VS1063A PATCHES

"VLSI Solution Audio Decoder/Encoder"

Project Code: VS1063
Project Name: Support

All information in this document is provided as-is without warranty. Features are subject to change without notice.

Revision History

Rev.	Date	Author	Description
1.46	2013-01-16	POj	mp4 parsing problem fixed.
1.45	2012-11-23	POj	Really fixes the 8000 Hz playing at 2000 Hz problem.
1.42	2012-10-18	POj	Encoding/codec settle delay reduced to 0.0625s.
1.41	2012-10-03	POj	Fixes to codec mode with UART (now all formats).
1.4	2012-08-23	POj	AAC: ADTS supports short frame length (parametric_x.config1 bit 9).
1.34	2012-08-17	POj	ADTS decoding left error variable uncleared. Caused audio to be skipped if data stream element was first in the frame.
1.33	2012-06-28	POj	Version for encoding patches only (minimal size). Fixed subsonic filter initialization for 16k and 32k rates in codec mode.
1.32	2012-06-07	POj	Application hook at 0x60 to disable zero-sample insertion. Fix to audioBufFill bug introduced in 1.3 (did get negative sometime). Two new ADC modes for codec/encoding. FLAC decoding fixes.
1.31	2012-03-02	POj	Extra stub functions prevent AEC crashing.
1.3	2012-01-25	POj	Interrupt handling changed, clock-adder always used to allow higher clocks, 16kHz and 32kHz supported in codec mode.
1.2	2011-11-22	POj	Exact 44.1kHz encoding rate, Mp4 fix, FLAC uses DO_NOT_JUMP, encoder/codec mode uses faster handling for HDAT0 and HDAT1. ADMixer fixed.
1.11	2011-11-03	POj	Headerless codec mode linear pcm defaults to 16 bits.
1.10	2011-10-31	POj	Fix for mp3 encoding of the highest band. Headerless codec mode playback rate fixed. ADC channel swap problem fixed for encoding mode.

Contents

List of Figures

1 Description

There are some known problems in the VS1063a firmware. This patch fixes the issues.

- MP3 Encoder: a wrong Huffman code is generated from one of the tables.
- SetRate: sample-exact samplerate change can go wrong with small probability.
- Encoders: if SCI_VOL is 0, monitoring volume can stay set to mute.
- MP3 Encoder: quantizer of the highest band could get corrupted.
- Codec mode: when not using RIFF WAV header, wrong playback samplerate is used.
- Encoders: if ADC has already been on since last hardware reset, channels can get swapped on ADC activation.
- Codec mode: when not using RIFF WAV header, linear PCM defaults to 0 bits per sample, i.e. no sound produced.
- AAC/MP4: StreamDiscard() missing in header processing.
- FLAC decoding does not set/clear DO_NOT_JUMP during headers.
- InitHardware initializes wrong curFctl/newFctl (2000 Hz).
- Encoders: 44.1kHz rate is not exact, it is actually 44.2kHz.
- Encoders/codec: HDAT0 and HDAT1 interrupt handler takes a lot of cycles.
- ADMixer: configuration from wrong variable, channel-swap
- Max 5.0x clock is possible, encoding combinations may sometimes need more.
- Codec: 16 kHz and 32 kHz rates are not supported.
- During high interrupt loads some interrupts may be missed. This problem mostly occurs with the 44.1 kHz resampler.
- Codec: using AEC crashes the codec mode after a few seconds.
- FLAC: handling of wasted bit does not compensate the level.
- FLAC: Stream buffer management error at FLAC decode startup.
- AAC: ADTS loop did not clear error indicator.
- AAC: ADTS now partly supports short frame length (parametric_x.config1 bit 9).
- AAC/MP4: some atoms were thought to have subatoms when they did not.

When the patch is loaded, it starts automatically (writes 0x0050 to AIADDR) and restarts the system (you should wait for DREQ to rise). The patch must be re-loaded after each hardware or software reset. If you replace software reset by writing 0x0050 to AIADDR, you do not need to reload the patch (except after encoding mode, which always needs to end with a standard software reset). Normally software reset is not needed when you use the cancel mechanism to change songs.

To start encoding mode, first set the encoding parameters, then set the SM_ENCODE bit in the SCI_MODE register (without software reset), then write 0x50 to SCI_AIADDR.

There are two versions. Normally you would use the vs1063a-patches. If you only use encoder and don't have enough memory in your controller for the vs1063a-patches, you can use the stripped-down version vs1063a-encpatches, which only contains fixes for several encoding-related issues.

Chip	File	IRAM	Description
VS1063A	vs1063a-patches.plg	0x50..837	Full, compressed plugin
VS1063A	vs1063a-patches.c	0x50..837	Full, old array format
VS1063A	vs1063a-patches.cmd	0x50..837	Full, legacy cmd format
VS1063A	vs1063a-encpatches.plg	0x50..264	Encoding only, compressed plugin
VS1063A	vs1063a-encpatches.c	0x50..264	Encoding only, array format
VS1063A	vs1063a-encpatches.cmd	0x50..264	Encoding only, legacy cmd format

Note that some X and Y data memory is also used (X:0x1800..0x1841, X:0x1c62..0x1ce7, Y:0x1800..0x1810).

This patch uses the application address to start automatically (the last entry in the patch tables writes to SCI_AIADDR), but does not use it afterwards.

Both old loading tables and the new compressed plugin format is available. The new plugin format is recommended, because it saves data space and future plugins, patches, and application will be using the new format.

Note: after using encoding mode, use a hardware reset or ADMixer may have its channels swapped.

Unless otherwise specified, this patch is **not** compatible with other vs1063a plugins (although at the time of this writing there are none). You need to give a software reset and load the right code when you switch between different plugins.

1.1 Additional Information

If you find a bug, a curious feature, or are unsure how to use VS1063, let us know.

VSDSP Forum is at **http://www.vsdsp-forum.com/** .

Support e-mail address is **support@vlsi.fi** .

1.2 Fixes in Detail

MP3 Encoder: Audio quality - Wrong Huffman code

A missing bit mask in the Huffman encoding function causes a wrong code to be generated from one of the quantization tables. The problem manifests itself as occasional high-pitched sounds in the playback of the encoded file. The severity and frequency of the problem changes depending on the selected encoding parameters and the audio being encoded.

This patch adds the correct masking operation, restoring the intended audio quality for all audio sources.

SetRate: Incorrect Samplerate

The samplerate nybble in FREQCTLH may sometimes be set wrong, causing incorrect playback rate. **However, we have not been able to trigger the problem in normal slave decoding mode.**

VS1063A provides a sample-exact samplerate change so you can for example mix mp3 files with any samplerate one after another and get the correct-sounding playback. If you turn off resyncs, you can also play short wav files back-to-back without giving cancel between them.

Also volume control is synchronized to output samples to work optimally with bass/treble controls.

Due to missing Disable/Enable in one branch of SetRate function, it can override interrupt routine's change of the FREQCTLH register. The situation occurs when audio buffer is empty when SetRate is called and the clock multiplier does not change. This can happen mainly during the start of mp4 decoding, but we have only been able to trigger the problem in the standalone player/recorder, not in slave decoding mode.

This patch adds the Disable/Enable to the critical section of code.

Workaround without the patch: volume change also updates the FREQCTL value.

Encoders: if SCI_VOL is 0, monitoring volume can stay set to mute.

The encoding mode mutes monitoring through DAC for the first second to give time for the analog circuits and digital filters time to settle before encoding. The normal playback volume set by SCI_VOL should then be restored. Especially with 48 kHz samplerate this does not seem to always happen when SCI_VOL is set to 0.

This patch increases SCI_VOL to 0x0101 in encoding modes, if it sees 0 in the register.

Workaround without the patch: change SCI_VOL after encoding has started.

MP3 Encoder: Quantizer Selector Overwritten

The quantizer selection information of the highest used band could get overwritten, and thus audio between approximately 13 kHz and 16 kHz could get intermittently suppressed, i.e. not encoded.

This patch avoids the overwrite, allowing material containing upto 16 kHz signals to be encoded correctly.

Codec Mode: Wrong Playback Rate when no RIFF Header

The encoding mode sets the DAC to the same rate than the ADC is run at, i.e. 12 kHz, 24 kHz, or 48 kHz.

When codec mode uses RIFF WAV header, the playback rate will be set according to the WAV sample rate field. When RIFF WAV header is not used, the samplerate is not set. This is wrong, the playback should be the same rate as the encoding rate.

This patch fixes the problem and codec mode now sets the playback rate to equal the encoding rate when RIFF WAV header is not in use.

Encoding Mode: Swapped Channels

When ADC has been active and is deactivated then reactivated without a hardware reset in-between, left and right channels can get swapped. If ADMixer is used, deactivated and used later with the same rate, the problem does not appear.

Channels can be swapped in these cases.

- If encoding after using ADMixer without hardware reset
- If encoding after a previous encoding without hardware reset
- If using ADMixer after encoding without hardware reset
- If using ADMixer with two different rates without hardware reset

This patch fixes the problem for the encoding mode. You can start encoding multiple times without a hardware reset in-between. Now this patch also corrects ADMixer.

Codec mode: No Sound for LPCM

When not using RIFF WAV header (headerless mode), the sample size defaults to 0 bits. This causes linear PCM to produce no sound.

With the patch the sample size is 16 bits for linear PCM mode in headerless mode.

AAC/MP4: StreamDiscard() Missing

A missing StreamDiscard() in header processing can with some files cause the “mdat” chunk name to be missing a byte, thus AAC decoding will not start. This patch fixes the problem.

DO_NOT_JUMP not used with FLAC

FLAC decoding does not set/clear DO_NOT_JUMP during headers. So, if you seek in the file during header, the header size fields can be corrupted and header processing does not end.

This patch adds DO_NOT_JUMP handling to FLAC decoding.

InitHardware Initializes to 2 kHz

InitHardware initializes wrong curFctl/newFctl value (2000 Hz), but sets hwSampleRate to 8000 Hz. Calling InitHardware twice and then playing 8000 Hz audio results playback with wrong samplerate (2000 Hz instead of 8000 Hz).

This patch fixes the problem.

Encoders/codec: HDAT0 and HDAT1

Reading 16-bit data through HDAT0 takes about 100 cycles from the interrupt handler, and HDAT1 requires some processing as well. With low data rates this is not a problem, but the 100 cycles of latency is too much for the 44.1 kHz Resampler.

This patch replaces the normal SCI interrupt handler with another version in encoding and codec modes (decoder mode is not changed). The new version handles HDAT0 reads in 45 cycles and HDAT1 reads in 35 cycles. This allows the 44.1 kHz resampler to perform correctly, otherwise it can miss interrupts (DAC, SCI and ADC interrupts have higher priority than SRC).

WRAMADDR and WRAM handling have also been hand-written.

Note: the rewritten SCI interrupt handler is interruptable, so DO NOT in any circumstances perform another SCI operation before DREQ has risen or you have given the appropriate time for the current operation to finish.

ADMixer: Configuration Problem

ADMixer takes channel and samplerate configuration from the wrong variable, thus resulting virtually always in stereo mode and 192 kHz rate. Also, input channels could

get swapped when using ADMixer with different samplerates, or using ADMixer after encoding mode without a hardware reset.

This patch reads settings from `adMixerConfig` and also corrects channel swap.

CLOCKF allows max 5.0x clock

You are able to set the internal clock upto 5.0× using the CLOCKF register, but some encoder/codec combinations could use even more.

With this patch the clock adder part is used for both encoding and codec modes. Note this if you use UART TX mode and calculate the UART divider yourself!

Codec mode: using AEC crashes the codec

Using AEC crashes the codec mode after a few seconds for an unknown reason. It seems calling the AEC functions through an extra stub functions prevent the crashing.

FLAC decoding: wasted bits

The FLAC decoding does not compensate the signal level when "wasted bits" are used. This causes for example the signal level to be halved occasionally.

This patch fixes the problem.

FLAC decoding: buffer management error at decode startup

The FLAC decoding extends the stream buffer from the normal 2048 bytes to 12288 bytes to reduce the required peak data transfer rate. In certain conditions this extend does not work correctly and the FLAC file decoding can get stuck (decoding headers) or causes noise at the start of the playback.

This patch fixes the problem.

AAC: ADTS loop did not clear error indicator

ADTS decoding left error variable un-cleared. If errors were encountered in decoding, for example because of bit errors, the error variable gets set. If the frame sizes are correct (the next header is found where expected), the decoder is not reinitialized, leaving a non-zero value in the error variable.

If the error variable is non-zero and the first element in the AAC frame is a non-audio frame, then the element loop is exited without decoding the audio element.

This patch fixes the problem by clearing the error variable before each ADTS frame.

AAC/MP4: Parsing issue

Some MP4 atoms were thought to have subatoms when they did not, causing too much data to be read. This made the file unplayable. The issue is now fixed.

1.3 New and Extended Features

Encoders: 44.1kHz Rate Not Exact

The analog to digital converter (ADC) allows only a few fixed rates, and the audio path resampler allows only integer dividers, so only some sample rates are exact in the encoding and codec modes. For example when you request 24000 Hz, you get the exact rate, but when you request 44100 Hz, the resulting rate is actually 44201 Hz.

This patch adds a hardware/software Resampler, which provides an exact requested rate. The resampler can be disabled by setting AICTRL3 bit 9 (0x0200) during the encoding mode activation. The resampler takes about 15 MHz at 44.1 kHz samplerate. If you use Ogg Vorbis files with 32 kHz or higher sampling rate you probably need 5.5× clock (see 5.0× fix later in this document).

Also, if encoding rate is already exact, the resampler is not activated. SCI_DECODE_TIME is set to 0x8000 when the resampler has been activated.

When you use the resampler, use vs1063a with at least 4.5× clock. Recommended clock setting will be given in vs1063 datasheet when we have verified them.

Codec: 16 kHz and 32 kHz rates are not supported.

Due to the encoder mode always selecting the next highest ADC rate, 16 kHz and 32 kHz are not possible in codec mode.

This patch changes the ADC rate and decimation rate to support also 16 kHz and 32 kHz encoding rates in codec mode.

New ADC modes for codec/encoding modes

ADC channel modes 6 and 7 produce stereo files but select only left (6) or right (7) channel data.

Application Hook at 0x60

Normally zero samples are inserted to the audio FIFO whenever new data to be decoded does not arrive quickly enough and the audio buffer is getting too empty (less than 64 stereo samples).

However, vs1053 and vs1063 also have DAC FIFO underrun detection, which causes the audio output to behave cleanly even if new samples are not inserted. If there are no new samples to be sent to DAC, the signal will be faded slowly towards zero.

Writing 0x60 to SCI_AIADDR causes the zero sample insertion to be disabled. Additionally whenever any decoder outputs samples, value of 1 will be written to SCI_AICTRL0. The user must clear the value.

Write 0 to SCI_AIADDR to disable these features.

Support for AAC Short Frames in ADTS (DAB+)

AAC decoding specifies two transfer lengths: 1024 sample and 960 sample frames. The former is normally used in all AAC files. The latter short frame length is used in broadcast applications that require audio frames to be an integer multiple of 20ms. DAB+ is one such system.

To play the AAC files using short frame length at the correct pitch, use 13.1072MHz or higher crystal, but report the XTAL in SCI_CLOCKF 6.25% slower (i.e. the ratio 960/1024).

There are a few bugs in the decoding of short frame length files in VS1063a. These problems are corrected by this patch when using ADTS format.

The flag to indicate short frames is only available in the mp4 container format. Systems using DAB+ prefer to use the ADTS container, so the short frame length flag needs to be conveyed as side information.

When decoding ADTS, the short frame length flag is taken from parametric_x.config1 bit 9. To set the decoding into short frame length mode, write (after software reset, patch loading and other initialization) WRAMADDR = 0x1e03, WRAM = 0x0210, to set parametric_x.config1 to specify short frames (0x0200) and do not automatically upsample (0x0010).

Note: the short frame length decoding in the mp4 format is not yet patched. If you have mp4 files with the short frame length flag set, please send it to us as an example.

2 How to Load a Plugin

A plugin file (.plg) contains a data file that contains one unsigned 16-bit array called plugin. The file is in an interleaved and RLE compressed format. An example of a plugin array is:

```
const unsigned short plugin[10] = { /* Compressed plugin */
  0x0007, 0x0001, 0x8260,
  0x0006, 0x0002, 0x1234, 0x5678,
  0x0006, 0x8004, 0xabcd,
};
```

The vector is decoded as follows:

1. Read register address number `addr` and repeat number `n`.
2. If `(n & 0x8000U)`, write the next word `n` times to register `addr`.
3. Else write next `n` words to register `addr`.
4. Continue until array has been exhausted.

The example array first tells to write 0x8260 to register 7. Then write 2 words, 0x1234 and 0x5678, to register 6. Finally, write 0xabcd 4 times to register 6.

Assuming the array is in `plugin[]`, a full decoder in C language is provided below:

```
void WriteVS10xxRegister(unsigned short addr, unsigned short value);

void LoadUserCode(void) {
  int i = 0;

  while (i<sizeof(plugin)/sizeof(plugin[0])) {
    unsigned short addr, n, val;
    addr = plugin[i++];
    n = plugin[i++];
    if (n & 0x8000U) { /* RLE run, replicate n samples */
      n &= 0x7FFF;
      val = plugin[i++];
      while (n--) {
        WriteVS10xxRegister(addr, val);
      }
    } else {           /* Copy run, copy n samples */
      while (n--) {
        val = plugin[i++];
        WriteVS10xxRegister(addr, val);
      }
    }
  }
}
```