***ROUGH DRAFT

DLL User's Guide for DDC 264 EVM

July 24, 2012

****

Please note this is a first draft and is provided AS IS. Due to limited bandwidth, we cannot provide any further support for creating customized GUIs for our marketing EVMs

****

This document is a DLL user's guide for interfacing with the DDC264 EVM.

1. **DLL API**
   a. **Open**
      Finds device connected to PC. After detection, opens connection. Creates valid handle to device driver but future access is accomplished by other methods. Must be called to access device functionality.
      i. **Declaration**
         //Creates instance of USB device
         CCyUSBDevice *USBDevice = new CCyUSBDevice(NULL);
         if (USBDevice->Open(0)) //To check if USB device is open

   b. **Close**
      Closes communication
      i. **Declaration**
         //If this is how instance of USB Device is created...
         CCyUSBDevice *USBDevice = new CCyUSBDevice(NULL);
         USBDevice->Close();

   c. **Read Device Descriptors**
      This function reads device descriptors from the Cypress USB chip
      i. **Function Declaration**
         int __stdcall ReadDeviceDescriptors(int *USBdevCount, int *bLengthPass, int *bDescriptorTypePass, long *bcdUSBPass, int *bDeviceClass, int *bDeviceSubClass, int *bDeviceProtocol, int *bMaxPacketSize0, long *idVendor, long *idProduct, long *bcdDevice, int *iManufacturer, int *iProduct, int *iSerialNumber, int *bNumConfigurations);
      ii. **Description**
         Creates an instance of USB Device class to get the device descriptors. This can be found in the Cypress API header file. Returns arrays of values, one set of values per device detected. The user can use own programming environment to select which device to use. Returns number of devices.

d. Read Interface Descriptors

Reads interface descriptors from Cypress USB chip

i. Function Declaration

int __stdcall ReadInterfaceDescriptors(int *USBdev, int *bLengthPass, int *bDescriptorTypePass, int *bInterfaceNumberPass, int *bAlternateSettingPass, short *bNumEndpointsPass, int *bInterfaceClassPass, int *bInterfaceSubClassPass, int *bInterfaceProtocolPass, int *iInterfacePass);

ii. Description

Creates an instance of USB Device class to get the interface descriptors. This can be found in the Cypress API header file. Returns -1 if no USB devices are open. Returns 0 upon completion of function.

e. Read Endpoint Descriptors

Reads endpoint descriptors from Cypress USB chip

i. Function Declaration

int __stdcall ReadEndPointDescriptors(int *USBdev, int *bLengthPass, int *bDescriptorTypePass, long *bEndpointAddressPass, long *bmAttributesPass, int *wMaxPacketSizePass, int *bIntervalPass);

ii. Description

Creates an instance of USB Device class to get the number of endpoints exposed by interface. Reads the fields of each descriptor. Returns -1 if no USB devices are open. Returns 0 upon completion of function.

f. Transfer Data Out

Writes string of bytes to USB. Primary way to send data from PC to USB

i. Function Declaration

int __stdcall XferINTDataOut(int *USBdev, int *Data01, int *Data02, int *Data03, int *Data04);

ii. Description

Creates an instance of USB Device class. If the bulk out endpoint is reached, data (int) is transferred. Returns -1 if no USB devices are open. Returns 0 upon completion of function.

g. Transfer Data Out2

Writes string of bytes to USB. Primary way to send data from PC to USB

i. Function Declaration

int __stdcall XferDataOut(int *USBdev, unsigned char * Data, long *DataLength);

ii. Description

Creates an instance of USB Device class. If the bulk out endpoint is reached, data (char) is transferred. Returns -1 if no USB devices are open. Returns 0 upon completion of function.

h. Transfer Data In

Writes string of bytes to PC. Primary way to send data from USB to PC

i. Function Declaration

int __stdcall XferINTDataIn(int *USBdev, int *INTData, long *DataLength);

ii. Description

Creates an instance of USB Device class. If the bulk in endpoint is reached, data (int) is transferred. Returns -10 if USB endpoint cannot be reached. Returns -2 if cannot open USB device.

i. Transfer Data In2

Writes string of bytes to PC. Primary way to send data from USB to PC

i. Function Declaration

int __stdcall XferDataIn(int *USBdev, unsigned char *Data, long *DataLength);

ii. Description

Creates an instance of USB Device class. If the bulk in endpoint is reached, data (char) is transferred. Returns -10 if USB endpoint cannot be reached. Returns -2 if cannot open USB device.

j. Capture Data

Captures data and stores it. Primary way to send data from USB to computer

i. Function Declaration

long __stdcall CaptureData(int *USBdev, long *Data, long *DataCount, unsigned char *string0);

ii. Description

Creates an instance of USB Device class. Shifts out 0x10FF to start conversion. If bulk in end point is reached, sets size of data transferred then transfers data. Shifts out 0x1000 to end conversion. Returns -2 if can't open the USB device. Returns packets received.

k. FPGA registers

Writes to FPGA registers and reads back

i. Function Declaration

long __stdcall WriteFPGARegsC(int *USBdev, long DUTSelect, long * RegsIn, long * RegsOut, long * RegEnable);

ii. Description

Creates an instance of USB Device class. Writes, reads back FPGA register content, then resets CONV. Returns -1 if no USB devices are open. Returns 0 upon completion of function.

2. **Calling DLL function**

   i. Function Declaration (in VB)

   Public Declare Function ReadDeviceDescriptors Lib "USB_IO_for_VB6.dll" (ByRef Single_Element_Array_USB_Device As Short, ByRef Array_bLength As Short, ByRef Array_bDescriptorType As Short, ByRef Array_bcdUSB As Integer, ByRef Array_bDeviceClass As Short, ByRef Array_bDeviceSubClass As Short, ByRef Array_bDeviceProtocol As Short, ByRef Array_bMaxPacketSize0 As Short, ByRef Array_idVendor As Integer, ByRef Array_idProduct As Integer, ByRef Array_bcdDevice As Integer, ByRef Array_iManufacturer As Short, ByRef Array_iProduct As Short, ByRef Array_iSerialNumber As Short, ByRef Array_bNumConfigurations As Short) As Short

   ii. Description

   Shows how to call DLL functions from VB