*Digital Audio Networking Processor*

# CobraNet™

## Silicon Series

CS18100x, CS18101x, CS18102x, *and* CM-2
CS49610x, CS49611x, *and* CS49612x

# Hardware User's Manual

**Version 2.3**

*Preliminary Product Information*

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

©Copyright 2005 Cirrus Logic, Inc.

CIRRUS LOGIC®
http://www.cirrus.com

JUN '05
DS651UM23

# Table of Contents

List of Figures....4
1.0 .Introduction....5
2.0 Features....6
2.1 CobraNet....6
2.2 CobraNet Interface....6
2.3 Host Interface....7
2.4 Asynchronous Serial Interface....7
2.5 Synchronous Serial Audio Interface....7
2.6 Audio Clock Interface....7
2.7 Audio Routing and Processing....7
3.0 Hardware....8
4.0 Pinout and Signal Descriptions....9
4.1 CS1810xx & CS4961xx Package Pinouts....10
4.1.1 CS1810xx/CS4961xx Pinout....10
4.1.2 CM-2 Connector Pinout....11
4.2 Signal Descriptions....12
4.2.1 Host Port Signals....12
4.2.2 Asynchronous Serial Port (UART Bridge) Signals....12
4.2.3 Synchronous Serial (Audio) Signals....13
4.2.4 Audio Clock Signals....13
4.2.5 Miscellaneous Signals....14
4.2.6 Power and Ground Signals....14
4.2.7 System Signals....15
4.3 Characteristics and Specifications....16
4.3.1 Absolute Maximum Ratings....16
4.3.2 Recommended Operating Conditions....16
4.3.3 Digital DC Characteristics....16
4.3.4 Power Supply Characteristics....16
5.0 Synchronization....17
5.1 Synchronization Modes....17
5.1.1 Internal Mode....18
5.1.2 External Word Clock Mode....18
5.1.3 External Master Clock Mode....18
6.0 Digital Audio Interface....19
6.1 Digital Audio Interface Timing....20
6.1.1 Normal Mode Data Timing....21
6.1.2 $I^2S$ Mode Data Timing....21
6.1.3 Standard Mode Data Timing....22
7.0 Host Management Interface (HMI)....23
7.1 Hardware....23
7.4 Protocol and Messages....28
7.4.1 Messages....28
7.4.1.1. Translate Address....29
7.4.1.2. Interrupt Acknowledge....29
7.4.1.3. Goto Packet....29
7.4.1.4. Goto Translation....29
7.4.1.5. Packet Received....30
7.4.1.6. Packet Transmit....30
7.4.1.7. Goto Counters....30
7.4.2 Status....31


©Copyright 2005 Cirrus Logic, Inc.

7.4.3 Data ..... 32
7.4.3.1. Region length ..... 32
7.4.3.2. Writable Region ..... 32
7.4.3.3. Translation Complete ..... 32
7.4.3.4. Packet Transmission Complete ..... 32
7.4.3.5. Received Packet Available ..... 32
7.4.3.6. Message Togglebit ..... 32
8.0 HMI Reference Code ..... 33
8.1 HMI Definitions ..... 33
8.2 HMI Access Code ..... 34
8.3 CM-1, CM-2 Auto-detection ..... 36
9.0 Mechanical Drawings and Schematics ..... 37
9.1 CM-2 Mechanical Drawings ..... 38
9.2 CM-2 Schematics ..... 44
9.3 CS1810xx/CS4961xx Package ..... 51
9.4 Temperature Specifications ..... 52
10.0 Ordering Information ..... 53
10.1 Device Part Numbers ..... 53
10.2 Device Part Numbering Scheme ..... 53


©Copyright 2005 Cirrus Logic, Inc.

# List of Figures

Figure 1. CobraNet Data Services ..... 5
Figure 2. CobraNet Interface Hardware Block Diagram ..... 8
Figure 3. Audio Clock Sub-system ..... 17
Figure 4. Channel Structure for Synchronous Serial Audio at 64FS (One Sample Period) - CS18100x/CS49610x & CS18101x/CS49611x ..... 19
Figure 5. Channel Structure for Synchronous Serial Audio at 128FS (One Sample Period) - CS18102x/CS49612x ..... 19
Figure 6. Timing Relationship between FS512_OUT, DAO1_SCLK and FS1 ..... 20
Figure 7. Serial Port Data Timing Overview ..... 20
Figure 8. Audio Data Timing Detail - Normal Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x ..... 21
Figure 9. Audio Data Timing Detail - Normal Mode, 128FS - CS18102x/CS49612x ..... 21
Figure 10. Audio Data Timing Detail - $I^2S$ Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x ..... 21
Figure 11. Audio Data Timing Detail - $I^2S$ Mode, 128FS - CS18102x & CS49612x ..... 21
Figure 12. Audio Data Timing Detail - Standard Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x ..... 22
Figure 13. Audio Data Timing Detail - Standard Mode, 128FS - CS18102x/CS49612x ..... 22
Figure 14. Host Port Read Cycle Timing - Motorola Mode ..... 25
Figure 15. Host Port Write Cycle Timing - Motorola Mode ..... 25
Figure 16. Parallal Control Port - Intel Mode Read Cycle ..... 27
Figure 17. Parallel Control Port - Intel Mode Write Cycle ..... 27
Figure 18. CM-2 Module Assembly Drawing, Top ..... 38
Figure 19. CM-2 Module Assembly Drawing, Bottom ..... 39
Figure 20. General PCB Dimensions ..... 40
Figure 21. Example Configuration, Side View ..... 41
Figure 22. Faceplate Dimensions ..... 42
Figure 23. Connector Detail ..... 43
Figure 24. CM-2 RevF Schematic Page 1 of 7 ..... 44
Figure 25. CM-2 RevF Schematic Page 2 of 7 ..... 45
Figure 26. CM-2 RevF Schematic Page 3 of 7 ..... 46
Figure 27. CM-2 RevF Schematic Page 4 of 7 ..... 47
Figure 28. CM-2 RevF Schematic Page 5 of 7 ..... 48
Figure 29. CM-2 RevF Schematic Page 6 of 7 ..... 49
Figure 30. CM-2 RevF Schematic Page 7 of 7 ..... 50
Figure 31. 144-Pin LQFP Package Drawing ..... 51
Figure 32. Device Part Numbering Explanation ..... 53


©Copyright 2005 Cirrus Logic, Inc.

# 1.0 Introduction

This document is intended to help hardware designers integrate the CobraNet™ interface into an audio system design. It covers the CS18100x, CS18101x, CS18102x, CS49610x, CS49611x, and CS49612x members of the CobraNet™ Silicon Series of devices, where "x" is the ROM version (ROM ID). This document also describes the CM-2 module with schematics, mechanical drawings, etc.

CobraNet is a combination of hardware (the CobraNet interface), network protocol, and firmware. CobraNet operates on a switched Ethernet network and provides the following additional communications services.

- Isochronous (Audio) Data Transport
- Sample Clock Distribution
- Control and Monitoring Data Transport

The CobraNet interface performs synchronous-to-isochronous and isochronous-to-synchronous conversions as well as the data formatting required for transporting real-time digital audio over the network.

The CobraNet interface has provisions for carrying and utilizing control and monitoring data such as *Simple Network Management Protocol* (SNMP) through the same network connection as the audio. Standard data transport capabilities of Ethernet are shown here as unregulated traffic. Since CobraNet is Ethernet based, in most cases, data communications and CobraNet applications can coexist on the same physical network. Figure 1 illustrates the different data services available through the CobraNet system.

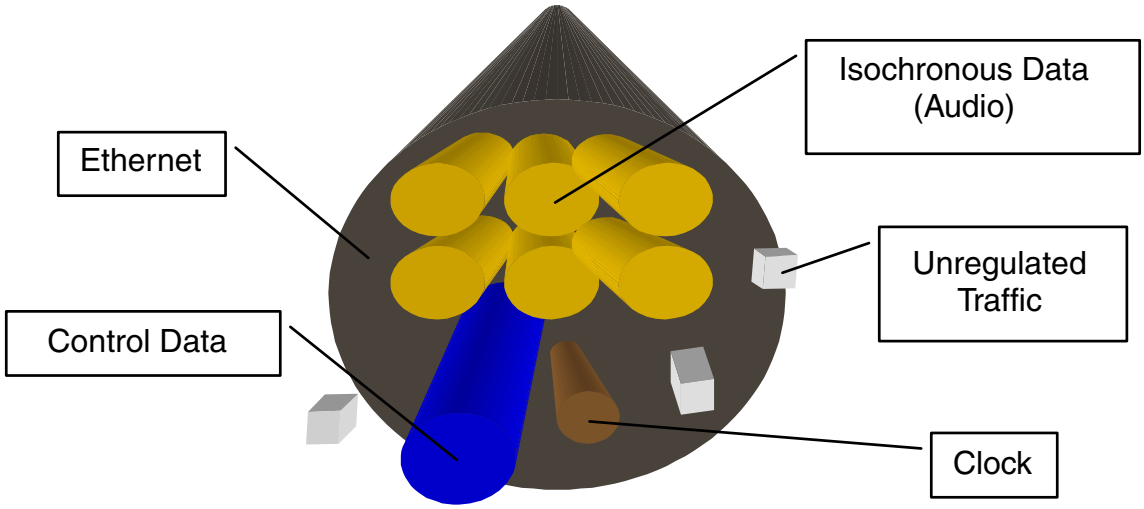



**Figure 1. CobraNet Data Services**


©Copyright 2005 Cirrus Logic, Inc.

# 2.0 Features

## 2.1 CobraNet

- Real-time Digital Audio Distribution via Ethernet
- No Overall Limit on Network Channel Capacity
- Fully IEEE 802.3 Ethernet Standards Compliant
- Fiber optic and gigabit Ethernet variants are fully supported.
- Ethernet infrastructure can be used simultaneously for audio and data communications.
- Free *CobraCAD*™ Audio Network Design Tool
- High-quality Audio Sample Clock Delivery Over Ethernet
- Bit-transparent 16-, 20-, and 24-bit Audio Transport
- Professional 48-kHz and 96-kHz sample rate
- Select Latency as Low as 1.33ms
- Flexible Many-to-many Network Audio Routing Capabilities
- Reduced-cost, Improved-performance, Convergent Audio Distribution Infrastructure

## 2.2 CobraNet Interface

- 120 MIPS Customer-configurable Audio DSP
- Auto-negotiating 100Mbit Full-duplex Ethernet Connections
- Up to 32-channel Audio I/O Capability
- Implements CobraNet Protocol for real-time transport of audio over Ethernet.
- Local Management via 8-bit Parallel Host Port
- UDP/IP Network Stack with Dynamic IP Address Assignment via BOOTP or RARP
- Remote Management via Simple Network Management Protocol (SNMP)
- Economical Three-chip Solution
- Available Module form factor allows for flexible integration into audio products.
- Non-volatile Storage of Configuration Parameters
- Safely Upgrade Firmware Over Ethernet Connection
- LED Indicators for Ethernet Link, Activity, Port Selection, and Conductor Status
- Watchdog Timer Output for System Integrity Assurance
- Comprehensive Power-on Self-test (POST)
- Error and Fault Reporting and Logging Mechanisms


©Copyright 2005 Cirrus Logic, Inc.

## 2.3 Host Interface

- 8-bit Data, 4-bit Address
- Virtual 24-bit Addressing with 32-bit Data
- *Polled*, *Interrupt*, and *DMA* Modes of Operation
- Configure and Monitor CobraNet Interface
- Transmit or Receive Ethernet Packets at *Near-100-Mbit* Wire Speed

## 2.4 Asynchronous Serial Interface

- Full-duplex Capable
- 8-bit Data Format
- Supports all Standard Baud Rates

## 2.5 Synchronous Serial Audio Interface

- Up to Four Bi-directional Interfaces Supporting up to 32 Channels of Audio I/O
- 64FS (3.072 MHz) Bit Rate for CS18100x/CS49610x and CS18101x/CS49611x
- 128FS (6.144 MHz) Bit Rate for CS18102x/CS49612x
- Accommodates Many Synchronous Serial Formats Including $I^2S$
- 32-bit Data Resolution on All Audio I/O

## 2.6 Audio Clock Interface

- 5 Host Audio-clocking Modes for Maximum Flexibility in Digital Audio Interface Design
- Low-jitter Master Audio Clock Oscillator (24.576 MHz)
- Synchronize to Supplied Master and/or Sample Clock
- Sophisticated jitter attenuation assures network perturbations do not affect audio performance.

## 2.7 Audio Routing and Processing

- Single-channel Granularity in Routing From Synchronous Serial Audio Interface to CobraNet Network
- Two levels of inward audio routing affords flexibility in audio I/O interface design in the host system.
- Local Audio Loopback and Output Duplication Capability
- Peak-read Audio Metering with Ballistics


©Copyright 2005 Cirrus Logic, Inc.

# 3.0 Hardware

Figure 2 shows a high-level view of the CobraNet CM-2 interface hardware architecture.

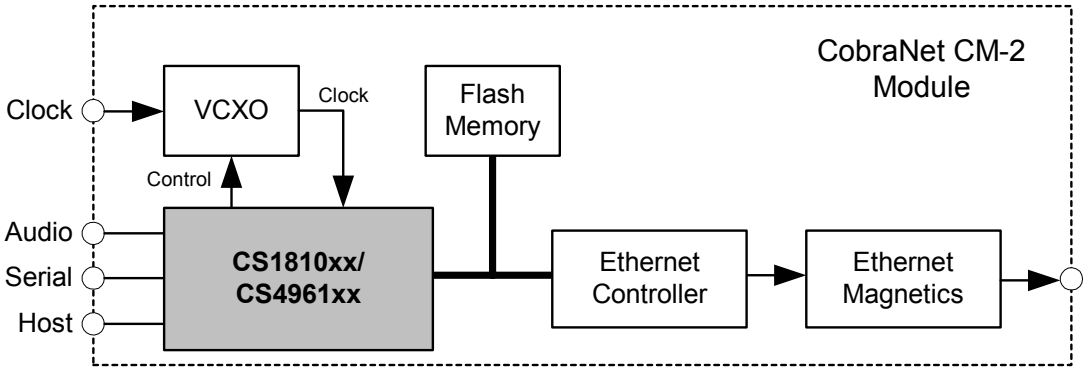



**Figure 2. CobraNet Interface Hardware Block Diagram**

Flash memory holds the CobraNet firmware and management interface variable settings.

The CS1810xx or CS4961xx network processor is the heart of the CobraNet interface. It implements the network protocol stacks and performs the synchronous-to-isochronous and isochronous-to-synchronous conversions. The network processor has a role in sample clock regeneration and performs all interactions with the host system.

The sample clock is generated by a voltage-controlled crystal oscillator (VCXO) controlled by the network processor. The VCXO frequency is carefully adjusted to achieve lock with the network clock.

The Ethernet controller is a standard interface chip that implements the 100-Mbit Fast Ethernet standard. As per Ethernet requirements the interface is transformer isolated.


©Copyright 2005 Cirrus Logic, Inc.

# 4.0 Pinout and Signal Descriptions

This section details the chip pinout and signal interfaces for each module and is divided as follows:


- "CS1810xx & CS4961xx Package Pinouts" on page 10
- "Host Port Signals" on page 12
- "Asynchronous Serial Port (UART Bridge) Signals" on page 12
- "Synchronous Serial (Audio) Signals" on page 13
- "Audio Clock Signals" on page 13
- "Miscellaneous Signals" on page 14
- "Power and Ground Signals" on page 14
- "System Signals" on page 15



©Copyright 2005 Cirrus Logic, Inc.

# 4.1 CS1810xx & CS4961xx Package Pinouts

## *4.1.1 CS1810xx/CS4961xx Pinout*

Table 1 lists the pinout for the 144-pin LQFP CS1810xx/CS4961xx device. The interfaces for these signals are expanded in the following sections.

**Table 1. CS1810xx/CS4961xx Pin Assignments**

| Pin # | Pin Name | Pin # | Pin Name | Pin # | Pin Name | Pin # | Pin Name |
|---|---|---|---|---|---|---|---|
| 1 | VCXO_CTRL | 37 | DATA1 | 73 | VDDIO | 109 | HADDR1 |
| 2 | MCLK_SEL | 38 | $\overline{WE}$ | 74 | ADDR10 | 110 | HADDR0 |
| 3 | DBDA | 39 | DATA0 | 75 | ADDR14 | 111 | HDATA7 |
| 4 | DBCK | 40 | DATA15 | 76 | GND | 112 | HDATA6 |
| 5 | NC | 41 | DATA14 | 77 | ADDR13 | 113 | VDDIO |
| 6 | NC | 42 | DATA13 | 78 | NC | 114 | HDATA5 |
| 7 | NC | 43 | DATA12 | 79 | NC | 115 | HDATA4 |
| 8 | DAO_MCLK | 44 | VDDIO | 80 | NC | 116 | GND |
| 9 | TEST | 45 | DATA11 | 81 | NC | 117 | HDATA3 |
| 10 | VDDD | 46 | DATA10 | 82 | ADDR15 | 118 | HDATA2 |
| 11 | HS3 | 47 | GND | 83 | VDDD | 119 | VDDD |
| 12 | NC | 48 | DATA9 | 84 | ADDR16 | 120 | HDATA1 |
| 13 | GND | 49 | DATA8 | 85 | ADDR17 | 121 | HDATA0 |
| 14 | DAO2_LRCLK | 50 | NC | 86 | GND | 122 | GND |
| 15 | DAO1_DATA3 | 51 | NC | 87 | ADDR18 | 123 | XTAL_OUT |
| 16 | DAO1_DATA2/HS2 | 52 | NC | 88 | ADDR19 | 124 | XTO |
| 17 | DAO1_DATA1/HS1 | 53 | NC | 89 | $\overline{OE}$ | 125 | XTI |
| 18 | VDDIO | 54 | VDDD | 90 | $\overline{CS1}$ | 126 | GND_a |
| 19 | DAO1_DATA0/HS0 | 55 | ADDR12 | 91 | VDDIO | 127 | FILT2 |
| 20 | DAO1_SCLK | 56 | ADDR11 | 92 | $\overline{MUTE}$ | 128 | FILT1 |
| 21 | GND | 57 | GND | 93 | $\overline{HRESET}$ | 129 | VDDA |
| 22 | DAO1_LRCLK | 58 | ADDR9 | 94 | GND | 130 | VDDD |
| 23 | UART_TX_OE | 59 | ADDR8 | 95 | WATCHDOG | 131 | DAI1_DATA3 |
| 24 | VDDD | 60 | VDDIO | 96 | IOWAIT | 132 | DAI1_DATA2 |
| 25 | UART_TXD | 61 | ADDR7 | 97 | REFCLK_IN | 133 | GND |
| 26 | UART_RXD | 62 | ADDR6 | 98 | VDDD | 134 | DAI1_DATA1 |
| 27 | GND | 63 | GND | 99 | GPIO0 | 135 | DAI1_DATA0 |
| 28 | NC | 64 | ADDR5 | 100 | GPIO1 | 136 | VDDIO |
| 29 | DATA7 | 65 | $\overline{CS2}$ | 101 | GND | 137 | DAI1_SCLK |
| 30 | DATA6 | 66 | VDDD | 102 | $\overline{HACK}$ | 138 | DAI1_LRCLK |
| 31 | DATA5 | 67 | ADDR4 | 103 | $\overline{HDS}$ | 139 | GND |
| 32 | DATA4 | 68 | ADDR3 | 104 | $\overline{HEN}$ | 140 | $\overline{HREQ}$ |
| 33 | VDDIO | 69 | GND | 105 | HADDR3 | 141 | NC |
| 34 | DATA3 | 70 | ADDR2 | 106 | HADDR2 | 142 | NC |
| 35 | DATA2 | 71 | ADDR1 | 107 | HR/$\overline{W}$ | 143 | IRQ1 |
| 36 | GND | 72 | ADDR0 | 108 | GPIO2 | 144 | IRQ2 |


©Copyright 2005 Cirrus Logic, Inc.

### 4.1.2 CM-2 Connector Pinout

Table 1 lists the pinout for the four pinout connectors on the CM-2 board (J1-J4). The interfaces for these signals are expanded following the table.

**Table 2. CM-2 Pin Assignments**

| Conn. | Pin # | Pin Name |
|---|---|---|
| J1/J2 | A1 | UART_RXD |
| J1/J2 | A2 | UART_TX_OE |
| J1/J2 | A3 | $\overline{\text{HACK}}$ |
| J1/J2 | A4 | HR/$\overline{\text{W}}$ |
| J1/J2 | A5 | $\overline{\text{HDS}}$ |
| J1/J2 | A6 | $\overline{\text{HREQ}}$ |
| J1/J2 | A7 | $\overline{\text{HEN}}$ |
| J1/J2 | A8 | HADDR0 |
| J1/J2 | A9 | HADDR1 |
| J1/J2 | A10 | HADDR2 |
| J1/J2 | A11 | HDATA0 |
| J1/J2 | A12 | HDATA1 |
| J1/J2 | A13 | HDATA2 |
| J1/J2 | A14 | HDATA3 |
| J1/J2 | A15 | HDATA4 |
| J1/J2 | A16 | HDATA5 |
| J1/J2 | A17 | HDATA6 |
| J1/J2 | A18 | $\overline{\text{HRESET}}$ |
| J1/J2 | A19 | HDATA7 |
| J1/J2 | A20 | HADDR3 |
| J1/J2 | B1 | UART_TXD |
| J1/J2 | B2 | GND |
| J1/J2 | B3 | VCC_+3.3V |
| J1/J2 | B4 | GND |
| J1/J2 | B5 | VCC_+3.3V |
| J1/J2 | B6 | GND |
| J1/J2 | B7 | VCC_+3.3V |
| J1/J2 | B8 | GND |
| J1/J2 | B9 | VCC_+3.3V |
| J1/J2 | B10 | GND |
| J1/J2 | B11 | VCC_+3.3V |
| J1/J2 | B12 | GND |
| J1/J2 | B13 | VCC_+3.3V |
| J1/J2 | B14 | GND |
| J1/J2 | B15 | VCC_+3.3V |
| J1/J2 | B16 | GND |
| J1/J2 | B17 | VCC_+3.3V |
| J1/J2 | B18 | RSVD1 |
| J1/J2 | B19 | GND |
| J1/J2 | B20 | VCC_+3.3V |
| J3/J4 | A1 | RSVD2 |
| J3/J4 | A2 | $\overline{\text{MUTE}}$ |
| J3/J4 | A3 | FS1 |
| J3/J4 | A4 | MCLK_OUT |
| J3/J4 | A5 | MCLK_IN |
| J3/J4 | A6 | REFCLK_IN |
| J3/J4 | A7 | DAO1_SCLK/DAI1_SCLK |
| J3/J4 | A8 | DAO1_DATA0 |
| J3/J4 | A9 | DAO1_DATA1 |
| J3/J4 | A10 | DAO1_DATA2 |
| J3/J4 | A11 | DAO1_DATA3 |
| J3/J4 | A12 | DAI1_DATA0 |
| J3/J4 | A13 | DAI1_DATA1 |
| J3/J4 | A14 | DAI1_DATA2 |
| J3/J4 | A15 | DAI1_DATA3 |
| J3/J4 | A16 | RSVD3 |
| J3/J4 | A17 | WATCHDOG |
| J3/J4 | A18 | RSVD4 |
| J3/J4 | A19 | AUX_POWER2 |
| J3/J4 | A20 | AUX_POWER0 |
| J3/J4 | B1 | GND |
| J3/J4 | B2 | VCC_+3.3V |
| J3/J4 | B3 | GND |
| J3/J4 | B4 | VCC_+3.3V |
| J3/J4 | B5 | GND |
| J3/J4 | B6 | VCC_+3.3V |
| J3/J4 | B7 | GND |
| J3/J4 | B8 | VCC_+3.3V |
| J3/J4 | B9 | GND |
| J3/J4 | B10 | VCC_+3.3V |
| J3/J4 | B11 | GND |
| J3/J4 | B12 | VCC_+3.3V |
| J3/J4 | B13 | GND |
| J3/J4 | B14 | VCC_+3.3V |
| J3/J4 | B15 | GND |
| J3/J4 | B16 | GND |
| J3/J4 | B17 | VCC_+5V |
| J3/J4 | B18 | VCC_+5V |
| J3/J4 | B19 | AUX_POWER3 |
| J3/J4 | B20 | AUX_POWER1 |


©Copyright 2005 Cirrus Logic, Inc.

## 4.2 Signal Descriptions

### *4.2.1 Host Port Signals*

The host port is used to manage and monitor the CobraNet interface. Electrical operation and protocol is detailed in the "Host Management Interface (HMI)" on page 23 of this Manual.

The host port can operate in two modes in order to accomodate Motorola® or Intel® style interfaces. The default mode is Motorola. Intel mode is set via a firmware modification.

**Table 2-1: Host Port Signals**

| Signal | Description | Direction | CM-2 Pin # | CS1810xx/ CS4961xx Pin # | Notes |
|---|---|---|---|---|---|
| HDATA[7:0] | Host Data | In/Out | J1:A19, A[17:11] | 111, 112, 114, 115, 117, 118, 102, 121 | Host port data. |
| HADDR[3:0] | Host Address | In | J1:A20, A[10:8] | 105, 106, 109,110 | Host port address. |
| HR$\overline{W}$ | Host Direction | In | J1:A4 | 107 | Host port transfer direction (Motorola mode). |
| $\overline{HRD}$ | Host Read | In | J1:A4 | 107 | Host Read (Intel mode). |
| $\overline{HREQ}$ | Host Request | Out | J1:A6 | 140 | Host port data request. |
| $\overline{HACK}$ | Host Alert | Out | J1:A3 | 102 | Host port interrupt request. |
| $\overline{HDS}$ | Host Strobe | In | J1:A5 | 103 | Host port strobe (Motorola mode). |
| $\overline{HWR}$ | Host Write | In | J1:A5 | 103 | Host Write (Intel mode). |
| $\overline{HEN}$ | Host Enable | In | J1:A7 | 104 | Host Port Enable. |
| $\overline{HCS}$ | Select | In | J1:A7 | 104 | Select (Intel mode). |

### *4.2.2 Asynchronous Serial Port (UART Bridge) Signals*

Level-shifting drive circuits are typically required between these signals and any external connections.

| Signal | Description | Direction | CM-2 Pin # | CS1810xx/ CS4961xx Pin # | Notes |
|---|---|---|---|---|---|
| UART_RXD | Asynchronous Serial Receive Data | In | J1:A1 | 26 | Pull-up to VCC if unused. |
| UART_TXD | Asynchronous Serial Transmit Data | Out | J1:B1 | 25 | |
| UART_TX_OE | Transmit Drive Enable | Out | J1:A2 | 23 | Enable transmit (active high) drive for two wire multi-drop interface. |


©Copyright 2005 Cirrus Logic, Inc.

### 4.2.3 Synchronous Serial (Audio) Signals

The synchronous serial interfaces are used to bring digital audio into and out of the system. Typically the synchronous serial is wired to ADCs and/or DACs. Detailed timing and format is described in "Digital Audio Interface" on page 19.

| Signal | Description | Direction | CM-2 Pin # | CS1810xx/ CS4961xx Pin # | Notes |
|---|---|---|---|---|---|
| DAO1_SCLK | Audio Bit Clock | Out | J3:A7 | 20 | Synchronous serial bit clock.<br>64 FS for CS18100x & CS49610x (2x1 channel)<br>64 FS for CS18101x & CS49611x (2x4 channels)<br>128 FS for CS18102x & CS49612x (4x4 channels)<br>Typically tied to DAI1_SCLK. |
| DAO1_DATA[3:0] | Audio Output Data | Out | J3:A18, B18 | 15-17, 19 | Output synchronous serial audio data<br>DAO1_DATA[3:1] not used for CS18100x & CS49610x. |
| DAI1_DATA[3:0] | Audio Input Data | In | J3: A[15:12] | 131, 132, 134, 135 | Input synchronous serial audio data<br>DAI1_DATA[3:1] not used for CS18100x & CS49610x. |
| DAI1_SCLK | Audio Bit Clock | In | J4:A7 | 137 | Should be tied to DAO1_SCLK.<br>Synchronous serial bit clock. |

### 4.2.4 Audio Clock Signals

See "Synchronization" on page 17 for an overview of synchronization modes and issues.

| Signal | Description | Direction | CM-2 Pin # | CS1810xx/ CS4961xx Pin # | Notes |
|---|---|---|---|---|---|
| DAI1_LRCLK | Sample clock input | In | | 138 | Should be tied to DAO1_LRCLK for all devices. |
| DAO1_LRCLK (FS1) | Sample clock output | Out | J3:A3 | 22 | FS1 (word clock) for CS18100x/CS49610x and CS18101x/CS49611x. |
| DAO2_LRCLK (FS1) | Sample clock output | Out | J3:A3 | 14 | FS1 (word clock) for CS18102x & CS49612x. |
| REFCLK_IN | Reference clock | In | J3:A6 | 97 | Clock input for synchronizing network to an external clock source, for redundancy control and synchronization of FS divider chain to external source. See "Synchronization" on page 17 for more detail. |
| MCLK_IN | Master audio clock input | In | J3:A5 | 8* | For systems featuring multiple CobraNet interfaces operating off a common master clock. See "Synchronization" on page 17 for more detail. |
| MCLK_OUT | Master audio clock output | Out | J3:A4 | 8* | Low jitter 24.576 MHz master audio clock. |

*An external multiplexor controlled by this pin is required for full MCLK_IN and MCLK out implementation.


©Copyright 2005 Cirrus Logic, Inc.

### 4.2.5 Miscellaneous Signals

| Signal | Description | Direction | CM-2 Pin # | CS1810xx/ CS4961xx Pin # | Notes |
|---|---|---|---|---|---|
| $\overline{\text{HRESET}}$ | Reset | In | J1:A18 | 93 | System reset (active low). **10 ns** max rise time. **1 ms** min assertion time. |
| WATCHDOG | Watch Dog | Out | J3:A17 | 95 | Toggles at 750 Hz nominal rate to indicate proper operation. Period duration in excess of 200 ms indicates hardware or software failure has occurred and the interface should be reset. Note that improper operation can also be indicated by short pulses (<100 ns). |
| $\overline{\text{MUTE}}$ | Interface Ready | Out | J3:A2 | 92 | Asserts (active low) during initialization and when a fault is detected or connection to the network is lost. |
| NC | No Connect | - | - | 28, 50-53, 78-81, 141, 142 | |

### 4.2.6 Power and Ground Signals

| Signal | Description | CM-2 Pin # | CS1810xx/CS4961xx Pin # | Specification |
|---|---|---|---|---|
| VCC_+3V | System Digital +3.3 v | J1:B20, B17, B15, B13, B11, B9, B7, B5, B3<br><br>J3:B14, B12, B10, B8, B6, B4, B2 | N/A | 3.3 ± 0.3v, 500 mA Typ., 750 mA Max. |
| VCC_+5V | | J3;B[18:17] | N/A | Backwards Compatibility |
| VDDD | | N/A | 10, 24, 54, 66, 83, 98, 119, 130 | +1.8 V @ 500mA Typ. for Core Logic |
| VDDIO | | N/A | 18, 33, 44, 60, 73, 91, 113, 136 | +3.3 V @ 120mA Typ. for I/O Logic |
| VDDA | | N/A | 129 | Filtered +1.8 V @ 10mA Typ. |
| AUX_POWER [3-0] | | J3:B[20:19], A[20:19] | N/A | |
| GND | Digital Ground | J1:B19, B16, B14, B12, B10, B8, B6, B4, B2<br><br>J3:B16, B15, B13, B11, B9, B7, B5, B3, B1 | 13, 21, 27, 36, 47, 57, 63, 69, 76, 86, 94, 101, 116, 122, 126, 133, 139 | |


©Copyright 2005 Cirrus Logic, Inc.

### 4.2.7 System Signals

Use these CS1810xx/CS4961xx signals stricktly in the manner described in *CM-2 Schematics* (Section 9.2 on page 44). Each signal is briefly described below.

| Signal | Description | CS1810xx/CS4961xx Pin # |
|---|---|---|
| VCXO_CTRL | A Delta-sigma DAC Output for Controlling the On-board VCXO | 1 |
| MCLK_SEL | Control Signal for Selecting MCLK Sources | 2 |
| DBDA, DBCK | I2C Debugger Interface | 3, 4 |
| TEST | Used for testing during manufacturing. Keep grounded for normal operation. | 9 |
| DATA[15:0] | Data Bus for Flash & Ethernet Controller(s) | 29-32, 34, 35, 37, 39-43, 45, 46, 48, 49 |
| ADDR[19:0] | Address Bus for Flash & Ethernet Controller(s) | 55, 56, 58, 59, 61, 62, 64, 67, 68, 70-72, 74, 75, 77, 82, 84, 85, 87, 88 |
| $\overline{WE}$ | Write Enable for Flash and Ethernet Controller(s) | 38 |
| $\overline{CS1}$ | Chip Select for Flash Memory Device | 90 |
| $\overline{CS2}$ | Chip Select for Ethernet Controller(s) | 65 |
| $\overline{OE}$ | Output Enable | 89 |
| IOWAIT | Wait State Signal from Ethernet Controller(s) | 96 |
| GPIO[2:0] | General-purpose I/O Signals | 99, 100, 108 |
| XTI | Reference Clock Input / Crystal Oscillator Input | 125 |
| XTO | Crystal Oscillator Output | 124 |
| XTAL_OUT | A Buffered Version of XTI | 123 |
| FILT2, FILT1 | PLL Loop Filter | 127, 128 |
| DAO_MCLK | MCLK Input | 8 |
| HS[3:0] | CS1810xx/CS4961xx Boot Mode Selection | 11, 16, 17, 19 |


©Copyright 2005 Cirrus Logic, Inc.

# 4.3 Characteristics and Specifications

## *4.3.1 Absolute Maximum Ratings*

| Parameter | | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| DC power supplies: | Core supply | VDD | –0.3 | 2.0 | V |
| | PLL supply | VDDA | –0.3 | 2.0 | V |
| | I/O supply | VDDIO | –0.3 | 5.0 | V |
| | \|VDDA – VDD\| | | - | 0.3 | V |
| Input current, any pin except supplies | | $I_{in}$ | - | +/- 10 | mA |
| Input voltage on FILT1, FILT2 | | $V_{filt}$ | | 2.0 | V |
| Input voltage on I/O pins | | $V_{inio}$ | - | 5.0 | V |
| Storage temperature | | $T_{stg}$ | –65 | 150 | °C |

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

## *4.3.2 Recommended Operating Conditions*

| Parameter | | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| DC power supplies: | Core supply | VDD | 1.71 | 1.8 | 1.89 | V |
| | PLL supply | VDDA | 1.71 | 1.8 | 1.89 | V |
| | I/O supply | VDDIO | 3.13 | 3.3 | 3.46 | V |
| | \|VDDA – VDD\| | | | | 0.3 | V |
| Ambient operating temperature | | $T_A$ | | - | | °C |
| | - CQ | | 0 | | + 70 | |
| | - DQ | | - 40 | | + 85 | |

## *4.3.3 Digital DC Characteristics*

(measurements performed under static conditions.)

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| High-level input voltage | $V_{IH}$ | 2.0 | - | - | V |
| Low-level input voltage, except XTI | $V_{IL}$ | - | - | 0.8 | V |
| Low-level input voltage, XTI | $V_{ILXTI}$ | - | - | 0.6 | V |
| Input Hysteresis | $V_{hys}$ | | 0.3 | | V |
| High-level output voltage at $I_O$ = –8.0 mA$_O$ = –16.0 mA | $V_{OH}$ | VDDIO * 0.9 | - | - | V |
| Low-level output voltage at $I_O$ = 8.0 mA$_O$ = –16.0 mA | $V_{OL}$ | - | - | VDDIO * 0.1 | V |
| Input leakage current (all pins without internal pull-up resistors except XTI) | $I_{IN}$ | - | - | 5 | µA |
| Input leakage current (pins with internal pull-up resistors, XTI) | $I_{IN-PU}$ | - | - | 50 | µA |

## *4.3.4 Power Supply Characteristics*

(measurements performed under operating conditions))

| Parameter | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Power supply current: | | | | | |
| Core and I/O operating: VDD | (Note 1) | - | 500 | - | mA |
| PLL operating: VDDA | | - | 10 | - | mA |
| With external memory and most ports operating: VDDIO | | - | 120 | - | mA |

NOTES:1. Dependent on application firmware and DSP clock speed.


©Copyright 2005 Cirrus Logic, Inc.

# 5.0 Synchronization

Figure 3 shows clock related circuits for the CS1810xx/CS4961xx and board design (CM-2). This circuitry allows the synchronization modes documented below to be achieved. Modes are distinguished by different settings of the multiplexors and software elements.

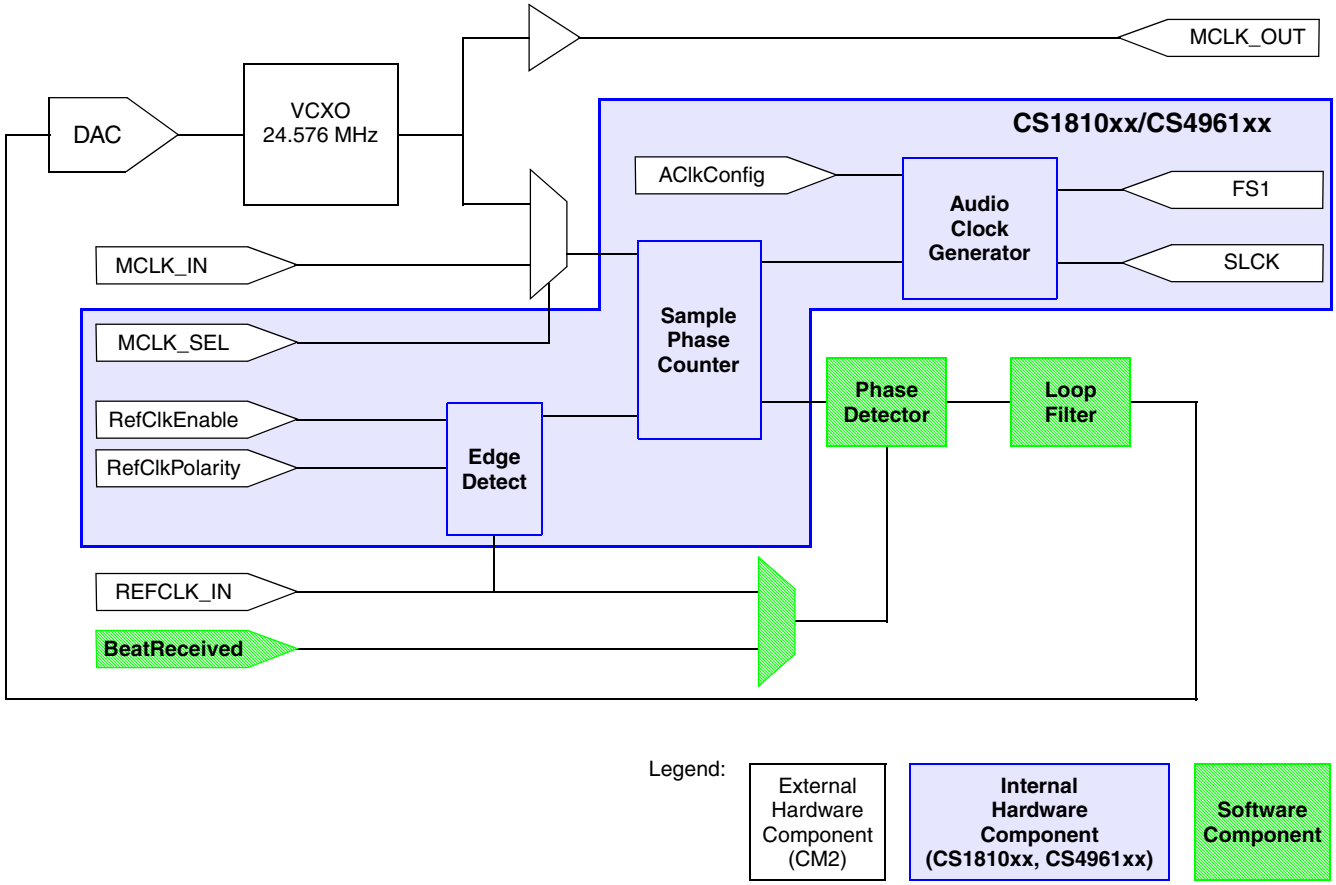



**Figure 3. Audio Clock Sub-system**

## 5.1 Synchronization Modes

Clock synchronization mode for conductor and performer roles is independently selectable via management interface variables *syncConductorClock* and *syncPerformerClock*. The role (conductor or performer) is determined by the network environment including the *conductor priority* setting of the device and the other devices on the network. It is possible to ensure you will never assume the conductor role by selecting a conductor priority of *zero*. However, it is not reasonable to assume that by setting a high conductor priority, you will always assume the conductor role. For more information, refer to CobraNet Programmer's Reference Manual.


©Copyright 2005 Cirrus Logic, Inc.

The following synchronization modes are further described below:


- "Internal Mode" on page 18
- "External Word Clock Mode" on page 18
- "External Master Clock Mode" on page 18


### 5.1.1 Internal Mode

All CobraNet clocks are derived from the onboard VCXO. The master clock generated by the VCXO is available to external circuits via the master clock output.

**Conductor**—The VCXO is "parked" according to the *syncClockTrim* setting.

**Performer**—The VCXO is "steered" to match the clock transmitted by the Conductor.

### 5.1.2 External Word Clock Mode

All CobraNet clocks are derived from the onboard VCXO. The VCXO is steered from an external clock supplied to the reference clock input. The clock supplied can be any integral division of the sample clock in the range of 750Hz to 48kHz.

External synchronization lock range: ±5 μs. This specification indicates drift or wander between the supplied clock and the generated network clock at the conductor. Absolute phase difference between the supplied reference clock and generated sample clock is dependant on network topology.

**Conductor**—This mode gives a means for synchronizing an entire CobraNet network to an external clock.

**Performer**—The interface disregards the fine timing information delivered over the network from the conductor. Coarse timing information from the conductor is still used; fine timing information is instead supplied by the reference clock. The external clock source must be synchronous with the network conductor. This mode is useful in installations where a house sync source is readily available.

### 5.1.3 External Master Clock Mode

The VCXO is disabled and MCLK_IN is used as the master clock for the node. This is a "hard" synchronization mode. The supplied clock is used directly by the CobraNet interface for all timing. This mode is primarily useful for devices with multiple CobraNet interfaces sharing a common master audio clock. The supplied clock must be 24.576 MHz. The supplied clock must have a ±37 ppm precision.

**Conductor**—The entire network is synchronized to the supplied master clock.

**Performer**—The node will initially lock to the network clock and will "jam sync" via the supplied master clock. The external clock source must be synchronous with the network conductor.


©Copyright 2005 Cirrus Logic, Inc.

# 6.0 Digital Audio Interface

The CS18101x/CS49611x, CS18102x/CS49612x, and CM-2 support four bi-directional synchronous serial interfaces. The CS18100x & CS49610x support one bi-directional synchronous serial interface. All interfaces operate in master mode with DAO1_SCLK as the bit clock and FS1 as the frame clock. A sample period worth of synchronous serial data includes two (or four) audio channels. CobraNet supports two synchronous serial bit rates: 48 Khz and 96 KHz. However, 96 kHz sample rate is not available when using CS18102x/CS49612x with 16X16 channels. Bit rate is selected by the *modeRateControl* variable. All synchronous serial interfaces operate from a common clock at the same bit rate.

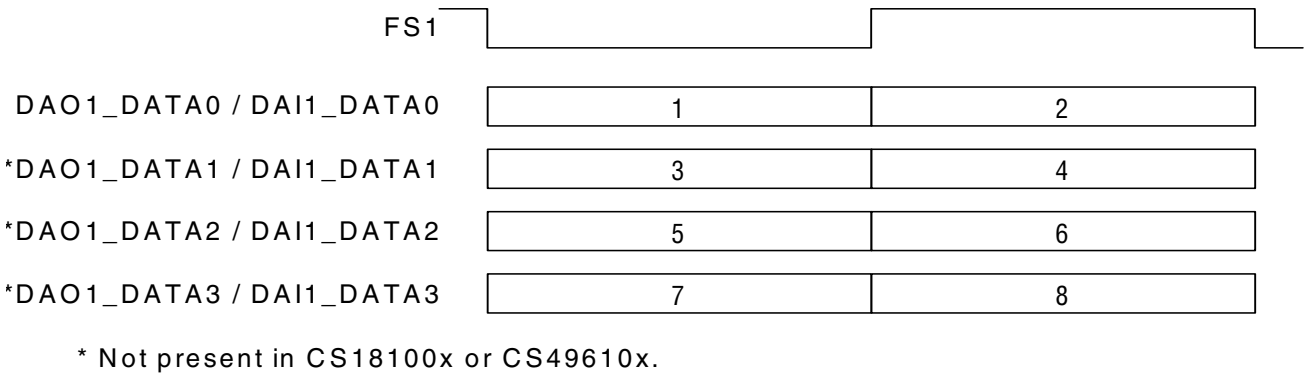



**Figure 4. Channel Structure for Synchronous Serial Audio at 64FS (One Sample Period) - CS18100x/CS49610x & CS18101x/CS49611x**

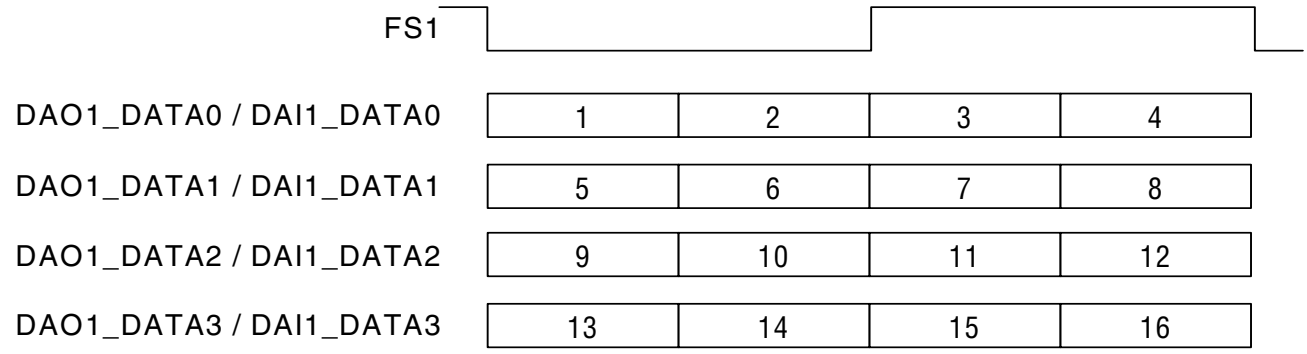



**Figure 5. Channel Structure for Synchronous Serial Audio at 128FS (One Sample Period) - CS18102x/CS49612x**

Default channel ordering is shown above. Note that the first channel always begins after the rising or falling edge of FS1 (depending on the mode).

DAI1_SCLK period depends on the sample rate selected. Up to 32 significant bits are received and buffered by the DSP for synchronous inputs. Up to 32 significant bits are transmitted by the DSP for synchronous outputs. Bit 31 is always the most significant (sign) bit. A 16-bit audio source must drive to bit periods 31-16 with audio data and bits 15-0 should be actively driven with either a dither signal or zeros. Cirrus Logic recommends driving unused LS bits to zero.


©Copyright 2005 Cirrus Logic, Inc.

Although data is always transmitted and received with a 32-bit resolution by the synchronous serial ports, the resolution of the data transferred to/from the Ethernet may be less. Incoming audio data is truncated to the selected resolution. Unused least significant bits on outgoing data is zero filled.

## 6.1 Digital Audio Interface Timing

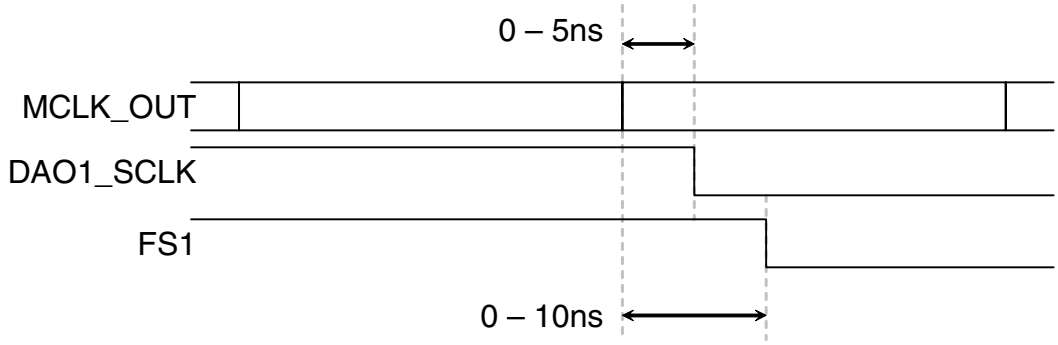



**Figure 6. Timing Relationship between FS512_OUT, DAO1_SCLK and FS1**

An DAO1_SCLK edge follows an MCLK_OUT edge by 0.0 to 5.0ns. An FS1 edge follows a MCLK_OUT edge by 0.0 to 10.0ns.

*Note: The DAO1_SCLK and FS1 might be synchronized with the either the falling edge or the rising edge of MCLK_OUT. Which edge is impossible to predict since it depends on power up timing.*

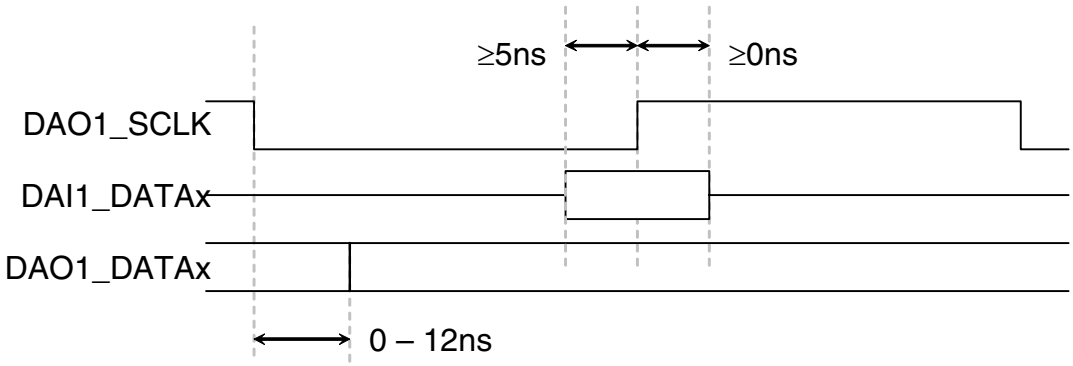



**Figure 7. Serial Port Data Timing Overview**

Setup times for DAI1_DATAx and FS1 are 5.0 ns with a hold time of 0.0 ns with respect to the DAI1_SCLK edge. Clock to output times for DAO1_DATAx is 0.0 to 12.0 ns from the edge of DAO1_SCLK.


©Copyright 2005 Cirrus Logic, Inc.

### 6.1.1 Normal Mode Data Timing

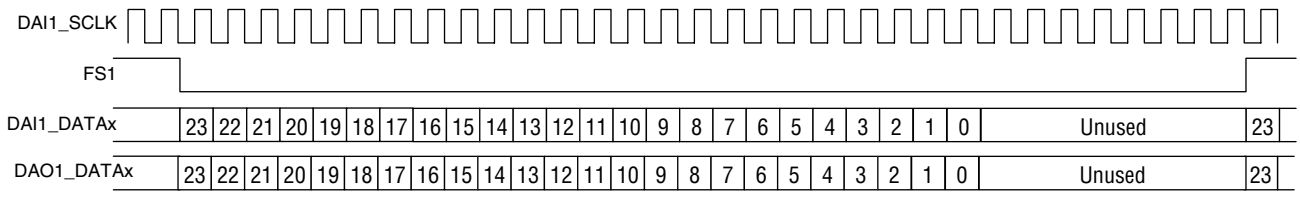



**Figure 8. Audio Data Timing Detail - Normal Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x**

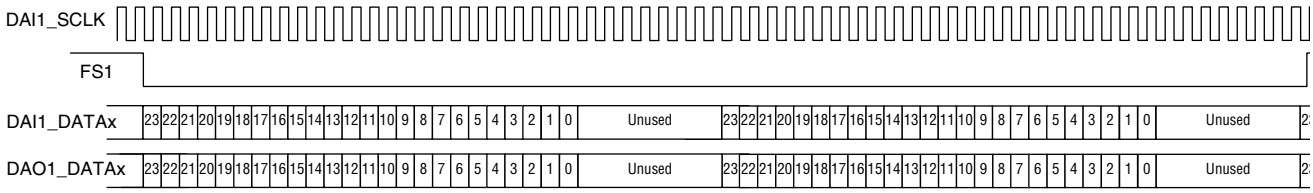



**Figure 9. Audio Data Timing Detail - Normal Mode, 128FS - CS18102x/CS49612x**

Each audio channel is comprised of 32 bits of data, regardless of audio sample size. The figure above shows 24-bit audio data.

The MSB is left justified and is aligned with FS1. Data is sampled on the rising edge of DAI_SCLK and data changes on the falling edge.

### 6.1.2 $I^2S$ Mode Data Timing

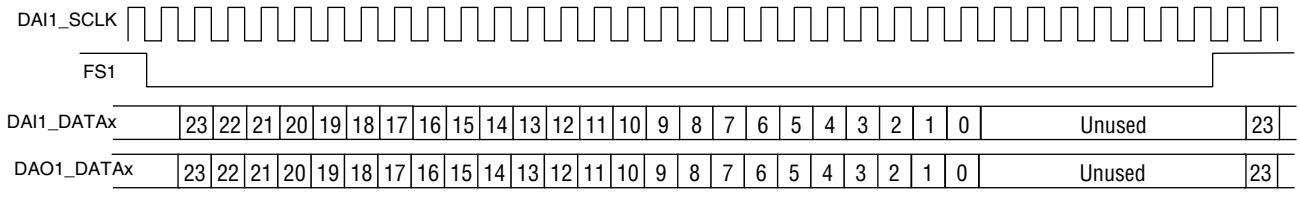



**Figure 10. Audio Data Timing Detail - $I^2S$ Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x**

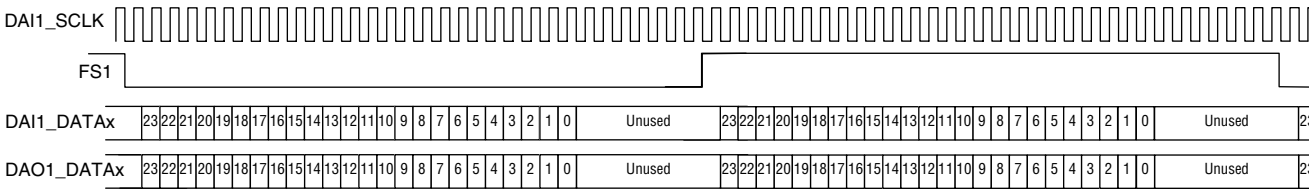



**Figure 11. Audio Data Timing Detail - $I^2S$ Mode, 128FS - CS18102x & CS49612x**

Each audio channel is comprised of 32 bits of data, regardless of audio sample size. The figure above shows 24-bit audio data.

The MSB is left justified and arrives one bit period following FS1. Data is sampled on the rising edge of DAI_SCLK and data changes on the falling edge.


©Copyright 2005 Cirrus Logic, Inc.

### *6.1.3 Standard Mode Data Timing*

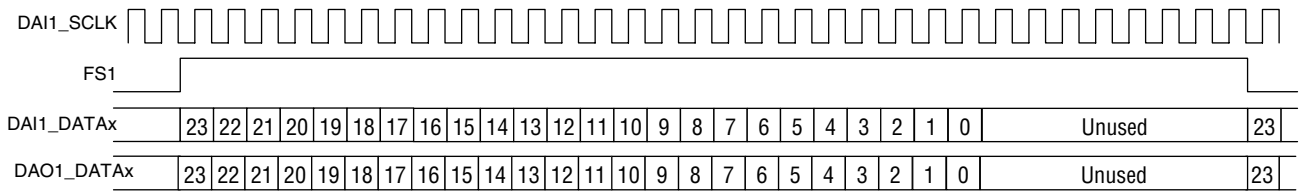



**Figure 12. Audio Data Timing Detail - Standard Mode, 64FS - CS18100x/CS49610x, CS18101x/CS49611x**

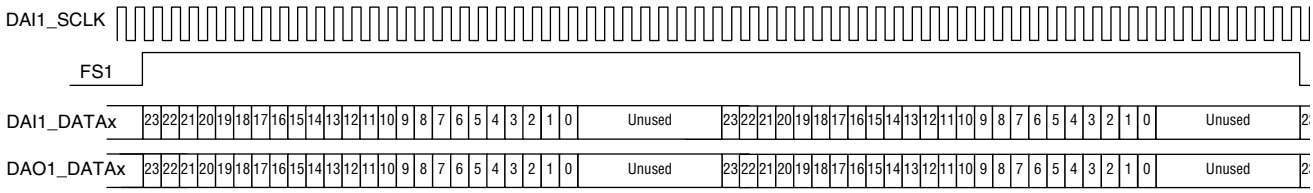



**Figure 13. Audio Data Timing Detail - Standard Mode, 128FS - CS18102x/CS49612x**

Each audio channel is comprised of 32 bits of data, regardless of audio sample size. The figure above shows 24-bit audio data.

The MSB is left justified and is aligned with FS1. Data is sampled on the rising edge of DAI_SCLK and data changes on the falling edge.


©Copyright 2005 Cirrus Logic, Inc.

# 7.0 Host Management Interface (HMI)

## 7.1 Hardware

The host port is 8 bits wide with 4 bits of addressing. Ten of the 16 addressable registers are implemented. The upper two registers can be used to configure and retrieve the status on the host port hardware. However, only the first 8 are essential for normal HMI communications. It is therefore feasible, in most applications, to utilize only the first 3 address bits and tie the most significant bit (A3) low.

Host port hardware supports Intel® *(little-endian)*, Motorola®, and Motorola multiplexed bus *(big-endian)* protocols. Standard CobraNet firmware configures the port in the Motorola, *big-endian* mode.

The host port memory map is shown in Table 3. Refer also to "HMI Definitions" on page 33 and "HMI Access Code" on page 34.

| Host Address | Register |
|---|---|
| 0 | Message A (MS) |
| 1 | Message B |
| 2 | Message C |
| 3 | Message D (LS) |
| 4 | Data A (MS) |
| 5 | Data B |
| 6 | Data C |
| 7 | Data D (LS) |
| 8 | Control |
| 9 | Status |

**Table 3. Host port memory map**

The message and data registers provide separate bi-directional data conduits between the host processor and the CS1810xx/CS4961xx. A 32-bit word of data is transferred to the CS1810xx/CS4961xx when the host writes the D message or data register after presumably previously writing the A, B, and C registers with valid data. Data is transferred from the CS1810xx/CS4961xx following a read of the D message or data register. Again, presumably the A, B, and C registers are read previously.

Two additional hardware signals are associated with the host port: $\overline{\text{HACK}}$ and $\overline{\text{HREQ}}$. Both are outputs to the host.

$\overline{\text{HACK}}$ may be wired to an interrupt request input on the host. $\overline{\text{HACK}}$ can be made to assert (logic 0) on specific events as specified by the *hackEnable MI* variable. $\overline{\text{HACK}}$ is deasserted (logic 1) by issuance of the *Acknowledge Interrupt* message (see *"Messages"* below).


©Copyright 2005 Cirrus Logic, Inc.

$\overline{\text{HREQ}}$ may be wired to a host interrupt or DMA request input. $\overline{\text{HREQ}}$ is used to signal the host that data is available (read case, logic 0) or space is available in the host port data channel (write case, logic 1).

The read and write case are distinguished by the HMI based on the preceding message. *Identify, Goto Translation* (read), *Goto Packet* (read) and *Goto Counters* cause $\overline{\text{HREQ}}$ to represent read status. *Goto Translation* (write) and *Goto Packet* (write) switch $\overline{\text{HREQ}}$ to write mode. All other commands have no effect on $\overline{\text{HREQ}}$ operation.

In general, the host can read from the CS1810xx/CS4961xx when $\overline{\text{HREQ}}$ is low and can write data to CS1810xx/CS4961xx when $\overline{\text{HREQ}}$ is high.

## 7.2 Host Port Timing - Motorola® Mode

($C_L$ = 20 pF)

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Address setup before $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low | $t_{mas}$ | 5 | - | ns |
| Address hold time after $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low | $t_{mah}$ | 5 | - | ns |
| **Read** | | | | |
| Delay between $\overline{\text{HDS}}$ then $\overline{\text{HEN}}$ low or $\overline{\text{HEN}}$ then $\overline{\text{HDS}}$ low | $t_{mcdr}$ | 0 | - | ns |
| Data valid after $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low with HR$\overline{\text{W}}$ high | $t_{mdd}$ | - | 19 | ns |
| $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low for read | $t_{mrpw}$ | 24 | - | ns |
| Data hold time after $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high after read | $t_{mdhr}$ | 8 | - | ns |
| Data high-Z after $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high after read | $t_{mdis}$ | - | 18 | ns |
| $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high to $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low for next read | $t_{mrd}$ | 30 | - | ns |
| $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high to $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low for next write | $t_{mrdtw}$ | 30 | - | ns |
| HR/$\overline{\text{W}}$ rising to $\overline{\text{HREQ}}$ falling | $t_{mrwirqh}$ | - | 12 | ns |
| **Write** | | | | |
| Delay between $\overline{\text{HDS}}$ then $\overline{\text{HEN}}$ low or $\overline{\text{HEN}}$ then $\overline{\text{HDS}}$ low | $t_{mcdw}$ | 0 | - | ns |
| Data setup before $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high | $t_{mdsu}$ | 8 | - | ns |
| $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low for write | $t_{mwpw}$ | 24 | - | ns |
| HR$\overline{\text{W}}$ setup before $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low | $t_{mrwsu}$ | 24 | - | ns |
| HR$\overline{\text{W}}$ hold time after $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high | $t_{mrwhld}$ | 8 | - | ns |
| Data hold after $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high | $t_{mdhw}$ | 8 | - | ns |
| $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high to $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low with HR$\overline{\text{W}}$ high for next read | $t_{mwtrd}$ | 30 | - | ns |
| $\overline{\text{HEN}}$ or $\overline{\text{HDS}}$ high to $\overline{\text{HEN}}$ and $\overline{\text{HDS}}$ low for next write | $t_{mwd}$ | 30 | - | ns |
| HR$\overline{\text{W}}$ rising to $\overline{\text{HREQ}}$ falling | $t_{mrwbsyl}$ | - | 12 | ns |

NOTES:1. The system designer should be aware that the actual maximum speed of the communication port may be limited by the firmware application. Hardware handshaking on the $\overline{\text{HREQ}}$ pin/bit should be observed to prevent overflowing the input data buffer.


©Copyright 2005 Cirrus Logic, Inc.

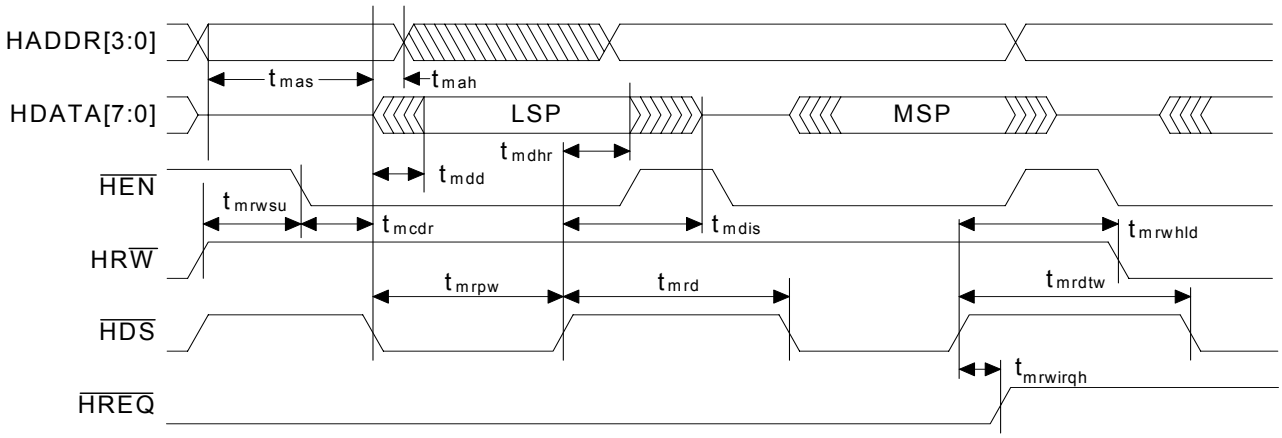



**Figure 14. Host Port Read Cycle Timing - Motorola Mode**

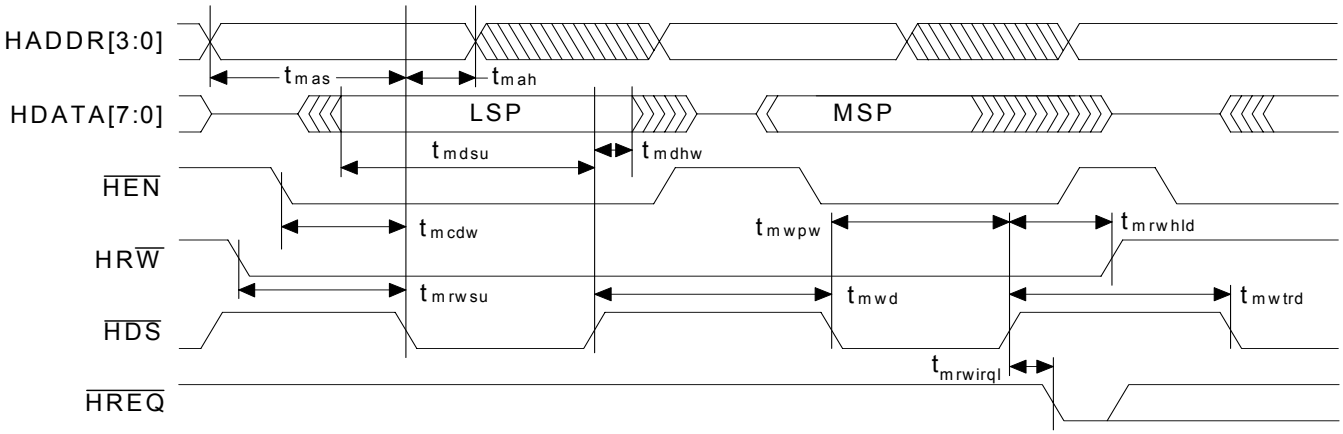



**Figure 15. Host Port Write Cycle Timing - Motorola Mode**


©Copyright 2005 Cirrus Logic, Inc.

## 7.3 Host Port Timing - Intel® Mode

($C_L$ = 20 pF)

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Address setup before $\overline{HCS}$ and $\overline{HRD}$ low or $\overline{HCS}$ and $\overline{HWR}$ low | $t_{ias}$ | 5 | - | ns |
| Address hold time after $\overline{HCS}$ and $\overline{HRD}$ low or $\overline{HCS}$ and $\overline{HWR}$ high | $t_{iah}$ | 5 | - | ns |
| **Read** | | | | |
| Delay between $\overline{HRD}$ then $\overline{HCS}$ low or $\overline{HCS}$ then $\overline{HRD}$ low | $t_{icdr}$ | 0 | - | ns |
| Data valid after $\overline{HCS}$ and $\overline{HRD}$ low | $t_{idd}$ | - | 18 | ns |
| $\overline{HCS}$ and $\overline{HRD}$ low for read | $t_{irpw}$ | 24 | - | ns |
| Data hold time after $\overline{HCS}$ or $\overline{HRD}$ high | $t_{idhr}$ | 8 | - | ns |
| Data high-Z after $\overline{HCS}$ or $\overline{HRD}$ high | $t_{idis}$ | - | 18 | ns |
| $\overline{HCS}$ or $\overline{HRD}$ high to $\overline{HCS}$ and $\overline{HRD}$ low for next read | $t_{ird}$ | 30 | - | ns |
| $\overline{HCS}$ or $\overline{HRD}$ high to $\overline{HCS}$ and $\overline{HWR}$ low for next write | $t_{irdtw}$ | 30 | - | ns |
| $\overline{HRD}$ rising to $\overline{HREQ}$ rising | $t_{irdirqhl}$ | - | 12 | ns |
| **Write** | | | | |
| Delay between $\overline{HWR}$ then $\overline{HCS}$ low or $\overline{HCS}$ then $\overline{HWR}$ low | $t_{icdw}$ | 0 | - | ns |
| Data setup before $\overline{HCS}$ or $\overline{HWR}$ high | $t_{idsu}$ | 8 | - | ns |
| $\overline{HCS}$ and $\overline{HWR}$ low for write | $t_{iwpw}$ | 24 | - | ns |
| Data hold after $\overline{HCS}$ or $\overline{HWR}$ high | $t_{idhw}$ | 8 | - | ns |
| $\overline{HCS}$ or $\overline{HWR}$ high to $\overline{HCS}$ and $\overline{HRD}$ low for next read | $t_{iwtrd}$ | 30 | - | ns |
| $\overline{HCS}$ or $\overline{HWR}$ high to $\overline{HCS}$ and $\overline{HWR}$ low for next write | $t_{iwd}$ | 30 | - | ns |
| $\overline{HWR}$ rising to $\overline{HREQ}$ falling | $t_{iwrbsyl}$ | - | 12 | ns |

NOTES:1. The system designer should be aware that the actual maximum speed of the communication port may be limited by the firmware application. Hardware handshaking on the $\overline{HREQ}$ pin/bit should be observed to prevent overflowing the input data buffer.


©Copyright 2005 Cirrus Logic, Inc.

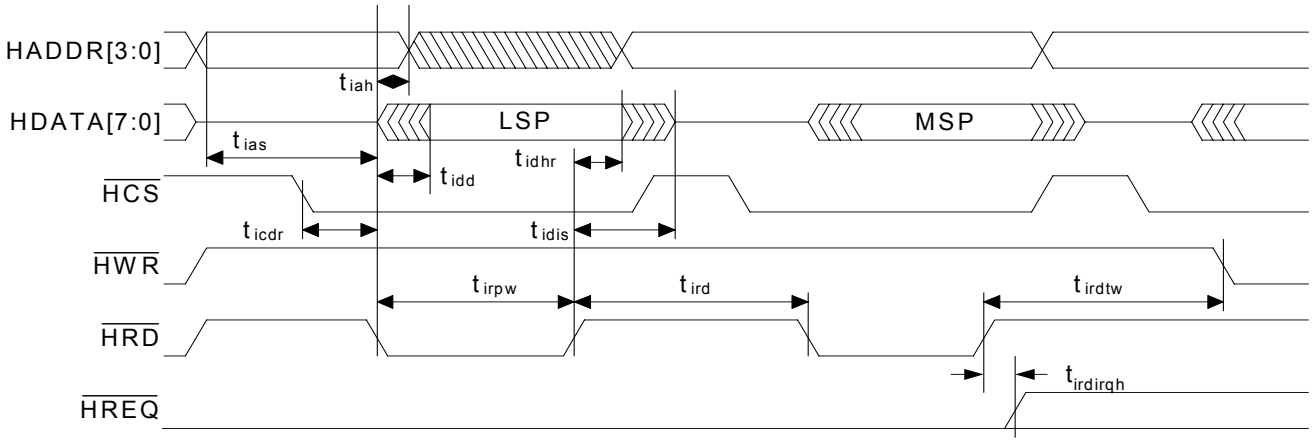



**Figure 16. Parallal Control Port - Intel Mode Read Cycle**

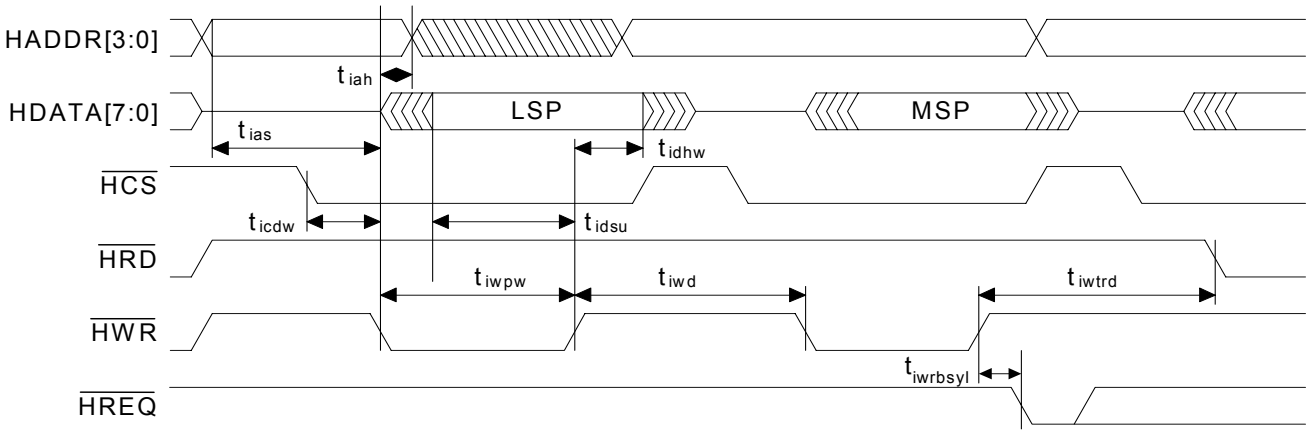



**Figure 17. Parallel Control Port - Intel Mode Write Cycle**


©Copyright 2005 Cirrus Logic, Inc.

## 7.4 Protocol and Messages

The message conduit is used to issue commands to the CS1810xx/CS4961xx and retrieve HMI status. The data conduit is used to transfer data dependent on the HMI state as determined by commands issued by the host via the message conduit.

### *7.4.1 Messages*

Messages are used to efficiently invoke action in the CS1810xx/CS4961xx. To send a message, the host optionally writes to the A, B, and C registers. Writing to the D register transmits the message to the CS1810xx/CS4961xx. A listing of all HMI messages is shown in Table 4. Refer also to "HMI Definitions" on page 33 and "HMI Access Code" on page 34.

| Message | DRQ Handshake Mode | A | B | C | D |
|---|---|---|---|---|---|
| *Translate Address* | n/c | Address (MS) | Address | Address (LS) | 0xB3 |
| *Acknowledge Interrupt* | n/c | n/c | n/c | n/c | 0xB4 |
| *Identify* | read | n/c | n/c | 7 | 0xB5 |
| *Goto Packet Transmit Buffer* | write | n/c | n/c | 6 | 0xB5 |
| *Goto Translation* | write | n/c | n/c | 5 | 0xB5 |
| *Acknowledge Packet Receipt* | n/c | n/c | n/c | 4 | 0xB5 |
| *Transmit Packet* | n/c | n/c | n/c | 3 | 0xB5 |
| *Goto Counters* | read | n/c | n/c | 2 | 0xB5 |
| *Goto Packet Receive Buffer* | read | n/c | n/c | 1 | 0xB5 |
| *Goto Translation* | read | n/c | n/c | 0 | 0xB5 |

**Table 4. HMI messages**


©Copyright 2005 Cirrus Logic, Inc.

#### 7.4.1.1. Translate Address

*Translate Address* does not actually update the address pointers but initiates the processing required to eventually move them. The host can accomplish other tasks, including HMI Reads and Writes while the address translation is being processed. A logical description of *Translate Address* is given below. A contextual use of the *Translate Address* operation is shown in the reference implementations. Refer also to "HMI Definitions" on page 33 and "HMI Access Code" on page 34.

```
void TranslateAddress(
   long address )
{
   int msgack = MSG_D;
   MSG_A = ( address & 0xff0000 ) >> 16;
   MSG_B = ( address & 0xff00 ) >> 8;
   MSG_C = address & 0xff;
   MSG_D = CVR_TRANSLATE_ADDRESS;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```

#### 7.4.1.2. Interrupt Acknowledge

Causes HACK to be de-asserted.

```
void InterruptAck( void )
{
   int msgack = MSG_D;
   MSG_D = CVR_INTERRUPT_ACK;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```

#### 7.4.1.3. Goto Packet

Moves HMI pointers to bridgeRxPktBuffer (write = 0) or bridgeTxPktBuffer (write = 1).

```
void GotoPacket(
   bool write )
{
   int msgack = MSG_D;
   MSG_C = write ? MOP_GOTO_PACKET_TRANSMIT : MOP_GOTO_PACKET_RECEIVE;
   MSG_D = CVR_MULTIPLEX_OP;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```

#### 7.4.1.4. Goto Translation

Moves HMI data pointers to the results of the most recently completed translate address operation. The *write* parameter dictates the operation of the $\overline{\text{HREQ}}$ signal and only needs to be supplied for applications using hardware data handshaking via this signal.

```
void GotoTranslation(
   bool write = 0 )
{
   int msgack = MSG_D;
   MSG_C = write ? MOP_GOTO_TRANSLATION_WRITE : MOP_GOTO_TRANSLATION_READ;
   MSG_D = CVR_MULTIPLEX_OP;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```


©Copyright 2005 Cirrus Logic, Inc.

#### 7.4.1.5. Packet Received

Sets bridgeRxPkt = bridgeRxReady thus acknowledging receipt of the packet in bridgeRxPktBuffer.

```
void PacketReceive( void )
{
   int msgack = MSG_D;
   MSG_C = MOP_PACKET_RECEIVE;
   MSG_D = CVR_MULTIPLEX_OP;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```

#### 7.4.1.6. Packet Transmit

Sets bridgeTxPkt = bridgeTxPktDone+1 thus initiating transmission of the contents of bridgeTxPktBuffer. Presumably bridgeTxPktBuffer has been previously written with valid packet data.

```
void PacketTransmit( void )
{
   int msgack = MSG_D;
   MSG_C = MOP_PACKET_TRANSMIT;
   MSG_D = CVR_MULTIPLEX_OP;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```

#### 7.4.1.7. Goto Counters

Moves HMI data pointers to interrupt status variables (beginning at *hackStatus*).

```
void GotoCounters( void )
{
   int msgack = MSG_D;
   MSG_C = MOP_GOTO_COUNTERS;
   MSG_D = CVR_MULTIPLEX_OP;
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```


©Copyright 2005 Cirrus Logic, Inc.

### *7.4.2 Status*

HMI status can always be retrieved by reading the message conduit. Status is updated in a pipelined manner whenever the Message D register is read. Reading the message conduit gives the current status as of the last time the conduit was read. Bitfields in the HMI Status Register are outlined in Table 5 below. Refer also to "HMI Definitions" on page 33 and "HMI Access Code" on page 34.

| Status | Bit(s) |
|---|---|
| Reserved | [31:24] |
| *Region Length* | [23:8] |
| Reserved | [7:5] |
| *Writable Region* | 4 |
| *Translation Complete* | 3 |
| *Packet Transmission Complete* | 2 |
| *Received Packet Available* | 1 |
| *Message Togglebit* | 0 |

**Table 5. HMI status bits**


©Copyright 2005 Cirrus Logic, Inc.

### *7.4.3 Data*

Before accessing data, address setup must be performed. Address setup consists of issuing a *Translate Address* request, waiting for the request to complete, then issuing a *Goto Translation*.

Pipelining requires that a *"garbage read"* be performed following an address change. The second word read contains the data for the address requested. No similar pipelining issue exists with respect to write operations.

#### 7.4.3.1. Region length

Distance from the original pointer position (as per *Translate Address*) to the end of the instantiated region. A value of *0* indicates an invalid pointer.

#### 7.4.3.2. Writable Region

When set, this bit indicates the address pointer is positioned within a writable region. *MI* variables may be modified in a writable region by writing data to the data conduit.

#### 7.4.3.3. Translation Complete

When set, this bit indicates that the address translator is available (translation results are available and a new translation request may be submitted). This bit is cleared when a *Translate Address* message is issued and is set when the translation completes.

#### 7.4.3.4. Packet Transmission Complete

This bit is cleared when transmission is initiated by issuance of the *Transmit Packet* message. The bit is set when the packet has been transmitted and the transmit buffer is ready to accept a new packet.

#### 7.4.3.5. Received Packet Available

This bit is set when a packet is received into the packet bridge. It is cleared when the packet data is read and receipt is acknowledged by issuance of an *Acknowledge Packet Receipt* message. Note that *Received Packet Available* only goes low when there are no longer any pending received packets for the packet bridge. The packet bridge has the capacity to queue multiple packets in the receive direction.

#### 7.4.3.6. Message Togglebit

This bit toggles on completion of processing of each message. A safe means for the host to acknowledge processing of messages is as follows:

```
void WaitToggle( void )
{
   int msgack = MSG_D; /* clean pipeline */
   msgack = MSG_D; /* record current state of togglebit */
   MSG_D = YOUR_COMMAND_HERE; /* issue command */
   /* wait for togglebit to flip */
   while( !( ( msgack ^ MSG_D ) & ( 1 << MSG_TOGGLE_BO ) ) );
}
```


©Copyright 2005 Cirrus Logic, Inc.

# 8.0 HMI Reference Code

The following C code provides examples in using HMI messages, HMI status, and the HMI memory map.

## 8.1 HMI Definitions

```
/*============================================================================
** hmi.h
** CobraNet Host Management Interface example code
** Definitions
**----------------------------------------------------------------------------
** $Header$
** Copyright (c) 2004, Peak Audio, a division of Cirrus Logic, Inc.
**============================================================================*/
#define MSG_A 0
#define MSG_B 1
#define MSG_C 2
#define MSG_D 3
#define DATA_A 4
#define DATA_B 5
#define DATA_C 6
#define DATA_D 7
#define CONTROL 8
#define STATUS 9

#define CVR_SET_ADDRESS 0xb2   /* Not availbale on CS1810xx/CS4961xx/CM-2. */
                               /*CM-1 and Reference Design only. */
#define CVR_TRANSLATE_ADDRESS 0xb3
#define CVR_INTERRUPT_ACK 0xb4
#define CVR_MULTIPLEX_OP 0xb5

#define MOP_GOTO_TRANSLATION_READ 0
#define MOP_GOTO_TRANSLATION_WRITE 5
#define MOP_GOTO_PACKET_RECEIVE 1
#define MOP_GOTO_PACKET_TRANSMIT 6
#define MOP_GOTO_COUNTERS 2
#define MOP_PACKET_TRANSMIT 3
#define MOP_PACKET_RECEIPT 4
#define MOP_IDENTIFY 7

#define MSG_TOGGLE_BO 0
#define MSG_RXPACKET_BO 1
#define MSG_TXPACKET_BO 2
#define MSG_TRANSLATION_BO 3
#define MSG_WRITABLE_BO 4
#define MSG_LENGTH_BO 8
```


©Copyright 2005 Cirrus Logic, Inc.

## 8.2 HMI Access Code

```
/*=========================================================================
** hmi.c
** CobraNet Host Management Interface example code
** Simple edition
**-------------------------------------------------------------------------
** $Header$
** Copyright (c) 2004, Peak Audio, a division of Cirrus Logic, Inc.
**=========================================================================*/
#include "hmi.h"

/* variables model HMI state */
long PeekLimit;
long PeekPointer = -1;
long PokeLimit;
long PokePointer = -1;

/* access host port hardware */
#define HMI_BASE 0

unsigned char ReadRegister(
   int hmiregister )
{
   return *(unsigned char volatile *const) ( hmiregister + HMI_BASE );
}

void WriteRegister(
   int hmiregister,
   unsigned char value )
{
   *(unsigned char volatile *const) ( hmiregister + HMI_BASE ) = value;
}

void SendMessage(
   unsigned char message )
{
   int msgack = ReadRegister( MSG_D );
   /* issue (last byte of) message */
   WriteRegister( MSG_D, message );
   /* wait for acceptance of message */
   while( !( ( msgack ^ ReadRegister( MSG_D ) ) & ( 1 << MSG_TOGGLE_BO ) ) );
}

void SetAddress(
   long address )
{
   /* translate address */
   WriteRegister( MSG_A, ( address & 0xff0000 ) >> 16 );
   WriteRegister( MSG_B, ( address & 0xff00 ) >> 8 );
   WriteRegister( MSG_C, address & 0xff );
   SendMessage( CVR_TRANSLATE_ADDRESS );
   /* wait for completion of translate address */
```


©Copyright 2005 Cirrus Logic, Inc.

```
   while( !( ReadRegister( MSG_D ) & ( 1 << MSG_TRANSLATION_BO ) ) );
   /* goto translation */
   WriteRegister( MSG_C, MOP_GOTO_TRANSLATION_READ );
   SendMessage( CVR_MULTIPLEX_OP );
   /* "garbage" read clears data pipeline */
   ReadRegister( DATA_D );
   /* maintain local pointers */
   PeekPointer = PokePointer = address;
   PeekLimit = PokeLimit = PeekPointer +
      ReadRegister( MSG_C ) + ( ReadRegister( MSG_B ) << 8 );
   /* read-only region addressed */
   if( !( ReadRegister( MSG_A ) & ( 1 << MSG_WRITABLE_BO ) ) ) {
      PokeLimit = PokePointer;
   }
}

unsigned long Peek(
   long address )
{
   if( address != PeekPointer ) {
      SetAddress( address );
   }
   if( PeekPointer >= PeekLimit ) {
      throw "Peek addressing error!";
   }
   unsigned long value = ReadRegister( DATA_A ) << 24;
   value += ReadRegister( DATA_B ) << 16;
   value += ReadRegister( DATA_C ) << 8;
   value += ReadRegister( DATA_D );
   PeekPointer++; /* maintain local pointer */
   return value;
}

void Poke(
   long address,
   unsigned long value )
{
   if( address != PokePointer ) {
      SetAddress( address );
   }
   if( PokePointer >= PokeLimit ) {
      throw "Poke addressing error or read-only!";
   }
   WriteRegister( DATA_A, (unsigned char) ( ( value >> 24 ) & 0xff ) );
   WriteRegister( DATA_B, (unsigned char) ( ( value >> 16 ) & 0xff ) );
   WriteRegister( DATA_C, (unsigned char) ( ( value >> 8 ) & 0xff ) );
   WriteRegister( DATA_D, (unsigned char) ( value & 0xff ) );
   /* maintain local pointers */
   PokePointer++;
   PeekPointer = -1; /* force SetAddress()next Peek() to freshen data */
}
```


©Copyright 2005 Cirrus Logic, Inc.

## 8.3 CM-1, CM-2 Auto-detection

The following function is useful for systems that support both the CM-1 and CM-2 or where a CobraNet interface is an optional add-in.

Detect() returns 0 if no CobraNet interface module is detected, 1 for CM-1 and 2 for CM-2.

```
int Detect( void ) {
   /* check for presence of CM-1 */
   MSG_B = 0x55; /* write to CM-1 CVR register */
   DATA_A = 0xaa; /* write to unused CM-1 register to flip data bus */
   if( MSG_B == 0x55 ) { /* read back CVR */
      /* redo same detection with different data */
      MSG_B = 0x3c;
      DATA_A = 0xc3;
      if( MSG_B == 0x3c ) {
         return 1; /* CM-1 detected */
      }
   }
   /* check for presence of CM-2 */
   /* issue identify command */
   MSG_C = MOP_IDENTIFY;
   MSG_D = CVR_MULTIPLEX_OP;
   int msgack = MSG_D; /* clean pipeline */
   msgack = MSG_D;
   /* wait for togglebit to flip in response to command */
   int tm0 = gettimeofday();
   while( !( ( MSG_D ^ toggle ) & ( 1 << MSG_TOGGLE_BO ) ) ) {
      int tm1 = gettimeofday();
      if( ( tm1 - tm0 ) > time_out ) {
         return 0; /* command timed out, no CobraNet interface present */
      }
   }
   int garbage = MSG_D; /* clean pipeline */
   /* verify identify results */
   if( DATA_A == 'C' ) if( DATA_B == 'S' )
         if( DATA_C == ( 18101 >> 8 ) ) if( DATA_D == ( 18101&0xff ) {
      return 2; /* CM-2 detected */
   }
   return 0; /* no interface or non-supported interface */
}
```


©Copyright 2005 Cirrus Logic, Inc.

# 9.0 Mechanical Drawings and Schematics

The section contains detailed drawings of the CM-2 board and CS1810xx/CS4961xx device package design. The mechanical drawings are arranged as follows:


- "CM-2 Module Assembly Drawing, Top" on page 38
- "General PCB Dimensions" on page 40
- "Example Configuration, Side View" on page 41
- "Faceplate Dimensions" on page 42
- "Connector Detail" on page 43
- "CM-2 RevF Schematic Page 1 of 7" on page 44
- "CM-2 RevF Schematic Page 2 of 7" on page 45
- "CM-2 RevF Schematic Page 3 of 7" on page 46
- "CM-2 RevF Schematic Page 4 of 7" on page 47
- "CM-2 RevF Schematic Page 5 of 7" on page 48
- "CM-2 RevF Schematic Page 6 of 7" on page 49
- "CM-2 RevF Schematic Page 7 of 7" on page 50
- "144-Pin LQFP Package Drawing" on page 51



©Copyright 2005 Cirrus Logic, Inc.

## 9.1 CM-2 Mechanical Drawings

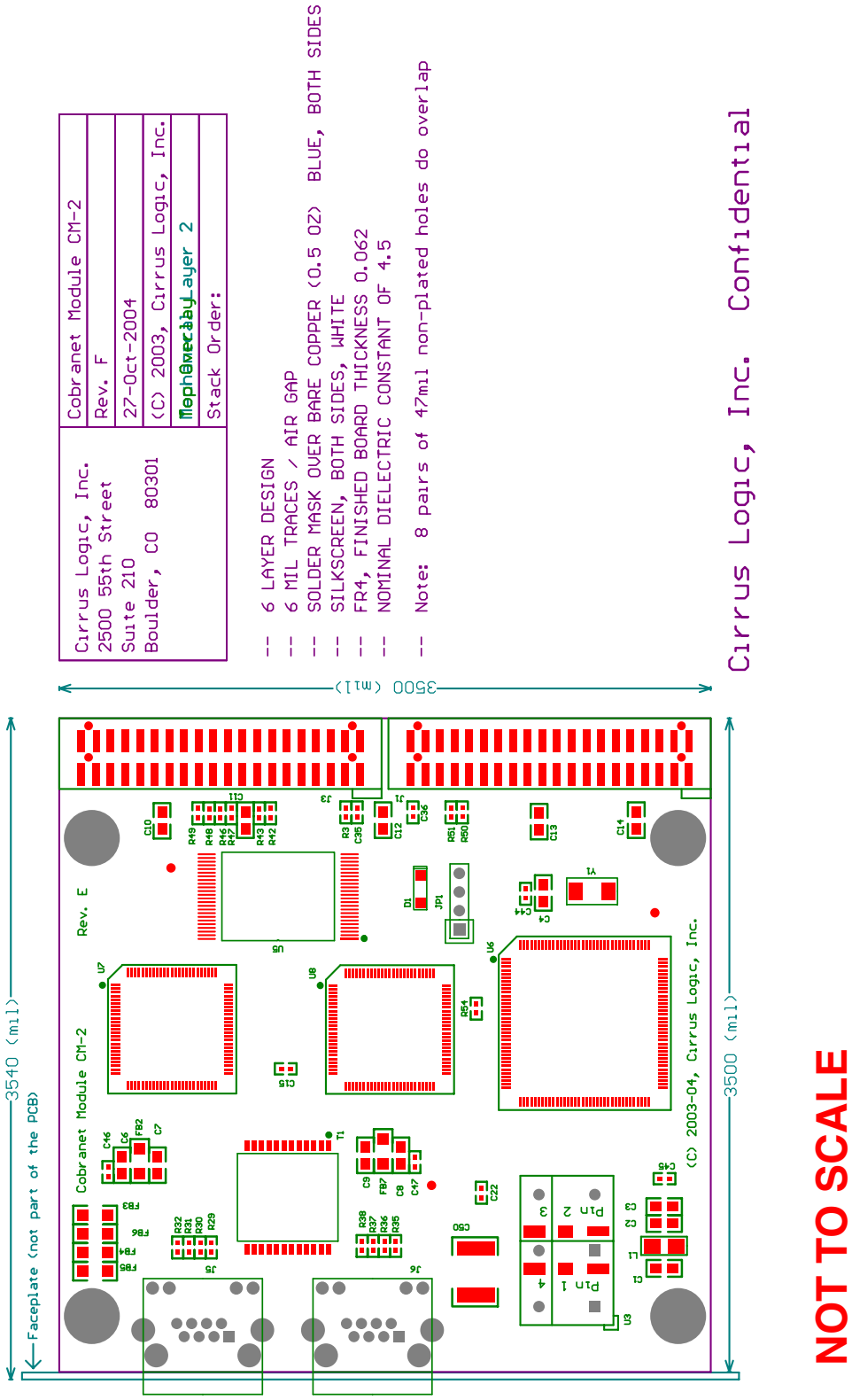



**Figure 18. CM-2 Module Assembly Drawing, Top**


©Copyright 2005 Cirrus Logic, Inc.

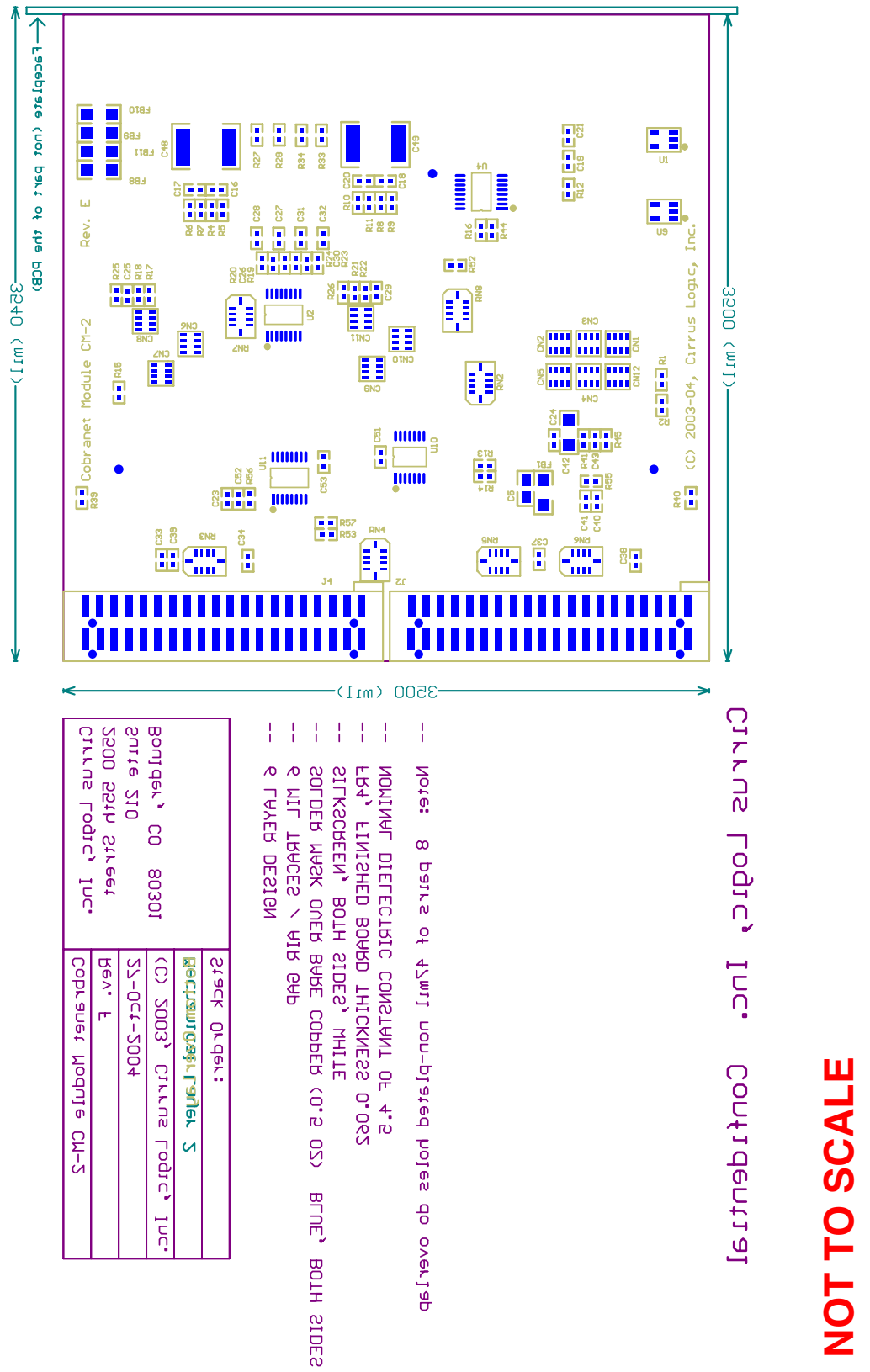



**Figure 19. CM-2 Module Assembly Drawing, Bottom**


©Copyright 2005 Cirrus Logic, Inc.

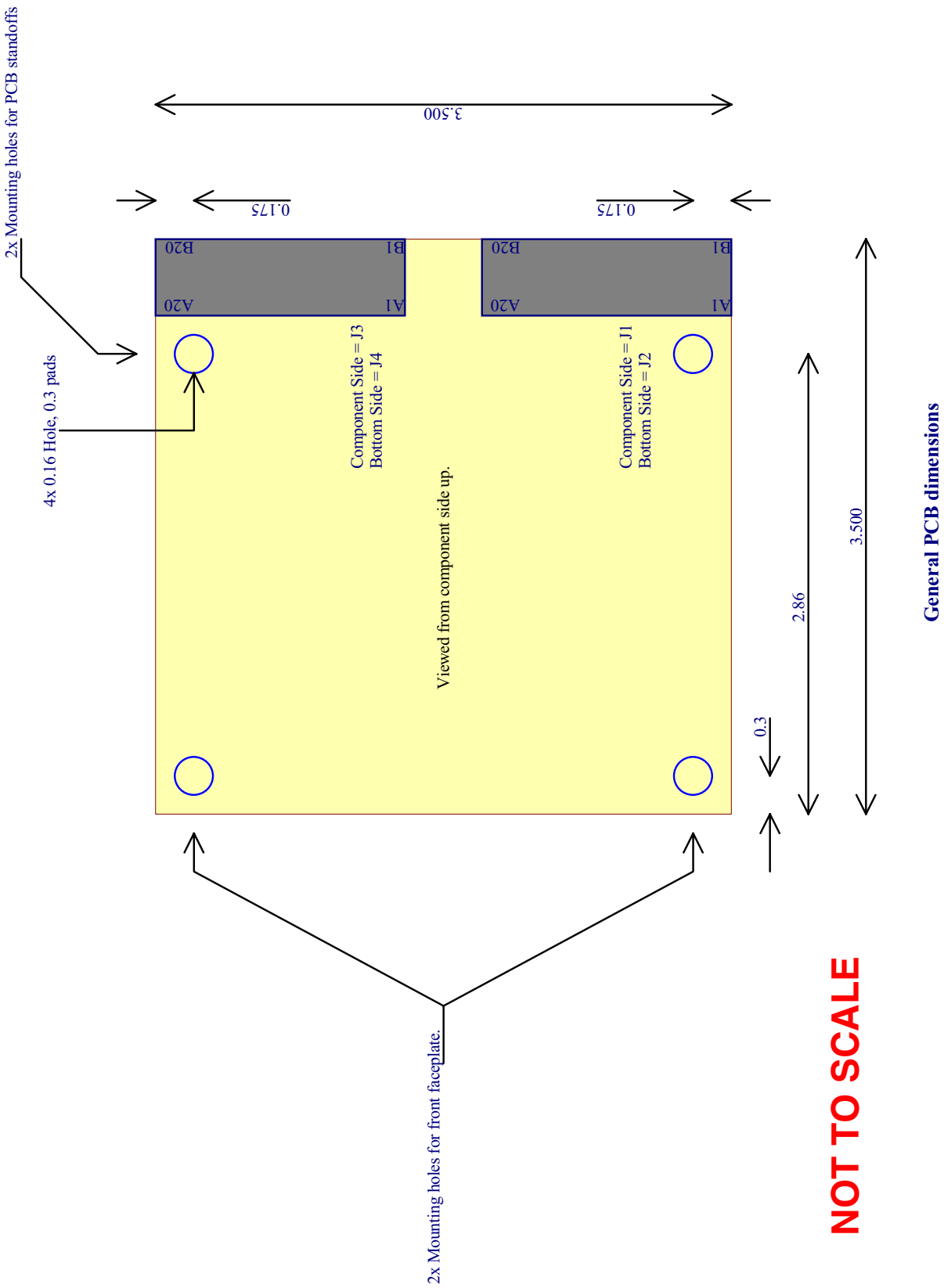
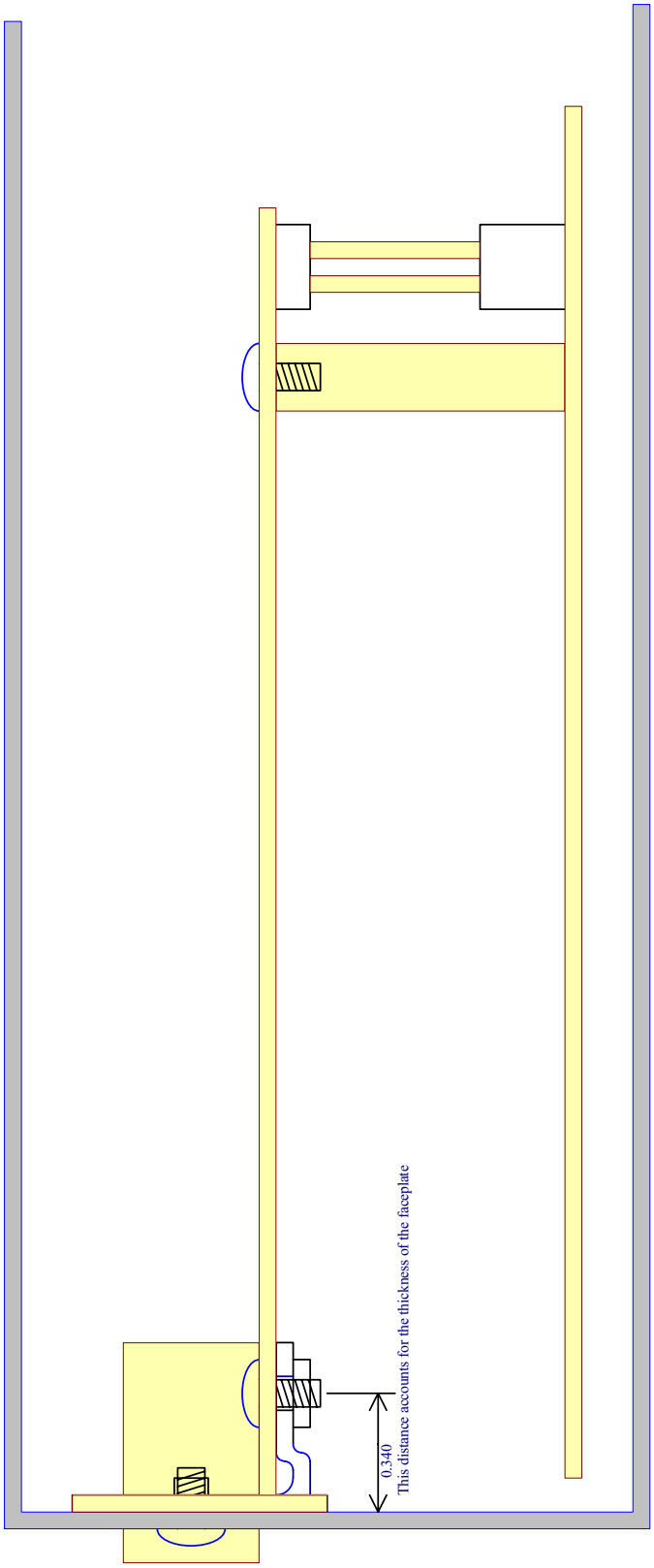



**Figure 20. General PCB Dimensions**


©Copyright 2005 Cirrus Logic, Inc.




NOT TO SCALE

**Figure 21. Example Configuration, Side View**


©Copyright 2005 Cirrus Logic, Inc.

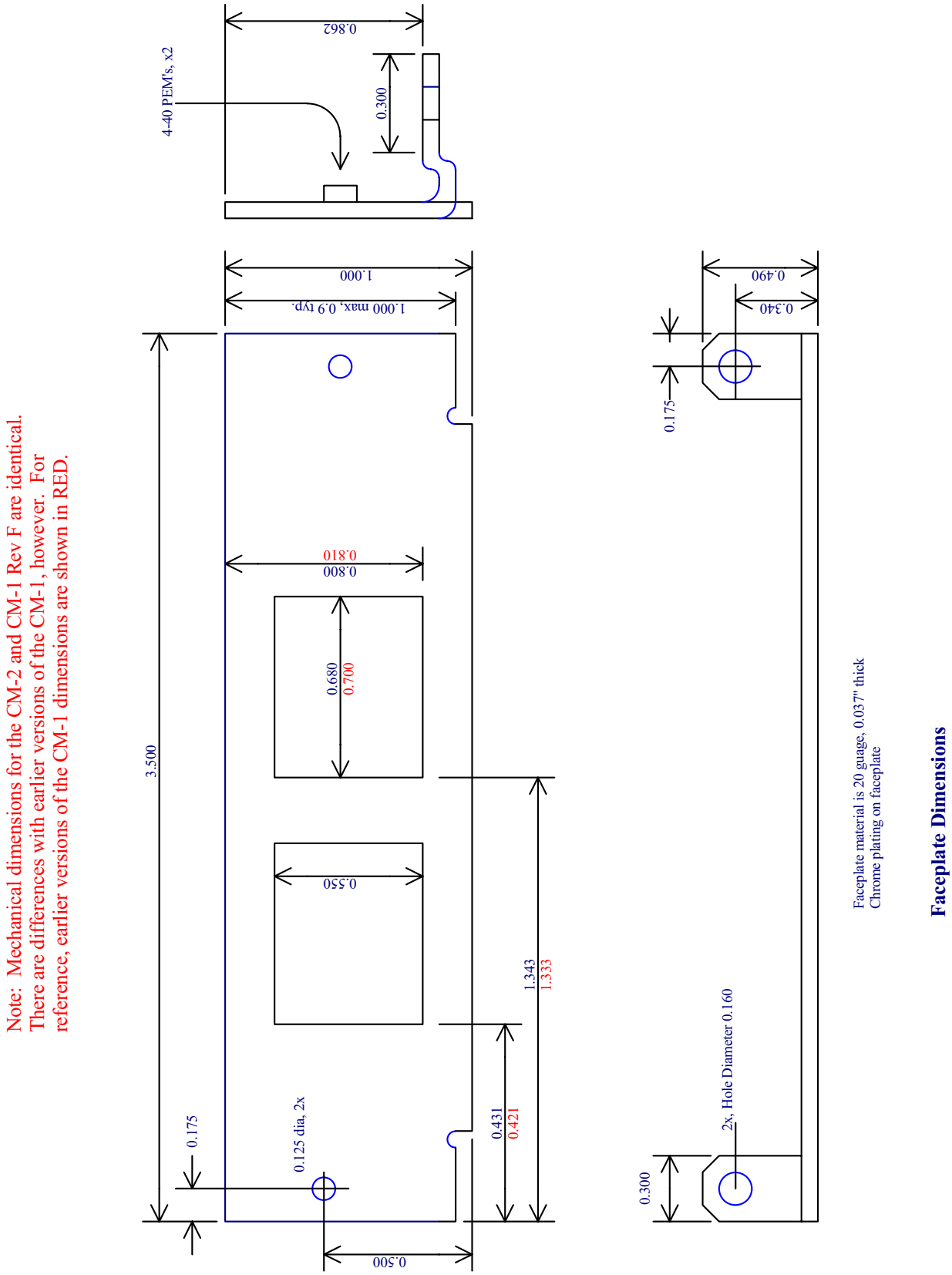
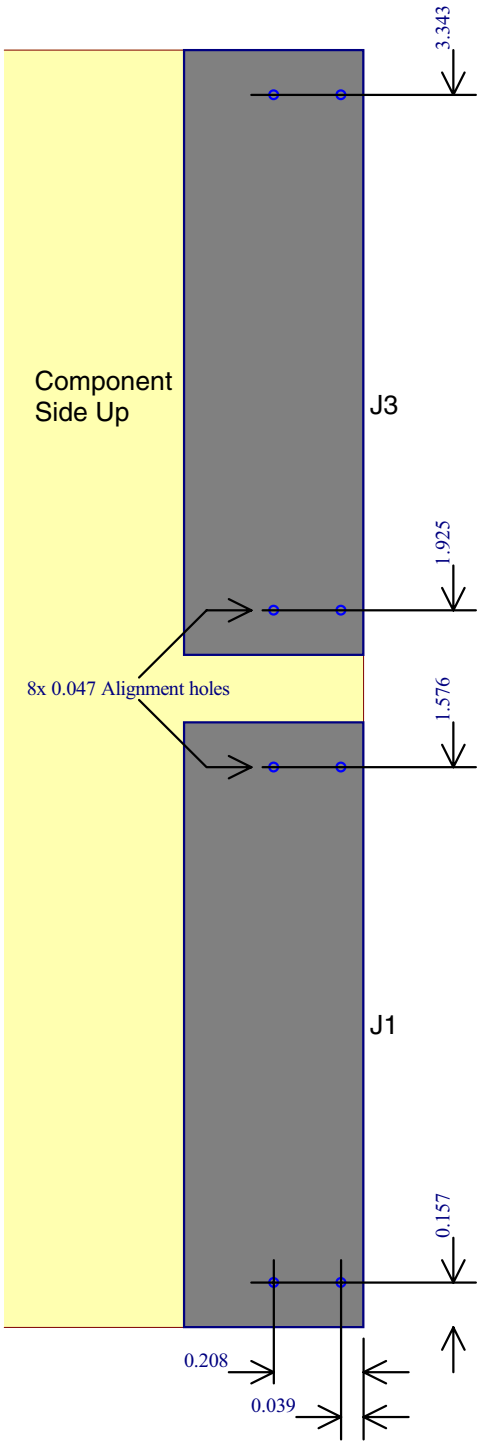



**Figure 22. Faceplate Dimensions**


©Copyright 2005 Cirrus Logic, Inc.

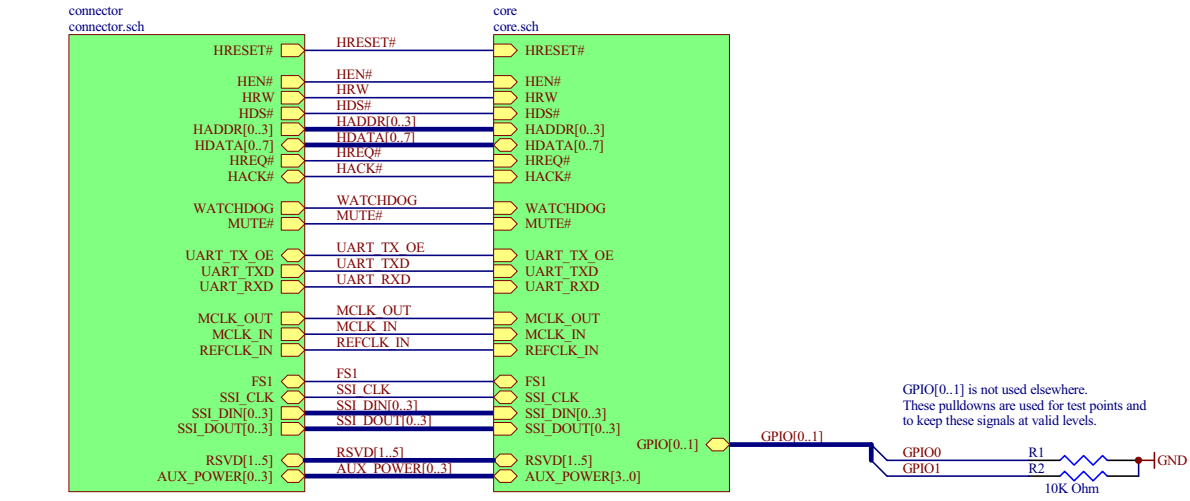



NOT TO SCALE

**Figure 23. Connector Detail**


©Copyright 2005 Cirrus Logic, Inc.

# 9.2 CM-2 Schematics



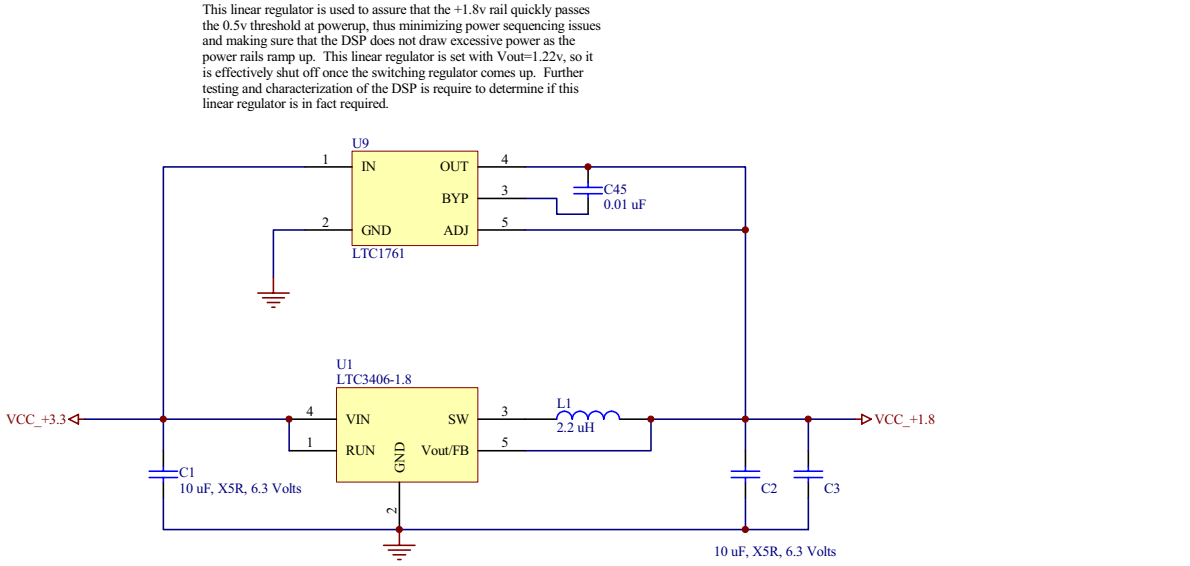




**Figure 24. CM-2 RevF Schematic Page 1 of 7**


©Copyright 2005 Cirrus Logic, Inc.

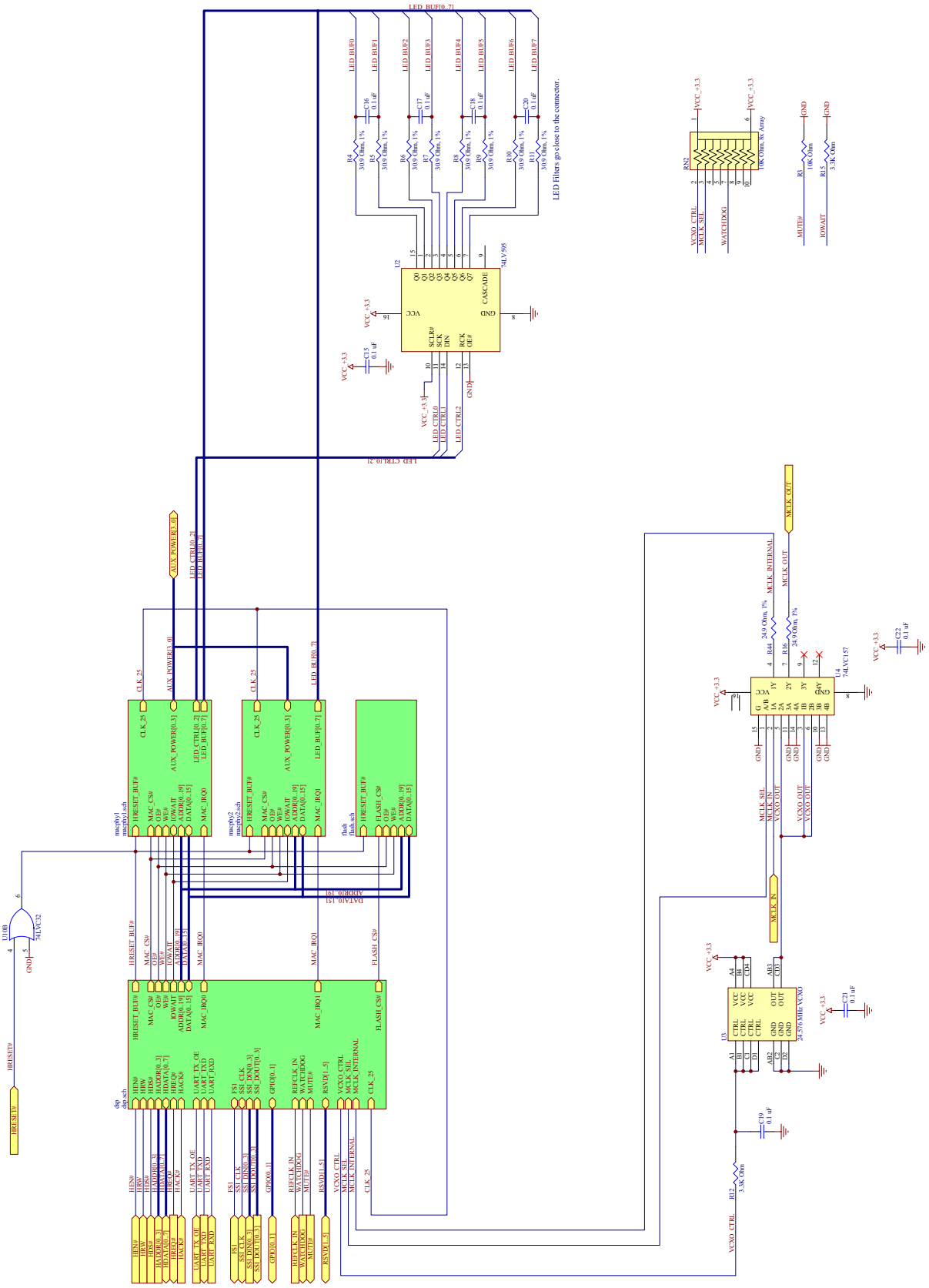


**Figure 25. CM-2 RevF Schematic Page 2 of 7**


**©Copyright 2005 Cirrus Logic, Inc.**

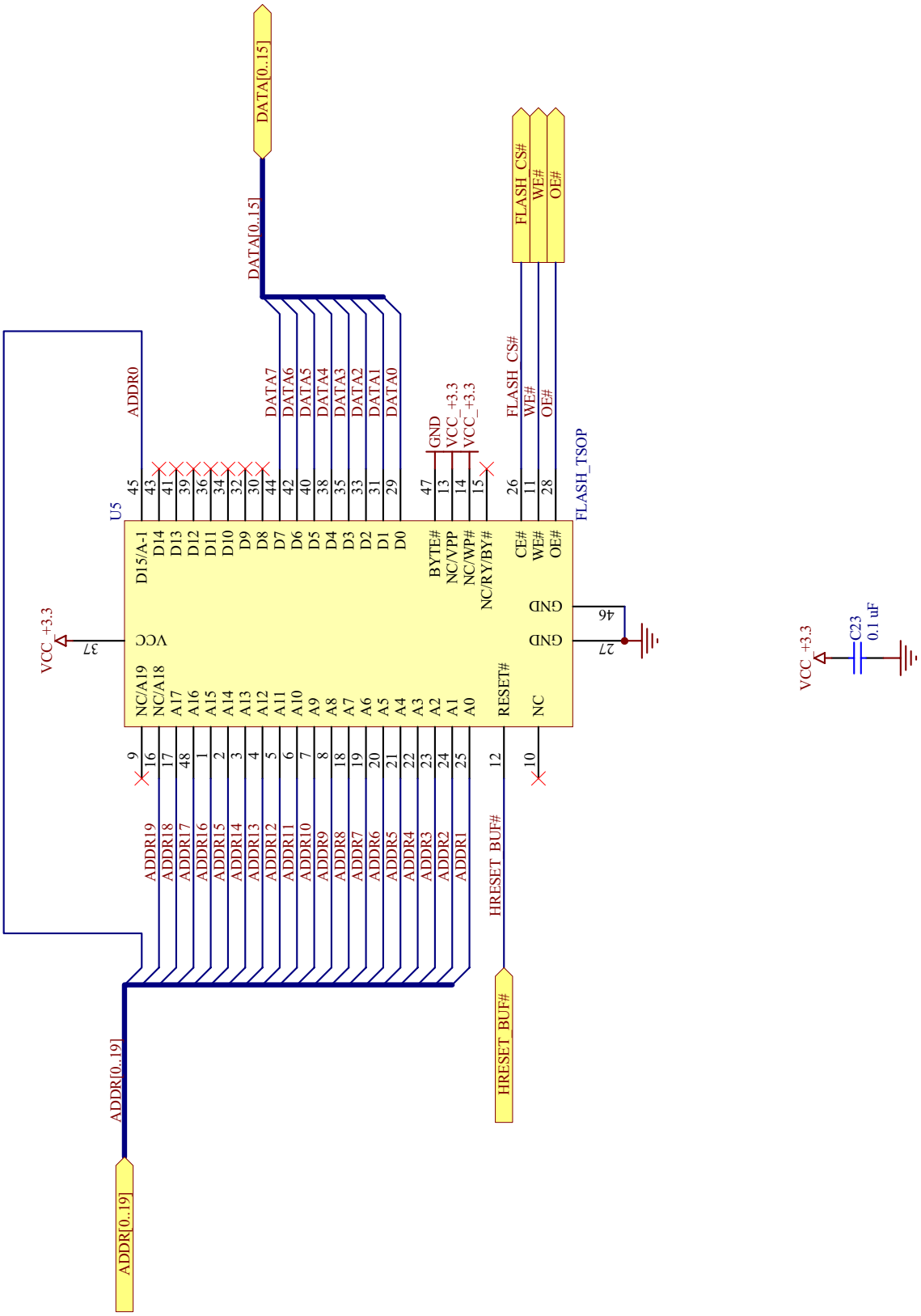



**Figure 26. CM-2 RevF Schematic Page 3 of 7**


©Copyright 2005 Cirrus Logic, Inc.

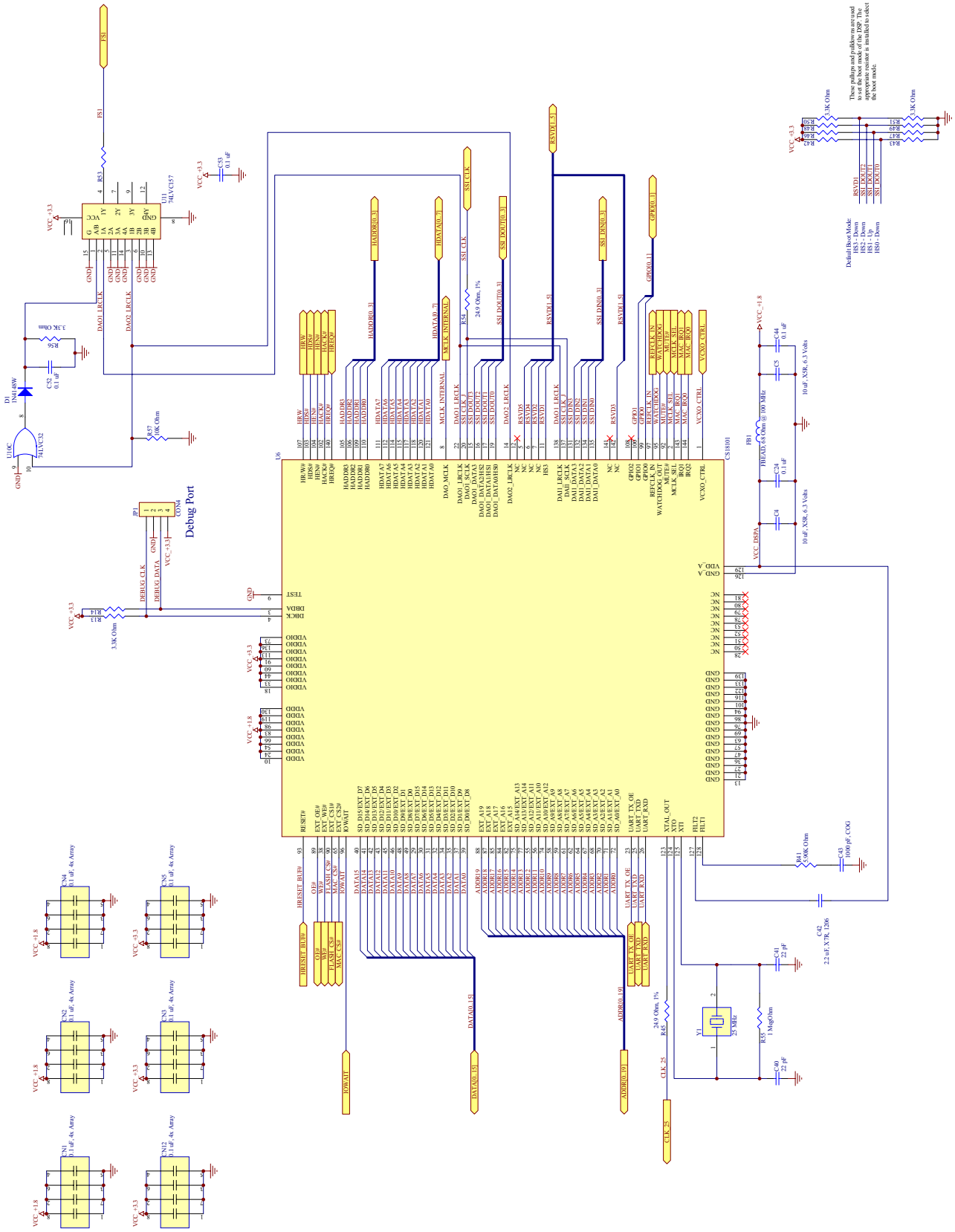


**Figure 27. CM-2 RevF Schematic Page 4 of 7**


©Copyright 2005 Cirrus Logic, Inc.

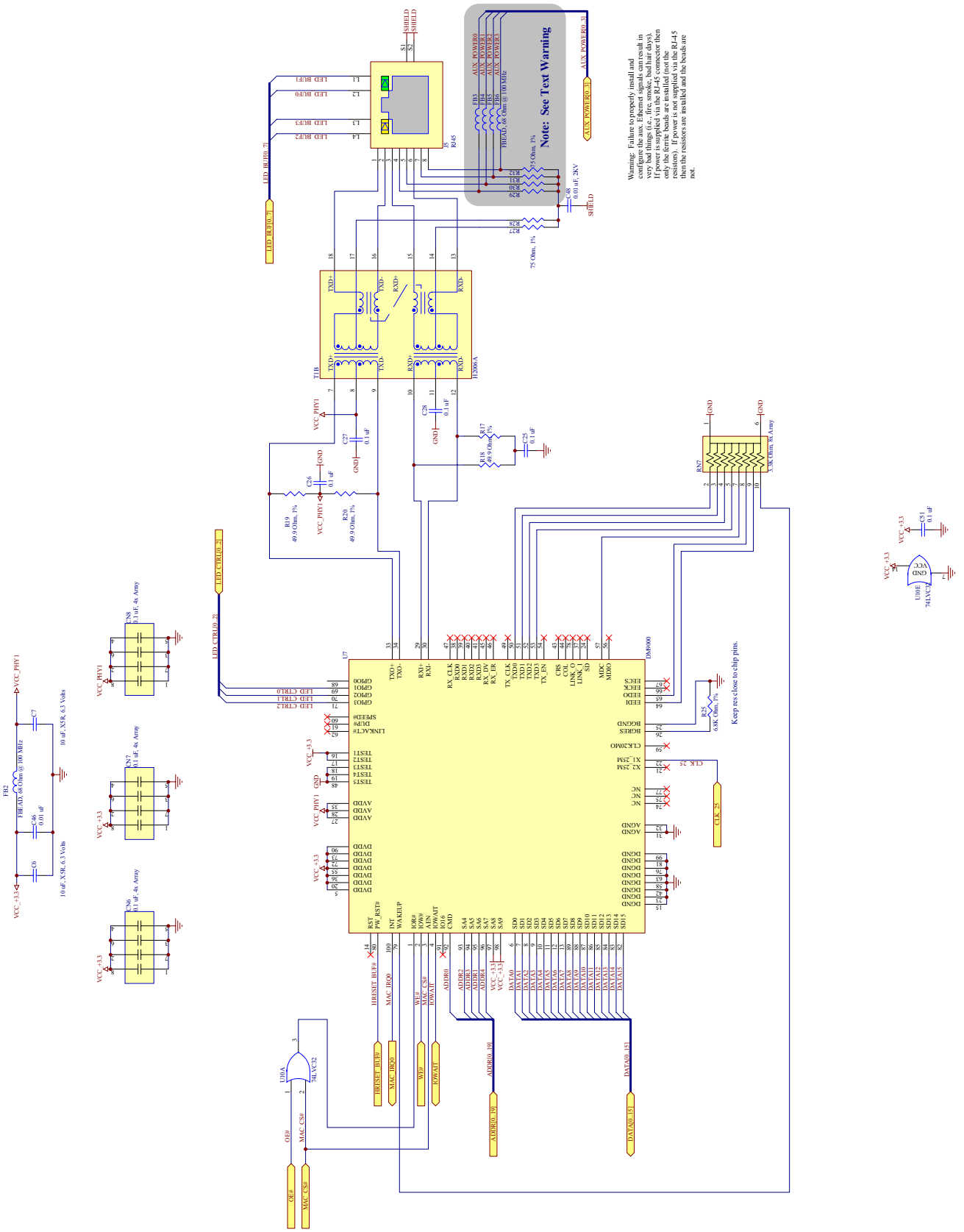



**Figure 28. CM-2 RevF Schematic Page 5 of 7**


©Copyright 2005 Cirrus Logic, Inc.

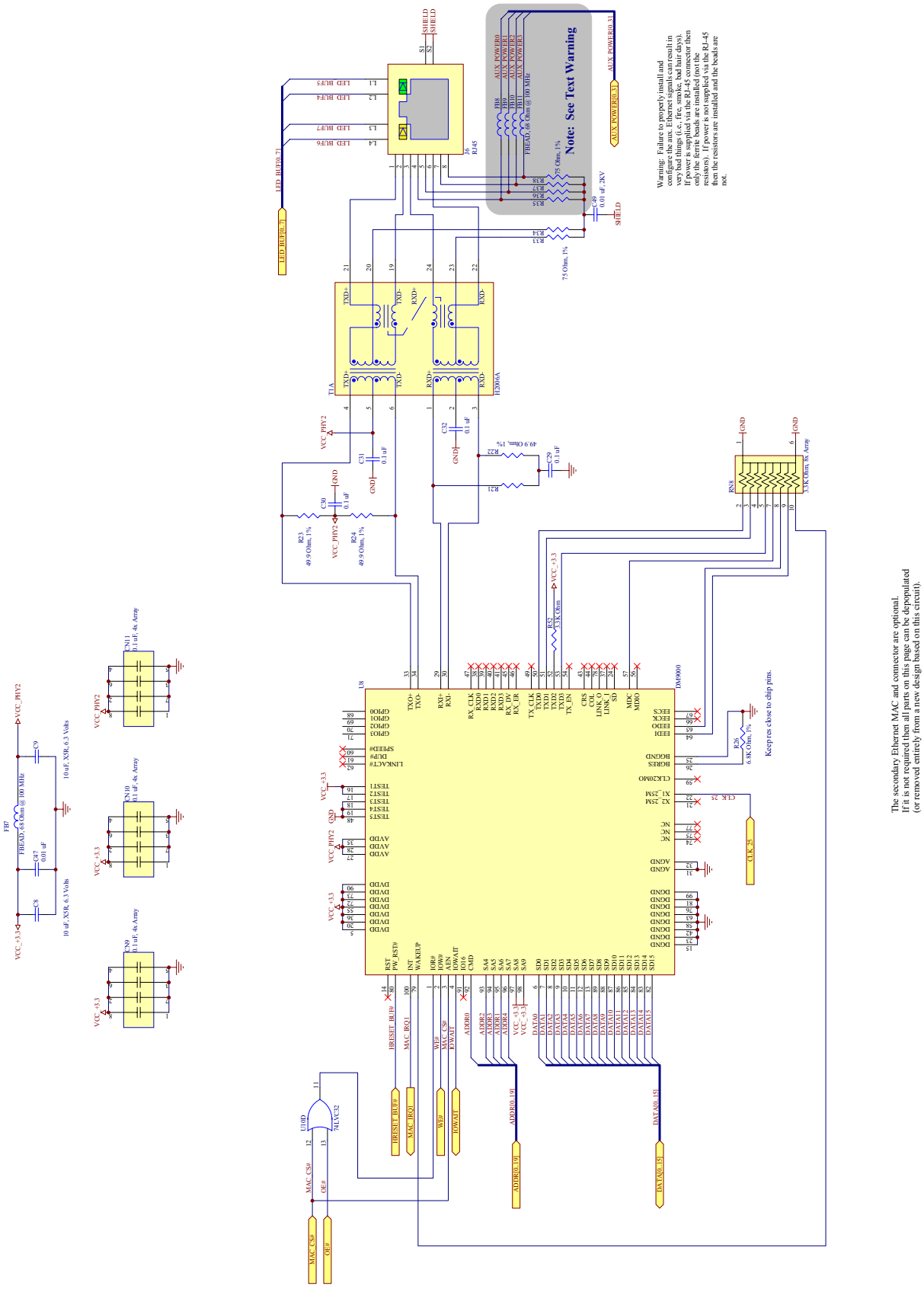



**Figure 29. CM-2 RevF Schematic Page 6 of 7**


©Copyright 2005 Cirrus Logic, Inc.

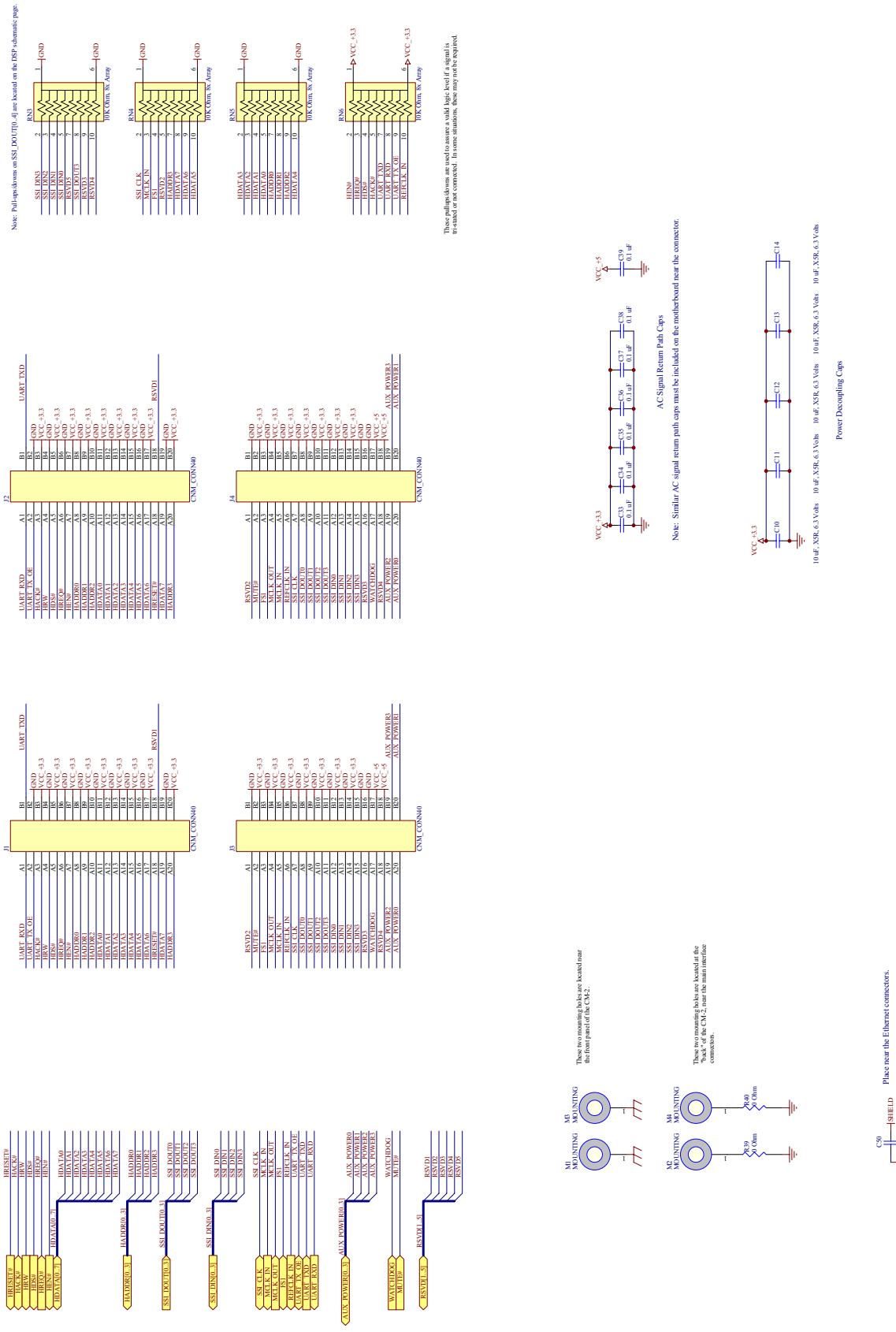
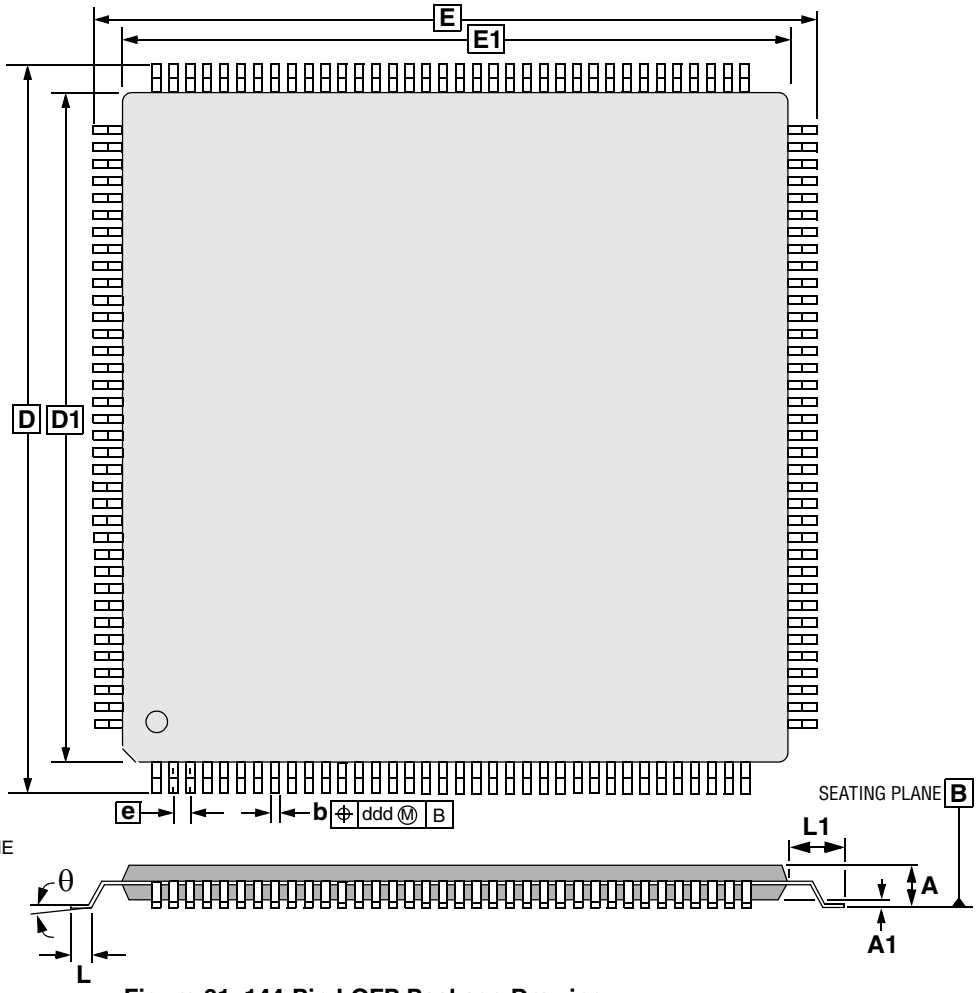


**Figure 30. CM-2 RevF Schematic Page 7 of 7**


**©Copyright 2005 Cirrus Logic, Inc.**

## 9.3 CS1810xx/CS4961xx Package




Notes:

1. Controlling dimension is millimeter.
2. Dimensioning and tolerancing per ASME Y14.5M-1994.

**Figure 31. 144-Pin LQFP Package Drawing**

| DIM | MILLIMETERS | | | INCHES | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | --- | --- | 1.60 | --- | --- | .063” |
| A1 | 0.05 | --- | 0.15 | .002” | --- | .006” |
| b | 0.17 | 0.22 | 0.27 | .007” | .009” | .011” |
| D | | 22.00 BSC | | | .866” | |
| D1 | | 20.00 BSC | | | .787” | |
| E | | 22.00 BSC | | | .866” | |
| E1 | | 20.00 BSC | | | .787” | |
| e | | 0.50 BSC | | | .020” | |
| θ | 0° | --- | 7° | 0° | --- | 7° |
| L | 0.45 | 0.60 | 0.75 | .018” | .024” | .030” |
| L1 | | 1.00 REF | | | .039” REF | |
| TOLERANCES OF FORM AND POSITION | | | | | | |
| ddd | | 0.08 | | | .003” | |


©Copyright 2005 Cirrus Logic, Inc.

## 9.4 Temperature Specifications

- Thermal Coefficient (junction-to-ambient): $\theta_{ja}$ - 38° C / Watt
- Ambient Temperature Range: 0-70 deg C
- Junction Temperature Range: 0-125 deg C

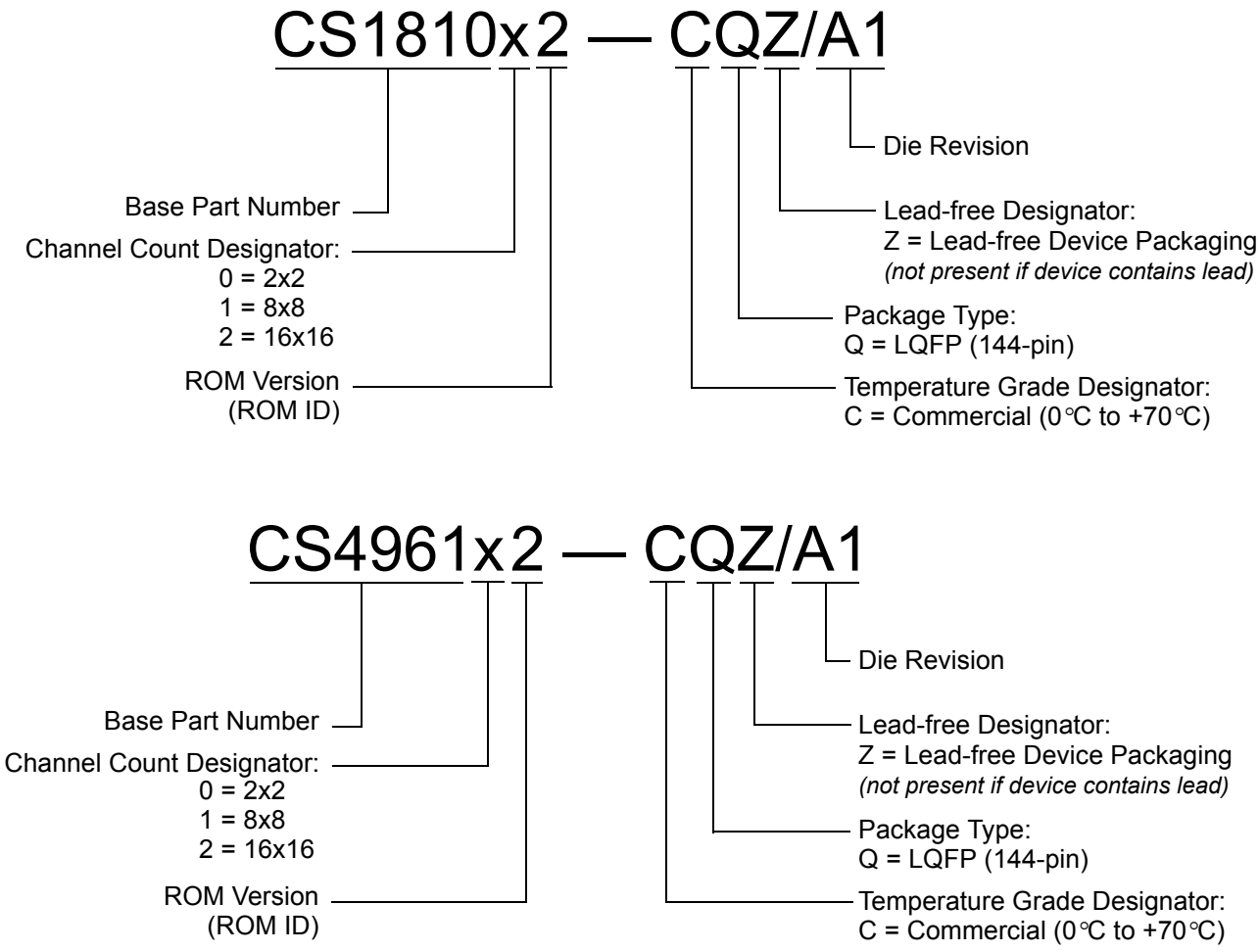

©Copyright 2005 Cirrus Logic, Inc.

# 10.0 Ordering Information

## 10.1 Device Part Numbers

| | | | | |
|---|---|---|---|---|
| **CS181002-CQ/A1** | 2x2 Channels | 0°C to +70°C | 144-pin LQFP | |
| **CS181012-CQ/A1** | 8x8 Channels | 0°C to +70°C | 144-pin LQFP | |
| **CS181022-CQ/A1** | 16x16 Channels | 0°C to +70°C | 144-pin LQFP | |
| **CS181002-CQZ/A1** | 2x2 Channels | 0°C to +70°C | 144-pin LQFP | Lead Free |
| **CS181012-CQZ/A1** | 8x8 Channels | 0°C to +70°C | 144-pin LQFP | Lead Free |
| **CS181022-CQZ/A1** | 16x16 Channels | 0°C to +70°C | 144-pin LQFP | Lead Free |
| **CS496102-CQZ/A1** | 2x2 Channels + DSP | 0°C to +70°C | 144-pin LQFP | Lead Free |
| **CS496112-CQZ/A1** | 8x8 Channels + DSP | 0°C to +70°C | 144-pin LQFP | Lead Free |
| **CS496122-CQZ/A1** | 16x16 Channels + DSP | 0°C to +70°C | 144-pin LQFP | Lead Free |

## 10.2 Device Part Numbering Scheme




*Note: Go to the Cirrus Logic Internet site at http://www.cirrus.com to find contact information for your local sales representative.*

**Figure 32. Device Part Numbering Explanation**


©Copyright 2005 Cirrus Logic, Inc.

## Contacting Cirrus Logic Support

For all product questions and inquiries contact a Cirrus Logic Sales Representative.
To find the one nearest to you go to **www.cirrus.com**

IMPORTANT NOTICE

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

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

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

Motorola is a registered trademark of Motorola, Inc.

Intel is a registered trademark of Intel, Inc.


**©Copyright 2005 Cirrus Logic, Inc.**

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Cirrus Logic:

CPB181012-CM2Z-FB CPB181012-CM2Z-MT CPB181022-CM2Z-FB CPB181022-CM2Z-MT CPB496122-CM2Z-FB CPB496122-CM2Z-MT CS496112-IQZ CS496112-IQZR CS496122-IQZ CS496122-IQZR CPB181012-CM2-FB CPB181012-CM2-MT CPB181022-CM2-FB CPB181022-CM2-MT CPB496122-CM2-FB CPB496122-CM2-MT CS181002-CQ CS181002-CQR CS181002-CQZ CS181002-CQZR CS181012-CQ CS181012-CQR CS181012-CQZ CS181012-CQZR CS181022-CQ CS181022-CQR CS181022-CQZ CS181022-CQZR CS496102-CQZ CS496102-CQZR CS496112-CQZ CS496112-CQZR CS496122-CQZ CS496122-CQZR

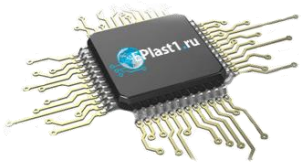
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.




**Как с нами связаться**

**Телефон:** 8 (812) 309 58 32 (многоканальный)
**Факс:** 8 (812) 320-02-42
**Электронная почта:** org@eplast1.ru
**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.