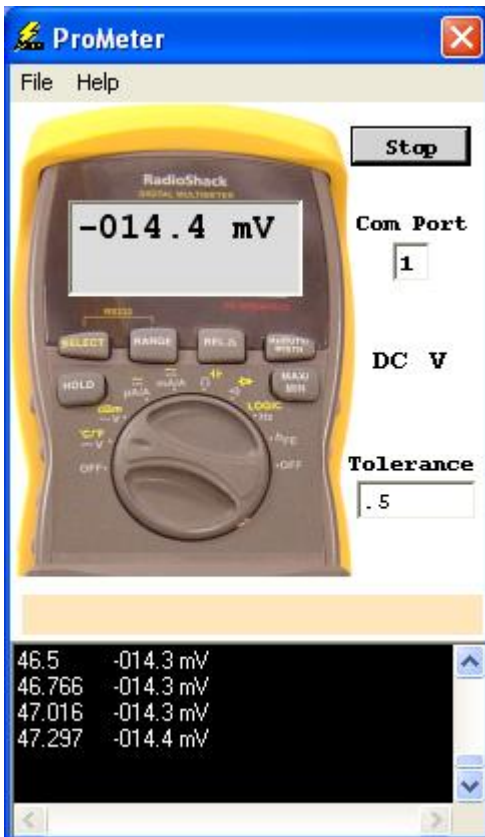


Important

**Requires ZeusPro
V1.70 or later**

Interface your PC to a Portable Digital Multi-Meter
as seen in
June 2006 of Nuts & Volts Magazine

Pick up an issue at
<http://www.nutsvolts.com>



The Radio Shack 22-812 Multi-Meter is one of the most popular meters you will encounter. It features many features found on much more expensive meters, one of which is an RS232 Interface.

When I first saw this meter my geek alarms started ringing. Wow! Just think of what I could do with a digital multi-meter with a RS232 interface. As a natural born geek any time I see something with an RS232 interface my mind starts to wander. My wife says I get this far away look in my eyes. My children have even used the "Yes Dad and it even has an RS232 interface" excuse as a way to get me to purchase something for them.

OK enough wandering lets get back on topic.

While the meter comes with its own software just imagine the possibilities if you could interface the 812 with your own programs. In this article we are going to do just that. I will show you how to connect and communicate with the 812. We will build a couple of reusable functions, and then program our own data logger.

The Interface

The physical connection to the 812 is made through a 9-pin female connection at the top of the meter with the included 9-pin serial cable. The 812 communicates at 4800 baud, 8 bits and no parity. Once connected to the meter, your program needs to raise DTR. The meter will then start to transmit data. One more thing you must turn on the interface by hitting the Select and Range buttons at the same time as shown in Figure 2.



Figure 2

To interface with the meter we will use a programming language called Zeus. Zeus is a simple Windows programming environment that specializes in interface design. You can pick up a copy of ZeusPro from the KRMicros Web site:

<http://www.krmicros.com/Development/ZeusPro/ZeusPro.htm>

The compiled programs are available as well along with the source code.

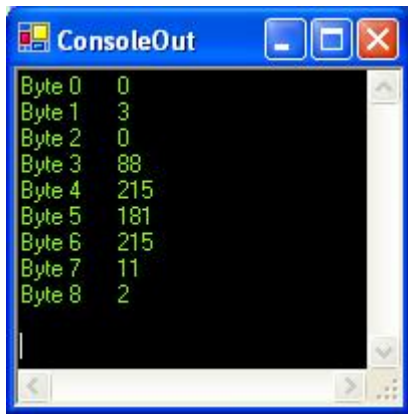


Figure 3

The meter data is transmitted in 9 byte chunks we will call a packet. I have included a program called RS812_Program1.txt. This program collects the 9 bytes of data and displays them on the console as shown in Figure 3.

The heart of RS812_Program1 is a function called GetPacket. This function collects the 9 bytes of data and places them in an array variable called Packets. Once collected we can manipulate or display the data.

Table 1 shows a complete breakdown of the 9 bytes of data.

Byte	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	Mode							
1	Hz	Ohms	K	M	F	A	V	m
2	u	n	dBm	S	%	hFE	REL	MIN
3	4D	4C	4G	4B	DP3	4E	4F	4A
4	3D	3C	3G	3B	DP2	3E	3F	3A
5	2D	2C	2G	2B	DP1	2E	2F	2A
6	1D	1C	1G	1B	MAX	1E	1F	1A
7	Beep	Diode	Bat	Hold	-	~	RS232	Auto
8	Checksum							

Table 1

The first byte tells you what mode or function the meter is in. Table 2 shows a list of these modes.

```

RS812_Program1.txt
func main()
  dim x as integer

ComOpen(1,baud=4800,port=1,parity=0,stop=1,
bits=8)
  ComBGSuspend(1,0)

Loop:
  GetPackets()
  ConsoleCls()
  for x = 0 to 8
    print "Byte "+x,Packets(x)
  next
  print
  Goto Loop
endfunc

-----
'Get RS812 Meter Packets
-----

func GetPackets()
  global Packets(10) as integer
  dim x as integer
  dim tcount as integer
  dim z as integer
  dim CSum as integer
  dim tcounts as integer
  ComPurge(1) : Pause(100) : ComDTR(1,1)

loop:
  CSum = 0
  for x = 0 to 8
    z = ComWaitForByte(1,500)
    if z = -1 then goto ProcErrors
    if x < 8 then
      CSum = CSum + z
    endif
    packets(x) = z
  next

  CSum = (CSum + 57 ) & 255
  if CSum <> Packets(8) or Packets(0) > 25
then goto ProcErrors
  ComDTR(1,0) : Pause(100) : ComPurge(1)
  exit(1)

ProcErrors:
  print "Data Error"
  ComDTR(1,0) : Pause(100) : ComPurge(1)
  ComPurge(1) : Pause(100) : ComDTR(1,1)
  tcounts= tcounts + 1
  if tcounts > 3 then exit(0)
  goto loop
endfunc

```

RS812_Program1.txt

The actual data value is contained in bytes 3-6. These bytes represent the actual digit displayed. For example, byte 6 is digit 1, byte 5 is digit 2, byte 4 is digit 3 and byte 3 is digit 4. The actual bits map to the segments shown in Figure 4.

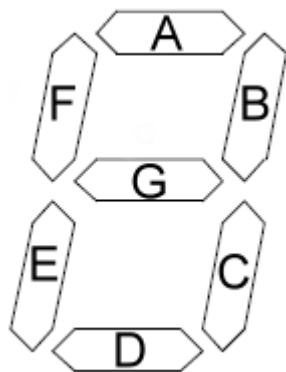


Figure 4

To breakdown the data take a look at RS812_Program2.txt. In this program we collect the 9 bytes of data then jump to one of the 17 handler functions based on the mode value (Byte 0). The 17 handler functions are shown in Table 3.

doWIDTH
doDUTY
doTEMP
doDBm
doCAPS
doDIODE
doLOGIC
doHFE
doFREQ
doOHMS
doCONT
doDCmA
doACmA
doDCuA
doACuA
doACVolts
doDCVolts

Table 3

Once the data is collected and converted the value is displayed in the Console as shown in Figure 5.

Mode	Function
0	DC V
1	AC V
2	DC uA
3	DC mA
4	DC A
5	AC uA
6	AC mA
7	AC A
8	OHM
9	CAP
10	HZ
11	NET HZ
12	AMP HZ
13	DUTY
14	NET DUTY
15	AMP DUTY
16	WIDTH
17	NET WIDTH
18	AMP WIDTH
19	DIODE
20	CONT
21	HFE
22	LOGIC
23	DBM
24	EF
25	TEMP

Table 2

The functionality of each of these handler functions is pretty much the same. I broke them down so that you could easily change the way each piece of data is handled which will make it easy customization.

The function GetDigits does all the real work. It builds the reading value based on the segment bits in bytes 3-6.



Figure 5

Turn it Up

By this point you should have a basic understanding of the protocol or at the very least you have run the sample programs. Now its time for us to turn up the volume. Lets create a basic form interface that will allow us to display the information a bit more efficiently.

The Program RS8112_Program3.txt will display the form shown in Figure 6.

The program is simple to use.

Step 1

Connect the meter to your PC.

Step 2

Turn the meter.

Step 3

Turn on the RS232 interface.

Step 4

Enter the Com Port into the Com Port Field

Step 5

Hit the Start/Stop button.

The only difference between Program 2 and Program 3 is that the display information is sent to a few FormLabels. The program also monitors the FormButton to start and stop the program.

Experiment with the program. You could add a graphic bar to display voltage levels or even a simulated wave form based on the Duty Cycle.

As an example of a more practical application, let's keep track of measurement changes as shown in Figure 7. In Program RS812_Program4.txt we added a few lines of code to the DoReadings function.

```
reading = res
if abs(abs(reading) - abs(lastreading)) >= FormTextBox(METER_TOL) then
  print float(int(Getms()-msstart)/1000),res
  lastreading = reading
endif
```

What this code does is keep track of readings and if it varies from the last reading more than the tolerance that you set it will display the number of seconds since you started tests.

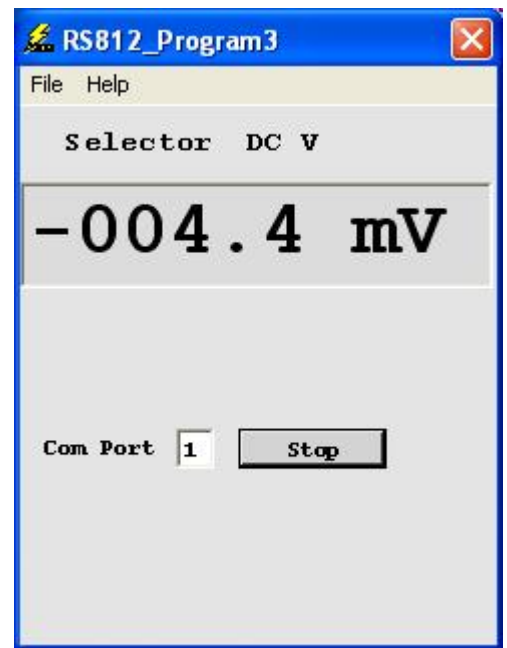


Figure 6

Keep in mind that the console trims itself a bit if more than 20,000 characters are displayed. If you want to keep track of a lot of measurements use the File command to save the measurements to a text file.

Going Further

If you really want to get fancy, start experimenting with line graphics and plot your readings over time. Add File options to save and retrieve the data. Now that you have a working interface its time to let your inner geek out.

Have fun.

As a bonus I have built couple meter applications with ZeusPro. One for the Desktop and one for the Pocket PC. Called ProMeter they use bitmaps to display the meter and will log the readings to a file called MeterLog.txt. They are already built for you and can be found on the KRMicros Website at: <http://www.kronosrobotics.com/Projects/rsmeter.shtml>

You will also find a copy of all the sourcecode here.

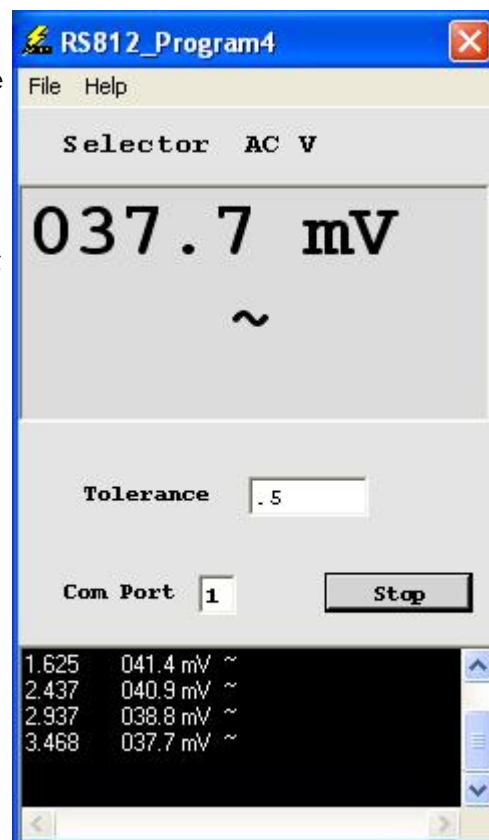


Figure 7