



## ***Grayhill 3Dxx Display Products***

Developer's Manual

## Revision History

Revision	Date	Description
A	09/13/2018	Original Release
B	12/10/2018	Modified Recovery OS section in Appendix; Changed document name

---

## Table of Contents

INTRODUCTION .....	5
LOGIN / PASSWORD .....	5
UPDATING FILES ON THE DEVICE USING <i>APP_UPDATE</i> .....	5
MAKING A BACKUP OF THE LINUX KERNEL IMAGE.....	6
SETTING FLASH FILE SYSTEM ‘READ/WRITE’ MODE .....	8
‘BOOT QUIET’ MODE .....	8
SETTING THE LCD BACKLIGHT .....	8
SETTING THE REAL TIME CLOCK.....	9
CHANGING THE CAN BIT RATE .....	10
BUZZER.....	10
CAMERA INTERFACE .....	11
SWITCHED INPUT .....	15
USB FLASH.....	18
AVAILABLE UTILITIES AND LOCATIONS .....	18
ETHERNET CONFIGURATION.....	20
COPYING FILES FROM ETHERNET .....	22
CONFIGURING AN APPLICATION TO RUN AT BOOTUP.....	22
SECURE SHELL (SSH).....	23
DISPLAYING “.RAW” IMAGE FILES .....	23
FRAME BUFFER ALPHA BLENDING .....	24
SCRIPT FILES .....	25
APPENDIX .....	26
UPDATING FILES ON THE DEVICE USING THE RECOVERY OS .....	26

HOW TO DISABLE THE RECOVERY OS..... 26

## Introduction

This manual describes some basic Linux features for a Grayhill 3Dxx Display product.

Currently the following 3Dxx Display models are supported:

- Model 3D50 – 5 Inch Display
- Model 3D70 – 7 Inch Display
- Model 3D2104 – 10.4 Inch Display

This document is intended for use by software developers who are familiar with Linux platforms.

## Login / Password

- At the “ghiimx6 login:” prompt, enter “**root**” (no password is required).
- To change the password, enter

```
passwd <user>
```

where <user> is the username of the account for the password you are changing.  
The system will then prompt for the new password.

**\*\* Be sure to use care if changing the root password! \*\***

## Updating Files on the Device Using *app\_update*

The 3Dxx Series displays contain a utility application that is used for installing files on the display. The update application is called *app\_update.exe* and is found in folder */bin*. It is launched automatically on bootup from script file *launchappupdate*, which is found in folder */etc/init.d*. The update application runs in the background and will detect the insertion of a USB flash drive.

When a USB flash drive is inserted, the application checks the root of the USB flash drive for a folder named *3D50\_update*. If the folder exists, the application will then look in the folder for a script file named *app\_update.sh*. If the script file is found, it will be launched. The script file can be customized as needed. The USB flash drive is mounted at */mnt/usbhd*. There is not much security in the way this update takes place; thus the source code for the *app\_update* application is available for users to modify as needed.

**NOTE:** The *app\_update* script will not be run again until the next power cycle or soft reboot. Re-inserting the USB stick on the same boot cycle will not re-run the script.

### Example script *app\_update.sh*

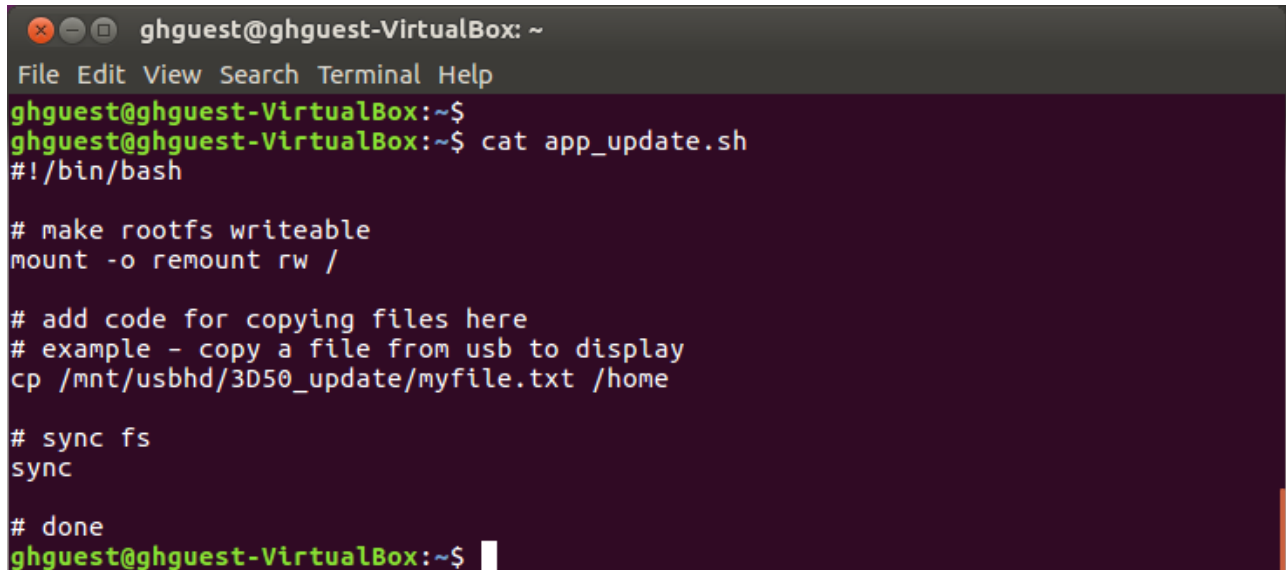
```
#!/bin/bash

# make rootfs writeable
mount -o remount,rw /

# add code for copying files here
# example - copy a file from usb to display
cp /mnt/usbhd/3D50_update/myfile.txt /home

# sync fs
sync
```

# done



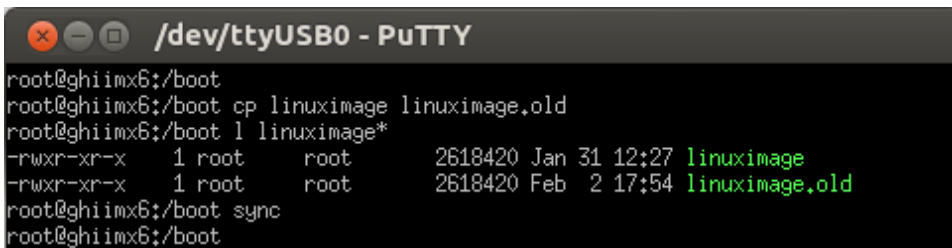
```
ghguest@ghguest-VirtualBox: ~  
File Edit View Search Terminal Help  
ghguest@ghguest-VirtualBox:~$  
ghguest@ghguest-VirtualBox:~$ cat app_update.sh  
#!/bin/bash  
  
# make rootfs writeable  
mount -o remount rw /  
  
# add code for copying files here  
# example - copy a file from usb to display  
cp /mnt/usbhd/3D50_update/myfile.txt /home  
  
# sync fs  
sync  
  
# done  
ghguest@ghguest-VirtualBox:~$
```

## Making a Backup of the Linux Kernel Image

The linux kernel that gets loaded at bootup is called **linuximage** and is located in the following folder:

/boot

If a new kernel is to be loaded, first backup the existing linuximage file by making a copy of it.  
For example:



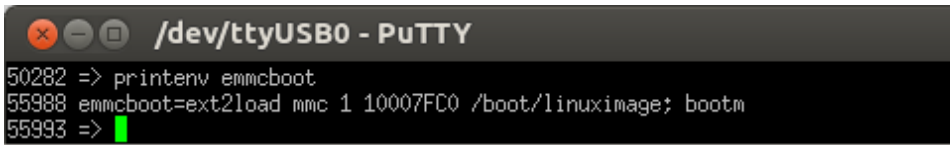
```
/dev/ttyUSB0 - PuTTY  
root@ghiiix6:/boot  
root@ghiiix6:/boot cp linuximage linuximage.old  
root@ghiiix6:/boot ls linuximage*  
-rwxr-xr-x 1 root root 2618420 Jan 31 12:27 linuximage  
-rwxr-xr-x 1 root root 2618420 Feb 2 17:54 linuximage.old  
root@ghiiix6:/boot sync  
root@ghiiix6:/boot
```

Then copy the new **linuximage** file into the /boot folder.

In the event that the linux kernel will not boot, the backup copy of the kernel can be used to boot. To do this, the environment variable **emmcboot** needs to be modified. This is done in the u-boot bootloader. The default setting for emmcboot is as follows:

```
emmcboot=ext2load mmc 1 10007FC0 /boot/linuximage; bootm
```

U-boot screenshot:

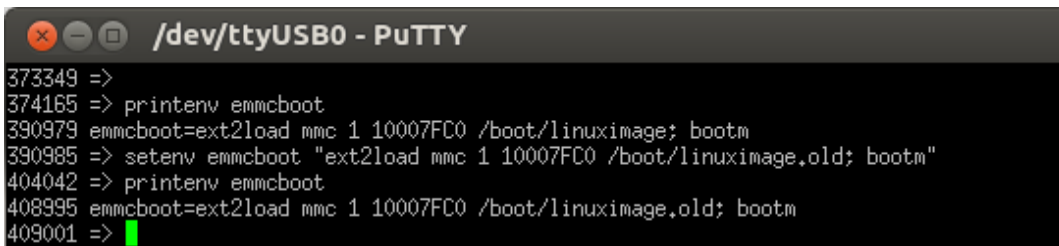


```
/dev/ttyUSB0 - PuTTY
50282 => printenv emmcboot
55988 emmcboot=ext2load mmc 1 10007FC0 /boot/linuximage; bootm
55993 =>
```

Change the setting as follows:

```
emmcboot=ext2load mmc 1 10007FC0 /boot/linuximage.old; bootm
```

U-boot screenshot:



```
/dev/ttyUSB0 - PuTTY
373349 =>
374165 => printenv emmcboot
390979 emmcboot=ext2load mmc 1 10007FC0 /boot/linuximage; bootm
390985 => setenv emmcboot "ext2load mmc 1 10007FC0 /boot/linuximage.old; bootm"
404042 => printenv emmcboot
408995 emmcboot=ext2load mmc 1 10007FC0 /boot/linuximage.old; bootm
409001 =>
```

Once the backup linux kernel has been successfully booted, the backup can be copied to linuximage and the emmcboot environment variable returned to its original setting.

## Setting Flash File System 'Read/Write' Mode

By default, the 3Dxx Display internal flash file system is configured to be read-only so that the system boots up faster and to protect the file system from corruption.

- To immediately set the 3Dxx Display file system to read-write mode enter this console command:

```
mount -o remount,rw/
```

- The above command only remains in effect until the next reboot.
- Grayhill provides scripts for setting the read/write mode in the display.
  - `setwriteable.sh`
  - `setreadonly.sh`

### NOTE

It is not recommended to leave the 3Dxx Display flash file system in read-write mode on fielded units unless it is needed. This can slow device boot-up.

## 'Boot Quiet' Mode

The display can be set to "boot quiet" mode which will disable most of the serial console output that is normally produced during bootup. The benefit resulting from this is a faster bootup time.

To put the display in boot quiet mode, the 'bootargs' environment variable must be modified by adding the 'quiet' option:

```
bootargs=console=ttyMXC0,115200 ip=dhcp lpj=7905280 rootfstype=ext4  
root=/dev/mmcblk0p1 ro rootwait board-ghi_imx6.pn=3D50VT-100 quiet
```

Grayhill provides scripts for setting/unsetting boot quiet mode in the display.

- `setbootquiet.sh`
- `clrbootquiet.sh`

## Setting the LCD Backlight

The LCD Backlight setting is a value between 0 (min) and 100 (max) inclusive.

**Setting from command line:**

```
echo xx > /sys/class/backlight/pwm-backlight.0/brightness
```

where xx is value between 0 - 100



**Setting from an application:**

**Sample C Code:**

```
int    file = 0 ;
int    val ;
int    count = 0 ;
char   cValue[5] ;

if ( file == 0 )
{
    file = open( "/sys/class/backlight/pwm-backlight.0/brightness", O_WRONLY);

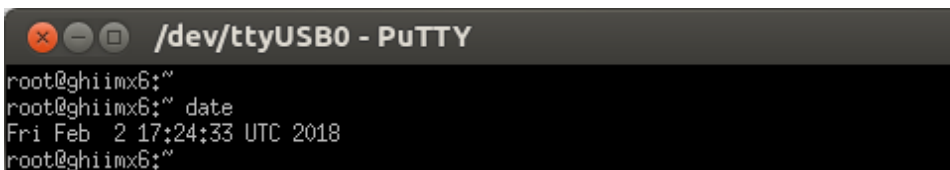
    if ( file > 0 )
    {
        // brightness between 0 - 100
        val = 50 ;
        sprintf( cValue, "%d", val ) ;

        count = write( file, cValue, strlen(cValue) ) ;

        close( file ) ;
    }
}
```

## Setting the Real Time Clock

To view the date and time on the 3Dxx display, the linux command 'date' can be used:



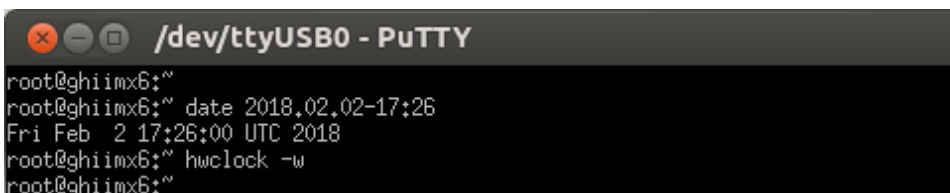
```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~# date
Fri Feb  2 17:24:33 UTC 2018
root@ghiimx6:~#
```

To modify the date and time on the 3Dxx display, use the date command followed by the command:

```
hwclock -w
```

Note: the filesystem must first be made writeable for the hwclock -w command to work.

Example of setting the date and time:



```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~# date 2018,02,02-17:26
Fri Feb  2 17:26:00 UTC 2018
root@ghiimx6:~# hwclock -w
root@ghiimx6:~#
```

## Changing the CAN Bit Rate

To change the bit rate of the CAN port, file **can-pre-up** in folder **/etc/network** must be modified. By default, the CAN bit rate is set at 250 Kbps in the following line:

```
BITRATE=250000
```

The following bit rate values have been tested on the 3Dxx displays:

<b>BITRATE</b>	<b>Kbps</b>
125000	125
250000	250
500000	500
1000000	1000

## Buzzer

Display models 3D70 and 3D2104 contain an internal buzzer that can be commanded on and off via software. The buzzer interface is a control file; writing the appropriate value to the file turns the buzzer on or off.

### **Controlling the buzzer from command line:**

```
echo xx > /sys/class/backlight/pwm-backlight.3/brightness
```

where xx is value between 0 – 25

### **Controlling the buzzer from an application:**

#### **Sample C Code:**

```
int file = 0 ;
int val ;
int count = 0 ;
char cValue[5] ;

if ( file == 0 )
{
    file = open( "/sys/class/backlight/pwm-backlight.3/brightness", O_WRONLY);

    if ( file > 0 )
    {
        val = 20 ;
        sprintf( cValue, "%d", val ) ;

        // buzzer on
        count = write( file, cValue, strlen(cValue) ) ;

        //...
```

```
    val = 0 ;
    sprintf( cValue, "%d", val ) ;

    // buzzer off
    count = write( file, cValue, strlen(cValue) ) ;

    close( file ) ;
}
}
```

## Camera Interface

The Grayhill 3Dxx displays contain multiple camera inputs, the number of which varies by model. The camera input sensor parameters provide support for NTSC and PAL formats. The camera output can be displayed on the display LCD.

The following camera display parameters can be modified via software:

- Window size and position
- Color Parameters – brightness, contrast, saturation, hue
- Rotation
- Camera output to LCD foreground or background with color key

Camera output is displayed at 30 fps.

### Interface:

Applications can interface with the camera driver using the *Camera* class.

### Data Types:

```
#define FOREGROUND (1)
#define BACKGROUND (0)

// These are the only allowed values for VIDEO_COLOR_KEY_xxx:
#define VIDEO_COLOR_KEY_BLACK (0x00000000)
#define VIDEO_COLOR_KEY_RED (0x00FF0000)
#define VIDEO_COLOR_KEY_GREEN (0x0000FF00)
#define VIDEO_COLOR_KEY_BLUE (0x000000FF)
#define VIDEO_COLOR_KEY_YELLOW (0x00FFFF00)
#define VIDEO_COLOR_KEY_CYAN (0x0000FFFF)
#define VIDEO_COLOR_KEY_MAGENTA (0x00FF00FF)
#define VIDEO_COLOR_KEY_WHITE (0x00FFFFFF)

typedef struct _DISPLAYPARAMS
{
    unsigned int top; // top left window y-coordinate
    unsigned int left; // top left window x-coordinate
                    // (must be divisible by 4)
```

```

    unsigned int height;    // window vertical size
    unsigned int width;    // window horizontal size
                          // NOTE: top + height must not exceed height of display
                          // and left + width must not exceed display width
    unsigned int rotate;   // 0-7, see below
    unsigned int fg;       // FOREGROUND or BACKGROUND + VIDEO_COLOR_KEY_xxx
} DISPLAYPARAMS, *PDISPLAYPARAMS;

```

```

#define HUE_CODE_00 (0x00)
#define HUE_CODE_7F (0x7F)
#define HUE_CODE_80 (0x80)

```

```

typedef struct _COLORPARAMS
{
    unsigned int brightness; // 0-255
    unsigned int saturation; // 0-255
    unsigned int hue;        // HUE_CODE_00, HUE_CODE_7F, or HUE_CODE_80
    unsigned int contrast;   // 0-255
} COLORPARAMS, *PCOLORPARAMS;

```

```

typedef struct _SENSORPARAMS // Must be set according to camera input type
{
    // NTSC PAL
    unsigned int top;        // 4    5
    unsigned int left;       // 0    4
    unsigned int height;     // 480 567
    unsigned int width;      // 640 640
} SENSORPARAMS, *PSENSORPARAMS;

```

**Rotation**

The camera output always operates in native landscape mode. Use the following rotation values to support other display and camera orientations:

Value	Rotation
0	No rotation
1	Vertical flip
2	Horizontal flip
3	180 deg
4	90 deg Right
5	90 deg Right with Vertical flip
6	90 deg Right with Horizontal flip
7	90 deg Left

**Frame Buffer (fbdev)**

```

#define FB_DEV_0 (0) // Graphics sent to /dev/fb0
#define FB_DEV_1 (1) // Graphics sent to /dev/fb1

```

**Header file**

camera.h

**Library**  
libghdrv.so

## API Functions

### Camera::Camera

#### Syntax

```
Camera::Camera(int camnum, int fbdev)
```

#### Parameters

```
int camnum
    [in]
    Camera Number. Valid model ranges: 1-2 (3D50), 1-3 (3D70), 1-4 (3D2104)
int fbdev
    [in]
    Frame buffer. FB_DEV_0 or FB_DEV_1
```

The “fbdev” value indicates whether the Graphics are sent to fb0 or fb1.

Graphics	Video	Foreground Allowed?	Background Allowed?
fb0	fb1	Yes	No
fb1	fb0	Yes	Yes

#### Return Value

```
int none
```

### Camera::setdisplayparams

Sets the following display window parameters:

- Origin (x and y coordinates of window top left corner)  
NOTE: the **left** parameter (top left window x-coordinate) requires 4 pixel alignment and therefore must be divisible by 4
- Size (height and width of window)
- Rotation
- Foreground or Background with color key

NOTE: when using background mode, the camera video only shows through where the graphics data is set to the color matching the specified color key. Graphics of any other color will appear on top of the camera video image.

#### Syntax

```
int Camera::setdisplayparams (DISPLAYPARAMS *p)
```

#### Parameters

```
DISPLAYPARAMS *p
    [in]
    Refer to DISPLAYPARAMS structure
```

### Return Value

int 0 = success, -1 = failure

## Camera::setcolorparams

Sets the following camera color parameters:

- brightness
- saturation
- contrast
- hue

### Syntax

```
int Camera::setcolorparams (COLORPARAMS *p)
```

### Parameters

COLORPARAMS            \*p  
                          [in]  
                          Refer to COLORPARAMS structure

### Return Value

int 0 = success, -1 = failure

## Camera::setsensorparams

Sets the camera sensor parameters

### Syntax

```
int Camera::setsensorparams (SENSORPARAMS *p)
```

### Parameters

SENSORPARAMS            \*p  
                          [in]  
                          Refer to SENSORPARAMS structure

### Return Value

int always 0

## Camera::show

Enables or disables the camera

### Syntax

```
int Camera::show(int enable)
```

### Parameters

int enable  
      [in]  
      1=enable, 0=disable

### Return Value

int 0 = success, -1 = failure

## Sample Code

```
#include "camera.h"

COLORPARAMS color;
DISPLAYPARAMS disp;
int cameranum = 1; // camera input 1

Camera cam(cameranum);

// configure display parameters
disp.top = 0;
disp.left = 80;
disp.height = 480;
disp.width = 640;
disp.rotate = 4; // rotate 90 degree right
disp.fg = FOREGROUND;
cam.setdisplayparams(&disp);

// camera on
cam.show(1);

// change color parameters
color.brightness = 50;
color.saturation = 128;
color.contrast = 128;
color.hue = 0;

// configure color parameters
cam.setcolorparams(&color);

....

// camera off
cam.show(0);
```

## Switched Input

The switched power input of the Grayhill 3Dxx displays can be used to put the display into a low power state. The switched input driver provides an interface for an application to (1) query the state of the switched input and (2) register for notification when the switched input turns off. These functions allow the application to perform any shutdown activities prior to the display powering off.

An application can interface to the switched input via software library API functions.

### Header file

ghilib.h

### Library

libghio.so

## API Functions

### ghiolib\_pwrswitchRegister()

Register to receive power switch off notification

#### Syntax

```
int ghiolib_pwrswitchRegister(void)
```

#### Parameters

none

#### Return Value

int 0 = success, -1 = failure

### ghiolib\_pwrswitchUnRegister()

Un-register for receiving power switch off notification

#### Syntax

```
int ghiolib_pwrswitchUnRegister(void)
```

#### Parameters

none

#### Return Value

int 0 = success, -1 = failure

### ghiolib\_pwrswitchQueryState (unsigned \*state)

Query the state of the power switch

#### Syntax

```
int ghiolib_pwrswitchQueryState(unsigned *state)
```

#### Parameters

```
unsigned *state  
         [out]  
         1=on, 0=off
```

#### Return Value

int 0 = success, -1 = failure

### ghiolib\_pwrswitchReleaseHold()

Release power switch hold, allowing display to enter low power mode

#### Syntax

```
int ghiolib_pwrswitchReleaseHold(void)
```



## Parameters

none

## Return Value

int 0 = success, -1 = failure

## Sample Code

```
#include ghiolib.h

static unsigned char exit_app = 0;

static void signalhandler(int sig)
{
    int ret_val;
    unsigned data = 0xAA;

    printf("Received signal from driver\n");

    // query the switch state
    ret_val = ghiolib_pwrswitchQueryState( &data );
    if ( ret_val != 0 )
        printf("", ret_val) ;
    else
        printf("power switch state = 0x%x\n", data) ;

    if ( data == 0 )
    {
        printf("SignalHandler: Switched power is off, entering LP mode in 2
secs\n");
        sleep(2);
        ret_val = ghiolib_pwrswitchReleaseHold();
        if (ret_val != 0)
            printf ("ghiolib_pwrswitchReleaseHold failed : %d\n", ret_val);

        /* signal app thread to exit */
        exit_app = 1;
    }
}

int  ret_val ;
unsigned data = 0xAA ;

// we will capture the signal
signal(SIGPWR, signalhandler);

// register for shutdown signal
ret_val = ghiolib_pwrswitchRegister();
if (ret_val != 0) {

    printf("TESTPWRSWITCH : Waiting for signal from driver\n");
    do
```

```
{
    sleep(1);
}
while ( !exit_app ) ;

// unregister for shutdown signal
ret_val = ghiolib_pwrswitchUnRegister();
if (ret_val != 0) {
    printf ("UnRegister with power switch driver failed : %d\n", ret_val);
}
}
```

## USB Flash

### Mounting/Unmounting USB Flash Drive:

To mount a USB flash drive, enter the following:

```
mount /mnt/usbhd
```

To unmount a USB flash drive, enter the following:

```
umount /mnt/usbhd
```

### Copying files to/from USB Flash:

To copy a file from the 3Dxx display to the USB flash drive, type the following:

```
cp <path/srcfile> /mnt/usbhd/<path/destfile>
```

where <path/srcfile> is the path and source filename  
and <path/destfile> is the path and destination filename

To copy a file from the USB flash drive to the 3Dxx display, type the following:

```
cp /mnt/usbhd/<path/srcfile><path/destfile>
```

## Available Utilities and Locations

This section illustrates several useful software utilities contained on the 3Dxx displays.

Utility: **getghpartnum.sh**

Folder: **/usr/bin**

Description: Reports the Grayhill part numbers of the following software components:  
bootloader (U-boot)  
linux OS (Kernel)  
linux file system (Rootfs)

Example:

```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~
root@ghiimx6:~# getghpartnum.sh
U-boot : 3DPR1942-1-C
Kernel : 3DPR1967-4-5
Rootfs : 3DPR1967-5-5
root@ghiimx6:~#
```

Utility: **getkernelversion.sh**

Folder: **/usr/bin**

Description: Reports the build date and time of the linux kernel

Example:

```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~
root@ghiimx6:~# getkernelversion.sh
Kernel version : Nov 15 2017 - 09:54:34
root@ghiimx6:~#
```

Utility: **getrootfsversion.sh**

Folder: **/usr/bin**

Description: Reports the build date and time of the linux rootfs

Example:

```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~
root@ghiimx6:~# getrootfsversion.sh
Rootfs version : Jun 29 2017 - 16:49:51
root@ghiimx6:~#
```

Utility: **getubootversion.sh**

Folder: **/usr/bin**

Description: Reports the build date and time of the U-boot bootloader

Example:

```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:~
root@ghiimx6:~# getubootversion.sh
U-boot version : Apr 28 2017 - 11:15:24
root@ghiimx6:~#
```

Utility: **fbgrab.exe**

Folder: **/bin**

Description: Takes a screenshot of the frame buffer and converts it to a .png format image.

## Ethernet Configuration

The default method for the display to obtain an IP address is DHCP. To find the ethernet IP address of the 3Dxx Display, enter the following command:

- `ifconfig eth0`

If the display has a valid IP address, it will be displayed after the “inet addr:” tag.



```
root@ghiimx6:~  
root@ghiimx6:~# ifconfig eth0  
eth0      Link encap:Ethernet  HWaddr 00:10:25:0C:74:90  
          inet addr:192.168.40.74  Bcast:192.168.40.255  Mask:255.255.255.0  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:443 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:46741 (45.6 KiB)  TX bytes:978 (978.0 B)  
root@ghiimx6:~#
```

If the “inet addr:” tag is not present and a valid IP Address is not returned, restart the ethernet connection using the following commands:

- `ifdown eth0`



```
root@ghiimx6:~  
root@ghiimx6:~# ifdown eth0  
root@ghiimx6:~#
```

- `ifup eth0`

```
root@ghiimx6:~  
root@ghiimx6:~# ifup eth0  
eth0: Freescale FEC PHY driver [Generic PHY] (mii_bus;phy_addr=1:01, irq=-1)  
udhcpc (v1.18.5) started  
root@ghiimx6:~# Sending discover...  
PHY: 1:01 - Link is Up - 100/Full  
Sending discover...  
Sending discover...  
Sending select for 192.168.40.74...  
Lease of 192.168.40.74 obtained, lease time 259200  
deleting routers  
route: SIOCDELRT: No such process  
adding dns 192.168.60.41  
adding dns 192.168.60.44  
root@ghiimx6:~#
```

Then try entering the `ifconfig eth0` command again.

The DHCP method for IP address assignment is configured with the following line in file 'interfaces', which is located in folder `/etc/network`:

```
iface eth0 inet dhcp
```

To change from DHCP to a static IP address, modify the interfaces file as follows:

```
iface eth0 inet static  
    address <ipaddress>  
    netmask <subnet>
```

For example:

```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:/etc/network
root@ghiimx6:/etc/network cat interfaces
#
# /etc/network/interfaces
#
auto can0 can1 eth0

# onboard CAN
iface can0 inet static
    address 127.42.23.180
    netmask 255.255.255.0
    pre-up /etc/network/can-pre-up

iface can1 inet static
    address 127.42.23.181
    netmask 255.255.255.0
    pre-up /etc/network/can-pre-up

iface eth0 inet static
    address 192.168.40.111
    netmask 255.255.255.0
root@ghiimx6:/etc/network █
```

## Copying Files from Ethernet

Files can be copied between the 3Dxx display and the Virtual Box over the ethernet connection using the Secure Copy (scp) command.

```
scp<source filespec>root@<ipaddress>: <dest filespec>
```

For example:

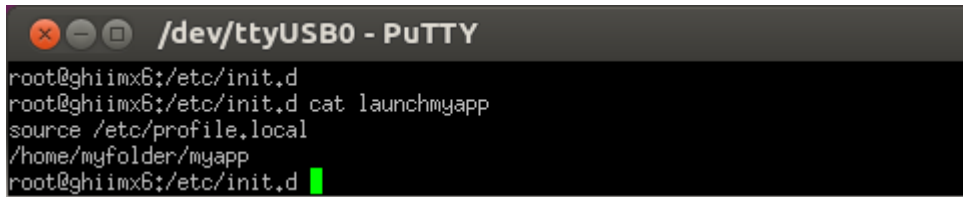
```
ghguest@ghguest-VirtualBox: ~/build-gh7indemo-Qt_5_9_3_3Dxx-Debug
File Edit View Search Terminal Help
ghguest@ghguest-VirtualBox:~/build-gh7indemo-Qt_5_9_3_3Dxx-Debug$
ghguest@ghguest-VirtualBox:~/build-gh7indemo-Qt_5_9_3_3Dxx-Debug$ scp gh7indemo root@192.168.40.102:/home/demo7in
gh7indemo
100% 6477KB 6.3MB/s 00:01
ghguest@ghguest-VirtualBox:~/build-gh7indemo-Qt_5_9_3_3Dxx-Debug$ █
```

The above example copies file gh7indemo to folder demo7in on the display with ip address 192.168.40.102.

## Configuring an Application to Run at Bootup

There are 2 steps to configuring an application to run automatically at bootup:

1. Create a launch script in folder /etc/init.d for starting the application
  2. Create a soft link in folder /etc/rc.d for running the launch script
- Create launch script  
In folder /etc/init.d on the display, create a launch script.  
For example, launch script *launchmyapp*.



```
/dev/ttyUSB0 - PuTTY
root@ghiimx6:/etc/init.d
root@ghiimx6:/etc/init.d cat launchmyapp
source /etc/profile.local
/home/myfolder/myapp
root@ghiimx6:/etc/init.d █
```

- Create soft link  
In folder /etc/rc.d on the display, create a soft link to the launch script.  
For example, soft link *S20myapp*:

```
ln -s /etc/init.d/launchmyapp /etc/rc.d/S20myapp
```

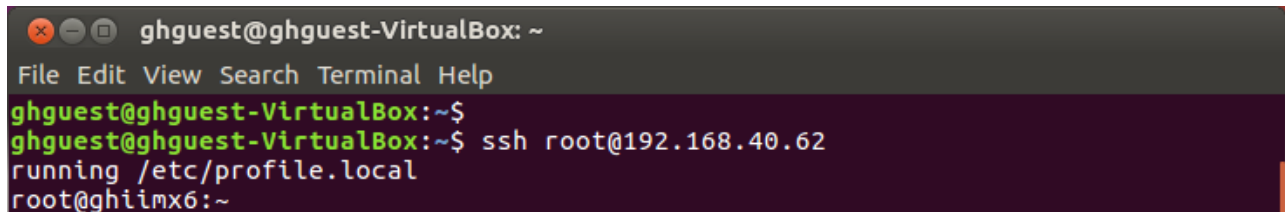
## Secure Shell (ssh)

The Linux secure shell command `ssh` can be used to log into the display remotely over ethernet. The `ssh` command is useful for transferring files and executing terminal commands on the display. The following command will open a terminal connection to the display with `ssh`:

```
ssh root@ipaddress
```

For example:

```
ssh root@192.168.40.62
```

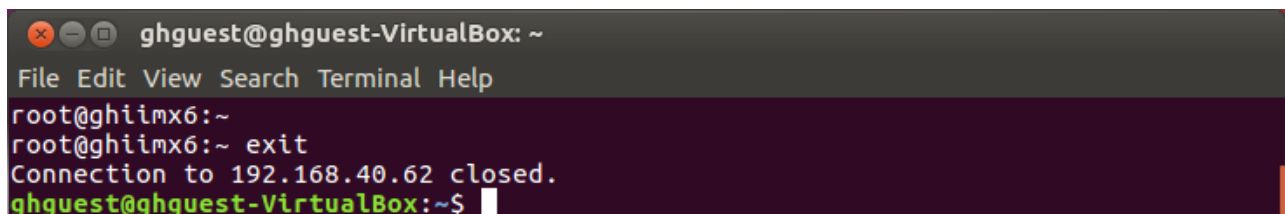


```
ghguest@ghguest-VirtualBox: ~
File Edit View Search Terminal Help
ghguest@ghguest-VirtualBox:~$
ghguest@ghguest-VirtualBox:~$ ssh root@192.168.40.62
running /etc/profile.local
root@ghiimx6:~
```

The prompt has now changed to “`root@ghiimx6:~`” because it is now running as a terminal on the display. Linux shell commands can now be run on the display.

To disconnect the `ssh` terminal, enter the command:

```
exit
```



```
ghguest@ghguest-VirtualBox: ~
File Edit View Search Terminal Help
root@ghiimx6:~
root@ghiimx6:~ exit
Connection to 192.168.40.62 closed.
ghguest@ghguest-VirtualBox:~$ █
```

## Displaying “.raw” Image Files

The utility FFmpeg, a free download available from <https://www.ffmpeg.org> can be used for converting .bmp or .png image files to .raw format. A .raw image file can be displayed on the 3Dxx display by writing it directly to the frame buffer.

The following command lines show how to use FFmpeg to convert image files to .raw format.

Enter the following command line for converting from .png to .raw:

```
ffmpeg -vcodec png -i infile.png -vcodec rawvideo -f rawvideo -pix_fmt rgb565
outfile.raw
```

where:

*infile* is the .bmp file to convert  
*outfile* is the converted .raw file

Enter the following command line for converting from .bmp to .raw:

```
ffmpeg -vcodec bmp -i infile.bmp -vcodec rawvideo -f rawvideo -pix_fmt rgb565
outfile.raw
```

where:

*infile* is the .bmp file to convert  
*outfile* is the converted .raw file

A .raw file can be written to the frame buffer with the following command line:

```
cat filename.raw > /dev/fb0
```

NOTE: in order to be displayed properly, the raw file must be correctly sized to the frame buffer. Therefore, the dimensions of the .bmp or .png file used to generate the .raw file should be as follows:

Display Model	.bmp / .png Dimensions
3D50	800x480
3D70	800x480
3D2104	1024x768

## Frame Buffer Alpha Blending

The 3Dxx display provides for alpha blending between the 2 frame buffers (fb0, and fb1).Frame buffer fb1 is 'on top of' fb0. This allows, for example, text in fb1 to overlaygraphics in fb0.

The alpha setting is a value between 0x00 and 0xFF, and is interpreted as follows:

alpha      fb0      fb1



0x00	100%	0%
...		
0x80	50%	50%
...		
0xFF	0%	100%

By default, the alpha setting is 0x80, which is 50% blending between fb0 and fb1.

## Script Files

Linux script file use UNIX-style line endings, i.e., each line is terminated with a Line Feed. With a DOS or Windows style line ending, the line endings are Carriage Return and Line Feed. If a UNIX-style line ending is not used, the script will not function correctly.

## Appendix

### Updating Files on the Device Using the Recovery OS

There is a Recovery OS mechanism on the 3Dxx Series displays for updating files on the device. This mechanism is typically used for updating the linux kernel and root file system. The following steps describe how to activate the recovery OS.

- At the root of a USB flash drive, create a folder called:  
GH3D\_scripts
- In this folder, place script file called:  
recos\_startup.sh  
The recos\_startup script can be customized to perform limited operations such as copying files from the USB flash drive to the display.
- With the display powered off, plug the USB flash drive into the display then power it up. When the display is powered up, it will detect whether or not a USB flash drive is present. If so, it will then look for a folder named GH3D\_scripts in the root folder of the USB flash drive. If the folder is present, it will then look for a script named recos\_startup.sh. If the script is found, it will be launched.

### How to Disable the Recovery OS

The Recovery OS launcher can be disabled if so desired. This is done by setting the 'ubootusb' environment variable with the value 'ignore'. For example:

```
root@ghiimx6:~ ghsetenv ubootusb ignore
```

Reading it back will show:

```
root@ghiimx6:~ ghprintenv ubootusb  
ubootusb=ignore
```

To enable the Recovery OS launcher if it has been disabled, remove the 'ubootusb' environment variable as follows:

```
root@ghiimx6:~ ghsetenv ubootusb
```

Reading it back will show:

```
root@ghiimx6:~ ghprintenv ubootusb  
## Error: "ubootusb" not defined
```