Programming samples for RFOptic's USB API

1 General

RFOptic USB interface is based on the built in generic HID device driver present under Linux and Microsoft Windows operating systems. Therefor no installation is necessary to interface with the RFoF products. The command set API is product specific and it is documented in a separate document available upon request. Many of the interface commands are common but some are specific to particular products. The foregoing is a set of sample codes used to interface with the RFOptic's RFoF products.

Several steps must be followed to establish a USB interface to a device. First the appropriate device is identified by its USB Product ID and Vendor ID. Next the device can be opened for communication and interrogated for its Part and serial numbers. Once the correct device is identified, command packets can be sent to the device and responses obtained after device processing time is complete.

2 C/C++ USB interface code samples

These samples or similar code is used in RFOptic product control software. The USB identifiers are:

#define VID	0x04D8	
#define OLDRFoF_PROG_PID	0x003C	// used in older hardware revisions
#define RFoF_HF_PID	0xEEC7	// used in RFoF HSFDR and RFoF HF or ODLs
#define RFoF_PROG_PID	0xEEBF	// used in RFoF programmable series

2.1 Identify the desired USB device from the hive

```
libusb_device **devs;           // pointer to pointer of devices, used to retrieve a list of devices
libusb_context *ctx = NULL;     // a libusb session
libusb_device_descriptor desc;
GlobalLogfileText(3, "\nDiscovery of connected USB devices");

// discovery of connected RFoF devices
iRes = libusb_init(&ctx);           //initialize a library session
if (iRes < 0)
        GlobalLogfileText(4, "\nUSB init context error %d\n"); //there was an error
iCount = libusb_get_device_list(ctx, &devs); //get the list of devices

for (int i = 0; i < iCount; i++)
{
        iRes = libusb_get_device_descriptor(devs[i], &desc);
        if (iRes < 0)
                GlobalLogfileText(4, "failed to get device descriptor\n");
        else if (desc.idVendor == VID)
        {
                if (desc.idProduct == RFoF_HF_PID) // RFoF HF/ODL
                {
                        getDevicePath(devs[i], path);
                        OpenHIDdevice(devs[i])); // open and communicate with the unit
        ....
        }
}
```


Phone: 972-76-5400180• US: 1-708-RFOPTIC,
21 Yagia Kapaim, Petach Tikva 4913020 • Israel
www.rfoptic.com

2.2 Open and close a device

It is suggested to keep the device open throughout the control session because under some circumstances the OS will take control of the device and it will be unavailable to the control program until control of the device is returned to the control program.

```
int	CHIDdevice::OpenHIDdevice(libusb_device* dev)
{
	char path[15];
	getDevicePath(dev, path);
	if (libusb_open(dev, &m_hDevice))
	{
		printf(">>> %s - Cannot Open Device", path);
		if (m_hDevice)
			libusb_close(m_hDevice);
		m_hDevice = NULL;
		return 0;
	}
	else
	{
		if (libusb_kernel_driver_active(m_hDevice, 0) == 1)
		{
			//find out if kernel driver is attached
			if (libusb_detach_kernel_driver(m_hDevice, 0) != 0) // detach it
			{
				printf(">>>Cannot Detach Kernel Driver\n");
				if (m_hDevice)
				{
					libusb_close(m_hDevice);
					m_hDevice = NULL;
				}
				return 0;
			}
		}
		if (libusb_claim_interface(m_hDevice, 0) < 0)		// claim interface 0
		{
			printf(">>> %s - Cannot Claim Interface\n", path);
			if (m_hDevice)
			{
				libusb_close(m_hDevice);
				m_hDevice = NULL;
			}
			return 0;
		}
		else
			return 1;	// success
	}
}

void	CloseHIDdevice()
{
	if (m_hDevice)
	{
		libusb_close(m_hDevice);
		m_hDevice = NULL;
	}
}
```


Phone: 972-76-5400180• US: 1-708-RFOPTIC,
21 Yagia Kapaim, Petach Tikva 4913020 • Israel
www.rfoptic.com

2.3 Communication; send a command packet and receive a response

All communication is performed in Packets of 64/65 bits depending on the OS as shown. The packet should be defined as a structure that is packed on a byte level using compiler directives.

```
#define USB_PACKET_SIZE 64
#pragma pack()
struct USBpacket
{
#ifndef MS_WINDOWS
      unsigned char report;
#endif
      unsigned char command;
      unsigned char contents[USB_PACKET_SIZE-1];
 };

HIDerrorCode SendReceiveCmd(HANDLE dev_handle, unsigned char *pData, int size, int iUSBdelay)
{
int iBytes; //used to find out how many bytes were written
int iRes, i;
Cstring str;

// The RFoF module out endpoint is 1
iRes = libusb_bulk_transfer(dev_handle, (1 | LIBUSB_ENDPOINT_OUT), pData, size, &iBytes, 0);
if (iRes)
{
            str.Format("USB Write: %s", libusb_error_name(iRes));
            GlobalLogfileText(5, str);
            return eFail;
}
if (iRes == 0 && iBytes == size) //we wrote the size bytes successfully
{
            i = 5;      // just in case data is not ready – this almost never happens
            while (i)
            {
                        usleep(iUSBdelay*1000);  // wait for device to respond 5ms to 10ms

                        // The RFoF module in endpoint is 1
                        iRes = libusb_bulk_transfer(dev_handle, (1 | LIBUSB_ENDPOINT_IN), pData, size, &iBytes, 0);
                        if (iRes == 0 && iBytes == size)
                                    break;
                        str.Format("USB Read: try %d, &s", i--, libusb_error_name(iRes));
                        GlobalLogfileText(5, str);
                        }
                        if (iRes != 0 || iBytes != size)
                                    return eFail;                    // read error
            }
            else
            {
                        str.Format("USB Write: %s", libusb_error_name(iRes));
                        GlobalLogfileText(5, str);
                        return eFail;
            }
            return eSuccess;
}
```


Phone: 972-76-5400180• US: 1-708-RFOPTIC,
21 Yagia Kapaim, Petach Tikva 4913020 • Israel
www.rfoptic.com

2.4 Sample Send and Receive of the Identify command

```
HIDerrorCode CRFoFmodule::identify()
{
	HIDerrorCode result = eFail;
	USBpacket usbPacket;

	// send identify command
	CString str;
	GlobalLogfileText(0, _T("Identify command - "));

	//initialize all bytes clear, so unused pad bytes are = 0x00.
	memset((void*)&usbPacket, 0x00, sizeof(usbPacket));
	usbPacket.command = CMD_IDENTIFY;
	result = sendReceiveCommand(&usbPacket);
	if (result != eSuccess)
	{
		GlobalLogfileText(0, _T("Failed to receive acknowledgement from the module\n"));
		return eFail;
	}
	// Data from unit is in the usbPacket see API document for details
	GlobalLogfileText(0, _T("Succeedded\n"));
	return eSuccess;
}
```

2.5 Sample of Command and Response packet data

> 00 34 00 00 00