**Report**

**Laboratory #5**

*“Asynchronous Serial Communications”*

by

Nayara Oliva Ferreira

11/9/2012

# 1. Introduction

In lab number 5 the task desired is to make the microcontroller communicate with the PC. We want the PC to send some signals via a terminal program (HyperTerminal) and the microcontroller gives the PC some text back to read. To accomplish this task is necessary to use the built-in Universal Synchronous/Asynchronous Receiver Transmitter (USART).

Some configuration are required to this lab like a 19 200 baud rate , 8 data bits , the transmitter and the receiver enabled and no use of interrupts.

In the program we need to configure all these requirements so can work properly with the PIC16F877. In the program also we need to be watching constantly what is coming from the PC so we can transmit what is asked. When is received from the PC the letter lower case ‘f’ we have to transmit our first name. When is received from the PC the letter lower case ‘l’ we have to transmit our last name. When is received from the PC the letter case ‘p’ we have to transmit our PID number. When is received from the PC the number 2 we have to lit LED 2 (port C). When is received from the PC the number 3 we have to lit LED 3 (port C).When is received from the PC the upper case letter ‘M’ we have to transmit the main menu provided by the teacher of this class.

All the background configuration of the terminal program to this lab is gave in the Lab #5 sheet.

# 2. Design and Implementation

Starting by the configurations necessary to the program work properly first we need to set the baud rate to 12 simply moving the decimal number to the register SPBRG, which is in bank 1. The calculus to the baud rate is presented below, with the previous information that we are using high speed:

Baud rate : Fosc/(16(X+1))

19 200 = 4MHz/(16(X+1))

307 200(X+1) = 4MHz

(X+1) = 4MHz/307 200

X = 12.02

X

After that we set TXSTA to ‘10110000’ where are selecting the asynchronous mode ,enabling the transmission, the baud rate to high speed. Set RCSTA, in bank 0, to ‘10100110’ where we are enabling the serial port, a single receive, continuous receive.

After initializing the registers the next subroutine that is going to be described is the main. The first thing we are going to do is transmit to display the main menu provided by the teacher after that there is a subroutine inside the main called main2 just to make sure the main menu is not going to be displayed again unless we receive a upper case letter ‘M’. In the main 2 we are going to check if we receive a word by checking RCIF inside the register PIR1 and keep checking until we have a word and after that we check if there are any overrun error of frame error. If there is an overrun error we clear CREN to fix it and if there is a frame error we update RCREG to fix it. After checking and fixing the errors we finally read the word that is stored in RCREG and store the word in TEMP, which is a temporary register. Made that we call the subroutine CHECKVALUE.

The CHECKVALUE is going to check if we are receiving ‘f’,’l’,’p’ ,’2’ or ‘3’ so we can transmit the text required by the lab. To do that we first clear the Z flag in the STATUS register move whatever is in W register to TEMP , subtract W from the TEMP and check if the Z flag is set. If the Z flag is set we call the display subroutine of the value being checking at that moment.

To display the first name, last name , the PID number and the main menu provided by the teacher we need a counter because we are going use a lookup table for each of this functions where is going to return chars where each of this char is going to be a letter or a space that we want to display. With this counter we can go all the way through the lookup table and display the whole text in question. So we move this counter to W register , call the lookup table , move the result to another temporary location called X and move the X to F register. After that is necessary to check if the Z flag of the STATUS register is set because at the lookup table to know we finished the transmission the last value moved to W register was 0. If the Z flag is not set we then know we can transmit and finally call the transmit subroutine. If the Z flag is set we return.

In the case of the RC2 and RC3 display subroutine we are not going to use a lookup table because all we want is to lit the LED. So all we need to do at the beginning of the subroutine is to check if the LED is on, if it is turn it off and return. If the LED is off then turn it on and return.

# 3. Test Results

The program worked as described in the flow chart . The program displayed correctly the results required on the terminal program. The program was tested and checked by the teacher.

# 4. Summary and Conclusions

As a conclusion is easy to realize that USART provide a greater flexibility when sending and receiving data. Communication with HyperTerminal used for the lab is the perfect example of this reality. It’s pretty simple to communicate with the PC. But still necessary extra attention for the registers configuration so the communication can work exactly as you want.

# Appendix A. Flow-chart

Display first name/last name/PID#/Menu

Display RC2/Rc3

Check the value

Main

Init

Check if the LED is on

Clear Z flag

Select bank 0

Clear the counter

Y

Set baud rate to 12

N

Move counter to W

Move w to temp

Check if have received a word

Turn On

Set TXSTA to 10100110

N

Subtract w- ‘f’

Return

Call lookup table

Have received a word?

Is Z flag set?

Set port C pin 2 and 3 as outputs

Turn OFF

Move result from lookup table to another temporary location

Y

Select bank 1

Return

Y

N

Transmit first name

Set RCSTA to 10110000

Read the word

Go and do the same steps for l,p,M,2 and 3 in the order

Go to Main

Check the Z flag

Is Z flag set?

After checking all of them return

Return

Call TXMIT,increment counter go back to the beggining

# Appendix B. Source Code

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Laboratory # 5

;\*

;\* This lab involves the use of microncontroller's built-in Universal Synchronous/Asynchronous Receiver Transmiter (USART). The USART will be

;\* used to communicate with the laboratorie's personal Computer via terminal program.

;\*

;\* EE3954 Microprocessors and Microcontrollers

;\* Student: Nayara Oliva Ferreira

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; Define some 8-bit file register labels

; See PIC16F877 datasheet for location of the Special Function Registers

\_\_CONFIG 0x3F39 ; WDT disable, XT osc, PWRT enable, LVProg disable, CP off

STATUS equ 0x03

PORTC equ 0x07 ; bank 0

TRISC equ 0x07 ; bank 0

PIR1 equ 0x0C ; PIR1 in bank 0

RCSTA equ 0x18 ; bank 0

TXSTA equ 0x18 ; bank 1

TXREG equ 0x19 ; bank 0

SPBRG equ 0x19 ; bank 1

RCREG equ 0x1A ; RC register

PCL equ 0x02 ; in all BANKS

TEMP equ 0x70 ; temperatary value store

TXIF equ 0x04 ;

X equ 0x71 ; in all banks

TEMP1 equ 0x72 ; in all banks

Z equ d'2' ;

RP0 equ d'5' ; bank select low bit in status reg.

F equ d'1' ;

RCIF equ d'5' ;

OERR equ d'1' ;

FERR equ d'2' ;

CREN equ d'4' ;

org 0x0000 ; program starts at Program Memory location 0x0000

nop ; put here so we can breakpoint at 0x0000

goto Inicio ; Jump over Tables so they can be in first 255 loc.

; Set Configuration bits in code:

; ALL DATA TABLES HERE

MENU: addwf PCL,F ; do computed goto's to get characters

retlw 0x0c ; 'new page'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw A'M' ; M

retlw 0x09 ; 'tab'

retlw A'M' ; M

retlw A'a' ; a

retlw A'i' ; i

retlw A'n' ; n

retlw A' ' ; 'space'

retlw A'M' ; M

retlw A'e' ; e

retlw A'n' ; n

retlw A'u' ; u

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw A'f' ; f

retlw 0x09 ; 'tab'

retlw A'F' ; F

retlw A'i' ; i

retlw A'r' ; r

retlw A's' ; s

retlw A't' ; t

retlw A' ' ; space

retlw A'N' ; N

retlw A'a' ; a

retlw A'm' ; m

retlw A'e' ; e

retlw 0x0d ; 'carriage return'

retlw A'\n' ; 'new line'

retlw A'\n' ; 'new line'

retlw 0x09 ; 'tab'

retlw A'l' ; l - this is lowercase "L"

retlw 0x09 ; 'tab'

retlw A'L' ; L

retlw A'a' ; a

retlw A's' ; s

retlw A't' ; t

retlw A' ' ; 'space'

retlw A'N' ; N

retlw A'a' ; a

retlw A'm' ; m

retlw A'e' ; e

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw A'p' ; p

retlw 0x09 ; 'tab'

retlw A'P' ; P

retlw A'I' ; I

retlw A'D' ; D

retlw A' ' ; 'space'

retlw A'#' ; #

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw A'2' ; 2

retlw 0x09 ; 'tab'

retlw A'T' ; T

retlw A'o' ; o

retlw A'g' ; g

retlw A'g' ; g

retlw A'l' ; l

retlw A'e' ; e

retlw A' ' ; 'space'

retlw A'R' ; R

retlw A'C' ; C

retlw A'2' ; 2

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw A'3' ; 3

retlw 0x09 ; 'tab'

retlw A'T' ; T

retlw A'o' ; o

retlw A'g' ; g

retlw A'g' ; g

retlw A'l' ; l

retlw A'e' ; e

retlw A' ' ; 'space'

retlw A'R' ; R

retlw A'C' ; C

retlw A'3' ; 3

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw 0x00 ; end of table - character

FIRSTNAME: addwf PCL,F ; do computed goto's to get the first name

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw 0x0d ; 'carriage return'

retlw 0x0a ; 'new line'

retlw 0x09 ; 'tab'

retlw a'N' ; N

retlw a'a' ; a

retlw a'y' ; y

retlw a'a' ; a

retlw a'r' ; r

retlw a'a' ; a

retlw 0x00 ; end of table - character

LASTNAME: addwf PCL,F ; do computed goto's to get the last name

retlw 0x09 ; 'tab'

retlw 0x09 ; 'tab'

retlw 0x0d ; 'carriage return

retlw 0x0a ; 'new line''

retlw 0x09 ; 'tab'

retlw a'F' ; F

retlw a'r' ; r

retlw a'i' ; i

retlw a't' ; t

retlw a'z' ; z

retlw a'e' ; e

retlw a'n' ; n

retlw 0x00 ; end of table - character

PID: addwf PCL,F ; do computed goto's to get Pid#

retlw 0x09 ; 'tab'

retlw 0x0a ; 'new line'

retlw 0x0a ; 'new line'

retlw a'P' ; P

retlw a'1' ; 1

retlw a'0' ; 0

retlw a'0' ; 0

retlw a'1' ; 1

retlw a'4' ; 4

retlw a'8' ; 8

retlw a'0' ; 0

retlw a'2' ; 2

retlw a'7' ; 7

retlw 0x00 ; end of table - character

Inicio: bsf STATUS,RP0 ; access bank 1

movlw b'10110001' ; RC6: TX (output); RC7: RX (input)

movwf TRISC ; set the port?s inputs/outputs

movlw b'10100110' ; setup the transmission

movwf TXSTA ; 8-bits, asynch., etc.

movlw d'12' ; set baud rate to 57,600

movwf SPBRG

bcf STATUS,RP0 ; access bank 0

movlw b'10110000' ; SPEN enable, CREN enable

movwf RCSTA ; enable the serial port

MAIN: call DMENU

main2: btfss PIR1,RCIF ; received a word yet?

goto main2

call RCV

movf RCREG,W ; read the word

movwf TEMP ; store it in a temporary location

call CHECKVALUE

goto main2 ; keep displaying it

CHECKVALUE: bcf STATUS,Z ; clear the Z bit

sublw A'f' ; subtract f from w

btfsc STATUS,Z ; check if they are equal

call DFIRSTNAME ; if the are display my first name

movf TEMP,W ; if it's not move TEMP to W

sublw A'l' ; subtract TEMP of W

btfsc STATUS,Z ; check if they are equal

call DLASTNAME ; if they are diplay my last name

movf TEMP,W ; if they are not move TEMP to W

sublw A'p' ; subtract p of W

btfsc STATUS,Z ; check if they are equal

call DPID ; if they are display my PID#

movf TEMP,W ; but if they're not copy TEMP to W

sublw A'2' ; subtract 2 of W

btfsc STATUS,Z ; check if they are equal

call DLED2 ; if they are diplay toggle RC2

movf TEMP,W ; if they are not they copy TEMP to W

sublw A'3' ; subtract 3 of W

btfsc STATUS,Z ; check if they are equal

call DLED3 ; if they are display toggle RC3

movf TEMP,W ; if they are not copy TEMP to W

sublw a'M' ; subtract W of W

btfsc STATUS,Z ; check if they are equal

call DMENU ; if they are display the main

return ; if they are not return

DFIRSTNAME: clrf X ; set for first character in table

FNNXT: movf X,W ; put counter into 'W'

call FIRSTNAME ; go get a character from table

movwf TEMP1 ; char is in W - store it temporarily

movf TEMP1,F ;

btfsc STATUS,Z ; test if it is 0x00 - end of table

return

call TXMIT ; transmit W to terminal

incf X,F ; increment table counter to display na next char.

goto FNNXT ; go back and keep going to lookup table to display the rest of it.

DLASTNAME: clrf X ; set for first character in table

LNNXT: movf X,W ; put counter into 'W'

call LASTNAME ; go get a character from table

movwf TEMP1 ; char is in W - store it temporarily

movf TEMP1,F ;

btfsc STATUS,Z ; test if it is 0x00 - end of table

return

call TXMIT ; transmit W to terminal

incf X,F ; increment table counter to display na next char.

goto LNNXT ; go back and keep going to lookup table to display the rest of it.

DPID: clrf X ; set for first character in table

PNXT: movf X,W ; put counter into 'W'

call PID ; go get a character from table

movwf TEMP1 ; char is in W - store it temporarily

movf TEMP1,F ;

btfsc STATUS,Z ; test if it is 0x00 - end of table

return

call TXMIT ; transmit W to terminal

incf X,F ; increment table counter to display na next char.

goto PNXT ; go back and keep going to lookup table to display the rest of it.

DMENU: clrf X ; set for first character in table

MNXT: movf X,W ; put counter into 'W'

call MENU ; go get a character from table

movwf TEMP1 ; char is in W - store it temporarily

movf TEMP1,F ;

btfsc STATUS,Z ; test if it is 0x00 - end of table

return

call TXMIT ; transmit W to terminal

incf X,F ; increment table counter to display na next char.

goto MNXT ; go back and keep going to lookup table to display the rest of it.

DLED2: btfss PORTC,2 ; Check if the led is on

goto ON2 ; if is not on call the routine to turn it

goto OFF2 ; call the routine to turn it off

ON2: bsf PORTC,2 ; If is not on then turn it on

return

OFF2: bcf PORTC,2 ; If is on turn it off

return

DLED3: btfss PORTC,3 ; Check if the led is on

goto ON3 ; if is not on call the routine to turn it

goto OFF3 ; call the routine to turn it off

ON3: bsf PORTC,3 ; If is not on then turn it on

return

OFF3: bcf PORTC,3 ; If is on turn it off

return

TXMIT: btfss PIR1,TXIF ; is TXREG empty?

goto TXMIT ; if not, test again

movwf TXREG ; send the data

return ; return to calling progam.

RCV: btfss PIR1,RCIF ; is data in RCREG ?

goto RCV ; no data yet, so try again

btfsc RCSTA,OERR ; did overrun error occur?

goto CLROR ; if so, go clear it.

btfsc RCSTA,FERR ; Frame error ? (FERR = 2)

goto CLRFE

movf RCREG, W ; clears FERR, Rcvd data is lost

return ; return to calling program

CLROR: bcf RCSTA,CREN ; clear the cont. receive bit

bsf RCSTA,CREN ; turn on cont. receive bit

goto RCV ; OERR is clear, check again

CLRFE: movf RCREG,W ; Clear the frame error this way

goto RCV ; OERR is clear, check again

end