Microcontroller Applications in Amateur Radio

July 11, 2009

You have all seen many projects involving microcontrollers, especially PICs. But many homebrew projects don't take full advantage of this tool. In this presentation we will look at the variety of ways in which microcontrollers may be applied.

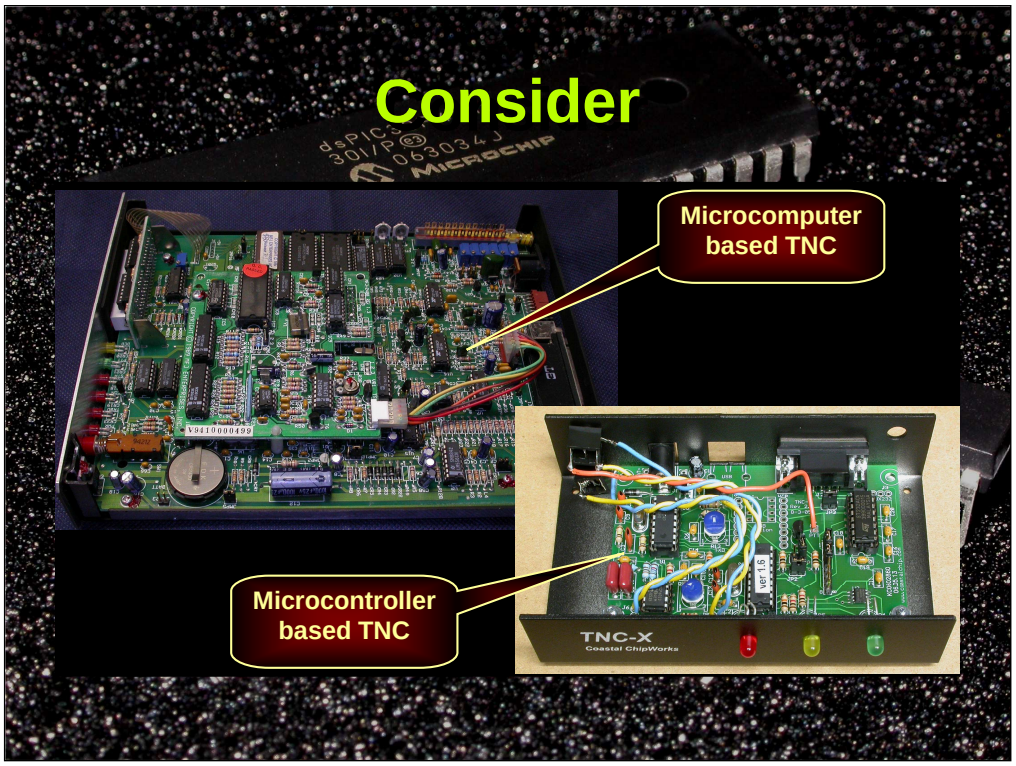
It used to be that any time we needed some computation in a project, it made the project complex and expensive.

Today, however, projects that don't involve microcontrollers have become comparatively complex and expensive.

Back in the bad old days, microprocessors were relatively expensive, and they required support circuitry, which was often complex and expensive itself.

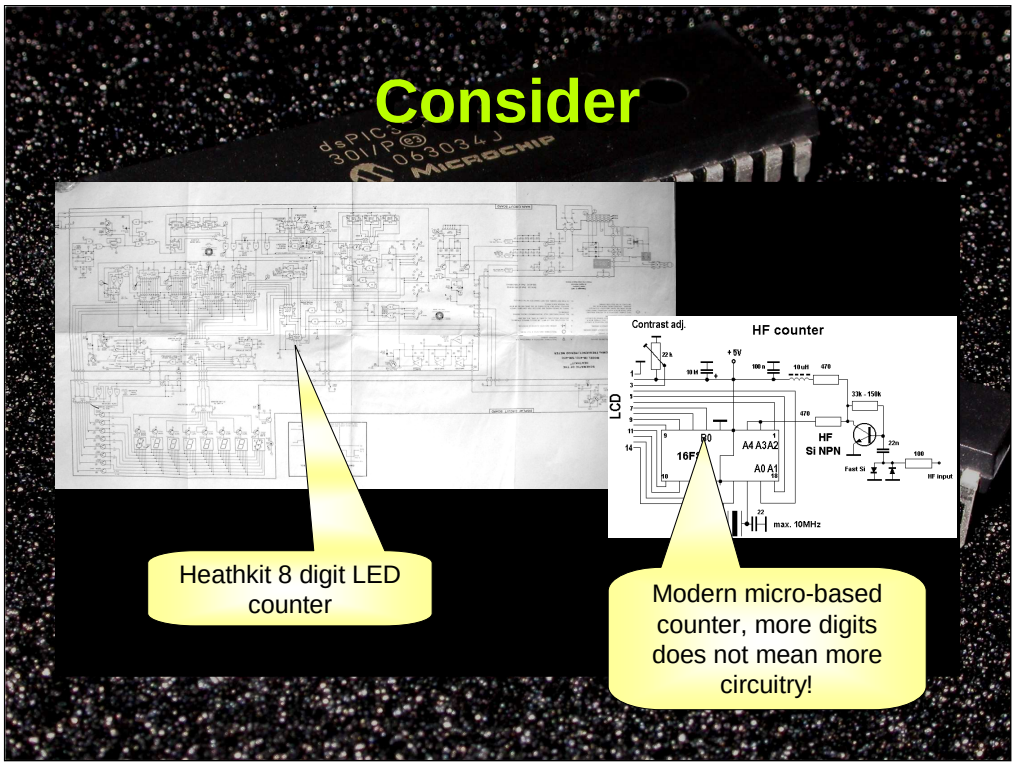
Microcontrollers differ from microprocessors in that they are more self-contained. Over the past couple of years, microcontrollers have become less expensive and more self contained. Today's microcontrollers can be had for less than a dollar, and often require no more support circuitry than a bypass cap on the power supply.

Microchip is far and away the largest manufacturer, having shipped a microcontroller for every man, woman and child on the planet. Although we will occasionally mention other manufacturers, we often use Microchip's "PIC" interchangeably with microcontroller.



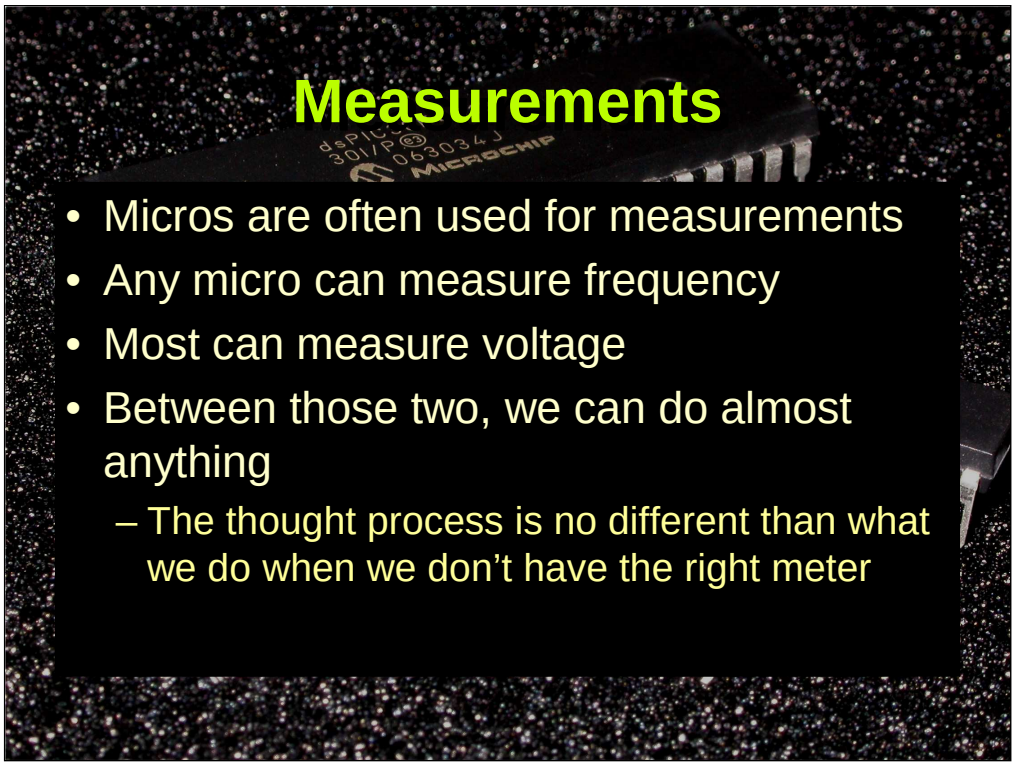
Back in the “old days”, a project requiring computation meant that we not only have a processor of some sort, but also a huge array of “support” chips. Consider the TNC .. The old, Z80-based TNC required not only the CPU, but also complex clock circuits, I/O controllers, external RAM and ROM, an external USART, lots and lots of supporting logic, and various level translating circuits.

In contrast, the modern TNC has a MAX-232 for level translation, and some external RAM in this particular case, but the microcontroller itself include the clock, USART, ROM (actually FLASH), I/O controllers, and most of the external logic. The result is a much simpler, and much less expensive, device.



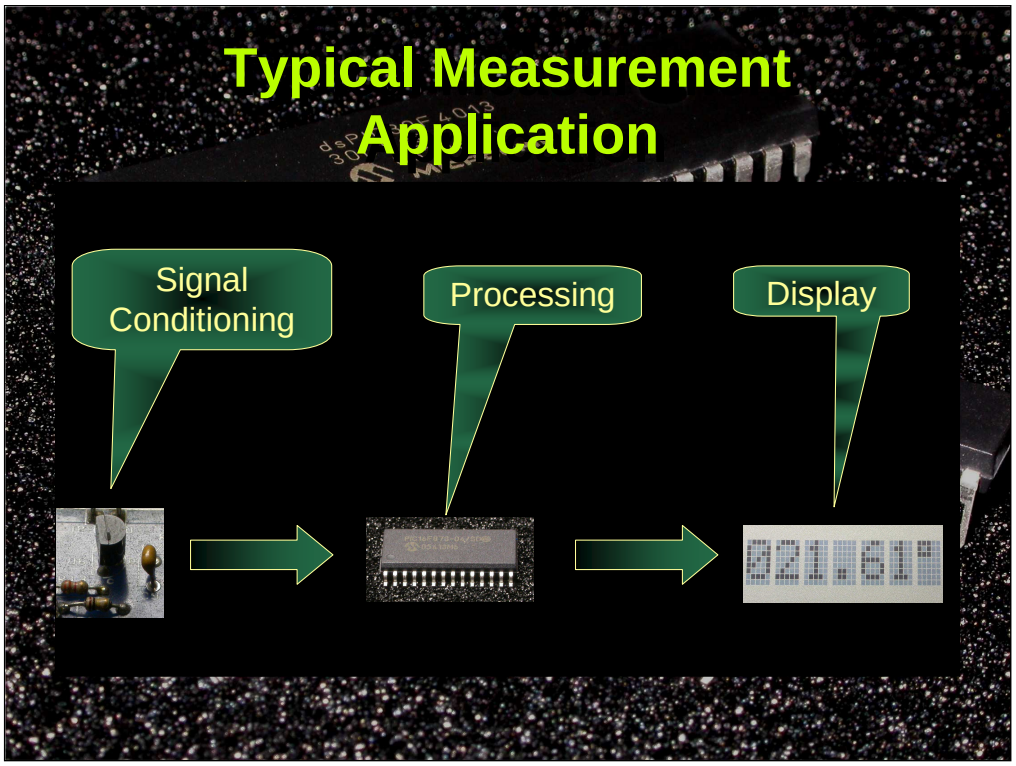
Micros can dramatically reduce the complexity of some projects, especially projects that require digital displays. Consider the frequency counter. As the schematics show, a frequency counter used to be quite a complex thing. Today's PIC-based frequency counter consists of nothing more than a little signal conditioning, a microcontroller, and a display.

The reduction in complexity brings with it a corresponding reduction in cost. Where frequency counters used to cost hundreds of dollars, a microcontroller based frequency counter can easily be constructed for under $15 in parts; less if one can be happy with Morse output!

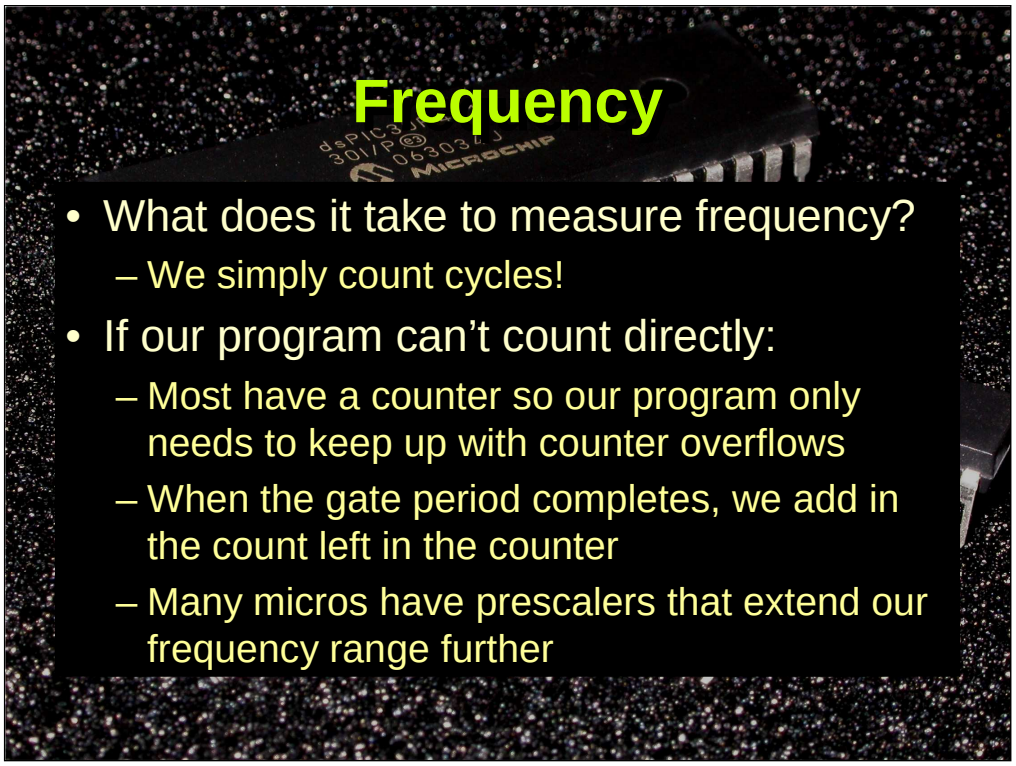


Measurement is an application where microcontrollers seem to be most frequently used. Certainly, it is an area where the benefits are obvious, and those folks who build stuff often don't have the measurement equipment they need. Low cost micro-based measurement equipment is thus a fertile ground for projects.

Any microcontroller can measure frequency, and many have analog inputs. Given those two abilities, along with the ability to do calculations, almost anything can be measured.

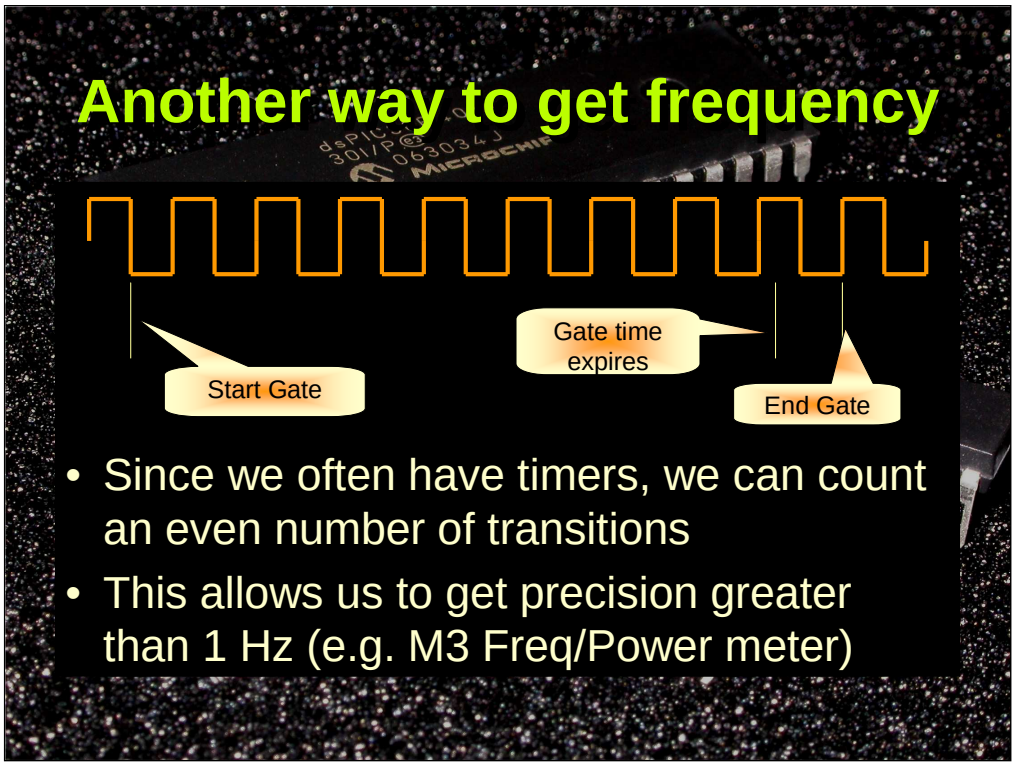


The typical microcontroller based measurement device follows a fairly standard model. First, the input signal is conditioned in some way. Next, the conditioned signal is fed to the microcontroller, and finally, the result is displayed. Indeed, the bulk of the wiring is in the connections to the display, so a micro plus display with some available connections makes a nice experimenter's platform for a variety of measurements.



Frequency measurement is one of the simplest tasks to accomplish with a microcontroller. We call them frequency *counters* because they **count**! All it takes to count frequency is to simply amplify the input signal so it falls in the logic range of voltage, and then count how many times the signal rises and falls over a known time.

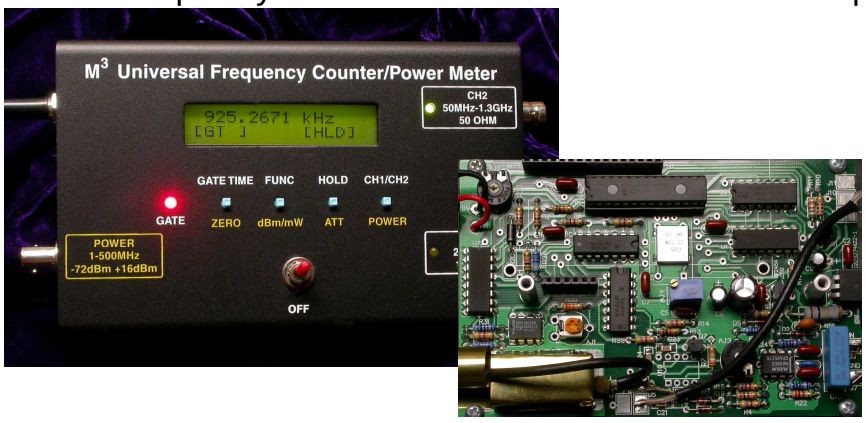
Most micros can execute between one and five million instructions per second. Presuming it takes, say, 10 instructions per cycle to count, this limits the maximum frequency to a few hundred kHz. However, most micros have an internal counter, which can multiply the range considerably, and many also have a prescaler, which can further extend the range. As a result, cheap, PIC-based counters using a 1 MIPS micro can typically count beyond 50 MHz, adequate for most homebrew uses.

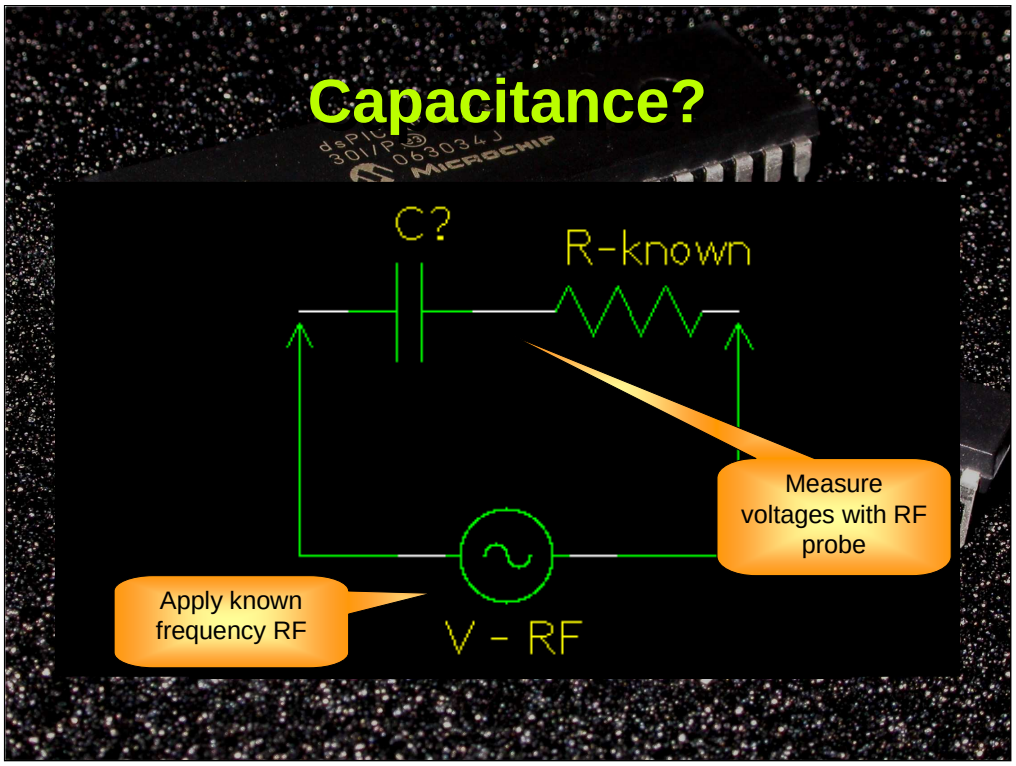


Simply counting has the limitation that the precision depends on the length of time we count. If we wait one second, the best resolution we can achieve is 1 Hz.

But most micros include some sort of precision timer. By counting an integral number of cycles, and measuring the time, the resolution can be increased. Since the microcontroller's oscillator is limited in accuracy to the tolerance of the crystal, this isn't typically an advantage for RF, but for lower frequencies this capability to increase the precision beyond 1 Hz can be useful.

The M-Cubed Frequency Counter/Power Meter uses this technique.



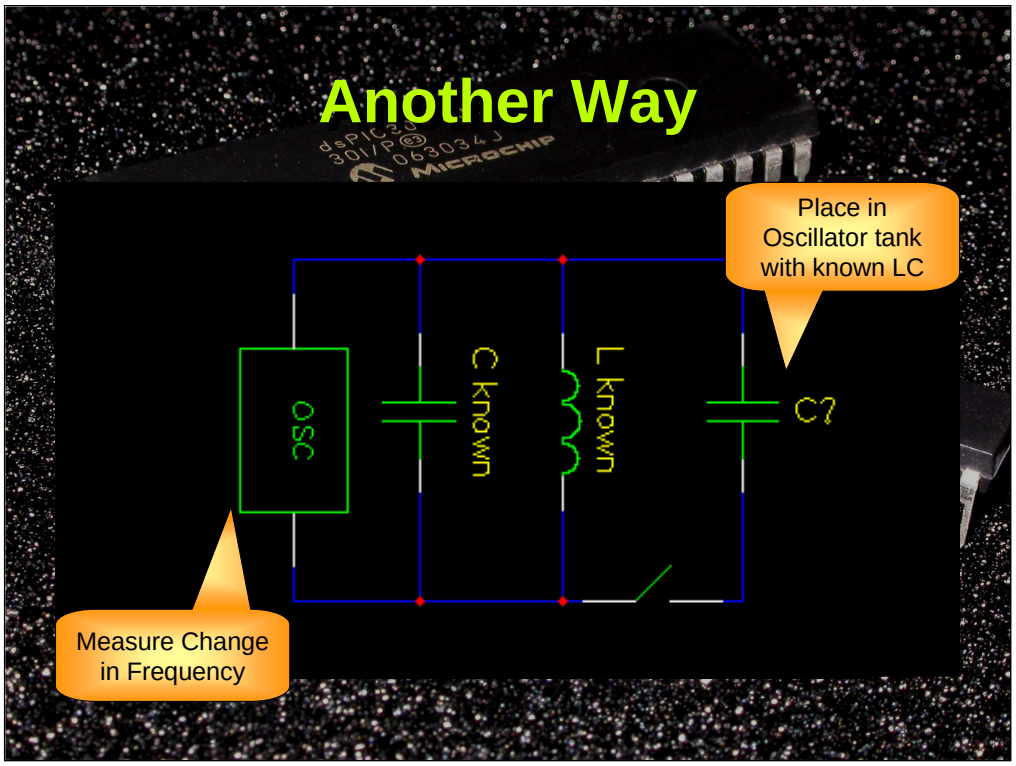


Since we can measure voltage, it is pretty simple to see how we could also measure current. But what about, say, capacitance? Microcontroller based LC meters are common, but how do they work?

If we think about the problem of measuring a capacitor, one obvious approach is to put the cap in series with a known resistor, apply a source of RF from, say, a Pixie or Fireball 40, and using our K7QO RF probe (you have built one of those, haven't you?), measure voltages. By knowing the drops across the R and C it is a simple matter to come up with Xc, and from there, C.

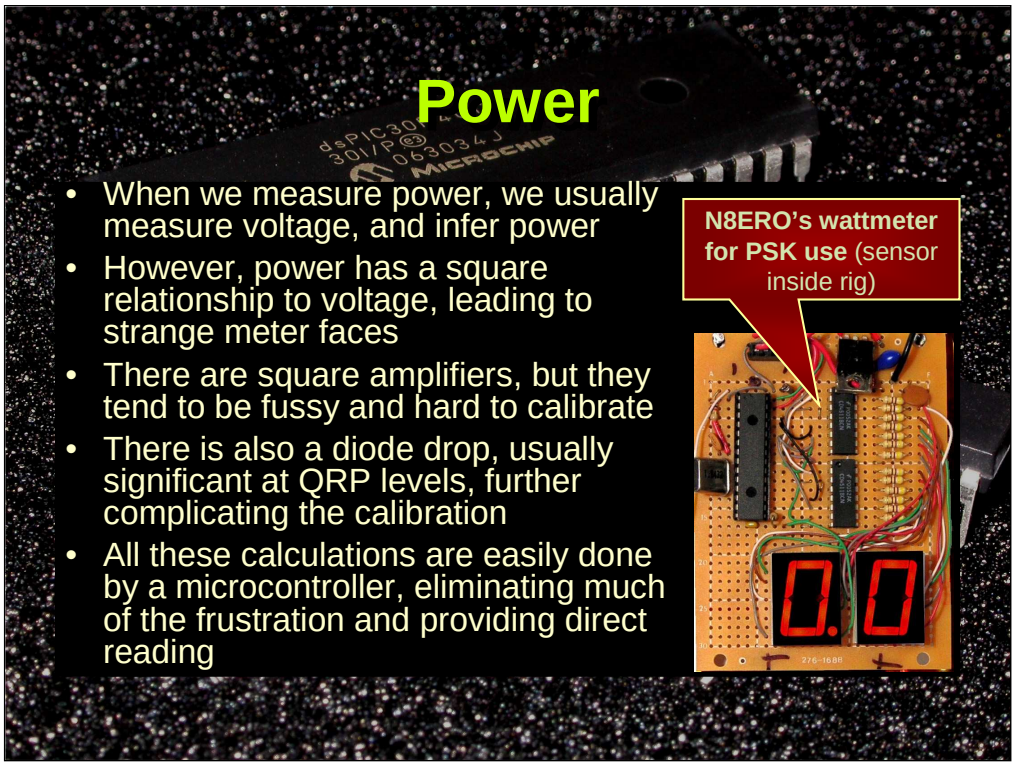
The same approach can be taken using a microcontroller. Of course, one would have to essentially duplicate the RF probe, and it will be necessary to scale the resulting voltage, but the approach is the same.

This is essentially the technique used by the MFJ-259/269 to measure capacitance.



Since the microcontroller can measure frequency so easily, another approach comes to mind. If we place the unknown C along with a known L in the tank of an oscillator, by measuring the resultant oscillator frequency, we can deduce the C. Given an inductance of known value to a high tolerance, this approach can be quite a bit more accurate than measuring voltage due to the ease and precision with which the frequency may be measured.

This is the approach used by the AmQRP Club's Elsie.

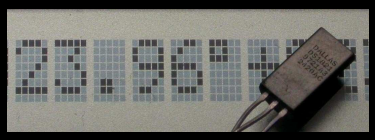


Sometime the micro can help us simplify some messy problems. When we measure power, we typically really measure voltage. But power has a square relationship with voltage, so we end up with messy meter faces, or cantankerous log amplifiers. Further, our sensor typically has a diode drop, which might be ignored at QRO power levels, but typically not at QRP.

With the microcontroller, it is a simple matter to do the math, and then provide a direct reading display. The homebrew example shown was used to manage the levels for a PSK rig. The sensor was inside the rig, and an op amp transmits the analog voltage level to a microcontroller for measurement, calculation and display.

Special Sensors

- Besides the obvious analog sensors, there are many digital sensors
- For temperature, for example, the Dallas Semiconductor DS1821 provides direct reading without the hassle of thermocouples or RTDs



- Analog Devices has digital power and phase sensors
- There are many other sensors to measure almost anything

There are many interesting digital sensors which are useful and compatible with microcontrollers. For example, the Dallas Semiconductor DS1821 (shown) provides direct reading temperature sensing over a single wire. Analog temperature sensing is typically messy, involving amplifiers, wire compensation, etc., and usually involves very small signal levels leading to reliability issues. The DS1821 simply sends you the temperature in degrees … no amplifiers, no calibration adjustments, just the facts, ma'am.

The M-Cubed Power meter uses a digital power sensor from Analog Devices, allowing greater reliability and stability than analog techniques.

There are additional sensors to measure just about anything, most not very accessible without providing some sort of computation in the circuit.

Combined Measurements

- We know we can check a diode with an ohmmeter

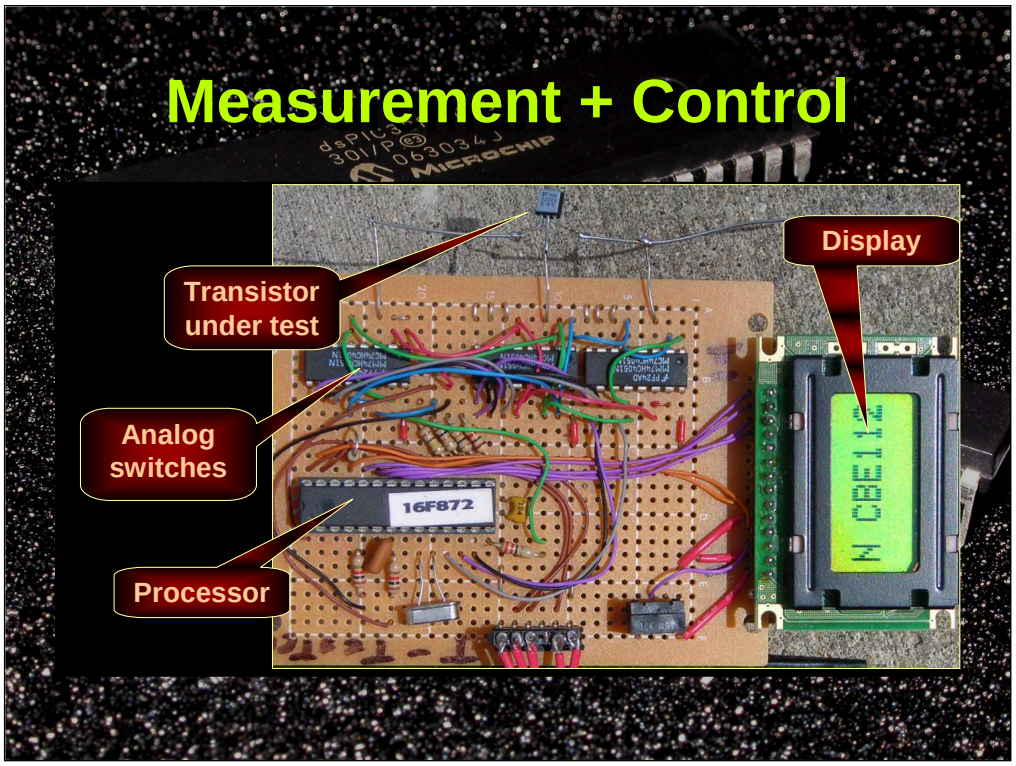
- If we put current through the diode and measured the drop, we could determine germanium or silicon
- If three leads, we could identify NPN or PNP
- If we continued on and calculated the gains, we could identify which lead is which

We have all tested diodes with an ohmmeter, and we know that were we to put current through the diode, the forward drop would be around 0.7 volts for a silicon diode, 0.3 for germanium, so we could at least determine that much very easily.

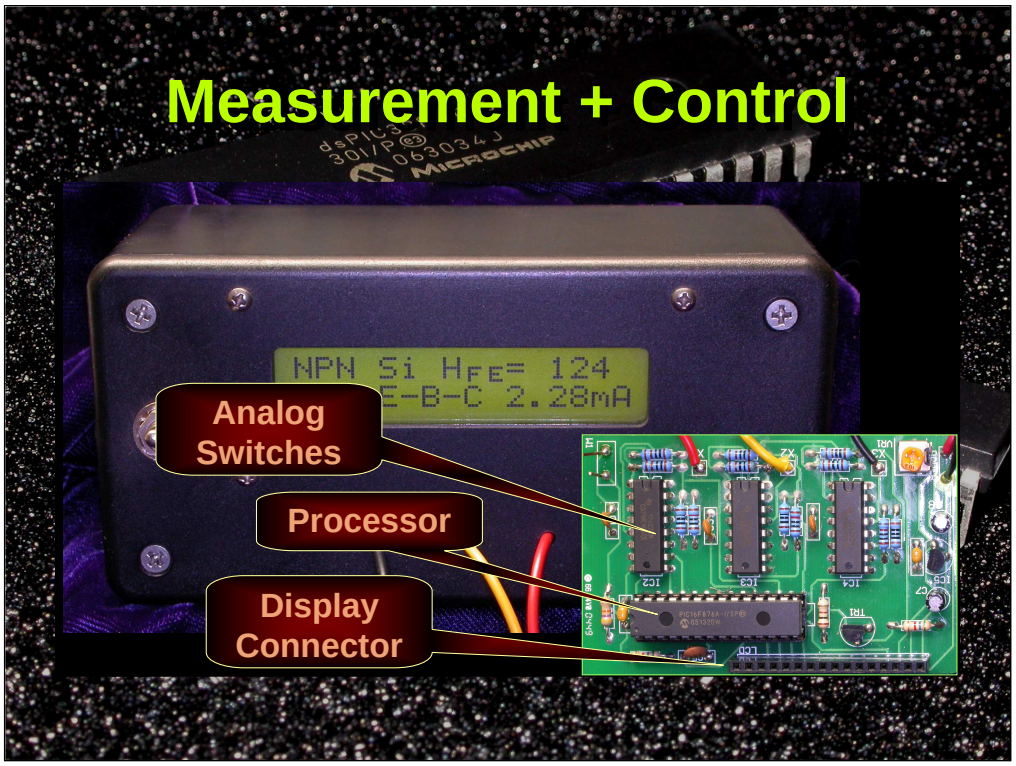
With a transistor, we could identify the base, and whether PNP or NPN by the same sort of technique.

If we took it a step further, and measured the gain in each direction, we could identify the emitter, base, and collector.

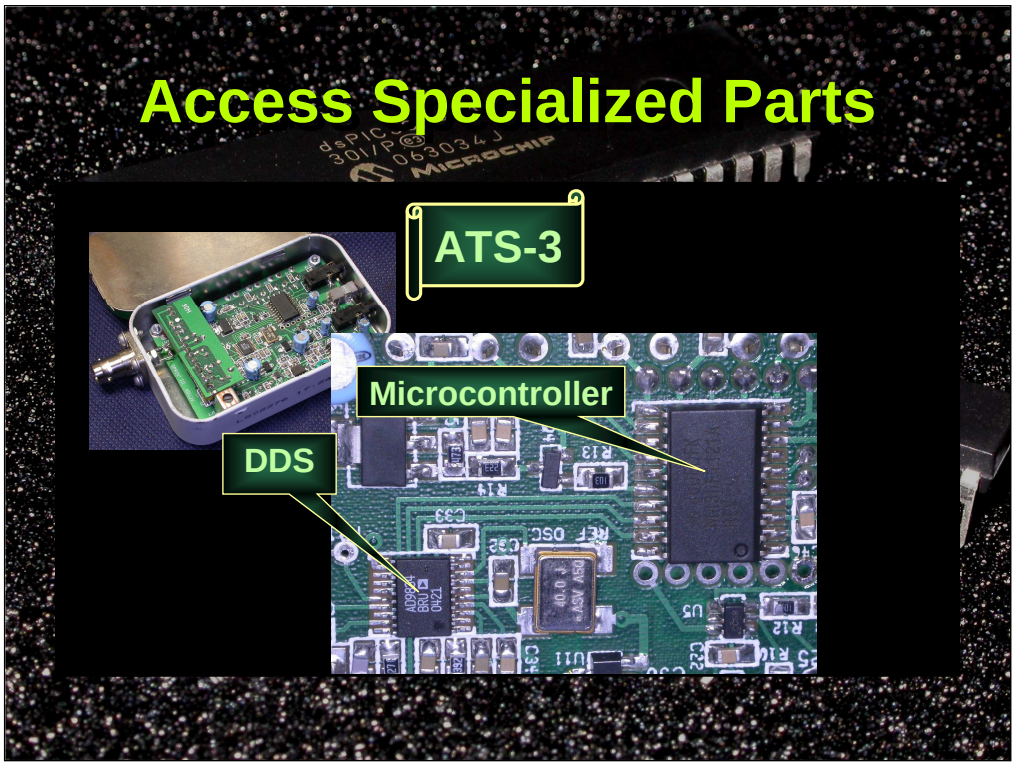
Since the microcontroller can *control*, as well as measure, we could imagine a device with some microcontroller operated switches that would do this analysis for us.



Such a device is well within the reach of the experimenter. Here is a device from N8ERO that does exactly that … it can determine whether a bipolar is PNP or NPN, and can identify which lead is which.

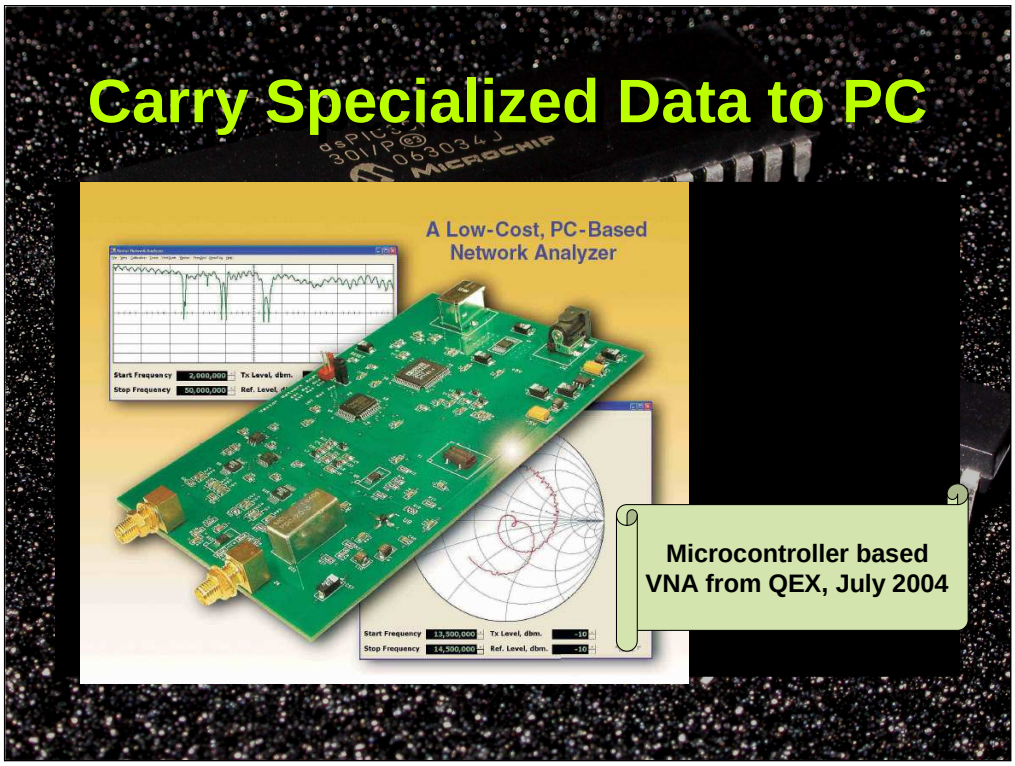


A commercial device, the M-Cubed Semiconductor Analyzer is also available, very similar in circuitry, but more advanced in that it can not only identify the part, but provide the characteristics, and will identify a wide variety of two and three terminal devices.



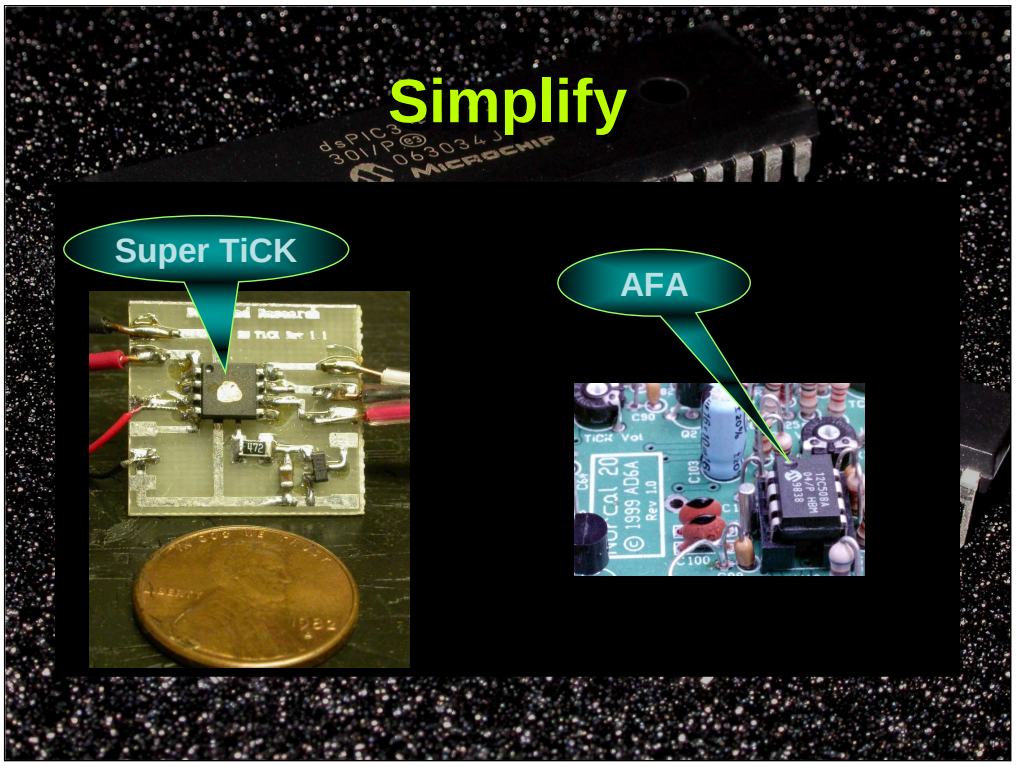
Sensors aren't the only specialized parts that a microcontroller makes accessible. DACs, PLLs, and all sorts of output devices are available and interesting to the experimenter, but messy to use without some sort of computation ability.

Steve Weber's ATS-3, for example, uses a microcontroller not only as a keyer, but as a way to control a DDS chip. DDS provides rock-solid RF over a wide range of frequency without the stability and temperature compensation issues of analog VFOs.



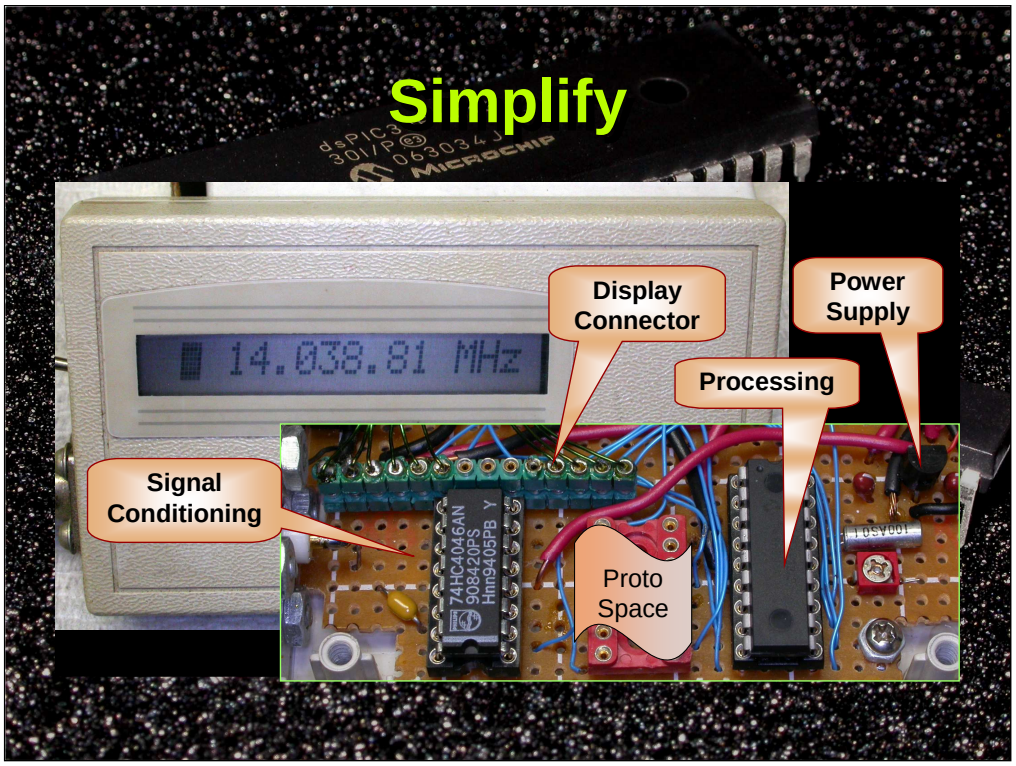
This is an article from QEX.

Here we have an example of access to specialized parts for measurement, plus control, plus interface to more processing (PC). In this case, the microcontroller manipulates the DDS which excites the circuit, and access measurements from two power+phase sensing chips. These measurements are then passed to a PC for analysis. For a hundred dollars or so one can have an instrument that previously would have cost tens of thousands.



For QRPers, probably the most obvious case of using a microcontroller to simplify otherwise complex circuitry is the TiCK. In many QRP radios, this little part replaces dozens of discrete components. Perhaps you don't recall, but older, 7400 based keyers required dozens of parts. Today's keyer is almost always a single microcontroller.

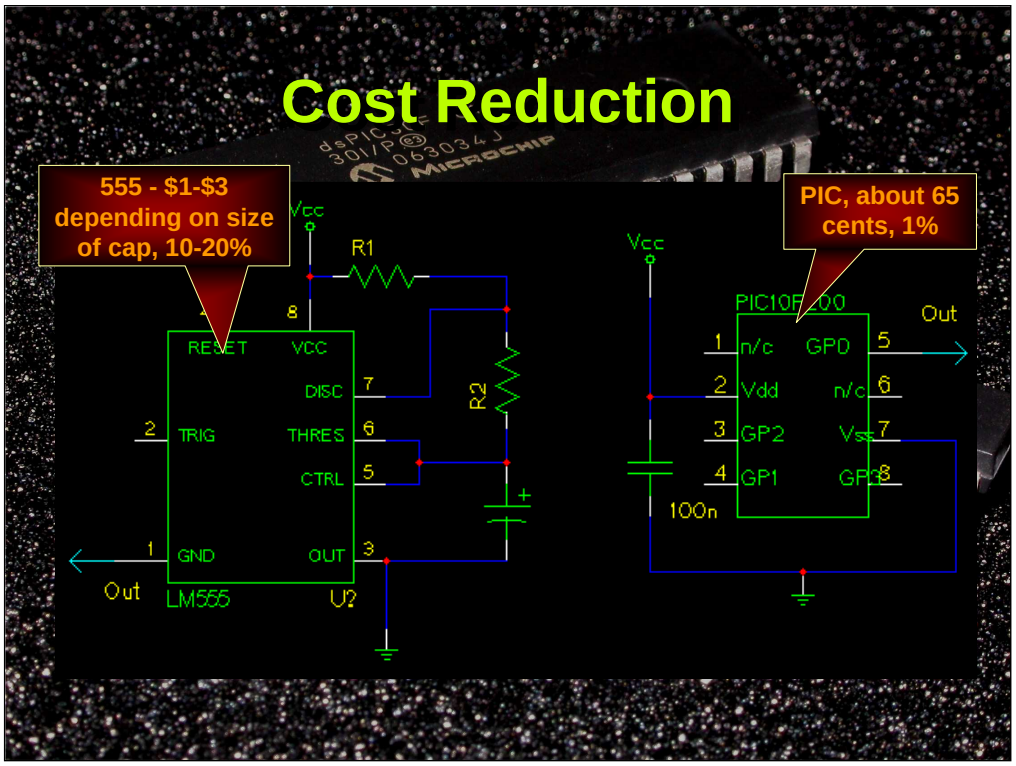
The AFA is another example of a microcontroller being used to reduce not only complexity, but also size. By providing Morse output this little part provides digital frequency output without the space of a display.



Earlier we mentioned counters, and the counter is probably the stereotypical way of using a microcontroller to simplify our designs. Rather than the hundreds of parts needed for a frequency counter in that old century, today's frequency counter involves very little indeed.

Previously, the counter required decade counters to count up each digit, and seven segment decoders for each digit, as well as driver transistors for each segment. An LCD requires only six logic level signals from the processor, which can do the counting and separating the count into digits. The only additional circuitry required is something to take the input signal and turn it into CMOS logic levels.

One bit of complexity in counters we don't see in many microcontroller circuits; not only is there a crystal to set the processor clock more accurately than the internal clock, but there is a trimmer to adjust the crystal spot on the desired frequency (although some designs deal with this in software).



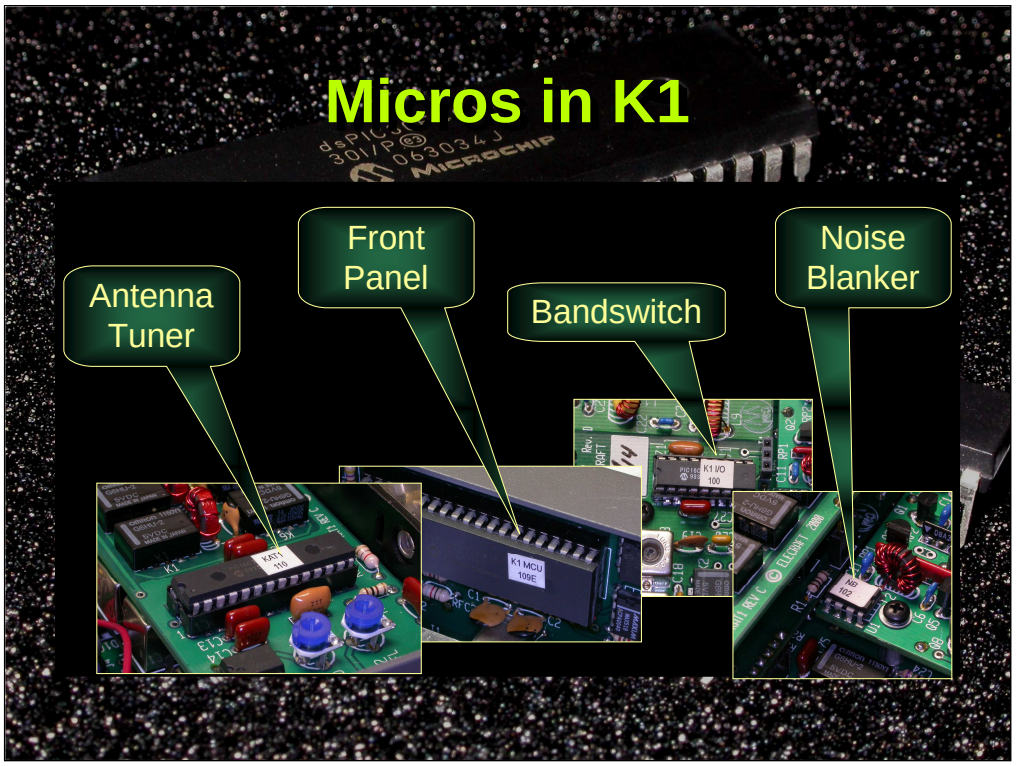
With the dramatic decrease in price of microcontrollers, they may often be applied to reduce the cost of even simple circuits.

Consider a simple 555 based astable multvibrator. The 555 itself is fairly inexpensive, but if a longer period is required, and large value capacitor is needed, and these can get to be quite expensive.

Using a 10F series PIC, which sells for 46 cents, quantity one, the only external part needed is a bypass cap on the power supply.

The 555 itself is accurate to 0.1%, but capacitors with that tolerance cannot be obtained. Even a 5% capacitor could easily cost 6 or 7 dollars. The internal oscillator of the PIC has a 1% precision.

In actual practice, the 555 would probably be driving some other logic, at least some of which would be implemented internally in the microcontroller firmware, further reducing the cost and complexity of the microcontroller approach relative to the 555.

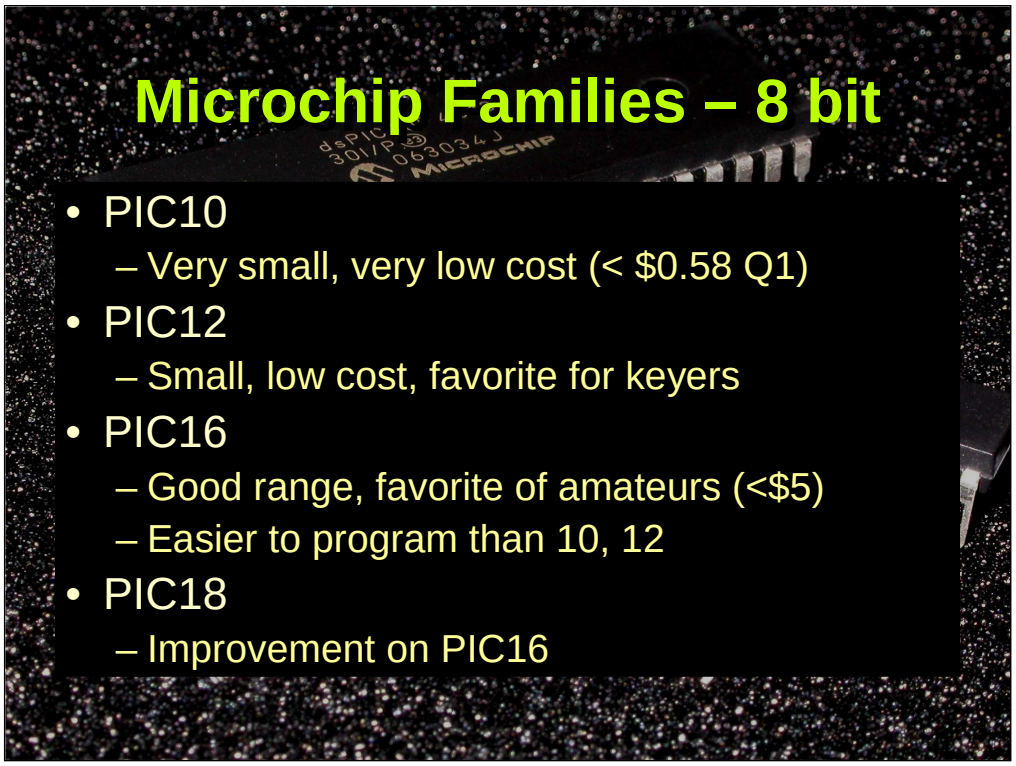


Microcontrollers are often found buried in commercial equipment, both for simplification and cost reduction, as well as to add features that might be complex to implement another way.

In the basic K1, an microcontroller manages the front panel, and another manages the bandswitching, as well as power level.

The noise blanker accessory in both the K1 and K2 is simplified and made more effective through the use of another microcontroller.

Automatic antenna tuners, including those embedded in radios, almost always involve a microcontroller.



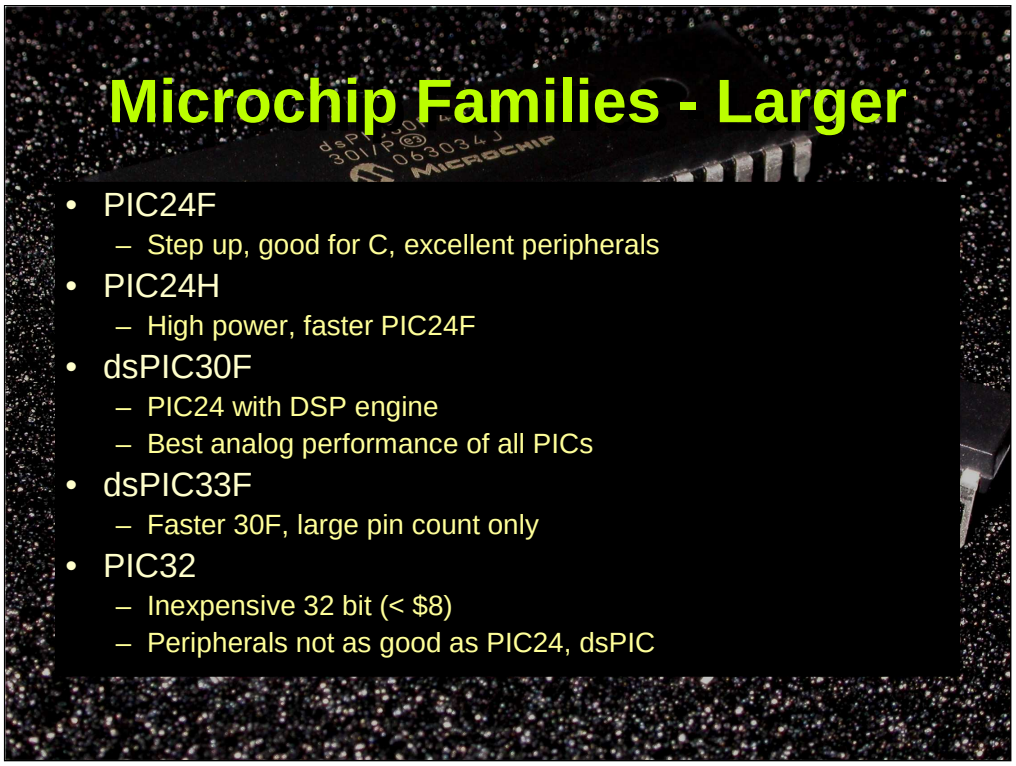
There are quite a range of microcontrollers. Microchip, the big bear in the woods, has perhaps a thousand models. These are arranged into a number of families.

The PIC10 and PIC12 are low cost, low pin count devices. Because of their limited capability, they can be annoying to program for all but the simplest applications. Nevertheless, keyer chips are almost always PIC12's. PIC10's are 6 pin devices, PIC12 8 pin.

The PIC16 is the hobbyist favorite. It fixes most of the programming annoyances of the PIC12 and is available in packages from 14 to 68 pins. The PIC16 also adds a wider range of peripherals. Some parts have A/D converters with front end muxes, I2C and SPI ports, comparators, PWM outputs, etc.

The PIC18 fixes the few remaining annoyances of the PIC16. The PIC18 also adds some more elaborate peripherals.

The 8 bit parts are generally programmed in assembly language. The capabilities are limited enough that using a high level language can be challenging. But there are only 35 instructions on the PIC16, a couple less for the 10/12 and a couple more for the 18, so the assembler is easy to learn.



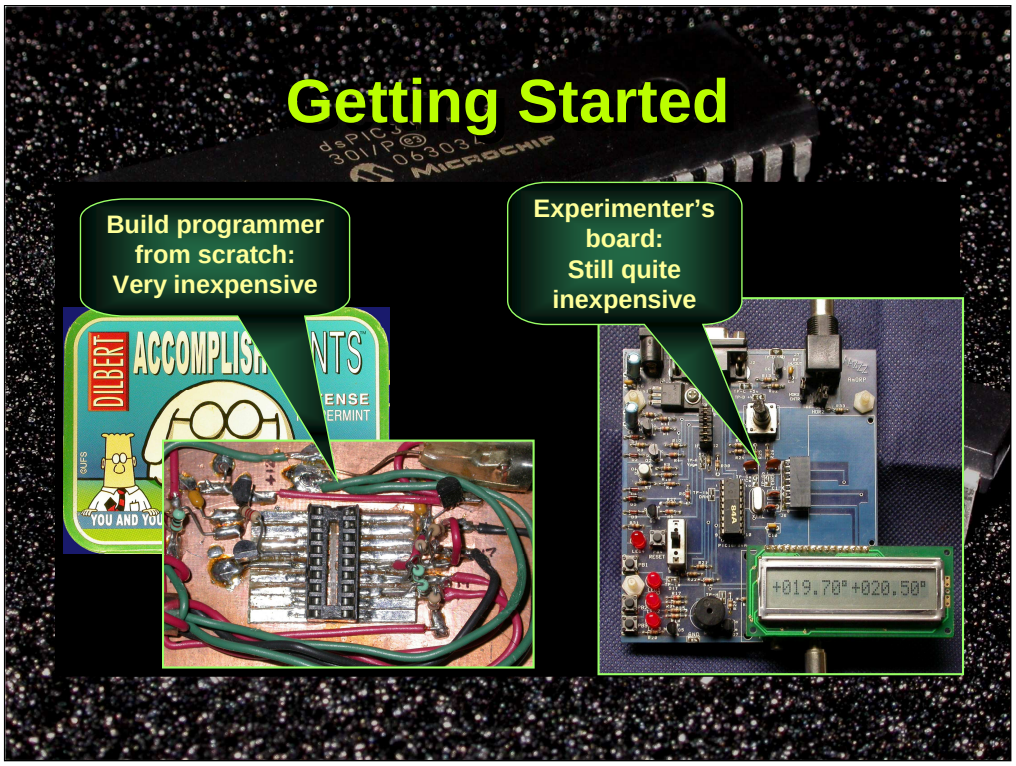
The 16 bit families raise the bar quite a bit. They have a large instruction set, difficult to learn but well suited to high level languages, and the larger memory also makes the high level language more attractive. They all share an instruction set, but the dsPICs add a DSP engine useful for filtering etc.

The 16 bit parts range from 16 to 40 MIPS and tend to have very fancy peripherals, making them startlingly more capable than the 8 bit parts, but nonetheless not a lot more expensive.

The dsPIC30F is the only 5 volt series; all the others are 3.3V. The 5 volt supply gives the 30F an improved analog capability.

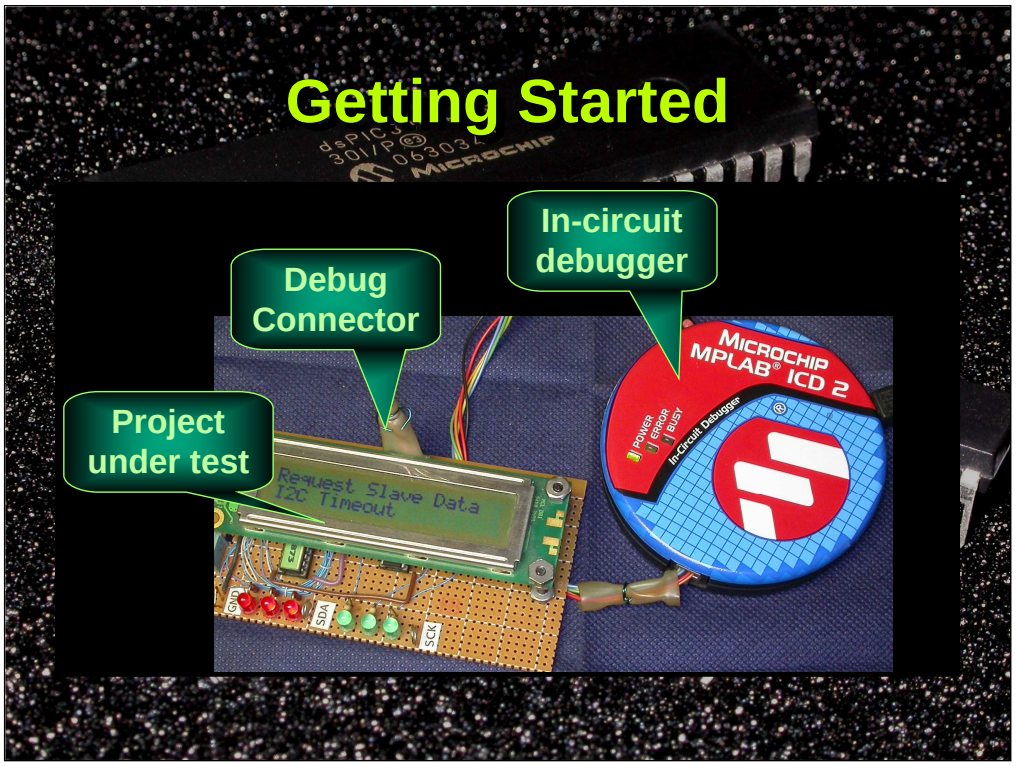
Most PICs can be gentle on batteries and contain special features that allow simple monitoring to be done at power levels significantly below typical battery internal leakage. The 24H/33F is an exception; these parts get performance at the expense of power.

The 32 bit line is relatively new and not as well developed as the other lines. Nevertheless, the parts are quite inexpensive (<$8), and for applications where 32 bit is useful are an interesting addition to the line.



There are a number of ways to get started. With the PIC, building a programmer is a popular choice, and very inexpensive. The programmer on the left is Covington's "No-Parts PIC Programmer", so called because almost anyone would be likely to have the parts laying around. There are dozens, perhaps hundreds of similar programmer designs on the web, all more or less equivalent.

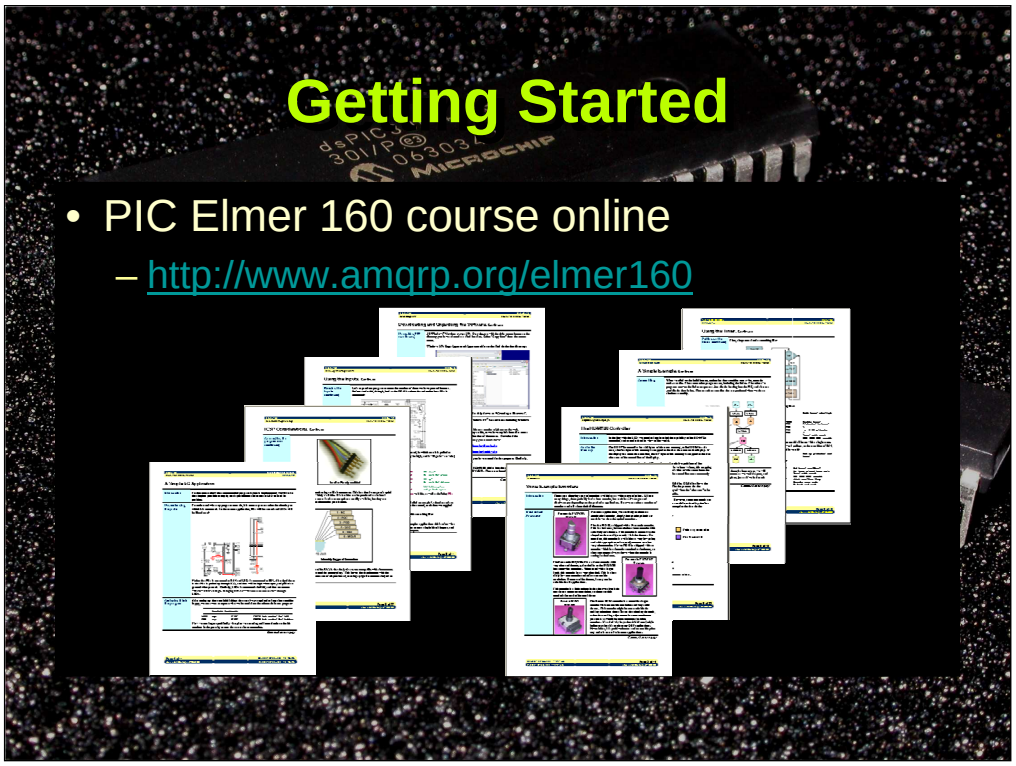
Another alternative is some sort of experimenter's board with a programmer onboard. The PIC-EL, originally kitted by AmQRP and now available from Kanga is another good choice.



For the experimenter interested in a more high-end approach, Microchip's ICD2 not only can program virtually all PICs, but can also debug them in-circuit. While the debugger must be considered in the circuit design, the ability to examine the microcontroller while it is operating in the intended circuit can greatly simplify development in many cases.

Recently Microchip introduced the ICD-3 which adds some nice features to the ICD2.

Other manufacturers have similar debugging and experimenter boards, but because they lack Microchip's reach, they are generally less capable or more expensive.



Shameless plug

Many amateurs have gotten their start in PIC16 experimentation by following my online course, Elmer 160. This course assumes you have a PIC-EL or are willing to build up the individual experiments. The course consists of a series of PDF lessons available free from the American QRP Club website.

Next Steps

- Pick a micro family suited to your kind of projects
  - Research!
- Get equipped to work with that family
  - Equipment
  - Software
  - Knowledge
- **Invent great new stuff**