अभियंता

Think outside the 'bots

Communicating through UART & PIC

–

Submission File:

Team A:

Devanshi Chauhan
Jainam Shah
Harsh Manghnani
Hemangi Siddhpura

Table Of Content

No.	Topic	Page No.
1	Overview	2
2	Communication Module	2
3	Types of Communication	2
4	UART Protocol	3
5	Terminologies Related to Communication	3
6	Difference between UART and USART	4
7	UART module in PIC16F877	4
8	Advantages and Disadvantages of UART	12
9	Task 2.1 - UART from Scratch	13
10	Task 2.2 - UART using internal hardware	14
11	Task 2.3 - Communicating two PIC controllers	17
12	Problems Faced	18
13	Conclusion	18
14	References	19

1. Overview

The task is about interfacing two PIC controllers with each other through implementing a UART Protocol. In the PIC microcontroller, there is an internal hardware which implements the UART protocol. In order to learn UART protocol from scratch it is required to start implementing the protocol manually (without using internal dedicated hardware or software serial libraries). Then further we will be switching to communicate two MCUs using internal hardware in order to learn using internal hardware functioning and usage from the datasheet.

2. Communication Module

Communication Module manages exchange of information between different devices. It is capable to provide wired and wireless communication between microcontrollers, PCs, Modems, Smartphones, Wifi devices, Bluetooth devices etc. Communication can be established between the controllers, controller and PC, or controller and external device.

3. Types of Communication

1) Serial communication

Serial communication is a process of sending one bit information at a time in which each bit consumes one clock cycle to transfer information from one end to the other end. As all the bits are sent sequentially over a single wire, the overall communication cost reduces and also data transfer speed reduces.

2) Parallel communication

Parallel communication is a process of sending several bits of information at the same time in which only one clock is consumed to pass on several bits of data from one end to the other end. As all the bits are transmitted simultaneously, it uses multiple wires to convey information in binary as a result cost increases and also data transfer speed increases.

4. UART Protocol

UART stands for Universal Asynchronous Receiver/Transmitter. It is a standalone protocol or basically an internal circuitry for communication. As the name suggests it is used for transmitting and receiving data. It transmits data asynchronously, which means instead of clock signal it uses start and stop bits to the data packet being transferred[7]. By this process UART knows when to start and stop reading bits. These incoming bits are read on a particular frequency called baud rate which can be set to sample the incoming data accordingly. The advantages of UART over I2C and SPI protocols is that no clock is needed to transmit for data synchronization and there is an advantage of parity bit for error detection.

5. Terminologies Related to UART

1) Shift register:

Shift registers are used to shift the data received from msb to lsb in serial communication. They are basically D flip flops connected serially in order to pass on the data bits. It takes one clock cycle to shift one bit of data.

2) Baud Rate:

Baud rate is the measure of the speed of data transfer between transmitter and receiver. It is measured in bits per second(bps). Baud rate is mainly used for serial communication. In UART, we have to set the same baud rate for communication between transmitter and receiver because if we do not have the same baud rate then we will get undesired output due to speed difference.

3) Data frame:

In UART protocol, each byte or word is framed with start and stop bits so the complete message will undergo a process known as framing. Output of the framing process is known as data frame. The data frame consists of a start bit, data bits, a parity bit(optional) and one or two stop bits. UART communication takes place using data frames.

6. Difference between UART and USART

UART	USART
Standalone communication protocol	It supports multiple protocols like LIN, RS-485,IrDA, etc
Commonly used for low speed applications	Most suitable for high speed communication
Data rate is relatively low	Data rate is much higher
The clock is generated locally and both devices are configured to run at the same baud rate	The clock signal is generated by the transmitter and sent to the receiver during data transmission
The baud rate of the receiver module must be known prior to any communication	The baud rate is useless as the receiver module will be using the clock coming from the transmitter device

Tabel 1[6]. Difference between UART & USART

7. UART module in PIC16F877

After getting an idea of UART protocol, let’s get information on dedicated hardware in PIC16F877 for UART serial communication.

UART can be configured in following modes[3]:

- Asynchronous(Full Duplex)
- Synchronous-Master(Half Duplex)
- Synchronous-Slave(Half Duplex)

Register used in UART:

No.	Register	Description	Address
1	TXSTA	TRANSMIT STATUS AND CONTROL REGISTER	98h
2	RCSTA	RECEIVE STATUS AND CONTROL REGISTER	18h
3	SPBRG	USART Baud Rate Generator	99h
4	TXREG	USART Transmit Register. Hold data to be transmitted.	19h
5	RCREG	USART Receiver Register. Hold received data	1Ah

Tabel 2. Registers used in Uart with address and description

UART Baud Rate Generator(BRG):

Baud rate is a very important parameter in UART protocol. To communicate between transmitter and receiver it is necessary to set the same baud rate of transmitter and receiver[5]. The BRG supports both the asynchronous and synchronous modes of the USART. The SPBRG register controls the period of a free running 8-bit timer. PIC supports high as well as low speed uart communication depending upon the value of the BRGH bit in asynchronous mode of communication. In synchronous mode of communication this bit is ignored.

Calculation of SPBRG using Baud Rate Formula:

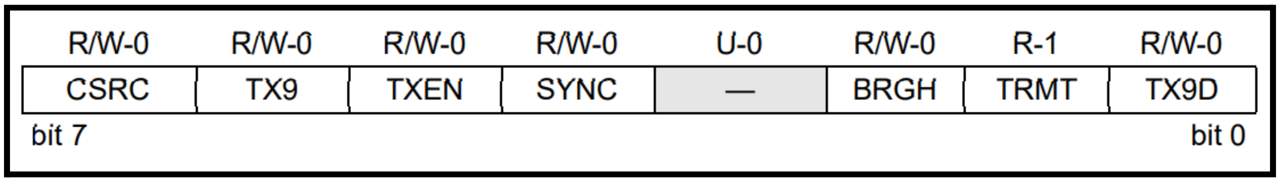
SYNC	BRGH = 0(Low Speed)	BRGH =1 (High Speed)
0(Asynchronous)	Baud Rate $= \frac{Fosc}{64(X+1)}$	Baud Rate $= \frac{Fosc}{16(X+1)}$
1(Synchronous)	Baud Rate $= \frac{Fosc}{4(X+1)}$	

X = Value of SPBRG(0-255)

Tabel 3. Calculation for SPBRG using baud rate formula

Transmitter:

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)



R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

Fig 1[8]. TXSTA Register configuration

Bit No.	Bit Name	Description
7	CSRC	Clock Source Select bit Asynchronous mode: Don't care Synchronous mode: 1 = Master mode (clock generated internally from BRG) 0 = Slave mode (clock from external source)
6	TX9	9-bit Transmit Enable bit 1 = Selects 9-bit transmission 0 = Selects 8-bit transmission
5	TXEN	Transmit Enable bit 1 = Transmit enabled 0 = Transmit disabled
4	SYNC	USART Mode Select bit 1 = Synchronous mode 0 = Asynchronous mode
3	-	Unimplemented: Read as '0'
2	BRGH	High Baud Rate Select bit Asynchronous mode: 1 = High speed 0 = Low speed Synchronous mode: Unused in this mode

Bit No.	Bit Name	Description
1	TMRT	Transmit Shift Register Status bit 1 = TSR empty 0 = TSR full
0	TX9D	9th bit of Transmit Data, can be parity bit

Tabel 4. Bit wise Explanation of TXSTA

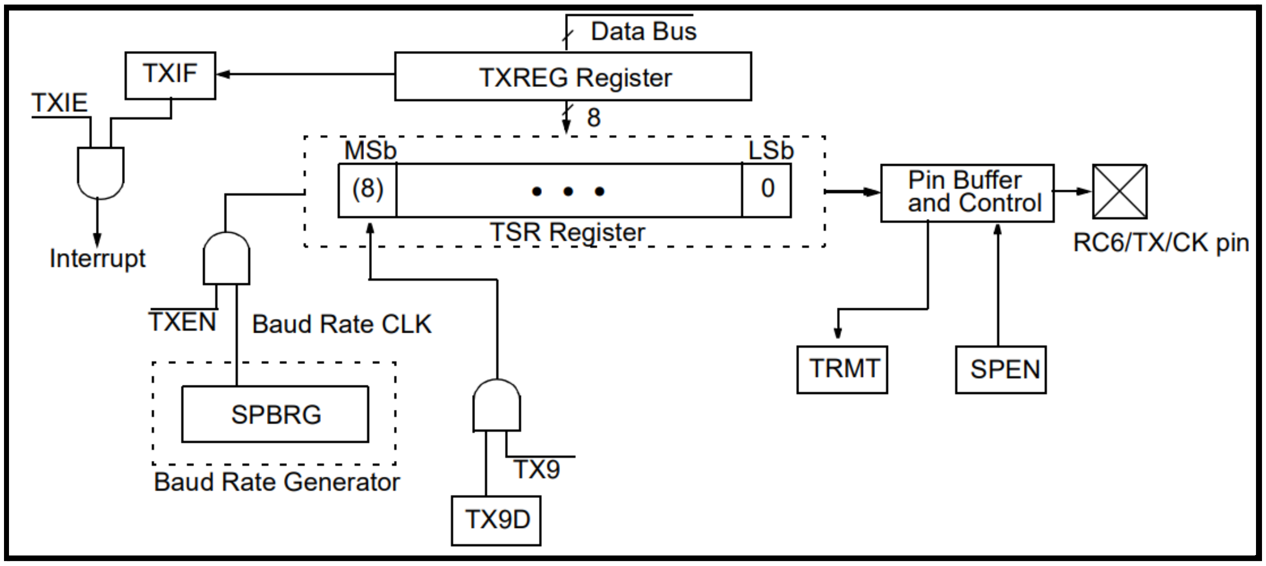


Fig 2[8]. USART TRANSMIT BLOCK DIAGRAM

TSR(Transmitter Shift Register) is the most important register of transmitter. As shown in fig 2, initially, data is loaded into the TXREG buffer register. Further an internal 8 or 9 bit clock from the SPBRG register loads data in the TSR register by enabling the TXEN bit. As soon as the TXREG buffer becomes empty, the TXIF flag will be set by the hardware(provided TXIE is set). TXIF indicates the status of the TXREG register and TRMT shows the status of the TSR register. TRMT is used to determine whether TSR is empty or not.

Note:

- The TSR register is not mapped in data memory, so it is not available to the user.
- Flag bit TXIF is set when the enable bit TXEN is set. TXIF is cleared by loading TXREG.

- Clearing the enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter.

The value of bits in TXSTA changes according to fig 3 when one word is transmitted and it's value changes according to fig 4 when two words are transmitted.

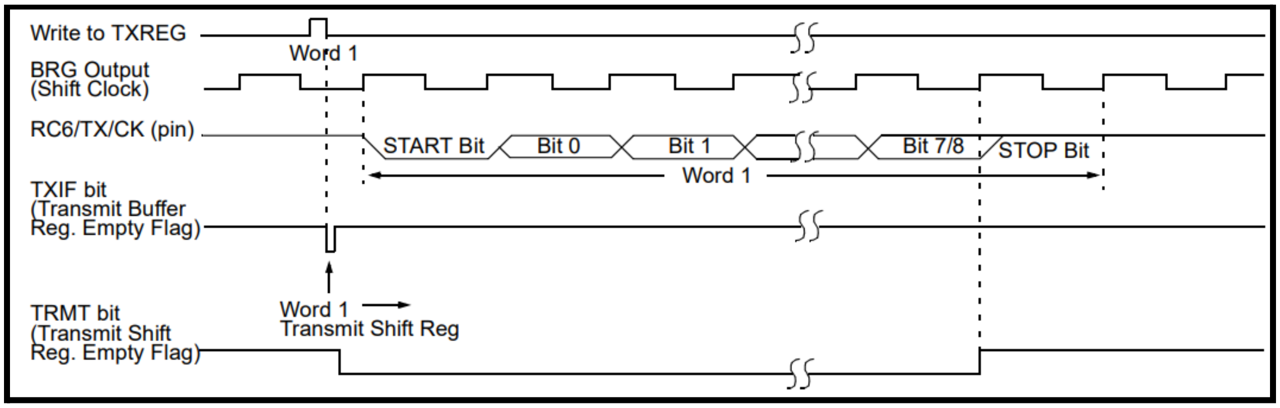

Write to TXREG
Word 1
BRG Output (Shift Clock)
RC6/TX/CK (pin)
START Bit
Bit 0
Bit 1
Bit 7/8
STOP Bit
Word 1
TXIF bit (Transmit Buffer Reg. Empty Flag)
TRMT bit (Transmit Shift Reg. Empty Flag)
Word 1 → Transmit Shift Reg

Fig 3[8].change in bits of register when a word is transmitted

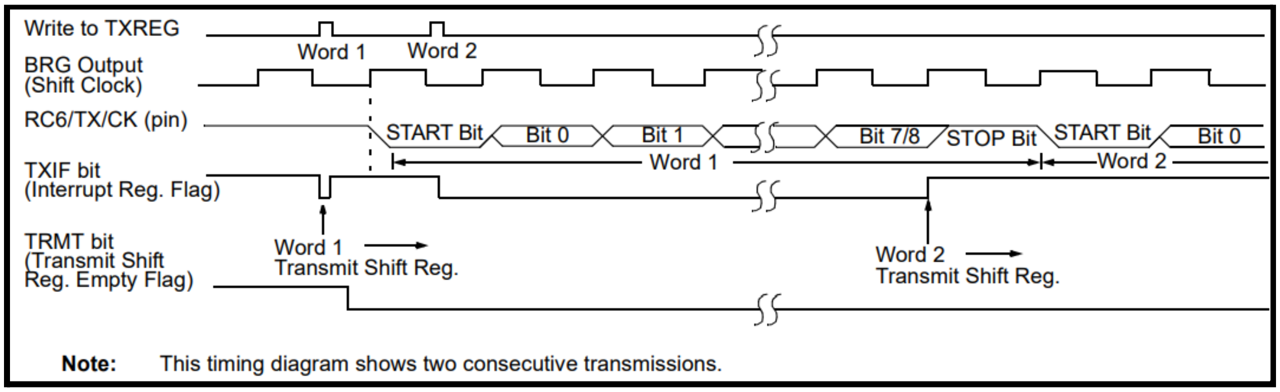
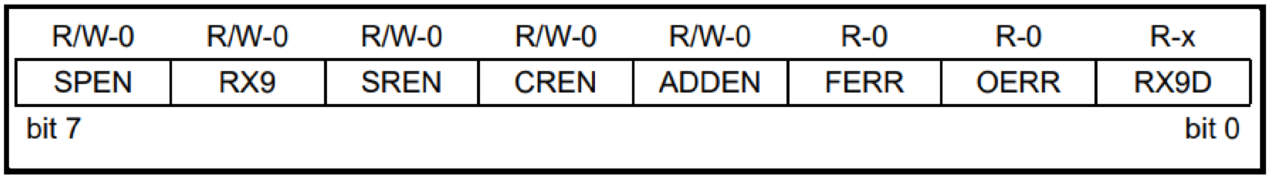

Write to TXREG
Word 1
Word 2
BRG Output (Shift Clock)
RC6/TX/CK (pin)
START Bit
Bit 0
Bit 1
Bit 7/8
STOP Bit
START Bit
Bit 0
Word 1
Word 2
TXIF bit (Interrupt Reg. Flag)
TRMT bit (Transmit Shift Reg. Empty Flag)
Word 1 → Transmit Shift Reg.
Word 2 → Transmit Shift Reg.
Note: This timing diagram shows two consecutive transmissions.

Fig 4[8].change in bits of register when two words are transmitted

Receiver:

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Fig 5[8]. RCSTA Register configuration

Bit No.	Bit Name	Description
7	SPEN	Serial Port Enable bit 1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins) 0 = Serial port disabled
6	RX9	9-bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception
5	SREN	Single Receive Enable bit Asynchronous mode: Don't care Synchronous mode - master: 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. Synchronous mode - slave: Don't care
4	CREN	Continuous Receive Enable bit Asynchronous mode: 1 = Enables continuous receive 0 = Disables continuous receive Synchronous mode: 1 = Enables continuous receive until enable bit CREN is cleared 0 = Disables continuous receive
3	ADDEN	Address Detect Enable bit Asynchronous mode 9-bit (RX9 = 1): 1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

Bit No.	Bit Name	Description
2	FERR	Framing Error bit 1 = Framing error (can be updated by reading RCREG register and receive next valid byte) 0 = No framing error
1	OERR	Overrun Error bit 1 = Overrun error (can be cleared by clearing bit CREN) 0 = No overrun error
0	RX9D	9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

Tabel 5. Bit wise Explanation of RCSTA

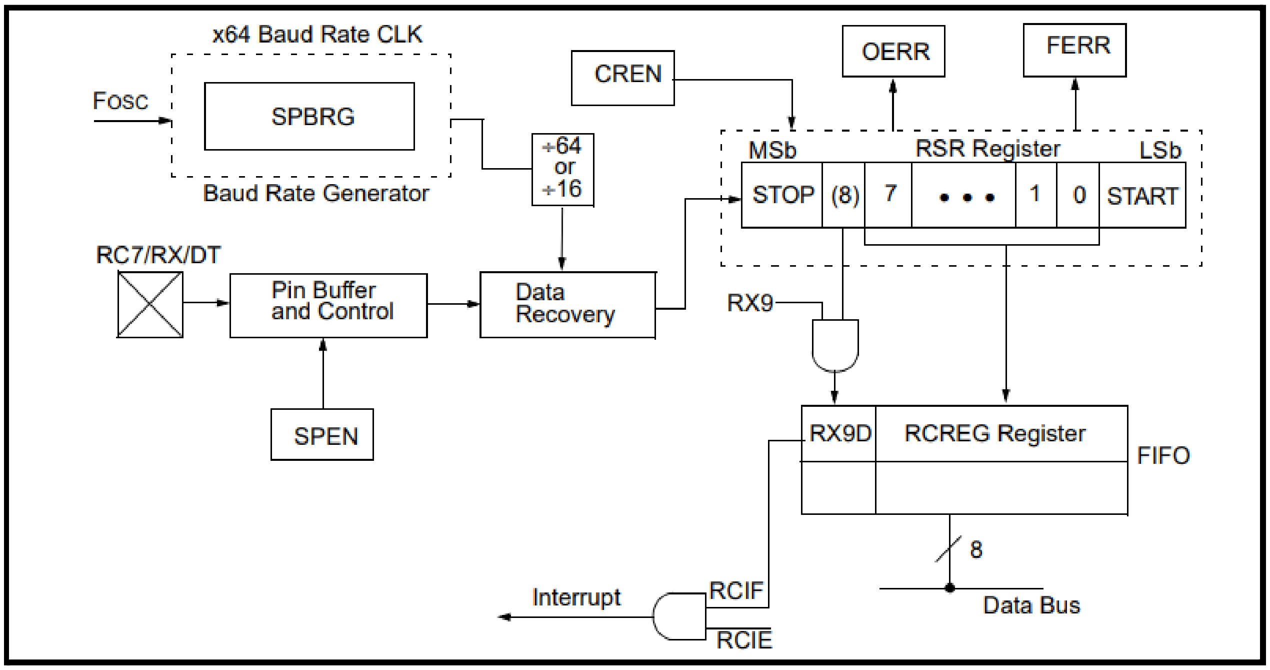


Fig 6[8]. USART RECEIVER BLOCK DIAGRAM

As shown in fig 6, data is received on an RX/C7 pin. Once the Asynchronous mode is selected, reception is enabled by setting the CREN bit. RSR(Receiver Shift Register) is the most important register of receiver in PIC. First data frame is received in RSR and then RCREG buffer will be loaded with receiving data. When RCREG will be full it will set RCIF flag(provided RCIE is enabled).

In fig 7 the value of bits in RCSTA changes as shown when two words are received.

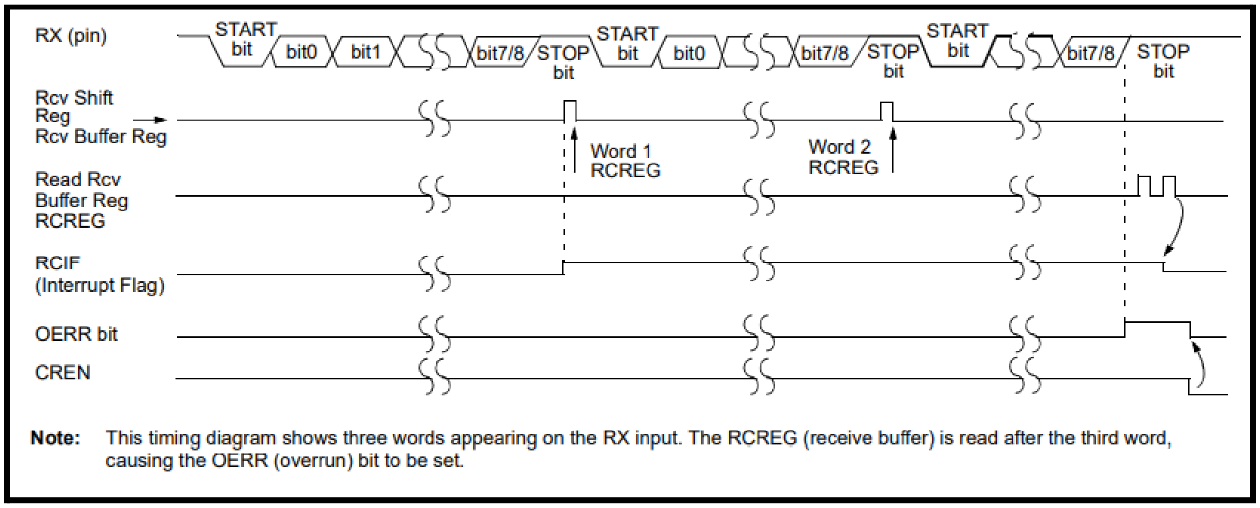

RX (pin)
START bit
bit0
bit1
bit7/8
STOP bit
Rcv Shift Reg
Rcv Buffer Reg
Word 1 RCREG
Word 2 RCREG
Read Rcv Buffer Reg RCREG
RCIF (Interrupt Flag)
OERR bit
CREN
Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

Fig 7[8]. change in bits of register when two words are received

Steps To Send Char:

1. Wait till the previous char is transmitted and TXIF will be set when the TXREG is empty.
2. Clear the TXIF for next cycle.
3. Load the new character to be transmitted into TSR.

```
void UART_send_char(char bt)
{
    while(!TXIF);
    TXREG = bt;
}
```

Steps To Receive Char:

1. Wait till the Data is received. RCIF will be set once the data is received in the RCREG register.
2. Clear the receiver flag(RCIF) for next cycle.
3. Copy/Read the received data from RCREG register.

```
char UART_get_char()
{
    while(!RCIF);      // hold the program till RX buffer is free
```

```
        return RCREG;       //receive the value and send it to main function

    }
```

Steps to send string:

1. We made a user defined for sending string for that we used pointers.
2. Inside an infinite loop which will only get into when receiving a value at a particular character.
3. Then the send char method is called and position incremented.

```
    void UART_send_string(char* st_pt){
        while(*st_pt)
        UART_send_char(*st_pt++);
    }
```

Steps to receive string:

1. We wrote user defined for receiving string
2. Inside an infinite loop we passed value at particular character
3. Then we called send character method and pass on the incremented value

```
    void UART_send_string(char* st_pt){
        while(*st_pt)
        UART_send_char(*st_pt++);
    }
```

8. Advantages and Disadvantages of UART[4]

Advantages:

- Uses only two wires
- No clock signal required
- Provides a parity bit for error detection
- Cost and size will be much lesser

Disadvantages:

- Size of the data frame is limited upto 9 bits.

- Multiple slave or multiple master systems are not supported.
- Limited speed is a barrier for the application which requires higher data transmission rate.
- For wireless transmission more antennas and receivers are required due to which cost increases.

9. Task 2.1 - UART from scratch [Failure]

(A) UART receiver - without using internal hardware

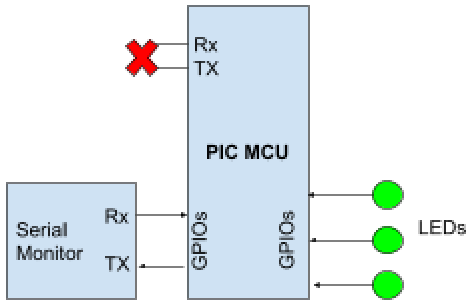


Fig 8. RCSTA Register configuration

(B) Controlling the LED from serial monitor

In this section you will be controlling 3 LEDs from your serial monitor. The received binary string then will be analyzed and the respective LED will be turned on.

OUTPUT: sending 1 from serial monitor will toggle LED-1 , sending 2 from serial monitor will toggle LED-2 and likewise for LED-3 also.

Implementation of 2.1 (A)&(B):

- In this task, first we declared a variable that works as a 9 bit register for storing data bits along with a start bit which is logic 0.
- For sampling data bits at 9600 baud rate, we kept timer0 in polling mode to generate interrupt at every 100us.
- For detecting the start bit we used hardware interrupt to detect the falling edge at the receiver pin. For that confirmation we connected LED at RB0 pin which has external interrupt facility available.
- We write code for hardware interrupt in which whenever the falling edge comes we enable timer0.

- As we cannot write two ISRs in the same code, we go for polling mode in the main function, so in an infinite loop we generate sample instance signal just to check whether our sampling is correct or not. It was correct. After that we stored it into a variable by shifting bits and compared the numbers but we didn't get the output.

10. Task 2.2 - UART using internal hardware

(A) Implement receiver

To connect a serial monitor on hardware serial (UART pins) and implement the same LED control task (this time the communication will happen using internal dedicated hardware) (refer figure 2).

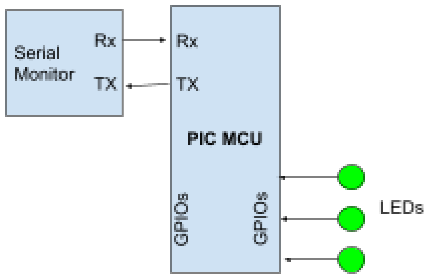


Fig 9. RCSTA Register configuration

(B) Implement transmitter

Extend its functionality by reprinting the character entered in the serial monitor.
OUTPUT: when you enters 1 from from serial monitor it will print 1 on serial monitor and also it will turn on the LED 1 and likewise for all 3 LEDs

Implementation of 2.2 (A)&(B):

- In order to configure the UART module in PIC for serial communication, first of all we have to configure registers as discussed in section 7.

1. User defined function UART_Initialize is written as shown in the given code 2.2(A).

```
void Initialize_UART(void)
{
   TRISC6 = 0; //TX Pin set as output
   TRISC7 = 1; // RX Pin set as input
```

```
    SPBRG = ((_XTAL_FREQ/16)/Baud_rate) - 1;
    BRGH  = 1;  // for high baud_rate

    SYNC  = 0;    // Asynchronous
    SPEN  = 1;    // Enable serial port pins

    TXEN  = 1;    //enable transmission
    CREN  = 1;    // enable reception

    TX9   = 0;    // 8-bit transmission mode selected
    RX9   = 0;    // 8-bit reception mode selected
}
```

2. To receive data from serial monitor(TASK 2.2_A):

- It is necessary to check whether the RCREG register is full or not.
- When RCREG is full it will set RCIF bit after that data sent by the serial monitor will be obtained.
- In the case of continuous data reception, if any data overwrites the value of previous value then OERR bit will be set. To clear this bit or resolve this problem we can clear the value of CREN(To stop continuous data reception). Once OERR is cleared, we can set the CREN bit again.
- The above mentioned steps can be written as shown below in code 2.2(B).

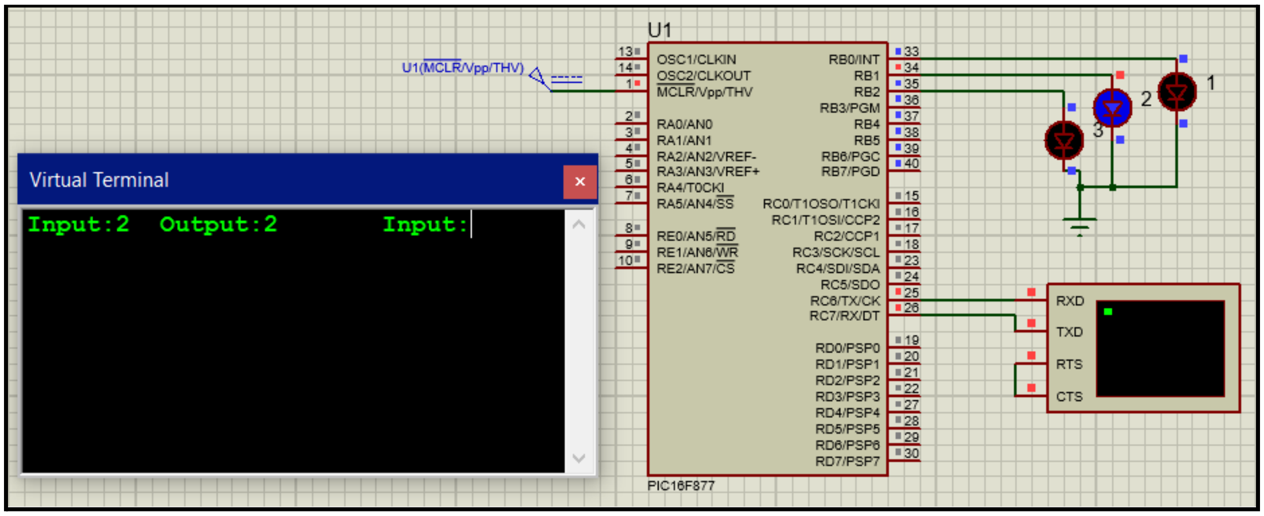
```
char UART_get_char(){
    if(OERR){
        CREN = 0;
        CREN = 1;
    }
    while(!RCIF);
    return RCREG;
}
```

3. To transmit data to serial monitor(TASK 2.2_B):

- We have to check whether the TXREG register is empty or not.
- When TXREG is empty it will set the TXIF bit after which values can be sent in the TXREG buffer for sending data in the serial monitor.
- In the case of sending continuous data or string we can use a pointer which points to the base value of that string or character array.
- The above mentioned steps can be written as shown below in code 2.2(C).

```
//To send a character
void UART_send_char(char bt){
   while(!TXIF);
   TXREG = bt;}
// To send STRING
void UART_send_string(char* st_pt){
   while(*st_pt)
      UART_send_char(*st_pt++);
}
```

Output :



```
Input:2  Output:2       Input:
```

Fig 10. RCSTA Register configuration

11. Task 2.3 - Communicating two PIC controllers

Communicate with UART

Here you have to control the LEDs connected on MCU-1 by switches connected with MCU-2. In order to pass the commands both the MCUs must be communicating with internal dedicated hardware of UART. (refer figure 3)

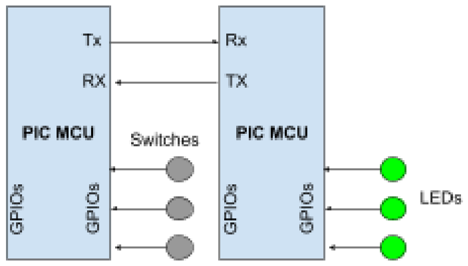


Fig 11. RCSTA Register configuration

Implementation of 2.3:

- In this task we have to communicate between two PICs. For that we can use one PIC as transmitter and one PIC as receiver.
- As discussed in Task 2.2 we can use Code 2.2(B) and Code 2.2(C) for transmitter and Code 2.2(A) for receiver.

Note :- We must keep the baud rate of both PICs the same.

Output :

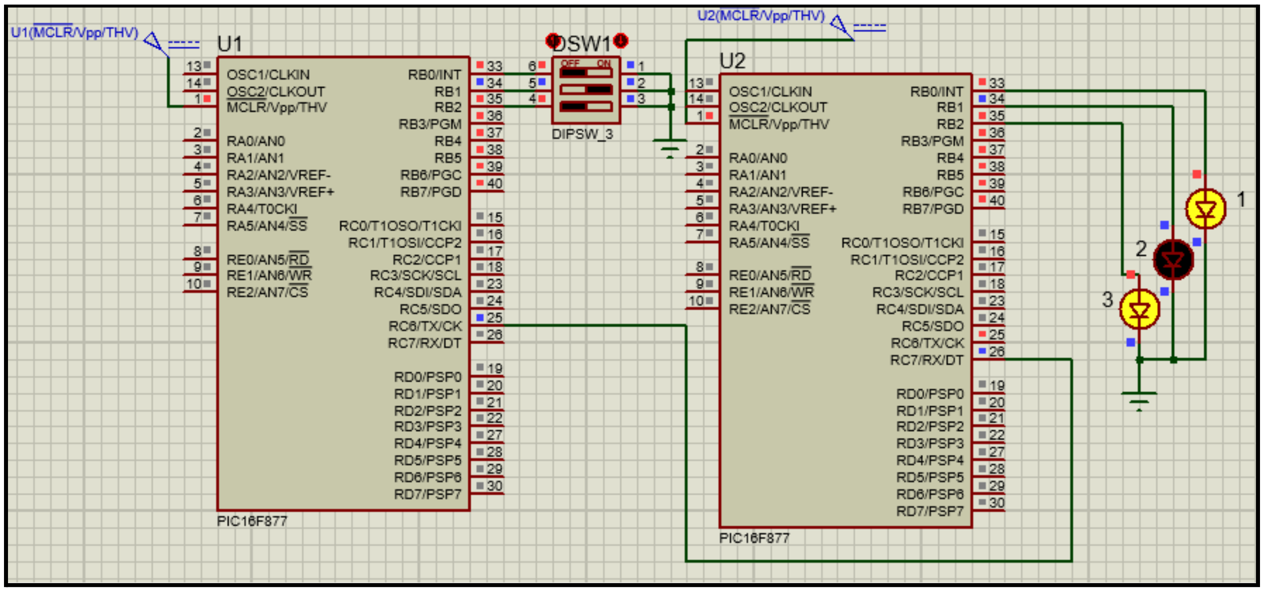


Fig 13. RCSTA Register configuration

12. Problems Faced

- First we weren't able to detect the start bit so after reading the datasheet we found out the approach for hardware interrupt on the falling edge to detect the start bit.
- It showed triangular waves instead of square waves, when we tried to generate sampling instance signals for all frequencies.

13. Conclusion

In this task, the team carried out learning regarding communication protocol (UART). They got to know that UART is an asynchronous protocol that uses baud rate to synchronize data instead of clock signal. It sends data by forming a frame. Initially, a start bit is sent, then data bits are sent and lastly one or two stop bits are sent. UART is used in ICs, RAM, ROM, peripheral devices, etc. And finally, I created a simple communication application MCU-To-MCU and tested it.

14. References

1. UNDERSTANDING USART MODULE
2. UART Interrupt Pic Microcontroller, Example Code MPLAB XC8 Compiler
3. UART configuration modes
4. UART vs I2C vs SPI – Communication Protocols and Uses
5. Using UART of PIC Microcontroller with MPLAB XC8
6. UART | Serial Communication With PIC Microcontrollers Tutorial
7. Basics of UART Communication
8. PIC16F877 Datasheet