A DIVISION OF CIRRUS LOGIC

CRD4912

Reference Design for CS4912

Features

- Complete Evaluation Platform for CS4912 and Associated Application Firmware
- 2 Analog Audio Inputs, 4 Analog Outputs
  - — Input from CS4222 Codec or CS5330 ADC
  - — Output from CS4912 and/or CS4222 Codec
  - — LFE circuit provides 5th Output for Subwoofer
- S/PDIF Digital Audio Input and Outputs
- Host Control Port allows Initialization & Control from Personal Computer
- Stand Alone Operation with On-board Switches & Indicators
  - — CS4912 Auto-boots from EEPROM
- CRD4912 Application Firmware Kits:
  - — Car Audio Kit with Crossover Filters, Graphic/Parametric EQ's, Compressor, Limiter, Noise Gate, 4-channel Time Delay
  - — Dolby® Surround ProLogic™ Kit with 3D Virtual Surround for 2 Speaker Playback
  - — Musical Instrument Reverb/Effects Kit
- Complete Design Documentation Provided
  - — Schematics, Layout information, HDL for Programmable Logic

Description

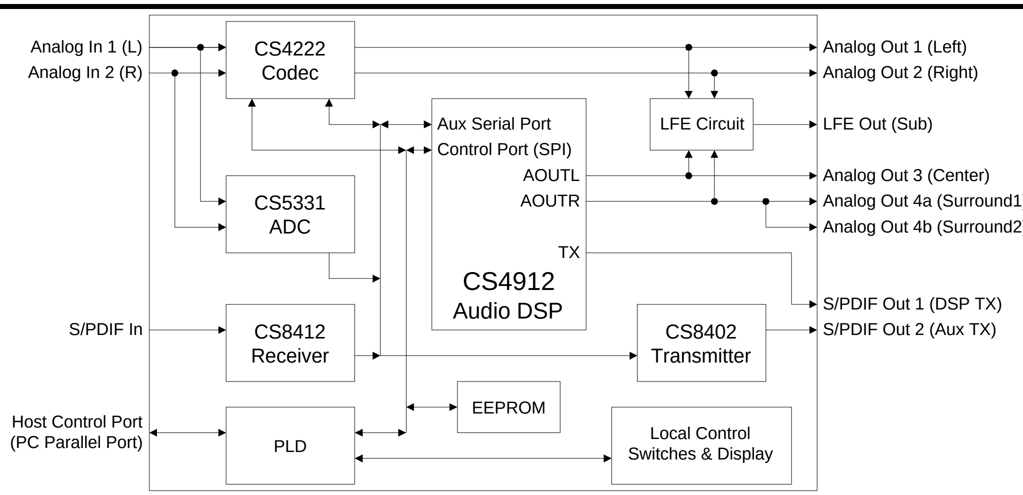
The CRD4912 is a full-featured evaluation platform for the CS4912 Audio Processor and associated DSP application firmware. This design is a superset of the circuitry required for typical CS4912 applications. Complete documentation is provided to allow designers to extract the circuitry needed for their specific applications.

The CRD4912 provides flexible analog and digital I/O options for the CS4912 device. The board can be booted and configured from a standard PC through the CRD4912 Host Control port, or the board may be operated in stand-alone mode. In stand-alone mode the CS4912 is automatically booted from an on-board EEPROM device, and the board is controlled using on-board switches and 7-segment displays.

Several application firmware kits, including a Car Audio kit, a Dolby Pro Logic decoder kit with 3D virtual surround capability, and a Reverb/Effects kit are available for the CRD4912. Each kit includes DSP firmware for the CS4912 device, plus a PC-based Graphic User Interface (GUI) application which allows remote control of the board to simplify demonstration and evaluation.

ORDERING INFORMATION

CRD4912-01	Reference Design Board
CF4912CAR-01	Firmware Kit, Car Audio
CF4912DPL-01	Firmware Kit, ProLogic Decode
CF4912EFF-01	Firmware Kit, Reverb/Effects



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

OVERVIEW

The CRD4912 provides a platform for demonstrating and evaluating the capabilities of the CS4912 part. The platform can be controlled and booted via a host computer parallel port, or can auto-boot from on-board EEPROM and be controlled via a simple user interface involving push buttons and LED indicators.

HARDWARE DESCRIPTION

The CRD4912 platform is designed around the 18 MIPS RAM-based CS4912 DSP engine. The CS4912 DSP includes an on-chip 16-bit stereo DAC and a S/PDIF transmitter. Stereo audio inputs are provided on two phono jacks. Analog to digital conversion (ADC) is provided by either a 20-bit CS4222 stereo codec or an 18-bit CS5330A/31A ADC. The ADC input to the DSP is jumper selectable.

Four channel digital to analog conversion (DAC) is provided by the stereo DAC on the CS4912 and the stereo DAC on the CS4222 codec. The right channel output of the CS4912 is sent to two buffered outputs for ease in connecting surround channels in Pro Logic applications. A low frequency effects (LFE) channel is derived from the analog outputs for connection to a subwoofer. Power amplifiers are provided for all 5 analog channels, and can be bypassed for high quality line level outputs.

Digital audio inputs to the DSP are provided via a CS8412 S/PDIF receiver and a digital audio port. Source selection between digital inputs is jumper selectable.

Digital outputs include the built-in S/PDIF transmitter on the CS4912 and an additional CS8402 S/PDIF transmitter. The DSP digital audio output is also available on the digital audio port. The platform can be booted up in either master (standalone) or in slave mode. In master mode, the DSP loads firmware from the on-board EEPROM during power up. In slave mode, the firmware must be downloaded by the host via a parallel port interface. Boot up mode is selected by a master/slave switch. Platform control is via on-board switches, or remotely by a host computer. The control source can be selected by the local/remote switch. A block diagram of the hardware platform is shown in Figure 1.

Master Mode

When the platform is in master (stand alone) mode, the DSP application firmware is loaded from an on board EEPROM immediately after reset. The DSP will begin executing the code automatically after the firmware load is completed. In master mode, the platform can be controlled by either the on board switches or a computer host via the parallel port.

DSP Boot

The boot process is always executed when the platform is in master mode and a hardware reset occurs. The boot process transfers data through the serial control port into program and data memory. This procedure is controlled by a program stored internally in ROM. When the CS4912 BOOT pin is held high during the rising edge of reset, the boot ROM takes control of the serial control port. The first two bytes contain the starting address for the following block of data. The starting address is 13 bits with the 13^{th} bit specifying program or data memory. 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. Two bytes containing 0xFF and three bytes containing a check sum must follow the last block of data. If the checksum is verified, the ROM issues a DSP software reset and the CS4912 will begin executing the code. If the checksum is not verified, the DSP asserts nREQ and operation halts.

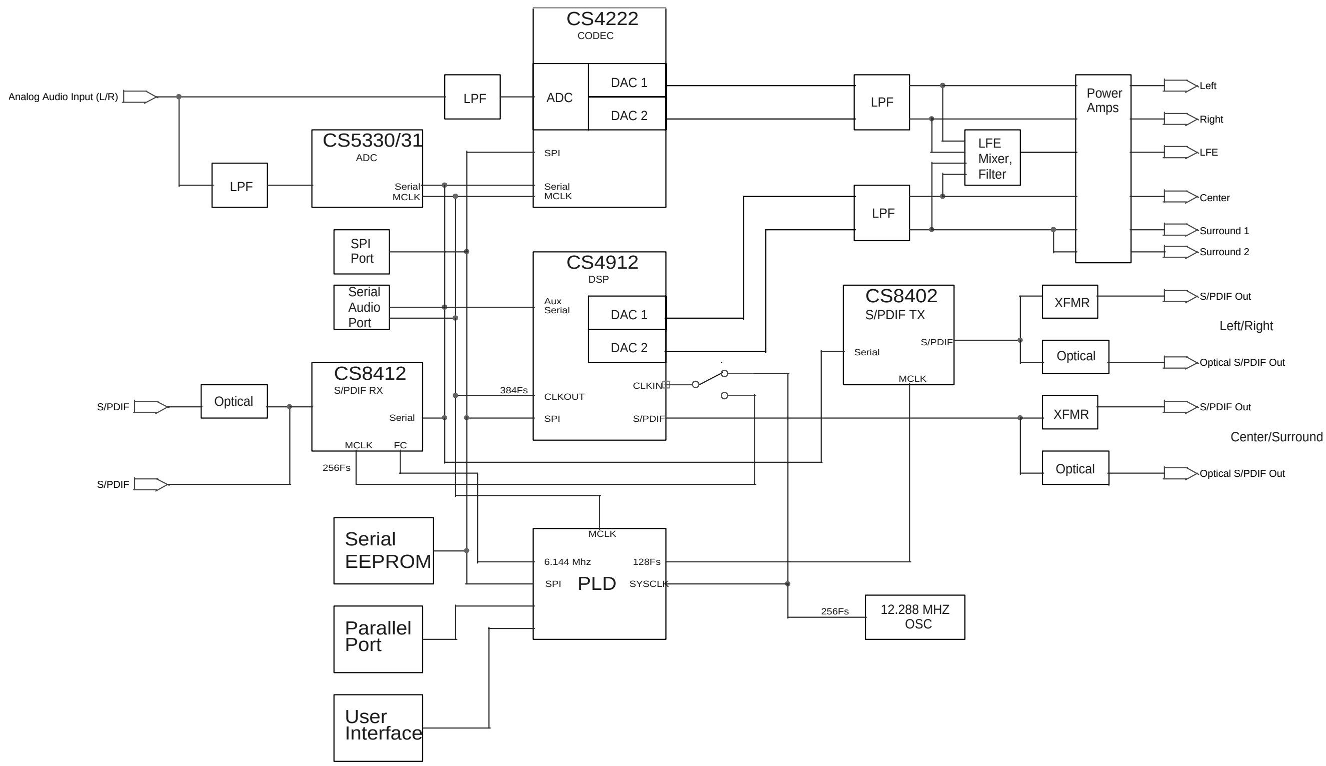


Figure 1. CRD4912 Block Diagram

Serial Bus Master

Since the CS4912 is a slave-only device, a serial bus master has been added to the system to arbitrate data transfer between the user interface, the EEPROM, the Codec, and the CS4912. The serial bus master generates the serial data clock, chip select signals, and some addressing data using the Motorola serial peripheral interface (SPI) protocol. During boot-up the bus master must load the firmware resident in the EEPROM into the DSP. After boot, the bus master must allow the DSP software to read and write data to the EEPROM, write commands to the codec, read user switch and hardware configuration information from the platform, and write status information to the platform. The complexity of the serial bus master has been reduced as much as possible by tightly integrating external logic with existing DSP functionality. The CS4912 general purpose outputs XF[4:1] along with nREQ act as commands to the serial bus master. A list of commands is shown in Table 1.

The bus master must be transparent when the platform is in slave mode to prevent bus contention with commands from the parallel port. The bus master will be disabled during the boot process if the platform is in slave mode. The SBM will become active after boot if the platform is in local mode. The initial state of the DSP software parameters (volume, tone, balance, etc.) can be stored into EEPROM by the user by pressing an unusual combination of user interface buttons (i.e. simultaneously depressing select, up and down buttons).

Boot Process

During the boot process, the EEPROM is initialized by the SBM, and data is clocked out of the EEPROM into the CS4912. The first byte stored in the EEPROM is the CS4912 address (0) with the read/write bit cleared (write), which initializes the

XF[4:1]	nREQ	Command	Description
000X	0	STATUS_CMD	Read Status Byte
000X	1	END_CMD	Terminate CONTROL1, or EEPROM_RET commands
001X	0	EEPROM_W_CMD	DSP Data -> EEPROM
010X	1	CONTROL1_CMD	SBM Control 1 Data -> DSP
011X	1	EEPROM_RET_CMD	SBM EEPROM Data -> DSP
100X	0	PARAM_CMD	DSP Parameter Data -> SBM
100X	1	END_BOOT_CMD	End Boot Process
101X	0	EEPROM_R_CMD	EEPROM Data -> DSP
110X	0	CODEC_CMD	DSP Data -> Codec
111X	0	EEPROM_SR_CMD	EEPROM Status Data -> DSP

Table 1. Serial Bus Master Commands

DSP. The SBM continues to clock data out of the EEPROM until the boot up process is terminated by the DSP software setting the XF4 pin high. If the boot ROM check sum verification fails, the nREQ pin is asserted low and boot ROM execution halts. The bus master will not respond in this state, and a hardware reset must be issued by the user to attempt to successfully reload the DSP. The pseudo code in Figure 2 shows the sequence of events during the boot procedure.

Slave Mode

The platform can be configured via a switch to boot in slave mode where the platform is programmed by a host computer via a parallel port interface. In slave mode, the host can download code directly into the CS4912 RAM, or the serial EEPROM, as well as write to the codec. The SBM boot function is disabled during boot and the SBM functionality is provided by the host. The platform can be controlled by either the host or the on board switches in slave mode.

Local Mode

In local mode the platform accepts control from the on board switches, and commands from the parallel port are ignored. Read capability of the parallel port is still retained.

```
// initial state to enable boot ROM in SPI mode
nRESET = 0
BOOT = 1
4912_CSn = 0
EEPROM_CSn = 1

// come out of reset
nRESET = 1
4912_CSn = 1

// initialize EEPROM
EEPROM_nCS = 0
send read command 0x03
send start address 0x0000

// clock data into CS4912
BOOT = 0
4912_nCS = 0
// first byte of EEPROM data is the CS4912 address
// with the R/nW bit cleared (0x00)
while (XF4 = 0) do
    continue to clock data
end while

// Terminate read
DSP software sets XF4
4912_nCS = 1
EEPROM_nCS = 1
```

Figure 2. Boot Procedure

User Control

Mode Switches and LEDs

Four slide switches may be read by the DSP software to select operating modes or other application specific functions. There are also four green LED indicators adjacent to the mode switches which are controlled by the DSP software.

Parameters

There is a total of 16 parameters that can be changed by the user. The current active parameter can be changed by a parameter select momentary switch. The currently selected mode is displayed on a 7-segment LED display. On power up, parameter 0 is the active parameter.

Parameter Value

The value of the active parameter can be incremented or decremented by the user via two momentary switches; up and down. The value of the currently selected parameter is displayed as two BCD nibbles on two 7-segment displays. The DSP parameters are 16 24-bit values, so the 100 levels displayed are dependent on the DSP software implementation. After the user increments a parameter, the parameter display is updated by the DSP.

Master/Slave and Reset

A slide switch selects whether the platform comes out of reset in master (stand alone) or slave mode. A red LED indicator is lit when the platform is in slave mode. A reset momentary switch is also provided that performs a hardware reset on the platform. A red LED is provided to indicate if reset is asserted.

Local/Remote

A slide switch selects whether the DSP is controlled by the onboard switches (local mode), or commands from the parallel port interface (remote mode). Although some incorrect behavior may be observed, control can be changed between the two modes without rebooting the platform. A red LED indicates when the board is in remote mode.

Speaker/Line Level Outputs

Jumpers are provided to bypass the on-board power amplifiers to maximize audio quality at the outputs.

Writing User Control Data

The user control switches are polled by the DSP software. The state of the user switches is latched into a shift register and clocked into the CS4912 via the SPI interface when XF[4:1] = 010X and nREQ = 1. The command is terminated by setting XF[4:1] = 0. The switch scan rate should be between 0.05 and 0.1 sec. The default state of the momentary switches (parameter select, up, down) is set (high). S7 indicates the sample rate of the platform and is updated automatically by data from the S/PDIF receiver and the state of the MCLK select signal to the SBM.

Byte 1	Byte 2
DSP Address, 0	Control Data

Control Data Write Sequence

Control Data

7	6	5	4	3	2	1	0
SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0

SW0	Parameter Select (Select)		
SW1	Decrement parameter data (Down)		
SW2	Increment parameter data (Up)		
SW3	Mode 0		
SW4	Mode 1		
SW5	Mode 2		
SW6	Mode 3		
SW7	Fs	= 0	48 kHz sample rate
		= 1	44.1 kHz sample rate

Reading Status Data

The status data byte containing mode and parameter data is read out of the CS4912 by the serial bus master and latched. Data transfer is initiated by the DSP when XF[4:1] = 000X and nREQ = 0.

Byte 1	Byte 2
DSP Address, 1	Mode/Parameter

Mode/Parameter Write Sequence

Mode/Parameter Data

7	6	5	4	3	2	1	0
M3	M2	M1	M0	P3	P2	P1	P0

The upper nibble of the Mode/Parameter byte controls the state of LED indicators adjacent to mode slide switches. The lower nibble of the byte containing the active parameter data is latched into a 7-segment display driver. The display driver displays hex alphanumerics for values greater than 9. P0 is the least significant bit. The function of the mode LED's and the parameter assignments are application dependent.

M0 Mode 0 switch indicator

M1 Mode 1 switch indicator

M2 Mode 2 switch indicator

M3 Mode 3 switch indicator

Reading Parameter Data

The parameter data byte is read out of the DSP by the serial bus master and latched into two 7-segment display drivers. Data transfer is initiated by the DSP when XF[4:1] = 100X and nREQ = 0.

Byte 1	Byte 2
DSP Address, 1	Parameter Data

Parameter Read Sequence

Parameter Data

7	6	5	4	3	2	1	0
PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

PD[3:0] is sent to the least significant 7-segment display and PD[7:4] is sent to the most significant display. The nibbles are BCD encoded data.

EEPROM Writes

Data transfer from the DSP to the EEPROM is initiated when XF[4:1] = 001X and nREQ = 0. The SBM addresses the DSP with the R/W bit set (read). The SBM then releases the data bus lines and asserts the EEPROM chip select. The DSP writes the EEPROM command byte and two start address bytes followed by one or more data bytes. An unlimited number of bytes can be transferred in a single session as long as nREQ = 0. When nREQ = 1, the session is terminated and the chip selects of both devices are deasserted.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5		Byte N
DSP Address, 1	EEPROM Command	MSB Address	LSB Address	Data 0	…	Data (N-5)

EEPROM Write Sequence

Data	Command
0x06	Enable Write Operations
0x04	Disable Write Operations
0x05	Read Status Register
0x01	Write Status Register
0x03	Read Data
0x02	Write Data

Table 2. EEPROM Commands

EEPROM Reads

Data is read from the EEPROM via a two step process. First data is read from the EEPROM and written to a latch in the SBM, then the data is retrieved from the SBM by the DSP in a separate operation. The EEPROM read cycle is initiated when XF[4:1] = 101X and nREQ = 0. The DSP is addressed by the SBM with the R/W bit high (read). The SBM then releases the data bus lines and asserts the EEPROM chip select and begins clocking data between the two devices. The DSP sends the EEPROM command byte and two EEPROM start address bytes. The DSP then writes an extra dummy data byte to the EEPROM. During the period the dummy data is being clocked out of the DSP, valid data is clocked into the SBM (and to the DSP, which can not be accessed). The data transfer is then completed when nREQ = 1, the EEPROM data is latched into the SBM, and the chip selects of the EEPROM and DSP are deselected. The EEPROM data is stored in the SBM for later retrieval by the DSP.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
DSP Address, 1	EEPROM Command	MSB Address	LSB Address	Dummy Data

EEPROM Read Sequence

The EEPROM data is retrieved from the SBM and written into the DSP by setting XF[4:1] = 011X and nREQ = 1. The SBM addresses the DSP with the R/W bit cleared (write). The previously stored EEPROM data byte is then clocked into the DSP. The process is terminated by setting XF[4:1] = 0. Only one byte at a time can be retrieved from the EEPROM by this command sequence. Reading the EEPROM data from the SBM is non-destructive.

Byte 1	Byte 2
DSP Address, 0	EEPROM Data

EEPROM Data Retrieve Sequence

Codec Writes

Data is transferred from the DSP to the CS4222 codec when XF[4:1] = 110X and nREQ = 0. The DSP is addressed by the SBM with the R/W bit high (read). The SBM then releases the data bus lines and asserts the codec chip select and begins clocking data between the two devices. The SBM will continue to clock data between the DSP and the codec as long as nREQ = 0. When nREQ = 1, the session is terminated and the chip selects of both devices are deasserted. Note that the first byte sent to the codec by the DSP is the codec address, 001000 with the read/write bit cleared.

Byte 1	Byte 2	Byte 3	Byte 4		Byte N
DSP Address, 1	Codec Address, 0	Data 0	Data 1	…	Data (N - 3)

Codec Write Sequence

Remote Mode

In remote mode the platform accepts control from the parallel port and all on board switch commands are ignored. The local/remote switch is polled once after a reset. Changing modes is only allowed during a reset condition. The following describes the host interface.

Parallel Port Interface

The parallel port interface is buffered and connected to a programmable logic device. The interface uses addressed I/O to maximize the port functionality. Four bytes of input data and 4 nibbles of output data are available to the platform. Control signals consist of a hardware reset, a data strobe, two address lines, and an acknowledge/service request. The interface definition is shown in Table 3. Parallel Port Pin Assignment.

Pin #	Computer Function	Type	Platform Function	Type	Description
1	nSTROBE	O	nSTROBE	I	data latch
2	D0	O	D0	I	general purpose data inputs
3	D1	O	D1	I	“”
4	D2	O	D2	I	“”
5	D3	O	D3	I	“”
6	D4	O	D4	I	“”
7	D5	O	D5	I	“”
8	D6	O	D6	I	“”
9	D7	O	D7	I	“”
10	nACK	I	nACK	O	acknowledge or service request
11	nBUSY	I	S0	O	general purpose data output
12	PERROR	I	S1	O	“”
13	SELECT	I	S2	O	“”
14	nAUTOFEED	O	RESET	I	board level reset
15	nERROR	I	S3	O	general purpose data output
16	nINIT	O	ADDR0	I	address bit 0
17	nSELECTIN	O	ADDR1	I	address bit 1
18 - 25	GND	-	GND	-	

Table 3. Parallel Port Pin Assignment

Four registers are used to latch data from the parallel port. The addressing assignments and bit definitions are shown in Table 4.

ADDR[1:0]	D[7:0] Destination	S[3:0] Source
00	Data 0	Status 0
01	Data 1	Status 1
10	Data 2	Status 2
11	Data 3	Reserved

Table 4. Parallel Port Address Assignment

Data 0

7	6	5	4	3	2	1	0
4912_CSn	4222_CSn	EE_CSn	MISO_DIR	MOSI_DIR	MISO_OUT	MOSI_OUT	SPICLK

SPICLK SPI clock

MOSI_OUT SPI MOSI output

MISO_OUT SPI MISO output

MOSI_DIR SPI MOSI direction
0 = input
1 = output

MISO_DIR SPI MOSI direction
0 = input
1 = output

EE_nCS EEPROM chip select, active low

4222_nCS CS4222 chip select, active low

4912_nCS CS4912 chip select, active low (must be cleared during reset to set SPI mode.)

Data 0 initial state = 0110 0000

Data 1

7	6	5	4	3	2	1	0
DISPLAY7	DISPLAY6	DISPLAY5	DISPLAY4	DISPLAY3	DISPLAY2	DISPLAY1	DISPLAY0

DISPLAY[7:0] 7-segment display data.

Data 1 initial state = 0000 0000

Data 2

7	6	5	4	3	2	1	0
res	DISPLAY_LE2n	DISPLAY_LE1n	DBDA_DIR	DBDA_OUT	DBCLK	PIO_DIR	PIO_OUT

PIO_OUT DSP general purpose I/O data output

PIO_DIR PIO direction control
0 = input
1 = output

DBCLK CS4912 debug port clock

DBDA_OUT Debug port data output

DBDA_DIR Debug port data direction
0 = CS4912 -> DBDA_IN
1 = DBDA_OUT -> CS4912

DISPLAY_LE1n 7-segment display latch enable, active low. Controls the active parameter display and the LSB of the parameter data display.

DISPLAY_LE2n 7-segment display latch enable, active low. Controls the MSB of the parameter data display.

res Reserved

Data 2 initial state = X110 0010

Data 3

7	6	5	4	3	2	1	0
res	res	res	res	res	res	4912_BOOT	DAP_CSn

4912_BOOT CS4912 boot pin. The state of this pin during reset determines if the boot ROM code is in control of the serial port. This bit is not cleared on reset.
0 = boot ROM disabled
1 = boot ROM enabled

DAP_CSn External SPI port chip select.
0 = enabled
1 = disabled

Data 3 initial state = XXXX XXX0

Status 0

3	2	1	0
res	nREQ	MISO_IN	MOSI_IN

MOSI_IN SPI MOSI data input

MISO_IN SPI MISO data input

nREQ CS4912 service request output, active low

res Reserved

Status 1

3	2	1	0
FS	S_MODE	MASTER	LOCAL

LOCAL — Local/remote switch
0 = host control
1 = local control

MASTER — Master/Slave switch
0 = master mode (auto boot)
1 = slave mode

S_MODE — Serial Audio Mode
0 = I^2S
1 = Left Justified

FS — Platform Sample rate
0 = 48 kHz
1 = 44.1 kHz

Status 2

3	2	1	0
res	res	PIO_IN	DBDA IN

DBDA IN — debug port input

PIO_IN — CS4912 general purpose I/O pin input

res — Reserved

Status 3

3	2	1	0
XF4	XF3	XF2	XF1

Please see Table 1. Serial Bus Master Commands for further information on XF[4:1].

CS4912 Programming

A code and data image can be directly downloaded into the part by a DOS or GUI based host program. The SBM acts as a master in communicating with the DSP (MOSI is the data output, MISO is data input). The program must reset the CRD4912 platform into SPI mode with the 4912_BOOT bit set before the image is sent to the device. The entire memory map of the CS4912 is available to be programmed. Monitoring of the nREQ and XF[4:1] pins is provided so that firmware download success can be determined. See the section on Boot Process in this document for more information.

EEPROM Programing

The host can download firmware into the serial EEPROM via the parallel port. In this case, the SBM acts as a slave (MOSI is data input, MISO is data output). The EEPROM is selected by asserting EE_CSn low, sending a EEPROM command byte (block write) followed by a two byte start address (MSB first). The EEPROM supports 32 byte block writes i.e. 32 bytes can be written before EE_CSn must be deasserted. After programming the EEPROM, the host should execute the EEPROM command to disable writes to prevent possible corruption of the memory. Please see the section on EEPROM Writes in this document for more information.

Debug Port

Access to the bidirectional debug port is provided by the DBDA_OUT, DBDA_IN, DBDA_DIR, and DBCLK bits.

DSP Control

The host can control the platform using a messaging protocol when in remote mode, or the on-board switches can be used when in local mode.

Messaging Protocol

After a successful boot, the bus master transfers control data into, and status or configuration data out of the CS4912. The data transfer protocol is message based. A message is composed of a message identifier byte followed by one or more data bytes. The number of data bytes is message dependent. The CS4912 signals the bus master that a message is ready to be sent by asserting nREQ low. If nREQ is high, the CS4912 is ready to receive a message. For more information on the messaging protocol, see the relevant firware kit documentation.

EEPROM

Size

The CS4912 RAM is organized as a 5k x 24-bit program memory and a 3k x 24-bit data memory. The maximum download image size is therefore 24k x 8 bits, The maximum serial EEPROM size is currently 128k bits organized as 16k x 8 bits, which will limit the current program + data image size to 5.33k words.

Organization

The EEPROM memory contains program memory, data memory , and user configuration data. *The first byte of the EEPROM must be the CS4912 address with the read/write bit cleared (0x00).* Configuration data requires a minimum of 16 x 24-bit words reserved for parameter data, and one byte for mode data (49 bytes).

Address	Type
0000	DSP Address
0001 - XXXX	Program
XXXX - YYYY	Data
YYYY - ZZZZ	Config Data

Table 5. EEPROM Memory Map

Digital Audio

The CRD4912 platform provides a S/PDIF receiver, 2 S/PDIF transmitters, and a digital audio I/O port. The DSP digital audio source is chosen by JP1, JP2, and JP3. JP1 selects between the CS4222 ADC, or the CS5330 ADC. JP2 selects between the CS8412 S/PDIF receiver or the SAP_SDIN input from an external source. JP3 selects between an ADC or digital audio source. When the CS8412 is selected as the audio source, SW1 must also select the CS8412 to be the reference clock input to the DSP.

Master Clock

The master clock (MCLK) for the platform is always supplied by the DSP PLL output CLOCKOUT and is 384 Fs. The reference clock for the DSP PLL can be selected by SW1, and is either the onboard (12.288 MHz) oscillator, or the CS8412 S/PDIF receiver MCLK output. When the oscillator is selected, the XO_SEL jumper determines the sample frequency. XO_SEL is low when the platform runs at a 48 kHz sample rate (12.288 MHz) and high when the platform runs at a 44.1 kHz rate (11.2896 MHz). When the CS8412 MCLK output is the reference clock source, the platform will automatically sync to either 48 or 44.1 kHz sample rate.

The MCLK input to the CS5330/31A is anded with system reset to insure proper power up of the device.

Format

The CRD4912 platform can run in one of two serial audio formats, left justified and I^2S. The serial mode is hard coded in the DSP software. In I^2S mode, 2 channel 18-bit data is input to and output from the DSP. When using the CS4222 as the ADC source in I^2S mode, the lower 2 bits are discarded. In left justified mode, 4 channel 20-bit data is input to the DSP, and 6 channel, 20-bit data is output. The CS4222 defaults on power up into I^2S mode and must be commanded via the serial control port into left justified mode. The DSP can determine which serial format mode is requested by the user by testing the S_MODE bit in the control data byte. Serial audio data channel assignment is application dependent.

Digital Audio Port

Digital audio data I/O is provided by the digital audio port (DAP). The DAP buffers the DSP AUX serial data port and connects to a 20-pin header. The DAP_SDIN is a buffered input and can be selected as input to the DSP via header jumper JP2 and JP3.

Pin	Signal	Type
1	DAP_SDIN	I
3	AUXOUT	O
5	AUXLR	O
7	AUXCLK	O
9	MCLK	O
11	HALFOSC	O
13	VCC	P
2,4,6,8,10,12,14	GND	P
15 - 20	NC	-

Table 6. Digital Audio Port (DAP)

S/PDIF Input

Both consumer and optical S/PDIF inputs are provided by a CS8412 S/PDIF receiver. The platform will automatically lock to either 48 kHz or 44.1 kHz data streams. The onboard oscillator is divided by 2 to provide a 6.144 MHz reference frequency to the CS8412. The F[2:0] bits are then decoded by the PLD and sent as the Fs bit to the DSP. The serial output format of the CS8412 and CS8402 is controlled independently by the S_MODE switch. The serial mode of the DSP and codec are dependent on the software application code. The user is responsible for setting the S_MODE switch into the proper position.

S/PDIF Output

The CS4912 has a built in S/PDIF transmitter that is transformer isolated and connected to a phono

F[2:0]	Description	Fs
000	out of range	0
001	48 kHz ± 4%	0
010	44.1 kHz ± 4%	1
011	32 kHz ± 4%	1
100	48 kHz ± 400 ppm	0
101	44.1 kHz ± 400 ppm	1
110	44.056 kHz ± 400 ppm	1
111	32 kHz ± 400 ppm	1

Table 7. CS8412 Frequency Mapping

jack, and also sent to an optical transmitter. The DSP AUX port is also connected to a CS8402 S/PDIF transmitter which provides transformer and optically coupled output formats. The channel assignment is software dependent, but normal Pro-Logic code sends Left/Right data to the CS8402 and the CS4912 outputs the center/surround data. The serial input format of the CS8402 is controlled by the S_MODE switch and can be one of two values, left justified or I^2S. The 128 Fs MCLK input to the CS8402 is derived from the 384 Fs MCLK by first multiplying by 2, then dividing by 3 followed by a divide by 2 to give 50% duty cycle output.

Analog Audio I/O

On-board ADC's

Onboard A/D conversion will be provided by either an 18-bit CS5330/31A A/D converter or a 20-bit CS4222 codec. Both parts can be co-resident on the board, with serial data selection performed by JP2 and JP3. Full scale input to the platform is 5.6 Vpp. The input to the CS5330/31 is attenuated by 0.707 to give a full scale input of 4.0 Vpp.

The serial data output (SDATA) on the CS5330/31A is buffered due to the ~15k input impedance of AUXIN on the CS4912. Impedances less than 47k on SDATA at power up will cause the converter to enter master mode and begin sourcing LRCLK and SCLK resulting in bus contention.

Analog Output

The platform has seven analog output channels, Left, Right, Center, Surround1, Surround2, Mono, and Low Frequency Effect (LFE). The two surround outputs are mono and are derived from the same output channel. The Mono output is the inverted sum of Surround and Center. The CS4222 full scale input is 5.6 Vpp. Full scale output of the CS4222 is 5.76 Vpp (differential). Full scale output of the CS4912 is 2.88 V pp and is multiplied by 2 to give an output voltage of 5.76 Vpp. The CS4912 output is 3.0 Vpp and is multiplied by 2 to give a 6 Vpp output.

DAC

Two DAC's are present on the CS4912 and are used to output Center and Surround channel audio. Two more DAC channels may be present on the platform in a CS4222 codec. These two channels are used for Left and Right channel outputs. The CS4912 auxiliary digital audio port (AUX) is used to send the Left and Right synthesized data to the codec.

Output Filtering

Output filters for the Left, Right, Center, and Surround channels for the CS4912 and CS4222 are provided.

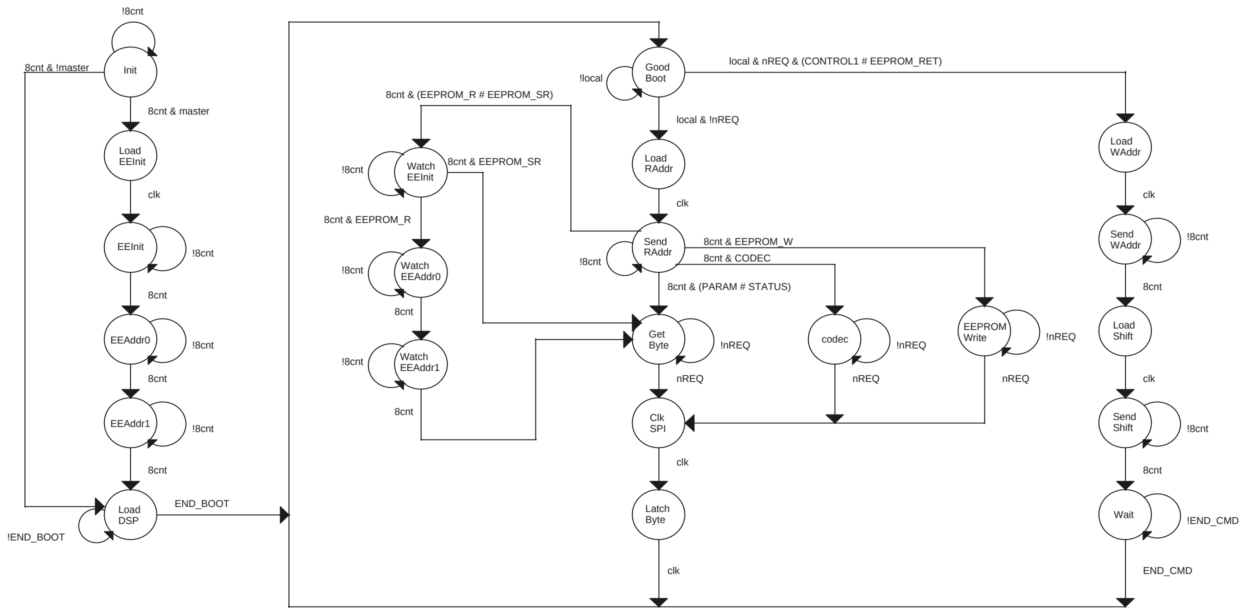
LFE Channel

A low frequency effects channel is derived from the sum of the left, right and center channels. The sum is filtered with a single pole low pass filter, preceded by a passive single pole high pass filter with the following characteristics:

F_{CL} = 100 Hz

F_{CH} = 3 Hz

Since the summing node is past the 4912/4222 balance/volume controls, there may be some issue with the LFE channel being fully Dolby complient. Provisions for summing all 4 DAC outputs are



CMD	nREQ	Description
000x	0	STATUS
000x	1	END
001x	0	EEPROM_W
010x	1	CONTROL1
011x	1	EEPROM_RET
100x	0	PARAM
100x	1	END_BOOT
101x	0	EEPROM_R
110x	0	CODEC
111x	0	EEPROM_SR

Figure 3. State Diagram, Serial Bus Master

made on the platform, only 3 of which should be used at a time to prevent clipping.

On-board power amplifiers

Power amplifiers are provided for 5 outputs (left, right, surround1, surround2, LFE). The power amplifiers are capable of approximately 1 W, and can be bypassed via jumpers.

Programmable Logic

A programmable logic device (PLD) is used to provide the SBM functions, the parallel port interface, and various system control signal interfaces. The PLD is implemented in an Altera EPM7128STC100-15 device which is in circuit programmable thru a 10 pin interface. The PLD master clock is the oscillator frequency divided by 8 (1.5 Mhz) which runs the SBM state machine. The state diagram for the SBM is shown in Figure 3. The SBM commands are described in Table 1. The PLD equations are shown in the Appendix.

Power Supply

The platform requires +12V DC and +5V DC supplies. Status LED’s are provided on both supply inputs which are protected from overvoltage/reverse polarity conditions by zener diodes. The +5V input is split into a digital and analog supply. All three supplies are filtered with ferrite beads. Bulk capacitance is also provided at the input to each supply.

Reset Control

A power up reset/power monitor circuit is provided that holds reset low for approximately 200 ms. Reset can be initiated by a momentary switch, the parallel port, or the power supply dropping below 4.40 V.

Mute

A mute circuit has been added to the analog outputs to suppress transients that occur during power up/down. The DACs will produce a 2.2V transient at power up due to the on-chip voltage references. The output op-amps will also produce power up/down transients on their outputs due to voltage reference and/or power supply slew rates or power supply voltage transients. The mute circuit consits of a shunt NPN transistor with a very low $V_{ce\ (sat)}$. When un-muted, the transistor must be prevented from conducting current when the collector swings negative or harmonic distortion will occur due to impedance modulation effects. The transistor can be prevented from conducting when the collector voltage becomes negative by not allowing any current to flow in the base, or holding the base at a lower voltage than the minimum collector voltage, which prevents the CB junction from becoming forward biased. A switched capacitor voltage inverter is used on the CRD4912 to provide a negative voltage to the base.

A relay or FET is placed in series with the +12V supply to prevent the output op-amps from powering up before the mute circuit becomes active.

DSP SOFTWARE REQUIREMENTS

Local Mode

The CS4912 DSP software is required to perform the following tasks when the platform is in Local control mode (in addition to the DSP application):

- Poll and decode control data.
  - select switch
  - up/down switches
  - mode[3:0] switches
  - sample frequency (44.1, 48 kHz) bit
- Write mode/parameter data to 7-segment display.
- Write parameter data to 2 7-segment displays.
- Store parameter settings in EEPROM.
- Read parameter settings from EEPROM.
- Set up the CS4222 if the platform is to run in left justified serial format.
- Support debug port access.

Remote Mode

The CS4912 DSP software is required to perform the following tasks when the platform is in remote control mode (in addition to the DSP application).

- Read PIO bit to determine if platform is in local or remote mode.
- Receive and decode messages from the host. See relevant applications firmware kit documentation for messaging protocol details.
- Support debug port access.

The DSP serial mode (SPI) is hard coded into the application software.

GUI REQUIREMENTS

All slave operating modes will be controlled by a unified program with a GUI. The GUI will control and display mode and parameter data. Other options the unified GUI will be able to support are:

- Hardware Reset
- Software Reset
- Download code into the CS4912
  - user can select file
- Download code into the EEPROM
  - user can select file
- Support messaging protocol commands
- Save local copies of parameters, option to save to EEPROM. Local copies are stored to disk.
- Restore factory default parameter tables to EEPROM and GUI.
- Write parameter table to EEPROM, user option to select location

- Read parameter tables from EEPROM
- Show serial mode configuration (left justified, I^2S)
- Polling of F_s bit for auto setup of sample rate. Also has manual select if parallel port interface is unreliable.
- Debug support

APPENDIX

```
TITLE “CRD4912B Embedded Controler v1.0”;

%
        File: h:\CRD4912\PLD\crd4912b.tdf
        Engineer: Wayne C. Wilson
        Revision: 1.0
        Revision Date: 8/14/98
        Revision Notes:
        v1.0: first Rev B version. Added nPFO, resetn inputs and resetctrln output to Rev A version.

        Design Notes: osc = 12.288 Mhz.

%


CONSTANT STATUS_CMD           = B”000X”;
CONSTANT EEPROM_W_CMD         = B”001X”;
CONSTANT CONTROL_CMD          = B”010X”;
CONSTANT EEPROM_RET_CMD       = B”011X”;
CONSTANT PARAM_CMD            = B”100X”;
CONSTANT EEPROM_R_CMD         = B”101X”;
CONSTANT CODEC_CMD            = B”110X”;
CONSTANT EEPROM_SR_CMD        = B”111X”;
CONSTANT EEINIT_DATA          = B”0000011”;
CONSTANT 4912RADDR_DATA       = B”0000001”;
CONSTANT ADDR_A               = B”00”;
CONSTANT ADDR_B               = B”01”;
CONSTANT ADDR_C               = B”10”;
CONSTANT ADDR_D               = B”11”;


SUBDESIGN crd4912b
(
        sysresetn,
        osc,
        mclk,
        mclk_sel,
        xf[4..1],
        nReq,
        switch[6..0],
        f[2..0],
        s_mode,
        d[7..0],
        nStrobe,
        addr[1..0],
        local,
        xo_sel,
        4912tx,
```

```
        master,
        nPFO,
        resetn                  :INPUT;

        4912_CSn,
        4222_CSn,
        EE_CSn,
        DAP_CSn,
        SPIclk,
        dbclk,
        4912_boot,
        status[3..0],
        mode_led[3..0],
        display[7..0],
        display_len1,
        display_len2,
        halfosc,
        thirdmclk,
        nAack,
        8402_M2,
        8402_M0,
        buf_4912tx,
        buf_sysresetn,
        resetctrln              :OUTPUT;


        dbda,
        pio,
        mosi,
        miso                    :BIDIR;

)


VARIABLE
        oscdiv[2..0]            :DFF;
        mclkdiv[2..0]           :DFF;
        bitcnt[2..0]            :DFFE;
        8cnt                    :SOFT;
        shift[7..0]             :DFF;
        shiftin                 :DFF;
        load                    :SOFT;
        SPIdata[7..0]           :SOFT;
        mosi_tri                :TRI;
        miso_tri                :TRI;
        dbda_tri                :TRI;
        pio_tri                 :TRI;
        mode_latch[3..0]        :DFFE;
        eeprom[7..0]            :DFFE;
```

```
shiftena              :LCELL;
freq                  :SOFT;
port_latch_a[7..0]    :DFFE;
port_latch_b[7..0]    :DFFE;
port_latch_c[6..0]    :DFFE;
port_latch_d[1..0]    :DFFE;
mclkmult              :LCELL;
xf_B[4..1]            :DFFE;


sysclk,
display_len1_bit,
display_len2_bit,
display_bit[7..0],
pio_dir_bit,
pio_out_bit,
SPIclk_bit,
mosi_out_bit,
mosi_ena_bit,
miso_out_bit,
miso_ena_bit,
EE_CSn_bit,
dbda_out_bit,
dbclk_bit,
dbda_dir_bit,
DAP_CSn_bit,
4222_CSn_bit,
4912_CSn_bit,
4912boot_bit          :SOFT;


ss                    : MACHINE OF BITS (sstate[4..0])
                      WITH STATES (
                                 sInit,
                                 sLoadEEInit,
                                 sEEInit,
                                 sEEAddr0,
                                 sEEAddr1,
                                 sLoadDSP,
                                 sGoodBoot,
                                 sLoadWAddr,
                                 sSendWAddr,
                                 sLoadShift,
                                 sSendShift,
                                 sWait,
                                 sLoadRAddr,
                                 sSendRAddr,
                                 sGetByte,
                                 sLatchByte,
```

```
                                        sEEPROMWrite,
                                        sCodec,
                                        sClkSPI,
                                        sWatchEEInit,
                                        sWatchEEAddr0,
                                        sWatchEEAddr1
                                                    );

BEGIN

%
        ss state machine for controlling the SPI data bus.
%
        ss.clk = sysclk;
        ss.reset = !sysresetn;

        CASE ss IS
              WHEN sInit =>
                    IF (master & 8cnt) THEN
                          ss = sLoadEEInit;
                    ELSIF (!master) THEN
                          ss = sLoadDSP;
                    ELSE
                          ss = sInit;
                    END IF;
              WHEN sLoadEEInit =>
                    ss = sEEInit;
              WHEN sEEInit =>
                    IF (8cnt) THEN
                          ss = sEEAddr0;
                    ELSE
                          ss = sEEInit;
                    END IF;
              WHEN sEEAddr0 =>
                    IF (8cnt) THEN
                          ss = sEEAddr1;
                    ELSE
                          ss = sEEAddr0;
                    END IF;
              WHEN sEEAddr1 =>
                    IF (8cnt) THEN
                          ss = sLoadDSP;
                    ELSE
                          ss = sEEAddr1;
                    END IF;
              WHEN sLoadDSP =>
                    IF (nReq & xf_B[].q == PARAM_CMD) THEN
                          ss = sGoodBoot;
                    ELSE
```

```
                ss = sLoadDSP;
            END IF;
        WHEN sGoodBoot =>
            IF (local & nReq & (xf_B[].q == CONTROL_CMD # xf_B[].q == EEPROM_RET_CMD)) THEN
                        ss = sLoadWAddr;
            ELSIF (local & !nReq) THEN
                ss = sLoadRAddr;
            ELSE
                ss = sGoodBoot;
            END IF;
        WHEN sLoadWAddr =>
            ss = sSendWAddr;
        WHEN sSendWAddr =>
            IF (8cnt) THEN
                ss = sLoadShift;
            ELSE
                ss = sSendWAddr;
            END IF;
        WHEN sLoadShift =>
            ss = sSendShift;
        WHEN sSendShift =>
            IF (8cnt) THEN
                ss = sWait;
            ELSE
                ss = sSendShift;
            END IF;
        WHEN sWait =>
            IF (xf_B[].q == STATUS_CMD) THEN
                ss = sGoodBoot;
            ELSE
                ss = sWait;
            END IF;
        WHEN sLoadRAddr =>
            ss = sSendRAddr;
        WHEN sSendRAddr =>
            IF (8cnt & (xf_B[].q == STATUS_CMD # xf_B[].q == PARAM_CMD)) THEN
                ss = sGetByte;
            ELSIF (8cnt & xf_B[].q == CODEC_CMD) THEN
                ss = sCodec;
            ELSIF (8cnt & xf_B[].q == EEPROM_W_CMD) THEN
                ss = sEEPROMWrite;
            ELSIF (8cnt & ((xf_B[].q == EEPROM_R_CMD) # (xf_B[].q == EEPROM_SR_CMD))) THEN
                ss = sWatchEEInit;
            ELSE
                ss = sSendRAddr;
            END IF;
        WHEN sGetByte =>
            IF (nReq) THEN
                ss = sClkSPI;
```

```
            ELSE
                  ss = sGetByte;
            END IF;
      WHEN sLatchByte =>
            ss = sGoodBoot;
      WHEN sCodec =>
            IF (nReq) THEN
                  ss = sClkSPI;
            ELSE
                  ss = sCodec;
            END IF;
      WHEN sEEPROMWrite =>
            IF (nReq) THEN
                  ss = sClkSPI;
            ELSE
                  ss = sEEPROMWrite;
            END IF;
      WHEN sClkSPI =>
            ss = sLatchByte;
      WHEN sWatchEEInit =>
            IF (8cnt & xf_B[].q == EEPROM_R_CMD) THEN
                  ss = sWatchEEAddr0;
            ELSIF (8cnt & xf_B[].q == EEPROM_SR_CMD) THEN
                  ss = sGetByte;
            ELSE
                  ss = sWatchEEInit;
            END IF;
      WHEN sWatchEEAddr0 =>
            IF (8cnt) THEN
                  ss = sWatchEEAddr1;
            ELSE
                  ss = sWatchEEAddr0;
            END IF;
      WHEN sWatchEEAddr1 =>
            IF (8cnt) THEN
                  ss = sGetByte;
            ELSE
                  ss = sWatchEEAddr1;
            END IF;
      WHEN OTHERS =>
            ss = sInit;
      END CASE;

%
        control signal definition. default level is high, a bit is low when
        its corresponding switch is depressed.

        parameter select= switch0
        decrement parameter= switch1
```

```
increment parameter = switch2
mode 0                      = switch3
mode 1                      = switch4
mode 2                      = switch5
mode 3                      = switch6


hardware control signals
SPIclk          output is inverted phase from shift register clock.
*_CSn           SPI device chip selects, active low
DAP_CSn         external SPI device chip select.
4912_boot  must be high during reset to enable boot ROM.
nAack           used to notify host for request for service, active low.
dbda            CS4912 debug port data I/O pin.
dbclk           CS4912 debug port clock.
resetctrln tied to the MR pin of the MAX708 reset generator. AND’s parallel port and PFO resets.
%

SPIclk = master & !sysclk & sysresetn & (sEEInit # sEEAddr0 # sEEAddr1 # sLoadDSP)
                # !master & sLoadDSP & sysresetn & SPIclk_bit
                # local & !sysclk & !sLoadDSP & !sGoodBoot & !sLoadShift
                & !sLatchByte & !sLoadRAddr & !sLoadWAddr & !sWait
                # !local & sGoodBoot & SPIclk_bit;


4912_CSn = !(master & sLoadDSP
                     # !master & sLoadDSP & !4912_CSn_bit
                     # local & !(!sSendWAddr & !sSendRAddr & !sLoadShift & !sSendShift
                     & !sGetByte & !sEEPROMWrite & !sCodec & !sClkSPI & !sWatchEEInit
                     & !sWatchEEAddr0 & !sWatchEEAddr1 & !sLatchByte & !sWait)
                     # !local & sGoodBoot & !4912_CSn_bit
                     # sInit
                );

4222_CSn = !(local & (sCodec # sClkSPI & xf_B[].q == CODEC_CMD)
                     # !local & sGoodBoot & !4222_CSn_bit
                );

EE_CSn = !( master & (sEEInit # sEEAddr0 # sEEAddr1 # sLoadDSP)
                # !master & sLoadDSP & !EE_CSn_bit
                # local & (sEEPROMWrite # sWatchEEInit # sWatchEEAddr0
                           # sWatchEEAddr1
                           # xf_B[].q == EEPROM_W_CMD & (sClkSPI # sLatchByte)
                           # xf_B[].q == EEPROM_R_CMD & (sGetByte # sClkSPI # sLatchByte)
                           # xf_B[].q == EEPROM_SR_CMD & (sGetByte # sClkSPI # sLatchByte)
                           )
                # !local & sGoodBoot & !EE_CSn_bit
           );
```

```
DAP_CSn = local # !local & DAP_CSn_bit;
4912_boot = master & sInit # !master & 4912boot_bit;
nAack = nReq;
buf_sysresetn = GLOBAL(sysresetn);
buf_4912tx = 4912tx;

dbda_tri.in = dbda_out_bit;
dbda_tri.oe = dbda_dir_bit;
dbda = dbda_tri.out;
dbclk = dbclk_bit;

resetctrln = resetn & nPFO;



%
    SPI MOSI tri-state buffer. local/remote switch selects the data source; SBM or host.
    disabled during DSP to EEPROM or codec communications in local mode, or by the mosi_ena_bit
    in remote mode.
%
    mosi_tri.in = !master & sLoadDSP & mosi_out_bit
                    # local & !sLoadDSP & shift7.q
                    # !local & sGoodBoot & mosi_out_bit;
    mosi_tri.oe = !master & sLoadDSP & mosi_ena_bit
                    # local & !(!sSendRAddr & !sSendWAddr & !sLoadShift
                          & !sSendShift & !((sGetByte # sClkSPI)
                          & (xf_B[].q != EEPROM_R_CMD) & (xf_B[].q != EEPROM_SR_CMD)
                          & xf_B[].q != CODEC_CMD))
                    # !local & sGoodBoot & mosi_ena_bit;
    mosi = mosi_tri.out;



%
    SPI MISO tri-state buffer. Used as an output during EEPROM access,
    all other times as an input.
%
    miso_tri.in = master & sEEInit & shift7.q
                    # !local & sGoodBoot & miso_out_bit;
    miso_tri.oe = master & !(!sEEInit & !sEEAddr0 & !sEEAddr1)
                    # !local & sGoodBoot & miso_ena_bit;
    miso = miso_tri.out;



%
    freq indicates the platform sample rate. The f[2..0] bits from the CS4912
    are decoded along with mclk_sel and xo_sel. mclk_sel indicates the source
    of mclk. xo_sel indicates the frequency of the on board oscillator.

    freq        = 1 Fs = 44.1 kHz
                = 0 Fs = 48 kHz.
```

```
        mclk_sel    = 0 on board oscillator
                    = 1 CS8412 mclk

        xo_sel      = 0 48 kHz
                    = 1 44.1 kHz

        f[2..0] mclk_sel xo_selsample frequency      freq
         X         0     0                           0
         X         0     1                           1
         0         1     X     out of range          0
         1         1     X     48 khz +/- 4 percent  0
         2         1     X     44.1 khz +/- 4 percent 1
         3         1     X     32 khz +/- 4 percent  1
         4         1     X     48 khz +/- 400ppm     0
         5         1     X     44.1 khz +/- 400ppm   1
         6         1     X     44.056 khz +/- 400ppm  1
         7         1     X     32 khz +/- 400ppm     1

%
        freq = mclk_sel & f1 # mclk_sel & f2 & f0 # !mclk_sel & xo_sel;



%
        serial format select.
        s_mode      = 0        Format 4. I2S compatible, 2 CH in, 2 CH out, SCLK = 64 Fs
                    = 1        Format 1. left justified, 4 CH in, 6 CH out, SCLK = 128 Fs
%
        8402_M0 = s_mode;
        8402_M2 = !s_mode;



%
        PIO tri_state buffer. PIO is used for local/remote input to the DSP, can be used as a
        general purpose I/O in remote mode.
%
        pio_tri.in = local # !local & pio_out_bit;
        pio_tri.oe = local # !local & pio_dir_bit;
        pio = pio_tri.out;



%
        SPI shift register.
%
        shift[].clk = sysclk;
        shift[].clrn = GLOBAL(sysresetn);
        shift[].ena = !shiftena;
        shift[7..1].d = load & SPIdata[7..1] # !load & shift[6..0].q;
        shift0.d = load & SPIdata0 # !load & shiftin.q;
```

```
        shiftena = !load & !sEEInit & !sSendWAddr & !sSendRAddr & !sSendShift
                        & !sGetByte & !sClkSPI;
        load = sLoadEEInit # sEEAddr0 # sEEAddr1 # sLoadRAddr # sLoadWAddr # sLoadShift;
        SPIdata[6..0] = sLoadEEInit & EEINIT_DATA
                              # sLoadRAddr & 4912RADDR_DATA
                              # sLoadShift & xf_B[].q == CONTROL_CMD & switch[]
                              # sLoadShift & xf_B[].q == EEPROM_RET_CMD & eeprom[6..0].q;
        SPIdata7 = sLoadShift & (xf_B[].q == CONTROL_CMD) & freq
                        # sLoadShift & (xf_B[].q == EEPROM_RET_CMD) & eeprom7.q;


%
        The SPI input is sampled on the rising edge of SPIclk and shifted into shift[] on the
        falling edge of SPIclk. miso is the normal input, except during an EEPROM_R_CMD or
        EEPROM_SR_CMD cycle where mosi becomes the input.
%
        shiftin.clk = !sysclk;
        shiftin.clrn = GLOBAL(sysresetn);
        shiftin.d = (xf_B[].q != EEPROM_R_CMD) & (xf_B[].q != EEPROM_SR_CMD) & miso
                              # (xf_B[].q == EEPROM_R_CMD) & mosi
                              # (xf_B[].q == EEPROM_SR_CMD) & mosi;


%
        SPI port bit counter.
%
        bitcnt[].clk = sysclk;
        bitcnt[].clrn = GLOBAL(sysresetn);
        bitcnt[].d = !sGoodBoot & (bitcnt[].q + 1);
        bitcnt[].ena = !sLoadEEInit & !sLoadDSP & !sLoadShift & !sLoadWAddr
                              & !sLoadRAddr & !sLatchByte & !sCodec & !sEEPROMWrite & !sClkSPI
                              & !sWait;

        8cnt = bitcnt2 & bitcnt1 & bitcnt0;


%
        status data latch stores active mode data in the upper nibble of the status data byte.
        The active parameter data in the lower nibble is latched into a 7-segment LED display driver.
        The active mode data is output on mode_led[] to discrete LED’s.
%
        mode_latch[].clk = sysclk;
        mode_latch[].clrn = GLOBAL(sysresetn);
        mode_latch[].ena = sLatchByte & xf_B[].q == STATUS_CMD;
        mode_latch[].d = shift[7..4].q;

        mode_led[] = mode_latch[].q;
        display[] = local & shift[7..0].q # !local & display_bit[];
```

```
        display_len1 = sysresetn & !(local & sLatchByte & xf_B[].q == STATUS_CMD
                                        # !local & !display_len1_bit);
        display_len2 = sysresetn & !(local & sLatchByte & xf_B[].q == PARAM_CMD
                                        # !local & !display_len2_bit);



%
        eeprom data latch retains data from a eeprom read cycle or eeprom status register
        read cycle. the data is retrieved by the DSP during a EEPROM_RET_CMD cycle.
%
        eeprom[].clk = sysclk;
        eeprom[].clrn = GLOBAL(sysresetn);
        eeprom[].ena = sLatchByte & ((xf_B[].q == EEPROM_R_CMD) # (xf_B[].q == EEPROM_SR_CMD));
        eeprom[].d = shift[].q;



%
        oscillator divider generates sysclk = osc/8 (1.5 Mhz) and osc/2 (halfosc), the
        6.144 Mhz fck reference for the CS8412.
%
        oscdiv[].clk = osc;
        oscdiv[].clrn = GLOBAL(sysresetn);
        oscdiv[].d = oscdiv[].q + 1;
        sysclk = oscdiv2.q;
        halfosc = oscdiv0.q;



%
        128 Fs MCLK generator. Generates a 128 Fs clock from a 384 Fs MCLK
        by multiplying by 2, dividing by 3, then dividing by 2.
%
        mclkmult = EXP(!mclk) $ mclk;
        mclkdiv[].clk = mclkmult;
        mclkdiv[].clrn = GLOBAL(sysresetn);
        mclkdiv0.d = !mclkdiv0.q & !mclkdiv1.q;
        mclkdiv1.d = mclkdiv0.q & !mclkdiv1.q;
        mclkdiv2.d = mclkdiv1.q & !mclkdiv2.q # !mclkdiv1.q & mclkdiv2.q;
        thirdmclk = mclkdiv2.q;

%
        xf[] pin buffers. Used to sync and store xf inputs from the CS4912.
%

        xf_B[].clk = sysclk;
        xf_B[].clrn = GLOBAL(sysresetn);
        xf_B[].ena= !(!sLoadDSP & !sGoodBoot & !sWait);
        xf_B[].d = xf[];
```

```
% **************** Parallel Port Logic *********************** %


%
       Parallel port data latch A

       Bit        Name            Function
       0          SPIclk_bit SPI clock
       1          mosi_out_bitMOSI data out
       2          miso_out_bitMISO data out
       3          mosi_ena_bitMOSI data out enable
                       = 0 disabled
                       = 1 enabled
       4          miso_ena_bitMISO data out enable
                       = 0 disabled
                       = 1 enabled
       5          EE_CSn_bit serial EEPROM chip select
       6          4222_CSn_bitCS4222 chip select
       7          4912_CSn_bitCS4912 chip select
                       must be low during reset for SPI mode

       default = 0110 0000


%
       port_latch_a[].clk = nStrobe;
       port_latch_a7.clrn = GLOBAL(sysresetn);
       port_latch_a[6..5].prn = GLOBAL(sysresetn);
       port_latch_a[4..0].clrn = GLOBAL(sysresetn);
       port_latch_a[].ena = addr[] == ADDR_A;
       port_latch_a[].d = d[];

       SPIclk_bit = port_latch_a0.q;
       mosi_out_bit= port_latch_a1.q;
       miso_out_bit= port_latch_a2.q;
       mosi_ena_bit= port_latch_a3.q;
       miso_ena_bit= port_latch_a4.q;
       EE_CSn_bit = port_latch_a5.q;
       4222_CSn_bit = port_latch_a6.q;
       4912_CSn_bit = port_latch_a7.q;

%
       Parallel port data latch B

       Bit        Name                  Function
       0-7        display_bit[7..0]7 segment display data
```

```
        default = 0000 0000
%
        port_latch_b[].clk = nStrobe;
        port_latch_b[].clrn = GLOBAL(sysresetn);
        port_latch_b[].ena = addr[] == ADDR_B;
        port_latch_b[].d = d[7..0];

        display_bit[]= port_latch_b[7..0].q;



%
        parallel port data latch C.

        Bit         Name                   Function
        0           pio_out_bit      CS4912 PIO data output
        1           pio_dir_bit      pio direction
                                           = 0 in
                                           = 1 out
        2           dbclk_bit        debug port clock
        3           dbda_out_bit     debug port data
        4           dbda_dir_bit     debug port direction bit
                                           = 0 in
                                           = 1 out
        5           display_len1_bit active parameter latch enable
        6           display_len2_bit parameter data latch enable
        7           -                reserved

        default = X110 0010
%

        port_latch_c[].clk = GLOBAL(nStrobe);
        port_latch_c[6..5].prn = GLOBAL(sysresetn);
        port_latch_c[4..2].clrn = GLOBAL(sysresetn);
        port_latch_c1.prn = GLOBAL(sysresetn);
        port_latch_c0.clrn = GLOBAL(sysresetn);
        port_latch_c[].ena = addr[] == ADDR_C;
        port_latch_c[].d = d[6..0];

        pio_out_bit = port_latch_c0.q;
        pio_dir_bit = port_latch_c1.q;
        dbclk_bit  = port_latch_c2.q;
        dbda_out_bit = port_latch_c3.q;
        dbda_dir_bit = port_latch_c4.q;
        display_len1_bit= port_latch_c5.q;
        display_len2_bit= port_latch_c6.q;
```

```
%
        Parallel port latch D. Note that the 4912boot bit is not cleared on sysresetn, but
        will be low at power up.

        Bit         Name             Description
        0           DAP_CSn          external SPI port chip select, active low
        1           4912boot_bitCS4912 boot pin.
                                                = 0 boot ROM disabled
                                                = 1 boot ROM enabled
        2-7                                Reserved

        default = XXXX XXX0
%
        port_latch_d[].clk = GLOBAL(nStrobe);
        port_latch_d[].ena = addr[] == ADDR_D;
        port_latch_d0.clrn = GLOBAL(sysresetn);
        port_latch_d[].d = d[1..0];

        DAP_CSn_bit= port_latch_d0.q;
        4912boot_bit= port_latch_d1.q;


%
        status output mux.
%
        status[] = addr[] == ADDR_A & (VCC, nReq, mosi, miso)
                        # addr[] == ADDR_B & (freq, s_mode, master, local)
                        # addr[] == ADDR_C & (VCC, VCC, pio, dbda)
                        # addr[] == ADDR_D & xf_B[].q;


END;
```

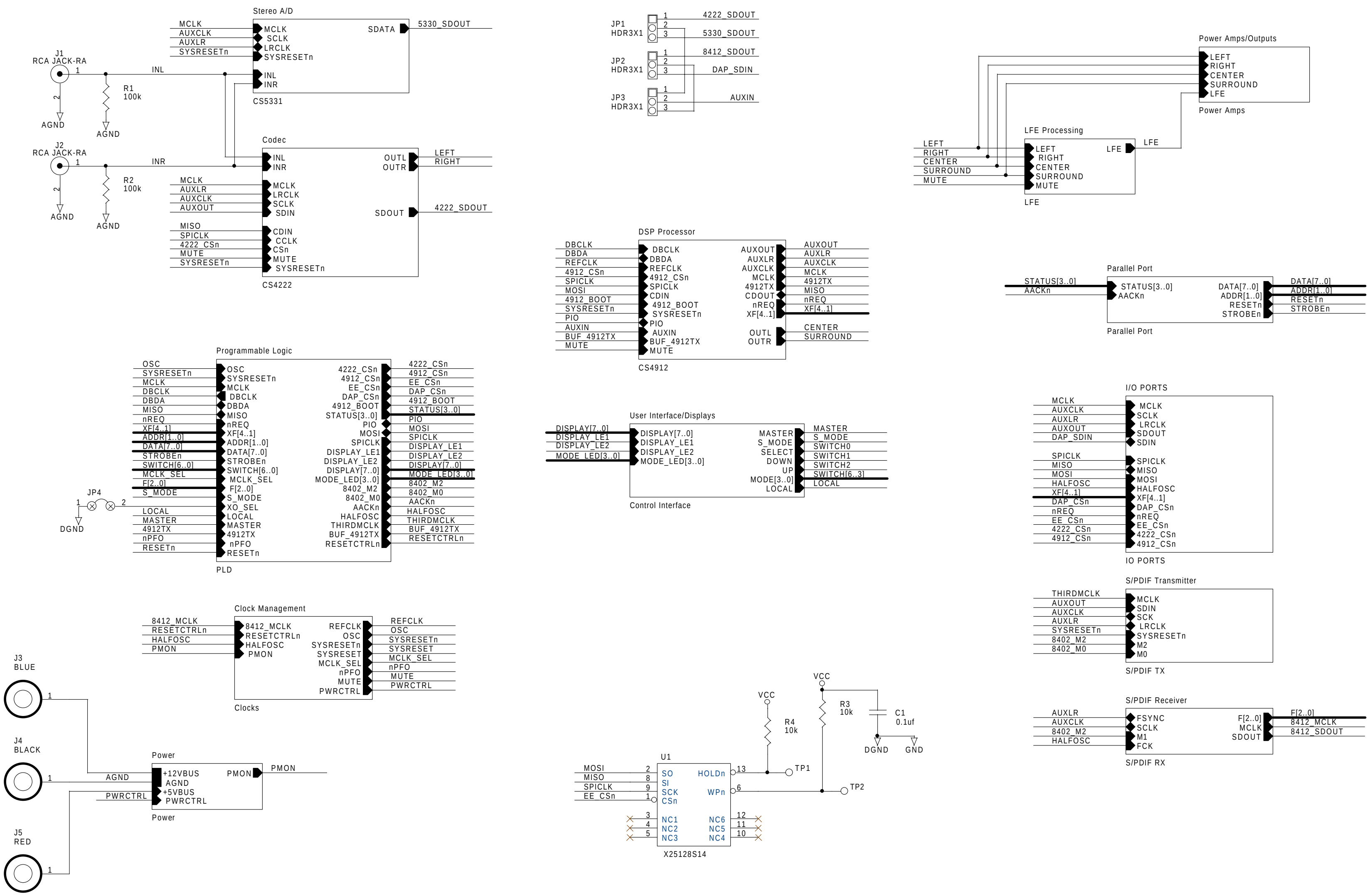


Figure 4. CRD4912 Schematic, Top View

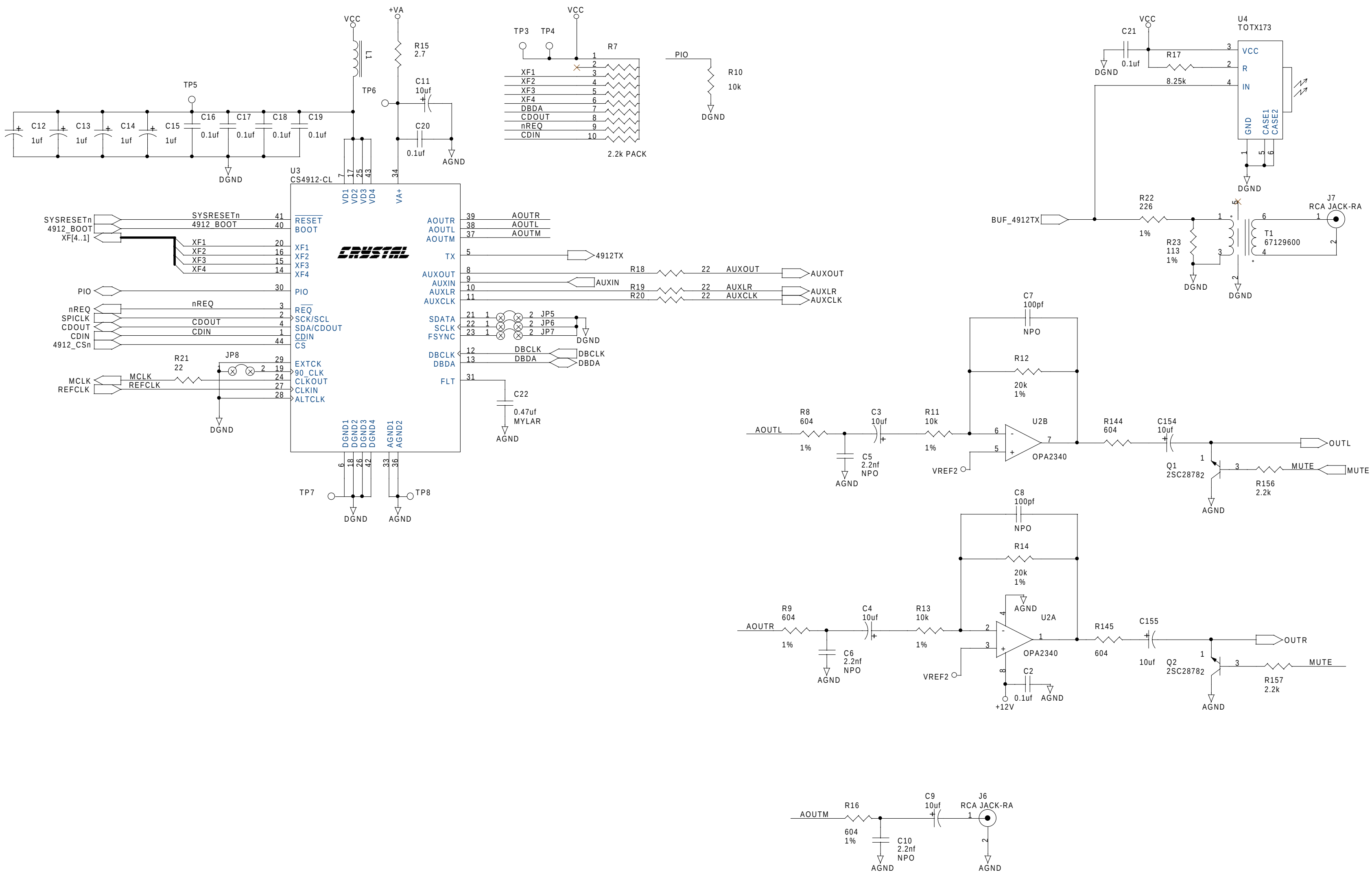


Figure 5. CS4912

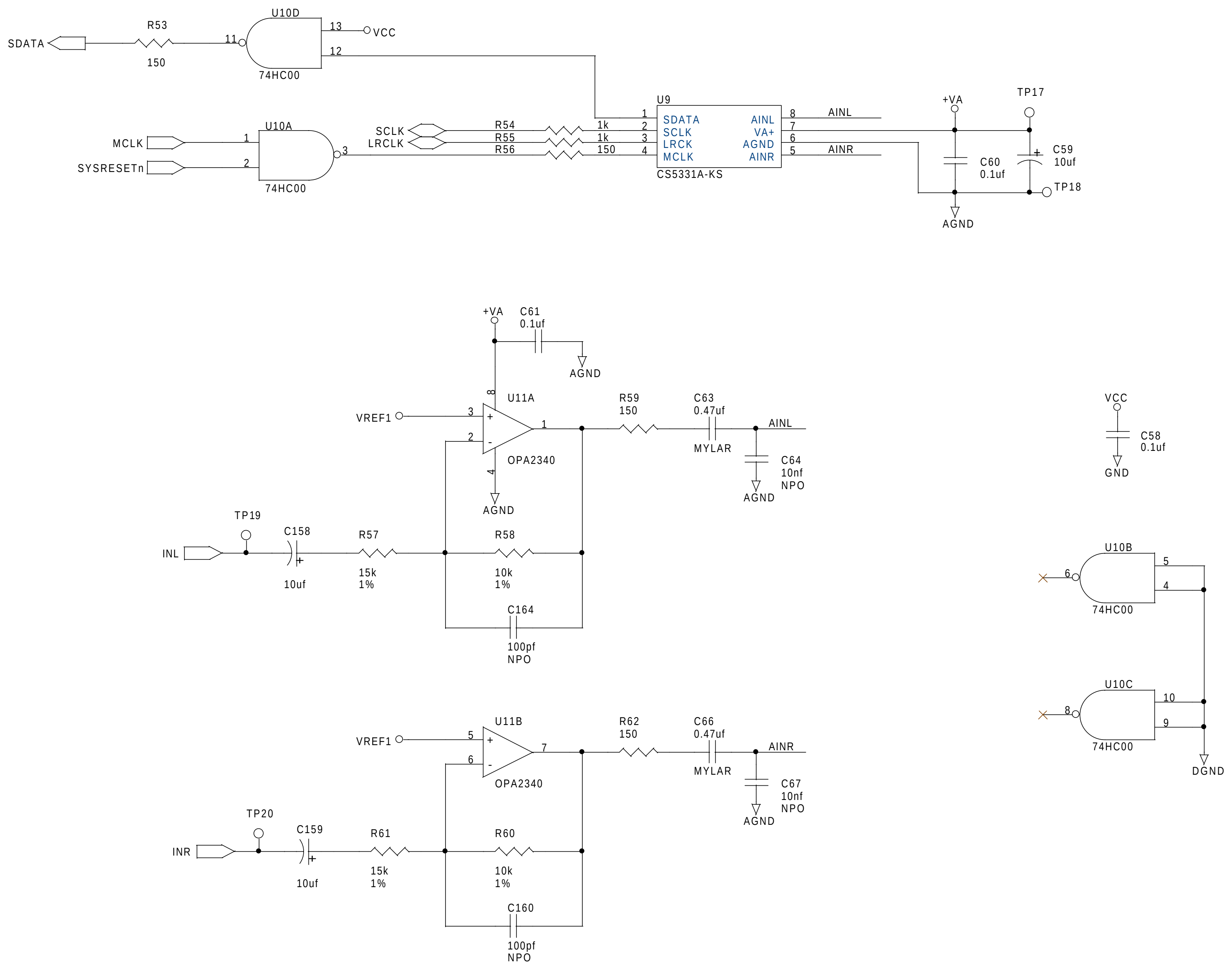


Figure 6. CS5330A/31A

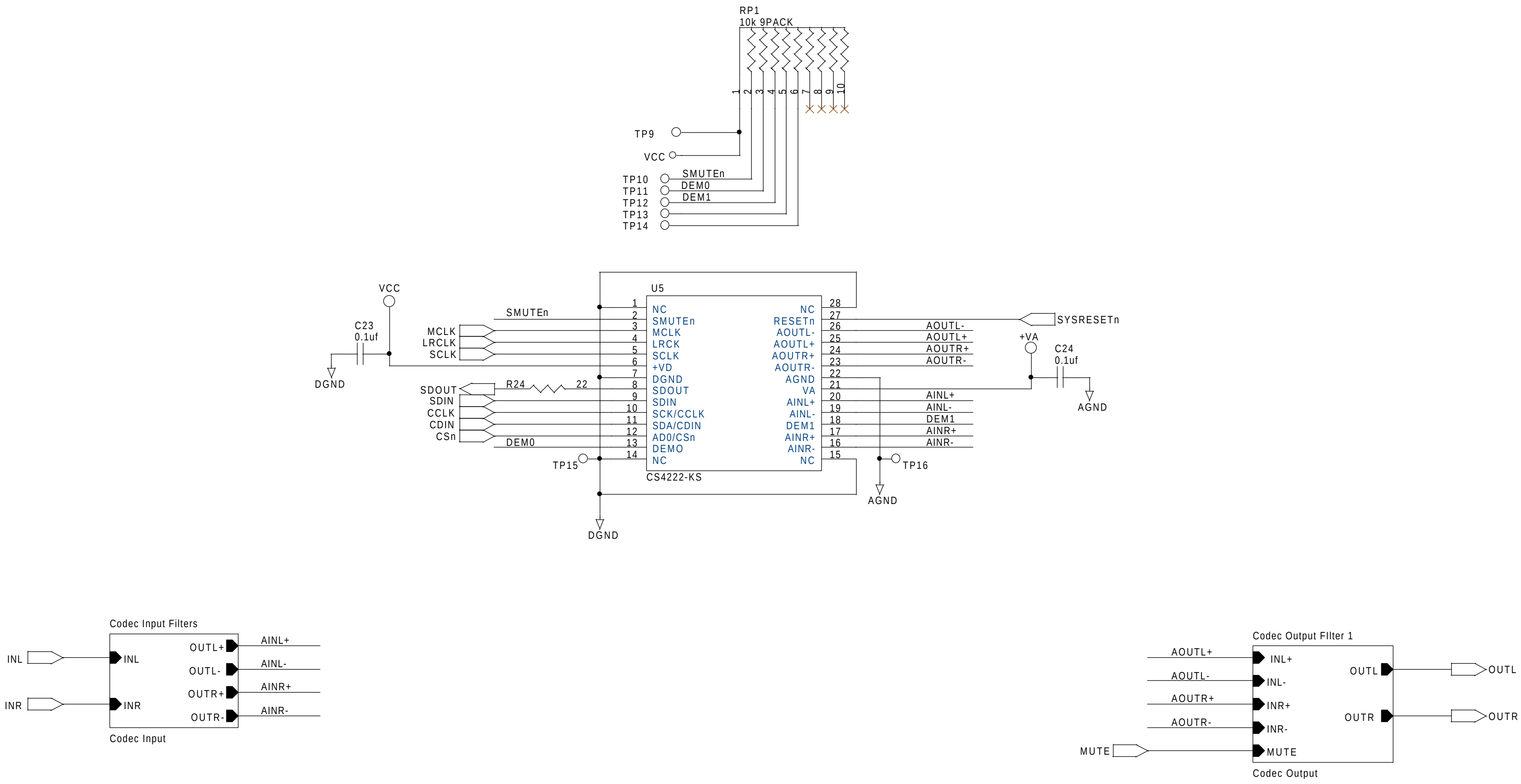


Figure 7. CS4222

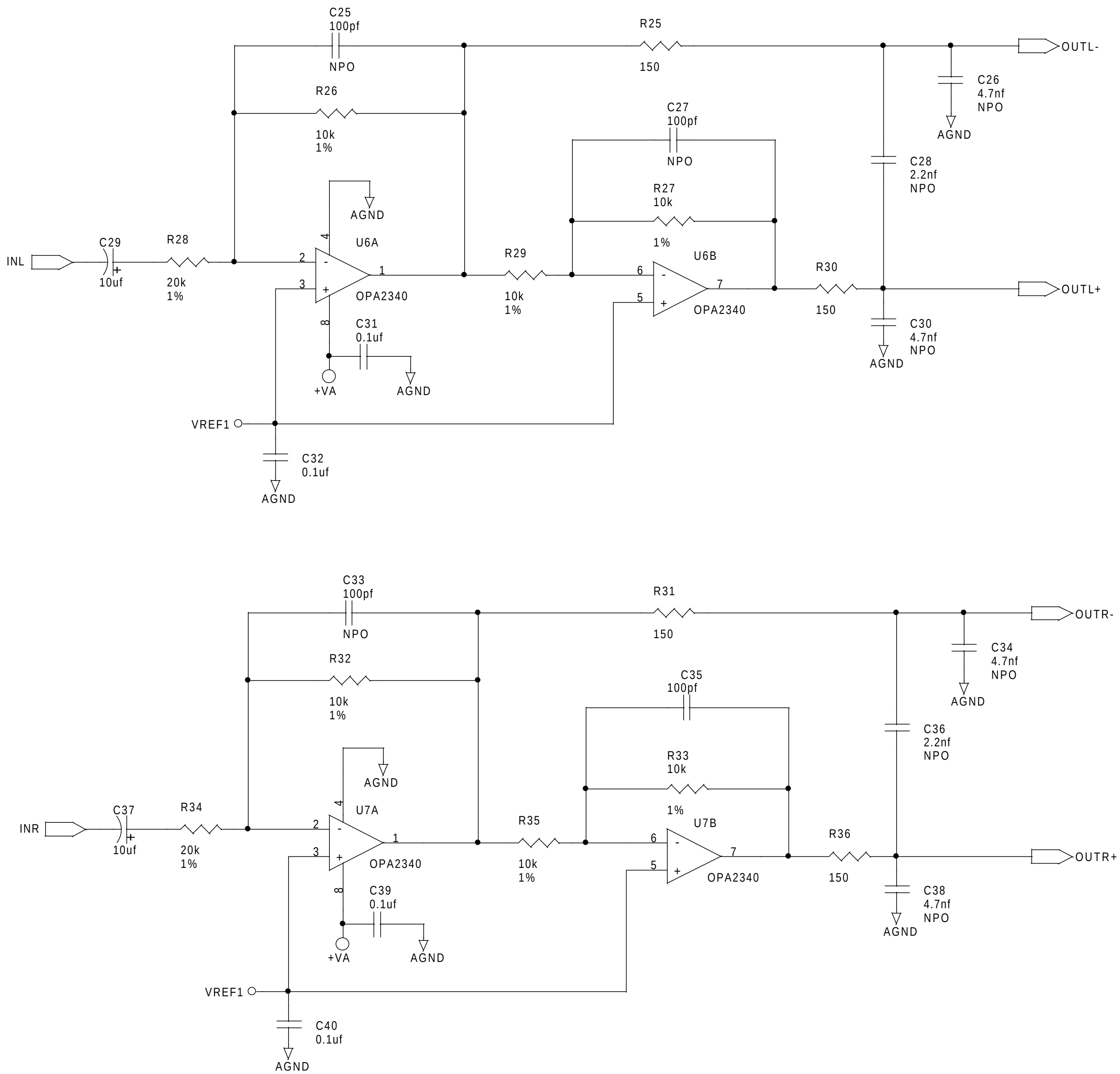


Figure 8. CS4222 Input

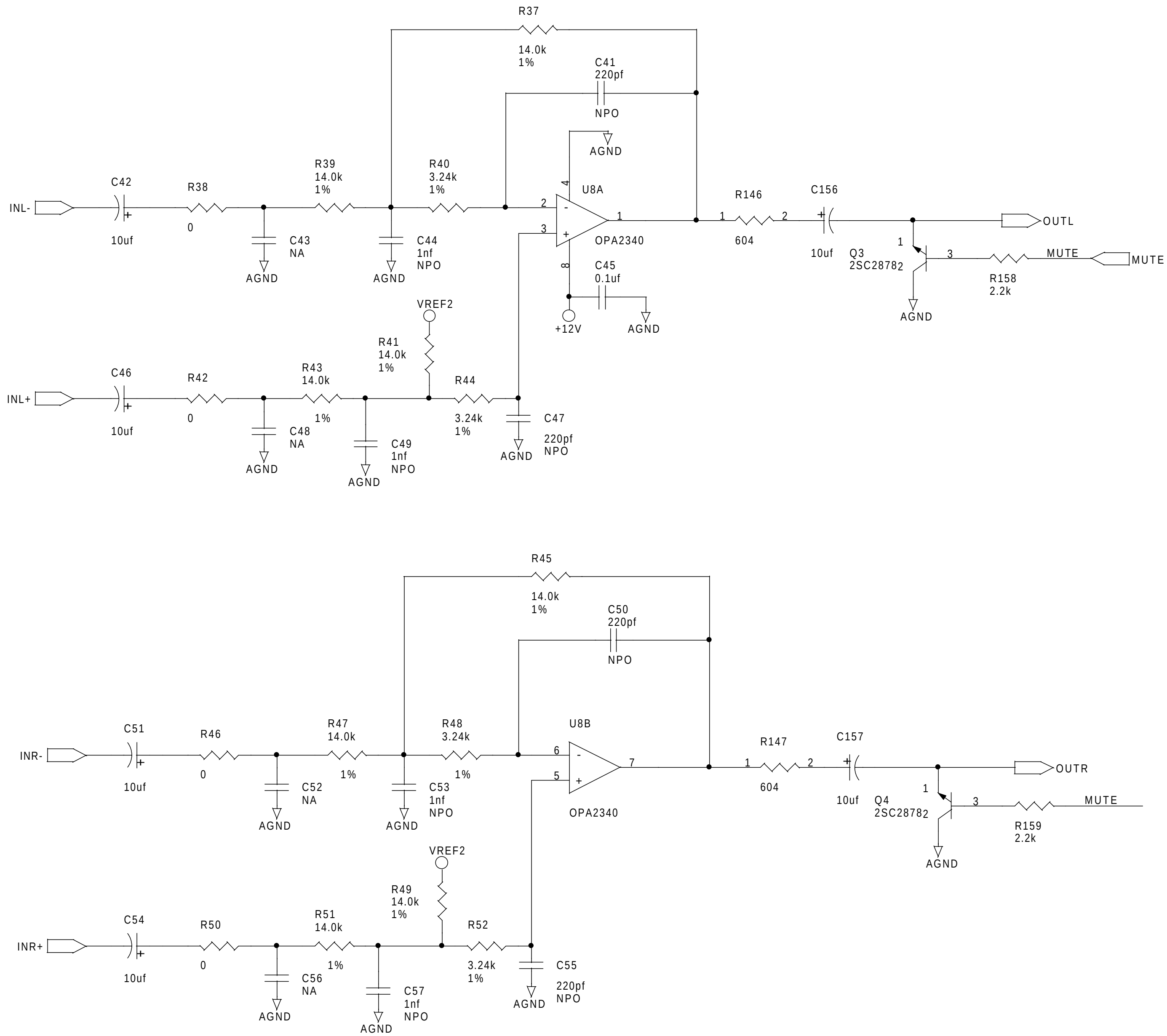


Figure 9. CS4222 Output

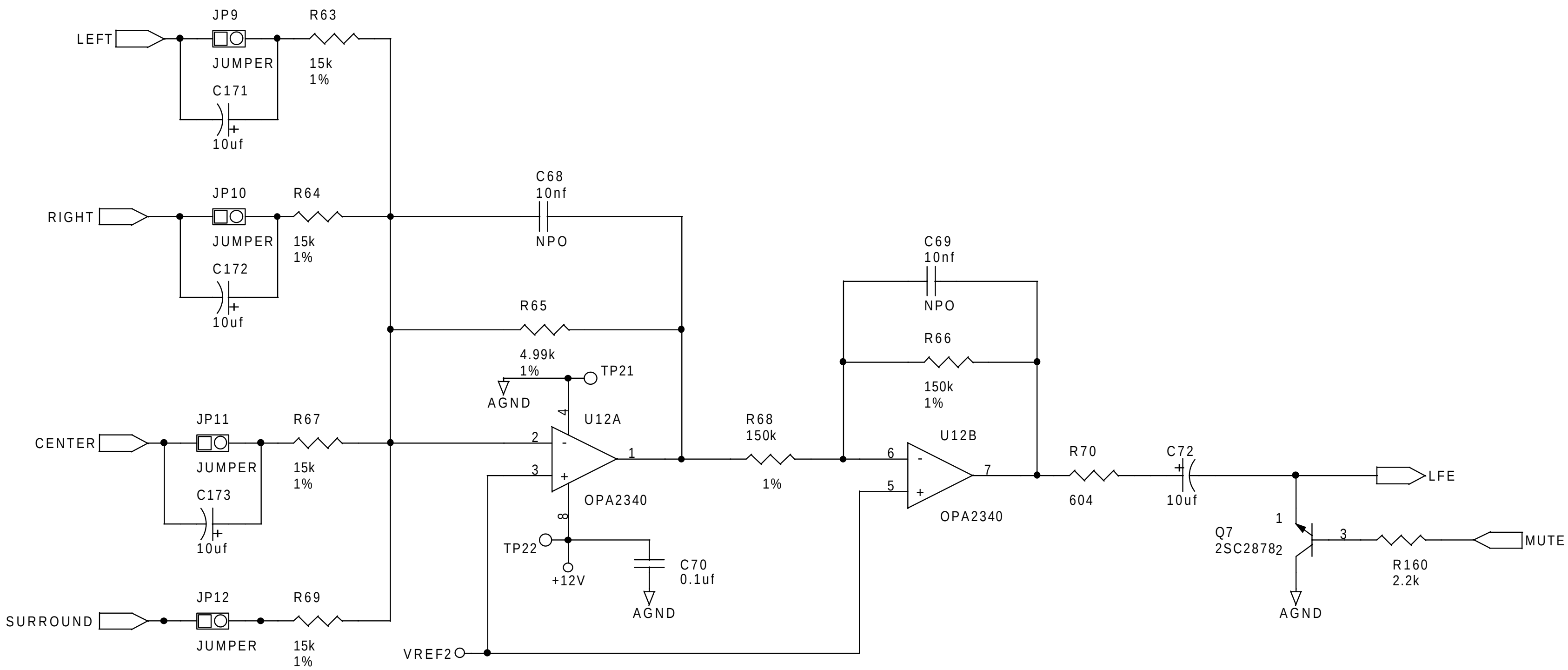


Figure 10. LFE Mixer

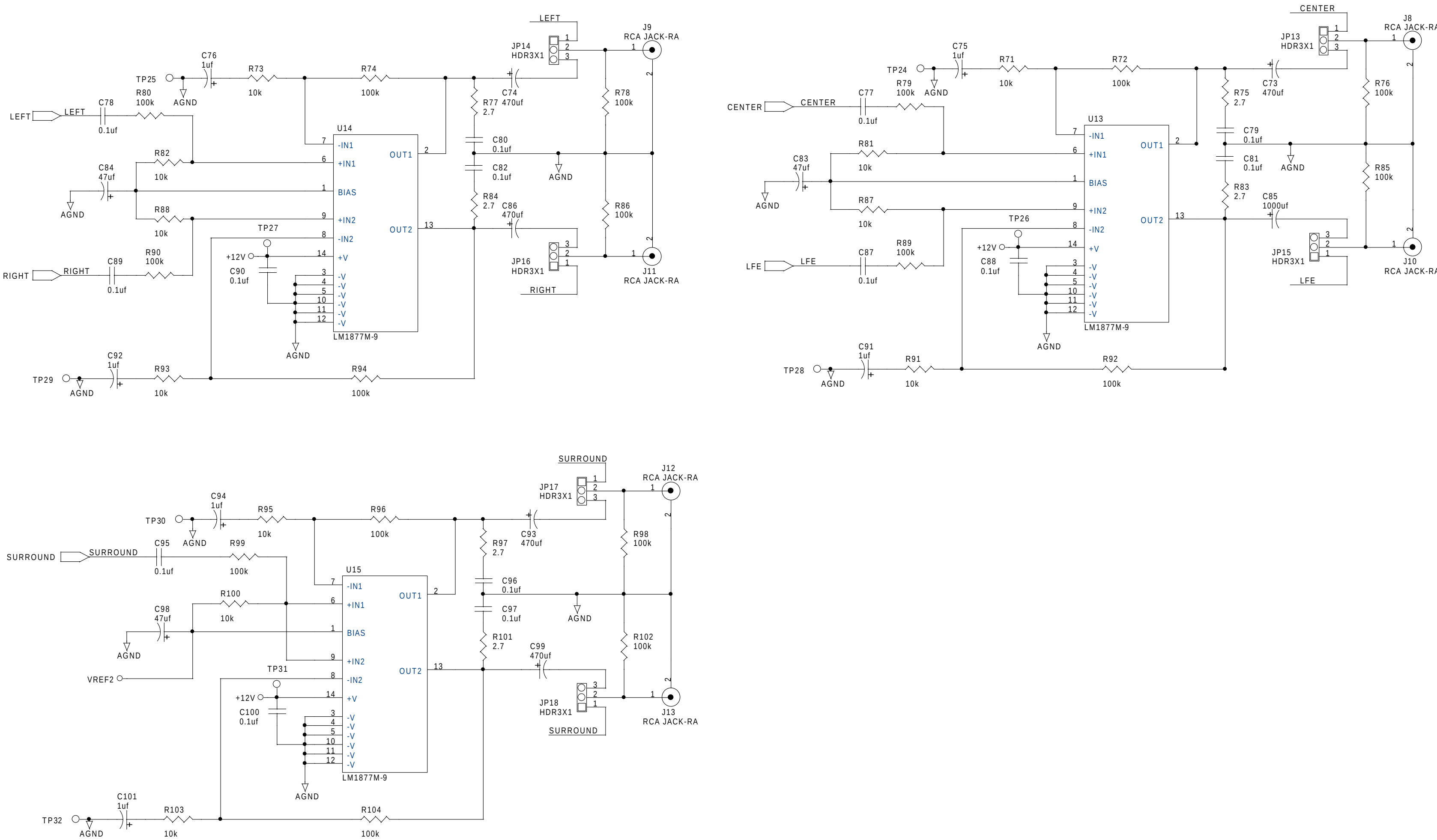


Figure 11. Power Amps

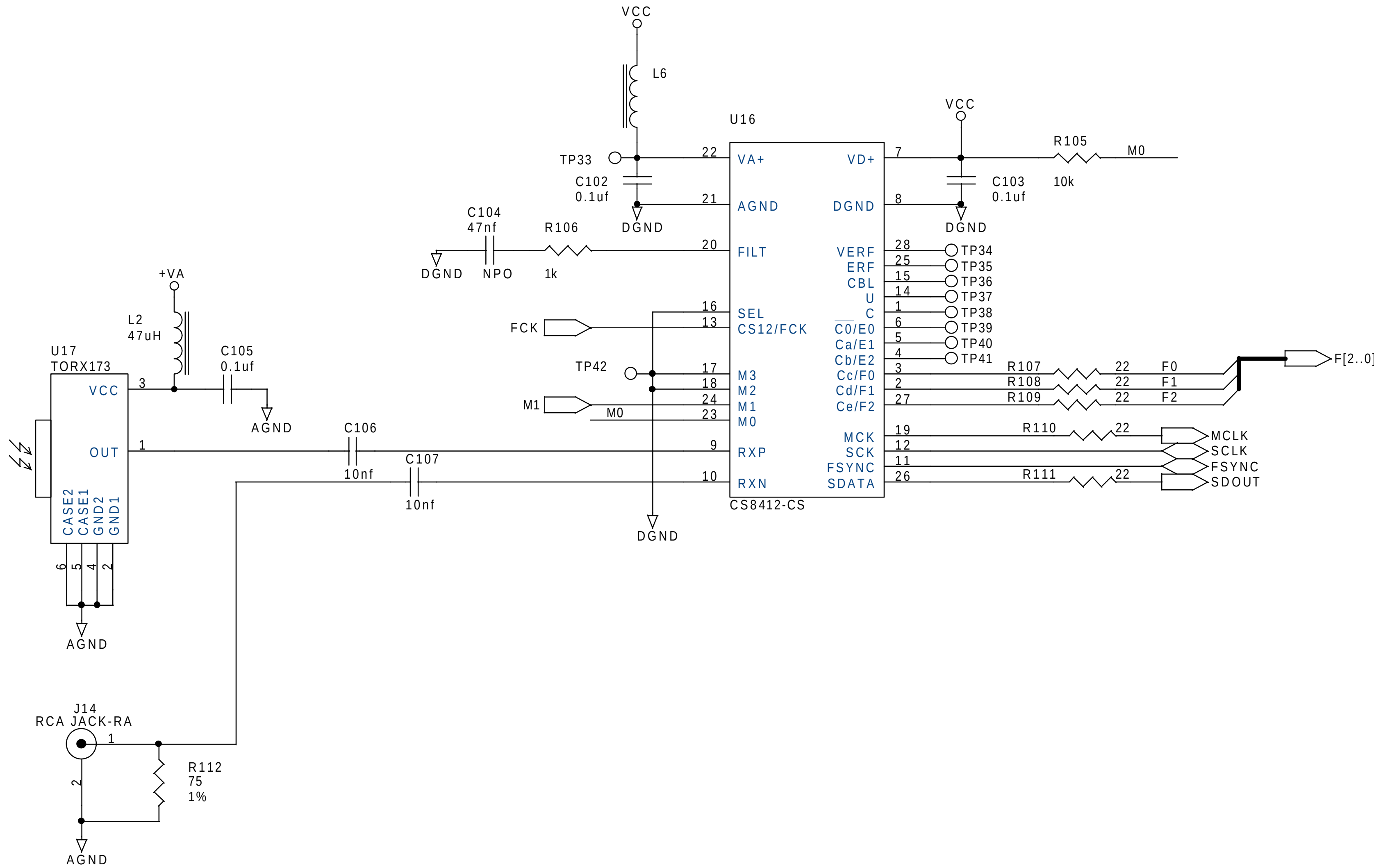


Figure 12. S/PDIF Receiver

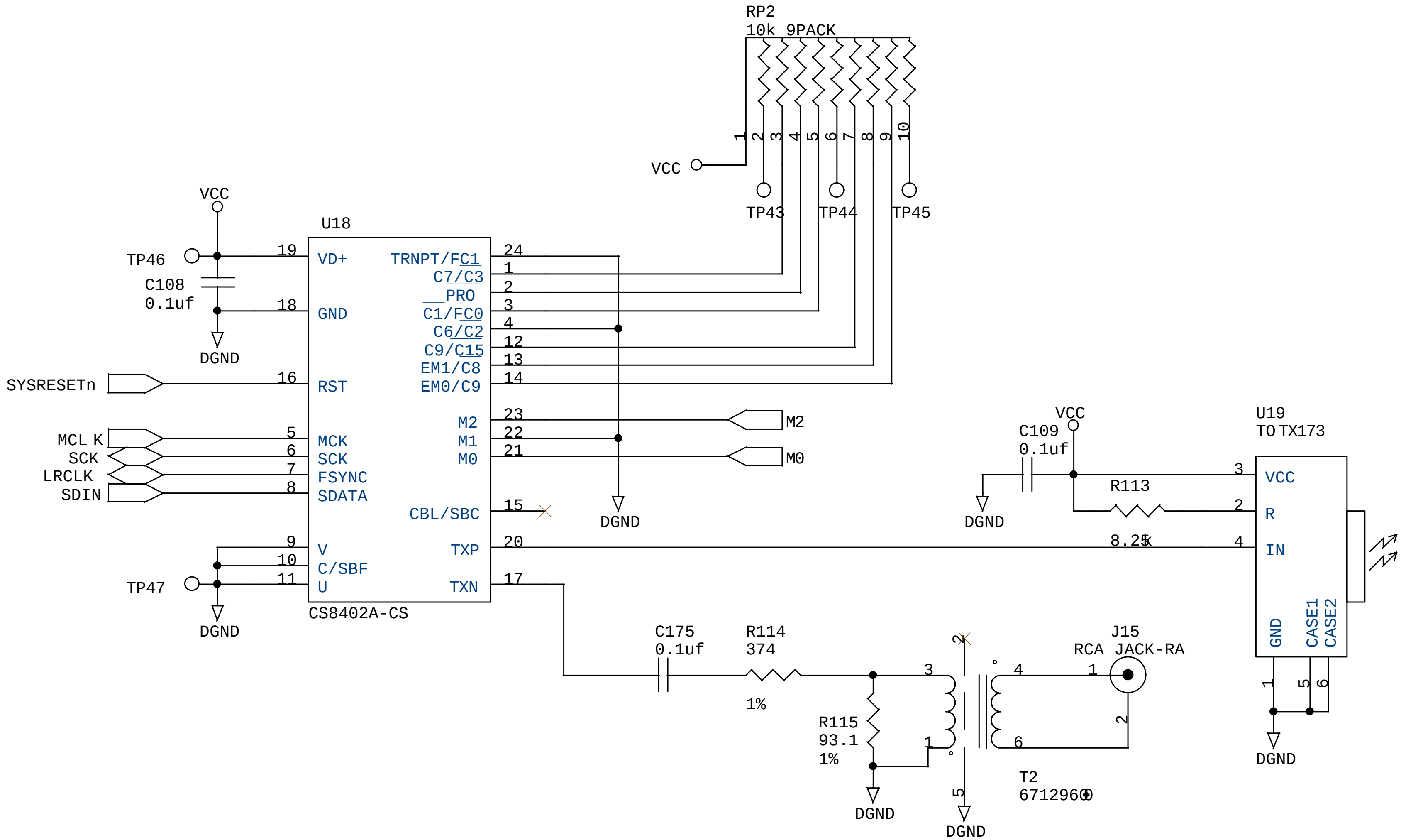


Figure 13. S/PDIF Transmitter

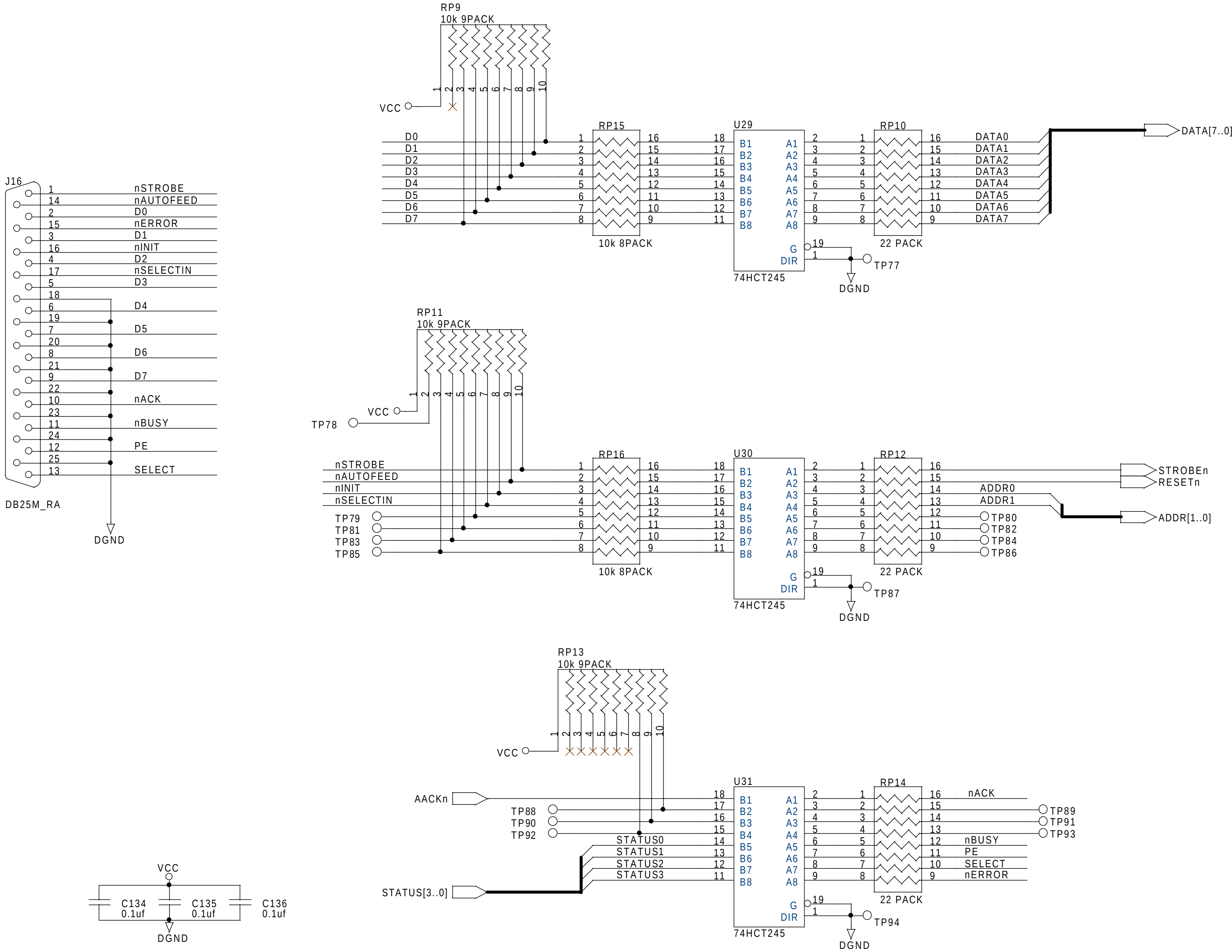


Figure 14. Parallel Port

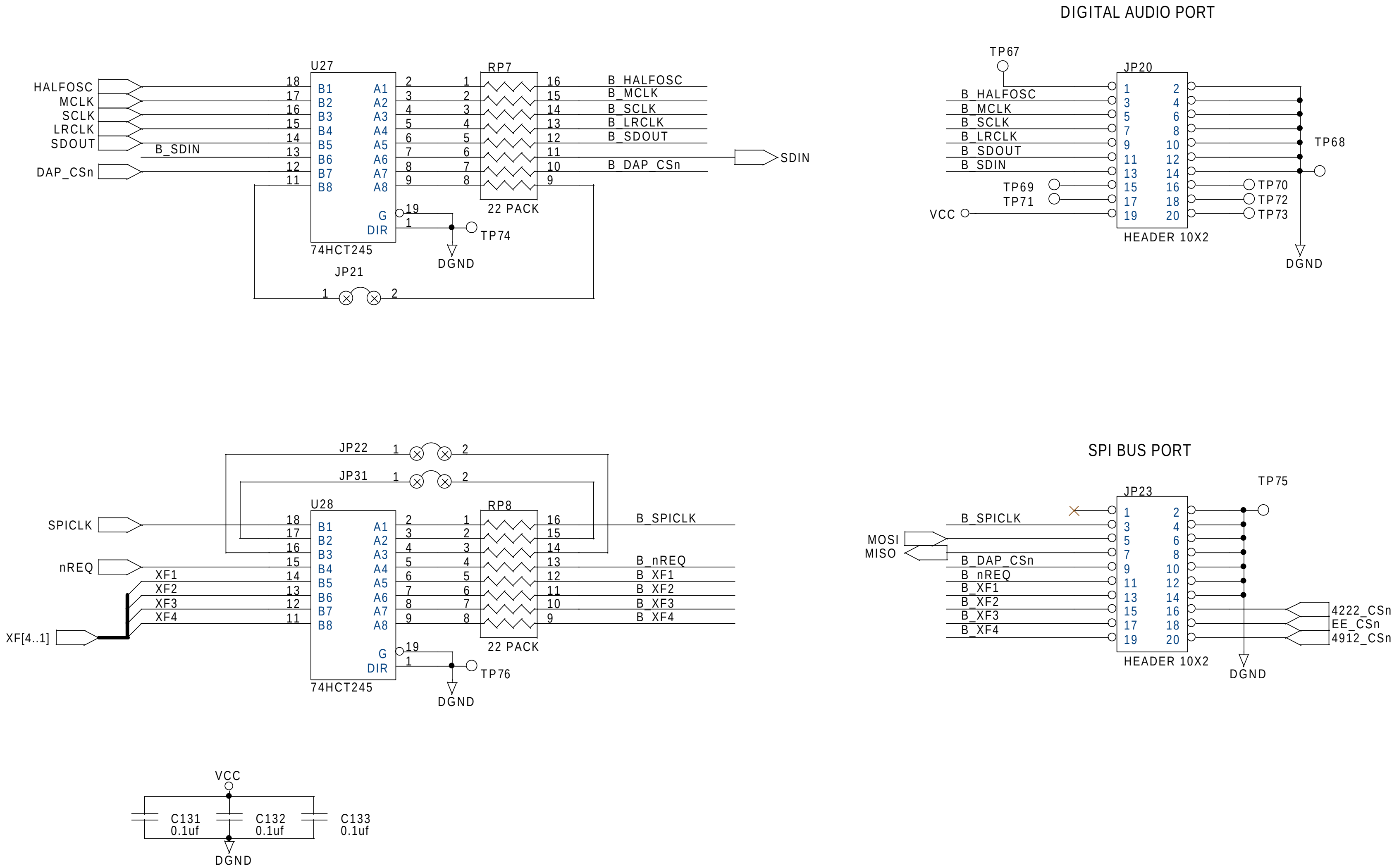


Figure 15. DAP and SPI Ports

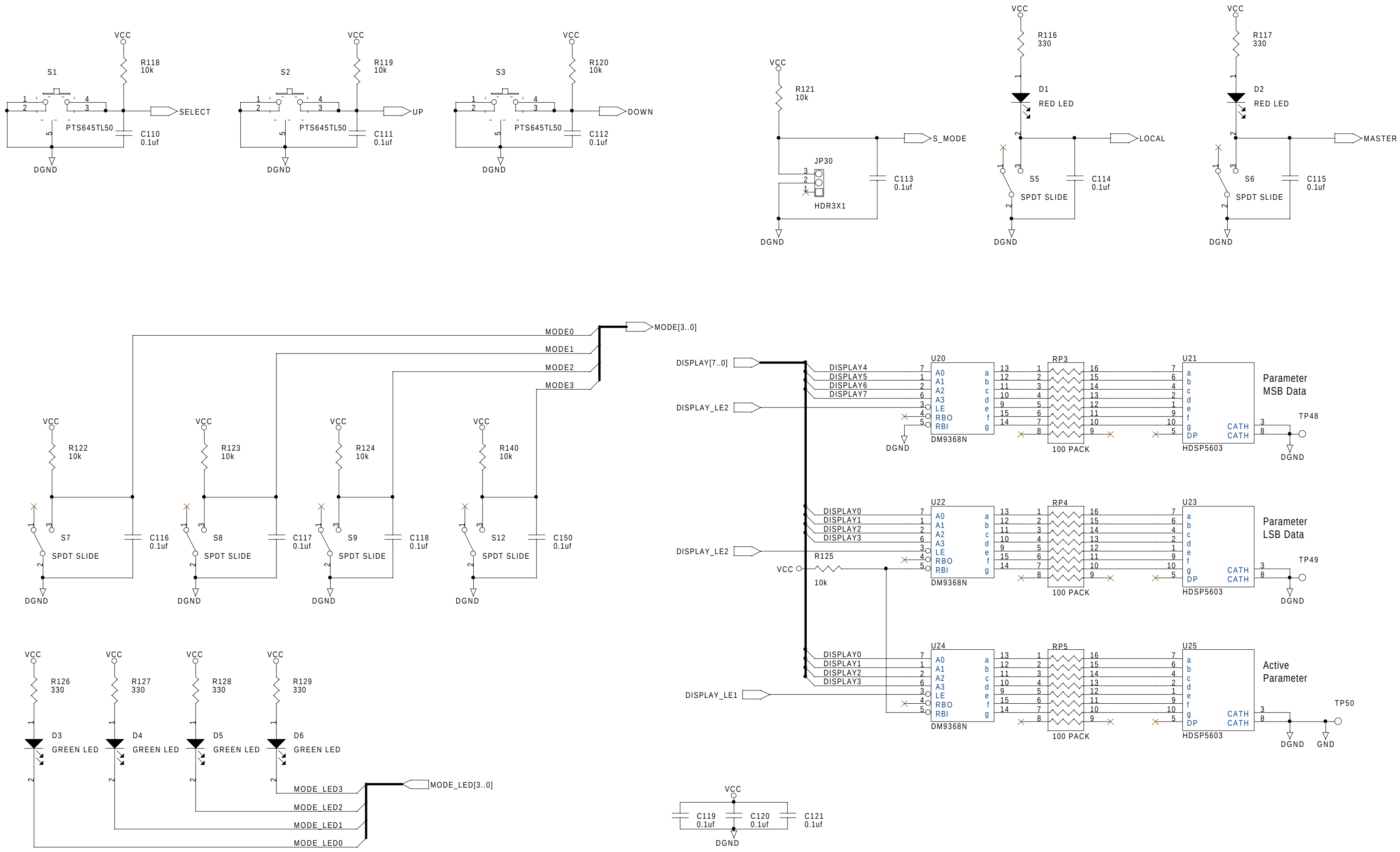


Figure 16. Switches and Indicators

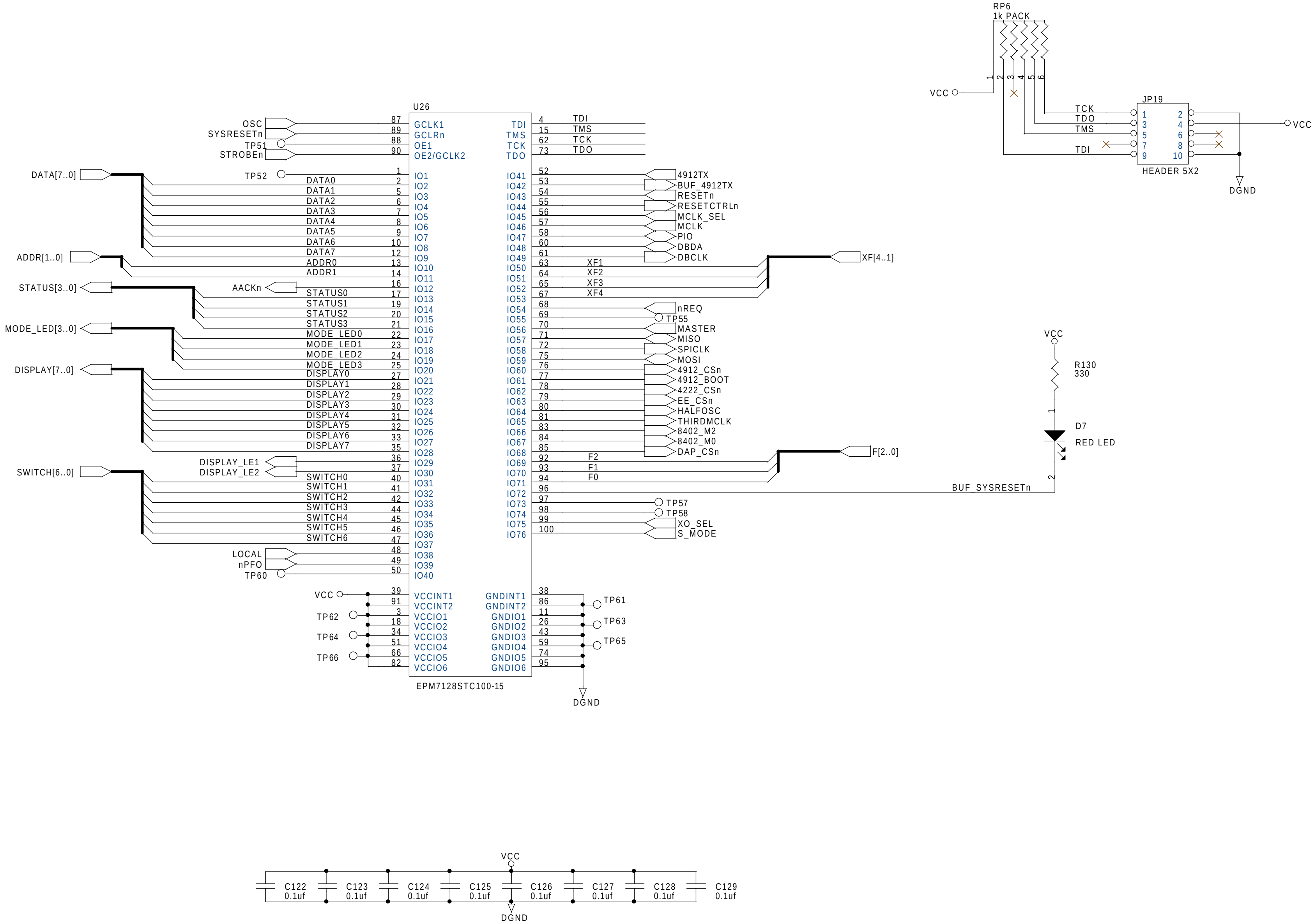


Figure 17. Programmable Logic

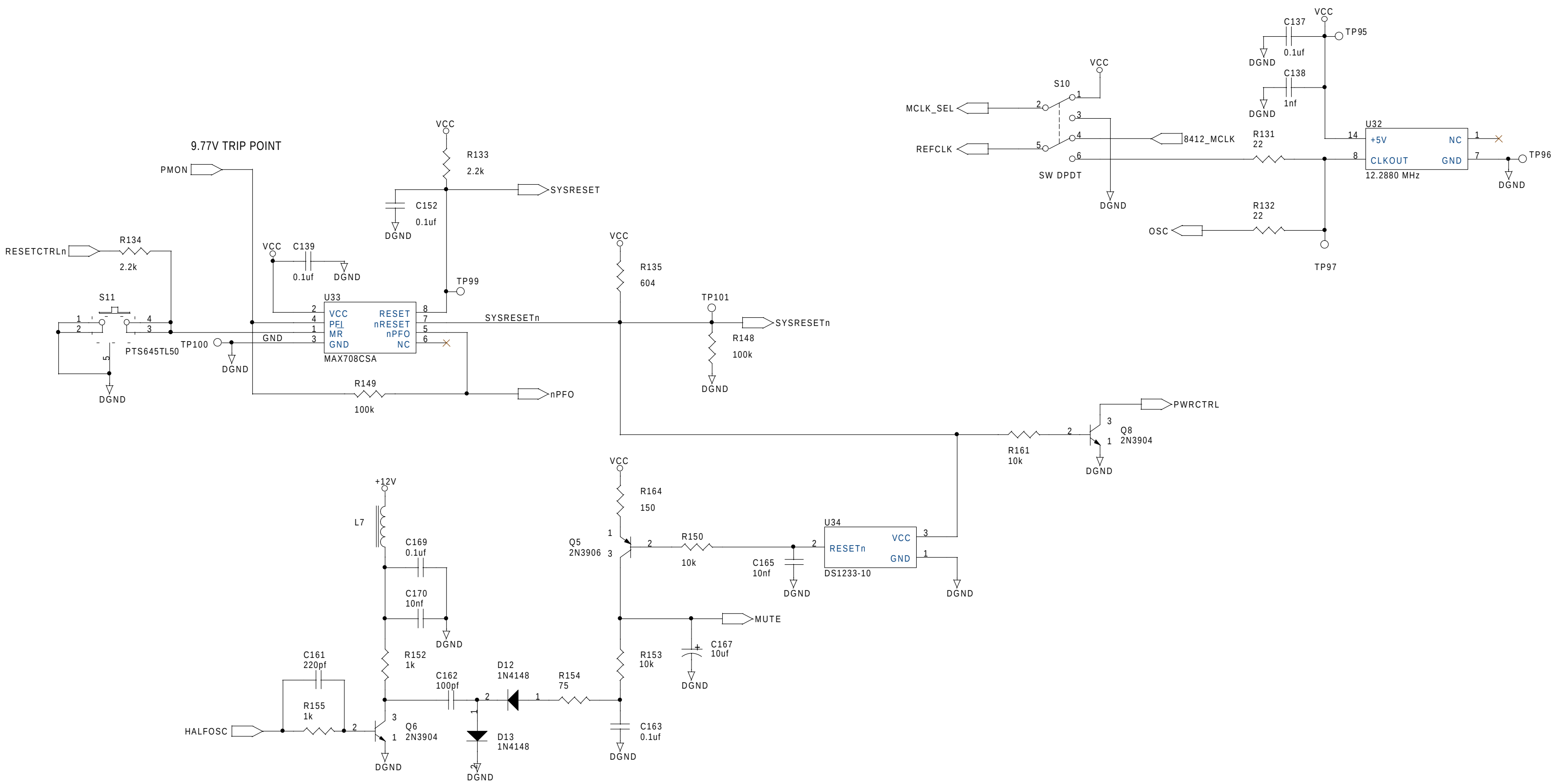
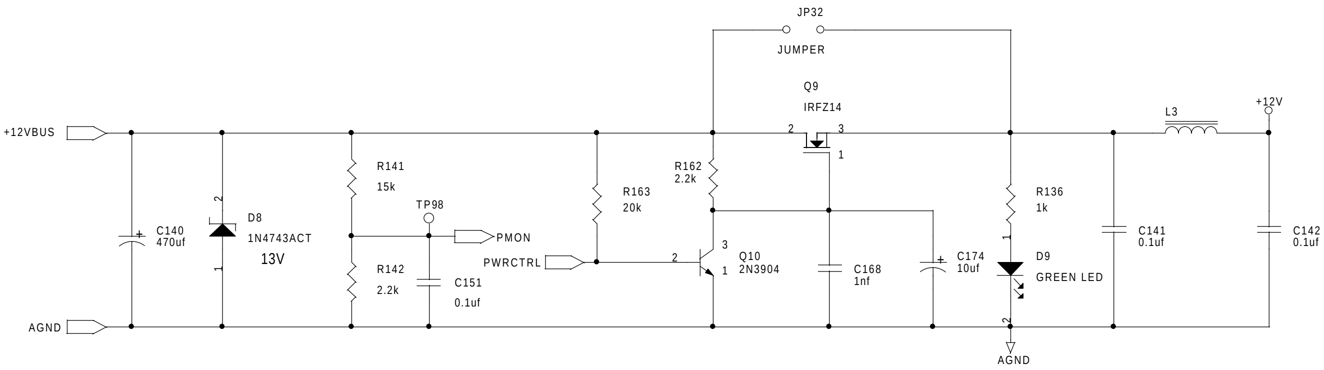
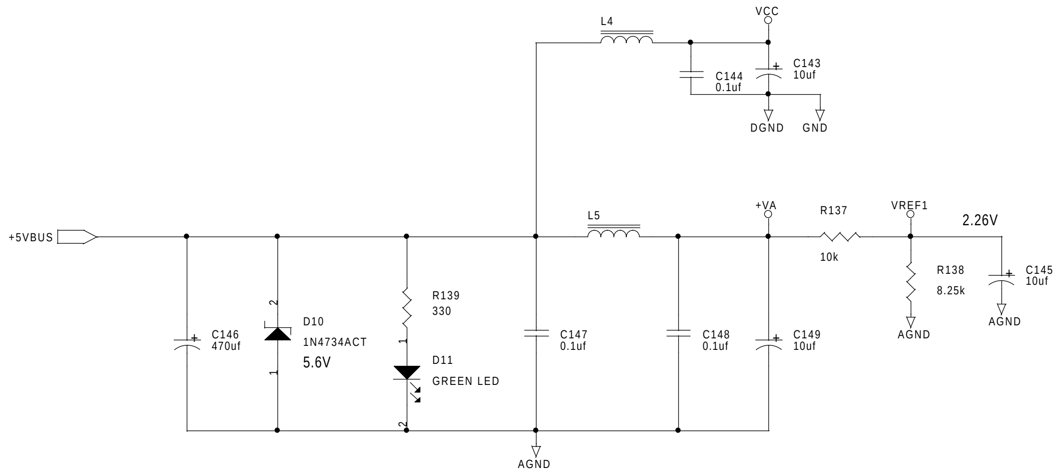
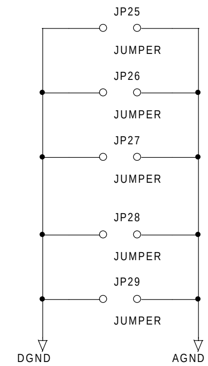


Figure 18. Reset and Oscillator



DEVICE	VCC	DGND
U10	14	7
U27	20	10
U28	20	10
U29	20	10
U30	20	10
U31	20	10





NOTES:

1. NO POP Q10. PLACE C174 ACROSS THE EMITTER AND COLLECTOR OF Q10.

Figure 19. Power

SMART Analog™