

VS1053B WAV PCM RECORDER

VSMPG “VLSI Solution Audio Decoder”

Project Code:
Project Name: VSMPG

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

| Revision History | | | |
|------------------|------------|--------|---|
| Rev. | Date | Author | Description |
| 1.00 | 2013-07-12 | HH | Documentation update. Software unchanged. |
| 0.90 | 2010-06-28 | HH | Initial release. |

Contents

| | |
|---|-----------|
| VS1053b WAV PCM Recorder Front Page | 1 |
| Table of Contents | 2 |
| 1 Introduction | 3 |
| 2 The VS1053b WAV PCM Recorder | 4 |
| 2.1 Limitations and Requirements | 4 |
| 2.2 Running the VS1053b WAV PCM Recorder | 5 |
| 2.2.1 VS1053b WAV PCM Recorder Registers | 5 |
| 2.2.2 Loading and Starting the Code | 6 |
| 2.2.3 Reading PCM Data | 7 |
| 2.2.4 Reading Additional Data while Recording PCM | 8 |
| 2.2.5 Finishing PCM Recording | 8 |
| 2.2.6 Filling File Size Data to a PCM RIFF Header | 9 |
| 2.3 Recording Levels and Automatic Gain Control (AGC) | 10 |
| 2.3.1 Reading the Recording Level | 10 |
| 2.3.2 Setting AGC | 11 |
| 2.3.3 Building a Useful VU Meter | 12 |
| 2.3.4 Converting from Linear to Decibel Scale | 14 |
| 3 Performance of the Recorder | 15 |
| 3.1 Playback Performance | 15 |
| 3.1.1 Playback Interchannel Isolation | 15 |
| 3.1.2 Playback Noise and THD Ratios | 16 |
| 3.1.3 Playback Frequency Response | 17 |
| 3.2 Recording Performance | 18 |
| 3.2.1 Recording Noise and THD Ratios | 19 |
| 3.2.2 Recording Frequency Response | 20 |
| 3.2.3 Recording Monitor Frequency Response | 21 |
| 4 How to Load a Plugin | 22 |
| 5 Contact Information | 23 |

List of Figures

| | | |
|---|---|----|
| 1 | Example colour VU meter. | 12 |
| 2 | Example monochrome VU meter. | 12 |
| 3 | Real record display with monochrome VU meter. | 13 |
| 4 | Playback noise + THD ratios. | 16 |
| 5 | Playback frequency response. | 17 |
| 6 | Recording noise + THD ratios. | 19 |
| 7 | Recording frequency response. | 20 |
| 8 | Recording image rejection frequency response. | 20 |
| 9 | Recording monitor frequency response. | 21 |

1 Introduction

This document is an instruction manual on how to use the VS1053b WAV PCM Recorder application. The document is arranged as follows.

Chapter 2 describes the application, then tells how to load and run it. It also discusses recording levels and recording level meters.

Chapter 3 contains performance measurements.

Chapter 4 tells how to load a plugin code to a VS10XX chip.

Finally, Chapter 5 contains VLSI Solution's contact information.

2 The VS1053b WAV PCM Recorder

The *VS1053b WAV PCM Recorder Application* is a 48 kHz WAV/PCM stereo encoder for VLSI Solution's VS1053b integrated circuit.

The Recorder is tweaked for best possible audio quality. It also offers a data buffer which is significantly larger than the buffer offered by VS1053b's firmware for 48 kHz stereo PCM (404 ms vs 10.5 ms). This particularly helps if saving audio data to an SD card, because SD cards can lock up for significant times when doing internal block erase operations.

Unlike VS1053b's firmware encoder, the Recorder includes a standard RIFF WAV header in its output.

Some key features of the recorder are:

- Recordings are made at 48 kHz in 16-bit stereo PCM format, at a bitrate of 1.536 Mbit/s.
- Dynamic range 95 dB, S/N ratio 86 dB, THD 0.013 %, frequency response 20-20000 Hz. (Chapter 3.2)
- 76.5 KiB internal audio buffer (404 ms at 48 kHz).

The VS1053b WAV PCM Recorder is provided as a plugin, downloadable at <http://www.vlsi.fi/en/support/software/vs10xxapplications.html>.

2.1 Limitations and Requirements

- Maximum SPI (SCI) clock speed is internal clock / 7. For a typical crystal of 12.288 MHz and internal 4.5x clock of 55.296 MHz, the maximum speed is then 7.8 Mbit/s. If a higher speed is used, there may be occasional data read errors.
- For the sample rate to be exactly 48 kHz, the crystal needs to be 12.288 MHz. With other clocks, the sample rate is $f_s = \frac{48 \times c}{12.288}$ where c is the crystal frequency in MHz.

2.2 Running the VS1053b WAV PCM Recorder

2.2.1 VS1053b WAV PCM Recorder Registers

| Register | Bits | Description |
|-------------|------|--|
| SCI_MODE | 14 | Select MIC/LINE1 |
| | 12 | Set to 1 when told in the instructions |
| | 1 | VU meter stereo mode activation |
| SCI_AICTRL0 | 15:0 | Maximum signal level, set to 0 |
| SCI_AICTRL1 | 15:0 | Recording gain ($1024 = 1\times$) or 0 for automatic gain control |
| SCI_AICTRL2 | 15:0 | Maximum autogain amplification ($1024 = 1\times$, $65535 = 64\times$) |
| SCI_AICTRL3 | 0 | W: Finish recording, set to 0 |
| | 1 | R: Recording finished, set to 0 |
| | 2 | R: There is at least one byte to read, set to 0 |
| | 15:3 | Unused, always set to zero and they stay zero |

Before activating PCM recording, the user **must** initialize registers SCI_AICTRL0 - SCI_AICTRL3. SCI_AICTRL1 and SCI_AICTRL2 can be altered during recording.

SCI_AICTRL0 records the maximum absolute value of the signal. The maximum value of this linear register is 0x7F7F. For more information on how to use this register, see Chapter 2.3.1. SCI_AICTRL0 is updated once for every sample.

SCI_AICTRL1 controls linear recording gain. 1024 is equal to digital gain 1 (recommended for best quality), 2048 is equal to digital gain 2, and so on. If the user wants to use automatic gain control (AGC), SCI_AICTRL1 should be set to 0. Speech applications are often better off using some AGC, as this helps to get relatively uniform speech loudness in recordings.

SCI_AICTRL2 controls the maximum AGC gain if SCI_AICTRL1 is set to 0. This limits amplification of noise when there is no signal. For more information on recording levels, see Chapter 2.3.

SCI_AICTRL3 offers run-time controls.

2.2.2 Loading and Starting the Code

To load and start the VS1053b WAV PCM Recorder, do the following steps:

1. If you have been doing ADPCM recording, clear bit SM_ADPCM from register SCI_MODE (0).
2. Set the VS1053b clock to the highest value just below 55.3 MHz. Example: if the external clock is 12.288 MHz, this can be done by setting the clock to 4.5X, or 55.296 MHz by writing 0xC000 to register SCI_CLOCKF (3). If you use another input clock, adjust SCI_CLOCKF accordingly.
Examples: Value for 12 MHz is 0xC3e8, 12.288 MHz is 0xC000, 13 MHz is 0xA4E2.
3. Set SCI_BASS (2) to 0.
4. Disable any potential user application by setting SCI_AIADDR (10) to 0.
5. Disable all interrupts except the SCI interrupt by writing 0x2 to VS1053's internal register VS1053_INT_ENABLE. This is done by first writing 0xC01A to SCI_WRAMADDR (7), then 0x2 to SCI_WRAM (6).
6. Load the plugin profile you intend to run (Chapter 4). Note that you will have to redo this each time you want to activate recording.
7. Set bit SM_ADPCM (12) in register SCI_MODE (0) to 1. At the same time, you can also select LINE input instead of MIC input by setting bit SM_LINE1 (14) to 1. If you want to use the VU meter in stereo mode, set also bit SM_LAYER12 (1) to 1. Do *not* set SM_RESET (2) at the same time!
8. Set recording level control registers SCI_AICTRL1 (13) and SCI_AICTRL2 (14). Typical good values for conservative AGC are 0 and 4096, respectively. For a HiFi application, recommended values are 1024 and 0, respectively.
9. If you want to use a VU meter, write 0x8080 to SCI_AICTRL0 (12) (or 0x8000 if you want to use the mono mode of the VU meter).
10. Set a proper value (often 0) to SCI_AICTRL3 (15).
11. Activate the encoder by writing 0x34 to register SCI_AIADDR (10).
12. Wait until DREQ pin is high before reading any data.

2.2.3 Reading PCM Data

After PCM recording has been activated, registers SCI_HDAT0 and SCI_HDAT1 have new functions.

The PCM buffer size is 39166 16-bit words, or approximately 78 KiB. The fill status of the buffer can be read from SCI_HDAT1. If SCI_HDAT1 is greater than 0, you can read that many 16-bit words from SCI_HDAT0.

If data is not read fast enough from SCI_HDAT0, the buffer overflows and returns to empty state. A data overflow will result in an incorrect file. However, because of the large size of the bitstream buffer, this situation should be avoidable with all but the slowest memory cards.

If you are having trouble with receiving data, notice that all WAV PCM files always begin with the following 4 bytes: 0x52 0x49 0x46 0x46 (the string "RIFF"). If you get 0x49 0x52 0x46 0x46 ("IRFF") instead, you are storing the least and most significant byte in the 16-bit data words incorrectly. If you get 0x00 0x00 0x46 0x46, you have read data from SCI_HDAT0 too soon after starting the application.

2.2.4 Reading Additional Data while Recording PCM

You can get extra side information while recording PCM data to see whether VS1053b is still working. The following VS1053b X memory addresses may be read for extra data:

| X Memory Address | Description |
|------------------|--------------------------------------|
| 0x8 | 16 LSb's of recording time (seconds) |
| 0x9 | 16 MSb's of recording time (seconds) |
| 0xC | 16 LSb's of average bitrate (bits/s) |
| 0xD | 16 MSb's of average bitrate (bits/s) |
| 0x1800 | 16 LSb's of sample counter |
| 0x1801 | 16 MSb's of sample counter |

To read the average bitrate, do the following. First write 0xC to SCI_WRAMADDR (7). Then read from SCI_WRAM (6). This is 16 least significant bits of the bitrate. Then read the 16 most significant bits by reading SCI_WRAM again.

You can read the recording time as a sanity check that VS1053b is working: if the register contents don't change every second, you'll have to take protective measures.

2.2.5 Finishing PCM Recording

To create a proper WAV PCM file, the WAV file needs to be shut down properly. The following algorithm can be used to implement this:

1. Set bit 0 of SCI_AICTRL3 (15) to 1.
2. Continue reading data through SCI_HDAT0 and SCI_HDAT1 as usual, but check SCI_AICTRL3's bit 1 from time to time. When this bit turns to 1, the PCM recorder has finished writing to the buffer.
3. Write the remaining words from the bitstream buffer as normal using SCI_HDAT0 and SCI_HDAT1.
4. Reset VS1053b to normal state using software reset. Remember to clear register SCI_MODE (0) bit SM_ADPCM (12) if you don't wish to start ADPCM recording. Remember to also set bit SM_LAYER12 (1) to an appropriate value.
5. Fill in file size data to the PCM RIFF header, as explained in Chapter 2.2.6.
6. If you want to restart recording, you have to completely reload and restart the recording application.

2.2.6 Filling File Size Data to a PCM RIFF Header

After a file has been written, its header bytes needs to be modified. This must be done because the header contains the size of audio data, which is unknown when recording starts. Thus, the user needs to read the first bytes of the recording file, modify them, then rewrite them.

The following shows a RIFF header for a 16-bit stereo file. Note that 2- and 4-byte values are little-endian (lowest byte first).

| File Offset | Field Name | Size | Bytes | Description |
|-------------|---------------|------|-------------------|---------------------------|
| 0 | ChunkID | 4 | "RIFF" | |
| 4 | ChunkSize | 4 | F0 F1 F2 F3 | File size - 8 |
| 8 | Format | 4 | "WAVE" | |
| 12 | SubChunk1ID | 4 | "fmt " | |
| 16 | SubChunk1Size | 4 | 0x10 0x0 0x0 0x0 | 20 |
| 20 | AudioFormat | 2 | 0x01 0x0 | 0x1 for PCM |
| 22 | NumOfChannels | 2 | 0x02 0x0 | 1 for mono, 2 for stereo |
| 24 | SampleRate | 4 | 0x80 0xbb 0x0 0x0 | 0xbb80 for 48 kHz |
| 28 | ByteRate | 4 | 0x00 0xee 0x2 0x0 | 0x2ee00 for 48 kHz stereo |
| 32 | BlockAlign | 2 | 0x02 0x00 | 2 for mono, 4 for stereo |
| 34 | BitsPerSample | 2 | 0x10 0x00 | 16 bits / sample |
| 36 | SubChunk3ID | 4 | "data" | |
| 40 | SubChunk3Size | 4 | D0 D1 D2 D3 | Data size (File Size-36) |
| 44 | Samples... | | | Audio samples |

You have to fill in the header size datas F and D after finishing recording. Other fields have been written correctly by the Recorder to begin with. If S is the size of the complete file in bytes, put $S - 8$ to F and $S - 36$ to D . E.g. if the file size is $S = 1000000$ bytes, then $F = S - 8 = 999992$ and $D = S - 36 = 999964$, is F0...F3 would be 0x38, 0x42, 0x0f, 0x00 and D0...D3 = 0x1c, 0x42, 0x0f, 0x00.

The PCM data is read from SCI_HDAT0 and written into file as follows. The high 8 bits of SCI_HDAT0 should be written as the first byte to a file, then the low 8 bits. Note that this is contrary to the default operation of some 16-bit microcontrollers, and you may have to take extra care to do this right.

Below is an example of a valid header for a 48 kHz stereo PCM file that has a final length of 1798768 (0x1B7270) bytes:

```

0000 52 49 46 46 68 72 1b 00 57 41 56 45 66 6d 74 20 |RIFFhr..WAVEfmt |
0010 10 00 00 00 01 00 02 00 80 bb 00 00 00 ee 02 00 |.....w...|
0020 02 00 10 00 64 61 74 61 44 72 1b 00          |....dataDr.....|

```

2.3 Recording Levels and Automatic Gain Control (AGC)

The Recorder offers signal level monitoring through SCI_AICTRL0. It is recommended that devices that offer recording would show a signal level in a decibel scale. For the decibel scale, see Chapter 2.3.4. This can be done by showing and clearing SCI_AICTRL0 contents at regular intervals.

A good VU meter should be implemented in such a way that it visually advises the user to avoid using the last 6 dB of the available dynamic range.

2.3.1 Reading the Recording Level

The recording level meter has two settings: mono and stereo. Mono mode records the combined highest sample value while stereo mode records the highest absolute value for each channel separately. It is compatible with the Ogg Vorbis Encoder VU meter.

Recording Level Meter: Stereo Mode

Activate stereo mode by setting SCI_MODE (0) register bit SM_LAYER12 (1) to 1.

To read the left and right channel levels, repeat the following loop:

- Write 0x8080 to SCI_AICTRL0.
- Wait for at least 1/50 s. Note: Constant reading of side information causes load on the VS1053 and may cause unexpected crackles in sound.
- Check whether SCI_AICTRL0 & 0x8080 is 0. If not, wait a little more and read again.
- Left channel value is SCI_AICTRL0 & 0x7F00.
- Right channel value is (SCI_AICTRL0 & 0x7F) × 256.
- Use the values as explained in this Chapter. Repeat the loop.

Recording Level Meter: Mono Mode

Activate mono mode by clearing SCI_MODE (0) register bit SM_LAYER12 (1) to 0.

To read the level, repeat the following loop:

- Write 0x8000 to SCI_AICTRL0.
- Wait for at least 1/50 s. Note: Constant reading of side information causes load on the VS1053 and may cause unexpected crackles in sound.
- Check whether SCI_AICTRL0 & 0x8000 is 0x0. If not, wait a little more and read again.
- Use the value as explained in this Chapter. Repeat the loop.

2.3.2 Setting AGC

When the highest dynamic range and sound fidelity is required, AGC should be turned off and recording gain should be set to 1 (SCI_AICTRL1 = 1024). A good example of this would be music recording, although there might be cases where recording level control would be needed even with these cases.

However, in some cases it is required that the audio dynamic range is compressed. An example of such a case is when a device should retain a uniform recording level of a discussion of several people or of one person moving closer and further from the recording device. In such a case, it is a good idea to use AGC.

The AGC unit adjusts signal power in such a way that the maximum sample value from a sine wave would become as close to 16300 as possible. If the signal is too strong, recording level is decreased, and vice versa. The maximum recording level can be set with register SCI_AICTRL2.

When AGC is used, conservative maximum gain values often give the best sounding results. Example: 12dB (SCI_AICTRL1 = 0, SCI_AICTRL2 = 4096). In some cases more extreme values may help to make quiet speech more intelligible, but such values may also add excessive background noise and make sound quality less pleasing.

2.3.3 Building a Useful VU Meter

In an encoder application, if the recording level is too low, extraneous background noise may be introduced to the sound. Conversely, if the recording level is so high that the highest values cannot be represented numerically, signal clipping occurs, and this may cause severe distortion to sound.

In a recording device, it is useful to have a VU meter that shows the signal level so that both too low signal levels and clipping is avoided. This is very important so that the user has a chance to either adjust the recording or input signal level.

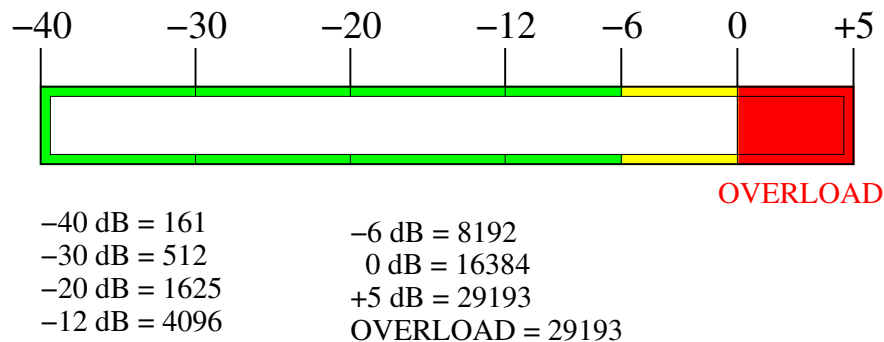


Figure 1: Example colour VU meter.

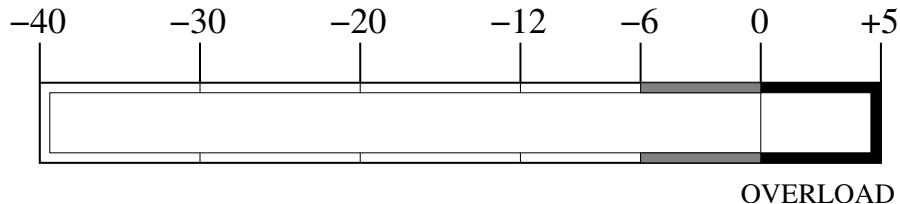


Figure 2: Example monochrome VU meter.

Figures 1 and 2 show example VU meters. The 0 dB reference point has been set to signal level 16384, which is one half of the maximum amplitude, leaving a 6 dB headroom for the device.

The lowest signal level shown is a choice that can vary a lot depending on the application. While -40 dB is a high-fidelity favourite, a VU meter will work just as well with a lower limit of -20 dB. If display space is scarce, low limit could even be set to -12 dB.

Between -6 and +6 dB the VU meter precision should preferably be 1 dB, and at most 2 dB. Below -12 dB step size can be several decibels.

The important thing in a VU meter is to visually show the user that it is not recommended to regularly use the highest 6 dB of the recording scale, and that an overload situation ($\geq +5$ dB, or over linear value 29193) is an error condition. To help this the OVERLOAD symbol should be kept lit or blinking for at least 2 seconds each time an overload situation occurs. Many of these cues are often ignored in digital VU meters, and partly as a result of this even professional recordings are all too often made at recording levels that destroy signal integrity.



Figure 3: Real record display with monochrome VU meter.

Figure 3 shows a real implementation of a recording display that uses a Monochrome VU Meter. Information shown on the top line includes recording time, free disc space, overload warning and a bitrate counter. All this data except from the free disc space number have been obtained from the Recorder.

Recording gain and AGC can be set at the center of the screen if a profile is used that support these functions. However, they are greyed out in this picture because they are not available in the Stereo Music profile that has been used.

The VU Meter is at the bottom. The solid line is the current recording level as read from SCI_AICTRL0 and converted to Decibel scale as shown in Chapter 2.3.4. The greyed line is the top level of the last 3 seconds. If the greyed line $\geq +5$ dB, the orange OVERLOAD message blinks twice a second for three seconds.

2.3.4 Converting from Linear to Decibel Scale

To convert from linear to dB scale, use the formula

$$dB = 20 \times \log_{10}\left(\frac{m}{32768}\right) + 96$$

where dB is the result and m is a value returned by the Recording Level Meter.

To implement a good approximation of this formula from linear to dB scale on architectures where multiplications and logarithms are expensive operations, the following code can be used:

```
const unsigned short linToDBTab[5] = {36781, 41285, 46341, 52016, 58386};

/*
  Converts a linear 16-bit value between 0..65535 to decibels.
  Reference level: 32768 = 96dB (largest VS1053b number is 32767 = 95dB).
  Bugs:
    - For the input of 0, 0 dB is returned, because minus infinity cannot
      be represented with integers.
    - Assumes a ratio of 2 is 6 dB, when it actually is approx. 6.02 dB.
*/
unsigned short LinToDB(unsigned short n) {
    int res = 96, i;

    if (!n)                /* No signal should return minus infinity */
        return 0;

    while (n < 32768U) { /* Amplify weak signals */
        res -= 6;
        n <<= 1;
    }

    for (i=0; i<5; i++)    /* Find exact scale */
        if (n > linToDBTab[i])
            res++;

    return res;
}
```

3 Performance of the Recorder

All figures depicted in this chapter are from actual measurements using a VS1053b + VS1000 Hi-Fi Recorder. During the recording there have been among other things memory card I/O activity (to store the files) and OLED display activity. Thus the results are representative of an actual system using the VS1053, not specially tuned laboratory boards.

A Rohde & Schwarz UPV Audio Analyzer / Signal Generator was used for the tests. The board was connected to the analyzer with a mini-plug to RCA cable.

Unless otherwise noted, all measurements in this section have been performed with 48 kHz 16-bit stereo PCM WAV files at 1001.23 Hz with a 200 k Ω pick-up, volume at full (-0 dB).

3.1 Playback Performance

This playback section is only provided for the completeness. It is not a property of the VS1053b WAV PCM Recorder.

Below is a summary of the playback characteristics, followed by more detailed measurement results.

| Parameter | Typ | Unit |
|---|------------|------------|
| Full Scale Output Voltage (200 k Ω) | 1.89 | Vpp |
| Full Scale Output Voltage (200 k Ω) | 670 | mVrms |
| Background noise on empty channel (200 k Ω) | 12.8 | μ Vrms |
| Dynamic Range (DAC unmuted, A-weighted) | 94 | dB |
| S/N Ratio (full scale signal) | 88 | dB |
| Interchannel Isolation (Cross Talk) | 53 | dB |
| Frequency Response (20-20000 Hz) | ± 0.15 | dB |

3.1.1 Playback Interchannel Isolation

| Parameter | Left | Right | Unit | Isolation |
|-------------------|------|-------|-------|-----------|
| Full Left Signal | 670 | 1.44 | mVrms | 53.4 dB |
| Full Right Signal | 1.44 | 670 | mVrms | 53.4 dB |

3.1.2 Playback Noise and THD Ratios

| Signal / dB | SN+THD / dB | S/N / dB | THD / dB |
|-------------|-------------|----------|----------|
| 0 | 66.9 | 88.7 | 68.0 |
| -1 | 67.8 | 88.1 | 68.8 |
| -2 | 68.6 | 88.2 | 69.6 |
| -3 | 69.3 | 87.6 | 70.4 |
| -4 | 70.0 | 87.3 | 71.1 |
| -5 | 70.7 | 86.8 | 71.8 |
| -6 | 71.4 | 86.3 | 72.5 |
| -8 | 73.1 | 84.7 | 74.3 |
| -10 | 74.6 | 83.0 | 76.2 |
| -12 | 76.1 | 81.3 | 78.1 |
| -15 | 76.6 | 78.3 | 81.0 |
| -18 | 75.0 | 75.5 | 81.3 |
| -24 | 69.4 | 69.5 | 77.7 |
| -30 | 63.5 | 63.5 | 71.0 |
| -36 | 57.6 | 57.7 | 65.0 |
| -42 | 51.5 | 51.5 | 60.0 |
| -48 | 45.5 | 45.7 | 53.0 |
| -54 | 39.6 | 39.5 | 47.0 |
| -60 | 33.4 | 33.5 | 41.0 |
| -66 | 27.5 | 27.7 | 35.0 |
| -72 | 22.0 | 22.2 | 29.0 |
| -78 | 16.5 | 16.5 | 24.0 |

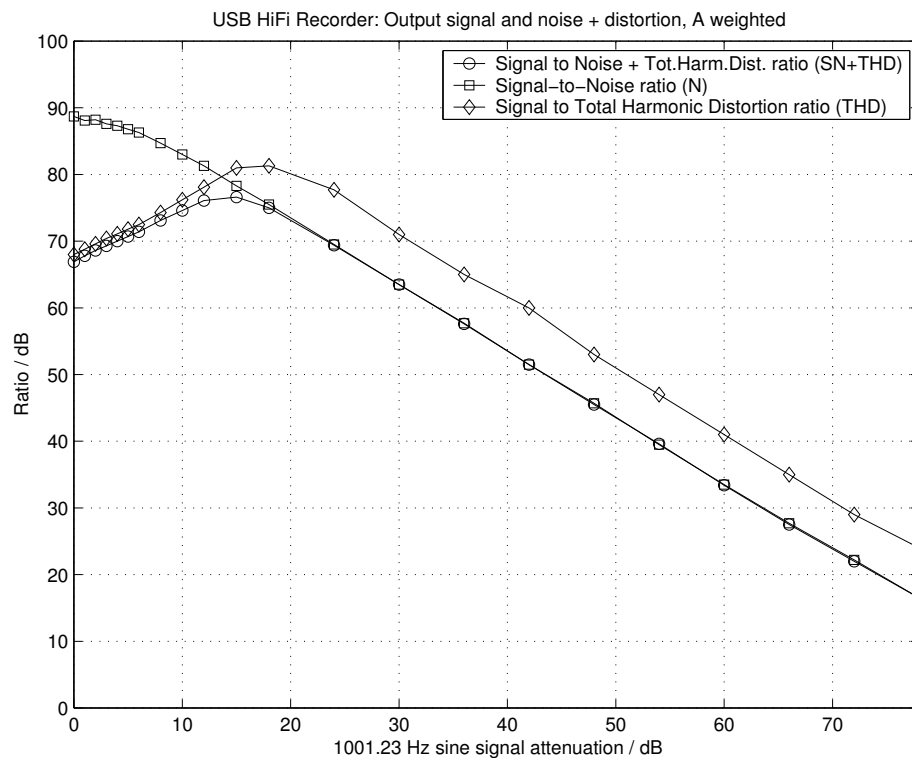


Figure 4: Playback noise + THD ratios.

Figure 4 presents the Noise and THD ratios for 1001.23 Hz 16-bit PCM files that have a signal level between 0 and -78 dB. A-weighting has been used for the measurements. A 200 k Ω load was used.

3.1.3 Playback Frequency Response

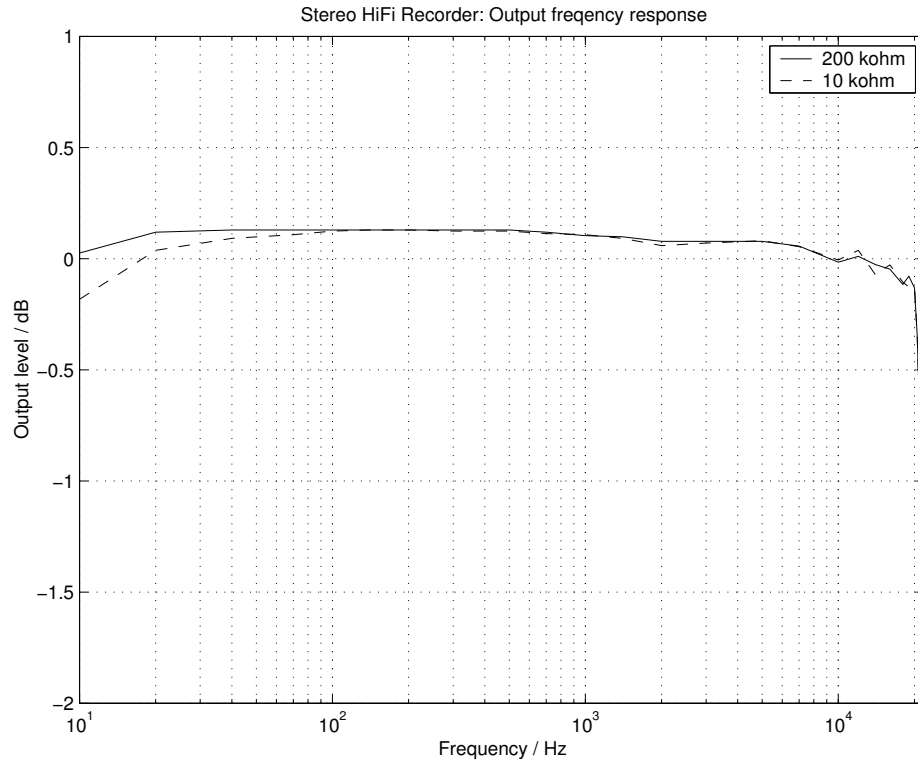


Figure 5: Playback frequency response.

Figure 5 presents the frequency response of playback at 48 kHz. The response is within ± 0.15 dB for the whole human hearing frequency area.

3.2 Recording Performance

Unless otherwise noted, all measurements in this section have been performed with the 48 kHz 16-bit stereo PCM WAV profile with a test frequency of 1001.23 Hz, line input with a 10Ω drive from the signal generator, at unity gain (SCI_AICTRL1 = 1024). Resulting 16-bit stereo output files have then been analyzed with MatLab.

| Parameter | Typ | Unit |
|-------------------------------------|-----------|------------|
| Maximum Amplitude | 3100 | mVpp AC |
| Total Harmonic Distortion | 0.013 | % |
| S/N Ratio | 86 | dB |
| Dynamic Range | 95 | dB |
| Interchannel Isolation (Cross Talk) | 95 | dB |
| Impedance | 80 | k Ω |
| Frequency Response (20-20000 Hz) | ± 0.8 | dB |
| Frequency Response (70-20000 Hz) | ± 0.1 | dB |

3.2.1 Recording Noise and THD Ratios

| Signal / dB | SN+THD / dB | S/N / dB | THD / dB |
|-------------|-------------|----------|----------|
| 2.1 | 21.6 | 43.2 | 21.6 |
| 1.5 | 26.3 | 50.2 | 26.3 |
| 0.8 | 39.1 | 50.7 | 39.4 |
| 0.0 | 76.7 | 86.3 | 77.3 |
| -0.8 | 76.3 | 84.5 | 76.9 |
| -3.9 | 73.3 | 83.5 | 73.7 |
| -6.8 | 71.4 | 83.6 | 71.6 |
| -12.9 | 66.1 | 79.7 | 66.3 |
| -18.9 | 65.9 | 74.4 | 66.6 |
| -24.7 | 66.4 | 69.0 | 69.8 |
| -30.7 | 62.6 | 63.4 | 71.9 |
| -36.7 | 57.0 | 57.5 | 66.3 |
| -42.8 | 51.2 | 51.3 | 65.1 |
| -48.8 | 44.6 | 44.8 | 58.5 |
| -54.8 | 38.4 | 38.4 | 59.6 |
| -60.8 | 32.1 | 32.1 | 50.5 |
| -66.8 | 26.6 | 26.7 | 43.8 |
| -72.9 | 21.1 | 21.2 | 38.0 |
| -78.9 | 15.3 | 15.4 | 32.7 |
| -84.7 | 9.5 | 9.5 | 33.2 |
| -90.7 | 3.5 | 3.5 | 31.9 |
| -96.7 | -2.5 | -2.5 | 26.7 |
| -102.8 | -8.7 | -8.7 | 19.8 |

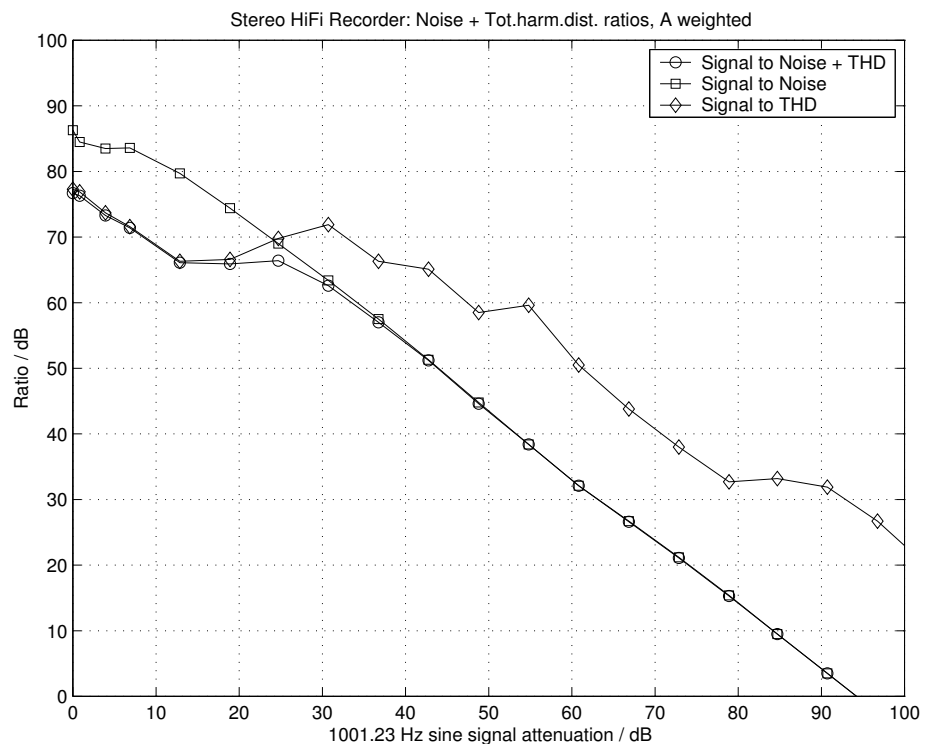


Figure 6: Recording noise + THD ratios.

Figure 6 shows the Noise and THD ratios of recording a 1001.23 Hz sine signal at different signal levels. 0 dB level is 1.1 V(rms), or 3.1 Vpp. The A-weighting curve has been used.

3.2.2 Recording Frequency Response

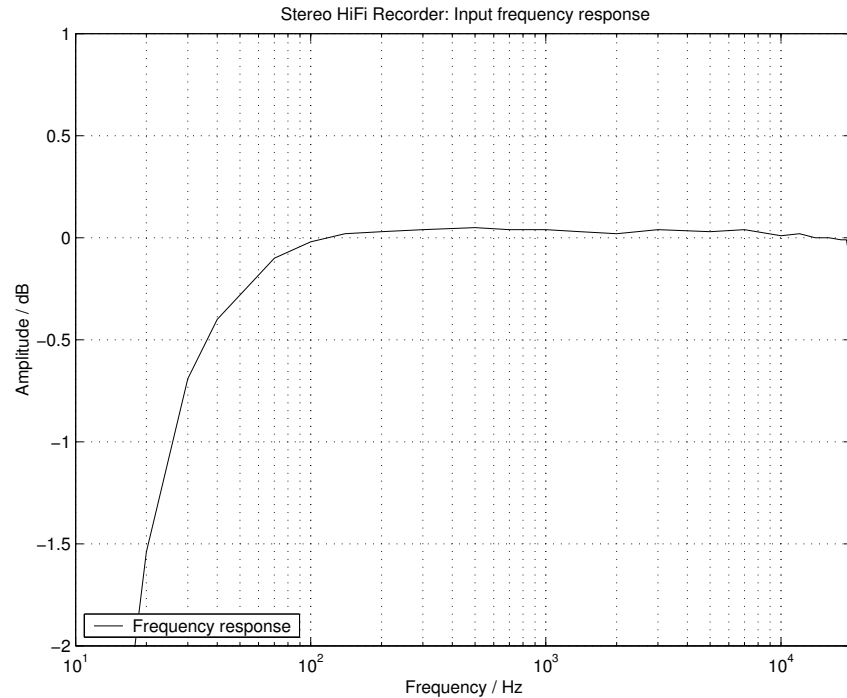


Figure 7: Recording frequency response.

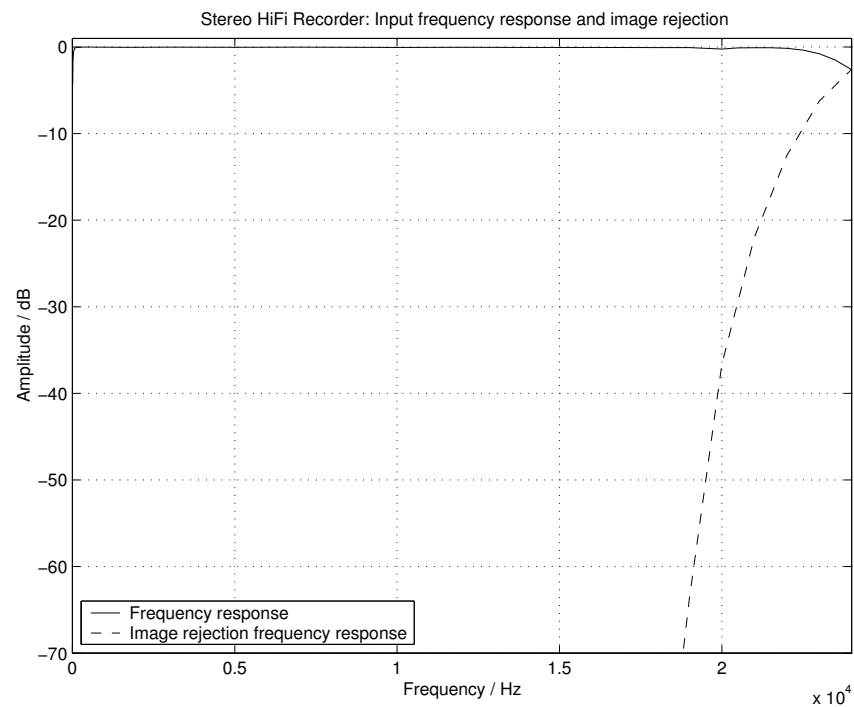


Figure 8: Recording image rejection frequency response.

Figure 7 presents recording frequency response and Figure 8 image rejection.

3.2.3 Recording Monitor Frequency Response

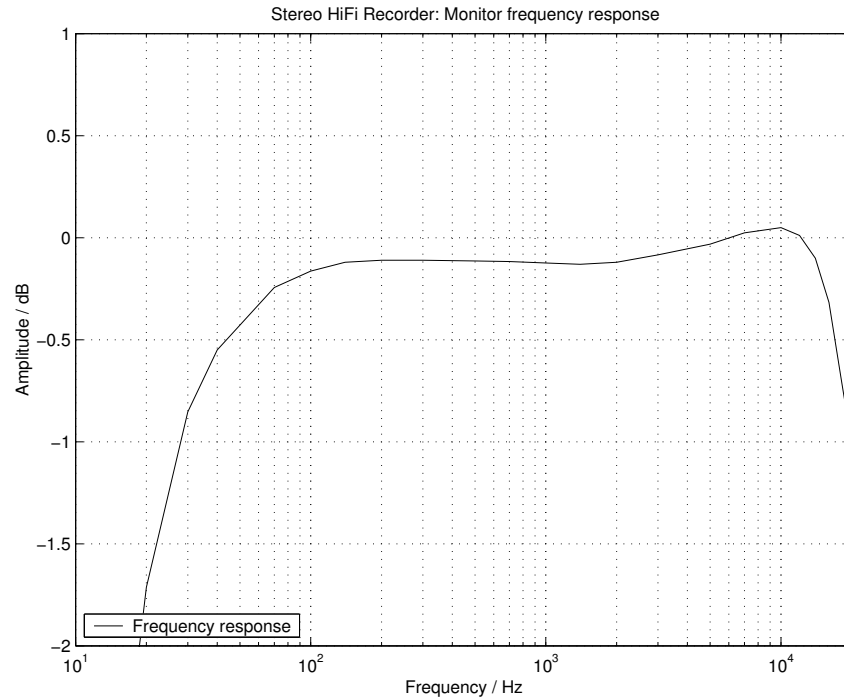


Figure 9: Recording monitor frequency response.

Figure 9 presents the recording monitor frequency response. Unlike other results under Chapter 3.2, these have been measured from line output with a 200 k Ω impedance.

Note that the recording monitor does *not* represent recording quality that goes to the output file, which is presented in Chapters 3.2.1 and 3.2.2.

4 How to Load a Plugin

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

The example vector 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 vector is in vector `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);
            }
        }
        i++;
    }
}
```

5 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

Fax: +358-3-3140-8288
Phone: +358-3-3140-8200
Commercial e-mail: sales@vlsi.fi
URL: <http://www.vlsi.fi/>

For technical support or suggestions regarding this document, please participate at
<http://www.vsdsp-forum.com/>

For confidential technical discussions, contact
support@vlsi.fi