BNCS Class Library

External GRD Interface Class

Written by Tim Alden, Ed Delauter and Dave Yates

Overview

This class provides all the required connectivity between your driver application and a BNCS GRD.

The class definition is as follows:

```
#pragma once
#include <windows.h>
#include <bncsdef.h>

#ifndef GTHUNKS_EXPORTS
#undef GTHUNKS_API
#define GTHUNKS_API
#endif

class GTHUNKS_API cGRD
{
        HWND hWndSpawn;
        PULONG txcount;
        PULONG rxcount;
        ULONG deftx,defrx;
        static LRESULT WINAPI CallbackProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
        LRESULT (*Cxptevent)(cGRD *cG, UINT iDevice, UINT iSrc, UINT iDest, LPCSTR szMask);
        HWND hWndGRD;
protected:
        virtual LRESULT xptevent(cGRD *cG, UINT iDevice, UINT iSrc, UINT iDest, LPCSTR szMask);
public:
        UINT iGRD;
        int iStatus;

        cGRD(void);
        ~cGRD(void);
        int connect(UINT iExtGRD);
        void setcounters(PULONG lpTX,PULONG lpRX);
        int getsrc(int iDest);
        bool setdest(int iSrc, int iDest);
        void incrx();
        void notify( LRESULT(*in)(cGRD *,UINT iDevice, UINT iSrc, UINT iDest, LPCSTR szMask));
};
```

Required files

The class is contained in the library **cGRD.lib** and a debug version in **cGRDdb.lib**

The header files **<cGRD.h>** and **<bncsdef.h>** should be referenced by your application.

The following constants are defined in the **<bncsdef.h>** header file, and are valid values for the iStatus member variable:

```
#define DISCONNECTED              3
#define INVALID_DRIVERNUM         4
#define CANT_FIND_GRD             5
#define CANT_REGISTER_CLASSWND    9
#define XPT_CHANGE                15
#define XPT_LOCK                  16
```

Usage

There are two methods of getting destination change information:

- C callback
- virtual function

C Callback

Create an instance to a cGRD class (don't' forget to delete it when you exit….)

```
        cGRD *g;            // somewhere in your header files….

        g =  new cGRD;
        g->notify( notify );        // set the notify function
        g->connect( 100 );          // connect to the GRD

        int d = g->getsrc( 1 );     // get source on dest 1
        g->setdest( 1, 1 );         // set dest 1 to source 1


LRESULT notify (cGRD *cG, UINT iDevice, UINT iSrc, UINT iDest, LPCSTR szMask)
{
        switch( cG->iStatus )
        {
                case XPT_CHANGE:
                        qDebug( " crosspoint made device %d, %d to %d, mask %s",
                                                                iDevice,
                                                                iSrc,
                                                                iDest,
                                                                szMask );
                        break;
        }
        return 0;
}
```

Virtual function

Derive your class from cGRD and then reimpliment the virtual function

virtual LRESULT xptevent(cGRD *cG, UINT iDevice, UINT iSrc, UINT iDest, LPCSTR szMask);

e.g.

```
#include “cGRD.h”

class myclass : public cGRD
{
public:
        myclass();
        virtual LRESULT xptevent(cGRD *cG, UINT iDevice, UINT iSrc,UINT iDest, LPCSTR szMask);
}
```

See the notify function above for an example of how to implement the xptevent function.