

CS4953x4/CS4970x4
32-bit Audio DSP Family

CS4953x4/CS4970x4

System Designer's Guide

Preliminary Product Information

This document contains information for a new product.
Cirrus Logic reserves the right to modify this product without notice.

Contacting Cirrus Logic Support

For all product questions and inquiries, contact a Cirrus Logic Sales Representative.

To find the one nearest you, go to www.cirrus.com.

IMPORTANT NOTICE

"Preliminary" product information describes products that are in production, but for which full characterization data is not yet available.

Cirrus Logic, Inc. and its subsidiaries ("Cirrus") believe that the information contained in this document is accurate and reliable. However, the information is subject to change without notice and is provided "AS IS" without warranty of any kind (express or implied). Customers are advised to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, indemnification, and limitation of liability. No responsibility is assumed by Cirrus for the use of this information, including use of this information as the basis for manufacture or sale of any items, or for infringement of patents or other rights of third parties. This document is the property of Cirrus and by furnishing this information, Cirrus grants no license, express or implied under any patents, mask work rights, copyrights, trademarks, trade secrets or other intellectual property rights. Cirrus owns the copyrights associated with the information contained herein and gives consent for copies to be made of the information only for use within your organization with respect to Cirrus integrated circuits or other products of Cirrus. This consent does not extend to other copying such as copying for general distribution, advertising or promotional purposes, or for creating any work for resale.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). CIRRUS PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED FOR USE IN PRODUCTS SURGICALLY IMPLANTED INTO THE BODY, AUTOMOTIVE SAFETY OR SECURITY DEVICES, LIFE SUPPORT PRODUCTS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF CIRRUS PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK AND CIRRUS DISCLAIMS AND MAKES NO WARRANTY, EXPRESS, STATUTORY OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, WITH REGARD TO ANY CIRRUS PRODUCT THAT IS USED IN SUCH A MANNER. IF THE CUSTOMER OR CUSTOMER'S CUSTOMER USES OR PERMITS THE USE OF CIRRUS PRODUCTS IN CRITICAL APPLICATIONS, CUSTOMER AGREES, BY SUCH USE, TO FULLY INDEMNIFY CIRRUS, ITS OFFICERS, DIRECTORS, EMPLOYEES, DISTRIBUTORS AND OTHER AGENTS FROM ANY AND ALL LIABILITY, INCLUDING ATTORNEYS' FEES AND COSTS, THAT MAY RESULT FROM OR ARISE IN CONNECTION WITH THESE USES.

Cirrus Logic, Cirrus, the Cirrus Logic logo designs, DSP Composer, Cirrus Extra Surround, Cirrus Original Multichannel Surround, and Cirrus Original Surround are trademarks of Cirrus Logic, Inc. All other brand and product names in this document may be trademarks or service marks of their respective owners.

Dolby, Dolby Digital, Dolby Headphone, Dolby Virtual Speaker, AC-3, and Pro Logic are registered trademarks of Dolby Laboratories, Inc. AAC, Dolby Headphone2, Dolby Virtual Speaker2, and Dolby Digital Surround EX are trademarks of Dolby Laboratories, Inc. Supply of an implementation of Dolby technology does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Dolby Laboratories, to use the implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Dolby Laboratories.

DTS and DTS Digital Surround are registered trademarks of the Digital Theater Systems, Inc. DTS-ES, DTS-ES 96/24, DTS Neo:6, DTS 96/24 are trademarks of the Digital Theater Systems, Inc. It is hereby notified that a third-party license from DTS is necessary to distribute software of DTS in any finished end-user or ready-to-use final product.

THX® is a registered trademark of Lucasarts Entertainment Company Corporation.

Re-equalization is a trademark of Lucasfilm, Ltd.

SRS is a registered trademark of SRS Labs, Inc. Purchaser of CS4953x4/CS4970x4 must sign a license for use of the chip and display of the SRS Labs trademarks. Any products incorporating the CS4953x4/CS4970x4 must be sent to SRS Labs for review. The SRS symbol is a trademark of SRS Labs, Inc. in the United States and selected foreign countries. Neither the purchase of the CS4953x4/CS4970x4, nor the corresponding sale of audio enhancement equipment conveys the right to sell commercialized recordings made with any SRS technology/solution. SRS Labs requires all set makers to comply with all rules and regulations as outlined in the SRS Trademark Usage Manual.

Intel is a registered trademark of Intel Corporation.

Motorola and SPI are registered trademarks of Motorola, Inc.

I2C is a trademark of Philips Semiconductor Corp.

HDMI is a trademark of HDMI Licensing.

BLU-RAY and BLU-RAY DISC are trademarks of SONY KABUSHIKI KAISHA CORPORATION.

Microsoft and Windows XP are registered trademarks of Microsoft Corporation.

Contents

Contents	1-iii
Figures	1-vii
Tables	1-ix
Preface	P-1
P.1 Introduction to CS4953x4/CS4970x4 System Designer's Guide	P-1
P.2 Overview of the CS4953x4/CS4970x4 DSP	P-1
P.2.1 Chip Features.....	P-2
P.3 CS4953x4/CS4970x4 Chip Functional Overview	P-4
P.3.1 DSP Core	P-4
P.3.2 Security Extension module.....	P-4
P.3.3 Debug Controller (DBC)	P-4
P.3.4 Digital Audio Output (DAO1, DAO2) Controller.....	P-4
P.3.5 Digital Audio Input (DAI1) Controller	P-4
P.3.6 Compressed Data Input / Digital Audio Input (DAI2) Controller	P-4
P.3.7 Direct Stream Digital (DSD) Controller.....	P-5
P.3.8 General Purpose I/O	P-5
P.3.9 Serial Control Ports (SPI™ or I2C™ Standards)	P-5
P.3.10 SDRAM Controller.....	P-5
P.3.11 DMA Controller.....	P-6
P.3.12 Timers	P-6
P.3.13 Clock Manager and PLL.....	P-6
P.3.14 Programmable Interrupt Controller.....	P-6
P.4 Firmware Overview	P-6
P.4.1 Robust DSP Manager Firmware API	P-6
P.4.2 DSP Condenser	P-7
P.5 CS40700x Pin Descriptions	P-7
P.5.1 Power and Ground	P-7
P.5.1.1 Power	P-7
P.5.1.2 Ground	P-8
P.5.1.3 Decoupling	P-8
P.5.2 PLL Filter.....	P-8
P.5.2.1 Analog Power Conditioning	P-8
P.5.2.2 PLL	P-9
P.5.3 Clocking	P-9
P.5.4 Control.....	P-10
P.5.4.1 Operational Mode	P-10
P.6 CS4970x4 Pin Assignments	P-11
P.7 CS4953x4 Pin Assignments	P-19
Chapter 1. Operational Modes	1-1
1.1 Overview	1-1
1.1.1 Supported Serial Flash Devices	1-1
1.2 Operational Mode Selection	1-2
1.3 Booting the DSP in Master Boot Mode	1-2
1.3.1 Performing a Master Boot.....	1-6
1.3.1.1 Master Boot Protocol	1-7
1.3.1.2 Messages Read from CS4953x4/CS4970x4	1-8

Chapter 2. Serial Communication Mode	2-1
2.1 Introduction	2-1
2.2 Communication Using the Serial Control Port	2-1
2.3 Serial Control Port Configuration	2-1
2.4 SPI Port	2-1
2.4.1 SPI System Bus Description.....	2-3
2.4.2 SPI Bus Dynamics.....	2-3
2.4.2.1 SCP1_BSY Behavior	2-4
2.4.3 SPI Messaging	2-4
2.4.3.1 Performing a Serial SPI Write.....	2-4
2.4.3.2 SPI Write Protocol	2-5
2.4.3.3 Performing a Serial SPI Read.....	2-5
2.4.3.4 SPI Read Protocol	2-6
2.4.3.5 SCP1_IRQ Behavior.....	2-9
2.5 I2C Port	2-9
2.5.1 I ² C System Bus Description	2-10
2.5.2 I ² C Bus Dynamics	2-11
2.5.3 I2C Messaging.....	2-14
2.5.3.1 SCP1_BSY Behavior	2-14
2.5.3.2 Performing a Serial I2C Write	2-15
2.5.3.3 I2C Write Protocol.....	2-16
2.5.3.4 Performing a Serial I2C Read.....	2-16
2.5.3.5 I2C Read Procedure	2-18
2.5.3.6 SCP1_IRQ Behavior.....	2-20
Chapter 3. Audio Input Interfaces	3-1
3.1 Introduction	3-1
3.2 Digital Audio Input Port Description	3-1
3.2.1 DAI Pin Description	3-1
3.2.2 Supported DAI Functional Blocks.....	3-2
3.2.3 BDI Port.....	3-3
3.2.4 Digital Audio Formats	3-3
3.2.4.1 I2S Format	3-3
3.2.4.2 Left-Justified Format	3-3
3.3 DAI Hardware Configuration	3-4
3.3.1 DAI Hardware Naming Convention	3-4
3.4 Digital Audio Input Port Description	3-7
3.4.1 DSD Pin Description.....	3-7
3.4.2 Supported DSD Functional Blocks	3-7
Chapter 4. Audio Output Interface	4-1
4.1 Introduction	4-1
4.2 Digital Audio Output Port Description	4-1
4.2.1 DAO Pin Description.....	4-1
4.2.2 Supported DAO Functional Blocks	4-3
4.2.3 DAO Interface Formats.....	4-3
4.2.3.1 I2S Format	4-3
4.2.3.2 Left-Justified Format	4-3
4.2.3.3 One-line Data Mode Format (Multichannel).....	4-4
4.2.4 DAO Hardware Configuration.....	4-4
4.2.5 DAO Hardware Naming Convention.....	4-4
4.2.6 S/PDIF Transmitter.....	4-9

Chapter 5. External Memory Interfaces	5-1
5.1 SDRAM Controller	5-1
5.1.1 SDRAM Controller Interface	5-2
5.1.2 SDRAM Interface Signals.....	5-2
5.1.3 Configuring SDRAM Parameters.....	5-3
5.2 SPI Flash Interface	5-6
 Chapter 6. System Design Requirements for SPDIF and HDMI™ Technology Interfaces.....	 6-1
6.1 Introduction	6-1
6.1.1 Designing a SPDIF Input Interface	6-1
6.1.1.1 SPDIF Clocking	6-1
6.1.2 Designing an HDMI Input Interface	6-1
6.1.2.1 HDMI Clocking.....	6-1
6.1.2.2 Decoding Stream Types Over HDMI	6-2
6.1.3 Other System Design Considerations	6-2
 Chapter 7. Overview of Common Firmware Modules	 7-1
7.1 Introduction	7-1
7.2 CS4953x4/CS4970x4 Firmware	7-1
7.2.1 Firmware Modules	7-1
7.2.2 Overlay Architecture	7-1
7.3 Firmware Messaging	7-2
7.3.1 Communication Overview.....	7-2
7.3.2 Writing to the DSP	7-2
7.3.3 Solicited Read	7-3
7.3.4 Unsolicited Read	7-3
7.3.5 Index Configuration	7-4
7.3.6 Unsolicited Messages from DSP to the Host Microcontroller	7-4
7.3.7 DSP_AUTODETECT_MSG.....	7-4
7.3.8 DSP_LAST_ACCN_MSG.....	7-5
7.4 CS4953x4/CS4970x4 DSP Manager API Description	7-6
7.4.1 Microcontroller Interface (API).....	7-6
7.4.2 DSP_CFG_xxx Registers.....	7-7
7.4.2.1 Using DSP Condenser to Change/Load Firmware Modules	7-10
7.4.2.2 Using DSP Condenser to Change the Audio Input Source	7-11
7.4.3 Status Registers	7-12
7.5 Legacy API Still in Use	7-13
7.5.1 Legacy Audio Manager.....	7-13
7.6 OS Firmware Module	7-16
7.6.1 Overview.....	7-16
7.6.2 OS-A and OS-B Module Manager	7-16
7.6.3 Other DSP Audio Manager Registers.....	7-17
 Chapter 8. DSP Condenser.....	 8-1
8.1 Overview	8-1
8.1.1 Purpose of DSP Condenser	8-1
8.2 Development Flow	8-2
8.3 Elements of a Project.	8-4
8.3.1 General Page	8-4

8.3.2 Search paths Page	8-5
8.3.3 Audio sources Page	8-6
8.3.4 Sample rates Page	8-7
8.3.5 Firmware components Page.....	8-8
8.3.6 PPM modes Page.....	8-9
8.3.7 Stream types Page	8-10
8.3.8 Power-up state Page	8-11
8.3.9 WAV update Page	8-12
8.4 Hardware and Software Requirements	8-13
8.4.1 Hardware requirements	8-13
8.4.2 Software requirements	8-13
8.5 Creating a Condenser Project using a Model	8-13
8.5.1 Using the Wizard to Create a Project	8-13
8.6 Creating a Flash Image	8-15
8.6.1 How to create an image.....	8-15
8.6.2 What Does the Image Contain?	8-16
8.7 Using DSP Condenser	8-17
8.7.1 How to use DSP Composer with DSP Condenser	8-17
8.7.1.1 Best Practices	8-17
8.7.1.2 Creating Projects	8-18
8.7.2 Capturing Snapshots	8-19
8.8 Creating a Flash Image	8-20
8.8.1 SPI Flash Image Format.....	8-20
8.8.1.1 Master Boot Image (egg.img)	8-21
8.8.1.2 Updatable Flash Image.....	8-21
8.8.1.3 DSP State Space	8-21
8.8.1.4 Microcontroller Unit (MCU) Scratchpad	8-22
8.8.2 Creating a Serial Flash Image Automatically.....	8-22
8.8.2.1 Using the DSP Condenser Wizard to Create a Serial Flash Image.....	8-23
8.8.2.1.1 Modifying a DSP Condenser Template to Create a Custom Project	8-25
8.8.3 Using DSP Condenser to Program a Flash Image.....	8-25
8.8.4 Field Upgrade of Flash Image	8-25
8.8.4.1 Steps for Carrying Out a Field Upgrade of Flash Image.....	8-26
8.9 DSP Response after Master Boot.	8-26
8.10 Host Activity	8-27
Chapter 9. Using Runtime Condenser	9-1
9.1 DSP Condenser Runtime Application	9-1
9.1.1 Usage	9-1
9.1.1.1 Standard Launch	9-1
9.1.1.2 Command Line with DSP Image.....	9-3
9.1.1.3 Command Line without DSP Image.....	9-3
9.2 Standard Launch	9-4
9.2.1 Connection Group	9-4
9.2.1.1 Connection to Board Button.....	9-4
9.2.1.2 Connection Status Led	9-4
9.2.1.3 Use JP1 Checkbox	9-4
9.2.2 Command Group	9-4
9.2.2.1 Read DSP Button	9-5
9.2.2.2 Reset DSP Button.....	9-5
9.2.2.3 Auto Read Checkbox.....	9-5

9.2.2.4 Command Field	9-5
9.2.2.4.1 Hexadecimal Command	9-5
9.2.2.4.2 Configuration File Command	9-5
9.2.2.4.3 Comments	9-5
9.2.2.5 Browse Button	9-5
9.2.3 DSP Status Group	9-6
9.2.3.1 DSP Reset LED	9-6
9.2.3.2 Interrupt Request LED	9-6
9.2.3.3 Busy LED	9-6
9.2.4 Log Window	9-6
9.2.4.1 Color Coding	9-6
9.2.4.2 Menu Items	9-6
9.2.5 Source Selection Group	9-7
9.2.5.1 Input Source Combo Box	9-7
9.2.5.2 Apply Button	9-7
9.2.5.3 Reset DSP When Changing Source Checkbox	9-7
9.2.6 Source Status Group	9-7
9.2.6.1 Manual Refresh Button	9-8
9.2.6.2 Auto Refresh Button	9-8
9.2.6.3 Auto Read Checkbox	9-8
9.2.6.4 Source Status Text Fields	9-8
9.2.7 DSP Manager API Group	9-8
9.2.7.1 Source	9-10
9.2.7.2 MCLK Factor	9-10
9.2.7.3 Output Fs	9-10
9.2.7.4 Decoder and Decoder Mode	9-10
9.2.7.5 MPM and MPM Mode	9-10
9.2.7.6 VPM and VPM Mode	9-10
9.2.7.7 PPM	9-10
9.2.7.8 PPM Mode List	9-10
9.2.7.9 Refresh Button	9-10
9.2.7.10 Apply Button	9-10
Appendix A. FAQ	A-1
A.1 Introduction	A-1
A.2 List of Questions and Answers	A-1
Appendix B. Optional Features	B-1
B.1 Introduction	B-1
B.2 Communication Using the Parallel Control Port	B-1
Appendix C. Loading/Unloading Firmware Modules	C-1
C.1 Introduction	C-1
C.1.1 DTS 96/24™ and DTS-ES™	C-1
C.1.1.1 Loading DTS 96/24 Decoder	C-1
C.1.1.2 Switching to DTS-ES from DTS 96/24 Decoder	C-1
C.1.2 DTS-ES Matrix and DTS Neo6™	C-1
C.1.2.1 Loading DTS-ES Decoder in Matrix Mode	C-2
C.1.3 Dolby Digital®	C-2
C.1.3.1 Loading Dolby Digital Decoder for Stereo Output	C-2
C.1.3.2 Setting Dolby Digital to AC3 DRC Mode	C-2
C.1.4 Cirrus Logic Signal Generator (SGEN)	C-2
C.1.4.1 Loading the Cirrus Logic SGEN Module	C-2

- C.1.5 Dolby Digital®PLUS C-3
 - C.1.5.1 Loading Dolby Digital Plus for Stereo Output..... C-3
 - C.1.5.2 Setting DRC Modes for Dolby Digital Plus C-3
- C.1.6 Dolby® TrueHD..... C-3
 - C.1.6.1 Loading Dolby TrueHD for Stereo Downmix Output C-3
 - C.1.6.2 Setting DRC Modes for Dolby TrueHD C-4
- C.1.7 Dolby ProLogic® IIx C-4
 - C.1.7.1 Loading Dolby ProLogic IIx C-4
 - C.1.7.2 Using Dolby ProLogic IIx to Output HD Audio Streams C-4
- C.1.8 Dolby Virtual Speaker® 2..... C-5
 - C.1.8.1 Loading Dolby Virtual Speaker 2..... C-5
 - C.1.8.2 Loading Dolby Virtual Speaker 2 with Dolby ProLogic II C-5
 - C.1.8.3 Removing Dolby Virtual Speaker 2 C-5
- C.1.9 Dolby Headphone® 2..... C-5
 - C.1.9.1 Loading Dolby Headphone 2..... C-5
 - C.1.9.2 Loading Dolby Headphone 2 with Dolby ProLogic II C-6
 - C.1.9.3 Removing Dolby Headphone 2 C-6
- C.1.10 DTS-HD™ High Resolution Audio C-6
 - C.1.10.1 Loading DTS-HD High Resolution Audio for Stereo Downmix Output C-6
- C.1.11 DTS-HD™ Master Audio..... C-7
 - C.1.11.1 Loading DTS-HD Master Audio for Stereo Downmix Output C-7
- C.1.12 Crossbar (Downmix and Upmix) C-7
 - C.1.12.1 Loading Crossbar with Legacy and PCM Modules C-7
 - C.1.12.2 Loading Crossbar for Dual Zone Output with Logic 7 and HD Decoders... C-7
- C.1.13 Intelligent Room Calibration 2 (IRC2) C-8
 - C.1.13.1 Configuring the DSP for IRC2 C-8

Revision History C-9

Figures

- Figure P-1. CS4970x4 Chip Functional Block Diagram P-2
- Figure P-2. CS4953x4 Chip Functional Block Diagram P-3
- Figure P-3. PLL Filter Topology P-9
- Figure P-4. Crystal Oscillator Circuit Diagram P-10
- Figure 1-1. Operation Mode Block Diagram 1-1
- Figure 1-2. CS497004, LQFP 144-Pin Package, SPI Control, Master Boot Typical Connection Diagram ..1-4
- Figure 1-3. CS497004/CS4963x4, LQFP 128-Pin Package, SPI Control, Master Boot Typical Connection Diagram 1-5
- Figure 1-4. Master Boot Flow 1-7
- Figure 2-1. SPI Serial Control Port Internal Block Diagram 2-2
- Figure 2-2. Block Diagram of SPI System Bus 2-3
- Figure 2-3. SPI Write Flow Diagram 2-5
- Figure 2-4. SPI Read Flow Diagram 2-6
- Figure 2-5. Sample Waveform for SPI Write Functional Timing 2-8
- Figure 2-6. Sample Waveform for SPI Read Functional Timing 2-8

Figure 2-7. Serial Control Port Internal Block Diagram	2-9
Figure 2-8. Block Diagram of I2C System Bus	2-10
Figure 2-9. I2C Start and Stop Conditions	2-11
Figure 2-10. I2C Address with ACK and NACK	2-12
Figure 2-11. Data Byte with ACK and NACK	2-13
Figure 2-12. Repeated Start Condition with ACK and NACK	2-13
Figure 2-13. Stop Condition with ACK and NACK	2-14
Figure 2-14. I2C Write Flow Diagram	2-15
Figure 2-15. I2C Read Flow Diagram	2-17
Figure 2-16. Sample Waveform for I2C Write Functional Timing	2-19
Figure 2-17. Sample Waveform for I2C Read Functional Timing	2-19
Figure 3-1. DAI Port Block Diagram	3-3
Figure 3-2. Left-justified Format (Rising Edge Valid SCLK)	3-4
Figure 3-3. DSD Port Block Diagram	3-8
Figure 4-1. DAO Block Diagram	4-2
Figure 4-2. I2S Compatible Serial Audio Formats (Rising Edge Valid	4-3
Figure 4-3. Left-justified Digital Audio Formats (Rising Edge Valid DAO_SCLK)	4-3
Figure 4-4. One-line Data Mode Digital Audio Formats	4-4
Figure 5-1. SDRAM Interface Block Diagram	5-1
Figure 8-1. DSP Condenser-produced Control Code	8-1
Figure 8-2. DSP Condenser Wizard, General Page	8-4
Figure 8-3. DSP Condenser Wizard, Search paths Page	8-5
Figure 8-4. DSP Condenser Wizard, Audio sources Page	8-6
Figure 8-5. DSP Condenser Wizard, Sample Rates Page	8-7
Figure 8-6. DSP Condenser Wizard, Firmware components Page	8-8
Figure 8-7. DSP Condenser Wizard, PPM modes Page	8-9
Figure 8-8. DSP Condenser Wizard, Stream types Page	8-10
Figure 8-9. DSP Condenser Wizard, Power-up state Page	8-11
Figure 8-10. DSP Condenser Wizard, WAV update Page	8-12
Figure 8-11. DSP Condenser Project Flash Memory Organization	8-16
Figure 8-12. DSP Composer Sample Project, "WhizBang Model" Directory Structure:	8-18
Figure 8-13. DSP Composer Snapshot	8-19
Figure 8-14. Sample Deliverables Directory Struction	8-20
Figure 8-15. Blank SPI Flash Format	8-21
Figure 8-16. Flash Image Creation Process Flow	8-22
Figure 8-17. New Project Window in DSP Condenser	8-23
Figure 8-18. DSP Condenser Flash Image Build Log	8-24

Figure 8-19. Displaying CDM Window	8-25
Figure 8-20. DSP Response after Successful Master Boot	8-27
Figure 8-21. Changing Concurrency Modes	8-28
Figure 9-1. DSP Condenser Wizard Menu Items	9-1
Figure 9-2. Flash Image Build Log	9-1
Figure 9-3. Program Flash on Board	9-2
Figure 9-4. Run Runtime GUI	9-2
Figure 9-5. DSP API Manager Input Source	9-2
Figure 9-6. Command Line Launch of DSP Condenser without Image	9-3
Figure 9-7. DSP Condenser Runtime with DSP Image XML Provided	9-4
Figure 9-8. Connection Group	9-4
Figure 9-9. Command Group	9-5
Figure 9-10. DSP Status Group	9-6
Figure 9-11. DSP Condenser Runtime Source Selection Group	9-7
Figure 9-12. DSP Condenser Runtime Source Status Group	9-7
Figure 9-13. DSP Manager API Group with Image Information	9-9
Figure 9-14. DSP Manager API Group without Image Information	9-9
Figure A-1. Out-of-Date DSP Composer Notice	A-2

Tables

Table P-1. Core Supply Pins	P-7
Table P-2. I/O Supply Pins	P-7
Table P-3. Core and I/O Ground Pins	P-8
Table P-4. PLL Supply Pins	P-8
Table P-5. PLL Filter Pins	P-9
Table P-6. Reference PLL Component Values	P-9
Table P-7. DSP Core Clock Pins	P-10
Table P-8. Reset Pin	P-11
Table P-9. Hardware Strap Pins	P-11
Table P-10. CS4970x4 Pin Assignments for 144-Pin and 128-Pin Packages	P-12
Table P-11. CS4953x4 Pin Assignments for 144-Pin and 128-Pin Packages	P-19
Table 1-1. Operation Modes	1-2
Table 1-2. Supported SPI Flash Read Format	1-3
Table 1-3. Boot Read Messages from CS4953x4/CS4970x4	1-8
Table 2-1. Serial Control Port SPI Signals	2-2
Table 2-2. Serial Control Port 1 I2C Signals	2-10
Table 3-1. Digital Audio Input Port	3-1

Table 3-2. Bursty Data Input (BDI) Pins	3-3
Table 3-3. Input Data Format Configuration (Input Parameter A)	3-5
Table 3-4. Input SCLK Polarity Configuration (Input Parameter B)	3-5
Table 3-5. Input LRCLK Polarity Configuration (Input Parameter C)	3-6
Table 3-6. Input DAI Mode Configuration (Input Parameter D)	3-6
Table 3-7. DSDI Audio Input Port	3-7
Table 4-1. Digital Audio Output (DAO1 & DAO2) Pins	4-1
Table 4-2. Output Clock Mode Configuration (Parameter A)	4-5
Table 4-3. DAO1 & DAO2 Clocking Relationship Configuration (Parameter B)	4-5
Table 4-4. Output DAO_SCLK/LRCLK Configuration (Parameter C)	4-5
Table 4-5. Output Data Format Configuration (Parameter D)	4-8
Table 4-6. Output DAO_LRCLK Polarity Configuration (Parameter E)	4-9
Table 4-7. Output DAO_SCLK Polarity Configuration (Parameter F)	4-9
Table 4-8. Output Channel Configuration (Parameter G)	4-9
Table 4-9. S/PDIF Transmitter Pins	4-10
Table 4-10. S/PDIF Transmitter Configuration	4-10
Table 4-11. DSP Bypass Configuration	4-11
Table 5-1. SDRAM Interface Signals	5-2
Table 5-2. SDRAM Interface Parameters	5-4
Table 7-2. DSP_LAST_ACCN_MSG Messages	7-5
Table 7-1. DSP_AUTODETECT_MSG Messages	7-5
Table 7-3. Microcontroller Interface API	7-7
Table 7-4. DSP_CFG_xxx Firmware Configuration Registers	7-7
Table 7-5. Firmware Status Registers	7-12
Table 7-6. Legacy Audio Manager	7-14
Table 7-7. OS Module Variables	7-17
Table 9-1. Translation from Input Sources to Board Configuration Values	9-7

P.1 Introduction to CS4953x4/CS4970x4 System Designer's Guide

This design guide contains development guidelines for customers using the Cirrus Logic CS4953x4/CS4970x4 DSP chip-set. All information needed to create a DSP application are presented in this consolidated documentation set. This manual describes a simpler development path than was previously available from Cirrus Logic or its competitors.

Note: This manual will be updated frequently as enhancements are added to the Cirrus Logic DSP system design tools documented here.

This new development path described here has the following benefits:

- Provides a methodology/toolset to help customers design, develop, update, and upgrade a product to achieve shorter design-to-market results than previously available.
- Increased intelligence built into the Cirrus Logic CS4953x4/CS4970x4 DSP allows the DSP to take on most DSP-related control functionality that had previously been exercised by the system microcontroller.
- Replaces complex customer-written control code in the system microcontroller with C source code provided by Cirrus Logic.
- Generates one composite Flash image containing both configuration and DSP codes that customers use on their systems.
- Allows the customer to choose product features and simplifies the interface between the microcontroller and the Cirrus Logic DSP.

P.2 Overview of the CS4953x4/CS4970x4 DSP

The CS4953x4/CS4970x4 are programmable audio DSPs that combines a programmable, 32-bit fixed-point general purpose DSP with dedicated audio peripherals. Its audio-centric interfaces facilitate the coding of high-precision audio applications and provide a seamless connection to external audio peripheral ICs.

The CS4953x4/CS4970x4 is a 32-bit RAM-based processor that provides up to 150 MIPS of processing power and includes all standard codes in ROM. It has been designed with a generous amount of on-chip program and data RAM, and has all necessary peripherals required to support the latest standards in consumer entertainment products. In addition, external SDRAM memory interface can be used to expand the data memory. This device is suitable for a variety of high-performance audio applications. These include:

- Audio/Video Receivers
- DVD Receivers
- Stereo TVs
- Car Audio Head Units and Amplifiers
- Mini Systems
- Shelf Systems
- Digital Speakers
- Set-top Boxes

P.2.1 Chip Features

The CS4953x4/CS4970x4 include the following features:

- Various Decoding/processing Standards
- 12-channel Serial Audio Inputs
- Dual 32-bit Audio DSP with Dual MAC
- Large On-chip X,Y, and Program RAM
- Supports SDRAM Memory
- Parallel Control Using Motorola® or Intel® Communication Standards
- Customer Software Security Keys
- 16-channel PCM Output
- Dual S/PDIF Transmitters
- Two Serial Control Ports Using SPI™ or I²C™ Standards
- Digital Audio Input (DAI) Port for Audio Data Delivery in I²S or LJ Format
- GPIO Support for All Common Sub-circuits

Figure P-1 illustrates the functional block diagram for the CS4953x4/CS4970x4 DSPs.

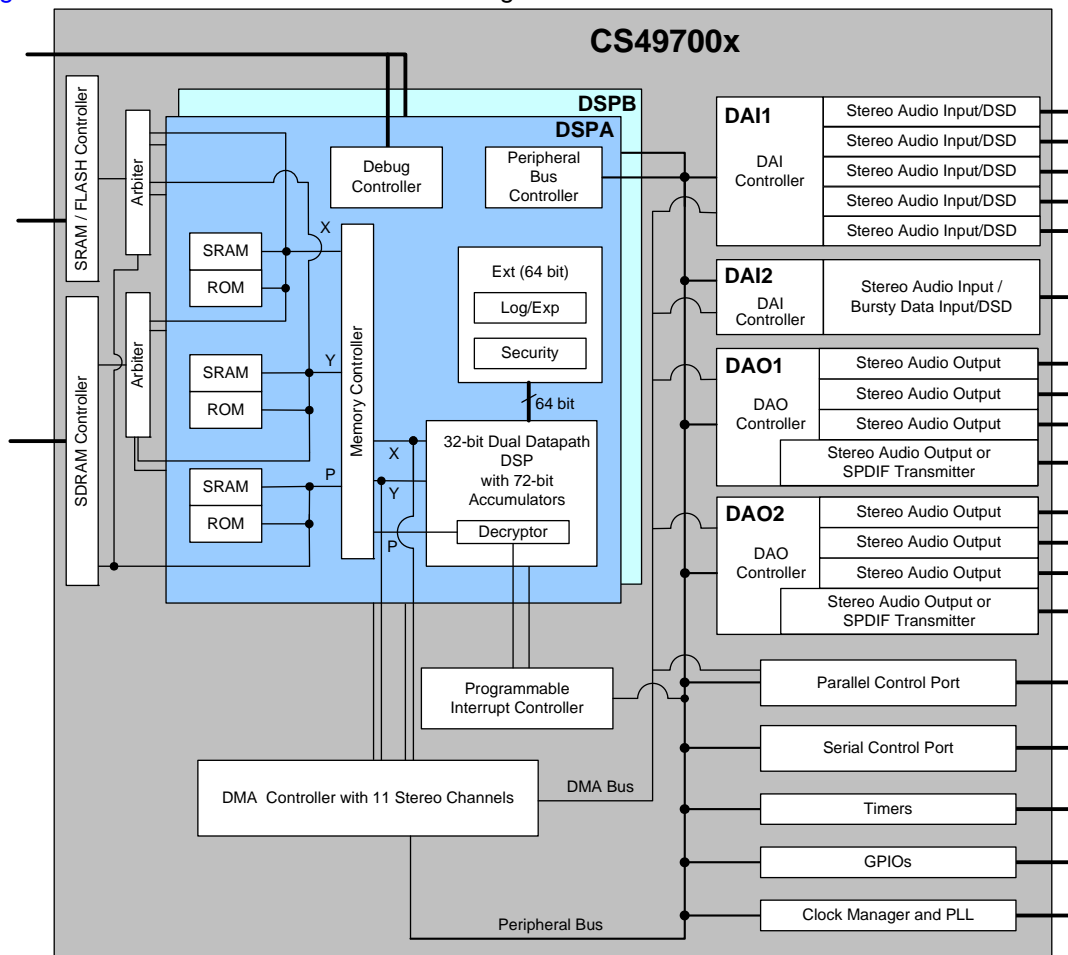


Figure P-1. CS4970x4 Chip Functional Block Diagram

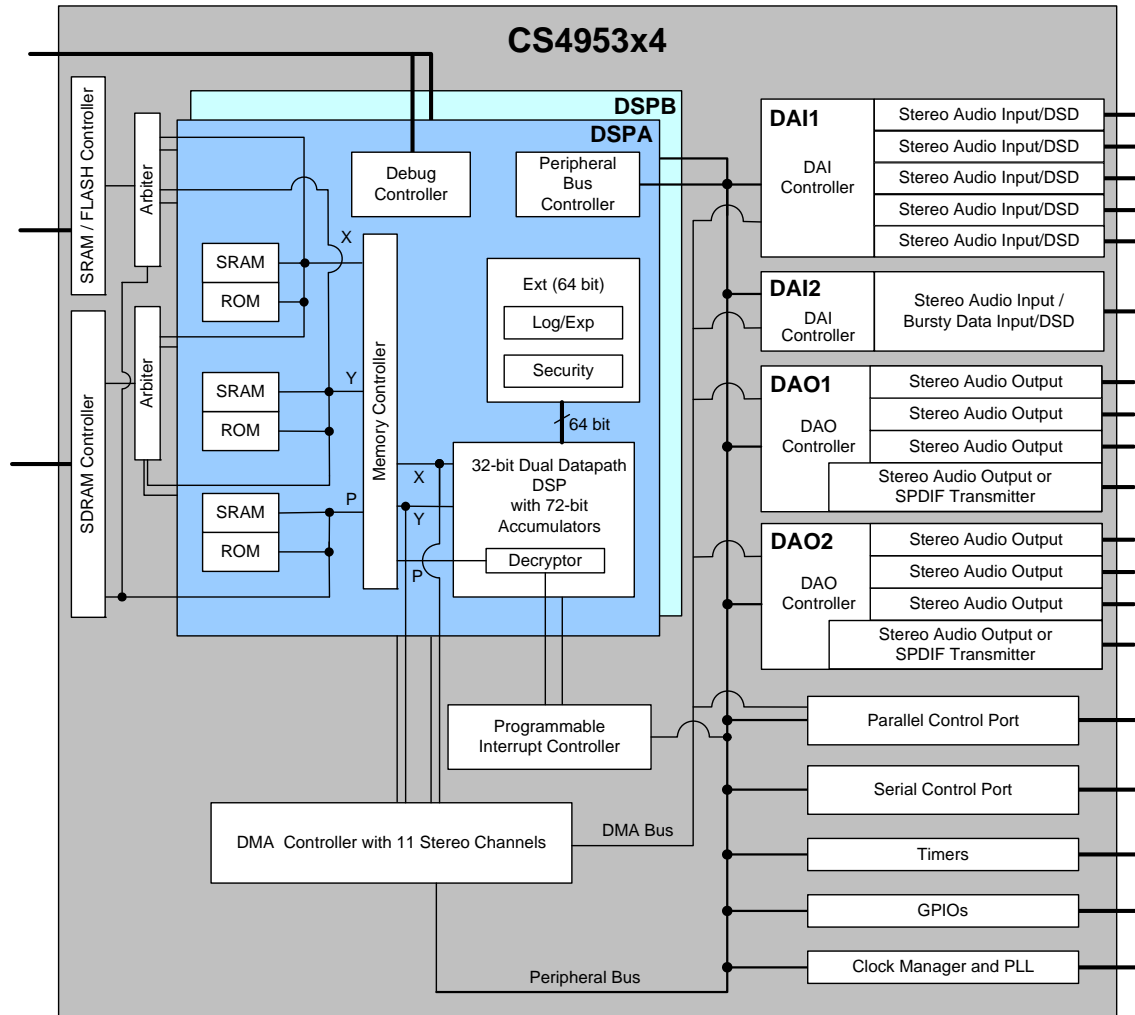


Figure P-2. CS4953x4 Chip Functional Block Diagram

See AN288 for a list of decoders, post-processors, virtualizers that CS4953x4/CS4970x4 DSPs support. HD Decoder are only available on the CS4970x4 DSPs.

Audio decoding/processing algorithms, post-processing application codes, and/or Cirrus Framework™ modules and the associated application notes are available through the Cirrus Software Licensing Program. Standard post-processing code modules are only available to customers who qualify for the *Cirrus Framework™ CS4953x4/CS4970x4 Family DSP Programming Kit*. Please refer to the *Related Documents* section of the Framework™ manual for information on additional documentation support.

The CS4953x4/CS4970x4 DSP contains sufficient on-chip memory to support decoding of all major audio decoding algorithms available today. The CS4953x4/CS4970x4 DSP supports a glueless SDRAM interface for increased all-channel delays. The memory interface also supports connection to an external SPI Flash for code storage, thus allowing products to be field upgraded as new audio algorithms are developed.

This chip, teamed with the Cirrus Logic certified decoder library, Cirrus Logic digital interface products, and mixed-signal data converters, enables the design of next-generation digital entertainment products.

P.3 CS4953x4/CS4970x4 Chip Functional Overview

The CS4953x4/CS4970x4 chip support a maximum clock speed of 150 MHz in a 144-pin LQFP or 128-pin LQFP package. A high-level functional description of the CS4953x4/CS4970x4 chip is provided in this section.

P.3.1 DSP Core

The DSP core for the CS4953x4/CS4970x4 DSP is a pair of general purpose, 32-bit, fixed-point, fully programmable digital signal processors that achieve high performance through an efficient instruction set and highly parallel architecture. The device uses two's complement fractional number representation, and employs busses for two data memory spaces and one program memory space.

CS4953x4/CS4970x4 core enhancements include portability of the design, speed improvement, and improvements for synthesis, verification, and testability.

P.3.2 Security Extension module

This module is a 64-bit extension module that allows the CS4953x4/CS4970x4 devices to be placed in a secure mode where decryption is activated via a 128-bit key. This key is written to the security extension in two 64-bit move instructions. Secure mode is disabled by default, and must be explicitly enabled.

P.3.3 Debug Controller (DBC)

An I²C Slave debug controller (DBC) is integrated within the CS4953x4/CS4970x4 DSP core. Two pins are reserved for connecting a PC host to the debug ports on either DSP. The debug port consists of two modules, an I²C Slave and a debug Master. The DBC Master sends dedicated signals into the DSP core to initiate debug actions and it receives acknowledge signals from the core to indicate the requested action has been taken. Basically, this interface allows the DBC to insert instructions into the pipeline. The core will acknowledge the action when it determines the pipeline is in the appropriate state for the inserted action to be taken.

P.3.4 Digital Audio Output (DAO1, DAO2) Controller

The CS4953x4/CS4970x4 family DSPs each have two Digital Audio Output (DAO) controllers, each of which contains 4 stereo output ports. One port on each DAO can be used as a S/PDIF transmitter. The DAO ports can transmit up to 16 channels of audio sample data in I²S-compatible format. The audio samples are stored in 16 channel buffers which are 32 bits wide. Four of the channels can also serve as output buffers for the two S/PDIF transmitters. The O/S can dedicate DMA channels to fill the DAO data buffers from memory. DAO control is handled through the peripheral bus.

P.3.5 Digital Audio Input (DAI1) Controller

The Digital Audio Input (DAI) controller for the CS4953x4/CS4970x4 has four stereo input ports and DAI control is handled through the peripheral bus. Each DAI pin can be configured to load audio samples in a variety of formats. In addition to accepting multiple formats, the DAI controller has the ability to accept multiple stereo channels on a single DAI1_DATAx pin. All four DAI pins are Slaves and normally share the same serial input clock pins (DAI1_SCLK and DAI1_LRCLK). Pins DAI1_DATA[3:0] may also be reconfigured to use the DAO serial input clock pins (DAOx_SCLK and DAOx_LRCLK). A single global configuration register provides a set of enable bits to ensure that ports may be started synchronously.

P.3.6 Compressed Data Input / Digital Audio Input (DAI2) Controller

The DAI2 controller has one input port and its own SCLK and LRCLK and can be used for accepting PCM data in the same way as DAI1, but is used primarily for the delivery of compressed data. When configured

for compressed data input, custom internal hardware is enabled that off-loads some pre-processing of the incoming stream to help maximize the MIPS available in the DSP core for user-customized applications.

P.3.7 Direct Stream Digital (DSD) Controller

The DSD controller for the CS4953x4/CS4970x4 also has a DSD controller which allows the DSP to be integrated into a system that supports SACD audio. The DSD controller pins are shared with the DAI1 and DAI2 ports. The DSD port consists of a bit clock (DSD_CLK) and six DSD data inputs (DSD[5:0]).

P.3.8 General Purpose I/O

A 32-bit general-purpose I/O (GPIO) port is provided on the CS4953x4/CS4970x4 DSPs to enhance system flexibility. Many of the functional pins can be used for either GPIO or peripherals.

Each GPIO pin can be individually configured as an output, an input, or an input with interrupt. A GPIO interrupt can be triggered on a rising edge (0-to-1 transition), falling edge (1-to-0 transition), or logic level (either 0 or 1). Each pin configured as an input with interrupt can be assigned its own interrupt trigger condition. All GPIOs share a common interrupt vector.

P.3.9 Serial Control Ports (SPI™ or I²C™ Standards)

The CS4953x4/CS4970x4 have two serial control ports (SCP) that support SPI™ and I²C™ Master/Slave communication modes. The serial control port allows external devices such as microcontrollers to communicate with the CS4953x4/CS4970x4 chips through either I²C or SPI serial communication standards and can be configured as either a Master or a Slave.

The CS4953x4/CS4970x4 SPI and I²C serial communication modes are identical from a functional standpoint. The main difference between the two is the protocol being implemented between the CS4953x4/CS4970x4 and the external device. In addition, the I²C Slave has a true I²C mode that utilizes data flow mechanisms inherent to the I²C protocol. If this mode is enabled, the I²C Slave will hold SCP1_CLK low to delay a transfer as needed.

By default, SCP1 is configured as a Slave for external device-controlled data transfers. As a Slave, it cannot drive the clock signal nor initiate data transfers.

By default, SCP2 is configured as a Master to access a SPI Flash for either booting the DSP or retrieving configuration information. As a Master, it can drive the clock signal at up to 1/2 of the DSP's core clock speed.

The CS4953x4/CS4970x4 has two additional serial communication pins not specified in either the I²C or SPI specification. The port uses the $\overline{\text{SCP1_IRQ}}$ pin to indicate that a read message is ready for the host. The port uses the $\overline{\text{SCP1_BSY}}$ pin to warn the host to pause communication.

P.3.10 SDRAM Controller

The CS4953x4/CS4970x4 supports a glueless external SDRAM interface to extend the data memory of the DSP during runtime. The SDRAM controller provides 2-port access to X and Y memory space, a quad-word read buffer, and a double-buffered quad-word write buffer. One SDRAM controller port is dedicated to P memory space and the second port is shared by X and Y memories. The X/Y port has dual write buffers and a single read buffer, and the P memory port has a single read buffer. One of these buffers is four 32-bit words (128 bits). Every "miss" to the read buffer will cause the SDRAM controller to burst eight 16-bit reads on the SDRAM interface. The SDRAM controller supports SDRAMs from 2 MB to 64 MB with various row, bank, and column configurations. The SDRAM controller runs synchronous to the DSP core clock, which is the global chip clock.

P.3.11 DMA Controller

The DMA controller contains 12 stereo channels. The O/S uses 11 stereo channels, 6 for the DAO (2 are for the S/PDIF transmitters), 4 for the DAI, and one for the parallel control port. The addition of the DMA channel for the parallel control port allows compressed audio data to be input over this port. The DMA block is able to move data to/from X or Y memory, or alternate between both X and Y memory. The DMA controller moves data to/from X and/or Y memory opportunistically (if the core is not currently accessing that particular memory space during the current cycle). The DMA controller has a “Dead Man’s” timer so that if the core is running an inner loop and accessing memory every cycle, the DMA controller can interrupt the core to run a DMA cycle.

P.3.12 Timers

A 32-bit timer block runs off the CS4953x4/CS4970x4 DSP clock. The timer count decrements with each clock tick of the DSP clock when the timer is enabled. When the timer count reaches zero, it is re-initialized, and may be programmed to generate an interrupt to the DSP.

P.3.13 Clock Manager and PLL

The CS4953x4/CS4970x4 Clock Manager and PLL module contains an Analog PLL, RTL Clock Synthesizer, and Clock Manager. The Analog PLL is a customized analog hard macro that contains the Phase Detector (PD), Charge Pump, Loop Filter, VCO, and other non-digital PLL logic. The Clock Synthesizer is a digital design wrapper around the analog PLL that allows clock frequency ranges to be programmed. The Clock Manager is a digital design wrapper for the Clock Synthesizer that provides the logic (control registers) necessary to meet chip clocking requirements.

The Clock Manager and PLL module generates two Master clocks:

- Global chip clock (clocks the DSP core, internal memories, SDRAM, Flash, and all peripherals)
- Oversampled audio clock. This clock feeds the DAO block which has dividers to generate the DAO_MCLK, DAO_SCLK, and DAO_LRCLK.

P.3.14 Programmable Interrupt Controller

The Programmable Interrupt Controller (PIC) forces all incoming interrupts to be synchronized to the global clock, HCLK. The PIC provides up to 16 interrupts to the DSP Core. The interrupts are prioritized with interrupt 0 as the highest priority and interrupt 15 as the lowest priority. Each interrupt has a corresponding interrupt address that is also supplied to the DSP core. The interrupt address is the same as the IRQ number (interrupt 0 uses interrupt address 0 and interrupt 15 uses interrupt address 15). Both an enable mask and a run mask are provided for each interrupt. The enable mask allows the enabled interrupts to generate a PIC_REQ signal to the DSP core, and the run mask allows the enabled interrupts to generate a PIC_CLR, thereby bringing the core out of its halt state when it accepts the interrupt.

P.4 Firmware Overview

P.4.1 Robust DSP Manager Firmware API

Cirrus Logic includes a robust firmware API to allow customers to develop DSP applications with a smaller microcontroller code set than was necessary in previous Cirrus Logic DSP products. See [Chapter 7, "Overview of Common Firmware Modules"](#) for information about the new DSP Manager firmware API from Cirrus Logic.

P.4.2 DSP Condenser

Cirrus Logic provides the customer with the DSP Condenser application to implement the design capabilities described in [Section P.1](#). The DSP Condenser application is the vehicle that allows system designers to quickly program the CS4953x4/CS4970x4 DSP with the customer's design and to access Cirrus Logic powerful firmware suite.

See additional information about the DSP Condenser application in [Chapter 8, "DSP Condenser"](#).

P.5 CS40700x Pin Descriptions

P.5.1 Power and Ground

The following sections describe the CS4953x4/CS4970x4 power and ground pins. Decoupling and conditioning of the power supplies is also discussed. Following the recommendations for decoupling and power conditioning will help to ensure reliable performance.

P.5.1.1 Power

The CS4953x4/CS4970x4 Family of DSPs take two supply voltages — the core supply voltage (VDD) and the I/O supply voltage (VDDIO). There is also a separate analog supply voltage required for the internal PLL (VDDA). These pins are described in the following tables and descriptions.

The DSP Core supply voltage pins require a nominal 1.8V. The DSP I/O supply voltage pins require a nominal 3.3V.

Table P-1. Core Supply Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
10	42	VDD1	Input	1.8V DSP Core supply. This powers all internal logic and the on-chip SRAMs and ROMs
24	55	VDD2		
54	83	VDD3		
66	95	VDD4		
83	112	VDD5		
98	125	VDD6		
119	12	VDD7		
130	22	VDD8		

Table P-2. I/O Supply Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
18	50	VDDIO1	Input	3.3V I/O supply
33	62	VDDIO2		
44	73	VDDIO3		
60	89	VDDIO4		
73	100	VDDIO5		
91	120	VDDIO6		
113	8	VDDIO7		
136	28	VDDIO8		

P.5.1.2 Ground

For two-layer circuit boards, care should be taken to have sufficient grounding between the DSP and parts in which it will be interfacing (DACs, ADCs, S/PDIF Receivers, microcontrollers, and especially external memory). Insufficient grounding can degrade noise margins between devices resulting in data integrity problems.

Table P-3. Core and I/O Ground Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
13	45	GND1	Input	Core Ground.
27	56	GND2		
57	86	GND3		
69	98	GND4		
86	115	GND5		
101	127	GND6		
122	15	GND7		
133	25	GND8		
21	53	GNDIO1		I/O Ground
36	67	GNDIO2		
47	76	GNDIO3		
63	92	GNDIO4		
76	105	GNDIO5		
94	122	GNDIO6		
116	9	GNDIO7		
139	31	GNDIO8		

P.5.1.3 Decoupling

It is necessary to decouple the power supply by placing capacitors directly between the power and ground of the CS4953x4/CS4970x4. Each pair of power/ground pins (VDD1/GND1, etc.) should have its own decoupling capacitor. The recommended procedure is to place a 0.1 μ F capacitor as close as physically possible to each power pin connected with a wide, low-inductance trace. A bulk capacitor of at least 10 μ F is recommended for each power plane.

P.5.2 PLL Filter

P.5.2.1 Analog Power Conditioning

In order to obtain the best performance from the CS4953x4/CS4970x4's internal PLL, the analog power supply VDDA must be as noise free as possible. A ferrite bead and two capacitors should be used to filter the VDDIO to generate VDDA. This power scheme is shown in the *Typical Connection* diagrams.

Table P-4. PLL Supply Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
129	21	VDDA	Input	PLL supply. This voltage must be 3.3V. This must be clean, noise-free analog power.
126	19	GNDA	Input	PLL ground. This ground should be as noise free as possible.

P.5.2.2 PLL

The internal phase locked loop (PLL) of the CS4953x4/CS4970x4 requires an external current reference resistor. The resistor is used to calibrate the PLL and must meet the tolerances specified below. The layout topology is shown in the typical connection diagrams. Care should be taken when laying out the current sense circuitry to minimize trace lengths between the DSP and resistor, and to keep high-frequency signals away from the resistor. Any noise coupled onto these traces will be directly coupled into the PLL, which could affect performance. Please see tables below for pin numbers and external component values.

Table P-5. PLL Filter Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
128	20	PLL_REF_RES	Input	Current Reference Resistor for PLL filter

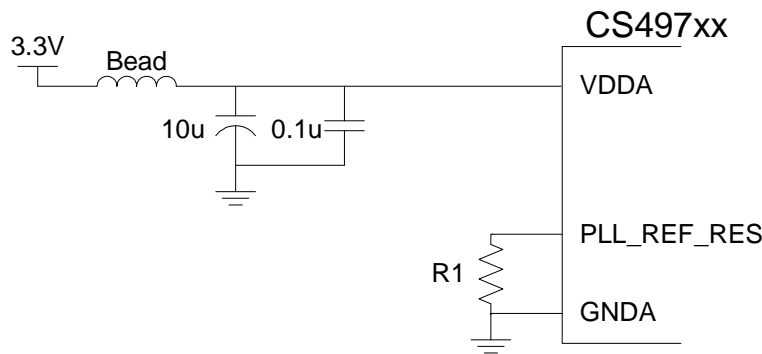


Figure P-3. PLL Filter Topology

Table P-6. Reference PLL Component Values

Symbol	Reference Value
R1	5.1 k Ω , 1%

P.5.3 Clocking

The CS4953x4/CS4970x4 incorporates a programmable phase locked loop (PLL) clock synthesizer. The PLL takes an input reference clock and produces all the clocks required to run the DSP and peripherals.

In A/V Receiver designs that require low-jitter clocks, the XTI pin is typically connected to an external 12.288 MHz or 24.576 MHz (recommended) oscillator that is used throughout the system.

The CS4953x4/CS4970x4 has a built-in crystal oscillator circuit. A parallel resonant-type crystal is connected between the XTI and XTO pins as shown in [Figure P-4](#). The value of C1 is specific to each crystal. The CS4953x4/CS4970x4 data sheet specifies acceptable crystal parameters (including C_L and ESR). When a crystal is used, XTAL_OUT is used to clock other devices in the system such as the S/PDIF receiver.

The PLL is controlled by the clock manager in the DSP O/S application software. AN288, "CS4953xx/CS4953x4/CS4970x4 Firmware User's Manual" should be referenced regarding what CLKIN input frequency and PLL multiplier values are supported.

Table P-7. DSP Core Clock Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
123	16	XTAL_OUT	Output	Buffered version of XTI.
124	17	XTI	Input	Reference Clock Input/Crystal Oscillator Input. An external clock may be input directly to this pin or one end of a crystal may be connected to this pin.
125	18	XTO	Output	Crystal Oscillator Output. One end of a crystal oscillator is connected to this pin. This pin cannot be used to drive external circuitry.

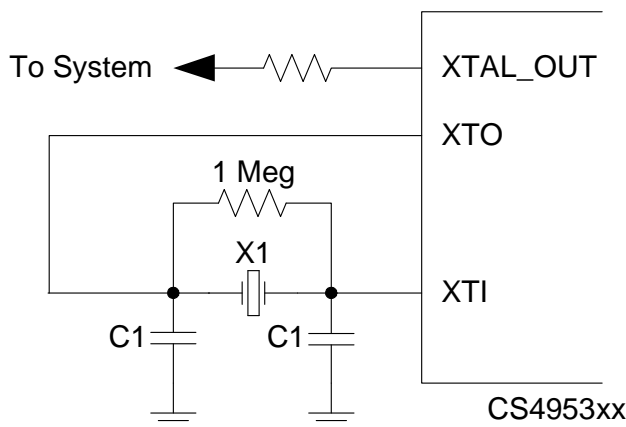


Figure P-4. Crystal Oscillator Circuit Diagram

P.5.4 Control

The CS4953x4/CS4970x4 supports 5 control interface protocols: SPI, I²C, Motorola parallel, Intel parallel, and Multiplexed Intel parallel mode. For modes other than SPI, contact your Cirrus Logic FAE or representative. All slave serial control modes between the DSP and the host microcontroller use the Serial Control Port 1 (SCP1) pins. Parallel slave control modes are implemented on the Parallel Control Port (PCP) pins. A second serial control port (SCP2) is available for master mode applications.

P.5.4.1 Operational Mode

The control interface protocol used is determined by the state of the Hardware Strap pins, HS[4:0] which are sampled at the rising edge of RESET. The HS[4:0] pins should be pulled to VDD or GND using 10 kΩ resistors according to the specific control mode desired as shown in [Table 1-1, "Operation Modes" on page 1-2](#).

The following sections describe the pins used for the 5 control modes. For an example diagram of a system connection, please see [Figure 1-2 on page 1-4](#). For information on timing diagrams and messaging protocol to the CS4953x4/CS4970x4, see [Chapter 1, "Operational Modes"](#).

Configuration and control of the CS4953x4/CS4970x4 decoder and its peripherals are indirectly executed through a messaging protocol supported by the operating system (OS) running on the DSP. In other words, successful communication can only be accomplished by following the low-level hardware communication format and high-level messaging protocol. The specifications of the messaging protocol used by the OS can be found in AN288, "CS4953x4/CS4970x4 Firmware User's Manual". The system designer only needs to read the subsection describing the communication mode being used. Other chapters in this manual explain each communication mode in more detail.

Table P-8. Reset Pin

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
93	121	RESET	Input	Reset, async. active-low Chip Reset Reset should be low at power-up to initialize the DSP and to guarantee that the device is not active during initial power-on stabilization periods.

Table P-9. Hardware Strap Pins

LQFP-144 Pin #	LQFP-128 Pin #	Pin Name	Pin Type	Pin Description
7	39	DAO2_DATA1, HS4 , GPIO19	Input (at Reset) Output under Normal Operation	Operational Mode Select Pull-up or Pull-down resistors on these pins set the DSP operational mode at reset. Hardware Strap Mode Select <u>The state of these pins is latched at the rising edge of RESET.</u> The boot ROM uses the state of these pins to select the boot mode.
11	43	DAO2_DATA0, HS3 , GPIO18		
16	48	DAO1_DATA2, HS2 , GPIO16		
17	49	DAO1_DATA1, HS1 , GPIO15		
19	51	DAO1_DATA0, HS0		

P.6 CS4970x4 Pin Assignments

Table P-10 shows the pin assignments for both the 144- and 128-pin packages.



Table P-10. CS4970x4 Pin Assignments for 144-Pin and 128-Pin Packages

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
1	-	GPIO28	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN
2	-	GPIO29	General Purpose Input/Output	1. XMTA_IN	1. S/PDIF Pass-thru Input.	3.3V (5V tol)	BiDir	IN
3	36	DBDA	Debug Data			3.3V (5V tol)	In/OD	IN
4	37	DBCK	Debug Clock			3.3V (5V tol)	In/OD	IN
5	-	GPIO21	General Purpose Input/Output	1. DAO2_DATA3 2. XMTB 3. UART_TX_ENABLE	1. Digital Audio Output 3. 2. Outputs IEC60958/61937 format bi-phase mark encoded S/PDIF data. 3. Enable the UART_TX pin.	3.3V (5V tol)	BiDir	IN
6	38	GPIO20	General Purpose Input/Output	1. DAO2_DATA2	1. Digital Audio Output 2. 2. EEPROM Boot Chip Select.	3.3V (5V tol)	BiDir	IN
7	39	GPIO19	General Purpose Input/Output	1. DAO2_DATA1 2. HS4	1. Digital Audio Output 1. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN
8	40	DAO_MCLK	Audio Master Clock			3.3V (5V tol)	BiDir	IN
9	41	TEST	Test			3.3V (5V tol)	In	
10	42	VDD1	Core power supply voltage			1.8V	PWR	
11	43	GPIO18	General Purpose Input/Output	1. DAO2_DATA0 2. HS3	1. Digital Audio Output 0. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN
12	44	GPIO22	General Purpose Input/Output	DAO2_SCLK	PCM Audio Bit Clock.	3.3V (5V tol)	BiDir	IN
13	45	GNDD1	Core ground			0V	PWR	
14	46	GPIO23	General Purpose Input/Output	DAO2_LRCLK	Serial PCM Audio Sample Rate Clock for the serial data pins: (DAO2_DATA0, DAO2_DATA1, DAO2_DATA2, DAO2_DATA3).	3.3V (5V tol)	BiDir	IN
15	47	GPIO17	General Purpose Input/Output	1. DAO1_DATA3 2. XMTA	1. Digital Audio Output 3. 2. S/PDIF Audio Output A.	3.3V (5V tol)	BiDir	IN
16	48	GPIO16	General Purpose Input/Output	1. DAO1_DATA2 2. HS2	1. Digital Audio Output 2. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN
17	49	GPIO15	General Purpose Input/Output	1. DAO1_DATA1 2. HS1	1. Digital Audio Output 1. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN
18	50	VDDIO1	I/O power supply voltage			3.3V	PWR	
19	51	DAO1_DATA0	Digital Audio Output 0	HS0	Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN
20	52	DAO1_SCLK	PCM Audio Bit Clock			3.3V (5V tol)	BiDir	IN
21	53	GNDIO1	I/O ground			0V	PWR	
22	54	DAO1_LRCLK	PCM Audio Sample Rate Clock			3.3V (5V tol)	BiDir	IN
23	-	GPIO31	General Purpose Input/Output	UART_CLK	UART Clock.	3.3V (5V tol)	BiDir	IN
24	55	VDD2	Core power supply voltage			1.8V	PWR	



Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
25	-	GPIO25	General Purpose Input/Output	1. UART_TXD 2. EE_CS	1. UART Output. 2. EEPROM Boot Chip Select.	3.3V (5V tol)	BiDir	IN
26	-	GPIO24	General Purpose Input/Output	UART_RXD	UART Input.	3.3V (5V tol)	BiDir	IN
27	56	GNDD2	Core ground			0V	PWR	
28	57	SD_DQM0	SDRAM Data Mask 0			3.3V (5V tol)	OUT	
29	58	SD_D7	SDRAM Data Bit 7	EXT_D7	Flash Data Bit 7.	3.3V (5V tol)	BiDir	IN
30	59	SD_D6	SDRAM Data Bit 6	EXT_D6	Flash Data Bit 6.	3.3V (5V tol)	BiDir	IN
31	60	SD_D5	SDRAM Data Bit 5	EXT_D5	Flash Data Bit 5.	3.3V (5V tol)	BiDir	IN
32	61	SD_D4	SDRAM Data Bit 4	EXT_D4	Flash Data Bit 4.	3.3V (5V tol)	BiDir	IN
33	62	VDDIO2	I/O power supply voltage			3.3V	PWR	
34	63	SD_D3	SDRAM Data Bit 3	EXT_D3	Flash Data Bit 3.	3.3V (5V tol)	BiDir	IN
35	64	SD_D2	SDRAM Data Bit 2	EXT_D2	Flash Data Bit 2.	3.3V (5V tol)	BiDir	IN
36	67	GNDIO2	I/O ground			0V	PWR	
37	65	SD_D1	SDRAM Data Bit 1	EXT_D1	Flash Data Bit 1.	3.3V (5V tol)	BiDir	IN
38	66	EXT_WE	Flash Write Enable			3.3V (5V tol)	OUT	
39	68	SD_D0	SDRAM Data Bit 0	EXT_D0	Flash Data Bit 0	3.3V (5V tol)	BiDir	IN
40	69	SD_D15	SDRAM Data Bit 15	EXT_D15	Flash Data Bit 15	3.3V (5V tol)	BiDir	IN
41	70	SD_D14	SDRAM Data Bit 14	EXT_D14	Flash Data Bit 14	3.3V (5V tol)	BiDir	IN
42	71	SD_D13	SDRAM Data Bit 13	EXT_D13	Flash Data Bit 13	3.3V (5V tol)	BiDir	IN
43	72	SD_D12	SDRAM Data Bit 12	EXT_D12	Flash Data Bit 12	3.3V (5V tol)	BiDir	IN
44	73	VDDIO3	I/O power supply voltage			3.3V	PWR	
45	74	SD_D11	SDRAM Data Bit 11	EXT_D11	Flash Data Bit 11	3.3V (5V tol)	BiDir	IN
46	75	SD_D10	SDRAM Data Bit 10	EXT_D10	Flash Data Bit 10	3.3V (5V tol)	BiDir	IN
47	76	GNDIO3	I/O ground			0V	PWR	
48	77	SD_D9	SDRAM Data Bit 9	EXT_D9	Flash Data Bit 9	3.3V (5V tol)	BiDir	IN
49	78	SD_D8	SDRAM Data Bit 8	EXT_D8	Flash Data Bit 8	3.3V (5V tol)	BiDir	IN
50	79	SD_DQM1	SDRAM Data Mask 1			3.3V (5V tol)	OUT	
51	80	SD_CLKOUT	SDRAM Clock Output			3.3V (5V tol)	OUT	
52	81	SD_CLKIN	SDRAM Clock Input			3.3V (5V tol)	In	
53	82	SD_CLKEN	SDRAM Clock Enable			3.3V (5V tol)	OUT	
54	83	VDD3	Core power supply voltage			1.8V	PWR	
55	84	SD_A12	SDRAM Address Bit 12	EXT_A12	Flash Address Bit 12	3.3V (5V tol)	OUT	
56	85	SD_A11	SDRAM Address Bit 11	EXT_A11	Flash Address Bit 11	3.3V (5V tol)	OUT	
57	86	GNDD3	Core ground			0V	PWR	
58	87	SD_A9	SDRAM Address Bit 9	EXT_A9	Flash Address Bit 9	3.3V (5V tol)	OUT	
59	88	SD_A8	SDRAM Address Bit 8	EXT_A8	Flash Address Bit 8	3.3V (5V tol)	OUT	



Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
60	89	VDDIO4	I/O power supply voltage			3.3V	PWR	
61	90	SD_A7	SDRAM Address Bit 7	EXT_A7	Flash Address Bit 7	3.3V (5V tol)	OUT	
62	91	SD_A6	SDRAM Address Bit 6	EXT_A6	Flash Address Bit 6	3.3V (5V tol)	OUT	
63	92	GNDIO4	I/O ground			0V	PWR	
64	93	SD_A5	SDRAM Address Bit 5	EXT_A5	Flash Address Bit 5	3.3V (5V tol)	OUT	
65	94	EXT_CS2	Chip Select 2			3.3V (5V tol)	OUT	
66	95	VDD4	Core power supply voltage			1.8V	PWR	
67	96	SD_A4	SDRAM Address Bit 4	EXT_A4	Flash Address Bit 4	3.3V (5V tol)	OUT	
68	97	SD_A3	SDRAM Address Bit 3	EXT_A3	Flash Address Bit 3	3.3V (5V tol)	OUT	
69	98	GNDD4	Core ground			0V	PWR	
70	99	SD_A2	SDRAM Address Bit 2	EXT_A2	Flash Address Bit 2	3.3V (5V tol)	OUT	
71	101	SD_A1	SDRAM Address Bit 1	EXT_A1	Flash Address Bit 1	3.3V (5V tol)	OUT	
72	102	SD_A0	SDRAM Address Bit 0	EXT_A0	Flash Address Bit 0	3.3V (5V tol)	OUT	
73	100	VDDIO5	I/O power supply voltage			3.3V	PWR	
74	103	SD_A10	SDRAM Address Bit 10	EXT_A10	Flash Address Bit 10	3.3V (5V tol)	OUT	
75	104	SD_BA0	SDRAM Bank Address 0	EXT_A13	Flash Address Bit 13	3.3V (5V tol)	OUT	
76	105	GNDIO5	I/O ground			0V	PWR	
77	106	SD_BA1	SDRAM Bank Address 1	EXT_A14	Flash Address Bit 14	3.3V (5V tol)	OUT	
78	107	SD_WE	SDRAM Write Enable			3.3V (5V tol)	OUT	
79	108	SD_CAS	SDRAM Column Address Strobe			3.3V (5V tol)	OUT	
80	109	SD_RAS	SDRAM Row Address Strobe			3.3V (5V tol)	OUT	
81	110	SD_CS	SDRAM Chip Select			3.3V (5V tol)	OUT	
82	111	EXT_A15	Flash Address Bit 15			3.3V (5V tol)	OUT	
83	112	VDD5	Core power supply voltage			1.8V	PWR	
84	113	EXT_A16	Flash Address Bit 16			3.3V (5V tol)	OUT	
85	114	EXT_A17	Flash Address Bit 17			3.3V (5V tol)	OUT	
86	115	GNDD5	Core ground			0V	PWR	
87	116	EXT_A18	Flash Address Bit 18			3.3V (5V tol)	OUT	
88	117	EXT_A19	Flash Address Bit 19			3.3V (5V tol)	OUT	
89	118	EXT_OE	Flash Output Enable			3.3V (5V tol)	OUT	
90	119	EXT_CS1	Active-low Flash chip select			3.3V (5V tol)	OUT	
91	120	VDDIO6	I/O power supply voltage			3.3V	PWR	
92	-	GPIO30	General Purpose Input/Output	XMTB_IN	S/PDIF Pass-thru Input	3.3V (5V tol)	BiDir	IN



Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
93	121	RESET	Chip Reset			3.3V (5V tol)	In	
94	122	GNDIO6	I/O ground			0V	PWR	
95	123	GPIO33	General Purpose Input/Output	SCP1_MOSI	SPI Mode Master Data Output/Slave Data Input	3.3V (5V tol)	BiDir	IN
96	-	GPIO32	General Purpose Input/Output	1. $\overline{\text{SCP1_CS}}$ 2. IOWAIT	1. SPI Chip Select 2. SRAM Hold-Off Handshake	3.3V (5V tol)	BiDir	IN
97	124	GPIO34	General Purpose Input/Output	1. SCP1_MISO 2. SCP1_SDA	1. SPI Mode Master Data Input/Slave Data Output 2. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN
98	125	VDD6	Core power supply voltage			1.8V	PWR	
99	126	GPIO35	General Purpose Input/Output	SCP1_CLK	SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN
100	-	GPIO36	General Purpose Input/Output	SCP1_IRQ	Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir/OD	IN
101	127	GNDD6	Core ground			0V	PWR	
102	-	GPIO37	General Purpose Input/Output	1. $\overline{\text{SCP1_BSY}}$ 2. $\overline{\text{PCP_BSY}}$	1. Serial Control Port 1 Input Busy 2. Parallel Control Port Input Busy	3.3V (5V tol)	BiDir/OD	IN
-	128	GPIO37	General Purpose Input/Output	1. SCP1_BSY	1. Serial Control Port 1 Input Busy	3.3V (5V tol)	BiDir/OD	IN
103	-	GPIO38	General Purpose Input/Output	1. $\overline{\text{PCP_WR}}$ 2. $\overline{\text{PCP_DS}}$ 3. SCP2_CLK	1. Parallel Port Write Select (Intel Mode) 2. Parallel Port Data Strobe (Motorola and Multiplexed Mode) 3. SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN
-	1	GPIO38	General Purpose Input/Output	1. SCP2_CLK	1. SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN
104	-	GPIO39	General Purpose Input/Output	1. $\overline{\text{PCP_CS}}$ 2. SCP2_CS	1. Parallel Port Chip Select (Intel/Motorola/Multiplexed Mode) 2. SPI Chip Select	3.3V (5V tol)	BiDir	IN
105	-	GPIO11	General Purpose Input/Output	1. PCP_A3 2. PCP_AS 3. SCP2_MISO 4. SCP2_SDA	1. Parallel Control Port Address Bit 3 2. Parallel Control Port Address Strobe 3. SPI Mode Master Data Input/Slave Data Output 4. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN
-	2	GPIO11	General Purpose Input/Output	1. SCP2_MISO 2. SCP2_SDA	1. SPI Mode Master Data Input/Slave Data Output 2. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN



Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
106	-	GPIO10	General Purpose Input/Output	1. PCP_A2 2. PCP_A10 3. SCP2_MOSI	1. Parallel Control Port Address Bit 2 2. Parallel Control Port Address Bit 10 3. SPI Mode Master Data Output/ Slave Data Input	3.3V (5V tol)	BiDir	IN
-	3	GPIO10	General Purpose Input/Output	1. SCP2_MOSI	1. SPI Mode Master Data Output/ Slave Data Input	3.3V (5V tol)	BiDir	IN
107	-	GPIO40	General Purpose Input/Output	1. $\overline{\text{PCP_RD}}$ 2. $\overline{\text{PCP_R/W}}$	1. Parallel Read Select (Intel Mode) 2. Parallel Read/Write Select (Motorola and Multiplexed Mode)	3.3V (5V tol)	BiDir	IN
108	-	GPIO41	General Purpose Input/Output	1. $\overline{\text{PCP_IRQ}}$ 2. $\overline{\text{SCP2_IRQ}}$	1. Parallel Control Port Data Ready Interrupt Request 2. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir/OD	IN
109	-	GPIO9	General Purpose Input/Output	1. PCP_A1 2. PCP_A9	1. Parallel Control Port Address Bit 1 2. Parallel Control Port Address Bit 9	3.3V (5V tol)	BiDir	IN
-	4	GPIO9	General Purpose Input/Output	1. $\overline{\text{SCP1_IRQ}}$	1. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir	IN
110	-	GPIO8	General Purpose Input/Output	1. PCP_A0 2. PCP_A8	1. Parallel Control Port Address Bit 0 2. Parallel Control Port Address Bit 8	3.3V (5V tol)	BiDir	IN
-	5	GPIO8	General Purpose Input/Output	1. $\overline{\text{SCP2_IRQ}}$	1. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir	IN
111	-	GPIO7	General Purpose Input/Output	1. PCP_D7 2. PCP_AD7	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
-	6	GPIO7	General Purpose Input/Output	1. SCP1_CS 2. IOWAIT	1. SPI Chip Select 2. SRAM Hold-Off Handshake	3.3V (5V tol)	BiDir	IN
112	-	GPIO6	General Purpose Input/Output	1. PCP_D6 2. PCP_AD6	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
-	7	GPIO6	General Purpose Input/Output	1. $\overline{\text{SCP2_CS}}$	1. SPI Chip Select	3.3V (5V tol)	BiDir	IN
113	8	VDDIO7	I/O power supply voltage			3.3V	PWR	
114	-	GPIO5	General Purpose Input/Output	1. PCP_D5 2. PCP_AD5	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
115	-	GPIO4	General Purpose Input/Output	1. PCP_D4 2. PCP_AD4	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
116	9	GNDIO7	I/O ground			0V	PWR	



Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
117	-	GPIO3	General Purpose Input/Output	1. PCP_D3 2. PCP_AD3	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
-	10	GPIO3	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN
118	-	GPIO2	General Purpose Input/Output	1. PCP_D2 2. PCP_AD2	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
-	11	GPIO2	General Purpose Input/Output	1. UART_TXD	1. UART Output	3.3V (5V tol)	BiDir	IN
119	12	VDD7	Core power supply voltage			1.8V	PWR	
120	-	GPIO1	General Purpose Input/Output	1. PCP_D1 2. PCP_AD1	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
	13	GPIO1	General Purpose Input/Output	1. UART_RXD	1. UART Input	3.3V (5V tol)	BiDir	IN
121	-	GPIO0	General Purpose Input/Output	1. PCP_D0 2. PCP_AD0	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN
	14	GPIO0	General Purpose Input/Output	1. UART_CLK 2. EE_CS	1. UART Clock	3.3V (5V tol)	BiDir	IN
122	15	GNDD7	Core ground			0V	PWR	
123	16	XTAL_OUT	Buffered Reference Clock Input/ Crystal Oscillator Input			3.3V (5V tol)	OUT	
124	17	XTI	Reference Clock Input/Crystal Oscillator Input			3.3V (5V tol)	ANA	
125	18	XTO	Crystal Oscillator Output 1			3.3V	ANA	
126	19	GNDA	PLL ground			1.8V	PWR	
127	-	NC	Do Not Connect on PCB			1.8V	ANA	
128	20	PLL_REF_RE S	Current Reference Output for PLL. Connect to resistor.			3.3V	ANA	
129	21	VDDA	PLL power.			3.3V	PWR	
130	22	VDD8	Core power supply voltage			1.8V	PWR	
131	23	GPIO14	General Purpose Input/Output	1. DAI1_DATA3 2. DSD3	1. PCM Audio Input Data 3 2. DSD Audio Input Data 3	3.3V (5V tol)	BiDir	IN
132	24	GPIO13	General Purpose Input/Output	1. DAI1_DATA2 2. DSD2	1. PCM Audio Input Data 2 2. DSD Audio Input Data 2	3.3V (5V tol)	BiDir	IN
133	25	GNDD8	Core ground			0V	PWR	
134	26	GPIO12	General Purpose Input/Output	1. DAI1_DATA1 2. DSD1	1. PCM Audio Input Data 1 2. DSD Audio Input Data 1	3.3V (5V tol)	BiDir	IN
135	27	DAI1_DATA0	PCM Audio Input Data 0	1. TM0 2. DSD0	1. 2. DSD Audio Input Data 0	3.3V (5V tol)	In	
136	28	VDDIO8	I/O power supply voltage			3.3V	PWR	
137	29	DAI1_SCLK	PCM Audio Input Bit Clock	DSD_CLK	DSD Audio Input Clock	3.3V (5V tol)	In	

Table P-10. CS4970x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State
138	30	DAI1_LRCLK	PCM Audio Input Sample Rate (Left/Right) Clock	DSD4	DSD Audio Input Data 4	3.3V (5V tol)	In	
139	31	GNDIO8	I/O ground			0V	PWR	
140	-	GPIO42	General Purpose Input/Output	1. DAI2_LRCLK 2. BDI_REQ 3. PCP_IRQ 4. PCP_BSY	1. PCM Audio Input Sample Rate Clock 2. Bursty Data Input Request 3. Parallel Control Port Data Ready Interrupt Request 4. Parallel Control Port Input Busy	3.3V (5V tol)	BiDir/OD	IN
-	32	GPIO42	General Purpose Input/Output	1. DAI2_LRCLK 2. BDI_REQ 3. PCP_IRQ 4. PCP_BSY	1. PCM Audio Input Sample Rate Clock 2. Bursty Data Input Request 3. Parallel Control Port Interrupt 4. Parallel Port Bus	3.3V (5V tol)	BiDir/OD	IN
141	33	GPIO43	General Purpose Input/Output	1. DAI2_SCLK 2. BDI_CLK	1. PCM Audio Input Bit Clock 2. Bursty Data Input Bit Clock	3.3V (5V tol)	BiDir	IN
142	34	DAI2_DATA	PCM Audio Input Data	1. DAI1_DATA4 2. DSD5 3. BDI_DATA	1. PCM Audio Input Data 4 2. DSD Audio Input Data 5 3. Bursty Data Input Data	3.3V (5V tol)	In	
143	-	GPIO27	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN
144	-	GPIO26	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN
-	35	GPIO26	General Purpose Input/Output	1. DAO2_DATA3 2. XMTB 3. UART_TX_ENABLE	1. Digital Audio Output 3. 2. Outputs IEC60958/61937 format bi-phase mark encoded S/PDIF data 3. Enable the UART_TX pin	3.3V (5V tol)	BiDir	IN



P.7 CS4953x4 Pin Assignments

Table P-11 shows the names and functions for each pin.

Table P-11. CS4953x4 Pin Assignments for 144-Pin and 128-Pin Packages

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
1	-	GPIO28	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN	Y
2	-	GPIO29	General Purpose Input/Output	1. XMTA_IN	1. S/PDIF Pass-thru Input.	3.3V (5V tol)	BiDir	IN	Y
3	36	DBDA	Debug Data			3.3V (5V tol)	In/OD	IN	Y
4	37	DBCK	Debug Clock			3.3V (5V tol)	In/OD	IN	Y
5	-	GPIO21	General Purpose Input/Output	1. DAO2_DATA3 2. XMTB 3. UART_TX_ENABLE	1. Digital Audio Output 3. 2. Outputs IEC60958/61937 format bi-phase mark encoded S/PDIF data. 3. Enable the UART_TX pin.	3.3V (5V tol)	BiDir	IN	Y
6	38	GPIO20	General Purpose Input/Output	1. DAO2_DATA2 2. EE_CS	1. Digital Audio Output 2. 2. EEPROM Boot Chip Select.	3.3V (5V tol)	BiDir	IN	Y
7	39	GPIO19	General Purpose Input/Output	1. DAO2_DATA1 2. HS4	1. Digital Audio Output 1. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN	Y
8	40	DAO_MCLK	Audio Master Clock			3.3V (5V tol)	BiDir	IN	Y
9	41	TEST	Test			3.3V (5V tol)	In		
10	42	VDD1	Core power supply voltage			1.8V	PWR		
11	43	GPIO18	General Purpose Input/Output	1. DAO2_DATA0 2. HS3	1. Digital Audio Output 0. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN	Y
12	44	GPIO22	General Purpose Input/Output	DAO2_SCLK	PCM Audio Bit Clock.	3.3V (5V tol)	BiDir	IN	Y
13	45	GNDD1	Core ground			0V	PWR		
14	46	GPIO23	General Purpose Input/Output	DAO2_LRCLK	Serial PCM Audio Sample Rate Clock for the serial data pins: (DAO2_DATA0, DAO2_DATA1, DAO2_DATA2, DAO2_DATA3).	3.3V (5V tol)	BiDir	IN	Y
15	47	GPIO17	General Purpose Input/Output	1. DAO1_DATA3 2. XMTA	1. Digital Audio Output 3. 2. S/PDIF Audio Output A.	3.3V (5V tol)	BiDir	IN	Y
16	48	GPIO16	General Purpose Input/Output	1. DAO1_DATA2 2. HS2	1. Digital Audio Output 2. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN	Y





Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
17	49	GPIO15	General Purpose Input/Output	1. DAO1_DATA1 2. HS1	1. Digital Audio Output 1. 2. Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN	Y
18	50	VDDIO1	I/O power supply voltage			3.3V	PWR		
19	51	DAO1_DATA0	Digital Audio Output 0	HS0	Hardware Strap Mode Select.	3.3V (5V tol)	BiDir	IN	
20	52	DAO1_SCLK	PCM Audio Bit Clock			3.3V (5V tol)	BiDir	IN	Y
21	53	GNDIO1	I/O ground			0V	PWR		
22	54	DAO1_LRCLK	PCM Audio Sample Rate Clock			3.3V (5V tol)	BiDir	IN	Y
23	-	GPIO31	General Purpose Input/Output	UART_CLK	UART Clock.	3.3V (5V tol)	BiDir	IN	Y
24	55	VDD2	Core power supply voltage			1.8V	PWR		
25	-	GPIO25	General Purpose Input/Output	UART_TXD	UART Output.	3.3V (5V tol)	BiDir	IN	Y
26	-	GPIO24	General Purpose Input/Output	UART_RXD	UART Input.	3.3V (5V tol)	BiDir	IN	Y
27	56	GNDD2	Core ground			0V	PWR		
28	57	SD_DQM0	SDRAM Data Mask 0			3.3V (5V tol)	OUT		
29	58	SD_D7	SDRAM Data Bit 7	EXT_D7	Flash Data Bit 7.	3.3V (5V tol)	BiDir	IN	Y
30	59	SD_D6	SDRAM Data Bit 6	EXT_D6	Flash Data Bit 6.	3.3V (5V tol)	BiDir	IN	Y
31	60	SD_D5	SDRAM Data Bit 5	EXT_D5	Flash Data Bit 5.	3.3V (5V tol)	BiDir	IN	Y
32	61	SD_D4	SDRAM Data Bit 4	EXT_D4	Flash Data Bit 4.	3.3V (5V tol)	BiDir	IN	Y
33	62	VDDIO2	I/O power supply voltage			3.3V	PWR		
34	63	SD_D3	SDRAM Data Bit 3	EXT_D3	Flash Data Bit 3.	3.3V (5V tol)	BiDir	IN	Y
35	64	SD_D2	SDRAM Data Bit 2	EXT_D2	Flash Data Bit 2.	3.3V (5V tol)	BiDir	IN	Y
36	67	GNDIO2	I/O ground			0V	PWR		
37	65	SD_D1	SDRAM Data Bit 1	EXT_D1	Flash Data Bit 1.	3.3V (5V tol)	BiDir	IN	Y



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
38	66	EXT_WE	Flash Write Enable			3.3V (5V tol)	OUT		
39	68	SD_D0	SDRAM Data Bit 0	EXT_D0	Flash Data Bit 0	3.3V (5V tol)	BiDir	IN	Y
40	69	SD_D15	SDRAM Data Bit 15	EXT_D15	Flash Data Bit 15	3.3V (5V tol)	BiDir	IN	Y
41	70	SD_D14	SDRAM Data Bit 14	EXT_D14	Flash Data Bit 14	3.3V (5V tol)	BiDir	IN	Y
42	71	SD_D13	SDRAM Data Bit 13	EXT_D13	Flash Data Bit 13	3.3V (5V tol)	BiDir	IN	Y
43	72	SD_D12	SDRAM Data Bit 12	EXT_D12	Flash Data Bit 12	3.3V (5V tol)	BiDir	IN	Y
44	73	VDDIO3	I/O power supply voltage			3.3V	PWR		
45	74	SD_D11	SDRAM Data Bit 11	EXT_D11	Flash Data Bit 11	3.3V (5V tol)	BiDir	IN	Y
46	75	SD_D10	SDRAM Data Bit 10	EXT_D10	Flash Data Bit 10	3.3V (5V tol)	BiDir	IN	Y
47	76	GNDIO3	I/O ground			0V	PWR		
48	77	SD_D9	SDRAM Data Bit 9	EXT_D9	Flash Data Bit 9	3.3V (5V tol)	BiDir	IN	Y
49	78	SD_D8	SDRAM Data Bit 8	EXT_D8	Flash Data Bit 8	3.3V (5V tol)	BiDir	IN	Y
50	79	SD_DQM1	SDRAM Data Mask 1			3.3V (5V tol)	OUT		
51	80	SD_CLKOUT	SDRAM Clock Output			3.3V (5V tol)	OUT		
52	81	SD_CLKIN	SDRAM Clock Input			3.3V (5V tol)	In		Y
53	82	SD_CLKEN	SDRAM Clock Enable			3.3V (5V tol)	OUT		
54	83	VDD3	Core power supply voltage			1.8V	PWR		
55	84	SD_A12	SDRAM Address Bit 12	EXT_A12	Flash Address Bit 12	3.3V (5V tol)	OUT		
56	85	SD_A11	SDRAM Address Bit 11	EXT_A11	Flash Address Bit 11	3.3V (5V tol)	OUT		
57	86	GNDD3	Core ground			0V	PWR		



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
58	87	SD_A9	SDRAM Address Bit 9	EXT_A9	Flash Address Bit 9	3.3V (5V tol)	OUT		
59	88	SD_A8	SDRAM Address Bit 8	EXT_A8	Flash Address Bit 8	3.3V (5V tol)	OUT		
60	89	VDDIO4	I/O power supply voltage			3.3V	PWR		
61	90	SD_A7	SDRAM Address Bit 7	EXT_A7	Flash Address Bit 7	3.3V (5V tol)	OUT		
62	91	SD_A6	SDRAM Address Bit 6	EXT_A6	Flash Address Bit 6	3.3V (5V tol)	OUT		
63	92	GNDIO4	I/O ground			0V	PWR		
64	93	SD_A5	SDRAM Address Bit 5	EXT_A5	Flash Address Bit 5	3.3V (5V tol)	OUT		
65	94	EXT_CS2	Chip Select 2			3.3V (5V tol)	OUT		
66	95	VDD4	Core power supply voltage			1.8V	PWR		
67	96	SD_A4	SDRAM Address Bit 4	EXT_A4	Flash Address Bit 4	3.3V (5V tol)	OUT		
68	97	SD_A3	SDRAM Address Bit 3	EXT_A3	Flash Address Bit 3	3.3V (5V tol)	OUT		
69	98	GNDD4	Core ground			0V	PWR		
70	99	SD_A2	SDRAM Address Bit 2	EXT_A2	Flash Address Bit 2	3.3V (5V tol)	OUT		
71	101	SD_A1	SDRAM Address Bit 1	EXT_A1	Flash Address Bit 1	3.3V (5V tol)	OUT		
72	102	SD_A0	SDRAM Address Bit 0	EXT_A0	Flash Address Bit 0	3.3V (5V tol)	OUT		
73	100	VDDIO5	I/O power supply voltage			3.3V	PWR		
74	103	SD_A10	SDRAM Address Bit 10	EXT_A10	Flash Address Bit 10	3.3V (5V tol)	OUT		
75	104	SD_BA0	SDRAM Bank Address 0	EXT_A13	Flash Address Bit 13	3.3V (5V tol)	OUT		
76	105	GNDIO5	I/O ground			0V	PWR		
77	106	SD_BA1	SDRAM Bank Address 1	EXT_A14	Flash Address Bit 14	3.3V (5V tol)	OUT		
78	107	SD_WE	SDRAM Write Enable			3.3V (5V tol)	OUT		



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
79	108	SD_CAS	SDRAM Column Address Strobe			3.3V (5V tol)	OUT		
80	109	SD_RAS	SDRAM Row Address Strobe			3.3V (5V tol)	OUT		
81	110	SD_CS	SDRAM Chip Select			3.3V (5V tol)	OUT		
82	111	EXT_A15	Flash Address Bit 15			3.3V (5V tol)	OUT		
83	112	VDD5	Core power supply voltage			1.8V	PWR		
84	113	EXT_A16	Flash Address Bit 16			3.3V (5V tol)	OUT		
85	114	EXT_A17	Flash Address Bit 17			3.3V (5V tol)	OUT		
86	115	GNDD5	Core ground			0V	PWR		
87	116	EXT_A18	Flash Address Bit 18			3.3V (5V tol)	OUT		
88	117	EXT_A19	Flash Address Bit 19			3.3V (5V tol)	OUT		
89	118	EXT_OE	Flash Output Enable			3.3V (5V tol)	OUT		
90	119	EXT_CS1	Active-low Flash chip select			3.3V (5V tol)	OUT		
91	120	VDDIO6	I/O power supply voltage			3.3V	PWR		
92	-	GPIO30	General Purpose Input/Output	1. CSW_U 2. XMTB_IN	1. Channel status user data input 2. S/PDIF Pass-thru Input	3.3V (5V tol)	BiDir	IN	Y
93	121	RESET	Chip Reset			3.3V (5V tol)	In		
94	122	GNDIO6	I/O ground			0V	PWR		
95	123	GPIO33	General Purpose Input/Output	SCP1_MOSI	SPI Mode Master Data Output/Slave Data Input	3.3V (5V tol)	BiDir	IN	Y
96	-	GPIO32	General Purpose Input/Output	1. $\overline{\text{SCP1_CS}}$ 2. IOWAIT	1. SPI Chip Select 2. SRAM Hold-Off Handshake	3.3V (5V tol)	BiDir	IN	Y
97	124	GPIO34	General Purpose Input/Output	1. SCP1_MISO 2. SCP1_SDA	1. SPI Mode Master Data Input/Slave Data Output 2. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN	Y
98	125	VDD6	Core power supply voltage			1.8V	PWR		



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
99	126	GPIO35	General Purpose Input/Output	SCP1_CLK	SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN	Y
100	-	GPIO36	General Purpose Input/Output	SCP1_IRQ	Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir/OD	IN	Y
101	127	GNDD6	Core ground			0V	PWR		
102	-	GPIO37	General Purpose Input/Output	1. SCP1_BSY 2. PCP_BSY	1. Serial Control Port 1 Input Busy 2. Parallel Control Port Input Busy	3.3V (5V tol)	BiDir/OD	IN	Y
-	128	GPIO37	General Purpose Input/Output	1. $\overline{\text{SCP1_BSY}}$	1. Serial Control Port 1 Input Busy	3.3V (5V tol)	BiDir/OD	IN	Y
103	-	GPIO38	General Purpose Input/Output	1. $\overline{\text{PCP_WR}}$ 2. $\overline{\text{PCP_DS}}$ 3. SCP2_CLK	1. Parallel Port Write Select (Intel Mode) 2. Parallel Port Data Strobe (Motorola and Multiplexed Mode) 3. SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN	Y
-	1	GPIO38	General Purpose Input/Output	1. SCP2_CLK	1. SPI/I ² C Control Port Clock	3.3V (5V tol)	BiDir/OD	IN	Y
104	-	GPIO39	General Purpose Input/Output	1. $\overline{\text{PCP_CS}}$ 2. $\overline{\text{SCP2_CS}}$	1. Parallel Port Chip Select (Intel/Motorola/Multiplexed Mode) 2. SPI Chip Select	3.3V (5V tol)	BiDir	IN	Y
105	-	GPIO11	General Purpose Input/Output	1. $\overline{\text{PCP_A3}}$ 2. $\overline{\text{PCP_AS}}$ 3. SCP2_MISO 4. SCP2_SDA	1. Parallel Control Port Address Bit 3 2. Parallel Control Port Address Strobe 3. SPI Mode Master Data Input/Slave Data Output 4. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN	Y
-	2	GPIO11	General Purpose Input/Output	1. SCP2_MISO 2. SCP2_SDA	1. SPI Mode Master Data Input/Slave Data Output 2. I ² C Mode Master/Slave Data IO	3.3V (5V tol)	BiDir/OD	IN	Y
106	-	GPIO10	General Purpose Input/Output	1. PCP_A2 2. PCP_A10 3. SCP2_MOSI	1. Parallel Control Port Address Bit 2 2. Parallel Control Port Address Bit 10 3. SPI Mode Master Data Output/Slave Data Input	3.3V (5V tol)	BiDir	IN	Y
-	3	GPIO10	General Purpose Input/Output	1. SCP2_MOSI	1. SPI Mode Master Data Output/Slave Data Input	3.3V (5V tol)	BiDir	IN	Y
107	-	GPIO40	General Purpose Input/Output	1. $\overline{\text{PCP_RD}}$ 2. $\overline{\text{PCP_R/W}}$	1. Parallel Read Select (Intel Mode) 2. Parallel Read/Write Select (Motorola and Multiplexed Mode)	3.3V (5V tol)	BiDir	IN	Y



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
108	-	GPIO41	General Purpose Input/Output	1. $\overline{\text{PCP_IRQ}}$ 2. $\overline{\text{SCP2_IRQ}}$	1. Parallel Control Port Data Ready Interrupt Request 2. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir/OD	IN	Y
109	-	GPIO9	General Purpose Input/Output	1. PCP_A1 2. PCP_A9	1. Parallel Control Port Address Bit 1 2. Parallel Control Port Address Bit 9	3.3V (5V tol)	BiDir	IN	Y
-	4	GPIO9	General Purpose Input/Output	1. $\overline{\text{SCP1_IRQ}}$	1. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir	IN	Y
110	-	GPIO8	General Purpose Input/Output	1. PCP_A0 2. PCP_A8	1. Parallel Control Port Address Bit 0 2. Parallel Control Port Address Bit 8	3.3V (5V tol)	BiDir	IN	Y
-	5	GPIO8	General Purpose Input/Output	1. $\overline{\text{SCP2_IRQ}}$	1. Serial Control Port Data Ready Interrupt Request	3.3V (5V tol)	BiDir	IN	Y
111	-	GPIO7	General Purpose Input/Output	1. PCP_D7 2. PCP_AD7	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
-	6	GPIO7	General Purpose Input/Output	1. SCP1_CS 2. IOWAIT	1. SPI Chip Select 2. SRAM Hold-Off Handshake	3.3V (5V tol)	BiDir	IN	Y
112	-	GPIO6	General Purpose Input/Output	1. PCP_D6 2. PCP_AD6	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
-	7	GPIO6	General Purpose Input/Output	1. $\overline{\text{SCP2_CS}}$	1. SPI Chip Select	3.3V (5V tol)	BiDir	IN	Y
113	8	VDDIO7	I/O power supply voltage			3.3V	PWR		
114	-	GPIO5	General Purpose Input/Output	1. PCP_D5 2. PCP_AD5	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
115	-	GPIO4	General Purpose Input/Output	1. PCP_D4 2. PCP_AD4	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
116	9	GNDIO7	I/O ground			0V	PWR		
117	-	GPIO3	General Purpose Input/Output	1. PCP_D3 2. PCP_AD3	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
-	10	GPIO3	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN	Y
118	-	GPIO2	General Purpose Input/Output	1. PCP_D2 2. PCP_AD2	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
-	11	GPIO2	General Purpose Input/Output	1. UART_TXD	1. UART Output	3.3V (5V tol)	BiDir	IN	Y
119	12	VDD7	Core power supply voltage			1.8V	PWR		
120	-	GPIO1	General Purpose Input/Output	1. PCP_D1 2. PCP_AD1	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
	13	GPIO1	General Purpose Input/Output	1. UART_RXD	1. UART Input	3.3V (5V tol)	BiDir	IN	Y
121	-	GPIO0	General Purpose Input/Output	1. PCP_D0 2. PCP_AD0	1. Parallel Control Port Data Bus 2. Parallel Control Port Multiplexed Address and Data Bus	3.3V (5V tol)	BiDir	IN	Y
	14	GPIO0	General Purpose Input/Output	1. UART_CLK	1. UART Clock	3.3V (5V tol)	BiDir	IN	Y
122	15	GNDD7	Core ground			0V	PWR		
123	16	XTAL_OUT	Buffered Reference Clock Input/Crystal Oscillator Input			3.3V (5V tol)	OUT		
124	17	XTI	Reference Clock Input/Crystal Oscillator Input			3.3V (5V tol)	ANA		
125	18	XTO	Crystal Oscillator Output 1			3.3V	ANA		
126	19	GNDA	PLL ground			1.8V	PWR		
127	-	NC	Do Not Connect on PCB			1.8V	ANA		
128	20	PLL_REF_RES	Current Reference Output for PLL. Connect to resistor.			3.3V	ANA		
129	21	VDDA	PLL power.			3.3V	PWR		
130	22	VDD8	Core power supply voltage			1.8V	PWR		
131	23	GPIO14	General Purpose Input/Output	1. DAI1_DATA3 2. DSD3	1. PCM Audio Input Data 3 2. DSD Audio Input Data 3	3.3V (5V tol)	BiDir	IN	Y
132	24	GPIO13	General Purpose Input/Output	1. DAI1_DATA2 2. DSD2	1. PCM Audio Input Data 2 2. DSD Audio Input Data 2	3.3V (5V tol)	BiDir	IN	Y
133	25	GNDD8	Core ground			0V	PWR		
134	26	GPIO12	General Purpose Input/Output	1. DAI1_DATA1 2. DSD1	1. PCM Audio Input Data 1 2. DSD Audio Input Data 1	3.3V (5V tol)	BiDir	IN	Y
135	27	DAI1_DATA0	PCM Audio Input Data 0	DSD0	DSD Audio Input Data 0	3.3V (5V tol)	In		Y
136	28	VDDIO8	I/O power supply voltage			3.3V	PWR		
137	29	DAI1_SCLK	PCM Audio Input Bit Clock	DSD_CLK	DSD Audio Input Clock	3.3V (5V tol)	In		Y



Table P-11. CS4953x4 Pin Assignments (Continued) for 144-Pin and 128-Pin Packages (Continued)

LQFP-144 Pin #	LQFP-128 Pin #	Function 1 (Default)	Description of Default Function	Secondary Functions	Description of Secondary Functions	Pwr	Type	Reset State	Pullup at Reset
138	30	DAI1_LRCLK	PCM Audio Input Sample Rate (Left/Right) Clock	DSD4	DSD Audio Input Data 4	3.3V (5V tol)	In		Y
139	31	GNDIO8	I/O ground			0V	PWR		
140	-	GPIO42	General Purpose Input/Output	1. DAI2_LRCLK 2. BDI_REQ 3. PCP_IRQ 4. PCP_BSY	1. PCM Audio Input Sample Rate Clock 2. Bursty Data Input Request 3. Parallel Control Port Data Ready Interrupt Request 4. Parallel Control Port Input Busy	3.3V (5V tol)	BiDir/OD	IN	Y
-	32	GPIO42	General Purpose Input/Output	1. DAI2_LRCLK 2. BDI_REQ	1. PCM Audio Input Sample Rate Clock 2. Bursty Data Input Request	3.3V (5V tol)	BiDir/OD	IN	Y
141	33	GPIO43	General Purpose Input/Output	1. DAI2_SCLK 2. BDI_CLK	1. PCM Audio Input Bit Clock 2. Bursty Data Input Bit Clock	3.3V (5V tol)	BiDir	IN	Y
142	34	DAI2_DATA	PCM Audio Input Data	1. DAI1_DATA4 2. DSD5 3. BDI_DATA	1. PCM Audio Input Data 4 2. DSD Audio Input Data 5 3. Bursty Data Input Data	3.3V (5V tol)	In		Y
143	-	GPIO27	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN	Y
144	-	GPIO26	General Purpose Input/Output			3.3V (5V tol)	BiDir	IN	Y
-	35	GPIO26	General Purpose Input/Output	1. DAO2_DATA3 2. XMTB 3. UART_TX_ENABLE	1. Digital Audio Output 3. 2. Outputs IEC60958/61937 format bi-phase mark encoded S/PDIF data 3. Enable the UART_TX pin	3.3V (5V tol)	BiDir	IN	Y

§§¹

1. The “§§” symbol is used throughout this manual to indicate the end of the text flow in a chapter.

Chapter 1

Operational Modes

1.1 Overview

The CS4953x4/CS4970x4 DSP has several operational modes that can be used to conform to many system configurations. The DSP is configured to Master Boot as shown in [Figure 1-1](#) in order to simplify connectivity and to reduce the complexity of the microcontroller code used in the system.

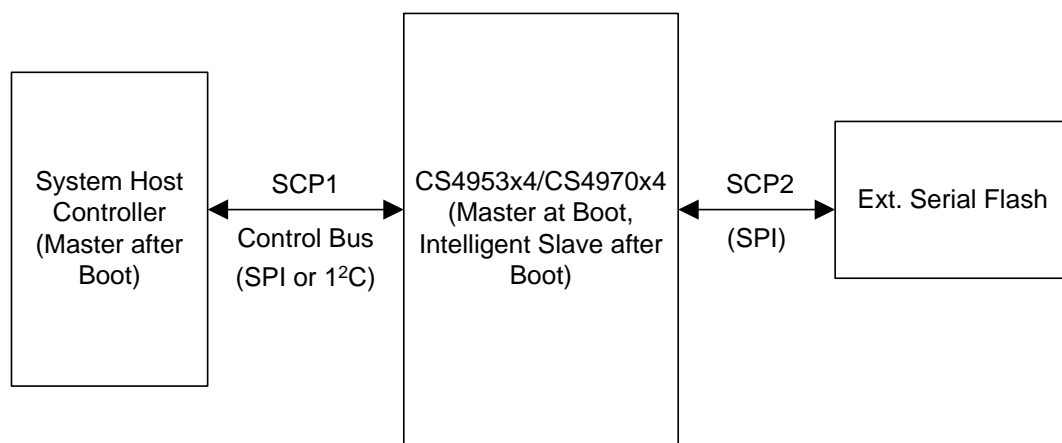


Figure 1-1. Operation Mode Block Diagram

The DSP is configured to boot from external serial SPI Flash when reset is released. Once the DSP is booted, it turns into an intelligent Slave device. The DSP will swap out decoders from external SPI Flash when stream changes without the necessity of interacting with the host. The DSP will change any mid- or post-processing module with a single command from the host. The DSP takes on most of the control functionality from the microcontroller. This significantly reduces the microcontroller development effort thereby reducing time to market of the end product.

1.1.1 Supported Serial Flash Devices

Cirrus logic recommends SPI serial flash devices with a minimum storage capacity of 8 Mbits. The Cirrus recommended flash devices are:

- SST25VF080B (tested) - See the data sheet for this device at:
<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en549423>
- EN25P32 (tested) - See the data sheet for this device at:
<http://www.eonssi.com/upfile/p2008929182330.pdf>
- M25P80 (tested) - See the data sheet for this device at:
<http://www.icbase.com/pdf/STM/STM30070501.pdf>
- SST25LFxxx
- AT45DB041D - See the data sheet for this device at:

http://www.datasheet4u.com/html/A/T/4/AT45DB041D_ATMELCorporation.pdf.html

1.2 Operational Mode Selection

The operational mode for the CS4953x4/CS4970x4 is selected by the values of the HS[4:0] pins on the rising edge of $\overline{\text{RESET}}$. This value determines the communication mode used until the part is reset again. This value also determines the method for loading application code. The table below shows the different operational modes and the HS[4:0] values for each mode.

Table 1-1. Operation Modes

HS[4:0]					Mode	Boot Master Device	Boot Slave Device
X	0	0	0	1	Master SCP2 SPI2	CS4953x4/ CS4970x4	SPI (Mode 2) External ROM ^{1, 3, 4,}
X	1	0	0	1	Master SCP2 SPI3	CS4953x4/ CS4970x4	SPI (Mode 3) External ROM ^{2, 3, 4, 5,}

1. In SPI Master mode 2, the following defaults are used: SPI Command Byte 0xE8, Image Start address 0x0 is sent as a 24-bit value, 4 dummy bytes sent following the address (and before reading image data), SPI clock frequency = $F_{\text{dclk}} / 2$. This mode supports the Atmel SPI Flash memory.
2. In SPI Master mode 3, the following defaults are used: SPI command byte 0x03, Image Start address 0x0 is sent as a 24-bit value, no dummy bytes, SPI clock frequency = $F_{\text{dclk}} / 2$. This mode supports the ST SPI EEPROM devices.
3. For all SPI Master boot modes, by default Pin 25 (144-pin package) and Pin 14 (128-pin package) is used as EE_CS.
4. For Master boot modes, the following defaults are used: clock ratio=1:1, Endian Mode = Big Endian, Chip Select polarity = active-low, 0-cycle delay from CS/Address Change to Output Enable, 4-cycle delay from CS to Read Access, SCP1- SPI mode.
5. F_{dclk} is specified in the CS4953xx and /CS4970x4 Data Sheets.

1.3 Booting the DSP in Master Boot Mode

When designing an AVR, the DSP must be connected as shown in [Figure 1-2](#) and [Figure 1-3](#). Pay special attention to the pull-up and pull-down resistors on each pin. These resistors are mode-select pins and impact the behavior of the DSP after power-up. When the DSP is connected as shown in [Figure 1-2](#) and [Figure 1-3](#), the DSP will boot from SPI Flash when Reset is released. The Read command word that is put out on the address bus when Reset is released depends on the resistor on Pin 11. [Table 1-1](#) describes the format of the Read command sent out by the DSP when Reset is released. The designer must select a SPI Flash that supports the read format listed in [Table 1-1](#). During internal development and testing, the EON EN25P32, SST25VF, and the ST M25P80 SPI Flash devices were used for testing. If the SPI Flash does not support the Read command listed in [Table 1-1](#), please contact your Cirrus Logic representative for additional assistance.

Table 1-2. Supported SPI Flash Read Format

Pin 11 (144-Pin Package) Cycle Type Operation	Max Freq. (MHz) ¹	Bus Cycle ²																	
		1		2		3		4		5		6		7		8		9	
		SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO	SCP2-MOSI	SCP2-MISO
High	50	0x03	Hi-Z	0x00	Hi-Z	0x00	Hi-Z	0x00	Hi-Z	x ³	D _{OUT} (Byte 0)	X	D _{OUT} (Byte 1)	X	D _{OUT} (Byte 2)	X	D _{OUT} (Byte 3)	X	D _{OUT} (Byte 4)
Low	50	0x68	Hi-Z	0x00	Hi-Z	0x00	Hi-Z	0x00	Hi-Z	X	Hi-Z	X	Hi-Z	X	Hi-Z	X	Hi-Z	X	D _{OUT}

1. SCP2_SCLK is 12.288 MHz (based on 24.576 MHz in CLKIN) during initial boot and then switches to 50 MHz once the PLL is locked.
2. One bus cycle is eight clock periods.
3. x = Don't care

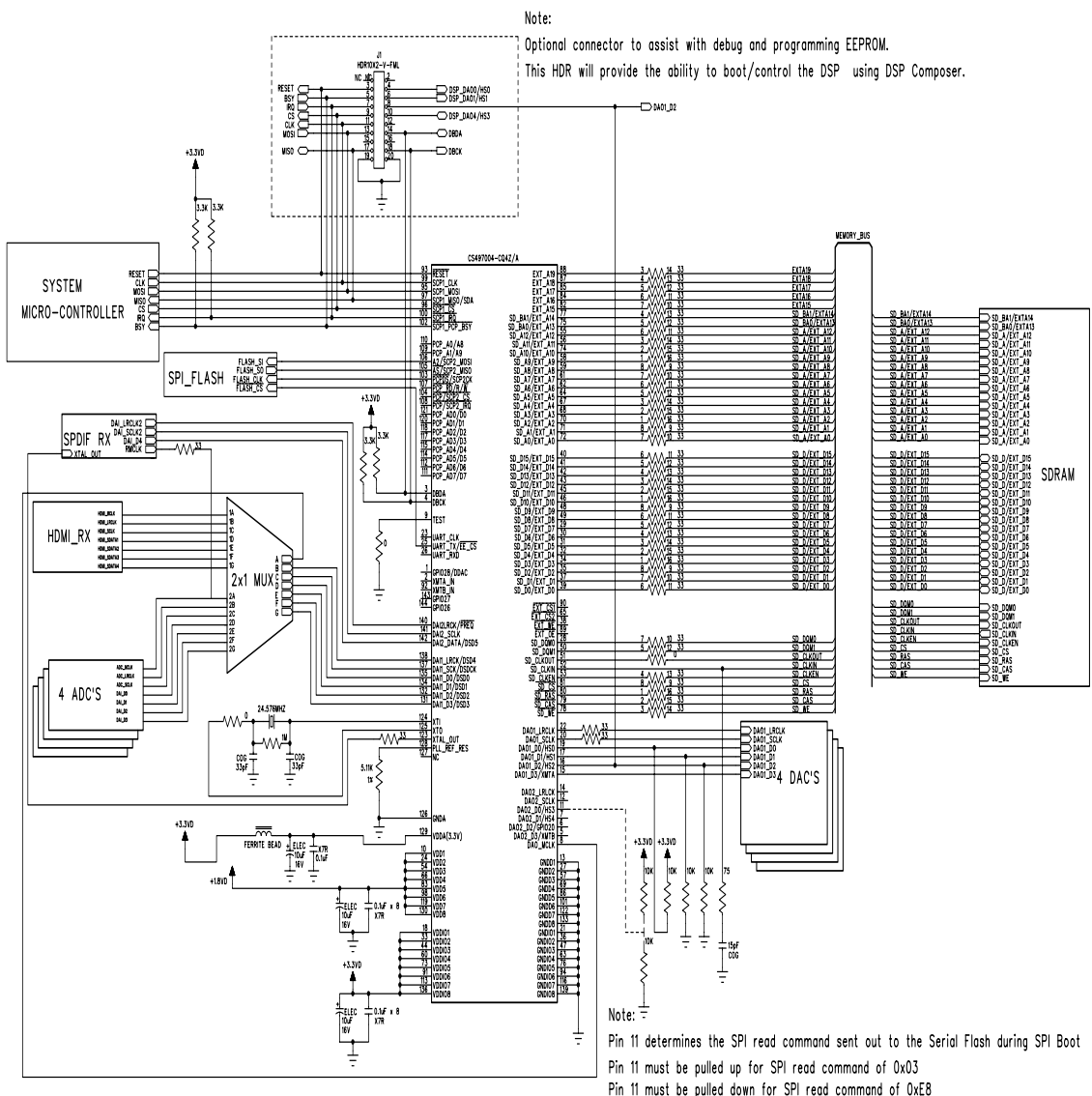


Figure 1-2. CS497004, LQFP 144-Pin Package, SPI Control, Master Boot Typical Connection Diagram

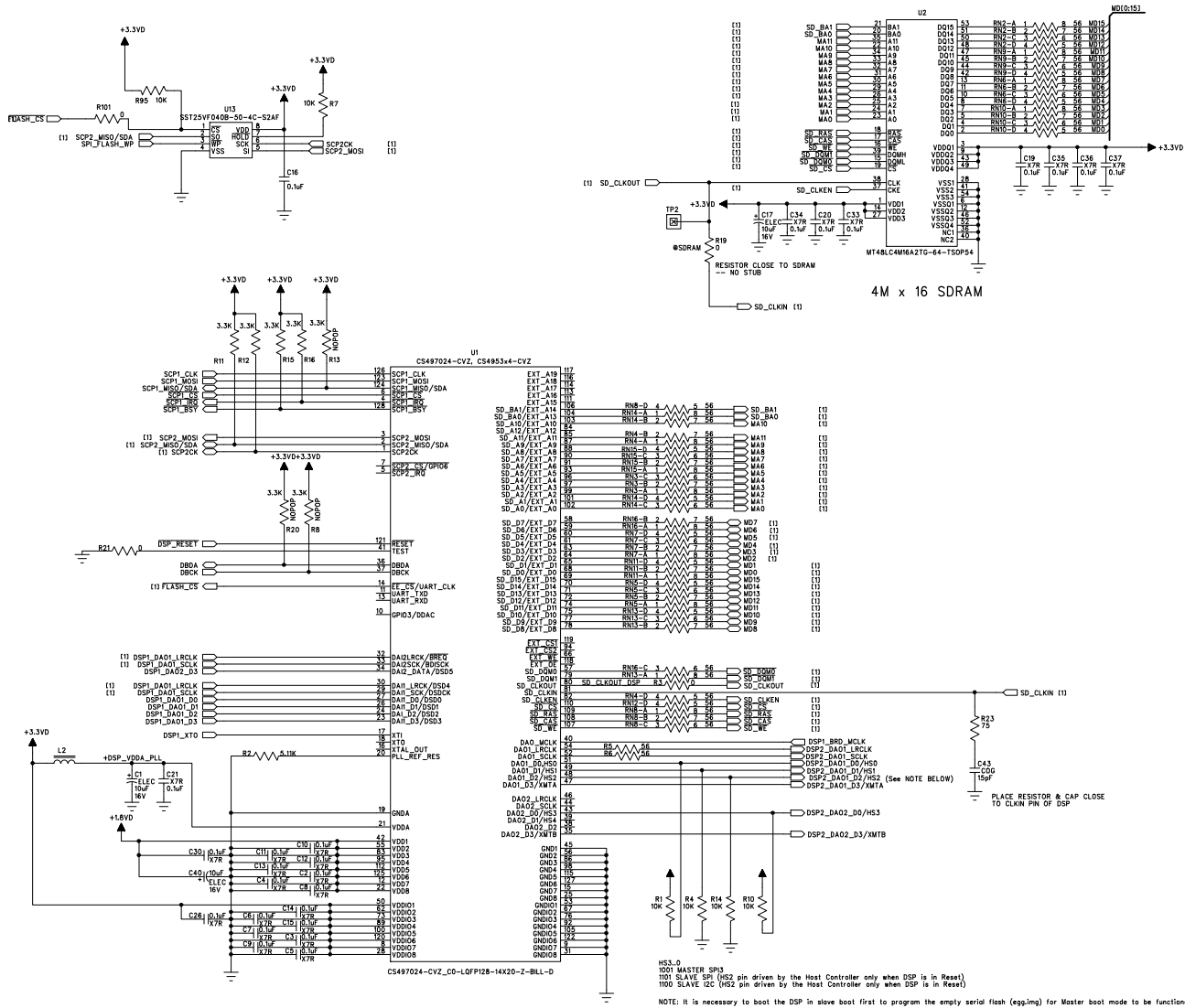


Figure 1-3. CS497004/CS4963x4, LQFP 128-Pin Package, SPI Control, Master Boot Typical Connection Diagram

HS3_0
100 MASTER SPI
101 SLAVE SPI (HS2 pin driven by the Host Controller only when DSP is in Reset)
100 SLAVE I2C (HS2 pin driven by the Host Controller only when DSP is in Reset)

NOTE: It is necessary to boot the DSP in slave boot first to program the empty serial flash (egging) for Master boot mode to be functional

The typical connection diagram in [Figure 1-2](#) and [Figure 1-3](#) is shown with a microcontroller controlling the DSP in SPI mode. Details of the SPI protocol can be found in [Section 2.4, "SPI Port" on page 2-1](#). An external SDRAM is also shown in [Figure 1-2](#) and [Figure 1-3](#). The external SDRAM is required for systems that require HD decoders. Care must be taken when routing this interface. This interface will run at 150 MHz during normal operation. Details of the SDRAM interface can be found in [Chapter 5, "External Memory Interfaces"](#).

An 2x10 header J1 is shown in [Figure 1-2](#) and [Figure 1-3](#). This Header is optional but highly recommended. The Header performs three functions:

1. Allows a blank SPI Flash to be programmed with the Cirrus Logic CDB USB Master board
2. Allows a user to use the Cirrus Logic DSP Composer development tool on an end product.
3. Allows a Cirrus Logic representative to easily debug an end system.

A SPDIF_RX, HDMI_RX and ADCs are shown in [Figure 1-2](#) and [Figure 1-3](#). These outputs of these devices are connected to inputs of the DSP. These devices provide data to the DAI port of the DSP. Details of the DAI port are described in [Chapter 3, "Audio Input Interfaces"](#).

DACs are shown in [Figure 1-2](#). The output of the DSP is connected to the input of the DACs. The DSP has a total of 8 output lines and can generate up to 16 channels of audio. Details of the DAO port are described in [Chapter 4](#).

Special attention must be paid to configuring the clocks in an AVR system. As seen in [Figure 1-2](#), the input and output clock domains are independent but all synchronous to MCLK. The DSP is a Slave to all clocks on the input side and hence the S/PDIF_RX, HDMI_RX and ADCs must be configured to be the Master to MCLK, SCLK, and LRCLK for the DSP. On the output side, the DSP is a Slave to MCLK and a Master to SCLK and LRCK. The frequency of the SCLK and LRCLK can be configured at design time in DSP Condenser. Refer to [Chapter 8, "DSP Condenser"](#) for more details.

1.3.1 Performing a Master Boot

When the DSP is connected as shown in [Figure 1-2](#) and [Figure 1-3](#), the boot procedure to boot the DSP is described in [Figure 1-4](#).

Pseudocode and flowcharts will be used to describe each of these boot procedures in detail. The flow charts use the following messages:

- Read_* – Read from CS4953x4/CS4970x4

Please note that * above can be replaced by SPI™ or I²C. The system designer should also refer to the control port sections of this document in [Chapter 2, "Serial Communication Mode"](#), for the details of writing to and reading from the CS4953x4/CS4970x4.

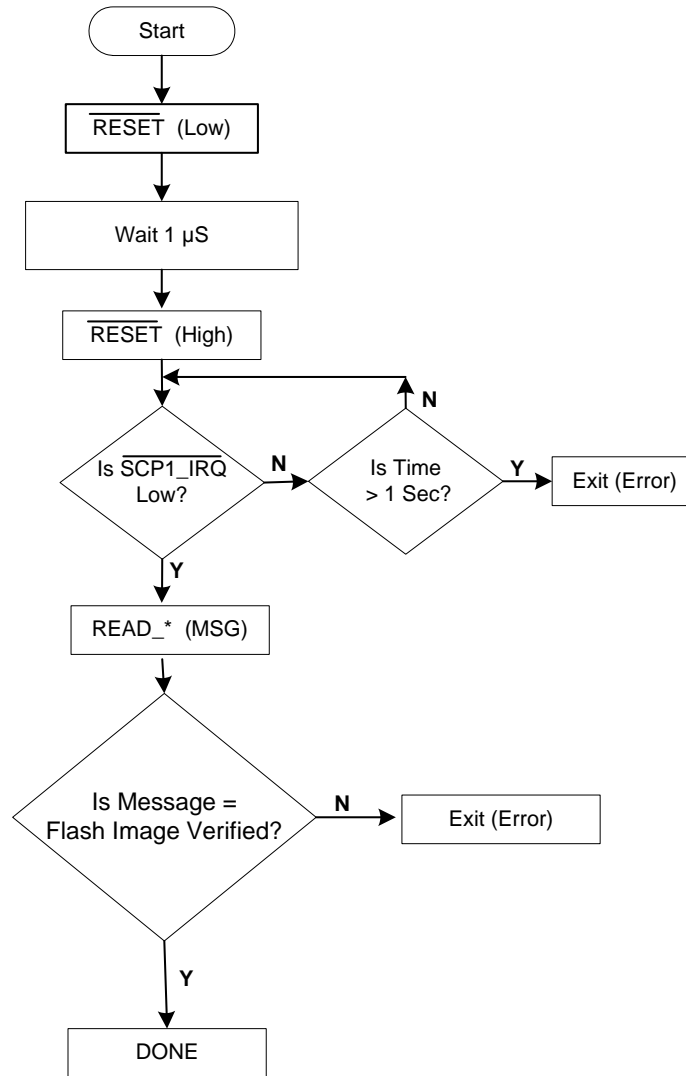


Figure 1-4. Master Boot Flow

1.3.1.1 Master Boot Protocol

1. Set **RESET** Low. A download sequence is started when the host holds the **RESET** pin low for the required time.
2. Wait for 1 μ S.
3. Set **RESET** High. As soon as Reset goes High, the SCP2 lines will start toggling and the DSP will begin to boot from SPI Flash. The commands sent out to the SPI Flash are listed in [Table 1-1](#)
4. Wait for SCP1_IRQ to go Low.
5. Read the message from the DSP. Current DSP messages are described in [Table 1-2](#).
6. If the Master boot is successful, the DSP will respond with the “Flash image verified” message¹.

1. A “Flash image verified” message does not guarantee that the complete Flash image has been programmed correctly or is bit-exact. This message only verifies that certain key words are at specified locations. It is the user’s responsibility to ensure that the Flash is programmed correctly by reading the contents of the Flash.

7. If the message is "Flash image verified," then the DSP will be processing audio at this time from the default port.

1.3.1.2 Messages Read from CS4953x4/CS4970x4

Table 1-2 defines the boot read messages, in mnemonic and actual hex value, used in CS4953x4/CS4970x4 boot sequences.

Table 1-3. Boot Read Messages from CS4953x4/CS4970x4

MNEMONIC	VALUE (Command/Data)
FLASH IMAGE VERIFIED	0xEF000020 ¹ 0x00000001
FLASH IMAGE VERIFICATION FAILED	0xEF000020 0x80000001
Reserved for future Boot Messages	Reserved for future Boot Messages

1. For a description of 0xEF000020 command, refer to DSP_STATUS variable in Table 7-5, Firmware Status Registers.

§§

Chapter 2

Serial Communication Mode

2.1 Introduction

The CS4953x4/CS4970x4 uses the Serial Control Port (SCP) to communicate with external devices such as host microprocessors using either I²C or SPI serial communication formats.

2.2 Communication Using the Serial Control Port

The SCP1 Port is configured as a Slave and the SCP2 Port is configured as a Master. The CS4953x4/CS4970x4 DSP serial ports communicate using the SCP_CLK, SCP_MOSI, and SCP_MISO (SPI serial Master and Slave modes), and SCP_SDA (for I²C serial Master and Slave modes) pins.

In both SPI and I²C modes, the serial control port performs 8-bit transfers. The SCP1 Port can request a read from the host by activating the $\overline{\text{SCP1_IRQ}}$ pin. The port can also indicate that the host should stop sending data by activating the $\overline{\text{SCP1_BSY}}$ pin.

It is mandatory for the host to obey the $\overline{\text{SCP1_BSY}}$ pin status. Messages sent to the DSPs host control port (SCP1) when $\overline{\text{SCP1_BSY}}$ pin is low will be lost.

The CS4953x4/CS4970x4 SPI and I²C serial communication modes are identical from a functional standpoint. The main difference between the two is the actual protocol being implemented between the CS4953x4/CS4970x4 and the host.

The CS4953x4/CS4970x4 has two serial ports. The O/S currently supports only Slave mode host communication on SCP1, and Master mode communication on SCP2 for booting from a SPI Flash.

2.3 Serial Control Port Configuration

The serial control port configuration for an operating mode is determined by register settings in DSP Condenser. Refer to [Chapter 8, "DSP Condenser"](#) for more details. The CS4953x4/CS4970x4 OS currently supports two serial control port configurations for host control:

- SPI Slave (Write Address = 0x80, Read Address = 0x81). See [Section 2.4](#).
- I²C Slave (Write Address = 0x80, Read Address = 0x81) (Contact your Cirrus FAE or representatives for complete details about this implementing this interface.)

Procedures for configuring the serial control port for SPI and I²C communication modes are provided in this chapter.

2.4 SPI Port

The CS4953x4/CS4970x4 Serial Peripheral Interface (SPI) bus has been developed for 8-bit digital control applications, such as those requiring microcontrollers. SPI communication is accomplished with 5 lines: Serial Chip Select ($\overline{\text{SCP1_CS}}$), Serial Control Clock (SCP1_CLK), Master Out/Slave In data (SCP1_MOSI), and a Master In/Slave Out data (SCP1_MISO). Although the separate data I/O lines provide full-duplex capabilities, the CS4953x4/CS4970x4 chips only uses a half-duplex SPI-bus. Each

device on the bus may respond to one or more unique commands, and can operate as either a transmitter or receiver.

A device is considered the Master in a transaction if it drives the \overline{CS} pin of another device, and is also Mastering the SCP1_CLK line. A block diagram of the CS4953x4/CS4970x4 and CS4953x4/CS4970x4 SPI Serial Control Port is provided in [Figure 2-1](#).

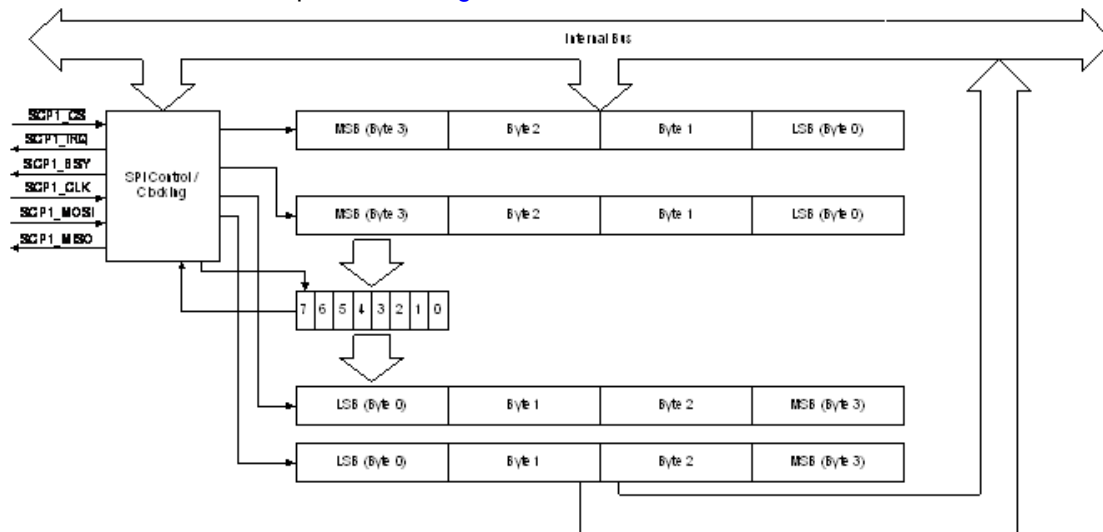


Figure 2-1. SPI Serial Control Port Internal Block Diagram

[Table 2-1](#) shows the signal names, descriptions, and pin number of the signals associated with the SPI Serial Control Port on the CS4953x4/CS4970x4.

Table 2-1. Serial Control Port SPI Signals

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SCP1_CS	SPI Chip Select, Active Low In serial SPI Slave mode, this pin is used as the active-low chip-select input signal. In SPI serial Master mode, if this pin is driven low by another Master device on the bus, it will cause a mode fault to occur.	96	6	Input
SCP1_CLK	SPI Control Port Bit Clock In Master mode, this pin serves as the serial control clock output. In serial Slave mode, this pin serves as the serial control clock input.	99	126	I/O
SCP1_MOSI	SPI Mode Master Data Output/Slave Data Input SCP1_MOSI in SPI Slave mode this pin serves as the data input, in SPI Master mode this pin serves as the data output.	95	3	I/O
SCP1_MISO	SPI Mode Master Data Input/Slave Data Output In SPI Slave mode this pin serves as the data input. In SPI Master mode this pin serves as the data output.	97	2	I/O
SCP1_IRQ	Serial Control Port Data Ready Interrupt Request Output, Active Low This pin is driven low when the DSP has a message for the host to read. The pin will go high when the host has read the message and the DSP has no further messages. This pin reflects the state of the SCP1 port Transmit Buffer Empty Flag.	100	4	Open Drain
SCP1_BSY	Serial Control Port 1 Input Busy, Output, Active Low This pin is driven low when the control port's receive buffer is full. This pin reflects the state of the SCP1 or PCP Receive Buffer Full Flag.	102	128	Open Drain
SCP2_CS	SPI Chip Select, Active Low In serial SPI Slave mode, this pin is used as the active-low chip-select input signal. In SPI serial Master mode, if this pin is driven low by another Master device on the bus, it will cause a mode fault to occur.	104	7	Input

Table 2-1. Serial Control Port SPI Signals (Continued)

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SCP2_CLK	SPI Control Port Bit Clock In Master mode, this pin serves as the serial control clock output. In serial Slave mode, this pin serves as the serial control clock input.	103	1	I/O
SCP2_MISO	SPI Mode Master Data Input/Slave Data Output In SPI Slave mode this pin serves as the data input. In SPI Master mode this pin serves as the data output.	105	2	I/O
SCP2_MOSI	SPI Mode Master Data Output/Slave Data Input SCP2_MOSI in SPI Slave mode this pin serves as the data input, in SPI Master mode this pin serves as the data output.	106	3	I/O
EE_CS	Master Mode SPI Flash Chip Select, Active Low	25	14	Output

2.4.1 SPI System Bus Description

The SPI bus is a multi-Master bus. This means that more than one device capable of controlling the bus can be connected to it. Generation of clock signals on the SPI bus is always the responsibility of Master devices; each Master generates its own clock signals when transferring data on the bus. Bus clock signals from a Master cannot be altered by any other device on the bus, otherwise a collision will occur. The Slave chip-select signals can only be controlled by Master devices.

The CS4953x4/CS4970x4 has two serial ports. The O/S currently supports only Slave mode host communication on SCP1, and Master mode communication on SCP2 for booting from a SPI Flash.

SCP1_MOSI (Master Out/Slave In) and SCP1_MISO (Master In/Slave Out) are bidirectional lines that change their behavior depending on whether the device is operating in Master or Slave mode. Only the Master can drive the MOSI signal while only the Slave can drive the MISO signal.

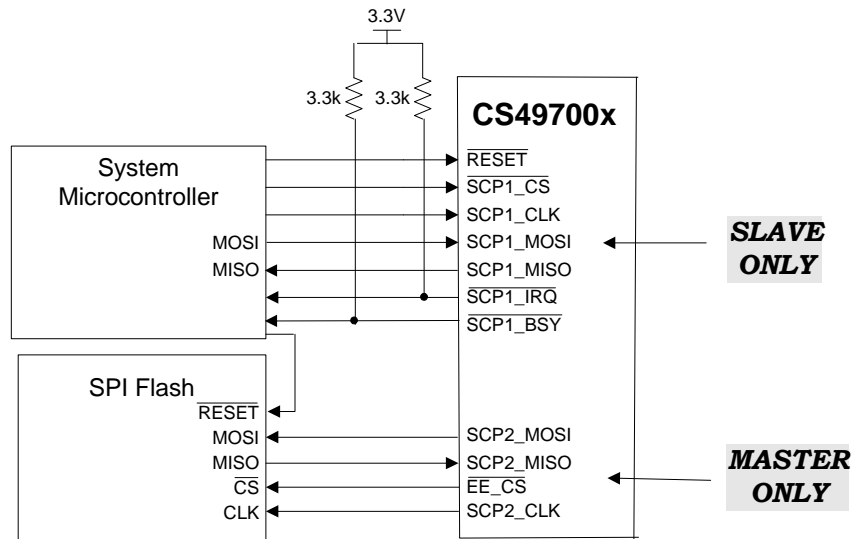


Figure 2-2. Block Diagram of SPI System Bus

As seen in Figure 2-2, two serial ports are available on the CS4953x4/CS4970x4. Each can be configured to either a Master or Slave. For audio applications, SCP1 is configured as a Slave port and SCP2 is configured as a Master port. SCP2 is used only in systems that are booting from SPI Flash.

2.4.2 SPI Bus Dynamics

A SPI transaction begins by the Master driving the Slave chip select SCP1_CS low. SPI transactions end by the Master driving the SCP1_CS high. This SPI bus is considered busy while any device's SCP1_CS

signal is low. The bus is free only when all Slave $\overline{\text{SCP1_CS}}$ signals are high. A high-to-low transition on the $\overline{\text{SCP1_CS}}$ line defines an SPI Start condition. A low-to-high transition on the $\overline{\text{SCP1_CS}}$ line defines an SPI Stop condition. Start and Stop conditions are always generated by the Master. The bus is considered to be busy after the Start condition. The bus is considered to be free again following the Stop condition.

The data bits of the SCP1_MOSI and SCP1_MISO line are valid on the rising edge of SCP1_CLK. It is the Slave's responsibility to accept or supply bytes on the bus at the rate at which the Master is driving SCP1_CLK.

All data put on the SCP1_MOSI and SCP1_MISO lines must be in 8-bit bytes. The number of bytes that can be transmitted per transfer is unrestricted. Data is transferred with the most-significant bit (MSB) first. For the CS4953x4/CS4970x4 Slave SPI port, the first byte is an address byte that is always sent by the Master after a Start condition. This address byte is an "I²C-type" command of a 7-bit address + a R/W bit. The 7-bit SPI address is 1000000b (0x80).

If the SPI transaction is a write from Master to the CS4953x4/CS4970x4 ($\overline{\text{R/W}} = 0$, Address = 0x80), then the Master will clock the SCP1_CLK signal and drive the SCP1_MOSI signal with data bytes for the CS4953x4/CS4970x4 to read. If the SPI transaction is a read to the Master from the CS4953x4/CS4970x4 ($\overline{\text{R/W}} = 1$, Address = 0x81), then the Master will drive the SCP1_CLK signal and read the SCP1_MISO signal with the data bytes from the CS4953x4/CS4970x4.

2.4.2.1 $\overline{\text{SCP1_BSY}}$ Behavior

The $\overline{\text{SCP1_BSY}}$ signal is not part of the SPI protocol, but it is provided so that the Slave can signal to the Master that it cannot receive any more data. A falling edge of the $\overline{\text{SCP1_BSY}}$ signal indicates the Master must halt transmission. Once the $\overline{\text{SCP1_BSY}}$ signal goes high, the suspended transaction may continue. The host must obey the $\overline{\text{SCP1_BSY}}$ pin or control data will be lost

2.4.3 SPI Messaging

Messaging to the CS4953x4/CS4970x4 using the SPI bus requires usage of all the information provided in the *SPI Bus Description* and *Bus Dynamics* above. For control and application image downloading, SPI transactions to the CS4953x4/CS4970x4 will involve 4-byte words. A detailed description of the serial SPI communication mode is provided in this section. This includes:

- A flow diagram and description for a serial SPI write
- A flow diagram and description for a serial SPI read

2.4.3.1 Performing a Serial SPI Write

Information provided in this section is intended as a functional description indicating how to perform an SPI write from an external device (Master) to the CS4953x4/CS4970x4 DSP (Slave). The system designer must ensure that all timing constraints of the SPI Write Cycle are met (see the CS4953x4/CS4970x4 datasheet for timing specifications). When performing an SPI write, the same protocol is used whether writing single-word messages to the boot firmware, writing multiple-word overlay images to the boot firmware, or writing multiple-word messages to the application firmware. The example shown in this section can be generalized to fit any SPI write situation.

The flow diagram shown in [Figure 2-3](#) illustrates the sequence of events that define the SPI write protocol.

[Figure 2-4](#) describes the Serial SPI Write protocol.

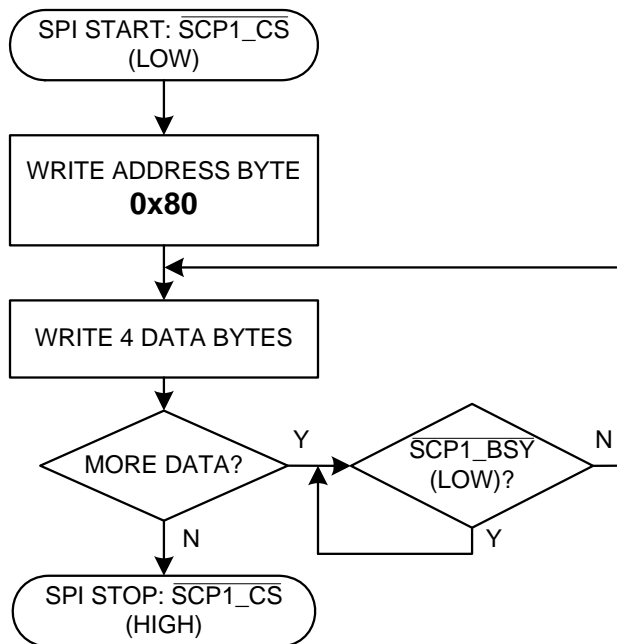


Figure 2-3. SPI Write Flow Diagram

2.4.3.2 SPI Write Protocol

1. A SPI transfer is initiated when the chip select $\overline{\text{SCP1_CS}}$ is driven low. $\overline{\text{SCP1_CS}}$ driven low indicates that CS4953x4/CS4970x4 is in SPI Slave mode.
2. This is followed by a 7-bit address and the read/write bit set low for a write. So, the Master should send 0x80. The 0x80 byte represents the 7-bit SPI address 1000000b, and the least significant bit set to '0', designates a write.
3. The Master should then clock the 4-byte data word into the Slave device, most-significant bit first, one byte at a time. The data byte is transferred to the CS4953x4/CS4970x4 DSP (Slave) on the falling edge of the eighth serial clock. For this reason, the serial clock should be held low so that eight transitions from low-to-high-to-low will occur for each byte.
4. If the Master has no more data words to write to the CS4953x4/CS4970x4, then proceed to **Step 6**. If the Master has more data words to write to the CS4953x4/CS4970x4, then proceed to **Step 5**.
5. The Master should poll the $\overline{\text{SCP1_BSY}}$ signal until it goes high. If the $\overline{\text{SCP1_BSY}}$ signal is low, it indicates that the CS4953x4/CS4970x4 is busy performing some task that requires halting the serial control port. Once the CS4953x4/CS4970x4 is able to receive more data words, the $\overline{\text{SCP1_BSY}}$ signal will go high. Once the $\overline{\text{SCP1_BSY}}$ signal is high, proceed to **Step 3**.
6. The Master finishes the SPI write transaction by driving the CS4953x4/CS4970x4 $\overline{\text{SCP1_CS}}$ signal high.

2.4.3.3 Performing a Serial SPI Read

Information provided in this section is intended as a functional description indicating how an external device (Master) performs an SPI read from the CS4953x4/CS4970x4 (Slave). The system designer must ensure that all timing constraints of the SPI read cycle are met (see the CS4953x4/CS4970x4 datasheet for timing specifications).

When performing a SPI read, the same protocol is used whether reading a single byte or multiple bytes. From a hardware perspective, it makes no difference whether communication is a single byte or multiple

bytes of any message length, so long as the correct hardware protocol is followed. The example shown in this section can be generalized to fit any SPI read situation.

The flow diagram shown in [Figure 2-4](#), illustrates the sequence of events that define the SPI read protocol. The Serial SPI read protocol is described in [Section 2.4.3.4](#).

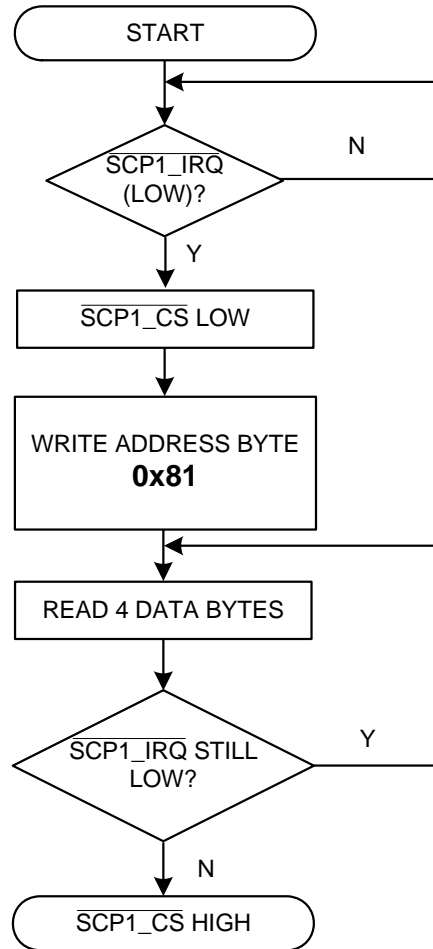


Figure 2-4. SPI Read Flow Diagram

2.4.3.4 SPI Read Protocol

1. A SPI read transaction is initiated by the CS4953x4/CS4970x4 Slave driving $\overline{\text{SCP1_IRQ}}$ low to indicate that it has data to be read.
2. The Master begins a SPI transaction driving chip select ($\overline{\text{SCP1_CS}}$) low.
3. This is followed by a 7-bit address and the read/write bit set high for a read. So, the Master should send 0x81. The 0x81 byte represents the 7-bit SPI address 1000000b, and the least significant bit set to '1', designates a read.
4. After the falling edge of the serial control clock (SCP1_CLK) for the read/write bit, the Master can begin clocking out the 4-byte word from the CS4953x4/CS4970x4 on the MISO pin. Data clocked out of the CS4953x4/CS4970x4 by the Master is valid on the rising edge of SCP1_CLK and data transitions occur on the falling edge of SCP1_CLK. The serial clock should be held low so that eight transitions from low-to-high-to-low will occur for each byte.

-
5. If $\overline{\text{SCP1_IRQ}}$ is still low after 4 bytes, then proceed to **Step 4** and read another 4 bytes out of the CS4953x4/CS4970x4 Slave.
 6. If $\overline{\text{SCP1_IRQ}}$ is high, the $\overline{\text{SCP1_CS}}$ line of CS4953x4/CS4970x4 should be driven high to end the read transaction.

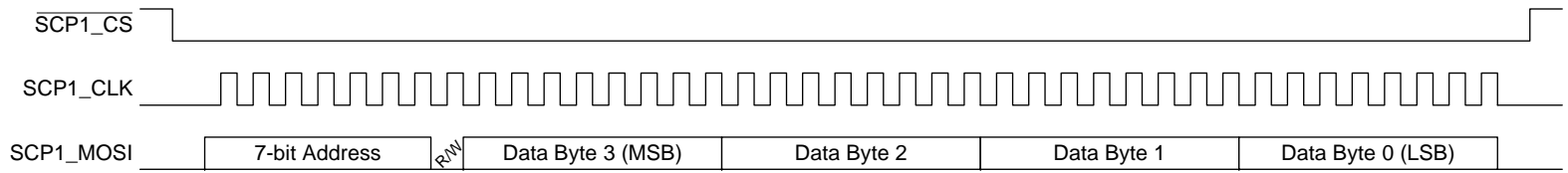


Figure 2-5. Sample Waveform for SPI Write Functional Timing

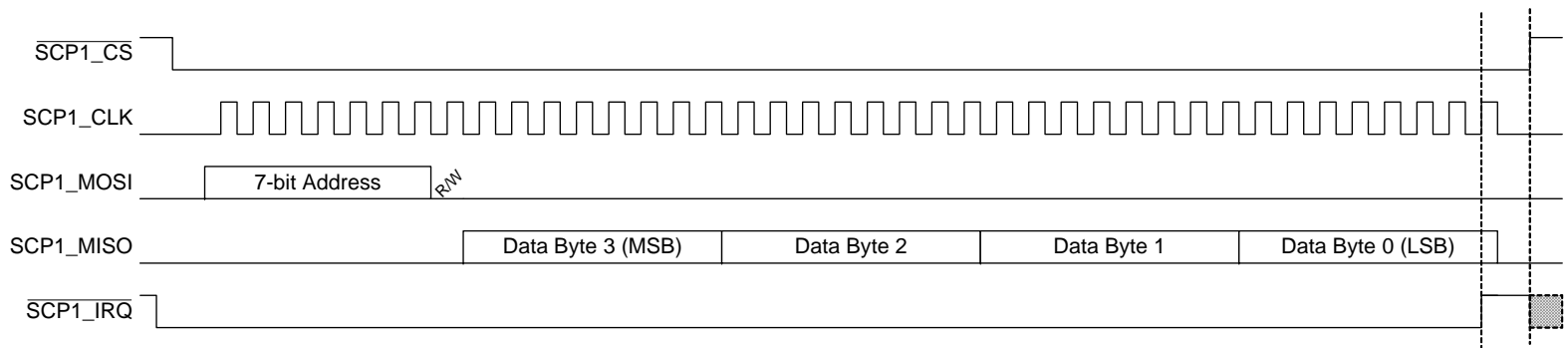


Figure 2-6. Sample Waveform for SPI Read Functional Timing

Note:

1. IRQ remains low until the rising edge of the clock for the last bit of the last byte to be read from the SPI Slave.
2. After going high, IRQ remains high until the $\overline{\text{CS}}$ signal is raised to end the SPI transaction. If there are more bytes to read, IRQ will fall after CS has gone high.

2.4.3.5 SCP1_IRQ Behavior

The SCP1_IRQ signal is not part of the SPI protocol, but is provided so that the Slave can signal that it has data to be read. A high-to-low transition on SCP1_IRQ indicates to the Master that the Slave has data to be read. When a Master detects a high-to-low transition on SCP1_IRQ, it should send a Start condition and begin reading data from the Slave.

SCP1_IRQ is guaranteed to remain low (once it has gone low), until the rising edge of SCP1_CLK for the last bit of the last byte to be transferred out of the CS4953x4/CS4970x4. If there is no more data to be transferred, SCP1_IRQ will go high at this point. After going high, SCP1_IRQ is guaranteed to stay high until the rising edge of SCP1_CS.

This end-of-transfer condition signals the Master to end the read transaction by clocking the last data bit out of the CS4953x4/CS4970x4 and then driving the CS4953x4/CS4970x4 SCP1_CS line high to signal that the read sequence is over. If SCP1_IRQ is still low after the rising edge of SCP1_CLK on the last data bit of the current byte, the Master should continue reading data from the serial control port. It should be noted that all data should be read out of the serial control port during one cycle or a loss of data will occur. In other words, all data should be read out of the chip until SCP1_IRQ signals the last byte by going high as described above.

2.5 I²C Port

The CS4953x4/CS4970x4 I²C bus has been developed for 8-bit digital control applications, such as those requiring microcontrollers. The I²C bus interface is a bidirectional serial port that uses 2 lines (data and clock) for data transmission and reception with software-addressable external devices. Each external device interfaced to the CS4953x4/CS4970x4 I²C port has the ability to communicate directly with the other devices and is assigned a unique address whether it is a CPU, memory, or some other device. A block diagram of the CS4953x4/CS4970x4 I²C Serial Control Port is provided in Figure 2-7.

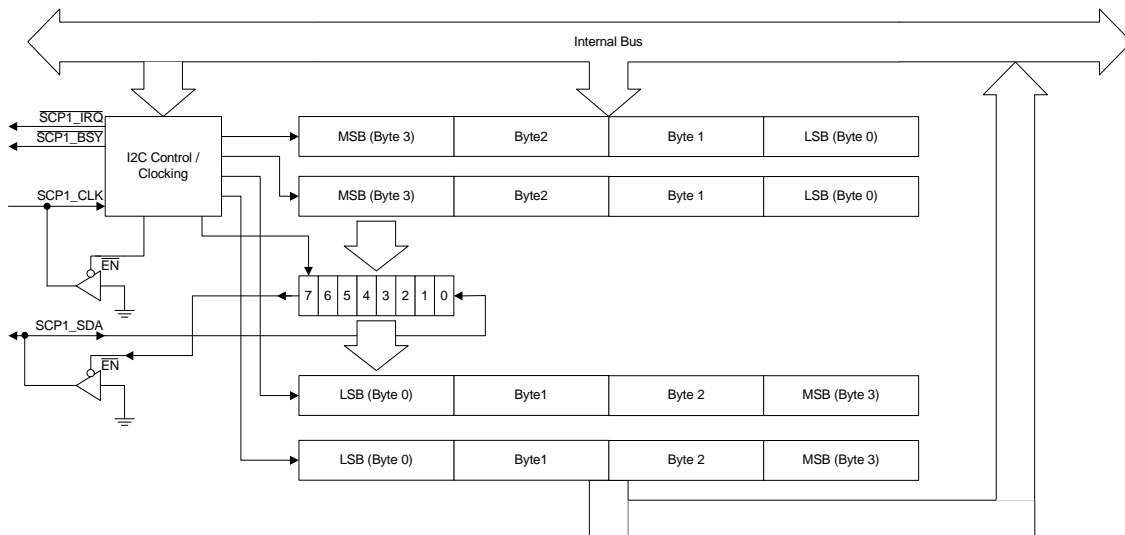


Figure 2-7. Serial Control Port Internal Block Diagram

2.5.1 I²C System Bus Description

Devices can be considered Masters or Slaves when performing data transfers. A Master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device addressed by the initiator is considered a Slave.

The CS4953x4/CS4970x4 has two serial ports. The O/S currently supports only Slave mode host communication on SCP1, and Master mode communication on SCP2 for booting from a SPI Flash.

Both SCP1_SDA and SCP1_CLK are bidirectional lines. When the bus is free, both lines are pulled high by resistors. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

As seen in [Figure 2-8](#), two serial ports are available on the CS4953x4/CS4970x4. SCP1 is configured as Slave and SCP2 is configured as a Master. SCP2 is used only in systems that are booting from SPI Flash.

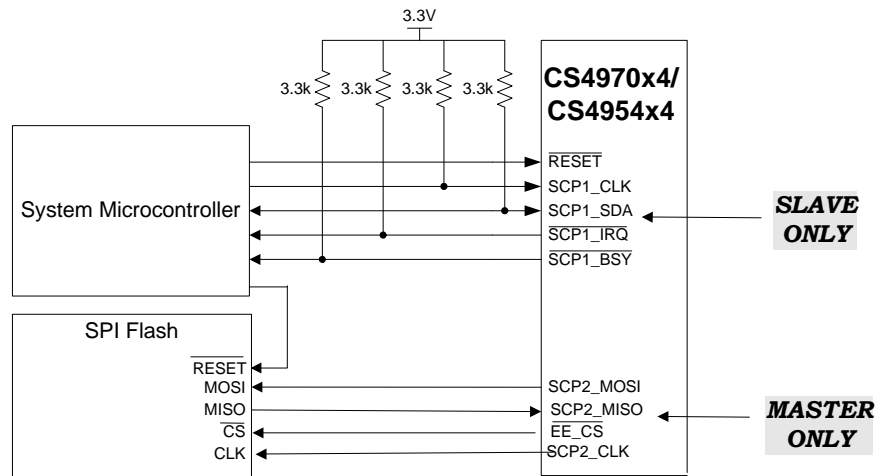


Figure 2-8. Block Diagram of I²C System Bus

[Table 2-2](#) shows the signal names, descriptions, and pin number of the signals associated with the I²C Serial Control Port on the CS4953x4/CS4970x4.

Table 2-2. Serial Control Port 1 I²C Signals

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SCP1_CLK	I ² C Control Port Bit Clock. In Master mode, this pin serves as the serial control clock output (open drain in I ² C mode / output in SPI mode). In serial Slave mode, this pin serves as the serial control clock input. In I ² C Slave mode the clock can be pulled low by the port to stall the Master.	99	126	Open Drain
SCP1_SDA	Bidirectional Data I ² C Mode Master/Slave Data IO. In I ² C Master and Slave mode, this open drain pin serves as the data input and output.	97	124	Open Drain
SCP1_IRQ	Control Port Data Ready Interrupt Request, Output, Active Low This pin is driven low when the DSP has a message for the host to read. The pin will go high when the host has read the message and the DSP has no further messages.	100	4	Open Drain

Table 2-2. Serial Control Port 1 I²C Signals (Continued)

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SCP1_BSY	Serial Control Port 1 Input Busy, Output, Active Low This pin is driven low when the control port's receive buffer is full. Internal Buffer is 4 bytes (1 DSP Word) deep.	102	128	Open Drain
SCP2_CS	SPI Chip Select, Active Low In serial SPI Slave mode, this pin is used as the active-low chip-select input signal. In SPI serial Master mode, if this pin is driven low by another Master device on the bus, it will cause a mode fault to occur.	104	7	Input
SCP2_CLK	SPI Control Port Bit Clock In Master mode, this pin serves as the serial control clock output. In serial Slave mode, this pin serves as the serial control clock input.	103	1	I/O
SCP2_MISO	SPI Mode Master Data Input/Slave Data Output In SPI Slave mode this pin serves as the data input. In SPI Master mode this pin serves as the data output.	105	2	I/O
SCP2_MOSI	SPI Mode Master Data Output/Slave Data Input SCP2_MOSI in SPI Slave mode this pin serves as the data input, in SPI Master mode this pin serves as the data output.	106	3	I/O
EE_CS	Master Mode SPI Flash Chip Select, Active Low	25	14	Output

2.5.2 I²C Bus Dynamics

The Start condition for an I²C transaction is defined as the first falling edge on the SCP1_SDA line while SCP1_CLK is high. An I²C Stop condition is defined as the first rising edge on the SCP1_SDA line while SCP1_CLK is high. Hence for valid data transfer, SCP1_SDA must remain stable during the high period of the clock pulse. Start and Stop conditions are always generated by the Master. The bus is considered to be busy after the Start condition. The bus is considered to be free again following the Stop condition. The bus stays busy if a repeated Start condition is generated instead of a Stop condition. In this respect, the Start and repeated Start conditions are functionally identical.

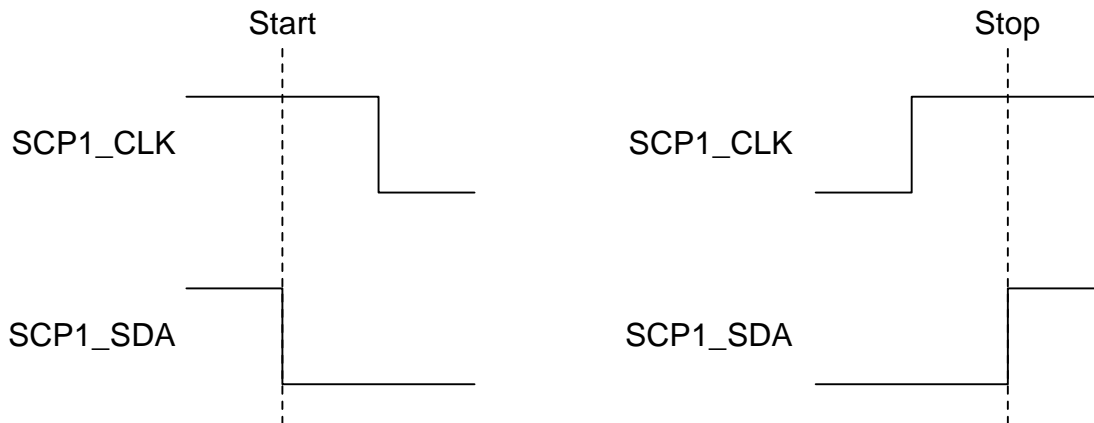


Figure 2-9. I²C Start and Stop Conditions

The number of bytes that can be transmitted per transfer is unrestricted. Data is transferred with the most-significant bit (MSB) first. The first byte is an address byte that is always sent by the Master after a Start or repeated Start condition. This byte must be a 7-bit I²C Slave address + R/W bit. The 7-bit I²C address for

the CS4953x4/CS4970x4 is 1000000b (0x80). The $\overline{R/W}$ bit is used to notify the Slave if the current transaction is for the Master to write data to the Slave ($\overline{R/W} = 0$) or read data from the Slave ($\overline{R/W} = 1$).

After the Master has sent the address byte, the Master releases the SCP1_SDA line. If the Slave received the address byte, it will drive the SCP1_SDA line low to acknowledge (ACK) to the Master that the byte was received. The SCP1_SDA line must remain stable and low during the high period of the next clock pulse. When a Slave doesn't acknowledge the Slave address, the data line must be left high by the Slave (NACK).

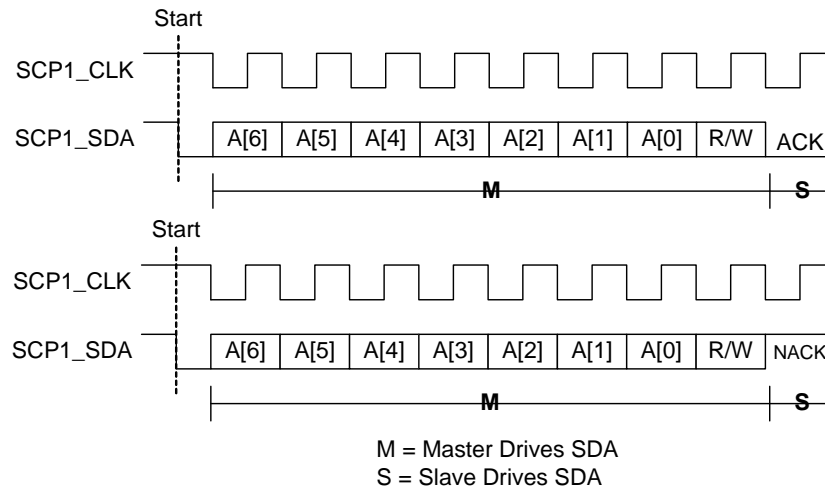


Figure 2-10. I²C Address with ACK and NACK

For write operations, the $\overline{R/W}$ bit must be set to zero ($\overline{R/W} = 0$, Address = 0x80). After the 8-bit data byte has been clocked, the Master will release the SCP1_SDA line. If the Slave received the byte correctly, it will drive the SCP1_SDA line low for the next bit clock to acknowledge (ACK) that the data was received. If the data was not received correctly, the Slave can communicate this by leaving the SCP1_SDA line high (NACK).

For read operations, the $\overline{R/W}$ bit must be set to one ($\overline{R/W} = 1$, Address = 0x81), then the Master will read a data byte from the Slave. After the 8-bit data byte has been clocked, the Master will release the SCP1_SDA line. If the Master received the byte correctly, it will drive the SCP1_SDA line low for the next bit clock to acknowledge (ACK) that the data was received. If the data was not received correctly, the Master can communicate this by leaving the SCP1_SDA line high (NACK). The protocol on the last byte, however, is different. When the Master receives the last byte, it signals the end of the data to the Slave by allowing SCP1_SDA to float high (NACK).

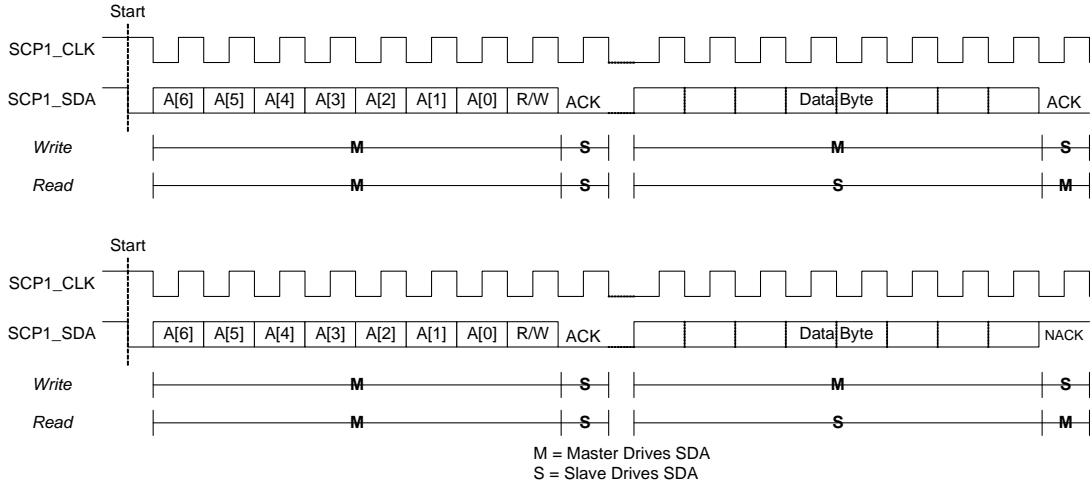


Figure 2-11. Data Byte with ACK and NACK

After an ACK or NACK from the Master or Slave, the Slave must leave the SCP1_SDA line high so the Master can then generate either another Start condition as shown in [Figure 2-12](#) to start a new transfer or a Stop condition as shown in [Figure 2-13](#) to abort the transfer.

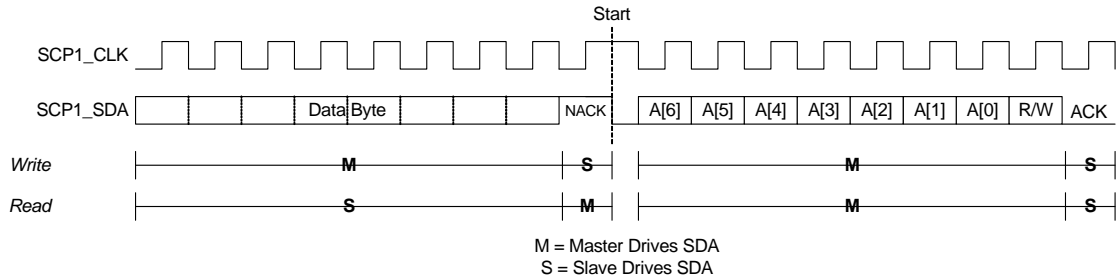


Figure 2-12. Repeated Start Condition with ACK and NACK

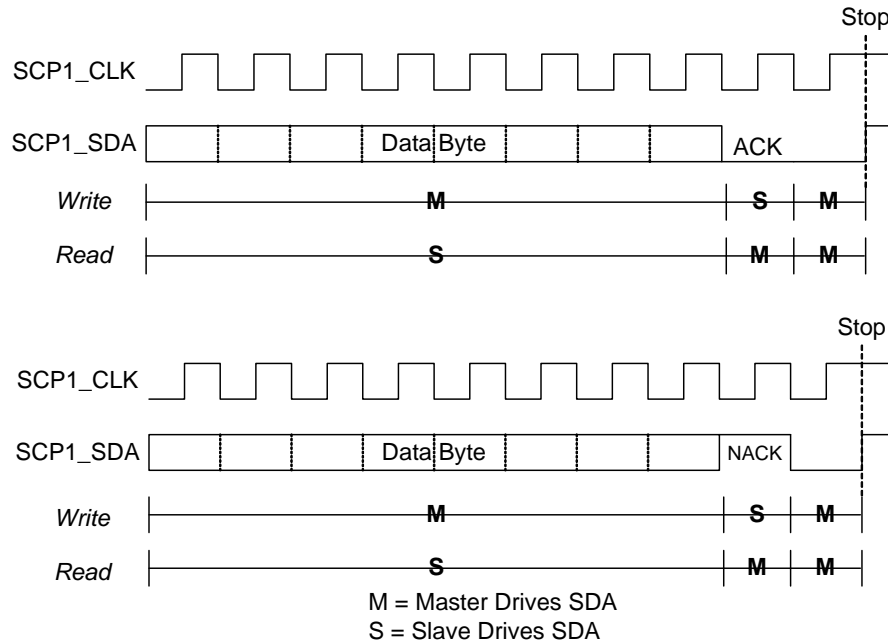


Figure 2-13. Stop Condition with ACK and NACK

If a Slave can't receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the SCP1_CLK line low to force the Master into a wait state. Data transfer then continues when the Slave is ready for another byte of data and releases SCP1_CLK.

2.5.3 I²C Messaging

Messaging to the CS4953x4/CS4970x4 using the I²C bus requires usage of all the information provided in the above I²C [Section 2.5.1 "I²C System Bus Description"](#) on page 10 and [Section 2.5.2 "I²C Bus Dynamics"](#) on page 11. Every I²C transaction to the CS4953x4/CS4970x4 involves 4-byte words used for control and application image download. A detailed description of the serial SPI communication mode is provided in this section. This includes:

- A flow diagram and description for a serial I²C write
- A flow diagram and description for a serial I²C read

2.5.3.1 SCP1_BSY Behavior

The $\overline{\text{SCP1_BSY}}$ signal is not part of the I²C protocol, but it is provided so that the Slave can signal to the Master that it cannot receive any more data. A falling edge of the $\overline{\text{SCP1_BSY}}$ signal indicates the Master must halt transmission. Once the $\overline{\text{SCP1_BSY}}$ signal goes high, the suspended transaction may continue. It is important for the host to obey the $\overline{\text{SCP1_BSY}}$ pin status for proper communication with the DSP.

2.5.3.2 Performing a Serial I²C Write

Information provided in this section is intended as a functional description indicating how to use the configured serial control port to perform a I²C write from an external device (Master) to the CS4953x4/CS4970x4 DSP (Slave). The system designer must ensure that all timing constraints of the I²C write cycle are met (see the CS4953x4/CS4970x4 datasheet for timing specifications). When writing to the CS4953x4/CS4970x4, the same protocol described in this section will be used when writing single-word messages to the boot firmware, writing multiple-word overlay images to the boot firmware, and writing multiple-word messages to application firmware. The examples given can therefore be expanded to fit any I²C writing situation.

The flow diagram shown in [Figure 2-14](#) illustrates the sequence of events that define the I²C write protocol for SCP1. [Section 2.5.3.3.](#) describes the Serial I²C Write protocol.

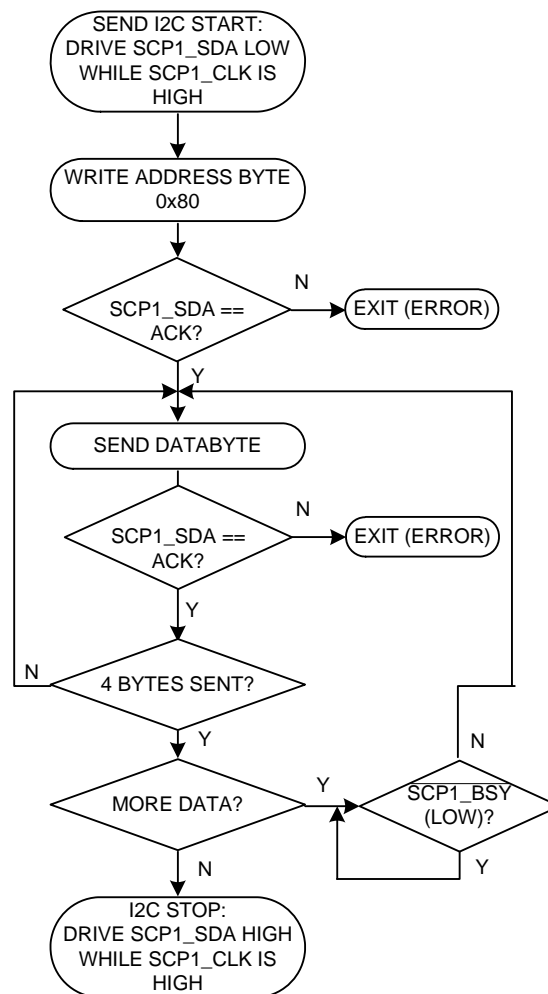


Figure 2-14. I²C Write Flow Diagram

2.5.3.3 I²C Write Protocol

1. An I²C transfer is initiated with an I²C start condition which is defined as the data (SCP1_SDA) line falling while the clock (SCP1_CLK) is held high.
2. This is followed by a 7-bit address and the read/write bit held low for a write. So, the Master should send 0x80. The 0x80 byte represents the 7-bit I²C address 1000000b, and the least significant bit set to '0', designates a write.
3. After each byte (including the address and each data byte) the Master must release the data line and provide a ninth clock for the CS4953x4/CS4970x4 DSP (Slave) to acknowledge (ACK) receipt of the byte. The CS4953x4/CS4970x4 drive the data line low during the ninth clock to acknowledge. If for some reason the CS4953x4/CS4970x4 does not acknowledge (NACK), it means that the communications channel has been corrupted and the CS4953x4/CS4970x4 should be re-booted. A NACK should never happen here.
4. The Master should then clock one data byte into the device, most-significant bit first.
5. The CS4953x4/CS4970x4 (Slave) will (and must) acknowledge (ACK) each byte that it receives which means that after each byte, the Master must provide an acknowledge clock pulse on SCP1_CLK and release the data line, SCP1_SDA.
6. If the Master has no more data words to write to the CS4953x4/CS4970x4, then proceed to **Step 8**. If the Master has more data words to write to the CS4953x4/CS4970x4, then proceed to **Step 7**.
7. The Master should poll the $\overline{\text{SCP1_BSY}}$ signal until it goes high. If the $\overline{\text{SCP1_BSY}}$ signal is low, it indicates that the CS4953x4/CS4970x4 is busy performing some task that requires pausing the serial control port. Once the CS4953x4/CS4970x4 is able to receive more data words, the $\overline{\text{SCP1_BSY}}$ signal will go high. Once the $\overline{\text{SCP1_BSY}}$ signal is high, proceed to **Step 4**.

Note: The DSP's I²C port also implements clock stretching to indicate that the host should pause communication. So the host has the option of checking for SCP1_CLK held low rather than $\overline{\text{SCP1_BSY}}$ low.

8. At the end of a data transfer, a stop condition must be sent. The stop condition is defined as the rising edge of SCP1_SDA while SCP1_CLK is high.

2.5.3.4 Performing a Serial I²C Read

Information provided in this section is intended as a functional description indicating how to use the configured serial control port to perform an I²C read from an external device (Master) to the CS4953x4/CS4970x4 DSP (Slave). The system designer must ensure that all timing constraints of the I²C Read Cycle are met (see the CS4953x4/CS4970x4 datasheet for timing specifications). I²C read transactions from the CS4953x4/CS4970x4 will always involve reading 4-byte words.

Figure 2-15 illustrates the sequence of events that define the I²C read protocol for SCP1. This protocol is discussed in the high-level procedure found in [Section 2.5.3.5](#).

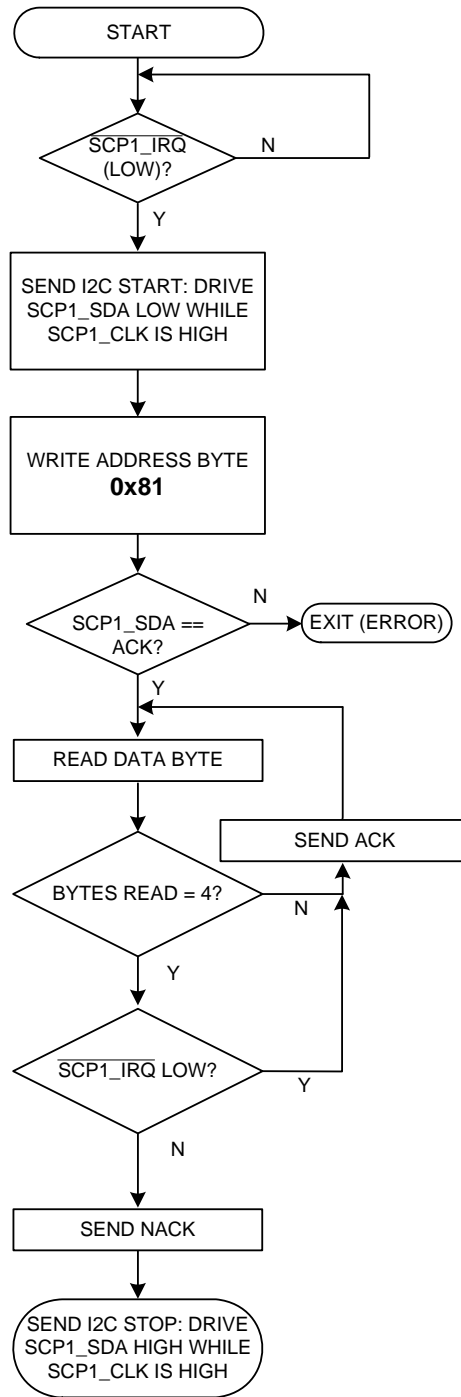
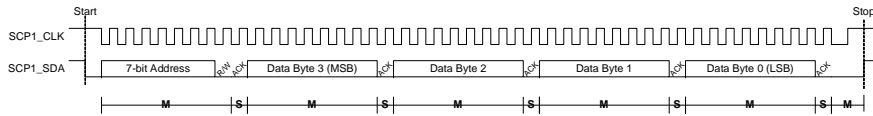


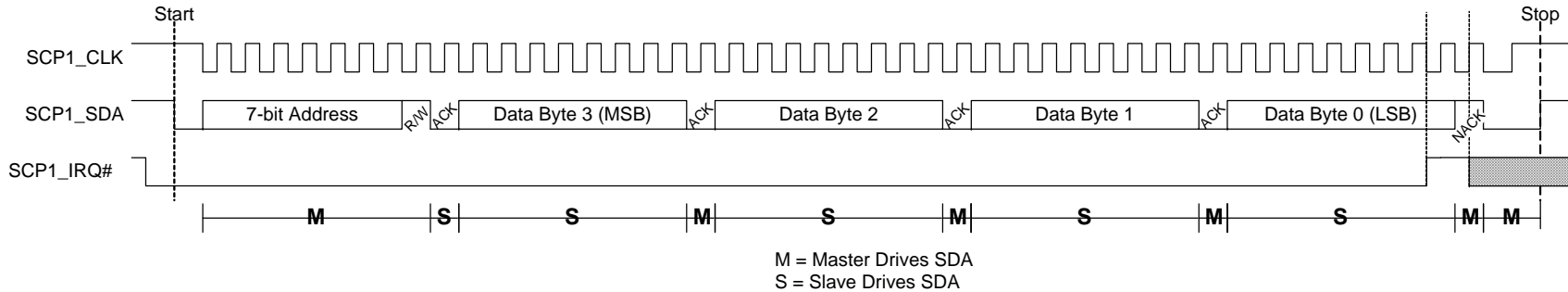
Figure 2-15. I²C Read Flow Diagram

2.5.3.5 I²C Read Procedure

1. An I²C read transaction is initiated by CS4953x4/CS4970x4 driving $\overline{\text{SCP1_IRQ}}$ low, signaling that it has data to be read.
2. The Master responds by sending an I²C Start condition which is SCP1_SDA going low while SCP1_CLK is held high.
3. This is followed by a 7-bit address and the read/write bit set high for a read. So, the Master should send 0x81. The 0x81 byte represents the 7-bit I²C address 1000000b, and the least significant bit set to '1', designates a read.
4. After the address byte, the Master must release the data line and provide a ninth clock for the CS4953x4/CS4970x4 DSP (Slave) to acknowledge (ACK) receipt of the byte. The CS4953x4/CS4970x4 will drive the data line low during the ninth clock to acknowledge. If for some reason CS4953x4/CS4970x4 does not acknowledge (NACK), it means that the communications channel has been corrupted and the CS4953x4/CS4970x4 should be re-booted. A NACK should never happen here.
5. The data is ready to be clocked out on the SCP1_SDA line at this point. Data clocked out by the host is valid on the rising edge of SCP1_CLK and data transitions occur just after the falling edge of SCP1_CLK.
6. After the CS4953x4/CS4970x4 has written the byte to the Master on the SCP1_SDA line, it will release the SCP1_SDA line. If the Master has more bytes in the 4-byte word to read, then proceed to Step 7. If the Master is finished reading all bytes of the 4-byte word, then proceed to **Step 8**.
7. The Master should drive the SCP1_SDA line low for the 9th SCP1_CLK clock to acknowledge (ACK) that the byte was received from the CS4953x4/CS4970x4. The Master should then return to Step 5 to read another byte of the 4-byte word.
8. If $\overline{\text{SCP1_IRQ}}$ is still low after reading a 4-byte word, proceed to Step 7. If $\overline{\text{SCP1_IRQ}}$ has risen, proceed to **Step 9**.
9. The Master should let the SCP1_SDA line stay high for the 9th SCP1_CLK clock as a no-acknowledge (NACK) to CS4953x4/CS4970x4. This, followed by an I²C stop condition (SCP1_SDA driven high, while SCP1_CLK is high) signals an end of read to CS4953x4/CS4970x4. See [Section 2.4.3.5 on page 2-9](#) for an in-depth explanation of $\overline{\text{SCP1_IRQ}}$.

Figure 2-16. Sample Waveform for I²C Write Functional Timing

Note: The I²C Slave is always responsible for driving the ACK for the address byte.

Figure 2-17. Sample Waveform for I²C Read Functional Timing

Note:

1. The I²C Slave drives the ACK for the address byte.
2. The I²C Master is responsible for controlling ACK during I²C reads. In general, the receiver in an I²C transaction is responsible for providing ACK.
3. SCP1_IRQ# remains low until the rising edge of the clock for the last bit of the last byte read from the I²C Slave.
4. A NACK is sent by the Master after the last byte to indicate the end of the read cycle. This must be followed with an I²C Stop condition or I²C Repeated-Start condition.
5. If there are more data words to read, IRQ will fall at the rising edge of CLK for the NACK. Otherwise, IRQ remains high until an I²C Stop condition or an I²C Repeated-Start condition occurs.

2.5.3.6 SCP1_IRQ Behavior

Once the BOOT_ASSIST_A (.ULD file) has been downloaded in accordance to Steps 1 through 8 in [Section 2.3.1.](#) or Steps 1 through 8 in [Section 2.3.2.](#), the SCP1_IRQ pin is functionally enabled.

The SCP1_IRQ signal is not part of the I²C protocol, but is provided so that the Slave can signal that it has data to be read. A high-to-low transition on SCP1_IRQ indicates to the Master that the Slave has data to be read. When a Master detects a high-to-low transition on SCP1_IRQ, it should send a Start condition and begin reading data from the Slave.

SCP1_IRQ is guaranteed to remain low (once it has gone low), until the falling edge of SCP1_CLK for the last bit of the last byte to be transferred out of CS4953x4/CS4970x4 (that is, the rising edge of SCP1_CLK before the ACK). If there is no more data to be transferred, SCP1_IRQ will go high at this point. After going high, SCP1_IRQ is guaranteed to stay high until the next rising edge of SCP1_CLK (that is, it will stay high until the rising edge of SCP1_CLK for the ACK/NACK bit).

This end-of-transfer condition signals the Master to end the read transaction by clocking the last data bit out of CS4953x4/CS4970x4 and then sending a NACK to CS4953x4/CS4970x4 to signal that the read sequence is over. At this point, the Master should send an I²C stop condition to complete the read sequence. If SCP1_IRQ is still low after the rising edge of SCP1_CLK on the last data bit of the current byte, the Master should send an acknowledge and continue reading data from the serial control port. It should be noted that all data should be read out of the serial control port during one cycle or a loss of data will occur. In other words, all data should be read out of the chip until SCP1_IRQ signals the last byte by going high as described above.

§§

Chapter 3

Audio Input Interfaces

3.1 Introduction

CS4953x4/CS4970x4 support a wide variety of audio data formats through various input and output ports. Hardware availability is entirely dependent on whether the software application code being used supports the required mode. This data sheet presents most of the modes available with the CS4953x4/CS4970x4 hardware. However, not all of the modes are available with any particular piece of application code. The Application Note that serves as the User's Guide for the particular code being used should be consulted when determining if a particular mode is supported. In addition, if a particular mode is desired that is not presented here or in the Cirrus Logic firmware application note, please contact your sales representative regarding its availability.

3.2 Digital Audio Input Port Description

The CS4953x4/CS4970x4 Digital Audio Input ports (DAI1 and DAI2) are designed to provide five digital audio input pins (10 channels total) that can be configured to load audio samples in a number of formats, or accept multiple stereo channels on a single input port.

DAI features include:

- Five digital audio input pins capable of supporting many audio formats
- Two independent input clock domains (DAI1_SCLK/DAI1_LRCLK or DAI2_SCLK/DAI2_LRCLK)
- Up to 32-bit data widths
- Sample rates up to 192 KHz
- Two simultaneous serial compressed audio data inputs in IEC61937 format for dual-decode support
- 2, 4, or 6 channels on a single pin (DAI_DATAx)
- Simultaneous 8-channel PCM for multichannel DVD audio and compressed audio

3.2.1 DAI Pin Description

The digital audio input port (DAI) on CS4953x4/CS4970x4, is used for both compressed and PCM digital audio data input. [Table 3-1](#) shows the mnemonic and pin description of the pins associated with the DAI port on the CS4953x4/CS4970x4.

Table 3-1. Digital Audio Input Port

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
DAI1_LRCLK	Sample Rate Clock 1 PCM Audio Input Sample Rate (LeftRight) Clock DAI1_LRCLK is the sample rate input clock for the serial PCM audio data on DAI_DATA[3:0].	138	30	Input
DAI1_SCLK	Bit Clock 1 PCM Audio Input Bit Clock DAI1_SCLK is the bit clock input for the serial PCM audio data on DAI_DATA[3:0].	137	29	Input

Table 3-1. Digital Audio Input Port (Continued)

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
DAI1_DATA0	PCM or Compressed Audio Input Data 0 PCM Audio Input Data 0 Serial data input that can accept PCM audio data that is synchronous to DAI_SCLK1/DAI_LRCLK1 or DAO1_SCLK/DAO1_LRCLK..	135	27	Input
DAI1_DATA1	PCM or Compressed HDAudio Input Data 1	134	26	Input
DAI1_DATA2	PCM or Compressed HD Audio Input Data 2	132	24	Input
DAI1_DATA3	PCM or Compressed HD Audio Input Data 3	131	23	Input
DAI2_LRCLK	Sample Rate Clock 2 PCM Audio Input Sample Rate (LeftRight) Clock DAI2_LRCLK is the sample rate input clock for the serial PCM audio data on DAI2_DATA.	140	32	Input
DAI2_SCLK	Bit Clock 2 PCM Audio Input Bit Clock DAI2_SCLK is the bit clock input for the serial PCM audio data on DAI2_DATA.	141	33	Input
DAI2_DATA or DAI1_DATA4	PCM or Compressed Audio Input Data PCM Audio Input Data DAI2_DATA is the PCM audio data input synchronous to DAI2_SCLK/DAI2_LRCLK	142	34	Input

3.2.2 Supported DAI Functional Blocks

The CS4953x4/CS4970x4 DAI has many functional blocks for realizing various audio system configurations. The use of these functions is dependent on the firmware currently available on the CS4953x4/CS4970x4. [Figure 3-1](#) shows the functional block diagram of the features currently supported with the CS4953x4/CS4970x4 DAI.

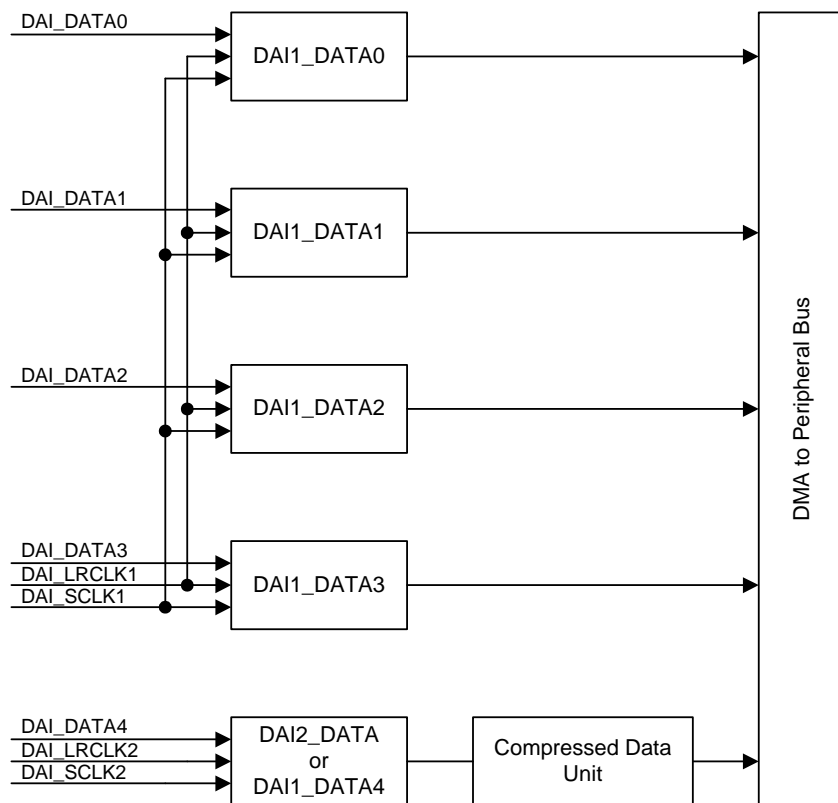


Figure 3-1. DAI Port Block Diagram

Currently supported are 4 lines of linear PCM input (DAI_DATA[3:0]) and 1 line of compressed audio or linear PCM (DAI_DATA4). These two inputs can have their own clock domains. The firmware currently available can operate on only one of these inputs at a time, providing for compressed data decode, stereo PCM processing, or multichannel PCM processing. Please see AN288, "CS4953x4/CS4970x4 Firmware User's Manual" for details about configuring the firmware to select these different inputs to process.

3.2.3 BDI Port

Note: Currently not supported in the O/S.

The Bursty Data Input (BDI) port on the CS4953x4/CS4970x4 shares pins with the DAI port pins, and is used for input of bursty compressed audio data. The compressed data is clocked in with a bit clock (BDI_CLK). Bursty compressed audio data input requires the use of a "throttle" signal, BDI_REQ to signal to the host that the CS4953x4/CS4970x4 is capable of accepting data.

Table 3-2. Bursty Data Input (BDI) Pins

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
$\overline{\text{BDI_REQ}}$	Data Request, Active Low BDI_REQ is the bursty delivery flow control output for bursty audio data. It indicates whether the DSP can accept more data.	140	32	Output
BDI_CLK	Bit Clock 2 Bursty Audio Input Bit Clock BDI_CLK is the bit clock input for the bursty serial audio data on BDI_DATA.	141	33	Input
BDI_DATA	Compressed Audio Bursty Input Data BDI_DATA is the serial bursty audio data input that corresponds to the BDI_CLK serial bit clock..	142	34	Input

3.2.4 Digital Audio Formats

The DAI has 5 stereo data input pins that are fully configurable including support for I²S, and left-justified formats. DAI ports are programmed for Slave operation, where DAIn_LRCLK and DAIn_SCLK are inputs only. This subsection describes some common audio formats that CS4953x4/CS4970x4 supports. It should be noted that the input ports use up to 32-bit PCM resolution and 16-bit compressed data word lengths.

3.2.4.1 I²S Format

For I²S, data is presented most-significant bit (MSB) first, one SCLK delay after the transition of DAIn_LRCLK, and is valid on the rising edge of DAIn_SCLK. For the I²S format, the left subframe is presented when DAIn_LRCLK is low, and the right subframe is presented when DAIn_LRCLK is high.

3.2.4.2 Left-Justified Format

Figure 3-2 illustrates the left-justified format with a rising-edge DAIn_SCLK. Data is presented most-significant bit first on the first DAIn_SCLK after a DAIn_LRCLK transition and is valid on the rising edge of DAIn_SCLK. For the left-justified format, the left subframe is presented when DAIn_LRCLK is high and

the right subframe is presented when DAI_n_LRCLK is low. The left-justified format can also be programmed for data to be valid on the falling edge of DAI_n_SCLK.

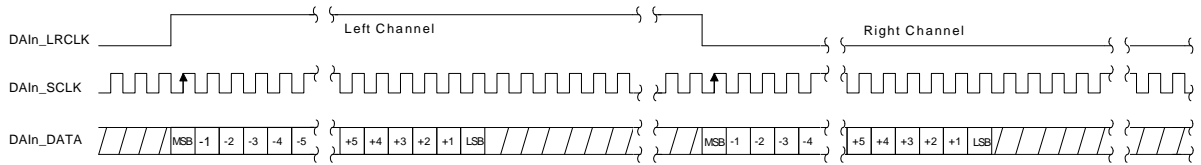


Figure 3-2. Left-justified Format (Rising Edge Valid SCLK)

3.3 DAI Hardware Configuration

DAI hardware configuration must be done at design time using DSP Composer. See [Chapter 8, "DSP Condenser"](#) for more details.

3.3.1 DAI Hardware Naming Convention

The naming convention of the input hardware configuration is as follows:

INPUT *A B C D*

Where *A*, *B*, *C*, and *D* are the parameters used to fully define the input port. The parameters are defined as follows:

- **A** - Data Format
- **B** - SCLK Polarity
- **C** - LRCLK Polarity.
- **D** - DAI Mode

[Table 3-3](#), [Table 3-4](#), [Table 3-5](#), and [Table 3-6](#) show the different values for each parameter as well as the hex message that needs to be sent to configure the port. When creating the hardware configuration message, only one hex message should be sent per parameter.

Table 3-3. Input Data Format Configuration (Input Parameter A)

A Value	Data Format	Hex Message
0 (default)	I ² S 24-bit	Starting from DAI_D0 to DAI_D4: 0x81800020 0xFFFF0000 0x81400020 0x01001F00 0x81800021 0xFFFF0000 0x81400021 0x01001F00 0x81800022 0xFFFF0000 0x81400022 0x01001F00 0x81800023 0xFFFF0000 0x81400023 0x01001F00 0x81800024 0xFFFF0000 0x81400024 0x01001F00
1	Left Justified 24-bit	Starting from DAI_D0 to DAI_D4: 0x81800020 0xFEEF0000 0x81400020 0x00001F00 0x81800021 0xFEEF0000 0x81400021 0x00001F00 0x81800022 0xFEEF0000 0x81400022 0x00001F00 0x81800023 0xFEEF0000 0x81400023 0x00001F00 0x81800024 0xFEEF0000 0x81400024 0x00001F00

Table 3-4. Input SCLK Polarity Configuration (Input Parameter B)

B Value	SCLK Polarity (Both DAI and CDI Port)	Hex Message
0 (default)	Data Clocked in on SCLK Rising Edge	Starting from DAI_D0 to DAI_D4: 0x81800020 0xFFDFFFFFFF 0x81800021 0xFFDFFFFFFF 0x81800022 0xFFDFFFFFFF 0x81800023 0xFFDFFFFFFF 0x81800024 0xFFDFFFFFFF

Table 3-4. Input SCLK Polarity Configuration (Input Parameter B) (Continued)

B Value	SCLK Polarity (Both DAI and CDI Port)	Hex Message
1	Data Clocked in on SCLK Falling Edge	Starting from DAI_D0 to DAI_D4: 0x81400020 0x00200000 0x81400021 0x00200000 0x81400022 0x00200000 0x81400023 0x00200000 0x81400024 0x00200000

Table 3-5. Input LRCLK Polarity Configuration (Input Parameter C)

C Value	LRCLK Polarity (Both DAI and CDI Port)	HEX Message
0 (default)	LRCLK=Low indicates Channel 0 (i.e. Left)	Starting from DAI_D0 to DAI_D4: 0x81800020 0xFFEFFFFFFF 0x81800021 0xFFEFFFFFFF 0x81800022 0xFFEFFFFFFF 0x81800023 0xFFEFFFFFFF 0x81800024 0xFFEFFFFFFF
1	LRCLK=Low indicates Channel 1 (i.e. Right)	Starting from DAI_D0 to DAI_D4: 0x81400020 0x00100000 0x81400021 0x00100000 0x81400022 0x00100000 0x81400023 0x00100000 0x81400024 0x00100000

Table 3-6. Input DAI Mode Configuration (Input Parameter D)

D Value	Description	HEX Message
0 (default)	DAI2_LRCLK/SCLK – Slave Compressed Data Input on DAI_D4	0x81000025 0X1008D110
1	DAI1_LRCLK/SCLK - Slave Compressed Data Input on DAI_D0 with only DAI_D4 enabled	0x81000025 0X0000D190
2	DAI2_LRCLK/SCLK - Slave PCM Data Input on DAI_D4 Routed to Center and LFE	0x81000025 0X1558E11F
3	DAI1_LRCLK/SCLK - Slave PCM Data Input on DAI_D0 Routed to Center and LFE	0x81000025 0X1008C10F

Table 3-6. Input DAI Mode Configuration (Input Parameter D) (Continued)

D Value	Description	HEX Message
4	DAI1_LRCLK/SCLK - Slave Compressed Data on DAI_D0 and DAI_D0 to DAI_D4 enabled	0x81000025 0X0000D11F

3.4 Digital Audio Input Port Description

CS4953x4/CS4970x4 is capable of accepting DSD audio data directly. DSD data differs from PCM in that audio is provided as a contiguous stream of 1's and 0's on a single line. There is no framing clock (LRCLK), and there is only one channel per line. The CS4953x4/CS4970x4 supports internal conversion of DSD data to PCM which can then be processed by the DSP.

The CS4953x4/CS4970x4 DSD port is designed to accept DSD audio data from up to 6 pins simultaneously (6 channels total)

DSD features include:

- Six DSD Input Pins
- One Shared DSD_CLK for All Data Pins
- Supports 44.1 KHz and 88.2 KHz Sample Rates

3.4.1 DSD Pin Description

Table 3-7 shows the mnemonic and pin description of the pins associated with the DSD port on CS4953x4/CS4970x4.

Table 3-7. DSDI Audio Input Port

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
DSD_CLK	Bit clock used for latching the DSD audio data. This clock is shared by DSD[5:0].	137	29	Input
DSD0	DSD Audio Input 0	135	27	Input
DSD1	DSD Audio Input 1	134	26	Input
DSD2	DSD Audio Input 2	132	24	Input
DSD3	DSD Audio Input 3	131	23	Input
DSD4	DSD Audio Input 4	138	34	Input
DSD5	DSD Audio Input 5	142	30	Input

3.4.2 Supported DSD Functional Blocks

Figure 3-3 shows the functional block diagram of the features currently supported with the CS4953x4/CS4970x4 DSD Port.

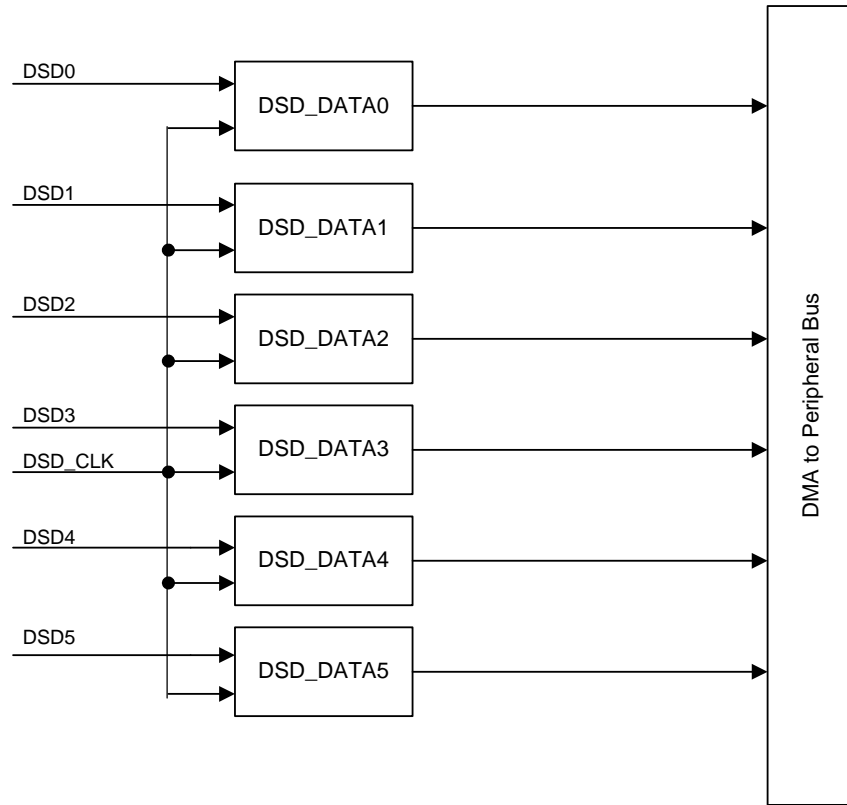


Figure 3-3. DSD Port Block Diagram

§§

Chapter 4

Audio Output Interface

4.1 Introduction

The CS4953x4/CS4970x4 has two output ports - Digital Audio Output port 1 & 2 (DAO1 & DAO2). Each port can output 8 channels of up to 32-bit PCM data. The Digital Audio Output ports are both implemented with a modified 3-wire Inter-IC Sound (I²S) interface along with an oversampling Master clock (MCLK). The I²S interface includes a frame clock at the current sampling frequency (LRCLK), a bit clock for clocking the bits of the audio word (SCLK), and 4 audio data output signals (DATA[3:0]). Each of the output data signals can be connected to the digital stereo input of an audio digital-to-analog converter (DAC) for up to 8 channels of stereo PCM output per DAO port for a total of 16 digital audio output channels.

Each DAO port may Slave to an externally generated SCLK and LRCLK or it may Master these clocks if MCLK is provided. Each port can be configured as having an independent clock domain in Slave mode, or the ratio of the two clocks can be set to even multiples of each other in Master mode. Additionally, the two ports can be ganged together into a single clock domain. The port supports data rates from 32 KHz to 192 KHz. Each port can also be configured to provide a 32-KHz to 192-KHz S/PDIF transmitter (XMTA and XMTB) as an output. [Figure 4-1](#) illustrates the DAO block diagram.

4.2 Digital Audio Output Port Description

4.2.1 DAO Pin Description

[Figure 4-1](#) identifies the pins associated with the Digital Audio Output Ports (DAO1 and DAO2).

Table 4-1. Digital Audio Output (DAO1 & DAO2) Pins

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
DAO1_LRCLK	Sample Rate Clock	22	54	I/O
DAO1_SCLK	Bit Clock	20	52	I/O
DAO1_DATA0	Digital Audio Output	19	51	Output
DAO1_DATA1	Digital Audio Output	17	49	Output
DAO1_DATA2	Digital Audio Output	16	48	Output
DAO1_DATA3/XMTA	Digital Audio Output	15	47	Output
DAO2_LRCLK	Sample Rate Clock	14	46	I/O
DAO2_SCLK	Bit Clock	12	44	I/O
DAO2_DATA0	Digital Audio Output	11	43	Output
DAO2_DATA1	Digital Audio Output	7	39	Output
DAO2_DATA2	Digital Audio Output	6	38	Output
DAO2_DATA3/XMTB	Digital Audio Output	5	35	Output
DAO_MCLK	Master Clock	8	40	I/O

DAO_MCLK is the Master clock and is firmware configurable to be either an input (Slave) or an output (Master). If MCLK is to be used as an output, the internal PLL must be used. As an output MCLK can be configured to provide a 128Fs, 256Fs, or 512Fs clock, where Fs is the output sample rate.

DAO1_SCLK is the bit clock used to clock data out on DAO1_DATA[3:0].

DAO1_LRCLK is the data framing clock whose frequency is equal to the sampling frequency for the DAO1 data outputs.

DAO1_DATA[3:0] are the data outputs and are typically configured for outputting two channels of I²S or left-justified PCM data. DAO1_DATA0 may also be configured to provide output for four or six channels of PCM data. The DAO1_DATA3 (XMTA) pin can alternatively serve as an S/PDIF transmitter output.

DAO2_SCLK is the bit clock used to clock data out on DAO2_DATA[3:0].

DAO2_LRCLK is the data framing clock which has a frequency equal to the sampling frequency for the DAO2 data outputs.

DAO2_DATA[3:0] are the data outputs and are typically configured for outputting two channels of I²S or left-justified PCM data. DAO2_DATA0 may also be configured to provide output for four or six channels of PCM data. The DAO2_DATA3 (XMTB) pin can alternatively serve as an S/PDIF transmitter output..

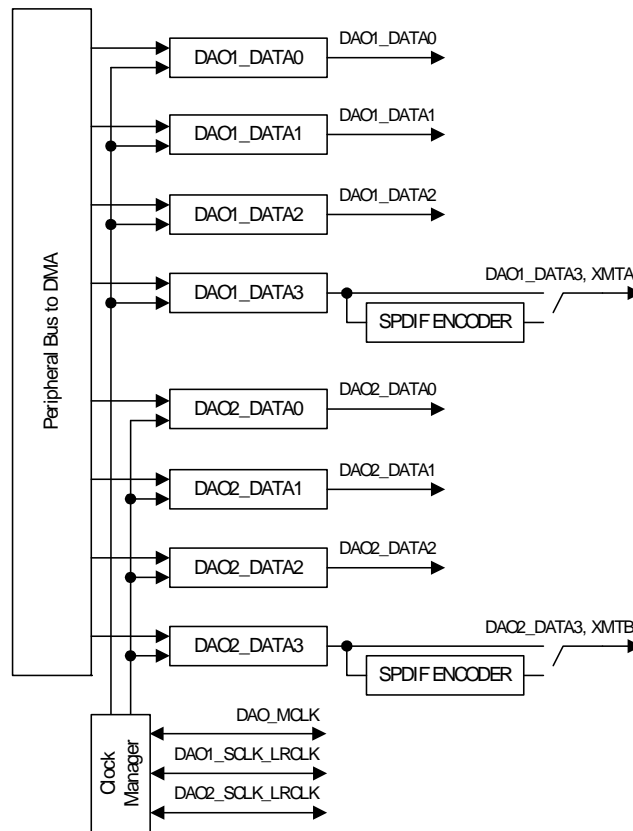


Figure 4-1. DAO Block Diagram

4.2.2 Supported DAO Functional Blocks

As mentioned earlier in the previous section, two DAO ports, DAO1_DATA3 and DAO2_DATA3, are unique in that they are designed to serve as either an output for I²S or left-justified PCM data or as S/PDIF transmitters (XMTA and XMTB). When configured as S/PDIF transmitters, the ports encode digital audio data according to the Sony[®]/Philips[®] Digital Interface Format (S/PDIF), also known as IEC60958, or the AES/EBU interface format. Please see [Figure 4-1](#) for a block diagram of the supported functional blocks for the DAO on the CS4953x4/CS4970x4

4.2.3 DAO Interface Formats

The DAO interface has 8 stereo data outputs that are fully configurable including support for I²S, left-justified, and multichannel (one-line data mode) formats. This section describes some common audio formats that the CS4953x4/CS4970x4 DAO interface supports. It should be noted that the digital output ports provide up to 32-bit PCM resolution.

Note: DAO1_DATAx is valid with respect to DAO1_SCLK/DAO1_LRCLK and DAO2_DATAx is valid with respect to DAO2_SCLK/DAO2_LRCLK.

4.2.3.1 I²S Format

In this format, data is presented most-significant bit (MSB) first, one DAO_SCLK delay after the transition of DAO_LRCLK, and is valid on the rising edge of DAO_SCLK. The left subframe is presented when DAO_LRCLK is low, and the right subframe is presented when DAO_LRCLK is high. [Figure 4-2](#) provides details on I²S compatible (maximum of 32 bits) serial audio formats.

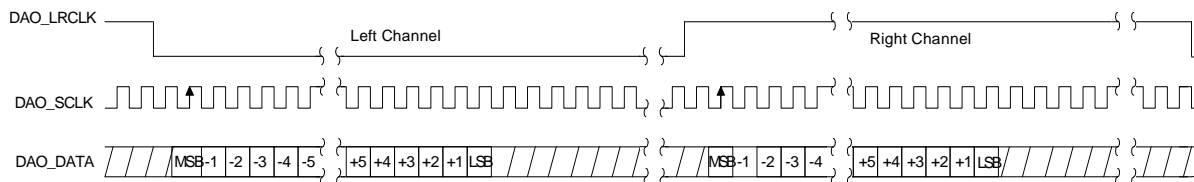


Figure 4-2. I²S Compatible Serial Audio Formats (Rising Edge Valid)

4.2.3.2 Left-Justified Format

[Figure 4-3](#) illustrates the left-justified (LJ) format with a rising edge DAO_SCLK. Data is presented most-significant bit first on the first DAO_SCLK after a DAO_LRCLK transition and is valid on the rising edge of DAO_SCLK. The left subframe is presented when DAO_LRCLK is high and the right subframe is presented when DAO_LRCLK is low. This format can also be programmed for data to be valid on the falling edge of DAO_SCLK.

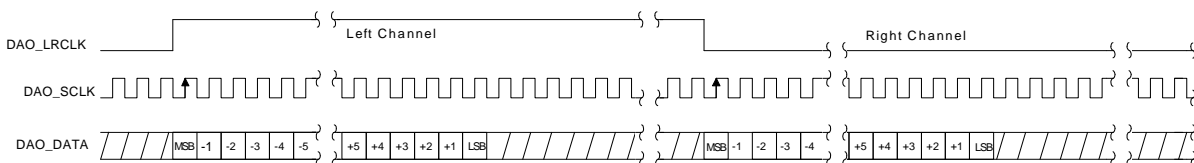


Figure 4-3. Left-justified Digital Audio Formats (Rising Edge Valid DAO_SCLK)

4.2.3.3 One-line Data Mode Format (Multichannel)

The CS4953x4/CS4970x4 is capable of multiplexing all digital audio outputs on one line, as illustrated in [Figure 4-4](#). This mode is available only through special request. Please contact your local Cirrus representative for further details.

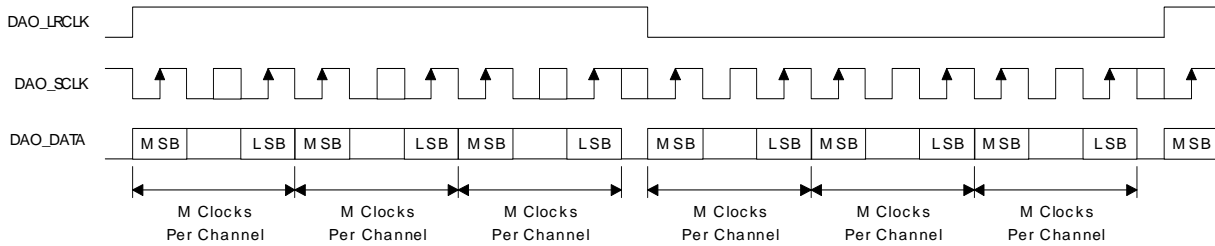


Figure 4-4. One-line Data Mode Digital Audio Formats

4.2.4 DAO Hardware Configuration

DAO hardware configuration must be done at design time using DSP Condenser. See [Chapter 8, "DSP Condenser"](#) for more details.

4.2.5 DAO Hardware Naming Convention

The DAO naming convention is as follows:

OUTPUT A B C D E F

where the parameters are defined as:

- **A** - DAO Clock Mode (Master/Slave for DAO_LRCLK and DAO_SCLK)
- **B** - DAO1/DAO2 Clock Relationship
- **C** - DAO_MCLK, DAO_SCLK, DAO_LRCLK Frequency Ratio
- **D** - Data Format (I2S, Left Justified)
- **E** - DAO_LRCLK Polarity
- **F** - DAO_SCLK Polarity
- **G** - Output Channel Configuration

[Table 4-2](#), [Table 4-3](#), [Table 4-4](#), [Table 4-5](#), [Table 4-6](#), and [Table 4-7](#) show the different values for each parameter as well as the hex message that needs to be sent to configure the port. When creating the hardware configuration message, only one hex message should be sent per parameter.

Note:All DAO hardware config messages must be sent to DSPB. For more details on sending messages to DSP B refer to AN288, section 2.1.4

Table 4-2 shows values and messages for DAO output clock mode configuration parameters.

Table 4-2. Output Clock Mode Configuration (Parameter A)

A Value	DAO 1 & 2 Modes (MCLK, LRCLK and SCLK)	Hex Message
0 (default)	DAO_MCLK - Slave DAO1_LRCLK - Slave DAO1_SCLK - Slave DAO2_LRCLK - Slave DAO2_SCLK - Slave	0x8140002C 0x00002000 0x8100002D 0x00002000
1	DAO_MCLK - Slave DAO1_LRCLK - Master DAO1_SCLK - Master DAO2_LRCLK - Master DAO2_SCLK - Master	0x8180002C 0xFFFFDFFF 0x8180002D 0xFFFFDFFF
2	DAO_MCLK - Slave DAO1_LRCLK - Slave DAO1_SCLK - Slave DAO2_LRCLK - Master DAO2_SCLK - Master	0x8140002C 0x00002000 0x8180002D 0xFFFFDFFF

Please refer to the Table 4-3, Table 4-4, Table 4-5, Table 4-6, and Table 4-7 for a visual example of the clocking directions for the settings in Table 4-2.

Table 4-3 shows values and messages for the data format configuration parameters.

Table 4-3. DAO1 & DAO2 Clocking Relationship Configuration (Parameter B)

B Value	DAO1 & DAO2 Clocking Relationship	Hex Message
0 (default)	DAO2 dependent on DAO1 clocks <i>* DAO2_LRCLK & DAO2_SCLK are driven by DAO1_LRCLK & DAO1_SCLK</i>	0x8140002B 0x00002000
1	DAO2 independent of DAO1 clocks	0x8180002B 0xFFFFDFFF

Table 4-4 shows values and messages for the output DAO_SCLK/LRCLK frequency configuration parameter.

Table 4-4. Output DAO_SCLK/LRCLK Configuration (Parameter C)

C Value	DAO_SCLK Frequency	Hex Message
0 (default)	DAO_MCLK = 256 FS DAO1_SCLK = DAO_MCLK / 4 = 64 FS DAO1_LRCLK = DAO1_SCLK / 64 = FS DAO2_SCLK = DAO_MCLK / 4 = 64 FS DAO2_LRCLK = DAO2_SCLK / 64 = FS	0x8100003D 0x00007711 0x8100003E 0x00007711 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
1	DAO_MCLK = 256 FS DAO1_SCLK = DAO_MCLK / 2 = 128 FS DAO1_LRCLK = DAO1_SCLK / 128 = FS DAO2_SCLK = DAO_MCLK / 2 = 128 FS DAO2_LRCLK = DAO2_SCLK / 128 = FS	0x8100003D 0x00017701 0x8100003E 0x00017701 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000040

Table 4-4. Output DAO_SCLK/LRCLK Configuration (Parameter C) (Continued)

C Value	DAO_SCLK Frequency	Hex Message
2	DAO_MCLK = 256 FS DAO1_SCLK = DAO_MCLK / 1 = 256 FS DAO1_LRCLK = DAO1_SCLK / 256 = FS DAO2_SCLK = DAO_MCLK / 1 = 256 FS DAO2_LRCLK = DAO2_SCLK / 256 = FS	0x8100003D 0x00037700 0x8100003E 0x00037700 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000060
3	DAO_MCLK = 128 FS DAO1_SCLK = DAO_MCLK / 2 = 64 FS DAO1_LRCLK = DAO1_SCLK / 64 = FS DAO2_SCLK = DAO_MCLK / 2 = 64 FS DAO2_LRCLK = DAO2_SCLK / 64 = FS	0x8100003D 0x00007701 0x8100003E 0x00007701 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
4	DAO_MCLK = 128 FS DAO1_SCLK = DAO_MCLK / 1 = 128 FS DAO1_LRCLK = DAO1_SCLK / 128 = FS DAO2_SCLK = DAO_MCLK / 1 = 128 FS DAO2_LRCLK = DAO2_SCLK / 128 = FS	0x8100003D 0x00017700 0x8100003E 0x00017700 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000040
5	DAO_MCLK = 512 FS DAO1_SCLK = DAO_MCLK / 8 = 64 FS DAO1_LRCLK = DAO1_SCLK / 64 = FS DAO2_SCLK = DAO_MCLK / 8 = 64 FS DAO2_LRCLK = DAO2_SCLK / 64 = FS	0x8100003D 0x00007713 0x8100003E 0x00007713 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
6	DAO_MCLK = 512 FS DAO1_SCLK = DAO_MCLK / 4 = 128 FS DAO1_LRCLK = DAO1_SCLK / 128 = FS DAO2_SCLK = DAO_MCLK / 4 = 128 FS DAO2_LRCLK = DAO2_SCLK / 128 = FS	0x8100003D 0x00017711 0x8100003E 0x00017711 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000040
7	DAO_MCLK = 512 FS DAO1_SCLK = DAO_MCLK / 2 = 256 FS DAO1_LRCLK = DAO1_SCLK / 256 = FS DAO2_SCLK = DAO_MCLK / 2 = 256 FS DAO2_LRCLK = DAO2_SCLK / 256 = FS	0x8100003D 0x00037701 0x8100003E 0x00037701 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000060
8	DAO_MCLK = 384 FS DAO1_SCLK = DAO_MCLK / 4 = 96 FS DAO1_LRCLK = DAO1_SCLK / 96 = FS DAO2_SCLK = DAO_MCLK / 4 = 96 FS DAO2_LRCLK = DAO2_SCLK / 96 = FS	0x8100003D 0x00037211 0x8100003E 0x00037211 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000030

Table 4-4. Output DAO_SCLK/LRCLK Configuration (Parameter C) (Continued)

C Value	DAO_SCLK Frequency	Hex Message
9	DAO_MCLK = 384 FS DAO1_SCLK = DAO_MCLK / 2 = 192 FS DAO1_LRCLK = DAO1_SCLK / 192 = FS DAO2_SCLK = DAO_MCLK / 2 = 192 FS DAO2_LRCLK = DAO2_SCLK / 192 = FS	0x8100003D 0x00077201 0x8100003E 0x00077201 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000050
10	DAO_MCLK = 1024 Fs DAO1_SCLK = DAO_MCLK/16 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/16 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007717 0x8100003E 0x0007717 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
11	DAO_MCLK = 64 Fs DAO1_SCLK = DAO_MCLK/1 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/1 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007700 0x8100003E 0x0007700 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
12	DAO_MCLK = 384 Fs DAO1_SCLK = DAO_MCLK/6 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/6 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007712 0x8100003E 0x0007712 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
13	DAO_MCLK = 768 Fs DAO1_SCLK = DAO_MCLK/12 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/12 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007732 0x8100003E 0x0007732 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
14	DAO_MCLK = 1152 Fs DAO1_SCLK = DAO_MCLK/18 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/18 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007752 0x8100003E 0x0007752 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020
15	DAO_MCLK = 192 Fs DAO1_SCLK = DAO_MCLK/3 = 64 Fs DAO1_LRCLK = DAO1_SCLK/64 = Fs DAO2_SCLK = DAO_MCLK/3 = 64 Fs DAO2_LRCLK = DAO2_SCLK/64 = Fs	0x8100003D 0x0007702 0x8100003E 0x0007702 0x8180002C 0xFFFFFFFF8F 0x8140002C 0x00000020

Table 4-5 shows values and messages for the data format configuration parameters.

Table 4-5. Output Data Format Configuration (Parameter D)

D Value	DAO Data Format Of DAO_DATA0, 1, 2 (or DAO_DATA0 for Multichannel Modes)	Hex Message
0 (default)	I ² S 32-bit	0x81000030 0x00000001 0x81000031 0x00000001 0x81000032 0x00000001 0x81000033 0x00000001 0x81000034 0x00000001 0x81000035 0x00000001 0x81000036 0x00000001 0x81000037 0x00000001
1	Left Justified 32-bit	0x81000030 0x00000000 0x81000031 0x00000000 0x81000032 0x00000000 0x81000033 0x00000000 0x81000034 0x00000000 0x81000035 0x00000000 0x81000036 0x00000000 0x81000037 0x00000000 0x8180002C 0xFFFFFBFF 0x8180002D 0xFFFFFBFF
2	TDM on DAO1_D0	0x81000030 0x00011701 0x81000031 0x00000001 0x81000032 0x00000001 0x81000033 0x00000001 0x81000034 0x00000001 0x81000035 0x00000001 0x81000036 0x00000001 0x81000037 0x00000001

Table 4-6 shows values and messages for the DAO_LRCLK polarity configuration parameter.

Table 4-6. Output DAO_LRCLK Polarity Configuration (Parameter E)

E Value	DAO_LRCLK Polarity	Hex Message
0 (default)	LRCLK=Low indicates Left Subchannel	0x8140002C 0x00000700 0x8140002D 0x00000700
1	LRCLK=Low indicates Right Subchannel	0x8180002C 0xFFFFFBFF 0x8140002C 0x00000300 0x8180002D 0xFFFFFBFF 0x8180002D 0x00000300

Table 4-7 shows values and messages for the DAO_SCLK polarity configuration parameter.

Table 4-7. Output DAO_SCLK Polarity Configuration (Parameter F)

F Value	DAO_SCLK Polarity	Hex Message
0 (default)	Data Valid on Rising Edge (clocked out on falling)	0x8180002C 0xFFFFEFFF 0x8180002D 0xFFFFEFFF
1	Data Valid on Falling Edge (clocked out on rising)	0x8140002C 0x00001000 0x8140002D 0x00001000

Table 4-8 shows values and messages for the output channel configuration parameter.

Table 4-8. Output Channel Configuration (Parameter G)

G Value	Channel Configuration	Hex Message
0 (default)	2 Channels	0x8180002C 0xFFFF88FF 0x8140002C 0x00000700
1	6 Channels	0x8180002C 0xFFFF88FF 0x8140002C 0x00000400

4.2.6 S/PDIF Transmitter

Two S/PDIF transmitters are provided on the XMTA and XMTB pins that can output an IEC60958-compliant S/PDIF stream. The modulation clock source for the S/PDIF transmitter is the clock present on the DAO_MCLK pin. The sample rate of the transmitter will be the same setting as for the respective DAO port.

The DSP has an internal multiplexer that can be used to route an external S/PDIF signal from the XMTA_IN or XMTB_IN pin directly to the respective XMTA/XMTB S/PDIF output pin instead of the internally generated S/PDIF signal.

To summarize the XMTA/XMTB S/PDIF output pins can be configured as:

- I²S output - Default
- S/PDIF Transmitter - Sent configuration from [Table 4-10](#)
- DSP Bypass - Send configuration from [Table 4-11](#)

The DSP Bypass configuration is typically used to route a S/PDIF stream from input to the output for recording applications or as a PCM bypass for dual-zone applications. A soft reset is required when switching between any of the above modes.

Table 4-9. S/PDIF Transmitter Pins

Pin Name	Pin Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
XMTA_IN	S/PDIF Input for XMTA mux The XMTA_IN S/PDIF inputs is muxed with XMTA to allow switching the S/PDIF output between the internal and external sources	2	-	Input
XMTB_IN	S/PDIF Input for XMTB mux The XMTB_IN S/PDIF inputs is muxed with XMTB to allow switching the S/PDIF output between the internal and external sources	92	-	Input
DAO1_DATA3/XMTA	S/PDIF Audio Output A	15	47	Output
DAO2_DATA3/XMTB	S/PDIF Audio Output B	5	35	Output

Table 4-10. S/PDIF Transmitter Configuration

Description	Hex Message
Enable S/PDIF on Output A on DAO1_DATA3/XMTA (MCLK=256 Fs)	0x8100002e 0x00005080
Enable S/PDIF on Output A on DAO1_DATA3/XMTA (MCLK=512 Fs)	0x8100002e 0x00005080 0x8180003C 0xFFFFFFFF8F 0x8140003C 0x00000010
Enable S/PDIF on Output A on DAO1_DATA3/XMTA (MCLK=1024 Fs)	0x8100002e 0x00005080 0x8180003C 0xFFFFFFFF8F 0x8140003C 0x00000030
Enable S/PDIF on Output B on DAO2_DATA3/XMTB (MCLK=256 Fs)	0x8100002f 0x00005080
Enable S/PDIF on Output B on DAO2_DATA3/XMTB (MCLK=512 Fs)	0x8100002f 0x00005080 0x8180003C 0xFFFFFFFF8F 0x8140003C 0x00000010
Enable S/PDIF on Output B on DAO2_DATA3/XMTB (MCLK=1024 Fs)	0x8100002f 0x00005080 0x8180003C 0xFFFFFFFF8F 0x8140003C 0x00000030

Table 4-11. DSP Bypass Configuration

DAO_SCLK Polarity	Hex Message
Route XMTA_IN to DAO1_DATA3/XMTA and XMTB_IN to DAO2_DATA3/XMTB	0x81000052 0x0000001B
Route XMTA_IN to DAO1_DATA3/XMTA	0x81000052 0x00000013
Route XMTB_IN to DAO2_DATA3/XMTB	0x81000052 0x0000000b
Disable DSP Bypass	0x81000052 0x00000003

§§

Chapter 5

External Memory Interfaces

5.1 SDRAM Controller

Products in the CS4953x4/CS4970x4 family that use a 144-pin package support a glueless external SDRAM interface to extend the data and/or program memory of the DSP during runtime. The CS4953x4/CS4970x4 SDRAM controller provides two-port access to X, Y, and P memory space, a four-word read buffer, and a double-buffered four-word write buffer. One SDRAM controller port is dedicated to P memory space and the second port is shared by X & Y memories. An arbiter is wrapped around the first port to provide external SDRAM access through both X & Y memory space from 8000h-FFFFh. The second port is connected to provide external SDRAM access through P memory space from 8000h-FFFFh.

The X/Y port has dual write buffers and a single read buffer. The P memory port has a single read buffer. One of these buffers contains four 32-bit words (128 bits). Every miss to the read buffer will cause the SDRAM controller to burst eight 16-bit reads on the SDRAM interface. [Figure 5-1](#) shows a block diagram of the SDRAM external memory control interface for the CS4953x4/CS4970x4 chip.

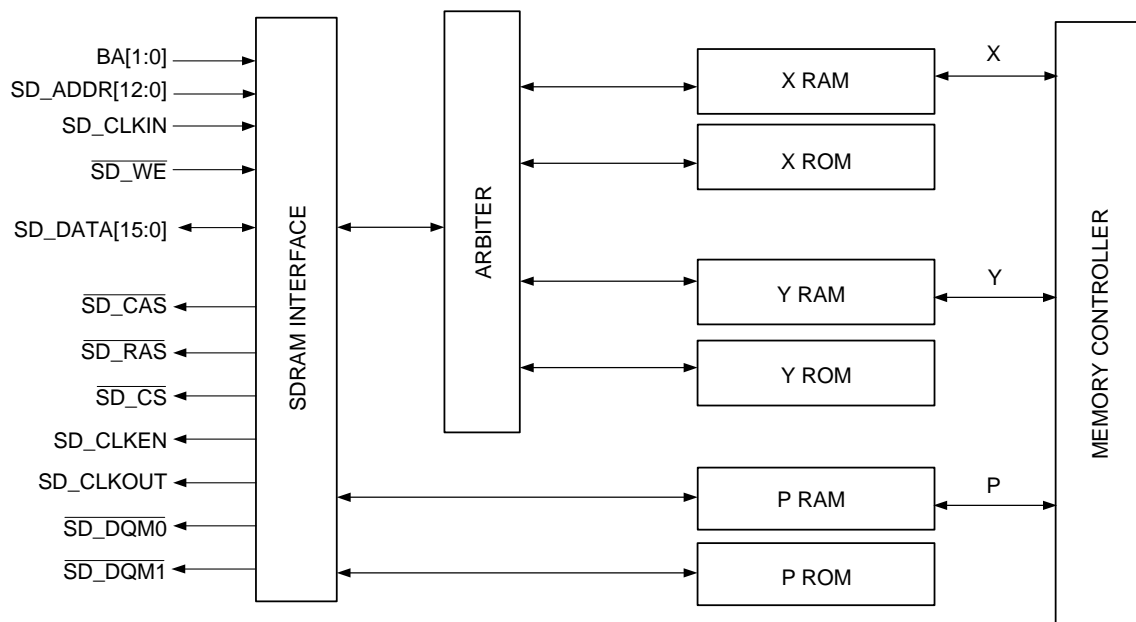


Figure 5-1. SDRAM Interface Block Diagram

5.1.1 SDRAM Controller Interface

The physical interface of the SDRAM controller consists of 16 data pins (SD_DATA[15:0]), 13 address pins (SD_ADDR[12:0]), 2 bank address pins (SD_BA[1:0]), and 9 control pins (SD_CS, SD_WE, SD_DQM1, SD_DQM0, SD_CAS, SD_RAS, SD_CLKOUT, SD_CLKIN, SD_CLKEN). SD_CS is the SDRAM chip select pin. The address and data pins are shared with the Flash interface. The CS4953x4/CS4970x4 supports SDRAMs from 2 Mbytes to 64 Mbytes with various row, bank, and column configurations. The size can be configured in the DynamicConfig0 register listed in Table 5-2. Timing parameters of the SDRAM port can be configured to meet various SDRAM requirements as described in Table 5-2. The default timing parameters have been chosen and tested to meet the requirements of Hynix HY57V641620HG-H. By default, the SDRAM port is configured for 64 Mbits with 4 banks, 12 rows, and 8 columns with a RAS and CAS latency of 3.

Note:When connected to a 16 Mbit SDRAM, the CS4953x4/CS4970x4 uses only SD_BA0 for bank selection.

5.1.2 SDRAM Interface Signals

Table 5-1 shows the signal names, descriptions, and pin number of the signals associated with the external SDRAM memory control port on the CS4953x4/CS4970x4 chip.

Table 5-1. SDRAM Interface Signals

Signal Name	Signal Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SD_CLKOUT	SDRAM clock output. This output is tri-stated when SDRAM interface is not used.	51	80	Output
SD_CLKIN	SDRAM Clock input Connects to trace from SDRAM device CLKIN pin.	52	81	Input
SD_CLKEN	SDRAM Clock Enable Output	53	82	Output
SD_RAS	SDRAM Row Address Strobe	80	109	Output
SD_CAS	SDRAM Column Address Strobe	79	108	Output
SD_CS	SDRAM Chip Select	81	110	Output
SD_DQM0	SDRAM Data Mask 0	28	57	Output
SD_DQM1	SDRAM Data Mask 1	50	79	Output
SD_WE	SDRAM Write Enable	78	107	Output
SD_A0	SDRAM Address 0	72	102	Output
SD_A1	SDRAM Address 1	71	101	Output
SD_A2	SDRAM Address 2	70	99	Output
SD_A3	SDRAM Address 3	68	97	Output
SD_A4	SDRAM Address 4	67	96	Output
SD_A5	SDRAM Address 5	64	93	Output
SD_A6	SDRAM Address 6	62	91	Output
SD_A7	SDRAM Address 7	61	90	Output
SD_A8	SDRAM Address 8	59	88	Output
SD_A9	SDRAM Address 9	58	87	Output

Table 5-1. SDRAM Interface Signals (Continued)

Signal Name	Signal Description	LQFP-144 Pin #	LQFP-128 Pin #	Pin Type
SD_A10	SDRAM Address 10	74	103	Output
SD_A11	SDRAM Address 11	56	85	Output
SD_A12	SDRAM Address 12	55	84	Output
SD_BA0 ¹	SDRAM Bank Address 0/Flash Address 13	75	104	Input
SD_BA1 ¹	SDRAM Bank Address 1/Flash Address 14	77	106	Input
SD_D0	SDRAM Bidirectional Data Bit 0	39	68	BiDir
SD_D1	SDRAM Bidirectional Data Bit 1	37	65	BiDir
SD_D2	SDRAM Bidirectional Data Bit 2	35	64	BiDir
SD_D3	SDRAM Bidirectional Data Bit 3	34	63	BiDir
SD_D4	SDRAM Bidirectional Data Bit 4	32	61	BiDir
SD_D5	SDRAM Bidirectional Data Bit 5	31	60	BiDir
SD_D6	SDRAM Bidirectional Data Bit 6	30	59	BiDir
SD_D7	SDRAM Bidirectional Data Bit 7	29	58	BiDir
SD_D8	SDRAM Bidirectional Data Bit 8	49	78	BiDir
SD_D9	SDRAM Bidirectional Data Bit 9	48	77	BiDir
SD_D10	SDRAM Bidirectional Data Bit 10	46	75	BiDir
SD_D11	SDRAM Bidirectional Data Bit 11	45	74	BiDir
SD_D12	SDRAM Bidirectional Data Bit 12	43	72	BiDir
SD_D13	SDRAM Bidirectional Data Bit 13	42	71	BiDir
SD_D14	SDRAM Bidirectional Data Bit 14	41	70	BiDir
SD_D15	SDRAM Bidirectional Data Bit 15	40	69	BiDir

1. For 16-Mbit parts SD_BA0 is used as a bank select. For 64-Mbit and 128-Mbit parts SD_BA[1:0] is the 2-bit bank select.

5.1.3 Configuring SDRAM Parameters

Not all SDRAM manufacturers conform to the same timing specifications. Therefore, the CS4953x4/CS4970x4 DSP must be configured to match the timing specifications for the SDRAM being used. All SDRAM parameters are set in DSP Condenser at design time. See [Chapter 8, "DSP Condenser"](#) for more details.

Refer to the External Memory Interface section in the CS4953x4/CS4970x4 datasheet for timing parameters that are summarized in [Table 5-2](#).

Table 5-2. SDRAM Interface Parameters

Mnemonic	Hex Message
Extmem_Setup_Control Bit 31:5 = 0 = Reserved Bit 4 = 0/1 = Disable/Enable Flash port Bit 3:1 = 0 = Reserved Bit 0 = 0/1 = Disable/Enable SDRAM port	0x8100005C 0xhhhhhhhh Default: 0x00000011
CRUSConfig Bit 31:9 = 0 = Reserved Bit 8 = Pin Mapping, where: 0 = Enable Flash and SDRAM Mapping 1 = Enable Flash and SRAM Mapping Bit 7:0 = 0 = Reserved	0x8100005D 0xhhhhhhhh Default: 0x00000000
DynamicRefresh Configure the refresh period Bit 31:11 = 0 = Reserved Bit 10:0 = REFRESH, where: 0x0 = refresh disabled. 0x1 = 1x16 = 16 HCLK ticks between refresh cycles 0x8 = 8x16 = 128 HCLK ticks between refresh cycles 0x1 to 0x7FF = REFRESH*16 HCLK ticks between refresh cycles Example: Refresh Period = 15.625 μ S, HCLK = 10 MHz REFRESH = (15.62 μ S*150MHz)/16 = 146.44 = 0x93	0x81000061 0xhhhhhhhh Default 0x00000075
DynamictrP Configure the precharge command period Bit 31:4 = 0 = Reserved Bit 3:0 = Trp, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles. Example: Trp = 20 nS, HCLK = 150 MHz Trp =20 nS*150 MHz - 1 = 3 -1= 0x2	0x81000062 0xhhhhhhhh Default 0x00000002
DynamictrRAS Configure the active to precharge command period Bit 31:4 = 0 = Reserved Bit 3:0 = Tras, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles. Example: Tras = 45 nS, HCLK = 150 MHz Tras =45 nS*150 MHz - 1 = 6.75 -1= 0x6	0x81000063 0xhhhhhhhh Default 0x00000004
DynamictrEX Configure the self refres exit time. Also known as Tsre Bit 31:4 = 0 = Reserved Bit 3:0 = Trex, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles. Example: Trex = 83 nS, HCLK = 150 MHz Trex = 83 nS * 150 MHz - 1 =12.45 -1 = 0xC	0x81000064 0xhhhhhhhh Default 0x00000009

Table 5-2. SDRAM Interface Parameters (Continued)

Mnemonic	Hex Message
DynamicAPR Configure the last data out to active command time. Bit 31:4 = 0 = Reserved Bit 3:0 = Tapr, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles.	0x81000065 0xhhhhhhhh Default 0x00000000
DynamicDAL Configure the data-in to active command time. Bit 31:4 = 0 = Reserved Bit 3:0 = Tdal, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles. Example: Tdal = 6 CLKs (40 nS), HCLK = 150 MHz Tdal = 6 - 1 = 0x5	0x81000066 0xhhhhhhhh Default 0x00000005
DynamicWR Configure the write recovery time. Also known as Tdpl, Trwl Bit 31:4 = 0 = Reserved Bit 3:0 = Tdal, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF = 16 DSP clk cycles. Example: Twr = 2 CLKs (13.3 nS), HCLK = 150 MHz Twr = 2 - 1 = 0x1	0x81000067 0xhhhhhhhh Default 0x00000001
DynamicRC Configure the active to active command time. Bit 31:5 = 0 = Reserved Bit 4:0 = Trc, where: 0x0 to 0x1E = (n + 1) DSP clk cycles. 0x1F = 16 DSP clk cycles. Example: Trc = 65 nS, HCLK = 150 MHz Trc = 65 nS * 150 MHz - 1 = 9.75 - 1 = 0x9	0x81000068 0xhhhhhhhh Default 0x00000007
DynamicRFC Configure the auto refresh period and auto refresh to active command time. Also known as Trrc Bit 31:5 = 0 = Reserved Bit 4:0 = Trc, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0x1F = 32 DSP clk cycles. Example: Trc = 65 nS, HCLK = 150 MHz Trc = 65 nS * 150 MHz - 1 = 9.75 - 1 = 0x9	0x81000069 0xhhhhhhhh Default 0x00000007
DynamicXSR Configure the exit self refresh to active command time. Bit 31:5 = 0 = Reserved Bit 4:0 = Txsr, where: 0x0 to 0x1E = (n + 1) DSP clk cycles. 0x1F = 32 DSP clk cycles. Example: Txsr = 83 nS, HCLK = 150 MHz Txsr = 83 nS * 150 MHz - 1 = 12.45 - 1 = 0xC	0x8100006A 0xhhhhhhhh Default 0x00000009

Table 5-2. SDRAM Interface Parameters (Continued)

Mnemonic	Hex Message
<p>DynamicRRD Configure the active bank A to active bank B latency Bit 31:4 = 0 = Reserved Bit 3:0 = Trrd, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF= 16 DSP clk cycles. Example: Trrd = 15 nS, HCLK = 150 MHz Trrd = 15 nS * 150 MHz - 1 = 2.25 - 1 = 0x2</p>	<p>0x8100006B 0xhhhhhhh Default 0x00000001</p>
<p>DynamicMRD Configure the load mode register to active command time. Also known as Trsa Bit 31:4 = 0 = Reserved Bit 3:0 = Tmrd, where: 0x0 to 0xE = (n + 1) DSP clk cycles. 0xF= 16 DSP clk cycles. Example: Tmrd = 1CLK (6.7 nS), HCLK = 150MHz Tmrd = 1-1 = 0x0</p>	<p>0x8100006C 0xhhhhhhh Default 0x00000000</p>
<p>DynamicConfig0 Configure the active bank A to active bank B latency Bit 31:12 = 0 = Reserved Bit 11:7 = External bus address mapping (Row, Bank, Column),where: 00001 = 16Mb (1Mx16), 2 banks, row length = 11, column length = 8 00101 = 64Mb (4Mx16), 4 banks, row length = 12, column length = 8 01001 = 128Mb (8Mx16), 4 banks, row length = 12, column length = 9 Bit 6:0 = 0 = Reserved</p>	<p>0x8100006E 0xhhhhhhh Default 0x00000280</p>
<p>DynamicRasCas0 Configure the active bank A to active bank B latency Bit 31:10 = 0 = Reserved Bit 9:8 = CAS latency (CAS),where: 00 = reserved 01 = one clock cycle 10 = two clock cycles 11 = three clock cycles Bit 7:2 = 0 = Reserved, rCAS latency (CAS),where: Bit 1:0 = RAS latency (RAS),where: 00 = reserved 01 = one clock cycle 10 = two clock cycles 11 = three clock cycles</p>	<p>0x8100006F 0xhhhhhhh Default 0x00000303</p>

5.2 SPI Flash Interface

The CS4953x4/CS4970x4 provides a glueless external Serial Peripheral Interface (SPI) Flash interface that supports connection to an external SPI Flash or EEPROM for code storage. This allows for products to be field-upgraded as new audio algorithms are developed. The SPI Flash interface allows autobooting from a SPI Flash device. Coefficients for filters may also be stored and recalled from SPI Flash.

§§

Chapter 6

System Design Requirements for SPDIF and HDMI™ Technology Interfaces

6.1 Introduction

This chapter describes system design requirements when designing SPDIF and HDMI™ technology interfaces to the CS4953x4/CS4970x4 DSP.

6.1.1 Designing a SPDIF Input Interface

The default input source state of the CS4953x4/CS4970x4 DSP is SPDIF with Autodetect. This setting is configurable in the flash_image.xml file should the user wish to use a different input source type.

6.1.1.1 SPDIF Clocking

Here are some clocking practices to follow when implementing SPDIF input to the CS4953x4/CS4970x4 DS:

- The SPDIF input with respect to the CS4953x4/CS4970x4 DSP is DAI_DATA4 / DAI2_DATA0).
- Glitch free clock transitions are required from the SPDIF receiver on stream changes (For example, when an input stream changes from 48KHz Dolby Digital to 96 KHz PCM and vice versa).
- The recommended MCLK to LRCLK (Fs) ratio is 256*Fs.
- Any noise on the clocks or incorrect value of the clocks could affect the audio output, even if the SPDIF input is not currently in use by the CS4953x4/CS4970x4 DSP.
- Ensure that DAI2 port clocks are active irrespective of the system input source.
- If the HDMI Receiver SPDIF out is used as an input to the CS4953x4/CS4970x4 DSP, ensure that the HDMI firmware (or additional hardware if required) provides the correct and stable clocks and data is sent out to the CS4953x4/CS4970x4 DSP.

6.1.2 Designing an HDMI Input Interface

6.1.2.1 HDMI Clocking

Here are some clocking practices to follow when implementing HDMI input to the CS4953x4/CS4970x4 DS:

- Set CS4953x4/CS4970x4 MCLK as a slave and mastered by the HDMI Rx. MCLK must stay at 24 MHz for all stream types (Legacy and HD).
- In the event of an input stream change affecting the clocks, Cirrus Logic recommends holding the CS4953x4/CS4970x4 DSP in Reset until the clocks are stable. For example, the Host Controller can probe the HDMI Rx to determine when the clocks are stable.

6.1.2.2 Decoding Stream Types Over HDMI

When decoding audio streams over HDMI, follow these practices:

- In case of Multichannel PCM input over HDMI, specific AC3 decoder modes are used, that is, those modes specifically dedicated to Multichannel PCM inputs over HDMI. Refer to the sample `flash_image.xml` file, which is automatically installed when `dspcondenser.exe` is installed. Do not change the decoder mode when PCM over HDMI is detected.
- Before determining whether the audio output from the CS4953x4/CS4970x4 DSP has stopped, verify the status of audio output of the HDMI Rx fed into the CS4953x4/CS4970x4 DSP. HDMI Rx is known to behave differently for different input stream types, especially the HD streams.

For example:

In case of SI9135 chips, ensure that the audio FIFO and control registers are reset correctly.

- Ensure that the Extended Display Identification Data (EDID) configurations are available for all the stream types to be supported on the system.
- In case of PCM over HDMI, Cirrus Logic recommends that the Host Controller obtain the information to distinguish streams between stereo PCM and Multi-channel PCM from the HDMI Rx. The following stream information can be provided by the CS4953x4/CS4970x4 DSP:
 - Fs detection
 - Even though the DSP can obtain the number of channels when decoding multi-channel PCM, Cirrus Logic recommends that number of channels be derived from the Audio Information frame registers of the HDMI receiver.

6.1.3 Other System Design Considerations

- SPDIF output from the CS4953x4/CS4970x4 DSP is not available for 192 kHz streams.
- Ensure that the power rail (1.8V and 3.3V) ramp-up process is clean and glitch-free.
- Cirrus Logic recommends that only supported Flash devices listed in [Section 1.1.1](#) be used in the design.

§§

Chapter 7

Overview of Common Firmware Modules

7.1 Introduction

The purpose of this chapter is to serve as an introduction to firmware operation on the CS4953x4/CS4970x4 DSP. This chapter explains the general framework of operation on the CS4953x4/CS4970x4.

- New boot code now boots off the customer's SPI Flash device and allows the CS4953x4/CS4970x4 DSP chips to manage all boot tasks. See [Chapter 1, "Operational Modes"](#) for information about the boot process for the CS4953x4/CS4970x4 DSP chips.
- CS4953x4/CS4970x4 DSP chips carry out duties previously performed by the system's microcontroller:
 - DSP auto-switches overlays
 - DSP detects sample rate

Note: Systems using the Cirrus CS4953x4/CS4970x4 DSP chips are based on the assumption that dedicated Flash memory is available to the DSP. DSP processing activities such as auto-switching are more affected by the speed and memory size of the customer's Flash device than by the capability of the system microcontroller, and should be faster than previous system designs.

7.2 CS4953x4/CS4970x4 Firmware

7.2.1 Firmware Modules

The CS4953x4/CS4970x4 DSP provides both standard and advanced audio decoding and processing using factory-supplied firmware modules that are downloaded to the device and executed on one of the DSP cores. Each module provides a specific processing functionality. For instance, the DTS module provides DTS® decoding, the Dolby Headphone® module provides Dolby Headphone processing capability and the Bass Management module provides Bass Management processing.

7.2.2 Overlay Architecture

For memory management reasons, modules are grouped into overlays, which are separately loadable. Most overlays have a one-to-one correspondence with modules. Post processing overlays (PPMs) generally have a one-to-many correspondence: one overlay contains many modules.

Overlays are divided into classes, based on the type of audio processing the modules in that overlay class do. In the CS49700, the following overlay classes are defined:

- OS - operating system functions
- DEC - decoder modules (for example, AC3, DTDHDHRA)
- MPM - matrix processing modules (for example, PL2, NEO6, COMS2)
- VPM - virtualizer modules (for example, Dolby Headphone)
- PPM - post processor modules

At any given moment, only one overlay of each class can be loaded in the DSP (the OS actually is loaded into both DSP cores, DSP A and DSP B).

The overlay architecture thus imposes limits on the types of concurrent firmware features (or modules) are available. The DSP Condenser tool set helps system designers understand and work within the concurrency limitations imposed by the firmware architecture.

7.3 Firmware Messaging

While using the CS4953x4/CS4970x4 it is necessary to communicate with the DSP in order to control or monitor the various downloaded firmware modules. We refer to this process of communication *firmware messaging*. The purpose of this section is to cover the types and formats of these firmware messages. In general, with firmware messaging, the user can control the firmware module running on the DSP and subsequently perform various tasks such as:

- Configure the module after firmware download (e.g. enabling DTS decode, Kick-starting the DSP, etc.)
- Change run-time parameters (e.g. adjusting the Volume, switching Pro Logic II modes, etc.)
- Obtain information from the DSP (e.g. current state of the firmware registers, data stream information, etc.)

7.3.1 Communication Overview

From a “micro-programmer” point of view, the CS4953x4/CS4970x4 firmware modules can be thought of as a blocks of several 32-bit registers (variables) that either control the behavior of the firmware or store information about the state of the firmware at the time of operation. Each register has a unique *index*. Access to the register involves a combination of a specified *opcode* for that firmware module together with the register index. For each firmware module, the following Operation codes (Opcodes) are available:

- Write Opcode - Issues a command to write to a specific module.
- Read Opcode - Issues a command to read from a specific module.
- Read Response Opcode - Indicates module and index that has been read.

These available opcodes permit the following types of communication with the DSP:

- Writing to the DSP
- Solicited read from the DSP
- Unsolicited read from the DSP

The following sub-sections cover the communication types listed above.

7.3.2 Writing to the DSP

A write session consists of one 8-byte message from the host to the DSP. In essence, the write message consists of a 32-bit *command word* followed by a 32-bit *data word*. The command word is formed by combining the write opcode for that module with the index of the register that needs to be written. The 32-bit data word is the value of the data written to that register.

The following diagrams shows the format of a write message:

Write Command Word:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Module ID [6:0]								R/W		0		WordCount-1 [4:0]			
Index [15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Write Data Word:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								Data [31:0]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Options for Bits 23:22

23	22	Command
0	0	Write - current value overwritten
0	1	Write - new value OR'd with current value
1	0	Write - new value AND'd with current value

7.3.3 Solicited Read

A solicited read can be thought of as a request to read the contents of a specific register. This is comprised of a 32-bit solicited read command word which is a request to read a specific index (register) in a given module. The DSP, upon receiving this message, responds by sending back a 32-bit *Read Response* opcode and the requested 32-bit data word contained in the register.

The following diagrams show the format of solicited Read-command word and Read-Response Command Word message:

Read-Command Word:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
1	Module ID [6:0]						1	1	0	WordCount-1 [4:0]						
Index [15:0]																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Read-Response Command Word:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0	Module ID [6:0]						1	1	0	WordCount-1 [4:0]						
Index [15:0]																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Read-Response Data Word:

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0					
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	DATA WORD [31:0]														

7.3.4 Unsolicited Read

There are times when the DSP needs to inform the host of certain changes during its operation. For example, the DSP sending a message to the host indicating that the input stream has changed from Dolby Digital to DTS. This information is useful to the host so it can take appropriate steps such as downloading the correct decoder firmware module. Note that this type of message is an unsolicited message because it was initiated by the CS4953x4/CS4970x4 rather than the host.

These messages will come from the CS4953x4/CS4970x4 to indicate a change in the system that must be addressed. In the example used in the previous paragraph, the part is in autodetect mode and detects a new stream. An unsolicited read message will be sent by the CS4953x4/CS4970x4 to indicate the new stream type.

The 8-byte unsolicited read messages from the CS4953x4/CS4970x4 consist of a 4-byte read command word which defines the type of unsolicited message and an associated 4-byte data word that contains more information describing a system condition. The host senses that an unsolicited message is ready to be read because the IRQ pin for the port being used goes low (SCP1_IRQ or PCP_IRQ). Every time the existence of a message is detected, the host should read out the 8-byte unsolicited read message.

Unsolicited Read Command Word:

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
OPCODE [31:16]																INDEX [15:0]																

Unsolicited Read Data Word:

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
DATA WORD [31:0]																																

7.3.5 Index Configuration

Indices of the Firmware module can differ in properties that are important to the system firmware programmer.

Indices **NOT** marked by a '+' must have their values configured before runtime, using the DSP Condenser tool set. These register values will be loaded by the DSP auto-switching code. Indices marked by '+' may have their values changed at runtime using host write commands.

7.3.6 Unsolicited Messages from DSP to the Host Microcontroller

There are times when the DSP needs to inform the host of certain changes during its operation. For example, the DSP sending a message to the host indicating that the input stream has changed from Dolby Digital to DTS. This information is useful to the host so it can update the Front panel of the Unit. This type of message is an unsolicited message because it was initiated by the DSP rather than the host.

Changes that are reported to the host are:

- Changes in the input stream type reported by DSP_AUTODETECT_MSG
- Changes in the incoming bitstream reported by DSP_LAST_ACCN_MSG

7.3.7 DSP_AUTODETECT_MSG¹

The DSP_AUTODETECT_MSG is used to enable the DSP to notify the system host of changes in the state of the DSP during run-time. A list of autodetect messages is shown in [Table 7-1](#).

No Write Command.

No Read Request Command.

Unsolicited Read Response = 0x8100HHHH 0xhhhhhhh

0xHHHH = index, 0xhhhhhhh = data value

1. If auto-detect and other unsolicited messages are enabled, ensure that the system reads ALL the unsolicited messages from the DSP before sending a write command to the DSP. This practice applies to the “boot-up” state as well as mode changes applied in the DSP Manager. The auto-detect message identifies the stream type and the single or multi-word ACCN message. Failure to read all unsolicited messages may result in no audio output from the DSP.

Table 7-1. DSP_AUTODETECT_MSG Messages

Bit Number	Description
Bit 31	Decodable_Stream_Flag = 0/1 = This stream is not/is decode-able by the application.
Bits [15:0]	Bit 5 Non_IEC61937_Stream_Flag= 1/0 = This stream is not/is IEC61937 compressed data. If Non_IEC61937_Stream_Flag=1 Non_IEC61937 Stream Descriptor. 0x0020 = Silent Input Data (Out of Application Sync) 0x0021 = DTS Format-16 elementary stream 0x0022 = DTS Format-14 elementary stream 0x0023 = Linear 2.1 channel PCM stream 0x0623 = Linear 5.1 channel PCM stream 0x0823 = Linear 7.1 channel PCM stream 0x0024 = HDCD PCM Sync Detect (only available in HDCD application). 0x0025 = HDCD PCM Sync Lost (only available in HDCD application). If Non_IEC61937_Stream_Flag=0 IEC61937 Stream Descriptor: Data type descriptor in IEC61937 specification. Description of the Data-type field of Pc reproduced below from IEC61937 Specification 0x0000 = Never Reported. 0x0001 = AC-3 data. 0x0003 = Never Reported. 0x0004 = MPEG-1 Layer 1 data. 0x0005 = MPEG-1 Layer 2 or 3 data or MPEG-2 without extension. 0x0006 = MPEG-2 data with extension. 0x0007 = MPEG-2 AAC ADTS data. 0x0008 = MPEG-2 Layer 1 Low sampling frequency. 0x0009 = MPEG-2 Layer 2 or 3 Low sampling frequency. 0x000B = DTS-1 data (512-sample bursts). 0x000C = DTS-2 data (1024-sample bursts). 0x000D = DTS-3 data (2048-sample bursts). 0x000E–0x0010 = Reserved. 0x001C = MPEG-2 AAC ADTS data 0x0011 and 0x0211 = DTS-HD High-Resolution Audio (HRA) 0x0015 = Dolby Digital Plus 0x0016 = TrueHD 0x0311 = DTS-HD Low Bit Rate Audio 0x0411 and 0x0C11 = DTS-HD Master Audio, 48 KHz Fs 0x4211 = DTS-HD High Resolution Audio (HRA), 96 KHz Fs 0x4411 and 0x4C11 = DTS-HD Master Audio, 96 KHz Fs 0xC411 and 0xCC11 = DTS HD Master Audio, 192 kHz Fs multi-channel (non-stereo), output at 96 KHz per DTS specification.

7.3.8 DSP_LAST_ACCN_MSG

The DSP_LAST_ACCN_MSG will be updated every time a stream variable changes. The meaning of this register will depend on the value of the DSP_AUTODETECT_MSG. The valid DSP_LAST_ACCN_MSG are listed in [Table 7-2](#).

Table 7-2. DSP_LAST_ACCN_MSG Messages

Decoder Type	Description
PCM	Bits 31:0 = Reserved
HDCD	Bits 31:0 = Reserved
SGEN	Bits 31:0 = Reserved
LOGIC7	Bits 31:0 = Reserved

Table 7-2. DSP_LAST_ACCN_MSG Messages

Decoder Type	Description
AC3	Bits 31:30 = DSUREXMOD variable value Bits 29:28 = DHEADPHONMOD variable value Bit 27 = ADCONVTYP variable value Bit 24 = LFEON variable value Bits 20:16 = DIALNORM variable value Bits 15:13 = BSMOD variable value Bits 12:8 = BSID variable value Bits 5:4 = DSURMOD variable value Bits 2:0 = ACMOD variable value
DTS	Bits 31:28 = Reserved Bits 27:24 = LFF variable value Bits 23:20 = PCMR variable value Bits 15:12 = EXT_Audio ID Bits 11:8 = EXT_Audio Bits 7:0 = AMODE variable value
AAC	Bits 31:25 = Reserved Bits 24 = LFE Element Bits 23:4 = Reserved Bits 3:0 = Channel Configuration
DDPLUS	See the AN304DA application note, "Dolby® Digital Plus Decoder Module."
DTS-HD-HRA	See the AN304DB application note, "DTS-HD™ High Resolution Audio Decoder Module."
DTS-HD-MA	See the AN304DD application note, "DTS-HD™ Master Audio Decoder Module."
DTS-HD-LBR	See the AN304DE application note, "DTS Express™ / DTS-HD™ Low Bit Rate Secondary Audio Decoder Module"
TRUEHD	See the AN304DC application note, "Dolby® TrueHD Decoder Module."

7.4 CS4953x4/CS4970x4 DSP Manager API Description

The DSP Manager module is the firmware module that is responsible for much of the host communication, performing auto-switching, flash updates, and responding to host-initiated configuration changes. The DSP Manager API allows the host to:

- Directly control the current firmware configuration,
- to find out what the current configuration is
- Control what hardware is used for the audio source
- Control common runtime parameters such as gain, trim, and bass management speaker configuration
- Initiate field flash updates from audio stream
- Read/write an area of flash reserved for host use

7.4.1 Microcontroller Interface (API)

Write Command = 0xEF0nHHHH 0xhhhhhhh

OR Command = 0xEF40HHHH 0xhhhhhhh

AND Command = 0xEF80HHHH 0xhhhhhhh
 Read Request Command = 0xEFCnHHHH
 Read Response Message = 0x6FC0HHHH 0xhhhhhhh
 n = number of words – 1 (bits [20..16])
 0xHHHH = Index
 0xhhhhhhh = Data Value

Table 7-3. Microcontroller Interface API

Index	Variable	Description
0x0000	DSP_BOOT	<p>Bit 9 = Reserved</p> <p>Whenever a non-zero value is written to this register, the DSP manager firmware begins the specified configuration change. When the change is complete, this register value is reset to zero. The DSP manager may also (optionally) send an unsolicited message to the microcontroller upon completion of the operation.</p> <p>Bit 8 = configuration lock bit. If this bit is set, the DSP firmware assumes that the DSP_CFG_* variables are being modified by the microcontroller</p> <p>Bits 7: 0 = boot action</p> <p>0x00 = system is not in any of the following states. 0x01 = change to new configuration, as defined in DSP_CFG_xxx variables below 0x02 = save state in preparation for power down 0x08 = shut down 0x04 = save calibration parameters to DSP state flash area</p>

7.4.2 DSP_CFG_XXX Registers

The DSP_CFG_XXX registers in [Table 7-4](#) are used to specify the desired firmware configuration. The writeable parameters are deliberately defined in contiguous locations so that they can be written from the microcontroller to the DSP as efficiently as possible.

Table 7-4. DSP_CFG_XXX Firmware Configuration Registers

Index	Variable	Description
0x0001	DSP_CFG_INPUT_FS	<p>Input source sample rate, determined by DSP</p> <p>Note: In case of PCM input, DSP_CFG_INPUT_FS is sent out as an unsolicited message along with the PCM autodetect message (format = EF000001 0000000X = Input sample rate. Refer to Sample Rate table in the DSP_CFG_OUTPUT_FS variable.</p> <p>This message can be disabled using bit 2 in DSP_MSG_MASK in Table 7-5.</p>
0x0002	DSP_CFG_STREAM_TYPE	<p>Input source stream type, determined by DSP</p> <p>Note: Refer to Table 7-1 for stream types</p>
0x0003	DSP_CFG_INPUT_CHANNELS	<p>Read Only</p> <p>Value in this index determines the channels present in the input PCM stream over HDMI.</p> <p>Bits[7:0]=LFE ,Rb, Lb , Rs , Ls, R , C, L. Other bits RSVD. Not valid for SPDIF input source.</p> <p>Default:0xfffffff</p>
	DSP_CFG_STREAM_FS	<p>Input stream sample rate, as determined by DSP</p>

Table 7-4. DSP_CFG_XXX Firmware Configuration Registers (Continued)

Index	Variable	Description
0x0005	DSP_CFG_OUTPUT_FS	Output sample rate Bits 3:0 Sample Rate 0x0 = 48 KHz 0x1 = 44.1 KHz 0x2 = 32 KHz 0x3 = reserved 0x4 = 96 KHz 0x5 = 88.2 KHz 0x6 = 64 KHz 0x7 = reserved 0x8 = 24 KHz 0x9 = 22.05 KHz 0xA = 16 KHz 0xB = reserved 0xC = 192 KHz 0xD = 176.4 KHz 0xE = 128 KHz 0xF = reserved Default * = 0x00000000 (48KHz)†
0x0006	DSP_CFG_AUDIO_SRC	Input audio Source: 0x01 = SPDIF with autoswitch 0x02 = HDMI with autoswitch 0x03 = Multichannel analog Note: There will be no autodetect message when 0x03 is chosen as the input source. 0x04 = DSD. 0x05 = SPDIF no autoswitch 0x06 = HDMI no autoswitch 0x07 = Flash update mode
0x0007	DSP_CFG_DECODER	Bit 31:16 = DECODER MODE 0x00 = MODE0. 0x01 = MODE1 0x02 = MODE2 0x03 = MODE3 0x04 = MODE4 Bit 15:0 = DECODER TYPE 0x00 = none Other values determined by enumeration type created by flash image tool (Flasher).
0x0008	DSP_CFG_MATRIX	Bit 31:16 = MATRIX_DECODER MODE 0x00 = MODE0. 0x01 = MODE1 0x02 = MODE2 0x03 = MODE3 0x04 = MODE4 Bit 15:0 = MATRIX_DECODER ID 0x00 = none Other values determined by enumeration type created by flash image tool (Flasher).

Table 7-4. DSP_CFG_XXX Firmware Configuration Registers (Continued)

Index	Variable	Description
0x0009	DSP_CFG_VIRTUALIZER	<p>Bit 31:16 = VIRTUALIZER MODE 0x00 = MODE0. 0x01 = MODE1 0x02 = MODE2 0x03 = MODE3 0x04 = MODE4</p> <p>Bit 15:0 = VIRTUALIZER ID 0x00 = none Other values determined by enumeration type created by flash image tool (Flasher).</p>
0x0000A	DSP_CFG_PPM	<p>Bit 15:0 = PPM ID 0x00 = none Other values determined by enumeration type created by flash image tool (Flasher).</p>
0x0000B	DSP_CFG_PPM_MODE1	<p>Bit 31:16 = Module 2 mode Bit 15:0 = Module 1 mode Note: The module numbers are defined in DSP Condenser at Design time in the flash_image.xml file. Module IDs and Module Numbers for custom PPMs can be defined at Design time as well. See examples of how to change modes in Section 7.4.2.1..</p>
0x0000C	DSP_CFG_PPM_MODE2	<p>Bit 31:16 = Module 4 mode Bit 15:0 = Module 3 mode The module numbers are defined in DSP Condenser at Design time in the flash_image.xml file. Module IDs and Module Numbers for custom PPMs can be defined at Design time as well. See examples of how to change modes in Section 7.4.2.1..</p>
0x0000D	DSP_CFG_PPM_MODE3	<p>Bit 31:16 = Module 6 mode Bit 15:0 = Module 5 mode The module numbers are defined in DSP Condenser at Design time in the flash_image.xml file. Module IDs and Module Numbers for custom PPMs can be defined at Design time as well. See examples of how to change modes in Section 7.4.2.1..</p>
0x0000E	DSP_CFG_PPM_MODE4	<p>Bit 31:16 = Module 8 mode Bit 15:0 = Module 7 mode The module numbers are defined in DSP Condenser at Design time in the flash_image.xml file. Module IDs and Module Numbers for custom PPMs can be defined at Design time as well. See examples of how to change modes in Section 7.4.2.1..</p>
0x0000F	DSP_CFG_PPM_MODE5	<p>Bit 31:16 = Module 10 mode Bit 15:0 = Module 9 mode The module numbers are defined in DSP Condenser at Design time in the flash_image.xml file. Module IDs and Module Numbers for custom PPMs can be defined at Design time as well. See examples of how to change modes in Section 7.4.2.1..</p>

Table 7-4. DSP_CFG_xxx Firmware Configuration Registers (Continued)

Index	Variable	Description
0x00010	DSP_CFG_MAIN_SPEAKERS	<p>Bit 25:24 = SURROUND_RIGHT_BACK_SPEAKER_SIZE 00 - None 01 - Large 10 - Small</p> <p>Bit 21:20 = SURROUND_LEFT_BACK_SPEAKER_SIZE 00 - None 01 - Large 10 - Small</p> <p>Bit 17:16 = SURROUND_RIGHT_SPEAKER_SIZE 00 - None 01 - Large 10 - Small</p> <p>Bit 13:12 = SURROUND_LEFT_SPEAKER_SIZE 00 - None 01 - Large 10 - Small</p> <p>Bit 9:8 = CENTER_SPEAKER_SIZE 00 - None 01 - Large 10 - Small</p> <p>Bit 5:4 = FRONT_SPEAKER_SIZE 00 - Not used 01 - Large 10 - Small</p> <p>Bit 0 =1 Subwoofer present, 0= no subwoofer</p>
0x00011	Functionality Still Under Development	Functionality Still Under Development
0x00012	Functionality Still Under Development	Functionality Still Under Development
0x00013	DSP_CFG_MCLK_FACTOR	<p>Set MCLK Frequency to preferred value. 0x80 = 128 Fs 0x100 = 256 Fs 0x200 = 512 Fs Use 0x200 value for HDMI source. Note: See Section 7.4.2.2 for examples how to set the DSP_CFG_MCLK_FACTOR variable.</p>
0x00014	Functionality Still Under Development	Functionality Still Under Development
0x00015	Functionality Still Under Development	Functionality Still Under Development
0x00016	Functionality Still Under Development	Functionality Still Under Development
0x00017	Functionality Still Under Development	Functionality Still Under Development
0x00018	Functionality Still Under Development	Functionality Still Under Development
0x00019	Reserved	Reserved

Note: For any questions relating to the indices in [Table 7-4](#), contact your Cirrus Logic FAE or representative.

7.4.2.1 Using DSP Condenser to Change/Load Firmware Modules

The host processor can issue commands to change/load firmware modules based on following formats:

p=decoder mode, qq= decoder uld ID

r= mpm mode, ss= mpm uld ID

t= vpm mode, uu = vpm uld ID

vv= ppm uld ID

x= ppm mode defined for the first module ID (Module 1 as defined in DSP_CFG_PPM_MODE1 in [Table 7-4](#)) in "ppm modes" section in the flash_image.xml file.

w= ppm mode defined for the second module ID (Module 2 as defined in DSP_CFG_PPM_MODE1 in [Table 7-4](#)) in "ppm modes" section in the flash_image.xml file.

See [Example 1](#) for sample commands to change decoder module mode changes

Example 1 Sample Commands to Initiate Decoder Mode Changes

```
ucmd Ef00000000000100# Set Configuration Lock
ucmd Ef000007000p00qq# Change decoder mode
ucmd Ef000008000r00ss# Change MPM mode
ucmd Ef000009000t00uu# Change VPM mode
ucmd Ef00000A000000vv# Load PPM
ucmd Ef00000B000w000x# Change PPM mode
ucmd Ef00000000000001# Change to new configuration
```

7.4.2.2 Using DSP Condenser to Change the Audio Input Source

The user can issue commands to change the input source based on following formats:

w= input source (Refer to DSP_CFG_AUDIO_SRC in [Table 7-4](#)).

For example: w = 2 for HDMI with autodetect

w = 3 MULTICHANNEL ANALOG INPUT

xxxx = DSP_CFG_MCLK_FACTOR in [Table 7-4](#)

For example:

xxxx = 0200 for HDMI since MCLK is 24 MHz (512 Fs when Fs = 48 KHz)

Or,

xxxx = 0100 for SPDIF Inputs since MCLK is 12.288 MHz (256 Fs when Fs = 48 KHz)

Or,

xxxx= 0200(MCLK = 24MHz) based on the MCLK value for a multi-channel analog input(512 Fs when Fs = 48 KHz)

The DSP_CFG_MCLK_FACTOR should remain the same for a particular input source, although the frequency of the MCLK changes. The sample rate configurations in the flash_image.xml file will set the correct DAO clock values.

For example, If the input stream over HDMI is 96 KHz PCM, the MCLK frequency should remain at 24 MHz and the DSP_CFG_MCLK_FACTOR value (index 0x0013) should remain 0x0200 (512 Fs). DSP_CFG_MCLK_FACTOR has a direct relation with the frequency of MCLK rather than being a factor of the sampling frequency, when input the source is HDMI.

[Example 2](#) provides sample commands for changing the audio input source.

Example 2 Sample Commands to Change Audio Input Source

```
# All of the following commands are mandatory when changing Audio
# Input Sources.
ucmd ef00000000000100# Set Configuration Lock Mode
ucmd ef0000060000000w # Set Audio Input Source
```



```
ucmd ef000007000m00dd # Set decoder (Required only when
                        # switching Multichannel Analog Input
                        # source.)
ucmd ef0000130000xxxx # Set DSP_CFG_MCLK_FACTOR
ucmd Ef00000000000001# Change to new configuration
```

Where dd= decoder uld id capable of PCM decoding, m= decoder mode of the uld specified by 'dd'. Set m=0 and dd=00 to use the PCM decoder in OS.

You can combine input source change commands along with mode change/load command for any applicable firmware module.

Example 3 Sample Commands to Change Audio Input Source and Change/Load Module

```
ucmd ef00000000000100 # Set Configuration Lock Mode
ucmd ef0000060000000w # Set Audio Input Source
ucmd ef0000130000xxxx # Set DSP_CFG_MCLK_FACTOR
ucmd ef000008000y00zz # Set MPM mode
ucmd Ef00000000000001# Change to new configuration
```

Where the audio input source is changed to “w” and simultaneously an MPM module is specified with uld ID “zz” in mode “y” (uld ID and the mode is defined per the flash.h file created when the flash image was created).

See [Appendix C](#) for specific instructions on how to load and unload other decoder firmware modules.

Note: Current firmware does not support changing input source from HDMI/Multichannel source to SPDIF source using DSP Manager API commands. Cirrus Logic recommends using SPDIF as the default input source in the flash_image.xml file and toggling the DSP Reset to switch to SPDIF source from HDMI/Multichannel source.

7.4.3 Status Registers

The following registers in [Table 7-5](#) describe various status return values for the firmware.

Table 7-5. Firmware Status Registers

Index	Variable	Description
0x00020	DSP_STATUS	Read Only Error code if system is halted for some reason 0x00000000 Normal Operation 0x00000001 Flash image verified 0x80000001 Flash image verification failed 0x00008002 Flash image update in progress 0x00000002 Update Flash Successful 0x80000002 Update Flash Failed 0x00000003 External Memory check successful 0x80000003 External Memory check failed 0x00000010 Concurrency Mode Change Succeeded (auto-switching) 0x80000010 Concurrency Mode Change Failed (auto-switching) 0x00008010 Concurrency Mode Change Best Fit

Table 7-5. Firmware Status Registers (Continued)

Index	Variable	Description
0x00021	DSP_AUTODETECT_MSG	Read Only Refer to Section 7.3.7 for more information. There are times when the DSP needs to inform the host of certain changes during its operation. For example, the DSP sending a message to the host indicating that the input stream has changed from Dolby Digital to DTS. This information is useful to the host so it can update the Front panel of the Unit. This type of message is an unsolicited message because it was initiated by the DSP rather than the host. Changes that are reported to the host are: 1. Changes in the input stream type reported by DSP_AUTODETECT_MSG 2. Changes in the incoming bitstream reported by DSP_LAST_ACCN_MSG
0x00022	DSP_LAST_ACCN_MSG	Read Only Refer to Section 7.3.8 for more information
0x00023	DSP_FLASH_MCU_SCRATCHPAD	Read Only Address in flash of the beginning of the microcontroller scratch pad area
0x00024	DSP_MSG_MASK	Bits to configure what classes of unsolicited messages the DSP should generate Bit 0 = 0/1 Disables/enables Flash Image Verification message Bit 1 = 0/1 Disable/enables unsolicited autodetect messages. Bit 2 = 0/1 Disable/Enable unsolicited Fs message for PCM Input.
0x00025	DSP_FLASH_IMAGE_ADDRESS	Determines the address in the serial Flash where the writable Flash image begins.

7.5 Legacy API Still in Use

There are many common functions for which the legacy APIs should definitely be used, although the new DSP manager firmware makes the use of those APIs easier. For example, the legacy Audio Manager module contains the registers described in [Table 7-6](#), which should still be controlled through the legacy API:

7.5.1 Legacy Audio Manager

Write Command = 0x830nHHHH 0xhhhhhhh

Read Request Command = 0x83CnHHHH

Read Response Message = 0x03C0HHHH 0xhhhhhhh

n = number of words – 1 (bits [20..16])

0xHHHH = Index

0xhhhhhhh = Data Value

Table 7-6. Legacy Audio Manager

Index	Variable	Description
0x0000	GAIN	0x00000000-0x7FFFFFFF (-∞ dB to +24 dB). Overall System Gain. Signed value with decimal point to the right of bit 27 (5.27 format). Range is zero to $(16 \cdot 2^{-27})$. Negative values can be used to invert the phase of all the outputs. Default* = 0x08000000 (+0 dB)†
0x0001	MUTE	0/1 = Unmute/Mute Audio Default* = 0x00000000 (unmuted) †
0x0002	CHAN_L_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Left Channel Default* = 0x80000000†
0x0003	CHAN_C_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Center Channel Default* = 0x80000000†
0x0004	CHAN_R_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Right Channel Default* = 0x80000000†
0x0005	CHAN_Ls_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Left Surround Channel Default* = 0x80000000†
0x0006	CHAN_Rs_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Right Surround Channel Default* = 0x80000000†
0x0007	CHAN_Sbl_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Left Surround Back Channel Default* = 0x80000000†
0x0008	CHAN_Sbr_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Right Surround Back Channel Default* = 0x80000000†
0x0009	CHAN_LFE0_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for LFE0 Channel Default* = 0x80000000†
0x000A	CHAN_LFE1_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for LFE1 Channel Default* = 0x80000000†
0x000B	CHAN_LFE2_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for LFE2 Channel Default* = 0x80000000†

Table 7-6. Legacy Audio Manager (Continued)

Index	Variable	Description
0x000C	CHAN_LFE3_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for LFE3 Channel Default* = 0x80000000†
0x000D	Reserved	
0x000E	CHAN_Lt_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Left DualZone Channel Default* = 0x80000000†
0x000F	CHAN_Rt_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Right DualZone Channel Default* = 0x80000000†
0x0010	CHAN_Lk_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Left Auxiliary Channel Default* = 0x80000000†
0x0011	CHAN_Rk_TRIM	0x00000000 – 0x80000000 (0.0 to 1.0) Volume trim for Right Auxiliary Channel Default* = 0x80000000†
0x0012	DAO1_CHAN_0_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 0. A single internal channel may be mapped to multiple outputs. Default* = 0x00000000 (Left Channel Audio Data)
0x0013	DAO1_CHAN_1_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 1. A single internal channel may be mapped to multiple outputs. Default* = 0x00000002 (Right Channel Audio Data)
0x0014	DAO1_CHAN_2_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 2. A single internal channel may be mapped to multiple outputs. Default* = 0x00000003 (Left Surround Channel Audio Data)
0x0015	DAO1_CHAN_3_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 3. A single internal channel may be mapped to multiple outputs. Default* = 0x00000004 (Right Surround Channel Audio Data)
0x0016	DAO1_CHAN_4_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 4. A single internal channel may be mapped to multiple outputs. Default* = 0x00000001 (Center Channel Audio Data)
0x0017	DAO1_CHAN_5_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 5. A single internal channel may be mapped to multiple outputs. Default* = 0x00000007 (LFE0 Channel Audio Data)
0x0018	DAO1_CHAN_6_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 6. A single internal channel may be mapped to multiple outputs. Default* = 0x00000005 (Left Surround Back Channel Audio Data)
0x0019	DAO1_CHAN_7_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO1 channel 7. A single internal channel may be mapped to multiple outputs. Default* = 0x00000006 (Right Surround Back Channel Audio Data)

Table 7-6. Legacy Audio Manager (Continued)

Index	Variable	Description
0x001A	DAO2_CHAN_0_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 0. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000c (Left DualZone Channel Audio Data)
0x001B	DAO2_CHAN_1_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 1. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000d (Right DualZone Channel Audio Data)
0x001C	DAO2_CHAN_2_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 2. A single internal channel may be mapped to multiple outputs. Default* = 0x00000008 (LFE1 Channel Audio Data)
0x001D	DAO2_CHAN_3_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 3. A single internal channel may be mapped to multiple outputs. Default* = 0x00000009 (LFE2 Channel Audio Data)
0x001E	DAO2_CHAN_4_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 4. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000a (LFE3 Channel Audio Data)
0x001F	DAO2_CHAN_5_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 5. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000b (reserved)
0x0020	DAO2_CHAN_6_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 6. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000e (Left Auxiliary Channel Audio Data)
0x0021	DAO2_CHAN_7_REMAP	Selects which internal channel (L, C, R, etc) is routed to DAO2 channel 7. A single internal channel may be mapped to multiple outputs. Default* = 0x0000000f (Right Auxiliary Channel Audio Data)

7.6 OS Firmware Module

7.6.1 Overview

The managers for the two variations of the Operating System (OS) Firmware module (OS_A & OS_B) are identical except that some variables are disabled in one or the other.

7.6.2 OS-A and OS-B Module Manager

Write Command = 0x8100HHHH 0xhhhhhhh

Read Request Command = 0x81C0HHHH

Read Response Message = 0x01C0HHHH 0xhhhhhhh

0xHHHH = Index

0xhhhhhhh = Data Value

- A. Variable is valid in DSPA OS
- B. Variable is valid in DSPB OS

Table 7-7. OS Module Variables

Index	Variable	A	B	Description
0x0008	PCM_AUTODETECT_SILENCE_THRESHOLD	X		<p>Bit [31:0]: Number of samples (Left+Right) of silence upon which the DSP will declare Silence while having detected and currently playing PCM (Autodetect is enabled).</p> <p>Note: Valid for all decoders when configured for PCM pass-through and auto switch. It is recommended that system designers set this value large enough to avoid inter-track silence from PCM Compact Discs. This is a value that should be set to non-zero if the input source is a CD, but set to zero at all other times in order to prevent audio popping noise.</p> <p>Default*: 0x00017700† (The default allows for 1sec of inter-track silence at 48KHz)</p>

7.6.3 Other DSP Audio Manager Registers

The other registers of the DSP Audio Manager module API are controlled automatically by the DSP Manager based on configuration modes determined at design time and built into the Flash image.

§§

Chapter 8

DSP Condenser

8.1 Overview

Cirrus Logic provides the customer with the DSP Condenser application to implement the design capabilities described in [Section P.1](#). The DSP Condenser application has the following features:

- Customer specifies the DSP features such as the decoding modules to be used in the customer application.
- Application matches the appropriate Cirrus firmware with the customer-selected processing and decoding modules and loads it on the customer's Flash device for testing on Cirrus Logic development boards or on the customer's system.

How DSP Condenser provides the control code to the DSP is described in [Figure 8-1](#).

DSP Condenser-Produced Control code

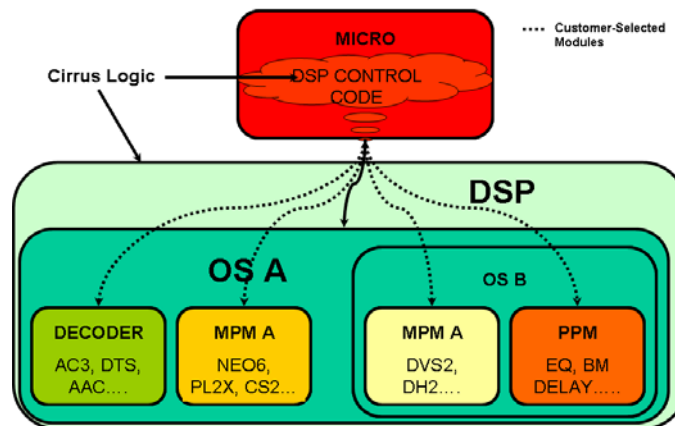


Figure 8-1. DSP Condenser-produced Control Code

8.1.1 Purpose of DSP Condenser

DSP Condenser is a set of tools and a methodology designed to make it easier to develop a Cirrus DSP-based audio system. Using Condenser tools, the system designer can create a flash image that is directly accessible by the DSP operating system, and which contains all of the DSP firmware and most of the firmware configurations necessary to implement the desired DSP audio processing flow, with minimal control required by the system micro-controller.

Conceptually, DSP condenser allows the system designer to:

- Choose, at a high level, what firmware components to include in the design, including decoder and other audio processing algorithms
- Use the DSP Composer graphical tool to tune the parameters to any or all of the firmware components in the design
- “Condense” the chosen firmware components and configurations into a single flash image, ready to be programmed and embedded in the system with the DSP
- Test the flash image using the same high-level API that the system micro-controller should use in the final system

In addition to tools supporting this development flow, the Condenser tool set includes sample micro-controller code, written in C language, which demonstrates how the microcontroller should control the DSP in the final design.

Using this high-level tool set and methodology brings the system designer the following advantages:

- DSP can directly access flash image, making firmware re-configuration faster and requiring less micro-controller effort.
- Most configuration information can be determined during system design rather than during system execution, making early test/validation possible.
- Condenser-based firmware designs provide a simple, high-level API for host control, simplifying the microcontroller coding.
- Condenser-based designs allow much faster auto-switching between compressed stream types, without any micro-controller interaction.
- Condenser tools allow a flash image update to be created as a .wav file for easy deployment to systems in the field. This allows bug fixes or new features to be easily distributed to end-users.

8.2 Development Flow

The basic steps for using DSP Condenser tools are:

1. Install all necessary firmware
2. Create a condenser project from a model
3. Use Composer to modify existing Composer projects within the Condenser project directory, or create new projects.
4. Use the condenser wizard to select firmware configurations from the various Composer projects
5. Create a flash image with the wizard
6. Test the flash image using the Condenser run-time GUI and the Cirrus eval board
7. Deploy the flash image to the desired system and begin micro-controller coding and debugging.

In more detail, use the following steps for the smoothest implementation using the DSP Condenser tool set:

1. Make sure you have all of the firmware components that you need already installed in the same place as your Cirrus DSP SDK. This includes any firmware that requires licenses from 3rd party IP vendors such as Dolby or SRS®.
2. Start the DSP Condenser Wizard from the SDK start menu subfolder for the DSP you have targeted for your system (e.g. CS4953X or CS497XX).
3. In the wizard, create a new project starting with one of the sample model projects that ship with the SDK.
4. In this initial phase of project startup, you need not be too concerned with all of the fields you will encounter in the wizard. For now, you should:
 - a. Enter some appropriate values for the "Project version" and "Manufacturer" in the General tab
 - b. Enable the Audio sources you want to support, and disable those you do not want to support. You can change your mind later, but it is good to make a reasonable decision now.
 - c. Select the Firmware components you want to support in your design, and make sure the others are de-selected. Don't worry about the "Modes" sections for now.
 - d. Enable the Stream types that you want your system to be able to decode. Don't worry too much about the "modes" sections. If you know what Matrix, Virtualizer, or Post-processor you want to run

- for any given stream type, you can specify it now. You can always come back and change these settings later.
- e. Specify a reasonable Power-up state for your system design.
 - f. Save the project.
5. Use the Build/Create flash image command in the wizard to actually create your first flash image. This is just a test of the build process - the image produced is not the final image.
 6. Use the Build/Explore build outputs command to open an explorer view of the project build structure. Notice that the project includes Composer projects (.cpa files), generated deliverables from those Composer projects (deliverables), as well as the actual outputs. The deliverables folders are the source of the "modes" that can be specified in the Condenser wizard.
 7. Close the Condenser project.
 8. Open Composer, open any of the .cpa files in the Condenser project, use Composer to try different firmware configuration settings.
 9. For each set of settings in a Composer project that you want to save and use in the Condenser project, create a Composer snapshot, and give it a meaningful name. After all snapshots are captured for this project, use the Composer Generate deliverables command to generate deliverables. Be sure to save the deliverables into the Condenser project "deliverables" folder! The snapshots won't be available to Condenser if you do not.
 10. Repeat steps 8-9 until all firmware modules included in the design have desired configurations saved as snapshots in at least one Composer project. Organizationally, it might be best to save the snapshots for each firmware component in a separate Composer project, although you may combine configurations for multiple components within a single Composer project if you like.
 11. After all desired configurations are saved in Composer-generated deliverables folders, open the Condenser project in the Condenser wizard again. Now you can go through all of the Firmware components and Stream types and assign snapshots (also known as "modes") to the appropriate firmware. Each "mode" for a component is a set of configuration messages that can be read and enabled in the DSP by a simple 3-word command from the microcontroller.
 12. After all modes are specified, be sure to update the concurrencies in the Stream types section as desired. A concurrency is the full set of firmware components that are to be loaded when a particular stream type is recognized by the DSP OS, including Decoder, Matrix, Virtualizer, and Post-processor components.
 13. After all concurrencies and modes are specified, use the Build/Create flash image command again to create a new flash image.
 14. You can also use the Build/Program flash on board command to program the flash on the Cirrus eval board.
 15. You can use the Build/Run runtime GUI command to exercise various elements of the project on the eval board.

See [Section 8.6.2 on page 8-19](#) for a step-by-step walkthrough of this entire process.

8.3 Elements of a Project

The contents of a project are represented in the DSP Condenser Wizard GUI using “Master–Details” arrangement. This arrangement consists of a tree of project elements (Master) on the left side of the GUI window and a set of properties (Details), displayed on the right, which vary according to the selection in the master. Refer to the following screenshots of DSP Condenser sections for detailed descriptions of a project's content.

8.3.1 General Page

The *General* page is used to define common project settings.

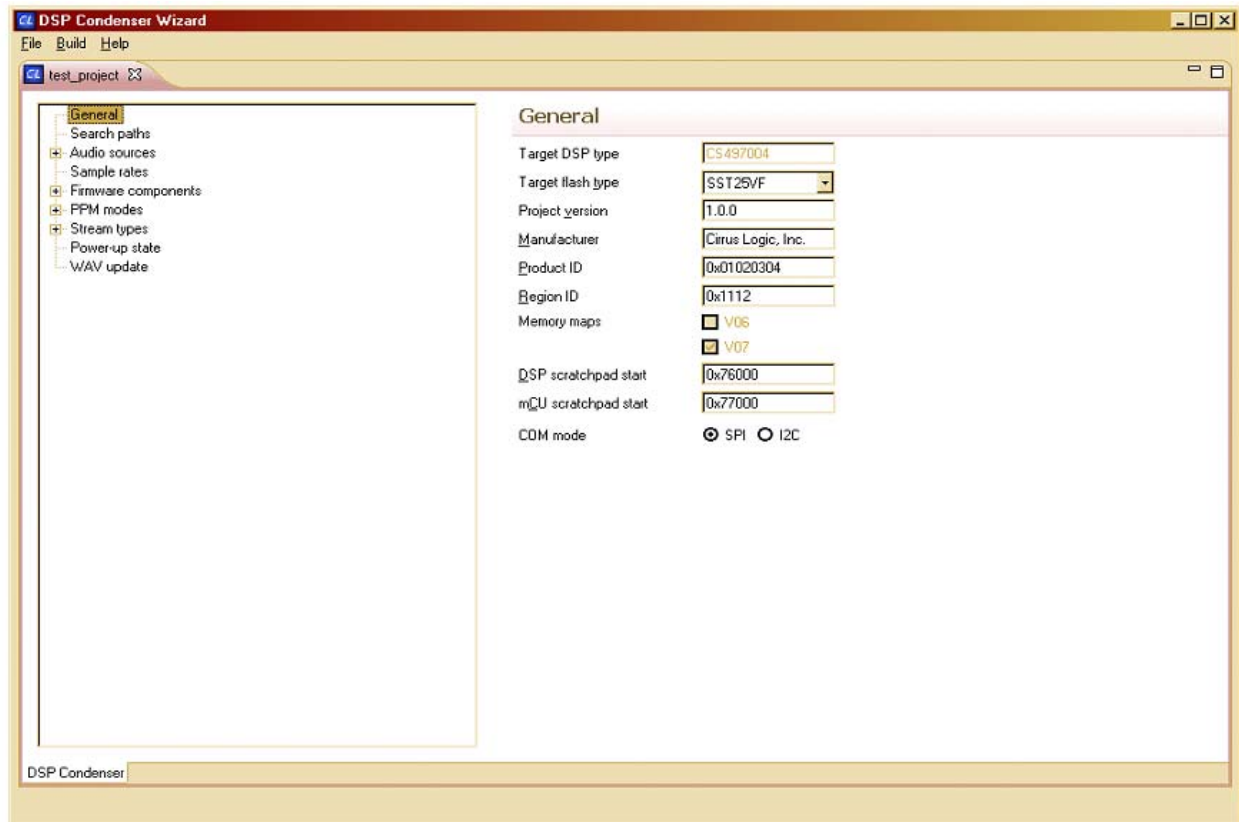


Figure 8-2. DSP Condenser Wizard, *General* Page

Target DSP type — Displays the target DSP chip type for the current project (read only).

Target flash type — Selects the type of serial flash used in current project.

Project version — The 20-character version number of flash image. This number should be incremented when new versions are released. The project version string will be programmed into the flash image and can then be accessed by the MCU using the DSP Manager API.

Manufacturer — The 20-character manufacturer's ID (optional).

Product ID — The project's unique, 32-bit product ID (optional).

Region ID — The 16-bit region ID (optional).

Memory maps — List of supported firmware versions (read only).

DSP scratchpad start — Beginning address of the flash area used by the DSP to save runtime settings.

MCU scratchpad start — Beginning address of the flash area used by the MCU to save runtime settings.

COM mode — Selects the serial communication protocol (SPI or I²C).

8.3.2 Search paths Page

The *Search paths* page is used to define the search paths for specified ULDs and CFGs. The search is performed first under subdirectories named for the supported versions (listed in the *General* page), then directly in the specified directories. The search is performed in the order in which the paths are specified in the list.

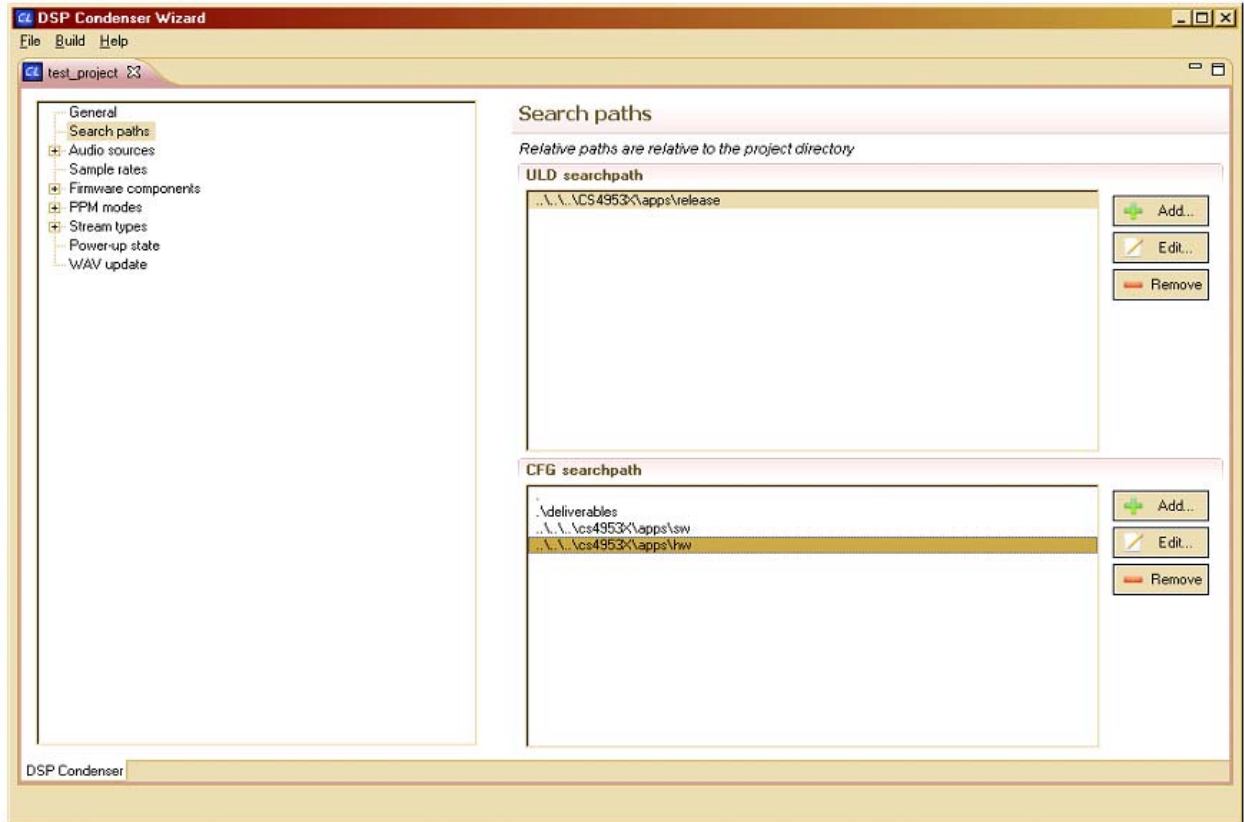


Figure 8-3. DSP Condenser Wizard, *Search paths* Page

8.3.3 Audio sources Page

The *Audio sources* page contains a list of all possible sources of audio presented to the DSP. Each audio source selected must have one or more .cfg files specified that contain the appropriate configuration messages so that the firmware OS can properly configure the DSP for receiving audio from that source.

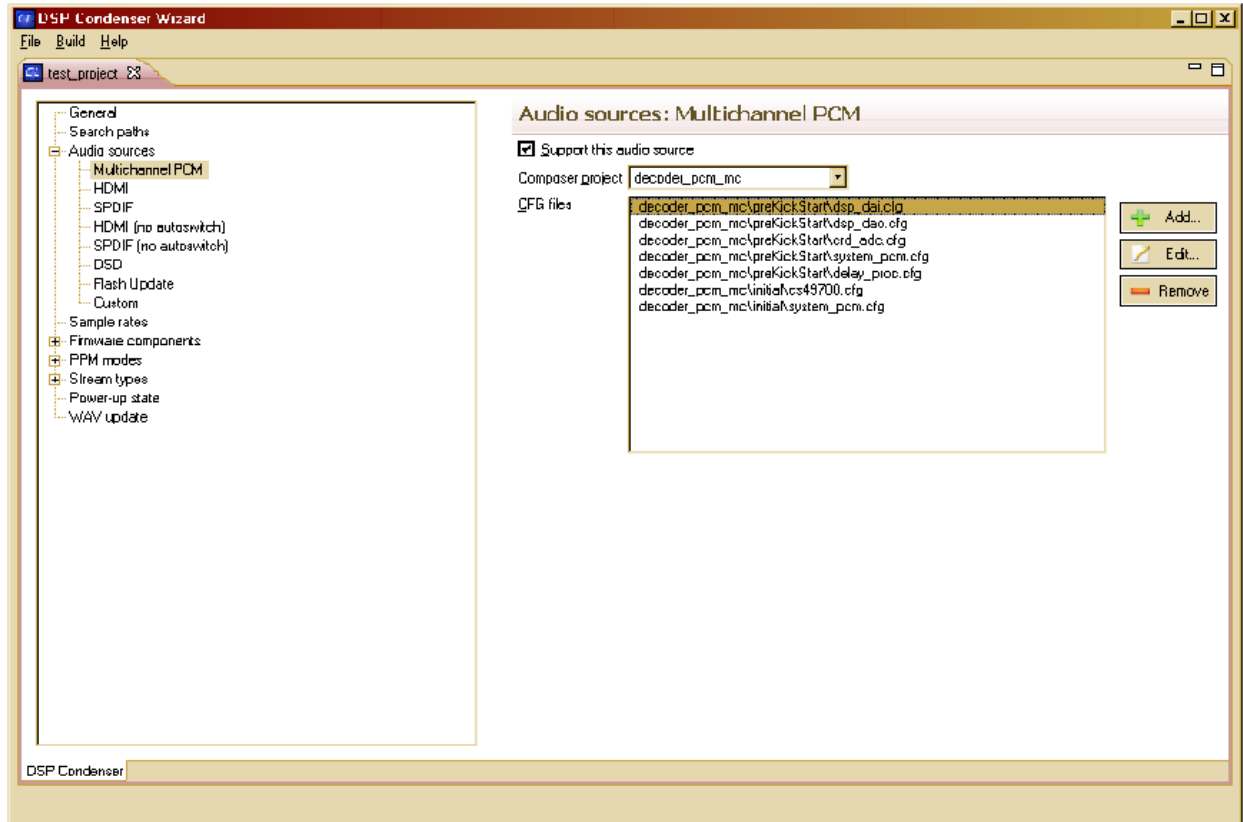


Figure 8-4. DSP Condenser Wizard, *Audio sources* Page

Support this audio source — Check box used to select support for the audio source highlighted.

Composer project — List of DSP Composer projects that have been saved in the current project's directory.

CFG files — List of .cfg files necessary to support the selected audio source.

8.3.4 Sample rates Page

The *Sample rates* page defines all the necessary hardware configurations to support various input and output sample rates.

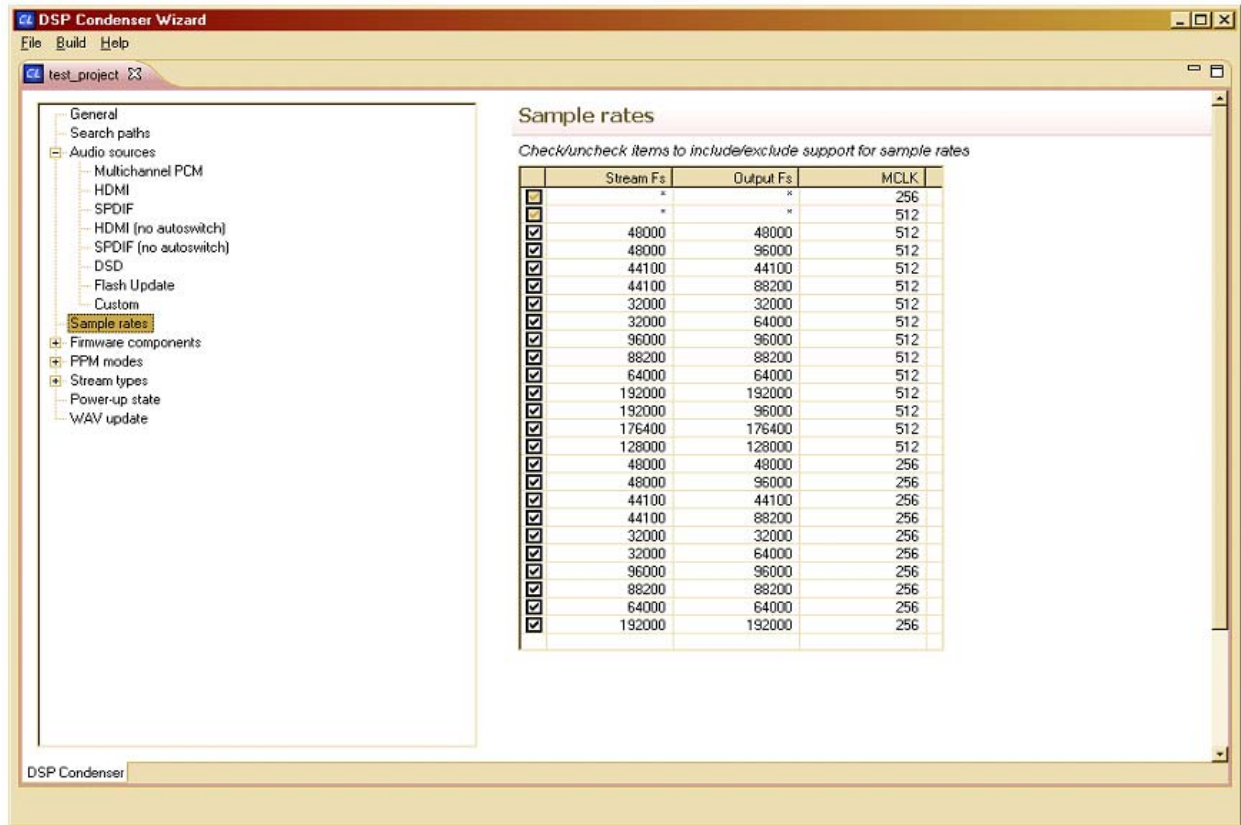


Figure 8-5. DSP Condenser Wizard, *Sample Rates* Page

8.3.5 Firmware components Page

The *Firmware components* page includes all of the firmware components (also referred to as "overlays" or ULDs) that are to be supported by the system being designed. This includes all decoders, matrix processors, virtualizers, and post processors grouped by category.

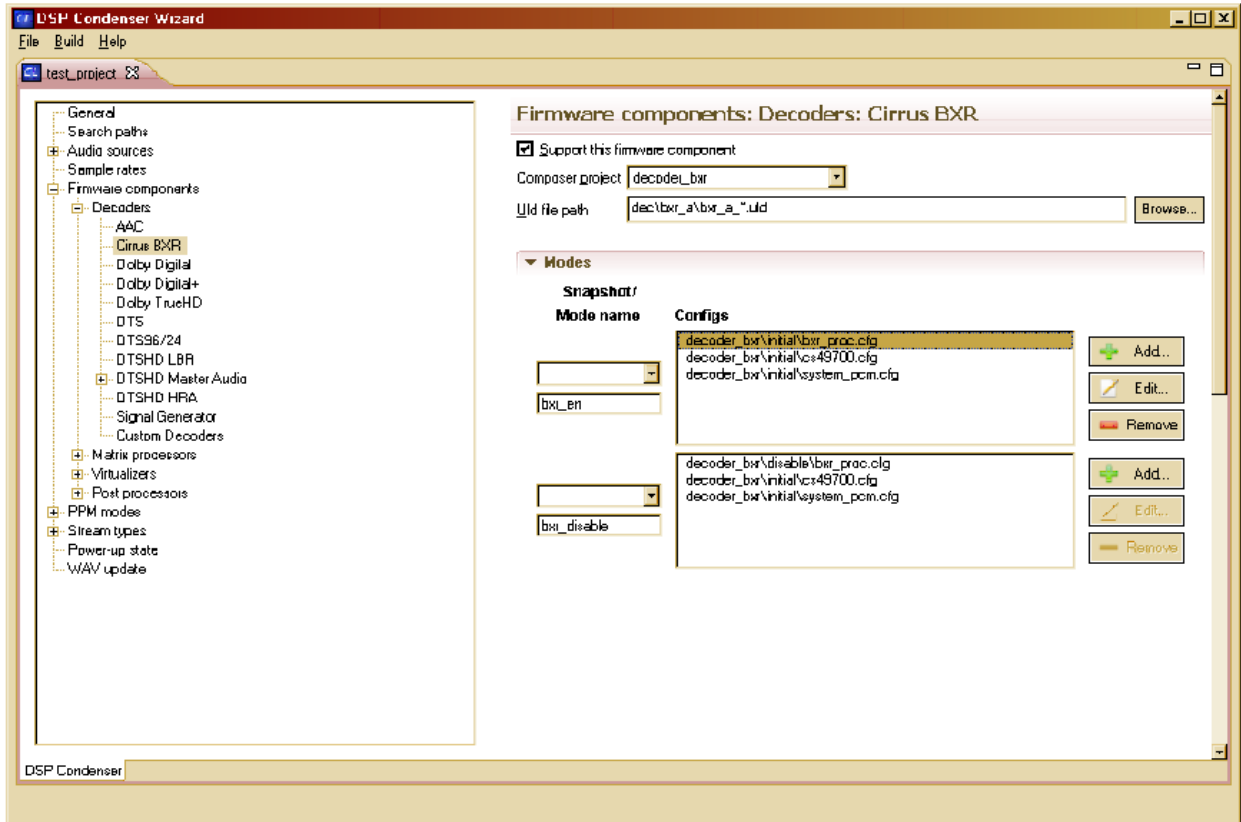


Figure 8-6. DSP Condenser Wizard, *Firmware components* Page

Support this firmware component — Check box used to select support for the firmware component highlighted on the left.

Composer project — List of DSP Composer projects that have been saved in the current project's directory.

Uld file path — Specifies the ULD search path. If the filename contains '*', then the latest version of the ULD will be used, otherwise the specific version will be used

Configs — One or more .cfg file definitions. Each .cfg definition is simply the name of a .cfg file. The filename will be searched for in the directories specified by the CFG search paths specified in the *Search paths* page. The files are concatenated to form a single set of control messages that constitute the "mode" The mode .cfg files for a particular ULD should come from the different snapshot subdirectories from the deliverables generated in a DSP Composer project that contains this ULD. The .cfg file that is used is the one that matches the ULD. For example, if the modes are for the ac3 decoder, the .cfg files would be the "decoder_ac3.cfg" files from the various snapshot subdirectories in an ac3 project deliverables folder. The mode should also include any ULD-specific .cfg files from the preKickstart subdirectory of the project deliverables folder, if any. For most modules there are no such .cfgs, but for the HD decoders there usually are.

Snapshot — List of snapshots created in DSP Composer for this project.

Mode name — Customized name for selected snapshot (this name is used later throughout the GUI). This is also incorporated in the output .h file produced, as part of an enum that can be used by MCU code to specify the index.

8.3.6 PPM modes Page

The *PPM modes* page defines which CFGs are necessary, depending on what mode is selected. PPMs are handled differently in the DSP Manager API than other overlays, so they are defined in the flash image differently as well. Unlike other overlays, PPM ULDs frequently have many different modules in them (rather than only one or two).

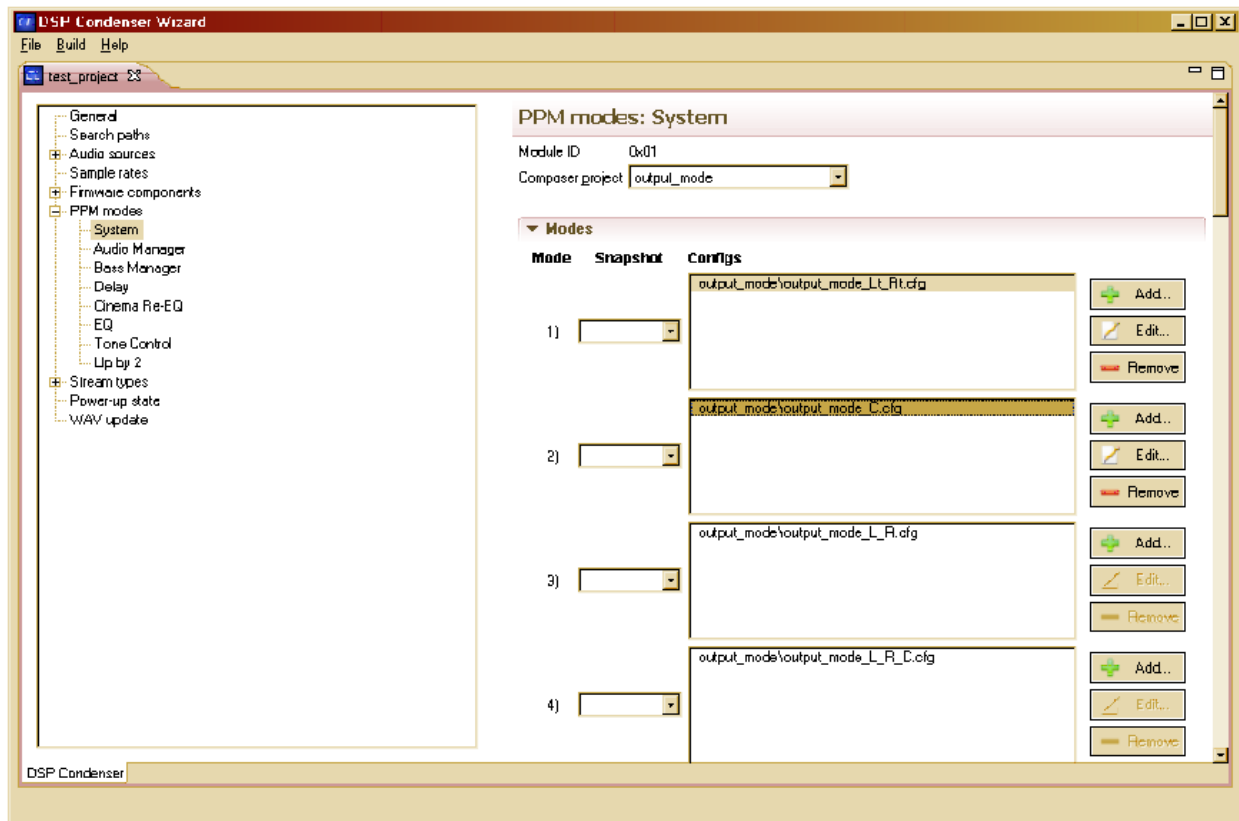


Figure 8-7. DSP Condenser Wizard, *PPM modes* Page

Module ID — The unique ID for the mode highlighted on the left (read only).

Composer project — list of DSP Composer projects that has been saved in current project's directory.

Snapshot — List of snapshots created in DSP Composer for this project.

Configs — One or more .cfg file definitions. For more detailed explanation see the *Configs* description under the *Firmware components* page.'

8.3.7 Stream types Page

The *Stream types* page contains a list of possible stream types. When autodetect occurs, the concurrency mode attached with the stream type is automatically loaded by the DSP. All modules and modes defined in the concurrency mode are loaded when a particular associated stream type is detected. Concurrency mode is the combination of firmware modules active at any one point in time.

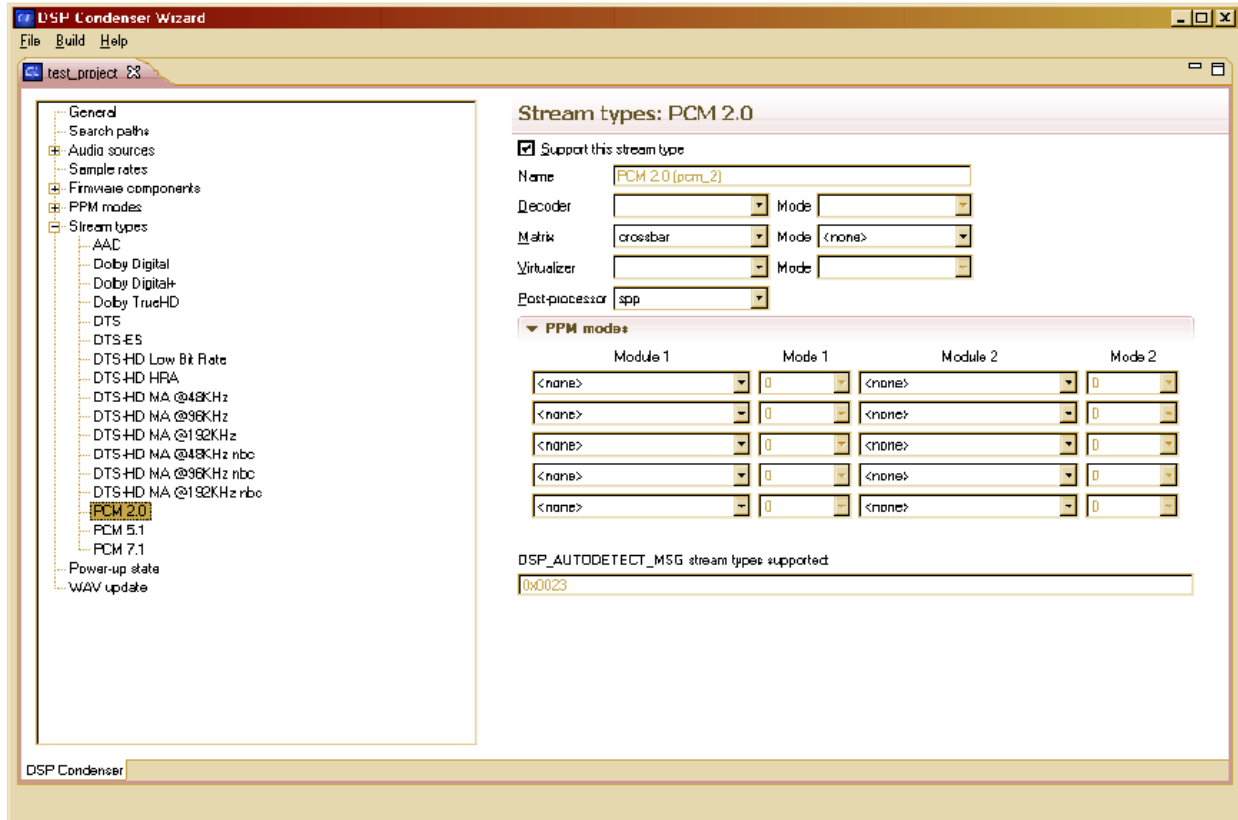


Figure 8-8. DSP Condenser Wizard, *Stream types* Page

Support this stream type — Check box used to select support for the stream type highlighted on the left.

Name — The stream type name (read only).

Decoder — Defines the decoder to run when the stream type is detected.

Matrix — Default matrix for the stream type.

Virtualizer — Default virtualizer for the stream type.

Post processor — Default post processor for the stream type.

PPM modes — List of supported PPM modules and their modes.

8.3.8 Power-up state Page

The *Power-up state* page is used to define the default DSP state at power-up. This page defines initial modules (and their modes) to be loaded immediately after master boot.

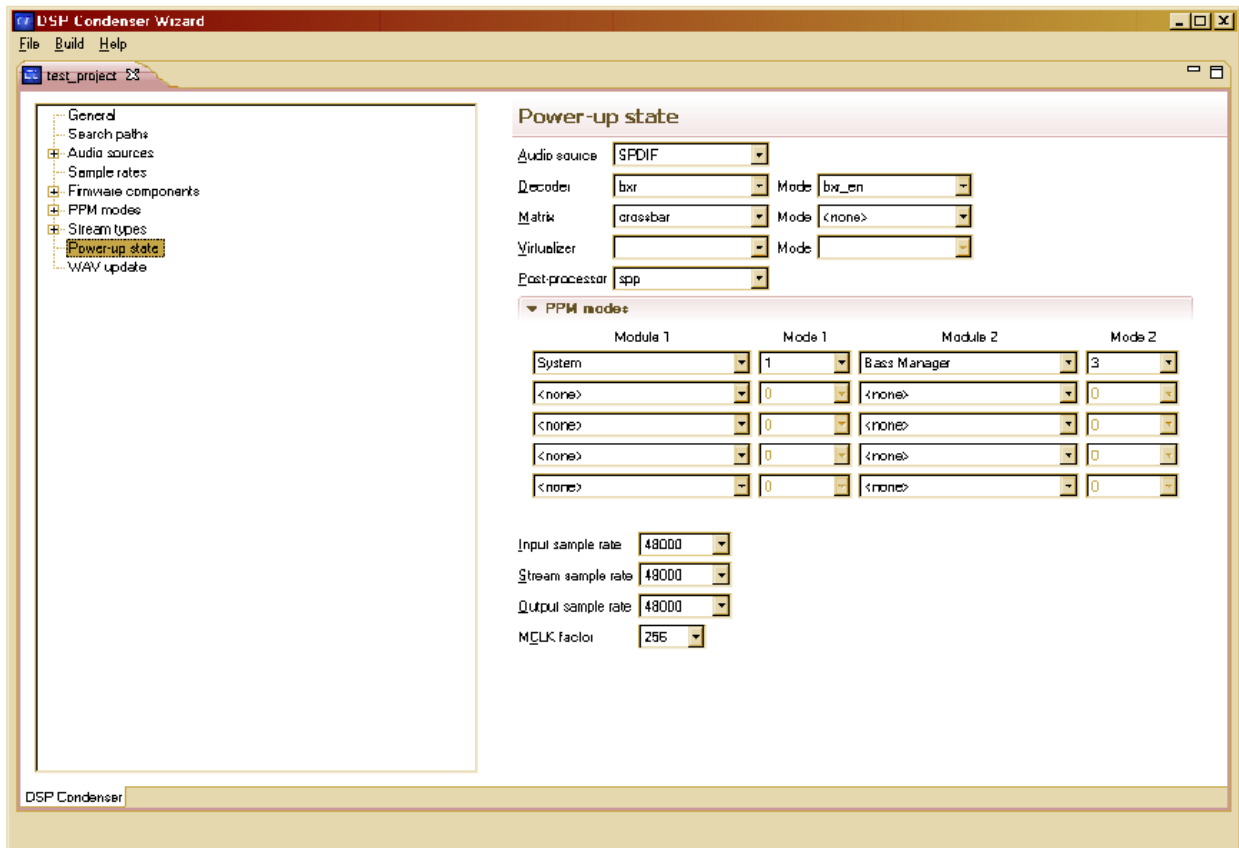


Figure 8-9. DSP Condenser Wizard, *Power-up state* Page

Audio source — default audio source for power-up state

Decoder — default decoder for power-up state

Matrix — default matrix for power-up state

Virtualizer — default virtualizer for power-up state

Post processor — default post processor for power-up state

PPM modes — lists of supported PPM modules and their modes for power-up state

Input sample rate — default input sample rate for power-up state

Stream sample rate — default stream sample rate for power-up state

Output sample rate — default output sample rate for power-up state

MCLK factor — default MCLK factor for power-up state

8.3.9 WAV update Page

For easy deployment to systems in the field, a flash image update file can be created. The field upgrade SPI Flash image is created using the flash_to_wav utility, and contains multiple data structures, ULDs, and configuration message sets. The flash update image is formatted as a .wav file. The first part of the file is the .wav header. The data payload of the .wav file is organized as an IEC-61937 (S/PDIF) digital stream.

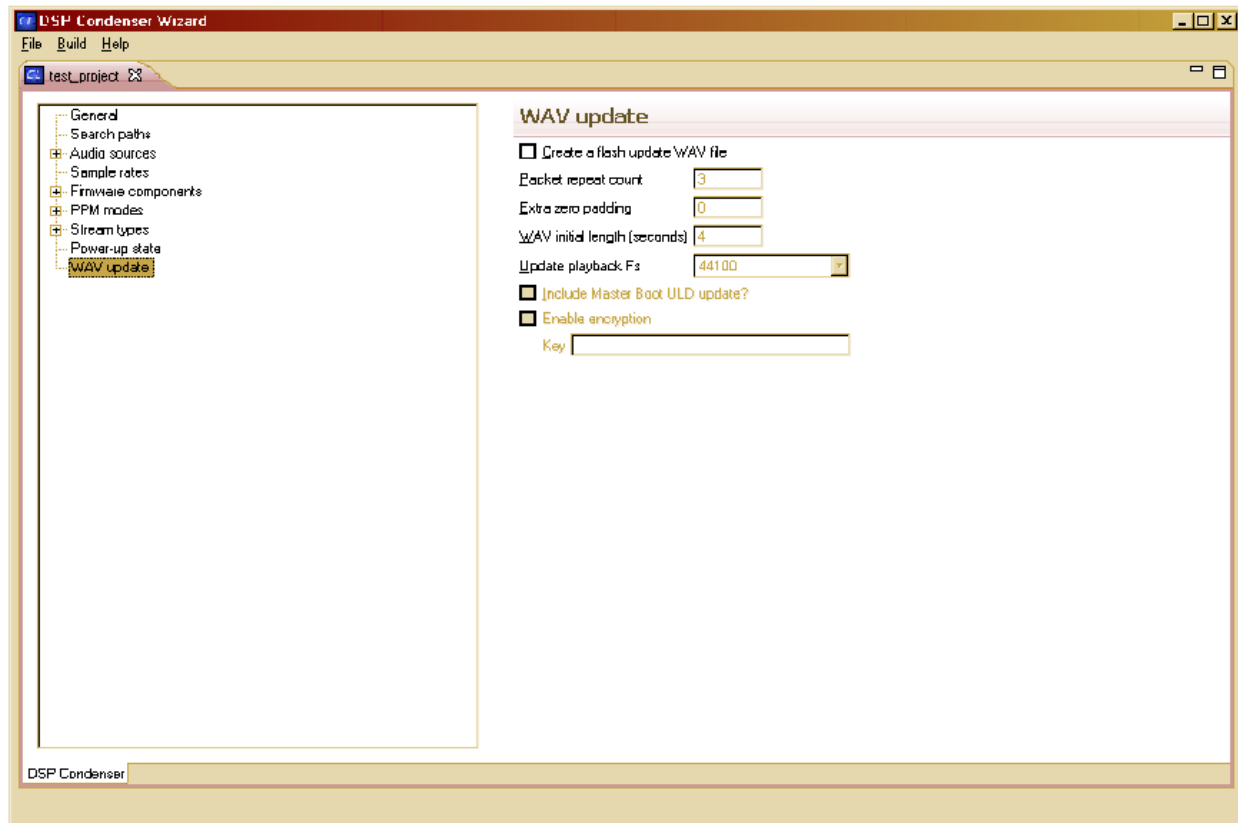


Figure 8-10. DSP Condenser Wizard, WAV update Page

Create a flash update WAV file — Select this check box to create flash update file.

Packet repeat count — Selects the number of times each packet should be repeated to ensure a good chance of correct parsing from the audio stream.

Extra zero padding — Selects the number of extra zeros that should be inserted between packets to allow time for CRC check calculation.

WAV initial length (seconds) — Selects the number of seconds that the update header information will take up at the front of the .wav file.

Update playback Fs — Selects the sample rate at which the .wav file will be transmitted to DSP.

Include Master Boot ULD update — The master boot ULD is supplied by Cirrus Logic. If you have been given an update file you may want to include it in the update .wav file.

Enable encryption — The encryption feature is not available for CS4953x or CS497x.

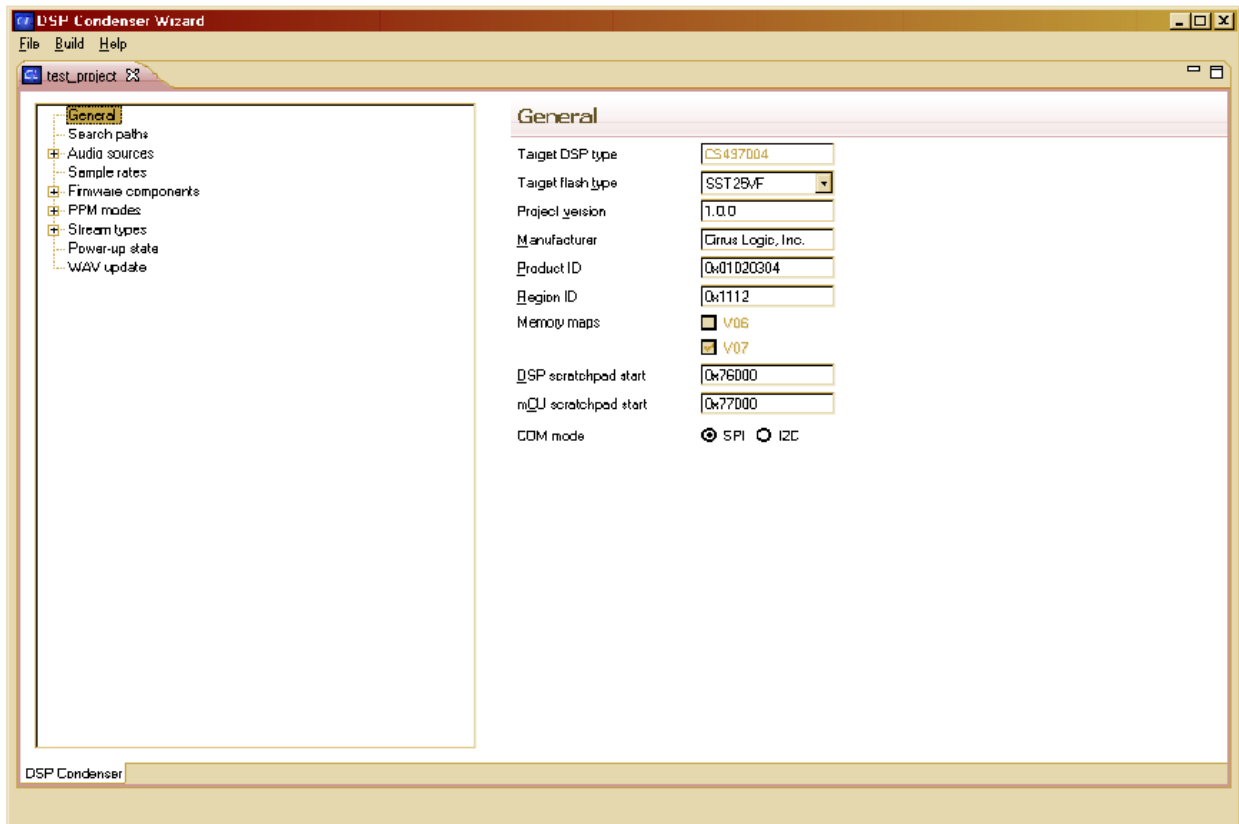
8.4 Creating a Condenser Project using a Model

8.4.1 Using the Wizard to Create a Project

1. Install the DSPCondenser.exe and the required firmware modules into the cs4953x_eval_kit install directory (Eg: C:\CirrusDSP).
2. Open dspcondenser wizard from the *Start Menu* ⇒ *CirusDSP.c* .
3. To create a new project, Click on *File* ⇒ *New project*.
4. Type a suitable project name. Choose the appropriate model. The models basically provide a starting point with example configurations for the project.
 - Sample model corresponds to a project with unlicensed technologies (suitable for CS4953x4, CS4970x4)
 - Sample_licensed_legacy model corresponds to a project with legacy licensed firmware modules (suitable for CS4953x4)
 - Sample_licensed model corresponds to a project with legacy and HD licensed firmware modules (suitable for CS4970x4)

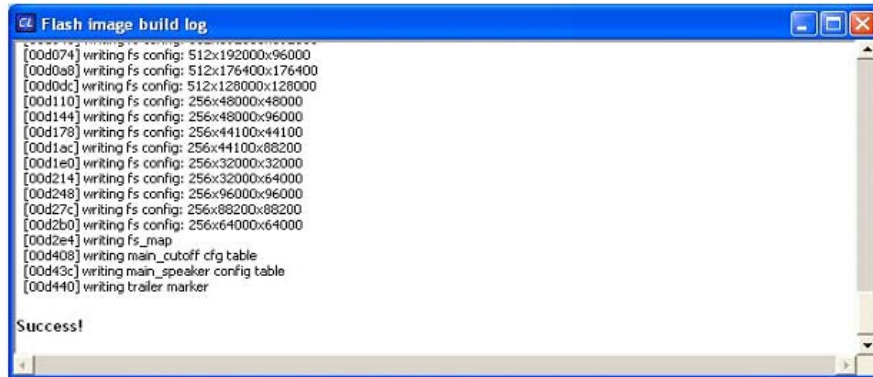
This creates a project with the provided project name in
C:\CirrusDSP\DSPCondenser\projects*<project_name>*

Click **OK** and the main screen of the new project window should appear as below.



Click on Build->Create Flash Image

The build log window will open displaying all the files (*.uld, *.cfg etc) listed in the project (underlying xml file) and also display if the image creation process was successful or not.. If there is a failure, scrolling through this log will give the error messages. The log will mention any missing uld or .cfg files.



8.5 Creating a Flash Image

8.5.1 How to create an image

To create a flash image for testing or final deployment, use the Build/Create flash image command in the Condenser wizard. This command runs a makefile that will:

- Determine the size of the flash image.
- Run a utility to actually create a hex ASCII flash image file which includes all specified DSP firmware and configuration files.
- Run a utility to create a .wav update file, if the project specifies to create one.

When creating a flash image, it is important that the target flash type be properly specified. If you are creating the flash image for testing on the Cirrus Logic evaluation board, the flash type should be SST25VF. If you are creating the flash image for deployment on your own board, this setting should be appropriate for the flash type used in your design.

After the flash image is created, you can use the Build/Explore build outputs command to see the results of the build. In the outputs folder are the following important files:

- **egg.img** — The startup DSP code that is programmed into flash at location zero
- **flash.img** — The main flash image that contains all other DSP firmware components and configuration files
- **flash_update.wav** — The flash.img file in .wav format, if you specified to create a .wav update file
- **flash.h** — A C .h file that contains information about the flash.img file which must be used by the micro-controller code as it uses the high-level DSP Manager API. Specifically, it contains a mapping of symbolic names for the various firmware components and modes defined in the condenser project to integer values used in host-to-DSP communication.

The other files in the outputs folder are intermediate output files that are not necessary for actual deployment or debugging.

8.5.2 What Does the Image Contain?

When the DSP operating system is running from a DSP Condenser project, it considers the flash attached to the DSP to be organized as shown:

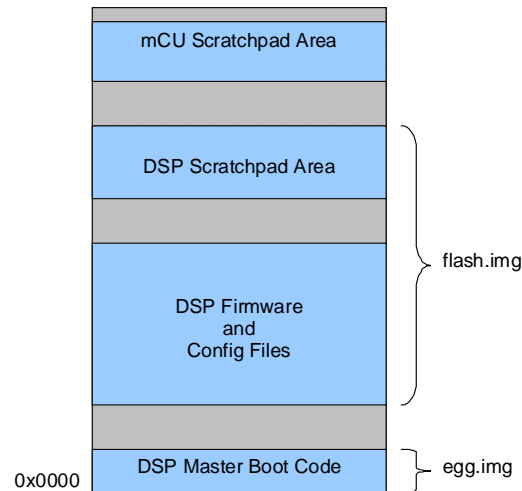


Figure 8-11. DSP Condenser Project Flash Memory Organization

The spaces between the used portions of the flash indicate that each of the four used portions of flash must be aligned on a flash sector boundary to allow programming each section individually.

As described in the preceding section, there are actually two images that are created by Condenser, the egg.img file and the flash.img file.

The egg.img file contains DSP master boot image that must be programmed at location zero in the flash. It is a generic image that is patched with specific parameters for the flash type and starting location of the second flash image.

The flash.img file is the main output of Condenser. It contains all of the DSP firmware (ULDs) that are included in the design, plus all of the configuration file information specified as "modes" in the Condenser project. In addition, it may optionally include some prefilled configuration file information that is loaded into an area of flash called the "DSP scratchpad area", which is a place where the micro-controller can store configuration file information that is determined at runtime or room calibration time after the system is deployed.

8.6 Using DSP Condenser

8.6.1 How to use DSP Composer with DSP Condenser

Part of the goal of DSP Condenser is to allow Cirrus customers to specify as many firmware parameters at design time as is reasonable for their design. These design-time parameters can then be stored in the Flash image and downloaded as “modes” using the DSP Manager API, significantly reducing host-to-DSP communications requirements. This also reduces ROM/Flash requirements for the host microcontroller, and allows DSP mode-switching to be faster, since the DSP can access Flash faster than typical host communications.

The easiest way to generate design-time configurations for later saving in the Flash image is to use DSP Composer projects with saved snapshots. Each snapshot of a DSP Composer project contains all of the control settings for all of the elements in the DSP Composer project design. A single project can contain as many snapshots as desired.

By using DSP Composer to generate these configurations, system designers can actually try the settings (using the CRD board) to verify correct parameter values. Once the controls are at the desired values for a particular “mode” of the system under design, those values should be saved as a snapshot in DSP Composer. Required naming conventions for snapshots will be discussed below. DSP Composer itself does not impose any naming restrictions, but for integration with the DSP Condenser tool set, some naming conventions must be followed.

The following description assumes that the Cirrus Logic SDK is installed in this location:

```
c:\CirrusDSP (the default location).
```

8.6.1.1 Best Practices

Directory Structure

DSP Condenser-based projects should start a new folder in the following location:

```
C:\CirrusDSP\DSPCondenser\projects
```

Within that folder, create subfolders for:

- `cpa_files`, where DSP Composer project `.cpa` files will be stored
- `Deliverables` (where DSP Composer “Generate Deliverables” output will be stored).

For example a project named “WhizBang Model” would have the directory structure shown in [Figure 8-12](#):

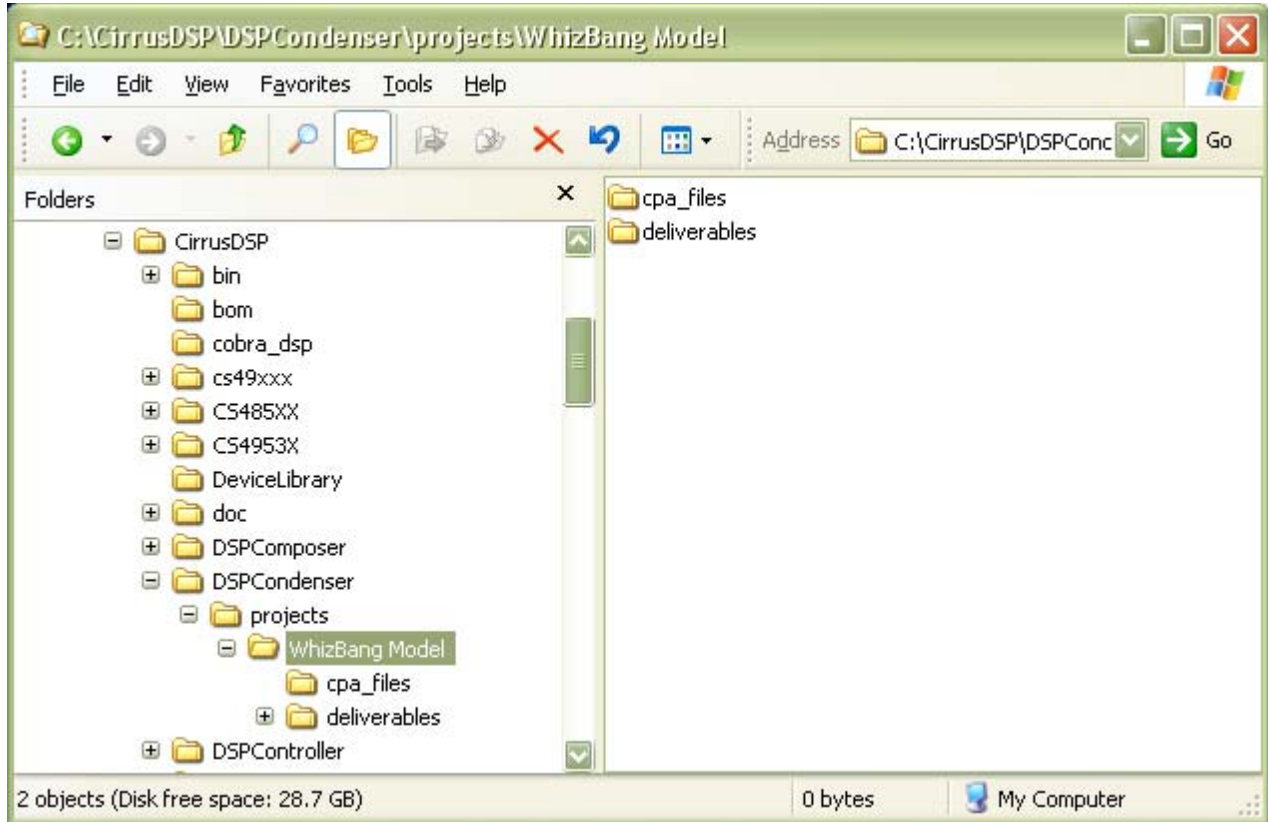


Figure 8-12. DSP Composer Sample Project, “WhizBang Model” Directory Structure:

8.6.1.2 Creating Projects

A total DSP Condenser system design includes all of the firmware components (also referred to as “overlays” or ULDs) that are to be supported by the system under design. This includes all decoders (for example, AC3, TRUEHD, PCM, HDCD), matrix processors (for example, NEO6, PL2X), virtualizers (for example, Dolby Headphone), and post processors (for example, PEQ, Bass Manager, Delay).

Because DSP Composer only supports projects that include one decoder, matrix processor, virtualizer, and PPM, multiple Composer projects must be used to specify all of the settings for all of the firmware components in the entire system design. Each project should be saved in the following directory:

```
c:\CirrusDSP\DSPCondenser\projects\

```

The number of project (.cpa) files required to represent the entire system design is, of course, design-dependent. Designers may choose to create a separate project for capturing settings for each component, or combine multiple components in a single project. For example, a project that includes AC3 must also have a PPM in order to compile, so SPP would be included. Snapshots that are captured from this project will include settings for AC3 and all of the modules in SPP. Regarding SPP configuration settings, the system designer can choose either of these options:

- Consider the SPP settings generated from this project as significant, that is, to be included in final system.
- Ignore SPP settings generated from this project

DSP Condenser allows either option.

Cirrus Logic recommends that each project be used to define settings for only one overlay, even though this means that more projects will be required. Such a separation makes it easier to separate out the effects of one component's settings during listening/testing of the project.

8.6.2 Capturing Snapshots

DSP Composer "snapshots" are sets of firmware module parameter settings, captured in DSP Composer using the Tools/Snapshots menu item. See [Figure 8-13](#) for an example of a snapshot created in DSP Composer. System designers should generate snapshots containing all of the desired preset configuration settings, and name them using a convention that is significant for the system designer. Snapshot names are not visible to end-users of the system.

For each Composer project, manipulate the controls of the firmware component being tuned in the project until a desired audio processing mode is achieved. Capture a snapshot of the control settings, giving the snapshot a name that is useful in remembering what the settings are supposed to mean.

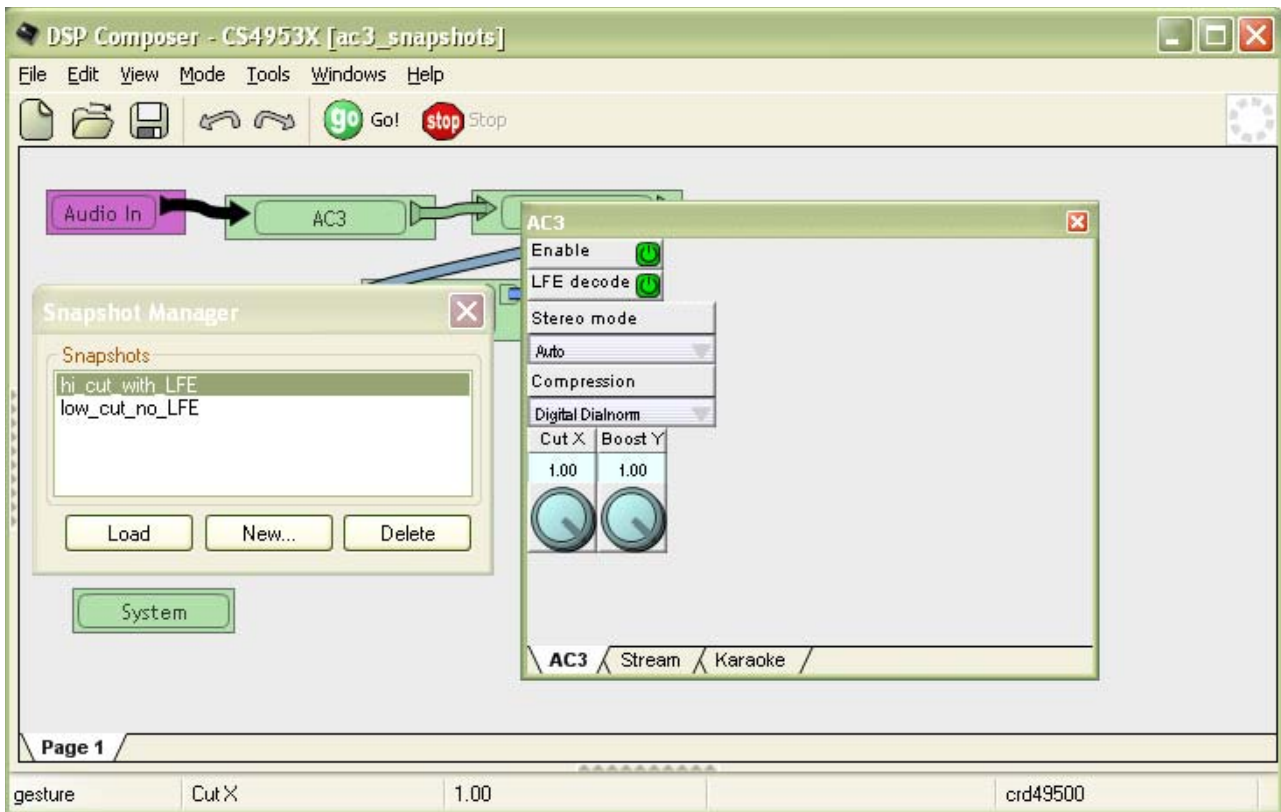


Figure 8-13. DSP Composer Snapshot

Once the different snapshots are verified, use the Tools/Generate Deliverables menu item to save the deliverables to the following folder:

C:\CirrusDSP\DSPCondenser\projects\<<project>\deliverables

The directory structure: should look something like

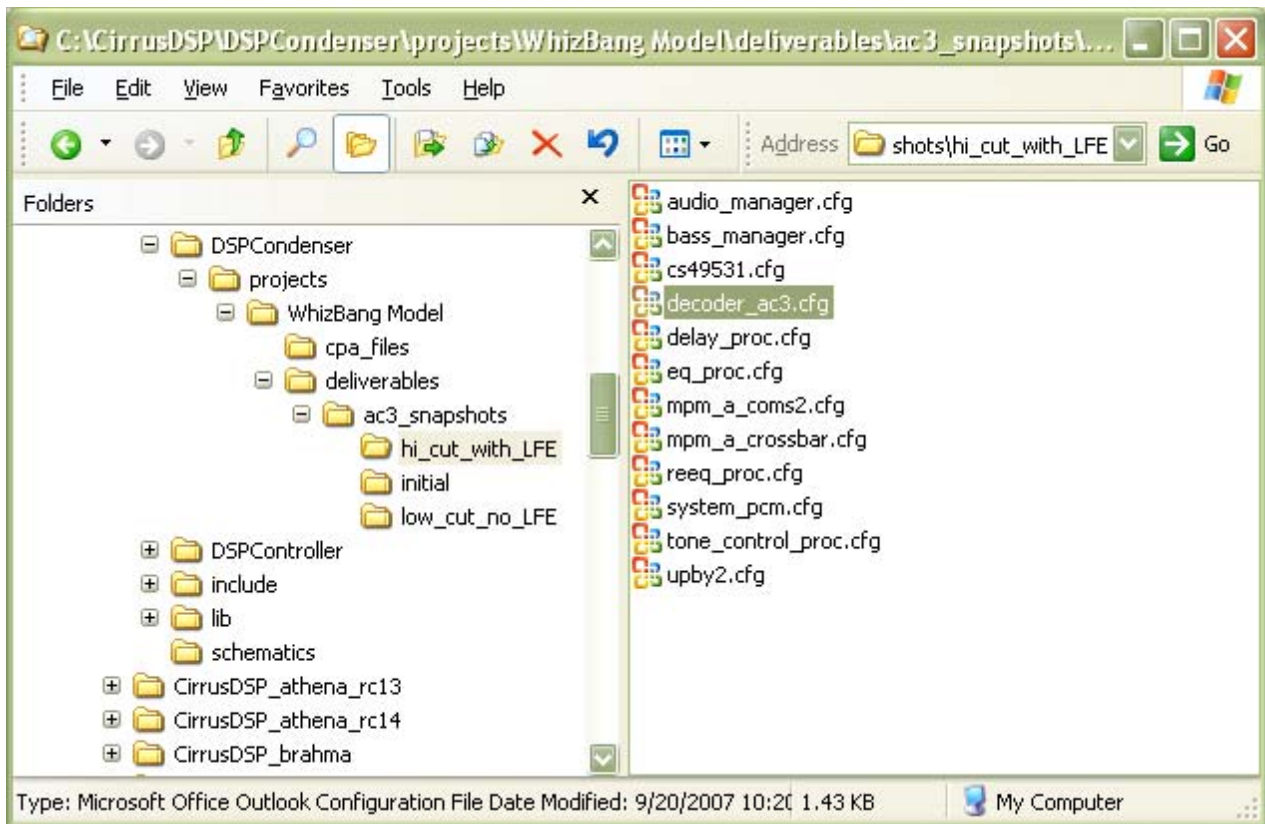


Figure 8-14. Sample Deliverables Directory Struction

When DSP Composer generates deliverables, the configuration parameters (.cfg file) for each firmware component is saved into a separate file, with all .cfg files for a specific snapshot saved into a directory named for the snapshot. These individual .cfg files will later be used by other DSP Condenser tools to be incorporated into the Flash image.

For an example of how to create DSP Composer deliverables, follow Steps 1 to 5 in the section, **“Creating DSP Composer Deliverables.”**

8.7 Creating a Flash Image

8.7.1 SPI Flash Image Format

The SPI Flash supports auto-switching and contains DSP firmware code and customer configuration data. [Figure 8-15](#) shows the format of the SPI Flash. The various components of the SPI Flash format are discussed in [Section 8.7.1.1.](#), [Section 8.7.1.2.](#), [Section 8.7.1.3.](#), and [Section 8.7.1.4.](#).

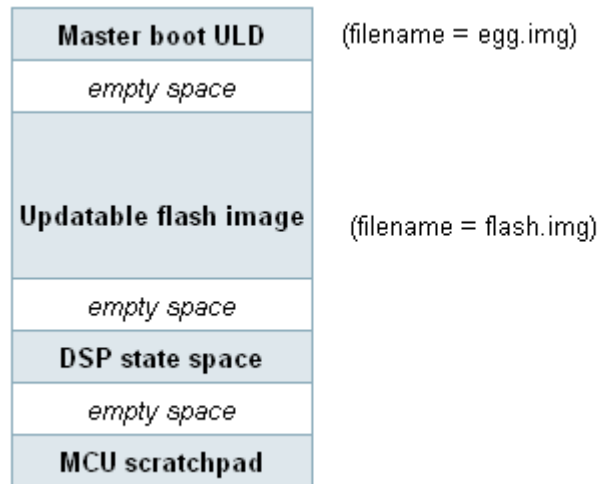


Figure 8-15. Blank SPI Flash Format

8.7.1.1 Master Boot Image (egg.img)

The Master Boot Image, which is created by the DSP condenser tool, is customer system-specific. The Master Boot image is located in a locked area of SPI Flash, and must start at start at location 0x0000. The area designated for the Master Boot image is **not** updated during Field Flash updates. See [Section 8.7.1.2](#). for more information. The Master Boot image contains code to:

- Lock the PLL to a nominal 150 MHz
- Read/write the attached Flash
- Validate the updatable Flash image. If image is not valid, automatically goes into Flash update mode
- Run Flash update from audio source
- If Flash image is valid, boot either the power-down saved state or a factory-set default state

The Master Boot image has a pointer in it to the flash_address where the remainder of the updatable Flash image is stored. This value is patched into the Master Boot image via uld_merge by the DSP Condenser tool during the design stage, when the Flash images are being created. It also has a pointer to the DSP State Space (described in [Section 8.7.1.3](#)) in Flash. This pointer is also patched into the Master Boot image when the Flash images are being created.

8.7.1.2 Updatable Flash Image

The Field Upgrade SPI Flash image is created using the flash_to_wav utility, and contains multiple data structures, ULDs, and configuration message sets. The Flash update image is formatted as a .WAV file, so the first part of the file is the .WAV header. The data payload of the .WAV file is organized as an IEC-61937 (SPDIF) digital stream.

8.7.1.3 DSP State Space

An area of Flash reserved for DSP usage. It is used to store:

- The power-off state
- Configuration values calculated by room calibration modules

Note: This area of Flash may be destroyed when a Flash image is updated in the field, thereby losing any room calibration values. However, it is intended that this space be separated from the rest of the Flash image as much as possible, so that it may be preserved between Flash updates.

8.7.1.4 Microcontroller Unit (MCU) Scratchpad

An area of Flash reserved for MCU usage. The middleware API provides access to this memory (indirect through the DSP). The DSP and middleware make no constraints on the use of this space. Flash image upgrade *must not destroy this area*

8.7.2 Creating a Serial Flash Image Manually

Figure 8-16 shows the serial Flash image creation process.

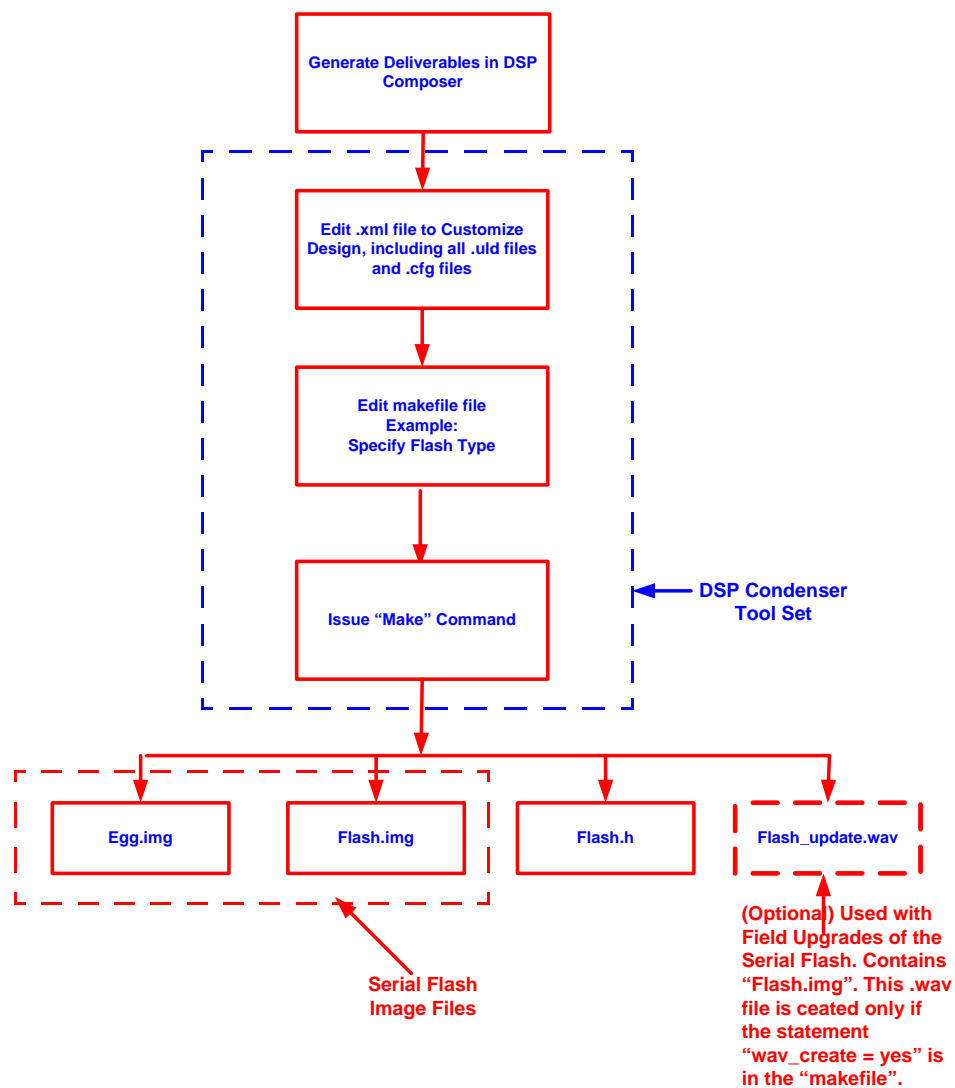


Figure 8-16. Flash Image Creation Process Flow

8.7.2.1 Using the DSP Condenser Wizard to Create a Serial Flash Image

To use the DSP Condenser wizard, follow these steps:

1. Install the DSPCondenser.exe and the required firmware modules into the cs4953x_eval_kit install directory (e.g., C:\CirrusDSP)
2. Open DSP Condenser wizard from the Start Menu→CirrusDSP
3. To create a new project, Click on File→New project.
4. Type a suitable project name. Choose the appropriate template. The templates basically provide a starting point with example configurations for the project. The following examples are shipped with the evaluation kit:
 - *Sample* template corresponds to a project with unlicensed technologies (suitable for CS4953x4, CS4970x4)
 - *Sample_licensed_legacy* template corresponds to a project with legacy licensed firmware modules (suitable for CS4953x4)
 - *Sample_licensed* template corresponds to a project with legacy and HD licensed firmware modules (suitable for CS4970x4)

This action creates a project with the provided project name in the following location:

C:\CirrusDSP\DSPCondenser\projects\<<project_name>

5. Click OK and the main screen of the new project window in DSP Condenser should appear as shown in [Figure 8-17](#). Ensure that the window that opens is maximized to show all the fields.

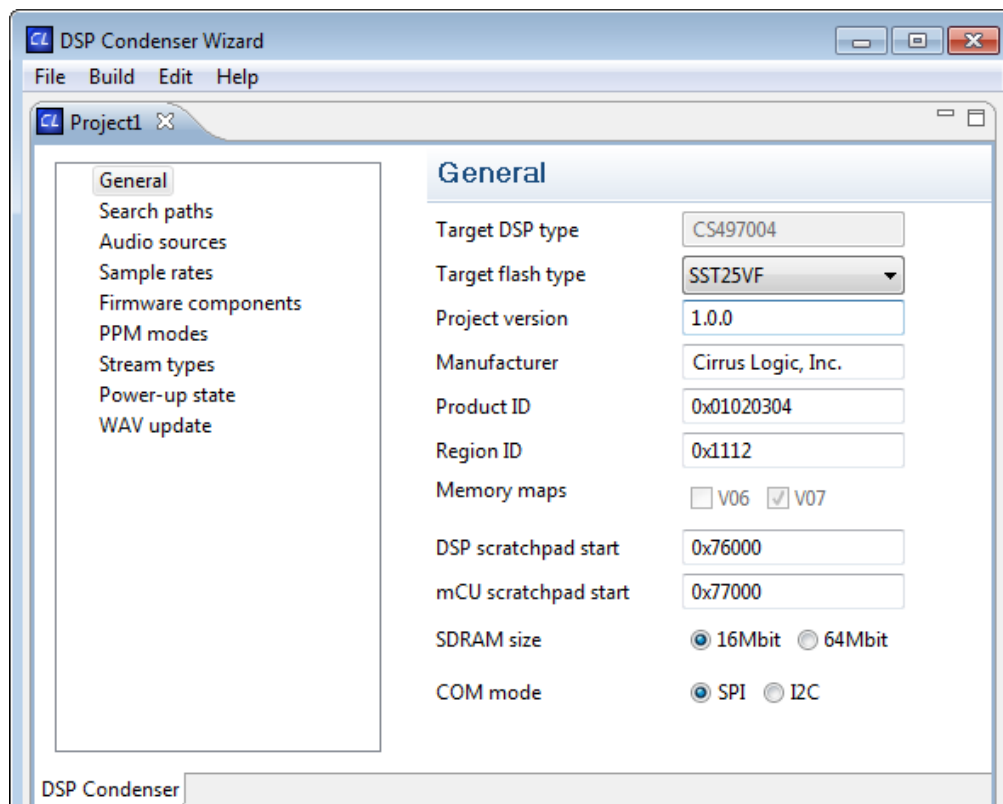


Figure 8-17. New Project Window in DSP Condenser

6. At this point, the user can choose to build a project in one of two ways:
 - A. Using one of the templates described in Step 4 without change
 - b. Modifying the selected template as described in [Section 7.4.2.1](#).
7. Click on Build→Create Flash Image
8. The build log window will open as shown in [Figure 8-18](#) displaying all the files (*.uld, *.cfg etc) listed in the project (underlying xml file) and also displaying the results of the build process, showing whether the image creation process was successful or not.. If there is a failure, scrolling through this log will give the error messages , The log will mention any missing uld or .cfg files.

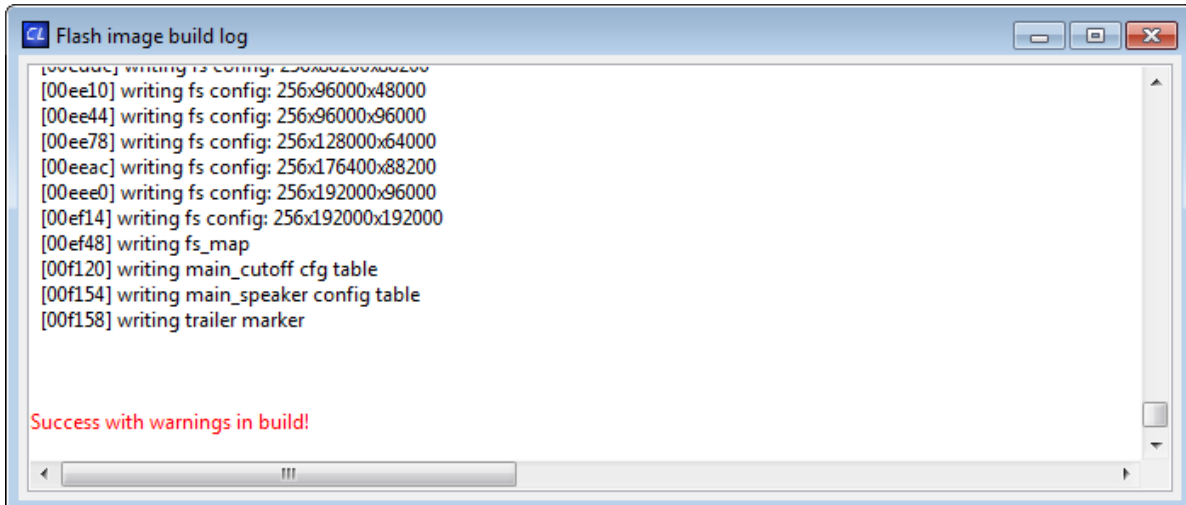


Figure 8-18. DSP Condenser Flash Image Build Log

9. Click on Build->Explore build outputs to use Windows Explorer to open the directory where the .img files, flash.h, and other project files are stored on the user's PC.
10. To program serial flash, press Build > Program Flash on Board.
11. Power-on the board with the USB cable connected between the PC and the board.
12. Double click on the Cirrus Device Manger (CDM) icon on the bottom right of your desktop and right click on the appropriate board detected and listed in CDM as shown in [Figure 8-19](#).

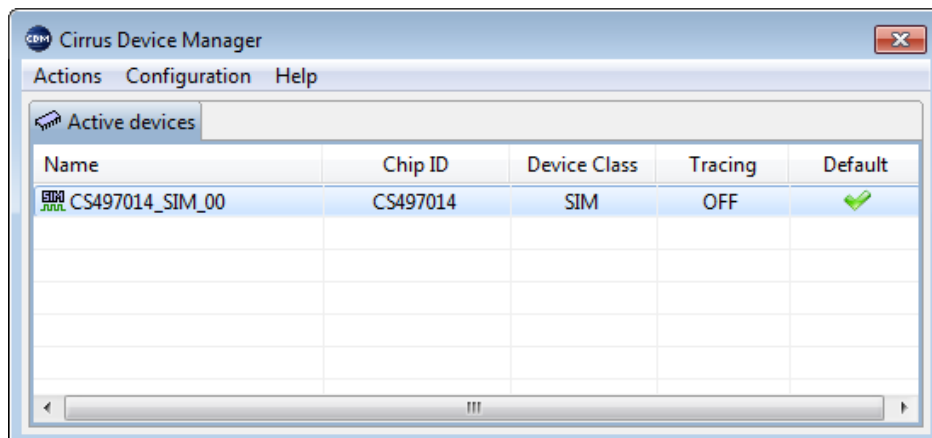


Figure 8-19. CDM Window

13. To program the serial Flash connected to the DSP, the egg.img, flash.img may be copied to:

C:\Cirrus DSP\CS4953x4\apps\`<board_name>`\sd\flash

Note: `<board_name>` can be "crd49500" or "ck49x_4953x4"

8.7.2.2 Making Snapshots of a Change

Perform the following steps to create several snapshots of changes in DSP Composer.

1. Make sure that the CS49x Board USB cable is plugged in.
2. Press the green Go button in DSP Composer. The Check and Compile window displays and closes, and then the Deploying Project dialog box shows progress and closes when deployment is complete.
3. Select the Compile Results tab at the bottom of DSP Composer.
4. Scroll down to see the results. You can see the message "Project compiled successfully" at the bottom of the display.
5. On the DSP Composer Menu, select Tools > Snapshots. The Snapshots Manager Dialog Box displays.
6. In DSP Composer, double-click the APP block in the DSP Composer project. The APP Panel displays.
7. Select the Audio Manager tab. On the APP Panel, note that the Gain, the left-most control, is at +0 dB.
8. Press the New button on the Snapshot Dialog box. The New Snapshot Name dialog box appears.
9. Name the snapshot `audio_mgr_0db` and click OK.
10. Go back to the APP control panel and change the Gain to +10 dB. You should hear the volume increase +10 dB.
11. On the Snapshot Manager Dialog Box, press the New button.
12. In the New Snapshot Name window, name the snapshot `audio_mgr_plus10db`.
13. Repeat steps 10–12 for –10 dB.
14. After you are finished creating snapshots, close the Snapshot window.
15. On the DSP Composer menu, select Tools > Generate Deliverables.
16. Save the Deliverables to the C:\CirrusDSP\DSPCondenser\deliverables directory.
17. Save the DSP Composer project.
18. Exit DSP Composer.

8.7.2.3 Integrating the Snapshots into the Flash Image

The names you give to your modes default to the snapshot name, but the mode can be renamed. The mode names are used in the output of the Condenser project as C language `#define` statements as an aid for the microprogrammer, allowing semantically meaningful names to be used in the microcode used to control the DSP through the DSP Manager API.

Perform the following steps to integrate snapshots into the flash image as modes:

1. In DSP Condenser Wizard, use File > Open Project to open the CS497004_sample_licensed.7z project.
2. Select the Prekick module modes > Audio Manager page.

3. On the right, select CS497004_audio_manager_change as the Composer project.
4. Select audio_mgr_0db as mode 1, audio_mgr_minus10db as mode 2, and audio_mgr_plus10db as mode 3.
5. Save the project
6. On the DSP Condenser Wizard menu, select Build > Create Flash Image.
Make sure that the flash image builds successfully. Scroll to the left to see the "Success" message. Close the log window after the flash image builds.
7. On the DSP Condenser Wizard menu, select Build > Program Flash on Board. The Program Flash log window appears.
Make sure that "Success" is the last word in the log. Scroll to the left to see the "Success" message.
8. Select Build > Run Runtime GUI Menu (current project).

8.7.2.4 Exploring the Build Outputs

After the flash image is created, you can use the DSP Condenser Wizard Build > Explore build outputs command to see the results of the build. The following files are in the outputs folder:

- **egg.img**—The Sstartup DSP code that is programmed into flash at location zero.
- **flash.img**—The main flash image that contains all other DSP firmware components and configuration files.
- **flash_update.wav**—The flash.img file in .wav format, if you specified to create a .wav update file.
- **flash.h**—A C .h file that contains information about the flash.img file which must be used by the microcontroller code as it uses the high-level DSP Manager API. Specifically, it contains a mapping of symbolic names for the various firmware components and modes defined in the condenser project to integer values used in host-to-DSP communication.

The other files in the outputs folder are intermediate output files that are not necessary for actual deployment or debugging.

8.7.3 Field Upgrade of Flash Image

Cirrus Logic recommends that an upgrade CD containing the .WAV file, which currently includes the contents of flash.img only, be used to implement field upgrades of the Flash Image. The creation of the upgrade CD is described in [Section 8.7.1.2](#). The .WAV file may contain up to five images. This format allows for mass production of a single update CD that can be used to upgrade more than one product. Alternatively, the CD may contain different tracks for different products, and then allow the microcontroller to determine which track to play. Refer to the application note, AN288MPF, *Serial Flash Programming Module* for more details about the field upgrade process.

8.7.3.1 Steps for Carrying Out a Field Upgrade of Flash Image

1. Program `egg.img` if the Serial Flash is empty. See [Section 8.7.2](#) and [Section 8.7.3](#).

Note: The file, `program_condenser_flash.bat`, can be edited to program `egg.img` only.

2. Boot the DSP in Master Boot mode.
3. Read boot messages messages/autodetect messages as described in [Section 8.8](#).

If flash.img has not been programmed the boot and status messages read:

```
# Boot messages
word 0: ef000020
word 1: 80000001
```


- ```
DSP Flash status messages
word 2: ef000020
word 3: 00008002
```
4. Issue the commands directly to put the DSP in update mode, described in the DSP\_CFG\_AUDIO\_SRC variable in [Table 7-4, "DSP\\_CFG\\_xxx Firmware Configuration Registers" on page 7-7](#). This step is not necessary if flash.img is not already programmed.
- ```
Ef000000000000100
Ef000006000000007
Ef000013000000100
Ef000000000000001
read status messages
word 0: ef000020
word 1: 00008002
```
5. Play the flash_update.wav file using a bit-exact sound card or a player to the S/PDIF input (SPDIF_RX0) of the CRD49700 evaluation board or the S/PDIF input port of target system.
6. Read the unsolicited mesg
- ```
c2000003 # SF status
40000000 # Found sync packets, Serial Flash update process
 # starts

c2000006
xxxxxxx #current address/packet
```
7. After update is over, read the unsolicited messages:
- ```
c2000003          # SF status
80000000          # SF update process finish
c2000006
xxxxxxx          #current address/packet
```

Refer to the application note, AN288MPF for detailed information about the messages in steps 6 and 7

8.8 DSP Response after Master Boot.

The DSP response after a successful Master Boot is shown in [Figure 8-20](#).

```
FPGA Configuration 1
Setup SPDIF Receiver
Setup CODEC
rx1
Master Boot complete!!

DSP Response:

word 0: ef000020
word 1: 00000001
DSP Response:

word 0: 81000000
word 1: 00000020
```

Figure 8-20. DSP Response after Successful Master Boot

The messages starting with “EF” correspond to the DSP Manager API in [Section 7.4 “CS4953x4/CS4970x4 DSP Manager API Description” on page 7-6](#), and the messages in [Figure 8-20](#) beginning with

'81' correspond to the autodetection messages/response. For a description of autodetect messages, see [Section 7.3.7 "DSP_AUTODETECT_MSG"](#) on page 7-4. [Figure 8-21](#) shows the autodetect message for silence.

8.9 Host Activity

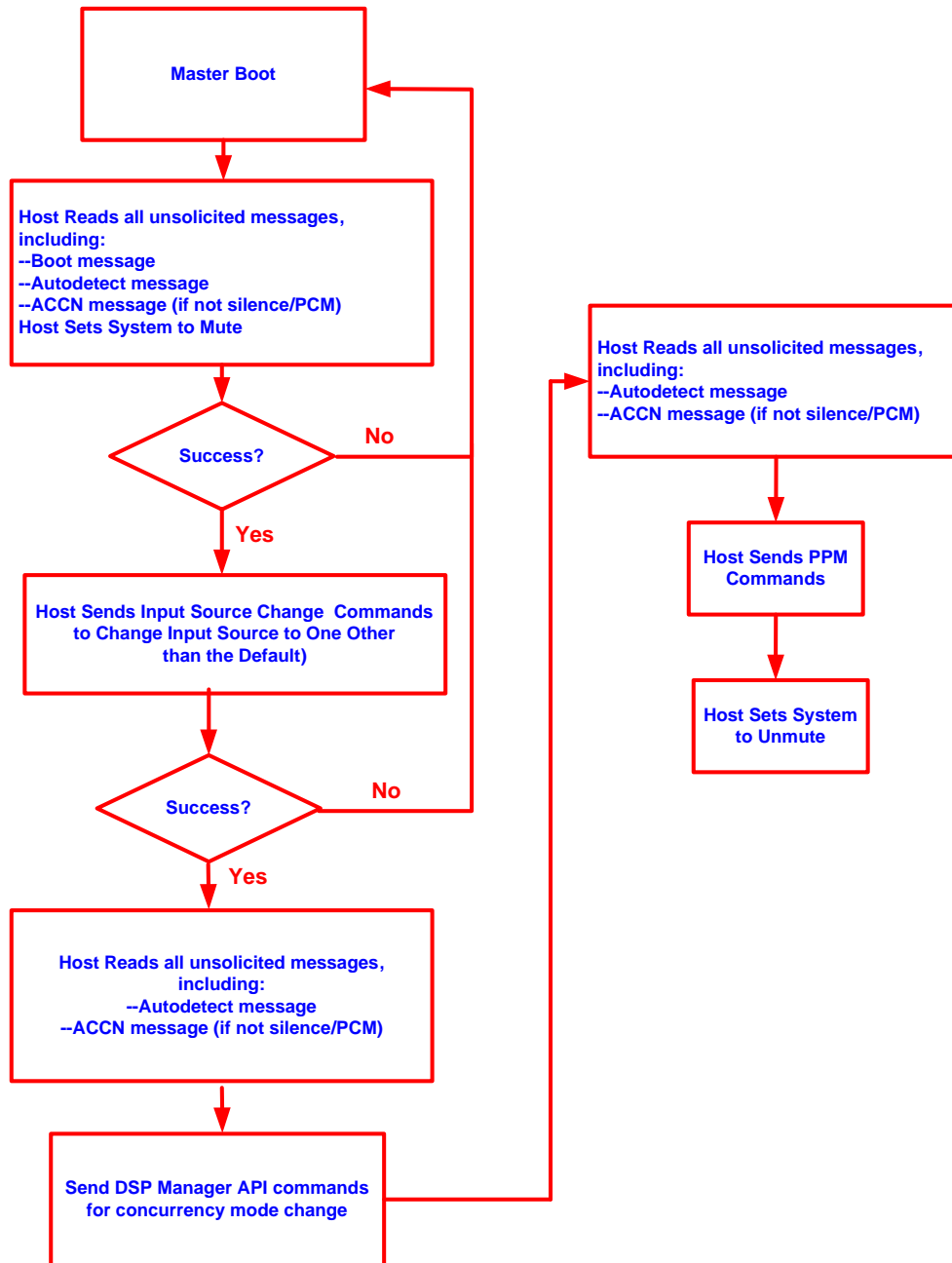
The host carries out the following activities after a successful boot.

1. Flush the DSP message buffer by reading all outstanding messages.
2. Retrieve status information from the DSP using the DSP manager read commands set out in [Table 7-5](#).
3. Send DSP Manager write commands to change concurrency mode if required. See examples of write commands in [Section 7.4.2.1 "Using DSP Condenser to Change/Load Firmware Modules"](#) on page 7-10 and in [Appendix C, "Loading/Unloading Firmware Modules"](#).

See the process flow for changing *concurrency*¹ modes in [Figure 8-21](#).

1. Concurrency mode is the combination of firmware modules active at any one point in time.

See Note below this flow chart.



Note: Current firmware does not support changing input source from HDMI/Multichannel source to SPDIF source using DSP Manager API commands. Cirrus Logic recommends using SPDIF as the default input source in the flash_image.xml file and toggling the DSP Reset to switch to SPDIF source from HDMI/Multichannel source.

Figure 8-21. Changing Concurrency Modes

Note: To perform Host Writes / Reads, follow the steps in [Figure 2-3 on page 2-5](#) (Writes) and [Figure 2-4 on page 2-6](#) (Reads).

§§

Chapter 9

Using Runtime Condenser

9.1 DSP Condenser Runtime Application

9.1.1 Usage

9.1.1.1 Standard Launch

The DSP Condenser Runtime is launched from the DSP Condenser application. The DSP Condenser application provides the .uld information to the runtime application. Use the following procedure.

1. Run DSP Condenser.
2. Create a flash DSP image using the DSP Condenser application as shown in [Figure 9-1](#).

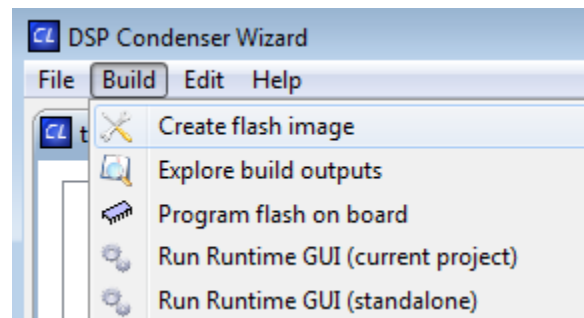


Figure 9-1. DSP Condenser Wizard Menu Items

3. Wait for the image to finish building as shown in [Figure 9-2](#).

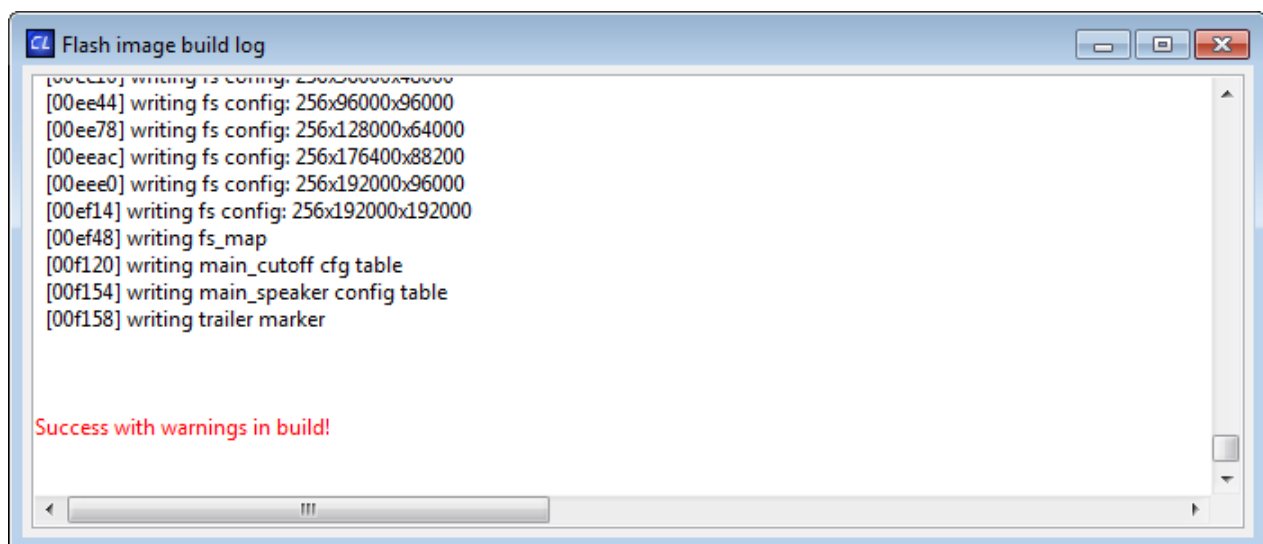


Figure 9-2. Flash Image Build Log

4. Flash the image using “Program flash on board” as shown in [Figure 9-3](#).

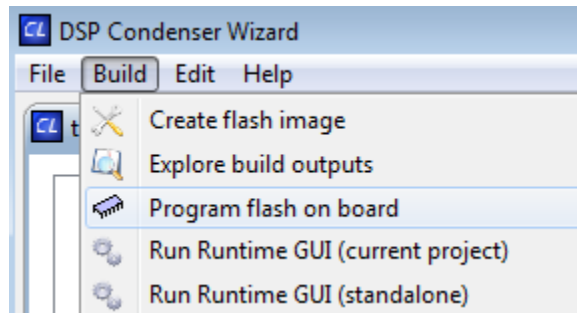


Figure 9-3. Program Flash on Board

5. Select "Run Runtime GUI (current project)" as shown in Figure 9-4.

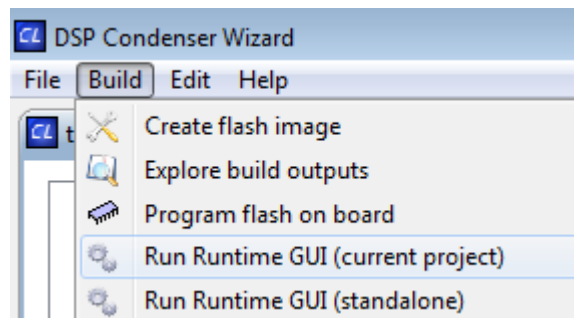


Figure 9-4. Run Runtime GUI

6. Select the DSP Manager API Input Source by changing the Source Combo Box as shown in Figure 9-5.

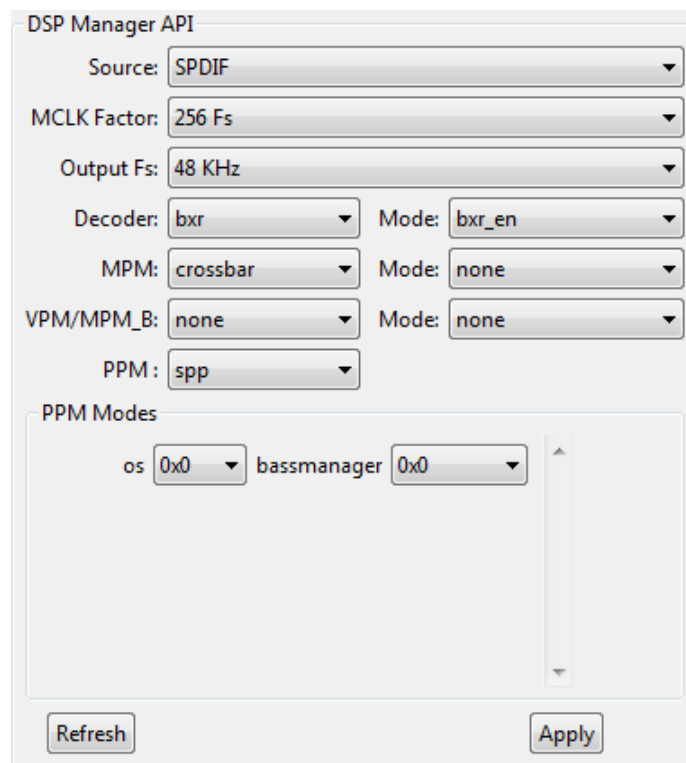


Figure 9-5. DSP API Manager Input Source

7. Select the Board Input Source item from the Source Status group. This configuration should match what was set in Step 6.
8. Press “Connect to Board”.
9. Press “Refresh” in the DSP Manager API Group to see the current DSP Manager API State.
10. Change the appropriate combo boxes in the DSP Manager API Group. These controls turn red to indicate the values have changed.
11. Press “Apply”. Any values that have changed are sent to the DSP.

9.2 Runtime GUI Current Project

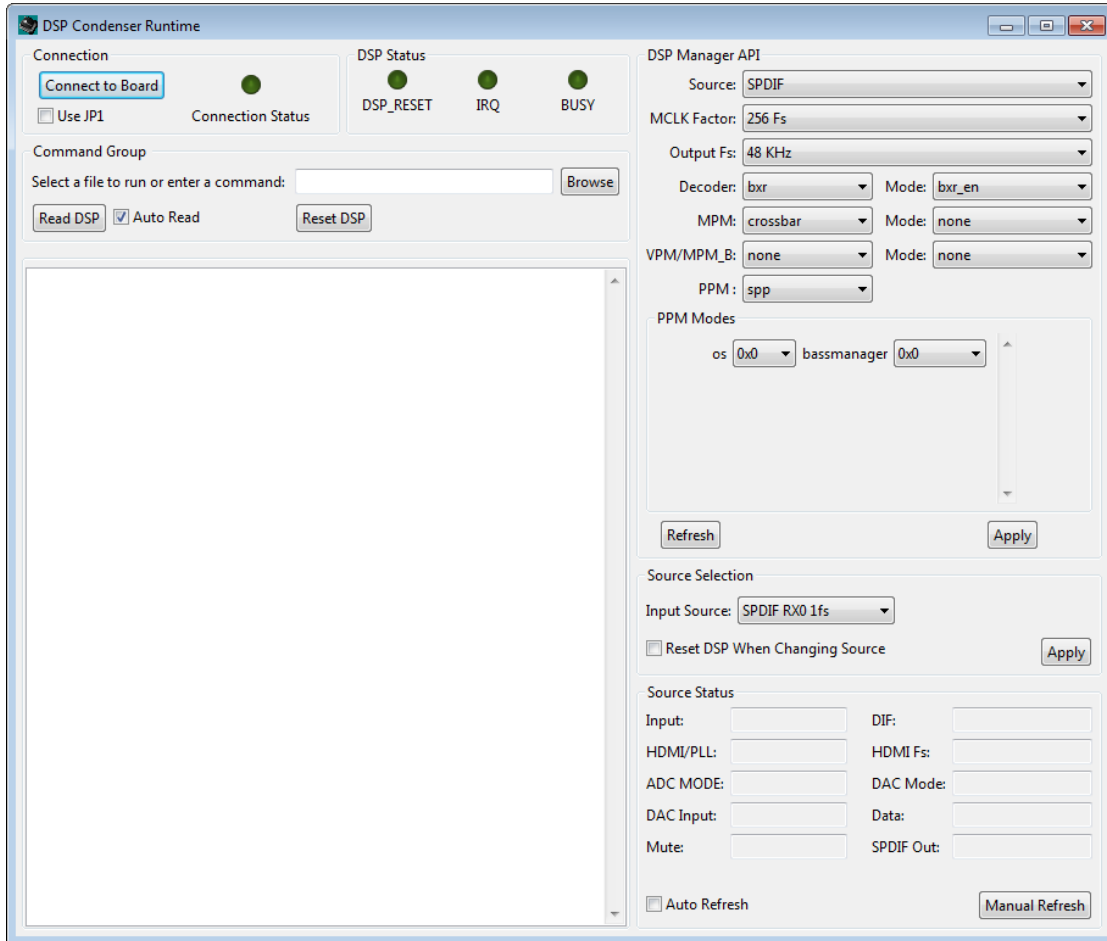


Figure 9-6. DSP Condenser Runtime with DSP Image XML Provided

9.2.1 Connection Group

The group seen in [Figure 9-7](#) controls the connection to the CDB49x board.

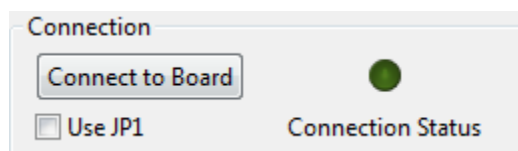


Figure 9-7. Connection Group

9.2.1.1 Connection to Board Button

This connects/disconnects to a CDB49x board through the USB port.

9.2.1.2 Connection Status Led

This shows if the DSP Condenser Runtime application is attached to a CDB49x board.

9.2.1.3 Use JP1 Checkbox1

Check the JP1 checkbox if a DSP board is connected to the JP1 header on the customer-designed board.

9.2.2 Command Group

The Command Group seen in [Figure 9-8](#) can issue commands that are not related to the DSP Manager API.

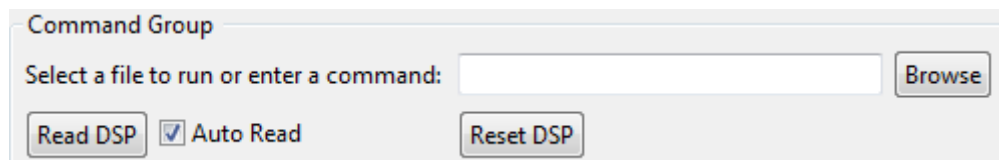


Figure 9-8. Command Group

9.2.2.1 Read DSP Button

When the “Read DSP” button is pressed (if there is an interrupt request), the runtime application issues an SCP read command to the DSP. The results of the DSP read are printed out in red to the “Log” window.

9.2.2.2 Reset DSP Button

When the “Reset DSP” button is pressed, it brings the nDSP_RESET line low, waits for 100 ms, and brings the nDSP_RESET line high.

9.2.2.3 Auto Read Checkbox

When the “Auto Read” checkbox is enabled (if there is an interrupt request), the runtime application issues a read command to the DSP. This occurs approximately every 500 ms. The results of the read are printed out in red to the “Log” window.

9.2.2.4 Command Field

The “Command” field allows the user to issue commands. The following commands are possible:

- Hexadecimal commands sent to the DSP
- Filenames of configuration files (*.cfg)
- Comments that start with a “#” sign

Note: Users are able to issue commands to the DSP using a console window.

9.2.2.4.1 Hexadecimal Command

Hexadecimal commands allow the user to transmit an 8-digit hexadecimal number. This number must be in the form of “XXXXXXXX”, where X is a hexadecimal digit. No hexadecimal identifier (“0x” or “h”) should be used.

9.2.2.4.2 Configuration File Command

The configuration file command is the full path name of the configuration file sent down. The configuration file must have a *.cfg extension and be a valid Cirrus DSP Configuration file. After the enter key is pressed, the commands in the configuration file are sent to the DSP and shown in black in the “Log” window.

9.2.2.4.3 Comments

A user can issue comments to the log by typing “#” as the first character in the “Command” field.

9.2.2.5 Browse Button

The “Browse” button allows the user to select a *.cfg file to be downloaded to the DSP. After a configuration file is selected, the full path and filename appear in the “Command Text” field. The user must hit enter in the “Command Text” field for the Cirrus DSP Configuration file to be loaded into the DSP.

9.2.3 DSP Status Group

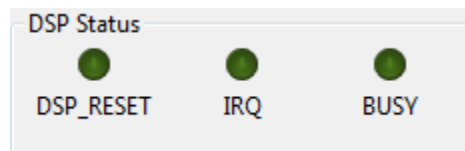


Figure 9-9. DSP Status Group

The LEDs in the DSP Status Group correspond to the signals on the DSP board.

Note: The signals on the DSP board are active low.

9.2.3.1 DSP Reset LED

When lit, the DSP is being held in reset.

9.2.3.2 Interrupt Request LED

When lit, the DSP has data to be read.

9.2.3.3 Busy LED

When lit, the DSP is busy and a DSP command cannot be issued.

9.2.4 Log Window

9.2.4.1 Color Coding

The following can occur in the log windows:

- Comment
 - # Comment
- MCU register command
 - brd_cfg -w 0 2801
- Board configuration command

- brd_cfg -w board 00 02 24 02
- DSP write
 - tx: 80000001
- DSP read
 - rx: ef000020

9.2.4.2 Menu Items

By right-clicking on the “Log” window, a menu is shown with the following options:

- Save As–Save the log as a text file for later viewing.
- Save As *.cfg–Save as a DSP configuration file. The configuration file includes all the writes since the last reset.
- Copy–Copies the selection to the clipboard.
- Select All–Select all of the text in the log file.
- Clear–Clears the “Log” window text.

9.2.5 Source Selection Group

The Source Selection Group allows the user to select which source is active on the board. When the connection is in JP1 mode, this group is disabled.

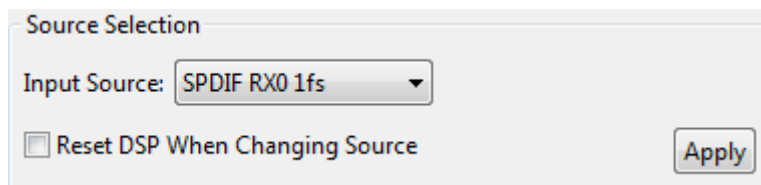


Figure 9-10. DSP Condenser Runtime Source Selection Group

9.2.5.1 Input Source Combo Box

The following input source exists:

Table 9-1. Translation from Input Sources to Board Configuration Values

Input Source	Board Configuration Value
S/PDIF RX0 1 fs	00 02 24 02 00
S/PDIF RX1 1 fs	02 02 24 02 00
HDMI Legacy	06 03 24 02 00
HDMI PCM 2 fs	06 42 24 12 00
HDMI PCM 4 fs	06 81 24 02 00
ADC Stereo	08 4F 24 52 00

9.2.5.2 Apply Button

The “Apply” button configures the board so the input source selected in [Table 9-1](#) is used.

9.2.5.3 Reset DSP When Changing Source Checkbox

When the “Reset DSP When Changing Source” box is checked, the DSP is reset after the board is programmed to handle a new source.

9.2.6 Source Status Group

This group shows the current board configuration. When the connection is in JP1 mode, this group is disabled.

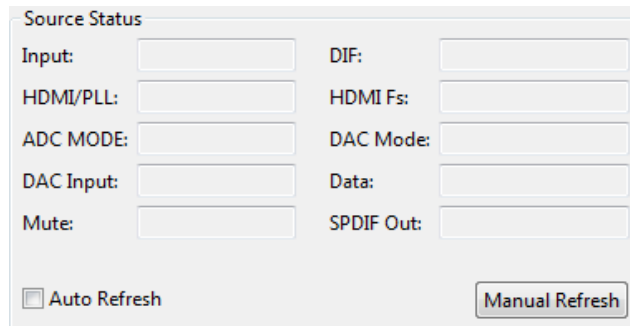


Figure 9-11. DSP Condenser Runtime Source Status Group

9.2.6.1 Manual Refresh Button

The “Manual Refresh” button reads the DSP board’s source configuration and updates the “Source Status” text fields.

9.2.6.2 Auto Refresh Button

When the “Auto Refresh” checkbox is set, the board configuration is read every 500 ms, and the “Source Status” fields are updated.

9.2.6.3 Auto Read Checkbox

When connected, the “Board Configuration” value is read and displayed in the “Source Status” text fields.

9.2.6.4 Source Status Text Fields

See the USB Micro Master documentation, Board Configuration section for more information on the “Source Status” text fields. When the connection is in JP1 mode, this group is disabled. The “Source Status Group” has the following text fields:

- Input—This reflects what is chosen in the source list as shown in [Table 9-1](#).
- HDMI/PLL—In an HDMI source mode, the options are “HDMI Sync” and HDMI noSync”. In an S/PDIF source mode, the options are “PLL Lock” and “PLL Unlock”. If in neither HDMI or S/PDIF mode, the value of the field is “N/A”. If the board is not configured, the field shows “Unconfigured”.
- Digital Data Input Format (DIF)—This indicates the input format to the DSP of the serial audio data. The possible values are “I2S”, “LJ” (left justified), “Online”, and “Unconfigured”.
- ADC Mode—One line for mic ADC use/DIF mode (for normal stereo analog in). The possible values are “SSM (1Fs)”, “DSM (2Fs)”, “QSM (4Fs)”, and “Unconfigured”.
- DAC Mode—The possible values are “Single Speed Mode (SSM)”, “Double Speed Mode (DSM)”, “Quad Speed Mode (QSM)”, “Reserved”, and “Unconfigured”.
- DAC Input—The input format for the DAC. The possible values are “Left Justified 24 bit (LJ 24 Bit)”, “I2S 24 Bit”, “Reserved”, and “Unconfigured”.
- Data—PCM/Compressed (in both S/PDIF and HDMI cases) or unlocked. The possible values are

“Compressed”, “Uncompressed”, and “Unconfigured”.

- Mute–Status of mute in DAC. The possible values are “On”, “Off”, and “Unconfigured”.
- HDMI Fs–This determines the DAI LRCLK frequency based on N, CTS values and/or the audio sampling frequency register. The possible values while in an HDMI Source mode are “Off”, “Video Passthru”, “I2S 1Fs”, “I2S 2Fs”, “I2S 4Fs”, and “I2S SPDIF”. If the source is not in an HDMI mode, the value is “N/A”. If the board has not been configured, the field shows “Unconfigured”.
- SPDIF OUT–The possible values are “DSP DAO2”, “RX1”, “HDMI”, “RX0”, “DSP DAO3 D3”, “DSP DAO3 D5”, and “Unconfigured”.

9.2.7 DSP Manager API Group

The DSP Manager API Group shows the current settings in the DSP Manager API. See the CS4953x4/CS4970x4 System Designer's Guide, “Overview of Common Firmware Modules” for additional information.

After a connection occurs (if a user changes any values), the values show up in red. Only the DSP Manager API Registers whose corresponding “Combo” box have changes are updated when the “Apply” button is pressed.

If the DSP Condenser starts with the DSP image information, the Decoder, Decoder Mode, MPM, MPM Mode, VPM, VPM Mode, PPM, and the PPM modes are populated with the names of the .ulds as shown in [Figure 9-12](#).

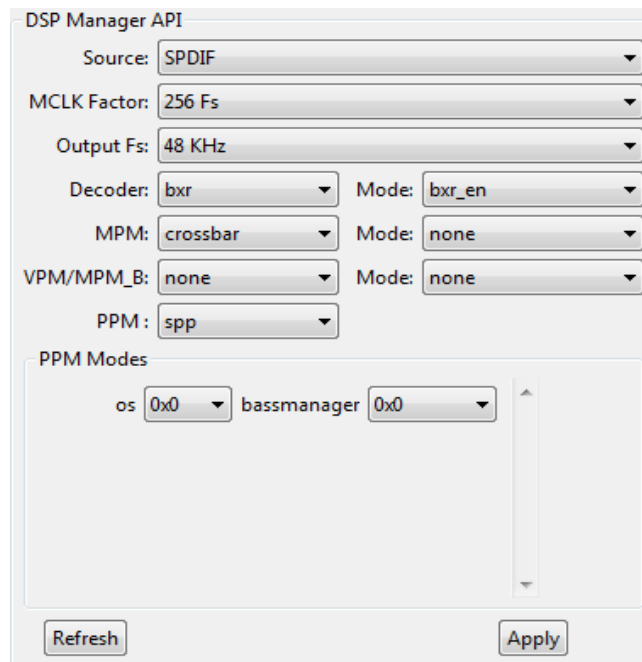


Figure 9-12. DSP Manager API Group with Image Information

If the DSP Condenser is started without image information (that is, “Runtime GUI Standalone” mode), these combo boxes are filled with hexadecimal values as show in [Figure 9-13](#).

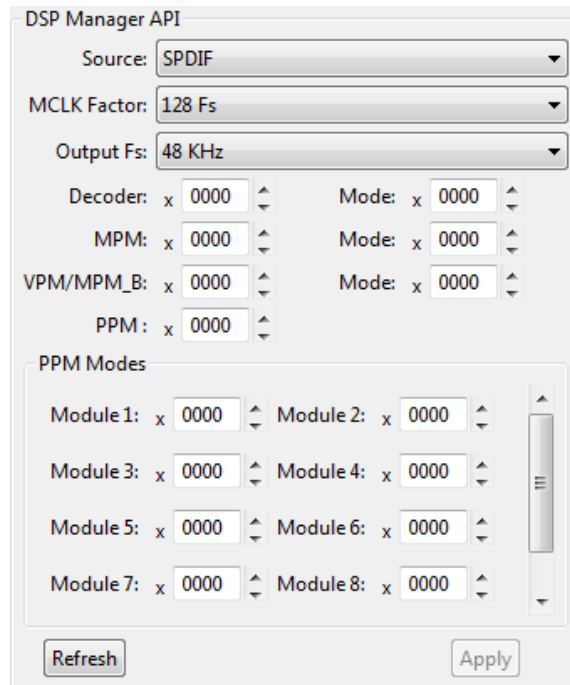


Figure 9-13. DSP Manager API Group without Image Information

9.2.7.1 Source

The source shows the value of the DSP_CFG_AUDIO_SRC Register. The possible options are determined by the flash image information. The values can be “SPDIF”, “SPDIF_NO_AUTOSWITCH”, “HDMI”, “HDMI_NO_AUTOSWITCH”, and “MULTICHANNEL”.

9.2.7.2 MCLK Factor

The MCLK Factor shows the value of the DSP_CFG_MCLK_FACTOR Register. The possible options are determined by the flash image information. Some of the possible values are “128Fs”, “256 Fs”, and “512 Fs”.

9.2.7.3 Output Fs

The Output Fs reflects the value of the DSP_CFG_OUTPUT_FS Register. The possible options are: "48 KHz", "44.1 KHz", "32 KHz", "96 KHz", "88.2 KHz", "64 KHz", "24 KHz", "22.05 KHz", "16 KHz", "192 KHz", "176.4 KHz", and "128 KHz".

9.2.7.4 Decoder and Decoder Mode

The Decoder and Decoder Mode reflect the value of the DSP_CFG_DECODER Register.

9.2.7.5 MPM and MPM Mode

The MPM and MPM Mode reflect the value of the DSP_CFG_MATRIX Register.

9.2.7.6 VPM and VPM Mode

The VPM and VPM Mode reflect the value of the DSP_CFG_VIRTUALIZER Register.

9.2.7.7 PPM

The PPM reflects the value of the DSP_CFG_PPM Register.

9.2.7.8 PPM Mode List

The PPM Mode Lists reflects the value of the DSP_PPM_MODE1-5 Registers.

9.2.7.9 Refresh Button

After the user connects to the board, the user is able to push the “Refresh” button to refresh all of the values in the DSP Manager API group. All combo boxes/spinner controls in the DSP Manager API group whose value is red is reset to black.

9.2.7.10 Apply Button

After the user connects to the board, the user is able to change the values in the DSP Manager API. Any values that have changed show up in red. When the user his the “Apply” button, the combo values in red are programmed down into the board. These commands show up in the “Log” window. If no changes are made and the “Apply” button is pressed, no values are written to the board.

When writing changes to the DSP Manager API, the DSP_BOOT register is written with a 0x00000100 before any changes are made to the register. After all changes are made, the DSP_BOOT register is programmed with a 0x00000001 value.

Appendix A

FAQ

A.1 Introduction

Appendix A contains design tips in the familiar Question and Answer format. The issues discussed are workarounds and techniques perfected by Cirrus Logic internal designers and support staff to smooth the path of DSP system designers.

A.2 List of Questions and Answers

Q 1. I would like to create a custom flash image. How does one edit the Flash_image.xml file?

A 1. View the sample flash_image.xml and follow the instructions in the sample file.

After the necessary tools and deliverables have been installed on the user's system, do the following:

1. Extract the sample_cirrus.zip file to "C:\CirrusDSP\DSPCCondenser\projects\sample_cirrus" folder.
2. Verify that "C:\CirrusDSP\DSPCCondenser\projects\sample_cirrus\flash_image.xml" folder exists.

For users that are not familiar with XML coding, here are some basic syntax protocols used in the sample flash_image.xml.

- <source> </source> are start/end markers for an input source.
- </concurrency_mode> end marker for a particular concurrency mode which defines all modules (and their configurations) loaded when a particular stream type is identified.
- <stream type< </stream type> are start/end markers for a stream type to associate a particular stream type with the corresponding decoder
- <concurrency_modes> </concurrency_modes> start/end marker to the section that lists all concurrency modes.
- <sample_rate_combinations> </sample_rate_combinations> This section configures the DSP output xclocks for all possible sample rates. Do **NOT** edit this section. Please contact Cirrus Logic for any issues regarding this section.
- <power_up_state> </power_up_state> are the start/end markers of the section that defines the default DSP state immediately after master boot.
- </mode> mode end/separation marker. Use this marker to separate types of modes.
- See [Section 7.3.6 "Unsolicited Messages from DSP to the Host Microcontroller" on page 7-4](#) and [Section 7.4.2 "DSP_CFG_XXX Registers" on page 7-7](#)
- </uld> current uld end marker or uld separation marker in the uld list.
- </ulds> end of uld list

See [Section 7.3.6 "Unsolicited Messages from DSP to the Host Microcontroller" on page 7-4](#) and [Section 7.4.2 "DSP_CFG_XXX Registers" on page 7-7](#)

See Step 7 in [Section 8.7.2 "Creating a Serial Flash Image Manually" on page 8-22](#) for the circumstances when the .xml files are edited.

Q 2.How do I create deliverables using DSP Composer?

A 2.Create DSP Composer Deliverables using the following procedure:

1. Install the latest version of the DSP Condenser. This should contain the 'flasher' utility, sample DSP Composer projects (.cpa) files and a sample .xml file (flash_image.xml) . The flash_image.xml is an input to the flasher utility.
2. Install the latest version of the firmware modules. For example, ac3.exe, aac.exe,dts.exe,os.exe,pl2.exe ..etc.
3. Open the applicable sample .cpa files (installed by dspcondenser.exe in:

C:\CirrusDSP\DSPCondenser\projects\sample_cirrus.zip

The sample files included in the sample_cirrus zip file at the time publication are:

- decoder_aac.cpa
 - decoder_ac3.cpa
 - decoder_dts9624.cpa
 - decoder_dts.cpa
 - decoder_pcm.cpa
4. DSP Composer is part of the evaluation kit installation. Make sure the latest version of the evaluation kit is installed in 'C:\CirrusDSP' folder to avoid any version mismatch issues. If there the version installed on your system is out-of-date, a screen pops up with a request to update the needed files. See the out-of-date evaluation kit notice in [Figure A-1](#).

Note: DSP Composer will be enhanced in the future to automatically recognize when an earlier version of DSP Composer exists and then replace the older version of the latest version of DSP Composer..

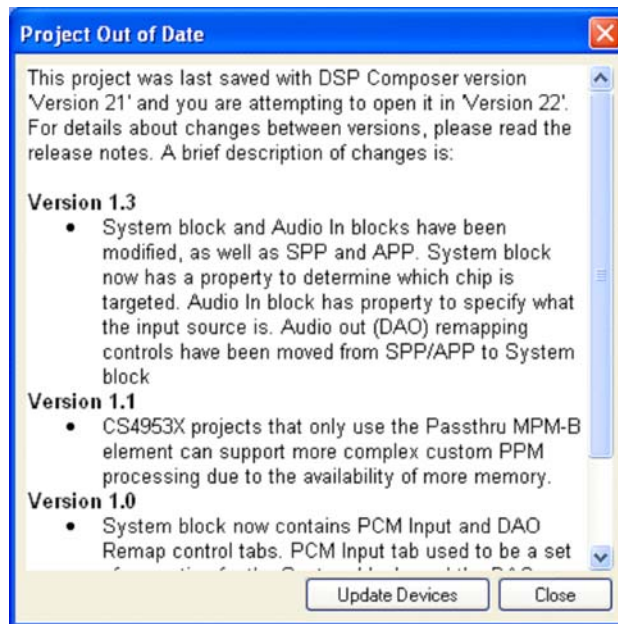


Figure A-1. Out-of-Date DSP Composer Notice

Click on 'Update Devices' to install the latest version of DSP Composer.

-
5. Click on '*Generate Deliverables*' in the '*Tools*' menu for each of the .cpa files opened. All the generated deliverables must be saved under one parent directory. For example:
C:\CirrusDSP\DSPCondenser\projects\sample_cirrus\deliverables.

Refer to the DSP Composer manual for more details. Also, refer to [Section 8.6.1.1, "Best Practices"](#) on [page 8-17](#) on suggested directory structure.

§§

Appendix B

Optional Features

B.1 Introduction

Appendix B describes optional features that are available to Cirrus Logic customers on request by contacting your Cirrus Logic FAE or representative. This feature is not currently supported by the CS4953x4/CS4970x4 product firmware.

B.2 Communication Using the Parallel Control Port

The CS4953x4/CS4970x4 is equipped with an 8-bit Parallel Control Port that can be used for host communication, providing faster control throughput for the system. The Parallel Control Port is capable of Intel®, Motorola®, and Multiplexed Intel communication modes.

§§

Appendix C

Loading/Unloading Firmware Modules

C.1 Introduction

Appendix C describes how to load and unload various firmware modules in DSP Condenser. This occurs the host changes concurrency mode(s). See [Section 7.4.2.1, "Using DSP Condenser to Change/Load Firmware Modules" on page 7-10](#) for more detailed information on loading and unloading firmware modules. Many of the firmware decoder modules mentioned in this chapter require a license from the vendor and cannot be downloaded to the CS497xx device without a license. Contact your Cirrus Logic Field Application Engineer (FAE) for licensing requirements for the third-party firmware mentioned here.

Note: In the following examples, "UCMD" command is used to send command to the CS497xx Evaluation Board. In the customer's system, the host processor does not need to preface the command with "UCMD."

C.1.1 DTS 96/24™ and DTS-ES™

See the Cirrus Logic application note, AN246DB for a complete description of the DTS 96/24 firmware module. and AN246DC for the DTS-ES module.

C.1.1.1 Loading DTS 96/24 Decoder

The host commands necessary to load the DTS 96/24 decoder to output decoded audio streams at Fs 96 kHz are.

```
UCMD Ef000000000000100 # Set configuration lock bit
UCMD Ef000007xxxx00yy # Change decoder mode
UCMD Ef000005000000004 # Change output Fs
UCMD Ef000000000000001 # Change to new configuration
```

X= dts_es decoder mode with DTS96/24™ enabled, yy= dts_es uld id

C.1.1.2 Switching to DTS-ES from DTS 96/24 Decoder

The host commands necessary to switch back to DTS-ES from DTS96/24 are:

```
UCMD Ef000000000000100
UCMD Ef000007000x00yy
UCMD Ef000005000000000
UCMD Ef000000000000001
```

X= dts_es mode (96/24 not enabled), yy= dts_es uld id

C.1.2 DTS-ES Matrix and DTS Neo6™

See the Cirrus Logic application note, AN246DC for a complete description of the DTS-ES firmware module. and AN246MPB for the DTS Neo6 module.

C.1.2.1 Loading DTS-ES Decoder in Matrix Mode

The host commands necessary to load DTS-ES in matrix mode are:

```
UCMD Ef000000000000100
UCMD Ef000007000x00yy
UCMD Ef000008000000zz #Loads DTS Neo6 required to load DTS ES Matrix
UCMD Ef000000000000001
```

X= DTS_ES decoder mode with matrix ON, yy= dts_es uld id, zz = neo 6 uld id

C.1.3 Dolby Digital®

See the Cirrus Logic application note, AN246DA for a complete description of the Dolby Digital firmware module.

C.1.3.1 Loading Dolby Digital Decoder for Stereo Output

The host commands necessary to load Dolby Digital for stereo output are

```
UCMD Ef000000000000100
UCMD Ef000007000x00yy
UCMD Ef00000B0000000z # Change PPM mode (Output mode)
UCMD Ef000000000000001
```

x = DD mode for stereo output (e.g., LoRo).

yy = ac3 decoder uld id

z = ppm mode with output mode L_R

C.1.3.2 Setting Dolby Digital to AC3 DRC Mode

The host commands necessary to load Dolby Digital Dynamic Range Compression (DRC) modes are

```
UCMD Ef000000000000100
UCMD Ef000007000x00yy
UCMD Ef000000000000001
```

x = AC3 DRC mode desired (as per flash_image.xml file), yy= AC3 uld id

C.1.4 Cirrus Logic Signal Generator (SGEN)

See the Cirrus Logic application note, AN246DH for a complete description of the SGEN firmware module.

C.1.4.1 Loading the Cirrus Logic SGEN Module

To load SGEN, follow these steps:

1. Change the input source to either "SPDIF with no autodetect" or "HDMI with no autodetect".
2. Change input source to SPDIF with no autodetect and load SGEN .uld using the following host commands:

```
UCMD Ef000000000000100
UCMD Ef000006000000005 # Selects SPDIF with no autodetect.
UCMD Ef000007000x00yy # Host command that loads SGEN
UCMD Ef000013000000100 # Sets DSP_CFG_MCLK_FACTOR.
```

```
UCMD Ef0000000000000001
```

x= SGEN mode, yy= SGEN uld id

C.1.5 Dolby Digital® Plus

See the Cirrus Logic application note, AN304DA for a complete description of the Dolby Digital Plus firmware module.

C.1.5.1 Loading Dolby Digital Plus for Stereo Output

The host commands necessary to load Dolby Digital Plus for stereo output are

```
UCMD Ef0000000000000100
```

```
UCMD Ef000007000x00yy
```

```
UCMD Ef00000B0000000z
```

```
UCMD Ef0000000000000001
```

x = DDplus mode for stereo output (Eg. LoRo).

yy = DDplus decoder uld id

z = ppm mode with output mode L_R

C.1.5.2 Setting DRC Modes for Dolby Digital Plus

The host commands necessary to set DRC modes for Dolby Digital Plus are.

```
UCMD Ef0000000000000100
```

```
UCMD Ef000007000x00yy
```

```
UCMD Ef0000000000000001
```

x = DDplus DRC mode desired (as per flash_image.xml file), yy= DDplus uld id

C.1.6 Dolby® TrueHD

See the Cirrus Logic application note, AN304DC for a complete description of the Dolby TrueHD firmware module.

C.1.6.1 Loading Dolby TrueHD for Stereo Downmix Output

The host commands necessary to load Dolby TrueHd for stereo downmix output are

```
ucmd ef0000000000000100
```

```
ucmd ef000007000m00nn      # Load True HD decoder stereo mode, if  
exists                      in the image.
```

```
ucmd ef000008000000pp      # Load setup crossbar.
```

```
ucmd ef000009000q00rr      # Load apply crossbar_b with main mixer  
enabled)
```

```
ucmd ef00000b0000000z
```

```
ucmd ef0000000000000001
```

m=TrueHD mode for stereo output, nn= TrueHD decoder uld id

pp= uld id of setup crossbar

q= Apply crossbar mode with main mixer enabled

rr= uld id of Apply Crossbar

z= ppm mode with output_mode L_R

Note: When the downmix option is chosen (also reflected in the output mode), the program chosen from the input stream changes to the one with lower number of channels and the CHANNEL ASSIGNMENT field in the ACCN message reflects this change. The FORMAT INFO field of the ACCN message still displays all the available programs in the stream.

C.1.6.2 Setting DRC Modes for Dolby TrueHD

The host commands necessary to set DRC modes for Dolby TrueHD are:

```
UCMD Ef000000000000100
UCMD Ef000007000x00yy
UCMD Ef000000000000001
```

x = TrueHD DRC mode desired (as per flash_image.xml file), yy= TrueHD uld id

C.1.7 Dolby ProLogic® IIx

See the Cirrus Logic application note, AN246MPM for a complete description of the Dolby ProLogic IIx firmware module.

C.1.7.1 Loading Dolby ProLogic IIx

The host commands necessary to load the Dolby ProLogic IIx legacy mode (pl2x_A) are

```
UCMD Ef000000000000100
UCMD Ef000008000x00yy
UCMD Ef000000000000001
```

x = PL2x_A mode, yy = PL2x_A uld id as per flash.h

PL2x_A can have a mode with crossbar dual zone mixer enabled. This is done by adding the crossbar configuration. along with PL2x_A configuration for a particular mode of PL2x_A

C.1.7.2 Using Dolby ProLogic IIx to Output HD Audio Streams

The host commands necessary to load Dolby ProLogic II x (pl2x_b) are:

```
UCMD Ef000000000000100
UCMD Ef000009000x00yy
UCMD Ef00000b0000000z
UCMD Ef000000000000001
```

x = PL2x_B mode, yy = PL2x_B uld id as per flash.h

z = ppm mode corresponding to desired output mode

In the flash_image.xml file if the first listed module id in the "ppm_modes" section is 0x01 (OS), and the ppm mode corresponding to output mode =0x09 is z= 0xa, then

To enable 7.1 output from a 5.1 HD audio stream input using PL2x_b , output mode should be set as below.

```
UCMD Ef000000000000100
UCMD Ef00000b0000000a
UCMD Ef000000000000001
```

C.1.8 Dolby Virtual Speaker® 2

See the Cirrus Logic application note, AN246MPL for a complete description of the Dolby Virtual Speaker 2 (DVS2) firmware module.

C.1.8.1 Loading Dolby Virtual Speaker 2

The host commands necessary to load Dolby Virtual Speaker 2 are:

```
UCMD Ef000000000000100
UCMD Ef000009000p000q
UCMD Ef000000000000001
```

p = DVS2 mode, q= DVS2 uld id as per flash.h

C.1.8.2 Loading Dolby Virtual Speaker 2 with Dolby ProLogic II

The host commands necessary to load Dolby Virtual Speaker 2 along with Dolby ProLogic II are:

```
UCMD Ef000000000000100
UCMD Ef000008000p000q
UCMD Ef000009000r000s
UCMD Ef000000000000001
```

p = PL2x_A mode, q = PL2x_A uld id as per flash.h
r = DVS2 mode, s = DVS2 uld id as per flash.h

C.1.8.3 Removing Dolby Virtual Speaker 2

DVS 2 can be removed by the following commands:

```
UCMD Ef000000000000100
UCMD Ef000008000p000q
UCMD Ef000009000000000
UCMD Ef000000000000001
```

C.1.9 Dolby Headphone® 2

See the Cirrus Logic application note, AN246MPK for a complete description of the Dolby Headphone 2 firmware module.

C.1.9.1 Loading Dolby Headphone 2

The host commands necessary to load Dolby Headphone 2 are:

```
UCMD Ef000000000000100
UCMD Ef000009000p000q
UCMD Ef000000000000001
```

p = DH2 mode, q= DH2 uld id as per flash.h

C.1.9.2 Loading Dolby Headphone 2 with Dolby ProLogic II

The host commands necessary to load Dolby Headphone 2 along with Dolby ProLogic II are:

```
UCMD Ef000000000000100
UCMD Ef000008000p000q
UCMD Ef000009000r000s
UCMD Ef000000000000001
```

p = PL2x_A mode, q = PL2x_A uld id as per flash.h
r = DH2 mode, s = DH2 uld id as per flash.h

C.1.9.3 Removing Dolby Headphone 2

Then DH2 can be removed by the following commands

```
UCMD Ef000000000000100
UCMD Ef000008000p000q
UCMD Ef000009000000000
UCMD Ef000000000000001
```

C.1.10 DTS-HD™ High Resolution Audio

See the Cirrus Logic application note, AN304DB for a complete description of the DTS-HD High Resolution Audio firmware module.

C.1.10.1 Loading DTS-HD High Resolution Audio for Stereo Downmix Output

Host commands necessary to load DTS-HD High Resolution Audio for Stereo downmix output are:

1. Load Apply crossbar_b

```
UCMD EF000000000000100
UCMD EF000009000X00YY
UCMD EF000000000000001
```

x= apply crossbar mode with main mixer disabled
yy= apply crossbar uld id

2. Set index 0x28 of the DTSHD-HRA API (SYS_SPEAKER_OUT) for stereo out:

```
UCMD 9F00002800000002
```

Note: The DSP would auto-detect, auto-switch DTSHRA 96/24 stream. So, there is no extra commands necessary as in the case of legacy DTS96/24

For 6.1 inputs,

“x” should be the appropriate DTSHD-HRA mode with relevant bits (14:13 or 12) set in the index 0x0000 of the API. The host can set this explicitly instead of having a mode in the image but every time any other mode change is performed, the decoder will go back to default state as defined in the 'concurrency modes' section of the flash_image.xml file.

C.1.11 DTS-HD™ Master Audio

See the Cirrus Logic application note, AN304DD for a complete description of the DTS-HD™ Master Audio firmware module.

C.1.11.1 Loading DTS-HD Master Audio for Stereo Downmix Output

The host commands necessary to load DTS-HD Master Audio for Stereo downmix output are:

1. Set output mode along with loading crossbar_b/apply_crossbar_b

```
UCMD Ef000000000000100
```

```
UCMD Ef000009000x00yy
```

```
UCMD Ef00000B0000000z
```

```
UCMD Ef00000000000001
```

2. Set index 0x27 of the (SYS_OUT SPEAKER) to 0x0a or 0x02, see DTS-HD Master Audio API in AN304DC for details.

```
UCMD A000002700000002
```

x = apply_crossbar_b mode (main mixer enabled for 5.1 and disabled for 7.1)

yy = apply_crossbar_b uld id

z = ppm mode with output mode L_R

Similarly, you could downmix to other speaker configurations.

Note: Note: The DSP would auto-detect, auto-switch DTSHD-MA 96 and 192 streams. Currently, 192KHz multi-channel streams are not supported at an output rate of 192 Khz.

C.1.12 Crossbar (Downmix and Upmix)

See the Cirrus Logic application note, AN246MPC for a complete description of the Crossbar firmware module.

C.1.12.1 Loading Crossbar with Legacy and PCM Modules

The host commands necessary to load crossbar_a (used with legacy and PCM (PCM if logic 7 is not used) are:

```
UCMD Ef000000000000100
```

```
UCMD Ef000008000x00yy
```

```
UCMD Ef00000000000001
```

x= crossbar mode (e.g., mode with dual zone enabled with host coefficients)

yy = crossbar uld id

C.1.12.2 Loading Crossbar for Dual Zone Output with Logic 7 and HD Decoders

The host commands necessary to load crossbar_b are:

```
UCMD Ef000000000000100
```

```
UCMD Ef000009000x00yy
```

```
UCMD Ef00000000000001
```

x= crossbar_b / apply crossbar_b mode (API is same as Crossbar for both crossbar_b and apply crossbar)

Cirrus Logic recommends that `apply_crossbar_b` be used for dual zone output with Logic 7 and HD decoders. Also, use `apply_crossbar_b` for DTSHD-MA main downmix as well as dual zone downmix.

C.1.13 Intelligent Room Calibration 2 (IRC2)

See the Cirrus Logic application note, AN246PPJ for a complete description of the IRC2 firmware module.

C.1.13.1 Configuring the DSP for IRC2

Follow these steps to configure the DSP for IRC2:

1. After `master_boot`, issue these commands:

```
UCMD Ef000000000000100
UCMD Ef00000600000003 # Change source to Multi-channel analog input)
UCMD Ef00001300000200 # Use Ef00001300000100 if your MCLK in
                        # MULTICHANNEL ANALOG INPUT is 12.288 MHz)
UCMD Ef000007000s00tt # No decoder loaded preferred but optional)
UCMD Ef000008000000rr
UCMD Ef000009000x000p
UCMD Ef0000a0000000q
UCMD Ef00000000000001
    p = uld id of IRC
    x = Default Mode of IRC as in the xml file
    q = uld ID of APP
    rr = uld ID of Crossbar
    tt= uld_id of serial_flash_pcm
    s= mode of serial_flash_pcm containing configuration command for
      the serial flash device connected
```

2. Run the IRC tests as per IRC2 appnote, AN246PPJ
3. Read the IQ coefficients one by one.
4. Store the EQ coefficient by writing to the serial flash. Refer to Serial Flash application note, AN288MPF for instructions on how to write to the serial flash.

The coefficients are saved in the sector with starting address defined in index 0x001E of DSP Manager. Use the command , `efc0001E`, to read).

Note: These coefficients are obtained from the xml file. The .xml file will have a PPM mode similar to the mode described in the following example.

Example:

```
<module id="0x55">
  <mode index="1" space="dsp_scratchpad" offset="0"
    sizeinbytes="0x844" />
```

The "dsp scratchpad" address is also specified in the .xml file. The size in bytes needs to be calculated based on the data that is written to the serial flash such as the number of channels, bands, etc., which includes the first word which is the "number of words".

When writing to the serial flash, the first location should have the number of words, calculated based on the number of bands, number of channels, etc

Example:

```
0x74000: 00000002
          : d5000025
          : 00200020
```

This example saves one coefficient.

5. Return to normal operation after writing the coefficients by performing a master boot.

6. The EQ coefficients may be invoked using these commands:

```
ucmd ef000000000000100
ucmd efxxxxxxxxxxxxx #(any other module-decoder, mpm)
ucmd ef00000a0000001c #(app)
ucmd ef00000c0000000* #(PPM mode as described in the example above Step
# 5)
ucmd ef000000000000001
```

Revision History

Revision	Date	Changes
UM1	-	Unpublished on the Web
UM2_beta	July, 2009	Added CS4953x4 product to title and to the descriptions in the various chapters in this manual. Added following footnote on Page 7-11 and a Note in section 8.4.22: "Current firmware does not support change of input source from HDMI/Multichannel source to SPDIF source using DSP Manager API commands. Cirrus recommends using SPDIF as the default input source in the flash_image.xml file and the DSP Reset should be toggled to switch to SPDIF source from HDMI/Multichannel source." Added 2nd typical connection diagram in Figure 1-3 Added description for 0x0003DSP_CFG_INPUT_CHANNELS to Table 7-4 .
UM2_beta2	August, 2009	Added Section 8.7.2.1 . Deleted former Chapter 9 describing the use of DSP Controller on the CS4953x4/CS4970x4 platform. DSP Controller is no longer a supported application.
UM2_beta3	September, 2009	Updated Section 1.2 , "Operational Mode Selection" on page 1-2. Updated Section 7.3.2 , "Writing to the DSP" on page 7-2, changing, changing Write Data Word: bit table to Data [31:0].
UM3	February, 2009	Changed chapter number for Chapter 7 in the previous revision to Chapter 8. Changed chapter number for Chapter 8 in the previous revision to Chapter 7. Updated Table 4-2 . Moved note, "DA02_LRCLK & DA02_SCLK are driven by DA01_LRCLK & DA01_SCLK" from the A1 parameter in Table 4-2 cell to the B0 parameter in Table 4-3 . Updated Table 4-5 . Added Table 4-8 . Added note to Section 4.2.3 . Updated Section C.1.13.1 .
UM4	November, 2011	Updated Table 4-10 to match values seen in Hardware User's Manual.
UM5	February, 2012	Added Chapter 9 . Added Section 8.7.2.2 ., Section 8.7.2.3 ., and Section 8.7.2.4 .
UM6	March, 2013	Updated description of Bits [15:0] in Table 7-1 .