USART Serial Port in AVR Microcontrollers

(Chapter 11 of the Mazidi’s book)
Contents

• Serial communication
• Serial communication programming in C
Serial ports in AVR
Serial ports in AVR

- **USART:** an old and widely used standard (pins 14 and 15 of ATmega32)
- **SPI:** very high-speed (~ two orders of magnitude faster than USART) synchronous serial port
  - For example for high-speed data transfer to EPROM
  - Pins 5, 6, and 7 of Atmega32
- **I2C (Inter IC)**
  - Connection using a shared bus connecting up to 128 devices
  - Pins 22 and 23 of ATmega32
Data transmission

• Data transmission between AVR and peripheral devices like sensors is straightforward
  – Connect the device to one of the AVR ports and read the port

• Data transmission between two AVRs or an AVR and a PC is more complex
  – Requires a communication protocol

• Communication Protocol: a set of standards and rules that orchestrates data communication between two or more devices
RS232

- An old standard for serial communication
- Still used in many devices like PCs
- Is used in serial port (COM port) of PCs
  - PCs used to have 2 COM ports, but today PCs have one port or no port
  - Replaced with USB ports
RS232

- Requires 3 pins:
- RXD = receive, TXD: transmit, GND = ground
- The RXD and TXD pins are cross-connected
• COM = communication port
• Find it at the back of your PC case!
• Pins 2, 3, and 5 are TXD, RXD, and GND, respectively.
Serial communication in AVR

- USART: a standard IC that provides both synchronous and asynchronous communication
- A peripheral IC of the AVR microcontrollers
- Connected to RXD and TXD pins of Atmega32
USART

• USART: Universal Synchronous/Asynchronous Receiver Transmitter

• An standard IC that can provide both synchronous and asynchronous communication

• It is controlled by some AVR registers
USART block diagram in AVR
• **The control registers specify:**
  – The mode of operation: synchronous or asynchronous
  – Parity bit: odd or even
  – Information unit: 5, 6, 7, 8, or 9 bits
  – Information unit separation: how to specify the transmission of a word starts and stops?
  – Transmission rate
Parity bit

- A way to detect error during data transmission
  - Due to external noises
- Even parity= the number of 1s must be an even number
- Add an extra bit to the 8-bit data, called parity bit
- How does it work?
  - if the number of 1s is already even, set it to 0, otherwise to 1
  - Send the parity bit with data
  - If the other side detects odd number of 1s, there is something wrong
Transmission rate

- Baud rate vs bit rate
- Baud rate: the number signal changes in a second
- Bit rate: the number of bits transmitted per second
- Baud rate is not necessarily equal to bit rate
  - Each signal may carry several bits!
- A signal with 8 levels can carry 3 bits
  - If baud rate = x, bit rate = 3x
- In USART, baud rate = bit rate
- The bit rate of both devices connected to the same serial port must be the same
AVR serial port programming

• Setting different registers
• 5 registers are associated with serial port:
  – UDR: USART data register
  – UCSRA, UCSRB, UCSRC: USART control and status register
  – UBRR: USART baud rate register
UDR

- USART data register
- Actually two registers: one for the transmit direction, the other for receive direction
- Share the same address and name
  - When UDR is read, the data received from the serial line is returned
  - When a data is written to UDR, it is directed to the transmit line
UDR

DATA BUS

RXB (receive data buffer register)

Stop

(1)

Receive shift register

(0)

Start

TXB (transmit data buffer register)

Stop

(1)

Transmit shift register

(0)

Start

RxD Pin

TxD Pin

University of Tehran 17
UCSR

- USART control and status register
- Three 8-bit registers to control the USART operation
RXC (Bit 7): USART Receive Complete
This flag bit is set when there are new data in the receive buffer that are not read yet. It is
cleared when the receive buffer is empty. It also can be used to generate a receive complete interrupt.

TXC (Bit 6): USART Transmit Complete
This flag bit is set when the entire frame in the transmit shift register has been trans-
mittted and there are no new data available in the transmit data buffer register (TXB).
It can be cleared by writing a one to its bit location. Also it is automatically cleared
when a transmit complete interrupt is executed. It can be used to generate a transmit complete interrupt.

UDRE (Bit 5): USART Data Register Empty
This flag is set when the transmit data buffer is empty and it is ready to receive new
data. If this bit is cleared you should not write to UDR because it overrides your last
data. The UDRE flag can generate a data register empty interrupt.
**UCSRA- continue**

<table>
<thead>
<tr>
<th>RXC</th>
<th>TXC</th>
<th>UDRE</th>
<th>FE</th>
<th>DOR</th>
<th>PE</th>
<th>U2X</th>
<th>MPCM</th>
</tr>
</thead>
</table>

**FE (Bit 4): Frame Error**
This bit is set if a frame error has occurred in receiving the next character in the receive buffer. A frame error is detected when the first stop bit of the next character in the receive buffer is zero.

**DOR (Bit 3): Data OverRun**
This bit is set if a data overrun is detected. A data overrun occurs when the receive data buffer and receive shift register are full, and a new start bit is detected.

**PE (Bit 2): Parity Error**
This bit is set if parity checking was enabled (UPM1 = 1) and the next character in the receive buffer had a parity error when received.
U2X (Bit 1): Double the USART Transmission Speed
Setting this bit will double the transfer rate for asynchronous communication.

MPCM (Bit 0): Multi-processor Communication Mode
This bit enables the multi-processor communication mode. The MPCM feature is not discussed in this book.
<table>
<thead>
<tr>
<th>RXCIE</th>
<th>TXCIE</th>
<th>UDRIE</th>
<th>RXEN</th>
<th>TXEN</th>
<th>UCSZ2</th>
<th>RXB8</th>
<th>TXB8</th>
</tr>
</thead>
</table>

**RXCIE (Bit 7): Receive Complete Interrupt Enable**
To enable the interrupt on the RXC flag in UCSRA you should set this bit to one.

**TXCIE (Bit 6): Transmit Complete Interrupt Enable**
To enable the interrupt on the TXC flag in UCSRA you should set this bit to one.

**UDRIE (Bit 5): USART Data Register Empty Interrupt Enable**
To enable the interrupt on the UDRE flag in UCSRA you should set this bit to one.

**RXEN (Bit 4): Receive Enable**
To enable the USART receiver you should set this bit to one.
TXEN (Bit 3): Transmit Enable
To enable the UART transmitter you should set this bit to one.

UCSZ2 (Bit 2): Character Size
This bit combined with the UCSZ1:0 bits in UCSRC sets the number of data bits (character size) in a frame.

RXB8 (Bit 1): Receive data bit 8
This is the ninth data bit of the received character when using serial frames with nine data bits. This bit is not used in this book.

TXB8 (Bit 0): Transmit data bit 8
This is the ninth data bit of the transmitted character when using serial frames with nine data bits. This bit is not used in this book.
Note on bit7: UCSRC and UBBR share the same address due to some technical issue. Set URSEL=1 when you want the data to be written to UCSRC, otherwise set URSEL=0 to write to UBBR.
USBS (Bit 3): Stop Bit Select
This bit selects the number of stop bits to be transmitted.
0 = 1 bit
1 = 2 bits

UCSZ1:0 (Bit 2:1): Character Size
These bits combined with the UCSZ2 bit in UCSRB set the character size in a frame and will be discussed more in this section.

UCPOL (Bit 2): Clock Polarity
This bit is used for synchronous mode only and will not be covered in this section.
Character size of the transmitted and received data- is same for both directions

<table>
<thead>
<tr>
<th>UCSZ2</th>
<th>UCSZ1</th>
<th>UCSZ0</th>
<th>Character Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

Note: Other values are reserved. Also notice that UCSZ0 and UCSZ1 belong to UCSRC and UCSZ2 belongs to UCSRB
UBRR

- USART Baud Rate Register
- 12 bits
- The most significant byte has a shared address with UCSRC!

\[
X = \left( \frac{\text{Fosc}}{(16(\text{Desired Baud Rate}))} \right) - 1
\]

**Desired Baud Rate =** \[\text{Fosc/ (16(X + 1))}\]

\[X= \text{UBBR[0-11]}\]
Example 11-1

With Fosc = 8 MHz, find the UBRR value needed to have the following baud rates:
(a) 9600  (b) 4800  (c) 2400  (d) 1200

Solution:

Fosc = 8 MHz => X = (8 MHz/16(Desired Baud Rate)) − 1
=> X = (500 kHz/(Desired Baud Rate)) − 1

(a) (500 kHz/ 9600) − 1 = 52.08 − 1 = 51.08 = 51 = 33 (hex) is loaded into UBRR
(b) (500 kHz/ 4800) − 1 = 104.16 − 1 = 103.16 = 103 = 67 (hex) is loaded into UBRR
(c) (500 kHz/ 2400) − 1 = 208.33 − 1 = 207.33 = 207 = CF (hex) is loaded into UBRR
(d) (500 kHz/ 1200) − 1 = 416.66 − 1 = 415.66 = 415 = 19F (hex) is loaded into UBRR
Sampling

- Sampling rate: 16 times faster than baud rate
- 2 out of 3 majority voting in cycles 8 to 9
USART in Tx mode

1. The UCSRB register is loaded with the value 08H, enabling the USART transmitter. The transmitter will override normal port operation for the TxD pin when enabled.

2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity, and one stop bit.

3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.

4. The character byte to be transmitted serially is written into the UDR register.

5. Monitor the UDRE bit of the UCSRA register to make sure UDR is ready for the next byte.

6. To transmit the next character, go to Step 4.
USART in Rx mode

1. The UCSRB register is loaded with the value 10H, enabling the USART receiver. The receiver will override normal port operation for the RxD pin when enabled.
2. The UCSRC register is loaded with the value 06H, indicating asynchronous mode with 8-bit data frame, no parity, and one stop bit.
3. The UBRR is loaded with one of the values in Table 11-4 (if Fosc = 8 MHz) to set the baud rate for serial data transfer.
4. The RXC flag bit of the UCSRA register is monitored for a HIGH to see if an entire character has been received yet.
5. When RXC is raised, the UDR register has the byte. Its contents are moved into a safe place.
6. To receive the next character, go to Step 5.

<table>
<thead>
<tr>
<th>RXC</th>
<th>TXC</th>
<th>UDRE</th>
<th>FE</th>
<th>DOR</th>
<th>PE</th>
<th>U2X</th>
<th>MPCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXCIE</td>
<td>TXCIE</td>
<td>UDRIE</td>
<td>RXEN</td>
<td>TXEN</td>
<td>UCSZ2</td>
<td>RXB8</td>
<td>TXB8</td>
</tr>
<tr>
<td>URSEL</td>
<td>UMSEL</td>
<td>UPM1</td>
<td>UPM0</td>
<td>USBS</td>
<td>UCSZ1</td>
<td>UCSZ0</td>
<td>UCPOL</td>
</tr>
</tbody>
</table>

University of Tehran 31
Synchronous mode

- Work in master-slave mode
- The master send clock signal to slave
Synchronous mode

- **UCPOL**: bit 7 in UCSRC
- **Baud rate**: \( f_{osc}/(2 \times (UBBR+1)) \)
USART programming in C

- Send a sequence of numbers started from 0 every 350ms to TXD pin
- Check RXD pin and if the received number is
  - 0x55 set PD.6 (bit 6 of port D) to 1
  - 0x66 set PD.6 to 0
- 1 stop bit, no parity, 8 bit, asynchronous

<table>
<thead>
<tr>
<th>UCSZ2</th>
<th>UCSZ1</th>
<th>UCSZ0</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RXC</th>
<th>TXC</th>
<th>UDRE</th>
<th>FE</th>
<th>DOR</th>
<th>PE</th>
<th>U2X</th>
<th>MPCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXCIE</td>
<td>TXCIE</td>
<td>UDRIE</td>
<td>RXEN</td>
<td>TXEN</td>
<td>UCSZ2</td>
<td>RXB8</td>
<td>TXB8</td>
</tr>
<tr>
<td>URSEL</td>
<td>UMSEL</td>
<td>UPM1</td>
<td>UPM0</td>
<td>USBS</td>
<td>UCSZ1</td>
<td>UCSZ0</td>
<td>UCPOI</td>
</tr>
</tbody>
</table>
Code vision configuration
USART programming in C

main()
{
    int a=0;
    DDRD.6=1;
    UCSRA=0x0;
    UCSRB=0x98; //10011000 (RXIE=1, RXEN=1, TXEN=1)
    UCSRC=0x86; // 10000110 (URSEL=1,asynch, no parity, one stop bit, 8 bit)
    UBRRH=0; //just set a rate that guarantees the data transfer can be completed before 350ms
    UBRL=0x08;
    #asm("sei");
    while(1)
    {
        UDR= a++;
        delay_ms(350);
    }
}

Interrupt [USART_RXC]  usart_rx_isr()
{
    char data;
    data=UDR;
    if(data==0x55)
        PORTD.6=1;
    if(data==0x66)
        PORTD.6=0;
}
USART in PCs

- Implemented by the 8251 IC