How to program two USB-1608G

The example below demonstrates how to read two USB-1608G. The example C# project can be found as an attachment at the bottom.

The program assumes you have two USB-1608G (Or 1608GX or 1608GX-2AO) devices configured in InstaCal for board number 0 and 1. For simplicity, both devices were configured for the same number of channels.

Data collection uses a do...while loop that continuously reads both device buffers one half at a time. The continuous mode is set using the Scan Options enumeration Background and Continuous. Data is collected by ping-ponging between the lower half then the upper. Care should be taken that data doesn't come in so fast that the program fails to keep up. When this happens a buffer overrun occurs which is condition where unread data get overwritten. If this happens, a bigger buffer or lower sample rate is usually the fix.

Depending on your use case, you may want to synchronize both USB-1608G using the AICKO and AICKI signals. The AICKO pin is the output clock and the AICKI pin is the input clock. To synchronize the two, wire the one to the other. For instance, AICKO (board number 0) to AICKI (board number 1). Board number 0 will be the clock provider and 1 will be the receiver. The output clock is always active, but the input clock isn’t. To use the input clock, include EXTCLOCK in the AInScan Scan Options parameter when programming the receiver. For example, ScanOptions.Background | ScanOptions.ScaleData | ScanOptions.Continuous | ScanOptions.ExtClock. If you go this route, the key is to program the clock receiver first then the clock provider. The reason for this is that the receiver waits for the clock signal before starting. When you start the provider (by calling AInScan), its output clock becomes active which simultaneously start the receive

This program is a 32-bit C# Console Application. If you start a new project, you must first add a Dot Net reference to the MccDaq object. The InstaCal installation our Dot Net UL object. Adding the reference is usually accomplished by right clicking the Project [under the Project Explorer] and selecting Add Reference.

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.

```
using System;
using MccDaq;
using System.IO;
namespace AinScanBackgroundContinuousScan
{
   class Program
   {
```

```
public const int BLOCKSIZE = 256;
public const int CHANCOUNT = 2;
public const int FIRSTCHANNEL = 0;
public const int LASTCHANNEL = 1;
public const int FREQ = 256;
public const int BUFFERSIZE = BLOCKSIZE * CHANCOUNT;
public const int HALFBUFFSIZE = BUFFERSIZE / 2;

static void Main(string[] args)
{

    MccDaq.ErrorInfo RetVal;

    int Rate = FREQ;

    bool ReadLower = true;

    MccBoard daq0 = new MccDaq.MccBoard(0);
    MccBoard daq1 = new MccDaq.MccBoard(1);

    IntPtr buffer0 = MccService.ScaledWinBufAllocEx(BUFFERSIZE);
    IntPtr buffer1 = MccService.ScaledWinBufAllocEx(BUFFERSIZE);


    short[] chArray = new short[CHANCOUNT]; //configuration array for channel
numbers
    Range[] chRange = new Range[CHANCOUNT]; //configuration array for input
ranges

    //choose channel order
    chArray[0] = 0;
    chArray[1] = 1;

    //select channel range
    chRange[0] = Range.Bip10Volts;
    chRange[1] = Range.Bip10Volts;

    RetVal = daq0.ALoadQueue(chArray, chRange, CHANCOUNT);
    IsError(RetVal);

    RetVal = daq1.ALoadQueue(chArray, chRange, CHANCOUNT);
    IsError(RetVal);

    //setup the acquisiton
    RetVal = daq0.AInScan(FIRSTCHANNEL,
                          LASTCHANNEL,
```

```
                        BUFFERSIZE,
                        ref Rate,
                        Range.Bip10Volts,
                        buffer0,
                        ScanOptions.Background | ScanOptions.ScaleData |
ScanOptions.Continuous
                    );
        IsError(RetVal);

        RetVal = daq1.AInScan(FIRSTCHANNEL,
                         LASTCHANNEL,
                         BUFFERSIZE,
                         ref Rate,
                         Range.Bip10Volts,
                         buffer1,
                         ScanOptions.Background | ScanOptions.ScaleData |
ScanOptions.Continuous | ScanOptions.ExtClock
                        );
        IsError(RetVal);

        int Count = 0;
        int Index = 0;
        short daqStatus;

        double[] Data0 = new double[HALFBUFFSIZE];
        double[] Data1 = new double[HALFBUFFSIZE];
        int rows = HALFBUFFSIZE / CHANCOUNT;
        System.ConsoleKeyInfo cki = new System.ConsoleKeyInfo();

        //Loop until key press
        do
        {
           //check status on the device receiving the clock
           RetVal = daq1.GetStatus(out daqStatus, out Count, out Index,
FunctionType.AiFunction);
           if ((Index >= HALFBUFFSIZE) & ReadLower) //check for 50% more data
           {
              //get lower half of buffer - ScaledWinBufToArray returns engineering units
              RetVal = MccService.ScaledWinBufToArray(buffer0, Data0, 0,
HALFBUFFSIZE);
              IsError(RetVal);
```

```
                RetVal = MccService.ScaledWinBufToArray(buffer1, Data1, 0,
HALFBUFFSIZE);
                IsError(RetVal);

                DisplayData(Data0, Data1, rows);
                ReadLower = false; //flag that controls the next read
            }
            else if ((Index < HALFBUFFSIZE) & !ReadLower)
            {
                //get the upper half  - ScaledWinBufToArray returns engineering units
                RetVal = MccService.ScaledWinBufToArray(buffer0, Data0, HALFBUFFSIZE,
HALFBUFFSIZE);
                IsError(RetVal);
                RetVal = MccService.ScaledWinBufToArray(buffer1, Data1, HALFBUFFSIZE,
HALFBUFFSIZE);
                IsError(RetVal);

                DisplayData(Data0, Data1, rows);
                ReadLower = true;//flag that controls the next read
            }
            System.Threading.Thread.Sleep(100);

        } while (!Console.KeyAvailable);

        cki = Console.ReadKey();


        //stop the  acquisition
        RetVal = daq0.StopBackground(FunctionType.AiFunction);
        RetVal = daq1.StopBackground(FunctionType.AiFunction);

        //free up memory
        MccService.WinBufFreeEx(buffer0);
        MccService.WinBufFreeEx(buffer1);

        WaitForKey();

    }

    //error check routine
    public static int IsError(ErrorInfo e)
    {
        if (e.Value != 0)
        {
            Console.WriteLine(e.Message);
```

```
        return 1;
    }
    return 0;
}

//routine to wait for a key press
public static void WaitForKey()
{
    Console.WriteLine("\nPress <SpaceBar> to continue...");

    System.ConsoleKeyInfo cki;
    do
    {
        cki = Console.ReadKey();
    } while (cki.Key != ConsoleKey.Spacebar);
}

//This routine prints the data to the screen
public static void DisplayData(double[] buf0, double[] buf1, int rows)
{
    //Writes data to screen and to file
    int i = 0;

    for (int row = 0; row < rows; row++)
    {
        //print daq0 array
        for (int c = 0; c < CHANCOUNT; c++)
        {
            Console.Write("{0}\t", buf0[i].ToString("0.0000").PadLeft(10));
            i++;
        }
        //print daq1 array
        i = 0;
        for (int c = 0; c < CHANCOUNT; c++)
        {
            Console.Write("{0}\t", buf1[i].ToString("0.0000").PadLeft(10));
            i++;
        }
        Console.Write("\r\n");
    }
}
```

```
    }
}
```

Measurement Computing Data Acquisition Knowledgebase
https://kb.mccdaq.com/KnowledgebaseArticle50798.aspx