Using the Record, Audio, and Video Interface Engine

STB_RAVE-AN100-R

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

07/15/08

REVISION HISTORY

Revision	*Date*	*Change Description*
STB_RAVE-AN100-R	07/15/08	Initial release.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

TABLE OF CONTENTS

Introduction 1
Using RAVE for Decode 1
Synchronizing Calls Between RAVE and Decoders 1
Start/Stop Decode 2
Decoder Flush 2
Watchdog 3
Starting Playback Before Decode 3
CDB and ITB Endianness 3
Software 3
Using RAVE to Record 4
CDB and ITB Endianness 4
Allocating Correctly Sized Buffers 5
Setting the Correct Wraparound Threshold 6
Handling More Than One Packet Format 6
Using a Subset of the CDB to Fit the Atom Size 6
Use M2M DMA to Construct Correctly Sized Buffers before Writing to Disk 6
SCT Format 7

LIST OF FIGURES

Figure 1: RAVE Decode Paths 1
Figure 2: Software RAVE Mechanism 4

LIST OF TABLES

Table 1: Transport Packet Sizes..... 5
Table 2: Prime Factors Per Format 5
Table 3: Atom Sizes Per Format..... 5

INTRODUCTION

This application note refers to the use of the record, audio, and video interface engine (RAVE) in the BCM7401, BCM7400, BCM7403, BCM7405, BCM7118, BCM7325, BCM7335, and BCM7452 chipsets.

The MPEG-2/DIRECTV® Digital Video Broadcasting (DVB)-compliant transport stream/packet elementary stream (PES) parser and demultiplexer is capable of simultaneously processing 256 packet identifiers (PID) via 128 PID channels in up to five independent external transport stream inputs and two internal playback channels. All 128 PID channels can be used by the RAVE, program clock reference (PCR) processing, message filter, and for output via the high-speed transport or remux module.

USING RAVE FOR DECODE

In previous Broadcom® chips (e.g., BCM7030, BCM7038), the transport connected to audio and video decoders using PID channels. This meant that the data decoders received transport to be parsed and indexed. In order to accomplish this, an internal transport engine was required by each decoder.

With RAVE, the internal transport engines are removed and controlled externally.

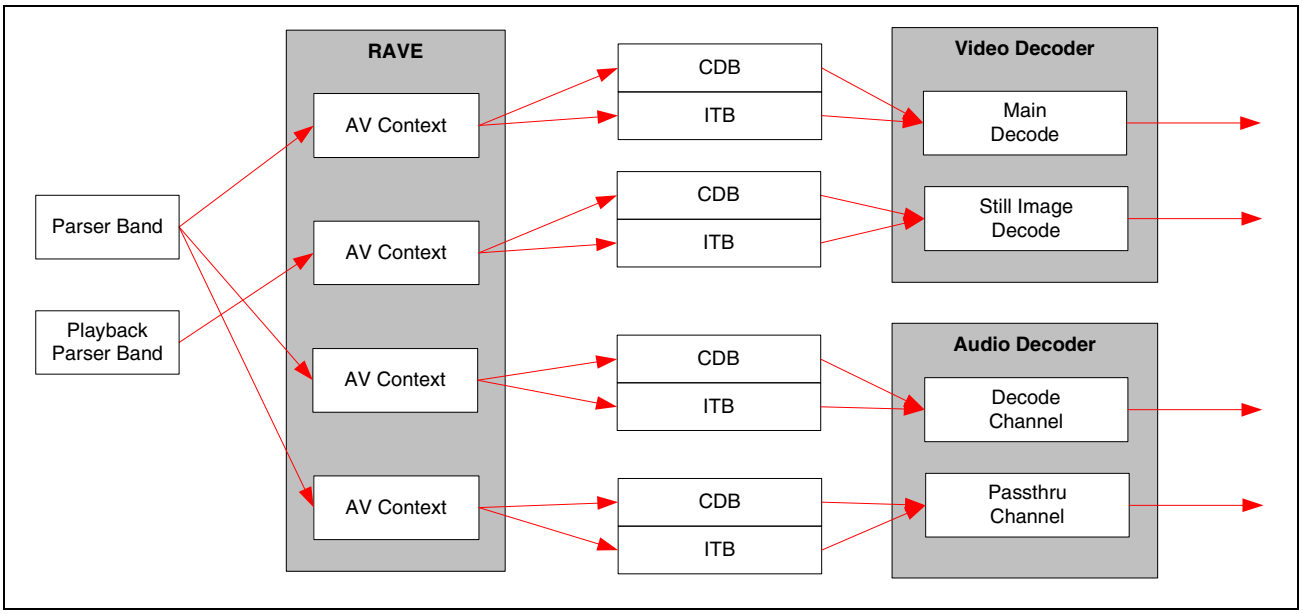

Figure 1: RAVE Decode Paths

SYNCHRONIZING CALLS BETWEEN RAVE AND DECODERS

Calls between the RAVE and the decoders must occur in correct order. In general, RAVE is responsible for setting the VALID pointer (the producer) and the decoders are responsible for moving the READ pointer (the consumer). Only RAVE can flush the context.

There is RAVE firmware and decoder firmware. The RAVE firmware manipulates the producer (valid ptr) of the RAVE compressed data buffer (CDB)/index table buffer (ITB) ring buffer. The decoder firmware manipulates the consumer (read ptr) of the same RAVE CDB/ITB ring buffer. Synchronization at state change is important.

Start/Stop Decode

Start/stop applies to both personal video recording (PVR) and channel changes. The initial state of a RAVE context is both disabled and flushed and has no PID channel connected.

When starting decode, you must:

1. Connect the PID channel to the RAVE context (BXPT_Rave_AddPidChannel).
2. Configure the context (BXPT_Rave_SetAvConfig).
3. Pass the context information to the decoder (BXPT_Rave_GetContextRegisters).
4. Start the decode (BXVD_StartDecode).
5. Enable the context (BXPT_Rave_EnableContext).

When stopping decode, you must:

1. Stop the decode (BXVD_StopDecode).
2. Disable the context (BXPT_Rave_DisableContext).
3. Flush the context (BXPT_Rave_FlushContext).
4. Disconnect the PID channel from the RAVE context (BXPT_Rave_RemovePidChannel).

This takes you back to the initial state.

Note: The examples below apply to the video and audio decoder.

It is very important that context is flushed only after it is disabled and before setting up the next decode. If playback is started and the context is flushed. even if playback is paused and/or the context is disabled, a small amount of data may be lost.

Note: The same ordering applies to audio decode (RAP), but video (XVD) is used in the examples that follow.

Decoder Flush

This is used in stream wrap around or trick mode transitions. To flush the decoder:

1. Disable the RAVE context (BXPT_Rave_DisableContext).
2. Disable the decoder (BXVD_DisableForFlush).
3. Flush the RAVE context (BXPT_Rave_FlushContext).
4. Flush the decoder (BXVD_FlushDecode). BXVD_FlushDecode also enables the decoder.
5. Enable the RAVE context (BXPT_Rave_EnableContext).

Steps 1 and 2 can occur in any order. Steps 3 and 4 can occur in any order. Step 5 must occur in order.

Watchdog

1. The decoder must reset when a watchdog interrupt fires. This is handled automatically inside the X Video Decoder (XVD) ISR Process watchdog.
2. Disable each RAVE context for the decoder (BXPT_Rave_DisableContext).
3. Flush each RAVE context (BXPT_Rave_FlushContext).
4. Call watchdog processing (BXVD_ProcessWatchdog).
5. Enable each RAVE context (BXPT_Rave_EnableContext)

This must be done on separate thread from Interrupt Service Routine (ISR).

Starting Playback Before Decode

As a general rule for set tops, decode should be started before playback. This ensures that no data is lost before decode is able to read its first byte.

This is also required for RAVE systems. The RAVE context has a band hold feature, which holds up playback if the RAVE context is full. If RAVE is disabled, the band hold does not apply and the set-top API simply pauses playback (BXPT_Playback_Pause) until the first consumer starts either decode or record.

CDB and ITB Endianness

The endianness of the RAVE CDB and ITB is configurable. The CDB defaults to the system endianness and ITB defaults to big endian. The decoders are also configurable, however, it is not necessary for the system programmer to know which setting is required. For both XVD and RAP, the BXXX_GetBufferConfig call populates the CDB/ITB configuration to be passed on to RAVE. The XVD and RAP PI's will automatically set the required endianness.

Note: The BXVD_GetBufferConfig should only be used to fetch the buffer alignment. The remaining buffer sizes are for legacy compatibility and have been deprecated.

For CDB and ITB Endianness for record operations, which must be programmed, see "CDB and ITB Endianness" on page 4.

Software

In order to support VC1 Simple/Main profiles, the ITB data produced by RAVE must be modified before it is consumed by the video decoder. Two RAVE contexts are required. The module in the XPT set-top API that performs this function is called Software RAVE (sw_rave). The code is found in bxpt_rave.c.

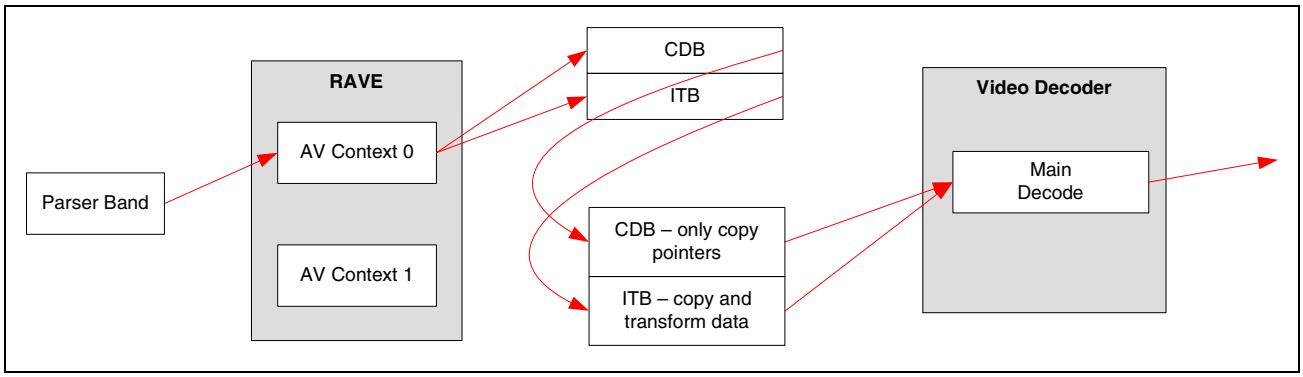

Figure 2: Software RAVE Mechanism

The software RAVE modules works as follows:

1. The set-top API configures both RAVE context 0 and 1 for video decode. Context 0 will be connected to the XPT PID channel. Context 1 will be connected to the video decoder.

   The connection from a pid_channel to a RAVE context is done through XPT registers. There is a data flow through XPT hardware.

   The connection from a RAVE context to a decoder is done through software. The XVD Porting Interface (PI) is told which context to use and the AVD firmware reads the RAVE registers directly. The data flow is only from AVD reading data from memory using the addresses in the RAVE registers.
2. Data is fed into playback and then into context 0.
3. The Software RAVE engine reads the data from context 0 using a combination of XPT playback interrupts and a low-frequency timer. The Software RAVE engine updates the READ pointer of context 0.
4. The CDB pointers from context 0 are copied into context 1. The CDB is not copied.

   The XPT PI always allocates a buffer for each RAVE context. The CDB allocated for context 1 is not used, so we allocate a buffer of just a few bytes which are unused.
5. The Software RAVE engine reads ITB from context 0 and modifies as needed. It copies the modified ITB data into context 1.
6. When the Software RAVE engine updates the VALID pointer in context 1, the video decoder can consume it.

USING RAVE TO RECORD

RAVE is also used for record operations. The CDB buffer serves as the data buffer. The ITB buffer serves as the optional start code table (i.e., index) buffer.

CDB AND ITB ENDIANNESS

The endianness of the RAVE CDB and ITB is configurable. The CDB defaults to the system endianness and ITB defaults to big endian. Broadcom recommends that data files are recorded in big-endian format and that index files are recorded in host endianness (i.e., little endian on little-endian systems, big endian on big-endian systems.) In order to achieve this, the CDB must be set to host endianness and ITB to big endian.

Although this is counter-intuitive, when reading from either direct memory access (DMA) or central processing unit (CPU) memory, the memory controller will swap on a little endian system. So, on a little-endian system, if the CDB is set as little-endian and ITB to big-endian, the end result will be CDB big endian, ITB little endian. On a big-endian system there is no memory controller swap, therefore, the end result will be CDB big endian, ITB big endian. In both cases, the end result is the desired CDB big-endian, ITB host-endian.

For an example of how RAVE is programmed, refer to bsettop_recpump_rave.c.

ALLOCATING CORRECTLY SIZED BUFFERS

When recording using RAVE-based platforms (BCM7401, BCM7400, BCM7118, etc.), special consideration is required when selecting record buffer sizes.

RAVE always writes whole packets to SDRAM. If the application is using Direct I/O to write to disk, it must always write in whole pages, therefore, the record buffer must always be a multiple of both the packet size and the disk page size.

The transport packet sizes that exist are shown in Table 1.

Table 1: Transport Packet Sizes

Format	*Size*
DIRECTV Transport	130
DIRECTV Transport with Timestamps	134
MPEG-2 Transport	188
MPEG-2 Transport with Timestamps	192

The prime factors of these packet sizes are shown in Table 2.

Table 2: Prime Factors Per Format

Format	*Prime Factors*
DIRECTV Transport	13, 5, 2
DIRECTV Transport with Timestamps	67, 2
MPEG-2 Transport	47, 4
MPEG-2 Transport with Timestamps	2, 3

Assuming the operating system page size is 4K, the application must allocate record buffers that are multiples of the atom sizes. Refer to Table 3 below for the atom sizes per format.

Table 3: Atom Sizes Per Format

Format	*Atom Size*
DIRECTV Transport	13 * 5 * 4096 = 266240
DIRECTV Transport with Timestamps	67 * 4096 = 274432
MPEG-2 Transport	47 * 4096 = 192512
MPEG-2 Transport with Timestamps	3 * 4096 = 12288

SETTING THE CORRECT WRAPAROUND THRESHOLD

RAVE record requires a wraparound threshold value. When RAVE writes a whole packet into this area, then a wraparound condition is triggered.

RAVE has one threshold value, which is shared by both CDB and ITB. ITB entries are 24 bytes and can be written in blocks of 10, therefore, this value must be at least 240. Current code requires the RAVE threshold value to be >240 and not >= 240. The reference software currently uses a wraparound threshold of 256.

In order to hide the wraparound requirement and calculation from more generic upper layers of software, the set-top API adjusts the CDB size for RAVE-based platforms so that a whole transport packet, which is also 4K aligned, is guaranteed to trigger this wraparound condition.

The reference code for each supported device includes code to manage the wraparound condition.

HANDLING MORE THAN ONE PACKET FORMAT

If an application must support more than one format, additional considerations must be made.

Using a Subset of the CDB to Fit the Atom Size

The approach that the reference software currently uses is to allocate a single buffer, then truncate it to meet the needs of all packet sizes. In order to do the math, an atom size is required for each packet format. See Table 3 on page 5 for details of atom sizes for the DIRECTV and MPEG-2 formats.

For MPEG-2 transport format, assuming a 2-MB buffer size is required, you could allocate:

192512 * 10 = 1925120 = 1.9M

Switching to DIRECTV, a buffer size that is divisible by 4096 and 130 and less than 1925120 is required. The number is:

226240 * 8 = 1809920 = 1.7M

When running in DIRECTV, this means that 115200 bytes will be unused.

Use M2M DMA to Construct Correctly Sized Buffers before Writing to Disk

On RAVE systems, PVR encryption is performed with Memory-to-Memory (M2M) Direct Memory Access (DMA) after the data has been written to the RAVE CDB. This DMA steps requires an additional buffer, therefore, the destination of the M2M DMA could be 4K page-aligned.

This technique can also be used for PES record, which may not have fixed packet boundaries.

SCT FORMAT

The start code table (SCT) format generated by RAVE differs slightly from the SCT format generated by the BCM7038 and previous chips. In summary, the RAVE SCT format differs from the BCM7038 format in the following ways:

- It is always a six-word format (24 bytes).
- The PTS is in a different location.
- In the current RAVE firmware, only seven bytes of payload can be captured after the start code. This will be fixed in future revisions of the hardware or firmware.

Refer to BSEAV/lib/bcmplayer/src/bcmindexer.c for #if B_HAS_RAVE code.

More information can be found in the BCM7401 Programmer's Register Reference Guides (7401-PRXXX-R) under "XPT_RAVE". Follow the additional data structure link, then search for "Start Code Index Entry" and find entry 0–5.

Broadcom Corporation

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