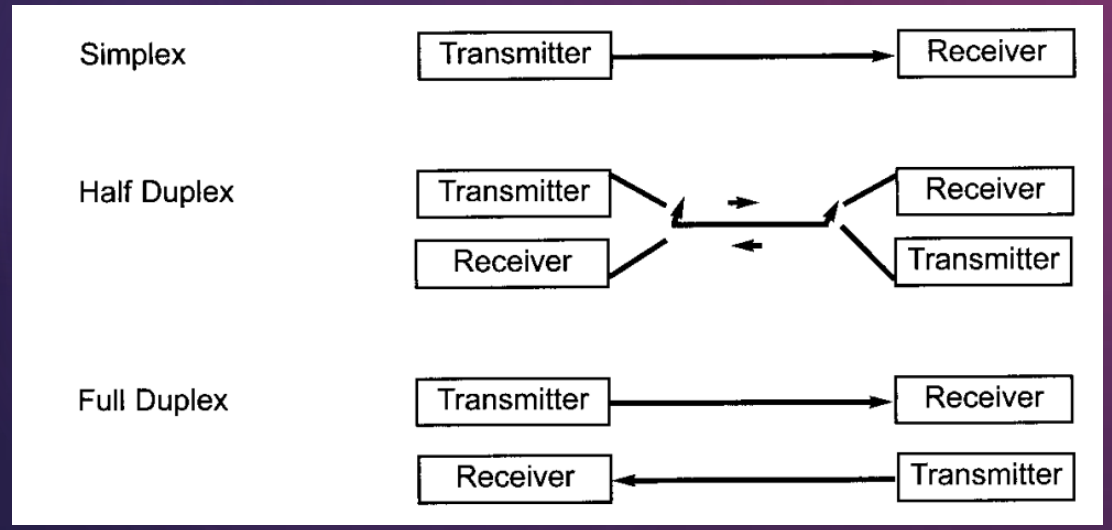
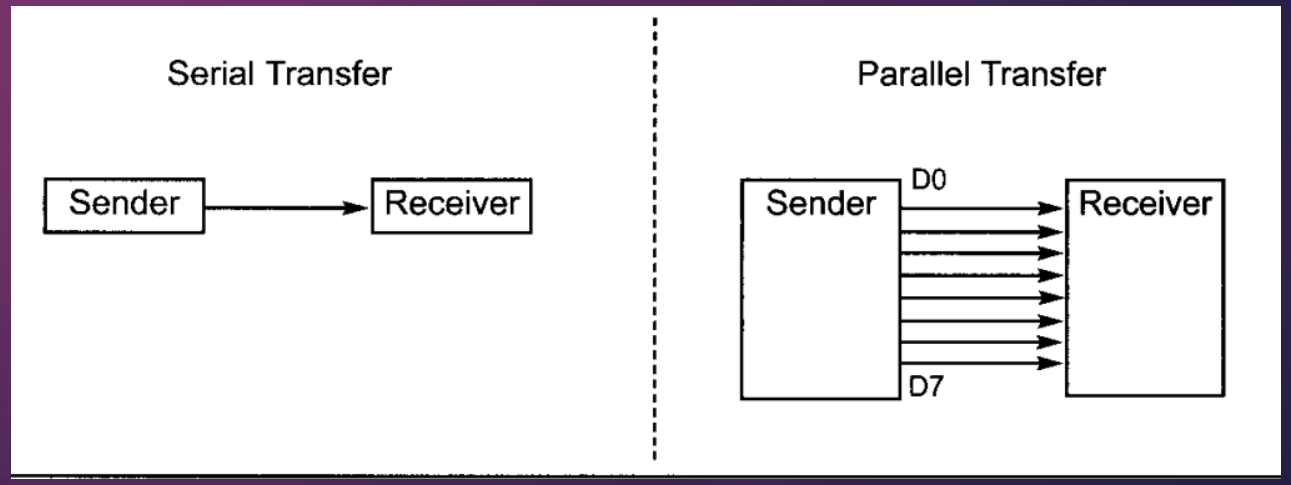
Microprocessor Interfacing & Programming

LECTURE 23 & 24

Basics of Serial Communication

- When a µP communicates with the outside world, it provides the data in byte-sized chunks.
- PIC18 chip has built in USART (universal synchronous asynchronous receiver transmitter).





Asynchronous serial communication and Data Framing

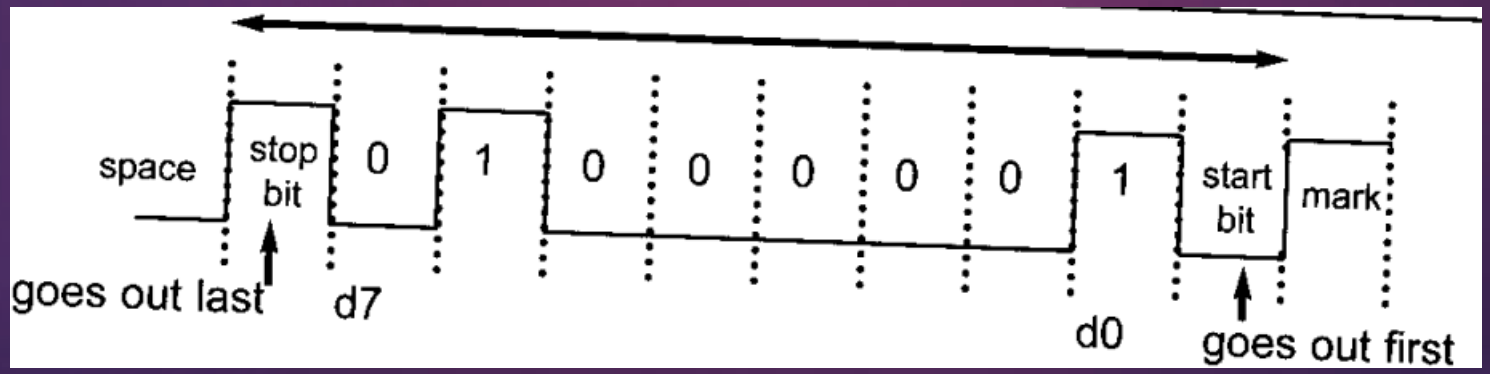
- The data coming in at the receiving end of the data line in a serial data transfer is all 0s and 1s.
- The sender and receiver must agree on a set of rules, a protocol , on how data is packed, hoe many bits constitute a character, and when the data begins and ends.

Start and Stop bits:

- In the asynchronous method, each character is placed between start and stop bits. This is called **framing**.
- The start bit is always one bit but the stop bit can be one or two bits.
- The start bit is always 0 (low) and the stop bit is 1(high).

Framing

- When there is no transfer, the signal is 1 (high), which is refereed to as **mask**.
- The 0 (low) is referred to as **space**.
- The transmission begins with a start bit followed by D0, the LSB, then the rest of the bits until the MSB (D7), and finally, the one stop bit indicating the end of the character.



Data transfer rate and UART Programming in Assembly

- 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**.
- PIC18 has both the synchronous and asynchronous features.

Asynchronous mode:

- In PIC18, six major registers are associated with UART.

1. SPBGR (serial port baud rate generator)
2. TXREG (Transfer register0
3. RCREG (Receive register)
4. TXSTA (transmit status and control register)
5. RCSTA (receive status and control register)
6. PIR1 (peripheral interrupt request register 1)

SPBRG register and baud rate

- PIC18 transfers and receives data serially at different baud rates.
- The baud rate in PIC18 is programmable.
- This is done by the help of 8 bit register called SPBRG.
- For a given crystal frequency, the value loaded into the SPBRG decided the baud rate.
- The relation between the value loaded into SPBRG 'X' and the Fosc (frequency of oscillator connected to OSC1 and OSC2 pins) is dictated by the following formula:

$$\text{Desired Baud Rate} = Fosc/(64X + 64) = Fosc/64(X + 1)$$

$$\text{Desired Baud Rate} = Fosc/64(X + 1) = 10\ MHz/64(X + 1) = 6250\ Hz/(X + 1)$$

To get the X value for different baud rates we can solve the equation as follows:

$$X = (156250/\text{Desired Baud Rate}) - 1$$

Value to be loaded in SPBRG

- As we know that PIC18 divides Fosc by 4 to get the instruction cycle time frequency.
- In case XTAL= 10MHz, the instruction cycle frequency is 2.5 MHz
- The PIC18's UART circuitry divides the instruction cycle frequency by 16 once more before it is used by an internal timer to set the baud rate.
- Therefore, 2.5 MHz divided by 16 gives 156,250 Hz. This is the number we use to find the SPBRG value.

Baud Rate	SPBRG (Decimal Value)	SPBRG (Hex Value)
38400	3	3
19200	7	7
9600	15	F
4800	32	20
2400	64	40
1200	129	81

Note: For Fosc = 10 MHz we have SPBRG = (156,250/BaudRate) – 1

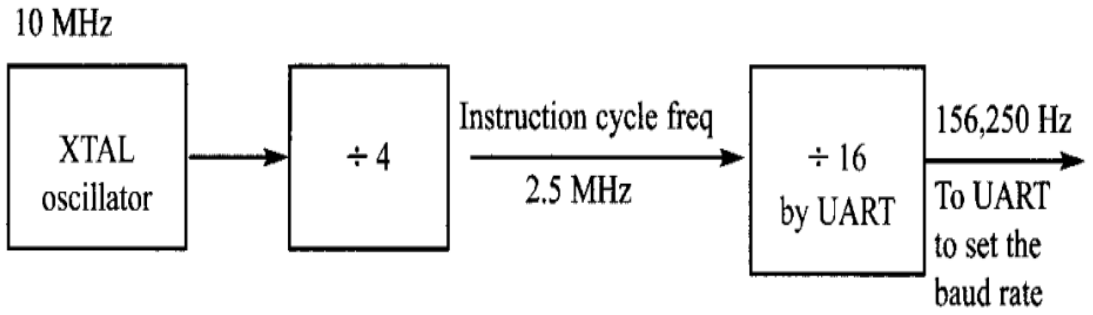
With Fosc = 10 MHz, find the BGRP value needed to have the following baud rates:
(a) 9600 (b) 4800 (c) 2400 (d) 1200

Solution:

Because Fosc = 10 MHz, we have 10 MHz/4 = 2.5 MHz for the instruction cycle frequency. This is divided by 16 once more before it is used by UART. Therefore, we have 2.5 MHz/16= 156250 Hz and X = (156250 Hz/Desired Baud Rate) – 1:

(a) (156250/ 9600) – 1 = 16.27 – 1 = 15.27 = 15 = F (hex) is loaded into SPBRG
(b) (156250/ 4800) – 1 = 32.55 – 1 = 31.55 = 32 = 20 (hex) is loaded into SPBRG
(c) (156250/ 2400) – 1 = 65.1 – 1 = 64.1 = 64 = 40 (hex) is loaded into SPBRG
(d) (156250/ 1200) – 1 = 130.2 – 1 = 129.2 = 129 = 81 (hex) is loaded into SPBRG

Notice that dividing the instruction cycle frequency by 16 is the setting upon Reset. We can get a higher baud rate with the same crystal by changing this default setting. This is done by making bit BRGH = 1 in the TXSTA register. This is explained at the end of this section.



Baud Rate	SPBRG (Decimal Value)	SPBRG (Hex Value)
19200	2	2
9600	5	5
4800	12	0C
2400	25	19
1200	51	33

Note: For Fosc = 4 MHz we have 4 MHz/4 = 1 MHz for instruction cycle freq. The frequency used by the UART is 1 MHz/16 = 62,500 Hz. That means SPBRG = (62500/Baud Rate) – 1

TXREG register

- It is another 8-bit register used for serial communication in PIC18.
- For a byte of data to be transferred via TX pin, it must be placed in the TXREG register.
- TXREG is a special function register (SFR) and can be accesses like any other register in PIC18.
- The moment a byte is written into TXREG, it is fetched into a register called TSR (transmit shift register).
- TSR frames the 8 bit data with start and stop bits and the 10 bit data is transferred serially via the TX pin.
- TXREG is accessibly by the programmer, TSR is not accessible and is strictly for internal use.

```
MOVLW 0x41      ;WREG=41H, ASCII for letter 'A'
MOVWF TXREG     ;copy WREG into TXREG

MOVFF PORTB,TXREG ;copy PORTB contents into TXREG
```

RCREG register

- Similarly, when the bits are received serially via the RX pin, PIC18 deframes them eliminating the stop and start bits, making a byte out of the data received, and then placing it in the RCREG register.

- The following code will dump the received byte into PORTB:

```
MOVFF RCREG,PORTB ;copy RXREG to PORTB
```

TXSTA and RCSTA registers

- TXSTA is an 8-bit register used to select the synchronous/asynchronous modes and data framing size.
- The BRGH bit is used to select a higher speed for transmission.
- The default is lower baud rate transmission.
- D6 of TXSTA register determines the framing of data by specifying the number of bits per character.
- We use an 8-bit data size.
- RCSTA is also an 8-bit register used to enable the serial port to receive data.

CSRC	TX9	TXEN	SYNC	0	BRGH	TRMT	TX9D

CSRC D7 Clock Source Select (not used in asynchronous mode, therefore D7 = 0.)

TX9 D6 9-bit Transmit Enable
1 = Select 9-bit transmission
0 = Select 8-bit transmission (We use this option, therefore D6 = 0.)

TXEN D5 Transmit Enable
1 = Transmit Enabled
0 = Transmit Disabled
We turn "on" and "off" this bit in order to start or stop data transfer.

SYNC D4 USART mode Select (We use asynchronous mode, therefore D4 = 0.)
1 = Synchronous
0 = Asynchronous

0 D3

BRGH D2 High Baud Rate Select
0 = Low Speed (Default)
1 = High Speed
We can double the baud rate with the same Fosc. See the end of this section for further discussion on this bit.

TRMT D1 Transmit Shift Register (TSR) Status
1 = TSR empty
0 = TSR full

The importance of the TSR register. To transfer a byte of data serially, we write it into TXREG. The TSR (transmit shift register) is an internal register whose job is to get the data from the TXREG, frame it with the start and stop bits, and send it out one bit at a time via the TX pin. When the last bit, which is the stop bit, is transmitted, the TRMT flag is raised to indicate that it is empty and ready for the next byte. When TSR fetches the data from TXREG, it clears the TRMT flag to indicate it is full. Notice that TSR is a parallel-in-serial-out shift register and is not accessible to the programmer. We can only write to TXREG. Whenever the TSR is empty, it gets its data from TXREG and clears the TXREG register immediately, so it does not send out the same data twice.

TXD9 D0 9th bit of Transmit Data (Because we use the 8-bit option, we make D0 = 0)
Can be used as an address/data or a parity bit in some applications

SPEN	RX9	SREN	CREN	ADDE	FERR	OERR	RX9D

SPEN D7 Serial port enable bit
1 = Serial port enabled, which makes TX and RX pins as serial port pins
0 = Serial port disabled

RX9 D6 9-bit Receive enable bit
1 = Select 9-bit reception
0 = Select 8-bit reception (We use this option; therefore, D6 = 0.)

SREN D5 Single receive enable bit (not used in asynchronous mode D5 = 0)

CREN D4 Continuous receive enable bit
1 = Enable continuous Receive (in asynchronous mode)
0 = Disable continuous Receive (in asynchronous mode)

ADDEN D3 Address delete enable bit (Because used with the 9-bit data frame D3 = 0)

FERR D2 Framing error bit
1 = Framing error
0 = No Framing error

OERR D1 Overrun error bit
1 = Overrun error
0 = No overrun error

TXD9 D0 9th bit of Receive data (Because we use the 8-bit option, we make D0 = 0)
Can be used as an address/data or a parity bit in some applications.

PIR1 register

- Two of the PIR1 register bits are used by UART.
- They are TXIF (transmit interrupt flag) and RCIF (receive interrupt flag).
- We monitor TXIF flag bit to make sure that all bits of the last byte are transmitted before we write another byte into the TXREG.
- By the same logic, we monitor RCIF flag to see is a byte of data has come in yet.

---	--	RCIF	TXIF	--	---	--	

RCIF Receive interrupt flag bit
1 = The UART has received a byte of data and it is sitting in the RCREG register (receive buffer), waiting to be picked up.
Upon reading the RCREG register, the RCIF is cleared to allow the next byte to be received.
0 = The RCREG is empty.

TXIF Transmit interrupt flag bit
0 = The TXREG register is full.
1 = The TXREG (transmit buffer) register is empty.

The importance of TXIF: To transmit a byte of data, we write it into TXREG. Upon writing a byte into TXREG, the TXIF flag is cleared. When the entire byte is transmitted via the TX pin, the TXIF flag bit is raised to indicate that it is ready for the next byte. So, we must monitor this flag before we write a new byte into TXREG, otherwise, we wipe out the last byte before it is transmitted.

Programming PIC18 to transfer data serially

1. The TXSTA register is loaded with the value 20H, indicating asynchronous mode with 8-bit data frame, low baud rate, and transmit enabled.
2. Make TX pin (RC6) an output for data to come out of the PIC.
3. The SPBRG is loaded with any value to set the baud rate.
4. SPEN bit in RCSTA register is set HIGH to enable the serial port of PIC18.
5. The byte to be transmitted serially is written into the TXREG register.
6. Monitor the TXIF bit of PIR1 register to make sure UART is ready for next byte.
7. To transfer the next byte, go to step 5.

Example

Write a program for the PIC18 to transfer the letter 'G' serially at 9600 baud, continuously. Assume XTAL = 10 MHz.

Solution:

```
        MOVLW B'00100000'  ;enable transmit and choose low baud rate
        MOVWF TXSTA        ;write to reg
        MOVLW D'15'        ;9600 bps (Fosc / (64 * Speed) - 1)
        MOVWF SPBRG        ;write to reg
        BCF TRISC, TX      ;make TX pin of PORTC an output pin
        BSF RCSTA, SPEN    ;enable the entire serial port of PIC18
OVER    MOVLW A'G'         ;ASCII letter 'G' to be transferred
S1      BTFSS PIR1, TXIF   ;wait until the last bit is gone
        BRA S1             ;stay in loop
        MOVWF TXREG        ;load the value to be transferred
        BRA OVER           ;keep sending letter 'G'
```

Write a program to transmit the message "YES" serially at 9600 baud, 8-bit data, and 1 stop bit. Do this forever.

Solution:

```
        MOVLW B'00100000'       ;enable transmit and choose low baud
        MOVWF TXSTA             ;write to reg
        MOVLW D'15'             ;9600 bps (Fosc / (64 * Speed) - 1)
        MOVWF SPBRG             ;write to reg
        BCF TRISC, TX           ;make TX pin of PORTC an output pin
        BSF RCSTA, SPEN         ;enable the serial port
OVER    MOVLW A'Y'              ;ASCII letter 'Y' to be transferred
        CALL  TRANS
        MOVLW A'E'              ;ASCII letter 'E' to be transferred
        CALL  TRANS
        MOVLW A'S'              ;ASCII letter 'S' to be transferred
        CALL  TRANS
        MOVLW 0x0               ;NULL to purge the buffer
        CALL TRANS
        BRA   OVER              ;keep doing it
TRANS ;----serial data transfer subroutine
S1      BTFSS PIR1, TXIF        ;wait until the last bit is gone
        BRA S1                  ;stay in loop
        MOVWF TXREG             ;load the value to be transmitted
        RETURN                  ;return to caller
```

Importance of TXIF flag

1. The byte character to be transmitted is written into the TXREG register.
2. The TXIF flag is set to 1 internally to indicate that TXREG has a byte and will not accept another byte until this one is transmitted.
3. The TSR (Transmit Shift Register) reads the byte from TXREG and begins to transfer the byte starting with the start bit.
4. The TXIF is cleared to indicate that the last byte is being transmitted and TXREG is ready to accept another byte.
5. The 8-bit character is transferred one bit at a time.
6. By monitoring the TXIF flag, we make sure that we are not overloading the TXREG register. If we write another byte into the TXREG register before the TSR has fetched the last one, the old byte could be lost before it is transmitted.

Programming PIC18 to recieve data serially

1. The RCSTA register is loaded with 90H, to enbale the continuous receive in addtion to the 8-bit data size option.
2. The TXSTA register is loaded with the value 00H to choose the low baud rate option.
3. SPBRG is loaded with any value to set the baud rate.
4. Make RX pin (RC7) an input for data to come into PIC18.
5. The RCIF flag bit of the PIR1 register is monitored for a HIGH to see if an entire byte has been received yet.
6. When RCIF is raised, the RCREG register has the byte. Its contents are moved into a safe place.
7. To receive the next byte, go to step 5.

Example

Program the PIC18 to receive bytes of data serially and put them on PORTB. Set the baud rate at 9600, 8-bit data, and 1 stop bit.

Solution:

```
      MOVLW B'10010000'       ;enable receive and serial port itself
      MOVWF RCSTA             ;write to reg
      MOVLW D'15'             ;9600 bps (Fosc / (64 * Speed) - 1)
      MOVWF SPBRG             ;write to reg
      BSF   TRISC, RX         ;make RX pin of PORTC an input pin
      CLRF  TRISB             ;make port B an output port
;get a byte from serial port and place it on PORTB
R1    BTFSS PIR1, RCIF        ;check for ready
      BRA   R1                ;stay in loop
      MOVFF RCREG, PORTB      ;save value into PORTB
      BRA   R1                ;keep doing that
```

Importance of RCIF flag

1. It receives the start bit indicating that the next bit is the first bit of the character byte it is about to receive.
2. The 8-bit character is received one bit at time. When the last bit is received, a byte is formed and placed in RCREG.
3. The stop bit is received. It is during receiving the stop bit that the PIC18 makes RCIF = 1, indicating that an entire character byte has been received and must be picked up before it gets overwritten by another incoming character.
4. By checking the RCIF flag bit when it is raised, we know that a character has been received and is sitting in the RCREG register. We copy the RCREG contents to a safe place in some other register or memory before it is lost.
5. After the RCREG contents are read (copied) into a safe place, the RCIF flag bit is forced to 0 by the UART itself. This allows the next received character byte to be placed in RCREG, and also prevents the same byte from being picked up multiple times.

Quadrupling the baud rate in PIC18

- There are 2 ways to increase the baud rate of data transfer:

1. Use a high frequency crystsl.
2. Change a bit in TXSTA register, as shown below.

Baud rates for BRGH = 0

When BRGH = 0, the PIC18 divides Fosc/4 (crystal frequency) by 16 once more and uses that frequency for UART to set the baud rate. In the case of XTAL = 10 MHz we have:

```
Instruction cycle freq. = 10 MHz / 4 = 2.5 kHz
and
2.5 MHz / 16 = 156,250 Hz because BRGH = 0
```

Baud rates for BRGH = 1

With the fixed crystal frequency, we can quadruple the baud rate by making BRGH = 1. When the BRGH bit (D2 of the TXSTA register) is set to 1, Fosc/4 of XTAL is divided by 4 (instead of 16) once more, and that is the frequency used by UART to set the baud rate. In the case of XTAL = 10 MHz, we have:

```
Instruction cycle freq. = 10 MHz / 4 = 2.5 MHz
and
2.5 MHz / 4 = 625000 Hz because BHRG = 1
```

This is the frequency used by UART to set the baud rate if BHRH = 1.

Find the SPBRG value (in both decimal and hex) to set the baud rate to each of the following:

(a) 9600 if BRGH = 1 (b) 4800 if BRGH = 1

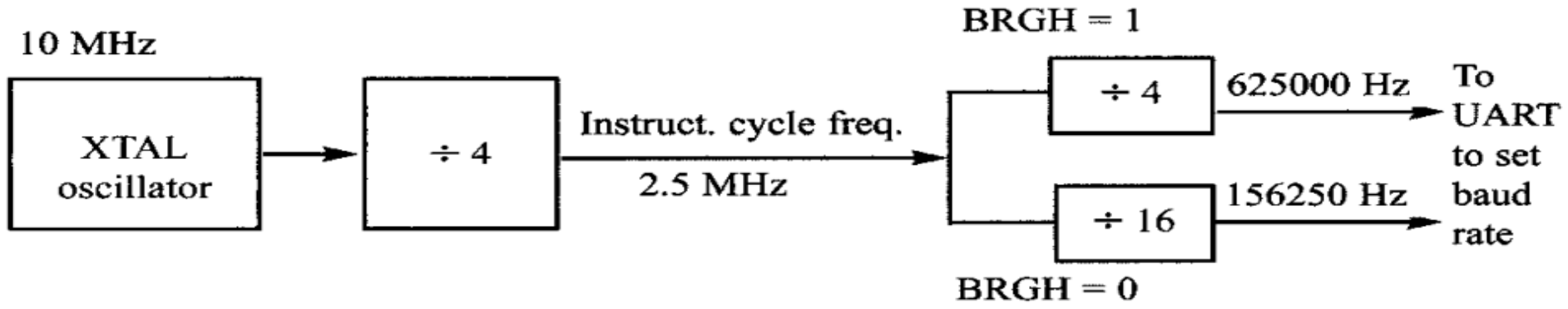
Assume that XTAL = 10 MHz.

Solution:

With XTAL = 10 MHz, Fosc/4 = 2.5 MHz. Because BRGH = 1, we have UART frequency = 2.5 MHz/4 = 625,000 Hz.

(a) (625,500 / 9600) – 1 = 64; therefore, SPBRG = 64 or SPBRG = 40H (in hex).

(b) (625,500 / 4800) – 1 = 129; therefore, SPBRG = 129 or SPBRG = 81H (in hex).



Example

Write a program for the PIC18 to transfer the letter 'G' serially at 57600 baud, continuously. Assume XTAL = 10 MHz. Use the BRGH = 1 mode

Solution:

```
        MOVLW B'00100100'  ;enable transmit and choose high baud rate
        MOVWF TXSTA        ;write to reg
        MOVLW D'10'        ;57600 bps (Fosc / (16 * Speed) - 1)
        MOVWF SPBRG        ;write to reg
        BCF TRISC, TX      ;make TX pin of PORTC an output pin
        BSF RCSTA, SPEN    ;enable the entire Serial port of PIC18
OVER    MOVLW A'G'         ;ASCII letter 'G' to be transferred
S1      BTFSS PIR1, TXIF   ;wait until the last bit is gone
        BRA S1             ;stay in loop
        MOVWF TXREG        ;load the value to be transferred
        BRA OVER           ;keep sending letter 'G'
```

Baud rate error calculation

- In calculating the baud rate, we have used the integer part for the SPBRG register because PIC microcontrollers can only use integer values.
- By dropping the decimal portion of the calculated values we run the risk of introducing error into the baud rate.

Ways of calculating this error:

Error = (Calculated value for the SPBRG – Integer part)/Integer part

For example, with the XTAL = 10 MHz and BRGH = 0 we have the following for the 9600 baud rate:

SPBRG value = (156250/9600) – 1 = 16.27 – 1 = 15.27 = 15

and the error is

(15.27 – 15)/16 = 1.7%

Another way to calculate the the error rate is as follows:

Error = (calculated baud rate – desired baud rate) / desired baud rate

Assuming XTAL = 10 MHz, calculate the baud rate error for the following:
(a) 2400 (b) 1200 (c) 19200 (d) 57600
Use the BRGH = 0 mode.

Solution:

(a) SPBRG Value = (156250/2400) – 1 = 65.1 – 1 = 64.1 = 64

Error = (64.1 – 64)/ 65 = 0.15%

(b) SPBRG Value (156250/1200) – 1 = 130.2 – 1 = 129.2 = 129

Error = (129.2 – 129)/130 = 0.15%

(c) SPBRG Value (156250/19200) – 1 = 8.138 – 1 = 7.138 = 7

Error = (7.138 – 7)/8 = 1.7%

(d) SPBRG Value (156250/57600) – 1 = 2.71 – 1 = 1.71 = 1

Error = (1.71 – 1)/2 = 35%

Such an error rate is too high. Let's round up the number to see what happens.
Error = (3 – 2.7)/3 = 10% This means we use SPBRG = 2 instead of SPBRG = 1.

Assuming XTAL = 10 MHz, calculate the baud rate error for the following:

(a) 2400 (b) 1200

Assume BRGH = 0

Solution:

(a) SPBRG Value = (156250/2400) – 1 = 65.1 – 1 = 64.1 = 64

and calculated baud rate is 156250/(64 + 1) = 2403

Error = (2403 – 2400)/2400 = 0.12%

(b) SPBRG Value (156250/1200) – 1 = 130.2 – 1 = 129.2 = 129

where the calculated baud rate is 156250/(129 + 1) = 1202

Error = (1202 – 1200)/1200 = 0.16%

Table 10-10: SPBRG Values for Various Baud Rates (XTAL = 10 MHz)

	BRGH = 0		BRGH = 1	
Baud Rate	**SPBRG**	**Error**	**SPBRG**	**Error**
38400	3	1.5%	15	1.7%
19200	7	1.7%	32	1.3%
9,600	15	1.7%	64	0.15%
4,800	32	1.3%	129	0.15%
SPBRG = (156250/Baud rate) – 1			SPBRG = (625000/Baud rate) – 1	

Table 10-11: SPBRG Values for Various Baud Rates (XTAL = 4 MHz)

	BRGH = 0		BRGH = 1	
Baud Rate	**SPBRG**	**Error**	**SPBRG**	**Error**
19200	2	8.3%	12	0.15%
9,600	6	8%	25	0.15%
4,800	12	0.15%	51	0.15%
2,400	25	0.16%	103	0.16%
SPBRG = (62500/Baud rate) – 1			SPBRG = (250000/Baud rate) – 1	

Serial port programming in C

Write a C program for the PIC18 to transfer the letter 'G' serially at 9600 baud, continuously. Use 8-bit data and 1 stop bit. Assume XTAL = 10 MHz.

Solution:

```
#include <P18F4580.h>
void main(void)
   {
     TXSTA=0x20;                  //choose low baud rate,8-bit
     SPBRG=15;                    //9600 baud rate/ XTAL = 10 MHz
     TXSTAbits.TXEN=1;
     RCSTAbits.SPEN=1;

     while(1)
        {
          TXREG='G';              //place value in buffer
          while(PIR1bits.TXIF==0);      //wait until all gone
        }
   }
```

Write a PIC18 C program to transfer the message "YES" serially at 9600 baud, 8-bit data, and 1 stop bit. Do this continuously.

Solution:

```
#include <P18F458.h>
void SerTx(unsigned char);
void main(void)
   {
     TXSTA=0x20;                  //choose low baud rate,8-bit
     SPBRG=15;                    //9600 baud rate/ XTAL = 10 MHz
     TXSTAbits.TXEN=1;
     RCSTAbits.SPEN=1;
     while(1)
        {
          SerTx('Y');
          SerTx('E');
          SerTx('S');
        }
   }
void SerTx(unsigned char c)
   {
     while(PIR1bits.TXIF==0);     //wait until transmitted
     TXREG=c;                     //place character in buffer
   }
```

Program the PIC18 in C to receive bytes of data serially and put them on PORTB. Set the baud rate at 9600, 8-bit data, and 1 stop bit.

Solution:

```
#include <P18F458.h>
void main (void)
  {
    TRISB = 0;                          //PORTB an output
    RCSTA=0x90;                         //enable serial port and receiver
    SPBRG=15;                           //9600 baud rate/ XTAL = 10 MHz
    while(1)                            //repeat forever
      {
        while(PIR1bits.RCIF==0);        //wait to receive
        PORTB=RCREG;                    //save value
      }
  }
```

Write an C18 program to send two different strings to the serial port. Assuming that SW is connected to pin PORTB.5, monitor its status and make a decision as follows:
SW = 0: send your first name
SW = 1: send your last name
Assume XTAL = 10 MHz, baud rate of 9600, and 8-bit data.

Solution:

```
#include <P18F458.h>
#define MYSW PORTBbits.RB5          //INPUT SWITCH
void main(void)
   {
      unsigned char z;
      unsigned char fname[]="ALI";
      unsigned char lname[]="SMITH";
      TRISBbits.TRISB5 = 1;   //an input
      TXSTA=0x20;             //choose low baud rate, 8-bit
      SPBRG=15;               //9600 baud rate/ XTAL = 10 MHz
      TXSTAbits.TXEN=1;
      RCSTAbits.SPEN=1;
      if(MYSW==0)                                 //check switch
         {
            for(z=0;z<3;z++)                      //write name
               {
                  while(PIR1bits.TXIF==0);  //wait for transmit
                  TXREG=fname[z];           //place char in buffer
               }
         }
      else
         {
            for(z=0;z<5;z++)                      //write name
               {
                  while(PIR1bits.TXIF==0);  //wait for transmit
                  TXREG=lname[z];           //place value in buffer
               }
         }
       while(1);
   }
```

Write a PIC18 C program to send the two messages "Normal Speed" and "High Speed" to the serial port. Assuming that SW is connected to pin PORTB.0, monitor its status and set the baud rate as follows:
SW = 0 9600 baud rate
SW = 1 38400 baud rate
Assume that XTAL = 10 MHz for both cases.

Solution:

```
#include <P18F458.h>
#define MYSW PORTBbits.RB5          //INPUT SWITCH
void main(void)
   {
      unsigned char z;
      unsigned char Mess1[]="Normal Speed";
      unsigned char Mess2[]="High Speed";
      TRISBbits.TRISB5 = 1;   //an input
      TXSTA=0x20;             //choose low baud rate, 8-bit
      SPBRG=15;               //9600 baud rate/ XTAL = 10 MHz
      TXSTAbits.TXEN=1;
      RCSTAbits.SPEN=1;
      if(MYSW==0)
         {
            for(z=0;z<12;z++)
               {
                  while(PIR1bits.TXIF==0); //wait for transmit
                  TXREG=Mess1[z];          //place value in buffer
               }
         }
      else
         {
            TXSTA=TXSTA|0x4;              //for high speed
            for(z=0;z<10;z++)
               {
                  while(PIR1bits.TXIF==0); //wait for transmit
                  TXREG=Mess2[z];          //place value in buffer
               }
         }
        while(1);
   }
```