Communication and Networking Riser Specification

Revision 1.2

Intel Corporation

LEGAL NOTICE

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel disclaims all liability, including without limitation claims, costs, damages, and expenses arising out of, directly or indirectly, any claim of product liability, personal injury or death, and liability for infringement of any proprietary rights, relating to any use of information in this specification.

THIS SPECIFICATION IS NOT INTENDED TO DIRECT OR INSTRUCT ANY PARTY IN THE DEVELOPMENT OF ANY IMPLEMENTATION WHERE FAILURE OF THE IMPLEMENTATION COULD CAUSE PERSONAL INJURY OR DEATH.

IN NO EVENT WILL INTEL BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF USE, DIRECT, INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, IRRESPECTIVE OF WHETHER INTEL HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Notice: Implementations developed using the information provided in this specification may infringe the patent rights of various parties including the parties involved in the development of this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights (including without limitation rights under any party's patents) are granted herein.

*Other names and brands may be claimed as the property of others.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

For questions on the Communication and Networking Riser Specification or licensing terms please contact Intel by sending e-mail to cnr.support@intel.com.

Copyright® 2000-2001 Intel Corporation. All rights reserved.

Table of Contents

1 Introduction 9
1.1 Related Documents 10
2 Architectural Overview 10
2.1 Baseline Riser Architecture 11
2.2 Audio, Modem, and Networking Partitioning 11
2.2.1 Audio-Down with Modem and Phone-Line Network-Up Partition 12
2.2.2 Audio-Down with 10/100 LAN-Up Partition 12
2.2.3 Audio-Up, Modem-Up, and Phone-Line Network-Up Partition 13
2.2.4 Combined Audio/Modem-Up with Phone-Line Network-Up Partition 13
2.2.5 Upgrade To Multi-Channel Audio Partition 14
2.3 Unsupported Configurations 14
2.4 Audio Hardware Scalability Model 14
3 Electrical Requirements 15
3.1 Signal Names and Pin-out 15
3.1.1 ATX Family Form Factors Signals 15
3.1.1.1 Common Signal Description for Type A and Type B Connectors 16
3.1.1.2 Type A Connector Unique Signal Description for the ATX Family Form Factors 19
3.1.1.3 Type B Connector Unique Signal Description for the ATX Family Form Factors 20
3.1.1.4 Type A Connector Pin Assignments 22
3.1.1.5 Type B Connector Pin Assignments 23
3.2 Signal Implementation Details 23
3.2.1 CDC_DN_ENAB# Implementation Details 23
3.2.1.1 AC ’97 Disable and Demotion Rules 24
3.2.1.2 CDC_DN_ENAB# Implementation Models 25
3.3 Communication and Network Riser Interface Electrical Specifications 28
4 Power Management and Interface Requirements 29
4.1 Power Management 29
4.1.1 AC ’97 Interface Power Management 29
4.1.1.1 Split Partition AC ’97 and MC ’97 CNR Board 30
4.1.1.2 AMC ’97 Combination Codec CNR Board 30
4.1.2 LAN Interface Power Management 30
4.1.3 USB Interface Power Management and Interface Requirements 30
4.1.3.1 USB Differential Pair and Over Current Sensing 31
4.1.3.2 USB Power Management 31
4.1.4 SMBus Interface Power Management 31
4.2 Reset Considerations 31
4.2.1 AC ’97 Interface Reset Considerations 32
4.3 AC ’97 Clocking Considerations 32
4.3.1 AC ’97 ACPI S0 “Working State” Clocking 32
4.3.2 AC ’97 ACPI S3 and S4 “Sleeping State” Clocking 32
5 Mechanical Requirements 33
5.1 CNR Connector Location for the ATX Family Form Factors 33
5.2 Communication and Networking Riser Board Dimensions 34
5.2.1 Board Dimensions for Standard Full Height ATX Family Form Factors 34
5.2.2 Board Dimensions for Low Profile ATX Family Form Factors 36
5.3 Communication and Networking Riser Edge Card Contact for the ATX Family Form Factors 37
5.4 Communication and Networking Riser Interface Connector Type 39
5.4.1 ATX Family CNR Connector Description 39
5.4.2 CNR Type A and Type B Connector Physical Requirements 40
5.4.3 CNR Type A and Type B Connector Performance Requirements 40
5.5 Communication and Networking Riser I/O Bracket Dimensions 42
5.5.1 Communication and Networking Riser Full Height I/O Bracket for the ATX Family 42
5.5.2 Communication and Networking Riser Low Profile I/O Bracket for the ATX Family 43
5.6 Thermal Requirements 43
5.7 EMI, RFI, and Shielding Requirements 43
6 Operating System, Software, and BIOS Requirements 43

6.1 Operating System Plug-and-Play Requirements 44
6.1.1 Plug and Play EEPROM Requirements 44
6.1.1.1 Plug and Play EEPROM Device Considerations 44
6.1.1.2 Plug and Play EEPROM Connectivity 45
6.1.2 EEPROM Master Configuration Space 45
6.1.2.1 EEPROM ID Register (Word 00h) 46
6.1.2.2 EEPROM Size Register (Word 02h) 46
6.1.2.3 CNR Compliance Register (Word 04h) 46
6.1.2.4 AC '97 Compliance Register (Word 06h) 47
6.1.2.5 Function ID Register (Word 08h) 47
6.1.2.6 Reserved Registers (Words 0Ah – 0Ch) 47
6.1.2.7 Audio Pointer Register (Word 0Eh) 47
6.1.2.8 Modem Pointer Register (Word 10h) 48
6.1.2.9 USB Pointer Register (Word 12h) 48
6.1.2.10 SMBus Pointer Register (Word 14h) 48
6.1.2.11 LAN Pointer Register (Word 16h) 48
6.1.2.12 Reserved Pointer Registers (Words 18h – 2Ch) 48
6.1.2.13 Last Valid Address Register (Word 2Eh) 48
6.1.2.14 Checksum Register (Word 30h) 48
6.1.3 Audio Function PnP EEPROM Contents 48
6.1.3.1 Audio CNR Vendor ID (Word AP+000h) 49
6.1.3.2 Audio CNR Model ID (Word AP+002h) 49
6.1.3.3 CNR Multi-Channel Signature (Word AP+004h) 49
6.1.3.4 Audio Multi-Channel Model ID (Word AP+006h) 50
6.1.3.5 Codec x Output Physical Description 50
6.1.3.6 Codec x Input Physical Description 51
6.1.4 Modem Function PnP EEPROM Contents 52
6.1.4.1 Modem CNR Vendor ID (Word MP+000h) 52
6.1.4.2 Modem CNR Model ID (Word MP+002h) 53
6.1.5 USB Function PnP EEPROM Contents 53
6.1.5.1 USB Option Register (Word UP+00h) 53
6.1.5.2 USB Compliance Register (Word UP+002h) 54
6.1.5.3 USB Port Information Register (Word UP+04h) 54
6.1.6 SMBus Section EEPROM Map 56
6.1.6.1 SMBus Compliance Register (Word SP+000h) 56
6.1.6.2 SMBus Function Reserved (Words SP+002h to SP+004h) 57
6.1.7 LAN Section EEPROM Map 57
6.1.7.1 LAN Option Register (Word LP+000h) 57
6.1.7.2 LAN CNR Vendor ID (Word LP+002h) 58
6.1.7.3 LAN CNR Model ID (Word LP+004h) 58
6.1.7.4 LAN Compliance Register (Word LP+006h) 58
6.2 BIOS Requirements 58
6.2.1 CNR Presence Detection 60
6.2.2 Verifying PnP EEPROM Contents 60
6.2.3 Verifying CNR Version Compliance 60
6.2.4 Verifying AC '97 Version Compliance 60
6.2.5 Determining Interfaces Required 61
6.2.6 AC '97 Audio Codec Detection 61
6.2.6.1 AC '97 Codec Presence and Function Detection 61
6.2.6.1.1 "Fast Codec Start" Codec Detection Algorithm 62
6.2.6.1.2 Standard Codec Detection Algorithm 63
6.2.6.2 AC '97 Function Enabling and Disabling 67
6.2.6.3 AC '97 Plug-and-Play Information Extraction 73
6.2.6.3.1 BIOS Steps for Audio Function PnP with CDC_DN_ENAB# High 73
6.2.6.3.2 BIOS Steps for Audio Function PnP With CDC_DN_ENAB# Low 74
6.2.7 USB Compliance and Options 76
6.2.8 LAN Options 77
6.3 Software and Driver Requirements 78
6.3.1 Audio/Modem Interfacing for Call Progress 78

6.3.2 Multi-Channel Audio Upgrade Considerations 78
6.3.2.1 Multi-Channel Audio Plug-and-Play Signatures 78
6.3.2.1.1 Motherboard Multi-Channel Audio Signature Assignment 79
6.3.2.1.2 CNR Multi-Channel Audio Signature Assignment 79
6.3.2.2 Requirements for Multi-Channel Audio Upgrades 79
7 Motherboard Design Rules 79
7.1 ATX Family Form Factors Design Rules 79
8 Communication and Networking Riser Design Rules 80

List of Figures

Figure 1 – Communication and Networking Riser Concept 9
Figure 2 – Baseline Communication and Networking Riser 10
Figure 3 – Split Partition: Audio-Down with Modem and Phone-Line Network CNR 12
Figure 4 – Split Codec Partition: Audio-Down with 10/100 LAN CNR 12
Figure 5 – Split Codec Partition: Audio, Modem, and Phone-Line CNR 13
Figure 6 – Split Codec Partition: Audio/Modem Codec and Phone-Line Network CNR 13
Figure 7 – Split Codec Partition: Multi-channel Audio Upgrade CNR 14
Figure 8 – AC '97 Interface Hardware Acceleration Scalability 15
Figure 9 – Type A CNR Connector pin-out for ATX Family Form Factors 22
Figure 10 – Type B CNR Connector pin-out for ATX Family Form Factors 23
Figure 11 – AC97_SDATA_IN2 Circuit 25
Figure 12 – CDC_DN_ENAB# & Address Control Block 26
Figure 13 – CDC_DN_ENAB# control and Codec Address circuitry 27
Figure 14 – CNR Connector location for the ATX Family Form Factors 33
Figure 15 – Dimensions for a Standard Full Height CNR board 34
Figure 16 – Dimensions for a Standard Full Height CNR board with I/O bracket 35
Figure 17 – Dimensions for Low Profile CNR board 36
Figure 18 – Dimensions for Low Profile CNR board with I/O bracket 37
Figure 19 – Full and Low Profile CNR edge card dimensions 37
Figure 20 – CNR Card Edge Connector Contacts 38
Figure 21 – CNR Card Edge Connector Bevel 38
Figure 22 – CNR Type A and Type B Connector Dimensions 39
Figure 23 – CNR Type A and Type B Connector Layout Recommendation 40
Figure 24 – Standard Full Height CNR I/O Bracket Dimensions 42
Figure 25 – Low Profile CNR I/O Bracket Dimensions 43
Figure 26 – SMBus Address Strapping 45
Figure 27 – BIOS Functional Flowchart for CNR Plug-and-Play Support 59
Figure 28 – AC '97 Presence and Function Detection Flowchart for Fast Codec Start Detection 65
Figure 29 – AC '97 Codec Presence and Function Detection Flowchart for Standard Codec Detection 66
Figure 30 – AC '97 Enabling and Disabling Flowchart 71
Figure 31 – AC '97 Enabling and Disabling Flowchart continued 72
Figure 32 – AC ’97 Plug-and-Play Information Flowchart 73

List of Tables

Table 1 – CNR Connector signals for the AC ‘97 Interface 16
Table 2 – CNR Connector signals for the USB 2.0 Interface 17
Table 3 – CNR Connector signals for the SMBus (PnP) EEPROM Interfaces 17
Table 4 – CNR Connector signals for the LAN Interface Microwire EEPROM Interfaces 17
Table 5 – CNR Connector power supplies and ground returns 18
Table 6 – CNR Connector Signals for the Eight-pin PLC LAN Interface 19
Table 7 – CNR Type A Connector Reserved Signals 20
Table 8 – CNR Connector Signals for the Seventeen-pin MII LAN Interface 20
Table 9 – CNR Type B Connector reserved signals 21
Table 10 – CDC_DN_ENAB# Pull-up Resistor Values 25
Table 11 – Communication and Networking Riser Electrical Specification 28
Table 12 – Recommended Power Distribution for the AC ’97 Interface Signals 30
Table 13 – Type A and Type B Connector Manufacturers 39
Table 14 – Type A and Type Connector Physical Requirements 40
Table 15 – Type A and Type B Connector Mechanical Performance Requirements 40
Table 16 – Type A and Type B Connector Electrical Performance Requirements 41
Table 17 – Type A and Type B Connector Environmental Performance Requirements 41
Table 18 – CNR EEPROM Master Configuration Space Register Map 46
Table 19 – PnP EEPROM Size Register Value Assignments 46
Table 20 – CNR Specification Version Mapping in PnP EEPROM 47
Table 21 – AC ‘97 Compliance Register Contents 47
Table 22 – CNR EEPROM Audio Section Register Map 49
Table 23 – Jack Sense Connection 51
Table 24 – S/P-DIF Output Configuration 51
Table 25 – S/P-DIF Input Configuration 52
Table 26 – CNR EEPROM Modem Section Register Map 52
Table 27 – CNR EEPROM USB Section Register Map 53
Table 28 – UPCx Bits and Number of Ports 53
Table 29 – USB Version to SPD Bit values 54
Table 30 – USB Compliance Register Contents 54
Table 31 – USB Port Information Register 55
Table 32 – USB Port Information Register- Bit Definitions 55
Table 33 – USB Port Mapping Example – 1 Port 55
Table 34 – USB Port Mapping Example – 2:4 Ports 56
Table 35 – CNR EEPROM SMBus Section Register Map 56
Table 36 – SMBus Compliance Register Contents 57
Table 37 – CNR EEPROM LAN Section Register Map 57
Table 38 – LAN Compliance Register Contents for eight-pin LAN Interface 58
Table 39 – LAN Compliance Register Contents for seventeen-pin LAN Interface (MII) 58
Table 40 – AC '97 Codec Function Mapping to Codec Register Value 64
Table 41 – Supported Configurations (with CNR installed) 69
Table 42 – Unsupported Configurations (with CNR installed) 70

Revision History

Version 1.0 - February 7, 2000

- Initial release.

Version 1.1 - October 18, 2000

- Added support for Universal Serial Bus (USB) Revision 2.0
- Incorporated Engineering Change Notices number 1 through 12. The ECN titles are provided below:
  ECN #1: "BIOS Action for LAN Interface Mismatch"
  ECN #2: "BIOS Action for Illegal AC '97 Codec Plug-and-Play Combinations"
  ECN #3: "SVID/SID Persistence During Power Down States"
  ECN #4: "BIOS Messages – Pause and Acknowledge"
  ECN #5: "CDC_DN_ENAB# Pull-up Resistor Change"
  ECN #6: "Increase Motherboard Keep-out Area"
  ECN #7: "Plug-and-Play EEPROM Compliance and Power Supply Requirement"
  ECN #8: "Clarification on AC '97 Codec Disabling and Demotion Requirements"
  ECN #9: "Update CNR Interface Version Compliance Algorithms"
  ECN #10: "Definition of CNR Pull-up Voltage on CDC_DN_ENAB# pin"
  ECN #11: "Optional Main Board AC '97 Codec Disable Circuit"
  ECN #12: "USB_OC# Pin Voltage Definitions"

Version 1.2 - November 8, 2001

- Added support to AC '97 sections to support three AC '97 codecs, including updating disable and demotion rules, and BIOS AC '97 detection and function algorithms.
- Added additional register definition to the USB section of the CNR PnP EEPROM, in order to better define the USB port usage.
- Added additional register definition to the Audio section of the CNR PnP EEPROM for AC '97 Codec supported input/output jacks.

1 Introduction

The Communication and Networking Riser (CNR) Specification defines a hardware scalable Original Equipment Manufacturer (OEM) motherboard riser and interface that supports the audio, modem, Universal Serial Bus (USB), and local area network (LAN) interfaces of core logic chipsets. The main objective of this specification is to reduce the baseline implementation cost of features that are widely used in the "Connected PC", while also addressing specific functional constraints of today's audio, modem, USB, and LAN subsystems.

PC users' demand for feature-rich PCs, combined with the industry's current trend towards lower cost, mandates higher levels of integration at all levels of the PC platform. Motherboard integration of communication technologies has been problematic to date, for a variety of reasons, including FCC and international telecom certification processes, motherboard space, and other manufacturer specific requirements.

Motherboard integration of the audio, modem, USB, and LAN subsystems is also problematic, due to the potential for increased noise, which in-turn degrades the performance of each system. The CNR specifically addresses these problems by physically separating these noise-sensitive systems from the noisy environment of the motherboard.

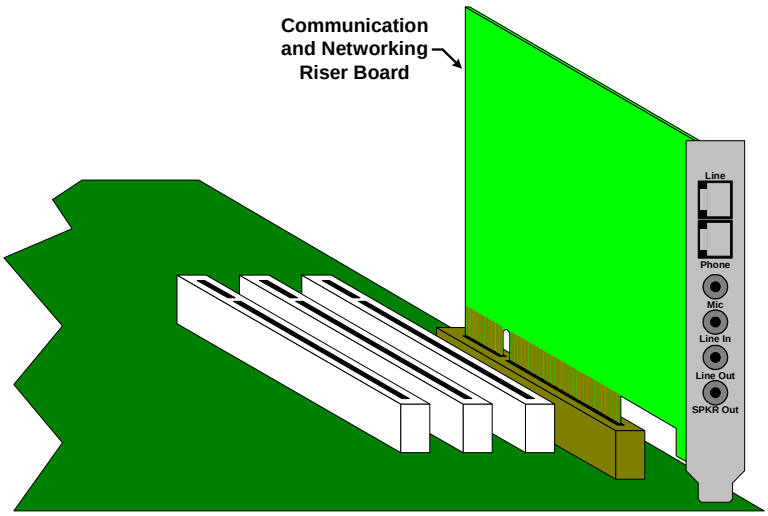


Figure 1 – Communication and Networking Riser Concept

With a standard riser solution, as defined in this specification, the system manufacturer is free to implement the audio, modem, USB, and/or LAN subsystems at a lower bill of materials (BOM) cost than would be possible by deploying the same functions in industry-standard expansion slots or in a proprietary method. With the added flexibility that hardware scalability brings, a system manufacturer has several motherboard acceleration options available, all stemming from the baseline CNR interface.

Note: ***The CNR Specification does <u>NOT</u> define an aftermarket standard I/O expansion slot. This specification defines a system manufacturer or system integrator, motherboard-only, riser interface, which is to be fully configured prior to the initial shipment of the system. Standard I/O expansion slots, such as those supported by the PCI bus architecture, are intended to continue serving as the retail channel upgrade medium.***

The CNR Specification describes interfaces that can support multi-channel audio, V.90 analog modem, phone-line based networking, 10/100 Ethernet based networking, as well as expandability for future technologies.

This specification defines the CNR architecture, required electrical characteristics of the riser interface, mechanical, and thermal requirements. The CNR Specification supports the ATX, microATX, and FlexATX form factors (now referred to as ATX family form factors). This specification does **<u>NOT</u>** support the NLX form factor. Future versions or revisions of this specification may support additional form factors, as they become available and/or widely used by the PC industry.

Note: ***Questions regarding the Communication and Networking Riser Specification may be sent, via e-mail, to:***

cnr.support@intel.com

1.1 Related Documents

- "Audio Codec '97 Component Specification, Revision 2.1" (http://developer.intel.com/pc-supp/platform/ac97)
- "ACPI (Advanced Configuration and Power Interface) Specification" (http://www.acpi.info/index.html)
- "PCI Bus Power Management Interface Specification, Revision 1.1" (http://www.pcisig.com)
- "Instantly Available PC Power Management Design Guide" (http://developer.intel.com/design/power/pcpower.htm)
- "Universal Serial Bus Specification, Revision 2.0" (http://www.usb.org)
- "System Management Bus Specification, Revision 1.1" (http://www.smbus.org/)
- "PCI Local Bus Specification, Revision 2.2" (http://www.pcisig.com)
- "ATX Specification, Version 2.03" (http://www.formfactors.org/)
- "microATX Motherboard Interface Specification, Version 1.0" (http://www.formfactors.org/)
- "Accelerated Graphics Port (AGP) Specification, Revision 2.0" (http://developer.intel.com/technology/agp)

2 Architectural Overview

The following sections provide an overview of the baseline architecture along with descriptions of both supported and unsupported CNR configurations. Included in the following sections are different configurations of audio, modem, and LAN subsections, either up on the CNR board or physically soldered down on the motherboard printed circuit board (PCB). In addition, a model for hardware scalability of the audio subsystem is described.

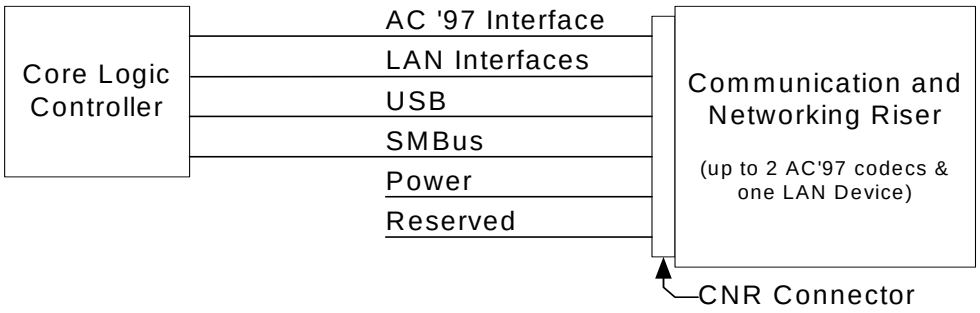


Figure 2 – Baseline Communication and Networking Riser

2.1 Baseline Riser Architecture

Figure 2 illustrates the baseline architecture of a motherboard with support for a CNR Connector and board.

The backbone of the CNR interface consists of an AC ’97-compliant AC-Link, a LAN interface, an SMBus* interface, and a USB* 2.0 interface.

- **AC ’97 Interface:** Used to support audio and/or modem functions on the CNR board.
- **LAN Interfaces:** Provides one of two LAN interfaces for networking functions. Interfaces include an eight-pin interface for use with platform LAN connection (PLC) based devices, and a seventeen-pin interface for Media Independent Interface (MII) based devices (commonly referred to as a PHY).
- **USB Interface:** Used to support technologies or functions that are being implemented on a USB 2.0 Interface.
- **SMBus Interface:** Used specifically to provide Plug-and-Play functionality for the CNR board.

Additional signals on the CNR interface support:

- **Power:** Signals that are required to support power management, as well as the main power supplies to operate the CNR board silicon and support circuitry.
- **Reserved:** Signals that will be used to support future features and power supply requirements.

2.2 Audio, Modem, and Networking Partitioning

In the CNR architecture there are several different ways to partition the audio, modem, USB, and LAN subsystems. Ultimately, the decision as to how the partitioning occurs is up to the system integrator. This decision is further based on the system integrators’ needs for feature set and management of the various motherboard and CNR designs.

When selecting a partition, the system integrator must consider the user model for implementing audio, modem, USB, and LAN technologies. In the consumer and small business environments, the networking technology may be a phone line based network connected to an RJ-11 jack. This network, along with a traditional analog modem provides for convergence of the network and communication systems at the RJ-11 jack. Thus, the system integrator should consider how the user would connect the analog modem and home phone networks to the phone wiring in the building. In the corporate environment, the analog modem will most likely not exist, but instead a traditional 10/100 Ethernet based LAN would be implemented. In this case, the LAN usually connects to an RJ-45 connector, thus simplifying the user model.

The following section shows some of the possible partitioning options for the audio, modem, and LAN subsystems between the motherboard PCB and the CNR board. Note that these are only a few examples of the numerous partitioning scenarios.

2.2.1 Audio-Down with Modem and Phone-Line Network-Up Partition

The configuration in Figure 3 deploys the audio codec on the motherboard with both the modem codec and phone-line PLC/PHY device on the CNR board. This configuration provides the flexibility to design the motherboard independent of the CNR board, thus decoupling government regulatory certification of the riser from the motherboard. In addition, a single motherboard design can then be used in a system targeted at either the consumer or commercial markets, depending on the CNR board installed.

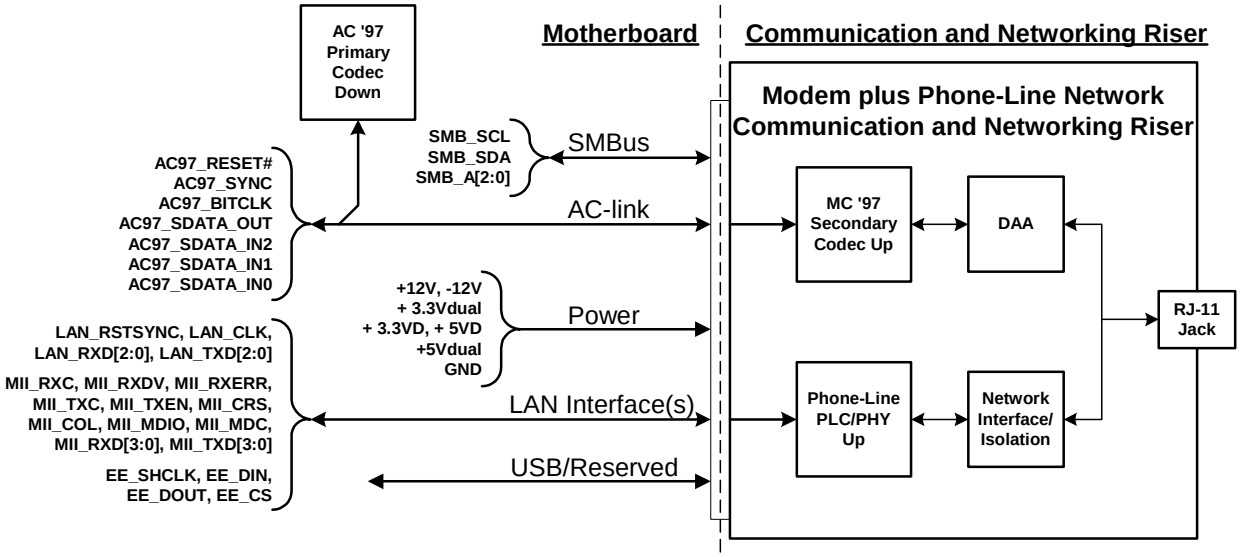


Figure 3 – Split Partition: Audio-Down with Modem and Phone-Line Network CNR

2.2.2 Audio-Down with 10/100 LAN-Up Partition

The configuration in Figure 4 deploys the audio codec on the motherboard with the 10/100 Ethernet based LAN PLC/PHY device on the CNR board. This configuration also provides the flexibility of allowing a single motherboard to be used in a system targeted at either the consumer or commercial markets depending on the CNR board installed.

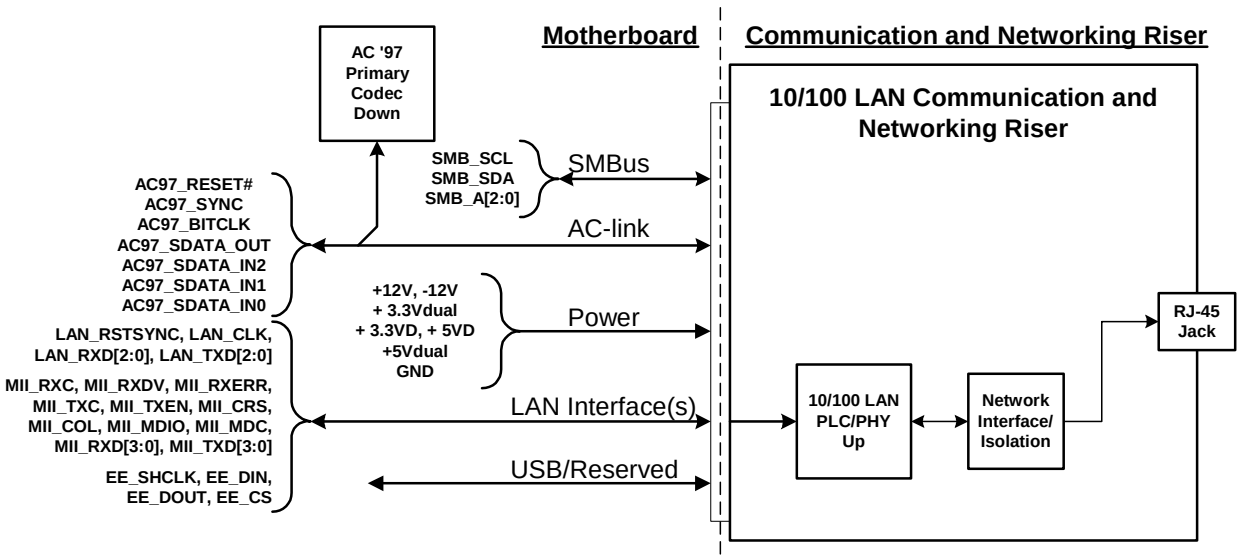


Figure 4 – Split Codec Partition: Audio-Down with 10/100 LAN CNR

2.2.3 Audio-Up, Modem-Up, and Phone-Line Network-Up Partition

The configuration in Figure 5 places the audio, modem, and phone-line networking components on the CNR board. In this configuration, the motherboard could be a minimally featured board, in which the system manufacturer has the opportunity to populate a fully featured riser, or a subset of a fully featured riser.

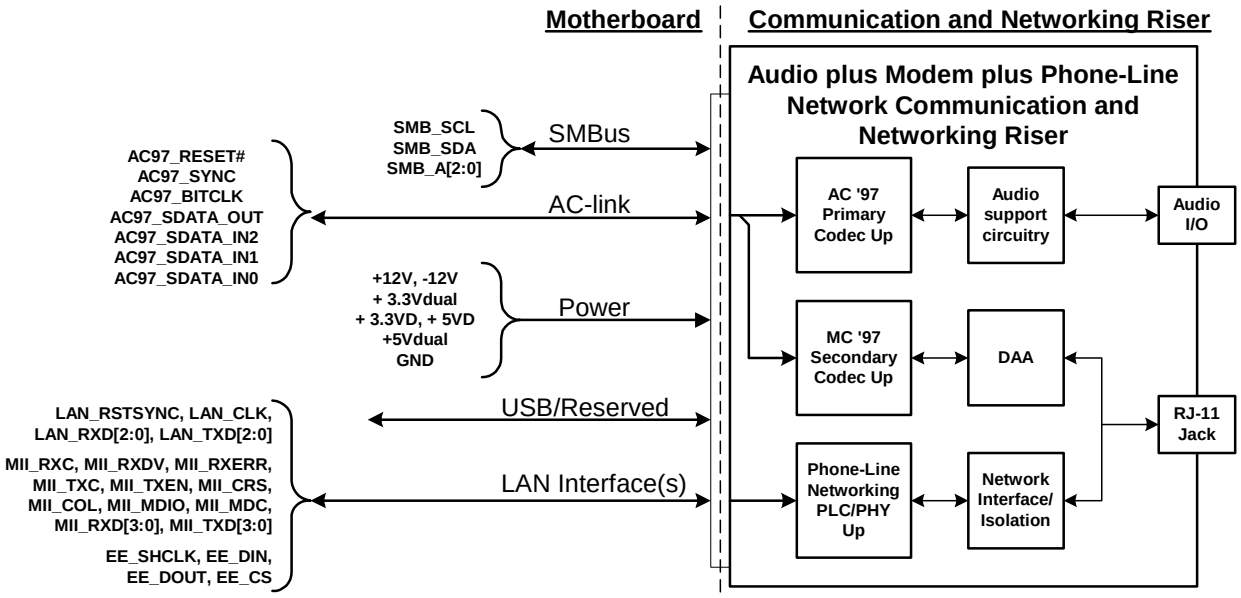


Figure 5 – Split Codec Partition: Audio, Modem, and Phone-Line CNR

2.2.4 Combined Audio/Modem-Up with Phone-Line Network-Up Partition

The configuration in Figure 6 employs an AMC ’97 combined audio/modem codec with a Phone-line PLC/PHY device. This configuration introduces a higher level of integration and potentially a lower implementation cost for the audio, modem, and LAN subsystems.

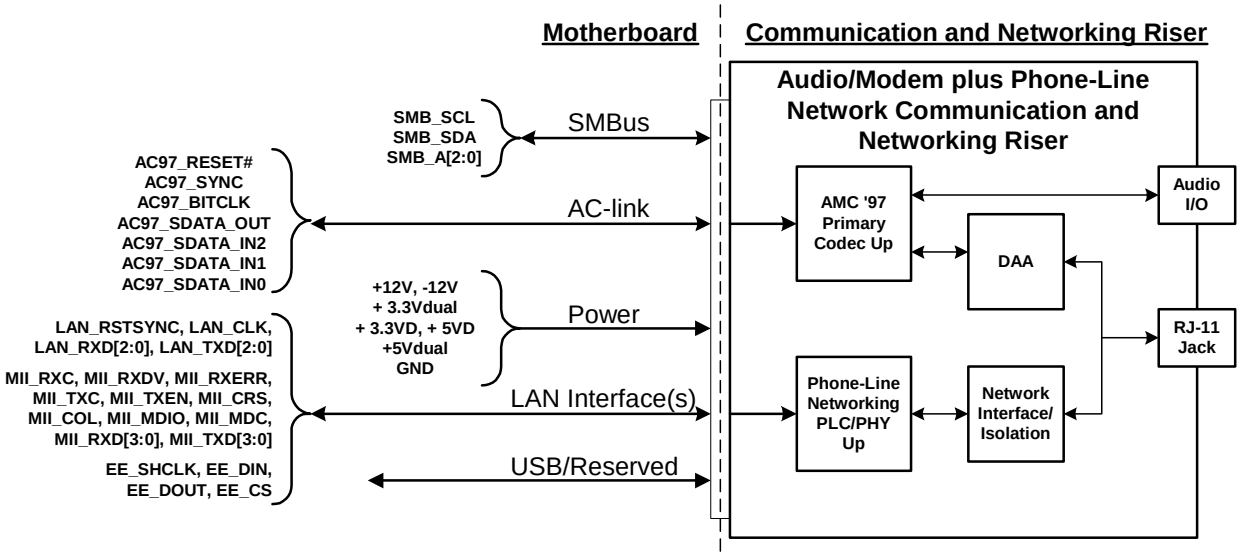


Figure 6 – Split Codec Partition: Audio/Modem Codec and Phone-Line Network CNR

2.2.5 Upgrade To Multi-Channel Audio Partition

The configuration in Figure 7 deploys the audio codec on the motherboard with an additional audio codec on the CNR board. This configuration demonstrates the flexibility of allowing a single motherboard to accept an audio upgrade from two-channel audio (stereo) to multi-channel audio (four or six channels) by simply installing a CNR board.

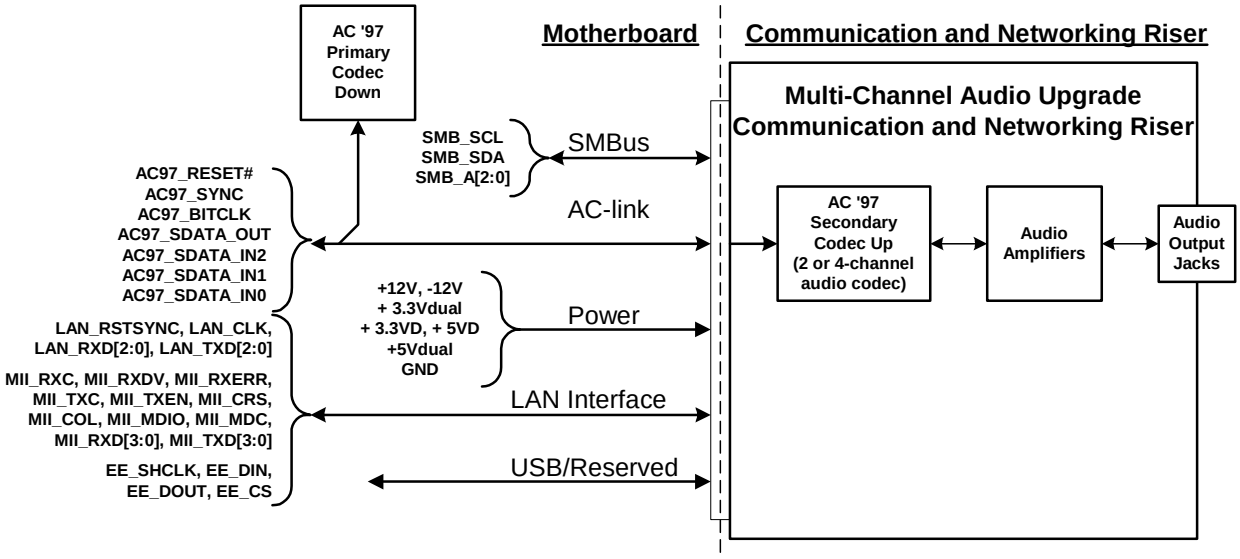


Figure 7 – Split Codec Partition: Multi-channel Audio Upgrade CNR

2.3 Unsupported Configurations

All CNR configurations **must** be limited to a single primary AC '97 codec, with up to two additional secondary AC '97 codecs. Any CNR configuration with more than a single primary AC '97 codec is unsupported, and will likely result in improper system behavior and/or component damage. The required support for CDC_DN_ENAB# pin functionality ensures that only one primary codec can ever reside on the AC-link for a given CNR configuration. Refer to Table 1 (and Section 3.2.1) for details on implementing the functionality of the CDC_DN_ENAB# signal.

As the LAN Interfaces are designed to support a single PLC/PHY device, the motherboard **must not** be designed to simultaneously support both a PLC/PHY device on the motherboard and additional PLC/PHY devices on the CNR board. In addition, the CNR must not be designed to simultaneously support more than a single PLC/PHY device. Failure to follow these requirements may cause damage to devices attached to the LAN Interfaces, due to contention on the signals of the LAN interfaces. Detailed information on the LAN interfaces can be found in the appropriate Core Logic Specification.

2.4 Audio Hardware Scalability Model

The baseline CNR architecture supports an “in-line” hardware acceleration model for both the audio and modem subsystems. In-line acceleration is the optimum form of hardware acceleration. An in-line accelerator is situated between the source of the pre-processed data and the final destination for the processed data. An accelerator fetches pre-processed data directly from main memory, processes it, and then passes it directly to the codec(s) via the AC-Link.

A system manufacturer may design their motherboard with provisions to support a baseline controller with a stuffing option for a PCI-based hardware-accelerated controller. The following figure illustrates the CNR architecture’s support for hardware acceleration.

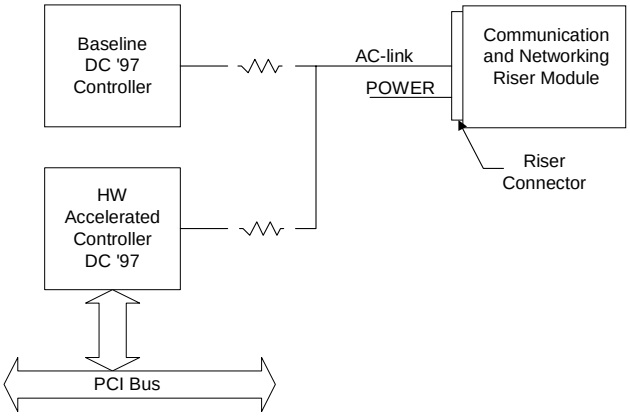


Figure 8 – AC '97 Interface Hardware Acceleration Scalability

The CNR architecture's hardware scalability enables a system manufacturer to choose which controller to deploy as the master of the AC-Link. The "selectable" controller arrangement, utilizing series resistors[1], provides a simple method for an accelerated controller to fully replace the baseline controller. From the perspective of the CNR board, an AC '97-compliant controller is mastering the AC-Link. The CNR board cannot discern whether it is being mastered by the baseline or accelerated controller.

At motherboard design time, a system manufacturer may design in provisions for a certain hardware-accelerated controller solution which, when mated with the capabilities of compatible CNR board, creates a range of possible motherboard solutions.

Note: ***The CNR architecture does not support after-market audio, modem, or LAN upgrades by only changing the CNR board. Once a system manufacturer or system integrator has shipped a system into the field, regardless of the motherboard/CNR board configuration capabilities, all future audio, modem, and/or LAN upgrades must be brought into the system via industry-standard expansion vehicles such as a PCI slot, Universal Serial Bus (USB) port, or through OEM upgrades.***

3 Electrical Requirements

3.1 Signal Names and Pin-out

The CNR connector is the interface between the motherboard and the CNR board. The connector provides all of the necessary signals to support several different configurations of audio, modem, USB, and/or LAN subsystems in the system, as previously mentioned.

The CNR provides support for two different LAN interfaces; an eight-pin PLC interface or a seventeen-pin interface (commonly known as MII or IEEE 802.3u). Support for these two interfaces is provided on two separate CNR connector pin-outs. The pin-out for the eight-pin LAN interface is defined in the Type A CNR connector (Figure 9), while the pin-out for the seventeen-pin LAN interface is defined in the Type B CNR connector (Figure 10).

3.1.1 ATX Family Form Factors Signals

The following sections provide detailed information regarding the signals on both the Type A and Type B CNR connectors, including signal names, descriptions, and pin number assignments for the ATX Family Form Factors.

For the following sections the signals designated as Input are sourced from the motherboard, while signals designated as Output are sourced from the CNR board. Those signals defined as In/Out may be sourced from either the motherboard or the CNR board.

[1] The resistor values should be chosen for the best signal quality. The baseline controller's resistor values may not necessarily be the same value as those of the accelerated controller.

3.1.1.1 Common Signal Description for Type A and Type B Connectors

This section describes the signals, which are common to both the Type A and Type B CNR connectors for the ATX Family Form Factors. The following tables are broken into sections to describe the signals on an interface-by-interface basis.

Note: It is strongly recommended that all of the signals listed in the following tables, if implemented on and not in use by devices on the motherboard, be routed to, and implemented on the Type A and Type B CNR connector.

Signal Name	Type	Pin Number	Signal Description
AC97_BITCLK	In/Out	B30	Serial data clock from primary codec to AC '97 Controller and any non-primary codecs. The nominal frequency of this signal is 12.288 MHz. For detailed information, refer to the current version of the AC '97 Component Specification. AC97_BITCLK is an output from a primary codec and an input to non-primary codecs. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
AC97_SYNC	Input	B28	Synchronization pulse from an AC '97-compliant controller to all of the AC '97-compliant codecs on the AC link. This signal is nominally a 1.3 μS wide pulse, which is used to synchronize the AC link. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
AC97_SDATA_OUT	Input	B29	AC '97 serial data from an AC '97-compliant controller to all of the AC '97-compliant codecs on the link. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
AC97_SDATA_IN0	Output	A29	AC '97 serial data from a primary AC '97-compliant codec to an AC '97-compliant Controller. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
AC97_SDATA_IN1	Output	A28	AC '97 serial data from an AC '97-compliant codec (primary or secondary) to an AC '97-compliant Controller. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
AC97_SDATA_IN2	Output	A27	AC '97 serial data from an AC '97-compliant codec (primary or secondary) to an AC '97-compliant Controller. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification. This signal may not be present in all AC '97 Controllers. Refer to Section 3.3.1.2 for information on designing with AC '97 Controllers supporting two or three codecs.
AC97_RESET#	Input	A26	Active low AC '97 link reset signal. For detailed information, refer to the current version of the AC '97 Component Specification. The reset state of this signal must meet the requirements of the current version of the AC '97 Component Specification.
CDC_DN_ENAB#	In/Out	B26	CDC_DN_ENAB# indicates whether the motherboard or the CNR is in control, or mastering, the AC '97 interface attached to the CNR Connector. When at a logic low level, the CDC_DN_ENAB# signal indicates that the primary codec on the motherboard is active and controlling the AC '97 Interface. In addition, the CNR will, when CDC_DN_ENAB# is low, demote its codecs to the next available address and to the next available SDATA_IN signal. See Section 3.2.1 for more details on the implementation of the CDC_DN_ENAB# signal. When at a logic high level, the CDC_DN_ENAB# signal indicates that a primary codec on the CNR is taking control of the AC '97 Interface. In addition, the motherboard will, when CDC_DN_ENAB# is high, disable all of its codecs. See Section 3.2.1 for more details of how to implement the CDC_DN_ENAB# signal. CDC_DN_ENAB# is an input to the AC '97 Controller, but should be connected to a GPIO pin for debug purposes.

Table 1 – CNR Connector signals for the AC '97 Interface

Signal Name	Type	Pin Number	Signal Description
USB+	In/Out	A13	Positive side of the differential USB data signal. For more information, refer to the Universal Serial Bus Specification. The state of this signal during reset must meet the Universal Serial Bus Specification.
USB-	In/Out	A15	Negative side of the differential USB data signal. For more information, refer to the Universal Serial Bus Specification. The state of this signal during reset must meet the Universal Serial Bus Specification.
USB_OC#	Output	B16	USB bus over-current signal. For more information, refer to the Universal Serial Bus Specification 2.0. The state of this signal during reset must meet the Universal Serial Bus Specification 2.0.

Table 2 – CNR Connector signals for the USB 2.0 Interface

Signal Name	Type	Pin Number	Signal Description
SMB_SCL	Input	B25	Serial clock line from the SMBus master to SMBus slave device(s) on the CNR board. For detailed information on this signal, refer to the current version of the System Management Bus Specification. The reset state of this signal must meet the current version of the System Management Bus Specification.
SMB_SDA	In/Out	A25	Bi-directional serial data line between the SMBus master to SMBus slave device(s) on the CNR board. For detailed information on this signal, refer to the current version of the System Management Bus Specification. The reset state of this signal must meet the current version of the System Management Bus Specification.
SMB_A2	Input	A24	This signal is bit 2 (MSB) of the 3-bit address of the SMBus EEPROM on the CNR board. Refer to Section 6.1.1.2 for detailed information on the connectivity of this signal. The state of this signal during reset must be the SMBus address for the CNR board.
SMB_A1	Input	A23	This signal is bit 1 of the 3-bit address of the SMBus EEPROM on the CNR board. Refer to Section 6.1.1.2 for detailed information on the connectivity of this signal. The state of this signal during reset must be the SMBus address for the CNR board.
SMB_A0	Input	B24	This signal is bit 0 (LSB) of the 3-bit address of the SMBus EEPROM on the CNR board. Refer to Section 6.1.1.2 for detailed information on the connectivity of this signal. The state of this signal during reset must be the SMBus address of the CNR board.

Table 3 – CNR Connector signals for the SMBus (PnP) EEPROM Interfaces

Signal Name	Type	Pin Number	Signal Description
EE_SHCLK	Input	B22	This signal is the serial clock signal from the core logic MAC Microwire* interface to the Microwire EEPROM (which stores MAC/PLC/PHY specific information) on the CNR board.
EE_DIN	Input	A21	This signal carries serial data from the core logic MAC Microwire* interface to the Microwire EEPROM (which stores MAC/PLC/PHY specific information) on the CNR board. The EE_DIN signal on the CNR connector must be connected to the DIN pin on the Microwire EEPROM.
EE_DOUT	Output	B21	This signal carries serial data from the Microwire EEPROM (which stores MAC/PLC/PHY specific information) on the CNR board to the core logic MAC Microwire* interface. The EE_DOUT signal on the CNR connector must be connected to the DOUT pin on the Microwire EEPROM.
EE_CS	Input	A22	The CNR board uses this signal to enable the serial EEPROM devices on the CNR board. When EE_CS is high (one) the Microwire EEPROM (for the LAN Interface) becomes active. When EE_CS is low (zero) the EEPROM is inactive. The resting state of this signal is low (zero). The state of this signal during reset must be low (zero).

Table 4 – CNR Connector signals for the LAN Interface Microwire EEPROM Interfaces

Signal Name	Type	Pin Number	Signal Description
+12V	Supply	A16	Positive 12-volt main power supply
-12V	Supply	B18	Negative 12-volt main power supply
+5VD	Supply	A19	Positive 5-volt main digital power supply
+3.3VD	Supply	B19	Positive 3.3-volt main digital power supply
+5Vdual	Supply	B15	Positive 5-volt main/standby power supply (can be used for USB power). +5Vdual supply provides full-rated power capacity during working or full-on state, and a limited power capacity during sleep or suspended states. When a +5Vdual supply is not available, this pin **must** be connected to a +5 volt standby power source. This signal **must not** be connected to a +5VD, as doing so eliminates the possibility of deep-sleep wake capabilities.
+3.3Vdual	Supply	A18	Positive 3.3-volt main/standby power supply. +3.3Vdual supply provides full-rated power capacity during working or full-on state, and a limited power capacity during sleep or suspended states. When +3.3Vdual is not available, this pin **must** be connected to a +3.3-volt standby power source. This signal **must not** be connected to a +3.3VD, as doing so eliminates the possibility of deep-sleep wake capabilities.
Ground	Ground	A3	Power supply and signal ground return path.
Ground	Ground	A6	Power supply and signal ground return path.
Ground	Ground	A9	Power supply and signal ground return path.
Ground	Ground	A14	Power supply and signal ground return path.
Ground	Ground	A17	Power supply and signal ground return path.
Ground	Ground	A20	Power supply and signal ground return path.
Ground	Ground	A30	Power supply and signal ground return path.
Ground	Ground	B4	Power supply and signal ground return path.
Ground	Ground	B7	Power supply and signal ground return path.
Ground	Ground	B10	Power supply and signal ground return path.
Ground	Ground	B13	Power supply and signal ground return path.
Ground	Ground	B17	Power supply and signal ground return path.
Ground	Ground	B20	Power supply and signal ground return path.
Ground	Ground	B23	Power supply and signal ground return path.
Ground	Ground	B27	Power supply and signal ground return path.

Table 5 – CNR Connector power supplies and ground returns

3.1.1.2 Type A Connector Unique Signal Description for the ATX Family Form Factors

This section describes the signals unique to the Type A CNR connector for the ATX Family Form Factors.

Note: ***It is <u>strongly</u> recommended that all of the signals listed in the following tables, if implemented on and not in use by devices on the motherboard, be routed to, and implemented on the Type A CNR connector.***

Signal Name	Type	Pin Number	Signal Description
LAN_CLK	Output	A10	Data clock from a LAN Interface compliant PLC to the Media Access Controller (MAC). The nominal frequency of this signal determines the data transfer rate between the PLC and the MAC. For detailed information, refer to the current version Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_RSTSYNC	Input	B9	This is a dual function pin that provides either a reset pulse or a synchronization pulse from the MAC to the LAN Interface compliant PLC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_TXD2	Input	A7	Bit 2 (MSB) of the 3-bit data bus transferring data from the MAC to the LAN Interface compliant PLC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_TXD1	Input	B8	Bit 1 of the 3-bit data bus transferring data from the MAC to the LAN Interface compliant PLC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_TXD0	Input	A8	Bit 0 (LSB) of the 3-bit data bus transferring data from the MAC to the LAN Interface compliant PLC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_RXD2	Output	B11	Bit 2 (MSB) of the 3-bit data bus transferring data from the LAN Interface compliant PLC to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_RXD1	Output	A11	Bit 1 of the 3-bit data bus transferring data from the LAN Interface compliant PLC device to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
LAN_RXD0	Output	B12	Bit 0 (LSB) of the 3-bit data bus transferring data from the LAN Interface compliant PLC device to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.

Table 6 – CNR Connector Signals for the Eight-pin PLC LAN Interface

Signal Name	Type	Pin Number	Signal Description
Reserved	N/A	A1	Reserved
Reserved	N/A	A2	Reserved
Reserved	N/A	A4	Reserved
Reserved	N/A	A5	Reserved
Reserved	N/A	A12	Reserved
Reserved	N/A	B1	Reserved
Reserved	N/A	B2	Reserved
Reserved	N/A	B3	Reserved
Reserved	N/A	B5	Reserved
Reserved	N/A	B6	Reserved
Reserved	N/A	B14	Reserved

Table 7 – CNR Type A Connector Reserved Signals

3.1.1.3 Type B Connector Unique Signal Description for the ATX Family Form Factors

This section describes the signals unique to the Type B CNR connector for the ATX Family Form Factors.

Note: ***It is <u>strongly</u> recommended that all of the signals listed in the following tables, if implemented on and not in use by devices on the motherboard, be routed to, and implemented on the Type B CNR Connector.***

Signal Name	Type	Pin Number	Signal Description
MII_RXD3	Output	A12	Bit 3 (MSB) of the 4-bit data bus transferring data from the MII compliant PHY to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXD2	Output	B11	Bit 2 of the 4-bit data bus transferring data from the MII compliant PHY to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXD1	Output	A11	Bit 1 of the 4-bit data bus transferring data from the MII compliant PHY to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXD0	Output	B12	Bit 0 (LSB) of the 4-bit data bus transferring data from the MII compliant PHY to the MAC. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXC	Output	A5	Data clock from a MII Interface compliant PHY to the MAC. For detailed information, refer to the current version Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXDV	Output	A4	Receive data valid signal from the MII compliant PHY to the MAC. This signal indicates that valid data is available on the MII_RXD[3:0] signals. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_RXERR	Output	B5	Receive error signal from the MII compliant PHY to the MAC. This signal indicates that an error has occurred during frame reception. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.

Table 8 – CNR Connector Signals for the Seventeen-pin MII LAN Interface

Signal Name	Type	Pin Number	Signal Description
MII_TXD3	Input	B6	Bit 3 (MSB) of the 4-bit data bus transferring data from the MAC to the MII compliant PHY. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_TXD2	Input	A7	Bit 2 of the 4-bit data bus transferring data from the MAC to the MII compliant PHY. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_TXD1	Input	B8	Bit 1 of the 4-bit data bus transferring data from the MAC to the MII compliant PHY. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_TXD0	Input	A8	Bit 0 (LSB) of the 4-bit data bus transferring data from the MAC to the MII compliant PHY. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_TXC	Output	B3	Data clock from the MAC to the MII compliant PHY. For detailed information, refer to the current version Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_TXEN	Input	B9	Transmit enable signal from the MAC to the MII compliant PHY. This signal indicates that the available on the MII_TXD[3:0] signals can be placed on the LAN wire. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_CRS	Output	A2	Carrier sense signal from the MII compliant PHY to the MAC. This signal indicates that there is traffic on the LAN wire. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_COL	Output	B2	Collision detect signal from the MII compliant PHY to the MAC. This signal indicates that a collision has occurred on the LAN wire. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_MDIO	In/Out	B1	Management data input/output signal between the Management Data Controller and the MII compliant PHY. This signal is used to carry bi-directional data for control and status registers. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.
MII_MDC	Input	A1	Management data clock signal from Management Data Controller to the MII compliant PHY. For detailed information on this signal, refer to the current version of the Core Logic Design Specification and the IEEE 802.3u Specification. The reset state of this signal must meet the requirements set forth in the current version of the Core Logic Design Specification.

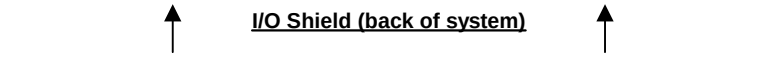
Table 8 (cont.) – CNR Connector Signals for the Seventeen-pin MII LAN Interface

Signal Name	Type	Pin Number	Signal Description
Reserved	N/A	A10	Reserved
Reserved	N/A	B14	Reserved

Table 9 – CNR Type B Connector reserved signals

3.1.1.4 Type A Connector Pin Assignments

Figure 9 provides the pin-out for the Type A CNR connector, using the eight-pin PLC LAN interface.

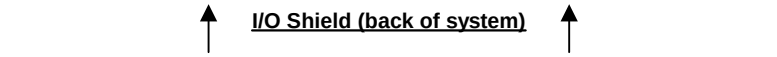


B1	RESERVED	RESERVED	A1
B2	RESERVED	RESERVED	A2
B3	RESERVED	GND	A3
B4	GND	RESERVED	A4
B5	RESERVED	RESERVED	A5
B6	RESERVED	GND	A6
B7	GND	LAN_TXD2	A7
B8	LAN_TXD1	LAN_TXD0	A8
B9	LAN_RSTSYNC	GND	A9
B10	GND	LAN_CLK	A10
B11	LAN_RXD2	LAN_RXD1	A11
B12	LAN_RXD0	RESERVED	A12
B13	GND	USB+	A13
B14	RESERVED	GND	A14
B15	+5Vdual	USB-	A15
B16	USB_OC#	+12V	A16
B17	GND	GND	A17
B18	-12V	+3.3Vdual	A18
B19	+3.3VD	+5VD	A19
	KEY	KEY	
B20	GND	GND	A20
B21	EE_DOUT	EE_DIN	A21
B22	EE_SHCLK	EE_CS	A22
B23	GND	SMB_A1	A23
B24	SMB_A0	SMB_A2	A24
B25	SMB_SCL	SMB_SDA	A25
B26	CDC_DN_ENAB#	AC97_RESET#	A26
B27	GND	AC97_SDATA_IN2	A27
B28	AC97_SYNC	AC97_SDATA_IN1	A28
B29	AC97_SDATA_OUT	AC97_SDATA_IN0	A29
B30	AC97_BITCLK	GND	A30

Figure 9 – Type A CNR Connector pin-out for ATX Family Form Factors

3.1.1.5 Type B Connector Pin Assignments

Figure 10 provides the pin-out for the Type B CNR connector, using the seventeen-pin MII LAN interface.



B1	MII_MDIO	MII_MDC	A1
B2	MII_COL	MII_CRS	A2
B3	MII_TXC	GND	A3
B4	GND	MII_RXDV	A4
B5	MII_RXERR	MII_RXC	A5
B6	MII_TXD3	GND	A6
B7	GND	MII_TXD2	A7
B8	MII_TXD1	MII_TXD0	A8
B9	MII_TXEN	GND	A9
B10	GND	RESERVED	A10
B11	MII_RXD2	MII_RXD1	A11
B12	MII_RXD0	MII_RXD3	A12
B13	GND	USB+	A13
B14	RESERVED	GND	A14
B15	+5Vdual	USB-	A15
B16	USB_OC#	+12V	A16
B17	GND	GND	A17
B18	-12V	+3.3Vdual	A18
B19	+3.3VD	+5VD	A19
	KEY	KEY	
B20	GND	GND	A20
B21	EE_DOUT	EE_DIN	A21
B22	EE_SHCLK	EE_CS	A22
B23	GND	SMB_A1	A23
B24	SMB_A0	SMB_A2	A24
B25	SMB_SCL	SMB_SDA	A25
B26	CDC_DN_ENAB#	AC97_RESET#	A26
B27	GND	AC97_SDATA_IN2	A27
B28	AC97_SYNC	AC97_SDATA_IN1	A28
B29	AC97_SDATA_OUT	AC97_SDATA_IN0	A29
B30	AC97_BITCLK	GND	A30

Figure 10 – Type B CNR Connector pin-out for ATX Family Form Factors

3.2 Signal Implementation Details

This section provides additional information on the implementation details for some of the signals found on the CNR connector. Both the CNR and Motherboard designers are required to implement the information provided in the following sections. Failure to implement the information provided will prevent broad interoperability between CNR risers and Motherboards.

3.2.1 CDC_DN_ENAB# Implementation Details

Implementing the complete functionality of the CDC_DN_ENAB# signal requires support from different aspects of the system. These different aspects include disable and demotion rules for the codecs on both the motherboard and the CNR, circuitry on the motherboard, and circuitry on the CNR. The following sections describe these requirements in detail. Note that both the motherboard and the CNR **must** implement each of these requirements to prevent interoperability issues.

3.2.1.1 AC ’97 Disable and Demotion Rules

The AC ’97 interface on the CNR is capable of supporting a maximum of three AC ’97 codecs. These codecs can be audio or modem codecs, and can be either on the motherboard, the CNR, or both. Given this, determining which codecs perform which function and when becomes complex. To manage this complexity, several rules have been developed. These rules can be divided into two classes, one class for the motherboard AC ’97 codecs, and another class for the CNR AC ’97 codecs.

The motherboard AC ’97 codec Disable and Demotion Rules are:

1. All AC ’97 *R2.2* codecs on the motherboard **must** always disable themselves when the CDC_DN_ENAB# signal is in a high state.
2. A motherboard AC ’97 codec **must** never change its address or SDATA_IN line used, regardless of the state of the CDC_DN_ENAB# signal.
3. On a motherboard containing an AC ’97 Controller supporting two AC ’97 codecs, the AC ’97 R2.2 codec on the motherboard **must** be connected to the SDATA_IN0 signal of the CNR connector.
4. On a motherboard containing an AC ’97 Controller supporting three AC ’97 codecs, the AC ’97 R2.2 codec on the motherboard **must** be connected to the SDATA_IN2 signal of the CNR connector.
5. A motherboard should not contain more than a single AC ‘97 codec.

The CNR AC ’97 codec Disable and Demotion Rules are:

1. A CNR with a single AC ’97 codec **must** always demote itself to the next available address on the AC ‘97 Interface, and switch to the next available SDATA_IN signal when the CDC_DN_ENAB# signal is in an active low state.
2. A CNR containing two AC ’97 codecs must always demote both codecs to the next available address on the AC ’97 interface when inserted into a system supporting three AC ’97 codecs, and when CDC_DN_ENAB# is in the active low state.
3. A CNR AC ’97 codec **must** never change its address or SDATA_IN line used when the CDC_DN_ENAB# signal is in a high state.
4. All CNR reference designs and CNR production boards **must** not use jumpers to implement the AC ’97 codec disabling and demotion rules. Instead the CNR board **must** use either build time stuffing options or electronic circuitry which monitors the state of the CDC_DN_ENAB# signal to disable and demote the AC ’97 codecs located on the CNR board.
5. A dual codec CNR card **must** contain circuitry which pulls the CDC_DN_ENAB# to a high state through a 1 kohm resistor when the SDATA_IN2 signal on the CNR Connector is not connected to a AC ’97 Controller supporting three AC ’97 codecs. Otherwise the CDC_DN_ENAB# signal must be pulled to a high state through a 100 kohm resistor.
6. A three codec CNR **must** pull the CDC_DN_ENAB# signal to a high state through a 1 kohm resistor.
7. On a motherboard containing an AC ’97 Controller supporting three codecs, the codecs on the CNR **must** always assign the codec addresses as follows. When CDC_DN_ENAB# is high codec address 00b is assigned to the codec attached to SDATA_IN0, codec address 01b is assigned to the codec attached to SDATA_IN1, and codec address 10b or 11b is assigned to the codec attached to SDATA_IN2. When CDC_DN_ENAB# is low codec address 01b is assigned to the codec attached to SDATA_IN0, codec address 10b or 11b is assigned to the codec attached to SDATA_IN1, and codec address 00b is assigned to the (down) codec attached to SDATA_IN2.

3.2.1.2 CDC_DN_ENAB# Implementation Models

The CDC_DN_ENAB# signal is used to indicate which device (motherboard or CNR) is controlling the AC '97 Interface. In other words, CDC_DN_ENAB# indicates where the primary codec resides. IF CDC_DN_ENAB# is high, the primary codec resides on the CNR riser. If CDC_DN_ENAB# is low, the primary codec resides on the motherboard. Both the CNR and the motherboard are responsible for monitoring the state of the CDC_DN_ENAB# signal, and responding to its state in a proper manner. The AC '97 Disable and Demotion Rules listed in Section 3.2.1.1 state that the pull-up resistor value on the CDC_DN_ENAB# pin must be changed depending on the state of the SDATA_IN2 signal. Since all of the codec address assignments and the state of CDC_DN_ENAB# must be set prior to bringing the codecs out of reset, all of the state determination and address control must be performed prior to the rising edge of AC97_RESET#. Table 10 provides the truth table for setting the state of the CDC_DN_ENAB# signal based on the state of the SDATA_IN2.

SDATA_IN2 State During AC '97 Reset	*CDC_DN_ENAB# pull-up Resistor Value*
Low	*100 kohm*
High	*1 kohm*

Table 10 – CDC_DN_ENAB# Pull-up Resistor Values

The easiest method to set a high or low value on the SDATA_IN2 signal is through the use of pull-up and pull-down resistors, as shown in Figure 11. The use of pull-up and pull-down resistors will require that while reset is active (low), the devices attached to the SDATA_IN2 signal must not drive the signal, otherwise a false level may be detected.

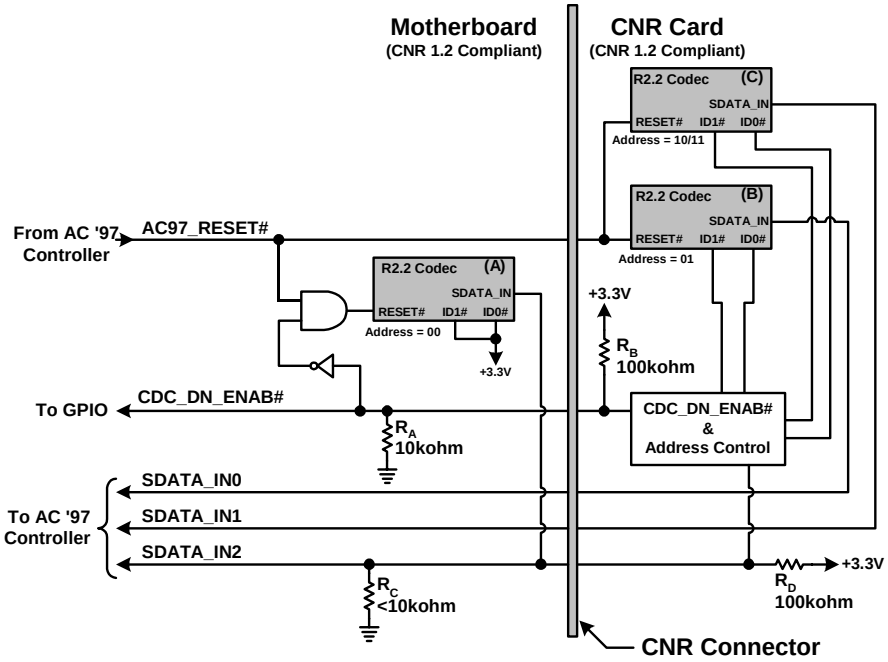


Figure 11 – AC97_SDATA_IN2 Circuit

Detection and Control:

Detecting the state of SDATA_IN2 and controlling the pull-up resistor value on CDC_DN_ENAB# is very straight forward. After the falling edge of AC97_RESET#, the state of the SDATA_IN2 signal is used to set the pull-up impedance on the CDC_DN_ENAB# signal. The block diagram shown in Figure 12 provides a high-level block diagram for one possible implementation of a detection and control circuit.

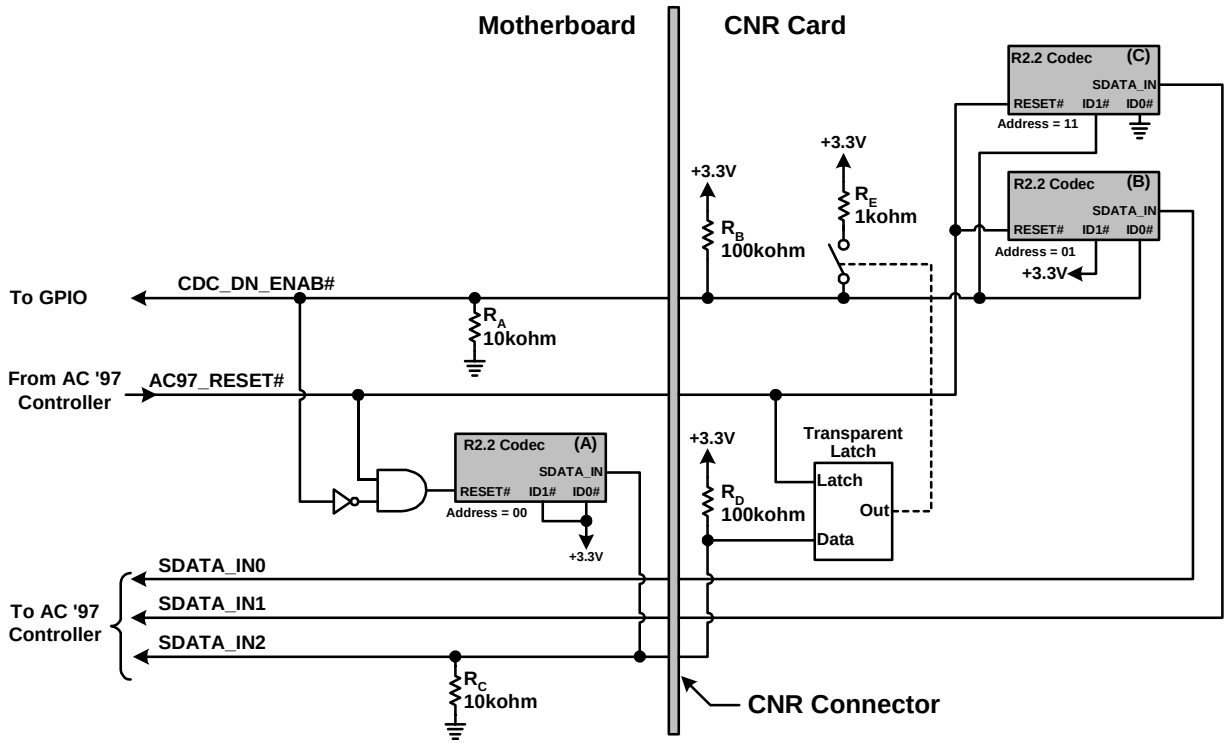


Figure 12 – CDC_DN_ENAB# & Address Control Block

The state of SDATA_IN2 is determined by the pull-up and pull-down resistors on the SDATA_IN2 line. When SDATA_IN2 is supported by the AC '97 Controller, pull-down resistor R_C is installed on the motherboard. Since resistor, R_C, is an order of magnitude smaller than the pull-up resistor, R_D installed on the CNR card, the SDATA_IN2 signal will assume a low state. The state of SDATA_IN2 is latched, at the rising edge of AC97_RESET#, by the Transparent Latch. (While AC97_RESET# is low, the Transparent Latch output follows the state of the Data pin, insuring that the circuit will be settled long before the rising edge of AC97_RESET#.) The Transparent Latch is required to prevent the normal operation of the SDATA_IN2 signal from changing the CDC_DN_ENAB# pull-up value. The output of the Transparent Latch is then used to control an electronic switch which will remain open when SDATA_IN2 is in a low state, and closed when SDATA_IN2 is in a high state. When the electronic switch is closed, the CDC_DN_ENAB# signal will be pulled high through a 1 kohm resistor, disabling the motherboard codec.

> ***Note:*** The above implementation requires that the AC '97 codec follow the recommendation of the AC '97 Revision 2.2 Specification that an Audio codec tri-state its SDATA_IN and BIT_CLK outputs during the period that AC97_RESET is active (low). See Section 10.2.1.1 of the "Audio Codec '97 Specification, Revision 2.2", dated September, 2000.

Codec address control does not require any additional circuitry. All control can be provided through proper connections between the codec address pins and the CDC_DN_ENAB# signal, as shown in Figure 12. Note that the addressing is not sequential and requires that the AC '97 Controller be able to support missing and/or non-sequential codec addressing.

Implementation:

There are several methods of implementing the block diagram shown in Figure 12. The schematic in Figure 13 is one possible implementation of the CDC_DN_ENAB# control and codec address circuitry. There are many other possible methods that could be used. It is ultimately up to the CNR card designer to determine the best and most cost effective implementation for their own design, provided that all of the AC '97 and CNR Specifications are met.

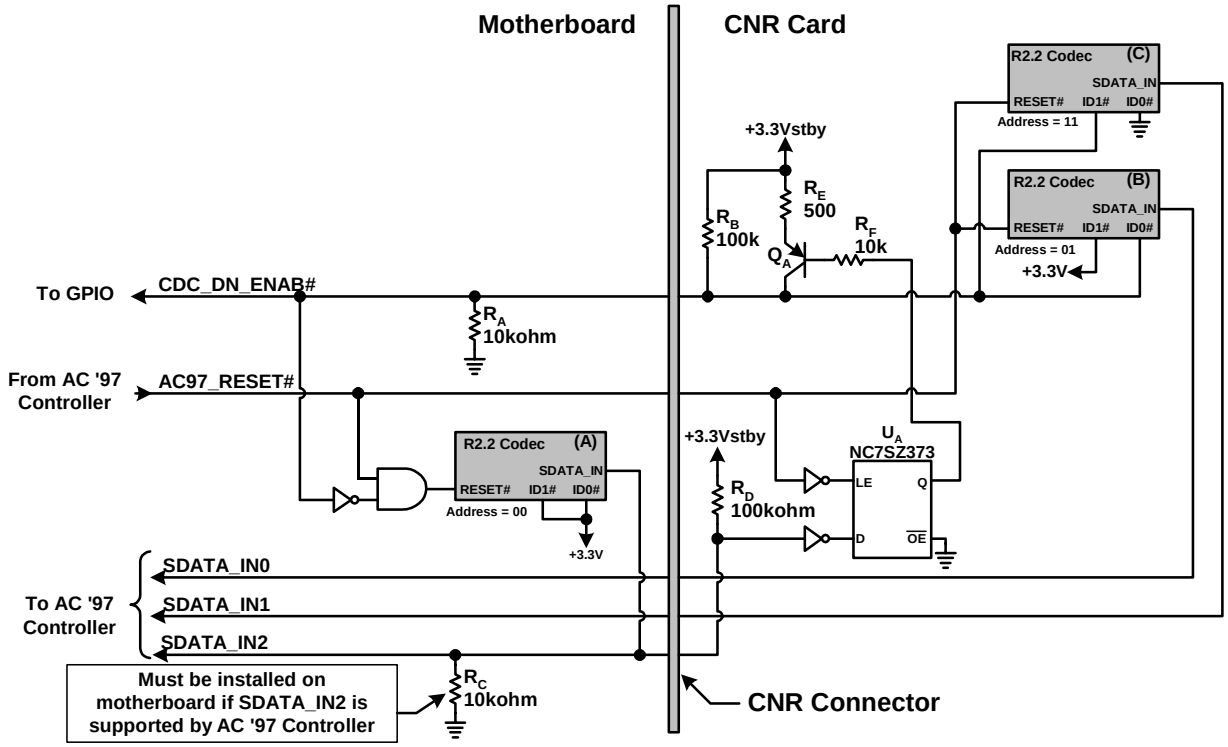


Figure 13 – CDC_DN_ENAB# control and Codec Address circuitry

With above circuit, when SDATA_IN2 is high, transistor Q_A will pull the CDC_DN_ENAB# signal to about 3Volts, resulting in the CNR mastering the AC '97 Interface, and codec (B) at address 00b, with codec (C) at address 01b. When SDATA_IN2 is pulled low, through R_C, transistor Q_A will be off and the CDC_DN_ENAB# signal will assume the appropriate state based on whether R_A is installed on the motherboard. Once AC97_RESET# goes from low to high, the state of the SDATA_IN2 is latched by U_A, preventing any further transitions in SDATA_IN2 from causing changes in CDC_DN_ENAB#. The standby voltage rail should be used for the pull-ups in modem codec solutions to prevent the modem codec address from changing during system standby.

3.3 Communication and Network Riser Interface Electrical Specifications

Signal Name	Min.	Max.	Units	Comments
CDC_DN_ENAB#				
Pull-up resistance	950	1050	Ohms	Value of R_B for CNR controlling the AC '97 Interface
Pull-up resistance	95000	105000	Ohms	Value of R_B for CNR not controlling the AC '97 Interface
Pull-down resistance	9500	10500	Ohms	Value of R_A for Motherboard
V_{IL}	0.5	0.9	Volts	
V_{IH}	1.65	3.6	Volts	
Leakage Current	-45	-45	µAmps	Maximum combined leakage current allowed for all digital pins connected to CDC_DN_ENAB# (excludes R_A and R_B)
USB+, USB-	--	--	--	Refer to the current version or release of the Universal Serial Bus Specification 2.0.
USB_OC#				
V_{OL}	--	1.0	Volts	Voltages less than V_{OL} indicate an over current condition[2,3]
V_{OH}	4.5	--	Volts	Voltages greater than V_{OH} indicate a normal operating condition[2,3]
I_{OL}	--	0	Amps	
I_{OH}	0.001	--	Amps	
AC97_BITCLK, AC97_SYNC, AC97_SDATA_OUT, AC97_SDATA_IN0, AC97_SDATA_IN1, AC97_SDATA_IN2, AC97_RESET#	--	--	--	Refer to the current version or release of the appropriate Audio Codec '97 Component Specification (Revision 2.1 or later).
AC97_SDATA_IN2				
Pull-up resistance	90000	110000	Ohms	Value for R_D on CNR card
Pull-down resistance	4700	10000	Ohms	Value for R_C including chipset pull-down on motherboard
V_{IL}	0.5	0.9	Volts	
V_{IH}	1.65	3.6	Volts	
Leakage Current	-45	45	UAmps	Maximum combined leakage current allowed for all digital pins connected to AC97_SDATA_IN2 (excludes R_C and R_D).\
T_{SU}	100		nS	Setup time of AC_SDATA_IN2 before rising edge of AC97_RESET#
T_H	10		nS	Hold time of AC97_SDATA_IN2 after rising edge of AC97_RESET#
LAN_CLK, LAN_RSTSYNC, LAN_TXD2, LAN_TXD1, LAN_TXD0, LAN_RXD2, LAN_RXD1, LAN_RXD0	--	--	--	Refer to the current version or release of the Core Logic Design Specification.
MII_RXD3, MII_RXD2, MII_RXD1, MII_RXD0, MII_RXC, MII_RXDV, MII_RXERR, MII_TXD3, MII_TXD2, MII_TXD1, MII_TXD0, MII_TXC, MII_TXEN, MII_CRS, MII_COL, MII_MDIO, MII_MDC	--	--	--	Refer to the current version or release of the IEEE 802.3u Specification.
EE_SHCLK, EE_DIN, EE_DOUT, EE_CS				Refer to the current version or release of the Core Logic Design Specification.
SMB_SCL, SMB_SDA, SMB_A2, SMB_A1, SMB_A0				These specifications are with reference to the SMBus Controller. A SMBus EEPROM attached to the SMBus Controller must be able to generate the V_{IL} and V_{IH} specification on its outputs, and accept the V_{OL} and V_{OH} levels on its inputs.
V_{IL}	-0.5	0.6	Volts	
V_{IH}	1.4	--	Volts	
V_{OL}	--	0.4	Volts	
V_{OH}	2.5	--	Volts	
AC '97 Interface Trace Characteristic Impedance	51	69	Ohms	Traces include: AC97_RESET#, AC97_BITCLK, AC97_SYNC, AC97_SDATA_OUT, AC97_SDATA_IN0, AC97_SDATA_IN1,

Table 11 – Communication and Networking Riser Electrical Specification

[2] It is the responsibility of the CNR designer to properly implement circuitry on the CNR board which absolutely indicates an over current condition to the motherboard, through the use of the USB_OC# signal. Additional information on USB over current can be found in the USB Revision 2.0 Specifications.

[3] It is the responsibility of the motherboard designer to properly translate the USB_OC# signal, from the CNR, to voltage and drive strength levels that are compatible with the motherboard USB controller.

Signal Name	Min.	Max.	Units	Comments
LAN Interface Trace Characteristic Impedance	51	69	Ohms	Traces include: LAN_RSTSYNC, LAN_CLK, LAN_TXD[2:0], LAN_RXD[2:0], MII_RXD[3:0], MII_RXC, MII_RXDV, MII_RXERR, MII_TXD[3:0], MII_TXC, MII_TXEN, MII_CRS, MII_COL, MII_MDIO, MII_MDC
SB Interface Trace **Characteristic Impedance**	76.5 51	103.5 69	Ohms Ohms	90 ohms ±15% (traces include USB+ and USB-) 60 ohms ±15% (traces include USB_OC#)
SMBus/Microwire Trace Characteristic Impedance	51	69	Ohms	Traces include: SMB_SCL, SMB_SDA, SMB_A2, SMB_A1, SMB_A0, EE_SHCLK, EE_DIN, EE_DOUT, EE_SLCT
+12V				
Tolerance	--	±10	%	Measured at the CNR connector.
Ripple Voltage	--	0.150	V_{PK-PK}	Measured at the CNR connector.
Supply Current	--	0.500	Amps	
-12V				
Tolerance	--	±15	%	Measured at the CNR connector.
Ripple Voltage	--	0.150	V_{PK-PK}	Measured at the CNR connector.
Supply Current	--	0.100	Amps	
+5VD				
Tolerance	--	±5	%	Measured at the CNR connector.
Ripple Voltage	--	0.075	V_{PK-PK}	Measured at the CNR connector.
Supply Current	--	1.00	Amps	
+3.3VD				
Tolerance	--	±5	%	Measured at the CNR connector.
Ripple Voltage	--	0.075	V_{PK-PK}	Measured at the CNR connector.
Supply Current	--	1.00	Amps	
+5Vdual/+5VSB				
Tolerance	--	±5	%	Measured at the CNR connector.
Ripple Voltage	--	0.075	V_{PK-PK}	Measured at the CNR connector .
Supply Current				
Active State	--	0.500	Amps	ACPI S0 State
Wake enabled	--	0.500	Amps	ACPI S3 and S4 states
Non-wake enabled	--	0.020	Amps	ACPI S3 and S4 states
+3.3Vdual				
Tolerance	--	±5	%	Measured at the CNR connector.
Ripple Voltage	--	0.075	V_{PK-PK}	Measured at the CNR connector.
Supply Current				
Active State	--	1.0	Amps	ACPI S0 state
Wake enabled	--	0.375	Amps	ACPI S3 and S4 states
Non-wake enabled	--	0.020	Amps	ACPI S3 and S4 states
CNR Total Power Dissipation	--	25	Watts	

Table 11 (cont.) – Communication and Networking Riser Electrical Specifications

4 Power Management and Interface Requirements

The CNR architecture supports Instantly Available PC power management including full support for "Off-yet Communicating" capabilities. The working state power sources delivered to the CNR connector enable normal working state operation, while auxiliary voltage sources are also delivered to the CNR connector to enable wake logic to remain active when the working state voltage rails have been shut off.

The following subsections focus on power management design considerations.

4.1 Power Management

Power distribution is dependent upon the supported features of the CNR board. The following subsections detail the power distribution requirements for each of the supported CNR interfaces.

- AC '97 interface
- LAN interface
- USB interface
- SMBus interface

4.1.1 AC '97 Interface Power Management

The following table defines the recommended power supplies for the AC '97 interface (for audio and/or modem subsystems) on the CNR board:

	+Vmain[4]	+12V/+5VD[5]	-12V	+Vdual[6]
AC-link (codec signals) ■ BIT_CLK ■ SDATA_IN	 ✓ (AC device) ✓ (AC device)			 ✓ (MC device, if primary) ✓ (MC device)
AC-link (controller signals) ■ SYNC ■ RESET ■ SDATA_OUT	 ✓ (AC & MC devices) ✓ (AC device) ✓ (AC & MC devices)			 ✓ (MC device)
Riser digital logic	✓			
Modem analog circuitry		✓		
Modem wake logic				✓
Audio analog circuitry		✓	✓ (optional)	

Table 12 – Recommended Power Distribution for the AC '97 Interface Signals

In order to enable wake from deep sleep modes, all CNR board driven AC-link signals, as well as all other digital logic on the CNR board must be powered by +Vdual[6]. The +Vdual power supply must be used to power both the digital portion of the audio and/or modem codec and the AC-link portion of the digital controller, as specified in the AC '97 Specification, Revision 2.1. In addition, it is recommended that the audio and/or modem subsystems locally regulate the +12V down to +5Vanalog for use by its analog circuitry (to reduce signal degradation potentially introduced by the noisy digital power supply).

When the system enters a sleep state where the main power rails are shut off (that is, ACPI S3, or S4), those portions not related to wake enabling (not connected to a standby power supply) will be completely shut off, consuming no power.

4.1.1.1 Split Partition AC '97 and MC '97 CNR Board

In the case of a split partition implementation (audio codec on motherboard and modem codec on riser), the power distribution must be implemented as shown in Table 12. This allows the audio portion of a CNR board to be powered from standard working state voltage sources that are shut off when the PC enters an ACPI sleep state of S3 or S4.

In order to support an Instantly Available PC, the modem portion of the CNR board must draw its power from the +Vdual power source. This enables a modem codec to power its wake logic when in an S3 or S4 ACPI sleep state.

4.1.1.2 AMC '97 Combination Codec CNR Board

The AMC '97 combination codec CNR board must implement the same power distribution strategy as for the split partitioned AC '97-plus-MC '97 CNR board. This leads to the possibility of an AMC '97 codec design using split power wells, thus enabling multi-voltage power distribution for different sections of the device.

4.1.2 LAN Interface Power Management

Power distribution requirements for the LAN devices are outside of the scope of this specification. The CNR designer should refer to the appropriate data sheets for the LAN components to determine proper power supply connections to support an Instantly Available PC and "Off-yet Communicating" capabilities.

4.1.3 USB Interface Power Management and Interface Requirements

The USB interface section of the CNR connector consists of three signals and a power supply. The following sections describe the requirements for use of a USB interface on the CNR connector. Note that in addition to the statements in these sections, the CNR designer must insure that their USB design is compliant with the Design Guidelines for the USB 2.0 Interface.

---

4 Where +Vmain refers to +3.3VD or +5VD.

5 The selection of either the +12V or the +5VD for use in an analog system depends on the ultimate performance desired by the CNR designer. Note that the +5VD power supply is generally much noisier than the +12V power supply.

6 Where +Vdual refers to +3.3Vdual or +5Vdual.

4.1.3.1 USB Differential Pair and Over Current Sensing

The CNR connector provides both the USB differential pair (USB+ and USB-) and the USB Over Current signal (USB_OC#). These signals are required to properly implement a USB device, function, or port on the CNR board. Note that it is the responsibility of the CNR board designer to meet all of the USB 2.0 specifications. In addition to the requirements called out in the Universal Serial Bus Specification Revision 2.0, the designer of the CNR riser must also meet the following requirements:

1. To minimize the possibility of USB speed related failures the following three requirements must be met.
   a) The CNR connector **must** be connected to the root hub on the motherboard.
   b) It is **strongly** recommended that the highest speed USB port supported on the motherboard be routed to the CNR connector.
   c) The CNR designer must insure that their USB design is compliant with the Design Guidelines for the USB 2.0 Interface.
2. If a USB device, function, or port is implemented on the CNR, the USB_OC# signal must be used to indicate if the device, function, or port attached to the USB+ and USB- signals has exceeded the maximum specified current (500 mA) by going low.
3. If USB port (15 kΩ) pull-down resistors are required, they must be located on the motherboard.
4. If USB function speed (1.5 kΩ) pull-up resistors are required, they must be located on the CNR board.

4.1.3.2 USB Power Management

Power for USB devices on the CNR can be provided through several of the supplies available on the CNR connector. However, if the USB device requires power while the system is in a sleep or suspended state, the +5Vdual power supply must be used. Assuming that the +5Vdual power supply is used to supply power to a wake enabled USB device on the CNR, the requirements for the type of USB device are further defined, as follows:

1. If the USB signals on the CNR board are routed to a function (e.g. DSL modem) then the maximum current consumption must not exceed 500 mA on the +5Vdual power supply in either and active system state or a wake enabled system state.
2. If the USB signals on the CNR board are routed to a USB hub device, then the maximum current consumption must not exceed 500 mA in an active system state or wake enabled system state.
3. If a hub is placed on the CNR, no more than four ports from the hub can be utilized (either as externally available USB ports, or functions on the CNR board).
4. If a hub is placed on the CNR, all devices or ports from the hub must be connected to low power or self powered USB devices or functions.

4.1.4 SMBus Interface Power Management

Power management for the SMBus interface is very straightforward. Since the BIOS only reads the contents of the SMBus EEPROM (electrically erasable programmable read only memory) during ACPI state S0, one can make the valid assumption that the +3.3VD power supply is sufficient for powering the SMBus EEPROM. In fact, if the CNR board designer follows this specification, which requires the SMBus PnP EEPROM to be fully compliant with the System Management Bus Specification, Revision 1.1 (dated December 11, 1998), there will not be any issues (as the System Management Bus Specification, Revision 1.1, requires that the leakage current on an unpowered SMBus device be less than 5uA). However, to ensure that the SMBus interface is not excessively loaded while the system is in a sleep state it is strongly recommended that the SMBus EEPROM be powered by the +3.3Vdual power supply (pin A18 of both the Type A and Type B connectors).

4.2 Reset Considerations

The following sections describe the considerations for the AC '97 interface, when resetting/restoring the CNR board to a known state.

4.2.1 AC '97 Interface Reset Considerations

The AC '97 architecture defines three types of reset that AC '97 compatible codecs must comprehend.

- Cold Reset Performs a complete codec hardware reset
- Warm Reset Brings the AC-link out of PR4 low-power mode, no internal initialization required
- Register Reset Reinitializes the codec via a software command

Prior to AC '97 Version 2.1, when the PC is sleeping in either the ACPI S3, S4, or S5 states, and a wake event occurs, the system brings the audio/modem subsystems back to full operation by reapplying power to the AC-link and asserting a cold reset sequence.

AC '97, Version 2.1 imposes a new requirement for AC-link RESET# behavior. This new requirement dictates that RESET# remain actively driven during S3, S4, or S5 states, so that auxiliary-powered modem codecs know, without doubt, that the AC-link RESET# was asserted, as opposed to floating at or near ground. Vdual powered circuitry would then look for the trailing low-to-high transition on the AC-link RESET# signal, which indicates that the AC-link was powered back up and a "resume" reset sequence had occurred.

This presents an issue for modem codecs that are designed to wake the system from S3 or S4 states. An auxiliary-powered modem codec must retain portions of its internal state, including the wake event state, after experiencing this resume sequence. The root issue is that the auxiliary-powered codec must be capable of determining how to interpret and deal with the AC-link RESET# assertion so that its internal state is not corrupted when resuming.

There are numerous ways, ranging from hardware-only solutions to hardware/driver solutions, of addressing this without impacting either this specification or the AC '97 Component Specification. Disclosure of any detailed implementation-specific information is beyond the scope of this specification.

4.3 AC '97 Clocking Considerations

4.3.1 AC '97 ACPI S0 "Working State" Clocking

In a multiple codec design, where audio is the primary codec (that is, the source of BIT_CLK), ACPI S0 "working state" power management of the primary codec can present a clocking issue for the secondary codec. Specifically, take the case when the system is currently in the working state (ACPI S0) and the operating system power management policy manager determines that the audio (primary codec) is idle. At this point, the operating system may decide to transition the audio subsystem to its lowest power state (for example, D3hot). If the audio driver were to then place the audio codec down to its lowest power state (including PR4), the AC-link would enter its low-power mode with BIT_CLK stopped.

Had the modem been in use, or had been needed at any point following this, it would be incapable of operating correctly since the audio driver had disabled its working state clock source (BIT_CLK).

For information on how this problem should be resolved, refer to the AC '97 Component Specification, Version 2.1.

Note: ***The audio and modem drivers must be capable of operating completely independent of each other (for example, in the case when either the audio or modem hardware has entered a sleep state or has been disabled). All interdependencies (such as speakerphone audio and control) must be dealt with at the API level.***

4.3.2 AC '97 ACPI S3 and S4 "Sleeping State" Clocking

An MC '97 modem codec, when configured as the secondary codec, depends upon BIT_CLK from the primary codec for its normal, working-state clock source. When the system is in ACPI S3 or S4 sleep states[7], the MC '97 must make provisions for a free-running clock source if needed for support of CallerID capture or other wake-event related circuitry. This clock source must be powered by Vdual supply, and its frequency is recommended to be as low as is both economically and technologically possible to conserve power while the PC is asleep.

[7] Or any other state where the primary audio codec is not providing a free running BIT_CLK. (that is, BIT_CLK is held low or unpowered).

5 Mechanical Requirements

The following sections provide the mechanical requirements for the CNR board and connector placement for the ATX family form factors. Note that the CNR Specification does **not** support the NLX form factor. Future versions or revisions of this specification will support additional form factors, as they become available and/or widely used by the PC industry.

5.1 CNR Connector Location for the ATX Family Form Factors

Figure 14 shows the placement of a CNR connector on the ATX family motherboards. Note that Figure 14 also shows the motherboard component height restrictions. Failure to meet the specified height restrictions may prevent a CNR board from properly seating in the CNR connector. In addition, the CNR connector mechanical location is shown relative to the PCI card edge connector. This implies that the absolute placement (relative to board edges and mounting holes) of the CNR connector must be determined from the motherboard specification for the appropriate form factor. If a PCI connector is not placed on the motherboard, the CNR connector placement must be determined relative to a phantom, or assumed PCI connector.

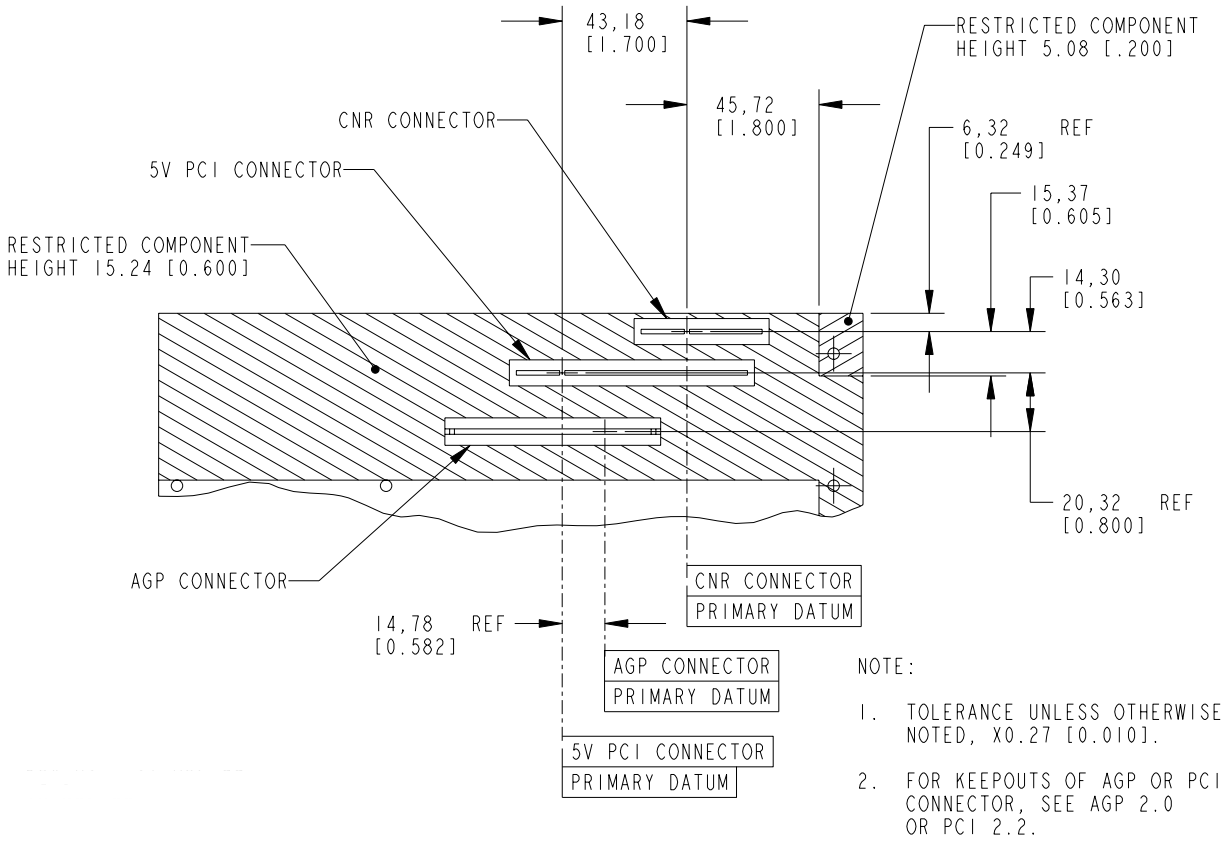


Figure 14 – CNR Connector location for the ATX Family Form Factors

5.2 Communication and Networking Riser Board Dimensions

The following sections provide the mechanical dimensioning information for the CNR board in the various form factors. Currently this is limited to the ATX family form factors including ATX, microATX, and FlexATX.

Note: ***In some instances it may be desirable to modify the overall height of the CNR board from that specified, in order to reduce the cost of the CNR board. The CNR Specification does not specifically prohibit a change in the overall height of the CNR board. However, if the CNR designer chooses to change the overall height, the responsibility for redesign of the I/O bracket and compatibility (mechanical fit, shock and vibration tolerance) with the intended form factor becomes the responsibility of the CNR board designer.***

5.2.1 Board Dimensions for Standard Full Height ATX Family Form Factors

Figure 15 and Figure 16 show the mechanical dimension information for a standard full height CNR circuit board and I/O bracket placement for the ATX family form factors. In addition to the component keep out areas shown on the drawing, there are component height restrictions for both the primary and secondary component sides of the board. The primary component side of a standard full height CNR board must not contain any components that exceed 0.570 inches (14.48mm) in height. The secondary component side of a standard full height CNR board must not contain any components that exceed 0.105 inches (2.67mm) in height.

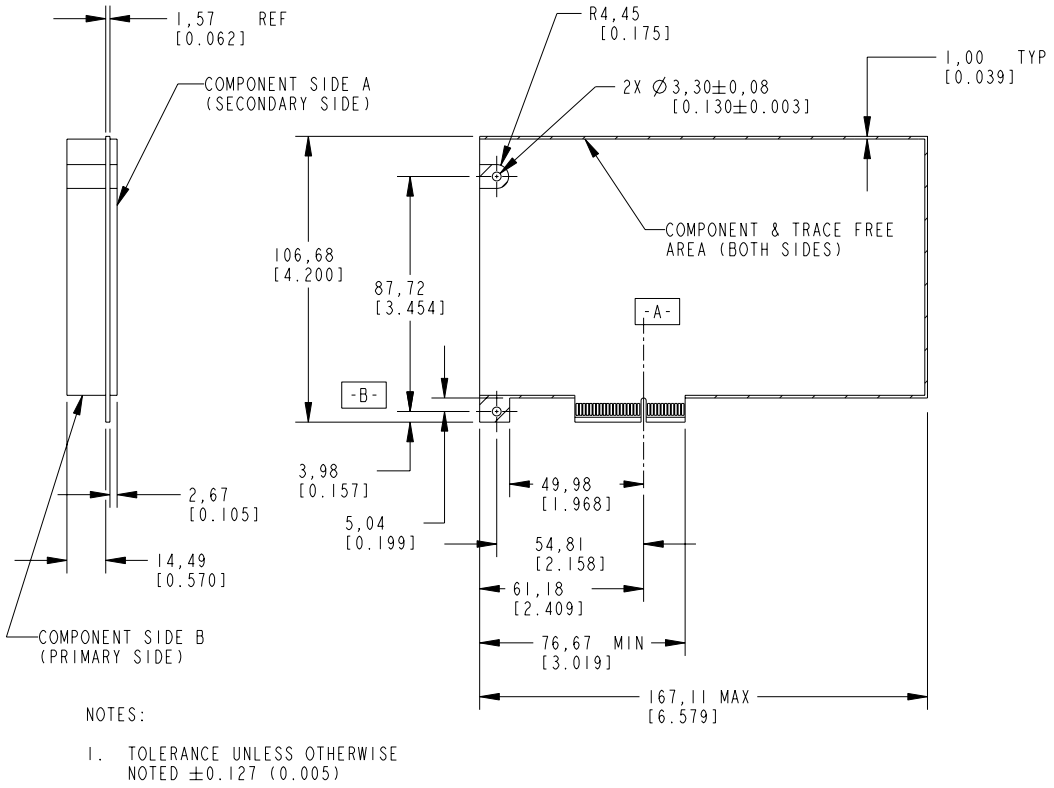


Figure 15 – Dimensions for a Standard Full Height CNR board

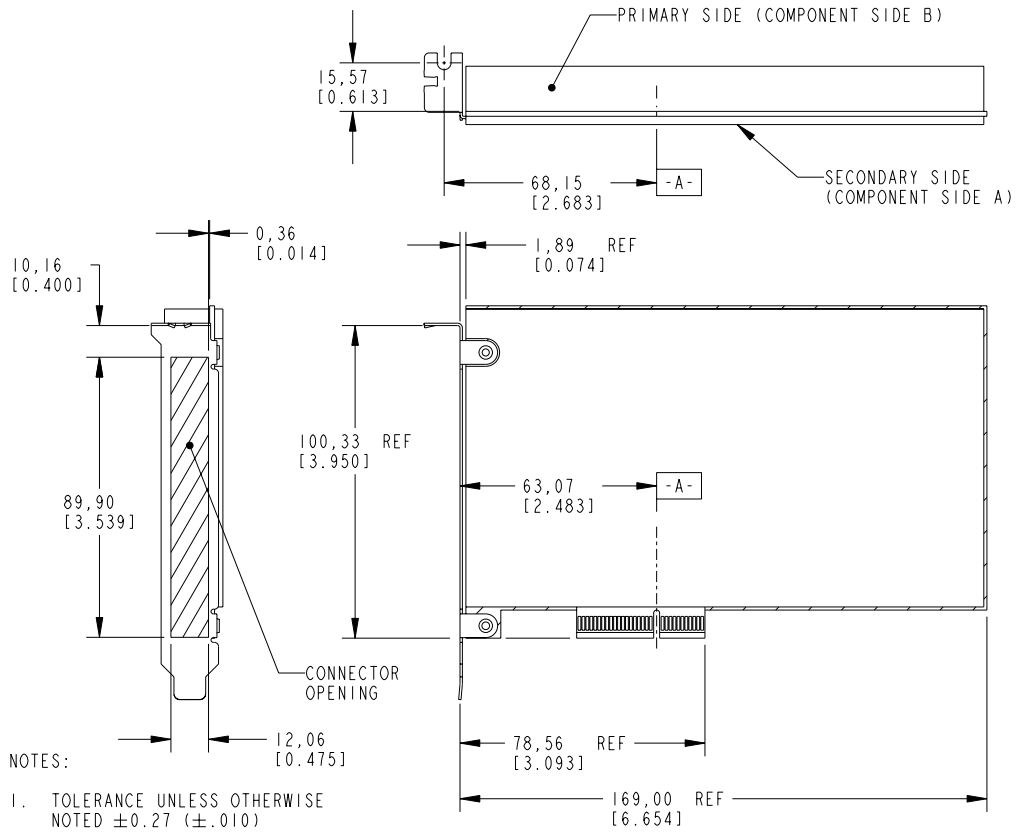


Figure 16 – Dimensions for a Standard Full Height CNR board with I/O bracket

5.2.2 Board Dimensions for Low Profile ATX Family Form Factors

Figure 17 and Figure 18 show the mechanical dimension information for a low profile CNR circuit board and I/O bracket placement for the ATX family form factors. In addition to the component keep out areas shown on the drawing, there are component height restrictions for both the primary and secondary component sides of the board. The primary component side of a low profile CNR board **must** not contain any components that exceed 0.570 inches (14.48mm) in height. The secondary component side of a low profile CNR board **must** not contain any components that exceed 0.105 inches (2.67mm) in height.

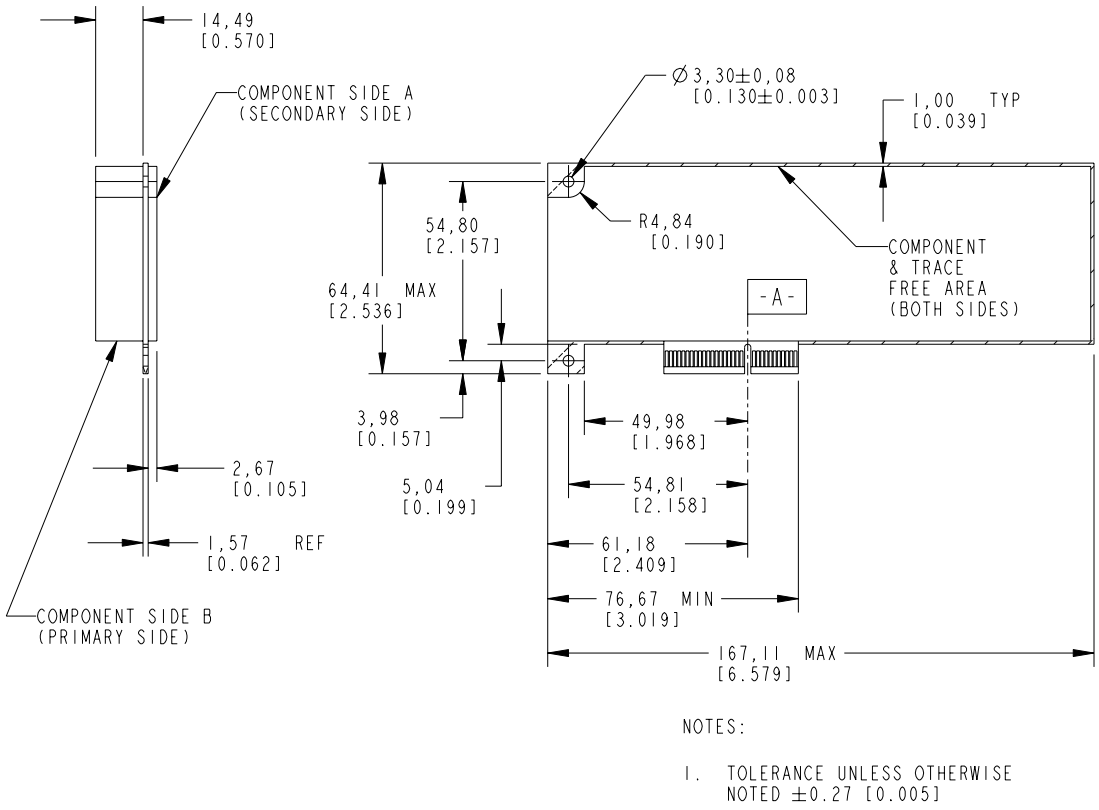


Figure 17 – Dimensions for Low Profile CNR board

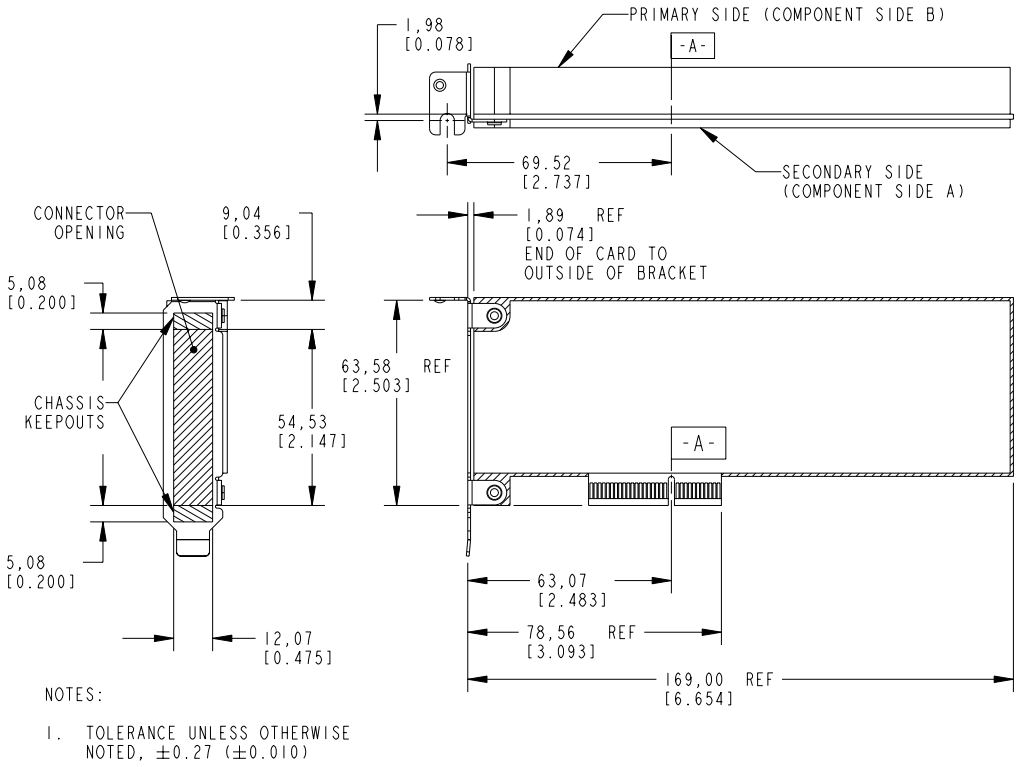


Figure 18 – Dimensions for Low Profile CNR board with I/O bracket

5.3 Communication and Networking Riser Edge Card Contact for the ATX Family Form Factors

Figure 19 shows the mechanical requirements for the edge card contacts of the CNR board for both the standard full height and low profile version of the ATX Family Form Factors.

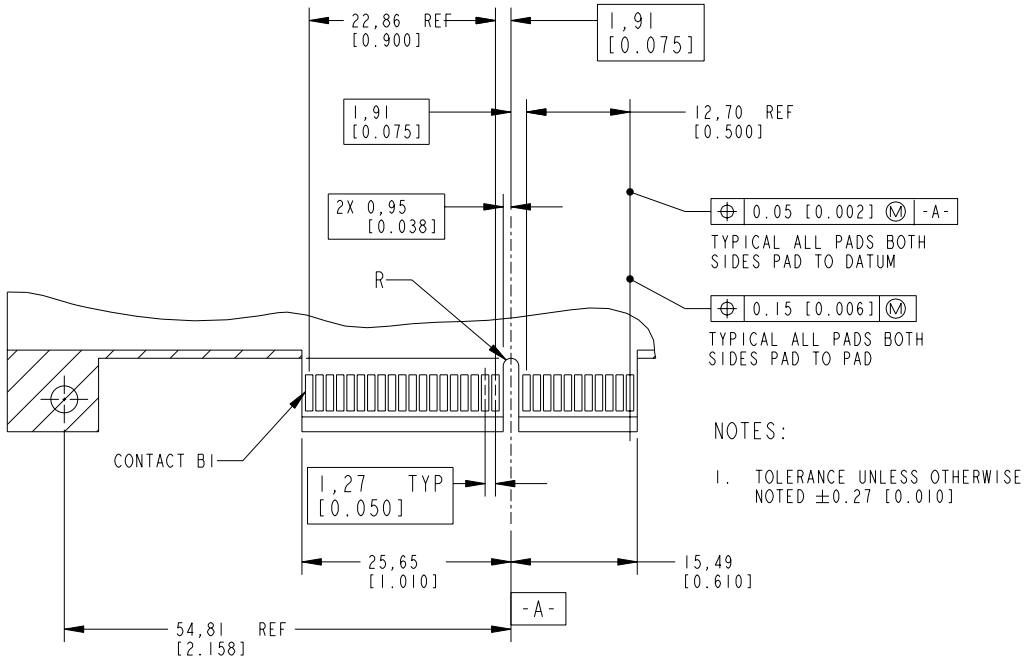


Figure 19 – Full and Low Profile CNR edge card dimensions

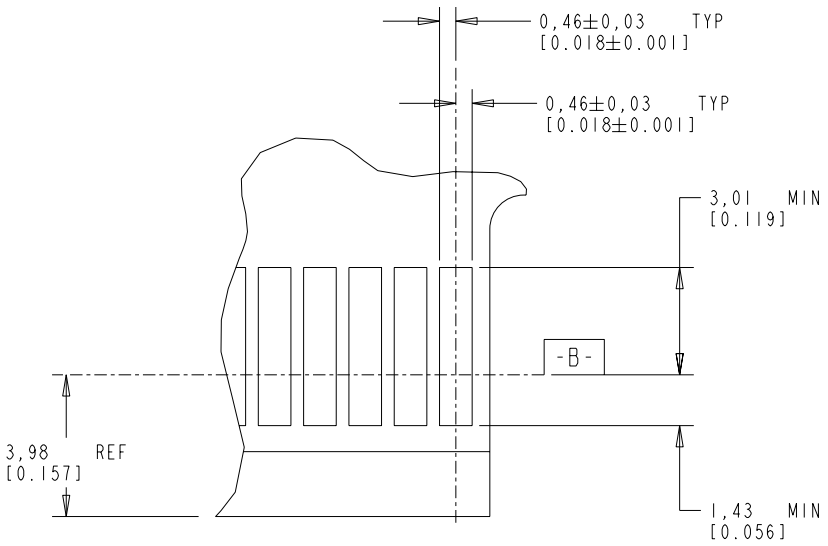


Figure 20 – CNR Card Edge Connector Contacts

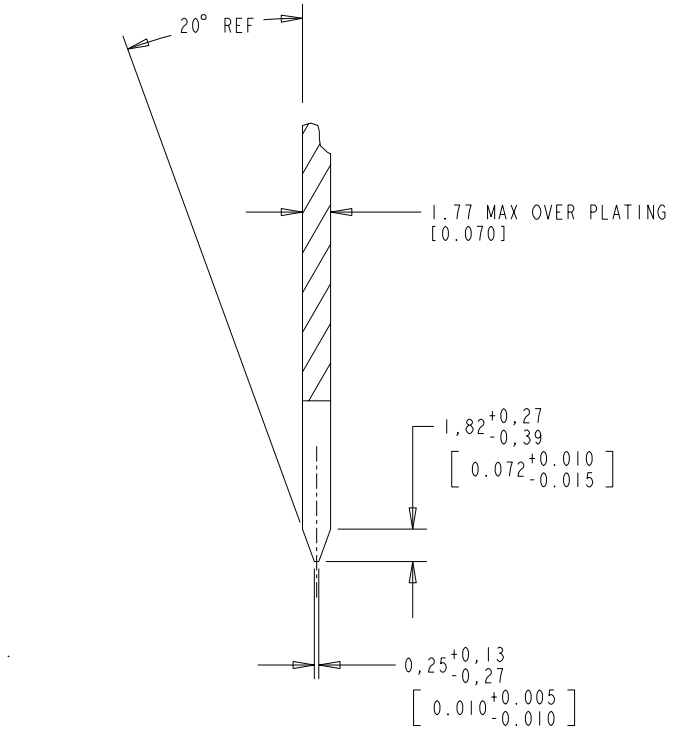


Figure 21 – CNR Card Edge Connector Bevel

5.4 Communication and Networking Riser Interface Connector Type

5.4.1 ATX Family CNR Connector Description

The CNR Type A and Type B Connector for the ATX family form factors is a 60-contact (arranged in a 2 by 30 configuration) edge card receptacle. The table below provides the manufacturer's part number for the known compliant connectors. The part numbers are accurate as of the creation of this specification, however, the designer must refer to the connector supplier's part number, and recommendations for layout detail tolerancing.

Manufacturer Name	Manufacturer's Part Number
Amp Incorporated	1-650090-2
Amp Incorporated	650090-6
Foxconn	EH03011-PL

Table 13 – Type A and Type B Connector Manufacturers

Refer to Figure 22 and Figure 23 for connector dimension and layout recommendations.

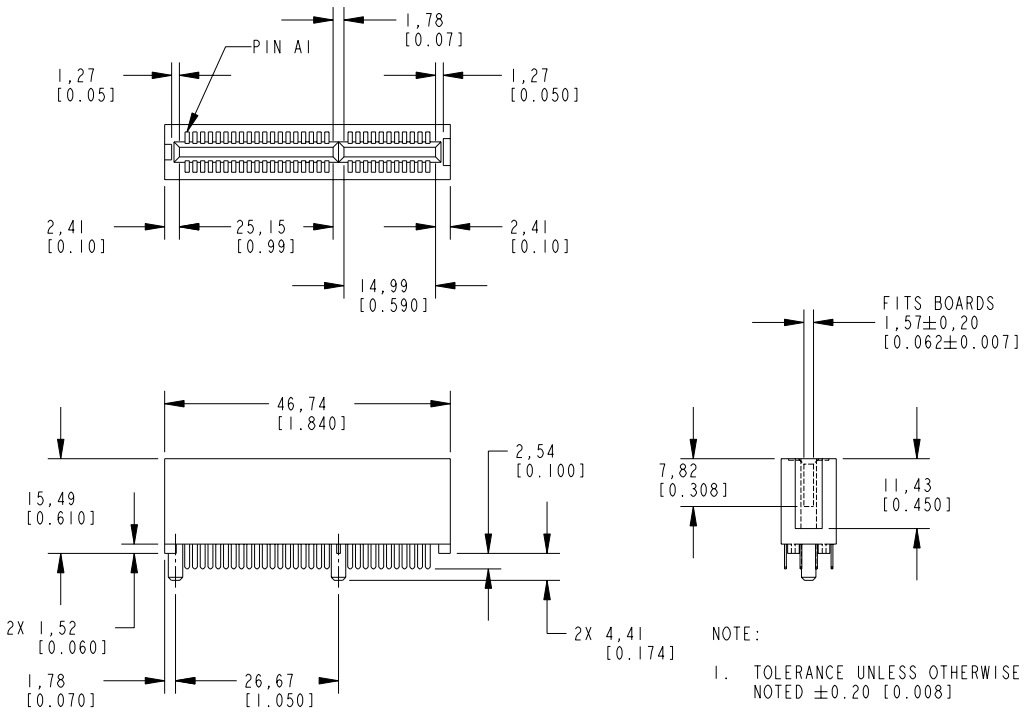


Figure 22 – CNR Type A and Type B Connector Dimensions

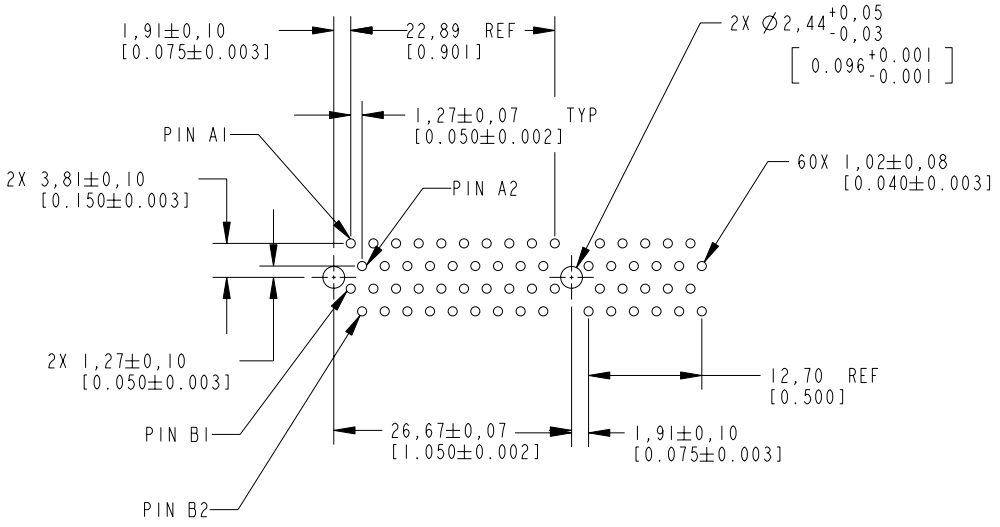


Figure 23 – CNR Type A and Type B Connector Layout Recommendation

5.4.2 CNR Type A and Type B Connector Physical Requirements

Connector Part	Material Requirement
Connector Housing	High temperature thermoplastic, UL flammability rating 94V-0 Color: Brown
Contacts	Phosphor bronze
Contact Finish	0.000762 millimeter (0.000030 inch) minimum gold over 0.001270 millimeter (0.000050 inch) minimum nickel in the contact area. Alternate finish: Gold flash over 0.001000 millimeter (0.000040 inch) minimum palladium or palladium-nickel over nickel in the contact area.

Table 14 – Type A and Type Connector Physical Requirements

5.4.3 CNR Type A and Type B Connector Performance Requirements

Parameter	Specification
Durability	100 mating cycles without physical damage or exceeding low level contact resistance requirement when mated with the recommended card edge.
Mating Force	1.7N (6 oz.) maximum average per opposing contact pair using MIL-STD-1344, Method 2013.1 and gauge per MIL-C-21097 with profile as shown in add-in board specification
Contact Normal Force	75 grams minimum

Table 15 – Type A and Type B Connector Mechanical Performance Requirements

Parameter	Specification
Contact Resistance	(Low signal level) 30 mΩ maximum initial, 10 mΩ maximum increase through testing. Contact resistance test per MIL-STD-1344, Method 3002.1
Insulation Resistance	1000 MΩ minimum per MIL STD 202, Method 302, Condition B
Dielectric Withstand Voltage	500 VAC RMS per MIL-STD-1344, Method D3001.1, Condition 1
Capacitance	2 pF maximum at 1MHz
Current Rating	1 A, 30 °C rise above ambient
Voltage Rating	125 V
Certification	UL Recognition and CSA Certification recommended

Table 16 – Type A and Type B Connector Electrical Performance Requirements

Parameter	Specification
Operating Temperature	-40 °C to 105 °C
Thermal Shock	-55 °C to 85 °C, 5 cycles per MIL-STD-1344, method 1003.1
Flowing Mix Gas Test	Battelle, Class II. Connector mated with board and tested per Battelle method

Table 17 – Type A and Type B Connector Environmental Performance Requirements

5.5 Communication and Networking Riser I/O Bracket Dimensions

The following sections provide the mechanical information for the I/O brackets to be used by the CNR board in the various form factors.

5.5.1 Communication and Networking Riser Full Height I/O Bracket for the ATX Family

Figure 24 shows the recommended dimension information for the I/O bracket to be used with the CNR board for the ATX family form factors.

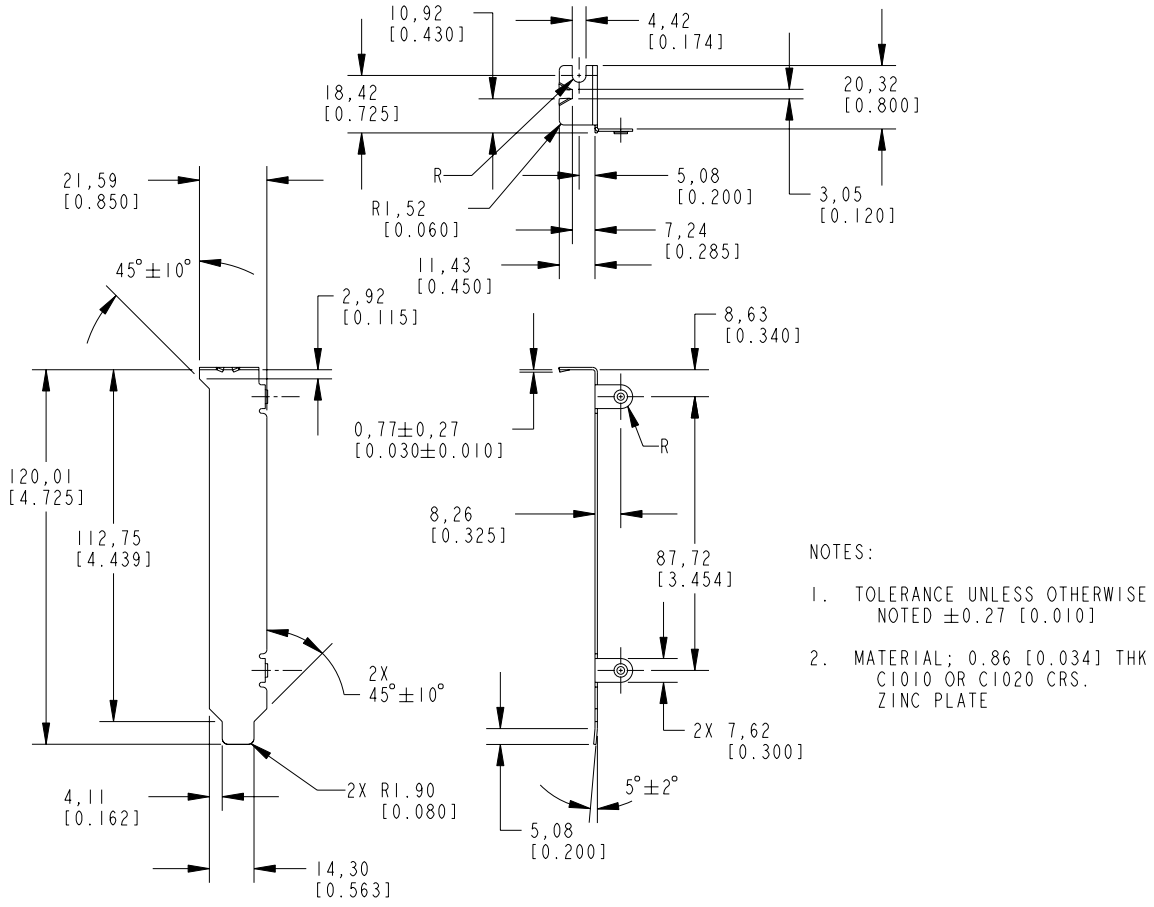


Figure 24 – Standard Full Height CNR I/O Bracket Dimensions

5.5.2 Communication and Networking Riser Low Profile I/O Bracket for the ATX Family

Figure 25 shows the recommended dimension information for the I/O bracket to be used with the low profile CNR board for use with the ATX family form factors.

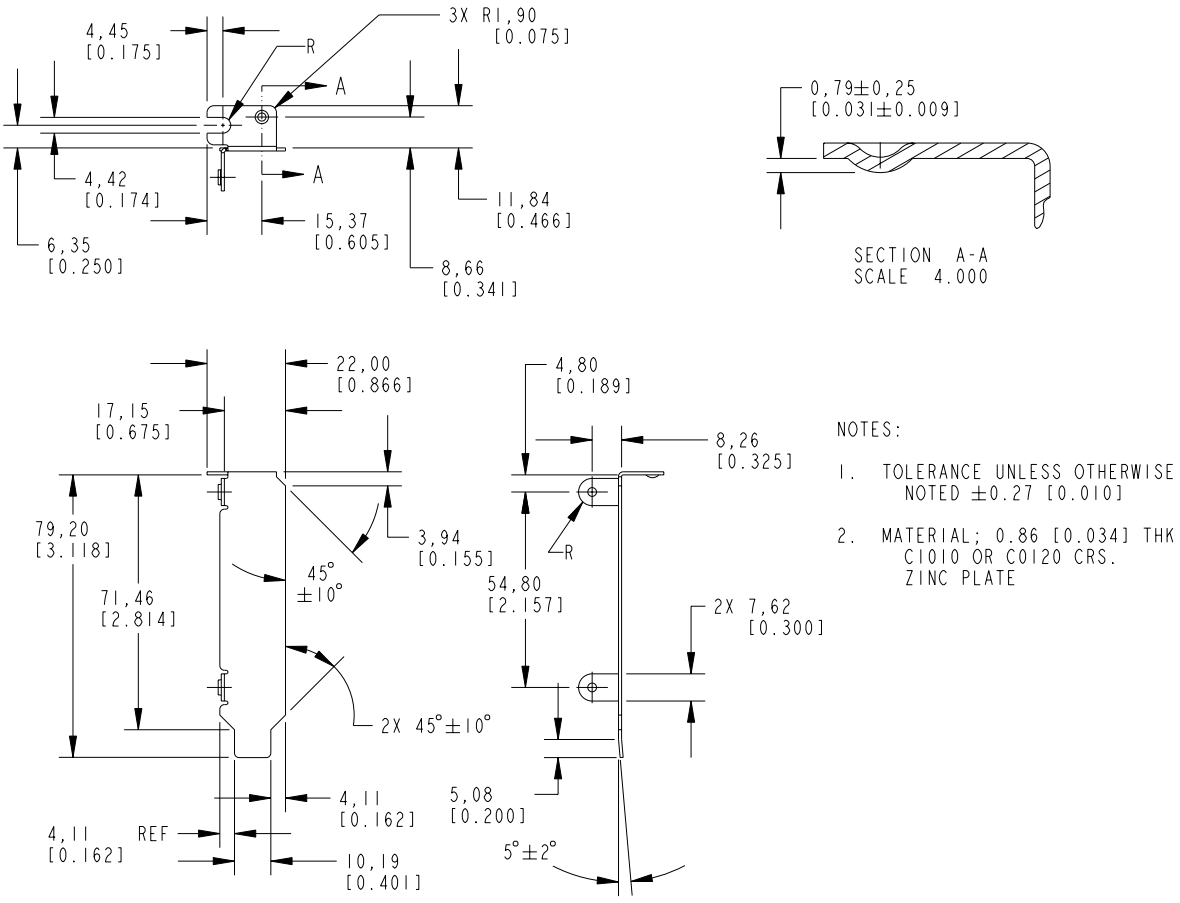


Figure 25 – Low Profile CNR I/O Bracket Dimensions

5.6 Thermal Requirements

It is the responsibility of the CNR board designer to ensure that the total power dissipation does not exceed a maximum of 25 watts under any and all modes of operation**.**

5.7 EMI, RFI, and Shielding Requirements

The CNR board contains several digital interfaces that are capable of generating EMI. It is important to ensure that the CNR is properly designed to allow a system containing the CNR board from receiving EMI compliance certifications (FCC, CISPR 22, etc.) This implies that the designer of the CNR board may need to provide appropriate filters (ferrite beads, capacitors, and/or common mode inductors) near the I/O bracket connectors, to aid in the reduction of EMI emissions.

6 Operating System, Software, and BIOS Requirements

A system implementing an architecture that uses CNR must always consider the CNR an extension of the motherboard. This implies that the BIOS has knowledge of riser capability on the motherboard, and as such, implements the appropriate routines to insure that all devices supplying the CNR interfaces are properly configured for Plug-and-Play support. Failure to provide these BIOS routines will risk an increase in the cost of support for the CNR enabled motherboard.

Driver development and validation are also considerations in CNR-based motherboard architectures. With the various interfaces available to a CNR board manufacturer, the opportunity to create multifunction solutions is readily available. This implies that there is also an opportunity for the drivers of these solutions to interact in undesirable ways, causing customer service calls. Thus, the CNR board vendor and the system integrator must perform adequate validation of the CNR board and its associated drivers

As mentioned previously, the BIOS is required to implement specific routines in support of Plug-and-Play for the CNR board. The following sections describe these required BIOS routines, as well as call progress, speakerphone, and telephone answering machine (TAM) support.

6.1 Operating System Plug-and-Play Requirements

Operating systems have improved dramatically through the past several years in their ability to use standardized register space to uniquely identify hardware added to a system and to appropriately install the driver(s) intended for the newly installed hardware. To insure that the same level of automation and that ease-of-use is maintained, the CNR **must** implement a Plug-and-Play (PnP) type interface that is compatible with today's operating systems. PCI based devices implement a standard style of PnP, with registers defined for Vendor ID (VID), Device ID (DID), Subsystem Vendor ID (SVID), and Subsystem ID (SID). Normally, the VID and DID are hardwired within the PCI device for each function implemented. However, since the analog front end (AFE) of the various functions available on the CNR can be provided by different suppliers, with each implementing the surrounding support circuitry in a different fashion, the SVID and SID need to be assigned on a solution by solution basis. Since the CNR carries the AFE solution, and various different vendors can provide this AFE solution, the SVID and SID information to properly install the correct drivers **must** be provided by the CNR.

The CNR PnP information will be provided to the operating system using a combination of a SMBus based EEPROM and specialized BIOS routines. The electrical interface to the EEPROM is provided on the CNR connector through the signals SMB_SCL, SMB_SDA, SMB_A2, SMB_A1, and SMB_A0. The following sections provide the required information for selecting, addressing, and programming the contents of the CNR Plug-and-Play EEPROM (PnP EEPROM).

As part of the PnP functionality of CNR, all PCI functions providing interfaces to the CNR must ensure that their SVID and SID registers remain unchanged across power transitions, as required by the PCI Local Bus Specification, Revision 2.2.

6.1.1 Plug and Play EEPROM Requirements

SMBus based EEPROM devices are very popular and available from several different manufacturers. The specific requirements for the EEPROM device are that it must be byte addressable and have its SMBus address configurable through external strapping for use on an SMBus that may contain one or more memory devices.

There are additional considerations, such as EEPROM size versus addressing that must be considered when selecting the PnP EEPROM device. These considerations and the electrical connectivity model are provided in the following two sections.

6.1.1.1 Plug and Play EEPROM Device Considerations

In general, SMBus EEPROM devices are available in a variety of memory sizes and addressing capabilities. Care must be taken by the designer of the CNR board to select an EEPROM device that is fully addressable and is large enough to store the required data. The following is a list of the specific requirements for the PnP EEPROM device.

1. The PnP EEPROM device **must** be compatible with the System Management Bus Specification, Revision 1.1 (dated December 11, 1998). Note that compliance with future versions of the System Management Bus Specification may cause hardware incompatibilities, until SMBus Controllers supporting the future specification releases are widely available.
2. The PnP EEPROM device must have three address pins available for configuring the device address. Note that SMBus EEPROM devices are available that have “No Connect” pins identified as address pins. All address pins must be active on the PnP EEPROM device.
3. The hardwired address (upper four bits of the device address) must be 1010b (Ah).
4. The memory contents must be configured as byte addressable (for example, 128 x 8).
5. The PnP EEPROM size must not exceed 256 bytes (2048 bits) in total size.

6. The CNR Board designer must allow PnP EEPROM device to be programmed through the CNR connector (i.e. EEPROM write protect must not be asserted).

6.1.1.2 Plug and Play EEPROM Connectivity

Implementing the PnP EEPROM in the CNR architecture requires consideration by both the motherboard designer and the CNR board designer. The motherboard designer is responsible for configuring the SMBus address of the CNR connector so that the address is not in conflict with other SMBus devices on the motherboard. The CNR board designer is responsible for ensuring an EEPROM device is selected that meets the requirements of Section 6.1.1.1, and that all EEPROM pins are properly routed to the CNR board edge contacts. The figure below shows a conceptual view of how the SMBus addresses could be assigned and configured for both the motherboard and the CNR.

Note: ***The CNR board designer must provide full address configuration of the Plug-and-Play EEPROM device through the strapping options provided by the motherboard. This implies that the SMBus EEPROM must expose all three address lines to the CNR connector for strapping.***

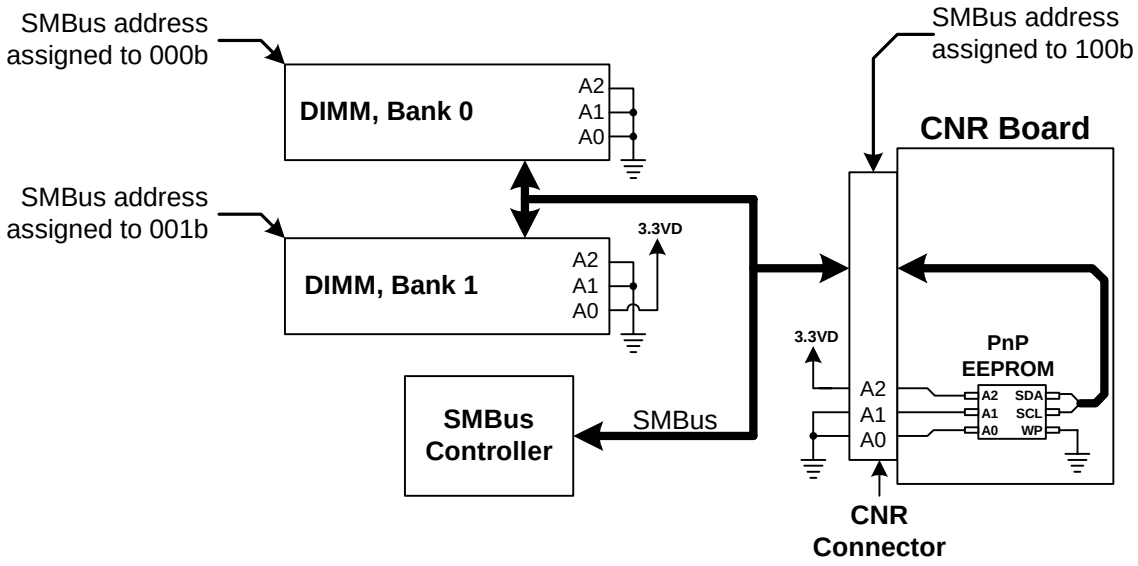


Figure 26 – SMBus Address Strapping

6.1.2 EEPROM Master Configuration Space

The CNR PnP EEPROM master configuration space is used by the BIOS to determine which functions are supported on the CNR. In addition, the master configuration space is used by BIOS to determine where in the PnP EEPROM the information for each function is located, along with the number of words (memory space) each function occupies. It is the responsibility of the BIOS to determine which functions are supported on the CNR, properly detect each function has the required AFE attached, extract that function's data from the PnP EEPROM, and finally place the extracted data within each function's register space, as required.

All registers in the master configuration space must be implemented. All reserved bits (denoted by *X*) must be programmed as a 0 (zero). All registers or bits in **bold** must be implemented and programmed.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
00h	**EEPROM ID**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**9249h**
02h	**EEPROM Size**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**SZ7**	**SZ6**	**SZ5**	**SZ4**	**SZ3**	**SZ2**	**SZ1**	**SZ0**	**N/A**
04h	**CNR Compliance**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0110h**
06h	**AC '97 Compliance**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0210h**
08h	**Function ID**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**LAN**	**SMB**	**USB**	**MDM**	**AUD**	**00xxh**
0Ah to 0Ch	**Reserved Registers**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**0000h**
0Eh	**Audio Pointer**	**AP15**	**AP14**	**AP13**	**AP12**	**AP11**	**AP10**	**AP9**	**AP8**	**AP7**	**AP6**	**AP5**	**AP4**	**AP3**	**AP2**	**AP1**	**AP0**	**0000h**
10h	**Modem Pointer**	**MP15**	**MP14**	**MP13**	**MP12**	**MP11**	**MP10**	**MP9**	**MP8**	**MP7**	**MP6**	**MP5**	**MP4**	**MP3**	**MP2**	**MP1**	**MP0**	**0000h**
12h	**USB Pointer**	**UP15**	**UP14**	**UP13**	**UP12**	**UP11**	**UP10**	**UP9**	**UP8**	**UP7**	**UP6**	**UP5**	**UP4**	**UP3**	**UP2**	**UP1**	**UP0**	**0000h**
14h	**SMBus Pointer**	**SP15**	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**0000h**
16h	**LAN Pointer**	**LP15**	**LP14**	**LP13**	**LP12**	**LP11**	**LP10**	**LP9**	**LP8**	**LP7**	**LP6**	**LP5**	**LP4**	**LP3**	**LP2**	**LP1**	**LP0**	**0000h**
18h to 2Ch	**Reserved Pointers**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**0000h**
2Eh	**Last Valid Address**	**LV15**	**LV14**	**LV13**	**LV12**	**LV11**	**LV10**	**LV9**	**LV8**	**LV7**	**LV6**	**LV5**	**LV4**	**LV3**	**LV2**	**LV1**	**LV0**	**N/A**
30h	**Checksum**	**CS15**	**CS14**	**CS13**	**CS12**	**CS11**	**CS10**	**CS9**	**CS8**	**CS7**	**CS6**	**CS5**	**CS4**	**CS3**	**CS2**	**CS1**	**CS0**	**N/A**

Table 18 – CNR EEPROM Master Configuration Space Register Map

6.1.2.1 EEPROM ID Register (Word 00h)

The EEPROM ID register contains a unique 16-bit value that identifies the PnP EEPROM as being present to the system. This is performed by providing a repeating pattern of binary digits (1001001001001001b).

6.1.2.2 EEPROM Size Register (Word 02h)

The EEPROM size register contains a 16-bit value that identifies the overall size of the PnP EEPROM. The BIOS uses this value to determine how to access the device for random memory reads. The table below shows the mapping of the actual PnP EEPROM size to the value entered in the EEPROM size register.

Memory Size (bytes)	EEPROM Size Register Value (hexadecimal)
N/A	0000h
2 bytes	0001h
4 bytes	0002h
8 bytes	0003h
16 bytes	0004h
32 bytes	0005h
64 bytes	0006h
128 bytes (1kbits)	0007h
256 bytes (2kbits)	0008h

Table 19 – PnP EEPROM Size Register Value Assignments

6.1.2.3 CNR Compliance Register (Word 04h)

The CNR compliance register provides a 16-bit value that identifies the version of the CNR specification the CNR board is compliant with. The BIOS uses this value to verify that the motherboard supports the features and version of the PnP EEPROM present on the CNR board.

The value of this register can be easily created by placing the numbers to the left of the decimal point in the CNR specification version number in the upper byte of the CNR compliance register (address 05h). Then, place the numbers to the right of the decimal point in the lower byte of the CNR compliance register (address 04h). See the example in Table 20.

The CNR Compliance Register must contain the value 0120h to be compliant with this release of the CNR Specification.

CNR Specification Version	CNR Compliance Register Upper Byte (Address 05h)	CNR Compliance Register Lower Byte (Address 04h)
0.80	00h	80h
1.00	01h	00h
1.10	01h	10h
1.20	**01h**	**20h**

Table 20 – CNR Specification Version Mapping in PnP EEPROM

6.1.2.4 AC '97 Compliance Register (Word 06h)

The AC '97 compliance register provides a 16-bit value that identifies the version of the AC '97 Specification BIOS must be compliant with, in order to properly access and determine the functionality of the codecs on the CNR board.

The value contained in this register is the oldest version of the AC '97 Specification that is interface and register set compatible with the current version of the AC '97 Specification. Table 21 provides examples of the values to be placed in the AC '97 Compliance Register, based on the currently released versions of the AC '97 Specification.

AC '97 Specification Version	Backward Compatible with AC '97 Specification Version	AC '97 Compliance Register Upper Byte (Address 07h)	AC '97 Compliance Register Lower Byte (Address 06h)
1.03	1.03	01h	03h
2.0	2.0	02h	00h
2.1	2.1	02h	10h
2.2	2.1	02h	10h

Table 21 – AC '97 Compliance Register Contents

6.1.2.5 Function ID Register (Word 08h)

The Function ID register is a 16-bit register that identifies the functions supported on the CNR board. Currently, bits 15 through 5 are reserved for future use (and must be programmed to zero); bits 4 through 0 address the current functionality of the CNR. If the bit in the register is set, then the corresponding function, or interface, is supported on the CNR board.

Note: Since the SMBus interface is required to be supported, for Plug-and-Play, the SMB bit (bit 3) of this register must always be set.

6.1.2.6 Reserved Registers (Words 0Ah – 0Ch)

The reserved registers are two 16-bit registers reserved for future expansion. They must always be programmed with zeros.

6.1.2.7 Audio Pointer Register (Word 0Eh)

The audio pointer register is a 16-bit register containing a pointer to the first address of the PnP EEPROM containing audio function data for the audio solution on the CNR board. The BIOS uses this value, along with the next non-zero pointer value, to determine how many words to read from the PnP EEPROM and place in the PCI configuration space for the audio function. If audio is not supported on the CNR board, then this register **must** be set to zero.

6.1.2.8 Modem Pointer Register (Word 10h)

The modem pointer register is a 16-bit register containing a pointer to the first address of the PnP EEPROM containing modem function data for the modem solution on the CNR board. The BIOS uses this value, along with the next non-zero pointer value, to determine how many words to read from the PnP EEPROM and place in the PCI configuration space for the modem function. If modem is not supported on the CNR board, this register **must** be set to zero.

6.1.2.9 USB Pointer Register (Word 12h)

The USB pointer register is a 16-bit register containing a pointer to the first address of the PnP EEPROM containing USB function data for the USB 2.0 solution on the CNR board. The BIOS uses this value, along with the next non-zero pointer value, to determine how many words to read from the PnP EEPROM for the USB function. If a USB function is not supported on the CNR board, then this register **must** be set to zero.

6.1.2.10 SMBus Pointer Register (Word 14h)

The SMBus pointer register is a 16-bit register containing a pointer to the first address of the PnP EEPROM containing SMBus function data for the SMBus solution on the CNR board. The BIOS uses this value, along with the next non-zero pointer value, to determine how many words to read from the PnP EEPROM for the SMBus function.

6.1.2.11 LAN Pointer Register (Word 16h)

The LAN pointer register is a 16-bit register containing a pointer to the first address of the PnP EEPROM containing LAN function data for the LAN solution on the CNR board. The BIOS uses this value, along with the next non-zero pointer value, to determine how many words to read from the PnP EEPROM for the LAN function. If LAN is not supported on the CNR board, this register **must** be set to zero.

6.1.2.12 Reserved Pointer Registers (Words 18h – 2Ch)

The reserved pointer registers are a total of 11 register locations that are reserved for future function support on the CNR board. These registers **must** be programmed with a 0000h value.

6.1.2.13 Last Valid Address Register (Word 2Eh)

The last valid address register contains the address of the last word of data used by the last non-zero pointer register. The BIOS uses this value to find the number of words that must be read from the PnP EEPROM for the last non-zero pointer.

6.1.2.14 Checksum Register (Word 30h)

The checksum register contains a 16-bit running sum of all bytes within the PnP EEPROM device, excluding the checksum register. While calculating the checksum, the carry from the 16th to the 17th bit is always ignored. The BIOS uses this value to determine if the PnP EEPROM contains valid data. If the checksum does not match, the BIOS should display an error message informing the user of a corrupted PnP EEPROM, before continuing the boot process.

6.1.3 Audio Function PnP EEPROM Contents

This area of the PnP EEPROM is reserved for the audio function. The intent of this section is that it contains the required Plug-and-Play information for the audio solution placed on the CNR board. The Plug-and-Play information contained in the PnP EEPROM for the audio function overrides any Plug-and-Play information for the audio solution installed on the motherboard.

Note: If the audio pointer register is non-zero, this section of the EEPROM must be implemented.

The starting address for the audio function data can be located anywhere within the PnP EEPROM memory space, as long as it starts at the address defined by the audio pointer register (word 0Eh). The audio function section uses a maximum of ten words (20 bytes).

The following table describes the contents of the audio function section of the PnP EEPROM.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
AP+000h	Audio CNR Vendor ID	AV15	AV14	AV13	AV12	AV11	AV10	AV9	AV8	AV7	AV6	AV5	AV4	AV3	AV2	AV1	AV0	0000h
AP+002h	Audio CNR Model ID	AM15	AM14	AM13	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	0000h
AP+004h	Multi-Chan. Signature	MC15	MC14	MC13	MC12	MC11	MC10	MC9	MC8	MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0	0000h
AP+006h	Multi-Chan Model ID	MM15	MM14	MM13	MM12	MM11	MM10	MM9	MM8	MM7	MM6	MM5	MM4	MM3	MM2	MM1	MM0	0000h
AP+008h	Codec 0 Output Physical Description	CP	AO	X	FCS	FSJP	RSJP	CLJP	HPJP	OOJP	JS03	JS02	JS01	JS00	SPOJ2	SPOJ1	SPOJ0	0000h
AP+00Ah	Codec 0 Input Physical Description	ATP	VTP	CDTP	PTP	LIJP	L2JP	MIJP	M2JP	OIJP	JS13	JS12	JS11	JS10	SPIJ2	SPIJ1	SPIJ0	0000h
AP+00Ch	Codec 1 Output Physical Description	CP	X	X	X	FSJP	RSJP	CLJP	HPJP	OOJP	JS03	JS02	JS01	JS00	SPOJ2	SPOJ1	SPOJ0	0000h
AP+00Eh	Codec 1 Input Physical Description	ATP	VTP	CDTP	PTP	LIJP	L2JP	MIJP	M2JP	OIJP	JS13	JS12	JS11	JS10	SPIJ2	SPIJ1	SPIJ0	0000h
AP+010h	Codec 2 Output Physical Description	CP	X	X	X	FSJP	RSJP	CLJP	HPJP	OOJP	JS03	JS02	JS01	JS00	SPOJ2	SPOJ1	SPOJ0	0000h
AP+012h	Codec 2 Input Physical Description	ATP	VTP	CDTP	PTP	LIJP	L2JP	MIJP	M2JP	OIJP	JS13	JS12	JS11	JS10	SPIJ2	SPIJ1	SPIJ0	0000h

Table 22 – CNR EEPROM Audio Section Register Map

6.1.3.1 Audio CNR Vendor ID (Word AP+000h)

The audio CNR vendor register contains a 16-bit value that is placed into the subsystem vendor ID register of the PCI configuration space for the audio function. The actual value placed in this location should be the PCI Special Interest Group (SIG) assigned vendor ID for the company that is providing the completed CNR board and drivers. The physical address for this data begins at the address defined by the audio pointer register.

6.1.3.2 Audio CNR Model ID (Word AP+002h)

The audio CNR model ID register contains a 16-bit value that is placed into the subsystem ID register of the PCI configuration space for the audio function. The actual value placed in this location should be a unique (to the manufacturer of the CNR board) model number. The physical address for this data begins at the address defined by the audio pointer register plus 002h.

6.1.3.3 CNR Multi-Channel Signature (Word AP+004h)

The CNR multi-channel signature register contains a 16-bit value that is used by the motherboard routines to determine if the CNR installed, for the purpose of multi-channel audio upgrades, is compliant with the motherboard it is being plugged into. The BIOS compares the contents of this register, with a value provided by the codec manufacturer placed on the motherboard to determine compatibility. The physical address for this register begins at the address defined by the audio pointer register plus 004h.

See Section 6.2.6 for more details on the implementation of the BIOS algorithms used with this register.

Refer to Section 6.3.2 for detailed descriptions of the creation of the CNR multi-channel signature.

6.1.3.4 Audio Multi-Channel Model ID (Word AP+006h)

The audio multi-channel model ID register contains a 16-bit value that is placed into the subsystem ID register of the PCI configuration space for the audio function, when the audio multi-channel audio signature register matches the value stored in the motherboard BIOS. The actual value placed in this location should be a unique (to the manufacturer of the CNR board) model number. The physical address for this register begins at the address defined by the audio pointer register plus 006h.

6.1.3.5 Codec x Output Physical Description

A copy of the Codec x Output Physical Description register is present for each of the three possible codecs. Word AP+008h contains information for Codec 0, Word AP+00Ch contains information for Codec 1, and Word AP+010h contains information for Codec 2. Each of the fields within the two registers for each codec defines the physical connections on the CNR card for the relevant codec. The exception to this is the 'AO' and 'FCS' bits, which are only present in the Codec 0 Output Physical Description register, and refers to the CNR audio subsystem as a whole.

CP - Bit 15 - Output Field Valid. A '1' indicates that a codec is present, and the values in this register and the corresponding *Codec x Input Physical Description* register are valid. A '0' indicates that this register and the *Codec x Input Physical Description* register have not been programmed.

AO - Bit 14 - Augment Only. This bit indicates whether the audio portion of this CNR is intended to augment capabilities on the motherboard. A '1' indicates that this board is intended to augment an existing motherboard audio solution, and is not intended for use as the only audio in the system. CNRs without Microphone In or Line In jacks present should generally have this bit set. This bit is only present in the Codec 0 Output Physical Description register, and is reserved in the Codec 1 or Codec 2 Ouput Physical Description registers.

FCS - Bit 12 - Fast Codec Start. This bit, when set (1), indicates that the AC'97 codecs present on the CNR are known to have the Codec Ready bit set within 800µs of the deassertion of AC_RESET#. This bit is only present in the Codec 0 Output Physical Description register, and is reserved in the Codec 1 or Codec 2 Ouput Physical Description registers.

FSJP - Bit 11 - Front Speaker Jack Present. This bit, when set (1), indicates that an external jack for Front Speaker output is present and connected to Codec *x.*

RSJP - Bit 10 - Rear Speaker Jack Present. This bit, when set (1), indicates that an external jack for Rear (Surround) Speaker ouput is present and connected to Codec *x.*

CLJP - Bit 9 - Center/LFE Jack Present. This bit, when set (1), indicates that an external jack for Center and Low Frequency Enchancement is present and connected to Codec *x*.

HPJP - Bit 8 - Headphone Jack Present. This bit, when set (1), indicates that an external jack for Headphone is present and connected to Codec *x* **.**

OOJP - Bit 7 - Other Output Jack(s) Present. A one in this bit indicates that there are other audio output jacks not described above on the CNR connected to Codec *x*.

JS0[3:0] Bits 6-3: Jack Sense 0 Connection. Indicates which jack the JS0 sense pin on Codec *x* is connected to, if any.

Physical Connector	JS0C[3:0]
None	0h
Line In	1h
Microphone	2h
Microphone In 2	3h
Front Speaker	4h
Rear/Surround Speaker	5h
Center/LFE Speaker	6h
Headphone Jack	7h
SPDIF Input Jack	8h
SPDIF Output Jack	9h
Reserved	Ah
Reserved	Bh
Reserved	Ch
Reserved	Dh
Other Input Jack(s)	Eh
Other Output Jack(s)	Fh

Table 23 – Jack Sense Connection

SPOJ[2:0]- Bits 2:0 - The SPOJ[2:0] bits indicate the type of S/P-DIF output connector present on the CNR attached to Codec *x*, if any. This field is used by the driver and other software to help the user with properly configuring and connecting their system, and can be used to provide the correct help instructions to the end user.

Physical Connector	SPOJ[2:0]
No SPDIF connector	000
Reserved	001
Reserved	010
Discrete Ouput SPDIF Optical	011
Discrete Ouput SPDIF Coax	100
SPDIF Output Optical shared with Line Out	101
SPDIF Output Optical shared with Mic In	110
SPDIF Output Optical shared with Other	111

Table 24 – S/P-DIF Output Configuration

6.1.3.6 Codec x Input Physical Description

Word AP+00Ah contains information for Codec 0, Word AP+00Eh contains information for Codec 1, and Word AP+012h contains information for Codec 2.

ATP - Bit 15 - AUX ATAPI Connector Present. This bit, when set (1), indicates that an ATAPI connector for AUX input is present and connected to Codec *x*

VTP - Bit 14 - Video ATAPI Connector Present. This bit, when set (1), indicates that an ATAPI connector for Video input is present and connected to Codec *x*.

CDTP - Bit 13 - CD ATAPI Connector Present. This bit, when set (1), indicates that an ATAPI connector for CD input is present and connected to Codec *x*.

PTP - Bit 12 - Phone ATAPI Connector Present. This bit, when set (1), indicates that an ATAPI connector for Phone input is present and connected to Codec *x*.

LIJP - Bit 11 - Line In Jack Present. This bit, when set (1), indicates that an external jack for Line In input is present connected to Codec *x*.

L2JP - Bit 10 - Line In 2 Jack Present. This bit, when set (1), indicates that an external jack for Line In 2 input is present and connected to Codec *x*.

MJP - Bit 9 - Microphone In Jack Present. This bit, when set (1), indicates that an external jack for Microphone In input is present and connected to Codec *x*.

M2JP - Bit 8 - Microphone 2 Jack Present. This bit, when set (1), indicates that an external jack for Microphone In 2 input is present connected to Codec *x*.

OIJP - Bit 7 - Other Input Jack(s) Present. A one in this bit indicates that there are other audio input jacks not described above on the CNR connected to Codec *x*.

JS1[3:0] Bits 6-3: Jack Sense 0 Connection. Indicates which jack the JS0 sense pin on Codec *x* is connected to, if any. The bit definitions for this field are the same as for the JS0[3:0] field, above.

SPIJ[2:0] Bits 2-0: The SPIJ[2:0] bits indicate the type of S/P-DIF input connector present on the CNR connected to Codec *x*, if any. This field is used by the driver and other software to help the user with properly configuring and connecting their system, and can be used to provide the correct help instructions to the end user.

Physical Connector	SPIJ[2:0]
No SPDIF connector	000
Reserved	001
Reserved	010
Discrete Input SPDIF Optical	011
Discrete Input SPDIF Coax	100
SPDIF Input Optical shared with Microphone	101
SPDIF Input Optical shared with Line In	110
SPDIF Input Optical shared with Other	111

Table 25 – S/P-DIF Input Configuration

6.1.4 Modem Function PnP EEPROM Contents

This area of the PnP EEPROM is reserved for the modem function. The intent of this section is that it contains the required Plug-and-Play information for the audio solution placed on the CNR board.

Note: If the modem pointer register is non-zero, then this section of the EEPROM must be implemented.

The starting address for the modem function data can be located anywhere within the PnP EEPROM memory space, as long as it starts at the address defined by the modem pointer register (word 10h). The modem function section uses a maximum of two words (four bytes).

The following table describes the contents of the modem function section of the PnP EEPROM.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
MP+000h	**Modem CNR Vendor ID**	**MV15**	**MV14**	**MV13**	**MV12**	**MV11**	**MV10**	**MV9**	**MV8**	**MV7**	**MV6**	**MV5**	**MV4**	**MV3**	**MV2**	**MV1**	**MV0**	**0000h**
MP+002h	**Modem CNR Model ID**	**MM15**	**MM14**	**MM13**	**MM12**	**MM11**	**MM10**	**MM9**	**MM8**	**MM7**	**MM6**	**MM5**	**MM4**	**MM3**	**MM2**	**MM1**	**MM0**	**0000h**

Table 26 – CNR EEPROM Modem Section Register Map

6.1.4.1 Modem CNR Vendor ID (Word MP+000h)

The modem CNR vendor ID register contains a 16-bit value that is placed into the subsystem vendor ID register of the PCI configuration space for the modem function. The actual value placed in this location should be the PCI SIG assigned vendor ID for the company that is providing the completed CNR board and drivers. The physical address for this data begins at the address defined by the modem pointer register.

6.1.4.2 Modem CNR Model ID (Word MP+002h)

The modem CNR model ID register contains a 16-bit value that is placed into the subsystem ID register of the PCI configuration space for the modem function. The actual value placed in this location should be a unique (to the manufacturer of the CNR board) model number. The physical address for this data begins at the address defined by the modem pointer register plus 002h.

6.1.5 USB Function PnP EEPROM Contents

This area of the PnP EEPROM is reserved for the USB function. The intent of this section is that it contains the information required to identify basic USB feature set support to the BIOS.

Note: If the USB pointer register is non-zero, then this section of the EEPROM must be implemented.

The starting address for the USB function data can be located anywhere within the PnP EEPROM memory space, as long as it starts at the address defined by the USB pointer register (word 12h). The USB function section uses a maximum of two words (four bytes).

The following table describes the contents of the USB function section of the PnP EEPROM.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
UP+000h	**USB Option Register**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**UPC2**	**UPC1**	**UPC0**	**PINF**	**HUB**	**WKE**	**SPD1**	**SPD0**	**000xh**
UP+002h	**USB Compliance**	**UC15**	**UC14**	**UC13**	**UC12**	**UC11**	**UC10**	**UC9**	**UC8**	**UC7**	**UC6**	**UC5**	**UC4**	**UC3**	**UC2**	**UC1**	**UC0**	**0000h**
UP+004h	**USB Port Information**	**UI15**	**UI14**	**UI13**	**UI12**	**UI11**	**UI10**	**UI9**	**UI8**	**UI7**	**UI6**	**UI5**	**UI4**	**UI3**	**UI2**	**UI1**	**UI0**	**0000h**

Table 27 – CNR EEPROM USB Section Register Map

6.1.5.1 USB Option Register (Word UP+00h)

The USB option register contains individual bits that describe the various functions supported by the USB function implemented on the CNR board. The following paragraphs describe each bit implemented in the USB option register. Note that bits 15 through 8 are reserved for future use and must be programmed as zero.

UPC[2:0] - Bits 7, 6 ,5 - These bits are used to indicate the number of physical USB ports that are implemented on the CNR board. They are defined as follows:

Number of Ports	UPC2	UPC1	UPC0
Undefined	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
Undefined	1	0	1
Undefined	1	1	0
Undefined	1	1	1

Table 28 – UPCx Bits and Number of Ports

Notes:

A. These bits are only valid when ***PINF*** is set (1).

B. As per this specification (4.1.3.2 USB Power Management), USB hubs may only support a maximum of 4 ports.

PINF - Bit 4 - When set (1) indicates that the USB port or USB hub ports implemented on the CNR board has additional port information as described by the USB Port Information register (Word UP+004h). When cleared (0), indicates that the USB port or hub if implemented on the CNR board does not have additional port information. A CNR board design should set this bit only if informaton pertaining to a USB port or USB hub on the CNR board cannot be retrieved or set by system software (BIOS and/or OS) through the use of standard USB methods. An IHV should set (1) this bit if:

- A USB port on the CNR board uses a proprietary connector (i.e not series “A”, “B” or “mini-B”).
- A USB hub on the CNR board can only provide system software with the number of physical downstream ports supported by hub but not the actual number usable by an end-user (e.g. The CNR board vendor may choose to expose to the end user, a number of ports that is less than that physically support by the USB Hub silicon. Some USB hub implementations cannot report this condition)
- A USB port is available to the end user, but as a dedicated device (e.g. a device is attached to the USB port(s) and cannot be removed by the end-user)

HUB - Bit 3 - When set (1), indicates that the USB function implemented on the CNR board is a USB 2.0 hub device. When cleared (0), indicates that the USB function implemented on the CNR board is a function other than a USB 2.0 hub device. The BIOS uses this bit, in conjunction with the other bits in the USB option register, to determine if the motherboard is capable of supporting the functions implemented on the CNR board.

WKE - Bit 2 - When set (1), indicates that the USB 2.0 function on the CNR Board is capable of supporting a wake-up event. When cleared (0), the USB function on the CNR board is not capable of supporting a wake-up event. The BIOS uses this bit to determine if the motherboard can support a wake event from the USB 2.0 function on the CNR board, from both motherboard feature support and power supply delivery standpoints.

SPD[1:0] - Bits 1 and 0 - These bits indicate the version of USB required to support the USB function on the CNR Board (whether a hub, device, function or port). For example, if the function on the CNR card required USB 2.0 high-speed mode to meet the advertised functionality, then the SPD bits would be set to 01b. The BIOS must verify that the USB signals routed to the CNR connector support the same USB mode called out by the SPD[1:0] bits. The following table describes the SPD[1:0] bit assignments for the various modes of USB 2.0.

USB Version	SPD1	SPD0
USB Full-speed/Low-speed	0	0
USB High-speed	0	1
Reserved	1	0
Reserved	1	1

Table 29 – USB Version to SPD Bit values

6.1.5.2 USB Compliance Register (Word UP+002h)

The USB Compliance Register provides a 16-bit value that BIOS must be compliant with USB Specification 2.0, in order to properly access and determine the functionality of the USB devices or functions on the CNR card.

USB Specification 2.0 adds the high-speed transfer rate, while maintaining full compatibility with existing full-speed/low-speed specifications. Transfer rate capability is dependant upon the USB host controller. Table 30 provides the values to be placed in the USB Compliance Register.

USB Specification Version	Backward Compatible with USB Specification Version	USB Compliance Register Upper Byte (Address UP+003h)	USB Compliance Register Lower Byte (Address UP+002h)
2.0	1.0	01h	00h

Table 30 – USB Compliance Register Contents

6.1.5.3 USB Port Information Register (Word UP+04h)

The USB Port Information register is used to describe the characteristics associated with each USB port implemented on the CNR board:

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
UP+004h	**USB Port Information**	**UI15**	**UI14**	**UI13**	**UI12**	**UI11**	**UI10**	**UI9**	**UI8**	**UI7**	**UI6**	**UI5**	**UI4**	**UI3**	**UI2**	**UI1**	**UI0**	**0000h**

Table 31 – USB Port Information Register

The Port Information register is only valid if ***PINF*** in the USB Option Register (Word UP+000h) is set (1). The USB Port Information register has the capability of providing information for a maximum of 4 USB ports.

The USB Port Information bits are used to describe certain characteristics associated with each USB port implemented on the CNR board. These information bits apply to both hub and non-hub implementations.

The following table defines the bits used in each nibble of the USB Port Information register to describe USB port information:

Bit	Description
3	This bit is reserved for future features and must be zero
2	This bit is reserved for future features and must be zero
1	**Port_Non_Visible (VIS):** If this bit is clear (0), then the port is visible[8] to the end user. If set (1), then the port is not visible[9] to the end user. Note that if system software detects that a device is attached to a non-visible port, system software should consider the device to be user non-removable.
0	**Proprietary_Port (PRO):** If this bit is set (1), then the port implements a proprietary receptacle (e.g. not series "A", "B", or "mini-B" receptacle) or is terminated at a function/device on the CNR card. If this bit is clear (0), then the implemented port receptacle is a standard series "A", "B", or "mini-B".

Table 32 – USB Port Information Register- Bit Definitions

Each nibble (4 bits) in the Port Information register is used to describe a USB port, where the ports are mapped as follows:

Nibble 1 (bits 3:0) of the Port Information register is used to describe:

- Information for a single port (***USB Option Register.Hub*** .eq. 0) implementation
- Information for port 1 of a multi-port (***USB Option Register.Hub*** .eq. 1) implementation

Nibbles 2, 3 and 4 (bits 15:4) are used to describe USB ports 2, 3 and 4 (multi-port implementation).

Table 30 illustrates the required port mapping for a CNR board that has a single port implementation:

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)							
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
UP+004h	**USB Port Information**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**VIS**	PRO

Nibble 1/Port 1

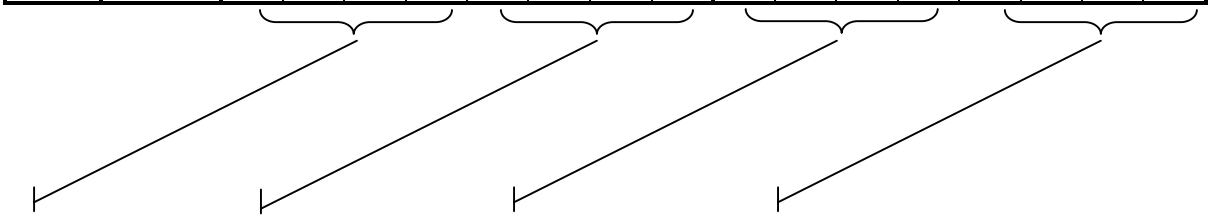
Table 33 – USB Port Mapping Example – 1 Port

Table 31 illustrates the required port mapping for a CNR board that has a multi-port implementation:

[8] A user visible USB port is one in which the end user *can* see a physical USB port receptacle and can connect devices to and disconnect devices from the receptacle.

[9] A user non-visible USB port is one in which the end user *cannot* see a physical port receptacle or one where a device is connected to a port receptacle but the user is unable to physically connect or disconnect the device.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)							
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
UP+004h	**USB Port Information**	**0**	**0**	**VIS**	**PRO**	**0**	**0**	**VIS**	**PRO**	**0**	**0**	**VIS**	**PRO**	**0**	**0**	**VIS**	**PRO**



Nibble 4/Port 4 Nibble 3/Port 3 Nibble 2/Port 2 Nibble 1/Port 1

Table 34 – USB Port Mapping Example – 2:4 Ports

Notes:

A. These bits are valid only when ***PINF*** is set (1).

B. A maximum of 4 ports may be mapped.

C. For single port implementations, bits 15:4 are undefined and must be zero.

D. For single and multi-port implementations, when ***PINF*** is clear (0) it is assumed that the implemented USB port(s) use a non-proprietary connector and are user-visible.

6.1.6 SMBus Section EEPROM Map

This area of the PnP EEPROM is reserved for the SMBus function. This section describes the information required to identify basic SMBus feature set support to the BIOS.

Note: If the SMBus pointer register is zero, then this section of the EEPROM must not be implemented.

The starting address for the SMBus function data can be located anywhere within the PnP EEPROM memory space, as long as it starts at the address defined by the SMBus pointer register (word 14h). The SMBus function section uses a maximum of three words (six bytes).

The following table describes the contents of the SMBus function section of the PnP EEPROM.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
SP+000h	**SMBus Com;;pliance**	**SC15**	**SC14**	**SC13**	**SC12**	**SC11**	**SC10**	**SC9**	**SC8**	**SC7**	**SC6**	**SC5**	**SC4**	**SC3**	**SC2**	**SC1**	**SC0**	**0100h**
SP+002h to SP+004h	**SMBus Function Reserved**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0000h**

Table 35 – CNR EEPROM SMBus Section Register Map

6.1.6.1 SMBus Compliance Register (Word SP+000h)

The SMBus Compliance Register provides a 16-bit value that identifies the version of the SMBus Specification BIOS must be compliant with, in order to properly access and determine the functionality of the SMBus PnP EEPROM device on the CNR board.

The value contained in this register must indicate System Management Bus, Revision 1.1 (dated December 11, 1998). Table 36 provides examples of the values to be placed in the SMBus Compliance Register, based off the released versions of the SMBus Specification.

CNR does not currently support Revision 2.0 of the System Management Bus Specification due to the limited availability of SMBus 2.0 devices and controllers. When SMBus 2.0 devices and controllers are widely available, the CNR Specification will be revised, if necessary.

SMBus Specification Version	Backward Compatible with SMBus Specification Version	SMBus Compliance Register Upper Byte (Address SP+001h)	SMBus Compliance Register Lower Byte (Address SP+000h)
1.0	1.0	01h	00h
1.1	1.0	01h	00h
2.0	2.0	02h	00h

Table 36 – SMBus Compliance Register Contents

6.1.6.2 SMBus Function Reserved (Words SP+002h to SP+004h)

These two reserved 16-bit words occupy the address space starting at the SMBus pointer register plus 002h, and continuing up to the SMBus pointer register plus 004h. The SMBus function reserved registers are reserved for future feature support.

6.1.7 LAN Section EEPROM Map

This area of the PnP EEPROM is reserved for the LAN function. This section describes the information required to identify basic LAN feature set support to the BIOS.

Note: If the LAN pointer register is non-zero, then this section of the EEPROM must be implemented.

The starting address for the LAN function data can be located anywhere within the PnP EEPROM memory space, as long as it starts at the address defined by the LAN pointer register (word 16h). The LAN function section uses a maximum of four words (eight bytes).

The following table describes the contents of the LAN function section of the PnP EEPROM.

		Odd Byte Address (i.e. 01h)								Even Address (i.e. 00h)								
Register Address	**Register Name**	**Bit 15**	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Default Value**
LP+000h	**LAN Option Register**	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	***X***	**INTF**	**0000h**
LP+002h	**LAN CNR Vendor ID**	**LV15**	**LV14**	**LV13**	**LV12**	**LV11**	**LV10**	**LV9**	**LV8**	**LV7**	**LV6**	**LV5**	**LV4**	**LV3**	**LV2**	**LV1**	**LV0**	**0000h**
LP+004h	**LAN CNR Model ID**	**LM15**	**LM14**	**LM13**	**LM12**	**LM11**	**LM10**	**LM9**	**LM8**	**LM7**	**LM6**	**LM5**	**LM4**	**LM3**	**LM2**	**LM1**	**LM0**	**0000h**
LP+006h	**LAN Compliance**	**LC15**	**LC14**	**LC13**	**LC12**	**LC11**	**LC10**	**LC9**	**LC8**	**LC7**	**LC6**	**LC5**	**LC4**	**LC3**	**LC2**	**LC1**	**LC0**	**0000h**

Table 37 – CNR EEPROM LAN Section Register Map

6.1.7.1 LAN Option Register (Word LP+000h)

The LAN option register contains individual bits that describe the required support of the LAN function implemented on the CNR board. The following paragraphs describe each bit implemented in the LAN option register. Note that bits 15 through 1 are reserved for future use and must be programmed as zero. The physical address for this data begins at the address defined by the LAN pointer register.

INTF - Bit 0 - When set (1), indicates that the LAN function implemented on the CNR board requires the MII (17-pin) interface (as implemented on the Type B CNR connector). When cleared (0), indicates that the LAN function implemented on the CNR board requires the eight-pin LAN Interface (as implemented on the Type A CNR connector). The BIOS uses this bit to determine if the motherboard is capable of supporting the LAN interface required by the LAN device implemented on the CNR board.

6.1.7.2 LAN CNR Vendor ID (Word LP+002h)

This register contains a 16-bit value that indicates the subsystem vendor ID register of the LAN function. The actual value placed in this location should be the PCI SIG assigned vendor ID number for the company that is providing the completed CNR board and drivers. The physical address for this data begins at the address defined by the LAN Pointer Register plus 002h.

6.1.7.3 LAN CNR Model ID (Word LP+004h)

This register contains a 16-bit value that indicates the subsystem ID register of the LAN function. The actual value placed in this location should be a unique (to the manufacturer of the CNR board) model number. The physical address for this data begins at the address defined by the LAN pointer register plus 004h.

6.1.7.4 LAN Compliance Register (Word LP+006h)

The LAN Compliance Register provides a 16-bit value that identifies the LAN Interface version that BIOS must be compliant with, in order to properly access and determine the functionality of the LAN components on the CNR board.

The value contained in this register is dependent on the LAN Interface used by the LAN device on the CNR. In the case of the eight-bit LAN Interface, the value is related to the Intel I/O Controller Hub (ICH) chip, while in the case of the seventeen-bit LAN Interface the value is related to the release date of the IEEE802.3(u) Standard. Table 38 and Table 39 provide examples of the values to be placed in the LAN Compliance Register. Refer to the manufacturers chipset design information for the appropriate value for this register.

Intel I/O Controller Hub Part Number	*LAN Compliance Register Upper Byte (Address LC+003h)*	*LAN Compliance Register Lower Byte (Address LC+002h)*
82801AA	*00h*	*00h*
82801AB	*00h*	*00h*
82801BA	*02h*	*A0h*
82801CA	*02h*	*A0h*

Table 38 – LAN Compliance Register Contents for eight-pin LAN Interface

IEEE802.3u Specification Version Date	*LAN Compliance Register Upper Byte (Address LC+003h)*	*LAN Compliance Register Lower Byte (Address LC+002h)*
June 14, 1995	*00h*	*A4h*
September 28, 1998	*05h*	*56h*

Table 39 – LAN Compliance Register Contents for seventeen-pin LAN Interface (MII)

6.2 BIOS Requirements

Support of a CNR connector on the motherboard requires specialized BIOS support. This BIOS support must perform a variety of functions specifically for the CNR, including identifying the CNR board presence, downloading information from the PnP EEPROM on the CNR board, and finally configuring, enabling, or disabling each of the interfaces and attached devices. The flowcharts below, along with the following sections, describe the motherboard BIOS requirements in support of a CNR Connector.

Note: ***There are several places, in the following sections, where the BIOS is required to place an error message on the display. Following each message the text "Press F1 to Resume" is also displayed. The BIOS is required to pause and wait for the F1 key to be pressed prior to continuing with the boot process.Note: The exact algorithms or timing requirements to perform specific functions on the SMBus interface are not provided. It is the responsibility of the BIOS developer to determine the proper method to access and implement standard SMBus functions called out in this specification.***

Note: ***Several bits within the PnP EEPROM space are defined as "reserved for future use" and thus programmed with a 0 (zero) value. The BIOS must not rely on these reserved bits always being zero. BIOS must mask all reserved bits during a read from the PnP EEPROM. This requirement is in place to insure forward compatibility with future versions of the CNR specification.***

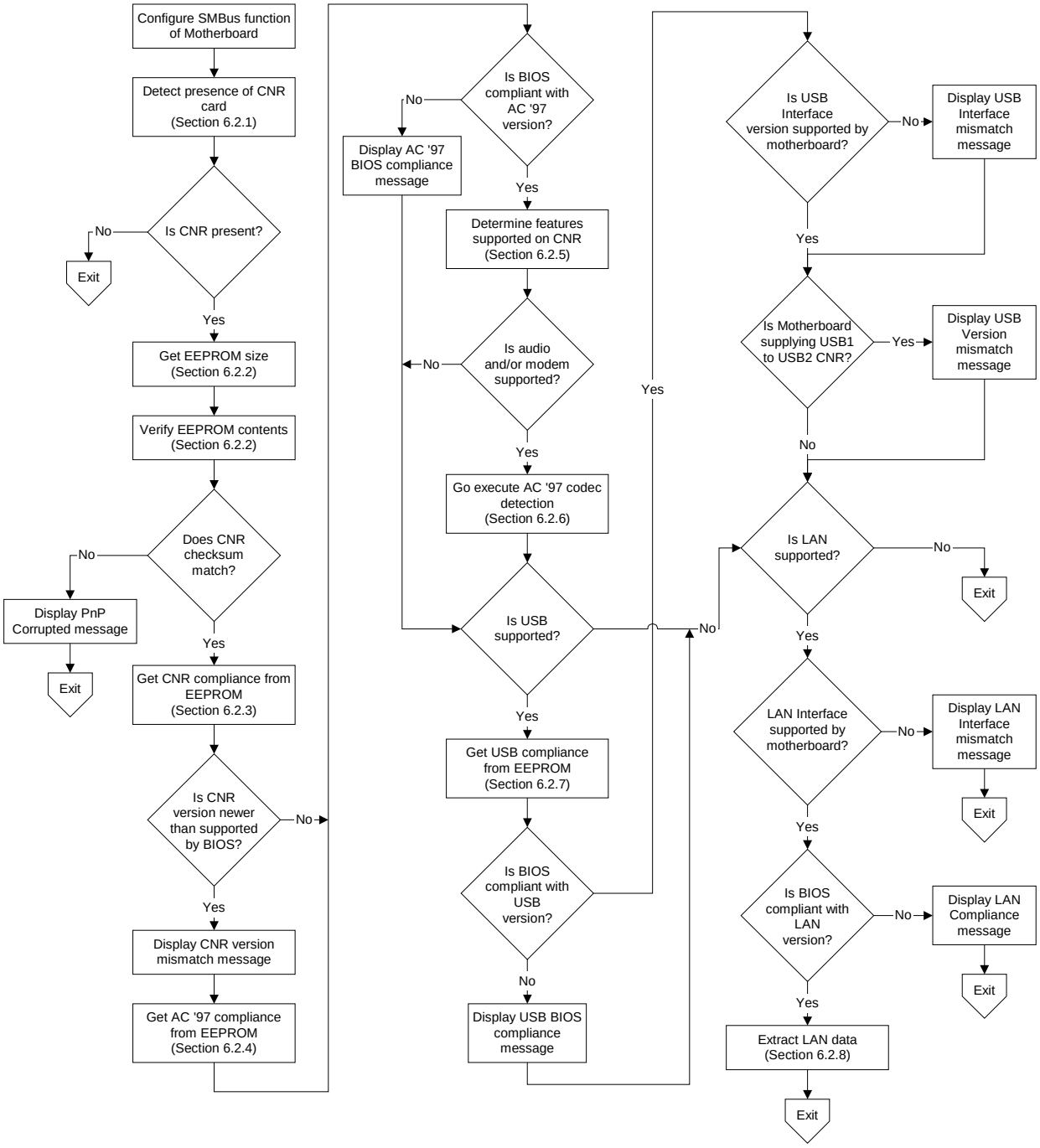


Figure 27 – BIOS Functional Flowchart for CNR Plug-and-Play Support

6.2.1 CNR Presence Detection

To determine the presence of the CNR board, the BIOS must perform the following steps:

1. Configure the motherboard SMBus controller.
2. Read the first two bytes (byte addresses 00h and 01h) of data from the PnP EEPROM, and concatenate them together, in the order read.
3. The result of the read and concatenation should be 4992h.
4. If the result is not 4992h, then the CNR is not installed, and the BIOS should continue with the regular boot process, by detecting motherboard devices.

6.2.2 Verifying PnP EEPROM Contents

Before the BIOS can use the data contained in the PnP EEPROM, it must first verify that the contents have not been corrupted. This accomplished by performing a checksum on the entire contents of the PnP EEPROM, as follows.

The following steps assume that the presence of the CNR PnP EEPROM has been verified.

1. Sequentially read the entire contents of the PnP EEPROM (starting at address 00h), adding each byte read to the previous byte(s) read. (The addition process is to be performed using a 16-bit register. Any carries out of the 16 bit are ignored).
2. During the sequential read process, the contents of byte addresses 02h – 03h **must** both added to the checksum calculation, as well as concatenated (with address 03h as the most significant byte). The resulting word can then be used to determine the overall size of the PnP EEPROM (refer to Table 19).
3. Byte addresses 30h and 31h must not be included in the checksum calculation. Instead the values stored in these two locations should be concatenated (with address 31h as the most significant byte) and stored for later use.
4. After all bytes have been read (and summed) from the PnP EEPROM, the resulting value is to be compared to the previously extracted and stored checksum (from Step #3).
5. If the calculated and extracted checksum values match the BIOS can continue on to the next step.
6. If the calculated and extracted checksum values do not match, the BIOS ***must*** display the message "**CNR Plug-and-Play EEPROM contents are damaged. CNR devices will be ignored. *Press F1 to Resume*.**", indicating that a checksum error exists on the installed riser board. After the F1 key is pressed, the BIOS should not continue with CNR routines, since the PnP EEPROM is corrupted.

6.2.3 Verifying CNR Version Compliance

After verifying that the PnP EEPROM contents are valid, through a checksum comparison, the BIOS must then make sure that it is capable of understanding the contents of the EEPROM. The BIOS performs this function by checking the CNR compliance register of the PnP EEPROM, as follows

The following steps assume that the previous steps have been completed.

1. Perform a direct read of addresses 04h and 05h. Concatenate the contents of these two registers together (with contents of address 05h as the most significant byte).
2. The BIOS must then compare the resulting number to the version of the CNR specification it is compliant with.
3. If the resulting number is greater than the BIOS CNR version number, the error message "**CNR version newer than motherboard. Some CNR functionality may be lost. *Press F1 to Resume***" ***must*** be displayed, indicating the version mismatch. After the F1 key is pressed, the remaining CNR configuration process can then continue, but features available with the newer version CNR PnP EEPROM will not be implemented. In other words, the BIOS will only be able to implement features of which it has knowledge.
4. If the resulting number is less than the BIOS CNR version number, then backwards compatibility can be assumed, and the BIOS can continue with the boot process.

6.2.4 Verifying AC '97 Version Compliance

After verifying compatibility with the PnP EEPROM version, the BIOS must then make sure that it understands the version of AC '97 required by the codecs on the CNR. The BIOS performs this function by checking the AC '97 Compliance Register of the PnP EEPROM, as follows

The following steps assume that the previous steps have been completed.

1. Perform a direct read of addresses 06h and 07h. Concatenate the contents of these two registers together (with contents of address 07h as the most significant byte).
2. The BIOS **must** then verify that it is compliant with the resulting AC '97 Compliance value.
3. If BIOS is not compliant with the resulting number the error message "**BIOS not compliant with the AC '97 devices on CNR. AC '97 functionality of CNR disabled.** ***Press F1 to Resume.***" ***must*** then be displayed indicating the version mismatch, and the audio and modem functions of the AC '97 controller function(s) are being disabled. After the F1 key is pressed, the boot process can then continue.
4. If the resulting number is the same as the BIOS CNR version number then the BIOS can continue with the boot process.

   Note: If compatibility between the AC '97 codecs on the CNR and the controller on the motherboard cannot be established, the BIOS must not allow AC '97 codecs to be recognized, as this will cause driver installation and compatibility issues.

6.2.5 Determining Interfaces Required

There are several different interfaces that could be used by a CNR board. To avoid confusion, it is the responsibility of the BIOS to make sure that the interfaces needed to support the feature set on the CNR board are supplied to the CNR connector by the motherboard. Any discrepancies between CNR interface requirements and motherboard interfaces routed to the CNR connector may be communicated to the user through messages on the display.

The following steps describe how to determine the interfaces required by the CNR board components.

1. Read byte address location 08h and 09h.
2. Concatenate the contents of these two registers together (with the contents of address 09h as the most significant byte).
3. Bits 4 through 0 identify which features are supported on the CNR board, and thus the required interfaces.
4. The BIOS must then verify that if a bit is set (bits 4 through 0), that the corresponding interface is routed to the CNR connector. If the interface is not routed to the CNR connector, an appropriate message may be displayed.

After verifying that each of the CNR interfaces is routed, the BIOS must then identify the pointer for each supported feature. This is described in following sections.

6.2.6 AC '97 Audio Codec Detection

After completing the previous steps, the BIOS needs to determine what AC '97 audio codecs are attached to the AC '97 Link. This must take into account AC '97 codecs on both the motherboard and on the CNR. The flowcharts shown , along with the following sections provide a high level overview of the process. The exact process and configuration of the AC '97 controller needs to be determined by the BIOS authors, from the documentation for the specific AC '97 controller.

6.2.6.1 AC '97 Codec Presence and Function Detection

Referring to Figure 28, Figure 29, and Table 40, the following steps describe how to detect and determine the type of codecs attached to the AC '97 interface.

First, determine the AC '97 compliance of the codecs on the CNR and/or Motherboard using either the CNR PnP EEPROM or the hard-coded value for the motherboard codec. The delay to wait for codecs to set the "codec ready" is dependant on the Fast Codec Start Bit in the CNR PnP EEPROM and the AC '97 version.

1. Check CDC_DN_ENABLE# signal is asserted (low).
2. If CDC_DN_ENABLE# is asserted (low) then

   2a. Check BIOS to get AC '97 Compliance and Fast Codec Information for motherboard codec.

2b. If the motherboard codec supports an 800us (Fast Codec Start) delay and is AC'97 v2.1 or v2.2 compliant then check the Fast Codec Start bit in the CNR PnP EEPROM to determine the delay. Jump to step 2d.

2c. If the motherboard codec does not support an 800us (Fast Codec Start) delay and is AC'97 v2.1 or v2.2 compliant then use the standard delay. Jump to step 2h.

2d. Check the CNR PnP EEPROM to get the Fast Codec Start bit for the CNR codec(s) by reading a word from AP+008h (Bit 14).

2e. Check the CNR PnP EEPROM to get AC '97 Compliance for CNR codec(s) by reading register 06h-07h.

2f. If the CNR codec(s) support an 800us (Fast Codec Start) delay and are AC'97 v2.1 or v2.2 compliant then use the 800uS delay and jump to step 2h.

2g. If the CNR codec does not support an 800us (Fast Codec Start) delay and is AC'97 v2.1 or v2.2 compliant then use the standard delay.

2h. For AC'97 versions later than v2.2, Fast Codec Start may be assumed.

3. If CDC_DN_ENAB# is not asserted (high) then

3a. Read the CNR PnP EEPROM to get Fast Codec Start bit for CNR codec to determine the delay (as described in steps 2d through 2h).

3b. For AC'97 versions later than v2.2, Fast Codec Start may be assumed.

6.2.6.1.1 "Fast Codec Start" Codec Detection Algorithm

Referring to Figure 28, the following steps describe a method to detect and determine the type of the AC '97 codec(s) on the interface.

1. Enable the audio and modem functions of the AC '97 digital controller.
2. De-assert the AC '97 reset signal (AC_RESET#). In an effort to reduce the overall boot time of the system, the BIOS developer may consider executing other sections of the BIOS boot code while waiting for the AC '97 codecs to complete their power-up sequence and set the "codec ready" bits.
3. Wait for all codecs on the AC '97 Interface to become ready by monitoring the state of the "codec ready" bit.

   3a. The BIOS needs to time out after **800 uS**, if the "codec ready" bit is not set in order to prevent a potential lock-up condition with BIOS continuously waiting for a codec that does not exist.

   3b. If the BIOS times out while waiting for a "codec ready" bit to be set on the primary codec, then BIOS should not check for a secondary codec or tertiary codec.

   3c. If the BIOS times out while waiting for a "codec ready" bit to be set on the secondary codec, then the BIOS can assume that there is no secondary codec in the system.

   3d. If the BIOS times out while waiting for a "codec ready" bit to be set on the tertiary codec, then BIOS can assume that a primary codec is present and possibly a secondary codec (if a "codec ready" is set for the secondary codec).
4. Identify the codec function(s) of the primary codec (codec address 00b).

   4a. Read registers 02h and 3Ch of the primary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the primary codec on the AC '97 Interface.
5. Identify the codec function(s) of the secondary codec (codec address 01b).

   5a. Read registers 02h and 3Ch of the secondary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the secondary codec on the AC '97 Interface.
6. Identify the codec function(s) of the tertiary codec (codec address 10b or 11b).

6a. Read registers 02h and 3Ch of the tertiary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the tertiary codec on the AC ’97 Interface.

7. If a primary codec was not found on the AC ’97 interface, then the AC ’97 controller must be disabled to ensure that the operating system does not attempt to load a driver.
8. If a primary codec was found, then the BIOS AC ’97 codec detection routines can continue. (Section 6.2.6.2)

6.2.6.1.2 Standard Codec Detection Algorithm

Referring to Figure 29, the following steps describe a method to detect and determine the type of the AC ’97 R2.1 codec(s) on the interface.

1. Enable the audio and modem functions of the AC ’97 digital controller.
2. De-assert the AC ’97 reset signal (AC_RESET#). In an effort to reduce the overall boot time of the system, the BIOS developer may consider executing other sections of the BIOS boot code while waiting for the AC ’97 codecs to complete their power-up sequence and set the “codec ready” bits.
3. Wait for all codecs on the AC ’97 Interface to become ready by monitoring the state of the “codec ready” bit.

   3a. The BIOS needs to time out after **600 mS**, if the “codec ready” bit is not set in order to prevent a potential lock-up condition with BIOS continuously waiting for a codec that does not exist.

   3b. If the BIOS times out while waiting for a “codec ready” bit to be set on the primary codec, then BIOS should not check for a secondary codec or tertiary codec.

   3c. If the BIOS times out while waiting for a “codec ready” bit to be set on the secondary codec, then the BIOS can assume that there is no secondary codec in the system.

   3d. If the BIOS times out while waiting for a “codec ready” bit to be set on the tertiary codec, then BIOS can assume that a primary codec is present and possibly a secondary codec (if a “codec ready” is set for the secondary codec).
4. Identify the codec function(s) of the primary codec (codec address 00b).

   4a. Read registers 02h and 3Ch of the primary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the primary codec on the AC ‘97 Interface.
5. Identify the codec function(s) of the secondary codec (codec address 01b).

   5a. Read registers 02h and 3Ch of the secondary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the secondary codec on the AC ’97 Interface.
6. Identify the codec function(s) of the tertiary codec (codec address 10b or 11b).

   6a. Read registers 02h and 3Ch of the tertiary codec. By using the values returned from registers 02h and 3Ch, and referring to Table 40, determine the functionality of the tertiary codec on the AC ’97 Interface.
7. If a primary codec was not found on the AC ’97 interface, then the AC ’97 controller must be disabled to ensure that the operating system does not attempt to load a driver.
8. If a primary codec was found, then the BIOS AC ’97 codec detection routines can continue. (Section 6.2.6.2)

		AC '97 Codec Registers	
		Register 02h	**Register 3Ch**
Codec Function	**Audio Codec (AC)**	Non-zero value	Zero Value
	Modem Codec (MC)	Zero value	Non-zero value
	Audio/Modem Codec (AMC)	Non-zero value	Non-zero value

Table 40 – AC '97 Codec Function Mapping to Codec Register Value

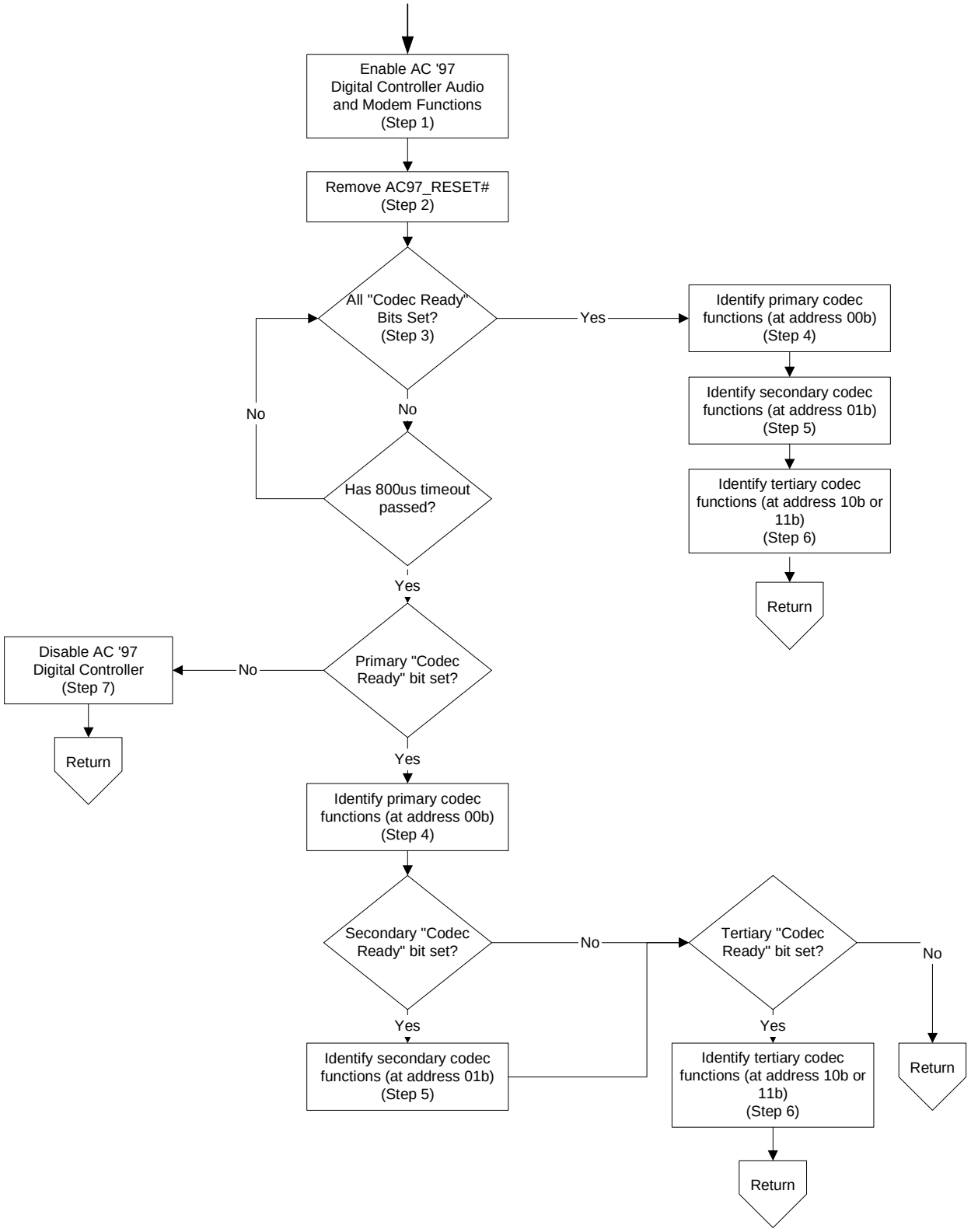


Figure 28 – AC '97 Presence and Function Detection Flowchart for Fast Codec Start Detection

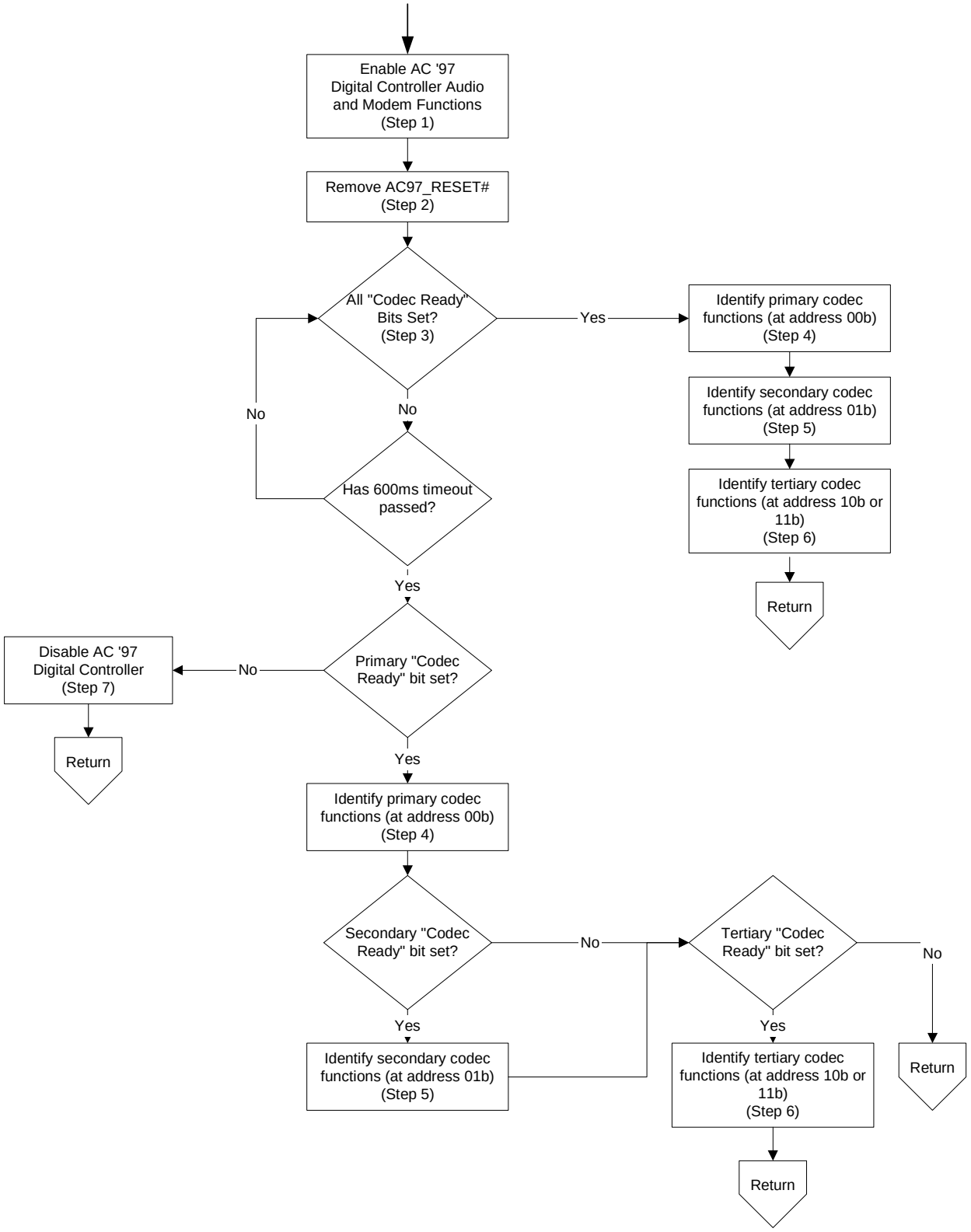


Figure 29 – AC '97 Codec Presence and Function Detection Flowchart for Standard Codec Detection

6.2.6.2 AC ’97 Function Enabling and Disabling

After identifying the presence and functionality of the codec(s) attached to the AC ’97 interface, the BIOS must then determine if the codec functions attached are legal combinations. The text below and the flow chart in Figure 30 describe the steps that the BIOS must perform to properly enable and/or disable in the AC ’97 controller.

1. Found a primary Audio Codec (AC) only:
   1a. Disable the modem function of the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).
   1b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).
   1c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   1d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
2. Found a primary modem codec (MC) only:
   2a. Disable the audio function of the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).
   2b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).
   2c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   2d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
3. Found a primary audio/modem codec (AMC) only:
   3a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).
   3b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   3c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
4. Found a primary audio codec (AC) and a secondary audio codec (AC):
   4a. Disable the modem function of the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).
   4b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).
   4c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   4d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
5. Found a primary audio/modem codec (AMC) and a secondary audio codec (AC):
   5a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)
   5b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   5c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
6. Found a primary audio codec (AC) and a secondary modem codec (MC) only:
   6a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).
   6b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self test).
   6c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
7. Found a primary Audio Codec (AC) and a secondary audio/modem codec (AMC):
   7a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)
   7b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).
   7c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).
8. Found a primary audio codec (AC), a secondary audio codec (AC) and a tertiary audio codec (AC):
   8a. Disable the modem function of the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).
   8b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

8c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

8d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

9. Found a primary audio codec (AC), a secondary audio codec (AC) and a tertiary modem codec (MC):

   9a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

   9b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

   9c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

10. Found a primary audio codec (AC), a secondary audio codec (AC) and a tertiary audio/modem codec (AMC):

    10a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

    10b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    10c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

11. Found a primary audio codec (AC), a secondary modem codec (MC) and a tertiary audio codec(AC):

    11a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

    11b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    11c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

12. Found a primary audio codec (AC), a secondary audio/modem codec (AMC) and a tertiary audio codec(AC):

    12a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

    12b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    12c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

13. Found a primary audio/modem codec (AMC), a secondary audio codec (AC) and a tertiary audio codec(AC):

    13a. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

    13b. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    13c. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

14. Found a primary modem codec (MC) and a secondary audio/modem codec (AMC):

    14a. Disable the modem function of the AC '97 digital controller (such that it is not visible to the Operating System's PCI Enumerator).

    14b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).

    14c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    14d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

    14e. BIOS ***must*** place the message "**Illegal AC '97 configuration. AC '97 Modem function disabled. Press F1 to Resume.**" on the display. After the F1 key is pressed, the boot process can then continue.

15. Found a primary audio/modem codec (AMC) and a secondary modem codec (MC):

    15a. Disable the modem function of the AC '97 digital controller (such that it is not visible to the Operating System's PCI Enumerator).

    15b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3).

    15c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

    15d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

    15e. BIOS ***must*** place the message "**Illegal AC '97 configuration. AC '97 Modem function disabled. Press F1 to Resume.**" on the display. After the F1 key is pressed, the boot process can then continue.

16. Found a primary audio/modem codec (AMC) and a secondary audio/modem codec (AMC):

16a. Disable the modem function of the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).

16b. Execute the Plug-and-Play BIOS code (Section 6.2.6.3)

16c. Assert AC97_RESET# (to allow PC_BEEP pass through during power on self-test).

16d. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

16e. BIOS **must** place the message “**Illegal AC ’97 configuration. AC ’97 Modem function disabled. Press F1 to Resume.**” on the display. After the F1 key is pressed, the boot process can then continue.

17. No codecs are present (Primary, Secondary, or Tertiary):

17a. Disable the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).

17b. Return to executing CNR Plug-and-Play BIOS code (Section 6.2.7).

18. Invalid Combination Detected. (Any combination not in this list is considered invalid)

18a. Disable the AC ’97 digital controller (such that it is not visible to the Operating System’s PCI Enumerator).

18b. BIOS ***must*** place the message “**Illegal AC ’97 configuration. AC ’97 Audio and Modem functions disabled. Press F1 to Resume.**” on the display. After the F1 key is pressed, the boot process can then continue.

The CNR Specification provides several different codec combinations that may or may not be considered legal combinations. This specification provides a set of rules that insure that the highest level of interoperability can be achieved. Table 41 provides a list of potential combinations that can be supported on CNR. Note that these combinations may, or may not be supported depending on the capabilities of the AC ’97 Controller (e.g. support for two Codecs versus three codecs). In addition, Table 42 provides a list of the unsupported codec combinations. Any attempt to support the codec combinations shown in Table 42 will result in unreliable operation and must be identified by the BIOS detection algorithms and result in a disabled AC ’97 Controller.

	Codec A (00b)	Codec B (01b)	Codec C (10b)
Option #1	Audio	Audio	Audio
Option #2	Audio	Audio	Modem
Option #3	Audio	Audio	Audio/Modem
Option #4	Audio	Modem	Audio
Option #5	Audio	Audio/Modem	Audio
Option #6	Audio/Modem	Audio	Audio

Table 41 – Supported Configurations (with CNR installed)

	Codec A (00b)	Codec B (01b)	Codec C (10b)
Option #1	Audio	Modem	Modem
Option #2	Audio	Modem	Audio/Modem
Option #3	Audio	Audio/Modem	Modem
Option #4	Audio	Audio/Modem	Audio/Modem
Option #5	Modem	Audio	Audio
Option #6	Modem	Audio	Modem
Option #7	Modem	Audio	Audio/Modem
Option #8	Modem	Modem	Audio
Option #9	Modem	Modem	Modem
Option #10	Modem	Modem	Audio/Modem
Option #11	Modem	Audio/Modem	Audio
Option #12	Modem	Audio/Modem	Modem
Option #13	Modem	Audio/Modem	Audio/Modem
Option #14	Audio/Modem	Audio	Modem
Option #15	Audio/Modem	Audio	Audio/Modem
Option #16	Audio/Modem	Modem	Audio
Option #17	Audio/Modem	Modem	Modem
Option #18	Audio/Modem	Modem	Audio/Modem
Option #19	Audio/Modem	Audio/Modem	Audio
Option #20	Audio/Modem	Audio/Modem	Modem
Option #21	Audio/Modem	Audio/Modem	Audio/Modem

Table 42 – Unsupported Configurations (with CNR installed)

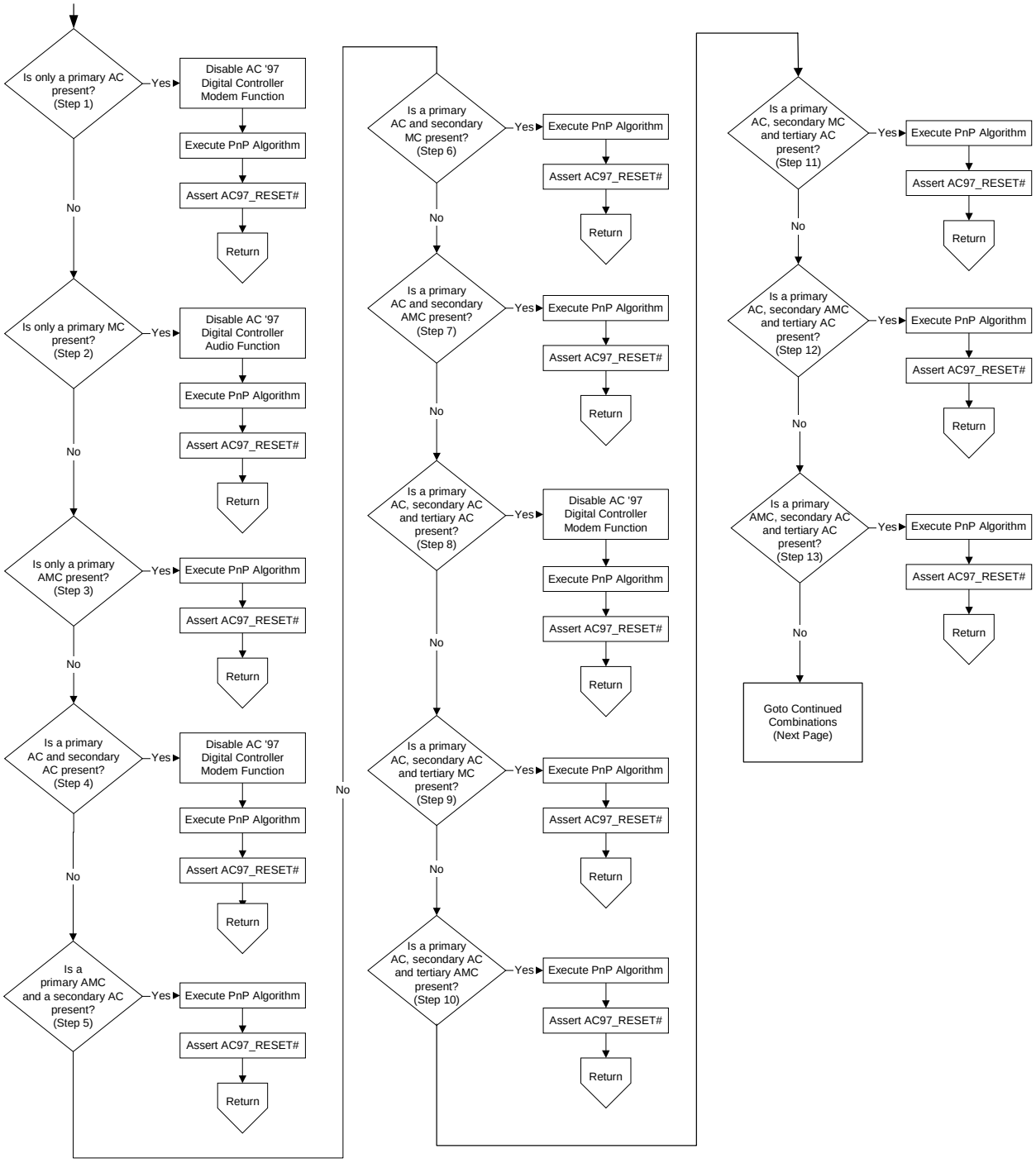


Figure 30 – AC '97 Enabling and Disabling Flowchart

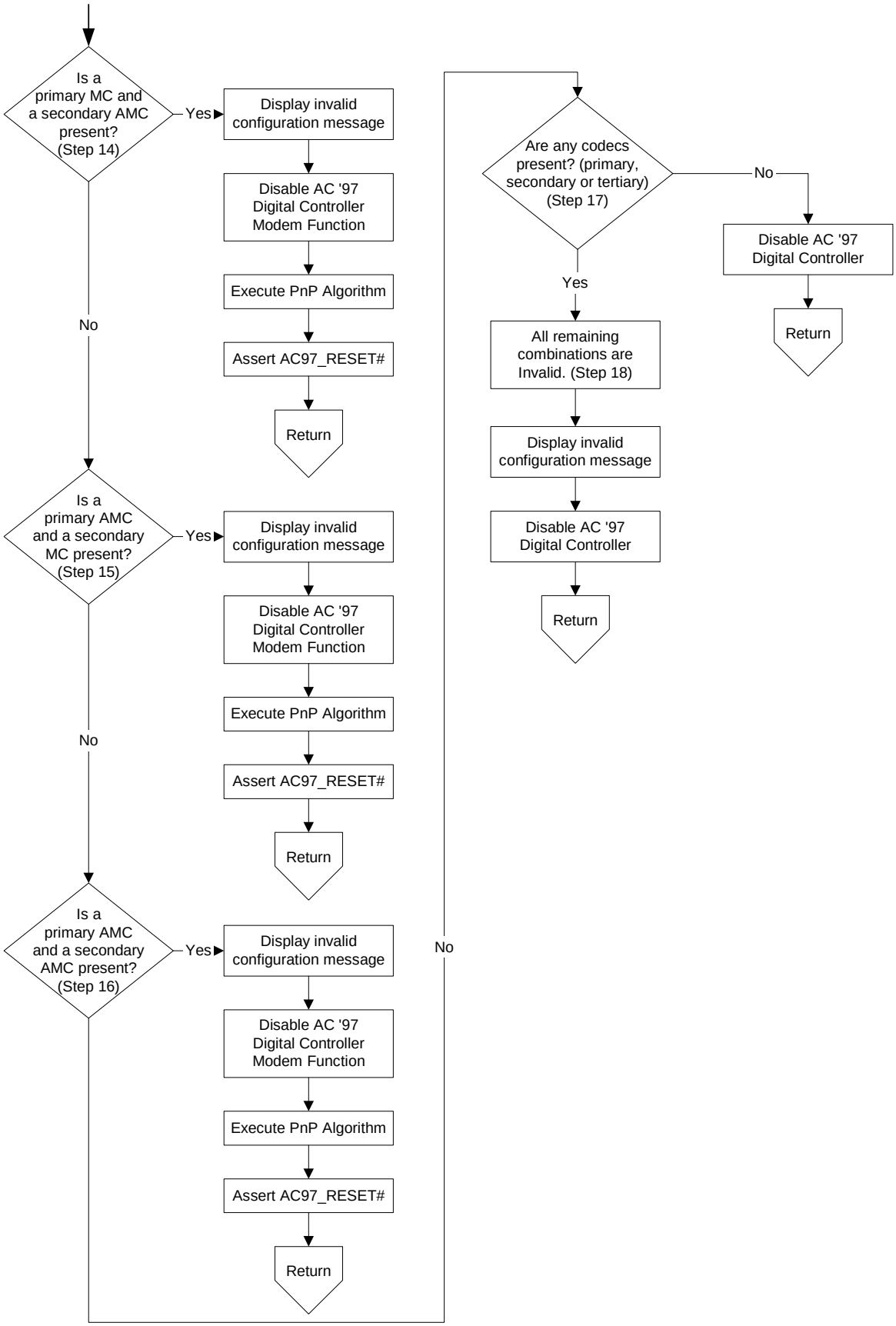


Figure 31 – AC '97 Enabling and Disabling Flowchart continued...

6.2.6.3 AC ’97 Plug-and-Play Information Extraction

Extraction and programming of the AC ‘97 Plug-and-Play information from the CNR PnP EEPROM is very straightforward. The BIOS simply determines where to retrieve the PnP information, based on knowledge of the AC ’97 codec location (on the motherboard or on the CNR) and then copies the information to the appropriate locations within the AC '97 controller PCI configuration space. The flowchart of Figure 32 and the text following it describe the steps to extract the AC ’97 PnP information from the appropriate location.

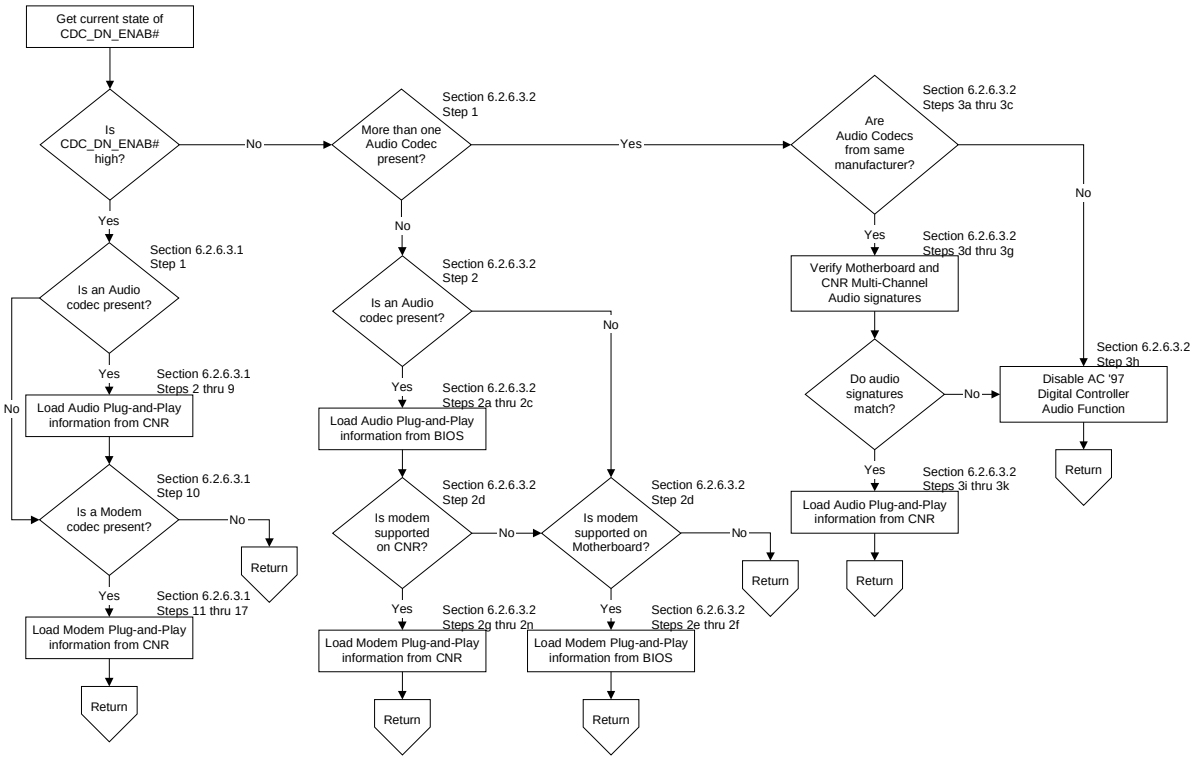


Figure 32 – AC ’97 Plug-and-Play Information Flowchart

To start the process of extracting PnP information, the BIOS **must** first read the state of the CDC_DN_ENAB# signal. If the state of the CDC_DN_ENAB# signal is high, then the CNR is controlling the AC ’97 interface, and thus providing all of the AC ’97 functionality and all AC ’97 PnP information. If the state of the CDC_DN_ENAB# signal is low, then the motherboard is mastering the AC ’97 interface, and thus is providing at least the primary codec, and possibly some of the AC ’97 PnP information. Once the state of the CDC_DN_ENAB# signal is known, then the following steps can be executed.

6.2.6.3.1 BIOS Steps for Audio Function PnP with CDC_DN_ENAB# High

1. The BIOS begins by determining if an audio codec (AC) is present. This information was originally found during “AC ’97 Codec Presence and Function Detection” (Section 6.2.6.1).
2. If an audio codec is present, the BIOS reads byte addresses 0Eh and 0Fh, from the PnP EEPROM. These two values are then concatenated (with the contents of address 0Fh as the most significant byte). The resulting concatenation becomes the audio pointer.
3. The BIOS then continues to read the successive pointer locations until it encounters a non-zero pointer. When this non-zero pointer is encountered (or the last valid address is reached, word 2Eh), the BIOS subtracts the AUDIO POINTER from the non-zero pointer. The resulting answer is the number of bytes to be read from the PnP EEPROM for the audio function.

   Note: For this version of the CNR specification, this number of bytes must be exactly 20 (ten 16-bit words).

4. The BIOS then reads the first two bytes of audio function PnP data from the address pointed to by the audio pointer (from Step 2 of this section). These two bytes are concatenated (with the second byte read being the most significant byte).
5. The resulting concatenation is then written to the subsystem vendor ID (SVID) register of the PCI configuration space for the AC '97 audio controller.
6. The BIOS then sequentially reads the next two bytes of audio function PnP data from the CNR EEPROM. These two bytes are concatenated (with the second byte read being the most significant byte).
7. The resulting concatenation is then written to the subsystem ID (SID) register of the PCI configuration space for the AC '97 audio controller.
8. The BIOS may then either read or ignore the remaining 16 bytes of information. This remaining information is only used for multi-channel audio PnP and CNR audio physical descriptions.
9. At this point, the BIOS has read the correct number of bytes (eight) for the audio function PnP.
10. Next, the BIOS determines if a modem codec (MC) is present. This information was originally found during "AC '97 Codec Presence and Function Detection" (Section 6.2.6.1).
11. If a modem codec is present, the BIOS reads byte addresses 10h and 11h from the PnP EEPROM. These two values are then concatenated (with the contents of address 11h as the most significant byte). The resulting concatenation becomes the modem pointer.
12. The BIOS then continues to read the successive pointer locations until it encounters a non-zero pointer. When this non-zero pointer is encountered (or the last valid address is reached, word 2Eh), the BIOS subtracts the modem pointer from the non-zero pointer. The resulting answer is the number of bytes to be read from the PnP EEPROM for the modem function.

    Note: For this version of the CNR specification, this number of bytes must <u>not</u> be less than four (two 16-bit words).

13. The BIOS then reads the first two bytes of modem function PnP data from the address pointed to by the modem pointer (from Step 11 of this section). These two bytes are concatenated (with the second byte read being the most significant byte).
14. The resulting concatenation is then written to the subsystem vendor ID (SVID) register of the PCI configuration space for the AC '97 modem controller.
15. The BIOS then sequentially reads the next two bytes of modem function PnP data from the CNR EEPROM. These two bytes are concatenated (with the second byte read being the most significant byte).
16. The resulting concatenation is then written to the subsystem ID (SID) register of the PCI configuration space for the AC '97 modem controller.
17. At this point, the BIOS has read the correct number of bytes (four) and the BIOS code can return to the calling BIOS code (Section 6.2.6.2)

6.2.6.3.2 BIOS Steps for Audio Function PnP With CDC_DN_ENAB# Low

1. The BIOS begins by determining if more than one audio codec (AC) is present. This information was originally found during "AC '97 Codec Presence and Function Detection" (Section 6.2.6.1).
2. If a single audio or modem codec is found then the BIOS executes the following steps:

    2a. Was a primary audio codec found in Section. If a primary audio codec was not found, then jump to Step 2d) of this section.

    2b. If a primary audio codec was found, the BIOS writes the subsystem vendor ID (SVID) register of the PCI configuration space for the AC '97 audio controller with a value hard coded into the BIOS (that is, the PCI SIG assigned vendor ID for the manufacturer of the motherboard).

    2c. Next, the BIOS writes the subsystem ID (SID) register of the PCI configuration space for the AC '97 audio controller with a value hard coded into the BIOS (that is, a 16-bit unique value identifying the model number of the motherboard).

2d. The BIOS then determines if a modem codec is located on the motherboard or CNR. This is performed checking the status of MDM bit (bit 1) of the function ID Register (08h) of the PnP EEPROM (previously read in Section 6.2.5). If the MDM bit is set, then jump to Step 2g of this section.

2e. If the MDM bit is cleared, then the BIOS writes the subsystem vendor ID (SVID) register of the PCI configuration space for the AC ’97 modem controller with a value hard coded into the BIOS (that is, the PCI SIG assigned vendor ID for the manufacturer of the motherboard).

2f. Next the BIOS writes the subsystem ID (SID) register of the PCI configuration space for the AC ’97 modem controller with a value hard coded into the BIOS (that is, a 16-bit unique value identifying the model number of the motherboard).

2g. If the MDM bit is set, the BIOS reads byte addresses 10h and 11h, from the PnP EEPROM. These two values are then concatenated (with the contents of address 11h as the most significant byte). The resulting concatenation becomes the modem pointer.

2h. The BIOS then continues to read the successive pointer locations until it encounters a non-zero pointer. When this non-zero pointer is encountered (or the last valid address is reached, word 2Eh), the BIOS subtracts the modem pointer from the non-zero pointer. The resulting answer is the number of bytes to be read from the PnP EEPROM for the modem function.

Note: For this version of the CNR specification, this number of bytes must <u>not</u> be less than 20 (ten 16-bit words).

2i. The BIOS then reads the first two bytes of modem function PnP data from the address pointed to by the modem pointer (from Step 11 of Section 6.2.6.3.1). These two bytes are concatenated (with the second byte read being the most significant byte).

2j. The resulting concatenation is then written to the subsystem vendor ID (SVID) register of the PCI configuration space for the AC '97 modem controller.

2k. The BIOS then sequentially reads the next two bytes of modem function PnP data from the CNR EEPROM. These two bytes are concatenated (with the second byte read being the most significant byte).

2m. The resulting concatenation is then written to the subsystem ID (SID) register of the PCI configuration space for the AC '97 modem controller.

2n. At this point BIOS has read the correct number of bytes (four) and the BIOS code can return to the calling BIOS code (Section 6.2.6.2)

3. If more than one audio codec is found, the BIOS executes the following steps:

3a. BIOS reads the VID1 and VID2 registers from the primary (motherboard) audio codec. The two 16-bit registers are then concatenated (with VID1 as the most significant word). BIOS follows this with a masking off (clearing) of the least significant eight-bits of the previous concatenation.

3b. Next, the BIOS reads the VID1 and VID2 registers from the secondary (CNR) audio codec. The two 16-bit registers are then concatenated (with VID1 as the most significant word). BIOS follows this with a masking off (clearing) of the least significant eight-bits of the previous concatenation.

3c. Finally BIOS compares the resulting two 32-bit words. If the two words are identical, the BIOS skips to Step 3d. If the two words are not identical, the BIOS skips to Step 3h.

3d. The BIOS next retrieves the motherboard multi-channel signature. If a multi-channel audio signature is not available (or is 0000h or FFFFh), then the BIOS **<u>must</u>** skip to Step 3h.

3e. Next, the BIOS reads byte addresses 0Eh and 0Fh, from the PnP EEPROM. These two values are then concatenated (with the contents of address 0Fh as the most significant byte). The resulting concatenation becomes the audio pointer (AP).

3f. The BIOS then reads the two bytes located at the addresses AP+004h and AP+005h, then concatenates the resulting reads together (with the value from AP+005h as the most significant byte). The resulting concatenation is the CNR audio multi-channel signature.

3g. The BIOS compares the CNR audio multi-channel signature (from Step 3f) to the motherboard multi-channel signature (from Step 3d). If the two values are identical, then skip to Step 3i.

3h. If the two values are not identical, then the BIOS **must** disable the audio function of the AC '97 digital controller (such that it is not visible to the Operating System's PCI Enumerator) and place the message "**CNR Multi-Channel Audio Upgrade not compatible with Motherboard. Audio Function has been disabled. Press F1 to Resume.**" on the display. After the F1 key is pressed, the boot process can then continue.

3i. If the two values are identical, the BIOS then the reads the two bytes located at the addresses AP+000h and AP+001h. BIOS concatenates the resulting reads together (with the value from AP+001h as the most significant byte). The resulting concatenation is then written to the subsystem vendor ID (SVID) register of the PCI configuration space for the AC '97 audio controller.

3j. The BIOS then the reads the two bytes located at the addresses AP+006h and AP+007h. The BIOS concatenates the resulting reads together (with the value from AP+007h as the most significant byte). The resulting concatenation is then written to the subsystem ID (SID) register of the PCI configuration space for the AC '97 audio controller.

3k. At this point, the BIOS has completed the Plug-and-Play for multi-channel audio and can return to the calling BIOS code (Section 6.2.6.2)

6.2.7 USB Compliance and Options

The USB interface, by definition, contains all of the necessary protocol to retrieve the Plug-and-Play information for any USB 2.0 function attached to the USB signals on the CNR board. However, there may be instances where a USB device, on a CNR board, requires motherboard features that have not been implemented on the motherboard for the CNR connector. Some of these features include power supply current (for wake capabilities) when a USB 2.0 compliant function is installed on the CNR.

In order to determine these type of feature differences, the BIOS **must** retrieve data from the USB option register, as follows.

1. Read, from the PnP EEPROM, byte addresses 12h and 13h. Concatenate the resulting two reads (with the contents of address 13h as the most significant byte). The resulting concatenation becomes the USB pointer.

2. The BIOS then continues to read the successive pointer locations until it encounters a non-zero pointer. When this non-zero pointer is encountered (or the last valid address register is reached, at word 2Eh), the BIOS subtracts the USB pointer from the non-zero pointer. The resulting answer is the number of bytes to be read from the PnP EEPROM for the USB function.

   Note: For this version of the CNR specification, this number of bytes must be six (three 16-bit word).

3. BIOS then reads the first two bytes of USB data from the address pointed to by the USB pointer (from Step 1 of this section). These two bytes are concatenated (with the second byte read being the most significant byte), producing the USB Option Register Contents. The value extracted for the USB Option Register should be temporarily stored until Step 4 is completed.

4. BIOS then reads the next two bytes of the USB data (addresses UP+002h and UP+003h). These two bytes are concatenated together (with the second byte read being the most significant byte). The resulting concatenation indicates the minimum level of BIOS compliance required to be able to support the USB devices or functions on the CNR. If BIOS is not able to support the devices or functions on the CNR, the message "**BIOS not compliant with USB devices on CNR.**" must be displayed.

5. BIOS now uses the value previously extracted for the USB Option Register and with the exception of the SPD[1:0] bits, is decoded and used as described in Section 6.1.5.1.

6. The BIOS may use the SPD[1:0] bits to determine if the motherboard supports USB version 2.0 as required by the CNR. If the USB version required by the CNR is greater than supported by the motherboard, the BIOS may display the message "**USB version required by the CNR is not supported by motherboard. The CNR USB function will operate at a lower speed. Press F1 to Resume.**" If the USB version required by the CNR is lower than supported by the motherboard, then the BIOS may display a message indicating the discrepancy. After the F1 key is pressed, the boot process can then continue.

7. The BIOS may then either read or ignore the remaining 2 bytes of information. This remaining information is only used for USB port physical descriptions.

8. At this point BIOS has read the correct number of bytes and can continue with the execution of the CNR BIOS code.

6.2.8 LAN Options

Extraction of the LAN Plug-and-Play information from the CNR PnP EEPROM is very straightforward. The BIOS first verifies that the motherboard supports the type of LAN interface required by the CNR board, then verifies that it is compliant with the LAN Interface version, and finally copies the information from the CNR PnP EEPROM to the appropriate locations within PCI configuration space for the LAN function, as described in the following steps.

1. Read, from the PnP EEPROM, byte addresses 16h and 17h. Concatenate the resulting two reads together (with the contents of address 17h as the most significant byte). The resulting concatenation becomes the LAN pointer.

2. The BIOS then continues to read the successive pointer locations until it encounters a non-zero pointer. When this non-zero pointer is encountered (or the last valid address is reached, word 2Eh), the BIOS subtracts the LAN pointer from the non-zero pointer. The resulting answer is the number of bytes to be read from the PnP EEPROM for the LAN function.

   Note: For this version of the CNR specification, this number of bytes must be eight (four 16-bit words).

3. BIOS then reads the first two bytes if the LAN function PnP data from the address pointed to by the LAN pointer (from step 1 of this section). These two bytes are concatenated (with the second byte being the most significant byte).

4. The least significant bit (INTF bit) of the resulting concatenation is then used to determine if the LAN interface required by the CNR board is the same as is provided by the motherboard (refer to Section 6.1.7.1 for information regarding the decoding of the INTF bit). If the LAN interfaces do not match, the BIOS must disable the MAC connected to the motherboard LAN interface (such that it is not visible to the Operating System's PCI Enumerator). The BIOS **must** also display the message "**CNR LAN Interface not compatible with Motherboard LAN Interface. LAN Function has been disabled. Press F1 to Resume.**" indicating the mismatch to the user. After the F1 key is pressed, the boot process can then continue.

5. BIOS then reads the next two bytes of LAN function PnP data. These two bytes are concatenated (with the second byte read being the most significant byte). The resulting concatenation is then stored for future use.

6. BIOS then sequentially reads the next two bytes of LAN function PnP data from the CNR EEPROM. These two bytes are concatenated (with the second byte read being the most significant byte). The resulting concatenation is then stored for future use.

7. BIOS then sequentially reads the next two bytes of LAN function PnP data from the CNR EEPROM. These two bytes are concatenated (with the second byte read being the most significant byte). BIOS uses this value to determine if it is compliant with the interface version required by the LAN devices on the CNR. If BIOS is not compliant the message **"BIOS not compliant with CNR LAN devices. LAN function disabled.**"

8. BIOS then takes the LAN function PnP data extracted in Steps 5 and 6 and writes it to the subsystem vendor ID (SVID) and the subsystem ID (SID) register of the PCI configuration space for the LAN controller (if needed).

9. At this point, BIOS has read the correct number of bytes (eight), and the BIOS code can continue with the balance of the CNR BIOS code.

6.3 Software and Driver Requirements

The following sections describe some of the issues that must be considered by both the designer and the user of a CNR board. Specifically described are what is involved in supporting call progress, speakerphone, and telephone answering machine (TAM) functionality now and in the future. In addition, there are considerations when using CNR as a medium for increasing the number of audio channels provided/supported in a given system configuration.

6.3.1 Audio/Modem Interfacing for Call Progress

Unlike previous versions of riser specifications (such as the *Audio/Modem Riser Specification*), the CNR specification does not support the analog mono-audio connections between an audio codec on the motherboard and a modem codec on the CNR board. This implies that support for features such as call progress monitoring, speakerphone, and TAM cannot be supported through a hardware means (unless both the motherboard and the CNR board provide headers for the analog mono-audio signals). Instead, the vendors of the audio and modem drivers are highly encouraged to develop (or take advantage of) standard software interfaces between the audio and modem subsystems, to allow the transfer of audio data for the purposes of supporting call progress monitoring, speakerphone, and TAM. This standard interface, if not currently available, may need to be developed, or may become available in future operating system releases.

Until a standard interface is developed, both the motherboard and CNR board designers are encouraged to implement standard telephony headers to allow cabling of analog mono audio modem signals between the modem subsystem on the CNR and the audio subsystem on the motherboard.

6.3.2 Multi-Channel Audio Upgrade Considerations

One possible use of CNR is to allow the system integrator to increase the number of supported audio channels from the standard two-channel audio normally found on today's motherboards to either four- or six-channel audio, by simply installing a CNR board with the appropriate audio codecs. The following sections describe the Plug-and-Play aspects of handling multi-channel audio upgrades using CNR, as well as the codec and driver requirements of multi-channel audio upgrades using CNR.

6.3.2.1 Multi-Channel Audio Plug-and-Play Signatures

The Plug-and-Play aspects of multi-channel audio upgrades provides a unique challenge, since the same CNR can be used as either a single two-channel audio subsystem, or as an upgrade to an existing two-channel subsystem (on the motherboard). To insure that the correct driver is loaded, regardless of how the CNR is used, requires additional support in assigning the correct subsystem ID (SID) to the entire audio solution. For example, the CNR used as a complete two-channel audio subsystem would most likely use a different driver than the same card used as a multi-channel audio upgrade, thus the need for different SID values depending on how the card is used.

In addition to the driver differences, there are also specific hardware requirements to ensure that the end-user receives a “high quality” experience with their audio subsystem. These hardware requirements include gain matching between audio channels (for example, front left to front right, front right to surround right, and so on), the number of inversions in the audio stream from the digital data stream to the output jack must be identical, and net phase shift between audio channels.

Given these requirements, and the ultimate goal of providing a “high quality” experience to the end-user, a method of matching motherboard and CNR audio subsystems has been created for the CNR Specification. This matching method simply uses a “signature” for the motherboard, and a “signature” for the CNR. Each subsystem stores its own signature (the motherboard signature is stored in the BIOS; the CNR signature is stored in the SMBus PnP EEPROM). The BIOS then verifies that the signatures and codec manufacturers match, and then enables the audio functions, and lastly programs the appropriate SID value (from the multi-channel model ID register of the SMBus PnP EEPROM).

The assignment of these signatures controlled only by the AC ’97 codec manufacturer. The process for receiving a signature value for a given motherboard or CNR design is very straightforward, and is described in the following sections.

6.3.2.1.1 Motherboard Multi-Channel Audio Signature Assignment

In order for a motherboard manufacturer to receive the signature for a given motherboard audio design, the codec vendor **must** review the audio design, and verify that it meets the requirements to be mated with a given CNR, for the purpose of multi-channel audio upgrades. This verification by the codec vendor can be as simple as a schematic review, or as complex as a full hardware qualification test. The decision of what level of review or testing is needed resides with the codec vendor and motherboard manufacturer.

Once the codec vendor has approved the motherboard audio subsystem, a signature number is provided by the codec vendor for inclusion in the motherboard manufacturer's BIOS code. In addition to the signature number, the codec vendor would provide a list of CNR manufacturers and model numbers that are compatible with the motherboard audio subsystem (and thus would have the same signature number).

6.3.2.1.2 CNR Multi-Channel Audio Signature Assignment

Like the motherboard, the CNR must also receive a signature number. This number is provided by the codec manufacturer to the CNR manufacturer, assuming that the CNR manufacturer follows the guidelines set forth by the codec manufacturer. With the goal of providing a high quality experience to the end-user, the codec vendor **must** put the appropriate reviews and checks in place to ensure that a CNR only receives a signature number when it meets the codec vendor's requirements.

In addition to the signature number, the codec vendor **must** provide a SID value to the CNR manufacturer. The CNR manufacturer would then place both the signature and the SID values in the SMBus PnP EEPROM of the CNR (at the correct address locations, see Section 6.1.3).

6.3.2.2 Requirements for Multi-Channel Audio Upgrades

In this multi-channel audio upgrade scenario, the audio codec(s) on the motherboard continues to remain active, providing the same basic functionality as prior to the upgrade. Installing a CNR can then increase the number of audio channels from two channels to four or six channels (or from four channels to six channels). Although this seems like a straightforward process, consider the following:

1. When installing a CNR as an upgrade to the number of audio channels supported, the system integrator is **required** to use a CNR board with audio codecs and drivers from the same manufacturer as the audio codec on the motherboard. This reduces potential driver compatibility and installation issues.
2. The drivers installed with a CNR upgrade that increase the number of audio channels, are **required** to be the same drivers provided with the CNR board used for the upgrade. This further reduces the possibility of driver compatibility and installation issues.

   Note: The above upgrades apply only to system manufacturers, system integrators, OEM upgrades. The CNR is not an end-user upgrade device.

7 Motherboard Design Rules

The following sections provide the design rules for the various form factors, including ATX, microATX, and FlexATX. Adherence to these design rules is required in order to claim compliance with the CNR specification.

7.1 ATX Family Form Factors Design Rules

1. CNR connectors **must** be placed in the left-most slot location and share the I/O bracket space with the adjacent PCI slot (or AGP), as shown in the previous drawings. Placing the CNR connector in any other slot location violates the CNR specification.
2. A motherboard design **must not** simultaneously support both an audio/modem riser (AMR) connector and a CNR connector, either placed or as a manufacturing stuffing option. This is to prevent excessive loading on those interfaces used by both specifications.
3. A given motherboard design **must** not support more than a single CNR connector.
4. Maximum trace capacitance for any AC-link trace in any form factor **must** not exceed 25 pF. This capacitance does not include the codec input or output capacitance, the input or output capacitance associated with the controller, or the capacitance associated with the CNR connector.

5. If the controller sourcing the AC-link signals does not incorporate internal pull-down resistors on the AC97_BITCLK and AC97_SDATA_IN[1:0] signals, the motherboard **must** provide these pull-down resistors. The pull-down value must not be less than 10 kΩ.

6. Maximum trace capacitance for any LAN Interface trace in any form factor **must** not exceed 20 pF. This capacitance does not include the PLC/PHY device input or output capacitance, the input or output capacitance associated with the core logic MAC, or the capacitance associated with the CNR connector.

7. The LAN Interface **must** only be routed between two devices on the motherboard. This implies that the LAN interface must be routed from the MAC to a PLC/PHY on the motherboard, or from the MAC to the CNR connector. Routing from the MAC to both a PLC/PHY and the CNR connector is not allowed.

8 Communication and Networking Riser Design Rules

The following paragraphs list the design rules for the various form factors, including ATX, microATX, and FlexATX. Adherence to these design rules is required in order to claim compliance with the CNR specification.

1. Maximum trace capacitance in a two-codec implementation for any AC-link trace on the CNR board **must** not exceed 15 pF. This capacitance does not include the codec input or output capacitance, or the capacitance associated with the CNR connector.

2. Maximum trace capacitance in a three-codec implementation for any AC-link trace on the CNR board **must** not exceed 20 pF. This capacitance does not include the codec(s) input or output capacitance or the capacitance associated with the CNR connector.

3. Maximum trace capacitance in a four-codec implementation for any AC-link trace on the CNR board **must** not exceed 25 pF. This capacitance does not include the codec(s) input or output capacitance or the capacitance associated with the CNR connector.

4. The CNR and AC ’97 component specifications allows for either +3.3 V or +5 V digital I/O. Care **must** be taken to ensure that both sides of the CNR connector are driven from compatible voltage rails.

5. Maximum trace capacitance of any LAN Interface trace on the CNR board **must** not exceed 6 pF. This capacitance does not include the PLC/PHY device input or output capacitance or the capacitance associated with the CNR connector.

6. All CNR reference designs and CNR production boards **must** not use jumpers to implement the AC ’97 codec disabling and demotion rules. Instead the CNR board **must** use either build time stuffing options or electronic circuitry which monitors the state of the CDC_DN_ENAB# signal to disable and demote the AC ’97 codecs located on the CNR board.

7. The SMBus Plug-and-Play EEPROM device **must** be compliant with the System Management Bus Specification, Revision 1.1 (dated December 11, 1998).