CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Tom Napier

# Count the Digits

## Designing a Frequency Meter

Sure, you could just buy a frequency meter, but if you're like Tom, you probably have all the necessary parts sitting around, so why not build your own? Before this project is done, you'll have a better understanding of frequency meters.

It's sensible to buy an accurate frequency meter, but a decent one costs more than $500. You can get similar performance from a home-built product that costs about $70 to build. I had most of the parts lying around, so I decided to design and build my own six-digit autoranging frequency meter. The leftover parts slightly hindered the design, so learn from my experience.

I liked the fact that this project involved tradeoffs among analog, digital, and firmware designs. It also involved tricky mechanical design. It was like a commercial design project without the marketing department leaning over my shoulder. And, I didn't have to worry about the manufacturing cost.

### WHAT IT DOES

A frequency meter is a pulse counter that turns on for an accurately known time and displays the accumulated count. Dividing the output of a crystal clock sets the on time. The end count is a linear function of the input frequency.

Twenty-five years ago, you would have strung together a crystal, TTL decade divider chips, display drivers, and as many seven-segment display chips as needed. The only analog part of the circuit was the high-speed comparator, which turned the input signal into appropriate TTL level pulses. It and the crystal oscillator were designed from discrete transistors and resistors, but the rest of the circuit was made from standard TTL building blocks.

Today, the comparator and the oscillator are standard blocks and the dividers and display drive are firmware functions. I used a PIC16C55 microcontroller to count the input pulses and drive the display. Normally, you would use an off-the-shelf LCD unit. I had old seven-segment display chips I wanted to use, so I compromised. Hence, my display has 0.3″ LED digits.

### HOW MANY DIGITS?

Two primary design decisions concern accuracy and resolution. Resolu-

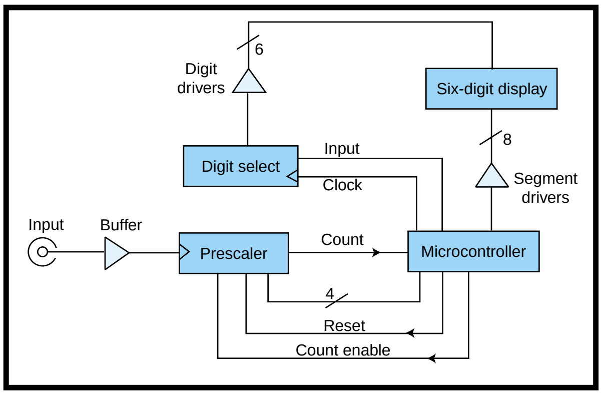


**Figure 1**—*The frequency meter uses a microcontroller as its counting display device. A prescaler extends its maximum input frequency to 50 MHz.*

tion refers to how many different results you can display. I used six digits, giving a resolution of one part per million (ppm). You can add more, but each extra digit increases the counting time by a factor of 10. Resolution is cheap, but it means nothing without accuracy.

Accuracy refers to how well the result compares to a standard. In a frequency meter, the accuracy is a function of the crystal oscillator. You can buy new crystal oscillators for $3 or surplus ones for $1. But, how accurate are these? The standard specification is ±100 ppm.

Does that mean that measuring frequency with six-digit resolution is a waste of time? The answer is no, for a couple of reasons. One reason is that six digits give a best resolution of 1 ppm, but only 5- to 10-ppm resolution when the first digit of the frequency is one. Another reason is that even if the absolute accuracy of a measurement is low, it can measure the difference between two frequencies with a high resolution.

However, a ±100 ppm oscillator can be accurate. Manufacturers state that over a wide temperature range (0° to 50° C), the frequency generated is within ±100 ppm of the frequency marked on the crystal. This error band allows room for how accurately the maker tuned the crystal to the correct frequency and how the frequency varies with temperature.

In most situations, the temperature variation is a parabola or an S-shape. This means that at room temperature, the actual frequency is closest to the nominal value and the variation per degree is less than around the extremes of the range. If you don't plan to use the frequency meter on cold days, you should have no problems and the temperature stability will work well.

The absolute accuracy of the crystal should be better than ±25 ppm. The number of digits stamped on the oscillator is a good guide. An oscillator marked 10.000000 MHz is probably more accurate than one marked 10.000 MHz, although it is unlikely to have the 0.1-ppm accuracy implied by the label. An oscillator claiming ±50-ppm accuracy is worth buying, but a more accurate crystal isn't worth the extra money.

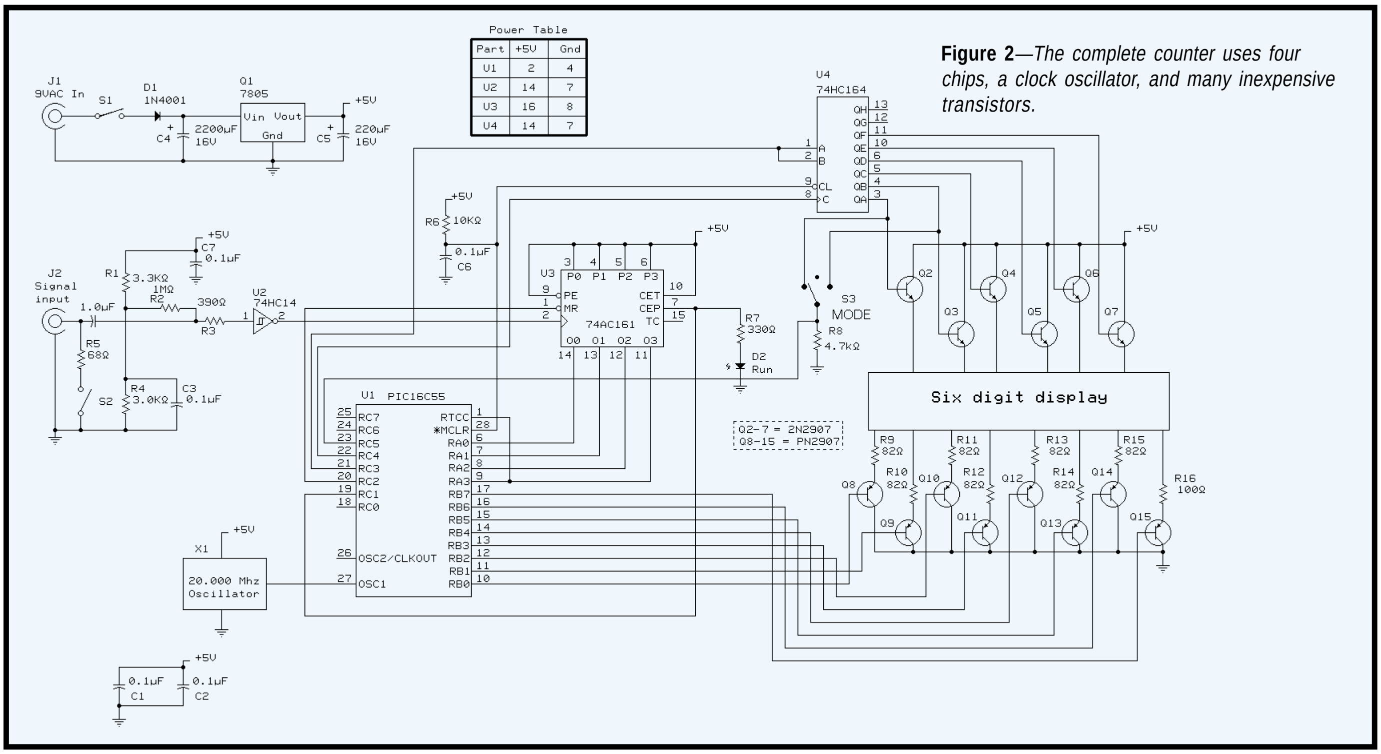
## HOW IT WORKS

The frequency meter counts input cycles for a fixed time period, usually a decimal fraction of one second, then displays the result. Adjust the time period to maximize the resolution so the resulting count is between 100,000 and 999,999. Then move the decimal to give a result in megahertz or kilohertz.

The frequency resolution is the reciprocal of the counting period. For example, when displaying frequencies greater than 10 MHz, the counting period is 1/100 of a second and the resolution is 100 Hz. At the opposite end of the scale, the counting period is 10 s, the resolution is 0.1 Hz, and frequencies up to 100 kHz can be displayed.

Most of the counting is done in the PIC's on-chip registers. What input rate can the PIC handle via its timer input pin, RTCC? Because you need to count every input pulse, you can't use the on-chip prescaler. Although this would let you handle higher input frequencies with the bare chip, you would lose resolution because there is no mechanism for reading the residual count in the prescaler. This count represents the bottom two decimal digits of the count.

Given a 50% on/off ratio in the input, the timer pin can handle periods of 40 ns greater than the instruction period. With a 20-MHz clock, the instruction period is 200 ns, so you can



| Part | +5V | Gnd |
|---|---|---|
| U1 | 2 | 4 |
| U2 | 14 | 7 |
| U3 | 16 | 8 |
| U4 | 14 | 7 |

**Figure 2**—*The complete counter uses four chips, a clock oscillator, and many inexpensive transistors.*

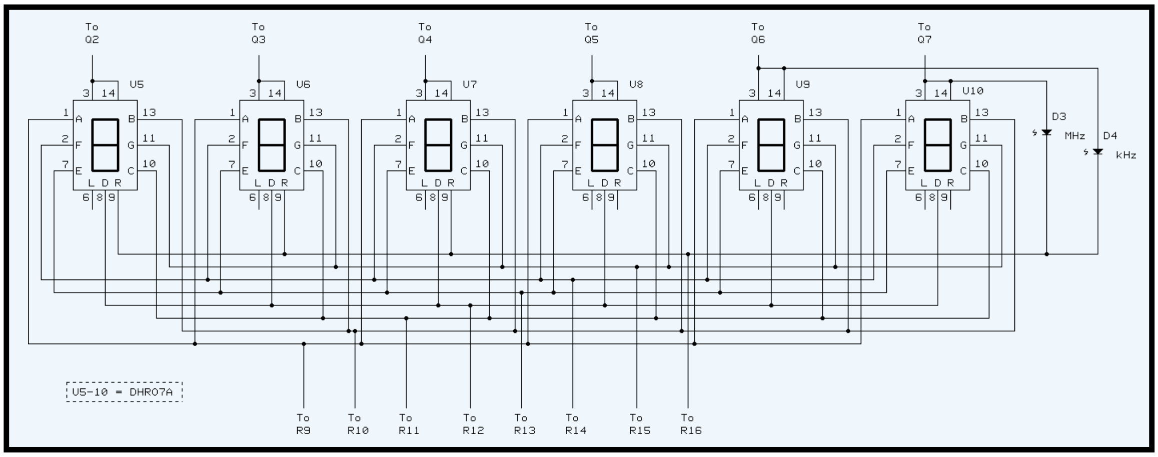


**Figure 3**—*The six seven-segment display digits are wired with the segment cathodes in parallel. The drive to the digit anodes is time multiplexed. Only three decimal points are connected.*

count frequencies up to 4.17 MHz. Because I wanted to measure frequencies greater than 30 MHz, this wasn't fast enough.

## PUSHING THE ENVELOPE

The answer is to add a four-bit high-speed prescaler chip. This allows the input frequency to be 16 times greater. You don't lose resolution because the external prescaler's terminal count can be read after the count period. The theoretical maximum counting rate is now more than 66 MHz. By limiting the frequency meter's specified range to 50 MHz, you gain a safety margin.

The 74AC161 prescaler chip has a count enable input that is switched on and off by the PIC to start and stop counting. The prescaler is reset before each count starts.

One advantage of using a prescaler is that it reduces the effect of an ill-formed or irregular input signal. The 74AC161 handles pulses 2-ns long, arriving less than 10-ns apart, and passes a 3-MHz squarewave to the PIC. Hence, this counter not only measures sinewave inputs, but also the mean arrival rate of pulses that occur in bursts. The rate is sometimes referred to as the "sequency" to distinguish it from a regularly occurring frequency.

The PIC synchronizes its timer pin input to its internal clock and accumulates a pulse count in its 8-bit timer register. So, you can read the timer register without fearing that it will change as you read it. By sampling more than once every 80 µs, you can detect when it overflows and add one to another 8-bit register. The total capacity of this register, the timer register, and the prescaler is 20 bits, or 1,048,576. That's sufficient to store a six-digit full-scale count.

## FIRMWARE CONSIDERATIONS

Because the on-chip timer counts the input, it can't keep track of real time. So, execute a fixed number of instructions during sampling periods. The timing period must be an integral number of instruction times, and the clock frequency must be chosen accordingly. I used a 20-MHz clock.

To update the display, the six LED digits are time multiplexed for 2 ms each and are continuously refreshed. An 8-bit PIC port drives the segments of the six digits in parallel.

Ideally, you would display the input count as it accumulates, but this isn't practical because you can't easily read the prescaler on the fly. Also, you are counting in pure binary and don't want to do binary-to-decimal conversions continually, a 20-bit conversion takes 300 µs.

To help, I converted a bug into a feature. The firmware is either in a count or a display loop. It does the binary-to-decimal conversion between the two. In the count loop, it tests for timer overflow every 50 µs and keeps accurate track of real time. In the display loop, it displays a static result. It tracks passing time to determine when to re-initiate count mode.

However, its time granularity is 2 ms (the digit multiplexing period), and there's no need for accurate timing. Shutting down the display loop during counting simplifies the code. The segment drive is turned off, and the display goes blank. A front panel LED indicates that a count is in progress. The whole job was completed with less than 256 instructions.

Getting the timing correct is tricky. For example, the timer register is sampled every 50 µs (250 instructions), but you can't execute a 1-µs loop 50 times. It takes eight instructions to compare the timer against its last value and to increment the next register if there is an overflow. This means that the loop count has to be only 47.

The next loop counts 200 of these 50-µs loops. It is executed 1, 10, 100, or 1000 times. Whenever a carry occurs in the counter register, the extra code is padded to an even number of microseconds, and the count for the next 50-µs loop is decremented accordingly. The first 50-µs loop has to be shorter than the rest to leave room for the code that turns off the counter chip when the count is done.

## DRIVING THE DISPLAY

The PIC port can't handle the peak segment current (~25 mA per pin) so I buffered the pins with eight TO-92 PNP transistors connected as emitter followers. A commercial product would use a driver chip.

The common anodes of the display chips are driven in sequence. The drive current, if all seven segments and the decimal point are lit, is approximately 200 mA, which is beyond the capacity of a PIC port pin. Again, I used emitter followers. Small NPN switching transistors can handle more than 500 mA. Their mean power dissipation is low. I used generic transistors from Radio

Shack.

There weren't enough spare port pins to drive the digit transistors separately. Normally, the alternative is a decoder chip, but it is has an active-low output. The 74HCT259 can act as an active-high decoder, but I found a solution that only required two PIC pins.

I hooked up a 74HCT164 shift register. If one PIC pin drives its clock and a second pin drives its data input, it's easy to insert a one-bit to step through six outputs to drive the six digits.

The shift register must be cleared when power is turned on to avoid activating several digits simultaneously. To display a count, the shift register is clocked at 500 Hz. A new one-bit is inserted every six clocks, and the appropriate segment pattern for each digit is multiplexed out the 8-bit port. The decimal points and the range indicators are treated as the eighth segment.

To minimize front panel controls, the meter is autoranging. It takes a series of measurements of each input, starting from the 10-ms test period. If the most significant digit of the resulting count is a zero, the firmware switches to the next longer period. If an overflow is detected during the count or during the binary-to-decimal conversion, the autoranger switches to the next shorter count period.

The decimal point position is set according to the range and two LEDs flag MHz or kHz. Because only three decimal point positions were needed, I wired the range indicator LEDs as if they were two more decimal points. This saved using discrete driver pins.

The mode control switch has auto, hold, and start positions. When it is in auto position, the meter cycles continuously. It counts for as long as necessary, then displays the result for 3 s.

If you need longer than 3 s to note a reading, you can switch to hold. If you switch to start, the meter will execute one count and display the result indefinitely. You have to switch to hold, then back to start to take another reading. If you purchase a commercial design, this function would be accomplished with a spring-loaded switch.

| | |
|---|---|
| Display: | Six-digits |
| Frequency ranges: | 0–50 MHz (sampling time = 10 ms) |
| | 0–10 MHz (sampling time = 100 ms) |
| | 0–1 MHz (sampling time = 1s) |
| | 0–100 kHz (sampling time = 10 s) |
| Input: | 68 Ω or 1 MΩ > 1.5 $V_{p-p}$ |
| Mode 1: | Continuous update, 3-s display between sampling periods |
| Mode 2: | Take one sample and hold |
| Frequency reference: | 20-MHz crystal, 50 ppm |

**Table 1—***This frequency meter has specifications comparable to a general-purpose commercial instrument.*

One PIC pin is needed to read the switch and up to six exclusive switch positions can be read. The same shift register that drives the display drives the two outer switch pins, and the common pin goes to a PIC input pin. Hence, as it multiplexes the display, the firmware also checks for switch closures. The switch cannot be read during a count.

## ANALOG

The signal input is a BNC connector. A 68-Ω resistor can be switched across it as a compromise between 75-Ω and 50-Ω cable termination. The input is AC-coupled.

Originally, I planned to make the input impedance 1 MΩ to be compatible with a standard 10× scope probe. A fast comparator would have given an input sensitivity of 50 $mV_{p-p}$. This plan collapsed when I noticed that the comparator had a specified input bias current of 16 µA.

My back-up plan was to substitute a 74AC14 Schmitt trigger chip for the comparator. This has a high input impedance but requires a 1.5-$V_{p-p}$ input to generate an output. So much for using a scope probe.

## IN A BOX...

I spent more time on the mechanical design than on any other aspect of the meter. A six-digit display needs a wide front panel but no great height, so I used a 1.5″ × 5″ plastic box from Radio Shack. The box has grooves for a front panel and a vertical circuit board half an inch behind the panel.

If I had mounted the display chips on a board in the rear grooves, they wouldn't have reached the front panel, even if I put them in sockets. So, I cut a piece of prototyping board and mounted it on spacers above a support board that fits into the rear grooves.

I soldered a strip of 16 wire-wrap pins to the display board. These pass through the supporting board and mate with a 25-pin single-in-line connector mounted edge-on to the main circuit board. The latter is horizontal and is screwed to the box's molded stand-offs.

The input socket and switches are glued to the support board and poke through holes in the front panel. The Schmitt trigger chip is mounted on the back of the support board. Electrical connections from the support board are made by more wire-wrap pins, which also line up with the 25-pin connector on the circuit board.

I had a transparent red filter panel from another box that was large enough to cover the display chips. It is standard procedure to glue it to a cutout in the front panel, but I used an old optical designer's tactic: if possible, eliminate air spaces between surfaces.

I spread a thin layer of clear silicone rubber over the front of the display chips after they were mounted in the board, before they were soldered. Then, I pressed the red filter on top of the display and squeezed out the air bubbles. I turned the assembly face down on a smooth surface and pressed down on each chip to minimize the thickness of the rubber. If you cover the filter with clear tape during assembly and testing, you will be able to read the display without scratching the filter.

The silicone rubber couples the light from the display into the plastic sheet without reflections, making a brighter, clearer display. There is a tiny, invisible gap between the red plastic and the front panel. Replacing a bad display chip won't be easy!

Despite my careful mechanical design, I drilled the front panel holes 0.1″ too far up and had to redo it. That's why the front panel looks fine, but the back panel has odd ventilation holes in it. Also, I forgot an on/off switch. Fortunately, there was room on the front panel for a slide switch, but it looks awkward.

## POWER TRIP

The unit requires 5 V at approximately 200 mA. A 9-VDC adapter and a three-terminal regulator on a heat sink provide the power. For compatibility with other instruments, I used a 9-VAC transformer and added a rectifier and storage capacitor to the box.

If you're wondering why I didn't use pin 0 of Port C of the PIC, it's because I once inserted my 16C55 backward in a socket. One bond wire blew, but everything else works. I have looked for a good home for that chip ever since.

*Tom Napier is a physicist and engineer who parlayed his design experience into an electronics consulting business. His compulsion to share his knowledge drives him to write magazine articles, but he regrets that he cannot offer individual design assistance. He will start using e-mail once the bugs have been worked out.*

## SOFTWARE

The project firmware is available on the *Circuit Cellar* web site.

## SOURCE

**PIC 16C55 microcontroller**
Microchip Technology, Inc.
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com