*).
The Interrupt sources in AT89c51

There are 5 sources of interrupt

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Default Interrupt priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>External 0 (via INT0)</td>
<td>1</td>
</tr>
<tr>
<td>Timer 0</td>
<td>2</td>
</tr>
<tr>
<td>External 1 (via INT1)</td>
<td>3</td>
</tr>
<tr>
<td>Timer 1</td>
<td>4</td>
</tr>
<tr>
<td>Serial port</td>
<td>5</td>
</tr>
</tbody>
</table>
**Interrupt Enable (IE)**

<table>
<thead>
<tr>
<th>EA</th>
<th>-</th>
<th>-</th>
<th>ES</th>
<th>ET1</th>
<th>EX1</th>
<th>ET0</th>
<th>EX0</th>
</tr>
</thead>
</table>

**EA** ... The global enable/disable interrupt.

**EX0** ... The individual enable/disable ext. 0 interrupt.

**ET0** ... The individual enable/disable T0 interrupt.

**EX1** ... The individual enable/disable ext. 1 interrupt.

**ET1** ... The individual enable/disable T1 interrupt.

**ES** ... The individual enable/disable serial port interrupt.
The Interrupt organization of AT89c51

Register TCON

INT0 (P3.2)

IT0

0

1

IE0

Register IE

INT1 (P3.3)

IT1

0

1

IE1

EF

Timer 0

TF0

Timer 1

TF1

UART

TI

RI

Interrupt Signal To CPU

EA

Timer Control (TCON)

<table>
<thead>
<tr>
<th>Bits for Timers</th>
<th>IE1</th>
<th>IT1</th>
<th>IE0</th>
<th>IT0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PX0</td>
<td>PT0</td>
<td>PX1</td>
<td>PT1</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>PX0</td>
<td>PT0</td>
<td>PX1</td>
<td>PT1</td>
</tr>
</tbody>
</table>

**PX0** ... The ext. 0 interrupt priority.

**PT0** ... The Timer0 interrupt priority.

**PX1** ... The ext. 1 interrupt priority.

**PT1** ... The Timer1 interrupt priority.

**PS** ... The serial port interrupt priority.
# Interrupt Source, Flags, Vectors

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Interrupt Flags</th>
<th>Interrupt Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>External 0</td>
<td>IE0</td>
<td>03 h</td>
</tr>
<tr>
<td>Timer 0</td>
<td>TF0</td>
<td>0B h</td>
</tr>
<tr>
<td>External 1</td>
<td>IE1</td>
<td>13 h</td>
</tr>
<tr>
<td>Timer 1</td>
<td>TF1</td>
<td>1B h</td>
</tr>
<tr>
<td>Serial port</td>
<td>TI, RI</td>
<td>23 h</td>
</tr>
</tbody>
</table>
Layout assembly program with all interrupt vectors

```assembly
ORG 0
JMP MAIN

ORG 03h
:-------- ISR for external 0 interrupt
RETI

ORG 0Bh
:-------- ISR for timer 0 interrupt
RETI

ORG 13h
:-------- ISR for external 1 Interrupt
RETI

ORG 1Bh
:-------- ISR for timer 1 interrupt
RETI

ORG 23h
:-------- ISR for UART interrupt
RETI

:-------- Main instructions

ORG 30h
MAIN:

:-------- Main instructions

JMP MAIN
```
Example

Draw block diagram and write assembly code using **ISR** for an embedded system based on AT89C55. It receives **data-bytes** from **GPS** module via its UART port with baud rate 9600 (TH1=FDh) and send them to P0 **during** receiving data-bytes from **transceiver** module via P1 and send it to P2 simultaneously.
INCLUDE 8051.mc

ORG 00H
LJMP MAIN

ORG 23H
; starting address FOR the ISR(serial)
CLR RI
MOV P0, SBUF
RETI ; returns from the interrupt

ORG 30H
; starting address FOR MAIN program

MAIN:

MOV P1, #FFH ; P1 is input port
MOV TMOD, #20H ; timer 1 is auto reload (mode 2)
MOV TH1, #FDH ; adjusting 9600 baud rate
MOV SCON, #50H ; 8-bit, 1 start, 1 stop, ren enabled
SETB EA ; enabling the global interrupt control
SETB ES ; enabling serial interrupt control
SETB TR1 ; starting timer 1

Again: MOV P2, P1
SJMP Again ; infinite loop
Example

Design embedded system (block diagram & assembly code) for crane based on AT89s8252 using ISR to control its 2 stepper-motors (connected with P2). The 2 motors rotate one step clockwise when they receive a pulse from INT0 and rotate one step anticlockwise when they receive a pulse from INT1.

INT0

INT1

P2

Motors Drivers

M1

M2
Stepper Motors’ patterns

<table>
<thead>
<tr>
<th>Motor2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Motor1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P2.7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>P2.3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2.6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P2.2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2.5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>P2.1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P2.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>P2.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Microprocessor 2017-18
ORG 0H
SJMP MAIN
ORG 03H
MOV A,R0
RR A
MOV R0, A
MOV P2,R0
RETI

;-----------------
ORG 13H
MOV A,R0
RL A
MOV R0, A
MOV P2,R0
RETI

;-----------------
ORG 30H
MAIN:
MOV P0, #FFH
MOV R0, #88H; OR 44H OR 22H OR 11H
MOV P2,R0
SETB EA
SETB EX0
SETB EX1
SETB IT0
SETB IT1
AGAIN:
MOV P1, P0
SJMP AGAIN
END

---

<table>
<thead>
<tr>
<th>Timer Control (TCON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits for Timers</td>
</tr>
<tr>
<td>TF1</td>
</tr>
</tbody>
</table>

---

Diagram of interrupt and timer control.