

CobraNet™

EV-2 Development System Manual

Introduction.....	2
Getting Started	4
<i>Required Materials</i>	<i>4</i>
Included:	4
Not Supplied:	4
<i>Setup Procedure</i>	<i>5</i>
<i>Switch and Connector Functionality</i>	<i>6</i>
J300	6
J401	6
J700	6
P450	6
P501	7
P504	7
SW200	7
SW201-SW204	7
SW500	7
Detailed Description of EV-2 Components.....	9
<i>The Microcontroller</i>	<i>9</i>
Microcontroller Memory Space:	9
Microcontroller Port Connections:	10
Interfacing the Microcontroller to the CM	11
Programming the Microcontroller	12
<i>Interfacing Serial Audio to the CM</i>	<i>13</i>
<i>FPGA</i>	<i>13</i>
Configuring the FPGA	15
Functional Discussion of FPGA Operation	15
<i>Hex Switches</i>	<i>18</i>
<i>EV-2 Schematics, Page-by-Page Description</i>	<i>19</i>
Block Diagram	19
Microcontroller and Hex Switches	19
A/D Converter	19
D/A Converter	19
Connectors and Interfaces	19
Optional VCXO	19
AES/EBU Transceiver	20
Power Supply Conditioning	20
FPGA	20
Appendix A: Definition of Terms	21
Appendix B: EV-2 Specifications	24
Appendix C: Other Resources.....	26
Appendix D: EV-2 Schematic Drawings.....	27
Appendix E: EV-2 Command Line Interface.....	36
Appendix F: CD-ROM Contents.....	43

Introduction

The EV-2 provides a means of evaluating the CM-1 or CM-2 CobraNet™ Modules and the Cirrus Logic CS181xx family of CobraNet chips. In addition to evaluating the CM-1 or CM-2 (hereafter collectively referred to as the CM except where differences between the CM-1 and CM-2 exist), the user may also use the EV-2 as a development platform and as an example interface for CMs, the CS181xx, and other CobraNet related projects. The EV-2 connects to the CM via the module's host interface. An 8051 type microcontroller interfaces to the CM's host port, and a simple audio router on the EV-2 allows multiple audio inputs and outputs to connect to the CM's serial audio interface. The included software provides a simple interface for audio routing on the EV-2, as well as development support.

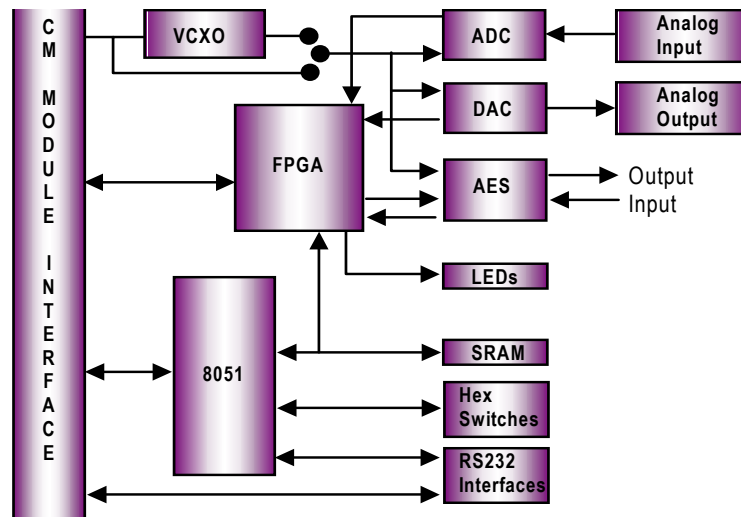


Figure 1. EV-2 Block Diagram

Features:

- Analog audio I/O: Two channels of analog audio input converted to high quality, 24-bit, 48 kHz digital audio. Two channels of 24-bit, 48 kHz digital audio converted to high quality, analog audio output. Refer to *Appendix B* for audio I/O specifications.
- Digital audio I/O: One stream of AES/EBU input and one stream of AES/EBU output. An AES/EBU stream is two channels of digital audio. The AES/EBU input stream is sample rate converted.
- 8051-type microcontroller: 64kB on-chip Flash Program Memory, 1kB internal SRAM, 32kB external SRAM and in-system programmability.
- Field programmability: The supplied EV-2 software provides a means to reprogram EV-2 microcontroller firmware for field upgrades or user development.
- RS232 Interfaces: Two RS232 interfaces, one direct to the CM and another to the microcontroller.
- Routing flexibility: Route from any audio source to any audio sink using the supplied EV-2 software. Route to and from the CM as well as within the EV-2.

- Sine wave generation: A sine wave test tone may be used as an alternate audio source. Minimal frequency and gain control is provided.
- Hex switches: Four hex formatted switches may be used for network identification of the Cobra-Net module and/or user development.
- Command line interface: The 8051, via its RS232 serial interface, can be used to configure the CM using a command line interface. Cobranet MI variables can be viewed and modified using this interface. Refer to Appendix E for a description of the Command line interface.
- LED display: Three LED indicators are provided and may be used for user development.
- Power supply: Uses standard computer ATX power supply (not included).

Required Materials

The CobraNet EV-2 Development Package ships with the following materials:

- Qty. (1)

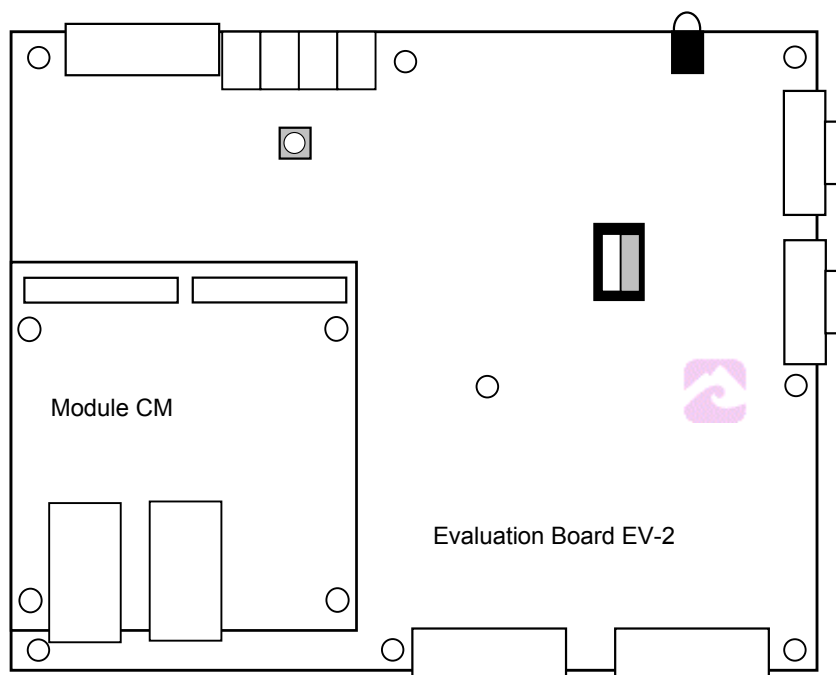


Figure 2. EV-2 Module w/CM PCB

- Two (2) ATX computer power supplies with cables are required, one for each EV-2 module. These devices are commonly available at computer retail stores.
- Audio cables.
- RS232 cables. (Not required to pass audio)

Setup Procedure

- Using the supplied Phoenix connectors, build audio input and output cables and two AES/EBU cables (if desired). These will be used to connect your audio input and output devices to the EV-2 modules. For analog audio pin assignments, see *Figure 3* or *Figure 5* below. For AES/EBU pin assignments, see *Figure 6* below.

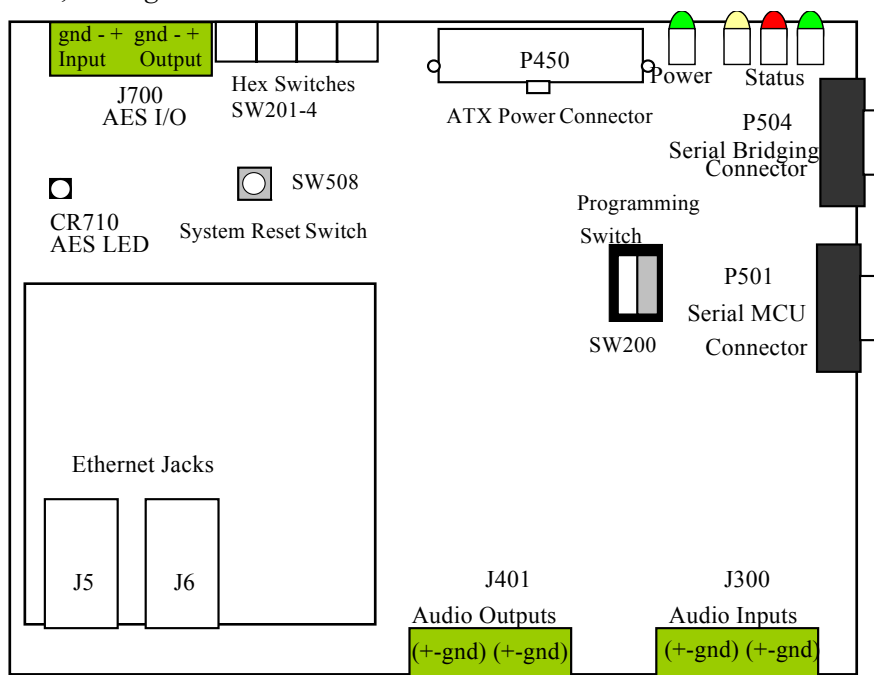


Figure 3. Connector, Switch and Jack Locations

- Connect a power supply to the ATX Power Connector at P450 on each EV-2 module.
- Connect the CAT5 crossover cable between the Ethernet jacks at J5 on each CM board.
- Connect a stereo audio source to the analog inputs at J300.
- Connect a stereo audio monitor to the analog outputs at J401.
- Apply power to both EV-2 modules.
- Verify that you have established a proper connection. See Table 1 on page 6 for Ethernet connector LED status .
- On the EV-2, the LED CR710, if on, indicates that the AES/EBU receiver does not detect a valid AES/EBU data input stream. If AES/EBU I/O is not being used, this can be disregarded. Otherwise, connect a proper AES/EBU signal to J700. Note that there must be a valid AES/EBU input for the AES/EBU output to work.

The units are now ready to pass audio. The audio input at J300 on one board should now appear at J401 on the other board and vice versa.

Condition	Module			
	CM-1		CM-2	
	Left LED	Right LED	Left LED	Right LED
Conductor	Flashing Green	Solid Orange	Flashing Orange	Flashing Green
Performer	Flashing Green	Solid Green	Solid Orange	Flashing Green
Fault	Flashing Red	Flashing Red	Flashing Orange	Flashing Orange

Table 1: Ethernet jack indicator legend.

Switch and Connector Functionality

J300

Audio Input Connector: Phoenix-style connector for two-channel balanced audio input, +14.4 dBu maximum (0 dBFS). Refer to *Figure 5* for the signal connection.

J401

Audio Output Connector: Phoenix-style connector for two-channel balanced audio output, +15.8 dBu maximum (0 dBFS). Refer to *Figure 5* for the signal connection.

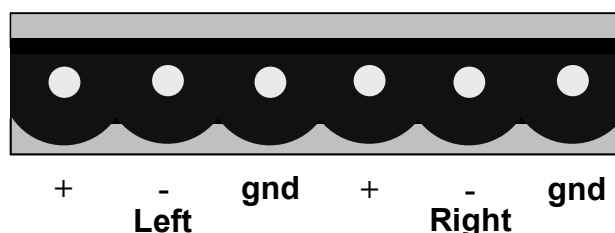


Figure 5. Analog Audio Input and Output Phoenix-style Connectors

J700

AES/EBU I/O Connector: Phoenix-style connector for an AES/EBU stream. Refer to *Figure 6* for the signal connection. For the AES/EBU tranceiver to operate properly a valid AES/EBU signal must be provided at the AES/EBU input.

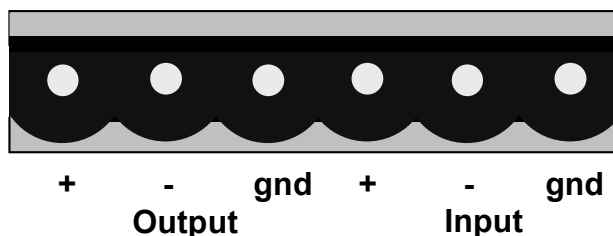


Figure 6. AES/EBU I/O Phoenix-style Connector

P450

ATX power supply connector: ATX power supply is not included with this kit.

P501

9-Pin, D-Type Connector: RS232 connection for communicating with the EV-2 microcontroller using the supplied routing software or a command line interface (see Appendix E). Data format is 19200, e, 8, 1.

P504

9-Pin, D-Type Connector: RS232 connection to the CM for serial bridging. The default data format is 19200 baud, 9-bit format for the CM-1. 9-bit format supports any 8 bit format with parity such as 19200, e, 8, 1. The default data format for the CM-2 is 19200 baud, 8-bit format.

SW200

Programming switch: The EV-2 microcontroller can be programmed via its serial port, connector P501. The supplied software can be used to perform field updates to the board's code and firmware. This programming capability is initially disabled, but can be enabled by setting the hex switches to FFF8H and then clicking on the "Hex Switches" display (see *Figure 8*). For more information about the programming mode, please refer to the *Programming the Microcontroller* section.

SW201-SW204

Hex switches: SW201-SW204 may be used to uniquely identify the unit on a network. Valid settings fall within the range 0000-FFEF (values FFF0-FFFF are reserved). Changing these values updates the value of the CobraNet module's SNMP variable, *sysName*, to the current hex switch value. Through SNMP, the user may query this variable. The SNMP response is of the form "PEAK_AUDIO_EVAL-SWwxyz", where the wxyz represents the hex values of the switches in ASCII format.

SW500

System reset switch: This momentary switch resets the EV-2 and attached CM, and initiates calibration operations for the analog-to-digital converter (ADC) and digital-to-analog converter (DAC).

Software

The EV-2 is supplied with the CNEval.exe application, which may be used to setup audio routes on the EV-2 (this should not be confused with routing audio over the CobraNet network). The EV-2 has eight sources of audio input, with each source consisting of a stereo pair of audio channels. The sources are:

- Four Synchronous Serial Interface (SSI) audio streams from the CM
- An AES/EBU audio input stream.
- Two audio streams from the ADC
- A sine wave generator

The ADC produces two sets of stereo audio streams, one high-quality and one low-latency. Please refer to the Cirrus Logic CS5396-KS documentation for more information on the ADC. The sine wave generator produces a stream of two identical 24-bit resolution sine waves. Using CNEval.exe, the user can route any of these eight source streams to any of the six output streams. The available output streams are the four SSI audio streams going to the CM, one going to the DAC and one to the AES/EBU transmitter.

CNEval.exe communicates with the EV-2 via an RS-232 serial connection. CNEval.exe can communicate using either COM1 or COM2 of the PC on which it is running. The connection from computer to EV-2 must be made as follows:

- Connect a straight-through, male-to-female, 9-pin RS232 cable to EV-2 connector P501.
- Select the appropriate PC serial port. The software will attempt to make contact with the EV-2.
- Once communication is established, the routing can then be configured. (See *Figure 7* below for an example of a routing scheme.)

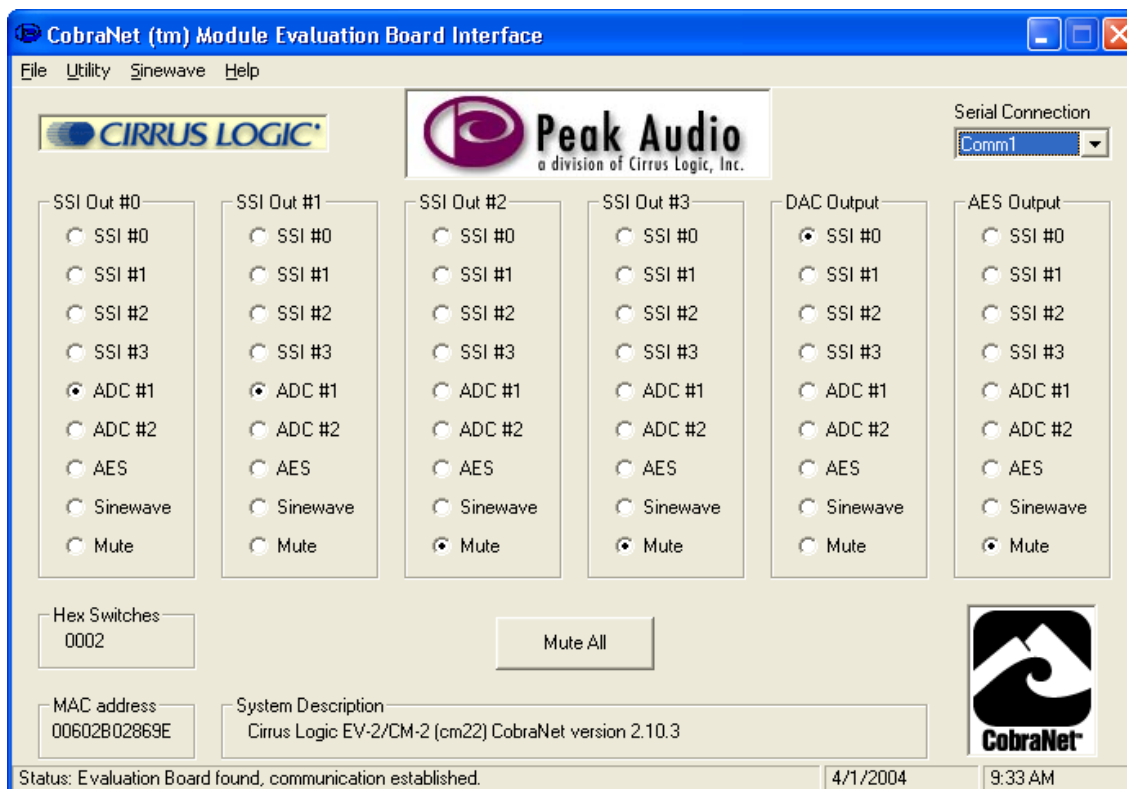


Figure 7. Screen Shot of EV-2 Software

The default on power up state of the EV-2 is for the ADC and DAC to be the source and sink respectively, using the CM's SSI #0 I/O stream. The audio is then transmitted/received via a CobraNet Bundle to/from the other CM. This allows evaluation of the CobraNet module in the analog domain without any configuration.

Clicking on any of the product or company logos will pull up the corresponding website using the computer's default browser (an Internet connection is required).

The EV-2 software also has a programming mode that may be used to perform field updates of the EV-2 microcontroller code. For more information about the programming mode, please refer to the section *Programming the Microcontroller* below.

Detailed Description of EV-2 Components

The Microcontroller

The microcontroller on the EV-2 is a Philips Semiconductor P89C51RD2. This microcontroller has 64 kByte of internal Flash Program Memory and 1 kByte of Static RAM. The microcontroller is field programmable using the provided CNEval.exe software. The microcontroller's clock rate is 33Mhz. Philips P89C51RD2 preliminary specification for programming information and part usage may be found on the Philips Semiconductor website: <http://www-us.semiconductors.philips.com/>.

Microcontroller Memory Space:

Besides the internal program and data memory space the microcontroller also has an external 64k data memory space. The microcontroller is hard-wired to execute from internal Flash Program Memory only. The Flash Program Memory has been segmented to store both Program and FPGA configuration data. The Program Memory map is shown in Table 2 on page 9 and the data memory map is shown in Table 3 on page 9:

Memory Location	Description
0x0000-0xBFFF	Program Memory
0xC000-0xFFFF	FPGA Configuration Data

Table 2: Flash Program Memory Map

Memory Location	Description
0x0000- 0x02FF	Internal Static RAM
0x0300-0x7FFF	External Static RAM
0x8000-0x87FF	Unused
0x8800	FPGA express mode configuration*
0x8801-0xFFFF	Unused

Table 3: Microcontroller Data Memory Map After Reset but Before FPGA Configuration

*After reset, the FPGA is the only device in the upper 32k of the data memory space. The microcontroller is then able to configure the FPGA and once configured the FPGA performs more sophisticated address decoding of the upper data memory space. Refer to the *FPGA* section of this document for a detailed description of the configuration process and a listing of the current EV-2 FPGA firmware memory map.

Microcontroller Port Connections:

Port 0: used for the address/data (AD) bus. Once configured, the FPGA latches the lower address byte from the AD lines.

Port 1: used for several purposes as shown in Table 4 on page 10.

Bit #	Name of Signal	I/O	Description
0	INIT_IO#	I	Used when configuring the FPGA. Refer to Xilinx Spartan datasheet for more detail.
1	PROGRAM#	O	Used to initiate the FPGA configuration. Refer to Xilinx Spartan datasheet for more detail.
2	MUTE#	I	Mute signal from the CM module
3	HEX_DATA_IN	O	Not used. May be used to concatenate settings from other hex switches.
4	HEX_CLOCK	O	Used to latch the hex switch values into a serial shift register.
5	HEX_SHIFT	O	Used to shift the hex switch values from the serial shift register.
6	HEX_DATA_OUT	O	The hex switch value from the serial shift register appears at this input.
7	RS232	O	This is not operational and should be ignored.

Table 4: Port 1 Signal Descriptions

Port 2: upper address bus. Port 2 is output only.

Port 3: See Table 5 on page 11.

Bit #	Name of Signal	I/O	Description
0	RXD	I	RS232 serial port receive signal.
1	TXD	O	RS232 serial port transmit signal
2	HREQ#	O	Connected to the CM module host request signal. See CobraNet Technical Datasheet for a complete description of this signal.
3	HACK#	I	Connected to the CM module host acknowledge signal. See CobraNet Technical Datasheet for a complete description of this signal. May be used as an interrupt request on the microcontroller.
4	Watchdog	I	Watchdog signal from the CM
5	MCU_P35	I/O	Connected to SCI_CLK via the FPGA.
6	WR#	O	Microcontroller write signal.
7	RD#	O	Microcontroller read signal.

Table 5: Port 3 Signal Descriptions

Interfacing the Microcontroller to the CM

Please refer to the EV-2 schematic, found in *Appendix D* for information regarding interfacing to the CM.

The CM has a host interface that allows a host processor (such as an 8051 microcontroller) to interface to the DSP on the CM. From a hardware perspective the interface to the CM-1 and CM-2 is almost the same,. The host interface signals are a data strobe signal, HDS#; a read/write line, HRW, an 8-bit bi-directional data bus, HD0-HD7, and three address lines, HA0-HA2 on the CM-1 and four address lines, HA0-HA3 on the CM-2. The HEN# line has been configured by the CobraNet software to be ignored or seen as a logic low. Given this host configuration, the interface of the microcontroller to the CM host port is straightforward. In addition to the above signals there are two more, HACK# and HREQ# which can be used as flags to indicate a state change on the CM.

With regard to the CM-1 which uses a Motorola DSP56303, care must be taken with the timing of HDS# and HWR. Motorola's timing specifications for the DSP56303 host port in a non-multiplexed, single data strobe mode requires a set up time from the falling edge of HWR# to the falling edge of HDS# of 4.7ns and the hold time from the rising edge of HDS# to the rising edge of HWR# of 3.3ns. The pulse of the HDS# signal must be wholly within the pulse of the HWR# signal with the constraints stated above. Please refer to Motorola's DSP56303 Technical Data sheet for complete information regarding timing and interface issues. This is available for download from the Motorola web site at <http://www.motorola.com/SPS/DSP/>.

In the EV-2 application, the host address lines are generated by the address latch in the FPGA (see Table 6 on page 14) and the host data bus is connected directly to the data bus of the microcontroller. The HREQ# and HACK# signals are connected to the two interrupt inputs of the microcontroller. These signals may be used for data handshaking and asynchronous notification respectively.

The final host signal, HRESET#, resets the CM when asserted low. Setting a bit in the host reset register (see Table 6 on page 14) controls this signal. See the discussion of the FPGA below for more information about this signal.

Supplemental information regarding the CM Host interface may be found in the section titled "Host Management Interface" in the CS181xx data sheet available on the Cirrus Logic website: www.cirrus.com.

Programming the Microcontroller

The EV-2 is designed so that field updates of both the microcontroller firmware and the FPGA firmware are possible. If only the efficacy and performance of the CobraNet paradigm is being evaluated, reprogramming of the microcontroller is not required. However, use of the field program capability may aid in the design of a CobraNet based product.

Modifying the Flash Program Memory of the microcontroller constitutes the update. To program the flash, either the provided software or a copy of WINISP may be used. (WINISP can be downloaded from the Philips Semiconductor website: <http://www-us.semiconductors.philips.com/>)

The programming instructions that follow pertain to the supplied EV-2 routing/programming software, CNEval.exe. Programming the microcontroller is a multi-stage process:

1. Install the EV-2 CNEval.exe software on your Windows-based computer.
2. Install an RS232 cable from port 1 or 2 on your PC to P501, the 9-pin D-type connector closest to the center of the board.
3. Run CNEval.exe
4. Change the hex switches (SW201-SW204) to FFF8 (as viewed when looking at the hex switches). You may, as an alternative, press the key combination *Control+Shift+F9* to enable the Utility-Program menu item.
5. Select "Program" from the Utility menu in CNEval.exe.
6. From the drop-down list box located in the main programming window, select a serial connection, either port COM1 or COM2 based on which is connected.
7. Located near the two serial RS232 connectors is a switch, SW200. This switch must be set to the program mode position. The program position is indicated by silk screen on the EV-2 board.
8. Push switch SW508, the momentary reset switch. SW508 is located just behind the hex switches.
9. Select which firmware to update, either the FPGA or the 8051.
10. Wait for programming to complete. Do not interrupt the programming process!
11. Once programming has completed for the microcontroller or the FPGA firmware, return the programming switch, SW200, to the normal operation position and press the reset switch, SW508.

12. Click OK to return to the routing window in CNEval.exe.

Note: When using Philip Semiconductor's WINISP application, instructions 2, 7, 8, 10 and 11 are applicable.

Interfacing Serial Audio to the CM

In general, interfacing to most off-the-shelf A/D and D/A converters is straightforward and the CM is no exception. Most signals for a direct connection to these as well as other audio ICs such as the CS8420 AES/EBU transceiver, are available on the CM module interface connector. Most converters provide for a choice of bit clock and sample (frame) clock polarity, as well as audio data formats such as SPI or I2S.

The A/D converter, a Cirrus Logic CS5396 introduces one wrinkle. The CS5396 needs a master clock of 256FS (one that is 256 times the sample rate frequency). This is not directly supplied on the CM interface connector. Additional circuitry is required to derive a 256FS clock from the 512FS clock supplied by the CM. The CS5396 is configured for slave operation, which means that it requires a bit clock and a sample (frame) clock input. The bit and sample clocks are direct connections from SCK and FS1 respectively, as is the data stream which comes from one of the SSI ports. The CM can be configured to clock data from either edge of the bit clock, as well as allowing for specifying the polarity of the sample clock. (See the CobraNet Technology Datasheet for more information.) This is important since the CS5396 works with sample pairs which need to be phase aligned. The polarity of the sample clock specifies this alignment. For the EV-2 application, the SSI ports of the CM have been programmed to send two channels per port. This allows a straightforward connection without any demultiplexing.

The connection to the CS4396 D/A converter is straightforward, as it is able to use the 512FS clock directly from the CM for its master clock. Like the CS5396, it uses the bit clock and sample clock directly from the CM. Data to the DAC and from the ADC are also direct except that they pass through a selector circuit in the FPGA. If a particular design does not include multiple sources of audio, then the connection can be direct to the CM interface connector.

FPGA

The Field Programmable Logic Array is a Xilinx Spartan XCS10XLVQ100-4. It is mapped into the microcontroller's memory space. The microcontroller must configure the FPGA after power on or reset. Express mode configuration is used for this part. Refer to the Xilinx Spartan XL family data sheet for more information on the Express mode configuration operation. This data sheet can be found at the Xilinx web site, <http://www.xilinx.com/>. The address for configuration is 0x8800. Once configured, the FPGA's two main functions are to decode the microcontroller's address signals and to route audio from a user selected source to a user selected destination. Secondary functions are to generate a sine wave signal and implement registers whose function are mostly of a control nature. A discussion of the memory decoding, routing, sine wave generation and other functions follow.

The memory map of the upper 32k of the microcontroller space after configuration, is shown in Table 6 on page 14. All bit-defined locations use the least significant microcontroller data bus signal AD0 as the

controlling bit. Other data bits are ignored on these registers. Power on and reset default for all registers is 0 unless specified otherwise.

Memory Location	R/W	Description
0x8000	W	Bit register for green LED, CR903. 0=LED on, 1=LED off. Refer to Table 10 on page 17 for this and other LED registers.
0x8001	W	Bit register for red LED, CR904. 0=LED on, 1=LED off.
0x8002	W	Bit register for yellow LED, CR905. 0=LED on, 1=LED off.
0x8004	W	Bit register for green LED blink control. 0=blink off, 1=blink on.
0x8005	W	Bit register for green LED blink control. 0=blink off, 1=blink on.
0x8006	W	Bit register for green LED blink control. 0=blink off, 1=blink on.
0x8008	W	DAC audio routing address (see Table 7 on page 16).
0x8009	W	Bit register for DAC mute signal. 0=mute on, 1=mute off.
0x800A	W	Bit register for DAC sample rate mode. 0=48k, 1=96k.
0x800B	W	Bit register for DAC reset signal. 0=reset on, 1=reset off.
0x8010	R	ADC Calibration Status. 1=Calibrating, 0=Ready. See the <i>Calibrating the ADC</i> section for details.
0x8010	W	Manual ADC Calibration. 0=Normal, 1=Calibrate. See the <i>Calibrating the ADC</i> section for details.
0x8011	W	Bit register for ADC slave/master control. 0=Slave, 1=Master.
0x8018	W	AES/EBU audio routing address (see Table 7 on page 16).
0x8019	W	Bit register for AES/EBU mute signal. 0=AES output muted, 1=Unmuted.
0x8020	W	SSI 0 audio routing address (see Table 7 on page 16).
0x8021	W	Bit register for SSI 0 mute signal. 0=Muted, 1=Unmuted.
0x8028	W	SSI 1 audio routing address (see Table 7 on page 16).
0x8029	W	Bit register for SSI 1 mute signal. 0=Muted, 1=Unmuted.
0x8030	W	SSI 2 audio routing address (Table 7 on page 16).
0x8031	W	Bit register for SSI 2 mute signal. 0=Muted, 1=Unmuted.
0x8038	W	SSI 3 audio routing address (Table 7 on page 16).
0x8039	W	Bit register for SSI 3 mute signal. 0=Muted, 1=Unmuted.
0x8040	Note 1	CM-1 Host Port ICR register. CM-2 Message-A register.

Table 6: Microcontroller Memory Map of Upper 32k After FPGA Configuration

Memory Location	R/W	Description
0x8041	Note 1	CM-1 Host Port CVR register. CM-2 Message-B register.
0x8042	Note 1	CM-1 Host Port ISR register. CM-2 Message-C register.
0x8043	Note 1	CM-1 Host Port IVR register. CM-2 Message-D register.
0x8044	Note 1	CM-1: Unused. CM-2 Data-A register.
0x8045	Note 1	CM-1 Host Port Data register high. CM-2 Data-B register.
0x8046	Note 1	CM-1 Host Port Data register middle. CM-2 Data-C register.
0x8047	Note 1	CM-1 Host Port Data register low. CM-2 Data-D register.
0x8048	R/W	CM-2 Host Control Register.
0x8049	R/W	CM-2 Host Status Register.
0x8051	W	Bit register for Host reset signal. 0=Asserted, 1=Deasserted.
0x8058	R/W	Auxiliary lines. Not used, for test purposes only.
0x8060	R	FPGA configuration major version.
0x8061	R	FPGA configuration minor version.
0x8062	R	FPGA configuration revision number.
0x8070	R/W	Sinewave Gain register. See Table 9 on page 17.
0x8078	R/W	Sinewave Frequency register. See Table 8 on page 16.

Table 6: Microcontroller Memory Map of Upper 32k After FPGA Configuration (Cont'd)

Note 1: The FPGA only decodes this address. The actual register located is on the CM. See the Motorola DSP56303 users manual for a discussion of each of these host port registers.

Configuring the FPGA

The FPGA is configured from data that is stored in the upper 16kbytes (0xC000-0xFFFF) of the microcontroller's Flash Program Memory. The microcontroller code for configuring the FPGA uses express mode which writes byte-wide data to the FPGA. Refer to the Xilinx Spartan XL family data sheet for more information on the express mode configuration operation. The address used for writing configuration data is 0x8800.

Functional Discussion of FPGA Operation

Routing of Audio Data

The routing of audio data is achieved by a simple 8 to 1 multiplexing operation; for each audio destination three data bits in an a register in the FPGA select the source. For example, the three data bits in the D/A audio routing register determine which audio source is selected to appear at the analog outputs (J401). Table 7 on page 16 shows the definition of the data bits and the respective

audio source.

microcontroller data bit AD2 AD1 AD0			Audio Source
0	0	0	CM SSI 0
0	0	1	CM SSI 1
0	1	0	CM SSI 2
0	1	1	CM SSI 3
1	0	0	ADC Low-latency
1	0	1	ADC High-quality
1	1	0	AES/EBU Input
1	1	1	Sine wave

Table 7: Definition of Audio Routing Register Bits

Sine Wave Generator

The FPGA contains a 32-sample, 24-bit, sine table. The table is stepped through at the sample clock rate so the resulting fundamental frequency is 48kHz / 32 samples = 1500Hz and 3000Hz at 96kHz. Limited control over frequency and gain is provided. Listed below are the values to write to the frequency and gain registers in the FPGA.

Frequency register data bits AD3 AD2 AD1 AD0				Frequency 48kHz sample rate (96kHz sample rate)
0	0	0	1	1.5 kHz (3.0 kHz)
0	0	1	0	3.0 kHz (6.0 kHz)
0	0	1	1	4.5 kHz (9.0 kHz)
0	1	0	0	6.0 kHz (12.0 kHz)
0	1	0	1	7.5 kHz (15.0 kHz)
0	1	1	0	9.0 kHz (18.0 kHz)
0	1	1	1	10.5 kHz (21.0 kHz)
1	0	0	0	12.0 kHz (24.0 kHz)

Table 8: Sine Wave Frequency Register Bit Definitions

Gain register data bits		Gain
AD1	AD0	
0	0	0dB
0	1	-6dB
1	0	-12dB
1	1	-18dB

Table 9: Sine Wave Gain Register Bit Definitions

LED Control

There are two bit registers to control the state of each of three LEDs. The mapping of control bits to LED behavior is described in Table 10 on page 17. The data bit is always AD0.

Enable	Blink	Status
0	0	off
1	0	on
0	1	off
1	1	blink

Table 10: LED Status

Calibrating the ADC

The CS5396 ADC requires approximately ten seconds to achieve proper operating temperature after which, for optimal performance, it is necessary to initiate a calibration cycle. After power is applied and after reset the EV-2 uses an internal timer to count off approximately 10 seconds before beginning the calibration cycle. Prior to and during the calibration, the ADC calibration status register will read 1. Once calibration has been completed, the status register will read 0. Calibration can also be initiated manually by the microcontroller by writing a 1 to the Manual ADC Calibration register and then immediately returning the register to the 0 state. The FPGA initiates this one-time power up/reset calibration.

Version Control

The FPGA contains three hardwired eight-bit registers that contain an ASCII version/revision number of the FPGA configuration file. The microcontroller can read these registers for version control and reporting purposes.

Mute Control

Muting comes from three different sources 1) the microcontroller can mute or unmute audio by writing to a bit control register. There is one mute bit control register for each output audio path, 2) the CM asserts its mute signal, and 3) all audio is unconditionally muted during a power on/reset calibration cycle. If the microcontroller initiates a calibration cycle using the Manual ADC Calibration register, the ADC automatically mutes itself (i.e., gates off its digital output stream until calibration has completed).

Hex Switches

Four pins on the 8051 allow the hex switches to be read. The EV-2 circuitry associated with the hex switches serves as a good example implementing this common CobraNet feature (see *Recommended User Interface Practices* section in the CobraNet Programmer's Manual for a discussion of use of this scheme). Requirements include a physical (hardware) mapping of the hex switches to a code (software) within the CM. The methodology to achieve this is listed below:

1. When viewing the hex switches from the front, the rightmost switch should be the first nibble of the serial data stream, and subsequent nibbles should follow working from the second rightmost switch to the leftmost switch.
2. Two of the four signals will be control signals: a Shift/Load# signal where Shift is high and Load is Low. The Load allows for parallel, asynchronous loading of the hex switch data into a shift register and the Shift allows for serial shifting of data out of that register. A clock signal to perform the shifting operation where data changes on the rising end of the clock. The 74HC165 IC is an example of a part that supports this protocol.
3. The other two signals are the shifted data output and an input that will be shifted serially into the shift register concatenating it with the hex data. The intent of this latter signal is to allow for the possibility of concatenating other data, additional hex switches or otherwise, for application specific enhancements.
4. The software will convert the serial hex data stream to a four-byte ASCII value that represents the switch settings.



Figure 8. Example Switch Setting

As shown in *Figure 8*, viewing hex switches from the front, the given switch positions would read as "CA30" in a software query of the hex switch setting. On the EV-2, the microcontroller is responsible for reading the switches through the hardware serial interface and converting those readings to an ASCII representation. This representation is then written to the CM through the host port. Specifically, the EV-2 microcontroller updates the CM's MI variable, *sysName*. Using SNMP, the user may query this variable. The SNMP response is of the form "PEAK_AUDIO_EVAL-SWwxyz", where the wxyz represents the hex values of the switches in ASCII format. In the example shown in *Figure 8*, a query of *sysName* would return "PEAK_AUDIO_EVAL_SWCA30".

EV-2 Schematics, Page-by-Page Description

The following sections provide detailed descriptions of the EV-2 schematic drawings contained in *Appendix D*.

Block Diagram

This page is a hierarchical block diagram showing an overview of all schematic pages and interconnects between pages.

Microcontroller and Hex Switches

This page shows an 8051-type microcontroller, its connections, and peripherals. Peripherals include 32kbytes of SRAM, hex switch interface, clock oscillator and programming switch. The programming switch circuitry has additional settings via jumpers R204 and R207 to program the Flash with either +5 volts or 12 volts. Voltage selected depends on the particular model of 8051-type microcontroller installed. Either R204 or R207 should be installed, but not both.

A/D Converter

This circuit is based on the Cirrus Logic CS5396 reference design. See the Cirrus Logic website, <http://www.cirrus.com/>, for a detailed description of the CS5396 and its reference design, the CDB5396. There is also a white paper at the Cirrus website titled *Techniques to Measure and Maximize the Performance of a 120 dB, 24-bit, 96 kHz A/D Converter Integrated Circuit* that provides an in-depth analysis of the analog input conditioning circuit. One notable deviation in the EV-2 circuit with respect to the Cirrus reference design is that the EV-2 circuit only runs at 48 kHz (64x oversampling mode) and only in stand-alone mode.

D/A Converter

This circuit uses the Cirrus Logic CS4396 D/A Converter. See the Cirrus Logic website, <http://www.cirrus.com/>, for a detailed description of the CS4396. On the EV-2 the mute transistors on the analog outputs are omitted. The CS4396 in the EV-2 design runs in single sample rate and stand-alone mode.

Connectors and Interfaces

This page shows the CM interface connectors, P510 and P511, as well as the RS232 interface. Also included on this page are:

- The master audio clock (FS512) divide-by-two circuit (U506) that provides a clock for the A/D converter.
- Buffering and serial termination (U510) for the bit clocks and sample (frame) clocks that go to the A/D, D/A converters and the AES/EBU transceiver. Buffering and termination reduces over and under shoot. Over and under shoot may adversely affect audio performance and increase RF emissions. This conditioning is recommended but not necessary in all applications.
- The reset switch circuit, SW508 and associated components.

Optional VCXO

Although the CM produces a high quality master clock, in some applications, the master clock may be compromised by long or noisy signal paths (i.e. ribbon cable connection). An optional VCXO circuit was included as an example of re-clocking the master clock (FS512). This circuit is designed to

attenuate jitter. The VCXO is not installed on the current EV-2 board. Installing this circuit in the EV-2 does not improve or degrade audio performance. To install the VCXO, remove R501 (see the drawing *Connectors and Interfaces* on page 5) and install all components on this page with the exception of R601.

AES/EBU Transceiver

This circuit uses the Cirrus Logic CS8420 AES/EBU Transceiver. See the Cirrus Logic website, <http://www.cirrus.com/>, for a detailed description of the CS8420 as well as the CDB8420, an evaluation board for the CS8420. The EV-2 application runs the CS8420 in AES/EBU transceiver mode with input sample rate conversion. For the AES/EBU transceiver to operate properly, a valid AES/EBU signal must be provided at the AES/EBU input.

Power Supply Conditioning

The main power connector is a standard ATX connector. The voltage mains are conditioned, as well as protected with transient voltage suppressor diodes. Numerous voltage regulators are used to filter and condition the power supplied to the analog audio section.

FPGA

This page shows the connections to the FPGA, which is a Xilinx XCS10XL-4VQ100 IC. See the FPGA discussion above for a detailed description of its functionality.

Appendix A: Definition of Terms

This Appendix contains brief definitions of many of the terms used in the discussion of CobraNet and CobraNet networks.

Audio Channel

A single audio signal. Audio channels on CobraNet have a 48KHz sampling rate and may be 16, 20 or 24 bit resolution. Multiple audio channels may be carried in a Bundle.

Broadcast Addressing

Broadcasting is a special case of Multicasting (see multicast below). Whereas it is possible, in some cases, to indicate intended recipients of multicast data, broadcast data is unconditionally received by all DTEs within a network domain.

Bundle

The basic network transmission unit under CobraNet. Up to 8 audio channels may be carried in a Bundle.

Category 5 Cable (CAT5)

CAT5 is inexpensive unshielded twisted pair (UTP) data grade cable. It is very similar to ubiquitous telephone cable but the pairs are more tightly twisted. CAT5 cable runs for Ethernet are limited to 100 meters due to signal radiation and attenuation considerations. A CAT5 run in excess of 100 meters may be overly sensitive to electromagnetic interference (EMI). It should be noted that not all CAT5 cable is UTP. Shielded CAT5 also exists but is rare due to its greater cost and a much shorter distance limitations than UTP CAT5.

CobraNet

CobraNet is a combination of hardware, software and protocol allowing distribution of many channels of digital audio over Fast Ethernet. CobraNet supports switched and repeater Ethernet networks. On a repeater network, CobraNet eliminates collisions and allows full bandwidth utilization of the network. CobraNet uses standard Ethernet packet structure and network infrastructure.

CobraNet Device

A device in compliance with the CobraNet specification for transmission and/or reception of digital audio and associated sample clock.

Conductor

CobraNet Device on the network supplying master clock. A conductor arbitration procedure insures that, at any time, there is one and only one conductor per network.

Crossover Cable

A crossover cable can be used to directly connect two network devices.

DTE

Short for Data Terminal Equipment, a DTE is any network device that produces or consumes data. A CobraNet Device is a DTE.

Ethernet

A Local Area Network (LAN) technology that allows transmission of information between computers. Ethernet is, by far, the most widely deployed LAN standard worldwide.

Fast Ethernet

A newer version of Ethernet, also known as 100BASE-T. It supports data transfer rates of 100Mbps. CobraNet operates on a Fast Ethernet network.

Full Duplex

Data can be transmitted and received simultaneously.

Half Duplex

Data can only be transmitted in one direction at a time.

Hub

Hub is not a technically concise term. The term can be used to refer to either a Repeater Hub or a Switching Hub.

Mbps

Short for megabits per second, it is a measure of data transfer speed.

Multicast Addressing

Data which is Multicast is addressed to a group of, or all DTEs on a network. All DTEs receive multicast addressed data and decide individually whether the data is relevant to them. A Switched Hub is typically not able to determine appropriate destination port or ports for multicast data and thus must send the data out all ports simultaneously just as a Repeater Hub does. Multicast addressing is to be avoided when possible since it uses bandwidth network wide and since all DTEs are burdened with having to decide whether multicast data is relevant to them.

Multicast Bundle

A multicast Bundle supports a one-to-many routing of audio on the network. Ethernet multicast addressing is used to deliver a multicast Bundle. Because a multicast bundle consumes bandwidth network-wide, use of this delivery service must be rationed on a switched network.

Network Topology

The physical and logical relationship of nodes in a network; i.e., a star, ring, tree, bus, etc.

Node

A processing location. A node can be a computer, a switch, a CobraNet device, or any other device that has a unique network address.

Repeater Hub

An Ethernet multi-port repeater. A data signal arriving in any port is electrically regenerated and reproduced out all other ports on the hub. An Ethernet network is typically wired in a star configuration and the hub is at the center. The use of hubs requires that all devices on the network run in half duplex mode.

Run Length

Each type of media has a limitation in the length of a point-to-point run between two devices. When maximum run length guidelines are exceeded it may not be possible to establish a valid network connection or data may be corrupted. Longer distances can be achieved by upgrading the media or using multiple runs in series.

Switch/Switching Hub

A Switch examines addressing fields on data arriving at each port and attempts to direct the data out the port or ports to which the data is addressed. Data may be buffered within the Switching Hub to

avoid the collision condition that may be experienced within a Repeater Hub. A network utilizing Switching Hubs realizes higher overall bandwidth capacity as data may be received through multiple ports simultaneously without conflict. Switches are full-duplex devices. A network utilizing switches to connect network segments is referred to as a switched network.

Unicast Addressing

Data which is unicast is addressed to a specific DTE. A Switching Hub may examine the unicast address field of the data and determine on which port the addressed DTE resides and direct the data only to that port. Delivery of an e-mail message is an example of unicast data addressing.

Unicast Bundle

A unicast Bundle supports a one-to-one routing of audio on the network. Ethernet unicast addressing is used to deliver a Unicast Bundle. Because unicast addressing is friendly to Ethernet switches, unicast Bundles should be used for audio delivery whenever possible.

Unregulated Traffic

Refers to any data transmitted onto a network by non-CobraNet devices. Unregulated traffic is particularly offensive on a repeater network as it interferes with CobraNet's collision avoidance mechanism and can result in audio dropouts. On a switched network, unregulated traffic is only a problem if it appears in such copious quantity as to overload the network.

Appendix B: EV-2 Specifications

A/D: Cirrus Logic CS5396-KS

AUDIO SPECIFICATIONS:

- Two input channels.
- Frequency Response: 20 Hz to 20 kHz, ± 0.2 dB, with a -1 dB from full-scale sine wave input.
- Total Harmonic Distortion plus noise: $< 0.001\%$ @ 1 kHz, with a full-scale sine wave input signal.
- Dynamic Range: 111 dB A-weighted, 108 dB unweighted, measured with a -60 dB from full-scale 1kHz sine wave input signal.
- Maximum Input Level: +14.4dBu, balanced differential.
- Input Impedance: 15 k Ohms.

DIGITAL SPECIFICATIONS

- A/D quantization: 24-bit resolution.
- Audio Sampling Rate: 48kHz.

CONNECTOR: 6-Pin Phoenix-type connector.

Digital I/O: Cirrus Logic CS8420-CS

- AES/EBU input and output. Input is sample rate converted.

CONNECTOR: 6-Pin Phoenix-type connector.

OTHER SPECIFICATIONS

- Power Consumption: < 10 W (includes CM)
- Power Connector: Uses standard ATX power supply connector (ATX power supply not included).
- RS232 Connection: Non-standard (RX/TX only) EIA-RS232C connection with auto loop back. Connectors are 9-pin D-types.

D/A: Cirrus Logic CS4396-KS

AUDIO SPECIFICATIONS:

- Two output channels.
- Frequency Response: 20 Hz to 20 kHz, ± 0.2 dB, with a -1 dB from full-scale sine wave input.
- Total Harmonic Distortion plus noise: $< 0.002\%$ @ 1 kHz, with a full-scale output signal.
- Dynamic Range: 112 dB A-weighted, 109 dB unweighted, both with a -60 dB from full-scale output signal.
- Maximum Output Level: +15.8 dBu differential.
- Output Impedance: 100 Ohms designed to drive a minimum load of 600 Ohms

DIGITAL SPECIFICATIONS

- D/A quantization: 24-bit resolution
- Audio Sampling Rate: 48kHz

CONNECTOR: 6-Pin Phoenix-type connector.

Appendix C: Other Resources

A comprehensive array of CobraNet information can be accessed at the Cirrus Logic public website. Among the resources available are: FAQs, white papers, datasheets, programmer's guides, network design guidelines, common network terminology, a listing of recommended and tested Ethernet equipment and set-up information for selected Ethernet switches.

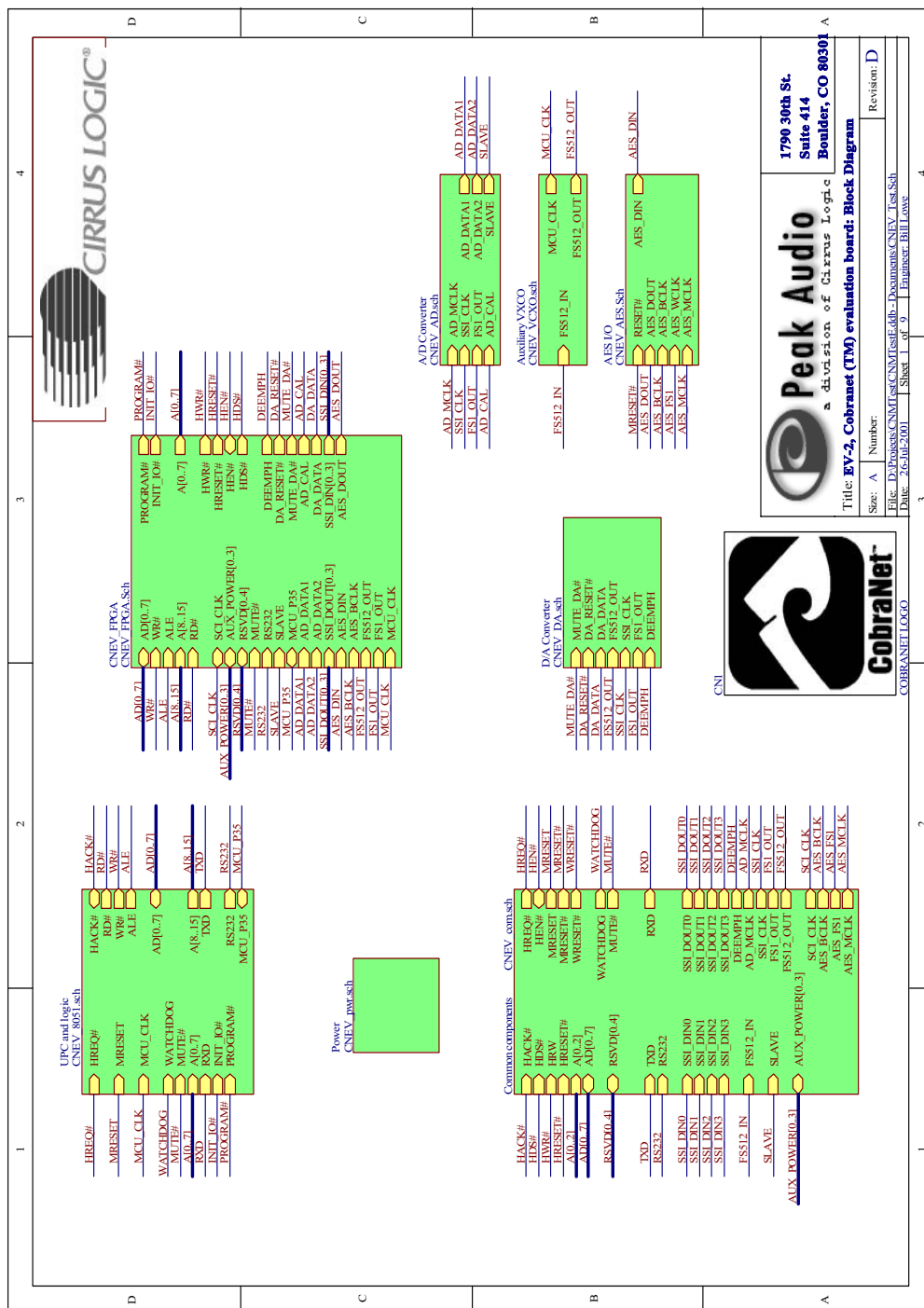
The URL for this site is: <http://www.cirrus.com>

A developer's website containing more in-depth technical information is also maintained which targets primarily CobraNet manufacturers and those considering integrating CobraNet into their products.

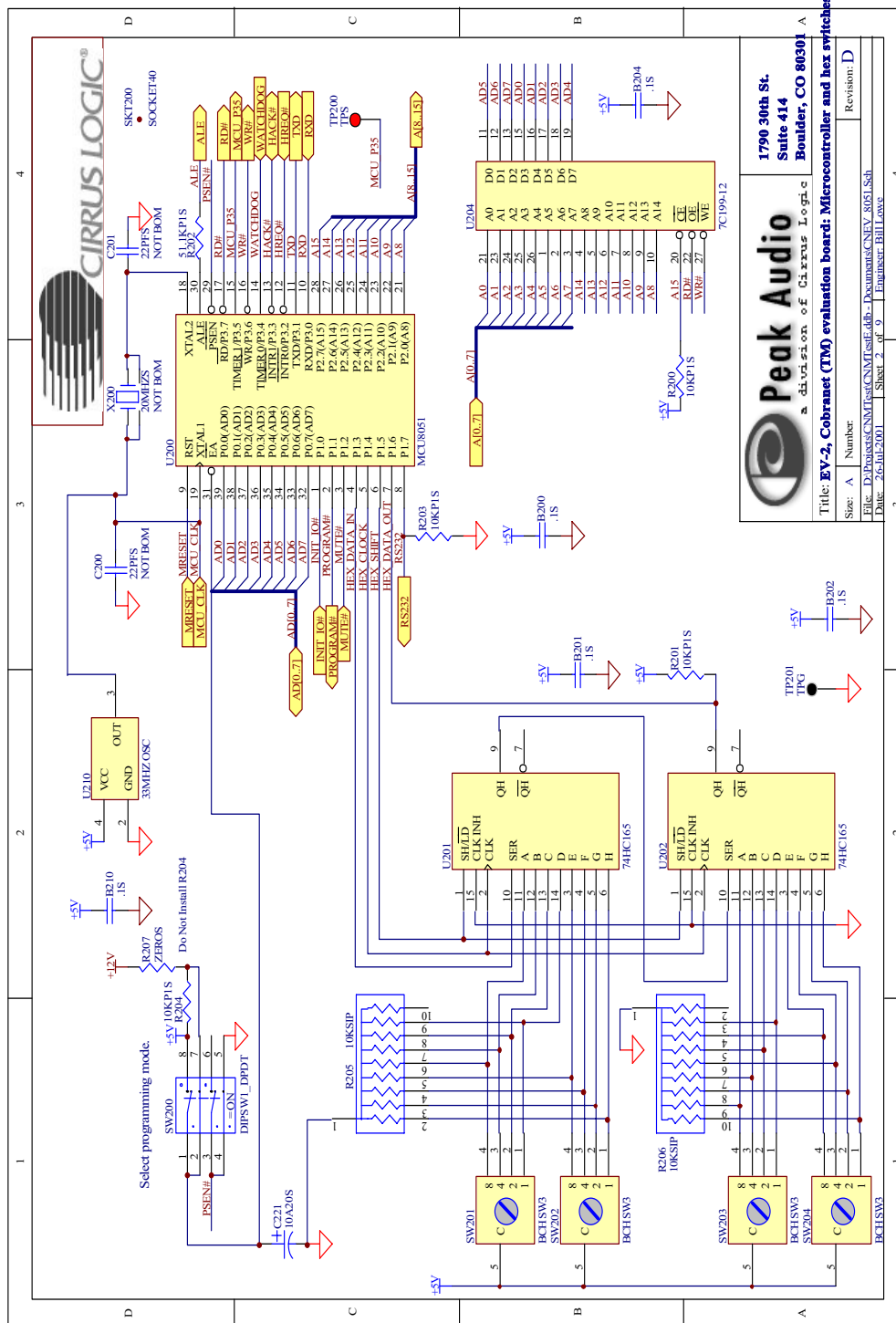
Access to the developer's website is granted subject to execution of a Non Disclosure Agreement. Please contact your local Cirrus Logic sales office or distributor for further details.

Although the EV-2 and CD-ROM shipped with this evaluation package should contain the most current CobraNet firmware and information, please check this against the developer's website or contact your local Cirrus Logic sales office or distributor for further details.

Page 27



Drawings



Peak Audio
a division of Cirrus Logic

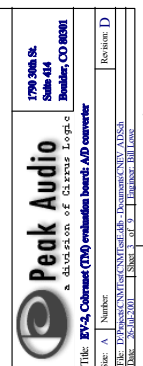
**1790 30th St.
Suite 414
Boulder, CO 80301**

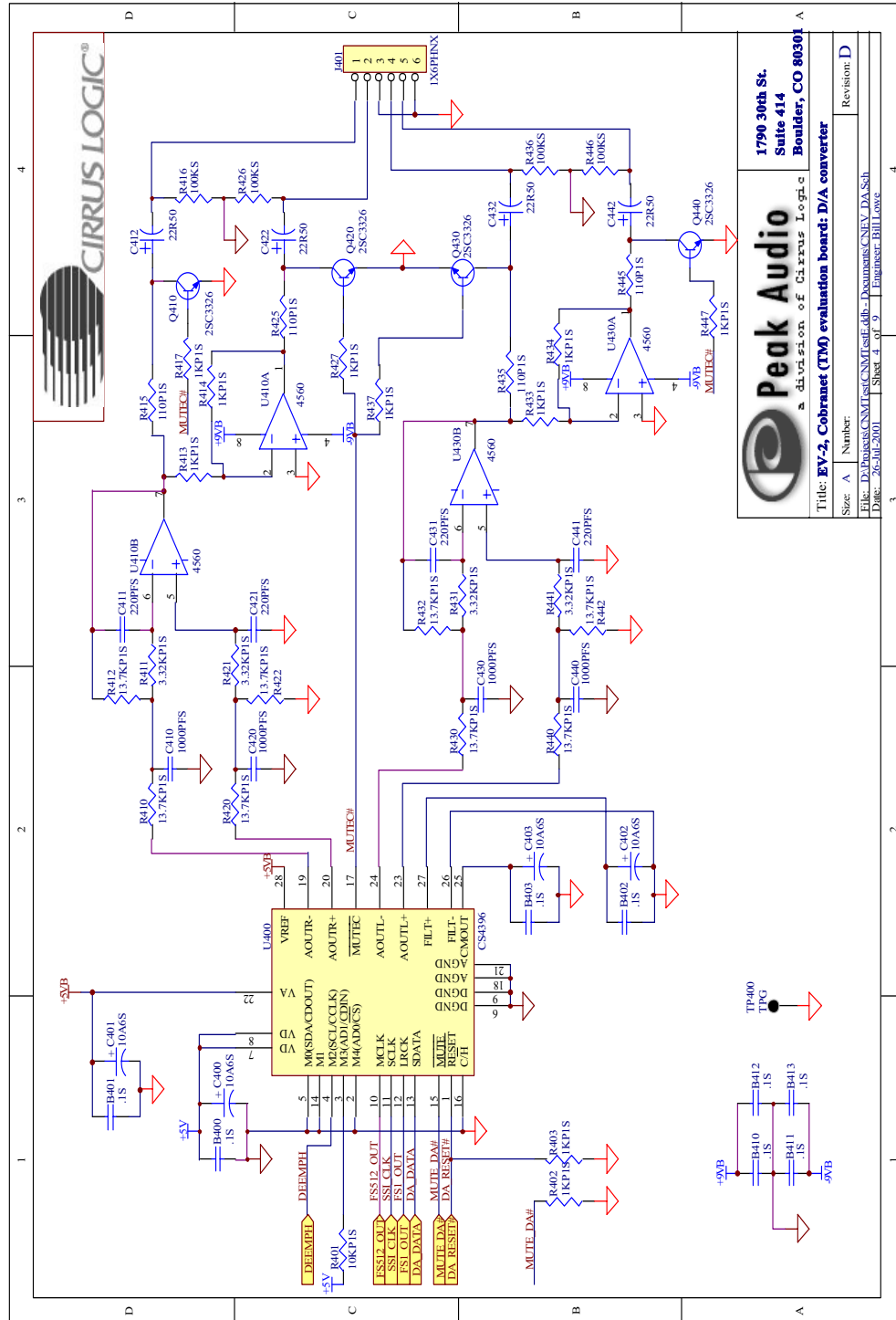
Title: **EV-2, CobraNet (TM) evaluation board: Microcontroller and hex switches**

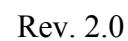
Size: A | Number: **D** | Revision: **D**

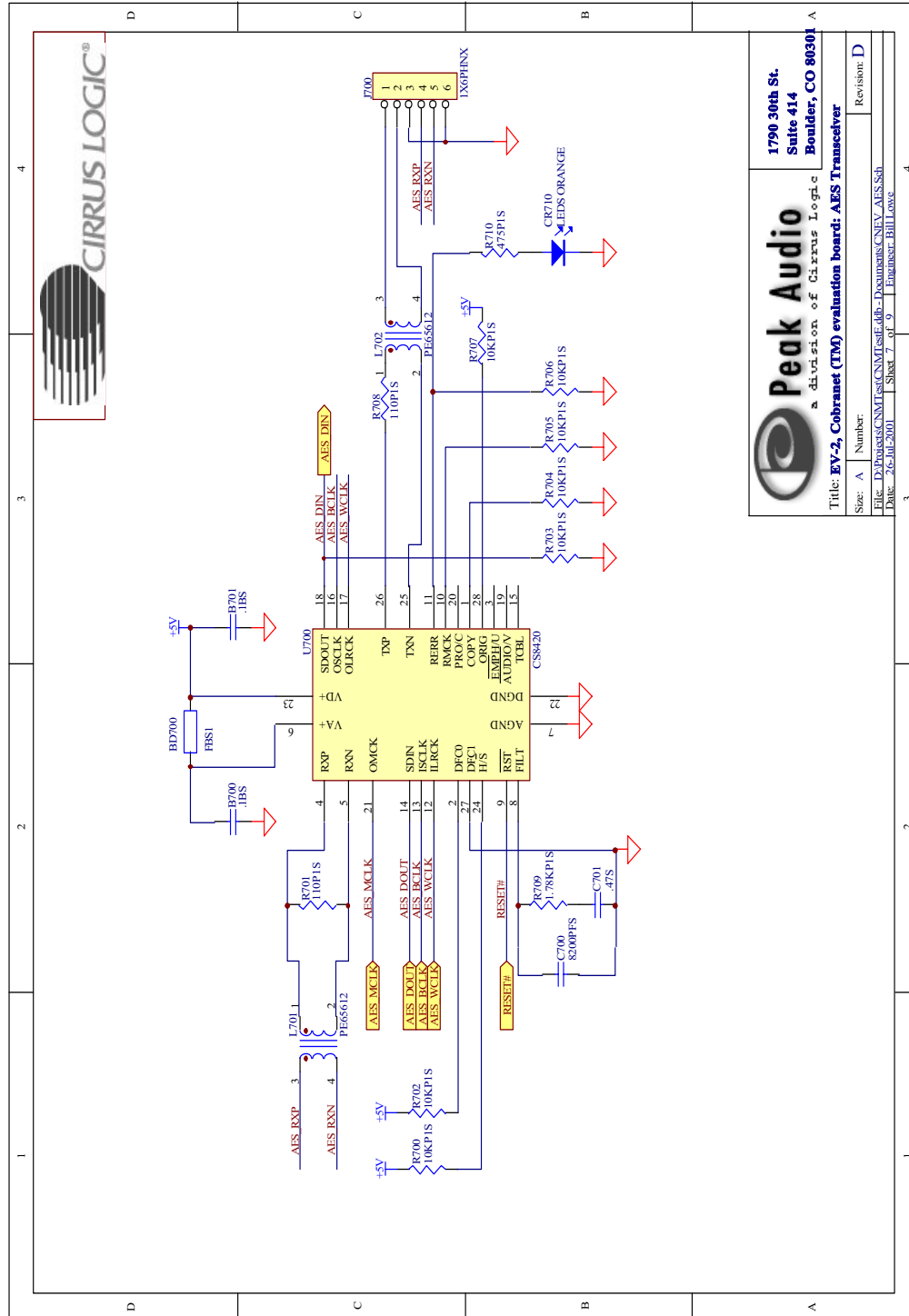
File: D:\btest\CMN\T\ev2\cmn\ev2.sch | Engineer: Bill Lowe

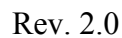
Date: 26-Jul-2001 | Sheet 2 of 9

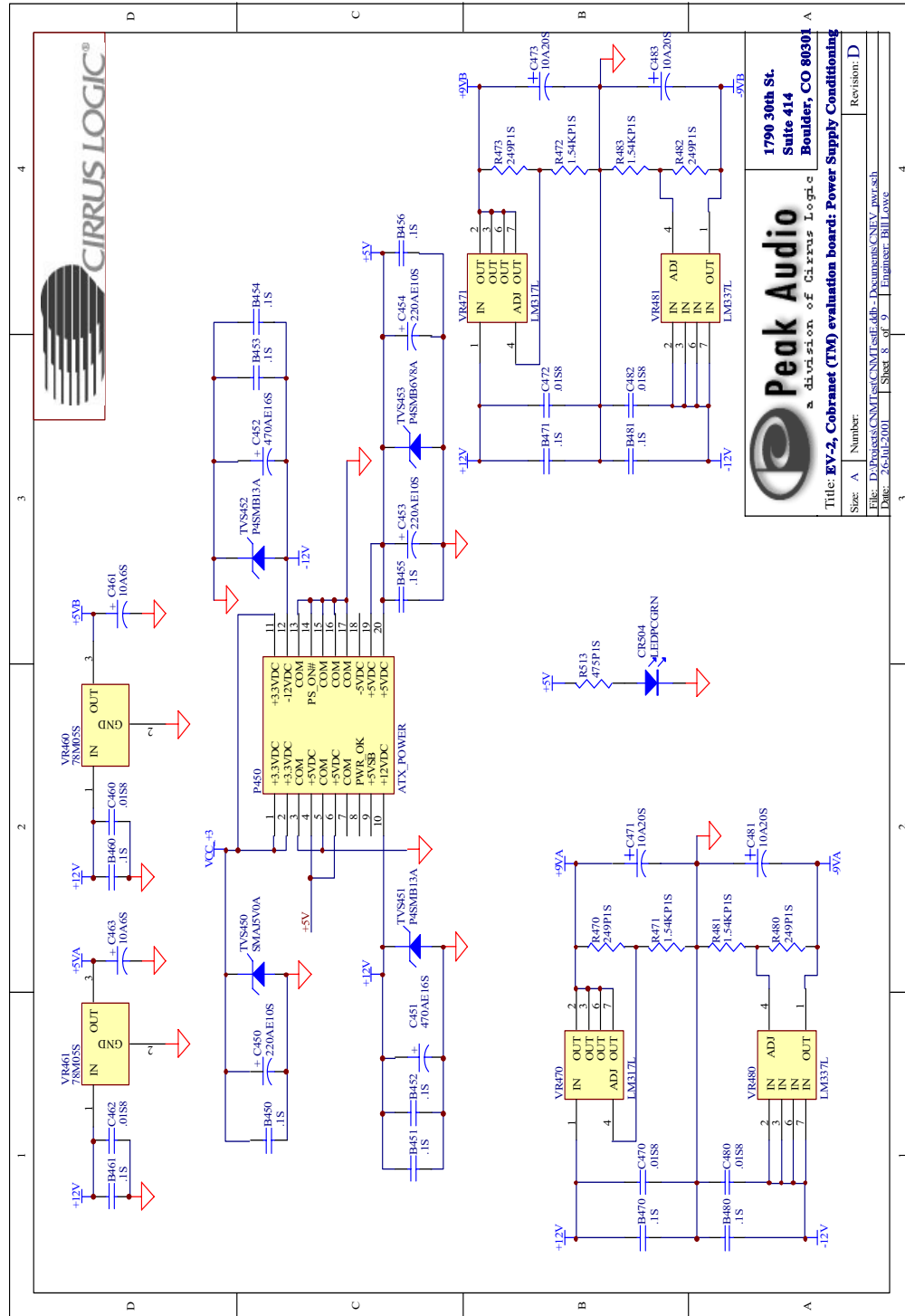


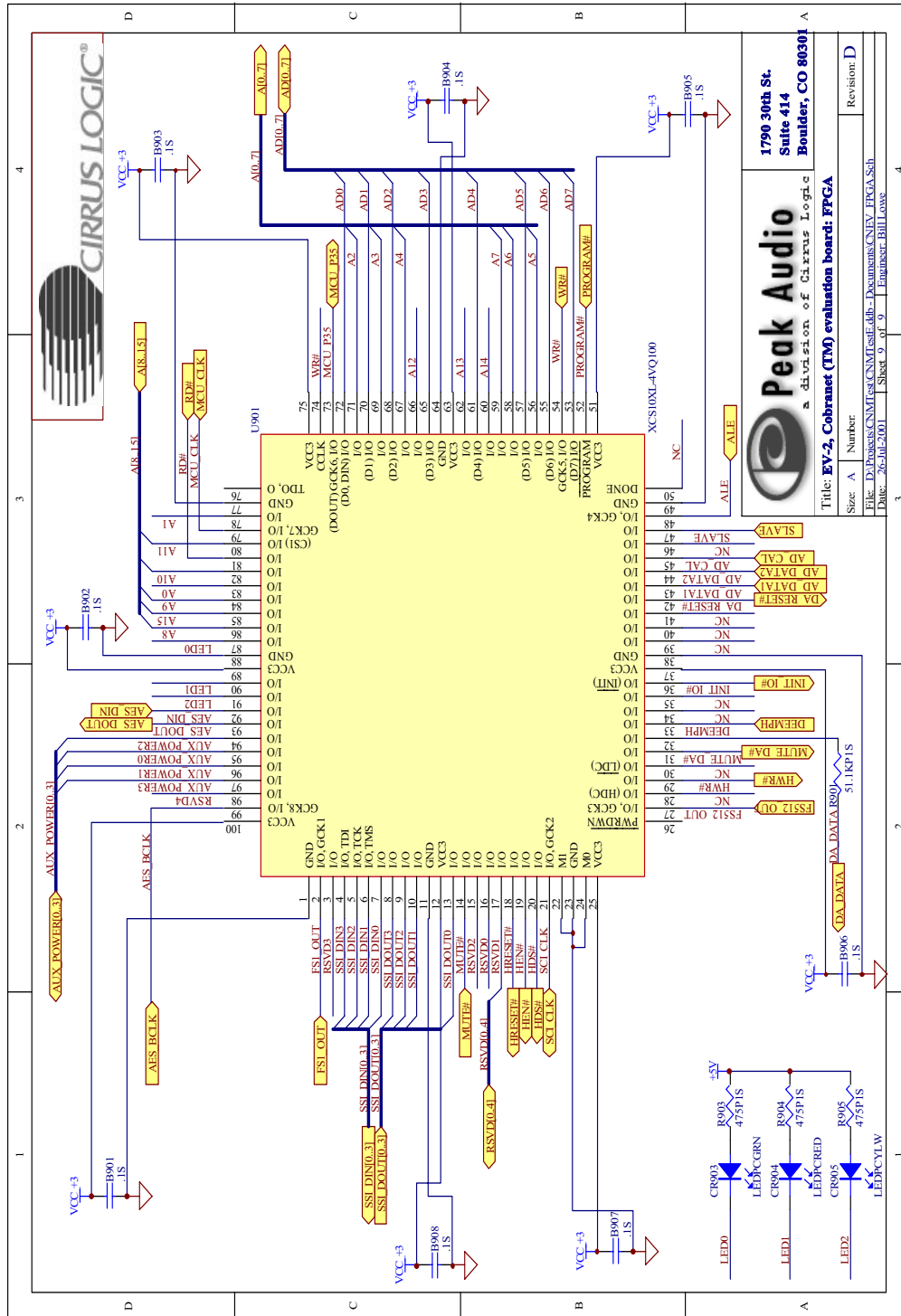












Appendix E: EV-2 Command Line Interface.

The EV-2 supports a simple command line interface (CLI). This interface allows the user to evaluate the CobraNet module (CM) and monitor and control Management Interface (MI) variables. A list and description of commands follow. Please reference the CobraNet Programmer's Manual for more information about the MI variables. Please note that there is a significant difference between the CM-1 and CM-2 MI variable format, i.e. how the raw data is stored in the CM's memory. Refer to the CobraNet Programmer's Manual regarding the MI variable formats.

Command line Syntax

The CLI is case insensitive, either upper or lower case in any combination may be used. Parameters enclosed within < > are mandatory with a few exceptions. At least one white space is required between command and parameters. Leading, lagging and multiple white spaces are ignored. A "ctrl-C" or "esc" will abort a command. With exceptions where noted all numeric values are to be entered or will be displayed in hexadecimal format with a leading "0x".

This command line interface allows the user to write scripts (i.e Python scripts) that monitor and control MI variables. Two commands are supported that allow the user to perform these tasks:

Peek <target> [offset]

This command will return the value at the given address location or the value of the MI variable.

<target> – a valid host address. The format of <target> is a hexadecimal number with a "0x" prefix.

Example: **peek 0xAB12C5**

<target> – the user may use an MI variable.

Example: **peek sysdescr**

In this example the CLI will return the system description in its entirety.

[offset] – optionally used with MI array variables only. Offset into the MI array. If not present on an MI array variable, the peek will return a value for the first location in the array.

Example: **peek txBundle 0x1000**

The above example will peek location 0x51100 or transmitter #1.

Example: **peek rxSubMap 0x2005**

This example will peek location 0x42005 or receiver #2, channel #5.

Poke <target> <value> [offset]

This command will set the value of the given location in CobraNet memory space to <value>.

<target> – same as peek above.

<value> – a hexadecimal with a "0x" prefix.

Example: **poke 0x40104 0x1011**

[offset] – optionally used with MI array variables only. Offset into the MI array. If not present on an MI array variable the poke will poke the <value> into the first location of the array.

Example: **poke txBundle 0x111 0x2000**

Please note the difference between the “**peek** <target = address>” and “<target = MI>” commands. The “**peek** <target = address>” command will return the raw value at the specified address location whereas the “**peek** “<target = MI>” command will return a properly formatted value. The difference is significant between the CM-1 and CM-2 where the variables are not stored in memory in the same format. An example is the *rxPriority* MI variable.

On the CM-1:

peek 0x40104	- command
0x101000	- what is returned
peek rxPriority	- command
0x1010	- what is returned

On the CM-2:

peek 0x40104	- command
0x1010	- what is returned
peek rxPriority	- command
0x1010	- what is returned

This difference here is that the “**peek 0x40104**” command returns the raw value as it is stored in memory, the “**peek rxPriority**” command returns the value as it should be represented. In this example the CM-1 returned a different value than the CM-2 for the “**peek 0x40104**” command but the same value for the “**peek rxPriority**” command; this is because the storage format of the variable in memory is different between the CM-1 and CM-2. The main memory architecture difference between the CM-1 and CM-2 is that the CM-1 has a 24-bit wide memory bus whereas the CM-2 has 32-bit wide memory bus. In the above example the *rxPriority* MI variable is an Integer16 data type. On the CM-1 this type of data uses the middle and upper byte of the 24-bit memory location to store the data whereas the CM-2 stores the data in the lower two bytes.

In summary, using the MI variable name will return data in the proper format. Using the address will return the raw data at the address location.

This also applies to the Poke command. When using the “**poke** <target = address> <value>” command you must pay attention to how the data (i.e. the <value>) is stored on the respective CM. Where supported, using the “**poke** <target=MI> <value> [offset]” command means that the format of <value> is independent of how the data is stored. This allows for writing a command line interface script that works on either the CM-1 or the CM-2.

There are a number of commands that control the state of the EV-2 board. Some of these commands should prove useful to the user when evaluating the CM. These commands are:

Route <input> <output>

<input> is the audio source on the EV-2 board, valid sources are:

- ssi0** – ssi port 0 from the CM.
- ssi1** – ssi port 1 from the CM.
- ssi2** – ssi port 2 from the CM.
- ssi3** – ssi port 3 from the CM.
- adc1** – digital audio output #1 from the Cirrus Logic CS5396 ADC
- adc2** – digital audio output #2 from the CS5396, this is a low group delay version of adc1.
- aes** – digital audio output from the Cirrus Logic CS8420 AES I/O IC.
- sine** – a 1500 Hz sinewave generated on the EV-2 board. This makes a useful test tone.
- mute** – sets the digital audio stream to all zeros.

<output> is the audio sink on the EV-2 board, valid outputs are:

- ssi0** – ssi port 0 to the CM.
- ssi1** – ssi port 1 to the CM.
- ssi2** – ssi port 2 to the CM.
- ssi3** – ssi port 3 to the CM.
- dac** – digital audio input to the Cirrus Logic CS4396 DAC.
- aes** – digital audio input from the Cirrus Logic CS8420 AES I/O IC.

Examples:

Route ssi0 dac

This will route the CM SSI stream #0 to the DAC.

Route sine aes

This will route a sinewave out to the AES output.

Please note that the route command only applies to audio on and from the perspective of the EV-2 board and not between Cobranet devices.

Led <color> <operation>

This command controls the state of the three LEDs on the EV-2 board. They may be used for development and debugging if so desired.

<color> = **red, yellow, green, all**. 'all' will perform the <operation> on all three LEDs.

<operation> =

- on**: turn led on
- off**: turn led off
- blink+**: blink LED, rate is about 2.2Hz
- blink-**: turn blink mode off
- toggle**: change the state of the LED.

Note: blink mode is independent of the LED state. When blink mode is turned off the state will be returned to the state that the LED was in prior to turning the blink mode on. 'Off', 'On', and 'Toggle' have no effect while blink mode is on but will execute while in blink mode, i.e. when blink mode is turned off the LED will assume the state of the last command.

Example: **led green toggle.**

This will change the state of the green LED. If it was on it will be off. If the LED was in blink mode the LED will continue to blink but will assume the opposite state prior to the toggle command if blinking is stopped.

Query <what>

This is a command to return information about the EV-2 and/or CM.

<what> =

system: returns information such as MAC address and software/firmware revision levels.

hex: returns the value of the hex switches on the EV-2 board.

status: Returns the status of the device. Generally the EV-2 and CM will be in one of two states: Ready or Calibrating. The latter indicates that the EV-2 has the ADC and DAC in calibration mode. This latter mode occurs only after power up and lasts for about 10 seconds.

cnmute: This returns the state of the mute signal coming from the CM. This is not to be confused with the mute choice of the route command. This is a signal on the CM interface that the CM controls. It is in either a muted or unmuted state. Querying this will return the state.

mac: The CM Ethernet MAC address is returned.

device: This will return the module type, CM-1 or CM-2.

hack: Returns the state of the CM HACK# signal, either a "0" or a "1".

hreq: Returns the state of the CM HREQ# signal, either a "0" or a "1".

mute: Returns a list of the outputs paths which are muted .

route: Returns a list of the input to output routing information.

gain: This parameter returns the gain information for the EV-2 sinewave.

frequency: Returns the frequency of the EV-2 sinewave.

Audio <operation> [value]

<operation> =

samplerate

[value] = **48k**: changes the CM samplerate to 48k.

96k: changes the CM samplerate to 96k. *

calibrate: put the audio converters through a calibration cycle

[value] = N/A

gain: sets the gain for the EV-2 test sinewave.

[value] = **0dB, -6dB, -12dB, -18dB** Default is 0dB.

frequency: sets the frequency for the EV-2 test sinewave.

[value] = **0x1-0x8** This value is a multiple of the fundamental frequency which is 1.5kHz for the 48kHz sample rate and 3.0 kHz for the 96kHz sample rate. Default is the fundamental.

*On EV-2 Rev. D boards, specific hardware modifications must be made in order to operate at 96k, please refer to the EV-2 96k modification document for more details.

TestEV <what>

This is used for manufacturing tests of the EV-2 and in general only the 'resetcn' will be useful to the user.

<what> =

memory – performs an EV-2 memory test, returns either a pass or fail.

watchdog – checks the CM watchdog signal to make sure it is in tolerance. Returns pass or fail and measured frequency of CM watchdog signal

host – performs a host test, returns pass or fail.

resetcn – resets the CM.

Packet <what> <arg2>

This command will perform a packet transfer between CM/EV-2 boards and is used to demonstrate and test the Cobranet packet bridge feature. Two packet types are supported, either command based or text based. The command packet will send an EV-2 line command to the other EV-2 which will return any results (an example would be a *query* command) as a text based packet. A text based packet will send text to the other EV-2 to be put out its 8051 serial port. This is much like an instant message.

<what> =

command: This tells the receiving CM that the following <arg2> is a line command.

<arg2>: this is any valid line command as described in this document.

text: this tells the receiving CM that the following <arg2> is text and will output the text to the serial port.

<arg2>: text that may be separated by spaces. Everything that follows the "text" parameter will be considered text.

on: turns packet processing on.

off: turns packet processing off.

Examples:

packet command query device

This example will return what type of CN module is on the other EV-2.

packet text how are you today?

This will send out the 8051 serial port of the other EV-2 the text: *how are you today?*

Please note that the **on** and **off** commands are from the perspective of the EV-2 and not the CM, i.e., the CM will still process commands from other sources if set up to do so.

Read <register>

This command reads the raw value of the host register. Note that the architecture is significantly different between the CM-1 and CM-2.

<register> =

(for CM-1, the DSP5303 host register interface.)

icr: read the icr register.

cvr: read the cvr register.

isr: read the isr register.

ivr: read the ivr register.

drh: read the drh (rxh) register

drm: read the drm (rxm) register

drl: read the drl (rxl) register.

(for CM-2, the CS181xx host register interface)

msg: read the msg register (returns four bytes)
data: reads the data register (returns four bytes)
msga: returns the value of the message A register.
msgb: returns the value of the message B register.
msgc: returns the value of the message C register.
msgd: returns the value of the message D register.
dataa: returns the value of the data A register.
datab: returns the value of the data B register.
datac: returns the value of the data C register.
datad: returns the value of the data D register.
control: returns the value of the control register.
status: returns the value of the status register.

Write <register> <value>

This command writes a value to the host register. Refer to the appropriate documentation regarding the host register interface on the CM-1 and CM-2. The architecture is significantly different.

<register> =

(for CM-1, the DSP5303 host register interface.)

icr: write the icr register.
cvr: write the cvr register.
isr: write the isr register.
ivr: write the ivr register.
drh: write the drh (rxh) register
drm: write the drm (rxm) register
drl: write the drl (rxl) register.

(for CM-2, the CS181xx host register interface)

msg: write the msg register (writes four bytes)
data: write the data register (writes four bytes)
msga: write the value to the message A register.
msgb: write the value to the message B register.
msgc: write the value to the message C register.
msgd: write the value to the message D register.
dataa: write the value to the data A register.
datab: write the value to the data B register.
datac: write the value to the data C register.
datad: write the value to the data D register.

<value> – a hexadecimal with a “0x” prefix. The individual registers are byte wide, a hexadecimal in the <value> parameter greater than a byte will only use the least significant byte.

Example: **write msgc 0xb3**

Please note that writing certain registers may trigger CobraNet events. Please refer to the CobraNet Programmer’s Manual for more information regarding these registers and the other host interface registers.

Peekev <target>

This command will return the value at the given address location for the EV-2 data memory. Please see the earlier discussion of the EV-2 memory map.

<target> = a valid address in hex format. The address is limited to two bytes.

Pokeev <target> <value>

This command will set the given address location in the EV-2 data memory to the given <value>. Please see the earlier discussion of the EV-2 memory map.

<target> – a valid address in hex format. The address is limited to two bytes.

<value> – a byte hexadecimal with a “0x” prefix.

Appendix F: CD-ROM Contents.

- EV-2 Manual (PDF)
- EV-2 Schematics (PDF)
- CobraNet Technology Datasheet (PDF)
- CM-1 Product Brief (PDF)
- CM-1 FW, Product Brief (PDF)
- CM-2 Product Brief (PDF)
- CS181xx Product Brief
- CM-1 and CM-2 Generic Firmware
- CobraCAD
- CobraNet Discovery
- EV-2 8031 Source Code
- EV-2 Xilinx VHDL Code
- CNEval.EXE
- CNEval.EXE Source Code

Contacting Cirrus Logic, Inc.

For further information on CobraNet™ products, contact: the Commercial Audio Products Division of Cirrus Logic, Inc.
2500 55th St. Suite 210 Boulder, CO 80301 (303) 245-5500

sales@peakaudio.com
www.cirrus.com

Copyright © 2001-04 Cirrus Logic, Inc. All rights reserved. CobraNet and Peak Audio are trademarks of Cirrus Logic, Inc.
