12/17/01
Rev 0.1

PowerQUICC II

WITH

PCI Local Bus

MPC8266ADS - PCI

User's Manual

Board Rev. PROTOTYPE-B

MOTOROLA
Semiconductor Products Sector

Digital DNA from Motorola

**For More Information On This Product,
Go to: www.freescale.com**

Table of Contents

Section 1
General Information

1.1 Introduction 1
1.2 Definitions, Acronyms, and Abbreviations 2
1.3 Related Documentation 2
1.4 Specifications 3
1.5 MPC8266ADS-PCI Features 3

Section 2
Hardware Preparation and Installation

2.1 Introduction 7
2.2 Unpacking Instructions 7
2.3 Hardware Preparation 7
2.3.1 Setting VDDL Level Range - JP5 9
2.3.2 Setting VDDL Supply Voltage Level 9
2.3.3 Setting MODCK(1:3) for PLLs Multiplication Factor - SW4 (#6 - #8) 10
2.3.4 Setting Hard - Reset Configuration Source - JP3 11
2.3.5 Setting Boot Source 12
2.3.6 Setting MODCKH(0:3) - for PLLs Multiplication Factors 13
2.3.7 Setting PCI_MODCK - for PCI Bus Clock 13
2.3.8 Setting PCI_ARBITER - for PCI Mode Enabled 13
2.3.9 Setting PCI_DLL - for PCI Mode Enabled 13
2.3.10 MPC8260 JTAG's TDI Source Selection - JP1 13
2.3.11 SDRAM DIMM I2C Slave Address Selection - SW2 14
2.3.12 Setting ATX Chassis Main AC Power 15
2.4 Installation Instructions 16
2.4.1 Host Controlled Operation 16
2.4.2 Stand Alone Operation 16
2.4.3 COP/JTAG Connector - P3 17
2.4.4 Terminal to MPC8266ADS-PCI RS-232 Connection 17
2.4.5 10/100-Base-T Ethernet Port Connection 18
2.4.6 Memory Installation 18
2.4.6.1 Flash Memory SIMM Installation 18
2.4.6.2 SDRAM DIMM Installation 19

Section 3
Operating Instructions

3.1 Introduction 21
3.2 Controls and Indicators 21
3.2.1 Power-On RESET Switch - S1 21
3.2.2 ABORT Switch - S2 21
3.2.3 SOFT RESET Switch - S3 21
3.2.4 HARD RESET - Switches - S2 & S3 21
3.2.5 SW3 - Reset Configuration Switch 22
3.2.6 SW2 - SDRAM DIMM Configuration Memory I2C Slave Address Switch . . . 22
3.2.7 SW1 - Software Options Switch 22
3.2.8 JP5 - VDDL Voltage Level Range Selection 22
3.2.9 JP6 - IDDL Measurement 22
3.2.10 JP2 - Thermal Sense Connector 22
3.2.11 JP7 - Optional Ventilator Supply 23
3.2.12 JP1 - COP/JTAG TDI Source Selection 23
3.2.13 JP4- IDDH Measurement 23
3.2.14 JP11 - VPP Source Selector 24
3.2.15 GND Bridges 24
3.2.16 ATM TX Indicator - LD1 24
3.2.17 Fast Ethernet Indicator - LD2 24
3.2.18 Fast Ethernet CLSN Indicator - LD3 24
3.2.19 Ethernet LINK Indicator - LD4 25
3.2.20 Ethernet TX Indicator - LD5 25
3.2.21 12V Indicator - LD6 25
3.2.22 ATM RX Indicator - LD7 25
3.2.23 Ethernet RX Indicator - LD8 25
3.2.24 5V Indicator - LD9 25
3.2.25 -12V Indicator - LD10 25
3.2.26 3.3V Indicator - LD11 25
3.2.27 RUN Indicator - LD12 25
3.2.28 ATM ON - LD13 25
3.2.29 Fast Ethernet Port Initially Enabled - LD14 25
3.2.30 RS232 Port 1 ON - LD15 26
3.2.31 RS232 Port 2 ON - LD16 26
3.2.32 VDDL Indication - LD17 26
3.3 Memory Map 26
3.4 MPC8266 Register Programming 30
3.4.1 System Initializations 31
3.4.2 Memory Controller Registers Programming 32

Section 4
Functional Description

4.1 Reset & Reset - Configuration . . . 39
4.1.1 Power - On Reset . . . 39
4.1.1.1 Power - On Reset Configuration . . . 39
4.1.2 Hard Reset . . . 40
4.1.2.1 COP/JTAG Port Hard - Reset . . . 40
4.1.2.2 Manual Hard Reset . . . 40
4.1.2.3 Internal Sources Hard - Reset . . . 41
4.1.2.4 Hard Reset Configuration . . . 41
4.1.3 Soft Reset . . . 46
4.1.3.1 COP/JTAG Port Soft - Reset . . . 46
4.1.3.2 Manual Soft Reset . . . 46
4.1.3.3 Internal Sources Soft - Reset . . . 47
4.1.4 PCI Bus Reset . . . 47
4.2 Local Interrupter . . . 47
4.2.1 ABORT Interrupt . . . 47
4.2.2 ATM UNI Interrupt . . . 47
4.2.3 Fast Ethernet PHY Interrupt . . . 48
4.2.4 PCI Interrupt . . . 48
4.3 Clock Generator . . . 51
4.3.1 MPC8266 Clock . . . 51
4.3.2 PCI Clock . . . 52
4.4 Bus Configuration . . . 52
4.4.1 Single MPC8266 Mode . . . 52
4.4.2 60X Bus Mode . . . 52
4.5 Buffering . . . 53
4.6 Chip - Select Generator . . . 53
4.7 Synchronous Dram DIMM (60X Bus) . . . 54
4.7.1 SDRAM Programming . . . 57
4.7.2 SDRAM Refresh . . . 57
4.7.3 L2-Cache Support Influence On SDRAM Design . . . 58
4.7.4 SDRAM DIMM Configuration Information . . . 58
4.8 Flash Memory SIMM . . . 58
4.8.1 Flash Programming Voltage . . . 60
4.8.2 Flash and L2Cache . . . 60
4.9 E2PROM Memory . . . 61
4.10 PCI Bus . . . 62
4.11 L2-CACHE Support . . . 63
4.11.1 L2 Cache Configuration & Control . . . 64
4.12 Communication Ports . . . 65
4.12.1 ATM Port . . . 65
4.12.2 100/10 Base - T Port . . . 65
4.12.2.1 LXT970 Control . . . 66

4.12.3 RS232 Ports 66
4.12.3.1 RS-232 Ports' Signal Description 67
4.13 Board Control & Status Register - BCSR 67
4.13.1 BCSR0 - Board Control - Status Register 0 68
4.13.2 BCSR1 - Board Control - Status Register 1 69
4.13.3 BCSR2 - Board Control - Status Register - 2 70
4.13.4 BCSR4 - Board Control - Status Register 4 74
4.13.5 BCSR3 and BCSR5- Board Control - Status Register 3 & 5 74
4.13.6 BCSR6 - Board Control / Status Register 6 75
4.13.7 BCSR7 - Board Control / Status Register 7 75
4.14 COP/JTAG Port 76
4.14.1 Fast Download Support 78
4.14.1.1 JTAG TAP Controller 80
4.14.1.2 JTAG Instruction Shift Register - JISR 81
4.14.1.3 JTAG Instruction Register - JIR 82
4.14.1.4 Data Shift Register 82
4.14.1.5 Download Control and Status Register 83
4.14.1.6 Bypass Register 83
4.14.1.7 JTAG Machine Bypass 83
4.14.1.8 Fast Download Operation 83
4.14.1.9 JTAG Generated Power-On Reset 84
4.15 Switches, Jumpers and Indicators 84

Section 5
Memory Map and Initialization

5.1 Memory Map 86
5.2 MPC8266 Register Programming 90
5.2.1 System Initializations 91
5.2.2 Memory Controller Registers Programming 92

Section 6
Physical Properties

6.1 Power Supply 98
6.1.1 5V Rail 100
6.1.2 3.3V Rail 100
6.1.3 5V Stand By Rail 100
6.1.4 VDDH Rail 100
6.1.5 VDDL Bus 100
6.1.6 12V Rail 100
6.1.7 -12V Rail 100
6.2 Connectors 101
6.2.1 ATX Power Connector 101

6.2.2 Fast Ethernet Port Connector 101
6.2.3 ATM 155 Port Connection 101
6.2.4 RS232 PortS Connector 101
6.2.5 CPM Expansion Connector 101
6.2.6 COP/JTAG Port Connector 101
6.2.7 Logic Analyzer Connectors 102
6.2.8 Mach's In System Programming (ISP) Connector 102
6.2.9 PCI Connectors 102
6.2.10 System Expansion Connector 102
6.3 PCB Layout 102

Section 7
Support Information

7.1 Interconnect signals 103
7.1.1 P1 - RS232 ports 1 and 2 Connectors 103
7.1.2 P2 - 100/10 - Base-T Ethernet port Connector 104
7.1.3 P3 - COP / JTAG Connector 104
7.1.4 P4 - CPM Expansion Connector 105
7.1.5 P5, P6, P10, P11, P13, P14, P16, P18, P19 - Logic Analyzer MICTOR Connectors 112
7.1.6 P7, P8, P9 - PCI Connectors 113
7.1.7 P12 - ATX Power Supply Connector 115
7.1.8 P15 - Mach/Lattice ISP Connector 115
7.1.9 P17 - System Expansion Connector 116
7.2 Programmable Logic Equations 122
7.2.1 U14 - BCSR Code 122
7.2.2 U10 - PCI Interrupt Controller Code 157
7.2.3 U23 - SDRAM MUX/LATCH High 179
7.2.4 U22 - SDRAM MUX/LATCH Low 183
7.3 Schematics and Bill Of Materials 188
7.3.1 Schematics 188
7.3.2 Bill of Materials 210

List of Tables

1-1 MPC8266ADS-PCI specifications . . . 3
2-1 . MODCK(1:3) Encoding . . . 11
2-2 . SW2 - SDRAM DIMM Configuration EEPROM Slave Address . . . 15
3-1 MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device . . . 27
3-2 MPC8266ADS-PCI Memory Map - E2PROM as Boot Device . . . 28
3-3 BCSR/FLASH Power On Reset Configuration . . . 31
3-4 E2PROM Power On Reset Configuration . . . 31
3-5 SIU REGISTERS' PROGRAMMING . . . 32
3-6 Memory Controller Initializations For 66Mhz - FLASH as Boot Device . . . 33
3-7 Memory Controller Initializations For 66Mhz - E2PROM as Boot Device . . . 34
3-8 Memory Controller Initializations For 66Mhz . . . 36
4-1 BCSR/FLASH Hard Reset Configuration Word . . . 42
4-2 E2PROM Hard Reset Configuration Word . . . 44
4-3 PCI Interrupt Register Description . . . 49
4-4 PCI Interrupt Mask Register Description . . . 50
4-5 MPC8266-MB Chip Select Assignments . . . 54
4-6 SDRAM DIMM (84MHz) Performance Figures . . . 56
4-7 66 MHz SDRAM DIMM Mode Register Programming . . . 57
4-8 Flash Memory Projected Performance Figures . . . 59
4-9 L2 Cache CFG(0:2) Settings . . . 64
4-10 BCSR0 Description . . . 68
4-11 BCSR1 Description . . . 69
4-12 BCSR2 Description . . . 71
4-13 FLASH Presence Detect (7:5) Encoding . . . 72
4-14 FLASH Presence Detect (4:1) Encoding . . . 72
4-15 EXTOOLI(0:3) Assignment . . . 72
4-16 PQ2 Board Version Encoding . . . 73
4-17 PQ2 Board Revision Encoding . . . 73
4-18 External Tool Revision Encoding . . . 73
4-19 L2 Cache Size Encoding . . . 73
4-21 PCI Board Present Signal Definitions . . . 74
4-20 BCSR4 Description . . . 74
4-22 BCSR3 and BCSR5 Description . . . 75
4-23 BCSR6 Description . . . 75
4-24 BCSR7 Description . . . 75
4-25 COP/JTAG Port Signals Description . . . 77
4-26 JTAG Instruction Codes . . . 82
5-1 MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device . . . 86
5-2 MPC8266ADS-PCI Memory Map - E2PROM as Boot Device . . . 88
5-3 FLASH Power On Reset Configuration . . . 91

5-4 E2PROM Power On Reset Configuration 91
5-5 SIU REGISTERS' PROGRAMMING 92
5-6 Memory Controller Initializations For 66Mhz - FLASH as Boot Device 92
5-7 Memory Controller Initializations For 66Mhz - E2PROM as Boot Device 93
5-8 Memory Controller Initializations For 66Mhz 95
6-1 Expansion Connectors Maximum Current Consumption 99
6-2 Maximum Power Consumption Per Add-In Card 99
7-1 P1 Connector 103
7-2 P2 - 100/10 Base-T Ethernet Connector 104
7-3 P3 - COP/JTAG Connector 104
7-4 P4 - CPM Expansion Connector 105
7-5 P7, P8, P9 - PCI Connectors 113
7-6 P12 - ATX Power Supply Connector 115
7-7 P15 - Lattice ISP Connector 116
7-8 P17 - System Expansion Connector 117
7-9 MPC8266ADS-PCI Bill Of Materials 210

List of Figures

1-1 MPC8266-MB Block Diagram 6
2-1 MPC8266ADS-PCI Top Side Part Location Diagram 8
2-2 VDDL Range Selection - JP5 9
2-3 VDDL Trimmer - R26 10
2-4 SW4 Description 11
2-5 Hard Reset Configuration Source Selection - JP3 12
2-6 SW3 Description 12
2-7 JP1 - TDI Source Selection 14
2-8 SW2 Description 14
2-9 Chassis AC Voltage Settings 15
2-10 Host Controlled Operation Scheme 16
2-11 Stand Alone Configuration 17
2-12 P3 - COP/JTAG Port Connector 17
2-13 P1A/P1B - RS232 Serial Port Connector 18
2-14 Flash Memory SIMM Insertion 19
2-15 SDRAM DIMM Insertion 20
3-1 SW1 - Description 22
3-2 JP2 - Therm Connector 23
3-3 JP7 - Ventilator Supply 23
3-4 JP10 - VPP Source Selection 24
4-1 PCI Host Configuration Registers 46
4-2 PCI Interrupt Routing Scheme 48
4-3 Main Clock Generator Scheme 51
4-4 PCI Clock Generator Scheme 52
4-5 SDRAM DIMM Connection Scheme - No L2 Cache 55
4-6 SDRAM DIMM - 60x Bus Connection Scheme with L2 Cache 56
4-7 FLASH SIMM Connection Scheme 60
4-8 E2PROM Connection scheme 62
4-9 PCI Bus Scheme 63
4-10 RS232 Serial Ports Connector 67
4-11 Debug Station Connection Schemes 76
4-12 COP/JTAG Port Connector 76
4-13 Fast Download JTAG System 79
4-14 JTAG TAP Controller State Diagram 81
6-1 MPC8266-MB Power Scheme 99

1

General Information

1.1 Introduction

This document is an operation guide for the MPC8266ADS-PCI board. It contains operational, functional and general information about the MPC8266ADS-PCI. This board is meant to serve as a platform for s/w and h/w development around the MPC8266 processor. Using its on-board resources and a debugger, a developer is able to download code, run it, set breakpoints, display memory and registers and connect proprietary h/w via the expansion connectors, to be incorporated into a desired system with the MPC8266 processor.

This board could also be used as a demonstration tool (i.e., application s/w may be programmed[1] into its Flash memory and ran in exhibitions etc.).

1. Either on or off-board.

1.2 Definitions, Acronyms, and Abbreviations

MPC8266ADS-PCI	PowerQUICC II PCI MotherBoard
MPC8266	PowerQuicc 2 with PCI local bus
VOYAGER	MPC8260 - PowerQUICC 2
PPC	PowerPC
PCI	Peripheral Components Interconnect
CPM	Communication Processor Module
SDRAM	Synchronous Dynamic Random Access Memory
VADS	Voyager Application Development System
Kbyte	1024 bytes
LSB	Least Significant Byte
lsb	least significant bit
Mbyte	1048576 bytes
DIMM	Dual In-line Memory Module
SIMM	Single In-line Memory Module
TBD	To Be Defined
UPM	User Programmable Machine
EVB	Evaluation Board
GPCM	General Purpose Chip-select Machine
GPL	General Purpose Line
BCSR	Board Control and Status Register
FLASH	Non volatile reprogrammable memory.
ZIF	Zero Input Force
BGA	Ball Grid Array
ADI	Application Development Interface.
COP	Common On-chip Processor
SAR	Segmentation And Reassembly
UTOPIA	Universal Test & OPerations Interface for ATM

1.3 Related Documentation

- MPC8266 - User’s Manual.
- VADS Users’ Manual.
- MPC2605 Data Sheet. (*http://mot-sps.com/books/dl156/pdf/mpc2605rev6.pdf*)
- PMC-SIERRA 5350 Long Form Data Sheet (*http://www.pmc-sierra.com/PDF_Files/960924.pdf*)
- PMC-SIERRA 5350 Errata Notice (*http://www.pmc-sierra.com/PDF_Files/970171.pdf*)
- PMC-SIERA 5350 Reference Design (*http://www.pmc-sierra.com/PDF_Files/961062.pdf*)

- LXT970A (by Level One) Data Sheet (*http://www.level1.com/product quickref.html#network*)
- LXT970 Demo Board User's Guide (*http://www.level1.com/product/ quickref.html#network)*

1.4 Specifications

The MPC8266ADS-PCI specifications are given in Table 1-1.

Table 1-1. MPC8266ADS-PCI specifications

CHARACTERISTICS	SPECIFICATIONS
Power requirements (no other boards attached)	+5Vdc @ TBD A (Typ.), TBD A (Max.) +3.3Vdc @ TBD A (Typ.), TBD A (Max.) +12Vdc - @TBD A Max. -12Vdc - @TBD A Max.
Microprocessor	MPC8266 running @ 66 MHz Bus Clock Frequency.
Addressing Total address range on PPC Bus: Total address range on Local Bus: Flash Memory SIMM (PPC Bus) Synchronous Dynamic RAM DIMM (PPC Bus) Synchronous DRAM On Local Bus	 4 Giga Bytes (32 address lines) 256 KBytes External (18 address lines) 4 Giga Bytes Internal (32 address lines internal decoding) 8 MByte, 32 bits wide expandable to 32 MBytes 16 MByte, 64 bits wide. Support for up to 64 MByte 4 MBytes, organized as 1 Meg X 32 bit.
Operating temperature	10^{O}C - 30^{O}C (room temperature)
Storage temperature	-25^{O}C to 85^{O}C
Relative humidity	5% to 90% (non-condensing)
Dimensions: Length Width Thickness	 12" (305 mm) 9" (229 mm) 0.063" (1.6 mm)

1.5 MPC8266ADS-PCI Features

- 64 bit MPC8266 Communication Processor (MPC8266), running @ 66MHz external bus frequency with PCI module functioning as Host Bridge.
- 16 MByte, Unbuffered, 168 pin Synchronous Dram DIMM, residing on 60X bus, controlled by SDRAM machine 1. Support for upto 64 MBytes DIMM (single-bank only for the 64 MBytes DIMM). Optional address Latch - Multiplexer is required if a cache

module is assembled.

- Support for Automatic DIMM Identification via MPC8266 I^2C port and DIMM's' serial E^2PROM.
- Support for PBI (Page Based Interleaving) in both Single and 60x bus[1] modes.
- Optional 1/2 MByte L2-Cache on-board using 2 MPC2605 Look-Aside cache modules.
- 8 MByte, 80 pin Flash SIMM, buffered from 60X bus. Support for upto 32 MByte, controlled by GPCM, 5V/12V Programmable, with Automatic Flash SIMM identification, via BCSR. Support for both On and OFF SIMM Flash reset.
- 5V/12V VPP (in-circuit programming voltage) for Flash SIMM - jumper selectable.
- 8 KBytes E^2PROM, buffered from the 60x bus, controlled by the GPCM.
- Board Control & Status Register - BCSR, Controlling Boards' Operation.
- Fast Download support via JTAG.
- Power-On Reset Option via JTAG.
- Programmable Power-On Reset and Hard-Reset Configuration via E^2PROM or via Flash memory for the PQ2 core and PCI module.
- PCI Local Bus is PCI Standard 2.2 compliant.
- 3 PCI slots are available to host up to 3 masters/targets cards @ 3.3V only - arbitration is supported by the on-chip Arbiter.
- PCI bus supports 25 - 66 MHz @ 3.3V devices (determined by the user).
- Simple generic Interrupt Controller to handle the PCI interrupts (4 in each PCI slot).
- ATX form factor board design in a standard ATX case (in order to comply with the power requirements for the PCI slots according to the PCI standard).
- Module Enable Indications for all on-board modules.
- High density (MICTOR) Logic Analyzer connectors, carrying all 60x and CPM signals, for fast logic analyzer connection.
- 155 Mbps ATM UNI on FCC1 with Optical I/F, connected to the MPC8266 via UTOPIA I/F, using the PMC-SIERA 5350.
- 100/10-Base-T Port on FCC2 with T.P. I/F, MII controlled, using Level-One LXT970.
- Dual RS232 port residing on SCC1 & SCC2.
- Module disable (i.e., low-power mode) option for all communication transceivers -BCSR controlled, enabling use of communication ports, off-board via the expansion connectors.
- Dedicated MPC8266 communication ports expansion connectors for convenient tools' connection, carrying also necessary bus signals, for transceivers' M/P I/F connection. Use is done with 2 X 128 pin DIN 41612 receptacle connectors.
- External Tools' identification & status read capability, via BCSR.
- Separate Power-On Reset Push - Button, Soft / Hard[2] Reset Push - Button and ABORT Push - Button.
- ATX Power Supply.

1. BCSR controlled for 60x bus mode.
2. Hard reset is applied by depressing BOTH Soft Reset & ABORT buttons.

- Multi-Range MPC8266 internal logic operation voltage - determined at production between three ranges, 1.7V to 1.9V, 1.8V to 2.0V or 2.3V to 2.7V, currently changeable within a range.
- Software Option Switch provides 8 S/W options via BCSR.
- COP/JTAG connector for the MPC8266.

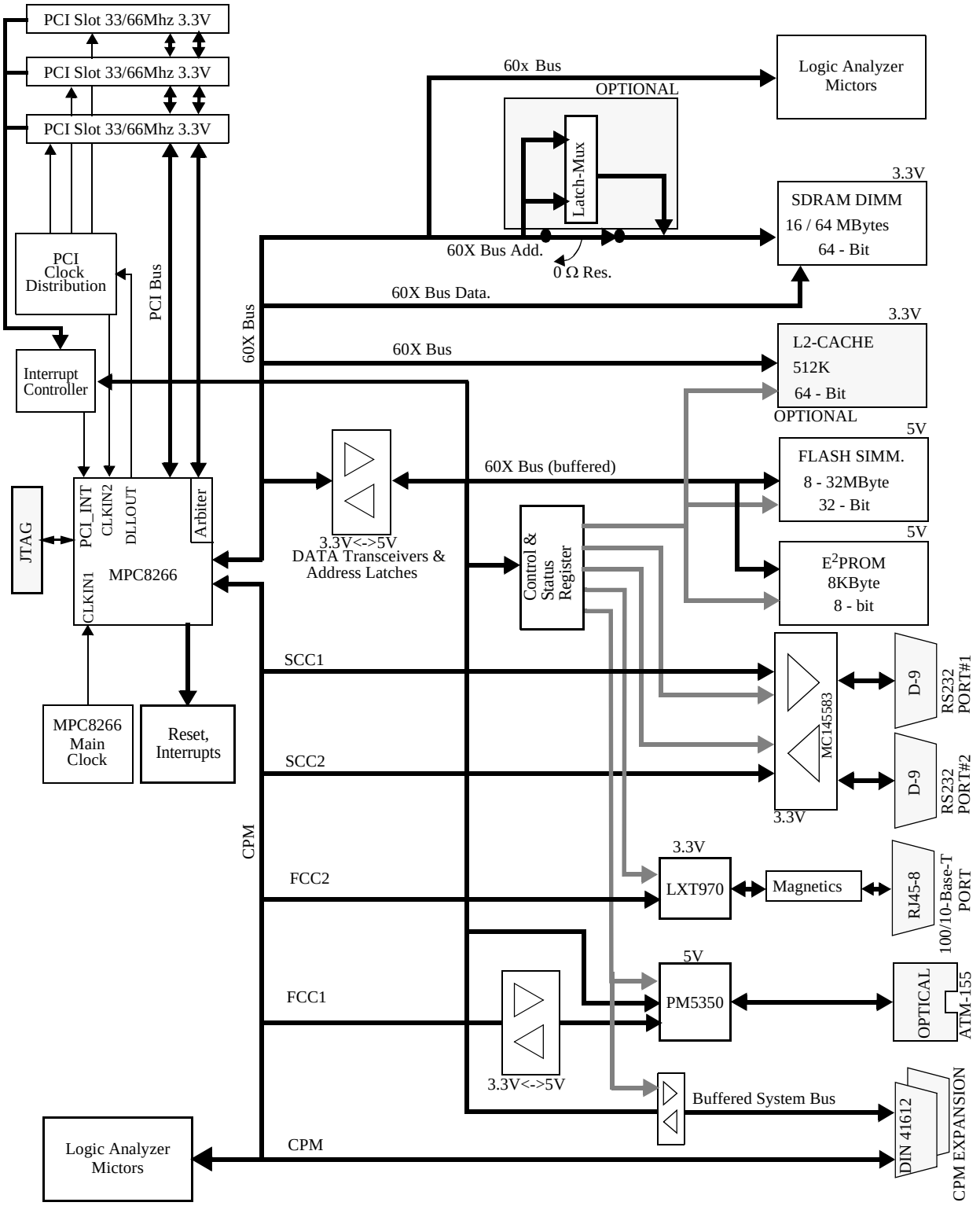


Figure 1-1. MPC8266-MB Block Diagram

2

Hardware Preparation and Installation

2.1 Introduction

This chapter provides unpacking instructions, hardware preparation, and installation instructions for the MPC8266ADS-PCI.

2.2 Unpacking Instructions

NOTE: If the shipping carton is damaged upon receipt, request carrier's agent to be present during unpacking and inspection of equipment.

Unpack equipment from shipping carton. Refer to packing list and verify that all items are present. Save packing material for storing and reshipping of equipment.

CAUTION

AVOID TOUCHING AREAS OF INTEGRATED CIRCUITRY; STATIC DISCHARGE CAN DAMAGE CIRCUITS.

2.3 Hardware Preparation

To select the desired configuration and ensure proper operation of the MPC8266ADS-PCI board, changes of the Dip-Switch settings may be required before installation. The location of the switches, indicators, Dip-Switches, and connectors is illustrated in Figure 2-1.. The board has been factory tested and is shipped with Dip-Switch settings as described in the following paragraphs.

Parameters can be changed for the following conditions:

- MPC8266's Internal Logic Supply Level Range Via JP5.
- MPC8266's Internal Logic Supply Level within range (VDDL) Via R26.
- MPC8266's MODCK(1:3). Determining Core's and CPM's PLLs multiplication factor via SW4.
- MPC8266's Boot Source - BCSR or Memory (FLASH/EEPROM) - via JP3.
- MPC8266's HARD Reset Configuration Source (FLASH/EEPROM) via SW3.
- MPC8266's MODCKH(0:3) via SW4 .
- MPC8266's PCI_MODCK via SW4.
- MPC8266's $\overline{\text{PCI\_ARBITER}}$ via SW3.
- MPC8266's PCI_DLL via SW3.
- MPC8266's JTAG's TDI Source Selection via JP1.
- SDRAM DIMM's I^2C Slave Address via SW2.
- ATX Chassis Main AC Power Selection.

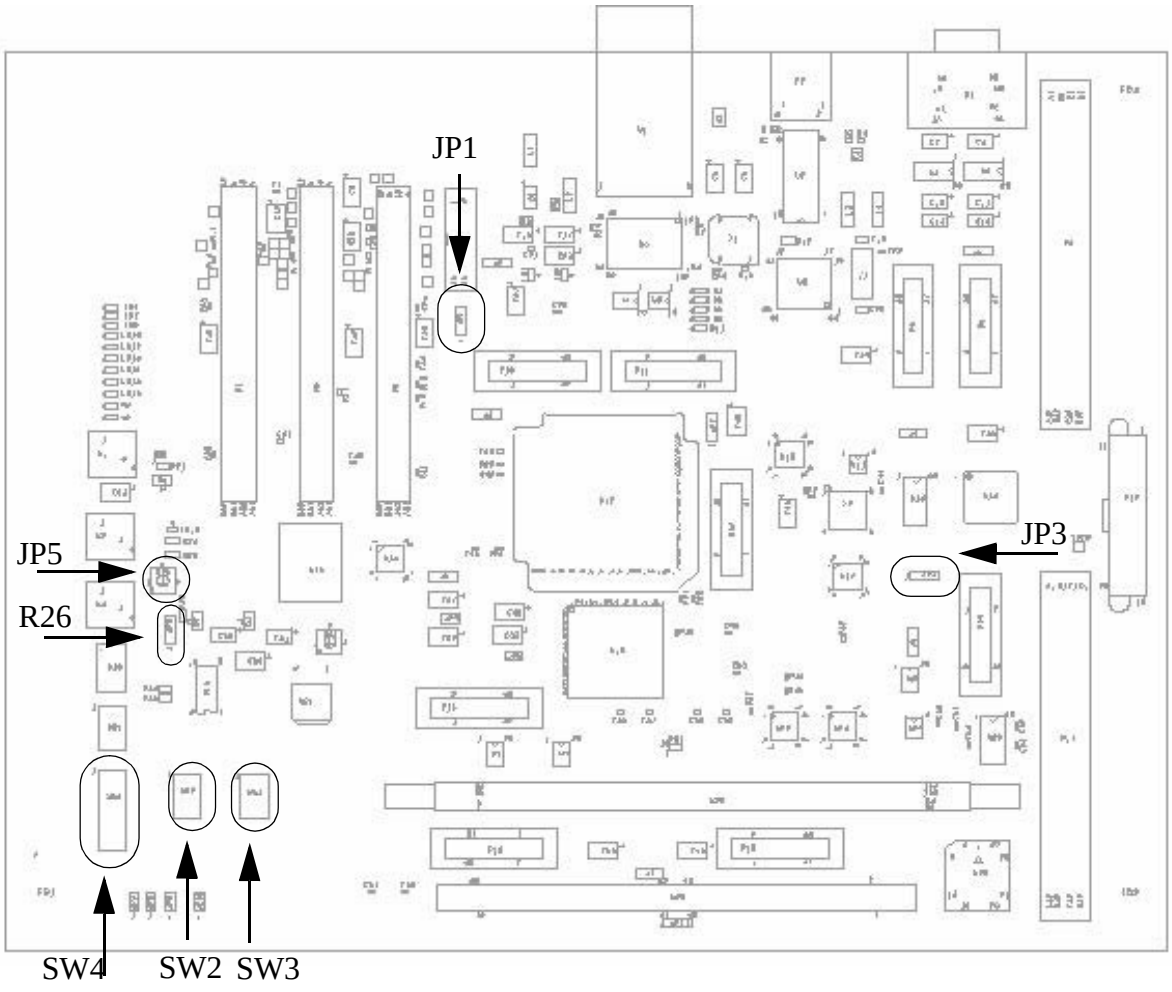


Figure 2-1. MPC8266ADS-PCI Top Side Part Location Diagram

2.3.1 Setting VDDL Level Range - JP5

To support future revisions of the MPC8266, provisions are taken to provide necessary voltage levels on VDDL, to match the process by which the MPC8266 is manufactured. Via JP5, three voltage level ranges are provided (JP5 setting options are shown in Figure 2-2.):

1. When a jumper is placed between positions **1 - 2** of JP5, a level range of **2.3V to 2.7V** on VDDL is selected. This level matches the current specification for the MPC8260 (Hip3).
2. When a jumper is placed between positions **2 - 3** of JP5, a level range of **1.7V to 1.9V** on VDDL is selected. This level range is a preparation for the next revision of the MPC8266(Hip4).
3. When a jumper is **misplaced** for JP5, a level range of **1.8V to 2.0V** is selected for VDDL. This is in preparation for 2V capable future devices.

.

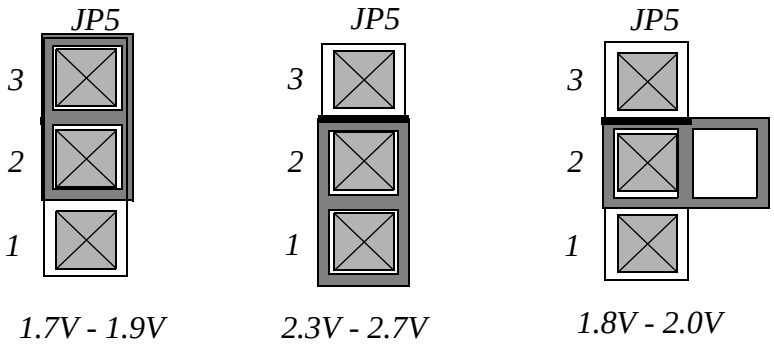


Figure 2-2. VDDL Range Selection - JP5

WARNING

JP5 is Factory Set according to the revision of MPC8266 with which it is assembled. Prior to changing a MPC8266 device, Extra Care should be taken with JP5 setup. If a selected Voltage Range is above the specification for the newly inserted MPC8266, PERMANENT DAMAGE might be inflicted to the device.

JP5 selects only a range of Voltage levels on VDDL. The actual level is selected by R26. See next paragraph.

2.3.2 Setting VDDL Supply Voltage Level

After VDDL's Voltage Level Range is selected via JP5, the actual level of VDDL is tuned via R26. VDDL may be measured upon JP4, using a DVM or any other high input impedance voltage measuring device.

VDDL level is factory set at the mid-range for the appropriate level range, but may be changed via R26. Rotating R26 CCW will reduce VDDL voltage down to range-low, while rotating it CW,

will increase VDDL upto range-high. LD17, provides visual indication for VDDL level, it illuminates brighter with rise of VDDL. VDDL change Vs. R26's rotation direction is shown in Figure 2-3.:

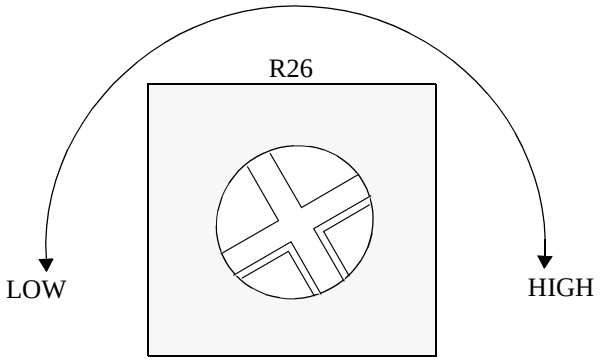


Figure 2-3. VDDL Trimmer - R26

WARNING

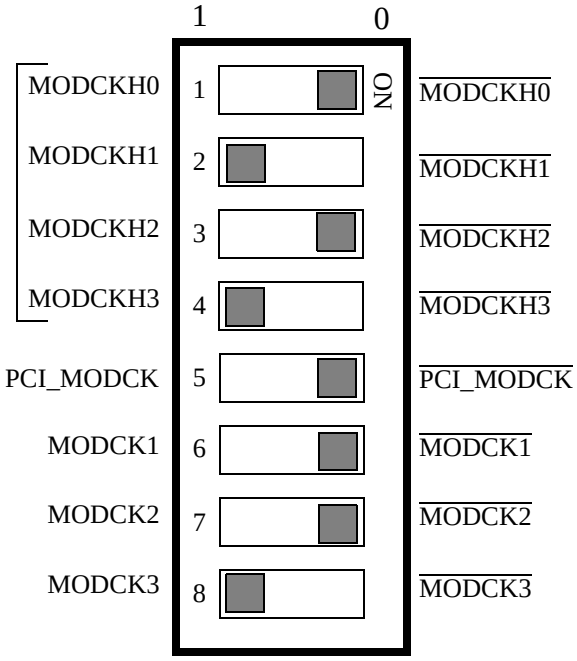
While in higher ranges of VDDL and higher ranges of internal operation frequencies, the MPC8266 might require some sort of COOLING measures to be taken. Failure in doing so, might result in PERMANENT DAMAGE inflicted to the MPC8266.

2.3.3 Setting MODCK(1:3) for PLLs Multiplication Factor - SW4 (#6 - #8)

After (1K cycles) the negation of the Power On Reset signal, the MPC8266 samples the 7 MODCK lines - the lower 3 on MODCK(1-3) and the upper four - MODCKH(0:3) field (read from the Hard-Reset configuration source when the PCI is disabled[1]), to establish the multiplication factors of the CPM's and Core's PLLs. The levels on **MODCK(1:3)** lines are set using **SW4**, switches **#6 - #8**. When an individual switch is at the **OFF** position its associated MODCK line is pulled-**high** ('1'), while when at the **ON** position, the associated MODCK is pulled-**down** ('0'). SW4 is shown in Figure 2-4., while the various combinations for SW4 #6 - #8 and their associated MODCK(1:3) values are shown in Table 2-1..

1.May be either boot FLASH or EEPROM on the ADS.

.



SW4
66MHz Bus and 66MHz PCI Bus Factory Set

Figure 2-4. SW4 Description

Table 2-1. . MODCK(1:3) Encoding

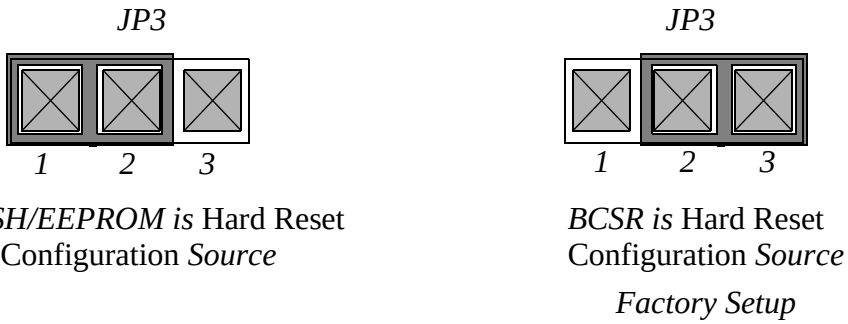
MODCK(1:3)	Switch 6	Switch 7	Switch 8
0	ON	ON	ON
1	ON	ON	OFF
2	ON	OFF	ON
3	ON	OFF	OFF
4	OFF	ON	ON
5	OFF	ON	OFF
6	OFF	OFF	ON
7	OFF	OFF	OFF

2.3.4 Setting Hard - Reset Configuration Source - JP3

The Boot sequence which starts when $\overline{\text{HRESET}}$ is asserted, may be from two sources:

1. BCSR (default Hard-Reset Configuration Word - $\overline{\text{CS0}}$ is assumed to be assigned to the FLASH)
2. Memories (FLASH/EEPROM - user controlled Hard-Reset Configuration Word)

When a jumper is placed between positions **1 - 2** of JP3, the Hard Reset Configuration source is a memory (FLASH/EEPROM) as configured by switch SW3-1. When a jumper is set between positions 2 - 3 of JP3, the Hard Reset Configuration source is the BCSR. See Figure 2-5..



FLASH/EEPROM is Hard Reset Configuration *Source*

BCSR is Hard Reset Configuration *Source*

Factory Setup

Figure 2-5. Hard Reset Configuration Source Selection - JP3

2.3.5 Setting Boot Source

The Hard - Reset configuration word[1], read by the MPC8266 while $\overline{\text{HRESET}}$ is asserted, may be taken from three sources:

1. Flash Memory SIMM
2. EEPROM
3. BCSR

For additional information as for the contents of the Hard-Reset configuration word see 4.1.2.4 "Hard Reset Configuration" on page 41.

SW3#1 actually assignes $\overline{\text{CS0}}$ to the FLASH (default when booting from the BCSR) or to the EEPROM. When SW3 #1 is **OFF**, the Hard Reset configuration word is taken from **EEPROM**, when it is **ON**, the Hard Reset configuration word is taken from the ***Flash SIMM***. See Figure 2-6..

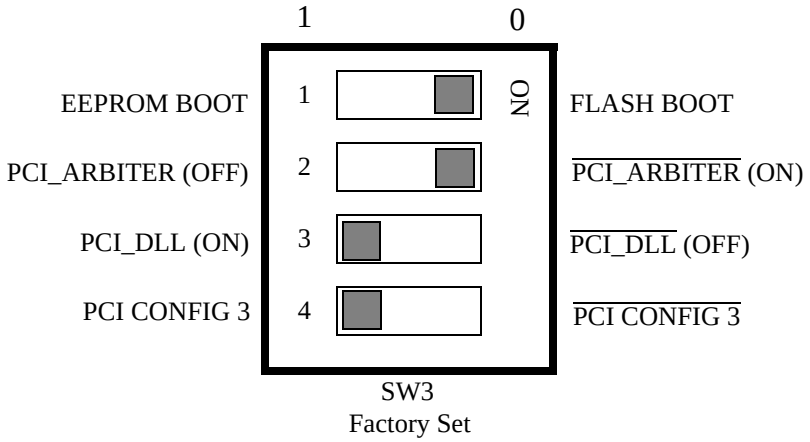


Figure 2-6. SW3 Description

1.In fact 8 Hard-Reset configuration words are read by a configuration master, however only the first is relevant for a single MPC8266.

2.3.6 Setting MODCKH(0:3) - for PLLs Multiplication Factors

When the Hard Reset configuration word is taken from Flash SIMM/EEPROM, the functionality of the MODCKH(0-3) bits in the Hard Reset Configuration Word depends on the mode of the PCI. When the PCI mode in the MPC8266 is enabled (set permanently on this board), the MODCKH(0:3) lines are taken from SW4 and the MODCKH(0:3) bits in the Hard Reset Configuration Word are ignored. When the PCI mode in the MPC8266 is disabled (not available on this board), MODCKH(0:3) are taken from the Hard Reset Configuration Word. SW4 #1 - #4 set the upper 4 bits of the MODCK field, during Hard Reset Configuration acquisition. When an individual switch of SW4 **#1 - #4** is at the **OFF** position, its corresponding MODCKH line is pulled-**high** (‘1’) during Hard Reset, while when at the **ON** position, pulled-***down*** (‘0’) (see Figure 2-4.).

2.3.7 Setting PCI_MODCK - for PCI Bus Clock

The settings of this line, determines the frequency of the PCI bus. When PCI_MODCK is set low, the PCI bus frequency is set by the the MODCK lines. When set high, the PCI bus frequency is half of what is set by the MODCK lines. When switch SW4 **#5** is at the **OFF** position, its corresponding PCI_MODCK line is pulled-**high** (‘1’ - enabled), while when at the **ON** position, pulled-***down*** (‘0 - disabled’) (see Figure 2-4.).

2.3.8 Setting $\overline{\text{PCI\_ARBITER}}$ - for PCI Mode Enabled

The settings of this line, determines the operation of the PCI Arbiter. When $\overline{\text{PCI\_ARBITER}}$ is set low, the PCI Arbiter in the MPC8266 is enabled. When set high, the PCI Arbiter is disabled and an external arbiter can be used. When switch SW3 **#2** is at the **OFF** position, its corresponding $\overline{\text{PCI\_ARBITER}}$ line is pulled-**high** (‘1’ - disabled), while when at the **ON** position, pulled-***down*** (‘0’ - enabled) (see Figure 2-6.).

2.3.9 Setting PCI_DLL - for PCI Mode Enabled

The settings of this line, determines the operation of the DLL for PCI Mode enabled. When PCI Mode is enabled, the DLL must be enabled. When PCI_DLL is set low, the DLL is disabled. When set high, the DLL is enabled. When switch SW3 **#3** is at the **OFF** position, its corresponding PCI_DLL line is pulled-**high** (‘1’ - enabled), while when at the **ON** position, pulled-***down*** (‘0’ - disabled) (see Figure 2-6.).

2.3.10 MPC8260 JTAG’s TDI Source Selection - JP1

A new JTAG machine was inserted in front of the MPC8266’s COP/JTAG port, this, to provide fast-download capability for the ADS.

Via JP1, it is possible to bypass[1] the new JTAG machine. When a jumper is placed between positions **1 - 2** of JP1, then, the Fast-download JTAG machine may be **enabled** to precede the MPC8266 in the JTAG chain. When a jumper is placed between positions **2**–**3** of JP1, the Fast-download JTAG machine is ***bypassed*** and the TDI input goes directly[2] from the COP/JTAG connector P3 to the MPC8266. See also 4.14.1 "Fast Download Support" on page 78.

1.HARD bypass. Even when the Fast download logic enabled via JP5, it wakes-up asynchronously bypassed.
2.Through a noise filtering network.

JP1 should be set between 2 - 3 if problems are encountered with the use of existing COP/JTAG debug equipment since this indicates that its software is not capable of using the fast download machine.

Figure 2-7. JP1 - TDI Source Selection

2.3.11 SDRAM DIMM I²C Slave Address Selection - SW2

The SDRAM DIMM has a serial configuration EEPROM, which is accessed using I^2C protocol. Each slave device on that bus has an 7 bit address field, 3 of which (the LSBs) within this device may be set externally.

Since the SDRAM DIMM configuration is read using the MPC8266's I^2C port, which may be required for user's application, an option is given to change the SDRAM DIMM configuration EEPROM slave address for the convenience of the user. SW2 is shown in Figure 2-8.:

.

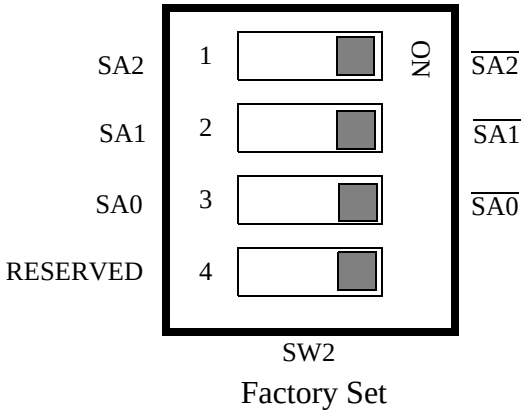


Figure 2-8. SW2 Description

The various position combinations of SW2 and their associated SDRAM DIMM configuration

EEPROM's I^2C slave addresses are shown in Table 2-2.:

Table 2-2. . SW2 - SDRAM DIMM Configuration EEPROM Slave Address

Slave Address [bin]	Switch 1	Switch 2	Switch 3
1010**000**	ON	ON	ON
1010**001**	ON	ON	OFF
1010**010**	ON	OFF	ON
1010**011**	ON	OFF	OFF
1010**100**	OFF	ON	ON
1010**101**	OFF	ON	OFF
1010**110**	OFF	OFF	ON
1010**111**	OFF	OFF	OFF

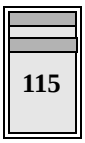
SW2 is factory set to: 1 - ON, 2 - ON, 3 - ON.

2.3.12 Setting ATX Chassis Main AC Power

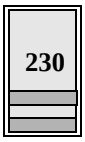
The ATX Chassis has a Power Supply which can operate from two different AC voltages - 115VAC and 230 VAC. It is important that the user will set the proper AC voltage on the Power Supply according to the outlet AC voltage where the chassis is going to be pluged. See Figure 2-9.for setting options.

WARNING

Before turning on the power of the ATX chassis, make sure that the AC voltage selector is set on the proper AC voltage. Failure in doing so, will result in PERMANENT DAMAGE inflicted to the chassis power supply.



115VAC Setting



230VAC Setting
Factory set

Figure 2-9. Chassis AC Voltage Settings

2.4 Installation Instructions

When the MPC8266ADS-PCI has been configured as desired by the user, it can be installed according to the required working environment as follows:

- Host Controlled Operation
- Stand-Alone

2.4.1 Host Controlled Operation

In this configuration the MPC8266ADS-PCI is controlled by a host computer via the COP port, which is a subset of the JTAG port. This configuration allows for extensive debugging using on-host debugger. The host is connected to the ADS by a COP controller provided by a third party.

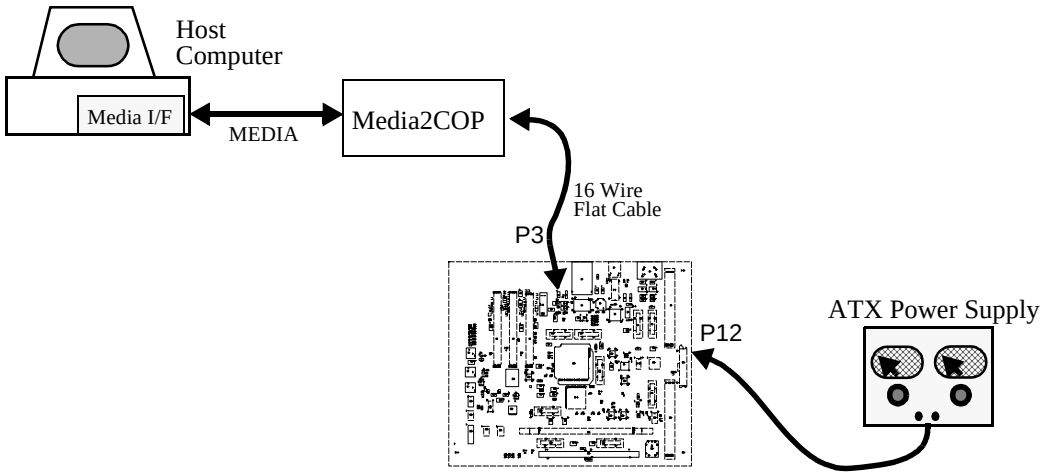


Figure 2-10. Host Controlled Operation Scheme

2.4.2 Stand Alone Operation

In this mode, the ADS is not controlled by the host via the COP port. It may connect to host via one of its other ports, e.g., RS232 port, Fast Ethernet port, ATM155 port etc. Operating in this mode requires an application program to be programmed into the board's Flash memory.

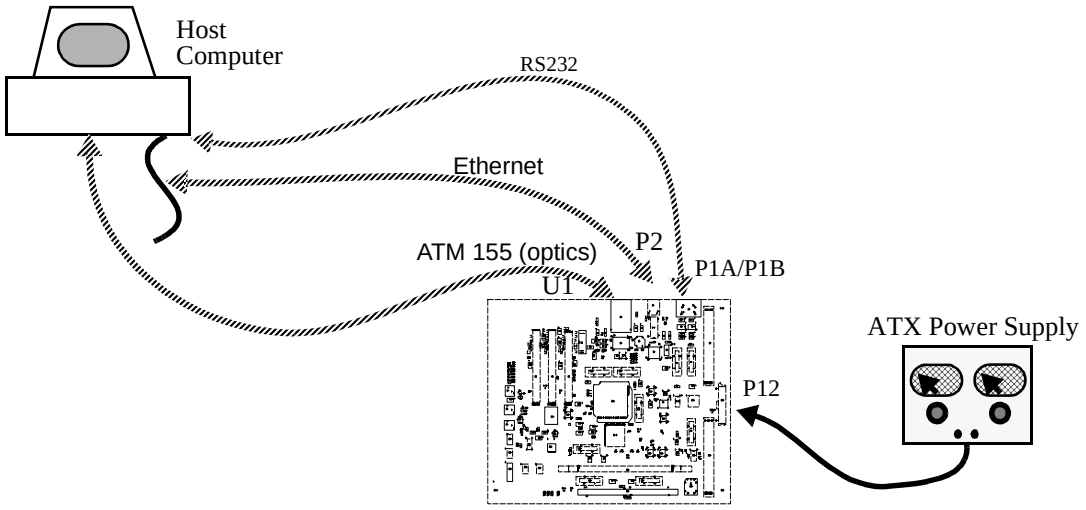


Figure 2-11. Stand Alone Configuration

2.4.3 COP/JTAG Connector - P3

The MPC8266ADS-PCI COP interface connector, P3, is a 16 pin, male, Header connector. The connection between the MPC8266ADS-PCI and the COP controller is by a 16 line flat cable, supplied with the COP controller board obtained from a third party developer. Figure 2-12. shows the pin configuration of the connector.

.

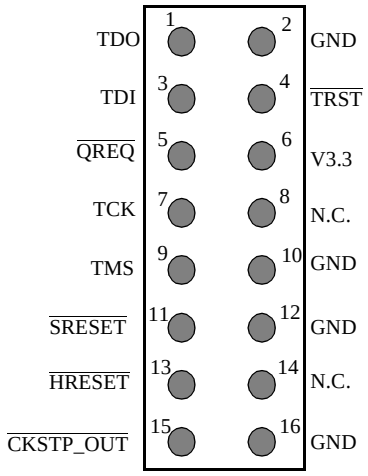


Figure 2-12. P3 - COP/JTAG Port Connector

2.4.4 Terminal to MPC8266ADS-PCI RS-232 Connection

A serial (RS232) terminal or any other RS232 equipment, may be connected to the RS-232 connectors P1A and P1B. The RS-232 connectors are a 9 pin, female, D-type connectors, arranged in a stacked configuration. P1B connected to SCC2 of the MPC8266 is the **lower** and P1A, connected to SCC1 of the MPC8266, is the ***upper*** in the stack.

The connectors are arranged in a manner that allows for 1:1 connection with the serial port of an IBM-AT[1] or compatibles, i.e. via a flat cable. The pinout which is identical for both P1A and P1B is shown in Figure 2-13..

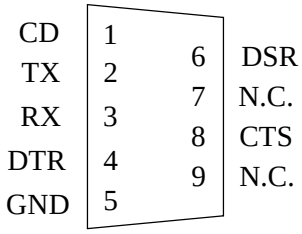


Figure 2-13. P1A/P1B - RS232 Serial Port Connector

2.4.5 10/100-Base-T Ethernet Port Connection

The 10/100-Base-T port connector - P2, is an 8-pin, 90^{o}, receptacle RJ45 connector. The connection between the 10/100-Base-T port to the network is done by a standard cable, having two RJ45/8 jacks on its ends. The pinout of P2 is described in Table 7-2. "P2 - 100/10 Base-T Ethernet Connector" on page 104.

2.4.6 Memory Installation

The MPC8266ADS-PCI is supplied with two types of memory modules:

- Synchronous Dynamic Memory DIMM
- Flash Memory SIMM.

2.4.6.1 Flash Memory SIMM Installation

To install a memory SIMM, it should be taken out of its package, put diagonally in its socket - U30 (no error can be made here, since the Flash socket has 80 contacts, while the SDRAM socket has 168) and then raised to a vertical position until the metal lock clips are locked. See Figure 2-14..

1.IBM-AT is a trademark of International Business Machines Inc.

CAUTION

The memory SIMMs have alignment nibble near their # 1 pin. It is important to align the memory correctly before it is twisted, otherwise damage might be inflicted to both the memory SIMM and its socket.

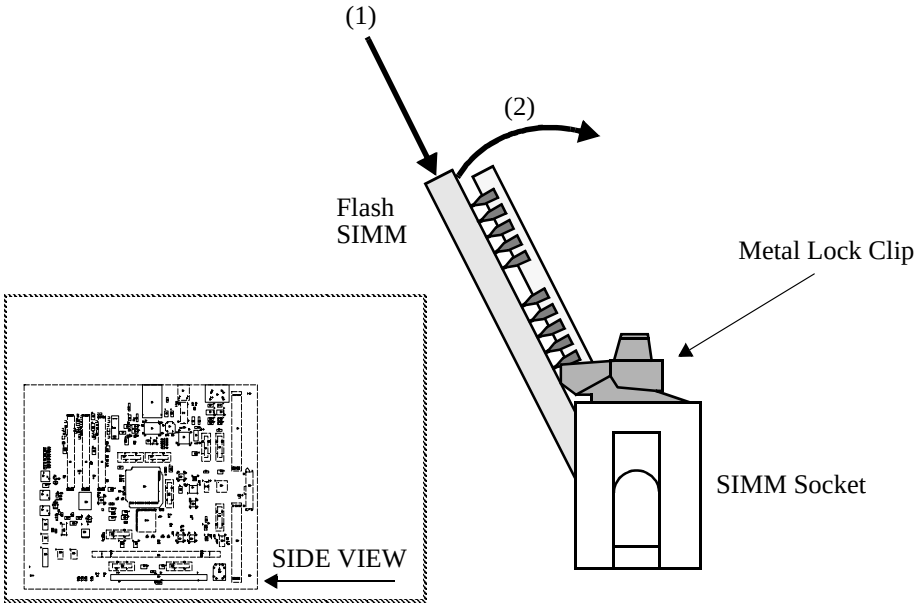


Figure 2-14. Flash Memory SIMM Insertion

2.4.6.2 SDRAM DIMM Installation

The SDRAM DIMM (U28) is inserted in a different manner: The 2 side-latches are pulled aside to unlocked position, the DIMM is placed in a vertical position (similar to its final position) between them, so that the keys, nibbled in the DIMM matches those on the socket and then, the DIMM should be pressed evenly and firmly into its place, locking the side locks on itself. The SDRAM insertion is shown in Figure 2-15.

.

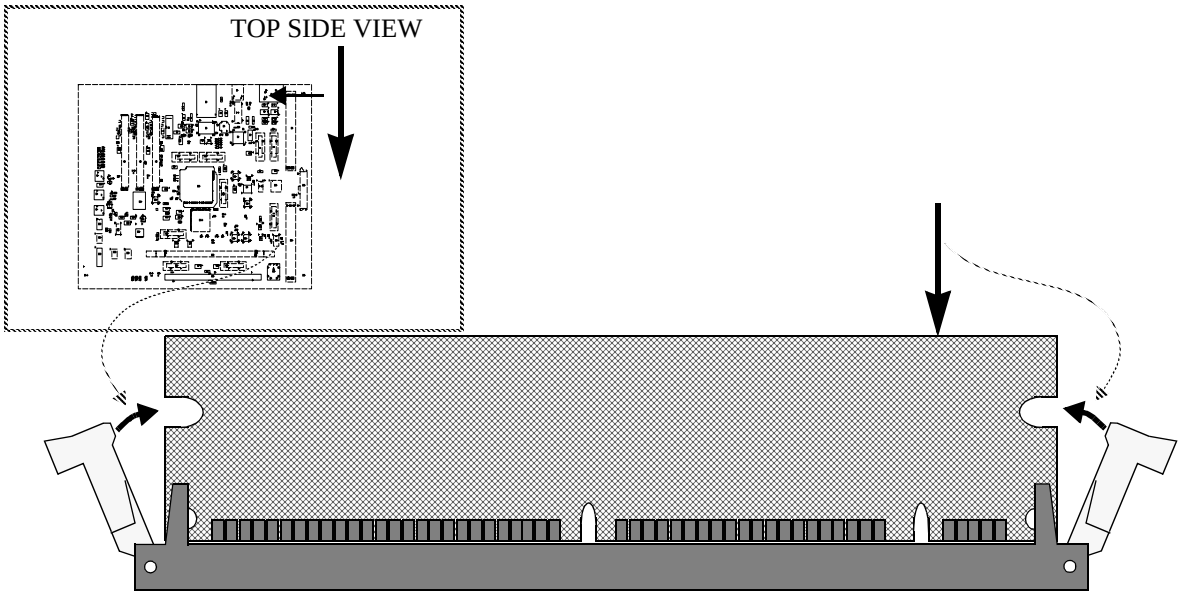


Figure 2-15. SDRAM DIMM Insertion

3

Operating Instructions

3.1 Introduction

This chapter provides necessary information to use the MPC8266ADS-PCI in host-controlled and stand-alone configurations. This includes controls and indicators, memory map details, and software initialization of the board.

3.2 Controls and Indicators

The MPC8266ADS-PCI has the following switches and indicators.

3.2.1 Power-On RESET Switch - S1

The Power-On RESET switch S1 performs Power-On reset to the MPC8266, as if the power was re-applied to the ADS. When the MPC8266 is reset that way, all configuration and all data residing in volatile memories are lost. After $\overline{\text{PORST}}$ signal is negated, the MPC8266 re-acquires the power-on reset and hard-reset configuration data from the hard-reset configuration source. (Flash | EEPROM | BCSR).

3.2.2 ABORT Switch - S2

The ABORT switch is normally used to abort program execution, this by issuing a level 0 interrupt to the MPC8266. If the ADS is in stand alone mode, it is the responsibility of the user to provide means of handling the interrupt, since there is no resident debugger with the MPC8266ADS-PCI. The ABORT switch signal is debounced, and may be disabled by software.

3.2.3 SOFT RESET Switch - S3

The SOFT RESET switch S2 performs Soft reset to the MPC8266 internal modules, maintaining MPC8266's configuration (clocks & chip-selects) and SDRAMs' contents. The switch signal is debounced, and it is not possible to disable it by software.

3.2.4 HARD RESET - Switches - S2 & S3

When BOTH switches - S2 and S3 are depressed simultaneously, HARD reset is generated to the MPC8266. When the MPC8266 is HARD reset, all its configuration is lost[1], including data stored in the SDRAMs and the MPC8266 has to be re-initialized.

1.Except for Hard-Reset configuration word, which is acquired only once, after PON-Reset.

3.2.5 SW3 - Reset Configuration Switch

SW3 is a 4-switch Dip-Switch. For its function see Section 2.3.5.

3.2.6 SW2 - SDRAM DIMM Configuration Memory I^2C Slave Address Switch

SW2 sets the I^2C slave address for the SDRAM DIMM's serial configuration memory. For its function see Section 2.3.11.

3.2.7 SW1 - Software Options Switch

SW1 is a 4-switch Dip-Switch. This switch is connected over SWOPT(0:2) lines which are available at BCSR2, S/W options may be manually selected, according to SW1 state. SW1 is factory set to all ON. See Figure 3-1.

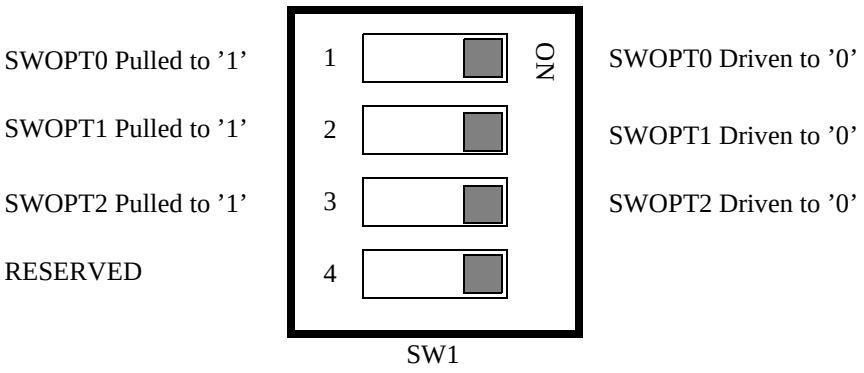


Figure 3-1. SW1 - Description

3.2.8 JP5 - VDDL Voltage Level Range Selection

JP3 selects between 3 different voltage level ranges available for VDDL. For further information over its function see Section 2.3.1.

3.2.9 JP6 - IDDL Measurement

JP4 resides in IDDL's main current flow. To measure IDDL, JP6 should be removed using a solder tool and a current meter should be connected instead with wires as short and thick as possible.

Warning

The job of removing JP6 and soldering the current meter connections instead is very delicate and should be done by a skilled technician.

If this process is done by unskilled hands or repeated more than 3 times, permanent damage may occur to the MPC8266ADS-PCI.

3.2.10 JP2 - Thermal Sense Connector

There are 2 dedicated pins THERM(0:1) which provide a way to take internal temperature

measurements of the MPC8266. These pins should be connected to GND for normal operation. JP1 is factory set with a jumper on its 2 - 3 positions, so that THERM1 is connected to GND.

.

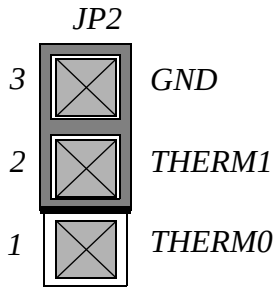


Figure 3-2. JP2 - Therm Connector

3.2.11 JP7 - Optional Ventilator Supply

An optional cooling ventilator for the MPC8266 may be powered via JP7, a 2 pin header connector (not assembled). In order to connect a ventilator to JP7, either a 0.1" pitch header should be soldered to it, to be connected to a matching female connector or the ventilator supply wires may be soldered directly to JP7, shown in Figure 3-3.

Warning

The job of soldering wires or header to J7 is very delicate and should be done by a skilled technician.

If this process is done by unskilled hands or repeated more than 3 times, permanent damage may occur to the MPC8266ADS.

.

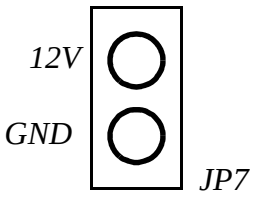


Figure 3-3. JP7 - Ventilator Supply

3.2.12 JP1 - COP/JTAG TDI Source Selection

JP1 sets the structure of the COP/JTAG chain on the ADS. For further information over its function see Section 2.3.10.

3.2.13 JP4- IDDH Measurement

JP4 resides in IDDH's main current flow. To measure IDDH, JP4 should be removed using a solder tool, and a current meter should be connected, with as wires as short and thick as possible.

Warning

The job of removing JP4 and soldering current meter connections instead is very delicate and should be done by a skilled technician.

If this process is done by unskilled hand or repeated more than 3 times, permanent damage might be inflicted to the MPC8266ADS-PCI.

3.2.14 JP11 - VPP Source Selector

JP11 selects the source for VPP - programming voltage for the Flash SIMM. When a jumper is located between pins **2 - 3** of JP11 (Factory Set), the VPP is connected to the VCC plane of the ADS, providing **5V VPP**. When a jumper is located between positions ***1 - 2*** of JP11, VPP is drawn from the 12V plane, that provides ***12V VPP***. JP11 options are shown in Figure 3-4.

.

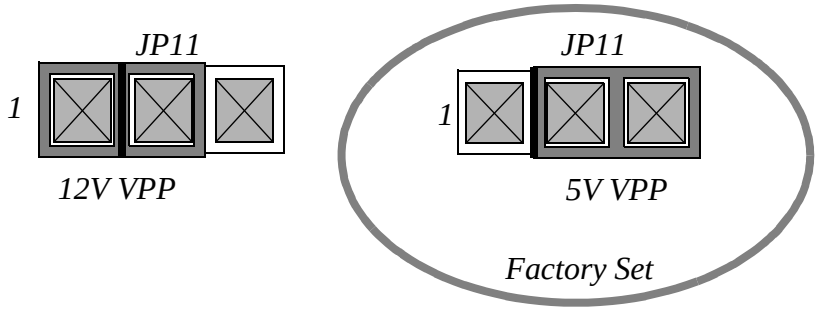


Figure 3-4. JP10 - VPP Source Selection

3.2.15 GND Bridges

There are 7 GND bridges on the MPC8266ADS-PCI. These bridges are meant to assist general measurements and logic-analyzer connection.

Warning

When connecting to a GND bridge, use only INSULATED GND clips. Otherwise, un-insulated clips may cause short- circuits, touching "HOT" points around them. Failure in doing so, might result in permanent damage to the MPC8266ADS-PCI.

3.2.16 ATM TX Indicator - LD1

The green ATM Receive LED indicator blinks whenever the PM5350 ATM-UNI is transmitting cells via the ATM port. It illuminates only when the ATM transceiver is enabled via BCSR1.

3.2.17 Fast Ethernet Indicator - LD2

When the LXT970 is enabled and is in 100 Mbps operation mode, the yellow led - LD2 lights.

3.2.18 Fast Ethernet CLSN Indicator - LD3

The red Ethernet Collision LED indicator CLSN, lights whenever a collision condition is detected

on the 10/100-Base-T port, i.e., simultaneous receive and transmit. This led functions in this duty provided that bits (7:6) of LXT970's register 19, are cleared.

3.2.19 Ethernet LINK Indicator - LD4

The yellow Ethernet Twisted Pair Link Integrity LED indicator - LINK, lights to indicate good link integrity on the 10/100-Base-T port. LD4 is off when the link integrity fails.

3.2.20 Ethernet TX Indicator - LD5

The green Ethernet Receive LED indicator blinks whenever the LXT970 is transmitting data via the 10/100-Base-T port.

3.2.21 12V Indicator - LD6

The green 12V led - LD6, indicates the presence of the +12V supply on the ADS.

3.2.22 ATM RX Indicator - LD7

The green ATM Receive LED indicator blinks whenever the PM5350 ATM-UNI is receiving cells via the ATM port. It illuminates only when the ATM transceiver is enabled via BCSR1.

3.2.23 Ethernet RX Indicator - LD8

The green Ethernet Receive LED indicator blinks whenever the LXT970 is receiving data from the 10/100-Base-T port.

3.2.24 5V Indicator - LD9

The green 5V led - LD9, indicates the presence of the +5V supply on the ADS.

3.2.25 -12V Indicator - LD10

The green -12V led - LD10, indicates the presence of the -12V supply on the ADS.

3.2.26 3.3V Indicator - LD11

The green 3.3V led - LD11, indicates the presence of the +3.3V supply on the ADS.

3.2.27 RUN Indicator - LD12

When the green RUN led - LD12 is lit, it indicates that the MPC8266 is performing cycles on the PPC Bus. When dark, the MPC8266 is either running internally or stuck.

3.2.28 ATM ON - LD13

When the yellow ATM ON led is lit, it indicates that the ATM-UNI transceiver - the PM5350, is enabled for communication. When it is dark, the ATM-UNI transceiver is disconnected from the MPC8266, enabling the use of its associated FCC1 pins off-board via the expansion connectors.

ATM ON led is controlled by BCSR1.

3.2.29 Fast Ethernet Port Initially Enabled - LD14

When the yellow ETH ON led is lit, it indicates that the fast ethernet port transceiver - the LXT970, is **initially** active. When it is dark, it indicates that the LXT970 is **initially** in power down mode, enabling the use of its associated FCC2 pins off-board via the expansion connectors.

The state of LD14 is controlled by BCSR1.

This is a **soft**-indication, i.e., since the LXT970 may be controlled via the MII port, it is possible that the state of LD14 does not reflect correctly the status of the LXT970.

Note

Application S/W should always seek to match the state of LD14 to the status of the LXT970, so that, this indication is made reliable as to the correct status of the LXT970.

3.2.30 RS232 Port 1 ON - LD15

When the yellow RS232 Port 1 ON led is lit, it designates, that the RS232 transceiver connected to P1A (upper DB9 connector), is active and communication via that medium is allowed. When darkened, it designates that the transceiver is in shutdown mode and its associated SCC1 pins may be used off-board via the expansion connectors.

3.2.31 RS232 Port 2 ON - LD16

When the yellow RS232 Port 2 ON led is lit, it designates that the RS232 transceiver connected to P1B (lower DB9 connector) is active and communication via that medium is allowed. When darkened, it designates, that the transceiver is in shutdown mode and its associated SCC2 pins may be used off-board via the expansion connectors.

3.2.32 VDDL Indication - LD17

The green VDDL indicator led - LD17 is lit to indicate a VDDL power activity. Since VDDL level may vary, LD17's illumination level also varies accordingly.

3.3 Memory Map

All accesses to MPC8266ADS-PCI's memory slaves are controlled by the MPC8266's memory controller. Therefore, the memory map is reprogrammable to the desire of the user. After Hard Reset is performed by the debug station, the debugger checks for existence, size, delay and type of the SDRAM DIMM and FLASH memory SIMM mounted on board and decides on the assignments of $\overline{CS0}$ and $\overline{CS4}$ (E^2PROM and FLASH) and programs the memory controller accordingly. The SDRAM, E^2PROM and the FLASH memory, respond to all types of memory access i.e., problem / supervisory, program / data and DMA.

This memory map is a recommended memory map and since it is a "soft" map, devices' address may be moved about the map, to the convenience of any user. There are actually two memory maps which depend on the device assigned to $\overline{CS0}$ (regardless of the boot source). The memory address for the device assigned to $\overline{CS0}$ is always the same as determined in the Hard-Reset configuration word. Since the FLASH and E^2PROM require different memory spaces, different

memory maps are devised for each case. For details see Table 3-1. and Table 3-2.

Table 3-1. MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device

Address Range	Memory Type	Device Name		Port Size	Memory Size
00000000 - 00FFFFFF	SDRAM DIMM	SDCUV6482 (16 MByte)	SDC8UV6484 (64 MByte)	64	64 MByte
01000000 - 03FFFFFF					
04000000 - 044FFFFF	Empty Space			-	5 MByte
04500000 - 04507FFF	BCSR(0:7)[a]			32	32 KByte
04500000 - 04507FE3	BCSR0				4 Byte
04500004 - 04507FE7	BCSR1				4 Byte
04500008 - 04507FEB	BCSR2				4 Byte
0450000C - 04507FEF	BCSR3				4 Byte
04500010 - 04507FF3	BCSR4				4 Byte
04500014 - 04507FF7	BCSR5				4 Byte
04500018 - 04507FFB	BCSR6				4 Byte
0450001C - 04507FFF	BCSR7				4 Byte
04508000 - 045FFFFF	Empty Space			-	~1 MByte
04600000 - 04607FFF[b]	ATM UNI Proc. Control	PMC5350 M/P I/F		8	32 KByte
04608000 - 046FFFFF	Empty Space			-	~1 MByte
04700000[c] - 0471FFFF	MPC8266 Internal MAP[d]			32	128 KByte
04720000 - 0472FFFF	Empty Space			-	64 KByte
04730000 - 04737FFF	PCI Interrupt Controller			32	32 KByte

Table 3-1. MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device

Address Range	Memory Type	Device Name			Port Size	Memory Size
04738000 - 047FFFFF	Empty Space				-	~800 KByte
04800000 - 04FFFFFF	PCI Memory	Agents PIMMR (via PCI Direct)				~ 8 MByte
05000000 - 7FFFFFFF	Empty Space	Tool Board is located at 60000000 and 70000000				~ 2 GByte
80000000 - BFFFFFFF	PCI Memory	PCI Agents GPL WIndows			32	1 Gbyte
C0000000 - C1FFFFFF	Empty Space				-	32 MByte
C2000000[e] - C2007FFF	E^2PROM	ATMEL AT28HC64B			8	32 KByte
C2008000 - FDFFFFFF	Empty Space					~1 GByte
FE000000[f] - FEFFFFFF	Flash SIMM			32M SIMM - SM73288	32	32 MByte
FF000000 - FF7FFFFF			16M SIMM - SM73248			
FF800000 - FFFFFFFF		8M SIMM - SM73228				

a. The device appears repeatedly in multiples of its port-size (in bytes) X depth. E.g., BCSR0 appear at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.
b. The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size tha may be controlled by the GPCM is 32 KBytes.
c. Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.
d. Refer to the MPC8266 User's Manual for complete description of the internal memory map.
e. An 8 Kbyte device is used (16 Kbyte and 32 Kbyte devices can also be used) so it appears repeatedly in 8Kbyte multiples starting from C2000000.
f. Set by hard-reset configuration.

Table 3-2. MPC8266ADS-PCI Memory Map - E^2PROM as Boot Device

Address Range	Memory Type	Device Name		Port Size	Memory Size
00000000 - 00FFFFFF	SDRAM DIMM	SDCUV6482 (16 MByte)	SDC8UV6484 (64 MByte)	64	64 MByte
01000000 - 03FFFFFF					

Table 3-2. MPC8266ADS-PCI Memory Map - E^{2}PROM as Boot Device

Address Range	Memory Type	Device Name	Port Size	Memory Size
04000000 - 044FFFFF	Empty Space		-	5 MByte
04500000 - 04507FFF	BCSR(0:7)[a]		32	32 KByte
04500000 - 04507FE3	BCSR0			4 Byte
04500004 - 04507FE7	BCSR1			4 Byte
04500008 - 04507FEB	BCSR2			4 Byte
0450000C - 04507FEF	BCSR3			4 Byte
04500010 - 04507FF3	BCSR4			4 Byte
04500014 - 04507FF7	BCSR5			4 Byte
04500018 - 04507FFB	BCSR6			4 Byte
0450001C - 04507FFF	BCSR7			4 Byte
04508000 - 045FFFFF	Empty Space		-	~1 MByte
04600000 - 04607FFF[b]	ATM UNI Proc. Control	PMC5350 M/P I/F	8	32 KByte
04608000 - 046FFFFF	Empty Space		-	~1 MByte
04700000[c] - 0471FFFF	MPC8266 Internal MAP[d]		32	128 KByte
04720000 - 0472FFFF	Empty Space		-	64 KByte
04730000 - 04737FFF	PCI Interrupt Controller		32	32 KByte
04738000 - 047FFFFF	Empty Space		-	~800 KByte
04800000 - 04FFFFFF	PCI Memory	Agents PIMMR (via PCI Direct)		~ 8 MByte
05000000 - 7FFFFFFF	Empty Space	Tool Board is located at 60000000 and 70000000		~ 2 GByte

Table 3-2. MPC8266ADS-PCI Memory Map - E^2PROM as Boot Device

Address Range	Memory Type	Device Name			Port Size	Memory Size
80000000 - BFFFFFFF	PCI Memory	PCI Agents GPL WIndows			32	1 Gbyte
C0000000 - C1FFFFFF	Empty Space				-	32 MByte
C2000000 - C2FFFFFF	Flash SIMM			32M SIMM - SM73288	32	32 MByte
C3000000 - C37FFFFF			16M SIMM - SM73248			
C3800000 - C3FFFFFF		8M SIMM - SM73228				
C4000000 - FFFFDFFF	Empty Space					~1 GByte
FFF00000[e] - FFFFFFFF	E^2PROM	ATMEL AT28HC64B			8	32 KByte

a. The device appears repeatedly in multiples of its port-size (in bytes) X depth. E.g., BCSR0 appears at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.
b. The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size that may be controlled by the GPCM is 32 KBytes.
c. Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.
d. Refer to the MPC8266 User's Manual for complete description of the MPC8266's internal memory map.
e. An 8 Kbyte device is used (16 Kbyte and 32 Kbyte devices can also be used) so it appears repeatedly in 8Kbyte multiples starting from FFF00000.

3.4 MPC8266 Register Programming

The MPC8266 provides the following functions on the MPC8266ADS-PCI:

1. System functions which include:
   - PPC Bus SDRAM Controller
   - Local Bus Host to PCI Bridge
   - Chip Select generator.
2. Communication functions which include:
   - ATM SAR
   - Fast Ethernet controller.
   - UART for terminal or host computer connection.

The internal registers of the MPC8266 must be programmed after Hard reset as described in the following paragraphs. The addresses and programming values are in **Hexadecimal** base.

For more information on the following initializations, see the MPC8266 User's Manual.

3.4.1 System Initializations

The Power-On Reset Configuration word is set in the BCSR or FLASH or in the E^2PROM. There are two configuration words - one for the BCSR and FLASH (when it is assigned to $\overline{CS0}$) and the other to the E^2PROM (when it is assigned to $\overline{CS0}$). The two configurations are detailed in Table 3-3. and Table 3-4. respectively.

Table 3-3. BCSR/FLASH Power On Reset Configuration[a]

Flash Address [hex]	Init Value[hex]	Description
0	0C / (1C[b])	Internal arbitration, Internal memory controller, Core enabled, Single MPC8266 (60X Bus mode[b]), 32 Bit boot port size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master.
8	B2	L2cache signals configured as BADDRx lines, DP(1:7) configured as L2 cache I/F and IRQ(6:7),Initial internal space @ 0x0F000000
10	36	Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ7}$ and $\overline{CS11}$ as $\overline{CS11}$.
18	60	$\overline{CS10}$ configured as $\overline{BCTL1}$, PCI autoload from FLASH

a. Programmed into the Flash (E^2PROM) memory in addresses 0x0, 0x8, 0x10 & 0x18
b. With L2 Cache

Table 3-4. E^2PROM Power On Reset Configuration[a]

EEPROM Address [hex]	Init Value[hex]	Description
0	04 / (14[b])	Internal arbitration, Internal memory controller, Core enabled, Single MPC8266 (60X Bus mode[b]), 8 Bit Boot size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master.
8	B2	L2cache signals configured as BADDRx lines, DP(1:7) configured as L2 cache I/F and IRQ(6:7),Initial internal space @ 0x0F000000
10	36	Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ7}$ and $\overline{CS11}$ as $\overline{CS11}$.
18	60	$\overline{CS10}$ configured as $\overline{BCTL1}$, PCI autoload from EEPROM

a. Programmed into the E^2PROM in addresses 0x0, 0x8, 0x10 & 0x18

b. With L2 Cache

Table 3-5. SIU REGISTERS' PROGRAMMING

Register	Init Value[hex]	Description
RMR	0001	Check-Stop Reset enabled.
IMMR	04700000	Internal space @ 0x047000000
SYPCR	FFFFFFC3	Software watchdog timer count - FFFF, Bus-monitor timing FF, PPC Bus-monitor - Enabled, Local Bus-monitor - Enabled, S/W watch-dog - disabled, S/W watch-dog (if enabled) causes reset, S/W watch-dog (if enabled) - prescaled.
BCR	004C0000 (88444000[a])	Single MPC8266 (60X Bus mode[a]), 0 wait-states on address tenure, No L2Cache (L2Cache assumed[a]), 1 clock hit delay (when L2cache available), 1-level Pipeline depth, Extended transfer mode enabled for PCC, Extended transfer mode disabled for Local Buses, Odd parity for PPC & Local Buses, External Master delay enabled, Internal space responds as 64 bit slave for external master (not relevant for this application).

a. With L2 Cache

3.4.2 Memory Controller Registers Programming

The memory controller on the MPC8266ADS-PCI is initialized to 66 MHz operation, i.e., registers' programming is based on 66 MHZ timing calculation. There are two possible initializations for the memory controller:

- Flash SIMM is assigned to $\overline{CS0}$ and E^2PROM is assigned to $\overline{CS4}$.
- Flash SIMM is assigned to $\overline{CS4}$ and E^2PROM is assigned to $\overline{CS0}$.

Both options are shown in Table 3-6.and Table 3-7.

Table 3-6. Memory Controller Initializations For 66Mhz - FLASH as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR0	SM73228XG1JHBG0 by Smart Modular Tech.	PPC	FF801801	Base at FF800000, 32 bit port size, no parity, GPCM
	SM73248XG2JHBG0 by Smart Modular Tech.		FF001801	Base at FF000000, 32 bit port size, no parity, GPCM
	SM73288XG4JHBG0 by Smart Modular Tech.		FE001801	Base at FE000000, 32 bit port size, no parity, GPCM
OR0	SM73228XG1JHBG0 by Smart Modular Tech.		FF800836	8MByte block size, CS early negate, 6 w.s., Timing relax
	SM73248XG2JHBG0 by Smart Modular Tech.		FF000836	16MByte block size, CS early negate, 6 w.s., Timing relax
	SM73288XG4JHBG0 by Smart Modular Tech.		FE000836	32MByte block size, CS early negate, 6 w.s., Timing relax
BR1	BCSR	PPC	04501801	Base at 04500000, 32 bit port size, no parity, GPCM
OR1			FFFF8010	32 KByte block size, all types access, 1 w.s.
BR2	All SDRAM DIMM Supported	PPC	00000041	Base at 0, 64 bit port size, no parity, Sdram machine 1
OR2	SDC2UV6482C-84 by Fujitsu		FF000C80	16MByte block size, 2 banks per device, row starts at A9, 11 row lines, internal bank interleaving allowed, normal AACK operation
	SDC8UV6484C-84 by Fujitsu		FC002CC0	64MByte block size, 4 banks per device, row starts at A9, 12 row lines, internal bank interleaving allowed, normal AACK operation
BR3	Depends on the SDRAM DIMM.	PPC	TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
OR3	Depends on the SDRAM DIMM.		TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
BR4	E^2PROM	PPC	C2000801	Base at C2000000, 8 bit port size, write protect disabled, no parity, GPCM
OR4	AT28HC64B-70JC by Atmel		FFFF8846	32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 4 w.s., Timing relax
BR5	PM5350 - ATM UNI	PPC	04600801	Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus.
OR5			FFFF8E36	32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read.

Table 3-6. Memory Controller Initializations For 66Mhz - FLASH as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR8	PCI Interrupt Controller	PPC	04731801	Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus.
OR8			FFFF8010	32 KByte block size, all types access, 1 w.s.

Table 3-7. Memory Controller Initializations For 66Mhz - E^2PROM as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR0	E^2PROM	PPC	FFF00801	Base at FFFFE000, 8 bit port size, write protect disabled, no parity, GPCM
OR0	AT28HC64B-70JC by Atmel		FFFF8846	32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 4 w.s., Timing relax
BR1	BCSR	PPC	04501801	Base at 04500000, 32 bit port size, no parity, GPCM
OR1			FFFF8010	32 KByte block size, all types access, 1 w.s.
BR2	All SDRAM DIMM Supported	PPC	00000041	Base at 0, 64 bit port size, no parity, Sdram machine 1
OR2	SDC2UV6482C-84 by Fujitsu		FF000C80	16MByte block size, 2 banks per device, row starts at A9, 11 row lines, internal bank interleaving allowed, normal AACK operation
	SDC8UV6484C-84 by Fujitsu		FC002CC0	64MByte block size, 4 banks per device, row starts at A9, 12 row lines, internal bank interleaving allowed, normal AACK operation
BR3	Depends on the SDRAM DIMM.	PPC	TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
OR3	Depends on the SDRAM DIMM.		TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.

Table 3-7. Memory Controller Initializations For 66Mhz - E^{2}PROM as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR4	SM73228XG1JHBG0 by Smart Modular Tech.	PPC	C3801801	Base at C3800000, 32 bit port size, no parity, GPCM
	SM73248XG2JHBG0 by Smart Modular Tech.		C3001801	Base at C3000000, 32 bit port size, no parity, GPCM
	ASM73288XG4JHBG0 by Smart Modular Tech.		C2001801	Base at C2000000, 32 bit port size, no parity, GPCM
OR4	SM73228XG1JHBG0 by Smart Modular Tech.		FF800836	8MByte block size, CS early negate, 6 w.s., Timing relax
	SM73248XG2JHBG0 by Smart Modular Tech.		FF000836	16MByte block size, CS early negate, 6 w.s., Timing relax
	SM73288XG4JHBG0 by Smart Modular Tech.		FE000836	32MByte block size, CS early negate, 6 w.s., Timing relax
BR5	PM5350 - ATM UNI	PPC	04600801	Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus.
OR5			FFFF8E36	32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read.
BR8	PCI Interrupt Controller	PPC	04731801	Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus.
OR8			FFFF8010	32 KByte block size, all types access, 1 w.s.

Table 3-8. Memory Controller Initializations For 66Mhz

Reg.	Device Type	Bus	Init Value [hex]	Description
PSDMR	SDC2UV6482C-84 (16 MByte)	PPC Single MPC8266 Bus Mode	416EB452	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C2AAB452	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A19 on BNKSEL2[a], A8 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
	SDC8UV6484C-84 (64 MByte)		412EB452	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(13 - 15) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C372B452	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A(16:17) on BNKSEL(1:2)[b], A6 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.

Table 3-8. Memory Controller Initializations For 66Mhz

Reg.	Device Type	Bus	Init Value [hex]	Description
	SDC2UV6482C-84 (16 MByte) **DIMM_SIZE** in BCSR0 should be **Cleared**.	PPC 60X Bus Mode	416EB45A **PBI** in BCSR0 should be **Cleared**	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C2AAB45A **PBI** in BCSR0 should be **Set**	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A19 on BNKSEL2[a], A8 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
	SDC8UV6484C-84 (64 MByte) **DIMM_SIZE** in BCSR0 should be **Set**.		412EB45A **PBI** in BCSR0 should be **Cleared**	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C372B45A **PBI** in BCSR0 should be **Set**	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A(16:17) on BNKSEL(1:2)[b], A6 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
LSDMR		-	0	No SDRAM on Local Bus which is used for PCI Bus ONLY.
PSRT	All PPC Bus Sdram Supported	PPC	21	Divide MPTPR output by 34 (PSRT +1) Generates refresh every 13.4 µsec, while 16 µsec required. Therefore is refresh redundancy of 5.4 msec throughout full SDRAM refresh cycle which completes in 27.4 msec. I.e., Application s/w may withhold the bus upto app. 5.4 msec in a 32.8 msec period, without jeopardizing the contents of the ppc bus SDRAM DIMM.
LSRT		-	0	No SDRAM on Local Bus which is used for PCI Bus ONLY.
MPTPR	All SDRAMs on board		1900	Divide Bus clock by 26 (MPTPR+1) (decimal)

a. Although this BSMA value corresponds to A(17:19) on BKSEL(0:2), when PBI is set, only the relevant BKSEL lines, according to number of internal SDARM banks, are VALID, in this case BKSEL2.
b. Although this BSMA value corresponds to A(15:17) on BKSEL(0:2), when PBI is set, only the relevant BKSEL lines, according to number of internal SDARM banks, are VALID, in this case BKSEL(1:2)

4

Functional Description

In this chapter the various modules combining the MPC8266ADS-PCI are described to their design details.

4.1 Reset & Reset - Configuration

There are several reset sources on the MPC8266-MB:

1. Power On Reset
2. Manual Hard-Reset
3. Manual Soft-Reset
4. PCI bus reset
5. MPC8266 Internal Sources. (See also the VOYAGER U/M)

4.1.1 Power - On Reset

The power on reset to the MPC8266 initializes the processor state after power up. A dedicated logic, using Seiko S-80728AN-DR-T1, which is a voltage detector of 2.8V +/- 2.4%, asserts $\overline{\text{PORESET}}$ input to the MPC8266 for a period of ~2.5sec. This time period is long enough to cover also the VDDL stabilization, powered by a different voltage regulator. It is assumed that the stabilization time for both linear regulators (see also **Section 6.1 Power Supply**) are about the same. Power-On-Reset may be generated manually as well by an on-board dedicated push-button (S1) or by the ATX chassis RESET button. Power-On Reset can also be generated by the JTAG logic, which is integrated with BCSR.

4.1.1.1 Power - On Reset Configuration

At the end of Power - On reset sequence, MODCK(1:3) are sampled by the MPC8266 to configure the various clock modes of the MPC8266 (core, cpm, bus, PCI...). Selection between the MODCK(1:3) combination options is done by means of dip-switches (Section 2.3.3) on the mother board while PCI_MODCKH(0:3) are obtained from the relevant dedicated pins (by means of dip-switches - Section 2.3.6) when the MPC8266 is in active PCI mode (determined by the state of $\overline{\text{PCI\_MODE}}$ pin). If the PCI is set to be inactive, the MODCKH(0:3) bits are obtained from the Hard Reset Configuration Word in the Flash or in the E^2PROM (depends on who is the boot device).

The configuration master is determined upon the rising edge of $\overline{\text{PORST}}$, according to the state of $\overline{\text{RSTCONF}}$ (Section 2.3.5) signal, driven low on this board, to set the MPC8266 as a configuration master.

After power-on reset negates, the hard-reset sequence starts, during which, many other different

options are configured (see **Section 4.1.2.4 "Hard Reset Configuration" on page 41**), among these options, are additional clock configuration bits - PCI_MODCKH(0:3) - the most significant bits of the MODCK field, which determine additional options for the clock generator. Although these bits are sampled whenever the hard-reset sequence is entered, they are **influential only once - after power-on reset**. If a hard reset sequence is entered later, MODCKH(0:3), although sampled, are don't care.

The PCI_MODCK signal, which is sampled concurrently with the PCI_MODCK(0:3) pins, determines the PCI bus clock frequency (see Section 2.3.7). When set high, it divides the PCI bus frequency by two. When reset low, the PCI bus frequency is as determined by the MODCK(1:3) and PCI_MODCKH(0:3) signals.

4.1.2 Hard Reset

Hard-Reset may be generated on the ADS by the following sources:

1. COP/JTAG Port
2. Manual Hard reset.
3. MPC8266's internal sources.

Hard-Reset, when generated, causes the MPC8266 to reset all its internal hardware except for PLL logic, re-acquires the Hard-reset configuration from its current source, and jumps to the Reset vector in the exception table. Since hard-reset resets also the refresh logic for dynamic RAMs, their content is lost as well.

$\overline{\text{HRESET}}$ when asserted, is extended internally by the MPC8266 for additional 512 bus clock cycles at the end of which, the MPC8266 waits for 16 bus clock cycles and then, re-checks the state of the $\overline{\text{HRESET}}$ line.

$\overline{\text{HRESET}}$ is an open-drain signal and must be driven with an open-drain gate by which ever external source is driving it. Otherwise, contention will occur over that line, which might cause permanent damage to either board logic and/or to the MPC8266 itself.

4.1.2.1 COP/JTAG Port Hard - Reset

To provide convenient hard-reset capability for a COP/JTAG controller, $\overline{\text{HRESET}}$ line appears at the COP/JTAG port connector. The COP/JTAG controller may directly generate hard-reset by asserting (low) this line.

4.1.2.2 Manual Hard Reset

To allow run-time Hard-reset, when the COP controller is disconnected from the MPC8266ADS-PCI and to support resident debuggers, manual Hard is facilitated. Depressing both Soft-Reset (S3) and ABORT (S2) buttons asserts the $\overline{\text{HRESET}}$ pin of the MPC8266, generating a HARD RESET sequence.

Since the $\overline{\text{HRESET}}$ line may be driven internally by the MPC8266, it must be driven to the MPC8266 with an open-drain gate. If off-board H/W connected to the MPC8266ADS-PCI is to drive $\overline{\text{HRESET}}$ line, then it should do so with an open-drain gate, this, to avoid contention over this line.

When Hard Reset is generated, the MPC8266 is reset in a destructive manner, i.e., the hard reset

configuration is re-sampled and all registers (except for the PLL's) are reset, including memory controller registers - reset of which results in a loss of dynamic memory contents.

To save on board's real-estate, this button is not a dedicated one, but is shared with the Soft-Reset button and the ABORT button - when both are depressed, Hard Reset is generated.

4.1.2.3 Internal Sources Hard - Reset

The MPC8266 has internal sources which generate Hard Reset. Among these sources are:

1. Loss of Lock Reset. When one of the PLLs (Core, CPM), is out of lock, hard-reset is generated.
2. Check-Stop Reset. When the core enters a Check-Stop state from some reason, hard-reset may be generated, depended on CSRE bit in the RMR.
3. Bus Monitor Reset. When the bus monitor is enabled and a bus cycle is not terminated, hard-reset is generated.
4. S/W Watch Dog Reset. When the S/W watch-dog is enabled, and application s/w fails to perform its reset routine, it will generate hard - reset.
5. COP/JTAG Reset (Internal). Hard reset may be forced by driving the $\overline{\text{HRESET}}$ line via the external pin's scan chain. Not useful for run time.

In general, the MPC8266 asserts a reset line HARD or SOFT for a period 512 clock cycles after a reset source has been identified. A hard reset sequence is followed by a soft reset sequence.

4.1.2.4 Hard Reset Configuration

When Hard-Reset is applied to the MPC8266 (externally as well as internally), it samples the Hard-Reset configuration word. This configuration may be taken from an internal default, in case $\overline{\text{RSTCONF}}$ is negated during $\overline{\text{HRESET}}$ asserted or taken from the Flash [1]/E^2PROM/BCSR (MS 8 bits of the data bus) in case $\overline{\text{RSTCONF}}$ signal is asserted along with $\overline{\text{HRESET}}$. The default configuration word can be taken from the E^2PROM/BCSR in case the Flash has been tampered with. The selection between the BCSR, FLASH and the E^2PROM as the source of the default configuration word is determined by a dedicated dip-switch (see Section 2.3.5) and a jumper (see Section 2.3.4).

During hard reset sequence, the configuration master[2] reads the Flash (or E^2PROM or BCSR) memory at addresses 0, 8, 0x18, 0x20,... a byte each time, to assemble the 32 bit configuration word. A total of 64 bytes of data is read from D(0:7) to acquire 8 full configuration words for system that may have upto 8 MPC8266 chips.

The configuration word for a single[3] MPC8266 is stored in the Flash memory SIMM, in the E^2PROM or as default in the BCSR, while the other seven words are not initialized, as there are no additional MPC8266 on the MPC8266ADS-PCI. The default configuration word is shown in Table 4-1. for the FLASH and in Table 4-2. for the E^2PROM. PCI module configuration is 256

1. In general, from any device residing on $\overline{\text{CS0}}$.
2. In general, The MPC8266 for which $\overline{\text{RSTCONF}}$ is asserted along with $\overline{\text{PORST}}$ asserted or in particular, the MPC8266 residing on the MPC8266ADS-PCI.
3. Although the MPC8266 as configuration master reads 8 configuration words, only the 1'st configuration word is influential.

Bytes long and should start at address 0x100.

There are four possible configuration words:

- MPC8266ADS-PCI without L2 Cache - FLASH/BCSR is the boot device. $\overline{CS0}$ is assigned to the FLASH and $\overline{CS4}$ is assigned to the E^2PROM.
- MPC8266ADS-PCI without L2 Cache - E^2PROM is the boot device. $\overline{CS0}$ is assigned to the E^2PROM and $\overline{CS4}$ is assigned to the FLASH.
- MPC8266ADS-PCI with L2 Cache - FLASH is the boot device. $\overline{CS0}$ is assigned to the FLASH and $\overline{CS4}$ is assigned to the E^2PROM.
- MPC8266ADS-PCI with L2 Cache - E^2PROM is the boot device. $\overline{CS0}$ is assigned to the E^2PROM and $\overline{CS4}$ is assigned to the FLASH.

.

Table 4-1. BCSR/FLASH Hard Reset Configuration Word

Field	Data Bus Bits	Prog Value [Bin]	Implication	Offset In Flash [Hex]	Value [Hex]
ERB	0	'0'	Internal Arbitration Selected.	0	0C / 1C[a]
EXMC	1	'0'	Internal Memory Controller. $\overline{CS0}$ active at system boot.		
CDIS	2	'0'	Core Enabled.		
EBM	3	'0' /'1'	'0' - Single MPC8266 Mode for boards without L2Cache '1' - 60X Bus Mode[a] for boards with L2Cache		
BPS	4:5	11	32 Bit Boot Port Size		
CIP	6	'0'	Sets Core Initial Prefix MSR[IP]=1, so that system exception table is placed at address 0xFFF00100 regardless of FLASH memory size		
ISPS	7	'0'	64 bit internal space for external master accesses. In fact don't care on this board since external master is not supported.		

Table 4-1. BCSR/FLASH Hard Reset Configuration Word

Field	Data Bus Bits	Prog Value [Bin]	Implication	Offset In Flash [Hex]	Value [Hex]
L2CPC	8:9	‘10’	CI/BADDR(29)/IRQ2 selected as BADDR(29) WT/BADDR(30)/IRQ3 selected as BADDR(30) L2_HIT/IRQ4 selected as unassigned CPU_BG/BADDR(31)/IRQ5 as BADDR(31)	8	B2
DPPC	10:11	‘11’	Data Parity Pin configuration as: DP0 as EXT_BR2 DP1 as EXT_BG2 DP2 as EXT_DBG2 DP3 as EXT_BR3 DP4 as EXT_BG3 DP5 as EXT_DBG3 DP6 as IRQ6 DP7 as IRQ7		
Reserved	12	’0’	Reserved.		
ISB	13:15	’010’	IMMR initial value 0x0F000000, i.e., the internal space resides initially at this address.		
BMS	16	’0’	Boot memory (Flash) at 0xFE000000.	10	36
BBD	17	’0’	ABB/IRQ2 pin is ABB DBB/IRQ3 pin is DBB		
MMR	18:19	’11’	Mask Masters Requests. Boot Master is PCI.		
LBPC	20:21	’01’	Local Bus pins function as PCI bus.		
APPC	22:23	’10’	MODCK1/AP(1)/TC(0) functions as BKSEL0 MODCK2/AP(2)/TC(1) functions as BKSEL1 MODCK3/AP(3)/TC(2) functions as BKSEL2 IRQ7~/APE~ functions as IRQ7~ CS11~/AP(0) functions as CS11~		
CS10PC	24:25	’01’	CS10~/BCTL1/DBG_DIS~ functions as BCTL1	18	45
ALD_EN	26	’0’	PCI Auto Load Enable. When high, PCI Bridge Configuration is done automatically from the FLASH/E^{2}PROM (CPM is configuration master - PPC core should be disabled) right after the Hard Configuration Word. When low, the PPC Core should configure the PCI Bridge.		
Reserved	27	’0’	Reserved.		
MODCK_HI[b]	28:31	‘0101’	Determines the Core’s frequency out of power-up reset. Core starts at 166MHz. Actually, not relevant when the PCI is active since the PCI_MODCK(0:3) take presidency.		

a. For L2 Cache Boards.
b. Applies only ONCE after power-up reset.

Table 4-2. E^{2}PROM Hard Reset Configuration Word

Field	Data Bus Bits	Prog Value [Bin]	Implication	Offset In Flash [Hex]	Value [Hex]
ERB	0	’0’	Internal Arbitration Selected.	0	04 / 14[a]
EXMC	1	’0’	Internal Memory Controller. $\overline{CS0}$ active at system boot.		
CDIS	2	’0’	Core Enabled.		
EBM	3	’0’ /’1’	‘0’ - Single MPC8266 Mode for boards without L2Cache ‘1’ - 60X Bus Mode[a] for boards with L2Cache		
BPS	4:5	‘01’	8 Bit Boot Port Size		
CIP	6	’0’	Sets Core Initial Prefix MSR[IP]=1, so that system exception table is placed at address 0xFFF00100 regardless of FLASH memory size		
ISPS	7	‘0’	64 bit internal space for external master accesses. In fact don’t care on this board since external master is not supported.		
L2CPC	8:9	‘10’	$\overline{CI}$/BADDR(29)/$\overline{IRQ2}$ selected as BADDR(29) $\overline{WT}$/BADDR(30)/$\overline{IRQ3}$ selected as BADDR(30) $\overline{L2\_HIT}$/$\overline{IRQ4}$ selected as unassigned $\overline{CPU\_BG}$/BADDR(31)/$\overline{IRQ5}$ as BADDR(31)	8	B2
DPPC	10:11	‘11’	Data Parity Pin configuration as: DP0 as EXT_BR2 DP1 as EXT_BG2 DP2 as EXT_DBG2 DP3 as EXT_BR3 DP4 as EXT_BG3 DP5 as $\overline{EXT\_DBG3}$ DP6 as $\overline{IRQ6}$ DP7 as $\overline{IRQ7}$		
Reserved	12	’0’	Reserved.		
ISB	13:15	’010’	IMMR initial value 0x0F000000, i.e., the internal space resides initially at this address.		

Table 4-2. E^{2}PROM Hard Reset Configuration Word

Field	Data Bus Bits	Prog Value [Bin]	Implication	Offset In Flash [Hex]	Value [Hex]
BMS	16	'0'	Boot memory (E^{2}PROM) at 0xFE000000.	10	36
BBD	17	'0'	$\overline{ABB}$/$\overline{IRQ2}$ pin is $\overline{ABB}$ DBB/IRQ3 pin is $\overline{DBB}$		
MMR	18:19	'11'	Mask Masters Requests. Boot Master is PCI.		
LBPC	20:21	'01'	Local Bus pins function as PCI bus.		
APPC	22:23	'10'	MODCK1/AP(1)/TC(0) functions as BKSEL0 MODCK2/AP(2)/TC(1) functions as BKSEL1 MODCK3/AP(3)/TC(2) functions as BKSEL2 IRQ7~/APE~ functions as IRQ7~ CS11~/AP(0) functions as CS11~		
CS10PC	24:25	'01'	CS10~/BCTL1/DBG_DIS~ functions as BCTL1	18	45
ALD_EN	26	'0'	PCI Auto Load Enable. When high, PCI Bridge Configuration is done automatically from the FLASH/E^{2}PROM (CPM is configuration source - PPC core should be disabled) right after the Hard Configuration Word. When low, the PPC Core should configure the PCI Bridge.		
Reserved	27	'0'	Reserved.		
MODCK_HI[b]	28:31	'0101'	Determines the Core's frequency out of power-up reset. Core starts at 166MHz. Actually, not relevant when the PCI is active since the PCI_MODCK(0:3) take presidency.		

a. For L2 Cache Boards.
b. Applies only ONCE after power-up reset.

The PCI configuration registers which are set at Hard-Reset sequence are shown in Figure 4-1.

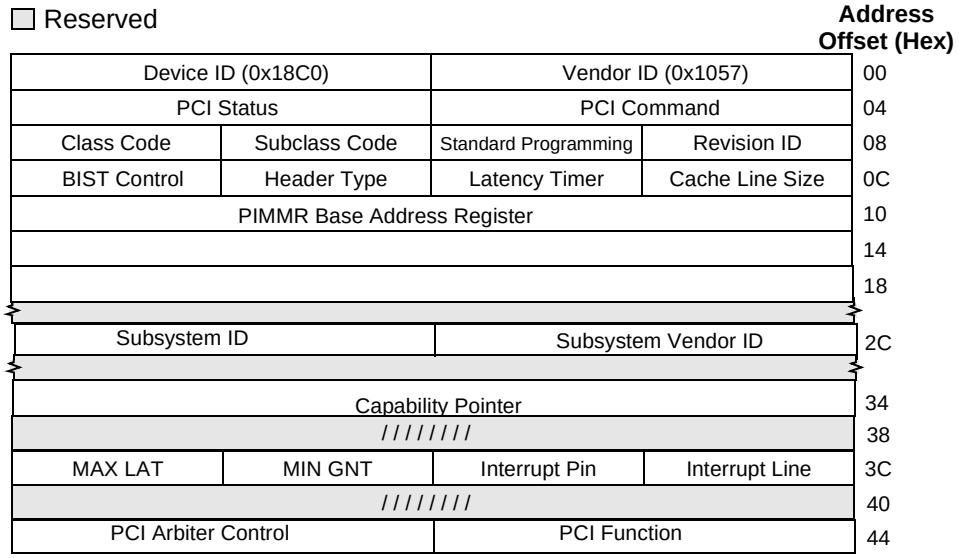


Figure 4-1. PCI Host Configuration Registers

4.1.3 Soft Reset

Soft - Reset may be generated on the board from the below sources:

1. COP/JTAG Port
2. Manual Soft Reset
3. Internal MPC8266 source.

Soft-Reset, when generated, causes the MPC8266 to reset its internal logic, while keeping its hard-reset configuration and memory controller setup and then jumping to the Reset vector in the exception table. Since soft-reset does not reset the refresh logic for dynamic RAMs, their contents is preserved.

$\overline{\text{SRESET}}$ when asserted, is extended internally by the MPC8266 for an additional 512 bus clock cycles at the end of which, the MPC8266 waits for 16 bus clock cycles and then, re-checks the state of the $\overline{\text{SRESET}}$ line.

$\overline{\text{SRESET}}$ is an open-drain signal and must be driven with an open-drain gate by every external source driving it. Otherwise, contention will occur over that line, which might cause permanent damage to either the boards' logic and / or to the MPC8266 itself.

4.1.3.1 COP/JTAG Port Soft - Reset

To provide convenient soft-reset capability for a COP/JTAG controller, $\overline{\text{SRESET}}$ line appears at the COP/JTAG port connector - P3. The COP/JTAG controller may directly generate Soft-reset by asserting (low) this line.

4.1.3.2 Manual Soft Reset

To allow run-time Soft-reset, when the COP controller is disconnected from the MPC8266ADS-PCI and to support resident debuggers, a Soft Reset push-button is provided. When the Soft Reset

push-button is depressed, the $\overline{\text{SRESET}}$ line is asserted to the MPC8266, generating a Soft Reset sequence.

Since the $\overline{\text{SRESET}}$ line may be driven internally by the MPC8266, it must be driven by an open-drain gate, to avoid contention over that line. If off-board H/W connected to the MPC8266ADS-PCI is to drive $\overline{\text{SRESET}}$ line, then, it should do so with an open-drain gate, this, to avoid contention over this line.

4.1.3.3 Internal Sources Soft - Reset

The only internal Soft-reset source is the COP/JTAG soft-reset, which may be generated using Public JTAG instructions to shift active-value (‘0’) to the $\overline{\text{SRESET}}$ pin via the boundary scan chain. This is not useful for run time.

4.1.4 PCI Bus Reset

The PCI Module in the MPC8266 can generate a reset signal dedicated for PCI devices which reside on the PCI bus. This is a reset to the PCI bus which is initiated by the PCI bus Host - the MPC8266 on this board. This reset can also be initiated by a Soft PCI Reset by setting a dedicated bit in a PCI control register (consult the MPC8266 User Manual for details).

4.2 Local Interrupter

There are external interrupts which are applied to the MPC8266 via its interrupt controller:

1. ABORT (NMI)
2. ATM UNI interrupt
3. Fast Ethernet PHY Interrupt
4. PCI interrupt

4.2.1 ABORT Interrupt

The ABORT (NMI), is generated by a push-button. When this button is depressed, the $\overline{\text{IRQ0}}$ input to the MPC8266 is asserted. The purpose of this type of interrupt, is to support the use of resident debugger if any is made available to the board. This interrupt is enabled by setting the MSR[EE] bit.

To support external (off-board) generation of an NMI, the $\overline{\text{IRQ0}}$ line, is driven by an open-drain gate. This allows for an external h/w, to also drive this line. If an external h/w indeed does so, it is compulsory that $\overline{\text{IRQ0}}$ is driven by an open-drain (or open-collector) gate.

4.2.2 ATM UNI Interrupt

To support ATM UNI (User Network I/F) event report by means of interrupt, the interrupt output of the UNI (INTB) is connected to $\overline{\text{IRQ7}}$ line of the MPC8266. This $\overline{\text{IRQ7}}$ input is shared with the Fast Ethernet PHY Interrupt. Since INTB of the UNI is an open-drain output, it is possible to connect additional (on and off-board) interrupt requesters on the same $\overline{\text{IRQ7}}$, provided that they drive $\overline{\text{IRQ7}}$ with open-drain gate as well. When an interrupt request appears in $\overline{\text{IRQ7}}$, it is

necessary to check the source of the interrupt whether it's the ATM UNI or the Fast Ethernet PHY.

4.2.3 Fast Ethernet PHY Interrupt

To support fast Ethernet Transceiver event report by means of interrupt, the interrupt output (FDS/MDINT) of the LX970A is connected to $\overline{\text{IRQ7}}$ line of the MPC8266. This $\overline{\text{IRQ7}}$ input is shared with the ATM UNI Interrupt. Since the FDS/MDINT is an open-rain output, it is possible to connect additional (on and off board) interrupt requests on the same $\overline{\text{IRQ7}}$, provided that they drive the $\overline{\text{IRQ7}}$ with an open-drain gate as well.

The FDS/MDINT is a dual functionality signal - on its FDS (Full-Duplex Status) function, it indicates whether the LXT970 is configured to Full/Half Duplex mode, while in its alternate function it serves as the transceivers MDINT active-low output. In order to achieve this functionality, bit 1 in register 17 (17.1) must be SET after the board comes out of Hard Reset. Setting this bit is allowed through the MDIO port of the LXT970, Data of which is driven / sampled by PC9 of the MPC8266 while its Clock is driven by PC10. Failure in doing so, will result in $\overline{\text{IRQ7}}$ pin of the MPC8266, constantly asserted (low).

4.2.4 PCI Interrupt

Each PCI slot can generate up to four interrupts to a total of twelve (3 slot x 4 interupts each). Each PCI expansion board can generate an interrupt at any given time. Since there is only one interrupt input available in the MPC8266, an Interrupt Controller is used. The Interrupt Controller receives all the possible interrupts from the PCI slots and generate one interrupt ($\overline{\text{IRQ6}}$) to the MPC8266.

A simple generic Interrupt Controller is implemented using a CPLD device. The Interrupt Controller is implemented as an Interrupt Register and an Interrupt Mask Register. The Interrupt Controller has its' own dedicated chip-select line ($\overline{\text{CS8}}$). A simple priority scheme is devised to prioritize the interrupts from different slots. The PCI IRQ routing are according to Figure 4-2..

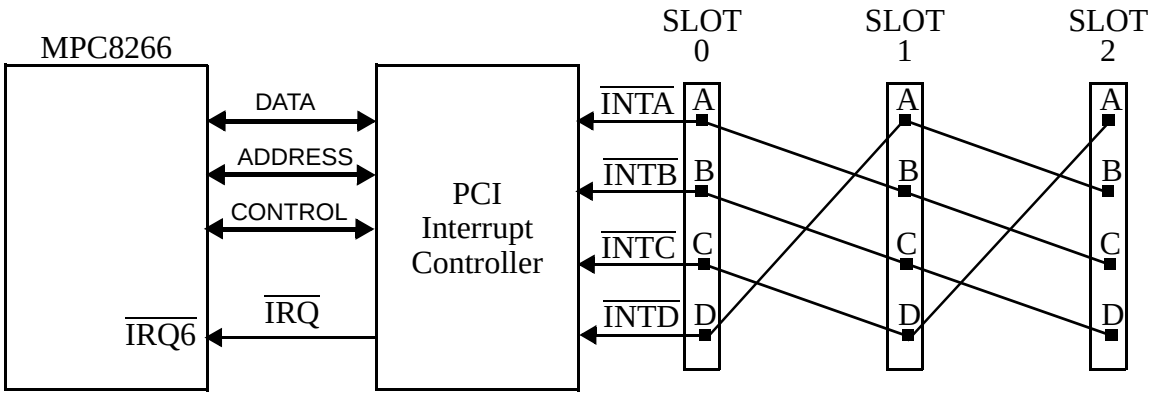


Figure 4-2. PCI Interrupt Routing Scheme

An interrupt request in any of the $\overline{\text{INTx}}$ lines, will set three interrupt bits in the PCI Interrupt

Register (if not masked in the Interrupt Mask Register) since there are three possible interrupt sources for every $\overline{\text{INTx}}$ line. It is up to the user to implement a polling process to verify the real interrupt source (by polling the Interrupt Pending bit in the PCI device) and clear the other two. The PCI Interrupt Register can be read at any time and accessed at offset 0x0 from $\overline{\text{CS8}}$ base address. The description of the PCI Interrupt Register is in Table 4-3..

Table 4-3. PCI Interrupt Register Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0	PCI0_INTA	***PCI Slot 0 $\overline{\text{INTA}}$***. PCI Slot 0 Interrupt A: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
1	PCI0_INTB	***PCI Slot 0 $\overline{\text{INTB}}$***. PCI Slot 0 Interrupt B: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
2	PCI0_INTC	***PCI Slot 0 $\overline{\text{INTC}}$***. PCI Slot 0 Interrupt C: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
3	PCI0_INTD	***PCI Slot 0 $\overline{\text{INTD}}$***. PCI Slot 0 Interrupt D: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
4	PCI1_INTA	***PCI Slot 1 $\overline{\text{INTA}}$***. PCI Slot 1 Interrupt A: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
5	PCI1_INTB	***PCI Slot 1 $\overline{\text{INTB}}$***. PCI Slot 1 Interrupt B: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
6	PCI1_INTC	***PCI Slot 1 $\overline{\text{INTC}}$***. PCI Slot 1 Interrupt C: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
7	PCI1_INTD	***PCI Slot 1 $\overline{\text{INTD}}$***. PCI Slot 1 Interrupt D: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
8	PCI2_INTA	***PCI Slot 2 $\overline{\text{INTA}}$***. PCI Slot 2 Interrupt A: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
9	PCI2_INTB	***PCI Slot 2 $\overline{\text{INTB}}$***. PCI Slot 2 Interrupt B: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
10	PCI2_INTC	***PCI Slot 2 $\overline{\text{INTC}}$***. PCI Slot 2 Interrupt C: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R
11	PCI2_INTD	***PCI Slot 2 $\overline{\text{INTD}}$***. PCI Slot 2 Interrupt D: '0' - no interrupt was requested '1' - an interrupt was requested and waiting to be handled	0	R

Table 4-3. PCI Interrupt Register Description

BIT	MNEMONIC	Function	PON DEF	ATT.
12-31	Reserved	Un-implemented		R/W

Also available is an Interrupt Mask Register which provides the user with the option to mask any of the possible PCI interrupt sources. It can be read or written at any time and accessed at offset 0x4 from $\overline{CS8}$ base address.The description of the PCI Interrupt Mask Register is in Table 4-4..

Table 4-4. PCI Interrupt Mask Register Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0	MPCI0_INTA	***Mask PCI Slot 0 $\overline{INTA}$***. Mask PCI Slot 0 Interrupt A: '0' - interrupt is available '1' - interrupt is masked	0	R/W
1	MPCI0_INTB	***Mask PCI Slot 0 $\overline{INTB}$***. Mask PCI Slot 0 Interrupt B: '0' - interrupt is available '1' - interrupt is masked	0	R/W
2	MPCI0_INTC	***Mask PCI Slot 0 $\overline{INTC}$***. Mask PCI Slot 0 Interrupt C: '0' - interrupt is available '1' - interrupt is masked	0	R/W
3	MPCI0_INTD	***Mask PCI Slot 0 $\overline{INTD}$***. Mask PCI Slot 0 Interrupt D: '0' - interrupt is available '1' - interrupt is masked	0	R/W
4	MPCI1_INTA	***Mask PCI Slot 1 $\overline{INTA}$***. Mask PCI Slot 1 Interrupt A: '0' - interrupt is available '1' - interrupt is masked	0	R/W
5	MPCI1_INTB	***Mask PCI Slot 1 $\overline{INTB}$***. Mask PCI Slot 1 Interrupt B: '0' - interrupt is available '1' - interrupt is masked	0	R/W
6	MPCI1_INTC	***Mask PCI Slot 1 $\overline{INTC}$***. Mask PCI Slot 1 Interrupt C: '0' - interrupt is available '1' - interrupt is masked	0	R/W
7	MPCI1_INTD	***Mask PCI Slot 1 $\overline{INTD}$***. Mask PCI Slot 1 Interrupt D: '0' - interrupt is available '1' - interrupt is masked	0	R/W
8	MPCI2_INTA	***Mask PCI Slot 2 $\overline{INTA}$***. Mask PCI Slot 2 Interrupt A: '0' - interrupt is available '1' - interrupt is masked	0	R/W
9	MPCI2_INTB	***Mask PCI Slot 2 $\overline{INTB}$***. Mask PCI Slot 2 Interrupt B: '0' - interrupt is available '1' - interrupt is masked	0	R/W

Table 4-4. PCI Interrupt Mask Register Description

BIT	MNEMONIC	Function	PON DEF	ATT.
10	MPCI2_INTC	***Mask PCI Slot 2 $\overline{INTC}$***. Mask PCI Slot 2 Interrupt C: '0' - interrupt is available '1' - interrupt is masked	0	R/W
11	MPCI2_INTD	***Mask PCI Slot 2 $\overline{INTD}$***. Mask PCI Slot 2 Interrupt D: '0' - interrupt is available '1' - interrupt is masked	0	R/W
12-31	Reserved	Un-implemented		R/W

4.3 Clock Generator

There are two main clock circuits on board:

1. MPC8266 System Clock
2. PCI Clock

4.3.1 MPC8266 Clock

The MPC8266 requires a single clock source as the main clock source. All MPC8266 60x bus timings are referenced to the main clock input - CLKIN1 (unlike the 8xx family, timings of which, are referenced to CLKOUT signal). The main clock input is in 1:1 ratio to the bus clock, with internal skew elimination (PLL). Use is done with 66MHz 3.3V clock generator (X2), which is connected to a low inter-skew buffer (U17)to split the load between all various clock consumers on both boards.

Special care is taken to isolate and terminate the clock route between the on-board PLL and the MPC8266, this to provide a "clean" clock input for proper operation. The main clock scheme is shown in Figure 4-3.

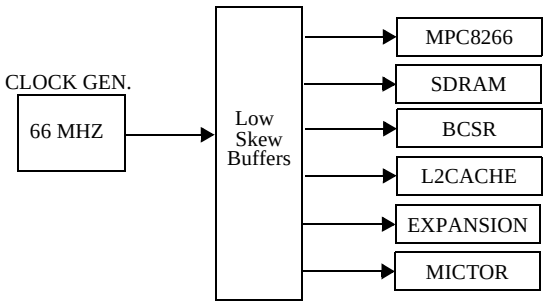


Figure 4-3. Main Clock Generator Scheme

4.3.2 PCI Clock

The PCI bus clock is derived internally from the main clock input CLKIN1. The generated PCI clock is output from a PCI-dedicated PLL (named DLL). That clock output is feeding an on-board low-skew and fast (low propagation delay PLL) clock distributor which distributes the PCI clock to all on-board PCI devices. One of the outputs is fed back to the PCI clock to the MPC8266 through CLKIN2 input. This clock input is driven to the DLL which synchronizes the DLL output clock to the CLKIN2 input clock and thus, maintains low skew between the DLL output and CLKIN2 input. All PCI bus timings are referenced to the CLKIN2 input clock. Special care was taken when the board layout was done to keep all copper traces away from the Clock Distributor outputs at the same lengths, including the output that is fed back to CLKIN2. This is in compliance with the PCI standard to achieve bus synchronization and low skew. The PCI clock scheme is shown in Figure 4-4.

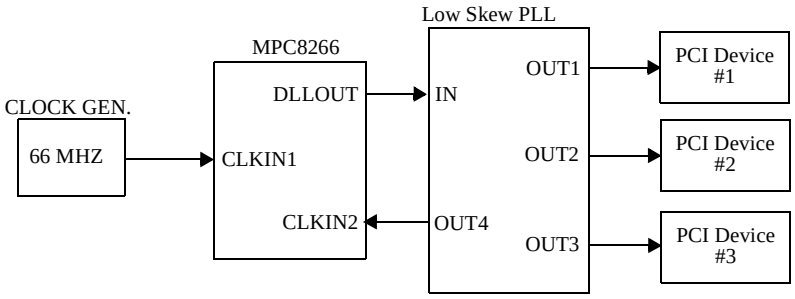


Figure 4-4. PCI Clock Generator Scheme

4.4 Bus Configuration

The MPC8266 may be configured in 2 possible bus modes depending on the presence of L2 cache on board.

1. Single MPC8266 Mode
2. 60X Bus Mode.

4.4.1 Single MPC8266 Mode

When a L2 Cache is not present on the board, the MPC8266 is configured in Single MPC8266 Mode. I.e., assuming only one MPC8266 on the 60x bus, with no support for external master access. This allows for internal address multiplexing to occur which makes the external address multiplexers redundant and therefore not assembled. This improves SDRAM performance.

4.4.2 60X Bus Mode

When L2 Cache is installed on the MPC8266ADS-PCI, the MPC8266 may no longer operate in single MPC8266 mode since the address must be seen as is by the cache. That requires the use of the external address multiplexers for the SDRAM. In this mode, SDRAM performance is decreased due to added wait-state, caused by the delay associated with the external multiplexers, on the 1’st access in a page,.

NOTE

In this mode, only devices which are 60x compatible (or devices which have 64 bit data bus and are buffered from the 60x bus) can operate on the 60x bus. This due to the 60x bus address tenure feature. This means that when the L2 Cache is used, the Flash, EEPROM, BCSR and PCI Interrupt Controller are not accesible. For further details, consult the MPC8266 User Manual.

4.5 Buffering

In order to achieve best performance, it is necessary to reduce the capacitive load over the 60X bus as much as possible. Therefore, the slower devices on the bus, i.e., the Flash SIMM, E^2PROM, ATM UNI M/P interface, PCI Interrupt Controller and the BCSR are buffered, while the SDRAM DIMM and the cache are not buffered from the 60X bus.

Latches are provided over address and strobe (when necessary) lines while transceivers are provided for data. Use is done with 74ALVT buffers (by Philips) which are 3.3V operated and 5V tolerant[1] and provide bus hold to reduce pull-up/pull-down resistors count (as required by the MPC8266). This type of buffers reduces noise on board due to reduced transitions' amplitude.

To further reduce noise and reflections, serial damping resistors are placed over SDRAM DIMM's address and all MPC8266 strobe lines.

The data transceivers are open only if there is an access to a valid[2] buffered board address or during Hard - Reset configuration[3]. That way data conflicts are avoided in case an unbuffered memory read or off-board memory is read - provided that it is not mapped to an address valid on board. It is the users' responsibility to avoid such errors.

The PCI bus is not buffered at all because the PCI Standard is very strict and defines exactly the electrical characteristics of the bus which is buffer free.

4.6 Chip - Select Generator

The memory controller of the MPC8266 is used as a chip-select generator to access on-board (and off-board) memories, saving boards' area, reducing cost, power consumption and increasing flexibility. To enhance off-board application development, memory modules (including the BCSRx) may be disabled via BCSR[4] in favor of an external memory connected via the expansion connectors. That way, a CS line may be used off-board via the expansion connectors, while its

1. Required for Flash, E^2PROM, Interrupt Controller and BCSR
2. An address which is covered in a Chip-Select region, that controls a buffered device.
3. To allow a configuration word stored in the Flash/E^2PROM memory to become active.

associated local memory is disabled.

When a CS region, assigned to a buffered[1] memory, is disabled via BCSR, the local data transceivers are disabled during access to that region, avoiding possible[2] contention over data lines.

The MPC8266 chip-select assignments to the various memories / registers on the MPC8266ADS-PCI are shown in Table 4-5.

Table 4-5. MPC8266-MB Chip Select Assignments

Chip Select:	Assignment	Bus	Timing Machine
$\overline{CS0}$	Flash SIMM / E^2PROM[a]	60X (Buffered)	GPCM
$\overline{CS1}$	BCSR	60X (Buffered)	GPCM
$\overline{CS2}$	SDRAM DIMM (single bank only)	60X (Main)	SDRAM Machine 1
$\overline{CS3}$	SDRAM DIMM[b]	60X (Main)	SDRAM Machine 1
$\overline{CS4}$	E^2PROM / Flash SIMM[a]	60X (Buffered)	GPCM
$\overline{CS5}$	ATM UNI Microprocessor I/F	60X (Main)	GPCM
$\overline{CS6}$	Communication Tool M/P Interface $\overline{CS1}$.	60X (Buffered)	GPCM/UPMx
$\overline{CS7}$	Communication Tool M/P Interface $\overline{CS2}$.	60X (Buffered)	GPCM/UPMx
$\overline{CS8}$	PCI Interrupt Controller	60X (Buffered)	GPCM
$\overline{CS(9\text{-}11)}$	Unused, user available	-	-

a. Selection is done by a dip-switch.
b. For two banks SDRAM DIMMs.

4.7 Synchronous Dram DIMM (60X Bus)

To enhance performance, especially in higher operation frequencies - 16 MBytes of SDRAM DIMM is provided on board. The SDRAM DIMM is unbuffered from the MPC8266 60X bus and is configured as 2 X 1M X 64. Use is done with SDC2UV6482C-84T-S, unbuffered 168 pin DIMM by Fujitsu or compatibles, which is composed of eight 2 X 1M X 8 sdram chips (MB81117822A- 84). The DIMM's data sheet may be obtained on the Internet at URL: ***http://***

4. After the BCSR is removed from the local memory map, there is no way to access it but to re-apply power to the MPC8266ADS-PCI.
1. When an unbuffered CS region is being accessed, buffers do not open anyway.
2. During read cycles.

www.fujitsumicro.com/products/memory/sdram_mod.html, while the sdram chips' data sheet may be obtained at URL: ***http://www.fujitsumicro.com/products/memory/sdrams.html***.

Unlike memory SIMMs which have few presence detect lines for configuration report, the DIMM's configuration information is stored in a 256 Byte Serial E^2PROM residing on the DIMM, compatible with I^2C protocol. In fact, all necessary information is in the 1st half of the E^2PROM, while the 2nd half is system available. On the MPC8266ADS-PCI, the DIMM's configuration E^2PROM is connected to the MPC8266 I^2C controller to inquire for SDRAM DIMM's configuration after hard-reset sequence.

The SDRAM's timing is controlled by SDRAM Machine #1 associated with 60X bus, via its assigned Chip Select lines (See Table 4-5.). The SDRAM Machine supports PBI (Page Bank Interleave) which increases the SDRAM throughput. The SDRAM connection scheme when no L2 cache is used is shown in Figure 4-5.

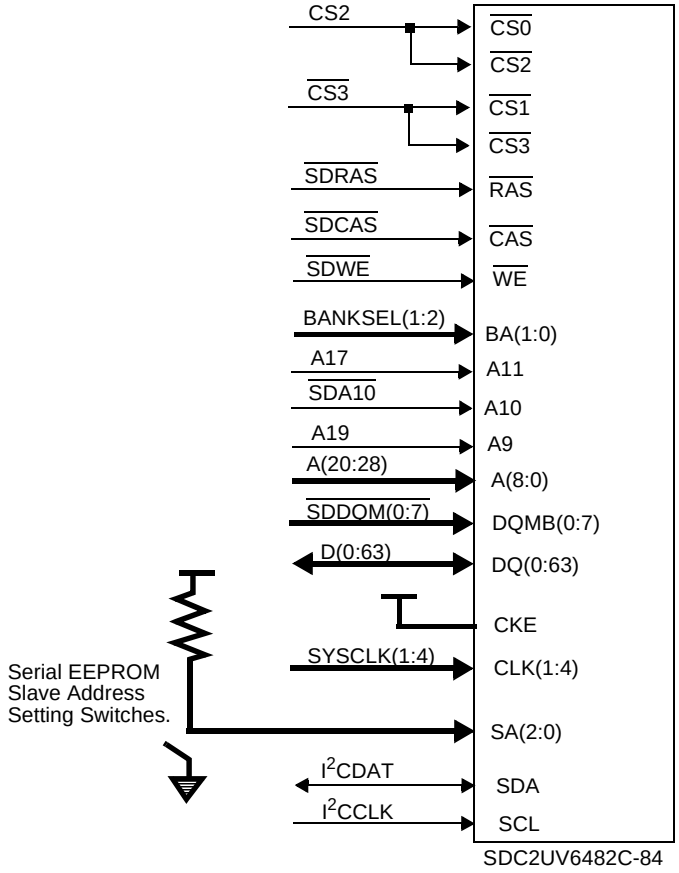


Figure 4-5. SDRAM DIMM Connection Scheme - No L2 Cache

The SDRAM connection scheme when L2 cache is installed is shown in Figure 4-6.

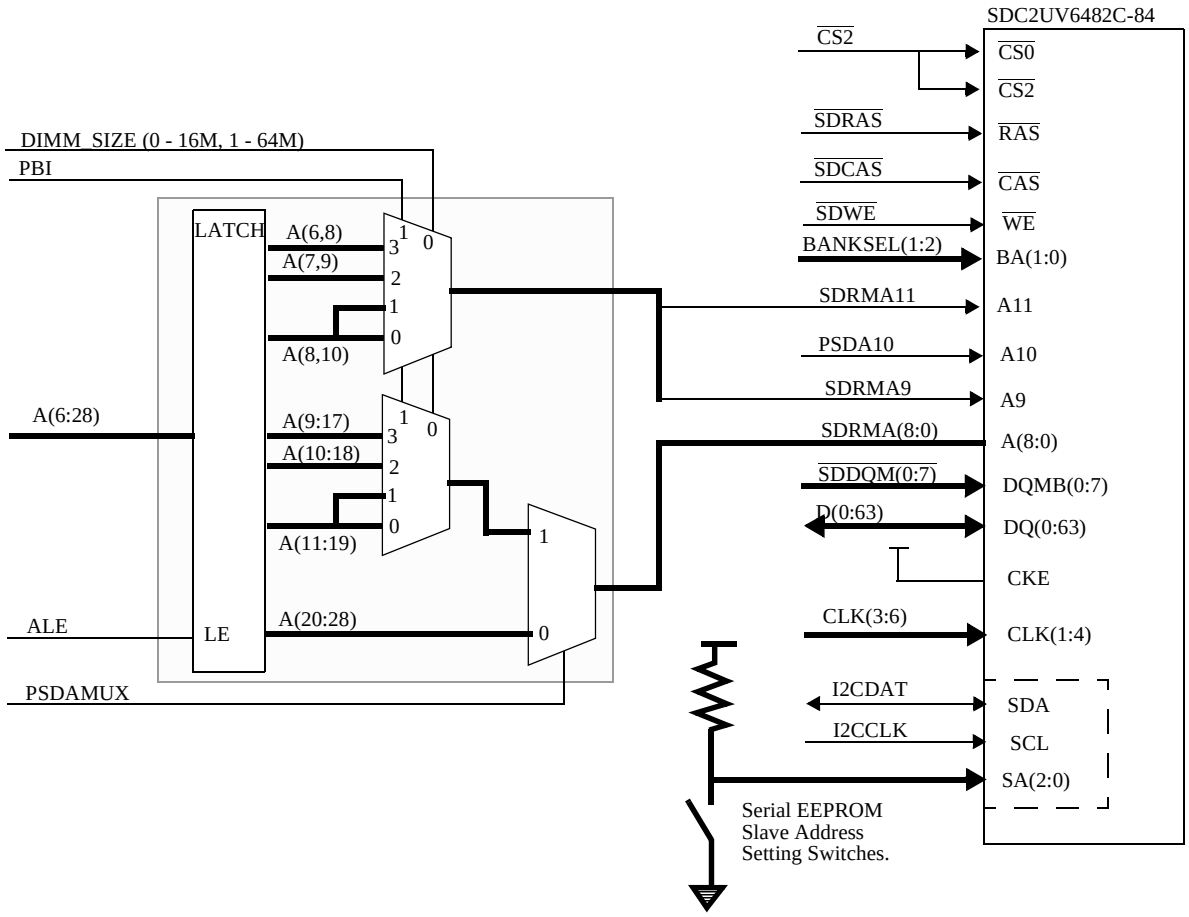


Figure 4-6. SDRAM DIMM - 60x Bus Connection Scheme with L2 Cache

The SDRAM performance, is shown in Table 4-6.:

Table 4-6. SDRAM DIMM (84MHz) Performance Figures

Cycle Type	System Clock Cycles @ 66MHz Bus Clock Freq.	System Clock Cycles @ 66MHz Bus Clock Frequency With L2CACHE
Burst Read - Page Miss	6[a],1,1,1	7[a],1,1,1
Burst Read - Page Hit	4[a],1,1,1	5[a],1,1,1
Burst Write - Page Miss	4[a],1,1,1	5[a],1,1,1
Burst Write - Page Hit	2[a],1,1,1	4[a],1,1,1
Refresh	8[b]	8[b]

a. From TS Asserted. First access may be longer due to internal pipeline delay.
b. Not including arbitration overhead.

4.7.1 SDRAM Programming

After power-up, the SDRAM needs to be initialized by means of programming to establish its mode of operation. The SDRAM is programmed according to the following procedure:

1. Issue Precharge-All command
2. Issue 8 CBR refresh commands
3. Issue MODE-SET command.

An SDRAM is programmed by issuing a Mode Register Set command. During that command, data is passed to the Mode Register through the SDRAMs' address lines. This command is fully supported by the SDARM machine of the MPC8266. Before that can take place, the SDRAM machine of the MPC8266 has to be initialized.

Mode Register programming values are shown in Table 4-7.:

Table 4-7. 66 MHz SDRAM DIMM Mode Register Programming

SDRAM Address Line[a]	SDRAM Mode Reg Field	Value	Meaning:
A11 (MSB)	Reserved	'0'	
A10	Reserved	'0'	
A9	Opcode	'0' / '1'	0 - Burst Read & Burst Write (Copy-Back data cache) 1 - Burst Read & Single Write (Write-Through Data cache)
A8	Reserved	'0'	
A7	Reserved	'0'	
A6 - A4	CAS Latency	'010'	Data Valid 2 Clocks cycles after CAS Asserted
A3	Burst Type	'0'	Sequential Burst
A2 - A0	Burst Length	'011'	8 Operand Burst Length

a. Actually SDRAMs' A0 is connected to MPC8266s' A28 and so on...

4.7.2 SDRAM Refresh

The SDRAM is refreshed using its auto-refresh mode. I.e., using SDRAM machine one's periodic timer, an auto-refresh command is issued to the SDRAM every 13.4 μsec, so that all 2048[1] SDRAM DIMM rows are refreshed within specified 32.8 msec, while leaving an interval of ~5.4 msec of refresh redundancy within that window, as a safety measure, to cover for possible delays

1. In fact each SDRAM component is composed of 2 internal banks each having 2048 rows, but they are refreshed in parallel.

in bus availability for the refresh controller.

4.7.3 L2-Cache Support Influence On SDRAM Design

To support an optional L2-Cache on the MPC8266-MB, the following measures need to be taken:

1. Optional Latches - Multiplexers are added over selected address lines. See Figure 4-6. These Latches - Multiplexers are normally by-passed by 0 Ω resistors that are not assembled in L2cache boards.
2. The MPC8266 supports additional wait-state on SDMUX line, so that the row-address may be allowed to propagate via the Latch - Multiplexers in time for the Activate command.
3. To support SDRAM PBI (Page Based Interleaving), the relative location of the Row-Address field, is shifted up the address lines, depended on the number of internal banks within a SDRAM DIMM. This since the Bank Select line(s) are inserted between the Column (LSB) and Row (MSB) address lines. As can be seen from Figure 4-6., PBI and DIMM_SIZE signals, driven via BCSR0, select the correct address line group for a specific DIMM size, with PBI set.

The performance of the SDRAM is decreased by the addition of the external multiplexers of the SDRAMs' address lines. The effects may be seen in Table 4-6.

4.7.4 SDRAM DIMM Configuration Information

Unlike memory SIMMs which have few presence detect lines for configuration report, the DIMM's configuration information is stored in a 256 Byte Serial EEPROM residing on the DIMM, compatible with I^2C protocol. In fact, all necessary information is in the first half of the EEPROM, while the second half is system available. On the board, the DIMM configuration EEPROM is connected to the MPC8266 I^2C controller, to inquire for SDRAM DIMM's configuration, after hard-reset sequence.

4.8 Flash Memory SIMM

The MPC8266-MB is provided with 8Mbyte of 95 nsec flash memory SIMM, the SM73228XG1JHBGO by Smart Modular Technology which is composed of four LH28F016SCT-L95 chips by Sharp, arranged as 2M X 32 in a single bank. Support is given also to 16MBytes and 32 MBytes simms. The Flash SIMM resides on an 80 pin SIMM socket and is buffered from the 60X bus to reduce capacitive load over it.

To minimize use of MPC8266s' chip-select lines, only one chip-select line ($\overline{CS0}$ or $\overline{CS4}$ if the E^2PROM is using $\overline{CS0}$) is used to select the Flash as a whole, while distributing chip-select lines among the module's internal banks is done by on-board programmable logic, according to the Presence-Detect lines of the Flash SIMM inserted to the MPC8266-MB.

The access time of the Flash memory provided with the MPC8266-MB is 95 nsec, however, devices with different delay are supported as well. By reading the delay section of the Flash SIMM Presence-Detect lines (see Table 4-13.), the debugger can establish (via register OR0 in

case $\overline{CS0}$ is used or OR4 if $\overline{CS4}$ is used) the correct number of wait-states needed to access the Flash SIMM (considering 66MHz system clock frequency).

The control over the Flash is done with the GPCM and a dedicated $\overline{CS0}$ (or $\overline{CS4}$) region which controls the whole bank. During hard - reset initialization[1], the debugger or any application S/W for that matter, reads the Flash Presence-Detect lines via BCSR and determines how to program registers BR0 & OR0 (or BR4 & OR4), within which the size and the delay of the region are determined.The performance of the flash memory is shown in Table 4-8.:

Table 4-8. Flash Memory Projected Performance Figures

	Number of System Clock Cycles @ 66 MHz Bus Clock Freq.
Cycle Type \ Flash Delay [nsec]	**95**
Read Access	8[a]
Write[b] Access	9[a]

a. From $\overline{TS}$ asserted. However, due to internal activity, these figures may be larger.
b. The figures in the table refer to the actual write access. The write operation continues internally and the device has to be polled for completion.

The Flash module may be disabled / enabled at any time by writing ’1’ /’0’ respectively to the

1. i.e., initialization that follow the hard reset sequence at system boot.

$\overline{\text{FlashEn}}$ bit in BCSR1. The Flash connection scheme is shown in Figure 4-7..

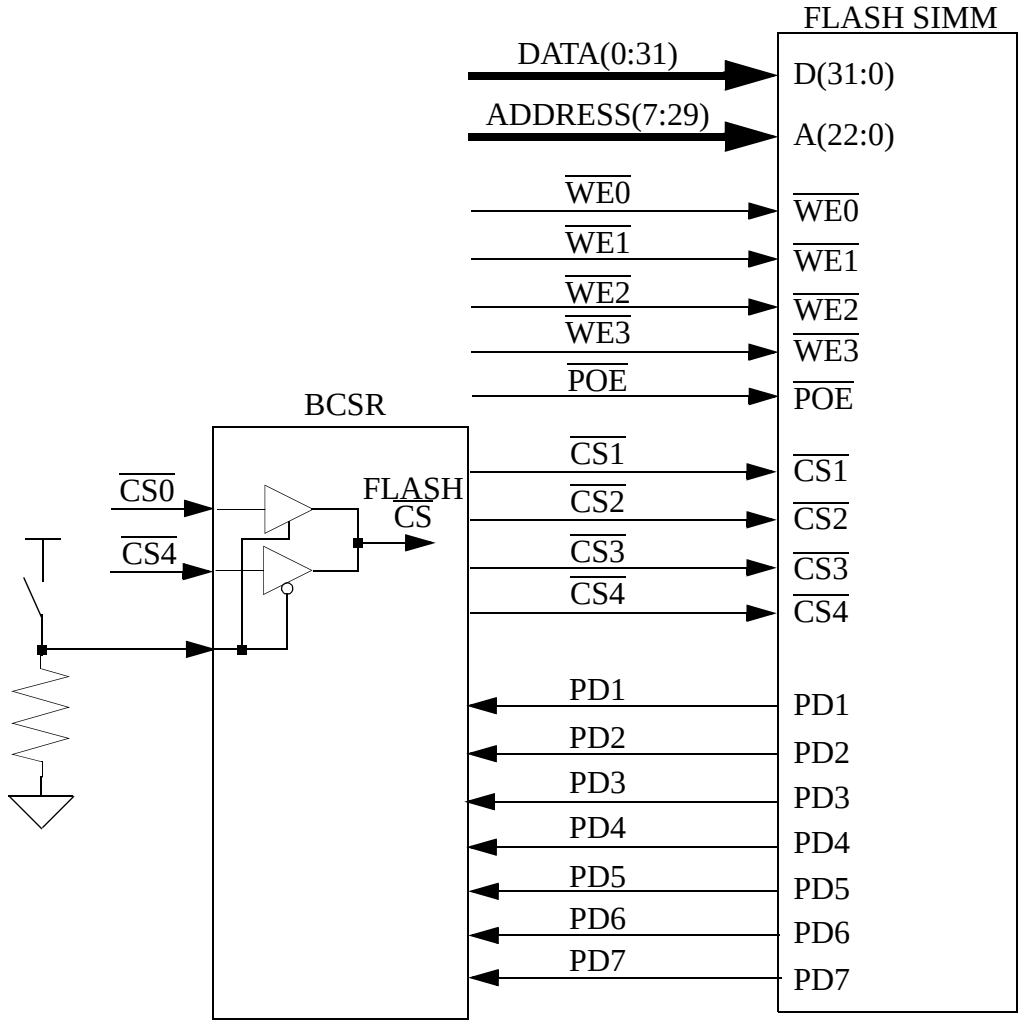


Figure 4-7. FLASH SIMM Connection Scheme

As can be seen in Figure 4-7., the FLASH $\overline{\text{CS}}$ is distributed to four $\overline{\text{CS}}$ signals. The distribution depends on the size of the FLASH module installed - it is read by the BCSR using the PD(1-7) pins.

The Hard-Reset configuration word stored in the FLASH differs from the one stored in the E^2PROM in the BPS field which is the Boot Port Size - the E^2PROM is 8 bits while the FLASH is 32 bits.

4.8.1 Flash Programming Voltage

Support is given to 5V as well as 12V programmable modules. The selection between VPP's voltage levels is done via a dedicated jumper. To avoid inadvertent programming or erasure of the Flash it is recommended to leave the jumper open so that no VPP is applied to the Flash SIMM.

4.8.2 Flash and L2Cache

If the L2 cache is installed, the MPC8266 needs to be programmed to 60x bus mode. This requires the latches for the buffered address bus to the Flash (As well as all other slow static devices) to be

enabled. The 3 lowest order address lines for the Flash, are provided by the BADDR(27-29) lines of the MPC8266. However, BADRR29 function of the MPC8266 is multiplexed with $\overline{CI}$ (Cache Inhibit) function over the same pin. Therefore, prior to enabling the L2Cache, any code residing in the Flash, should be moved into the PowerPC bus SDRAM[1], prior to changing BADDR29 function to $\overline{CI}$ via SIUMCR.

4.9 E^{2}PROM Memory

The MPC8266-MB is provided with 8 KBytes of E^{2}PROM memory in a PLCC package. The E^{2}PROM resides on a socket in case it is desired to replace or re-program a different configuration for the board. The E^{2}PROM is used only for the purpose of supplying the Reset Configuration Word during power-on reset and for storing the PCI configuration data. It is used as a back-up for the Flash memory in case the Flash is not installed or the data it holds is incorrect. As a back-up, it holds the default Hard-Reset configuration word and the default PCI configuration. The Hard-Reset configuration word stored in the E^{2}PROM differs from the one stored in the FLASH in the BPS field which is the Boot Port Size - the E^{2}PROM is 8 bits while the FLASH is 32 bits. It uses a single chip-select, $\overline{CS0}$ or $\overline{CS4}$, which depends on the chip-select used by the Flash. The selection of the chip-select is done by a dip-switch. The E^{2}PROM connection scheme is shown in Figure 4-8.

The device used is ATMEL AT28HC64B, a 5V Byte alterable E^{2}PROM, 150ns access time with byte-wide JEDEC pinout. Although the device is placed in a socket, it can be programmed on-board. In order to program the device on-board, it has to be unlocked - it can be locked to prevent unauthorized alterations of its contents. The lock can be done by hardware or software. The hardware lock is done by write inhibit - the MPC8266 does not assert $\overline{WE}$ during write cycles (set in the BRx register). The software lock is achieved by writing a unique sequence to the device. To

1. It is required to do so anyway, since the L2Cache must operate within a full 64-bit data bus environment.

unlock, a different unique sequence has to be written.

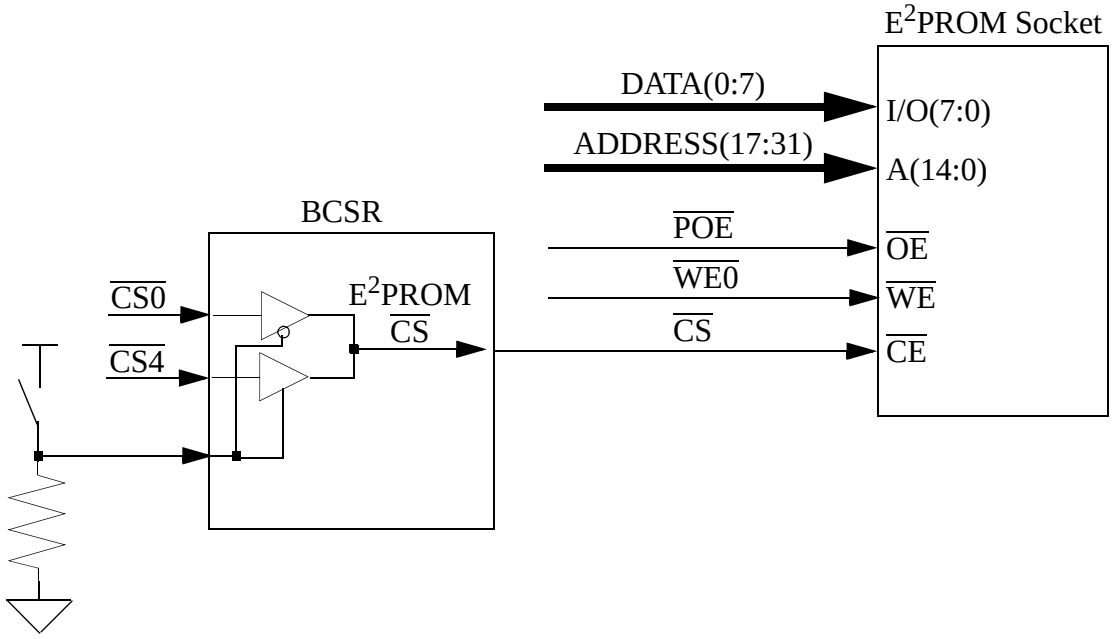


Figure 4-8. E^2PROM Connection scheme

Additional address lines are connected to the socket according to the JEDEC format as an option to use E^2PROM up to 32 KByte. To allow proper operation with the L2 Cache, the MPC8266 needs to be set to 60X bus mode in which the address bus for the E^2PROM[1] is latched.

4.10 PCI Bus

The MPC8266 has a PCI module which enables it to act as an Host (Master) or a Target. On this board, the MPC8266 serves only as a PCI host - a bridge between the PCI Bus and the PowerPC core.

The PQ2 PCI Bridge is designed to connect the PowerPC processor and memory system to the PCI system bus, to which I/O components are connected. The PCI Bridge enables the MPC8266 to gluelessly bridge PCI masters and agents to a PowerPC system host. It uses a 32-bit multiplexed, address/data bus that can run from 25MHz up to 66MHz. The interface provides address and data parity with error checking and reporting. It also provides three physical address spaces: 32-bit address memory; 32-bit address I/O; and the PCI configuration space.

The MPC8266 also includes an on-chip Arbiter which enables arbitration of up to three PCI masters. Only three PCI slots are supported on the MPC8266-MB because of the Arbiter capacity. Each slot can host either a PCI master or PCI target. The MPC8266 as a Bridge can support more PCI devices but that will require extra slots that can host PCI targets only. Therefore, to avoid

1. As well as all other slow static devices.

dedicated slots for PCI targets, only three slots are implemented.

The PCI Bridge is implemented on the PQ2 Local Bus. Due to PCI Standard restrictions, no other application can reside on the local bus. The PCI bus can operate at frequencies of 25MHz up to 66MHz @ 3.3V only. The 3.3V restriction is due to the MPC8266 which is not 5V compliant. The PCI bus layout is shown in Figure 4-9. Special care was taken when the layout of the MPC8266ADS-PCI was done so that the PCI standard recommendations are followed strictly.

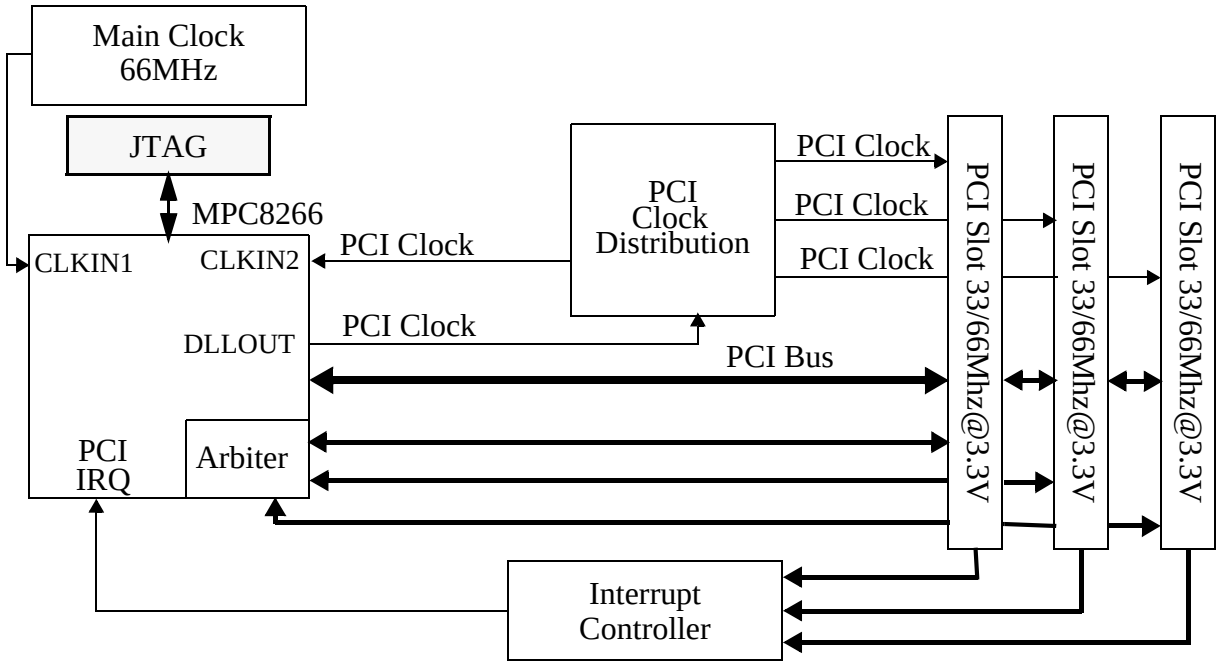


Figure 4-9. PCI Bus Scheme

The clock source for the MPC8266 is Main Clock 66MHz clock oscillator. The PCI Clock is derived internally from the Main Clock and output at DLLOUT. That clock is then distributed to each PCI device on the bus in a way that they are all synchronized (by keeping all clock traces the same length). The PCI Clock is also fed back to the MPC8266 for synchronization and skew elimination purposes.

An interrupt from any PCI slot is handled by a simple generic Interrupt Controller. Each slot can generate up to four interrupts for a total of twelve interrupts that the controller will support. It will be made of two register mapped in a dedicated $\overline{CS}$ region. One is an Interrupt Register (see Table 4-3.) and the second is Interrupt Mask Register (see Table 4-4.). A simple priority scheme will be devised (TBD) to allow the controller to support more than one interrupt concurrently.

4.11 L2-CACHE Support

To enhance benchmarking, optional support is provided for L2-Cache. Use is done with two

MPC2605 devices, each containing 256KBytes of look-aside[1] cache along with its control, providing a total of 512KByte L2-cache.

The cache is connected directly over the 60X bus and is supported gluelessly by the MPC8266. The cache data sheet may be obtained via the internet at URL: ***http://mot-sps.com/books/dl156/pdf/mpc2605rev5.pdf***.

The presence of the L2-Cache, calls for the introduction of latch - multiplexers over SDRAMs' address lines because the MPC2605 snooping logic needs to monitor the address as is (linear rather than multiplexed) and the bus works by the 60X bus protocol, allowing address pipelining[2]. These latch - multiplexers are soldered in place only in case a cache is installed on-board. Otherwise they are omitted and bypassed by 0 Ω resistors. See also **Section 4.7.3 L2-Cache Support Influence On SDRAM Design.**

4.11.1 L2 Cache Configuration & Control

The cache is configured via 5 configuration lines, CFG(0:4), for the following functions:

1. Cache size is set by CFG(0:2). The various settings of these lines per each cache module are encoded in Table 4-9.

Table 4-9. L2 Cache CFG(0:2) Settings

L2 Cache Size [Byte]	CFG(0:2)
256K	'000' (Reserved)
512K	'010' -1'st Module (A26 == 0)
	'011' - 2'nd Module (A26 == 1)

2. Snoop is Enabled - CFG3 driven low for both modules.
3. $\overline{\text{AACK}}$ assertion enabled - CFG4 driven high for both modules.

The caches' $\overline{\text{HRESET}}$ lines are connected directly to the $\overline{\text{SRESET}}$ line of the MPC8266 so that whenever Soft-reset is asserted to or by the MPC8266, the cache is reset along with it, loosing all data previously stored in it. The cache has 5 control lines that control its operation and state:

- $\overline{\text{PWRDWN}}$ - constantly set to high (no power down support on the MPC8266-MB)
- $\overline{\text{L2FLUSH}}$ - assertion of which[3] flushes out the cache array. This signal is controlled by BCSR0.
- $\overline{\text{L2MISS\_INH}}$ - in fact Cache-Lock. When Asserted the cache does not change its contents. Controlled by BCSR0.
- $\overline{\text{L2TAG\_CLR}}$ - Clears all tag memory. Controlled by BCSR0.
- $\overline{\text{L2UPDATE\_INH}}$ - In fact cache freeze (without information loss). Controlled by BCSR0.

All the above signals are connected directly to both cache modules.

1. i.e., residing on the same bus as the processor.
2. Only single level is allowed with the MPC8266.
3. For minimum 8 Bus clock cycles.

4.12 Communication Ports

The MPC8266-MB has several communication ports, to allow convenient evaluation of the CPM features. Obviously, it is not possible to provide all types of communication interfaces supported by the CPM, but it is made convenient to connect any communication interface devices to the MPC8266 via the CPM Expansion connectors, residing on the edge of the mother-board.

The communication ports' interfaces provided on the MPC8266-MB are listed below:

1. 155 Mbps ATM UNI on FCC1 with Optical interface, using the UTOPIA interface.
2. 100/10-Base-T Port on FCC2 with T.P. interface, MII controlled.
3. Dual RS232 ports residing on SMC1 & SMC2.

4.12.1 ATM Port

To support the MPC8266s' ATM controller, a 155.52Mbps User Network Interface (UNI) is provided on board, connected to FCC1 of the MPC8266 via UTOPIA I/F. Use is done with PM5350 S/UNI-155-ULTRA by PMC-SIERA. Although these transceivers are capable of supporting 51.84Mbps rate, support is given to 155.52Mbps only.

The control over the transceiver is done using the microprocessor interface of the transceiver, controlled by the MPC8266 memory controllers' GPCM. Since the UNI is 5V powered and the MPC8266 is 3.3V powered (5V intolerant), the UNI is buffered (LCX buffers) from the MPC8266 on both the receive part of UTOPIA interface and the microprocessor control ports.

The ATM transceiver may be enabled / disabled at any time by writing '0' /'1' respectively to the $\overline{\text{ATMEN}}$ bit in BCSRx. When $\overline{\text{ATMEN}}$ is negated, ('1') the microprocessor control port is also detached from the MPC8266 and its associated FCC may be used off-board via the expansion connectors.

The ATM transceiver reset input is driven by $\overline{\text{HRESET}}$ signal of the MPC8266, so that the UNI is reset whenever a hard-reset sequence occurs. The UNI may also be reset by either asserting ATM_RST bit in BCSR1 (see Table 4-11.) or by asserting ('1') the RESET bit in the Master Reset and Identify / Load Meters register via the UNI microprocessor interface.

The UNI transmit and receive clocks are fed with a 19.44 MHz +/- 20 ppm, clock generator, 5 V powered, while the receive and transmit fifos' clocks of the UTOPIA interface are provided by the MPC8266. The MPC8266 can provide the same clock for both UTOPIA transmit and receive or separate clocks for each, hard-configured[1].

The ATM SAR is connected to the physical medium by an optical interface. Use is done with HP's HFBR 5205 optical interface, which operates at 1300 nm with upto 2 Km transmission range.

4.12.2 100/10 Base - T Port

A fast Ethernet port with T.P. (100-Base-TX) I/F is provided on the MPC8266-MB. This port also supports 10 Mbps ethernet (10-Base-T) via the same transceiver - the LXT970 by Level One.

1. Using resistors.

The LXT970 is connected to FCC2 of the MPC8266 via MII interface, which is used for both - devices' control and data path. The initial configuration of the LXT970 is done be setting the desired values at 8 configuration signals: FDE, CFG(0:1) and MF(0:4). The MF(0:4) pins however, are controlled by 4 - voltage levels, this to allow each pin to configure two functions. On the MPC8266-MB these pins are driven by factory set 0Ω resistors, connected to a voltage divider, allowing future option change during production.

The LXT970 reset input is driven by $\overline{\text{HRESET}}$ signal of the MPC8266, resetting the transceiver whenever hard-reset sequence is taken. The LXT970 may also be reset by either asserting the FETH_RST bit in BCSR1 (see Table 4-11.) or by asserting bit 0.15 (MSB of LXT970 control register) via MII I/F.

To allow external use of FCC2, its pins appear at the CPM expansion connectors and the ethernet transceiver may be Disabled / Enabled at any time via the MIIs' MDIO port.

The LXT970 is able to interrupt the MPC8266 via $\overline{\text{IRQ7}}$ line. This line is shared also with the CPM expansion connectors. Therefore, any tool that is connected to $\overline{\text{IRQ7}}$ or $\overline{\text{IRQ6}}$ for that matter, should drive these lines with an Open Drain buffer. Both $\overline{\text{IRQ6}}$ and $\overline{\text{IRQ7}}$ are pulled-up on the MPC8266-MB.

4.12.2.1 LXT970 Control

The LXT970 is controlled via the MII management[1] port which is a 2 wire interface: a clock (MDC) and a bidirectional data line (MDIO). This is in fact a bus, i.e., up to 32 devices may reside over it, while the protocol defines a 5-bit slave address field, which is compared against the slave address set to each device by hardware during device reset, according to the levels on MF(4:0) pins. On the board, the slave address is hard-set to b00000. The MPC8266 interfaces this port using two PI/O pins: PC9 for MDIO and PC10 for MDC. There is no special support within the MPC8266 for the MDIO port and the protocol is implemented in S/W.

The MDIO port may interrupt a host in 2 ways: (a[2]) driving low the MDIO line during IDLE time or (b) using a dedicated interrupt line FDS/$\overline{\text{MDINT}}$ which may also serve as Full-Duplex indication. This line is connected to the MPC8266's DP7/CSE1/$\overline{\text{IRQ7}}$ line, appearing also at the CPM expansion connectors. After the LX970 is reset, the FDS/$\overline{\text{MDINT}}$ pin, wakes-up as FDS rather than $\overline{\text{MDINT}}$ and therefore, **MUST be initially programmed to MDINT function, by setting 17.1 bit**, otherwise, $\overline{\text{IRQ7}}$ may be constantly driven low, possibly generating interrupts to the MPC8266, if not masked properly.

Since $\overline{\text{IRQ7}}$ may also be driven by any tool, connected to the expansion connectors, it should be driven with an Open Drain buffer. $\overline{\text{IRQ7}}$ is pulled-up on the board.

4.12.3 RS232 Ports

To assist user's applications and to provide convenient communication channels with both a terminal and a host computer, two identical RS232 ports are provided on the MPC8266-MB, connected to SCC1 and SCC2 ports of the MPC8266. Use is done with MC145583 transceiver which generates RS232 levels internally using a single 3.3V supply and has a standby mode. When the $\overline{\text{RS232EN1}}$ or $\overline{\text{RS232EN2}}$ bits in BCSR1 are asserted (low), the corresponding

1. Also known as MII MDIO port.
2. Not supported on the board.

transceiver is enabled. When negated, the corresponding transceiver is in standby mode, within which the receiver outputs are tri-stated, enabling the use of the corresponding ports' pins off-board via the expansion connectors.

Nine pins, female D-Type stacked connector is used, configured to be directly (via a flat cable) connected to a standard IBM-PC like RS232 connector.

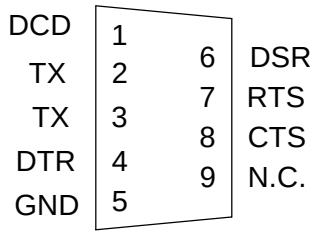


Figure 4-10. RS232 Serial Ports Connector

4.12.3.1 RS-232 Ports' Signal Description

In the list below, the directions 'I','O', and 'I/O' are relative to the MPC8266-MB board. (i.e.'I' means input to the MPC8266-MB)

- CD (O) - Data Carrier Detect. This line is always asserted by the MPC8266-MB.
- TX (O) - Transmit Data.
- RX (I) - Receive Data.
- DTR (I) - Data Terminal Ready. This signal is used by the software on the MPC8266-MB to detect if a terminal is connected to the board.
- DSR (O) - Data Set Ready. This line is always asserted by the MPC8266-MB.
- RTS (I) - Request To Send. This line is not connected in the MPC8266-MB.
- CTS (O) - Clear To Send. This line is always asserted by the MPC8266-MB.

4.13 Board Control & Status Register - BCSR

Most of the hardware options on the MPC8266-MB are controlled or monitored by the BCSR, which is a 32 bit wide read / write register file. The BCSR is accessed via the MPC8266s' memory controller (see Table 4-5.) and in fact includes 8 registers: BCSR0 to BCSR7. Since the minimum block size for a CS region is 32KBytes and only A(27:29) lines are decoded by the BCSR for register selection, BCSR0 - BCSR7 are duplicated inside that region.

The following functions are controlled / monitored by the BCSR:

1. PBI (60x Bus mode only)
2. L2 Cache Inhibit
3. L2 Cache Flush
4. L2 Cache Lock
5. L2 Cache tag Clear.

6. ATM Port Control which includes:
   - Transceiver Enable / Disable
   - Transceiver Reset.
7. Fast Ethernet Port Control which includes:
   - Transceiver Initial Enable
   - Transceiver Reset
8. RS232 port 1 Enable / Disable.
9. RS232 port 2 Enable / Disable.
10. Flash Size / Delay Identification.
11. $\overline{CS0}$ assignment after hard-Reset to FLASH SIMM / E^2PROM.
12. External (off-board) tools Support which include:
    - Tool Identification
    - Tool Revision
    - Tool Status Information
13. S/W Option Identification.
14. Board revision code.
15. Fast download via JTAG (optional).
16. Power-on Reset via JTAG (optional).
17. PCI cards Present Detect and card type.

Since part of the MPC8266-MBs' modules are controlled by the BCSR and since they may be disabled in favor of external hardware, the enable signals for these modules are presented at the CPM expansion connectors, so that off- board hardware may be mutually exclusive enabled with on-board modules.

4.13.1 BCSR0 - Board Control - Status Register 0

The BCSR0 is a control register on the MPC8266-MB. It is accessed at **offset 0** from BCSR base address. It may be read or written at any time[1]. BCSR0 gets its defaults upon Power-On reset. BCSR0 fields are described in Table 4-10..

Table 4-10. BCSR0 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0	PBI	***Page Base Interleaving***. In 60X mode (i.e., with L2-Cache), this bit should reflect (**system programmer responsibility**) the state of PBI bit in PSDMR. When active (**High**) it changes the address Muxing scheme for the SDRAM, so that to match a scheme where Bank Select signals are connected below lower row address lines. When Inactive, the addressing scheme is such that Bank Select lines are taken from the higher order address lines (above row address). This signal operates in conjunction to DIMM_SIZE signal below. **In Single MPC8266 Mode (i.e., without L2-Cache), this bit has no effect**.	0	R/W

1. Provided that BCSR is not disabled.

Table 4-10. BCSR0 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
1	DIMM_SIZE	***Sdram DIMM Size***. In 60X mode (i.e., with L2-Cache), this bit, in conjunction with PBI above, controls the address muxing scheme for the Sdram DIMM. When **Low**, the addressing scheme matches a **16 MByte** DIMM while when ***High***, the addressing scheme matches a ***64 MBytes*** DIMM. **In Single MPC8266 Mode (i.e., without L2-Cache), this bit has no effect**.	0	R/W
2	L2C_INH	***L2 Cache Inhibit***. When this bit is active (**low**), the L2 cache is inhibited and unable to respond to cacheable cycles. However, bus activity is still monitored by the cache so that it may respond immediately after this signal is negated. This signal is connected to the MPC2605's $\overline{\text{L2 UPDATE INH}}$. This signal has no function in a MPC8266-MB that does not have an L2 Cache installed.	0	R,W
3	L2C_FLUSH	***L2 Cache Flush***. When this bit is active (**low**) for min. 8 bus cycles, the MPC2605 initiates a process within which, valid lines are marked invalid, while dirty lines are written back to memory and marked invalid. This signal is connected to the $\overline{\text{L2 FLUSH}}$ signal of the MPC2605. This signal has no function in a MPC8266-MB that does not have an L2 Cache installed.	1	R,W
4	L2C_LOCK	***L2 Cache Lock***. When this bit is active (**low**), the MPC2605 will stop entering new data into the cache, while yet maintaining existing data and responding to cacheable cycles. This signal has no function in a MPC8266-MB that does not have an L2 Cache installed.	1	R,W
5	L2C_CLEAR	***L2 Cache Clear***. When this bit is active (**Low**) for min. 8 bus clock cycles, the L2 cache invalidates all its entries, without flushing, the same process as with $\overline{\text{HRESET}}$ asserted. However, it still monitors the bus, so it can immediately respond when this process ends. This signal is connected to the $\overline{\text{L2 TAG CLR}}$ of the MPC2605, but has no function when a cache is not installed on the MPC8266-MB.	1	R,W
6 - 31	Reserved	Un-implemented	0	R

4.13.2 BCSR1 - Board Control - Status Register 1

The BCSR1 is a control register on the MPC8266-MB. It is accessed at **offset 4** from BCSR base address. It may be read or written at any time[1]. BCSR1 gets its defaults upon Power-On reset. The fields are described in Table 4-11.

Table 4-11. BCSR1 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0	Reserved	Un-implemented.	0	R

1. Provided that BCSR is not disabled.

Table 4-11. BCSR1 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
1	FLASH_CS0	***FLASH $\overline{CS0}$***. When asserted (**low**) $\overline{CS0}$ is assigned to the FLASH SIMM and $\overline{CS4}$ is assigned to E^2PROM. When negated, $\overline{CS0}$ is assigned to the E^2PROM and $\overline{CS4}$ is assigned to the FLASH SIMM. The assignments selection is done via a dedicated jumper.	0	R
2	ATM_EN	***ATM Port Enable***. When asserted (**low**) the ATM UNI chip (PM5350) connected to FCC1 is enabled for transmission and reception. When negated, the ATM transceiver is in standby mode and its associated buffers[a] are in tri-state mode, freeing all its i/f signals for off-board use via the expansion connectors.	1	R,W
3	ATM_RST	***ATM Port Reset***. When asserted (**low**), the ATM port transceiver is in reset state. This line is driven also by $\overline{HRESET}$ signal of the MPC8266.	1	R,W
4	FETHIEN	***Fast Ethernet Port Initial Enable***. When asserted (**low**) the LXT970's MII port, residing on FCC2, is enabled after Power-Up or after FETH_RST is negated. When negated (**high**), the LXT970's MII port is isolated after Power-Up or after FETH_RST is negated and all i/f signals are tri-stated. After initial value has been set, this signal has no influence over the LXT970 and MII isolation may be controlled via MDIO 0.10 bit.	1	R,W
5	FETH_RST	***Fast Ethernet port Reset***. When active (**low**) the LXT970 is reset. This line is also driven by $\overline{HRESET}$ signal of the MPC8266. Since MDDIS pin of the LXT970 is driven low with this application, the negation of this signal causes all the H/W configuration bits to be sampled for initial values and device control is moved to the MDIO channel, which is the control path of the MII port.	1	R,W
6	RS232EN_1	***RS232 port 1 Enable***. When asserted (**low**) the RS232 transceiver for port 1, is enabled. When negated, the RS232 transceiver for port 1, is in standby mode and SCC1 pins are available for off-board use via the expansion connectors.	1	R,W
7	RS232EN_2	***RS232 port 2 Enable***. When asserted (**low**) the RS232 transceiver for port 2, is enabled. When negated, the RS232 transceiver for port 2, is in standby mode and SCC2 pins are available for off-board use via the expansion connectors.	1	R,W
8 - 31	Reserved	Un-implemented	0	R

a. Required for voltage levels adaptation.

4.13.3 BCSR2 - Board Control - Status Register - 2

BCSR2 is a status register which is accessed at **offset 8** from the BCSR base address. Its a read-

only register which may be read at any time[1]. BCSR2s' various fields are described in Table 4-12.

Table 4-12. BCSR2 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 7	TSTAT(0:7)	***Tool Status (0:7)***. This field is reserved for external tool status report. The exact meaning of each bit within this field is tool unique and therefore will be documented separately per each tool. These signals are available at the System expansion connector.	-	R
8 - 11	TOOLREV(0:3)	***TOOL Revision (0:3)***. This field may contain the revision code of an external tool connected to the MPC8266. The various combinations of this field will be described per each tool users' manual. These signals are available at the System expansion connector. The revision option for the external tools are shown in Table 4-18.		R
12 - 15	EXTTOLI(0:3)	***External Tools Identification***. These lines, which are available at the CPM expansion connectors, are intended to serve as tools' identifier. On-board S/W may check these lines to detect The presence of various tools (h/w expansions) at the CPM expansion connectors. For the external tools' codes and their associated combinations see Table 4-15.	-	R
16 - 17	SWOPT(0:1)[a]	***Software Option (0:1)***. This field shows the state of a dedicated dip-switches providing an option to manually change a program flow.	0	R
18 - 19	L2CSIZE(0:1)	***L2 Cache Size (0:1)***. This filed encodes the size of the L2 Cache, present on the MPC8266-MB. For the encoding of the various cache sizes see Table 4-19.	-	R
20 - 21	BVERN(0:1)	***Board Version Number (0:1)***. This field represents the version code, hard-assigned to the MPC8266-MB. See Table 4-16., for version encoding.	11	R
22 - 23	BREVN(0:1)	***Board Revision Number (0:1)***. This field represents the revision code, hard-assigned to the MPC8266-MB. See Table 4-17., for revisions' encoding.	-	R
24	SWOPT2	***Software Option 2***. This is the LSB of the field. Shows the state of a dedicated dip-switch providing an option to manually change a program flow.	0	R
25 - 27	FLASH_PD(7:5)	***Flash Presence Detect(7:5)***. These lines are connected to the Flash SIMM presence detect lines, which encode the Delay of Flash SIMM mounted on the Flash SIMM socket. For the encoding of FLASH_PD(7:5) see Table 4-13.	-	R
28 - 31	FLASH_PD(4:1)	***Flash Presence Detect(4:1)***. These lines are connected to the Flash SIMM presence detect lines which encode the type of Flash SIMM mounted on the Flash SIMM socket. For the encoding of FLASH_PD(4:1) see Table 4-14.	-	R

a. There is additional bit to this field. See next on the same table.

1. Provided that BCSR is not disabled.

Table 4-13. FLASH Presence Detect (7:5) Encoding

FLASH_PD(7:5)	FLASH DELAY [nsec]
000	Not Supported
001	150
010	100/120
011	80/90
100	70
101 - 111	Not Supported

Table 4-14. FLASH Presence Detect (4:1) Encoding

FLASH_PD(4:1)	Flash TYPE / SIZE
0000	SM73288XG4JHBG0 - 32 MByte (4 banks of 4 X 2M X 8) by Smart Modular Technology.
0001	SM73248XG2JHBG0 - 16 MByte (2 banks of 4 X 2M X 8) by Smart Modular Technology.
0010	SM73228XG1JHBG0 - 8 MByte (1 bank of 4 X 2M X 8) by Smart Modular Technology.
0011 - 1111	Not Supported

Table 4-15. EXTOOLI(0:3) Assignment

EXTTOOLI(0:3)	External Tool
0	T/ECOM - MPC8266 Communication tool
1	Reserved
2	T1 Circuit Emulation Tool
3 - E	Reserved
F	Tool Non Existent

Table 4-16. PQ2 Board Version Encoding

Version Number (0:1) [Hex]	PQ2 Board Version
0	PQ2 - Voyager ADS
1	Reserved
2	MPC8266 - Add In Card
3	MPC8266 - Motherboard

Table 4-17. PQ2 Board Revision Encoding

Revision Number (0:1) [Hex]	PQ2 Board Revision
0	ENG (Engineering)
1	PILOT
2	A
3	Reserved

Table 4-18. External Tool Revision Encoding

TOOLREV(0:3) [hex]	External Tool Revision
0	ENGINEERING
1	PILOT
2	A
3 - F	Reserved

Table 4-19. L2 Cache Size Encoding

L2CSIZE(0:1)	L2 Cache Size
’00’	Reserved
’01’	512 KBytes
’10’	Reserved
’11’	No L2 Cache

4.13.4 BCSR4 - Board Control - Status Register 4

BCSR4 is a status register which is accessed at **offset 0xC** from the BCSR base address. Its a read- only register which may be read at any time[1]. BCSR4s' various fields are described in Table 4-20.

Table 4-20. BCSR4 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 1	PCI0_PRSNT(0:1)	***PCI Slot 0 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is pluged in PCI slot 0 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 4-21.	11	R
2 - 3	PCI1_PRSNT(0:1)	***PCI Slot 1 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is pluged in PCI slot 1 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 4-21.	11	R
4 - 5	PCI2_PRSNT(0:1)	***PCI Slot 2 Present (0:1)***. This field holds a code that tells whether a PCI expansion board is pluged in PCI slot 2 and the total power requirements of the board according to the PCI spec. The different expansion board types are listed in Table 4-21.	11	R
6	M66EN	***66MHz Enable***. This field shows if one of the expansion boards used is not capable of operating in 66MHz mode: '1' - All expansion boards are 66MHz capable '0' - One of the expansion boards is not 66MHz capable	1	R
7 - 31	Reserved	Un Implemented	-	-

Table 4-21. PCI Board Present Signal Definitions

PCIx_PRSNT (0:1) [Hex]	Expansion Configuration
0	Expansion board present, 7.5W maximum
1	Expansion board present, 25W maximum
2	Expansion board present, 15W maximum
3	No expansion board present

4.13.5 BCSR3 and BCSR5- Board Control - Status Register 3 & 5

BCSR3 and BCSR5 are additional control / status registers which may be accessed as a word at **offset 0x10 to 0x14** from BCSR base address. These registers are not implemented. They may be read or written but with no valid data nor any effect on the board. The description of BCSR3 and

1. Provided that BCSR is not disabled.

BCSR5 is shown in Table 4-22.

Table 4-22. BCSR3 and BCSR5 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 31	Reserved	Un Implemented	-	-

4.13.6 BCSR6 - Board Control / Status Register 6

BCSR 6 is used for the JTAG Fast Download I/F Status & Control. Although it resides only over D(0:7) lines of the PPC data bus, it is accessed as a **word** at **offset 0x18** from BCSR base. For additional information on that I/F see 4.14.1 "Fast Download Support" on page 78. The description of BCSR6 is shown in Table 4-23.:

Table 4-23. BCSR6 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0	JTAG_EN	***JTAG Enable***. When this bit is active (**High**) the JTAG machine for fast download is enabled for use. When inactive, TDI is asynchronously driven to MTDO. In this mode (Power-On default) COP controller S/W may operate on the ADS in ENG revision compatible mode.	0	R/W
1 - 6	Reserved	Implemented. Read as b'000000'. Writes have no effect.	0	R
7	JTAG_RX_FULL	***JTAG Receive Full Flag***. When this signal is active (**High**), it indicates, that the JTAG Download register, was fully written by the Host and should be read by the download agent running on the ADS. After the agent has read data from the Jtag Download Data Register, this bit is cleared. This bit is also cleared by either Power-On Reset, JTAG TAP Reset asserted (TRST~) and by JTAG TAP reset state. This bit is Read-Only, writing it has no effect.	0	R
8 - 31	Reserved	Un-Implemented.	-	-

4.13.7 BCSR7 - Board Control / Status Register 7

BCSR7 is used as the JTAG Fast Download I/F data register. Although it resides only over D(0:7) lines of the PPC data bus, it is accessed as **word** at **offset 0x1C** from BCSR base. During download, the host loads this register with serial data through the JTAG I/F. The download agent running on the board should, after polling the asserted JTAG_RX_FULL flag, read the data in this register and write it to the board's memory. BCSR7 is described in Table 4-24.:

Table 4-24. BCSR7 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 7	JTAG_DOWNLOAD_DATA	JTAG Data. Data shifted into the JTAG Download Data register, may be read here. This is a read-only field. Writes have no effect.	0	R
8 - 31	Reserved	Un-Implemented.	-	-

4.14 COP/JTAG Port

The COP - Control Observation Port, is part of the MPC8266's JTAG machine, implemented as a set of additional instructions and logic within the JTAG permissions. This port may be connected to a dedicated debug station[1], for extensive system debug.

There are several third party debug solutions on the market. These debug-stations may be connected to the host computer via either Ethernet, Parallel-Port, RS232 or any other media.

The debug station connection scheme is shown in Figure 4-11..

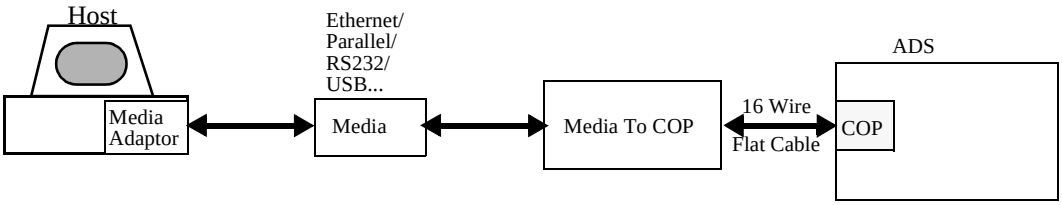


Figure 4-11. Debug Station Connection Schemes

To support debug station connection to the COP/JTAG port, a 16 pin generic header connector is provided on the MPC8266-MB, carrying the COP/JTAG signals as well as additional signals aiding in system debug. The pinout of this connector, which is a general Motorola recommendation for including a COP/JTAG port in a design, is shown in Figure 4-12. and detailed in Table 4-25..

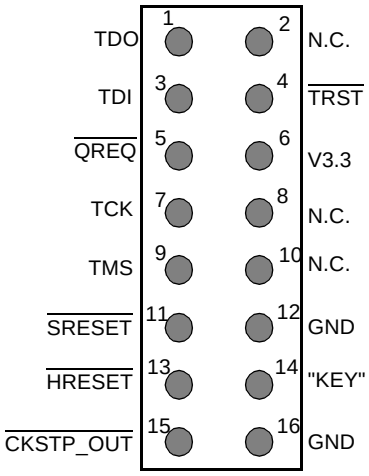


Figure 4-12. COP/JTAG Port Connector

1. Not provided with the MPC8266-MB.

Table 4-25. COP/JTAG Port Signals Description

Pin No.	Signal Name	Attribute	Description
1	TDO	O	Transmit Data Out. This the JTAG's serial data output driven by Falling edge of TCK.
2	N.C.	-	Not Connected.
3	TDI	I	Transmit Data In. This is the JTAG serial data input, sampled by the MPC8266 on the rising edge of TCK. This line is pulled up internally by the MPC8266.
4	$\overline{\text{TRST}}$	I	Test port Reset (L). When this signal is active (Low), it resets the JTAG logic. This line is pull-down on the MPC8266-MB with a 1KΩ resistor, to provide constant reset of the JTAG logic.
5	$\overline{\text{QREQ}}$	O	Quiescent Request (L). When asserted (low), this line indicates that the MPC8266 desires to enter low-power mode. This signal may be required by a debug station.
6	V3.3	O	3.3V power supply bus.
7	TCK	I	Test port Clock. This clock shifts in / out data to / from the MPC8266 JTAG port. Data is driven on the falling edge of TCK and is sampled both internally and externally on it's rising edge. TCK is pulled up internally by the MPC8266.
8	N.C.	-	Not Connected.
9	TMS	I	Test Mode Select. This signal qualified with TCK in a same manner as TDI, changes the state of the JTAG machine. This line is pulled up internally by the MPC8266.
10	N.C.	-	Not Connected.
11	$\overline{\text{SRESET}}$	I/O, O.D.	Soft Reset (L). This is the MPC8266's soft reset which is in fact a non-maskable interrupt, making the PowerPC take the reset exception from the reset vector. This line may be driven by the MPC8266 as well during soft-reset sequence, for 512 system clocks. This line is pulled up on the MPC8266-MB with a 1KΩ resistor. When driven externally, it **MUST be driven with an Open Drain gate. Failure in doing so might result in permanent damage to the MPC8266 and / or to board logic.**
12	GND	O	Digital GND. Main GND plane.
13	$\overline{\text{HRESET}}$	I/O, O.D.	MPC8266's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the MPC8266. During that sequence, asserted by the MPC for 512 system clocks. Pulled Up on the MPC8266-MB using a 1KΩ resistor. When driven by an external tool, **MUST be driven with an Open Drain gate. Failure in doing so might result in permanent damage to the MPC8266 and / or to board logic.**
14	N.C.	-	Not Connected.

Table 4-25. COP/JTAG Port Signals Description

Pin No.	Signal Name	Attribute	Description
15	$\overline{\text{XBR3}}$ ($\overline{\text{CKSTOP\_OUT}}$)	I/O	Normally configured as $\overline{\text{XBR3}}$ which has no function with this connector. May be configured as $\overline{\text{CKSTOP\_OUT}}$ - Check Stop Out (L). When asserted (Low) indicates that the MPC8266 core has entered a Check-Stop state.
16	GND	O	Digital GND. Main GND plane.

4.14.1 Fast Download Support

Download rates through the COP port are inherently slow due to very long COP scan chains and the extraneous amount of data transferred. In essence, an additional (to MPC8266's) JTAG machine was added in front of the MPC8266's JTAG port. This machine supports the minimal (public) set of JTAG instructions, required by JTAG rules to be a part of a JTAG chain, while having in fact, zero pins[1]. This machine includes a serial to parallel interface - the serial part is driven by JTAG, while the parallel is mapped into the PPC bus (embedded in BCSR's memory space). This configuration allows zero data overhead during downloads. The block diagram of the JTAG system on the board is shown in Figure 4-13..

1. No Boundary Scan pins, Excluding the JTAG I/F pins, which are not count for that matter.

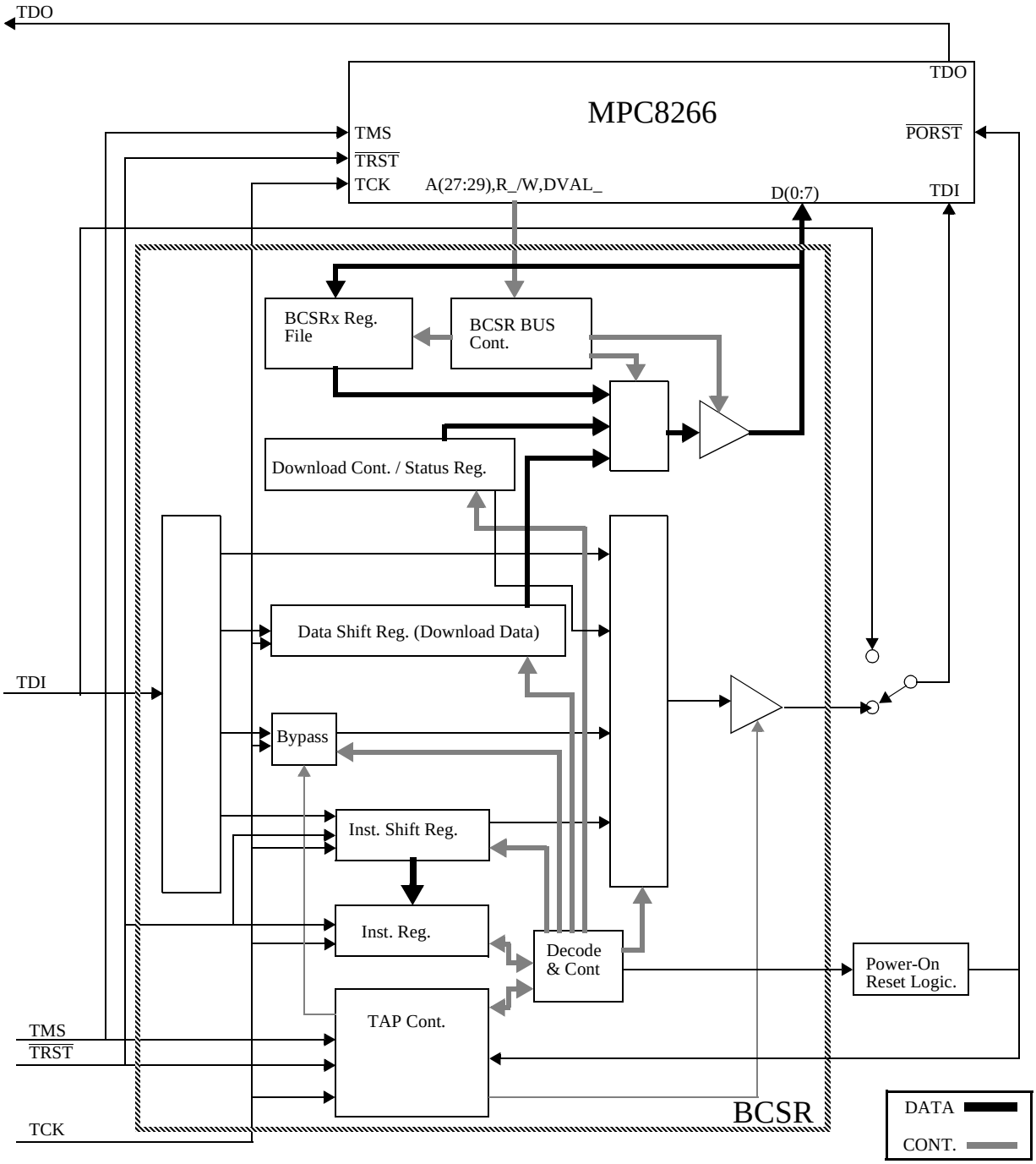


Figure 4-13. Fast Download JTAG System

As can be seen in the figure above, the JTAG machine includes the following components:

1. TAP Controller
2. Instruction Shift register
3. Instruction register

4. Data Shift register (Download Data register)
5. Download Command and Status register
6. Bypass register

4.14.1.1 JTAG TAP Controller

The TAP (Test Access Port) controller is a standard 16-state JTAG TAP controller. Its transitions are determined by the state of TMS upon rising edge of TCK. The state “Test Logic Reset” may be reached synchronously (according to rising TCK) from any state by simply driving TMS high for 5 consecutive rising edges of TCK. The test logic is also reset asynchronously by either driving $\overline{\text{TRST}}$ low by the debug station or by Power On Reset generated by the board’s Power-Up, or SW1 depressed, or by this logic[1].

The TAP controller state diagram is shown in Figure 4-14..

1. In a “Suicidal” manner.

.

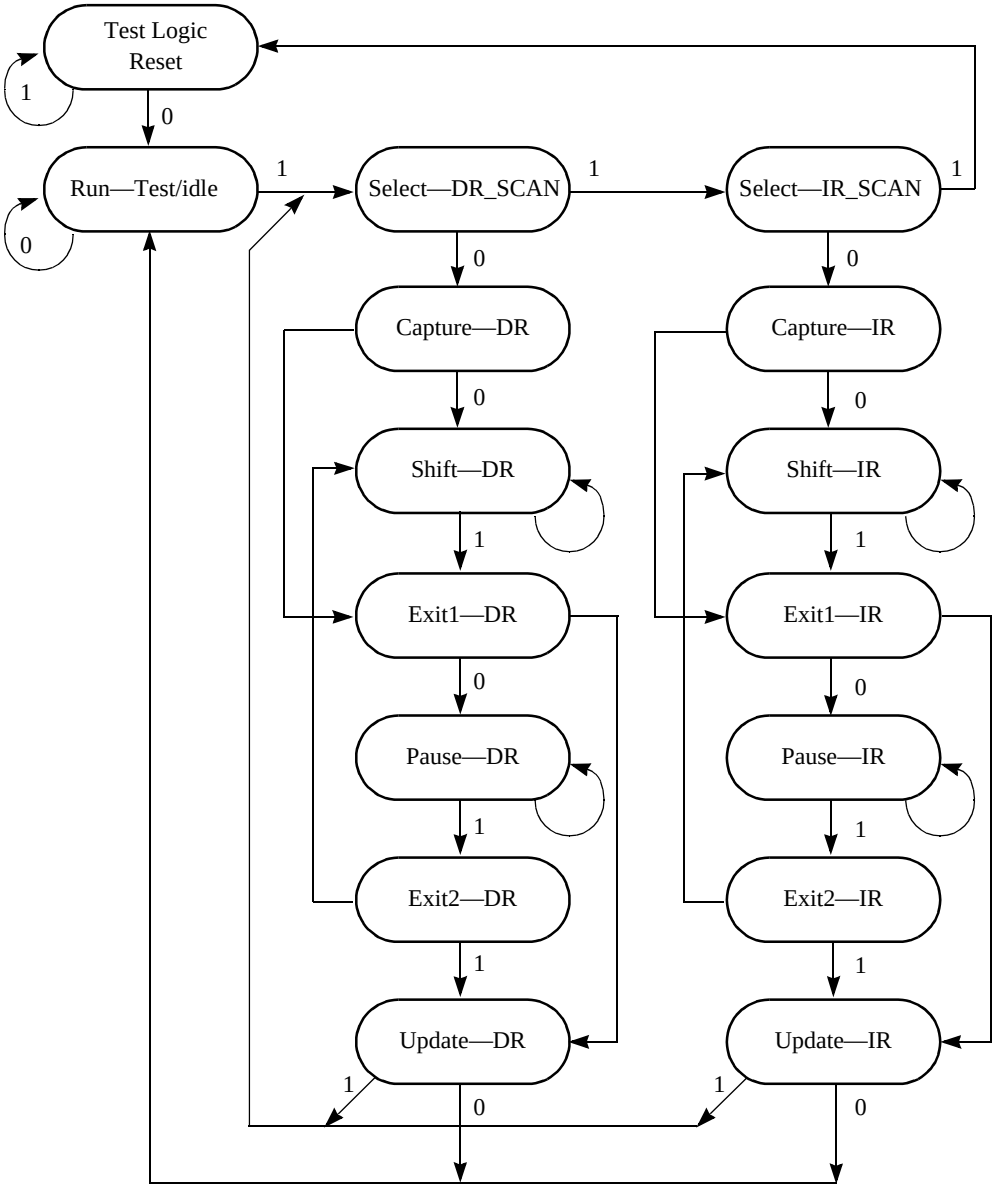


Figure 4-14. JTAG TAP Controller State Diagram

4.14.1.2 JTAG Instruction Shift Register - JISR

The instruction shift register is a 3 bit register, selected during Shift-IR state of the TAP controller. The instruction is shifted in during that state, while the output of the shift register (LSB) is driven into MTDO, to be concatenated to the next device on the JTAG chain (the MPC8266). That way, all devices on the JTAG chain may be shifted-in with the desired instruction for them. When the TAP controller moves into Update-IR state, the JTAG Instruction register is loaded with the value shifted into JISR.

During JTAG logic reset, the JISR, is reset into a default state of Bypass.

4.14.1.3 JTAG Instruction Register - JIR

The JTAG Instruction register holds the current JTAG instruction, which determines the JTAG logic operation at any given time. The instructions are loaded into it from JISR when the TAP enters the Update-IR state. The valid instructions and their associated functions are shown in Table 4-26.:

Table 4-26. JTAG Instruction Codes

Mnemonic	Code [Bin]	Function
EXTEST	'000'	**Execute Test**. JTAG public instruction. No function with this application, since there is no built-in test within this JTAG entity. **Defaults to Bypass**.
DOWNLOAD	'001'	**Download**. When JIR holds this instruction and the TAP controller is in Shift-DR mode, the input of the Download Shift register is connected to TDI - of the board, while the TDO of this machine reflects the status of JTAG_RX_FULL flag in the Download Control and Status register. Data may be shifted-in to be read by the download agent running on board. When the TAP controller moves into EXIT1-DR state, the Receive-Full flag is set in the Download Control & Status register, so that the download agent may learn about the presence of valid data in the JDSR.
UPLOAD	'010'	**Up-Load**. Not supported with current revision, **defaults to Bypass**.
SAMPLE/ PRELOAD	'011'	**Sample/Preload**. JTAG public instruction. Not implemented, since there are no external pins to this JTAG entity. **Defaults to Bypass**.
Reserved.	'100', '101'	**Reserved**, un-implemented, **defaults to Bypass**.
PON_RESET	'110'	**Power-On Reset**. When this code is loaded into the JIR, Power-On Reset is generated to the board, eventually resetting the TAP controller into Test-Logic Reset state. As a result the JIR is reset into Bypass code.
BYPASS	'111'	**Bypass**. JTAG public instruction. When the JIR holds this value and the TAP controller is in Shift-DR state, a single bit shift register is placed between the TDI-TDO of this JTAG machine.

4.14.1.4 Data Shift Register

The data shift register is an 8 bit shift register, shifted in (LSB first) on the rising edge of TCK. When JIR contains the DOWNLOAD code and the TAP controller is in Shift-DR state, the input of this shift register is connected to TDI input of the board. The TDO of this logic on the other hand, reflects the state of the JTAG_RX_FULL status bit in the Download Control & Status register. That way the host may check, prior to shifting in data, whether the agent has read the previous byte of data.

When JTAG_RX_FULL flag is active, data remains frozen in the register, until it is read by the agent. That way the host may shift in arbitrary data just to read back the status of JTAG_RX_FULL. Since that flag is set only by the passage through EXIT1_DR, this register will contain the last 8 bits shifted in.

This register is available on the PPC-Bus memory map D(0-7), so it may be read by a download agent running on-board. It is designated as BCSR7. For additional information on BCSR7 see Table 4-24. and Table 4-24..

4.14.1.5 Download Control and Status Register

This register has no direct JTAG access, however, it enables the operation of this machine and contains status information, set by this machine. It is available on the PPC-Bus memory map, designated as BCSR6. For further information on BCSR6, see Table 4-23. and Table 4-23..

4.14.1.6 Bypass Register

The bypass register is a single stage register which must exist in any JTAG implementation. Its purpose, is to shorten the scan chain as much as possible for a device residing on the scan chain. When the JIR contains the BYPASS code and the TAP controller is in Shift-DR state, this register is placed between TDI-TDO pair of this machine.

Any un-implemented instruction with this machine, defaults to the BYPASS instruction.

4.14.1.7 JTAG Machine Bypass

There are 3 levels of bypass for this JTAG implementation, this to provide compatibility with earlier versions of this board and debug tools. The bypass options are:

1. **Hard-wired Bypass**: When a jumper is set between positions 2-3, then the TDI input to the VADS, is connected directly to the TDI input of the MPC8266. This allows compatibility with existing debug tools which do not support the use of this machine and might suffer from added delay on TDI.
2. **Asynchronous Bypass**: After Power-On Reset, when a jumper is set between positions 1-2 (factory set), this machine defaults to asynchronous connection between TDI input of this machine and the MPC8266's TDI (with a 7.5 nsec delay). This allows compatibility with existing debug tools which have not been modified yet to support this machine and can tolerate this delay on the TDI line.
3. **JTAG Bypass**: After Power-On reset or JTAG reset, when this JTAG machine is enabled via BCSR6 and the jumper is set between positions 1-2 (factory set), it will be found in Bypass mode, i.e., the default instruction of the machine is BYPASS, so that when the TAP controller is moved into Shift-DR state, a single stage shift register is placed between the TDI input of the ADS and the TDI input of the MPC8266.

4.14.1.8 Fast Download Operation

This section describes the procedure needed to be taken by a debug-station programmer, so that this machine may be utilized. It is assumed here that the jumper remains factory-set (1-2) and the board is after Power-On reset and initialized. The procedure is as follows:

1. Enable this JTAG machine by writing '1' to the JTAG_EN bit in BCSR6. The machine is now enabled and in JTAG Bypass mode. See Table 4-23.. Remember that prior to this operation, the length of the instruction chain is 8 bit (MPC8266 only) while the data scan chain's length depended only on the scan selected within the MPC8266. After this operation, the length of the instruction chain is 11 bits (added 3 bits for this machine, preceding the MPC8266's in the chain), while the length of the data scan chain is added with either 1 bit (bypass) or 8 bit (download), preceding the MPC8266 in the chain (MSB side).
2. When this machine is in Bypass, download a s/w agent to free memory space. Remember that the data scan chain is 1-bit longer than it used to be (MSB side). This agent basically polls the JTAG_RX_FULL flag in the Download Command & Status register (see Table 4-

23.), when active - reads the Download Data register (See Table 4-24.) and puts the data byte read, where required. Obviously the minimum it should know, are the base address and size of data buffer being loaded, however its level of sophistication is upto the system programmer. Run this agent, again, using the same method.

3. Shift in DOWNLOAD instruction for this machine and BYPASS instruction for the MPC8266.
4. Move the TAP controller into Shift-DR state
5. Shift in a Byte of data. The data shifted out, is the state of JTAG_RX_FULL flag. If the byte shifted out does not contain a '0', it indicates that new data was shifted in before the agent was able to read the previous. That way the host may become aware of an error situation with the operation of the s/w agent. To play safe, it is recommended to shift in a arbitrary data until JTAG_RX_FULL is found cleared. Then a byte of valid data may be shifted-in.
6. Move the TAP to EXIT1-DR -> PAUSE-DR -> EXIT2-DR. The passage through EXIT1-DR sets the JTAG_RX_FULL flag in the Download Command & Status register. The agent then, can read the valid data from the Download Data register. After that data has been read by the agent, the JTAG_RX_FULL is cleared.
7. Repeat steps (4) to (6) above until the end of the buffer.

4.14.1.9 JTAG Generated Power-On Reset

Since some of the COP debug-stations are Ethernet driven, a need may arise to generate Power-On Reset from a remote location, through the JTAG and to be able to do so, when the board is stuck. To support such action, the PON_RESET instruction was introduced.

When JTAG is enabled in the Download Control & Status register, it is possible[1] to Power-On Reset the board through JTAG. The way to do it is:

Move the TAP controller into Shift-IR

Shift in PON_RESET code.

As a result of the above Power-On reset is generated, resetting the board including this JTAG machine, which goes back into "Test Logic Reset" state. The JTAG_EN bit in the Download Control & Status register is reset as well.

Before re-initializing the board, there is a need to wait about 3 seconds, for the board to recover. The state of $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ lines of the MPC8266, is visible via the COP/JTAG connector.

4.15 Switches, Jumpers and Indicators

The switches and jumpers on the MPC8266-MB include the following:

1. Since $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ lines appear at the COP/JTAG connector, it is possible to generate Hard-Reset and Soft-Reset, directly. Power-On reset however, may be remote generated only through JTAG.

1. Power-On Reset push button
2. Soft - Reset push-button
3. Abort push - button
4. Three dip-switches for Wake-up clock mode setting
5. Three dip-switches for S/W options
6. Three dip-switches for SDRAM DIMM Configuration Memory I^2C Slave Address
7. A dip-switch for selecting the source of the default Power-On Reset configuration word and PCI configuration - Flash or E^2PROM.

Since most of the board's logic is controlled via the BCSR and it is not visible whether a module is enabled or disabled, each module has a visible enable indicator. The following indicators are available on the MPC8266-MB:

1. 5V Power
2. 3.3V Power
3. SDRAM (60X Bus) enabled
4. Flash memory enabled
5. BCSR On Map
6. Run led
7. ATM Port Enabled
   - Transmit
   - Receive
8. Ethernet Port Enabled
   - Speed
   - Collision
   - Link Integrity
   - Receive
   - Transmit
9. RS232 Port 1 enabled
10. RS232 Port 2 enabled

5

Memory Map and Initialization

5.1 Memory Map

All accesses to MPC8266ADS-PCI's memory slaves are controlled by the MPC8266's memory controller. Therefore, the memory map is reprogrammable to the desire of the user. After Hard Reset is performed by the debug station, the debugger checks for existence, size, delay and type of the SDRAM DIMM and FLASH memory SIMM mounted on board and decides on the assignments of $\overline{CS0}$ and $\overline{CS4}$ (E^2PROM and FLASH) and programs the memory controller accordingly. The SDRAM, E^2PROM and the FLASH memory, respond to all types of memory access i.e., problem / supervisory, program / data and DMA.

This memory map is a recommended memory map and since it is a "soft" map, devices' address may be moved about the map, to the convenience of any user. There are actually two memory maps which depend on the device assigned to $\overline{CS0}$ (regardless of the boot source). The memory address for the device assigned to $\overline{CS0}$ is always the same as determined in the Hard-Reset configuration word. Since the FLASH and E^2PROM require different memory spaces, different memory maps are devised for each case. For details see Table 3-1. and Table 3-2.

Table 5-1. MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device

Address Range	Memory Type	Device Name		Port Size	Memory Size
00000000 - 00FFFFFF	SDRAM DIMM	SDCUV6482 (16 MByte)	SDC8UV6484 (64 MByte)	64	64 MByte
01000000 - 03FFFFFF					
04000000 - 044FFFFF	Empty Space			-	5 MByte

Table 5-1. MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device

Address Range	Memory Type	Device Name	Port Size	Memory Size
04500000 - 04507FFF	BCSR(0:7)[a]		32	32 KByte
04500000 - 04507FE3	BCSR0			4 Byte
04500004 - 04507FE7	BCSR1			4 Byte
04500008 - 04507FEB	BCSR2			4 Byte
0450000C - 04507FEF	BCSR3			4 Byte
04500010 - 04507FF3	BCSR4			4 Byte
04500014 - 04507FF7	BCSR5			4 Byte
04500018 - 04507FFB	BCSR6			4 Byte
0450001C - 04507FFF	BCSR7			4 Byte
04508000 - 045FFFFF	Empty Space		-	~1 MByte
04600000 - 04607FFF[b]	ATM UNI Proc. Control	PMC5350 M/P I/F	8	32 KByte
04608000 - 046FFFFF	Empty Space		-	~1 MByte
04700000[c] - 0471FFFF	MPC8266 Internal MAP[d]		32	128 KByte
04720000 - 0472FFFF	Empty Space		-	64 KByte
04730000 - 04737FFF	PCI Interrupt Controller		32	32 KByte
04738000 - 047FFFFF	Empty Space		-	~800 KByte
04800000 - 04FFFFFF	PCI Memory	Agents PIMMR (via PCI Direct)		~ 8 MByte
05000000 - 7FFFFFFF	Empty Space	Tool Board is located at 60000000 and 70000000		~ 2 GByte
80000000 - BFFFFFFF	PCI Memory	PCI Agents GPL WIndows	32	1 Gbyte

Table 5-1. MPC8266ADS-PCI Memory Map - FLASH (or BCSR) as Boot Device

Address Range	Memory Type	Device Name			Port Size	Memory Size
C0000000 - C1FFFFFF	Empty Space				-	32 MByte
C2000000[e] - C2007FFF	E^2PROM	ATMEL AT28HC64B			8	32 KByte
C2008000 - FDFFFFFF	Empty Space					~1 GByte
FE000000[f] - FEFFFFFF	Flash SIMM			32M SIMM - SM73288	32	32 MByte
FF000000 - FF7FFFFF			16M SIMM - SM73248			
FF800000 - FFFFFFFF		8M SIMM - SM73228				

a. The device appears repeatedly in multiples of its port-size (in bytes) X depth. E.g., BCSR0 appear at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.
b. The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size tha may be controlled by the GPCM is 32 KBytes.
c. Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.
d. Refer to the MPC8266 User's Manual for complete description of the internal memory map.
e. An 8 Kbyte device is used (16 Kbyte and 32 Kbyte devices can also be used) so it appears repeatedly in 8Kbyte multiples starting from C2000000.
f. Set by hard-reset configuration.

Table 5-2. MPC8266ADS-PCI Memory Map - E^2PROM as Boot Device

Address Range	Memory Type	Device Name		Port Size	Memory Size
00000000 - 00FFFFFF	SDRAM DIMM	SDCUV6482 (16 MByte)	SDC8UV6484 (64 MByte)	64	64 MByte
01000000 - 03FFFFFF					
04000000 - 044FFFFF	Empty Space			-	5 MByte

Table 5-2. MPC8266ADS-PCI Memory Map - E^{2}PROM as Boot Device

Address Range	Memory Type	Device Name	Port Size	Memory Size
04500000 - 04507FFF	BCSR(0:7)[a]		32	32 KByte
04500000 - 04507FE3	BCSR0			4 Byte
04500004 - 04507FE7	BCSR1			4 Byte
04500008 - 04507FEB	BCSR2			4 Byte
0450000C - 04507FEF	BCSR3			4 Byte
04500010 - 04507FF3	BCSR4			4 Byte
04500014 - 04507FF7	BCSR5			4 Byte
04500018 - 04507FFB	BCSR6			4 Byte
0450001C - 04507FFF	BCSR7			4 Byte
04508000 - 045FFFFF	Empty Space		-	~1 MByte
04600000 - 04607FFF[b]	ATM UNI Proc. Control	PMC5350 M/P I/F	8	32 KByte
04608000 - 046FFFFF	Empty Space		-	~1 MByte
04700000[c] - 0471FFFF	MPC8266 Internal MAP[d]		32	128 KByte
04720000 - 0472FFFF	Empty Space		-	64 KByte
04730000 - 04737FFF	PCI Interrupt Controller		32	32 KByte
04738000 - 047FFFFF	Empty Space		-	~800 KByte
04800000 - 04FFFFFF	PCI Memory	Agents PIMMR (via PCI Direct)		~ 8 MByte
05000000 - 7FFFFFFF	Empty Space	Tool Board is located at 60000000 and 70000000		~ 2 GByte
80000000 - BFFFFFFF	PCI Memory	PCI Agents GPL WIndows	32	1 Gbyte

Table 5-2. MPC8266ADS-PCI Memory Map - E^{2}PROM as Boot Device

Address Range	Memory Type	Device Name			Port Size	Memory Size
C0000000 - C1FFFFFF	Empty Space				-	32 MByte
C2000000 - C2FFFFFF	Flash SIMM			32M SIMM - SM73288	32	32 MByte
C3000000 - C37FFFFF			16M SIMM - SM73248			
C3800000 - C3FFFFFF		8M SIMM - SM73228				
C4000000 - FFFFDFFF	Empty Space					~1 GByte
FFF00000[e] - FFFFFFFF	E^{2}PROM	ATMEL AT28HC64B			8	32 KByte

a. The device appears repeatedly in multiples of its port-size (in bytes) X depth. E.g., BCSR0 appears at memory locations 4700000, 4700020, 4700040..., while BCSR1 appears at 4700004, 4700024, 4700044... and so on.
b. The internal space of the ATM UNI control port is 256 bytes, however, the minimal block size that may be controlled by the GPCM is 32 KBytes.
c. Initially at h0F000000 - h0F00FFFF, set by hard reset configuration.
d. Refer to the MPC8266 User's Manual for complete description of the MPC8266's internal memory map.
e. An 8 Kbyte device is used (16 Kbyte and 32 Kbyte devices can also be used) so it appears repeatedly in 8Kbyte multiples starting from FFF00000.

5.2 MPC8266 Register Programming

The MPC8266 provides the following functions on the MPC8266ADS-PCI:

1. System functions which include:
   - PPC Bus SDRAM Controller
   - Local Bus Host to PCI Bridge
   - Chip Select generator.
2. Communication functions which include:
   - ATM SAR
   - Fast Ethernet controller.
   - UART for terminal or host computer connection.

The internal registers of the MPC8266 must be programmed after Hard reset as described in the following paragraphs. The addresses and programming values are in **Hexadecimal** base.

For more information on the following initializations, see the MPC8266 User's Manual.

5.2.1 System Initializations

The Power-On Reset Configuration word is set in the FLASH or in the E^{2}PROM. There are two configuration words - one for the FLASH (when it is assigned to $\overline{CS0}$) and the other to the E^{2}PROM (when it is assigned to $\overline{CS0}$). The two configurations are detailed in Table 3-3. and Table 3-4. respectively.

Table 5-3. FLASH Power On Reset Configuration[a]

Flash Address [hex]	Init Value[hex]	Description
0	0C / (1C[b])	Internal arbitration, Internal memory controller, Core enabled, Single MPC8266 (60X Bus mode[b]), 32 Bit boot port size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master.
8	B2	L2cache signals configured as BADDRx lines, DP(1:7) configured as L2 cache I/F and IRQ(6:7),Initial internal space @ 0x0F000000
10	36	Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ7}$ and $\overline{CS11}$ as $\overline{CS11}$.
18	60	$\overline{CS10}$ configured as $\overline{BCTL1}$, PCI autoload from FLASH

a. Programmed into the Flash (E^{2}PROM) memory in addresses 0x0, 0x8, 0x10 & 0x18
b. With L2 Cache

Table 5-4. E^{2}PROM Power On Reset Configuration[a]

EEPROM Address [hex]	Init Value[hex]	Description
0	04 / (14[b])	Internal arbitration, Internal memory controller, Core enabled, Single MPC8266 (60X Bus mode[b]), 8 Bit Boot size, Exceptions vectored to 0xFFFxxxxx, Internal space 64 bit slave for external master.
8	B2	L2cache signals configured as BADDRx lines, DP(1:7) configured as L2 cache I/F and IRQ(6:7),Initial internal space @ 0x0F000000
10	36	Boot memory space @ 0xFE000000 - 0xFFFFFFFF, $\overline{ABB/IRQ2}$ pin is $\overline{ABB}$, $\overline{DBB/IRQ3}$ pin is $\overline{DBB}$, No masking on bus request lines, Local bus pins function as PCI, PCI is boot master, AP(1;3) configured as BNKSEL(0:2), $\overline{APE}$ configured as $\overline{IRQ7}$ and $\overline{CS11}$ as $\overline{CS11}$.
18	60	$\overline{CS10}$ configured as $\overline{BCTL1}$, PCI autoload from EEPROM

a. Programmed into the E^{2}PROM in addresses 0x0, 0x8, 0x10 & 0x18
b. With L2 Cache

Table 5-5. SIU REGISTERS' PROGRAMMING

Register	Init Value[hex]	Description
RMR	0001	Check-Stop Reset enabled.
IMMR	04700000	Internal space @ 0x047000000
SYPCR	FFFFFFC3	Software watchdog timer count - FFFF, Bus-monitor timing FF, PPC Bus-monitor - Enabled, Local Bus-monitor - Enabled, S/W watch-dog - disabled, S/W watch-dog (if enabled) causes reset, S/W watch-dog (if enabled) - prescaled.
BCR	004C0000 (88444000[a])	Single MPC8266 (60X Bus mode[a]), 0 wait-states on address tenure, No L2Cache (L2Cache assumed[a]), 1 clock hit delay (when L2cache available), 1-level Pipeline depth, Extended transfer mode enabled for PCC, Extended transfer mode disabled for Local Buses, Odd parity for PPC & Local Buses, External Master delay enabled, Internal space responds as 64 bit slave for external master (not relevant for this application).

a. With L2 Cache

5.2.2 Memory Controller Registers Programming

The memory controller on the MPC8266ADS-PCI is initialized to 66 MHz operation, i.e., registers' programming is based on 66 MHZ timing calculation. There are two possible initializations for the memory controller:

- Flash SIMM is assigned to $\overline{CS0}$ and E^2PROM is assigned to $\overline{CS4}$.
- Flash SIMM is assigned to $\overline{CS4}$ and E^2PROM is assigned to $\overline{CS0}$.

Both options are shown in Table 3-6.and Table 3-7.

Table 5-6. Memory Controller Initializations For 66Mhz - FLASH as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR0	SM73228XG1JHBG0 by Smart Modular Tech.	PPC	FF801801	Base at FF800000, 32 bit port size, no parity, GPCM
	SM73248XG2JHBG0 by Smart Modular Tech.		FF001801	Base at FF000000, 32 bit port size, no parity, GPCM
	SM73288XG4JHBG0 by Smart Modular Tech.		FE001801	Base at FE000000, 32 bit port size, no parity, GPCM
OR0	SM73228XG1JHBG0 by Smart Modular Tech.		FF800836	8MByte block size, CS early negate, 6 w.s., Timing relax
	SM73248XG2JHBG0 by Smart Modular Tech.		FF000836	16MByte block size, CS early negate, 6 w.s., Timing relax
	SM73288XG4JHBG0 by Smart Modular Tech.		FE000836	32MByte block size, CS early negate, 6 w.s., Timing relax

Table 5-6. Memory Controller Initializations For 66Mhz - FLASH as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR1	BCSR	PPC	04501801	Base at 04500000, 32 bit port size, no parity, GPCM
OR1			FFFF8010	32 KByte block size, all types access, 1 w.s.
BR2	All SDRAM DIMM Supported	PPC	00000041	Base at 0, 64 bit port size, no parity, Sdram machine 1
OR2	SDC2UV6482C-84 by Fujitsu		FF000C80	16MByte block size, 2 banks per device, row starts at A9, 11 row lines, internal bank interleaving allowed, normal AACK operation
	SDC8UV6484C-84 by Fujitsu		FC002CC0	64MByte block size, 4 banks per device, row starts at A9, 12 row lines, internal bank interleaving allowed, normal AACK operation
BR3	Depends on the SDRAM DIMM.	PPC	TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
OR3	Depends on the SDRAM DIMM.		TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
BR4	E^2PROM	PPC	C2000801	Base at C2000000, 8 bit port size, write protect disabled, no parity, GPCM
OR4	AT28HC64B-70JC by Atmel		FFFF8846	32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 4 w.s., Timing relax
BR5	PM5350 - ATM UNI	PPC	04600801	Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus.
OR5			FFFF8E36	32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read.
BR8	PCI Interrupt Controller	PPC	04731801	Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus.
OR8			FFFF8010	32 KByte block size, all types access, 1 w.s.

Table 5-7. Memory Controller Initializations For 66Mhz - E^2PROM as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR0	E^2PROM	PPC	FFF00801	Base at FFFFE000, 8 bit port size, write protect disabled, no parity, GPCM
OR0	AT28HC64B-70JC by Atmel		FFFF8846	32 KByte block size, $\overline{CS}$ output half a clock after address, all types access, 4 w.s., Timing relax

Table 5-7. Memory Controller Initializations For 66Mhz - E^{2}PROM as Boot Device

Reg.	Device Type	Bus	Init Value [hex]	Description
BR1	BCSR	PPC	04501801	Base at 04500000, 32 bit port size, no parity, GPCM
OR1			FFFF8010	32 KByte block size, all types access, 1 w.s.
BR2	All SDRAM DIMM Supported	PPC	00000041	Base at 0, 64 bit port size, no parity, Sdram machine 1
OR2	SDC2UV6482C-84 by Fujitsu		FF000C80	16MByte block size, 2 banks per device, row starts at A9, 11 row lines, internal bank interleaving allowed, normal AACK operation
	SDC8UV6484C-84 by Fujitsu		FC002CC0	64MByte block size, 4 banks per device, row starts at A9, 12 row lines, internal bank interleaving allowed, normal AACK operation
BR3	Depends on the SDRAM DIMM.	PPC	TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
OR3	Depends on the SDRAM DIMM.		TBD	Not used for the type of SDRAM supplied with the board. Value is specific for an SDRAM DIMM.
BR4	SM73228XG1JHBG0 by Smart Modular Tech.	PPC	C3801801	Base at C3800000, 32 bit port size, no parity, GPCM
	SM73248XG2JHBG0 by Smart Modular Tech.		C3001801	Base at C3000000, 32 bit port size, no parity, GPCM
	ASM73288XG4JHBG0 by Smart Modular Tech.		C2001801	Base at C2000000, 32 bit port size, no parity, GPCM
OR4	SM73228XG1JHBG0 by Smart Modular Tech.		FF800836	8MByte block size, CS early negate, 6 w.s., Timing relax
	SM73248XG2JHBG0 by Smart Modular Tech.		FF000836	16MByte block size, CS early negate, 6 w.s., Timing relax
	SM73288XG4JHBG0 by Smart Modular Tech.		FE000836	32MByte block size, CS early negate, 6 w.s., Timing relax
BR5	PM5350 - ATM UNI	PPC	04600801	Base at 04600000, 8 bit port size, no parity, GPCM on PPC bus.
OR5			FFFF8E36	32K Byte block size, delayed CS assertion, early CS and WE negation for write cycle, relaxed timing, 7 w.s. for read, 8 for write, extended hold time after read.
BR8	PCI Interrupt Controller	PPC	04731801	Base at 04730000, 32 bit port size, no parity, GPCM on PPC bus.
OR8			FFFF8010	32 KByte block size, all types access, 1 w.s.

Table 5-8. Memory Controller Initializations For 66Mhz

Reg.	Device Type	Bus	Init Value [hex]	Description
PSDMR	SDC2UV6482C-84 (16 MByte)	PPC Single MPC8266 Bus Mode	416EB452	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C2AAB452	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A19 on BNKSEL2[a], A8 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
	SDC8UV6484C-84 (64 MByte)		412EB452	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(13 - 15) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C372B452	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A(16:17) on BNKSEL(1:2)[b], A6 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, no extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.

Table 5-8. Memory Controller Initializations For 66Mhz

Reg.	Device Type	Bus	Init Value [hex]	Description
	SDC2UV6482C-84 (16 MByte) **DIMM_SIZE** in BCSR0 should be **Cleared**.	PPC 60X Bus Mode	416EB45A **PBI** in BCSR0 should be **Cleared**	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C2AAB45A **PBI** in BCSR0 should be **Set**	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A19 on BNKSEL2[a], A8 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
	SDC8UV6484C-84 (64 MByte) **DIMM_SIZE** in BCSR0 should be **Set**.		412EB45A **PBI** in BCSR0 should be **Cleared**	**Bank** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 1, A(15 - 17) on BNKSEL(0:2), A9 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
			C372B45A **PBI** in BCSR0 should be **Set**	**Page** Based Interleaving, Refresh enabled, normal operation code, address muxing mode 2, A(16:17) on BNKSEL(1:2)[b], A6 on PSDA10, 7 clocks refresh recovery, 3 clocks precharge to activate delay, 2 clocks activate to read/write delay, 4 beat burst length, 1 clock last data out to precharge, 1 clock write recovery time, extra cycle on address phase, normal timing for control lines, 2 clocks CAS latency.
LSDMR		-	0	No SDRAM on Local Bus which is used for PCI Bus ONLY.
PSRT	All PPC Bus Sdram Supported	PPC	21	Divide MPTPR output by 34 (PSRT +1) Generates refresh every 13.4 µsec, while 16 µsec required. Therefore is refresh redundancy of 5.4 msec throughout full SDRAM refresh cycle which completes in 27.4 msec. I.e., Application s/w may withhold the bus upto app. 5.4 msec in a 32.8 msec period, without jeopardizing the contents of the ppc bus SDRAM DIMM.
LSRT		-	0	No SDRAM on Local Bus which is used for PCI Bus ONLY.
MPTPR	All SDRAMs on board		1900	Divide Bus clock by 26 (MPTPR+1) (decimal)

a. Although this BSMA value corresponds to A(17:19) on BKSEL(0:2), when PBI is set, only the relevant BKSEL lines, according to number of internal SDARM banks, are VALID, in this case BKSEL2.
b. Although this BSMA value corresponds to A(15:17) on BKSEL(0:2), when PBI is set, only the relevant BKSEL lines, according to number of internal SDARM banks, are VALID, in this case BKSEL(1:2)

6

Physical Properties

6.1 Power Supply

The board gets the power from the ATX Power Supply (it seats in an ATX Chassis). All the power rails on the board are derived from the ATX Power Supply. There are 4 power rails with the MPC8266:

1. VDDH (I/O)
2. VDDL (Internal Logic)
3. VCCSYN (CPM PLL)
4. VCCSYN1 (Core PLL)

and there are 5 power rails on the MPC8266-MB:

1. VCC (5V) rail
2. Stand By (5V) rail
3. V3.3 (3.3V) rail
4. VDDL (1.7V-2.5V) rail
5. +12V rail

6. -12V rail

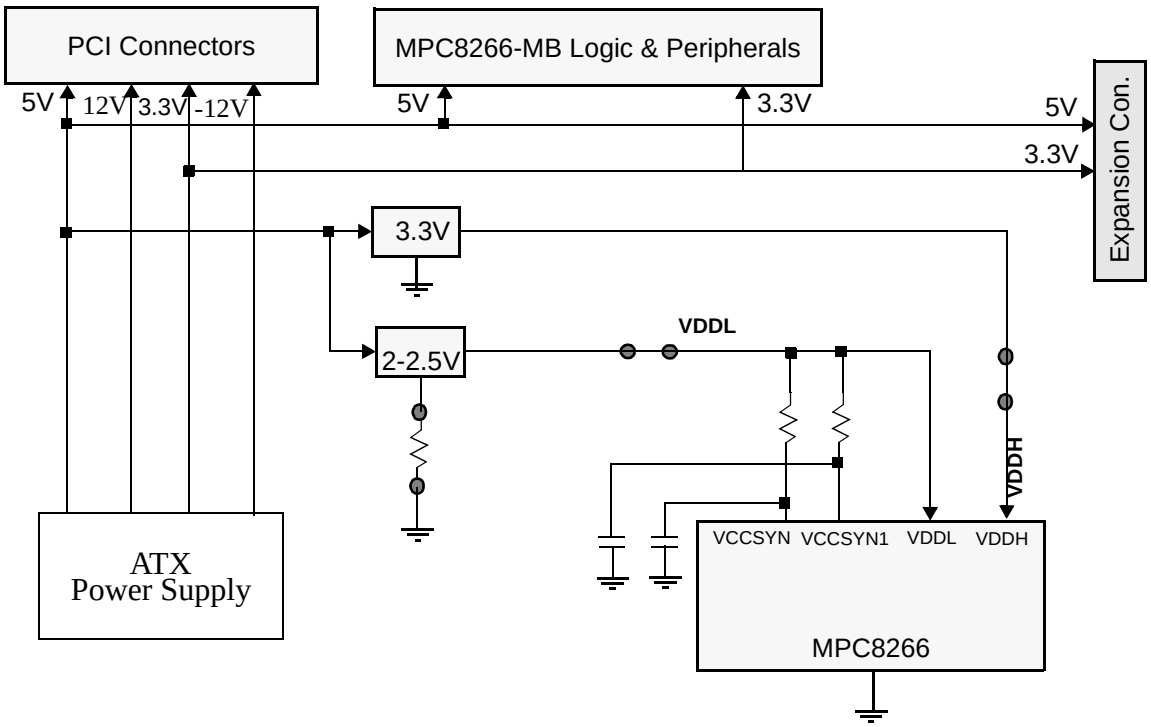


Figure 6-1. MPC8266-MB Power Scheme

To support off-board application development, the power buses are connected to the expansion connectors so that external logic may be powered directly from the board. The maximum current allowed to be drawn from the board on each bus also depends on the current drawn by the PCI bus. The figures are shown in Table 6-1.

Table 6-1. Expansion Connectors Maximum Current Consumption

Power Bus	Max. Current
VCC	TBD
V3.3	TBD

The PCI Standard specifies that each Add-In card should consume maximum 25Watt from all power sources combined. The maximum current consumption allowed per power source for a total of 25Watt according to the PCI Standard is shown in Table 6-2.

Table 6-2. Maximum Power Consumption Per Add-In Card

Power Rail	Add-In Card
5V	5A Max. (system depended)
3.3V	7.6A Max. (system depended)
12V	500mA
-12V	100mA

6.1.1 5V Rail

Some of the MPC8266-MB peripherals (not including the PCI Add-In cards which should be 3.3V ONLY on the PCI interface but can use 5V for other components on-board) reside on the 5V bus. Since the MPC8266 is not 5V tolerant, buffering is provided between 5V peripherals and the MPC8266, protecting the MPC8266 from the higher voltage level.

6.1.2 3.3V Rail

The MPC8266, SDRAM, PCI Add-In cards, address and data buffers are powered by the 3.3 bus, which is produced from the ATX power supply.

6.1.3 5V Stand By Rail

The 5V stand by power rail comes from the ATX Power Supply. Its' only use is to power the logic required to support the power button in the front panel on the ATX chasis.

6.1.4 VDDH Rail

The MPC8266's VDDH power bus (3.3V) is produced from the 5V bus using a low-voltage drop linear voltage regulator made by Micrel, the MIC29501-3.3BU.

A production option is made so that the level on this bus may be varied by means of trimming potentiometer - TR2. However this will requires replacing some components. This option allows the VDDH to be in the range of 3.0V - 3.6V.

6.1.5 VDDL Bus

The MPC8266's internal logic and the PLL are powered with a lower-voltage power source, voltage of which may be in 3 ranges of levels:

- 2.3V - 2.7V
- 1.7V - 1.9V
- 1.8V - 2.0V

Selection between the above range levels is done via a jumper, which selects between different resistor values within the VDDL's variable regulator feedback network, while the fine tuning within a range is done by means of a trimming potentiometer.

Changing the voltage to the Core logic of the MPC8266, obviously has an influence over the maximal speed of the core. There is the power-speed trade-off, i.e., lower operation speeds may be obtained with lower voltage supply.

6.1.6 12V Rail

The 12V bus from the ATX Power Supply supports the PCI slots and the VPP 12V option from programming the FLASH.

6.1.7 -12V Rail

The -12V bus from the ATX Power Supply supports the PCI slots.

6.2 Connectors

The MPC8266-MB has connectors attached, to serve the following functions:

1. ATX Power Supply
2. 100 / 10 - Base-T Ethernet port
3. ATM 155Mbps port
4. RS232 port 1
5. RS232 port 2
6. CPM Expansion
7. COP / JTAG
8. Logic Analyzer Connectors
9. Programmable logic In System Programming (ISP)
10. PCI Connectors
11. System Expansion

6.2.1 ATX Power Connector

The ATX power connector is a 20-lead, standard ATX power connector. The female part is soldered to the PCB, while the plug is connected to the power supply. That way fast connection / disconnection of power is facilitated.

6.2.2 Fast Ethernet Port Connector

The Ethernet connector on the MPC8266-MB is a Twisted-Pair (100/10-Base-T) connector. Use is done with 90^0 RJ45-8 connector.

6.2.3 ATM 155 Port Connection

The ATM 155 I/F to the media, is optical rather than electrical. Use is done with HP's HFBR 5205 optical I/F which is placed on the edge of the board for convenient connection.

6.2.4 RS232 PortS Connector

The RS232 port connector is a stacked 9 pin, 90^0, female D-Type connector, which saves on board space (made of two connectors for two ports).

6.2.5 CPM Expansion Connector

The CPM expansion connectors carries all CPM pins, i.e., Port A to Port D signals. Use done with DIN 41612, 128 pin T.H. PCB connector, residing on the board, allowing convenient vertical connection to off-board tools. Power supply pins are also provided through this connector.

6.2.6 COP/JTAG Port Connector

The debug port connector is a Motorola standard COP/JTAG connector for the 60X processors family.It is a generic 16 pin (2 X 8), Male, SMD, 90^o protected header connector.

6.2.7 Logic Analyzer Connectors

To support fast connection to HPs' 16500 Logic Analyzers series for debugging purposes, a set of dedicated connectors is provided. Use is done with 38 pin, SMT, high density, matched impedance MICTOR connectors made by AMP.

These connectors carry the unbuffered 60X signals and should be placed as near to the MPC8266 as possible to provide short PCB routes, yielding better reflections and crosstalk immunity. They do not carry the PCI bus signals due to the restrictions enforced by the PCI Standard. There are also connectors for the CPM signals.

6.2.8 Mach's In System Programming (ISP) Connector

This is a 10 pin generic 0.100" pitch header connector, providing In System Programming capability for Vantis made programmable logic on board.

6.2.9 PCI Connectors

A set of three standard PCI 3.3V keyed, 124 pin, 32-bit connectors is provided for connecting up to three PCI Add-In cards.

6.2.10 System Expansion Connector

The System Expansion Connector is a 128 pin, DIN 41612 connector, which provides a minimal system I/F required to interface to other tool-boards which may use the CPM Expansion Connector. This connector contains 16 bit (lower PPC bus) address lines, 16 bit (higher PPC bus) Data lines plus useful GPCM and UPM control lines.

6.3 PCB Layout

The MPC8266-MB layout was done in a manner suitable for high-frequency operation and it follows closely the PCI Standard layout recommendations. Following is a list of measures which are taken to meet this design goal:

- Traces are as short as possible.
- Clock signals and sensitive strobe signals are shielded and routed as a chain.
- Multilayer PCB, with ground and supply layers.
- PCI signals lengths and impedance according to PCI Standard Rev. 2.2.

7

Support Information

In this chapter all information needed for support, maintenance and connectivity to the MPC8266-ADS-PCI is provided.

7.1 Interconnect signals

The MPC8266ADS-PCI interconnects with external devices via the following set of connectors:

1. P1 - RS232 ports 1 and 2
2. P2 - 100 / 10 - Base-T Ethernet port
3. P3 - COP / JTAG
4. P4 - CPM Expansion
5. P5, P6, P10, P11, P13, P14, P16, P18, P19 - Logic Analyzer MICTOR Connectors
6. P7, P8, P9 - PCI Slots Connectors
7. P12 - ATX Power Supply Connector
8. P15 - Mach/Lattice In System Programming (ISP)
9. P17 - System Expansion

7.1.1 P1 - RS232 ports 1 and 2 Connectors

P1 is a dual 9 Pin D-Type connectors as described in Table 7-1.

Table 7-1. P1 Connector

Pin No.	Signal Name	Description
1	CD	Carrier Detect **output** from the MPC8266ADS-PCI.
2	TX	Transmit Data **output** from the MPC8266ADS-PCI.
3	RX	Receive Data **input** to the MPC8266ADS-PCI.
4	DTR	Data Terminal Ready **input** to the MPC8266ADS-PCI.
5	GND	Ground signal of the MPC8266ADS-PCI.
6	DSR	Data Set Ready **output** from the MPC8266ADS-PCI.
7	N.C.	No connect
8	CTS	Clear To Send **output** from the MPC8266ADS-PCI.

Table 7-1. P1 Connector

Pin No.	Signal Name	Description
9	N.C.	No connect

7.1.2 P2 - 100/10 - Base-T Ethernet port Connector

P2 is a RJ-45 Type Connector for Twisted Pair Ethernet as described in Table 7-2.

Table 7-2. P2 - 100/10 Base-T Ethernet Connector

Pin No.	Signal Name	Description
1	TPTX	Twisted-Pair Transmit Data positive output from the MPC8266ADS-PCI.
2	TPTX~	Twisted-Pair Transmit Data negative output from the MPC8266ADS-PCI.
3	TPRX	Twisted-Pair Receive Data positive input to the MPC8266ADS-PCI.
4	N.C.	Not connected, Bob Smith terminated on the MPC8266ADS-PCI.
5		
6	TPRX~	Twisted-Pair Receive Data negative input to the MPC8266ADS-PCI.
7	N.C.	Not connected, Bob Smith terminated on the MPC8266ADS-PCI.
8		

7.1.3 P3 - COP / JTAG Connector

P7 is a Motorola standard COP / JTAG connector for the 60X processors family. It is a 16 pin protected header connector as described in Table 7-3.

Table 7-3. P3 - COP/JTAG Connector

Pin No.	Signal Name	Attribute	Description
1	TDO	O	Transmit Data Output. This the MPC8266's JTAG serial data output driven by Falling edge of TCK.
2	GND	O	Digital GND. Main GND plane.
3	TDI	I	Transmit Data In. This is the JTAG serial data input of the ADS, sampled on the rising edge of TCK.
4	TRST#	I	Test port Reset~ (L). When this signal is active (Low), it resets the JTAG logic of the MPC8266. This line is pull-down on the ADS with a 1KΩ resistor, to provide constant reset of the JTAG logic.
5	QREQ#	O	Quiescent Request (L). When asserted (low), this line indicates that the MPC8266 desires to enter low-power mode. This signal may be required by a debug station.
6	3v3	O	3.3V power supply bus.

Table 7-3. P3 - COP/JTAG Connector

Pin No.	Signal Name	Attribute	Description
7	TCK	I	Test port Clock. This clock shifts in / out data to / from the JTAG logic. Data is driven on the falling edge of TCK and is sampled both internally and externally on it's rising edge. TCK is pulled up internally by the MPC8266.
8	N.C.	-	Not Connected.
9	TMS	I	Test Mode Select. This signal qualified with TCK in a same manner as TDI, changes the state of the JTAG machine. This line is pulled up internally by the MPC8266.
10	GND	O	Digital GND. Main GND plane.
11	SRESET#	I/O, O.D.	Soft Reset (L). This is the MPC8266's soft reset which is in fact a non-maskable interrupt, making the PPC take the reset exception from the reset vector. This line may be driven by the MPC8266 as well during soft-reset sequence, for 512 system clocks. This line is pulled up on the ADS with a 1KΩ resistor. When driven externally, it **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the MPC8266 and / or to ADS logic.**
12	GND	O	Digital GND. Main GND plane.
13	HRESET#	I/O, O.D.	MPC8266's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the MPC8266. During that sequence, asserted by the MPC8266 for 512 system clocks. Pulled Up on the ADS using a 1KΩ resistor. When driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the MPC8266 and / or to ADS logic.**
14	N.C.	-	Not Connected.
15	XBR3# (CKSTOP_OUT#)	I/O	Normally configured as XBR3# which has no function with this connector. May be configured as CKSTP_OUT# - Check Stop Out (L). When asserted (Low) indicates that the MPC8266 core has entered a Check-Stop state.
16	GND	O	Digital GND. Main GND plane.

7.1.4 P4 - CPM Expansion Connector

P4 is a 128 pin, 90^0, DIN 41612 connector, which allows for convenient expansion of the MPC8266's serial ports. This connector contains all CPM pins plus power supply pins, to provide for easy tool connection as described in Table 7-4.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
A1	RS_RXD1 (PD31[a])	I/O, T.S.	When RS232 port #1 is enabled, this signal is the receive data line for that port. When this port is disabled, this signal is tristated and may be used to any available alternate function for PD31.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
A2	RS_TXD1 (PD30)	I/O, T.S.	When RS232 port #1 is enabled, this signal is the transmit data line for that port. When this port is disabled, this signal may be used to any available alternate function for PD30.
A3	PD29	I/O, T.S	MPC8266's Port D 29 line. Parallel I/O or CPM dedicated line.May be used for any of it's available functions.
A4	RS_RXD2 (PD28)	I/O, T.S.	When RS232 port #2 is enabled, this signal is the receive data line for that port. When this port is disabled, this signal is tristated and may be used to any available alternate function for PD28.
A5	RS_TXD2 (PD27)	I/O, T.S.	When RS232 port #2 is enabled, this signal is the transmit data line for that port. When this port is disabled, this signal may be used to any available alternate function for PD27.
A6	PD26	I/O, T.S.	MPC8266's PD(26:18) Port D lines. Parallel I/O or CPM dedicated lines. May be used for any of their available functions.
A7	PD25		
A8	PD24		
A9	PD23		
A10	PD22		
A11	PD21		
A12	PD20		
A13	PD19		
A14	PD18		
A15	ATMRXPTY (PD17)	I/O, T.S.	ATM Receive Parity Line. When the ATM port is enabled, this line is connected to the receive parity of the PM5350 ATM UNI. When this port is disabled, this signal is tristated and may be used for any available function of PD17.
A16	ATMTXPTY (PD16)	I/O, T.S.	ATM Transmit Parity Line. When the ATM port is enabled, this line is connected to the transmit parity of the PM5350 ATM UNI. When this port is disabled, this signal may be used for any available function of PD16.
A17	I2CSDA (PD15)	I/O, T.S.	This signal is connected to the serial I^2C data line. This line may be used off-board as an I^2C data line for external I^2C device.
A18	I2CSCL (PD14)	I/O, T.S.	This signal is connected to the serial I^2C clock line. This line may be used off-board as an I^2C clock line for external I^2C device.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
A19	PD13	I/O, T.S.	MPC8266's PD(13:4) Port D lines. Parallel I/O or CPM dedicated lines. May be used for any of their available functions.
A20	PD12		
A21	PD11		
A22	PD10		
A23	PD9		
A24	PD8		
A25	PD7		
A26	PD6		
A27	PD5		
A28	PD4		
A29	ATMRCLKDIS	I	ATM Receive Clock Out Disable. When active (H), the ATMRCLK output, on pin C29 of this connector, is Tri-stated. When either not connected or driven low, ATMRCLK on pin C29, is enabled. This provides compatibility with ENG revision of T/ECOM communication tools.
A30	EXPVCC	O	5V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool.
A31			
A32			
B1	ATMTXEN# (PA31)	I/O, T.S.	ATM Transmit Enabled (L). When this signal is asserted (Low), while the ATM port is enabled and ATMTFCLK is rising, an octet of data, ATMTXD(7:0), is written into the transmit FIFO of the PM5350. When the ATM port is disabled, this line may be used for any available function of PA31.
B2	ATMTCA (PA30)	I/O, T.S.	ATM Transmit Cell Available (H). When this signal is asserted (High), while the ATM port is enabled, it indicates that the transmit FIFO of the PM5350 is empty and ready to except a new cell. When negated, it may show either that the transmit FIFO is Full or close to Full, depending on PM5350 internal programming. When the ATM port is disabled, this line may be used for any available function of PA30.
B3	ATMTSOC (PA29)	I/O, T.S.	ATM Transmit Start Of Cell (H). When this signal is asserted (High) by the MPC8266, while the ATM port is enabled, it indicates to the PM5350 the start of a new ATM cell over ATMTXD(7:0), i.e., the 1'st octet is present there. When the ATM port is disabled, this line may be used for any available function of PA29.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
B4	ATMRXEN# (PA28)	I/O, T.S.	ATM Receive Enable (L). When this signal is asserted (Low), while the ATM port is enabled and ATMRFCLK[b] goes high, on octet of data is available at the PM5350's ATMRXD(7:0) lines. When negated while ATMRFCLK goes high data on ATMRXD(7:0) is invalid, however driven. When the ATM port is disabled, this line may be used for any available function for PA28.
B5	ATMRSOC (PA27)	I/O, T.S.	ATM Receive Start Of Cell (H). When this signal is asserted (High), while the ATM port is enabled, it indicates, that the 1'st octet of data for the received cell is available at the PM5350's ATMRXD(7:0) lines. This line is updated over the rising edge of ATMRFCLK. When the ATM port is disabled, this line is tristated and may be used for any available function for PA27.
B6	ATMRCA (PA26)	I/O, T.S.	ATM Receive Cell Available (H). When this signal is asserted (High), while the ATM port is enabled and ATMRFCLK goes high, it indicates that the PM5350's receive FIFO is either full or that there are 4 empty bytes left in it - PM5350 internal programming dependent. When the ATM port is disabled, this line is tristated and may be used for any available function of PA26.
B7	ATMTXD0 (PA25)	I/O, T.S.	ATM Transmit Data (7^{c}:0). When the ATM port is enabled, this bus carries the ATM cell octets, written to the PM5350's transmit FIFO. This bus is considered valid only when ATMTXEN# is asserted and are sampled on the rising edge of ATMTFCLK. When the ATM port is disabled, these lines may be used for any available respective function.
B8	ATMTXD1 (PA24)		
B9	ATMTXD2 (PA23)		
B10	ATMTXD3 (PA22)		
B11	ATMTXD4 (PA21)		
B12	ATMTXD5 (PA20)		
B13	ATMTXD6 (PA19)		
B14	ATMTXD7 (PA18)		
B15	ATMRXD7 (PA17)	I/O, T.S.	ATM Receive Data (7^{c}:0). When the ATM port is enabled, this bus carries the cell octets, read from the PM5350 receive FIFO. This lines are updated on the rising edge of ATMRFCLK[b]. When the ATM port is disabled, these lines are tristated and may be used for any available respective function.
B16	ATMRXD6 (PA16)		
B17	ATMRXD5 (PA15)		
B18	ATMRXD4 (PA14)		
B19	ATMRXD3 (PA13)		
B20	ATMRXD2 (PA12)		
B21	ATMRXD1 (PA11)		
B22	ATMRXD0 (PA10)		

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
B23	PA9	I/O, T.S.	MPC8266's Port A (9:0). Parallel I/O or dedicated CPM lines. May be used for any of their available functions.
B24	PA8		
B25	PA7		
B26	PA6		
B27	PA5		
B28	PA4		
B29	PA3		
B30	PA2		
B31	PA1		
B32	PA0		
C1	FETHTXER (PB31)	I/O, T.S.	Fast-Ethernet[d] Transmit Error (H). When the Ethernet port is enabled, this signal will be asserted (High) by the MPC8266 when an error is discovered in the transmit data stream. When the port is operation at 100 Mbps, the LXT970 responds by sending invalid code symbols on the line. When the Ethernet port is disabled, this line may be used for any available function of PB31.
C2	FETHRXDV (PB30)	I/O, T.S.	Fast-Ethernet Receive Data Valid (H). When this signal is asserted (High) while the Fast Ethernet port is enabled and FETHRXCK goes high, it indicates that data is valid on the MII Receive Data lines - FETHRXD(3:0). When the Fast Ethernet port is disabled, this line is tristated and may be used for any available function go PB30.
C3	FETHTXEN (PB29)	I/O, T.S.	Fast-Ethernet Transmit Enable (H). The MPC8266 will assert (High) this line, to indicate data valid on the FETHTXD(3:0) lines. When the Fast-Ethernet port is disabled, this line may be used for any available function of PB29.
C4	FETHRXER (PB28)	I/O, T.S.	Fast-Ethernet Receive Error (H). When this signal is asserted (High) by the LXT970, while the Ethernet port is enabled and FETHRXCK goes high, it indicates that the port is receiving invalid data symbols from the network. When the Ethernet port is disabled, this line is tristated and may be used for any available function of PB28.
C5	FETHCOL (PB27)	I/O, T.S.	Fast-Ethernet Port Collision Detected (H). When this signal is asserted (High) by the LXT970, while the ethernet port is enabled, it indicates a Collision state over the line. When the LXT970 is in Full-Duplex mode, this line is inactive. When the Ethernet port is disabled, this line is tristated and may be used for any available function of the PB27.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
C6	FETHCRS (PB26)	I/O, T.S.	Fast-Ethernet Carrier Sense (H). When this signal is asserted (High), while the Ethernet port is enabled and the LXT970 is in half-duplex mode, it indicates that either the transmit or receive media are non-idle. When the LXT970 is in either full-duplex or repeater operation, it indicates that the receive medium is non-idle. When the Ethernet port is disabled, this line may be used for any available function of PB26.
C7	FETHTXD3 (PB25)	I/O, T.S.	Fast Ethernet Transmit Data (3:0). This is the MII transmit data bus. The MPC8266 drives these lines according to rising edge of FETHTXCK. When the ethernet port is disabled, these lines may be used for any available respective function.
C8	FETHTXD2 (PB24)		
C9	FETHTXD1 (PB23)		
C10	FETHTXD0 (PB22)		
C11	FETHRXD0 (PB21)	I/O, T.S.	Fast Ethernet Receive Data (3:0). This is the MII receive data bus. The LXT970 drives these lines according to rising edge of FETHRXCK. When the ethernet port is disabled, these lines are tristated and may be used for any available respective parenthesized function.
C12	FETHRXD1 (PB20)		
C13	FETHRXD2 (PB19)		
C14	FETHRXD3 (PB18)		
C15	PB17	I/O, T.S.	MPC8266's Port B (17:4) Parallel I/O lines. May be used to any of their available functions.
C16	PB16		
C17	PB15		
C18	PB14		
C19	PB13		
C20	PB12		
C21	PB11		
C22	PB10		
C23	PB9		
C24	PB8		
C25	PB7		
C26	PB6		
C27	PB5		
C28	PB4		
C29	ATMRCLK	O, T.S.	ATM Receive Clock. A divide by 8 of the ATM line clock recovered by the ATM receive logic. Provided to assist Circuit Emulation Tool. Enabled only when pin A29 of this connector is either not connected or driven low. Otherwise, Tri-stated.

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
C30	GND	O	Digital Ground. Connected to main GND plane of the ADS.
C31			
C32			
D1	PC31	I/O, T.S.	MPC8266's Port C (31:22) Parallel I/O lines. May be used to any of their available functions.
D2	PC30		
D3	PC29		
D4	PC28		
D5	PC27		
D6	PC26		
D7	PC25		
D8	PC24		
D9	PC23		
D10	PC22		
D11	ATMTFCLK (PC21)	I/O, T.S.	ATM Transmit FIFO Clock. Upon the rising edge of this clock (driven by the MPC8266), while the ATM port is enabled, the cell octets are written to the PM5350's transmit FIFO. This clock samples ATMTXD(7:0), ATMTXPTY, ATMTXEN# and ATMTSOC. When the ATM port is disabled, this line may be used for any available function of PC21.
D12	PC20	I/O, T.S.	MPC8266's Parallel I/O Port-C 20. Parallel I/O line. May be used for any of its available functions
D13	FETHRXCK (PC19)	I/O, T.S.	Fast-Ethernet Receive Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is extracted from the received data and driven to the MPC8266 to qualify incoming receive data. When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC19
D14	FETHTXCK (PC18)	I/O, T.S.	Fast-Ethernet Transmit Clock. When the Ethernet port is enabled, this clock (25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps) is normally extracted from the received data and driven to the MPC8266 to qualify out coming transmit data. In Slave mode (not used with this application) this clock should be input to the LXT970. When the Ethernet port is disabled, this line is tristated and may be used for any available function of PC18
D15	PC17	I/O, T.S.	MPC8266's Port C (17:15) Parallel I/O lines. May be used to any of their available functions.
D16	PC16		
D17	PC15		

Table 7-4. P4 - CPM Expansion Connector

Pin No.	Signal Name	Attribute	Description
D18	RS_CD1# (PC14)	I/O, T.S.	RS232 Port 1 Carrier Detect (L). Connected via RS232 transceiver to RS232 DTR1# input, allowing detection of a connected terminal to this port. This line is simply a PI/O input line to the MPC8266. When RS232 Port 1 is disabled, this line is tristated and may be used for any available function of PC14.
D19	PC13	I/O, T.S.	MPC8266's Port C 13 Parallel I/O line. May be used to any of its available functions.
D20	RS_CD2# (PC12)	I/O, T.S.	RS232 Port 2 Carrier Detect (L). Connected via RS232 transceiver to RS232 DTR2# input, allowing detection of a connected terminal to this port. This line is simply a PI/O input line to the MPC8266. When RS232 Port 2 is disabled, this line is tristated and may be used for any available function of PC12.
D21	PC11	I/O, T.S.	MPC8266's Port C 11 Parallel I/O line. May be used to any of its available functions.
D22	FETHMDC (PC10)	I/O, T.S.	Fast-Ethernet Port Management Data Clock. This slow clock (S/W generated) qualifies the management data I/O to read / write the LXT970's internal registers. When the Ethernet port is disabled, this line may be used for any available function of PC10.
D23	FETHMDIO (PC9)	I/O, T.S.	Fast-Ethernet Port Management Data I/O. This signal serves as bidirectional serial data line, qualified by FETHMDC, to allow read / write the LXT970's internal registers. When the Ethernet port is disabled, this line may be used for any available function of PC9.
D24	PC8	I/O, T.S.	MPC8266's Port C (8:0) Parallel I/O lines. May be used to any of their available functions.
D25	PC7		
D26	PC6		
D27	PC5		
D28	PC4		
D29	PC3		
D30	PC2		
D31	PC1		
D32	PC0		

a. The functions in parenthesis, are MPC8266's parallel I/Os.

b. Normally connected to ATMTFCLK on the ADS.

c. MS bit.

d. For that matter, both 100-Base-T and 10-Base-T.

7.1.5 P5, P6, P10, P11, P13, P14, P16, P18, P19 - Logic Analyzer MICTOR Connectors

These are 38 pin, SMT, high density, matched impedance connector made by AMP. They contain

the MPC8266 60X bus, 60X system and memory controller signals, unbuffered. The pinout of these connectors is shown in the schematics. For signal description of these connectors, see the MPC8266 User's Manual.

7.1.6 P7, P8, P9 - PCI Connectors

These are 2 X 62 , 3.3V keyed, 32 bit PCI connectors. The pinout of each connector is available in Table 7-5.

For signal descriptions for these connectors, see the PCI v2.2 Standard.

Table 7-5. P7, P8, P9 - PCI Connectors

Pin Number	Side B	Comments	Side A	Comments
1	-12V	Not Connected	TRST#	
2	TCK		+12V	
3	Ground		TMS	
4	TDO		TDI	
5	+5V		+5V	
6	+5V		INTA#	
7	INTB#	Not Connected	INTC#	Not Connected
8	INTD#	Not Connected	+5V	
9	PRSNT1#	Connected to GND	Reserved	Not Connected
10	Reserved	Not Connected	+3.3V(I/O)	
11	PRSNT2#	Connected to GND	Reserved	Not Connected
12	CONNECTOR KEY	3.3 volt key	CONNECTOR KEY	3.3 volt key
13	CONNECTOR KEY	3.3 volt key	CONNECTOR KEY	3.3 volt key
14	Reserved	Not Connected	3.3Vaux	Not Connected
15	Ground		RST#	
16	CLK		+3.3V (I/O)	
17	Ground		GNT#	
18	REQ#		Ground	
19	+3.3V (I/O)		PME#	Not Connected
20	AD[31]		AD[30]	
21	AD[29]		+3.3V	

Table 7-5. P7, P8, P9 - PCI Connectors

Pin Number	Side B	Comments	Side A	Comments
22	Ground		AD[28]	
23	AD[27]		AD[26]	
24	AD[25]		Ground	
25	+3.3V		AD[24]	
26	C/BE[3]#		IDSEL	
27	AD[23]		+3.3V	
28	Ground		AD[22]	
29	AD[21]		AD[20]	
30	AD[19]		Ground	
31	+3.3V		AD[18]	
32	AD[17]		AD[16]	
33	C/BE[2]#		+3.3V	
34	Ground		FRAME#	
35	IRDY#		Ground	
36	+3.3V		TRDY#	
37	DEVSEL#		Ground	
38	Ground		STOP#	
39	LOCK#	Not Connected	+3.3V	
40	PERR#		SDONE	Not Connected
41	+3.3V		SBO#	Not Connected
42	SERR#		Ground	
43	+3.3V		PAR	
44	C/BE[1]#		AD[15]	
45	AD[14]		+3.3V	
46	Ground		AD[13]	
47	AD[12]		AD[11]	
48	AD[10]		Ground	
49	M66EN	Coupled to GND, using a 0.01uF capacitor	AD[09]	
50	Ground		Ground	
51	Ground		Ground	

Table 7-5. P7, P8, P9 - PCI Connectors

Pin Number	Side B	Comments	Side A	Comments
52	AD[08]		C/BE[0]#	
53	AD[07]		+3.3V	
54	+3.3V		AD[06]	
55	AD[05]		AD[04]	
56	AD[03]		Ground	
57	Ground		AD[02]	
58	AD[01]		AD[00]	
59	+3.3V (I/O)		+3.3V (I/O)	
60	ACK64#	Not Connected	REQ64#	Not Connected
61	+5V		+5V	
62	+5V		+5V	

7.1.7 P12 - ATX Power Supply Connector

This is a standard ATX Form Factor Power Connector as described in Table 7-6.

Table 7-6. P12 - ATX Power Supply Connector

Pin	Signal	Pin	Signal
1	+3.3VDC	11	+3.3VDC-Sense
2	+3.3VDC	12	-12VDC
3	Groung	13	Groung
4	+5VDC	14	$\overline{\text{Power\_On}}$
5	Groung	15	Groung
6	+5VDC	16	Groung
7	Groung	17	Groung
8	Power_OK	18	-5VDC
9	+5VStand_By	19	+5VDC
10	+12VDC	20	+5VDC

7.1.8 P15 - Mach/Lattice ISP Connector

This is a 10 pin generic 0.100" pitch header connector, providing In System Programming (ISP)

capability for Latice made programmable logic on board. The pinout of P15 is shown in Table 7-7.

Table 7-7. P15 - Lattice ISP Connector

Pin No.	Signal Name	Attribute	Description
1	ISPTCK	I	ISP Test port Clock. This clock shifts in / out data to / from the programmable logic JTAG chain.
2	N.C.	-	Not Connected.
3	ISPTMS	I	ISP Test Mode Select. This signal qualified with ISPTCK, changes the state of the prog. logic JTAG machine.
4	GND	O	Digital GND. Main GND plane.
5	ISPTDI	I	ISP Transmit Data In. This is the prog. logic's JTAG serial data input, sampled on the rising edge of TCK.
6	VCC	O	5V power supply bus.
7	ISPTDO	O	ISP Transmit Data Output. This the prog. logic's JTAG serial data output driven by Falling edge of TCK.
8	GND	O	Digital GND. Main GND plane.
9	N.C.	-	Not Connected.
10	N.C.	-	Not Connected.

7.1.9 P17 - System Expansion Connector

P17 is a 128 pin, 90^0, DIN 41612 connector, which provides a minimal system I/F required to interface various types of communication transceivers. This connector contains 16 bit (lower PPC bus) address lines, 16 bit (higher PPC bus) Data lines plus useful GPCM and UPM control lines. The pinout of P17 is shown in Table 7-8.

Table 7-8. P17 - System Expansion Connector

Pin No.	Signal Name	Attribute	Description
A1	EXPA16	O	Expansion Address (16[a]:31). This is a Latched-Buffered version of the MPC8266's PPC Address lines (16:31), provided for external tool connection. To avoid reflection these lines are series terminated with 43 Ω resistors.
A2	EXPA17		
A3	EXPA18		
A4	EXPA19		
A5	EXPA20		
A6	EXPA21		
A7	EXPA22		
A8	EXPA23		
A9	EXPA24		
A10	EXPA25		
A11	EXPA26		
A12	EXPA27		
A13	EXPA28		
A14	EXPA29		
A15	EXPA30		
A16	EXPA31		
A17	EXP12V	O	These can be connected to the positive 12V source from the PCI edge connector thru J3. This line is fused by a 0.5A ressetable poly-switch.
A18			
A19	N.C.	-	Not Connected.
A20	EXP3.3V	O	3.3V Power Out. These lines are connected to the main 3.3V plane of the PQ2PCIAI-ADS, this, to provide 3.3V power where necessary for external tool connected.
A21			
A22			
A23			
A24			
A25	N.C.	-	Not Connected.

Table 7-8. P17 - System Expansion Connector

Pin No.	Signal Name	Attribute	Description
A26	EXPVCC	O	5V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool.
A27			
A28			
A29			
A30			
A31			
A32			
B1	GND	O	Digital Ground. Connected to main GND plane of the ADS.
B2			
B3			
B4	TSTAT0	I	Tool Status (0^a:7). These lines may be driven by an external tool to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 4-12. "BCSR2 Description" on page 71.
B5	TSTAT1		
B6	TSTAT2		
B7	TSTAT3		
B8	TSTAT4		
B9	TSTAT5		
B10	TSTAT6		
B11	TSTAT7		
B12	TOOLREV0	I	Tool Revision (0^a:3). These lines should be driven by an external tool with the Tool Revision Code, to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 4-12. "BCSR2 Description" on page 71.
B13	TOOLREV1		
B14	TOOLREV2		
B15	TOOLREV3		
B16	EXTOLI0	I	External Tool Identification (0^a:3). These lines should be driven by an external tool with the Tool Identification Code, to be read via BCSR2 of the ADS. These lines are pulled-up on the ADS, by 10 KΩ resistors. See also Table 4-12. "BCSR2 Description" on page 71
B17	EXTOLI1		
B18	EXTOLI2		
B19	EXTOLI3		
B20	N.C.	-	Not Connected
B21	EXP3.3V	O	3.3V Power Out. These lines are connected to the main 3.3V plane of the PQ2PCIAI-ADS, this, to provide 3.3V power where necessary for external tool connected.
B22			
B23			
B24			

Table 7-8. P17 - System Expansion Connector

Pin No.	Signal Name	Attribute	Description
B25	N.C.	-	Not Connected
B26	EXPVCC	O	5V Supply. Connected to ADS's 5V VCC plane. Provided as power supply for external tool.
B27			
B28			
B29			
B30			
B31			
B32			
C1	GND	O	Digital Ground. Connected to main GND plane of the ADS.
C2	CLK8	O	Buffered System Clock..
C3	GND	O	Digital Ground. Connected to main GND plane of the ADS.
C4	BTOOLCS1#	O	Buffered Tool Chip Select 1 (L). This is a buffered MPC8266's CS6# line, reserved for an external tool.
C5	BTOOLCS2#	O	Buffered Tool Chip Select 2 (L). This is a buffered MPC8266's CS7# line, reserved for an external tool.
C6	GND	O	Digital Ground. Connected to main GND plane of the ADS.
C7	ATMEN#	O	ATM Port Enable (L). This line enables the ATM port UNI's output lines towards the MPC8266. An external tool, using the same pins as does the ATM port should consult this signal before driving the same lines. **Failure to do so might result in permanent damage to the PM5350 ATM UNI.**
C8	ATMRST#	O	ATM Port Reset (L). This signal resets the ATM UNI (PM5350). An external tool may use this signal to its benefit.
C9	FETHRST#	O	Ethernet Port Reset (L). This signal resets the LXT970 Ethernet transceiver. An external tool may use this signal to its benefit.
C10	HRESET#	I/O, O.D.	MPC8266's Hard Reset (L). When asserted by an external H/W, generates Hard-Reset sequence for the MPC8266. During that sequence, asserted by the MPC8266 for 512 system clocks. Pulled Up on the ADS using a 1KΩ resistor. When driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so might result in permanent damage to the MPC8266 and / or to ADS logic.**
C11	IRQ6#	I	Interrupt Request 6 (L). Connected to MPC8266's DP6/CSE0/IRQ6# signal. Pulled up on the ADS with a 10 KΩ resistor. This line is shared with the ATM UNI's interrupt line and therefore, when driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so may result in permanent damage to the MPC8266 or to ADS logic.**

Table 7-8. P17 - System Expansion Connector

Pin No.	Signal Name	Attribute	Description
C12	IRQ7#	I	Interrupt Request 7 (L). Connected to MPC8266's DP7/CSE1/IRQ7# signal. Pulled up on the ADS with a 10 KΩ resistor. This line is shared with the Fast Ethernet transceiver's interrupt line and therefore, when driven by an external tool, **MUST be driven with an Open Drain gate. Failure to do so might result in permanent damage to the MPC8266 and / or to ADS logic.**
C13	GND	O	Digital Ground. Connected to main GND plane of the ADS.
C14	EXPD0	I/O, T.S.	Expansion Data (0^a:15). This is a double buffered version of the PPC bus D(0:15) lines, controlled by on-board logic. These lines will be driven only if BTOOLCS1# or BTOOLCS2# are asserted. Otherwise they are tristated. The direction of these lines is determined by buffered BCTL0, in function of W/R#.
C15	EXPD1		
C16	EXPD2		
C17	EXPD3		
C18	EXPD4		
C19	EXPD5		
C20	EXPD6		
C21	EXPD7		
C22	EXPD8		
C23	EXPD9		
C24	EXPD10		
C25	EXPD11		
C26	EXPD12		
C27	EXPD13		
C28	EXPD14		
C29	EXPD15		
C30	N.C.	-	Not Connected
C31			
C32			
D1	GND	O	Digital Ground. Connected to main GND plane of the ADS.
D2			
D3			
D4	EXPWE0#	O	Expansion Write Enable (0:1) (L). These are buffered GPCM Write Enable lines (0:1). They are meant to qualify writes to GPCM controlled 8/16 data bus width memory devices. This to provide eased access to various communication transceivers. EXPWE0# controls EXPD(0:7) while EXPWE1# controls EXPD(8:15). These lines may also function as UPM controlled Byte Select Lines, which allow control over almost any type of memory device.
D5	EXPWE1#		

Table 7-8. P17 - System Expansion Connector

Pin No.	Signal Name	Attribute	Description
D6	GND	O	Digital Ground. Connected to main GND plane of the ADS.
D7	EXPGL0#	O	Expansion General Purpose Lines (0:5) (L). These are buffered GPL(0:5)# lines which assist UPM control over memory device if necessary. These are output only signals and therefore, do not support H/W controlled UPM waits.
D8	EXPGL1#		
D9	EXPGL2#		
D10	EXPGL3#		
D11	EXPGL4#		
D12	EXPGL5#		
D13	GND	O	Digital Ground. Connected to main GND plane of the ADS.
D14	EXPALE	O	Expansion Address Latch Enable (H). This is the buffered MPC8266‘s ALE, provided for expansion board’s use.
D15	EXPCTL0	O	Expansion Control Line 0. This line is a buffered version of MPC8266’s BCTL0 (Bus Control Line 0) which serves as W/R#, provided for expansion board’s use.
D16	GND	O	Digital Ground. Connected to main GND plane of the ADS.
D17			
D18			
D19			
D20			
D21			
D22			
D23			
D24			
D25			
D26			
D27			
D28			
D29			
D30			
D31			
D32			

a.MS Bit.

7.2 Programmable Logic Equations

There are 4 programmable logic devices on board.

1. U14 - BCSR
2. U10 - PCI Interrupt Controller
3. U22, U23 - Address Latch/Mux when L2Cache is used

7.2.1 U14 - BCSR Code

```
MODULE pq2pcimb

TITLE 'MPC8265 PCI motherboard ads control status register'

"*******************************************************************************
"* In this file (2 - Prototype-B) the following changes were made  (08/14/01):
"* - Added support for default Hard Reset Configuration Word from BCSR
"*   by using jumper JP3.
"* - Added L2cache support.
"*******************************************************************************

"*******************************************************************************
"* Device declaration.
"*******************************************************************************

"*******************************************************************************
"* Pins declaration.
"*******************************************************************************
"* System i/f pins
"*******************************************************************************
SYSCLK              PIN  11 ;

BrdContRegCs_B      PIN  36 ;
DVal_B              PIN  43 ;
R_B_W               PIN  86 ; " BCTL0 signal
BCTL1               PIN  64 ; " Alternate Buffers Enable source

A7                  PIN  98 ; " for flash support
A8                  PIN   4 ; " for flash support
A27                 PIN   5 ;
A28                 PIN   7 ;
A29                 PIN   8 ;
```

```
D0              PIN  84   istype 'com' ;
D1              PIN  78   istype 'com' ;
D2              PIN  80   istype 'com' ;
D3              PIN  59   istype 'com' ;
D4              PIN  70   istype 'com' ;
D5              PIN  57   istype 'com' ;
D6              PIN  60   istype 'com' ;
D7              PIN  72   istype 'com' ;

"*****************************************************************************
"* Board Control Pins. Read/Write.
"*****************************************************************************
PBI             PIN  44  istype 'reg,buffer' ; " Page Base Interleaving
DimmSize        PIN  53  istype 'reg,buffer' ; " Sdram Dimm Size
L2Inh_B         PIN  15  istype 'reg,buffer' ; " flash enable.
L2Flush_B       PIN  54  istype 'reg,buffer' ; " 60x bus sdram enable
L2Lock_B        PIN  35  istype 'reg,buffer' ; " bursting sram enable
L2Clear_B       PIN  34  istype 'reg,buffer' ; " local bus sdram enable
SignaLamp0_B    PIN  79  istype 'reg,buffer' ; " status lamp 0 for misc s/w visual
SignaLamp1_B    PIN  81  istype 'reg,buffer' ; " status lamp 1 for misc s/w visual

AtmEn_B         PIN  94  istype 'reg,buffer' ; " atm uni enable
AtmRst_B NODE   istype 'reg,buffer' ; " atm uni reset bit
AtmRstOut_B     PIN  96  istype 'com' ; " atm uni reset driven by register or by HRESET_B
FEthEn_B        PIN  95  istype 'reg,buffer' ; " fast ethernet trans. enable
FEthRst_B   NODE    istype 'reg,buffer' ; " fast ethernet trans. reset bit
FEthRstOut_B    PIN   3  istype 'com' ; "fast eth trans reset driven by register
"or by HRESET_B
RS232En1_B      PIN  92  istype 'reg,buffer' ; " RS232 port 1 enable
RS232En2_B      PIN  93  istype 'reg,buffer' ; " RS232 port 2 enable

"*****************************************************************************
"* Board Status Registers Chip-Selects
"*****************************************************************************
Bcsr2Cs_B       PIN  42  istype 'com' ;
Bcsr4Cs_B       PIN  32  istype 'com' ;

"*****************************************************************************
"* Flash/EEPROM Associated Pins.
```

```
"*****************************************************************************
F_PD1              PIN  71 ;
F_PD2              PIN  69 ;
F_PD3              PIN  67 ;
F_PD4              PIN  65 ;

Cs0_B              PIN  31 ; " flash/eeprom chip-select input
Cs4_B              PIN  33 ; " eeprom/flash chip-select input

EEpromCs_B         PIN  83  istype 'com' ; " EEPROM chip-select

FlashCs1_B         PIN  58  istype 'com' ; " Flash bank1 chip-select
FlashCs2_B         PIN  48  istype 'com' ; " Flash bank2 chip-select
FlashCs3_B         PIN  45  istype 'com' ; " Flash bank3 chip-select
FlashCs4_B         PIN  55  istype 'com' ; " Flash bank4 chip-select

"*****************************************************************************"
* PM5350 ATM UNI Associated Pins.
"*****************************************************************************
AtmUniCsIn_B       PIN  22 ;
AtmUniCsOut_B      PIN  21  istype 'com' ; " remove if short of pins

"*****************************************************************************
"* Reset & Interrupt Logic Pins.
"*****************************************************************************

PORIn_B            PIN  97 ;

RstConf_B          PIN  28  istype 'com'; " Hard Reset master select.

Rst0               PIN  20 ; " connected to N.C. of Reset P.B.
Rst1               PIN  19 ; " connected to N.O. of Reset P.B.

HardReset_B        PIN  16  istype 'com' ; " Actual hard reset output (O.D.)
SoftReset_B        PIN  29  istype 'com' ; " Actual soft reset output (O.D.)

Abr0               PIN  18 ; " connected to N.C. of Abort P.B.
Abr1               PIN  17 ; " connected to N.O. of Abort P.B.

NMIEn      NODE    istype 'com' ; " enables T.S. NMI pin
```

```
NMI_B               PIN  41  istype 'com' ; " Actual NMI pin (O,O.D.)

"*****************************************************************************
"* Data Buffers Enables and Reset configuration support
"*****************************************************************************
TEA_B               PIN  61 ; "  Transfer Error Acknowledge.
PCIIntContCs_B      PIN  85 ; " PCI Interrupt Controller Chip Select input
DataBufEn_B         PIN   9  istype 'com,invert' ; " data buffer enable

ToolCs1_B           PIN  30 ; " comm tool cs line 1.
ToolCs2_B           PIN  91 ; " comm tool cs line 2.

ToolDataBufEn_B     PIN   6  istype 'com,invert' ; " tool data buffer enable

"*****************************************************************************
"* Hard Reset Configuration Logic
"*****************************************************************************
boot_device_B PIN  10 ; " selects EEPROM/FLASH_B as boot device
bcsrConfEn PIN  66 ; " selects Hard Reset Configuration Source
  " as BCSR or EEPROM/FLASH.
"*****************************************************************************
"* JTAG Logic
"*****************************************************************************
JtagEn              NODE istype 'reg,buffer';
Tdi                 PIN  56 ;
Tdo                 NODE istype 'reg,buffer' ; " clocked by falling Tck
TdoOut              PIN  82   istype 'com' ;
Tck                 PIN  14 ;
Tms                 PIN  47 ;
Trst_B              PIN  46 ;

PonResetOut         PIN  68  istype 'com';   " active high
"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************
"*****************************************************************************
"* System Hard Reset Configuration.
"*****************************************************************************
DataOeNODE istype 'com' ;" data bus output enable on read.
```

```
"*******************************************************************************
"* Control Register Enable Protection.
"*******************************************************************************

"*******************************************************************************
"* Reset & Interrupt Logic Pins.
"*******************************************************************************
RstDeb1NODE istype 'keep,com' ;   " reset push button debouncer
AbrDeb1NODE istype 'keep,com' ;   " abort push button debouncer

HardResetEnNODE istype 'com' ;   " enables T.S. hard reset pin
SoftResetEnNODE istype 'com' ;   " enables T.S. soft reset pin

"*******************************************************************************
"* data buffers enable.
"*******************************************************************************
SyncHardReset_B  NODE istype 'reg,buffer' ;" synchronized hard reset
DSyncHardReset_B  NODE istype 'reg,buffer' ;" double synchronized hard reset
HoldOffCnt2,
HoldOffCnt1,
HoldOffCnt0    NODE istype 'reg,buffer' ;      " data buf en hold-off counter
HoldOffTc      NODE istype 'com' ;             " terminal count for that counter

"*******************************************************************************
"* Power On Reset
"*******************************************************************************
S_PORIn_B     NODE  istype 'reg,buffer' ;   " synced pon reset.

"*******************************************************************************
"* JTAG Logic
"*******************************************************************************
JtagShiftDR0      NODE istype 'reg,buffer' ;    " Jtag shift data register
JtagShiftDR1      NODE istype 'reg,buffer' ;    "
JtagShiftDR2      NODE istype 'reg,buffer' ;    "
JtagShiftDR3      NODE istype 'reg,buffer' ;    "
JtagShiftDR4      NODE istype 'reg,buffer' ;    "
JtagShiftDR5      NODE istype 'reg,buffer' ;    "
JtagShiftDR6      NODE istype 'reg,buffer' ;    "
JtagShiftDR7      NODE istype 'reg,buffer' ;    "
```

```
JtagState0     NODE istype 'reg,buffer' ;    " Jtag state machine
JtagState1     NODE istype 'reg,buffer' ;    "
JtagState2     NODE istype 'reg,buffer' ;    "
JtagState3     NODE istype 'reg,buffer' ;    "

JtagResetState     NODE istype 'reg,buffer' ;    " sampling state on falling TCK
JtagShiftIrState   NODE istype 'reg,buffer' ;    " sampling state on falling TCK
JtagShiftDrState   NODE istype 'reg,buffer' ;    " sampling state on falling TCK
JtagTdoEnable      NODE istype 'reg,buffer' ;    " sampling state on falling TCK

JtagShiftIR0     NODE istype 'reg,buffer' ;    " Jtag Inst. Shift register
JtagShiftIR1     NODE istype 'reg,buffer' ;    "
JtagShiftIR2     NODE istype 'reg,buffer' ;    "

JtagIR0        NODE istype 'reg,buffer' ;    " Jtag Inst. register
JtagIR1        NODE istype 'reg,buffer' ;    "
JtagIR2        NODE istype 'reg,buffer' ;    "

JtagReceiveFull    NODE istype 'reg,buffer' ;   " indicates receive shift reg.
                                  " ready for read by memory cont

JtagReceiveFullReset NODE istype 'com' ;         " resets the receive full flag
SReadJtagDownloadData NODE istype 'reg,buffer';

JtagStateReset     NODE istype 'com' ;  " jtag state machine only
JtagReset         NODE istype 'com' ;  " global reset

TdoEnable         NODE istype 'com';   " enables the muxed TdoOut.

"*****************************************************************************
"* Misceleneous.
"*****************************************************************************
KeepPinsConnected  NODE istype 'com' ;

"*****************************************************************************
* Equations
*"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*****************************************************************************
```

```
"* SIMULATION = 1 ;
"*******************************************************************************
"* Signal groups
"*******************************************************************************
JTAG = 1;

Add = [A27..A29] ;

Data = [D0..D7] ;

ContReg = [PBI,
        DimmSize,
        L2Inh_B,
        L2Flush_B,
        L2Lock_B,
        L2Clear_B,
        SignaLamp0_B,
        SignaLamp1_B,
        AtmEn_B,
        AtmRst_B,
        FEthEn_B,
        FEthRst_B,
        RS232En1_B,
        RS232En2_B,
        JtagEn.fb] ;

ReadBcsr0 = [PBI,
         DimmSize,
         L2Inh_B,
         L2Flush_B,
         L2Lock_B,
         L2Clear_B,
         SignaLamp0_B,
         SignaLamp1_B] ;

ReadBcsr1 = [bcsrConfEn,
         boot_device_B,
         AtmEn_B,
         AtmRst_B.fb,
         FEthEn_B,
```

```
        FEthRst_B.fb,
        RS232En1_B,
        RS232En2_B] ;

ReadBcsr3 = [0,
        0,
        0,
        0,
        0,
        0,
        0,
        JtagEn.fb];

DrivenContReg = [PBI,
            DimmSize,
            L2Inh_B,
            L2Flush_B,
            L2Lock_B,
            L2Clear_B,
            SignaLamp0_B,
            SignaLamp1_B,
 AtmEn_B,
            FEthEn_B,
            RS232En1_B,
            RS232En2_B] ;

ClockedContReg = [PBI,
             DimmSize,
             L2Inh_B,
             L2Flush_B,
             L2Lock_B,
             L2Clear_B,
             SignaLamp0_B,
             SignaLamp1_B,
  AtmEn_B,
             AtmRst_B,
             FEthEn_B,
             FEthRst_B,
             RS232En1_B,
             RS232En2_B,
```

```
            JtagEn] ;

Bcsr0 = [PBI,          " for simulation
      DimmSize,
      L2Inh_B,
      L2Flush_B,
      L2Lock_B,
      L2Clear_B,
      SignaLamp0_B,
      SignaLamp1_B] ;

Bcsr1 = [bcsrConfEn," for simulation
 boot_device_B,
 AtmEn_B,
      AtmRst_B,
      FEthEn_B,
      FEthRst_B,
      RS232En1_B,
      RS232En2_B] ;

Bcsr6 = [JtagEn];    " for simulation

ToolCs =        [ToolCs1_B,ToolCs2_B] ;
FlashCsOut =    [FlashCs4_B,FlashCs3_B,FlashCs2_B,FlashCs1_B] ;
Reset =  [HardReset_B,SoftReset_B] ;
ResetEn =  [HardResetEn,SoftResetEn] ;
TransRst =  [AtmRstOut_B,FEthRstOut_B] ;
Rst =         [Rst1,Rst0] ;
Abr =         [Abr1,Abr0] ;
Debounce =  [RstDeb1,AbrDeb1] ;
SyncReset =     [SyncHardReset_B,DSyncHardReset_B] ;
RstCause =  [PORIn_B,Rst1,Rst0,Abr1,Abr0] ;
HoldOffCnt =    [HoldOffCnt2,HoldOffCnt1,HoldOffCnt0] ;
F_PD =         [F_PD4, F_PD3, F_PD2, F_PD1] ;
Cs =  [Cs0_B,Cs4_B,BrdContRegCs_B,PCIIntContCs_B,AtmUniCsIn_B,ToolCs1_B,ToolCs2_B] ;
BufEn =         [DataBufEn_B,ToolDataBufEn_B] ;
ConfAdd =       [A27,A28];

@ifndef L2CACHE {
CfgByte0 = [0,0,0,0,1,1,0,0];
```

```
CfgByte1 = [1,0,1,1,0,0,1,0];
CfgByte2 = [0,0,0,0,0,1,1,0];
CfgByte3 = [0,0,0,0,0,0,0,0];
}

@ifdef L2CACHE {
CfgByte0 = [0,0,0,1,1,1,0,0];
CfgByte1 = [1,0,1,1,0,0,1,0];
CfgByte2 = [0,0,0,0,0,1,1,0];
CfgByte3 = [0,0,0,0,0,0,0,0];
}

“*****************************************************************************
“* Power On Reset definitions
“*****************************************************************************
PON_RESET_ACTIVE = 0 ;

PON_RESET = (S_PORIn_B.fb == PON_RESET_ACTIVE) ;

“*****************************************************************************
“* Register Access definitions
“*****************************************************************************
BCSR0_ADD = 0 ;
BCSR1_ADD = 1 ;
BCSR2_ADD = 2 ;
BCSR3_ADD = 3 ;
BCSR4_ADD = 4 ;

VGR_WRITE_BCSR_0 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & !A28 & !A29) ;
VGR_WRITE_BCSR_1 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & !A28 & A29) ;
VGR_WRITE_BCSR_2 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & A28 & !A29) ;
VGR_WRITE_BCSR_3 = (!BrdContRegCs_B & !DVal_B & R_B_W & !A27 & A28 & A29) ;
VGR_WRITE_BCSR_4 = (!BrdContRegCs_B & !DVal_B & R_B_W & A27 & !A28 & !A29) ;
VGR_WRITE_BCSR_5 = (!BrdContRegCs_B & !DVal_B & R_B_W & A27 & !A28 & A29) ;
VGR_WRITE_BCSR_6 = (!BrdContRegCs_B & !DVal_B & R_B_W & A27 & A28 & !A29) ;
VGR_WRITE_BCSR_7 = (!BrdContRegCs_B & !DVal_B & R_B_W & A27 & A28 & A29) ;

VGR_READ_BCSR_0 = (!BrdContRegCs_B & !R_B_W & !A27 & !A28 & !A29) ;
VGR_READ_BCSR_1 = (!BrdContRegCs_B & !R_B_W & !A27 & !A28 & A29) ;
VGR_READ_BCSR_2 = (!BrdContRegCs_B & !R_B_W & !A27 & A28 & !A29) ;
```

```
VGR_READ_BCSR_3 = (!BrdContRegCs_B & !R_B_W & !A27 & A28 & A29) ;
VGR_READ_BCSR_4 = (!BrdContRegCs_B & !R_B_W & A27 & !A28 & !A29) ;
VGR_READ_BCSR_5 = (!BrdContRegCs_B & !R_B_W & A27 & !A28 & A29) ;
VGR_READ_BCSR_6 = (!BrdContRegCs_B & !R_B_W & A27 & A28 & !A29) ;
VGR_READ_BCSR_7 = (!BrdContRegCs_B & !R_B_W & A27 & A28 & A29) ;

“*****************************************************************************
“*****************************************************************************
“* BCSR 0 definitions.
“*****************************************************************************
“*****************************************************************************
PBI_IN_ACTIVE = 0;
DIMM_SIZE_16M = 0;
L2CACHE_INHIBITED = 0 ;
L2CACHE_FLUSHED = 0 ;
L2CACHE_LOCKED = 0 ;
L2CACHE_CLEARED = 0 ;
SIGNAL_LAMP_ON = 0 ;

“***********************************************
“******** Power On Defaults Assignments *********
“***********************************************
PBI_PON_DEFAULT = PBI_IN_ACTIVE;
DIMM_SIZE_PON_DEFAULT = DIMM_SIZE_16M;
L2CACHE_INH_PON_DEFAULT = L2CACHE_INHIBITED ;
L2CACHE_FLUSH_PON_DEFAULT = !L2CACHE_FLUSHED ;
L2CACHE_LOCK_PON_DEFAULT = !L2CACHE_LOCKED ;
L2CACHE_CLEAR_PON_DEFAULT = !L2CACHE_CLEARED ;
SIGNAL_LAMP0_PON_DEFAULT = !SIGNAL_LAMP_ON ;
SIGNAL_LAMP1_PON_DEFAULT = !SIGNAL_LAMP_ON ;

“*****************************************
“******** Data Bits Assignments **********
“*****************************************
PBI_DATA_BIT = [D0] ;
DIMM_SIZE_DATA_BIT = [D1] ;
L2CACHE_INH_DATA_BIT = [D2] ;
L2CACHE_FLUSH_DATA_BIT = [D3] ;
L2CACHE_LOCK_DATA_BIT = [D4] ;
L2CACHE_CLEAR_DATA_BIT = [D5] ;
```

```
SIGNAL_LAMP0_DATA_BIT = [D6] ;
SIGNAL_LAMP1_DATA_BIT = [D7] ;

“*****************************************************************************
“*****************************************************************************
“* BCSR 1 definitions.
“*****************************************************************************
“*****************************************************************************
BCSR_BOOT = 0 ;“ bcsrConfEn = 0 Hard Reset Conf Word from BCSR
MEMORY_BOOT = 1 ;“ bcsrConfEn = 1 Hard Reset Conf from EEPROM/FLASH
FLASH_BOOT = 0 ;“ boot_device_B = 0
EEPROM_BOOT = 1 ;“ boot_device_B = 1
ATM_ENABLED = 0 ;
ATM_RESET_ACTIVE = 0 ;
FETH_ENABLED = 0 ;
FETH_RESET_ACTIVE = 0 ;
RS232_1_ENABLE = 0 ;
RS232_2_ENABLE = 0 ;

“***********************************************
“******* Power On Defaults Assignments *********
“***********************************************
ATM_ENABLE_PON_DEFAULT = !ATM_ENABLED ;
ATM_RESET_PON_DEFAULT = !ATM_RESET_ACTIVE ;
FETH_ENABLE_PON_DEFAULT = !FETH_ENABLED ;
FETH_RESET_PON_DEFAULT = !FETH_RESET_ACTIVE ;
RS232_1_ENABLE_PON_DEFAULT = !RS232_1_ENABLE ;
RS232_2_ENABLE_PON_DEFAULT = !RS232_2_ENABLE ;

“*****************************************
“******* Data Bits Assignments **************
“*****************************************
CONF_WORD_DATA_BIT =      [D0] ;
BOOT_DEVICE_DATA_BIT =    [D1] ;
ATM_ENABLE_DATA_BIT =     [D2] ;
ATM_RESET_DATA_BIT =      [D3] ;
FETH_ENABLE_DATA_BIT =    [D4] ;
FETH_RESET_DATA_BIT =     [D5] ;
RS232_1_ENABLE_DATA_BIT = [D6] ;
RS232_2_ENABLE_DATA_BIT = [D7] ;
```

```
“*****************************************************************************
“*****************************************************************************
“* Jtag Command / Status reg definitions.
“*****************************************************************************
“*****************************************************************************
JTAG_ENABLED = 1;

STATE_JTAG_ENABLED = (JtagEn.fb == JTAG_ENABLED);

“***********************************************
“******* Power On Defaults Assignments *********
“***********************************************
JTAG_ENABLE_PON_DEFAULT = !JTAG_ENABLED ; “ eng compatible

“*******************************************
“******* Data Bits Assignments *************
“*******************************************
JTAG_ENABLE_DATA_BIT = [D0];

“*****************************************************************************
“* Flash Declarations.
“*****************************************************************************
FLASH_ENABLE_ACTIVE = 0 ;

“ the presence detect encoding for the below is fictional
“ needs to be updated with real data.

CP29020    = (F_PD == 8) ;   “ 1 X 2 MByte bank
SM73228XU1 = (F_PD == 2) ;   “ 1 X 8 MByte bank
SM73248XU2 = (F_PD == 1) ;   “ 2 X 8 MByte banks
SM73288XU4 = (F_PD == 0) ;   “ 4 X 8 MByte banks

FLASH_BANK1 = ( CP29020 #
            SM73228XU1 #
      (SM73248XU2 & !A8) #
           (SM73288XU4 & !A7 & !A8) ) ;

FLASH_BANK2 = ( (SM73248XU2 & A8) #
           (SM73288XU4 & !A7 & A8) ) ;
```

```
FLASH_BANK3 = ( A7 & !A8 & SM73288XU4 ) ;

FLASH_BANK4 = ( A7 & A8 & SM73288XU4 ) ;

"*****************************************************************************
"* ATM UNI Declarations.
"*****************************************************************************

"*****************************************************************************
"* Reset Declarations.
"*****************************************************************************

HARD_RESET_ACTIVE = 0 ;
SOFT_RESET_ACTIVE = 0 ;

HARD_RESET_ASSERTED = (SyncHardReset_B.fb == HARD_RESET_ACTIVE) ;

"*****************************************************************************
"* data buffers enable.
"*****************************************************************************
BUFFER_DISABLED = 1 ;
BUFFER_ENABLED = !BUFFER_DISABLED ;

BUFFER_HOLD_OFF = (HoldOffCnt.fb != 0) ; " the delay is required for read as well
                        " since a fast device (eg bcsr) may
                        " content with the flash/eeprom

END_OF_FLASH_EEPROM_READ = !DVal_B & (!Cs0_B # !Cs4_B) & !R_B_W & DSyncHardReset_B.fb ;
 " end of flash/eeprom read cycle.
                        " not during hard reset config

END_OF_PCI_INT_CONT_READ =  !DVal_B & !PCIIntContCs_B & !R_B_W ;
 " end of PCI Interrupt Controller read cycle.

END_OF_ATM_READ = !DVal_B & !AtmUniCsIn_B & !R_B_W ; " end of atm uni m/p i/f read cycle

END_OF_OTHER_CYCLE = (!DVal_B & Cs0_B & Cs4_B & AtmUniCsIn_B & PCIIntContCs_B #  " another access or
            !DVal_B & !AtmUniCsIn_B & R_B_W #       " atm uni write or
            !DVal_B & !ToolCs1_B & R_B_W #      " tool 1 write or
```

```
                !DVal_B & !ToolCs2_B & R_B_W #       " tool 2 write or
    !DVal_B & (!Cs0_B # !Cs4_B) & R_B_W #     " flash/eeprom write
    !DVal_B & !PCIIntContCs_B & R_B_W) ;    " PCI int cont write

"*****************************************************************************
"* Hard Reset Configuration Logic
"*****************************************************************************
HRESET_CFG_IN_BCSR = (bcsrConfEn == 1); " HRESET Conf Word in BCSR
HRESET_BOOT_IN_FLASH = ((bcsrConfEn == 0) & (boot_device_B == 0));
" HRESET Conf Word and Boot Code in FLASH
BOOT_IN_FLASH = ((bcsrConfEn == 1) & (boot_device_B == 0));
" HRESET Conf Word in BCSR and Boot Code in FLASH
HRESET_BOOT_IN_EEPROM = ((bcsrConfEn == 0) & (boot_device_B == 1));
" HRESET Conf Word and Boot Code in EEPROM
BOOT_IN_EEPROM = ((bcsrConfEn == 1) & (boot_device_B == 1));
" HRESET Conf Word in BCSR and Boot Code in EEPROM
HARD_RESET_ASSERTION = ( (HardReset_B == 0) & (SyncHardReset_B.fb == 0) &
                (DSyncHardReset_B.fb == 1) );

CS0_ASSERTED = (Cs0_B == 0);
CS4_ASSERTED = (Cs4_B == 0);

FIRST_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 0) & HRESET_CFG_IN_BCSR &
!R_B_W);
SCND_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 1) & HRESET_CFG_IN_BCSR &
!R_B_W);
THIRD_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 2) & HRESET_CFG_IN_BCSR &
!R_B_W);
FORTH_CFG_BYTE_READ = (CS0_ASSERTED & !DSyncHardReset_B.fb & (ConfAdd == 3) & HRESET_CFG_IN_BCSR
& !R_B_W);

"*****************************************************************************
"*****************************************************************************
"* JTAG LOGIC definitions.
"*****************************************************************************
"*****************************************************************************

"* signals' groups
"*-----------------

JtagShiftDR =       [JtagShiftDR0..JtagShiftDR7] ;
JtagC_S_Reg =       [JtagEn.fb,0,0,0,0,0,0,JtagReceiveFull.fb];
```

```
JtagState =        [JtagState0..JtagState3] ;
JtagShiftIR =      [JtagShiftIR0..JtagShiftIR2];
JtagIR    =        [JtagIR0..JtagIR2] ;
FallingTckSignals = [JtagResetState,
                     JtagShiftIrState,
                     JtagShiftDrState,
                     JtagTdoEnable];
Read = [BrdContRegCs_B,DVal_B,R_B_W,A27,A28,A29] ;  " for simulation only

"* Constant definition
"*---------------------

JTAG_RESET = 0;
JTAG_IDLE = 1;

JTAG_SELECT_DR = 2;
JTAG_CAPTURE_DR = 3;
JTAG_SHIFT_DR = 4;
JTAG_EXIT1_DR = 5;
JTAG_PAUSE_DR = 6;
JTAG_EXIT2_DR = 7;
JTAG_UPDATE_DR = 8;

JTAG_SELECT_IR = 9;
JTAG_CAPTURE_IR = 10;
JTAG_SHIFT_IR = 11;
JTAG_EXIT1_IR = 12;
JTAG_PAUSE_IR = 13;
JTAG_EXIT2_IR = 14;
JTAG_UPDATE_IR = 15;

STATE_JTAG_RESET = (JtagState.fb == JTAG_RESET) ;
STATE_JTAG_IDLE = (JtagState.fb == JTAG_IDLE) ;

STATE_JTAG_SELECT_DR = (JtagState.fb == JTAG_SELECT_DR) ;
STATE_JTAG_CAPTURE_DR = (JtagState.fb == JTAG_CAPTURE_DR) ;
STATE_JTAG_SHIFT_DR = (JtagState.fb == JTAG_SHIFT_DR) ;
STATE_JTAG_EXIT1_DR = (JtagState.fb == JTAG_EXIT1_DR) ;
STATE_JTAG_PAUSE_DR = (JtagState.fb == JTAG_PAUSE_DR) ;
STATE_JTAG_EXIT2_DR = (JtagState.fb == JTAG_EXIT2_DR) ;
```

```
STATE_JTAG_UPDATE_DR = (JtagState.fb == JTAG_UPDATE_DR) ;

STATE_JTAG_SELECT_IR = (JtagState.fb == JTAG_SELECT_IR) ;
STATE_JTAG_CAPTURE_IR = (JtagState.fb == JTAG_CAPTURE_IR) ;
STATE_JTAG_SHIFT_IR = (JtagState.fb == JTAG_SHIFT_IR) ;
STATE_JTAG_EXIT1_IR = (JtagState.fb == JTAG_EXIT1_IR) ;
STATE_JTAG_PAUSE_IR = (JtagState.fb == JTAG_PAUSE_IR) ;
STATE_JTAG_EXIT2_IR = (JtagState.fb == JTAG_EXIT2_IR) ;
STATE_JTAG_UPDATE_IR = (JtagState.fb == JTAG_UPDATE_IR) ;

"* Instruction codes

INST_CODE_BYPASS = 7;
INST_CODE_EXTEST = 0;
INST_CODE_DOWNLOAD = 1;
INST_CODE_UPLOAD = 2;     " not supported for 1st implementaion
INST_CODE_PON_RESET = 6;
INST_CODE_UN_IMPLEMENTED = (^b011 # ^b100 # ^b101);



NEXT_INST_BYPASS = (JtagShiftIR.fb == INST_CODE_BYPASS);
NEXT_INST_EXTEST = (JtagShiftIR.fb == INST_CODE_EXTEST);
NEXT_INST_DOWNLOAD = (JtagShiftIR.fb == INST_CODE_DOWNLOAD);
NEXT_INST_UPLOAD = (JtagShiftIR.fb == INST_CODE_UPLOAD);
NEXT_INST_PON_RESET = (JtagShiftIR.fb == INST_CODE_PON_RESET);
NEXT_INST_UN_IMPLEMENTED = (JtagShiftIR.fb == INST_CODE_UN_IMPLEMENTED);

INST_IS_BYPASS = (JtagIR.fb == INST_CODE_BYPASS);
INST_IS_EXTEST = (JtagIR.fb == INST_CODE_EXTEST);
INST_IS_DOWNLOAD = (JtagIR.fb == INST_CODE_DOWNLOAD);
INST_IS_UPLOAD = (JtagIR.fb == INST_CODE_UPLOAD);
INST_IS_PON_RESET = (JtagIR.fb == INST_CODE_PON_RESET);

READ_JTAG_DOWNLOAD_CSR = VGR_READ_BCSR_6;
WRITE_JTAG_DOWNLOAD_CSR = VGR_WRITE_BCSR_6;
READ_JTAG_DOWNLOAD_DATA = VGR_READ_BCSR_7;

RECEIVE_FULL = 1;

JTAG_DOWNLOAD_SHIFT_REG_FULL = (JtagReceiveFull.fb == RECEIVE_FULL);
```

```
"*****************************************************************************
"* Equations, state diagrams.                                                *
"*****************************************************************************

equations

"*****************************************************************************
"*****************************************************************************
"* BCSR 0
"*****************************************************************************
"*****************************************************************************
equations

ClockedContReg.clk = SYSCLK ;
ClockedContReg.ar = 0;
ClockedContReg.ap = 0;
DrivenContReg.oe = ^hfff ;

"*****************************************************************************
state_diagram PBI
state PBI_IN_ACTIVE:
if (VGR_WRITE_BCSR_0 &
(PBI_DATA_BIT.pin == !PBI_IN_ACTIVE) &
 (!PON_RESET # (PBI_PON_DEFAULT != PBI_IN_ACTIVE)) #
 (PON_RESET & (PBI_PON_DEFAULT == !PBI_IN_ACTIVE)) ) then
!PBI_IN_ACTIVE
else
PBI_IN_ACTIVE ;
state !PBI_IN_ACTIVE:
if (VGR_WRITE_BCSR_0 &
(PBI_DATA_BIT.pin == PBI_IN_ACTIVE) &
 (!PON_RESET # (PBI_PON_DEFAULT != !PBI_IN_ACTIVE)) #
 (PON_RESET & (PBI_PON_DEFAULT == PBI_IN_ACTIVE)) ) then
PBI_IN_ACTIVE
else
!PBI_IN_ACTIVE ;
"*****************************************************************************
state_diagram DimmSize
state DIMM_SIZE_16M:
```

```
if (VGR_WRITE_BCSR_0 &
(DIMM_SIZE_DATA_BIT.pin == !DIMM_SIZE_16M) &
 (!PON_RESET # (DIMM_SIZE_PON_DEFAULT != DIMM_SIZE_16M)) #
 (PON_RESET & (DIMM_SIZE_PON_DEFAULT == !DIMM_SIZE_16M)) ) then
!DIMM_SIZE_16M
else
DIMM_SIZE_16M ;
state !DIMM_SIZE_16M:
if (VGR_WRITE_BCSR_0 &
(DIMM_SIZE_DATA_BIT.pin == DIMM_SIZE_16M) &
 (!PON_RESET # (DIMM_SIZE_PON_DEFAULT != !DIMM_SIZE_16M)) #
 (PON_RESET & (DIMM_SIZE_PON_DEFAULT == DIMM_SIZE_16M)) ) then
DIMM_SIZE_16M
else
!DIMM_SIZE_16M ;
"*******************************************************************************
state_diagram L2Inh_B
state L2CACHE_INHIBITED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_INH_DATA_BIT.pin == !L2CACHE_INHIBITED) &
 (!PON_RESET # (L2CACHE_INH_PON_DEFAULT != L2CACHE_INHIBITED)) #
 (PON_RESET & (L2CACHE_INH_PON_DEFAULT == !L2CACHE_INHIBITED)) ) then
!L2CACHE_INHIBITED
else
L2CACHE_INHIBITED ;
state !L2CACHE_INHIBITED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_INH_DATA_BIT.pin == L2CACHE_INHIBITED) &
 (!PON_RESET # (L2CACHE_INH_PON_DEFAULT != !L2CACHE_INHIBITED)) #
 (PON_RESET & (L2CACHE_INH_PON_DEFAULT == L2CACHE_INHIBITED)) ) then
L2CACHE_INHIBITED
else
!L2CACHE_INHIBITED ;
"*******************************************************************************
state_diagram L2Flush_B
state L2CACHE_FLUSHED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_FLUSH_DATA_BIT.pin == !L2CACHE_FLUSHED) &
 (!PON_RESET # (L2CACHE_FLUSH_PON_DEFAULT != L2CACHE_FLUSHED)) #
 (PON_RESET & (L2CACHE_FLUSH_PON_DEFAULT == !L2CACHE_FLUSHED)) ) then
```

```
!L2CACHE_FLUSHED
else
L2CACHE_FLUSHED ;
state !L2CACHE_FLUSHED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_FLUSH_DATA_BIT.pin == L2CACHE_FLUSHED) &
 (!PON_RESET # (L2CACHE_FLUSH_PON_DEFAULT != !L2CACHE_FLUSHED)) #
 (PON_RESET & (L2CACHE_FLUSH_PON_DEFAULT == L2CACHE_FLUSHED)) ) then
L2CACHE_FLUSHED
else
!L2CACHE_FLUSHED ;
"*****************************************************************************
state_diagram L2Lock_B
state L2CACHE_LOCKED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_LOCK_DATA_BIT.pin == !L2CACHE_LOCKED) &
 (!PON_RESET # (L2CACHE_LOCK_PON_DEFAULT != L2CACHE_LOCKED)) #
 (PON_RESET & (L2CACHE_LOCK_PON_DEFAULT == !L2CACHE_LOCKED)) ) then
!L2CACHE_LOCKED
else
L2CACHE_LOCKED ;
state !L2CACHE_LOCKED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_LOCK_DATA_BIT.pin == L2CACHE_LOCKED) &
 (!PON_RESET # (L2CACHE_LOCK_PON_DEFAULT != !L2CACHE_LOCKED)) #
 (PON_RESET & (L2CACHE_LOCK_PON_DEFAULT == L2CACHE_LOCKED)) ) then
L2CACHE_LOCKED
else
!L2CACHE_LOCKED ;

"*****************************************************************************
state_diagram L2Clear_B
state L2CACHE_CLEARED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_CLEAR_DATA_BIT.pin == !L2CACHE_CLEARED) &
 (!PON_RESET # (L2CACHE_CLEAR_PON_DEFAULT != L2CACHE_CLEARED)) #
 (PON_RESET & (L2CACHE_CLEAR_PON_DEFAULT == !L2CACHE_CLEARED)) ) then
!L2CACHE_CLEARED
else
L2CACHE_CLEARED ;
```

```
state !L2CACHE_CLEARED:
if (VGR_WRITE_BCSR_0 &
(L2CACHE_CLEAR_DATA_BIT.pin == L2CACHE_CLEARED) &
 (!PON_RESET # (L2CACHE_CLEAR_PON_DEFAULT != !L2CACHE_CLEARED)) #
 (PON_RESET & (L2CACHE_CLEAR_PON_DEFAULT == L2CACHE_CLEARED)) ) then
L2CACHE_CLEARED
else
!L2CACHE_CLEARED ;
"*****************************************************************************
state_diagram SignaLamp0_B
state SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP0_DATA_BIT.pin == !SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP0_PON_DEFAULT != SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP0_PON_DEFAULT == !SIGNAL_LAMP_ON)) ) then
!SIGNAL_LAMP_ON
else
SIGNAL_LAMP_ON ;
state !SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP0_DATA_BIT.pin == SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP0_PON_DEFAULT != !SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP0_PON_DEFAULT == SIGNAL_LAMP_ON)) ) then
SIGNAL_LAMP_ON
else
!SIGNAL_LAMP_ON ;
"*****************************************************************************
state_diagram SignaLamp1_B
state SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP1_DATA_BIT.pin == !SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP1_PON_DEFAULT != SIGNAL_LAMP_ON)) #
 (PON_RESET & (SIGNAL_LAMP1_PON_DEFAULT == !SIGNAL_LAMP_ON)) ) then
!SIGNAL_LAMP_ON
else
SIGNAL_LAMP_ON ;
state !SIGNAL_LAMP_ON:
if (VGR_WRITE_BCSR_0 &
(SIGNAL_LAMP1_DATA_BIT.pin == SIGNAL_LAMP_ON) &
 (!PON_RESET # (SIGNAL_LAMP1_PON_DEFAULT != !SIGNAL_LAMP_ON)) #
```

```
 (PON_RESET & (SIGNAL_LAMP1_PON_DEFAULT == SIGNAL_LAMP_ON)) ) then
SIGNAL_LAMP_ON
else
!SIGNAL_LAMP_ON ;

“*****************************************************************************
“*****************************************************************************
“* BCSR1 State Machines
“*****************************************************************************
“*****************************************************************************
state_diagram AtmEn_B
state ATM_ENABLED:
if (VGR_WRITE_BCSR_1 &
(ATM_ENABLE_DATA_BIT.pin == !ATM_ENABLED) &
 (!PON_RESET # (ATM_ENABLE_PON_DEFAULT != ATM_ENABLED)) #
 (PON_RESET & (ATM_ENABLE_PON_DEFAULT == !ATM_ENABLED)) ) then
!ATM_ENABLED
else
ATM_ENABLED ;
state !ATM_ENABLED:
if (VGR_WRITE_BCSR_1 &
(ATM_ENABLE_DATA_BIT.pin == ATM_ENABLED) &
 (!PON_RESET # (ATM_ENABLE_PON_DEFAULT != !ATM_ENABLED)) #
 (PON_RESET & (ATM_ENABLE_PON_DEFAULT == ATM_ENABLED)) ) then
ATM_ENABLED
else
!ATM_ENABLED ;
“*****************************************************************************
state_diagram AtmRst_B
state ATM_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(ATM_RESET_DATA_BIT.pin == !ATM_RESET_ACTIVE) &
 (!PON_RESET # (ATM_RESET_PON_DEFAULT != ATM_RESET_ACTIVE)) #
 (PON_RESET & (ATM_RESET_PON_DEFAULT == !ATM_RESET_ACTIVE)) ) then
!ATM_RESET_ACTIVE
else
ATM_RESET_ACTIVE ;
state !ATM_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(ATM_RESET_DATA_BIT.pin == ATM_RESET_ACTIVE) &
```

```
 (!PON_RESET # (ATM_RESET_PON_DEFAULT != !ATM_RESET_ACTIVE)) #
 (PON_RESET & (ATM_RESET_PON_DEFAULT == ATM_RESET_ACTIVE)) ) then
ATM_RESET_ACTIVE
else
!ATM_RESET_ACTIVE ;
"*****************************************************************************
state_diagram FEthEn_B
state FETH_ENABLED:
if (VGR_WRITE_BCSR_1 &
(FETH_ENABLE_DATA_BIT.pin == !FETH_ENABLED) &
 (!PON_RESET # (FETH_ENABLE_PON_DEFAULT != FETH_ENABLED)) #
 (PON_RESET & (FETH_ENABLE_PON_DEFAULT == !FETH_ENABLED)) ) then
!FETH_ENABLED
else
FETH_ENABLED ;
state !FETH_ENABLED:
if (VGR_WRITE_BCSR_1 &
(FETH_ENABLE_DATA_BIT.pin == FETH_ENABLED) &
 (!PON_RESET # (FETH_ENABLE_PON_DEFAULT != !FETH_ENABLED)) #
 (PON_RESET & (FETH_ENABLE_PON_DEFAULT == FETH_ENABLED)) ) then
FETH_ENABLED
else
!FETH_ENABLED ;
"*****************************************************************************
state_diagram FEthRst_B
state FETH_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(FETH_RESET_DATA_BIT.pin == !FETH_RESET_ACTIVE) &
 (!PON_RESET # (FETH_RESET_PON_DEFAULT != FETH_RESET_ACTIVE)) #
 (PON_RESET & (FETH_RESET_PON_DEFAULT == !FETH_RESET_ACTIVE)) ) then
!FETH_RESET_ACTIVE
else
FETH_RESET_ACTIVE ;
state !FETH_RESET_ACTIVE:
if (VGR_WRITE_BCSR_1 &
(FETH_RESET_DATA_BIT.pin == FETH_RESET_ACTIVE) &
 (!PON_RESET # (FETH_RESET_PON_DEFAULT != !FETH_RESET_ACTIVE)) #
 (PON_RESET & (FETH_RESET_PON_DEFAULT == FETH_RESET_ACTIVE)) ) then
FETH_RESET_ACTIVE
else
```

```
!FETH_RESET_ACTIVE ;
"*****************************************************************************
state_diagram RS232En1_B
state RS232_1_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_1_ENABLE_DATA_BIT.pin == !RS232_1_ENABLE) &
 (!PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != RS232_1_ENABLE)) #
 (PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == !RS232_1_ENABLE)) ) then
!RS232_1_ENABLE
else
RS232_1_ENABLE ;
state !RS232_1_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_1_ENABLE_DATA_BIT.pin == RS232_1_ENABLE) &
 (!PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != !RS232_1_ENABLE)) #
 (PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == RS232_1_ENABLE)) ) then
RS232_1_ENABLE
else
!RS232_1_ENABLE ;
"*****************************************************************************
state_diagram RS232En2_B
state RS232_2_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_2_ENABLE_DATA_BIT.pin == !RS232_2_ENABLE) &
 (!PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != RS232_2_ENABLE)) #
 (PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == !RS232_2_ENABLE)) ) then
!RS232_2_ENABLE
else
RS232_2_ENABLE ;
state !RS232_2_ENABLE:
if (VGR_WRITE_BCSR_1 &
(RS232_2_ENABLE_DATA_BIT.pin == RS232_2_ENABLE) &
 (!PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != !RS232_2_ENABLE)) #
 (PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == RS232_2_ENABLE)) ) then
RS232_2_ENABLE
else
!RS232_2_ENABLE ;

"*****************************************************************************
"*****************************************************************************
```

```
"* BCSR 3
"*****************************************************************************
"*****************************************************************************
@ifdef JTAG {

state_diagram JtagEn
state JTAG_ENABLED:
if (WRITE_JTAG_DOWNLOAD_CSR &
(JTAG_ENABLE_DATA_BIT.pin == !JTAG_ENABLED) &
 (!PON_RESET # (JTAG_ENABLE_PON_DEFAULT != JTAG_ENABLED)) #
 (PON_RESET & (JTAG_ENABLE_PON_DEFAULT == !JTAG_ENABLED)) ) then
!JTAG_ENABLED
else
JTAG_ENABLED ;
state !JTAG_ENABLED:
if (WRITE_JTAG_DOWNLOAD_CSR &
(JTAG_ENABLE_DATA_BIT.pin == JTAG_ENABLED) &
 (!PON_RESET # (JTAG_ENABLE_PON_DEFAULT != !JTAG_ENABLED)) #
 (PON_RESET & (JTAG_ENABLE_PON_DEFAULT == JTAG_ENABLED)) ) then
JTAG_ENABLED
else
!JTAG_ENABLED ;   }
"*****************************************************************************
"*****************************************************************************
"*****************************************************************************
" External Read Registers' Chip-Selects
"*****************************************************************************
"*****************************************************************************
equations

Bcsr2Cs_B.oe = H ;
Bcsr4Cs_B.oe = H ;

!Bcsr2Cs_B = VGR_READ_BCSR_2 ;
!Bcsr4Cs_B = VGR_READ_BCSR_4 ;

"*****************************************************************************
"*****************************************************************************
"* Read Registers.
"* All registers have read capabilty. (BCSR2 and BCSR4 are read externally)
```

```
"*******************************************************************************
"*******************************************************************************
equations
DataOe = VGR_READ_BCSR_0 #
        VGR_READ_BCSR_1 #
@ifdef JTAG {    READ_JTAG_DOWNLOAD_DATA #
                 READ_JTAG_DOWNLOAD_CSR # }
        (HRESET_CFG_IN_BCSR & CS0_ASSERTED & !DSyncHardReset_B.fb) ;
Data.oe = DataOe.fb ;

when (VGR_READ_BCSR_0) then
Data = ReadBcsr0 ;
else when (VGR_READ_BCSR_1) then
Data = ReadBcsr1 ;
@ifdef JTAG {
        else when (READ_JTAG_DOWNLOAD_DATA) then
                Data = JtagShiftDR.fb;
        else when (READ_JTAG_DOWNLOAD_CSR) then
                Data = JtagC_S_Reg ;      }
        else when (FIRST_CFG_BYTE_READ) then
                Data = CfgByte0;
        else when (SCND_CFG_BYTE_READ) then
                Data = CfgByte1;
        else when (THIRD_CFG_BYTE_READ) then
                Data = CfgByte2;
        else when (FORTH_CFG_BYTE_READ) then
                Data = CfgByte3;

"**********************   brd_ctl   *********************************************
"*******************************************************************************
"*******************************************************************************
"* Reset Logic
"*******************************************************************************
"*******************************************************************************
"*******************************************************************************
equations

Reset.oe = ResetEn ;

Reset = 0 ;" open drain
```

```
RstDeb1 = !( Rst1 & (!( RstDeb1.com & Rst0) ) ) ;    " Reset push-button debouncer

AbrDeb1 = !( Abr1 & (!( AbrDeb1.com & Abr0) ) ) ;    " Abort push-button debouncer

HardResetEn = RstDeb1.com & AbrDeb1.com  ;" both buttons are depressed;

SoftResetEn = RstDeb1.com & !AbrDeb1.com ;" only reset button depressed

TransRst.oe = 3 ;" transceivers' reset, always enabled.

!AtmRstOut_B = !AtmRst_B.fb # !HardReset_B ;

!FEthRstOut_B = !FEthRst_B.fb # !HardReset_B ;

"*****************************************************************************
"* Hard reset configuration
"*****************************************************************************
equations

RstConf_B.oe = H;

RstConf_B = L;

"*****************************************************************************
"* NMI generation
"*****************************************************************************
equations

NMI_B.oe = NMIEn ;

NMI_B = 0 ;" O.D.

NMIEn = !RstDeb1.com & AbrDeb1.com ;" only abort button depressed

"*****************************************************************************
"* local data buffers enable
"*****************************************************************************
equations
```

```
SyncHardReset_B.clk = SYSCLK ;
SyncHardReset_B.ar = 0;
SyncHardReset_B.ap = 0;

DSyncHardReset_B.clk = SYSCLK ;
DSyncHardReset_B.ar = 0;
DSyncHardReset_B.ap = 0;

SyncHardReset_B := HardReset_B ;
DSyncHardReset_B := SyncHardReset_B.fb ;

DataBufEn_B.oe = H ;

!DataBufEn_B = ( !Cs0_B #     " covers also hard reset config
 !Cs4_B #
 !BrdContRegCs_B #
 !PCIIntContCs_B #
 !AtmUniCsOut_B #   " provides data-hold for write
 !ToolCs1_B #
 !ToolCs2_B ) &
      ( !BUFFER_HOLD_OFF ) ;

ToolDataBufEn_B.oe = H ;

!ToolDataBufEn_B = ( !ToolCs1_B #
                !ToolCs2_B ) &
  ( !BUFFER_HOLD_OFF ) ;

"*******************************************************************************
"* local data buffers disable (data contention protection)
"*******************************************************************************
"* Since with Voyager, hard-reset conf is read from flash/eeprom during HRESET
"* asserted and since these are all consequitive read cycles and since
"* the cycles following hard reset are also reads (boot) the hold-off
"* state machine may be left in NO_HOLD_OFF for HRESET_B asserted duration
"* without warrying about contention between flash and data buffers.

equations

HoldOffCnt.clk = SYSCLK ;
```

```
HoldOffCnt.ar = 0;
HoldOffCnt.ap = 0;

HoldOffTc = (HoldOffCnt.fb == 3) ;

when ( (((END_OF_FLASH_EEPROM_READ # END_OF_ATM_READ )
     & (HoldOffCnt.fb == 0)) #
       (HoldOffCnt.fb != 0)) & !(HoldOffCnt.fb == 4) & DSyncHardReset_B.fb  ) then
    HoldOffCnt := HoldOffCnt.fb + 1 ;
else
    HoldOffCnt := 0 ;

"*******************************************************************************
"* Flash/EEPROM Chip Selects
"*******************************************************************************
equations

FlashCsOut.oe = ^hf ;

!FlashCs1_B = CS0_ASSERTED & FLASH_BANK1 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK1 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
    CS4_ASSERTED & FLASH_BANK1 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs2_B = CS0_ASSERTED & FLASH_BANK2 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK2 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK2 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs3_B = CS0_ASSERTED & FLASH_BANK3 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK3 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK3 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

!FlashCs4_B = CS0_ASSERTED & FLASH_BANK4 & HRESET_BOOT_IN_FLASH #
          CS0_ASSERTED & FLASH_BANK4 & BOOT_IN_FLASH & DSyncHardReset_B.fb #
          CS4_ASSERTED & FLASH_BANK4 & (HRESET_BOOT_IN_EEPROM # BOOT_IN_EEPROM);

EEpromCs_B.oe = H ;

!EEpromCs_B = CS0_ASSERTED & HRESET_BOOT_IN_EEPROM #
    CS0_ASSERTED & BOOT_IN_EEPROM & DSyncHardReset_B.fb #
    CS4_ASSERTED & (HRESET_BOOT_IN_FLASH # BOOT_IN_FLASH) ;
```

```
"*****************************************************************************
"* ATM UNI Chip Select
"*****************************************************************************
equations

AtmUniCsOut_B.oe = H ;

!AtmUniCsOut_B = !AtmUniCsIn_B;

"*****************************************************************************
"* Power On Reset
"*****************************************************************************
equations

S_PORIn_B.clk = SYSCLK ;
S_PORIn_B.ar = 0;
S_PORIn_B.ap = 0;

S_PORIn_B := PORIn_B ;

"*****************************************************************************
"*****************************************************************************
"* JTAG LOGIC equations
"*****************************************************************************
"*****************************************************************************
equations

JtagStateReset = (!Trst_B # !PORIn_B) ;  " only for jtag state machine
JtagReset = (!Trst_B # !PORIn_B # !JtagResetState.fb);  " global reset

JtagState.clk = Tck;
JtagState.ar = JtagStateReset ;
JtagState.ap = 0;

"*****************************************************************************
"* Standard JTAG state machine

state_diagram JtagState
    state JTAG_RESET:
```

```
    if (!Tms) then
        JTAG_IDLE
    else
        JTAG_RESET;
state JTAG_IDLE:
    if (Tms) then
        JTAG_SELECT_DR
    else
        JTAG_IDLE;
state JTAG_SELECT_DR:        " DR
    if (!Tms) then
        JTAG_CAPTURE_DR
    else
        JTAG_SELECT_IR;
state JTAG_CAPTURE_DR:
    if (!Tms) then
        JTAG_SHIFT_DR
    else
        JTAG_EXIT1_DR;
state JTAG_SHIFT_DR:
    if (Tms) then
        JTAG_EXIT1_DR
    else
        JTAG_SHIFT_DR;
state JTAG_EXIT1_DR:
    if (!Tms) then
        JTAG_PAUSE_DR
    else
        JTAG_UPDATE_DR;
state JTAG_PAUSE_DR:
    if (Tms) then
        JTAG_EXIT2_DR
    else
        JTAG_PAUSE_DR;
state JTAG_EXIT2_DR:
    if (Tms) then
        JTAG_UPDATE_DR
    else
        JTAG_SHIFT_DR;
state JTAG_UPDATE_DR:
```

```
        if (!Tms) then
            JTAG_IDLE
        else
            JTAG_SELECT_DR;
    state JTAG_SELECT_IR:           " IR
        if (!Tms) then
            JTAG_CAPTURE_IR
        else
            JTAG_RESET;
    state JTAG_CAPTURE_IR:
        if (!Tms) then
            JTAG_SHIFT_IR
        else
            JTAG_EXIT1_IR;
    state JTAG_SHIFT_IR:
        if (Tms) then
            JTAG_EXIT1_IR
        else
            JTAG_SHIFT_IR;
    state JTAG_EXIT1_IR:
        if (!Tms) then
            JTAG_PAUSE_IR
        else
            JTAG_UPDATE_IR;
    state JTAG_PAUSE_IR:
        if (Tms) then
            JTAG_EXIT2_IR
        else
            JTAG_PAUSE_IR;
    state JTAG_EXIT2_IR:
        if (Tms) then
            JTAG_UPDATE_IR
        else
            JTAG_SHIFT_IR;
    state JTAG_UPDATE_IR:
        if (!Tms) then
            JTAG_IDLE
        else
            JTAG_SELECT_DR;
"*******************************************************************************
```

```
"* Jtag Dedicated State Signals (Falling Edge TCK)

equations

FallingTckSignals.clk = !Tck;
FallingTckSignals.ar = JtagStateReset ;
FallingTckSignals.ap = 0;

!JtagResetState := STATE_JTAG_RESET;   " Active-Low

JtagShiftIrState := STATE_JTAG_SHIFT_IR;

JtagShiftDrState := STATE_JTAG_SHIFT_DR;

JtagTdoEnable := (STATE_JTAG_SHIFT_IR # STATE_JTAG_SHIFT_DR) ;

"*******************************************************************************
"* Jtag Instruction Shift Register

equations

JtagShiftIR.clk = Tck;
JtagShiftIR.ap = JtagReset ;   "  reset -> bypass
JtagShiftIR.ar = 0;

when (JtagShiftIrState.fb & STATE_JTAG_ENABLED) then
    JtagShiftIR := [Tdi,JtagShiftIR0.fb,JtagShiftIR1.fb];
else when (STATE_JTAG_CAPTURE_IR & STATE_JTAG_ENABLED) then
    JtagShiftIR := INST_CODE_BYPASS;  " interim default
else
    JtagShiftIR := JtagShiftIR.fb;

"*******************************************************************************
"* Jtag Instruction Register

equations

JtagIR.clk = !Tck;
JtagIR.ap = JtagReset ;  " reset -> bypass
JtagIR.ar = 0;
```

```
when (STATE_JTAG_UPDATE_IR & (NEXT_INST_DOWNLOAD # NEXT_INST_PON_RESET) ) then
      JtagIR := JtagShiftIR.fb;
else when (STATE_JTAG_UPDATE_IR & (!(NEXT_INST_DOWNLOAD #
                                    NEXT_INST_PON_RESET) ) ) then
      JtagIR := INST_CODE_BYPASS ;
else
      JtagIR := JtagIR.fb;    " hold value

"*****************************************************************************
"* Download Shift / Bypass Register

equations

JtagShiftDR.clk = Tck;
JtagShiftDR.ar = JtagStateReset ;
JtagShiftDR.ap = 0;

when (JtagShiftDrState.fb & INST_IS_BYPASS #
     JtagShiftDrState.fb & INST_IS_DOWNLOAD & !JTAG_DOWNLOAD_SHIFT_REG_FULL) then
    [JtagShiftDR0..JtagShiftDR7] := [Tdi,JtagShiftDR0.fb,JtagShiftDR1.fb,
                                     JtagShiftDR2.fb,JtagShiftDR3.fb,
                                     JtagShiftDR4.fb,JtagShiftDR5.fb,
                                     JtagShiftDR6.fb];
else
    JtagShiftDR := JtagShiftDR.fb ;  " hold

"*****************************************************************************
"* Receive Full Flag

equations

JtagReceiveFull.clk = !Tck;

JtagReceiveFull.ar = JtagReceiveFullReset.com ;
JtagReceiveFull.ap = 0;

when (STATE_JTAG_EXIT1_DR & INST_IS_DOWNLOAD) then  " end of download byte
    JtagReceiveFull := RECEIVE_FULL;
else
```

```
    JtagReceiveFull := JtagReceiveFull.fb;          " maintain value

"*************************************************************************
"* Receive Full Flag Reset

SReadJtagDownloadData.clk = SYSCLK;
SReadJtagDownloadData.ar = 0;
SReadJtagDownloadData.ap = 0;

SReadJtagDownloadData := READ_JTAG_DOWNLOAD_DATA & !DVal_B;


JtagReceiveFullReset = ( SReadJtagDownloadData.fb #
                         !Trst_B #
                         !PORIn_B #
                         !JtagResetState.fb) ;

"*************************************************************************
"* TDO Selection

equations

Tdo.clk = !Tck;

Tdo.ar = 0;
Tdo.ap = JtagStateReset ;

when (STATE_JTAG_SHIFT_IR) then
  Tdo := JtagShiftIR2.fb;
else when (STATE_JTAG_SHIFT_DR & INST_IS_BYPASS) then
  Tdo := JtagShiftDR0.fb;
else when (STATE_JTAG_SHIFT_DR & INST_IS_DOWNLOAD) then
  Tdo := JtagReceiveFull.fb;

when (STATE_JTAG_ENABLED) then
  TdoOut = Tdo.fb;
else when (!STATE_JTAG_ENABLED) then
   TdoOut = Tdi;

TdoEnable = (STATE_JTAG_ENABLED & JtagTdoEnable.fb #
```

```
        !STATE_JTAG_ENABLED) ;

TdoOut.oe = TdoEnable.com;

"*****************************************************************************
"* Power On Reset Generation

equations

PonResetOut.oe = 1;

PonResetOut = INST_IS_PON_RESET ;

"*****************************************************************************
"* Auxiliary functions
"*****************************************************************************
equations

KeepPinsConnected = TEA_B & BCTL1 # KeepPinsConnected.com;
"
END
```

7.2.2 U10 - PCI Interrupt Controller Code

```
MODULE PCI_Interrupt_Controller

TITLE 'MPC8265 PCI motherboard ads PCI Interrupt Controller'

"*****************************************************************************
"* Device declaration.                                       *
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************
SYSCLK              PIN  5 ;

IntContCs_B         PIN  20 ;
DVal_B              PIN  29 ;
```

```
R_B_W           PIN  22 ; " BCTL0 signal

A7              PIN  27 ;
A8              PIN  33 ;
A27             PIN  34 ;
A28             PIN  35 ;
A29             PIN  36 ;

D0              PIN  37   istype 'com' ;
D1              PIN  38   istype 'com' ;
D2              PIN  39   istype 'com' ;
D3              PIN  40   istype 'com' ;
D4              PIN  12   istype 'com' ;
D5              PIN  11   istype 'com' ;
D6              PIN  10   istype 'com' ;
D7              PIN   9   istype 'com' ;
D8              PIN   3   istype 'com' ;
D9              PIN   2   istype 'com' ;
D10             PIN   1   istype 'com' ;
D11             PIN  48   istype 'com' ;

"*****************************************************************************
"* PCI Interrupt Pins.
"*****************************************************************************

PCI_IRQ_B       PIN  26  istype 'com,buffer' ; " PCI Interrupt to PQ2 (o.d.)

PCI_INTA_B      PIN  44 ;" PCI Interrupt from PCI card
PCI_INTB_B      PIN  45 ;" PCI Interrupt from PCI card
PCI_INTC_B      PIN  46 ;" PCI Interrupt from PCI card
PCI_INTD_B      PIN  47 ;" PCI Interrupt from PCI card

"*****************************************************************************
"* Reset Logic Pins.
"*****************************************************************************

PORIn_B         PIN  23 ; " power-on reset input
HardReset_B     PIN  25 ; " hard reset input
SoftReset_B     PIN  24 ; " soft reset input
```

```
"*****************************************************************************
"* Reset configuration support
"*****************************************************************************

TEA_B               PIN  61 ; "  Transfer Error Acknowledge.

"*****************************************************************************
"* Chassis Power-On Pins.
"*****************************************************************************

ChasisPowerIn_B PIN  15 ;   "Chassis Power Switch
PowerOn_B PIN  16  istype 'reg_t,invert' ;  " Power Supply Power-On

"*****************************************************************************
"* Global OE for ICT Pin.
"*****************************************************************************
OE PIN  14 ;  "global OE for ICT purposes

"*****************************************************************************
"* PCI Interrupt Register.
"*****************************************************************************
Slot0IntA NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt A
Slot0IntB NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt B
Slot0IntC NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt C
Slot0IntD NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt D
Slot1IntA NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt A
Slot1IntB NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt B
Slot1IntC NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt C
Slot1IntD NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt D
Slot2IntA NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt A
Slot2IntB NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt B
Slot2IntC NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt C
Slot2IntD NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt D

"*****************************************************************************
"* PCI Interrupt Mask Register.
"*****************************************************************************
Slot0IntAMask NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt A Mask
Slot0IntBMask NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt B Mask
Slot0IntCMask NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt C Mask
```

```
Slot0IntDMask NODE istype 'reg,buffer' ;  " PCI Slot 0 Interrupt D Mask
Slot1IntAMask NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt A Mask
Slot1IntBMask NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt B Mask
Slot1IntCMask NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt C Mask
Slot1IntDMask NODE istype 'reg,buffer' ;  " PCI Slot 1 Interrupt D Mask
Slot2IntAMask NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt A Mask
Slot2IntBMask NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt B Mask
Slot2IntCMask NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt C Mask
Slot2IntDMask NODE istype 'reg,buffer' ;  " PCI Slot 2 Interrupt D Mask

"*****************************************************************************
"* Data OE on Read.
"*****************************************************************************
DataOe NODE istype 'com' ;  " data bus output enable on read.

"*****************************************************************************
"* PCI Interrupt Request to PQ2.
"*****************************************************************************
PCI_Interrupt NODE istype 'com' ;  " generated Interrupt to PQ2

"*****************************************************************************
"* Chassis Power Switch Buffer.
"*****************************************************************************
Power_Buffer NODE istype 'reg,buffer' ;  " generated Interrupt to PQ2

"*****************************************************************************
"* Creating internal clock generator.
"*****************************************************************************
inv1 NODE istype 'com,keep' ;
inv2 NODE istype 'com,keep' ;
inv3 NODE istype 'com,keep' ;
inv4 NODE istype 'com,keep' ;
inv5 NODE istype 'com,keep' ;

counter0,
counter1,
counter2,
counter3,
counter4,
counter5,
```

```
counter6,
counter7,
countera0,
countera1,
countera2,
countera3,
countera4,
countera5,
countera6,
countera7       NODE istype 'reg,buffer' ;

"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*****************************************************************************
"* SIMULATION = 1 ;
"*****************************************************************************
"* Signal groups
"*****************************************************************************

IntReg     = [Slot0IntA,
      Slot0IntB,
      Slot0IntC,
      Slot0IntD,
       Slot1IntA,
       Slot1IntB,
      Slot1IntC,
       Slot1IntD,
      Slot2IntA,
      Slot2IntB,
       Slot2IntC,
       Slot2IntD] ;

IntMaskReg = [Slot0IntAMask,
      Slot0IntBMask,
      Slot0IntCMask,
      Slot0IntDMask,
       Slot1IntAMask,
       Slot1IntBMask,
      Slot1IntCMask,
```

```
      Slot1IntDMask,
     Slot2IntAMask,
     Slot2IntBMask,
      Slot2IntCMask,
      Slot2IntDMask] ;

Add       = [A27..A29] ;
Data      = [D0..D11] ;
Reset     = [HardReset_B,SoftReset_B] ;

counter   = [counter7,counter6,counter5,counter4,
           counter3,counter2,counter1,counter0] ;
countera  = [countera7,countera6,countera5,countera4,
            countera3,countera2,countera1,countera0] ;

"*****************************************************************************
"* Power On Reset definitions
"*****************************************************************************
PON_RESET_ACTIVE = 0 ;

PON_RESET = (PORIn_B == PON_RESET_ACTIVE) ;

"*****************************************************************************
"* Register Access definitions
"*****************************************************************************
IntReg_ADD    = 0 ;
IntMaskReg_ADD = 1 ;

"VGR_WRITE_IntReg    = (!IntContCs_B & !DVal_B & R_B_W & !A27 & !A28 & !A29) ;
VGR_WRITE_IntMaskReg = (!IntContCs_B & !DVal_B & R_B_W & !A27 & !A28 & A29) ;

VGR_READ_IntReg    = (!IntContCs_B & !R_B_W & !A27 & !A28 & !A29) ;
VGR_READ_IntMaskReg = (!IntContCs_B & !R_B_W & !A27 & !A28 & A29) ;

"*****************************************************************************
"* Interrupt Request Definitions.
"*****************************************************************************
IrqOe = (Slot0IntA.fb #
 Slot0IntB.fb #
 Slot0IntC.fb #
```

```
Slot0IntD.fb #
Slot1IntA.fb #
Slot1IntB.fb #
Slot1IntC.fb #
Slot1IntD.fb #
Slot2IntA.fb #
Slot2IntB.fb #
Slot2IntC.fb #
Slot2IntD.fb) ;

"*****************************************************************************
"*****************************************************************************
"* PCI Interrupt Register definitions.
"*****************************************************************************
"*****************************************************************************
Slot0IntA_Active = 1 ; " PCI Slot 0 Interrupt A asserted
Slot0IntB_Active = 1 ; " PCI Slot 0 Interrupt B asserted
Slot0IntC_Active = 1 ; " PCI Slot 0 Interrupt C asserted
Slot0IntD_Active = 1 ; " PCI Slot 0 Interrupt D asserted
Slot1IntA_Active = 1 ; " PCI Slot 1 Interrupt A asserted
Slot1IntB_Active = 1 ; " PCI Slot 1 Interrupt B asserted
Slot1IntC_Active = 1 ; " PCI Slot 1 Interrupt C asserted
Slot1IntD_Active = 1 ; " PCI Slot 1 Interrupt D asserted
Slot2IntA_Active = 1 ; " PCI Slot 2 Interrupt A asserted
Slot2IntB_Active = 1 ; " PCI Slot 2 Interrupt B asserted
Slot2IntC_Active = 1 ; " PCI Slot 2 Interrupt C asserted
Slot2IntD_Active = 1 ; " PCI Slot 2 Interrupt D asserted

"************************************************
"******* Power On Defaults Assignments *********
"************************************************
Slot0IntA_PON_DEFAULT = !Slot0IntA_Active ;
Slot0IntB_PON_DEFAULT = !Slot0IntB_Active ;
Slot0IntC_PON_DEFAULT = !Slot0IntC_Active ;
Slot0IntD_PON_DEFAULT = !Slot0IntD_Active ;
Slot1IntA_PON_DEFAULT = !Slot1IntA_Active ;
Slot1IntB_PON_DEFAULT = !Slot1IntB_Active ;
Slot1IntC_PON_DEFAULT = !Slot1IntC_Active ;
Slot1IntD_PON_DEFAULT = !Slot1IntD_Active ;
Slot2IntA_PON_DEFAULT = !Slot2IntA_Active ;
```

```
Slot2IntB_PON_DEFAULT = !Slot2IntB_Active ;
Slot2IntC_PON_DEFAULT = !Slot2IntC_Active ;
Slot2IntD_PON_DEFAULT = !Slot2IntD_Active ;

"*****************************************
"******* Data Bits Assignments **************
"*****************************************
Slot0IntA_DATA_BIT = [D0] ;
Slot0IntB_DATA_BIT = [D1] ;
Slot0IntC_DATA_BIT = [D2] ;
Slot0IntD_DATA_BIT = [D3] ;
Slot1IntA_DATA_BIT = [D4] ;
Slot1IntB_DATA_BIT = [D5] ;
Slot1IntC_DATA_BIT = [D6] ;
Slot1IntD_DATA_BIT = [D7] ;
Slot2IntA_DATA_BIT = [D8] ;
Slot2IntB_DATA_BIT = [D9] ;
Slot2IntC_DATA_BIT = [D10] ;
Slot2IntD_DATA_BIT = [D11] ;

"*****************************************************************************
"*****************************************************************************
"* PCI Interrupt Mask Register definitions.
"*****************************************************************************
"*****************************************************************************
Slot0IntAMask_Active = 1 ; " PCI Slot 0 Interrupt A Masked
Slot0IntBMask_Active = 1 ; " PCI Slot 0 Interrupt B Masked
Slot0IntCMask_Active = 1 ; " PCI Slot 0 Interrupt C Masked
Slot0IntDMask_Active = 1 ; " PCI Slot 0 Interrupt D Masked
Slot1IntAMask_Active = 1 ; " PCI Slot 1 Interrupt A Masked
Slot1IntBMask_Active = 1 ; " PCI Slot 1 Interrupt B Masked
Slot1IntCMask_Active = 1 ; " PCI Slot 1 Interrupt C Masked
Slot1IntDMask_Active = 1 ; " PCI Slot 1 Interrupt D Masked
Slot2IntAMask_Active = 1 ; " PCI Slot 2 Interrupt A Masked
Slot2IntBMask_Active = 1 ; " PCI Slot 2 Interrupt B Masked
Slot2IntCMask_Active = 1 ; " PCI Slot 2 Interrupt C Masked
Slot2IntDMask_Active = 1 ; " PCI Slot 2 Interrupt D Masked

"*********************************************
"******* Power On Defaults Assignments *********
```

```
"***********************************************
Slot0IntAMask_PON_DEFAULT = Slot0IntAMask_Active ;
Slot0IntBMask_PON_DEFAULT = Slot0IntBMask_Active ;
Slot0IntCMask_PON_DEFAULT = Slot0IntCMask_Active ;
Slot0IntDMask_PON_DEFAULT = Slot0IntDMask_Active ;
Slot1IntAMask_PON_DEFAULT = Slot1IntAMask_Active ;
Slot1IntBMask_PON_DEFAULT = Slot1IntBMask_Active ;
Slot1IntCMask_PON_DEFAULT = Slot1IntCMask_Active ;
Slot1IntDMask_PON_DEFAULT = Slot1IntDMask_Active ;
Slot2IntAMask_PON_DEFAULT = Slot2IntAMask_Active ;
Slot2IntBMask_PON_DEFAULT = Slot2IntBMask_Active ;
Slot2IntCMask_PON_DEFAULT = Slot2IntCMask_Active ;
Slot2IntDMask_PON_DEFAULT = Slot2IntDMask_Active ;

"*****************************************
"******* Data Bits Assignments **********
"*****************************************
Slot0IntAMask_DATA_BIT = [D0];
Slot0IntBMask_DATA_BIT = [D1];
Slot0IntCMask_DATA_BIT = [D2];
Slot0IntDMask_DATA_BIT = [D3];
Slot1IntAMask_DATA_BIT = [D4];
Slot1IntBMask_DATA_BIT = [D5];
Slot1IntCMask_DATA_BIT = [D6];
Slot1IntDMask_DATA_BIT = [D7];
Slot2IntAMask_DATA_BIT = [D8];
Slot2IntBMask_DATA_BIT = [D9];
Slot2IntCMask_DATA_BIT = [D10];
Slot2IntDMask_DATA_BIT = [D11];

"*****************************************************************************
"* Reset Declarations.
"*****************************************************************************
HARD_RESET_ACTIVE = 0 ;
Hard_Reset = (HardReset_B == HARD_RESET_ACTIVE) ;

SOFT_RESET_ACTIVE = 0 ;
Soft_Reset = (SoftReset_B == SOFT_RESET_ACTIVE) ;

"*****************************************************************************
```

```
"* ATX Power Declarations.
"*******************************************************************************
PowerOn  = 0 ;
PowerOff = 1 ;

"*******************************************************************************
"* Equations, state diagrams.                                  *
"*******************************************************************************
"*******************************************************************************

equations

"*******************************************************************************
"*******************************************************************************
"* PCI Interrupt Register
"*******************************************************************************
"*******************************************************************************
equations

IntReg.clk = SYSCLK ;
IntReg.ar = PON_RESET ;
IntReg.ap = 0 ;

"*******************************************************************************
state_diagram Slot0IntA
state Slot0IntA_Active:
if ( (Hard_Reset & (Slot0IntA_PON_DEFAULT == !Slot0IntA_Active)) #
    (!Hard_Reset & PCI_INTA_B) )
then
!Slot0IntA_Active
else
Slot0IntA_Active ;
state !Slot0IntA_Active:
if ( (Hard_Reset & (Slot0IntA_PON_DEFAULT == Slot0IntA_Active)) #
    (!Hard_Reset & !PCI_INTA_B) )
then
Slot0IntA_Active
else
!Slot0IntA_Active ;
"*******************************************************************************
```

```
state_diagram Slot0IntB
state Slot0IntB_Active:
if ( (Hard_Reset & (Slot0IntB_PON_DEFAULT == !Slot0IntB_Active)) #
    (!Hard_Reset & PCI_INTB_B) )
then
!Slot0IntB_Active
else
Slot0IntB_Active ;
state !Slot0IntB_Active:
if ( (Hard_Reset & (Slot0IntB_PON_DEFAULT == Slot0IntB_Active)) #
    (!Hard_Reset & !PCI_INTB_B) )
then
Slot0IntB_Active
else
!Slot0IntB_Active ;
"*****************************************************************************
state_diagram Slot0IntC
state Slot0IntC_Active:
if ( (Hard_Reset & (Slot0IntC_PON_DEFAULT == !Slot0IntC_Active)) #
    (!Hard_Reset & PCI_INTC_B) )
then
!Slot0IntC_Active
else
Slot0IntC_Active ;
state !Slot0IntC_Active:
if ( (Hard_Reset & (Slot0IntC_PON_DEFAULT == Slot0IntC_Active)) #
    (!Hard_Reset & !PCI_INTC_B) )
then
Slot0IntC_Active
else
!Slot0IntC_Active ;
"*****************************************************************************
state_diagram Slot0IntD
state Slot0IntD_Active:
if ( (Hard_Reset & (Slot0IntD_PON_DEFAULT == !Slot0IntD_Active)) #
    (!Hard_Reset & PCI_INTD_B) )
then
!Slot0IntD_Active
else
Slot0IntD_Active ;
```

```
state !Slot0IntD_Active:
if ( (Hard_Reset & (Slot0IntD_PON_DEFAULT == Slot0IntD_Active)) #
    (!Hard_Reset & !PCI_INTD_B) )
then
Slot0IntD_Active
else
!Slot0IntD_Active ;
"*****************************************************************************
state_diagram Slot1IntA
state Slot1IntA_Active:
if ( (Hard_Reset & (Slot1IntA_PON_DEFAULT == !Slot1IntA_Active)) #
    (!Hard_Reset & PCI_INTD_B) )
then
!Slot1IntA_Active
else
Slot1IntA_Active ;
state !Slot1IntA_Active:
if ( (Hard_Reset & (Slot1IntA_PON_DEFAULT == Slot1IntA_Active)) #
    (!Hard_Reset & !PCI_INTD_B) )
then
Slot1IntA_Active
else
!Slot1IntA_Active ;
"*****************************************************************************
state_diagram Slot1IntB
state Slot1IntB_Active:
if ( (Hard_Reset & (Slot1IntB_PON_DEFAULT == !Slot1IntB_Active)) #
    (!Hard_Reset & PCI_INTA_B) )
then
!Slot1IntB_Active
else
Slot1IntB_Active ;
state !Slot1IntB_Active:
if ( (Hard_Reset & (Slot1IntB_PON_DEFAULT == Slot1IntB_Active)) #
    (!Hard_Reset & !PCI_INTA_B) )
then
Slot1IntB_Active
else
!Slot1IntB_Active ;
"*****************************************************************************
```

```
state_diagram Slot1IntC
state Slot1IntC_Active:
if ( (Hard_Reset & (Slot1IntC_PON_DEFAULT == !Slot1IntC_Active)) #
    (!Hard_Reset & PCI_INTB_B) )
then
!Slot1IntC_Active
else
Slot1IntC_Active ;
state !Slot1IntC_Active:
if ( (Hard_Reset & (Slot1IntC_PON_DEFAULT == Slot1IntC_Active)) #
    (!Hard_Reset & !PCI_INTB_B) )
then
Slot1IntC_Active
else
!Slot1IntC_Active ;
"*****************************************************************************
state_diagram Slot1IntD
state Slot1IntD_Active:
if ( (Hard_Reset & (Slot1IntD_PON_DEFAULT == !Slot1IntD_Active)) #
    (!Hard_Reset & PCI_INTC_B) )
then
!Slot1IntD_Active
else
Slot1IntD_Active ;
state !Slot1IntD_Active:
if ( (Hard_Reset & (Slot1IntD_PON_DEFAULT == Slot1IntD_Active)) #
    (!Hard_Reset & !PCI_INTC_B) )
then
Slot1IntD_Active
else
!Slot1IntD_Active ;
"*****************************************************************************
state_diagram Slot2IntA
state Slot2IntA_Active:
if ( (Hard_Reset & (Slot2IntA_PON_DEFAULT == !Slot2IntA_Active)) #
    (!Hard_Reset & PCI_INTC_B) )
then
!Slot2IntA_Active
else
Slot2IntA_Active ;
```

```
state !Slot2IntA_Active:
if ( (Hard_Reset & (Slot2IntA_PON_DEFAULT == Slot2IntA_Active)) #
    (!Hard_Reset & !PCI_INTC_B) )
then
Slot2IntA_Active
else
!Slot2IntA_Active ;
"*****************************************************************************
state_diagram Slot2IntB
state Slot2IntB_Active:
if ( (Hard_Reset & (Slot2IntB_PON_DEFAULT == !Slot2IntB_Active)) #
    (!Hard_Reset & PCI_INTD_B) )
then
!Slot2IntB_Active
else
Slot2IntB_Active ;
state !Slot2IntB_Active:
if ( (Hard_Reset & (Slot2IntB_PON_DEFAULT == Slot2IntB_Active)) #
    (!Hard_Reset & !PCI_INTD_B) )
then
Slot2IntB_Active
else
!Slot2IntB_Active ;
"*****************************************************************************
state_diagram Slot2IntC
state Slot2IntC_Active:
if ( (Hard_Reset & (Slot2IntC_PON_DEFAULT == !Slot2IntC_Active)) #
    (!Hard_Reset & PCI_INTA_B) )
then
!Slot2IntC_Active
else
Slot2IntC_Active ;
state !Slot2IntC_Active:
if ( (Hard_Reset & (Slot2IntC_PON_DEFAULT == Slot2IntC_Active)) #
    (!Hard_Reset & !PCI_INTA_B) )
then
Slot2IntC_Active
else
!Slot2IntC_Active ;
"*****************************************************************************
```

```
state_diagram Slot2IntD
state Slot2IntD_Active:
if ( (Hard_Reset & (Slot2IntD_PON_DEFAULT == !Slot2IntD_Active)) #
    (!Hard_Reset & PCI_INTB_B) )
then
!Slot2IntD_Active
else
Slot2IntD_Active ;
state !Slot2IntD_Active:
if ( (Hard_Reset & (Slot2IntD_PON_DEFAULT == Slot2IntD_Active)) #
    (!Hard_Reset & !PCI_INTB_B) )
then
Slot2IntD_Active
else
!Slot2IntD_Active ;
"*******************************************************************************
"*******************************************************************************
"* PCI Interrupt Mask Register
"*******************************************************************************
"*******************************************************************************
equations

IntMaskReg.clk = SYSCLK ;
IntMaskReg.ar = 0 ;
IntMaskReg.ap = PON_RESET ;

"*******************************************************************************
state_diagram Slot0IntAMask
state Slot0IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntAMask_DATA_BIT.pin == !Slot0IntAMask_Active) &
     (!Hard_Reset # (Slot0IntAMask_PON_DEFAULT == !Slot0IntAMask_Active)) #
   (Hard_Reset & (Slot0IntAMask_PON_DEFAULT == !Slot0IntAMask_Active)) )
then
!Slot0IntAMask_Active
else
Slot0IntAMask_Active ;
state !Slot0IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot0IntAMask_DATA_BIT.pin == Slot0IntAMask_Active) &
```

```
    (!Hard_Reset # (Slot0IntAMask_PON_DEFAULT == Slot0IntAMask_Active)) #
   (Hard_Reset & (Slot0IntAMask_PON_DEFAULT == Slot0IntAMask_Active)) )
then
Slot0IntAMask_Active
else
!Slot0IntAMask_Active ;
"*****************************************************************************
state_diagram Slot0IntBMask
state Slot0IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntBMask_DATA_BIT.pin == !Slot0IntBMask_Active) &
    (!Hard_Reset # (Slot0IntBMask_PON_DEFAULT == !Slot0IntBMask_Active)) #
   (Hard_Reset & (Slot0IntBMask_PON_DEFAULT == !Slot0IntBMask_Active)) )
then
!Slot0IntBMask_Active
else
Slot0IntBMask_Active ;
state !Slot0IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntBMask_DATA_BIT.pin == Slot0IntBMask_Active) &
    (!Hard_Reset # (Slot0IntBMask_PON_DEFAULT == Slot0IntBMask_Active)) #
   (Hard_Reset & (Slot0IntBMask_PON_DEFAULT == Slot0IntBMask_Active)) )
then
Slot0IntBMask_Active
else
!Slot0IntBMask_Active ;
"*****************************************************************************
state_diagram Slot0IntCMask
state Slot0IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntCMask_DATA_BIT.pin == !Slot0IntCMask_Active) &
    (!Hard_Reset # (Slot0IntCMask_PON_DEFAULT == !Slot0IntCMask_Active)) #
   (Hard_Reset & (Slot0IntCMask_PON_DEFAULT == !Slot0IntCMask_Active)) )
then
!Slot0IntCMask_Active
else
Slot0IntCMask_Active ;
state !Slot0IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntCMask_DATA_BIT.pin == Slot0IntCMask_Active) &
```

```
    (!Hard_Reset # (Slot0IntCMask_PON_DEFAULT == Slot0IntCMask_Active)) #
  (Hard_Reset & (Slot0IntCMask_PON_DEFAULT == Slot0IntCMask_Active)) )
then
Slot0IntCMask_Active
else
!Slot0IntCMask_Active ;
"*****************************************************************************
state_diagram Slot0IntDMask
state Slot0IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntDMask_DATA_BIT.pin == !Slot0IntDMask_Active) &
    (!Hard_Reset # (Slot0IntDMask_PON_DEFAULT == !Slot0IntDMask_Active)) #
  (Hard_Reset & (Slot0IntDMask_PON_DEFAULT == !Slot0IntDMask_Active)) )
then
!Slot0IntDMask_Active
else
Slot0IntDMask_Active ;
state !Slot0IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot0IntDMask_DATA_BIT.pin == Slot0IntDMask_Active) &
    (!Hard_Reset # (Slot0IntDMask_PON_DEFAULT == Slot0IntDMask_Active)) #
  (Hard_Reset & (Slot0IntDMask_PON_DEFAULT == Slot0IntDMask_Active)) )
then
Slot0IntDMask_Active
else
!Slot0IntDMask_Active ;
"*****************************************************************************
state_diagram Slot1IntAMask
state Slot1IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot1IntAMask_DATA_BIT.pin == !Slot1IntAMask_Active) &
    (!Hard_Reset # (Slot1IntAMask_PON_DEFAULT == !Slot1IntAMask_Active)) #
  (Hard_Reset & (Slot1IntAMask_PON_DEFAULT == !Slot1IntAMask_Active)) )
then
!Slot1IntAMask_Active
else
Slot1IntAMask_Active ;
state !Slot1IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot1IntAMask_DATA_BIT.pin == Slot1IntAMask_Active) &
```

```
    (!Hard_Reset # (Slot1IntAMask_PON_DEFAULT == Slot1IntAMask_Active)) #
   (Hard_Reset & (Slot1IntAMask_PON_DEFAULT == Slot1IntAMask_Active)) )
then
Slot1IntAMask_Active
else
!Slot1IntAMask_Active ;
"*****************************************************************************
state_diagram Slot1IntBMask
state Slot1IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntBMask_DATA_BIT.pin == !Slot1IntBMask_Active) &
     (!Hard_Reset # (Slot1IntBMask_PON_DEFAULT == !Slot1IntBMask_Active)) #
   (Hard_Reset & (Slot1IntBMask_PON_DEFAULT == !Slot1IntBMask_Active)) )
then
!Slot1IntBMask_Active
else
Slot1IntBMask_Active ;
state !Slot1IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntBMask_DATA_BIT.pin == Slot1IntBMask_Active) &
     (!Hard_Reset # (Slot1IntBMask_PON_DEFAULT == Slot1IntBMask_Active)) #
   (Hard_Reset & (Slot1IntBMask_PON_DEFAULT == Slot1IntBMask_Active)) )
then
Slot1IntBMask_Active
else
!Slot1IntBMask_Active ;
"*****************************************************************************
state_diagram Slot1IntCMask
state Slot1IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntCMask_DATA_BIT.pin == !Slot1IntCMask_Active) &
     (!Hard_Reset # (Slot1IntCMask_PON_DEFAULT == !Slot1IntCMask_Active)) #
   (Hard_Reset & (Slot1IntCMask_PON_DEFAULT == !Slot1IntCMask_Active)) )
then
!Slot1IntCMask_Active
else
Slot1IntCMask_Active ;
state !Slot1IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot1IntCMask_DATA_BIT.pin == Slot1IntCMask_Active) &
```

```
    (!Hard_Reset # (Slot1IntCMask_PON_DEFAULT == Slot1IntCMask_Active)) #
  (Hard_Reset & (Slot1IntCMask_PON_DEFAULT == Slot1IntCMask_Active)) )
then
Slot1IntCMask_Active
else
!Slot1IntCMask_Active ;
"*****************************************************************************
state_diagram Slot1IntDMask
state Slot1IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot1IntDMask_DATA_BIT.pin == !Slot1IntDMask_Active) &
    (!Hard_Reset # (Slot1IntDMask_PON_DEFAULT == !Slot1IntDMask_Active)) #
  (Hard_Reset & (Slot1IntDMask_PON_DEFAULT == !Slot1IntDMask_Active)) )
then
!Slot1IntDMask_Active
else
Slot1IntDMask_Active ;
state !Slot1IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot1IntDMask_DATA_BIT.pin == Slot1IntDMask_Active) &
    (!Hard_Reset # (Slot1IntDMask_PON_DEFAULT == Slot1IntDMask_Active)) #
  (Hard_Reset & (Slot1IntDMask_PON_DEFAULT == Slot1IntDMask_Active)) )
then
Slot1IntDMask_Active
else
!Slot1IntDMask_Active ;
"*****************************************************************************
state_diagram Slot2IntAMask
state Slot2IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot2IntAMask_DATA_BIT.pin == !Slot2IntAMask_Active) &
    (!Hard_Reset # (Slot2IntAMask_PON_DEFAULT == !Slot2IntAMask_Active)) #
  (Hard_Reset & (Slot2IntAMask_PON_DEFAULT == !Slot2IntAMask_Active)) )
then
!Slot2IntAMask_Active
else
Slot2IntAMask_Active ;
state !Slot2IntAMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot2IntAMask_DATA_BIT.pin == Slot2IntAMask_Active) &
```

```
    (!Hard_Reset # (Slot2IntAMask_PON_DEFAULT == Slot2IntAMask_Active)) #
   (Hard_Reset & (Slot2IntAMask_PON_DEFAULT == Slot2IntAMask_Active)) )
then
Slot2IntAMask_Active
else
!Slot2IntAMask_Active ;
"*****************************************************************************
state_diagram Slot2IntBMask
state Slot2IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntBMask_DATA_BIT.pin == !Slot2IntBMask_Active) &
     (!Hard_Reset # (Slot2IntBMask_PON_DEFAULT == !Slot2IntBMask_Active)) #
   (Hard_Reset & (Slot2IntBMask_PON_DEFAULT == !Slot2IntBMask_Active)) )
then
!Slot2IntBMask_Active
else
Slot2IntBMask_Active ;
state !Slot2IntBMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntBMask_DATA_BIT.pin == Slot2IntBMask_Active) &
     (!Hard_Reset # (Slot2IntBMask_PON_DEFAULT == Slot2IntBMask_Active)) #
   (Hard_Reset & (Slot2IntBMask_PON_DEFAULT == Slot2IntBMask_Active)) )
then
Slot2IntBMask_Active
else
!Slot2IntBMask_Active ;
"*****************************************************************************
state_diagram Slot2IntCMask
state Slot2IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntCMask_DATA_BIT.pin == !Slot2IntCMask_Active) &
     (!Hard_Reset # (Slot2IntCMask_PON_DEFAULT == !Slot2IntCMask_Active)) #
   (Hard_Reset & (Slot2IntCMask_PON_DEFAULT == !Slot2IntCMask_Active)) )
then
!Slot2IntCMask_Active
else
Slot2IntCMask_Active ;
state !Slot2IntCMask_Active:
if (VGR_WRITE_IntMaskReg &
     (Slot2IntCMask_DATA_BIT.pin == Slot2IntCMask_Active) &
```

```
    (!Hard_Reset # (Slot2IntCMask_PON_DEFAULT == Slot2IntCMask_Active)) #
  (Hard_Reset & (Slot2IntCMask_PON_DEFAULT == Slot2IntCMask_Active)) )
then
Slot2IntCMask_Active
else
!Slot2IntCMask_Active ;
"*****************************************************************************
state_diagram Slot2IntDMask
state Slot2IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot2IntDMask_DATA_BIT.pin == !Slot2IntDMask_Active) &
    (!Hard_Reset # (Slot2IntDMask_PON_DEFAULT == !Slot2IntDMask_Active)) #
  (Hard_Reset & (Slot2IntDMask_PON_DEFAULT == !Slot2IntDMask_Active)) )
then
!Slot2IntDMask_Active
else
Slot2IntDMask_Active ;
state !Slot2IntDMask_Active:
if (VGR_WRITE_IntMaskReg &
    (Slot2IntDMask_DATA_BIT.pin == Slot2IntDMask_Active) &
    (!Hard_Reset # (Slot2IntDMask_PON_DEFAULT == Slot2IntDMask_Active)) #
  (Hard_Reset & (Slot2IntDMask_PON_DEFAULT == Slot2IntDMask_Active)) )
then
Slot2IntDMask_Active
else
!Slot2IntDMask_Active ;

"*****************************************************************************
"*****************************************************************************
"* Read Registers.
"* All registers have read capabilty.
"*****************************************************************************
"*****************************************************************************
equations

DataOe = !PON_RESET & !Hard_Reset & (VGR_READ_IntReg # VGR_READ_IntMaskReg) & OE;

Data.oe = DataOe ;

when (VGR_READ_IntReg)
```

```
then
Data = IntReg.fb ;
else
when (VGR_READ_IntMaskReg)
then
Data = IntMaskReg.fb ;

"*******************************************************************************
"*******************************************************************************
"* Generating Interrupt Request to the PQ2.
"*******************************************************************************
"*******************************************************************************
equations

PCI_Interrupt = (Slot0IntA.fb & !Slot0IntAMask.fb # " PCI Interrupt assertion
 Slot0IntB.fb & !Slot0IntBMask.fb # " depends on the masking of
 Slot0IntC.fb & !Slot0IntCMask.fb # " each PCI interrupt.
 Slot0IntD.fb & !Slot0IntDMask.fb #
 Slot1IntA.fb & !Slot1IntAMask.fb #
 Slot1IntB.fb & !Slot1IntBMask.fb #
 Slot1IntC.fb & !Slot1IntCMask.fb #
 Slot1IntD.fb & !Slot1IntDMask.fb #
 Slot2IntA.fb & !Slot2IntAMask.fb #
 Slot2IntB.fb & !Slot2IntBMask.fb #
 Slot2IntC.fb & !Slot2IntCMask.fb #
 Slot2IntD.fb & !Slot2IntDMask.fb) ;

PCI_IRQ_B.oe = IrqOe & OE ;  " Open-Drain output
!PCI_IRQ_B = PCI_Interrupt ; " Interrupt Request shows after OE

"*******************************************************************************
"*******************************************************************************
"* Generating PowerOn signal to the ATX Power Supply.
"*******************************************************************************
"*******************************************************************************
equations

inv1 = !inv5.com ;" generating internal clock oscilator
inv2 = !inv1.com ;
inv3 = !inv2.com ;
```

```
inv4 = !inv3.com ;
inv5 = !inv4.com ;

counter.ar = 0 ;
counter.ap = 0 ;
counter.clk = !inv5.com ;

when ( counter.fb == 255 ) then counter := 0 else counter := counter + 1 ;

countera.ar = 0 ;
countera.ap = 0 ;
countera.clk = ( counter.fb == 0 ) ;

when ( countera.fb == 255 ) then countera := 0 else countera := countera + 1 ;

Power_Buffer.ar = 0 ;
Power_Buffer.ap = 0 ;
Power_Buffer.clk = ( countera.fb == 0 ) ;
Power_Buffer := !ChasisPowerIn_B ;

PowerOn_B.oe  = H ;
PowerOn_B.ar  = 0 ;
PowerOn_B.ap  = 0 ;
PowerOn_B.clk = Power_Buffer.fb ;    " T Flip Flop toggles everytime the chassis
PowerOn_B.t   = H ;                  " power pushbutton is pressed.

"*****************************************************************************
@ifdef SIMULATION {

}

END
```

7.2.3 U23 - SDRAM MUX/LATCH High

```
MODULE vmuxhig6

TITLE  'Voyager Sdram 2nd Latch - Mux'
"*****************************************************************************
"* This file contains the 2'nd part of Address latch & mux for the Voyager ads
"* sdram.
```

```
"* The mux is required only with L2 Cache installed on board. Otherwise
"* it is not assembled and SdramA(13:8) are connected to voyager's proper
"* address lines.
"*****************************************************************************
"* In this file (2):
"* - pinout is changed for MAC211SP-7VC
"*****************************************************************************
"* In this file (3) 07/13/98:
"* - Added SdramA9 which is latched A10, saving another LCX373
"*****************************************************************************
"* In this file (4) Rev PILOT 02/18/99:
"* - Added support for PBI - Page Based Interleaving feature added with
"*   PQII rev A. To provide that support the following were made:
"*   - Added DimmSize input, which provides SDRAM dimm size information.
"*     Only 16Meg (reset default) and 64Meg DIMMs are supported.
"*     This signal originates in BCSR.
"*   - Added PBI indication. Since PBI's actual seeting is done within the
"*     PQII, it is the system programmer responsibility to set the correct
"*     value for this signal in BCSR, to ensure proper operation.
"*   - To make room for the above, SdramA(9:8) are moved to another
"*     device, along with A20.
"*   - The output of this mux is now qualified with the DimmSize and PBI
"*     information to provide the correct address lines to the sdram dimm.
"*****************************************************************************
"* In this file (5) Rev PILOT 03/21/99:
"* - Pinout changed to M4-64/32-7VC (TQFP48 package).
"*****************************************************************************
"* In this file (6) Rev Pilot 01/20/00:
"* - Changed polarity of SdramA8,9,11 since there is an inversion
"*   from the .Q
"*****************************************************************************
"* Device declaration.                                    *
"*****************************************************************************


"*****************************************************************************
"* Pins declaration.                                      *
"*****************************************************************************
"* Control pins
"*****************************************************************************
Ale_B               PIN     5;
```

```
R_C_B              PIN     1;
DimmSize           PIN     2;
PBI                PIN     3;

"*****************************************************************************
"* Address Input lines
"*****************************************************************************
" row
A6                 PIN     37;
A7                 PIN     38;
A8                 PIN     39;
A9                 PIN     40;
A10                PIN     44;
A11                PIN     45;
" col
A20                PIN     46;

"*****************************************************************************
"* Address Output lines
"*****************************************************************************
SdramA8            PIN     9 istype 'com' ;
SdramA9            PIN     24 istype 'com' ;
SdramA11           PIN     23 istype 'com' ;
"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************

"*****************************************************************************
"*****************************************************************************
"* Latched Address lines
"*****************************************************************************
LA6          NODE istype 'reg_D,buffer' ;
LA7          NODE istype 'reg_D,buffer' ;
LA8          NODE istype 'reg_D,buffer' ;
LA9          NODE istype 'reg_D,buffer' ;
LA10         NODE istype 'reg_D,buffer' ;
LA11         NODE istype 'reg_D,buffer' ;

" col
```

```
LA20            NODE istype 'reg_D,buffer' ;

"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*****************************************************************************
"* SIMULATION = 1 ;
"*****************************************************************************
"* Signal groups
"*****************************************************************************
Add = [A6..A11,A20] ;

LAdd = [LA6..LA11,LA20] ;

RowAddNormal = [LA8,LA10,LA11] ;
RowAddPBI_16M = [LA7,LA9,LA10] ; " LA7 not really required
RowAddPBI_64M = [LA6,LA8,LA9] ;

ColAddNormal = [LA8,LA10,LA20] ;
ColAddPBI_16M = [LA7,LA9,LA20] ; " LA7 not really required
ColAddPBI_64M = [LA6,LA8,LA20] ;

SdramAdd = [SdramA11,SdramA9,SdramA8] ;

ROW = (R_C_B == 1) ;
COL = !ROW ;

SIZE_16M = 0;
SIZE_64M = 1;

SDRAM_16M = (DimmSize == SIZE_16M);
SDRAM_64M = !SDRAM_16M;

SDRAM_NORMAL_MODE = (PBI == 0);
SDRAM_PBI_MODE = !SDRAM_NORMAL_MODE;

MuxCont = [PBI,DimmSize];

"*****************************************************************************
"* Equations, state diagrams.                                  *
```

```
"*******************************************************************************
"*******************************************************************************
"*******************************************************************************
"* Input Latch
"*******************************************************************************
"*******************************************************************************
equations

LAdd.le = Ale_B ;

LAdd.d = Add ;    " latching the address

"*******************************************************************************
"*******************************************************************************
"* Output Mux
"*******************************************************************************
"*******************************************************************************
equations

SdramAdd.oe = ^h7 ;   "always enabled

when (ROW & SDRAM_NORMAL_MODE) then
    SdramAdd = !RowAddNormal.q
else when (ROW & SDRAM_PBI_MODE & SDRAM_16M) then
    SdramAdd = !RowAddPBI_16M.q
else when (ROW & SDRAM_PBI_MODE & SDRAM_64M) then
    SdramAdd = !RowAddPBI_64M.q
else when (COL & SDRAM_NORMAL_MODE) then
    SdramAdd = !ColAddNormal.q
else when (COL & SDRAM_PBI_MODE & SDRAM_16M) then
    SdramAdd = !ColAddPBI_16M.q
else when (COL & SDRAM_PBI_MODE & SDRAM_64M) then
    SdramAdd = !ColAddPBI_64M.q;

@ifdef SIMULATION {
}
END
```

7.2.4 U22 - SDRAM MUX/LATCH Low

```
MODULE vmuxlow6
```

```
TITLE  'Voyager Sdram 1st Latch - Mux'
"*****************************************************************************
"* This file contains the 1'st part of Address latch & mux for the Voyager ads
"* sdram.
"* The mux is required only with L2 Cache installed on board. Otherwise
"* it is not assembled and A(21:28) are shortened to SdramA(7:0).
"*****************************************************************************
"* In this file (2):
"* - pinout is changed for MAC211SP-7VC
"*****************************************************************************
"* In this file (3) 07/13/98:
"* - Added SdramA9 which is latched A10, saving another LCX373
"*****************************************************************************
"* In this file (4) Rev PILOT 02/18/99:
"* - Added support for PBI - Page Based Interleaving feature added with
"*   PQII rev A. To provide that support the following were made:
"*   - Added DimmSize input, which provides SDRAM dimm size information.
"*     Only 16Meg (reset default) and 64Meg DIMMs are supported.
"*     This signal originates in BCSR.
"*   - Added PBI indication. Since PBI's actual seeting is done within the
"*     PQII, it is the system programmer responsibility to set the correct
"*     value for this signal in BCSR, to ensure proper operation.
"*   - To make room for the above, SdramA(9:8) are moved to another
"*     device, along with A20.
"*   - The output of this mux is now qualified with the DimmSize and PBI
"*     information to provide the correct address lines to the sdram dimm.
"*****************************************************************************
"* In this file (5) Rev PILOT (03/21/99):
"* - Pinout is changed to M4-64/32-7VC (48TQFP package).
"*****************************************************************************
"* In this file (6) rev PILOT (01/20/00):
"* - Inverted Polarity for Sdrama(7:0) due to inversion between .q and output
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                          *
"*****************************************************************************
"* Control pins
"*****************************************************************************
```

```
Ale_B           PIN     5;
AleIn           PIN     29;
AleOut_B        PIN     27 istype 'com' ; " inverted AleIn, connected externaly to
                        " Ale_B
R_C_B           PIN     33;
DimmSize        PIN     37;
PBI             PIN     25;

"*******************************************************************************
"* Address Input lines
"*******************************************************************************
" row
A10             PIN     34;
A11             PIN     1;
A12             PIN     2;
A13             PIN     3;
A14             PIN     9;
A15             PIN     10;
A16             PIN     11;
A17             PIN     12;
A18             PIN     35;
A19             PIN     36;
" col
A21             PIN     38;
A22             PIN     39;
A23             PIN     40;
A24             PIN     44;
A25             PIN     45;
A26             PIN     46;
A27             PIN     47;
A28             PIN     48;


"*******************************************************************************
"* Address Output lines
"*******************************************************************************
SdramA0         PIN     13 istype 'com' ;
SdramA1         PIN     14 istype 'com' ;
SdramA2         PIN     15 istype 'com' ;
SdramA3         PIN     16 istype 'com' ;
```

```
SdramA4          PIN     20 istype 'com' ;
SdramA5          PIN     21 istype 'com' ;
SdramA6          PIN     22 istype 'com' ;
SdramA7          PIN     23 istype 'com' ;
"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************

"*****************************************************************************
"*****************************************************************************
"* Latched Address lines
"*****************************************************************************
LA10        NODE istype 'reg_D,buffer' ;
LA11        NODE istype 'reg_D,buffer' ;
LA12        NODE istype 'reg_D,buffer' ;
LA13        NODE istype 'reg_D,buffer' ;
LA14        NODE istype 'reg_D,buffer' ;
LA15        NODE istype 'reg_D,buffer' ;
LA16        NODE istype 'reg_D,buffer' ;
LA17        NODE istype 'reg_D,buffer' ;
LA18        NODE istype 'reg_D,buffer' ;
LA19        NODE istype 'reg_D,buffer' ;
" col
LA21        NODE istype 'reg_D,buffer' ;
LA22        NODE istype 'reg_D,buffer' ;
LA23        NODE istype 'reg_D,buffer' ;
LA24        NODE istype 'reg_D,buffer' ;
LA25        NODE istype 'reg_D,buffer' ;
LA26        NODE istype 'reg_D,buffer' ;
LA27        NODE istype 'reg_D,buffer' ;
LA28        NODE istype 'reg_D,buffer' ;

"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z. ;
C, D, U    = .C., .D., .U. ;
"*****************************************************************************
"* SIMULATION = 1 ;
"*****************************************************************************
"* Signal groups
"*****************************************************************************
```

```
Add = [A10..A19,A21..A28] ;

LAdd = [LA10..LA19,LA21..LA28] ;

RowAddNormal = [LA12..LA19] ;
RowAddPBI_16M = [LA11..LA18] ;
RowAddPBI_64M = [LA10..LA17] ;

ColAdd = [LA21..LA28] ;

SdramAdd = [SdramA7..SdramA0] ;


ROW = (R_C_B == 1) ;
COL = !ROW ;

SIZE_16M = 0;
SIZE_64M = 1;

SDRAM_16M = (DimmSize == SIZE_16M);
SDRAM_64M = !SDRAM_16M;

SDRAM_NORMAL_MODE = (PBI == 0);
SDRAM_PBI_MODE = !SDRAM_NORMAL_MODE;

MuxCont = [PBI,DimmSize];

"*******************************************************************************
"* Equations, state diagrams.                                                  *
"*******************************************************************************
"*******************************************************************************
"*******************************************************************************
"* Input Latch
"*******************************************************************************
"*******************************************************************************
equations

!AleOut_B = AleIn ;  "  inverted Ale
```

```
AleOut_B.oe = H;

LAdd.le = Ale_B ;

LAdd.d = Add ;    " latching the address

"*****************************************************************************
"*****************************************************************************
"* Output Mux
"*****************************************************************************
"*****************************************************************************

equations

SdramAdd.oe = ^hff ;   "always enabled

when (ROW & SDRAM_NORMAL_MODE) then
    SdramAdd = !RowAddNormal.q
else when (ROW & SDRAM_PBI_MODE & SDRAM_16M) then
    SdramAdd = !RowAddPBI_16M.q
else when (ROW & SDRAM_PBI_MODE & SDRAM_64M) then
    SdramAdd = !RowAddPBI_64M.q
else when (COL) then
    SdramAdd = !ColAdd.q ;
" input open, running 0 msb to lsb
@ifdef SIMULATION {
}
END
```


7.3 Schematics and Bill Of Materials

This sectoin shows the schematicds of the MPC8266ADS-PCI aith the Bill Of Materials.

7.3.1 Schematics

Following are the schematics of the board.

Table of Contents

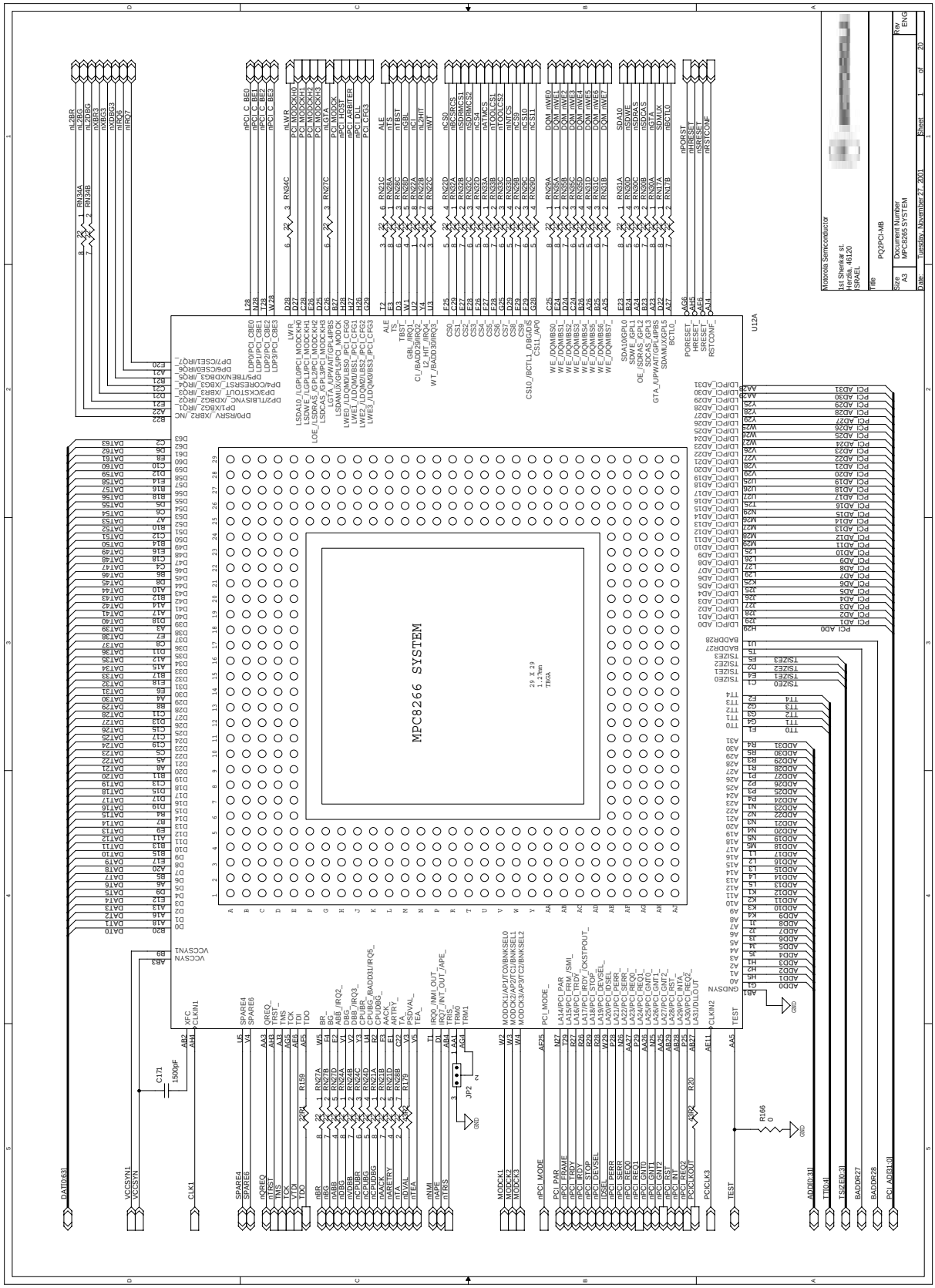
Page	Description
	Title Page
1	MPC8266 System
2	System Buffers
3	Expansion Buffers
4	BCSRs
5	FLASH and EEPROM Memories
6	SDRAM Memory
7	L2Cache
8	MPC8266 CPM
9	ATM
10	Fast-Ethernet
11	RS232
12	Pull-Up / Pull-Down Resistors
13	MPC8266 Power
14	PCI Connectors
15	PCI Connectors
16	PCI Interrupt Controller
17	Expansion Connectors (2 x 128 pin)
18	Logic Analyzer Connectors - Mictors
19	Power
20	JTAG, General Parts

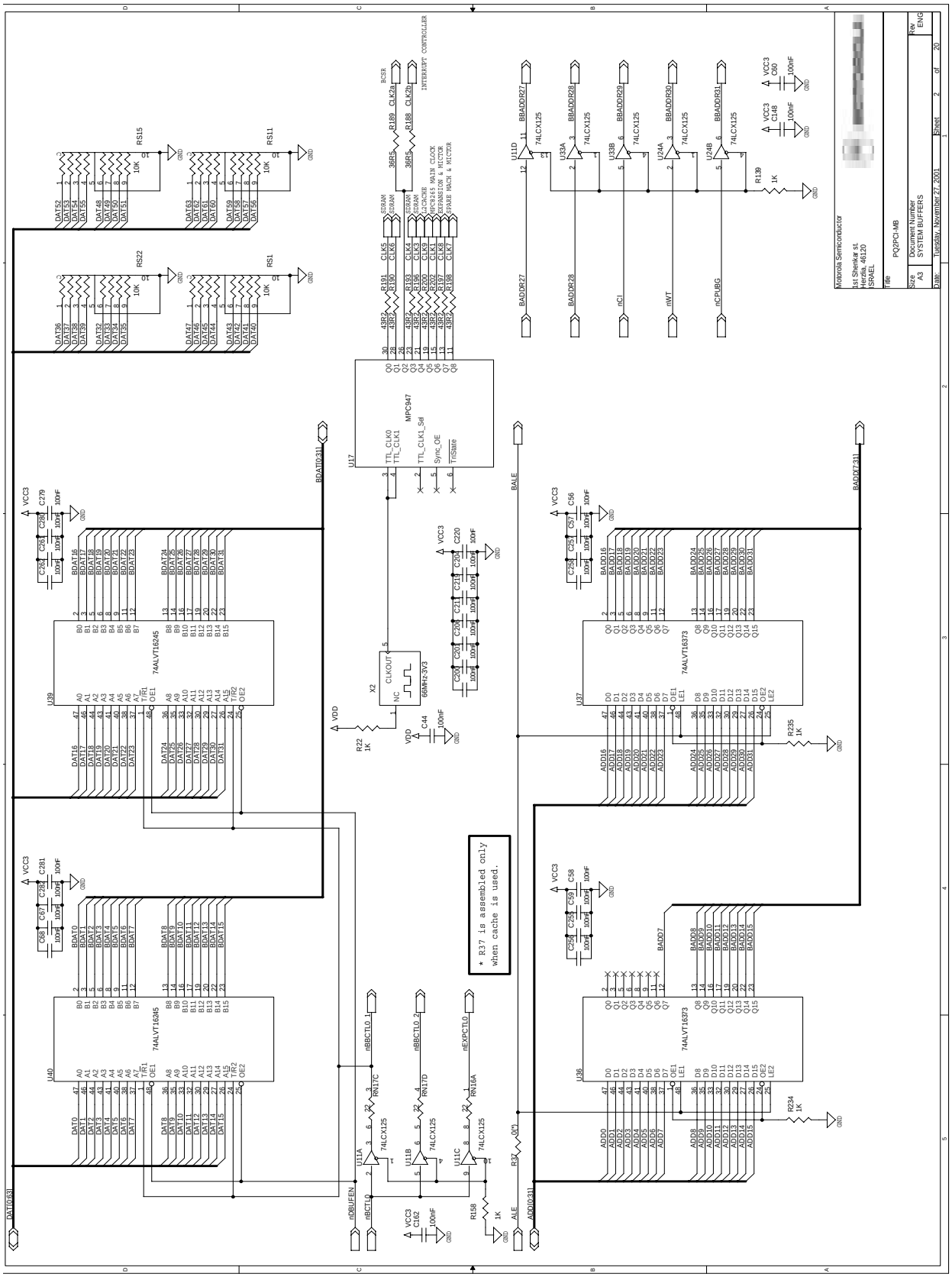
Digital DNA from Motorola

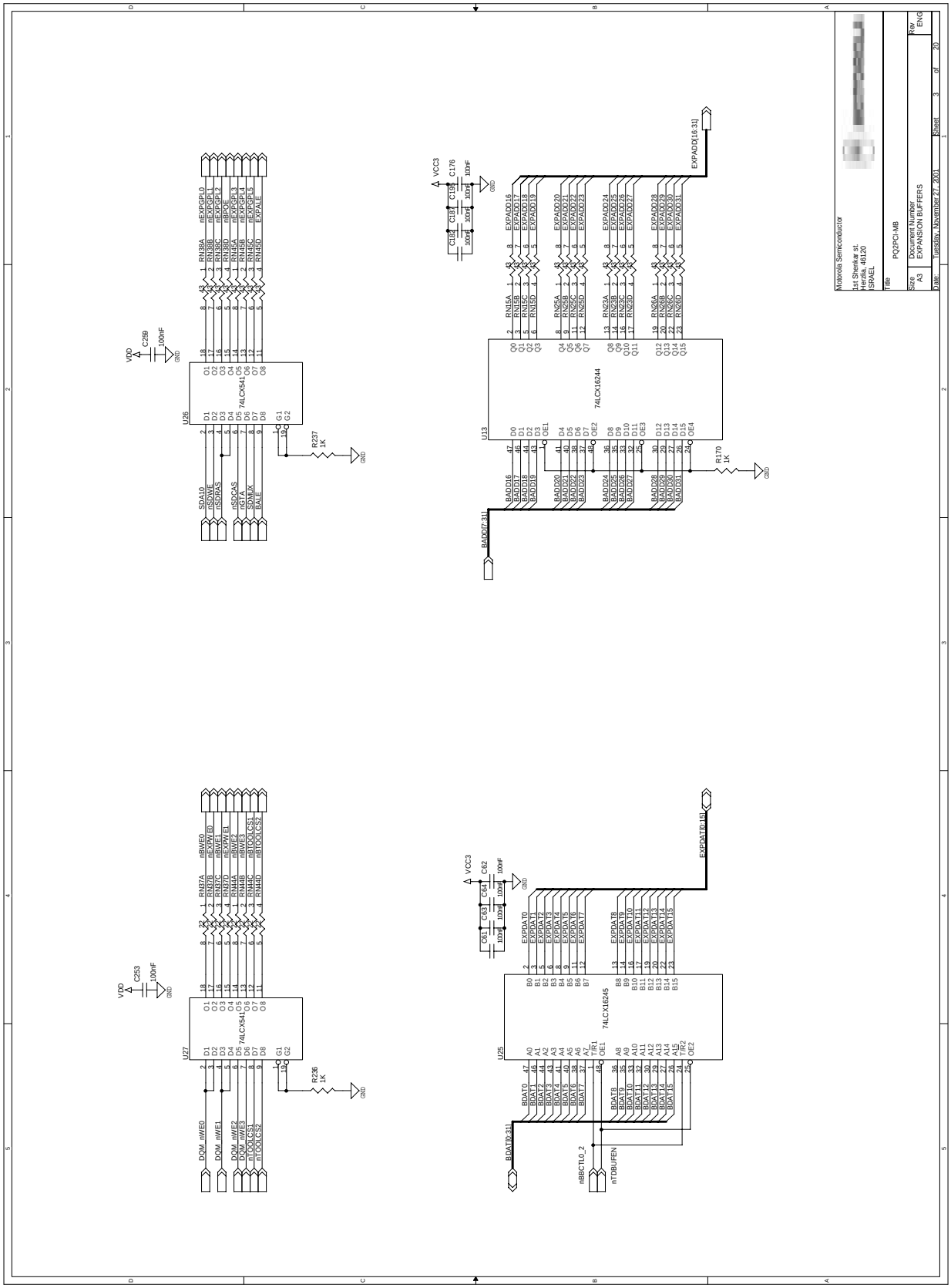
PowerQUICC II*

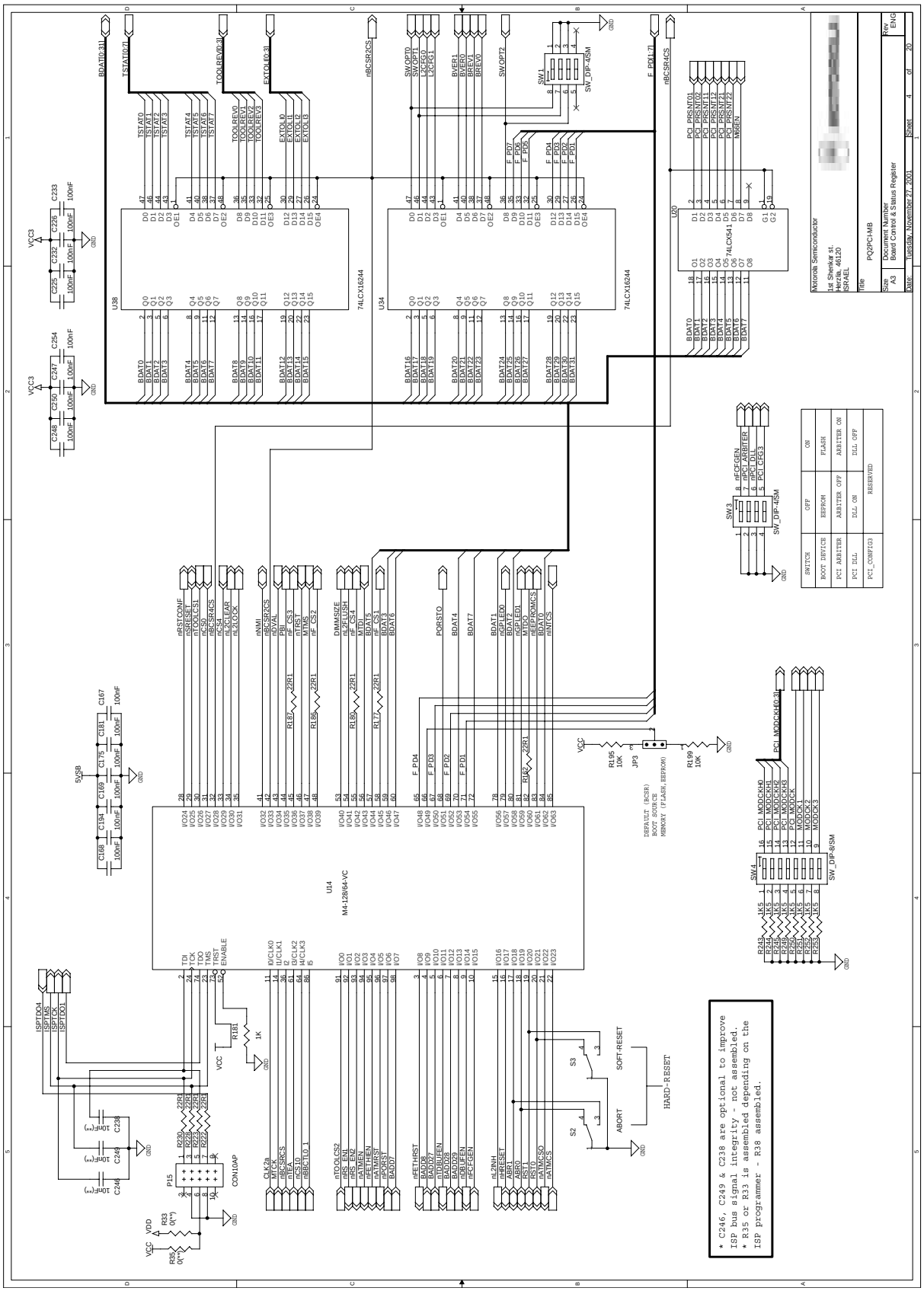
MPC8266 ADS - PCI

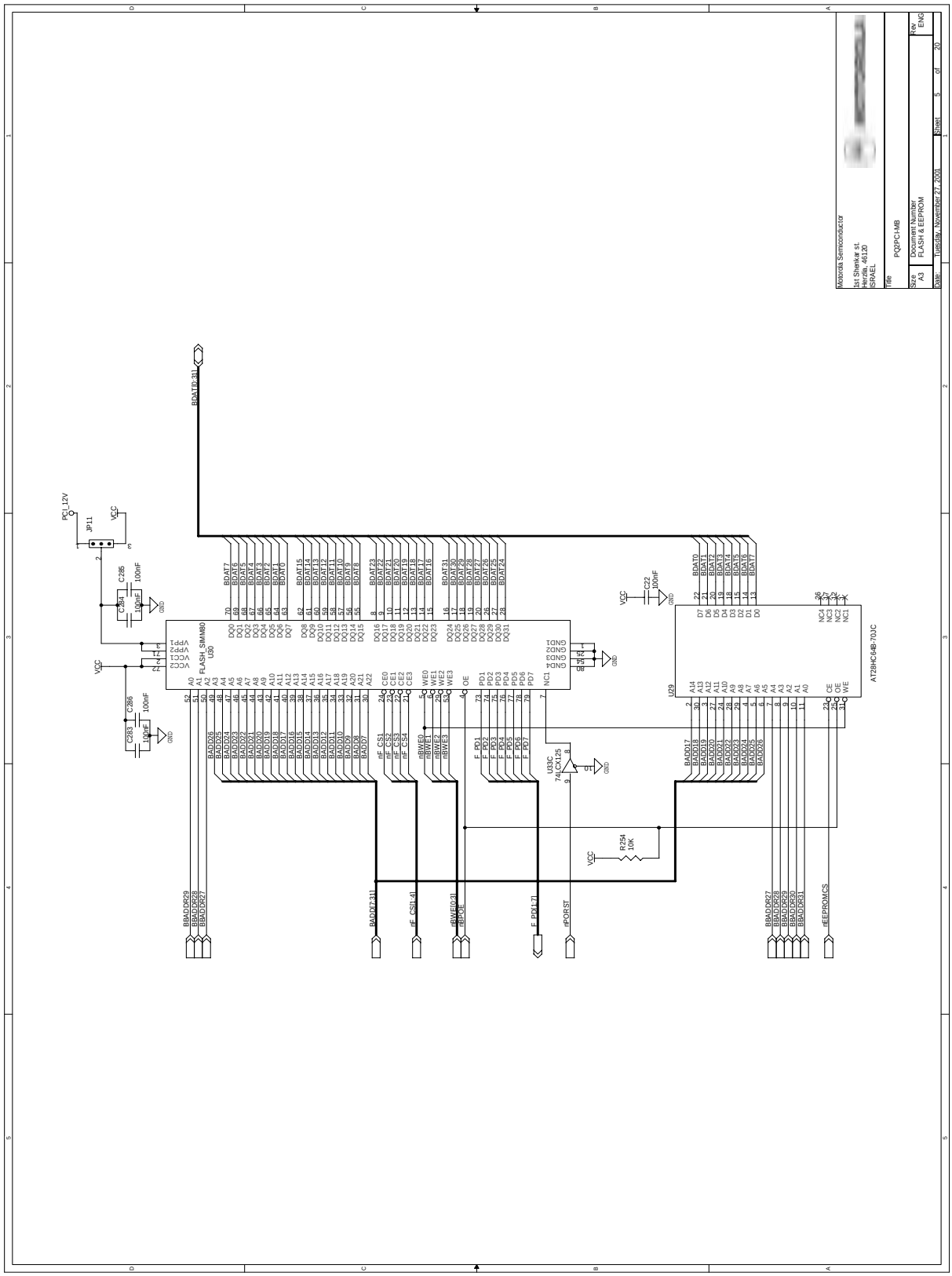
MPC8266 PCI MotherBoard

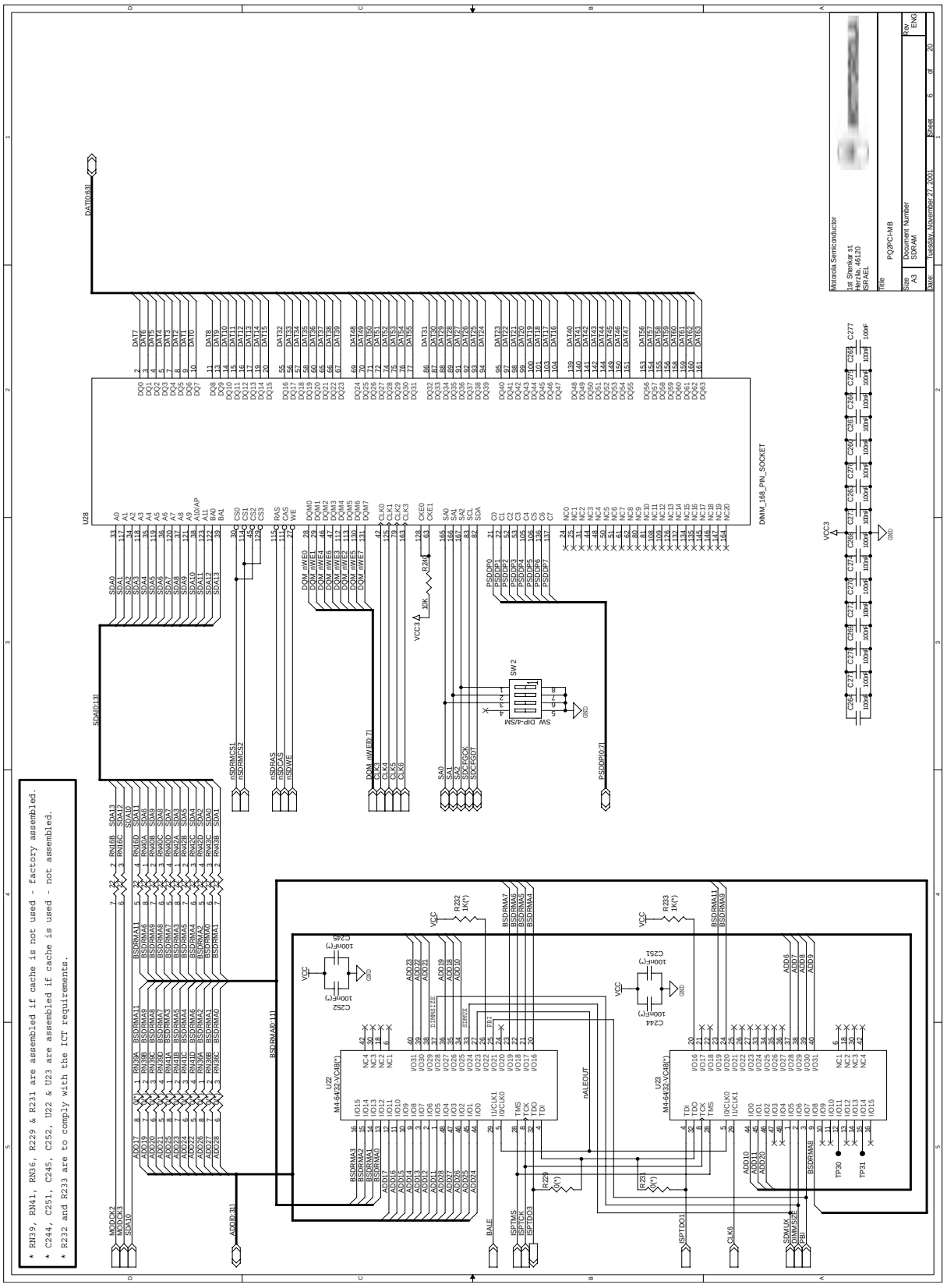


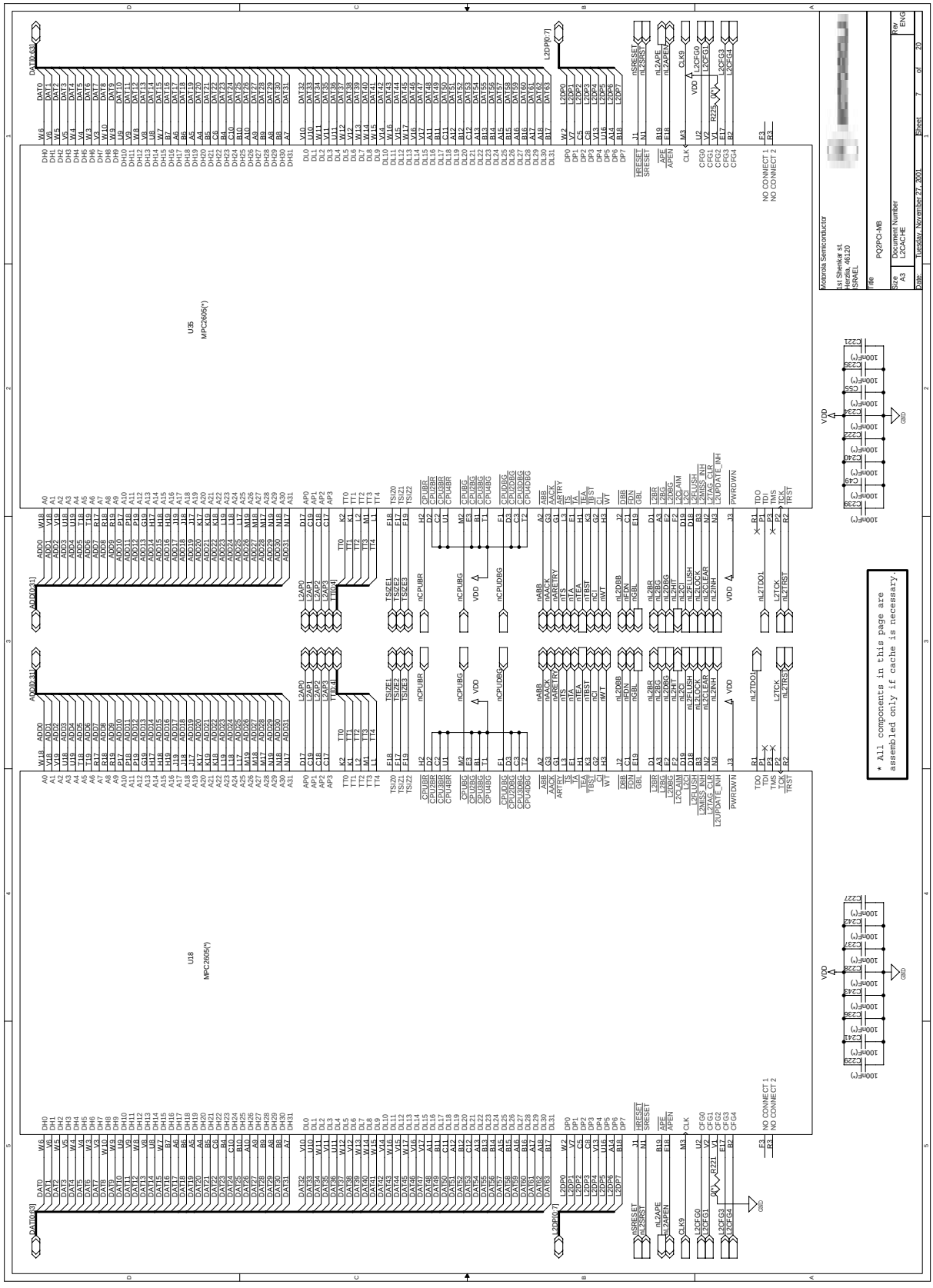


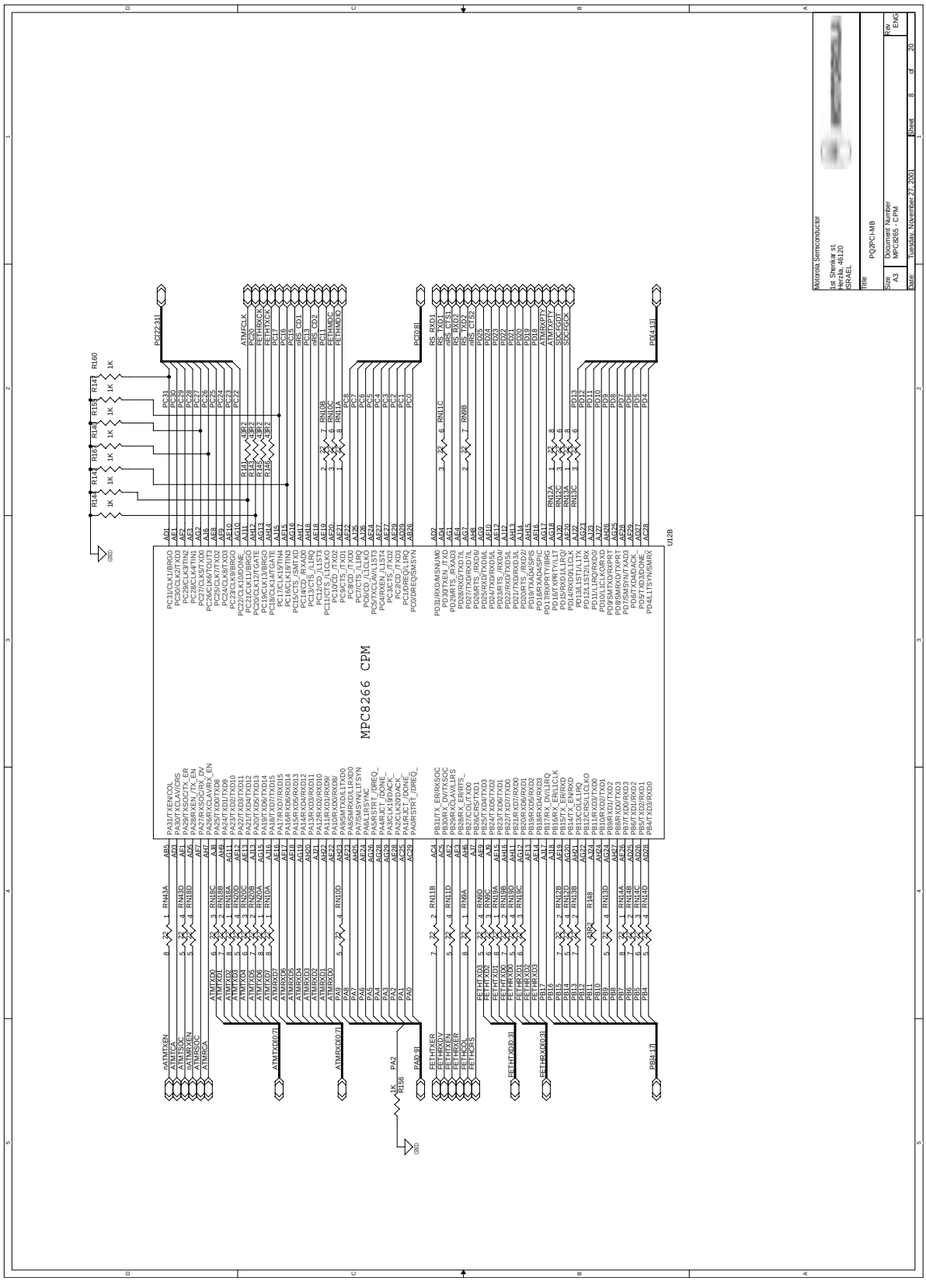


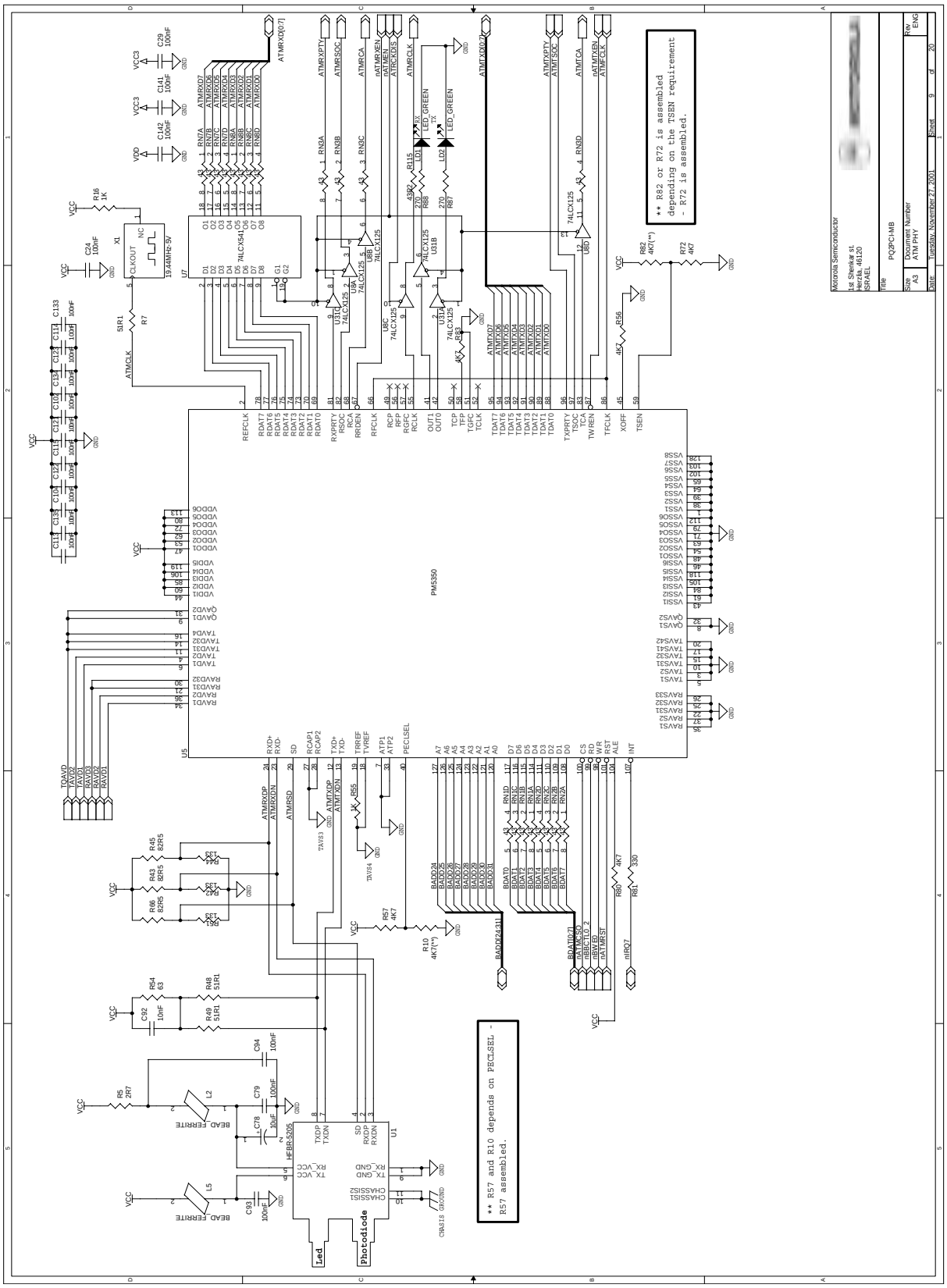


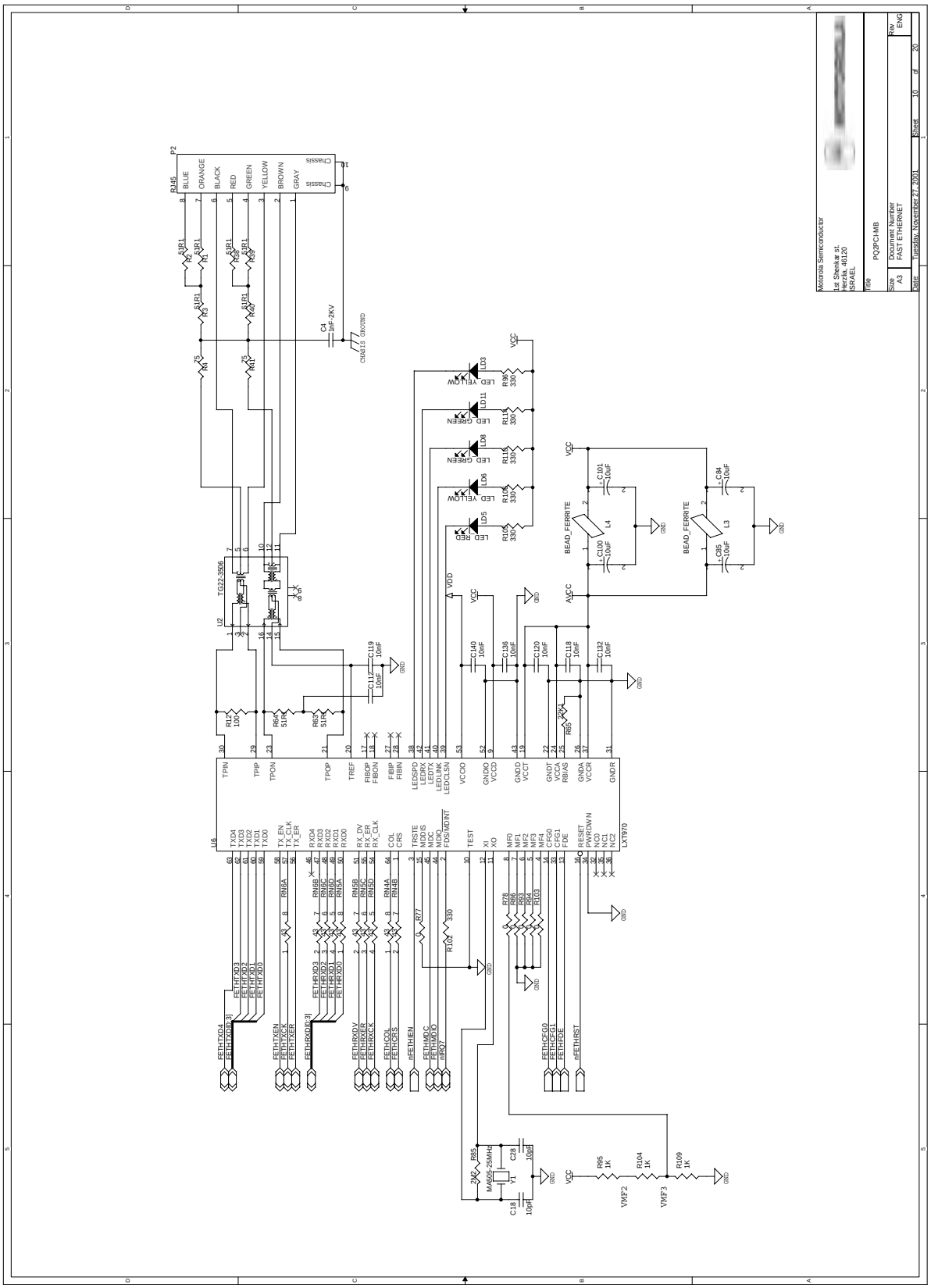


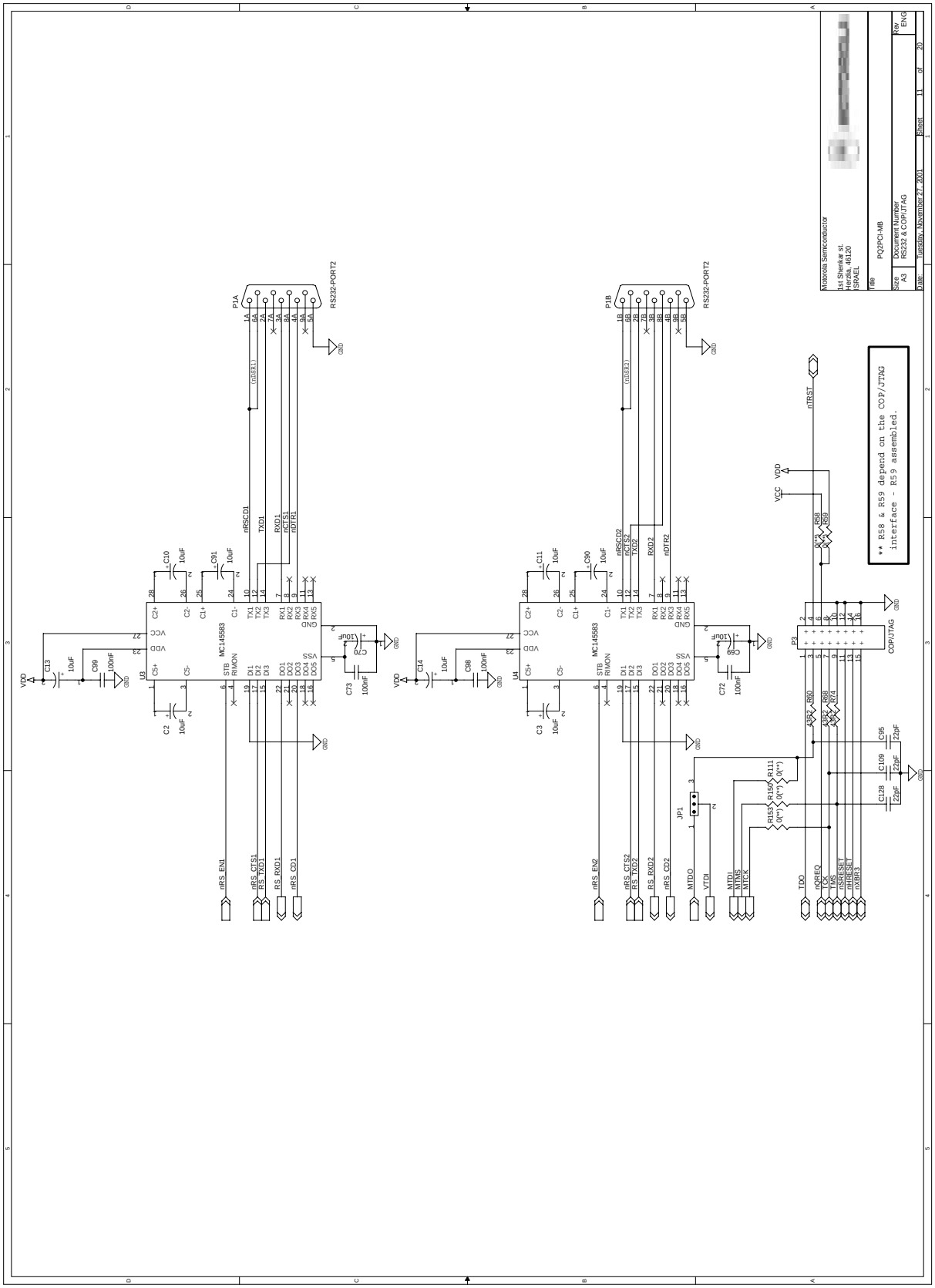


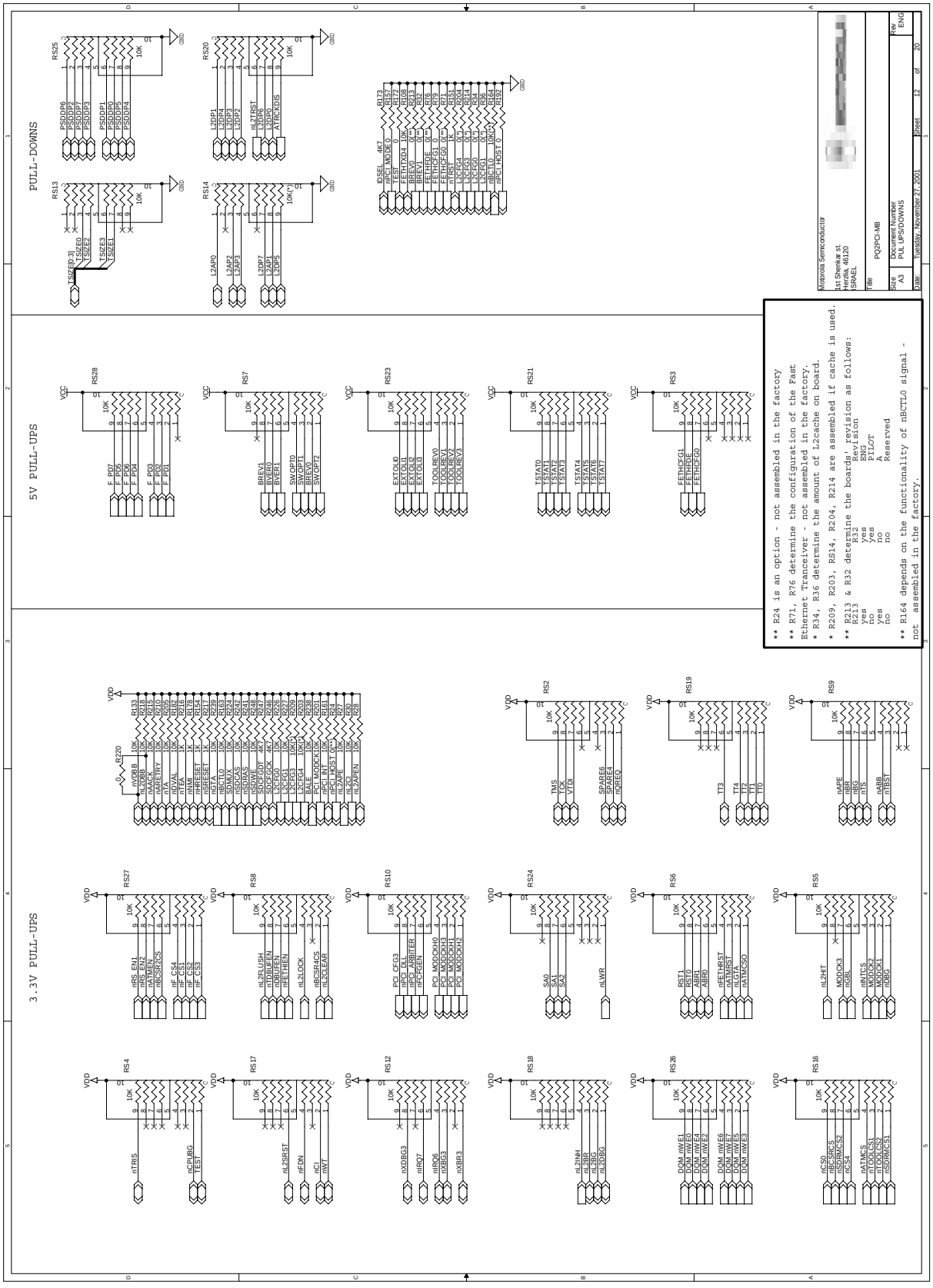


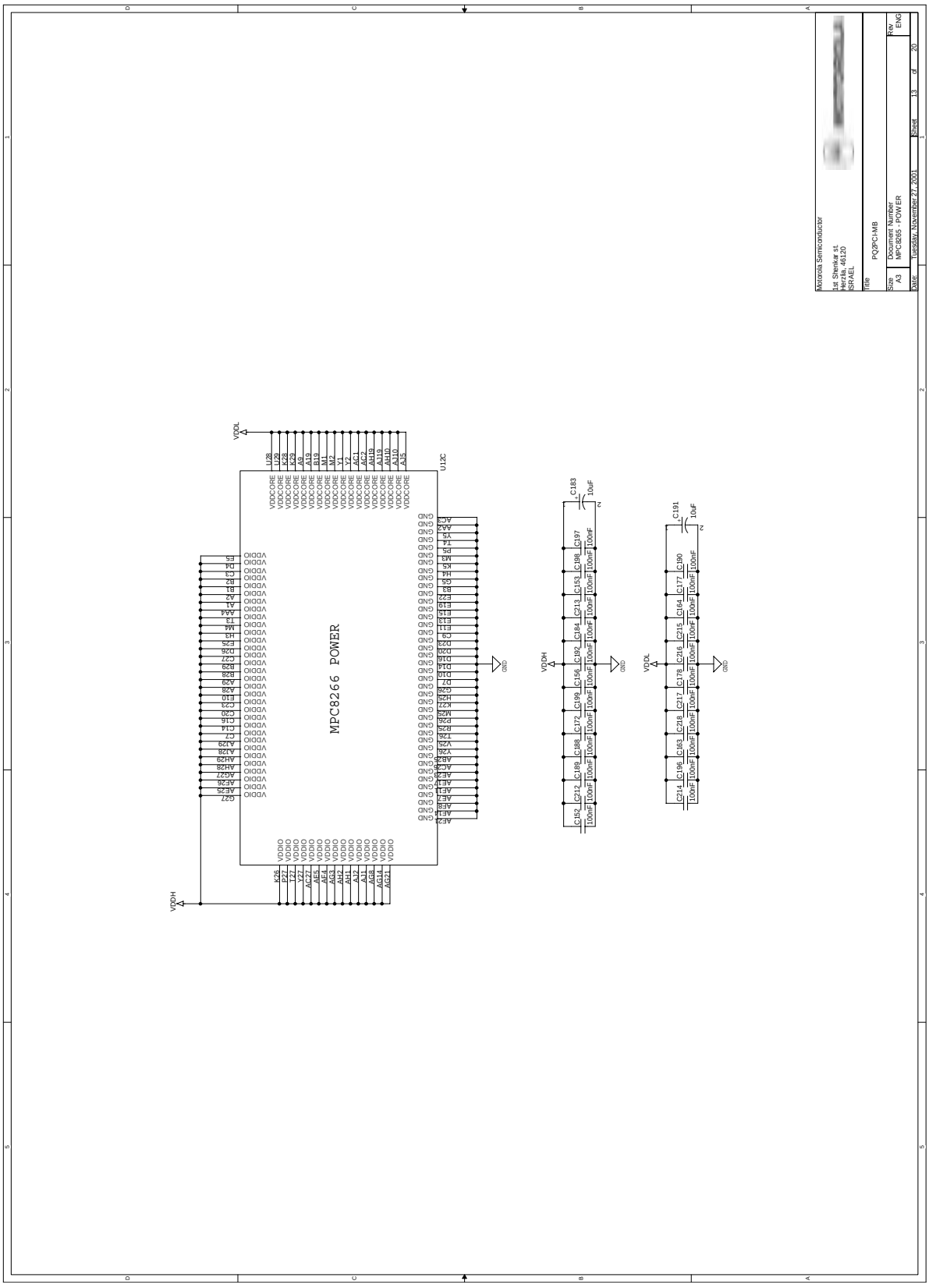


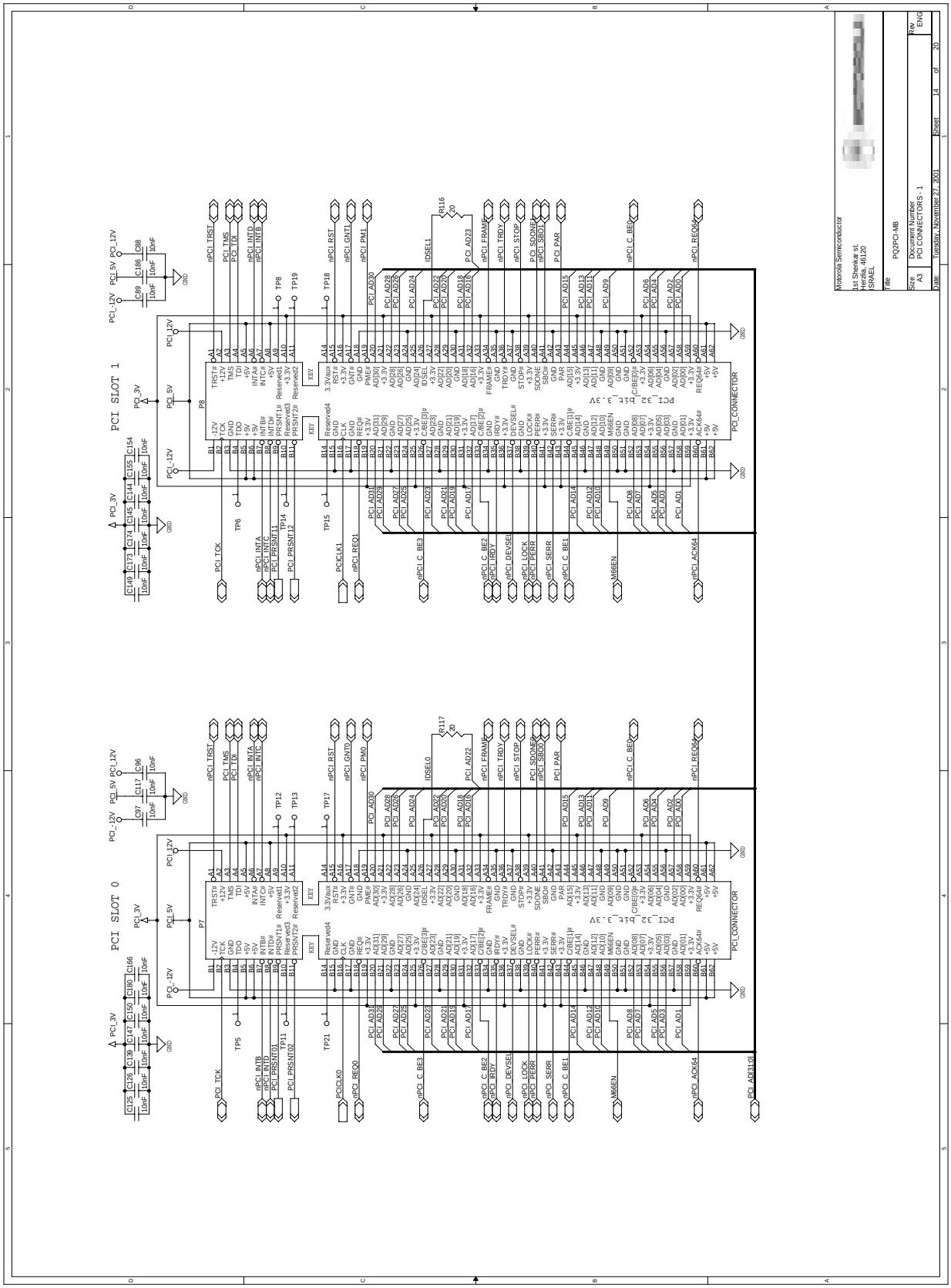


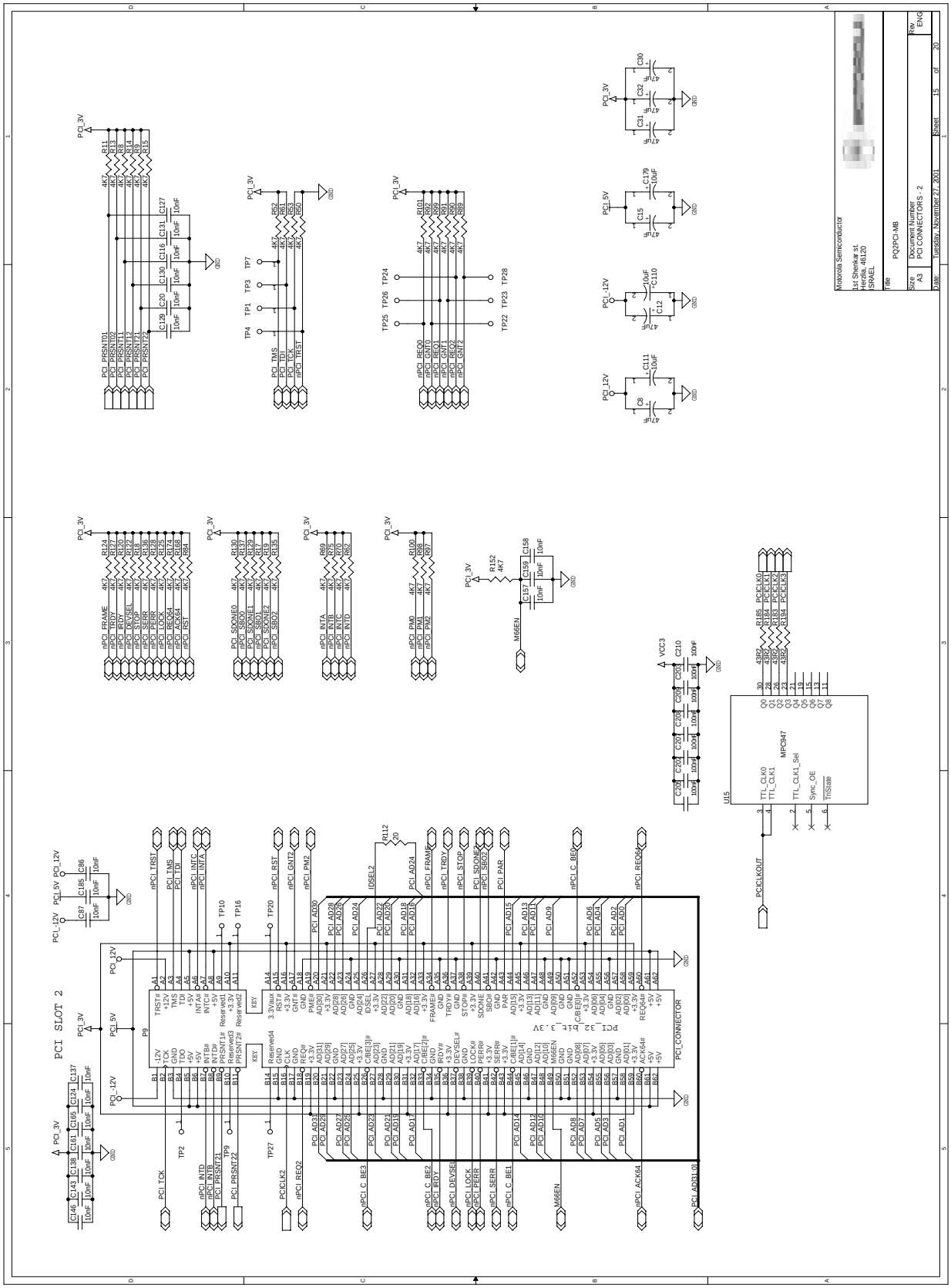


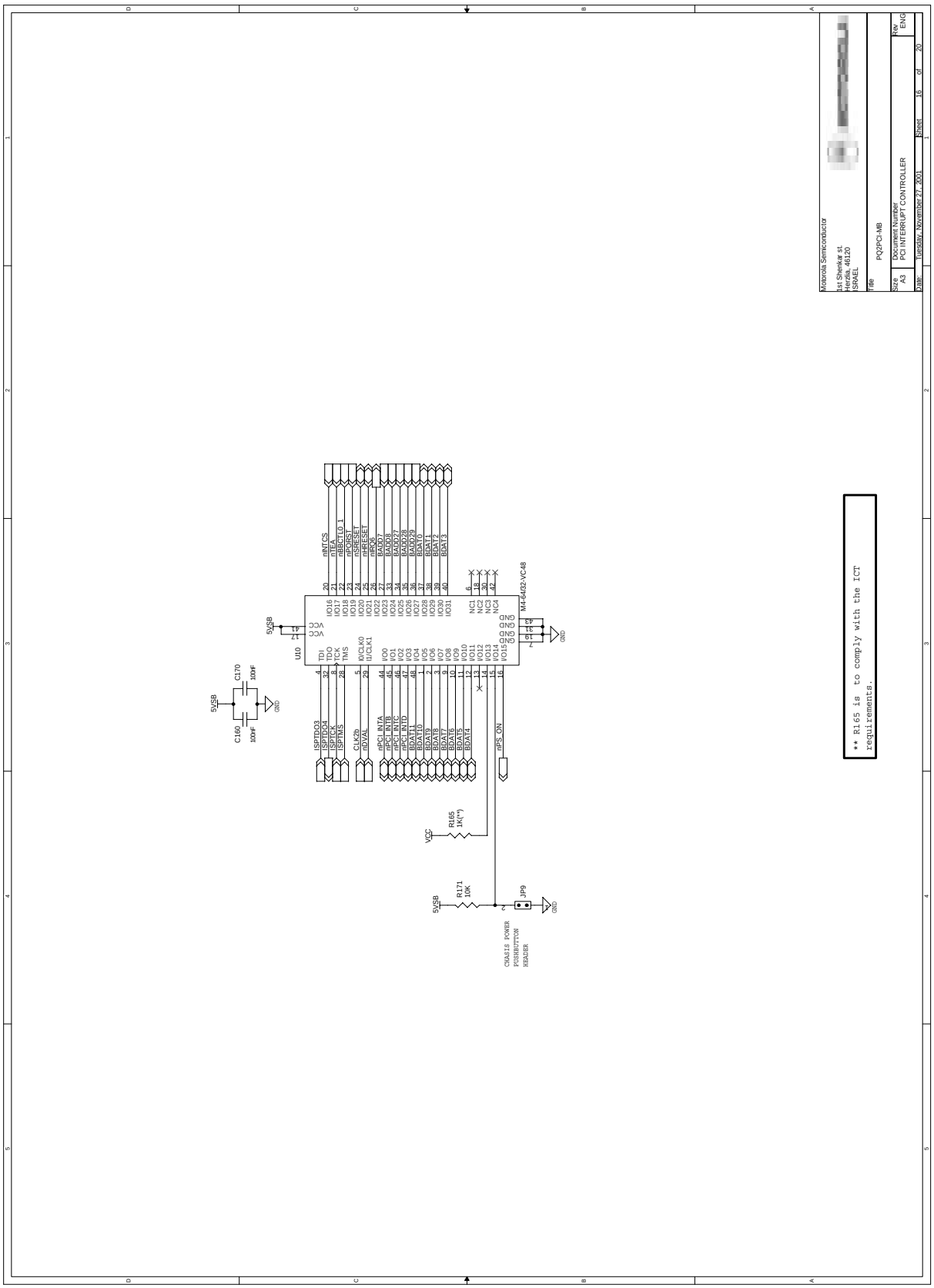


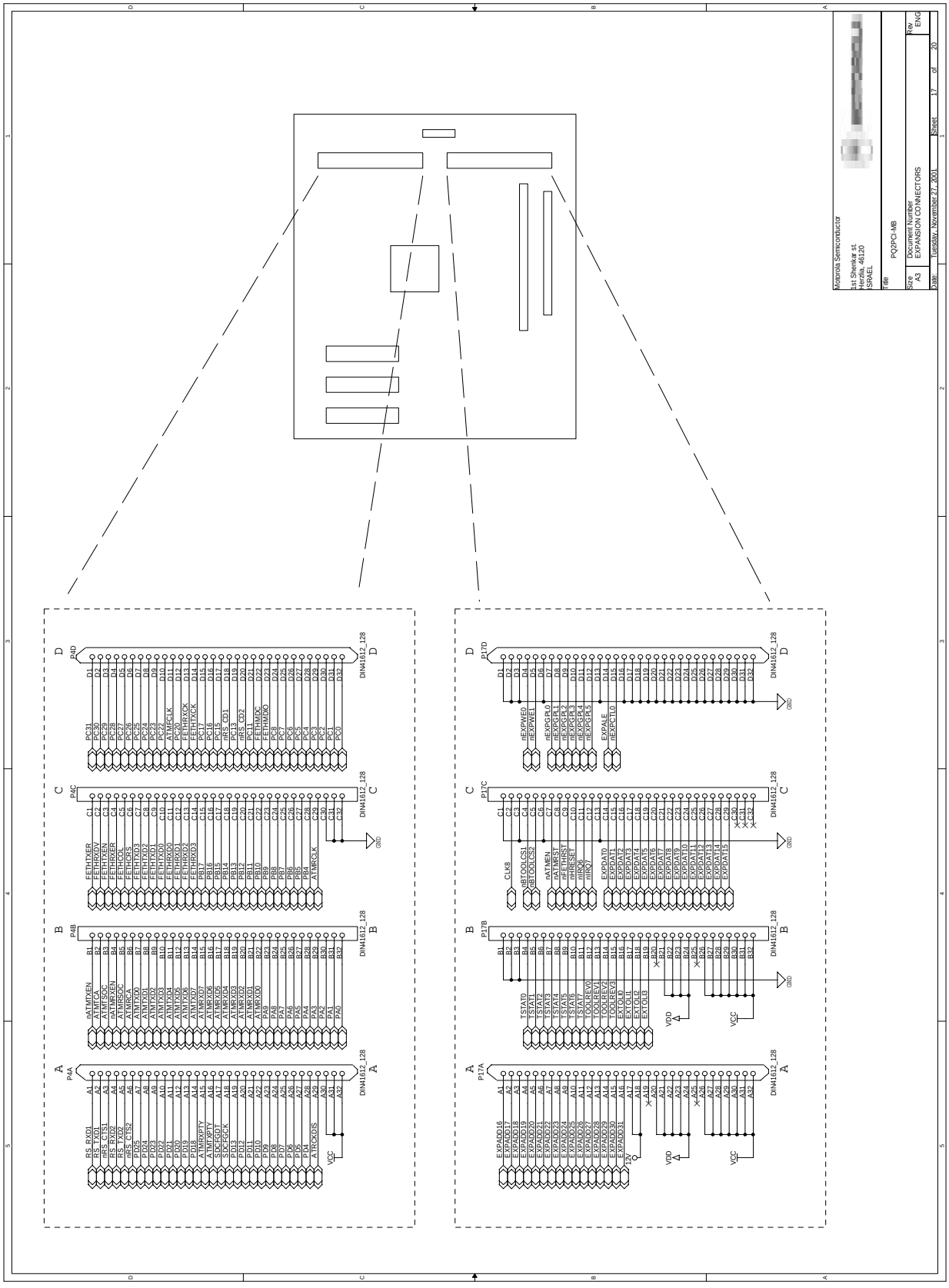


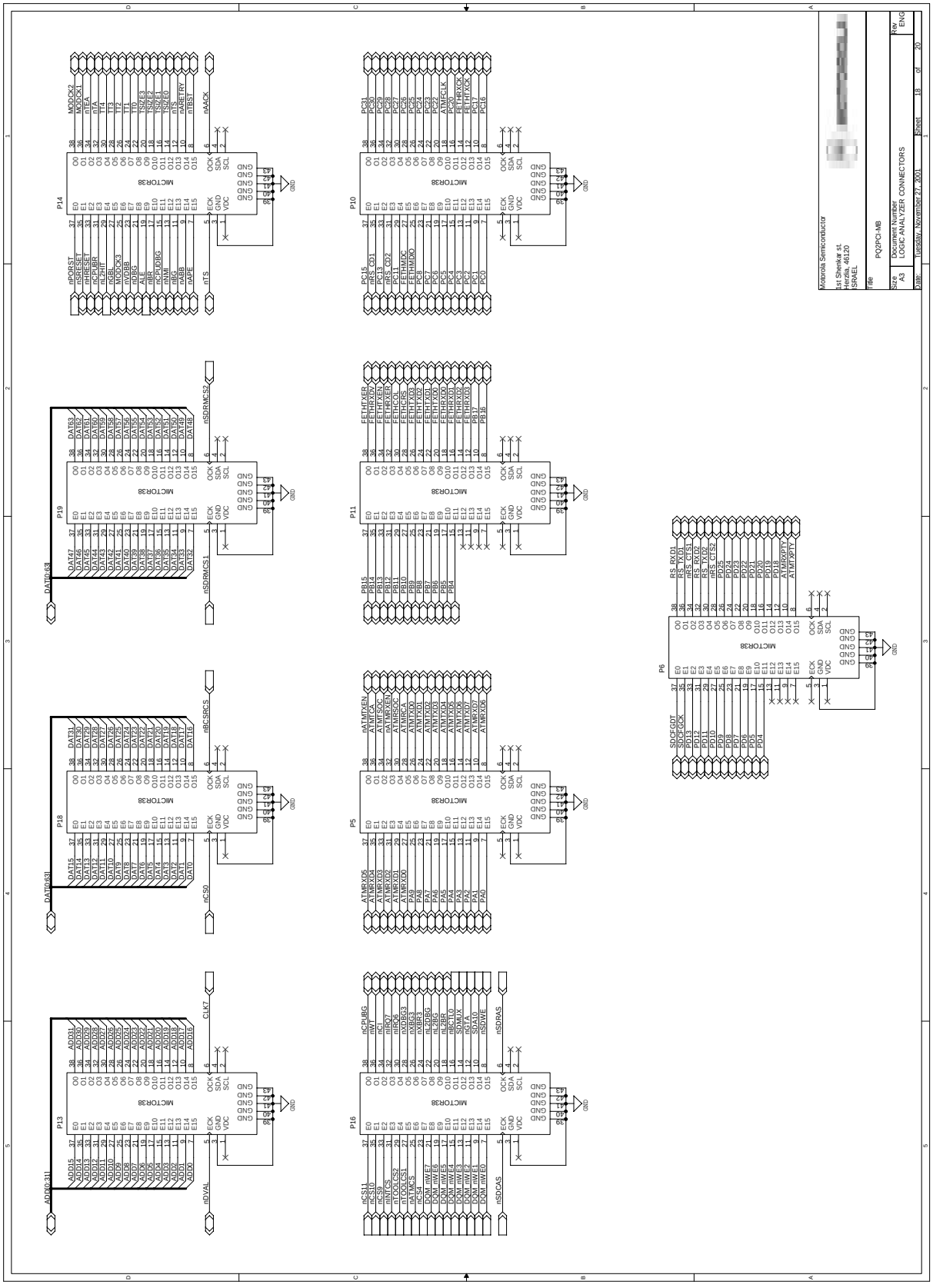


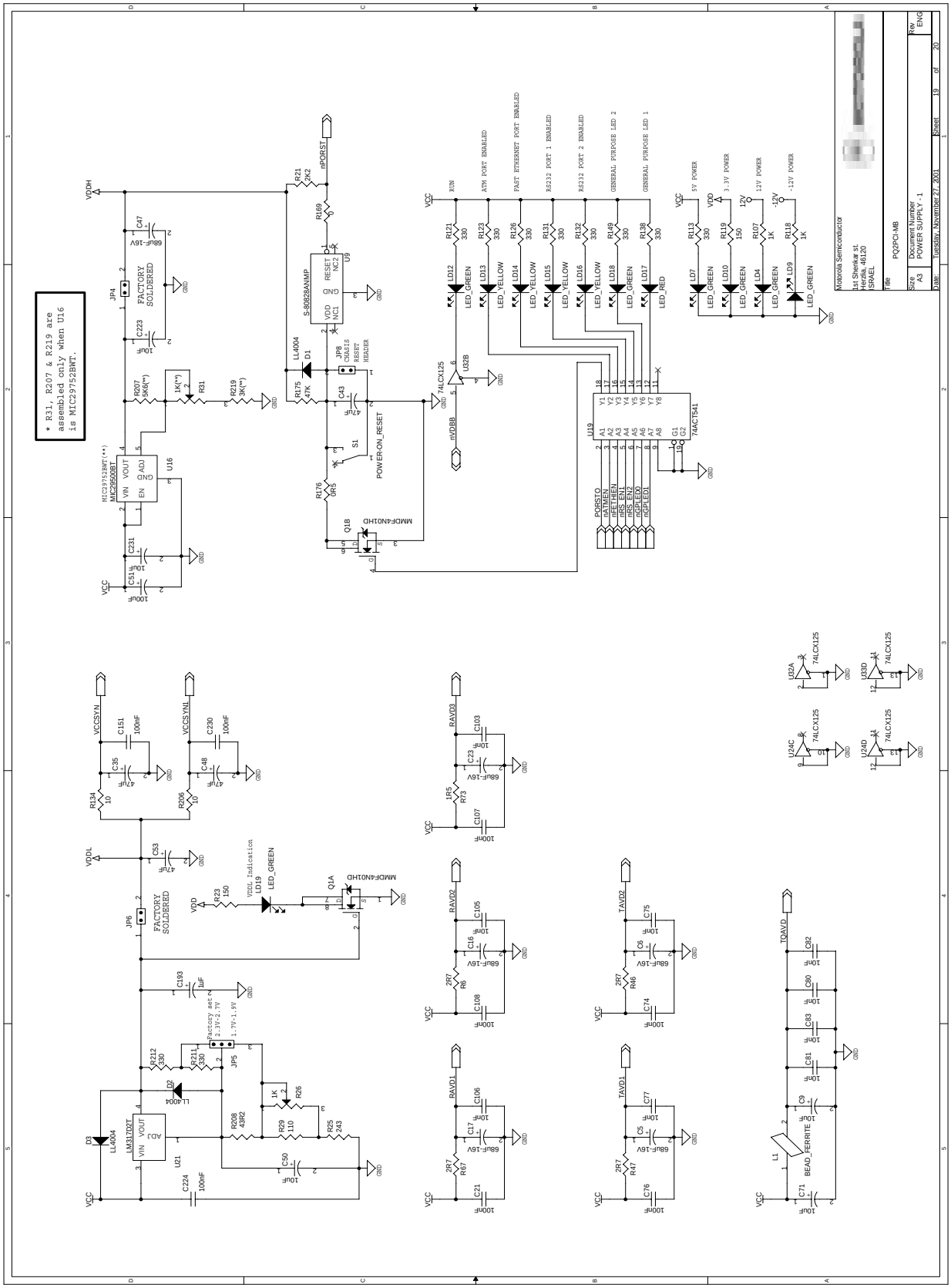


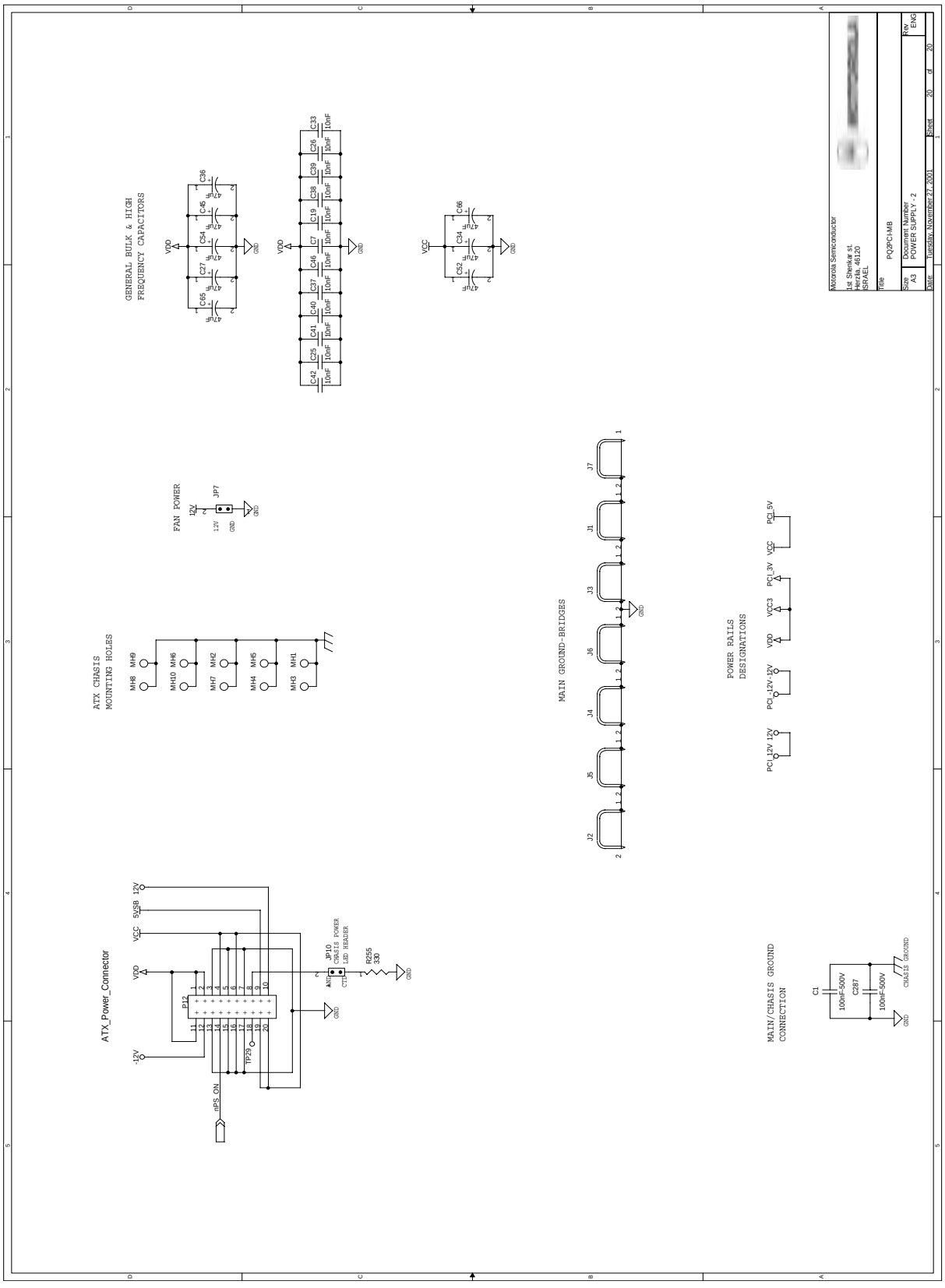












7.3.2 Bill of Materials

The following is the Bill Of Materials for the MPC8266ADS-PCI including the L2Cache option.

Table 7-9. MPC8266ADS-PCI Bill Of Materials

Reference	Qty	Footprint	Description	Manufacturer	Part Number	Note
C287,C1	2	1812	CAP 100nF 500V	JOHANSON	501S43W104MV4E	
C2,C3,C9,C10,C11,C13,C14, C50,C69,C70,C71,C78,C84, C85,C90,C91,C100,C101, C110,C111,C179,C183,C191, C223,C231	25	CASE_C	CAP 10uF 25V	AVX	TAJC106K025R	
C4	1	1210	CAP 1nF 2KV	AVX	1210B102K202NT	
C5,C6,C16,C17,C23,C47	6	CASE_D	CAP 68uF 20V	AVX	TAJD686M020R	
C7,C19,C20,C25,C26,C33, C37,C38,C39,C40,C41,C42, C46,C75,C77,C80,C81,C82, C83,C86,C87,C88,C89,C92, C96,C97,C103,C105,C106, C112,C116,C117,C118,C119, C120,C124,C125,C126,C127, C129,C130,C131,C132,C136, C137,C138,C139,C140,C143, C144,C145,C146,C147,C149, C150,C154,C155,C157,C158, C159,C161,C165,C166,C173, C174,C180,C185,C186	68	603	CAP 10nF	AVX	06035C103KAT2A	
C8,C12,C15,C27,C30,C31, C32,C34,C35,C36,C43,C45, C48,C52,C53,C54,C65,C66	18	CASE_D	CAP 47uF 16V	AVX	TAJD476K016	
C28,C18	2	1206	CAP 10pF	AVX	AV12065A100JATJ	
C21,C22,C24,C29,C44,C56, C57,C58,C59,C60,C61,C62, C63,C64,C67,C68,C72,C73, C74,C76,C79,C93,C94,C98, C99,C102,C104,C107,C108, C113,C114,C115,C121,C122, C123,C133,C134,C135,C141, C142,C148,C151,C152,C153, C156,C160,C162,C163,C164, C167,C168,C169,C170,C172, C175,C176,C177,C178,C181, C182,C184,C187,C188,C189, C190,C192,C194,C195,C196, C197,C198,C199,C200,C201, C202,C203,C204,C205,C206, C207,C208,C209,C210,C211, C212,C213,C214,C215,C216, C217,C218,C219,C220,C224, C225,C226,C230,C232,C233, C247,C248,C250,C253,C254, C255,C256,C257,C258,C259, C260,C261,C262,C263,C264, C265,C266,C267,C268,C269, C270,C271,C272,C273,C274, C275,C276,C277,C278,C279, C280,C281,C282,C283,C284, C285,C286	136	603	CAP 100nF	AVX	0603YC104KAT2A	
C49,C55,C221,C222,C227, C228,C229,C234,C235,C236, C237,C239,C240,C241,C242, C243,C244,C245,C251,C252	20	603	CAP 100nF	AVX	0603YC104KAT2A	
C51	1	CASE_C	CAP 100uF 16V	AVX	TAJD107K016R	
C95,C109,C128	3	1206	CAP 22pF	AVX	12065A220JAT00J	

Table 7-9. MPC8266ADS-PCI Bill Of Materials

Reference	Qty	Footprint	Description	Manufacturer	Part Number	Note
C171	1	1206	CAP 1500pF	AVX	12065A152JAT00J	
C193	1	CASE_A	CAP 1uF 25V	SIEMENS	B45196H5105K109	
C238,C246,C249	3	603	CAP 10nF	AVX	06035C103KAT2A	Optional
D1,D2,D3	3	SMD	DIODE	TSC	LL4004G	
JP1,JP2,JP3,JP5,JP11	5	3P_TH	HEADER 3Px1ROW	MOLEX	87156-0303	
JP4,JP6	2	100mil_TH	GND BRIDGE 100mil	PRECIDIP	PD-999-11-11010	
JP7,JP8,JP9,JP10	4	2P_TH	HEADER 2Px1ROW	MOLEX	87156-4003	
J1,J2,J3,J4,J5,J6,J7	7	5mm_TH	GND BRIDGE 5mm	PRECIDIP	PD-999-11-11210	
LD1,LD2,LD4,LD7,LD8,LD9, LD10,LD11,LD12,LD18,LD19	11	LED	LED GREEN	KINGBRIGHT	KPT-3216SGD	
LD3,LD6,LD13,LD14,LD15, LD16	6	LED	LED YELLOW	KINGBRIGHT	KPT-3216YD	
LD5	1	LED	LED RED	KINGBRIGHT	KPT-3216ID	
LD17	1	LED	LED RED	KINGBRIGHT	KPT-3216YD	
L1,L2,L3,L4,L5	5	FERRITE	FERRITE BEAD	FAIR RITE	2743021447	
P1	1	Stacked_9P_D-Type	CON 9P DUAL	EDA	8LE009009D306H	
P2	1	8P_100mil_TH	CON RJ45	MOLEX	43202-8110	
P3	1	16P_2x8_SMD	CON 16P HEADER	KCC	LPH-16SA-SG	
P4,P17	2	128P_DIN_FEM_conn_90	CON DIN 128	ERNI	23762	
P5,P6,P10,P11,P13,P14, P16,P18,P19	9	MICTOR38	CON MICTOR 38P	AMP	2-767004-2	
P7,P8,P9	3	PCI_2x60_TH	CON PCI 32BIT	AMP	145154-4	
P12	1	10X2_HEADER	CON 20P ATX PWR	MOLEX	39-29-9202	
P15	1	10P_2x5_SMD	HEADER 5Px2ROWS	SAMTEC	TSM-10501-SDV-AP	
Q1	1	SO8	IC DUAL TMOS	ON SEMICONDUCTOR	MMDF4N01HD	
RN1,RN2,RN3,RN4,RN5,RN6, RN7,RN8,RN15,RN23,RN25, RN26,RN38,RN45	14	0603X4	RES NET 8P 4R 43ohm	DALE	CRA06S0803430JRT	
RN9,RN10,RN11,RN12,RN13, RN14,RN16,RN17,RN18,RN19, RN20,RN21,RN22,RN24,RN27, RN28,RN29,RN30,RN31,RN32, RN33,RN34,RN35,RN37,RN40, RN42,RN43,RN44	28	0603X4	RES NET 8P 4R 22ohm	AVX	CRA3A4E220JT	
RN36,RN39,RN41	3	4X0603	RES NET 8P 4R 0ohm	DALE	CRA06S0803000RT	Not L2Cache
RS1,RS2,RS3,RS4,RS5,RS6, RS7,RS8,RS9,RS10,RS11, RS12,RS13,RS15,RS16,RS17, RS18,RS19,RS20,RS21,RS22, RS23,RS24,RS25,RS26,RS27, RS28	27	10P-8R	RES NET 10P 8R 10Kohm	ROHM	RS8A1002J	

Table 7-9. MPC8266ADS-PCI Bill Of Materials

Reference	Qty	Footprint	Description	Manufacturer	Part Number	Note
RS14	1	10P-8R	RES NET 10P 8R 10Kohm	ROHM	RS8A1002J	L2Cache
R1,R2,R3,R38,R39,R40,R48, R49,R63,R64	10	603	RES	ROEDERSTEIN	D1151R1FCS	
R41,R4	2	603	RES 75ohm	DRALORIK	D11075RFCS	
R5,R6,R46,R47,R67	5	1206	RES 2R7ohm	ROEDERSTEIN	D2502R7FCS	
R7	1	603	RES 51R1ohm	ROEDERSTEIN	D1151R1FCS	
R8,R9,R11,R13,R14,R15, R17,R18,R19,R50,R52,R53, R56,R57,R61,R62,R69,R70, R72,R75,R80,R83,R84,R89, R90,R91,R92,R97,R98,R99, R100,R101,R120,R122,R124, R125,R127,R128,R129,R130, R135,R136,R137,R152,R168, R173,R174,R246,R247	49	603	RES 4K7ohm	ROEDERSTEIN	D1104K7FCS	
R10,R82	2	603	RES 4K7ohm	ROEDERSTEIN	D1104K7FCS	Optional
R12	1	1206	RES 100ohm	ROEDERSTEIN	D25100RFCS	
R16,R22,R55,R139,R140, R142,R144,R147,R151,R154, R155,R156,R158,R160,R167, R170,R178,R181,R216,R217, R234,R235,R236,R237	24	603	RES 1Kohm	DRALORIK	D11001KFCS	
R20,R60,R68,R74,R115, R141,R143,R145,R146,R148, R179,R183,R184,R185,R190, R191,R193,R194,R196,R197, R198,R200,R202,R208	24	603	RES 43R2ohm	ROEDERSTEIN	D1143R2FCS	
R21	1	1206	RES 2K2ohm	ROEDERSTEIN	D2502K2FCS	
R119,R23	2	1206	RES 150ohm	ROEDERSTEIN	D25150RFCS	
R24,R32,R58,R59,R71,R76, R111,R150,R153,R213	10	603	RES 0ohm	ROEDERSTEIN	D11000RFCS	
R25	1	1206	RES 243ohm	ROEDERSTEIN	D25243RFCS	
R26	1	Var_Res	RES VAR 1Kohm	BOURNS	3362P-1-102	
R27,R28,R30,R108,R133, R161,R163,R171,R182,R195, R199,R201,R205,R210,R215, R218,R224,R226,R227,R238, R239,R240,R241,R242,R248, R254	26	603	RES 10Kohm	ROEDERSTEIN	D11010KFCS	
R29	1	1206	RES 110ohm	AVX	CR32111J-T	
R31	1	Var_Res	RES VAR 1Kohm	BOURNS	3362P-1-102	Optional
R33,R35	2	1206	RES 0ohm	ROEDERSTEIN	D25000RFCS	Optional
R34,R36,R37,R204,R214, R221,R225,R229,R231	9	603	RES 0ohm	ROEDERSTEIN	D11000RFCS	
R42,R44,R51	3	1206	RES 133ohm	DRALORIK	D25133RFCS	
R43,R45,R66	3	1206	RES 82R5ohm	DRALORIK	D2582R5FCS	
R54	1	1206	RES 63ohm	DRALORIK	D2563R4FCS	
R65	1	1206	RES 22K1ohm	DRALORIK	D2522K1FCS	

Table 7-9. MPC8266ADS-PCI Bill Of Materials

Reference	Qty	Footprint	Description	Manufacturer	Part Number	Note
R73	1	1206	RES 1R5ohm	ROEDERSTEIN	D2501R5FCS	
R77,R78,R79,R86,R93,R94, R103,R157,R166,R169,R172, R192,R220	13	603	RES 0ohm	ROEDERSTEIN	D11000RFCS	
R81,R96,R102,R105,R106, R110,R113,R114,R121,R123, R126,R131,R132,R138,R149, R211,R212,R255	18	1206	RES 330ohm	ROEDERSTEIN	D25330RJCS	
R85	1	1206	RES 2M2ohm	ROEDERSTEIN	D2502M2FCS	
R88,R87	2	1206	RES 270ohm	DRALORIK	D25270RFCS	
R95,R104,R109	3	1206	RES 1Kohm	DRALORIK	D25CRCW1206	
R118,R107	2	1206	RES 1Kohm	DRALORIK	D25001KFCS	
R112,R116,R117	3	1206	RES 20ohm	DRALORIK	D25020RFCS	
R134,R206	2	1206	RES 10ohm	ROEDERSTEIN	D25010RFCS	
R159,R162,R177,R180,R186, R187,R222,R223,R228,R230	10	603	RES 22R1ohm	ROEDERSTEIN	D1122R1FCS	
R164	1	603	RES 10Kohm	ROEDERSTEIN	D11010KFCS	Optional
R165	1	603	RES 1Kohm	DRALORIK	D11001KFCS	Optional
R175	1	1206	RES 47Kohm	ROEDERSTEIN	D25047KFCS	
R176	1	1206	RES 0R5ohm	ROEDERSTEIN	D250R50FCS	
R189,R188	2	603	RES 36R5ohm	ROEDERSTEIN	D1136R5FCS	
R203,R209	2	603	RES 10Kohm	ROEDERSTEIN	D11010KFCS	L2Cache
R207	1	1206	RES	ROEDERSTEIN	D2505K6FCS	Optional
R219	1	1206	RES	DRALORIK	D25-03KJ-S	Optional
R233,R232	2	603	RES 1Kohm	DRALORIK	D11001KFCS	L2Cache
R243,R244,R245,R249,R250, R251,R252,R253	8	1206	RES 1K5ohm	ROEDERSTEIN	D2501K5FCS	
SW1,SW2,SW3	3	8PIN_SMD	SWITCH DIP 4	GRAYHIL	90HBW04SR	
SW4	1	16PIN_SMD	SWITCH DIP 8	GRAYHIL	90HBW08S	
S1	1	KS12	PUSHBUTTON SPDT	C&K	KS12-R23-CQE	
S2	1	KS12	PUSHBUTTON SPDT	C&K	KS12-R21-CQE	
S3	1	KS12	PUSHBUTTON SPDT	C&K	KS12-R22-CQE	
U1	1	9P_TH	IC FIBER TRANS-CEIVER	HP	HFBR-5205	
U2	1	16P_SMT	IC 16P ETHERNET MAGNETICS	HALO	TG22-3506ND	
U3,U4	2	SSOP	IC RS232 TRANS-CEIVER	MOTOROLA	MC145583VFEL	
U5	1	128-Pin_PQFP	IC ATM PHY	PMC SIERA	PM5350-RC	
U6	1	64-Pin_PQFP	IC ETHERNET TRANS-CEIVER	LEVEL ONE	LXT970QC/AQC	

Table 7-9. MPC8266ADS-PCI Bill Of Materials

Reference	Qty	Footprint	Description	Manufacturer	Part Number	Note
U7,U20,U26,U27	4	TSSOP	IC 8BIT BUFFER	MOTOROLA	MC74LCX541DT	
U8,U11,U24,U31,U32,U33	6	14P_TSSOP_SMD	IC 4BIT BUFFER	MOTOROLA	MC74LCX125DT	
U9	1	SOT-23-5	IC VOLTAGE DETEC-TOR	SEIKO	S-80828ANMP-EDR-T2	
U10	1	48-Pin_TQFP	IC 64/32 CPLD	VANTIS	M4-64/32-7VC48	
U12	1	TBGA480	IC CPU	MOTOROLA	MPC8266	
	1	TBGA480	480 Pin Socket	E-TECH	BPW480-1265-29BA01H	
U13,U34,U38	3	TSSOP48	IC 16BIT BUFFER	MOTOROLA	MC74LCX16244DT	
U14	1	100PIN_TQFP	IC 128/64 CPLD	VANTIS	M4-128/64-7VC	
U15,U17	2	32-Pin_TQFP	IC LOW SKEW BUFFER	MOTOROLA	MPC947FA	
U16	1	TO220	IC 3V3 REGULATOR	MICREL	MIC29500-3.3BT	
U35,U18	2	241-Pin_PBGA	IC L2CACHE	MOTOROLA	MPC2605ZP66	L2Cache
U19	1	SO20W	IC 8BIT BUFFER	ON SEMICONDUCTOR	74ACT541DW	
U21	1	D2PAK_936	IC 3V3 REGULATOR	ON SEMICONDUCTOR	LM317D2T	
U23,U22	2	48-Pin_TQFP	IC 64/32 CPLD	VANTIS	M4-64/32-7VC48	L2Cache)
U25	1	TSSOP48	IC 16BIT TRANCEIVER	ON SEMICONDUCTOR	MC74LCX16245DT	
U28	1	DIMM_168_PIN	IC MEMORY	FUJITSU	SPDC2UV6482C-100T-S	
	1	DIMM_168_PIN	168PIN DIMM SOCKET	AMP	390040-6	
U29	1	PLCC_32	IC MEMORY	ATMEL	AT28HC64B-70JC	
	1	32PIN PLCC	32PIN PLCC SOCKET	AMP	822273-1	
U30	1	80PIN_SIMM	IC MEMORY	SMART MODULAR TECHNOLOGIES	SM73228XG1JHBG0	
	1	80PIN SIMM	80PIN SIMM SOCKET	AMP	822021-5	
U37,U36	2	SSOP48	IC 16BIT LATCH	PHILIPS	74ALVT16373DL	
U40,U39	2	SSOP48	IC 16BIT TRANS-CEIVER	PHILIPS	74ALVT16245DL	
X1	1	4P_TH	19.44MHz - 5V Clock Oscilator	COMCLOK	CM42AH	
X2	1	4P_TH	66MHz - 3.3V Clock Oscilator	M-TRON	M3H16FCD-3V3-66MHz	
	1	8P_SMD	8P_SMD_Socket	PRECIDIP	1109330841105	
Y1	1	XTAL-SMD	CRYSTAL	EPSON	MA505 +/-50PPM 25MHZ	

Digital DNA from Motorola

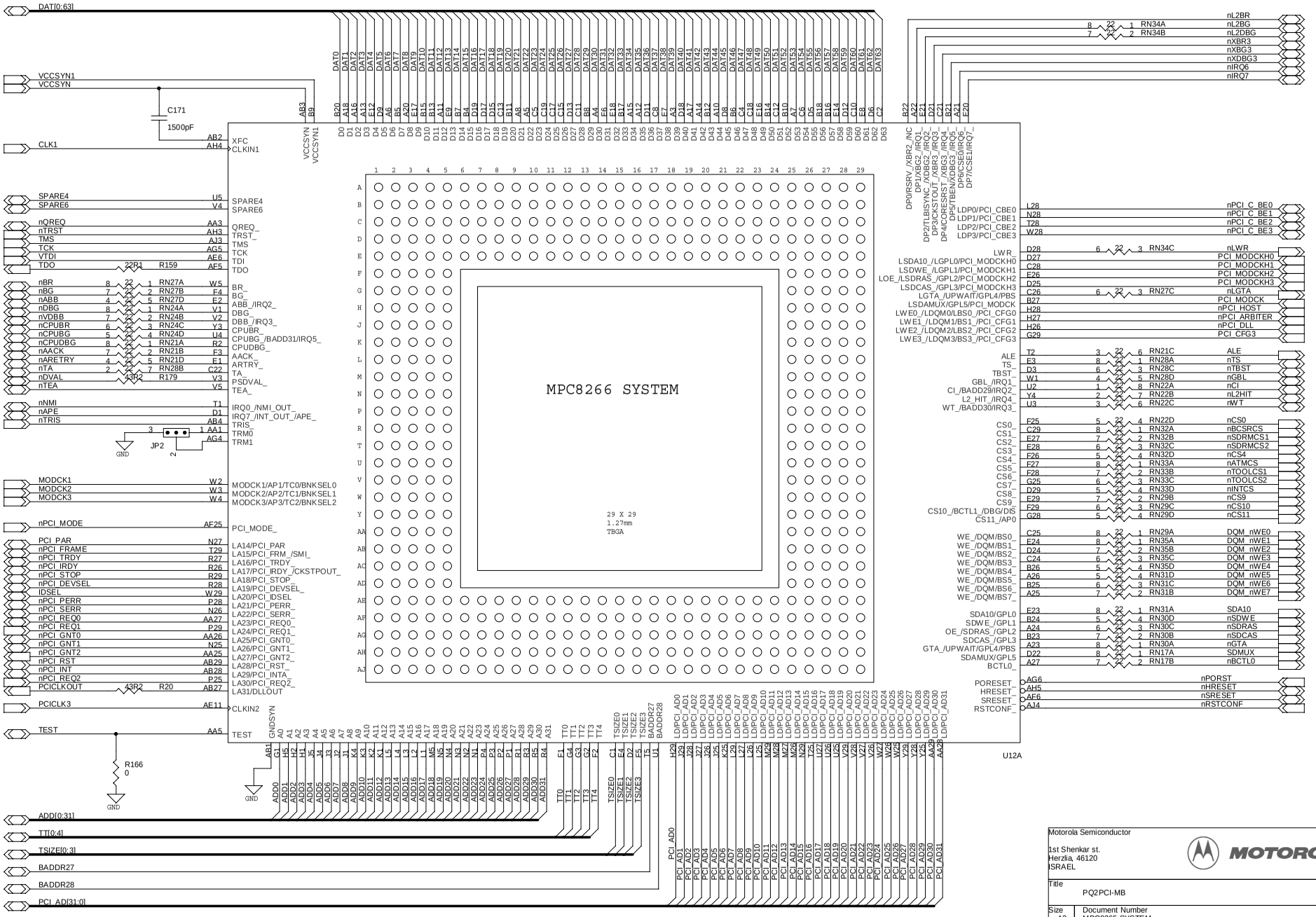
PowerQUICC II*

MPC8266 ADS - PCI

MPC8266 PCI MotherBoard

Table of Contents

Page	Description
	Title Page
1	MPC8266 System
2	System Buffers
3	Expansion Buffers
4	BCSRs
5	FLASH and EEPROM Memories
6	SDRAM Memory
7	L2Cache
8	MPC8266 CPM
9	ATM
10	Fast-Ethernet
11	RS232
12	Pull-Up / Pull-Down Resistors
13	MPC8266 Power
14	PCI Connectors
15	PCI Connectors
16	PCI Interrupt Controller
17	Expansion Connectors (2 x 128 pin)
18	Logic Analyzer Connectors - Mictors
19	Power
20	JTAG, General Parts



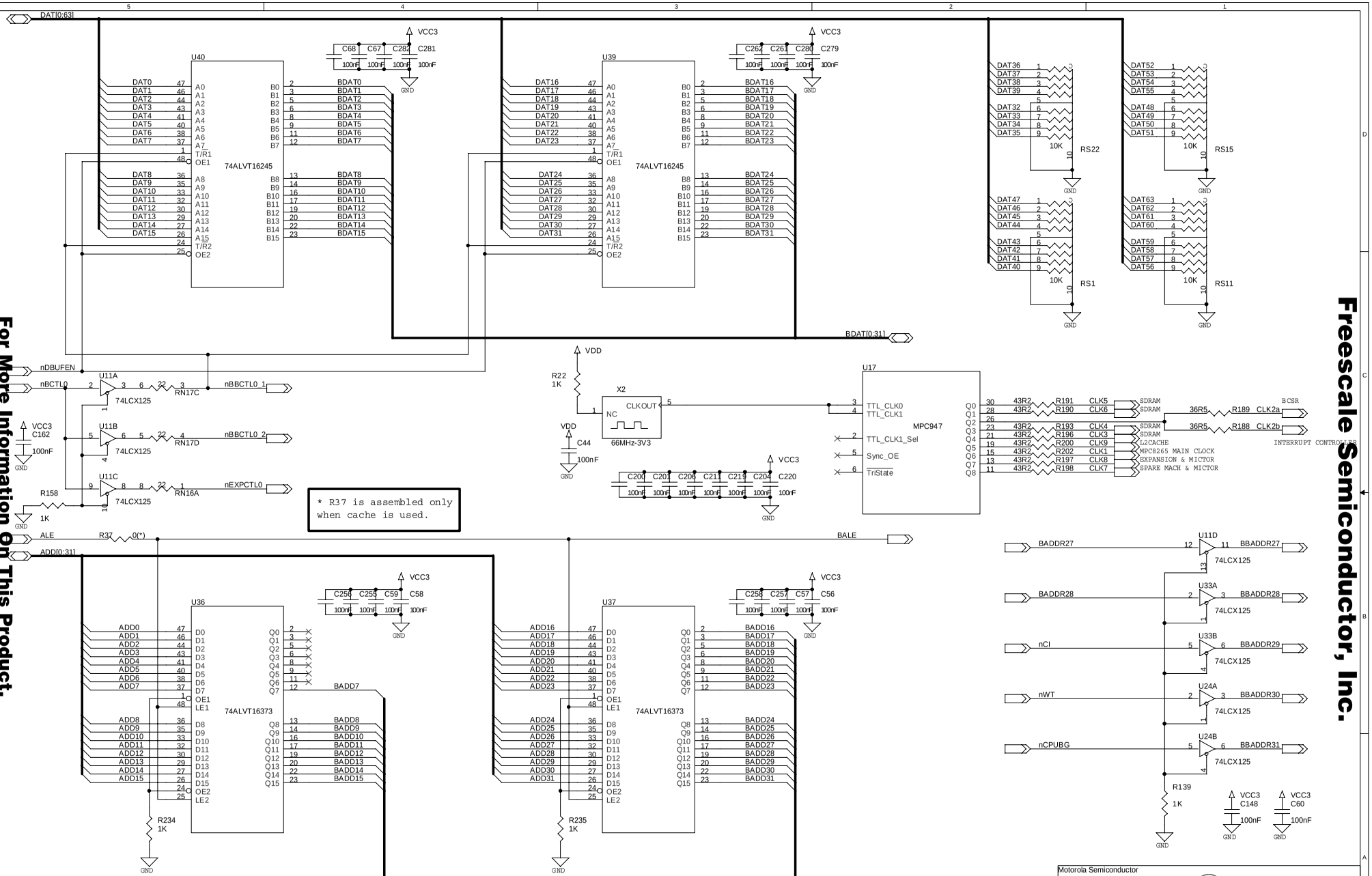
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title: PQ2PCI-MB

Size	Document Number	Rev
A3	MPC8265 SYSTEM	ENG

Date: Tuesday, November 27, 2001 Sheet 1 of 20



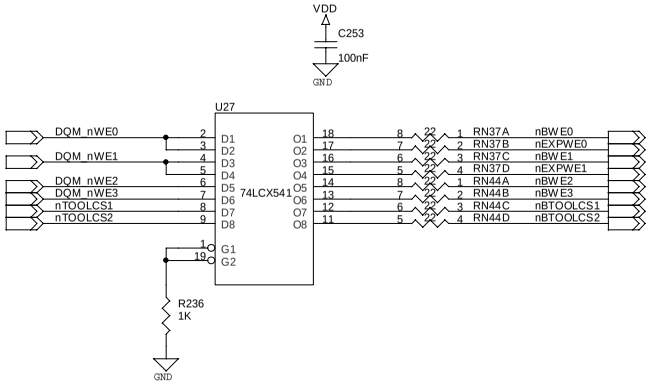
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

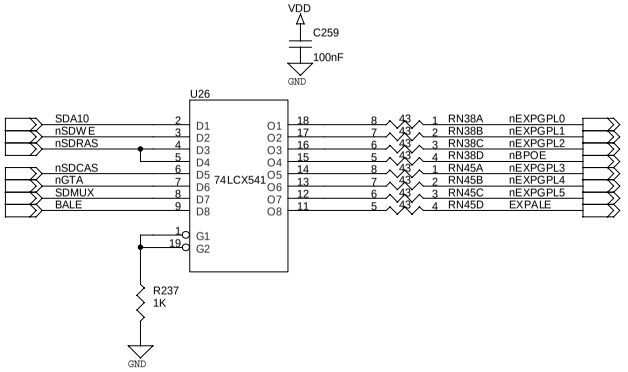
MOTOROLA

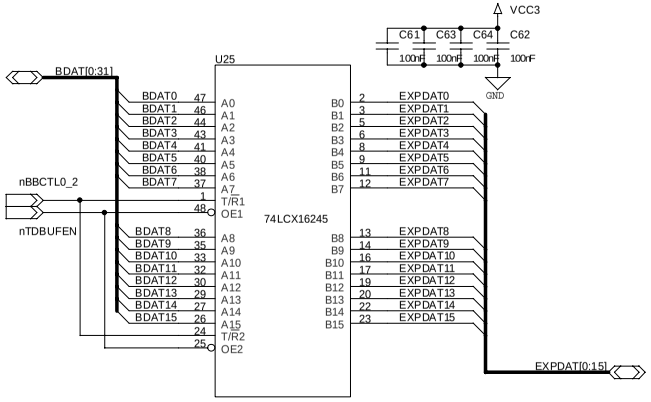
Title: PQ2PCI-MB

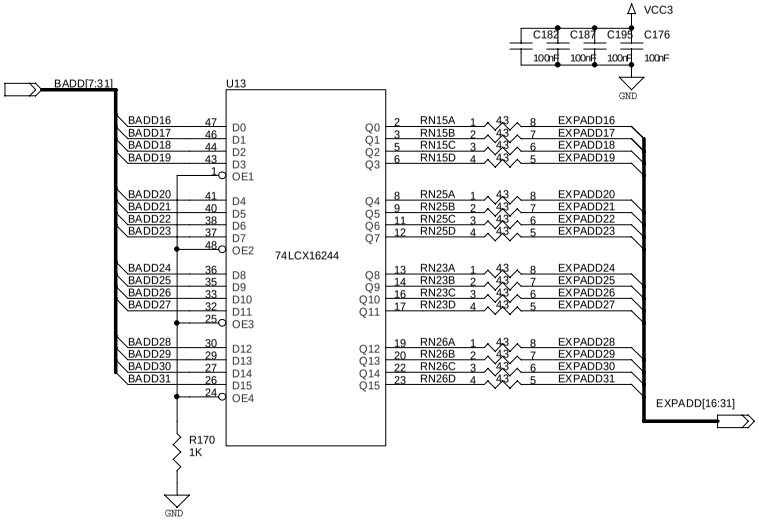
Size: A3 | Document Number: SYSTEM BUFFERS | Rev: ENG

Date: Tuesday, November 27, 2001 | Sheet 2 of 20





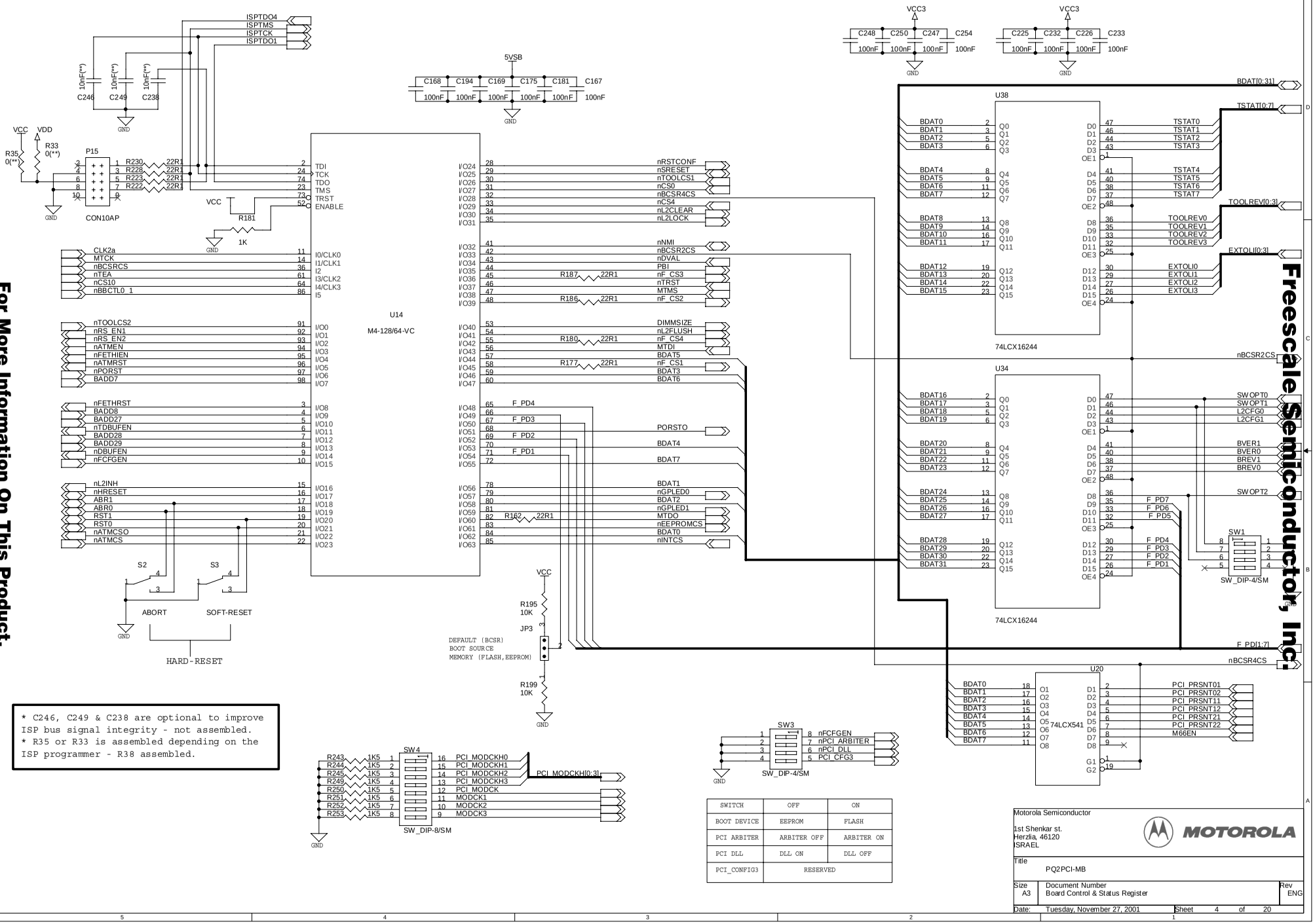




Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title	PQ2PCI-MB	
Size A3	Document Number EXPANSION BUFFERS	Rev ENG
Date: Tuesday, November 27, 2001	Sheet 3 of 20	



* C246, C249 & C238 are optional to improve ISP bus signal integrity - not assembled.
* R35 or R33 is assembled depending on the ISP programmer - R38 assembled.

SWITCH	OFF	ON
BOOT DEVICE	EEPROM	FLASH
PCI ARBITER	ARBITER OFF	ARBITER ON
PCI DLL	DLL ON	DLL OFF
PCI_CONFIG3	RESERVED	

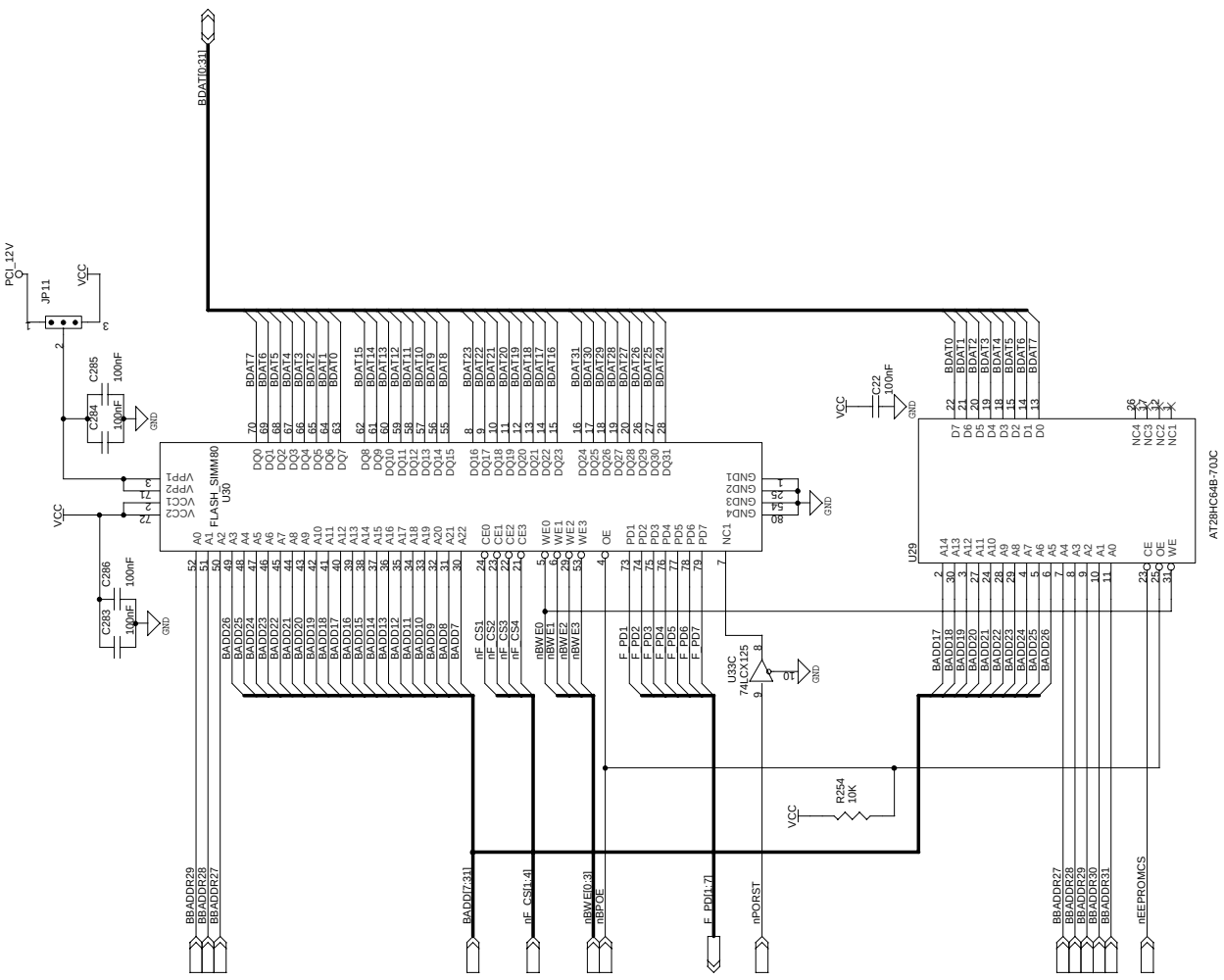
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

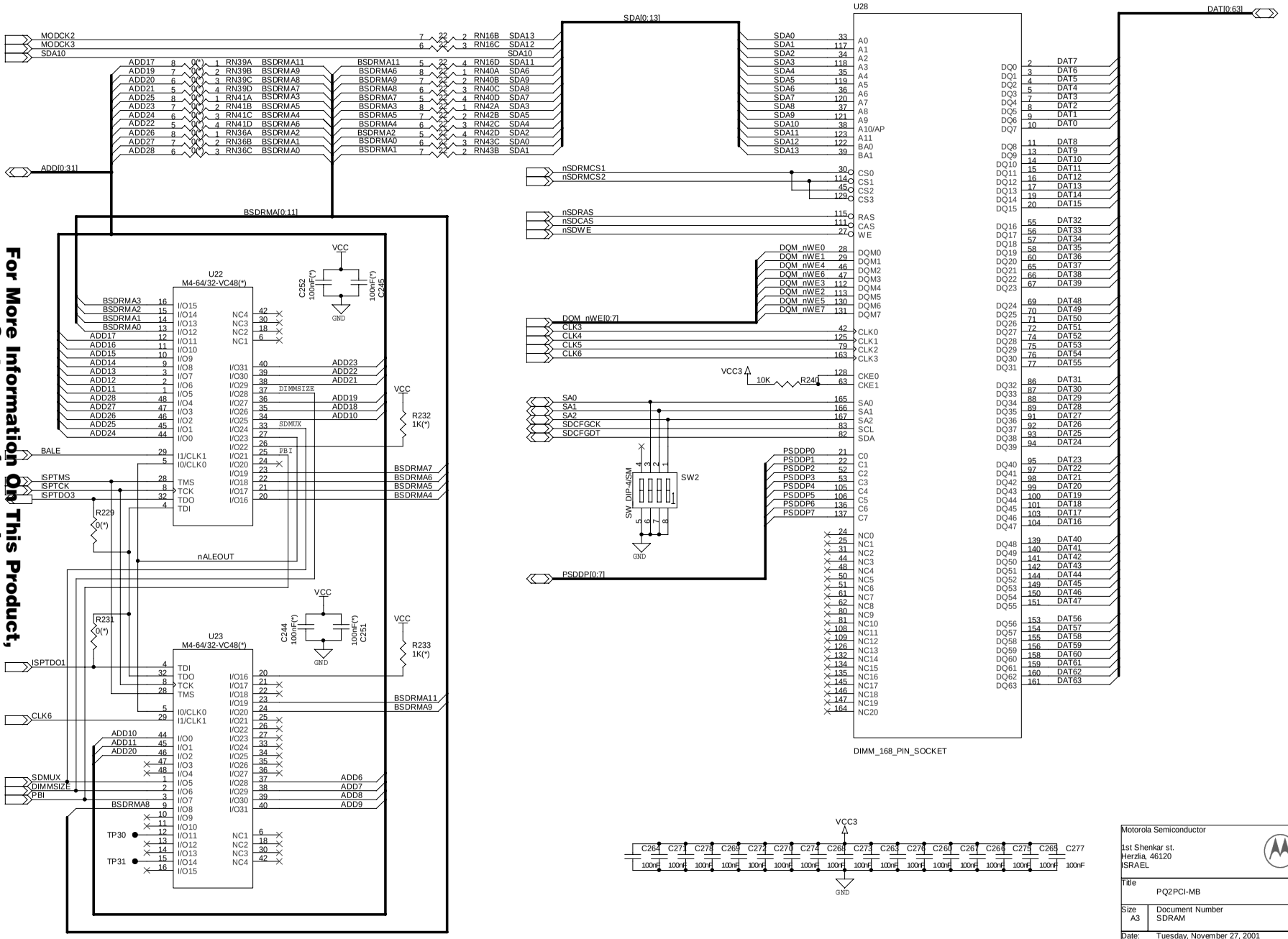
Title: PQ2PCI-MB

Size	Document Number	Rev
A3	Board Control & Status Register	ENG

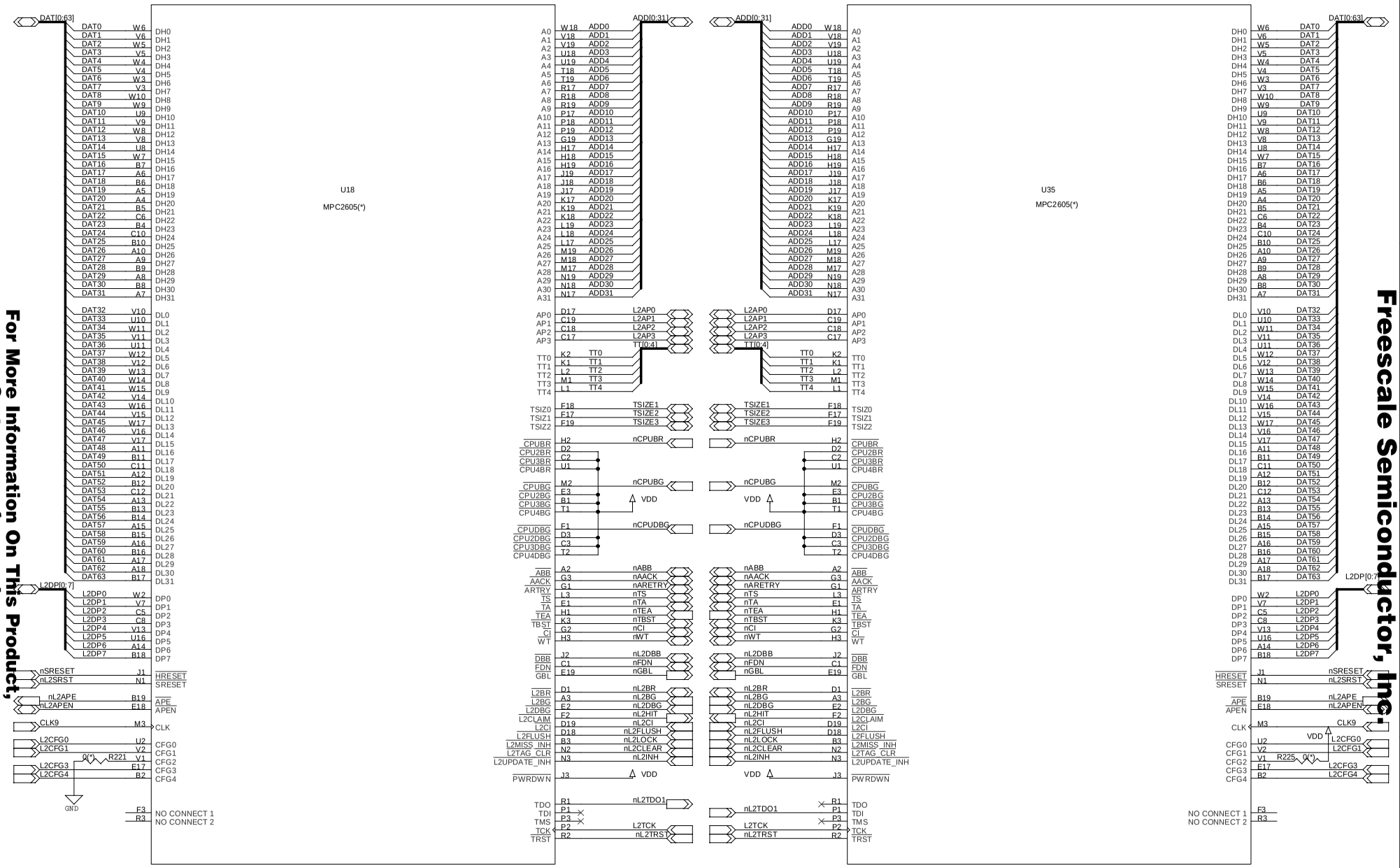
Date: Tuesday, November 27, 2001 Sheet 4 of 20

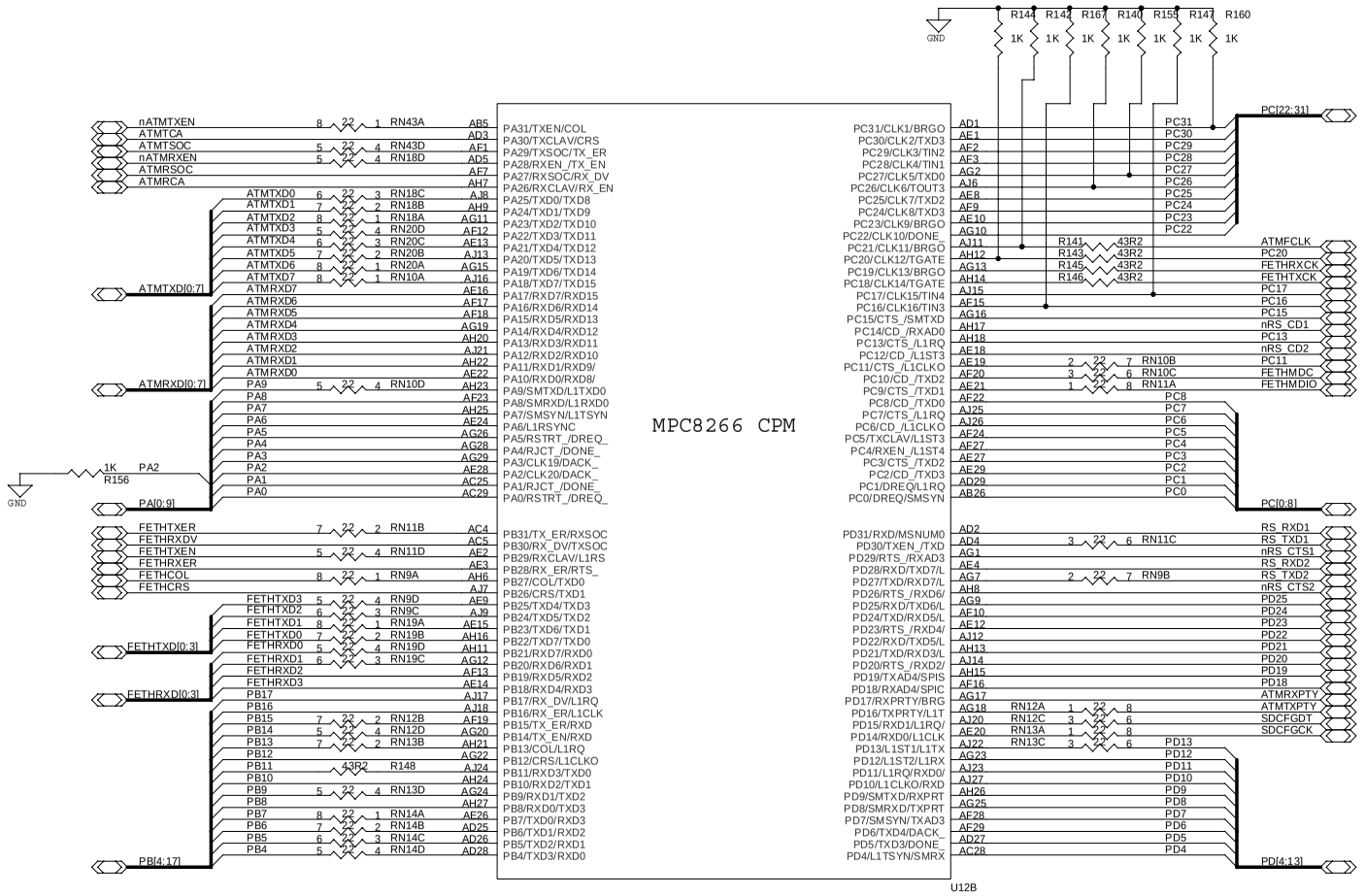


* RN39, RN41, RN36, R229 & R231 are assembled if cache is not used - factory assembled.
* C244, C251, C245, C252, U22 & U23 are assembled if cache is used - not assembled.
* R232 and R233 are to comply with the ICT requirements.



Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: SDRAM	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 6 of 20	





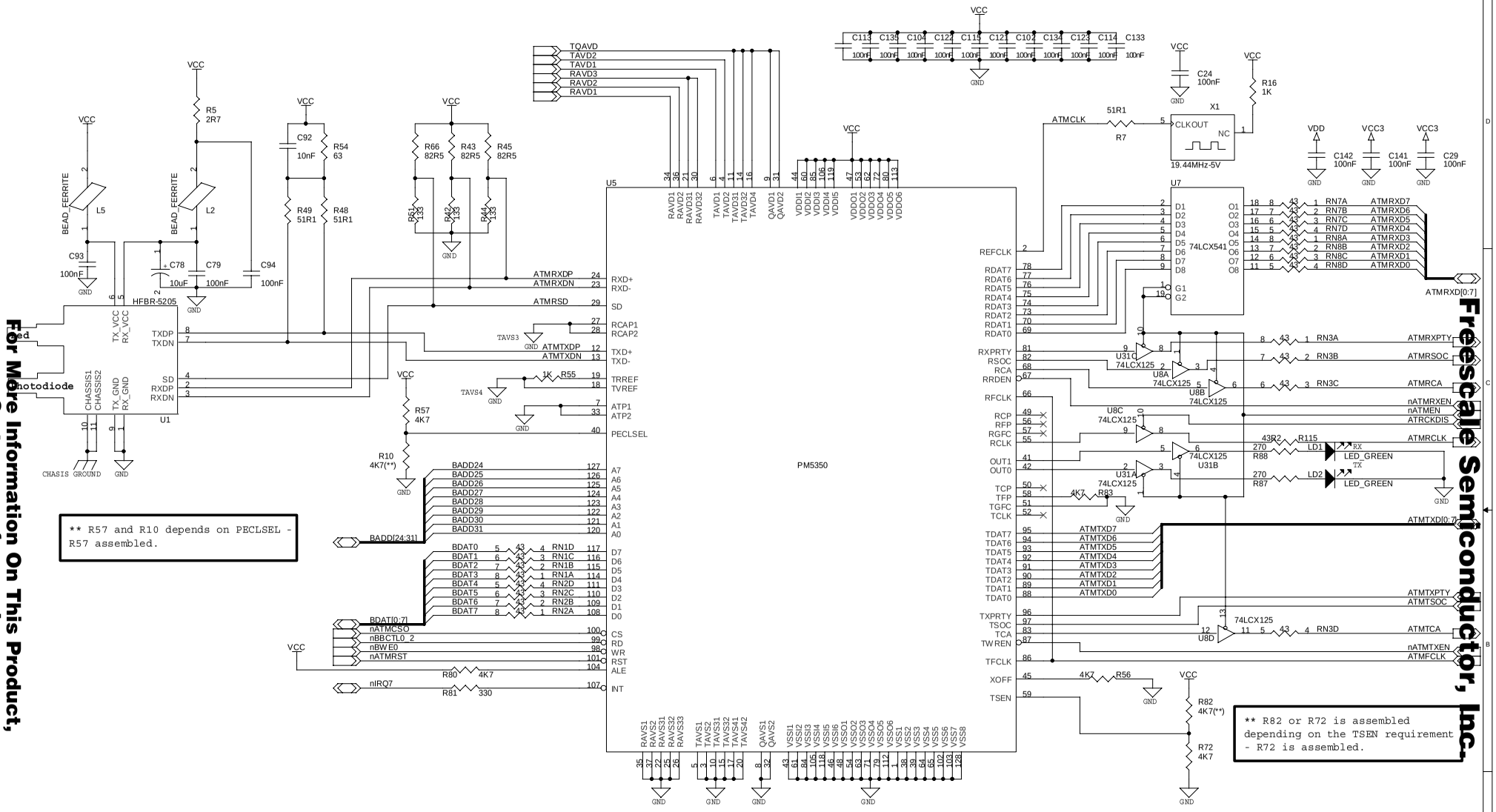
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

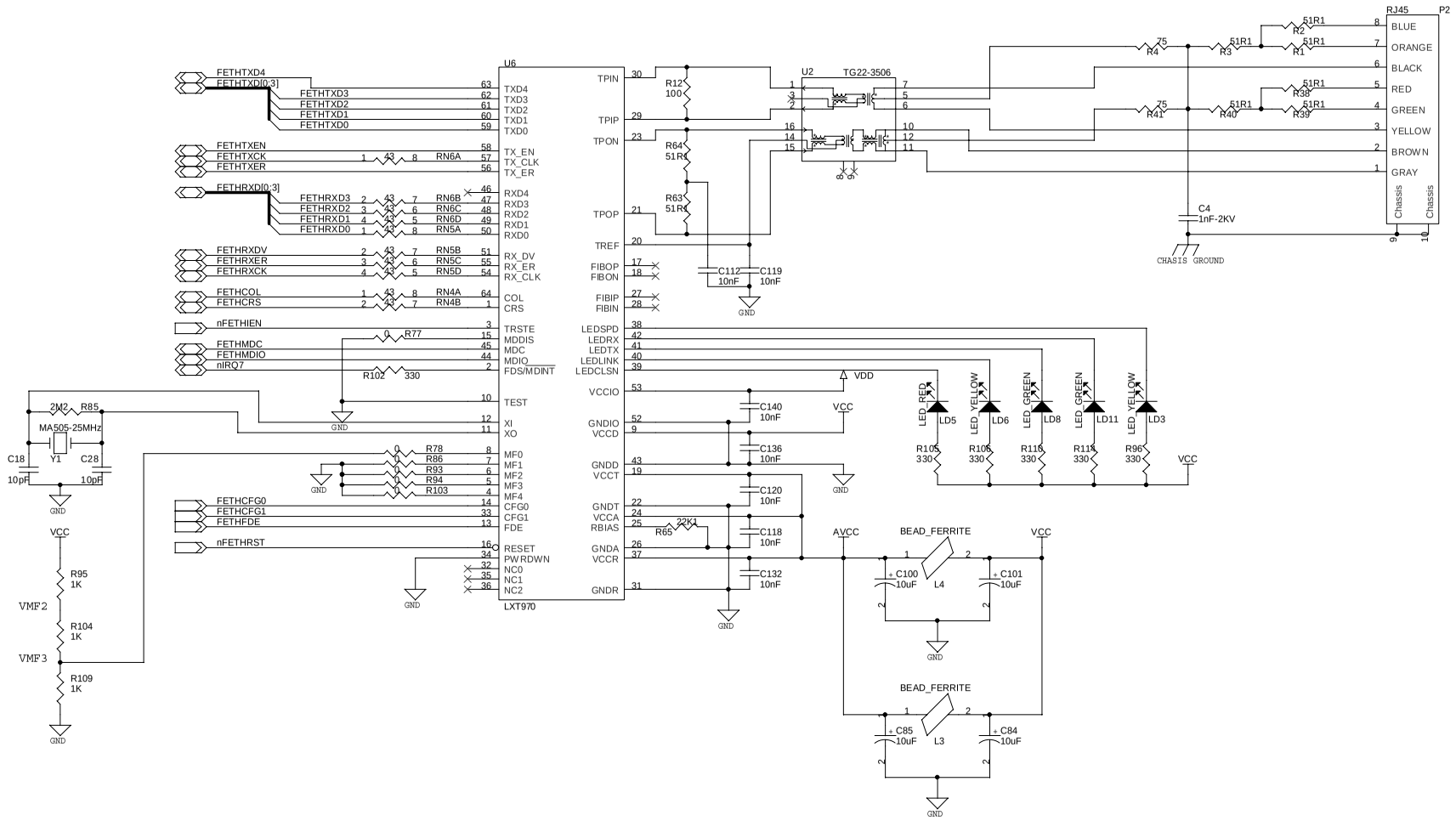
Title: PQ2PCI-MB

Size	Document Number	Rev
A3	MPC8265 - CPM	ENG

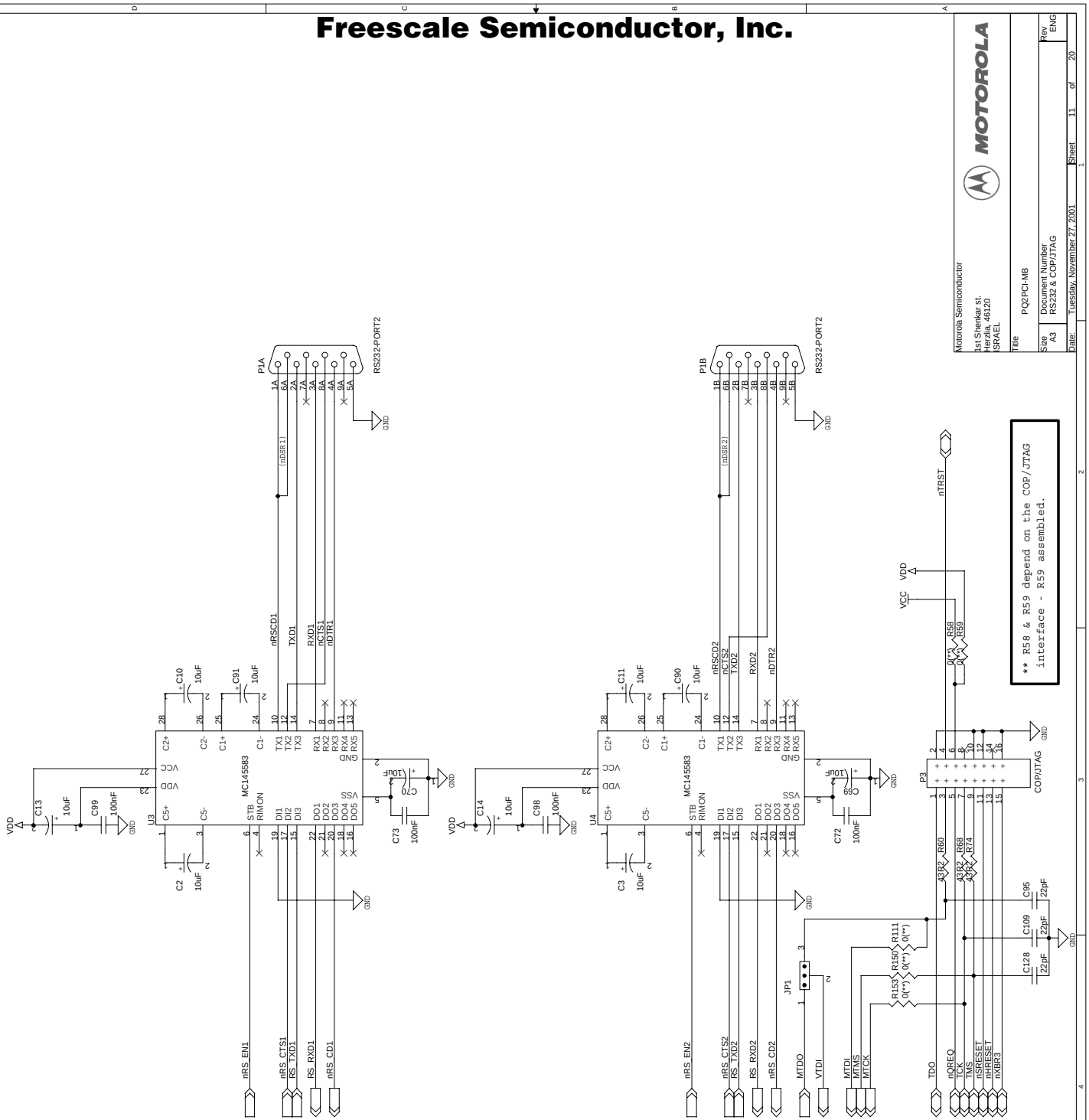
Date: Tuesday, November 27, 2001 Sheet 8 of 20



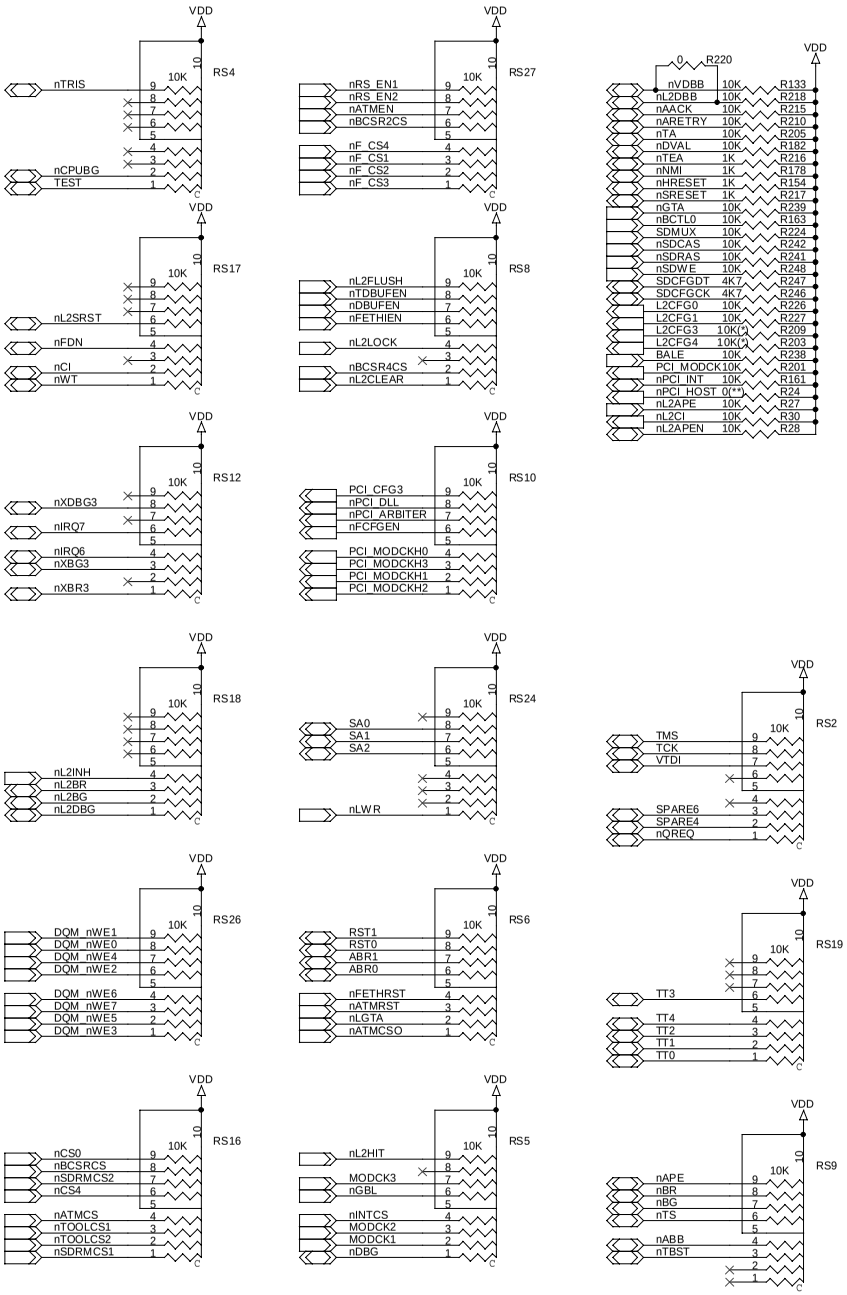
Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: ATM PHY	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 9 of 20	



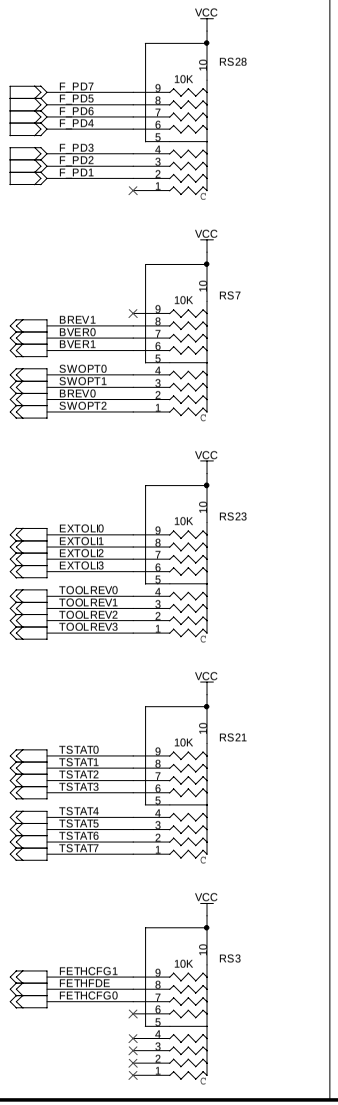
Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: FAST ETHERNET	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 10 of 20	



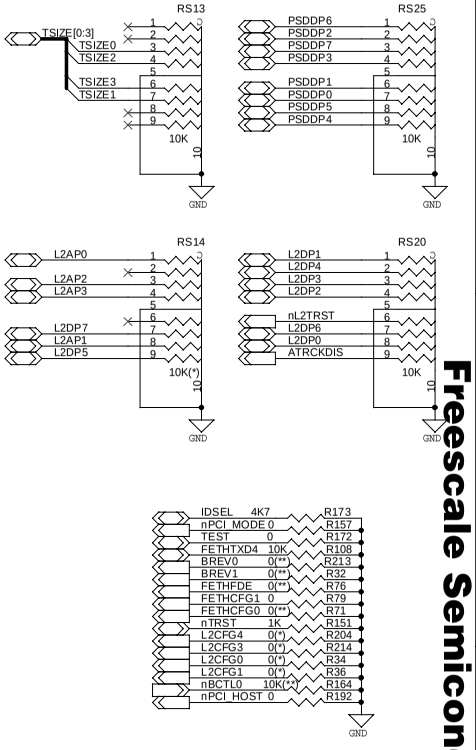
3.3V PULL-UPS



5V PULL-UPS



PULL-DOWNS



** R24 is an option - not assembled in the factory

** R71, R76 determine the configuration of the Fast Ethernet Tranceiver - not assembled in the factory.

* R34, R36 determine the amount of L2cache on board.

* R209, R203, RS14, R204, R214 are assembled if cache is used.

** R213 & R32 determine the boards' revision as follows:

R213	R32	Revision
yes	yes	ENG
no	yes	PILOT
yes	no	A
no	no	Reserved

** R164 depends on the functionality of nBCTL0 signal - not assembled in the factory.

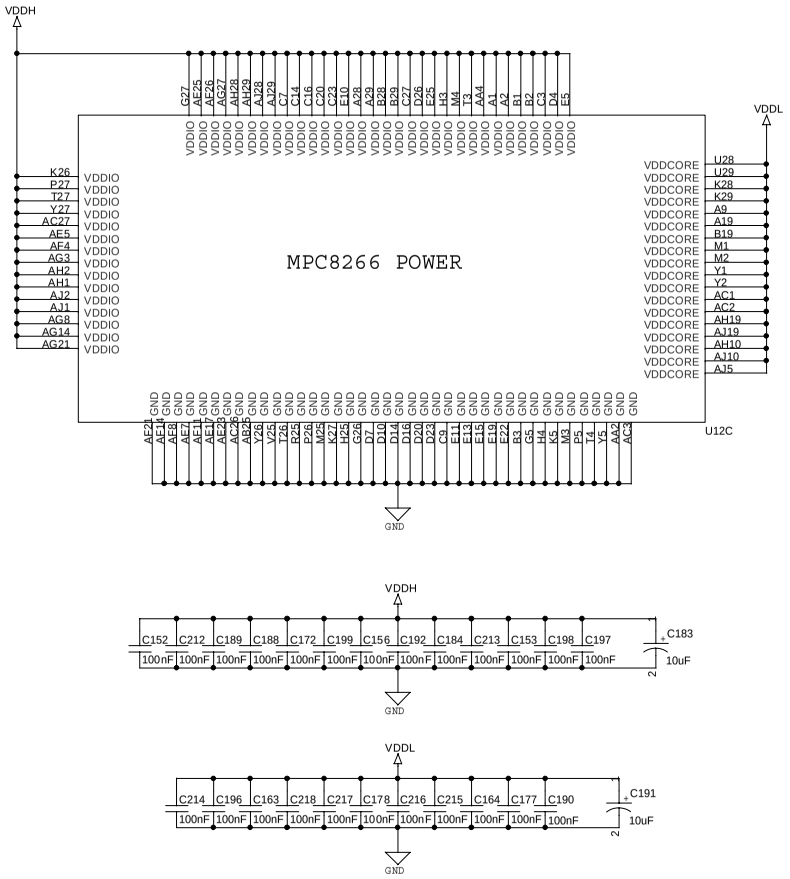
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title: PQ2PCI-MB

Size	Document Number	Rev
A3	PUL UPS/DOWNS	ENG

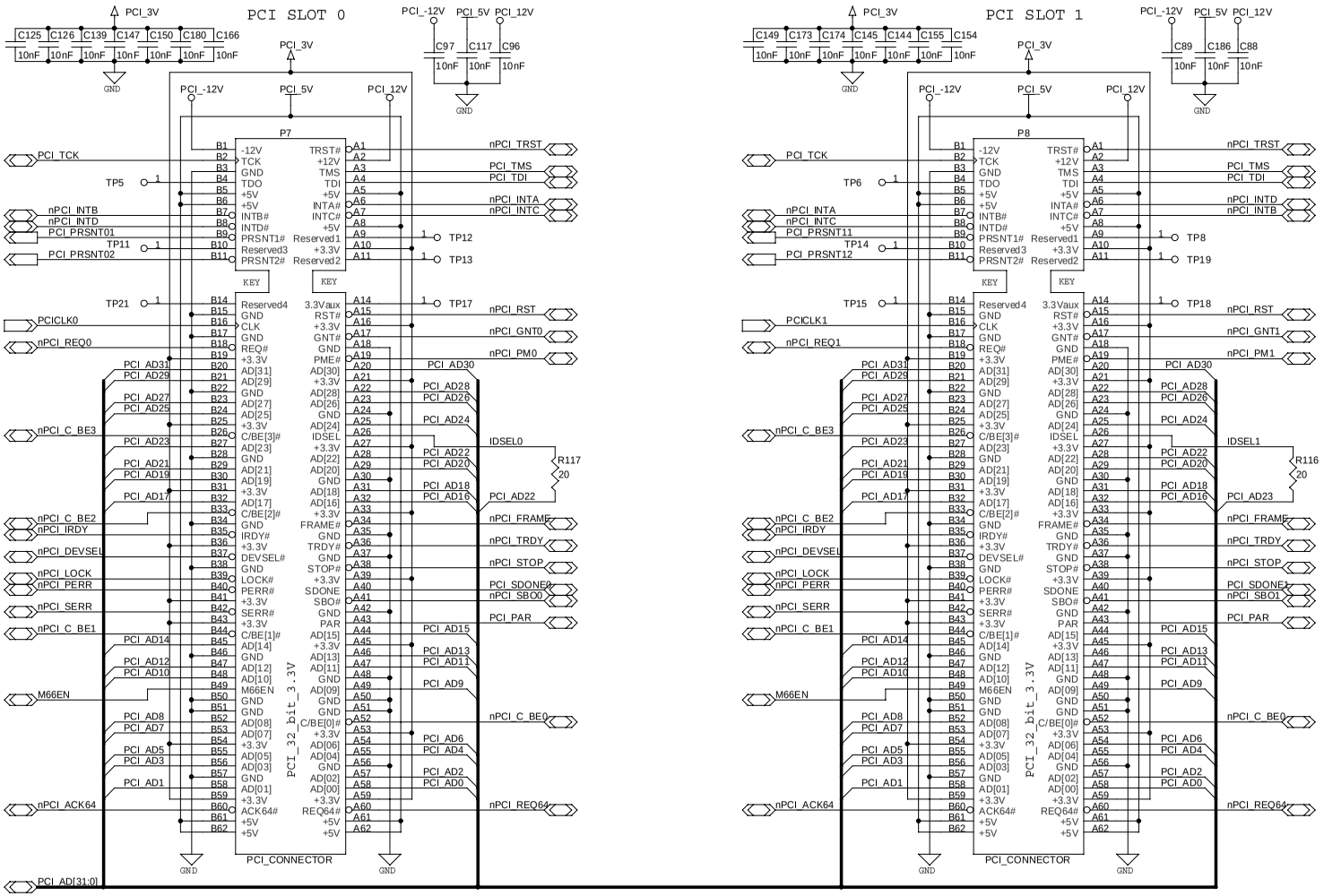
Date: Tuesday, November 27, 2001 Sheet 12 of 20



Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

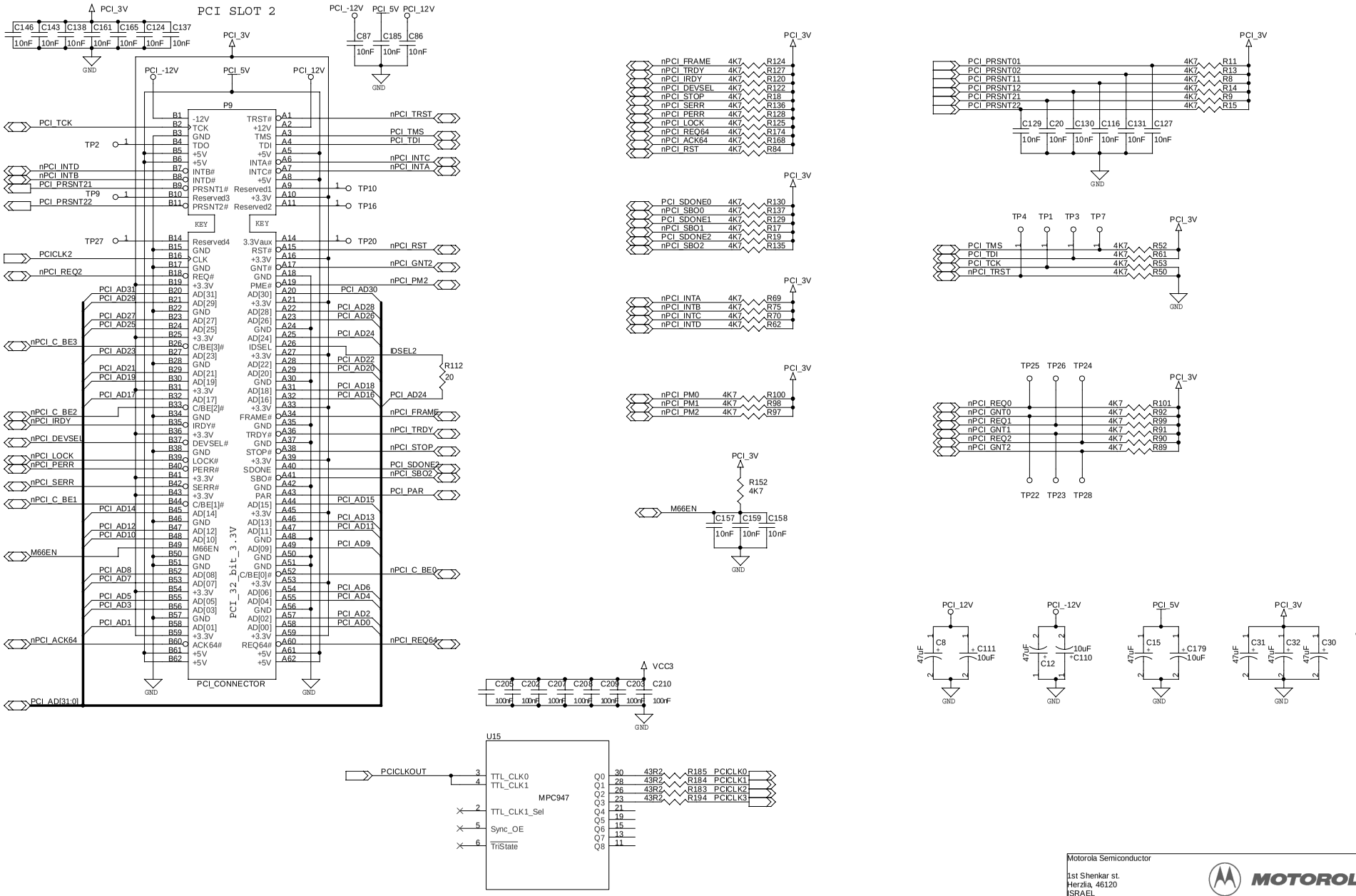
Title	PQ2PCI-MB	
Size A3	Document Number MPC8265 - POWER	Rev ENG
Date: Tuesday, November 27, 2001	Sheet 13 of 20	



Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

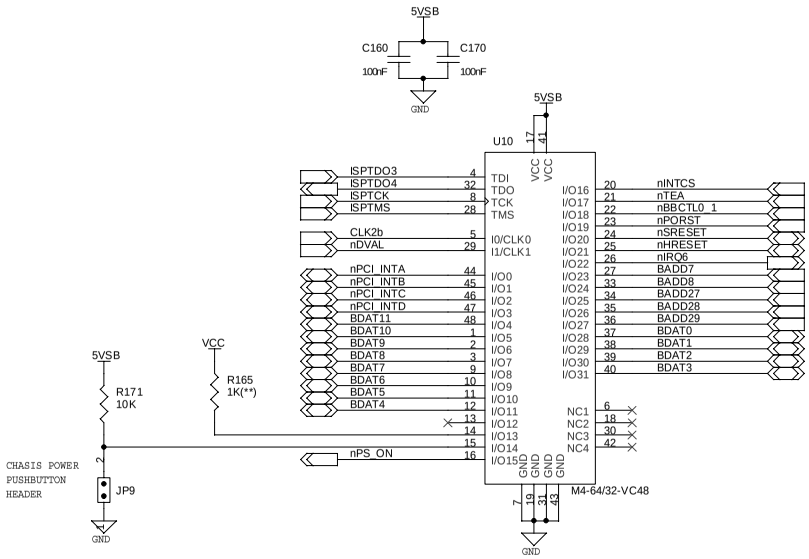
Title	PQ2PCI-MB	
Size A3	Document Number PCI CONNECTORS - 1	Rev ENG
Date: Tuesday, November 27, 2001	Sheet 14 of 20	



For More Information On This Product,
Go to: www.freescale.com

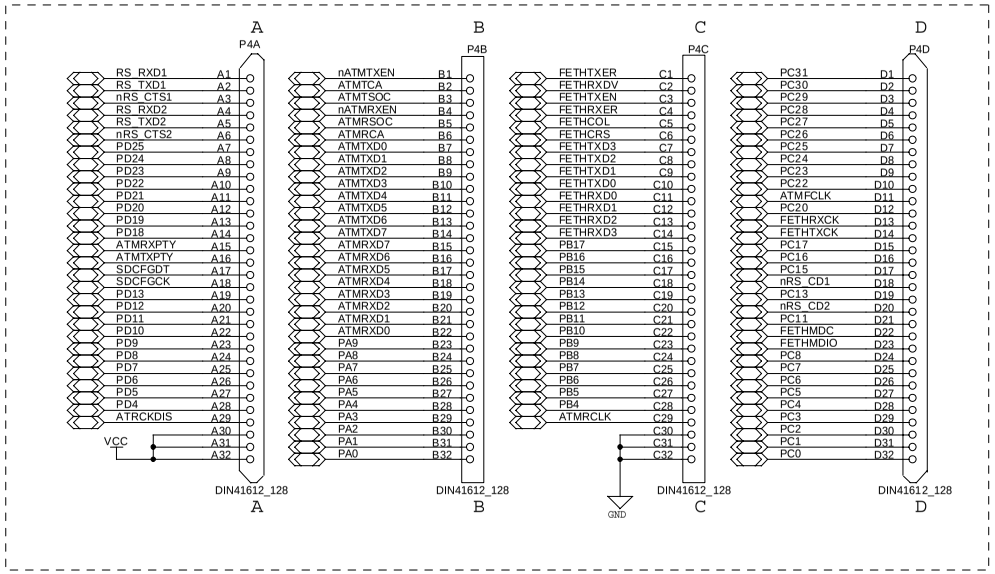
Freescale Semiconductor, Inc.

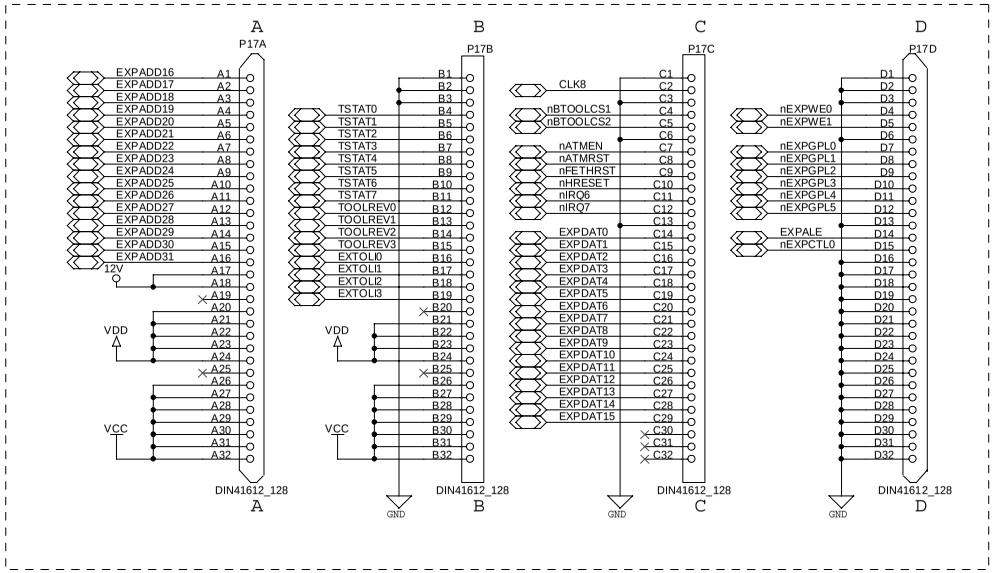
Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: PCI CONNECTORS - 2	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 15 of 20	

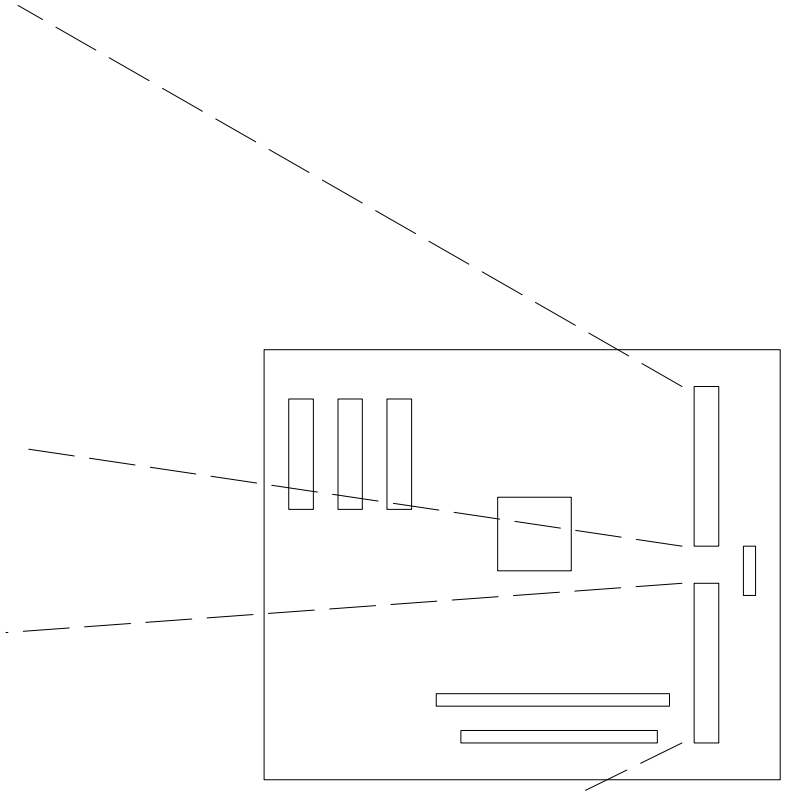


** R165 is to comply with the ICT requirements.

Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: PCI INTERRUPT CONTROLLER	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 16 of 20	



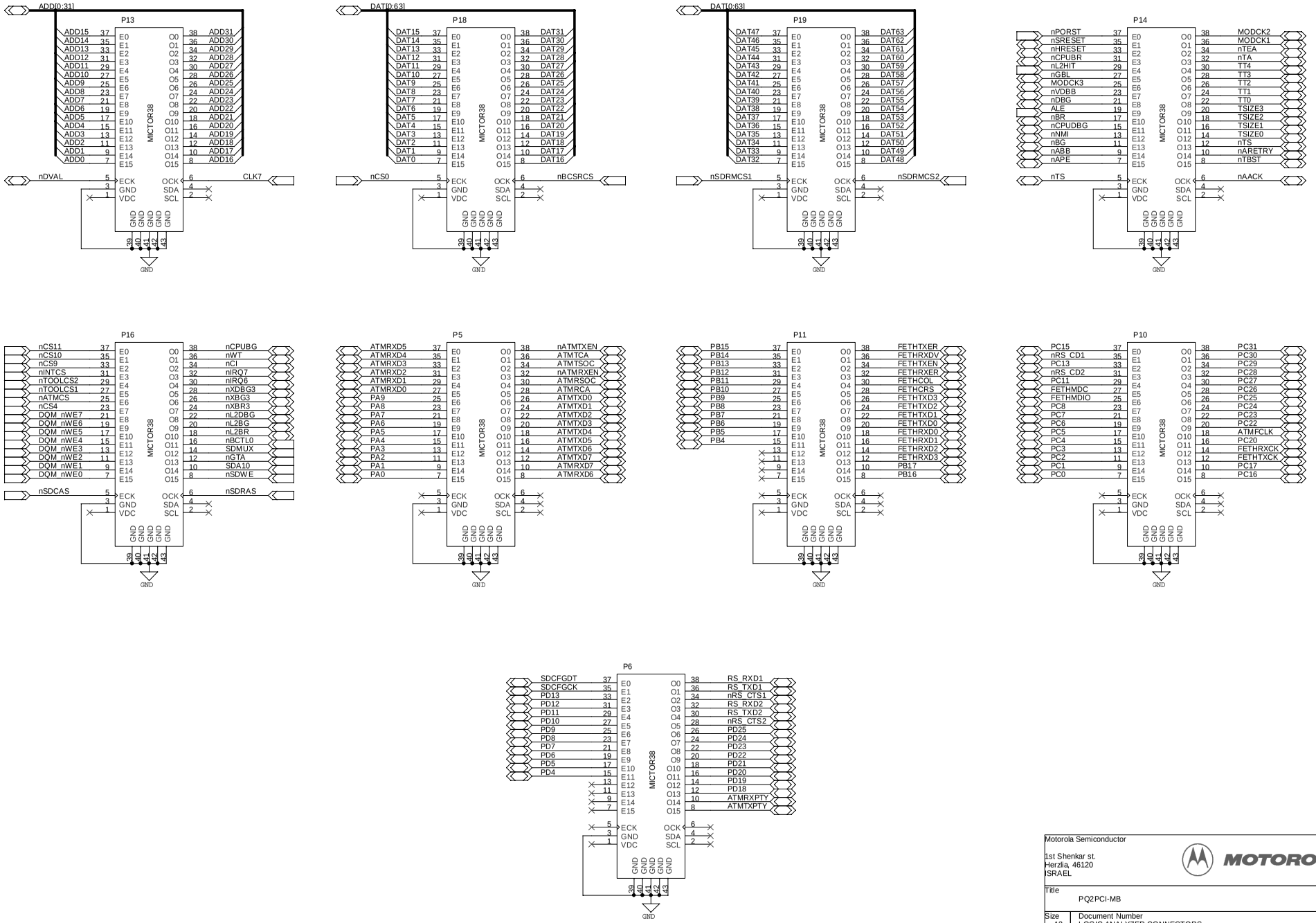




Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title	PQ2PCI-MB	
Size A3	Document Number EXPANSION CONNECTORS	Rev ENG
Date: Tuesday, November 27, 2001	Sheet 17 of 20	



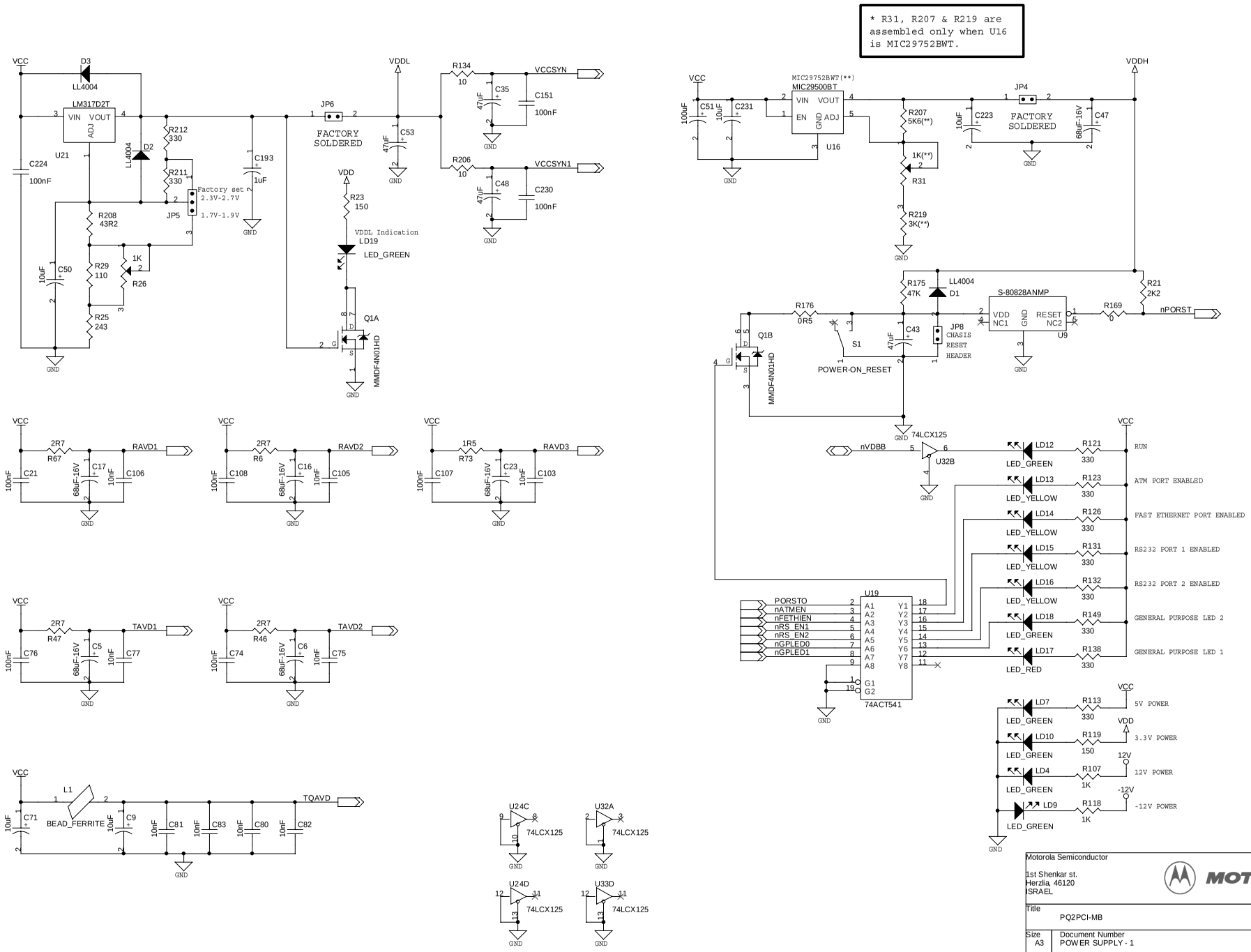
Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title: PQ2PCI-MB

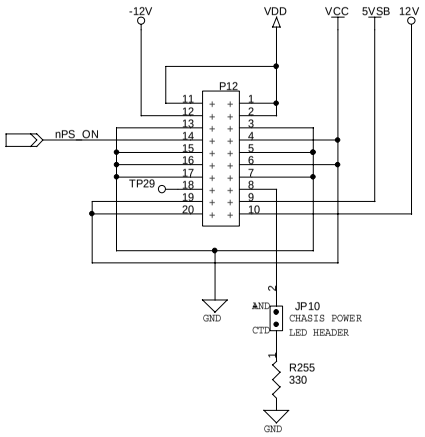
Size: A3 | Document Number: LOGIC ANALYZER CONNECTORS | Rev: ENG

Date: Tuesday, November 27, 2001 | Sheet 18 of 20

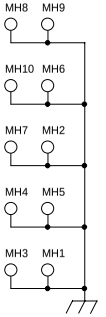


Motorola Semiconductor 1st Shenkar st. Herzlia, 46120 ISRAEL	MOTOROLA	
Title: PQ2PCI-MB		
Size: A3	Document Number: POWER SUPPLY - 1	Rev: ENG
Date: Tuesday, November 27, 2001	Sheet 19 of 20	

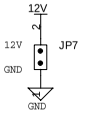
ATX_Power_Connector



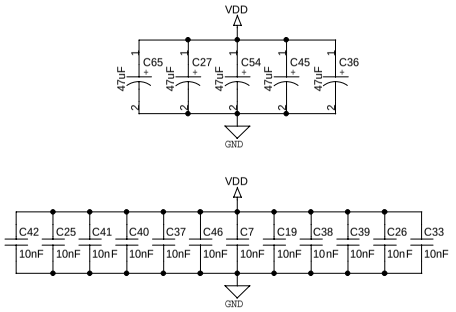
ATX CHASIS
MOUNTING HOLES

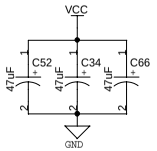


FAN POWER

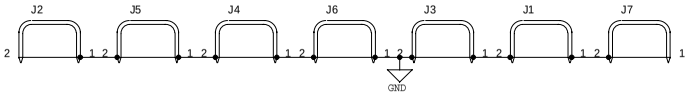


GENERAL BULK & HIGH
FREQUENCY CAPACITORS



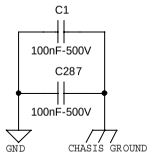


MAIN GROUND-BRIDGES



POWER RAILS
DESIGNATIONS

MAIN/CHASIS GROUND
CONNECTION



Motorola Semiconductor
1st Shenkar st.
Herzlia, 46120
ISRAEL

MOTOROLA

Title	PQ2PCI-MB	
Size A3	Document Number POWER SUPPLY - 2	Rev ENG
Date: Tuesday, November 27, 2001	Sheet 20 of 20	