Boot Software Updater (BSU)

Broadcom Corporation
5300 California Avenue
Irvine, California, USA 92617
Phone: 949-926-5000
Fax: 949-926-5203

Web: www.broadcom.com

Revision History

Revision	Date	Change Description
TBD	11/05/2013	Initial version, from A. Gin.
TBD	1/6/2014	BOLT and Security support.
TBD	2/15/2014	Updated “Setting up for the first time” section.
14.2.1	4/30/2014	Updated copying rockford\bsp sub-directories from CFE 14.1
14.3.1	9/2/2014	Updated for phase 14.3. Added BSU GUI. Added IR Input test. Overall review of the document.
15.1.1	2/10/2015	Add memory maps
15.1.2	2/16/2015	Fix memory map for ARM MEMC0
16.1.1	1/12/2016	Fix boot description for ARM (Section J).

Please send any comments, errors, or suggestions related to this document to agin@broadcom.com

Table of Contents

A. Introduction

BSU is a specialized Nexus based set-top box application intended to provide a framework for developing custom boot software updater applications. BSU feature set is limited to frontend, transport, graphics, and peripherals. BSU also provides services to Flash, Ethernet, and USB. The targeted feature set allows for a small memory footprint to meet the desired objective. To reduce the memory footprint, BSU is designed to use the uC/OS kernel.

B. Intended Audience

This document is intended for software developers designing software boot updater applications for a set-top box. Software boot updater applications are specialized applications designed for updating software/firmware for customer premise equipment, in this case, a set-top box. This document makes references to the CFE/BOLT boot loader, Nexus Middleware, and Magnum PortingInterface software packages. Detail descriptions of these software packages are outside the scope of this document. Please see the appropriate documentation provided with these software packages for additional information.

C. BSU Supported Platforms

See release notes for the URSR Release for the list of designated and non-designated platforms.

D. Requirements to Get Started

Here is what is needed to get started:

- Broadcom Reference Platform listed on the “BSU Supported Platforms” section of this document.
- Unified Reference Software Release Phase 13.4 or later
- Linux server with SDE 5.03 Toolchain for MIPS based platforms or ARM GNU Toolchain 4.5.4 for ARM based platforms.
- BOLT/CFE Source Release for the applicable platform.

E. BSU Features

Compact Size

BSU relies on the following to maintain the smallest possible size:

1. Small RTOS. BSU runs on top of Micrium's uC/OS-III real time operating system. The binary image size of this is 12KB.

2. Customizable Features List (CFL). This list is a single file, specifying which features are supported, but with the option of being disabled at compile time. CFL allows boot updater developers to easily disable features not required for their application. Using CFL eliminates the need to modify the source code.

3. Fully utilizes optimized boot updater porting interface modules.

4. Constant evolution of finding code which could be reduced or eliminated. We will find these areas, and we will include them so that our customers won't have to spend time, painstakingly shaving a few hundred bytes here or there, and having to remember to fold these into their custom builds.

Availability

BSU will be available as part of the Unified Reference Software Release, starting with URSR Phase 13.4. Please see "BSU Supported Platforms" section for a list of supported platforms.

BSU API

The BSU API covers areas which the Nexus API and uC/OS-III API do not cover, but are needed to complete boot loader software updater functionality. These areas include:

1. Flash Interface. Called BSU Flash Interface, this software interface is a series of API functions which will allow the user to open/close/read/write all of the different types of Flash.

2. USB Interface. Called BSU USB Interface, this software interface is a series of API functions which will allow the user to open/close/read from USB. Currently, write is not supported.

3. Ethernet Interface. Called BSU Net Interface, this software interface is a series of API functions which will allow the user to initialize the ethernet interface, perform DHCP, and access files across a network via TFTP.

4. Memory Interface. Called BSU Memory Interface. Allocate and free memory. Similar to malloc and free.

5. File System Interface. Called BSU File System Interface. Open, close, read, write, and seek files with FAT32 or a "raw" file system.

6. Security Interface. Called BSU Security Interface. Allows the ability to authenticate and/or decrypt an image in RAM.

Nexus API

BSU is designed for the Nexus middleware. Therefore, a limited Nexus API set (based on CFL configuration) is available when developing BSU applications. Nexus API is supported on a large number of Broadcom Set-top Box SoCs.

uC/OS API

The uC/OS API is a Micrium defined interface. The binary for this RTOS is provided free of charge to our customers, for all MIPS and ARM based set-top projects. For source code, the customer will need to obtain a license directly from Micrium. The BSU build system will allow the ability to link to the uC/OS binary by default, or compile the uC/OS source code as an option.

GUI App

The purpose of the GUI app is to demonstrate what a simple BSU application can look like.

Menu Driven Test App

The purpose of the test app is two-fold:

1. a reference to show how to interface to the Nexus, BSU, and uC/OS API's.

2. to prove that the software and hardware works.

BOLT/CFE Command Shell

The BOLT/CFE command shell is provided from within the Menu Driven Test App as a menu option. The purpose of the BOLT/CFE command shell is to test different BOLT/CFE functions which may be of interest to the developer. For example, from within the BOLT/CFE command shell, you may perform such functions as "ifconfig", "ping", and "show devices", to name just a few.

F. BSU Overview

BSU Software Stack

The figure below shows how BSU fits into Broadcom's overall software architecture for cable, satellite and IP set-top boxes:

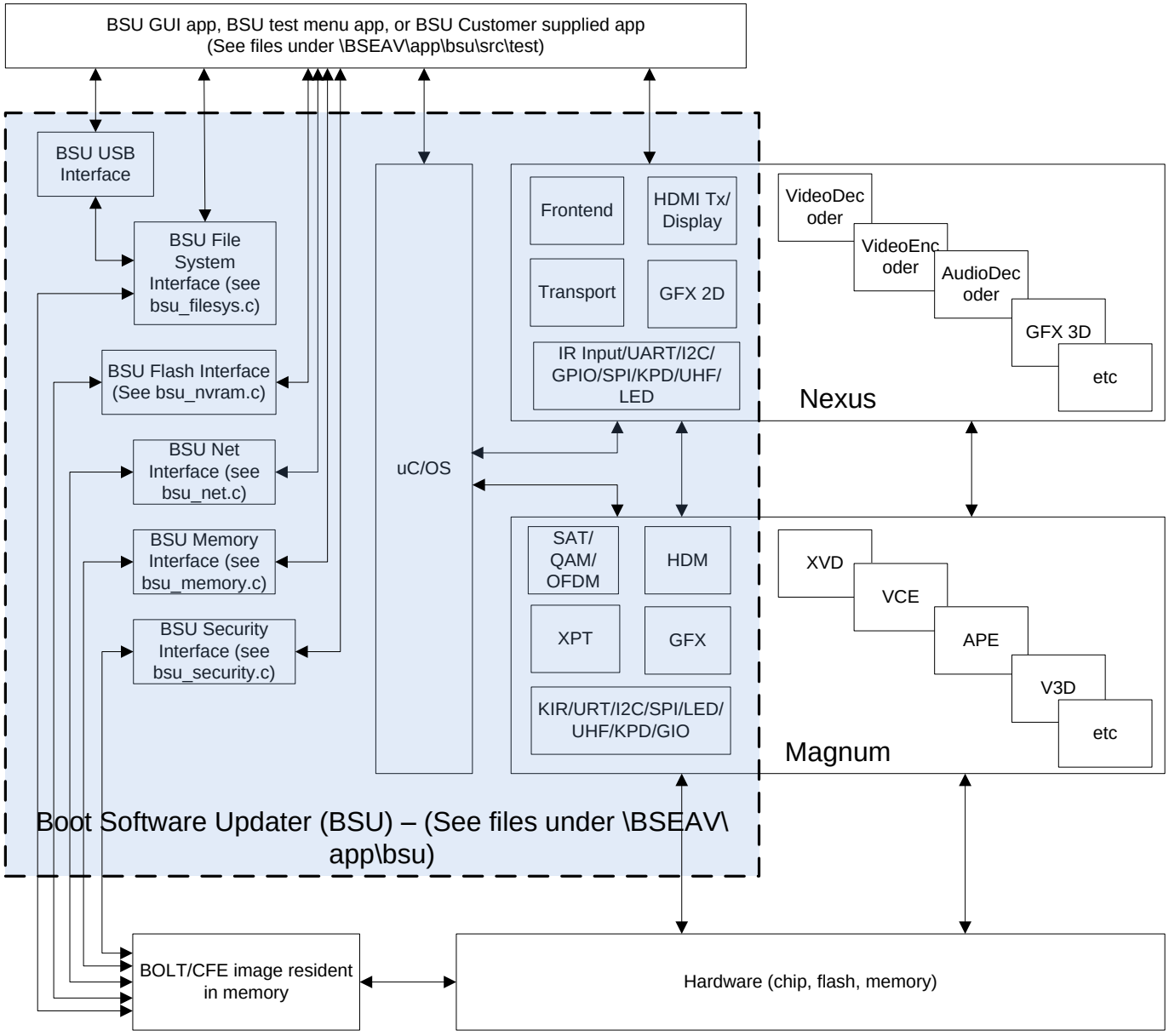


Figure 1: BSU and BSU Test Application Software Architecture

The components of the overall software architecture are shown in the following table:

Component	Function
Magnum	Low-level software modules that directly interacts with hardware and firmware. Within Magnum, there are porting interface modules, base modules, common utilities and higher-level syslibs.
Nexus	High-level software modules that integrate Magnum functionality and provide easy-to-use interfaces for end-user middleware and applications. For more information, please refer to the Nexus Architecture document.
BSU	See table below.
BOLT/CFE	BOLT/CFE image resident in memory.
uC/OS	The Micrium uC/OS real-time operating system binary. Source code available upon purchase from Micrium.

Table 1: Overall Software Architecture

BSU Sub-Component	Function
BSU Flash Interface	BSU provided functions to access flash.
BSU USB Interface	BSU provided functions to access USB.
BSU Ethernet Interface	BSU provided functions to access the Ethernet.
BSU Memory Interface	BSU provided functions to access memory.
BSU File System Interface	BSU provided functions to access data via a file system (i.e., FAT32 or raw)
BSU Security Interface	BSU provided functions to authenticate an image in RAM.
GUI Application	Sample GUI application
Menu Driven Test Application	Main menu test application from which one could choose additional test options, such as shown in the following table:

Table 2: BSU Sub-Components

Test	Description
Frontend Test	Demonstrate frontend functionality (ie., satellite, cable, off-air), based on the platform, via Nexus.
Message Test	Demonstrate ability to receive messages through the transport interface, via Nexus.
Colorbar Test	Show basic colorbars to prove 2d graphic support, via Nexus.
Flash Test	Demonstrates how to use the BSU Flash Interface, and, based on the type of flash (NAND/NOR/SPI) on board, will allow reading and writing to flash.
Ethernet Test	Demonstrates ifconfig, DHCP, and TFTP across an Ethernet interface.
USB Test	Demonstrates reading from a USB mass storage device via either FAT32 or raw access.
UART Test	Demonstrates sending and receiving characters across a different UART interface than the standard UART0.
Security Test	Demonstates the ability to authenticate and/or decrypt an image in RAM.
BOLT/CFE Command Shell	For debug purposes, allows user to enter various BOLT/CFE commands on this shell. For example, “ifconfig eth0 –auto”.

Table 3: Menu Driven Tests

F. BSU API

BSU Flash API

```
/**************************************************************************
 *
 *  bsu_nvram_open()
 *
 *  Open bsu interface to nvram.
 *
 **************************************************************************/
int bsu_nvram_open(void);

 /**************************************************************************
 *
 *  bsu_nvram_close()
 *
 *  Close bsu interface to nvram.
 *
 **************************************************************************/
int bsu_nvram_close(void);

 /**************************************************************************
 *
 *  bsu_nvram_read()
 *
 *  bsu nvram read interface
 *
 **************************************************************************/
int bsu_nvram_read(unsigned char *buffer,int offset,int length);

 /**************************************************************************
 *
 *  bsu_nvram_write()
 *
 *  bsu nvram write interface
 *
 **************************************************************************/
int bsu_nvram_write(unsigned char *buffer,int offset,int length);

 /**************************************************************************
 *
 *  bsu_nvram_erase()
 *
 *  bsu nvram erase interface
 *
 **************************************************************************/
int bsu_nvram_erase(void);
```

Example Code

The following is a simple BSU application illustrating the use of the BSU Flash API (error handling omitted for clarity):

```
int res;
uint8_t buffer[512];

strcpy(partition_name, "flash0.nvram");
cfe_set_devname(partition_name);
res = bsu_nvram_open();

/* read from flash */
res = bsu_nvram_read(buffer,read_offset,sizeof(buffer));

/* write to flash */
memset(buffer, 0, sizeof(buffer));
res = bsu_nvram_write(buffer,write_offset,sizeof(buffer));
```

BSU File System API

```
/***************************************************************************
 *
 *  bsu_fs_init()
 *
 *  BSU filesystem init
 *
 ***************************************************************************/
int bsu_fs_init(char *fsname,fileio_ctx_t **fsctx,void *device);

 /***************************************************************************
 *
 *  bsu_fs_uninit()
 *
 *  BSU filesystem uninit
 *
 ***************************************************************************/
int bsu_fs_uninit(fileio_ctx_t *fsctx);

 /***************************************************************************
 *
 *  bsu_fs_open()
 *
 *  BSU open interface
 *
 ***************************************************************************/
int bsu_fs_open(fileio_ctx_t *fsctx,void **ref,char *filename,int mode);

 /***************************************************************************
 *
 *  bsu_fs_close()
 *
 *  BSU close interface
 *
 ***************************************************************************/
int bsu_fs_close(fileio_ctx_t *fsctx,void *ref);

 /***************************************************************************
 *
 *  bsu_fs_seek()
 *
 *  BSU seek interface
 *
 ***************************************************************************/
int bsu_fs_seek(fileio_ctx_t *fsctx,void *ref,int offset,int how);

 /***************************************************************************
 *
 *  bsu_fs_read()
 *
 *  BSU read interface
 *
 ***************************************************************************/
int bsu_fs_read (fileio_ctx_t *fsctx,void *ref,uint8_t *buffer,int len);
```

```
/**************************************************************************
*
*  bsu_fs_write()
*
*  BSU write interface
*
**************************************************************************/
int bsu_fs_write(fileio_ctx_t *fsctx,void *ref,uint8_t *buffer,int len);
```

Example Code

The following is a simple BSU application illustrating the use of the BSU File System API:

1. A BSU File System API example to transfer a file from across the network, from the file, bsu_ethernet_test.c:

```
char *fname="stb-irva-09:/agin/src/a.txt";
fileio_ctx_t *fsctx;
void *filectx;
uint8_t buffer[1024];
int res;
int total;
int count;


res = bsu_fs_init("tftp",&fsctx,"");
if (res < 0) {
    printf("Could not init file system\n");
    return -1;
}

res = bsu_fs_open(fsctx,&filectx,fname,FILE_MODE_READ);
if (res < 0) {
    printf("Could not open %s",fname);
    return -1;
}
else {
total = 0;
count = 0;
for (;;) {
    res = bsu_fs_read(fsctx,filectx,buffer,sizeof(buffer));
    if (res < 0) break;

    /* print out the bytes in a nice format */
    printf("\n");
    if (res > 0) {
        int i,j;
        for (i=0; i<res; i++) {
            printf("%02x", buffer[i]);
```

```
            if (((i+1)%16==0)) {
                printf(" : ");
                for (j=i-15; j<res; j++) {
                    printf("%c", buffer[j]);
                    if ((j+1)%16==0) {
                        break;
                    }
                }
                printf("\n");
            }
            else if (i==res-1) {
                int k, rem;
                rem=(16-res%16);
                for (k=0; k<rem; k++)
                    printf("   ");
                printf(" : ");
                for (j=i-(15-rem); j<res; j++)
                    printf("%c", buffer[j]);
            }
        }
        printf("\n\n");
    }

    total += res;
    if (res != sizeof(buffer)) break;
    count++;
    if (count == 256) {
    printf(".");
    count = 0;
    }
    }
if (res < 0) printf("read error %s\n",bsu_errortext(res));
else printf("Total bytes read: %d\n",total);
bsu_fs_close(fsctx,filectx);
}

bsu_fs_uninit(fsctx);
```

BSU USB API

Accessing the USB is performed indirectly via the BSU File System API

Example Code

The following is a simple BSU application illustrating the use of accessing the USB via the BSU File System API. The USB device is recognized by BOLT/CFE as 'usbdisk0':

```
int count=0;
int res;
void *ref;
uint8_t buffer[512*8];
fsctx_usb0 = 0;

if (bsu_fs_init("fat",&fsctx_usb0,"usbdisk0") == 0) {
    printf("usb initialized with fat filesystem\n");
}
else {
    printf("error from bsu_fs_init\n");
    fsctx_usb0 = 0;
}

res = bsu_fs_open(fsctx_usb0, &ref, (char *)fname, FILE_MODE_READ);
if (res < 0) {
    printf("error from bsu_fs_open\n");
    break;
}

res = bsu_fs_seek(fsctx_usb0, ref, 0, FILE_SEEK_BEGINNING);
if (res < 0) {
    printf("error from bsu_fs_seek\n");
    break;
}

res = bsu_fs_read (fsctx_usb0, ref, buffer, sizeof(buffer));
if (res <0 ) break;

for (;;)
{
    res = bsu_fs_read(fsctx_usb0, ref, buffer, sizeof(buffer));
    if (res < 0) break;
    if (res != sizeof(buffer)) break;
    if( det_in_char() > 0 ) break;
    count++;
    if ((count % 256) == 0)
    {
        printf(".");
        fflush(stdout);
    }
}
```

```
res = fs_close(fsctx_usb0, ref);
if (res < 0) {
    printf("error from fs_close\n");
}
```

BSU Ethernet API

```
/**************************************************************************
 *
 *  bsu_net_init()
 *
 *  BSU network init
 *
 **************************************************************************/
int bsu_net_init(char *devname);

 /**************************************************************************
 *
 *  bsu_net_uninit()
 *
 *  BSU network uninit
 *
 **************************************************************************/
void bsu_net_uninit(void);

 /**************************************************************************
 *
 *  bsu_do_dhcp_request()
 *
 *  BSU DHCP request
 *
 **************************************************************************/
int bsu_do_dhcp_request(char *devname);
```

Example Code

The following is a simple BSU application illustrating the use of the BSU Ethernet API. This application will perform an "ifconfig eth0 –auto", obtaining an IP address with DHCP:

```
    int err;
    char *devname = "eth0";

    bsu_net_uninit();

    err = bsu_net_init(devname);
    if (err < 0) {
    printf("Could not activate device %s: %s\n",
        devname,bsu_errortext(err));
    return err;
    }

    bsu_do_dhcp_request(devname);
```

BSU Memory API

```
/**************************************************************************
 *
 *  bsu_malloc()
 *
 *  Allocate memory
 *
 **************************************************************************/
void *bsu_malloc(size_t size);

 /**************************************************************************
 *
 *  bsu_free()
 *
 *  Free memory
 *
 **************************************************************************/
void bsu_free(void *ptr);
```

Example Code

The BSU Memory API access memory via the resident BOLT/CFE image. Please refer to the CFE source code.

BSU Security API

Allow the ability to authenticate an image in RAM.

```
/**********************************************************************
 *
 *  bsu_security_region_verify()
 *
 *  Authenticate an image in memory.
 *
 **********************************************************************/
void bsu_security_region_verify(unsigned long region_end_address, unsigned long
sig_start_addr, int bVerify, int bDecrypt);
```

G. NEXUS API

Please refer to the Nexus documents located in the Unified Reference Software Release, in the following directory location:

```
bash-3.2$ pwd
/fe_lab/agin/src/URSR_Latest/nexus/docs
bash-3.2$ ls
Nexus_Architecture.pdf  Nexus_Memory.pdf        Nexus_SurfaceCompositor.pdf
OpenGL_ES_QuickStart.pdf
Nexus_Development.pdf   Nexus_MultiProcess.pdf  Nexus_Transcode.pdf
RefswDebuggingGuide.pdf
Nexus_Graphics.pdf      Nexus_Overview.pdf      Nexus_Usage.pdf
RefswPowerManagementOverview.pdf
bash-3.2$
```

Example Code

The following is a simple BSU application utilizing the Nexus API:

1. A Nexus API example to lock to the satellite frontend, from the file, bsu_satellite_test.c:

```
NEXUS_Platform_GetConfiguration(&platformConfig);

NEXUS_Frontend_GetDefaultAcquireSettings(&settings);
settings.capabilities.satellite = true;
frontend = NEXUS_Frontend_Acquire(&settings);
if (!frontend) {
    fprintf(stderr, "Unable to find satellite-capable frontend\n");
    return;
}

NEXUS_Frontend_GetDefaultSatelliteSettings(&satSettings);
satSettings.frequency = FREQ;
satSettings.mode = SATELLITE_MODE;
satSettings.lockCallback.callback = lock_callback;
satSettings.lockCallback.context = frontend;
NEXUS_Frontend_GetUserParameters(frontend, &userParams);

/* Map a parser band to the demod's input band. */
parserBand = PARSER_BAND;
```

```
    NEXUS_ParserBand_GetSettings(parserBand, &parserBandSettings);
    if (userParams.isMtsif) {
        parserBandSettings.sourceType = NEXUS_ParserBandSourceType_eMtsif;
        parserBandSettings.sourceTypeSettings.mtsif = NEXUS_Frontend_GetConnector(frontend);
/* NEXUS_Frontend_TuneXyz() will connect this frontend to this parser band */
    }
    else {
        parserBandSettings.sourceType = NEXUS_ParserBandSourceType_eInputBand;
        parserBandSettings.sourceTypeSettings.inputBand = userParams.param1;  /* Platform
initializes this to input band */
    }
    parserBandSettings.transportType = TRANSPORT_TYPE;
    NEXUS_ParserBand_SetSettings(parserBand, &parserBandSettings);

    NEXUS_Frontend_GetDiseqcSettings(frontend, &diseqcSettings);
    diseqcSettings.toneEnabled = TONE_MODE;
    diseqcSettings.voltage = VOLTAGE;
    NEXUS_Frontend_SetDiseqcSettings(frontend, &diseqcSettings);
    printf("Set DiseqcSettings\n");

    rc = NEXUS_Frontend_TuneSatellite(frontend, &satSettings);
    BDBG_ASSERT(!rc);

    while (1)
    {
        NEXUS_FrontendSatelliteStatus satStatus;
        rc = NEXUS_Frontend_GetSatelliteStatus(frontend, &satStatus);
        if (rc) {
            printf("unable to read status\n");
        }
        else {
            printf(
                "Sat Status\n"
                "  symbolRate %d\n"
                "  locked  %d\n",
                    satStatus.settings.symbolRate,
                    satStatus.demodLocked);
            if(satStatus.demodLocked)
                break;
        }
        BKNI_Sleep(1000);
    }
```

This example locks to the downstream channel, and waits in a loop, printing out status every second.

H. uC/OS API

Please refer to the Micirum uC/OS-III User's Manual, located here:

```
bash-3.2$ pwd
/fe_lab/agin/src/micrium-a9/Micrium/Software/uCOS-III/Doc
bash-3.2$ ls
Micrium-uCOS-III-UserManual.pdf
bash-3.2$
```

I. Building BSU

Overview

BSU is built as a separate, compressed binary and ELF application. This application can be booted from the network via TFTP, or over USB, or from flash. The TFTP method is the preferred method, to be used during development, due to the speed with which to load the code. The flash method is the preferred method when putting together the final, production worthy, boot software updater.

Setting up for the first time

- For MIPS based platforms
    - On your Linux server, make sure that you have the SDE 5.03 Toolchain installed. By default, the makefile will look for the toolchain under /opt/toolchains/sde_5.03, in order to link to the libc library. You can override this location by setting TOOLCHAIN_PATH to a different location. For example:

```
bash-3.2$ export TOOLCHAIN_PATH=/opt/toolchains/sde_5.03
```

    - Also, make sure that your path points to this location. For example:

```
bash-3.2$  set path = ( $path /opt/toolchains/sde_5.03/bin )
```

- For ARM based platforms
    - On your Linux server, make sure that you have the STBGCC 4.5.4 Toolchain installed. By default, the makefile will look for the toolchain under /opt/toolchains/stbgcc-4.5.4-2.5, in order to link to the libgcc library. You can override this location by setting TOOLCHAIN_PATH to a different location. For example:

```
bash-3.2$ export TOOLCHAIN_PATH=/opt/toolchains/stbgcc-4.5.4-2.5
```

- Also, make sure that your path points to this location. For example:

```
bash-3.2$  set path = ( $path /opt/toolchains/stbgcc-4.5.4-2.5/bin )
```

- Place the Unified Reference Software Release on a directory that your Linux server can build from.
- As a result of this step, the top-level directory should have the following sub-directories:

```
bash-3.2$ ls -l
total 167180
drwxrwxr-x 10 agin users      4096 Jul 29 10:20 BSEAV
drwxrwxr-x  6 agin users      4096 Jul 29 10:21 magnum
drwxrwxr-x 13 agin users      4096 Jul 29 10:21 nexus
drwxrwxr-x 21 agin users      4096 Jul 29 10:21 rockford
bash-3.2$
```

- Obtain and untar the BOLT and/or CFE source release package in this same location.

CFE

In the case of the CFE source release package at the time of URSR 14.3 development (v14.3), the top-level directory should have the following sub-directories (after un-tarring the CFE source release package):

```
bash-3.2$ ls -l
total 167180
drwxr-xr-x  3 agin users      4096 Aug 11 11:22 BRCM_CFE_Generic_14.3_E1
-rwxr--r--  1 agin users 157513768 Jul  9 00:42 BRCM_CFE_Generic_14.3_E1.zip
drwxrwxr-x 10 agin users      4096 Jul 29 10:20 BSEAV
drwxrwxr-x  6 agin users      4096 Jul 29 10:21 magnum
drwxrwxr-x 13 agin users      4096 Jul 29 10:21 nexus
drwxrwxr-x 21 agin users      4096 Jul 29 10:21 rockford
bash-3.2$
```

And the directory path to the relevant CFE source files may look like this:

```
bash-3.2$ ls -l
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/src/unified_cfe_v14.3/v14.3/CFE_V14_3/
total 40
drwxr-xr-x 16 agin users 4096 Aug 11 11:24 avs
drwxr-xr-x  3 agin users 4096 Aug 11 11:24 bsp
drwxr-xr-x 14 agin users 4096 Aug 11 11:24 build
drwxr-xr-x 11 agin users 4096 Aug 11 11:24 cfe
drwxr-xr-x  4 agin users 4096 Aug 11 11:24 opencable
drwxr-xr-x 14 agin users 4096 Aug 11 11:24 rdb
drwxr-xr-x  2 agin users 4096 Aug 11 11:24 scripts
drwxr-xr-x  5 agin users 4096 Aug 11 11:24 security
drwxr-xr-x  5 agin users 4096 Aug 11 11:25 shmoo
drwxr-xr-x  5 agin users 4096 Aug 11 11:25 splash
bash-3.2$
```

The path information to the relevant CFE source files is used to set the CFE path information in the makefile properly:

Change the Makefile located in \BSEAV\app\bsu\build to reflect the correct path:

```
include ../../../../nexus/build/os/$(B_REFSW_OS)/os_tools.inc

BOLT_DIR_NAME = BOLT
CFE_DIR_NAME =
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/src/unified_cfe_v14.3/v14.3/CFE_V14_3
BSEAV = $(shell cd "../../.." && ${PWD})
```

BOLT

In the case of the BOLT source release package at the time of URSR 14.3 development (v0.88), the top-level directory should have the following sub-directories (after un-tarring the CFE source release package):

```
bash-3.2$ ls -l
total 167180
drwxr-xr-x  3 agin users      4096 Aug 11 16:58 BRCM_BOLT_Generic_0_0_88_E1
-rwxr--r--  1 agin users  13440949 Aug  8 21:34 BRCM_BOLT_Generic_0_0_88_E1.zip
drwxrwxr-x 10 agin users      4096 Jul 29 10:20 BSEAV
drwxrwxr-x  6 agin users      4096 Jul 29 10:21 magnum
drwxrwxr-x 13 agin users      4096 Jul 29 10:21 nexus
drwxrwxr-x 21 agin users      4096 Jul 29 10:21 rockford
bash-3.2$
```

And the directory path to the relevant BOLT source files may look like this:

```
bash-3.2$ ls -l
BRCM_BOLT_Generic_0_0_88_E1/release_bolt_v0.88/src/BRCM_Src_BOLT_Generic_0_0_88_E1/bolt_v0.
88/
total 96
-rwxr--r--  1 agin users 7677 Aug  8 21:29 Makefile
-rwxr--r--  1 agin users 6919 Aug  8 21:29 Makefile.Bolt
drwxr-xr-x  5 agin users 4096 Aug 11 16:58 avs
drwxr-xr-x  2 agin users 4096 Aug 11 16:58 build
drwxr-xr-x  2 agin users 4096 Aug 11 16:58 common
drwxr-xr-x  3 agin users 4096 Aug 11 16:58 config
drwxr-xr-x  2 agin users 4096 Aug 11 16:58 custom
drwxr-xr-x  2 agin users 4096 Aug 11 16:58 doc
drwxr-xr-x  6 agin users 4096 Aug 13 11:44 dtc
drwxr-xr-x  2 agin users 4096 Aug 11 16:58 fsbl
drwxrwxr-x  3 agin users 4096 Aug 13 11:44 gen
drwxr-xr-x 11 agin users 4096 Aug 13 09:21 include
-rwxr--r--  1 agin users 2724 Aug  8 21:29 license.txt
drwxrwxr-x  3 agin users 4096 Aug 13 11:44 objs
drwxr-xr-x  4 agin users 4096 Aug 13 11:52 scripts
drwxr-xr-x 12 agin users 4096 Aug 11 16:58 security
drwxr-xr-x  7 agin users 4096 Aug 11 16:58 shmoo
drwxr-xr-x  3 agin users 4096 Aug 11 16:58 splash
drwxr-xr-x  9 agin users 4096 Aug 11 16:58 ssbl
drwxr-xr-x  3 agin users 4096 Aug 11 16:58 thirdparty
-rwxr--r--  1 agin users    6 Aug  8 21:29 version
drwxrwxr-x  7 agin users 4096 Aug 13 11:44 zlib-1.1.3
```

The path information to the relevant BOLT source files is used to set the BOLT path information in the makefile properly:

Change the Makefile located in \BSEAV\app\bsu\build to reflect the correct path:

```
include ../../../../nexus/build/os/$(B_REFSW_OS)/os_tools.inc

BOLT_DIR_NAME =
BRCM_BOLT_Generic_0_0_88_E1/release_bolt_v0.88/src/BRCM_Src_BOLT_Generic_0_0_88_E1/bolt_v0.
88
CFE_DIR_NAME = CFE
BSEAV = $(shell cd "../../.." && ${PWD})
```

Building the compressed binary and ELF application

From the Unified Reference Software Release, the working directory is \BSEAV\apps\bsu\build.

Setting the Environmental Variables

Use the same environmental variables you would use to build the URSR reference software, for your specific platform. For example, for the BCM97362SV reference board, you would set this:

```
stb-irva-09{agin}175: bash
bash-3.2$ cd /fe_lab/agin/src
bash-3.2$ cd URSR_Latest/BSEAV/app/bsu/build
bash-3.2$ export NEXUS_PLATFORM=97362
bash-3.2$ export PLATFORM=97362
bash-3.2$ export BCHP_VER=A0
```

Customizable Features List

The Customizable Features List (CFL) is located in the file, \BSEAV\apps\bsu\build\Makefile.cfl. The current settings are shown below:

```
#
# Customizeable Features List
#
export NEXUS_AUDIO_SUPPORT=n
export NEXUS_ASTM_SUPPORT=n
export NEXUS_FILE_SUPPORT=n
export NEXUS_FILE_MUX_SUPPORT=n
export NEXUS_HDMI_INPUT_SUPPORT=n
export NEXUS_INPUT_CAPTURE_SUPPORT=n
export NEXUS_INPUT_ROUTER_SUPPORT=n
export NEXUS_IR_BLASTER_SUPPORT=n
export NEXUS_PLAYBACK_SUPPORT=n
export NEXUS_POWER_MANAGEMENT=n
export NEXUS_PWM_SUPPORT=n
export NEXUS_RECORD_SUPPORT=n
export NEXUS_SERVER_SUPPORT=n
export NEXUS_SYNC_CHANNEL_SUPPORT=n
export NEXUS_SMARTCARD_SUPPORT=n
export NEXUS_STREAM_MUX_SUPPORT=n
export NEXUS_TOUCHPAD_SUPPORT=n
export NEXUS_TEMP_MONITOR_SUPPORT=n
export NEXUS_SIMPLE_DECODER_SUPPORT=n
export NEXUS_VIDEO_ENCODER_SUPPORT=n
export PLAYBACK_IP_SUPPORT=n
export V3D_SUPPORT=n
export MEDIA_ASF_SUPPORT=n
export MEDIA_AVI_SUPPORT=n

#
# If you want the GUI demo to have a background image, the following variables may be set
to 'y'.
#
ifeq ($(filter $(NEXUS_PLATFORM),97584), $(NEXUS_PLATFORM))
    export NEXUS_PICTURE_DECODER_SUPPORT=y
    export NEXUS_SURFACE_COMPOSITOR_SUPPORT=y
    export NEXUS_VIDEO_DECODER_SUPPORT=y
else
    export NEXUS_PICTURE_DECODER_SUPPORT=n
    export NEXUS_SURFACE_COMPOSITOR_SUPPORT=n
    export NEXUS_VIDEO_DECODER_SUPPORT=n
endif
```

As you can see, by default, almost all features which we could disable, are turned off.

Building the GUI App

to build the BSU GUI application:

```
bash-3.2$ make
```

Building the Menu Driven Test App

to build the BSU with the test menu:

```
bash-3.2$ make BUILD=test
```

If the build goes well, you should see the following at the beginning of the build log:

```
[Build ... Nexus]
Verifying nexus interfaces
[Thunk..... platform]
[Compile... nexus_platform.c (platform)]
[Compile... nexus_platform_config.c (platform)]
[Compile... nexus_platform_core.c (platform)]
[Compile... nexus_platform_97362.c (platform)]
[Compile... nexus_platform_interrupt.c (platform)]
[Compile... nexus_platform_server.c (platform)]
[Compile... nexus_map.c (platform)]
```

And this at the end of the build log:

```
[Compile ... barena.c]
[Compile ... bioatom.c]
[Compile ... bpool.c]
[Linking... bsu]
sde-ld: total time in link: 1.755732
sde-ld: data size 65955740
[Install... binaries]
[Install... BSU Complete]
bash-3.2$
```

And, if the build goes well, the relevant output files generated by the build will be located in the following sub-directory:

\obj.{PLATFORM}.BSEAV\bin

For example, for the BCM97362SV reference board, the relevant output files generated by the build will be located in the following sub-directory:

\obj.97362.BSEAV\bin

The important files located in the above object directories are described in the following table:

File	Description
bsu.gz	The gzip'd elf file
bsu.bin.gz	The gzip'd binary file

Table 4: Executable Files

In addition to the above files, a map file generated by the linker is located in the following directory:

\BSEAV\app\bsu\build\{PLATFORM}.sde.debug

This file is described in the following table:

File	Description
bsu.map	The map file generated by the linker

Table 5: Other Important Files

J.Running BSU

There are three ways to boot BSU: via TFTP, USB, or flash.

There are slight differences in the instructions, when using a MIPS platform (CFE) and an ARM platform (BOLT). In general, the following MIPS platform instructions will work on an ARM platform also. Where there are differences to run on an ARM platform, in the instructions below, it will be noted in *red italics*.

a. Booting BSU on a MIPS based platform (CFE)

1. Booting via TFTP.

First, locate the file (bsu.gz or bsu.bin.gz) onto a TFTP server.
Then, from the CFE prompt, issue the following ifconfig command (in red) in order to establish an IP address:

```
CFE> ifconfig eth0 -auto
Ethernet link is up: 100 Mbps Full-Duplex
Device eth0:  hwaddr 00-10-18-18-23-ED, ipaddr 10.13.134.103, mask 255.255.254.0
       gateway 10.13.134.1, nameserver 10.10.10.10, domain broadcom.com
*** command status = 0
```

Next, enter the following "boot" command (as shown below in red), from the CFE prompt. You should then see something similar to the following output (note that what is displayed at the end of the following log is the test menu app, built as a result of using "make BUILD=test". Note that the server location shown in the example below is my own:

If booting for an ARM platform, the –addr parameter should be 0x10000000 and the –max parameter should be 0x10000000.

```
CFE> boot -bsu -noclose -z -raw -addr=0x88010000 -max=0x8000000 stb-irva-
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz
Loader:raw Filesys:tftp Dev:eth0 File:stb-irva-
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz Options:(null)
Loading: .. 3376512 bytes read
Entry address is 0x88010000
Starting program at 0x88010000

_main(r0:0x8705f500 r1:0x00000000 r2:0x00000000 r3:0x8704ff4c)
BSU @ 0x8704ff4c
Signature is ok
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:182
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:209
MEMC0=1024MB, MEMC1=0MB
setting up tlb for an additional 768MB uncached memory at virtual address 0x10000000 and
an additional 768MB cached memory at virtual address 0x50000000on the 1024MB MEMC0
calculated MIPS clock speed = 0742500140 Hz
in root_task
*** 00:00:00.000 nexus_platform: Release 97584 14.3
*** 00:00:00.100 BXVD: BXVD_Open() - Hardware revision: M, Firmware revision: 16050d
*** 00:00:00.240 BCEC: CEC Logical Address Uninitialized
*** 00:00:00.250 nexus_frontend_7584: BHAB_InitAp(7584 image)
*** 00:00:00.640 bads_Leap_priv: LEAP FW version is 6.3.108.4
*** 00:00:00.650 nexus_statistics_api:
NEXUS_FrontendDevice_Open3128[frontend:IdleActiveStandby] 400 msec
*** 00:00:01.610 nexus_statistics_api:
NEXUS_FrontendDevice_Set3128Settings[frontend:IdleActiveStandby] 950 msec
*** 00:00:01.620 nexus_platform_frontend: Waiting for frontend(7584_SFF) 0 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 1 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 2 to initialize
*** 00:00:01.640 nexus_platform_frontend: Waiting for frontend(7584_SFF) 3 to initialize
*** 00:00:01.650 nexus_platform_frontend: Waiting for Oob channel to initialize
*** 00:00:01.660 nexus_platform: initialized
*** 00:00:01.660 nexus_statistics_module: platform[Default]
nexus/base/src/nexus_base_scheduler.c:824 410 msec
*** 00:00:01.670 nexus_statistics_callback: Timer callback 871e3d50:8801dcb8 from
platforms/common/src/nexus_platform_config.c:939 420 msec

BCM97584 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Sep  2
2014, 08:29:48
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: *** 00:00:02.010 BCHP_AVS: AVS support enabled!

BSU>
```

Note: for the '–max' parameter above, choose a value large enough for the CFE loader to cover areas which it could initialize potential global variable locations to zero. This information could be obtained from the map file generated by the linker (bsu.map). Choose a value at least larger than the address of '__end' minus the starting address of 0x88010000.

Alternatively, you can also boot the compressed elf file image (bsu.gz) by issuing the following command. Note that the server location shown in the example below is my own:

```
CFE> boot -bsu -noclose -z stb-irva-
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.gz
```

2. Booting from a USB mass storage device.

First, we need to know what the CFE named the USB mass storage device. To determine this, enter the following "show" command from the CFE prompt (as shown below in red):

```
CFE> show devices
Device Name          Description
-------------------  ------------------------------------------------------------
              uart0  16550 DUART at 0xB0406800 channel 0
         flash0.cfe  New NAND flash at 00000000 offset 00000000 size 2048KB spare 1024KB
      flash0.kernel  New NAND flash at 00000000 offset 00200000 size 5120KB spare 1024KB
      flash0.macadr  New NAND flash at 00000000 offset 00700000 size 1024KB spare 1024KB
       flash0.nvram  New NAND flash at 00000000 offset 00800000 size 1024KB spare 1024KB
     flash0.virtual  New NAND flash at 00000000 offset 00900000 size 2048KB spare 1024KB
     flash0.kreserv  New NAND flash at 00000000 offset 00B00000 size 2048KB spare 1024KB
      flash0.splash  New NAND flash at 00000000 offset 00D00000 size 1024KB spare 1024KB
      flash0.avail0  New NAND flash at 00000000 offset 00E00000 size 509952KB spare
2048KB
               eth0  GENET Internal Ethernet at 0xB0430800
           usbdisk0  USB Disk unit 0
*** command status = 0
CFE>
```

Based on the above information, the usb device is named 'usbdisk0'. Knowing that we placed the compressed binary file (bsu.bin.gz) on this USB memory stick, we can proceed to boot this file by issuing the following command:

If booting for an ARM platform, the –addr parameter should be 0x10000000 and the –max parameter should be 0x10000000.

```
CFE> boot -bsu -noclose -z -raw -addr=0x80010000 -max=0x800000 usbdisk0:bsu.bin.gz
```

You should see the same log as shown in the above TFTP method. Again, be aware of the '-max' parameter and its purpose.

Alternatively, you can also boot the compressed elf file image (bsu.gz, assuming that it is also placed on this USB memory stick) by issuing the following "boot" command (as shown below in red):

```
CFE> boot -bsu -noclose -z -elf usbdisk0:bsu.gz
```

3. Booting from a flash device.

Similar to the procedure for booting from a USB mass storage device, we need to know what the device name that CFE has associated the flash with. Issue the command shown below in red:

```
CFE> show devices
Device Name          Description
-------------------  ------------------------------------------------------------
              uart0  16550 DUART at 0xB0406800 channel 0
         flash0.cfe  New NAND flash at 00000000 offset 00000000 size 2048KB spare 1024KB
      flash0.kernel  New NAND flash at 00000000 offset 00200000 size 5120KB spare 1024KB
      flash0.macadr  New NAND flash at 00000000 offset 00700000 size 1024KB spare 1024KB
       flash0.nvram  New NAND flash at 00000000 offset 00800000 size 1024KB spare 1024KB
     flash0.virtual  New NAND flash at 00000000 offset 00900000 size 2048KB spare 1024KB
     flash0.kreserv  New NAND flash at 00000000 offset 00B00000 size 2048KB spare 1024KB
      flash0.avail0  New NAND flash at 00000000 offset 00D00000 size 248832KB spare
2048KB
               eth0  GENET Internal Ethernet at 0xB0430800
*** command status = 0
CFE>
```

You can see above that CFE has named the miscellaneous flash location, 'flash0.avail0'. This location has more than enough size to contain the compressed binary image, bsu.bin.gz, which should be approximately 1MB.

The next step is to flash the bsu.bin.gz in this memory location. We will do this with the following sequence of commands, starting with the "ifconfig" command (as shown below in red):

```
CFE> ifconfig eth0 -auto
100 Mbps Full-Duplex
Device eth0:  hwaddr 00-10-18-18-23-ED, ipaddr 10.13.134.103, mask 255.255.254.0
       gateway 10.13.134.1, nameserver 10.10.10.10, domain broadcom.com
*** command status = 0
```

Now, issue the "flash" command (shown below in red) to flash the binary image across the network via TFTP to flash. Note that the server location shown in the example below is my own:

```
CFE> flash -noheader stb-irva-09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.bin.gz
flash0.avail0
Reading stb-irva-09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.bin.gz: Done. 928470
bytes read

Programming...


>>>[ui_flash] Writing data (flash0.avail0) from source (stb-irva-
09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.bin.gz)
done. 928470 bytes written
*** command status = 0
```

Finally, we are able to issue the "boot" command to boot from flash (as shown below in red):

If booting for an ARM platform, the –addr parameter should be 0x10000000 and the –max parameter should be 0x10000000.

```
CFE> boot -bsu -noclose -z -raw -addr=0x80010000 -max=0x800000 flash0.avail0:
```

which should yield the same log as shown in the above TFTP method. Again, be aware of the '-max' parameter and its purpose.

Alternatively, we could have decided to flash the compressed elf image (bsu.gz) instead (as shown below in red). Note that the server location shown in the example below is my own:

```
CFE> flash -noheader stb-irva-09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.gz
flash0.avail0
Reading stb-irva-09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.gz: Done. 2714606
bytes read
Programming...

>>>[ui_flash] Writing data (flash0.avail0) from source (stb-irva
09:/agin/src/URSR_Latest/obj.97362/BSEAV/bin/bsu.gz)
done. 2714606 bytes written
*** command status = 0
CFE>
```

we are able to boot from flash:

```
CFE> boot -z -elf flash0.avail0:
```

which should yield the same log as shown in the above boot binary from flash method.

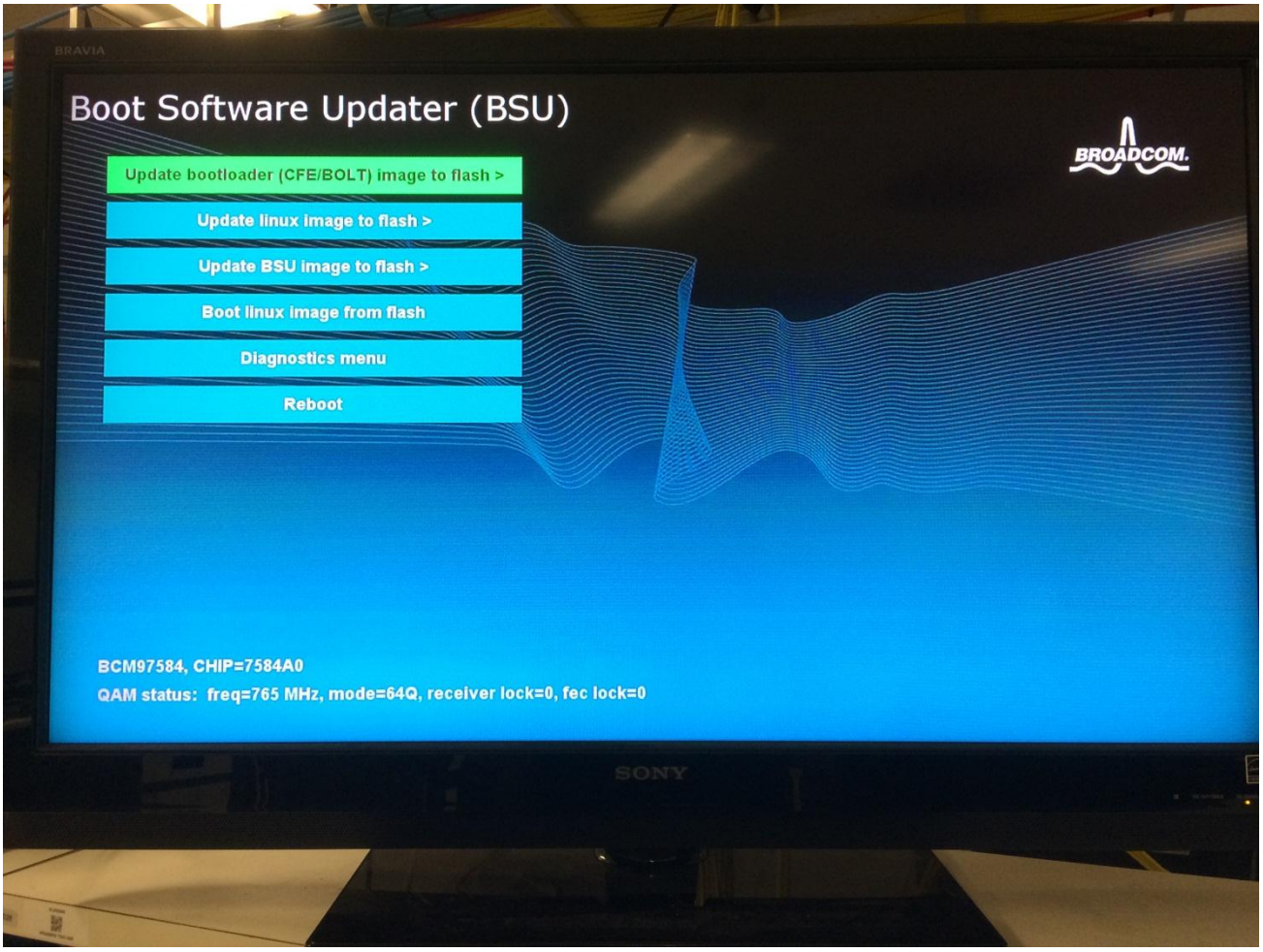
b. GUI App

If the build option, “make” was chosen to build the BSU, then, after boot-up, you will see, from the serial console, something similar to the following:

```
CFE> Ethernet link is up: 100 Mbps Full-Duplex
boot -bsu -noclose -z -raw -addr=0x88010000 -max=0x8000000 stb-irva
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz
Loader:raw Filesys:tftp Dev:eth0 File:stb-irva
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz Options:(null)
Loading: .. 3356480 bytes read
Entry address is 0x88010000
Starting program at 0x88010000

_main(r0:0x8705f500 r1:0x00000000 r2:0x00000000 r3:0x8704ff4c)
BSU @ 0x8704ff4c
Signature is ok
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:182
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:209
MEMC0=1024MB, MEMC1=0MB
setting up tlb for an additional 768MB uncached memory at virtual address 0x10000000 and
an additional 768MB cached memory at virtual address 0x50000000on the 1024MB MEMC0
calculated MIPS clock speed = 0742500192 Hz
in root_task
*** 00:00:00.000 nexus_platform: Release 97584 14.3
*** 00:00:00.100 BXVD: BXVD_Open() - Hardware revision: M, Firmware revision: 16050d
*** 00:00:00.240 BCEC: CEC Logical Address Uninitialized
*** 00:00:00.250 nexus_frontend_7584: BHAB_InitAp(7584 image)
*** 00:00:00.640 bads_Leap_priv: LEAP FW version is 6.3.108.4
*** 00:00:00.650 nexus_statistics_api:
NEXUS_FrontendDevice_Open3128[frontend:IdleActiveStandby] 400 msec
*** 00:00:01.610 nexus_statistics_api:
NEXUS_FrontendDevice_Set3128Settings[frontend:IdleActiveStandby] 950 msec
*** 00:00:01.620 nexus_platform_frontend: Waiting for frontend(7584_SFF) 0 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 1 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 2 to initialize
*** 00:00:01.640 nexus_platform_frontend: Waiting for frontend(7584_SFF) 3 to initialize
*** 00:00:01.650 nexus_platform_frontend: Waiting for Oob channel to initialize
*** 00:00:01.660 nexus_platform: initialized
*** 00:00:01.660 nexus_statistics_module: platform[Default]
nexus/base/src/nexus_base_scheduler.c:824 410 msec
*** 00:00:01.670 nexus_statistics_callback: Timer callback 871e3d50:8801f2f4 from
platforms/common/src/nexus_platform_config.c:939 420 msec
*** 00:00:02.010 BCHP_AVS: AVS support enabled!
usb initialized with fat file system
    00:00:02.590 BHDM: Tx0: Output Fmt:   720p     (YCbCr 4:4:4 24 bpp) 1280 [+]  x
720/720 [+]
    00:00:02.600 BHDM: Tx0: Pixel Clock: 74 MHz (TMDS Character Rate 74.25 / 1.001)
    00:00:02.620 BHDM: Tx0: Output Mode:  HDMI (Audio+Video)
*** 00:00:02.620 nexus_statistics_api: NEXUS_Display_AddOutput[display:Default] 160 msec
'Update bootloader (CFE/BOLT) image to flash >' will launch 'xxx'
*** 00:00:02.940 desktop: tuning 765 MHz...
'Update bootloader (CFE/BOLT) image to flash >' will launch 'xxx'
'Update bootloader (CFE/BOLT) image to flash >' will launch 'xxx'
```

The resulting GUI will look like this:



c. Menu Driven Test App

If the build option, “make BUILD=test” was chosen to build the BSU, then, after boot-up, you will see, from the serial console, a menu selection screen. An example of this is shown below:

```
CFE> boot -bsu -noclose -z -raw -addr=0x88010000 -max=0x8000000 stb-irva
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz
Loader:raw Filesys:tftp Dev:eth0 File:stb-irva
09:/agin/src/git/phase14.3/myrepo/obj.97584/BSEAV/bin/bsu.bin.gz Options:(null)
Loading: .. 3376512 bytes read
Entry address is 0x88010000
Starting program at 0x88010000

_main(r0:0x8705f500 r1:0x00000000 r2:0x00000000 r3:0x8704ff4c)
BSU @ 0x8704ff4c
Signature is ok
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:182
!!!Error BERR_NOT_INITIALIZED(1) at magnum/basemodules/dbg/ucos_iii/bdbg_os_priv.c:209
MEMC0=1024MB, MEMC1=0MB
setting up tlb for an additional 768MB uncached memory at virtual address 0x10000000 and
an additional 768MB cached memory at virtual address 0x50000000on the 1024MB MEMC0
calculated MIPS clock speed = 0742500140 Hz
in root_task
*** 00:00:00.000 nexus_platform: Release 97584 14.3
*** 00:00:00.100 BXVD: BXVD_Open() - Hardware revision: M, Firmware revision: 16050d
*** 00:00:00.240 BCEC: CEC Logical Address Uninitialized
*** 00:00:00.250 nexus_frontend_7584: BHAB_InitAp(7584 image)
*** 00:00:00.640 bads_Leap_priv: LEAP FW version is 6.3.108.4
*** 00:00:00.650 nexus_statistics_api:
NEXUS_FrontendDevice_Open3128[frontend:IdleActiveStandby] 400 msec
*** 00:00:01.610 nexus_statistics_api:
NEXUS_FrontendDevice_Set3128Settings[frontend:IdleActiveStandby] 950 msec
*** 00:00:01.620 nexus_platform_frontend: Waiting for frontend(7584_SFF) 0 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 1 to initialize
*** 00:00:01.630 nexus_platform_frontend: Waiting for frontend(7584_SFF) 2 to initialize
*** 00:00:01.640 nexus_platform_frontend: Waiting for frontend(7584_SFF) 3 to initialize
*** 00:00:01.650 nexus_platform_frontend: Waiting for Oob channel to initialize
*** 00:00:01.660 nexus_platform: initialized
*** 00:00:01.660 nexus_statistics_module: platform[Default]
nexus/base/src/nexus_base_scheduler.c:824 410 msec
*** 00:00:01.670 nexus_statistics_callback: Timer callback 871e3d50:8801dcb8 from
platforms/common/src/nexus_platform_config.c:939 420 msec


BCM97584 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Sep  2
2014, 08:29:48
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell
```

Frontend Test

Choosing option 1) will yield the “Frontend Test”. What happens next, depends on the board that you are running on, and any special environmental variables you selected to build with. Some platforms don’t require any additional environmental variables. For example, if you built for the BCM97362SV board, you would not need to specify any additional environmental variables. By default, building for the BCM97362SV board will cause the “Front end Test” to run the BSU satellite test. This code is located in the file, bsu_satellite_test.c. The satellite test basically uses Nexus function calls to tune and acquire the satellite frontend and then check for status. If the board is hooked up to a satellite modulator, the test will look like this:

Satellite Frontend Test

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 1
Set DiseqcSettings
Sat Status
  symbolRate 20000000
  locked  0
Frontend(0x82f1c1cc) - lock callback
  demodLocked = 1
  diseqc tone = 0, voltage = 111
*** 00:00:02.020 BCHP_AVS: AVS support enabled!
Sat Status
  symbolRate 20000000
  locked  1
```

QAM Test

Some boards are configured with a QAM frontend (for example, the BCM97435SV board).

The QAM test is located in the file, bsu_qam_test.c. The QAM test basically uses Nexus function calls to tune and lock the cable frontend and then check for status. If this board is hooked up to a cable headend, the test will look like this:

```
BCM97435 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:58:04
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 1*** 00:00:02.020 BCHP_AVS: AVS support enabled!

*** 00:00:50.960 tune_qam: tuning 765 MHz...
 served.
0*** 00:00:50.960 nexus_frontend_31xx: BHAB_InitAp(rev a image)
*** 00:00:50.970 nexus_frontend_31xx: BHAB_InitAp(rev a image)
Task terminated!!
Task terminated!!
Task terminated!!
*** 00:00:57.980 nexus_statistics_api:
NEXUS_Frontend_TuneQam[frontend:IdleActiveStandby] 7020 msec
*** 00:00:58.120 bhab_31xx_priv: AP initialization timeout. Starting over 0 times
*** 00:00:58.170 tune_qam: frontend locked
```

OFDM Test

Some boards are configured with a OFDM frontend (for example, the BCM97563SV board).

The OFDM test is located in the file, bsu_ofdm_test.c. The OFDM test basically uses Nexus function calls to tune and lock the OFDM frontend and then check for status. If this board is hooked up to a OFDM headend, the test will look like this:

```
BCM97563 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 15:18:20
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 1
Using built in default tune parameters for DVB-T
*** 00:00:01.060 bhab_7563_priv: DVB-T Unlocked
OFDM Lock callback: Fast lock status = Unlocked.
waiting for lock status...
*** 00:00:01.780 bhab_7563_priv: DVB-T Locked
OFDM Lock callback: Fast lock status = Locked.
```

In all of the above frontend cases, a successful frontend lock is depicted. Now, regardless of which frontend test is performed, once lock is obtained, you will be able to run the message test. The message test is shown below:

Message Test

```
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
Found PAT: id=0 size=20
  program 1: pid 0x3f
  program 2: pid 0x3e
```

Colorbar Test

Selecting option 3) from the main menu will result in the colorbar test being run. Assuming that an HDMI, component, and/or composite cable is attached between the set-top board and the screen, you will see the following on the serial console:

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 2*** 00:00:02.020 BCHP_AVS: AVS support enabled!

    00:00:02.530 BHDM_PRIV: HDMI Color Depth Setting: 24 bpp (0)
*** 00:00:02.540 BHDM: Tx0: Output Fmt:   720p     1280 [+]  x 720/720 [+]
*** 00:00:02.560 BHDM: Tx0: Output Mode:  HDMI (Audio+Video)
*** 00:00:02.570 nexus_statistics_api: NEXUS_Display_AddOutput[display:Default] 160 msec
Press any key to exit
```

in addition to having the color bars displayed on the screen:

Pressing any key after will result in the color bar test being turned off, and a return to the main menu:

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 2*** 00:00:02.020 BCHP_AVS: AVS support enabled!

    00:00:02.530 BHDM_PRIV: HDMI Color Depth Setting: 24 bpp (0)
*** 00:00:02.540 BHDM: Tx0: Output Fmt:   720p     1280 [+]  x 720/720 [+]
*** 00:00:02.560 BHDM: Tx0: Output Mode:  HDMI (Audio+Video)
*** 00:00:02.570 nexus_statistics_api: NEXUS_Display_AddOutput[display:Default] 160 msec
Press any key to exit
*** 00:01:05.430 nexus_statistics_api: NEXUS_Display_Close[display:Default] 130 msec


BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

USB Test

From the main menu, selection option 4) will result in the following being displayed:

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 4


================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice:
```

If you attach a USB memory stick to the USB connector on the board, you will see the following:

```
USB: New device connected to bus 0 hub 1 port 1
USB: Resetting device on bus 0 hub 1 port 1
USB: Locating Class 08 Vendor 0930 Product 6545: Mass-Storage Device
USBMASS: Unit 0 connected
```

This means that the BSU software (or, specifically, the BOLT/CFE) has detected the USB memory stick.

Selecting option 2) from this menu will treat the data on the USB memory stick as if it was formatted for a FAT32 file system:

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 2
usb initialized with fat file system
```

After initializing with the FAT filesystem, and assuming that you have a file on the USB memory stick, formatted with FAT32, you should be able to read the file. Option 3) will read a file. The file name is hardcoded in the source file, bsu_usb_test.c, with the variable, 'fname'. If everything is working, you should see a series of dots displayed on the serial console as the file is read:

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 3
.................................
```

When the file is fully read, the USB Menu will be shown again.

Option 1) will initialize the file system to 'raw'. What this means, is that the data on the memory stick will be treated as raw bytes:

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 1
usb initialized with raw file system
```

Selecting option 4) after initializing the file system to 'raw' will allow one to begin reading raw bytes. The software will run indefinitely, or at least until there is no more data to be read from the USB memory stick. Or, the reading will stop if one presses the ENTER key on the keyboard.

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 4
................................
```

Option 5) will un-initialize the file system, if it was previously initialized. Also, option 0) will un-initialize the file system, if it was previously initialized, before exiting the USB menu. If the file system was initialized, you will the message as shown below:

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 5
usb file system un-initialized
```

Selecting option 0) will exit back to the main menu.

```
================
  USB MENU
================
    0) Exit
    1) Init USB with raw file system
    2) Init USB with FAT file system
    3) Read a FAT file system file from USB
    4) Read raw from USB
    5) Un-init USB file system

Choice: 0

BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

Flash Test

Selecting option 5) from the main menu will transfer you to the Flash Menu:

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 5
nvram_devname not NULL


================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice:
```

Ignore the “nvram_devname not NULL” message. That is for information only.

You will notice that option 5) displays the current flash partition. The default flash partition is “flash0.nvram”. You can pick any flash partition as recognized by BOLT/CFE. You can display the list of partitions by issuing a “show devices” command from the BOLT/CFE prompt, or from within the BOLT/CFE command shell of BSU.

On my board, this is what is currently shown:

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 0


BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: g
BSU-CFE> show devices

Device Name          Description
------------------   ------------------------------------------------------------

              uart0  16550 DUART at 0xB0406800 channel 0
         flash0.cfe  New NAND flash at 00000000 offset 00000000 size 2048KB spare 1024KB
      flash0.kernel  New NAND flash at 00000000 offset 00200000 size 5120KB spare 1024KB
      flash0.macadr  New NAND flash at 00000000 offset 00700000 size 1024KB spare 1024KB
       flash0.nvram  New NAND flash at 00000000 offset 00800000 size 1024KB spare 1024KB
     flash0.virtual  New NAND flash at 00000000 offset 00900000 size 2048KB spare 1024KB
     flash0.kreserv  New NAND flash at 00000000 offset 00B00000 size 2048KB spare 1024KB
      flash0.avail0  New NAND flash at 00000000 offset 00D00000 size 248832KB spare
2048KB
               eth0  GENET Internal Ethernet at 0xB0430800
           usbdisk0  USB Disk unit 0
*** command status = 0
BSU-CFE> quit


BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

Continuing on, in the Flash test menu, selecting option 1) will result in reading 512 bytes from the current partition and read starting offset into a 512 byte buffer:

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 1

00000000:  01 0d 00 45 54 48 30 5f 50 48 59 3d 49 4e 54 01
00000010:  11 00 45 54 48 30 5f 4d 44 49 4f 5f 4d 4f 44 45
00000020:  3d 31 01 0f 00 45 54 48 30 5f 53 50 45 45 44 3d
00000030:  31 30 30 01 0f 00 45 54 48 30 5f 50 48 59 41 44
00000040:  44 52 3d 31 00 ff ff ff ff ff ff ff ff ff ff ff
00000050:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000060:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000070:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000080:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000090:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000a0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000b0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000c0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000d0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000e0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000000f0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000100:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000110:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000120:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000130:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000140:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000150:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000160:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000170:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000180:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
00000190:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001a0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001b0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001c0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001d0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001e0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
000001f0:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

After the read is completed, the read starting offset is incremented by the number of bytes read (512 in this case):

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x200)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice:
```

Selecting option 1) again will result in the next 512 bytes being read.

Selecting option 2) will write 512 bytes from the same buffer into the current partition and starting write offset:

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x200)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 2
writing 512 bytes at offset 0x0
```

Of course, you can change either the read or write starting offsets by selecting options 3) or 4), respectively:

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x200)
    2) Write to flash partition (starting offset=0x200)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 3

enter read starting offset in hex:  0


================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x200)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 4

enter write starting offset in hex:  0


================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice:
```

Finally, selecting option 5) will allow us to change the partition we are accessing:

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.nvram)
    6) Erase flash

Choice: 5
enter parition name:  flash0.cfe
new partition name=flash0.cfe
nvram_devname not NULL


================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.cfe)
    6) Erase flash

Choice:
```

Selecting option 0) will exit back to the main menu.

```
================
  FLASH MENU
================
    0) Exit
    1) Read from flash partition (starting offset=0x0)
    2) Write to flash partition (starting offset=0x0)
    3) Change read starting offset
    4) Change write starting offset
    5) Change partition (current partition=flash0.cfe)
    6) Erase flash

Choice: 0

BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

Ethernet Test

Selecting option 6) from the main menu will take you to the Ethernet menu:

```
BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 6


=================
  Ethernet Menu
=================
    0) Exit
    1) ifconfig eth0 -auto
    2) TFTP test

Choice:
```

Selecting option 1) will perform an “ifconfig eth0 –auto”, similar to performing it from the BOLT/CFE prompt:

```
=================
  Ethernet Menu
=================
    0) Exit
    1) ifconfig eth0 -auto
    2) TFTP test

Choice: 1
Ethernet link is up: 100 Mbps Full-Duplex
Device eth0:  hwaddr 00-10-18-03-E3-03, ipaddr 10.13.135.227, mask 255.255.254.0
        gateway 10.13.134.1, nameserver 10.10.10.10, domain broadcom.com
```

Selecting option 2) will perform a TFTP test:

```
=================
  Ethernet Menu
=================
    0) Exit
    1) ifconfig eth0 -auto
    2) TFTP test

Choice: 1
746869732069732061207361 6d706c65 : this is a sample
207465737420666 96c6520666f722074 :  test file for t
686520657468657 26e65742074667470 : he ethernet tftp
207465737420666f72204253552e      :  test for BSU.

Total bytes read: 62
```

The TFTP test, located in the file, \BSEAV\app\bsu\src\test\bsu_ethernet_test.c, relies on a hardcoded variable, ‘fname’, which is currently set to:

```
char *fname="stb-irva-09:/agin/src/a.txt";
```

which is currently set to a location on the network where I have a simple ASCII text file, ‘a.txt’. This text file contains the following:

```
this is a sample test file for the ethernet tftp test for BSU.
```

The TFTP test is basically reading this file across the network and displaying it’s contents onto the serial console.

Selecting option 0) will exit back to the main menu.

```
=================
  Ethernet Menu
=================
    0) Exit
    1) ifconfig eth0 -auto
    2) TFTP test

Choice: 0

BCM97362 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 29
2013, 19:27:33
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

UART Test

Selecting option 7) from the main menu will take you to the UART test:

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
Opening UART2
This application assumes that you have set the pinmuxing to enable UART2 on your
platform.
This is typically not enabled by default.
Printing 'hello' to terminal.
Starting loopback. Please type into the UART2 console. It should be echoed to that
console and also printed on the OS console.  Press 'q' to quit.
```

The above example was performed on my BCM97346DBSMB board. A second serial console was attached to J2305, labeled “INNER LOOP”. This connector served as UART2 for this example. When selecting option 7), the following was displayed on the UART2 serial console:

```
hello
```

From the UART2 serial console, I typed the following:

```
hello
hi uart0!
```

At which point, the UART0 serial console displayed:

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
Opening UART2
This application assumes that you have set the pinmuxing to enable UART2 on your
platform.
This is typically not enabled by default.
Printing 'hello' to terminal.
Starting loopback. Please type into the UART2 console. It should be echoed to that
console and also printed on the OS console.  Press 'q' to quit.
hi uart 0!
```

Pressing ‘q’ returns you to the main menu.

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
Opening UART2
This application assumes that you have set the pinmuxing to enable UART2 on your
platform.
This is typically not enabled by default.
Printing 'hello' to terminal.
Starting loopback. Please type into the UART2 console. It should be echoed to that
console and also printed on the OS console.  Press 'q' to quit.
hi uart 0!

BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

IR Input Test

Selecting option 8) from the main menu will take you to the IR Input test:

```
BCM97584 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Sep  2
2014, 08:49:08
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 8

Press buttons on the remote.  Pressing <ENTER> on the keyboard exits.
```

At this point, press any button on your IR remote control to display the codes which the chip detects (press ENTER to exit):

```
BCM97584 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Sep  2
2014, 08:49:08
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: 8

Press buttons on the remote.  Pressing <ENTER> on the keyboard exits.

irCallback: rc: 0, code: 0000f001, repeat: false
irCallback: rc: 0, code: 0000f001, repeat: true
irCallback: rc: 0, code: 0000f001, repeat: true
irCallback: rc: 0, code: 0000f001, repeat: false
irCallback: rc: 0, code: 0000f001, repeat: true
irCallback: rc: 0, code: 0000e002, repeat: false
irCallback: rc: 0, code: 0000e002, repeat: true
```

BOLT/CFE Command Shell

Selecting option g) from the main menu will take you to the BOLT/CFE command shell:

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: g
BSU-CFE>
```

From here, you can enter almost any BOLT/CFE command that you could enter from the real BOLT/CFE prompt.

For example:

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: g
BSU-CFE> ifconfig eth0 -auto
Ethernet link is up: 100 Mbps Full-Duplex
Device eth0:  hwaddr 00-10-18-04-23-D2, ipaddr 10.13.135.101, mask 255.255.254.0
        gateway 10.13.134.1, nameserver 10.10.10.10, domain broadcom.com
*** command status = 0
BSU-CFE> ping 10.13.134.1
10.13.134.1 (10.13.134.1) is alive
10.13.134.1 (10.13.134.1): 1 packets sent, 1 received
*** command status = 0
BSU-CFE>
```

Also, entering 'reboot' at the BSU-CFE> prompt will cause the software to reboot back to the main BOLT/CFE prompt.

The main purpose of the BOLT/CFE command shell is to prove that BOLT/CFE resident in memory is still functional.

Typing quit and pressing ENTER will return you to the main menu:

```
BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice: g
BSU-CFE> quit


BCM97346 Set Top Boot Software Updater (BSU) Code - Version 3.00, Compiled on Oct 30
2013, 11:15:05
Copyright (C) Broadcom Corporation 2003. All rights reserved.
    1) Front end test
    2) Display color bar
    3) Message test
    4) USB test
    5) Flash test
    6) Ethernet test
    7) UART test
    8) IR input test
    g) CFE command shell

Choice:
```

K. Security Considerations

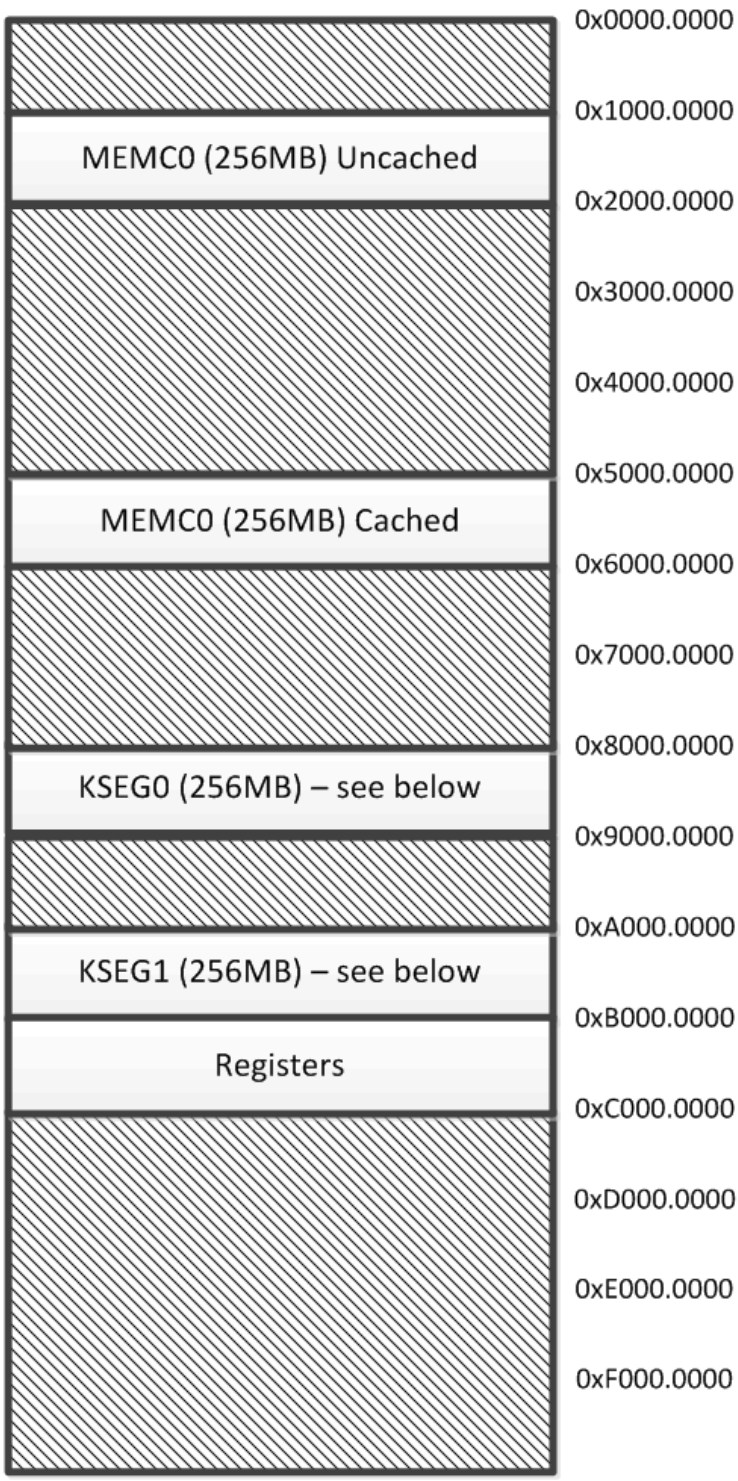
The Broadcom boot loader (CFE/BOLT) has two parts: The first stage boot loader (FSBL) and the second stage boot loader (SSBL). Upon CPU reset, the FSBL binary image is copied into an internal CPU RAM (instruction cache on a MIPS or 64KB SRAM resident on the ARM core) and beings execution from that RAM. The FSBL is responsible for initializing the DDR, copying the SSBL encrypted image from FLASH to DDR, decrypting it, authenticating it, and running it. The SSBL is responsible for copying an encrypted application (linux OS or BSU) from FLASH to DDR, decrypting it, authenticating it, and running it.

Decrypting and authenticating an encrypted image is performed specifically by the Broadcom Security Processor, given the encrypted image's key, size, and starting address.

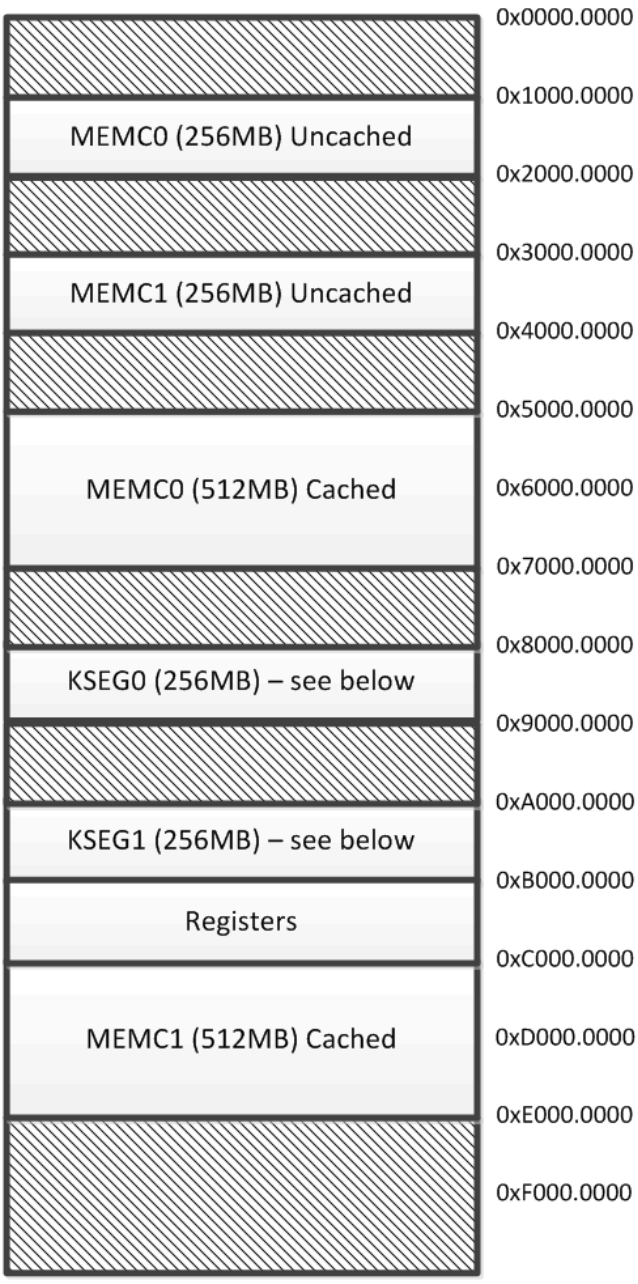
In the case of BSU loading a new binary (via OTA/Cable/Satellite/USB/Ethernet/UART) into memory, that binary may be authenticated via the same SSBL method mentioned above, via the BSU-Security API.

L. MIPS Memory Map

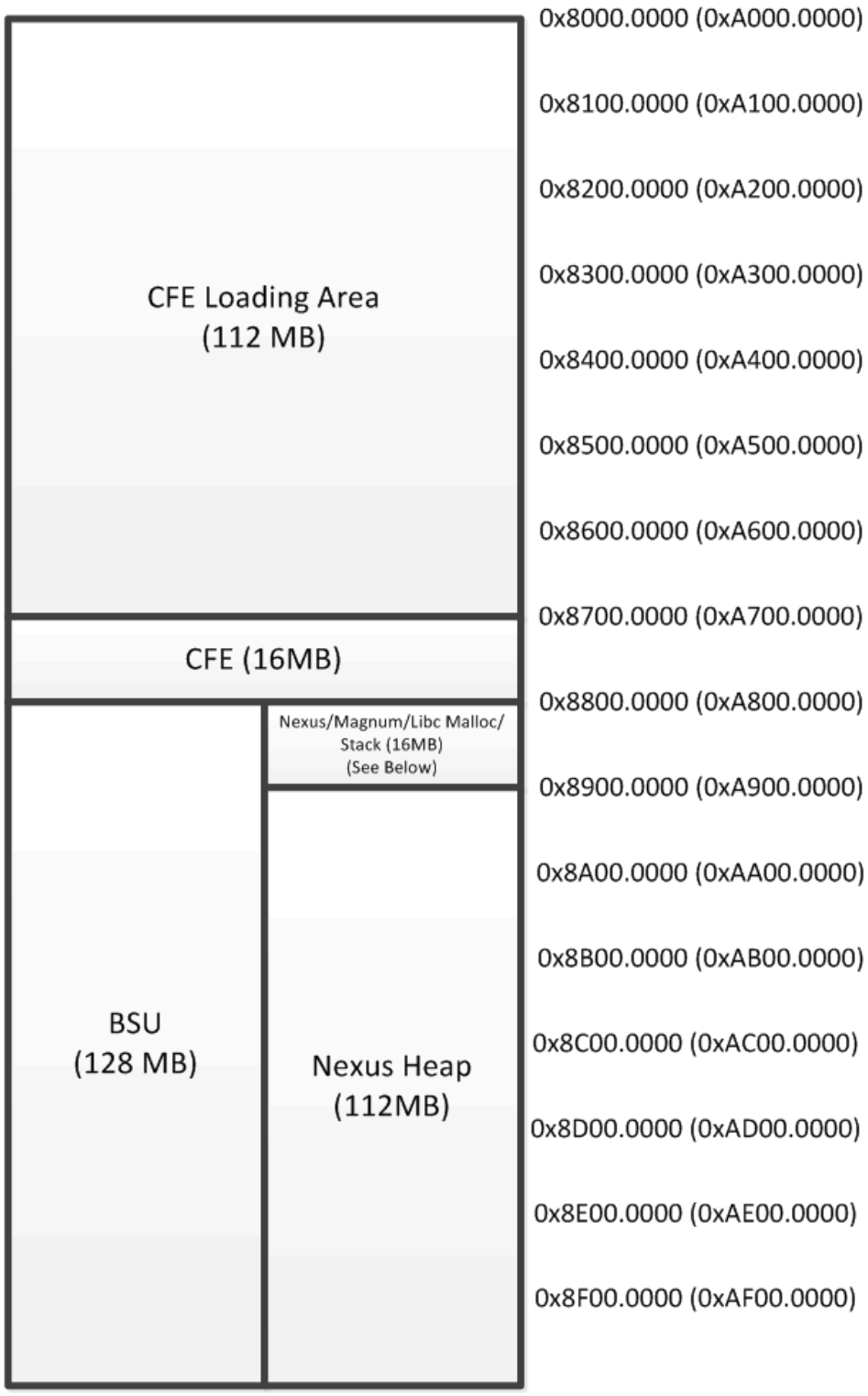
BSU MIPS Memory Map (512MB MEMC0 System)



BSU MIPS Memory Map (1024MB MEMC0, 1024MB MEMC1 System)



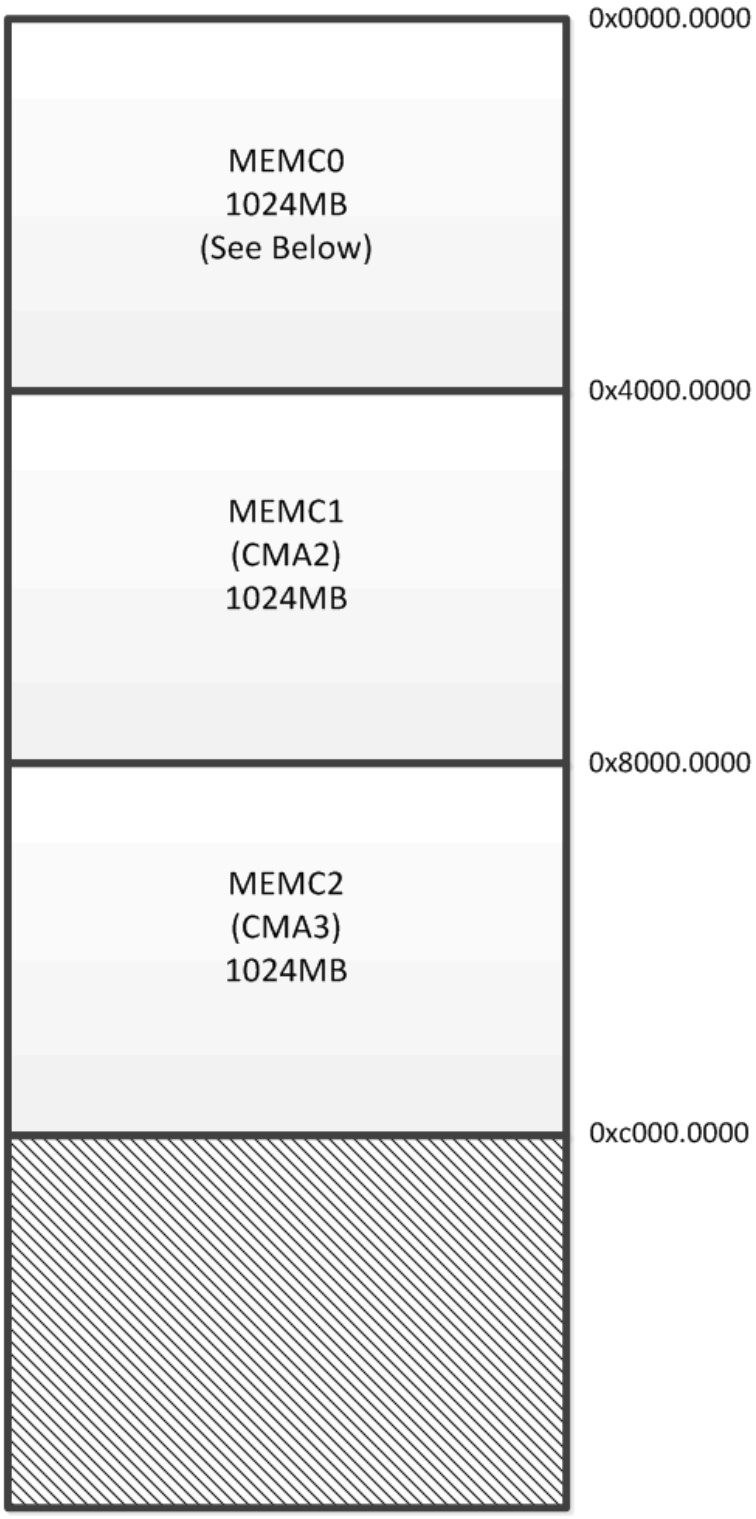
BSU MIPS KSEG0 (KSEG1) 256MB Memory Map



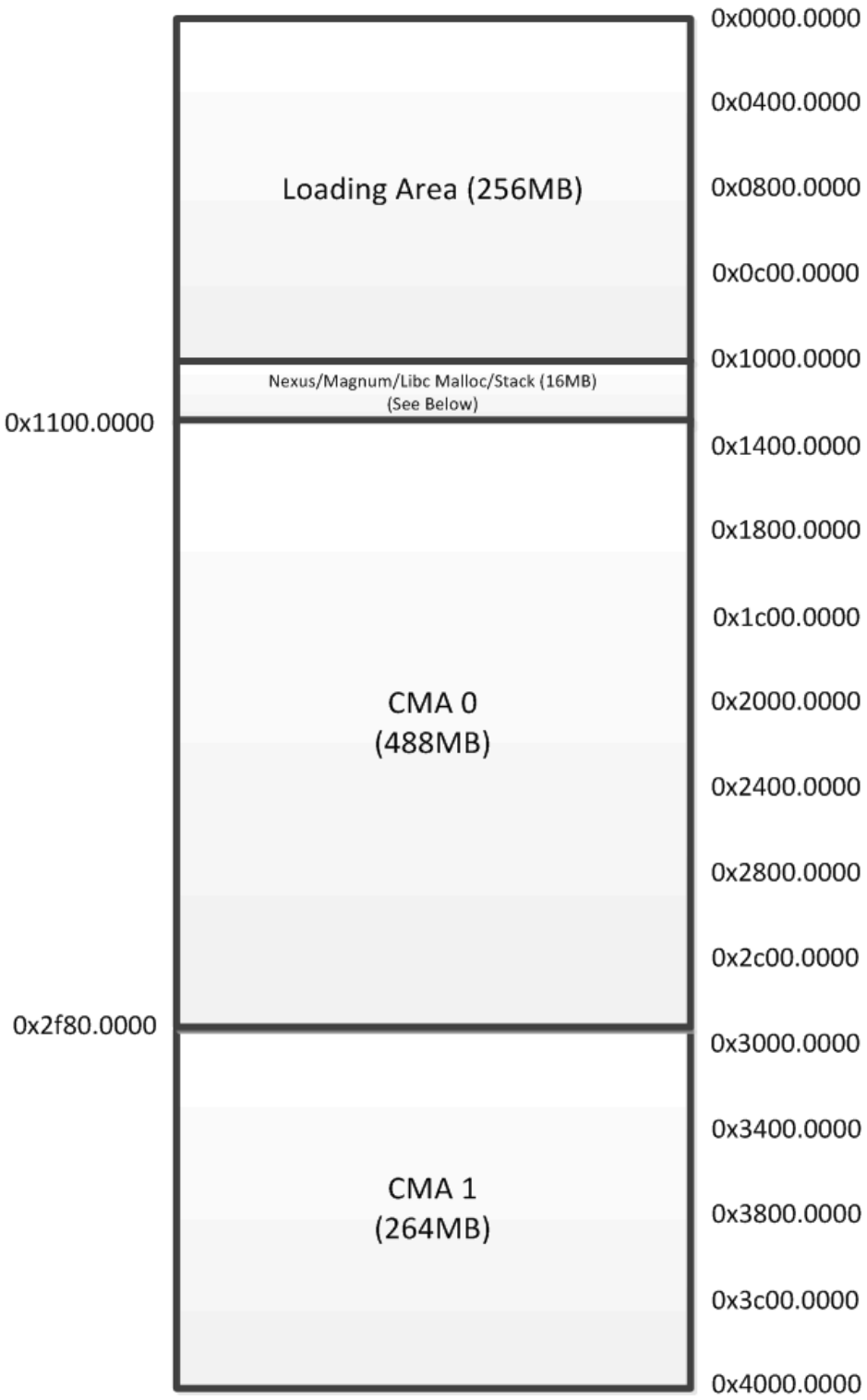
Nexus/Magnum/Libc Malloc/Stack Memory Map

M. ARM Memory Map

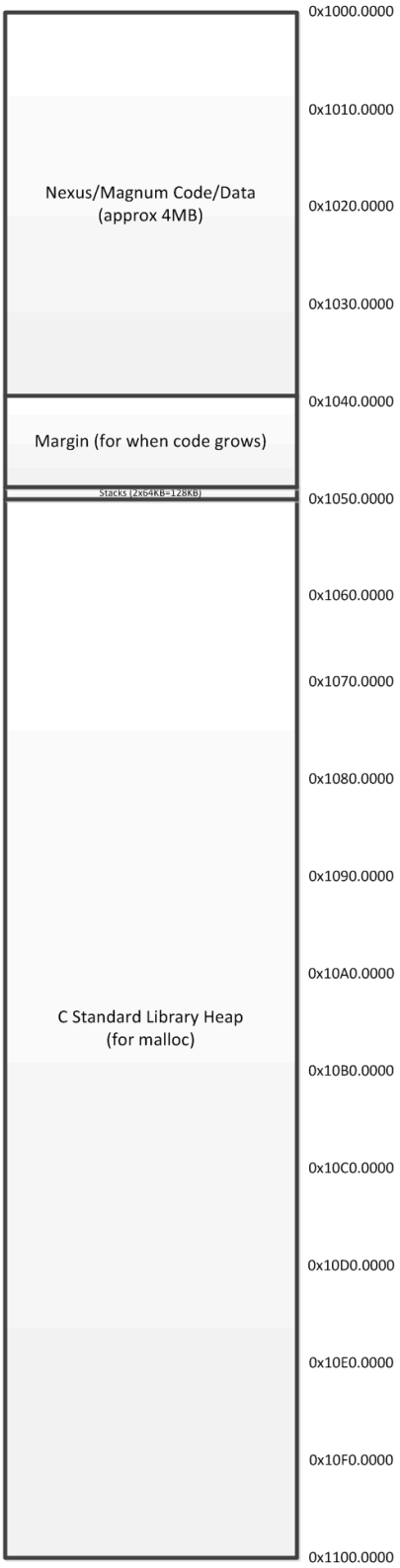
BSU ARM Memory Map (1GB/2GB/3GB System)



BSU ARM Memory Map (MEMC0)



Nexus/Magnum/Libc Malloc/Stack Memory Map



N. Directory Structure

The BSU portion of the Unified Reference Software Release is shown below:

Directory	*Notes*
\BSEAV\app\bsu	***Main BSU directory***
\BSEAV\app\bsu\build	***BSU build directory containing makefiles***
\BSEAV\app\bsu\build\arm	***BSU build ARM specific files***
\BSEAV\app\bsu\docs	***This document resides in this directory***
\BSEAV\app\bsu\src	***BSU source and include files***
\BSEAV\app\bsu\gui	***BSU gui related source files (used when performing "make")***
\BSEAV\app\bsu\src\test	***BSU test related source files (used when performing "make BUILD=test")***

Table 6: BSU Directory Structure

Additionally, it is required (because it contains the flash and USB interface) to obtain the CFE source release for the platform of interest. For phase 14.3's version of the CFE, the main directory structures are shown below:

Directory	Notes
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/bin	***CFE binary files***
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/docs	***CFE documentation files***
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/src	***CFE source files***
BRCM_CFE_Generic_14.3_E1/release_unified_cfe_v14.3/src/unified_cfe_v14.3/v14.3/CFE_V14_3/	***This is where the main source files reside. This is also where the Makefile variable CFE_DIR_NAME should point to.***

Table 7: CFE Directory Structure

Similarly, to having the CFE, it may be required to have the BOLT release for the platform of interest. This directory structure is shown below (for v0.88):

Directory	Notes
BRCM_BOLT_Generic_0_0_88_E1/release_bolt_v0.88/	***Main BOLT directory***
BRCM_BOLT_Generic_0_0_88_E1/release_bolt_v0.88/bin	***BOLT binary files***
BRCM_BOLT_Generic_0_0_88_E1/release_bolt_v0.88/src/BRCM_Src_BOLT_Generic_0_0_88_E1/bolt_v0.88	***This is where the main source files reside. This is also where the Makefile variable BOLT_DIR_NAME should point to.***

Table 8: BOLT Directory Structure

The output directories which the BSU build process will create is shown below:

Directory	Notes
\BSEAV\bsu\build\97362.sde.debug	***Temporary build output directory (97362 shown as an example), containing object and dependency files.***
\obj.97362	***Final output directory (97362 shown as an example)***
\obj.97362\BSEAV	***…***
\obj.97362\BSEAV\bin	***Contains the main bsu.gz and bsu.bin.gz output files.***
\obj.97362\nexus\bin	*Contains the nexus and magnum library files.*
\obj.97362\nexus\bin\exists	*Empty directory used by the build system*
\obj.97362\nexus\core	***Contains inc files built on the fly by the build system.***
\obj.97362\nexus\core\exists	***Empty directory used by the build system***
\obj.97362\nexus\core\syncthunk	*Contains the source and include files built on the fly by the build system for thunking each module.*
\obj.97362\nexus\core\syncthunk\exists	*Empty directory used by the build system*
\obj.97362\nexus\core\ucos_iii.sde	*Contains the nexus and magnum library files. Contains all of the base and module subdirectories, which contain the object files and dependency files and exists subdirectory.*

Table 9: BSU Output Directory Structure