Philips Semiconductors Monitor Control Interface (PSMCI)

These are the Application Programming Interface (API) provided to ease monitor control application development. They are collectively known as the PSMCI (Philips Semiconductors Monitor Control Interface).

1. GetMonitorDevice

Returns Handles to the Monitor Control HID Devices. These are the handles which shall be used for all kinds of communications with the Monitor Control Device. The Application must call this function once, before starting communication with the Monitor Control HID Devices.

```
BOOLEAN
GetMonitorDevice(
      OUT HANDLE * Hid_DeviceHandle,
      OUT PULONG   Hid_DeviceCount
      );
```

Parameters:

Hid_DeviceHandle
is a pointer to the available Monitor Control HID Devices. It should point to a buffer of size 127 Handles.

Hid_DeviceCount
is the count number of these available Monitor Control HID Devices.

Remarks:

The function checks for all the enumerated Monitor Control HID Devices. It also retrieves the Report Descriptor from each HID Device and puts the information in the proper structures for the future use.

Return values:

TRUE if successful; otherwise FALSE. Also FALSE if no Monitor Control HID Devices are found.

2. Hid_GetReport

Reads the report descriptor of the HID Device into the structure. This function needs to be called when the application wants to know whether that particular Usage is supported by the monitor or not. It will also return the valid range and the current value for the Usage.

```
Hid_GetReport (
      OUT PPHILIPS_MAINREPORT pMainReport,
      OUT PRANGE_REPORT pRangeReport,
      OUT PAUX_REPORT pAuxReport
      );
```

Parameters:

pMainReport
is a pointer to the PHILIPS_MAINREPORT structure that contains the main report information.

pRangeReport
is a pointer to the RANGE_REPORT structure that contains the range report information.

pAuxReport
is a pointer to the AUX_REPORT structure that contains other additional report information.

```
typedef struct _PHILIPS_MAINREPORT {

      HANDLE      HidDeviceHandle;
      USHORT      UsagePage;
      USHORT      Usage;
      PUCHAR      Buffer;
      PULONG      BufferLength;

} PHILIPS_MAINREPORT, *PPHILIPS_MAINREPORT;

typedef struct _RANGE_REPORT {

      PLONG       LogicalMin;
      PLONG       LogicalMax;
      PUCHAR      EnumValues;
      PUCHAR      EnumCount;

} RANGE_REPORT, *PRANGE_REPORT;

typedef struct _AUX_REPORT {

      PLONG       Reserved1;
      PLONG       Reserved2;

} RANGE_REPORT, *PAUX_REPORT;
```

HidDeviceHandle
is handle of the Monitor Control HID Device. Optional if there is only one Monitor Control HID Device. Default is the default Monitor Control HID Device.

UsagePage
is the Usage Page of the item.

Usage
is the Usage of the item.

Buffer
The data value for the UsagePage and Usage combination. For Hid_GetReport call, this is an output and for Hid_SetReport call, this is an input. The buffer of size *BufferLength* should already have been allocated before this call. The buffer should have sufficient length to contain the data to be read. Recommended length is 256 bytes. The data size of returned value is reflected in the BufferLength.

BufferLength
contains the length of the Buffer, i.e. the data size of the Usage being accessed. This is an input and output field. To determine the data size of a Usage, this field is written with some large number and Hid_GetReport call is issued. The return value in this field now indicates the proper data size. For Hid_SetReport call, the data size of the Usage is already known and is written to this field. The default value is 2.

LogicalMin
returns with Logical Minimum value of the Usage.

LogicalMax
returns with Logical Maximum value of the Usage.

EnumValues
contains the possible Enumerated values supported by the Usage, if applicable. The length of EnumValues buffer is given in the EnumCount. The count of the returned values are returned in the EnumCount.

EnumCount
contains the length of the EnumValues buffer. Returned with the count of the possible Enumrated values supported.

Reserved1, Reserved2
Reserved for future use.

Return values:

TRUE if successful; otherwise FALSE.

3. Hid_SetReport

Writes the data to the HID Device.

```
BOOLEAN
Hid_SetReport (
      IN PPHILIPS_MAINREPORT pMainReport
      );
```

Parameters:

pMainReport
is a pointer to the PHILIPS_MAINREPORT structure that contains the main report information.

Return values:

TRUE if successful; otherwise FALSE.

4. RegisterApplet

Registers the Applet with the Driver (DLL) and gives the name of the CallBack function to the Driver. This function is called once as part of the initialization of the applet to avoid problems in OSR2.1 (blue screen upon unplugging the device while the applet is running). It will give messages back to the applet on device plug-out and plug-in or when there is an interrupt data from the device. This allows the implementation of “hot-key” capability (a press on the button of the monitor pops up the applet).

```
DWORD RegisterApplet (
      OUT PCALLBACKAPPLET pCallBackApplet,
      OUT HWND hWnd
       );
```

Usage:

RegisterApplet(&CallBackApplet);

Parameters:

pCallBackApplet
is the CallBack function name. The name of the CallBack function can be anything.

hWnd
is the handle of the Applet Window.

```
typedef DWORD (WINAPI* PCALLBACKAPPLET)( PCALLBACK_APPLET_STRUCT)

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

}CALLBACK_APPLET_STRUCT, *PCALLBACK_APPLET_STRUCT;
```

dwEvent

contains the event information which has occurred:

01	Unknown Event
02	Device is Plugged-out
03	Device is Plugged-in
04	Interrupt pipe data is read. In this case the rest of the parameters contains the data information.

Return values:

TRUE if successful; otherwise FALSE.