A DIVISION OF CIRRUS LOGIC

CS4922

MPEG Audio Decoder System

Features

- DSP Optimized for Audio Decode, 24-bit Fixed Point w/48-bit Accumulator
- On-Chip Functional Blocks Include:
  - DSP with RAM and ROM Memories
  - CD Quality Stereo DAC with Output Filtering
  - Mono Output & Digital Volume Control
  - S/PDIF Transmitter, Bidirectional PCM Audio Port
  - Programmable Phase Locked Loop for Clocking
  - Dedicated Compressed Serial Input Interface
- MPEG-1 & MPEG-2 Layers 1 & 2 With All Sample/Bit Rates and Ancilliary Data Support.
- MPEG-1 & MPEG-2 Packetized Audio Stream and Elementary Stream Input
- PCM Synthesis for Auxiliary Audio
- Pin Compatibility with CS4920A and Primary Feature/Firmware Compatible
- +5 Volt Only CMOS, 44 pin PLCC

Description

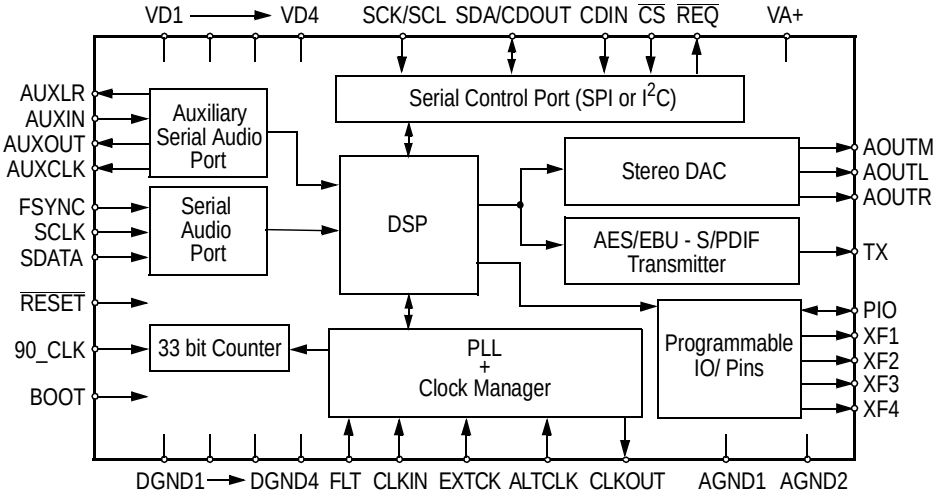
The CS4922 is a complete audio decompression subsystem implemented in a single high integration mixed signal CMOS chip. The CS4922 is a performance enhanced and cost reduced version of the CS4920A audio decoder IC product. The CS4920A has been widely used in direct broadcast system set-top boxes as well as other media terminals including PC adapter cards.

The CS4922 is tailored for MPEG-2 audio decompression to include the necessary hardware and firmware to ensure proper audio/video synchronization. In addition to audio decoding this programmable DSP solution provides robust error concealment and feature implementations like ancilliary data support and PCM synthesis. The CS4922 enhancements include additional MIPS, a bi-directional I^2S port and more general purpose output pins.

The CS4922 will supersede the CS4920A as the most popular MPEG audio decoder on the market and the only complete MPEG audio subsystem IC in the world.

ORDERING INFORMATION

CS4922-CL	44-pin PLCC
CDB4922	Evaluation Board



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

Cirrus Logic, Inc.
Crystal Semiconductor Products Division
P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.crystal.com

Table Of Contents

ANALOG CHARACTERISTICS (T_A = 25 °C; VA+, VD+ = 5V; CLKIN = 27 MHz; Full-Scale Output Sinewave, 1.125 kHz; Word Clock = 48 kHz (PLL in use); Logic 0 = GND, Logic 1 = VD+; Measurement Bandwidth is 20 Hz to 20 kHz; Local components as shown in "Typical Connection Diagram"; SPI mode, I^2S audio data; unless otherwise specified.)

Parameter*		Symbol	Min	Typ	Max	Units
Dynamic Performance						
DAC Resolution			16	-	-	Bits
DAC Differential Nonlinearity		DNL	-	-	±0.9	LSB
Total Harmonic Distortion	AOUTL, AOUTR (Note 1) AOUTM	THD	- -	0.01 0.02	0.015 0.03	% %
Instantaneous Dynamic Range (DAC not muted, A weighted)	AOUTL, AOUTR (Note 1) AOUTM	IDR	85 80	90 85	- -	dB dB
Interchannel Isolation	(Note 1)		-	85	-	dB
Interchannel Gain Mismatch			-	-	0.2	dB
Frequency Response			-3.0	-	+0.2	dB
Full Scale output Voltage	AOUTL, AOUTR (Note 1) AOUTM		2.66 2.7	2.88 3.0	3.2 3.3	Vpp Vpp
Gain Drift			-	100	-	ppm/°C
Deviation from Linear Phase			-	-	5	Deg
Out of Band Energy	(Fs/2 to 2Fs)		-	-60	-	dB
Analog Output Load	Resistance: Capacitance:		8 -	- -	- 100	kΩ pF
Power Supply						
Power Supply Rejection	(1 kHz)		-	40	-	dB
Power Supply Consumption	 VA+ VD+		- - -	 20 100	 40 140	 mA mA

Note: 1. 10 kΩ, 100pF load for each analog signal (Left, Right).
30 kΩ, 100pF load for analog Mono signal.

D/A INTERPOLATION FILTER CHARACTERISTICS

(See graphs toward the end of this data sheet)

Parameter	Symbol	Min	Typ	Max	Units
Passband (to -3 dB corner)(Fs is conversion freq.)		0	-	0.476Fs	Hz
Passband Ripple		-	-	±0.1	dB
Transition Band		0.442Fs	-	0.567Fs	Hz
Stop Band		≥0.567Fs	-	-	Hz
Stop Band Rejection		50	-	-	dB
Stop Band Rejection with Ext. 2Fs RC filter		57	-	-	dB
Group Delay		-	12/Fs	-	s

* Refer to *Parameter Definitions* at the end of this data sheet.
Specifications are subject to change without notice.

ABSOLUTE MAXIMUM RATINGS

(AGND, DGND = 0V, all voltages with respect to ground.)

Parameter	Symbol	Min	Max	Units
DC Power Supplies: Positive Digital	VD+	-0.3	6.0	V
Positive Analog	VA+	-0.3	6.0	V
| | VA+ | - | VD+ | |		-	0.4	V
Input Current, Any Pin Except Supplies	I_{in}	-	±10	mA
Digital Input Voltage	V_{IND}	-0.3	(VD+) + 0.4	V
Ambient Operating Temperature (power applied)	T_{Amax}	-55	125	°C
Storage Temperature	T_{stg}	-65	150	°C

WARNING:Operation at or beyond these limits may result in permanent damage to the device.
Normal operation is not guaranteed at these extremes.

RECOMMENDED OPERATING CONDITIONS

(AGND, DGND = 0V; all voltages with respect to ground.)

Parameter	Symbol	Min	Typ	Max	Units
DC Power Supplies: Positive Digital	VD+	4.50	5.0	5.50	V
Positive Analog	VA+	4.50	5.0	5.50	V
| | VA+ | - | VD+ | |		-	-	0.4	V
Ambient Operating Temperature	T_A	0	-	70	°C

DIGITAL CHARACTERISTICS

(T_A = 25 °C; VA+, VD+ = 5V±10%; measurements performed under static conditions.)

Parameter	Symbol	Min	Typ	Max	Units
High-Level Input Voltage	V_{IH}	TBD	2.25	-	V
Low-Level Input Voltage	V_{IL}	-	-	0.8	V
High-Level Output Voltage at I_o = -2.0 mA	V_{OH}	VD x 0.9	-	-	V
Low-Level Output Voltage at I_o = 2.0 mA	V_{OL}	-	-	VD x 0.1	V
Input Leakage Current (Note 2)	I_{in}	-	-	1.0	µA

Note: 2. Not Valid for pin numbers 9, 12, 13, and 30 which are configured with on-chip pull-down resistors. Not valid for pin number 29 which is a static input signal and should be tied to either VD+ or DGND.

SWITCHING CHARACTERISTICS - CLOCKS

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter	Symbol	Min	Typ	Max	Units
Master Clock Frequency	CLKIN	0.256	27	30	MHz
Master Clock Duty Cycle	CYCK	40	50	60	%
Alternate Clock P = 0 (Note 3)	ALTCLK	-	512Fs	-	Hz
P = 1		-	768Fs	-	Hz
Clock Output	CLKOUT	-	-	256 Fs	MHz

Note: 3. ALTCLK is required to run at 512 or 768 times the sample frequency.

SWITCHING CHARACTERISTICS - EXTERNAL FLAGS

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter	Symbol	Min	Typ	Max	Units
Rise time of XF1-XF4 (Note 4)	t_{rxf}			200	ns
Fall time of XF1-XF4	t_{fxf}			100	ns

Note: 4. Assumes 2kΩ pull-up to 5V supply on XF1-XF4 pins.

SWITCHING CHARACTERISTICS - PROGRAMMABLE INPUT/OUTPUT

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter	Symbol	Min	Typ	Max	Units
I_O = 0 ***(Note 5)***					
Input Frequency	f_{pio}			350	kHz
Risetime of PIO	t_{rpio}			200	ns
Fall time of PIO	t_{fpio}			200	ns
I_O = 1 ***(Note 5)***					
Rise time of PIO	t_{rpo}			200	ns
Fall time of PIO	t_{fpo}			200	ns

Note: 5. I_O is a bit in the LINT register.

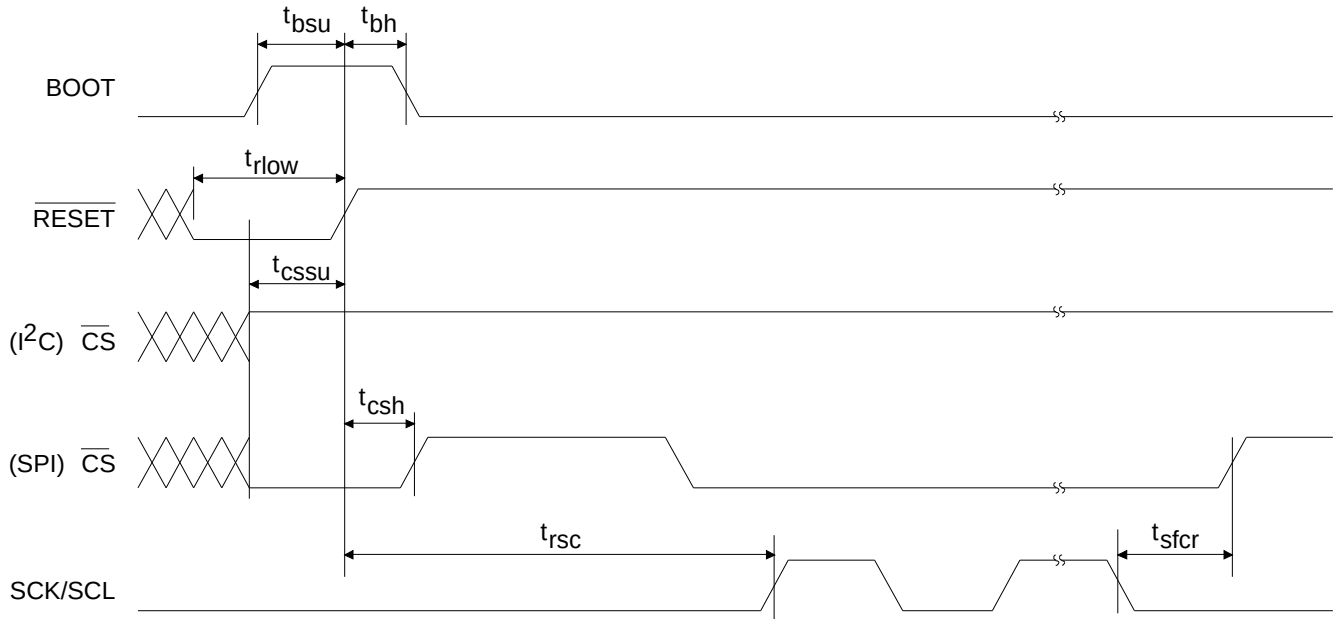
SWITCHING CHARACTERISTICS - BOOT INITIALIZATION

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter		Symbol	Min	Max	Units
BOOT Setup Time to $\overline{\text{RESET}}$ Rising		t_{bsu}	350	-	ns
$\overline{\text{RESET}}$ Rising to Boot Hold Time		t_{bh}	450	-	ns
$\overline{\text{CS}}$ Setup Time to $\overline{\text{RESET}}$ Rising	(Note 6)	t_{cssu}	200	-	ns
$\overline{\text{RESET}}$ Rising to $\overline{\text{CS}}$ Hold Time		t_{csh}	400	-	ns
$\overline{\text{RESET}}$ Low Time		t_{rlow}	50	-	µs
SCK/SCL Delay Time from $\overline{\text{RESET}}$ Rising	(Note 7)	t_{rsc}	2	-	ms
SCK/SCL falling to $\overline{\text{CS}}$ rising on last byte of download		t_{sfcr}	3	-	µs

Note: 6. The mode of the Serial Control Port is selected by $\overline{\text{CS}}$. $\overline{\text{CS}}$ = 1 is I^2C® . $\overline{\text{CS}}$ = 0 is SPI mode.

7. This delay is necessary after any rising edge of $\overline{\text{RESET}}$ to allow time for the part to initialize and for the on-board PLL to stablize.



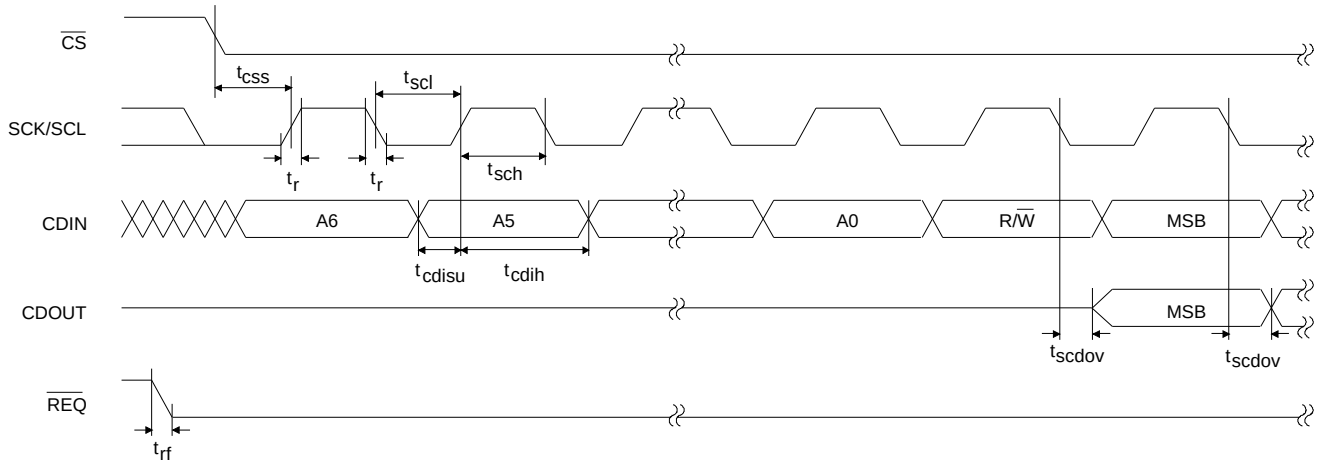
Boot Timing

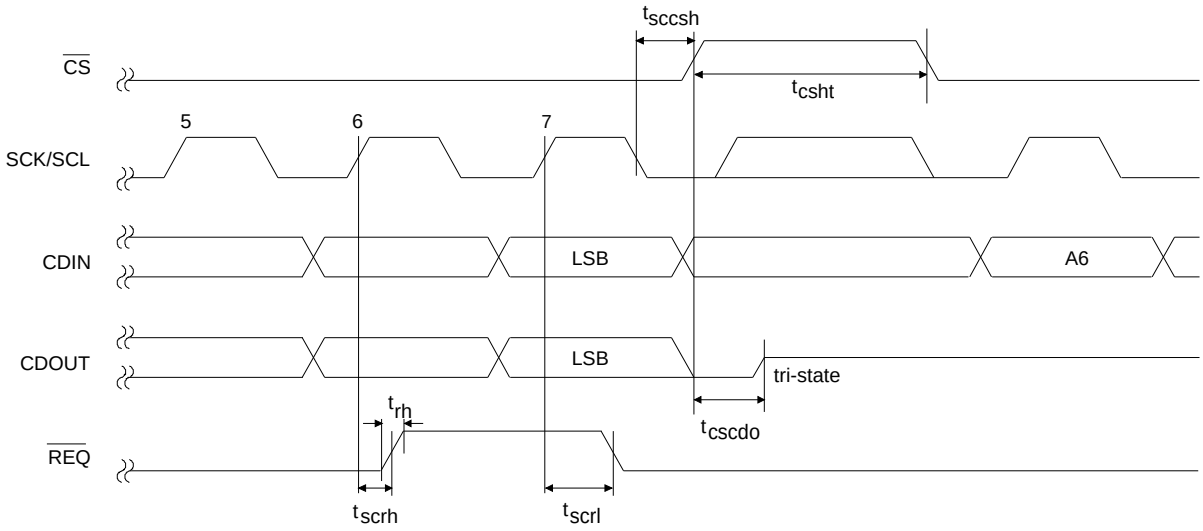
SWITCHING CHARACTERISTICS - CONTROL PORT

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter		Symbol	Min	Max	Units
SPI Mode ($\overline{CS}$ = 0)					
SCK/SCL Clock Frequency mode) mode)	(slow (fast	f_{sck} f_{sck}	- -	350 2000	kHz
$\overline{CS}$ Falling to SCK/SCL Rising	(slow mode)	t_{css}	20	-	ns
Rise Time of Both CDIN and SCK/SCL Lines	(slow mode)	t_r	-	50	ns
Fall Time of Both CDIN and SCK/SCL Lines mode)	(slow mode) (fast	t_f t_f	- -	300 50	ns ns
SCK/SCL Low Time mode) mode)	(slow (fast	t_{scl} t_{scl}	1100 150	- -	ns ns
SCK/SCL High Time mode) mode)	(slow (fast	t_{sch} t_{sch}	1100 150	- -	ns ns
Setup Time CDIN to SCK/SCL Rising mode)	(slow mode) (fast	t_{cdisu}	250 50	- -	ns ns
Hold Time SCK/SCL Rising to CDIN	(Note 8)	t_{cdih}	50	-	ns
Transition Time from SCK/SCL to CDOUT Valid	(Note 9)	t_{scdov}	-	40	ns
Time from SCK/SCL Rising to $\overline{REQ}$ Rising	(Note 9)	t_{scrh}	-	200	ns
Rise Time for $\overline{REQ}$ (Note 10)		t_{rr}	-	50	ns
Fall Time for $\overline{REQ}$ (Note 11)		t_{rf}	-	20	ns
Hold Time for $\overline{REQ}$ from SCK/SCL Rising	(Note 11)	t_{scrl}	0	-	ns
Time from SCK/SCL Falling to $\overline{CS}$ Rising		t_{sccsh}	20	-	ns
High Time Between Active $\overline{CS}$		t_{csht}	200	-	ns

Note: 8. Data must be held for sufficient time to bridge 300(50) ns transition time of SCK/SCL.
9. CDOUT should NOT be sampled during this time period.
10. $\overline{REQ}$ will only go HIGH if there is no data in SCPOUT at the rising edge of SCL/SCK during a READ operation as shown. DSP frequency is 20 MHz. Pull-up resistor is 2 kΩ. C_L = 20 pF.
11. If $\overline{REQ}$ went HIGH as indicated in note 7, then $\overline{REQ}$ will hold high at least until the next rising edge of SCK/SCL. If data is in SCPOUT at this time $\overline{REQ}$ will go active LOW again. This condition should be treated as a new READ process. Address and R/$\overline{W}$ bit should be sent again.





SPI Control Port Timing

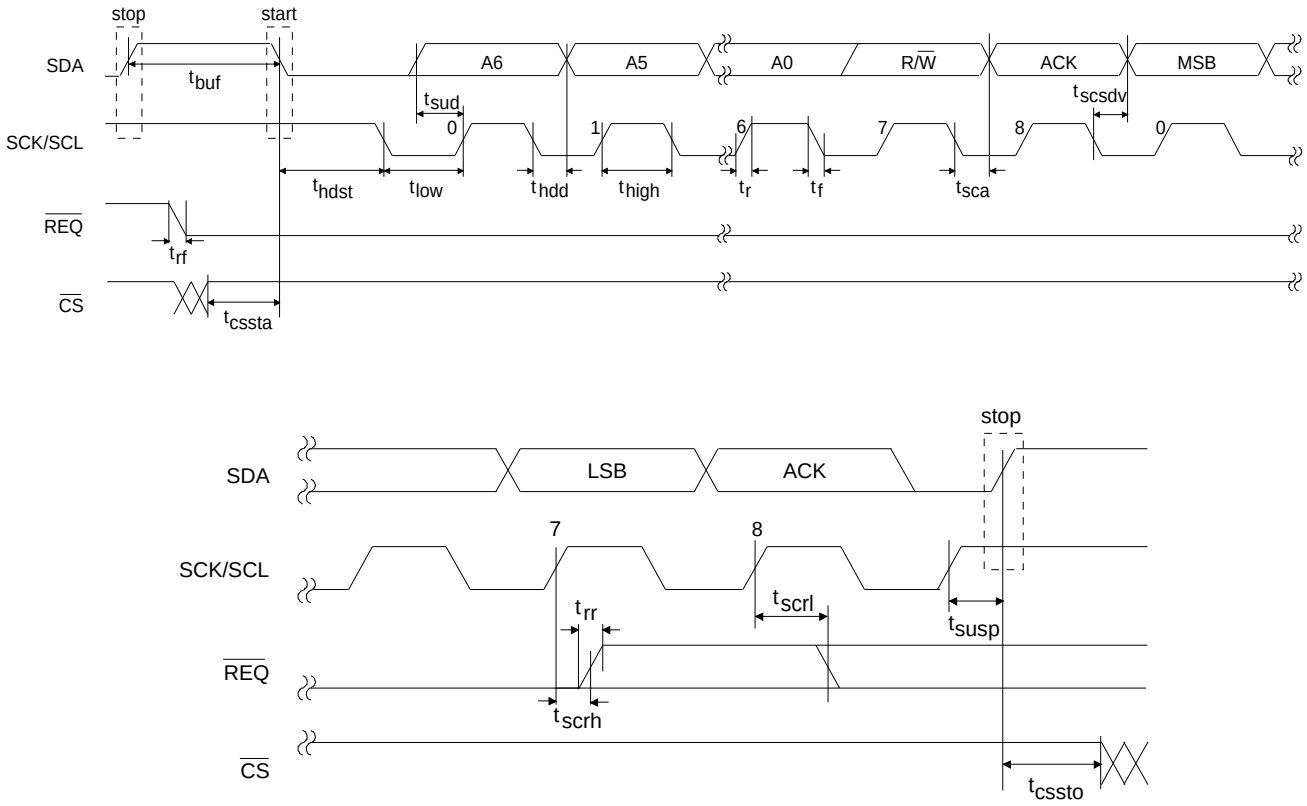
SWITCHING CHARACTERISTICS - CONTROL PORT

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = DGND, Logic 1 = VD+, C_L = 20pF)

Parameter		Symbol	Min	Max	Units
I²C® Mode (CS=1)	***(Note 12)***				
SCK/SCL Clock Frequency mode) mode)	(slow (fast	f_{scl}		100 400	kHz
Bus Free Time Between Transmissions		t_{buf}	4.7		µs
Start Condition Hold Time (prior to first clock pulse)		t_{hdst}	4.0		µs
Clock Low Time	slow fast	t_{low}	4.7 1.2		µs
Clock High Time	slow fast	t_{high}	4.0 1.0		µs
SDA Setup Time to SCK/SCL Rising		t_{sud}	250		ns
SDA Hold Time from SCK/SCL Falling	(Note 13)	t_{hdd}	0		µs
Rise Time of Both SDA and SCK/SCL	(Note 14)	t_r		50	ns
Fall Time of Both SDA and SCK/SCL		t_f		300	ns
Time from SCK/SCL Falling to CS4920 ACK		t_{sca}		40	ns
Time from SCK/SCL Falling to SDA Valid During READ Operation		t_{scsdv}		40	ns
Time from SCK/SCL Rising to $\overline{REQ}$ Rising	(Note 15)	t_{scrh}		200	ns
Hold Time for $\overline{REQ}$ from SCK/SCL Rising	(Note 16)	t_{scrl}	0		ns
Rise Time for $\overline{REQ}$ 15)	(Note	t_{rr}		50	ns
Fall Time for $\overline{REQ}$ 16)	(Note	t_{rf}		20	ns
Setup Time for Stop Condition		t_{susp}	4.7		µs

Note: 12. Use of I²C® bus compatible interface requires a license from Philips. I²C is a registered trademark of Philips Semiconductor.

13. Data must be held for sufficient time to bridge the 300ns transition time of SCK/SCL.
14. This rise time is shorter than the I²C specifications recommend, please refer to the section on SCP communications for more information.
15. $\overline{REQ}$ will only go HIGH if there is no data in the SCPOUT register at the rising edge of SCL/SCK during a READ operation as shown. DSP frequency is 20 MHz. Pull-up resistor is 2 kΩ C_L = 20pF.
16. if $\overline{REQ}$ went HIGH as indicated in Note 15 then $\overline{REQ}$ will hold HIGH at least until the next rising edge of SCK/SCL. If data is in the SCPOUT register at this time $\overline{REQ}$ will go active LOW again. This condition should be treated as a new READ process. The address and R/$\overline{W}$ should be sent again following a new START condiiton.



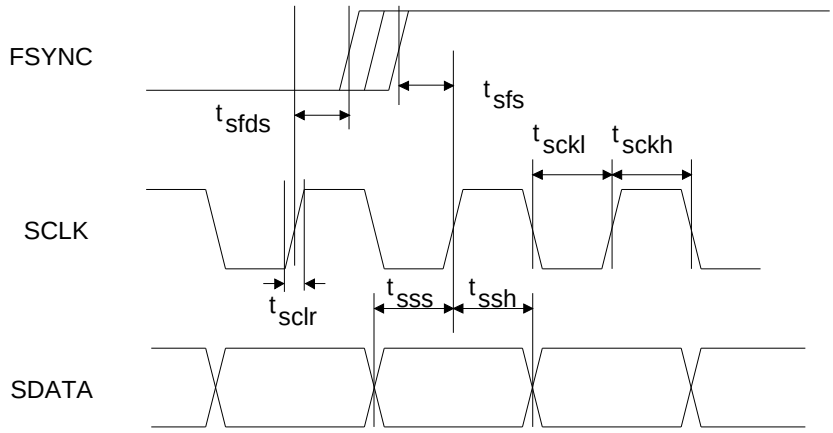
I²C® Control Port Timing

SWITCHING CHARACTERISTICS - SERIAL AUDIO PORT

(T_A = 25 °C; VA+, VD+ = 5V; Inputs: Logic 0 = GND, Logic 1 = VD+; C_L = 20 pF)

Parameter		Symbol	Min	Typ	Max	Units
SCLK Frequency			-	-	12.5	MHz
SCLK Pulse Width Low		t_{sckl}	25	-	-	ns
SCLK Pulse Width High		t_{sckh}	25	-	-	ns
SCLK rising to FSYNC edge delay	(Note 17)	t_{sfds}	20	-	-	ns
SCLK rising to FSYNC edge setup	(Note 17)	t_{sfs}	20	-	-	ns
SDATA valid to SCLK rising setup	(Note 17)	t_{sss}	20	-	-	ns
SCLK rising to SDATA hold time	(Note 17)	t_{ssh}	20	-	-	ns
Rise time of SCLK		t_{sclr}	-	-	20	ns

Note: 17. The table above assumes data is output on the falling edge and latched on the rising edge. The SCLK edge is selectable in setting the EDG bit in the ASICN register. The diagram is for EDG = 1.

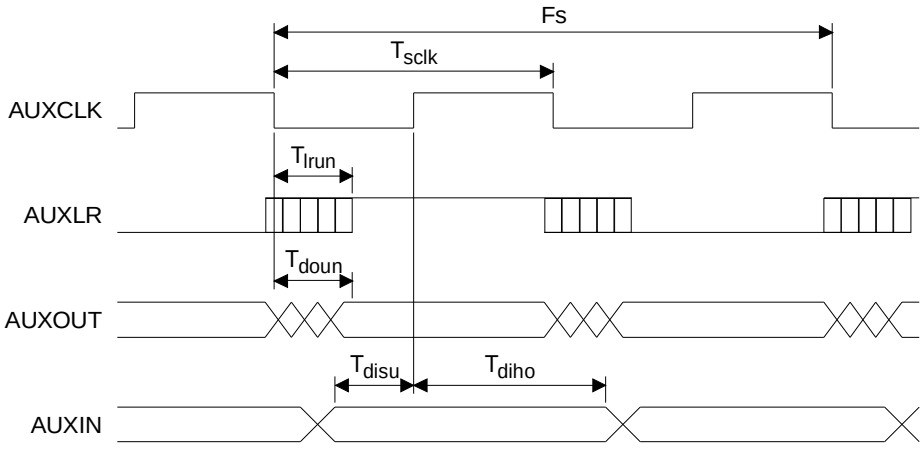


Serial Audio Port Timing

SWITCHING CHARACTERISTICS - AUXILIARY DIGITAL AUDIO PORT

Parameter	Symbol	Min	Typ	Max	Units
Input Sample Rate (Note 18)	Fs	16	-	48	kHz
AUXCLK Period (Note 19)	tsclk	-	1/(32Fs) 1/(64Fs) 1/(128Fs)	-	ns
AUXCLK to AUXLR valid	tlrun	0	-	25	ns
AUXCLK to AUXOUT data valid	tdoun	0	-	25	
AUXIN data setup time to AUXCLK	tdisu	50	-	-	ns
AUXIN data hold time from AUXCLK	tdiho	3	-	-	ns

Note: 18. Fs determined by clock input rate and configuration of on-chip pll.
19. AUXCLK frequency selectable @ 32, 64, or 128 Fs via AUXCN register bits 1:0.



Auxiliary Audio Port Timing

1. THEORY OF OPERATION

1.1. Introduction

The CS4922 is a complete audio subsystem on a chip. It consists of a general-purpose Digital Signal Processor (DSP), and a number of supplementary analog and digital blocks. These supplementary blocks include a programmable PLL clock multiplier, a serial audio input port, an auxiliary serial audio port, a CD quality stereo Digital-to-Analog Converter (DAC), an AES/EBU - S/PDIF compatible digital audio transmitter, and a serial control port. Figure 1 shows a typical connection diagram for the CS4922 in which a micro controller is used for loading the program code.

The CS4922 is based on a programmable DSP core. A typical application is the decoding of a compressed digital audio signal. Serial audio data broadcast on networks such as cable TV, direct broadcast satellite TV, or the telephone system can be decompressed and converted to standard analog and digital signals. A wide variety of standard and proprietary decompression algorithms can be supported. An assembler, a simulator, and a debugger are available. CS4922 DSP code is available which performs industry standard MPEG 1 and 2, layers I and II.

The DSP has a 24-bit fixed point data path, 5K words of program RAM, and 3K words of data RAM. The execution unit includes a 48-bit accumulator. Typical ALU instructions read operands

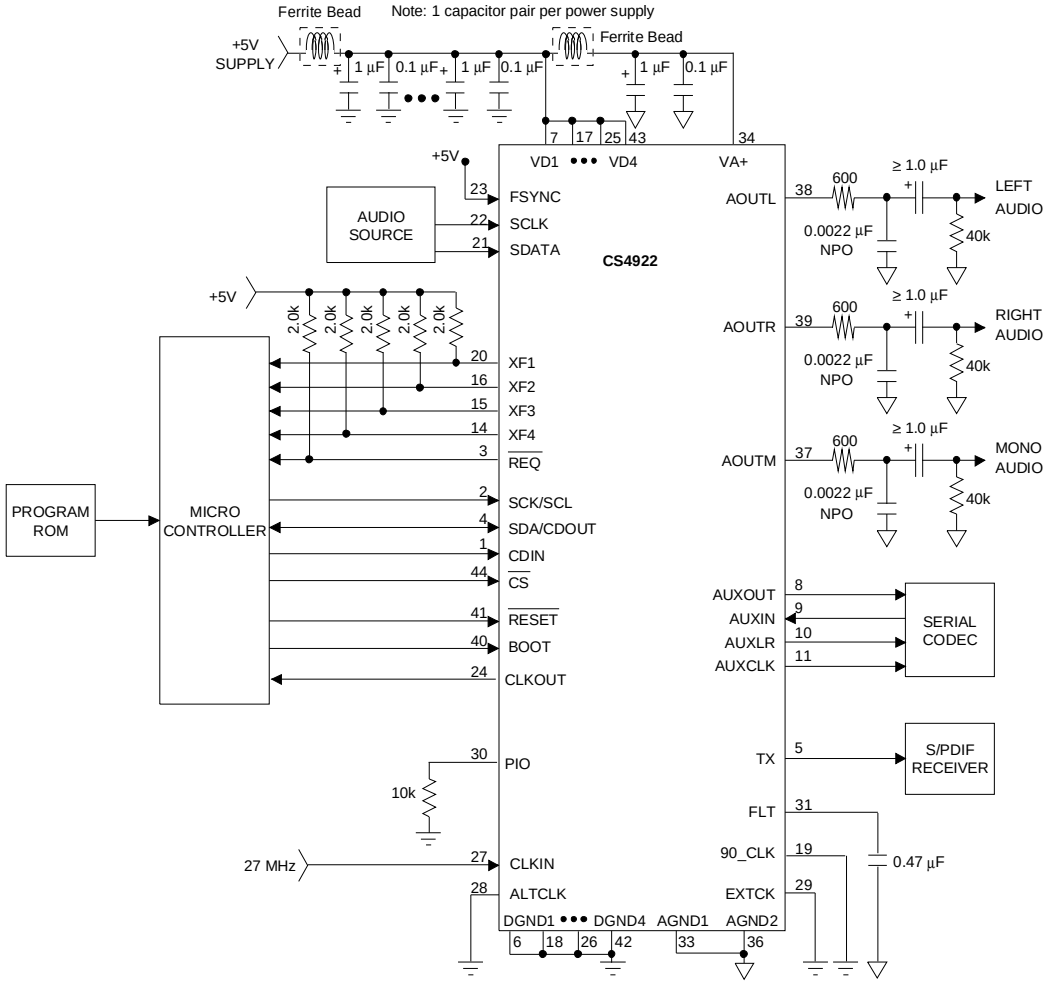


Figure 1. Typical Connection Diagram

from memory and store results back to memory. Modulo and bit reverse addressing are supported. For a sample rate of 48 kHz, the DSP can provide 18 MIPS.

The CS4922 includes a flexible clock manager. This section allows clock inputs on CLKIN to range from 256 kHz to 30 MHz, while producing the necessary DSP and DAC clocks through a programmable PLL.

The digital audio data is input through the serial audio port. Multiple data formats are supported by this port.

For analog reproduction of the digital input, a stereo DAC using delta-sigma architecture is built-in. Switched-capacitor filters perform most of the reconstruction process. Only a simple external passive filter is needed to complete reconstruction.

In addition to the analog output, an AES/EBU - S/PDIF compatible output is provided. This allows the designer the flexibility of transmitting the audio data in a standard digital format to an external system.

To facilitate the downloading of DSP code to the CS4922, a serial control port, communicating in either I^2C® or SPI format, is used. This port may also be used during run time to issue control commands to the DSP.

2. PERIPHERALS

Six on-chip peripherals make the audio decoder ideal for decoding broadcast digital audio signals. It has a PLL clock manager, a CD quality DAC, a digital audio transmitter, a three pin serial port for inputting audio data, a serial bi-directional auxiliary port for digital audio data, and an SPI/I^2C port for serial control information. Each peripheral has I/O mapped data, control, and status registers. Many peripherals can also generate interrupts.

2.1. Clock Manager

The clock manager is primarily a clock multiplier circuit that takes CLKIN input of any frequency from 256 kHz to 30 MHz and produces all the internal clocks required to run the DSP and the audio peripherals. A block diagram of the clock manager is shown in Figure 2.

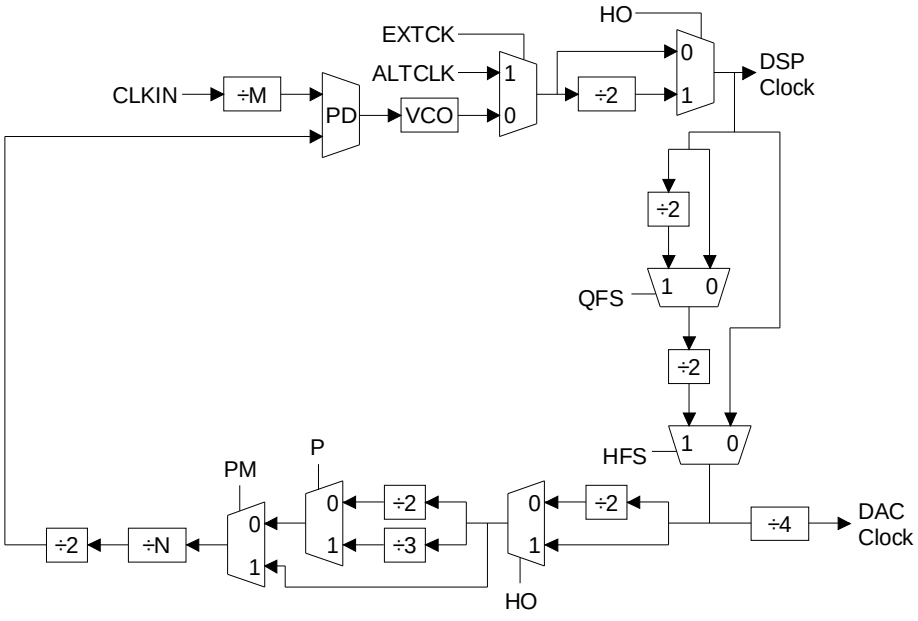


Figure 2. Clock Manager

At the heart of the clock manager circuit is a PLL (Phase-Locked Loop) circuit. The PLL is configured to produce the appropriate DSP Clock for the desired sample rate. All other internal clocks required for the DAC and other peripherals are derived from this root clock.

The PLL's internal VCO requires a capacitor to be connected to the FLT pin (pin 31). The typical value of the FLT capacitor is 0.47 µf , which is sufficient for all allowable CLKIN input frequencies. It must be stressed that the best analog performance can only be achieved by placing the capacitor as close as possible to the FLT pin and that the proper layout precautions be taken to avoid noise coupling onto the FLT pin.

The PLL generates the DSP clock by dividing the CLKIN input frequency by M and locking that reference clock to a divided down version of the DSP clock, whose frequency is set by the value of N in the CM1 register and the HO, HFS, P, and PM bits in the CM0 register. The clock manager circuitry can support sample rate frequencies in the range of 8 KHz to 50 KHz.

A typical CLKIN input frequency is the 27 MHz MPEG reference clock. Many other input frequency combinations are possible, such as the compressed bit rate of the input audio stream. Table 1 shows the recommended clock manager configuration for common sample rates with a 27 MHz CLKIN.

The external clock pin (EXTCK) specifies the function of ALTCLK. When EXTCK Is low, the internal VCO of the PLL is used to provide the DSP clock. If EXTCK is high, an external oscillator connected to ALTCLK can be used to generate the DSP clock. Note that the frequency of ALTCLK should typically be 512Fs or 768Fs for proper DAC clock generation. (The sampling rate is commonly called Fs). Additionally, the duty cycle of ALTCLK should be as close to 50% as possible.

The CLKOUT pin is a divided version of the DSP clock. A diagram of the CLKOUT generation circuit is shown in Figure 3.

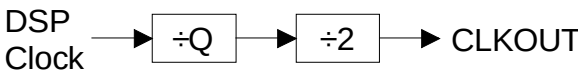


Figure 3. CLKOUT Generation Circuit

Fs (KHz)	M	N	P	PM	HFS	QFS	DAC Clock (MHz)	DSP Clock (MHz)
8	1125	64	1	1	1	1	1.536	24.576
16	3375	256	0	1	1	1	2.048	32.768
22.05	625	98	1	1	1	0	4.233	33.868
24	375	64	1	1	1	0	4.608	36.864
32	3375	512	0	1	1	0	4.096	32.768
44.1	625	196	1	1	0	0	8.467	33.868
48	375	128	1	1	0	0	9.216	36.864

Table 1. Enhanced Clock Manager Configuration

The DSP clock is divided by a programmable divider and an additional divide by 2 before being output. The divider output is stored in the ten bit field Q in the CM0 register. The divide by 2 guarantees a 50% duty cycle output. The Q value provides effective divides ranging from 1 to 1024, which means the frequency of CLKOUT can vary from the DSP clock frequency divided by 2 to the DSP clock frequency divided by 2048. CLKOUT can be used to synchronize external devices or generate most compressed bit rate clocks.

2.2. 33-bit Counter

The 33-bit-counter can be used to support MPEG synchronization of audio and video. This loadable counter is targeted to operate at 90 kHz. The 90 kHz clock may be derived from a 27 MHz master clock provided at CLKIN (if available) or from a 90 kHz clock provided at Pin 19 90_CLK. The selection of the counter clock is made via the register bit DIV in register CM0. When set, the DIV bit divides the clock at CLKIN by 300 and provides the divided clock to 33-bit-counter.

2.3. Digital to Analog Converter

The digital to analog converter (DAC) is a dual channel CD quality DAC. It is designed with delta sigma architecture. The baseband audio is interpolated to 128Fs (192Fs) before going into the modulator. The modulator is third order and is followed by a 1 bit DAC/switch capacitor filter stage. An external passive filter completes the reconstruction process. The output is single ended with a drive capability down to 8 kΩ. Figure 4 is a block diagram of the DAC.

The interpolation filter produces images which are attenuated by at least 56 dB from .584Fs to 128Fs (192Fs). At a 48 kHz sample rate, a full scale signal at 20 kHz will produce an image at 28 kHz which is attenuated by more than 60 dB.

The out-of-band quantization noise from the delta sigma modulator extends from .417Fs to 128Fs (192Fs). This noise is attenuated by the switch capacitor filter and the continuous time filters. The total quantization noise and thermal noise from the analog filters integrated over the .417Fs to 128Fs (192Fs) is more than 50 dB below full scale power.

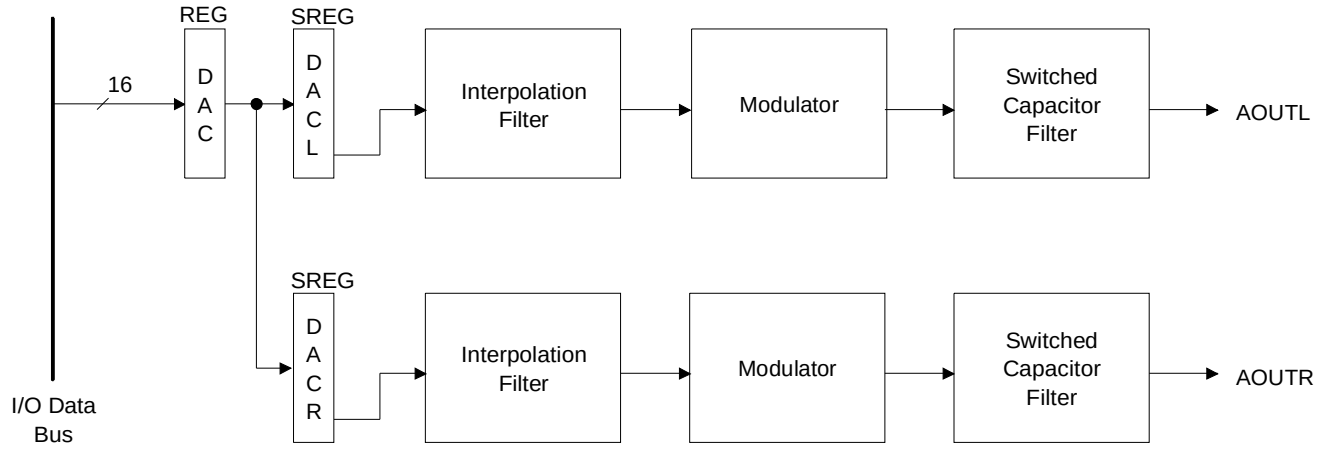


Figure 4. DAC

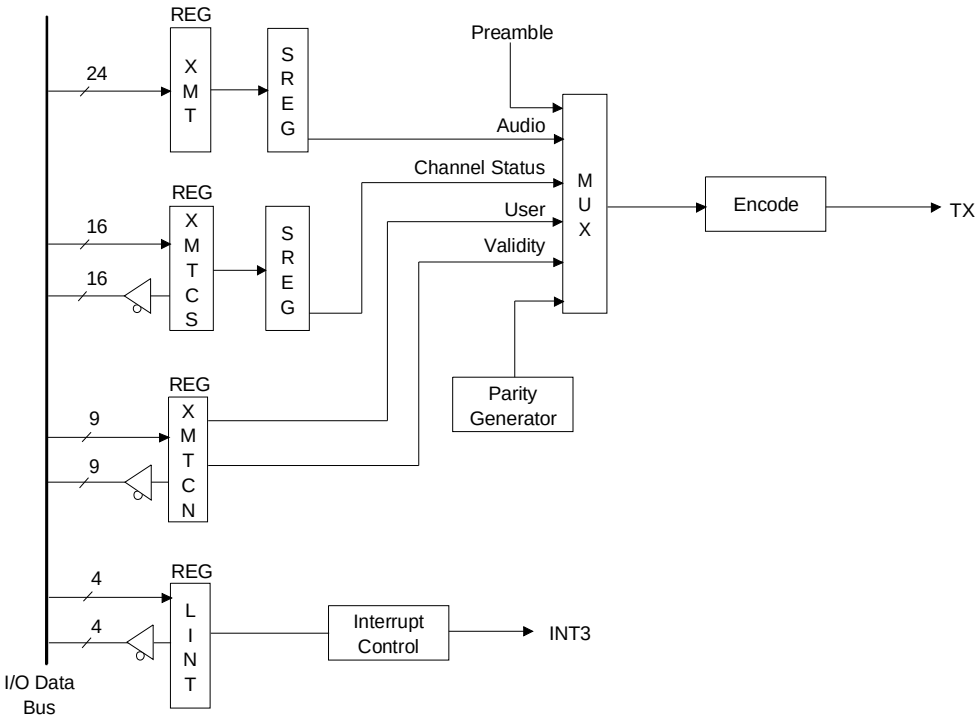


Figure 5. Digital Audio Transmitter

Left and right audio data are written to the DAC output register. This is a 16 bit write only register. The high 16 bits of the I/O data bus can be written to this register. The DACL and DACR output shift registers must be updated at the sample rate determined by the clock manager block. Feeding all zeros to the DAC register will produce a typical signal level of 2.2 volts.

2.4. Digital Audio Transmitter

The transmitter encodes digital audio data according to the Sony®/Philips® Digital Interface Format (S/PDIF) or the AES/EBU interface format. The encoded data is output on the TX pin. More information on the S/PDIF and AES/EBU standards are available from Crystal's application note library. The block diagram of the transmitter is shown in Figure 5.

The transmitter has a 24 bit write only register for audio data (XMT), a 16 bit read/write register for channel status data (XMTCS), and a read/write control register (XMTCN). The audio and channel status data are read from their registers and multiplexed with the validity and user bits from the control register, and the internally generated parity bit.

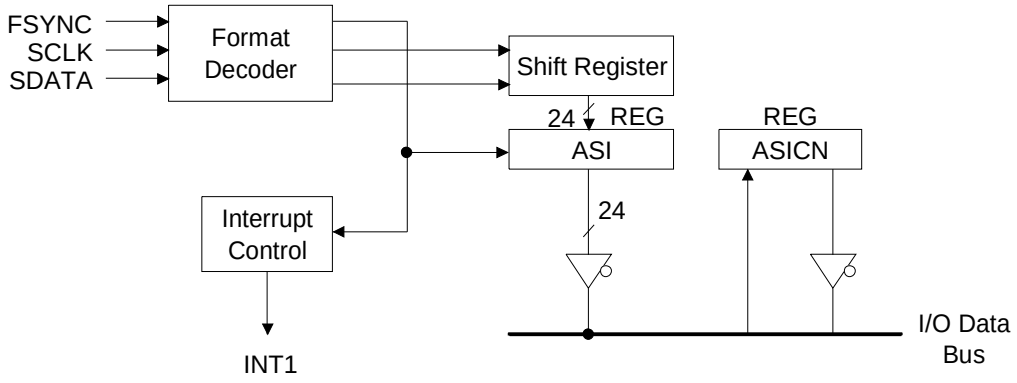


Figure 6. Audio Serial Input Port

2.5. Audio Serial Input Port

The audio serial input port has a three pin interface consisting of FSYNC, SCLK, and SDATA. SCLK clocks SDATA (serial data input) into the 24 bit input shift register. The contents of this shift register can be loaded into the Audio Serial Input (ASI) register by transitions of FSYNC or by a counter time out. An interrupt can be generated when ASI is loaded. Figure 6 shows a block diagram of the audio port.

The pulse mode (PUL) control bit in the audio serial port control register (ASICN) specifies whether both edges (PUL=0) or one edge (PUL=1) of FSYNC loads ASI. When configured to load on both edges, normal mode, up to 24 bits of data after a transition of FSYNC are clocked into the shift register. The next transition of FSYNC loads ASI with the contents of the shift register. Any number of SCLK periods can occur between FSYNC edges; the first 24 bits (or less) are accepted.

When the serial port is configured to load ASI on only one FSYNC edge, pulse mode, any number of SCLK periods can occur between active edges. ASI will be loaded on the active edge of FSYNC and every 16 or 24 SCLK periods after an active edge of FSYNC. The CNT16 bit in ASICN selects between 16 and 24 SCLK's. If FSYNC never toggles, ASI will be continuously loaded every 16 or 24 SCLK periods.

The serial data is typically entered into the port MSB first. If at least 24 SCLK's occur between the times that ASI is loaded, the MSB of the input data will be loaded into the MSB of ASI. If the number of SCLK's between the times that ASI is loaded equals 24 minus N, the MSB will be loaded into the MSB minus N location in ASI. Shifting in software may be required to align the data.

The delay (DEL) bit in ASICN shifts the timing between FSYNC and SDATA by one SCLK period. When DEL is low, the MSB of the data in should occur immediately following a transition of FSYNC. When DEL is high, the MSB should occur one SCLK period after a transition of FSYNC.

The edge (EDG) bit internally inverts SCLK. When EDG is low, the falling edge of SCLK latches SDATA into the shift register. When EDG is high, the rising edge of SCLK latches SDATA.

The polarity (POL) bit in ASICN inverts FSYNC. This switches the active edge of FSYNC in pulse mode. When not in pulse mode, FSYNC can be used to identify left and right channels of stereo audio data. When POL is low, FSYNC high identifies the left channel and when POL is high, FSYNC high identifies the right channel. Figure 7 shows the timing relationships of the various formats.

2.6. Auxiliary Digital Audio Port

The CS4922 auxiliary port provides a path for the internal DSP core to directly read and write framed PCM digital audio data. The auxiliary port is designed to operate in a full duplex mode that can support simultaneous PCM input and output.

The DSP configures the port by programming the AUXCN Auxiliary Port Control Register. The port has the capability to support multiple digital audio formats. The formats are determined by AUXCN and each format is illustrated in Figures 8, 9, and 10. The input and output formats are controlled with the same register bits and thus are always in the same mode. The input and output sampling rates are the same as the sample rate for the on-chip DAC and are configured via the CM0 and CM1 clock manager control registers. The AUXCN mode settings provide the capability to support 16, 18, and 20 bit samples with up to 4 channels of input and 6 channels of output. The auxiliary interface is enabled or disabled by AUXCN. AUXCN also provides a mechanism for scaling AUXCLK to be 32, 64, or 128Fs.

The CS4922 Auxiliary digital audio port physically is implemented with four device pins: AUXCLK pin 11, AUXLR pin 10, AUXIN pin 9, and AUXOUT pin 8. AUXCLK is utilized as the primary synchronous clock. AUXOUT is the serial audio data output pin and AUXIN is the serial audio data input pin. AUXLR is an output pin used for framing the auxiliary digital audio port. AUXLR cycles at the same Fs as the on-chip stereo DAC. Fs is programmed by the DSP. AUXLR and AUXOUT transition with the falling edge of AUXCLK. The rising edge of AUXCLK samples AUXIN.

2.7. Serial Control Port

The serial control port (SCP) can operate in I^2C or SPI compatible modes. In either mode, the control port performs eight bit transfers and is always configured as a slave. As a slave, it cannot drive the clock signal nor initiate data transfers. The port can request to be serviced by activating the $\overline{REQ}$ pin. The port is an asynchronous interface which provides interrupts and handshaking signals to allow communication between the on-chip DSP and an off-chip device such as a micro controller. Figure 12 shows a block diagram of the port.

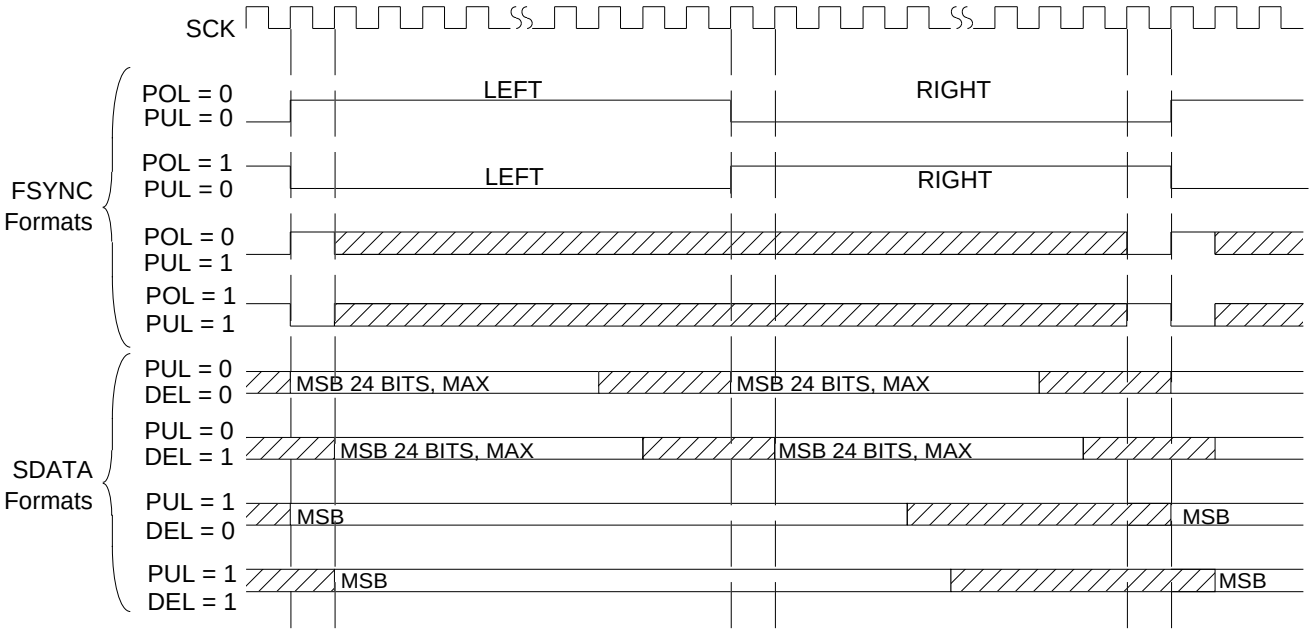


Figure 7. Audio Serial Input Formats

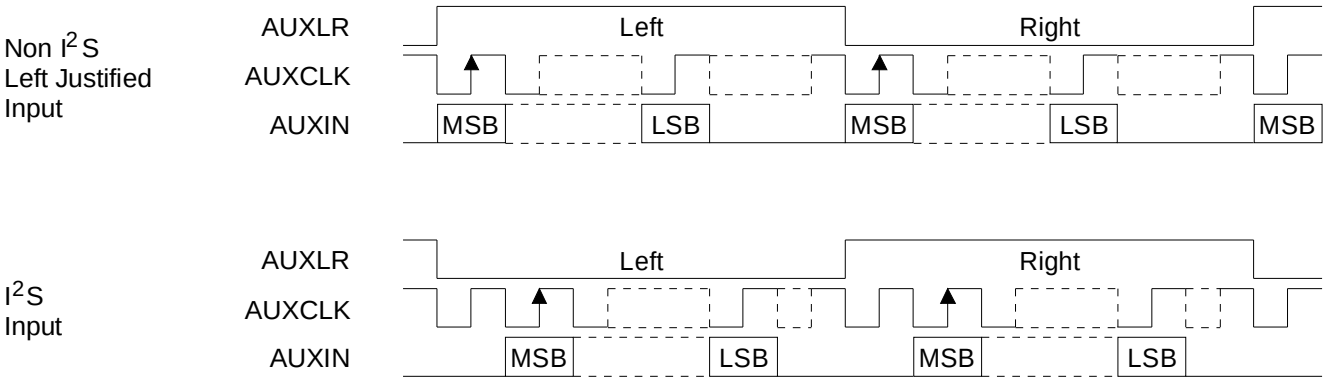


Figure 8. Auxiliary Data Input Formats

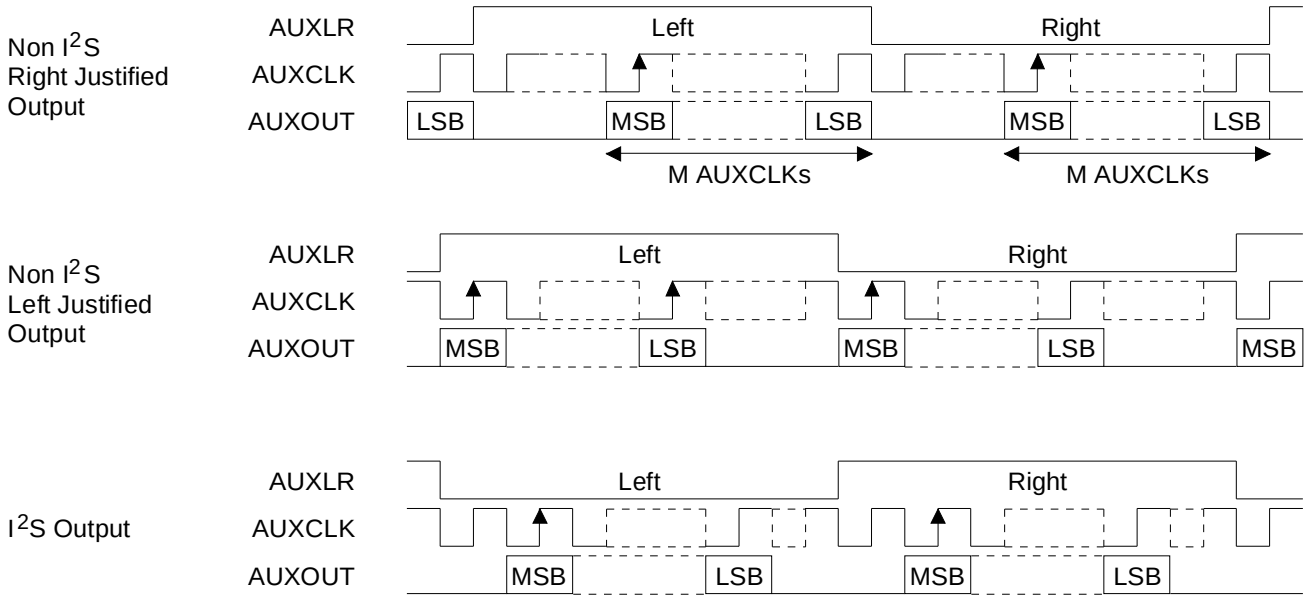


Figure 9. Auxiliary Data Output Formats

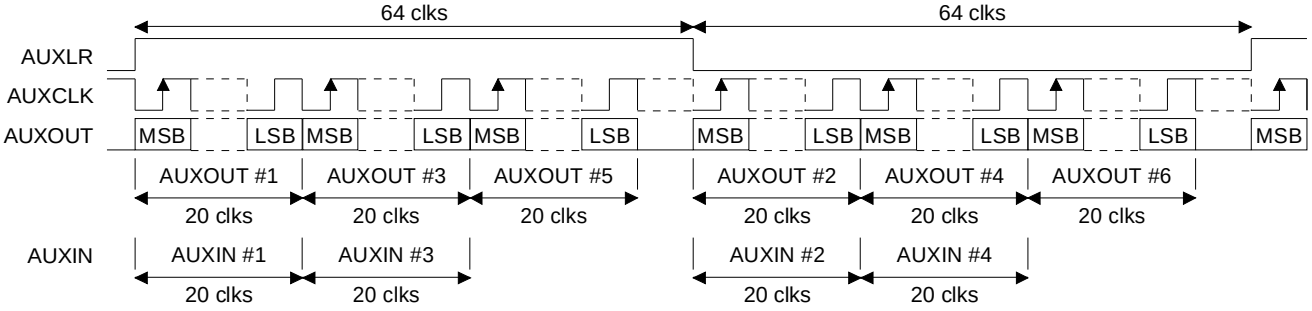


Figure 10. Multi-channel Auxiliary Data Formats

2.7.1. I²C mode

The status of $\overline{CS}$ sets the mode of the SCP during a hardware and software reset. If $\overline{CS}$ is high during a reset the mode is I²C. Note that in most systems where I²C is the preferred control mode, $\overline{CS}$ is connected to the digital supply.

For normal I²C operation SCL/SCK, SDA, and $\overline{REQ}$ are used. $\overline{CS}$ and CDIN are typically connected to the digital supply. SCL/SCK is the serial clock input which is always driven by an external device. SDA is the serial data Input/Output signal. $\overline{REQ}$ is the active low request signal, which is driven low when there is valid data in the serial control port output SCPOUT register.

As an I²C compatible port, data is communicated on the SDA pin and is clocked by the rising edge of SCL/SCK. The Philips I²C bus specification provides details of this interface. Note the CS4922 does not meet the rise time specification of the SCL/SCK signal. For more details please refer to the section on Rise Time of SCL/SCK.

Figure 11 shows the relative timing necessary for an I²C write operation for a single byte. A 'write' is defined as the transfer of data from an I²C bus master to the CS4922 serial control port. A transfer is initiated with a start condition followed by a 7 bit address and a read/write bit (set low for a write). This address is the address assigned to the device being written to during the transfer. In the case of the CS4922, this address is stored in the SCPCN register. Immediately following power up, the CS4922's Address checking Enable (AEN) bit is set to zero. The AEN bit must be set high for the CS4922 to compare the address of the intended I²C device on the bus to its internal address. This means the CS4922 will respond to any address on the I²C bus until its address is initialized and address checking is enabled. To avoid bus conflicts the CS4922 should be held in reset ($\overline{RESET}$ active low) until the master is ready to communicate with the CS4922 and sets the address in the SCPCN. The address can only be set using the I²C bus interface, so the master should use the intended I²C address when downloading microcode to the CS4922 to avoid conflict with other devices on the bus. Once the microcode is loaded into the CS4922 the microcode should either initialize the I²C address or provide a means for the master to program the I²C address. If the CS4922 is the only device on the I²C bus, address checking is optional. However, I²C bus protocol is still required. In other words, the address bits and read/write bit are still required.

If a write to the CS4922 is specified, 8 bits of data on SDA will be shifted into the input shift register as shown in Figure 12. When the shift register is full, the 8 bit data is transferred to the Serial Control Port Input (SCPIN) register on the falling edge of the 8th data bit. An acknowledge (ACK) is sent back to the master and the input ready flag (IRDY) flag is set. This flag generates an interrupt on interrupt line 2 if the input ready interrupt enable

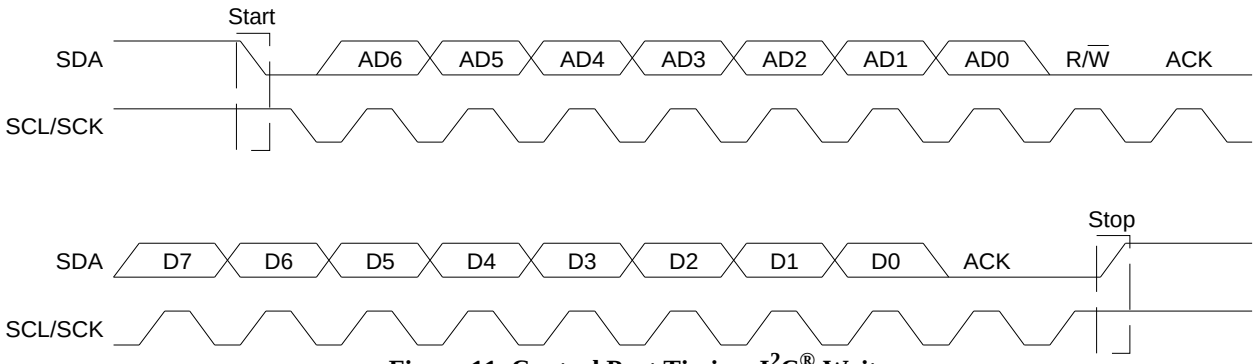


Figure 11. Control Port Timing, I²C® Write

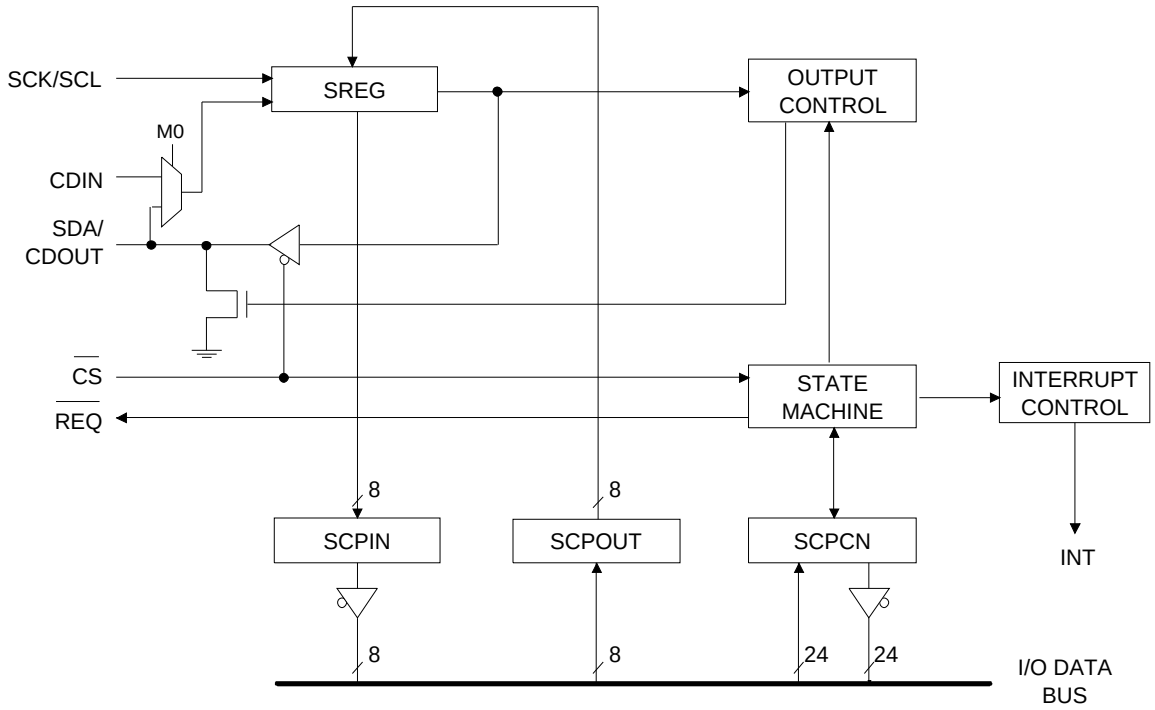


Figure 12. Serial Control Port

(IRIEN) bit is set high. The interrupt vector is 0006H.

The master can continue to send data, but is will be rejected if the IRDY has not yet been cleared. This flag is cleared by reading the SCPIN register. If a byte is rejected, the reject (REJ) flag in the Long Interrupt (LINT) register is set. A rising edge of the REJ flag generates an interrupt on interrupt line 3 if the reject interrupt enable (RJIEN) bit is set high. The REJ flag is cleared by reading SCPCN. If the CS4922 fails to ACK it is possible that the byte was rejected and it should be transmitted again. If the second attempt fails the CS4922 should be issued a hardware reset to reinitialize the communication path.

If the DSP core of the CS4922 wants to send a byte to the master, it first writes the byte to the Serial Control Port Output (SCPOUT) register. Note the DSP only sends 8 bits per transfer to the SCPOUT register. A write to the SCPOUT sets the request pin ($\overline{REQ}$) active low. The master must recognize the request and issue a read operation to the DSP. Figure 13 shows the relative timing of a single byte read. The master must send the 7 bit address (if address checking is enabled it must match the address in the SCPCN register) and the read bit. For I^2C protocol, it is always the device receiving the transfer that must ACK. Therefore, the CS4922 will ACK the address and the read bit. After the ACK by the CS4922 (the falling edge of SCL/SCK), the serial shift register is loaded with the byte to be sent and the most significant bit is placed on the SDA line.

The 8 bit value in the serial shift register is shifted out by the master. The data is valid on the rising edge of SCL/SCK and transitions immediately following the falling edge. For I^2C the $\overline{REQ}$ line will be de-asserted immediately following the rising edge of the last data bit of the current byte being transfered, if there is no data in the SCPOUT register. The $\overline{REQ}$ line is guaranteed to stay de-asserted (high) until the rising edge of the SCL/SCK for the

ACK. This signals the host that the transfer is complete.

If there is data placed in SCPOUT prior to the rising edge of SCL/SCK for the last data bit, then $\overline{REQ}$ will remain asserted (low). Immediately following the falling edge of SCL/SCK for the ACK, the new data byte will be loaded into the serial shift register. The host should continue to read this new byte. It is importat to note that once the data is in the shift register, clocks on the SCL/SCK line will shift the data bits out of the shift register. A STOP condition on the bus will not prevent this from occuring. The host must read the byte prior to any other bus activity or the data will be lost.

If data is placed in SCPOUT after the rising edge of SCL/SCK for the last data bit, but before the rising edge of SCL/SCK for the ACK, $\overline{REQ}$ will not be asserted until after the rising edge of SCL/SCK for the ACK. This should be treated as a completed transfer. The data written to SCPOUT will not be loaded into the shift register on the falling edge of SCL/SCK for the ACK. Therefore, a new read operation is required to read this byte.

2.7.2. Rise Time on SCL/SCK

The Philips I^2C bus specification allows for rise times of the SCL/SCK line up to 1 µs. The CS4922 does not meet this specification. If the I^2C bus master(s) has a rise time in excess of 50 ns the CS4922 will be unable to reliably communicate across the bus. In some systems a stronger pull-up resistor on the SCL/SCK line will provide the rise time needed for proper operation, but this is only helpful when the current rise time is near 50 ns. In cases where the CS4922 will be used in a system where a longer rise time on SCL/SCK is expected, a CMOS compatible buffer should be used. Figure 14 shows the necessary connections. Note the buffer is only used for the SCL/SCK connecting directly to the CS4922. Other devices on the I^2C bus may need to hold SCL/SCK low while accepting data.

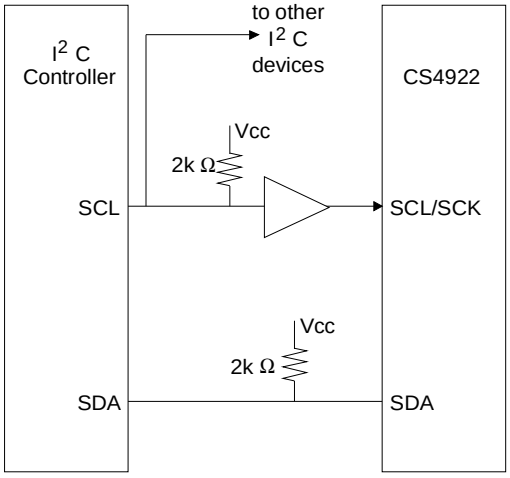


Figure 14. I^2C® Connection Diagram

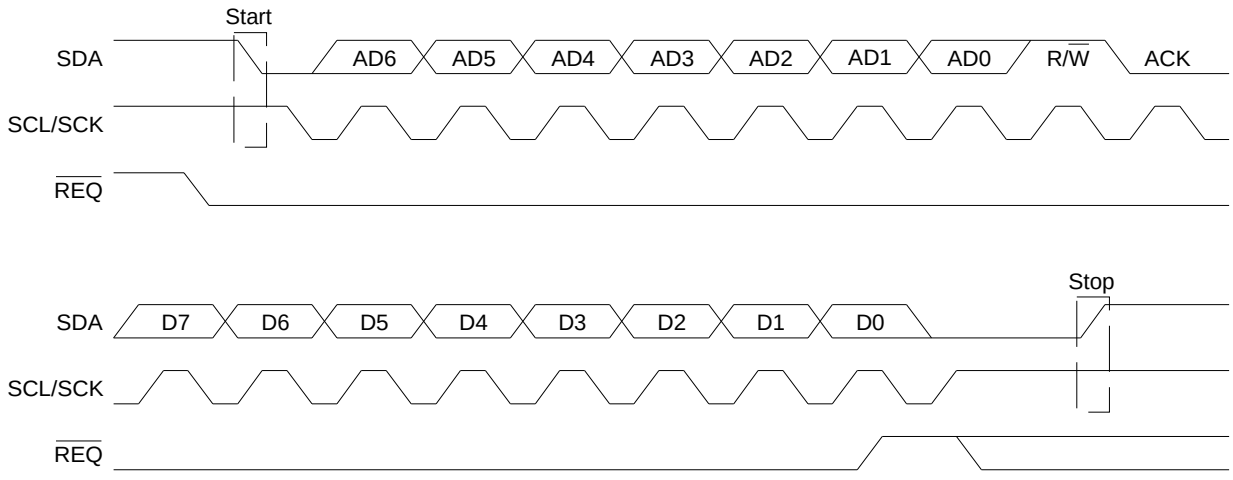


Figure 13. Control Port Timing, I^2C® Read

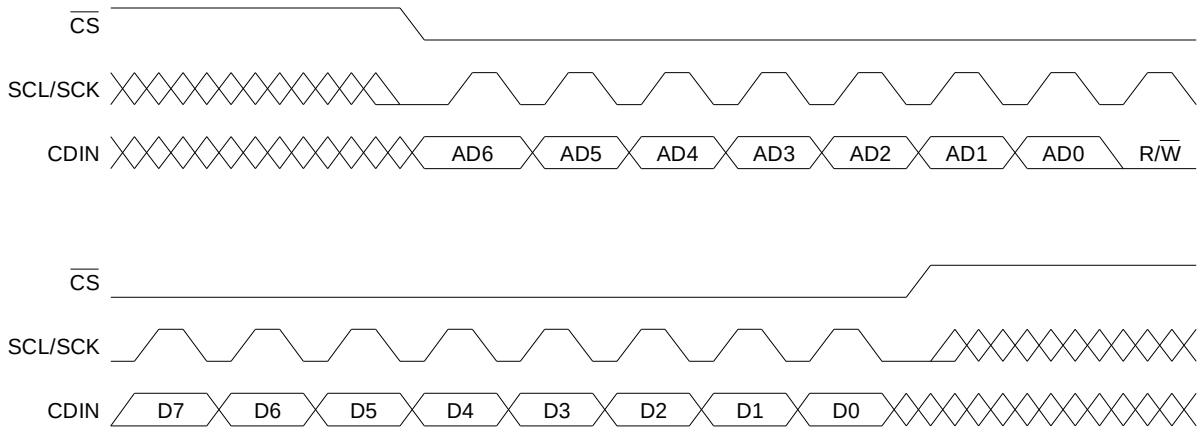


Figure 15. Control Port Timing, SPI Write

2.7.3. SPI mode

The status of $\overline{CS}$ sets the mode of the SCP during a hardware and software reset. If $\overline{CS}$ is low during a reset the mode is SPI. It is important to note that $\overline{CS}$ should be low when either a hardware of software reset is issued to ensure the mode remains SPI.

For normal SPI operation SCL/SCK, $\overline{CS}$, CDIN, CDOUT and $\overline{REQ}$ are used. SCL/SCK is the serial clock input which is always driven by an external device. $\overline{CS}$ is the active low enable signal. CDIN is the control data input. CDOUT is the control data output. $\overline{REQ}$ is the active low request signal, which is driven low when there is valid data in the serial control port output SCPOUT register.

As an SPI compatible port, data is communicated on the CDIN and CDOUT pins and is clocked by the rising edge of SCL/SCK. $\overline{CS}$ is used to select the device on which the CDIN and CDOUT signals will be valid.

Figure 15 shows the relative timing necessary for an SPI write operation of a single byte. A 'write' is defined as the transfer of data from an SPI bus master to the CS4922 serial control port via CDIN. A transfer is initiated with $\overline{CS}$ being driven active low. This is followed by a 7 bit address and a read/write bit (set low for a write). For SPI mode, this address is typically not used, however it is still necessary to clock an address across the bus followed by the read/write bit.

If a write to the CS4922 is specified, 8 bits of data on CDIN will be shifted into the input shift register as shown in Figure 12. When the shift register is full, the 8 bit data is transferred to the Serial Control Port Input (SCPIN) register on the falling edge of the 8th data bit.

If the DSP core of the CS4922 wants to send a byte to the master, it first writes the byte to the Serial Control Port Output (SCPOUT) register. Note the DSP only sends 8 bits per transfer to the SCPOUT register. A write to the SCPOUT sets the request pin ($\overline{REQ}$) active low. The master must recognize the request and issue a read operation to the DSP. Figure 16 shows the relative timing of a single byte read. The master must send the 7 bit address (if address checking is enabled it must match the address in the SCPCN register) and the read bit. After the the falling edge of SCL/SCK for the read/write bit, the serial shift register is loaded with the byte to be sent and the most significant bit is placed on the CDOUT line.

The 8 bit value in the serial shift register is shifted out by the master. The data is valid on the rising edge of SCL/SCK and transitions immediately following the falling edge. For SPI, the $\overline{REQ}$ line will be de-asserted immediately following the rising

edge of the second to last data bit, of the current byte being transferred, if there is no data in the SCPOUT register. The $\overline{REQ}$ line is guaranteed to stay de-asserted (high) until the rising edge of the SCL/SCK for the last data bit. This signals the host that the transfer is complete.

If there is data placed in SCPOUT prior to the rising edge of SCL/SCK for the second to last data bit, then $\overline{REQ}$ will remain asserted (low). Immediately following the falling edge of SCL/SCK for the last data bit, the new data byte will be loaded into the serial shift register. The host should continue to read this new byte. It is important to note that once the data is in the shift register, clocks on the SCL/SCK line will shift the data bits out of the shift register. The host should read the byte prior to any other bus activity or the data will be lost. If $\overline{CS}$ is de-asserted SCK/SCL will not shift the data out. However the data is still in the shift register. Once $\overline{CS}$ becomes active (low) each SCL/SCK will shift the data out of the register.

If data is placed in SCPOUT after the rising edge of SCL/SCK for the second to last data bit, but before the rising edge of SCL/SCK for the last data bit, $\overline{REQ}$ will not be asserted until after the rising edge of SCL/SCK for the last data bit. This should be treated as a completed transfer. The data written to SCPOUT will not be loaded into the shift register on the falling edge of SCL/SCK for the last data bit. Therefore, a new read operation is required to read this byte.

2.8. User Definable Pins

The CS4922 has five pins which can be defined by the user, XF1-XF4 and PIO. The XF signals can be used as output pins. The PIO signal may be either an input or an output pin.

When active the XF4:XF1 bits in the LINT register are mapped directly to the XF pins. (For backwards compatibility with the CS4920A, XF1 can be accessed via the CM0 register as well.) An external pull-up (2.2 kΩ typical) is required for proper operation on each pin. An example application for this signal would be to use the pin as a request for audio data.

The PIO signal direction is configured by the I_O bit in the LINT register. When I_O = 0 the PIO signal is configured as an input. The I_O bit is zero

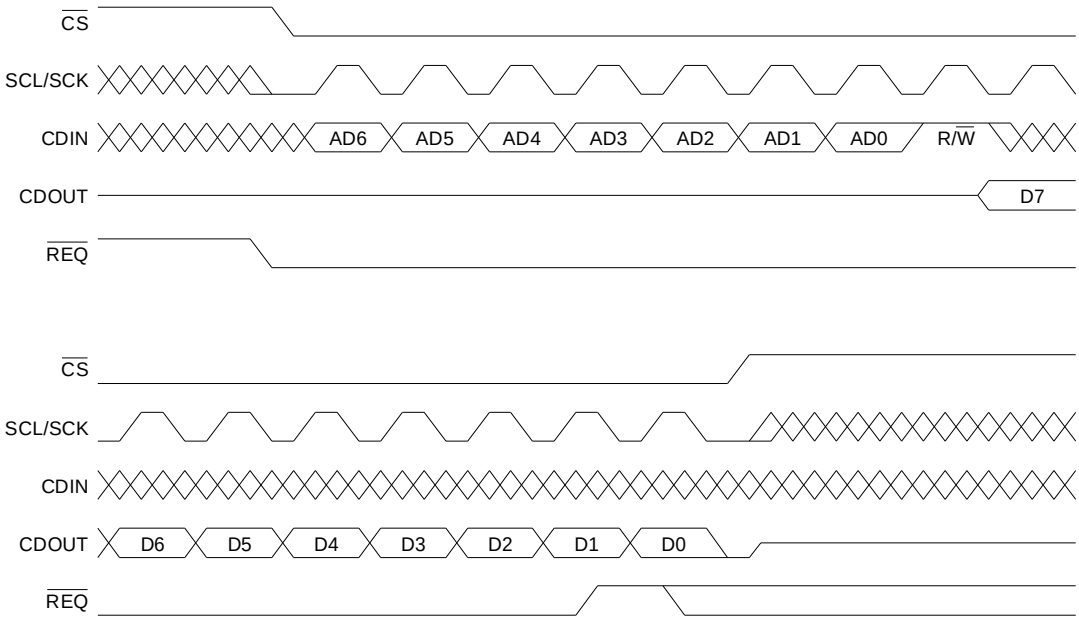


Figure 16. Control Port Timing, SPI Read

following the power up and any reset. As an input, the level on the PIO pin is mapped to the PIN bit of the LINT register. A rising edge of the PIN bit will generate a interrupt on line 3 if the PIEN bit is set to one. The LINT register must be read to clear this interrupt.

If the I_O bit is set to a logic one, the PIO pin is an output. A pull-down resistor (10 kΩ typical) is required for proper operation. As an output, the POUT bit of the LINT register is mapped directly to the PIO pin.

3. BOOT PROCEDURE

Program and data RAM must be loaded from external memory after power up or when a new program needs to be loaded. During the loading procedure (boot), data is transferred through the serial control port to program and data memory. This procedure is controlled by a program stored internally in ROM.

Following a power up or reset, the fast mode bit (FSTB) is cleared. This places the CS4922 into a 'fast mode'. While the serial control port (SCP) still conforms to the data format determined by $\overline{CS}$ on power up, the port can be operated at much higher bit rates to facilitate faster downloading of the DSP code. Once the code has been loaded the software can set the FSTB for normal communication in either SPI or I^2C. Since the CS4922 is always a slave this fast mode will not affect the operation of other devices sharing the same communication bus.

The boot procedure is initiated by a low to high transition of the reset ($\overline{RESET}$) pin with the BOOT pin high. This initializes the program counter to location 1000_H, the first location in ROM. After the ROM program transfers data from the control port to memory, it issues a software reset. This is done by writing a one to the RS (reset) bit in the control register. The software reset clears all registers including the program counter, which transfers control to the new program in RAM.

A hardware reset ($\overline{RESET}$ pin toggled low) has the same affect as a software reset. During the boot procedure, all interrupts, except the debug interrupt, are disabled.

The CS4922 will boot from a micro controller using the serial control port. When booting, it can communicate in an I^2C or SPI format. If the $\overline{CS}$ (chip select) pin is high when boot is initiated, the port will communicate in I^2C format. If the $\overline{CS}$ pin is low when boot is initiated, the port will communicate in SPI. Please refer to the timing requirements found at the beginning of this document.

Nodes in an I^2C network have unique network addresses. A message in an I^2C network consists of the address of the node receiving the message followed by the message data. When the control port is configured for I^2C format, it normally compares the address to an address stored in an internal register. During the boot procedure, the control port is programmed to ignore the address. The SCP section on I^2C operation explains the mechanics of writing to the CS4922.

The boot program in internal ROM expects data transferred through the control port to have the proper file format. The first two bytes contain the starting address for the following block of data. The starting address is 13 bits with the 13th bit specifying program or data memory. Therefore, the upper 3 bits of these two bytes are discarded internally. The second two bytes contain the length of the block of data. Successive bytes are concatenated into 24 bit words. These words are sequentially loaded into program and data memory beginning at the starting address.

Any number of blocks of data can be loaded. Two bytes containing FF and 3 bytes containing a check

sum must follow the last block of data. This check sum is generated by summing all the previous data, address, and length bytes and truncating to 24 bits. This check sum is compared to the value calculated internally. If they do not match, the $\overline{\text{REQ}}$ (request) pin is pulled low and the processor does not issue the software reset. It stays in a loop until boot is initiated again.

4. COMMON CS4922 INTERFACE ROUTINES

The following common CS4922 routines are modified excerpts from the CDB4922 evaluation board software. When reviewing these routines, keep in mind that timing requirements will change based on the performance of the host processor. Please refer to the "Switching Characteristics" , "Serial Control Port", and "Boot Procedure" sections in the CS4922 data sheet.

4.1. Boot Procedure Routine

The CS4922's on-board RAM is ready to be loaded after a RESET pulse while the BOOT pin is high. The host then sends the CS4922's microcode image to the CS4922's serial host port (the host may send the data in either I^2C or SPI mode, based on the state of the $\overline{\text{CS}}$ pin during the RESET pulse). Upon completion of download the CS4922's $\overline{\text{REQ}}$ pin should be checked to verify that the download was successful. If the REQ pin is high, then the download was good. (note: the $\overline{\text{REQ}}$ will remain high if no data is sent to the CS4922 or if the data is stopped. Therefore, for debugging purposes a microcode image with a bad checksum should be used to simulate a bad download and verify that the image is truly being sent successfully.)

```
/*
      Function    :        boot_4922()
      Return      :        None
      Arguments   :        None
      Description :        This function boots the CS4922.
*/

void boot_4922(void)
{
      BOOT(HIGH);
      pause(1);            /* hold 1 us */
      RESET(LOW);
      pause(50);           /* hold 50 us */
      RESET(HIGH);
      pause(1);            /* hold 1 us */
}
```

Figure 17. Sample pseudo C code for initializing the CS4922 for downloading.

```
CS(HIGH);                                                   /* Set the CS pin high */
                                                            /* Note: for I2C mode this pin should be kept high */

byte_load_count=0;
boot_4922();                                                /* toggle the lines to initiate download */
pause(2000);                                                /* delay 2 ms */

Send_I2C(START);                                            /* Send I2C start condition */
Write_Byte(ADDR4922 & WRITE);                               /* Send I2C address with R/W bit set to WRITE */
                                                            /* Note: I2C address is not checked, but still must be sent */
if (get_ACK())                                              /* Check if the CS4922 acknowledges the address byte sent */
{
      printf("error sending address byte for write condition \n");
      fclose(scp_in);
      return(1);
}
do
{
      if (fgets( &hexin, 80, scp_in ) != NULL)              /* Get the next byte in the microcode image file to send to
                                                            the CS4922 */
      {
                  sscanf( &hexin, "%2x", &val );
                  Write_Byte( val );                        /* Send the next byte to the CS4922 */

                  byte_load_count++;
                  if (get_ACK())                            /* Check if the CS4922 acknowledges the byte sent */
                  {
                              error = 1;
                              printf("Error transferring data on byte %d,
                              no ACK\n",byte_load_count);
                              break;
                  }
      }
} while ( !feof(scp_in));

Send_I2C(STOP);                                             /* Send I2C stop condition */

pause(2000);                                                /* give time (2ms) for 4922 to respond */

if (get_req() && !error)                                     /* Check the status of the CS4922's REQ pin */
      printf ("Download complete\n");                       /* REQ = high, PASS */
else
{
      printf("Download failure\n");                         /* REQ = low, FAIL */
}
```

Figure 18. Sample pseudo C code for downloading the CS4922 in I^2C Mode.

```
CS(LOW);                                        /* Set the CS pin low */
byte_load_count=0;
boot_4922();                                    /* toggle the lines to initiate download */
pause(2000);                                    /* delay 2 ms */
CS(HIGH);                                       /* Set the CS pin high */

wait(200);                                      /* delay 200 ns */

CS(LOW);                                        /* Set the CS pin low */
Write_Byte(ADDR4922 & WRITE);                   /* Send address with R/W bit set to WRITE */
                                                /* Note: address is not checked, but still must be sent */


do
{
      if (fgets( &hexin, 80, scp_in ) != NULL)  /* Get the next byte in the microcode image file to send to
                                                the CS4922 */
      {
                  sscanf( &hexin, "%2x", &val );
                  Write_Byte( val );            /* Send the next byte to the CS4922 */
                  byte_load_count++;

      }
} while ( !feof(scp_in));

pause(2000);                                    /* give time (2ms) for 4922 to respond */

CS(HIGH);                                       /* Set the CS pin high */

if (get_req() && !error)                        /* Check the status of the CS4922's REQ pin */
      printf("Download complete\n");            /* REQ = high, PASS */
else
{
      printf("Download failure\n");             /* REQ = low, FAIL */
}
```

Figure 19. Sample pseudo C code for downloading the CS4922 in SPI Mode.

4.2. Writing a Byte(s) to the Host Port

The host port of the CS4922 can be configured for I^2C or SPI mode. Writing a byte(s) to host port in I^2C mode begins with the host sending an I^2C start condition (SDA pin goes low while the SCL/SCK pin is high). The host must then send a 7-bit address followed by a R/W bit set to zero (the address must be sent even if the host port is not configured for address checking). The CS4922 will respond with a one bit acknowledge (ACK) after receiving the address and R/W bit. If the byte was received successfully, the ACK will be low. At this point the I^2C interface is configured to receive bytes from the host. The host may send multiple bytes (checking for ACKs between every byte). The write process is stopped by the host sending an I^2C stop condition (SDA pin goes high while the SCL/SCK pin is high).

Writing a byte(s) to the host port in SPI mode begins with the host setting the $\overline{CS}$ pin of the CS4922 low. The host must then send a 7-bit address followed a R/W bit set to zero (the address must be sent even if the host port is not configured for address checking). At this point the SPI interface is configured to receive bytes from the host. The host may send multiple bytes. The write process is stopped by the host raising the $\overline{CS}$ pin of the CS4922.

```
SCL(HIGH);          /* Set SCL in default state */
SDA(HIGH);          /* Set SDA in default state */

CS(LOW);

pause(5);           /* hold for 5 us */

Write_Byte(ADDR4922 & WRITE);

Write_Byte(first_byte);

Write_Byte(second_byte);

Write_Byte(third_byte);
                    .
                    .
                    .
CS(HIGH);
```

Figure 20. Sample pseudo C code for writing bytes to the CS4922 in SPI Mode.

```
SCL(HIGH);          /* Set SCL in default state */
SDA(HIGH);          /* Set SDA in default state */
Send_I2C(START);
pause(5);           /* hold for 5 us */
Write_Byte(ADDR4922 & WRITE);
if(get_ACK())
{
      printf("error sending address byte for write condition\n");
      return(1);
}

Write_Byte(first_byte);
if(get_ACK())
{
      printf("error sending byte \n");
      Send_I2C(STOP);
      return(ERROR);
}

Write_Byte(second_byte);
if(get_ACK())
{
      printf("error sending byte \n");
      Send_I2C(STOP);
      return(ERROR);
}

Write_Byte(third_byte);
if(get_ACK())
{
      printf("error sending byte \n");
      Send_I2C(STOP);
      return(ERROR);
}
                    .
                    .
                    .
Send_I2C(STOP);
```

Figure 21. Sample pseudo C code for writing bytes to the CS4922 in I^2C Mode.

```
/*---------------------------------------------------------------------*/
/*
      Function    :     Send_I2C()
      Return      :     None
      Arguments   :     select, tells the function if its a start or stop condition
      Description :     This function is used to begin or end an I2C data transfer.
*/

void Send_I2C(char select)

{

      if (select==START)
      {
            SDA(LOW);                     /* drive SDA low while SCL is high for start condition */
            pause(5);                     /* hold for 5 us */
            SCL(LOW);                     /* drive SCL low for writing address */
            pause(5);                     /* hold for 5 us */
      }
      else  /* STOP condition */
      {
            SDA(LOW);                     /* make sure SDA is low */
            pause(5);                     /* hold 5 us */
            SCL(HIGH);                    /* drive SCL high */
            pause(5);                     /* hold 5 us */
            SDA(HIGH);                    /* drive SDA for the STOP condition */
      }

}
```

Figure 22. Sample pseudo C code for sending an I^2C start/stop condition.

```
/*---------------------------------------------------------------------*/
/*
      Function    :     get_ACK()
      Return      :     returns 1 if I2C bus fails to ACK else
                        returns 0
      Arguments   :     None
      Description :     The get_ACK function will send the 9th clock of the SCL line and verify the slave has
                        acknowledged the transfer by driving the SDA line low.
*/

char get_ACK(void)

{
      char ack;

      SCL(HIGH);                          /* latch the ACK */

      /* Read the SDA pin */
      ack = (char) (((inportb(gbStat_reg)) & SDA_POS)>>SDA_BIT);

      SCL(LOW);

      return(ack);

}
```

Figure 23. Sample pseudo C code for reading an I^2C ACK.

```
/*---------------------------------------------------------------------*/
/*
      Function    :        Write_Byte()
      Return      :        None
      Arguments   :        char write_byte
      Description :        this function will write write_byte onto the I2C lines MSB first
*/

void Write_Byte(char write_byte)
{
      int  bit_number;

      for (bit_number=7;bit_number>=0;bit_number--)
      {
            if((write_byte>>bit_number)&0x01)      /* check each bit to write */
                  SDA(HIGH);                       /* Send a one */
            else
                  SDA(LOW);                        /* Send a zero */

            SCL(HIGH);                             /* latch the bit into the CS4922 */
            pause(5);                              /* hold for 5 us */
            SCL(LOW);                              /* Set the clock line back low */

      }

      SDA(HIGH);     /*release bus so 4922 can ACK*/
}
```

Figure 24. Sample pseudo C code for writing a byte to the I^2C CS4922 host port.

```
/*---------------------------------------------------------------------*/
/*
      Function    :        Write_Byte()
      Return      :        None
      Arguments   :        char write_byte
      Description :        this function will write write_byte onto the SPI lines MSB first
*/

void Write_Byte(char write_byte)
{
      int  bit_number;

      for (bit_number=7;bit_number>=0;bit_number--)
      {
            if((write_byte>>bit_number)&0x01)      /* check each bit to write */
                  CDIN(HIGH);                      /* Send a one */
            else
                  CDIN(LOW);                       /* Send a zero */

            SCL(HIGH);                             /* latch the bit into the CS4922 */
            pause(5);                              /* hold for 5 us */
            SCL(LOW);                              /* Set the clock line back low */

      }

      CDIN(HIGH);     /*return data line to default state*/
}
```

Figure 25. Sample pseudo C code for writing a byte to the SPI CS4922 host port.

4.3. Reading a Byte(s) from the Host Port

Reading a byte(s) from the host is initiated by the CS4922 setting the $\overline{REQ}$ pin low. The $\overline{REQ}$ pin will remain low until the data is read from the host port of the CS4922. When the host port is configured for I^2C mode, the host will initialize the CS4922's serial port for a read operation by sending an I^2C start condition followed by a 7-bit address and the R/W bit set to one. The host then will read 8-bits of data from the CS4922, check the status of the $\overline{REQ}$ pin, and then send an I^2C ACK. If the $\overline{REQ}$ pin was low, then the CS4922 has another byte pending and the host must read the next byte (if the host instead sends an I^2C stop condition, then the next byte will be lost). The host must send an I^2C stop condition if the status of the $\overline{REQ}$ pin was high (the $\overline{REQ}$ pin may go low again after the host sends the ACK, but the host must still send an I^2C stop condition).

For an SPI read, the host will initialize the CS4922's serial port for a read operation by setting the $\overline{CS}$ pin low followed by sending 7-bits of address and the R/W bit set to one to the CS4922's host port. The host then will read 7-bits of data from the CS4922, check the status of the $\overline{REQ}$ pin, and then read the 8^{th} bit of data. If the $\overline{REQ}$ pin was low, then the CS4922 has another byte pending and the host must read the next byte (if the host instead raises the $\overline{CS}$ pin, then the next byte will be lost). After reading the 8^{th} bit the host must raise the $\overline{CS}$ pin to a high state if the status of the $\overline{REQ}$ pin was high (the $\overline{REQ}$ pin may go low again after the host reads the 8^{th} bit, but the host must still raise the $\overline{CS}$ pin).

```
while ( get_req() && !kbhit());      /* Wait until REQ pin goes low */

if (!get_req())
      SCP_Read();                    /* read stream of output */
```

Figure 26. Sample pseudo C code for reading bytes from the CS4922 host port.

```
char SCP_Read( void )
{

      unsigned char DataIn = 0;
      char notendofread = 1;

      /* Setup port for I2C read */
      Send_I2C(START);

      Write_Byte(ADDR4922 | READ);

      if (get_ACK())
      {
                  printf("error sending address byte for read condition \n");
                  return(ERROR);
      }

      /* Read data until REQ goes high */
      do
      {
                  notendofread = Read_Byte(&DataIn);
                  printf("%02x ", DataIn );
       }while (notendofread);

      return(0);
}
```

Figure 27. Sample pseudo C code function for handling multi-byte reads in I^2C mode from the CS4922 host port.

```
char SCP_Read( void )
{

        unsigned char DataIn = 0;
        char notendofread = 1;

        /* Setup port for SPI read */
        CS(LOW);

        Write_Byte(ADDR4922 | READ);

        /* Read data until REQ goes high */
        do
        {
                        notendofread = Read_Byte(&DataIn);
                        printf("%02x ", DataIn );
         }while (notendofread);

        return(0);
}
```

Figure 28. Sample pseudo C code function for handling multi-byte reads in SPI mode from the CS4922 host port.

```
        /*-------------------------------------------------------------------*/
/*
        Function    :       Read_Byte()
        Return      :       returns flag to indicate if a STOP condition has been sent. Returns 1 if no STOP has been sent, 0 if
                            STOP has been sent.
        Arguments   :       *int -> address of byte to be read from the CS4922 SCP port.
        Description :       This function reads a byte of data  from the SCP port and replies with an ACK if there is still more
                            data to be read or a stop condition if the REQ is high after the last data bit.
*/

char Read_Byte(unsigned char *DataIn)
{
        int bit_number;
        int end_of_data = 0;

        *DataIn=0;
        for (bit_number=0; bit_number<8; bit_number++)
        {
                SCL(HIGH);

                if (bit_number == 7)                    /* is this the 8th bit ? */
                if (get_req())                          /* Yes, is the REQ line low ? */
                                                        end_of_data = 1; /* Yes, stop reading data */
        /* Read the data from the port mapped to gbStat_reg one bit at a time */
                *DataIn |= (((inportb(gbStat_reg) & CDOUT_POS) >> 4) << (7 - bit_number));

                SCL(LOW);
        }

        if (end_of_data)
        {
                Send_I2C(STOP);
                return(0);
        }
        else
        {
                Send_ACK();
                return(1);
        }
}
```

Figure 29. Sample pseudo C code function for handling a byte read in I^2C mode from the CS4922 host port.

```
/*-------------------------------------------------------------------*/
/*
        Function        :       Read_Byte()
        Return          :       returns flag to indicate if a CS has been changed. Returns 1 CS remains low, 0 if CS has set high.
        Arguments       :       *int -> address of byte to be read from the CS4922 SCP port.
        Description     :       This function reads a byte of data  from the SCP port. If the REQ is high after the second to last
                                data bit the function will raise the CS .
*/


char Read_Byte(unsigned char *DataIn)
{
        int bit_number;
        int end_of_data = 0;

        *DataIn=0;
        for (bit_number=0; bit_number<8; bit_number++)
        {
                        SCL(HIGH);

                        if (bit_number == 6)                            /* is this the 7th bit ? */
                        if (get_req())                                  /* Yes, is the REQ line low ? */
                                                                        end_of_data = 1; /* Yes, stop reading data */
        /* Read the data from the port mapped to gbStat_reg one bit at a time */
                        *DataIn |= (((inportb(gbStat_reg) & CDOUT_POS) >> 4) << (7 - bit_number));

                        SCL(LOW);
         }

        if (end_of_data)
        {
                        CS(HIGH)
                        return(0);
        }
        else
                        return(1);

}
```

Figure 30. Sample pseudo C code function for handling a byte read in SPI mode from the CS4922 host port.

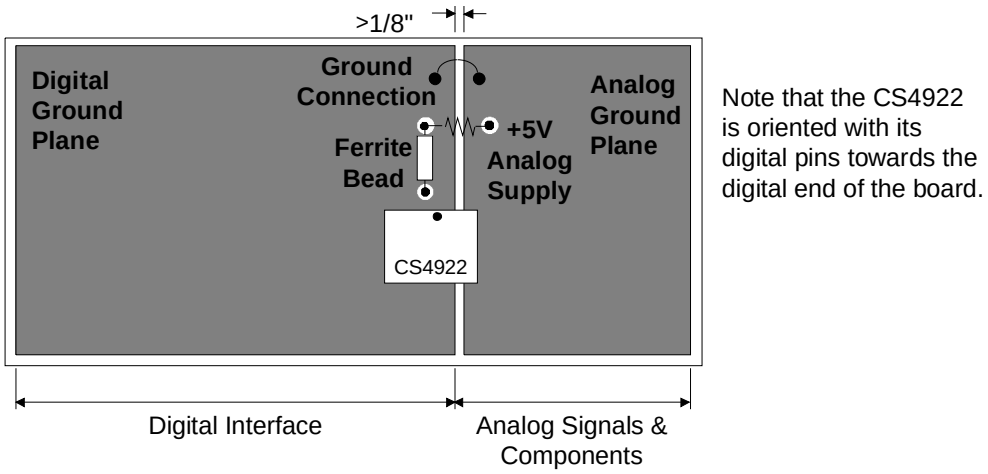


Figure 31. CS4922 Suggested Layout

5. POWER SUPPLY AND GROUNDING

When using separate supplies, the digital power should be connected to the CS4922 via a ferrite bead, positioned closer than 1" to the device (see Figure 31). The CS4922 VA+ pin should be derived from the cleanest power source available. If only one supply is available, use the suggested arrangement in Figure 1.

The CS4922 should be positioned such that the analog pins (pins 29 - 39) are over the analog ground plane, while the rest of the pins lay over the digital ground plane as illustrated in Figures 31 and 32. The analog and digital grounds on the CS4922 are not connected internally; this should be accomplished externally through a point-to-point connection across the ground split as shown in Figure 31. A separate power plane for the chip is preferable.

Figure 32 illustrates the optimum ground and decoupling layout for the CS4922 assuming a surface-mount socket and surface mount capacitors. Surface-mount sockets are useful since the pad locations are exactly the same as the actual chip; therefore, given that space for the socket is left on the board, the socket can be optional for production. Figure 32 depicts mostly the top layer containing signal traces and assumes the bottom or interlayer contains a solid ground plane (analog or digital), except where the digital supply needs to run to the power pins. The important points with regards to this diagram are that the ground plane is SOLID under the CS4922 and connects all ground pins with thick traces providing the absolute lowest impedance between ground pins. The decoupling capacitors are placed as close as possible to the device which, in this case, is the socket boundary. The lowest value capacitor is placed closest to the chip. Vias are placed near the AGND and DGND pins, under the IC, and should be attached to the solid ground plane (analog or digital) on another layer. The negative side of the decoupling capacitors should also attach to the same solid ground plane. Traces bringing the power to the CS4922 should be wide thereby keeping the impedance low.

If using through-hole sockets, effort should be made to find a socket with the minimum height which will minimize the socket impedance. When using a through-hole socket, the vias under the chip in Figure 32 are not needed since the pins serve the same function.

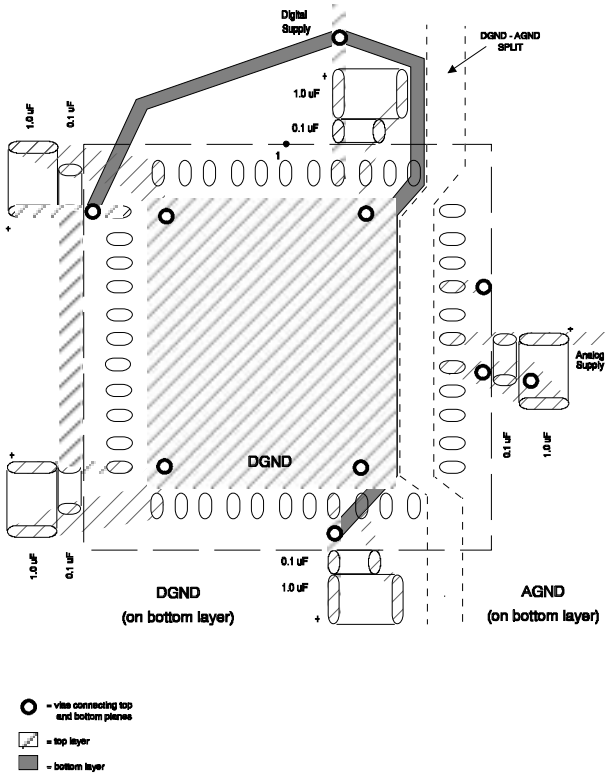


Figure 32. CS4922 Surface Mount Decoupling Layout

6. DAC FILTER RESPONSE PLOTS

Figures 33 through 36 show the overall frequency response, passband ripple and transition band for the CS4922 DACs. Figure 36 shows the DACs' deviation from linear phase. Fs is the selected sample frequency. Since the sample frequency is programmable, the filters will adjust to the selected sample frequency. Fs is also the FSYNC frequency

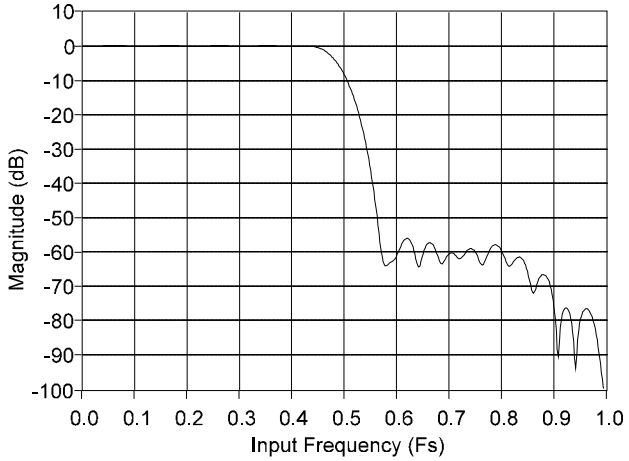


Figure 33. DAC Frequency Response

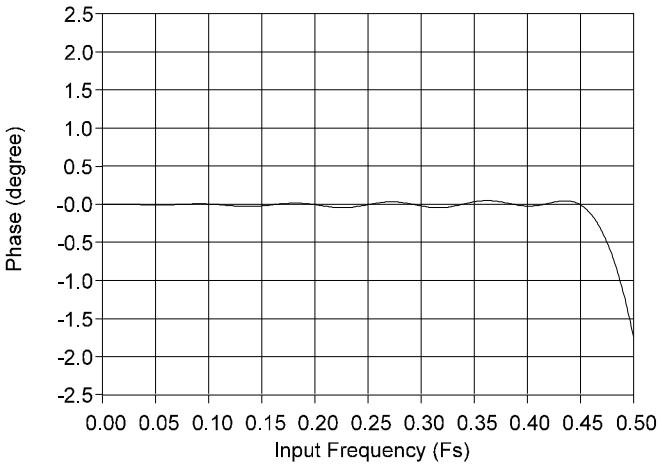


Figure 34. DAC Phase Response

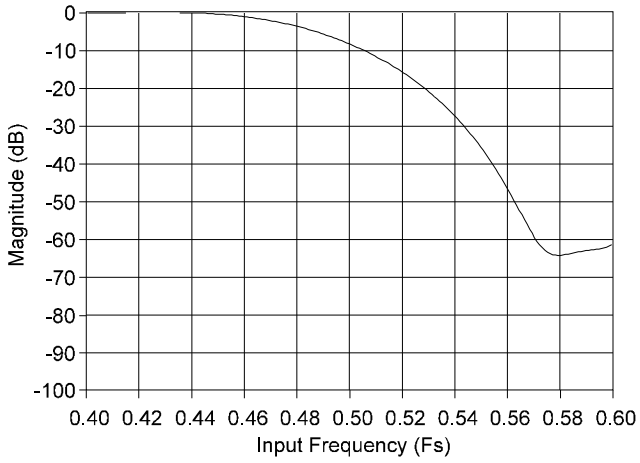


Figure 35. DAC Transition Band

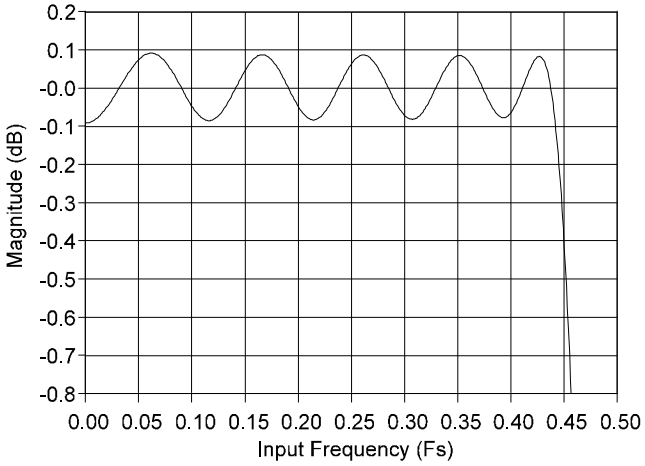


Figure 36. DAC Passband Ripple

7. REGISTER DESCRIPTIONS

Address	Register Name	Register Description
00000b	DAC	Digital to Analog Converter Output Register
00001b		Reserved
00010b	ASI	Audio Serial Input Register
00011b	ASICN	Audio Serial Input Control Register
00100b	SCPIN, SCPOUT	Serial Control Port Data I/O Register
00101b	SCPCN	Serial Control Port Control Register
00110b	CM0	Clock Manager Control Register 0
00111b	CM1	Clock Manager Control Register 1
01000b	XMT	Digital Audio Transmitter Data Register
01001b	XMTCS	Digital Audio Transmitter Channel Status Register
01010b	XMTCN	Digital Audio Transmitter Control Register
01011b	LINT	Long Interrupt Register
01100b	DBPIN, DBPOUT	Debugger Data I/O Register
01101b	DBPST	Debugger Data Status Register
01110b	CNT24	Playback Timing Clock [31-9]
01111b	CNT9	Playback Timing Clock [8-0]
10000b	STATUS	DSP Status Register
10001b	SHADOW	DSP Shadow Status Register
10010b	AUXOUT/AUXIN	Auxiliary Audio Port Data Register
10011b	AUXCN	Auxiliary Audio Port Control Register

Table 2. Control Register Map

7.1. Digital to Analog Converter Output Register

Address 00000b DAC

Bit Number	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	Reserved

Bit	Mnemonic	Function
23:8	DAC	Left & Right 16-bit DAC output register Value: 16-bit 2's complement data
7:0	-	Reserved

7.2. Audio Serial Input Register

Address 00010b ASI

Bit Number	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Mnemonic	Function
23:0	ASI	The contents of the audio serial input shift register are loaded into the ASI by transitions of FSYNC or by a shifter count rollover. *Value:* MSB first serial data

7.3. Audio Serial Input Control Register

Address 00011b ASICN

Bit Number	23	22	21	20	19	18	Reserved

Bit	Mnemonic	Function
23	EDG	Selects which edge of SCLK to use for clocking SDATA *Value:* 0= Sample on falling edge 1= Sample on rising edge
22	POL	In non-pulse mode with PUL low, POL determines LR polarity of FSYNC. When PUL is high, the POL bit determines the valid edge of FSYNC. *Value:* PUL=0 0= FSYNC high 1= FSYNC low PUL=1 0= FSYNC rising edge samples SDATA 1= FSYNC falling edge samples SDATA
21	DEL	Specifies a delay in sampling SDATA following edge of FSYNC *Value:* 0= No delay 1= Delay of one SCLK period
20	PUL	SDATA Pulse Mode Control *Value:* 0= FSYNC is the R/L signal for SDATA 1= FSYNC identifies the start of a frame, but does not distinguish between left and right samples.
19	CNT16	Selects the ASI input word length (shift count length) *Value:* 0= 24 Bits 1= 16 Bits
18	FSI	FSYNC internal *Value:* If POL=0 and FSYNC pin is tied low, this bit can be used to emulate FSYNC's function internally.
17:0	-	Reserved

7.4. Serial Control Port Data I/O Register

Address 00100b SCPIN, SCPOUT

Bit Number	Reserved	7	6	5	4	3	2	1	0

Bit	Mnemonic	Function
23:8	-	Reserved
7:0 7:0	SCPIN SCPOUT	Serial control port data input register Serial control port data output register

7.5. Serial Control Port Control Register

Address 00101b SCPCN

Bit Number	23-17	16	15	14	13	12	11	10	Reserved

Bit	Mnemonic	Function
23:17	ADDR	Seven bit programmable address of the audio decoder
16	AEN	Enables Control Port address checking *Value:* 0= Address checking disabled 1= Address checking enabled, message address compared to value of ADDR
15	ORIEN	Data output ready interrupt enable *Value:* 0= SCP ORDY interrupt disabled 1= SCP ORDY interrupt enabled
14	IRIEN	Data input ready interrupt enable *Value:* 0= SCP IRDY interrupt disabled 1= SCP IRDY interrupt enabled
13	ORDY	Output ready status bit (read only) *Value:* 0= SCPOUT full or not ready 1= Empty, ready for new byte
12	IRDY	Input ready status bit *Value:* 0= SCPIN empty, not ready 1= SCPIN full, ready for input to DSP
11	MO	Mode control bit *Value:* 0= I^2C 1= SPI
10	FSTB	Fast mode control bit *Value:* 0= Fast mode serial control (default at reset) 1= Slow mode serial control
9:0	-	Reserved

7.6. Clock Manager Control Register Zero

Address 00110b CM0

Bit Number	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9:0

Bit	Mnemonic	Function
23	QBP	Reserved (always = 0)
22	PM	Phase lock loop backward compatibility switch adjusts the feedback in the PLL. Allows for CS4920A CM1 values. *Value:* 0 = Old mode (CS4920A compatible) 1 = New mode
21	MUTE	Controls muting on mono analog output *Value:* 0 = Mono output unmute 1 = Analog mute mono
20	MONOOUT	Controls mono analog output *Value:* 0 = Mono output held at approximately 2.1 volts 1 = Mono output enabled
19	HO	Half rate DSP selector, bit can be changed without breaking PLL lock *Value:* 0 = DSP and DAC clocks run at full speed 1 = DSP and DAC clocks run at half speed
18	HFS	Half sample rate *Value:* 0 = DSP runs at 4x DAC clock frequency 1 = DSP runs at 8x DAC clock frequency
17	DIV	System counter clock select *Value:* 0 = 33-bit counter clocked from 90_CLK 1 = 33-bit counter clocked from CLKIN/300
16	XF1	External flag, mapped to XF1 pin, This bit location is maintained for software compatibility with the CS4920A *Value:* New applications should access this feature through the LINT register
15	SCREN	SCR counter enable *Value:* 0 = 33-bit counter disabled 1 = 33-bit counter enabled
14	FS	Read only status bit which is a 50% duty cycle sample rate clock
13	DACEN	DAC enable *Value:* 0 = DAC output held at approximately 2.1 volts 1 = DAC output enabled
12	SLW	Slow control bit for phase locked loop *Value:* 0 = The internal VCO of PLL is pulled to minimum output frequency 1 = The internal VCO of PLL is allowed to run nominal rate
11	FSRS	Sample rate clock reset . Can be used to synchronize internal Fs with external events. *Value:* 0 = Internal Fs allowed to run 1 = Internal Fs held at reset
10	P	Over-sample ratio rate selection of the DAC's *Value:* 0 = DAC rate = 128Fs 1 = DAC rate = 192Fs
9:0	Q	Divide ratio between the DSP clock and CLKOUT *Value:* 10-bit divider

7.7. Clock Manager Control Register One

Address 00111b CM1

Bit Number	23	22	21:10	9:0

Bit	Mnemonic	Function
23	QFS	Quarter sample flag, provides faster DSP clocks for slower sample rates, only active when HFS = 1 *Value:* 0 = DSP runs at 4x (HFS = 0) or 8x (HFS = 1) the DAC clock frequency 1 = DSP runs at 16x the DAC clock frequency
22	LDS	Reserved for factory test
21:10	M	12 Bit Value used to divide the CLKIN input
9:0	N	10-bit multiplier value used in the feedback path of the PLL.

7.8. Digital Audio Transmitter Data Register

Address 01000b XMT

Bit Number	23:0

Bit	Mnemonic	Function
23:0	XMT	24-bit audio output register. Left and right audio data written to this register is formatted to meet IEC958 standards

7.9. Digital Audio Transmitter Channel Status Register

Address 01001b XMTCS

Bit Number	23:8	

Bit	Mnemonic	Function
23:8	XMTCS	16 Bit value or multiple 16-Bit value for insertion into channels status bits of IEC958 transmition stream. Has two modes of operation determined by XMTCN19; XMTCN19 low equals consumer mode only, XMTCN19 high supports full programmability. *Value:* XMTCN19 = 0, Consumer mode only, XMTCS read once per Block. Only most important status bits supported: 0-5, 8-15, 24 and 25. XMTCS8 is logical command status bit 0. XMTCN19 = 1, XMTCS must be reloaded every 16 sub-frames. Supports all channel status bits
7:0	-	Reserved

7.10.Digital Audio Transmitter Control Register

Address 01010b XMTCN

Bit Number	23	22	21	20	19	18	17	16	15	

Bit	Mnemonic	Function
23	V	Validity Bit *Value:* Bit 28 of Sub-Frame [31:0]
22	U	User Bit *Value:* bit 29 of Sub-Frame [31:0]
21	DADR	Map DAC I/O address onto XMT register *Value:* 0 = XMT only responds to XMT data register 1 = XMT also responds to DAC output
20	OE	Output enable *Value:* 0 = TX output disabled 1 = TX output enabled
19	CSMD	Channel status mode *Value:* 0 = XMTCS is read once per block 1 = XMTCS is read after each 16 subframes
18	BLKST	Block start *Value:* A low to high transition specifies a new channel status block boundary
17	TSTP	Test mode *Value:* Test mode only, must be set to zero for normal operation
16	TSTDAC	Test bit - Test DAC *Value:* When high, digital output from DAC is output through TX. Normal transmission disabled
15	DACLRB	Test bit - Left/Right bit of DAC *Value:* Left/Right bit of DAC. Special test bit intended for test purposes only
14:0	-	Reserved

7.11.Long Interrupt Register

Address 01011b LINT

Bit Number		19:16	15:13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Mnemonic	Function
23:20	-	Reserved
19:16	XF4:XF1	External flag control bits mapped to external XF4-XF1 device pins
15:13	RESERVED	Always low
12	PTST	Phased locked loop test bit *Value:* 0 = XF2 Pin = LINT17 1 = XF2 Pin = PLL lock status
11	I_O	PIO output control bit *Value:* 0 = PIO is input, value @ I_O LINT[8] 1 = PIO is an output
10	POUT	PIO output control bit *Value:* Output bit to PIO pin when PIO set as output
9	PIEN	PIO interrupt enable, If set, a rising edge of the PIO pin bit will generate an interrupt. The LINT register must be read to clear this interrupt *Value:* 0 = PIO interrupt disabled 1 = PIO interrupt enabled
8	PIN	PIO external pin input status bit if I_O LINT [11] is set to 0
7	RJIEN	Reject SCPIN interrupt enable *Value:* 0 = Interrupt from SCPIN REJ disabled 1 = Interrupt level 3 generated with low to high transition of REJ
6	REJ	SCPIN reject status. Read only. High when input data rejected
5	BYIEN	XMT port output channel status BYTCK interrupt *Value:* 0 = Interrupt from BYTCK disabled 1 = Interrupt from BYTCK enabled
4	BYTCK	XMT port output channel byte clock. Toggles every 8 sub-frames. If BYIEN high, Interrupt 3 generated on rising edge of BYTCK. *Value:* Completes a cycle every 16 sub-frames
3	CBIEN	XMT port output channel status block interrupt enable *Value:* 0 = Interrupt from CBL disabled 1 = Interrupt from CBL enabled
2	CBL	XMT port channel status block clock. CBL status bit goes high at the beginning of a channel status block boundary and low 64 sub-frames later, generates interrupt 3 on rising edge if CBIEN set high. *Value:* Completes a cycle every 192 frames
1	LKIEN	PLL lock broken interrupt enable - Interrupt 3 occurs if LKIEN = 1 and rising edge of lock LINT[0] occurs *Value:* 0 = Lock interrupt disable 1 = Lock interrupt enable
0	LOCK	On-chip phase locked loop status bit *Value:* 0 = PLL is locked 1 = PLL is un-locked

7.12.Debugger Data I/O Register

Address 01100b DBPIN, DBPOUT

Bit Number		7:0

Bit	Mnemonic	Function
23:8	-	Reserved
7:0	DBPIN	8-bit debugger port data input register *Value:* Written serially by host, Read byte by DSP
7:0	DBOUT	8-bit debugger port data output register *Value:* Write byte by DSP, Read serially by host

7.13.Debugger Data Status Register

Address 01101b DBPST

Bit Number		13	12	

Bit	Mnemonic	Function
23:14	-	Reserved
13	ORDY	Debug port output ready flag for data byte reads from the DSP. *Value:* 1 = DBPOUT Data register is empty 0 = DBPOUT Data register is not empty
12	IRDY	Debug port input ready flag for data byte writes from the host to the DSP. *Value:* 1 = Input data register loaded, Byte data valid 0 = No valid input byte available
11:0	-	Reserved

7.14.Playback Timing Clock MSB

Address 01110b CNT24

Bit Number	23:0

Bit	Mnemonic	Function
23:0	STC_MSB	Upper 24-bits of 33-bit playback reference clock *Value:* Loaded by DSP & clocked from pin 19 90_CLK or pin 27 CLKIN

7.15.Playback Timing Clock LSB

Address 01111b CNT9

Bit Number		9	8:0

Bit	Mnemonic	Function
23:10	-	Reserved
9	VALID	STC clock valid signal *Value:* 1 = 33-bit counter value stable for next two instructions for a valid read 0 = 33-bit counter value not stable for a read
8:0	STC_LSB	Lower 9-bits of 33-bit playback reference clock *Value:* Loaded by DSP & clocked from pin 19 90_CLK or pin 27 CLKIN

7.16.DSP Status Register

Address 10000b STATUS

Bit Number		7:5	4	3	2	1	0

Bit	Mnemonic	Function
23:8	-	Reserved
7:5	STACK PTR	DSP core stack pointer
4	N	DSP core negative flag
3	Z	DSP core zero flag
2	V	DSP core overflow flag
1	U	DSP core unnormalized flag
0	C	DSP core carry flag

7.17.DSP Shadow Status Register

Address 10001b SHADOW

Bit Number		4	3	2	1	0

Bit	Mnemonic	Function
23:5	-	Reserved
4	SN	Shadow of DSP core negative flag
3	SZ	Shadow of DSP core zero flag
2	SV	Shadow of DSP core overflow flag
1	SU	Shadow of DSP core unnormalized flag
0	SC	Shadow of DSP core carry flag

7.18.Auxiliary Audio Port Data Register

Address 10010b AUXOUT/AUXIN

Bit Number	23:0	19:0

Bit	Mnemonic	Function
23:9	AUXOUT	Auxiliary audio port data output register, DSP write *Value:* AUXDAT[23] is MSB, LSB determined by AUXCN[10-9]
19:0	AUXIN	Auxiliary audio port data input register, DSP read *Value:* AUXDAT[0] is LSB, MSB determined by AUXCN[8-7]

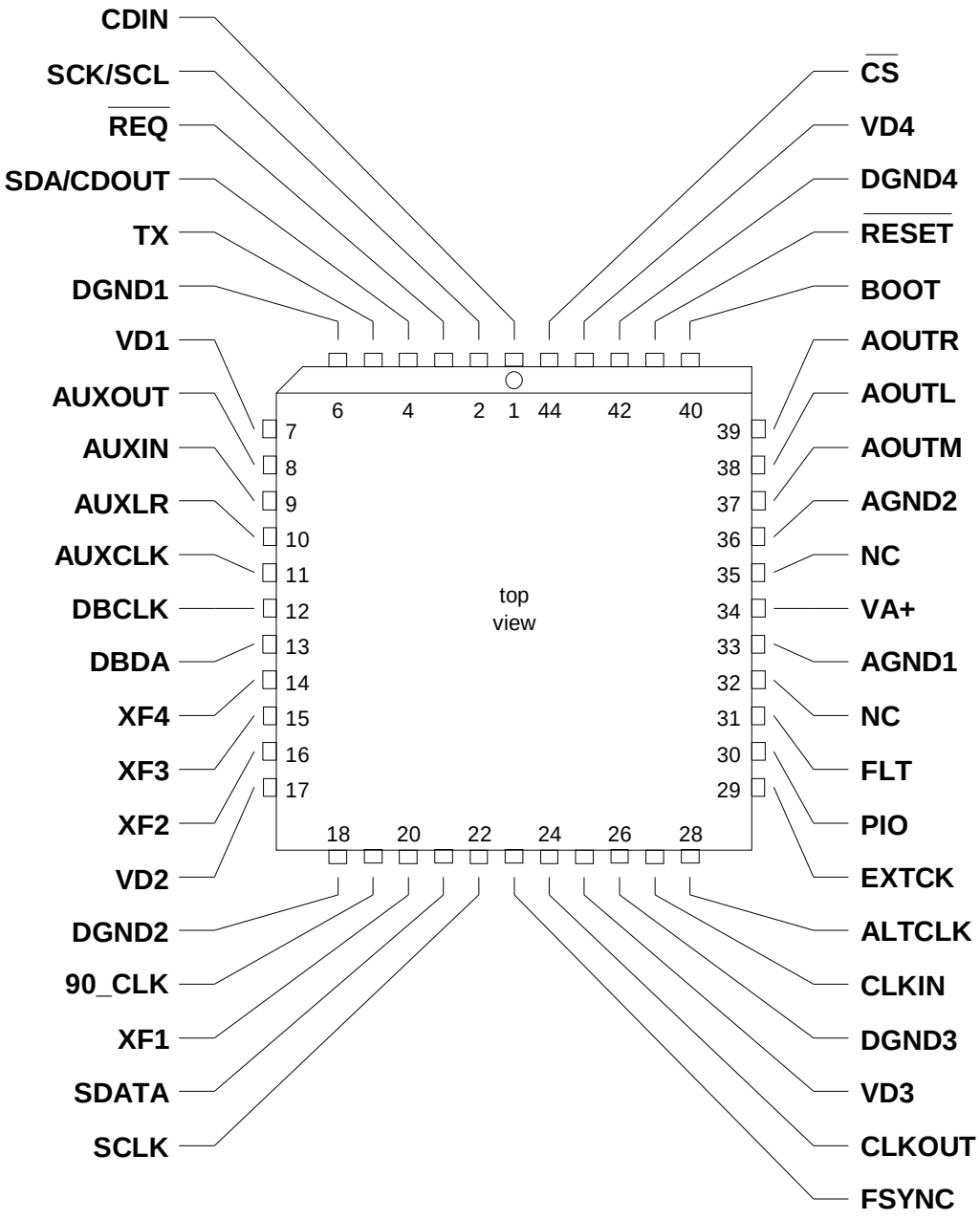
7.19.Auxiliary Audio Port Control Register

Address 10011b AUXCN

Bit Number		12	11	10:9	8:7	6:5	4	3	2	1:0

Bit	Mnemonic	Function
23:13	-	Reserved
12	AUXCKEN	Auxiliary audio port clock output enable *Value:* 0 = AUXCLK held low 1 = AUXCLK enabled
11	DADR	Auxiliary data output selection control *Value:* 0 = AUXDAT responds to its own I/O address 1 = AUXDAT also responds to dac I/O address
10:9	OUTBTN	Auxiliary output sample resolution selection *Value:* 00 = 16 bits per sample 01 = 18 bits per sample 10 = 20 bits per sample
8:7	INBTN	Auxiliary input sample resolution selection *Value:* 01 = 18 bits per sample 10 = 20 bits per sample
6:5	OUTCHN	Auxiliary output channel count selection *Value:* 00 = 2 Channels 01 = 4 Channels 10 = 6 Channels
4	INCHN	Auxiliary input channel count selection *Value:* 0 = 2 Channels 1 = 4 Channels
3	JSTF	Auxiliary sample AUXLR justification selection for non-I^2S modes. *Value:* 0 = Auxiliary samples right justified to AUXLR 1 = Auxiliary port left justified to AUXLR
2	I2SM	Auxillary port I2S mode selection *Value:* 0 = Auxiliary port not I2S compatible 1 = Auxiliary port I2S compatible
1:0	SCLKR	Auxiliary port SCLK rate *Value:* 00 = SCLK @ 32FS 01 = SCLK @ 64FS 10 = SCLK @ 128FS

8. PIN DESCRIPTIONS



8.0.1. Power Supplies

VD1, VD2, VD3, VD4 - Positive Digital Power Supply, PINS 7, 17, 25, 43.

The +5V supply is connected to these pins to power the various digital subcircuits on the chip. See decoupling section in this data sheet for decoupling recommendations.

DGND1, DGND2, DGND3, DGND4 - Digital Ground, PINS 6, 18, 26, 42.

Digital power supply ground.

VA+ - Positive Analog Power Supply, PIN 34.

The analog +5V supply for the analog-to-digital converter and the PLL. Analog performance is highly dependent on the quality of this supply. See decoupling section in this data sheet for decoupling recommendations.

AGND1, AGND2 - Analog Ground, PIN 33, 36.

Analog power supply ground.

8.0.2. Digital-to-Analog Converter

AOUTL, AOUTR - Analog Outputs, Left and Right Channels, PINS 38, 39.

These DAC outputs are centered at approximately 2.2V. An external filter is required to diminish out-of-band noise. See Typical Connection Diagram, Figure 1.

AOUTM - Mono Analog Output, PIN 37.

Mono is the summation of AOUTL and AOUTR. Mono output is 180° out-of-phase with the sum of AOUTL and AOUTR. Mono is centered at approximately 2.2V. An external filter is required to diminish out-of-band noise. See Typical Connection Diagram, Figure 1.

8.0.3. Serial Audio Port

FSYNC - Frame Synchronization Clock Input, PIN 23.

FSYNC transitions delineate left and right audio data, or the start of a data frame, as determined by the PUL bit in the ASICN. The edge definition is determined by the POL bit.

SCLK - Serial Clock Input, PIN 22.

SCLK is used to clock the serial audio data on SDATA into the device. The EDG bit in the ASICN determines the active edge. The DEL bit can be used to delay data by one SCLK after an FSYNC transition.

SDATA - Serial Audio Data Input, PIN 21.

Audio data input to SDATA is clocked into the device by SCLK.

8.0.4. Digital Audio Transmitter

TX - Transmitter Output, PIN 5.

Biphase mark encoded data is output at logic levels from the TX pin. This output typically connects to the input of an RS-422 or optical transmitter. With additional external circuitry, the port can support either AES/EBU or S/PDIF formats.

8.0.5. Clock Manager

CLKOUT - Clock Output, PIN 24.

CLKOUT can be used to synchronize peripheral devices such as a micro controller or an audio source. The clock frequency is determined by a divide by Q in the clock manager. The maximum CLKOUT frequency is the maximum DSP frequency divided by 2.

ALTCLK - Clock Input, PIN 28.

When EXTCK is high, ALTCLK is an input for an externally generated clock. This clock directly becomes the DSP clock and the clock frequency should be 512Fs or 768Fs.

EXTCK - External Clock Select, PIN 29.

Setting EXTCK high allows ALTCLK to be used as an input for an external VCO. Setting EXTCK low disables ALTCLK. Note that EXTCK should be tied directly to either digital power or ground for proper operation.

FLT - PLL Filter, PIN 31.

A capacitor (typically 0.47 μF) connected to this pin filters the control voltage for the on-chip VCO. Trace length should be minimized to the pin.

CLKIN - Clock Input, PIN 27.

A clock input to the CLKIN is used to synchronize the PLL's. The permissible frequency range is from 256 kHz to 30 MHz. It is typical to have SCLK for the audio data and CLKIN to be derived from the same clock source to avoid asynchronous noise between the audio source and the DSP.

90_CLK - Optional SCR/PCR 33-Bit Counter Clock, PIN 19

The 90_CLK pin is an input clock signal (typically 90 kHz) which is used to clock the internal 33-bit counter. The 33-bit counter is enabled by SCREN = 1 in the CM0 register. The 33-bit counter's clock source is set to 90_CLK when DIV = 0 in the CM0 register. Otherwise when DIV = 1, the 33-bit counter will be clocked by CLKIN ÷ 300.

8.0.6. Control

DBCLK, DBDA - Debug Port, PINS 12, 13.

It is required that a pull-up be used (typically 2.2 kΩ) on pin 13.

$\overline{\text{RESET}}$ - PIN 41.

The CS4922 enters a reset state while $\overline{\text{RESET}}$ is low. When in reset condition, all internal registers are set to 0, the digital audio transmitter, serial control port, and ALTCLK pin are disabled, and the stereo DAC is muted. Normal operation is resumed one internal clock cycle after the rising edge of $\overline{\text{RESET}}$.

BOOT - PIN 40.

Boot enable pin. Pin must be set high to initiate the download of a program. While BOOT is high, $\overline{\text{RESET}}$ must be toggled high. This starts the internal boot program.

XF1, XF2, XF3, XF4 - External Flags, PINS 20, 16, 15, 14.

The XF pins are software controllable outputs via the LINT register. These pins are open drain so an external pullup is required (typically 2.2 kΩ) for proper operation of the pins.

PIO - Programmable Input/Output, PIN 30.

The PIO pin is a software controllable input/output pin via the LINT register. The pin is configured as an output when I_O = 1, and the output on the pin is high when POUT = 1 or low when POUT = 0 in the LINT register. The pin is configured as an input when I_O = 0, and a rising edge on the input of PIO pin will cause an interrupt if PIEN = 1 in the LINT register. The input level to the PIO pin can be read by the DSP via the PIN bit of LINT register. When configured as an input, a high level on the PIO pin will set PIN = 1 and a low level will set PIN = 0.

8.0.7. Serial Control Port

$\overline{\text{REQ}}$ - Request Output, PIN 3.

This pin is driven low when the DSP needs servicing from an external device. A write to the SCPOUT will cause the $\overline{\text{REQ}}$ to go low. A pull-up resistor is required for proper operation (2.2kΩ is typical).

$\overline{\text{CS}}$ - Chip Select Input, PIN 44.

In SPI format, all communication between the host and the CS4922 is initiated when the host drives the $\overline{\text{CS}}$ pin low. This pin also serves as the communication format select during a reset or power up. When $\overline{\text{CS}}$ is high during a reset or power up the SCP will be configured in $I^2C^®$ mode. When low, it is configured in SPI mode. The mode is selectable in software by setting the M0 bit in the SCPCN.

SCK/SCL - Serial Clock Input, PIN 2.

SCK/SCL clocks data into or out of the serial control port. This is always driven by an external device because the CS4922 always operates in slave mode.

SDA/CDOUT - Serial Data I/O / Control Data Output, PIN 4.

In SPI mode, CDOUT is a data output for the serial control data. In I^2C interface mode, SDA is a bi-directional data I/O. It is required that a pull-up be used (2.2 kΩ is typical in I^2C mode).

CDIN - Control Data Input, PIN 1.

In SPI mode, CDIN is the data input for the serial control port. It has no function in I^2C mode. The pin should be connected to either digital power or ground when the CS4922 is used in I^2C systems.

8.0.8. Auxiliary Digital Audio Port

AUXLR - Auxiliary Sample Clock, PIN 10.

This output signal determines which channel is currently being input on the AUXIN pin or output on the AUXOUT pin. It is also the sample clock, Fs.

AUXIN - Auxiliary Data Input, PIN 9.

Two's complement MSB first serial data is input on this pin. The data is clocked by AUXCLK and the channnel is determined by AUXLR.

AUXOUT - Auxiliary Data Output, PIN 8.

Two's complement MSB first serial data is output on this pin. The data is clocked by AUXCLK and the channel is determined by AUXLR.

AUXCLK - Auxiliary Serial Clock Output, PIN 11.

This is the auxiliary audio port serial clock output. This output is used to clock data in on the AUXIN pin and shift data out on the AUXOUT pin. Its frequency is selectable in software.

9. PARAMETER DEFINITIONS

Resolution

The number of bits in the input words to the DACs.

Differential Nonlinearity

The worst case deviation from the ideal codewidth; expressed in LSBs.

Total Harmonic Distortion (THD)

THD is the ratio of the test signal amplitude to the rms sum of all the in-band harmonics of the test signal.

Instantaneous Dynamic Range

The Signal-to-(Noise + Distortion) ratio (S/(N+D)) with a 1 kHz, -60dB from full scale DAC input signal, with 60dB added to compensate for the small signal. Use of a small signal reduces the harmonic distortion components of the noise to insignificant levels. Units are in dB.

Interchannel Isolation

The amount of 1kHz signal present on the output of the grounded input channel with 1 kHz, 0dB signal present on the other channel. Units are in dB.

Interchannel Gain Mismatch

The difference in output voltages for each channel with a full scale digital input. Units are in dB.

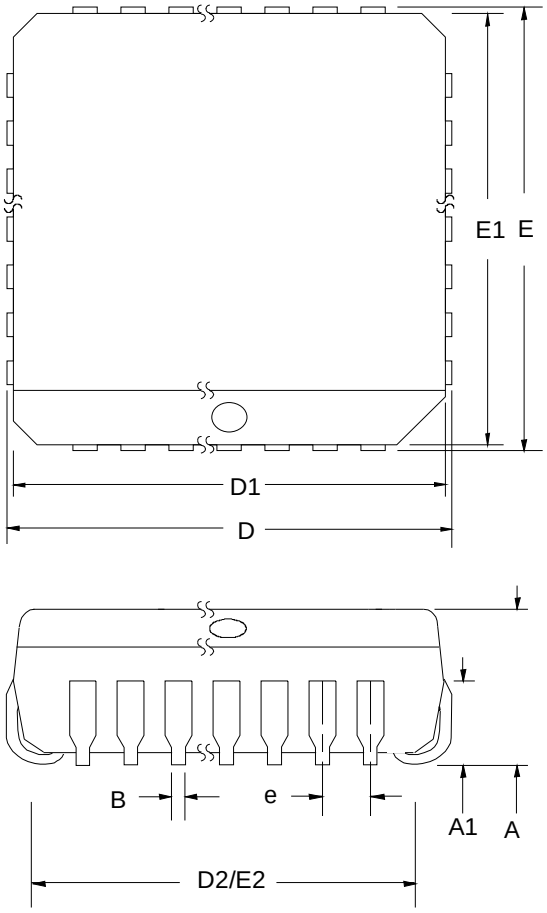
Frequency Response

Worst case variation in output signal level versus frequency over 10 Hz to 20 kHz. Units in dB.

Out of Band Energy

The ratio of the rms sum of the energy from 0.46×Fs to 2.1×Fs compared to the rms full-scale signal value. Tested with 48 kHz Fs giving a out-of-band energy range of 22 kHz to 100 kHz.

10.PACKAGE DIMENSIONS



44 pin
PLCC

DIM	NO. OF TERMINALS					
	MILLIMETERS			INCHES		
	MIN	NOM	MAX	MIN	NOM	MAX
A	4.20	4.45	4.57	0.165	0.175	0.180
A1	2.29	2.79	3.04	0.090	0.110	0.120
B	0.33	0.41	0.53	0.013	0.016	0.021
D/E	17.40	17.53	17.65	0.685	0.690	0.695
D1/E1	16.51	16.59	16.66	0.650	0.653	0.656
D2/E2	14.99	15.50	16.00	0.590	0.610	0.630
e	1.19	1.27	1.35	0.047	0.050	0.053

Evaluation Board for CS4922

Features

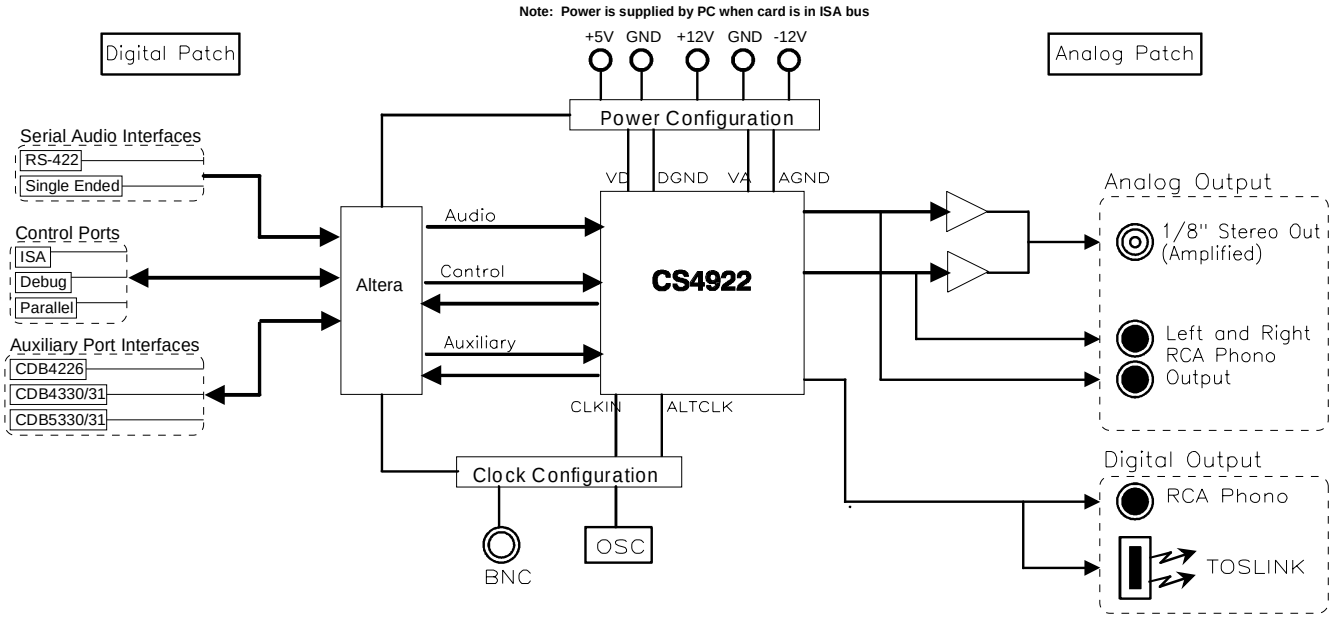
- Demonstrates recommended layout and grounding arrangements.
- Interfaces to parallel port or ISA bus of a personal computer for easy evaluation.
- On-board or externally supplied clocks.
- Accepts digital audio data transmitted either single ended or RS-422 format.
- Digital and analog patch areas.
- Interfaces directly with the CDB433X, CDB533X and CDB4226 evaluation boards.

Description

The CDB4922 evaluation board provides an effective means to evaluate the CS4922 MPEG audio decoder. Real time compressed audio data can be input to the CS4922 in an RS-422 or single ended format. Software is shipped with the board that allows MPEG files to be played over the ISA bus or parallel port of a PC compatible computer. The CS4922 analog output is available through right, left and mono RCA jacks. The CS4922's IEC958 compatible digital output is accessible by both an optical connector and a RCA jack. Several stake headers are provided on the board to interface the auxiliary port of the CS4922 directly to the evaluation boards of the CS4226, CS4330/31 and CS5330/31.

ORDERING INFO

CDB4922 Evaluation Board



Cirrus Logic, Inc.
Crystal Semiconductor Products Division
P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.crystal.com

CDB4922 EVALUATION BOARD USERS MANUAL

Crystal Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function and/or design. Crystal Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

Introduction

This manual provides general description, hardware preparation, installation instructions, operating instructions and hardware description for the CS4922 Evaluation Board (hereafter referred to as CDB4922).

GENERAL DESCRIPTION

Overall

The CDB4922 evaluation board provides an effective means to evaluate the CS4922 multistandard audio decoder. Real time compressed audio data can be input via one of two stake headers. The audio data is received either single ended or by an RS-422 receiver. Software is shipped with the board that allows MPEG files to be played over the ISA bus or parallel port of an IBM compatible computer. The CS4922 analog output is available through right, left and mono RCA jacks. The CS4922's IEC-958 compatible digital output is accessible by both an optical connector and a standard RCA jack. On board headers also allow access to the CS4922's auxiliary port. These headers were designed to interface directly with the CS433X, CS533X and CS4226 evaluation boards. If it is desired to use another external device to interface to the CS4922 auxiliary port, it can be easily wire-wrapped to these headers.

The CDB4922 evaluation board also demonstrates the robust timing capabilities of the CS4922 clock manager. The evaluation board can be configured to accept an external timing source, or an on-board oscillator to generate the necessary timing signals.

Control of the CS4922 can be accomplished in many ways. Software is provided with the board that can communicate with the CS4922 over either the ISA bus or parallel port of an IBM compatible computer. If lower level control of the CS4922 is desired, the removal of one chip allows control of all the CS4922 signals via stake headers. An additional connector allows access to the CS4922's debug port, which is used with the CS4922 software debugger.

HARDWARE DESCRIPTION

Hardware Preparation

The CDB4922 is shipped ready to play MPEG files from either the ISA bus or the parallel port of a PC. The board is configured as follows:

- On-Board 27MHz crystal oscillator
- ISA address 0x340
- Interrupt 7
- Internal Data (from PC)

If for some reason one of these settings will cause a conflict in your computer, or if the board has been changed from its original configuration please refer to the Jumper Settings section of this document on how change the settings to work in your system.

The cable used for evaluation through the parallel port is shown in Figure 1. If for some reason the cable from the factory gets damaged, another can be easily made by connecting a DB25M connector to a 26 pin IDC with wire 26 left as a no connect.

CS4922 Schematic

Figure 2 shows the schematic diagram of the CS4922. The capacitors shown are decoupling capacitors for each digital and analog power pin. The capacitors are placed as close to the chip as layout allows to filter high frequency noise coming from the power supply. Note that the .1uF capacitor

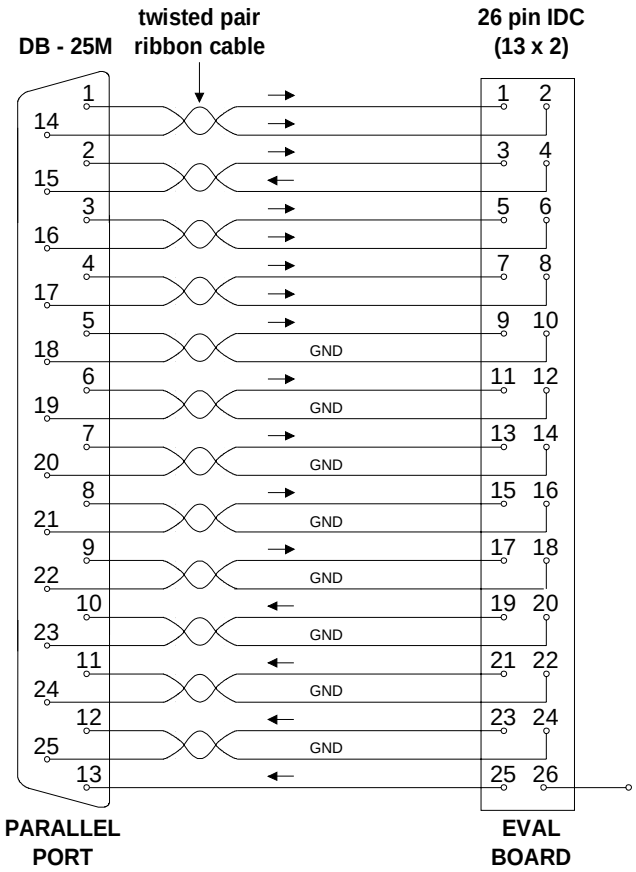


Figure 1. Twisted Cable

should be placed closer to the chip than the 1uF capacitor for improved performance. In addition to the decoupling capacitors the FLT (Phase Locked Loop filter pin) pin requires an external capacitor to analog ground.

Test pins have been provided for all input and output signals to the CS4922. These pins can be used to monitor the CS4922 with an oscilloscope or a logic analyzer. If low level control of the CS4922 is desired without use of the provided software (i.e. through an external microcontroller), U12 can be removed and all inputs of the CS4922 can be driven without contention. NOTE: If U12 is removed and it is desired to use the AUXIN signal from J4 or J5, a jumper must be connected between the CS4922 AUXIN pin and J4 or J5.

Serial Audio Input Interface

Figures 3 and 4 show the schematics for the serial audio input interface. For TTL or CMOS level inputs, jumper J7 provides an interface for SDATA, SCLK, and FSYNC as shown in Figure 3. Jumper J11 provides an RS422 differential interface for SDATA and SCLK. It also allows for the transmission of the 4922SCLK signal if an external device such as an audio encoder is to be slaved to the CS4922. Jumpers J9 and J10 determine which data will be buffered to the CS4922 through the on board EPLD. Jumper J12 determines if 4922SCLK signal is to be transmitted over the RS422 header.

Auxiliary Interface

Jumpers J4, J5, and J6 interface the CS4922's auxiliary port to evaluation boards of other Crystal products. J4 allows the auxiliary port to connect directly to the CDB4226, a surround sound CODEC evaluation board. The signals 4922SDA_CDOUT and 4922SCK_SCL have been included on this header so that both the CS4922 and CS4226 can be controlled off the same I2C lines. J5 allows the CDB4922 to connect directly to the CDB5330, an analog to digital converter evaluation board. J6 allows the CDB4922 to connect directly to the CDB4330, a digital to analog converter evaluation board. Although these stake headers were designed to interface directly with other Crystal evaluation boards, they can be used to connect the CS4922 auxiliary port to any appropriate CMOS serial audio interface.

The signal /AUXINSEL should be grounded (PIN 2 - J5 or PIN18 - J4) if an external digital audio source for the auxiliary port is to be used. This will be taken care of by design when connecting to other Crystal boards. This signal tells the EPLD to route the signals from the stake headers to the CS4922.

Control Interface

Figure 6 and 7 show the two control port interfaces for the CDB4922. The ISA port interface is designed for an industry standard architecture (ISA) bus in an IBM compatible PC. The host port interface is designed to be used with a parallel port in an IBM compatible personal computer. Application software is available for downloading microcode, sending commands, receiving messages and playing MPEG files with the CS4922. This software is compatible with both interfaces.

Reset Schematic

Figure 8 shows the reset circuitry for the CS4922. U9 insures that the part is held in reset until proper power levels are maintained and provides the debouncing mechanism for the reset switch.

Power Supply Schematic

Figure 9 shows the connections for the power supply circuitry. **NOTE: Care should be take NOT to connect power to binding posts J20, J21, J23, J25, J26 when the card is plugged into the ISA bus or irreparable damage can occur to both the card and the computer.** These binding posts are for stand alone mode only. When the card is in the ISA bus it derives all voltages from the bus. As shown in the schematic a variety of power and ground configurations can be used with the CS4922 through the removal and placement of different components. The power supply circuitry section in Jumper Setting describes the different configurations and applies to both ISA and stand alone operation.

If it is desirable to have different grounds for the digital and analog planes then both FB3 and FB4 should be removed. If these planes are to be the same then only one of FB3 or FB4 should be installed as to prevent a ground loop.

Digital Interface

Figure 10 shows the schematic for the digital transmitter coming from the CS4922. Both an optical jack and an RCA jack are available for digital output. As configured at the factory the RCA output is S/PDIF compatible. If AES/EBU is desired, R47 should be removed, R46 should be replaced with a 110 Ohm resistor and pin 3 of T1 should be connected to pin 12 of U5 via TP59. Pins 4 and 6 of T1 would then provide a differential output and can be connected to an XLR connector.

Analog Outputs

Three RCA jacks are provided to output the analog signals generated by the CS4922's DACs. Left, right and mono are all available. The output circuit is shown in Figure 11. The DACs are capable of driving a minimum of 8kOhm loads and thus are not intended to drive speakers except through an external amplifier.

A 1/8" stereo phono jack also provides an amplified analog stereo output of the CS4922's DACs as shown in Figure 11. This output can be used to drive headphones if desired.

EPLD Schematic

Figure 12 shows the schematic for the field programmable logic device (EPLD) that is used on the CDB4922. This device is used for routing signals and for communicating with the application software sent with the CDB4922. If direct control of the CS4922 is desired this device (U12) should be removed so that there is not contention on the CS4922's inputs.

Clock Manager

Figure 13 shows the schematic for the clock manager portion of the CDB4922. This arrangement allows for a complete evaluation of the CS4922 robust timing capabilities. The onboard crystal is socketed to allow for different clock speeds and an external BNC connector is provided so that an external clock or waveform generator can be used to evaluate the CS4922. A complete explanation of how to configure the clock manager is found in the *Jumper Settings* portion of this datasheet under *Clock Configuration Jumpers*.

Spare Gates

Figure 14 shows the spare gates available on the board. If these gates are to be used, the corresponding pull-up resistor on the input should be removed. The test points allow a convenient way to hookup signals to the inputs and output.

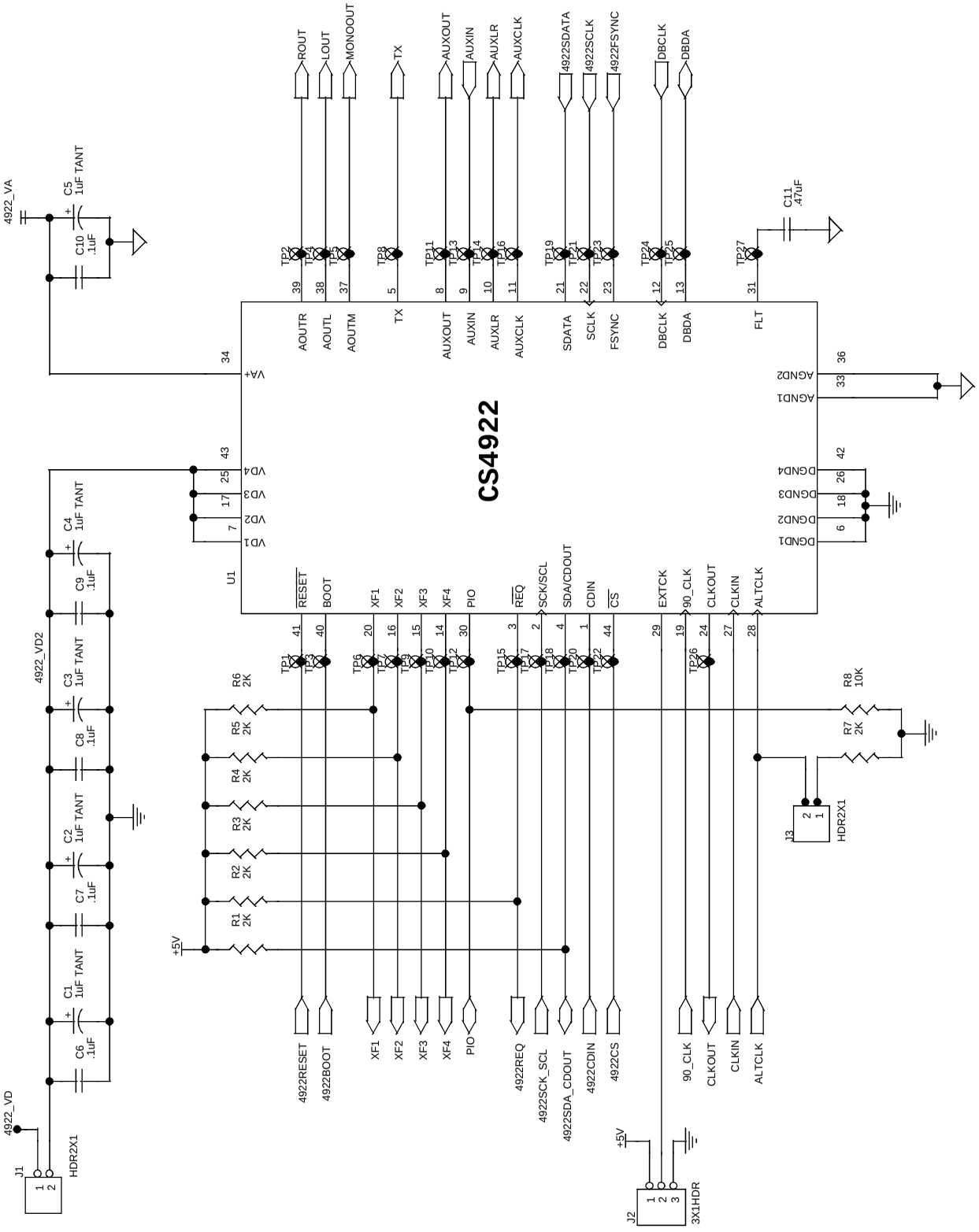


Figure 2. CS4922 Schematic

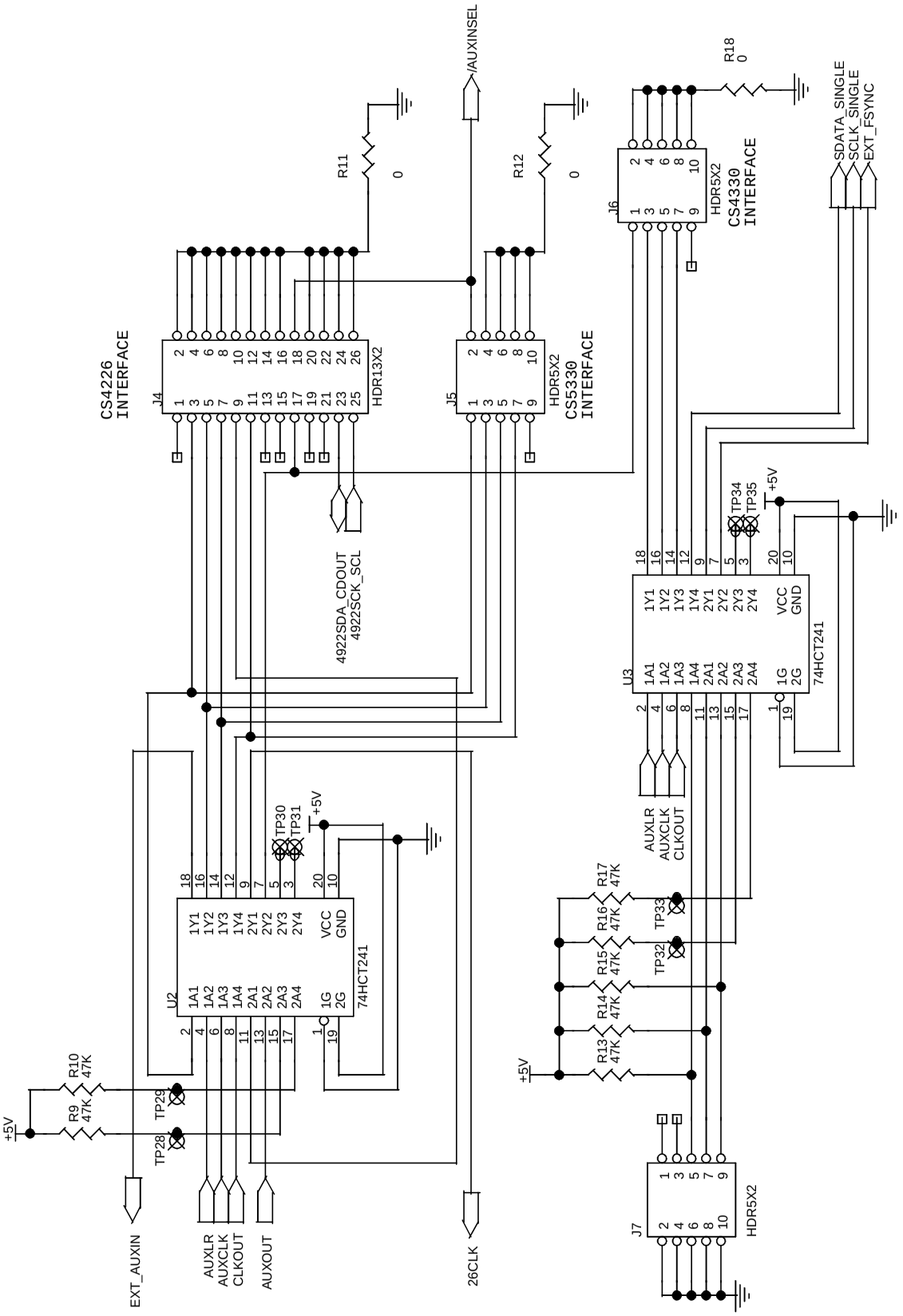


Figure 3. Codec Interface

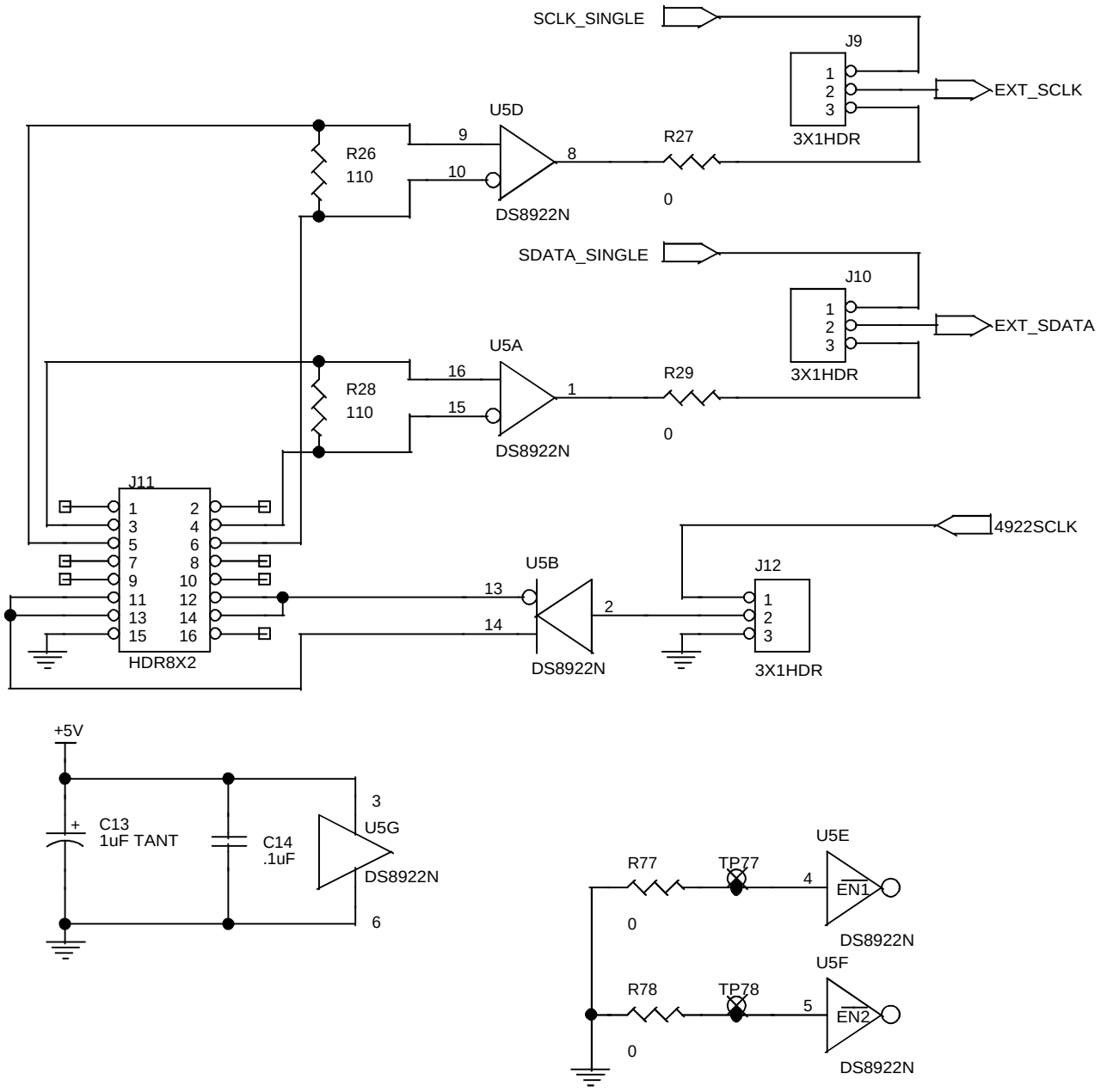


Figure 4. RS422 Interface

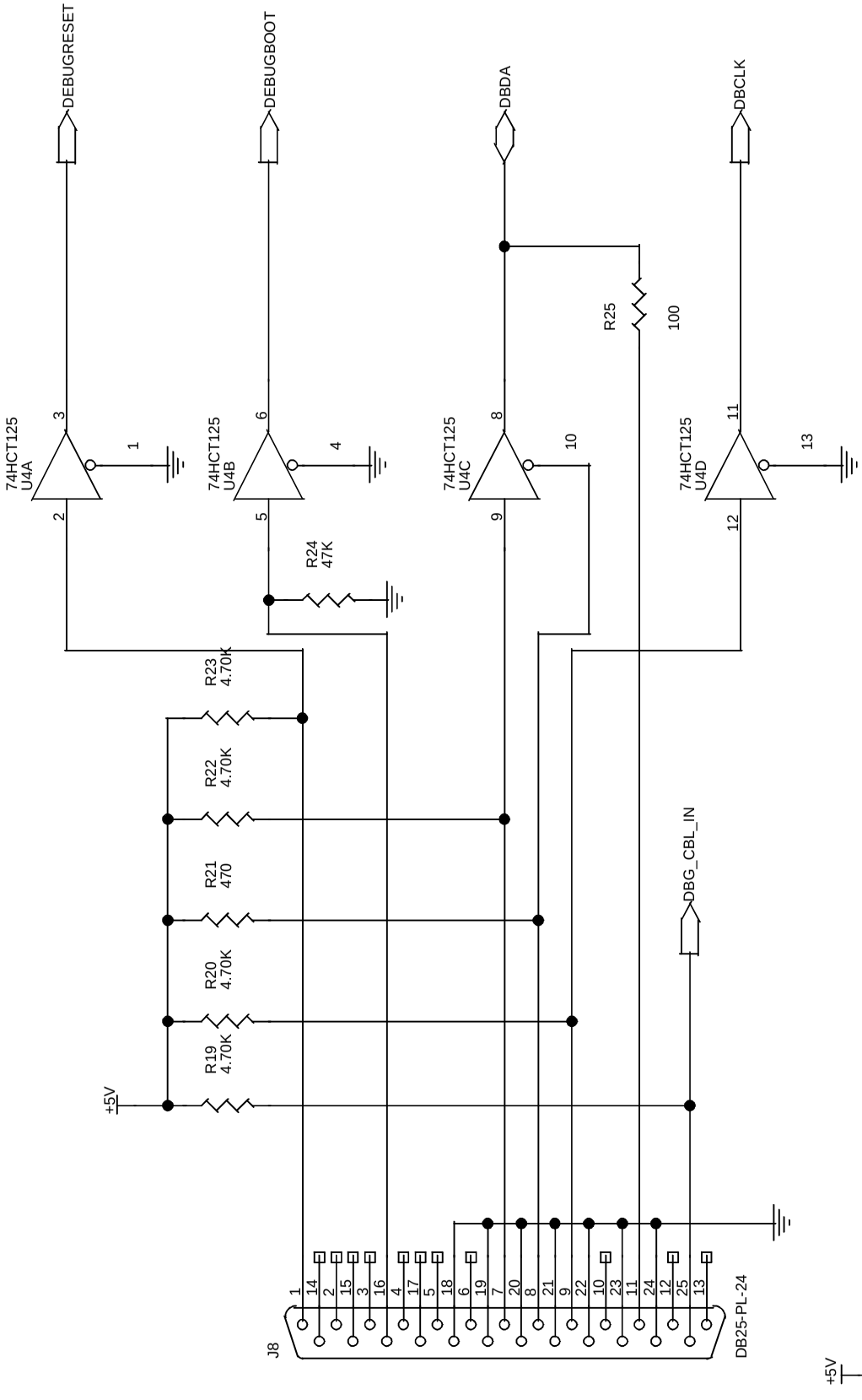


Figure 5. Debug Interface

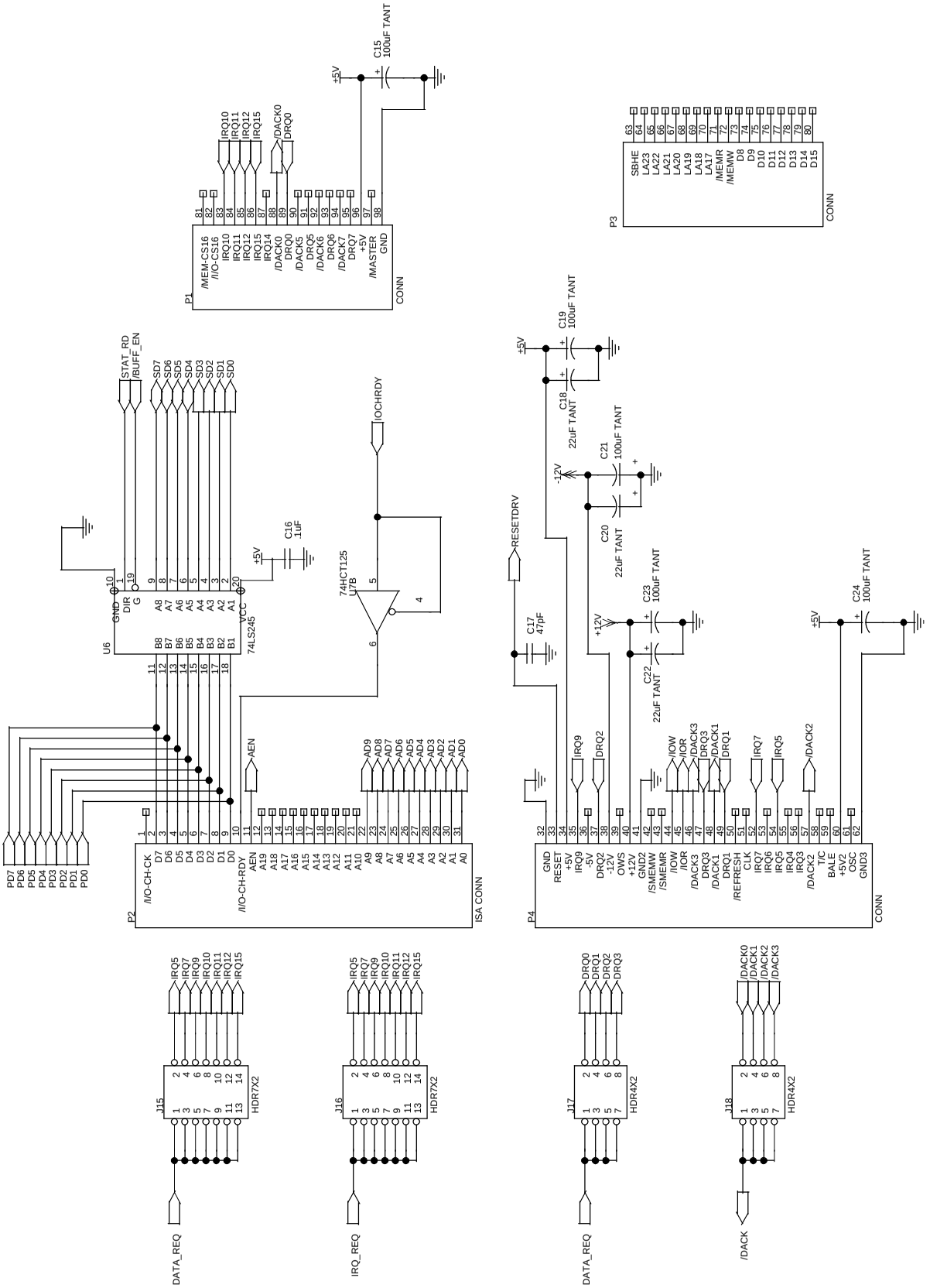


Figure 6. ISA Port Interface

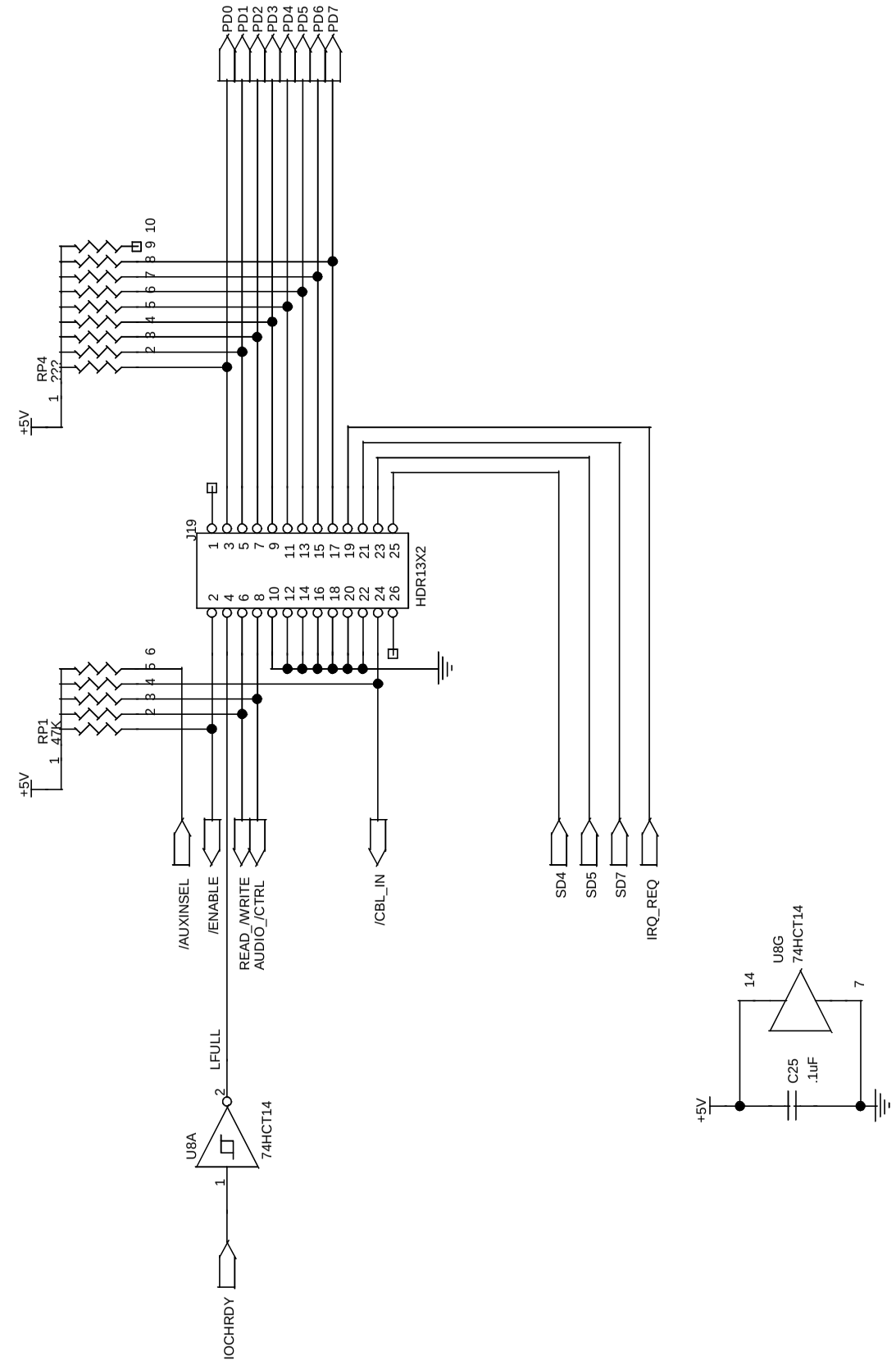


Figure 7. Host Port Interface

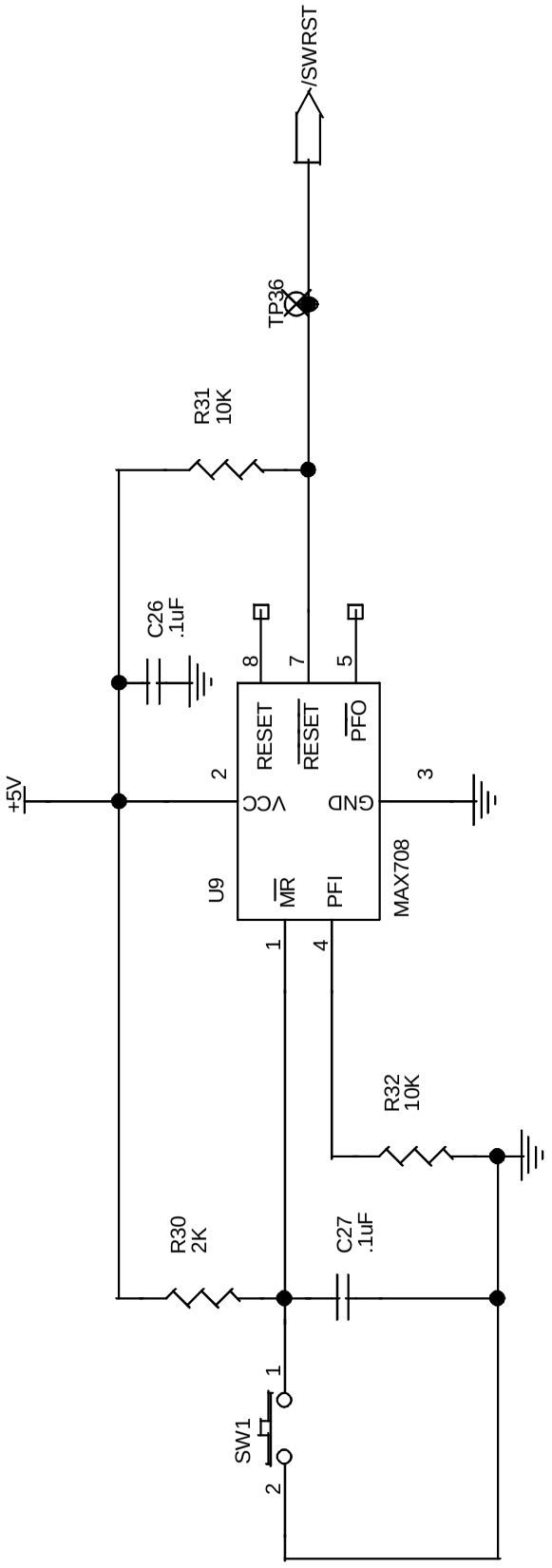


Figure 8. Reset Schematic

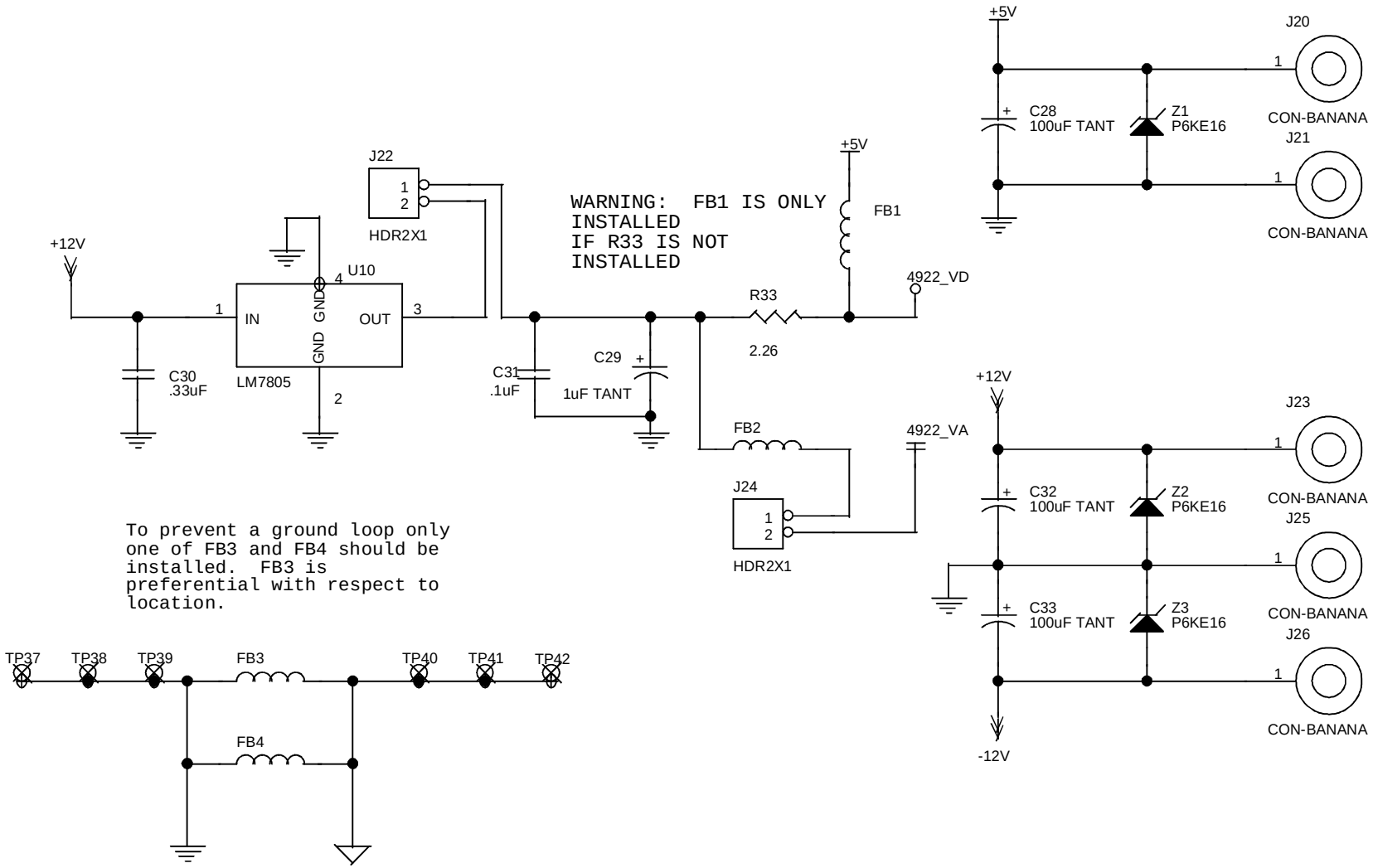


Figure 9. Power Supply Schematic

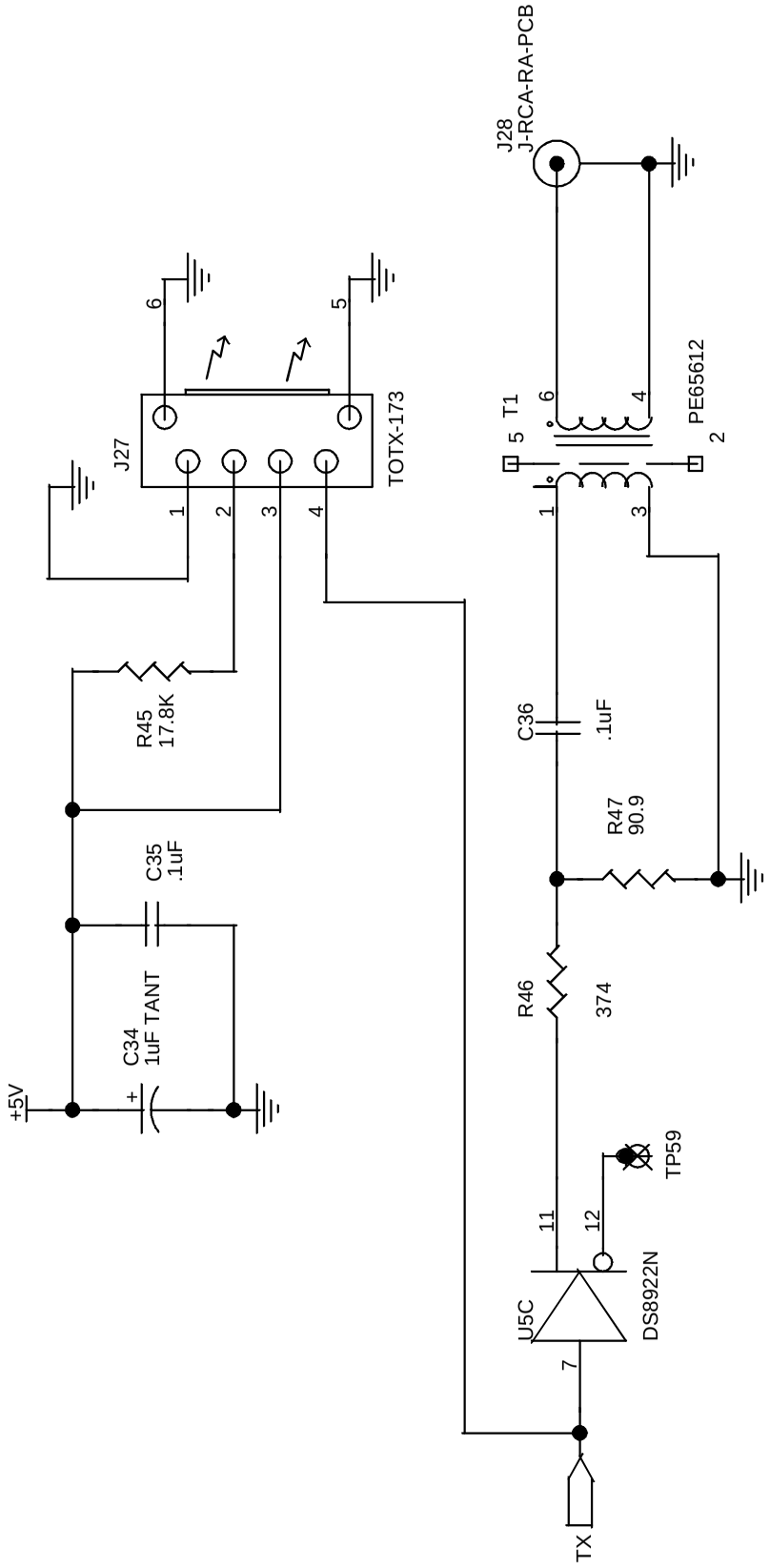


Figure 10. Digital Transmitter

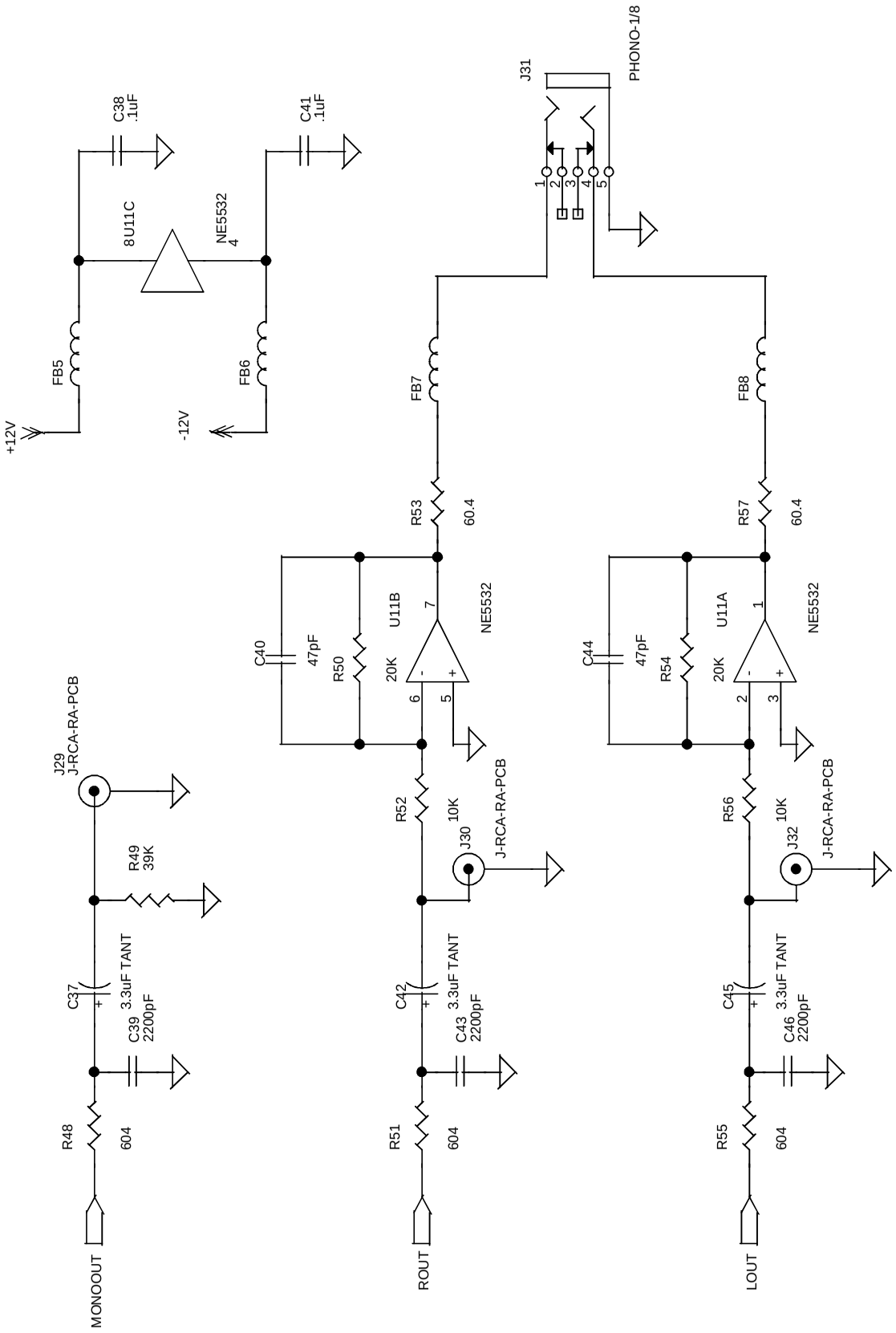


Figure 11. Analog Output

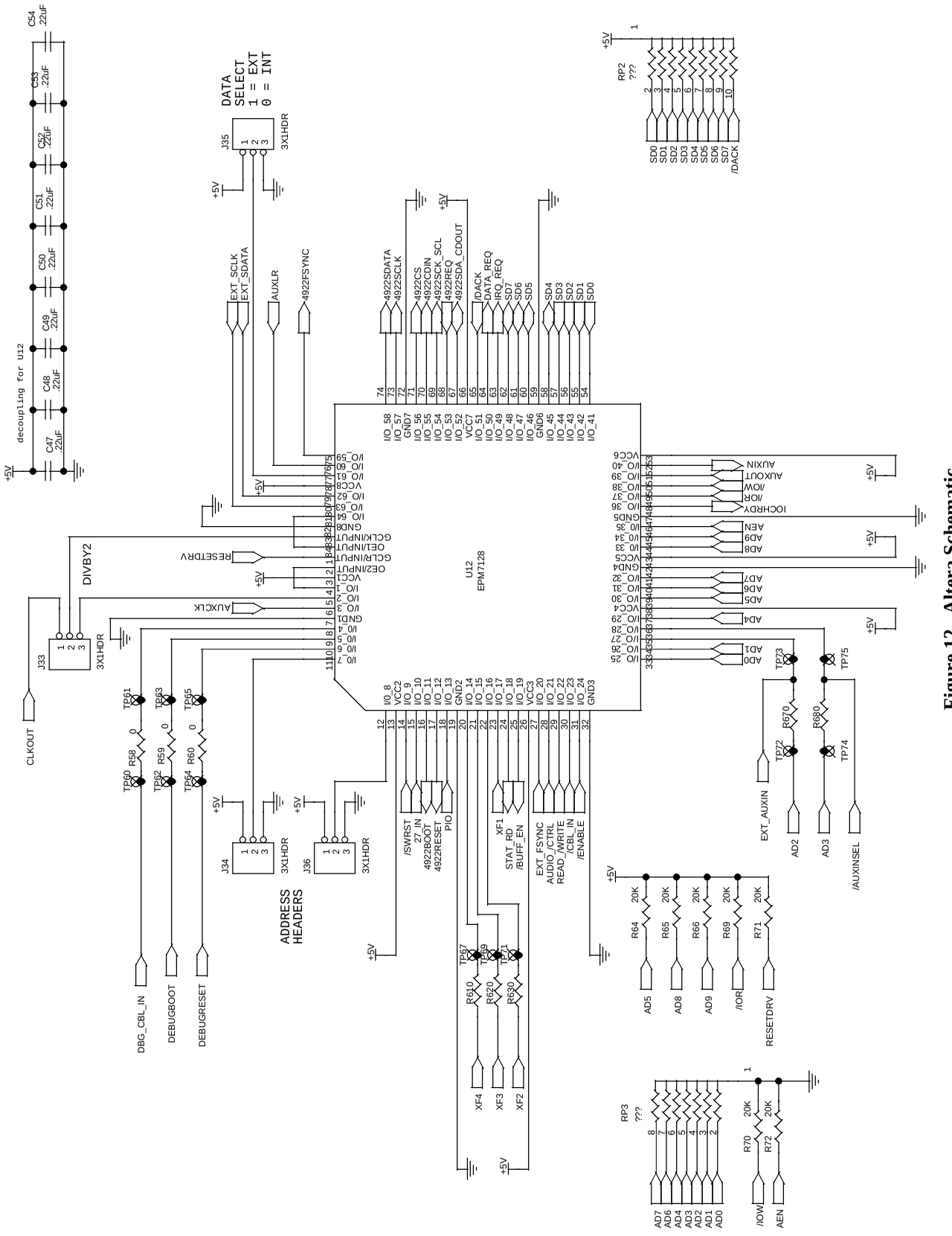


Figure 12. Altera Schematic

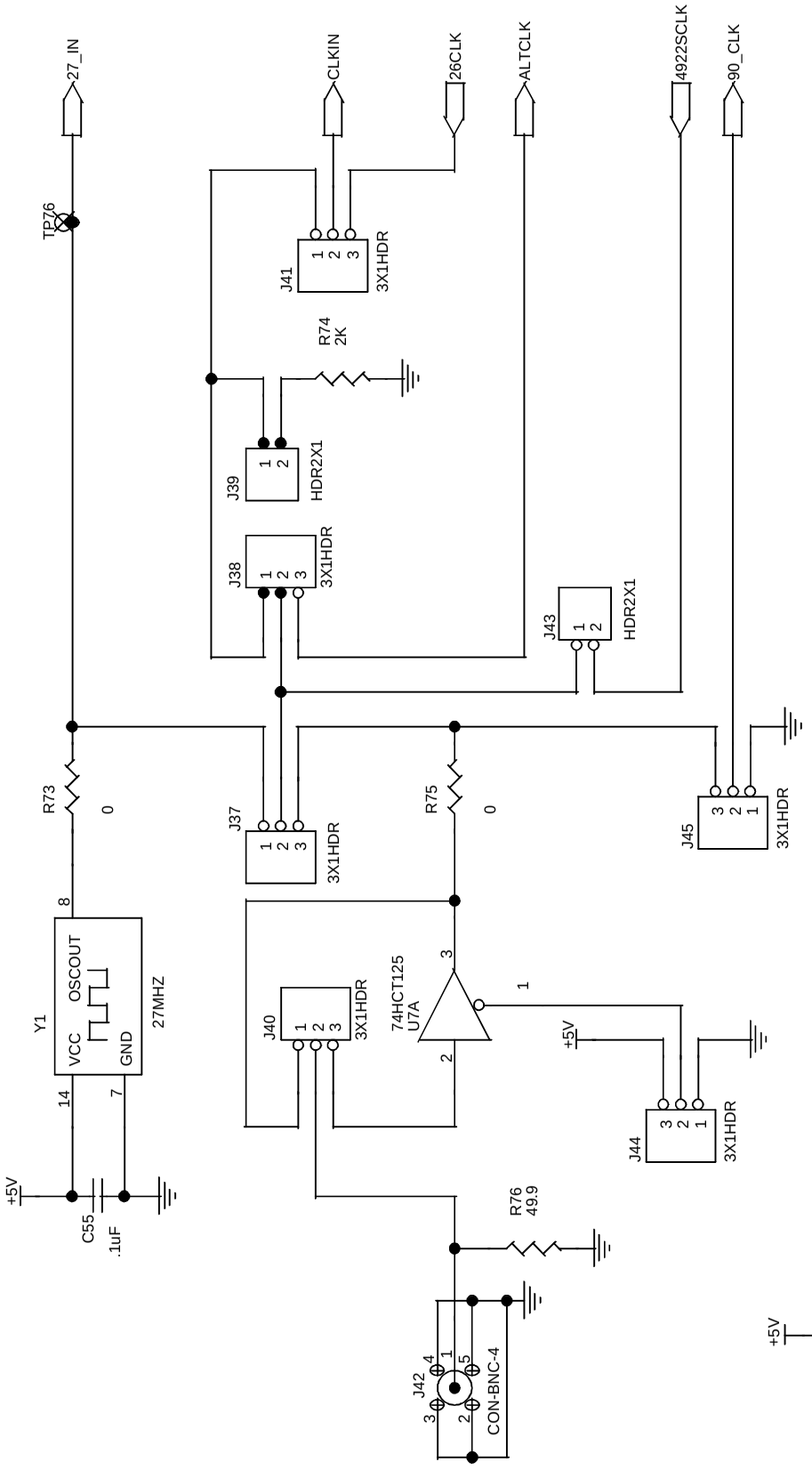


Figure 13. Clock Manager

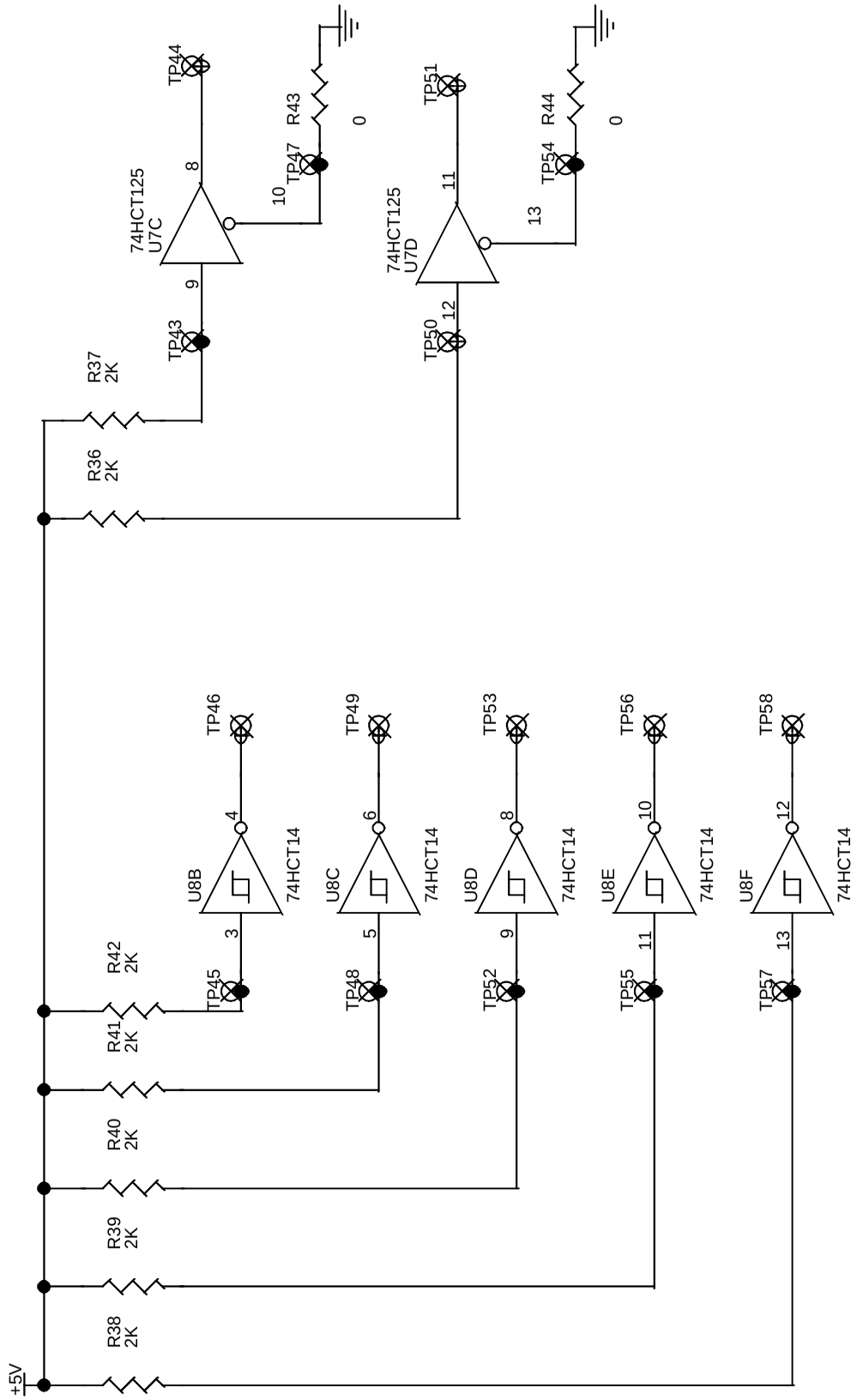


Figure 14. Spare Gates

JUMPER SETTINGS

Power Jumpers

(J1, J22 and J24)

J1 - This jumper connects power to the digital portion of the CS4922. It can be used to measure the current being drawn by the digital side of the CS4922 by connecting an ammeter in series with the jumper. DEFAULT ON

J24 - This jumper connects power to the analog portion of the CS4922. It can be used to measure the current being drawn by the analog side of the CS4922 by connecting an ammeter in series with the jumper. DEFAULT ON

J22 - This jumper connects the 5 volt output of the on-board regulator to J24 through FB2 and to J1 through R33.

WARNINIG: Care should be take NOT to connect power to binding posts J20, J21, J23, J25, J26 when the card is plugged into the ISA bus or irreparable damage can occur to both the card and the computer.

Clock Configuration Jumpers

(J2, J3, J37, J38, J39, J40, J41, J43, J44 and J45)

J2 - This jumper is connected to EXTCLK and allows the user to choose whether the CS4922 will run off CLKIN or ALTCLK. When in the ENAB position (EXTCLK pulled high), the CS4922 runs in ALTCLK mode. When in the DIS position (EXTCLK pulled low), the CS4922 runs in CLKIN mode.

J3 - This jumper grounds ALTCLK when the CS4922 is running in CLKIN mode.

J37 - This jumper allows the user to decide the source for CLKIN and ALTCLK. When in the OSC position, the on board oscillator is the source. When in the BNC position, the off-board connector is the source. If J37 has no jumper then the 4922SCLK signal can drive CLKIN or ALTCLK. If J37 is set then no jumper should be placed on J43.

J38 - This jumper allows the user to decide whether CLKIN or ALTCLK is used. When in the CKIN position, clock signals are routed to CLKIN. When in the ALT position, the clock signals are routed to ALTCLK.

J39 - This jumper grounds CLKIN when the CS4922 is running in ALTCLK mode. NOTE: To ground CLKIN, J41 must be in the J38 position.

J40 - This jumper allows a bypass of the onboard BNC clock buffer. When in the bypass position, the BNC input bypasses the buffer. When in the buffer position, the BNC input is fed into the buffer. NOTE: If J40 is in the BYPASS position, J44 should also be in the bypass position to avoid contention. See description of J44.

J41 - This jumper decides the source for CLKIN. When in the J38 position (see silk-screen, a line is painted from this position to J38) 4922SCLK, the OSC, or the BNC will drive CLKIN (see jumper table). When in the 26CLK position, 26CLK will drive CLKIN. NOTE: If running in ALTCLK

Source	Destination	J2	J3	J37	J38	J39	J40	J41	J43	J44	J45
OSC	CLKIN	DIS	ON	OSC	CLKIN	OFF	BUFFER	J38	OFF	BYPASS	OFF
OSC	ALTCLK	ENAB	OFF	OSC	ALT	ON	BUFFER	J38	OFF	BYPASS	OFF
BUFF-BNC	CLKIN	DIS	ON	ENC	CLKIN	OFF	BUFFER	J38	OFF	BUFFER	OFF
BUFF-BNC	ALTCLK	ENAB	OFF	ENC	ALT	ON	BUFFER	J38	OFF	BUFFER	OFF
BNC	CLKIN	DIS	ON	ENC	CLKIN	OFF	BYPASS	J38	OFF	BYPASS	OFF
BNC	ALTCLK	ENAB	OFF	ENC	ALT	ON	BYPASS	J38	OFF	BYPASS	OFF
4922SCLK	CLKIN	DIS	ON	OFF	CLKIN	OFF	BUFFER	J38	ON	BYPASS	OFF
4922SCLK	ALTCLK	ENAB	OFF	OFF	ALT	ON	BUFFER	J38	ON	BYPASS	OFF
26CLK	CLKIN	DIS	ON	OFF	CLKIN	OFF	BUFFER	26CK	OFF	BYPASS	OFF

mode this jumper should be set to pins 1 and 2 so CLKIN can be properly grounded.

J43 - This jumper allows 4922SCLK to drive either ALTCLK or CLKIN. If J41 is set then no jumper should be placed on J37.

J44 - This jumper enables the BNC buffer. When in the BUFFER position, the buffer is enabled. When in the BYPASS position, the buffer is disabled. NOTE: If J40 is in the BYPASS position, J44 should also be in the BYPASS position to avoid contention. See description of J40.

J45 - This jumper allows the 90_CLK pin of the CS4922 to be driven by an external source. When in the OFF position 90_CLK is grounded. When in the SEL position 90_CLK is driven. The table below shows the jumper settings to drive 90_CLK with a buffered external clock and drive the CS4922 using CLKIN derived from the on-board oscillator.

Source	OSC
Destination	CLKIN
J2	DIS
J3	ON
J37	OSC
J38	CLKIN
J39	OFF
J40	BUFFER
J41	J38
J43	OFF
J44	BUFFER
J45	SEL

Settings to Drive 90_CLK from External Source

Serial Data Selection Jumpers

(J9, J10, and J12)

J9 - This jumper selects whether the EXT_SCLK signal is derived from the differential or single ended input header. When DIFF is selected, EXT_SCLK is derived from the differential input and when single is selected, EXT_SCLK is derived from the single ended input.

J10 - This jumper selects whether the EXT_SDATA signal is derived from the differential or single ended input header. When DIFF is selected, EXT_SCLK is derived from the differential input and when single is selected, EXT_SCLK is derived from the single ended input.

J12 - This jumper allows 4922SCLK to be output through the RS422 header so that external devices can be slaved to the CS4922's SCLK. When in the SCLK position the SCLK is output on the header, when in the ground position this output is grounded.

J35 - This header determines what data the Altera feeds to the ASI port of the CS4922. When in the EXT position, EXT_SCLK and EXT_SDATA are used. and when in the INT position data off the ISA bus or off the parallel port connector is used.

Address Jumpers

(J34 and J36)

J34, J36 - These jumpers allow different addresses to be assigned to the card when in the ISA bus. The following table describes the different addresses available.

ISA I/O ADDR	J34	J36
260	GND	GND
280	GND	5V
320	5V	GND
340	5V	5V

INTERFACE DESCRIPTIONS

Auxiliary Port Interface Headers

(J4, J5, and J6)

J4 - CS4226 Interface - This header is designed to interface the auxiliary port of the CS4922 with a CS4226 surround CODEC evaluation board. 4922SDA_CDOUT and 4922SCK_SCL have been included so that both the CS4922 and the CS4226 can be controlled off the same I2C lines.

J5 - CS5330/31 Interface - This header is designed to interface the auxiliary port of the CS4922 with the CS5330/31 analog to digital converter evaluation board.

J6 - CS4330/31 Interface - This header is designed to interface the auxiliary port of the CS4922 with the CS4330/31 digital to analog converter evaluation board.

Serial Audio Input Interface Headers

(J7 and J11)

J7 - TTL or CMOS Serial Audio Input - This header provides pins for FSYNC, SDATA, and SCLK signals. These signals are buffered and can be chosen for input into the CS4922 through jumpers J9, J10, and J35 (see description above).

J11 - Differential Serial Audio Input - This header receives the signals SDATA and SCLK fro an RS-422 line driver. These signals can be chosen for input into the CS4922 through jumpers J9, J10, and J35 (see description above). The connector also provides for the transmission of 4922SCLK with J12 set appropriately (see description).

Control Port Interface Connectors

(P1, P2, P3, P4, J15, J16, J17, J18, and J19)

P1, P2, P3, P4 - ISA Interface - These connectors are designed to interface the CDB4922 evaluation board to an Industry Standard Architecture (ISA) system bus. Software shipped with the evaluation board is designed to download and communicate with the CS4922 through the ISA bus or the parallel interface.

J15, J16 - These headers are used to determine which interrupt the CDB4922 will use.

J17, J18 - These headers are used to determine the DMA channel that the CDB4922 can access. Currently there is not software support for DMA transfers between the PC and the CDB4922.

J19 - Parallel Port Interface - This connector is designed to interface the CDB4922 evaluation board to the parallel port of an IBM compatible personal computer. Software shipped with the evaluation board is designed to download and communicate with the CS4922 through the parallel interface or the ISA bus.

Debug Interface

(J8)

J8 - Debug Interface - J8 is an interface to the debugger of the CS4922. Software is available to interface the debug port of the CS4922 through the parallel port of an IBM compatible personal computer.

Output Descriptions

(J27, J28, J29, J30, J31, and J32)

J28 - **S/PDIF** or **AES/EBU** output - This connector provides the digital audio output from the CS4922. As configured at the factory this output is S/PDIF compatible. If AES/EBU is desired, R47 should be removed, R46 should be replaced with a 110 Ohm resistor and pin 3 of T1 should be connected to pin 12 of U5 via TP59. Pins 4 and 6 of T1 would then provide a differential output and should be connected to an XLR connector.

J27 - **Optical S/PDIF output** - This connector provides an optical S/PDIF output of the CS4922's digital output.

J29 - **Mono Analog** output - This RCA jack provides the mono analog output of the CS4922. These DACs are capable of driving 8kohm loads

and thus are not intended to drive speakers except through an external amplifier.

J30 - **Left Analog** output - This RCA jack provides the left analog output of the CS4922. These DACs are capable of driving 8kohm loads and thus are not intended to drive speakers except through an external amplifier.

J32 - **Right Analog** output - This RCA jack provides the right analog output of the CS4922. These DACs are capable of driving 8kohm loads and thus are not intended to drive speakers except through an external amplifier.

J31 - **1/8" Stereo Phono Jack** - This 1/8" inch jack provides an amplified analog stereo output of the CS4922 DAC's. It can be used to drive headphones if desired.

POWER SUPPLY CIRCUITRY

The schematic diagram in Figure 9 shows the power supply circuitry for the CDB4922. If the card is in the ISA slot it derives all voltages from this connection and NO VOLTAGE SHOULD BE APPLIED TO BINDING POSTS J20, J21, J23, J25 AND J26. These connectors are for stand alone mode alone. The following table describes the different power supply configurations and is applicable to both ISA and stand alone operation.

4922 Digital	4922 Analog	J22	R33	FB1
+5V	+5V	Not Installed	Installed	Installed
+5V	+5V Reg	Installed	Not Installed	Installed
+5V Reg	+5V Reg	Installed	Installed	Not Installed

+5V - 5 volt digital power shared by entire board.

+5V Reg - 5 volt power coming from on board voltage regulator. Connected only to CS4922 so it is cleanest supply available.

4922 Digital - refers to the CS4922 digital power pins

4922 Analog - refers to the CS4922 analog power supply pins.

If it is desirable to have different grounds for the digital and analog planes then both FB3 and FB4 should be removed. If these planes are to be the same then only one of FB3 or FB4 should be installed as to prevent a ground loop.

APPENDIX A: SOFTWARE TOOLS FOR THE CDB4922

A variety of software tools is shipped with the CDB4922 to allow the user to communicate and evaluate the CS4922. These tools are DOS based and can be run with the card plugged into the ISA slot or hooked up to the parallel port of a personal computer running DOS. For all programs the default address is 0x340 in the ISA bus but the program will reconfigure itself to run in standalone if one of the parallel port address' is entered on the command line. The program will display what address it is writing to upon execution. If any of the programs do not work, verification of this address is the first step in troubleshooting.

CDB22_LD.EXE

CDB22_LD.EXE is a program that boots the CS4922 and downloads the microcode. Its usage is as follows:

Usage: *cdb22_ld <input_file.sim> [-pxxx]*
xxx = parallel port address (0x278, 378 or 3bc)
or ISA bus address (0x260, 280, 320, or 340)*
** = default*

For example if you wanted to load ALLD0307.SIM to the part through the parallel cable hooked up to a parallel port at address 0x378 you would type:

cdb22_ld alld0307.sim -p378

This program reads the REQ line after download to make sure that the download was successful. If for some reason (no power, no part, no clock, no cable etc..) the download was unsuccessful the program will respond back with the message:

Download Failure

CDB22CMD.EXE

CDB22CMD.EXE is a program that sends commands to the CS4922 in the I2C format. This transmission can be watched on the SDATA and SCL lines of the CS4922 to see correct I2C write. Its usage is as follows:

Usage: *cdb22cmd <CS4922 Command> [-pxxx] [-my]*
xxx = parallel port address (0x278, 378 or 3bc)
or ISA bus address (0x260, 280, 320, or 340)*
y = single or double message mode (1 or 2)*
** = default*

The message mode can be selected from the command line through the switch -m. For example if you had downloaded ALLS0307.SIM (single messaging code) and wanted to send MSG_CM1_INIT through the parallel at address 0x3BC the syntax would be:

cdb22cmd 220000 -p3bc -m1

If for some reason this program does not receive an ACK from the CS4922 it will respond back with a message:

Error Sending Message Byte

CDB22_RD.EXE

CDB22_RD.EXE is a program that read back responces from the CS4922. This transmission can be watched on the SDATA and SCL lines of the CS4922 to see a correct I2C read. This program polls on the REQ line of the CS4922 to determine if the CS4922 has a message to send. It should be noted that without sending a command via CDB22CMD.EXE that illicits a response this program will do nothing. Its usage is as follows:

Usage: *cdb22_rd [-pxxx]*
xxx = parallel port address (0x278, 378 or 3bc)
or ISA bus address (0x260, 280, 320, or 340)*
** = default*

22_PLAY

22_play is a program that will spool an MPEG file from the PC to the CS4922 through the CDB4922. It is interrupt driven with the XF line of the CS4922 causing interrupts to occur and consequently data to be downloaded and decoded. Its usage is as follows:

Usage: *22_play <mp2 file> [-px -my -sz -r -iw]*
-px : x = parallel port address (0x278, 378 or 3bc) or ISA bus address (0x260, 280, 320, or 340)
-my : y = single or double message mode (1 or 2)
-sz : z = PCR_INIT, PCR_INIT = 0 to 1ffffffff
-r : no soft reset sent
-iw : interrupt (5, 7, 9, 10, 12, 15)*

The program expects microcode to be loaded to the CS4922 already and will automatically send the appropriate messages to the part for initialization. The switches -p and -m control the address and message mode respectively. The switch -s allows a PES file to be played with the number following the -s being the initial PCR of the file. The -r tells the program not to send a software reset after the file is finished. The switch -i instructs the program which interrupt to use.

• Notes •


SMART
Analog
™