SDR14X.DLL

SDR-14/IQ ACTIVEX CONTROL SPECIFICATION

Document Version 1.08
2009-06-09

MOETRONIX

www.moetronix.com

Table of Contents

Introduction

The SDR14X ActiveX control provides a convenient way to interface various application programs to the SDR-14 and SDR-IQ hardware. It is written in C++ but since it exposes an ActiveX interface, can be used by Visual Basic and other languages that support ActiveX controls. The full Microsoft Visual Studio project source is available for free for those who may wish to customize the ActiveX control or spin their own SDR-14 driver.

SDR14X ActiveX Control Registration

The SDR14X ActiveX control must be registered on the users system before being used by the application.
There is a utility located in the Windows/System directory called regsvr32 that can be called from a command prompt to register the control.
It can be registered by shelling to the command prompt in the folder where the sdr14x.dll file resides and typing:
C:\windows\system32\regsvr32 sdr14x.dll
An easier way is to use the register.bat file provided in the sdr14x.dll distribution. It also includes the reg32svr.exe utility so one can just unzip all the files to a folder and run the batch file from there by clicking on it. To remove the control use the unregster.bat file or type:
C:\windows\system32\regsvr32 /u sdr14x.dll

Visual Basic Applications

Using the SDR14X control in a Visual Basic application is straightforward. (Probably will only work with VB 6.0 and not the .NET versions.)

1. Right-click on the toolbox. On the popup menu, click on "Components...".
2. Select the SDR14X ActiveX Control module in the Components selection view.
3. Click the OK button and the SDR14X control should now be displayed in the toolbox as shown.
4. The control can now be added to the form in the same manner as any other Visual Basic control.

Using the Control

.

Initialization

One can initialize the control in the Form's Load event as shown in the following example that specifies the IFFilter as 50KHz the Input as the HF preamp input using the AD6620 for complex data:

```
Private Sub Form_Load()
    Call SDR14X 1.InitControl( )
    SDR14X1.IFfilter = 3
    SDR14X1.InputSource = 129
    End Sub
```

Event Handlers

Notifications are sent to the application when the control's status changes. These notifications fire Event Handlers in the VB application. To create an event handler for the DataRdy notification, perform the following steps:
In the project Form code view window, select the control in the element select dropdown (left-hand dropdown) list. The default name for the control is SDR14X1.
In the event dropdown list (right-hand dropdown), select the DataRdy event. This will generate an empty handler for the DataRdy event.
Add processing to the DataRdy handler, such as shown in the following example:

```
Private Sub SDR14X1_DataRdy(ByVal SamplesAvailable As Long)
    Dim Samples As Integer
    Dim MyData(0 To 2047) As Integer
    Samples = SDR14X1.GetIntegerData( MyData(0) )
        REM   Process MyData
End Sub
```

Visual C++ MFC Applications

Using the SDR14X control in a Visual C++ MFC application can be done using the Visual studio wizards.

1. Create an MFC Dialog application using the Microsoft Visual Studio Application Wizard. Be sure to leave the checkbox for ActiveX Controls checked.
2. Click on the Project menu and select "Add To Project", then "Components and Controls...". When the Components and Controls Gallery dialog appears, double-click on the "Registered ActiveX Controls" folder. Scroll to the end of the controls display window and double click on the "SDR14X Control" item. Click OK when it prompts you to "Insert this component?".
3. A Confirm Classes dialog should appear asking if you want to accept the component, class name, and file names it will generate for you. Click OK to accept these.
4. You may now close the Components and Controls Gallery dialog window. Note that the SDR14X control has been added to the Controls toolbar.
5. The control can now be added to the form in the same manner as any other Windows control. The control will not be visible so it can be placed anywhere out of the way.

Adding the Control as a Member Variable

In an MFC dialog application, you will access the control as a member variable of the dialog form. To do this:

1. Start the Class Wizard and select the Member Variables tab, then double-click on the control ID for the SDR14X control to activate the Add Member Variable dialog window.
2. Note that the Category is set to Control and the Variable type is set to CSDR14X . Type in a name for the control's DDX member name that you wish to use for all accesses to the control:
3. Click the OK button and you will return to the Class Wizard dialog, which will display the newly added member mapping:

You can now access the SDR14X control in your dialog application through this member variable.

Using the Control

To initialize the control, add initialization code such as the following to the dialog's OnDlgInit function.

```
{
    m_ctlSDR14X.InitControl();              //Only need to do this once upon program entry
   m_ctlSDR14X.SetIFfilter (  3  ) ;
   m_ctlSDR14X.SetInputSource( 129 ) ;
              //wait for status to be idle
   m_ctSDR14X.Start( 1,2 ) ;               //just get one block and then stop.
}
```

In this example, the 50KHz IF filter and the HF preamp input are selected. The AD6620 is also specified so the data will be complex.

Event Handlers

Notifications are sent to the application when the control's status changes. These notifications fire Event Handlers in the MFC application. To create an event handler for the DataRdy notification, perform the following steps:

1. Start the Class Wizard and select the Message Maps tab.
2. In the Object IDs window, select the ID for the SDR14Xcontrol. Note that all of the events that the SDR14Xcontrol can fire are listed in the Messages list to the right.
3. Double-click on the DataRdy event and click the OK button on the Add Member Function dialog that pops up. The Class Wizard window will display the handler you've added similar to the following illustration:
4. Add processing to the OnDataRdyWinpskxctrl1 handler, such as shown in the following example:

```
void CExampleApp::OnDataRdySDR14X(long SamplesAvailable )
{
              short samples = m_CtrlSDR14X.GetLongData( m_pMyData );        //where m_pData is the users data array to hold the data
}
```

Methods

These are the ActiveX Methods (functions) that may be called by the user. Both Visual Basic and C syntax is shown.

CloseI2CPort

The CloseI2CPortmethod commands the SDR-14 to close the auxiliary 2 wire I2C port.

Syntax

VB	Sub CloseI2Cport()
C++	void CloseI2Cport();

Return Values

This method does not return a value.

Remarks

Not implemented in SDR-IQ.

CloseSpiPort

The CloseApiPort method commands the SDR-14 to close the Synchronous serial port.

Syntax

VB	Sub CloseSpiPort ()
C++	void CloseSpiPort ();

Return Value

This method does not return a value.

Remarks

Not implemented in SDR-IQ.

CloseUartPort

The CloseUartPort method commands the SDR-xx to close the UART port.

Syntax

VB	Sub CloseUartPort ()
C++	void CloseUartPort ();

Return Value

This method does not return a value.

Remarks

GetDoubleData

The GetDoubleData method is called to obtain the SDR-xx data in double precision floating point format after receiving a DataRdy Event.

Syntax

VB	Function GetDoubleData (DataArray As Double) As Long
C++	long GetDoubleData (double *DataArray);

DataArray

A pointer to the users data array that is to be filled with SDR-14 data.
The function converts the 16 bit signed sample data from the SDR-14 and converts it to double precision floating point data and copies it into DataArray. Also, a DC spur offset compensation value is added to the data to reduce the zero Hz spur.
The users data array must be at least as large as (Blocks x 8192)/2

Return Value

The number of samples placed in the users array is returned. If the data is complex, then there are two double values per sample.
For real mode data: Number samples = (Blocks x 8192) / 2
For complex mode data: Number samples = (Blocks x 8192) / 4

Remarks

The following example demonstrates a typical use of the GetDoubleData method as the result of a DataRdy event notification.
Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*

```
REM Assume blocks is 8 so array must be at 8 x (8192/2) = 32768
.Private Sub  ExampleX_DataRdy()
        Dim Samples As Long
        Dim MyData(0 To 32767) As Double
        Samples = ExampleX.GetDoubleData( MyData(0) )
        .....
        .....do something with MyData[]
End Sub
```

GetFftScreenData

The GetFftScreenData method is called to obtain the XSDR-14 FFT data in 32 bit Integer format after receiving a DataRdy Event. This method is useful for plotting FFT data on a bitmap screen of file because it automatically scales the FFT data to match the coordinates of the plot area.
The Start and Stop Frequency parameters are defined differently depending on whether the FFT is Real Or Complex(Using the SDR14 NCO to convert to baseband Complex data)

If Real: then StartFreq and StopFreq are the actual start and stop frequency range to display in Hz. within the 0 to 33MHz real capture band.

If Complex: then the StartFreq and StopFreq are the Relative frequencies in Hz to whatever SDR14 NCO center frequency has been set. (+/- xxx Hz) where the range of xxx depends on the IF bandwidth/ Sampling Frequency selected.

Syntax

VB	Function GetFftScreenData (PlotHeight As Long, PlotWidth As Long,MaxdB As Double, MindB As Double, StartFreq As Long, EndFreq As Long, PlotBuf As Long) As Boolean
C++	BOOL GetFftScreenData (long MaxHeight, long Plotwidth, double MaxdB, double MindB, long StartFreq, long EndFreq, long *PlotBuf);

PlotHeight

The plot height in pixels when plotting amplitude versus frequency. Can also specify color range for example if set to 255, then the fft output will range from 0 to 255 where ***255 is the minimum fft value***.

PlotWidth

The plot width in pixels when plotting amplitude versus frequency. The users PlotBuf must be large enough to contain at least this many elements.

MaxdB

Specifies the maximum FFT value in dB to display. This value will map to the screen top pixel position of zero.(screen height coordinates range from 0(top) to PlotHeight(bottom).

MindB

Specifies the minimum FFT value in dB to display. This value will map to the screen bottom pixel position of PlotHeight.(screen height coordinates range from 0(top) to PlotHeight(bottom).

StartFreq

The starting frequency in Hz, to map to the first element in the users data array = PlotBuf[0].
For complex FFT, StartFreq is +/- the frequency delta of the AD6620 NCO. In other words a start frequency of zero is the NCO frequency. The range is - SampleRate/2 to +SampleRate/2. StartFreq must also be less than the EndFreq.
For Real Non-AD6620 data, StartFreq ranges from 0 to 33,333,333Hz and must be less than EndFreq

EndFreq

The End frequency in Hz, to map to the last element in the users data array = PlotBuf[PlotWidth - 1].
For complex FFT, EndFreq is +/- the frequency delta of the AD6620 NCO. In other words an End frequency of zero is the NCO frequency. The range is - SampleRate/2 to +SampleRate/2. EndFreq must also be greater than StartFreq.
For Real Non-AD6620 data, EndFreq ranges from 0 to 33,333,333Hz and must be greater than StartFreq

PlotBuf

A pointer to the users buffer that will be filled with the FFT data represented as plot height coordinates. The array indexes are the x coordinates of the user plot, and the value in PlotBuf[x] is the y screen coordinate.

Return Value

A boolean is returned which is TRUE if the FFT data is close to over ranging, FALSE if it is not.

Remarks

Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*
This method provides an easy way to plot the FFT data by performing all the scaling and conversions needed to easily map to the screen. The FFT parameters need to be setup by calling the SetupFFT(..) method once before using this method to enable the FFT system.

Examples: (mixing basic and C syntax)

Assume the user screen plot bitmap area is 640 by 320 pixels.

```
Long MyPlotData[640];          REM  create buffer to hold plot data.
```

Real Mode: Want to plot amplitude in dB vs Frequency in Hz. from 13MHz to 18MHz with the vertical scale ranging from –20dB to –110dB.

```
On DataReady Event call the following:
        SDRXCtrl.GetFFTScreenData( 320, 640,  -20, -110, 13000000, 18000000, MyPlotData[0] )
        for( x = 0; x<640 x++)
            DrawLineTo(  x, MyPlotData[x]  )
```

Complex Mode: Want to plot amplitude in dB vs Frequency in Hz. from 13.95 MHz to 14.05 MHz with the vertical scale ranging from –20dB to –110dB.

```
SDRXCtrl.NCOfrequency = 14000000;          REM must set the SDR14 center frequency to desired value
SDRXCtrl.IFfilter = 4;                     REM set IF BW to 100KHz
SDRXCtrl.IFgain = 18;                      REM  set IF gain to some valid value

On DataReady Event call the following:
        SDRXCtrl.GetFFTScreenData( 320, 640,  -20, -110, -50000, 50000, MyPlotData[0] )
        for( x = 0; x<640 x++)
            DrawLineTo(  x, MyPlotData[x]  )
```

GetFftDoubleData

The GetFftDoubleData method is called to obtain the XSDR-14 FFT data in double precision floating point format after receiving a DataRdy Event.

Syntax

VB	Function GetFftDoubleData (DataArray As Double) As Boolean
C++	BOOL GetFftDoubleData (double *DataArray);

DataArray

A pointer to the users data array that is to be filled with SDR-xx data.
The function copies all FFT points into the users DataArray. The users data array must be at least as large as (FFT size) for complex data or (FFT size/2) for real data. Each array element corresponds to the corresponding FFT frequency bin.

Return Value

A boolean is returned which is TRUE if the FFT data is close to over ranging, FALSE if it is not.

Remarks

Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*
This method provides a way to obtain the SDR-xx FFT data in double precision format. The range of the data is 0dB to -140dB. The FFT parameters need to be setup by calling the SetupFFT(..) method before using this method.

GetIntegerData

The GetIntegerData method is called to obtain the SDR-xx raw data in 16 bit Integer format after receiving a DataRdy Event.

Syntax

VB	Function GetIntegerData (DataArray As Integer) As Long
C++	long GetIntegerData (short *DataArray);

DataArray

A pointer to the users data array that is to be filled with SDR-xx raw data.
The function copies the 16 bit signed sample data from the SDR-xx into DataArray.
The users data array must be at least as large as (Blocks x 8192)/2

Return Value

The number of samples placed in the users array is returned. If the data is complex, then there are two 16 bit values per sample.
For real mode data: Number samples = (Blocks x 8192) / 2
For complex mode data: Number samples = (Blocks x 8192) / 4

Remarks

The following example demonstrates a typical use of the GetIntegerData method as the result of a DataRdy event notification.
Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*

```
REM Assume blocks is 8 so array must be at 8 x (8192/2) = 32768
.Private Sub  ExampleX_DataRdy()
              Dim Samples As Long
              Dim MyData(0 To 32767) As Integer
              Samples = ExampleX.GetLongData( MyData(0) )
              .....
              .....do something with MyData[]
End Sub
```

GetUartData

The GetUartData method is called to obtain the SDR-xx aux UART data after receiving a UartDataRdy Event.

Syntax

VB	Function GetUartData(UartRxArray As Integer) As Integer
C++	short GetUartData(short *UartRxArray)

UartRxArray

A pointer to the users data array that is to be filled with SDR-14 Aux UART data.
The function converts the 8 bit UART data from the SDR-14 and converts it to a 16 bit integer and copies it into UartRxArray.
The users data array must contain at least 32 elements.

Return Value

The number of UART Rx data placed in the users array is returned.

Remarks

Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*
In general, the UART may buffer up to 32 bytes before issuing the UartDataRdy Event so the application must be able to handle multiple data bytes in the array.

InitControl

The InitControl Method initializes the ActiveX system and starts the internal thread that controls the USB port. This method MUST be called before any other methods or properties are used.

Syntax

VB	Sub InitControl( )
C++	void InitControl( )

Return Value

None.

Remarks

Note *This method activates the internal ActiveX worker thread and begins status monitoring of the USB port. The user program must be ready to handle events from the SDR14X control before calling this method. It only needs to be called once.*

OpenI2CPort

The OpenI2CPortmethod is called to open the transmit only SDR-14 auxiliary 2 wire I2C port.

Syntax

VB	Sub OpenI2CPort(AdrWidth As Integer)
C++	void OpenI2CPort(short AdrWidth);

AdrWidth

Specifies the I2C address width. Only valid values are 7 or 10 bit addresses.

Return Value

This method does not return a value.

Remarks

Opens an I2C serial port on the SDR-14. Data can only be transmitted from the SDR-14 using the SDI SDO signals.
Not implemented in SDR-IQ.

OpenSpiPort

The OpenSpiPort method is called to open the Transmit only SDR-14 auxiliary 3 wire Synchronous Serial port.

Syntax

VB	Sub OpenSpiPort (ClkPolarity As Integer)
C++	void OpenSpiPort (short ClkPolarity);

ClkPolarity

Specifies the Clock polarity. 0 = Inactive Low clock(positive edge), 1=Inactive High clock(negative edge)

Return Value

This method does not return a value.

Remarks

Opens a synchronous serial port on the SDR-14. Data can only be transmitted from the SDR-14 using the SCK, SDO, and LE signals.
Up to 64 bits can be transmitted at a time using the SendSpiData Method.
Not implemented in SDR-IQ.

OpenUartPort

The OpenUartPort method is called to open the SDR-xx auxiliary UART port.

Syntax

VB	Sub OpenUartPort(BaudRate As Integer, Parity As Integer, FlowControl As Boolean)
C++	void OpenUartPort(short BaudRate, short Parity, BOOL FlowControl);

BaudRate

The UART baud rate. (only 1200,2400,4800,9600, and 19200 bps is supported).

Parity

Specifies the UART parity. 0 = None, 1=Odd, 2=Even

FlowControl

Enables HW Flow Control. true=Enabled, false=None

Return Value

This method does not return a value.

Remarks

Opens an Asynchronous serial port on the SDR-14. Flow Control is limited to outgoing transmission and uses the SDI line as an input to determine if it is ok to send data to an external device. Unless Parity is enabled, the UART always sends 2 stop bits.
A UartDataRdy Event is fired whenever one or more bytes are received by the UART. The GetUartData Method can then be called to retrieve the received data.

ResetFftAverage

The ResetFftAverage method resets the FFT averaging buffers.

Syntax

VB	Sub ResetFftAverage ()
C++	void ResetFftAverage ();

Return Values

This method does not return a value.

Remarks

This method resets the FFT averaging buffers. When doing long averages and then the NCO frequency changes or some other parameter that would invalidate the previous averaging.

Send6620RawCmd

The Send6620RawCmd method is called to send custom AD6620 setup data to the SDR-xx hardware.

Syntax

VB	Sub Send6620RawCmd(Address As Integer,D0 As Integer,D1 As Integer,D2 As Integer,D3 As Integer,D4 As Integer)
C++	void Send6620RawCmd(short Address, short D0, short D1, short D2, short D3, short D4);

Address

The AD6620 Register Address to place the data.

D0, D1, D2, D3, D4

The Data to place in the specified AD6620 Register. Even though D0-D4 are passed as 16 bit Integers, they represent 8 bit data values and so have a range of 0 to 255.

Return Value

This method does not return a value.

Remarks

Refer to the Analog Devices AD6620 data sheet for details of the register data. This method is only for those who wish to customize their own filters and is not needed for most applications that can use the default AD6620 setups provided by the this ActiveX control.

SendI2CData

The SendI2Cdata method is called to send data from the SDR-14 aux I2C port to a slave I2C device.

Syntax

VB	Sub SendI2Cdata (Address As Integer, TxDataArray As Integer, NumDataItems As Integer)
C++	void SendI2Cdata (short Address, short * TxDataArray , short NumDataItems);

Address
Address of Slave I2C destination device.
TxDataArray
A pointer to the users data array that is to be sent by the SDR-14 Aux I2C port.
NumDataItems
Number of elements to send in TxDataArray

Return Value
This method does not return a value.

Remarks
Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*
Even though the data array is 16 bit Integers, each integer represents a byte of data to be sent by the I2C port and so has a range of 0 to 255.
Not implemented in SDR-IQ.

SendSpiData

The SendSpiData method is called to send data out the SDR-14 aux synchronous serial port

Syntax

VB	Sub SendSpiData(Data32High As Long, Data32Low As Long, NumberBits As Integer)
C++	void SendSpiData(long Data32High, long Data32Low, short NumberBits);

Data32High
Bits 63 to 32 of data to send.
Data32Low
Bits 31to 0of data to send.
NumberBits
Number of bits to send from Data32High and Data32Low. Range is 1 to 64.

Return Value
This method does not return a value.

Remarks
This method is used to send an arbitrary number of bits out the SPI port. The data to send is in two variables representing bits 0 to 31 and 31 to 63. The data should be right justified in the two integers.
Not implemented in SDR-IQ.

SendUartData

The SendUartData method is called to send data out the SDR-xx aux UART.

Syntax

VB	Sub SendUartData(UartTxArray As Integer, NumDataItems As Integer)
C++	void SendUartData(short * UartTxArray, short NumDataItems);

UartTxArray
A pointer to the users data array that is to be sent by the SDR-xx Aux UART.
NumDataItems
Number of bytes to send in UartTxArray

Return Value
This method does not return a value.

Remarks
Note *Note that, in VB, the array is passed by passing the first element in the array by reference, rather than passing just the array name*
Even though the data array is 16 bit Integers, each integer represents a byte of data to be sent by the UART and so has a range of 0 to 255.

SetupFFT

The SetupFFT method is called to enable and setup the FFT system.

Syntax

VB	Sub SetupFFT (FftPoints As Long, FftAverages As Integer, Boolean FftInverted)
C++	void SetupFFT (long FftPoints , short FftAverages , BOOL FftInverted);

FftPoints
Size of FFT to calculate. Range is 2048 to 262144 and must be a power of 2.
FftAverages
Number of bin by bin sample averages per FFT calculations. A value of 1 is no averages. Range is 1 to 32768
FftInverted
False == no inversion,True inverts frequency spectrum.

Return Value
This method does not return a value.

Remarks
Call to enable the FFT system and also setup its parameters. If FFT's are not required, do not call this routine as it will add some CPU overhead even if the FFT data is not retrieved. Calling this routine with zero for FftPoints will disable the FFT system.

Start

The Start method commands the SDR-xx to start capturing data.

Syntax

VB	Sub Start(Blocks As Integer, Mode As Integer)
C++	void Start(short Blocks, short Mode);

Blocks

Specifies the number of 8192 byte blocks to acquire before signaling a DataRdy Event.
Blocks range is 1 to 128. (Blocks are a fixed size of 8192 bytes or 4096 Real 16 bit samples, or 2048 16 bit Complex I/Q samples)
In general, the block size must be greater than or equal to the number of samples required by the FFT size.

For a real FFT, Blocksize >= FFTSIZE/4096. (Ex: with a FFT size of 16384, the block size should be 4.)
For a complex FFT, Blocksize >= FFTSIZE/2048. (Ex: with a FFT size of 32768, the block size should be 16.)

In Contiguous capture mode, block size is always one since the data is sent as soon as it is captured by the SDR14.

Mode

Specifies the capture mode.
Mode=0 Contiguous capture of data without resetting the SDR-14 FIFO. DataRdy Event fires after every 8192 byte block. This mode is used for obtaining data for demodulation and can only be used if the sample rate is less than about 198K Samples/Sec.

Mode=1 Continuous capture of specified blocks. After the specified number of 8192 byte blocks of data is captured, the DataRdy Event is fired and the SDR-14 FIFO is then reset. The process then repeats.

Mode=2 Captures specified blocks of 8192 byte data, fires the DataRdy Event, and then stops.

Mode=3 Same as Mode 1 except waits for HW sync signal on SDR-14 before capturing the N blocks of data. (Used for sample rates > 198 KSPS).

Moe = 4 Same as Mode 2 except waits for HW sync signal on SDR-14 before capturing the N blocks of data. (Used for sample rates > 198 KSPS).

Return Value

This method does not return a value.

Remarks

Modes 3 and 4 are only available on the SDR-14 and require a hardware modification to be able to use.
Before calling the Start method, the following Properties must be set up:
InputSource
IFfilter (if Input Source is Complex)

Stop

The Stop method commands the SDR-xx to stop capturing data.

Syntax

VB	Sub Stop()
C++	void Stop();

Return Value

This method does not return a value.

Remarks

The Stop method commands the SDR-xx to stop capturing data. It is possible that one more DataRdy event will fire after this command is received.

Properties

These are the ActiveX Properties that may be accessed by the user. Both Visual Basic and C syntax is shown. These are read and written using access methods starting with the word "Get" or "Set". Some properties are read only.

BootRevision

Data Type

VB	Integer
C++	short

Access

Read Only

Remarks

Value represents the current SDR-xx Boot Code version. Divide value by 100 to obtain version. Example 123 is version 1.23.

ClockFrequency

Data Type

VB	Long
C++	long

Access

Read/Write

Remarks

This value represents the primary A/D sample rate of the SDR-xx which is nominally 66.666667MHz. Although this actual frequency is crystal controlled and cannot be changed, modifying this property allows the software to calibrate to the actual sample frequency by setting it to the actual measured sample frequency. This property is used to calculate the AD6620 NCO frequency and does not affect any real mode data.

DeviceName

VB	String
C++	CString

Access

Read Only

Remarks

This String contains the SDR-xx device name which is normally "SDR-14" or “SDR-IQ”

FirmwareRevision

VB	Integer
C++	short

Access

Read Only

Remarks

Value represents the current SDR-14 Firmware Code version. Divide value by 100 to obtain version. Example 123 is version 1.23.

HFgain

VB	Integer
C++	short

Access

Read/Write

Remarks

Value	Description
0	HF filter/preamp at maximum gain
-10	HF filter/preamp with -10dB attenuator
-20	HF filter/preamp with -20dB attenuator
-30	HF filter/preamp with -30dB attenuator

Value represents the current SDR-14 Firmware Code version. Divide value by 100 to obtain version. Example 123 is version 1.23.

IFfilter

VB	Integer

C++	short

Access
Read/Write

Remarks
The filter value only takes affect when the Start Method is called.
The SDR-IQ only supports BW values 0 through 5, and 12.

Value	Description
0	5 KHz BW IF filter (Decimation Rate=8192, Sample Rate=8138 Hz)
1	10 KHz BW IF filter (Decimation Rate=4096, Sample Rate=16276 Hz)
2	25 KHz BW IF filter (Decimation Rate=1764, Sample Rate=37793 Hz)
3	50 KHz BW IF filter (Decimation Rate=1200, Sample Rate=55.555 KHz)
4	100 KHz BW IF filter (Decimation Rate=600, Sample Rate=111.111 KHz)
5	150 KHz BW IF filter (Decimation Rate=420, Sample Rate=158.730 KHz)
6	250 KHz BW IF filter (Decimation Rate=220, Sample Rate=303.030 KHz)
7	500 KHz BW IF filter (Decimation Rate=116, Sample Rate=574.713 KHz)
8	1 MHz BW IF filter (Decimation Rate=52, Sample Rate= 1.282051 MHz)
9	1.5 MHz BW IF filter (Decimation Rate=32, Sample Rate= 2.08333 MHz)
10	2 MHz BW IF filter (Decimation Rate=20, Sample Rate= 3.333333 MHz)
11	4 MHz BW IF filter (Decimation Rate=16, Sample Rate= 4.166667 MHz)
12	190 KHz BW IF filter (Decimation Rate=340, Sample Rate= 196.078 KHz)

IFgain

VB	Integer
C++	short

Access
Read/Write

Remarks

Value	Description
0	0 dB IF Gain
6	6 dB IF Gain
12	12 dB IF Gain
18	18 dB IF Gain
24	24 dB IF Gain

IQOffset

VB	Double
C++	Double

Access
Read Only

Remarks
IQOffset is an SDR-xx DC offset compensation value that can b added to both the I and Q data to reduce the Zero Hz spur in the baseband data. It is recalculated whenever the IFfilter or IFgain is set or changed.

InputSource

VB	Integer
C++	short

Access
Read/Write

Remarks
The SDR-IQ only supports Input source value 129, HF preamp complex data.

Value	Description
0	Direct Input AD6620 Bypassed Real Data
1	HF Preamp Input AD6620 Bypassed Real Data
128	Direct Input AD6620 Enabled Complex I/Q Data
129	HF Preamp Input AD6620 Enabled Complex I/Q Data

InterfaceVersion

VB	Integer
C++	short

Access
Read Only
Remarks
Value represents the current SDR-xx ASCP Interface version. Divide value by 100 to obtain version. Example 123 is version 1.23.

NCOfrequency

VB	Long
C++	long

Access
Read/Write
Remarks
Value represents the current AD6620 NCO frequency in Hz. The range is 1 to 33,333,333 Hz.

SampleRate

VB	Long
C++	long

Access
Read Only
Remarks
Value represents the current data sample rate out of the SDR-xx. If the AD6620 down converter is active, this value is sample rate after decimation by the AD6620. This value is updated AFTER setting the IFfilter Property.

SerialNumber

VB	String
C++	CString

Access
Read Only
Remarks
This String contains the SDR-14 serial number which is normally "ZZxxxxxx" where ZZ is a date code and xxxxxx is the serial number.

Events

These are the ActiveX Events that are "fired" by the control to indicate a status change, or that new data is available to be processed by the user. Both Visual Basic and C syntax is shown.

DataRdy

The DataRdyEvent is fired to indicate data is available from the SDR-xx.

Syntax

VB	Sub DataRdy(BlocksAvailable As Long)
C++	void DataRdy(long BlocksAvailable);

BlocksAvailable

Specifies the number of Data Blocks that are available to be read.
For Complex I/Q data, a block is 2048 complex 16 bit samples (8192 bytes)
For Real data, a block is 4096 16 bit samples (8192 bytes)

Remarks

This Event will fire very quickly so the user must be able to process the data quickly. With a block count of 1(8192 bytes), this event can come every 10mSeconds or so. The user has the option of retrieving the SDR-xx data in several formats. One can get the raw data in 16 bit, 32 bit, or double floating point formats. One can also get the data after it has been processed with an FFT algorithm which provides spectral information.

StatusChange

The StatusChange Event is fired to indicate the current SDR-xx Status.

Syntax

VB	Sub StatusChange (Status As Long)
C++	void StatusChange (long Status);

Status

Specifies the SDR-xx status.
0 == SDR-xx USB not connected
1 == SDR-xx Idle
2 == SDR-xx Running
3 == SDR-xx Error
4 == SDR-IQ A/D Clipping Error (Not available in the SDR-14)

Remarks

This status event will fire periodically when Idle or not connected not just when the status changes.

UartDataRdy

The UartDataRdy Event is fired to indicate there is data available from the SDR-xx Auxiliary UART port.

Syntax

VB	Sub UartDataRdy()
C++	void UartDataRdy();

Remarks

When this event is fired, one would call GetUartData (..) to obtain the UART data. If the Auxiliary UART port is not used then this even does not have to be implemented.