



PIC18F4550 USART

Introduction

Several devices such as GPS, GSM, RFID, sensors, etc need to communicate with the PIC microcontroller for transmitting or receiving information. To communicate with the PIC microcontroller, several communication protocols are used such as RS232, SPI, I2C, CAN, etc. Basically, a protocol is a set of rules agreed by both, the sender and the receiver, on -

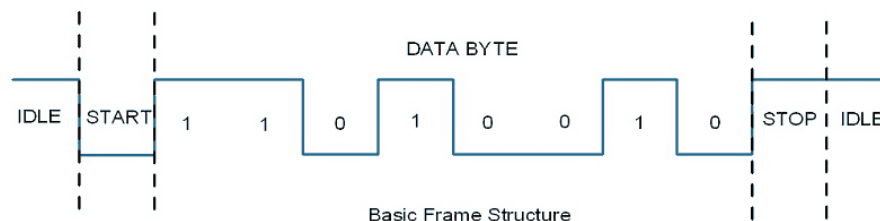
- How the data is packed?
- How many bits constitute a character?
- When the data begins and ends?

PIC18F4550 has an in-built USART module which is useful for serial communication. With the help of USART, we can send/receive data to a computer or other devices. USART is also used in interfacing PIC with various modules like Wi-Fi (ESP8266), Bluetooth, GPS, GSM, etc.

We will see how the communication is established between PIC microcontroller and PC through USART using RS232 protocol. We will also see how to communicate with laptops, which do not have an RS232 DB9 port, and instead use a USB port.

Let us start with the serial communication using PIC18F4550.

Asynchronous Communication: PIC18F4550 has a built-in asynchronous receiver-transmitter. Asynchronous means each character (data byte) is placed in between the start and stop bits. The start bit is always 0 (low) and the stop bit is always 1 (high).



Bit Rate & Baud Rate: The rate of data transfer in serial data communication is stated in bps (bits per second). Another widely used terminology for bps is baud rate; means, a number of changes in signal per second. Here the signal is in bits, therefore bit rate = baud rate.

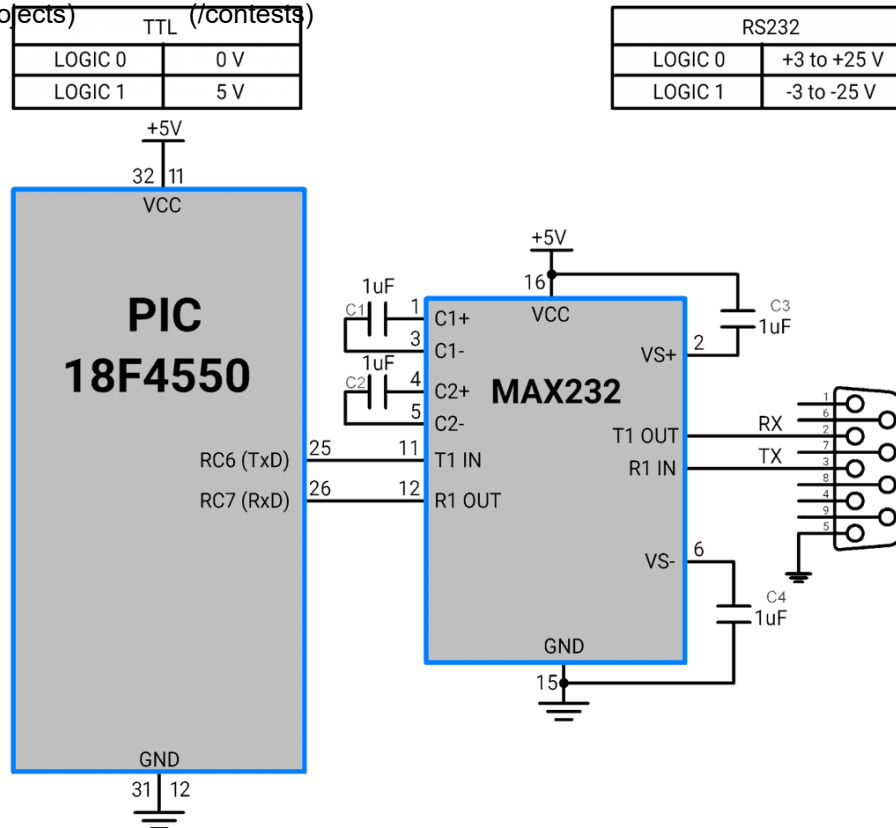
Interface: Although there are many pins in the DB9 connector, we do not need all. We only use pins RX, TX, and GND.

Level Conversion

Convert PIC18F TTL levels to RS232 levels and vice-versa



PC has RS232 levels whereas the PIC microcontroller has TTL levels. The RS232 has different voltage levels for logic 0 and 1. To make it compatible with the PIC TTL voltage levels, we have to use a MAX232 IC.



PIC18F TTL to RS232 Level conversion

Baud Rate Calculation

How to Calculate the Baud Rate in PIC18F4550?

$$\text{Desired Baud Rate} = \frac{F_{osc}}{64 * (X + 1)}$$

a value that will be loaded into the SPBRG register (16-bit) of the PIC18F4550 to get the desired baud rate. The value of SPBRG for the desired baud rate is calculated as,

$$SPBRG = \frac{F_{osc}}{64 * \text{Desired Baud Rate}} - 1$$

E.g.

Suppose, F_{osc} = 8 MHz and Baud Rate = 9600 bps

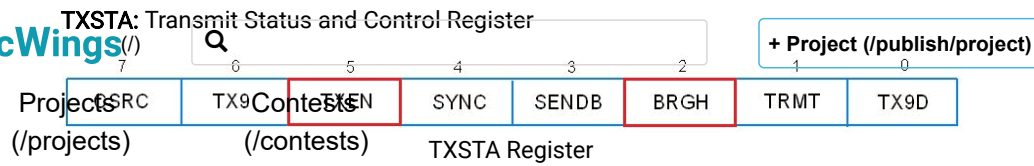
Then, $SPBRG = ((8 \text{ MHz}) / (64 \times 9600)) - 1$

$SPBRG = (8 \text{ MHz} / 64 \times 9600) - 1$

Therefore, $SPBRG = 12$

The above formula depends on the **BRGH** bit in the **TXSTA** register.

Let's see the TXSTA register, which is used for transmission setting, in detail as follows



TXEN: Transmit Enable Bit

1 = Enable the transmission

0 = Disable the transmission

BRGH: High Baud Rate select bit

0 = Low speed

$$SPBRG = \frac{F_{osc}}{64 * Desired Baud Rate} - 1$$

1 = High speed

$$SPBRG = \frac{F_{osc}}{16 * Desired Baud Rate} - 1$$

Load the calculated value directly to the SPBRG register.

CSRC bit is not used for asynchronous communication.

TX9 : 9th Transmit Enable Bit

0 = Select 8-bit transmission

1 = Select 9-bit transmission

SYNC : USART Mode Select Bit

0 = Asynchronous mode

1 = Synchronous mode

SENCB : Send Break Character bit

1 = Send Sync Break on next transmission (cleared by hardware upon completion)

0 = Sync Break transmission completed

TRMT : Transmit Shift Register Status Bit

0 = TSR full

1 = TSR empty

TX9D: 9th bit of transmitting data

Can be Address / Data bit or a parity bit.

In PIC18F4550, the RCSTA register is used for serial data receive settings.

RCSTA: Receive Control and Status Register



Platforms
(/explore)

Projects
(/projects)

Contests
(/contests)

RCSTA Register



SPEN: Serial Port Enable

1 = Enable Serial Port for communication

0 = Disable Serial Port for communication

RX9: 9-bit Receive Enable bit

1 = Enable 9-bit reception

0 = Enable 8-bit reception

Generally, we use 8-bit reception

SREN: Single Receive Enable bit

Not used

CREN: Continuous Receive Enable bit

1 = Enable receiver for continuous reception of data byte

0 = Disable receiver

ADDEN: Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1)

1 = Enable address detection, enable interrupt, and load the receive buffer when the RSR bit is set.

0 = Disable address detection, all bytes are received and the ninth bit can be used as a parity bit.

Asynchronous mode 8-bit (RX9 = 0)

Don't care (any 0 or 1)

FERR: Framing Error bit

1 = Framing error (can be updated by reading the RCREG register and receiving the next valid byte)

0 = No framing error

OERR: Overrun Error bit

1 = Overrun error can be cleared by clearing bit CREN.

0 = No overrun error

RX9D: 9th bit of the Receiving Data

This can be address/data bit or a parity bit and must be calculated by user firmware.



Data Buffer and interrupt Flag for Serial Communication

For transmitting data and reception of data **TXREG** and **RCREG** 8-bit data registers are allocated in PIC18F4550 respectively.



Transmit Register



Receive Register

- When we have to transmit data, we directly copy that data to the TXREG register. After completing the transmission of 8-bit data, the **TXIF** interrupt flag is generated.
- This **TXIF** (transmit interrupt flag) is located in the **PIR1** register. TXIF flag is set when the 8-bit data is transmitted. Then, the buffer is ready to receive another data for transmission.
- Also, **RCIF** (receive interrupt flag) is located in the **PIR1** register. When this flag is set, it indicates that the complete data byte is received by the **RCREG** register. Read the **RCREG** register immediately. Now, the RCREG register is ready to receive another data.
- When the RCIF flag is not set, the PIC microcontroller has to wait for the reception of the complete data byte.

Steps for Programming PIC18F4550 USART

Initialization

1. Initialize the Baud Rate by loading a value into the SPBRG register.
2. Then set bit SPEN in the RCSTA for enabling Serial Port.
3. Then set bit BRGH in the TXSTA for low or high speed.
4. Also clear bit SYNC in the TXSTA register for asynchronous communication.
5. Set bit TXEN in the TXSTA register to enable transmission.
6. Set bit CREN in the RCSTA register to enable reception.



Platforms
(/explore)

Projects
(/projects)

Contests
(/contests)

```
float temp;
TRISC6=0;          /* Make Tx pin as output*/
TRISC7=1;          /* Make Rx pin as input*/

/* Baud rate=9600, SPBRG = (F_CPU /(64*9600))-1*/
temp= (( (float) (F_CPU) / (float) baud_rate ) - 1);
SPBRG = (int) temp;

TXSTA = 0x20;      /* Enable Transmit(TX) */
RCSTA = 0x90;      /* Enable Receive(RX) & Serial */
}
```

Transmit mode

1. Copy the data which we want to transmit into the TXREG register.
2. Monitor the flag TXIF which is set when the transmission is completed.

```
Char USART_TransmitChar (char out)
{
    while (TXIF == 0); /* Wait for transmit interrupt flag*/
    TXREG = out;        /* Write char data to transmit register */
}
```

Receive mode

1. Monitor the flag RCIF until it is set to 1, which indicates a complete 1 byte is received in the RCREG register.
2. Also, check for OERR bit. If it is set then disable and enable CREN.
3. Then, read the RCREG register immediately to avoid overflow.

```
Char USART_ReceiveChar()
{
    while(RCIF==0); /*wait for receive interrupt flag*/
    if(RCSTAbits.OERR)
    {
        CREN = 0;
        NOP();
        CREN=1;
    }
    return(RCREG); /*received in RCREG register and return to main progr
}
```

Application



Platforms
(/explore)

Projects
(/projects)

Contests
(/contests)

```

Generating an echo on PC using PIC18F4550 USART communication module
http://www.electronicwings.com
*/

#include <pic18f4550.h>
#include "Configuration_Header_File.h"
#include "LCD_16x2_8-bit_Header_File.h"

void USART_Init(long);
void USART_TxChar(char);
char USART_RxChar();

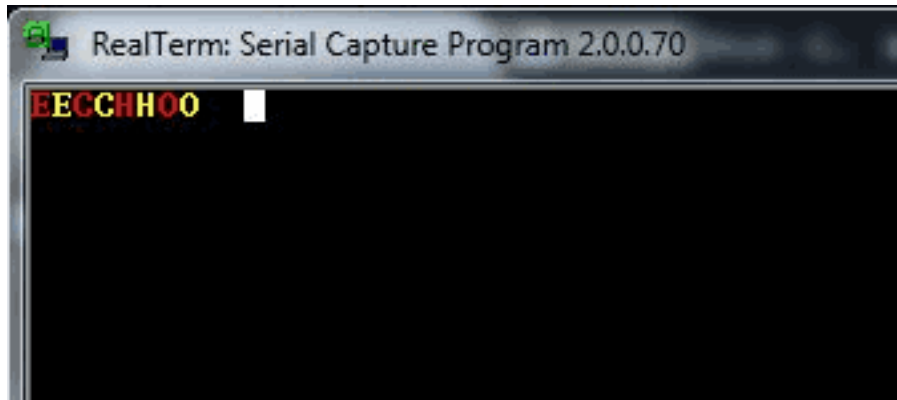
#define F_CPU 8000000/64

#define Baud_value (((float)(F_CPU))/(float)baud_rate)-1)

/*****CODE FOR ECHO GENERATION USING USART*****/
void main()
{
    OSCCON=0x72;
    char data in;

```

Output



USART Communication using Interrupt Service Routine (ISR)

- The above USART communication is done by polling the interrupt flag continuously.
- However, polling the interrupt flag increases power consumption in the microcontroller, and also it is unable to perform other tasks as the interrupt polling is executing continuously.
- Interrupt Service Routine (ISR) executes the application code whenever an interrupt occurs.
- And if the interrupt does not occur, the microcontroller controller can execute other tasks.



Steps for Programming PIC18F4550 USART using Interrupt

Initialization

1. Initialize the Baud Rate by loading value to the SPBRG register.
2. Then set bit SPEN in the RCSTA for Serial Port enable.
3. Then set bit BRGH in the TXSTA for low or high speed.
4. Also, clear bit SYNC in the TXSTA registers for asynchronous communication.
5. Set bit TXEN in the TXSTA register to enable transmission.
6. Set bit CREN in the RCSTA register to enable reception.

Enable GIE, PEIE, RCIE and TXIE for ISR.

```
#define F_CPU 8000000/64

void USART_Init(long baud_rate)
{
    float temp;
    TRISC6=0;          /* Make Tx pin as output*/
    TRISC7=1;          /* Make Rx pin as input*/
    temp=(( float) (F_CPU) / (float) baud_rate) - 1);
    SPBRG = (int) temp; /* Baud rate=9600 SPBRG=(F_CPU/(64*9600))-1*/
    TXSTA = 0x20;      /* TX enable;*/
    RCSTA = 0x90;      /* RX enable and serial port enable*/
    INTCONbits.GIE = 1; /* Enable Global Interrupt */
    INTCONbits.PEIE = 1; /* Enable Peripheral Interrupt */
    PIE1bits.RCIE = 1; /* Enable Receive Interrupt*/
    PIE1bits.TXIE = 1; /* Enable Transmit Interrupt*/
}
```

Application Code for ECHO using ISR

EW

ElectronicWings

Platforms

(/explore)

Projects

(/projects)

Contests


(/contests)

Generating an echo on PC using PIC18F4550 USART ISR

<http://www.electronicwings.com>

Q

+ Project (/publish/project)



```
#include <pic18f4550.h>
#include "Configuration_Header_File.h"
#include "LCD_16x2_8-bit_Header_File.h"

void USART_Init(long);

#define F_CPU 8000000/64
char out;

void main()
{
    OSCCON=0x72;
    LCD_Init();
    USART_Init(9600);
    LCD_String_xy(1,0,"Receive");
    LCD_Command(0xC0);
    while(1);
}
```





([https://www.mouser.in?](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[mouser.in?](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[utm_source=el](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[ectronicswing](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[s&utm_mediu](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[m=display&ut](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[m_campaign=](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[mouser-](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[componentsli](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[sting&utm_co](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[ntent=0x0\)](https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

Components Used

Powered By

PICKit 4 MPLAB
PICKit 4 MPLAB X 1

 ([https://www.mouser.i](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[n/ProductDetail/Micro](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[chip-](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[Technology/PG164140](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[?](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[qs=r5DSvIrkXmLKDuY](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[NJlmlWw%3D%3D&ut](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[m_source=electronic](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[swing&utm_medium=d](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[isplay&utm_campaign](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[=mouser-](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[componentslisting&ut](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)
[m_content=0x0\)](https://www.mouser.in/ProductDetail/Microchip-Technology/PG164140?qs=r5DSvIrkXmLKDuYNJlmlWw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

 [Datasheet \(/componen](/components/pickit-4-mplab/1/datasheet)
[ts/pickit-4-](/components/pickit-4-mplab/1/datasheet)
[mplab/1/dat](/components/pickit-4-mplab/1/datasheet)
[asheet\)](/components/pickit-4-mplab/1/datasheet)

Components Used

Powered By

PIC18f4550

PIC18f4550

X 1

(https://www.mouser.in/ProductDetail/Microchip-Technology/PIC18F4550-I-P?qs=oKK8NaWdAJs8nLDXBGwMXw%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

Datasheet (/components/pic18f4550/1/datasheet)

MAX232 RS-232 Drivers

MAX232 RS-232 Drivers

X 1

(https://www.mouser.com/ProductDetail/Maxim-Integrated/MAX232CSE%2b?qs=1THa7WoU59E0tHin0%252BHRBQ%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

Datasheet (/components/max232-rs-232-drivers/1/datasheet)

EW

ElectronicWings

(/)

Q

MOUSER

+ Project Electronics (/project)

(https://www.mouser.in?utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

Platforms

(/explore)

Projects

(/projects)

Contests

(/contests)

Components Used

Powered By

(https://www.mouser.com/ProductDetail/Adafruit/3309?qs=1JqqoYsYnNdWG1zFt1fzZg%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

USB to Serial Converter CP2104

USB to Serial Converter CP2104

X 1

(https://www.mouser.com/ProductDetail/Adafruit/3309?qs=1JqqoYsYnNdWG1zFt1fzZg%3D%3D&utm_source=electronicswing&utm_medium=display&utm_campaign=mouser-componentslisting&utm_content=0x0)

Downloads

PIC18F4550 USART Communication Project File

Dow (/api/download/platform-attachment/255)ld

Comments

Comment

rodch98

(/users/rodch98/profile)

2018-03-12 01:40:48

To change the external oscillator from 8MHz to 20 Mhz what should i change? i been trying almost everything i could but i just ran out of ideas.

Reply Like

lokeshc

(/users/lokeshc/profile)

2018-03-12 20:09:49

In above program, internal oscillator is used with 8MHz frequency.

Do you want to use external oscillator frequency?

+ Project (/publish/project)



Reply Like 1

Contests

rodch98 (contests)

(/users/rodch98/profile)
2018-03-12 20:52:57

yes, i'm trying with a 20Mhz crystal external oscillator, this code work just fine with 8Mhz external too, but i dont get why doesn't work for 20mhz, i made all calculations but it does work

Reply Like



lokeshc

(/users/lokeshc/profile)
2018-03-12 23:52:08

I have not tried it at 20MHz.

But I am sharing a link of Pdf file for configuration setting manual by Microchip it may help you.

<http://ww1.microchip.com/downloads/en/DeviceDoc/51537a.pdf>

You can find settings related to pic18f4550 mcu. Let me know it works for you or not.

Reply Like

rodch98

(/users/rodch98/profile)
2018-03-15 16:18:21

It worked ! i change somebits i got wrong. Is there any function to receive strings?

Reply Like

lokeshc

(/users/lokeshc/profile)
2018-03-15 19:45:57

No, because we receive character by character.

But you can use receive_char() function to receive string. Or you can create new function as per your requirement.

Reply Like

altgtrmc10

(/users/altgtrmc10/profile)
2019-07-31 01:12:11 • Edited

Can you explain to me how you did it? I'm trying with a 20MHz crystal external oscillator too.

Reply Like

lokeshc

(/users/lokeshc/profile)
2019-07-31 08:37:58

You can refer mentioned document in above comment for external frequency.

Reply Like

mohanraj

(/users/mohanraj/profile)
2018-07-08 04:28:38

i want to get the 11 byte from my sensor & i made this change in my code but i am unable to receive exactly value
if i print the in rx_data[] once the i received data it print wrongly

is this my below code is correct or not ?????

```
char rx_data[]={0}; //global declare
```



```
char USART_RxChar()
```

```
{/}
```



```
while(RCIF==0); /*wait for receive interrupt flag*/
if(RCSTAbits.Overflow)
```

```
CREN = 0;
```

```
NOP();
```

```
CREN=1;
```

```
}
```

```
//return RCREG; /*receive data is stored in RCREG register and return */
```

```
rx_data[i]= RCREG;
```

```
i++;
```

```
}
```

Reply Like

lokeshc

(/users/lokeshc/profile)
2018-07-08 08:31:43



it is looking ok but what are you doing in the main code?

you can directly return RCREG to function which will execute loop to store data into the array.

Reply Like

lokeshc

(/users/lokeshc/profile)
2018-07-10 06:58:16



I can't tell you the exact reason why your code is not showing proper received value.

It seems in your code that you are incrementing 'i' in your main code's loop and in receiver loop too. It may create an issue in proper indexing.

why are you using RCIE bit? because you are not using any ISR routine.

To get proper output simply you can apply loop in your main code which you did right now and send RCREG only from your RX_Char() function. when you receive RCREG in your main code you can store it in your defined array.

Reply Like

geoffrey luora

(/users/geoffrey luora/profile)
2019-08-16 15:51:23



How can one transmit integers or double of float through USART.

Actually converting these (double, float, int) to strings because the transmit function accepts characters.

Reply Like

lokeshc

(/users/lokeshc/profile)
2019-08-18 11:40:47



We can't directly transmit integer or as they are of 8, 16, 32 bit. While we can transmit data on serial interface is 8 or 9 bit. So better to convert integer or float into string and then send it.

While at the receiver side, after receiving string convert it into integer or float that's it. Hope it will help you.

Reply Like

othmanetalmouute

(/users/othmanetalmouute/profile)
2020-03-20 17:26:21



Hi,

Can we use a double usart on same pic18f4550? , need to communicate with gps and gsm a fingerprint in same time

Reply Like 1 🗨️



arunkshirsagar78

(/users/arunkshirsagar78/profile)

2020-06-12 15:26:42

+ Project (/publish/project)



Platforms
(/explore)

Projects
(/projects)

Hey Otomine , you can use pic from similar series which has 2 USARTs
Reply Like 1 1/2

lemowilliams552

(/users/lemowilliams552/profile)

2020-04-11 07:50:51

nice work....really helpful

Reply Like 1 1/2

chathumisamaraweera

(/users/chathumisamaraweera/profile)

2020-10-21 10:06:38

Thank you very much for the tutorial. I used an external 20 Mhz oscillator and RS485 protocol to transmit characters to PC.. Thanks again.. :-)

Reply Like

AdarshD

(/users/AdarshD/profile)

2021-01-08 16:48:21

hi im trying UART with interrupt im getting error: variable has incomplete type 'void'

plz revert back

Reply Like

AdarshD

(/users/AdarshD/profile)

2021-01-08 18:20:11

```
void __interrupt() ISR(void)
{
while(RCIF==0);
out=RCREG; /* Copy received data from RCREG register */
//LCD_Char(out);
while(TXIF==0);
TXREG=out; /* Transmit received(ECHO) to TXREG register */
}
```

after doing changes as above it worked

Reply Like

About Us (/about)

Business Offering (/business-services)

Host Platform (/launch-platform)

Contact Us (/contactus)

Terms of Service (/terms-of-service)

Cookies Policy (/cookie-policy)

Privacy Policy (/privacy-policy)

Connect On:

Facebook(<https://www.facebook.com/electronicwings>)LinkedIn(<https://www.linkedin.com/company/electronicwin>)Youtube(<https://www.youtube.com/channel/UCNdqkukBtk4>)Instagram(https://www.instagram.com/electronicwings_co
igshid=1cip10jjttko)

ElectronicWings

© 2023