

UART Communication

Prior to Day 17



ME430: MECHATRONICS

Communication Standards/Protocols

- USB
- Bluetooth
- HTTP
- FTP
- RS232
- CAN
- LIN
- RS485
- SPI
- TTL Serial (our UART)
- Parallel Ports
- TCP/IP
- I²C
- Zigbee
- SimplicTI
- WiFi
- 802.15.4
- RF



ME430: MECHATRONICS

Microcontroller communication

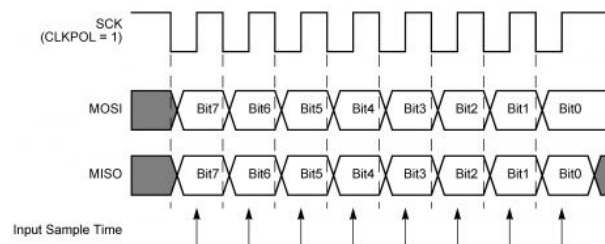
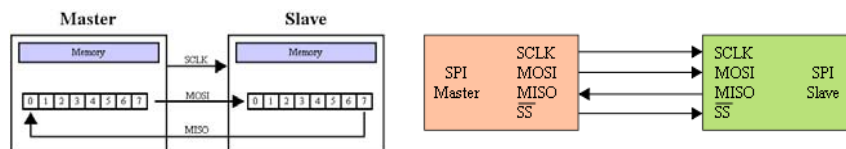
- SPI
 - Serial Peripheral Interface
- I²C
 - Inter-Integrated Circuit (by Philips)
- UART
 - Universal asynchronous receiver/transmitter



ME430: MECHATRONICS

SPI

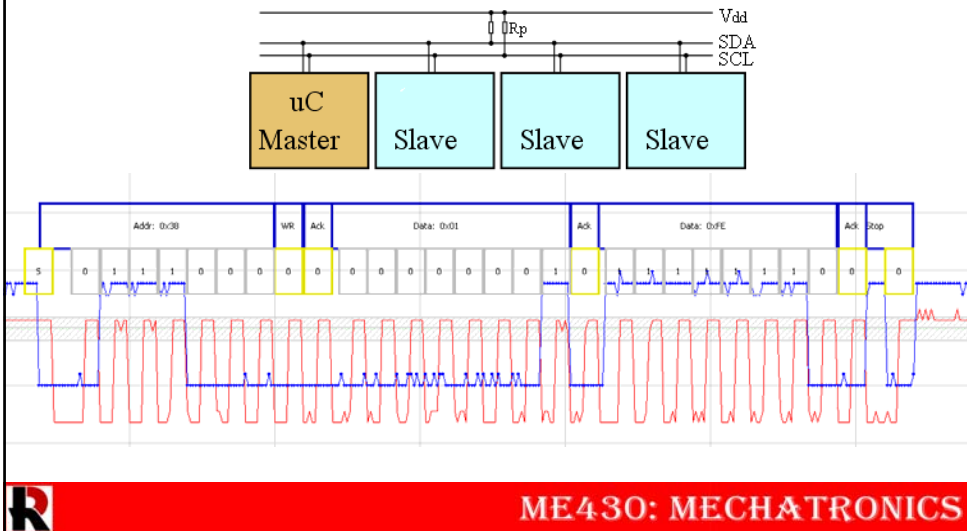
- Simple synchronous communication



ME430: MECHATRONICS

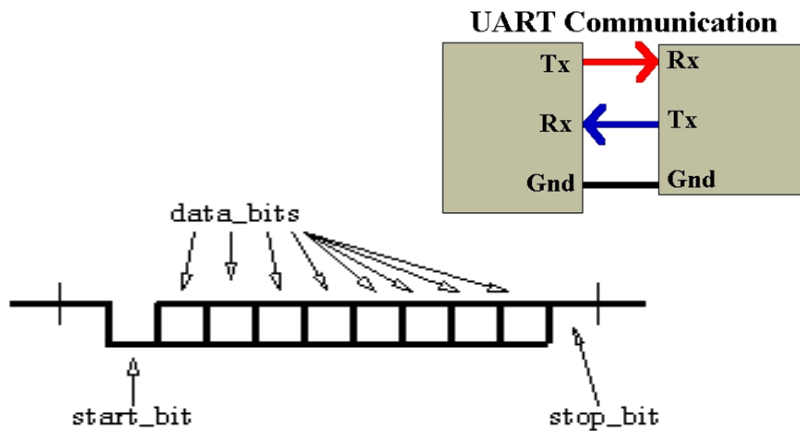
I²C

More complex synchronous communication protocol



UART

- Simple Asynchronous Communication



Hardware Peripherals

- Timers
- ADC
- PWM
- USART
 - Universal synchronous/asynchronous
receiver/transmitter



ME430: MECHATRONICS

UART vs. USART

- Synchronous Mode
 - SPI Master
- Asynchronous Mode
 - UART

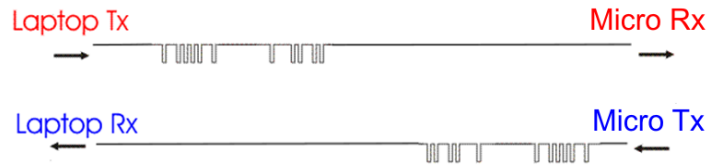
- UART is the most common microcontroller to PC communication



ME430: MECHATRONICS

PC to micro Communication

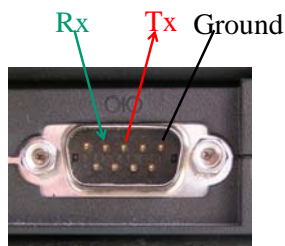
- Independent Rx & Tx



ME430: MECHATRONICS

RS232 Serial Communication

- Usually (or maybe we should say previously) UART is/was connected via an RS232 port, also known as a DB9 Serial Port, or just called, more simply, a "Serial Port"



Laptop Serial Port



Serial Cable



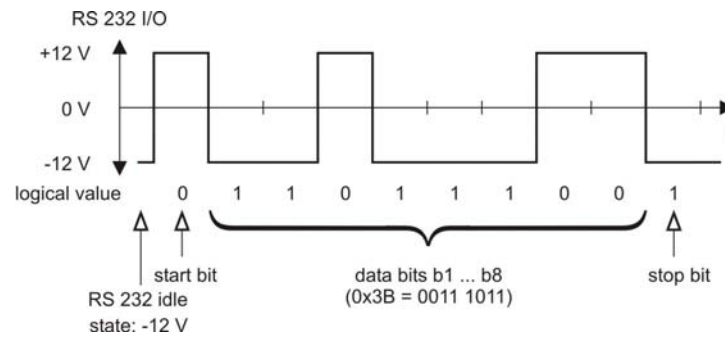
18F4520



ME430: MECHATRONICS

RS232

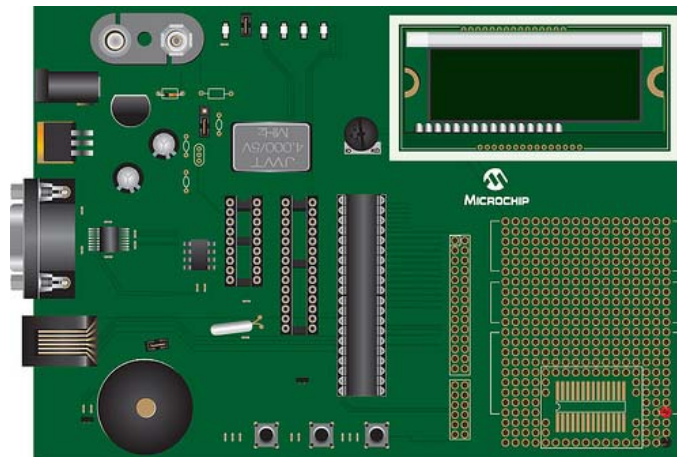
- Same concept, different voltage levels



ME430: MECHATRONICS

Getting to RS232 voltages

- Max232 charge pump



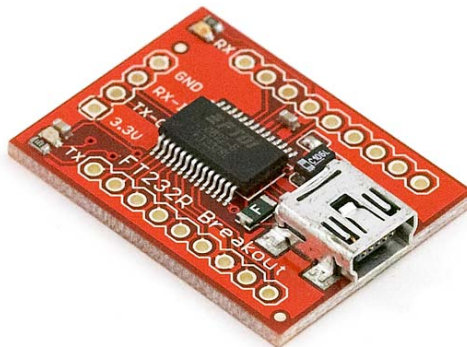
ME430: MECHATRONICS

UART over USB

- FTDI chip
(or similar)

Avoid RS232 levels
Avoid DB9 connectors
Create a virtual COM
port over USB

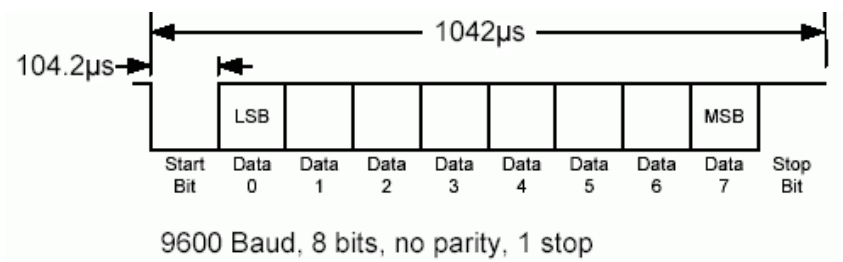
Really handy with
today's PCs!



ME430: MECHATRONICS

UART Timing

- Both clocks must match



ME430: MECHATRONICS

Common baud rates

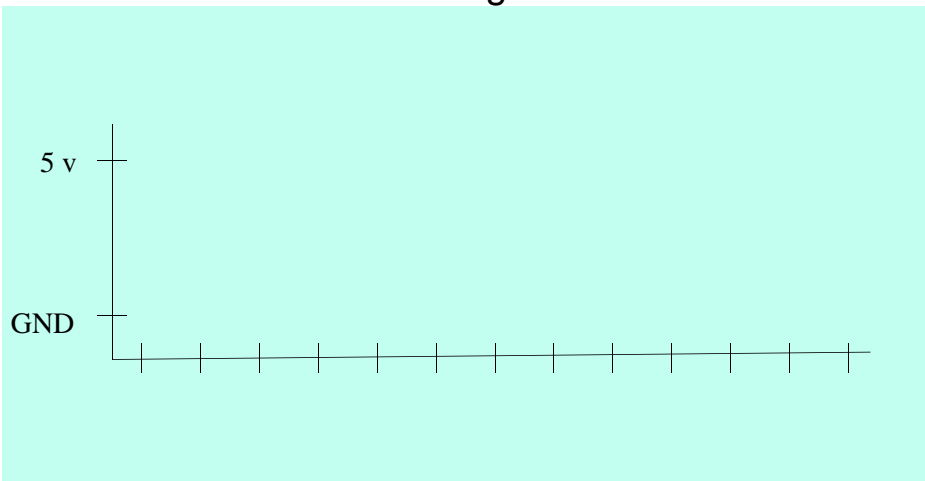
- 115200 bits per second
- 57600
 - (31250 for MIDI)
- 19200
 - (28800 & 14400 back in the day!)
- 9600
 - (and many slower speeds for older products)



ME430: MECHATRONICS

Your turn

- Draw and label sending 0x05 @ 19200



ME430: MECHATRONICS

PIC specifics

- C18 Library functions

Function	Description
BusyUSART	Is the USART transmitting?
CloseUSART	Disable the USART.
DataRdyUSART	Is data available in the USART read buffer?
getcUSART	Read a byte from the USART.
getsUSART	Read a string from the USART.
OpenUSART	Configure the USART.
putcUSART	Write a byte to the USART.
putsUSART	Write a string from data memory to the USART.
putrsUSART	Write a string from program memory to the USART.
ReadUSART	Read a byte from the USART.
WriteUSART	Write a byte to the USART.
baudUSART	Set the baud rate configuration bits for enhanced USART.



ME430: MECHATRONICS

Example OpenUSART

Include: usart.h

Prototype: void OpenUSART(unsigned char *config*,
unsigned int *spbrg*);

```
OpenUSART ( USART_TX_INT_OFF &
            USART_RX_INT_OFF &
            USART_ASYNC_MODE &
            USART_EIGHT_BIT &
            USART_CONT_RX &
            USART_BRGH_HIGH,
            25
            );
```

Interrupt on Transmission:

USART_TX_INT_ON Transmit interrupt ON
USART_TX_INT_OFF Transmit interrupt OFF

Interrupt on Receipt:

USART_RX_INT_ON Receive interrupt ON
USART_RX_INT_OFF Receive interrupt OFF

USART Mode:

USART_ASYNC_MODE Asynchronous Mode
USART_SYNC_MODE Synchronous Mode

Transmission Width:

USART_EIGHT_BIT 8-bit transmit/receive
USART_NINE_BIT 9-bit transmit/receive

Slave/Master Select*:

USART_SYNC_SLAVE Synchronous Slave mode
USART_SYNC_MASTER Synchronous Master mode

Reception mode:

USART_SINGLE_RX Single reception
USART_CONT_RX Continuous reception

Baud rate:

USART_BRGH_HIGH High baud rate
USART_BRGH_LOW Low baud rate

* Applies to Synchronous mode only



ME430: MECHATRONICS

Timing calculation

spbrg

This is the value that is written to the baud rate generator register which determines the baud rate at which the USART operates. The formulas for baud rate are:

Asynchronous mode, high speed:

$$FOSC / (16 * (spbrg + 1))$$

Asynchronous mode, low speed:

$$FOSC / (64 * (spbrg + 1))$$

```
OpenUSART( USART_TX_INT_OFF &
            USART_RX_INT_OFF &
            USART_ASYNCH_MODE &
            USART_EIGHT_BIT &
            USART_CONT_RX &
            USART_BRGH_HIGH,
            25 );
```

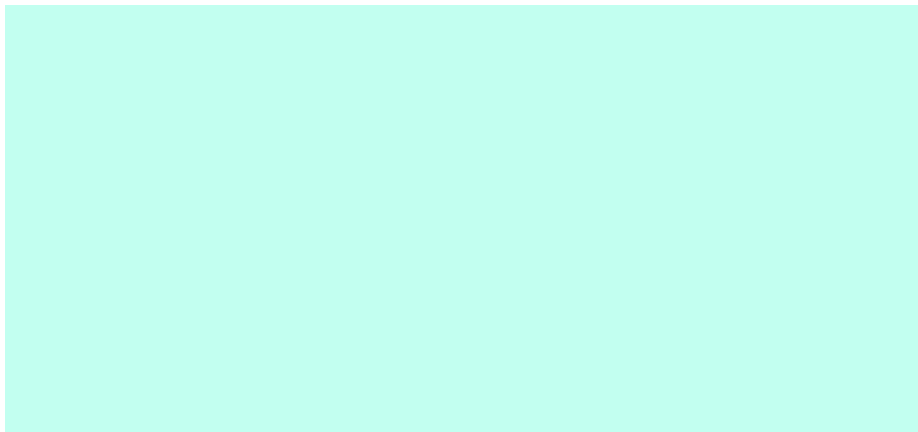
Assume 4 MHz. What's
is the bps here?



ME430: MECHATRONICS

Your turn

- Calculate *spbrg* for 19200 if you are using a 4MHz clock?



ME430: MECHATRONICS

Sending data

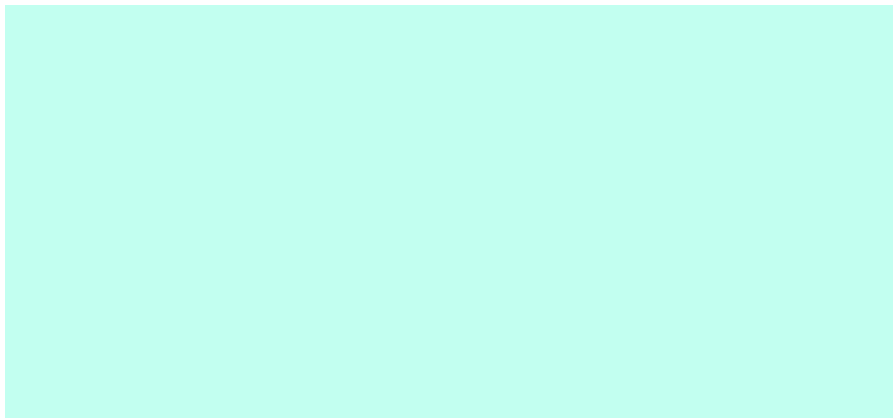
- C18 Library functions

Function	Description
BusyUSART	Is the USART transmitting?
CloseUSART	Disable the USART.
DataRdyUSART	Is data available in the USART read buffer?
getcUSART	Read a byte from the USART.
getsUSART	Read a string from the USART.
OpenUSART	Configure the USART.
putcUSART	Write a byte to the USART.
putsUSART	Write a string from data memory to the USART.
putrsUSART	Write a string from program memory to the USART.
ReadUSART	Read a byte from the USART.
WriteUSART	Write a byte to the USART.



Your turn

- Create a char x = 17 and send it via UART
 - Assume the OpenUART is done already



ASCII data sent via **printf**

- Include <stdio.h>

```
printf("Hello World!\n");
```

```
printf("ADC = %d", RA0result);
```



Receiving data

- C18 Library functions

Function	Description
BusyUSART	Is the USART transmitting?
CloseUSART	Disable the USART.
DataRdyUSART	Is data available in the USART read buffer?
getcUSART	Read a byte from the USART.
getsUSART	Read a string from the USART.
OpenUSART	Configure the USART.
putcUSART	Write a byte to the USART.
putsUSART	Write a string from data memory to the USART.
putrsUSART	Write a string from program memory to the USART.
ReadUSART	Read a byte from the USART.
WriteUSART	Write a byte to the USART.



Your turn

- Create a char x, read a byte from the UART, store the read UART byte into x



ME430: MECHATRONICS

Polling for receive

- Check for data then read a byte

```
while(1)
{
    char result;
    if (DataRdyUSART())
    {
        result = ReadUSART();

        // Do something with result
    }
}
```



ME430: MECHATRONICS

Interrupts on the receive

```

RCONbits.IPEN = 1;    // Priority mode interrupts

// configure USART for 19200 buad rate
OpenUSART( USART_TX_INT_OFF &
           USART_RX_INT_ON &
           USART_ASYNC_MODE &
           USART_EIGHT_BIT &
           USART_BRGH_HIGH,
           12 );

IPR1bits.RCIP = 1; // Make the UART Rx interrupt a high priority interrupt
INTCONbits.GIEH <= 1;

```

This should be GIEH as shown here, not GIE as in the video



ME430: MECHATRONICS

Simple Rx interrupt

```

void high_isr(void)
{
    if (PIR1bits.RCIF)
    {
        char newByte;
        newByte = RCREG;
        PIR1bits.RCIF = 0; // Clear the interrupt flag

        // Do something with the newByte
    }
}

```



ME430: MECHATRONICS

More advanced Rx interrupt

```

if (PIR1bits.RCIF)
{
    static unsigned char txCounter = 0; // Makes the variable remember the prior state
    char newByte = RCREG;
    PIR1bits.RCIF = 0; // Clear the interrupt flag
    if(newByte == MESSAGE_TERMINATOR)
    {
        if( newMessageAvailable == 0)
        {
            clearBuffer(newRxMessage);
            loadNewMessage();
            newMessageAvailable = 1;
        }
        clearBuffer(tempRxBuffer);
        txCounter=0;
    }
    else
    {
        if(txCounter < MAX_RX_MESSAGE_SIZE)
        {
            tempRxBuffer[txCounter] = newByte;
            txCounter++;
        }
    }
}

```



ME430: MECHATRONICS

Example of code in main for ISR

```

while(1)
{
    if( newMessageAvailable )
    {
        XLCDClear();

        sprintf(line1,"%s",newRxMessage);
        newMessageAvailable = 0;

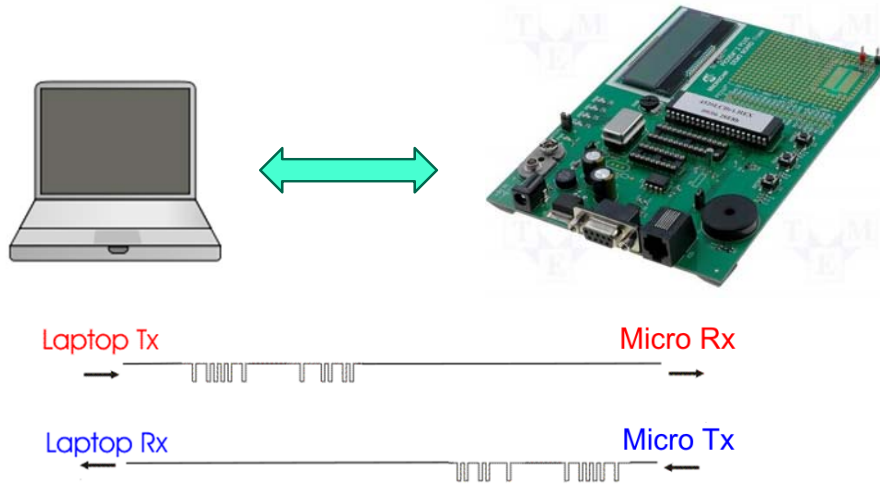
        XLCDL1home();
        XLCDPutRamString(line1);
    }
}

```



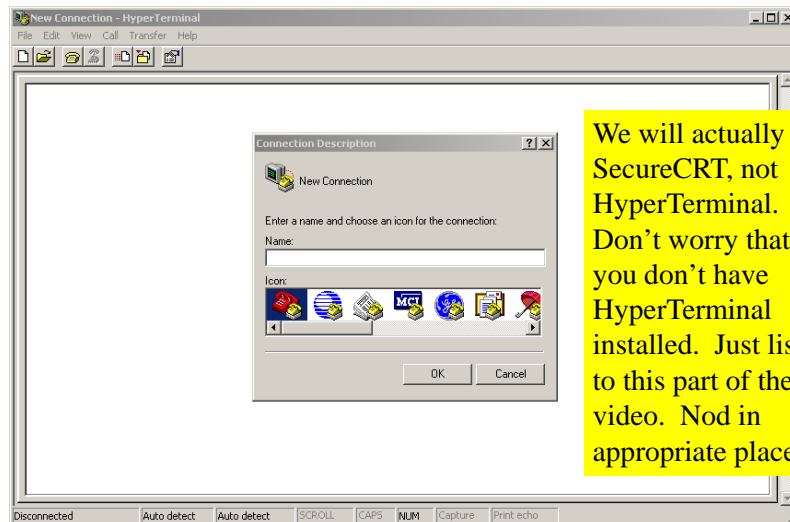
ME430: MECHATRONICS

Other end of communication



ME430: MECHATRONICS

Windows HyperTerminal

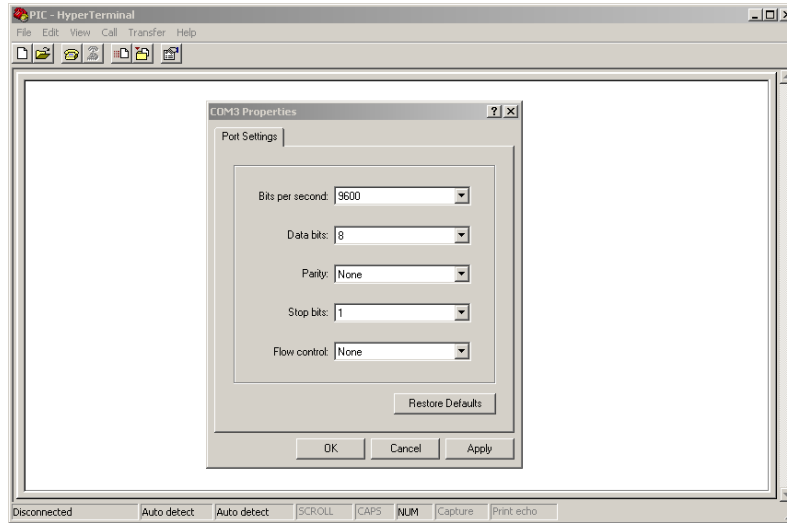


We will actually use SecureCRT, not HyperTerminal. Don't worry that you don't have HyperTerminal installed. Just listen to this part of the video. Nod in appropriate places.

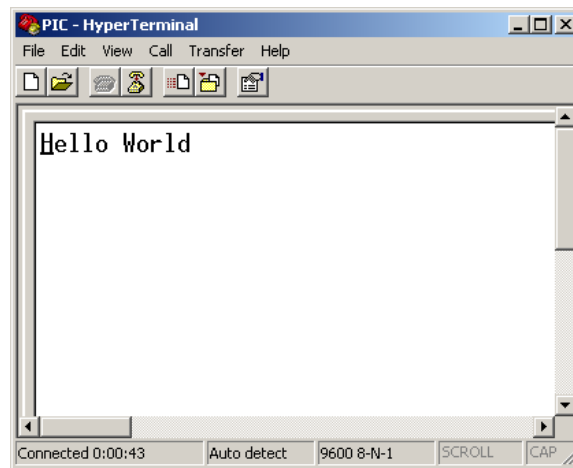


ME430: MECHATRONICS

Setting PC configuration

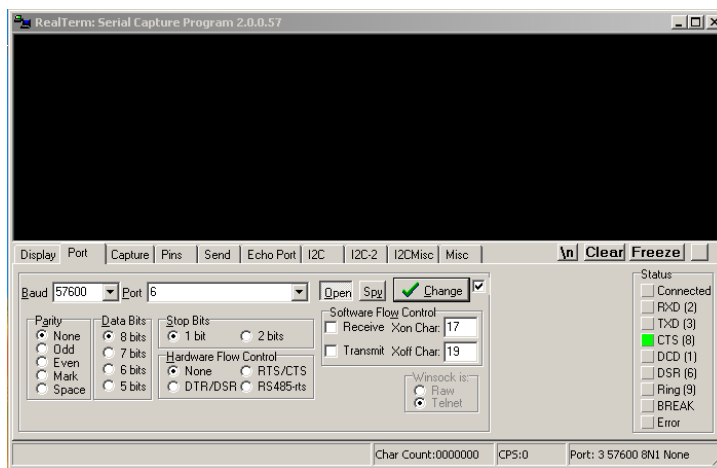


Sending ASCII data



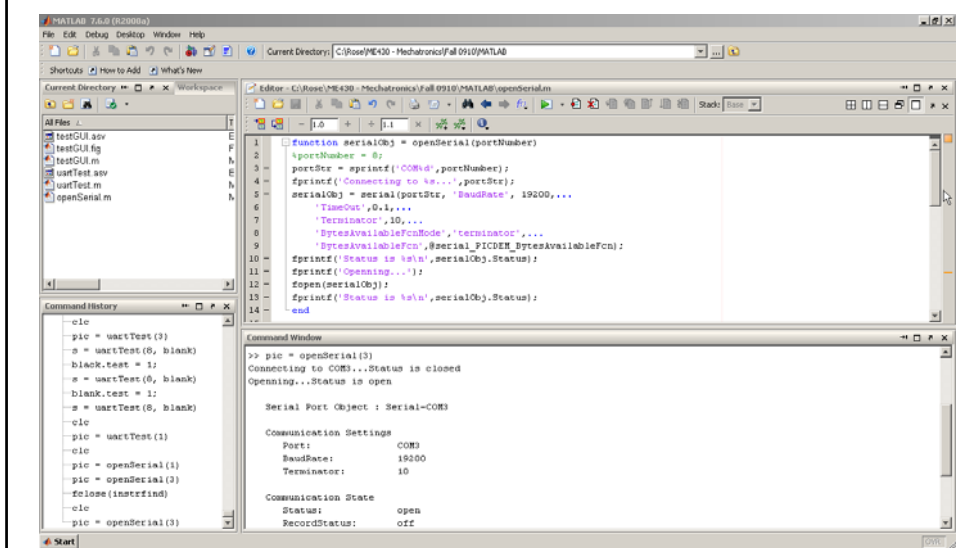
RealTerm

- Better serial program



ME430: MECHATRONICS

MATLAB



ME430: MECHATRONICS