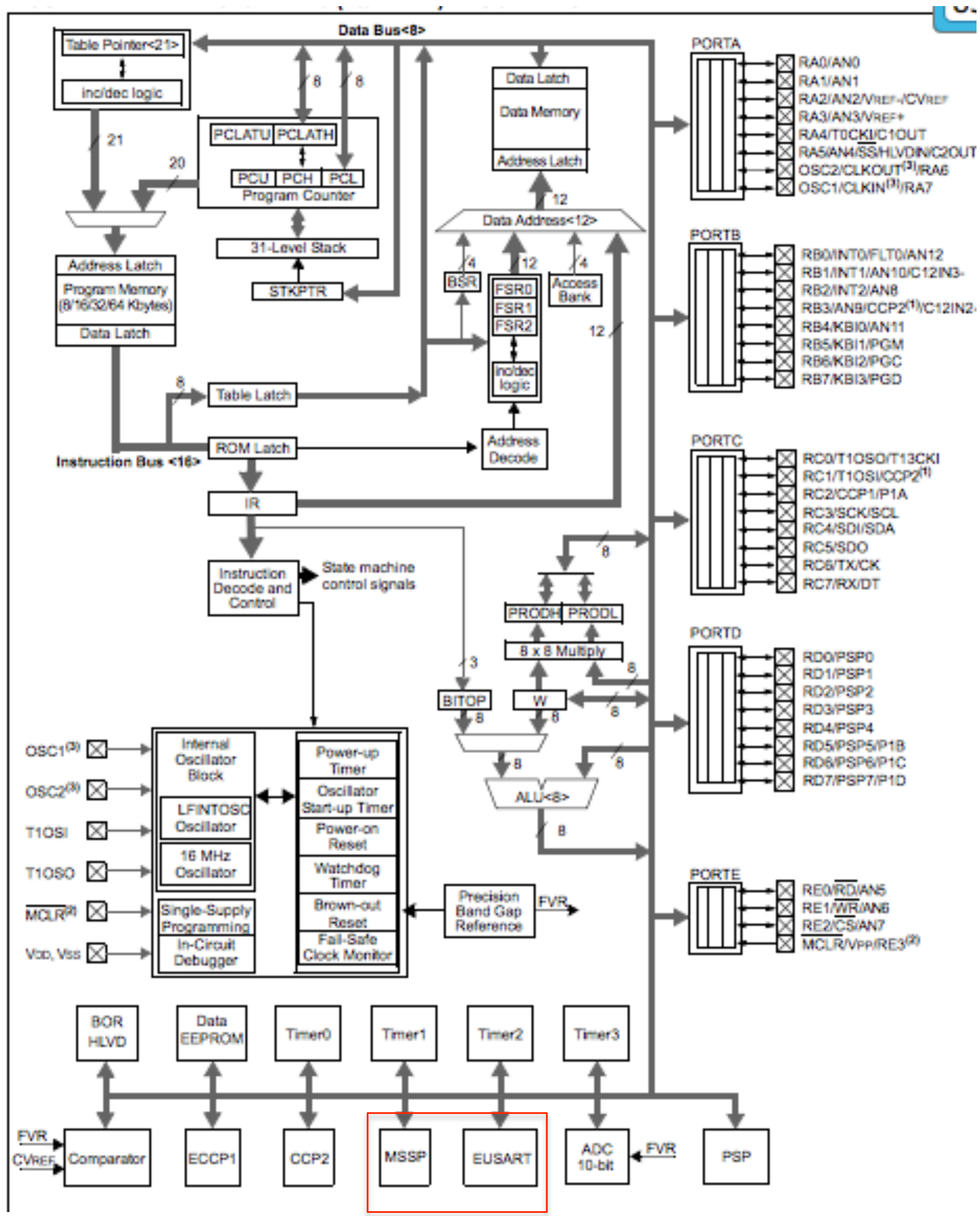
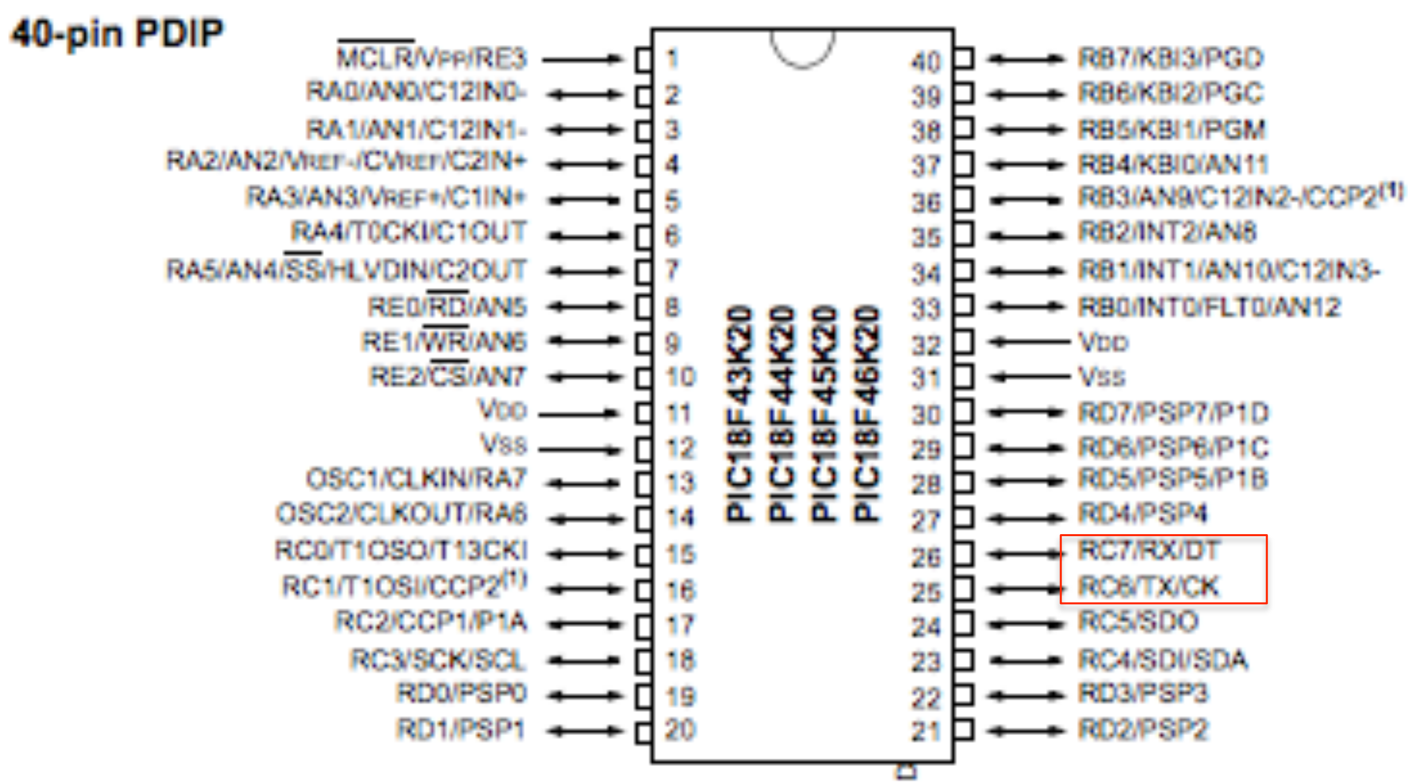
Introduction to USAR Lab

Dr. Farid Farahmand

PIC Serial Interface

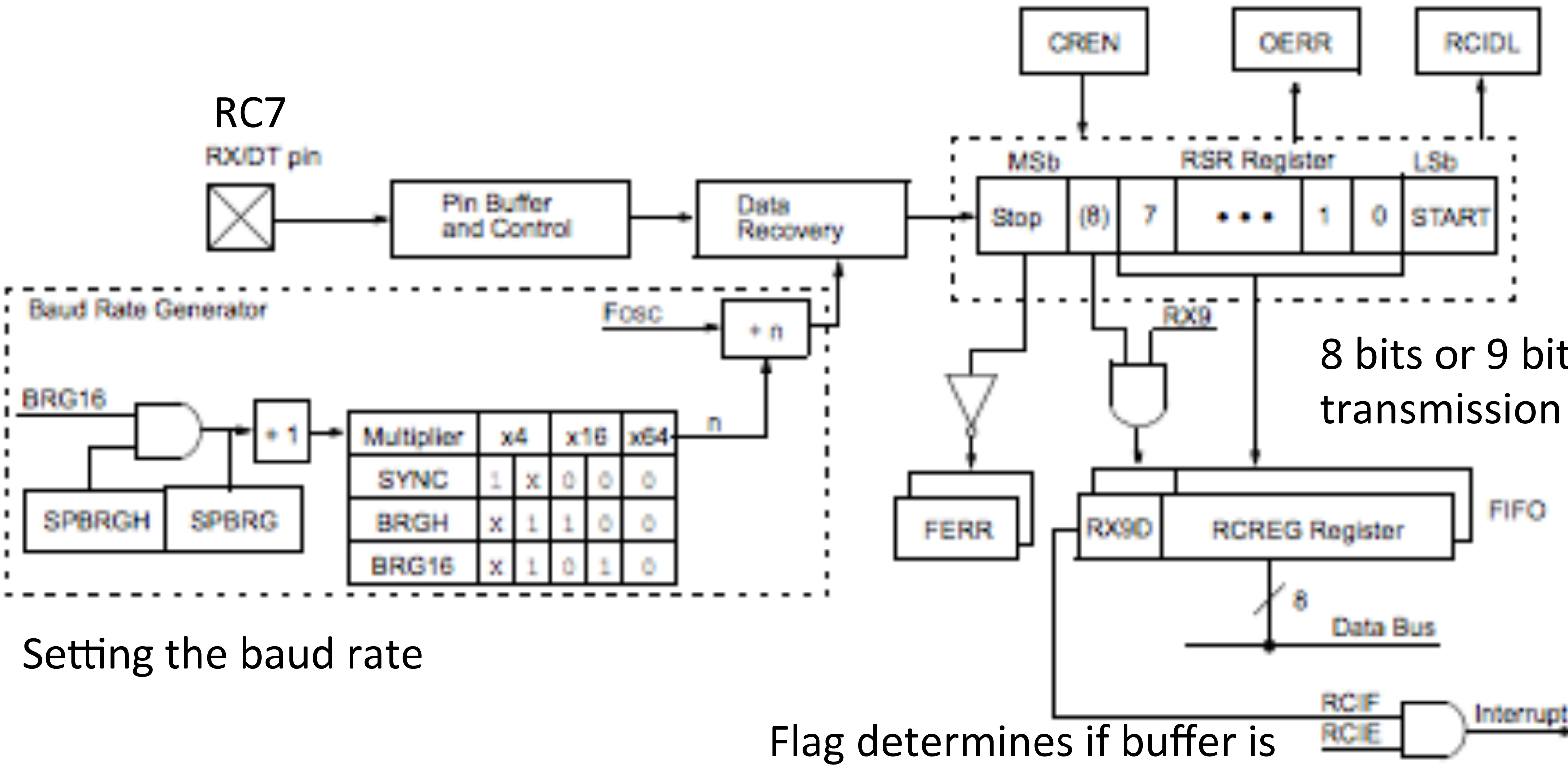


PIC Out for USART



EUSART RECEIVE BLOCK DIAGRAM

RC7



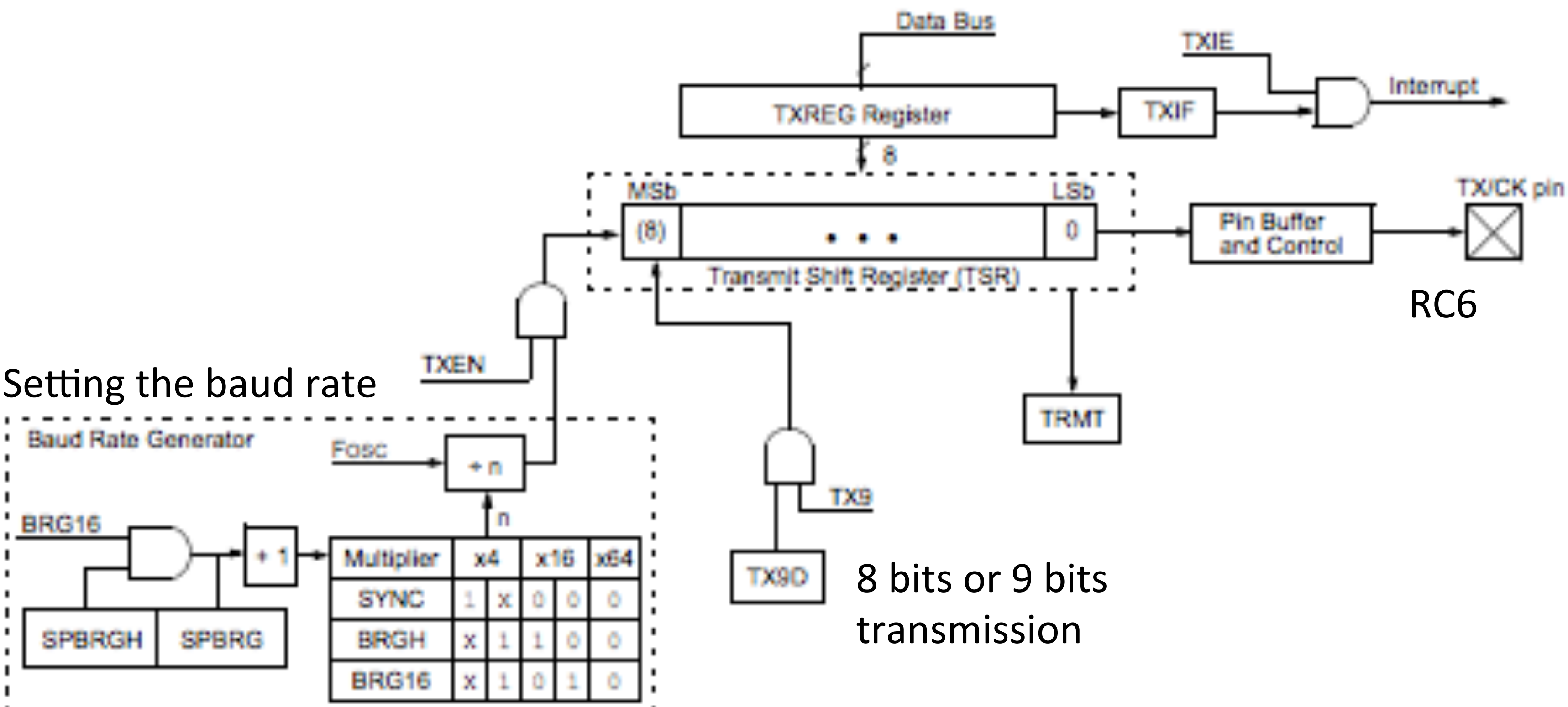
8 bits or 9 bits transmission

Setting the baud rate

Flag determines if buffer is empty or full

EUSART TRANSMIT BLOCK DIAGRAM

Flag determines if buffer is empty or full



Multiplier	x4		x16		x64
SYNC	1	x	0	0	0
BRGH	x	1	1	0	0
BRG16	x	1	0	1	0

RC6

Setting the baud rate

8 bits or 9 bits transmission

REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

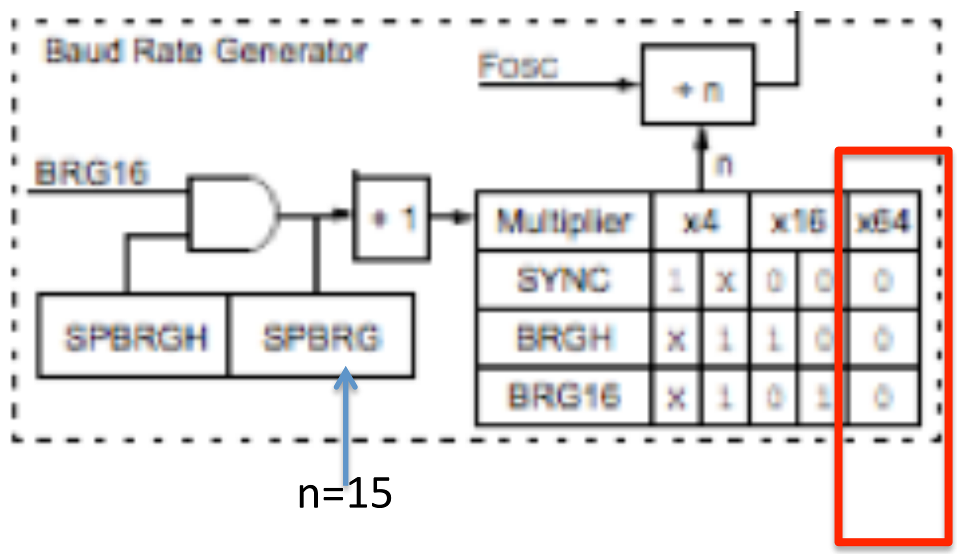
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREG	EUSART Transmit Register							
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN
SPBRGH	EUSART Baud Rate Generator Register, High Byte							
SPBRG	EUSART Baud Rate Generator Register, Low Byte							

REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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

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 = Fcso/[64(n+1)]→n=15



Example

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

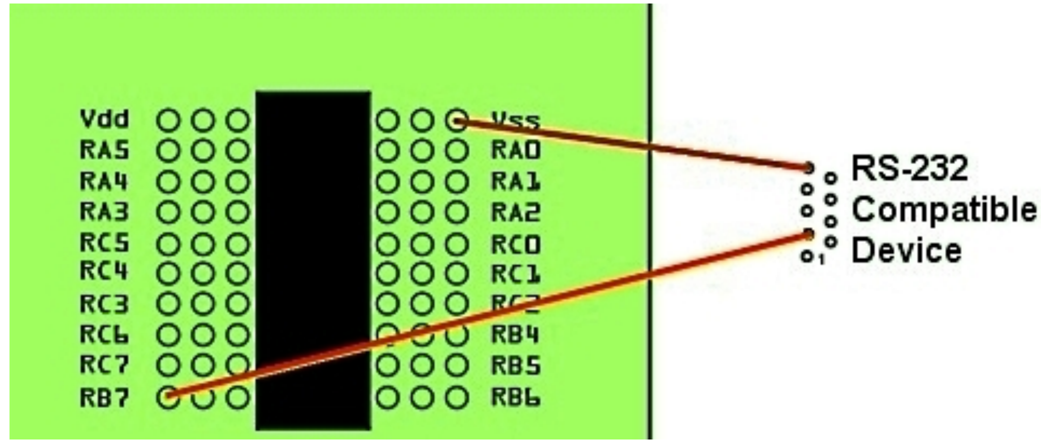
```
;; This is your actual assembly code....
Main:
    ; changing the variable in assembly

    MOVLW   B'00100000'
    MOVWF   TXSTA
    MOVLW   D'12'  ;Based on 8 MHz clock defined in .c program
    MOVWF   SPBRG
    MOVLW   B'00110000' ; Bits must be inverted
    MOVWF   BAUDCON
    BCF     TRISC, TX
    BSF     RCSTA, SPEN
OVER
    MOVLW   A'H'
    CALL    TRANSMIT ; Go to the subroutine
    MOVLW   A'E'
    CALL    TRANSMIT
    MOVLW   A'L'
    CALL    TRANSMIT
    MOVLW   A'L'
    CALL    TRANSMIT
    MOVLW   A'O'
    CALL    TRANSMIT
    MOVLW   A' '
    CALL    TRANSMIT
    BRA     OVER
;---------------------------------------
TRANSMIT ; Subroutine to transmit
SSS
    BTFSS   PIR1,TXIF ;The Interrupts are enabled in the .c progr
    BRA     SSS
    MOVWF   TXREG
return
```

```
OSCCON = 0x60;            // IRCFx = 110 - Clock 8MHz
OSCTUNEbits.PLLEN = 0;    // x4 PLL disabled

PIE1bits.TXIE = 1;
PIE1bits.RCIE = 1;
INTCONbits.PEIE = 1;          // Peripheral interrupts
INTCONbits.GIE = 1;           // Interrupting enabled.
```

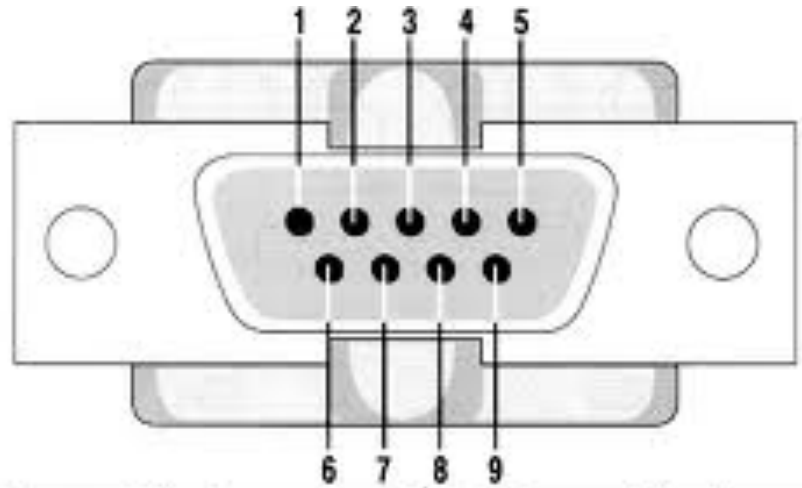
C- Version
Transmitting Characters

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

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!
            PORTDbits.RD0 = ~PORTDbits.RD0; //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

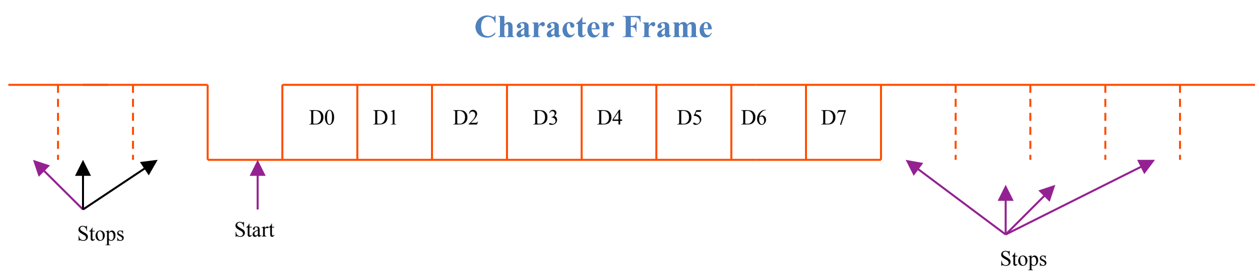


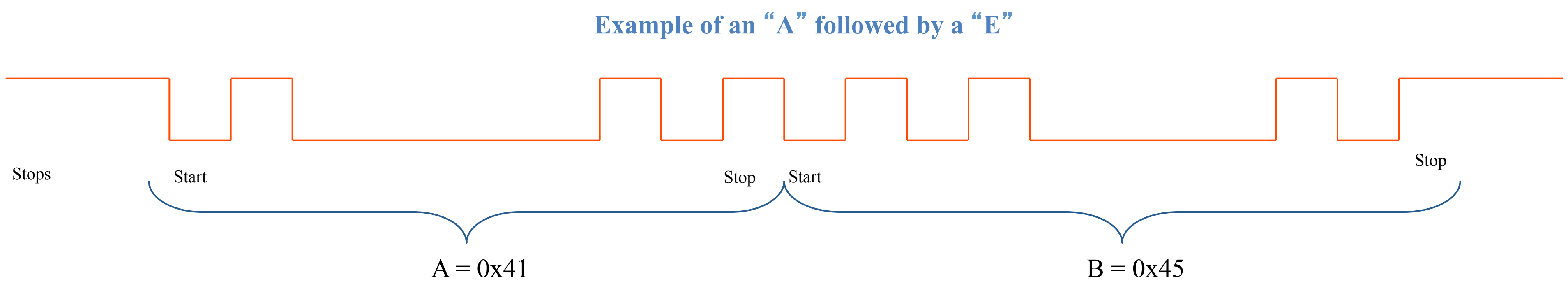
- Connect RC7 and GND pins on the board to the DB9 connector as shown below
- Note that in general we SHOULD use something like Maxim's MAX232 in order to ensure voltage compatibility between the PIC and the RS232 or the terminal
- However, it turns out that by INVERTING polarity of the signals on TX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



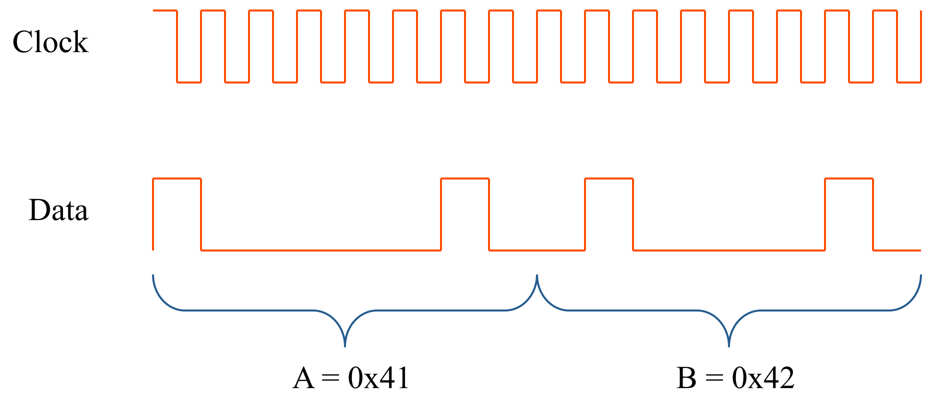
Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Asynchronous Transmission

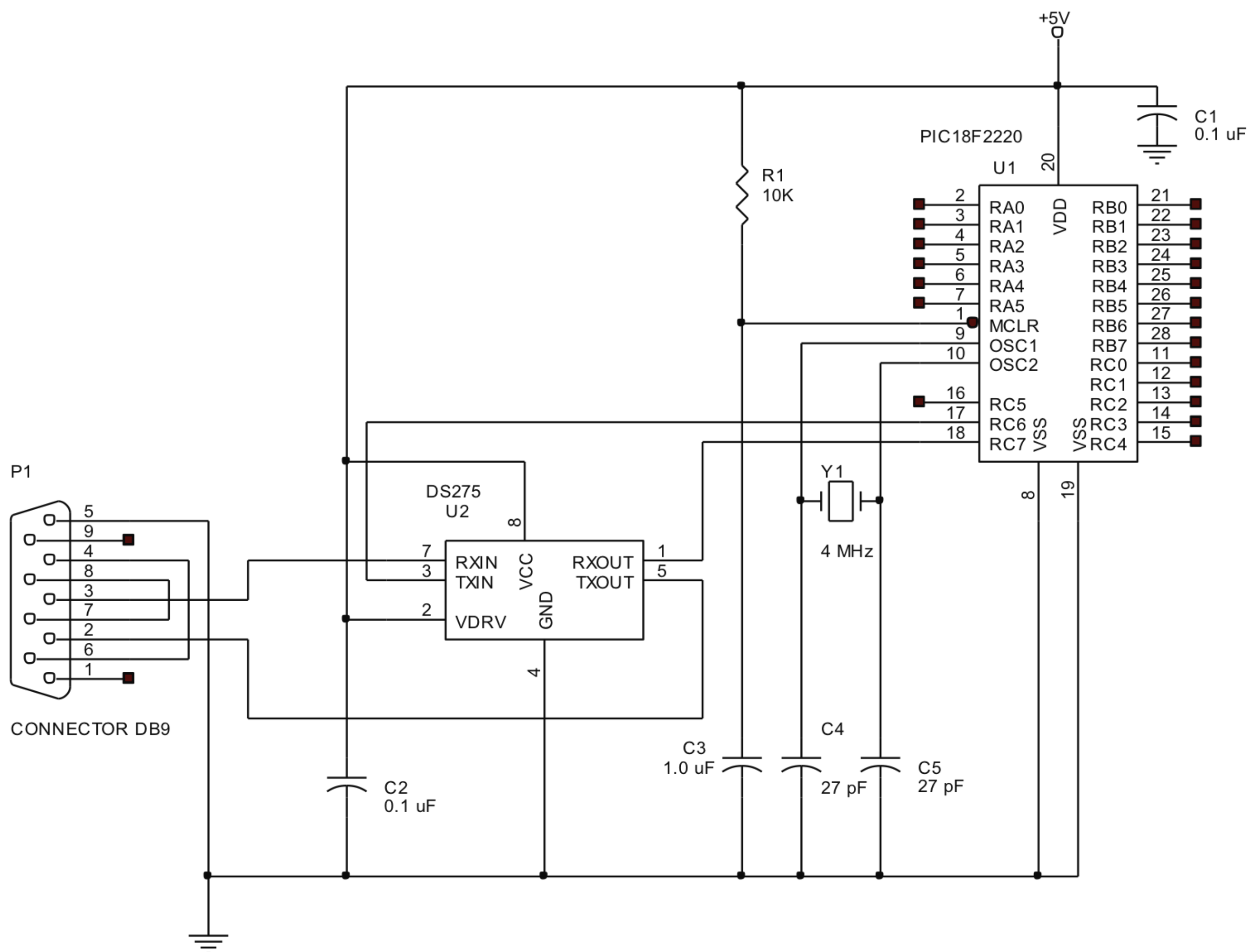




Synchronous Transmission



The USART interfaced to a standard DB9 connector for RS-232C data

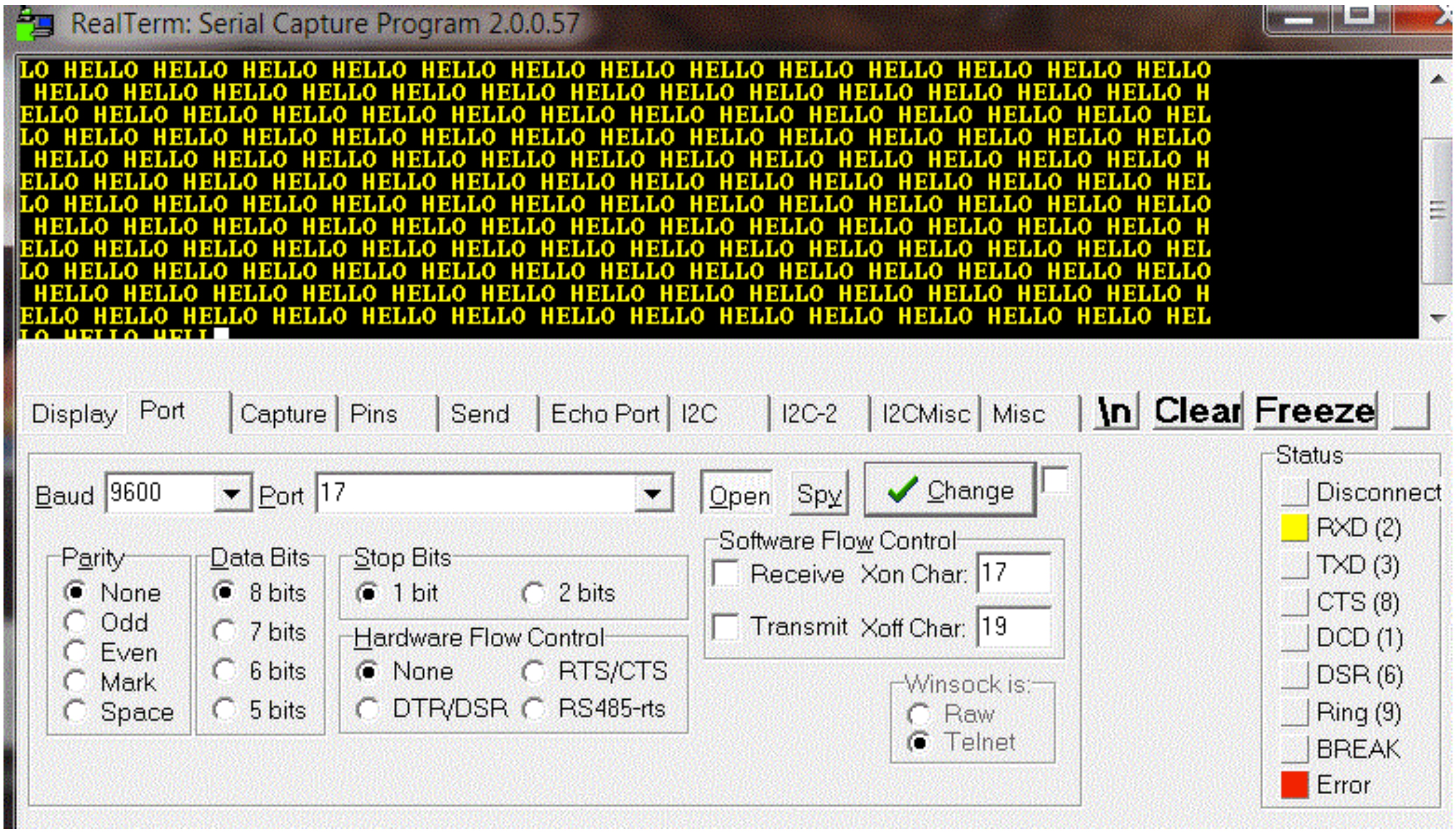


Terminal

- 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 si also possible to use #pragma config FOSC = HS to generate 4MHz clock.

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$\text{Desired Baud Rate} = \frac{F_{OSC}}{64([SPBRGH:SPBRG] + 1)}$$

Solving for SPBRGH:SPBRG:

$$X = \frac{\frac{F_{OSC}}{\text{Desired Baud Rate}}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

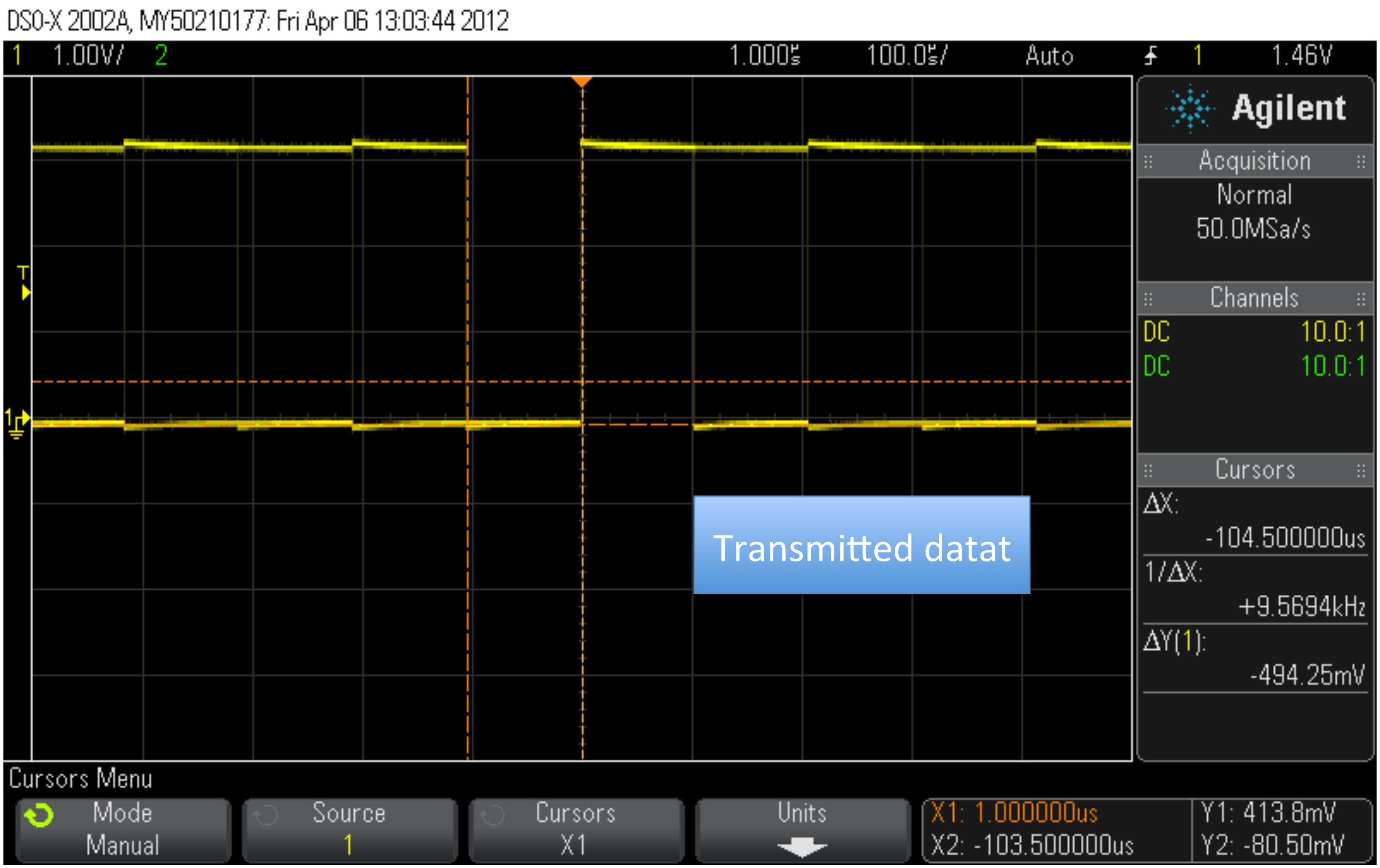
$$\text{Calculated Baud Rate} = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$\text{Error} = \frac{\text{Calc. Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

Baud Rate Measurement (about 9600 bps)



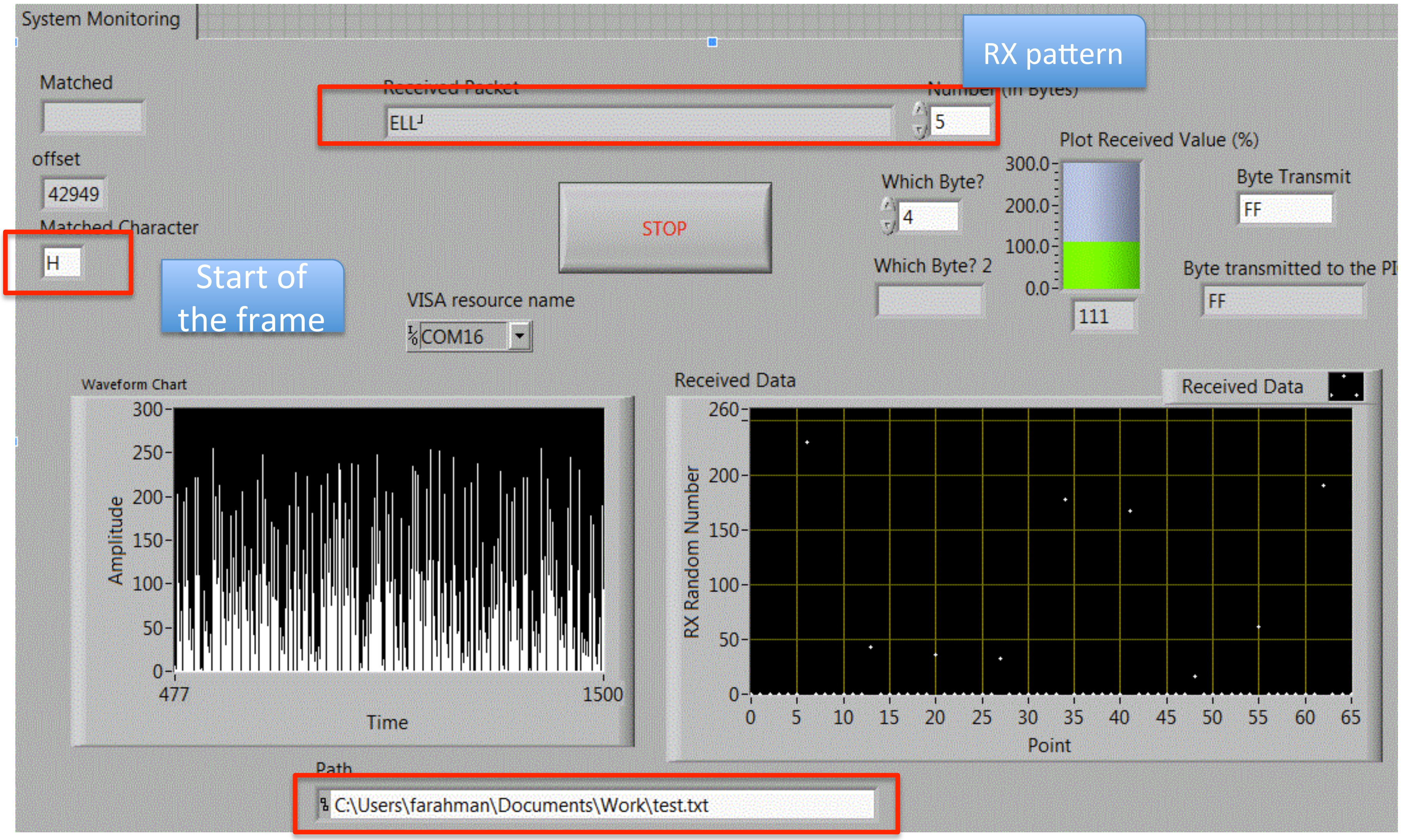
Testing

- You can test your output in the Simulation mode using the SIM Usart1 feature

LabVIEW (1)

- 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)



System Monitoring
RX pattern
Matched
Received Packet
Number (in Bytes)
ELL
5
offset
42949
Plot Received Value (%)
Which Byte?
4
Byte Transmit
FF
Matched Character
H
Start of the frame
STOP
Which Byte? 2
300.0
200.0
100.0
0.0
111
Byte transmitted to the PI
FF
VISA resource name
COM16
Waveform Chart
Amplitude
300
250
200
150
100
50
0
477
1500
Time
Received Data
Received Data
RX Random Number
260
200
150
100
50
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65
Point
Path
C:\Users\farahman\Documents\Work\test.txt

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