Settop Reference Software User Guide

STB_RefSw-SWUM100-D2

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

10/22/08

REVISION HISTORY

Revision	*Date*	*Change Description*
STB_RefSw-SWUM100-D2	10/23/08	Initial release.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

TABLE OF CONTENTS

LIST OF FIGURES

Section 1: Introduction

REFERENCE SOFTWARE DELIVERABLES

This document provides an overview of the Settop Reference Software, including the Settop API, Brutus, and Nexus. It is included with each release of the Broadcom Settop Reference Software release. Each Reference Software release contains all the software and documentation needed to run a Broadcom Settop Reference Platform. You should have received the following:

Hardware Deliverables

- Settop box
- "One For All" programmable remote control

Software Deliverables

- Linux kernel source
- Linux kernel binaries
- Linux uclibc root filesystem binary (source available upon request)
- Linux uclibc toolchain binary (source available upon request)
- CFE bootloader binary (source available upon request)
- Reference software source

Documentation Deliverables

- Release Notes
- Reference Software User Guide (this document)
- Porting Interface Documentation
- Brutus Usage and Debugging Guides
- Settop API documentation
- Nexus documentation (Nexus platforms)
- SQA Test Results

QUICK START

If you would like to get up and running quickly, please refer to the *Brutus Installation Guide* (document number STB_Brutus-SWUM20x) for instructions on how to configure, build, and run Brutus as a part of the Broadcom Reference Software.

Section 2: Reference Software Stack

INTRODUCTION

The Reference Software Release is a complete top-to-bottom set-top box software stack. It consists of four main layers, as illustrated in Figure 1. The layers differ depending on whether Nexus is included in the stack, and the Nexus one is shown on the right.

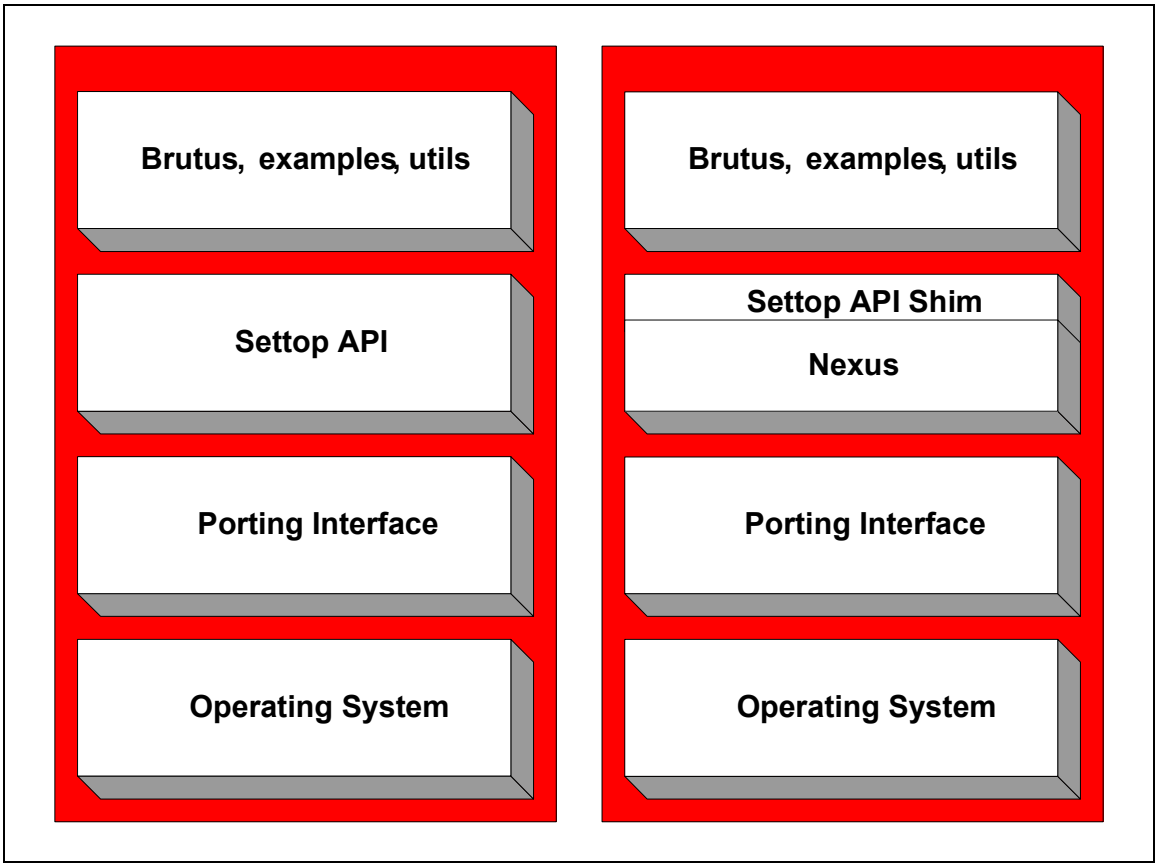

Figure 1: Reference Software Stack

BRUTUS OVERVIEW

Brutus is an easy-to-use application that demonstates a wide range of features on Broadcom's Settop reference platforms. The actual functionality depends on the platform.

Brutus is not intended to be a production-quality application. It demonstrates a wide set of features in an integrated fashion. The source code for Brutus can be used as example code for real applications.

Some of the features supported in Brutus are:

- Front-end Interfaces
  - QAM64, QAM256, QAM1024
  - QPSK DVB and Direct TV, 8PSK
  - VSB
  - Analog RF and A/V line input
- S-Video, Composite, Component, DVI/HDMI, RF modulator, L/R Audio, SPDIF outputs
- Single or dual decode
- Single or dual output
- PIP (Picture-in-Picture)
- Channel change
  - PSI scanning
  - Programmable channel map
- PVR
  - Single or dual record and playback
  - Time-shifting (simultaneous record and playback)
  - Single or dual channel encoder
  - Wide range of host, decoder and STC trick modes (including Broadcom proprietary trick modes like 1x rewind)
- DES/3DES PVR encryption and decryption
- GUI
  - Broadcom embedded windowing system and widget set
  - Anti-aliased TrueType fonts supported through FreeType
  - Image rendering (including PNG, JPEG and BMP support)
  - Wide range of user input devices supported
  - Single or dual instance GUI
- On-screen status
  - Constellations
  - Front-end and back-end status
  - Playback and decode FIFO monitors

The Reference Software team also develops numerous example applications, utilities and test applications. These applications are included in the source code tree in the following locations:

- BSEAV/api/examples
- BSEAV/api/utils

SETTOP API OVERVIEW

The Settop API is an application-level C-language API that offers cross-platform, cross-OS portability. It implements a single usage model that is easy to understand and extensible to support the features of the Broadcom Reference Settop systems. Some of the Settop API's design goals include:

- Easy to learn and use
- Designed to be safely extended without breaking existing applications
- Support for a full range of features

The Settop API handles a wide range of system programming tasks internally. These include:

- Interrupt handling
- Memory management
- Multi-threading and synchronization
- Optimal file I/O for PVR
- Multiple OS support

This API facilitates the writing of set-top applications.

SETTOP API SHIM LAYER

To support Settop API applications on Nexus platforms, Broadcom developed the Settop API Shim layer. This allows Settop API customers to port their existing applications to Nexus platforms. For example, the Brutus application runs unmodified on Settop API and Nexus platforms using the Settop API Shim layer.

NEXUS OVERVIEW

Nexus middleware is Broadcom's next generation of middleware for Settop Box applications. Nexus middleware is designed to address the demands of modern Settop Box applications. Nexus middleware design goals are:

- **Granularity**: Nexus provides control over every part of the system
- **Modularity**: Maximize reuse, yet still allow for customer features
- **Synchronization**: Provide an automatic and efficient sychronized system
- **Customization**: Allow customers to develop customized software without modifying Broadcom software deliverables.
- **Convergence of Settop Box and Digital Television (DTV) Features**: Add video enhancement features to set-top box products, add PVR features to DTV products, etc.

For a detailed discussion of Nexus middleware, please refer to the *Nexus Architecture Guide* (document number STB_Nexus-SWUM10x) and the *Nexus Usage Manual* (document number STB_Nexus-SWUM20x).

PORTING INTERFACE OVERVIEW

The porting interface is a low-level C-language API that offers full control of the Broadcom set-top chips. This API provides the following:

- Modularization for each block on the chip
- Consistent API and design methodology
- Support for interrupt safety through clearly marked ISR code
- Portability across multiple operating systems like Linux and VxWorks
- Support for single and multi-threading systems
- Fully ANSI C-compliant

On the BCM7038-based platform the porting interface is code-named “Magnum.” For previous platforms, it was referred to simply as the Porting Interface. This layer also includes other elements such as base modules and system libraries.

OPERATING SYSTEM OVERVIEW

Broadcom supports a variety of operating systems, including Linux and VxWorks. While each layer in the architecture requires some operating-system-specific code, a large majority of the code is platform-independent. The operating system consists of the following components:

- Bootloader
- BSP (Board Support Package)
- Kernel
- Root Filesystem (Linux only)
- Toolchain

Source code for all Linux modules can be obtained upon request.

Section 3: Brutus

Brutus is an easy-to-use application that demonstrates a wide range of features on Broadcom Reference Settop platforms.

HIGH-LEVEL DESIGN GOALS

Brutus completes the following goals:

- Integrates the many features of the Settop reference platforms into a cohesive application.
- Is configurable to support many of the usage modes of the Settop reference platforms.
- Is easy to use.
- Has a well-designed graphical user interface.
- Provides intuitive use of input devices like IR remotes.
- Supports all Broadcom Settop reference platforms, including both satellite and cable.
- Runs on multiple operating systems, such as Linux and VxWorks.
- Runs on both disk-based and diskless systems.

BRUTUS ARCHITECTURE

The Brutus application consists of several modules as shown in Figure 2 on page 7. The only interface to any chip is through the Settop API.

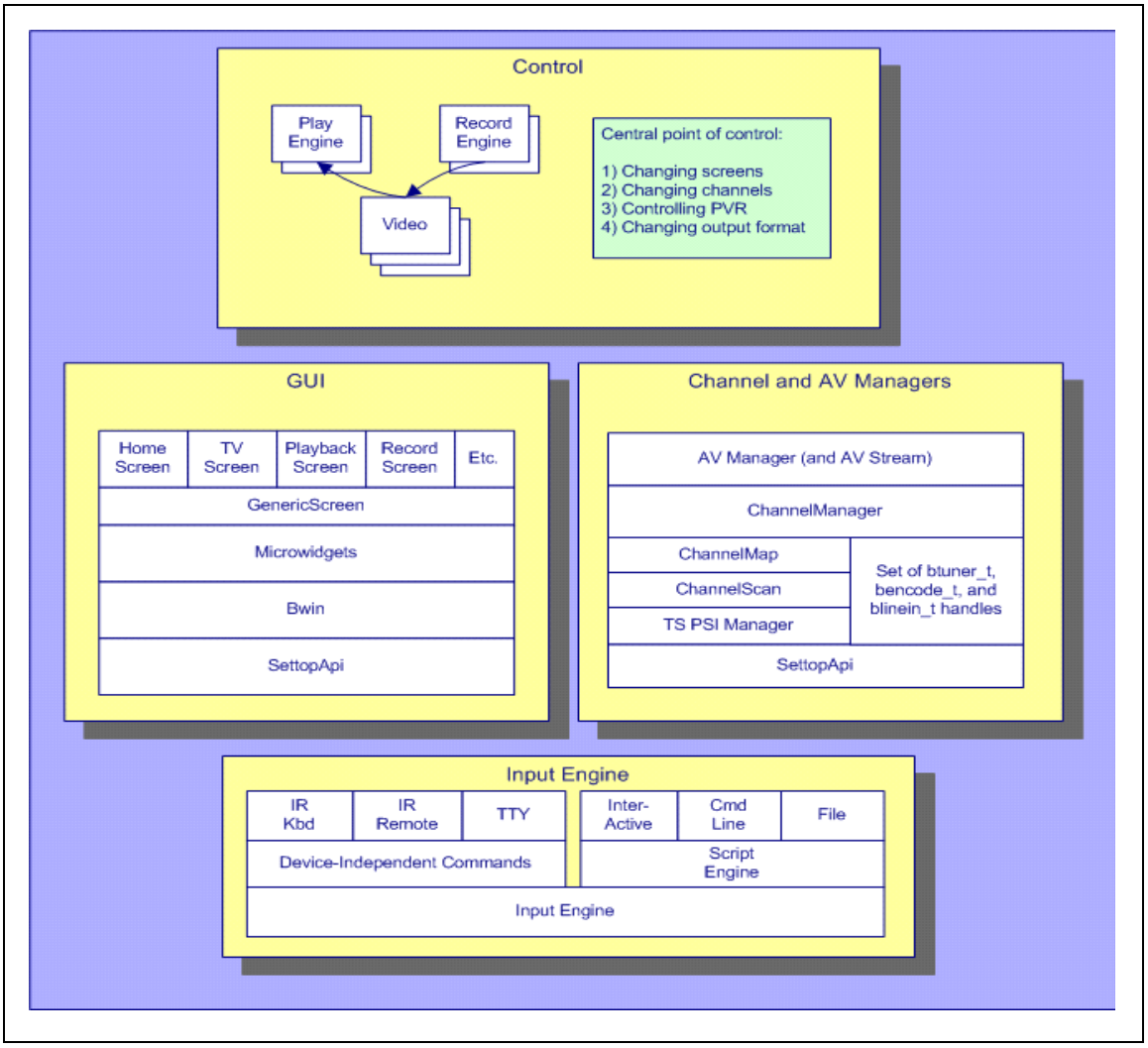

Figure 2: Brutus Architecture

BRUTUS MODULES

Control

The Control class coordinates the various components of Brutus. It keeps a list of screens and facilitates changing screens and resizing decode windows. It manages a set of Play and Record engines for doing Personal Video Recording (PVR). It manages channel changes, including incremental channel digit entry, and deferral of the start of decode in order to speed up repeated channel changes. It also manages the change of display formats, which involves rescaling graphics and possibly restarting decode.

Some of the engines have callbacks directly into the user interface. For instance, PlayEngine can notify the PvrTimeline widget that its state or position has changed. Control is responsible for setting up these connections.

Graphical User Interface (GUI)

Brutus's UI is built on top of a series of software layers. The layers include:

- Settop API graphics
  - Part of the Settop API that provides frame buffer functionality
- bwin
  - A minimal windowing system with font- and image-rendering support
  - Written in C
  - Source code is included under BSEAV/lib/bwin
- Microwidgets
  - A TV-friendly widget library
  - Written in C++
  - Modeled after Trolltech’s Qt library
  - Source code is included under BSEAV/lib/mlibs

The UI follows a single-threaded, cooperative, multitasking model. This means that most application work is performed during idle time when the UI is done painting. Idle time processing must usually be done in short intervals (no blocking); otherwise the UI response time is slow.

Channel Manager and AV Manager

The Channel Manager takes a channel map as an input. The channel map is a file that lists frequencies, symbol rate and other information needed to obtain channels. By default, this file is channels.txt.

The Channel Manager scans the listed frequencies for PSI information. It uses the tspsimgr library to parse the PSI information and build a channel list. It makes this channel list available to Brutus.

The Channel Manager also keeps the set of tuners which can be checked out for exclusive use.

The AV Manager wraps the Channel Manager and adds the concept of an AV Stream. The AV Stream encapsulates everything needed to decode or record a stream, including its bstream_t (analog and/or digital), btuner_t, bencode_t, blinein_t, and the channel number currently on the stream.

Input Engine

Brutus manages a variety of input devices. See Brutus Input Devices for details. The devices are plugged into Bwin's event loop so that Brutus can multiplex the I/O and need not create a thread per input device. Each input device's commands must be translated to a set of device-independent commands.

The input engine also manages Brutus scripting. There are three scripting modes available:

- Interactive mode–This mode uses the **`./brutus --exec`** syntax, and lets the user type in scripting commands at a prompt. Type **`help`** to see the available commands.
- Script file–Load and execute script files stored on the file system. You can specify the files on the command line or select from an onscreen list through the Admin screen.
- Command line text–Specify script commands to run on the command line. This is helpful for testing or if you want to come up in a specific initial state.

Other Optional Features

The following features have special uses and are not shown in Figure 2 on page 7:

- Cheops: A Broadcom-proprietary media protocol used for streaming media and metadata over Ethernet and wireless connections
- XML and HTTP libraries: Small, efficient, Broadcom-proprietary libraries used by the Cheops protocol

BRUTUS USAGE GUIDE

Brutus can be customized using channel maps, program guides, and configuration files. Please refer to the *Brutus Usage Guide* (document number STB_Brutus-SWUM30x) for more details.

Section 4: Settop API

The Settop API provides a high-level C-language API for Broadcom set-top boxes.

HIGH-LEVEL DESIGN GOALS

The goals of the high-level design include:

- Portability
  - The Settop API was written in ANSI C.
  - The same API (with binary compatibility) must run on all Broadcom cable and satellite set-top boxes.
  - It must support multiple operating systems such as Linux and VxWorks.
- Simplicity
  - It is simple to understand and use.
  - The Settop API must hide platform-specific settings and architecture.
- Completeness
  - It supports a full set of features for each reference platform.
  - Not every possible option is supported, but it supports all common scenarios supported by each platform.
- Safety
  - Invalid usage modes must result in immediate error codes.
  - Source code that uses the Settop API as designed will be resilient to any API changes in the future. This is because all public structures require initialization functions and can be extended without impact.
  - All private structures are kept private in the source code. Opaque handles are used.
- Readiness
  - You can refer to the Settop API as example code to get your own system up and running.
  - You can use the Settop API as production quality code.
  - You can modify the Settop API to meet your own needs.

DATA FLOW DIAGRAM

Figure 3 illustrates how the various modules of the Settop API work together. To better understand the usage of the Settop API, use this diagram to trace the flow of execution.

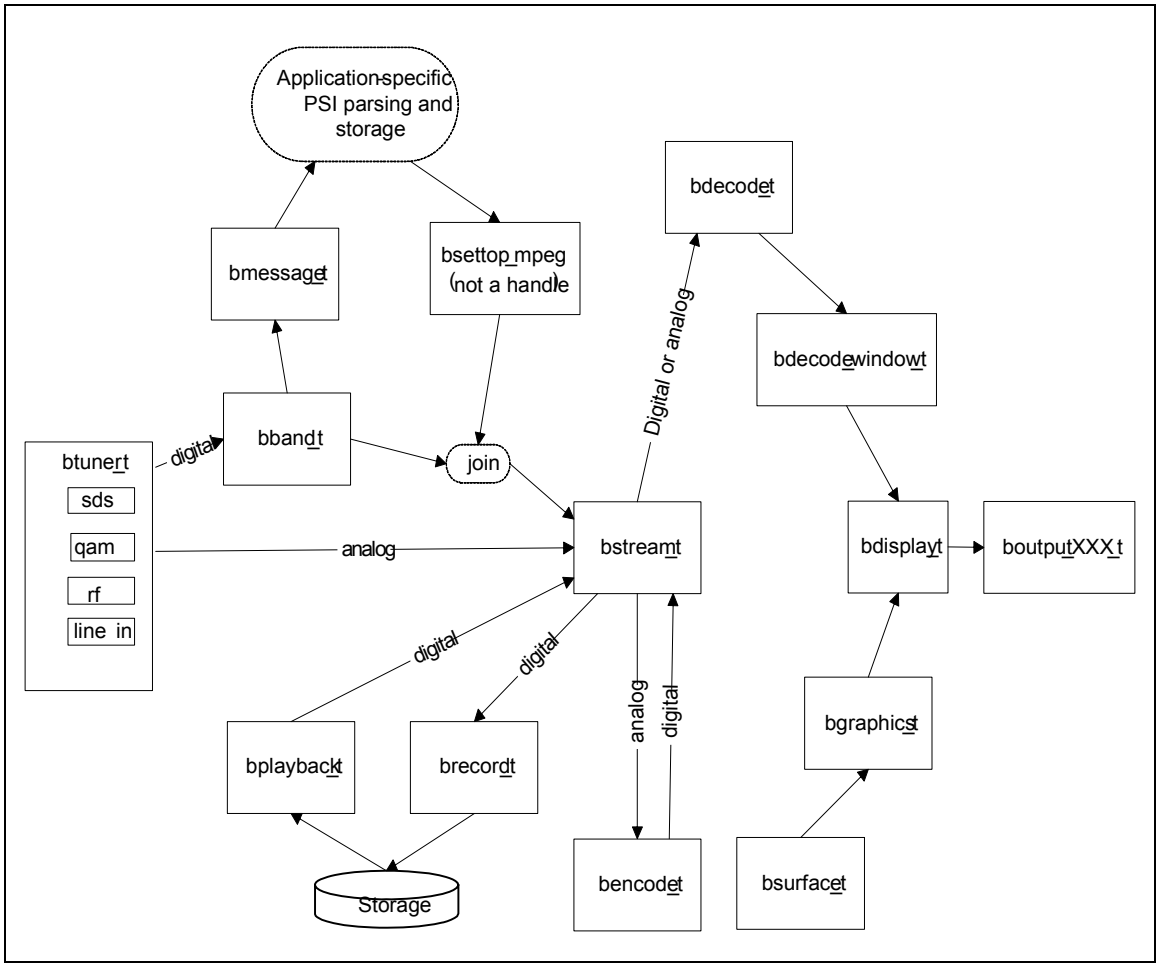

Figure 3: Settop API Data Flow

The Settop API is supported on all Broadcom Settop platforms. Because of changes in Broadcom's porting interface architecture, the Settop API implementation has several main "build systems." Regardless of the build system, the public API remains the same. This is an internal concern, but you will see this distinction throughout this documentation and the code.

NON-NEXUS PLATFORMS

- Kernel or user mode supported
- The Porting Interface is found in magnum/portinginterface
- Settop API implementation is largely in BSEAV/api/src/magnum

NEXUS PLATFORMS

- Kernel or user mode supported
- Support for Settop API applications
- The Porting Interface is found in magnum/portinginterface
- The Settop API Shim layer implementation is in BSEAV/api/src/nexus
- Nexus implementation is largely in nexus/modules

PROXY

- A thin layer for marshalling Settop API/Nexus calls across a system boundary including the kernel/user or network layer
- No porting interface implementation
- On non-Nexus platforms, Settop API implementation is largely in BSEAV/api/src/proxy
- On Nexus platforms, Nexus implementation is largely in nexus/platforms/PLATFORM/src/linuxkernel, where PLATFORM represents a specific platform, for example, 97405 (the BCM97405) or 97325 (the BCM97325).

DRIVER MODELS

In the generic reference software stack, the location of device drivers is not specified. This is because device drivers can be placed in a variety of locations in the stack depending on the operating system and the design methodology. There are three main driver methodologies: kernel mode, user mode using no driver. These methodologies are illustrated in Figure 4 on page 13 and Figure 5 on page 14.

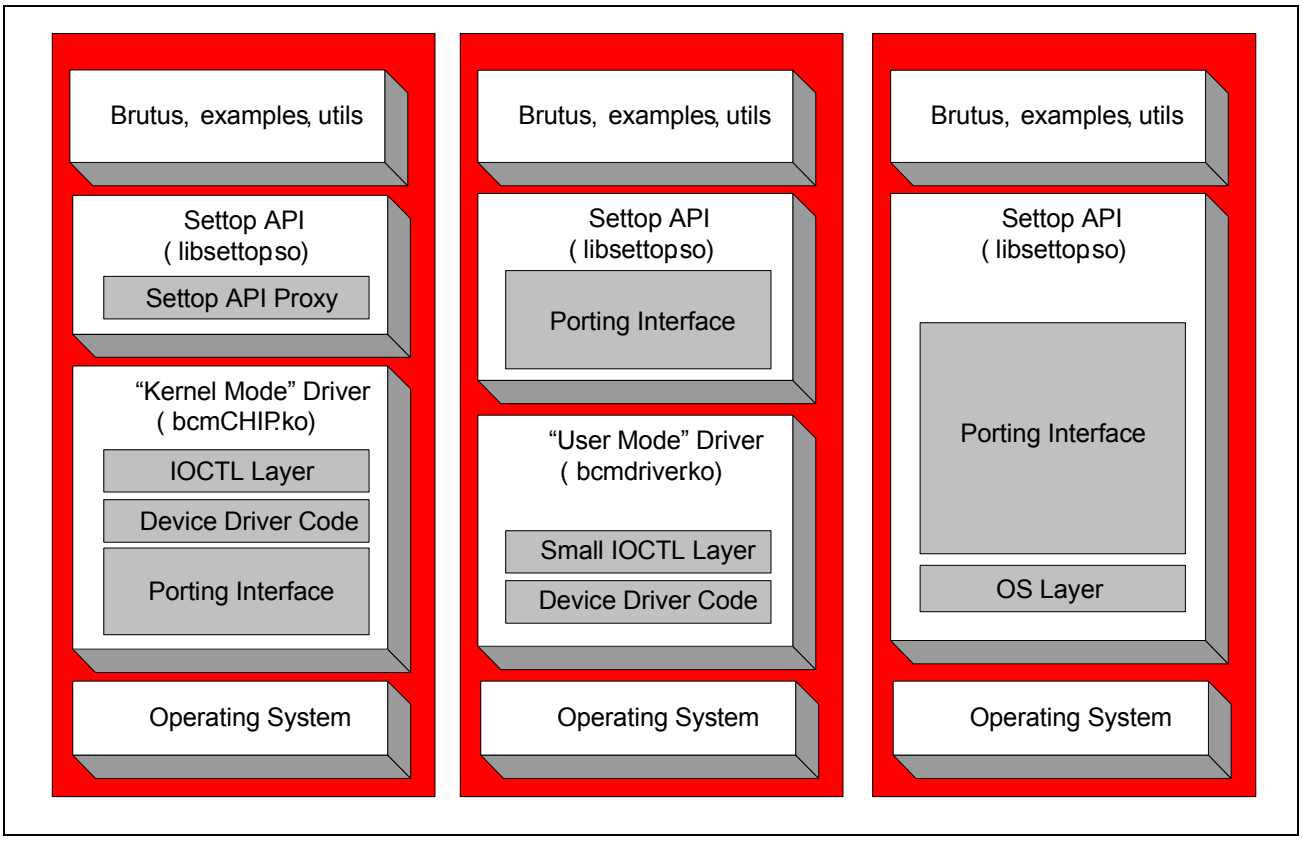

Figure 4: Driver Models (non-Nexus Platforms): Kernel Mode, User Mode, No Driver

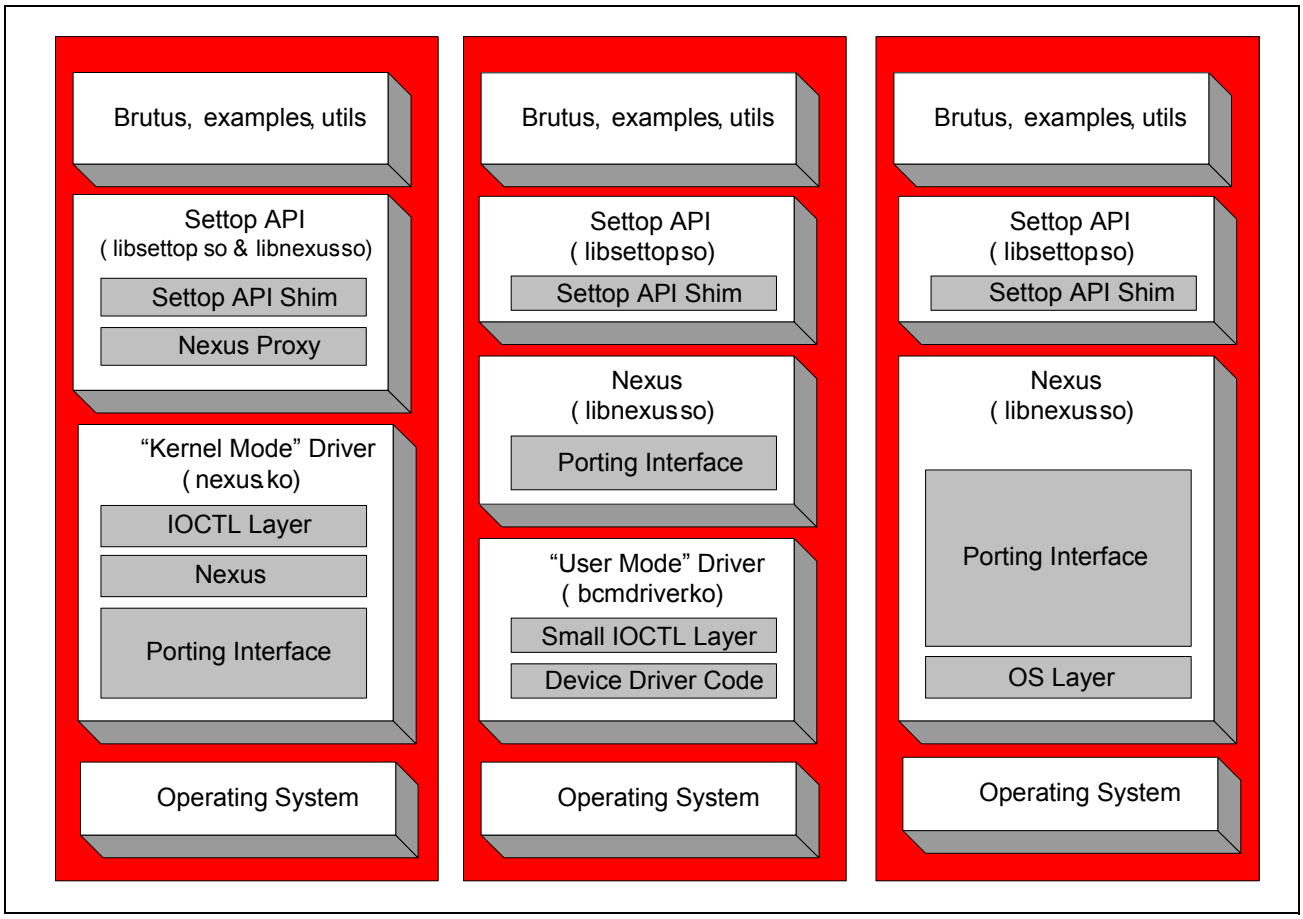

Figure 5: Driver Models (Nexus Platforms): Kernel Mode, User Mode, No Driver

KERNEL MODE DRIVER

Non-Nexus Platforms

In this architecture, both the Settop APIs and porting interface run inside the driver in kernel mode. The IOCTL layer must expose all Settop API interface functionality from kernel mode to user applications. This can be used on operating systems such as Linux where there are user and kernel modes. Multiple applications can run concurrently and the driver can handle concurrency.

Nexus Platforms

On Nexus platforms, the Nexus and porting interface run inside the driver in kernel mode. The IOCTL layer exposes all Nexus interface functionality from kernel mode to user applications. This can be used on operating systems such as Linux where there are a user and kernel modes. Multiple applications can run concurrently and the driver can handle concurrency.

To support Settop API application, Nexus platforms adds a Settop API Shim layer. The Settop API Shim layer is responsible to map Settop API calls to Nexus calls. The Settop API Shim layer is not required for application written for Nexus middleware.

THE USER MODE DRIVER

Non-Nexus Platforms

In this architecture, both the Settop APIs and porting interface run outside the driver in user mode. A minimal IOCTL layer is used to handle interrupts and provide access to registers and memory. A dedicated thread running at high priority is used to service interrupts. Only one application can run the porting interface at a time.

Nexus Platforms

Similar to non-Nexus platforms, the Nexus and porting interface run in user mode. A minimal IOCTL layer is used to handle interrupts and provide access to registers and memory. A dedicated thread running at high priority is used to service interrupts. Only one application can run the porting interface at a time.

Once again, to support Settop API application, Nexus platforms adds a Settop API Shim layer. The Settop API Shim layer is responsible to map Settop API calls to Nexus calls. Settop API Shim layer is not required for application written for Nexus middleware.

NO DRIVER

On some operating systems, like VxWorks, there is no kernel mode/user mode distinction. Thus there is no driver. In this case, the system runs all porting interface and OS-specific code from Settop API/Nexus.

On Nexus platforms, to support Settop API applications, Nexus platforms adds a Settop API Shim layer. The Settop API Shim layer is responsible for mapping Settop API calls to Nexus calls. Settop API Shim layer is not required for application written for Nexus middleware.

SETTOP API MODULES

BCIPHER

The cipher interface provides access to the PVR DES/3DES encryption and decryption engine. It supports decrypting or encrypting MPEG-2 transport data off-line, as well as encrypting or decrypting any raw data for other applications.

BCONFIG

The config interface allows the application to get information about the board, its high-level features, and all chips on the board.

BDECODE

The decode interface is used to decode both digital and analog streams. It controls both video and audio decode.

BDISPLAY

A display is a coordinated set of video/audio outputs that are usually sent to one TV set. Each display may have multiple outputs (e.g. svideo, composite), but each output will carry the same content. Decode windows are boxes of video content on a display. The decode windows and audio volume are controlled through the display API.

BENCODE

The encoder API works by receiving an analog stream and returning a digital stream. After encoding has started, the analog stream is still valid and can be decoded while the digital stream is either decoded or recorded. Alternately, you can decode the digital stream and ignore the analog stream.

BGRAPHICS

The graphics interface supports Broadcom's newest frame buffer graphics model as well as the multiple-surface graphics architecture. All pixel formats are supported along with chromakey and alpha blending effects.

BMESSAGE

The message PI allows you to collect MPEG-2 transport messages, allowing applications to parse PSI or PSIP message formats.

BPCM

The PCM interface is used to write PCM data to a PCM playback channel. PCM playback can be used for sound effects or music (often with a CPU-based decoder for formats like MP3).

BPVR

PVR is a common name for playback and recording of digital video using a set-top box's hard drive. It is also referred to as "DVR" (Digital Video Recorder). The PVR interface of the Settop API handles all threading, indexing, and file I/O issues involved in implementing a PVR. It supports two index formats, as well as playback with no index. Indexes are required for trick modes. The Settop API supports trick modes and time-shifting (simultaneous playback and record of the same stream).

BSMARTCARD

The SmartCard interface supports a read/write interface to the SmartCard, along with reset functions.

BSTREAM

The stream interface is used to manipulate digital and analog streams. A stream can be decoded, encoded, recorded, and played back.

BTUNER

The tuner interface represents the front end of the set-top box. For digital and analog signals, the tuner API performs both tuning and demodulation functions.

BUSER_INPUT AND BUSER_OUTPUT

The user I/O interface provides support to LEDs (including 7-segment LEDs), IR remote controls, the front panel keypad, and IR keyboards.

BVBI

The bvbi API supports closed-captioning, teletext and other text stream formats. It can decode and capture VBI from an analog or digital stream. The VBI information can be routed out to certain analog displays (e.g. composite and svideo) or captured as data for use by the application.

SETTOP API PROGRAMMING REFERENCE

The Settop API has complete API-level documentation that is generated from source code comments. You will find a compressed archive of documentation in the release.

SETTOP API EXAMPLE APPLICATIONS

The following examples show how a simple Settop API application works. More examples can be found in the source code under BSEAV/api/examples.

LIVE DECODE

```
#include "bsettop.h"
#include <stdio.h>

int main(void)
{
    bresult rc = bsettop_init(BSETTOP_VERSION);
    btuner_t tuner = btuner_open(B_ID(0));
    bdecode_t decode = bdecode_open(B_ID(0));
    bdisplay_t display = bdisplay_open(B_ID(0));
    bdecode_window_t window = bdecode_window_open(B_ID(0), display);
    btuner_qam_params qam;
    bstream_t stream;
    bstream_mpeg mpeg;
    bband_t band;
    if (rc) goto error;

    btuner_qam_params_init(&qam, tuner);
    qam.symbol_rate = 5056900;
    qam.mode = 64;

    bstream_mpeg_init(&mpeg);
    mpeg.video[0].pid = 0x21;
    mpeg.audio[0].pid = 0x24;
    mpeg.audio[0].format = baudio_format_ac3;
    mpeg.pcr_pid = 0x21;

    band = btuner_tune_qam(tuner, 765000000, &qam);
    if (!band) goto error;

    stream = bstream_open(band, &mpeg);
    if (!stream) goto error;

    if (bdecode_start(decode, stream, window))
       goto error;

    getchar(); /* press enter to stop decode */
    return 0;
    error:
         return 1;
/* No close curly brace? */
```

PVR ENCODE AND RECORD

```
#include "bsettop.h"
#include <stdio.h>

int main(void)
{
   bresult rc = bsettop_init(BSETTOP_VERSION);
   btuner_t tuner = btuner_open(B_ID(0));
   bencode_t encode = bencode_open(B_ID(0));
   brecord_t record = brecord_open(B_ID(0));
   btuner_analog_params analog;
   bstream_t analog_stream, digital_stream;
   brecord_file_t file = brecord_file_open("stream.mpg", "stream.nav");
   brecord_params record_params;
   if (rc) goto error;

   bencode_settings_init(&encode_settings, bencode_quality_best, encode);
   btuner_analog_params_init(&analog, tuner);
   brecord_params_init(&record_params, record);
   record_params.index_format = bindex_format_bcm.

   analog_stream = btuner_tune_rf(tuner, 63.25, &analog);
   if (!analog_stream)
      goto error;

   digital_stream = bencode_start(encode, analog_stream, &encode_settings);
   if (!digital_stream)
      goto error;

   if (brecord_start(record, digital_stream, file, &record_params))
      goto error;

   getchar(); /* press enter to stop decode */
return 0;
/* No close curly brace? */
```

SETTOP API UTILITIES

There are several utility applications included with the Settop API. Source code can be found under BSEAV/api/utils. These include the following utilites:

- decode: Tune from any source and decode it.
- getpids: Tune a digital source and scan for PSI information.
- playback: Play an mpeg file and perform trick modes if an index is present. This utility includes a command prompt for in-depth testing.
- record: Tune a digital source or encode an analog source and record to disk.
- userio: Read/write from the front panel.

Section 5: The Porting Interface

The Porting Interface is a low-level C-language API that offers full control of the Broadcom set-top box chips.

- Starting with the BCM7038, the Porting Interface was rearchitected under the code-name “Magnum.” The main concepts of the Porting Interface remain the same between the old and new architectures. You can refer to both the new and old architectures as “the Porting Interface.”
- The higher-level syslibs and the lower-level base modules help support the Porting Interface. Therefore, even though only the middle layer is technically the Porting Interface, Broadcom refers to the entire block as “the Porting Interface.”

HIGH LEVEL ARCHITECTURE

Figure 6 illustrates how the Settop Reference Software interacts with the platform hardware and the operating system.

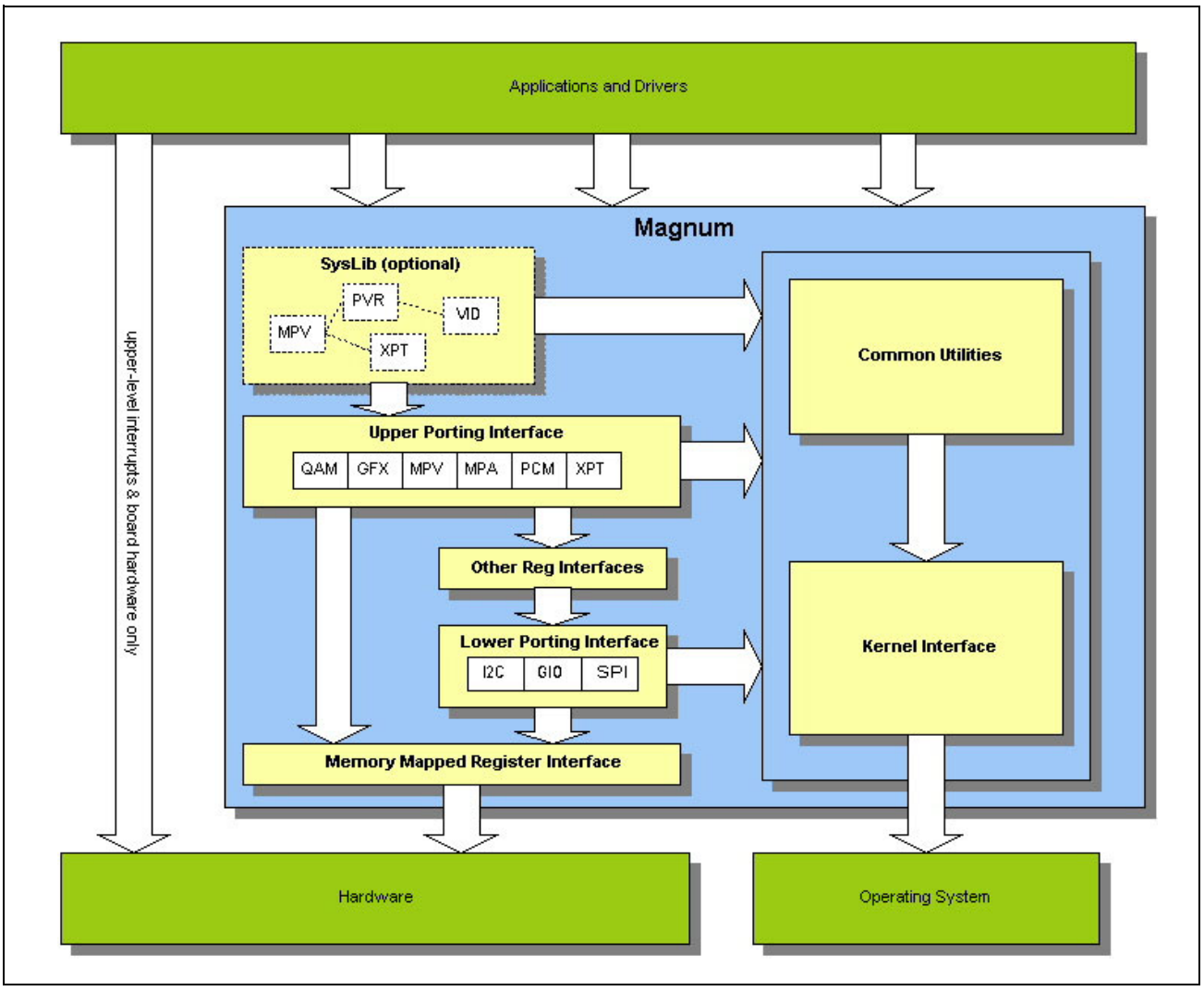

Figure 6: Settop Architecture Block Diagram

DOCUMENTATION

Extensive design and API documentation for Porting Interface is included the standard software release bundle.

Broadcom Corporation

5300 California Avenue
Irvine, CA 92617
Phone: 949-926-5000
Fax: 949-926-5203