# How to configure USB-QUAD08 channel for quadrature encoder input

The example program below demonstrates (using C++) how to configure a USB-QUAD08 channel for quadrature encoder input. The key is to set the cbCConfigScan Mode parameter to **ENCODER|ENCODER_MODE_X1|ENCODER_MODE_BIT_32** (for X1 mode). You also have to specify the CTR32BIT in the Options for the cbCInScan function. For example, **BACKGROUND|CONTINUOUS|CTR32BIT** for continuous operation.

It also demonstrates device discovery which doesn't require the user to run InstaCal. InstaCal must still be installed because doing so installs the device drivers. At the bottom of this article you will find a Zip file that contains the Visual Studio 2008 project.

```
// VC_2008_USB_QUAD08_Encoder.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"


/* Include files */
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include "cbw.h"

#define RATE 100
#define ChanCount 1
#define COUNT 100
#define MAXNUMDEVS 100

void main ()
  {
  /* Variable Declarations */
  int i=0;
  long curCount=0;
  long curIndex=0;

  int ULStat = 0;

  long Rate = RATE;
  short Status;
  long halfbuf = COUNT/2;
  bool NextReadUpper = false;


  int numberOfDevices = MAXNUMDEVS;
```

```
  DaqDeviceDescriptor inventory[MAXNUMDEVS];
  DaqDeviceDescriptor DeviceDescriptor;

  int BoardNum = -1;


  float Rev = (float)CURRENTREVNUM;
  ULStat = cbDeclareRevision(&Rev);
  cbErrHandling(PRINTALL, STOPALL);

  printf ("Demonstration of cbConfigScan() in BACKGROUND mode\n\n");

//Ignore InstaCal device discovery
cbIgnoreInstaCal();

//locate USB devices
ULStat = cbGetDaqDeviceInventory(USB_IFC, inventory, &numberOfDevices);
for( i = 0; i < numberOfDevices; i++)
{
DeviceDescriptor = inventory[i];

//Product ID for USB-QUAD08 = 0xCA
//Product IDs can be found in ULProps.txt located in
// C:\Program Files (x86)\Measurement Computing\DAQ
if(DeviceDescriptor.ProductID == 0xCA)
{
BoardNum = i;
ULStat = cbCreateDaqDevice(BoardNum, DeviceDescriptor);
printf("Device Name: %s\n", DeviceDescriptor.ProductName);
break;
}
}

  if(BoardNum < 0)
  {
  printf("USB device not found...press any key to exit\n");
  getch();
  return;
  }

  //allocate buffer
  HANDLE MemHandle = 0;
  MemHandle = cbWinBufAlloc32(COUNT);
```

```
  unsigned long *DataArray = (unsigned long*)MemHandle;

  ULStat = cbCConfigScan(BoardNum,
  0,
  ENCODER|ENCODER_MODE_X1|ENCODER_MODE_BIT_32,
  CTR_DEBOUNCE500ns,
  CTR_TRIGGER_AFTER_STABLE,
  CTR_RISING_EDGE,
  CTR_TICK208PT3ns,
  0);
  if(ULStat != 0)
printf("Error Code %d\n",ULStat);
  unsigned int Options = BACKGROUND|CONTINUOUS|CTR32BIT;

  //Read Channel 0 using options BACKGROUND|CONTINUOUS|CTR32BIT;
  ULStat = cbCInScan(BoardNum,
0,
0,
COUNT,
&Rate,
MemHandle,
Options);
if(ULStat != 0)
printf("Error Code %d\n",ULStat);

   while(!_kbhit())
   {
     //this loop reads the low half of the buffer then the upper half continuously.
     //it uses a NextReadUpper flag so that each half of the buffer is read once

      ULStat = cbGetStatus(BoardNum,&Status,&curCount,&curIndex,CTRFUNCTION);

      if(ULStat != 0){
      printf("Error Code %d\n",ULStat);
      break;
      }
      if((curIndex > halfbuf) && (NextReadUpper == false))
      {
      NextReadUpper = true;
      for (i = 0; i< halfbuf; i++)
      printf ("%ld\n",DataArray[i]);
```

```
        }
        else if((curIndex < halfbuf) && (NextReadUpper == true))
        {
        NextReadUpper = false;
        for (i = 0; i< halfbuf; i++)
        printf ("%ld\n",DataArray[i]);

        }
        Sleep(10);
    }


cbStopBackground(BoardNum,CTRFUNCTION);
cbWinBufFree(MemHandle);
cbReleaseDaqDevice(BoardNum);

printf("Completed...press any key to exit\n");
getch();
}
```