Application Note for CDB4812GTR Effects Code

1.0 INTRODUCTION

Cirrus Logic, Inc. has developed a suite of sound effects for use on the CS4812 DSP + CODEC chip. The sound effects included in this code release are targeted for use in electric guitar amplifiers. The high level of integration offered by the CS4812 device makes this high quality effects solution attractive, even in lower cost guitar amplifiers.

Highlighting the suite of effects is a very convincing spring reverb algorithm recently developed at Cirrus Logic, Inc. In A/B comparisons, this algorithm has proven to be almost indistinguishable from the actual mechanical spring reverb.

Rounding out the effects suite are several effects typically found in expensive guitar amplifiers such as chorus, flange, tremolo, and delay.

The CS4812 is also capable of providing combination effects such as chorus + spring reverb, delay + spring reverb and spring reverb + tremolo.

The CDB4812GTR firmware allows an external microcontroller to change the DSP effects parameters in real time via an SPI or $I^2C^{®}$ serial communication interface. This application note will provide the microcontroller programmer with a description of these application control messages.

1.1 Description of Effects Modules

The effects of the CDB4812GTR firmware are organized into 2 stages with a flexible signal flow that allows many different effects to be generated. Figure 1 shows the audio signal flow.

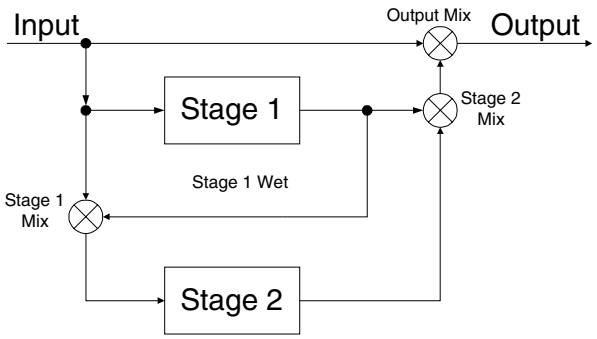

Figure 1. Signal Flow Diagram

You can choose from several effects in each stage. The user may choose one effect from each stage at a time or bypass a stage altogether.

Caution Although all of the effects can be adjusted in real-time, changing parameters may produce audible artifacts.

The system designer needs to select which parameters to change in real-time and whether to temporarily mute the audio outputs during parameter changes.

Table 1 shows the available effects for each stage.

STAGE 1	STAGE 2
Chorus/Flange	Spring Reverb
Tremolo	Delay

Table 1. Effects Modules

Each effect module has several parameters that may be adjustable in real-time. These effects are given in Tables 1 through 8.

CIRRUS LOGIC®
P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.cirrus.com

1.1.1 Chorus/Flange

The chorus/flange effects module consists of a delay line with feedback. Figure 2 shows the Chorus/Flange block diagram and Table 2 shows the parameters that are available for real-time adjustment.

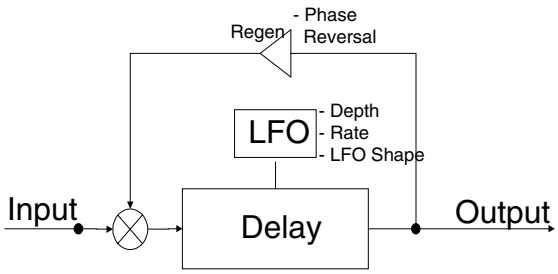

Figure 2. Chorus/Flange

Chorus/Flange Parameters
Depth
Rate
Regen
LFO Shape
Phase Reversal

Table 2. Chorus/Flange Parameters

Depth here refers to the delay line length. *Rate* here refers to the rate of modulation of the delay line. *Regen* here refers to the amount of feedback from output added back to input. *LFO shape* here refers to the delay line modulation waveform shape. *Phase reversal* here refers to the phase of the regen feedback.

1.1.2 Tremolo

The Tremolo effects module consists of an amplitude modulation block. Figure 3 shows the Tremolo block diagram and Table 3 shows the parameters available for real-time adjustment.

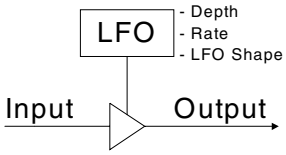

Figure 3. Tremolo

Tremolo Parameters
Depth
Rate

Table 3. Tremolo Parameters

Depth here refers to the ratio of the amplitude modulation compared to unity gain. A depth of zero corresponds to no modulation and a depth of 1 corresponds to full amplitude modulation. *Rate* here refers to the rate of amplitude modulation. The LFO shape refers to the modulation waveform shape.

1.1.3 Spring Reverb

The spring reverb module consists of 2 main sub-blocks, the chirp module and the reverb module. Figure 4 shows the spring reverb block diagram and Table 4 shows the parameters available for real-time adjustment.

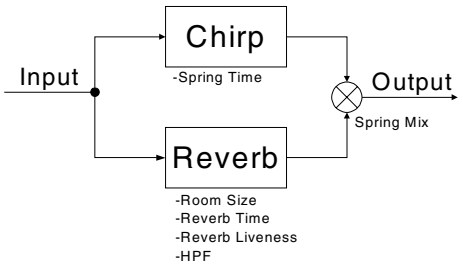

Figure 4. Spring Reverb

Spring Reverb Parameters
Spring Time
Spring Mix
Reverb Time
Reverb Liveness

Table 4. Spring Reverb Parameters

Spring Reverb Parameters
Room Size
High Pass Filter

Table 4. Spring Reverb Parameters (Continued)

Spring Time refers to the decay time of the initial metallic spring or chirp sound. *Spring mix* refers to the relative amount of chirp sound relative to standard reverb sound. *Reverb liveness* refers to the high frequency decay rate in the standard reverb sound. *Room Size* refers to the parameter that controls the apparent size of the modelled room. *High pass filter* refers to the output filter which reduces the high frequency content of the reverb part of the module.

1.1.4 Delay

Figure 5 shows the Delay effects block diagram. This module consists of a long delay block with feedback. Table 5 shows the parameters available for real-time adjustment.

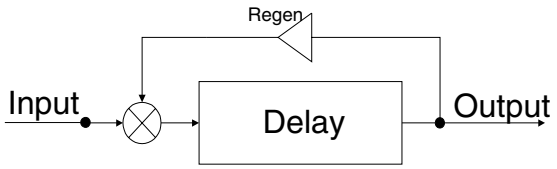

Figure 5. Delay

Delay Parameters
Delay
Regen

Table 5. Delay Parameters

Delay here refers to the length of the delay line. *Regen* here refers to the relative amount of feedback from output to input.

1.1.5 Mixer

There are 3 mixers in the overall effects processing signal chain. The main signal flow diagram in Figure 1 on page 1 shows the mixer locations and Table 6 shows the mixer parameters available for real-time adjustment.

Mixer Parameters
Stage 1 Wet/Dry Mix
Stage 2 Mix
Output Wet/Dry Mix

Table 6. Mixer Parameters

1.1.6 Signal Flow Control and LFO Shapes Selection

Signal flow options are controlled via bits in the Routing Register. Table 7 shows the bit fields in this register.

Bit	Description
0	Reserved (write as 0)
1	Delay Module (0/1 = disable/enable)
2	Spring Reverb Module (0/1 = disable/enable)
3	Reserved (write as 0)
4	Chorus/Flange Module (0/1 = disable/enable)
5	Tremolo Module (0/1 = disable/enable)
6	Chorus Feedback Phase Reversal (0/1 = disable/enable)
7	Stage Swap (0/1 = disable/enable)
8	Input Mute (0/1 = disable/enable)
9	Reserved (write as 0)
10	Reserved (write as 0)
11	LFO shape bitfield 0
12	LFO shape bitfield 1
13-14	Reserved
15	Invert Final Outputs
16-23	Reserved

Table 7. Routing Flag Register Bits

Only one effect from each stage can be selected at a time. Error checking is not done by the application code and is the responsibility of the host.

Caution Failure to obey the routing rules can lead to undesired results.

The Stage Swap bit swaps the effect processing blocks in stage 1 and stage 2. All other parameters such as Stage 1 Mix and Stage 2 mix do not get swapped.

The input mute bit mutes the input signal rather than the output signal. This action allows the enabled effects to decay smoothly.

The LFO shape is meaningful for Chorus/Flange and Tremolo effects. For Chorus/Flange, the LFO shape determines the delay modulation waveform. For Tremolo, the LFO shape determines the amplitude modulation waveform.

The LFO shape is coded in a 2 bit field in the Routing Flag register bits 11 and 12. Table 8 shows the LFO shapes that are available.

Bit 12	Bit 11	LFO Shape
0	0	Sine Wave
0	1	Triangle
1	0	Inverted Rectified Sine
1	1	Trapezoid

Table 8. LFO Shape Bits

1.2 Application Messaging Protocol

Application messaging between the micro-controller can use either the SPI or I^2C serial interface. The physical messaging syntax is different for the SPI and I2C and is available in the *CS4812 Data Sheet*. There are several accessible registers in the control port. The microcontroller will address multiple control port registers during the boot process, but actual application messaging will only involve sending message bytes to the DSP input control port register which has a MAP address of 0x10. Further detail about the MAP register is given in the *CS4812 Data Sheet*. The boot process is also described in the *CS4812 Data Sheet* as a flow chart, but the specific procedure will also be given as microcontroller C code for the user's convenience.

Every application message sent to the CS4812 consists of three bytes. The first data byte is the opcode, specifying the Audio Effects Parameter. The following 2 bytes are the data to be written to the specified parameter. The parameter can take two forms, depending on the opcode you specify. The parameter can either be a positive signed fraction with range 0x0000-0x7FFF, representing 0.0 to 1.0 or an unsigned hex integer with range 0x0000-0xFFFF.

Caution This outlined protocol is rigid and must be followed without exception – any departure from the protocol can result in improper chip operation.

The protocol does not implement error checking or flow control. Table 9 shows a list of effects and their respective opcodes.

Effect	Opcode	Range
Output Wet/Dry Mix	0x00	0x0000-0x7FFF
Spring Level	0x01	0x0000-0x7FFF
Reverb Liveness	0x02	0x0000-0x7FFF
Reverb Time	0x03	0x0000-0x7FFF
Spring Time	0x04	0x0000-0x7FFF
Reverb HPF	0x05	0x0000-0x7FFF
Reverb Room Size	0x06	0x0000-0x7FFF
Chorus Rate	0x07	0x0000-0x7FFF
Chorus Depth	0x08	0x0000-0x7FFF
Chorus Delay	0x09	0x0000-0x7FFF
Chorus Regen	0x0a	0x0000-0x7FFF
Tremolo Rate	0x0b	0x0000-0x7FFF
Tremolo Depth	0x0c	0x0000-0x7FFF
Delay Time	0x0d	0x0000-0x7FFF
Delay Regen	0x0e	0x0000-0x7FFF
Routing Register	0x0f	0x0000-0xFFFF
Stage 1 Wet/Dry Mixl	0x10	0x0000-0x7FFF
Stage 2 Mix	0x11	0x0000-0x7FFF

Table 9. Effect Parameter Opcodes and Ranges

1.2.1 Application Messaging Example

An application messaging example is given below to illustrate a complete host-boot procedure in pseudo-code. The microcontroller should observe a 50 microsecond byte-to-byte latency to prevent corruption of the DSP input buffer. Please note that this pseudo-code example is believed to be correct but is subject to change.

```
//Start of Program
int Chorus_Regen_Value[3] = {0x0A, 0x40, 0x00};
int *ptr;

while (TRUE) {
// Assert DSP Reset HW
Signal(RST, low);
// De-assert DSP Reset HW
Signal(RST, high);

// Configure the converters and clocks
//LJ-24 bit, DSCK=1, SCK/Fs=64
Write_CP_Byte( 0x0d, 0xab);
//DAC from DSP
Write_CP_Byte( 0x06, 0x90);
//DSP=16.384 MIPS,
Write_CP_Byte( 0x03, 0x80)
//give PIO control to DSP;
Write_CP_Byte( 0x0f, 0x06);

// Send Control Port Configuration
// Set /DSPRS bit in CP reg 4 to 0
Write_CP_Byte( 0x04, 0xa4);
// Set DSPBOOT bit in CP reg 4 to 1
Write_CP_Byte( 0x04, 0xa5);
// Set /DSPRS bit in CP reg 4 to 1
Write_CP_Byte( 0x04, 0xa7);

// Prepare DSP for Application Download
Write_CP_Byte( 0x10, 0x00);
Write_CP_Byte( 0x10, 0x00);
Write_CP_Byte( 0x10, 0x04);

// Wait for Boot Ready response from DSP with 0.5 second time-out
Time = 0;
while (Read_Signal(nREQ) && (Time < 0.5)) {}
if ((Time >= 0.5) continue;
if (Read_CP_Byte( 0x1b) != 0x01) continue;

// Download Application Code
Send_CP_Msg(0x10, Application_Download_Image, Download_Image_Size);

// Wait for DSP Boot Success response from DSP with 0.5 second time-out
Time = 0;
while (Read_Signal(nREQ) && (Time < 0.5)) {}
if (Time >= 0.5) continue;
if (Read_CP_Byte( 0x1b) != 0x02) continue;

// Acknowledge DSP Boot Success
Write_CP_Byte( 0x10, 0x00);
Write_CP_Byte( 0x10, 0x00);
Write_CP_Byte( 0x10, 0x05);

// Clear DSPBOOT bit in CP reg 4
Write_CP_Byte( 0x04, 0xa6);

// Send Desired Application Parameter Configuration
Send_CP_Msg(0x10, Application_Parameter_Image, Parameter_Image_Size);

// Example showing setting of Chorus_Regen_Value to 0.5.
Send_CP_Msg(0x10, Chorus_Regen_Value, 3);
```

```
break;
}

// Main Program Loop

while (TRUE) {
// Read Control Input Switches
Read_Control_Inputs(*Control_Parameters);
// Send Commands to DSP
if (Control_Parameter_Change) {
Send_CP_Msg(0x10, Application_Parameter, 3)
}
}
// Subroutines

// Send single byte message to DSP Control Port
void Write_CP_Byte(byte MAP, byte Data) {
CP_Start_Condition();
Send_Byte(Chip_Address);
Send_Byte(MAP_Byte);
Send_Byte(Data);
CP_Stop_Condition();
}
// Send multi-byte message to DSP Control Port
void Send_CP_Msg(byte MAP, byte *Data_Ptr, int Byte_Count) {
int i, *p = Data_Ptr;
CP_Start_Condition();
Send_Byte(Chip_Address);
Send_Byte(MAP_Byte);
for ( i=0; i<Byte_Count; i++)0x
Send_Byte(*p++);
CP_Stop_Condition();
}

// These are low level routines that are specific to the micro-controller and the SPI or I2C
interface used.
void Send_Byte (int Byte) {....}
void CP_Start_Condition(void) {...}
void CP_Stop_Condition(void) {...}
```

• Notes •


SMART
Analog
™