Just a few brief notes about the WinUSB implementation in the HPSDR DLL for Winrad.

The functions implemented are only :

Initialize_HPSDR(...)
ReadFromHPSDR(...)
ReadNBFromHPSDR(...)
ReadRawFromHPSDR(...)
WriteToHPSDR(...)
WriteNBToHPSDR(...)

which were what I needed for the DLL. Here following the synopsis for each one.

int Initialize_HPSDR(HPSDR_WUSB_INFO& hpw)

It accepts as input a reference to a structure like this :

```
   struct  HPSDR_WUSB_INFO
   {
            HANDLE deviceHandle;
            WINUSB_INTERFACE_HANDLE winUSBHandle;
            UCHAR  deviceSpeed;
            UCHAR  bulkInPipe;
            UCHAR  rawInPipe;
            UCHAR  bulkOutPipe;
            int    FX2linesloaded;
            int    FPGAbytesloaded;
            int    numEndP;
            char*  FX2fname;
            char*  FPGAfname;
           char   lastMsg[256];
   };
```

The fields FX2name and FPGAname must be filled before invoking Initializing_HPSDR
with the names of the files containing the FX2 and the Mercury FPGA firmware,
The other fields are then filled by the initialization code, which also loads the
two firmware files into the relative chips. The return code of Initialize_HPSDR
is an integer value containing the number of end points found (which for HPSDR
currently are three) or -1 in case of error.

unsigned long ReadFromHPSDR(UCHAR *buf, int size)

Reads from EP6 a buffer of <size> bytes (Multiple of 512). Do not return until the read is completed.

bool ReadNBFromHPSDR(UCHAR *buf, int size, OVERLAPPED& olap)

Reads from EP6 a buffer of <size> bytes (Multiple of 512). Returns immediately, and the olap handle
of the OVERLAPPED structure (defined in winbase.h) is set when the read is completed.

bool ReadRawFromHPSDR(UCHAR *buf)

Reads from EP4 a buffer of 4096 bytes. Do not return until the read is completed.

bool WriteToHPSDR(UCHAR *buf, int size)

Writes to EP2 a buffer of <size> bytes (Multiple of 512). Do not return until the completion of the write.

bool WriteNBToHPSDR(UCHAR *buf, int size, OVERLAPPED& olap)

Writes to EP2 a buffer of <size> bytes (Multiple of 512). Returns immediately, and the olap handle of the OVERLAPPED structure (defined in winbase.h) is set when the write is completed.

Experimentally I saw that using a buffer size for reads and writes of 1024 bytes (512 x 2) has the positive effect of a very low latency, but at the expenses of an icreased CPU time, especially the kernel CPU time, as probably there are a greater number of context switchings between ring 0 and ring 3.
Using as size 2048 reduces the CPU load, but increases a little the latency time.
I compiled and distributed two DLLs, one with a size of 1024, and the other with 2048, to be used according to the speed of your CPU.

Please note that you must also install the Microsoft Driver Development KIT (DDK), as you will need the include files

..\inc\api
..\inc\ddk

and also the library

..\lib\wxp\i386

from that development kit. The DDK is downloadable free of charge from the Microsoft site.