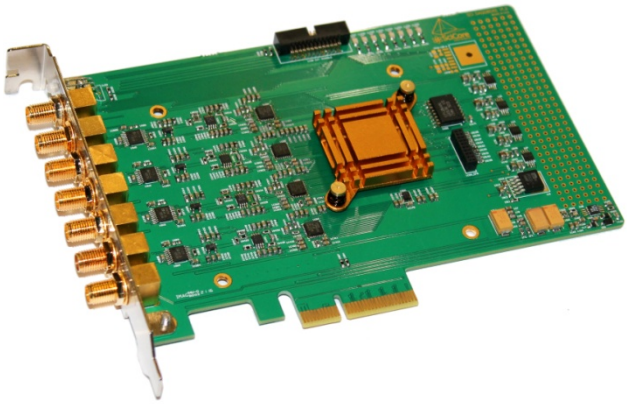
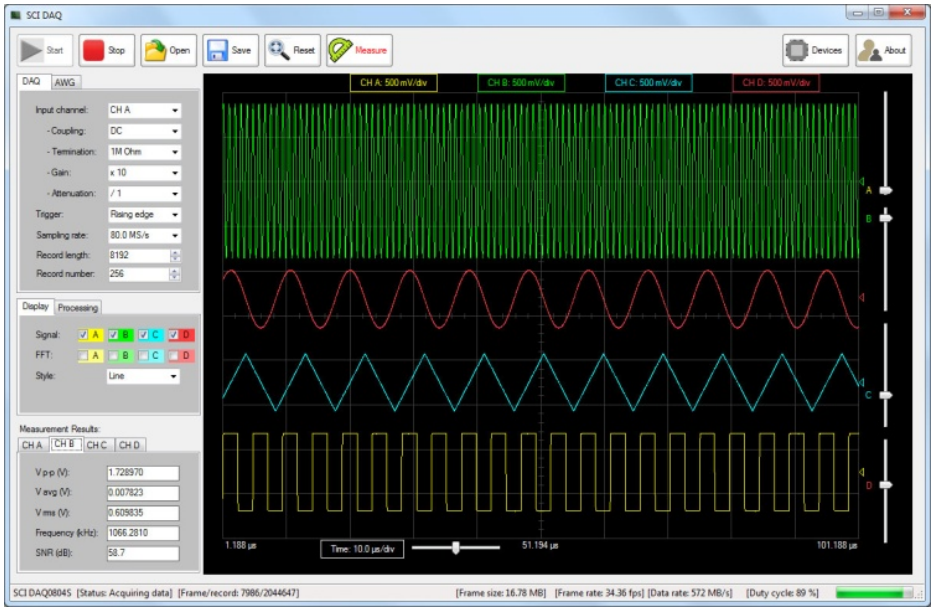
SCI-DAQ Software Development Kit (SDK)

User Manual V1.0



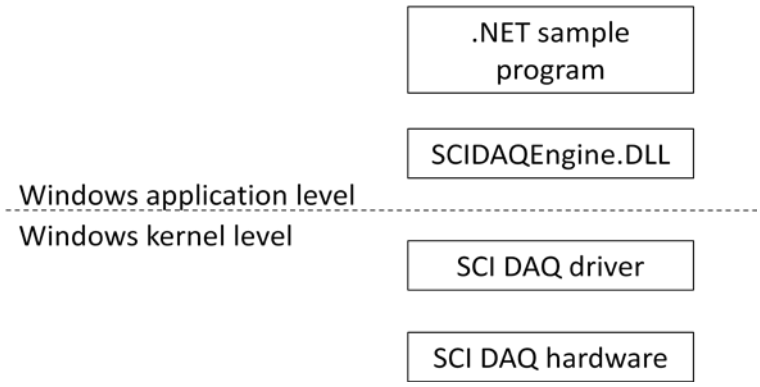


SciCore Instruments, Inc.

Chapter 1. SCI-DAQ SDK Overview

1-1. SDK structure

The overall structure of the SDK is described as the diagram below:

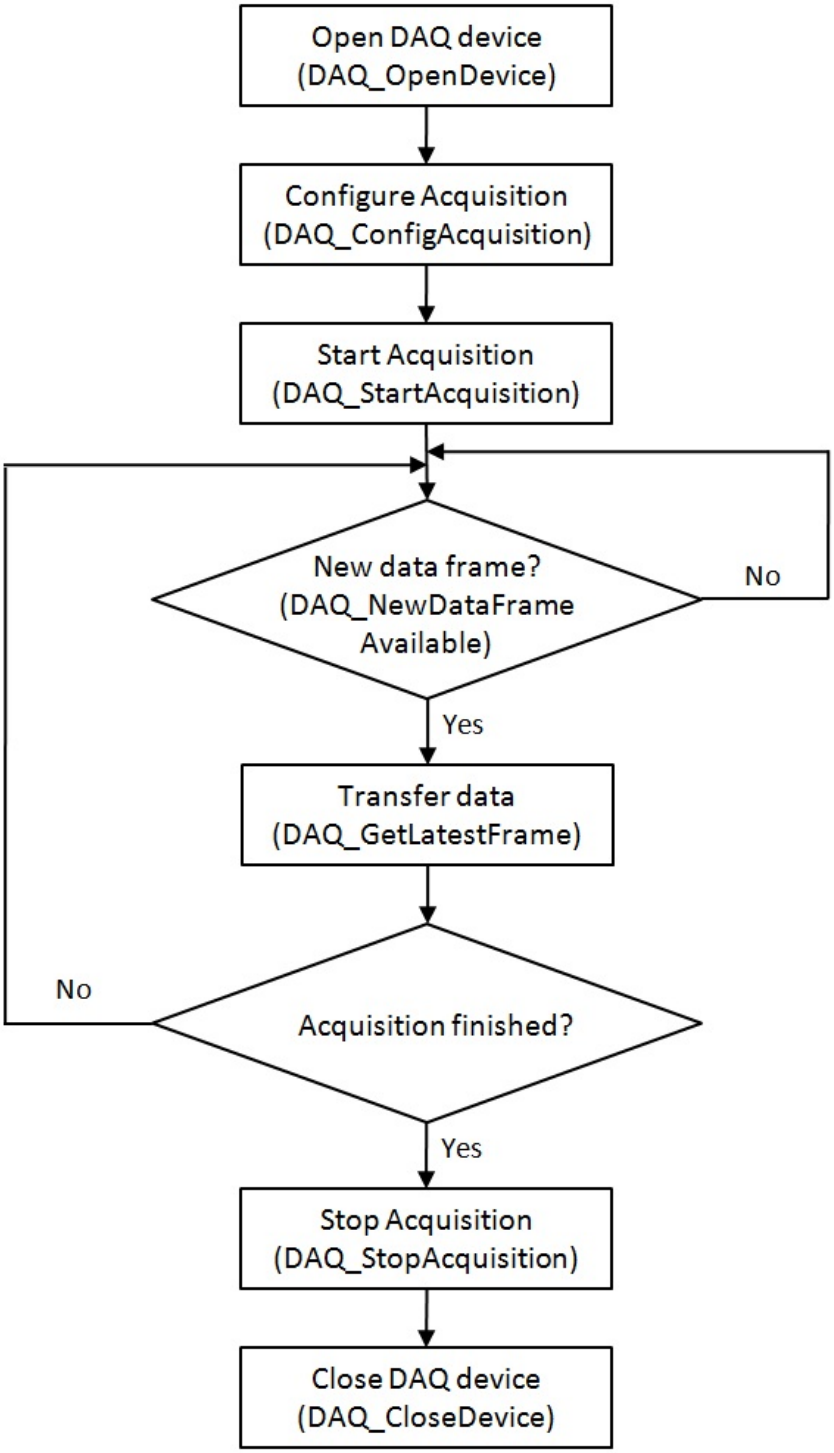


Located at the lowest level is the SCI DAQ hardware which is physically connected to the host PC via a PCI express bus or a USB cable. The SCI DAQ driver is responsible for direct reading and writing the control registers in the DAQ hardware to control the DMA (direct memory access) process, and transferring the acquired data from the DAQ hardware to the memory of the PC. The communicator between the driver and Window applications is SCIDAQEngine.DLL, which is a dynamically linked library providing the application programming interface (API) for application software running on Windows. Software sample programs based on Microsoft .NET development platform are provided to demonstrate the simplified software programming interface to the DAQ hardware.

After successful installation of the DAQ hardware and driver, the user can run the "SCI DAQ" software program to check the functions of the DAQ device. The user can also develop windows application software including the data acquisitions functions by calling the API functions supported by the SDK. It is recommended that the user copy the "SCIDAQEngine.DLL" to the same folder where the application software starts, so the application software can talk with the driver to control hardware functions.

1-2. Programming flow chart

The complete flow chart of a data acquisition task performed by the application software is provided below.



This flow chart includes following necessary steps:

Step 1: Open the DAQ device by calling the "DAQ_OpenDevice" API function.

Step 2: Configure the acquisition task by calling the "DAQ_ConfigureAcquisition" API function and providing parameters defining the size of a data frame for the DAQ to acquire.

Step 3: Start the acquisition by calling the "DAQ_StartAcquisition" API function.

Step 4: Check if there is a new data frame available for transfer to the application software, by calling the "DAQ_NewDataFrameAvailalbe" API function. If no repeat step 4; if yes go to step 5.

Step 5: Transfer the newly acquired data frame to the application software, by calling the "DAQ_GetLatestFrame" API function.

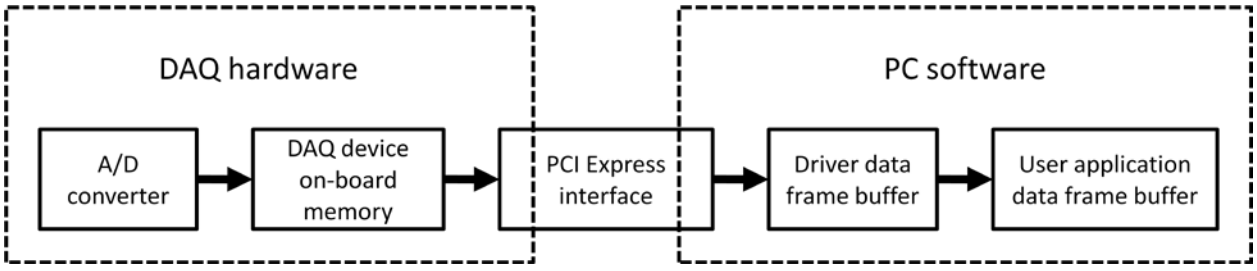
Step 6: Check if the acquisition task is finished. If no go to step 4 again; if yes go to step 7.

Step 7: Stop the data acquisition process by calling the "DAQ_StopAcquisition" API function.

Step 8: If there are other data acquisition tasks then go to step 2 again, otherwise close the DAQ device by calling the "DAQ_CloseDevice" API function.

1-3. Data transfer process

The complete process of the data transfer from the A/D converter on the DAQ hardware to the user application software is described in the diagram below:



1. The A/D converter transfers the acquired data points to the on-board memory of the DAQ hardware. The data points are organized into multiple data records with first data point in a data record synchronized with the trigger event, after the user calling the "DAQ_ConfigureAcquisition" API function.

2. The multiple data records are transferred to the data buffer in the PC memory via the PCI express interface, after the user calling the "DAQ_StartAcquisition" API function. The transferred data records are organized into data frames and stored in the data frame buffer

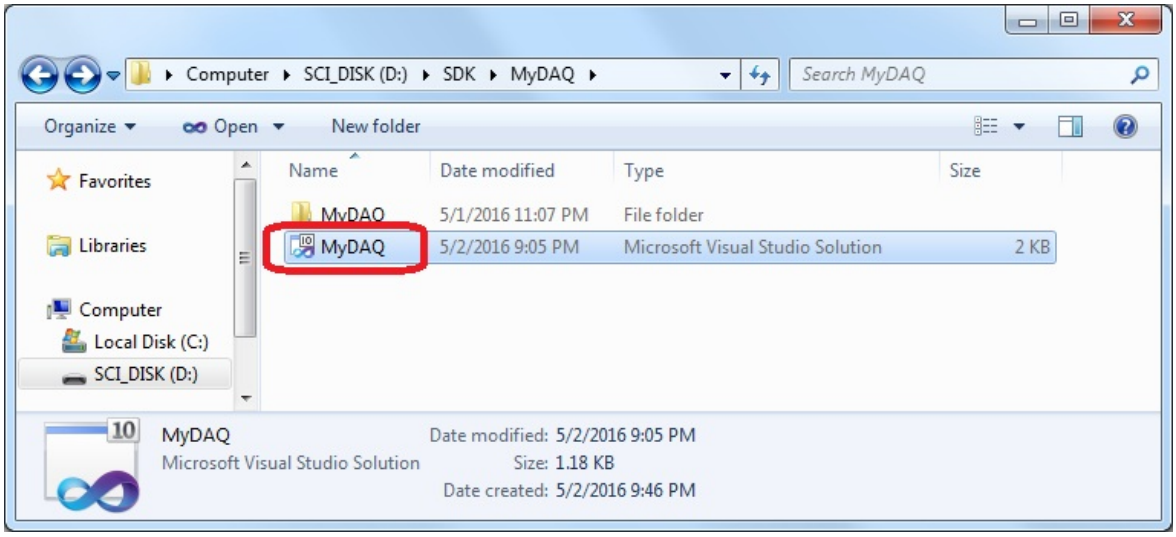
allocated by the DAQ driver software. When a new data frame is completely transferred, the driver setup an internal flag indicating a new data frame is available.

3. The user application software inquiries the driver if a new data frame is available by calling the "DAQ_NewDataFrameAvailalbe" API function. If there is a new frame available the application software then call the "DAQ_GetLatestFrame" API function to copy the data from the driver data frame buffer to the application software data buffer for further processing or display. The driver software will clear the internal flag indicating no new data frame is available, until another new data frame become available again later.

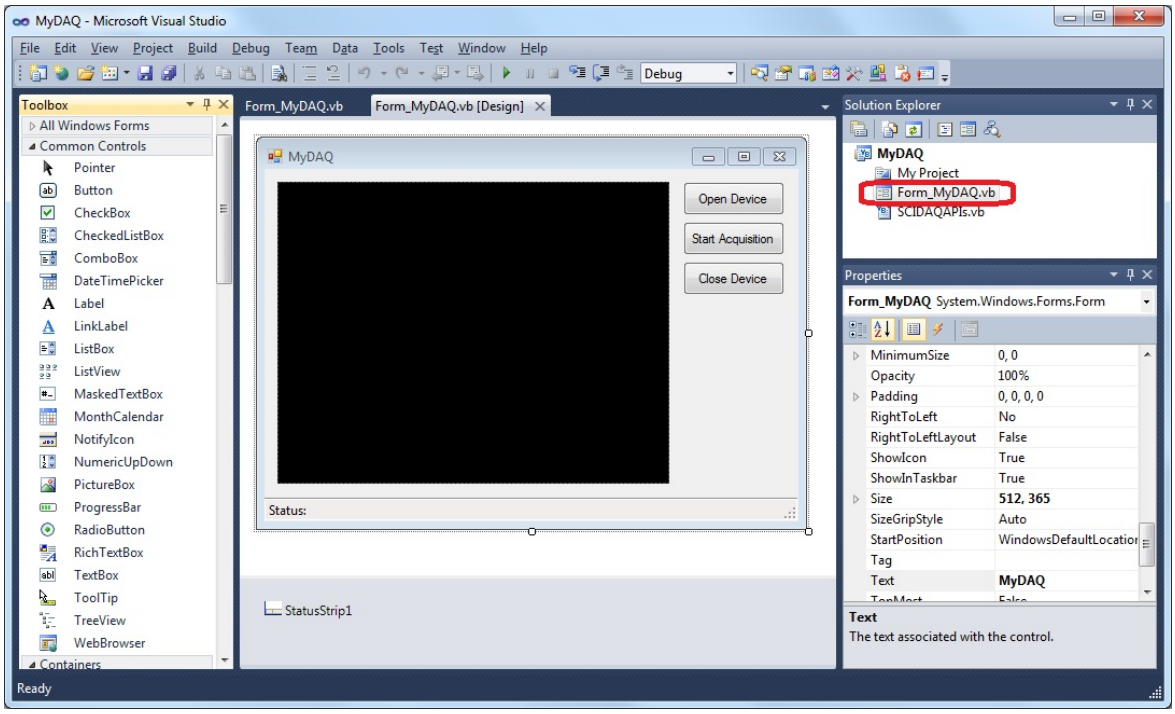
Chapter 2. SDK demonstration

This chapter provides a demonstration of writing a Windows application software performing some simple data acquisition tasks using this SDK. This demonstration is provided in Visual Basic.NET using Microsoft Visual Studio 2010 develop tool. The design concepts apply to other programming languages supported by newer versions of Microsoft Visual Studio.

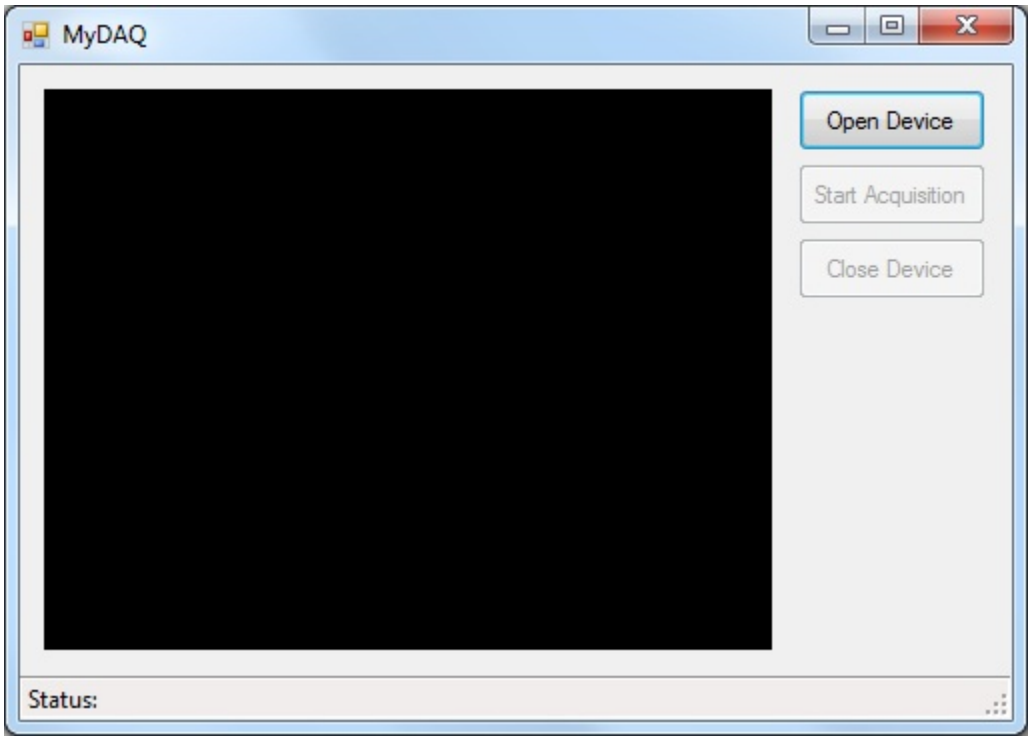
1. Open "MyDAQ.sln" in MicroSoft Visual Studio.



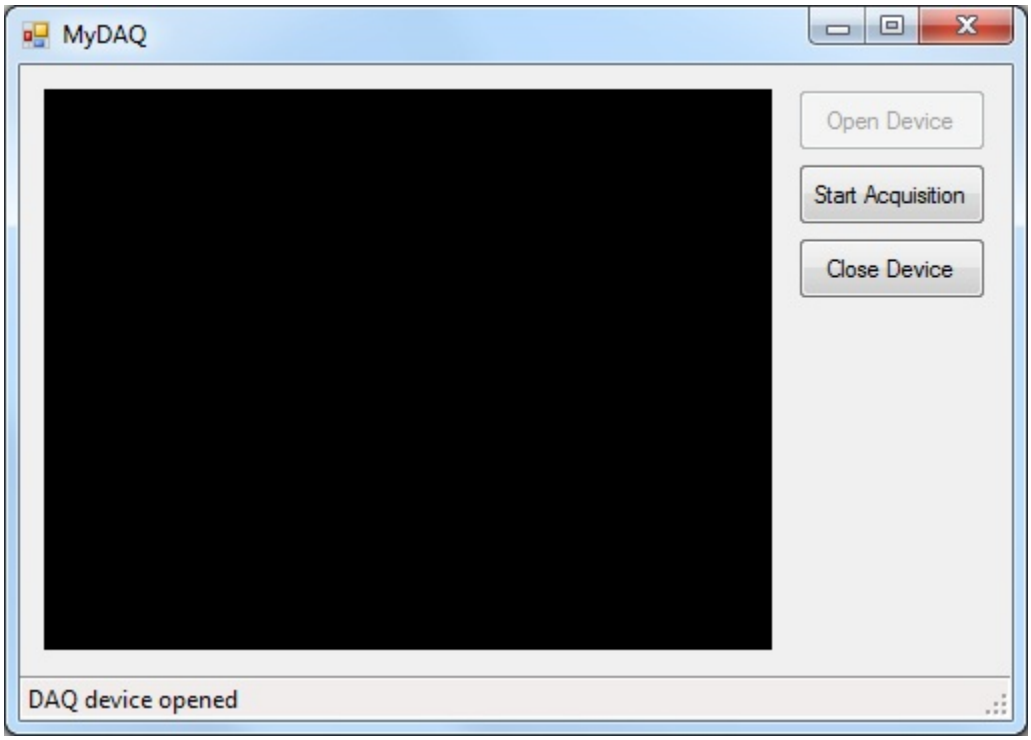
2. Double click the "Form_MyDAQ.vb" in Solution Explorer to view the main form in design mode. There are three buttons "Open Device", "Start Acquisition", and "Close Device" in the form.



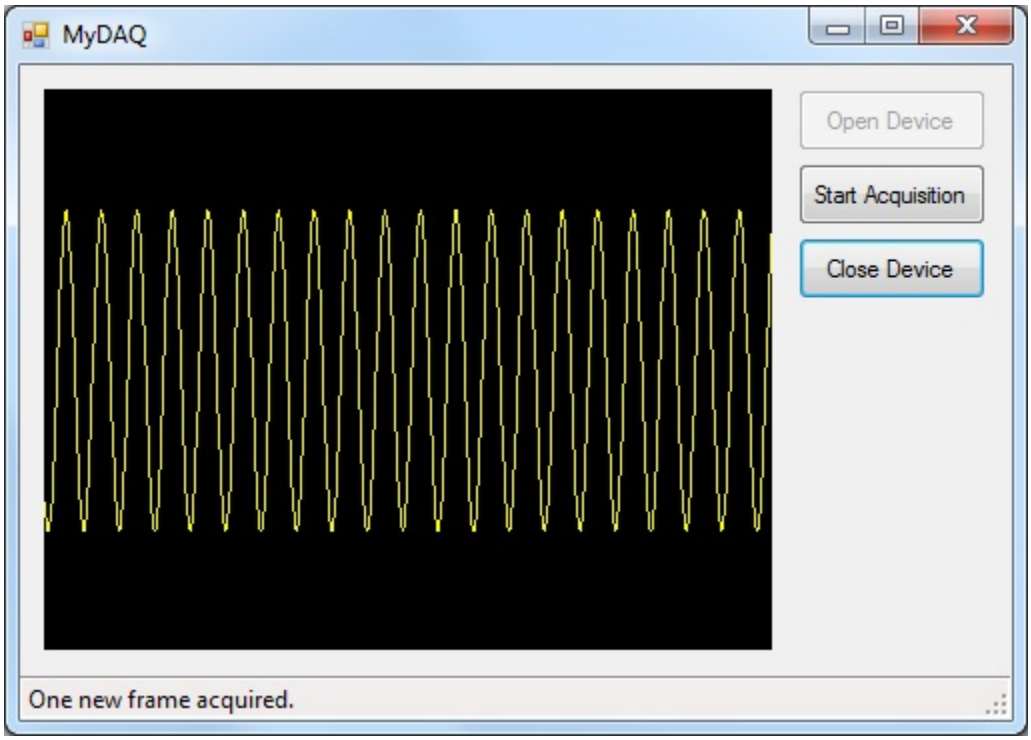
3. Click Menu (Debug->Start Debuging) or press F5 key to run the program.



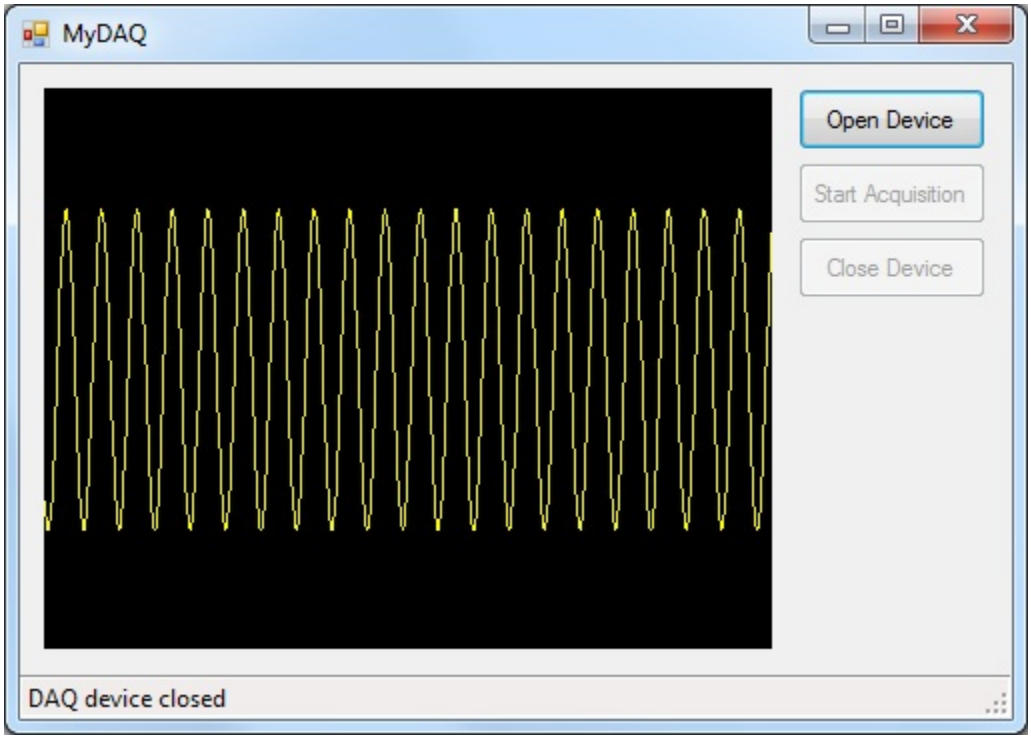
4. Click "Open Device" button to open the DAQ device.



5. Click "Start Acquisition" button to acquire one data frame and display the data in the PictureBox. In this example, a 2Vpp sine wave signal is connected to channel A input of the DAQ device.



6. Click "Close Device" button to close the DAQ device.



7. The complete source code of this demo program is shown below. The API functions used in this program are highlighted in red, and they are called in the order as describe in the previous flow chart.

```
Public Class Form_MyDAQ
    Dim DAQRecordLength As UInteger = 1024 'Number of data points per record
    Dim DAQRecordNumber As UInteger = 256 'Number of records per frame
    Dim RawDataFrame() As Short 'Raw data buffer
    Dim Flag_DAQOpened As Boolean 'If the device is opened
    Dim gr As Graphics 'Graphic resources for drawing the data
    Private Sub Button_OpenDevice_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_OpenDevice.Click
        If DAQ_OpenDevice(0) = 0 Then  'The device has been successfully opened
            Flag_DAQOpened = True
            StatusLabel.Text = "DAQ device opened"
            Button_OpenDevice.Enabled = False
            Button_StartAcq.Enabled = True
            Button_CloseDevice.Enabled = True
        End If
    End Sub
    Private Sub Button_StartAcq_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_StartAcq.Click
        Button_StartAcq.Enabled = False
        If Flag_DAQOpened Then
            DAQ_ConfigAcquisition(DAQRecordLength, DAQRecordNumber) 'Configure acquisition settings
            ReDim RawDataFrame(DAQRecordLength * DAQRecordNumber - 1) 'Allocate data buffer
            DAQ_StartAcquisition() 'Start the acquisition
            StatusLabel.Text = "Acquiring data..."
            While Not DAQ_NewDataFrameAvailable()  'Check if any new data frame is available
                Application.DoEvents()  'Wait if data is not ready
            End While
            DAQ_GetLatestFrame(RawDataFrame(0)) 'Transfer the data to the raw data buffer
            StatusLabel.Text = "One new frame acquired."
            DisplayRawData()  'Display the raw data in the picture box
            DAQ_StopAcquisition() 'Stop acquisition
        End If
        Button_StartAcq.Enabled = True
    End Sub
    Private Sub Button_CloseDevice_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button_CloseDevice.Click
        If Flag_DAQOpened Then
            DAQ_CloseDevice() 'Close the data acquisition device
            StatusLabel.Text = "DAQ device closed"
            Flag_DAQOpened = False
            Button_OpenDevice.Enabled = True
            Button_StartAcq.Enabled = False
            Button_CloseDevice.Enabled = False
        End If
    End Sub
    Private Sub DisplayRawData()
        Dim i, x, y, lastx, lasty As Integer
        gr = PictureBox_Signals.CreateGraphics
        gr.Clear(Color.Black)
        For i = 0 To DAQRecordLength - 1 Step 1
            x = i / (DAQRecordLength - 1) * PictureBox_Signals.Width
            y = (0.5 + RawDataFrame(i) / 16384) * PictureBox_Signals.Height
            If i > 0 Then gr.DrawLine(Pens.Yellow, x, y, lastx, lasty)
            lastx = x
            lasty = y
        Next
        gr.Dispose()
    End Sub
End Class
```

Please note in this example the data display is using the GDI+ methods drawing the PictureBox. The line drawing speed is pretty slow in this process. The data acquisition speed is actually much faster than drawing the data.

Above SDK demonstration program can be found in the USB installation disk come with the data acquisition device.

Chapter 3. API functions

This chapter provides a list of library functions provided by SCI-DAQ SDK.

DAQ_CheckSystem

Description: Check how many compatible DAQ devices are installed in the system, and assign a device ID starting from 0 for each device detected.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_CheckSystem Lib "SCIDAQEngine.dll" (
ByRef DeviceNum As Integer
) As Integer
```

Parameters:

[out] DeviceNum: The pointer that contains the value of number of compatible devices found.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_ OpenDevice

Description: Open a specified device for data acquisition tasks.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_OpenDevice Lib "SCIDAQEngine.dll" (
ByVal DeviceID As Integer
) As Integer
```

Parameters:

[in] DeviceID: The ID of the device to be opened.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_ CloseDevice

Description: Close the currently opened device.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_CloseDevice Lib "SCIDAQEngine.dll" (
) As Integer
```

Parameters:

None

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_GetDeviceModel

Description: Get the device model value for the specified device ID.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetDeviceModel Lib "SCIDAQEngine.dll" (
ByVal DeviceID As Integer,
ByRef DeviceModelValue As Integer
) As Integer
```

Parameters:

[in] DeviceID: The ID of the device to check its model.

[out] DeviceModelValue: The pointer that contains the value of the device model

DeviceModelValue	Device model name
5	DAQ0504M
6	DAQ1502S
8	DAQ0804S
9	DAQ2502S
-1	Unknown

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_ConfigAcquisition

Description: Set parameters to configure the device ready for data acquisition.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_ConfigAcquisition Lib "SCIDAQEngine.dll" (
ByVal RecordLen As UInteger,
ByVal RecordNum As UInteger
) As Integer
```

Parameters:

[in] RecordLen: The number of data points to acquire per record.

[in] RecordNum: The number of records to acquire per data frame.

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_StartAcquisition

Description: Start data acquisition immediately.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_StartAcquisition Lib "SCIDAQEngine.dll" (
) As Integer
```

Parameters:

None

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_StopAcquisition

Description: Stop current data acquisition process.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_StopAcquisition Lib "SCIDAQEngine.dll" (
) As Integer
```

Parameters:

None

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_NewDataFrameAvailable

Description: Check if there is a new data frame available for transfer.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_NewDataFrameAvailable Lib "SCIDAQEngine.dll" (
) As Boolean
```

Parameters:

None

Return value:

true: There is a new data frame ready for transfer.
false: No new data frame is available.

DAQ_GetLatestFrame

Description: Check if there is a new data frame available for transfer.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetLatestFrame Lib "SCIDAQEngine.dll" (
ByRef DataBuffer_CHA As Short,
Optional ByRef DataBuffer_CHB As Short = Nothing,
Optional ByRef DataBuffer_CHC As Short = Nothing,
Optional ByRef DataBuffer_CHD As Short = Nothing
) As Integer
```

Parameters:

[out] DataBuffer_CHA: Pointer to the buffer to be filled with the data acquired from Channel A.
[out] DataBuffer_CHB: Pointer to the buffer to be filled with the data acquired from Channel B.
[out] DataBuffer_CHC: Pointer to the buffer to be filled with the data acquired from Channel C.
[out] DataBuffer_CHD: Pointer to the buffer to be filled with the data acquired from Channel D.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_GetRecordLength

Description: Get the number of data points per record in the current acquisition settings.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetRecordLength Lib "SCIDAQEngine.dll" (ByRef
RecordLength As UInteger) As Integer
```

Parameters:

[out] RecordLength: The pointer that contains the value of the record length.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_GetMaxRecordLength

Description: Get the maximum number of data points per record supported by current acquisition device.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetMaxRecordLength Lib "SCIDAQEngine.dll" (
ByRef MaxRecordLength As UInteger
) As Integer
```

Parameters:

[out] MaxRecordLength: The pointer that contains the value of the maximum record length.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_GetRecordNumber

Description: Get the number of records per data frame in the current acquisition settings.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetRecordNumber Lib "SCIDAQEngine.dll" (ByRef
RecordNumber As UInteger) As Integer
```

Parameters:

[out] RecordNumber: The pointer that contains the value of the record number.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_SetInputChannel

Description: Select the analog input channel in the data acquisition device.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetInputChannel Lib "SCIDAQEngine.dll" (
ByVal InputChannel As UInteger
) As Integer
```

Parameters:

[In] InputChannel: The input channel to be selected.

Value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_GetInputChannel

Description: Get the analog input channel currently selected.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetInputChannel Lib "SCIDAQEngine.dll" (
ByRef InputChannel As UInteger
) As Integer
```

Parameters:

[Out] InputChannel: The pointer that contains the value of the selected input channel.

Value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_SetCoupling

Description: Set the coupling method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetCoupling Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByVal Coupling As UInteger
) As Integer
```

Parameters:

[In] Chan: Selected input channel

[In] Coupling: Selected coupling method

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Coupling value	Coupling method
0	DC coupling
1	AC coupling

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_GetCoupling

Description: Get the coupling method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetCoupling Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByRef Coupling As UInteger
) As Integer
```

Parameters:

[In] Chan: Selected input channel

[Out] Coupling: The pointer that contains the value of selected coupling method

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Coupling value	Coupling method

0	DC coupling
1	AC coupling

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_SetTermination

Description: Set the termination method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetTermination Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByVal Termination As UInteger
) As Integer
```

Parameters:
[In] Chan: Selected input channel
[In] Termination: Selected termination method

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Termination value	Termination method
0	1M Ohm termination
1	50 Ohm termination

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_GetTermination

Description: Get the termination method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetTermination Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByRef Termination As UInteger
) As Integer
```

Parameters:
[In] Chan: Selected input channel

[Out] Termination: The pointer that contains the value of selected termination method

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Termination value	Termination method
0	1M Ohm termination
1	50 Ohm termination

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_SetGain

Description: Set the gain for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetGain Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByVal Gain As UInteger
) As Integer
```

Parameters:
[In] Chan: Selected input channel
[In] Gain: The selected gain index

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Gain index	Gain
0	x 1
1	x 10
2	x 100

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_GetGain

Description: Get the gain for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetGain Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByRef Gain As UInteger
) As Integer
```

Parameters:

[In] Chan: Selected input channel

[Out] Gain: The pointer that contains the value of selected gain index

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Gain index	Gain
0	x 1
1	x 10
2	x 100

Return value:

0: Function finished successfully.

other value: Function failed.

DAQ_SetAttenuation

Description: Set the attenuation method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetAttenuation Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByVal Attenuation As UInteger
) As Integer
```

Parameters:

[In] Chan: Selected input channel

[In] Attenuation: Selected attenuation index

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Attenuation index	Attenuation
0	Divided by 1

1	Divided by 2

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_GetAttenuation

Description: Get the attenuation method for the selected channel.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetAttenuation Lib "SCIDAQEngine.dll" (
ByVal Chan As UInteger,
ByRef Attenuation As UInteger
) As Integer
```

Parameters:
[In] Chan: Selected input channel
[Out] Attenuation: The pointer that contains the value of the selected attenuation index

Chan value	Input channel
0	Channel A
1	Channel B
2	Channel C
3	Channel D

Attenuation index	Attenuation
0	Divided by 1
1	Divided by 2

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_SetTrigger

Description: Set the trigger method.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetTrigger Lib "SCIDAQEngine.dll" (
ByVal Trigger As UInteger
) As Integer
```

Parameters:
[In] Trigger: Selected trigger method

Trigger value	Trigger method
0	Line trigger

1	Rising edge trigger
2	Falling edge trigger

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_GetTrigger

Description: Get the trigger method.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetTrigger Lib "SCIDAQEngine.dll" (
ByRef Trigger As UInteger
) As Integer
```

Parameters:
[Out] Trigger: The pointer that contains the value of selected trigger method

Trigger value	Trigger method
0	Line trigger
1	Rising edge trigger
2	Falling edge trigger

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_SetSamplingRate

Description: Set the sampling rate for the current data acquisition device by specifying the sampling rate index.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_SetSamplingRate Lib "SCIDAQEngine.dll" (
ByVal SamplingRateID As UInteger
) As Integer
```

Parameters:
[In] SamplingRateID: The index of the selected sampling rate.

Return value:
0: Function finished successfully.
other value: Function failed.

DAQ_GetSamplingRate

Description: Get both the sampling rate index and sampling rate in MHz for the current data acquisition device.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_GetSamplingRate Lib "SCIDAQEngine.dll" (
ByRef SamplingRateID As UInteger,
ByRef SamplingRate_MHz As Single
) As UInteger
```

Parameters:

[Out] SamplingRateID: The pointer that contains the value of index of the sampling rate.
[Out] SamplingRate: The pointer that contains the value of the sample rate in MHz.

Return value:

0: Function finished successfully.
other value: Function failed.

DAQ_CheckSamplingRates

Description: Get the sampling rate in MHz associated with the sampling rate, and the total number of sampling rates supported by current data acquisition device.

Syntax:

```
[C/C++]
[Visual Basic]
Declare Function DAQ_CheckSamplingRates Lib "SCIDAQEngine.dll" (
ByVal SamplingrateID As Integer,
ByRef Samplingrate_MHz As Single,
ByRef NumSamplingRates As UInteger
) As Integer
```

Parameters:

[In] SamplingRateID: The index of the sampling rate to check.
[Out] SamplingRate_MHz: The pointer that contains the value of the sample rate in MHz.
[Out] NumSamplingRates: The pointer that contains the value of total number of sampling rates supported by current data acquisition device.

Return value:

0: Function finished successfully.
other value: Function failed.

SciCore Instruments Contact Information

SciCore Instruments, Inc.

P.O. Box 135

Long Valley, New Jersey 07853

E-mail: info@scicoreinstruments.com

Web site: www.scicoreinstruments.com

For comments and technical questions regarding SCI-DAQ SDK, send e-mail to:

support@scicoreinstruments.com