Starting with Serial

Chapter 10
Sections 1, 2, 9, 10

Dr. Iyad Jafar
Outline

- Introduction
- Synchronous Serial Communication
- Asynchronous Serial Communication
- Physical Limitations
- Overview of PIC 16 Series
- The 16F87xA USART
- Summary
Introduction

- Microcontrollers need to move data to and from external devices

- In general, two approaches
  - **Parallel**
    - Data word bits are transferred at the same time
    - A wire is dedicated for each bit
    - Simple and fast but expensive
    - Short distances
  - **Serial**
    - Bits are transferred one after another over the same link/wire
    - Requires complex hardware to transmit and receive
    - Slow
    - Short and long distances
Introduction

- Two memories of the same size. However, one uses parallel transfer while the other uses serial.
Serial Communication

• Bits are transferred one after another on the same wire !!!

• Challenges
  • How to distinguish the start and end of the bit ?
  • How to determine the start and end of a word ?

• Two approaches
  • Synchronous serial communication
    • A separate clock signal is sent in parallel with the data
    • Each clock cycle represents one bit duration
  • Asynchronous serial communication
    • No clock signal !
    • Timing is derived from the data itself
Serial Communication

Synchronous

Asynchronous
Serial Communication

- Data inside the memory and microprocessor is formatted in parallel. How to transmit it serially?
- Shift registers

![Shift registers diagram](image1.png)

![Parallel data in-out](image2.png)
Synchronous Serial Communication

General Serial Link

Synchronous link implemented using a microcontroller
Synchronous Serial Communication

Advantages
• Simple hardware
• Efficient
• High speed

Disadvantages
• Extra line for the clock
• The bandwidth needed for the clock is twice the data bandwidth
• Data and clock may lose synchronization over long distance
Asynchronous Serial Communication

- No clock signal!
- The transmitter and receiver should operate a clock at the same rate
- To synchronize the clocks of the transmitter and receiver, data is framed with a start and stop bits
Asynchronous Serial Communication

- Framing

Diagram showing the process of asynchronous serial communication, including start bit, idle state, first data bit, start synchronisation, last data bit, extra 'parity' bit, stop bit, and earliest possible new start bit.
Asynchronous Serial Communication

- **Synchronization**

![Diagram of asynchronous serial communication with stages including idle state, start bit, midpoint of stop bit, first data bit, and receiver clock running at multiple of expected bit rate.](image)
Physical Limitations

- Time Constant effect
Physical Limitations

- Transmission Line Effects
  - Characteristic impedance and reflections
  - Lines should be terminated properly
Physical Limitations

- **Electromagnetic Interference**
  - Generated due to high voltage rates of change.
  - How to minimize:
    - At source:
      - reduce voltage rate of change.
    - In communication link:
      - large separation from source of interference.
      - Increase data voltage.
      - Screening
      - Use optical links
    - At receiver:
      - Use filtering techniques
Physical Limitations

- **Ground Differentials**
  - With longer wires, ground potential at one point might not be the same at another point.
  - **Solutions:**
    - Differential transmission.
    - Electrical isolation
    - Use optical communication links
Overview of the PIC 16 Series

- We have already seen the PIC 16F84A
- Other members in the series have more features:
  - Additional I/O ports
  - More HW timers
  - A/D converters
  - LCD Drivers
  - USARTs
  - Synchronous Serial
  - Comparators
  - ....
Overview of the PIC 16 Series
Overview of the PIC 16 Series
Interrupt Logic for 16F874A/16F877A
# Overview of the PIC 16 Series

<table>
<thead>
<tr>
<th>Device</th>
<th>Pins</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>16F873A</td>
<td>28</td>
<td>3 parallel ports, 3 counter/timers, 2 capture/compare/PWM, 2 serial, 5 10-bit ADC, 2 comparators</td>
</tr>
<tr>
<td>16F876A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16F874A</td>
<td>40</td>
<td>5 parallel ports, 3 counter/timers, 2 capture/compare/PWM, 2 serial, 8 10-bit ADC, 2 comparators</td>
</tr>
<tr>
<td>16F877A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The 16F87xA USART

- The 16F87XA family has a Universal Synchronous Asynchronous Receiver Transmitter (USART)
  - Configurable
  - Half duplex synchronous master or slave
  - Full-duplex asynchronous transmitter and receiver
- The USART shares pins with PORTC
  - pin 7 being the receive line
  - pin 6 being the transmit line
- Operation involves the following registers
  - TXSTA (0x98)  TXREG (0x19)  RCSTA (0x18)
  - RCREG (0x1A)  SPBRG (0x99)  PIE1 (0x8C)
  - PIR1 (0x0C)  INTCON (0x0B, 0x8B, 0x10B, 0x18B)
  - TRISC (0x87)
The 16F87xA USART

- Asynchronous USART Transmitter Block Diagram
The 16F87xA USART

• Asynchronous USART Transmitter  Operation Notes
  • Data is transmitted LSB first on RC6 pin
  • The shift register TSR is buffered by the TXREG (19H) and is not accessible as a memory location
  • Transmission is controlled by the TXEN bit which enables the clock to start the transmission
  • To enable serial transmission on RC6, bit SPEN in RCSTA register has to be set
  • To transmit data, it must be loaded in the TXREG. It is transferred to TSR immediately if no transmission or after the stop bit from previous transmission is sent out
  • Transmission status is provided by two bits:
    • TXIF flag in PRR1 register indicates the status of TXREG. It is set when data is transferred to TSR. It is cleared on writing to TXREG. (TXIF is cleared by hardware and it is read-only).
    • TRMT flag in TXSTA it is set when the shift register is empty
  • Parity bit can be sent out by using TXD9 bit and TX9 in TXSTA
The 16F87xA USART

**TXSTA (98H)**

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R-1</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>—</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
</tr>
</tbody>
</table>

- **bit 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)

- **bit 6 TX9**: 9-bit Transmit Enable bit
  - 1 = Selects 9-bit transmission
  - 0 = Selects 8-bit transmission

- **bit 5 TXEN**: Transmit Enable bit
  - 1 = Transmit enabled
  - 0 = Transmit disabled

  **Note**: SREN/CREN overrides TXEN in Sync mode.

- **bit 4 SYNC**: USART Mode Select bit
  - 1 = Synchronous mode
  - 0 = Asynchronous mode

- **bit 3 Unimplemented**: Read as ‘0’

- **bit 2 BRGH**: High Baud Rate Select bit
  - **Asynchronous mode**: 1 = High speed
  - 0 = Low speed
  - **Synchronous mode**: Unused in this mode.

- **bit 1 TRMT**: Transmit Shift Register Status bit
  - 1 = TSR empty
  - 0 = TSR full

- **bit 0 TX9D**: 9th bit of Transmit Data, can be Parity bit
The 16F87xA USART

**Steps for Using the asynchronous transmitter**

1. Clear TRISC<6> bit to configure RC6 as output
2. Set the SPBRG (0x99) register and BRGH (TXSTA<2>) bit to choose the appropriate baud rate
3. Enable asynchronous serial port by clearing the SYNC (TXSTA<4>) bit and setting the SPEN bit (RCTSA<7>)
4. If interrupts are desired, set the TXIE (PIE1<4>), GIE (INTCON<7>), and PEIE (INTCON<6>) bits
5. If 9-bit transmission is desired, set the TX9 (TXSTA<6>) bit
6. Enable transmission by setting the TXEN (TXSTA<5>), which will set the TXIF (PIR1<4>) bit
7. If 9-bit transmission is selected, then the ninth bit should be loaded in TX9D (TXSTA<0>)
8. Load data in TXREG (0x19) to start the transmission
The 16F87xA USART

- Timing of asynchronous transmission

- Registers involved in asynchronous transmission
The 16F87xA USART

- Asynchronous Receiver
The 16F87xA USART

• Asynchronous USART Receiver Operation Notes
  • Data is received LSB first on RC7 pin
  • Reception is enabled by the CREN bit
  • At the heart of the block is the RSR register. Once a stop bit is detected, data is transferred to RCREG register, if it is empty, and the RCIF flag is set. (RCIF is cleared by hardware and it is read-only). On-receive interrupt can be enabled by RCIE bit
  • The RCREG is FIFO double buffered register
    • can be used to receive bytes while reception continues in RSR
    • It can be read twice to read the received two bytes
    • If a stop bit is detected in RSR and the RCREG is still full, an overrun error occurs and is indicated in OERR bit (The word is RSR is lost)
    • If OERR bit is set, shifting stops in RSR and transfers to the RCREG is inhibited!
    • To clear the framing error, clear the CREN bit.

• If the stop bit is received as clear in RSR a framing error occurs and is indicated by the FERR bit.

• The 9th bit of data RCD9 and FERR are also double buffered. It is essential to read the RCSTA register before the RCREG to avoid losing the corresponding values of RCD9 and FERR
The 16F87xA USART

**RCSTA (18H)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7   | **SPEN**     | Serial Port Enable bit  
|     |              | 1 = Serial port enabled (Configures RX/DT and TX/CK pins as serial port 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** | Enables single receive  
|     |              | 0 = Disables single receive  
|     |              | This bit is cleared after reception is complete.                                                                                           |
|     | **Synchronous mode - slave** | Unused in this mode                                                                                                                         |
| 4   | **CREN**     | Continuous Receive Enable bit  
|     | **Asynchronous mode** | Enables continuous receive  
|     |              | 0 = Disables continuous receive                                                                                                             |
|     | **Synchronous mode** | Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)  
|     |              | 0 = Disables continuous receive                                                                                                             |
| 3   | Unimplemented | Read as '0'                                                                                                                              |
| 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.                                                                                               |
The 16F87xA USART

Steps for Using the asynchronous receiver

1. Set the SPBRG (0x99) register and BRGH (TXSTA<2>) bit to choose the appropriate baud rate
2. Enable asynchronous serial port by clearing the SYNC (TXSTA<4>) bit and setting the SPEN bit (RCTSA<7>)
3. If interrupts are desired, set the RCIE (PIE1<5>), GIE (INTCON<7>), and PEIE (INTCON<6>) bits
4. If 9-bit reception is desired, set the RX9 (RCSTA<6>) bit
5. Enable the reception by setting bit CREN (RCSTA<4>)
6. The RCIF (PIR1<5>) will be set when reception of one word is complete and an interrupt will be generated if RCIE is set
7. Read the RCSTA (0x18) to get the 9th bit and determine if any error occurred (OERR, FERR)
8. Read the 8-bit received data by reading RCREG (0x1A)
9. If any error occurred, clear the error by clearing the CREN
The 16F87xA USART

- Timing of asynchronous reception

![Timing Diagram]

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.

- Registers involved in asynchronous reception

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on: POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>0Bh, 8Bh, 10Bh, 18Bh</td>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>R0IF</td>
<td>0000 000x</td>
<td>0000 000u</td>
</tr>
<tr>
<td>0Ch</td>
<td>PIR1</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>18h</td>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>—</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>0000 -00x</td>
<td>0000 -00x</td>
</tr>
<tr>
<td>1Ah</td>
<td>RCREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>8Ch</td>
<td>PIE1</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>98h</td>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>—</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>0000 -010</td>
<td>0000 -010</td>
</tr>
<tr>
<td>99h</td>
<td>SPBRG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
</tbody>
</table>
The 16F87xA USART

- **The BAUD Rate Generator**
  - The BAUD rate for USART is controlled by the value in the `SPREG (99H)`, the `SYNC` and the `BRGH` bits in the `TXSTA (19H)`.

<table>
<thead>
<tr>
<th>SYNC</th>
<th>BRGH = 0</th>
<th>BRGH = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (asynchronous)</td>
<td>$\frac{F_{osc}}{64(SPBRG + 1)}$</td>
<td>$\frac{F_{osc}}{16(SPBRG + 1)}$</td>
</tr>
<tr>
<td>1 (synchronous)</td>
<td>$\frac{F_{osc}}{4(SPBRG + 1)}$</td>
<td></td>
</tr>
</tbody>
</table>
Example

A program to transmit 3 bytes stored in locations 0x40, 0x41, and 0x42 serially with no parity at a rate of 9.6 Kbps. Assume PIC 16F877A with oscillator frequency of 20 MHz

Requirements

1. setup the serial port for transmission
2. choose the appropriate value of SPBRG and BRGH to produce the required rate
```assembly
#include p16F877A.inc ; include the definition file for 16F77A
org 0x0000 ; reset vector
goto START
org 0x0004 ; define the ISR
goto ISR
org 0x0006 ; Program starts here

ISR
bsf STATUS, RP0

START
bcf STATUS, RP1 ; select bank 1
bcf TRISC, 6 ; set RC6 as output
movlw D'31'  ; set the SPBRG value
movwf SPBRG
bsf TXSTA, TXEN
bsf STATUS, RP0 ; select bank0
bsf RCSTA, SPEN ; enable serial transmission
movlw 0x40
movwf FSR ; FSR has the address of the first element
```
**Example**

```
TX
    movf INDF, W ; read byte to transmit
    movwf TXREG ; store in the transmission register
    incf FSR, F ; increment FSR to point to next address

WAIT
    btfss PIR1, TXIF ; check if the TXREG is empty
    goto WAIT
    movf FSR, W
    sublw 0x43
    btfss STATUS, Z ; check if all values were transmitted
    goto TX

DONE
    goto DONE
end
```
Summary

- Serial communication transmits bits one after another in two modes: synchronous and asynchronous.
- Stable and accurate clocking plays an important role in serial communication.
- It is cheaper to use serial communication over long distances.
- Some members of the 16 series are equipped with synchronous and asynchronous communication ports.
- These ports can be configured to operate in different modes and rates.