Introduction to USAR Lab

Dr. Farid Farahmand
PIC Serial Interface
PIC Out for USART
EUSART RECEIVE BLOCK DIAGRAM

Setting the baud rate

Flag determines if buffer is empty or full

8 bits or 9 bits transmission
EUSART TRANSMIT BLOCK DIAGRAM

Flag determines if buffer is empty or full

Setting the baud rate

8 bits or 9 bits transmission
### Registers Associated with Asynchronous Transmission

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INTOIE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTOIF</td>
<td>RBIF</td>
</tr>
<tr>
<td>PIR1</td>
<td>PSPIF(^1)</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
</tr>
<tr>
<td>PIE1</td>
<td>PSPIE(^1)</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
</tr>
<tr>
<td>IPR1</td>
<td>PSPIP(^1)</td>
<td>ADIP</td>
<td>RCIP</td>
<td>TXIP</td>
<td>SSPIP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
</tr>
<tr>
<td>TXREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
</tr>
<tr>
<td>BAUDCON</td>
<td>ABDVOF</td>
<td>RCIDL</td>
<td>DTRXP</td>
<td>CKTXP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
</tr>
<tr>
<td>SPBRGH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) Standard mode
REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
</tr>
<tr>
<td>PIR1</td>
<td>PSPIF&lt;sup&gt;1&lt;/sup&gt;</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
</tr>
<tr>
<td>PIE1</td>
<td>PSPIE&lt;sup&gt;1&lt;/sup&gt;</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
</tr>
<tr>
<td>IPR1</td>
<td>PSPIP&lt;sup&gt;1&lt;/sup&gt;</td>
<td>ADIP</td>
<td>RCIP</td>
<td>TXIP</td>
<td>SSPIP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
</tr>
<tr>
<td>RCREG</td>
<td>EUSART Receive Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
</tr>
<tr>
<td>BAUDCON</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>DTRXP</td>
<td>CKTXP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
</tr>
<tr>
<td>SPBRGH</td>
<td>EUSART Baud Rate Generator Register, High Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRG</td>
<td>EUSART Baud Rate Generator Register, Low Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example:

- Assuming 9600 baud rate, Asynch, Clock frequency is 10 MHZ, 8 bit character
- What should we write into register SPBRG register?
- We assume: SYNC = 0, BRGH = 0, BRG16 = 0
  - Thus, baud rate = $F_{cso}/[64(n+1)] \rightarrow n=15$
Example

• This program allows to print characters on a remote PC terminal
• The clock setting and interrupts are shown below.

```assembly
;; This is your actual assembly code....
Main:
 ; changing the variable in assembly

  MOVLL B'00100000'
  MOVWF TXSTA
  MOVLL D'12'; Based on 8 MHz clock defined in .c program
  MOVWF SPERG
  MOVLL B'00110000'; Bits must be inverted
  MOVWF BAUDCON
  BCF TRISC, TX
  BSE RCSTA, SPEN

OVER
  MOVLL A'H'
  CALL TRANSMIT ; Go to the subroutine
  MOVLL A'E'
  CALL TRANSMIT
  MOVLL A'L'
  CALL TRANSMIT
  MOVLL A'L'
  CALL TRANSMIT
  MOVLL A'O'
  CALL TRANSMIT
  MOVLL A'O'
  CALL TRANSMIT
  MOVLL A'
  CALL TRANSMIT
  MOVLL A'
  CALL TRANSMIT
  BRF OVER

;-----------------------------------------------
TRANSMIT ; Subroutine to transmit

SSS
  BTFS PI1, TXIF ; The Interrupts are enabled in the .c program
  BRA SSS
  MOVWF TXREG

return
```

```plaintext
OSCCON = 0x60;  // IRCFx = 110 - Clock 8MHz
OSCTUNEbits.PLLEN = 0;  // x4 PLL disabled
PIE1bits.TXIE = 1;
PIE1bits.RCIE = 1;
INTCONbits.PEIE = 1;  // Peripheral interrupts
INTCONbits.CIE = 1;  // Interrupting enabled.
```
C- Version
Transmitting Characters

```c
const rom char trans_string[] = {'\n', 'P', 'l', 'e', 'a', 's', 'E', 'n', 't', 'e', 'r', '
', 'o', 'u', 'm', 'b', 'e', 'r', 'p';

while(1)
{
    for(mycount=0; mycount < N; mycount++) //Goes from 0 to N-1
    {
        mychar = trans_string[mycount];
        //printf( "First Char = %c, %d \n", trans_string[mycount], mycount); //for test!

        do
        {
            //nothing really!
            PORTbits.RDO = ~PORTbits.RDO; //Toggle the bit - this is equivalent
            // to clock speed (8MHz/4)/4=250 KHz
        }while (PIR1bits.TXIF == 0);
        TXREG = mychar;
    } //end of for loop
}
```
Interfacing

- Connect RC7 and GND pins on the board to the DB9 connector as shown below.
- Note that in general we SHOULD use something like Maxim's MAX232 in order to ensure voltage compatibility between the PIC and the RS232 or the terminal.
- However, it turns out that by INVERTING polarity of the signals on TX and RX pins of USART, it is possible to interface to the terminal.
- We achieved this through setting the BAUDCON register.
Asynchronous Transmission

Character Frame

Example of an “A” followed by a “E”

A = 0x41

B = 0x45
Synchronous Transmission

Clock

Data

A = 0x41
B = 0x42
The USART interfaced to a standard DB9 connector for RS-232C data
Terminal

• Download a PC terminal software such as Hyper Terminal or RealTerm (http://sourceforge.net/projects/realterm/)
• If you only have a USB port you may need a USB/Serial Cable and driver
• Set the Hyper Terminal to 9600, N,1,0
• Power up your board and run the program in DEBUG mode.
• You should see the characters displayed on the terminal

Make sure your PICKIT is connected to ICSP connector at all the time. The interface to the PC is via pin 3 (GND) and pin 6 (TX) of P2 connector on the board.
Output Displayed Using RealTerm (pay attention to the settings)
Baud Rate

- Note that by probing TX pin (RC7) we can ensure the baud rate is set properly.
- The following is a sample calculation for determining the SPBRG value if the clock is 16 MHz.
- Note that for 4 MHz clock the actual baud rate is 9615 bps, and error of 0.16 percent which is tolerable.
- It is also possible to use `#pragma config FOSC = HS` to generate 4MHz clock.
Baud Rate Measurement
(about 9600 bps)

Transmitted data
Testing

- You can test your output in the Simulation mode using the SIM Usart1 feature.
We have designed a LabVIEW VI to receive the transmitted data.
In this case HELlx is transmitted. X is an 8-bit random value.
All the received values are plotted and saved in a file.
See the VI in the next slide.
LabVIEW (2)
LabVIEW (3) Practice

• A few interesting changes to the LabVIEW program can be changed:
  – Create an alarm button such that if a value greater than 32 was detected, a RED LED is turned on and the buzzer is activated.
  – Count and display the number of inputs recorded in each session.
  – In LabVIEW go to Tools ➔ Build Application (EXE) and create an executable file. Can you use this file on a PC that does not have LabVIEW software?
  – In LabVIEW go to Tools ➔ Web Publishing and create a web-version of the program. You should be able to remotely monitor the received values. You can save the file as MONITOR. You will also need to download the RUNTIME program to interface with the VI remotely.
Receiving Bytes From PC

• We need to change the levels.
• The PIC does not have sufficient forgivenes.
This is the signal from PC’s RS232