SPIR-113: ScopeMe Software Manual

Ward Andrew Wurtz

August 28, 2003

Contents

1 Introduction

ScopeMe is software developed to communicate with a Tektronix TDS 210 Digital Oscilloscope with communications module using a Linux machine. It has been used to communicate with the scope though the serial port. The parallel port may work, but has not been tested.

The program was designed for testing the cables and splitters for use with a CAEN 792AA VME QDC [1]. The information for programming the scope can be found in [2].

ScopeMe was programmed in the C programming language.

2 User's Manual

This section describes how to setup and use the various functions of the ScopeMe program. ScopeMe uses the simple philosophy that all options must be set on the scope, and the computer exists only to get the relevant data from the scope.

2.1 Getting started

1. Connect a 9-pin null modem cable to the back of the scope.

2. Connect the other end to the back of the computer that will be communicating with the scope.

3. Ensure that the user has permission to read and write to the device file (e.g /dev/ttyS0) that corresponds to the correct serial port.

4. Turn the computer and the scope on and ensure that the scope serial port is set to the same speed as the computer serial port (likely 9600 baud) by pressing "UTILITY", "Options","RS232 Setup" and "Baud" until "9600" is displayed.

5. Open a command prompt and type in the following command:

   ```
   echo "ID?" > /dev/ttyS0 && cat /dev/ttyS0
   ```

   The screen should display

   ```
   ID TEK/TDS 210,CF:91.1CT,FV:v1.19 TDS2CM:CMV:v1.04
   ```

   or a similar response depending on the particulars of the scope. Press CTRL-C to kill cat. If the above step went as described, the interface with the scope is working and is ready to proceed. If it did not work, something is wrong and the setup needs to be checked.

2.2 Configuring, Compiling and Starting ScopeMe

1. If the scope is using a device other than /dev/ttyS0, you will need to alter the line in src/scopeme.c that reads

   ```
   #define DEV_TTY "/dev/ttyS0"
   ```

   to whatever device is being used (e.g. /dev/ttyS1).

2. Compile ScopeMe by typing

   ```
   make
   ```

   at the command line. ScopeMe should now be compiled. ScopeMe was developed using egcs-2.91.66 for Redhat Linux 6.1 and GNU make 3.77.

3. The following commands are accepted by the makefile:

   - make: compiles ScopeMe.
   - make clean: deletes all compiled files.
   - make tar: creates a tar-ball using the tar command.

4. To start scopeme type

   ```
   ./scopeme
   ```

in the appropriate directory. The following should appear:

```
Scope Me! /dev/ttyS0 is set to 9600 baud
Scopeme by Ward Wurtz
NSERC Summer Student 2003

Will attempt to communicate with scope on /dev/ttyS0
If the program freezes, press CTRL-C and ensure the
scope is properly connected.
scope_query: Opening /dev/ttyS0
scope_query: Writing: ID?
scope_query: Reading from scope...
scope_query: Read: ID TEK/TDS 210,CF:91.1CT,FV:v1.19 TDS2CM:CMV:v1.04
scope_query: Closing /dev/ttyS0
Communication with scope was successful!


Select from the following
1. Measure Cross-Talk or Attenuation
2. Get Waveforms from Scope
3. Get Hardcopy from Scope
4. Measure Length of Cables in ns
a. Send Command to Scope
b. Send Query to Scope
q. Quit
```

If so, ScopeMe is up and running. If the program freezes on the line

```
scope_query: Reading from scope...
```

then it is failing to communicate with the scope.

5. Notice, however, that if the user does not have access to the serial port, ScopeMe will not detect this failure and the main menu will appear. However the line

   ```
   scope_query: Read: ID TEK/TDS 210,CF:91.1CT,FV:v1.19 TDS2CM:CMV:v1.04
   ```

   will not appear, signalling the error. This should be fixed in future versions of ScopeMe.

6. Sections 2.3, 2.4, 2.5 and 2.6 can be read in any order as they describe mutually exclusive operations of ScopeMe.

2.3 Measuring Attenuation

Measuring attenuation and cross-talk is the most specific use of ScopeMe and the primary reason ScopeMe was developed. It is also the least useful for general-purpose work and a user may with to remove this option from the main menu in scopeme.c if it is of no practical use.

This option is used for measuring cross-talk and attenuation in the cable adapters and splitters from the CAEN V792AA QDC. See reference [1] for details on why one would want to use this function.

1. Setup the scope as described in [1] with the original signal coming to channel 2 and the signal after the test cable going to channel 1.

2. Setup the scope options as desired. There must be one complete waveform on the screen so that the scope can measure the frequency of the wave.

3. Select option

   ```
   1. Measure Cross-Talk or Attenuation
   ```

   from the ScopeMe main menu.

4. Enter the filename where you would like to store the data. This option is non-destructive and the data will be appended to the end of the given file.

5. The message

   ```
   Set input settings as desired and press ENTER to begin
   data acquisition for the first cable.
   ```

   should appear on the screen.

6. After the scope is setup, press the ENTER key.

7. ScopMe will display the message

   ```
   Getting data from the scope (this may take a while):
   ```

   on the screen and the program will take a moment to get the data. It performs some averaging to improve the quality of the data. One may desire to set averaging on the scope using "Acquire" and "Average", making sure to select the desired number of averages.

8. The scope should output the measured frequency observed on channel 2 with the RMS voltages from channels 1 and 2.

   ```
   Frequency on Channel 2: 25.002090 MHz (avg 25.002090 MHz)
   RMS Voltage on Channel 1: 0.001074 V (avg 0.023912 V)
   RMS Voltage on Channel 2: 1.424930 V (avg 1.424930 V)
   Press ENTER for next cable or 'q' then ENTER for a new setting:
   ```

9. Pressing ENTER will repeat the previous measurement. It is expected that a different cable be used for this second measurement. ScopeMe will compute the sample standard deviation between different cables as a measure of how different cables vary in their cross-talk and attenuation.

10. Pressing 'q' then ENTER will save the information to the file and conclude the measurements for this setting.

11. After pressing ’q’ then ENTER, the results will be displayed as they are in the file starting with the frequency on channel 2, it’s sample standard deviation, the voltage on channel 1 with it’s sample standard deviation and the voltage on channel 2 with its sample standard deviation.

12. Pressing ENTER will allow the program to continue measuring for another data-point. Pressing ’q’ then ENTER will return to the main menu.

13. If you continue, the message

    Set input settings as desired and press ENTER to begin
    data acquisition for the first cable.

    will appear again and the CH2 voltage or frequency can be adjusted for the next set of measurements.

2.4 Getting Waveforms

This section describes how to get a waveform from the scope. The output consists of three columns of data. The first column is the time axis which is zeroed on the trigger event. The second column is the voltage on channel 1 and the third column in the voltage on channel 2. Each data set contains 2500 data points.

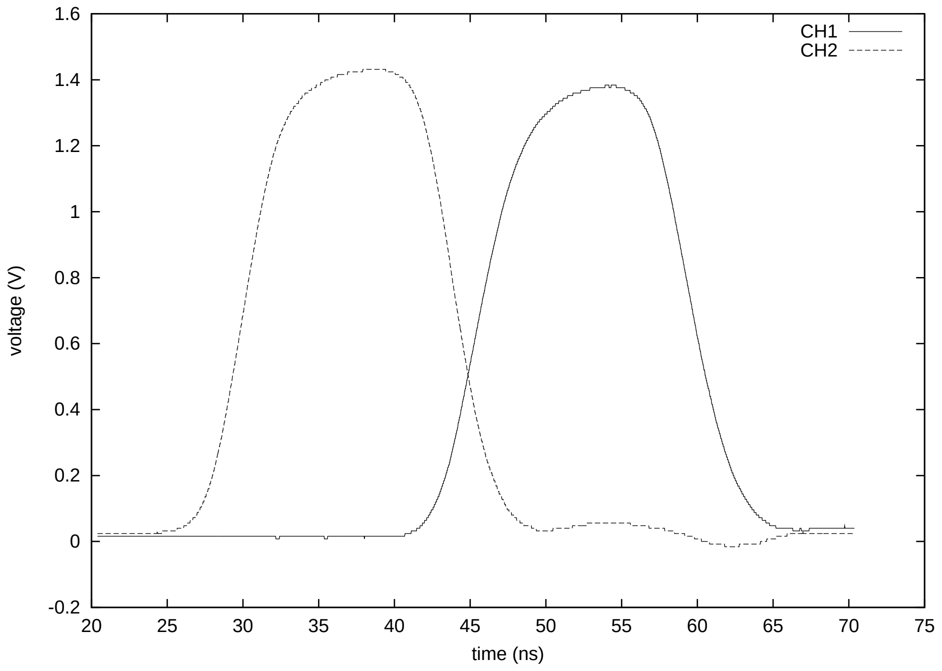

Figure 1: Waveform obtained by ScopeMe and plotted by GNUPlot

1. Setup the scope exactly how you want it. Remember that only data on the display will be downloaded.

2. Select option

2. Get Waveforms from Scope

from the ScopeMe main menu.

3. Enter the filename where you would like to store the data. For example, use "output.txt".

4. After the filename has been entered, the screen will display

   Getting image from scope...

   and you will have to wait a short time while the scope downloads the data.

5. ScopeMe should return to the main menu when completed. The data can now be graphed using GNUPlot or used with whatever software is desired.

2.5 Getting Hardcopies

This section describes how to get a hardcopy from the scope. A hardcopy is simply a picture of the scope's display. It is available in many formats, but ScopeMe supports only the encapsulated postscript format.

It is up to the user to set the options on the oscilloscope. The software does not set any options. It merely acts as a conduit to get the information from the scope into a file.

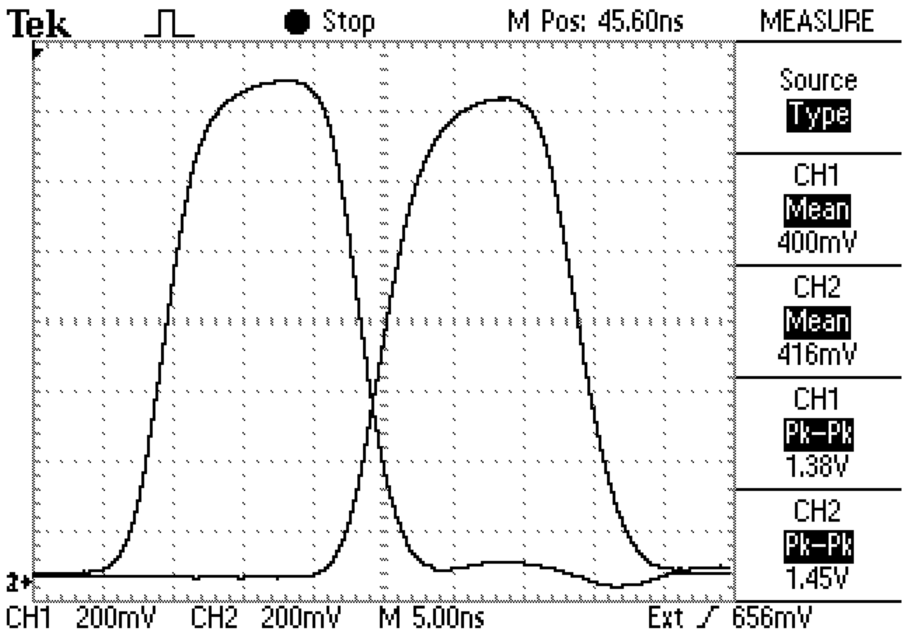

Figure 2: Hard copy obtained by ScopeMe

1. Press "Utility", "Options" and "Hard Copy Setup" on the scope. Set "Format" to "EPSIMAGE" and "Port" to "RS232".

2. Setup the scope exactly how you want it. Remember, a hardcopy takes a picture of the entire display, including the side panel. Make sure it is displaying what you want it to display.

3. Select option

   3. Get Hardcopy from Scope

   from the ScopeMe main menu.

4. Enter the filename to store the data when prompted. For our example, it should be something like "output.eps".

5. After the filename has been entered, the screen will display

   Getting hard copy... (this may take about 1 min at 9600 baud)

   and you will be forced to wait while the scope downloads the data.

6. ScopeMe should return to the main menu when completed. The postscript image may be viewed using ghostview or another viewing program.

2.6 Measuring the Propagation Time of a Cable

It may be desirable to know how long a cable is. One way to measure it is to measure it's physical length in meters. Another way is to measure how long it takes a signal to propagate down the cable. That is the kind of measurement that scopeme does. See [1] for details on testing results.

The following algorithm is used by ScopeMe to compute the time it takes for a pulse to travel down the test cable and probe.

1. Get the waveform data from the scope.

2. Compute the average value of each pulse. Recall that a pulse is attenuated by the cable so the average value on channel 1 will be less than channel 2.

3. Find t_1 and t_2 which are defined as the first time the pulse reaches its average voltage for channels 1 and 2 respectively.

4. Compute $|t_2 - t_1|$ and use this as the amount of time it took the pulse to travel down the cable.

To use this function:

1. Select

   4. Measure Length of Cables in ns

   from the main menu.

2. The filename to store output, the assembly number and the cable number must be input into ScopeMe. After each cable, ScopeMe will automatically increment the cable number by one until it reaches 15. After 15, the cable number will zero and ScopeMe will increase the assembly number by one.

3. ScopeMe will display

   ```
   Set input settings as desired and press ENTER to begin
   data acquisition for current cable.
   ```

   on the screen. Do as it instructs.

4. ScopeMe will display

   ```
   Getting data from scope...
   ```

   and will pause for a few seconds until data acquisition has completed.

5. After it has obtained the data, ScopeMe will display

   ```
   Getting data from scope...done
   Computing cable length...done
   cable 0 of assembly 0:
   The length is: 13.900000 ns
   Press ENTER to repeat for the next cable or ’q’ then ENTER to quit.
   ```

   and wait for instructions. Pressing ENTER will cause ScopeMe to loop to step 3. Pressing ’q’ then ENTER will cause ScopeMe to return to the main menu.

6. ScopeMe writes the measurements to the disk using three columns. The three columns are the assembly number, the cable number and the time computed in the previous step.

3 Programmer’s Manual

3.1 Files

Scopeme consists of the following files:

- Makefile: Stores information for GNU Make. See section 2.2.
- scopeme: The binary. See section 2.2.
- scope_driver.c and scope_driver.h: Functions that communicate directly with /dev/ttyS0 (or /dev/ttyS1, etc.) where the scope is connected. See section 3.2.
- scope_fns.c and scope_fns.h: Functions that rely on scope_driver.c for communication with the scope and perform some data processing but do not communicate with the keyboard or print to the screen under normal operation. These functions perform useful and generic operations. See section 3.3.
- scopeme.c: Provides high-level operations and communication between the ScopeMe program and the user. See section 3.4.
- latex/scopeme.tex: this document.

Each function is documented in the corresponding ".h" file. This document explains the basic concepts while the ".h" file explains how to call the function and what its return values are.

3.2 Low-Level Functions

The low level functions are stored in scope_driver.c and their descriptions are in scope_driver.h. These functions read and write to /dev/ttyS0, /dev/ttyS1, etc. to communicate with the scope. They will communicate at a speed specified by calling scope_speed(), which should match the speed on the oscilloscope. See section 2.2 for details on setting the speed.

The following functions are defined in scope_driver.h:

1. scope_speed(): Sets the baud rate for the serial port.
2. scope_command(): Sends a command to the scope and expects no reply.
3. scope_query(): Sends a command to the scope and blocks until the scope responds. If ScopeMe hangs, it is probably because scope_query() has blocked but the scope did not receive a proper query command so no value is being returned.
4. scope_query_noblock(): Same as scope_query() but waits a short period of time then tries to read without blocking. Thus, it will not cause the program to hang but it may assume the scope is done sending information before it is actually done.
5. scope_measure(): Gets an immediate measurement from the scope in double format.
6. scope_hardcopy(): Gets a postscript hardcopy from the scope. See section 2.5.

Each time a driver is called, it opens and closes the device (say /dev/ttyS0). This has some unfortunate side effects. Some scope settings disappear after the device is closed. Also, sometimes, after transferring a large piece of data, the next connection made to the device will not work properly. For that reason, a function has been added to scope_fns.c to make sure that the scope is accepting data before an important command is sent. scope_driver.c should be re-written to keep the device open during the entire communication with the scope.

3.3 Medium-Level Functions

The medium level functions are stored in scope_fns.c and their descriptions are in scope_fns.h. These functions rely on scope_driver.c to communicate with the scope and provide some computation based on the results of the scope queries. However, they do not interact with the user.

The following functions are defined in scope_fns.h:

1. scope_datapoints(): gets the data-points used by 2.4 and 2.6 in double format. All units are SI.
2. scope_amplitude_measure(): gets the measurement information used by section 2.3. Returns the frequency on channel 2 and the voltages from channel 1 and channel 2. All units are SI.
3. scope_wait(): waits until the scope is not busy. Used to prevent lockups where a query is waiting for a response but the scope was not ready and did not receive the query.

3.4 High-Level Functions

The high level functions of ScopeMe are contained in the file scopeme.c. These functions call the functions contained in scope_driver.c and scope_fns.c with options from the user and make the output of these functions useful. The high level functions are controlled by a menu driven system that interacts with stdin and stdout. See the scopeme.c source for more details.

References

[1] Jennifer Robb and Ward Wurtz. *SPIR-112 Cross-talk and Attenuation Tests for Cables and Splitters Used with CAEN 792AA VME QDC*. Internal Document, Summer 2003.

[2] Tektronix. *Programmer Manual: TDS 200-Series Digital Real-Time Oscilloscope*. Tektronix part number 071-0493-01.