plug & play instruments
oscilloscopes

Cleverscope Ltd
Phone +64 9 524 7456
Fax +64 9 524 7457
Email support@cleverscope.com
28 Ranfurly Rd, Epsom
P.O. Box 26-527
Auckland 1003
New Zealand

V1.1
30 April 07

Application example

Using the Cleverscope DLL with Labwindows CVI

Summary

This application example uses the Labwindows C code environment to produce a working executable which has simple controls for the Signal Generator, and Scope graph controls.

The application may be used as an example of interfacing to the DLL using C code.

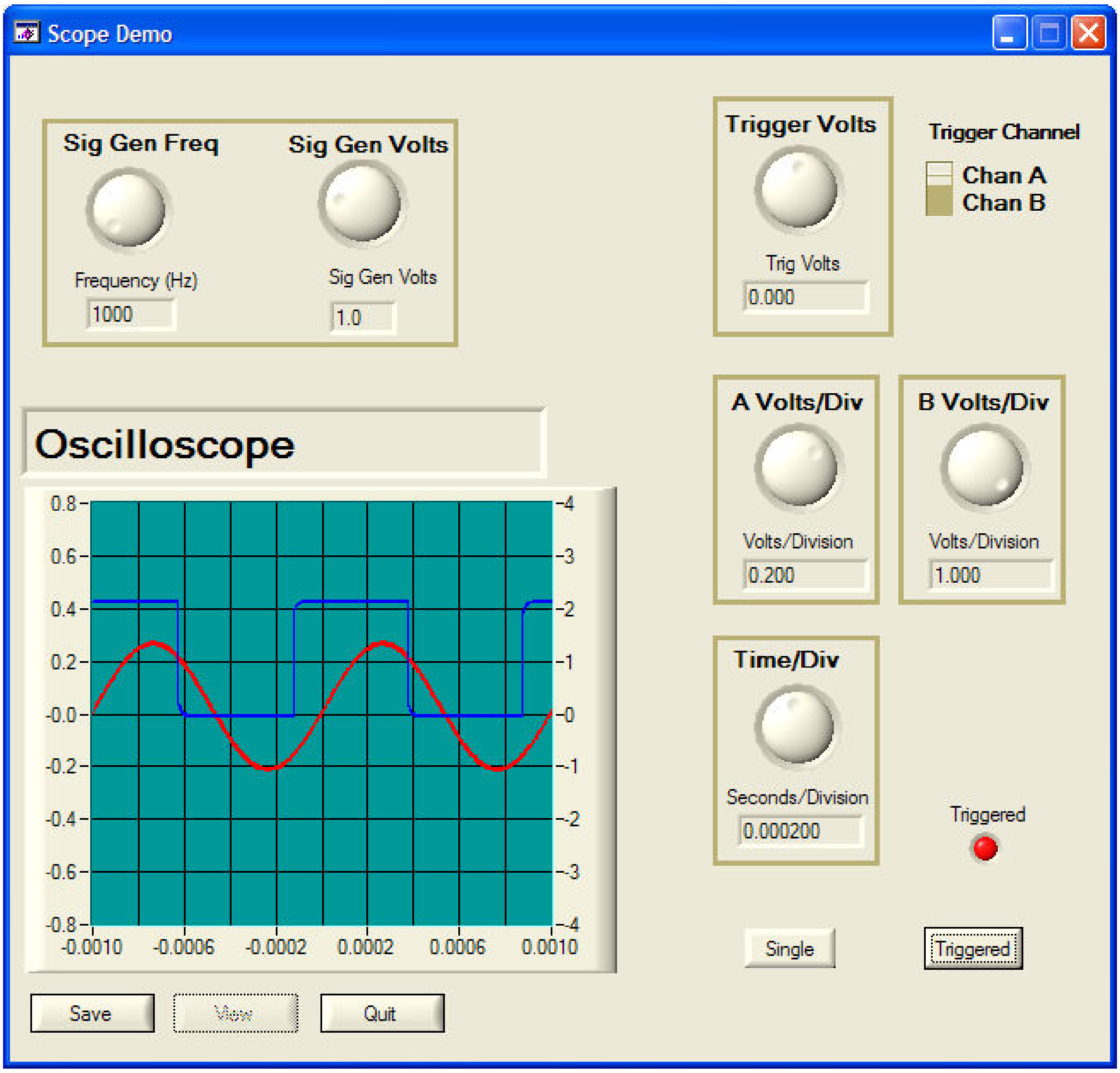
Users who do not have Labwindows can still benefit from the example, as the interface portion will be directly transferable to Borland C++ Builder or Microsoft Visual C. We will in time produce examples for these environments.

Important Information

1. To use the application you will have to have the National InstrumentsLV7.1 Run Time library installed. This happens automatically when you install Cleverscope, so make Cleverscope has been installed on the target machine.
2. The DLL uses the STD CALL method of parameter passing. Ensure your environment is setup this way.
3. Connect a signal source to the Chan A input (such as the Sig Gen Output) so the application has something to trigger off.

The demo application

The ScopeDemo.exe demonstration application has the following front panel:

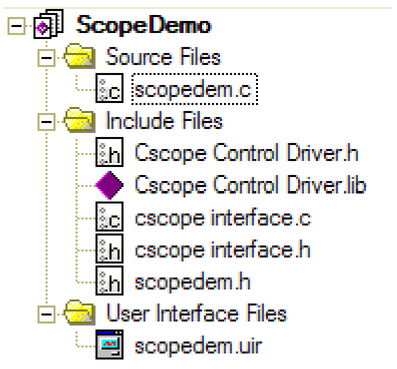


You can see there are controls to set the time base, the A and B channel gains, the trigger channel and amplitude, and the signal generator frequency and amplitude. As such it represents a simple example for use of the major features of the Cleverscope Acquisition Unit.

Note that the application connects immediately and makes one single acquisition to refresh the screen at start up – so connect a signal to the acquisition unit, and have it on, and connected before running the application.

Application Structure

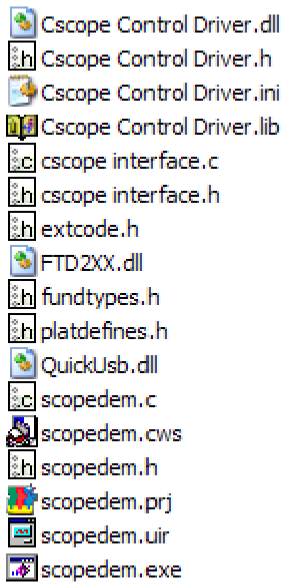
Here is the application structure:
To debug the application you will need Labwindows CVI – the National Instruments C development environment.



```
ScopeDemo
  Source Files
    scopedem.c
  Include Files
    Cscope Control Driver.h
    Cscope Control Driver.lib
    cscope interface.c
    cscope interface.h
    scopedem.h
  User Interface Files
    scopedem.uir
```

The application **scopedem.c** communicates with the Cleverscope interface file **cscope interface.c** using the **cscope interface.h** header file. **Cscope interface.c** communicates with the acquisition unit hardware via USB using the **cscope driver.dll** file, which is accessed using the **cscope driver.h** header file. You will also need the **ftd2xx.dll** and quickusb.dll files for USB support, and you may need other header files or the cscope.lib file to build an application.

The full directory of files is shown here:



```
Cscope Control Driver.dll
Cscope Control Driver.h
Cscope Control Driver.ini
Cscope Control Driver.lib
cscope interface.c
cscope interface.h
extcode.h
FTD2XX.dll
fundtypes.h
platdefines.h
QuickUsb.dll
scopedem.c
scopedem.cws
scopedem.h
scopedem.prj
scopedem.uir
scopedem.exe
```

- Cscope Control driver.dll – contains the executable code to control the Cleverscope Acquisition Unit
- Cscope Control driver.h – contains a header file with the C code definitions required to call the DLL. The definitions are based on the STD CALL calling protocol.
- Cscope Control Driver.ini – this file is not used, but present for future possibilities.
- Cscope Control driver.lib – a linker file used by Labwindows CVI
- Cscope interface.c – the C code that is used to talk to the DLL in the example application
- Cscope Interface.h – a header file that allows the main application to call functions in **cscope interface.c**.
- Extcode.h, fundtypes.h, platdefines.h –header files required to explain types used by the Labwindows system if you are using an alternative compiler.
- FTD2XX.dll – a DLL called by cscope control driver.dll to communicate via the USB 1.1 link with the Cleverscope Acquisition unit.
- QuickUSB.dll – a DLL called by cscope control driver.dll to communicate via the USB 2.0 link with the Cleverscope Acquisition unit.
- Scopedem.c – the example application C code source.
- Scopedem.cws, scopedem.prj –Labwindows project definition and build files.
- Scopedem.exe – an executable to show the output of the project.
- Scopedem.h – an automatically generated header file that allows scopedem.c to receive events and call backs from the user interface.
- Scopedem.uir – the Labwindows user interface definition.

Key Functions in cscope interface

These are the key functions:

```
int scope_init (void);
```
This function initialises the acquisition unit. It returns 0 if no error, a positive integer otherwise.

```
int scope_close (void);
```
This function closes the acquisition unit. It returns 0 if no error, a positive integer otherwise.

```
int scope_config (double a_div, double b_div, double t_div, int
number_of_points, double freq, double sigvolts, double trigvolts, int
trig_chan);
```
This function configures the acquisition unit. It returns 0 if no error, a positive integer otherwise.

```
int   scope_acquire (void);
```
This function starts an acquisition based on the acquire variable setup. It returns 0 if no error, a positive integer otherwise.

```
int scope_read_waveform (double a_waveform[max_samples], double
b_waveform[max_samples], int *num_samples,  double *delta_t, double *t_zero);
```
This function returns the last transferred waveforms for Channels A and B, and sampling details. It returns 0 if no error, a positive integer otherwise.

```
int check_for_samples(void);
```
This function returns 1 if a trigger has occurred and samples have been returned, otherwise 0.

Key variables in scope interface.c

The **acquire** variable holds the current acquisition configuration. The function **cscope config** simply manipulates this variable.

Here is the default setup for acquire:

```
acquire.AcquireMode = 0;                //automatic
acquire.AcquisitionMode = 1;            //peak captured
acquire.Acquirer =  4;                  //cleverscope is the acquirer
acquire.TransferChans = 2;              //transfer both channels
acquire.AMaxScale = 2;                  // Volts range = +/-2
acquire.AMinScale = -2;
acquire.BMaxScale = 2;
acquire.BMinScale = -2;
acquire.AProbe = 0;                     //x1
acquire.BProbe = 0;                     //x1
acquire.ACoupling = 1;                  //DC
acquire.BCoupling = 1;                  //DC
acquire.ABandwidth = 1;                 //100 MHz
acquire.BBandwidth = 1;                 //100 MHz
acquire.TriggerSource = 0;              //A Chan trigger
acquire.TriggerAmplitude = 0;           //Trigger at zero volts
acquire.ATriggerAmplitude = 0;
acquire.BTriggerAmplitude = 0;
acquire.TriggerFilter = 0;              //No trigger filter
acquire.TrigSlope = 0;                  //rising
acquire.TriggerHoldoff = 0;
acquire.DigPatternRqd = 0;              //not used
acquire.DigPattern =  0;                //not used
acquire.ExtTrigThreshold = 0;
acquire.DigInputThreshold = 2;
acquire.StartTime =  -0.005;            //-5 msecs
acquire.StopTime =   0.005;             //5 msecs
acquire.PreTrigTime = 0.005;
acquire.Port = 0;
acquire.NumDivisions = 10;
acquire.NumSeqFrames = 1;
acquire.NumBuffers = 2;
acquire.SigGenFreq = 1000;              //1kHz output
acquire.SigGenAmp = 1;                  //1V amplitude
acquire.SigGenOffset = 0;
acquire.SigGenWaveform = 0;             //sine
acquire.SigGenSweep = 0;
acquire.SigGenFunc = 0;
acquire.SigGenFreq2 = 0;
acquire.SigGenPhase = 0;
acquire.Trig2Function = 0;              //not used
acquire.MinTriggerPeriod = 0.0000001;
acquire.MaxTriggerPeriod = 1;
acquire.TriggerCount = 1;
acquire.Trig2Slope = 0;
acquire.Trig2SourceChan = 0;
acquire.Trig2Level = 0;
acquire.DigPattern2Rqd = 0;
acquire.DigPattern2 = 0;
acquire.Trigger2Source = 0;
acquire.WaveformAverages = 1;
acquire.ValueChanged = 1;
samples_required = 1000;
SamplerResolution = 0;                  //10 bit sampler
```

See the Cscope driver DLL documentation for the full definition of the acquire variable.