NETL@BUSB

USB from Ring 3

By Markus Montkowski

NETL@BUSB

Reasons for a Ring 3 library

- More people think they can write an Application than they can write a driver
- Not all devices need a driver for system integration
- Help testing in driver development

NETL@BUSB

Design goals

- Easy to use
- Support all transfer types
- Usable by multiple Applications
- Provide information about devices
- PNP Support

NETL@BUSB

Feature List for USBCALLS

- Transfers Types
  - Control
  - Bulk
  - Isochronous
  - Interrupt
- Notification about Device de/attach
- Number of Devices
- Device Reports
- REXX interface

NETL@BUSB

System Information

UsbQueryNumberDevices

ULONG *pulNumDev

UsbQueryDeviceReport

ULONG ulDevNumber

ULONG *ulBufLen

CHAR *pData

NETL@BUSB

PNP Notifications

UsbRegisterChangeNotification

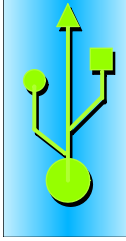
PUSBNOTIFY pNotifyID

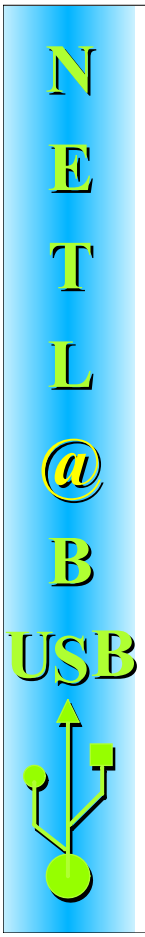
HEV hDeviceAdded

HEV hDeviceRemoved

UsbDeregisterNotification

USBNOTIFY NotifyID





PNP Notifications cont.

UsbRegisterDeviceNotification

- PUSBNOTIFY pNotifyID
- HEV hDeviceAdded
- HEV hDeviceRemoved
- USHORT usVendor
- SHORT usProduct
- SHORT usBCDVersion

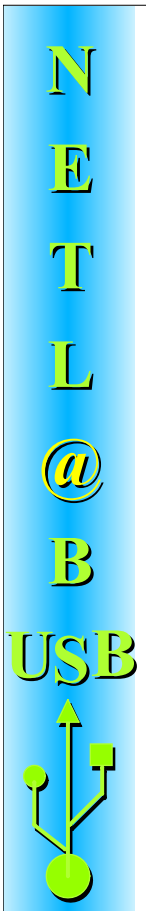
Open/ Close Calls

UsbOpen

- PUSBHANDLE pHandle
- USHORT usVendor
- USHORT usProduct
- USHORT usBCDDevice
- USHORT usEnumDevice

UsbClose

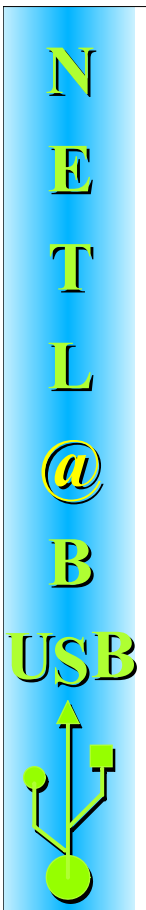
- USBHANDLE Handle



Control Transfers

UsbCtrlMessage

- USBHANDLE Handle
- UCHAR ucRequestType
- UCHAR ucRequest
- USHORT usValue
- USHORT usIndex
- USHORT usLength
- UCHAR *pData
- ULONG ulTimeout



Standard Control Transfers

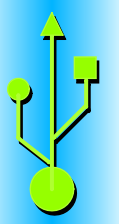
- UsbDeviceSetConfiguration
- UsbGetStringDescriptor
- ...

NETL@BUSB

Bulk Transfers

UsbBulkRead

- USBHANDLE Handle
- UCHAR Endpoint
- UCHAR Interface
- USHORT *usNumBytes
- UCHAR *pData
- ULONG ulTimeout

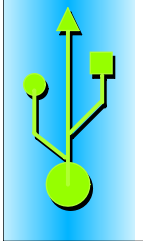


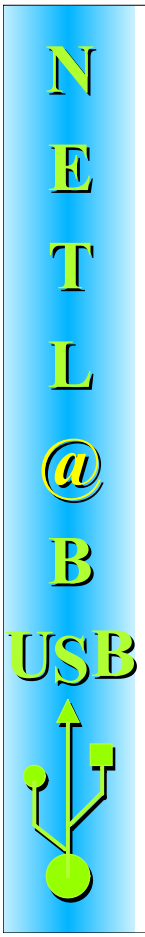
NETL@BUSB

Bulk Transfers cont.

UsbBulkWrite

- USBHANDLE Handle
- CHAR Endpoint
- CHAR Interface
- SHORT usNumBytes
- CHAR *pData
- ONG ulTimeout





Interrupt Transfers

UsbIrqStart

- USBHANDLE Handle
- UCHAR Endpoint
- UCHAR Interface
- USHORT usNumBytes
- UCHAR *pData
- PHEV pHevModified

UsbIrqStop

- USBHANDLE Handle
- HEV HevModified

Isynchronous Transfers

UsbIsoStart

- USBHANDLE Handle
- CHAR Endpoint
- CHAR Interface
- ISOHANDLE *phIso

UsbIsoStop

- SOHANDLE hIso

Iso Ringbuffer Access

UsbIsoDequeue

ISOHANDLE hIso

UCHAR * pBuffer

ULONG ulNumBytes

UsbIsoEnqueue

ISOHANDLE hIso

const UCHAR * pBuffer

ULONG ulNumBytes

Iso Ringbuffer Information

UsbIsoPeekQueue

ISOHANDLE hIso

UCHAR * pByte

ULONG ulOffset

UsbIsoGetLength

ISOHANDLE hIso

ULONG *pulLength

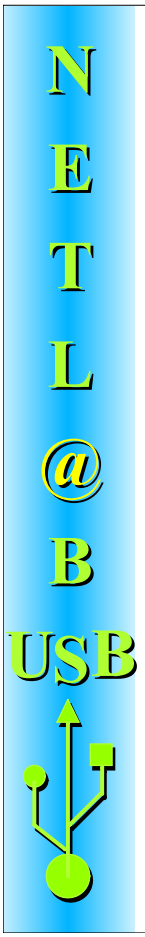
NETL@BUSB

REXX Interface

- Same Function Set
- Loaded with UsbLoadFuncs
- Rx Prefix to name RxUsbQueryNumberDevices etc.

NETL@BUSB

Example USB Radio

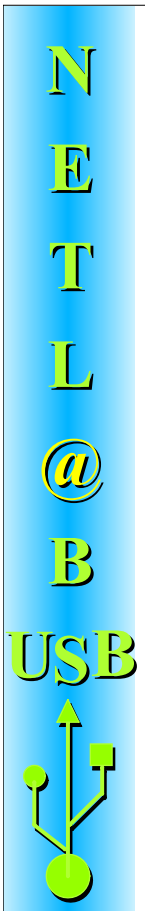


Register for Notification

```
...
rc =DosCreateEventSem(NULL,&pRadio->hRadioPluged,0,FALSE);
if (!rc)
({
  rc = DosCreateEventSem(NULL,&pRadio->hRadioUnpluged,0,FALSE);
  if (rc)
    DosCloseEventSem(pRadio->hRadioPluged);
  else
  {
    SEMRECORD aSems[2];
    aSems[0].hsemCur = (HSEM)pRadio->hRadioPluged;
    aSems[0].ulUser  = 0;
    aSems[1].hsemCur = (HSEM)pRadio->hRadioUnpluged;
    aSems[1].ulUser  = 1;
    rc = DosCreateMuxWaitSem(NULL, &pRadio->hMuxWait, 2,
                             (PSEMRECORD)&aSems, DCMW_WAIT_ANY);
    if (!rc)
    {
      pRadio->hMonThread = _beginthread(NotifyThread,NULL,8192,pRadio);
      rc = g_USBFuncs.pUsbRegisterDeviceNotification( &pRadio->NotifyID,
                                                      pRadio->hRadioPluged,
                                                      pRadio->hRadioUnpluged,
                                                      0x04b4, 0x1002,
                                                      USB_ANY_PRODUCTVERSION);
...
```

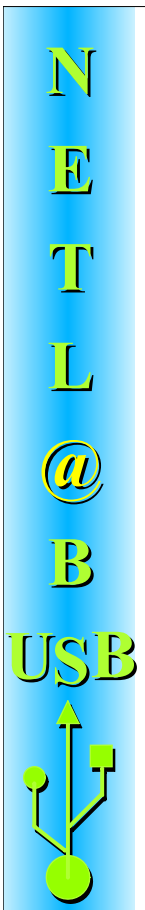

Notification Thread

```
void _Optlink NotifyThread(void* args)
{
  PRADIOPRIVATE pRadio = (PRADIOPRIVATE)args;
  ULONG ulWhich;
  ULONG ulCnt;
  pRadio->ulState |= RADIO_USBCHECK;
  while (pRadio->ulState  & RADIO_USBCHECK)
  {
    DosWaitMuxWaitSem(pRadio->hMuxWait,SEM_INDEFINITE_WAIT,&ulWhich);
    if (!pRadio->ulState)
      break;
    if (ulWhich)
    {
      ProcessRadioUnpluged(pRadio);
      DosResetEventSem(pRadio->hRadioPluged, &ulCnt);
    }
    else
    {
      ProcessRadioPluged(pRadio);
      DosResetEventSem(pRadio->hRadioPluged, &ulCnt);
    }
  }
  g_USBFuncs.pUsbDeregisterNotification(pRadio->NotifyID);
  DosCloseMutexSem(pRadio->hMuxWait);
  DosCloseEventSem(pRadio->hRadioPluged);
  DosCloseEventSem(pRadio->hRadioUnpluged);
}
```



ProcessRadioPlugged

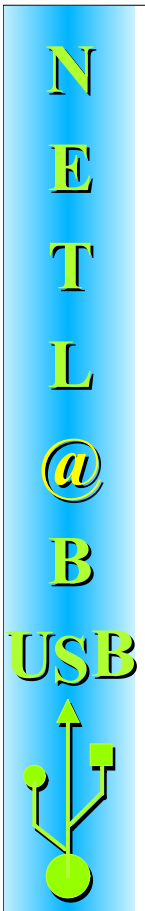
```
void ProcessRadioPluged(PRADIOPRIVATE pRadio)
{
  APIRET rc;
  if (!pRadio->ulNumRadios)
  {
    if (pRadio->Setup.fOnOnAttach)
    {
      DosBeep(220,50);
      DosBeep(440,50);
      pRadio->Setup.fTurnOn = TRUE;
      DosPostEventSem(pRadio->hEvtFreqChange);
    }
  }
  pRadio->ulNumRadios++;
  WinInvalidateRegion(pRadio->pWidget->hwndWidget, NULL, TRUE);
}
```

Frequency Thread

```
void _Optlink FreqThread(void* args)
{
  PRADIOPRIVATE pRadio = (PRADIOPRIVATE)args;
  ULONG ulCnt;
  while (pRadio->ulState  & RADIO_USBCHECK)
  {
    DosWaitEventSem(pRadio->hEvtFreqChange, SEM_INDEFINITE_WAIT);
    if (!pRadio->ulState)
      break;
    do
    {
      ...

      if(pRadio->ulNumRadios &&
         pRadio->Setup.fTurnOn)
      {
        XSTRING strSetup;
        RadioSetFreq(pRadio->Setup.ulCurrentFreq);
        WinInvalidateRect(pRadio->pWidget->hwndWidget,NULL, FALSE);
        DosSleep(80);
        if(RadioGetStereo())
        {
          pRadio->ulState |= RADIO_STEREO;
          pRadio->ulState &= ~RADIO_SCAN;  // End Scanning if we tuned in.
        }
        ...
```

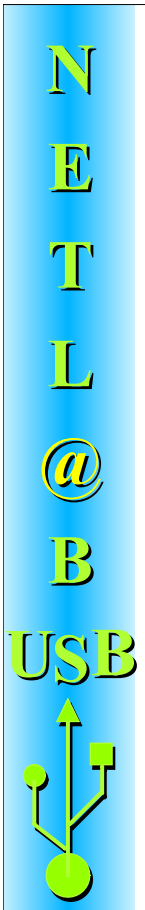


RadioSetFreq

```
APIRET RadioSetFreq(ULONG ulNewFreq)
{
  double dFreq;
  ULONG ulFreq;
  USBHANDLE Handle;
  APIRET rc;
  UCHAR ucData[8];
  dFreq = ulNewFreq / 100.0;
  ulFreq = ((dFreq+10.7)*80);

  rc = g_USBFuncs.pUsbOpen( &Handle,
                            0x04b4,
                            0x1002,
                            USB_ANY_PRODUCTVERSION,
                            USB_OPEN_FIRST_UNUSED);
  if (!rc)
  {
    rc = g_USBFuncs.pUsbCtrlMessage( Handle,
                                     0xC0, 0x01,
                                     ulFreq>>8,ulFreq,
                                     1,(UCHAR*)&ucData,
                                     0);

    g_USBFuncs.pUsbClose(Handle);
  }
  return(rc);
}
```

RadioGetStereo

```
BOOL RadioGetStereo()
{
  USBHANDLE Handle;
  APIRET rc;
  UCHAR ucData[8];
  BOOL fStereo = FALSE;
  rc = g_USBFuncs.pUsbOpen( &Handle,
                            0x04b4,
                            0x1002,
                            USB_ANY_PRODUCTVERSION,
                            USB_OPEN_FIRST_UNUSED);
  if (!rc)
  {
    rc = g_USBFuncs.pUsbCtrlMessage( Handle,
                                     0xC0, 0x00,
                                     0, 0x00,
                                     1,(UCHAR*)&ucData,
                                     0);
    fStereo = (ucData[0]&0x01)==0x00;

    g_USBFuncs.pUsbClose(Handle);
  }
  return(fStereo);
}
```

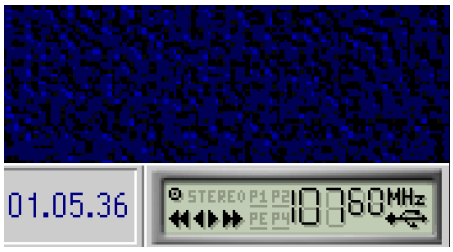
NETL@BUSB

RadioPower

```
APIRET RadioPower(BOOL fTurnOn)
{
  USBHANDLE Handle;
  APIRET rc;
  UCHAR ucData[8];
  BOOL fStereo = FALSE;
  rc = g_USBFuncs.pUsbOpen( &Handle,
                            0x04b4,
                            0x1002,
                            USB_ANY_PRODUCTVERSION,
                            USB_OPEN_FIRST_UNUSED);
  if (!rc)
  {
    rc = g_USBFuncs.pUsbCtrlMessage( Handle,
                                     0xC0, 0x02,
                                     fTurnOn?1:0, 0,
                                     1,(UCHAR*)&ucData,
                                     0);
    g_USBFuncs.pUsbClose(Handle);
  }
  return(rc);
}
```


NETL@BUSB

Result: XCenter Radio Widget



NETL@B USB

Useful information links

- General info docs etc ***www.usb.org***
- USB device information ***www.linux-usb.org***
- Sources for many linux USB drivers
  www.sourceforge.net
- The OS/2 DDK with sources of USB drivers
  service.boulder.ibm.com/ddk/
- OS/2 USB Project at ***www.netlabs.org***
  - **CVS CVSROOT=:pserver:guest@www.netlabs.org:e:/netlabs.cvs/usb**
  - **Contact usbguy@netlabs.org**