SoftScope SDK 1.1

User's Guide

Tel : +82-2 470-0491 Fax: +82-2 474-0659

Address : *soft*DSP Co., Ltd.

Jungil Bldg 203, 552-1 Sungnae-dong, Kangdong-ku, Seoul, Korea

Home page: http://www.softdsp.com

CONTENTS

1. Overview

1.1. Introduction

SoftScope Software Development Kit (SDK) consists of various library functions, which control SDS 200/SDS 200A, together with examples. It provides 46 API functions for version 1.1. The examples are written in Visual C++ 6.0, C++ Builder 6.0 and Visual Basic 6.0.
Afterwards, the examples for other programming platforms will be provided as well.

1.2. Files provided

SoftScope SDK 1.1 includes the following contents.

1.2.1. Lib / library files and header files

SoftScope11.dll	DLL library (version 1.1)
SoftScope11.lib	library file for Visual C++
SoftScope11builder.lib	library file for C++ Builder

1.2.2. Include / Collection of library header files

SoftScopeAPI.h	API function definition for C++
SoftScope_type.h	Type definition for C++
SoftScopeAPI.bas	API function definition for Visual Basic
SoftScope_type.bas	Type definition for Visual Basic

1.2.3. Example / Example programs

C++ Builder6	Test software for C++ Builder 6
Visual C++ 6	Test software for Visual C++ 6
Visual Basic 6	Test software for Visual Basic 6

1.2.4. Exe / Executable files

BCTest.exe	Executable files compiled by C++ Builder
VCTest.exe	Executable files compiled by Visual C++
VBTest.exe	Executable files compiled by Visual Basic

1.2.5. Doc / Documents

SoftScopeSDK 1.1 Users Guide.pdf	User’s Guide for SDK 1.1
SoftScopeSDK 1.1 Function Reference Guide.pdf	Function Reference Guide for SDK 1.1

2. Programming Method

2.1. Data Type

2.1.1. Typedef & Global Const

2.1.1.1.C++ Typedef

The arguments and return values, which are used by API functions of library, are all defined by enumeration type in SoftScope_type.h. Some examples are listed below.

```
// Typedef Section
typedef short RAW_DATA;
typedef double PHYSICAL_DATA;

typedef enum TIME_DIV {
      // RIS Time mode
      _200NS_DIV = 6, _400NS_DIV = 7,
      _1US_DIV = 8, _2US_DIV = 9, _4US_DIV = 10,
       …
      _2S_DIV = 27, _4S_DIV = 28, _10S_DIV = 29,
} TIME_DIV;
typedef enum VOLTAGE_DIV {
// RELAY 0,1 OFF
      _10MV_DIV = 1, _20MV_DIV = 2, _50MV_DIV = 3,
      // RELAY 0 ON, 1 OFF
      _100MV_DIV = 4,_200MV_DIV = 5, _500MV_DIV = 6, _1V_DIV = 7,
      // RELAY 0,1 ON
      _2V_DIV = 8, _5V_DIV = 9, _10V_DIV = 10
} VOLT_DIV;
```

2.1.1.2.Visual Basic Global Const

For Visual Basic, the values corresponding to SoftScope_type.bas are defined.

```
Global Const S_200NS_DIV As Long = 6
Global Const S_400NS_DIV As Long = 7
Global Const S_1US_DIV As Long = 8
…
Global Const S_20US_DIV As Long = 12
Global Const S_40US_DIV As Long = 13
Global Const S_100US_DIV As Long = 14
```

2.1.2. Raw data and Physical data

The data acquired from SDS 200 / SDS 200A will have the followings.

- **RAW_DATA**
  A/D converted value which is two-byte (16 bits) short data type.
  In the case of SDS 200, the value is between 0 and 4095.
  In the case of SDS 200A, the value is between 0 and 1023.
  If RAW_DATA holds the value of 4096 (for SDS 200) or 1024 (for SDS 200A), these imply the special value as RIS_MISSING.
- **PHYSICAL_DATA**
  It is the value that RAW_DATA is applied to current Volt/div and calibration data and converted it to voltage value. It comprises eight-byte (64 bits) double data type.

2.2. Default Assumption

It is assumed that the screen is composed of 500x400 on pixel basis.
For X axis, it is composed of 500 scale units and divided into 10 divisions. One div is made of 50 scale units. For Y axis, it comprises 400 scale units and divided into 8 divisions. One div is composed of 50 scale units.
For X axis the unit can be time or pixel. For Y axis, the unit can be voltage or pixel.

2.3. API Functions

The functions provided by the library all starts with sd_, and there are 46 API functions in total for Version 1.1.

2.3.1. Initialization/Finalization

For SDS 200 / SDS 200A, proper initialization is required for smooth operation.
This includes USB communication initialization, internal buffer generation, calibration data reading and initialization of internal parameters. The function for initialization is as follows.

sdInitialize()

When every processing is over, SDS 200 / SDS 200A should be finalized.
The function for finalization is as follows.

sdFinalize()

2.3.2. Control functions

In order to control SDS 200 / SDS 200A, various parameters should be set.
The functions for this are shown below.

2.3.2.1. Channel Control

sdSetChannelOnOff()	To turn on and off the related channel
sdSetChannelCoupling()	To set the coupling attribute of the related channel
sdSetBandwidthLimitOnOff()	To turn on and off Bandwidth Limit of the related channel

2.3.2.2. Voltage Control

sdSetChannelVoltage()	To set Voltage/div value of the related channel
sdSetChannelOffset()	To set Offset value of the related channel on pixel basis
sdSetChannelOffsetVoltage()	To set Offset value of the related channel on voltage basis

2.3.2.3. Horizontal-axis Control

sdSetTimeDiv()	To set current Time/div value
sdSetDelayOnOff()	To set current Delay status
sdSetDelayOffset()	To set Delay value on pixel basis
sdSetDelayOffsetTime()	To set Delay value on time basis

2.3.2.4. Trigger Control

sdSetTriggerSource()	To set current trigger source
sdSetTriggerOffset()	To set current trigger offset on pixel basis
sdSetTriggerOffsetVoltage()	To set current trigger offset on voltage basis
sdSetTriggerMode()	To set current trigger mode
sdSetTriggerSlope()	To set current trigger slope
sdSetTriggerHighFrequencyRejectOnOff()	To turn on and off High Frequency Reject

2.3.2.5. Other Control

sdSetETSOnOff()	To set the activation of ETS mode
sdSetPeakDetectOnOff ()	To turn on and off Peak Detect
sdSetBufferSize()	To set mode of buffer size

2.3.3. Acquisition of parameters

The purpose is to acquire the parameters to control current SDS 200 / SDS 200A.

The functions for this are described below.

2.3.3.1. Acquisition of buffer size

sdGetDataBufferSize()	To get current buffer size

2.3.3.2. Acquisition of channel parameters

sdGetChannelOnOff()	To get On/Off status of the related channel
sdGetChannelCoupling()	To get coupling status of the related channel
sdGetBandwidthLimitOnOff()	To get On/Off status of Bandwidth Limit of the related channel

2.3.3.3. Acquisition of voltage parameters

sdGetChannelVoltage()	To get Volt/div value of the related channel
sdGetChannelOffset()	To get Offset value of the related channel on pixel basis
sdGetChannelOffsetVoltage()	To get Offset value of the related channel on voltage basis

2.3.3.4. Acquisition of horizontal-axis parameters

sdGetTimeDiv()	To get current Time/div value
sdGetDelayOnOff()	To get current Delay status
sdGetDelayOffset()	To get Delay value on pixel basis
sdGetDelayOffsetTime()	To get Delay value on time basis

2.3.3.5. Acquisition of trigger parameters

sdGetTriggerSource()	To get current trigger source
sdGetTriggerOffset()	To get current trigger Offset on pixel basis
sdGetTriggerOffsetVoltage()	To get current trigger Offset on voltage basis
sdGetTriggerMode()	To get current trigger mode status
sdGetTriggerSlope()	To get current trigger slope status
sdGetTriggerHighFrequencyRejectOnOff()	To get On/Off status of High Frequency Reject

2.3.3.6. Acquisition of other parameters

sdGetETSOnOff()	To check if ETS mode is activated
sdSetPeakDetectOnOff ()	To turn on and off Peak Detect

2.3.3.7. Handling errors

sdGetErrorMessages()	To get error messages if an error occurs

2.3.3.8. Acquisition and conversion of data

Using the adjusted parameters, SDS 200 / SDS 200A transmits acquired data to the user over USB cable. The functions for this are as follows.

sdGetDataBuffer()	To get data and report the data location of the buffer
sdGetData()	To fill the buffer data with voltage value
sdRaw2Physical()	To turn A/D converted data into physical value

sdStoreData()	To acquire the data samples and fill the buffer inside SoftScope library
sdGetDataVal()	To get the stored data from the buffer inside SoftScope library

2.4. Handling Errors

When the API of SoftScope is used, most return values become bool value, which shows the occurrence of errors.

If the return value is false, in other words, if an error occurs, further processing should be done by **sdGetErrorMessages()** function after the reason of the error is identified.

If an error occurs, the following error values will be returned and their meaning is as follows.

ERROR_MEMORY_ALLOCATION	Not enough memory for allocation
ERROR_USB_OPEN	USB Open Error
ERROR_USB_NO_SDS200_FOUND	No SDS 200(A) found
ERROR_USB_MANY_SDS200	Too many SDS 200(A) found
ERROR_USB_DEVICE_OPEN	USB Device Open Error
ERROR_USB_WINDRIVER_VERSION	USB Incorrect WinDriver version
ERROR_USB_WRITE_REGISTER	Can't write FPGA registers
ERROR_USB_READ_REGISTER	Can't read FPGA registers
ERROR_USB_READ_DATA	Can't read data from buffer
ERROR_USB_DATA_CONVERSION	Can't convert data from usb to raw data
ERROR_CALIBRATION_DATA_READ	Can't read calibration data from hardware
ERROR_CALIBRATION_DATA_WRITE	Can't write calibration data to hardware
ERROR_CALIBRATION_DATA_FORMAT	Format error of Calibration1 format

3. SDS 200 / SDS 200A Hardware

3.1. Three Operational Modes of SDS 200 / SDS 200A

3.1.1. Realtime mode

In Realtime mode SDS 200 / SDS 200A acquires 10,000(in 10K buffer size mode) sample data from the input source.
It can show the real data simultaneously. But because of the sampling speed limit of A/D converter, it is limited only to 10us/div time/div.
In Realtime mode and 500K buffer size mode, SDS 200A can acquire 500,000 sample data from the input source but this operation is limited from 1ms/div to 400ms/div.

3.1.2. ETS(Equivalent Time Sampling) mode

In ETS mode SDS 200/ SDS 200A acquires waveform as well as TDC value, and rearrange the waveforms using the TDC value to form one waveform.
For example, with first acquisition sampled data forms waveform A. With second acquisition sampled forms waveform B. Over and over again waveforms are interpolated to give a complete waveform. SDS 200 / SDS 200A has a 200ps time resolution in RIS mode.

3.1.3. Roll mode

To handle low frequency signals, SDS 200 / SDS 200A has a roll mode. In roll mode acquired waveform is displayed and moved from the right side of the screen to the left. As a result, it appears that the waveform is moving constantly.

SDS 200 / SDS 200A operates each mode in the following time scale base:

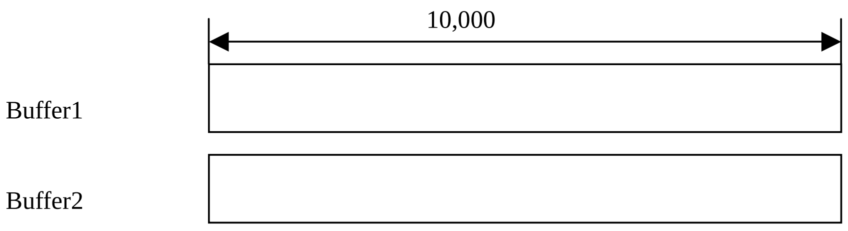
Sampling Mode	ETS Mode	Realtime Mode	Roll Mode
Buffer Size - 10K	**200ns/Div ~4us/Div**	**10us/Div~400ms/Div**	**1s/Div ~10s/Div**
Buffer Size - 500K	**-**	**1ms/Div~400ms/Div**	**-**

3.2. Buffer

The Buffer size can be informed by sdGetBufferSize() function. Basically in 10K buffer size mode, 10,000 sampling data are transferred over USB, however, the data size of each channel will be reduced to a half if two channels are used. In the case of ETS mode and Roll mode, the number of incoming data at a time varies case by case.
SdRaw2Physical() function plays such a role, and SdGetDataBuffer() function reports the pointer location of internal memory.

3.2.1. Internal buffer structure of SoftScope dll

Basically, SDS 200 / SDS 200A collects the data of 10,000 points at once and delivers it to PC. However, this value is little different for ETS Mode, RIS/RT Mode and Roll Mode. SoftScope dll has 2 memories that can store up to 10,000 data that has 16bit unsigned type so that it can receive all data regardless of all data.



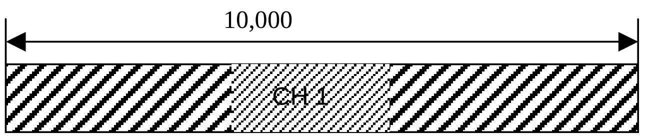
Data from SDS200 consists of 12 bit and exact data is made by multiplying calibration information to this data. Physical signal is made by multiplying uniform value according to the current volt/div setting again. SoftScope displays the data on the screen using 12 bit Data(0 – 4095) by default and converts it to voltage value when delivering the data outside.

3.2.2. Real Time Mode(10K) - Time/Div : 10us – 400ms

SDS 200 / SDS 200A collects the data of 10,000 points at one time and delivers it to PC. If two channels are used at the same time, No. 1 channel is used for the first half of the buffer and No. 2 channel is used for the second half of the buffer.

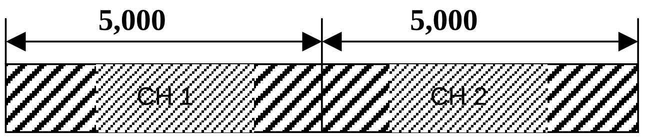
1. **If One Channel is used (Use 10,000 points)**

Buffer1



2. **If Two Channels are used(Use 5,000 points for each channel)**

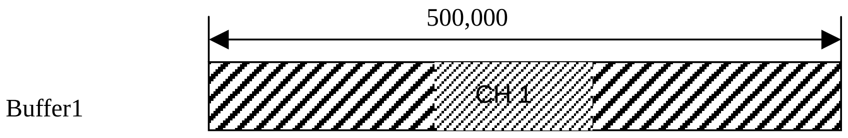
Buffer1



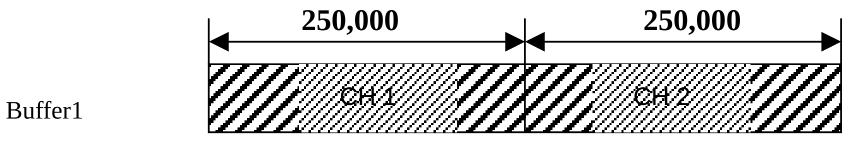
3.2.3. Real Time Mode(500K) - Time/Div : 1ms – 400ms

SDS 200A can collect the data of 500,000 points at one time and delivers it to PC in 500K buffer size mode. If two channels are used at the same time, No. 1 channel is used for the first half of the buffer and No. 2 channel is used for the second half of the buffer.

1. **If One Channel is used (Use 500,000 points)**



2. If Two Channels are used(Use 250,000 points for each channel)



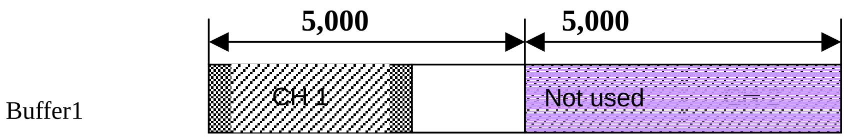
3.2.4. Roll Mode - Time/Div : 1s – 10s

Roll Mode is to receive data little by little real time and to scatter it in the screen since it takes too long to receive all data at one time if Time/div is very late. As a result, the screen seems to be flown from the right to the left.

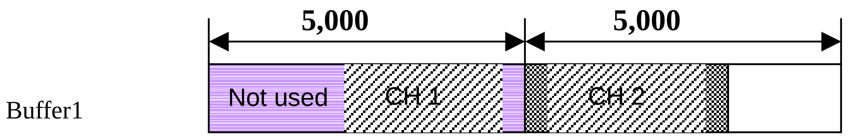
Since Roll Mode receives 3000 point data for each channel, it will display 6 point data for each x axis, different from real time mode which prints 20 points for each x axis.

1. If Turning on one Channel (Use 3000 points for each channel)

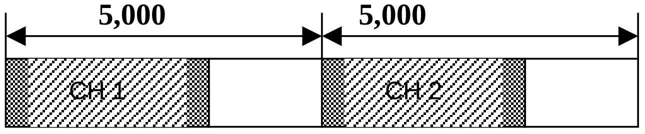
SoftScope Buffer (If using CH1)



SoftScope Buffer (If using CH2)



2. If using 2 Channels at the same time (Use 3000 points for each channel)

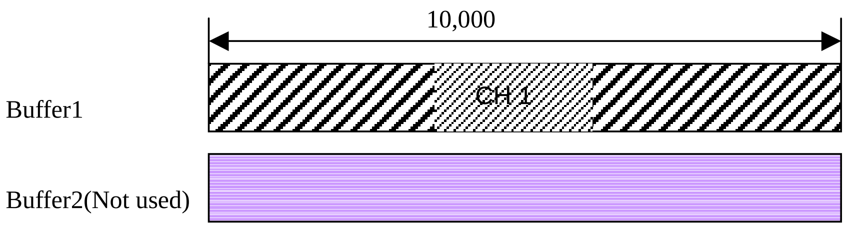


3.2.5. ETS Mode - Time/Div : 200ns – 4us

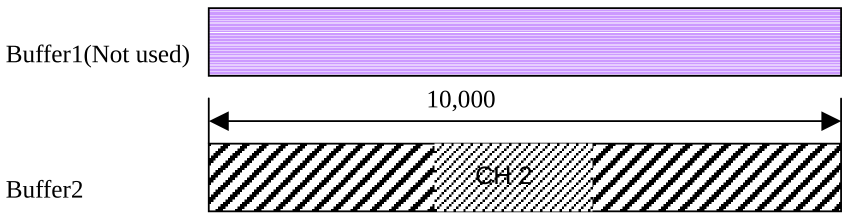
ETSMode uses the data of 10,000 points for each channel when using 2 channels at the same time.
The value below 200ns/div is for software process and same in terms of hardware.

1. If Turning on one Channel (Use 10,000 points)

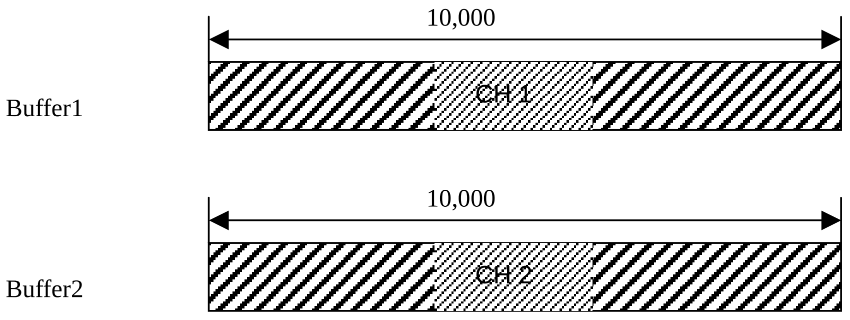
SoftScope Buffer (If using CH1)



SoftScope Buffer (If using CH2)



2. If Turning on Two Channels



In ETS mode, the buffer is all turned into RIS_MISSING value and then filled with new values. Therefore, if the buffer is not fully filled with new data in any case, RIS_MISSING value will be found instead.

From a point of view that a buffer is filled with data, it is identical to real-time mode, however, internally, the data are sampled repeatedly and the buffer is filled with them. If the buffer is not filled with data, there will be RIS_MISSING value.

3.3. Sampling Speed/Sampling Rate

The time interval between the acquired data can be computed by the following equation.

$$sampling\_speed = \frac{10 \times Time/div}{Number\_of\_Data}$$

$$sampling\_time = \frac{1}{sampling\_speed}$$

For example, if 10,000 data are acquired by 10us/div, it yields $10*10*10^{-6}/10^{4} = 10^{-8}$ second, and its reciprocal value, i.e., the sampling speed will be 100MS/s.

3.4. ETS (Equivalent Time Sampling)

A user can make decision to use ETS capability with sdSetETSOnOff function on condition that it is under 4us/div.

When ETS is turned off and if it is under 4us/div, the data will be sampled at 100MS/s for single channel and 50MS/s for two channels, respectively. The buffer will be filled in with sampled data. Unfilled part will be filled with RIS_MISSING value instead.

SoftScope program will fill the unfilled RIS_MISSING value by Sinc Interpolation.

4. Programming Example

4.1. Basic Notion

4.1.1. Setup for acquiring data

The following procedures should be done to get proper data from SDS 200 / SDS 200A.

4.1.1.1.Initialization and Finalization

When the program is started and ended, initialization and finalization should be carried out by sdIntialze() and sdFinalize() functions. In the case of initialization, the return value should be checked for error occurrence.

If finalization is not properly fulfilled after initialization, the USB cable should be plugged into and unplugged out of SDS 200 / SDS 200A.

4.1.1.2.Acquisition of parameters

- To turn on/off channel – sdSetChannelOnOff
- To set voltage axis – sdSetChannelVoltage
- To set channel coupling – sdSetChannelCoupling
- To set Channel offset – sdSetChannelOffset/sdSetChannelOffsetVoltage
- To set Time axis – sdSetTimeDiv
- To set Delay mode – sdSetDelayOnOff
- To set delay value – sdSetDelayOffset/sdSetDelayOffsetTime
- To set trigger source – sdSetTriggerSource
- To set trigger voltage – sdSetTriggerOffset/sdSetTriggerOffsetVoltage
- To set trigger mode – sdSetTriggerMode
- To set trigger slope – sdSetTriggerSlope
- To set ETS if it is under 4ns/div – sdSetETSOnOff

4.1.2. Acquisition of data

There are three ways to obtain data as follows.

1. Method 1 - To generate a data buffer and fill it with voltage value sdGetData
2. Method 2 - To read A/D converter value and turn it into voltage value, referring to the internal buffer value of SoftScope11.dll
3. Method 3 - To put the value into the internal buffer of SoftScope11.dll using sdStoreData function, and to fetch the buffer value using sdGetDataVal function sdGetDataBuffer/sdRaw2Physical

The first method is the fastest way but the programming language should allocate memory buffer in the heap. The second method is a little bit slower but doesn't need to allocate memory. The final method is the slowest but the easiest way to get data from SDS 200 / SDS 200A.
They are explained below with C++ and Visual Basic examples.

4.2. C++ Example

4.2.1. Get data using method 1

```
int bufferSize = 10000;
ch1Data = new PHYSICAL_DATA[bufferSize];
ch2Data = new PHYSICAL_DATA[bufferSize];
bufferSize = sdGetDataBufferSize();

ch1On = sdGetChannelOnOff(_CH1);
ch2On = sdGetChannelOnOff(_CH2);

// Get Data using sdGetData Function
if (ch1On && ch2On) {
    sdGetData(_BOTH, ch1Data, ch2Data);
}
else if (ch1On && !ch2On) {
      sdGetData(_CH1, ch1Data, ch2Data);
}
else if (!ch1On && ch2On) {
      sdGetData(_CH2, ch1Data, ch2Data);
}
```

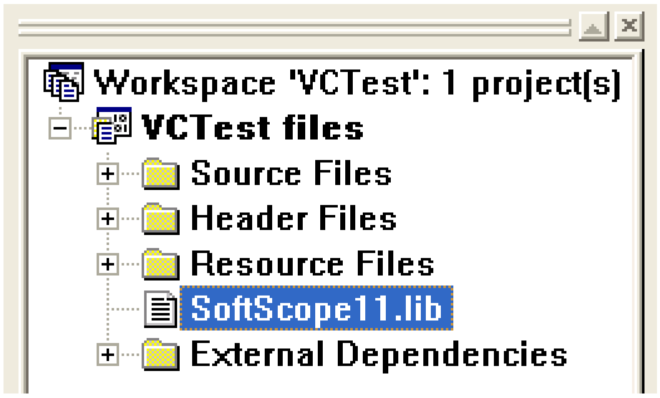
4.2.2. Get data using method 2

```
int bufferSize = sdGetDataBufferSize();
RAW_DATA* ch1Data, *ch2Data;
ch1On = sdGetChannelOnOff(_CH1);
ch2On = sdGetChannelOnOff(_CH2);

//Acquire Data.
if (ch1On && ch2On) {
      sdGetDataBuffer(_BOTH, &ch1Data, &ch2Data);
}
else if (ch1On && !ch2On) {
      sdGetDataBuffer(_CH1, &ch1Data, &ch2Data);
}
else if (!ch1On && ch2On) {
      sdGetDataBuffer(_CH2, &ch1Data, &ch2Data);
}

// Transform Raw data into Physical Data
for (int i = 0; I < bufferSize; i++) {
      PHYSICAL_DATA data1, data2;
      if (ch1On && ch2On) {
            data1 = sdRaw2Physical(_CH1, ch1Data[I]);
            data2 = sdRaw2Physical(_CH2, ch2Data[I]);
        }
        else if (ch1On && !ch2On) {
            data1 = sdRaw2Physical(_CH1, ch1Data[I]);
        }
        else if (!ch1On && ch2On) {
            data2 = sdRaw2Physical(_CH2, ch2Data[I]);
        }
}
```

4.2.3. VC++ programming example

There are some examples in Example directory for Visual C++. As the examples are written in Visual C++ 6.0, it is recommended to use this version As far as VC++ is concerned, header and Library should be linked properly, which can be done by the following procedures.

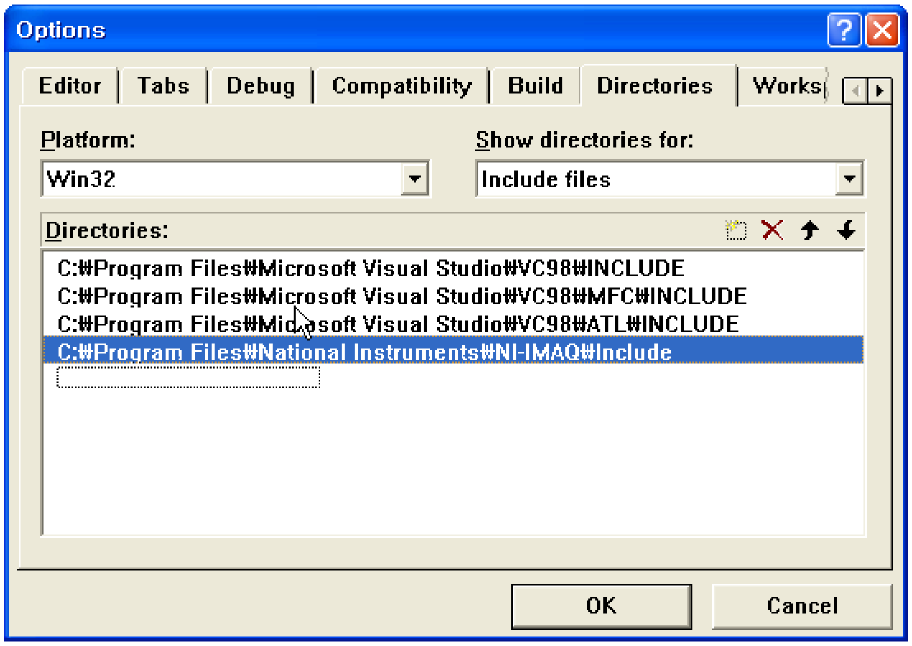
1. To add SoftScope11.lib to a project



```
Workspace 'VCTest': 1 project(s)
  VCTest files
    Source Files
    Header Files
    Resource Files
    SoftScope11.lib
    External Dependencies
```

2. To set the directory where headers exist

   Tools → Options → Directory



Platform: Win32

Show directories for: Include files

Directories:

```
C:₩Program Files₩Microsoft Visual Studio₩VC98₩INCLUDE
C:₩Program Files₩Microsoft Visual Studio₩VC98₩MFC₩INCLUDE
C:₩Program Files₩Microsoft Visual Studio₩VC98₩ATL₩INCLUDE
C:₩Program Files₩National Instruments₩NI-IMAQ₩Include
```

4.2.4. C++ Builder programming example

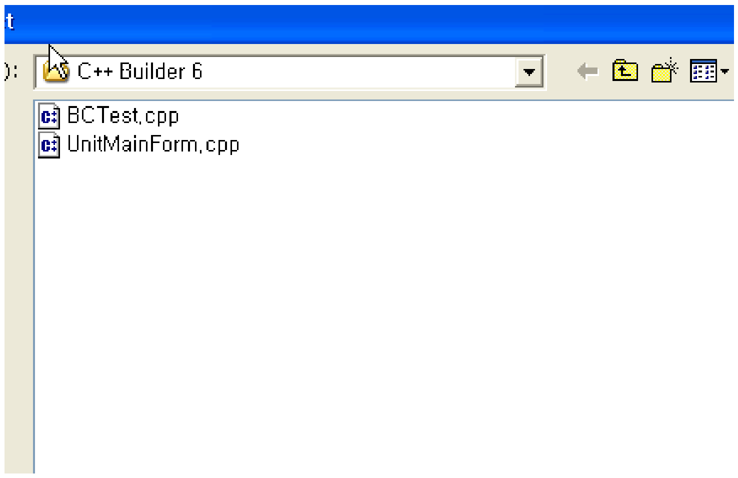
There are some examples in Example directory for C++ Builder.

As the examples are written in C++ Builder 6.0, it is recommended use this version.

In the case of VC++, header and Library should be linked properly, which can be achieved by the following steps.

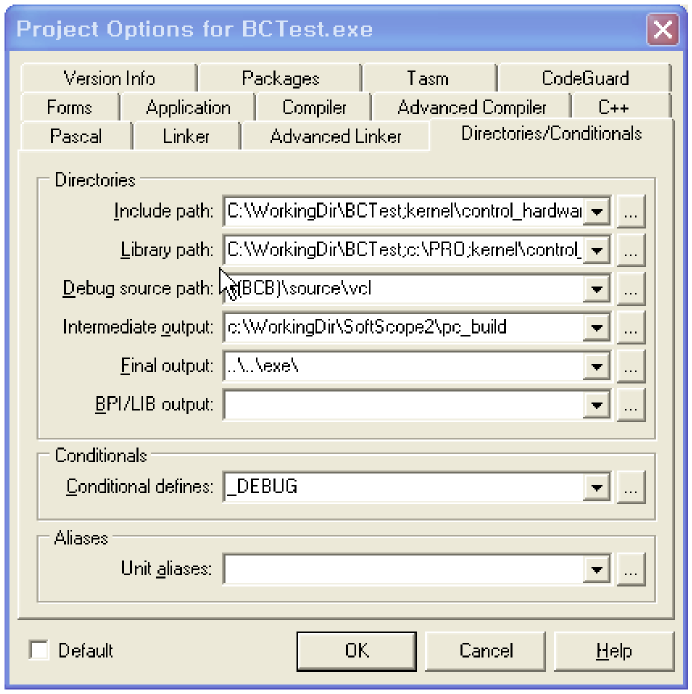
1. To add SoftScopeBuilder11.lib to a project

   Project → Add to Project



2. To set the directory in which headers exist

   Project → Options → Directories/Conditionals



Project Options for BCTest.exe

Version Info | Packages | Tasm | CodeGuard | Forms | Application | Compiler | Advanced Compiler | C++ | Pascal | Linker | Advanced Linker | Directories/Conditionals

Directories

Include path: C:\WorkingDir\BCTest;kernel\control_hardwa

Library path: C:\WorkingDir\BCTest;c:\PRO;kernel\control_

Debug source path: $(BCB)\source\vcl

Intermediate output: c:\WorkingDir\SoftScope2\pc_build

Final output: ..\..\exe\

BPI/LIB output:

Conditionals

Conditional defines: _DEBUG

Aliases

Unit aliases:

4.3. Visual Basic

Basically, as there is no pointer in Visual Basic, Long Type data should be used for its role. Besides, as allocating 10,000 Double Type data can cause some problem in Visual Basic, it is recommended to acquire the pointer of a buffer and convert its value.

4.3.1. Get data using method 2

4.3.1.1.1 Channel Acquisition

```
 Size = sdGetDataBufferSize()
Res = sdGetDataBuffer(S_CH1, ptr, Void)
If (Res = 0) Then
    MsgBox "Error Buffer Read"
    Call sdFinalize
    End
End If
For I = 0 To Size - 1
    ' Get Data
    CopyMemoryRead Val, ptr + 2 * I, 2
    fVal = sdRaw2Physical(S_CH1, Val)
Next I
```

4.3.1.2.2 Channel Acquisition

```
 Res = sdGetDataBuffer(S_BOTH, Ptr1, Ptr2)
 If (Res = 0) Then
          MsgBox "Error Buffer Read"
        Call sdFinalize
        End
    End If
    For I = 0 To Size - 1
        ' Get Data
        CopyMemoryRead Val1, Ptr1 + 2 * I, 2
        CopyMemoryRead Val2, Ptr2 + 2 * I, 2

        fVal1 = sdRaw2Physical(S_CH1, Val1)
        fVal2 = sdRaw2Physical(S_CH2, Val2)
    Next I
```

4.3.2. Get data using method 3

4.3.2.1.1 Channel Acquisition

```
Res = sdStoreData(S_CH1)
    If (Res = 0) Then
        MsgBox "Error Buffer Read"
        Call sdFinalize
        End
    End If
    For I = 0 To Size - 1
        ' Get Data
        fVal = sdGetDataVal(S_CH1, I)
        If (fVal <> SD_DATA_ERROR) Then
            ' Display Data
          Else
            MsgBox "Error in data getting"
        End If
    Next I
```

4.3.2.2.2 Channel Acquisition

```
Res = sdStoreData(S_BOTH)
If (Res = 0) Then
    MsgBox "Error Buffer Read"
    Call sdFinalize
    End
```

```
End If
For I = 0 To Size - 1
    ' Get Data
    fVal1 = sdGetDataVal(S_CH1, I)
    fVal2 = sdGetDataVal(S_CH2, I)
    ' Display Data
Next I
```

CopyMemoryRead function, which is used to read the data, is provided by SoftScope_API.bas
Detailed information about them can be obtained from the reference and examples.

4.4. Visual Basic Example

There are some examples for Visual Basic in Example directory.
In this example, how to get data from one-channel or two-channel mode as well as how to save the result in a file are described.
For Visual Basic programming environment, SoftScopeAPI.bas and SoftScope_type.bas module should be included.