PICMG® EEEP

Embedded EEPROM Base Specification

Draft 0.08_{115}

October 20, 2009

Open Modular Computing Specifications

50

Table of Contents

1 Introduction....1
1.1 Contributors....1
1.2 Name and Logo Usage....1
1.3 Intellectual Property....2
1.3.1 Necessary Claims (referring to mandatory or recommended features)....2
1.3.2 Unnecessary Claims (referring to optional features or non-normative elements)....2
1.3.3 Third Party Disclosures....3
1.4 Copyright Notice....3
1.5 Trademarks....3
1.6 Special Word Usage....3
1.7 Acronyms / Definitions....4
1.8 Applicable Documents and Standards....7
1.9 Statement of Compliance....8
2 General....9
3 Detection....10
3.1 Detecting EeeP EEPROM....10
3.1.1 High Level Check....10
3.1.2 Sample I2C Transfer....10
3.2 Detecting COM0 R1.0 EEPROM....11
3.2.1 High Level Check....11
3.2.2 Sample I2C Transfer....11
3.3 Identifying Device Type Standard/Extended Index....11
3.3.1 Sample I2C Transfer....11
4 EEPROM Header Components....12
4.1 Concepts....12
4.2 Common EeeP EEPROM Header....12
4.2.1 Don't Care Byte:....12
4.2.2 EEPROM ID....13
4.2.3 EeeP Specification Revision....13
4.2.4 Block Offset:....13
4.2.5 Device Descriptor Byte....13
4.3 Universal Device Identifier....14
4.3.1 Vendor ID....14
4.3.2 Device ID....14
4.3.3 Device Flavor....14
4.3.4 Device Unique Revision ID....14

5 EEPROM Header Examples....15
5.1 COM0 R2.0 Carrier Board EEPROM Header....15
5.1.1 EeeP Header....15
5.1.2 COM0 ID....15
5.1.3 Device Id....15
5.1.4 Carrier Board Type....15
5.1.5 COM0 Specification Revision....15
5.1.6 USB Descriptor Byte....15
5.1.7 SATA / SAS Device Descriptor Byte....17
5.1.8 LAN Descriptor Byte....18
5.1.9 Miscellaneous I/O Descriptor Byte....18
5.1.10 Miscellaneous I/O Descriptor Byte 2....19
5.1.11 Digital Display Interface Descriptor Byte....19
5.1.12 PCI Express Lane Descriptor Data Structure....20
5.2 COM0 R2.0 Module EEPROM Header....23
5.2.1 EeeP Header....23
5.2.2 COM0 ID....23
5.2.3 Device Id....23
5.2.4 Module Type....23
5.2.5 COM0 Specification Revision....23
5.3 Expansion EEPROM Header....23
5.3.1 Description....23
6 Dynamic Descriptor Blocks....24
6.1 Overview....24
6.2 Common Dynamic Block Header....24
6.2.1 Dynamic Block Ids....25
6.2.2 Dynamic Length/Offset....25
6.3 Express Card Topology....25
6.3.2 Examples:....27
6.4 System Information Descriptor Block....28
6.4.1 Description....28
6.5 Chassis Information Descriptor Block....29
6.5.1 Description....29
6.6 Module Information Descriptor Block....30
6.6.1 Description....30
6.7 LFP Device Descriptor Block....31
6.7.1 Description....31
6.7.2 Display Interface....31
6.7.3 Raw Data....31

6.8 Vendor Specific Block....32
6.8.1 Description....32
6.8.2 Vendor ID....32
6.9 CRC16 Block....32
6.9.1 Description....32
6.10 Expansion EEPROM descriptor....35
6.10.1 Description....35
7 Sample EEPROM Content....36
7.1 Sample Carrier Board EEPROM Content....36
7.1.1 COM0R20 Type 2 Carrier Board With Expansion EEPROM....36
7.1.2 COM0R20 Type 2 Carrier Board....38
7.2 Sample COM0R20 Module EEPROM Content....41
7.2.1 Sample 1 All in CRC....41
7.2.2 Sample 2 With Data outside CRC....42
7.3 Sample Expansion EEPROM Content....44
7.3.1 Sample Chassis Expansion EEPROM....44
8 Standard Data Formats....45
8.1 Compressed ASCII PNPID....45
8.1.1 Definition....45
8.1.2 Example....45
8.2 EEPROM Specification Revision....45
8.2.1 Definition....45
8.2.2 Examples....45
9 Headers....46
9.1 EeeP.h....46
9.2 COM0EEP.h....54
10 Revision History....61

Dynamic Block Structures

Common Dynamic Block Header....24
Express Card Extended Topology....26
System Info....28
Chassis Info....29
Module Information....30
LFP Device Descriptor....31
Vendor Specific....32
CRC....32
Additional EEPROM descriptor....35

C Sample Code

Detect COM0 R2.0 FRUPROM....10
Detect COM0 R1.0 FRUPROM....11
Decode PCIe Link Width & Start Lane....21
CRC-CCITT implementation....33

MASM Sample Code

Decode PCIe Link Width & Start Lane....22
CRC-CCITT 386 implementation....34

EEPROM Header Structures

Common EEPROM Header....12
Universal Device Identifier....14
COM0 R2.0 Carrier Board EEPROM Header....15
Module EEPROM Header....23
Module EEPROM Header....23

Index of Tables

Table 1-1: Terms and Definitions....4
Table 4-1: Device Descriptor Byte....13
Table 5-1: USB Descriptor Byte....16
Table 5-2: SATA / SAS Device Descriptor Byte....17
Table 5-3: LAN Descriptor Byte....18
Table 5-4: Miscellaneous I/O Descriptor Byte....19
Table 5-5: Miscellaneous I/O Descriptor Byte2....19
Table 5-6: Digital Display Interface Descriptor Byte....20
Table 5-7: PCI Express Lane Descriptor Nibble Descriptor....21
Table 5-8: PCI Express Valid Starting Lane Configurations....21
Table 6-1: Com Express Ports....26
Table 6-2: Express Card Block Example 1....27
Table 6-3: Express Card Block Example 2....27
Table 6-4: System Information Descriptor Block Example....28
Table 6-5: Module Information Descriptor Block Example....29
Table 6-6: Chassis Information Descriptor Block Example....30
Table 6-7: LFP Device Descriptor Block Example....31
Table 6-8: Vendor Specific Block Example....32
Table 6-9: CRC16 Block Example....33
Table 6-10: Expansion EEPROM descriptor Example....35
Table 7-1: COM0R20 Type 2 Carrier Board With Expansion EEPROM....36
Table 7-2: COM0R20 Type 2 Carrier Board 38
Table 7-3: Sample 1 All in CRC....41
Table 7-4: Sample 2 With Data outside CRC....42
Table 7-5: Sample Chassis Expansion EEPROM....44

1 Introduction

1.1 Contributors

The following companies were members of the COM Express™ subcommittee and participated in developing this specification:

EDITOR Note: list of active companies at the time this is being updated

- Kontron

1.2 Name and Logo Usage

The PCI Industrial Computer Manufacturers Group policies regarding the use of its logos and trademarks are as follows:

Permission to use the PICMG® organization logo is automatically granted to designated members only as stipulated on the most recent Membership Privileges document (available at www.picmg.org) during the period of time for which their membership dues are paid. Nonmembers may not use the PICMG® organization logo.

The PICMG® organization logo must be printed in black or color as shown in the files available for download from the member's side of the Web site. Logos with or without the "Open Modular Computing Specifications" banner can be used. Nothing may be added or deleted from the PICMG® logo.

The use of the COM Express logo is a privilege granted by the PICMG® organization to companies who have purchased the relevant specifications (or acquired them as a member benefit), and that believe their products comply with these specifications. Manufacturers' distributors and sales representatives may use the COM Express logo in promoting products sold under the name of the manufacturer. Use of the logos by either members or non-members implies such compliance. Only PICMG Executive and Associate members may use the PICMG® logo. PICMG® may revoke permission to use logos if they are misused. The COM Express logo can be found on the PICMG web site, www.picmg.org.

The COM Express™ logo must be used exactly as shown in the files available for download from the PICMG® Web site. The aspect ratios of the logos must be maintained, but the sizes may be varied.Nothing may be added to or deleted from the COM Express™ logo.

The PICMG® name and logo and the COM Express name and logo are registered trademarks of PICMG®. Registered trademarks must be followed by the ® symbol, and the following statement must appear in all published literature and advertising material in which the logo appears:

PICMG, the COM Express name and logo and the PICMG logo are registered trademarks of the PCI

Industrial Computers Manufacturers Group.

1.3 Intellectual Property

The Consortium draws attention to the fact that it is claimed that compliance with this specification may involve the use of a patent claim(s) ("IPR"). The Consortium takes no position concerning the evidence, validity or scope of this IPR.

The holder of this IPR has assured the Consortium that it is willing to license or sublicense all such IPR to those licensees (Members and non-Members alike) desiring to implement this specification. The statement of the holder of this IPR to such effect has been filed with the Consortium.

Attention is also drawn to the possibility that some of the elements of this specification ***may*** be the subject of IPR other than those identified below. The Consortium ***shall*** not be responsible for identifying any or all such IPR.

No representation is made as to the availability of any license rights for use of any IPR inherent in this specification for any purpose other than to implement this specification.

This specification conforms to the current PICMG® Intellectual Property Rights Policy and the Policies and Procedures for Specification Development and does not contain any known intellectual property that is not available for licensing under Reasonable and Non-discriminatory terms. In the course of Membership Review the following disclosures were made:

1.3.1 Necessary Claims (referring to mandatory or recommended features)

No disclosures in this category were made during subcommittee review.

1.3.2 Unnecessary Claims (referring to optional features or non-normative elements)

No disclosures in this category were made during subcommittee review.

1.3.3 Third Party Disclosures

(Note that third party IPR submissions do not contain any claim of willingness to license the IPR.)

No disclosures in this category were made during subcommittee review.

Refer to PICMG® IPR Policies and Procedures and the company owner of the patent for terms and conditions of usage.

PICMG® makes no judgment as to the validity of these claims or the licensing terms offered by the claimants.

THIS SPECIFICATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE CONSORTIUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

Compliance with this specification does not absolve manufacturers of COM Express™ equipment from the requirements of safety and regulatory agencies (UL, CSA, FCC, IEC, etc.).

PICMG® and the COM Express™ logos are trademarks of the PCI Industrial Computer Manufacturers Group.

All other brand or product names may be trademarks or registered trademarks of their respective holder.

1.4 Copyright Notice

PICMG has made every attempt to ensure that the information in this document is accurate yet the information contained within is supplied “as-is”.

1.5 Trademarks

Intel and Pentium are registered trademarks of Intel Corporation. ExpressCard is a registered trademark of Personal Computer Memory Card International Association (PCMCIA). PCI Express is a registered trademark of Peripheral Component Interconnect Special Interest Group (PCI-SIG). COM Express is a registered trademark of PCI Industrial Computer Manufacturers Group (PICMG). I2C is a registered trademark of NXP Semiconductors. CompactFlash is a registered trademark of CompactFlash Association. Winbond is a registered trademark of Winbond Electronics Corp. AVR is a registered trademark of Atmel Corporation. Microsoft®, Windows®, Windows NT®, Windows CE and Windows XP® are registered trademarks of Microsoft Corporation. VxWorks is a registered trademark of WindRiver. All product names and logos are property of their owners.

1.6 Special Word Usage

Mandatory features are indicated by the use of the word “***shall***.”

Recommended features are indicated by the use of the word “***should***.”

159 Optional features are indicated by the use of the word "***may***."

160 *1.7 Acronyms / Definitions*

161 **Table 1-1: Terms and Definitions**

Term	Definition
a.out	a.out is a file format used in older versions of Unix-like computer operating systems for executable, object code, and, in later systems, shared libraries. The name stands for assembler output
AC '97	Audio CODEC (Coder-Decoder)
ACPI	Advanced Configuration Power Interface – standard to implement power saving modes in PC-AT systems
Atomic Error Checking	This Is used here to refer to the mechanism of validating all arguments before modifications are carried out.
Basic Module	COM Express™ 125mm x 95mm Module form factor.
BBS	Bios Boot Specification.
BIOS	Basic Input Output System – firmware in PC-AT system that is used to initialize system components before handing control over to the operating system.
Big-endian	Register Value: 0x0A0B0C0D Memory Order: 0x0A, 0x0B, 0x0C, 0x0D The most significant byte (*MSB*) value, which is `0x0A` in our example, is stored at the memory location with the lowest address, the next byte value in significance, `0x0B`, is stored at the following memory location and so on. This is akin to Left-to-Right reading in hexadecimal order.
Carrier Board	An application specific circuit board that accepts a COM Express™ Module.
CCTV	Closed Circuit Television
CVBS	Composite Video Baseband Signal
Compact Module	COM Express™ 95x95 Module form factor
DDC	Display Data Control – VESA (Video Electronics Standards Association) standard to allow identification of the Dynamic of a VGA monitor
DIMM	Dual In-line Memory Module
DisplayPort	DisplayPort is a digital display interface standard put forth by the Video Electronics Standards Association (VESA). It defines a new license free, royalty free, digital audio/video interconnect, intended to be used primarily between a computer and its display monitor.
DRAM	Dynamic Random Access Memory
DVI	Digital Visual Interface - a Digital Display Working Group (DDWG) standard that defines a standard video interface supporting both digital and analog video signals. The digital signals use TMDS.
EAPI	• Embedded Application Programming Interface Software interface for COM Express specific industrial functions • System information • Watchdog timer • I2C Bus • Flat Panel brightness control • User storage are. • GPIO
eDP	Embedded Display Port.
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFI	Extensible Firmware Interface(Next Generation BIOS)
EFI BIOS	Used to explicitly distinguish EFI and Legacy BIOS.
EFP	External Flat Panel
Extended Module	COM Express™ 155mm x 110mm Module form factor.
FR4	A type of fiber-glass laminate commonly used for printed circuit boards.
Gb	Gigabit
GBE	Gigabit Ethernet
GPI	General Purpose Input
GPIO	General Purpose Input Output

Term	Definition
GPO	General Purpose Output
HDA	Intel High Definition Audio (HD Audio) refers to the specification released by Intel in 2004 for delivering high definition audio that is capable of playing back more channels at higher quality than AC97.
HDMI	High Definition Multimedia Interface
I^2C	Inter Integrated Circuit – 2 wire (clock and data) signaling scheme allowing communication between integrated circuits, primarily used to read and load register values.
IDE	Integrated Device Electronics – parallel interface for hard disk drives – also known as PATA
Legacy Device	Relics from the PC-AT computer that are not in use in contemporary PC systems: primarily the ISA bus, UART-based serial ports, parallel printer ports, PS-2 keyboards, and mice. Definitions vary as to what constitutes a legacy device. Some definitions include IDE as a legacy device.
LAN	Local Area Network
Little-endian	Register Value: 0x0A0B0C0D Memory Order: 0x0D, 0x0C, 0x0B, 0x0A The least significant byte (*LSB*) value, 0x0D, is at the lowest address. The other bytes follow in increasing order of significance.
LFP	Local Flat Panel
LPC	Low Pin-Count Interface: a low speed interface used for peripheral circuits such as Super I/O controllers, which typically combine legacy-device support into a single IC.
LS	Least Significant
LVDS	Low Voltage Differential Signaling – widely used as a physical interface for TFT flat panels. LVDS can be used for many high-speed signaling applications. In this document, it refers only to TFT flat-panel applications.
MS	Most Significant
NA	Not Available
NC	No Connect
NTSC	National Television Standards Committee – video broadcast standard used in North America
OEM	Original Equipment Manufacturer
PAL	Phase Alternating Line – video broadcast standard used in many European countries.
PATA	Parallel AT Attachment – parallel interface standard for hard-disk drives – also known as IDE, AT Attachment, and as ATA
PC-AT	"Personal Computer – Advanced Technology" – an IBM trademark term used to refer to Intel x86 based personal computers in the 1990s
PCB	Printed Circuit Board
PCI	Peripheral Component Interface
PCI Express **PCIE**	Peripheral Component Interface Express – next-generation high speed Serialized I/O bus
PEG	PCI Express Graphics
PHY	Ethernet controller physical layer device
Pin-out Type	A reference to one of six COM Express™ definitions for the signals that appear on the COM Express™ Module connector pins.
PNPID	Microsoft Plug-And-Play ID (PNP ID). This ID can be registered at the Microsoft web page (http://www.microsoft.com/whdc/system/pnppwr/pnp/pnpid.mspx) free of charge. The PNP ID Format is XXX where 'A'<=X<='Z'
PS2 **PS2 Keyboard** **PS2 Mouse**	"Personal System 2" - an IBM trademark term used to refer to Intel x86 based personal computers in the 1990s. The term survives as a reference to the style of mouse and keyboard interface that were introduced with the PS2 system.
R_a	Roughness Average – a measure of surface roughness, expressed in units of length.
ROM	Read Only Memory – a legacy term – often the device referred to as a ROM can actually be written to, in a special mode. Such writable ROMs are sometimes called Flash ROMs. BIOS is stored in ROM or Flash ROM.
RTC	Real Time Clock – battery backed circuit in PC-AT systems that keeps system time and date as well as certain system setup parameters
SAS	Serial Attached SCSI – high speed serial version of SCSI
SCSI	Small Computer System Interface – an interface standard for high end disk drives and other computer peripherals
SPD	Serial Presence Detect – refers to serial EEPROM on DRAMs that has DRAM Module configuration information
SPI	Serial Peripheral Interface

Term	Definition
SO-DIMM	Small Outline Dual In-line Memory Module
S0, S1, S2, S3, S4, S5	System states describing the power and activity level S0 Full power, all devices powered S1 S2 S3 Suspend to RAM System context stored in RAM; RAM is in standby S4 Suspend to Disk System context stored on disk S5 Soft Off Main power rail off, only standby power rail present
SATA	Serial AT Attachment: serial-interface standard for hard disks
SDVO	Serialized Digital Video Output – Intel defined format for digital video output that can be used with Carrier Board conversion ICs to create parallel, TMDS, and LVDS flat-panel formats as well as NTSC and PAL TV outputs
SM Bus	System Management Bus
Super I/O	An integrated circuit, typically interfaced via the LPC bus that provides legacy PC I/O functions including PS2 keyboard and mouse ports, serial and parallel port(s) and a floppy interface.
TFT	Thin Film Transistor – refers to technology used in active matrix flat-panel displays, in which there is one thin film transistor per display pixel.
TMDS	Transition Minimized Differential Signaling - a digital signaling protocol between the graphics subsystem and display. TMDS is used for the DVI digital signals.
TPM	Trusted Platform Module, chip to enhance the security features of a computer system.
USB	Universal Serial Bus
VGA	Video Graphics Adapter – PC-AT graphics adapter standard defined by IBM.
WDT	Watch Dog Timer.
XAUI	10 Gigabit / sec Attachment Unit Interface.

1.8 Applicable Documents and Standards

The following publications are used in conjunction with this standard. When any of the referenced specifications are superseded by an approved revision, that revision ***shall*** apply. All documents ***may*** be obtained from their respective organizations.

- Advanced Configuration and Power Interface Specification Revision 4.0, June 16, 2009 Copyright © 1996-2003 Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation. All rights reserved. http://www.acpi.info/
- ANSI/TIA/EIA-644-A-2001: Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits, January 1, 2001. http://www.ansi.org/
- ANSI INCITS 361-2002: AT Attachment with Packet Interface - 6 (ATA/ATAPI-6), November 1, 2002. http://www.ansi.org/
- ANSI INCITS 376-2003: American National Standard for Information Technology – Serial Attached SCSI (SAS), October 30, 2003. http://www.ansi.org/
- Audio Codec '97 Revision 2.3 Revision 1.0, April 2002 Copyright © 2002 Intel Corporation. All rights reserved. download.intel.com/support/motherboards/desktop/sb/ac97_r23.pdf
- Display Data Channel Command Interface (DDC/CI) Standard (formerly DDC2Bi) Version 1, August 14, 1998 Copyright © 1998 Video Electronics Standards Association. All rights reserved. http://www.vesa.org/summary/sumddcci.htm
- ExpressCard® Standard 2.0, June 2009 Copyright © 2009 PCMCIA. All rights reserved. http://www.expresscard.org/
- HDA - High Definition Audio Specification, Revision 1.0, April 15, 2004 Copyright © 2002 Intel Corporation. All rights reserved. http://www.intel.com/standards/hdaudio/
- IEEE 802.3-2005, IEEE Standard for Information technology, Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications." http://www.ieee.org
- IEEE 802.3ae (Amendment to IEEE 802.3-2005), Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation. http://www.ieee.org
- Intel Low Pin Count (LPC) Interface Specification Revision 1.1, August 2002 Copyright © 2002 Intel Corporation. All rights reserved. http://developer.intel.com/design/chipsets/industry/lpc.htm
- PCI Express Base Specification Revision 2.0, December 20, 2006, Copyright © 2002-2006 PCI Special Interest Group. All rights reserved. http://www.pcisig.com/
- PCI Express Card Electromechanical Specification Revision 2.0, April 11, 2007, Copyright © 2002-2007 PCI Special Interest Group. All rights reserved. http://www.pcisig.com/
- PCI Local Bus Specification Revision 3.0, February 3, 2004 Copyright © 1992, 1993, 1995, 1998, and 2004 PCI Special Interest Group. All rights reserved. http://www.pcisig.com/
- PICMG® Policies and Procedures for Specification Development, Revision 2.0, September 14, 2004, PCI Industrial Computer Manufacturers Group (PICMG®), 401 Edgewater Place, Suite 500, Wakefield, MA 01880 USA, Tel: 781.224.1100, Fax: 781.224.1239. http://www.picmg.org/

- PICMG® EAPI - Embedded Application Software Interface Specification, Revision 1.0, 2009, PCI Industrial Computer Manufacturers Group (PICMG®), 401 Edgewater Place, Suite 500, Wakefield, MA 01880 USA, Tel: 781.224.1100, Fax: 781.224.1239. http://www.picmg.org/
- SDIO, Secure Digital Input/Output
  SD Specifications Part E1 SDIO Specification Version 2.00, February 8, 2007
  Copyright 2007 SD Card Association
  http://www.sdcard.org
- Serial ATA: High Speed Serialized AT Attachment Revision 1.0a January 7, 2003
  Copyright © 2000-2003, APT Technologies, Inc., Dell Computer Corporation, Intel Corporation, Maxtor Corporation, Seagate Technology LLC. All rights reserved.
  http://www.sata-io.org/
- Smart Battery Data Specification Revision 1.1, December 11, 1998.
  www.sbs-forum.org
- System Management Bus (SMBus) Specification Version 2.0, August 3, 2000
  Copyright © 1994, 1995, 1998, 2000 Duracell, Inc., Energizer Power Systems, Inc., Fujitsu, Ltd., Intel Corporation, Linear Technology Inc., Maxim Integrated Products, Mitsubishi Electric Semiconductor Company, PowerSmart, Inc., Toshiba Battery Co. Ltd., Unitrode Corporation, USAR Systems, Inc. All rights reserved.
  http://www.smbus.org/
- USB 3.0 Specification, Revision 1.0, November 12, 2008, 2000
  Copyright © 2007-2008 Hewlett-Packard Company, Intel Corporation, , Microsoft Corporation, NEC Corporation, ST-NXP Wireless and Texas Instruments. All rights reserved.
  http://www.usb.org/
- SPI, Serial Peripheral Interface Bus
  http://elm-chan.org/docs/spi_e.html
- Trusted Platform Module (TPM), Trusted Computing Group Specification 1.2 Revision 103, July 9, 2007, http://www.trustedcomputinggroup.org
- DisplayPort Standard Version 1.1 http://www.vesa.org
- High-Definition Multimedia Interface specification version 1.3 http://www.hdmi.org

1.9 Statement of Compliance

Statements of compliance with this specification take the form specified in the PICMG® Policies and Procedures for Specification Development:

"This product provides software support for PICMG® EEEP Revision 1.0."

Products making this simple claim of compliance must provide, at a minimum, all features defined in this specification as being mandatory by the use of the keyword "***shall***" in the body of the specification. Such products ***may*** also provide recommended features associated with the keyword "***should***" and permitted features associated with the keyword "***may***" as well.

Because the specification provides for a number of recommended and permitted features beyond the mandatory minimum set and a wide range of performance Dynamic, more complete descriptions of product compliance are encouraged

2 General

The purpose of this specification is to provide a standardized method for data storage using EEPROMs in modern Multi-Vendor enviroments.

3 Detection

3.1 Detecting EeeP EEPROM

3.1.1 High Level Check

```
if(!memcmp(
        &EeePEEP[0x01]          ,
        "3P"                    ,
        0x02
      )
  )
{
   // Found EeeP EEPROM
}
```

C_SAMPLE 1: Detect COM0 R2.0 FRUPROM

3.1.2 Sample I2C Transfer

Device Address : 0xAE(0x57)

Index Type : Standard

Start<0x57><W>Ack<0x00>Ack

Start<0x57><R>Ack<0xXX>Ack<'3'>Ack<'P'>Ack<'0x10'>Nak Stop

Device Address : 0xAE(0x57)

Index Type : Extended

Start<0x57><W>Ack<0x00>Ack<0x00>Ack

Start<0x57><R>Ack<0xXX>Ack<'3'>Ack<'P'>Ack<'0x10'>Nak Stop

XX is used to designate don't care, byte.

3.2 Detecting COM0 R1.0 EEPROM

3.2.1 High Level Check

```
if(!memcmp(
        &COM0EEP[0xE0]          ,
        "COMExpressConfig"      ,
        0x10
      )
  )
{
   // Found COM0R10 EEPROM
}
```

C_SAMPLE 2: Detect COM0 R1.0 FRUPROM

3.2.2 Sample I2C Transfer

Device Address : 0xAE(0x57)

Index Type : Standard

Start<0x57><W>Ack<0xE0>Ack

Start<0x57><R>Ack<'C'>Ack<'O'>Ack<'M'>Ack<'E'>Ack<'x'>Ack<'p'>Ack
<'r'>Ack<'e'>Ack<'s'>Ack<'s'>Ack<'C'>Ack<'o'>Ack
<'n'>Ack<'f'>Ack<'i'>Ack<'g'>Nak Stop

3.3 Identifying Device Type Standard/Extended Index

3.3.1 Sample I2C Transfer

Device Address : 0xAE(0x57)

Index Type : Unknown

Start<0x57><W>Ack<0x00>Ack<0x01>Ack

Start<0x57><R>Ack<'3'>Ack<'P'>Ack<'0x10'>Ack<'0xXX'>Ack<'DeviceDesc'>Nak
Stop

3.3.1.1 Standard Indexed Device

With a Standard Indexed device the MSB Of the Extended Index is uses as the Complete Index and the LSB is written to the Don't care Byte of the EEPROM, causing an increment to the internal counter. Starting the Read operation at Device Offset 0x01.The Device Descriptor at offset 5 can then be used to interpret the Device correct method of device access.

3.3.1.2 Extended Indexed Device

With an Extended Indexed device the Index is programed as normal. Starting the Read operation at Device Offset 0x0001.

4 EEPROM Header Components.

4.1 Concepts

There are multiple goals for an EEPROM header. One of the primary goals is to provide a mechanism for content Identification. This goal is achieved through inclusion of a header Id. A Secondary Goal is to provide information at fixed offsets. The purpose of providing information at fixed offsets is to allow for easier access. One of the costs of storing dynamic information in an EEPROM is that at an early system state it may not be convenient or possible to parse complex structures or hierarchies in order to extract information. To this extend it is advisable to segregate the information we wish to store in EEPROMs into 2 categories. Low Cost Access Information and Normal Cost Access Information. The Low Cost Access Information may be stored in the EEPROM Header. To Prevent Contention Header space is allocated in the following order.

1. EeeP Specific Header components.
2. Platform Specific Header Additions.
3. Vendor Specific Header Additions.

4.2 Common EeeP EEPROM Header

```
typedef struct EeePCmn_s{
    uint8_t     DontCareByte;/* 0x00 Don't Care Byte
                             *      The purpose of this
                             *      Byte is to reduce the Damage
                             *      Extended Index read access is
                             *      used on a Standard Index Device
                             *
                             */
    uint8_t     EepId[2]    ; /* 0x01 3P   */
#       define  EEEP_EEPROM_MARKER    "3P"
    uint8_t     SpecRev     ; /* 0x03 EeeP Specification Revision
                             */

    uint8_t     BlkOffset   ; /* 0x04 Absolute Offset to
                             *      First Dynamic Block
                             *      in words(2 bytes)
                             */
    uint8_t     DeviceDesc ; /* 0x05 Device Descriptor
                             */
}EeePCmn_t;
```

EEP_HDR_C_STRUCT 1: Common EEPROM Header

4.2.1 Don't Care Byte:

The purpose of this Byte is to reduce the the damage, should the incorrect access method be used on a standard EEPROM. I.E. extended Index read access is used to read from a Standard Index Device. In this scenario the second Command/Index byte is interpreted as a Data byte and is written to the MSB Index.

4.2.2 EEPROM ID

EEPROM Identifier. Defined as ASCII “3P”. To allow EEPROM Identification.

4.2.3 EeeP Specification Revision.

See 8.2 EEPROM Specification Revision. on page 45.

4.2.4 Block Offset:

Absolute offset from the start of the EEPROM to the first Dynamic Block. If 0 Dynamic Blocks Aren't supported. It is measured in units of 2 bytes.

4.2.5 Device Descriptor Byte

Table 4-1: Device Descriptor Byte

Bits	Description	Values	Meaning
5 - 7	Page Write Length/Alignment The Maximum number of internal low order address bits auto incremented on page writes	0	1 Byte
		1	8 Bytes
		2	16 Bytes
		3	32 Bytes
		4	64 Bytes
		5	128 Bytes
		6	256 Bytes
		7	512 Bytes
4	Standard/Extended Index	0	Standard Index/Command
		1	Extended Index/Command
0 - 3	Device Size Size 2^(8+n) Bytes (256 << n) Bytes Addressable Bytes Standard Index 8bit - 11Bit Extended Index 16bit - 19Bit 2048 KBit - 4096 MBit 256 Bytes - 512 KBytes	0x01	256 Bytes 2 KBits
		0x02	512 Bytes 4 KBits
		0x03	1 KBytes 8 KBits
		0x04	2 KBytes 16 KBits
		0x05	4 KBytes 32 KBits
		0x06	8 KBytes 64 KBits
		0x07	16 KBytes 128 KBits
		0x08	32 KBytes 256 Kbits
		0x09	64 KBytes 512 Kbits
		0x0A	128 KBytes 1 Mbits
		0x0B	256 KBytes 2 Mbits
		0x0C	512 KBytes 4 Mbits
		0x0D - 0x0F	Reserved

EEPROM Header Components.

4.3 Universal Device Identifier

```
typedef struct UDIdEep_s{
    uint8_t     VendId[2]  ; /* 0x06 Vendor Unique PNPID         */
    uint8_t     DeviceId[2]; /* 0x08 Vendor Specific Device ID   */
    uint8_t     DeviceFlav ; /* 0x0A Device Specific Flavor ID   */
    uint8_t     RevId      ; /* 0x0B Device Specific Revision ID */
}UDIdEep_t;
```

EEP_HDR_C_STRUCT 2: Universal Device Identifier

Standardized compact mechanism to uniquely identify device revisions. The Vendor ID and Device ID can combined to form a PNP-ISA ID. The Primary goal of this structure is to provide a standardized mechanism for dealing with eccentric device behavior, by allowing easy Identification.

4.3.1 Vendor ID

Compressed ASCII PNPID(see chapter 8.1 page 45).

4.3.2 Device ID

Vendor Assigned 16Bit value to uniquely Identify all products. Stored In 'Big-endian' (see page 4).

4.3.3 Device Flavor

Vendor Assigned 8Bit value to distinguish product flavors I.E. to distinguish stuffing options.

4.3.4 Device Unique Revision ID

Vendor Assigned Device Revision ID. No attempt is made here to give significance to this value. No assumptions should be made as to the meaning of this value, not even the assumption that higher digits mean newer product revisions. This is to allow for the complex nature of product revisions in the embedded market. I.E. the following mapping is allowed.

RID	Board Revision
0	0,5
1	1
2	1,1
3	1.0.1
4	1.0.1.2

5 EEPROM Header Examples

5.1 COM0 R2.0 Carrier Board EEPROM Header

```
typedef struct COM0R20_CB_s{
    EeePCmn_t    EeePHdr  ; /* 0x00 EeeP Common Header */
    uint8_t      GenId[4] ; /* 0x06 "COM0"   */
    UDIdEep_t    DevId    ; /* 0x0C Unique Device Id   */
    uint8_t      CBType   ; /* 0x10 Carrier Board Type     */
    uint8_t      SpecRev  ; /* 0x11 COM0 Specification Revision */
    uint8_t      UsbDesc  ; /* 0x12 USB Descriptor Byte */
    uint8_t      SasDesc;   /* 0x13 LAN Descriptor Byte */
    uint8_t      LanDesc;   /* 0x14 LAN Descriptor Byte */
    uint8_t      MiscIo1;   /* 0x15 Miscellaneous I/O Descriptor Byte 1 */
    uint8_t      MiscIo2;   /* 0x16 Miscellaneous I/O Descriptor Byte 2 */
    uint8_t      DDIDesc;   /* 0x17 Digital Display Interface Descriptor
                             * Byte
                             */
    uint8_t      Reserved0; /* 0x18 Reserved */
    uint8_t      Reserved1; /* 0x19 Reserved */
    uint8_t     LaneMap[16];/* 0x1A PCI Express Lane Information */
}COM0R20_CB_t;
```

EEP_HDR_C_STRUCT 3: COM0 R2.0 Carrier Board EEPROM Header

5.1.1 EeeP Header

Standard EeeP EEPROM Header

5.1.2 COM0 ID

COM0 Header Id to allow for identification of COM0 Specific Header Content.

5.1.3 Device Id

Unique Device Identification Structure. To allow Identification of COM0 Carrier Board.

5.1.4 Carrier Board Type.

Carrier Board Type.

5.1.5 COM0 Specification Revision.

See 8.2 EEPROM Specification Revision. on page 45.

5.1.6 USB Descriptor Byte

The configuration EEPROM ***shall*** use one byte to indicate how many USB ports the Carrier Board uses. Note that per the Module fill order, described in Section 3.3 of this document, the USB ports ***shall*** be populated in a low to high manner.

365 **Table 5-1: USB Descriptor Byte**

Bits	Description	Values	Meaning
7	Reserved	0	
4 - 6	USB 3.0 Port Count.	0	No Ports Implemented
		1	1 Port Implemented
		...	
		4	4 Ports Implemented
0 - 3	USB Port Count	0	No Ports Implemented
		1	1 Port Implemented
		...	
		8	8 Ports Implemented

366

367 5.1.7 SATA / SAS Device Descriptor Byte

368 The configuration EEPROM ***shall*** use a byte to indicate how many SATA and SAS devices
369 the Carrier Board uses, per the following table.

370 **Table 5-2: SATA / SAS Device Descriptor Byte**

Channel Number	Bit	Description	Values	Meaning
SATA / SAS Channel 4	7	Port Type	0	SATA Device
			1	SAS Device
	6	Port Implemented	0	Not Implemented
			1	Implemented
SATA / SAS Channel 3	5	Port Type	0	SATA Device
			1	SAS Device
	4	Port Implemented	0	Not Implemented
			1	Implemented
SATA / SAS Channel 2	3	Port Type	0	SATA Device
			1	SAS Device
	2	Port Implemented	0	Not Implemented
			1	Implemented
SATA / SAS Channel 1	1	Port Type	0	SATA Device
			1	SAS Device
	0	Port Implemented	0	Not Implemented
			1	Implemented

5.1.8 LAN Descriptor Byte

The configuration EEPROM ***shall*** use one byte to indicate the LAN options used by the Carrier Board. The following table shows the LAN Descriptor Byte definition.

Table 5-3: LAN Descriptor Byte

Description	Bits	Values	Meaning
Reserved	3- 7		
GBE2.	2	0	Not Implemented
		1	Implemented
GBE1.	1	0	Not Implemented
		1	Implemented
GBE0.	0	0	Not Implemented
		1	Implemented

5.1.9 Miscellaneous I/O Descriptor Byte

The configuration EEPROM ***shall*** use one byte to indicate usage by the Carrier Board of the following miscellaneous I/O signals.

- AC ’97 / HDA Digital Interface
- Watchdog Timer
- External BIOS ROM
- Thermal Protection
- Battery Low
- Suspend
- Wake

The following table shows the Miscellaneous I/O Descriptor Byte definition.

386 **Table 5-4: Miscellaneous I/O Descriptor Byte**

Feature/Function	Bits	Values	Description
AC '97 / HDA (AC/HD_**** pins)	7	0	Not Implemented
		1	Implemented
External BIOS ROM (BIOS_DISABLE#)	6	0	Not Implemented
		1	Implemented
Battery Low (BATLOW#)	5	0	Not Implemented
		1	Implemented
Wake 1 (WAKE1#)	4	0	Not Implemented
		1	Implemented
AC '97 / HDA (AC/HD_**** pins)	3	0	Not Implemented
		1	Implemented
External BIOS ROM (BIOS_DISABLE#)	2	0	Not Implemented
		1	Implemented
Battery Low (BATLOW#)	1	0	Not Implemented
		1	Implemented
Wake 1 (WAKE1#)	0	0	Not Implemented
		1	Implemented

387 ### 5.1.10 Miscellaneous I/O Descriptor Byte 2

388 The configuration EEPROM ***shall*** use one byte to indicate usage by the Carrier Board of
389 the following miscellaneous I/O signals.

390 • SSC

391 • SD Card I/O

392 **Table 5-5: Miscellaneous I/O Descriptor Byte2**

Feature/Function	Bits	Values	Description
Reserved.	2 - 7	0	
SDIO on GPIO	1	0	Not Implemented
		1	Implemented
Spread Spectrum Clocking	0	0	Not Implemented
		1	Implemented

393 ### 5.1.11 Digital Display Interface Descriptor Byte

394 The configuration EEPROM ***shall*** use one byte to indicate the digital display interface
395 connections supported. This byte is only used for Module Type 6. The following table
396 shows the Display Descriptor Byte definition.

Table 5-6: Digital Display Interface Descriptor Byte

DDI Port	Bits	Values	Description
DDI3	6 - 7	0	Not Implemented
		1	Embedded Display Port Implemented
		2	Display Port Implemented
		3	HDMI/DVI Implemented
Reserved	5	0	Reserved
DDI2	3 - 4	0	Not Implemented
		1	Embedded Display Port Implemented
		2	Display Port Implemented
		3	HDMI/DVI Implemented
DDI1	0 - 2	0	Not Implemented
		1	Embedded Display Port Implemented
		2	Display Port Implemented
		3	HDMI/DVI Implemented
		4	SDVO Implemented
		5 - 7	Reserved

Notes: further details about the type of video signals are beyond the scope and purpose of the configuration

5.1.12 PCI Express Lane Descriptor Data Structure

Lane numbers 0 through 31 refer to the 32 possible COM Express™ lanes. Each COM Express™ connector PCI Express lane is allocated four bits to describe how the lanes are grouped on the Carrier Board to form PCI Express links.

The 32 PCIe lanes are described in an array of 16 bytes, each lane using 4 bits. A lane is described by three bits indicating the width of the link it is part of, and a single bit defining the PCIe Generation of the link. This definition implies that the start lane of a link must be aligned at a boundary which is an integral multiple of the width of the link it is part of, e.g. an x4link can start only at lane no. 0,4,8,16,20,24, and 28.

A link (along with its width and starting lane) can be decoded by inspecting the single Mapping Nibble of a lane which is part of the link. Figure "C_SAMPLE 3: Decode PCIe Link Width & Start Lane" on page 21 is showing this for lane 15.".

412 **Table 5-7: PCI Express Lane Descriptor Nibble Descriptor**

Bits	Usage	Values	Description
0 - 3	Link Width/Alignment	0	Not Implemented
		1	x1 Link Width
		2	x2 Link Width
		3	x4 Link Width
		4	x8 Link Width
		5	x16 Link Width
		6	x32 Link Width
		7	Reserved
3	PCI Express Generation.	0	PCI Express Gen 1.
		1	PCI Express Gen 2.

413 **Table 5-8: PCI Express Valid Starting Lane Configurations**

Link Width	Valid Starting Lane
x1	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,...
x2	0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
x4	0,4,8,12,16,20,24,28
x8	0,8,16,24
x16	0,16
x32	0

```
unsigned LaneIndex=15                          // Example for Lane 15
unsigned ArrayIndex=LaneIndex/2          // 7
unsigned ByteShift =(LaneIndex&1)<<2     // Either 4, 0
unsigned EncLinkWidth =(LaneMapping[ArrayIndex]>>ByteShift)&0x7
if(EncLinkWidth){
  EncLinkWidth--;
  // Lane Used
  unsigned LinkWidth=1<<(EncLinkWidth)  // 1,2,4,8,16,32
  // The Following 3 lines Are equivilant
  // and are simply provided to aid in
  // understanding
#if 0
  unsigned StartingLane=(LaneIndex/LinkWidth)*LinkWidth;
#elseif 0
  unsigned StartingLane=(LaneIndex>>EncLinkWidth)<<EncLinkWidth;
#else
  unsigned StartingLane=LaneIndex&(~(LinkWidth - 1));
#endif
}
```

C_SAMPLE 3: Decode PCIe Link Width & Start Lane

415

```
LinkWidth TEXTEQU <ah>
StartLane TEXTEQU <al>

GetLaneInfo  PROC NEAR STDCALL PUBLIC \
                        USES BX CX,   \
                        LaneIndex:BYTE
        movzx       bx, LaneIndex
        ; Adjust Lane index to Array Index
        ; Load Bit 0 to Carry Flag
        shr         bx, 1
        ; Get Lane Mapping Information
        mov         bl, cs:LaneMap[bx]
        jnc         @F
        ; Adjust for High/Low nibble
        shr         bl, 4
@@:
        ; Prepare for Unused
        xor         ax, ax
        ; Mask Don't care bits
        and         bl, 7
        jz          UnusedLane
        mov         cl, bl
        dec         cl
        mov         bl, 1
        shl         bl, cl
        mov         LinkWidth, bl
        ; Delete invalid Bits
        ; to get starting lane
        dec         bl
        not         bl
        mov         StartLane, LaneIndex
        and         StartLane, bl
UnusedLane:

        ret
GetLaneInfo  ENDP
```

MASM_SAMPLE 1: Decode PCIe Link Width & Start Lane

5.2 COM0 R2.0 Module EEPROM Header

```
typedef struct COM0R20_M_s{
    EeePCmn_t   EeePHdr  ; /* 0x00 EeeP Common Header */
    uint8_t     GenId[4] ; /* 0x06 "COM0"   */
    UDIdEep_t   DevId    ; /* 0x0A Unique Device Id   */
    uint8_t     MType    ; /* 0x10 Module Type        */
    uint8_t     SpecRev  ; /* 0x11 COM0 Specification Revision */
}COM0R20_M_t;
;
```

EEP_HDR_C_STRUCT 4: Module EEPROM Header

5.2.1 EeeP Header

Standard EeeP EEPROM Header

5.2.2 COM0 ID

COM0 Header Id to allow for identification of COM0 Specific Header Content.

5.2.3 Device Id

Unique Device Identification Structure. To allow for Identification of COM0 Modules.

5.2.4 Module Type.

Module Type.

5.2.5 COM0 Specification Revision.

See 8.2 EEPROM Specification Revision. on page 45.

5.3 Expansion EEPROM Header

```
typedef struct Exp_EEP_s{
    EeePCmn_t   EeePHdr  ; /* 0x00 EeeP Common Header */
    UDIdEep_t   DevId    ; /* 0x06 Unique Device Id   */
}Exp_EEP_t;
```

EEP_HDR_C_STRUCT 5: Module EEPROM Header

5.3.1 Description

Skeleton Header For Expansion EEPROMs.

6 Dynamic Descriptor Blocks.

6.1 Overview.

This is a method to allow dynamic reuse of a finite space. The theory is that different devices will wish to describe different features and that not all devices will wish to describe all features. Additional this allows for the fact that different devices may require different space to describe the same features.

The mechanism is simple, it is in effect a linked list of descriptors. Each descriptor has an ID and and its own size., Blocks are defined as being contiguous.

Once a starting point is defined it is possible to parse the link list to discover those block which if present configure or define the current operation. One of the most useful application of this mechanism is to allow for the optional description of complex topologies for express card slots.

It should also be noted here that the goal of these link list systems is not to require dynamic generation but to allow dynamic parsing. It is valid to have vendor or platform specific fixed offset, wrapped with link list structures.

At run time the EEPROM layout is to be viewed as Static. All changes to the EEPROM structure **shall** be immediately followed by a system restart. This is to allow for caching of EEPROM Offsets to reduce EEPROM access Overhead. EEPROM Layout **shall** only be modified in the context of either system maintenance or system manufacturing.

6.2 Common Dynamic Block Header

```
ypedef struct DBlockIdHdr_s{
    uint8_t   DBlockId       ; /* 0x00 Dynamic Block Id       */
    uint8_t   DBlockLength[2]; /* 0x01 Block Length/
                                *      Offset to next Block
                                *      in words (2 Bytes)
                                */
}DBlockIdHdr_t;
```

CAP_C_STRUCT 1: Common Dynamic Block Header

6.2.1 Dynamic Block Ids

Designates the Dynamic Block Type.

Id	Description
0x00	unused
0x01-0xBF	Reserved
0xC0 - 0xCF	Standard/Platform Specific Block
0xD1	System Descriptor
0xD2	Module Descriptor
0xD3	Chassis Descriptor
0xD4	LFP Device Data
0xD5-0xCF	Reserved
0xE0	Express Card Topology
0xE1-0xEF	Reserved
0xF0	Vendor Specific
0xF1	Expansion EEPROM Descriptor
0xF2	CRC
0xF3-0xFE	Reserved
0xFF	Unknown should be ignored

6.2.2 Dynamic Length/Offset

Specifies the size of the current block in bytes. Stored in 'Big-endian' (see page 4).

Offset Value	Description
0x0000	End of list
0x0001-0xFFFE	Offset to next Tag
0xFFFF	End of list

6.3 Express Card Topology

The configuration EEPROM **may** use two blocks to indicate if the Carrier Board supports up to 2 Express Card slots. If no blocks are present the carrier is assumed to have no Express Card Slots.

Dynamic Descriptor Blocks.

```
typedef struct ExpCardBlock_s{
    DBlockIdHdr_t  DBHdr         ; /* 0x00 Dynamic Block Header */
    uint8_t ExpressCardNumber    ; /* 0x03 0 - 1 */
    uint8_t ComExpressPorts      ; /* 0x04
                                   *      +=======+==================+
                                   *      | Bits  | Descriptions     |
                                   *      +=======+==================+
                                   *      | 0 - 4 | PCI Express Port |
                                   *      |       | Mapping.         |
                                   *      |       | Port Starting    |
                                   *      |       | Lane.            |
                                   *      |       | 0 - 31           |
                                   *      +-------+------------------+
                                   *      | 5 - 7 | USB Port Mapping |
                                   *      |       | 0 - 7            |
                                   *      +=======+==================+
                                   */
    uint8_t SwitchPortNumber[1] ; /* 0x05 An Array of
                                   *      Port/Device
                                   *      numbers Terminated
                                   *      with 0x7F
                                   */
}ExpCardBlock_t;
```

CAP_C_STRUCT 2: Express Card Extended Topology

6.3.1.1 Express Card Number

Number designating Express Card Slot Number.

6.3.1.2 Com Express Port

Table 6-1: Com Express Ports

Bits	Description	Value	Meaning
5 - 7	Express Card USB Port Mapping	0	USB Port 0
		...	
		7	USB Port 7
0 - 4	Express Card PCI Express Port Mapping. Port Starting Lane.	0	PCI Express Lane 0
		...	
		31	PCI Express Lane 31

6.3.1.3 Express Port Number

A 0x7F terminated array of Port numbers. Designating in order the topological mapping of the Express Card Slot. This is to allow for the description of complex Topologies where then Express card slot is not connected directly to the COMExpress Connector, but is located behind 1 or more PCI Express Packet Switches.

6.3.2 Examples:

Table 6-2: Express Card Block Example 1

Offset	Width	Name	Value	Description
0	1	Block Id	0xE0	Express Card Extended Topology
1	2	Block Length	6	Relative Offset to Next Dynamic Block
3	1	Express Card Number	0x01	Express Card Slot 2
4		Com Express Port{0:4}	2	PCI Express Lane 2
4	1	Com Express Port{5:7}	1	USB Port 1
5	1	Switch Port Number[0]	0x7F	End of Port List

Table 6-3: Express Card Block Example 2

Offset	Width	Name	Value	Description
0	1	Block Id	0xE0	Express Card Extended Topology
1	2	Block Length	5	Dynamic Block Size (Words)
3	1	Express Card Number	0x01	Express Card Slot 2
4		Com Express Port{0:4}	16	PCI Express Lane 16
4	1	Com Express Port{5:7}	1	USB Port 1
5	1	Switch Port Number[0]	1	1st port on Switch 1
6	1	Switch Port Number[1]	3	3rd port on Switch 2
7	1	Switch Port Number[2]	3	3rd port on Switch 3
8	1	Switch Port Number[3]	0x7F	End of Port List
9	1	Pack Byte	0x00	Word Alignment Pack Byte

This is a Descriptor for Express Card 1

1. It is Attached to a Switch which is attached to the PEG port, so the
   - starting Lane = 16.
1. Look up the PEG port bus number, lets say its bus 1.
2. Now look for a device at Bus=01,Device=01. It is a Bridge so we look up the Bus number. Lets say its 2
3. Now look for a device at Bus=02,Device=03. It is a Bridge so we look up the Bus number. Lets say its 4
4. Now look for a device at Bus=04,Device=03. It is a Bridge so we look up the Bus number. Lets say its 5

Dynamic Descriptor Blocks.

5. So the Next Port number == 0x7F so we know we now have our Express Card Port. Namely
   - Express Card Port Bridge is Bus=04, Device=03.
   - Device Address is Bus 5.

6.4 System Information Descriptor Block

```
typedef struct SystemInfo_s{
    DBlockIdHdr_t DBHdr    ; /* 0x00 Dynamic Block Header  */
    uint8_t Manufacturer   ; /* 0x03 Number of ASCIIZ String */
    uint8_t ProductName    ; /* 0x04 Number of ASCIIZ String */
    uint8_t Version        ; /* 0x05 Number of ASCIIZ String */
    uint8_t SerialNumber   ; /* 0x06 Number of ASCIIZ String */
    uint8_t SKU_Number     ; /* 0x07 Number of ASCIIZ String */
    uint8_t Family         ; /* 0x08 Number of ASCIIZ String */
    uint8_t AssetTagNumber; /* 0x09 Number of ASCIIZ String */
    char StringsBlock[1] ;  /* 0x0A String Block */
}SystemInfo_t;
```

CAP_C_STRUCT 3: System Info

6.4.1 Description

This Block is loosely based on the SMBIOS 2.6 standards System Description block. The Primary use of this block is to populate the SMBIOS block type 1 if supported.

Table 6-4: System Information Descriptor Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xD1	System Descriptor
1	2	Block Length	24	Dynamic Block Size (Words)
3	1	Manufacturer	1	String 1
4	1	Product Name	2	String 2
5	1	Version	3	String 3
6	1	Serial Number	4	String 4
7	1	SKU Number	0	String 0
8	1	Family	5	String 5
9	1	Asset Tag Number	0	String0
10	1	String0	“\0”	Zero Length String “Unused”
11	6	String1	“PICMG\0”	
17	11	String2	“COMExpress\0”	
28	8	String3	“1.1.0.1\0”	
36	6	String4	“SN123\0”	
42	5	String5	“COM0\0”	
47	1	Pack Byte	0x00	Word Alignment Pack Byte

6.5 Chassis Information Descriptor Block

```
typedef struct ChassisInfo_s{
    DBlockIdHdr_t DBHdr   ; /* 0x00 Dynamic Block Header  */
    uint8_t Manufacturer  ; /* 0x03 Number of ASCIIZ String */
    uint8_t ChassisType   ; /* 0x04 ENUM */
    uint8_t Version       ; /* 0x05 Number of ASCIIZ String */
    uint8_t SerialNumber  ; /* 0x05 Number of ASCIIZ String */
    uint8_t AssetTagNumber; /* 0x06 Number of ASCIIZ String */
    char  StringsBlock[1] ; /* 0x07 String Block */
}ChassisInfo_t;
```

CAP_C_STRUCT 4: Chassis Info

6.5.1 Description

This block is loosely based on the SMBIOS 2.6 standards CHASIS Description block. The Primary use of this block is to populate the SMBIOS block type 3, if supported.

Table 6-5: Module Information Descriptor Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xD3	Chassis Descriptor
1	2	Block Length	23	Dynamic Block Size (Words)
3	1	Manufacturer	1	String 1
4	1	Chassis Type		Chassis Type ENUM
5	1	Version	3	String 3
6	1	Serial Number	4	String 4
7	1	Asset Tag	0	String 0
8	1	String0	“\0”	Zero Length String “Unused”
9	6	String1	“PICMG\0”	
15	11	String2	“COMExpress\0”	
26	8	String3	“1.1.0.1\0”	
34	6	String4	“SN123\0”	
40	5	String5	“COM0\0”	
45	1	Pack Byte	0x00	Word Alignment Pack Byte

Dynamic Descriptor Blocks.

6.6 Module Information Descriptor Block

```
typedef struct ExtI2CDeviceDesc_s{
    DBlockIdHdr_t DBHdr        ; /* 0x00 Dynamic Block Header  */
    uint8_t       DeviceAddr[2]; /* 0x03 Encoded 7/10 Bit Device Address */
    uint8_t       DeviceBus    ; /* 0x05 Device Bus       */
    uint8_t       DeviceDesc   ; /* 0x06 Device Descriptor */
}ExtI2CDeviceDesc_t;
```

CAP_C_STRUCT 5: Module Information

6.6.1 Description

This block is loosely based on the SMBIOS 2.6 standards Module Description block. The Primary use of this block is to populate the SMBIOS block type 3, if supported.

Table 6-6: Chassis Information Descriptor Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xD2	Module Descriptor
1	2	Block Length	24	Dynamic Block Size (Words)
3	1	Manufacturer	1	String 1
4	1	Product Name	2	String 2
5	1	Version	3	String 3
6	1	Serial Number	4	String 4
7	1	Asset Tag	0	String 0
8	1	Feature Flag	0x08	Replaceable
9	1	Location	0	String0
10	1	Board Type	10	Processor Memory Module
11	1	String0	“\0”	Zero Length String “Unused”
12	6	String1	“PICMG\0”	
18	11	String2	“COMExpress\0”	
29	8	String3	“1.1.0.1\0”	
37	6	String4	“SN123\0”	
43	5	String5	“COM0\0”	

6.7 LFP Device Descriptor Block

```
typedef struct LFPDataBlock_s{
    DBlockIdHdr_t DBHdr      ; /* 0x00 Dynamic Block Header  */
    uint8_t       Interface      ; /* 0x03 Display Interface */
    uint8_t       RawData[1]; /* 0x04 Display Raw Data
                               *      DisplayID
                               *      EDID
                               *      UDS
                               *      EPI
                               */
}LFPDataBlock_t;
```

CAP_C_STRUCT 6: LFP Device Descriptor

6.7.1 Description

This block is defined to allow storage of Flat panel Timing information. This is a simple wrapper and should be viewed as a virtual EPROM.

6.7.2 Display Interface

Value	Description
1	LVDS
2	SDVOB - LVDS
3	SDVOC - LVDS
4	DDI1 eDP
5	DDI2 eDP
6	DDI3 eDP

6.7.3 Raw Data

It is expected that this block is used to contain one of the following, data formats EDID, Display ID, UDS, EPI. It is outside the scope of this specification to define the content of this data, please refer to the relevant specifications.

Table 6-7: LFP Device Descriptor Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xD2	LFP Display Descriptor Block
1	2	Block Length	66	Dynamic Block Size (Words)
3	1	Interface	1	LVDS
4	128	EDID Data	0x00, 0x00, 0xFF, 0xFF ...	EDID Data

Dynamic Descriptor Blocks.

6.8 Vendor Specific Block

```
typedef struct VendBlockHdr_s{
    DBlockIdHdr_t DBHdr      ; /* 0x00 Dynamic Block Header  */
    uint8_t       VendId[2]  ; /* 0x03 Compressed ASCII PNPID  */
    /* After This Point is only
     * Suggested
     */
    //uint8_t       VendBlockId; /* 0x04 Vendor Specific Block Id  */
    //uint8_t       VendData[1]; /* 0x05 Vendor Data */
}VendBlockHdr_t;
```

CAP_C_STRUCT 7: Vendor Specific

6.8.1 Description

Due to the Nature of the Module Market it is to be expected that an EPROM may have Vendor specific blocks not only form the Carrier Board Manufacturer but also from several Module vendors also.

To Prevent “Vendor Specific block” contention, All Vendor specific blocks are distinguished by compressed ASCII form of the Vendor PNPID.

6.8.2 Vendor ID

Compressed ASCII PNPID(see chapter 8.1 page 45).

Table 6-8: Vendor Specific Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xF0	Vendor Block
1	2	Block Length	3	Dynamic Block Size (Words)
3	2	Vendor Id	0xA741	Encoded PMG
5	1	….	...	Etc

6.9 CRC16 Block

```
typedef struct CRC16ChkBlock_s{
    DBlockIdHdr_t DBHdr         ; /* 0x00 Dynamic Block Header  */
    uint8_t       CrC16[2]      ; /* 0x03 CRC16 Checksum */
}CRC16ChkBlock_t;
```

CAP_C_STRUCT 8: CRC

6.9.1 Description

This two-byte field contains the calculated 16-bit CRC-CCITT (polynomial 0x1021) for previous bytes in the EEPROM. The following algorithm and data structures (shown in C) are to be followed in calculating and checking the code. The polynomial calculated with this algorithm is x16 + x12 + x5 + 1.

The reason for Defining the CRC as a Dynamic Block rather than having it at a fixed location, are

1. Allows for dynamic length support
2. Allows for the possibility of having data/structures outside the CRC.

Table 6-9: CRC16 Block Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xF2	CRC Block
1	2	Block Length	3	Dynamic Block Size (Words)
3	2	CRC16	0xA741	CRC16 Checksum
5	1	Pack Byte	0x00	Word Alignment Pack Byte

```
uint16_t
Crc16 (
       __IN const void ptr    , /* Pointer to Buffer */
       __IN size_t     count   /* Num bytes to CRC  */
       )
{
  uint16_t crc=0;
  size_t  i;
  while (--count >= 0) {
    crc = crc ^ (int)*ptr++ << 8;
    for (i = 0; i < 8; ++i){
      if (crc & 0x8000)
        crc = crc << 1 ^ 0x1021;
      else
        crc = crc << 1;
    }
  }
  return crc;
}

/* Sample Usage Set Crc*/
CRC16ChkBlock_t CRCBlock;
uint16_t crctmp=Crc16(pointer, len);
CRCBlock.CrC16[1]=(uint8_t)(crctmp&0xFF);
CRCBlock.CrC16[0]=(uint8_t)((crctmp>>8)&0xFF);

/* Sample Usage Check Crc*/
CRC16ChkBlock_t CRCBlock;
uint16_t crctmp1=CRCBlock.CrC16[1]|(CRCBlock.CrC16[0]<<8)
uint16_t crctmp2=Crc16(pointer, len);
if (crctmp1==crctmp2){
  // Valid CRC16
}
```

C_SAMPLE 4: CRC-CCITT implementation

Dynamic Descriptor Blocks.

```
@proto_Crc16 TYPEDEF PROTO STDCALL :NEAR32 PTR :uint32_t
Crc16           PROTO            @proto_Crc16

Crc16 PROC STDCALL PUBLIC USES BX ECX ESI \
        BufPtr:NEAR32 PTR, Count:uint32_t
        mov     esi, BufPtr
        mov     ecx, Count
        xor     bx, bx
ReadLoop:
        mov     al, ds:[esi]
        xor     bh, al
        mov     al, 8
LoopByte:
        shl     bx, 1
        jnc     @F
        xor     bx, 01021h
@@:
        dec     al
        jnz     LoopByte
        inc     esi
        loopd   ReadLoop
        mov     ax, bx
Exit:
        ret
Crc16 ENDP
```

MASM_SAMPLE 2: CRC-CCITT 386 implementation

6.10 Expansion EEPROM descriptor

```
typedef struct ExtI2CDeviceDesc_s{
    DBlockIdHdr_t DBHdr        ; /* 0x00 Dynamic Block Header  */
    uint8_t       DeviceAddr[2]; /* 0x03 Encoded 7/10 Bit Device Address */
    uint8_t       DeviceBus    ; /* 0x05 Device Bus        */
    uint8_t       DeviceDesc   ; /* 0x06 Device Descriptor */
}ExtI2CDeviceDesc_t;
```

CAP_C_STRUCT 9: Additional EEPROM descriptor

6.10.1 Description

This allows for system where System information is stored in multiple distributed EEPROMs. I.E. Chassis information is stored in a separate EEPROM integrated directly in the CHASSIS.

Table 6-10: Expansion EEPROM descriptor Example

Offset	Width	Name	Value	Description
0	1	Block Id	0xF1	Expansion EEPROM Block
1	2	Block Length	4	Dynamic Block Size (Words)
3	2	Device Addr	0x00A0	Device Address
5	1	Device Bus	0x00	I2C Bus
6		Device Desc{0:3}	0x00	256 Byte Device
6		Device Desc{4}	0x00	Standard Index
6	1	Device Desc{5:7}	0x00	Reserved
7	1	Pack Byte	0x00	Word Alignment Pack Byte

7 Sample EEPROM Content

7.1 Sample Carrier Board EEPROM Content

7.1.1 COM0R20 Type 2 Carrier Board With Expansion EEPROM

Table 7-1: COM0R20 Type 2 Carrier Board With Expansion EEPROM

Offset	Width	Name	Value	Description
0	1	DontCareByte	0	Don't Care Byte
1	2	EepId	“3P”	EEPROM Marker
3	1	EeeP	0x10	EeeP Specification Version
4	1	BlkOffset	21	Absolute offset to First Block (words)
5	0	Device Desc{0:3}	0	256 Byte Device
5	0	Device Desc{4}	0	Standard Index Device
5	1	Device Desc{5:7}	1	Page Write Length 8 Bytes
6	4	Com0Id	“COM0”	Com0 Header Identifier
10	2	Vendor Id	0xA741	Encoded PMG
12	2	Device Id	0x0001	Vendor Specific Device Id
14	1	Device Flavor	0x01	Device Specific Flavor Id.
15	1	Device Rid	0x01	Device Specific Revision Id
16	1	CBType	0x02	Carrier Board Type
17	1	SpecRev	0x20	COM0 Specification Revision
18	0	UsbDesc{0:3}	8	8 USB High Speed Ports
18	0	UsbDesc{4:6}	0	0 USB Super Speed Ports
18	1	UsbDesc{7}	0	Reserved
19	0	SasDesc{0:1}	1	PORT0 SATA
19	0	SasDesc{2:3}	1	PORT1 SATA
19	0	SasDesc{4:5}	3	PORT2 SAS
19	1	SasDesc{6:7}	0	PORT3 Not Implemented
20	1	LanDesc	0x01	GBE0 Implemented GBE1 Not Implemented GBE2 Not Implemented GB1E0 Not Implemented
21	1	MiscIo1	0xFF	WAKE0 Implemented WAKE1 Implemented SUS Implemented BATLOW Implemented THRMP Implemented EBROM Implemented WDT Implemented AC97 Implemented
22	1	MiscIo2	0x01	SSC Implemented SDIO Not Implemented
23		DDIDesc{0:2}		DDI1 Not Implemented
23		DDIDesc{3:6}	0	DDI2 Not Implemented

Offset	Width	Name	Value	Description
23	1	DDIDesc{7:8}	0	DDI3 Not Implemented
24	1	Reserved0	0x00	Reserved 0
25	1	Reserved1	0x00	Reserved 1
26	1	LaneMap[0]	0x11	Gen1 x1 Lane 00 & 01
27	1	LaneMap[1]	0x99	Gen2 x1 Lane 02 & 03
28	1	LaneMap[2]	0x22	Gen1 x2 Lane 04 - 05
29	1	LaneMap[3]	0x00	Unused
30	1	LaneMap[4]	0x00	Unused
31	1	LaneMap[5]	0x00	Unused
32	1	LaneMap[6]	0x00	Unused
33	1	LaneMap[7]	0x00	Unused
34	1	LaneMap[8]	0x55	Gen1 x16 Lane 16 - 31
35	1	LaneMap[9]	0x55	Gen1 x16 Lane 16 - 31
36	1	LaneMap[10]	0x55	Gen1 x16 Lane 16 - 31
37	1	LaneMap[11]	0x55	Gen1 x16 Lane 16 - 31
38	1	LaneMap[12]	0x55	Gen1 x16 Lane 16 - 31
39	1	LaneMap[13]	0x55	Gen1 x16 Lane 16 - 31
40	1	LaneMap[14]	0x55	Gen1 x16 Lane 16 - 31
41	1	LaneMap[15]	0x55	Gen1 x16 Lane 16 - 31
42				
42	1	Block Id	0xE0	Express Card Extended Topology
43	2	Block Length	5	Relative Offset to Next Dynamic Block
45	1	Express Card Number	0x01	Express Card Slot 2
46		Com Express Port{0:4}	16	PCI Express Lane 16 “PEG Port”
46	1	Com Express Port{5:7}	1	USB Port 1
47	1	Switch Port Number[0]	1	1st port on Switch 1
48	1	Switch Port Number[1]	3	3rd port on Switch 2
49	1	Switch Port Number[2]	3	3rd port on Switch 3
50	1	Switch Port Number[3]	0x7F	End of Port List
51	1	Fill Byte	0x00	Fill Byte to reach Word Boundary
52				
52	1	Block Id	0xD1	System Descriptor
53	2	Block Length	26	Dynamic Block Size(Words)
55	1	Manufacturer	1	String 1
56	1	Product Name	2	String 2
57	1	Version	3	String 3
58	1	Serial Number	4	String 4
59	1	SKU Number	0	String 0
60	1	Family	5	String 5
61	1	Asset Tag Number	0	String0
62	1	String0	“\0”	Zero Length String “Unused”

Offset	Width	Name	Value	Description
63	6	String1	“PICMG\0”	
69	11	String2	“COMExpress\0”	
80	8	String3	“1.1.0.1\0”	
88	11	String4	“SN12345678\0”	
99	5	String5	“COM0\0”	
104				
104	1	Block Id	0xF2	Extended EEPROM Device Descriptor. See ' 7.3 Sample Expansion EEPROM Content' on page 44
105	2	Block Length	3	Dynamic Block Size(Words)
107	1	Device Addr	0xA0	Device Address
108	1	Device Bus	0x00	I2C Bus
109		Device Desc{0:3}	0	256 Byte Device
109		Device Desc{4}	0	Standard Index Device
109	1	Device Desc{5:7}	0	Reserved
110				
110	1	Block Id	0x00	Unused Space
111	2	Block Length	70	
113	137	Unused		
250				
250	1	Block Id	0xFE	CRC
251	2	Block Length	0x0000	End Of List
253	2	CRC	0xXXXX	
255	1	Pack Byte	0x00	Word Alignment Pack Byte

7.1.2 COM0R20 Type 2 Carrier Board

Table 7-2: COM0R20 Type 2 Carrier Board

Offset	Width	Name	Value	Description
0	1	DontCareByte	0	Don't Care Byte
1	2	EepId	“3P”	EEPROM Marker
3	1	EeeP	0x10	EeeP Specification Version
4	1	BlkOffset	21	Absolute offset to First Block (words)
5	4	Com0Id	“COM0”	Com0 Header Identifier
9	2	Vendor Id	0xA741	Encoded PMG
11		Device Desc{0:3}	0	256 Byte Device
11		Device Desc{4}	0	Standard Index Device
11	1	Device Desc{5:7}	1	Page Write Length 8 Bytes
12	2	Device Id	0x0001	Vendor Specific Device Id
14	1	Device Flavor	0x01	Device Specific Flavor Id.
15	1	Device Rid	0x01	Device Specific Revision Id
16	1	CBType	0x02	Carrier Board Type

Offset	Width	Name	Value	Description
17	1	SpecRev	0x20	COM0 Specification Revision
18	0	UsbDesc{0:3}	8	8 USB High Speed Ports
18	0	UsbDesc{4:6}	0	0 USB Super Speed Ports
18	1	UsbDesc{7}	0	Reserved
19	0	SasDesc{0:1}	1	PORT0 SATA
19	0	SasDesc{2:3}	1	PORT1 SATA
19	0	SasDesc{4:5}	3	PORT2 SAS
19	1	SasDesc{6:7}	0	PORT3 Not Implemented
20	1	LanDesc	0x01	GBE0 Implemented GBE1 Not Implemented GBE2 Not Implemented GB1E0 Not Implemented
21	1	Misclo1	0xFF	WAKE0 Implemented WAKE1 Implemented SUS Implemented BATLOW Implemented THRMP Implemented EBROM Implemented WDT Implemented AC97 Implemented
22	1	Misclo2	0x01	SSC Implemented SDIO Not Implemented
23		DDIDesc{0:2}		DDI1 Not Implemented
23		DDIDesc{3:6}	0	DDI2 Not Implemented
23	1	DDIDesc{7:8}	0	DDI3 Not Implemented
24	1	Reserved0	0x00	Reserved 0
25	1	Reserved1	0x00	Reserved 1
26	1	LaneMap[0]	0x11	Gen1 x1 Lane 00 & 01
27	1	LaneMap[1]	0x99	Gen2 x1 Lane 02 & 03
28	1	LaneMap[2]	0x22	Gen1 x2 Lane 04 - 05
29	1	LaneMap[3]	0x00	Unused
30	1	LaneMap[4]	0x00	Unused
31	1	LaneMap[5]	0x00	Unused
32	1	LaneMap[6]	0x00	Unused
33	1	LaneMap[7]	0x00	Unused
34	1	LaneMap[8]	0x55	Gen1 x16 Lane 16 - 31
35	1	LaneMap[9]	0x55	Gen1 x16 Lane 16 - 31
36	1	LaneMap[10]	0x55	Gen1 x16 Lane 16 - 31
37	1	LaneMap[11]	0x55	Gen1 x16 Lane 16 - 31
38	1	LaneMap[12]	0x55	Gen1 x16 Lane 16 - 31
39	1	LaneMap[13]	0x55	Gen1 x16 Lane 16 - 31
40	1	LaneMap[14]	0x55	Gen1 x16 Lane 16 - 31
41	1	LaneMap[15]	0x55	Gen1 x16 Lane 16 - 31
42				
42	1	Block Id	0xE0	Express Card Extended Topology
43	2	Block Length	8	Dynamic Block Size (Words)
45	1	Express Card Number	0x01	Express Card Slot 2

Sample EEPROM Content

Offset	Width	Name	Value	Description
46		Com Express Port{0:4}	16	PCI Express Lane 16 “PEG Port”
46	1	Com Express Port{5:7}	1	USB Port 1
47	1	Switch Port Number[0]	1	1st port on Switch 1
48		Switch Port Number[1]	3	3rd port on Switch 2
48		Switch Port Number[2]	3	3rd port on Switch 3
48	1	Switch Port Number[3]	0x7F	End of Port List
49	1	Pack Byte	0x00	Word Alignment Pack Byte
50				
50	1	Block Id	0xD1	System Descriptor
51	2	Block Length	24	Dynamic Block Size (Words)
53	1	Manufacturer	1	String 1
54	1	Product Name	2	String 2
55	1	Version	3	String 3
56	1	Serial Number	4	String 4
57	1	SKU Number	0	String 0
58	1	Family	5	String 5
59	1	Asset Tag Number	0	String0
60	1	String0	“\0”	Zero Length String “Unused”
61	6	String1	“PICMG\0”	
67	11	String2	“COMExpress\0”	
78	8	String3	“1.1.0.1\0”	
86	6	String4	“SN123\0”	
92	5	String5	“COM0\0”	
97	1	Pack Byte	0x00	Word Alignment Pack Byte
98				
98	1	Block Id	0xD3	Chassis Descriptor
99	2	Block Length	17	Dynamic Block Size (Words)
101	1	Manufacturer	1	String 1
102	1	Chassis Type		Chassis Type ENUM
103	1	Version	3	String 2
104	1	Serial Number	4	String 3
105	1	Asset Tag	0	String 0
106	1	String0	“\0”	Zero Length String “Unused”
107	6	String1	“PICMG\0”	
113	8	String2	“1.1.0.1\0”	
121	11	String3	“SN12345678\0”	
132				
132	1	Block Id	0x00	Unused Space
133	2	Block Length	59	
135	115	Unused		

Offset	Width	Name	Value	Description
250				
250	1	Block Id	0xFE	CRC
251	2	Block Length	0x0000	End Of List
253	2	CRC	0xXXXX	
255	1	Pack Byte	0x00	Word Alignment Pack Byte

551 ## *7.2 Sample COM0R20 Module EEPROM Content*

552 ### 7.2.1 Sample 1 All in CRC

553 **Table 7-3: Sample 1 All in CRC**

Offset	Width	Name	Value	Description
0	1	DontCareByte	0	Don't Care Byte
1	2	EepId	“3P”	EEPROM Marker
3	1	EeeP	0x10	EeeP Specification Version
4	1	BlkOffset	9	Absolute offset to First Block (words)
5		Device Desc{0:3}	0	256 Byte Device
5		Device Desc{4}	0	Standard Index Device
5	1	Device Desc{5:7}	0	Reserved
6	4	Com0Id	“COM0”	Com0 Header Identifier
10	2	Vendor Id	0xA741	Encoded PMG
12	2	Device Id	0x0001	Vendor Specific Device Id
14	1	Device Flavor	0x01	Device Specific Flavor Id.
15	1	Device Rid	0x01	Device Specific Revision Id
16	1	MType	0x02	Module Type
17	1	SpecRev	0x20	COM0 Specification Revision
18				
18	1	Block Id	0xD2	Module Descriptor
19	2	Block Length	24	Dynamic Block Size (Words)
21	1	Manufacturer	1	String 1
22	1	Product Name	2	String 2
23	1	Version	3	String 3
24	1	Serial Number	4	String 4
25	1	Asset Tag	0	String 0
26	1	Feature Flag	0x08	Replaceable
27	1	Location	0	String0
28	1	Board Type	10	Processor Memory Module
29	1	String0	“\0”	Zero Length String “Unused”
30	6	String1	“PICMG\0”	
36	11	String2	“COMExpress\0”	
47	8	String3	“1.1.0.1\0”	
55	6	String4	“SN123\0”	

Offset	Width	Name	Value	Description
61	5	String5	“COM0\0”	
66				
66	1	Block Id	0x00	Unused Space
67	2	Block Length	92	Dynamic Block Size (Words)
69	181	Unused		
250				
250	1	Block Id	0xFE	CRC
251	2	Block Length	0x0000	End Of List
253	2	CRC	0xXXXX	
255	1	Pack Byte	0x00	Word Alignment Pack Byte

7.2.2 Sample 2 With Data outside CRC

Table 7-4: Sample 2 With Data outside CRC

Offset	Width	Name	Value	Description
0	1	DontCareByte	0	Don't Care Byte
1	2	EepId	“3P”	EEPROM Marker
3	1	EeeP	0x10	EeeP Specification Version
4	1	BlkOffset	9	Absolute offset to First Block (words)
5		Device Desc{0:3}	0	256 Byte Device
5		Device Desc{4}	0	Standard Index Device
5	1	Device Desc{5:7}	0	Reserved
6	4	Com0Id	“COM0”	Com0 Header Identifier
10	2	Vendor Id	0xA741	Encoded PMG
12	2	Device Id	0x0001	Vendor Specific Device Id
14	1	Device Flavor	0x01	Device Specific Flavor Id.
15	1	Device Rid	0x01	Device Specific Revision Id
16	1	MType	0x02	Module Type
17	1	SpecRev	0x20	COM0 Specification Revision
18				
18	1	Block Id	0xD2	Module Descriptor
19	2	Block Length	27	Dynamic Block Size (Words)
21	1	Manufacturer	1	String 1
22	1	Product Name	2	String 2
23	1	Version	3	String 3
24	1	Serial Number	4	String 4
25	1	Asset Tag	0	String 0
26	1	Feature Flag	0x08	Replaceable
27	1	Location	0	String0
28	1	Board Type	10	Processor Memory Module
29	1	String0	“\0”	Zero Length String “Unused”
30	6	String1	“PICMG\0”	

Offset	Width	Name	Value	Description
36	11	String2	“COMExpress\0”	
47	8	String3	“1.1.0.1\0”	
55	11	String4	“SN12345678\0”	
66	5	String5	“COM0\0”	
71	1	Pack Byte	0x00	Word Alignment Pack Byte
72				
72	1	Block Id	0x00	Unused Space
73	2	Block Length	24	Dynamic Block Size (Words)
75	45	Unused		
120				
120	1	Block Id	0xFE	CRC
121	2	Block Length	3	Dynamic Block Size (Words)
123	2	CRC	0xXXXX	0
125	1	Pack Byte	0x00	Word Alignment Pack Byte
126				
126	1	Block Id	0x00	Unused Space
127	2	Block Length	10	Dynamic Block Size (Words)
129	17	Unused		
146				
146	1	Block Id	0xF0	Vendor Block
147	2	Block Length	53	Dynamic Block Size (Words)
149	2	Vendor Id	0xA741	Encoded PMG
151	101	Storage	...	Some Dynamic Storage
252				
252	1	Block Id	0x00	Unused Space
253	2	Block Length	0	End Of List
255	1	Pack Byte	0x00	Word Alignment Pack Byte

7.3 Sample Expansion EEPROM Content

7.3.1 Sample Chassis Expansion EEPROM

Table 7-5: Sample Chassis Expansion EEPROM

Offset	Width	Name	Value	Description
0	1	DontCareByte	0	Don't Care Byte
1	2	EepId	“3P”	EEPROM Marker
3	1	EeeP	0x10	EeeP Specification Version
4	1	BlkOffset	6	Absolute offset to First Block (words)
5		Device Desc{0:3}	0	256 Byte Device
5		Device Desc{4}	0	Standard Index Device
5	1	Device Desc{5:7}	0	Reserved
6	2	Vendor Id	0xA741	Encoded PMG
8	2	Device Id	0x0001	Vendor Specific Device Id
10	1	Device Flavor	0x01	Device Specific Flavor Id.
11	1	Device Rid	0x01	Device Specific Revision Id
12				
12	1	Block Id	0xD3	Chassis Descriptor
13	2	Block Length	17	Dynamic Block Size (Words)
15	1	Manufacturer	1	String 1
16	1	Chassis Type		Chassis Type ENUM
17	1	Version	3	String 2
18	1	Serial Number	4	String 3
19	1	Asset Tag	0	String 0
20	1	String0	“\0”	Zero Length String “Unused”
21	6	String1	“PICMG\0”	
27	8	String2	“1.1.0.1\0”	
35	11	String3	“SN12345678\0”	
46				
46	1	Block Id	0x00	Unused Space
47	2	Block Length	102	Dynamic Block Size (Words)
49	201	Unused		
250				
250	1	Block Id	0xFE	CRC
251	2	Block Length	0x0000	End Of List
253	2	CRC	0xXXXX	
255	1	Pack Byte	0x00	Word Alignment Pack Byte

8 Standard Data Formats

8.1 Compressed ASCII PNPID

8.1.1 Definition

Compressed ASCII is defined as 5 bits per character, "00001b" = "A" ... "11010b" = "Z".

Byte	Bits	Description
0	{7}	0
0	{6:2}	First character in compressed ASCII
0	{1:0}	Second character in compressed ASCII bits{4:3}
1	{7:5}	Second character in compressed ASCII bits{2:0}
1	{4:0}	Third character in compressed ASCII

8.1.2 Example

Hex	Interpreted
0xA741	'PMG'

8.2 EEPROM Specification Revision.

8.2.1 Definition

This is an 8Bit value specifying the specification version used to creating the EEPROM content.

Bits	Description
{7:4}	Specification Version.
{3:0}	Specification Revision.

8.2.2 Examples

Value	Decoded
0x10	1.0
0x20	2.0
0x21	2.1

9 Headers

9.1 EeeP.h

```
/*
 *<KHeader>
 *+=========================================================================
 *I              EApiDK Embedded Application Development Kit
 *+=========================================================================
 *I  $HeadURL: https://eapidk.svn.sourceforge.net/svnroot/eapidk/trunk/include/EeeP.h $
 *+=========================================================================
 *I   Copyright: Copyright (c) 2002-2009, Kontron Embedded Modules GmbH
 *I      Author: John Kearney,                  John.Kearney@kontron.com
 *I
 *I     License: All rights reserved. This program and the accompanying
 *I              materials are licensed and made available under the
 *I              terms and conditions of the BSD License which
 *I              accompanies this distribution. The full text of the
 *I              license may be found at
 *I              http://opensource.org/licenses/bsd-license.php
 *I
 *I              THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN
 *I              "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF
 *I              ANY KIND, EITHER EXPRESS OR IMPLIED.
 *I
 *I Description: Embedded EEPROM
 *I
 *+-------------------------------------------------------------------------=
 *I
 *I  File Name            : STDEEP.h
 *I  File Location        : include
 *I  Last committed       : $Revision: 21 $
 *I  Last changed by      : $Author: dethrophes $
 *I  Last changed date    : $Date: 2009-10-20 17:10:24 +0200 (Di, 20 Okt 2009) $
 *I  ID                   : $Id: EeeP.h 21 2009-10-20 15:10:24Z dethrophes $
 *I
 *+=========================================================================
 *</KHeader>
 */

/* Structures for Embedded EEPROM */
#ifndef __EEEP_H__
#define __EEEP_H__

#ifndef EEEP_UINT32_C
#  define EEEP_UINT8_C(x)   ((uint8_t)(x))
#  define EEEP_UINT16_C(x)  ((uint16_t)(x))
#  define EEEP_UINT32_C(x)  ((uint32_t)(x))
#endif

//#pragma pack(push)
#pragma pack(1)


/*
 * Detecting EeeP EEPROM
 *
 * High Level Check
 * if(!memcmp(
 *        &EeePEEP[0x01]           ,
 *        "3P"                     ,
 *        0x02
 *      )
 *   )
 * {
 *    // Found EeeP EEPROM
```

```
 * }
 *
 * Sample I2C Transfers
 *
 *  Device Address : 0xAE(0x57)
 *  Index Type     : Standard
 *    Start<0x57><W>Ack<0x00>Ack
 *    Start<0x57><R>Ack<0x00>Ack<'3'>Ack<'P'>Ack<0x10>Nak Stop
 *
 *  Device Address : 0xAE(0x57)
 *  Index Type     : Extended
 *    Start<0x57><W>Ack<0x00>Ack<0x00>Ack
 *    Start<0x57><R>Ack<0x00>Ack<'3'>Ack<'P'>Ack<0x10>Nak Stop
 *
 */



#define EEEP_VER_MASK_VER     EEEP_UINT8_C(0xF0)
#define EEEP_VER_MASK_REV     EEEP_UINT8_C(0x0F)
#define EEEP_VER_GET_VER(x)   EEEP_UINT8_C (((x)>>4)&0x0F )
#define EEEP_VER_GET_REV(x)   EEEP_UINT8_C (((x)>>0)&0x0F )
#define EEEP_VER_CREATE(Version,Revision) (\
                      EEEP_UINT8_C(\
                        (((Version )&0x0F )<<4)|\
                        (((Revision)&0x0F )<<0) \
                      )\
                    )

/* EeeP Standard Revision */
#define EEEP_VER      0
#define EEEP_REVISION 5
#define EEEP_VERSION EEEP_VER_CREATE(EEEP_VER, EEEP_REVISION)




/*
 * EEPROM Common Header
 *
 */
typedef struct EeePCmn_s{
    uint8_t     Reserved0  ; /* 0x00 Don't Care Byte
                             *      The purpose of this
                             *      Byte is to reduce the Damage
                             *      Extended Index read access is
                             *      used on a Standard Index Device
                             *
                             */
    uint8_t     EepId[2]   ; /* 0x01 3P   */
#       define  EEEP_EEPROM_MARKER    "3P"
    uint8_t     SpecRev    ; /* 0x03 EeeP Specification Revision
                             *      +=======+==================+
                             *      | Bits  | Descriptions     |
                             *      +=======+==================+
                             *      | 0 - 3 | Revision         |
                             *      +-------+------------------+
                             *      | 4 - 7 | Version          |
                             *      +=======+==================+
                             */

    uint8_t     BlkOffset  ; /* 0x04 Absolute Offset to
                             *      First Dynamic Block
                             *      in words(2 bytes)
                             */
    uint8_t     DeviceDesc ; /* 0x05 Device Descriptor
                             * +========+========================+
                             * | Bits   | Description            |
                             * +========+========================+
                             * | 0 - 3  | Size 2^(8+n) Bytes     |
                             * |        | (256 << n) Bytes       |
```

```
                                * |        |                         |
                                * |        | Addressable Bytes       |
                                * |        |   Standard Index        |
                                * |        |     8bit - 11Bit        |
                                * |        |   Extended Index        |
                                * |        |     16bit - 19Bit       |
                                * |        | 2048 KBit - 4096 MBit   |
                                * |        | 256 Bytes - 512 KBytes  |
                                * |        |                         |
                                * +--------+-------------------------+
                                * | 4      | Std/Ext Index           |
                                * +--------+-------------------------+
                                * | 5 - 7  | Page Write Length/      |
                                * |        | Alignment               |
                                * |        | 0 = 1 Byte              |
                                * |        | 1 = 8 Byte              |
                                * |        | 2 = 16 Byte             |
                                * |        | 3 = 32 Byte             |
                                * |        | 4 = 64 Byte             |
                                * |        | 5 = 128 Byte            |
                                * |        | 6 = 256 Byte            |
                                * |        | 7 = 512 Byte            |
                                * +========+=========================+
                                */
}EeePCmn_t;


/*
 * EeeP
 * Universal Device Identifier
 *
 * Standardized Compact
 * Mechanism to Uniquely
 * Identify Device Revisions
 *
 */
typedef struct UDIdEep_s{
    uint8_t     VendId[2]  ; /* 0x00 Vendor Unique PNPID         */
    uint8_t     DeviceId[2]; /* 0x02 Vendor Specific Device ID   */
    uint8_t     DeviceFlav ; /* 0x04 Device Specific Flavor ID   */
    uint8_t     RevId      ; /* 0x05 Device Specific Revision ID */
}UDIdEep_t;

/*
 * EeeP
 * Expansion EEPROM Header
 *
 */
typedef struct Exp_EEP_s{
    EeePCmn_t   EeePHdr  ; /* 0x00 EeeP Common Header */
    UDIdEep_t   DevId    ; /* 0x06 Unique Device Id   */
}Exp_EEP_t;




/*
 * Dynamic Block Common Header
 *
 */
typedef struct DBlockIdHdr_s{
    uint8_t   DBlockId       ; /* 0x00 Block Id       */
    uint8_t   DBlockLength[2]; /* 0x01 Block Length/
                                *      Offset to next Block
                                *      in words (2 Bytes)
                                */
}DBlockIdHdr_t;

/*
 *
```

```
 *
 *
 * Block Types
 *
 *
 *
 */

/*
 * Block Ids
 *
 */
#define EEEP_BLOCK_ID_UNUSED            EEEP_UINT8_C(0x00)
/*
 * 0xC0 -0xCF See Platform Specific Headers
 *
 */
#define EEEP_BLOCK_ID_VENDOR_SPECIFIC   EEEP_UINT8_C(0xF0)
#define EEEP_BLOCK_ID_EXP_EEPROM        EEEP_UINT8_C(0xF1)
/*
 * 0xE0 -0xEF See Platform Specific Headers
 *
 */
#define EEEP_BLOCK_ID_SYSTEM_DESC       EEEP_UINT8_C(0xD1)
#define EEEP_BLOCK_ID_MODULE_DESC       EEEP_UINT8_C(0xD2)
#define EEEP_BLOCK_ID_CHASSIS_DESC      EEEP_UINT8_C(0xD3)
#define EEEP_BLOCK_ID_CRC_CHK           EEEP_UINT8_C(0xF2)
#define EEEP_BLOCK_ID_IGNORE            EEEP_UINT8_C(0xFF)

#define EEEP_OFFSET_VALUE_EOL           EEEP_UINT16_C(0x0000)
#define EEEP_OFFSET_VALUE_EOL_ALT       EEEP_UINT16_C(0xFFFF)

/*
 * CRC Checksum Block
 *
 */
typedef struct CRC16ChkBlock_s{
    DBlockIdHdr_t DBHdr         ; /* 0x00 Dynamic Block Header  */
    uint8_t       CrC16[2]      ; /* 0x03 CRC16 Checksum */
}CRC16ChkBlock_t;


/*
 *      System Information
 *
 *  see http://www.dmtf.org/standards/documents/SMBIOS/DSP0134.pdf
 */

typedef struct SystemInfo_s{
    DBlockIdHdr_t DBHdr   ; /* 0x00 Dynamic Block Header  */
    uint8_t Manufacturer  ; /* 0x03 Number of ASCIIZ String */
    uint8_t ProductName   ; /* 0x04 Number of ASCIIZ String */
    uint8_t Version       ; /* 0x05 Number of ASCIIZ String */
    uint8_t SerialNumber  ; /* 0x06 Number of ASCIIZ String */
    uint8_t SKU_Number    ; /* 0x07 Number of ASCIIZ String */
    uint8_t Family        ; /* 0x08 Number of ASCIIZ String */
    uint8_t AssetTagNumber; /* 0x09 Number of ASCIIZ String */
    char StringsBlock[1] ;  /* 0x0A String Block */
}SystemInfo_t;

/*
 *      Chassis Information
 *
 *  see http://www.dmtf.org/standards/documents/SMBIOS/DSP0134.pdf
 */
typedef struct ChassisInfo_s{
    DBlockIdHdr_t DBHdr   ; /* 0x00 Dynamic Block Header  */
    uint8_t Manufacturer  ; /* 0x03 Number of ASCIIZ String */
    uint8_t ChassisType   ; /* 0x04 ENUM */
    uint8_t Version       ; /* 0x05 Number of ASCIIZ String */
    uint8_t SerialNumber  ; /* 0x05 Number of ASCIIZ String */
```

```
    uint8_t AssetTagNumber; /* 0x06 Number of ASCIIZ String */
    char  StringsBlock[1] ; /* 0x07 String Block */
}ChassisInfo_t;

/*
 *     Base Board (or Module) Information
 *
 *  see http://www.dmtf.org/standards/documents/SMBIOS/DSP0134.pdf
 */
typedef struct ModuleInfo_s{
    DBlockIdHdr_t DBHdr  ; /* 0x00 Dynamic Block Header  */
    uint8_t Manufacturer ; /* 0x03 Number of ASCIIZ String */
    uint8_t Product      ; /* 0x04 Number of ASCIIZ String */
    uint8_t Version      ; /* 0x05 Number of ASCIIZ String */
    uint8_t SerialNumber ; /* 0x06 Number of ASCIIZ String */
    uint8_t AssetTag     ; /* 0x07 Number of ASCIIZ String */
    uint8_t FeatureFlag  ; /* 0x08 A collection of
                           *      flags that identify
                           *      features of this
                           *      baseboard.
                           *      +=======+=========================+
                           *      | Bits  | Descriptions            |
                           *      +=======+=========================+
                           *      | 0     | Is Motherboard          |
                           *      +-------+-------------------------+
                           *      | 1     | Requires Daughter Board |
                           *      +-------+-------------------------+
                           *      | 2     | Removable               |
                           *      +-------+-------------------------+
                           *      | 3     | Replaceable             |
                           *      +-------+-------------------------+
                           *      | 4     | Hot Swap Capable        |
                           *      +-------+-------------------------+
                           *      | 5 - 7 | Reserved                |
                           *      +=======+=========================+
                           */
#       define SMBIOS_IS_Motherboard  EEEP_UINT8_C(1 << 0)
#       define SMBIOS_REQ_DAUGHTER    EEEP_UINT8_C(1 << 1)
#       define SMBIOS_REMOVABLE       EEEP_UINT8_C(1 << 2)
#       define SMBIOS_REPLACEABLE     EEEP_UINT8_C(1 << 3)
#       define SMBIOS_HOT_SWAP_CAP    EEEP_UINT8_C(1 << 4)
    uint8_t Location     ; /* 0x09 Number of ASCIIZ String */
    uint8_t BoardType    ; /* 0x0A SMBIOS_BoardTypes_t */
    char StringsBlock[1] ; /* 0x0B String Block */
}ModuleInfo_t;
typedef enum SMBIOS_BoardTypes_e{
  SMBIOS_BoardType_Unknown=0,
  SMBIOS_BoardType_Other,
  SMBIOS_BoardType_ServerBlade,
  SMBIOS_BoardType_ConnectivitySwitch,
  SMBIOS_BoardType_SystemManagementModule,
  SMBIOS_BoardType_ProcessorModule,
  SMBIOS_BoardType_IO_Module,
  SMBIOS_BoardType_Memory_Module,
  SMBIOS_BoardType_DaughterBoard,
  SMBIOS_BoardType_Motherboard,
  SMBIOS_BoardType_ProcessorMemory_Module,
  SMBIOS_BoardType_Interconnect_Board,
}SMBIOS_BoardTypes_t;


/*
 * Display Device Data Block
 *
 */
typedef struct LFPDataBlock_s{
    DBlockIdHdr_t DBHdr      ; /* 0x00 Dynamic Block Header  */
    uint8_t       Interface      ; /* 0x03 Display Interface */
#       define    EEEP_DISP_INT_LVDS   EEEP_UINT8_C(0x02)
#       define    EEEP_DISP_INT_SDVOB  EEEP_UINT8_C(0x03)
```

```
#       define    EEEP_DISP_INT_SDVOC  EEEP_UINT8_C(0x04)
#       define    EEEP_DISP_INT_DDI1     EEEP_UINT8_C(0x05)
#       define    EEEP_DISP_INT_DDI2     EEEP_UINT8_C(0x06)
#       define    EEEP_DISP_INT_DDI3     EEEP_UINT8_C(0x07)
    uint8_t       RawData[1]; /* 0x04 Display Raw Data
                               *      DisplayID
                               *      EDID
                               *      UDS
                               *      EPI
                               */
}LFPDataBlock_t;

/*
 * Vendor Specific Dynamic Block Header
 *
 */
typedef struct VendBlockHdr_s{
    DBlockIdHdr_t DBHdr      ; /* 0x00 Dynamic Block Header  */
    uint8_t       VendId[2]  ; /* 0x03 Compressed ASCII PNPID  */
    /* After This Point is only
     * Suggested
     */
    //uint8_t       VendBlockId; /* 0x04 Vendor Specific Block Id  */
    //uint8_t       VendData[1]; /* 0x05 Vendor Data */
}VendBlockHdr_t;


/*
 * Descriptor for Additional EEPROMS
 *
 * I.E. For Chassis/System/Base Board EEPROMs
 */
typedef struct ExtI2CDeviceDesc_s{
    DBlockIdHdr_t DBHdr        ; /* 0x00 Dynamic Block Header  */
    uint8_t       DeviceAddr[2]; /* 0x03 Encoded 7/10 Bit Device Address */
    uint8_t       DeviceBus    ; /* 0x05 Device Bus       */
#       define    EEEP_I2CBuSID_I2C        EEEP_UINT8_C(0x00)
#       define    EEEP_I2CBuSID_SMB        EEEP_UINT8_C(0x01)
#       define    EEEP_I2CBuSID_LVDS   EEEP_UINT8_C(0x02)
#       define    EEEP_I2CBuSID_DDI1     EEEP_UINT8_C(0x03)
#       define    EEEP_I2CBuSID_DDI2     EEEP_UINT8_C(0x04)
#       define    EEEP_I2CBuSID_DDI3     EEEP_UINT8_C(0x05)
    uint8_t       DeviceDesc   ; /* 0x06 Device Descriptor
                                 * +========+=======================+
                                 * | Bits   | Description           |
                                 * +========+=======================+
                                 * | 0 - 3  | Size 2^(8+n) Bytes    |
                                 * |        | (256 << n) Bytes      |
                                 * |        |                       |
                                 * |        | Addressable Bytes     |
                                 * |        |   Standard Index      |
                                 * |        |     8bit - 11Bit      |
                                 * |        |   Extended Index      |
                                 * |        |     16bit - 19Bit     |
                                 * |        | 2048 KBit - 4096 MBit |
                                 * |        | 256 Bytes - 512 KBytes|
                                 * |        |                       |
                                 * +--------+-----------------------+
                                 * | 4      | Std/Ext Index         |
                                 * +--------+-----------------------+
                                 * | 5 - 7  | Reserved              |
                                 * +========+=======================+
                                 */
#   define EEEP_EXT_INDX EEEP_UINT8_C(1<<4)
}ExtI2CDeviceDesc_t;


/*
 * Platform Specific Headers
 *
 */
```

```
#include "COM0EEP.h"


#pragma pack(pop)   // n = 2 , stack popped

/*
 * CPU Independent Multi Byte
 * Big Endian Memory Access
 */
void
EeeP_Set16BitValue_BE(
    uint8_t *pBuffer,
    uint16_t Value
     )
{
  pBuffer[1]=(Value   )&0xFF;
  pBuffer[0]=(Value>>8)&0xFF;
}
uint16_t
EeeP_Get16BitValue_BE(
    uint8_t *pBuffer
     )
{
  return (pBuffer[1]    ) |
         (pBuffer[0]<< 8) ;
}
EeeP_Set32BitValue_BE(
    uint8_t *pBuffer,
    uint32_t Value
     )
{
  pBuffer[3]=(Value    )&0xFF;
  pBuffer[2]=(Value>> 8)&0xFF;
  pBuffer[1]=(Value>>16)&0xFF;
  pBuffer[0]=(Value>>24)&0xFF;
}
uint32_t
EeeP_Get32BitValue_BE(
    uint8_t *pBuffer
     )
{
  return (pBuffer[3]    ) |
         (pBuffer[2]<< 8) |
         (pBuffer[1]<<16) |
         (pBuffer[0]<<24) ;
}
/*
 * CPU Independent Multi Byte
 * Little Endian Memory Access
 */
void
EeeP_Set16BitValue_LE(
    uint8_t *pBuffer,
    uint16_t Value
     )
{
  pBuffer[0]=(Value   )&0xFF;
  pBuffer[1]=(Value>>8)&0xFF;
}
uint16_t
EeeP_Get16BitValue_LE(
    uint8_t *pBuffer
     )
{
  return (pBuffer[0]    ) |
         (pBuffer[1]<< 8) ;
}
EeeP_Set32BitValue_LE(
    uint8_t *pBuffer,
    uint32_t Value
```

```
1058          )
1059      {
1060        pBuffer[0]=(Value    )&0xFF;
1061        pBuffer[1]=(Value>> 8)&0xFF;
1062        pBuffer[2]=(Value>>16)&0xFF;
1063        pBuffer[3]=(Value>>24)&0xFF;
1064      }
1065      uint32_t
1066      EeeP_Get32BitValue_LE(
1067          uint8_t *pBuffer
1068           )
1069      {
1070        return (pBuffer[0]    ) |
1071               (pBuffer[1]<< 8) |
1072               (pBuffer[2]<<16) |
1073               (pBuffer[3]<<24) ;
1074      }
1075
1076
1077      #endif /* __EEEP_H__ */
1078
```

9.2 COM0EEP.h

```
/*
 *<KHeader>
 *+=========================================================================
 *I               EApiDK Embedded Application Development Kit
 *+=========================================================================
 *I  $HeadURL: https://eapidk.svn.sourceforge.net/svnroot/eapidk/trunk/include/COM0EEP.h $
 *+=========================================================================
 *I   Copyright: Copyright (c) 2002-2009, Kontron Embedded Modules GmbH
 *I      Author: John Kearney,                  John.Kearney@kontron.com
 *I
 *I     License: All rights reserved. This program and the accompanying
 *I              materials are licensed and made available under the
 *I              terms and conditions of the BSD License which
 *I              accompanies this distribution. The full text of the
 *I              license may be found at
 *I              http://opensource.org/licenses/bsd-license.php
 *I
 *I              THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN
 *I              "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF
 *I              ANY KIND, EITHER EXPRESS OR IMPLIED.
 *I
 *I Description: COM0 R2.0 Specific Structures and Data
 *I
 *+-------------------------------------------------------------------------=
 *I
 *I  File Name            : COM0EEP.h
 *I  File Location        : include
 *I  Last committed       : $Revision: 20 $
 *I  Last changed by      : $Author: dethrophes $
 *I  Last changed date    : $Date: 2009-10-16 16:54:23 +0200 (Fr, 16 Okt 2009) $
 *I  ID                   : $Id: COM0EEP.h 20 2009-10-16 14:54:23Z dethrophes $
 *I
 *+=========================================================================
 *</KHeader>
 */

/* Structures for COM0 STDEEP */
#ifndef __COM0EEP_H__
#define __COM0EEP_H__

/*
 * Detecting COM0 R1.0 EEPROM
 *
 * High Level Check
 * if(!memcmp(
 *        &COM0EEP[0xE0]           ,
 *        "COMExpressConfig"       ,
 *        0x10
 *      )
 *   )
 * {
 *    // Found COM0R10 EEPROM
 * }
 *
 * Sample I2C Transfer
 *  Device Address : 0xAE(0x57)
 *  Index Type     : Standard
 *    Start<0x57><W>Ack<0xE0>Ack
 *    Start<0x57><R>Ack<'C'>Ack<'O'>Ack<'M'>Ack<'E'>Ack<'x'>Ack<'p'>Ack
 *                    <'r'>Ack<'e'>Ack<'s'>Ack<'s'>Ack<'C'>Ack<'o'>Ack
 *                    <'n'>Ack<'f'>Ack<'i'>Ack<'g'>Nak
Stop
 *
 */

/* COM0 R2.0 Standard Revision */
#define COM0R20_VER      2
```

```
#define COM0R20_REVISION 0
#define COM0R20_VERSION EEEP_VER_CREATE(COM0R20_VER, COM0R20_REVISION)


/*
 * COM R2.0
 * Carrier Board EEPROM Header
 *
 */
typedef struct COM0R20_CB_s{
    EeePCmn_t   EeePHdr  ; /* 0x00 EeeP Common Header */
    uint8_t     GenId[4] ; /* 0x06 "COM0"   */
    UDIdEep_t   DevId    ; /* 0x0C Unique Device Id   */
    uint8_t     CBType   ; /* 0x10 Carrier Board Type     */
    uint8_t     SpecRev  ; /* 0x11 COM0 Specification Revision
                            *      +=======+==================+
                            *      | Bits  | Descriptions     |
                            *      +=======+==================+
                            *      | 0 - 3 | Revision         |
                            *      +-------+------------------+
                            *      | 4 - 7 | Version          |
                            *      +=======+==================+
                            */
    uint8_t     UsbDesc  ; /* 0x12 USB Descriptor Byte
                            *
                            *      +=======+================================+
                            *      | Bits  | Descriptions                   |
                            *      +=======+================================+
                            *      | 0 - 3 | Count of High Speed Ports      |
                            *      +-------+--------------------------------+
                            *      | 4 - 6 | Count of Super Speed Ports     |
                            *      +-------+--------------------------------+
                            *      | 7     | Reserved                       |
                            *      +=======+================================+
                            */
#       define COM0R20_USB3_PCNT_MASK        EEEP_UINT8_C(0x07)
#       define COM0R20_USB3_PCNT_OFFSET      EEEP_UINT8_C(0x04)
#       define COM0R20_USB_PCNT_MASK         EEEP_UINT8_C(0x0F)
#       define COM0R20_USB_PCNT_OFFSET       EEEP_UINT8_C(0x00)

    uint8_t     SasDesc;   /* 0x13 LAN Descriptor Byte
                            *
                            *      +=======+=====+=======+====================+
                            *      | SAS/  | Bit | Value | Meaning            |
                            *      | SATA  |     |       |                    |
                            *      | Port  |     |       |                    |
                            *      +=======+=====+=======+====================+
                            *      | 0     | 0   | 0     | Not Implemented    |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | Implemented        |
                            *      |       +-----+-------+--------------------+
                            *      |       | 1   | 0     | SATA Device        |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | SAS  Device        |
                            *      +-------+-----+-------+--------------------+
                            *      | 1     | 2   | 0     | Not Implemented    |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | Implemented        |
                            *      |       +-----+-------+--------------------+
                            *      |       | 3   | 0     | SATA Device        |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | SAS  Device        |
                            *      +-------+-----+-------+--------------------+
                            *      | 2     | 4   | 0     | Not Implemented    |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | Implemented        |
                            *      |       +-----+-------+--------------------+
                            *      |       | 5   | 0     | SATA Device        |
                            *      |       |     +-------+--------------------+
                            *      |       |     | 1     | SAS  Device        |
                            *      +-------+-----+-------+--------------------+
```

```
                                *       | 3     | 6   | 0     | Not Implemented   |
                                *       |       |     +-------+-------------------+
                                *       |       |     | 1     | Implemented       |
                                *       |       +-----+-------+-------------------+
                                *       |       | 7   | 0     | SATA Device       |
                                *       |       |     +-------+-------------------+
                                *       |       |     | 1     | SAS  Device       |
                                *       +-------+-----+-------+-------------------+
                                */

#       define COM0R20_SAS_CONNECTOR_PRESENT EEEP_UINT8_C(1<<0)
#       define COM0R20_SATA_SAS_DEVICE       EEEP_UINT8_C(1<<1)

#       define COM0R20_SAS_CHANNEL_0         EEEP_UINT8_C(0)
#       define COM0R20_SAS_CHANNEL_1         EEEP_UINT8_C(2)
#       define COM0R20_SAS_CHANNEL_2         EEEP_UINT8_C(4)
#       define COM0R20_SAS_CHANNEL_3         EEEP_UINT8_C(6)

    uint8_t     LanDesc;    /* 0x14 LAN Descriptor Byte
                             *
                             *      +=======+=======+========================+
                             *      | Bits  | Value | Meaning                |
                             *      +=======+=======+========================+
                             *      | 0     | 0     | GBE0 Not Implemented   |
                             *      |       +-------+------------------------+
                             *      |       | 1     | GBE0 Implemented       |
                             *      +-------+-------+------------------------+
                             *      | 1     | 0     | GBE1 Not Implemented   |
                             *      |       +-------+------------------------+
                             *      |       | 1     | GBE1 Implemented       |
                             *      +-------+-------+------------------------+
                             *      | 2     | 0     | GBE2 Not Implemented   |
                             *      |       +-------+------------------------+
                             *      |       | 1     | GBE2 Implemented       |
                             *      +-------+-------+------------------------+
                             *      | 4 - 7 | Reserved                       |
                             *      +=======+================================+
                             */
#       define COM0R20_GBE0_PRESENT          EEEP_UINT8_C(1<<0)
#       define COM0R20_GBE1_PRESENT          EEEP_UINT8_C(1<<1)
#       define COM0R20_GBE2_PRESENT          EEEP_UINT8_C(1<<2)
#       define COM0R20_GB1E0_PRESENT         EEEP_UINT8_C(1<<3)

    uint8_t     MiscIo1;    /* 0x15 Miscellaneous I/O Descriptor Byte 1
                             *
                             *      +=======+=======+========================+
                             *      | Bits  | Value | Meaning                |
                             *      +=======+=======+========================+
                             *      | 0     | 0     | WAKE0 Not Implemented  |
                             *      |       +-------+------------------------+
                             *      |       | 1     | WAKE0 Implemented      |
                             *      +-------+-------+------------------------+
                             *      | 1     | 0     | WAKE1 Not Implemented  |
                             *      |       +-------+------------------------+
                             *      |       | 1     | WAKE1 Implemented      |
                             *      +-------+-------+------------------------+
                             *      | 2     | 0     | SUS Not Implemented    |
                             *      |       +-------+------------------------+
                             *      |       | 1     | SUS Implemented        |
                             *      +-------+-------+------------------------+
                             *      | 3     | 0     | BATLOW Not Implemented |
                             *      |       +-------+------------------------+
                             *      |       | 1     | BATLOW Implemented     |
                             *      +-------+-------+------------------------+
                             *      | 4     | 0     | THRMP Not Implemented  |
                             *      |       +-------+------------------------+
                             *      |       | 1     | THRMP Implemented      |
                             *      +-------+-------+------------------------+
                             *      | 5     | 0     | EBROM Not Implemented  |
                             *      |       +-------+------------------------+
```

```
                                *       |       | 1     | EBROM Implemented      |
                                *       +-------+-------+------------------------+
                                *       | 6     | 0     | WDT Not Implemented    |
                                *       |       +-------+------------------------+
                                *       |       | 1     | WDT Implemented        |
                                *       +-------+-------+------------------------+
                                *       | 7     | 0     | AC97 Not Implemented   |
                                *       |       +-------+------------------------+
                                *       |       | 1     | AC97 Implemented       |
                                *       +-------+-------+------------------------+
                                */
#       define COM0R20_WAKE0_PRESENT         EEEP_UINT8_C(1<<0)
#       define COM0R20_WAKE1_PRESENT         EEEP_UINT8_C(1<<1)
#       define COM0R20_SUS_PRESENT           EEEP_UINT8_C(1<<2)
#       define COM0R20_BATLOW_PRESENT        EEEP_UINT8_C(1<<3)
#       define COM0R20_THRMP_PRESENT         EEEP_UINT8_C(1<<4)
#       define COM0R20_EBROM_PRESENT         EEEP_UINT8_C(1<<5)
#       define COM0R20_WDT_PRESENT           EEEP_UINT8_C(1<<6)
#       define COM0R20_AC97_PRESENT          EEEP_UINT8_C(1<<7)

    uint8_t     MiscIo2;    /* 0x16 Miscellaneous I/O Descriptor Byte 2
                                *
                                *       +=======+=======+========================+
                                *       | Bits  | Value | Meaning                |
                                *       +=======+=======+========================+
                                *       | 0     | 0     | SSC Not Implemented    |
                                *       |       +-------+------------------------+
                                *       |       | 1     | SSC Implemented        |
                                *       +-------+-------+------------------------+
                                *       | 1     | 0     | SDIO Not Implemented   |
                                *       |       +-------+------------------------+
                                *       |       | 1     | SDIO Implemented       |
                                *       +-------+-------+------------------------+
                                *       | 2 - 7 | Reserved                       |
                                *       +=======+================================+
                                */
#       define COM0R20_SSC_PRESENT           EEEP_UINT8_C(1<<0)
#       define COM0R20_SDIO_PRESENT          EEEP_UINT8_C(1<<1)
    uint8_t     DDIDesc;    /* 0x17 Digital Display Interface Descriptor Byte
                                *
                                *       +=======+======+=======+==================+
                                *       | DDI   | Bits | Value | Meaning          |
                                *       | Port  |      |       |                  |
                                *       +=======+======+=======+==================+
                                *       | 1     | 0-2  | 0     | Not Implemented  |
                                *       |       |      +-------+------------------+
                                *       |       |      | 1     | eDisplay Port    |
                                *       |       |      +-------+------------------+
                                *       |       |      | 2     | Display Port     |
                                *       |       |      +-------+------------------+
                                *       |       |      | 3     | HDMI/DVI         |
                                *       |       |      +-------+------------------+
                                *       |       |      | 4     | SDVO             |
                                *       |       |      +-------+------------------+
                                *       |       |      | 5 - 7 | Reserved         |
                                *       +-------+------+-------+------------------+
                                *       | 2     | 3-4  | 0     | Not Implemented  |
                                *       |       |      +-------+------------------+
                                *       |       |      | 1     | eDisplay Port    |
                                *       |       |      +-------+------------------+
                                *       |       |      | 2     | Display Port     |
                                *       |       |      +-------+------------------+
                                *       |       |      | 3     | HDMI/DVI         |
                                *       +-------+------+-------+------------------+
                                *       |       | 5    | Reserved                 |
                                *       +-------+------+-------+------------------+
                                *       | 3     | 6-7  | 0     | Not Implemented  |
                                *       |       |      +-------+------------------+
                                *       |       |      | 1     | eDisplay Port    |
                                *       |       |      +-------+------------------+
                                *       |       |      | 2     | Display Port     |
```

```
                            *      |       |      +-------+------------------+
                            *      |       |      | 3     | HDMI/DVI         |
                            *      +-------+------+-------+------------------+
                            */
#       define COM0R20_DDI_NOT_USED          EEEP_UINT8_C(0x0)
#       define COM0R20_DDI_eDispPort         EEEP_UINT8_C(0x1)
#       define COM0R20_DDI_DispPort          EEEP_UINT8_C(0x2)
#       define COM0R20_DDI_HDMI              EEEP_UINT8_C(0x3)
#       define COM0R20_DDI_SDVO              EEEP_UINT8_C(0x4)
#       define COM0R20_DDI_BITMASK           EEEP_UINT8_C(0x3)

#       define COM0R20_DDI1_OFFSET           EEEP_UINT8_C(0x0)
#       define COM0R20_DDI2_OFFSET           EEEP_UINT8_C(0x3)
#       define COM0R20_DDI3_OFFSET           EEEP_UINT8_C(0x6)
     uint8_t     Reserved0; /* 0x18 Reserved */
     uint8_t     Reserved1; /* 0x19 Reserved */
#       define COM0R20_VGA_PRESENT           EEEP_UINT8_C(1<<4)
#       define COM0R20_LVDSCB_PRESENT        EEEP_UINT8_C(1<<3)
#       define COM0R20_LVDSCA_PRESENT        EEEP_UINT8_C(1<<2)
#       define COM0R20_SDVOCC_PRESENT        EEEP_UINT8_C(1<<1)
#       define COM0R20_SDVOCB_PRESENT        EEEP_UINT8_C(1<<0)

    uint8_t    LaneMap[16];/* 0x1A Lane Information
                            *      +=======+=======+======================+
                            *      | Bits  | Value | Meaning              |
                            *      +=======+=======+======================+
                            *      | 0 - 2 |  0    | Not Implemented      |
                            *      |       +-------+----------------------+
                            *      |       |  1    |  x1 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  2    |  x2 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  3    |  x4 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  4    |  x8 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  5    | x16 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  6    | x32 Link Width       |
                            *      |       +-------+----------------------+
                            *      |       |  7    | Reserved             |
                            *      +-------+-------+----------------------+
                            *      | 3     |  0    | Gen 1                |
                            *      |       +-------+----------------------+
                            *      |       |  1    | Gen 2                |
                            *      +-------+-------+----------------------+
                            */
#       define COM0R20_PCIEx_LANE_WIDTH_MASK EEEP_UINT8_C(0x07)
#       define COM0R20_PCIEx_GEN2            EEEP_UINT8_C(1<<3)
}COM0R20_CB_t;
/*
 * Pci Express Lane Mapping
 *
 * Example 1
 * Standard Type 2
 * 6 x1
 * 1 x16
 * +======+=======+==================+
 * | Byte | Value | Description      |
 * +======+=======+==================+
 * | 00   | 11    | x1 Lane 00 & 01  |
 * | 01   | 11    | x1 Lane 02 & 03  |
 * | 02   | 11    | x1 Lane 04 & 05  |
 * | 03   | 00    | Unused           |
 * | 04   | 00    | Unused           |
 * | 05   | 00    | Unused           |
 * | 06   | 00    | Unused           |
 * | 07   | 00    | Unused           |
 * | 08   | 55    | x16 Lane 16 - 31 |
 * | 09   | 55    | x16 Lane 16 - 31 |
```

```
 * | 10   | 55    | x16 Lane 16 - 31 |
 * | 11   | 55    | x16 Lane 16 - 31 |
 * | 12   | 55    | x16 Lane 16 - 31 |
 * | 13   | 55    | x16 Lane 16 - 31 |
 * | 12   | 55    | x16 Lane 16 - 31 |
 * | 14   | 55    | x16 Lane 16 - 31 |
 * | 15   | 55    | x16 Lane 16 - 31 |
 * +======+=======+==================+
 *
 * Example 2
 * Type 2
 * 1 x4
 * 2 x1
 * 1 x16
 * +======+=======+==================+
 * | Byte | Value | Description      |
 * +======+=======+==================+
 * | 00   | 33    | x4 Lane 00 - 03  |
 * | 01   | 33    | x4 Lane 00 - 03  |
 * | 02   | 11    | x1 Lane 04 & 05  |
 * | 03   | 00    | Unused           |
 * | 04   | 00    | Unused           |
 * | 05   | 00    | Unused           |
 * | 06   | 00    | Unused           |
 * | 07   | 00    | Unused           |
 * | 08   | 55    | x16 Lane 16 - 31 |
 * | 09   | 55    | x16 Lane 16 - 31 |
 * | 10   | 55    | x16 Lane 16 - 31 |
 * | 11   | 55    | x16 Lane 16 - 31 |
 * | 12   | 55    | x16 Lane 16 - 31 |
 * | 13   | 55    | x16 Lane 16 - 31 |
 * | 12   | 55    | x16 Lane 16 - 31 |
 * | 14   | 55    | x16 Lane 16 - 31 |
 * | 15   | 55    | x16 Lane 16 - 31 |
 * +======+=======+==================+
 *
 * Example 3
 * Type 2
 * 2 x2
 * 2 x1
 * 4 x4
 * +======+=======+==================+
 * | Byte | Value | Description      |
 * +======+=======+==================+
 * | 00   | 22    | x2 Lane 00 - 01  |
 * | 01   | 11    | x1 Lane 02 & 03  |
 * | 02   | 00    | Unused           |
 * | 03   | 00    | Unused           |
 * | 04   | 00    | Unused           |
 * | 05   | 00    | Unused           |
 * | 06   | 00    | Unused           |
 * | 07   | 00    | Unused           |
 * | 08   | 33    | x4  Lane 16 - 19 |
 * | 09   | 33    | x4  Lane 16 - 19 |
 * | 10   | 33    | x4  Lane 20 - 23 |
 * | 11   | 33    | x4  Lane 20 - 23 |
 * | 12   | 33    | x4  Lane 24 - 27 |
 * | 13   | 33    | x4  Lane 24 - 27 |
 * | 14   | 33    | x4  Lane 28 - 31 |
 * | 15   | 33    | x4  Lane 28 - 31 |
 * +======+=======+==================+
 *
 */

/*
 * COM R2.0
 * Module EEPROM Header
 *
 */
typedef struct COM0R20_M_s{
    EeePCmn_t   EeePHdr  ; /* 0x00 EeeP Common Header */
```

```
1501        uint8_t     GenId[4] ; /* 0x06 "COM0"   */
1502        UDIdEep_t   DevId    ; /* 0x0A Unique Device Id   */
1503        uint8_t     MType    ; /* 0x10 Module Type        */
1504        uint8_t     SpecRev  ; /* 0x11 COM0 Specification Revision
1505                               *      +=======+==================+
1506                               *      | Bits  | Descriptions     |
1507                               *      +=======+==================+
1508                               *      | 0 - 3 | Revision         |
1509                               *      +-------+------------------+
1510                               *      | 4 - 7 | Version          |
1511                               *      +=======+==================+
1512                               */
1513       /*
1514        * Module Fixed Offset Information
1515        *
1516        */
1517    }COM0R20_M_t;
1518
1519    #define COM0R20_BLOCK_ID_EXP_CARD_DESC     EEEP_UINT8_C(0xE0)
1520
1521
1522    /*
1523     * Express Card Slot Descriptor
1524     *
1525     */
1526    typedef struct ExpCardBlock_s{
1527        DBlockIdHdr_t  DBHdr        ; /* 0x00 Dynamic Block Header */
1528        uint8_t ExpressCardNumber   ; /* 0x03 0 - 1 */
1529        uint8_t ComExpressPort      ; /* 0x04
1530                                      *      +=======+==================+
1531                                      *      | Bits  | Descriptions     |
1532                                      *      +=======+==================+
1533                                      *      | 0 - 4 | PCI Express Port |
1534                                      *      |       | Mapping.         |
1535                                      *      |       | Port Starting    |
1536                                      *      |       | Lane.            |
1537                                      *      |       | 0 - 31           |
1538                                      *      +-------+------------------+
1539                                      *      | 5 - 7 | USB Port Mapping |
1540                                      *      |       | 0 - 7            |
1541                                      *      +=======+==================+
1542                                      */
1543        uint8_t SwitchPortNumber[1] ; /* 0x05 An Array of
1544                                      *      Port/Device
1545                                      *      numbers Terminated
1546                                      *      with 0x7F
1547                                      */
1548    #       define COM0R20_EXPCARD_MAP_EOL EEEP_UINT8_C(0x7F)
1549    }ExpCardBlock_t;
1550
1551
1552
1553
1554    #endif /* __COM0EEP_H__ */
1555
```

10 Revision History

Revision	Date	Notes
D0.00	Jul 5, 09	Initial Version
D0.01	Jul 10, 09	Improved Format
D0.02	Jul 16, 09	Improved Wording
D0.03	Oct.01.09	Reworked in accordance with Feedback.
D0.04	Oct.05.09	Improved & Expanded Sample Contents.
D0.05	Oct.05.09	Improved for Architecture independent Big-endian Access. Added More Block definitions. Added Device Flavor ID, to allow for stuffing options.
D0.06	Oct.14.09	Added Line Numbers. Improved Tables. Other minor improvements.
D0.07	Oct.16.09	Cleaned up format etc, to look more PICMG. Started Calling EeeP Renamed Document Changed Block Sizes to Words
D0.08	Oct.20.09	General Cleanup