Asynchronous communication of PIC16F628 (USART)

On this page, I will explain about USART which is the communication port equipped with PIC16F628.
USART is the initial of Universal Synchronous Asynchronous Receiver Transmitter.
Because this function is equipped with PIC16F873, so some part of pages for PIC16F873 are used in the following explanation.

The USART can be used following communicate mode.

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

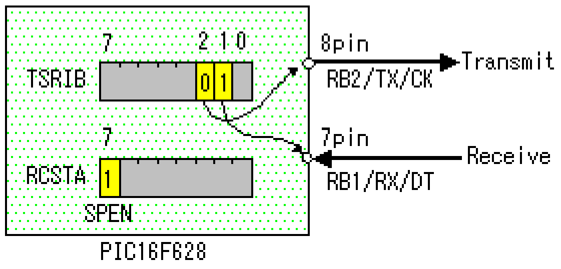
Generally, in the communication of the USART, one data block is composed of 8 bits.
The USART module also has a multi-processor communication capability using 9-bit address detection.

On this page, I will explain about Asynchronous mode communication.

About asynchronous communication

When doing a data communications, the condition of "0" and "1" to have let out from the side of the sending it must be able to be recognized in the receiving side. In the asynchronous communication, it puts a start bit to the head of the transferred data(8 bits or 9 bits) and it puts a stop bit at the end of the data. Recognition in the data block is done by it. The start bit is an L level and the stop bit is the signal of the H level.
The circuit of the condition not to transfer data is H level. When becoming an L level (Start bit) from this condition, the receiving side recognizes that the data transfer begins. After that, according to the signaling speed, the transfer of the data is done. The transfer of the block ends when a stop bit (H level) is detected last. A signaling speed is controlled by the timer which is independent in the sender and the receiving side. So, it isn't possible to do correct communication when there is an error in this timer.
In the asynchronous mode communication of USART, the RX port is used for receiving and the TX port is used for the transmission, so, it is possible to send and receive at the same time. (Full duplex)

Designation of port



In the asynchronous mode, it uses RB2/TX/CK (pin 8) for the data transmission and RB1/RX/DT (pin 7) is used for the data receiving. The mode of the input/output of the TX port, the RX port must be set to the TRISC register to use a USART. Bit 2 of B port is set to output mode (0) and bit 1 is set to input mode (1).

To use RB1 and RB2 as the USART port, the SPEN bit of the RCSTA register must be made "1".

Designation of Asynchronous mode

Asynchronous mode is designated by clearing bit SYNC of the TXSTA register.

Designation of signaling speed

The signaling speed of the USART is controlled by BRG(Baud Rate Generator). BRG is used both the asynchronous and synchronous modes of the USART.
BRG is controlled by the SPBRG register. It is the register to control a free run timer with byte. In case of asynchronous mode, BRGH bit of the TXSTA register is used for the control of the signaling speed too.
When writing a value in the SPBRG register, BRG timer is reset.

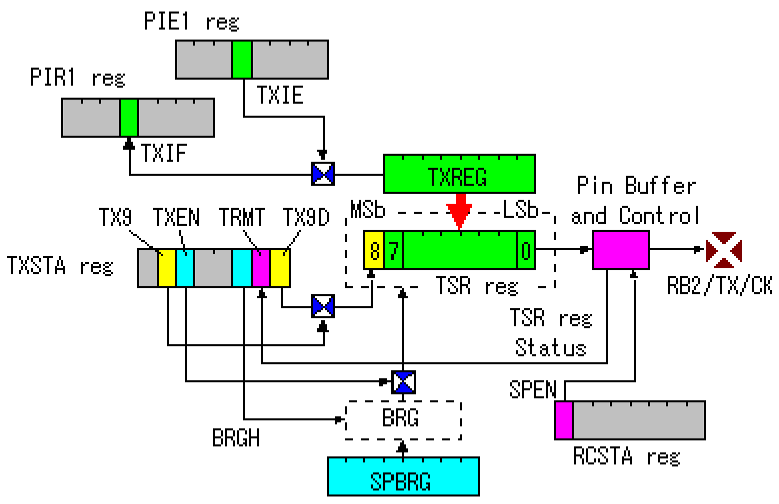
The clock frequency of the PIC and signaling speed can be calculated by the following formula.

Mode	Low-speed (BRGH=0)	High-speed (BRGH=1)
Asynchronous (SYNC=0)	**Fosc/(64(X + 1))**	**Fosc/(16(X + 1))**
Synchronous (SYNC=1)	**Fosc/(4(X + 1))**	

Fosc is the clock frequency of the PIC.
X is the value of SPBRG register. It is 0 to 255

For the details of the value of the SPBRG register, refer to "SFR explanation for PIC16F873 (17)".

Transmitter operation



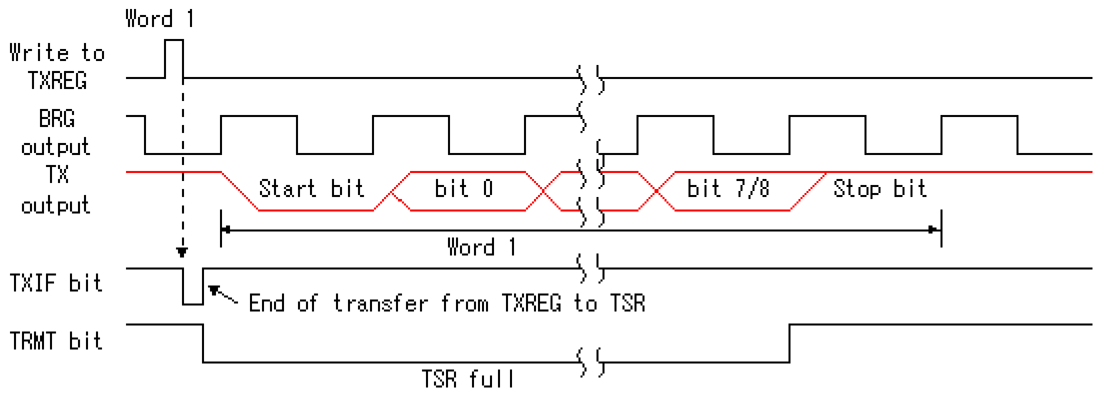
When doing an asynchronous transmission with the USART, it makes the SPEN bit of the RCSTA register "1" and it makes RB2 port as TX port.
When loading a transmit data to the TXREG register, it is transferred to the TSR (Transmit Shift Register) by the hardware and it is transmitted from the TX port.
The transmission of the data is done by setting the TXEN bit of the TXSTA register.
LSb(Least Significant bit) is transmitted first.
When the contents of the TXREG are sent to the TRS, the TXIF bit of the PIR1 register becomes "1" and the interruption occurs. But, the TX1E bit of the PIE1 register must become "1". This interruption means that the contents of the TXREG has been sent to the TSR and it

becomes the condition that it is possible to load data in the TXREG. The data can be continuously transmitted if detecting this interruption and setting the following data to the TXREG.(Back to Back) The TXIF bit can not be cleared in software. The TXIF bit is cleared when data is loaded in the TXREG.
The TRMT bit of the TXSRA register is set to "1" when the data of the TSR register is transmitted. This bit doesn't have an interruption logic. So, to confirm the empty of the TSR register, you should poll TRMT bit periodically.
Because the TSR register is not mapped in data memory, you can not read/write the contents of this register directly. A signaling speed is controlled by BRG(Baud Rate Generator).
Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). In the case, ninth bit as parity is transmitted by setting parity bit contents to the TX9D bit of the TXSTA register and setting the TX9 bit. The TX9D bit must be set before setting the data to TXREG. A transmission is started as soon as loading data in the TXREG. It isn't normally transmitted when the ninth bit isn't set to TX9D before it.
When clearing TXEN bit while transmitting data, a transmitter is reset and the RB2/TX/CK pin becomes a high impedance condition.

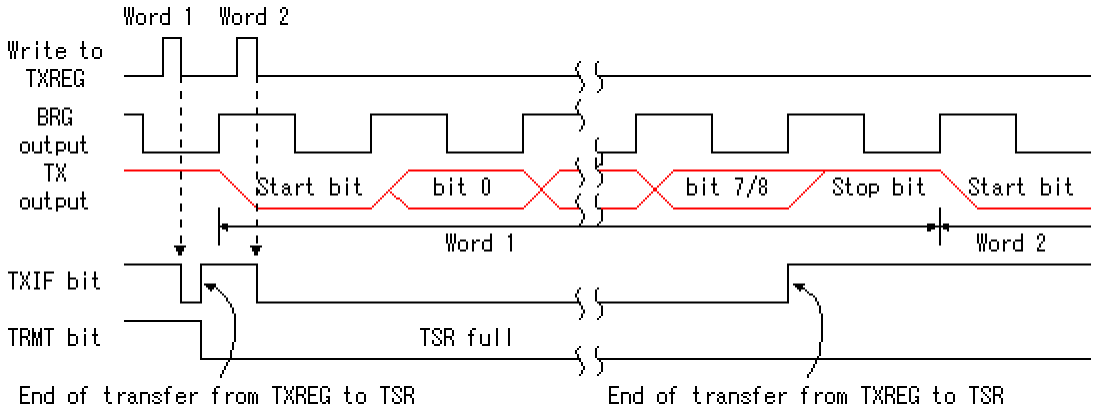
Steps to follow when setting up an Asynchronous Transmission

1. Initialize the SPBRG register for the appropriate baud rate.
   If a high speed baud rate is desired, set bit BRGH.
2. Enable the asynchronous serial port by clearing SYNC bit of the TXSTA register and setting SPEN bit of the RCSTA register.
3. If interrupts are desired, then set enable bit TXIE of the PIE1 register.
4. If 9-bit transmission is desired, then set transmit bit TX9 of the TXSTA register.
5. Enable the transmission by setting bit TXEN of the TXSTA register.
   In this point, the TXREG register is in the condition to write that is possible and TXIF bit of the PIR1 register becomes "1".
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register.
   (Start transmission)

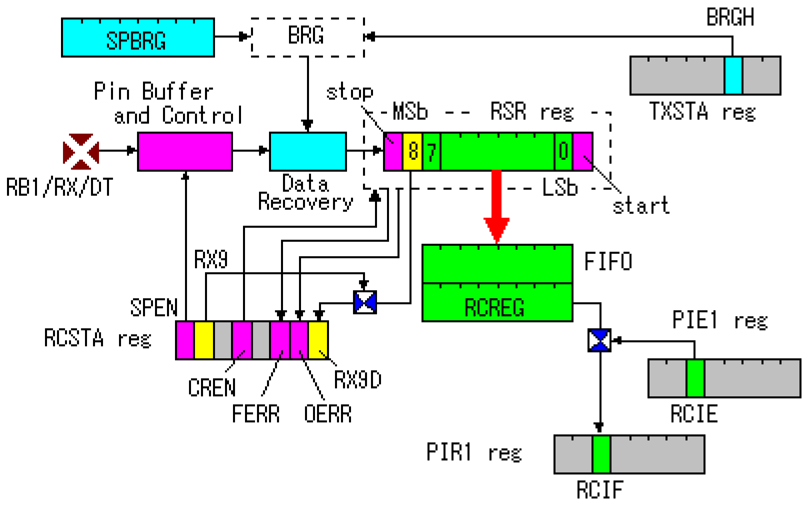
Transmission time chart of 1 Word



Transmission time chart of the continuation transmission (Back to Back)



Receiver operation

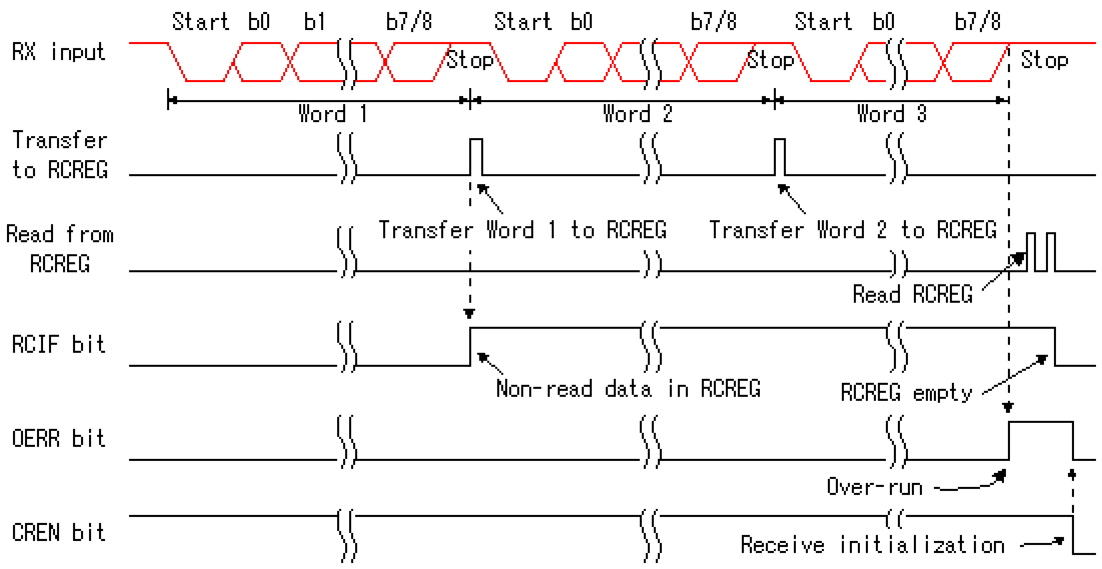


When doing an asynchronous transmission with the USART, it makes the SPEN bit of the RCSTA register "1" and it makes RB1 port as RX port. The data which was received at the RX port is received with the RSR register through the Data recovery circuit. Data at the RX port is sampled three times to do the judgment of the H level or the L level. A received data is stored in the RSR register according to the signaling speed which is specified by the SPBRG register and the BRGH bit of the TXSTA register.
When detecting a stop bit, the contents of the RSR register are transferred to the RCREG register. The RCIF bit of the PIR1 register is set when data is stored in the RCREG register and the interruption occurs. To make interruption occur, the RCIE bit of the PIE1 register must be set beforehand.
The RCREG register is composed in two FIFO(First In First Out) buffers and can store data for 2 block. This is the protection when delay in the reading processing by the software. RCIF bit is read only bit and is cleared with the hardware when all RCREG registers are read.
If both RCREG registers are full(It isn't read) when the receiving completes with the RSR register, the OERR bit of the RCSTA register is set and the occurring of the overrun error is expressed. The data which was stored in the RSR register at this time is lost. Also, receiving operation isn't done. It clears the CREN bit of the RCSTA register once and it sets to return this condition normally again. By this operation, OERR bit is cleared.
The FERR bit of the RCSTA register is set when the frame error detects with the RSR register. The RX9D bit and the FERR bit which is stored in the RCSTA register are rewritten every time it receives data in 1 block. So, the FERR bit of the RCSTA register must be checked before reading the contents of the RCREG register. When receiving a normal frame behind the error frame, the information of FERR disappears. So, the normality of the received data can not be judged.

Steps to follow when setting up an Asynchronous Reception

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH of the TXSTA register.
2. Enable the asynchronous serial port by clearing bit SYNC of the TXSTA register and setting bit SPEN of the RCSTA register.
3. If interrupts are desired, then set enable bit RCIE of the PIE1 register.
4. If 9-bit reception is desired, then set bit RX9 of the RCSTA register.
5. Enable the reception by setting bit CREN of the RCSTA register.
6. Flag bit RCIF of the PIR1 register will be set when reception is complete and an interrupt will be generated if enable bit RCIE of the PIE1 register is set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN of the RCSTA register.

Receiving time chart

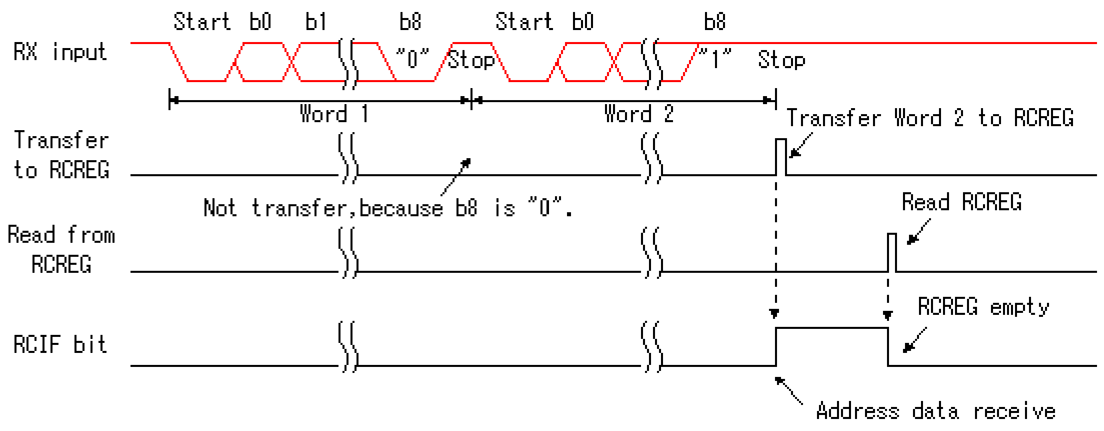


This time chart shows the situation which the over-run occurs.
In usual operation, because the contents of RCREG are read by the interruption of RCIF, the over-run doesn't occur.

Address detection

The 9th bit can be used for the address data detection. I don't understand a way of using address data.
The operation is as follows. This operation is effective in case of 9-bit transfer.
Set RX9 bit and ADDEN bit of the RCSTA register. In this way, only when the

data of the 9th bit is "1", received data is stored in the RCREG register. The data that the 9th bit is "0" isn't stored.


RX input
Start
b0
b1
b8
"0"
Stop
Start
b0
b8
"1"
Stop
Word 1
Word 2
Transfer to RCREG
Transfer Word 2 to RCREG
Not transfer,because b8 is "0".
Read from RCREG
Read RCREG
RCREG empty
RCIF bit
Address data receive