Introduction to Microcontrollers

Slide 1

While this presentation doesn't go into depth about all the microcontroller families in existence, you'll see what people in your neighborhood are doing with microcontrollers. Afterwards, you'll have a chance to try out one for yourself.

Slide 2

This presentation briefly describes what a microcontroller is and how it differs from a CPU. Then some of the ways industry characterizes microcontrollers is described. This way you'll know where the microcontroller you get to try out fits into the grand scheme of things. The easiest to learn microcontrollers use a programming language called BASIC, so you'll see one particular microcontroller that is programmed in this language. I brought some of my microcontroller projects to show you. I hope that seeing them will peak your interest for developing your own microcontroller projects. Finally, we'll wrap up with some practice on a microcontroller that you'll get to test drive.

Slide 3

What is a microcontroller? Essentially, it's a smart chip. That's all. It's a piece of electronics that you can modify its behavior. Not by physical means however. Not for example, by throwing a switch. Instead, it modifies its behavior through the flow of current through semiconductor switches. The rules or procedures for modifying its behavior are stored inside the memory of the microcontroller. Those instructions are called its program.

Unlike the CPU in your desktop PC, a microcontroller is programmed internally with a single application. CPUs are typically programmed externally on hard drives with multiple applications. Microcontrollers typically run at slower speeds and consume less energy. You normally won't see a microcontrollers stored inside a box on your desk like a PC. Instead, microcontrollers are embedded into other products. By being embedded, you're not aware of the microcontroller's presence, but the embedded product acts smarter.

Slide 4

Microcontrollers can sense the environment around them. They can be made aware of the presence of external agents or events. For example, a microcontroller can detect an intruder, be aware of the time, know if its raining or too cold, if its moving and how, and the temperature of your car's radiator.

According to its inputs, a microcontroller can act appropriately. It can move physical objects, control the flow of gases or liquids, record data or events, operate switches, turn on an alarm, communicate with others, or just say "Hello".

Since microcontrollers can sense the world and then react, they have behaviors. These are, for the most part, predictable behaviors. You can instruct a microcontroller to react a

specific way to a specific event and it will do so with fatigue or distraction. In many ways, a microcontroller can be superior to a human when it comes to monitoring events and reacting to them.

Because they sense their world and react, embedded products are more precise in their actions and a whole lot smarter. Being precise lets then react more minutely to subtle changes. Being smarter means, they can detect more of the world around them and then react appropriately. Embedded products have more functionality with fewer parts.

Slide 5

Everyone knows where their PC is located, but where are the microcontrollers controlling the world around them located? Here are some places you'll find microcontroller.

Typical homes have one or two PCs. There can be ore than a dozen microcontrollers in the same house. Microchip, a microcontroller manufacturer, recently announced they had sold over six billion microcontrollers, or nearly one for every person on earth.

Slide 6

Microcontrollers come in all sizes and flavors. Here are a few ways to classify microcontrollers. This is by no means complete.

First, microcontroller are divided into categories based on their architecture. The number of bits that form their instruction sets and how their memory is organized is important. The first microcontrollers were 4-bit. A bit is a single binary value of 0 or 1. Therefore, a 4-bit microcontroller is generally limited to data or commands that are contained within 4 bit width parcels. There are ways to get around this by using multiples of four bit blocks, but they're still using data and commands in blocks of four bits. The first 4-bit microcontrollers were released in the mid 1970's.

The most popular microcontrollers are 8-bit. In fact over 50% of the microcontrollers sold today are still 8 bit even though high end 16, 32, and even 64 bit microcontroller are available.

The type of memory a microcontroller uses is a part of its architecture. Some microcontrollers have program memory designed to be used once. So you had better be sure your program is written correctly. Some microcontrollers contain memory that can be erased and reused as needed. This can be done through UV light or current.

There's a multitude of languages in use to program microcontrollers. The first ones were programmed directly in machine code, or in ones and zeros. Later compilers and interpreters became available so we could program microcontrollers in easy for humans to understand languages like BASIC and C.

Slide 7

Here are two popular microcontrollers that I recommend as jumping off points. Both are 8-bit microcontrollers manufactured by Microchip Inc. However, they're not purchased

directly from Microchip. Instead two different companies program and sell these microcontrollers.

These microcontrollers are marketed primarily as educational products, but are still powerful enough for many industrial uses.

Of the two, the BASIC Stamp by Parallax is the oldest and most established. The PICAXEs typically are simpler, but can contain features not available in the BASIC Stamps.

Slide 8

Here are some of my microcontroller projects. They use either the BASIC Stamp or the PICAXE

Slide 9

How about test driving a BASIC Stamp? I have ten BASIC Stamp Homework boards that you can try. Afterwards, you can take them home. The Homework board is a complete BASIC Stamp 2 designed for portability. You can take it anywhere to practice breadboarding and programming.

Slide 10

On the left side of the breadboard is a single black row of receptacles. This is where you insert wires to make your connections to the BASIC Stamp I/O pins. At the top is another black receptacle. The left side provides +5 volts to your circuit and the right side provides the ground connections your circuit needs.

Connections across the breadboard are made horizontally. There are no vertical holes connected to each other inside the breadboard There is a break across the center of the breadboard and that's provided for integrated chips that may be a part of your circuit.

To connect an LED and resistor circuit to an I/O pin requires a jumper wire. A jumper wire is also required to connect electronic components that are vertical of each other

Let me show you how a light-emitting diode and resistor can be connected to an I/O pin. Afterwards, we'll write a program to make the LED blink.

Slide 11

The circuit we're going to build is a simple LED. Because it's connected to a BASIC Stamp, you'll be able to change the behavior of the LED. It will blink as often and as fast as you want to program it to.

In this side view you'll notice that one lead of a 470 ohm resistor plugs into an I/O port of the Stamp. The other lead plugs into the bread board on the left side of the central trench. The long lead of the LED plugs into the same row as the resistor lead. The short lead of the LED plugs into a hole on the right side of the breadboard. In the same row as the short LED lead, plug in one end of a jumper wire.

Here's one very important point to know. Your LED must have a resistor connected to it. Without a resistor, the current through the LED will be so great that it will either burn out the LED or damage one of the BASIC Stamp's I/O pins. A value of 240 to 1,000 ohms will be perfect. Whether the resistor is placed before or after the LED is not important. Just the fact that it's in the circuit is enough to limit the current and protect both the LED and BASIC Stamp.

Slide 12

From the top, you can see the resistor lead plugs into the I/O lead labeled zero. That jumper wire goes to a hole on the right side, or Vss of the breadboard. Vss is ground. We now have a circuit of the resistor and LED that runs from an I/O pin of the BASIC Stamp, through a resistor, ED, then ending at ground.

Slide 13

Here's what your circuit should look like when you're done.

Slide 14

This is an example of the BASIC Stamp Editor. Programs are developed for your BASIC Stamp in this application and then downloaded into the Stamp. Once downloaded, the program remains inside the BASIC Stamp, whether or not the Stamp has a battery. When the Stamp is powered up, it immediately begins running the program you downloaded into it.

Be sure to save your program in the Editor. Because once the program is downloaded into a Stamp, you can't retrieve it.

The four BASIC commands you need to know for today's test drive are,

HIGH
LOW
PAUSE
GOTO

HIGH connects one I/O pin to +5 volts internally
LOW connects one I/O pin to ground internally
PAUSE halts program execution for a specified time in units of milliseconds
GOTO sends the program execution to a particular location in the program.

For example,
HIGH 3 connects any component connected to I/O pin 3 to +5 volts
LOW 0 connects any component connected to I/O pin 0 to ground
PAUSE 1500 halts program execution for 1,500 milliseconds or 1.5 seconds
GOTO Rabbit sends program execution to the location where the word Rabbit is

You'll also need to know what a label is.

A label can be any word or word-number combination. A label is not a command, so the BASIC Stamp never executes the label. Instead, a label marks a specific location in the program. To make program execution jump to a specific location, use the GOTO statement and point it to the label in question.

Slide 15

Now that you know how to make one LED blink, I challenge you to make a stoplight from three LEDs and three resistors.