Grayhill 3D Series 6.5 Inch Display

Software Developer’s Guide – Windows CE 6.0
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12/29/2011</td>
<td>Ported from 5.0 document RevG to Windows CE 6.0 R3</td>
</tr>
<tr>
<td>A</td>
<td>6/21/2012</td>
<td>- Added overlay to 5 Layer Display Controller section;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- added overlay enable/disable to Camera Driver section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added power get/set IOCTL’s to LCD Backlight section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added Switched Power Input section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Modified options in Switched Power Functionality section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changed baud rate in Console Application section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Removed backlight reference from OEMLauncher section</td>
</tr>
<tr>
<td>B</td>
<td>11/16/2012</td>
<td>Modified CAN driver section for CAN frame/bitrate configuration</td>
</tr>
</tbody>
</table>
# Table of Contents

- **INTRODUCTION** ................................................................. 4
- **LCD DISPLAY** ................................................................. 4
- **DEVELOPMENT TOOLS** .......................................................... 4
- **SDK COMPONENTS** .............................................................. 4
- **SDK INSTALLATION** .............................................................. 5
- **MEMORY AVAILABILITY** ......................................................... 5
- **REMOVABLE STORAGE** .......................................................... 5
- **RESETTING THE UNIT** ........................................................... 5
- **SWITCHED POWER FUNCTIONALITY** ......................................... 6
  - No Action (Default) Option ...................................................... 6
  - Suspend Option ................................................................. 6
  - Shutdown Option ............................................................... 6
- **CONSOLE APPLICATION** ........................................................ 7
- **CUSTOM INTERFACES** ........................................................... 8
  - 5 Layer Display Controller ..................................................... 9
  - Camera Driver Interface ....................................................... 10
  - CAN Driver Interface .......................................................... 13
  - Keypad Driver Interface ....................................................... 16
  - Display (LCD) Backlight ....................................................... 18
  - OEM Launcher ................................................................. 20
  - Keypad Backlight ............................................................... 22
  - Buzzer Driver Interface ....................................................... 23
  - Switched Power Input ......................................................... 24
  - Optional Touch Screen Interface ............................................ 26
- **ETHERNET DEBUGGING CAPABILITIES** ..................................... 27
- **APPENDIX A - DISPLAY PINOUT** ............................................ 31
- **APPENDIX B – BREAKOUT BOARD SCHEMATIC** ........................... 31
Introduction

The Software Developer's Guide provides details on how to interface with the custom components of the Grayhill 3D Series Display using the Grayhill SDK. This document is intended for use by software developers who are familiar with programming in C/C++. Experience developing applications for Windows CE platforms using Microsoft development tools is a definite plus.

LCD Display

The Grayhill 3D Series 6.5-inch Display contains a 16-bit VGA LCD (640x480 pixel).

Development Tools

The Grayhill 3D Series Display is a Microsoft Windows CE 6.0 R3 device. Standard Microsoft development tools such as Microsoft Visual Studio 2005 and Microsoft Visual Studio 2008 can be used to create applications that run on the device. Please visit the Microsoft web site for information on loading and usage of their development tools.

If Microsoft Visual Studio 2005 is being used for development, it is recommended that Visual Studio 2005 Service Pack 1 is installed. This will allow for an ethernet connection to be made between Visual Studio 2005 on the development PC and the Display unit.

Below are links for downloading some of the Microsoft development tools:

Microsoft Visual Studio SP1


SDK Components

There are custom device drivers for several components within the Grayhill 3D Series Display. The Vehicle Display SDK provides developers with interfaces for accessing these drivers and adapting the functionality of the components to their applications.

The SDK provides header files (*.H) for compiling with C/C++ applications, and library files (*.LIB) for linking with C/C++ applications. The header files typically contain function prototypes, which define the name and syntax of each available function. The library files provide the link information so an application can call the available functions through the DLL's provided in the Windows CE operating system.

Most components of the Grayhill 3D Series Display are supported through standard Windows CE 6.0 programming interfaces. The standard APIs provided by Microsoft are documented and available through their developer web sites.
SDK Installation

To install the Grayhill 3D Series Display SDK on your PC, execute the file: VehicleDisplaySDK.msi

The installation application will guide you through the installation procedure.

Memory Availability

The 3D Series Display contains 64MB of RAM. The OS image is typically about 16MB in size and is executed from RAM. The remaining RAM is approximately 48MB, and is divided into 2 partitions: Storage memory and Program memory. Storage memory is used for file writing and data storage; program memory is used for program execution and memory allocation by application programs. The ratio of Storage memory to Program memory can be modified using a standard Windows CE control panel applet.

The 3D Series Display also contains 64MB of non-volatile Flash memory. The Flash memory has 22MB reserved for storing the OS image. The remaining 42MB is available for file storage. The Flash memory is accessible to the file system as directory /FlashDisk. Any files stored in this directory or any of its subdirectories will be maintained through power down.

Removable Storage

The 3D Series Display contains the following removable storage device capability:

- Media
  - USB flash drive
- Directory
  - \Hard Disk

Resetting the Unit

The 3D Series Display can be reset by pressing and holding the topmost keypad button (key F1/Reset) for approximately 10 seconds.
Switched Power Functionality

The switched power input of the display can be used to put the display into a low power state and then to wake the device out of the low power state.

The low power state that is commanded is configurable using registry settings. The following configuration options are available:

- No Action (default)
- Suspend
- Shutdown

**No Action (default) option**

This option is intended to allow the display application to poll the input and command the low power state directly (see Switched Power Input Interface section). To configure the display to take no action based on the state of the switched power input, the following registry setting is used:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\GRAYHILL\POWERMONITOR]
"SwitchedAction"=dword:0 ; 0=no action
```

**Suspend option**

To configure the display to automatically suspend and resume based on the state of the switched power input, the following registry settings are used:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\GRAYHILL\POWERMONITOR]
"SwitchedAction"=dword:2 ; 2=suspend
"SuspendDelay"=dword:EA60 ; 60000 ms
```

In the event that the display is powered up with the switched power off, the display will automatically be suspended after a pre-configured amount of time. This is done so that the display would not remain powered on indefinitely and thus drain the vehicle battery. The time delay is specified with the “SuspendDelay” registry entry. By default, this value is 1 minute (60,000 milliseconds).

**Shutdown option**

To configure the display to automatically shutdown based on the state of the switched power input, the following registry setting is used:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\GRAYHILL\POWERMONITOR]
"SwitchedAction"=dword:4 ; 4=shutdown
```

**Switched Power configuration utility**

Grayhill has a utility application (PowerSwitchConfig.exe) available that can be used to modify the switched power configuration in the registry. To download the application, go to: [http://www.grayhill.com/offhwyfiles.aspx](http://www.grayhill.com/offhwyfiles.aspx).
From the console command line (discussed in the next section), enter:

```
powerswitchconfig <option> <delay>
```

where `<option>` is one of the following values:

- 0 = no action
- 2 = suspend
- 4 = shutdown

and `<delay>` is a value between 0 and 4,294,967,295 milliseconds

If the `<option>` parameter is not specified, the current configuration is returned.

The `<delay>` parameter is only used for option 2 (suspend). If option 2 is specified with no `<delay>` parameter, the current value will remain unchanged. The `<delay>` parameter is ignored if specified with any other options.

So, for example, to set the switched power functionality to option 2 (suspend) with a suspend delay of 90 seconds, enter

```
powerswitchconfig 2 90000
```

**NOTE:** The display must be rebooted for the change(s) to take effect.

### Console Application

The 3D Series Display software includes a console application, which provides a command line interface for programmers to enter various shell commands. The console application can be accessed by connecting to serial port 1 of the display unit with HyperTerminal or other terminal emulation program. Configure the port settings as follows:

- Bits per second 115200
- Data Bits 8
- Parity none
- Stop Bits 1
- Flow Control none

The console application starts automatically on startup of the display unit. From the HyperTerminal session, the message

```
Press ENTER to launch console...
```

will be displayed.

Press `<Enter>` and the command prompt will appear:

```
[WindowsCE] >
```

Typing `Help` and `<Enter>` at the command prompt will display a list of available commands.
Custom Interfaces

The 3D Series Display contains the following custom component interfaces:

- 5 Layer Display Controller
- Camera driver
- CAN driver
- Keypad driver
- Display (LCD) Backlight
- OEM Launcher
- Keypad Backlight
- Buzzer
- Switched Power Input
- Touch Screen (optional)

This section explains how to access the functionality of these components using the Grayhill SDK. The programming interfaces and API functions provided with the SDK are explained, with the syntax and parameters defined. Sample C code is also provided where appropriate.
5 Layer Display Controller

The Grayhill 3D Series Display uses a 5 layer display controller. The display output is composed of 5 distinct frame layers named Layer0 through Layer4. This feature allows the display to efficiently render any combination of the application frame buffer, frame buffer overlay, and 3 live video frame buffers.

Layer0

Layer0 is dedicated to the application frame buffer. Its size is fixed to the resolution of the display (640x480). It has the lowest priority of the 5 layers. It is essentially always on and is only hidden from view when overlapped by the upper layers.

Layer1, Layer2, Layer3

Layer1, Layer2, and Layer3 are dedicated to the live video capture windows. The sizes of the windows for these layers are controlled independently by the CameraWindow API function (discussed in the next section). The device includes a camera video input driver that supports 3 camera video inputs. Each camera video input is displayed in its own window, and can be used individually or in any combination simultaneously. The size of each window may be specified by the application. Overlapping of windows is not recommended. Following are examples of valid window configurations:

Layer4

Layer 4 is an overlay to the application frame buffer, layer 0, and it is the same size as layer 0. The overlay layer can be enabled or disabled independently using the CameraState API function (discussed in the next section). It provides limited alpha blending, such that a 16-bit pixel value of 0 is transparent and all other values are opaque.
Camera Driver Interface

The Grayhill 3D Series Display device includes a camera video input driver that supports 3 camera video inputs. Each camera video input is displayed in its own window, and can be used individually or in any combination simultaneously.

Interface

An application can interface with the Camera driver through the following API function(s):

- CameraState
- CameraWindow
- CameraFlip

CameraState

Enable/Disable camera video input and/or overlay frame buffer.

Syntax

BOOL CameraState( DWORD dwCamMask )

Parameters

DWORD dwCamMask
[in]
Mask indicating the camera inputs to enable or disable. A value of 1=Enable (ON), a value of 0=Disable (OFF).

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CAM1ENA</td>
<td>0 = display of camera 1 disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = display of camera 1 enabled</td>
</tr>
<tr>
<td>1</td>
<td>CAM2ENA</td>
<td>0 = display of camera 2 disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = display of camera 2 enabled</td>
</tr>
<tr>
<td>2</td>
<td>CAM3ENA</td>
<td>0 = display of camera 3 disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = display of camera 3 enabled</td>
</tr>
<tr>
<td>3</td>
<td>OVERLAY</td>
<td>0 = overlay frame buffer disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = overlay frame buffer enabled</td>
</tr>
<tr>
<td>4:31</td>
<td>-</td>
<td>Unused</td>
</tr>
</tbody>
</table>

Return Value

BOOLEAN
TRUE indicates success; FALSE indicates failure. For extended error information, call GetLastError( ).

Required Files

Header File: CameraIF.H
Link Library: CameraIF.LIB
CameraWindow
Configure camera video input window size.

Syntax
BOOL CameraWindow( WORD CamNum, DWORD X_Origin, DWORD Y_Origin, DWORD Width, DWORD Height )

Parameters
WORD CamNum
[in]
Camera Number. Valid range 1-3.

DWORD X_Origin
[in]
X-coordinate (horizontal) of upper left corner of window. Valid range 0-639.

DWORD Y_Origin
[in]
Y-coordinate (vertical) of upper left corner of window. Valid range 0-479.

DWORD Width
[in]
Horizontal size of window in pixels. Valid range 0-639.

DWORD Height
[in]
Vertical size of window in pixels. Valid range 0-480.

Return Value
BOOLEAN
TRUE indicates success; FALSE indicates failure. For extended error information, call GetLastError().

Required Files
Header File: CameraIF.H
Link Library: CameraIF.LIB
**CameraFlip**

Flip the camera video vertically and/or horizontally

**Syntax**

BOOL CameraFlip( WORD Mask)

**Parameters**

WORD Mask  
[in]  
Set individual bits in Mask to enable flipping  
bit0 - camera 1 flip x axis  
bit1 - camera 1 flip y axis  
bit2 - camera 2 flip x axis  
bit3 - camera 2 flip y axis  
bit4 - camera 3 flip x axis  
bit5 - camera 3 flip y axis

**Return Value**

BOOLEAN  
TRUE indicates success; FALSE indicates failure. For extended error information, call GetLastError().

**Required Files**

Header File: CameraIF.H  
Link Library: CameraIF.LIB

**Sample C Code**

```c
#define BIT_CAM1 0x0001
#define BIT_CAM2 0x0002
#define BIT_CAM3 0x0004

DWORD dwMask ;

// configure camera #1 window
CameraWindow( 1, 276, 127, 440, 333 ) ;

// configure camera #2 window
CameraWindow( 2, 23, 119, 218, 151 ) ;

// configure camera #3 window
CameraWindow( 3, 23, 316, 218, 151 ) ;

// enable all 3 camera inputs
dwMask = BIT_CAM1 | BIT_CAM2 | BIT_CAM3 ; //dwMask = 0x0007
CameraState( dwMask ) ;

// flip camera #1 on the y-axis
dwMask = 0x0002; // bit 1 set  
CameraFlip(dwMask);
```
CAN Driver Interface

The Grayhill 3D Series Display device includes three CAN controller modules.

Interface

The CAN Driver is implemented as a streams driver. The application will use the Windows CE streams driver interface available as part of the standard SDK to communicate directly with the CAN driver.

The available CAN ports are:

- CAN1:
- CAN2:
- CAN3:

The bitrate and frame type parameters for each CAN port can be set via the streams IOCTL interface.

Supported bitrates are 125Kbps, 250Kbps, 500Kbps, and 1Mbps.

The CAN port can be configured to support Standard frames, Extended frames, or both.

The default bitrate is 250Kbps and the default frame type is Extended frame format.

<table>
<thead>
<tr>
<th>IOCTL</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCTL_CAN_SET_BITRATE</td>
<td>Set the bit rate for CAN communications</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BITRATE_125K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BITRATE_250K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BITRATE_500K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BITRATE_1M</td>
</tr>
<tr>
<td>IOCTL_CAN_SET_FRAMETYPE</td>
<td>Sets the frame type to standard, extended, or both</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRAME_STANDARD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRAME_EXTENDED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FRAME_STDEXT</td>
</tr>
</tbody>
</table>

If the driver is configured for both Standard and Extended frames, applications must use the most significant bit (bit 31) in the Frame ID field to identify/indicate whether a data frame is standard or extended format.

Set bit 31 for standard frames, Clear bit 31 for extended frames.

It is the responsibility of the application to clear this bit before using the frame ID.

Program Flow to Transmit a CAN message:

```
CreateFile
Configure bitrate and frametype
WriteFile
```

...
CloseHandle

Program Flow to Receive a CAN message:

CreateFile
Configure bitrate and frametype
ReadFile

CloseHandle

Data Types

typedef struct _CANMSG
{
    UINT32 ID;
    BYTE Length;    // Data Length Code of the Msg (0..8)
    BYTE Data[8];   // Data 0 .. 7
} CANMSG, *PCANMSG;

Required Files

Header File: candrv.h

Sample C Code

HANDLE hFile;
CANMSG Msg;
DWORD dwBytesIn;
DWORD bitrate;
DWORD frametype;

// open the CAN port
hFile = CreateFile("CAN1:",
    GENERIC_READ | GENERIC_WRITE, 0,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL);

if (hFile == INVALID_HANDLE_VALUE)
{
    printf("Error opening CAN port \r\n");
    printf("GetLastError() = %d\r\n", GetLastError());
    return 0;
}

bitrate = BITRATE_125K;  // 125Kbps
if (!DeviceIoControl(hCAN, IOCTL_CAN_SET_BITRATE, &bitrate, sizeof(DWORD),
    NULL, 0, NULL, NULL))
{
// error setting bitrate
}

frametype = FRAME_STDEXT;     // both Standard and Extended frame formats
if (!DeviceIoControl(hCAN, IOCTL_CAN_SET_FRAMETYPE, &frametype, sizeof(DWORD), NULL, 0, NULL, NULL))
{
    // error setting frame type
}

// read from the CAN port
ReadFile(hFile, &Msg, sizeof(CANMSG), &dwBytesIn, NULL);

if (dwBytesIn == (sizeof(CANMSG)) )
{
    if ((Msg.ID & FRAME_TYPE_MASK) == FRAME_TYPE_STANDARD)
    {
        // received Standard frame, clear out the frame type indicator bit
        Msg.ID &= ~FRAME_TYPE_MASK;
    }
    else
    {
        // received Extended frame
    }

    // process CAN message
    ...  
    ...
}

// close the CAN port
CloseHandle( hFile ) ;
Keypad Driver Interface

The Grayhill 3D Series Display device includes a 5-button keypad that may be used to enhance an application’s behavior. The keys are arranged vertically on the display, with Key 1 on top and Key 5 on the bottom.

The virtual key code map is as follows:

<table>
<thead>
<tr>
<th>Key</th>
<th>Virtual Key Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x88</td>
</tr>
<tr>
<td>2</td>
<td>0x87</td>
</tr>
<tr>
<td>3</td>
<td>0x86</td>
</tr>
<tr>
<td>4</td>
<td>0x85</td>
</tr>
<tr>
<td>5</td>
<td>0x84</td>
</tr>
</tbody>
</table>

Interface

An application may process individual key presses by capturing the windows message corresponding to key up and key down events. Each key can then be programmed with customized behavior.
Sample C Code

```c
LRESULT WindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch( message )
    {
        case WM_KEYDOWN :
            switch ( wParam )
            {
                case 0x84 :
                    // process keypress
                    ...
                    break ;
                case 0x85 :
                    // process keypress
                    ...
                    break ;
                case 0x86 :
                    // process keypress
                    ...
                    break ;
                case 0x87 :
                    // process keypress
                    ...
                    break ;
                case 0x88 :
                    // process keypress
                    ...
                    break ;
            }
        break ;
        case WM_KEYUP :
            break ;
    }
    break ;
}
```
Display (LCD) Backlight

The display backlight can be controlled from an application via the BKL streams driver. Using the appropriate IOCTLS defined in the LcdBL_ioctl.h header file, a developer can get/set the values of backlight intensity via the DeviceIOControl system call.

<table>
<thead>
<tr>
<th>IOCTL</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCTL_LCD_BACKLIGHT_GET_LEVEL</td>
<td>Returns the percentage value of the LCD backlight level</td>
<td>Type: DWORD, Direction: IN, Range: 0-100</td>
</tr>
<tr>
<td>IOCTL_LCD_BACKLIGHT_SET_LEVEL</td>
<td>Sets the percentage value of the LCD backlight level</td>
<td>Type: DWORD, Direction: OUT, Range: 0-100</td>
</tr>
<tr>
<td>IOCTL_POWER_GET</td>
<td>Returns the power state of the LCD backlight</td>
<td>Type: CEDEVICE_POWER_STATE, Direction: IN, D0=On, D4=Off</td>
</tr>
<tr>
<td>IOCTL_POWER_SET</td>
<td>Sets the power state of the LCD backlight</td>
<td>Type: CEDEVICE_POWER_STATE, Direction: (see NOTE below), D0=On, D4=Off</td>
</tr>
</tbody>
</table>

**NOTE:** IOCTL_POWER_SET should be used in the DeviceIOControl( ) function as if it were a GET (in) operation rather than a SET (out) operation. Thus, instead of specifying the power state in the InBuffer function parameter, it should be specified in the OutBuffer function parameter. Similarly, the buffer size should be specified in the OutBufferSize parameter rather than the InBufferSize parameter. This special case pertains only to IOCTL_POWER_SET. See the Sample C Code below for an example.

If the power state of the LCD backlight is not D0, set the desired level and set the power state to D0. By default, the display backlight is left OFF on device startup. It is left up to a user application to control turning on the backlight when appropriate. User applications can be automatically executed on startup using the provided OEM Launcher application, which will be discussed in a subsequent section.

**Sample C Code**

```c
HANDLE hDrv;
DWORD dwValue;
DWORD dwBytes;

// get handle to the keypad backlight driver
hDrv = CreateFile(L"BKL1:\", GENERIC_READ|GENERIC_WRITE, 0, NULL,
                 OPEN_EXISTING, 0, NULL);
if (hDrv != INVALID_HANDLE_VALUE)
```

if (hDrv != INVALID_HANDLE_VALUE)
{  
    // desired brightness level  
    dwValue = 100 ;

    // issue the request  
    DeviceIoControl(hDrv, IOCTL_LCD_BACKLIGHT_SET_LEVEL, &dwValue,  
                     sizeof(DWORD), NULL, 0, &dwBytes, NULL ) ;

    // power state ON  
    pwr_state_out = D0 ;

    // issue the request  
    DeviceIoControl( hDrv, IOCTL_POWER_SET, NULL, 0,  
                     (BYTE *)&pwr_state_out, sizeof(CEDEVICE_POWER_STATE),  
                     &dwBytes, NULL ) ;

    // close handle  
    CloseHandle(hDrv);  
}
OEM Launcher

The Grayhill 3D Series Display device includes an OEM Launcher application that automatically executes on startup. The function of the OEM Launcher application is to launch user applications automatically on startup. The behavior of the application is as follows:

1. The registry is checked to see if a startup application is specified under the following key:

    HKEY_LOCAL_MACHINE\SOFTWARE\GRAYHILL\OEMLAUNCHER
    LaunchApplication=

    If the “LaunchApplication” key contains an application name, the OEM Launcher searches the following locations (in the order listed) for the specified file:

    | Media          | Location         |
    |----------------|------------------|
    | USB Flash drive | \Hard Disk       |
    | Flash memory   | \FlashDisk       |

    If the application is found, it is immediately executed and the OEM Launcher is terminated.

    If any of the following conditions are met, the OEM launcher continues to step 2:
    • No application is specified in the registry
    • The application specified in the registry is not found in any of the possible locations
    • The application fails to launch

2. The OEM Launcher searches the following startup locations for any .EXE files:

    | Media          | Location         |
    |----------------|------------------|
    | USB Flash drive | \Hard Disk\Startup |
    | Flash memory   | \FlashDisk\Startup |

    Every .EXE file found in all of the startup locations will be launched.

Sample file copy application

Grayhill has a sample application (samplecopy.exe) available that demonstrates how, when used with OEMLauncher, to automatically copy files from the USB Flash drive to the Display. To download the application, go to: http://www.grayhill.com/offhwyfiles.aspx.

Samplecopy launches a batch file (called MyFileCopy.bat) which copies all files from the \Startup\MyFiles folder of the USB flash drive to the \FlashDisk\Startup folder of the Display.

To execute the application and batch file automatically on bootup:
1. On the USB flash drive, create a folder called \Startup
2. Copy samplecopy.exe to the USB flash drive \Startup folder
3. Copy MyFileCopy.bat to the USB flash drive \Startup folder
4. Create a folder on the USB flash drive called \Startup\MyFiles
5. Copy desired files (to be copied to the Display) to the USB flash drive \Startup\MyFiles folder
6. Insert the USB flash drive into the Display
7. Reboot the display

On bootup, the Display will automatically launch the *samplecopy* application, which in turn will launch the batch file (MyFileCopy.bat).

The batch file can then be modified as needed. File names and folder names can be easily changed without having to re-build the application.
Keypad Backlight

The Grayhill 3D Series Display device includes a keypad backlight. The keypad backlight can be controlled from an application via the KPB streams driver. Using the appropriate IOCTLs defined in the KeypadBL_ioctl.h header file, a developer can get/set the values of keypad backlight intensity via the DeviceIOControl system call.

<table>
<thead>
<tr>
<th>IOCTL</th>
<th>Description</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCTL_KPD_BACKLIGHT_UP</td>
<td>Increases the keypad backlight intensity level one step</td>
<td></td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_DOWN</td>
<td>Decreases the keypad backlight intensity level one step</td>
<td></td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_MAX</td>
<td>Sets the keypad backlight intensity to the maximum level</td>
<td></td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_MIN</td>
<td>Sets the keypad backlight intensity to the minimum level</td>
<td></td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_GET_LEVEL</td>
<td>Reads the value of the keypad backlight intensity level</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Range: 0-100</td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_SET_LEVEL</td>
<td>Sets the keypad backlight intensity level to the specified value</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Range: 0-100</td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_GET_ENABLED</td>
<td>Queries to determine if the keypad backlight is enabled</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0=disabled, 1=enabled</td>
</tr>
<tr>
<td>IOCTL_KPD_BACKLIGHT_SET_ENABLED</td>
<td>Enables the keypad backlight</td>
<td>Type: DWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Direction: OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0=disabled, 1=enabled</td>
</tr>
</tbody>
</table>

Sample C Code

```c
HANDLE hDrv;
DWORD dwValue;
DWORD dwBytes;

// get handle to the keypad backlight driver
hDrv = CreateFile(L"KPB1:","GENERIC_READ|GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);

if (hDrv != INVALID_HANDLE_VALUE)
{
    // desired brightness level
    dwValue = 50;

    // issue the request
    DeviceIoControl(hDrv, IOCTL_KPD_BACKLIGHT_SET_LEVEL, &dwValue, sizeof(DWORD), NULL, 0, &dwBytes, NULL);

    // close handle
    CloseHandle(hDrv);
}
```

11/19/12 Grayhill Confidential
Buzzer Driver Interface

The Grayhill 3D Series Display device includes an audio Buzzer.

Interface

An application can interface with the Buzzer driver through the following API function(s).

- BuzzerState

BuzzerState

This function sets the Buzzer state to ON (1) or OFF (0).

Syntax

BOOL BuzzerState ( DWORD dwState )

Parameters

DWORD dwState [in]

32-bit unsigned value containing the desired state value.

Return Value

BOOLEAN

TRUE indicates success; FALSE indicates failure. For extended error information, call GetLastError().

Required Files

Header File: BuzzerIF.H
Link Library: BuzzerIF.LIB

Sample C Code

```c
DWORD dwVal ;
dwVal = 0x0001 ;
if ( !BuzzerState(dwVal) )
{
    printf("BuzzerState() FAILED\r\n") ;
    printf("GetLastError() = %d\r\n", GetLastError()) ;
}
else
{
    printf("Buzzer state set to = %u \r\n", dwVal) ;
}
```
Switched Power Input

The Grayhill 3D Series Display device includes a switched power input.

Interface

An application can query the state of the switched power input through the following API function:

- GetSwitchPwrState

GetSwitchPwrState

This function requests the state of the switched power input (1=ON, 0=OFF).

Syntax

BOOL GetSwitchPwrState ( DWORD *dwState )

Parameters

DWORD *dwState
[out]
pointer to 32-bit unsigned value containing the state value (1=ON, 0=OFF)

Return Value

BOOLEAN
TRUE indicates success; FALSE indicates failure. For extended error information, call GetLastError( ).

Required Files

Header File: PwrFaultDrvIF.H
Link Library: PwrFaultDrvIF.LIB

Sample C Code

```c
DWORD dwState;
if (!GetSwitchPwrState(&dwState))
{
    // error
    printf("error reading state\r\n");
} else
{
    if ( dwState == 0 )
    {
        // switched power is OFF - prepare to suspend
```
TODO: Add code here...

// go to suspend
SetSystemPowerState(TEXT("Suspend"), 0, POWER_FORCE);
Optional Touch Screen Interface

The Grayhill 3D Series Display device may include an optional touch screen interface.

Interface
An application may process touch screen (stylus) input by capturing the Windows CE messages corresponding to mouse events. The following messages are supported:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WM_LBUTTONDOWN</td>
<td>The user pressed the screen</td>
</tr>
<tr>
<td>WM_LBUTTONUP</td>
<td>The user released the stylus from the screen</td>
</tr>
<tr>
<td>WM_LBUTTONDBLCLK</td>
<td>The user double-tapped the screen</td>
</tr>
<tr>
<td>WM_MOUSEMOVE</td>
<td>The user moved the stylus while the tip was pressed to the screen</td>
</tr>
</tbody>
</table>

Sample C Code

```c
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message) {
    case WM_LBUTTONDOWN:
        // process button down message
        ...
        break;
    case WM_LBUTTONUP:
        // process button up message
        ...
        break;
    case WM_LBUTTONDBLCLK:
        // process button double-click message
        ...
        break;
    case WM_MOUSEMOVE:
        // process mouse move message
        ...
        break;
    }
}
```
Ethernet Debugging Capabilities

How to connect to the 3D Series Display from Microsoft Visual Studio 2005/2008 via ethernet

On the 3D Series Display, from the console command prompt:

1. Obtain the IP address of the Display unit using the `ipconfig` utility
2. For Visual Studio 2005:
   - Launch batch file `Dbg2005.bat`
   - OR-
   - For Visual Studio 2008:
     - Launch batch file `Dbg2008.bat`

On the PC, in Visual Studio 200x:

3. From the Tools menu, select **Device Options**
4. In the Options dialog box, select **Device Tools → Devices**

5. Click the **Properties** button
6. In the Properties dialog box under Transport, select TCP Connect Transport then click **Configure**
7. In the Configure TCP/IP Transport dialog box, under Device IP address, check **Use Specific IP address**. Enter the IP address of the Display unit (obtained in step 1).

8. Click ‘OK’
9. Click ‘OK’ in the Properties dialog
10. Click ‘OK’ in the Options dialog
11. From the Tools menu, select **Connect to Device**
12. Connection should be established!
Debugging a large application

When debugging a large application in Visual Studio (such as the Grayhill DashDemo application), it may be necessary to adjust the memory allocation settings of the Display. Otherwise, when attempting to download the application and associated debugging files, an error message like the following might be received on the development PC:

![Microsoft Visual Studio error dialog]

If this occurs, launch the System applet from the Control Panel on the Display. Move the slider a little to the right and retry debugging the application. Adjust as necessary until the application downloads successfully to the Display.

![System Properties dialog]

If the updated memory settings are not maintained through a power reset. However, the 3D Series Display provides an option for making the memory adjustment persistent through a power reset. To do so, perform the following steps:

1. Adjust memory division as desired
2. Close the System Properties applet
3. Close the Control Panel window. The updated memory settings will be saved in the registry.
4. Copy the file MemAdjust.exe from the \Windows folder to folder \Flashdisk\Startup. This will cause the MemAdjust application to be automatically launched on bootup (see the OEM
Launcher section under Custom Interfaces). MemAdjust will reset the memory settings using the value saved in the registry.

NOTE: If the MemAdjust application is not placed in the \Flashdisk\Startup folder, or if it is removed from the folder, the Display will boot with the original default memory settings.
Appendix A - Display Pinout

Connector Information

Connector on Display: AMP 776231-1  
Mating Connector: AMP 776164-1  
Contacts: AMP 770520-1 or AMP 770854-1

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video 1 Signal</td>
<td>Orange</td>
</tr>
<tr>
<td>2</td>
<td>Video 1 Return</td>
<td>Brown</td>
</tr>
<tr>
<td>3</td>
<td>CAN 1 High</td>
<td>Blue</td>
</tr>
<tr>
<td>4</td>
<td>CAN 1 Shield</td>
<td>Blue</td>
</tr>
<tr>
<td>5</td>
<td>CAN 1 Low</td>
<td>Blue</td>
</tr>
<tr>
<td>6</td>
<td>CAN 2 High</td>
<td>Blue</td>
</tr>
<tr>
<td>7</td>
<td>CAN 2 Low</td>
<td>Blue</td>
</tr>
<tr>
<td>8</td>
<td>COM 1 Return</td>
<td>Green</td>
</tr>
<tr>
<td>9</td>
<td>COM 2 RTS</td>
<td>Green</td>
</tr>
<tr>
<td>10</td>
<td>COM 2 CTS</td>
<td>Green</td>
</tr>
<tr>
<td>11</td>
<td>COM 3 Return</td>
<td>Green</td>
</tr>
<tr>
<td>12</td>
<td>COM 3 RTS</td>
<td>Green</td>
</tr>
<tr>
<td>13</td>
<td>Video 2 Signal</td>
<td>Orange</td>
</tr>
<tr>
<td>14</td>
<td>Video 2 Return</td>
<td>Brown</td>
</tr>
<tr>
<td>15</td>
<td>Ethernet TPO+</td>
<td>Red</td>
</tr>
<tr>
<td>16</td>
<td>Ethernet TPO-</td>
<td>Red</td>
</tr>
<tr>
<td>17</td>
<td>CAN 3 Shield</td>
<td>Blue</td>
</tr>
<tr>
<td>18</td>
<td>CAN 3 High</td>
<td>Blue</td>
</tr>
<tr>
<td>19</td>
<td>JAN 3 Low</td>
<td>Blue</td>
</tr>
<tr>
<td>20</td>
<td>COM 1 CTS</td>
<td>Green</td>
</tr>
<tr>
<td>21</td>
<td>COM 1 RTS</td>
<td>Green</td>
</tr>
<tr>
<td>22</td>
<td>COM 2 Return</td>
<td>Green</td>
</tr>
<tr>
<td>23</td>
<td>COM 3 TXD</td>
<td>Green</td>
</tr>
<tr>
<td>24</td>
<td>Video 3 Signal</td>
<td>Orange</td>
</tr>
<tr>
<td>25</td>
<td>Video 3 Return</td>
<td>Brown</td>
</tr>
<tr>
<td>26</td>
<td>Ethernet TPI+</td>
<td>Red</td>
</tr>
<tr>
<td>27</td>
<td>Ethernet TPI-</td>
<td>Red</td>
</tr>
<tr>
<td>28</td>
<td>CAN 1 High</td>
<td>Blue</td>
</tr>
<tr>
<td>29</td>
<td>CAN 1 Shield</td>
<td>Blue</td>
</tr>
<tr>
<td>30</td>
<td>CAN 1 Low</td>
<td>Blue</td>
</tr>
<tr>
<td>31</td>
<td>COM 1 RXD</td>
<td>Green</td>
</tr>
<tr>
<td>32</td>
<td>COM 1 TXD</td>
<td>Green</td>
</tr>
<tr>
<td>33</td>
<td>COM 2 RXD</td>
<td>Green</td>
</tr>
<tr>
<td>34</td>
<td>COM 2 TXD</td>
<td>Green</td>
</tr>
<tr>
<td>35</td>
<td>COM 3 RXD</td>
<td>Green</td>
</tr>
</tbody>
</table>

Connector Pinout: (looking into the back of the display or the back of the mating connector)

Function Color Key

- Video 1
- Video 2
- Video 3
- Ethernet
- CAN 1
- CAN 2
- CAN 3
- COM 1
- COM 2
- COM 3
- POWER
Appendix B – Breakout Board Schematic