Atlas / Brutus 2.0

Users Guide

Revision History

Revision	Date	Change Description	Editor
1.0	05/23/2013	Initial version	D. Tokushige
1.1	06/12/2013	URSR 13.2 Release	D. Tokushige
1.2	07/23/2013	Updated Audio/Video codecs in Atlas.cfg	D. Tokushige
1.3	03/13/2014	Update auto start procedure, screen captures, and Lua commands	D. Tokushige
1.4	04/04/2014	Added VBI related Lua commands	D. Tokushige
1.5	04/04/2014	Added VBI window and Lua content mode command	D. Tokushige
1.6	04/11/2014	Added atlas.setPowerMode() Lua command and power screen descriptions.	D. Tokushige
1.7	05/01/2014	Fix missing parameter in Lua commands for scanXXX()	D. Tokushige
1.8	05/23/2014	Fix scanQam Lua command "annex" options	D. Tokushige
1.9	07/16/2014	Update default remote type and add supported UHF remote information.	D. Tokushige
1.10	07/30/2014	Add Lua commands: setHdmiInput, setSpdifInput, setAudioProcessing	D. Tokushige
1.11	08/12/2014	Remove temp 2160p settings. 2160p is now handled automatically (URSR 14.3 or later)	D. Tokushige
1.12	09/05/2014	Added playback channel type to channels.xml	D. Tokushige
1.13	09/25/2014	Add updated power management screen and descriptions.	D. Tokushige
1.14	10/29/2014	Updated procedure to auto-run atlas from rc.user. Enables atlas lua prompt.	D. Tokushige
1.15	11/18/2014	Add audio screen settings descripions	D. Tokushige
1.16	12/01/2014	Correct setAudioProcessing Lua command	D. Tokushige
1.17	02/24/2015	Updated atlas.scanSat() function	D. Tokushige
1.18	03/03/2015	Updated screenshots of GUI (except tuner screen). Added "adc" option to satellite channels.xml grammar. Added RF4CE remote information.	D. Tokushige
1.19	03/05/2015	Add network screen and netapp build option information.	D. Tokushige
1.20	05/11/2015	Add option to run Atlas as background process.	D. Tokushige
1.21	08/14/2015	Add Atlas Client/Server IP Streaming, Auto	D. Tokushige

		Network Discovery, Bluetooth remote image, and Performance Tools.	
1.22	08/14/2015	Update Lua commands and alpha sort.	D. Tokushige
1.23	10.16.2015	Update Channel Map entry for ip Channels On Channels.xml	J. Rubio
1.24	11/03/2015	Add Bluetooth Remote ESR PN #. Notes on placing remote in Pairing mode.	R. Jew
1.25	12/02/2015	Add instructions on how to build and run Bluetooth A2DP.	R. Jew
1.26	12/11/2015	Correct channelStream() Lua command argument list. Update hdmi output Lua command argument list.	D. Tokushige
1.27	01/25/2016	Updated playlistDiscovery() and playlistShow() to allow programmatic control of IP streaming and discovery. All Lua commands now return -1 on error and 0 otherwise. Atlas.debug() command is new.	D. Tokushige
1.28	02/19/2016	Remove NEXUS_MODE from NxClient build instructions	D. Tokushige
1.29	03/09/2016	Update playlistDiscovery() and playlistshow() Lua documentation to include return parameters and reference to sample script.	D. Tokushige
1.30	05/25/2016	Atlas can do Udp streaming using bip UDP streamer. Use udpUrl.xml	J. Rubio
1.31	05/27/2016	Added mosaic channel list format in channels.xml	D. Tokushige
1.32	07/21/2016	Updated Lua command list by adding ip transcoding related command.	M. Patel
1.33	08/02/2016	Updated the Lua command list by adding the ip client transcode (enable and profile set) related commands.	M. Patel
1.34	12/16/2016	Add integrated WiFi build option	D. Tokushige
1.35	01/09/2017	Update main menu and display screenshots and add description/Lua command of CPU Test feature.	D. Tokushige
1.36	01/24/2017	Add Wifi Protected Setup screenshots and description/Lua command.	D. Tokushige
1.37	01/27/2017	Wifi auto connection feature, WPS, and restart procedure	D. Tokushige
1.38	03/07/2017	Add wifiConnectState() Lua command.	D. Tokushige

1.39	03/20/2017	Add getChannelStats() Lua command	J. Rubio
1.40	03/20/2017	Update IP Channels with interface option	J. Rubio
1.41	05/31/2017	Add Lua commands for HDR PLM setting. Added HDR PLM section and included example mosaic channel showing PLM settings, graphics, and video window manipulation.	D. Tokushige
1.42	07/13/2017	Add Lua commands to enable/disable IR remote handling and to show all available Lua commands.	D. Tokushige

Table of Contents

Figures

Tables

Introduction

Atlas is a demonstration application that is delivered as part of the Broadcom Set-Top Reference Software release. It offers a wide variety of functionality in an integrated architecture and provides an example of how the Set-Top Reference Software can be used to create a final product. It also provides a means to test and use Broadcom reference boards, with actual functionality depending on the platform. While Atlas is designed primarily as a test or demonstration application, its source code can be used as the basis for real consumer applications.

This document describes how to configure, build, and install the Atlas application as a part of the Broadcom® Set-Top Reference Software.

Software Terms

- **Bootloader**—The code that is first executed when the set-top powers up which is responsible for initial configuration of the hardware and loading the kernel.
- **Kernel**—The operating system which controls the set-top and allows the reference software drivers and applications to run
- **Root file system**—On some operating systems (e.g., Linux®) it is customary to have at least a minimal disk layout for management of applications and libraries
- **Toolchain**—The compiler and linker that runs on your build server and creates binaries that can run on the Set-Top.
- **Reference Software**—The source code delivered by Broadcom to control the features of the Broadcom Reference Platform.

Prerequisites

- A Broadcom Reference Board running Linux Kernel
  - Refer to the *BroadcomReferencePlatformSetup.pdf* document for instructions.
- A Build Server and Network Setup Ready
  - Refer to the *BroadcomReferencePlatformSetup.pdf* document for instructions
- Knowledge of how to use `vi` (a Linux text editor) and be able to edit text on the set-top.
- Root access to the Linux build server and be able to switch between root and user mode when instructed.
- The capability of connecting the set-top and build server to the same Ethernet network.
- A dynamic host configuration protocol (DHCP) server running on the network that the set-top can use.
- The ability to run the `bash` shell.

Building Atlas

Configure Linux Kernel Source

Part of the build process is building Linux drivers. Building drivers requires that the compiler have access to the Linux kernel header files. These header files must be the same ones used to build the actual Linux kernel you are running on the set-top.

The Makefile uses the $(LINUX) environment variable to override the location of the Linux kernel source. By default, it is `/opt/brcm/linux`. You can override it if needed.

The reference software's driver Makefile does not attempt to configure the Linux kernel source for your chip. Instead, it checks if autoconf.h exists (in $(LINUX)/include/linux/ for 2.6.31, in $(LINUX)/include/generated/ for 2.6.37 and later). If it does not, the driver build will fail. You should configure your kernel source, then build the driver again. If autoconf.h does exist, the driver Makefile assumes that the kernel source has been configured correctly.

Expand Reference Software Source

The reference software source is delivered as a tarball. You can untar it in place:

```
tar zxvf refsw_release_unified-YYYYMMDD.src.tgz
```

When you unpack your source, you will see three (or more) subdirectories that contain source code.

- **BSEAV**—Reference Software drivers, libraries, and applications
- **magnum**—Porting interface for magnum platforms and some application utility code
- **rockford**—Magnum-based application and system code
- **nexus**—Nexus API and source code

Environment Variables

Reference Software makefiles rely on the following user defined environment variables which must be exported prior to building source code.

Export	Description
NEXUS_PLATFORM=97445	Platform
BCHP_VER=A0	Chip version for the main chip on board (all uppercase)

B_REFSW_ARCH=arm-linux	Underlying core/OS
LINUX=<<kernel source>>	Location of Linux kernel source.
PATH=$PATH:<<toolchain>>	Location of toolchain used to cross-compile and link.
<<board subtype>>	Additional export indicating specific Broadcom reference board subtype which may limit available features. See the platform-specific Release Notes.

<div align="center">

Table 1 - Atlas Build Environment Variables

</div>

Feature Options

Use the following compile-time exports to enable/disable optional Atlas features. The values listed below indicate the default option setting.

Export	Description
NETAPP_SUPPORT=n	For platforms with wireless capabilities, use this export to include Wifi scan/connection support. This option is incompatible with WLAN_SUPPORT.
WLAN_SUPPORT=n	For platforms with **integrated** wireless capabilities (97271), use this export to include WiFi scan/connection support. This option is incompatible with NETAPP_SUPPORT.
DCC_SUPPORT=y	Controls digital closed captioning support.
POWERSTANDBY_SUPPORT=y NEXUS_POWER_MANAGEMENT=y	Determines power management support. Both exports should be set in a similar fashion.
PLAYBACK_IP_SUPPORT=y LIVEMEDIA_SUPPORT=y	Add IP support
NXCLIENT_SUPPORT=n	Enable Atlas to run in NxClient mode.
PERFTOOLS_SUPPORT=n	Enable bsysperf and bmemperf tool support.

<div align="center">

Table 2 – Optional Feature Build Environment Variables

</div>

User Mode vs Kernel Mode Configuration

Broadcom Reference Software supports Magnum and Nexus running in either Linux user mode or kernel mode.

When running in user mode, API calls from the application are handled like normal function calls. When running in kernel mode, API calls from the application are marshaled across the kernel/user boundary using the Proxy Nexus public API.

The Proxy API translates all Nexus public API calls into Linux I/O controls (IOCTLs). User and kernel memory mapping is handled automatically. The calling application does not see anything differently; it looks exactly like a C API. The full Nexus public API implementation is then compiled into the driver, running in kernel mode. This means that there are two implementations of the Nexus public APIs running on the system: the full implementation in kernel mode and a proxy implementation in user mode.

The benefit of a kernel mode implementation is that interrupts are handled at interrupt time and, therefore, have less latency. The detriment of a kernel mode implementation is that every Nexus public API call must be marshaled across the kernel/user boundary.

The Proxy implementation is automatically generated by Perl scripts. They parse the public header files and create the IOCTL's, user mode code, and kernel mode code.

See nexus/build/os/linuxkernel/module_rules.pl.

To build a kernel-mode version of Atlas, simply export NEXUS_MODE=proxy before calling make:

```
cd BSEAV/app/atlas/build
export PLATFORM=97445
export BCHP_VER=A0
export NEXUS_MODE=proxy
make clean
make install
```

After building, you find drivers, libraries, and an application in the BSEAV/../obj.PLATFORM/BSEAV/bin directory. The only difference is that the kernel mode driver (nexus.ko) is much larger than the user mode driver (bcmdriver.ko), and the shared libraries are much smaller in the kernel mode build.

Little Endian vs Big Endian Linux

All the directions listed so far assume little-endian Linux. To switch to big-endian, you only need to make a few changes:

1. Switch the endianess jumper on the board from little- to big-endian.
2. Flash the big-endian CFE bootloader using Broadband Studio.

You cannot flash big-endian CFE using little-endian CFE. You must already have the board in big-

endian mode, which prevents you from running little-endian CFE.

3. Install and boot big-endian Linux

3. Big-endian Linux has separate binaries, but not separate source.

4. Configure kernel source on the build server for big-endian Linux.

5. Set the B_REFSW_ARCH variable when building Brutus

B_REFSW_ARCH defaults to mipsel-linux, where "el" stands for "endian little." You should set:

```
cd BSEAV/app/atlas/build
export PLATFORM=97231
export BCHP_VER=B0
export B_REFSW_ARCH=mips-linux
make clean
make install
```

Be aware that PVR .nav files are in host endianess. You must rerecord your existing streams for big-endian systems.

Compile and Link

The Atlas makefile will build Atlas, and also all the required drivers and libraries.

```
bash
cd BSEAV/app/atlas/build
make clean
make
```

Executing the following command will combine all required runtime files into the proper directory structure and compress it into a tar file called refsw-YYYYMMDD.NEXUS_PLATFORM-linux-atlas.bin.tgz.

```
make install
```

Altas build files will reside in a directory outside of the source file tree. This "out-of-source" build allows users to build for multiple platforms without overwriting the resulting executables.

```
BSEAV/../obj.NEXUS_PLATFORM/BSEAV/bin
```

Initializing Hard Disk Drive for PVR

In most cases, the hard disk drive on the set-top is only needed for the PVR. If you are using a hard drive-based root file system, then you should use the stbutil after booting an initrd version of Linux. See BroadcomReferencePlatformSetup.pdf for more instructions. The method using stbutil creates four partitions, including /dev/sda4 for PVR.

If you are using a Flash, NFS, or initrd root file system, the following directions show you how to manually configure your hard drive for PVR. You can partition the hard drive with one partition for the entire disk.

Here are some simple instructions:

```
fdisk /dev/sda
Command: p
Command: d 1
Command: d 2
# Delete whatever partitions happen to be there

Command: n
# Select p, then 1. Then accept all defaults in order to make a
# single maximum-sized partition.

Command: w

# See note below on ext2 usage
mkfs.ext4 -E stripe-width=32 -i 262144  /dev/sda1
tune2fs -c 0 -i 0 -o journal_data_writeback /dev/sda1

# The following mount command must be performed every
# time the set-top boots.
mount /dev/sda1 /data
```

Broadcom recommends using the ext4 file system with a limited number of inodes, and with the stripe-width and journal_data_writeback options shown in the example above.. Another option is to use ext2, which has no journal. By limiting the number of inodes in the file system, you decrease ext2 fsck time to make it very comparable to the journaling ext4 filesystem. The ext3 filesystem is not recommended for PVR.

> **Note:** If you choose ext2, make sure to mount with an explicit "-t ext2" just in case the drive was formatted as ext3. If you use ext3, Direct I/O will not work and PVR performance suffers.

The Atlas "start install" procedure looks for /data and creates a videos directory for PVR. When it is time to make your own root file system image which autoboots Atlas, you will add the command to mount this partition.

Installing Atlas

There are 3 ways to install Atlas on the set top box:

1. Install Atlas on a remote server and use NFS to access it.

2. Install Atlas in flash from a remote server over NFS. Once installed Atlas runs from flash.

3. Install Atlas in flash from a remote server over TFTP. Once installed, Atlas runs from flash.

Install Using NFS to Run Across Network

Now you are ready to run Atlas. The easiest way to do this is to mount your Build Server using NFS and run Atlas across the network. You should have already made your source directory mountable in the previous steps. If you cannot, you have to move the contents of obj.[PLATFORM]/BSEAV/bin to a directory that is mountable.

Follow these steps from the set-top, assuming your mount point on the set-top is /mnt/nfs. Replace `?????/BSEAV/bin` with whatever mount point you chose on your build server.

```
mount y.y.y.y:??????/BSEAV/bin /mnt/nfs

# Unpack and run Atlas over NFS
cd /mnt/nfs
tar zxvf refsw-YYYYMMDD.NEXUS_PLATFORM-linux-atlas.bin.tgz
start install
start atlas
```

In this method, you are unpacking the tarball across NFS. The "start install" step configures device nodes and creates some directories and symlinks to /data Using "start atlas" loads your drivers, verifies that you are linking to the right shared libraries, and starts the application—Atlas should start.

Install to Flash Using NFS

The recommended method of installing the application to flash is to mount the flash file system in Linux and copy the Atlas binary via NFS.

 Writing to flash through Linux can take several minutes.

The following instructions are examples only. Things that may vary include your installation location, hard drive partitions, mount point, etc.

```
# Remount the flash filesystem with write privileges (if necessary)
mount -o remount,rw /

# Mount NFS to load the Atlas binary
mount yyyy:??????/BSEAV/bin /mnt/nfs

# Mounting the hard-drive is required before "start install"
mount -t ext2 /dev/sda1 /mnt/hd
# Unpack the Atlas tarball into /home/atlas, which writes
# to flash.
mkdir /home/atlas
cd /home/atlas
tar zxvf /mnt/nfs/ refsw-YYYYMMDD.NEXUS_PLATFORM-linux-atlas.bin.tgz
start install
```

```
# Optionally you can make it autoboot by creating this file:
cat >/root/rc.user
mount /dev/sda1 /mnt/hd
mount /dev/sda4 /data
cd /home/atlas
start atlas &
# Press Ctrl-D to save the file

mount -o remount,ro / #(if remounted earlier)
sync;sync

# You can now start Atlas
start atlas
```

Install to Flash Using TFTP Server (No NFS)

If you are running a Windows-only setup, then you may not have NFS available. You can still load Atlas into flash using TFTP, such as the PumpKIN TFTP server.

This process requires that you use the flash and a temporary ramdisk; it does not work with the initrd kernel. The following directions assume you have already flashed the kernel and are using a read-only flashed root file system.

```
# Boot linux and login as root

# Create a ramdisk and load the tarball into it
mkdir /tmp/mnt
mount -nt tmpfs -o size=8192k,mode=777 /dev/null /tmp/mnt
cd /tmp/mnt
tftp -g -r refsw-YYYYMMDD.NEXUS_PLATFORM-linux-atlas.bin.tgz y.y.y.y

# Note that y.y.y.y is the IP address of your TFTP server. Writing Atlas to
flash might take several minutes. Each file is printed separately.

# Untar from ramdisk into flash
mount -o remount,rw / #(if mounted ro)
mkdir /home/atlas
cd /home/atlas
tar zxvf /tmp/mnt/ refsw-YYYYMMDD.NEXUS_PLATFORM-linux-atlas.bin.tgz
start install
mount -o remount,ro /#(if remounted earlier)
sync;sync

# You can now start Atlas
start atlas
```

Set Atlas to Auto-Start on Bootup

Therefore, to make Atlas autostart create a file called /root/rc.user, similar to the following (specifics may vary):

```
mount /dev/sda1 /mnt/hd
mount /dev/sda4 /data
cd /home/atlas
/bin/cttyhack start atlas
```

Do not use a trailing `&` in the last line – doing so will cause issues with the Atlas Lua command prompt.

Once you've made Atlas autostart, if you do not get a root prompt, you will need to boot with an initrd kernel and mount flash to modify the `rc.user` file.

```
#Reboot and Ctrl-C if your set-top currently autoboots Linux
#Boot the initrd kernel from TFTP server

Login: root

ubiattach -m 0
mount -t ubifs ubi0:rootfs /mnt/flash

cd /mnt/root
mv rc.user rc.user.bak
cd /
umount /mnt/flash
# reboot
```

You may want to configure a set-top that automatically starts Atlas by default, yet also has an option to abort that start. This can be accomplished with a simple uclinux script. Here's a sample:

```
mount /dev/sda1 /mnt/hd
mount /dev/sda4 /data
echo "Press ENTER in 5 seconds to abort Atlas"
REPLY=nothing
read -t 5
if [ x$REPLY == xnothing ]; then
    cd /home/atlas
    /bin/cttyhack start atlas
fi
```

If you wish to run atlas in the background, you must disable the Lua prompt with the "-noprompt" command like option:
```
start atlas -noprompt &
```

Running Atlas

```
start install
start atlas
```

The "start install" step configures device nodes and creates some directories and symlinks to /mnt/hd. This step is typically only done once.

Using "start atlas" loads your drivers, verifies that you are linking to the right shared libraries, and starts the application.

If Atlas was built with NxClient support, you will need to start NxServer before running Atlas.
```
nexus nxserver&
```

If you wish to run atlas in the background, you must disable the Lua prompt with the "-noprompt" command like option:
```
start atlas –noprompt &
```

Command Line Arguments

Atlas has the following optional command line arguments:

Export	Description
-script [script filename]	Runs the given Lua script (unlike the runScript() Lua command, you must specify an absolute path).
-noprompt	Runs atlas without the Lua command prompt. This allows you to run Atlas as a background process as follows: start atlas –noprompt &
-cfg [atlas.cfg file]	Use settings in given custom atlas.cfg file.

Table 3 - Atlas Command Line Arguments

Channel List

The Atlas channel list is saved in XML format in a file called "channels.xml". This file can be edited by hand or can be generated automatically after performing a channel scan in Atlas. The channel list XML file begins with a versioned <atlas> tag. In the event of a change in XML format, an incompatible channels.xml file will not be accepted and the user will be notified.

The <stream> tag (and all inclusive tags) can be omitted if pids are unknown. Atlas will search for pids and auto generate new channels when tuned to the given frequency. It will make the first tune a little slower and your channel lineup will be ordered based on tuning order.

Table 4 - PID Video Types contains the list of available "videotype" values for the <pid> tag. Table 5 - PID Audio Types contains the list of available "audiotype" values for the <pid> tag.

Channel List XML Format

```
<atlas version="3.0">
    <channellist name="default">

        <!-- STREAMER Channel -->
        <channel major="1" minor="1" type="streamer">
            <stream number="0" type="ts">
                <pid videopid="17" videotype="mpeg2"></pid>
                <pid audiopid="20" audiotype="ac3"></pid>
            </stream>
        </channel>

        <!-- QAM Channel -->
        <channel major="1" minor="2" type="qam" freq="111000000" mode="256"
            bandwidth="6000000" symrate="5360537" annex="B">
            <stream number="0" type="ts">
                <pid videopid="65" videotype="mpeg2"></pid>
                <pid audiopid="68" audiotype="ac3"></pid>
            </stream>
        </channel>

        <!-- VSB Channel -->
        <channel major="1" minor="3" type="vsb" freq="549000000" mode="8"
            symrate="5381119">
            <stream number="0" type="ts">
                <pid videopid="65" videotype="mpeg2"></pid>
                <pid audiopid="68" audiotype="ac3"></pid>
            </stream>
        </channel>

        <!-- OFDM Channel (mode can be: "dvbt", "dvbt2", or "isdbt") -->
        <channel major="1" minor="4" type="ofdm" mode="dvbt" freq="500"
            bandwidth="8" >
            <stream number="0" type="ts">
                <pid videopid="49" videotype="mpeg2"></pid>
                <pid audiopid="52" audiotype="ac3"></pid>
            </stream>
        </channel>

        <!-- SAT Channel -->
        <channel major="1" minor="5" type="sds" mode="dvb" freq="1207.0"
            diseqc="13" tone="true" adc="0">
            <stream number="0" type="ts">
                <pid videopid="5922" videotype="mpeg2"></pid>
                <pid audiopid="5923" audiotype="mpeg"></pid>
            </stream>
        </channel>


        <!-- IP Channel for you can have type=ip -->
        <!-- in the url you specify type http,udp,rtsp, or rtp -->
        <!-- for auto Pids delete the stream and pid tags -->
        <!-- Optional:Specify the network interface name -->
        <!-- UDP Channel -->
        <channel major="9" minor="1" type="ip" url="udp://193.168.1.103:1234" interface="eth0">
```

```
        <stream>
            <pid videopid="69" videotype="mpeg2"></pid>
            <pid audiopid="68" audiotype="mpeg"></pid>
        </stream>
    </channel>

    <!-- RTP Channel -->
    <!-- Optional:Specify the network interface name -->
     <channel major="9" minor="2" type="ip" url="rtp://193.168.2.1:1235/" interface="eth0" >
    </channel>

    <!-- HTTP Channel -->
    <!-- END of URL you can specify filename or leave it blank  -->
    <!-- HTTP DTCP-IP and SSL Channels are automatically detected  -->
<!-- Optional:Specify the network interface name -->
    <channel major="9" minor="3" type="ip"
            url="http://193.168.2.1:5000/filename.mpg" interface="eth0">
    </channel>

    <!-- Playback Channel -->
<!-- "number" attribute refers to the program number.  If not -->
    <!-- specified, Atlas will use the first program in the playback -->
    <!-- file.  You can see the available program numbers if you cat -->
    <!-- the video's corresponding .nfo file -->
<channel major="99" minor="3" type="playback"
            filename="q64Spiderman.mpg" number="1">
</channel>

    <!-- Mosaic Channel -->
    <!-- This channel is comprised of multiple channel tiles. Each -->
    <!-- platform will only display the number of mosaics it supports -->
    <!-- regardless of the number specified in the channels.xml file -->
    <channel major="98" minor="1" type="mosaic">
        <channel type="ip" url="http://193.168.2.1:5000/filename.mpg">
        </channel>
        <channel type="playback" filename="cnnticker.mpg" number="0">
        </channel>
        <channel type="qam" freq="111000000" mode="256"
            bandwidth="6000000" symrate="5360537" annex="B">
        </channel>
    </channel>

    <!-- Mosaic Channels can also be used to demonstrate HDR PLM -->
    <!--  the mosaic channel can control the graphics PLM setting -->
    <!--  by setting the "gfxplm" attribute.  Sub channels can -->
    <!--  set the "plm" attribute which controls the video window PLM -->
    <!--  Note that in the absence of "gfxplm" or "plm" attributes, -->
    <!--  Atlas will default PLM to "true".  Also notice that graphics -->
    <!--  can be added using the <label> tag.  These graphics will be -->
    <!--  scaled and positioned based on the given coordinates as -->
    <!--  percentages of the screen.  Users can specify these -->
    <!--  percentages to the tenth.  In the example below, the label -->
    <!--  desktop_background.brcmLTD.jpg is scaled based on the full -->
    <!--  screen coordinates.  While it does cover the entire screen, -->
    <!--  video windows will always cut through it unless the zorder -->
    <!--  is increased.  <label> cputest.jpg is a child of a sub -->
```

```
        <!--  channel so its coordinates are determined based on that -->
        <!--  sub channel's origin.  Note that height is not specified - -->
        <!--  height will be auto calculated based on the original image -->
        <!--  dimensions and the given width of 25%.  Because this <label> -->
        <!--  has an increased zorder, it will reside on top of the -->
        <!--  video window. To see what this mosaic channel looks like, -->
        <!--  see the section on HDR Programmable Luminance Mapping (PLM) -->
      <channel major="1" minor="2" type="mosaic" gfxplm="true">
        <label filename="desktop_background.brcmLTD.jpg" text="HDR Demo" x="0" y="0" width="100"
height="100"></label>
        <channel type="playback" plm="true" filename="World_HDR_HD_VP9.webm" x="25" y="5" width="50"
height="50">
           <label filename="cputest.jpg" text="fullscreen" zorder="1" x="70" y="25" width="25"></label>
        </channel>
        <channel type="playback" filename="Geostorm_SDR_AVC.mp4" x="0" y="66.7" width="33.4"
height="33.4"> </channel>
        <channel type="playback" plm="false" filename="Guardians_SDR_HD_AVC.mp4" x="33.4" y="66.7"
width="33.4" height="33.4"> </channel>
        <channel type="playback" filename="Underdog_HDR_HD_VP9.webm" x="66.7" y="66.7" width="33.4"
height="33.4"> </channel>
      </channel>


    </channellist>
</atlas>
```

Playback .nfo Files

Playback .nfo files are also comprised of XML. All supported files in the "videos" directory without a corresponding .nfo file are automatically probed on Atlas startup. If an existing .nfo file's XML is found to be of an incompatible version, it will be recreated.

.nfo files may contain references to multiple streams within a given playback file. Each of these streams will be represented as separate videos in Atlas.

.nfo XML Format

```
<atlas version="4.1">
  <infofile filename="mummy_MI_5element_q64.mpg"
indexname="mummy_MI_5element_q64.mpg" path="videos/">
    <stream number="1" type="ts">
      <pid type="playback" videopid="17" videotype="mpeg2"></pid>
      <pid type="playback" audiopid="20" audiotype="ac3"></pid>
    </stream>
    <stream number="2" type="ts">
      <pid type="playback" videopid="33" videotype="mpeg2"></pid>
      <pid type="playback" audiopid="36" audiotype="ac3"></pid>
    </stream>
    <stream number="3" type="ts">
      <pid type="playback" videopid="65" videotype="mpeg2"></pid>
      <pid type="playback" audiopid="68" audiotype="ac3"></pid>
    </stream>
```

```
        </infofile>
    </atlas>
```

Video Codec	videotype
MPEG-1 (ISO/IEC 11172)	"mpeg1"
MPEG-2 (H.222/H.262)	"mpeg2"
MPEG-4 part 2 (MPEG-4 Visual)	"mpeg4part2"
H.263	"h263"
MPEG-4 AVC (H.264)	"h264"
MPEG-4 AVC Scalable Video Coding (H.264 SVC)	"h264_svc"
MPEG-4 AVC Multiview Video Coding (H.264 MVC)	"h264_mvc"
VC-1 (SMPTE 421M)	"vc1"
VC-1 (SMPTE 421M) simple/main profile	"vc1simplemain"
DivX ;-) 3.11 Alpha	"divx311"
Audio Video Standard (AVS)	"avs"
RealVideo RV40	"rv40"
On2 TrueMotion VP6	"vp6"
On2 TrueMotion VP7	"vp7"
On2 TrueMotion VP8	"vp8"
Sorenson Spark (Sorenson H.263)	"spark"
Motion JPEG (M-JPEG)	"motionjpeg"
High Efficiency Video Coding (HEVC)	"h265"

Table 4 - PID Video Types

Audio Codec	audiotype
MPEG layer I/II	"mpeg"
MPEG layer III	"mp3"
Advanced Audio Codec (AAC)	"aac"

AAC Audio Data Transport Stream (ADTS)	"aacadts"
AAC Low Overhead Audio Stream (LOAS)	"aacloas"
High Efficiency (HE) AAC	"aacplus"
HE AAC LOAS	"aacplusloas"
HE AAC ADTS	"aacplusadts"
Dolby Digital	"ac3"
Dolby Digital Plus	"ac3plus"
DTS Digital Surround	"dts"
Linear Pulse-Code Modulation (LPCM) for DVD	"lpcmdvd"
LPCM for HD-DVD	"lpcmhddvd"
LPCM for Bluray	"lpcmbluray"
DTS High Definition	"dtshd"
Windows Media Audio (WMA)	"wmastd"
WMA TS	"wmastdts"
WMA Pro	"wmapro"
Audio Video Standard	"avs"
Pulse-Code Modulation (PCM)	"pcm"
Waveform Audio File (WAV)	"pcmwav"
Adaptive Multi-Rate Audio (AMR)	"amr"
Dynamic Resolution Adaptation Audio (DRA)	"dra"
Cooker/Gecko/RealAudio G2/RealAudio 8 Low Bitrate	"cook"
Adaptive Differential Pulse Code Modulation	"adpcm"
Low Complexity Subband Coding (SBC)	"sbc"
Legacy DTS	"dtslegacy"
Ogg Vorbis Audio	"vorbis"
LPCM for Firewire	"lpcm1394"
Pulse-Code Modulation (PCM)	"g711"
G.723.1 Audio	"g723_1"
ITU-T ADPCM Speech Codec	"g726"

G.729 (CS-ACELP)	"g729"
Free Lossless Audio Codec (FLAC)	"flac"

Table 5 - PID Audio Types

UDP/RTP Streamer List

The Atlas UDP/RTP Streaming list is saved in XML format in a file called "udpUrl.xml". This file can be edited by hand. The UDP/RTP streamer XML file begins with a versioned <udpStreamer> tag. In the event of a change in XML format, an incompatible udpUrl.xml file will not be accepted and the user will be notified.

Bip UDP/RTP Streamer XML file contains <stream> tags with an entry "url" , the URL is passed into BIP UDP/RTP streamer to start streamer the file while Atlas is running. Once UDP/RTP streaming has started it will not stop until the user quits atlas.

You may change the UdpUrl.xml name as this is user configurable thought the atlas.cfg variable.

UDP_URL_LIST = "udpUrl.xml"   #"Name of the list of UDP/RTP URLS to stream out"

UdpUrl XML Format

```
<udpStreamer version="1.0">
<--sample entries to send out udp or rtp streams using bip UDP/RTP streamer>
<-- Formart is udp(rtp)://ip:port/filename >
<stream url="udp://10.14.80.123:1234/AbcMPEG2HD.mpg"></stream>
<stream url="rtp://10.14.80.123:1236/avatar_AVC_15M.ts"></stream>
</udpStreamer>
```

Atlas.cfg Configuration File

atlas.cfg contains the run time options (name/value pairs) for Atlas. These values are useful to alter the behavior of Atlas without rebuilding but cannot be changed once atlas has been started. The default atlas.cfg lists the internal default values used by Atlas. Any change to a name/value pair in atlas.cfg will override the internal defaults.

Feature Guide

Remote Controllers

Atlas supports the use of multiple remote control devices. The user can specify the current IR remote device type in the atlas.cfg file (see REMOTE_TYPE_PRIMARY). The default primary remote type is "Broadcom".

Atlas supports the following IR remote controls:

URC-8811	URC-8820	Broadcom	URC-Black	E* 148785

Atlas supports the following UHF remote controls:

E* 131961   Directv RC-34

Remote	Type	Mode	Atlas.cfg
OneForAll URC-8811	IR	<CBL>	REMOTE_TYPE_PRIMARY="OneForAll"
OneForAll URC-8820	IR	<CBL/SAT>	REMOTE_TYPE_PRIMARY= "OneForAll"
Broadcom	IR		REMOTE_TYPE_PRIMARY= "Broadcom"
URC-Black	IR	<STB>	REMOTE_TYPE_PRIMARY= "OneForAll"
EchoStar 148785	IR		REMOTE_TYPE_PRIMARY= "CirEchoStar"
EchoStar 131961	UHF		REMOTE_TYPE_SECONDARY="EchoStarUHF"
Directv RC-34	UHF		REMOTE_TYPE_SECONDARY="DirectvUHF"

Table 6 – Remote Controller Atlas.cfg Settings

Atlas supports the following Bluetooth remote controls:

Universal Electronics Bluetooth Remote

Bluetooth remote pairing can be completed using the "Bluetooth" main menu option or using the Bluetooth based Lua commands. Note that other Bluetooth remotes will pair properly and may work, but some keys may not be mapped properly at this time.

ESR part number for this remote: 558374-00

Putting Bluetooth Remote in Pairing Mode:

Hold both "home"(⌂) button and "back"(⤺) button for 5 seconds. When in pairing mode, the red light on the top of the remote should start blinking.

Atlas has preliminary support for the following RF4CE remote control:

Remote Solutions CRB36 FCC ID: TX4CRB36A

Note that there are two versions of this remote. The one pictured above has a logo printed below the 0 button and supports both MSO and ZRC profiles. The version without the logo only supports MSO.

This remote does not require any changes to the atlas.cfg and it will work in conjunction with the specified REMOTE_TYPE_PRIMARY and REMOTE_TYPE_SECONDARY. However, unlike IR and RF based remotes, the RF4CE remote requires additional setup before starting Atlas.

Remote Solutions Setup

Note that RF4CE remote support is NOT automated within the Atlas build system and remote binding is not fully integrated in the application itself. Most of the following setup procedure will be handled programmatically in the future.

1.   Before building Atlas: export RF4CE_SUPPORT=y

2.   On STB, Insmod the kernel mod driver

     a.   cd BSEAV/linux/driver/zigbee

     b.   ./setup_7366C0.sh

3.   On STB, run the Zigbee server process in the background:

Broadcom Corporation Proprietary and Confidential

        a.   cd BSEAV/lib/zigbee/broadbee_mailbox_host/projects/SoC_mailboxHostSide

        b.   ./SoC_mailboxHost.elf stack_binary/broadbee_zrc11_target_v0p06.bin &

4.   After starting Atlas, bind new remote using Atlas Lua command line:

        a.   Atlas lua> atlas.addRf4ceRemote("my remote")

        b.   Complete the follow steps within 30 seconds

        c.   Press and hold the Setup button on the remote control until the indicator LED changes from red to green.

        d.   Press the info button on the remote control.

Limitations

OneForAll URC-8820

This remote looks very similar to other OneForAll remotes, but has one major difference - the bottom two rows of buttons do not operate when in CBL/SAT mode. Unfortunately, means Picture-in-Picture (PiP) functionality is not available using this remote.

EchoStar 148785/131961

This remote is designed to work exclusively with Dish Network satellite boxes and consumer TVs so some of the buttons will not work with Atlas. Volume and Mute always default to TV mode so they will not control Atlas.

Directv RC-34

This remote is designed to work exclusively with Directv Network satellite boxes and consumer TVs so some of the buttons were remapped. The red button controls show/hide Picture-in-Picture feature. The green button controls Picture-in-Picture swap.

Universal Electronics Bluetooth Remote

This remote is has limited buttons so many features like Picture-in-Picture and PVR trick modes are not be available (but you can still use Lua commands).

Remote Solutions CRB36

This remote is has limited buttons so many features like Picture-in-Picture and PVR trick modes are not be available (but you can still use Lua commands).

Wi-Fi Wireless Network Support

Wireless network connectivity is supported for platforms with integrated WiFi. This is accomplished in Atlas through the use of WPA Supplicant. WPA Supplicant and the Broadcom Wireless Lan (WL) drivers are loaded by Atlas on startup and will remain resident even after quitting. Connections to wireless networks will remain after Atlas is closed and Atlas can resume use of the connection when it is restarted.

Building

 Wi-Fi support is available in both Nexus and NxClient Atlas modes and can be enabled/disabled with the following export:

```
export WLAN_SUPPORT=y
```

Note that even if enabling WLAN_SUPPORT, it will only be available to platforms with integrated WiFi support.

Running

Atlas will automatically install the WL drivers and initialize WPA Supplicant on startup if necessary. If the user wishes to force driver reload and WPA Supplicant initialization, they should run the following command:

```
atlas_wlan_prep.sh -f
```

WPA Supplicant saves the last connected network to the wpa_supplicant.conf file. In the event that the WPA Supplicant is stopped and restarted, it will attempt to connect to this saved network automatically. Wpa_supplicant.conf is a text file and can be edited if desired to manipulate the default network connection attempt.

Wpa_cli is a command line interface which can be used to interact directly with WPA Supplicant. Changes made using the wpa_cli application will also be reflected in Atlas. This application is included with WPA Supplicant but is also copied into the Atlas tarball.

Atlas also support WiFi Protected Setup (WPS) button press which allows the user to securely connect to a WPS supported Access Point (AP) without the need for logins or passwords. Users can

click the WPS button on the Atlas Network screen and then press the WPS button on the target AP. Atlas will negotiate a secure connection. See the Graphical User Interface (GUI) section for more details.

Bluetooth Audio Streaming (A2DP)

Advanced Audio Distribution Profile (A2DP) profile defines how multimedia audio can be streamed from one device to another over a Bluetooth connection (also called Bluetooth Audio Streaming). For example, music can be streamed from a settop box, to a wireless headset or wireless speakers.

Building

Bluetooth A2DP is only supported when Atlas is built for nxclient mode. Additionally set these build variables for support:

```
export NETAPP_SUPPORT=y
export NXCLIENT_SUPPORT=y
export NEXUS_MODE=client
```

Running

Similar to discovering and connecting to a Bluetooth remote, place Bluetooth headphones or speakers in discovery mode. Then go to the Bluetooth Menu and here it will discover surrounding Bluetooth devices. Cursor over the a2dp unit and connect. Now audio should be sent to the speakers/headset.

Limitations

-If you are playing Bluetooth audio, audio quality will automatically drop if you are simultaneously scanning for Bluetooth devices.

- Video lip sync with Bluetooth audio is not implemented.

Tuner Sharing

Like all resources in the system, Atlas collects all tuners into a global resource manager. Tuners are checked out for exclusive use whenever a channel is decoded and/or recorded. When you are done decoding and recording, the tuner is checked back in. When you are in playback mode, there is no tuner needed because a live channel is not being decoded. When playback is stopped and live decoding is resumed again, Atlas tries to acquire another tuner.

This dynamic tuner assignment allows single-display and dual-display systems to make full use of the tuner resources without hard-coding which tuner must be used for a certain display.

If the platform has a non-orthogonal set of tuners, however (that is, dual display system with only one QAM tuner), you can scan QAM channels and have them in the channel list, but if you are decoding or recording an QAM channel on one display, the other display will have no tuner available. In this case, the GUI reports a "(no tuner)" message alongside the current channel number. If you stop record or decode on the first display, the tuner will become available. The other display can start decoding or recording the channel.

Watching Live TV

Choosing a Channel

While Atlas contains a default channels.xml file, odds are it will not match what your head-end is broadcasting. To create an appropriate channels.xml file, you can edit it manually using a text editor (see section "Channel List XML Format"), or start a channel scan.

There are several ways to change channels in Atlas:

1. Press the <CH+> and <CH-> buttons on your remote. Rapidly pressing these keys will change the channel number without actually tuning. This allows you to quickly traverse the channel list. Once rapid key pressing stops, Atlas will tune to the given channel. Note that some remotes will allow you to hold the button down for rapid traversal.

2. Type in channel number using 10 key buttons. Atlas uses 2 part channels numbers with a major channel number and minor channel number separated by a dot. If an entered <major.minor> number does not match a channel in the channel list, no tuning will occur. However, entering only the major channel number will automatically select the first channel with a matching major number. For example, if your channel list has the following channels: 1.1, 2.1, 3.4, 3.5, pressing the <3> key followed by <enter> key will result in Atlas tuning to channel 3.4.

3. The <CH+/-> buttons can be used in combination with the 10 key buttons to quickly jump to a point in the channel list. Given the same channel list: 1.1, 2.1, 3.4, 3.5, if the <3> key is pressed followed by the <CH+> key, Atlas will tune to channel 3.5. This allows you to jump quickly to a distant position in a long channel list.

Jump/Last Channel

Atlas remembers the last tuned channel and provides a way to tune back without retyping the channel number. Pressing the <last>, <recall>, <prev>, <jump>, or <enter> button will tune to the last channel. This allows you to toggle between two channels using a single button press.

Volume/Mute

The volume/mute buttons on most supported remotes will allow the user to set audio levels on the set-top box output. Set-top volume is set to 100% by default.

Streaming Recorded or Live TV

Atlas Server

Each instance of Atlas contains a built in streaming IP server with automatic discovery. Atlas server catalogs and streams both playback streams and live channels to Atlas clients. It also implements an automatic discovery mechanism which simplifies connecting one Atlas STB to another across a common network.

Atlas server can be disabled using an option in the atlas.cfg runtime configuration file.

Auto Discovery

Once started, Atlas server will broadcast identifying information about itself to other Atlas STBs on the network. This broadcast will continue at regular intervals which will allow clients to update their playlists on server changes, and will also indicate when a given server disconnects from the network. This discovery and playlist exchange will occur automatically and does not require any interaction from the user.

This automatic discovery mechanism requires IP multicast which may not be supported by the network the STB is connected to. If this is the case, you can connect the STBs to a common router (wired/wireless) or switch, which will enable auto discovery to work.

There are a number of configuration options in the atlas.cfg file that can be used to modify the behavior of auto discovery.

Atlas Client

In addition to being a streaming IP server, each Atlas is also a streaming IP client. Once a remote Atlas server is discovered, an Atlas client can stream content and display it on screen using either the graphical user interface or custom Lua commands.

Atlas can always stream to itself and will have an entry in the list of available playlists representing locally available streams.

Performance Tools

Atlas supports both bsysperf and bmemperf performance monitoring tools. Export PERFTOOLS_SUPPORT="y" before building atlas to enable.

- You will have to start the Boa web server before starting atlas. Cd to obj.XXXXX/nexus/bin and run "boa". Boa will printout a url you can use to connect to the performance tools web page.

- Start atlas

- Using a web browser on your computer (same network as STB), connect to the Atlas IP address given when starting Boa. bsysperf and bmemperf tools will both be available.

Graphical User Interface (GUI)

By default, the graphical user interface is rendered into a 960x540 size surface and then scaled to fit the current display resolution. This can be changed using GRAPHICS_SURFACE_WIDTH and GRAPHICS_SURFACE_HEIGHT variables.

Main Screen

The <menu> button will show/hide the Atlas GUI main menu.

Figure 1 - Menu Window

Each button on the main screen shows additional panels when clicked. The +/- button on each screen allows the user to customize which controls are visible on screen. It may also give access to additional more advanced options that are not available by default. The "Buffers/TSM" checkbox will show/hide a new, persistent menu screen which remains shown even when the menu is hidden.

CPU Test

Most items on the main screen lead to other submenus or popup windows but the CPU test is an exception. The CPU test feature starts/stops tasks (one per CPU core) which continuously decompress a JPEG image in the background to stress test the CPU. There are 11 levels to choose from ranging from OFF to 100% CPU utilization. The range between 1 and 10 can be adjusted using the CPUTEST_MAX_DELAY value in the atlas.cfg configuration file.

Volume

Changing volume either using the remote control or entering Lua commands, will result in a volume meter displayed on screen momentarily. It will be displayed on top of any other on screen windows.
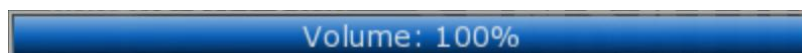

Figure 2 - Volume Popup

Decode Screen

Clicking on the "Decode" button on the Main Screen will show the Decode Screen.

Figure 3 - Decode Window

The decode menu displays decoder specific data about the current channel or playback. There are no editable fields in this menu. Values will update based on the currently decoded stream in the main window.

Display Screen

Clicking on the "Display" button on the Main Screen will show the Display Screen.

Figure 4 - Display Window

This screen allows the user to manipulate display settings. All changes apply immediately except for "auto format" which takes effect on the next channel change.

VBI Screen

Clicking on the "VBI" button on the Display Screen will show the VBI Screen.

Figure 5 - VBI Window

Options on this screen only apply to the standard definition display.

Streaming Player Screen

Clicking on the "Streaming Player" button the Main Screen will show the Streaming Player Screen.
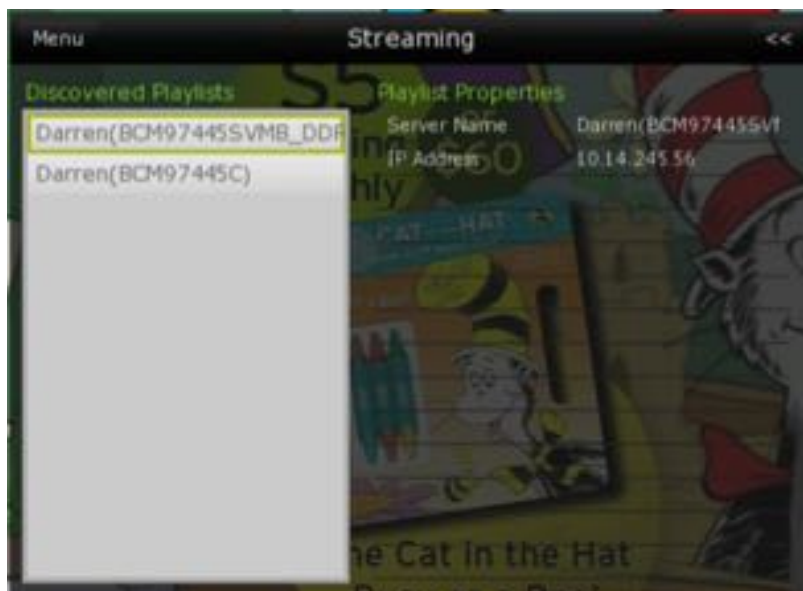

Figure 6 - Streaming Player Window

This screen shows the list of automatically discovered Atlas servers on the network. This includes the loopback server as well (Atlas can stream to itself). Selecting a given playlist will show any associated metadata in the right pane of the window. Note that the server name broadcast to other Atlas instances can be changed in the atlas.cfg configuration file (ATLAS_NAME entry). Clicking on a server will display the list streams available for streaming.

Figure 7 - Streams Window

Selecting an available stream from a server's list will display associated metadata in the right pane. Clicking on a stream will start streaming and playback in the main window. Once streaming begins, the URL will be displayed in the lower left corner of the main window. The channel banner will also indicate: "STREAMING". Changing channel

To stop streaming, channel up/down using the remote or use Lua commands. Note that the stop streaming functionality will be extended in future Atlas version to include the stop button as well as 10key entry of a new channel.

If the server the user is currently streaming or browsing disconnects, the Streaming Player Screen will revert to the list of servers and the disconnected server will be removed. Streaming will eventually stop and Atlas will tune to the last channel.

Auto Discovery Popup

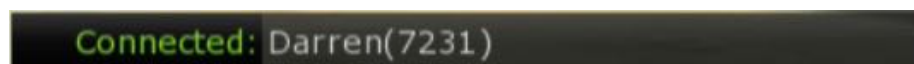When a new remote Atlas is discovered on the network, an onscreen popup will display for 3 seconds.

Figure 8 - Auto Discovery Connect Popup

When a currently discovered Atlas disconnects from the network, an onscreen popup will display for 3 seconds.
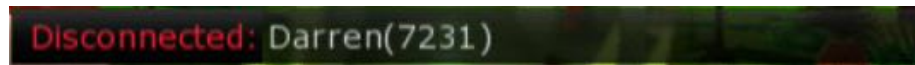

Figure 9 - Auto Discovery Disconnect Popup

If there are multiple connection/disconnection events, they will be displayed in series every 3 seconds. Note that the length of time the Auto Discovery Popup is displayed can be changed in the atlas.cfg configuration file.

Playback Screen

Clicking on the "Playback" button on the Main Screen will show the Playback Screen.
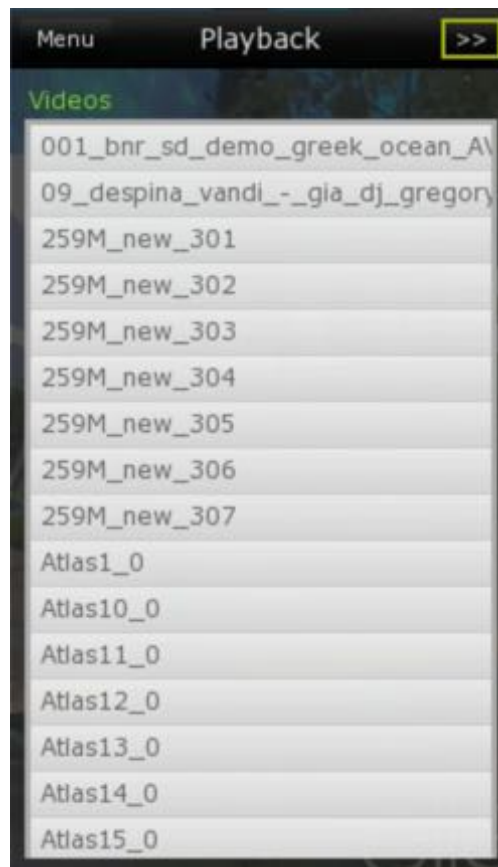

Figure 10 - Playback Window

This screen lists the available playback streams. Selecting a stream will start playback and return focus to the Main Screen. Information in this screen is loaded from the .nfo files that reside in the

videos directory. Playback streams are named as a combination of filename and stream number (<filename>_<stream>).
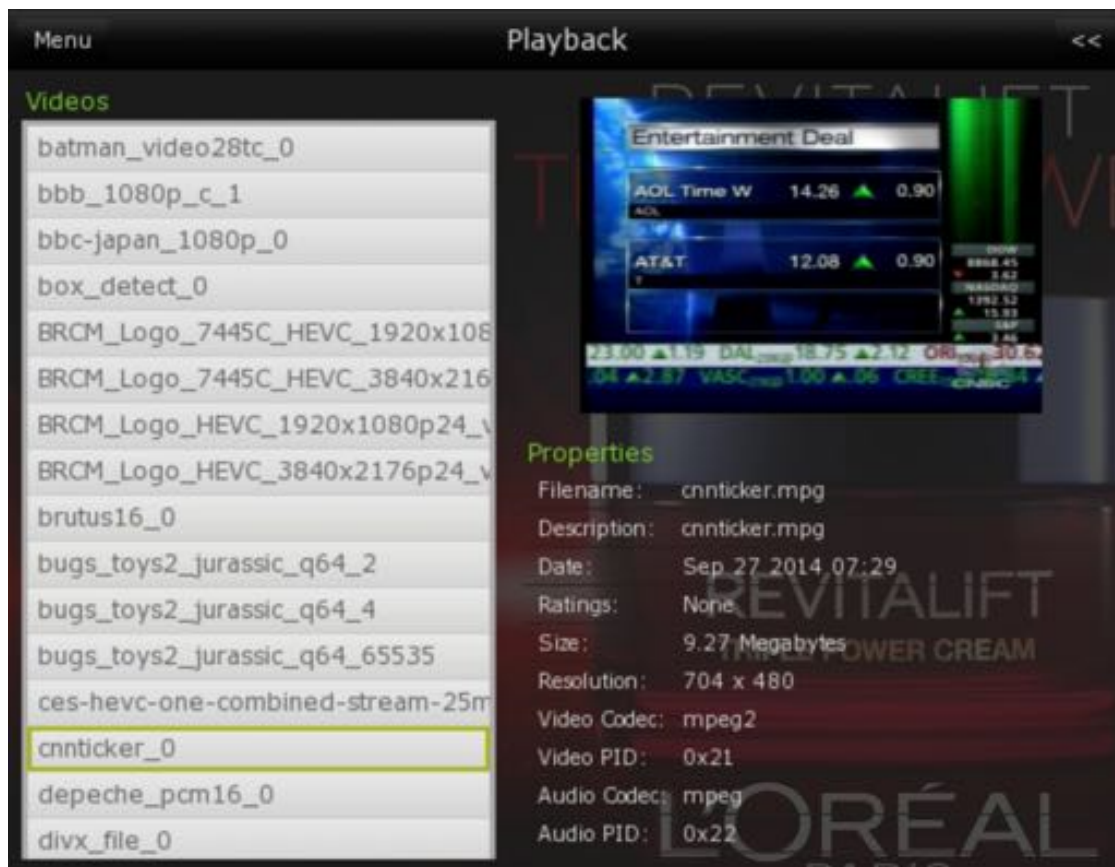
Figure 11 - Expanded Playback Window

Clicking the expand/collapse button in the upper right corner will expand the playback screen to give additional information about the currently selected playback file in the list. Thumbnail images are extracted from the stream each time the highlighted list item is changed (uses Nexus Still Decoder). Properties will update in real time – for example, file size will steadily increase for playback files that are currently recording.

Audio Screen

Clicking on the "Audio" button on the Main Screen will show the Audio Screen.

Figure 12 - Audio Window

The audio screen allows the user to change various audio settings. Each change is applied immediately.

Audio Option	Description
Pid	Choose the audio pid to decode from the list of available pids
PCM	Audio options for PCM outputs: • Stereo – output basic stereo with no post-processing • Auto Vol Level – applies auto volume leveling • Dolby Volume – Dolby Labs volume leveling • TruVolume – DTS/SRS labs volume leveling
Spdif	Audio options for SPDIF outputs: • Pcm – output PCM stereo • Compressed – output source compressed format
Hdmi	Audio options for SPDIF outputs: • Pcm – output PCM stereo • Compressed – output source compressed format

Downmix	Dowmix options vary based on codec. Dolby AC-3/AC-3+: • Auto • Surround - (LtRt) stereo output where surround channels can be decoded using Dolby Pro-Logic. (not suitable for later mono downmixing) • Standard – (LoRo) left and right channel output only DTS: • Auto • Surround - (LtRt) stereo output where surround channels are encoded in the stream. (not suitable for later mono downmixing) • Standard – (LoRo) left and right channel output only AAC/AAC+: • Matrix – stereo output where surround channels are encoded in the stream. (not suitable for later mono downmixing) • ARIB – Japan based downmix standard • LeftRight – (LoRo) left and right channel output only All Others: • None – no downmixing • Left – left channel duplicated on right channel • Right – right channel duplicated on left channel • Mono – mixed left and right output on both channels
Dual Mono	Sets the output for a dual mono stream. This option has no effect for other types of streams.
Compression	Sets the level of Dolby Dynamic Range Compression (DRC). This is only active for AC-3 and AC-3+ audio streams.
Dialog Normalization	Apply Dolby dialog normalization values present in stream. This is only active for AC-3 and AC-3+ audio streams.

Table 7 - Audio Screen Options

Scan Screen

Clicking on the "Scan QAM" button on the Main Screen will show the Scan QAM Screen.
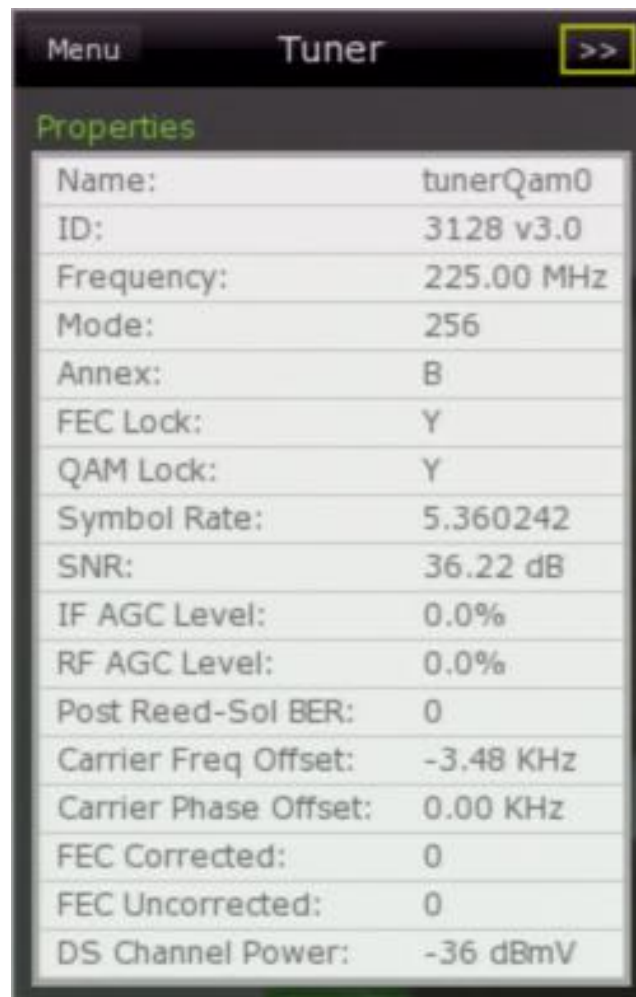
Figure 13 - Scan Window

The scan screen allows the user to scan for unknown channels using the given tuner. The user can do a full replacement scan or selectively append to the channel list with newly found channels. Duplicate, encrypted, and audio only channels are not saved (although this can be overridden. Once the scan completes, the user is prompted to save the channel list to disk. If the new channel list is not saved, you can still tune to the new channels, but they will be lost after quitting Atlas.

Tuner Screen

Clicking on the "Tuner" button on the Main Screen will show the Tuner Screen.

Figure 14 - Tuner Window

The tuner screen shows properties of the tuner used in the main window. These properties are updated based on the UI_GRID_UPDATE_TIMEOUT setting in the atlas.cfg configuration file.
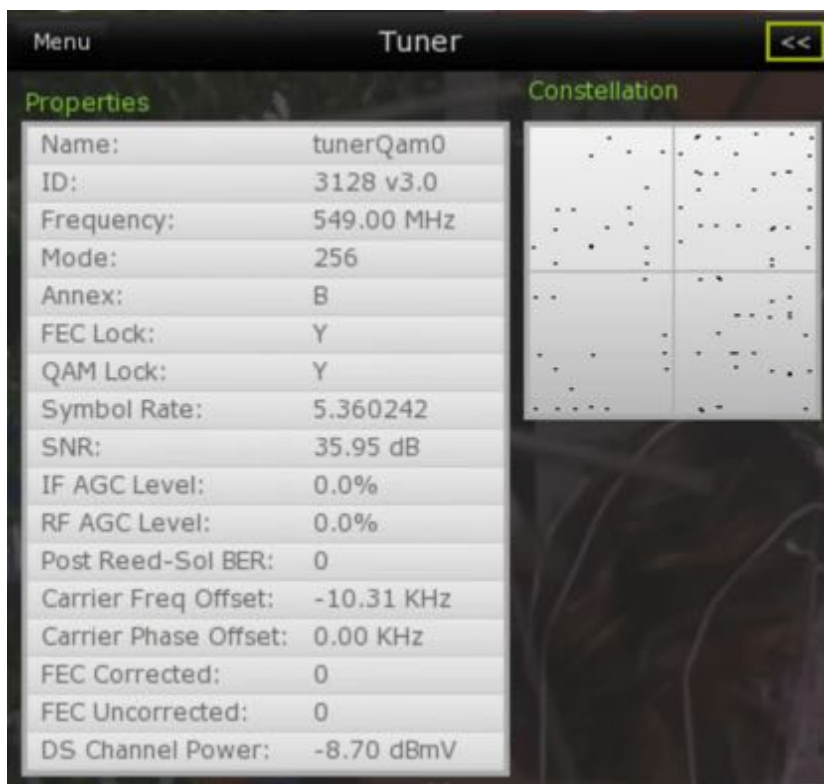
Figure 15 - Expanded Tuner Window

The expanded tuner screen shows the constellation. Dots appear in a grid like pattern when the tuner is locked. The dots will periodically be cleared based on the UI_GRID_UPDATE_TIMEOUT setting in the atlas.cfg configuration file.

Channel Number Screen

The channel number screen is displayed whenever the main menu is present or for several seconds after a channel change occurs. It also shows the current channel type and current recordings that are active.

Buffers/TSM Screen

Clicking on the "Buffers/TSM" button on the Main Screen will show the Buffers/TSM Screen.

Figure 16 - Buffers / TSM Window

When displayed, the buffers screen remains on screen even when the other menu screens are hidden. It is designed to be independent of the other screens once shown. Similar to the channel number screen, the user can navigate to other menus without closing this screen

PVR Timeline Screen

Once a file is playing, a PVR timeline will be displayed on screen showing the max duration of the stream as well as the position of the currently displayed content.
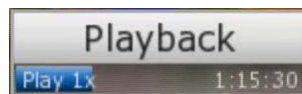
Figure 17 - Timeline Window

Wifi Network Screen

On Wifi capable platforms, there will be a "Network" option in the main menu (you must enable support before building – see Feature Options). Clicking on this button will open the Wifi networks screen. Upon opening, Atlas will begin scanning for available wifi networks and display them by SSID. It will also show their available modes, signal strength, and security status. You can also add a channel column by setting NETWORK_WIFI_LIST_CHANNEL_IGNORE to false in the atlas.cfg file. As long as the network screen is visible, wifi scanning will continue every 5 seconds (see UI_NETWORK_WIFI_UPDATE_TIMEOUT in atlas.cfg).

Click on a network in the scanned networks list to connect to it. If the network requires a password, an onscreen keyboard will be shown. Upon successful connection to a particular wifi network, a blue wifi icon will be displayed next to its entry in the network list. The panel will also expand to show connection status. This expanded screen will auto hide if the network connection is disconnected or can be manually hidden using the "<<" button in the upper right.

Figure 18 - Wifi Network Window

Figure 19 - Network Password Window

The Wifi Protected Setup (WPS) button can be used with WPS compatible access points to create a connection without using SSIDs or passwords. Press the WPS button on the networking screen and Atlas will scan for WPS button presses on nearby access points. Then press and release the WPS button on an access point or wireless router and Atlas will negotiate a protected connection automatically. Atlas will listen for an AP WPS button press for 2 minutes. Negotiating a connection can take up to 30 seconds depending on AP and network conditions.

Figure 20 - Network Wifi Protected Setup (WPS)

Power Mode Screen

If the user presses the remote control power button, Atlas will display the power mode screen. This allows the user to choose the type of standby mode to enter. You must build Atlas with the following exports:

- export POWERSTANDBY_SUPPORT=y

- export NEXUS_POWER_MANAGEMENT=y

Figure 21 - Power Window

[S1] Active Standby: There is no active display, decoding, or playback, but active recordings are allowed to continue.

[S2] Passive Standby: There is no active display, decoding, playback, tuning, or recordings. The system configures the supported wake-up devices that exist in AON and ON-OFF blocks.

[S3] Deep Sleep Standby: There is no active display, decoding, playback, tuning, or recordings. And the entire chip is power-gated except for the AON block.

Pressing the remote power button (or using the atlas.powerMode(0) Lua command) while Atlas is in standby, will power ON and return to the previously tuned channel. Any playbacks or recordings that were auto terminated during the previous transition to OFF, will not be restarted.

Default values for the Linux power options can be set in the atlas.cfg configuration file. These options are listed under POWER_MGT_XXXX (see atlas.cfg for descriptions of each option).

PVR Playback

Playback can be started using either the GUI Playback screen or a Lua command (see Lua Scripting). When playback hits the end of the file, it will automatically loop back and continue playing from the beginning. If playback reaches the beginning (in the case of rewind trick modes), the trick mode will be cancelled and playback will begin normally.

Encrypted Playback

40nm & 28nm Platforms only.

Encrypted Playback requires the platform to have NEXUS SECURITY, HSM. Encrypted Playback will not work if any component is missing. To start an encrypted playback make sure the info file is setup correctly with the encryption Key and Algorithm type embedded in every pid. The CPD/CP inline encryption does encryption per pid.

Sample Info File (cnnticket_3des.nfo)
 # cat /mnt/nfs/cnnticker_3des.nfo
<atlas version="4.0">
  <infofile filename="cnnticker_3des.mpg" indexname="cnnticker_3des.nav" indexrequired="true"
path="/mnt/nfs/" date="1386959543" size="9721856.000000">
    <stream number="0" type="ts" width="704" height="480">
      <pid type="playback" videopid="33" videotype="mpeg2">
     <security enckey="12,34,56,78,9a,bc,de,f0,12,34,fe,ed,ba,be,be,ef" enctype="aes128"></security>
      </pid>
      <pid type="playback" audiopid="34" audiotype="mpeg">

*   <security enckey="12,34,56,78,9a,bc,de,f0,12,34,fe,ed,ba,be,be,ef"*
enctype="aes128"></security>
*   </pid>*
* </stream>*
* </infofile>*
</atlas>

The File will appear in the Playback List. Select the Encrypted Playback File and it should start playing.

PVR Trick Modes

After playback has started, trick modes can be performed using the PVR buttons on the remote. The trick mode buttons will increment or decrement the playback speed. For example, if you press the rewind button twice to playback at the 2nd rewind speed, and then press the fast-forward button once, you will return to the 1st rewind speed. Trick mode speeds are currently printed to the debug console.

Trick Mode	Effect
Fast Forward	NEXUS_NORMAL_DECODE_RATE * FastForward presses
Rewind	NEXUS_NORMAL_DECODE_RATE * Rewind presses
Pause	Pause either normal playback or fast-forward or rewind
Stop	Stop playback and return to live TV
Play	Resume normal playback from Fast-Forward/Rewind/Pause

Table 8 - PVR Trick Modes

PVR Recording

Atlas allows the user to record live TV to disk for later playback. The number of simultaneous recordings is dependent on the number of available tuners and disk write bandwidth.

Start a Recording

1. Tune to the desired channel.

---

2. Press the <record> button on the remote.

3. If successful, the recorded channel will be displayed in the channel number screen in red.

Start Encrypted Record Mode

40nm and 28nm Platforms only.

Platform must support NEXUS Security and HSM to enable encryption. Pressing the record button will bring up a PVR GUI menu that has encryption options. Please select The Encryption option you would like and select the "Ok" button.

You also can record using Lua Command line. Format is
recordStart("filename.mpg","/path/videos","encryption")

You have three options (des,3des,aes) .
Example
#atlas lua>atlas.recordStart("Atlas.mpg","./videos","aes")

This feature is currently for testing encrypted record. Platforms with no encryption support will record all programs in the clear and the Lua command line will not accept the third encryption argument.

Start Multiple Recordings

If your platform supports enough tuners, after recording, you may be able to tune to another channel. This stop decoding and converts the existing record session into a background recording.

4. Tune to a new channel.

5. Press the <record> button on the remote.

6. If successful, the recorded channel will be added to the channel number screen in red. You should now see both recorded channels in red.

If the total number of active recordings equals the number of available tuners, you will only be able to tune to channels that are recording.

Stop Recording(s)

Pressing the <stop> key on the remote will stop the current channel's record session (if tuned to a recording channel).

Tune 1.1 → Record 1.1 → Tune 2.1 → Record 2.1 → Tune 1.1 → Stop (1.1) → Stop (2.1)

If not tuned to a recording channel, the record sessions will be stopped in the same order in which they were started.

Tune 1.1 → Record 1.1 → Tune 2.1 → Record 2.1 → Tune 3.1 → Stop (1.1) → Stop (2.1)

After a recording is stopped, the red record indicator corresponding to that record session is removed from the channel window. This also frees up the record resources allowing them to be reused in another future recording.

Picture-in-Picture (PiP) Mode

Picture-in-Picture mode allows the user to display two content sources at the same time. One source will be displayed full screen, while the other in a smaller window on top. The smaller window can be adjusted both in size and position.
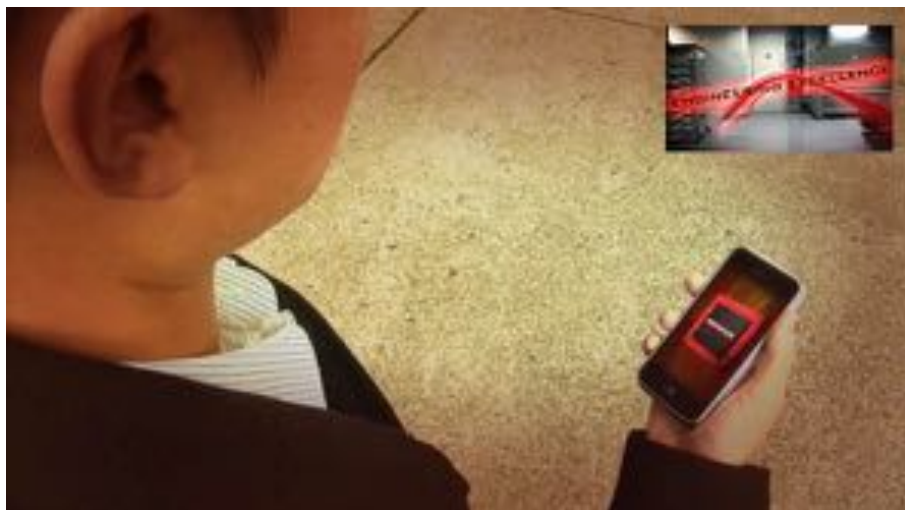
Figure 22 - Picture-in-Picture Example

PiP Show/Hide

To show or hide the PiP window, press the <PiP> button on the remote controller. This will open a smaller window showing content identical to the full screen content. Subsequent channel changes will only affect the full screen window. The PiP window will remain playing its current content.

Atlas will remember the last tuned PiP channel and will tune to that channel if closed and later reopened.

PiP Swap

The <Swap> button will swap the content of the full screen and PiP window. Because only the full screen content can be changed (channel change or playback), altering the PiP source can only be accomplished by first swapping to the full screen, making a source change, and then swapping back.

PiP Size

The size of the PiP window is expressed as a percentage of the full screen window's width and height. In the atlas.cfg configuration file, set PIP_PERCENTAGE to adjust the size of the PiP window. Default PiP window size is 25% - this will set the PiP window to be 1/8th the area of the full screen window.

PiP Window Location

The on screen location of the PiP window is expressed as a number indicating the quadrant of the full screen in which it is positioned (0:upper right, 1:upper left, 2:lower left, 3:lower right). In the atlas.cfg configuration file, set PIP_POSITION to adjust the position of the Pip window.

Programmable Luminance Mapping (PLM)

Programmable Luminance Mapping (PLM) is Broadcom's hardware/software solution for seamless conversion of SDR and HDR video and graphics on a wide variety of SDR and HDR10 televisions. PLM encompasses customizable non-linear transforms for color space conversion, EOTF luminance range conversion, and resolution conversion. PLM is enabled by default but can be defeated in Atlas to illustrate its benefits on screen.

HDR and PLM status will be displayed on screen for each video window. This status will only be shown if the output device is HDR capable. Users can force HDR output over HDMI by setting FORCE_HDMI_HDR_OUTPUT = "true" in atlas.cfg. While this will have an impact on video/graphics quality when connected to a non-HDR television, the advantages of PLM can be readily seen when comparing enabled and disabled output.
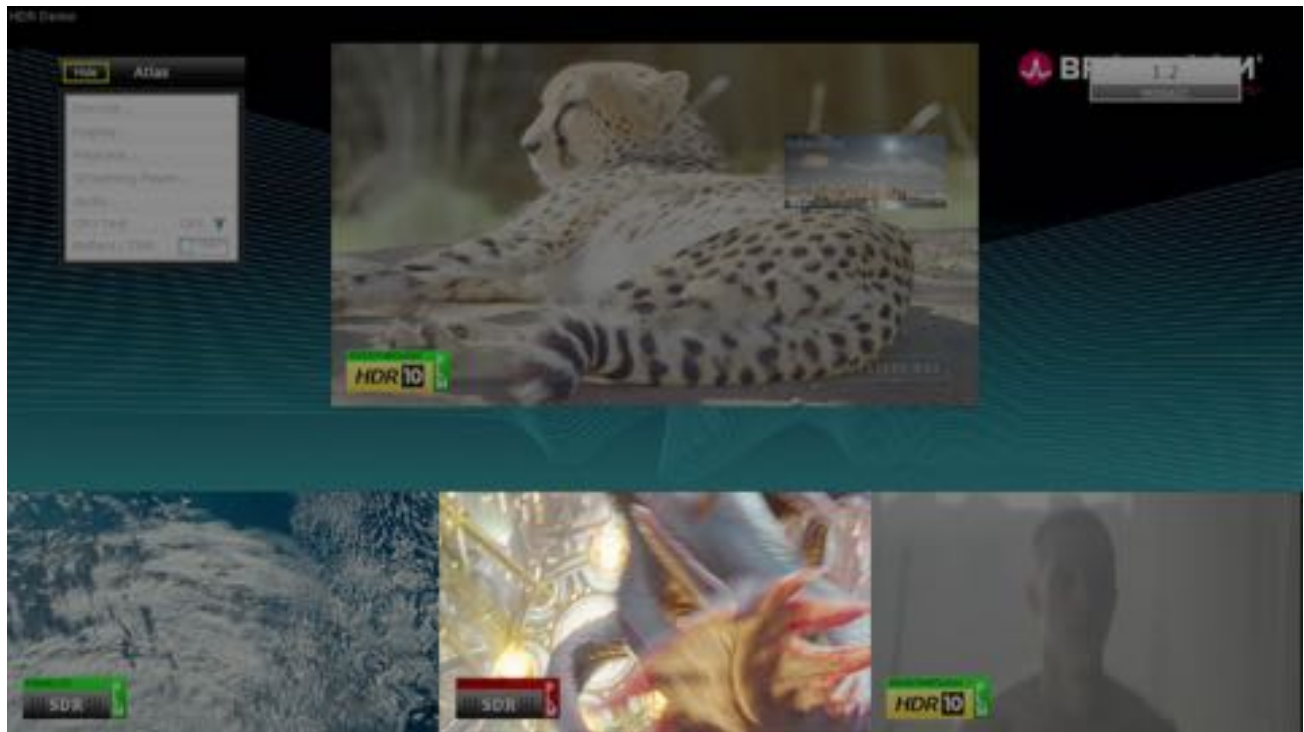
Figure 23 - HDR PLM Mosaic Channel

Mosaic channels can enable/disable PLM for each of its sub channels independently. See the Channel List section for more details.

PLM settings can also be altered dynamically using Lua commands setPlmVideo() and setPlmGraphics(). See the section on Lua Scripting for more details. Note that these commands will only have an effect if the HDMI output is in HDR mode.

Lua Scripting

The Atlas scripting language shall be Lua (pronounced LOO-ah). Lua is a lightweight, fast, simple scripting language originally designed for embedding into larger applications. Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode for a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping. It is written entirely in C and is easy to embed into C/C++ applications. It is currently in use in many major commercial applications such as Adobe Lightroom and World of Warcraft. The Lua library is only 203K and is one of the very fastest interpreted scripting languages. Lua's build times are quite low, and it is distributed under the MIT license which makes it ideal for reference software development and releases. For more information visit: http://www.lua.org.

Shell Prompt

Atlas provides an interactive shell that the user can use to input Lua commands one by one. It provides instant feedback and can be used to not only alter the state of Atlas, but also to retrieve status and debugging information. Lua scripting code can be entered directly as well as any Atlas specific custom commands.

Help

Typing "help" at the Atlas prompt will display the basic features of the interactive shell itself. Typing "help <Atlas command>" will display more detailed information about the given custom command (TBD).

Auto Complete

The Atlas Lua shell also has a built in auto complete feature for the custom commands. The user can type the beginning of any custom command followed by the <tab> key and the shell will fill in a potential command. Repeated <tab> key presses will cycle through other possible auto completed commands. Once all possibilities are cycled through, the next <tab> returns to the original user entry. For example:

1. atlas lua> atlas.ch<ctab> → atlas lua> atlas.channelDown(

2. <tab> again → atlas lua> atlas.channelListDump(

3. <tab> again → atlas lua> atlas.channelListLoad(

4. <tab> again → atlas lua> atlas.channelListSave(

5. <tab> again → atlas lua> atlas.channelTune(

6. <tab> again → atlas lua> atlas.channelUp(

7. <tab> again → atlas lua> atlas.ch

Command History

Atlas Lua commands can require multiple arguments which make retyping them tedious and error prone so the Atlas shell also supports a **persistent** command history. Each command entered is saved in a history list (lua_history.txt) in the Atlas bin directory. To recall a previously entered command, use the <up> arrow and <down> arrow keys. This command history is automatically restored when starting Atlas so the user has access to it even if the platform is rebooted.

The lua_history.txt file which saves the command history, is a simple text file and can be edited directly if desired.

Script File

Atlas commands can be combined with Lua programming language calls in a script text file. This script file can be launched automatically on Atlas start by specifying it in following the "-script" command line option (see Command Line Arguments section).

```
# start atlas –script test.lua
```

Atlas will skip the initial tuning to the first channel in the channel list and instead begin executing commands in the test.lua file. test.lua must reside in the "scripts" directory in the Atlas bin directory.

Atlas can also run a given Lua script file from the Lua shell prompt:

```
atlas lua> atlas.runScript("test.lua")
```

Atlas script handling is not reentrant so a given script cannot use the atlas.runScript() command to run other scripts.

Atlas Lua Commands

Atlas extends the Lua scripting language to include custom commands that can be used in conjunction with standard Lua commands. Optional parameters are listed in [brackets]. All Lua commands (except where noted) will return -1 on error and 0 otherwise.

atlas.bluetoothConnectFromDiscList()

This command attempts to connect to the Bluetooth device indicated by the given index. You can retrieve the list of possible connections and corresponding indexes using atlas.bluetoothDiscoveryStart() and atlas.bluetoothGetDiscoveryListInfo() commands.

1.   Index (int): index indicating discovered Bluetooth device.

atlas.bluetoothDisconnect()

Disconnect from the Bluetooth device corresponding to the given index. Use atlas.BluetoothGetConnectedListInfo() to find the index associated with a given device.

1.   Index (int): index indicating connected Bluetooth device.

atlas.bluetoothDiscoveryStart()

Start discovering Bluetooth capable devices in range.

atlas.bluetoothGetConnectedListInfo()

Get the indexed list of currently connected Bluetooth devices.

atlas.bluetoothGetDiscoveryListInfo()

Get the indexed list of currently discovered Bluetooth devices.

atlas.bluetoothGetSavedListInfo()

Get the indexed list of previously connected Bluetooth devices (cached).

atlas.channelDown()

This command tunes to the previous channel in the currently loaded channel list.

atlas.channelListDump()

This command prints out the current channel list.

atlas.channelListLoad([filename], [listname], [append])

This command loads the given channel list from the given file name and either appends or replaces the current channel list.

1. filename (string): default: "channels.xml"

2. listname (string): default: "default"

3. append (bool): if true:append new channel list channels to existing channel list. Default false:replace existing channel list.

atlas.channelListSave([filename], [listname], [append])

This command saves the current channel list to the given file name and either appends or replaces the current channel list.

1. filename (string): default: "channels.xml"

2. listname (string): default: "default"

3. append (bool): if true:append current channel list channels to file channel list. Default false:replace existing channel list.

atlas.channelStream(url)

This command streams and IP channel from a remote Atlas server and displays it on screen.

1. url (string): url used to specify host, port, file and program number. ("http://xxx.xxx.xxx.xxx:8089/cnnticker.mpg?program=0") Note that if the query string is omitted, Atlas will stream the first program in the given file. You can also specify live channels for streaming in a similar manner ("http://xxx.xxx.xxx.xxx:8089/channel?1.1") but the query string is required in this case.

atlas.channelTune(channelNum, [tunerIndex])

This command tunes to the given channel number in the currently loaded channel list.

1. channelNum (int.int): logical channel number to tune to (this channel must exist in the channels.xml file). If only a major channel number is given (i.e. 4) then Atlas will tune to the first matching channel it finds in the channel list (so if 4.2 is the first channel with major number 4, Atlas will tune to 4.2). If major channel number 0 is given, Atlas will tune to the first channel in the channel list.

2. *tunerIndex (int):* default is the next available tuner. This optional parameter can be used to force Atlas to using a particular tuner index to tune with.

atlas.channelUntune()

This command untunes from the currently tuned channel.

---

atlas.channelUp()

This command tunes to the next channel in the currently loaded channel list.

atlas.closedCaptionEnable(enable)

This command enables/disables Atlas digital closed captioning support.

   1.   enable (bool): true turns on digital closed captioning

atlas.closedCaptionMode(mode)

This command sets the type of digital closed captioning.

   1.   mode (int): 1 for 608 captioning or 2 for 708 captioning

atlas.debug(text)

This command prints the given text both to the debug console and displays it on screen in the lower right hand corner. For more complex strings, use the string.format() command as follows: atlas.debug(string.format("hello world: %d", nNumber))

atlas.encodeStart([filename], [path])

This command starts a transcode of the currently tuned channel and saves it to a file. Filename and path parameters are TBD.

atlas.encodeStop()

This command stops a current transcode.

atlas.getChannelStats()

This command exposes Channel Statistics of the current Channel. If the current Channel supports this feature, then the Channel Statistics will be printed on the console.

atlas.ipClientTranscodeEnable(enable)

This command enables/disables the next ip streaming transcoding support for a given client.

1. enable (bool): true turns on ip streaming transcoding.

atlas.ipClientTranscodeProfile(profile)

This command sets the next ip streaming transcode profile for a given client. Its only get set if the ip streaming transcoding is enabled using the above lua command.

1. profile (int): 1 for 480p30 and 2 for 720p30 transcode profile.

atlas.irRemoteEnable(enable)

This command enables/disables IR remote control handling.

1. enable (bool): true enables IR remote key handling. False disables.

atlas.exit()

This command exits Atlas. (shortcut command: exit)

atlas.playbackListDump()

This command prints out the current playback list.

atlas.playbackListRefresh()

This command reloads the available playback list from disk.

atlas.playbackStart(filename, [indexFilename], [path])

This command starts playback for the given media file.

1. filename (string): filename of media file to play

2. indexFilename (string): filename of the index file

3. path (string): default:"./videos"

atlas.playbackStop([filename])

This command stops playback for the given media file.

1. filename (string): filename of media file playback to stop. You can omit this parameter and the current playback will be stopped.

atlas.playbackTrickMode(trick)

This command sets the trick mode for the currently playing file.

1. trick (string):

    a. "play" – resume normal playback

    b. "pause" – pause playback

    c. "i" – decode only I-frames

    d. "ip" – decode only I & P frames

    e. "all" – decode all frames

    f. "fa" – frame advance

    g. "fr" – frame reverse

    h. "rate(float)" – set trick mode rate (1.0 is normal speed)

    i. "host(type, *modifier*, *slowrate*)" – set host trick mode where type = I, ip, or all. Modifier = 1 for forward, -1 for reverse (I only). Slowrate = decoder repeat rate (1=1x, 2=2x, etc).

    j. "seek(pos)" – jump to position in millisecs

atlas.playlistDiscovery([index])

This command displays the list of discovered playlist from remote servers. It will printout the host IP address followed by a logical name to help identify the STB.

1. index (number): if specified, index will direct atlas to printout the discovery information for the given playlist. Index == 0 is the same as not giving an index and will result in all discovered playlists being printed.

Returns: This command has 3 return parameters: ip, name, and error.

- IP (string): IP address of playlist (index parameter must be given)

- Name (string): Name of the Atlas instance

- Error (int): Error code associated with command. Value < 0 indicate an error has occurred.

TIP: playlistDiscovery(), playlistShow() and channelStream() can be used together to discover servers, search playlist contents, and tune to an IP channel in a playlist. See BSEAV/app/atlas/scripts/test.lua for an example on how to programmatically discover servers, query playlists, and stream content from a remote Atlas.

atlas.playlistShow(host, [index])

This command displays the playlist contents based on the given host IP address (see atlas.playlistDiscovery()). For each playlist entry it will print out the index and stream name.

1. host (string): IP address of host STB

2. index (number): optional index to show corresponding playlist only. Index == 0 is the same as not giving an index and will result in all playlist contents being printed.

Returns: This command has 2 return parameters: url and error.

- url (string): the url corresponding to the given host and index. This url can be used directly for streaming (see atlas.channelStream()). The url value will only be valid if an index parameter is given (i.e. atlas.playlistShow("xxx.xxx.xxx.xxx", 2)).

- Error (int): Error code associated with command. Value < 0 indicate an error has occurred.

atlas.recordStart([filename], [path], [encryption])

This command starts a recording.

1. filename (string): default:"AtlasXX.mpg" Filename to save the new recording to.

2. Path (string): default:"./videos"

3. Encryption (string): "aes","des","3des" and default is "none"

atlas.recordStop([filename])

This command stops a recording.

1. filename (string): default: current recording. Filename associated with recording.

atlas.rf4ceRemoteAdd(name)

Add the RF4CE remote corresponding to the given name.

1. name (string): name of RF4CE remote to add.

atlas.rf4ceRemoteRemove(name)

Remove the RF4CE remote corresponding to the given name.

2. name (string): name of RF4CE remote to remove.

atlas.runScript(filename)

This command executes the given Lua script.

1. Filename (string): file name of Lua script to execute in the "scripts" directory.

*atlas.scanOfdm(startFreq, endFreq, bandwidth, [stepFreq], [*append*], [*mode*])*

This command scans the given frequency range for OFDM channels.

1. startFreq (Hz): frequency to begin scanning

2. endFreq (Hz): frequency to end scanning

3. bandwidth (Hz): channel bandwidth

4. *stepFreq (Hz):* default is bandwidth. This is the used to determine which frequencies to scan between the startFreq and endFreq.

5. *append (bool)*: default true:add newly found channels to channel list, if false:replace existing channel list

6. *mode (int)*:
    if ISDB-T:NEXUS_FrontendOfdmMode_eIsdbt,

if DVB-T:NEXUS_FrontendOfdmMode_eDvbt,
if DVB-T2:NEXUS_FrontendOfdmMode_eDvbt2

7. *symbolRateMin (int):* default symbol rate is based on given mode parameter. For platforms that do not support scanning a range of symbol rates at the same time, should specify the same value for both symbolRateMin and symbolRateMax.

8. *symbolRateMax (int):* default symbol rate is based on the given mode parameter. For platforms that do not support scanning a range of symbol rates at the same time, should specify the same value for both symbolRateMin and symbolRateMax.

atlas.scanQam(startFreq, endFreq, bandwidth, [stepFreq], [append], [mode], [annex], [symbolRateMin], [symbolRateMax])

This command scans the given frequency range for QAM channels.

1. startFreq (Hz): frequency to begin scanning

2. endFreq (Hz): frequency to end scanning

3. bandwidth (Hz): channel bandwidth

4. *stepFreq (Hz):* default is bandwidth. This is the used to determine which frequencies to scan between the startFreq and endFreq.

5. *append (bool)*: default true:add newly found channels to channel list, if false:replace existing channel list

6. *mode (int)*: default if auto supported: NEXUS_FrontendQamMode_eAuto_64_256, otherwise: NEXUS_FronendQamMode_e256

7. *annex* (int): default:2 1:A, 2:B, 4:C

8. *symbolRateMin (baud): minimum symbol rate to scan (note: if advanced scan is not supported for your given tuner, set both minimum and maximum symbol rate to the same value)*

9. *symbolRateMax (baud): maximum symbol rate to scan (note: if advanced scan is not supported for your given tuner, set both minimum and maximum symbol rate to the same value)*

atlas.scanSat(startFreq, endFreq, bandwidth, [stepFreq], [append], [adc])

This command scans the given frequency range for SDS channels.

1. startFreq (Hz): frequency to begin scanning

2. endFreq (Hz): frequency to end scanning

3. bandwidth (Hz): channel bandwidth

4. *stepFreq (Hz):* default is bandwidth. This is the used to determine which frequencies to scan between the startFreq and endFreq.

5. *append (bool)*: default true:add newly found channels to channel list, if false:replace existing channel list

6. *adc (uint)*: default is 0. Determines which ADC to use for scanning.

atlas.scanVsb(startFreq, endFreq, bandwidth, [stepFreq], [append], [mode])

This command scans the given frequency range for VSB channels.

1. startFreq (Hz): frequency to begin scanning

2. endFreq (Hz): frequency to end scanning

3. bandwidth (Hz): channel bandwidth

4. *stepFreq (Hz):* default is bandwidth. This is the used to determine which frequencies to scan between the startFreq and endFreq.

5. *append (bool)*: default true:add newly found channels to channel list, if false:replace existing channel list

6. *mode (int)*: NEXUS_FronendVsbMode_e8

atlas.setAspectRatio(aspect)

This command sets the aspect ratio.

atlas.setAudioProcessing(processing)

This command sets the audio post processing for audio outputs set to PCM. This setting has no effect on audio outputs not set to PCM.

1.   processing (integer):

    a.   0 = None (Stereo)

    b.   1 = Auto Volume Level

    c.   3 = Dolby Volume

    d.   4 = SRS TruVolume

atlas.setAudioProgram(pid)

This command sets the current audio PID.

1.   Pid (integer): audio PID number

atlas.setAutoVideoFormat(auto)

This command toggles the auto video format setting (on channel change).

1.   auto (bool): true or false

atlas.setBoxDetect(boxDetect)

This command toggles Letter/Pillar box detection.

1.   boxDetect (bool): true or false

1.   aspect (int): NEXUS_AspectRatio

atlas.setColorSpace(color)

This command sets the component video color space.

1.   color space (int): NEXUS_ColorSpace

atlas.setContentMode(mode)

This command sets the video window content mode.

1. mode (int):

    a. 0 = zoom

    b. 1 = box

    c. 2 = pan scan

    d. 3 = full

    e. 4 = stretch

    f. 5 = pan scan no A/R

atlas.setCpuTestLevel(level)

This command sets the CPU test level. Atlas will continuously decompress the images/cputest.jpg image test file using one thread per CPU core. Pauses will occur between image decompressions to adjust overall CPU usage levels. Custom adjustments to the 1 – 9 levels can be accomplished using the CPUTEST_MAX_DELAY value in the atlas.cfg

    a. 0 = off

    b. 1 – 9 = a fraction of the CPUTEST_MAX_DELAY to adjust CPU usage

    c. 10 = 100%

atlas.setDebugLevel(module, level)

This command sets the debug output level for a given module. This allows you to change the debug output while Atlas is running. Any changes made will not survive after restarting Atlas.

1. Module (string): module name

2. Level (string):

    a. "trace"

    b. "msg"

    c.   "wrn"

    d.   "err"

    e.   "log"

atlas.setDeinterlacer(deinterlace)

This command toggles Motion Adaptive Deinterlacer (MAD).

1.   deinterlace (bool): true or false

atlas.setDolbyDRC(level)

This command sets the Dolby dynamic range compression level.

1.   level (integer):

    a.   0 = None

    b.   1 = Light

    c.   2 = Medium

    d.   3 = Heavy

atlas.setDolbyDialogNorm(level)

This command sets the Dolby dialog normalization level.

1.   level (bool): on = true, off = false

atlas.setDownmix(downmix)

This command sets the audio downmix.

1.   downmix (integer):

    a.   0 = None

    b.   1 = Left

    c.   2 = Right

d. 3 = Monomix

e. 4 = Matrix

f. 5 = Arib

g. 6 = LeftRight

atlas.setDualMono(dualmono)

This command sets the audio dualmono.

1. dualmono (integer):

    a. 0 = Left

    b. 1 = Right

    c. 2 = Stereo

    d. 3 = Monomix

atlas.setHdmiAudioType(type)

This command sets the HDMI output type.

1. type (integer):

    a. 0 = PCM

    b. 1 = Compressed

    c. 2 = Multichannel

    d. 3 = Encode DTS

atlas.setHdmiInput(input)

This command sets the HDMI input type. Valid options may be limited by both source audio stream and EDID reported capabilities of the receiving device. Unsupported AAC or WMA-Pro streams will be converted to multichannel PCM if the receiver supports it. Unsupported AC-3+ streams will be converted to AC-3 or multichannel PCM if the receiver supports it.

1. input (integer):

    a. 0 = PCM

    b. 1 = Compressed

    c. 3 = Multichannel PCM

    d. 4 = Encode to DTS (for AAC source audio streams only)

atlas.setMpaaDecimation(decimation)

This command toggles MPAA decimation.

1. decimation (bool): true or false

atlas.setMute(mute)

This command sets muting.

1. mute (bool): true mutes audio

atlas.setPlmGraphics(enable)

This command enables/disables Programmable Luma Mapping (PLM) for graphics. The default setting is determined by the channel list channel. Changes made using this API will be retained by the channel until the channel list is reloaded.

1. enable (bool): on = true, off = false

atlas.setPlmVideo(videoWin, enable)

This command enables/disables Programmable Luma Mapping (PLM) for video. PLM can be set independently for each video window. Changes made using this API will be retained by the channel until the channel list is reloaded.

1. videoWin (integer):

    a. 0 = Main Window

    b. 1 = PiP Window

      c.   2 = Mosaic 1 Window

      d.   3 = Mosaic 2 Window

      e.   4 = Mosaic 3 Window

      f.   5 = Mosaic 4 Window

2.   enable (bool): on = true, off = false

atlas.setPowerMode(mode)

This command sets the power mode to turn on/off the Atlas application.

3.   mode (integer):

      a.   0 = On

      b.   1 = S1 Active Standby

      c.   2 = S2 Passive Standby

      d.   3 = S3 Deep Sleep Standby

atlas.setSpdifInput(input)

This command sets the SPDIF input type. AC-3+ streams will be converted to AC-3.

1.   input (integer):

      a.   0 = PCM

      b.   1 = Compressed (for AC-3 or DTS source audio streams only)

      c.   2 = Encode to DTS (for AAC source audio streams only)

      d.   3 = Encode to AC-3 (for AAC source audio streams only)

atlas.setSpdifType(type)

This command sets the s/pdif output type.

1.   type (integer):

      a. 0 = PCM

      b. 1 = Compressed

      c. 2 = Encode DTS

      d. 3 = Encode AC-3

atlas.setVbiAmol(amolType)

This command sets the SD display VBI Nielsen Automated Measurement of Lineups (AMOL) type.

1. amolType (integer):

      a. 0 = Type I

      b. 1 = Type II 1mb

      c. 2 = Type II 2mb

atlas.setVbiCgms(enable)

This command enables/disables the SD display VBI Copy Generation Management System (CGMS) copy protection setting.

1. Enable (bool): true turns on CGMS, false disables it.

atlas.setVbiClosedCaptions(enable)

This command enables/disables the SD display VBI Closed Captions Pass-through setting.

1. Enable (bool): true turns on closed captions pass-through, false disables it.

atlas.setVbiDcs(dcsType)

This command sets the SD display VBI Dwight Cavendish Systems (DCS) copy protection.

1. dcsType (integer):

      a. 0 = Off

      b. 1 = On1

    c.   2 = On2

    d.   3 = On3

atlas.setVbiGemstar(enable)

This command enables/disables the SD display VBI Gemstar copy protection setting.

1. Enable (bool): true turns on Gemstar, false disables it.

atlas.setVbiMacrovision(macrovisionType)

This command sets the SD display VBI Macrovision copy protection type.

1. macrovisionType (integer):

    a.   0 = None

    b.   1 = AGC only

    c.   2 = AGC 2 lines

    d.   3 = AGC 4 lines

    e.   5 = AGC only RGB

    f.   6 = AGC 2 lines RGB

    g.   7 = AGC 4 lines RGB

    h.   8 = Test 1

    i.   9 = Test 2

atlas.setVbiTeletext(enable)

This command enables/disables the SD display VBI Teletext setting.

2. Enable (bool): true turns on Teletext, false disables it.

atlas.setVbiVps(enable)

This command enables/disables the SD display VBI Video Program System (VPS) setting.

3. Enable (bool): true turns on VPS, false disables it.

atlas.setVbiWss(enable)

This command enables/disables the SD display VBI Widescreen Signaling (WSS) setting.

4. Enable (bool): true turns on WSS, false disables it.

atlas.setVideoFormat(format)

This command requests a particular video format.

1. format (int): NEXUS_VideoFormat or (string): see below for available strings:

    "NTSC",

    "NTSC Japan",

    "Pal-M",

    "Pal-N",

    "Pal-Nc",

    "Pal-B",

    "Pal-B1",

    "Pal-D",

    "Pal-D1",

    "Pal-G",

    "Pal",

    "Pal-H",

    "Pal-K",

    "Pal-I",

    "480p",

    "576p",

    "1080i",

    "1080i 50Hz",

    "720p",

    "720p 50Hz",

```
"1080p",

"1080p 50Hz",

"1080p 24Hz",

"1080p 25Hz",

"1080p 30Hz",

"2160p 24Hz",

"2160p 25Hz",

"2160p 30Hz"
```

atlas.setVolume(level)

This command sets the volume level.

1.   Level (int): 0-100

atlas.showPip(show)

This command shows/hides the Picture-in-Picture window.

1.   Show (bool): true shows the Picture-in-Picture window, false hides it.

atlas.sleep(time)

This command sleeps the Lua script engine for the given period of time in milliseconds.

1.   Time (integer): length of time to sleep in milliseconds.

atlas.swapPip()

This command swaps the size and position of the main video window and the Picture-in-Picture window.

atlas.wifiConnect(ssid, password)

This command connects to the WiFi network denoted by the given SSID and password.

1.   ssid (string): SSID of network to connect to.

2. Password (string): password used to connect.

atlas.wifiConnectState()

This command returns a string indicating the current WiFi connect state.

1. "CONNECTING"

2. "CONNECTED"

3. "DISCONNECTING"

4. "DISCONNECTED"

5. "SCANNING"

6. "FAILURE ASSOC REJECT"

7. "FAILURE NETWORK NOT FOUND"

8. "FAILURE SSID TEMP DISABLED"

atlas.wifiDisconnect()

This command disconnects from the currently connected WiFi network.

atlas.wifiScanStart()

This command starts scanning for nearby WiFi networks. It will print out networks as it finds them.

atlas.wifiWps(scan)

This command starts the process of looking for a Wifi Protected Setup (WPS) connection with an AP or wireless router.

1. scan (bool): true starts WPS scan. False cancels WPS scan. Scan will run for at most 2 minutes.

help

This command prints basic help information.

Sample Lua Scripts

Lua scripts can be executed both at the command line when running Atlas, and also interactively from the Lua shell prompt.

Continuous Channel Up

```
for i = 1,20 do
   atlas.channelUp()
   atlas.sleep(5000)  -- give time for video to display on screen
end
```

Play All Videos in Directory

```
for fname in dir("./videos") do
   if ((nil ~= string.find(fname, ".mpg")) or
       (nil ~= string.find(fname, ".mp4")) or
       (nil ~= string.find(fname, ".ts"))) then
      print("LUA> Now Playing: " .. fname)
      atlas.playbackStart(fname)
      atlas.sleep(5000)
      atlas.playbackStop(fname)
   end
end
```

Show PiP and Swap

```
atlas.showPip(true)
atlas.sleep(3000)
atlas.channelUp()
atlas.sleep(3000)
atlas.swapPip()
atlas.sleep(1000)
atlas.swapPip()
atlas.sleep(1000)
atlas.swapPip()
atlas.sleep(1000)
atlas.showPip(false)
```

Lua References

"The Programming Language Lua", http://www.lua.org, Official Lua Website

"Lua 5.1 Reference Manual", http://www.lua.org/manual/5.1, Official Lua Reference Manual

"lua-users wiki: Lua Tutorial", http://lua-users.org/wiki/Lua/Tutorial, Lua Tutorial

---

"lua-users", http://lua-users.org, lua-users

"Programming in Lua, Second Edition", http://www.inf.puc-rio.br/~roberto/pil2, Programming in Lua, Second Edition

Advanced Options

Exports

To facilitate debugging, Atlas and Nexus sometimes use environment variables to adjust internal state.

- For user mode, export the variables as shown below.

- For kernel mode, export config="<variable>=<value>".

export msg_modules=XXXX,XXXX

Set the BDBG interface modules which should run at MSG level. Grep the src code for BDBG_MODULE() statements to find the module names. For finer grained run time control of debug output, see Lua command atlas.setDebugLevel(module, level).

export sync_disabled=yes

Disable the bsync module. TSM will still be enabled, but compositor sync and precision lipsync will not be active.

export force_vsync=yes

Do not enable TSM (time stamp managed) mode for audio or video decode. "No TSM" is also called vsync mode (even when applied to audio which has no vsync per se).

export not_realtime_isr=yes

By default, ISR processing thread in user mode runes at the highest priority. This option makes it run at normal priority which maximizes our chances of catching isr race conditions. This should be used for testing only.

export no_watchdog=yes

Disable audio and video decoder watchdog processing. The only reason to disable it is to catch decoder bugs during development. It's also useful on the BCM74xx if you are using the ARC prompt, which halts the decoder and generates watchdogs.

export no_brcm_trick_modes=yes

If set, only use host trick modes.

export jail_avd=yes

Use memory controller's address range checkers to see if any AVD (BCM740x video decoder) client is accessing memory outside of its allocations.

export jail_xpt=yes

Use memory controller's address range checkers to see if any XPT client is accessing memory outside of its allocations.

export avoffset=OFFSET

Apply an AV offset (a.k.a. PTS offset) to both audio and video decode. OFFSET is a decimal value in PTS units (45 KHz for MPEG-2 TS, 27 MHz for DSS).

export cont_count_ignore=yes

Do not enable transport continuity counter checking in XPT or RAVE. All packets will reach the decoders.

export hdmi_bypass_edid=yes

HDMI will not read or parse the EDID.