UART Protocol

Chapter 11

Sepehr Naimi

www.NicerLand.com

1

Topics

- Communication theory
  - Parallel vs. Serial
  - Direction: Simplex, Half duplex, Full duplex
  - Synchronization: Synchronous vs. Asynchronous
  - Line coding
- UART protocol
- UART in AVR
  - UART Registers
  - Some programs

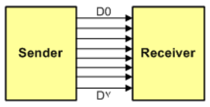
2

2

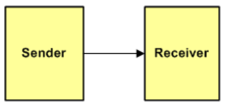
Parallel vs. Serial

- Parallel
  - For short distances
  - Not applicable for long distances
  - More expensive
  - Cross-talk problem

Parallel Transfer

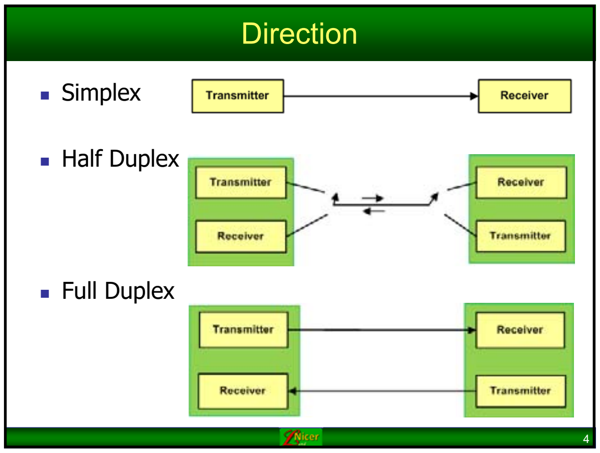


Serial Transfer



3

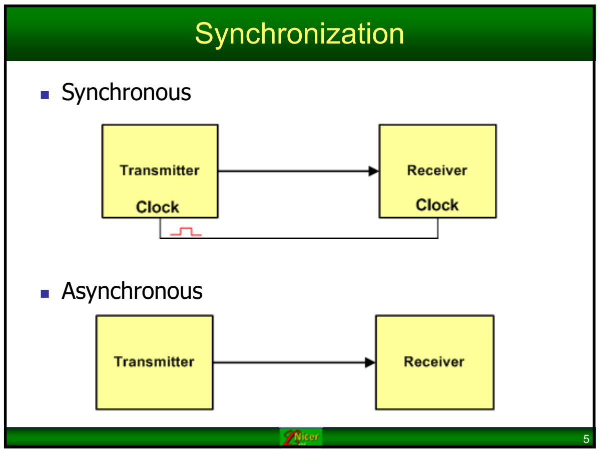
3



Direction

- Simplex
- Half Duplex
- Full Duplex

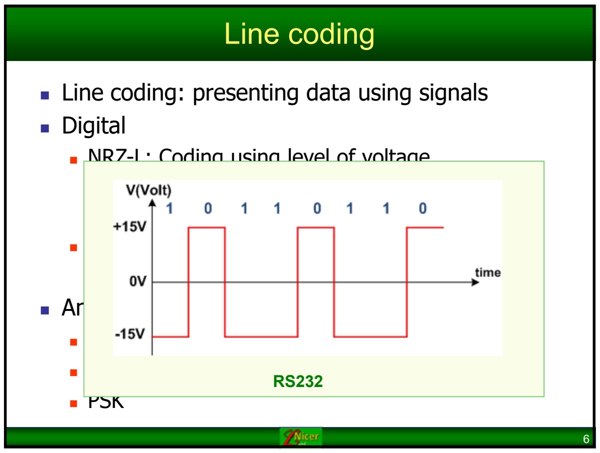
4



Synchronization

- Synchronous
- Asynchronous

5



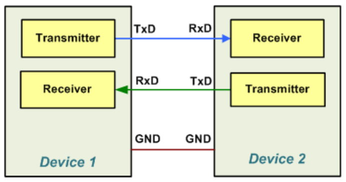
Line coding

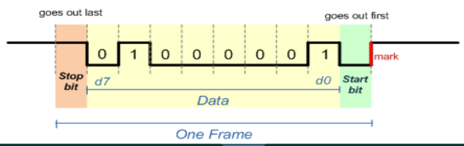
- Line coding: presenting data using signals
- Digital
  - NRZ-L: Coding using level of voltage
  -
- Ar
  -
  -
  - PSK

6

UART Protocol

- Serial
- Asynchronous
- Full duplex





Nicer 7

7

USART (Universal Synchronous Asynchronous Receive Transmit)

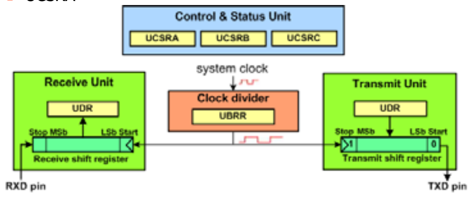
- USART devices can be used to communicate asynchronously (UART) and synchronously.
- Since the synchronous capability of USART is not used nowadays, we concentrate on UART.

Nicer 8

8

UART in AVR

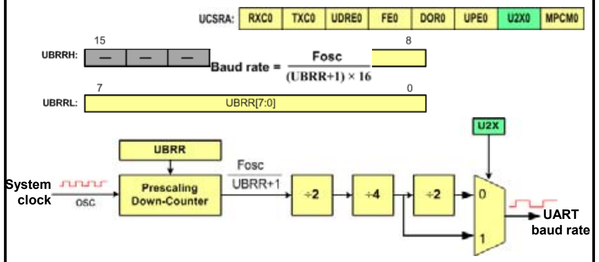
- Control registers: (initialize speed, data size, parity, etc)
  - UBRR, UCSRA, UCSRB, and UCSRC
- Send/receive register
  - UDR
- Status register
  - UCSRA



Nicer 9

9

Baud rate



10

Example: Find the UBRR value for 9600bps.

- Solution:

$$\text{Baud rate} = \frac{\text{Fosc}}{(\text{UBRR}+1) \times 16} \Rightarrow 9600 = \frac{16\text{ MHz}}{(\text{UBRR}+1) \times 16}$$

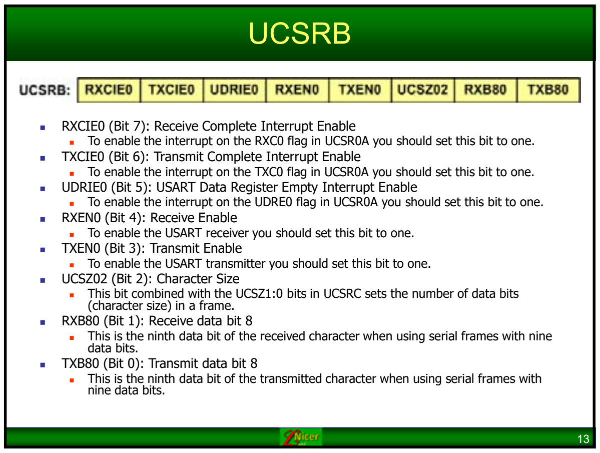
$$\Rightarrow (\text{UBRR}+1) = \frac{1\text{MHz}}{9600} = 104.166 \Rightarrow \text{UBRR} = 103$$

11

Some Standard Baud rates

1200
2400
4800
9600
19,200
38,400
57,600
115,200

12



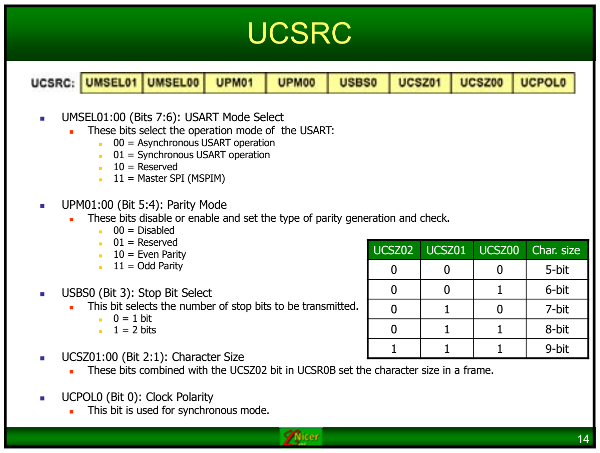
UCSRB

UCSRB:	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80

- RXCIE0 (Bit 7): Receive Complete Interrupt Enable
  - To enable the interrupt on the RXC0 flag in UCSR0A you should set this bit to one.
- TXCIE0 (Bit 6): Transmit Complete Interrupt Enable
  - To enable the interrupt on the TXC0 flag in UCSR0A you should set this bit to one.
- UDRIE0 (Bit 5): USART Data Register Empty Interrupt Enable
  - To enable the interrupt on the UDRE0 flag in UCSR0A you should set this bit to one.
- RXEN0 (Bit 4): Receive Enable
  - To enable the USART receiver you should set this bit to one.
- TXEN0 (Bit 3): Transmit Enable
  - To enable the USART transmitter you should set this bit to one.
- UCSZ02 (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.
- RXB80 (Bit 1): Receive data bit 8
  - This is the ninth data bit of the received character when using serial frames with nine data bits.
- TXB80 (Bit 0): Transmit data bit 8
  - This is the ninth data bit of the transmitted character when using serial frames with nine data bits.

Nicer 13

13



UCSRC

UCSRC:	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0

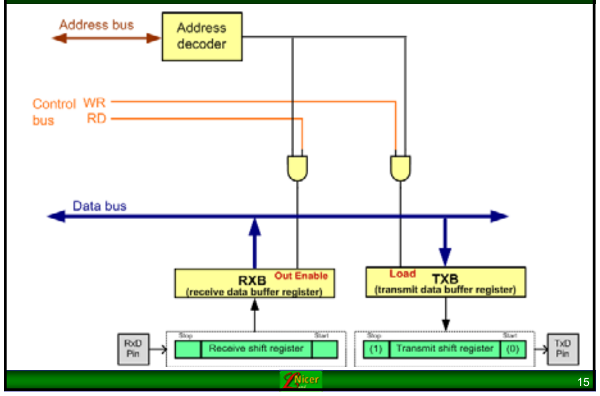
- UMSEL01:00 (Bits 7:6): USART Mode Select
  - These bits select the operation mode of the USART:
    - 00 = Asynchronous USART operation
    - 01 = Synchronous USART operation
    - 10 = Reserved
    - 11 = Master SPI (MSPIM)
- UPM01:00 (Bit 5:4): Parity Mode
  - These bits disable or enable and set the type of parity generation and check.
    - 00 = Disabled
    - 01 = Reserved
    - 10 = Even Parity
    - 11 = Odd Parity
- USBS0 (Bit 3): Stop Bit Select
  - This bit selects the number of stop bits to be transmitted.
    - 0 = 1 bit
    - 1 = 2 bits
- UCSZ01:00 (Bit 2:1): Character Size
  - These bits combined with the UCSZ02 bit in UCSR0B set the character size in a frame.
- UCPOL0 (Bit 0): Clock Polarity
  - This bit is used for synchronous mode.

UCSZ02	UCSZ01	UCSZ00	Char. size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	1	1	9-bit

Nicer 14

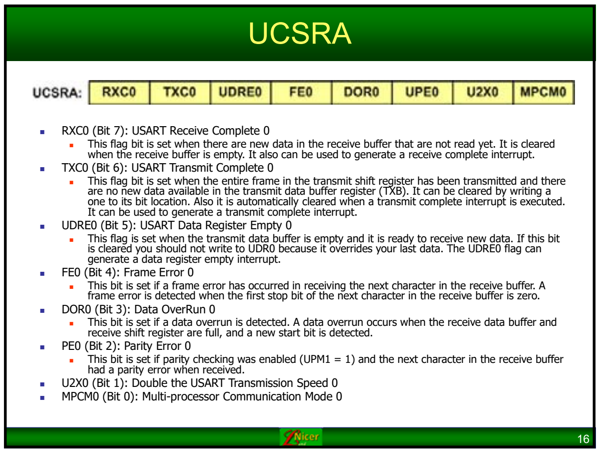
14

UDR



Nicer 15

15



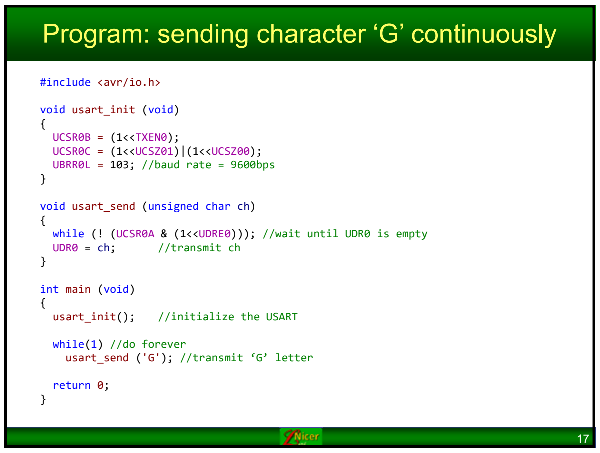
UCSRA

UCSRA:							
RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0

- RXC0 (Bit 7): USART Receive Complete 0
  - 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.
- TXC0 (Bit 6): USART Transmit Complete 0
  - This flag bit is set when the entire frame in the transmit shift register has been transmitted 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.
- UDRE0 (Bit 5): USART Data Register Empty 0
  - 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 UDR0 because it overrides your last data. The UDRE0 flag can generate a data register empty interrupt.
- FE0 (Bit 4): Frame Error 0
  - 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.
- DOR0 (Bit 3): Data OverRun 0
  - 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.
- PE0 (Bit 2): Parity Error 0
  - 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.
- U2X0 (Bit 1): Double the USART Transmission Speed 0
- MPCM0 (Bit 0): Multi-processor Communication Mode 0

16

16

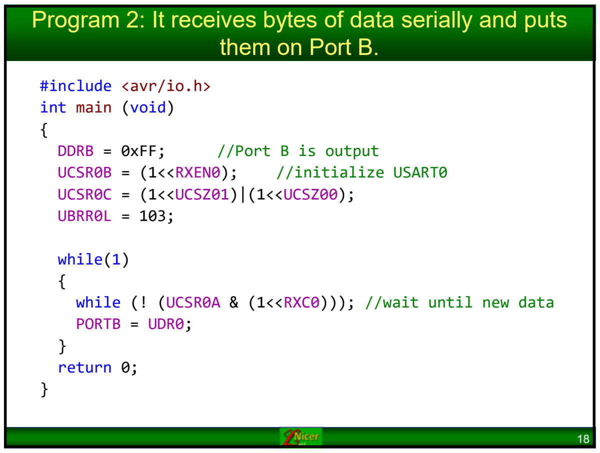


Program: sending character ‘G’ continuously

```
#include <avr/io.h>

void usart_init (void)
{
  UCSR0B = (1<<TXEN0);
  UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
  UBRR0L = 103; //baud rate = 9600bps
}

void usart_send (unsigned char ch)
{
  while (! (UCSR0A & (1<<UDRE0))); //wait until UDR0 is empty
  UDR0 = ch;       //transmit ch
}

int main (void)
{
  usart_init();    //initialize the USART

  while(1) //do forever
    usart_send ('G'); //transmit ‘G’ letter

  return 0;
}
```

17

17



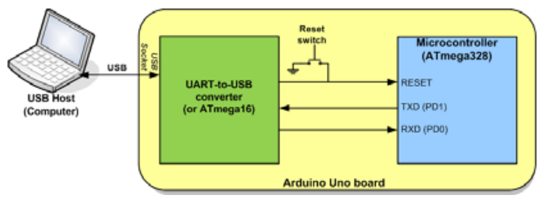
Program 2: It receives bytes of data serially and puts them on Port B.

```
#include <avr/io.h>
int main (void)
{
  DDRB = 0xFF;      //Port B is output
  UCSR0B = (1<<RXEN0);     //initialize USART0
  UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
  UBRR0L = 103;

  while(1)
  {
    while (! (UCSR0A & (1<<RXC0))); //wait until new data
    PORTB = UDR0;
  }
  return 0;
}
```

18

18

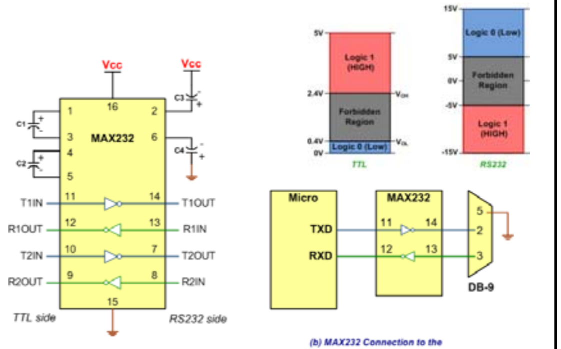
USB to serial converter in Arduino



19

19

MAX232



(a) Inside MAX232

(b) MAX232 Connection to the Microcontroller / Microprocessor

20

20