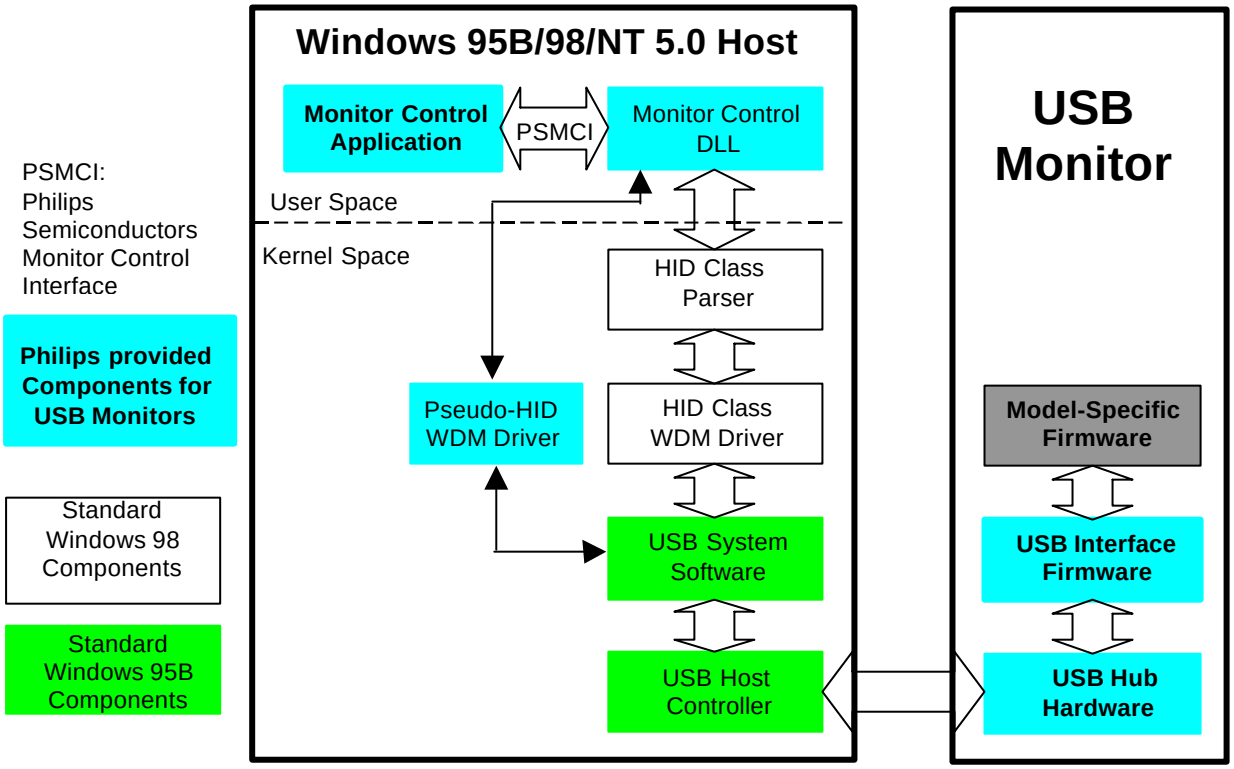
USB Monitor Control

Introduction

The components needed to implement USB monitor control is given in the figure below. They include software running on the host system to hardware and firmware running on the USB monitors.



Monitor Control Software Suite

Microsoft will release HID drivers in the Windows 98 (Memphis). To enable monitor makers to ship USB monitor control in Windows 95B (OSR2.1), Philips Semiconductors has provided a complete monitor control software suite. This includes:

1. The Monitor Control Application. This is a GUI (Graphical User Interface) sample to enable users to do simple monitor control. This applet talks to the underlying layers through the PSMCI (Philips Semiconductors Monitor Control Interface).
2. PSMCI (Philips Semiconductors Monitor Control Interface). This is a user friendly API (Application Programming Interface) provided by the Philips Monitor Control DLL (Dynamic Link Library) and driver. This interface allows monitor makers to write their own fancy applets and provide product differentiation.
3. Monitor Control DLL (Dynamic Link Library). This DLL takes care of the details of HID data structures and converts it to the user friendly PSMCI. With this DLL, the programmers need not have the WDM DDK (Device Driver Kit) to compile their applications.
4. Pseudo-HID WDM (Win32 Driver Model) driver. This is the driver that emulates the Memphis HID driver functionality.

The software suite for OSR2.1 is also portable to the Windows 98 when it is released. Another software suite based strictly on Microsoft HID drivers will also be provided in due course. Because of the PSMCI, the applet remains largely unchanged when switching between different versions of the operating systems. Also the firmware that is already shipped with the monitors remains unchanged since it is compliant to the HID Specification and Monitor Control Class Specification.

With this monitor control software suite, Philips Semiconductors has provided a turnkey solution for USB monitor control. The other components in the solution include the USB silicon devices: PDIUSBH11, PDIUSBH11A, PDIUSBH12, PDIUSBD11, PDIUSBD10 with their associated firmware reference codes and evaluation boards complete with schematics, BOM (Bill Of Material) list and gerber files.

Monitor Control Firmware Guidelines

Monitor Control Class specifies that a monitor is a HID device. According to HID specification, one needs to specify the characteristics of the Data Items to be read from or written to that particular device in a structure known as Report Descriptor. Each Data Item will have a Usage Value linked to that particular Data Item. One or more Data Items can be grouped together under a Report ID.

Monitor Control Class Specification does not specify the Report ID values. So it is up to each monitor manufacturer to specify the Report IDs and the grouping of the Data Items under these Report IDs.

In general the Report Descriptor can differ from one monitor to another. One can design and build Monitor Control Application (and potentially the underlying drivers) that will work with different monitors. However, since the applet and the monitor normally comes from the same vendor, it is also perfectly OK to have the applet "tied" with a particular monitor firmware.

After spending a lot of time and with wealth of information available to us, we have come out with a simple, versatile and accurate implementation, which should comply with the final Monitor Control Class Specification. The following describes our Report Descriptor implementation and should be used as a guideline:

1. All Data Items for monitor control are of Feature type. This is a logical choice.
2. The Usage values of the Data Items follow the usage values of the USB Monitor Control Class specification or the Control Code values of the VESA Monitor Control Command Set specification.
3. Each Data Item is assigned one Report ID. This minimizes the USB bandwidth during monitor control (changing brightness means only brightness command is sent on the USB).
4. All Data Items have Report IDs equal to the Usage values. As the Usage values for a particular Usage Page are unique numbers so will be the Report IDs. The uniqueness of the Report IDs is a requirement according to the HID Specification. This gives some kind of "standardized" assignment and may simplify firmware coding.
5. There may be potential conflict between Usage values of different Usage Pages. The strategy is to start off with the VESA Virtual Control and VESA Command Usage Pages first. There are no conflicts in Usage values for these two pages. Other Usage values can then be added from other Usage pages with remaining non-conflicting Report ID.
6. Security Key as described in the following paragraph has to be incorporated in the device firmware to use the monitor control software suite.

Security Key:

This is a Data Item, which has to be part of the REPORT DESCRIPTOR. This is needed to run the Philips Semiconductor's Monitor Control Software Suite.

The item has been defined in the following manner:

Report Count (7)	95h,07h
Report Size(8)	75h,08h
Report ID (FD)	85h,0FDh
Usage(FD)	09h,0FDh
Feature(DATA,VAR,ABS)	B1h,02h

The above Item is under Usage Page Monitor (80h). When inquired, the firmware should return a data packet with the following 7bytes of data:

50h, 68h, 69h, 6Ch, 69h, 70h, 73h.

So including the Report ID, the data packet would be FD, 50h, 68h, 69h, 6Ch, 69h, 70h, 73h.

Device Arrival/Removal Message

You should call RegisterApplet, while starting the Applet. This will leave the CallBack function name with the driver. Driver will call the CallBack function whenever the Monitor Control HID device is plugged-out or Plugged-in.

The CallBack function has the CALLBACK_APPLET_STRUCT as the parameter. The following members of this structure is used:

HidDeviceHandle will contain the Handle of the device, whose status is changed.
DwEvent will contain

02 Device plugged-out
03 Device Plugged-in

Rest of the members of the structure are irrelevant to this call.

This command supports the multiple plug-out and plug-in of the Monitor Control HID device. Now for OSR2.1, you need to call this function as part of your initialization routine to avoid any problems while plug-out and plug-in. The information of CallBack function to the Applet is optional. You may do the following things when a CallBack function is called to inform the Plug-out or Plug-in of the Monitor Control HID device:

1. Ignore the message. In this case you are assuming the monitor being plugged-in is same. If a different monitor is being plugged-in then you should not ignore the command.
2. Close and open the Applet.

3. Refresh the Applet with the latest status, like gray the controls of the Applet.

Essentially by doing this you do not have to close and reopen the Monitor Control Applet, while the Device is plugged-out and then plugged-in.

Accessing Interrrupt Pipe

The data over the interrupt pipe is accessed through a callback function provided by the Applet. Please follow the steps below for the interrupt pipe data access:

Call RegisterApplet (WINAPI* PCallBackApplet)

This will leave the CallBack function name with the driver and driver will call the CallBack function whenever there is data read from the interrupt pipe.

```
The typedef struct _CALLBACK_APPLET_STRUCT
{
        DWORD                   dwEvent,
        PPHILIPS_MAINREPORT     pMainReport,
        PRANGE_REPORT           pRangeReport,
        PAUX_REPORT             pAuxReport

}CALLBACK_APPLET_STRUCT, *PCALLBACK_APPLET_STRUCT;
```

So the CallBackApplet will look like

DWORD WINAPI CallBackFunction(PCALLBACK_APPLET_STRUCT param)

and you should call the RegisterApplet as RegisterApplet(&CallBackApplet);

Reference Documents

1. USB Monitor Control Class Spec. Rev. 1.0.
2. VESA Monitor Control Command Set V1pR0D13 or later.
3. Device Class Definition for Human Interface Devices (HID) V1.0.
4. USB Spec. Rev. 1.0, especially Chapter 9: USB Device Framework.