

Control and Navigation 3

Cornerstone Electronics Technology and Robotics III

(Notes primarily from “Underwater Robotics – Science Design and Fabrication”, an excellent book for the design, fabrication, and operation of Remotely Operated Vehicles ROVs)

- **Administration:**
 - Prayer
- **Advanced Control Options:**
 - **Disadvantages of High-Tech Control Systems:**
 - The cost and time commitment for a high-tech control system can be open ended. Be realistic with your financial and time budget before over committing to a control technology.
 - As the complexity of the control system increases, the number of possible breakdowns increases.
 - When an automated feedback control system fails, the results can be devastating. See: <http://www.youtube.com/watch?v=5hfs3vpZO2s> or
 - **Benefits of High-Tech Control Systems:**
 - Fully automated control systems may be a mission requirement.
 - The mission objectives may necessitate more than just “Forward-Off-Reverse” control; a more advanced control system can adjust the intensity, speed, or force of a device.
 - A high-tech control system can make the operation of an apparatus more intuitive for the operator.
 - Automating simple onboard tasks frees the operator to focus on more intricate duties.
 - By incorporating high-tech systems on the vehicle, you learn skills and technological knowledge that can be applied in future projects and career advancement.
 - Improvements can be added in steps as you gain experience with technology.
 - **Possibilities for Control Systems:** Figure 1 illustrates some possibilities for an advanced control system.

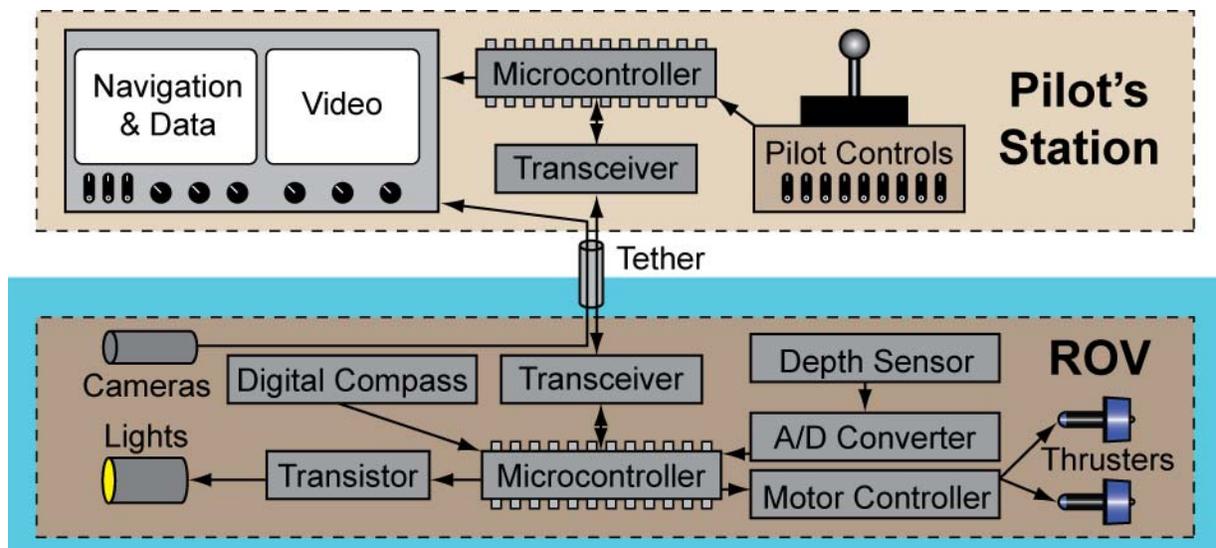


Figure 1: One Possible Advanced Control System

- **Microcontrollers:**

- A microcontroller (MCU) is a digital integrated circuit that can be programmed to control electrical or electronic devices.
- A microcontroller in essence is a tiny, cheap, stand-alone mini-computer on a single chip (no monitor, keyboard, or mouse) that is suited for controlling electrical/electronic applications.

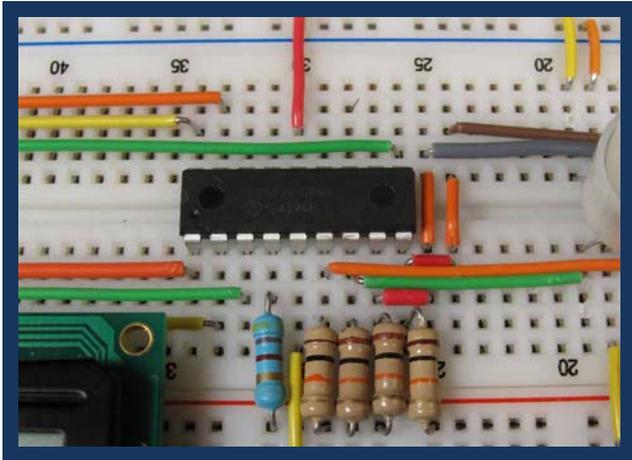


Figure 2: 18-Pin PIC16F88 Microcontroller Figure 3: 40-Pin PIC16F877A Microcontroller

- A microcontroller can monitor multiple sensors, activate warning devices, send signals to visual displays, communicate with other microcontrollers, send multiple commands over a single pair of wires, make decisions, and control multiple output devices.
- Microcontrollers contain the following on the same microchip:
 - A central processing unit (CPU)
 - Memory (both ROM, read-only memory, and RAM, random-access memory)
 - ROM type memory is used to store the program code.
 - RAM is used for data storage, stack management tasks, and register stacks.
 - Some digital input and output ports (I/O ports)
 - The digital I/O ports are the means by which the microcontroller interfaces with the environment.
- Microcontrollers will also contain other devices:
 - Timers
 - Serial and parallel ports to allow data transmission to other devices
 - Analog-to-digital converters (ADC)
 - Digital-to-analog converters (DAC)
- All the components on the microcontroller are located on a single piece of silicon.

- Programming a microcontroller means that the programmer inputs a set of commands into the microcontroller that are executed when the microcontroller is turned on.

- Levels of Programming Languages:

- MCUs are programmed in machine language code (binary code) which looks like:

```
0000100001001001
0001100000000011
0111100000000111
```

- Machine language code is the native language for PIC MCUs.
 - Assembly level code makes programming commands more recognizable; however, it forces the programmer to deal with the MCUs internal structure. Assembly code looks like:

```
movlw      h'07'
addwf     INDF, w
btfsc    STATUS,C
```

- Another difficulty with assembly-level code is that each line of machine code must have a line of assembly code written
 - High-level language: A programmer needs a programming language that relates to problem solving more than the internal structure of a microcontroller. High level computer languages offer formats close to English language. The purpose of developing high level languages is to enable people to write programs easily and in their own native language environment (English).

- MeLabs PicBasic Pro code appears like:

```
For c = 1 TO 100      ' Count from 1 to 100
SOUND 1, [75,100]   ' Generate tone on pin 1
Pause 20             ' Delay 20 milliseconds
Next                 ' Return to FOR and add 1 to c
```

- The most common high-level language for programming MCUs is C.

- Programming the microcontroller unit (MCU) can be programmed either by removing the microcontroller from the circuit and programming it in an adapter or by programming the microcontroller while it is still in the circuit (In-Circuit Serial Programming). See Figures 4 and 5.

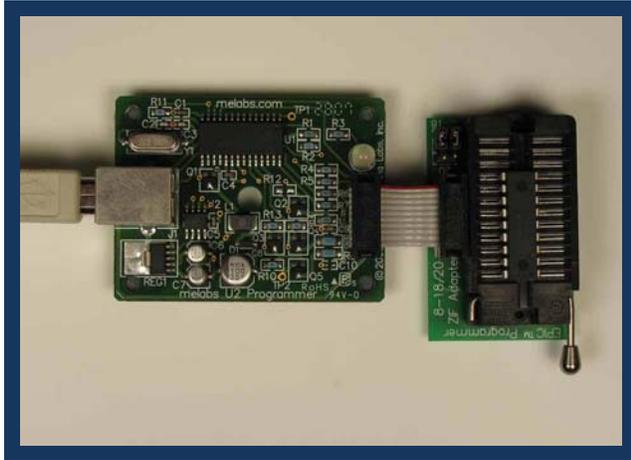


Figure 4: Programming a Microcontroller With an Adapter

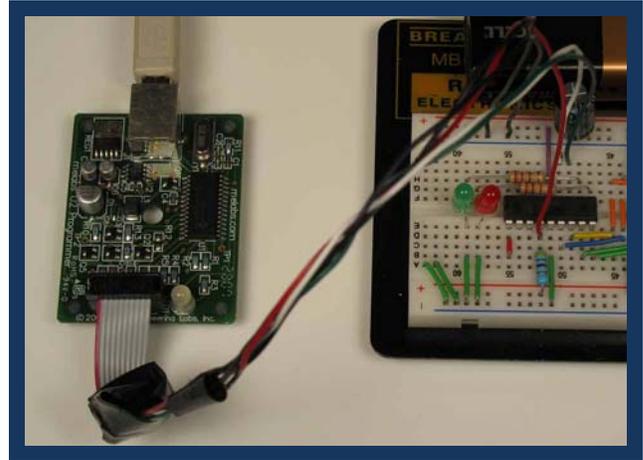


Figure 5: In-Circuit Serial Programming (ICSP)

- Flash devices which can be reprogrammed, e.g. PIC16F88.
 - Program memory can be erased and rewritten 100,000 times.
 - Data stored in the EEPROM can be erased and rewritten 1,000,000 times.
 - Data stored in the EEPROM will be retained over 40 years.
- One Time Programmable (OTP) microcontrollers can only be programmed once.
- For additional programs in PicBasic Pro, see: <http://cornerstonerobotics.org/picbasic.php>
- For a curriculum based upon PicBasic Pro, see Lesson 11 through Lesson 36 at: <http://cornerstonerobotics.org/curriculumyear2.php>
- Microcontroller Pin Functions:
 - Each pin on a microcontroller has a particular purpose. The datasheet for the microcontroller will provide a description for use(s) or function(s) for each pin.
 - Many pins have multiple functions. For example, Pin #6 on the Microchip Technologies PIC16F88 can function as a bidirectional input/output pin (RB0), an external interrupt pin (INT), or Capture input Capture output PWM output pin (CCP1). Refer to the particular microcontroller datasheet to configure the different functions offered by each pin.

- Although pin functions differ from one microcontroller to the next, there are some universal features found on most microcontrollers. For example, the two microcontrollers below are made by two different manufacturers, Atmel and Microchip Technologies. However, they both have pins that perform the same function.
 - The Vcc and GND pins (in red) connect the MCU to power.
 - Both MCUs have a RESET or Master Clear (MCLR) pins (in green) to reset in case there is an operational glitch.
 - Each microcontroller is equipped with two pins (in blue) to connect to a crystal or oscillator circuit that supplies timing pulses to the MCU.
 - Both microcontrollers have a USART (Universal Synchronous/Asynchronous Receiver/Transmitter) pins (in orange) for data transmission.
 - Many MCUs include pins (in purple) that feature analog-to-digital conversion.

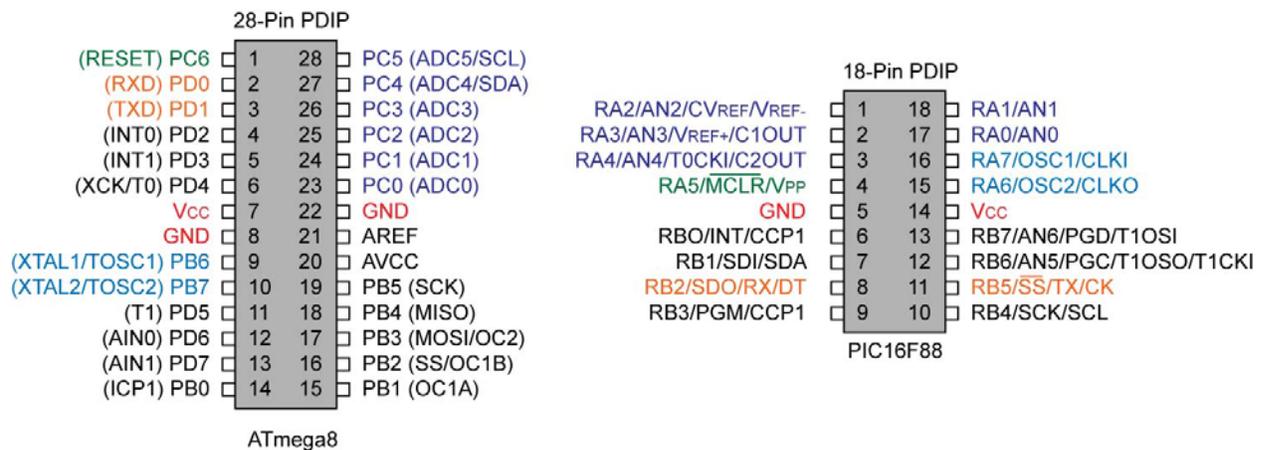


Figure 6: Common Functions on Two Different Microcontrollers

- Most of the pins on a microcontroller are general purpose input/output (I/O) pins. These pins enable the MCU to relate to its environment. Input pins are connected to external sensors or communicate with other devices and receive data from them. Output pins are connected to and control external actuators (like dc motors) or other devices. Typically, I/O pins are at one of two logic states. Logic circuits in general are designed to input and output only two types of signals: "high" (1) and "low" (0), as represented by +3 volts or +5 volts (depending upon the microcontroller) for a "high" state and zero volts for a "low" state.

- A Microcontroller Pin Configured as an Input Pin:
 - A microcontroller input pin monitors its voltage level and the microcontroller can make decisions based upon that input pin voltage.
 - For example, in the following schematic and program, the microcontroller monitors the input pin's voltage and then decides which LED to light. The PIC16F88 monitors the input pin RB0 which is connected to switch S1 and a 10K pull-down resistor. If the switch S1 is pressed, the input Pin RB0 will be forced to +5 volts (RB0 = 1) and LED1 will light and LED2 will be off. When switch S1 is released, RB0 is pulled down to 0 volts by the 10K pull-down resistor (RB0 = 0). LED2 will now light and LED1 will turn off.

Program in "Pseudocode":

```

start:
  IF RB0 = 1 THEN      'If RB0 = +5V
    HIGH RB1          'RB1 +5V
    LOW RB2           'RB2 0V
    PAUSE 100        'Hold 100ms
  ELSE                'If RB0 = 0
    HIGH RB2          'RB2 +5V
    LOW RB1           'RB1 0V
    PAUSE 100        'Hold 100ms
  ENDIF
  GOTO start
  END
  
```

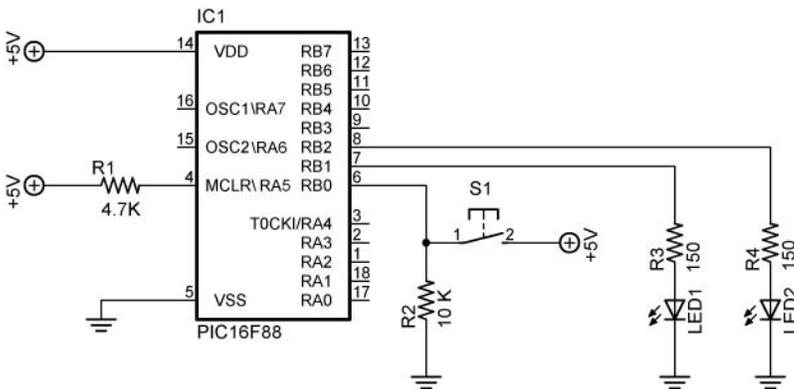


Figure 7: Microcontroller Controlled by Input Pin RB0 and the Corresponding Program

- A Microcontroller Pin Configured as an Output Pin:
 - An output pin sends out either a high or low state that controls a device or another microcontroller.
 - In the example in Figure 7, the two output pins RB1 and RB2 control the LEDs connected to them by causing the pin to go to +5 volts or 0 volts.
 - The output pin can also send a series of ON and OFF pulses to send data or commands.

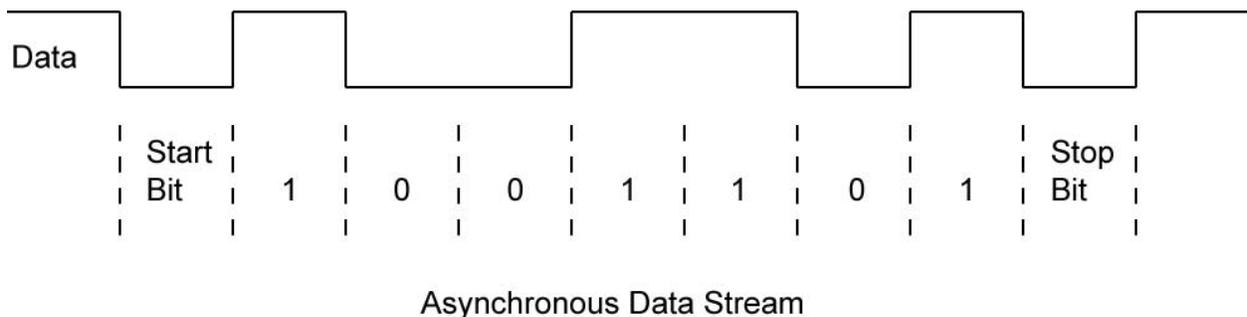


Figure 8: A Data Stream of ON/OFF Pulses

- Microcontroller Limitations:
 - MCUs have limited speed and memory. This inadequacy is normally not a problem for entry level ROVs.
 - Their power output is minimal, normally around 20 mA at 3 – 5 volts. This limitation can be overcome with the use of transistors which amplify the microcontroller output to drive higher powered devices.
- **Introduction to Electronic Signals and Communication:**
 - To be effective, a microcontroller must communicate with devices connected to it. For this to happen, two features must be present:
 - First, a medium, such as electricity, light, radio waves, or sound must be available for use in the propagation of energy from one device to another.
 - Second, there must be a standard language (communication protocol) that the devices use to communicate encoded messages.
 - For example, Figure 9 below pictures the PIC16F88 microcontroller on the right sending an electronic (the medium) asynchronous serial communication protocol signal (the language) to the fiber-optic transmitter. The transmitter converts the electronic signal to light pulses (a change in medium) which are sent through the fiber-optic cable to the fiber-optic receiver. The fiber-optic receiver converts the light pulses back into an electronic signal which is sent to the PIC16F88 on the left. This microcontroller then uses this signal to control the motor speed.

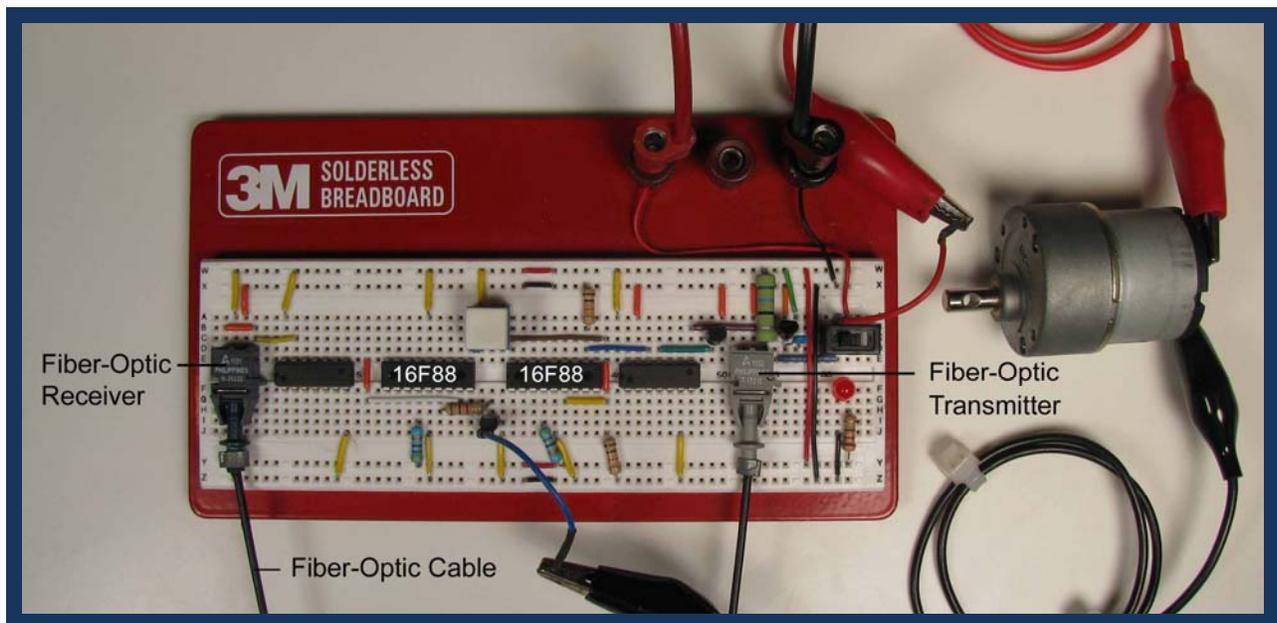


Figure 9: Using Light to Communicate PWM Signals from a Microcontroller

- As the distance between the source of a signal and its destination increases, accurate data transmission becomes increasingly difficult. Electrical distortion can enter the signal.

- Analog vs. Digital Signals:
 - A signal is a fluctuating quantity or impulse whose variations represent information. The amplitude or frequency of voltage, current, electric field strength, light, and sound can be varied as signals representing information.
 - Analog signals:
 - A signal of continuous change without interruption.
 - Example: Potentiometer to control an LED

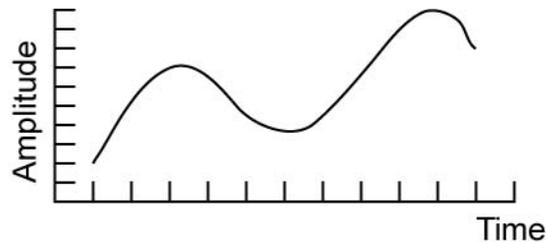


Figure 10: Analog Signal

- Digital Signals:
 - A signal that has discrete values.
 - Example: Digital thermometer or digital multimeter

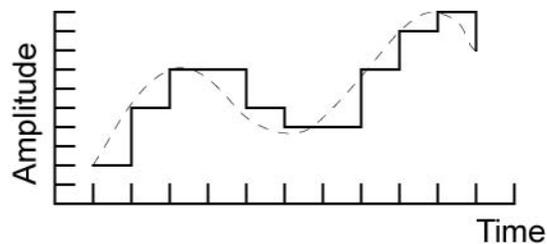


Figure 11: Digital Signal

- Digital communications is normally favored over analog communications since the digital signal is very uniform and noise is less likely to severely alter its shape or amplitude.
- **Analog Data Transmission:**
 - Useful for transmitting signals from analog sensors
 - The simplest way to send an analog signal is to use an analog voltage.
 - The voltage signal suffers degradation over distance since the resistance of the wire creates a voltage drop which increases with distance. Measuring current rather than voltage is a preferred method since the current does not decrease with distance.
 - A voltage analog signal is also prone to interference from other electronics circuits close by and radio frequency waves.

- Another method to transmit analog signals is to use high-frequency oscillations, then modulate the amplitude (magnitude) or the frequency (cycles per second) of the oscillations. This is the technique used to send AM and FM radio signals.

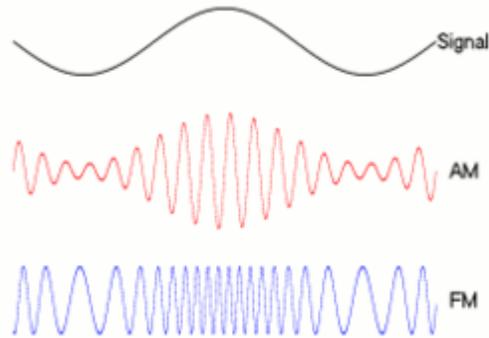


Figure 12: Amplitude and Frequency Modulation

From: <http://iitg.vlab.co.in/?sub=59&brch=163&sim=261&cnt=474>

- Methods of Protecting Analog Signals from Noise Interference:
 - Coaxial Cable: The foil or wire mesh surrounding the center conductor is connected to ground. It partially shields the analog signal that is sent in the center conductor from outside interference.

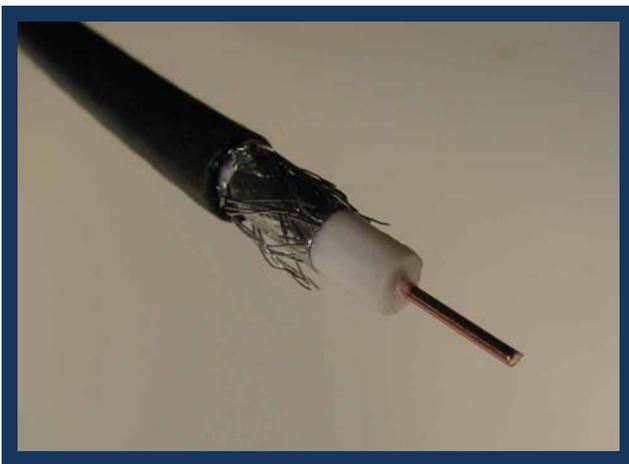


Figure 13: Coaxial Cable



Figure 14: Twisted Pair

- Twisted Pair Wire:
 - The fact that the wires are twisted around each other is significant. This tends to cause external interference to act on one of the wires in a pair in such a way as to cancel the interference acting on the other wire. This is because the wires twisted around each other occupy almost exactly the same space and the current travels in the two wires in opposite directions. The twisted pair is coupled with a differential system to clear up the signal.

- With the differential system each signal is transmitted on two lines at the same time. On one, the signal is transmitted as a POSITIVE (+) signal, on the other as a NEGATIVE (-) signal. At the receiving end of the cable the receiver device gets two signals. Both of them however, have been changed by the noise that penetrated the cable. The changes came in the form of unwanted voltage added to the wanted signal. At this point it is important to note that the unwanted voltage got added to both lines at the same time and by the same amount. The essence of the DIFFERENTIAL system is that the receiver is designed to take the difference between the two signals on the two lines. In doing that, the noise part of the signal, equal on both lines, gets eliminated, and what remains is clear signal.

As indicated above, the DIFFERENTIAL system works well if the noise added is equal on the two lines, i.e. the POSITIVE (+) and the NEGATIVE (-). To ensure that the noise hits both of these lines identically, both of them need to occupy theoretically the same physical space. Practically, the closest we can get to this requirement is to have the two lines TWISTED together tightly. See:

<http://www.connectworld.net/twisted-pair-cables.html>

- There are two types of twisted pair cables, shielded (STP) and unshielded (UTP).
- Video demonstrating benefits of twisted pair cable:
http://www.youtube.com/watch?v=ed0atAyyfF4&feature=results_video&playnext=1&list=PL4734FEF95BA36B69
- 4-20 mA Protocol:
 - Current signals are less susceptible to electrical interference than voltage signals.
 - The 4-20mA current loop signaling protocol has been with us for many years, and despite all the digital advances remains popular. The signal is transmitted via a 4-20 mA current loop from the sensor device to the control system. Only one variable can be transmitted per loop.
 - A current signal is immune to any electrical interference and can be transmitted over long distances.
 - If the line breaks in a 4-20mA system the current drops to zero raising an alarm, whereas in a 0-20mA system this cannot be done thus open circuit cannot be differentiated from a live circuit carrying minimum current.

- Avoid Placing Signal Wires Near High Currents:
 - Wires that conduct large and shifting currents can produce electro-magnetic fields that bring in electrical noise to nearby circuits or conductors.
 - Avoid running electrical signal wires beside these large current handling wires.
- **Digital Data Formats:**
 - Standard codes have been established to interpret the patterns of 1s and 0s used in digital communications.
 - Decimal Numbering System:
 - Based upon ten fingers
 - Decimal has 10 numerals (0, 1, 2,3,4,5,6,7,8, and 9).
 - Example: 306

When the symbols for the first digit are exhausted, the next-higher digit (to the left) is incremented, and counting starts over at 0. In decimal, counting proceeds like so:

0,
 1,
 2, ...,7
 8,
 9,
10, (rightmost digit starts over, and the next digit to the left (in bold) is incremented)
 11,
 12, ...
 ...
 98,
 99, (rightmost two digits start over, and next digit to the left (in bold) is incremented)
100,
 101,
 102, ...

After a digit reaches 9, an increment resets it to 0 but also causes an increment of the next digit to the left.

- Complete Control and Navigation 3 LAB 1 – Counting in Decimal
- Decimal numbering system is a weighted system - that is, the position of each digit in a decimal number is assigned a weight.

Positive Powers of Ten (Whole Numbers)									
Position of Digit Left to Right		7	6	5	4	3	2	1	0
Decimal Weight		10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
Decimal Equivalent		10,000,000	1,000,000	100,000	10,000	1,000	100	10	1

- Recall that:
 - For any integer a, $a^0 = 1$.
 - For any integer a, $a^1 = a$.
 - $10^4 = 10 \times 10 \times 10 \times 10 = 10,000$.
- For whole numbers, each position is given a positive power of ten, e.g., 10^3 .
- A decimal number is the sum of the weights of each digit. For example:

$$\begin{aligned}
 \text{The decimal number } 306 &= (3 \times 10^2) + (0 \times 10^1) + (6 \times 10^0) \\
 &= (3 \times 100) + (0 \times 10) + (6 \times 1) \\
 &= 300 + 0 + 6 \\
 &= 306
 \end{aligned}$$

- **Binary Numbering System:**

- Introduction: Digital electronic circuits can be in only two states: on or off. This two state system is called binary and is suited for computers. 2-Way switches are simpler than 10-way switches.
- The binary numbering system has only 2 different numerals (0 and 1).
- To distinguish a binary number from a decimal number, the prefix % will be added to a binary number, e.g., %1100111.
- Each binary digit is called a bit. A group of eight bits is called a byte. The byte %11010111 in binary is equal to 215 in decimal.
- Counting in Binary:
 - Binary has only 2 different numerals (0 and 1), unlike decimal which has 10 numerals (0,1,2,3,4,5,6,7,8, and 9).

Counting in binary is similar to counting in any other number system. Beginning with a single digit, counting proceeds through each symbol, in increasing order. Decimal counting uses the symbols **0** through **9**, while binary only uses the symbols **0** and **1**. Each 0 or 1 is a **binary digit**, or bit.

In binary, counting is the same except that only the two symbols 0 and 1 are used. Thus after a digit reaches 1 in binary, an increment resets it to 0 but also causes an increment of the next digit to the left:

0,
 1,
10, (rightmost digit starts over, and next digit to the left (in bold) is incremented)
 11,
100, (rightmost two digits start over, and next digit to the left (in bold) is incremented)
 101, ...

From http://en.wikipedia.org/wiki/Binary_numeral_system#Counting_in_binary

- Complete Control and Navigation 3 LAB 2 – Counting in Binary
- Complete Control and Navigation 3 LAB 3 – LED Display of Binary Numbers

- Binary numbering system is a weighted system, like a decimal number, the position of each bit in a binary number is assigned a weight.

Positive Powers of Two (Whole Numbers)								
Position of Bit Left to Right	7	6	5	4	3	2	1	0
Binary Weight	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Equivalent	128	64	32	16	8	4	2	1

This table only shows eight bit positions. Many more bits may be added to the left if needed.

- A binary number is the sum of the weights of each bit. For example, when converting the binary number 1101 to a decimal:

$$\begin{aligned}
 \text{The binary number } 1101 &= (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\
 &= (1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) \\
 &= 8 + 4 + 0 + 1 \\
 &= 13 \text{ (decimal)}
 \end{aligned}$$

- Not all binary codes are weighed. For example, two codes that are unweighed are the ASCII and 7-segment codes.

- **Hexadecimal Numbering System:**

- The hexadecimal numbering system has only 16 different numerals (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,and F).
- To distinguish a hexadecimal number from a decimal number, the prefix \$ will be added to a hexadecimal number, e.g., \$1F.
- Counting in Binary:

Counting in hexadecimal is similar to counting in any other number system. Beginning with a single digit, counting proceeds through each symbol, in increasing order.

0,
 1,
 2,...
 E,
 F,
10, (rightmost digit starts over, and next digit to the left (in bold) is incremented)
 11,
 12, ...
 FE,
 FF,
100, (rightmost two digits start over, and next digit to the left (in bold) is incremented)
 101, ...

- Hexadecimal numbering system is a weighted system,
 - Like a decimal number, the position of each digit in a hexadecimal number is assigned a weight.

Positive Powers of Sixteen (Whole Numbers)					
Position of Digit Left to Right	4	3	2	1	0
Binary Weight	16^4	16^3	16^2	16^1	16^0
Decimal Equivalent	268,435,456	16,777,216	256	16	1

- A hexadecimal number is the sum of the weights of each digit. For example, when converting the hexadecimal number 306 to a decimal:

$$\begin{aligned}
 \text{The hexadecimal number } 306 &= (3 \times 16^2) + (0 \times 16^1) + (6 \times 16^0) \\
 &= (3 \times 256) + (0 \times 16) + (6 \times 1) \\
 &= 768 + 0 + 6 \\
 &= 774 \text{ (decimal)}
 \end{aligned}$$

- Negative Numbers:
 - The most common method of representing signed numbers in digital data formats is the two's complement system.
 - To get the two's complement negative notation of an integer, you write out the number in binary. You then invert the digits, and add one to the result. Search the web for more details.
- ASCII Characters:
 - The **American Standard Code for Information Interchange (ASCII)** is a character encoding scheme. ASCII is a 7-bit character code for representing English characters as numbers, with each letter assigned a number from 0 to 127. For example, the ASCII code for lowercase "r" is 114 (decimal) or 1110010 (binary). Most computers use ASCII codes to represent text, which makes it possible to transfer data from one computer to another. See: <http://www.ascii-code.com/>
- Floating Point Numbers: There are several methods to store a number such as 3.14159 in digital formats.
 - First, the number can be broken into two numbers, one to the left of the decimal, and one to the right. For example, 3.14159 can be stored as two numbers, 3 and 14159.
 - With the next method, the number can be represented in ASCII characters. 3.14159 would be:

ASCII Character	Decimal Code
3	51
.	46
1	49
4	52
1	49
5	53
9	57

- Finally, the floating point number can be converted to scientific notation and stored in parts. For example, IEEE-754 format contains the **S**ign, the **E**xponent, and the **M**antissa:

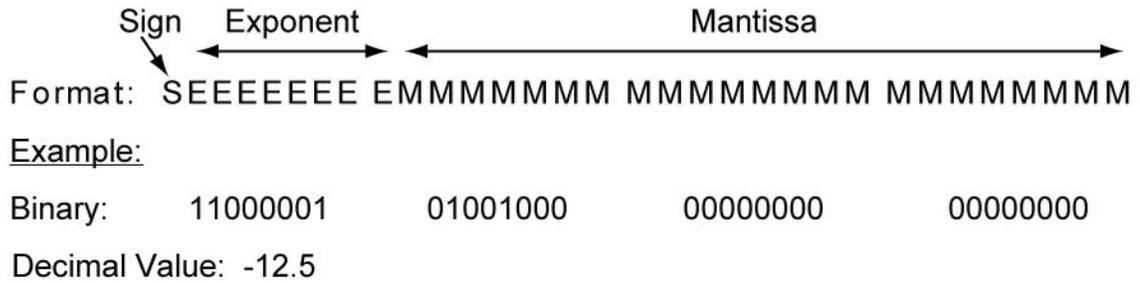


Figure 15: IEEE 754 Floating Point Format and Example

- Images and Sounds:
 - Colors can be conveyed digitally by sending a 1-byte scale (binary - 00000000 to 11111111, decimal - 0 to 255, or hexadecimal - 0 to FF) of the three colors red, green, and blue (RGB color model). See the examples in Figure 16.

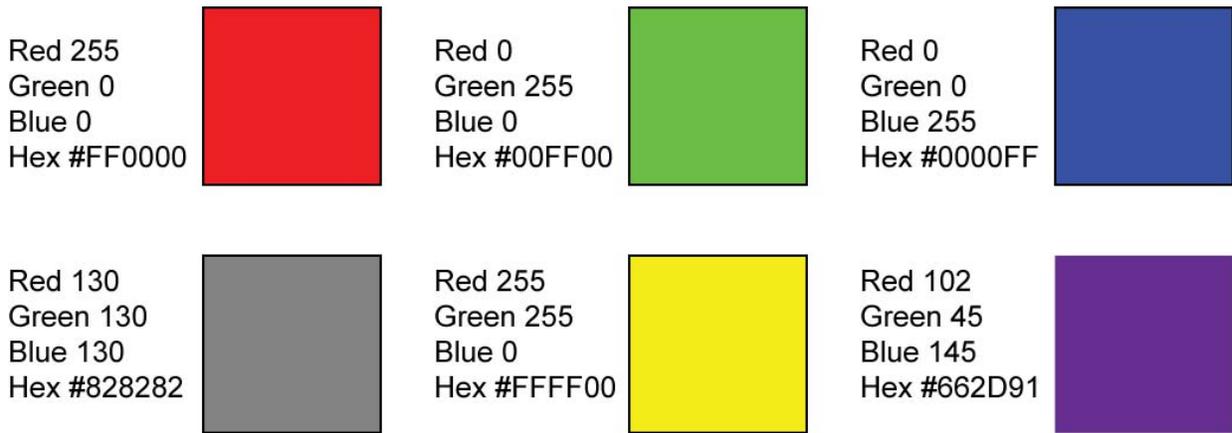


Figure 16: Examples of RGB Color Codes

- Images are formed by denoting RGB color value of each pixel in the image. There can be millions of pixels in an image.



Figure 17: An Image Delineating the Pixels that Make Up the Image

- A movie is a sequence of still images that give the impression of motion.
- Sound can also be converted to a digital format.

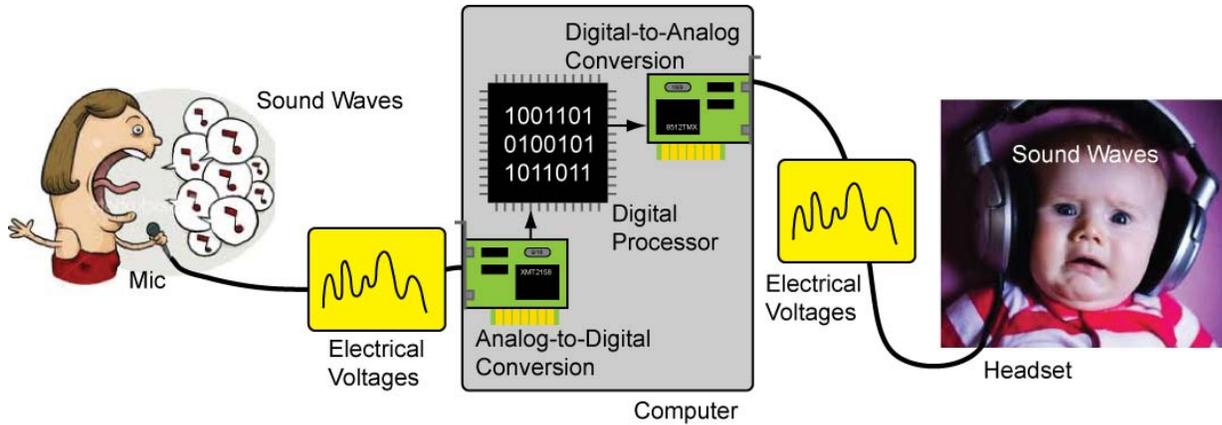


Figure 18: Sound Converted to Digital Signal Then Converted Back to Sound
 Images from <http://www.fanpop.com/spots/singing/images/430336/title/sing-photo>
 and <http://www.smothergoose.com/>

○ **Digital Data Transmission:**

- Serial and Parallel Transmission of Digital Data:
 - Serial data transmission sends one bit at a time over a single communication line.

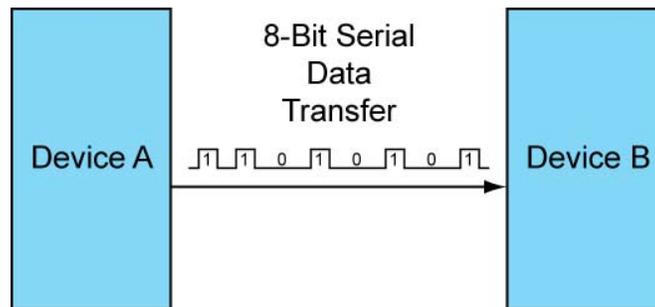


Figure 19: 8-Bit Serial Data Transmission

- Parallel data transmission sends eight bits at the same time over eight different lines or wires.

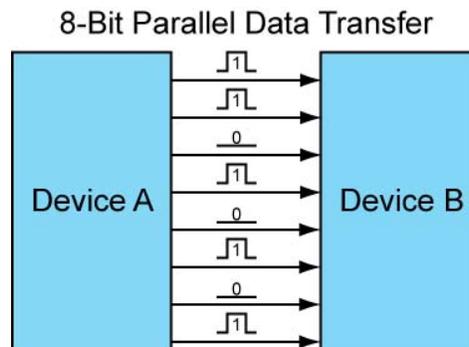


Figure 20: 8-Bit Parallel Data Transmission

- Serial data transmission is utilized in long distance data communication while parallel data transmission is used for high speed data transfer over short distances.
- Serial and parallel data transmissions may include additional wires for support functions.
- Synchronous vs. Asynchronous Data Transmission:
 - All data transfer methods requires coordination between the sender and receiver; creating a need for synchronization between the sender and the receiver. There are two formats for synchronizing between the two ends of the communication - synchronous and asynchronous. Both formats can be used in serial and parallel data transmission.
 - Synchronous Data Transmission: The synchronous signaling methods use two different signals. A pulse on one signal line indicates when another bit of information is ready on the other signal line. In synchronous transmission, the stream of data to be transferred is encoded and sent on one line, and a periodic pulse of voltage which is often called the "clock" is put on another line that tells the receiver about the beginning and the ending of each bit (or byte).

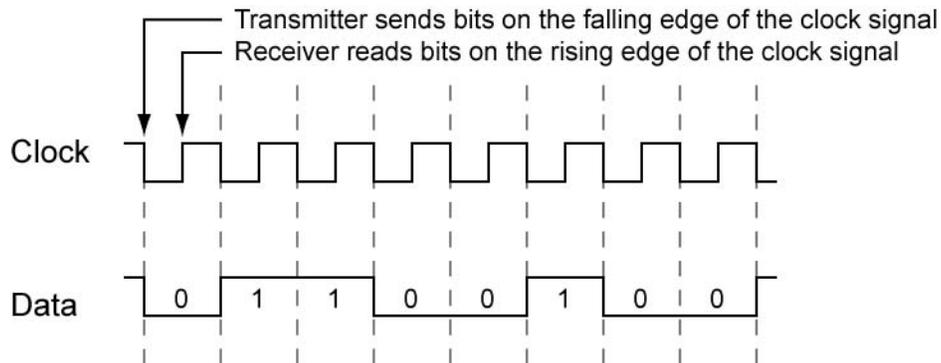


Figure 21: Synchronous Data Transmission Signals

- Asynchronous Data Transmission: A transmission technique that does not require a common clock between the communicating devices; timing signals are derived from special characters in the data stream itself.

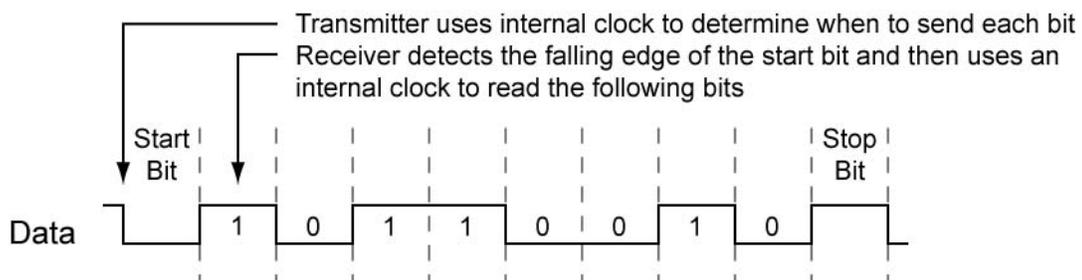


Figure 22: Asynchronous Data Transmission Signal

- Common Serial Data Transmission Protocols:
 - RS-232:
 - For communications over short to moderately distances (30 meters or more).
 - Asynchronous communication protocol
 - Though dated, RS-232 is useful for robotics, ROVs, AUVs, and other computer controlled devices.
 - Formal RS-232 will require a “232” driver such as the MAX232. The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply RS-232 voltage levels from a single 5-V supply. Higher voltages levels are sent over long cables since cable resistance reduces the voltages the further the signal has to travel.
 - RS-485:
 - Improved version of RS-232
 - Asynchronous communication protocol
 - Uses two signal wires and one ground wire
 - More reliable over longer distances
 - More than two devices can share the same set of wires
 - I2C and SPI:
 - Used for communication with close by sensors, other microcontrollers, and other digital devices
 - Refer to the web and books for further information.
 - Ethernet:
 - Not an easy protocol to master
 - Most commonly used protocol for transferring data in computer networks and the internet
 - USB:
 - A fast serial communication standard
 - Used for connecting computer peripherals
 - Not easy to learn
 - Serial-to-USB adapters are available that simplify utilizing USB

Electronics Technology and Robotics II

Digital Fundamentals LAB 2 – Counting in Binary

- **Purpose:** The purpose of this lab is to develop the student’s skill in counting in binary.
- **Materials:**
 - 1 - Pencil
- **Procedure:**
 - In Table 1, fill in the binary equivalent for the decimal given:
 - In Table 2, fill in the next binary number if you are counting:

Decimal Number	Binary Number
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Table 1

Binary Number	%1001
Next Binary Number	
Binary Number	%0001101
Next Binary Number	
Binary Number	%100100111
Next Binary Number	
Binary Number	%00001111
Next Binary Number	
Binary Number	%10001001010
Next Binary Number	
Binary Number	%11111111111
Next Binary Number	

Table 2

Electronics Technology and Robotics II

Digital Fundamentals LAB 3 – LED Display of Binary Numbers

- **Purpose:** The purpose of this lab is to show the student a visual representation of binary numbers and to teach several ways of demonstrating a binary state.
- **Materials:**
 - 1 – Analog/Digital Trainer
- **Procedure:**
 - Connect four consecutive HI/LOW toggle switches to four consecutive LEDs on the analog/digital trainer.
 - Remember that 0 is represented by an off or LOW state (0V) and a 1 is represented by an on or HIGH state (+5V).
 - Give an LED display for each of the following binary numbers:
 - %0000
 - %0001
 - %0101
 - %1111
 - Using the toggle switches, give a visual display of counting from %0000 to %1111

Cornerstone Electronics Technology and Robotics III

Control and Navigation 3 Lab 3 – MAX232 Converter Chip

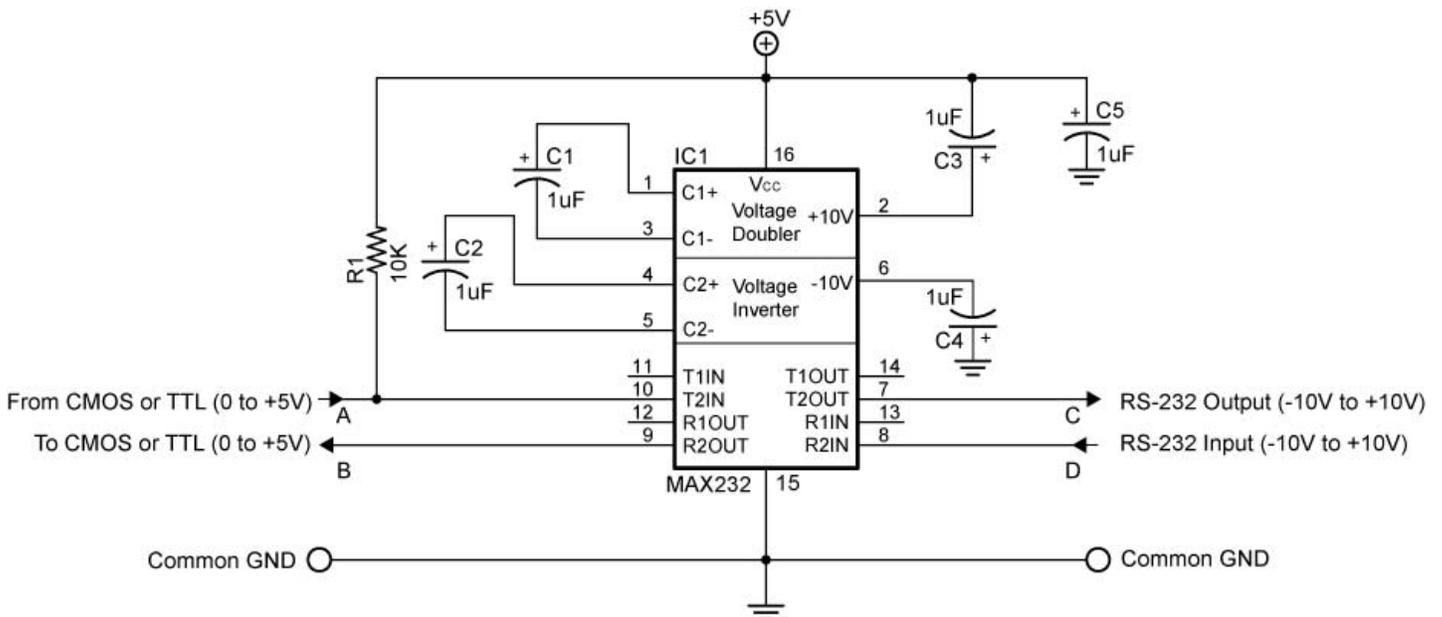
- **Purpose:** The purpose of this lab is to demonstrate the conversion of RS-232 signals to CMOS or TTL signals using the MAX232 converter chip.

- **Apparatus and Materials:**
 - 1 – Breadboard with +5V Power Supply
 - 1 – Function Generator
 - 1 - Oscilloscope
 - 1 – MAX232 RS232 Driver/Receiver
 - Different types of MAX232s require different external capacitors. For example, the MAX232A uses 0.1 uF capacitors.
 - MAX233 and MAX233A do not use external capacitors
 - The 10K pull-up resistor R1 helps with the MAX232 noise sensitivity
 - 5 – 1uF Capacitors
 - 1 – 10K Resistor

- **Procedure:**
 - Wire the circuit below. **Be very careful** that you install the capacitors with the polarity placed in the correct manner.
 - **RS-232 to TTL:**
 - Connect the oscilloscope Channel 1 to Point D, (RS-232 input), and the common ground.
 - View only Channel 1 trace.
 - Set the Channel 1 VOLT/DIV to 5V.
 - Set the TIME/DIV to 0.2 ms.
 - Position the trace in the center of the screen.
 - Also connect the function generator to Point D, (RS-232 input), and the common ground.
 - Set the waveform to generate a square wave.
 - Set the frequency to approximately 1 kHz.
 - Viewing Channel 1 on the oscilloscope, adjust the amplitude of the square wave such that it is from +10V to -10V. This square wave signal will simulate a RS-232 signal input into the MAX232.
 - Now position Channel 1 trace to top of the oscilloscope screen.
 - Connect the oscilloscope Channel 2 to Point B, (To CMOS or TTL), and the common ground.
 - View both Channel 1 and Channel 2 traces.
 - Set the Channel 2 VOLT/DIV to 5V.
 - Position the trace on the lower half of the screen.
 - Apply power to the circuit and verify that the MAX232 converts the RS-232 signal (+10V to -10V) in Channel 1 to a CMOS or TTL signal (0V to +5V) in Channel 2. Notice the inversion of the signal.

○ **TTL to RS-232:**

- Connect the oscilloscope Channel 1 to Point A, (From CMOS or TTL), and the common ground.
 - View only Channel 1 trace.
 - Set the Channel 1 VOLT/DIV to 2V.
 - Set the TIME/DIV to 0.2 ms.
 - Position the trace in the center of the screen.
- Also connect the function generator to Point A, (From CMOS or TTL), and the common ground.
 - Set the waveform to generate a square wave.
 - Set the frequency to approximately 1 kHz.
 - Viewing Channel 1 on the oscilloscope, adjust the amplitude of the square wave such that it is from +2.5V to -2.5V.
 - Use the offset control to make this signal shift to 0V to +5V. This square wave will simulate a CMOS or TTL signal input into the MAX232.
 - Now position Channel 1 trace to top of the oscilloscope screen.
- Connect the oscilloscope Channel 2 to Point C, (RS-232 Output), and the common ground.
 - View both Channel 1 and Channel 2 traces.
 - Set the Channel 2 VOLT/DIV to 5V.
 - Position the trace in the lower half of the screen.
- Apply power to the circuit and verify that the MAX232 converts the CMOS or TTL signal (0V to +5V) to a RS-232 signal (+10V to -10V). Notice the inversion of the signal.



MAX232 Dual RS-232 Driver Circuit