Software for Radio Shack 22-812 Digital Multimeter

Don Peterson (someonesdad1@gmail.com)
Version: 20 Aug 2009
Minor changes: 8 Mar 2010

This package contains the following files:

`readme.pdf`	This file
`rs22812.py`	Python script that can talk to the meter over the serial port and print readings to the console.
`rs22812_linux.py`	A modified script that can be used on Linux. Includes more sophisticated option parsing.
`meter.py`	GUI script using wxPython that shows the current reading on the screen and logs readings to a file.

Needed hardware

You will need a Radio Shack 22-812 digital multimeter (currently selling for around $70). It has a 9 pin female D connector that allows you to connect a serial cable to the meter. The two python scripts can then be used to gather readings from the meter.

You'll also need a serial cable with a 9 pin male connector on one end. The other end will need to fit your computer's serial port. Computers typically don't come with serial ports anymore but you can buy USB to serial adapters for around $15 each.

Needed software

You'll need python, an open source general purpose programming language. You can get it from http://www.python.org. The current version at this writing was 2.6.2, which is what I tested with.

You'll also need PySerial from http://pyserial.sourceforge.net. This is a python module that makes it simple to talk to a serial port.

If you want to use the GUI interface in `meter.py`, you'll need to install wxPython, an open source project that wraps the C++ wxWidgets library in python clothing. You can get wxPython from http://www.wxpython.org. I developed the program with wxPython 2.8.10.1. Since the `meter.py` script uses only simple widgets, the script will likely work with nearly any version of wxPython.

rs22812.py

This script contains a python object that encapsulates the effort needed to talk to the meter. It can be used in one of two ways.

First, you can use it in your own code. When you instantiate the RS22812 object, you pass in the COM port (either a number (on MS Windows) or a device name). Then you call the `GetReading()` method and you'll get a tuple of strings returned. The strings are:

1. Reading
2. Mode

3. Modifiers

The Reading string will be a number followed by a unit. If the `~` character is appended to the unit, it means it was an AC measurement.

The Mode string will indicate the mode of the instrument. See the comments in the code for what these are. The information is largely redundant with what's already given in the Reading string.

The Modifier tuple contains strings that tell more about the meter's state. Some that you'll see are:

`AUTO`	The meter is in auto ranging mode. If not present, the range is fixed.
`REL`	Readings are relative. When the REL button is pressed, the reading is stored and subtracted from subsequent readings. Press the REL button again to turn this mode off.
`MAX`	Shown when the MAX/MIN button is pressed. This displays the maximum reading seen.
`MIN`	Press the MAX/MIN button twice to see the minimum reading.
`HOLD`	The last reading is frozen.

The second way to use the script is as a console data logger. Run the script as `python rs22812.py` and it will print out readings to stdout. For example, here were some samples from a 2 volt peak-to-peak square wave:

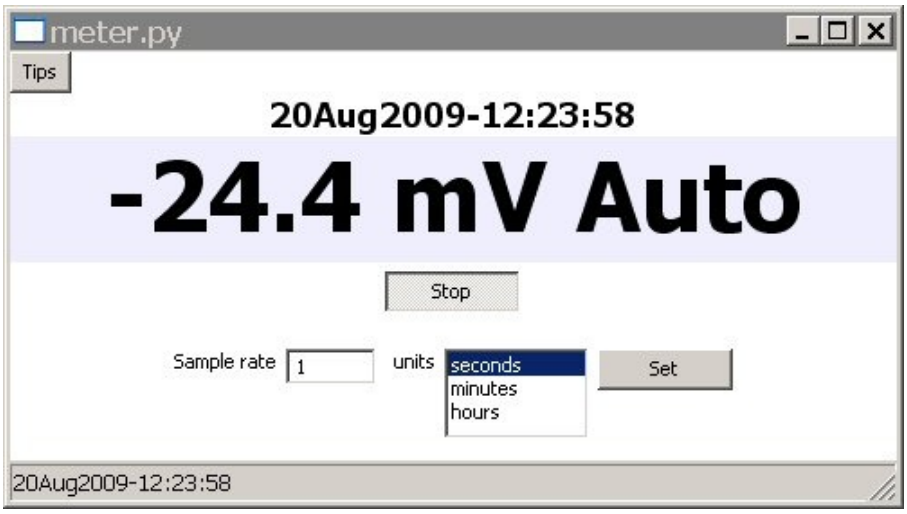
```
19Aug2009-11:28:19 [1] ('-1.008 V', 'DC V', ())
19Aug2009-11:28:20 [2] ('0.998 V', 'DC V', ())
19Aug2009-11:28:22 [3] ('1.007 V', 'DC V', ())
19Aug2009-11:28:24 [4] ('1.007 V', 'DC V', ())
19Aug2009-11:28:26 [5] ('-1.008 V', 'DC V', ())
19Aug2009-11:28:27 [6] ('-1.007 V', 'DC V', ())
19Aug2009-11:28:29 [7] ('-1.007 V', 'DC V', ())
19Aug2009-11:28:31 [8] ('1.007 V', 'DC V', ())
19Aug2009-11:28:32 [9] ('1.007 V', 'DC V', ())
19Aug2009-11:28:34 [10] ('1.007 V', 'DC V', ())
19Aug2009-11:28:36 [11] ('-1.007 V', 'DC V', ())
19Aug2009-11:28:38 [12] ('-1.007 V', 'DC V', ())
19Aug2009-11:28:39 [13] ('-1.007 V', 'DC V', ())
19Aug2009-11:28:41 [14] ('1.007 V', 'DC V', ())
19Aug2009-11:28:43 [15] ('1.007 V', 'DC V', ())
```

The readings are time-stamped, numbered, and you can see the tuple of information that `GetReading()` returns. In this case, the `AUTO` modifier was off because you would otherwise occasionally see 0F on the screen (represents overflow while the meter is autoranging).

You can see that the meter is not quite able to keep up with a 1 second sampling interval.

meter.py

This is a convenience GUI utility that lets you start and stop the readings and set the sampling rate. The readings made are automatically appended to a log file with a time stamp. Here's a screen shot of the program running:



The time of the last reading is shown at the top; next is the reading and any modifier. The `Stop` button shows Start when readings aren't being made and shows Stop when readings are being made (it's a toggle button).

The `Tips` button can be used to turn tips on. When you hold the mouse over an object, a window will pop up after a delay explaining what the object is.

The `Sample rate` box is used to set the sampling rate. You only need to press the `Set` button if you want to change the sampling rate after the `Start` button has been pressed.

You'll want to make sure you change the meter.py script to include the proper serial port designator for your system -- it is unlikely you'll use the same default as I did.

The data read from the meter are logged in a file. Here's a sample of the data:

```
Program started 19Aug2009-11:40:20
  Start button pressed 19Aug2009-11:40:26
  New sampling interval 1 s 19Aug2009-11:40:26
    19Aug2009-11:40:27 [1] ('-0.999 V', 'DC V', ('Auto',))
    19Aug2009-11:40:29 [2] ('-0.999 V', 'DC V', ('Auto',))
    19Aug2009-11:40:30 [3] ('-1.000 V', 'DC V', ('Auto',))
    19Aug2009-11:40:32 [4] ('1.008 V', 'DC V', ('Auto',))
    19Aug2009-11:40:33 [5] ('1.008 V', 'DC V', ('Auto',))
    19Aug2009-11:40:35 [6] ('1.008 V', 'DC V', ('Auto',))
  Stop button pressed 19Aug2009-11:40:35
  New sampling interval 2 s 19Aug2009-11:40:39
  Start button pressed 19Aug2009-11:40:43
    19Aug2009-11:40:43 [1] ('1.008 V', 'DC V', ('Auto',))
    19Aug2009-11:40:46 [2] ('-1.003 V', 'DC V', ('Auto',))
    19Aug2009-11:40:49 [3] ('-0.999 V', 'DC V', ('Auto',))
    19Aug2009-11:40:52 [4] ('1.008 V', 'DC V', ('Auto',))
  Stop button pressed 19Aug2009-11:40:53
  Program ended 19Aug2009-11:40:59
```

The readings are timestamped as well as numbered from when the `Start` button was pressed. You can't see it from this example, but there's a blank line inserted before "`Program started...`" for each invocation of the program. This, along with the indentation, makes it easier to find relevant data.

The `settings` variable at the beginning of the file lets you change various settings in the program.

If you start the program without the digital multimeter turned on, the program will block indefinitely because there is no timeout set for the serial port. The `Stop` button won't work and you'll just have

to kill the program unless you want to turn on the meter.