Visual Basic (VB DOT NET) USB-1616HS AInScan Example

The following example demonstrates how to continuously read a group of channels using the AInScan function and the USB-1616HS. Although not tested, the example should also work with the USB-2500 and PCI-2500 series because they are all of the same family of devices. Helper functions are used but not shown as they typically are do not demonstrate MccDaq functionality. To view these function load the project that is attached below as a zip file.

To access the API, add a reference to the MccDaq object. Adding the reference is usually accomplished by right clicking the Project [under the Project Explorer] and selecting Add Reference.

The functionality is fairly simple, the program puts data into the buffer and the program retrieves it one half buffer at a time. To command the API to sample continuously, the Scan Options enumeration Background and Continuous is used. How it collects data is it ping-pongs between reading the lower half then the upper in attempt to keep up with the incoming data. Care should be taken that data doesn't come so fast that the program fails to keep up. When this happens a buffer overrun occurs which is condition where unread data get overwritten.

As an additional check, the data is written to a text file that can be viewed in NotePad, Excel, DASYLab and MatLab. Below is a screen from DASYLab.

Disclaimer:
The attached Code or Example is provided As Is. It has not been tested or validated as a product, for use in a deployed application or system, or for use in hazardous environments.
You assume all risks for use of the Code or Example

```
'import statements help reduce line length and typing
Imports MccDaq
Imports MccDaq.ChannelType
Imports MccDaq.Range
Imports MccDaq.ScanOptions
Imports MccDaq.DigitalPortType
Imports MccDaq.FunctionType
Imports MccDaq.ErrorInfo.ErrorCode
Imports MccDaq.MccService
Imports System.Convert
Imports System.Threading.Thread
Imports System.Timers
Imports System.Text
Imports System.IO

Module Module1
```

```
    Const BLOCKSIZE As Integer = 50 'samples per channel
    Const FIRSTCHANNEL As Integer = 0
    Const LASTCHANNEL As Integer = 3
    Const CHANNELCOUNT As Integer = LASTCHANNEL - FIRSTCHANNEL + 1
    Const SAMPLERATE As Integer = 100 'desired sample rate
    Const BUFFERSIZE As Integer = CHANNELCOUNT * BLOCKSIZE ' size of the buffer
    Const HALFBUFFER As Integer = BUFFERSIZE / 2 'size of half buffer
    Const TOTAL As Integer = SAMPLERATE * CHANNELCOUNT * 60 * 5 '5 minutes

    'AInScan running options
    Dim theOptions As MccDaq.ScanOptions = Continuous + Background + ConvertData


    Dim CurrentCount As Integer = 0 'indicates total samples taken
    Dim CurrentIndex As Integer = 0 'points to the last scan read


    Dim Status As Short = 0

    Dim scale As Single = Convert.ToSingle(10.0 / 32768) 'volts per bit for the +/- 10 volt range

    Dim theDevice As MccBoard 'device object
    Dim buffer As IntPtr
    Dim ret As ErrorInfo
    Dim ReadLower As Boolean = True 'semaphore flag that prevents duplicate buffer reads

    'text file name and path. Use ASC extension if using DASYLab software to read the file.
    Dim Path As String = "C:\Users\Public\Documents\theDataFile.asc"

    Dim fStream As StreamWriter
    '''''''''''''''''''''''''''''''''' main function '''''''''''''''''''''''''''''''''''''''
    Sub Main()

        Dim ret As MccDaq.ErrorInfo
        Dim bNum As Integer
        Dim str As String = "USB-1616HS"

        bNum = GetBoardNum(str)

        If bNum = -1 Then
            Console.WriteLine("No {0} detected!", str)
            WaitForKey()
            End
        End If
        'read the first 3 channels
        Dim First As Integer = FIRSTCHANNEL
        Dim Last As Integer = LASTCHANNEL

        'holds the actual rate returned by the api
        Dim sRate As Integer = SAMPLERATE


        'fetch the board object
        theDevice = New MccBoard(bNum)

        'allocate a buffer
        buffer = MccService.WinBufAllocEx(BUFFERSIZE)

        'Start the acquisition
        ret = theDevice.AInScan(First, Last, BUFFERSIZE, SAMPLERATE, Bip10Volts, buffer, 
theOptions)
        If ret.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then Stop


        'create a text file to hold the data
        fStream = New StreamWriter(Path)
        CreateFileHeader()


        ''''''''''''''''' Continous LOOP ''''''''''''''''''''''
        Do


            ReadBuffer()

        Loop Until ((Console.KeyAvailable = True) Or (CurrentCount > TOTAL))

        '''''''''''''''''' End Loop '''''''''''''''''''''''''''''''

        theDevice.StopBackground(AiFunction)
        fStream.Close()
        WaitForKey()

    End Sub '''''''''''''''''''''' End Main function ''''''''''''''''''''''''''''''''''''''


    'function that demonstrates the ping-pong method of continuous buffer reading
    Sub ReadBuffer()

        'ping pong between low half and upper half
        ret = theDevice.GetStatus(Status, CurrentCount, CurrentIndex, 
MccDaq.FunctionType.AiFunction)
        If ret.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then Stop
        If ((CurrentIndex >= HALFBUFFER) And (ReadLower = True)) Then
            'read the lower half and set Upper flag to zero so that next read is upper half
            Dim theArray(BUFFERSIZE) As UShort
            ret = MccDaq.MccService.WinBufToArray(buffer, theArray, 0, HALFBUFFER)
            If ret.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then Stop

            ReadLower = False
            DisplayData(theArray, HALFBUFFER \ CHANNELCOUNT)

        ElseIf ((CurrentIndex < HALFBUFFER) And (ReadLower = False)) Then
            'read the upper half and set Upper flag to one so that next read is lower half
            Dim theArray(BUFFERSIZE) As UShort
            ret = MccDaq.MccService.WinBufToArray(buffer, theArray, HALFBUFFER,
HALFBUFFER)
            If ret.Value <> MccDaq.ErrorInfo.ErrorCode.NoErrors Then Stop

            ReadLower = True
            DisplayData(theArray, HALFBUFFER \ CHANNELCOUNT)

        End If

    End Sub
```

Measurement Computing Data Acquisition Knowledgebase
https://kb.mccdaq.com/KnowledgebaseArticle50488.aspx