User's Manual

MPC86XADS
Version-A
January 14, 2003

MOTOROLA
intelligence everywhere™

digital dna™

MPC86XADS

User's Manual

Important Notice to Users

While every effort has been made to ensure the accuracy of all information in this document, Motorola assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident, or any other cause. Motorola further assumes no liability arising out of the application or use of any information, product, or system described herein: nor any liability for incidental or consequential damages arising from the use of this document. Motorola disclaims all warranties regarding the information contained herein, whether expressed, implied, or statutory, *including implied warranties of merchantability or fitness for a particular purpose*. Motorola makes no representation that the interconnection of products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting or license to make, use or sell equipment constructed in accordance with this description.

Trademarks

This document includes these trademarks:

Motorola and the Motorola logo are registered trademarks of Motorola, Inc.

Windows is a registered trademark of Microsoft Corporation in the U.S. and other countries.

Intel is a registered trademark of Intel Corporation.

Motorola, Inc., is an Equal Opportunity / Affirmative Action Employer.

For an electronic copy of this book, visit Motorola's web site at http://e-www.motorola.com/

1 - General Information

1•1 Introduction

This document is an operation guide for the MPC86xADS board. It can contain the following parts: XPC855T / MPC855T, XPC857T / MPC857T, XPC8860X / MPC860X, XPC862X / MPC862X, XPC866X / MPC866X. It contains operational, functional and general information about the ADS. The MPC86xADS is meant to serve as a platform for s/w and h/w development around the MPC86X family processors. Using its on-board resources and its associated debugger, a developer is able to download his code, run it, set breakpoints, display memory and registers and connect his own proprietary h/w via the expansion connectors, to be incorporated to a desired system with the MPC86x processor.
This board is compatible with the MPC8xxFADS for SW point of view.

This board could also be used as a demonstration tool, i.e., application s/w may be burned[A] into its flash memory and ran in exhibitions etc.'.

1•2 MPC86x Family Support

The MPC86xADS supports the following MPC8xx family members:

- XPC855T / MPC855T
- XPC857T / MPC857T
- XPC8860DE / MPC860DE
- XPC860DP / MPC860DP
- XPC860EN / MPC860EN
- XPC860P / MPC860P
- XPC860SR / MPC860SR
- XPC860T / MPC860T
- XPC862DT / MPC862DT
- XPC862SR / MPC862SR
- XPC862T / MPC862T
- XPC859X / MPC859X
- XPC866X / MPC866X

A. Either on or off-board.

1•3 Abbreviations' List

- ADS - the MPC86xADS, the subject of this document.
- UPM - User Programmable Machine
- GPCM - General Purpose Chip-select Machine
- GPL - General Purpose Line (associated with the UPM)
- I/R - Infra-Red
- BCSR - Board Control & Status Register.
- ZIF - Zero Input Force
- BGA - Ball Grid Array
- SIMM - Single In-line Memory Module

1•4 Related Documentation

- MPC86x User's Manuals.
- ADI Board Specification.
- Motorola 10BaseT MC68160 phy.
- LSI 10/100BaseT 80225 phy.
- Infineon E1/T1 PEB2256 Framer.
- NEC ATM 155Mhz uPD98404 phy
- *idt ATM 25Mhz* IDT77V107 phy

1•5 SPECIFICATIONS

The MPC86xADS specifications are given in TABLE 1-1.

TABLE 1-1. MPC86xADS Specifications

CHARACTERISTICS	*SPECIFICATIONS*
Power requirements (no other boards attached)	+5Vdc @ 1.4 A (typical), 3 A (maximum) +12Vdc - @1A.
Microprocessor	MPC86x running upto @ 75 MHz Bus Speed
Addressing Total address range: Flash Memory Dynamic RAM optional not populated. Synchronous DRAM	 4 GigaBytes 2 MByte, 32 bits wide expandable to 8 MBytes 4 MByte, 32 bits wide EDO SIMM, Optional Support for up to 32 MByte, EDO or FPM SIMM 8 MBytes, SDRAM.
Operating temperature	0^{O}C - 30^{O}C
Storage temperature	-25^{O}C to 85^{O}C
Relative humidity	5% to 90% (non-condensing)
Dimensions: Length Width Thickness	 9.173" (233 mm) 6.3" (160 mm) 0.063" (1.6 mm)

1•6 MPC86xADS Features

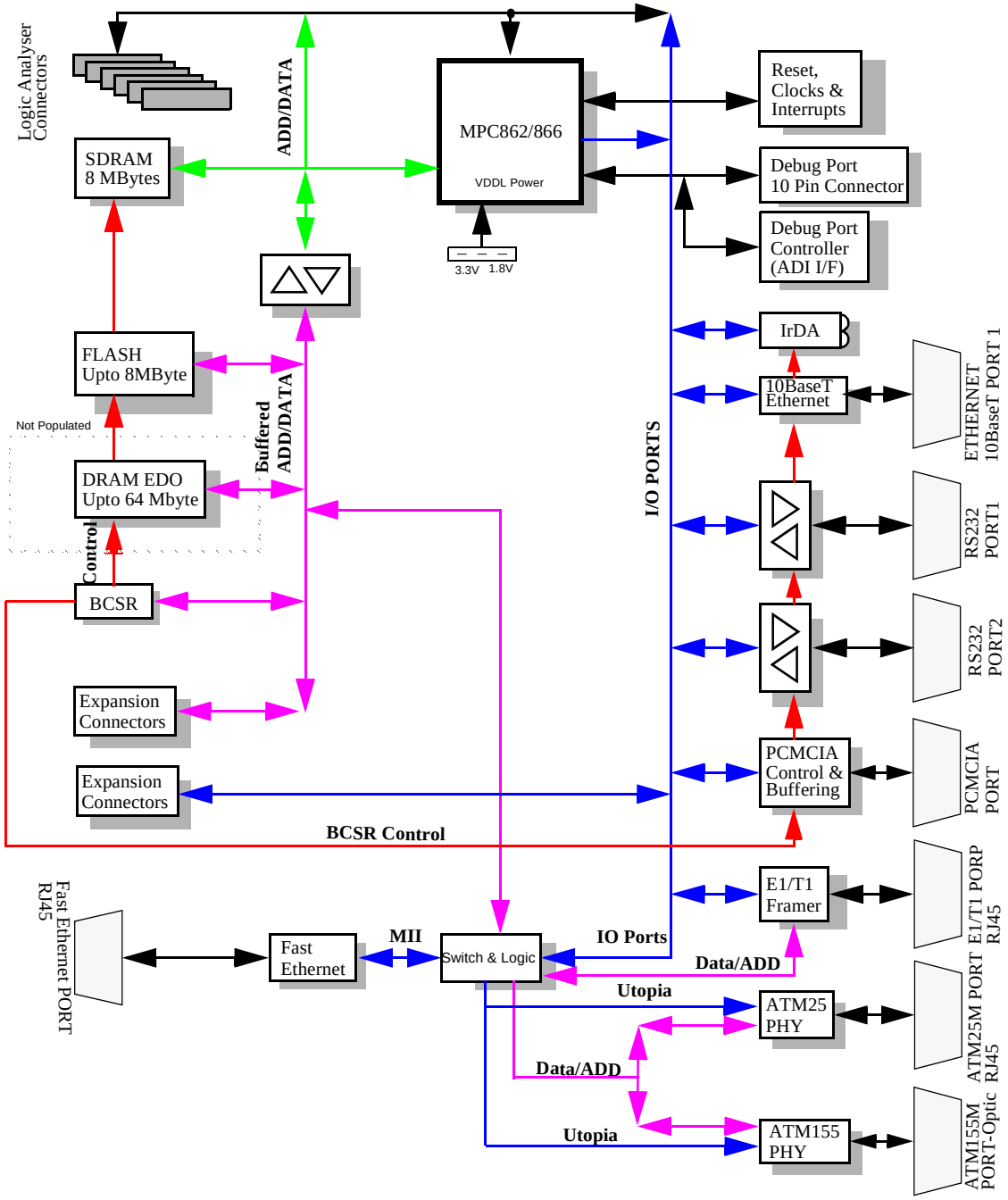
- ***Compatible with the old MPC86xADS Board.***
- MPC862/866, running upto 75 MHz bus frequency, mounted on ZIF BGA socket.
- 8 MByte, Unbuffered, Synchronous Dram On-Board.
- 4 MByte EDO 60nsec delay DRAM SIMM. Support for 4 - 32 MByte FPM or EDO Dram SIMM, with Automatic Dram SIMM identification. 16 Bit Data-Bus Width Support. EDO DRAM will not be populated on board. it is optional.
- 2 MByte Flash SIMM. Support for upto 8 MByte, 5V or 12V Programmable, with Automatic Flash SIMM identification. Can be change up to 8MByte.
- Dual RS232 port with Low-Power Option per each port.
- T1/E1 connected to TDMB using infineon PEB2256 framer for serial ATM or just T1/E1.
- Fast Ethernet connected to PCMCIA port or Port-D using LSI-Logic 80225.
- **ATM Mode operate in Utopia Split or mux and also can work multy phy or singe phy.**
- ATM25 connected to PCMCIA Port & Port-D in split mode, or only Port-D in mux mode using IDT77V106.
- ATM155 connected to PCMCIA Port & Port-D in split mode, or only Port-D in mux mode using NEC uPD98404 device.
- Memory Disable Option for each local memory map slaves.
- Board Control & Status Register - 5 BCSR, Controlling Board's Operation.
- Programmable Hard-Reset Configuration via BCSR.
- 5V only PCMCIA Socket With Full Buffering, Power Control and Port Disable Option. Complies with PCMCIA 2.1+ Standard.
- Module Enable Indications.
- 10-Base-T Port On-Board, with Stand-By Mode.
- IrDA (4MBps) Port with Stand-By Mode.
- Dual RS232 port with Low-Power Option per each port.
- On - Board Debug Port Controller & also ADI I/F.
- MPC86xADS Serving as Debug Station for Target System option.
- Optional Hard-Reset Configuration Burned in Flash[A].
- External Tools' Identification Capability, via BCSR.

A. Available only if supported also on the MPC86x.

- Expansion connectors includes all the CPM ports & bus signals in order to control external peripherals.
- Soft / Hard[A] Reset Push - Button
- ABORT Push - Button
- Single[B] 5V Supply.
- Reverse / Over Voltage Protection for Power Inputs.
- 3.3V VDDL/VDDH for older version then MPC866. 1.8V VDDL & 3.3V HDDH for MPC866, done by jumper.
- Power Indications for Each Power Bus.
- External dip switches for selections the ATM & Fast Ethernet options in the MPC86x.

A. Hard reset is applied by depressing BOTH Soft Reset & ABORT buttons.
B. Unless a 12V supply is required for a PCMCIA card or for a 12V programmable Flash SIMM.

FIGURE 1-1 MPC86xADS Block Diagram



1•7 MPC86xADS Goals

The MPC86xADS is meant to become a general platform for s/w and h/w development around the MPC86x family. Using its on-board resources and its associated debugger, the developer is able to load his code, run it, set breakpoints, display memory and registers and connect his own proprietary h/w via the expansion connectors, to be incorporated to a system with the MPC.

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

2 - Hardware Preparation and Installation

2•1 INTRODUCTION

This chapter provides unpacking instructions, hardware preparation, and installation instructions for the MPC86xADS.

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.

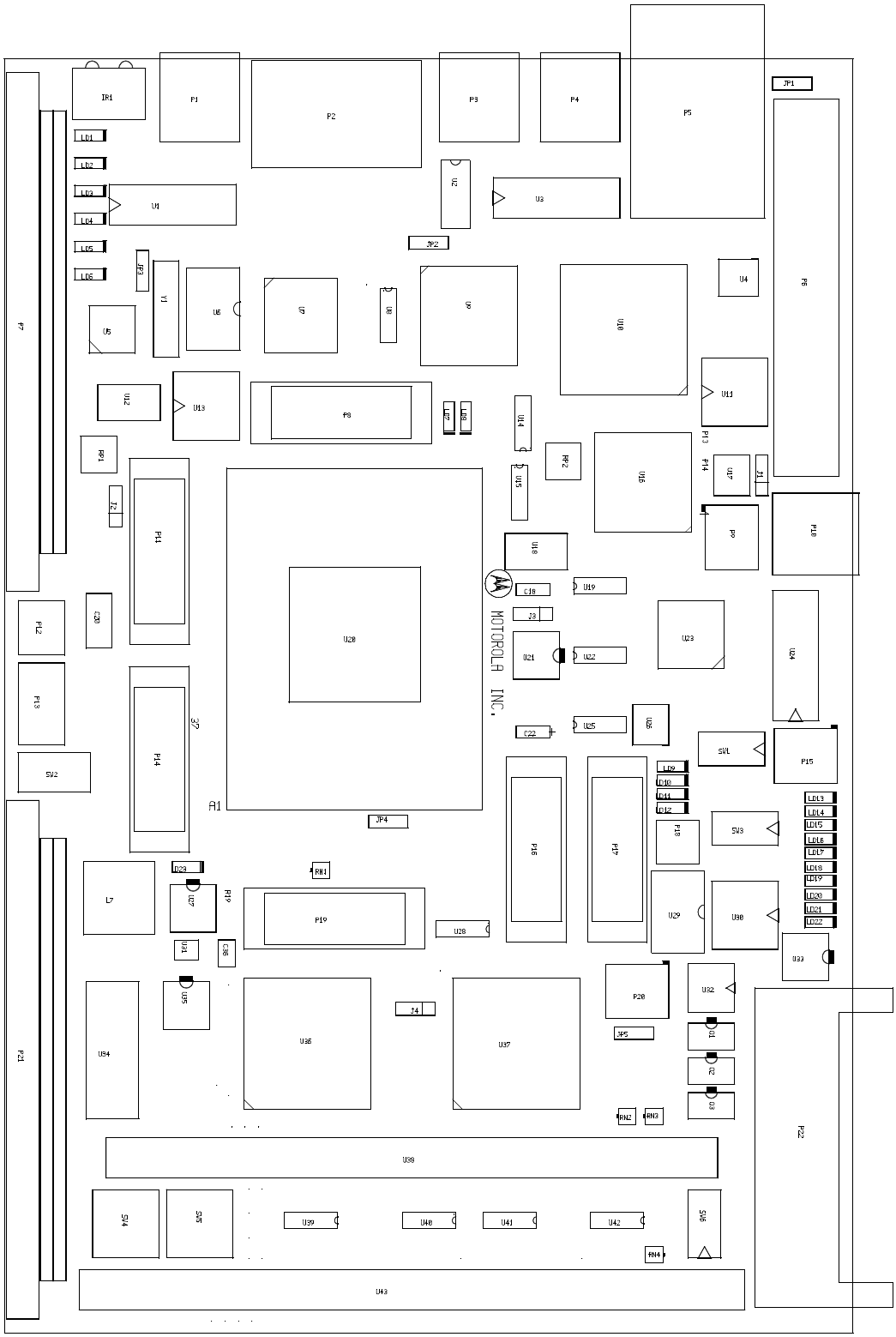
2•3 HARDWARE PREPARATION

To select the desired configuration and ensure proper operation of the MPC86xADS board, changes of the Dip-Switch settings or jumpers may be required before installation. The location of the switches, LEDs, Dip-Switches, jumpers and connectors is illustrated in FIGURE 1-2. 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:

- ADI port address
- MPC Clock Source
- MPC Internal Logic Supply Source
- Debug Mode Indication Source
- ATM Mode - Split, mux, single phy or multy phy.
- Fast Ethenet source Port, PortD/PCMCIA-Port.
- PCMCIA Enable.
- MPC I/O port connected to Expansion Connector.
- RS232 Port-(1) on SMC2 enable with conjunction with ATM single phy.

A. Either on or off-board.

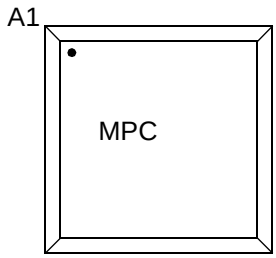
FIGURE 1-2 MPC86xADS Top Side Part Location diagram



2•3•1 MPCs' Replacing - U20

Before replacing the MPC the user should turn off the power. When replacing U20 with another MPC it should be noticed where is the MPCs' A1 pin. Put the new MPC in the same direction as the old one. MPC866 thickness is 2.5mm while the other MPC86x thickness is 1.7mm. The board has a socket with clamshell inorder that the socket will fit to both devices the board has been supplied with a spacer to put between the socket cover (shell) and the device. For MPC86x that is not MPC866 add the spacer between the device and the socket cover.

FIGURE 1-3 MPC TOP VIEW



2•3•2 ADI Port Address Selection

The MPC86xADS can have eight possible slave addresses set for its ADI port, enabling up to eight MPC86xADS boards to be connected to the same ADI board in the host computer. The selection of the slave address is done by setting switches 1, 2 & 3 in the Dip-Switch - SW1. Switch 1 stands for the most-significant bit of the address and switch 3 stands for the least-significant bit. If the switch is in the 'ON' state, it stands for logical'1'. In FIGURE 1-4 DS1 is shown to be configured to address'0'.

FIGURE 1-4 Configuration Dip-Switch - SW1

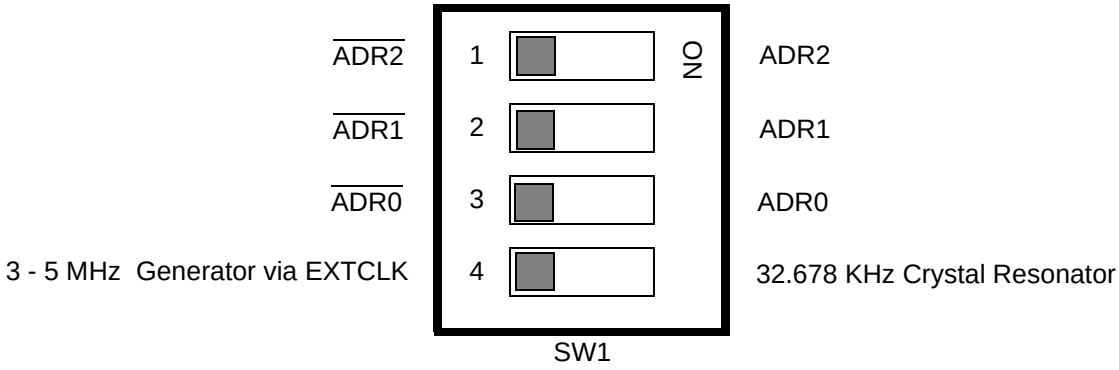


TABLE 2-1. describes the switch settings for each slave address:

TABLE 2-1. ADI Address Selection

ADDRESS	*Switch 1*	*Switch 2*	*Switch 3*
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF

TABLE 2-1. ADI Address Selection

ADDRESS	*Switch 1*	*Switch 2*	*Switch 3*
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

2•3•3 Clock Source Selection

Switch #4 on SW1 selects the clock source for the MPC. When it is in the'ON' position while the ADS is powered-up, the on-board 32.768 KHz crystal resonator becomes the clock source and the PLL multiplication factor becomes 1:513. When switch #4 is in the'OFF' position while the ADS is powered-up, the on-board 4MHz clock generator becomes the clock source while the PLL multiplication factor becomes 1:5. ***Note: For device other then MPC866, SW3 -1 should be 'ON***'

2•3•4 VDDL Source Selection (J3)

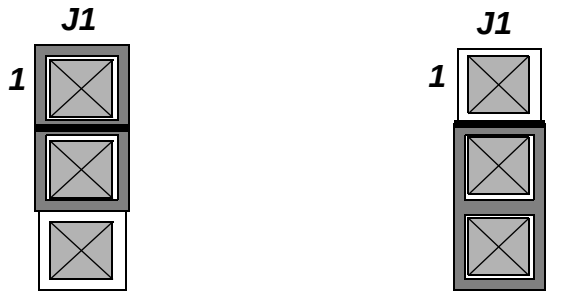
This board can be use for MPC866 and its derivative and MPC862 and its derivative. For MPC866 VDDL should be 1.8V - J3 connect pins 1-2. For MPC862 and its derivative VDDL should be 3.3V. J3 connect pins 2-3.

2•3•5 Debug Mode Indication Source Selection

Jumper J1 selects between VFLS(0:1) signals and FRZ signal of the MPC as an indication for debug mode state. Since with the MPC86xs, each of these signals has alternate function, it may be necessary to switch between the two sources, in favor of alternate function being used.

When a jumper is positioned between pins 1 and 2 of J1 - VFLS(0:1) are selected towards the debug-port controller. When a jumper is placed between positions 2 - 3 of J1(2) - FRZ signal is selected.

FIGURE 2-1 J1 - VFLS / FRZ Selection



VFLS(0:1) Selected ***FRZ Selected***

2•3•6 ATM Mode - Split, mux, single phy or multy phy & Fast-Ethernet source Control, Expansion connector.

According to SW3(2,3,4) the user can select the Fast-Ethernet & ATM source and function. The table

bellow show the configuration options.

TABLE 2-2. ATM & Fast-Ethernet configuration.

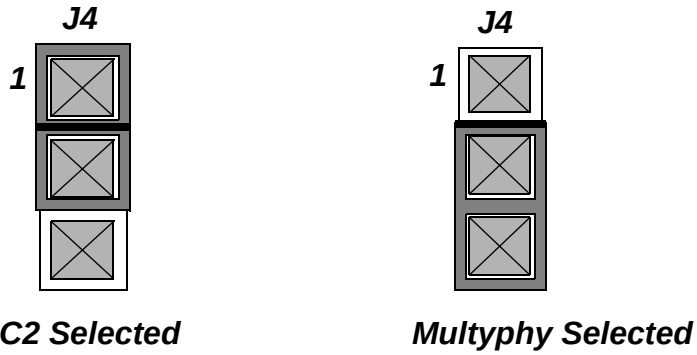
SW3				
Switch 2	***Switch 3***	***Switch 4***	***PORT-D***	***PORT-PCMCIA***
ON	ON	ON	Expansion Connector	Expansion Connector
ON	ON	OFF	Expansion Connector	Not Define
ON	OFF	ON	ATM MUX	PCMCIA
ON	OFF	OFF	ATM MUX	MII
OFF	ON	ON	ATM SPLIT	
OFF	ON	OFF	Not Define	Expansion Connector
OFF	OFF	ON	Fast-Ethernet	PCMCIA
OFF	OFF	OFF	Fast-Ethernet	Expansion Connector

Note: Inorder to work in single phy the user should config the unused phy to any address and the wanted phy to different address, The ATM25 config phy address is in address 0x2000008, the ATM155 phy address register is in 0x2000129. Then drive the RxAdd(1)-PB17 & RxAdd(0)-PB16 and TxAdd(1)-PB21 & TxAdd(0)-PB20 to the wanted phy address, by driving them constantly.

2•3•7 RS232 - 1 on SMC2 enable by operating the ATM on single phy. (J4)

Jumper J4 selects between ATM multy phy and enable RS232 port 1. Pins PB21 and PB20 have a multiple functions TXD and RXD for RS232-1 and ATM TXADD1 and TXADD0 on the board. the MPC allow to work with ATM single phy and SMC2 so the board has this ability too. By J4, if J4 connect between 1,2 it drive the address pins to the ATM phy's to be 0b00011 so the user can work with single phy, he can select which phy by writing to the address phy register this address. If J4 connect 2,3 it allow the operation with multy phy, and the RS232 on SMC2 should not be enable by the appropriate BCSR register.
Note: The ATM ADD that are used on the board are only RXADD0, RXADD1, TXADD0 and TXADD1.

FIGURE 2-2 J4 - SMC2 or Multyphy.



2•4 INSTALLATION INSTRUCTIONS

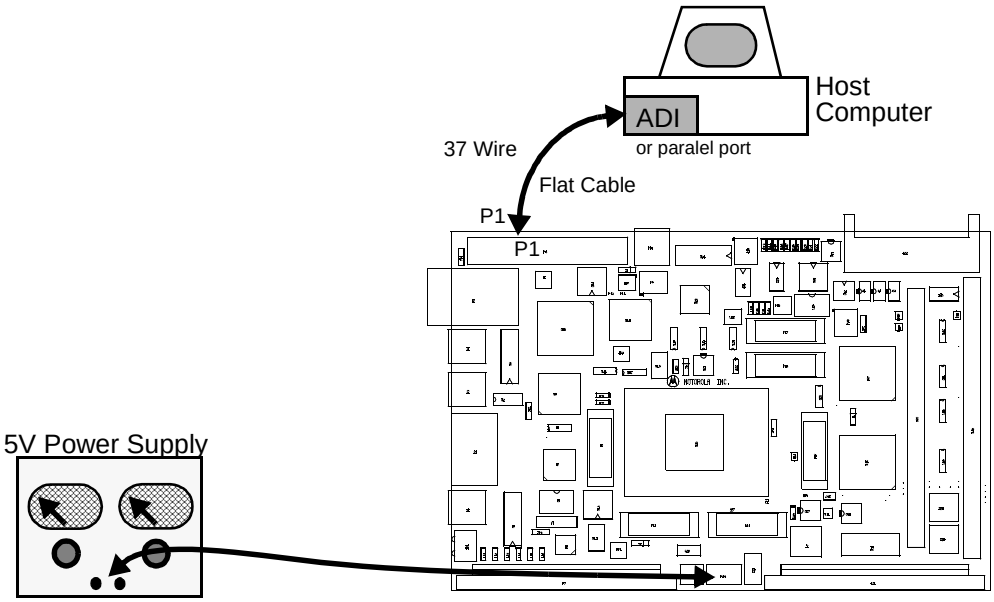
Because the board shipped with no DRAM EDO and all the SW around are base on that there is an DRAM the user should change BR2, BR3, BR4 and OR4 to: BR2 & BR3 valid bit should be 0 (bit 31) BR4 = 0x000000C1 and OR4 should be 0xFC800a00, this configuration will map the SDRAM to ADD 0 & 0x3000000 for 8MByte. When the MPC86xADS has been configured as desired by the user, it can be installed according to the required working environment as follows:

- Host Controlled Operation
- Debug Port.
- Stand-Alone

2•4•1 Host Controlled Operation

In this configuration the MPC86xADS is controlled by a host computer via the ADI through the debug port. This configuration allows for extensive debugging using on-host debugger.

FIGURE 2-3 Host Controlled Operation Scheme



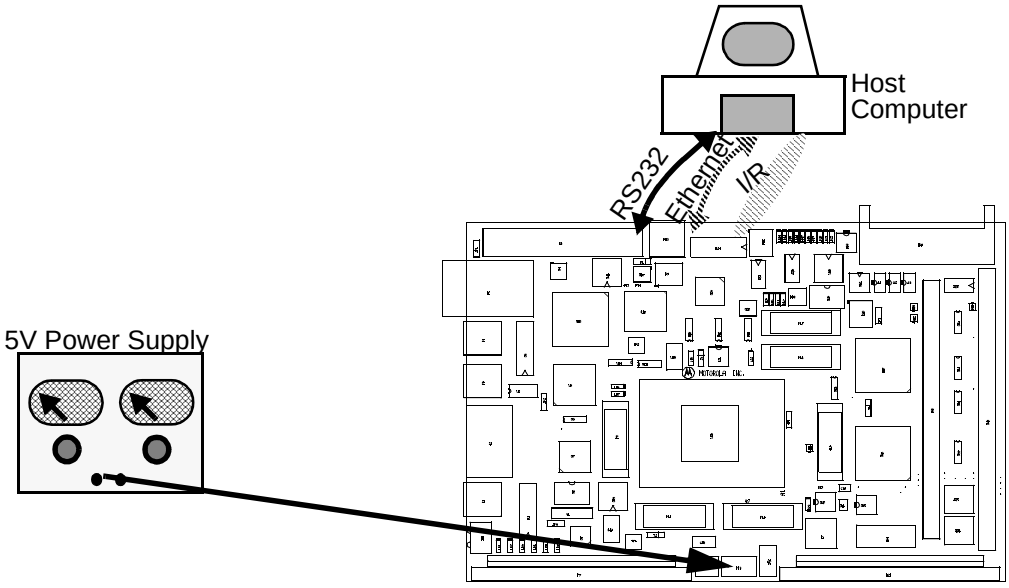
2•4•2 Stand Alone Operation

In this mode, the ADS is not controlled by the host via the ADI/Debug port. It may connect to host via one of its other ports, e.g., RS232 port, I/R port, Ethernet port, etc.'. Operating in this mode requires an application program to be programmed into the board's Flash memory (while with the host controlled operation, no memory is required at all).

2•4•3 Debug Port.

In this mode of operation the user control the board via P9 through an external tool connected to its computer to the parallel port. The board can work as it work in a host mode but it controlled via the MPC debug pins and not through the ADI connector.

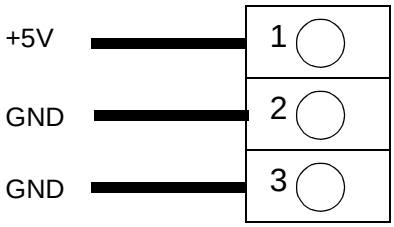
FIGURE 2-4 Stand Alone Configuration



2•4•4 +5V Power Supply Connection

The MPC86xADS requires +5 Vdc @ 3A max, power supply for operation. Connect the +5V power supply to connector P13 or P13A as shown below:

FIGURE 2-5 P13: +5V Power Connector



P13 is a 3 terminal block power connector with power plug. The plug is designed to accept 14 to 22 AWG wires. It is recommended to use 14 to 18 AWG wires. To provide solid ground, two Gnd terminals are supplied. It is recommended to connect both Gnd wires to the common of the power supply, while VCC is connected with a single wire.

P13A is a Connector that should be use with the power supply which supplied with the board box. Plug the power supply to P13A.

NOTE

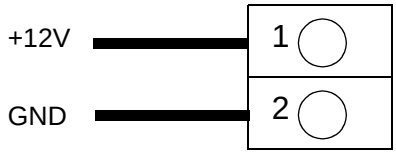
Since hardware applications may be connected to the MPC86xADS via the Expansion connectors' P7 & P21, power consumption should be taken into consideration when a power supply is connected to the MPC86xADS. In other words when adding a HW into the expansion connectors remember that the additional will not have more power consumption then 1A

2•4•5 P21: +12V Power Supply Connection

The MPC86xADS requires +12 Vdc @ 1 A max, power supply for the PCMCIA channel Flash programming capability or for 12V programmable Flash SIMM. The MPC86xADS can work properly without the +12V power supply, if there is no need to program either a 12V programmable PCMCIA flash card or a 12V programmable Flash SIMM.

Connect the +12V power supply to connector P12 as shown below:

FIGURE 2-6 P12: +12V Power Connector



P12 is a 2 terminal block power connector with power plug. The plug is designed to accept 14 to 22 AWG wires. It is recommended to use 14 to 18 AWG wires.

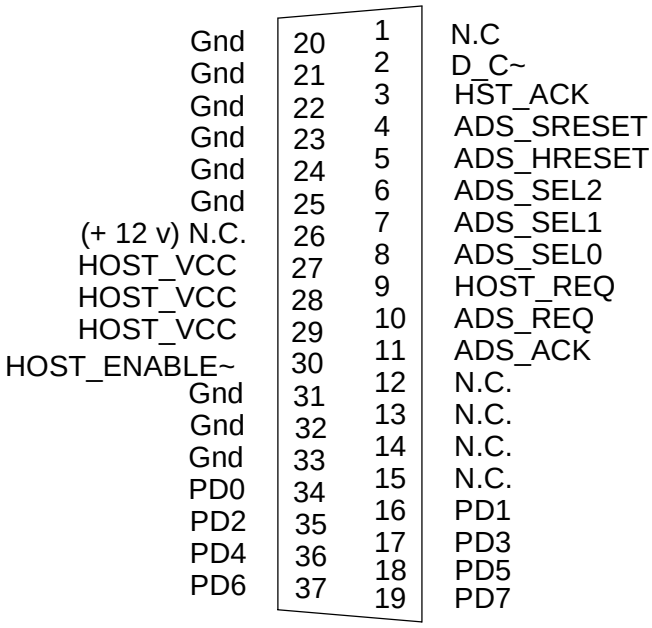
2•4•6 ADI Installation

For ADI installation on various host computers, refer to APPENDIX D -.

2•4•7 Host computer to MPC86xADS Connection

The MPC86xADS ADI interface connector, P6, is a 37 pin, male, D type connector. The connection between the MPC86xADS and the host computer is by a 37 line flat cable, supplied with the ADI board. Or by BDM Connector. FIGURE 2-7 below shows the pin configuration of the ADI connector.

FIGURE 2-7 P6 - ADI Port Connector



NOTE: Pin 26 on the ADI is connected to +12 v power supply, but it is not used in the MPC86xADS.

2•4•8 Debug Port Connector.

Through this connector the user can control the board like it done from the ADI connector. Today most of the control SW use this connector through a command converter box that connected from the other side to the PC parallel port.

FIGURE 2-8 BDM Connectoe.

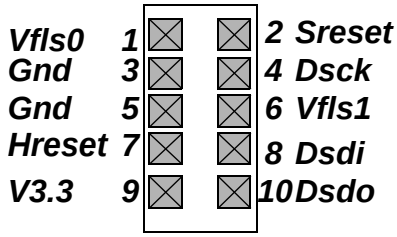
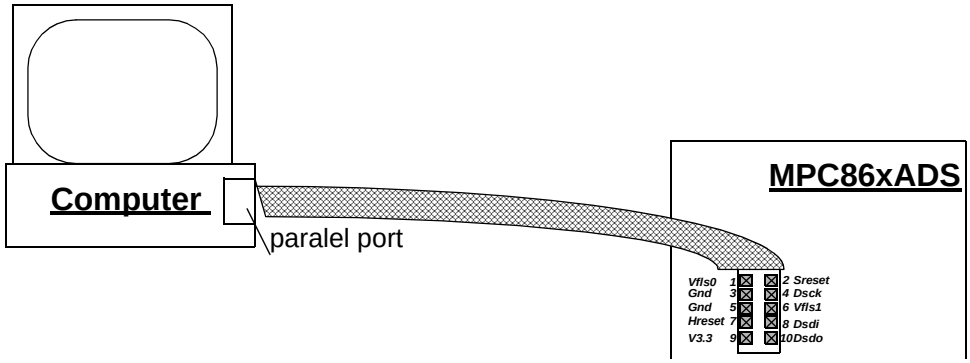


FIGURE 2-9 BDM connector connected to the board

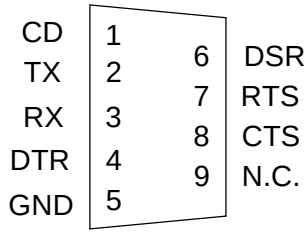


2•4•9 Terminal to MPC86xADS RS-232 Connection

A serial (RS232) terminal or any other RS232 equipment, may be connected to the RS-232 connectors P2A and P2B. The RS-232 connectors is a 9 pin, female, Stacked D-type connector as shown in FIGURE 2-10.

The connectors are arranged in a manner that allows for 1:1 connection with the serial port of an Personal Computer. via a flat cable.

FIGURE 2-10 PA7, PB7 - RS-232 Serial Port Connectors



NOTE: The RTS line (pin 7) is not connected on the MPC86xADS.

2•4•10 Memory Installation

The MPC86xADS has two types of memory SIMM:

- Dynamic Memory SIMM, will not populated only the socket will be soldered.
- Flash Memory SIMM.

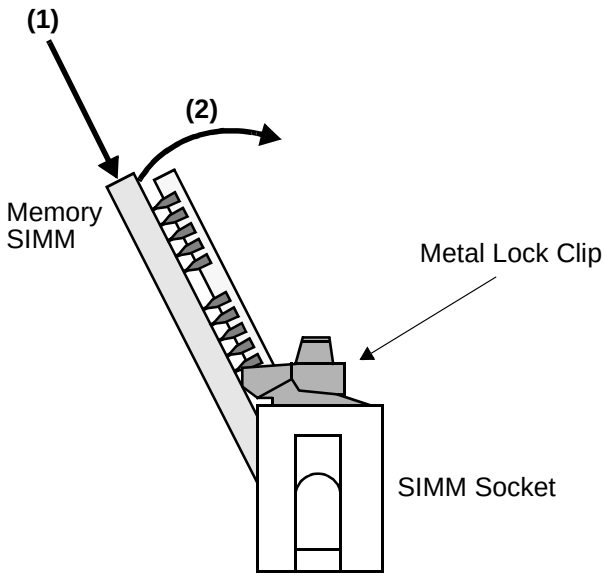
To install a memory SIMM, it should be taken out of its package, put diagonally in its socket (no error can be made here, since the Flash socket has 80 contacts, while the DRAM socket has 72) and then twisted to a vertical position until the metal lock clips are locked. See FIGURE 2-11 "Memory SIMM Installation" below.

Hardware Preparation and Installation

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 2-11 Memory SIMM Installation



3 - OPERATING INSTRUCTIONS

3•1 INTRODUCTION

This chapter provides necessary information to use the MPC86xADS 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 MPC86xADS has the following switches and indicators.

3•2•1 ABORT Switch SW5

The ABORT switch is normally used to abort program execution, this by issuing a level 0 interrupt to the MPC. 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 MPC86xADS. The ABORT switch signal is debouncing, and can not be disabled by software.

3•2•2 SOFT RESET Switch SW6

The SOFT RESET switch SW5 performs Soft reset to the MPC internal modules, maintaining MPC's configuration (clocks & chip-selects) Dram and SDram contents. The switch signal is debouncing, and it is not possible to disable it by software. At the end of the Soft Reset Sequence, the Soft Reset Configuration is sampled and becomes valid.

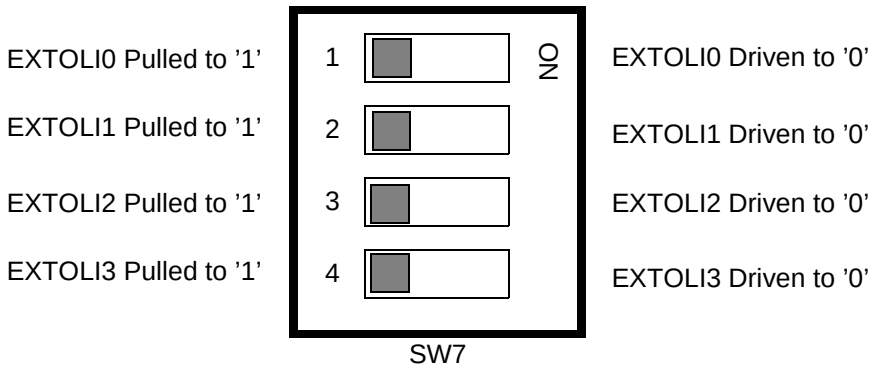
3•2•3 HARD RESET - Switches SW5 & SW6

When BOTH switches - SW5 and SW6 are depressed simultaneously, HARD reset is generated to the MPC. When the MPC is HARD reset, all its configuration is lost, including data stored in the DRAM or SDRAM and the MPC has to be re-initialized. At the end of the Hard Reset sequence, the Hard Reset Configuration stored in BCSR0 becomes valid.

3•2•4 SW7 - Software Options Switch

SW7is a 4-switches Dip-Switch. This switch is connected over EXTOLI(0:3) lines which are available at BCSR, S/W options may be manually selected, according to SW7 state.

FIGURE 3-1 SW7 - Description



3•2•5 SW3 - ATM Fast-Ethernet configuration.

Table 3-1 desires the way of configuring the board in order to be fit to the MPC866/862 way of operation. As the user know the MPC86x can be configure in ATM or/and Fast ethernet (MPC866 & MPC862 has both ATM and Fast Ethernet) the board can be configure for many options over portD and PCMCIA Port like ATM Mux, ATM Split, Single phy or multy phy & Fast ethernet or connect an external HW through the

Expansion Connectors. SW3 (2,3,4) select all these options.
Note: Before changing the position of the SW3(2,3,4) the user should turn off the power.
When the switches are in option(0) position the user can use PD & PCMCIA Ports through the Expansion Connectors. Option(2,3) select ATM-Mux through Port-D and PCMCIA or MII through PCMCIA Port. Option(4), select ATM split. options(6,7) select PortD for MII and PCMCIA for PCMCIA or Expansion Connector.
NOTE: In case of board with PHYs ATM25 add ATM 155, for selection ATM in Single Phy or Multy Phy the user should configure the following Pins to outputs, drive Port-B(16,17 ATM Rx-Add) & Port-B(19,20, ATM TX-ADD) to the wanted address like the internal phy default address for one of the wanted phys. The other phy can be left for its default address.

TABLE 3-1. ATM & Fast-Ethernet configuration.

	SW3				
Option#	***Switch 2***	***Switch 3***	***Switch 4***	***PORT-D***	***PORT-PCMCIA***
0	ON	ON	ON	Expansion Connector	Expansion Connector
1	ON	ON	OFF	Expansion Connector	Not Define
2	ON	OFF	ON	ATM MUX	PCMCIA
3	ON	OFF	OFF	ATM MUX	Fast-Ethernet
4	OFF	ON	ON	ATM SPLIT	
5	OFF	ON	OFF	Not Define	Expansion Connector
6	OFF	OFF	ON	Fast-Ethernet	PCMCIA
7	OFF	OFF	OFF	Fast-Ethernet	Expansion Connector

3•2•6 GND Bridges

There are 3 GND bridges on the MPC86xADS. They are meant to assist general measurements and logic-analyzer connection.

Warning

When connecting to a GND bridge, use only INSULATED GND clips. Failure in doing so, might result in permanent damage to the MPC86xADS.

3•2•7 Ethernet 10Base-T. ETH ON - LD21

When the yellow ETH ON led is lit, it indicates that the ethernet port transceiver - the MC68160 EEST, is active. When it is dark, it indicates that the EEST is in power down mode, enabling the use of its associated SCC pins off-board via the expansion connectors.

3•2•8 IRD ON - LD20

When the yellow IRD ON led is lit, it indicates that the Infra-Red transceiver - the TFDS6000, is active and enables communication via that medium. When it is dark, the I/R transceiver is in shutdown mode, enabling the use of its associated SCC pins off-board via the expansion connectors.

3•2•9 RS232 Port 1 ON - LD19

When the yellow RS232 Port 1 ON led is lit, it designates, that the RS232 transceiver connected to PA2, is active and communication via that medium is allowed. When darkened, it designates that the transceiver

is in shutdown mode, so its associated MPC pins may be used off-board via the expansion connectors.

3•2•10 RS232 Port 2 ON - LD22

When the yellow RS232 Port 2 ON led is lit, it designates that the RS232 transceiver connected to PB2, is active and communication via that medium is allowed. When darkened, it designates, that the transceiver is in shutdown mode, so its associated MPC pins may be used off-board via the expansion connectors.

3•2•11 Ethernet RX Indicator - LD4

The green Ethernet Receive LED indicator blinks whenever the EEST is receiving data from one of the Ethernet port.

3•2•12 Ethernet TX Indicator - LD5

The green Ethernet Receive LED indicator blinks whenever the EEST is transmitting data via the Ethernet port.

3•2•13 Ethernet JABB Indicator - LD6

The red Ethernet TP Jabber LED indicator - JABB, lights whenever a jabber condition is detected on the TP ethernet port.

3•2•14 Ethernet CLSN Indicator LD3

The red Ethernet Collision LED indicator CLSN, blinks whenever a collision condition is detected on the ethernet port, i.e., simultaneous receive and transmit.

3•2•15 Ethernet PLR Indicator - LD1

The red Ethernet TP Polarity LED indicator - PLR, lights whenever the wires connected to the receiver input of the ethernet port are reversed. The LED is lit by the EEST, and remains on while the EEST has automatically corrected for the reversed wires.

3•2•16 Ethernet LIL Indicator - LD2

The yellow Ethernet Twisted Pair Link Integrity LED indicator - LIL, lights to indicate good link integrity on the TP port. The LED is off when the link integrity fails.

3•2•17 5V Indicator - LD13

The yellow 5V led, indicates the presence of the +5V supply at P13.

3•2•18 RUN Indicator - LD23

When the green RUN led - LD23 is lit, it indicates that the MPC is not in debug mode, i.e., VFLS0 & VFLS1 == 0 (or FRZ == 0, which ever selected by J1).

3•2•19 FLASH ON - LD17

When the yellow FLASH ON led is lit, it indicates that the FLASH SIMM is enabled in the BCSR1 register. I.e., any access done to the CS0~ address space will hit the flash memory. When it is dark, the flash is disabled and CS0~ may be used off-board via the expansion connectors.

3•2•20 DRAM ON - LD15

When the yellow DRAM ON led is lit, it indicates the DRAM SIMM is enabled in BCSR1. Therefore, any access made to CS2~ (or CS3~) will hit on the DRAM. When it is dark, it indicates that either the DRAM is disabled in BCSR1, enabling the use of CS2~ and CS3~ off-board via the expansion connectors.

3•2•21 SDRAM ON - LD14

When the yellow SDRAM ON led is lit, it indicates the SDRAM is enabled in BCSR1. Therefore, any access made to CS4~ (will hit on the SDRAM. When it is dark, it indicates that either the SDRAM is disabled in BCSR1, enabling the use of CS4~ off-board via the expansion connectors.

3•2•22 PCMCIA ON - LD17

When the yellow PCMCIA ON led is lit, it indicates the following:

1) Address & strobe buffers are driven towards the PCMCIA card
2) Data buffers are driven to / from the PCMCIA card whenever CE1A~ or CE2A~ signals are asserted.
3) Card status lines are driven towards the MPC from the PCMCIA card.

When it is dark, it indicates that all the above buffers are tri-stated and the pins associated with PCMCIA channel A, may be used off-board via the expansion connectors.

3•2•23 Fast-Ethernet Full Duplex LD9

The function of this led is to indicate Full Duplex Detect.

3•2•24 Fast-Ethernet Link + Activity LD10

The function of this led is to indicate the occurrence of Link or Activity.

3•2•25 Fast-Ethernet Collision LED LD11

The function of this led is to indicate Full Duplex Detect.

3•2•26 Fast-Ethernet Link LED - speed LD12

The function of this led is to indicate Full Duplex Detect.

3•2•27 ATM25Mhz RX-LED LD8

This led indicates when it is light that there is a receive on the ATM25 twisted pair lines.

3•2•28 ATM25Mhz TX-LED LD7

This led indicates when it is light that there is a transmit on the ATM25 twisted pair lines.

3•3 MEMORY MAP

All accesses to MPC86xADS's memory slaves are controlled by the MPC's memory controller. Therefore, the memory map is re programmable 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 EDO DRAM and FLASH memory SIMMs mounted on board and initializes the chip-selects accordingly. The SDRAM, DRAM and the FLASH memory respond to all types of memory access i.e., user / supervisory, program / data and DMA.

In the following paragraph there is a description of memory map for 2 options: Compatible Mode and MPC86xADS New Mode.

Compatible Mode is using an EDO DRAM and 8MByte SDRAM. In this Mode all the programmable registers are remain the same, (all the memory map is the same as the MPC8xxFADS board) ***except OR4 Mask register bits, that will be changed according to SDRAM size to 0xFF80.***

MPC86xADS New Mode is where the EDO DRAM is not used. In this case the SDRAM will be mapped differently, see the TABLE 3-2. "Memory Map in MP86xADS New Mode," on page 22, TABLE 3-3. "Memory Map in Compatible Mode" on page 23. The following programmable changes should be made in order to work in the board in the MPC86xADS New Mode:

- Programming BR2 Base Address bits for EDO DRAM should be not valid (L-bit should be cleared).
- Programming OR4 Mask Register bits for SDRAM should be changed according to SDRAM size, where the 2 MS bits are not masked. For 8MByte SDRAM, OR4 Mask bits = 0xFC80 and BIH should be 0

.

TABLE 3-2. Memory Map in MP86xADS New Mode,

ADDESS RANGE	*Memory Type*	*Device Type*				*Port Size*
00000000 - 007FFFFF[a]	SDRAM	8MByte				32
						32
						32
						32
02000000 - 020FFFFF	Communication ports: CS5, ATM25, ATM155, T1/E1 Framer, BCSR5,6					
02000000 - 020000FF	ATM25					
02000100 - 020001FF	ATM155					
02000200 - 020002FF	T1/E1 Framer,					
02000300 - 020003FF	Control register					
02100000 - 02107FFF	BCSR(0:4)[b]					32[c]
02100000 - 02107FE3	BCSR0					
2100004 - 02107FE7	BCSR1					
2100008 - 02107FEB	BCSR2					
210000C - 02107FEF	BCSR3					
2100010 - 02107FF3	BCSR4					
02108000 - 021FFFFF	Empty Space					
02200000 - 02207FFF	MPC Internal MAP[d]					32
02208000 - 027FFFFF	Empty Space					
02800000 - 029FFFFF	Flash SIMM	MCM29F020	MCM29F040 SM732A1000A	MCM29F080 SM732A2000		32
02A00000 - 02BFFFFF						32
02C00000 - 02FFFFFF						32
03000000 - 037FFFFF	SDRAM[a]	(for 8MByte)				32
03400000 - FFFFFFFF	Empty Space					

a. 0 - 0x007F_FFFF, 0x0300_0000 - 0x037F_FFFF are both mapped to SDRAM (8MByte).

b. The device appears repeatedly in multiples of its size. E.g., BCSR0 appears at memory locations 2100000, 2100020, 2100040..., while BCSR1 appears at 2100004, 2100024, 2100044... and so on.

c. Only upper 16 bit (D0-D15) are in fact used.
d. Refer to the relevant MPC User's Manual for complete description of the MPC internal memory map.

.

TABLE 3-3. Memory Map in Compatible Mode

ADDESS RANGE	*Memory Type*	*Device Type*				*Port Size*
00000000 - 003FFFFF	DRAM SIMM	MB321Bx[a]08	MB322Bx[a]08	MC324Cx[a]00	MB328Cx[a]00	32
00400000 - 007FFFFF						32
00800000 - 00FFFFFF						32
01000000 - 01FFFFFF						32
02000000 - 020FFFFF	Comunication ports: CS5, ATM25, ATM155, T1/ E1 Framer, BCSR 5					
02000000 - 020000FF	ATM25					
02000100 - 020001FF	ATM155					
02000200 - 020002FF	T1/E1 Framer,					
02000300 - 020003FF	Control register					
02100000 - 02107FFF	BCSR(0:4)[b]					32[c]
02100000 - 02107FE3	BCSR0					
2100004 - 02107FE7	BCSR1					
2100008 - 02107FEB	BCSR2					
210000C - 02107FEF	BCSR3					
2100010 - 02107FF3	BCSR4					
02108000 - 021FFFFF	Empty Space					
02200000 - 02207FFF	MPC Internal MAP[d]					32
02208000 - 027FFFFF	Empty Space					
02800000 - 029FFFFF	Flash SIMM	MCM29F020	MCM29F040 SM732A1000A	MCM29F080 SM732A2000		32
02A00000 - 02BFFFFF						32
02C00000 - 02FFFFFF						32
03000000 - 037FFFFF	SDRAM (for 8MByte)					32
03400000 - FFFFFFFF	Empty Space					

a. x ∈ [B,T]

b. The device appears repeatedly in multiples of its size. E.g., BCSR0 appears at memory locations 2100000, 2100020, 2100040..., while BCSR1 appears at 2100004, 2100024, 2100044... and so on.
c. Only upper 16 bit (D0-D15) are in fact used.
d. Refer to the relevant MPC User's Manual for complete description of the MPC internal memory map.

3•4 MPC Registers' Programming

The MPC provides the following functions on the MPC86xADS:

1) DRAM Controller
2) SDRAM Controller
3) Chip Select generator.
4) UART for terminal or host computer connection.
5) Ethernet controller.
6) Infra-Red Port Controller
7) General Purpose I/O signals.
8) ATM controller.
9) T1/E1 (TDM) controller.
10) Fast Ethernet Controller.

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

OPERATING INSTRUCTIONS

For better understanding the of the following initialization refer to the MPC86x User's Manual for more information.

TABLE 3-4. SIU REGISTERS' PROGRAMMING

Register	*Init Value[hex]*	*Description*
SIUMCR	01012440	Internal arbitration, External master arbitration priority - 0, External arbitration priority - 0, PCMCIA channel II pins - PCMCIA, Debug Port on JTAG port pins, FRZ/IRQ6~ - FRZ, debug register - locked, No parity for non-CS regions, DP(0:3)/ IRQ(3:6)~ pins - DP(0:3), reservation disabled, SPKROUT - Tri-stated, BS_A(0:3)~ and WE(0:3)~ are driven just on their dedicated pins, GPL_B5~ enabled, GPL_A/ B(2:3)~ function as GPLs.
SYPCR	FFFFFF88	Software watchdog timer count - FFFF, Bus-monitor timing FF, Bus-monitor - Enabled, S/W watch-dog - Freeze, S/W watch-dog - disabled, S/W watch-dog (if enabled) causes NMI, S/W (if enabled) not prescaler.
TBSCR	00C2	No interrupt level, reference match indications cleared, interrupts disabled, no freeze, time-base disabled.
RTCSC	00C2	Interrupt request level - 0, 32768 Hz source, second interrupt disabled, Alarm interrupt disabled, Real-time clock - FREEZE, Real-time clock enabled.
PISCR	0082	No level for interrupt request, Periodic interrupt disabled, clear status, interrupt disabled, FREEZE, periodic timer disabled.

3•4•1 Memory Controller Registers Programming

The memory controller on the MPC86xADS is initialized to 50 MHz operation. I.e., registers' programming is based on 50 MHZ timing calculation except for refresh timer which is initialized to 16.67Mhz, the lowest frequency at which the ADS may wake up. Since the ADS may be made to wake-up at 25MHz[A] as well, the initialization are not efficient, since there are too many wait-states inserted. Therefore, additional set of initialization is provided to support efficient 25MHz operation.

The reason for initializing the ADS for 50Mhz is to allow proper (although not efficient) ADS operation through all available ADS clock frequencies.

A. The only parameter which is initialized to the start-up frequency, is the refresh rate, which would have been inadequate if initialized to 50Mhz while board is running at a lower frequency. Therefore, for best bus bandwidth availability, refresh rate should be adapted to the current system clock frequency.

OPERATING INSTRUCTIONS

Warning

Due to availability problems with few of the supported memory components, the below initialization were not tested with all parts. Therefore, the below initialization are liable to CHANGE, throughout the testing period.

TABLE 3-5. Memory Controller Initialization For 50Mhz with DRAM-EDO

Register	*Device Type*	*Init Value [hex]*	*Description*
BR0	All Flash SIMMs supported.	02800001	Base at 2800000, 32 bit port size, no parity, GPCM
OR0	MCM29F020-90	FFE00D34	2MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F040-90 SM732A1000A-9	FFC00D34	4MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F080-90 SM732A2000-9	FF800D34	8MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F020-12	FFE00D44	2MByte block size, all types access, CS early negate, 8 w.s., Timing relax
	MCM29F040-12 SM732A1000A-12	FFC00D44	4MByte block size, all types access, CS early negate, 8 w.s., Timing relax
	MCM29F080-12 SM732A2000-12	FF800D44	8MByte block size, all types access, CS early negate, 8 w.s., Timing relax
BR1	BCSR	02100001	Base at 2100000, 32 bit port size, no parity, GPCM
OR1		FFFF8110	32 KByte block size, all types access, CS early negate, 1 w.s.
BR2	All Dram SIMMs Supported	00000081	Base at 0, 32 bit port size, no parity, UPMA
OR2	MCM36100/200-60/70	FFC00800	4MByte block size, all types access, initial address multiplexing according to AMA.
	MCM36400/800-60/70 MT8/16D432/832X-6/7	FF000800	16MByte block size, all types access, initial address multiplexing according to AMA.
BR3	MCM36200-60/70	00400081	Base at 400000, 32 bit port size, no parity, UPMA
	MCM36800-60/70 MT16D832X-6/7	01000081	Base at 1000000, 32 bit port size, no parity, UPMA
OR3	MCM36200-60/70	FFC00800	4MByte block size, all types access, initial address multiplexing according to AMA
	MCM36800-60/70 MT16D832X-6/7	FF000800	16MByte block size, all types access, initial address multiplexing according to AMA.

OPERATING INSTRUCTIONS

TABLE 3-5. Memory Controller Initialization For 50Mhz with DRAM-EDO

Register	*Device Type*	*Init Value [hex]*	*Description*
BR4 Compatible Mode	K4S643232-TC60	030000C1	Base at 3000000, on UPM B
OR4 Compatible Mode		FFC00A00	4 MByte block size, all types access, initial address multiplexing according to AMB.
BR4 MPC86x New Mode	K4S643232-TC60	0x000000C1	Base at 0x0, on UPM B
OR4 MPC86x New Mode		0xFC800800	4 MByte block size, all types access, initial address multiplexing according to AMB.
BR5	Comm peripheral	0x02000401	
OR5	Comm peripheral	0xFFF009A6	
MPTPR	All Dram SIMMs Supported	0400	Divide by 16 (decimal)
MAMR	MB321BT08TASN60	40A21114[a] 60A21114[b] C0A21114[c]	refresh clock divided by 40[a] or 60[b] or C0[c], periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB322BT08TASN60	20A21114[a] 30A21114[b] 60A21114[c]	refresh clock divided by 20[a] or 30[b] or 60[c], periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB324CT00TBSN60	40B21114[a] 60B21114[b] C0B21114[c]	refresh clock divided by 40[a] or 60[b] or C0[c], periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB328CT00TBSN60	20B21114[a] 30B21114[b] 60B21114[c]	refresh clock divided by 20[a] or 30[b] or 60[c], periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
MBMR	KS643232C-TC60	D0802114[c] 80802114[d]	refresh clock divided by D0 or 80, periodic timer enabled, type 0 address multiplexing scheme, 1 cycle disable timer, GPL4enabled, 1 loop read, 1 loop write, 4 beats refresh burst.

a. Assuming 16.67 MHz BRGCLK.
b. Assuming 25MHz BRGCLK

c. For 50MHz BRGCLK
d. Assuming 32MHz BRGCLK.

TABLE 3-6. Memory Controller Initialization For 50Mhz with No DRAM-EDO

Register	*Device Type*	*Init Value [hex]*	*Description*
BR0	All Flash SIMMs supported.	02800001	Base at 2800000, 32 bit port size, no parity, GPCM
OR0	MCM29F020-90	FFE00D34	2MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F040-90 SM732A1000A-9	FFC00D34	4MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F080-90 SM732A2000-9	FF800D34	8MByte block size, all types access, CS early negate, 6 w.s., Timing relax
	MCM29F020-12	FFE00D44	2MByte block size, all types access, CS early negate, 8 w.s., Timing relax
	MCM29F040-12 SM732A1000A-12	FFC00D44	4MByte block size, all types access, CS early negate, 8 w.s., Timing relax
	MCM29F080-12 SM732A2000-12	FF800D44	8MByte block size, all types access, CS early negate, 8 w.s., Timing relax
BR1	BCSR	02100001	Base at 2100000, 32 bit port size, no parity, GPCM
OR1		FFFF8110	32 KByte block size, all types access, CS early negate, 1 w.s.
BR2	All Dram SIMMs Supported	00000089	Invalid bank
OR2	MCM36100/200-60/70	FFC00800	Invalid bank
	MCM36400/800-60/70 MT8/16D432/832X-6/7	FF000800	
BR3	MCM36200-60/70	00400089	Invalid bank
	MCM36800-60/70 MT16D832X-6/7	01000089	Invalid bank
OR3	MCM36200-60/70	FFC00800	4MByte block size, all types access, initial address multiplexing according to AMA
	MCM36800-60/70 MT16D832X-6/7	FF000800	16MByte block size, all types access, initial address multiplexing according to AMA.
BR4 Compatible Mode	K4S643232-TC60	030000C1	Base at 3000000, on UPM B
OR4 Compatible Mode		FFC00A00	4 MByte block size, all types access, initial address multiplexing according to AMB.

OPERATING INSTRUCTIONS

TABLE 3-6. Memory Controller Initialization For 50Mhz with No DRAM-EDO

Register	*Device Type*	*Init Value [hex]*	*Description*
BR4 MPC86x New Mode	K4S643232-TC60	0x000000C1	Base at 0x0, on UPM B
OR4 MPC86x New Mode		0xFC800800	4 MByte block size, all types access, initial address multiplexing according to AMB.
BR5	Comm peripheral	0x02000401	
OR5	Comm peripheral	0xFFF009A6	
MPTPR	All Dram SIMMs Supported	0400	Divide by 16 (decimal)
MAMR	MB321BT08TASN60	40A21114[a] 60A21114[b] C0A21114[c]	refresh clock divided by 40[a] or 60[b] or C0[c], periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB322BT08TASN60	20A21114[a] 30A21114[b] 60A21114[c]	refresh clock divided by 20[a] or 30[b] or 60[c], periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB324CT00TBSN60	40B21114[a] 60B21114[b] C0B21114[c]	refresh clock divided by 40[a] or 60[b] or C0[c], periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB328CT00TBSN60	20B21114[a] 30B21114[b] 60B21114[c]	refresh clock divided by 20[a] or 30[b] or 60[c], periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
MBMR	KS643232C-TC60	D0802114[c] 80802114[d]	refresh clock divided by D0 or 80, periodic timer enabled, type 0 address multiplexing scheme, 1 cycle disable timer, GPL4enabled, 1 loop read, 1 loop write, 4 beats refresh burst.

a. Assuming 16.67 MHz BRGCLK.
b. Assuming 25MHz BRGCLK
c. For 50MHz BRGCLK
d. Assuming 32MHz BRGCLK.

TABLE 3-7. UPMA Initializations for 60nsec DRAMs @ 50MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset in UPM		0	8	18	20	30	3C
Contents @ Offset +	0	8FFFEC24	8FFFEC24	8FAFCC24	8FAFCC24	C0FFCC84	33FFCC07
	1	0FFFEC04	0FFFEC04	0FAFCC04	0FAFCC04	00FFCC04	X
	2	0CFFEC04	08FFEC04	0CAFCC00	0CAFCC00	07FFCC04	X
	3	00FFEC04	00FFEC0C	11BFCC47	03AFCC4C	3FFFCC06	X
	4	00FFEC00	03FFEC00	X	0CAFCC00	FFFFCC85	
	5	37FFEC47	00FFEC44	X	03AFCC4C	FFFFCC05	
	6	X	00FFCC08	X	0CAFCC00	X	
	7	X	0CFFCC44	X	03AFCC4C	X	
	8		00FFEC0C		0CAFCC00	X	
	9		03FFEC00		33BFCC4F	X	
	A		00FFEC44		X	X	
	B		00FFCC00		X	X	
	C		3FFFC847		X		
	D		X		X		
	E		X		X		
	F		X		X		

TABLE 3-8. UPMA Initializations for 60nsec EDO DRAMs @ 50MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset in UPM		0	8	18	20	30	3C
Contents @ Offset +	0	8FFBEC24	8FFFEC24	8FFFCC24	8FFFCC24	C0FFCC84	33FFCC07
	1	0FF3EC04	0FFBEC04	0FEFCC04	0FEFCC04	00FFCC04	X
	2	0CF3EC04	0CF3EC04	0CAFCC00	0CAFCC00	07FFCC04	X
	3	00F3EC04	00F3EC0C	11BFCC47	03AFCC4C	3FFFCC06	X
	4	00F3EC00	0CF3EC00	X	0CAFCC00	FFFFCC85	
	5	37F7EC47	00F3EC4C	X	03AFCC4C	FFFFCC05	
	6	X	0CF3EC00	X	0CAFCC00	X	
	7	X	00F3EC4C	X	03AFCC4C	X	
	8		0CF3EC00		0CAFCC00	X	
	9		00F3EC44		33BFCC4F	X	
	A		03F3EC00		X	X	
	B		3FF7EC47		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

TABLE 3-9. Memory Controller Initializations For 20Mhz

Register	*Device Type*	*Init Value [hex]*	*Description*
BR0	All Flash SIMMs supported.	02800001	Base at 2800000, 32 bit port size, no parity, GPCM

OPERATING INSTRUCTIONS

TABLE 3-9. Memory Controller Initializations For 20Mhz

Register	*Device Type*	*Init Value [hex]*	*Description*
OR0	MCM29F020-90	FFE00D20	2MByte block size, all types access, CS early negate, 2 w.s.
	MCM29F040-90 SM732A1000A-9	FFC00D20	4MByte block size, all types access, CS early negate, 2 w.s.
	MCM29F080-90 SM732A2000-9	FF800920	8MByte block size, all types access, CS early negate, 2 w.s., Timing relax
	MCM29F020-12	FFE00D30	2MByte block size, all types access, CS early negate, 3 w.s.
	MCM29F040-12 SM732A1000A-12	FFC00D30	4MByte block size, all types access, CS early negate, 3 w.s.
	MCM29F080-12 SM732A2000-12	FF800930	8MByte block size, all types access, CS early negate, 3 w.s.
BR1	BCSR	02100001	Base at 2100000, 32 bit port size, no parity, GPCM
OR1		FFFF8110	32 KByte block size, all types access, CS early negate, 1 w.s.
BR2	All Dram SIMMs Supported	00000081	Base at 0, 32 bit port size, no parity, UPMA
OR2	MB321/2BT08TASN60	FFC00800	4MByte block size, all types access, initial address multiplexing according to AMA.
	MB324/8CT00TBSN60	FF000800	16MByte block size, all types access, initial address multiplexing according to AMA.
BR3[a]	MB322BT08TASN60	00400081	Base at 400000, 32 bit port size, no parity, UPMA
	MB328CT00TBSN60	01000081	Base at 1000000, 32 bit port size, no parity, UPMA
OR3	MB322BT08TASN60	FFC00800	4MByte block size, all types access, initial address multiplexing according to AMA
	MB328CT00TBSN60	FF000800	16MByte block size, all types access, initial address multiplexing according to AMA.
BR4 Compatible Mode	K4S643232-TC60	030000C1	Base at 3000000, on UPM B
OR4 Compatible Mode		FFC00A00	4 MByte block size, all types access, initial address multiplexing according to AMB.
BR4 MPC86x New Mode	K4S643232-TC60	0x000000C1	Base at 0x0, on UPM B
OR4 MPC86x New Mode		0xFC800A00	4 MByte block size, all types access, initial address multiplexing according to AMB.
BR5	Comm peripheral	0x02000401	

TABLE 3-9. Memory Controller Initializations For 20Mhz

Register	*Device Type*	*Init Value [hex]*	*Description*
OR5	Comm peripheral	0xFFF009A6	
MPTPR	All Dram SIMMs Supported	0400	Divide by 16 (decimal)
MAMR	MB321BT08TASN60	60A21114	refresh clock divided by 60, periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB322BT08TASN60	30A21114	refresh clock divided by 30, periodic timer enabled, type 2 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB324CT00TBSN60	60B21114	refresh clock divided by 60, periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
	MB328CT00TBSN60	30B21114	refresh clock divided by 30, periodic timer enabled, type 3 address multiplexing scheme, 1 cycle disable timer, GPL4 disabled for data sampling edge flexibility, 1 loop read, 1 loop write, 4 beats refresh burst.
MBMR	KS643232C-TC60	42802114[b]	refresh clock divided by 42, periodic timer enabled, type 0 address multiplexing scheme, 1 cycle disable timer, GPL4 enabled, 1 loop read, 1 loop write, 4 beats refresh burst.

a. BR3 is not initialized for MB321xx or MB324xx EDO DRAM SIMMs.
b. Assuming 16.67MHz BRGCLK

TABLE 3-10. UPMA Initializations for 60nsec EDO DRAMs @ 20MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset in UPM		0	8	18	20	30	3C
Contents @ Offset +	0	8FFFCC04	8FFFCC04	8FEFCC00	8FEFCC00	80FFCC84	33FFCC07
	1	08FFCC00	08FFCC08	39BFCC47	09AFCC48	17FFCC04	X
	2	33FFCC47	08FFCC08	X	09AFCC48	FFFFCC86	X
	3	X	08FFCC08	X	09AFCC48	FFFFCC05	X
	4	X	08FFCC00	X	39BFCC47	X	
	5	X	3FFFCC47	X	X	X	
	6	X	X	X	X	X	
	7	X	X	X	X	X	
	8		X		X	X	
	9		X		X	X	
	A		X		X	X	
	B		X		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

TABLE 3-11. UPMB Initialization for KS643232C-TC60 upto 32MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset In UPM		0	8	18	20	30	3C
Contents @ Offset +	0	0126CC04	0026FC04	0E26BC04	0E26BC00	1FF5FC84	7FFFFC07
	1	0FB98C00	10ADFC00	01B93C00	10AD7C00	FFFFFC04	X
	2	1FF74C45	F0AFFC00	1FF77C45	F0AFFC00	FFFFFC84	X
	3	X	F1AFFC00	X	F0AFFC00	FFFFFC05	X
	4	X	EFBBBC00	X	E1BBBC04	X	
	5	1FE77C34[a]	1FF77C45	X	1FF77C45	X	
	6	EFAABC34	X	X	X	X	
	7	1FA57C35	X	X	X	X	
	8		X		X	X	
	9		X		X	X	
	A		X		X	X	
	B		X		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

a. MRS initialization. Uses Free space.

TABLE 3-12. UPMB Initialization for KS643232C-TC60, 32+MHz - 50MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset In UPM		0	8	18	20	30	3C
Contents @ Offset +	0	1F07FC04	1F07FC04	1F27FC04	1F07FC04	1FF5FC84	7FFFFC07
	1	EEAEFC04	EEAEFC04	EEAEBC00	EEAEBC00	FFFFFC04	X
	2	11ADFC04	10ADFC04	01B93C04	10AD7C00	FFFFFC04	X
	3	EFBBBC00	F0AFFC00	1FF77C47	F0AFFC00	FFFFFC04	X
	4	1FF77C47	F0AFFC00	X	F0AFFC00	FFFFFC84	
	5	1FF77C34[a]	F1AFFC00	X	E1BBBC04	FFFFFC07	
	6	EFEABC34	EFBBBC00	X	1FF77C47	X	
	7	1FB57C35	1FF77C47	X	X	X	
	8		X		X	X	
	9		X		X	X	
	A		X		X	X	
	B		X		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

a. MRS initialization, Uses free space.

4 - Functional Description

In this chapter the various modules combining the MPC86xADS are described to their design details.

4•1 Reset & Reset - Configuration

There are several reset sources on the ADS:

1) Regular Power On Reset
2) Manual Soft-Reset
3) Manual Hard-Reset
4) MPC Internal Sources. (See the appropriate Spec or U/M)

4•1•1 Regular Power - On Reset

The regular power on reset operates, using a device - the DALAS **DS1818.** The reference voltage of this device is the MAIN VDDH bus of the MPC while the reset line asserted, is the HRESET* line.

When HRESET~ is asserted to the MPC, Hard-Reset configuration is made available to the MPC, via BCSR0. See 4•1•5•2 "Hard Reset Configuration" on page 39 and TABLE 4-10. "BCSR0 Description" on page 59.

4•1•2 Manual Soft Reset

To support application development not around the debug port and resident debuggers, a soft reset push-button is provided. (SW6) Depressing that button, asserts the SRESET* pin of the MPC, generating a SOFT RESET sequence.

When the SRESET~ line is asserted to the MPC, the Soft-Reset configuration is made available to the MPC, by the debug-port controller. See 4•1•5•3 "Soft Reset Configuration" on page 39.

4•1•3 Manual Hard Reset

To support application development not around the debug port, a Hard-Reset push-button is provided[A]. When the Soft Reset push-button (SW6) is depressed in conjunction with the ABORT push-button (SW5), the HRESET* line is asserted, generating a HARD RESET sequence. The button sharing is for economy and board space saving and does not effect in any way, functionality.

4•1•4 MPC Internal Sources

Since the HRESET* and SRESET* lines of the MPC are open-drain and the on-board reset logic drives these lines with open-drain gates, the correct operation of the internal reset sources of the MPC is facilitated. As a rule, an internal reset source asserts HRESET* and / or SRESET* for a minimum time of 512 system clocks. It is beyond the scope of this document to describe these sources, however Debug-Port Soft / Hard Resets which are part of the development system[B], are regarded as such.

4•1•5 Reset Configuration

During reset the MPC device samples the state of some external pins to determine its operation modes and pin configuration. There are 3 kinds of reset levels to the MPC each level having its own configuration sampled:

1) Power - On Reset configuration
2) Hard Reset configuration
3) Soft Reset Configuration.

A. It is not a dedicated button.
B. And therefore mentioned.

4•1•5•1 Power - On Reset Configuration

Just before PORESET* is negated by the external logic, the power-on reset configuration which include the MODCK(1:2) pins is sampled. These pins determine the clock operation mode of the MPC. Two clock modes are supported on the MPC86xADS:

1) 1:5 PLL operation via on-board clock generator.

   In this mode MODCK(1:2) are driven with'11' during[A] power on reset.

2) 1:513 PLL operation via on-board clock generator.

   In this mode MODCK(1:2) are driven with'00'. during power-on reset.

4•1•5•2 Hard Reset Configuration

During HARD reset sequence, when RSTCONF* pin is asserted, the MPC data bus state is sampled to acquire the MPC's hard reset configuration. The reset configuration word is driven by BCSR0 register, defaults of which are set during power-on reset. The BCSR0 drives half of the configuration word, i.e., data bits D(0:15) in which the reserved bits are designated RSRVxx. If the hard-reset configuration is to be changed[B], BCSR0 may be written with new values, which become valid after HARD reset is applied to the MPC.

On the ADS, the RSTCONF* line is always driven during HARD reset, i.e., no use is possible with the MPC's internal HARD reset configuration defaults.

The system parameters to which BCSR0 defaults during power-on reset and are driven at hard-reset, are listed below:

1) Arbitration: internal arbitration is selected.
2) Interrupt Prefix: The internal default is interrupt prefix at 0xFFF00000. It is overridden to provide interrupt prefix at address 0, which is located within the DRAM.
3) Boot Disable: Boot is enabled.
4) Boot Port Size: 32 bit boot port size is selected.
5) Initial Internal Space Base: Immediately after HARD reset, the internal space is located at $FF000000.
6) Debug pins configuration: PCMCIA port B[C] pins become PCMCIA port B pins.
7) Debug port pins configuration. Debug port pins are on the JTAG port.
8) External Bus Division Factor: 1:1 internal to external clocks' frequencies ratio is selected.

4•1•5•3 Soft Reset Configuration

The rising edge of SRESET* is used to configure the development port. Before the negation of SRESET*, DSCK[D] is sampled to determine for debug-mode enable / disable. After SRESET* is negated, if debug mode was enabled, DSCK is sampled again for debug-mode entry / non-entry.

DSDI is used to determine the debug port clock mode and is sampled after the negation of SRESET*.

The Soft Reset configuration is provided by the debug-port controller via the ADI I/F. Option is given to enter debug mode directly or only after exception.

4•2 Local Interrupter

The only external interrupt which is applied to the MPC via its interrupt controller is the ABORT (NMI),

A. The MODCK lines are in fact driven longer - by HRESET~ line.
B. With respect the ADS's power-on defaults.
C. Where they exist.
D. DSCK is configured at hard-reset to reside on the JTAG port.

which is generated by a push-button. When this button is depressed, the NMI input to the MPC is asserted. The purpose of this type of interrupt, is to support the use of resident debuggers if any is made available to the ADS. All other interrupts to the MPC, are generated internally by the MPC's peripherals and by the debug port.

To support external (off-board) generation of an NMI, the IRQ0* line which is routed as an NMI input, is driven by an open-drain gate. This allows for external h/w to also drive this line. If an external h/w indeed does so, it is compulsory that IRQ0* is driven by an open-drain (or open-collector) gate.

4•3 Clock Generator

There are 2 ways to clock the MPC on the MPC86xADS when using other device then MPC866:

1) 3 - 5MHz Clock generator U29 connected to CLK4IN input. 1:5 PLL mode. (SW1 / 4 OFF)
2) 32.768 KHz crystal resonator Y2 via EXTAL-XTAL pair of the MPC, 1:513 initial PLL multiplication factor. (SW1 / 4 ON)

The selection between the above modes is done using Dip-switch (SW1 / 4) with dual functionality: it is responsible to the combination driven to the MODCK lines during power-on reset and to the connection of the appropriate capacitor between MPC's XFC and VDDSYN lines to match the PLL's multiplication factor. When 1:5 mode is selected, a capacitor of 5.6nF is connected, while when 1:513 mode is selected a 0.56µF capacitor is connected parallel to it via a digital switch (U52). The capacitors' values are calculated to support a wider range of multiplication factors as possible.

When mode (2) above is selected, the output of the clock generator is gated from EXTCLK input and driven to'0' constantly so that a jitter-free system clock is generated.

On-board logic is clocked by the 20Mhz Clock Oscylator. This clock generator is used, so that on-board logic is always clocked, even when the MPC is removed from its socket.

For MPC866 device there is SW3 / 1 this switch is selecting the digital switch U52 to connect XFC to GND or to let it get one of the above capacitors for the PLL. If SW3 / 1 is ON it select MPC other then MPC866 if it is OFF it select MPC866.

4•4 Buffering

As the ADS meant to serve also as a hardware development platform, it is necessary to buffer the MPC from the local bus, so the MPC's capacitive drive capability is not wasted internally and remains available for user's off-board applications via the expansion connectors.

Buffers are provided for address and strobe lines while transceivers are provided for data. Since the capacitive load over dram's address lines might[A] exceed 200 pF, the dram address lines are separately buffered. Use is done with 74LCX buffers which are 3.3V operated and are 5V tolerant. This type of buffers reduces noise on board due to reduced transitions' amplitude.

To further reduce noise and reflections, series resistors are placed over dram's address and strobe lines.

The data transceivers open only if there is an access to a valid[B] [C] board address or during Hard - Reset configuration[D]. That way data conflicts are avoided in case an 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.

A. Depended on dram SIMM's internal structure.
B. An address which covered in a Chip-Select region.
C. Except for SDRAM, which is Unbuffered.
D. To allow a configuration word stored in Flash memory become active.

4•5 Chip - Select Generator

The memory controller of the MPC is used as a chip-select generator to access on-board[A] memories, saving board's area reducing cost, power consumption and increasing flexibility. To enhance off-board application development, memory modules (including the BCSRx) may be disabled via BCSR1[B] 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 associated local memory is disabled.

When a CS region is disabled via BCSR1, the local data transceivers do not open during access to that region, avoiding possible[C] contention over data lines.

The MPC's chip-selects assignment to the various memories / registers on the ADS are as shown in TABLE 4-1. "MPC86xADS Chip Selects' Assignment" below:

TABLE 4-1. MPC86xADS Chip Selects' Assignment

Chip Select:	*Assignment*
CS0*	Flash Memory
CS1*	BCSR
CS2*	DRAM Bank 1
CS3*	DRAM Bank 2[a]
CS4*	SDRAM
CS5*	Communication Peripherals
CS(6-7)*	Unused, user available

a. If exists.

4•6 DRAM

The DRAM EDO is not supplied with the board. the user can put its own DRAM EDO on U38 DRAM SIMM.The MPC86xADS is able to operate with 4 MBytes of 60nsec delay EDO Dram SIMM. Support is given to any 5V powered FPM / EDO Dram SIMM configured as 1M X32 upto 2 X 4M X 32, with 60 nsec or 70nsec delay.

All dram configurations are supported via the Board Control & Status Register (BCSR), i.e., DRAM size (4M to 32M) and delay (60 / 70 nsec) are read from BCSR2 and the associated registers (including the UPM) are programmed accordingly.

Dram timing control is performed by UPMA of the MPC via CS2 (and CS3 for a dual-bank SIMM) region(s), i.e., RAS and CAS signals' generation, during normal[D] access as well as during refresh cycles and the necessary address multiplexing[E] are performed using UPMA. CS2* and CS3* signals are buffered from the DRAM and each is split to 2 to overcome the capacitive load over the Dram SIMM RAS lines.

The DRAM module may enabled / disabled at any time by writing the DRAMEN~ bit in BCSR1. See TABLE

A. Phriphrials And off-board. See further.
B. After the BCSR is removed from the local memory map, there is no way to access it but to re-apply power to the ADS.
C. During read cycles.
D. Normal i.e.: Single Read, Single Write, Burst Read & Burst Write.
E. Taking into account support for narrower bus widths.

4-11. "BCSR1 Description" on page 60.

Note: The DRAM is not populated on the board the user can populate is own DRAM in order to expand its memory or to run old SW which run on the old MPC8xxFADS.

4•6•1 DRAM 16 Bit Operation

To enhance evaluation capabilities, support is given to Dram with 16-bit and 32-bit data bus width. That way users can tailor dram configuration, to get best fit to their application requirements. When the DRAM is in 16 bit mode, half of it can not be used, i.e., the memory portion that is connected to data lines D(16:31).

To configure the DRAM for 16 bit data bus width operation, the following steps should be taken:

1) Set the Dram_Half_Word bit in BCSR1 to Half-Word. See TABLE 4-11. "BCSR1 Description" on page 60
2) The Port Size bits of BR2~ (and of BR3~ for a 2-bank DRAM simm) should be set to 16 bits.
3) The AM bits in OR2 register should be set to **1/2** of the nominal **single-bank** DRAM simm volume or to **1/4** of the nominal **dual-bank** DRAM simm volume.

If a Dual-Bank DRAM simm is being used:

4) The Base-Address bits in BR3 register should be set to DRAM_BASE + **1/4** Nominal_Volume, that is, if a contiguous block of DRAM is desired.
5) The AM bits of OR3 register, should be set to **1/4** Nominal_Volume.

If the above is executed out of running code, than this code **should not reside on the DRAM** while executing, otherwise, erratic behavior is likely to be demonstrated, resulting in a system crash.

4•6•2 DRAM Performance Figures

The projected performance figures for the dram are shown in TABLE 4-2. "Regular DRAM Performance

Figures" on page 43 and in TABLE 4-3. "EDO DRAM Performance Figures" on page 43.

TABLE 4-2. Regular DRAM Performance Figures

	Number of System Clock Cycles			
System Clock Frequency [MHz]	*50*		*25*	
DRAM Delay [nsec]	*60*	*70*	*60*	*70*
Single Read	6	6	3	4
Single Write	4	4	3	3
Burst Read	6,2,3,2	6,3,2,3	3,2,2,2	4,2,2,2
Burst Write	4,2,2,2	4,2,2,2	3,1,2,2	3,2,2,2
Refresh	21[a b]	25[a b]	13[a b]	13[a b]

a. Four-beat refresh burst.
b. Not including arbitration overhead.

TABLE 4-3. EDO DRAM Performance Figures

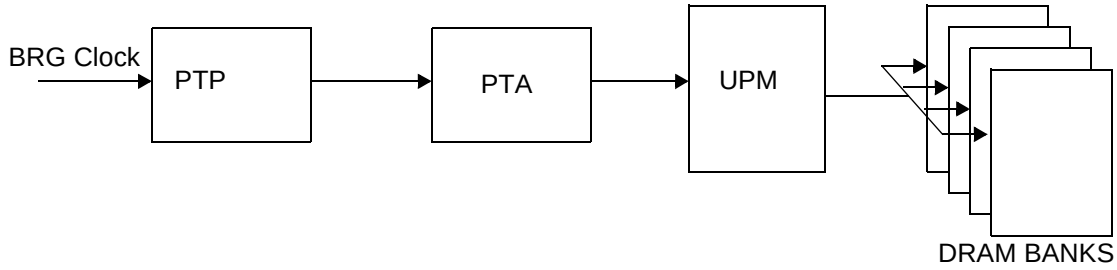
	Number of System Clock Cycles			
System Clock Frequency [MHz]	*50*		*25*	
DRAM Delay [nsec]	*60*	*70*	*60*	*70*
Single Read	6	6	3	4
Single Write	4	4	2	3
Burst Read	6,2,2,2	6,3,2,2	3,1,1,1	4,1,2,2
Burst Write	4,2,2,2	4,2,2,2	2,1,1,1	3,2,2,2
Refresh	21[a b]	25[a b]	13[a b]	13[a b]

a. Four-beat refresh burst.
b. Not including arbitration overhead.

4•6•3 Refresh Control

The refresh to the dram is a CAS before RAS refresh, which is controlled by UPMA as well. The refresh logic is clocked by the MPC's BRG clock which is not influenced by the MPC's low-power divider.

FIGURE 4-1 Refresh Scheme



As seen in FIGURE 4-1 "Refresh Scheme" above, the BRG clock is twice divided: once by the PTP (Periodic Timer Prescaler) and again by another prescaler - the PTA, dedicated for each UPM. If there are more than one dram banks, than refresh cycles are performed for consecutive banks, therefore, refresh should be made faster. The formula for calculation of the PTA is given below:

$$PTA = \frac{Refresh\_Period \ \mathbf{X} \ Number\_Of\_Beats\_Per\_Refresh\_Cycle}{Number\_Of\_Rows\_To\_Refresh \ \mathbf{X} \ T\_BRG \ \mathbf{X} \ MPTPR \ \mathbf{X} \ Number\_Of\_Banks}$$

Where:

- PTA - Periodic Timer A filed in MAMR. The value of the 2'nd divider.
- Refresh_Period is the time (usually in msec) required to refresh a dram bank
- Number_Of_Beats_Per_Refresh_Cycle: using the UPM looping capability, it is possible to perform more than one refresh cycle per refresh burst (in fact upto 16).
- Number_Of_Rows_To_Refresh: the number of rows in a dram bank
- T_BRG: the cycle time of the BRG clock
- MPTPR: the value of the periodic timer prescaler (2 to 64)
- Number_Of_Banks: number of dram banks to refresh.

If we take for example a MCM36200 SIMM which has the following data:

- Refresh_Period == 16 msec
- Number_Of_Beats_Per_Refresh_Cycle: on the ADS it is 4.
- Number_Of_Rows_To_Refresh == 1024
- T_BRG == 20 nsec (system clock @ 50 Mhz)
- MPTPR arbitrarily chosen to be 16
- Number_Of_Banks == 2 for that SIMM

If we assign the figures to the PTA formula we get the value of PTA should be 97 decimal or 61 hex.

4•6•4 Variable Bus-Width Control

Since a port's width determines its address lines' connection scheme, i.e., the number of address lines required for byte-selection varies (1 for 16-bit port and 2 for 32-bit port) according to the port's width, it is necessary to change address connections to a memory port if its width is to be changed. E.g.: if a certain memory is initially configured as a 32-bit port, the list significant address line which is connected to that memory's A0 line should be the MPC's A29. Now, if that port is to be reconfigured as a 16-bit port, the LS address line becomes A30.

If a linear[A] address scheme is to be maintained, all address lines connected to that memory are to be shifted one bit, this obviously involves extensive multiplexing (passive or active). If linear addressing

scheme is not a must, than only minimal multiplexing is required to support variable port width.

In TABLE 4-4. "DRAM ADDRESS CONNECTIONS" below, the ADS's dram address connection scheme is presented:

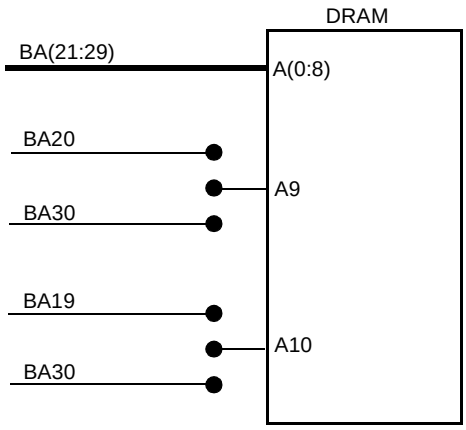
TABLE 4-4. DRAM ADDRESS CONNECTIONS

Width	*32 - Bit*		*16 - Bit*	
	Depth		*Depth*	
Dram ADD	*4 M*	*1 M*	*4 M*	*1 M*
A0	BA29	BA29	BA29	BA29
A1	BA28	BA28	BA28	BA28
A2	BA27	BA27	BA27	BA27
A3	BA26	BA26	BA26	BA26
A4	BA25	BA25	BA25	BA25
A5	BA24	BA24	BA24	BA24
A6	BA23	BA23	BA23	BA23
A7	BA22	BA22	BA22	BA22
A8	BA21	BA21	BA21	BA21
A9	BA20	BA20	BA20	BA30
A10	BA19		BA30	

As can seen from the table above, most of the address lines remain fixed while only 2 lines (the shaded cells) need switching. The switching scheme is shown in FIGURE 4-2 "DRAM Address Lines' Switching Scheme" on page 46. The switches on that figure are implemented by active multiplexers controlled by the BCSR1/Dram_Half_Word* bit.

A. Consequent addresses lead to adjacent memory cells

FIGURE 4-2 DRAM Address Lines' Switching Scheme



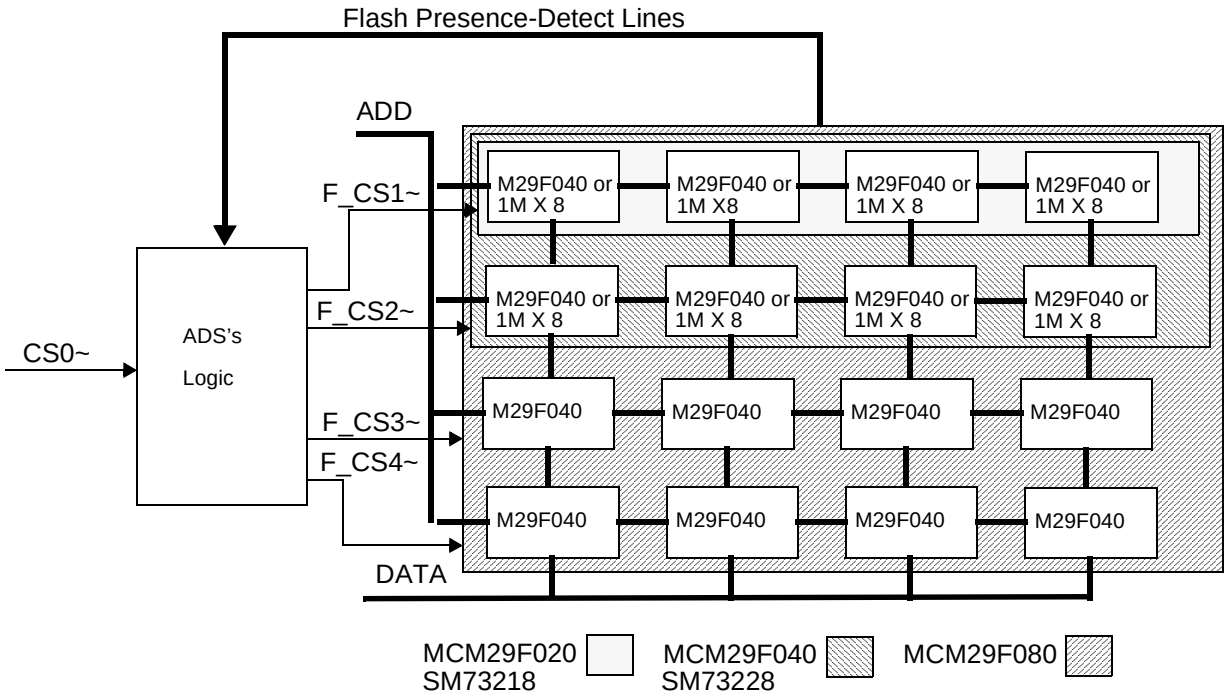
4•7 Flash Memory SIMM

The MPC86xADS is provided with 2Mbyte of 90 nsec flash memory SIMM - the MCM29020 by Motorola. Support is given also to 4MBytes MCM29F040, 8 MBytes MCM29F080, 4 MBytes SM73218 and to 8 MBytes SM73228 by Smart Technology. The Motorola SIMMs are internally composed of 1, 2 or 4 banks of 4 Am29F040 compatible devices, while the Smart SIMMs are arranged as 1 or 2 banks of four 28F008 devices by Intel. The flash SIMM resides on an 80 pin SIMM socket.

To minimize use of MPC's chip-select lines, only one chip-select line (CS0~) is used to select the flash as a whole, while distributing chip-select lines among the internal banks is done via on-board programmable logic, according to the Presence-Detect lines of the Flash SIMM inserted to the ADS.

Functional Description

FIGURE 4-3 Flash Memory SIMM Architecture



The access time of the Flash memory provided with the ADS is 90 nsec, however, 120 nsec devices may be used as well. Reading the delay section of the Flash SIMM Presence-Detect lines, the debugger es-tablishes (via OR0) the correct number of wait-states (considering 50MHz system clock frequency).

The Motorola SIMMs are built of AMD's Am29F0X0 devices which are 5V programmable, i.e., there is no need for external programming voltage and the flash may be written almost[A] as a regular memory.

The SMART parts however, require 12V ± 0.5% programming voltage to be applied for programming. If on-boards programming of such device is required, a 12V supply needs to be connected to the ADS (P12). Otherwise, for normal[B] Flash operation, 12V supply is not required.

The control over the flash is done using the GPCM and a dedicated CS0~ region, controlling the whole bank. During hard - reset initializations, the debugger reads the Flash Presence-Detect lines via BCSR2 and decides how to program BR0 & OR0 registers, within which the size and the delay of the region are determined.

The performance of the flash memory is shown in TABLE 4-5. "Flash Memory Performance Figures" below :

TABLE 4-5. Flash Memory Performance Figures

	Number of System Clock Cycles			
System Clock Frequency [MHz]	*50*		*25*	
Flash Delay [nsec]	*90*	*120*	*90*	*120*
Read / Write[a] Access [Clocks]	8	10	4	5

A. A manufacturer specific dedicated programming algorithm should be implemented during flash programming.
B. I.e., Read-Only.

a. The figures in the table refer to the actual write access. The write operation continues internally and the device has to be polled for operation completion.

The Flash module may disabled / enabled at any time by writing'1' /'0' the FlashEn~ bit in BCSR1.

4•8 Synchronous Dram

To enhance performance, especially in higher operation frequencies - 8 MBytes of SDRAM is provided on board. The SDRAM is unbuffered from the MPC bus and is configured as 4 X 512K X 32. Use is done with MT48LC2M32B2 chips by Micron or compatibles.

To enhance performance, the SDRAM is unbuffered from the MPC, saving the delay associated with address and data buffers. Since only 1 memory chip is involved, it does not adversely effect overall system performance. The SDRAM does not reside on a SIMM but is soldered directly to the ADS pcb. The SDRAM may be enabled / disabled at any time by writing 1 / 0 to the SDRAMEN bit in BCSR1. See TABLE 4-11. "BCSR1 Description" on page 60.

The SDRAM's timing is controlled by UPMB via its assigned CS (See TABLE 4-1. "MPC86xADS Chip Selects' Assignment" on page 41) line. Unlike a regular dram the synchronous dram has a CS input in addition to the RAS and CAS signals.

The sdram connection scheme is shown in FIGURE 4-4 "SDRAM Connection Scheme" on page 50.

The SDRAM's performance figures, are shown in TABLE 4-8. "Estimated SDRAM Performance Figures":

This SDRAM has 11 ROW and 8 Column. The suggested interface between an MPC8xx and an SDRAM is illustrated in Figure bellow is clear that this is a glue less interface. For a 32 bit bus, one 32 bit SDRAM devices is connected.The control is driven by the UPMB on the MPC8xx, so the CS on the SDRAM is interfaced to CS4 on the MPC8xx. Any other chip select line excluding CS0 would do.The DQM signals of the used SDRAM devices select byte lanes and are connected to the appropriate Byte Strobe (BS0:3)signals on the MPC8xx. A10 SD is connected to GPL0,since this has the functionality to either drive an address on the line, or a defined level. This is required as A10 SD acts as both an address line and a control line. RAS and CAS are generated by GPL1 and GPL2 respectively. The WE is generated by GPL3. CLK is driven by the MPC8xx's CLKOUT signal which is a reference point with respect to the MPC8xx's Memory Controller. As the SDRAM used in the example has 2048 rows and 256 columns, we have to use 11 row address lines and 8 column address lines.The BS line are connected to line A10, A9 MPC and are used as high order address bit. Please remember that the MPC8xx address lines have a different numbering scheme than the SDRAM address lines when reading the address line mapping for 32 bit in the Table below.

TABLE 4-6. SDRAM ADD pin refer to MPC8xx Pins

MPC8xx	*SDRAM*
A9, A10	BS1, BS0
A11:A21	11 ROW
A22:A29	8 Column

Via the UPM Register AMx =0b000, the address bits A11:21 MPC are mapped to lines A19:29 MPC as row addresses. As we start with line A21 MPC to connect to A8 SD, A20 MPC to A9 SD we need

to provide the left over row address A10 SD not by using line A19 MPC (which would show A10 MPC as multiplexed row address), but by using GPL0 as described above. In the UPM Register MxMR, we program GPL0 to show A10 MPC to complete row addressing.
As in this case there are 4 banks in the SDRAM device, Sometimes, a single 32 bit SDRAM bank is not enough memory for an application. Connecting multiple 32 bit SDRAM based banks to the MPC8xx is fairly straightforward. Extending the interface described above is easy.

Above, the most significant row address bit is connected to BS SD.A10 MPC is used due to the address size of

19 bits (8/11 address multiplex!) covered by the example SDRAM device. For an SDRAM device with two

BS lines,BS0 SD and BS1 SD, we simply use the next address bit,e.g.,A10, A9 MPC, with more significance to keep the memory mapping linear. In essence, we use address lines for the binary encoding of the bank

selection.

TABLE 4-7. SDRAM Connected to MPC

MPC output ADD	*SDRAM ADD*	*MPC Internal Column ADD*	*MPC Internal Row ADD*	
A29	A0	A29	A21	
A28	A1	A28	A20	
A27	A2	A27	A19	
A26	A3	A26	A18	
A25	A4	A25	A17	
A24	A5	A24	A16	
A23	A6	A23	A15	
A22	A7	A22	A14	
A21	A8		A13	
A20	A9		A12	
GPL0	A10 (AP)		A11	
A10 Note1	NC(A11)			
A10 / A9 Note1	BS0		A10	
A9 / A8 Note 1	BS1		A9	

Note1 Incase of the user wonts to change the SDRAM to a larger one 16M A11 of the SDRAM is connected to A10, this connection is already exist on the board layout, the user has to connect BS0, BS1 to MPC add

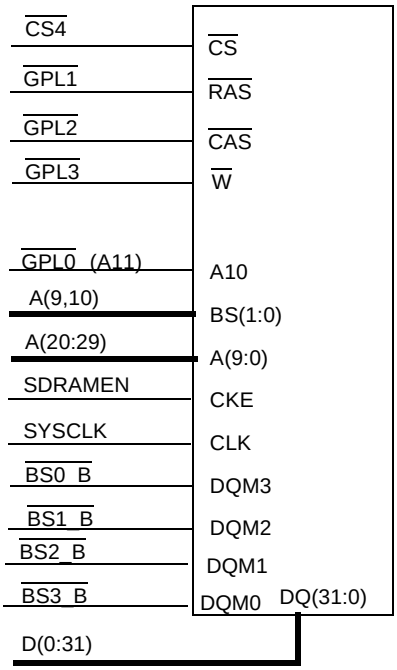
A9 & A8 this will be done by moving a resistors on board called RJ1 and RJ2 from pins 1, 2 to 2, 3.

TABLE 4-8. Estimated SDRAM Performance Figures

	Number of System Clock Cycles	
System Clock Frequency [MHz]	*50*	*25*[a]
Single Read	5	3
Single Write	3+1[b]	2 + 1[b]
Burst Read	5,1,1,1	3,1,1,1
Burst Write	3,1,1,1 + 1[b]	2,1,1,1 + 1[b]
Refresh	21 [c]	13 [b]

a. In fact upto 32MHz.
b. One additional cycle for RAS precharge
c. 4-beat Refresh Burst, not including arbitration overhead.

FIGURE 4-4 SDRAM Connection Scheme



4•8•1 SDRAM Programming

After power-up, the sdram needs to be initialized by means of programming, to establish its mode of oper-

ation. The Sdram is programmed by issuing a Mode Register Set command. During that command, data is passed to the Mode Register through the Sdram's address lines. This command is fully supported by the UPM by means of a dedicated Memory Address Register and the UPM command run option.

Mode Register programming values are shown in TABLE 4-9. "SDRAM's Mode Register Programming" below: In order to operate the SDRAM in higher speed then 50Mhz the user should read the application note in "http://e-www.motorola.com/brdata/PDFDB/docs/AN2066.pdf"
and also use the MPC860COD09 MPC860 UPM Programming Tool - UPM860
and MPC860COD10 UPM860 Manual for MPC860 UPM Programming Tool on web page
http://e-www.motorola.com/webapp/sps/site/prod_summary.jsp?code=MPC860&nodeId=01M98657

TABLE 4-9. SDRAM's Mode Register Programming

	Value @ Frequency	
SDRAM Option	*50MHz*	*25MHz*
Burst Length	4	4
Burst Type	Sequential	Sequential
CAS Latency	2	1
Write Burst Length	Burst	Burst

4•8•1•1 SDRAM Initializing Procedure

After Power-up the SDRAM needs to be initialized in a certain manner, described below:

1) UPMB should be programmed with values described in TABLE 3-11. "UPMB Initialization for KS643232C-TC60 upto 32MHz" on page 36 or in TABLE 3-12. "UPMB Initialization for KS643232C-TC60, 32+MHz - 50MHz" on page 37.
2) Memory controller's MPTPR, MBMR, OR4 and BR4 registers should be programmed according to TABLE 3-9. "Memory Controller Initializations For 20Mhz" on page 32 or TABLE 3-5. "Memory Controller Initialization For 50Mhz with DRAM-EDO" on page 26.
3) MAR should be set with proper value (0x48 for upto 32MHz or 0x88 for 32 - 50 MHz)
4) MCR should be written with 0x80808105 to run the MRS command programmed in locations 5 - 8 of UPMB.
5) MBMR's TLFB field should be changed to 8, to constitute 8-beat refresh Bursts.
6) MCR should be written with 0x80808130 to run the refresh sequence (8 refresh cycles are performed now)
7) MBMR's TLFB field should be restored to 4, to provide 4-beat refresh Bursts for normal operation. The SDRAM is initialized and ready for operation.

4•8•2 SDRAM Refresh

The SDRAM is refreshed using its auto-refresh mode. I.e., using UMPB's periodic timer, a burst of four auto-refresh commands is issued to the SDRAM every 62.4 μsec, so that all 2048 SDRAM rows are refreshed within specified 32.8 msec.

4•9 Communication Ports

Since the ADS board is meant to serve all the MPC86x family, it contains all the modules that are possible to be configured on the MPC86x. The various communication ports are as below:

- SCC1 10BaseT Ethernet.
- SCC3 IRD
- SMC1, SMC2 RS232.
- TDMB Serial ATM on E1/T1
- FETHC - Fast Ethernet Controller. On PCMCIA Port or Port - D
- ATM mux/Split, Mux On Port - D or Split on both PCMCIA and Port - D MultyPhy or SinglePhy

4•9•1 Ethernet Port

An Ethernet port with T.P. (10-Base-T) I/F is provided on the MPC86xADS. The comm. port over which this port resides, is determined according to the MPC type[A]. Use is done with the MC68160 EEST 10-base-T transceiver, used also with the MPC86xADS.

To allow alternative use of the Ethernet's SCC pins, they appear at the expansion connectors. Ports expansion connector (P7) of the this board, while the Ethernet transceiver may be Disabled / Enabled at any time by writing'1' /'0' to the EthEn~ bit in BCSR1.

4•9•2 Infra-Red Port

An infra-Red communication port is provided with the ADS - the Temic's TFDS 6000 integrated transceiver, which incorporates both the receiver and transmitter optical devices with the translating logic and supports Fast IrDA (upto 4 Mbps). The comm. port over which this port resides, is determined according to the MPC type[A].

To allow alternative use of the I/Or's SCC or its pins, the infra-red transceiver may be disabled / enabled at any time, by writing '1' /'0' to the IrdEn~ bit in BCSR1, while all pins appear expansion connector, P7 of this board.

4•9•2•1 Infra-Red Port Rate Range Selection

The TFDS6000 has 2 bit-rate ranges:

1) 9600 Bps to 1.2 MBps
2) 1.2 MBps to 4 MBps.

Selection between the 2 ranges is determined by the state of the transceiver's TX input on the falling edge of IrdEn~.

When TX input is LOW at least 200 nsec before the falling edge of IrdEn~, then, the LOWER range is selected. If TX is HIGH for that period of time, then, the HIGHER range is selected.

4•9•3 RS232 Ports

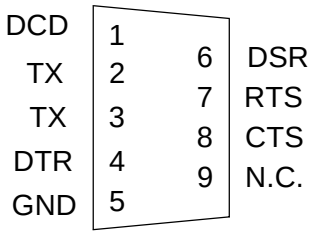
To assist user's applications and to provided convenient communication channels with both a terminal and a host computer, two identical RS232 ports are provided on the ADS. The MPC's communication ports to which these RS232 ports is routed, is established according to the type of MPC. Use is done with MAX3241ECAI transceivers which generates RS232 levels internally using a single 3.3V supply and are equipped with OE and shutdown mode. When the $\overline{\text{RS232EN1}}$ or $\overline{\text{RS232EN2}}$ bits in BCSR1 are asserted (low), the associated transceiver is enabled. When negated, the associated transceiver enters standby mode, in which the receiver outputs are tri-stated, enabling use of the associated port's pins, off-board via

A. I.e., routing is done on the daughter board.

the expansion connectors. The SMC2 are conflict with the ATM address pins. In order to work with SMC2 the user should work in ATM single phy, look J4 description.

Use is done with 9 pins, female D-Type stacked connector, configured to be directly (via a flat cable) connected to a standard IBM-PC like RS232 connector.

FIGURE 4-5 RS232 Serial Ports' Connector



4•9•3•1 RS-232 Ports' Signal Description

In the list below, the direction','I/O' are relative to the ADS board. (I.e.'I' means input to the ADS)

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

4•9•4 Utopia Bus and MII Operation & Interface

The MPC862DB supports multy phy, single phy, muxed and split Utopia bus operation. The muxing logic is shown in FIGURE 4-6 "UTOPIA and MII Buses interfaces & control" on page 54 and is realized by using a series of logic switches on the data lines as appropriate. In the split bus configuration the transmit and receive data signals are separated. In this configuration the MPC86xADS limits the operation of several other port functions including disabling of the MII, 100BaseT Ethernet interface. In the muxed bus configuration the MPC86xADS multiplexes ("mux"s) the transmit and receive data, soc and clock utopia signals. This configuration allows the MPC86xADS to provide the operation of other port functions such as 100BaseT Ethernet (MII). The board also allows the MPC862/6 to be exercised as both a Utopia level 2 "master" and as a Utopia level 2 "slave" device. In master mode the MPC862/6 Utopia interface is software programmable to use either the bus muxing (combined TX and RX bus) or the split bus configurations (separate TX and RX bus). The HW on the board can select the mode of operation by set of switches, SW2(2,3,4) select the ATM FastEthernet mode of operation.The ATM25 interface and the ATM155 interface can be used as slave devices. The ATM on the MPC should be configured to output the Utopia clock.

NOTE:

In the split bus configuration (master or slave modes)
the 100BaseT Ethernet interface is necessarily disabled.

The MPC862/6 provides simultaneous control and support of multiple physical interfaces connected to the Utopia bus. The Utopia bus addressing allows up to 31 physical devices to be connected. In this board there is use of only one ATM25 and one ATM155. only 4 of 10 ATM ADDRESS pin is used on the Board 2 for receive an 2 for transmit. PB16 for RXADD0, PB17 for RXADD1, PB20 for TXADD0 and PB21 for

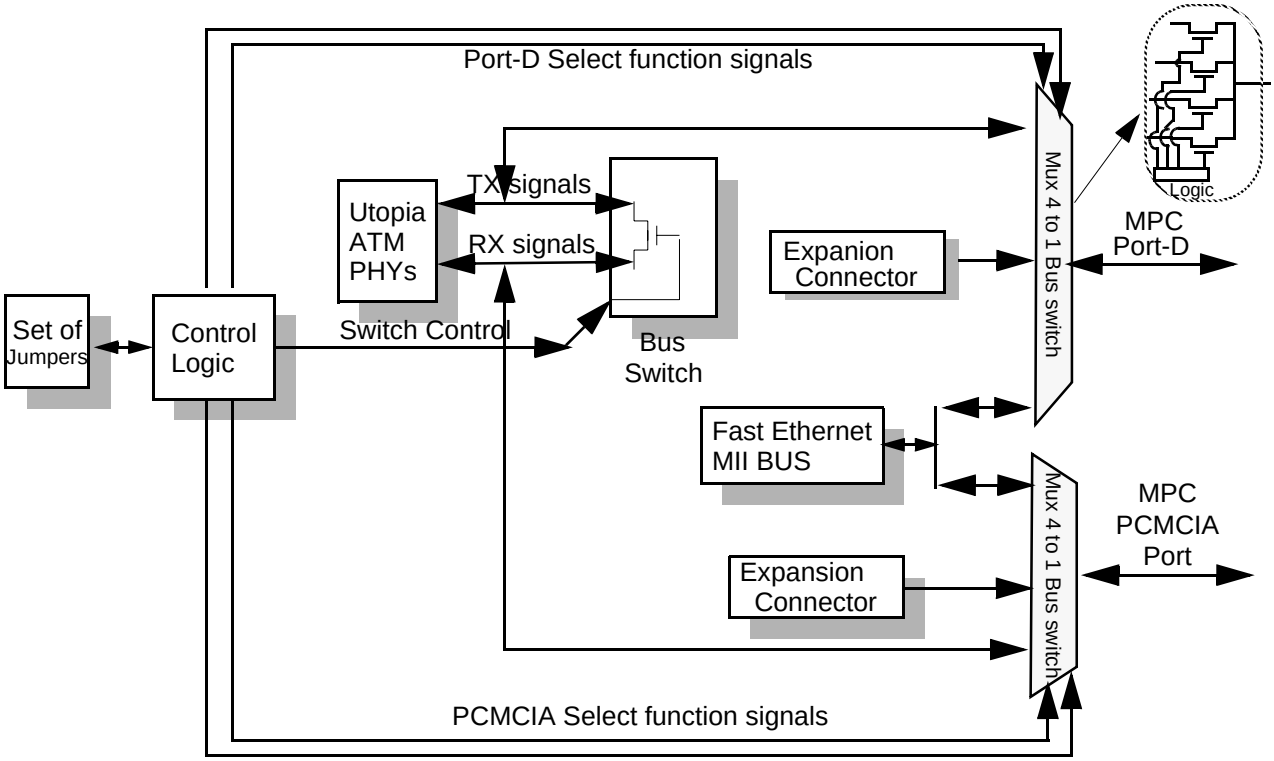
TXADD1.
Note in case of using ATM, RS232-2 can not be use. (marked as RS2 on the board silk)

NOTE:

The device interrupts are ANDed together to provide a single interrupt to the MPC86x.

The Fast Ethernet will be able to connect to PCMCIA port or to Port D. It will be according to a control logic. The figure below shows all the connections to Port-D and PCMCIA port with the control logic that selects each mode of operation.

FIGURE 4-6 UTOPIA and MII Buses interfaces & control



4•9•5 ATM25

ATM 25M PHY is connected to Utopia bus. Supports MUX mode through port-D for transmit and receive utopia signals or in split modetransmit signals through Port-D and receive utopia signals through PCMCIA port. Implementation is done using IDT IDT77107 device. The ATM25 memory mapped to ADD 0x2000000.

4•9•6 ATM155

ATM 155 PHY is connected to Utopia bus. Supports MUX mode through port-D for transmit and receive utopia signals or in split modetransmit signals through Port-D and receive utopia signals through PCMCIA port. I. Implementation is done using NEC uPD98404 device.
Note: in order to operate the NEC-uPD98404 correctly the SW should init the ATM on the MPC devise and also init the PIO and then reset the ATM PHY by driving to address 0x2000300 the value 0x08 (reset the uPD98404). The ATM155 is mapped to 0x2000100.

4•9•7 Serial ATM (Over E1/T1)

The MPC862/6 has the capability to perform ATM TC layer. The ATM layer is a serial ATM output connected to TDMB. Implementation is done using Infineon PEB2256 E1/T1 PHY device. The MPC862/6 drive it through TDMB. The PEB2256 is mapped to ADD 0x2000200. 2.048M osc supplied in the board box for E1.

4•9•8 Fast Ethernet.

The MPC8626ADS provides a 100BaseT Ethernet interface connected to the MPC862/6 via an MII inteface. The MPC862 provides simultaneous operation of both fast ethernet and the Utopia bus.

The board provides the necessary hardware interfaces and bussing logic to support this simultaneous feature.
The phy address is 0b01111.

Note: inorder to configure the board for the desire configuration ATM, Fast-Ethernet use TABLE 2-2. "ATM & Fast-Ethernet configuration." on page 11 table.

4•10 PCMCIA Port

To enhance PCMCIA i/f development, a dedicated PCMCIA port is provided on the ADS. Support is given to 5V only PC-Cards, PCMCIA standard 2.1+ compliant. All the necessary control signals are generated by the MPC itself. To protect MPC signals from external hazards, and to provide sufficient drive capability, a set of buffers and latches is provided over PC-Card's address, data & strobe lines.

To conform with the design spirit of the ADS, i.e., making as much as possible MPC resources available for external application development, input buffers are provided for input control signals, controlled by the PCC_EN~ bit in BCSR1, so the PCMCIA port may be Disabled / Enabled at any time, by writing '1' / '0' to that bit. When the PCMCIA channel is disabled, its associated pins are available for off-board use via the expansion connectors.

A loudspeaker is provided on board and connected to SPKROUT line of the MPC. The speaker is buffered from the MPC and low-pass filtered. When the PCC_EN~ bit in BCSR1 is negated (high) the speaker buffer is tri-stated so the SPKROUT signal of the MPC may be used for alternate function.

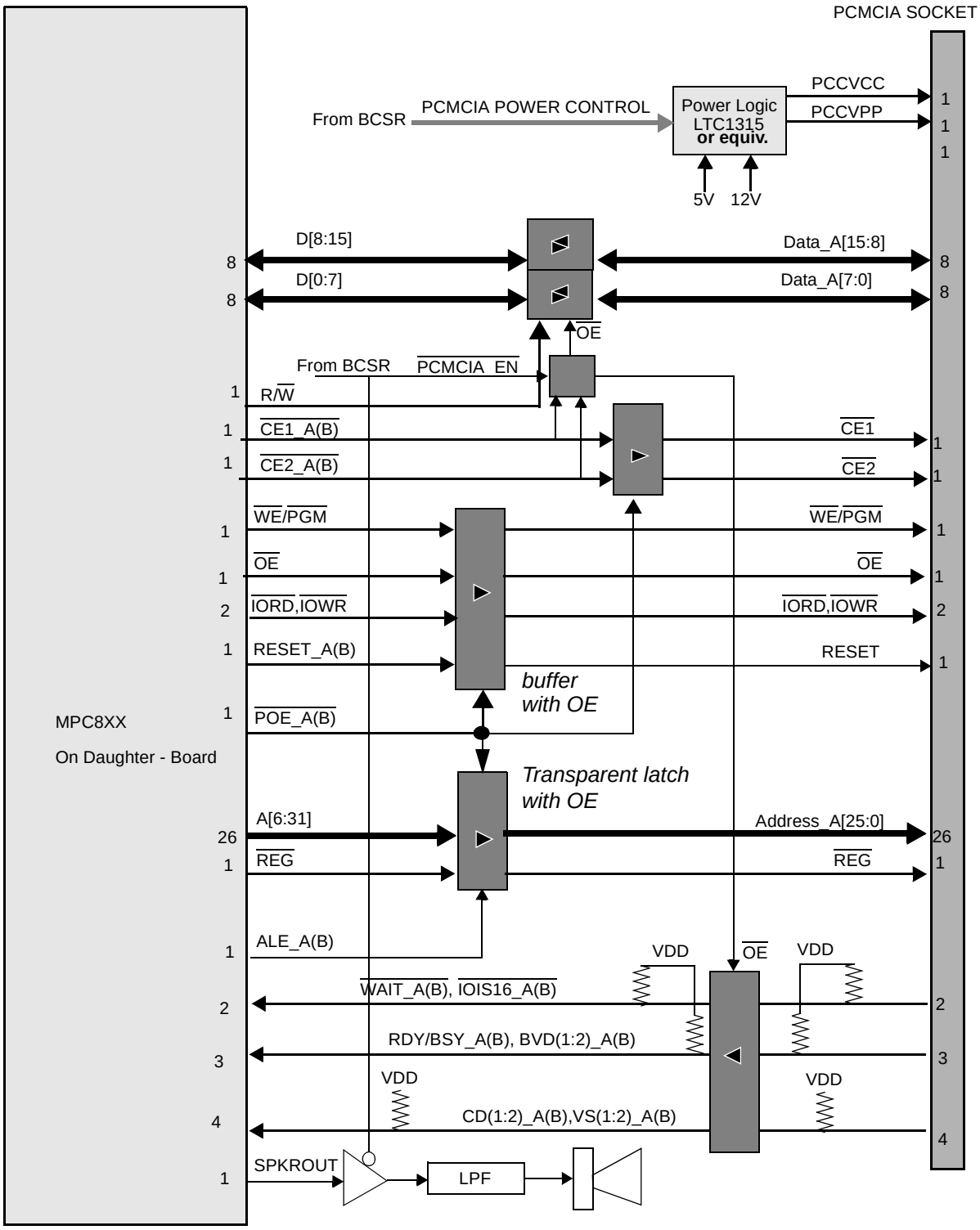
Since it is not desirable[A] to apply control signals to unpowered PC-Card, the strobe / data signal buffers / transceivers are tri-stated and may be driven only when the PC-Card is powered.

The block diagram of the PCMCIA port is shown in FIGURE 4-7 "PCMCIA Port Configuration" on page 56.

A. This since the PC-Card might have protection diodes on its inputs, which will force down input signals regardless of their driven level.

Functional Description

FIGURE 4-7 PCMCIA Port Configuration



4•10•1 PCMCIA Power Control

To support hot-insertion[A] the socket's power is controlled via a dedicated PCMCIA power controller the LTC1315 made by LINEAR TECHNOLOGY. This device, controlled by BCSR1, switches 12V VPP for card programming and controls gates of external MOSFET transistors, through which the PC Card VCC is switched.

When a card is inserted while the channel is enabled via BCSR1, i.e., both of the CD(1:2)* (Card Detect) lines are asserted (low), the status of the voltage select lines VS(1:2)* should be read to determine the PC Card's operation voltage level according to which, PCCVCC(0:1) bits in BCSR1 should be set, to drive the correct VCC (5V) to the PC-Card.

When a card is being removed from the socket while the channel is enabled via BCSR1, the negation of CD1~ and CD2~ may be sensed by the MPC and power supply to the card may be cut.

WARNNING

Any application S/W handling the PCMCIA channel must check the Voltage-Sense lines before Power is applied to the PC-Card. Otherwise, if 5V power is applied to a 3.3V-Only card, permanent damage will be inflicted to the PC-Card.

4•11 Board Control & Status Register - BCSR

Most of the hardware options on the MPC86xADS are controlled or monitored by the BCSR, which is a 32[B] bit wide read / write register file. The BCSR is accessed via the MPC's CS1 region and in fact includes 5 registers: BCSR0 to BCSR4. Since the minimum block size for a CS region is 32KBytes, BCSR0 - BCSR4 are multiply duplicated within that region. See also TABLE 3-3. "Memory Map in Compatible Mode" on page 23.

The following functions are controlled / monitored by the BCSR:

1) MPC's Hard Reset Configuration.
2) Flash Module Enable / Disable
3) Dram Module Enable / Disable
4) Dram port width - 32 bit / 16 bit.
5) SDRAM Module Enable / Disable.
6) Ethernet port Enable / Disable.
7) Infra-Red port Enable / Disable.
8) RS232 port 1 Enable / Disable.
9) RS232 port 2 Enable / Disable.
10) BCSR Enable / Disable.
11) Hard Reset Configuration Source - BCSR0 / Flash[C] Memory
12) PCMCIA control which include:
    - Channel Enable / Disable.

A. I.e., card insertion when the ADS is powered
B. In fact only the upper 16 bits - D(0:15) are used, but the BCSR is mapped as a 32 bit wide register and should be accessed as such.
C. Provided that support is provided also within the MPC.

- PC Card VCC appliance.
- PC Card VPP appliance.

13) Ethernet Port Control.
14) Dram Type / Size and Delay Identification.
15) Flash Size / Delay Identification.
16) External (off-board) tools identification or S/W option selection switch - SW7status.
17) Daughter Board ID.
18) Board Revision code
19) Reset E1/T1 device
20) Reset ATM155.
21) Reset ATM25.
22) Reset Fast Ethernet PHY.

4•11•1 BCSR Disable Protection Logic

The BCSR itself may be disabled in favor of off-board logic. To avoid accidental disable of the BCSR, an event from which only power re-appliance recovers, protection logic is provided:

The BCSR_EN~ bit resides on BCSR1. This bit wakes-up active (low) during power-up and may not be changed[A] unless BCSR_EN_PROTECT~ bit in BCSR3 is written with '1' previously.

After the BCSR_EN_PROTECT~ is written with '1' to unprotect the BCSR_EN~ bit there is only one shot at disabling the BCSR, since, immediately after any write to BCSR1, BCSR_EN_PROTECT~ is re-activated and BCSR_EN~ is re-protected and the disabling procedure has to be repeated if desired.

4•11•2 BCSR0 - Hard Reset Configuration Register

BCSR0 is located at offset 0 on BCSR space. It may be read or written at any time[B]. BCSR0 gets its defaults upon MAIN[C] Power-On reset. During Hard-Reset data contained in BCSR0 is driven on the data bus to provide the Hard-Reset configuration for the MPC, this, if the Flash_Configuration_Enable~ bit in BCSR1 is not active. BCSR0 may be written at any time to change the Hard-Reset configuration of the MPC. The new values become valid when the next Hard-Reset is issued to the MPC regardless of the Hard-Reset source. The description of BCSR0 bits is shown in TABLE 4-10. "BCSR0 Description" on page

A. It may be written but will not be influenced.
B. Provided that BCSR is not disabled.
C. I.e., when VDDH to the MPC is powered.

59.

TABLE 4-10. BCSR0 Description

BIT	MNEMONIC	FUNCTION	PON DEF.	ATT
0	ERB	External Arbitration. When '0' during Hard-Reset, Arbitration is performed internally. When '1' during Hard-Reset, Arbitration is performed externally.	0	R,W
1	IP	Interrupt Prefix. When '0' during Hard-Reset, Interrupt prefix set to 0xFFF00000, if '1' Interrupt Prefix set to 0.	0	R,W
2	Reserved	Implemented[a]	0	R,W
3	BDIS	Boot Disable. When '0' during Hard-Reset, CS0~ region is enabled for boot. When '1', CS0~ region is disabled for boot.	0	R,W
4 - 5	BPS(0:1)	Boot Port Size. Determines the port size for CS0~ at boot. '00' - 32 bit, '01' - 8 bit, '10' - 16 bit, '11' - reserved.	'00'	R,W
6	Reserved	Implemented[a]	0	R,W
7 - 8	ISB(0:1)	Initial Space Base. Value during Hard-Reset determines the initial base address of the internal MPC memory map. When '00' - initial space at 0, when '01' - initial space at 0x00F00000, when '10' - initial space at 0xFF000000, when '11' - initial space at 0xFFF00000.	'10'	R,W
9 - 10	DBGC(0:1)	Debug Pins Configuration. Value during Hard-Reset determines the function of the PCMCIA channel II pins. When '00' - these pins function as PCMCIA channel II pins, when '01' - they serve as Watch-Points,'10' - Reserved, when '11' - they become show-cycle attribute pins, e.g., VFLS, VF...	'11'	R,W
11-12	DBPC(0:1)	Debug Port Pins Configuration. Value during Hard-Reset determines the location of the debug port pins. When '00' - debug port pins are on the JTAG port, when '01' - debug port non-existent, '10' - Reserved, when '11' debug port is on PCMCIA channel II pins.	'00'	R,W
13 - 14	EBDF(0:1)[b]	External Bus Division Factor. Value during Hard Reset determines the factor upon which the CLKOUT of the MPC external bus, is divided with respect to its internal MPC clock. When '00' - CLKOUT is GCLK2 divided by 1, when '01', CLKOUT is GCLK2 divided by 2.	'00'	R,W
15	Reserved	Implemented[a].	'0'	R,W
16 - 31	Reserved	Un-Implemented	-	-

a. May be read and written as any other fields and are presented at their associated data pins during Hard-Reset.
b. Applicable for MPC's revision A or above. Otherwise have no influence.

4•11•3 BCSR1 - Board Control Register 1

The BCSR1 serves as a control register on the ADS. It is accessed at offset 4 from BCSR base address. It may be read or written at any time[A]. BCSR1 gets its defaults upon Power-On reset. . BCSR1 fields are described in TABLE 4-11. "BCSR1 Description" on page 60.

A. Provided that BCSR is not disabled.

TABLE 4-11. BCSR1 Description[a]

BIT	MNEMONIC	Function	PON DEF	ATT.
0	FLASH_EN	***Flash Enable***. When this bit is active (**low**), the Flash memory module is enabled on the local memory map. When in-active, the Flash memory is removed from the local memory map and CS0~, to which the Flash memory is connected may be used off-board via the expansion connectors.	0	R,W
1	DRAM_EN	***Dram Enable***. When this bit is active (**low**), the DRAM module is enabled on the local memory map. When in-active, the DRAM is removed from the local memory map and CS2~ and CS3~[b], to which the DRAM is connected may be used off-board via the expansion connectors.	0	R,W
2	ETHEN	***Ethernet Port Enable***. When asserted (**low**) the EEST connected to SCC1 is enabled. When negated (high) that EEST is in standby mode, while all its system i/f signals are tri-stated.	1	R,W
3	IRDEN	***Infra-Red Port Enable***. When asserted (**low**), the Infra-Red transceiver, connected to SCC2 is enabled. When negated, the Infra-Red transceiver is put in shutdown mode. And SCC2 pins are available for off-board use via the expansion connectors.	1	R,W
4	FLASH_CFG_EN	***Flash Configuration Enable***. When this bit is asserted (**low**): (A) - the Hard-Reset configuration held in BCSR0 is NOT driven on the data bus during Hard-Reset and (B) - configuration data held at the 1'st word of the flash memory is driven to the data bus during Hard-Reset. [c]	1	R,W
5	CNT_REG_EN_PROTECT	***Control Register Enable Protect***. When this bit is active (**low**) the BCSR_EN bit in that register can not be written. When in-active, BCSR_EN may be written to remove the BCSR from the memory map. After any write to BCSR1 this bit becomes active again. This bit is a read-only[d] bit on that register.	0	R
6	BCSR_EN	***BCSR Enable***. When this bit is active (**low**) the Board Control & Status Register is enabled on the local memory map. When inactive, the BCSR may not be read or written and its associated CS1~ is available for off-board use via the expansion connectors. This bit may be written with '1' only if CNT_REG_EN_PROTECT bit is negated (1). When the BCSR is disabled it still continues to configure the board according the last data held in it even during Hard-Reset.	0	R,W
7	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 the relevant MPC communication port pins are available for off-board use via the expansion connectors.	1	R,W
8	PCCEN	***PC Card Enable***. When asserted (**low**), the on-board PCMCIA channel is enabled, i.e., address and strobe buffers are enabled to / from the card. When negated, all buffers to / from the PCMCIA channel are disabled allowing off-board use of its associated lines.	1	R,W

TABLE 4-11. BCSR1 Description[a]

BIT	MNEMONIC	Function	PON DEF	ATT.
9	PCCVCC0	***Pc Card VCC Select 0***. These signal in conjunction with PCCVCC1 determine the voltage applied to the PCMCIA card's VCC. Possible values are 0 / 3.3 / 5 V. For the encoding of these lines and their associated voltages see TABLE 4-12. "PCCVCC(0:1) Encoding" on page 62.	0	R,W
10 - 11	PCCVPP(0:1)	***PC Card VPP***. These signals determine the voltage applied to the PCMCIA card's VPP. Possible values are 0 / 5 / 12 V. For the encoding of these lines and their associated voltages see TABLE 4-13. "PCCVPP(0:1) Encoding" on page 62.	'11'	R,W
12	Dram_Half_Word	***Dram Half Word***. When this bit is active (**low**) and the steps listed in 4•6•1 "DRAM 16 Bit Operation" on page 42, are taken, the DRAM becomes 16 bit wide. When inactive the DRAM is 32 bit wide.	1	R,W
13	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 the relevant MPC communication port pins are available for off-board use via the expansion connectors.	1	R,W
14	SDRAMEN	***SDRAM Enable***. When this bit is active (**high**), the SDRAM module is enabled on the local memory map. When in-active, the DRAM is place in low-power mode, in fact removed from the local memory map, allowing its associated CS line, to be used off-board via the expansion connectors.	1	R,W
15	PCCVCC1	***Pc Card VCC Select 1***. These signal in conjunction with PCCVCC0 determine the voltage applied to the PCMCIA card's VCC. Possible values are 0 / 3.3 / 5 V. For the encoding of these lines and their associated voltages see TABLE 4-12. "PCCVCC(0:1) Encoding" on page 62.	0	R,W
16 - 31	Reserved	Un-implemented	-	-

a. Shaded areas are additions with respect to the MPC86xADS.
b. In case a Single Bank DRAM SIMM is used CS3~ is free as well.
c. Provided that this option is supported by the MPC by driving address lines low and asserting CS0~ during Hard-Reset.
d. It is written in BCSR3.

TABLE 4-12. PCCVCC(0:1) Encoding

PCCVCC(0:1)	*PC-Card VCC [V]*
00	0
01	5
10	3.3
11	0

TABLE 4-13. PCCVPP(0:1) Encoding

PCCVPP(0:1)	*PC Card VPP [V]*
00	0
01	5
10	12[a]
11	Hi-Z

a. Provided that a 12V power supply is applied.

4•11•4 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[A]. BCSR2's various fields are described in TABLE 4-14. "BCSR2 Description" on page 63.

A. Provided that BCSR is not disabled.

TABLE 4-14. BCSR2 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 3	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. There are additional 3 presence detect lines which encode the SIMM's delay but appear in BCSR3. For the encoding of FLASH_PD(4:1) see TABLE 4-15. "Flash Presence Detect (4:1) Encoding" on page 63.	-	R
4	Reserved	Un-implemented	-	-
5 - 8	DRAM_PD(4:1)	***Dram Presence Detect***. These lines are connected to the DRAM SIMM presence detect lines which encode the size and the delay of the DRAM SIMM mounted on the DRAM SIMM socket. For the encoding of DRAM_PD(4:1) see TABLE 4-16. "DRAM Presence Detect (2:1) Encoding" on page 64 and TABLE 4-17. "DRAM Presence Detect (4:3) Encoding" on page 64.	-	R
9 - 12	Un used		-	R
13 - 15	Un used		-	R
13 - 31	Reserved	Un-implemented.	-	-

TABLE 4-15. Flash Presence Detect (4:1) Encoding

FLASH_PD(4:1)	FLASH TYPE / SIZE
0 - 3	Reserved
4	SM732A2000 / SM73228 - 8 Mbyte SIMM, by SMART Modular Technologies.
5	SM732A1000A / SM73218 - 4 Mbyte SIMM, by SMART Modular Technologies.
6	MCM29080 - 8 MByte SIMM, by Motorola
7	MCM29040 - 4 MByte SIMM, by Motorola
8	MCM29020 - 2 MByte SIMM, by Motorola
9 - F	Reserved

TABLE 4-16. DRAM Presence Detect (2:1) Encoding

DRAM_PD(2:1)	*DRAM TYPE / SIZE*
00	MCM36100 by Motorola or MT8D132X by Micron- 4 MByte SIMM
01	MCM36800 by Motorola or MT16D832X by Micron - 32 MByte SIMM
10	MCM36400 by Motorola or MT8D432X by Micron - 16 MByte SIMM
11	MCM36200 by Motorola or MT16D832X by Micron - 8 MByte SIMM

TABLE 4-17. DRAM Presence Detect (4:3) Encoding

DRAM_PD(4:3)	*DRAM DELAY*
00	Reserved
01	Reserved
10	70 nsec
11	60 nsec

WARNING

Since EXTOLI(0:3) lines may be DRIVEN LOW ('0') by the dip-switch, OFF-BOARD tools should NEVER DRIVE them HIGH. Failure in doing so, might result in PERMANENT DAMAGE to the ADS and / or to OFF-BOARD logic.

4•11•5 BCSR3 - Board Control / Status Register 3

BCSR3 is an additional control / status register which may be accessed at offset 0xC from BCSR base address. BCSR3 gets its defaults during Power-On reset and may be read or written at any time. The description of BCSR3 is shown in TABLE 4-18. "BCSR3 Description" on page 65.

TABLE 4-18. BCSR3 Description

BIT	MNEMONIC	Function	PON DEF	ATT.
0 - 1	Reserved	Implemented	'00'	R
2 - 7	Reserved	.	-	R
5[a]	CNT_REG_EN_PROTECT	***Control Register Enable Protect***. When this bit is active (**low**) the BCSR_EN bit in that register can not be written. When in-active, BCSR_EN may be written to remove the BCSR from the memory map. After any write to BCSR1 this bit becomes active again. This bit is a write-only bit on that register.	0	W
6 - 7	Reserved	Un-Implemented	-	-
8	Reserved	.	-	R
9 - 11	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 - U43. There are additional 4 presence detect lines which encode the SIMM's Type but appear in BCSR2. For the encoding of FLASH_PD(7:5) see TABLE 4-19. "FLASH Presence Detect (7:5) Encoding" on page 65.	-	
12	Reserved		-	R
13	Reserved	Implemented	'0'	R
14 - 15	Reserved		-	R

a.

TABLE 4-19. FLASH Presence Detect (7:5) Encoding

FLASH_PD(7:5)	Flash Delay [nsec]
000	Not Supported
001	150
010	120
011	90
100 - 111	Not Supported

4•11•6 BCSR4 - Board Control / Status Register 4

The BCSR4 serves as a control register on the ADS. It is accessed at offset 10H from BCSR base address. It may be read or written at any time[A]. BCSR4 gets its defaults upon Power-On reset. BCSR4 fields are

described in TABLE 4-20. "BCSR4 Description" on page 66.
.

TABLE 4-20. BCSR4 Description

BIT	*MNEMONIC*	*Function*	*PON DEF*	*ATT.*
0	ETHLOOP	***Ethernet port Diagnostic Loop-Back***. When active (**high**), the MC68160 EEST is configured into diagnostic Loop-Back mode, where the transmit output is internally fed back into the receive section.	0	R,W
1	TFPLDL~	***Twisted Pair Full-Duplex***. When active (**low**), the MC68160 EEST is put into full-duplex mode, where, simultaneous receive and transmit are enabled.	1	R,W
2	TPSQEL~	***Twisted Pair Signal Quality Error Test Enable***. When active (**low**), a simulated collision state is generated within the EEST, so the collision detection circuitry within the EEST may be tested.	1	R,W
3	SIGNAL_LAMP	Signal Lamp. When this signal is active (low), a dedicated LED illuminates. When in-active, this led is darkened. This led is used for S/W signalling to user.	1	R,W
4	un used		1	R,W
5	un used		1	R,W
6	un used		1	R,W
7	un used.		0	R,W
8	un used		1	R,W
9	un used		1	R,W
10	un used		1	R,W
11	un used		1	R,W
12	un used		1	R,W
13 - 31	Reserved	Un-implemented	-	-

4•11•7 BCSR5 - Board Control / Status Register 5

The BCSR5 serves as a control register on the ADS. It is accessed at Address 0x2000300 on CS5. It may be written at any time. BCSR5 gets its defaults upon Power-On reset. BCSR5 fields are described in TABLE 4-21. "BCSR5 Description" on page 67

A. Provided that BCSR is not disabled.

TABLE 4-21. BCSR5 Description

BIT	MNEMONIC	Function	PON DEF	ATT
7	RESET_MII	Reset Fast Ethernet phy. Driving 1 to this pin reset the PHY. On power up reset this pin drive the value of MPC poreset signal. Write 0x01 in order to reset the phy.	0	Write only
6	MII_RX_EN	MII RX-signals three-state. driving this pin to 1 disable the output of the fast ethernet phy RX signals. Write 0x02 in order to disable the RX-Signals.	0	Write only
5	RESET_ATM25	Reset ATM25 phy. Driving 1 to this pin reset the PHY. On power up reset this pin drive the value of MPC poreset signal. Write 0x04 in order to reset the phy.	0	Write only
4	RESET_ATM155	Reset ATM155 phy. Driving 1 to this pin reset the PHY. On power up reset this pin drive the value of MPC poreset signal. Write 0x08 in order to reset the phy.	0	Write only
3	RESET_Framer	Reset E1/T1 Framer. Driving 1 to this pin reset the FRAMER. On power up reset this pin drive the value of MPC poreset signal. Write 0x10 in order to reset the framer.	0	Write only
2	un used		0	Write only
1	un used		0	Write only
0	un used			Write only

4•12 Debug Port Controller

The debug port of the MPC86xADS is implemented on-board, connected to the MPC via the JTAG[A] port. Since the location[B] of the debug port is determined via the Hard-Reset configuration, It is important that the relevant configuration bits (see 4•1•5 "Reset Configuration" on page 38) are not changed, if working with the local debug port is desired.

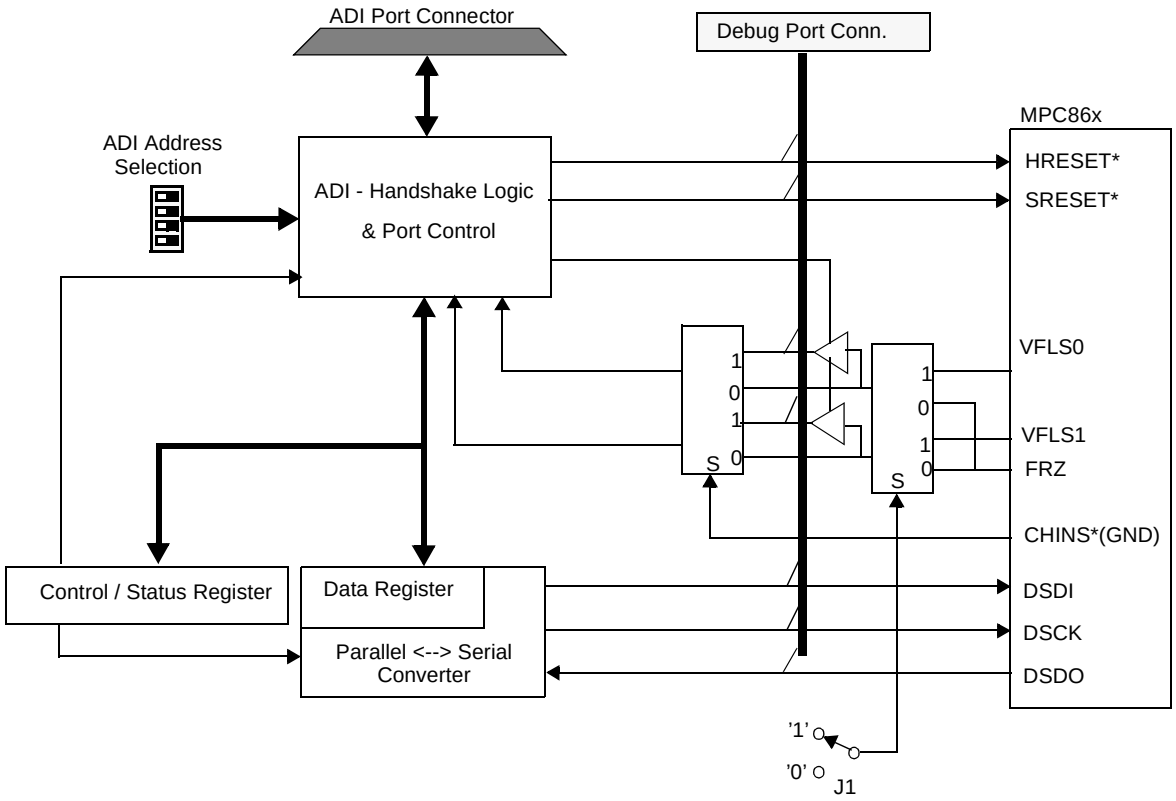
The debug port controller is interfaced to host computer via Motorola's ADI port, which is an 8-bit wide parallel port. Since the debug port is serial, conversion is done by hardware between the parallel and serial protocols.

The MPC's debug port is configured at SOFT-Reset to "Asynchronous Clock Mode" i.e., the debug port generates the debug clock - DSCK, which is asynchronous with the MPC system clock.

The debug port controller block diagram is shown in FIGURE 4-8 "Debug Port Controller Block Diagram" on page 68.

A. The debug port location is determined by the HARD - Reset configuration.
B. In terms of MPC pins.

FIGURE 4-8 Debug Port Controller Block Diagram



To allow for an external debug port controller to be incorporated with the ADS and to allow target system debug by the ADS, a standard 10 pin, debug port connector is provided and the local debug port controller may be disabled[A] by removing the ADI bundle from the its connector.

When the ADI's 37 lead cable is disconnected from either the ADI connector or from the ADS's 37 pin connector, the debug port controller is disabled allowing either the connection of an external debug port controller, or independent s/w run, i.e., the MPC boots from the flash memory to run user's application without debug port controller intervention. This feature becomes especially handy regarding demo's.

In this state, VFLS(0:1) or FRZ[B] signals are routed to the debug port connector, so that, the external debug port controller has run mode status information.

The ADI I/F supports upto 8 boards connected on the same bundle. Address selection is done by SW1 / 1,2,3. See 2•3•2 "ADI Port Address Selection" on page 9.

The debug port I/F has two registers: a control / status register and a data register. The control / status register hold I/F related control / status functions, while the data register serves as the parallel side of the Transmit / Receive shift register.

The control / status register is accessed when D_C~ bit is low while the data register is accessed when D_C~ is driven high by the host via the ADI port.

4•12•1 MPC86xADS As Debug Port Controller For Target System

The ADS may be used as a debug port controller for a target system, provided that the target system has

A. I.e., debug port controller outputs are tri-stated, allowing debug port to be driven by an external debug-tool.
B. Depended on H/W settings.

a 10 pin header connector matching the one on the ADS.

In this mode of operation, the on-board debug port controller, is connected to the target system's debug-port connector (see 4•12•1•1 "Debug Port Connection - Target System Requirements" below). Since DSDO signal is driven by the MPC, it is a must, to remove the local MPC from its socket, to avoid contention over this line.

When either the local MPC is removed from its socket or the daughter board is removed from the ADS, all ADS's modules are inaccessible, except for the debug-port controller. All module-enable indications are darkened, regardless of their associated enable bits in the BCSR. Pull-up resistors are connected to Chip-Select lines, so they do not float when the MPC is removed from its socket, avoiding possible contention over data-bus lines.

4•12•1•1 Debug Port Connection - Target System Requirements

In order for a target system may be connected to the ADS, as a debug port controller, few measures need to be taken on the target system:

1) A 10-pin header connector should be made available, with electrical connections matching FIGURE 4-9 "Standard Debug Port Connector" on page 71.
2) Pull-down resistors, of app. 1KΩ should be connected over DSDI[A] and DSCK[A] signals. These resistors are to provide normal[B] operation, when a debug-port controller, is not connected to the target system
3) The debug-port should be enabled and routed to the desired pins. See the DBGC and DBPC fields within the HARD-RESET configuration word.

4•12•2 Debug Port Control / Status Register

The control / status register is an 8 bit register (bit 7 stands for MSB). For the description of the ADI control status register see TABLE 4-22. "Debug Port Control / Status Register" on page 70.

A. Remember that the location of DSDI and DSCK is determined by the HARD-Reset configuration.
B. Normal - i.e., boot via CS0~.

TABLE 4-22. Debug Port Control / Status Register

BIT	*MNEMONIC*	*Function*	*I/F Reset DEF*	*ATT.*
7	MpcRst	***Mpc Reset***. When this status only bit indicates when active (**high**) that either a SOFT or a HARD reset is driven by the MPC.	-	R
6	TxError	***Transmit Error***. When this status only bit is active (**high**) it indicates that the last transmission towards the MPC, was cut by an internal MPC866 reset source. This bit is updated for each byte sent.	-	R
5	InDebug	***In Debug Mode***. When this status only bit is active (**high**) it indicates that the MPC is in debug mode[a].	-	R
4 - 3	DebugClockFreq	***Debug Clock Frequency Select***. This field controls a frequency divider which divides DSCK. For the division factors and associated DSCK frequencies see TABLE 4-23. "DSCK Frequency Select" below.	'00'	R/W
2	StatusRequest	***Status Request***. When the host writes this bit active (**low**), the I/F will issue a status read request to the host by asserting ADS_REQ line to the host. When the host writes the control register with this bit negated, no status read request is issued. Upon I/F reset this bit wakes-up active.	0	R/W
1	DiagLoopBack	***Diagnostic Loopback Mode***. When this control bit is active (**low**) the I/F is placed in Diagnostic Loopback Mode. I.e., DSDI is connected internally to DSDO, DSDI is tri-stated, and each data byte sent to the I/F data register, is sampled back into the receive shift register. This mode allows for complete ADI I/F test, upto transmit and receive shift registers. Upon I/F reset this bit wakes-up active.	0	R/W
0	DebugEntry	***Debug Mode Entry***. When this bit is active (**low**), the MPC will enter debug mode instantly after SOFT reset. When inactive, the MPC will start executing normally and will enter debug mode only after exception. Upon I/F reset this bit wakes-up active.	0	R/W

a. Provided that the PCMCIA channel II pins are configured as debug pins - i.e, VFLS(0:1) signals are available. If not, the debug port can not be operated correctly.

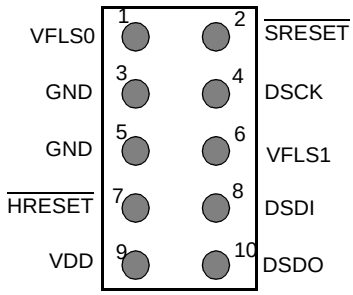
TABLE 4-23. DSCK Frequency Select

DebugClockFreq	*DSCK Frequency [MHz]*
00	10
01	5
10	2.5

TABLE 4-23. DSCK Frequency Select

DebugClockFreq	*DSCK Frequency [MHz]*
11	1.25

FIGURE 4-9 Standard Debug Port Connector



4•12•3 Standard MPCXXX Debug Port Connector Pin Description

The pins on the standard debug port connector are the maximal group needed to support debug port controllers for both the MPC5XX and MPC866 families. Some of the pins are redundant for the MPC866 family but are necessary for the MPC5XX family.

4•12•3•1 VFLS(0:1)

These pins indicate to the debug port controller whether or not the MPC is in debug mode. When both VFLS(0:1) are at '1', the MPC is in debug mode. These lines may serve alternate functions with the MPC, in which case FRZ needs to selected, on either the ADS or target system[A].

4•12•3•2 HRESET*

This is the Hard-Reset bidirectional signal of the MPC. When this signal is asserted (low) the MPC enters hard reset sequence which include hard reset configuration. This signal is made redundant with the MPC866 debug port controller since there is a hard-reset command integrated within the debug port protocol. However, the local debug port controller uses this signal for compatibility with MPC5XX existing boards and s/w.

4•12•3•3 SRESET*

This is the Soft-Reset bidirectional signal of the MPC866. On the MPC5XX it is an output. The debug port configuration is sampled and determined on the rising-edge[B] of SRESET* (for both processor families). On the MPC866 it is a bidirectional signal which may be driven externally to generate soft reset sequence. This signal is in fact redundant regarding the MPC866 debug port controller since there is a soft-reset command integrated within the debug port protocol. However, the local debug port controller uses this signal for compatibility with MPC5XX existing boards and s/w.

4•12•3•4 DSDI - Debug-port Serial Data In

Via the DSDI signal, the debug port controller sends its data to the MPC. The DSDI serves also a role

A. If a target system needs to use VFLS(0:1) alternate function, then, FRZ line should be connected to both VFLS(0:1) pins on the debug port connector.

B. In fact that configuration is divided into 2 parts, the first is sampled 3 system clock cycles prior to the rising edge of SRESET* and the second is sampled 8 clocks after that edge.

during soft-reset configuration. (See 4•1•5•3 "Soft Reset Configuration" on page 39).

4•12•3•5 DSCK - Debug-port Serial Clock

During asynchronous clock mode, the serial data is clocked into the MPC according[A] to the DSCK clock. The DSCK serves also a role during soft-reset configuration. (See 4•1•5•3 "Soft Reset Configuration" on page 39).

4•12•3•6 DSDO - Debug-port Serial Data Out

DSDO is clocked out by the MPC according to the debug port clock, in parallel[B] with the DSDI being clocked in. The DSDO serves also as "READY" signal for the debug port controller to indicate that the debug port is ready to receive controller's command (or data).

4•13 Power

There are 3 power buses with the MPC866s:

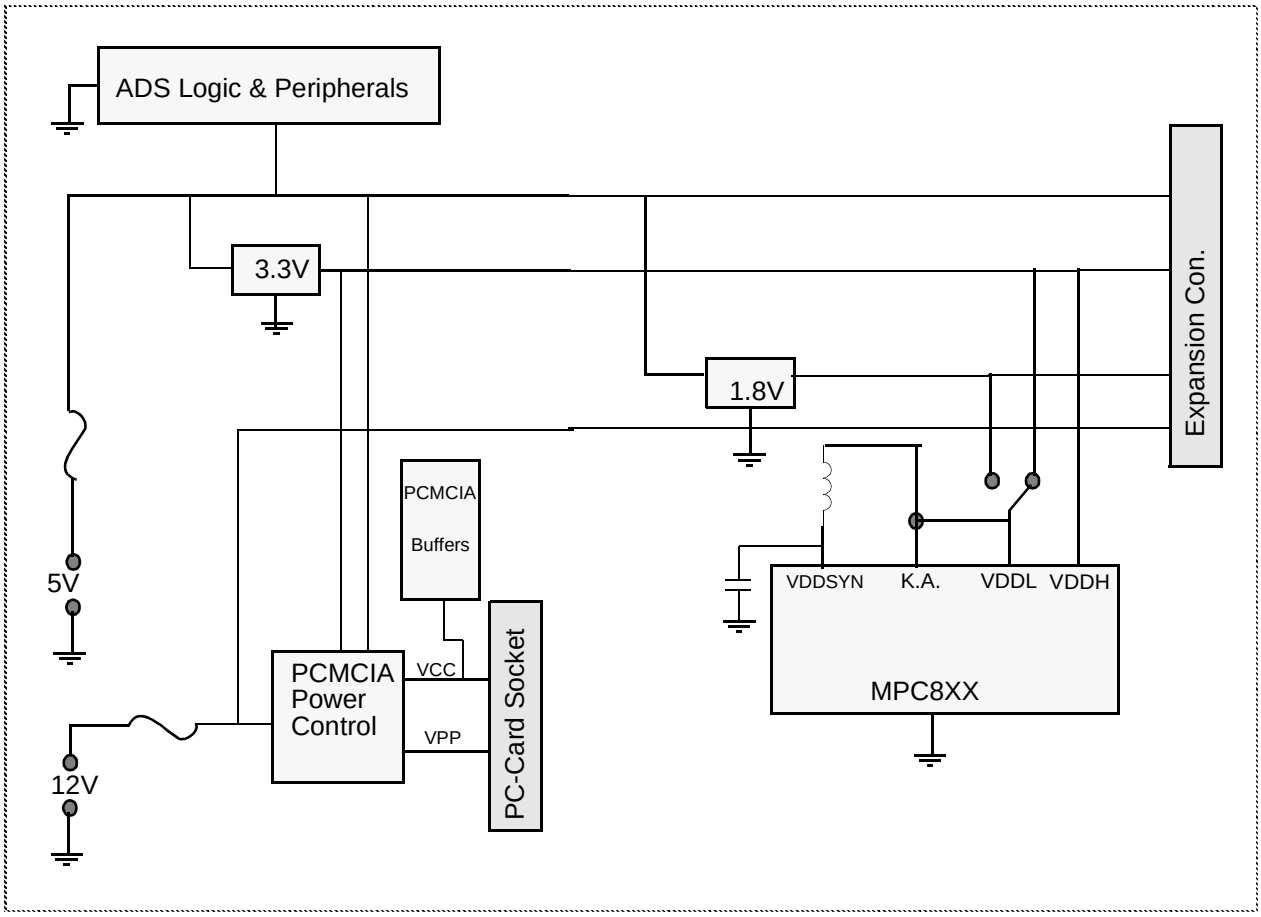
1) I/O
2) Internal Logic
3) PLL

and there are 4 power buses on the MPC86xADS:

1) 5V bus
2) 3.3V bus.
3) 12V bus.

A. I.e., DSDI must meat setup / hold time to / from rising edge of the DSCK.
B. I.e., full-duplex communication.

FIGURE 4-10 MPC86xADS 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 is shown in TABLE 4-24. "Off-board Application Maximum Current Consumption" below.

TABLE 4-24. Off-board Application Maximum Current Consumption

Power BUS	*Current*
5V	1.5A
3.3V	1.5A
12V	100 mA.

To protect on board devices against supply spikes, decoupling capacitors (typically 0.1μF) are provided between the devices' power leads and GND, located as close as possible to the power leads.

4•13•1 5V Bus

Some of the ADS peripherals reside on the 5V bus. Since the MPC is not 5V friendly, a 3.3V to 5V buffers added between the MPC and the 5v devices, so it may operate with 5V levels on its lines with no damage. The 5V bus is connected to an external power connector via a fuse (5A).

To protect against reverse-voltage or over-voltage being applied to the 5V inputs a set of high-current diodes and zener diode is connected between the 5V bus GND. When either over or reverse voltage is applied to the ADS, the protection logic will blow the fuse, while limiting the momentary effects on board.

4•13•2 3.3V Bus

The MPC itself as well as the SDRAM, the address and data buffers are powered by the 3.3[A] bus, which is produced from the 5V bus using a special low-voltage drop, linear voltage regulator made by Micrel, the MIC29500-3.3BT which is capable of driving upto 5A, facilitating operation of external logic as well.

4•13•3 12V Bus

The sole purpose of the 12V bus is to supply VPP (programming voltage) for the PCMCIA card and for the Flash SIMM[B]. It is connected to a dedicated input connector via a fuse (1A) and protected from over / reverse voltage application.

If the 12V supply is not required for either the PC-Card and for the flash SIMM, the 12V input to the ADS may be omitted.

A. At full speed. When lower performance is needed the internal logic may be powered from the 2V bus.
B. If necessary.

5 - Support Information

In this chapter all information needed for support, maintenance and connectivity to the MPC86xADS is provided.

5•1 Interconnect Signals

The MPC86xADS interconnects with external devices via the following set of connectors:

1) P1 - 10BaseT Ethernet port
2) P2A - RS232 port 1
3) P2B - RS232 port 2
4) P3 - T1/E1 RJ45 Connector.
5) P4 - ATM25 RJ45 Connector.
6) P5 - ATM155 multy mode optical connector.
7) P6 - ADI Port connector.
8) P7 - Serial Ports' Expansion connector.
9) P8, P11, P14, P16, P17 and P19 Mictor, Logic Analyser connectors.
10) P9 - External Debug port controller input / output
11) P10 - 100BaseT Ethernet port. RJ45.
12) P12 - 12V Power In.
13) P13 - 3 Pin 5V Power In
14) P13A - Power Jack connector 2.1mm. Will be connected to the supplied power supply.
15) P15 - ISP connector for Mach programing.
16) P18 - BNC connector - not populated.
17) P20 - JTAG connector for Altera programing.
18) P21 - MPC862/6 Address Data & control signals, Expansion connector.
19) P22 - PCMCIA port
20) IR1 - Infra - Red interface.

5•1•1 P1 - 10BaseT Ethernet Port Connector

The Ethernet connector on the MPC86xADS - P1, is a Twisted-Pair (10-Base-T) compatible connector. Use is done with 90^{o}, 8-pin, RJ45 connector, signals of which are described in TABLE 5-1 "P1 - Ethernet Port Interconnect Signals" below.

TABLE 5-1 P1 - Ethernet Port Interconnect Signals

Pin No.	*Signal Name*	*Description*
1	TPTX	Twisted-Pair Transmit Data positive output from the MPC86xADS.
2	TPTX~	Twisted-Pair Transmit Data negative output from the MPC86xADS.
3	TPRX	Twisted-Pair Receive Data positive input to the MPC86xADS.
4	-	Not connected

Support Information

TABLE 5-1 P1 - Ethernet Port Interconnect Signals

Pin No.	*Signal Name*	*Description*
5	-	Not connected
6	TPRX~	Twisted-Pair Receive Data negative input to the MPC86xADS.
7	-	Not connected
8	-	Not connected

5•1•2 PA2, PB2 - RS232 Ports' Connectors

The RS232 ports' connectors - PA2 and PB2 are 9 pin, 90º, female D-Type Stacked connectors, signals of which are presented in TABLE 5-2. "PA2, PB2 Interconnect Signals" below

TABLE 5-2. PA2, PB2 Interconnect Signals

Pin No.	*Signal Name*	*Description*
1	CD	Carrier Detect **output** from the MPC86xADS.
2	TX	Transmit Data **output** from the MPC86xADS.
3	RX	Receive Data **input** to the MPC86xADS.
4	DTR	Data Terminal Ready **input** to the MPC86xADS.
5	GND	Ground signal of the MPC86xADS.
6	DSR	Data Set Ready **output** from the MPC86xADS.
7	RTS (N.C.)	Request To Send. This line is not connected in the MPC86xADS.
8	CTS	Clear To Send **output** from the MPC86xADS.
9	-	Not connected

5•1•3 P3 - T1/E1 RJ45 Connector.

The T1/E1 connector on the MPC86xADS - P3, is a Twisted-Pair 8 pin connector. signals of which are described in TABLE 5-3. "P3 - T1/E1 RJ45 Connector." below.

TABLE 5-3. *P3 - T1/E1 RJ45 Connector*.

Pin No.	*Signal Name*	*Description*
1	TPRX	Twisted-Pair Receive Data positive input to the MPC86xADS.
2	TPRX~	Twisted-Pair Receive Data negative input to the MPC86xADS.
3	GND	Connected to GND
4	TPTX	Twisted-Pair Transmit Data positive output from the MPC86xADS.
5	TPTX~	Twisted-Pair Transmit Data negative output from the MPC86xADS.

TABLE 5-3. *P3 - T1/E1 RJ45 Connector*.

Pin No.	*Signal Name*	*Description*
6	GND	Connected to GND
7	-	Not connected
8	-	Not connected

5•1•4 P4 - ATM25 RJ45 Connector.

The ATM25 connector on the MPC86xADS - P4, is a Twisted-Pair 8 pin connector. signals of which are described in TABLE 5-4. "P4 - ATM25 RJ45 Connector." below.

TABLE 5-4. P4 - ATM25 RJ45 Connector.

Pin No.	*Signal Name*	*Description*
1	TPTX	Twisted-Pair Transmit Data positive input to the MPC86xADS.
2	TPTX~	Twisted-Pair Transmit Data negative input to the MPC86xADS.
3	Not Connected	Not Connected
4	Not Connected	Not connected
5	Not Connected	Not connected
6	Not connected	Not connected
7	TPRX	Twisted-Pair Receive Data positive output from the MPC86xADS.
8	TPRX~	Twisted-Pair Receive Data positive output from the MPC86xADS.

5•1•5 P5 - ATM155 multy mode optical connector.

This connector is optical connector with RX and TX SC type also for receive and transmit.

5•1•6 P6 ADI - Port Connector

The ADI port connector - P6, is a 37-pin, Male, 90^{o}, D-Type connector, signals of which are described in TABLE 5-5. "P1 - ADI Port Interconnect Signals" below:

TABLE 5-5. P1 - ADI Port Interconnect Signals

Pin No.	*Signal Name*	*Description*
1		Not connected with this application
2	D_C~	Data / Control selection. When '1', the debug port controller's data register is accessed, when '0' the debug port controller's control register is accessed.
3	HST_ACK	Host Acknowledge input signal from the host.
4	ADS_SRESET	When asserted ('1') and the ADS is selected by the host, generates Soft Reset to the MPC.
5	ADS_HRESET	When asserted ('1') and the ADS is selected by the host, generates Hard Reset to the MPC.
6	ADS_SEL2	ADI I/F address line 2 (MSB).

Support Information

TABLE 5-5. P1 - ADI Port Interconnect Signals

Pin No.	*Signal Name*	*Description*
7	ADS_SEL1	ADI I/F address line 1.
8	ADS_SEL0	ADI I/F address line 0 (LSB).
9	HOST_REQ	HOST Request input signal from the host
10	ADS_REQ	ADS Request output signal from the MPC86xADS to the host
11	ADS_ACK	ADS Acknowledge output signal from the MPC86xADS to the host
12		Not connected with this application
13		Not connected with this application
14		Not connected with this application
15		Not connected with this application
16	PD1	Bit 1 of the ADI port data bus
17	PD3	Bit 3 of the ADI port data bus
18	PD5	Bit 5 of the ADI port data bus
19	PD7	Bit 7 of the ADI port data bus
20 - 25	GND	Ground.
26		Not connected with this application
27 - 29	HOST_VCC	HOST VCC input from the host. Used to qualify ADS selection by the host. When host is off, the debug port controller is disabled.
30	HOST_ENABLE~	HOST Enable input signal from the host. (Active low). Indicates that the host computer is connected to ADS. Used, in conjunction with HOST_VCC and ADS_SEL(2:0) to qualify ADS selection by the host.
31 - 33	GND	Ground.
34	PD0	Bit 0 of the ADI port data bus
35	PD2	Bit 2 of the ADI port data bus
36	PD4	Bit 4 of the ADI port data bus
37	PD6	Bit 6 of the ADI port data bus

5•1•7 MPC86XADS's P7 - Serial Ports' Expansion Connector

P8 is a 96 pin, 900, DIN 41612 connector, which allows for convenient expansion of the MPC's serial ports. Note:

Support Information

The contents of TABLE 5-6. "MPC86XADS's P7 - Interconnect Signals" below**, might conflict with MPC8XXFADS's schematic page 14. This since, that the schematic page is named in MPC821/860 terms. In such case, this table overrides!**

TABLE 5-6. MPC86XADS's P7 - Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
A1	ETHRX	I/O	10Base-T Ethernet port receive data.
A2	ETHTX	I/O	10Base-T Ethernet Port transmit data.
A3	IRDRXD	I/O	IrDA port receive data.
A4	IRDTXD	I/O	IrDA port transmit data.
A5	MPD11	I/O	This pin connected through Bus Switch to MPC PORT D11, and controlled by SW3(2,3,4).
A6	MPD10	I/O	This pin connected through Bus Switch to MPC PORT D10, and controlled by SW3(2,3,4).
A7	MPD9	I/O	This pin connected through Bus Switch to MPC PORT D9, and controlled by SW3(2,3,4).
A8	MPD8	I/O	This pin connected through Bus Switch to MPC PORT D8, and controlled by SW3(2,3,4).
A9	ETHTCK	I/O	10Base-T Ethernet port transmit clock.
A10	ETHRCK	I/O	10Base-T Ethernet port receive clock.
A11	PA5	I/O	MPC86x PA5 Pin.
A12	PA4	I/O	MPC86X PA[4] PIN.
A13	PA3	I/O	MPC86x PA3 Pin.
A14	PA2	I/O	MPC86x PA2 Pin.
A15	PA1	I/O	MPC86x PA1 Pin.
A16	PA0	I/O	MPC86x PA0 Pin.
A17	VCC	-	
A18	PA11	I/O	MPC86x PA11 Pin.
A19	PA10	I/O	MPC86x PA10 Pin.
A20	PA9	I/O	MPC86x PA9 Pin.
A21	PA8	I/O	MPC86x PA8 Pin.
A22	GND	-	
A23	GND	-	
A24	IRQ7~	I, L	This pin connected through Bus Switch to MPC IRQ7~, and controlled by SW3(2,3,4).

Support Information

TABLE 5-6. MPC86XADS's P7 - Interconnect Signals

Pin No.	Signal Name	Attribute	Description
A25	FRZ	I/O	MPC86x FRZ pin.
A26	ETHEN~	O	Ethernet Enable Pin from mach.
A27	IRQ3~	I, L	MPC86x RQ3~ Pin.
A28	IRQ2~	I, L	RSV/IRQ2~. See PM2(26).
A29	IRQ1~	I, L	MPC86x RQ1~ Pin.
A30	NMI~	I, L	MPC86x NMI~ Pin.
A31	RS_EN1~	O,L	RS232 _1Enable from mach.
A32	GND	-	
B1	PB31	I/O	MPC86x PB31 Pin.
B2	PB30	I/O	MPC86x PB30 Pin.
B3	PB29	I/O	MPC86x PB29 Pin.
B4	PB28	I/O	MPC86x PB28 Pin.
B5	PB27	I/O	MPC86x PB27 Pin.
B6	PB26	I/O	MPC86x PB26 Pin.
B7	RSTXD1	I/O	MPC86x PB25 / This pin use on the board as RS232_1 TXD signal.
B8	RSRXD1	I/O	MPC86x PB24 / This pin use on the board as RS232_1 RXD signal.
B9	RSDTR1~	I/O	MPC86x PB23 / This pin use on the board as RS232_1 DTR signal.
B10	RSDTR2~	I/O	MPC86x PB22 / This pin use on the board as RS232_2 DTR signal.
B11	TXD2.	I/O	MPC86x PB21 / This pin use on the board as RS232_2 TXD signal.
B12	RSRXD2	I/O	MPC86x PB20 / This pin use on the board as RS232_2 RXD signal.
B13	E_TENA	I/O	MPC86x PB19 / This pin use on the board as Ethernet 10Base-T TENA signal.
B14	PB18	I/O	MPC86x PB18 PIN.
B15	PB17	I/O	MPC86x PB17 PIN.
B16	PB16	I/O	MPC86x PB16 PIN
B17	PB15	I/O	MPC86x PB15 PIN
B18	PB14	I/O	MPC86x PB14 PIN
B19	GND	-	

Support Information

TABLE 5-6. MPC86XADS's P7 - Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
B20	RXCLAV	I/O	MPC86x PC15 / Use as RXCLAV for ATM or INPACK for PCMCIA, The selection is done by SW3(2,3,4).
B21	PC14	I/O	MPC86x PC14 PIN.
B22	PC13	I/O	MPC86x PC13 PIN.
B23	PC12	I/O	MPC86x PC12 PIN.
B24	E_CLSN	I/O	10Base-T Ethernet port collision signal.
B25	E_RENA	I/O	10Base-T Ethernet port RENA signal.
B26	PC9	I/O	PMC86x PC9 PIN.
B27	PC8	I/O	PMC86x PC8 PIN.
B28	PC7	I/O	PMC86x PC7 PIN.
B29	PC6	I/O	PMC86x PC6 PIN.
B30	PC5	I/O	PMC86x PC5 PIN.
B31	PC4	I/O	PMC86x PC4 PIN.
B32	GND	-	
C1	VCC		
C2			
C3			
C4			
C5			
C6	RS_EN2~	O, L	RS232 _2Enable from mach.
C7	GND		
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15	MPD15	I/O	This pin connected through Bus Switch to MPC PORT D15, and controlled by SW3(2,3,4).
C16	MPD14	I/O	This pin connected through Bus Switch to MPC PORT D14, and controlled by SW3(2,3,4).

Support Information

TABLE 5-6. MPC86XADS's P7 - Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
C17	MPD13	I/O	This pin connected through Bus Switch to MPC PORT D13, and controlled by SW3(2,3,4).
C18	MPD12	I/O	This pin connected through Bus Switch to MPC PORT D12, and controlled by SW3(2,3,4).
C19	MPD7	I/O	This pin connected through Bus Switch to MPC PORT D7, and controlled by SW3(2,3,4).
C20	MPD6	I/O	This pin connected through Bus Switch to MPC PORT D6, and controlled by SW3(2,3,4).
C21	VCC	-	
C22	HRESET~	I/O, L	MPC86x Hreset PIN
C23	SRESET~	I/O, L	MPC86x Sreset PIN
C24	N.C.	-	Not Connected
C25	VCC	-	
C26	MPD3	I/O	This pin connected through Bus Switch to MPC PORT D3, and controlled by SW3(2,3,4).
C27	VPPIN	I/O	+12V input for PCMCIA flash programming. Parallel to P12 of the MPC86xADS.
C28			
C29	GND	-	
C30	MPD4	I/O	This pin connected through Bus Switch to MPC PORT D4, and controlled by SW3(2,3,4).
C31	GND	-	
C32	MPD5	I/O	This pin connected through Bus Switch to MPC PORT D5, and controlled by SW3(2,3,4).

5•1•8 P8, P11, P14, P16, P17 and P19 Mictor, Logic Analyser connectors.

These connectors are 38 pin, receptacle MICTOR connectors made by AMP. Each connector connects to a dedicated adaptor for HP 16500 series of logic analyzer, which interconnects to two 16 bit pods.

TABLE 5-7. P17 Interconnect Signals

Pin#	MPC862/6 Signal Name	Pin#	MPC862/6 Signal Name
1	N.C.	2	N.C.
3	GND	4	N.C.
5	EXTCLK	6	ALEA
7	N.C.	8	IRQ2b
9	AT1	10	IRQ3b
11	TEXP	12	DP0
13	RESETA	14	DP1
15	PDEAb	16	DP2
17	MODCK1	18	DP3
19	MODCK2	20	FRZ
21	PORSTb	22	SPKROUT
23	RSTCNFb	24	IPA0
25	HRESETb	24	IPA1
27	SRESETb	28	IPA2
29	WAITBb	30	IPA3
31	WAITAb	32	IPA4
33	GPL4Ab	34	IPA5
35	GPL4Bb	36	IPA6
37	CE1Ab	38	IPA7

TABLE 5-8. P19 - Interconnect Signals

Pin#	MPC862/6 Signal Name	Pin#	MPC862/6 Signal Name
1	N.C.	2	N.C.
3	GND	4	N.C.

Support Information

TABLE 5-8. P19 - Interconnect Signals

Pin#	MPC862/6 Signal Name	Pin#	MPC862/6 Signal Name
5	TAb	6	TEAb
7	VFLS0	8	FCSb
9	VFLS1	10	BCSRCSb
11	AT2	12	DRMCS1b
13	VF2	14	DRMCS2b
15	VF0	16	SDRMCSb
17	VF1	18	CS5b
19	AT0	20	CS6b
21	AT3	22	CS7b
23	BCD1~	24	BS0Ab
25	BGb	26	BS1Ab
27	BRb	28	BS2Ab
29	BIb	30	BS3Ab
31	GPL5Bb	32	WE0b
33	BURSTb	34	WE1b
35	RWb	36	WE2b
37	TSb	38	WE3b

TABLE 5-9. P11 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
1	N.C.	2	N.C.
3	GND	4	N.C.
5	DRMWb	6	CE2Ab
7	GPL5Ab	8	PA0
9	GPL3b	10	PA1
11	GPL2b	12	PA2
13	EDOOEb	14	PA3
15	PC4	16	PA4
17	PC5	18	PA5

TABLE 5-9. P11 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
19	PC6	20	ETHRCK
21	PC7	22	ETHTCK
23	PC8	24	PA8
25	PC9	26	PA9
27	ERENA	28	PA10
29	ECLSN	30	PA11
31	PC12	32	IRDTXD
33	PC13.	34	IRDRXD
35	PC14	36	ETHTX
37	BINPAKb	38	ETHRX

TABLE 5-10. P14 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
1	GND	2	N.C.
3	N.C.	4	N.C.
5	SYSCLK	6	MODCK1
7	A0	8	A16
9	A1	10	A17
11	A2	12	A18
13	A3	14	A19
15	A4	16	A20
17	A5	18	A21
19	A6	20	A22
21	A7	22	A23
23	A8	24	A24
25	A9	26	A25
27	A10	28	A26

TABLE 5-10. P14 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
29	A11	30	A27
31	A12	32	A28
33	A13	34	A29
35	A14	36	A30
37	A15	38	A31

TABLE 5-11. P8 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
1	N.C.	2	N.C.
3	GND	4	N.C.
5	IRQ1b	6	IRQ7b
7	NMIb	8	PB16
9	PD3	10	PB17
11	PD4	12	PB18
13	PD5	14	ETENA
15	PD6	16	RSRXD2
17	PD7	18	RSTXD2
19	PD8	20	RSDTR2b
21	PD9	22	RSDTR1b
23	PD10	24	RSRXD1
25	PD11	26	RSTXD1
27	PD12	28	PB26
29	PD13	30	PB27
31	PD14	32	PB28
33	PD15	34	PB29
35	PB14	36	PB30
37	PB15	38	PB31

Support Information

TABLE 5-12. P16 - Interconnect Signals

Pin #	MPC862/6 Signal Name	Pin #	MPC862/6 Signal Name
1	N.C.	2	N.C.
3	GND	4	N.C.
5	REGAb	6	TSIZ1
7	D0	8	D16
9	D1	10	D17
11	D2	12	D18
13	D3	14	D19
15	D4	16	D20
17	D5	18	D21
19	D6	20	D22
21	D7	22	D23
23	D8	24	D24
25	D9	26	D25
27	D10	28	D26
29	D11	30	D27
31	D12	32	D28
33	D13	34	D29
35	D14	36	D30
37	D15	38	D31

5•1•9 P9 - External Debug Port Controller Input Interconnect.

The debug port connector - P9, is a 10 pin, Male, header connector, signals of which are described in TABLE 5-16. "P13 - Interconnect Signals" below

TABLE 5-13. P9 - Interconnect Signals

Pin No.	Signal Name	Attribute	Description
1	VFLS0	O	Visible history FLushes Status 0. Indicates in conjunction with VFLS1, the number of instructions flushed from the core's history buffer. Indicates also whether the MPC is in debug mode. If not using the debug port, may be configured for alternate function. When the ADS is disconnected from the ADI bundle, it may be FRZ signal, depended on J1's position.
2	SRESET~	I/O	Soft Reset line of the MPC. Active-low, Open-Drain.
3	GND		Ground.
4	DSCK	I/O	Debug Serial Clock. Over the rising edge of which serial date is sampled by the MPC from DSDI signal. Over the falling edge of which DSDI is driven towards the MPC and DSDO is driven by the MPC. Configured on the MPC's JTAG port. When the debug-port controller is on the local MPC or when the ADS is a debug-port controller for a target system - OUTPUT, when the FADI bundle is disconnected from the ADS - INPUT.
5	GND		Ground
6	VFLS1	O	See VFLS0. When the ADS is disconnected from the ADI bundle, it may be FRZ signal, depended on J1's position.
7	HRESET~	I/O	Hard Reset line of the MPC. Active-low, Open-Drain
8	DSDI	I/O	Debug Serial Data In of the debug port. Configured on the MPC's JTAG port. When the debug-port controller is on the local MPC or when the ADS is a debug-port controller for a target system - OUTPUT, when the ADI bundle is disconnected from the ADS - INPUT.
9	V3.3	O	3.3V Power indication. This line is merely for indication. No significant power may be drawn from this line.
10	DSDO	I/O	Debug Serial Data Output from the MPC. Configured on the MPC's JTAG port. When the debug-port controller is on the local MPC or when the ADI bundle is disconnected from the ADS - OUTPUT, when the ADS is a debug-port controller for a target system - INPUT.

5•1•10 P10 - 100BaseT Ethernet Port Connector

The Ethernet connector on the MPC86xADS - P1, is a Twisted-Pair (100-Base-T) compatible connector. Use is done with 90^{o}, 8-pin, RJ45 connector, signals of which are described in TABLE 5-14. "P10 - Ethernet Port Interconnect Signals" below.

TABLE 5-14. P10 - Ethernet Port Interconnect Signals

Pin No.	Signal Name	Description
1	TPTX	Twisted-Pair Transmit Data positive output from the MPC86xADS.
2	TPTX~	Twisted-Pair Transmit Data negative output from the MPC86xADS.
3	TPRX	Twisted-Pair Receive Data positive input to the MPC86xADS.
4	-	Not connected
5	-	Not connected
6	TPRX~	Twisted-Pair Receive Data negative input to the MPC86xADS.
7	-	Not connected
8	-	Not connected

5•1•11 P12 - 12V Power Connector

The 12V power connector - P12, is a two-lead, 2 part, terminal block connector, identical in type to the 5V connector. P12 supplies, when necessary, programming voltage to the Flash SIMM and / or to the PCMCIA

TABLE 5-15. P12 - Interconnect Signals

Pin Number	Signal Name	Description
1	12V	12V input from external power supply.
2	GND	GND line from external power supply.

5•1•12 P13 - 5V Power Connector

The 5V power connector - P13, is a 3-lead, two-part terminal block. The male part is soldered to the pcb, while the receptacle is connected to the power supply. That way fast connection / disconnection of power is facilitated and physical efforts are avoided on the solders, which therefore maintain solid connection over time.

TABLE 5-16. P13 - Interconnect Signals

Pin Number	Signal Name	Description
1	5V	5V input from external power supply.
2	GND	GND line from external power supply.
3	GND	GND line from external power supply.

5•1•13 P13A - Power Jack connector 2.1mm.

P13A is a Plug Jack connector 2.1mm that is connected to the power supply supplied with the board. Inorder to operate the board the user shpould plug in the connector of the power supply to this connector.

5•1•14 P15 - Mach's In System Programming (ISP)

This is a 10 pin generic 0.100" pitch header connector, providing In System Programming capability for Vantis made programmable logic on board. The pinout of P11 is shown in TABLE 5-17. ". P15 - ISP Connector - Interconnect Signals" below:

TABLE 5-17. . P15 - ISP Connector - Interconnect Signals

Pin No.	Pin 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.
6	VCC	O	5V/3.3 power supply bus. the power can be change by RJ4 by connecting a resistor from 1, 2 to power 5V, or 2, 3 to power 3.3V
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.

5•1•15 P18 - BNC connector - not populated.

Is a BNC connector in order to drive clock in to MPC EXTCLK pin this BNC is not populated. the user can use this connector but he should first connect RJ3 pins 2, 3. It is factory connect pins 1, 2.

5•1•16 P20 - JTAG connector for Altera programing.

This is a 10 pin generic 0.100" pitch header connector, providing In System Programming capability for altera made programmable logic on board. The pinout of P11 is shown in TABLE 5-18. "P20 - JTAG connector for Altera programing." below:

TABLE 5-18. *P20 - JTAG connector for Altera programing.*

Pin No.	Pin Name	Attribute	Description
1	TCK	I	Test port Clock. This clock shifts in / out data to / from the programmable logic JTAG chain.
2	GND	O	Digital GND. Main GND plane.

TABLE 5-18. *P20 - JTAG connector for Altera programing.*

Pin No.	Pin Name	Attribute	Description
3	TDO	O	Transmit Data Output. This the prog. logic's JTAG serial data output driven by Falling edge of TCK.
4	VCC	O	5V power supply bus.
5	TMS	I	Test Mode Select. This signal qualified with TCK, changes the state of the prog. logic JTAG machine.
6	N.C.	-	Not Connected.
7	N.C.	-	Not Connected.
8	N.C.	-	Not Connected.
9	TDI	I	Transmit Data In. This is the prog. logic's JTAG serial data input.
10	GND	O	Digital GND. Main GND plane.

5•1•17 Expansion Connector ADD, Data control & PCMCIA Port.

P21 is a 96 pin, 900, DIN 41612 connector, which allows for convenient expansion of the MPC's.This connector include the necessary signals inorder to connect external peripherals with address data and control signals, also in this connector there are the PCMCIA signals that can be use for ATM Split mode. in order to use ATM Split from the Expansion connectors the user should control it through SW3(2,3,4) most of the pins are on P7 and the PCMCIA pins are on P21. This connector is also include MII signals for using in external board the signals are CRS, MDIO, TXEN and COL. For using external Fast ethernet the user should connect P21(A25) to GND.

TABLE 5-19. MPC86XADS's P21 - Interconnect Signals

Pin No.	Signal Name	Attribute	Description
A1	EXATMRSC	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXSOC.
A2	EXATMRD0	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD0. For external MII this pin is MIIRXD3.
A3	EXATMRD1	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD1. For external MII this pin is MIIRXD2.
A4	EXATMRD2	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD2. For external MII this pin is MIIRXD1.
A5	EXATMRD3	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD3. For external MII this pin is MIIRXD0.
A6	EXATMRD4	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD4. For external MII this pin is MIIRXCLK.
A7	EXATMRD5	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD5. For external MII this pin is MIIRXERR
A8	EXATMRD6	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD6. For external MII this pin is MIITXERR.
A9	EXATMRD7	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXD7. For external MII this pin is MIIRXDV.
A10	GND		GND
A11	EXATMRCK	O	In External ATM Split connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the RXCLK. For external MII this pin is MIITXD0.
A12	GND		GND
A13	N.C		
A14	BWE0b	O	MPC86x WE0 Pin. For external peripheral.
A15	BDRMWb	O	MPC86x GPL0 signal use for DRAM write signal.
A16	BEDOOEb	O	MPC86x GPL1 Pin use for DRAM oe~ signal.
A17	BGPL2b	O	MPC86x GPL2 Pin.
A18	BGPL3b	O	MPC86x GPL3 Pin.

Support Information

TABLE 5-19. MPC86XADS's P21 - Interconnect Signals

Pin No.	Signal Name	Attribute	Description
A19	GPL4Ab	O	MPC86x GPL4A Pin.
A20	GPL4Bb	O	MPC86x GPL4B Pin.
A21	GPL5Ab	O	MPC86x GPL5A Pin.
A22	GPL5Bb	O	MPC86x GPL5B Pin
A23	GND		
A24	BSYSCLK3	O	MPC86x CLKOUT Pin, drive by zero delay buffer.
A25	GND		GND.
A26	BS0Ab	O	MPC86x BS0b Pin. (buffered)
A27	GND		GND.
A28	BRW2b	O	MPC86x RWb Pin. (buffered)
A29	BTSb	O	MPC86x TSb Pin (buffered).
A30	TAb	O	MPC86x TA Pin.
A31	CS7b	I	MPC86x CS7 Pin.
A32	CS6b	I	MPC86x CS6 Pin
B1	3.3V	I/O	Power 3.3V.
B2	3.3V	I/O	Power 3.3V.
B3	3.3V	I/O	Power 3.3V.
B4	3.3V	I/O	Power 3.3V.
B5	3.3V	I/O	Power 3.3V.
B6	3.3V	I/O	Power 3.3V.
B7		I/O	
B8	GND	I/O	
B9	EXPMITXD3	I/O	MPC86x CE2A / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIITXD3.
B10	EXPMITXD2	I/O	MPC86x CE1A / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIITXD2.
B11	EXPMITXD1	I/O	MPC86x ALE / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIITXD1.
B12	EXPCOL	I/O	MPC86x MIICOL / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIICOL, also P21(A25) must be connected to GND.

Support Information

TABLE 5-19. MPC86XADS's P21 - Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
B13	EXPTXEN	I/O	MPC86x MIITXEN / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIITXEN, also P21(A25) must be connected to GND.
B14	EXPMDIO	I/O	MPC86x MIIDIO / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIIMDIO, also P21(A25) must be connected to GND.
B15	EXPCRS	I/O	MPC86x MIICRS / In External MII connected to the Expansion connectors, P7 & P21 when putting SW3(2 =,ON 3 = ON, 4 = ON). This pin is the MIICRS, also P21(A25) must be connected to GND.
B16	GND	I/O	GND
B17	N.C		
B18	GND	I/O	GND
B19	N.C	-	
B20	GND	I/O	GND
B21	N.C		
B22	GND	I/O	GND
B23	N.C		
B24	GND	I/O	GND
B25	N.C		
B26	GND	I/O	GND
B27	N.C		
B28	GND	I/O	GND
B29	N.C		
B30	GND	I/O	GND
B31	N.C		
B32	GND	-	GND

Support Information

TABLE 5-19. MPC86XADS's P21 - Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
C1	BD0	I/O	MPC86x Buford D0 - D7 to connect and control the external peripheral connected to the expansion connectors.
C2	BD1	I/O	
C3	BD2	I/O	
C4	BD3	I/O	
C5	BD4	I/O	
C6	BD5	I/O	
C7	BD6	I/O	
C8	BD7	I/O	
C9	BA16		MPC86x A16 - A31(buffered BA16 - BA31) to connect and control the external peripheral connected to the expansion connectors.
C10	BA17		
C11	BA18		
C12	BA19		
C13	BA20		
C14	BA21		
C15	BA22		
C16	BA23		
C17	BA24		
C18	BA25		
C19	BA26		
C20	BA27		
C21	BA28		
C22	BA29		
C23	BA30		
C24	BA31		
C25	MIISELb		
C26	N.C		
C27	N.C	I/O	
C28	N.C		
C29	N.C	-	
C30	N.C	I/O	
C31	N.C	-	

TABLE 5-19. MPC86XADS's P21 - Interconnect Signals

Pin No.	Signal Name	Attribute	Description
C32	N.C		

5•1•18 PCMCIA Port Connector

The PCMCIA port connector - P22, is a 68 - pin, Male, 90^0, PC Card type, signals of which are presented in TABLE 5-20. "P22 - PCMCIA Connector Interconnect Signals" below

TABLE 5-20. P22 - PCMCIA Connector Interconnect Signals

Pin No.	Signal Name	Attribute	Description
1	GND		Ground.
2	PCCD3	I/O	PCMCIA Data line 3.
3	PCCD4	I/O	PCMCIA Data line 4.
4	PCCD5	I/O	PCMCIA Data line 5.
5	PCCD6	I/O	PCMCIA Data line 6.
6	PCCD7	I/O	PCMCIA Data line 7.
7	BCE1A~	O	PCMCIA Chip Enable 1. Active-low. Enables EVEN numbered address bytes.
8	PCCA10	O	PCMCIA Address line 10.
9	OE~	O	PCMCIA Output Enable signal. Active-low. Enables data outputs from PC-Card during memory read cycles.
10	PCCA11	O	PCMCIA Address line 11.
11	PCCA9	O	PCMCIA Address line 9.
12	PCCA8	O	PCMCIA Address line 8.
13	PCCA13	O	PCMCIA Address line 13.
14	PCCA14	O	PCMCIA Address line 14.
15	WE~/PGM~	O	PCMCIA Memory Write Strobe. Active-low. Strobes data to PC-Card during memory write cycles.
16	RDY	I	+Ready/-Busy signal from PC-Card. Allows PC-Card to stall access from the host, in case a previous access's processing is not completed.
17	PCCVCC	O	5V VCC for the PC-Card. Switched by the MPC86xADS, via BCSR1.
18	PCCVPP	O	12V/5V VPP for the PC-Card programming. 12V available only if12V is applied to P8. Controlled by the MPC86xADS, via BCSR1.
19	PCCA16	O	PCMCIA Address line 16.

Support Information

TABLE 5-20. P22 - PCMCIA Connector Interconnect Signals

Pin No.	Signal Name	Attribute	Description
20	PCCA15	O	PCMCIA Address line 15.
21	PCCA12	O	PCMCIA Address line 12.
22	PCCA7	O	PCMCIA Address line 7.
23	PCCA6	O	PCMCIA Address line 6.
24	PCCA5	O	PCMCIA Address line 5.
25	PCCA4	O	PCMCIA Address line 4.
26	PCCA3	O	PCMCIA Address line 3.
27	PCCA2	O	PCMCIA Address line 2.
28	PCCA1	O	PCMCIA Address line 1.
29	PCCA0	O	PCMCIA Address line 0.
30	PCCD0	I/O	PCMCIA Data line 0.
31	PCCD1	I/O	PCMCIA Data line 1.
32	PCCD2	I/O	PCMCIA Data line 2.
33	WP	I	Write Protect indication from the PC-Card.
34	GND		Ground
35	GND		Ground
36	CD1~	I	Card Detect 1~. Active-low. Indicates in conjunction with CD2~ that a PC-Card is placed correctly in socket.
37	PCCD11	I/O	PCMCIA Data line 11.
38	PCCD12	I/O	PCMCIA Data line 12.
39	PCCD13	I/O	PCMCIA Data line 13.
40	PCCD14	I/O	PCMCIA Data line 14.
41	PCCD15	I/O	PCMCIA Data line 15.
42	BCE2A~	O	PCMCIA Chip Enable 2. Active-low. Enables ODD numbered address bytes.
43	VS1	I	Voltage Sense 1 from PC-Card. Indicates in conjunction with VS2 the operation voltage for the PC-Card.
44	IORD~	O	I/O Read. Active-low. Drives data bus during I/O-Cards' read cycles.
45	IOWR~	O	I/O Write. Active-low. Strobes data to the PC-Card during I/O-Card write cycles.
46	PCCA17	O	PCMCIA Address line 17.
47	PCCA18	O	PCMCIA Address line 18.
48	PCCA19	O	PCMCIA Address line 19.

Support Information

TABLE 5-20. P22 - PCMCIA Connector Interconnect Signals

Pin No.	*Signal Name*	*Attribute*	*Description*
49	PCCA20	O	PCMCIA Address line 20.
50	PCCA21	O	PCMCIA Address line 21.
51	PCCVCC	O	5V VCC for the PC-Card. Switched by the MPC86xADS, via BCSR1.
52	PCCVPP	O	12V/5V VPP for the PC-Card programming. 12V available only if12V is applied to P8. Controlled by the MPC86xADS, via BCSR1.
53	PCCA22	O	PCMCIA Address line 22.
54	PCCA23	O	PCMCIA Address line 23.
55	PCCA24	O	PCMCIA Address line 24.
56	PCCA25	O	PCMCIA Address line 25.
57	VS2	I	Voltage Sense 2 from PC-Card. Indicates in conjunction with VS1 the operation voltage for the PC-Card.
58	RESET	O	Reset signal for PC-Card.
59	WAITA~	I	Cycle Wait from PC-Card. Active-low.
60	INPACK~	I	Input Port Acknowledge. Active-low. Indicates that the Pc-Card can respond to I/O access for a certain address.
61	PCREG~	O	Attribute Memory or I/O Space - Select. Active-low. Used to select either attribute (card-configuration) memory or I/O space.
62	BVD2	I	Battery Voltage Detect 2. Used in conjunction with BVD1 to indicate the condition of the PC-Card's battery.
63	BVD1	I	Battery Voltage Detect 1. Used in conjunction with BVD2 to indicate the condition of the PC-Card's battery.
64	PCCD8	I/O	PCMCIA Data line 8.
65	PCCD9	I/O	PCMCIA Data line 9.
66	PCCD10	I/O	PCMCIA Data line 10.
67	CD2~	I	Card Detect 2~. Active-low. Indicates in conjunction with CD1~ that a PC-Card is placed correctly in socket.
68	GND		Ground.

5•2 MPC86xADS Part List

In this section the MPC86xADS's bill of material is listed according to their reference designation.

TABLE 5-21. MPC86xADS Part List

Reference Designation	*Part Description*	*Manufacturer*	*Part #*
C1,C23,C45,C46,C47,C48, C49,C50,C55,C56,C57,C265,	1000P(1nF) 50V 10% X7R 1206 SMD	AVX	AVX12065C102KA
C2,C3,C4,C5,C6,C7,C8,C9, C10,C13,C14,C16,C17,C26, C27,C28,C30,C31,C32,C33, C37,C38,C39,C40,C41,C42, C44,C51,C52,C58,C60,C62, C64,C65,C66,C67,C68,C69, C74,C77,C81,C83,C84,C85, C89,C91,C92,C94,C95,C96, C97,C101,C102,C103,C104, C106,C107,C110,C111,C112, C113,C115,C120,C121,C125, C133,C134,C137,C139,C141, C142, C143,C144,C145,C146,C148, C150,C152,C154,C155,C157, C160,C161,C162,C163,C165, C167,C168,C169,C170,C172, C174,C175,C177,C180,C181, C183,C185,C188,C190,C191, C193,C194,C195,C196,C200, C201,C202,C203,C204,C205, C208,C211,C212,C213,C214, C217,C218,C219,C220,C221, C223,C225,C226,C227,C228, C230,C232,C233,C235,C236, C237,C238,C239,C241,C245, C247,C253,C254,C256,C257, C258,C259,C261,C262,C263, C264,C266,C268,C270,C272, C273,C276,C277,C279,C280, C281,C282,C283,C285,C288, C290,C295,C296,C297,C298, C300,C301,C302,C306,C308, C312,C313,C314,C315,C316, C317,C318,C319,C320,C321, C322,C323,C324,C325,C326, C327,C329,C330	100NF(0.1uF)16V 10% X7R 0603	EPCOS	0603X7R104K016P07
C11,C12,C15,C24,C25,C29, C34,C70,C71,C72,C75,C78, C79,C80,C82,C87,C88,C90, C93,C98,C99,C100,C105, C108,C109,C114,C116,C117, C118,C119,C126,C135,C136, C140,C147,C149,C151,C153, C156,C171,C176,C178,C179, C182,C184,C186,C187,C192, C198,C199,C206,C207,C209, C210,C216,C222,C224,C229, C231,C242,C243,C248,C250, C274,C275,C284,C291,C292, C303,C305,C307,C309,C310, C328,C331	0.01uF 2KV X7R 1825 10% SMD	JOHANSON DIELECTRIC	202S49W103KV4E
C18,C158,C289,C304	1uF 25V 10% SMD A TANT	SPRAGUE	293D105X9025A2T
C19,C21,C234,C249,C251,C255,C260,C294,C299,C311	120PF 50V 5% SMD COG 1206	AVX	1206 5A 121 JTR
C20	68UF 20V 20% SIZE D or E CAP	SPRAGUE	293D686X9020E2T

Support Information

TABLE 5-21. MPC86xADS Part List

Reference Designation	Part Description	Manufacturer	Part #
C22,C59,C63,C73,C76,C122, C127,C128,C129,C130,C131, C132,C159,C189,C197,C267, C61,C164,C166,C173	10uF 10V 10% SIZE A TANT SMD CAP	SPRAGUE	293D106X9010A2T
C35,C278	100UF/10V TNT D SMT 10%	SIEMENS	B45196-H2107K
C36	10NF 50V 10% NPO 1210	VITRAMON	VJ1210A103KXAT
C43	4.7uF 10v/10% Size A TNT SMD T/R	AVX	TAJA475K10
C54,C53	470PF 50V 5% COG SMD 1206	AVX	12065A471JA
C86,C124	68PF 50V 5% COG SMD 1206	SIEMENS	B37871K5680J62
C123	39NF 5% 50V 1206 SMD	SIEMENS	B37872K5393J62
C138	3900PF 50V 5% COG 1210 SMD	SIEMENS	B37949K5392J62
C215	10nF-2KV 0.01uF 2KV X7R 1825 10% SMD	JOHANSON DIELECTRIC	202S49W103KV4E
C244,C240	10PF 50V 5% COG 1206	AVX	AV12065A100JATJ
C246	0.56uF 16V X7R 1206 10% SMD	AVX	1206YC564KAT2A
C252	5.6nF 5600pF (5.6n) X7R 0603 5% SMD	AVX	06035C562JAT2A
C269	100PF 50V 10% COG 1206-SMD	AVX	12065A101KAT00J
C271	1uF 25V 10% SMD A TANT	SPRAGUE	293D105X9025A2T
D2,D1	LL4148D	ON SEMICONDUCTOR	RLS4148
D3,D4,D5,D6,D11,D12,D13	LL4004G SMD	TSC	LL4004G
D7,D10	MBRD620CT	ON SEMICONDUCTOR	MBRD620CT
D8	1SMC12AT3 ZENER DIODE	ON SEMICONDUCTOR	1SMC12AT3
D9	DIODE 1SMC5.OAT3	ON SEMICONDUCTOR	1SMC5.OAT3
FR1,FR2,FR3,FR4,FR5,FR6,FR7,FR8,FR9 ,FR10,FR11, FR12,FR13,FR14,FR15,FR16, FR17,FR18,FR19	3 Pin Ferrite FERRITE NFM60R30T222T	MURATA	NFM60R30T222T
F1	POLYSWITCH SMD DEVICE	RAYCHEM	SMD150/33-2 (X153)
F2	SMD260 POLYSWITCH 5.2A	RAYCHEM	SMD260
IR1	TFDS6500 OBSOLETE	TELEFUNKEN	TFDS6500
LD1,LD3,LD6,LD11	LED_RED	KINGBRIGHT	KPT-3216ID
LD2,LD9,LD12,LD14,LD16, LD17,LD18,LD19,LD20,LD21, LD22	LED_YELLOW	KINGBRIGHT	KPT-3216YD
LD4,LD5,LD7,LD8,LD10,LD13,LD15,LD23	LED_GREEN	KINGBRIGHT	KPT-3216SGD
L1,L2,L3,L4,L5,L8,L9,L10,L11,L12	BLM18AG121SN1 CHIP FERRITE BID 120 OHM 0603	MURATA	BLM18AG121SN1
L6	PT12133 8,2MH INDUCTOR	BOURNS	PT12133
L7	ACM1110-102-2P COMMON MODE CHOCK COIL TO DC LIN	TDK	ACM1110-102-2P
L13	3.3uH 3.3uH CHIP INDUCTOR T/R	TDK	NL322522T-3R3J

Support Information

TABLE 5-21. MPC86xADS Part List

Reference Designation	Part Description	Manufacturer	Part #
P1,P3,P4,P10	8P RJ45 90'PC TEL JACK	ERIC-CAMBRIDGE	0170-88
P2	RS232-PORT2 9P DUAL F/90'DCON+TAIL	EDA	8LE009009D306H
P5	HFBR-5205 ATM MULTIM FIBR TRANSC	Agilent (HP)	HFBR-5205
P6	CON DB37. 37P D 90'+TAIL M 7.2/8.08mm	KCC	DNR 37P CB SG
P7,P21	CONN-96 96P DIN C F 90'PC+TAIL	ELCO	268477096002025
P8,P11,P14,P16,P17,P19	MICTOR38 38P LOG ANL MICTOR CON	AMP	2-767004-2
P9,P15,P20	10PIN TERM STRIP SHORT SMD 5X2	SAMTEC	TSM10501-SDV AP
P12	PWR2 2PIN PC STRGHT POW CON	WIELEND BAMBERG	8113S253303253
P13	PWR3 3PIN PC STRGHT POW CON	WIELEND BAMBERG	8113S253303353
P18	SMB Straight Jack for PCB	SUHNER	82SMB-50-0-1/111
P22	PCMCIA TOP 90'SMD CON	AMP	AMP 822021-5
Q1,Q2,Q3	MMDF3N03HD	ON SEMICONDUCTOR	MMDF3N03HD
RJ1,RJ2,RJ3	00-ohm on 1-2	AVX	
RJ4,R10,R11,R12,R14,R15, R16,R29,R72,R75,R78,R79, R96,R103,R104,R105,R118, R124,R127,R138,R139,R144, R145,R146,R156,R163,R165, R166,R167,R188,R190,R208,R212, R217,R218,R221,R223 R21,R22,R23,R55,R215,0 R226,R236 R24,R25,R26,R222,R224,	0 ohm 1% 0.1W 0603 SMD T/R	AVX	CJ10-000-T
RN1,RN2,RN3,RN4,RN7,RN8, RN20,RN23,RN24,RN25,RN26, RN27,RN28,RN29	10K 5% 8R 10P SMD CHIP RE NETW	ROHM	RS8A1002J-OR-MNR15EORPJ103
RN5,RN6,RN9,RN11,RN12, RN14,RN15,RN16,RN17,RN18, RN19	22ohm 5% 4R 8P SMD CHIP RE NETW	AVX	CRA3A4E220JT
RN10,RN13,RN21,RN22	0ohm 5% 4R 8P SMD CHIP RE NETW	DALE	CRA06S0803 000 RT
RP1,RP2	1K single-turn TRIMMING POT	BOURNS	3362P-1-102
R1,R27,R86,R135	2K 1% 1/4W METAL 1206	ROEDERSTEIN	D25002KFCS
R2	10 OHM 1% 1/4W SMD 1206	DRALORIK	D25 010RFCS
R3,R4,R9,R58,R59,R67,R69, R71,R74,R76,R80,R81,R84, R85,R87,R88,R89,R94,R134, R149,R189,R196,R206,R209, R210,R219,R220,R228,R229, R230,R231,R232,R233,R234, R235,R237	10K 1% 0.1W 0603 SMD T/R	ROEDERSTEIN	D11 010KFCS
R5,R66	47 OHM 1% 1/4W SMD1206	ROEDERSTEIN	D25047RFCS
R6,R61	820 OHM 5% 0805 SMD	BOURNS	CR0805-JW-821
R8,R7	110 OHM 5%1/4W MTL1206	AVX	CR32 111J-T

Support Information

TABLE 5-21. MPC86xADS Part List

Reference Designation	Part Description	Manufacturer	Part #
R13,R20,R35,R36,R90,R91, R97,R98,R99,R107,R113, R114,R115,R116,R121,R123, R125,R136,R137,R140,R148, R151,R152,R155,R164,R182, R187,R191,R193,R194,R207, R213,R214	1K 1% 0603 SMD T/R	DRALORIK	D11 001KFCS
R17,R18	49.9 OHM 1% SMD 1206	ROEDERSTEIN	D25 49R9FCS
R19	5,1K 5% 1/4W 1206 SMD	ROEDERSTEIN	D25 05K1JCS
R28,R37,R42,R52,R65,R172,R176,R177,R178,R179,R180, R181,R183,R185,R186	150 OHM 1% 1/4W 1206	ROEDERSTEIN	D25-150RFCS
R30,R38,R73,R122,R126,R31,R32130	100 OHM 1% 1/4W 1206	ROEDERSTEIN	D25-100RFCS
R33,R34,R49	82.5 ohm 1% 0603 SMD T/R	DRALORIK	D11 82R5FCS
R39,R44,R48,R56	5.6 Ohm 1% 1206 SMD	DRALORIK	D25-5R6-1%
R40,R41	33.2 1% 1/8W 1206 SMD	ROEDERSTEIN	D25 33R2FCS
R45,R43	2.7ohm 1% 1206 SMD	RCD	MC1206 2R74FT
R47,R46	432 OHM 1% 0603 SMD	DRALORIK	D11 432RFCS
R51,R50	63.4 ohm 1% 0603 SMD T/R	DRALORIK	D11 63R4FCS
R54,R53	39 OHM 1% 1/4W 1206	DRALORIK	D25039RFC5
R57	620 OHM 1% 0603 SMD	DALE	CRCW0603-6200F
R60,R62,R77,R82,R92,R93, R106,R108,R109,R110,R111, R133,R147,R169,R170,R171, R174	510 OHM 0603 1% SMD	DALE	CRCW0603-5100F
R64,R63	90.9 OHM 1% 0603 SMD	DALE	CRCW0603-90R9F
R68,R102	243 OHM 1% 1/4W 1206	ROEDERSTEIN	D25243RFCS
R70,R128,R129,R130,R131, R132,R192,R195,R197,R198, R201,R203	22.1ohm 1% 0.1W 0603 SMD T/R	ROEDERSTEIN	D11 22R1FCS
R83	294 OHM 1% 1/4W 1206	DRALORIK	D25294RFCS
R95	143 OHM 1% 1/8W 1206	ROEDERSTEIN	D25 143RFCS
R100	243 OHM 1% 1/4W 1206	ROEDERSTEIN	D25243RFCS
R101,R175	51.1ohm 1% 0.1W 0603 SMD T/	ROEDERSTEIN	D11 51R1FCS
R112,R119,R141,R142,R205	75 ohm 1% 0603 SMD T/R	DRALORIK	D11 075RFCS
R117	0 ohm or 510_1% ohm 0603		
R120	143 OHM 1% 1/8W 1206	ROEDERSTEIN	D25 143RFCS
R143,R150,R153,R154,R184	47.5K 1% 0603 SMD	DALE	CRCW0603-4752F
R157,R158,R159,R160	24.9 Ohm 1% 1206 SMD	DRALORIK	D25 24R9FCS
R161	20M 5% 1/4W 1206 SMD	ROEDERSTEIN	D25 020MJCS
R162	200K 1% 1206 SMD RES	ROEDERSTEIN	D25 200KFCS
R168,R173,R200,R204,R227	100 OHM 1% 1/4W 1206	ROEDERSTEIN	D25-100RFCS

Support Information

TABLE 5-21. MPC86xADS Part List

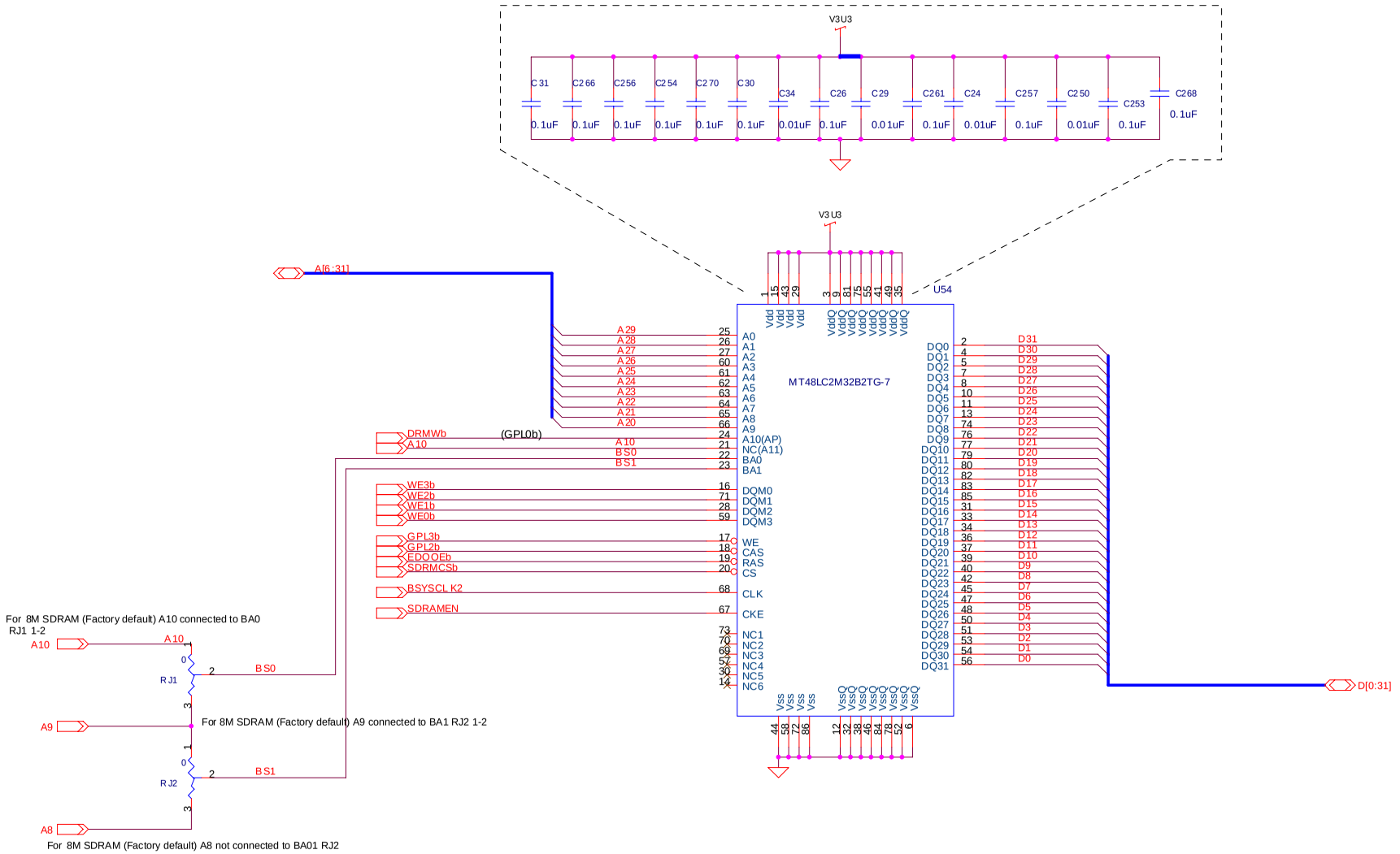
Reference Designation	Part Description	Manufacturer	Part #
R199,R202,R211,R216	124K 1% 1/8W 1206	ROEDERSTEIN	D25 124KFCS
SK1	SEP-1162 PIEZO SPEAKER	SOUNDTECH	SEP-1162
SW1,SW3,SW6,	SW DIP-4/SM 4POS 8PIN SEALD DIP SW. SMD	GRAYHIL	90HBW04PR
SW2	E101MD1ABE SINGLE TOGGLE SW. 0N-ON TO PCB	MORS	APM5236
SW5	ABORT-BLACK SPDT BLK PUSHBUTTON	C&K	KS12R22-CQE
SW6	SRESET-RED SPDT RED PUSHBUTTON	C&K	KS12R23-CQE
U1	PE-68026T SMT	PULSE ENG.	PE-68026 T/R
U2	T1144 T1/E1 Transformer 3.3V	PULSE ENG	T1144
U3	PE-67583	PULSE ENG	PE-67583D
U4	19.44M 3V SMD CK OSC 20PPM 7X5mm	AEL CRYSTALS	4303DS-019M440000S005
U5	MC68160	Motorola	MC68160
U6	1.544MHz. 3V TH 25PPM HS CLK OSC 2.048MHz. 3V TH 25PPM CLK OSC.	M-TRON	1.544MHz. M3H16FCD 2.048MHz M3H16FCD
U7	PEB2256 E1/T1/J1 FRAMER & LINE INTERFACE	INFINEON TECHNOLOGIES	PEB2256-V1.2
U8, U48	IDTQS32X384 CMOS 20 BIT BUS SWITCHES	IDT	IDTQS32X384Q1
U9	IDT77V107L25PF SINGLE ATM PHY WITH UTOPIA	IDT	IDT77V107L25PF
U10	uPD98404 ATM PHY	NEC - Electronics	UPD98404GJ-KEU
U11,U30	74LS244DW	ON-SEMICONDUCTOR	SN74LS244DW
U12,U18	LM317MDT Regulator.	Motorola	LM317MDT
U13	QS3244D HIGH SPEED CMOS 24:6 MUX/ DEMUX	IDT	QS3244
U14,U15,U19,U22,U25	QS33X253	IDT	QS33X253Q1
U16	M4A3-128/64-55VC MACH M4A3-128/64-55VC	Lattice	MACH M4A3-128/64-55VC
U17	20Mhz - 20.0M 3V SMD CLK OSC 20PPM 7X5mm	AEL CRYSTALS	4303DS-020M00000S005
U20	MPC862/6		
U21	74LCX08	ON SEMICONDUCTOR	74LCX08D
U23	80225 10/100BaseT PHY	LSI-LOGIC	NQ80225
U24	TG22-3506ND TRANSFORMETR TG22-3506	HALO	TG22-3506ND
U26	25.0M 3V SMD CLK OSC 20PPM 7X5mm	AEL CRYSTALS	4303DS-025M000000S005
U27	74AC14D	ON SEMICONDUCTOR	74AC14D
28,U40,U42,U49,U50,U51,U57	IDT74LVCH162244APF	IDT	IDT74LVCH162244APF TVSOP
U29	4.0MHz 3.3V TH CLK OSC 50PPM	AEL CRYSTALS	AEL1203BS-4.00MHZ

Support Information

TABLE 5-21. MPC86xADS Part List

Reference Designation	*Part Description*	*Manufacturer*	*Part #*
U31	DS1818-10 3.3V ECONORESET+PUSHBUTTON	DALLAS	DS1818R-10/T&R (SOT23)
U32	LTC1315 DUAL PCMCIA VPP SWITCH	LINEAR TECH	LTC1315CG
U33, U46	74LCX125D	ON SEMICONDUCTOR	74LCX125D
U34	MIC29500-3.3BT TO220	MICREL	MIC29500-3.3BT
U35	74AC05D	ON SEMICONDUCTOR	74AC05D
U36	M4A3-192/96-6VC	Lattice	M4A3-192/96-6VC
U37	PROGRAMABLE LOGIC DEVICE-ALTERA	Altera	EPM3128ATC144-10
U38	SIMM72 4MB EDO DRAM	TUSHIBA MICRON MICRON	THM3210CSG-60 MT2D132M-6X MT2D132M-60X
U39,U41	IDT74LVCH162373APF	IDT	IDT74LVCH162373APF TVSOP
U43	FLASH SIMM80 55132T9DX SIMM FLASH	WHITE MICROELECTRONICS SOUTHLAND MICRO SYSTEM	WPF512K32-70PSC5T WPF512K32-70PSC5T WPF512K32-70PSC5T 55132T9DX
U44,U45	MAX3241ECAI 28 SSOP	MAXIM	MAX3241ECAI/EEAI
U47	32.0M 3V SMD CLK OSC 20PPM 7X5mm	AEL CRYSTALS	4303DS-032M000000S005
U52	IDT74CBTLV3253PG LOW VOLTG 1-OF-4 DUAL MUX/DEMUX	IDT	IDT74CBTLV3253PG
U53	CY2309SC-1H 3.3V ZD BUFFER 16P SOIC	CYPRESS SEMICONDUCTOR IDT	CY2309SC-1H IDT 2309-1HDC
U54	MT48LC2M32B2TG-7 SDRAM 2MX32	MICRON	MT48LC2M32B2TG-8
U55,U56,U58	IDT74LVCHR162245APF	IDT	IDT74LVCHR162245APF
Y1	20MHz SMD CRYSTAL	MODERN ENTERPRISE CORPORATION	HC49/USM4 20.00MHZ
Y2	32768HZ, 32.768KHZ CRYSTAL SMD	RALTRON	RSM-200-32.768KHz.

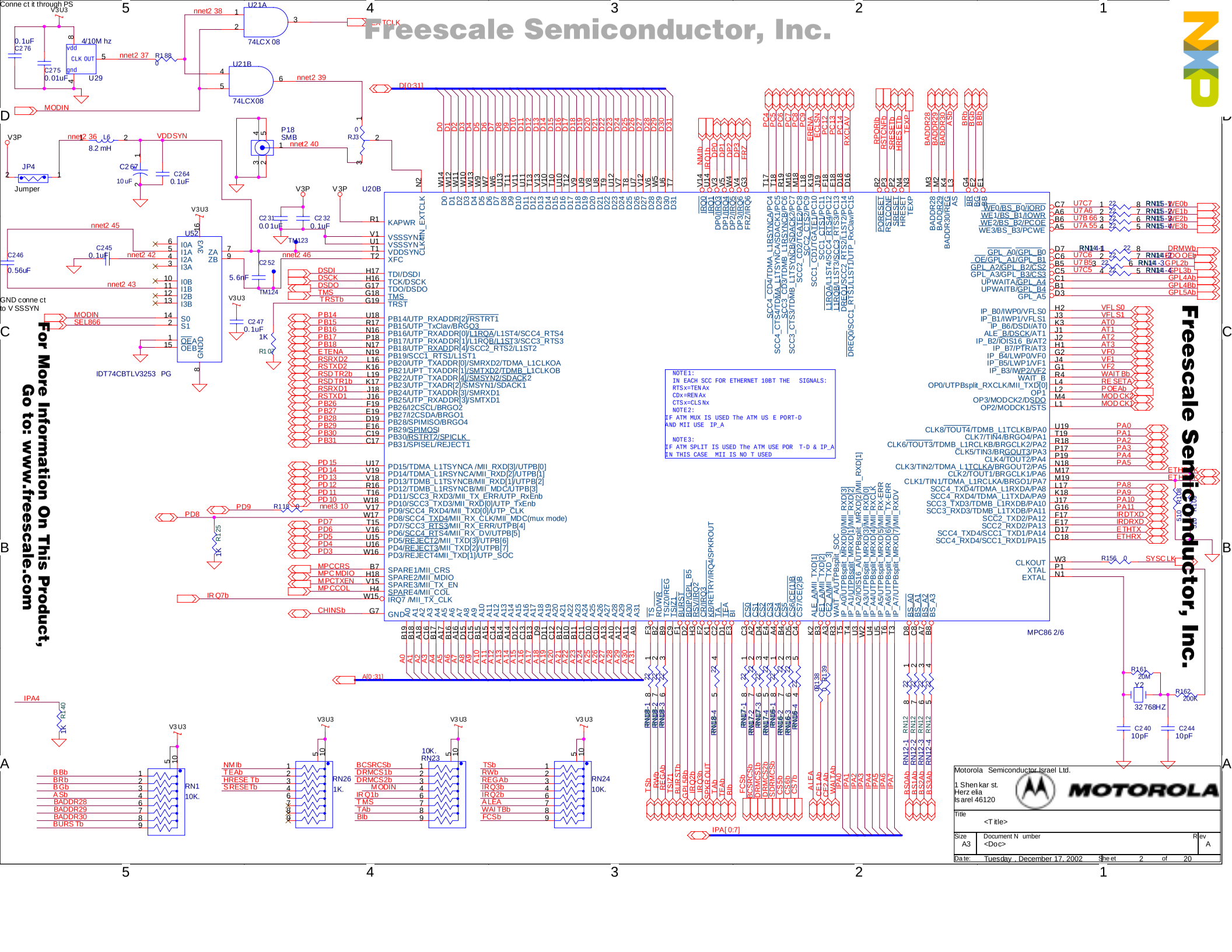
APPENDIX A - Schematics. 1-20.

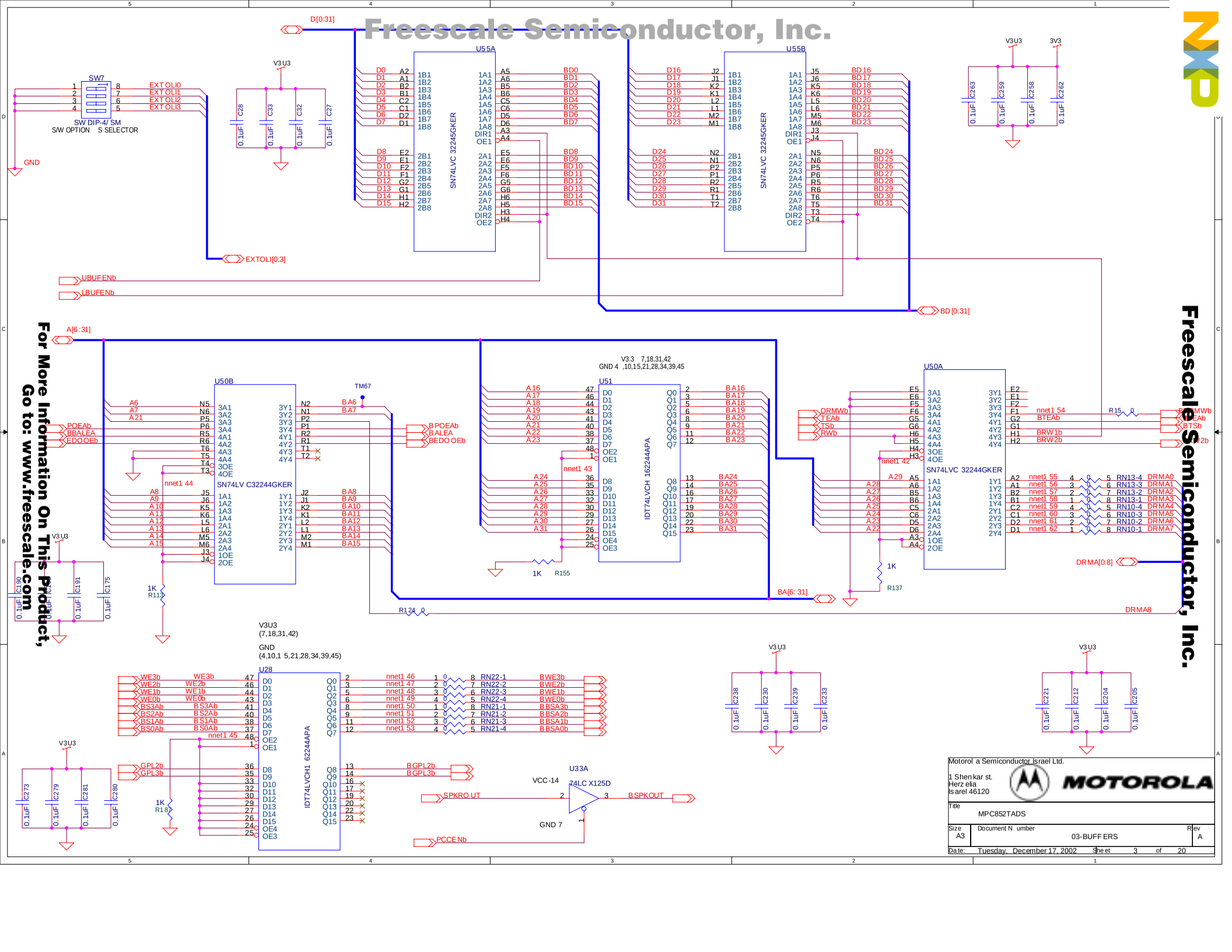


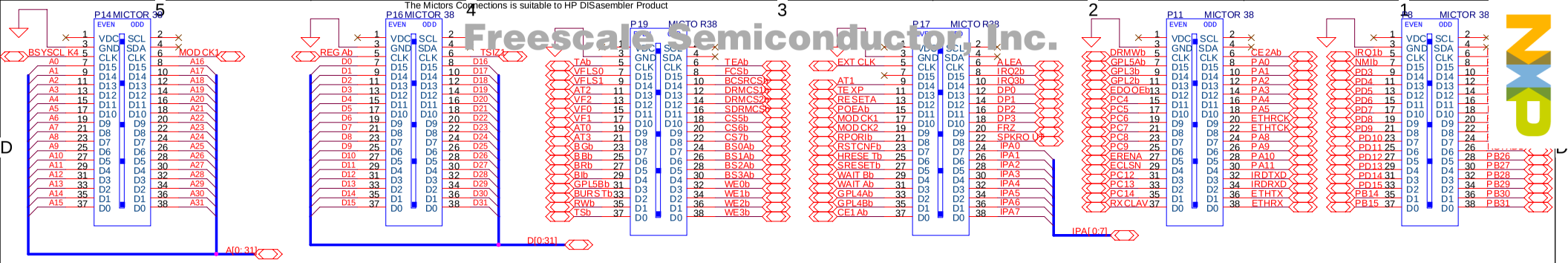
Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Israel 46120

MOTOROLA

Title	MPC86 6ADS	
Size A3	Document Number 01-SDRAM	Rev A
Date: Tuesday, December 17, 2002	Sheet 1 of 20	


Freescale Semiconductor, Inc.
For More Information On This Product,
Go to: www.freescale.com
Freescale Semiconductor, Inc.
Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Israel 46120
MOTOROLA
Title
<Title>
Size A3
Document Number <Doc>
Rev A
Date: Tuesday, December 17, 2002
Sheet 2 of 20
MPC86 2/6
IDT74CBTLV3253 PG
NOTE1:
IN EACH SCC FOR ETHERNET 10BT THE SIGNALS:
RTSx=TENAx
CDx=RENAx
CTSx=CLSNx
NOTE2:
IF ATM MUX IS USED The ATM US E PORT-D AND MII USE IP_A
NOTE3:
IF ATM SPLIT IS USED The ATM USE POR T-D & IP_A IN THIS CASE MII IS NO T USED

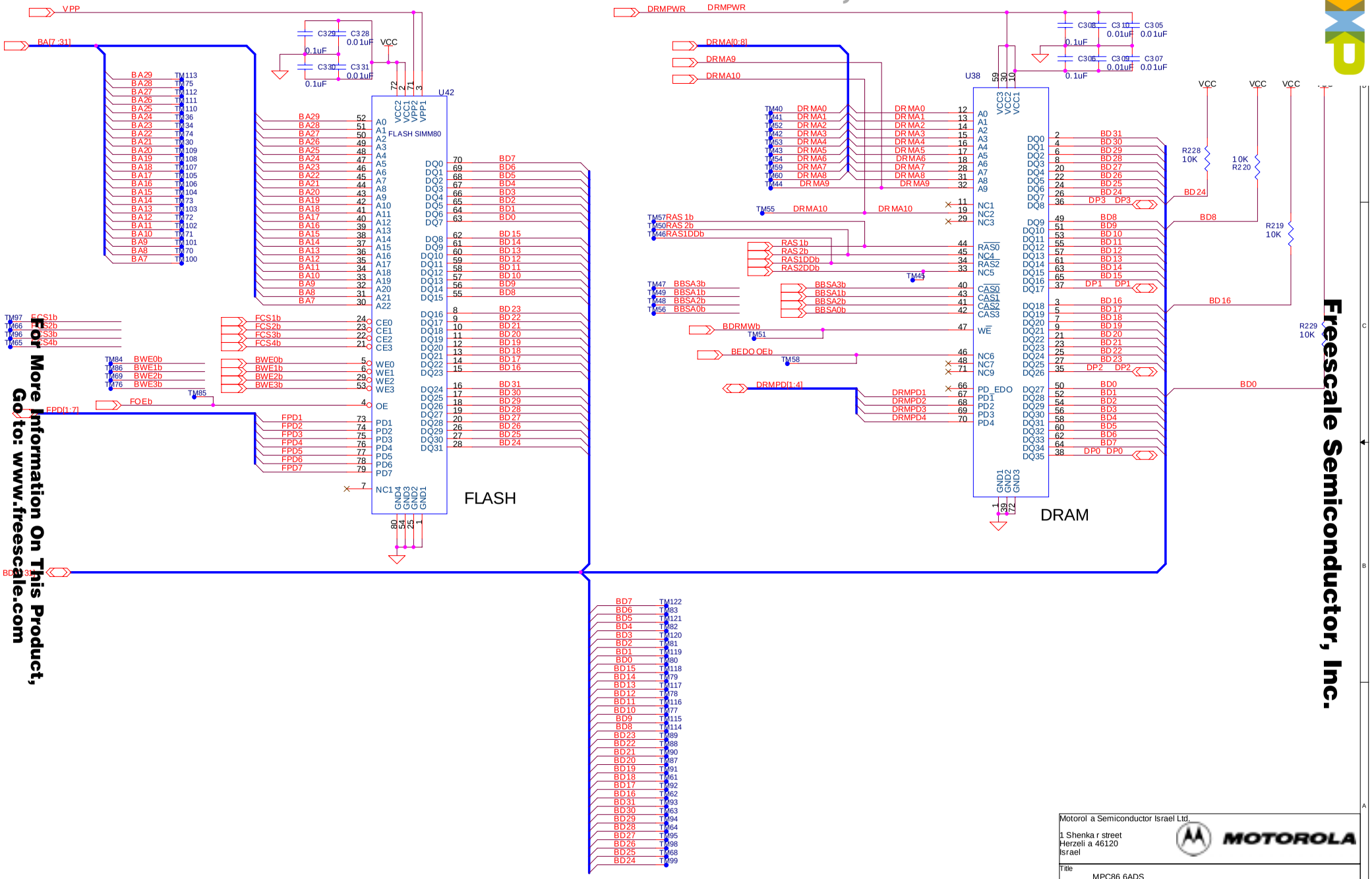




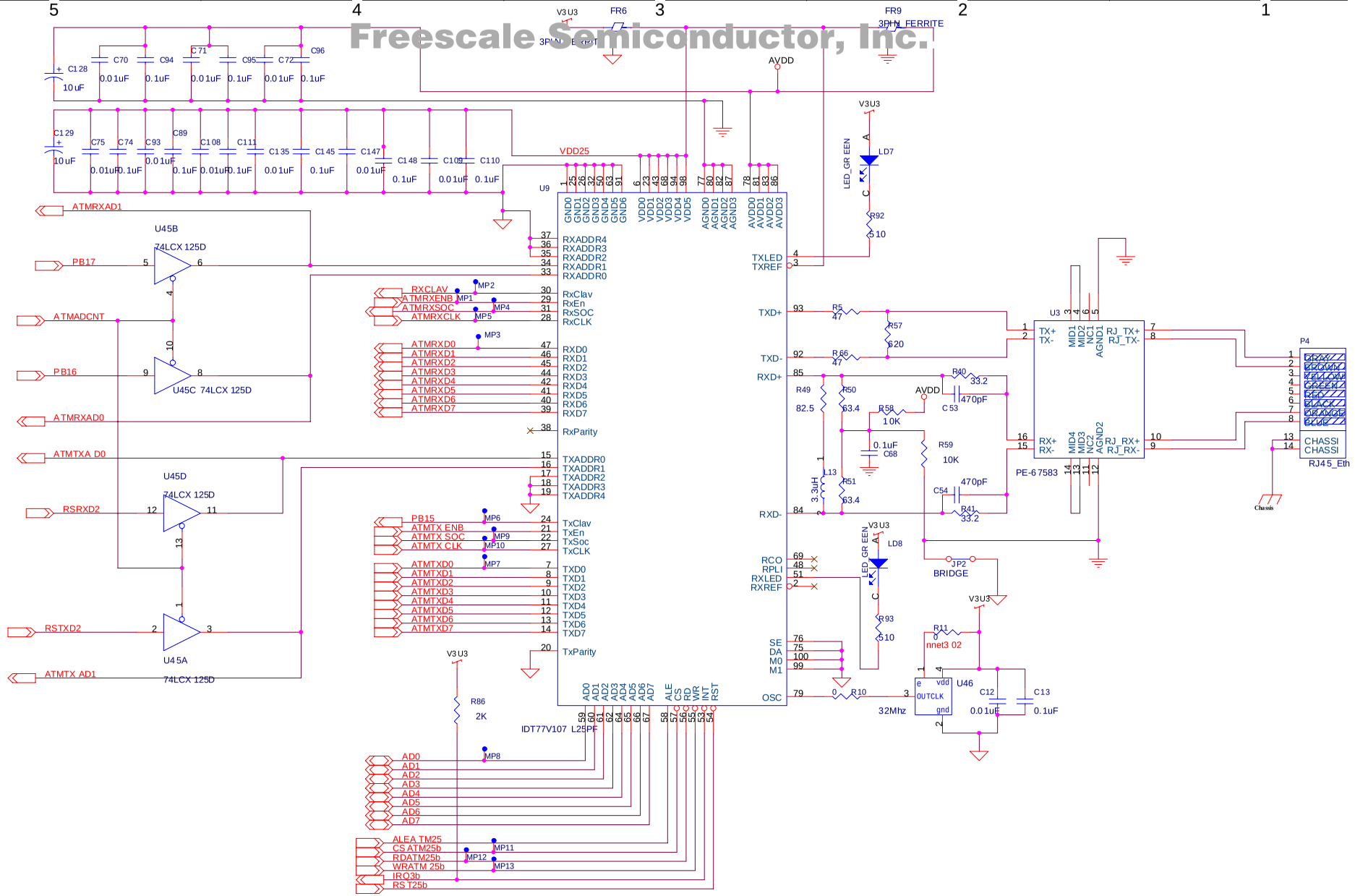
Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Isarel 46120

MOTOROLA

Title	MPC8 66ADS	
Size A3	Document N umber 04-LOGIC ANALYZER	Rev A
Date: Tuesday , December 17, 2002	Sheet 4 of 20	



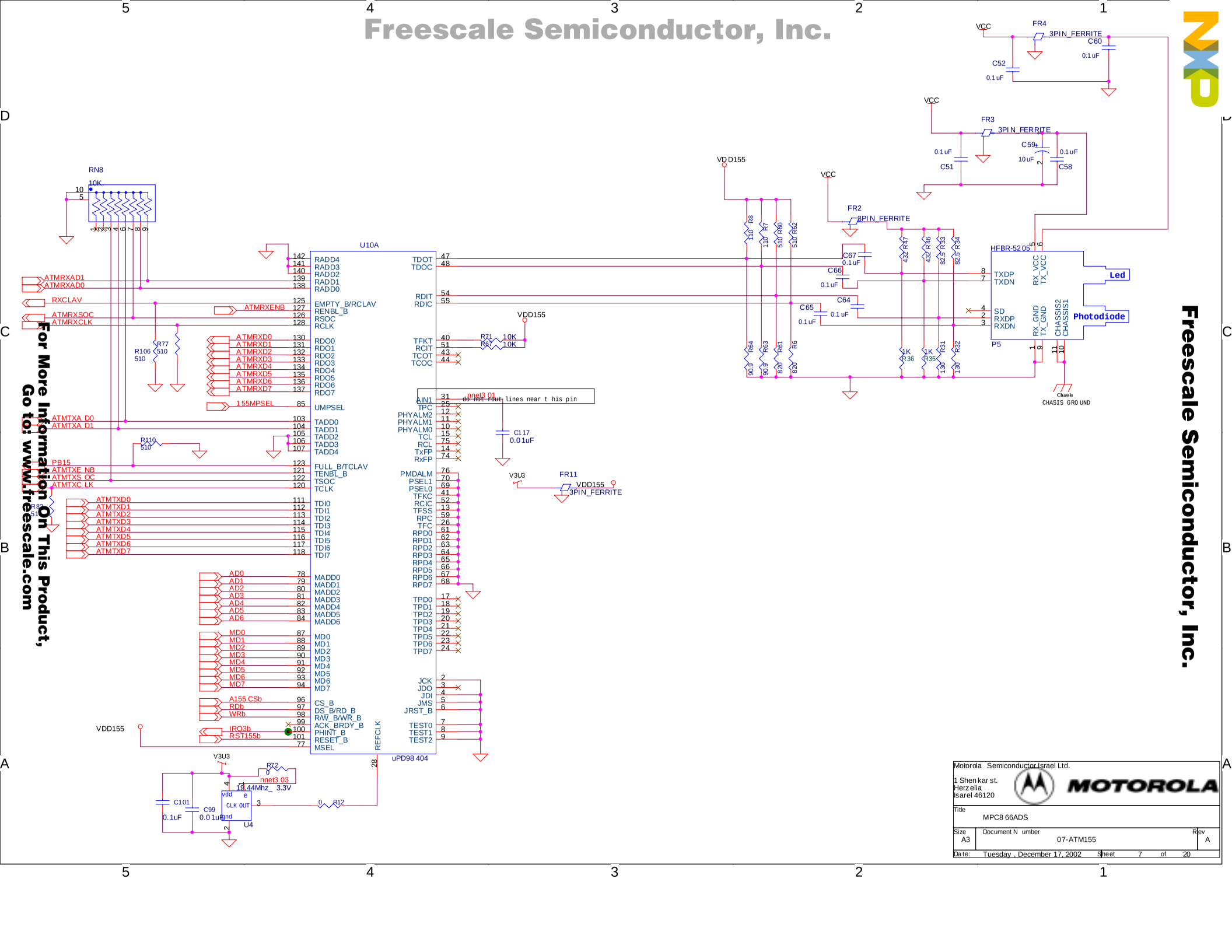
Motorola Semiconductor Israel Ltd.	
1 Shenka r street Herzeli a 46120 Israel	MOTOROLA
Title: MPC86 6ADS	
Size: A3	Document N umber: 05-FLASH&DRAM — Rev: A
Date: Tuesday, December 17, 2002	Sheet 5 of 20



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

Freescale Semiconductor, Inc.

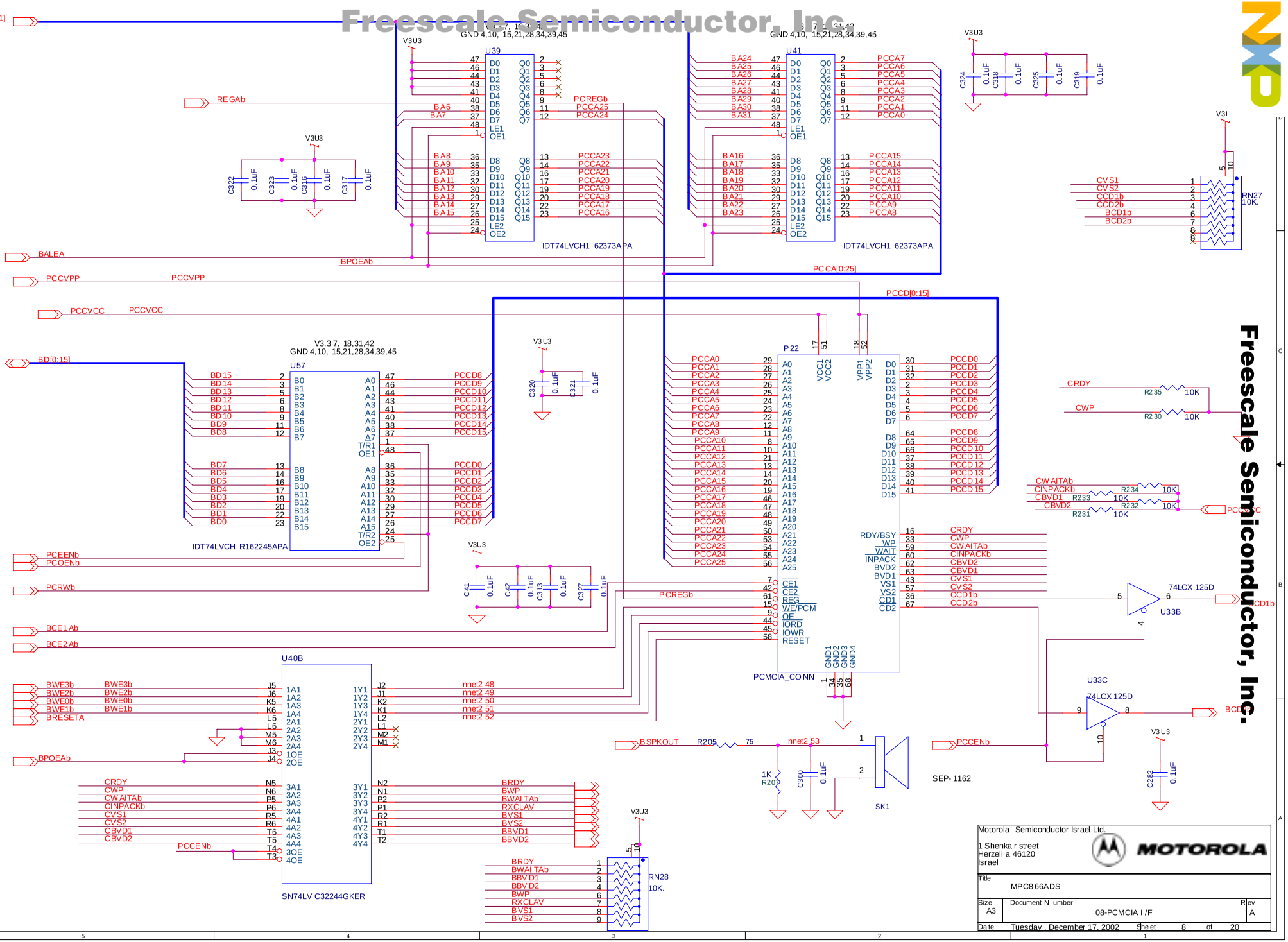
Motorola Semiconductor Israel Ltd.	
1 Shenkar st. Herzelia Israel 46120	MOTOROLA
Title	MPC8 66ADS
Size: A3	Document Number: 06-ATM25 — Rev: A
Date: Tuesday , December 17, 2002	Sheet 6 of 20



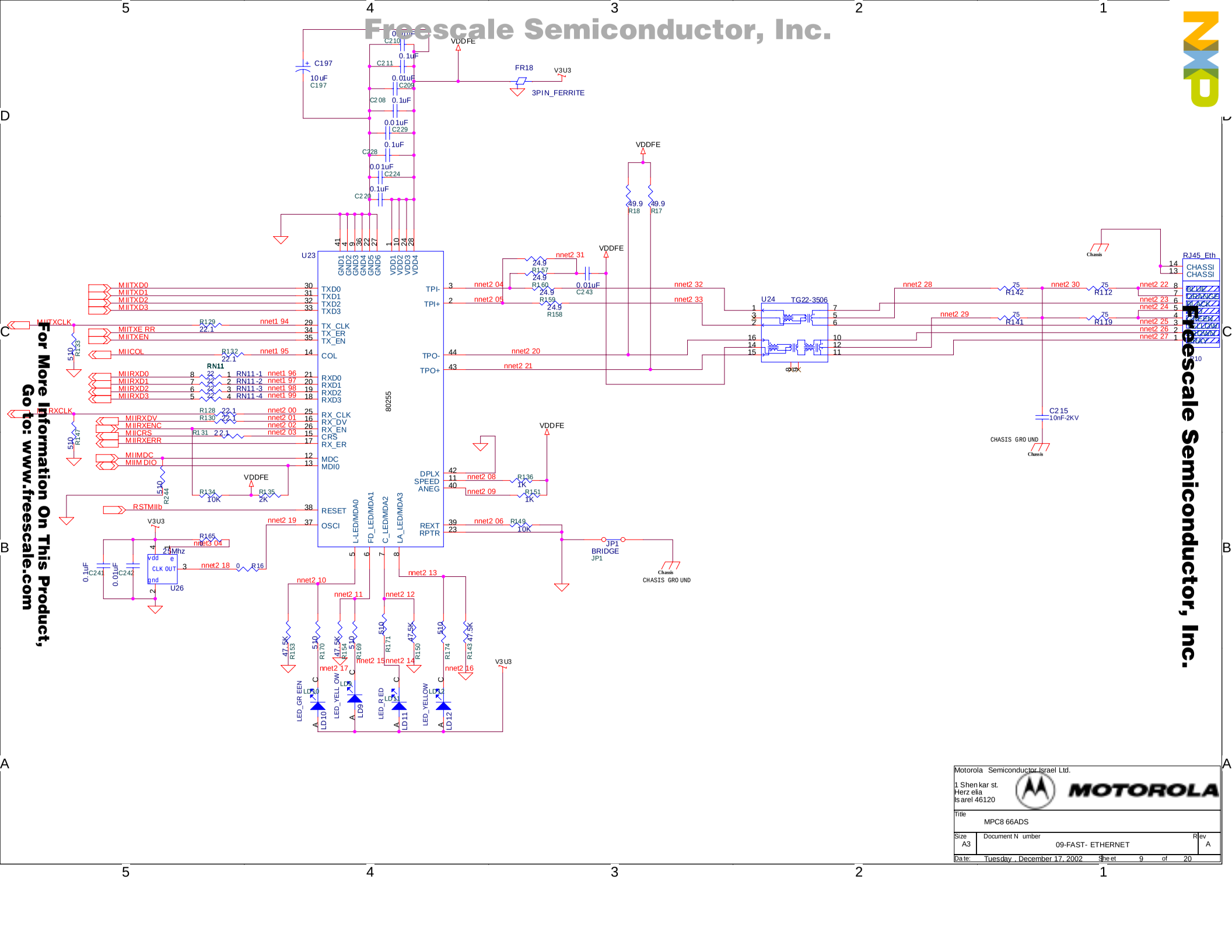
For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

Motorola Semiconductor Israel Ltd.	
1 Shenkar st. Herzelia Isarel 46120	MOTOROLA
Title	MPC8 66ADS
Size A3	Document Number 07-ATM155 Rev A
Date: Tuesday , December 17, 2002	Sheet 7 of 20



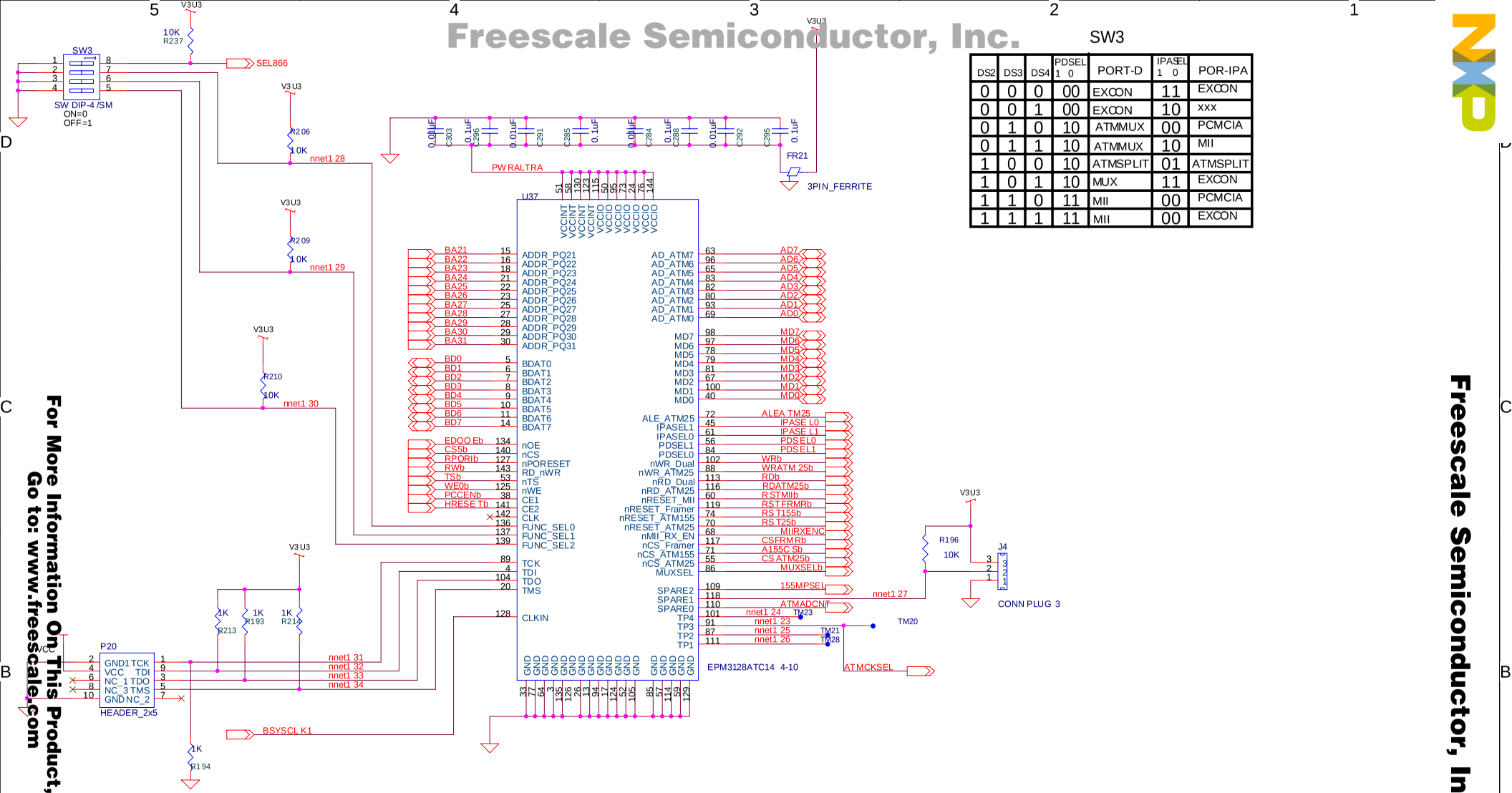
Motorola Semiconductor Israel Ltd.	
1 Shenka r street Herzeli a 46120 Israel	MOTOROLA
Title: MPC8 66ADS	
Size: A3	Document N umber: 08-PCMCIA I /F — Rev: A
Date: Tuesday , December 17, 2002	Sheet 8 of 20



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

Freescale Semiconductor, Inc.

Motorola Semiconductor Israel Ltd.	
1 Shenkar st. Herzelia Israel 46120	MOTOROLA
Title: MPC8 66ADS	
Size: A3	Document Number: 09-FAST- ETHERNET — Rev: A
Date: Tuesday , December 17, 2002	Sheet 9 of 20



DS2	DS3	DS4	PDSEL 1 0	PORT-D	IPASEL 1 0	POR-IPA
0	0	0	00	EXCON	11	EXCON
0	0	1	00	EXCON	10	xxx
0	1	0	10	ATMMUX	00	PCMCIA
0	1	1	10	ATMMUX	10	MII
1	0	0	10	ATMSPLIT	01	ATMSPLIT
1	0	1	10	MUX	11	EXCON
1	1	0	11	MII	00	PCMCIA
1	1	1	11	MII	00	EXCON

SW3

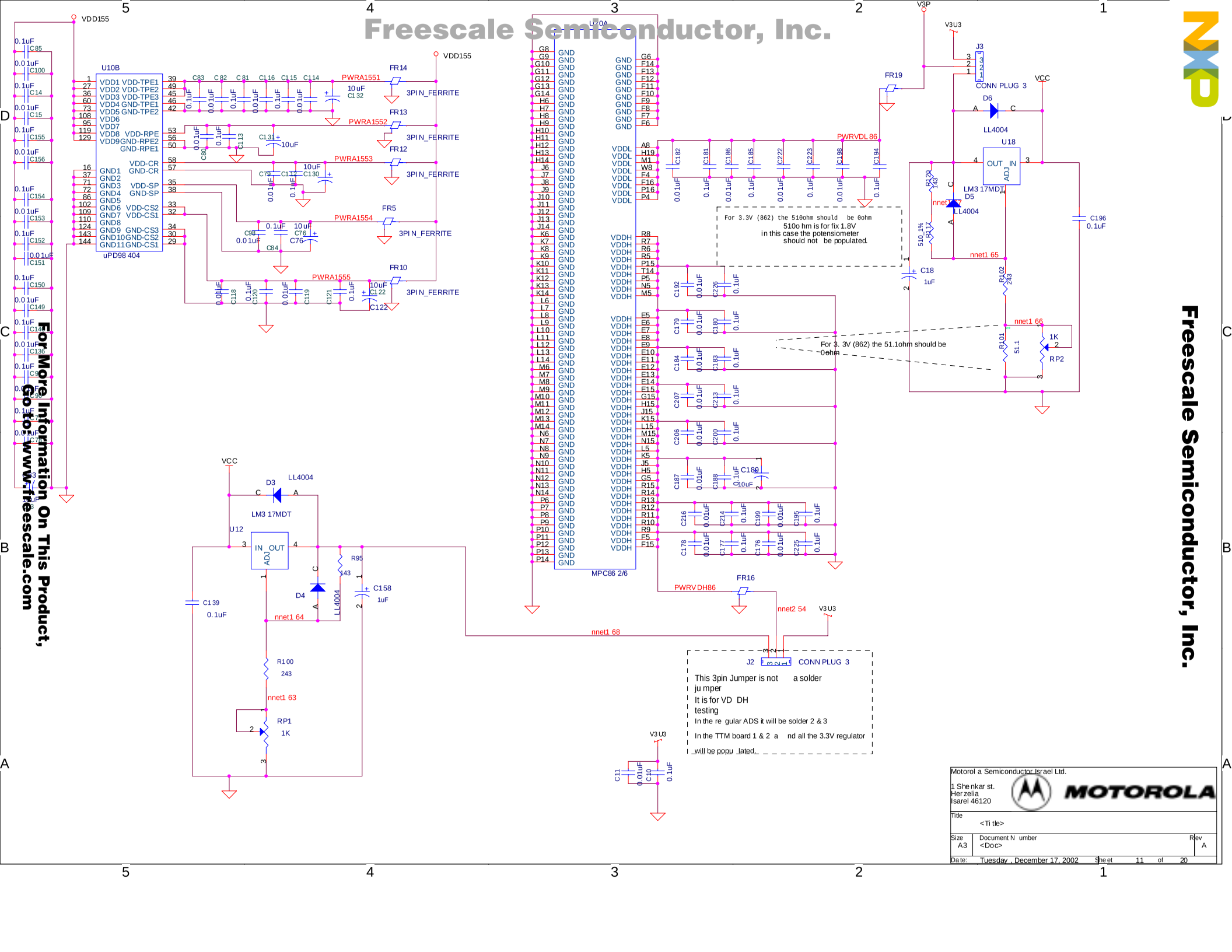
Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Israel 46120

MOTOROLA

Title: MPC8 66ADS

Size: A3 | Document Number: 10-CONTROL | Rev: A

Date: Tuesday, December 17, 2002 | Sheet 10 of 20

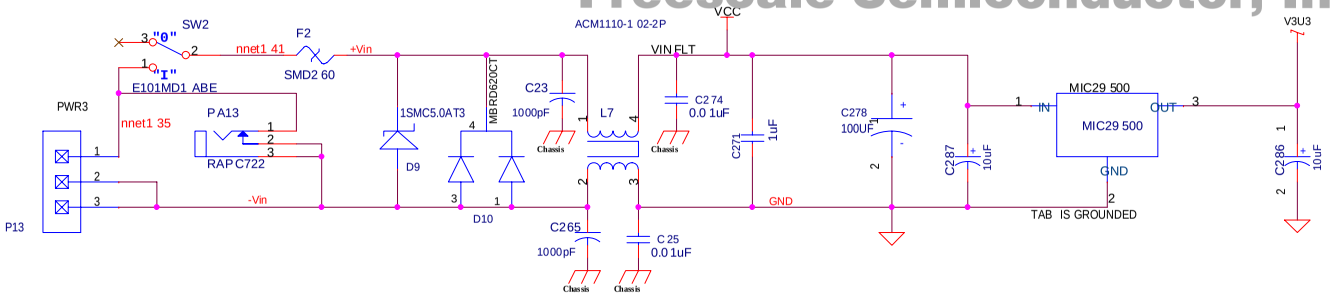


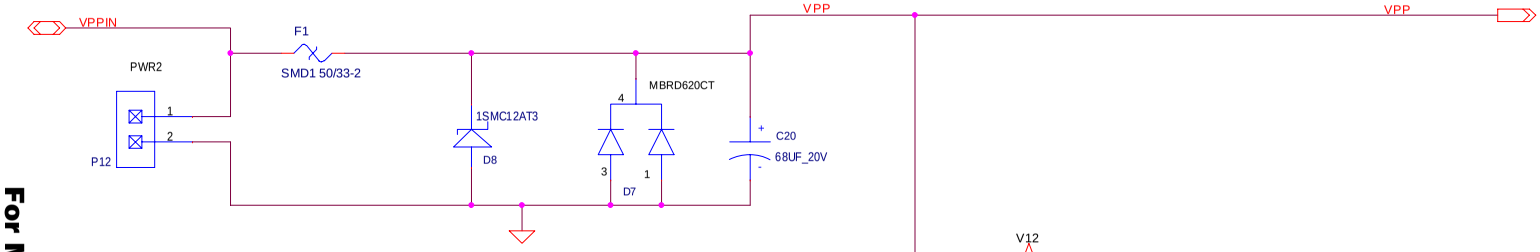
Motorol a Semiconductor Israel Ltd.
1 She nkar st.
Her zelia
Isarel 46120

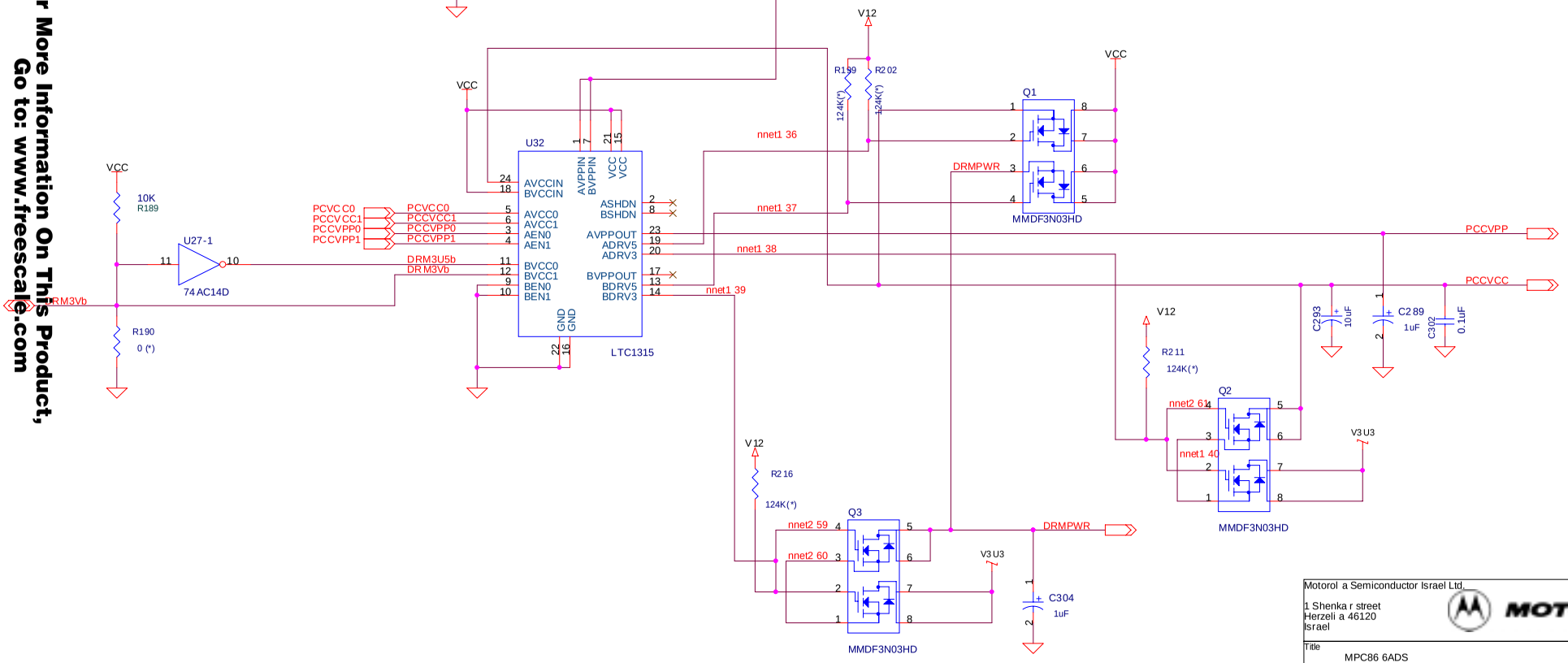
MOTOROLA

Title	<Ti tle>	
Size A3	Document N umber <Doc>	Rev A
Da te: Tuesday , December 17, 2002	She et 11 of 20	

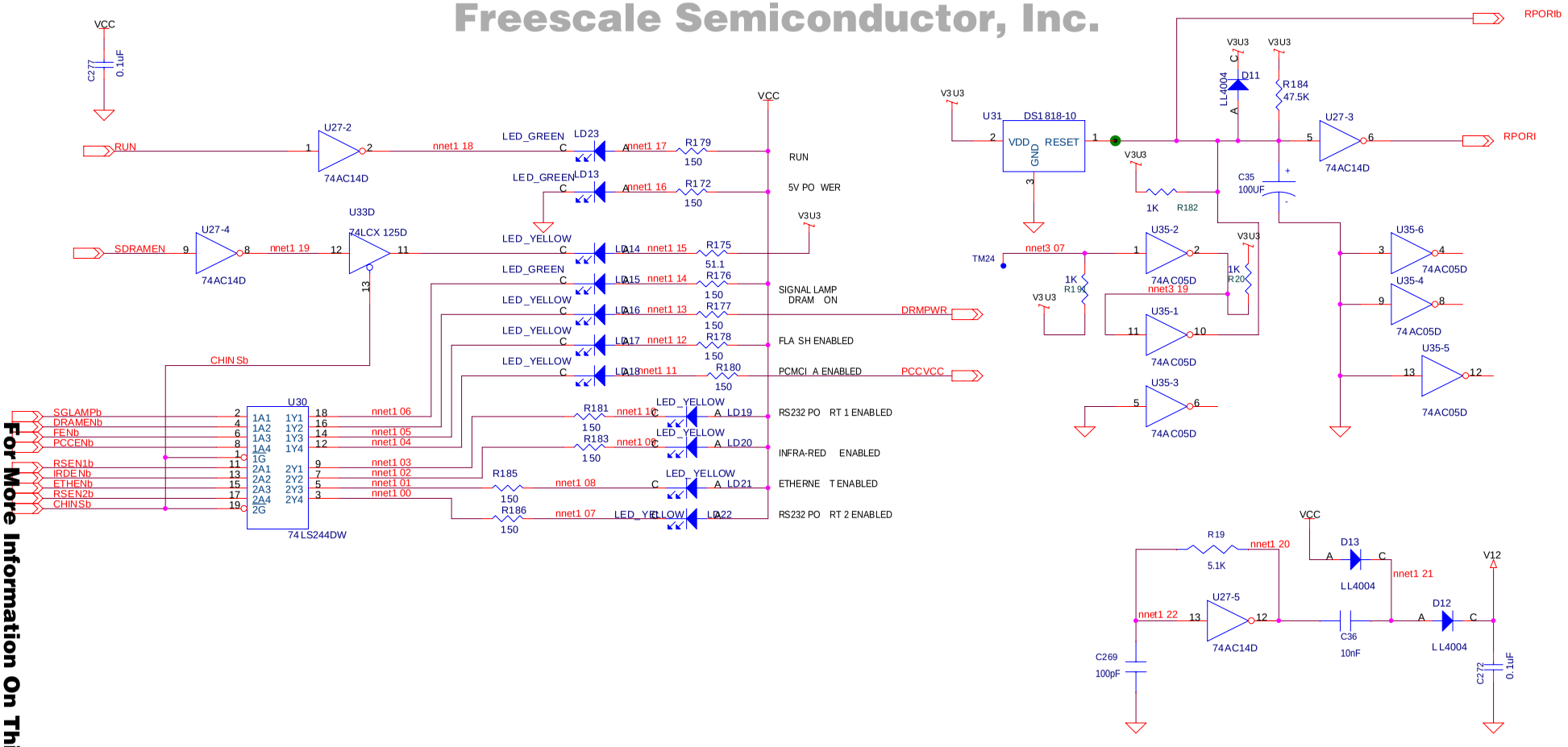
"PWR"







Motorola Semiconductor Israel Ltd.	
1 Shenkar street Herzeli a 46120 Israel	MOTOROLA
Title: MPC86 6ADS	
Size: A3	Document Number: 12-POWER — Rev: A
Date: Tuesday, December 17, 2002	Sheet 12 of 20



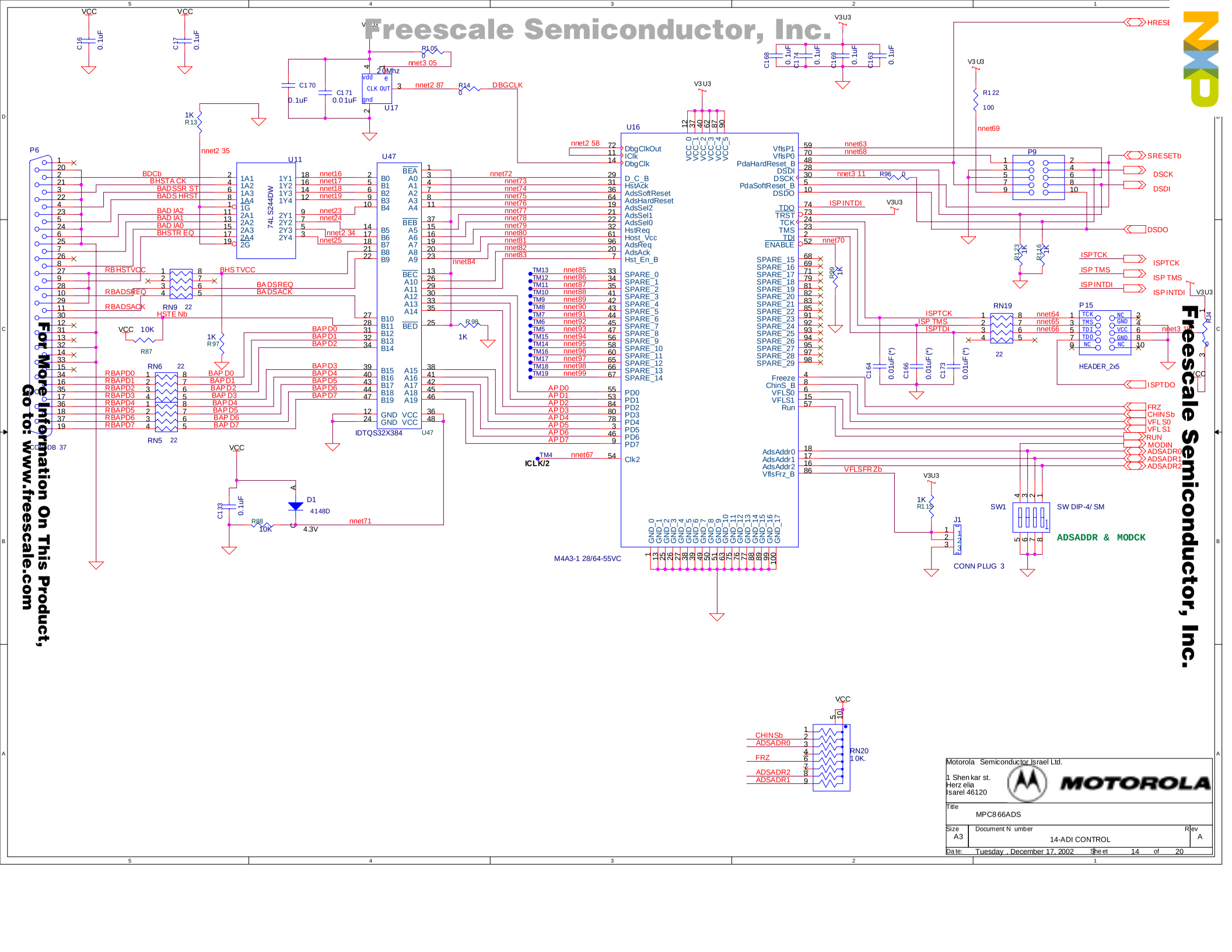
Motorola Semiconductor Israel Ltd.
1 Shenkar street
Herzeliya 46120
Israel

MOTOROLA

Title: MPC86 6ADS

Size	Document Number	Rev
A3	13-RESET & INDICATORS	A

Date: Tuesday, December 17, 2002 Sheet 13 of 20



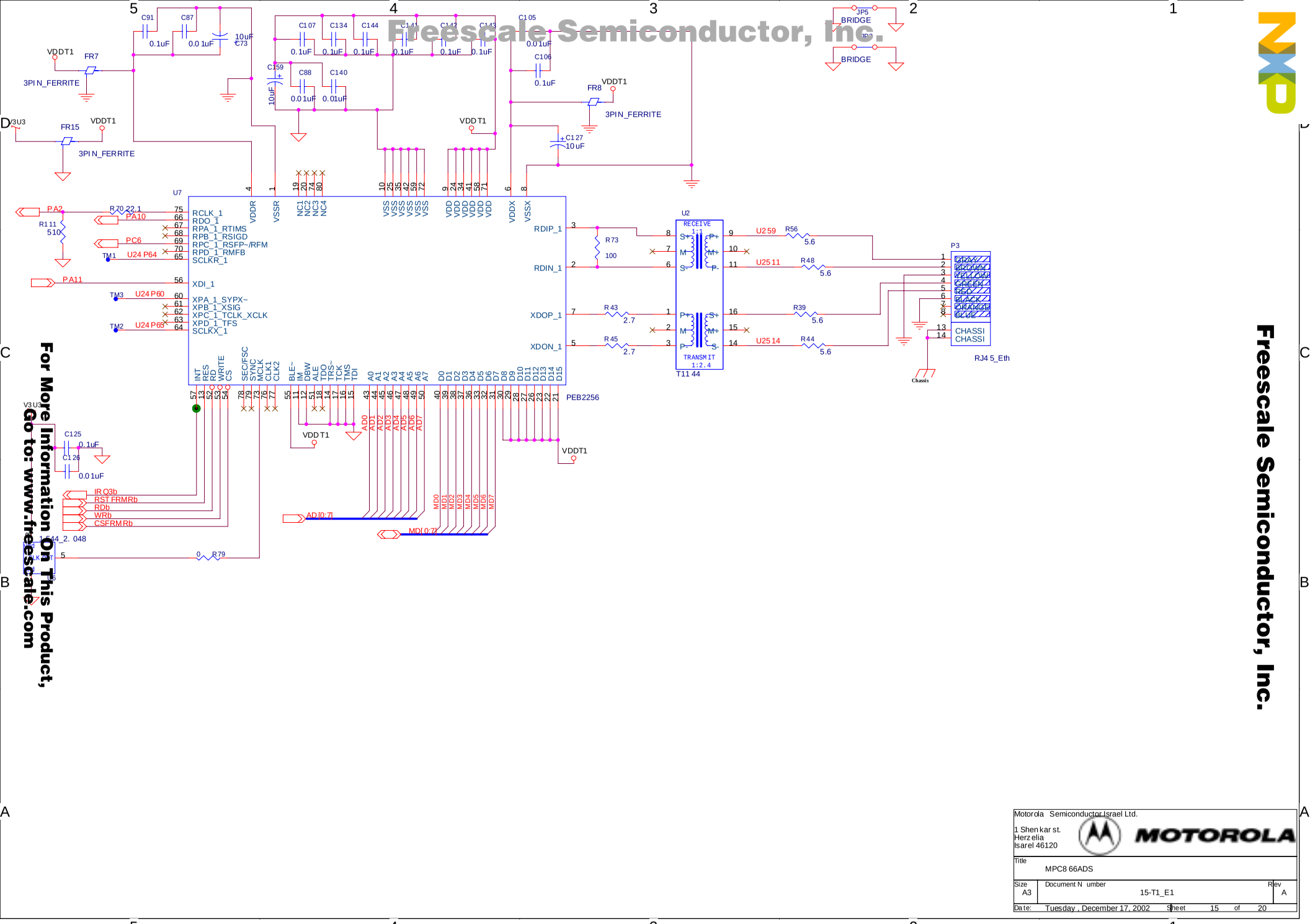
Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Israel 46120

MOTOROLA

Title: MPC866ADS

Size: A3 | Document Number: 14-ADI CONTROL | Rev: A

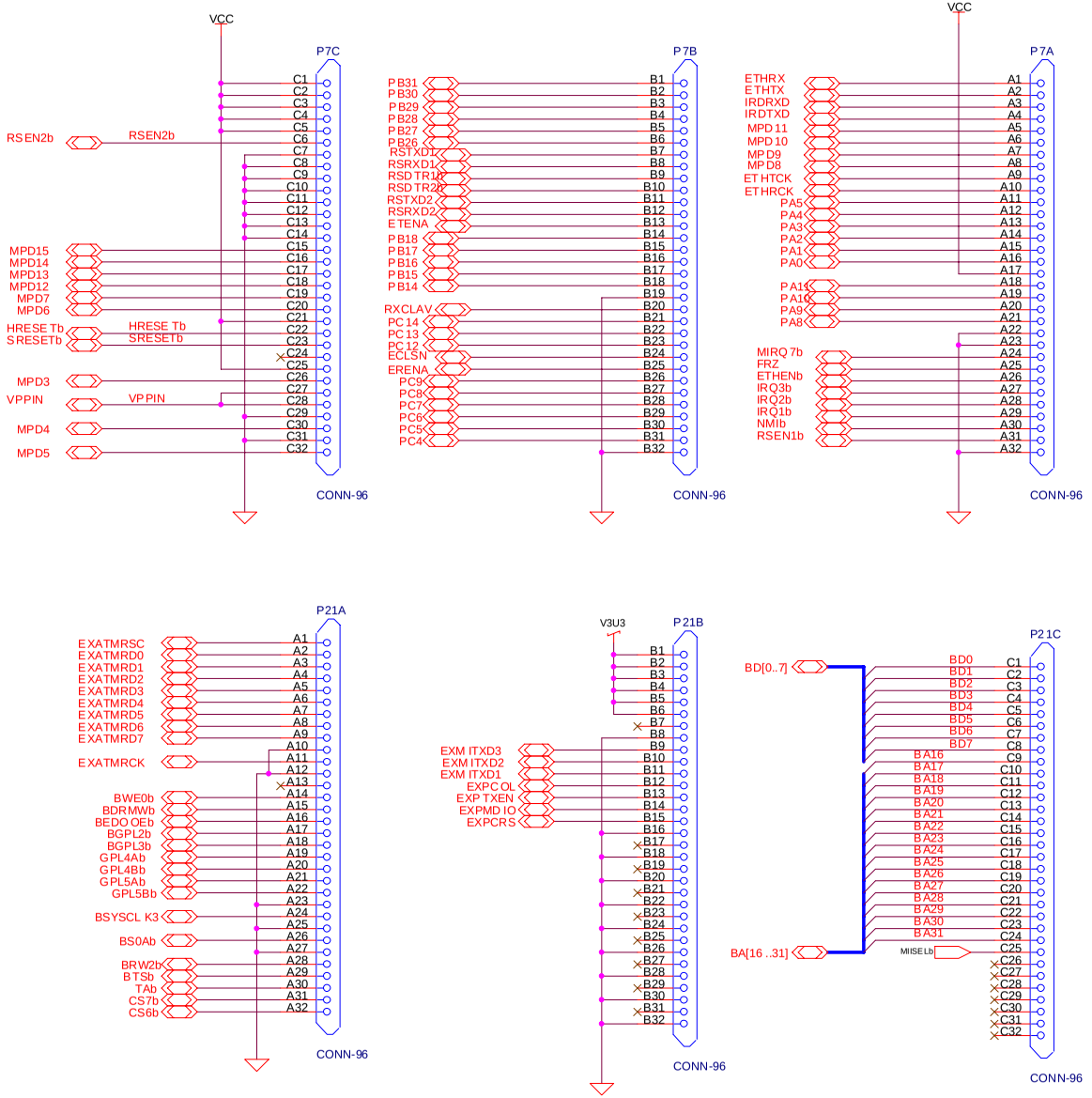
Date: Tuesday , December 17, 2002 | Sheet 14 of 20



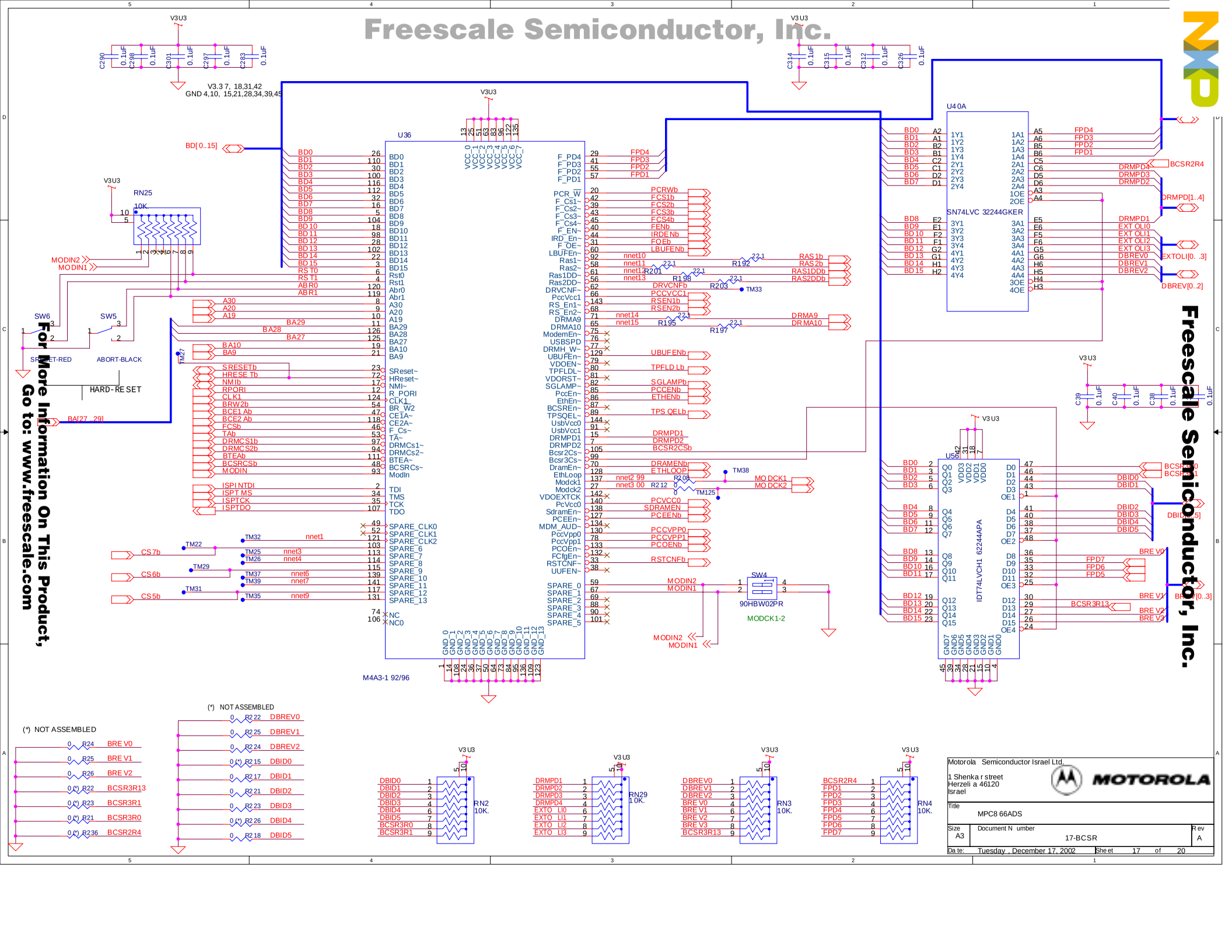
For More Information On This Product, Go to: www.freescale.com

Freescale Semiconductor, Inc.

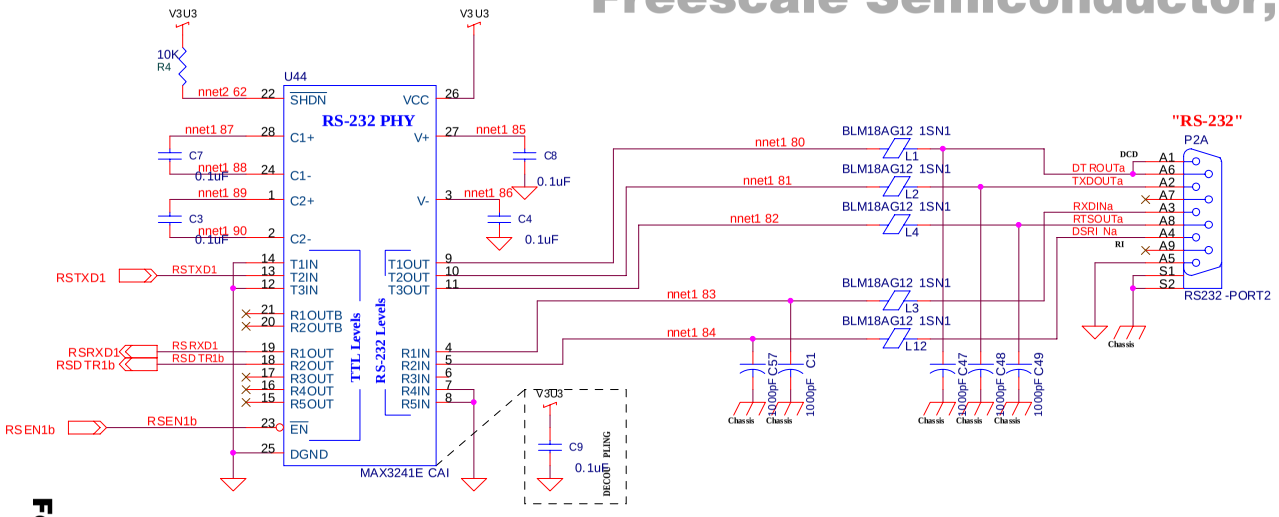
Motorola Semiconductor Israel Ltd.	
1 Shenkar st. Herzelia Isarel 46120	MOTOROLA
Title: MPC8 66ADS	
Size: A3	Document Number: 15-T1_E1 — Rev: A
Date: Tuesday , December 17, 2002	Sheet 15 of 20



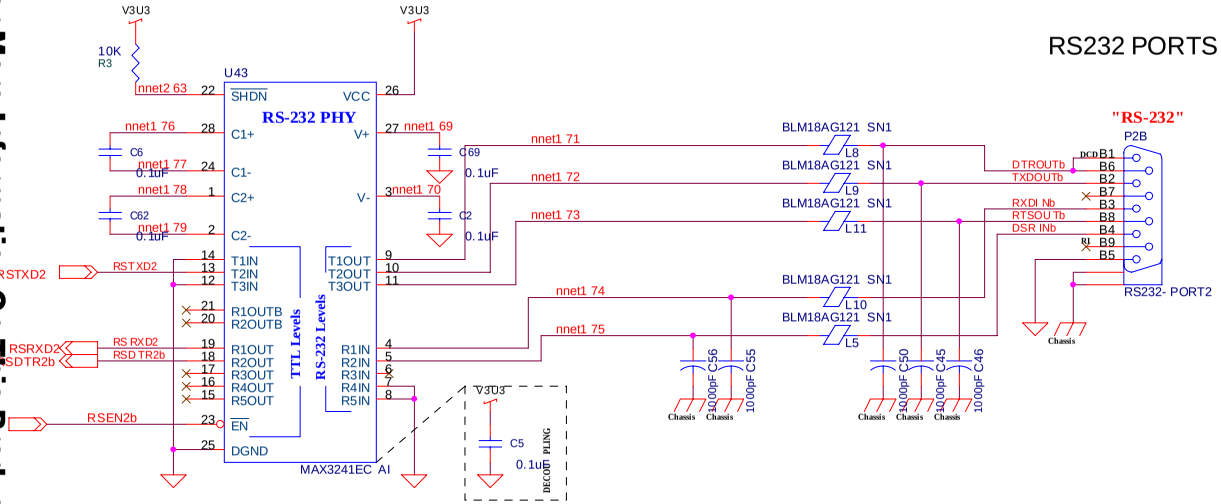
Motorola Semiconductor Israel Ltd. 1 Shenkar street Herzeliya 46120 Israel		MOTOROLA
Title: MPC866ADS		
Size: A3	Document Number: 16-EXP_CONN	Rev: A
Date: Tuesday, December 17, 2002	Sheet 16 of 20	

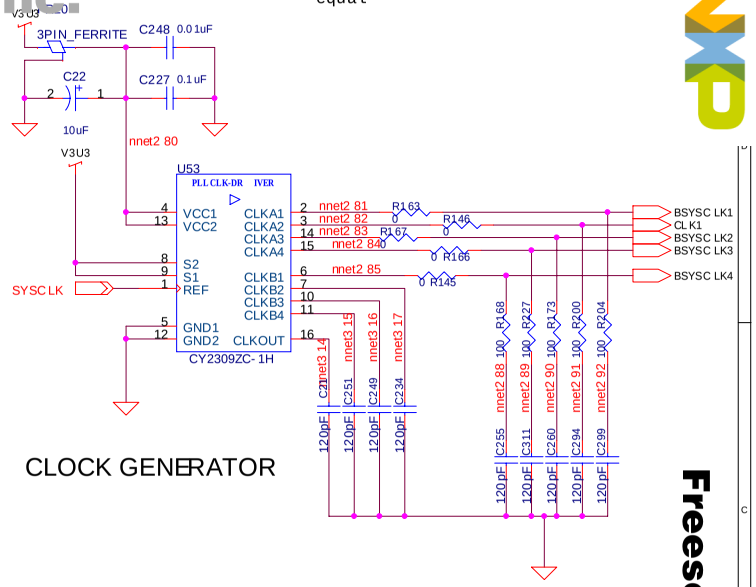


For output signals keep t race leng equal

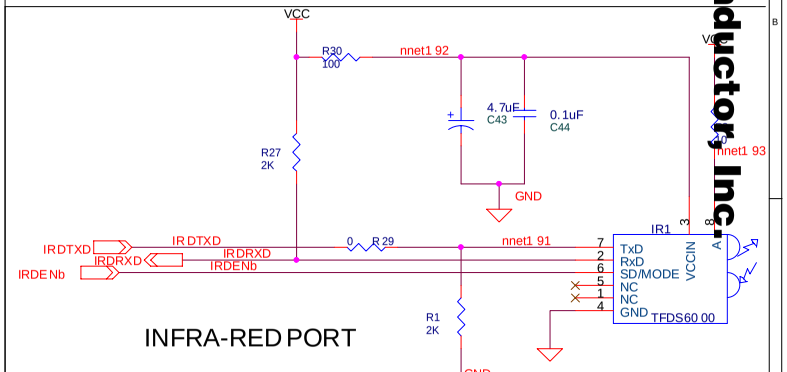


RS232 PORTS





CLOCK GENERATOR

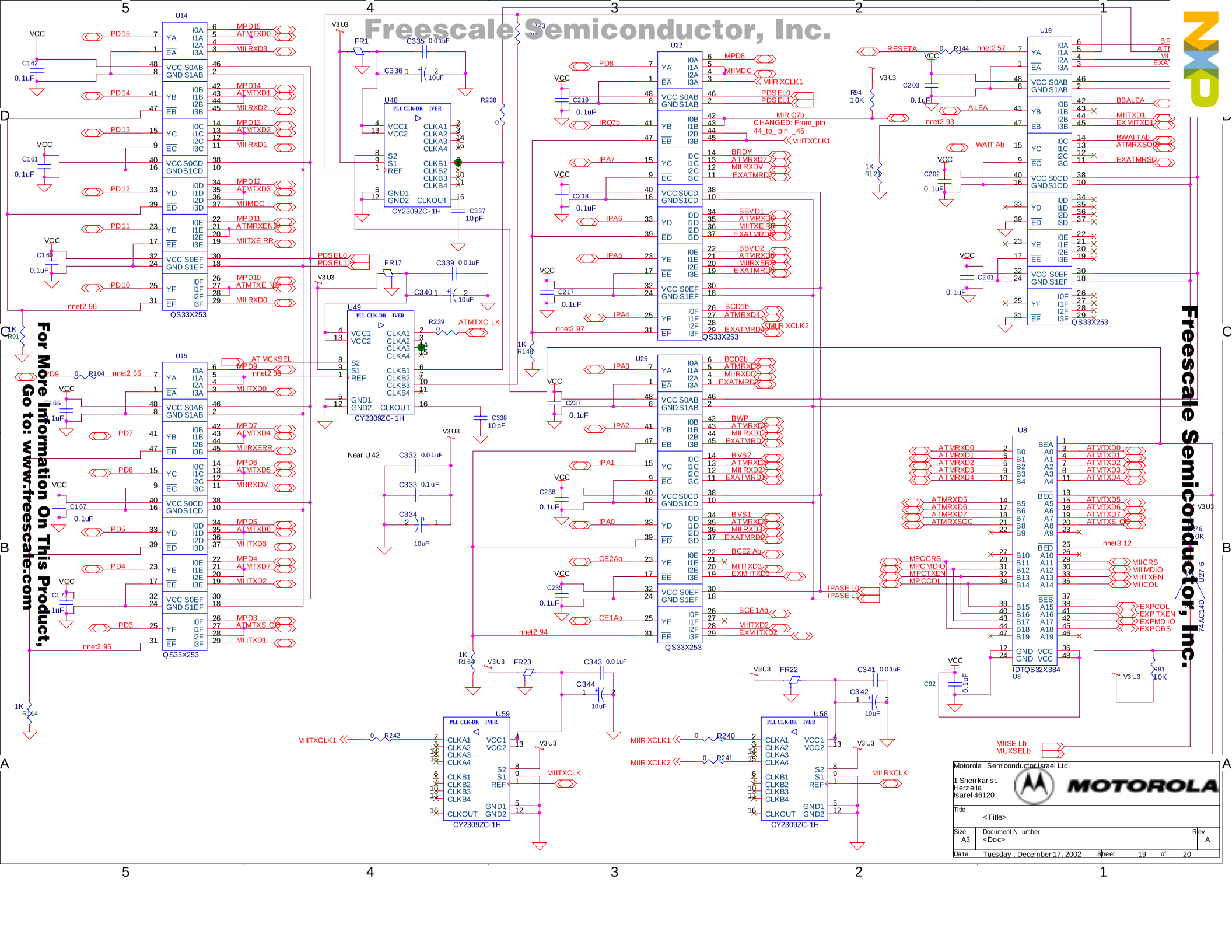


INFRA-RED PORT

Motorol a Semiconductor Israel Ltd.
1 Shen kar st.
Herz elia
Isarel 46120

MOTOROLA

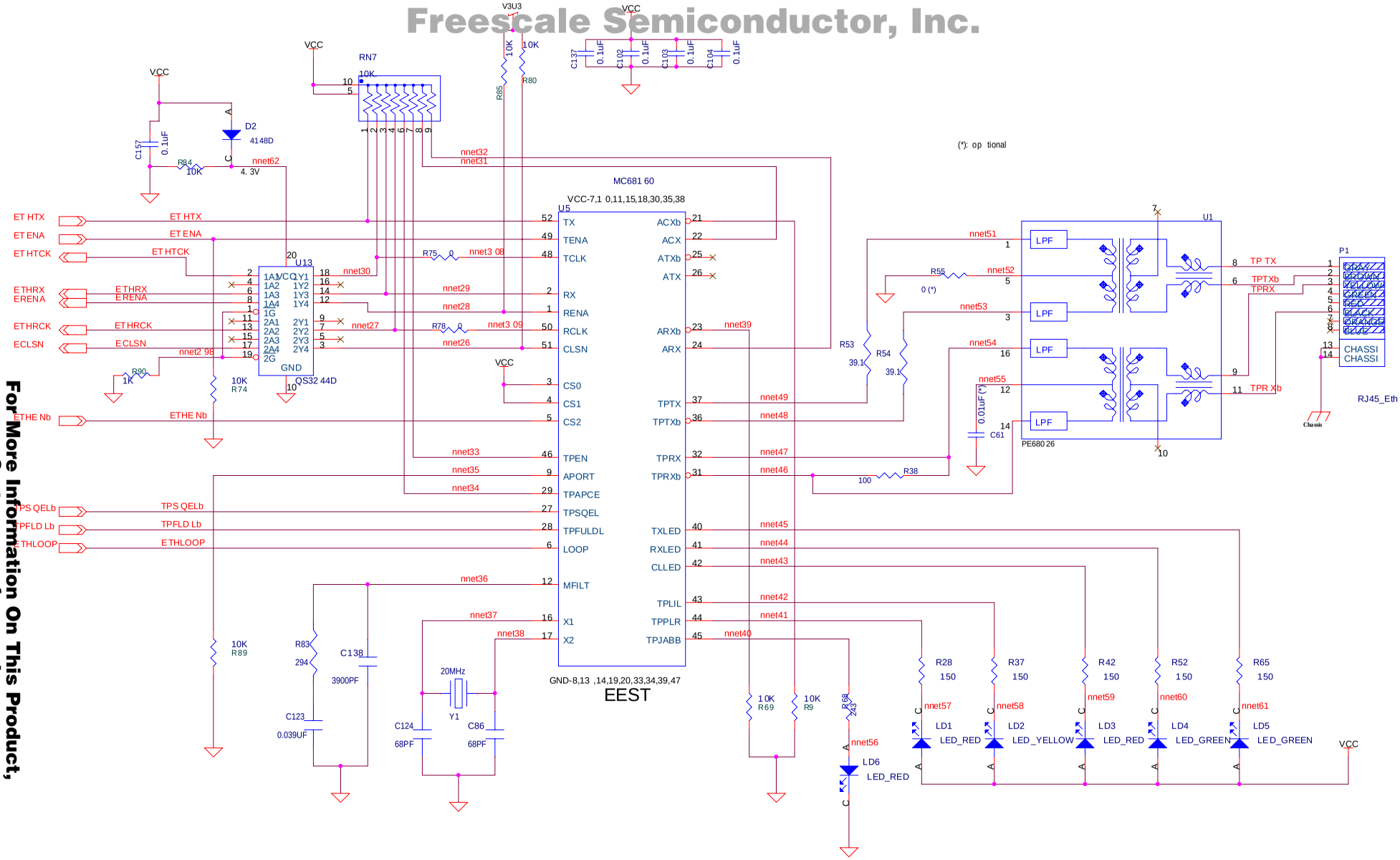
Title	MPC86 6ADS	
Size A3	Document N umber 18-RS232 & CLK	Rev A
Date: Tuesday, December 17, 2002	Sheet 18 of 20	



Motorola Semiconductor Israel Ltd.
1 Shenkar st.
Herzelia
Isarel 46120

MOTOROLA

Title	<Title>		
Size A3	Document N umber <Doc>		Rev A
Date: Tuesday , December 17, 2002	Sheet 19 of 20		



Motorol a Semiconductor Israel Ltd.	
1 Shenka r street Herzeli a 46120 Israel	MOTOROLA
Title: MPC86 6ADS	
Size: A3	Document N umber: 20-ETH ERNET — Rev: A
Date: Tuesday, December 17, 2002	Sheet 20 of 20

Support Information

APPENDIX B - Programmable Logic Equations

The MPC86xADS has 3 programmable logic devices on it. Use is done with Lattice - M4A3 - 128/64, Lattice - M4 - A3 - 192/96, Altera EPM3128ATC144-10 These devices support the following function on the ADS:

1) U16 - Debug Port Controller
2) U36 -The BCSR1-4, auxiliary board control functions, e.g,. buffers control, local interrupter, reset logic, etc'.
3) U37 - BCSR5. Control the ATM & fast ethernet devices and functionality.

2•0•1 U2 - Debug Port Controller

```
'*****************************************************************************
'* Pda ADS Debug Port Controller.                                            *
'* Mach controller for an interface between Sun ADI port at one side, to     *
'* debug port at the other.                                                  *
'*****************************************************************************
'* In this file (9):
'* - The pinout are changed to the new device M4A3-128/64-55VC
'*****************************************************************************
'* In this file (8):
'* - Added support for VFLS / FRZ switching, for both normal operation and
'*   external debug station connected to the FADS.
'*   Support includes:
'*   - separating vfls between MPC and debug port.
'*   - Selection between frz / vfls (default) is done by VflsFrz_B input pin.
'*   - Selection between on-board / off-board vfls/frz is done by ChinS_B coming
'*     from the expansion connectors.
'*   - VFLS(0:1) are driven towards the debug-port connector when board is not
'*     selected, i.e., either ADI is disconnected or addresses don't match.
'*****************************************************************************
'* In this file (7):
'* - BundleDelay field in the control register is changed to debug port
'*   clock frquency select according to the following values:
'*   0 - divide by 8 (1.25 Mhz)
'*   1 - divide by 4 (2.5 Mhz)
'*   2 - divide by 2 (5 Mhz)
'*   3 - divide by 1 (10 Mhz) default.
'* - Added clock divider for 2 , 4, 8 output of which is routed externaly
'*   to the i/f clock input.
'*****************************************************************************
'* In this file (6):
'* - RUN siganl polarity was changed to active-high, this, to support
'*   other changes for revision PILOT of the ads.
'*****************************************************************************
'* In this file (5) added:
'* - protection against spikes on the reset lines, so that the interface
'*   will not be reset by an accidental spike.
'* - D_C_B signal was synchronized to avoid accidental write to control
'*   during data write.
'* - DSDI is given value (H) prior to negation of SRESET* to comply with 5XX
'*   family
'*****************************************************************************
'* In this file (4) the polarity of address selection lines is reversed so
'* that ON the switch represent address line at high and vice-versa
'*****************************************************************************
'* In this file (3) the IClk is not reseted at all so it can be used to
'* sync pda reset signals inside.
'* Added consideration for reset generated by the pda:
'* - when pda is reset (i.e., its hard | soft reset signals are asserted,
'*   it is not allowed for the host to initiate data trnasfer towards the pda.
'*   It can however, access the control / status register to either change
'*   parameters and / or check for status.
'* - The status of reset signals is added to the status register, so it can
```

```
"*   be polled by the host.
"*************************************************************************


module dbg_prt7
title  'MPC821ADS Debug Port Controller.
       Originated for Ptec ADS, Shlomo Reches (MSIL) - October 26, 1993
       Modified for Pda ADS, Yair Liebman - (MSIL) - April 04, 1995'

"*************************************************************************
"* Device declaration.                                                   *
"*************************************************************************
"U02 device 'mach220a';

"*************************************************************************


"*************************************************************************
"* Pins declaration.                                                     *
"*************************************************************************
"ADI Port pins.

HstReq              PIN 32 ;         "Host to ADS, write pulse. (IN)

AdsAck              PIN 20 ISTYPE 'reg, buffer';  "ADS to host, write ack.
                                    "(OUT,3s)

AdsReq              PIN 96 ISTYPE 'reg, buffer';   "ADS to host, write
     "signal. (OUT,3s)

HstAck              PIN 31;         "Host to ADS, write ack.   (IN)

AdsHardReset        PIN 64;         "Host to ADS, Hard reset.  (IN)

AdsSoftReset        PIN 36;         "Host to ADS, Soft reset.  (IN)

HstEn_B             PIN 7;          "Host connected to ADS.    (IN)

HostVcc             PIN 61;         "Host to ADS, host is on.  (IN)

D_C_B               PIN 29;         "Host to ADS, select data
                          "or control access.        (IN)

AdsSel0             PIN 22;
AdsSel1             PIN 21;
AdsSel2             PIN 19;  "Host to ADS, card addr.   (IN)

AdsAddr0            PIN 18;
AdsAddr1            PIN 17;
AdsAddr2            PIN 16; "ADS board address switch. (IN)

AdsSelect_B         NODE ISTYPE 'com, buffer'; "ADS selection indicator. (OUT)
```

```
"*****************************************************************************
"* MPC pins. Including debug port.                                           *
"*****************************************************************************
PdaHardReset_B      PIN 48;       "Pda's hard reset input.  (I/O. o.d.)
PdaSoftReset_B      PIN 5;        "Pda's soft reset output. (I/O. o.d.)
VFLS0               PIN 6 ;
VFLS1               PIN 15 ;      "Debug/Trap mode, report. (IN)
Freeze              PIN 4;        "Alternative debug mode report (IN)
DSCK                PIN 30 istype 'com'; "Pda's debug port clock. (Out)
DSDI    PIN 28 istype 'com'; "Pda's debug serial data in (Out)
DSDO    PIN 10;   " Pda's debug serial data output (In)
"*****************************************************************************
"* Dedicated Debug Port pins.
"*****************************************************************************
VflsP0    PIN 70 istype 'com';
VflsP1              PIN 59 istype 'com';
VflsFrz_B           PIN 86;             " selectes between VFLS/FRZ from MPC

"*****************************************************************************
"* Mach to ADI data bus.                                                     *
"*****************************************************************************
PD7,
PD6,
PD5,
PD4,
PD3,
PD2,
PD1,
PD0 PIN 9,46,3,78,80,84,53,55;  "ADI data bus.(I/O)
"*****************************************************************************
"* Clock gen pins.                                                           *
"*****************************************************************************
DbgClk            PIN 14;               "Debug Clock input source. (IN)
DbgClkOut   PIN 72 istype 'com' ;  " to be connected to IClk. (out)

Clk2       PIN 54 istype 'reg, buffer'; "IClk divided by 2 (Out)
       " (Out for testing, may be node)
IClk   PIN 11;         " Connected to Clkout externally (In)
"*****************************************************************************
"* Misc.
"*****************************************************************************
Run          PIN 57 istype 'com';   "external indication
ChinS_B           PIN 8;                " active (L) when chips is In socket

"*****************************************************************************
"*****************************************************************************
"* Clock genrator Internals:
"*****************************************************************************
DbgClkDivBy2    NODE istype 'reg, buffer';
DbgClkDivBy4    NODE istype 'reg, buffer';
DbgClkDivBy8    NODE istype 'reg, buffer'; " counter (divider) signals.

Cstr0         NODE istype 'reg, buffer';
Cstr1         NODE istype 'reg, buffer'; " Clock Safe Transition Register
```

```
"*****************************************************************************
"* Reset active. (Active when at least one of the reset sources is active)    *
"*****************************************************************************
PrimReset       NODE istype 'com';        " Primary Reset. Host initiated
D_PrimReset     NODE istype 'com';        " delayed Reset
DD_PrimReset    NODE istype 'com';        " double delayed primary reset.

Reset           NODE istype 'com';        " Interface reset.
PdaRst          NODE istype 'reg, buffer'; " pda continued / initiated.
                                          " part of the status register.

"*****************************************************************************
"* ADS_ACK, ADS_REQ auxiliary internal control signals                        *
"*****************************************************************************
S_HstReqNODE istype 'reg';  "sync. host req.
DS_HstReqNODE istype 'reg';  " double sync. host req.

S_D_C_BNODE istype 'reg, buffer'; " synchronized data/ control selection

S_HstAckNODE istype 'reg, buffer'; " sync host ack
DS_HstAckNODE istype 'reg, buffer'; " double sync host ack

BundleDelay1,
BundleDelay0NODE istype 'reg, buffer';  "delay counter for bundle
" delay compensation
BndTmrExpNODE istype 'com';" terminal count for bundle
" delay timer.
PDOeNODE istype 'com';      "Mach to ADI data OE.

PdaHardResetEnNODE istype 'com';    " enables hard reset buffer.
PdaSoftResetEnNODE istype 'com';    " enables soft reset buffer.

"*****************************************************************************
"* Tx Shift Register                                                         *
"*****************************************************************************
TxReg7,
TxReg6,
TxReg5,
TxReg4,
TxReg3,
TxReg2,
TxReg1,
TxReg0NODE istype 'reg, buffer'; " Transmit latch and
" shift register
"*****************************************************************************
"* Tx Control Logic                                                          *
"*****************************************************************************
TxWordLen3,
TxWordLen2,
TxWordLen1,
TxWordLen0NODE istype 'reg, buffer'; " Counter, counts (on fast clock,
  " to gain 1/2 clock resolution)
```

```
  " transmission length

TxWordEndNODE istype 'com'; " Terminal count, sets transmission
  " length.
TxEnNODE istype 'reg, buffer'; " Transmit Enable.
TxClkSnsNODE istype 'reg, buffer'; " transmit clock polarity


"*****************************************************************************
"* Rx Shift Register                                                         *
"*****************************************************************************
RxReg0NODE istype 'reg, buffer'; " receive shift register
" and latch
"*****************************************************************************
"* Rx Control Logic                                                          *
"*****************************************************************************
DsdiEnNODE istype 'reg';   " enables dsdi towards

"*****************************************************************************
"* ADI control & status register bits.                                       *
"*****************************************************************************
StatusRequest_B     NODE istype 'reg, buffer';    "Status request
DebugEntry_B        NODE istype 'reg, buffer';    "Debug enable after reset (L)
DiagLoopBack_B      NODE istype 'reg, buffer';  "diagnostic loopback mode (L)
DbgClkDivSel0       NODE istype 'reg, buffer';
DbgClkDivSel1       NODE istype 'reg, buffer';    " DbgClk division select
InDebugMode         NODE istype 'reg, buffer';    " sync. VFLSs, became pin
TxError             NODE istype 'reg, buffer';    " tx interrupted by pda
                                    " internal reset.

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

"*****************************************************************************
"* Since all state machines operate at interface clock (IClk) there is no
"* need to have DbgClk driven during simulation (it will double the number
"* of vectors required). Therefore, an alternative clock generator was built
"* with which the 1/2 clock is the 1'st in the chain.
"* This alternative clock is compiled in if the SIMULATION variable is defined.
"* If not the original clock generator design is compiled, however simulation
"* will not pass then.
"*****************************************************************************
"* SIMULATION = 1;
"*****************************************************************************

"*****************************************************************************
"* Signal groups
"*****************************************************************************
AdsSel          = [AdsSel2,AdsSel1,AdsSel0];
AdsAddr         = [!AdsAddr2,!AdsAddr1,!AdsAddr0];

AdsRst          = [AdsHardReset, AdsSoftReset];
Rst             = [PdaHardReset_B, PdaSoftReset_B];
```

```
ClkOut          = [Clk2];
DbgClkDiv       = [DbgClkDivBy8, DbgClkDivBy4, DbgClkDivBy2];
DbgClkDivSel    = [DbgClkDivSel1, DbgClkDivSel0];
Cstr            = [Cstr1, Cstr0];

PD              = [PD7,PD6,PD5,PD4,PD3,PD2,PD1,PD0];
VFLS            = [VFLS0, VFLS1];
VflsP           = [VflsP0,VflsP1];
BndDly          = [BundleDelay1, BundleDelay0];  "bundle delay
 "compensation timer
TxReg           = [TxReg7..TxReg0];
RxReg           = [TxReg6,TxReg5,TxReg4,TxReg3,TxReg2,TxReg1,TxReg0,RxReg0];
AdiCtrlReg      = [DbgClkDivSel1, DbgClkDivSel0, StatusRequest_B,
                  DiagLoopBack_B,DebugEntry_B];
AdiStatReg      = [PdaRst, TxError, InDebugMode, DbgClkDivSel1, DbgClkDivSel0,
StatusRequest_B, DiagLoopBack_B, DebugEntry_B];
TxWordLen       = [TxWordLen3, TxWordLen2, TxWordLen1, TxWordLen0];
PortEn          = [AdsSel2,AdsSel1,AdsSel0,!AdsAddr2,!AdsAddr1,!AdsAddr0,
                                        HostVcc,HstEn_B];
"*****************************************************************************
"* Select Logic definitions
"*****************************************************************************
HOST_VCC_ACTIVE = 1;
HOST_EN_B_ACTIVE = 0;

HOST_IS_ON  = ((HstEn_B==HOST_EN_B_ACTIVE) & (HostVcc==HOST_VCC_ACTIVE));
HOST_IS_OFF = !HOST_IS_ON;

BOARD_IS_SELECTED = 0;
ADS_IS_SELECTED = (AdsSelect_B.fb==BOARD_IS_SELECTED) ;

"Data_Cntrl_B line levels.
DATA    = 1;
CONTROL = !DATA;

"*****************************************************************************
"* Reset Logic definitions
"*****************************************************************************
ADS_HARD_RESET_ACTIVE     = 1;
ADS_SOFT_RESET_ACTIVE     = 1;

"*****************************************************************************
"* Clock Logic definitions
"*****************************************************************************
SELECT_CHANGE_ALLOWED =  (DbgClkDiv.fb == 0);

DEBUG_CLOCK_DIV_BY_1 = (Cstr.fb == 0);
DEBUG_CLOCK_DIV_BY_2 = (Cstr.fb == 1);
DEBUG_CLOCK_DIV_BY_4 = (Cstr.fb == 2);
DEBUG_CLOCK_DIV_BY_8 = (Cstr.fb == 3);

"*****************************************************************************
"* AdsAck Logic definitions
```

```
"*****************************************************************************
BUNDLE_DELAY = 2;


HOST_REQ_ACTIVE     = 1;
ADS_ACK_ACTIVE      = 1;      "The other state is -  !ADS_ACK_ACTIVE
HOST_ACK_ACTIVE = 1;


HOST_WRITE_ADI            = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb ==HOST_REQ_ACTIVE) &
                        (AdsAck==!ADS_ACK_ACTIVE) &
                        (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_ADI_CONTROL      = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb ==HOST_REQ_ACTIVE) &
                        (AdsAck==!ADS_ACK_ACTIVE) &
                        (D_C_B==CONTROL) &
                        (S_D_C_B.fb == CONTROL) &
                        (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_ADI_DATA         = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb==HOST_REQ_ACTIVE) &
                        (AdsAck==!ADS_ACK_ACTIVE) &
                        (D_C_B==DATA) &
                        (S_D_C_B.fb ==DATA) &
                        (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_COMPLETE         = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb==!HOST_REQ_ACTIVE) &
                        (AdsAck==!ADS_ACK_ACTIVE) );

"*****************************************************************************
"* Control & Status register definitions
"*****************************************************************************
STATUS_REQUEST= 0;
DEBUG_ENTRY= 0;
DIAG_LOOP_BACK= 0;
IN_DEBUG_MODE   = 1;

TX_DONE_OK= 0;
TX_INTERRUPTED  = !TX_DONE_OK;

FRZ_SELECTED    = 0;


IS_STATUS_REQUEST = (StatusRequest_B.fb == STATUS_REQUEST);
DEBUG_MODE_ENTRY  = (DebugEntry_B.fb == DEBUG_ENTRY);
IN_DIAG_LOOP_BACK = (DiagLoopBack_B.fb == DIAG_LOOP_BACK);
IS_IN_DEBUG_MODE  = (InDebugMode.fb == IN_DEBUG_MODE);
FRZ_IS_SELECTED   = (VflsFrz_B == FRZ_SELECTED);


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

```
"* DSDI_ENABLE Logic definitions
"*******************************************************************************

DSDI_ENABLED = 1;
DSDI_DISABLED = 0;

STATE_DSDI_ENABLED = (DsdiEn.fb == DSDI_ENABLED);

"*******************************************************************************
"* Tx enable state machine
"*******************************************************************************
TX_ENABLED = 1;
TX_DISABLED = 0;

STATE_TX_ENABLED = (TxEn.fb == TX_ENABLED);
STATE_TX_DISABLED = (TxEn.fb == TX_DISABLED);

TX_WORD_LENGTH  = 14;  " In 1/2 IClk clocks

"*******************************************************************************
"* TxClkSns state machine
"*******************************************************************************
TX_ON_RISING = 0;
TX_ON_FALLING = 1;

STATE_TX_ON_RISING = (TxClkSns.fb == TX_ON_RISING);
STATE_TX_ON_FALLING = (TxClkSns.fb == TX_ON_FALLING);

"*******************************************************************************
"* AdsReq machine definitions.
"*******************************************************************************
ADS_REQ_ACTIVE = 1;     "The other state is -  !ADS_REQ_ACTIVE

HOST_READ_ADI             = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                        (AdsReq==ADS_REQ_ACTIVE) &
                        (HstReq==!HOST_REQ_ACTIVE) );

HOST_READ_ADI_DATA        = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                        (HstReq==!HOST_REQ_ACTIVE) &
                        (AdsReq==ADS_REQ_ACTIVE) &
                        (D_C_B==DATA) );

HOST_READ_ADI_CONTROL     = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                        (HstReq==!HOST_REQ_ACTIVE) &
                        (AdsReq==ADS_REQ_ACTIVE) &
                        (D_C_B==CONTROL) );


ADS_SEND_STATUS           = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb == !HOST_REQ_ACTIVE) &
                        (D_C_B==CONTROL) &
```

```
                    (AdsAck==ADS_ACK_ACTIVE) &
                     IS_STATUS_REQUEST );

"*****************************************************************************
"* ADI Data Bus definitions
"*****************************************************************************
DATA_BUFFERS_ENABLE       = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                    (HstAck==HOST_ACK_ACTIVE) &
                    (HstReq==!HOST_REQ_ACTIVE) );

STATUS_WORD_ON_ADI_BUS    = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                    (HstAck==HOST_ACK_ACTIVE) &
                    (HstReq==!HOST_REQ_ACTIVE) &
                    (D_C_B==CONTROL) );

READ_DATA_WORD_ON_ADI_BUS  = ( (AdsSelect_B.fb==BOARD_IS_SELECTED) &
                    (HstAck==HOST_ACK_ACTIVE) &
                    (HstReq==!HOST_REQ_ACTIVE) &
                    (D_C_B== DATA) );

"*****************************************************************************
"* Vfls / Frz select definitions
"*****************************************************************************
CHIP_IN_SOCKET = 0;

CHIP_IS_IN_SOCKET = (ChinS_B == CHIP_IN_SOCKET);



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



"*****************************************************************************
"* AdsSelect.                                                    *
"* ADS selection indicator. At low state, when host accesses the ADS.        *
"*****************************************************************************
equations

  !AdsSelect_B = HOST_IS_ON & (AdsSel==AdsAddr);  "AdsAddr is already inverted

"*****************************************************************************
"* Internal Logic Reset.
```

```
"*****************************************************************************
equations

  PrimReset = HOST_IS_OFF #    "internal logic reset
            ( (AdsHardReset == ADS_HARD_RESET_ACTIVE) &
            (AdsSoftReset ==ADS_SOFT_RESET_ACTIVE) &
             ADS_IS_SELECTED );
  D_PrimReset = PrimReset.fb;
  DD_PrimReset = D_PrimReset.fb;

  Reset = PrimReset.fb & D_PrimReset.fb & DD_PrimReset.fb;" spike filter


"*****************************************************************************
"* Reset PDA. (Connected to PDA hard and reset inputs) Asynchronous.         *
"*****************************************************************************
equations

  !PdaHardReset_B = H;

  PdaHardReset_B.oe = PdaHardResetEn;  "open-drain


  PdaHardResetEn = ADS_IS_SELECTED &
   (AdsHardReset==ADS_HARD_RESET_ACTIVE);

  !PdaSoftReset_B = H;

  PdaSoftReset_B.oe = PdaSoftResetEn;  "needs to be open-drain


  PdaSoftResetEn = ADS_IS_SELECTED &
   (AdsSoftReset==ADS_SOFT_RESET_ACTIVE );

"*****************************************************************************
"*****************************************************************************
"* Clock generator.
"* All i/f logic works on IClk, which is driven externally by the output
"* of DbgClk divider.
"* The debug clock divider is a 3 bit free-running counter, outputs of which
"* control a 4:1 mux, output of which drives IClk (externally).
"* Since mux control may change on the fly, a protection logic by means of
"* 2 bit register is provided, so that mux control is allowed to change
"* only when all divider outputs are high which assures a falling edge prior
"* to a rising edge.
"* Clk2 is infact the source for DSCK and is available outside for debug
"* purpose.
"*****************************************************************************
equations

DbgClkDiv.clk = DbgClk;

DbgClkDiv := DbgClkDiv.fb + 1; " free running counter.
```

```
"*****************************************************************************
"* Clock Safe Transition Register. (CSTR)
"* The goal of this register is to provide safe clock transitions, i.e., that
"* a trasition will not cause races over the clockout. E.g., in a transition
"* between divide by 1 and divide by any bigger order, a possible race may
"* occur since the divided outputs are delayed with respect to DbgClk.
"* Therefore, a safe transition may be performed only when all clocks are LOW.
"*****************************************************************************
equations

Cstr.clk = DbgClk;
"* Cstr.ar = Reset;

when (SELECT_CHANGE_ALLOWED) then
Cstr := DbgClkDivSel.fb;
else
Cstr := Cstr.fb;

"*****************************************************************************
"* Clock selector.
"* Controlled by the CSTR.
"*****************************************************************************
equations

DbgClkOut.oe = 1;

when (DEBUG_CLOCK_DIV_BY_1) then
DbgClkOut = DbgClk;
else when (DEBUG_CLOCK_DIV_BY_2) then
DbgClkOut = DbgClkDivBy2.fb;
else when (DEBUG_CLOCK_DIV_BY_4) then
DbgClkOut = DbgClkDivBy4.fb;
else when (DEBUG_CLOCK_DIV_BY_8) then
DbgClkOut = DbgClkDivBy8.fb;

"*****************************************************************************
"* Clk2.
"* IClk divided by 2 .
"*****************************************************************************

equations

  Clk2.clk = IClk;
  ClkOut.oe = 3;
  ClkOut.ar = Reset;

  Clk2 := !Clk2 & HOST_IS_ON;        "divide by 2



"*****************************************************************************
"* Bundle delay timer. This timer ensures data validity in the following casses:
"* 1) Host write to adi. In that case AdsAck is ASSERTED only after that timer
"*    expired.
```

```
"* 2) Host read from adi. In that case AdsReq is NEGATED after that timer
"*    expired, ensuring enough time for data propgation over the bundle.
"* The timer is async reset when both soft and hard reset is applied to the i/f.
"* The timer is sync. reset a clock after it expires.
"* Count starts when either HstReq or HstAck are detected asserted
"* (after proper synchronization)
"* The value upon which the terminal count is assereted, is in the control
"* register. When the interface is reset by the host, this value defaults
"* to its upper bound. Using the diagnostic loop-back mode this value
"* may be re-established for optimal performance. (by means of test & error)
"*******************************************************************************

equations

  BndDly.ar = Reset;
  BndDly.clk = IClk;


  when ( ( (HOST_WRITE_ADI_CONTROL # HOST_READ_ADI_CONTROL ) #
     (HOST_WRITE_ADI_DATA # HOST_READ_ADI_DATA) & !PdaRst.fb)
      & !BndTmrExp.fb) then
  BndDly := BndDly.fb +1;
  else
  BndDly := 0;

  BndTmrExp = (BndDly.fb == BUNDLE_DELAY) & !AdsAck ; "delay field
    "active low.


"*******************************************************************************
"* AdsAck.
"* Host write to ads ack. This state machine generates an automatic ADS_ACK,
"* during a host to ADS write.
"* When the host access the ADS data / control register, an automatic
"* acknowledge is generated, after data has been latched into either the
"* tx shift register or the control register.
"* Acknowledge is released when the host removes its write control line.
"*  (HstReq)
"*
"* The machine steps through these states :
"* 0 - !ADS_ACK_ACTIVE
"* 1 - ADS_ACK_ACTIVE
"*******************************************************************************
equations
  AdsAck.clk  = IClk;
  AdsAck.ar   = Reset;
  AdsAck.oe   = ADS_IS_SELECTED;

  S_HstReq.clk = IClk;
  DS_HstReq.clk = IClk;

  S_HstReq := HstReq;
  DS_HstReq := S_HstReq.fb & HstReq;"double synced
```

```
  S_D_C_B.clk = IClk;" synchronizing D_C_B selector
  S_D_C_B := D_C_B;

state_diagram AdsAck
  state !ADS_ACK_ACTIVE:
    if ( (HOST_WRITE_ADI_CONTROL #
        (HOST_WRITE_ADI_DATA & !PdaRst.fb) )  & BndTmrExp.fb) then
       ADS_ACK_ACTIVE
    else
      !ADS_ACK_ACTIVE;

  state ADS_ACK_ACTIVE:
    if ( DS_HstReq.fb==!HOST_REQ_ACTIVE ) then
      !ADS_ACK_ACTIVE
    else
      ADS_ACK_ACTIVE;
"*****************************************************************************
"* Transmit Enable logic.
"* Enables transmit of serial data over DSDI and generation of serial
"* clock over DSCK.
"* Transmission begins immediately after data written by the host is latched
"* into the transmit shift register and ends after 7 shifts were made to the
"* tx shift register.
"* Termination is done using a 4 bit counter TxWordLength which has a terminal
"* count (and reset) TxWordEnd.
"*****************************************************************************
equations

  TxEn.ar = Reset;
  TxEn.clk = IClk;" to provide 1/2 clock resolution

state_diagram TxEn
state TX_DISABLED:
if(HOST_WRITE_ADI_DATA & BndTmrExp.fb & !PdaRst.fb) then
TX_ENABLED
else
TX_DISABLED;
state TX_ENABLED:
if(TxWordEnd # PdaRst.fb) then
TX_DISABLED
else
TX_ENABLED;
"*****************************************************************************
"* Transmit Length Counter. This counter determines the length of transmission
"* towards the MPC. The fast clock is used here to allow 1/2 clock resolution
"* with the negation of TxEn, which enables DSCK outside.
"*****************************************************************************
equations

TxWordLen.ar = Reset;
TxWordLen.clk = IClk;

TxWordEnd = (TxWordLen.fb == TX_WORD_LENGTH);
```

```
when ( STATE_TX_ENABLED & !TxWordEnd & !PdaRst.fb) then
TxWordLen.d = TxWordLen.fb + 1;
else
TxWordLen.d = 0;

"*****************************************************************************
"* TxClkSns - Transmit Clock Sense.
"* Since Host req is synced acc to IClk and may be detected active when Clk2 is
"* either '1' or '0', DSCK and the clock according to which DSDI is sent and
"* DSDO is sampled should be changed.
"* When TxClkSns is '0' - DSCK will be !Clk2 while transmit will be done
"*   according to Clk2 and recieve by !Clk2.
"* When TxClkSns is '1' - DSCK will be Clk2 while transmit will be done
"*   according to !Clk2_B and recieve by Clk2.
"*****************************************************************************
equations

  TxClkSns.clk = IClk;
  TxClkSns.ar = Reset;

state_diagram TxClkSns

state TX_ON_RISING:
if (HOST_WRITE_ADI_DATA & BndTmrExp.fb & Clk2) then
TX_ON_FALLING
else
TX_ON_RISING;
state TX_ON_FALLING:
if (HOST_WRITE_ADI_DATA & BndTmrExp.fb & !Clk2) then
TX_ON_RISING
else
TX_ON_FALLING;



"*****************************************************************************
"* Tx shift Register.
"* 8 bits shift register which either shifts data out (MSB first) or holds
"* its data. The edge (in Clk2 terms) upon which the above actions are taken,
"* is determined by TxClkSns. The Tx shift register operates according to IClk.
"* The Tx shift register is 1'st written by the host (data cycle) and along
"* with write being acknowledged to the host data is shifted out via DSDI.
"*
"* In order of saving logic, the Tx shift register is shared with the Receive
"* shift register, this, due to the fact that when a bit is shifted out a FF
"* becomes available. Since the Tx shift register is shifted MSB first, its
"* LSB FFs are grdualy becoming available for received data.
"* To provide a 1/2 DSCK hold time for DSDI, a single FF receive SR is used
"* which is the source for the Tx shift register. (if 0 hold is required
"* for DSDI this FF may be ommited)
"*****************************************************************************
equations

  TxReg.clk = IClk;
```

```
  TxReg.ar = Reset;


  when ( HOST_WRITE_ADI_DATA & BndTmrExp.fb & !STATE_TX_ENABLED) then
  [TxReg7..TxReg1] := [PD7..PD1].pin;   " latching ADI data
  else when (STATE_TX_ENABLED & STATE_TX_ON_RISING & !Clk2 #
       STATE_TX_ENABLED & STATE_TX_ON_FALLING & Clk2) then
  [TxReg7..TxReg1] := [TxReg6..TxReg0].fb;  " shifting out MSB 1'st.
  else
      [TxReg7..TxReg1] := [TxReg7..TxReg1].fb;  " Holding value.


  when ( HOST_WRITE_ADI_DATA & BndTmrExp.fb & !STATE_TX_ENABLED) then
  TxReg0 := PD0.pin;
  else when (STATE_TX_ENABLED & STATE_TX_ON_RISING & !Clk2 #
       STATE_TX_ENABLED & STATE_TX_ON_FALLING & Clk2) then
TxReg0 := RxReg0.fb;
  else
  TxReg0 := TxReg0.fb;

"*****************************************************************************
"* Receive Shift Register.
"* A single stage shift register used as a source for the Tx shift register.
"* In normal mode the input for the Rx shift register is the pda's DSDO, while
"* in diagnostic loopback mode, data is taken directly from the Tx shift
"* serial output.
"*
"* The output of the Rx shift register is fed to the input of the Tx shift
"* register. When transmission (and reception) is done the received data
"* word is compossed of the Rx shift register (LSB) concatenated with the
"* 7 LSBs of the Tx shift register.
"*
"* The edge (in Clk2 terms) upon which data is shifted in is determined by
"* TxClkSns as with the Tx shift register but on opposite edges, i.e.,
"* data is shifted Out from the Tx shift register on the Falling edge of
"* DSCK while is shifted In to the Rx shift register on the Rising edge
"* DSCK. (DSCK terms are constant in that regard).
"*****************************************************************************
equations

  RxReg0.clk = IClk;
  RxReg0.ar = Reset;

  when ( STATE_TX_ENABLED & STATE_TX_ON_RISING & Clk2 #
       STATE_TX_ENABLED & STATE_TX_ON_FALLING & !Clk2) & (!IN_DIAG_LOOP_BACK) then
  RxReg0.d = DSDO;   "shift in ext data
  else when ( STATE_TX_ENABLED & STATE_TX_ON_RISING & Clk2 #
           STATE_TX_ENABLED & STATE_TX_ON_FALLING & !Clk2) & IN_DIAG_LOOP_BACK then
RxReg0.d = TxReg7.fb;"  shift in from transmit reg
  else
  RxReg0.d = RxReg0.fb;" hold value


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

```
"* AdsReq.
"* Host from ads, read acknowledge. This state machine generates an automatic
"* ADS read request from the host when either a byte of data is received in the
"* Rx shift register or the status request bit in the control register is
"* active during a previous host write to the control register.
"* When the host detectes AdsReq asserted, it assertes HstAck in return. HstAck
"* double synchronized from the ADI port and delayed using the bundle delay
"* compensation timer to negate AdsReq. When the host detects AdsReq negated
"* it knows that data is valid to be read. After the host reads the data it
"* negates HstAck.
"* The machine steps through these states :
"* 0 - !ADS_REQ_ACTIVE
"* 1 - ADS_REQ_ACTIVE
"*******************************************************************************
equations
  AdsReq.clk  = IClk;
  AdsReq.ar   = Reset;
  AdsReq.oe   = ADS_IS_SELECTED;

  S_HstAck.clk = IClk;
  DS_HstAck.clk = IClk;

  S_HstAck := HstAck;
  DS_HstAck := HstAck & S_HstAck;"double synced

state_diagram AdsReq
  state !ADS_REQ_ACTIVE:
    if ( TxEn.fb & TxWordEnd   #" end of data shift to PDA
        ADS_SEND_STATUS) then" end of control write and status required
      ADS_REQ_ACTIVE
    else
      !ADS_REQ_ACTIVE;

  state ADS_REQ_ACTIVE:
    if ( HOST_READ_ADI & BndTmrExp.fb ) then
        !ADS_REQ_ACTIVE
    else
      ADS_REQ_ACTIVE;

"*******************************************************************************
"* ADI control register.
"* The ADI control register is written upon host to ADI write with a
"* D_C_B line is in control mode. It also may be read when StatusRequest_B bit
"* is active.
"*
"* Control register bits description:
"*
"* DebugEntry_B: (Bit 0). When this bit is active (L), the pda will enter debug
"* mode immediately after reset, i.e., DSCK will be held high
"* after the rising edge of SRESET*. When negated, DSCK will be
"* held low after the rising edge of SRESET so the pda will start
"* running instantly.
"* DiagLoopBack_B: (Bit 1). When active (L), the interface is in Diagnostic
"*Loopback mode. I.e., the source for the Rx shift register is
```

```
"* the output of the Tx shift register. During that mode, DSCK
"*and DSDI are tri-stated, so no arbitrary data is sent to the
"*debug port. When inactive, the interface is in normal mode,
"*i.e., DSCK and DSDI are driven and the source of the Rx shift
"*register is DSDO.
"* StatusRequest_B: (Bit 2). When active (L) any write to the control register
"*will be followed by a status read cycle initiated by the
"*debug port controller, i.e., AdsReq will be asserted after
"*the write cycle ends. When inactive, a write to the control
"* register will not be followed by a read from status register.
"* DbgClkDivSel(1:0) : (Bits 4,3). This field selects the division of the
"*             DbgClk input. Division factors are set as follows:
"* 0 - by 1
"* 1 - by 2
"* 2 - by 4
"*3 - by 8
"*
"* Important!!! All bits wake up active (L) after reset.
"*
"*******************************************************************************
equations
  AdiCtrlReg.clk = IClk;
  AdiCtrlReg.ar = Reset;"All active low.

  when ( HOST_WRITE_ADI_CONTROL & BndTmrExp.fb) then
    AdiCtrlReg.d = [PD4.pin, PD3.pin, PD2.pin, PD1.pin, PD0.pin];
  else
    AdiCtrlReg.d = AdiCtrlReg.fb;



"*******************************************************************************
"* ADI Data Bus.
"* The Adi data bus is driven towards the host when the host reads the i/f.
"* When D_C_B line is high (data) the Rx shift register contents is driven. If
"* D_C_B is low (control) the status register contents is driven.
"* The status register contains all control register's bits (4:0) with the
"* addition of the following:
"*
"* InDebugMode: (Bit 5). When this bit is active (H), the mpc is in debug mode,
"* i.e., either Freeze or VFLS(0:1) lines are driven high.
"*     When mpc is not in socket, VflsP(0:1) coming from the debug port
"*     are selected.
"* TxError: (Bit 6). When this bit is active (H), it signals that the pda was
"* reset (internally) during data transmission. (i.e., data received
"* during that trnasmission is corrupted). This bit is reset (L) when
"* either happens: (1) - The interface is reset by the host (both
"*AdsHardReset and AdsSoftReset are asserted (H) by the host
"* while the board is selected). (2) - The host writes the interface with
"* D_C_B signal low (control) and with data bit 6 high. (3) - a new data
"* word is written to the Tx shift register. (I.e., error is not kept
"*indefinitely).
"* PdaRst: (Bit 7). When this bit is active (H), it means that either SRESET*
"*or HRESET* or both are driven by the pda. The host have to wait until
"* this bit negates so that data may be wrriten to the debug port.
```

```
"*******************************************************************************
equations
  PDOe = DATA_BUFFERS_ENABLE ;

  PD.oe = PDOe;

  when ( READ_DATA_WORD_ON_ADI_BUS) then
     PD = RxReg.fb;
  elsewhen ( STATUS_WORD_ON_ADI_BUS ) then
 PD = [PdaRst.fb, TxError.fb, InDebugMode.fb, DbgClkDivSel1.fb, DbgClkDivSel0.fb,
 StatusRequest_B.fb,  DiagLoopBack_B.fb, DebugEntry_B.fb ];

"*******************************************************************************
"* Reset Status
"*******************************************************************************
  PdaRst.clk = IClk;

  PdaRst := (!PdaHardReset_B # !PdaSoftReset_B) &
      (AdsSelect_B.fb==BOARD_IS_SELECTED);   " synchronized inside.

"*******************************************************************************
"* In debug Mode
"*******************************************************************************
  InDebugMode.clk = IClk;

  when (FRZ_IS_SELECTED & CHIP_IS_IN_SOCKET) then
    InDebugMode := Freeze;
  else when (!FRZ_IS_SELECTED & CHIP_IS_IN_SOCKET) then
    InDebugMode := (VFLS0 & VFLS1);
  else when (!CHIP_IN_SOCKET) then
    InDebugMode := (VflsP0.pin & VflsP1.pin);


"*******************************************************************************
"* TxError.
"* This bit of the status register is set ('1') when the pda internally resets
"* during data transmission over the debug port.
"* When this bit is writen '1' by the adi port (control) the status bit is
"* cleared. Writing '0' has no influence on that bit.
"*******************************************************************************

equations

 TxError.clk = IClk;
 TxError.ar = Reset;

state_diagram TxError

state TX_DONE_OK:
if(STATE_TX_ENABLED & PdaRst.fb) then
TX_INTERRUPTED
else
TX_DONE_OK;
state TX_INTERRUPTED:
```

```
if (HOST_WRITE_ADI_CONTROL & BndTmrExp.fb & PD6.pin
   # HOST_WRITE_ADI_DATA & BndTmrExp.fb & !PdaRst.fb) then
TX_DONE_OK
else
TX_INTERRUPTED;

"*****************************************************************************
"* DSCK.
"* PDA debug port, gated serial clock.
"*****************************************************************************
equations
  DSCK.oe = ADS_IS_SELECTED;

  when ( ADS_IS_SELECTED & !PdaSoftReset_B ) then
    DSCK = H;"debug mode enable
  else when ( ADS_IS_SELECTED & !TxEn.fb & PdaSoftReset_B ) then
DSCK = !DebugEntry_B.fb;"debug mode direct entry
  else when (ADS_IS_SELECTED & TxEn.fb & STATE_TX_ON_RISING) then
      DSCK = !Clk2;"debug port clock
  else when (ADS_IS_SELECTED & TxEn.fb & STATE_TX_ON_FALLING) then
      DSCK = Clk2;"debug port inverted clock
  else when (!ADS_IS_SELECTED) then
  DSCK = H;"default value, infact X

"*****************************************************************************
"* DSDI.
"* Debug Port Serial Data in. (from Pda).
"* To provide better hold time for DSDI from the last rising edge of DSCK,
"* a dedicated enable for DSDI is provided - DSDI_ENABLE.
"*****************************************************************************
equations
  DsdiEn.ar = Reset;
  DsdiEn.clk = IClk;

state_diagram DsdiEn
state DSDI_DISABLED:
if(HOST_WRITE_ADI_DATA & BndTmrExp.fb & !PdaRst.fb) then
DSDI_ENABLED
else
DSDI_DISABLED;
state DSDI_ENABLED:
if(STATE_TX_DISABLED # PdaRst.fb) then
DSDI_DISABLED
else
DSDI_ENABLED;
equations
  DSDI.oe = ADS_IS_SELECTED & !IN_DIAG_LOOP_BACK; "avoid junk driven on DSDI input
          "during diagnostic loop back mode.
  when (ADS_IS_SELECTED & !PdaSoftReset_B) then
  DSDI = H;
  else when (ADS_IS_SELECTED & !STATE_DSDI_ENABLED & PdaSoftReset_B ) then
  DSDI = L;
  else when (ADS_IS_SELECTED & STATE_DSDI_ENABLED) then
  DSDI = TxReg7.fb;
```

```
  else
  DSDI = L;

"*****************************************************************************
"* Debug Port VFLS pins.
"*****************************************************************************

equations

  VflsP.oe = !ADS_IS_SELECTED;

  when (!FRZ_IS_SELECTED) then
    VflsP = [VFLS0,VFLS1];
  else when (FRZ_IS_SELECTED) then
    VflsP = [Freeze,Freeze];

"*****************************************************************************
"* Run Led
"*****************************************************************************
  Run.oe = H;
  !Run = IS_IN_DEBUG_MODE;"when 1 lits a led.
```

2•0•2 U36 - Board Control & Status Register

```
"***"*****************************************************************************************
"* In this file (A):
"* - 2 files were merged together: bcsr11.abl and brd_ctl13.abl
"* - First all the history of bcsr11 is described and then all the history of brd_ctl13.
"* BCSR basis file was bcsr11. Then, brd_ctl13 was added.
"*****************************************************************************************
"* History of bcsr11
"*************************************************************************
"* In this file (6):
"* - Added board revision # at BCSR3: 0 ENG
"*            1 - PILOT.
"* - Flash Presence detect lines - added FlashPD(7:5).
"* - Changed polarity of Power-On Reset (now active high)
"* - DramEn becomes active-low to enhance debug-station support changes.
"*************************************************************************
"* In this file (7):
"* - Board revisiob code @ BCSR3 is changed to 2 - Rev A.
"*************************************************************************
"* In this file (8):
"* - Board revisiob code @ BCSR3 is changed to 3 - Rev B.
"* - Added RS232En2~ for 2'nd RS232 port.
"*************************************************************************
"* In this file (9):
"* - All status bits (except CntRegEnProtect~) are removed for external buffers.
"* - Added address line A27
"* - Added BCSR2CS~ and BCSR3CS~ for external status registers.
"* - Added controls on bcsr1:
"*   - SdramEn~
"*   - PccVcc1~
"* - Added BCSR4 with following controls:
"*   - UsbFethEn~ (bit 4) enables Usb or Fast Ethernet ports
"*   - UsbSpeed         (bit 5) Usb speed control ('1'- full speed)
"*   - UsbVcc0(bit 6) enables VCC for Usb channel
"*   - UsbVcc1  (bit 7) reserved for possible 3.3V usb power
"*   - VideoOn~(bit 8) enables video transceiver
"*   - VideoExtClkEn (bit 9) enables ext 27Mhz clock for video encoder
"*   - VideoRst~(bit 10) resets the video encoder.
"*   - SignaLamp(bit 3) used for s/w signaling to user.
"*************************************************************************
"* In this file (10):
"* - Added ethernet transceiver control signals to BCSR4:
"*   - EthLoop          (bit 0) sets the transceiver to internal loopback (H)
"*   - TPFFLDL~         (bit 1) sets the trans. to full-duplex mode. (L)
"*   - TPSQEL~          (bit 2) allows for testing the colission ciruitry
"*                              of the 68160.
"*   - ModemEn~        (bit 11) enables the modem tool with the MPC823FADSDB
"*   - Modem_Audio~     (bit 12) selects between modem / audio function for
"*                               modem tool with MPC823 daughter board.
"*************************************************************************
"* In this file (11):
"* - Corrected bug with UsbVcc0 and UsbVcc1 - they were writen according to
"*   PccVcc0 and PccVcc1 data bits, instead of UsbVcc0 and UsbVcc1 data bits.
"*   For Eng 823DB it has no influence since the USB power is not operational
"*   there, but for rev Pilot and up 823DB it is important.
"*************************************************************************
"* History of brd_ctl13
"*************************************************************************
```

Support Information

```
"* In this file (5):
"* 1) The use of BCLOSE~ is removed. This due to the assignment
"*    BCLOSE~ to GPL4A. In order of using of GPL4A bit in the upm to determine
"*    data sampling edge, GPL4A may not be used as a GPL. Therefore DramBankXCs~
"*    must envelope the cycle so that data buffers remain open throughout the
"*    cycle.
"* 2) Removed CS support for flash configuration. I.e., FlashCs1~ will not be
"*    asserted during hard reset. Flash configuration will be supported on
"*    silicon next revisions.
"*    data buffers will still open for flash configuration when hard reset
"*    asserted and flash configuration option bit, asserted.
"* 3) Since Bclose~ is no longer available, the data buffers will open
"*    asynchronously. I.e., driven directly by the various chip-selects.
"*    to provide data hold (0) on write cycles to flash, CSNT bit in the OR
"*    should be programmed active, while ACS == 00.
"*****************************************************************************
"* In This file (6), A12 and A11 are removed from the flash selection equation
"* since they can select only a 1/2 Mbyte of flash rather then 2Mbyte selection
"* needed. Therefore, only one bank of 2 Mbyte flash may be used (MCM29F020).
"* The rest of the CS are driven high constantly.
"*****************************************************************************
"* In this file (7):
"* - Pon Reset Out is removed. Pon Reset is driven directly to MPC.
"* - Modck0 becomes Modck2
"* - A9 and A10 replace A11 and A12 in flash bank selection
"* - Optional BufClose~ is removed.
"* - DramEn becomes active-low to support debug-station support changes.
"* - Added F_PD(1:3) to support SMART Flash SIMMs.
"* - Support for 32KHz crystal - renewed.
"*****************************************************************************
"* In this file (8):
"* - Added protection against data contention for write cycles after
"*   Flash read cycle. This is achieved using a state-machine which identifies
"*   end of flash read and a chain of internal gates serving as a delay line.
"*   This kind of solution guaranties a fixed delay over the data buffer enable
"*   signal, that is, only after a flash read cycle.
"*****************************************************************************
"* In this file (9):
"* - The addressing scheme of the flash is changed so that the bank does
"*   not occupy a space bigger than its real size. I.e. A9 and A10 use
"*   is conditioned with the module type.
"*****************************************************************************
" In this file (10):
"* - A bug is fixed in Smart flash memories presence detect encoding:
"*   for SM732A1000A - F_PD == 5 (was 2)
"*   for SM732A2000  - F_PD == 4 (was 3)
"*****************************************************************************
"* In this file (11):
"* - KAPORIn (power-on reset input) line is removed. It was unused previously.
"* - Instead of the the above, F_PD4 input is added, so excact identification
"*   may be given to any flash memory.
"* - Number of delay stages for flash turn-off time protection is decreased
"*   This to avoid possible problem in write to 0-w.s. memories.
"*****************************************************************************
"* In This file (12):
"* - Added ATA support for PCMCIA. I.e., PccEvenEn~ and PccOddEn~ become
"*   identical, enabled by logic OR of CE1~ and CE2~
"*****************************************************************************
"* In this file (13):
```

```
"* - A bug is fixed in pcmcia data buffers enables, so that 1'st pcmcia
"*   access after flash read, is not defected anymore.
"*****************************************************************************



module bcsr11
title  'MPC8XXFADS Board Control and Status Register.
        Originated for MPC821ADS, Yair Liebman - (MSIL) - April 14, 1995
        Revised for MC8XXFDAS, Yair Liebman - (MSIL) January 21, 1997'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************
"U11 device 'mach220a';

"*****************************************************************************
"*****************************************************************************



"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************
SYSCLKPIN  124;
cs5  PIN 117 ;
cs6             PIN 115;
cs7             PIN 103; "added haim
RGPORInPIN  12;
DriveConfig~PIN  62;

BrdContRegCs~PIN  48;
TA~  PIN  53;
R_W~ PIN  54;

A27  PIN  125;
A28  PIN  126;
A29  PIN  11;

D0   PIN  26;
D1   PIN  110;
D2   PIN  30;
D3   PIN  100;
D4   PIN  116;
D5   PIN  112;
D6   PIN  32;
D7   PIN  16;
D8   PIN  5;
D9   PIN  104;
D10  PIN  18;
D11  PIN  98;
D12  PIN  28;
D13  PIN  102;
D14  PIN  22;
D15  PIN  3;

"*****************************************************************************
"* Board Control Pins. Read/Write.
```

```
"*****************************************************************************
FlashEn~PIN 40 istype 'reg,buffer';   " flash enable.
DramEn~PIN 70 istype 'reg,buffer';   " dram enable
EthEn~PIN 86 istype 'reg,buffer';   " ethernet port enable
InfRedEn~PIN 44 istype 'reg,buffer';   " infra-red port enable
FlashCfgEn~PIN 132 istype 'reg,buffer';   " flash configuration enable
CntRegEn~PIN 87 istype 'reg,buffer';   " control register access enable
RS232En1~PIN 143 istype 'reg,buffer';   " RS232 port 1 enable
PccEn~PIN 85 istype 'reg,buffer';   " PCMCIA port enable
PccVcc0PIN 140 istype 'reg,buffer';   " PCMCIA operation voltage select 0
PccVpp0PIN 130 istype 'reg,buffer';   " PCMCIA programming voltage select
PccVpp1PIN 78 istype 'reg,buffer';   " PCMCIA programming voltage select
HalfWord~PIN 77 istype 'reg,buffer';   " 32/16 bit dram operation select
RS232En2~PIN 68 istype 'reg,buffer';   " RS232 port 2 enable
SdramEn~PIN 138 istype 'reg,buffer';   " sdram enable
PccVcc1PIN 66 istype 'reg,buffer';   " PCMCIA operation voltage select 1

EthLoop         PIN 128 istype 'reg,buffer';   " 68160 internal loop back
TPFLDL~         PIN 80 istype 'reg,buffer';   " 68160 full-duplex
TPSQEL~         PIN 89 istype 'reg,buffer';   " 68160 colission circuitry test.
SignaLamp~PIN 82 istype 'reg,buffer';   " status lamp for misc s/w visual signaling
UsbFethEn~ PIN 38 istype 'reg,buffer';   " Usb or Fast ethernet port enable
UsbSpeedPIN 76 istype 'reg,buffer';   " Usb speed control
UsbVcc0PIN 144 istype 'reg,buffer';   " Usb VCC select 0 line
UsbVcc1PIN 91 istype 'reg,buffer';   " Usb VCC select 1 line
VideoOn~PIN 79 istype 'reg,buffer';   " Video encoder enable
VideoExtClkEn PIN 142 istype 'reg,buffer';   " Enable external clock gen for video encoder
VideoRst~PIN 81 istype 'reg,buffer';   " Video Encoder reset
ModemEn~        PIN 75 istype 'reg,buffer';   " modem tool enable for MPC823FADSDB
Modem_Audio~    PIN 134 istype 'reg,buffer';   " Modem / Audio functions select
                                               " for modem tool with MPC823 d/b.


"*****************************************************************************
"* Board Status Pins. Read only.
"*****************************************************************************
"* removed to external buffers

"*****************************************************************************
"* Board Status Registers Chip-Selects
"*****************************************************************************
Bcsr2Cs~PIN 105 istype 'com';
Bcsr3Cs~PIN 99 istype 'com';
"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************

"*****************************************************************************
"* Addition Pins from brd_ctl13.
"*****************************************************************************

"*****************************************************************************
"* Flash Associated Pins.
"*****************************************************************************

F_PD1 PIN 57;
F_PD2 PIN 55;
F_PD3 PIN 41;
F_PD4 PIN 29;
```

```
FlashCs~ PIN 46;" flash bank chip-select

FlashCs1~PIN 42 istype 'com';" Flash bank1 chip-select
FlashCs2~PIN 39 istype 'com';" Flash bank2 chip-select
FlashCs3~PIN 43 istype 'com';" Flash bank3 chip-select
FlashCs4~PIN 45 istype 'com';" Flash bank4 chip-select

FlashOe~PIN 31 istype 'com';" Flash output enable.

"*****************************************************************************
"* Dram Associated Pins.
"*****************************************************************************
A9   PIN 21;
A10  PIN 19;
A19  PIN 10;
A20  PIN 9;
A30  PIN 8;" pda address lines inputs (IN)

SizeDetect1PIN 7;
SizeDetect0PIN 15; " dram simm size detect lines (IN)

DramBank1Cs~PIN 97;" 1'st bank chip-select(IN, L)
DramBank2Cs~PIN 94;" 2'nd bank chip-select (IN, L)

DramAdd10PIN 65 istype 'com';
DramAdd9PIN 71 istype 'com';" dram address lines

Ras1~PIN 92 istype 'com';
Ras1DD~PIN 61 istype 'com';
Ras2~PIN 58 istype 'com';
Ras2DD~PIN 56 istype 'com';" dram RAS lines.

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

Rst0           PIN 6;                " connected to N.C. of  Reset P.B.
Rst1           PIN 4;                " connected to N.O. of  Reset P.B.

HardReset~PIN 72 istype 'com';    " Actual hard reset output (O.D.)
SoftReset~PIN 23 istype 'com';    " Actual soft reset output (O.D.)
ResetConfig~PIN 33 istype 'com';    " Drives the RSTCONF* signal of the pda.

Abr0 PIN 120;     " connected to N.C. of  Abort P.B.
Abr1 PIN 119;     " connected to N.O. of  Abort P.B.

NMIEnNODE istype 'com';   " enables T.S. NMI pin
NMI~ PIN 17 istype 'com';    " Actual NMI pin (O.D.)

"*****************************************************************************
"* Power On Reset Configuration Support
"*****************************************************************************
ModInPIN 93;" MODCK dip-switch
Modck2PIN 27 istype 'com';" MODCK2 output
Modck1PIN 137 istype 'com';" MODCK1 output

ModckOeNODE istype 'com';" enables MODCKs towards PDA during
        " Hard Reset.
```

```
"*****************************************************************************
"* Data Buffers Enables and Reset configuration support
"*****************************************************************************
TEA~ PIN 111;     " Transfer Error Acknowledge.

PccCE1~PIN 47;
PccCE2~PIN 118;


UpperHalfEn~PIN 129 istype 'com,invert';    " bits 0:15 data buffer enable
LowerHalfEn~PIN 60 istype 'com,invert';    " bits 16:31 data buffer enable

PccEvenEn~PIN 127 istype 'com,invert';    " pcc upper byte data buffer enable
PccOddEn~PIN 133 istype 'com,invert';    " pcc lower byte data buffer enable


PccR_W~PIN 20 istype 'com';" pcmcia data buffers direction


"*****************************************************************************
"*****************************************************************************
"* System Hard Reset Configuration.
"*****************************************************************************


ERBNODE istype 'reg,buffer';   " External Arbitration
IP~NODE istype 'reg,buffer';   " Interrupt Prefix in MSR
BDISNODE istype 'reg,buffer';   " Boot Disable
RSV2NODE istype 'reg,buffer';   " reserved config bit 2
BPS0,
BPS1NODE istype 'reg,buffer';   " Boot Port Size
RSV6NODE istype 'reg,buffer';   " reserved config bit 6
ISB0,
ISB1NODE istype 'reg,buffer';   " Internal Space Base
DBGC0,
DBGC1NODE istype 'reg,buffer';   " Debug pins Config.
DBPC0,
DBPC1NODE istype 'reg,buffer';   " Debug Port pins Config
RSV13   NODE istype 'reg,buffer';   " reserved config bit 13
RSV14   NODE istype 'reg,buffer';   " reserved config bit 14
RSV15   NODE istype 'reg,buffer';   " reserved config bit 15

DataOeNODE istype 'com';" data bus output enable on read.

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

CntRegEnProtect~NODE istype 'reg,buffer';

"*****************************************************************************
"* Control Register Write (space saving) Mach 10 required for 52Mhz
"*****************************************************************************
Bcsr0Write~ NODE istype 'com';
Bcsr1Write~ NODE istype 'com';
Bcsr4Write~ NODE istype 'com';

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

```
"* Addition from brd_ctl13
"*****************************************************************************
"*****************************************************************************
"* Reset & Interrupt Logic Pins.
"*****************************************************************************
RstDeb1NODE istype 'com';    " reset push button debouncer
AbrDeb1NODE istype 'com';    " abort push button debouncer

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

ConfigHold2,
ConfigHold1,
ConfigHold0node istype 'reg,buffer';" supplies data hold time for
          " hard reset configuration

ConfigHoldEndnode istype 'com';

"*****************************************************************************
"* data buffers enable.
"*****************************************************************************
SyncHardReset~NODE istype 'reg,buffer';" synchronized hard reset
DSyncHardReset~ NODE istype 'reg,buffer';" double synchronized hard reset
SyncTEA~NODE istype 'reg,buffer';" needed since TEA~ is O.D.

HoldOffConsidered NODE istype 'reg,buffer';" data drive hold-off state
         " machine.
D_FlashOe~NODE istype 'com';" delayed flash output enable
DD_FlashOe~NODE istype 'com';" double delayed flash output
         " enable
TD_FlashOe~NODE istype 'com';" triple delayed flash output
         " enable
" QD_FlashOe~NODE istype 'com';  quad delayed
" PD_FlashOe~NODE istype 'com';  penta delayed

KeepPinsConnected node istype 'com';


"*****************************************************************************
"*****************************************************************************
H, L, X, Z = 1, 0, .X., .Z.;
C, D, U    = .C., .D., .U.;
"*****************************************************************************
"* SIMULATION = 1;
"* SLOW_PLL_LOCK = 1;
"* DRAM_8_BIT_OPERATION = 1;
"*****************************************************************************
"* Signal groups
"*****************************************************************************
Add = [A27,A29];
Data = [D0..D15];


ConfigReg = [ERB,IP~,RSV2,BDIS,
             BPS0,BPS1,RSV6,ISB0,
             ISB1,DBGC0,DBGC1,DBPC0,
       DBPC1,RSV13,RSV14,RSV15];

BPS = [BPS0,BPS1];" boot port size
```

```
ISB = [ISB0,ISB1];" Initial Internal Space Base
DBGC = [DBGC0,DBGC1];" Debug Pins Configuration
DBPC = [DBPC0,DBPC1];" Debug port location

ContReg = [FlashEn~,
           DramEn~,EthEn~,InfRedEn~,FlashCfgEn~,
           CntRegEnProtect~,CntRegEn~,RS232En1~,PccEn~,
           PccVcc0,PccVpp0,PccVpp1,HalfWord~,
           RS232En2~,SdramEn~,PccVcc1,EthLoop,
           TPFLDL~,TPSQEL~,SignaLamp~,UsbFethEn~,
           UsbSpeed,UsbVcc0,UsbVcc1,VideoOn~,
           VideoExtClkEn,VideoRst~,ModemEn~,Modem_Audio~];

ReadBcsr1 = [FlashEn~,DramEn~,EthEn~,InfRedEn~,
       FlashCfgEn~,CntRegEnProtect~.fb,CntRegEn~,RS232En1~,
       PccEn~,PccVcc0,PccVpp0,PccVpp1,
       HalfWord~,RS232En2~,SdramEn~,PccVcc1];

ReadBcsr4 = [EthLoop,
             TPFLDL~,TPSQEL~,SignaLamp~,UsbFethEn~,
             UsbSpeed,UsbVcc0, UsbVcc1 ,VideoOn~,
             VideoExtClkEn,VideoRst~,ModemEn~,Modem_Audio~];

DrivenContReg = [FlashEn~,DramEn~,EthEn~,InfRedEn~,
           FlashCfgEn~,CntRegEn~,RS232En1~,PccEn~,
                PccVcc0,PccVpp0,PccVpp1,HalfWord~,
      RS232En2~,SdramEn~,PccVcc1,EthLoop,
      TPFLDL~,TPSQEL~,SignaLamp~,UsbFethEn~,
      UsbSpeed,UsbVcc0,UsbVcc1,VideoOn~,
      VideoExtClkEn,VideoRst~,ModemEn~,Modem_Audio~];

PccVcc = [PccVcc0,PccVcc1];
PccVpp = [PccVpp0,PccVpp1];

WideContReg = [FlashEn~,
               DramEn~,EthEn~,InfRedEn~,FlashCfgEn~,
         CntRegEnProtect~,CntRegEn~,RS232En1~,PccEn~,
         PccVcc0,PccVpp0,PccVpp1,HalfWord~,
         RS232En2~,SdramEn~,PccVcc1,EthLoop,
         TPFLDL~,TPSQEL~,SignaLamp~,UsbFethEn~,
         UsbSpeed,UsbVcc0,UsbVcc1,VideoOn~,
         VideoExtClkEn,VideoRst~,ModemEn~,Modem_Audio~];

Bcsr2_3Cs~ = [Bcsr2Cs~,Bcsr3Cs~];

"*************************************************************************
"* Addition Signal groups from brd_ctl13
"*************************************************************************
F_PD = [F_PD4, F_PD3, F_PD2, F_PD1];
FlashCsOut = [FlashCs4~,FlashCs3~,FlashCs2~,FlashCs1~];
PdaAdd = [A9,A10,A19,A20,A30];
DramAdd = [DramAdd10,DramAdd9];
DramCS~ = [DramBank2Cs~,DramBank1Cs~];
Cs = [BrdContRegCs~,FlashCs~ ,DramBank1Cs~,DramBank2Cs~];
RAS = [Ras1~,Ras1DD~,Ras2~,Ras2DD~];
SD = [SizeDetect1,SizeDetect0];
Reset = [HardReset~,SoftReset~];
ResetEn = [HardResetEn,SoftResetEn];
Rst = [Rst1,Rst0];
```

```
Abr = [Abr1,Abr0];
Debounce = [RstDeb1,AbrDeb1];
RstCause = [Rst1,Rst0,Abr1,Abr0,!RGPORIn];
ConfigHold =[ConfigHold2, ConfigHold1, ConfigHold0];
SyncReset = [SyncHardReset~,DSyncHardReset~];
Modck = [Modck2, Modck1];
PccCs = [PccCE1~,PccCE2~];
LocDataBufEn = [UpperHalfEn~,LowerHalfEn~];
PccDataBufEn = [PccEvenEn~,PccOddEn~];
ModuleEn = [DramEn~,FlashEn~,PccEn~,CntRegEn~];
Stp = [TA~];


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

K_A_PON_RESET_ACTIVE = 1;

RESET_CONFIG_ACTIVE = 0;

"**** changed due to long lock delay of the pda *** 17,7,95 ******************

@ifndef SLOW_PLL_LOCK {

  KA_PON_RESET = (RGPORIn == K_A_PON_RESET_ACTIVE);
}

@ifdef SLOW_PLL_LOCK {
  PON_DEFAULT_ACTIVE = 0;

  KA_PON_RESET = (PonDefault~ == PON_DEFAULT_ACTIVE);
}

"*********** end of change ***************


RESET_CONFIG_DRIVEN = ((DriveConfig~ == RESET_CONFIG_ACTIVE) &
        (FlashCfgEn~ != FLASH_CFG_ENABLE));

"*****************************************************************************
"* Register Access definitions
"*****************************************************************************
CONFIG_REG_ADD = 0;
CONTROL_REG_1_ADD = 1;
STATUS_REG2_ADD = 2;
STATUS_REG3_ADD = 3;
CONTROL_REG_4_ADD = 4;

" MPC_WRITE_BCSR_0 = (!BrdContRegCs~ & !TA~ & !R_W~ & !A27 & !A28 & !A29 & !CntRegEn~);
" MPC_WRITE_BCSR_1 = (!BrdContRegCs~ & !TA~ & !R_W~ & !A27 & !A28 & A29 & !CntRegEn~);
MPC_WRITE_BCSR_3 = (!BrdContRegCs~ & !TA~ & !R_W~ & !A27 & A28 & A29 & !CntRegEn~);
"MPC_WRITE_BCSR_4 = (!BrdContRegCs~ & !TA~ & !R_W~ & A27 & !A28 & !A29 & !CntRegEn~);

BCSR_WRITE_ACTIVE = 0;

MPC_WRITE_BCSR_0 = (Bcsr0Write~.fb == BCSR_WRITE_ACTIVE);
MPC_WRITE_BCSR_1 = (Bcsr1Write~.fb == BCSR_WRITE_ACTIVE);
```

```
MPC_WRITE_BCSR_4 = (Bcsr4Write~.fb == BCSR_WRITE_ACTIVE);



MPC_READ = (!BrdContRegCs~ & R_W~ & !CntRegEn~);

MPC_READ_BCSR_0 = (!BrdContRegCs~ & R_W~ & !A27 & !A28 & !A29 & !CntRegEn~);
MPC_READ_BCSR_1 = (!BrdContRegCs~ & R_W~ & !A27 & !A28 & A29 & !CntRegEn~);
MPC_READ_BCSR_2 = (!BrdContRegCs~ & R_W~ & !A27 & A28 & !A29 & !CntRegEn~);
MPC_READ_BCSR_3 = (!BrdContRegCs~ & R_W~ & !A27 & A28 & A29 & !CntRegEn~);
MPC_READ_BCSR_4 = (!BrdContRegCs~ & R_W~ &  A27 & !A28 & !A29 & !CntRegEn~);


"*****************************************************************************
"*****************************************************************************
"* BCSR 0 definitions
"*****************************************************************************
"*****************************************************************************
INTERNAL_ARBITRATION = 0;
EXTERNAL_ARBITRATION = !INTERNAL_ARBITRATION;

IP_AT_0xFFF00000 = 0;"active low
IP_AT_0x00000000 = !IP_AT_0xFFF00000;

RSV2_ACTIVE = 1;

BOOT_DISABLE = 1;
BOOT_ENABLE = !BOOT_DISABLE;

BOOT_PORT_32 = 0;
BOOT_PORT_8 = 1;
BOOT_PORT_16 = 2;
BOOT_PORT_RESERVED = 3;

RSV6_ACTIVE = 1;

INT_SPACE_BASE_0x00000000 = 0;
INT_SPACE_BASE_0x00F00000 = 1;
INT_SPACE_BASE_0xFF000000 = 2;
INT_SPACE_BASE_0xFFF00000 = 3;

DEBUG_PINS_PCMCIA_2 = 0;
DEBUG_PINS_WATCH_POINTS = 1;
DEBUG_PINS_RESREVED = 2;
DEBUG_PINS_FOR_SHOW = 3;

DEBUG_PORT_ON_JTAG = 0;
DEBUG_PORT_NON_EXISTANT = 1;
DEBUG_PORT_RESERVED = 2;
DEBUG_PORT_ON_DEBUG_PINS = 3;

RSV13_ACTIVE = 1;
RSV14_ACTIVE = 1;
RSV15_ACTIVE = 1;



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

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


ERB_PON_DEFAULT = INTERNAL_ARBITRATION;

IP~_PON_DEFAULT = IP_AT_0x00000000;

RSV2_PON_DEFAULT = !RSV2_ACTIVE;

BDIS_PON_DEFAULT = BOOT_ENABLE;

BPS_PON_DEFAULT = BOOT_PORT_32;

RSV6_PON_DEFAULT = !RSV6_ACTIVE;

ISB_PON_DEFAULT = INT_SPACE_BASE_0xFF000000;

DBGC_PON_DEFAULT = DEBUG_PINS_PCMCIA_2;

DBPC_PON_DEFAULT = DEBUG_PORT_ON_JTAG;

RSV13_PON_DEFAULT = !RSV13_ACTIVE;

RSV14_PON_DEFAULT = !RSV14_ACTIVE;

RSV15_PON_DEFAULT = !RSV15_ACTIVE;

"*****************************************
"******* Data Bits Assignments *************
"*****************************************

ERB_DATA_BIT = [D0];
IP~_DATA_BIT = [D1];
RSV2_DATA_BIT = [D2];
BDIS_DATA_BIT = [D3];
BPS_DATA_BIT = [D4,D5];
RSV6_DATA_BIT = [D6];
ISB_DATA_BIT = [D7,D8];
DBGC_DATA_BIT = [D9,D10];
DBPC_DATA_BIT = [D11,D12];
RSV13_DATA_BIT = [D13];
RSV14_DATA_BIT = [D14];
RSV15_DATA_BIT = [D15];

"*****************************************************************************
"*****************************************************************************
"* BCSR 1 definitions.
"*****************************************************************************
"*****************************************************************************
HALF_WORD = 0;

ETH_ENABLED = 0;

DRAM_ENABLED = 0;

CONT_REG_ENABLE = 0;

RS232_1_ENABLE = 0;
```

```
RS232_2_ENABLE = 0;

PCC_ENABLE = 0;

PCC_VCC_CONT_0 = 0;
PCC_VCC_CONT_1 = 1;

" PCC_VPP_0 = 0;
" PCC_VPP_12 = 2;
" PCC_VPP_5 = 1;
" PCC_VPP_TS = 3;

PCC_VPP0 = 1;
PCC_VPP1 = 1;

FLASH_ENABLED = 0;

INF_RED_ENABLE = 0;

"* FLASH_CFG_ENABLE = 0;  needed to be defined ealier

DRAM_5V = 0;
DRAM_3V = !DRAM_5V;

CNT_REG_EN_PROTECT = 0; " inadvertant write protect

SDRAM_ENABLED = 1;


"*********************************************
"******* Power On Defaults Assignments *********
"*********************************************
FLASH_ENABLE_PON_DEFAULT = FLASH_ENABLED;
FLASH_CFG_ENABLE_PON_DEFAULT = !FLASH_CFG_ENABLE;

DRAM_ENABLE_PON_DEFAULT = DRAM_ENABLED;

ETH_ENABLE_PON_DEFAULT = !ETH_ENABLED;

CONT_REG_ENABLE_PON_DEFAULT = CONT_REG_ENABLE;

RS232_1_ENABLE_PON_DEFAULT = !RS232_1_ENABLE;
RS232_2_ENABLE_PON_DEFAULT = !RS232_2_ENABLE;

PCC_ENABLE_PON_DEFAULT = !PCC_ENABLE;

PCC_VCC_0_PON_DEFAULT = PCC_VCC_CONT_0;
PCC_VCC_1_PON_DEFAULT = PCC_VCC_CONT_0;

PCC_VPP0_PON_DEFAULT = PCC_VPP0;
PCC_VPP1_PON_DEFAULT = PCC_VPP1;    " T.S. as default

INF_RED_ENABLE_PON_DEFAULT = !INF_RED_ENABLE;

HALF_WORD_PON_DEFAULT = !HALF_WORD;

SDRAM_ENABLE_PON_DEFAULT = SDRAM_ENABLED;
```

```
CNT_REG_EN_PROTECT_PON_DEFAULT = CNT_REG_EN_PROTECT;

"*****************************************
"******* Data Bits Assignments *************
"*****************************************
FLASH_ENABLE_DATA_BIT = [D0];
DRAM_ENABLE_DATA_BIT = [D1];
ETH_ENABLE_DATA_BIT = [D2];
INF_RED_ENABLE_DATA_BIT = [D3];
FLASH_CFG_ENABLE_DATA_BIT = [D4];
CNT_REG_EN_PROTECT_DATA_BIT = [D5];
CONT_REG_ENABLE_DATA_BIT = [D6];
RS232_1_ENABLE_DATA_BIT = [D7];
PCC_ENABLE_DATA_BIT = [D8];
PCC_VCC_0_DATA_BIT = [D9];
PCC_VPP0_DATA_BIT = [D10];
PCC_VPP1_DATA_BIT = [D11];
HALF_WORD_DATA_BIT = [D12];
RS232_2_ENABLE_DATA_BIT = [D13];
SDRAM_ENABLE_DATA_BIT = [D14];
PCC_VCC_1_DATA_BIT = [D15];

"*****************************************************************************
"*****************************************************************************
"* BCSR 4 definitions.
"*****************************************************************************
"*****************************************************************************
ETH_LOOP = 1;

ETH_FULL_DUP = 0;

ETH_CLSN_TEST = 0;

SIGNAL_LAMP_ON = 0;

USB_FETH_ENABLED = 0;

USB_FULL_SPEED = 1;

USB_VCC_CONT_0 = 0;

VIDEO_ENABLED = 0;

VIDEO_EXT_CLK_ENABLED = 1;

VIDEO_RESET_ACTIVE = 0;

MODEM_ENABLED_FOR_823 = 0;

MODEM = 1;

"*********************************************
"******* Power On Defaults Assignments *********
"*********************************************
ETH_LOOP_PON_DEFAULT = !ETH_LOOP;
ETH_FULL_DUP_PON_DEFAULT = !ETH_FULL_DUP;
ETH_CLSN_TEST_PON_DEFAULT = !ETH_CLSN_TEST;
SIGNAL_LAMP_PON_DEFAULT = !SIGNAL_LAMP_ON;
USB_FETH_EN_PON_DEFAULT = !USB_FETH_ENABLED;
```

```
USB_SPEED_PON_DEFAULT = USB_FULL_SPEED;
USB_VCC_0_CONT_PON_DEFAULT = !USB_VCC_CONT_0;
USB_VCC_1_CONT_PON_DEFAULT = !USB_VCC_CONT_0;
VIDEO_ENABLE_PON_DEFAULT = !VIDEO_ENABLED;
VIDEO_EXT_CLK_EN_PON_DEFAULT = !VIDEO_EXT_CLK_ENABLED;
VIDEO_RESET_PON_DEFAULT = !VIDEO_RESET_ACTIVE;
MODEM_ENABLE_PON_DEFAULT = !MODEM_ENABLED_FOR_823;
MODEM_FUNC_SEL_PON_DEFAULT = MODEM;


"*****************************************
"******* Data Bits Assignments *************
"*****************************************

ETH_LOOP_DATA_BIT = [D0];
ETH_FULL_DUP_DATA_BIT = [D1];
ETH_CLSN_TEST_DATA_BIT = [D2];
SIGNAL_LAMP_DATA_BIT = [D3];
USB_FETH_EN_DATA_BIT = [D4];
USB_SPEED_DATA_BIT = [D5];
USB_VCC_0_DATA_BIT = [D6];
USB_VCC_1_DATA_BIT = [D7];
VIDEO_ENABLE_DATA_BIT = [D8];
VIDEO_EXT_CLK_EN_DATA_BIT = [D9];
VIDEO_RESET_DATA_BIT = [D10];
MODEM_ENABLE_DATA_BIT = [D11];
MODEM_FUNC_SEL_DATA_BIT = [D12];

"*********************************************************
"******* Addition declarations from brd_ctl13 *************
"*********************************************************

"***************************************
"******* Flash declarations *************
"***************************************

FLASH_ENABLE_ACTIVE = 0;

FLASH_ACTIVE = (FlashEn~ == FLASH_ENABLE_ACTIVE);

MCM29020 = (F_PD == 8);
MCM29040 = (F_PD == 7);
MCM29080 = (F_PD == 6);
SM732A1000A = (F_PD == 5);
SM732A2000 = (F_PD == 4);

FLASH_BANK1 = ( (MCM29020 # SM732A1000A) #
                (MCM29040 & !A10) #
                (MCM29080 & !A9 & !A10) #
                (SM732A2000 & !A9) );

FLASH_BANK2 = ( (MCM29040 & A10) #
                (MCM29080 & !A9 & A10) #
                (SM732A2000 & A9) );

FLASH_BANK3 = (A9 & !A10 & MCM29080);

FLASH_BANK4 = (A9 & A10 & MCM29080);
```

```
"***************************************
"******* DRAM declarations *************
"***************************************

DRAM_ENABLE_ACTIVE = 0;

DRAM_ACTIVE = (DramEn~ == DRAM_ENABLE_ACTIVE);

SIMM36100 = (SD == 0);
SIMM36200 = (SD == 3);
SIMM36400 = (SD == 2);
SIMM36800 = (SD == 1);

IS_HALF_WORD = (HalfWord~ == 0);

"*****************************************************************************
"* Reset Declarations.
"*****************************************************************************
KEEP_ALIVE_PON_RESET_ACTIVE = 0;
REGULAR_PON_RESET_ACTIVE = 0;
HARD_RESET_ACTIVE = 0;
SOFT_RESET_ACTIVE = 0;

HARD_CONFIG_HOLD_VALUE = 4;


DRIVE_MODCK_TO_PDA = (HardReset~ == HARD_RESET_ACTIVE);" have modck stable
        " during hard reset.

REGULAR_POWER_ON_RESET  = (!RGPORIn == REGULAR_PON_RESET_ACTIVE); "add haim

HARD_RESET_ASSERTED = (SyncHardReset~.fb == HARD_RESET_ACTIVE);

HARD_RESET_NEGATES = ( (SyncHardReset~.fb != HARD_RESET_ACTIVE )
        & (DSyncHardReset~.fb == HARD_RESET_ACTIVE));
        " detecting hard reset negation

"*****************************************************************************
"* data buffers enable.
"*****************************************************************************

BUFFER_DISABLED = 1;
BUFFER_ENABLED = !BUFFER_DISABLED;

CONTROL_REG_ENABLE_ACTIVE = 0;
FLASH_CONFIG_ENABLED_ACTIVE = 0;
PCMCIA_ENABLE_ACTIVE = 0;

GPL_ACTIVE = 0;

TEA_ASSERTS = (!TEA~ & SyncTEA~.fb);" first clock of TEA~ asserted

CONTROL_REG_ENABLED = (CntRegEn~ == CONTROL_REG_ENABLE_ACTIVE);

FLASH_CONFIGURATION_ENABLED = (FlashCfgEn~ == FLASH_CONFIG_ENABLED_ACTIVE);

PCC_ENABLED = (PccEn~ == PCMCIA_ENABLE_ACTIVE);

NO_HOLD_OFF = 0;
```

```
HOLD_OFF_CONSIDERED = 1;

STATE_HOLD_OFF_CONSIDERED = (HoldOffConsidered.fb == HOLD_OFF_CONSIDERED);
STATE_NO_HOLD_OFF = (HoldOffConsidered.fb == NO_HOLD_OFF);

END_OF_FLASH_READ = !TA~ & !FlashCs~ & R_W~;" end of flash read cycle.
END_OF_OTHER_CYCLE = (!TA~ & FlashCs~ # " another access or
         !TA~ & !FlashCs~ & !R_W~); " flash write


"* HOLD_OFF_PERIOD = (!R_W~ & !PD_FlashOe~.fb);
HOLD_OFF_PERIOD = (!R_W~ & !TD_FlashOe~.fb);



"*****************************************************************************
"* Equations, state diagrams.                                                *
"*****************************************************************************
"*                                                                           *
"* #######                                                                   *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #        #    #  #    #   #  #      #       #     #    #  ##   #  #        *
"* #####    #    #  #    #  #    #     #       #     #    #  # #  #   ####     *
"* #        #  # #  #    #  ######     #       #     #    #  #  # #       #    *
"* #        #   #   #    #  #    #     #       #     #    #  #   ##  #    #    *
"* #######   ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                           *
"*****************************************************************************
"* Configuration Register.
"* Gets its default pon reset values which are driven to the data bus when
"* during hard reset configuration.
"* If other values are required, this register may be written with new values
"* to become active for the next hard reset.
"* The state machines are built in a way that its power on value is changed in
"* one place - the declarations area.
"*****************************************************************************

equations

@ifdef SLOW_PLL_LOCK {

  !PonDefault~ = !DriveConfig~ #
         RGPORIn;

}

!Bcsr0Write~ = (!BrdContRegCs~ & !TA~ & !R_W~ & !A27 & !A28 & !A29 & !CntRegEn~);
!Bcsr1Write~ = (!BrdContRegCs~ & !TA~ & !R_W~ & !A27 & !A28 & A29 & !CntRegEn~);
!Bcsr4Write~ = (!BrdContRegCs~ & !TA~ & !R_W~ & A27 & !A28 & !A29 & !CntRegEn~);


ConfigReg.clk = SYSCLK;

state_diagram ERB
  state INTERNAL_ARBITRATION:
     if (MPC_WRITE_BCSR_0 &
         (ERB_DATA_BIT.pin == EXTERNAL_ARBITRATION) &
          (!KA_PON_RESET # (ERB_PON_DEFAULT != INTERNAL_ARBITRATION)) #
          (KA_PON_RESET & (ERB_PON_DEFAULT == EXTERNAL_ARBITRATION)) ) then
```

```
        EXTERNAL_ARBITRATION
    else
        INTERNAL_ARBITRATION;
  state EXTERNAL_ARBITRATION:
    if (MPC_WRITE_BCSR_0 &
        (ERB_DATA_BIT.pin == INTERNAL_ARBITRATION) &
         (!KA_PON_RESET # (ERB_PON_DEFAULT != EXTERNAL_ARBITRATION)) #
         (KA_PON_RESET & (ERB_PON_DEFAULT == INTERNAL_ARBITRATION)) ) then
        INTERNAL_ARBITRATION
    else
        EXTERNAL_ARBITRATION;
"*****************************************************************************
state_diagram IP~
  state IP_AT_0xFFF00000:
    if (MPC_WRITE_BCSR_0 &
        (IP~_DATA_BIT.pin == IP_AT_0x00000000) &
         (!KA_PON_RESET # (IP~_PON_DEFAULT != IP_AT_0xFFF00000)) #
         (KA_PON_RESET & (IP~_PON_DEFAULT == IP_AT_0x00000000)) ) then
        IP_AT_0x00000000
    else
        IP_AT_0xFFF00000;
  state IP_AT_0x00000000:
    if (MPC_WRITE_BCSR_0 &
        (IP~_DATA_BIT.pin == IP_AT_0xFFF00000) &
         (!KA_PON_RESET # (IP~_PON_DEFAULT != IP_AT_0x00000000)) #
         (KA_PON_RESET & (IP~_PON_DEFAULT == IP_AT_0xFFF00000)) ) then
        IP_AT_0xFFF00000
    else
        IP_AT_0x00000000;
"*****************************************************************************
state_diagram RSV2
  state !RSV2_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV2_DATA_BIT.pin == RSV2_ACTIVE) &
         (!KA_PON_RESET # (RSV2_PON_DEFAULT != !RSV2_ACTIVE)) #
         (KA_PON_RESET & (RSV2_PON_DEFAULT == RSV2_ACTIVE)) ) then
        RSV2_ACTIVE
    else
        !RSV2_ACTIVE;
  state RSV2_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV2_DATA_BIT.pin == !RSV2_ACTIVE) &
         (!KA_PON_RESET # (RSV2_PON_DEFAULT != RSV2_ACTIVE)) #
         (KA_PON_RESET & (RSV2_PON_DEFAULT == !RSV2_ACTIVE)) ) then
        !RSV2_ACTIVE
    else
        RSV2_ACTIVE;
"*****************************************************************************
state_diagram BDIS
  state BOOT_ENABLE:
    if (MPC_WRITE_BCSR_0 &
        (BDIS_DATA_BIT.pin == BOOT_DISABLE) &
         (!KA_PON_RESET # (BDIS_PON_DEFAULT != BOOT_ENABLE)) #
         (KA_PON_RESET & (BDIS_PON_DEFAULT == BOOT_DISABLE)) ) then
        BOOT_DISABLE
    else
        BOOT_ENABLE;
  state BOOT_DISABLE:
    if (MPC_WRITE_BCSR_0 &
```

```
        (BDIS_DATA_BIT.pin == BOOT_ENABLE) &
         (!KA_PON_RESET # (BDIS_PON_DEFAULT != BOOT_DISABLE)) #
         (KA_PON_RESET & (BDIS_PON_DEFAULT == BOOT_ENABLE)) ) then
        BOOT_ENABLE
    else
        BOOT_DISABLE;
"*****************************************************************************
state_diagram BPS
  state BOOT_PORT_32:
    if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_8) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_32)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_8)) ) then
        BOOT_PORT_8
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_16) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_32)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_16)) ) then
        BOOT_PORT_16
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_RESERVED) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_32)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_RESERVED)) ) then
        BOOT_PORT_RESERVED
    else
        BOOT_PORT_32;
  state BOOT_PORT_8:
    if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_32) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_8)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_32)) ) then
        BOOT_PORT_32
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_16) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_8)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_16)) ) then
        BOOT_PORT_16
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_RESERVED) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_8)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_RESERVED)) ) then
        BOOT_PORT_RESERVED
    else
        BOOT_PORT_8;
  state BOOT_PORT_16:
    if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_32) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_16)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_32)) ) then
        BOOT_PORT_32
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_8) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_16)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_8)) ) then
        BOOT_PORT_8
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_RESERVED) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_16)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_RESERVED)) ) then
```

```
        BOOT_PORT_RESERVED
    else
        BOOT_PORT_16;
  state BOOT_PORT_RESERVED:
    if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_32) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_RESERVED)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_32)) ) then
        BOOT_PORT_32
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_16) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_RESERVED)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_16)) ) then
        BOOT_PORT_16
    else if (MPC_WRITE_BCSR_0 &
        (BPS_DATA_BIT.pin == BOOT_PORT_8) &
         (!KA_PON_RESET # (BPS_PON_DEFAULT != BOOT_PORT_RESERVED)) #
         (KA_PON_RESET & (BPS_PON_DEFAULT == BOOT_PORT_8)) ) then
        BOOT_PORT_8
    else
        BOOT_PORT_RESERVED;
"*****************************************************************************
state_diagram RSV6
  state !RSV6_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV6_DATA_BIT.pin == RSV6_ACTIVE) &
         (!KA_PON_RESET # (RSV6_PON_DEFAULT != !RSV6_ACTIVE)) #
         (KA_PON_RESET & (RSV6_PON_DEFAULT == RSV6_ACTIVE)) ) then
        RSV6_ACTIVE
    else
        !RSV6_ACTIVE;
  state RSV2_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV6_DATA_BIT.pin == !RSV6_ACTIVE) &
         (!KA_PON_RESET # (RSV6_PON_DEFAULT != RSV6_ACTIVE)) #
         (KA_PON_RESET & (RSV6_PON_DEFAULT == !RSV6_ACTIVE)) ) then
        !RSV6_ACTIVE
    else
        RSV6_ACTIVE;

"*****************************************************************************
state_diagram ISB
  state INT_SPACE_BASE_0x00000000:
    if (MPC_WRITE_BCSR_0 &
        (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00F00000) &
         (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00000000)) #
         (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00F00000)) ) then
        INT_SPACE_BASE_0x00F00000
    else if (MPC_WRITE_BCSR_0 &
        (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFF000000) &
         (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00000000)) #
         (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFF000000)) ) then
        INT_SPACE_BASE_0xFF000000
    else if (MPC_WRITE_BCSR_0 &
        (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFFF00000) &
         (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00000000)) #
         (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFFF00000)) ) then
        INT_SPACE_BASE_0xFFF00000
    else
```

```
        INT_SPACE_BASE_0x00000000;

    state INT_SPACE_BASE_0x00F00000:
        if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00000000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00F00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00000000)) ) then
            INT_SPACE_BASE_0x00000000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFF000000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00F00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFF000000)) ) then
            INT_SPACE_BASE_0xFF000000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFFF00000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0x00F00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFFF00000)) ) then
            INT_SPACE_BASE_0xFFF00000
        else
            INT_SPACE_BASE_0x00F00000;

    state INT_SPACE_BASE_0xFF000000:
        if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00000000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFF000000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00000000)) ) then
            INT_SPACE_BASE_0x00000000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00F00000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFF000000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00F00000)) ) then
            INT_SPACE_BASE_0x00F00000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFFF00000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFF000000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFFF00000)) ) then
            INT_SPACE_BASE_0xFFF00000
        else
            INT_SPACE_BASE_0xFF000000;

    state INT_SPACE_BASE_0xFFF00000:
        if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00000000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFFF00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00000000)) ) then
            INT_SPACE_BASE_0x00000000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0x00F00000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFFF00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0x00F00000)) ) then
            INT_SPACE_BASE_0x00F00000
        else if (MPC_WRITE_BCSR_0 &
            (ISB_DATA_BIT.pin == INT_SPACE_BASE_0xFF000000) &
             (!KA_PON_RESET # (ISB_PON_DEFAULT != INT_SPACE_BASE_0xFFF00000)) #
             (KA_PON_RESET & (ISB_PON_DEFAULT == INT_SPACE_BASE_0xFF000000)) ) then
            INT_SPACE_BASE_0xFF000000
        else
            INT_SPACE_BASE_0xFFF00000;
"*********************************************************************
```

```
state_diagram DBGC
  state DEBUG_PINS_PCMCIA_2:
     if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_WATCH_POINTS) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_PCMCIA_2)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_WATCH_POINTS)) ) then
        DEBUG_PINS_WATCH_POINTS
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_RESREVED) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_PCMCIA_2)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_RESREVED)) ) then
        DEBUG_PINS_RESREVED
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_FOR_SHOW) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_PCMCIA_2)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_FOR_SHOW)) ) then
        DEBUG_PINS_FOR_SHOW
     else
        DEBUG_PINS_PCMCIA_2;

  state DEBUG_PINS_WATCH_POINTS:
     if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_PCMCIA_2) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_WATCH_POINTS)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_PCMCIA_2)) ) then
        DEBUG_PINS_PCMCIA_2
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_RESREVED) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_WATCH_POINTS)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_RESREVED)) ) then
        DEBUG_PINS_RESREVED
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_FOR_SHOW) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_WATCH_POINTS)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_FOR_SHOW)) ) then
        DEBUG_PINS_FOR_SHOW
     else
        DEBUG_PINS_WATCH_POINTS;

  state DEBUG_PINS_RESREVED:
     if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_PCMCIA_2) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_RESREVED)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_PCMCIA_2)) ) then
        DEBUG_PINS_PCMCIA_2
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_WATCH_POINTS) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_RESREVED)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_WATCH_POINTS)) ) then
        DEBUG_PINS_WATCH_POINTS
     else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_FOR_SHOW) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_RESREVED)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_FOR_SHOW)) ) then
        DEBUG_PINS_FOR_SHOW
     else
        DEBUG_PINS_RESREVED;
  state DEBUG_PINS_FOR_SHOW:
     if (MPC_WRITE_BCSR_0 &
```

```
        (DBGC_DATA_BIT.pin == DEBUG_PINS_PCMCIA_2) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_FOR_SHOW)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_PCMCIA_2)) ) then
        DEBUG_PINS_PCMCIA_2
    else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_WATCH_POINTS) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_FOR_SHOW)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_WATCH_POINTS)) ) then
        DEBUG_PINS_WATCH_POINTS
    else if (MPC_WRITE_BCSR_0 &
        (DBGC_DATA_BIT.pin == DEBUG_PINS_RESREVED) &
         (!KA_PON_RESET # (DBGC_PON_DEFAULT != DEBUG_PINS_FOR_SHOW)) #
         (KA_PON_RESET & (DBGC_PON_DEFAULT == DEBUG_PINS_RESREVED)) ) then
        DEBUG_PINS_RESREVED
    else
        DEBUG_PINS_FOR_SHOW;
"*****************************************************************************
state_diagram DBPC
  state DEBUG_PORT_ON_JTAG:
    if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_NON_EXISTANT) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_JTAG)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_NON_EXISTANT)) ) then
        DEBUG_PORT_NON_EXISTANT
    else if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_RESERVED) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_JTAG)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_RESERVED)) ) then
        DEBUG_PORT_RESERVED
    else if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_DEBUG_PINS) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_JTAG)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_DEBUG_PINS)) ) then
        DEBUG_PORT_ON_DEBUG_PINS
    else
        DEBUG_PORT_ON_JTAG;

  state DEBUG_PORT_NON_EXISTANT:
    if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_JTAG) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_NON_EXISTANT)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_JTAG)) ) then
        DEBUG_PORT_ON_JTAG
    else if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_RESERVED) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_NON_EXISTANT)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_RESERVED)) ) then
        DEBUG_PORT_RESERVED
    else if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_DEBUG_PINS) &
         (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_NON_EXISTANT)) #
         (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_DEBUG_PINS)) ) then
        DEBUG_PORT_ON_DEBUG_PINS
    else
        DEBUG_PORT_NON_EXISTANT;

  state DEBUG_PORT_RESERVED:
    if (MPC_WRITE_BCSR_0 &
        (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_JTAG) &
```

```
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_RESERVED)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_JTAG)) ) then
         DEBUG_PORT_ON_JTAG
      else if (MPC_WRITE_BCSR_0 &
         (DBPC_DATA_BIT.pin == DEBUG_PORT_NON_EXISTANT) &
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_RESERVED)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_NON_EXISTANT)) ) then
         DEBUG_PORT_NON_EXISTANT
      else if (MPC_WRITE_BCSR_0 &
         (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_DEBUG_PINS) &
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_RESERVED)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_DEBUG_PINS)) ) then
         DEBUG_PORT_ON_DEBUG_PINS
      else
         DEBUG_PORT_RESERVED;

   state DEBUG_PORT_ON_DEBUG_PINS:
      if (MPC_WRITE_BCSR_0 &
         (DBPC_DATA_BIT.pin == DEBUG_PORT_ON_JTAG) &
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_DEBUG_PINS)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_ON_JTAG)) ) then
         DEBUG_PORT_ON_JTAG
      else if (MPC_WRITE_BCSR_0 &
         (DBPC_DATA_BIT.pin == DEBUG_PORT_NON_EXISTANT) &
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_DEBUG_PINS)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_NON_EXISTANT)) ) then
         DEBUG_PORT_NON_EXISTANT
      else if (MPC_WRITE_BCSR_0 &
         (DBPC_DATA_BIT.pin == DEBUG_PORT_RESERVED) &
          (!KA_PON_RESET # (DBPC_PON_DEFAULT != DEBUG_PORT_ON_DEBUG_PINS)) #
          (KA_PON_RESET & (DBPC_PON_DEFAULT == DEBUG_PORT_RESERVED)) ) then
         DEBUG_PORT_RESERVED
      else
         DEBUG_PORT_ON_DEBUG_PINS;

"*****************************************************************************
state_diagram RSV13
   state !RSV13_ACTIVE:
      if (MPC_WRITE_BCSR_0 &
         (RSV13_DATA_BIT.pin == RSV13_ACTIVE) &
          (!KA_PON_RESET # (RSV13_PON_DEFAULT != !RSV13_ACTIVE)) #
          (KA_PON_RESET & (RSV13_PON_DEFAULT == RSV13_ACTIVE)) ) then
         RSV13_ACTIVE
      else
         !RSV13_ACTIVE;
   state RSV13_ACTIVE:
      if (MPC_WRITE_BCSR_0 &
         (RSV13_DATA_BIT.pin == !RSV13_ACTIVE) &
          (!KA_PON_RESET # (RSV13_PON_DEFAULT != RSV13_ACTIVE)) #
          (KA_PON_RESET & (RSV13_PON_DEFAULT == !RSV13_ACTIVE)) ) then
         !RSV13_ACTIVE
      else
         RSV13_ACTIVE;
"*****************************************************************************
state_diagram RSV14
   state !RSV14_ACTIVE:
      if (MPC_WRITE_BCSR_0 &
         (RSV14_DATA_BIT.pin == RSV14_ACTIVE) &
          (!KA_PON_RESET # (RSV14_PON_DEFAULT != !RSV14_ACTIVE)) #
```

```
         (KA_PON_RESET & (RSV14_PON_DEFAULT == RSV14_ACTIVE)) ) then
        RSV14_ACTIVE
    else
        !RSV14_ACTIVE;
  state RSV14_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV14_DATA_BIT.pin == !RSV14_ACTIVE) &
         (!KA_PON_RESET # (RSV14_PON_DEFAULT != RSV14_ACTIVE)) #
         (KA_PON_RESET & (RSV14_PON_DEFAULT == !RSV14_ACTIVE)) ) then
        !RSV14_ACTIVE
    else
        RSV14_ACTIVE;
"*******************************************************************************
state_diagram RSV15
  state !RSV15_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV15_DATA_BIT.pin == RSV15_ACTIVE) &
         (!KA_PON_RESET # (RSV15_PON_DEFAULT != !RSV15_ACTIVE)) #
         (KA_PON_RESET & (RSV15_PON_DEFAULT == RSV15_ACTIVE)) ) then
        RSV15_ACTIVE
    else
        !RSV15_ACTIVE;
  state RSV15_ACTIVE:
    if (MPC_WRITE_BCSR_0 &
        (RSV15_DATA_BIT.pin == !RSV15_ACTIVE) &
         (!KA_PON_RESET # (RSV15_PON_DEFAULT != RSV15_ACTIVE)) #
         (KA_PON_RESET & (RSV15_PON_DEFAULT == !RSV15_ACTIVE)) ) then
        !RSV15_ACTIVE
    else
        RSV15_ACTIVE;
"*******************************************************************************
"*******************************************************************************
"* BCSR 1
"*******************************************************************************
"*******************************************************************************
equations

WideContReg.clk = SYSCLK;
DrivenContReg.oe = ^hfffffff;

state_diagram FlashEn~
  state FLASH_ENABLED:
    if (MPC_WRITE_BCSR_1 &
        (FLASH_ENABLE_DATA_BIT.pin == !FLASH_ENABLED) &
         (!KA_PON_RESET # (FLASH_ENABLE_PON_DEFAULT != FLASH_ENABLED)) #
         (KA_PON_RESET & (FLASH_ENABLE_PON_DEFAULT == !FLASH_ENABLED)) ) then
        !FLASH_ENABLED
    else
        FLASH_ENABLED;
  state !FLASH_ENABLED:
    if (MPC_WRITE_BCSR_1 &
        (FLASH_ENABLE_DATA_BIT.pin == FLASH_ENABLED) &
         (!KA_PON_RESET # (FLASH_ENABLE_PON_DEFAULT != !FLASH_ENABLED)) #
         (KA_PON_RESET & (FLASH_ENABLE_PON_DEFAULT == FLASH_ENABLED)) ) then
        FLASH_ENABLED
    else
        !FLASH_ENABLED;

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

```
state_diagram DramEn~
  state DRAM_ENABLED:
      if (MPC_WRITE_BCSR_1 &
          (DRAM_ENABLE_DATA_BIT.pin == !DRAM_ENABLED) &
           (!KA_PON_RESET # (DRAM_ENABLE_PON_DEFAULT != DRAM_ENABLED)) #
           (KA_PON_RESET & (DRAM_ENABLE_PON_DEFAULT == !DRAM_ENABLED)) ) then
          !DRAM_ENABLED
      else
          DRAM_ENABLED;
  state !DRAM_ENABLED:
      if (MPC_WRITE_BCSR_1 &
          (DRAM_ENABLE_DATA_BIT.pin == DRAM_ENABLED) &
           (!KA_PON_RESET # (DRAM_ENABLE_PON_DEFAULT != !DRAM_ENABLED)) #
           (KA_PON_RESET & (DRAM_ENABLE_PON_DEFAULT == DRAM_ENABLED)) ) then
          DRAM_ENABLED
      else
          !DRAM_ENABLED;

"*****************************************************************************
state_diagram EthEn~
  state ETH_ENABLED:
      if (MPC_WRITE_BCSR_1 &
          (ETH_ENABLE_DATA_BIT.pin == !ETH_ENABLED) &
           (!KA_PON_RESET # (ETH_ENABLE_PON_DEFAULT != ETH_ENABLED)) #
           (KA_PON_RESET & (ETH_ENABLE_PON_DEFAULT == !ETH_ENABLED)) ) then
          !ETH_ENABLED
      else
          ETH_ENABLED;
  state !ETH_ENABLED:
      if (MPC_WRITE_BCSR_1 &
          (ETH_ENABLE_DATA_BIT.pin == ETH_ENABLED) &
           (!KA_PON_RESET # (ETH_ENABLE_PON_DEFAULT != !ETH_ENABLED)) #
           (KA_PON_RESET & (ETH_ENABLE_PON_DEFAULT == ETH_ENABLED)) ) then
          ETH_ENABLED
      else
          !ETH_ENABLED;
"*****************************************************************************
state_diagram InfRedEn~
  state INF_RED_ENABLE:
      if (MPC_WRITE_BCSR_1 &
          (INF_RED_ENABLE_DATA_BIT.pin == !INF_RED_ENABLE) &
           (!KA_PON_RESET # (INF_RED_ENABLE_PON_DEFAULT != INF_RED_ENABLE)) #
           (KA_PON_RESET & (INF_RED_ENABLE_PON_DEFAULT == !INF_RED_ENABLE)) ) then
          !INF_RED_ENABLE
      else
          INF_RED_ENABLE;
  state !INF_RED_ENABLE:
      if (MPC_WRITE_BCSR_1 &
          (INF_RED_ENABLE_DATA_BIT.pin == INF_RED_ENABLE) &
           (!KA_PON_RESET # (INF_RED_ENABLE_PON_DEFAULT != !INF_RED_ENABLE)) #
           (KA_PON_RESET & (INF_RED_ENABLE_PON_DEFAULT == INF_RED_ENABLE)) ) then
          INF_RED_ENABLE
      else
          !INF_RED_ENABLE;
"*****************************************************************************
state_diagram FlashCfgEn~
  state FLASH_CFG_ENABLE:
      if (MPC_WRITE_BCSR_1 &
          (FLASH_CFG_ENABLE_DATA_BIT.pin == !FLASH_CFG_ENABLE) &
```

```
         (!KA_PON_RESET # (FLASH_CFG_ENABLE_PON_DEFAULT != FLASH_CFG_ENABLE)) #
         (KA_PON_RESET & (FLASH_CFG_ENABLE_PON_DEFAULT == !FLASH_CFG_ENABLE)) ) then
        !FLASH_CFG_ENABLE
     else
        FLASH_CFG_ENABLE;
  state !FLASH_CFG_ENABLE:
     if (MPC_WRITE_BCSR_1 &
        (FLASH_CFG_ENABLE_DATA_BIT.pin == FLASH_CFG_ENABLE) &
         (!KA_PON_RESET # (FLASH_CFG_ENABLE_PON_DEFAULT != !FLASH_CFG_ENABLE)) #
         (KA_PON_RESET & (FLASH_CFG_ENABLE_PON_DEFAULT == FLASH_CFG_ENABLE)) ) then
        FLASH_CFG_ENABLE
     else
        !FLASH_CFG_ENABLE;
"*****************************************************************************
"* To avoid in advertant write to the Control Register Enable bit, which might
"* result in a need to re-power the board - protection logic is provided.
"* In order of writing the Control Register Enable this bit in the status register
"* must be negated. After any write to the control register, this bit asserts
"* again (to protected mode)
"*****************************************************************************
equations

CntRegEnProtect~.clk = SYSCLK;

state_diagram CntRegEnProtect~
  state CNT_REG_EN_PROTECT:
     if (MPC_WRITE_BCSR_3 &
        (CNT_REG_EN_PROTECT_DATA_BIT.pin == !CNT_REG_EN_PROTECT) &
         (!KA_PON_RESET # (CNT_REG_EN_PROTECT_PON_DEFAULT != CNT_REG_EN_PROTECT)) #
         (KA_PON_RESET & (CNT_REG_EN_PROTECT_PON_DEFAULT == !CNT_REG_EN_PROTECT)) ) then
        !CNT_REG_EN_PROTECT
     else
        CNT_REG_EN_PROTECT;
  state !CNT_REG_EN_PROTECT:
     if (MPC_WRITE_BCSR_3 &
        (CNT_REG_EN_PROTECT_DATA_BIT.pin == CNT_REG_EN_PROTECT) &
         (!KA_PON_RESET # (CNT_REG_EN_PROTECT_PON_DEFAULT != !CNT_REG_EN_PROTECT)) #
         (KA_PON_RESET & (CNT_REG_EN_PROTECT_PON_DEFAULT == CNT_REG_EN_PROTECT)) #
         MPC_WRITE_BCSR_1) then " any write to control reg 1
        CNT_REG_EN_PROTECT
     else
        !CNT_REG_EN_PROTECT;
"*****************************************************************************
"* protected by CntRegEnProtect~ to prevent from inadvertant write
"*****************************************************************************
state_diagram CntRegEn~
  state CONT_REG_ENABLE:
     if (MPC_WRITE_BCSR_1 & (CntRegEnProtect~.fb != CNT_REG_EN_PROTECT) &
        (CONT_REG_ENABLE_DATA_BIT.pin == !CONT_REG_ENABLE) &
         (!KA_PON_RESET # (CONT_REG_ENABLE_PON_DEFAULT != CONT_REG_ENABLE)) #
         (KA_PON_RESET & (CONT_REG_ENABLE_PON_DEFAULT == !CONT_REG_ENABLE)) ) then
        !CONT_REG_ENABLE
     else
        CONT_REG_ENABLE;
  state !CONT_REG_ENABLE:" in fact not applicable
     if (MPC_WRITE_BCSR_1 &
        (CONT_REG_ENABLE_DATA_BIT.pin == CONT_REG_ENABLE) &
         (!KA_PON_RESET # (CONT_REG_ENABLE_PON_DEFAULT != !CONT_REG_ENABLE)) #
         (KA_PON_RESET & (CONT_REG_ENABLE_PON_DEFAULT == CONT_REG_ENABLE)) ) then
```

```
        CONT_REG_ENABLE
    else
        !CONT_REG_ENABLE;
"*****************************************************************************
state_diagram RS232En1~
  state RS232_1_ENABLE:
    if (MPC_WRITE_BCSR_1 &
        (RS232_1_ENABLE_DATA_BIT.pin == !RS232_1_ENABLE) &
         (!KA_PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != RS232_1_ENABLE)) #
         (KA_PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == !RS232_1_ENABLE)) ) then
        !RS232_1_ENABLE
    else
        RS232_1_ENABLE;
  state !RS232_1_ENABLE:
    if (MPC_WRITE_BCSR_1 &
        (RS232_1_ENABLE_DATA_BIT.pin == RS232_1_ENABLE) &
         (!KA_PON_RESET # (RS232_1_ENABLE_PON_DEFAULT != !RS232_1_ENABLE)) #
         (KA_PON_RESET & (RS232_1_ENABLE_PON_DEFAULT == RS232_1_ENABLE)) ) then
        RS232_1_ENABLE
    else
        !RS232_1_ENABLE;
"*****************************************************************************
state_diagram PccEn~
  state PCC_ENABLE:
    if (MPC_WRITE_BCSR_1 &
        (PCC_ENABLE_DATA_BIT.pin == !PCC_ENABLE) &
         (!KA_PON_RESET # (PCC_ENABLE_PON_DEFAULT != PCC_ENABLE)) #
         (KA_PON_RESET & (PCC_ENABLE_PON_DEFAULT == !PCC_ENABLE)) ) then
        !PCC_ENABLE
    else
        PCC_ENABLE;
  state !PCC_ENABLE:
    if (MPC_WRITE_BCSR_1 &
        (PCC_ENABLE_DATA_BIT.pin == PCC_ENABLE) &
         (!KA_PON_RESET # (PCC_ENABLE_PON_DEFAULT != !PCC_ENABLE)) #
         (KA_PON_RESET & (PCC_ENABLE_PON_DEFAULT == PCC_ENABLE)) ) then
        PCC_ENABLE
    else
        !PCC_ENABLE;
"*****************************************************************************
state_diagram PccVcc0
  state PCC_VCC_CONT_0:
    if (MPC_WRITE_BCSR_1 &
        (PCC_VCC_0_DATA_BIT.pin == !PCC_VCC_CONT_0) &
         (!KA_PON_RESET # (PCC_VCC_0_PON_DEFAULT != PCC_VCC_CONT_0)) #
         (KA_PON_RESET & (PCC_VCC_0_PON_DEFAULT == !PCC_VCC_CONT_0)) ) then
        !PCC_VCC_CONT_0
    else
        PCC_VCC_CONT_0;
  state !PCC_VCC_CONT_0:
    if (MPC_WRITE_BCSR_1 &
        (PCC_VCC_0_DATA_BIT.pin == PCC_VCC_CONT_0) &
         (!KA_PON_RESET # (PCC_VCC_0_PON_DEFAULT != !PCC_VCC_CONT_0)) #
         (KA_PON_RESET & (PCC_VCC_0_PON_DEFAULT == PCC_VCC_CONT_0)) ) then
        PCC_VCC_CONT_0
    else
        !PCC_VCC_CONT_0;
"*****************************************************************************
state_diagram PccVpp0
```

```
  state PCC_VPP0:
     if (MPC_WRITE_BCSR_1 &
         (PCC_VPP0_DATA_BIT.pin == !PCC_VPP0) &
          (!KA_PON_RESET # (PCC_VPP0_PON_DEFAULT != PCC_VPP0)) #
          (KA_PON_RESET & (PCC_VPP0_PON_DEFAULT == !PCC_VPP0)) ) then
         !PCC_VPP0
     else
         PCC_VPP0;
  state !PCC_VPP0:
     if (MPC_WRITE_BCSR_1 &
         (PCC_VPP0_DATA_BIT.pin == PCC_VPP0) &
          (!KA_PON_RESET # (PCC_VPP0_PON_DEFAULT != !PCC_VPP0)) #
          (KA_PON_RESET & (PCC_VPP0_PON_DEFAULT == PCC_VPP0)) ) then
         PCC_VPP0
     else
         !PCC_VPP0;

"*****************************************************************************
state_diagram PccVpp1
  state PCC_VPP1:
     if (MPC_WRITE_BCSR_1 &
         (PCC_VPP1_DATA_BIT.pin == !PCC_VPP1) &
          (!KA_PON_RESET # (PCC_VPP1_PON_DEFAULT != PCC_VPP1)) #
          (KA_PON_RESET & (PCC_VPP1_PON_DEFAULT == !PCC_VPP1)) ) then
         !PCC_VPP1
     else
         PCC_VPP1;
  state !PCC_VPP1:
     if (MPC_WRITE_BCSR_1 &
         (PCC_VPP1_DATA_BIT.pin == PCC_VPP1) &
          (!KA_PON_RESET # (PCC_VPP1_PON_DEFAULT != !PCC_VPP1)) #
          (KA_PON_RESET & (PCC_VPP1_PON_DEFAULT == PCC_VPP1)) ) then
         PCC_VPP1
     else
         !PCC_VPP1;
"*****************************************************************************
state_diagram HalfWord~
  state HALF_WORD:
     if (MPC_WRITE_BCSR_1 &
         (HALF_WORD_DATA_BIT.pin == !HALF_WORD) &
          (!KA_PON_RESET # (HALF_WORD_PON_DEFAULT != HALF_WORD)) #
          (KA_PON_RESET & (HALF_WORD_PON_DEFAULT == !HALF_WORD)) ) then
         !HALF_WORD
     else
         HALF_WORD;
  state !HALF_WORD:
     if (MPC_WRITE_BCSR_1 &
         (HALF_WORD_DATA_BIT.pin == HALF_WORD) &
          (!KA_PON_RESET # (HALF_WORD_PON_DEFAULT != !HALF_WORD)) #
          (KA_PON_RESET & (HALF_WORD_PON_DEFAULT == HALF_WORD)) ) then
         HALF_WORD
     else
         !HALF_WORD;
"*****************************************************************************
state_diagram RS232En2~
  state RS232_2_ENABLE:
     if (MPC_WRITE_BCSR_1 &
         (RS232_2_ENABLE_DATA_BIT.pin == !RS232_2_ENABLE) &
          (!KA_PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != RS232_2_ENABLE)) #
```

```
         (KA_PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == !RS232_2_ENABLE)) ) then
        !RS232_2_ENABLE
    else
        RS232_2_ENABLE;
  state !RS232_2_ENABLE:
    if (MPC_WRITE_BCSR_1 &
        (RS232_2_ENABLE_DATA_BIT.pin == RS232_2_ENABLE) &
         (!KA_PON_RESET # (RS232_2_ENABLE_PON_DEFAULT != !RS232_2_ENABLE)) #
         (KA_PON_RESET & (RS232_2_ENABLE_PON_DEFAULT == RS232_2_ENABLE)) ) then
        RS232_2_ENABLE
    else
        !RS232_2_ENABLE;
"*****************************************************************************
state_diagram PccVcc1
  state PCC_VCC_CONT_0:
    if (MPC_WRITE_BCSR_1 &
        (PCC_VCC_1_DATA_BIT.pin == !PCC_VCC_CONT_0) &
         (!KA_PON_RESET # (PCC_VCC_1_PON_DEFAULT != PCC_VCC_CONT_0)) #
         (KA_PON_RESET & (PCC_VCC_1_PON_DEFAULT == !PCC_VCC_CONT_0)) ) then
        !PCC_VCC_CONT_0
    else
        PCC_VCC_CONT_0;
  state !PCC_VCC_CONT_0:
    if (MPC_WRITE_BCSR_1 &
        (PCC_VCC_1_DATA_BIT.pin == PCC_VCC_CONT_0) &
         (!KA_PON_RESET # (PCC_VCC_1_PON_DEFAULT != !PCC_VCC_CONT_0)) #
         (KA_PON_RESET & (PCC_VCC_1_PON_DEFAULT == PCC_VCC_CONT_0)) ) then
        PCC_VCC_CONT_0
    else
        !PCC_VCC_CONT_0;
"*****************************************************************************
state_diagram SdramEn~
  state SDRAM_ENABLED:
    if (MPC_WRITE_BCSR_1 &
        (SDRAM_ENABLE_DATA_BIT.pin == !SDRAM_ENABLED) &
         (!KA_PON_RESET # (SDRAM_ENABLE_PON_DEFAULT != SDRAM_ENABLED)) #
         (KA_PON_RESET & (SDRAM_ENABLE_PON_DEFAULT == !SDRAM_ENABLED)) ) then
        !SDRAM_ENABLED
    else
        SDRAM_ENABLED;
  state !SDRAM_ENABLED:
    if (MPC_WRITE_BCSR_1 &
        (SDRAM_ENABLE_DATA_BIT.pin == SDRAM_ENABLED) &
         (!KA_PON_RESET # (SDRAM_ENABLE_PON_DEFAULT != !SDRAM_ENABLED)) #
         (KA_PON_RESET & (SDRAM_ENABLE_PON_DEFAULT == SDRAM_ENABLED)) ) then
        SDRAM_ENABLED
    else
        !SDRAM_ENABLED;

"*****************************************************************************
"*****************************************************************************
"* BCSR4 State Machines
"*****************************************************************************
"*****************************************************************************


state_diagram UsbFethEn~
  state USB_FETH_ENABLED:
    if (MPC_WRITE_BCSR_4 &
```

```
        (USB_FETH_EN_DATA_BIT.pin == !USB_FETH_ENABLED) &
         (!KA_PON_RESET # (USB_FETH_EN_PON_DEFAULT != USB_FETH_ENABLED)) #
         (KA_PON_RESET & (USB_FETH_EN_PON_DEFAULT == !USB_FETH_ENABLED)) ) then
        !USB_FETH_ENABLED
    else
        USB_FETH_ENABLED;
  state !USB_FETH_ENABLED:
    if (MPC_WRITE_BCSR_4 &
        (USB_FETH_EN_DATA_BIT.pin == USB_FETH_ENABLED) &
         (!KA_PON_RESET # (USB_FETH_EN_PON_DEFAULT != !USB_FETH_ENABLED)) #
         (KA_PON_RESET & (USB_FETH_EN_PON_DEFAULT == USB_FETH_ENABLED)) ) then
        USB_FETH_ENABLED
    else
        !USB_FETH_ENABLED;

"*****************************************************************************


state_diagram UsbSpeed
  state USB_FULL_SPEED:
    if (MPC_WRITE_BCSR_4 &
        (USB_SPEED_DATA_BIT.pin == !USB_FULL_SPEED) &
         (!KA_PON_RESET # (USB_SPEED_PON_DEFAULT != USB_FULL_SPEED)) #
         (KA_PON_RESET & (USB_SPEED_PON_DEFAULT == !USB_FULL_SPEED)) ) then
        !USB_FULL_SPEED
    else
        USB_FULL_SPEED;
  state !USB_FULL_SPEED:
    if (MPC_WRITE_BCSR_4 &
        (USB_SPEED_DATA_BIT.pin == USB_FULL_SPEED) &
         (!KA_PON_RESET # (USB_SPEED_PON_DEFAULT != !USB_FULL_SPEED)) #
         (KA_PON_RESET & (USB_SPEED_PON_DEFAULT == USB_FULL_SPEED)) ) then
        USB_FULL_SPEED
    else
        !USB_FULL_SPEED;


"*****************************************************************************
state_diagram UsbVcc0
  state USB_VCC_CONT_0:
    if (MPC_WRITE_BCSR_4 &
        (USB_VCC_0_DATA_BIT.pin == !USB_VCC_CONT_0) &
         (!KA_PON_RESET # (USB_VCC_0_CONT_PON_DEFAULT != USB_VCC_CONT_0)) #
         (KA_PON_RESET & (USB_VCC_0_CONT_PON_DEFAULT == !USB_VCC_CONT_0)) ) then
        !USB_VCC_CONT_0
    else
        USB_VCC_CONT_0;
  state !USB_VCC_CONT_0:
    if (MPC_WRITE_BCSR_4 &
        (USB_VCC_0_DATA_BIT.pin == USB_VCC_CONT_0) &
         (!KA_PON_RESET # (USB_VCC_0_CONT_PON_DEFAULT != !USB_VCC_CONT_0)) #
         (KA_PON_RESET & (USB_VCC_0_CONT_PON_DEFAULT == USB_VCC_CONT_0)) ) then
        USB_VCC_CONT_0
    else
        !USB_VCC_CONT_0;
"*****************************************************************************
state_diagram UsbVcc1
  state USB_VCC_CONT_0:
    if (MPC_WRITE_BCSR_4 &
```

```
        (USB_VCC_1_DATA_BIT.pin == !USB_VCC_CONT_0) &
         (!KA_PON_RESET # (USB_VCC_1_CONT_PON_DEFAULT != USB_VCC_CONT_0)) #
         (KA_PON_RESET & (USB_VCC_1_CONT_PON_DEFAULT == !USB_VCC_CONT_0)) ) then
        !USB_VCC_CONT_0
    else
        USB_VCC_CONT_0;
  state !USB_VCC_CONT_0:
    if (MPC_WRITE_BCSR_4 &
        (USB_VCC_1_DATA_BIT.pin == USB_VCC_CONT_0) &
         (!KA_PON_RESET # (USB_VCC_1_CONT_PON_DEFAULT != !USB_VCC_CONT_0)) #
         (KA_PON_RESET & (USB_VCC_1_CONT_PON_DEFAULT == USB_VCC_CONT_0)) ) then
        USB_VCC_CONT_0
    else
        !USB_VCC_CONT_0;

"*****************************************************************************
state_diagram VideoOn~
  state VIDEO_ENABLED:
    if (MPC_WRITE_BCSR_4 &
        (VIDEO_ENABLE_DATA_BIT.pin == !VIDEO_ENABLED) &
         (!KA_PON_RESET # (VIDEO_ENABLE_PON_DEFAULT != VIDEO_ENABLED)) #
         (KA_PON_RESET & (VIDEO_ENABLE_PON_DEFAULT == !VIDEO_ENABLED)) ) then
        !VIDEO_ENABLED
    else
        VIDEO_ENABLED;
  state !VIDEO_ENABLED:
    if (MPC_WRITE_BCSR_4 &
        (VIDEO_ENABLE_DATA_BIT.pin == VIDEO_ENABLED) &
         (!KA_PON_RESET # (VIDEO_ENABLE_PON_DEFAULT != !VIDEO_ENABLED)) #
         (KA_PON_RESET & (VIDEO_ENABLE_PON_DEFAULT == VIDEO_ENABLED)) ) then
        VIDEO_ENABLED
    else
        !VIDEO_ENABLED;

"*****************************************************************************
state_diagram VideoExtClkEn
  state VIDEO_EXT_CLK_ENABLED:
    if (MPC_WRITE_BCSR_4 &
        (VIDEO_EXT_CLK_EN_DATA_BIT.pin == !VIDEO_EXT_CLK_ENABLED) &
         (!KA_PON_RESET # (VIDEO_EXT_CLK_EN_PON_DEFAULT != VIDEO_EXT_CLK_ENABLED)) #
         (KA_PON_RESET & (VIDEO_EXT_CLK_EN_PON_DEFAULT == !VIDEO_EXT_CLK_ENABLED)) ) then
        !VIDEO_EXT_CLK_ENABLED
    else
        VIDEO_EXT_CLK_ENABLED;
  state !VIDEO_EXT_CLK_ENABLED:
    if (MPC_WRITE_BCSR_4 &
        (VIDEO_EXT_CLK_EN_DATA_BIT.pin == VIDEO_EXT_CLK_ENABLED) &
         (!KA_PON_RESET # (VIDEO_EXT_CLK_EN_PON_DEFAULT != !VIDEO_EXT_CLK_ENABLED)) #
         (KA_PON_RESET & (VIDEO_EXT_CLK_EN_PON_DEFAULT == VIDEO_EXT_CLK_ENABLED)) ) then
        VIDEO_EXT_CLK_ENABLED
    else
        !VIDEO_EXT_CLK_ENABLED;
"*****************************************************************************
state_diagram VideoRst~
  state VIDEO_RESET_ACTIVE:
    if (MPC_WRITE_BCSR_4 &
        (VIDEO_RESET_DATA_BIT.pin == !VIDEO_RESET_ACTIVE) &
         (!KA_PON_RESET # (VIDEO_RESET_PON_DEFAULT != VIDEO_RESET_ACTIVE)) #
         (KA_PON_RESET & (VIDEO_RESET_PON_DEFAULT == !VIDEO_RESET_ACTIVE)) ) then
```

```
        !VIDEO_RESET_ACTIVE
      else
        VIDEO_RESET_ACTIVE;
  state !VIDEO_RESET_ACTIVE:
      if (MPC_WRITE_BCSR_4 &
        (VIDEO_RESET_DATA_BIT.pin == VIDEO_RESET_ACTIVE) &
         (!KA_PON_RESET # (VIDEO_RESET_PON_DEFAULT != !VIDEO_RESET_ACTIVE)) #
         (KA_PON_RESET & (VIDEO_RESET_PON_DEFAULT == VIDEO_RESET_ACTIVE)) ) then
        VIDEO_RESET_ACTIVE
      else
        !VIDEO_RESET_ACTIVE;
"*******************************************************************************
state_diagram SignaLamp~
  state SIGNAL_LAMP_ON:
      if (MPC_WRITE_BCSR_4 &
        (SIGNAL_LAMP_DATA_BIT.pin == !SIGNAL_LAMP_ON) &
         (!KA_PON_RESET # (SIGNAL_LAMP_PON_DEFAULT != SIGNAL_LAMP_ON)) #
         (KA_PON_RESET & (SIGNAL_LAMP_PON_DEFAULT == !SIGNAL_LAMP_ON)) ) then
        !SIGNAL_LAMP_ON
      else
        SIGNAL_LAMP_ON;
  state !SIGNAL_LAMP_ON:
      if (MPC_WRITE_BCSR_4 &
        (SIGNAL_LAMP_DATA_BIT.pin == SIGNAL_LAMP_ON) &
         (!KA_PON_RESET # (SIGNAL_LAMP_PON_DEFAULT != !SIGNAL_LAMP_ON)) #
         (KA_PON_RESET & (SIGNAL_LAMP_PON_DEFAULT == SIGNAL_LAMP_ON)) ) then
        SIGNAL_LAMP_ON
      else
        !SIGNAL_LAMP_ON;
"*******************************************************************************
state_diagram EthLoop
  state ETH_LOOP:
      if (MPC_WRITE_BCSR_4 &
        (ETH_LOOP_DATA_BIT.pin == !ETH_LOOP) &
         (!KA_PON_RESET # (ETH_LOOP_PON_DEFAULT != ETH_LOOP)) #
         (KA_PON_RESET & (ETH_LOOP_PON_DEFAULT == !ETH_LOOP)) ) then
        !ETH_LOOP
      else
        ETH_LOOP;
  state !ETH_LOOP:
      if (MPC_WRITE_BCSR_4 &
        (ETH_LOOP_DATA_BIT.pin == ETH_LOOP) &
         (!KA_PON_RESET # (ETH_LOOP_PON_DEFAULT != !ETH_LOOP)) #
         (KA_PON_RESET & (ETH_LOOP_PON_DEFAULT == ETH_LOOP)) ) then
        ETH_LOOP
      else
        !ETH_LOOP;
"*******************************************************************************
state_diagram TPFLDL~
  state ETH_FULL_DUP:
      if (MPC_WRITE_BCSR_4 &
        (ETH_FULL_DUP_DATA_BIT.pin == !ETH_FULL_DUP) &
         (!KA_PON_RESET # (ETH_FULL_DUP_PON_DEFAULT != ETH_FULL_DUP)) #
         (KA_PON_RESET & (ETH_FULL_DUP_PON_DEFAULT == !ETH_FULL_DUP)) ) then
        !ETH_FULL_DUP
      else
        ETH_FULL_DUP;
  state !ETH_FULL_DUP:
      if (MPC_WRITE_BCSR_4 &
```

```
        (ETH_FULL_DUP_DATA_BIT.pin == ETH_FULL_DUP) &
         (!KA_PON_RESET # (ETH_FULL_DUP_PON_DEFAULT != !ETH_FULL_DUP)) #
         (KA_PON_RESET & (ETH_FULL_DUP_PON_DEFAULT == ETH_FULL_DUP)) ) then
        ETH_FULL_DUP
    else
        !ETH_FULL_DUP;
"*****************************************************************************
state_diagram TPSQEL~
  state ETH_CLSN_TEST:
    if (MPC_WRITE_BCSR_4 &
        (ETH_CLSN_TEST_DATA_BIT.pin == !ETH_CLSN_TEST) &
         (!KA_PON_RESET # (ETH_CLSN_TEST_PON_DEFAULT != ETH_CLSN_TEST)) #
         (KA_PON_RESET & (ETH_CLSN_TEST_PON_DEFAULT == !ETH_CLSN_TEST)) ) then
        !ETH_CLSN_TEST
    else
        ETH_CLSN_TEST;
  state !ETH_CLSN_TEST:
    if (MPC_WRITE_BCSR_4 &
        (ETH_CLSN_TEST_DATA_BIT.pin == ETH_CLSN_TEST) &
         (!KA_PON_RESET # (ETH_CLSN_TEST_PON_DEFAULT != !ETH_CLSN_TEST)) #
         (KA_PON_RESET & (ETH_CLSN_TEST_PON_DEFAULT == ETH_CLSN_TEST)) ) then
        ETH_CLSN_TEST
    else
        !ETH_CLSN_TEST;
"*****************************************************************************
state_diagram ModemEn~
  state MODEM_ENABLED_FOR_823:
    if (MPC_WRITE_BCSR_4 &
        (MODEM_ENABLE_DATA_BIT.pin == !MODEM_ENABLED_FOR_823) &
         (!KA_PON_RESET # (MODEM_ENABLE_PON_DEFAULT != MODEM_ENABLED_FOR_823)) #
         (KA_PON_RESET & (MODEM_ENABLE_PON_DEFAULT == !MODEM_ENABLED_FOR_823)) ) then
        !MODEM_ENABLED_FOR_823
    else
        MODEM_ENABLED_FOR_823;
  state !MODEM_ENABLED_FOR_823:
    if (MPC_WRITE_BCSR_4 &
        (MODEM_ENABLE_DATA_BIT.pin == MODEM_ENABLED_FOR_823) &
         (!KA_PON_RESET # (MODEM_ENABLE_PON_DEFAULT != !MODEM_ENABLED_FOR_823)) #
         (KA_PON_RESET & (MODEM_ENABLE_PON_DEFAULT == MODEM_ENABLED_FOR_823)) ) then
        MODEM_ENABLED_FOR_823
    else
        !MODEM_ENABLED_FOR_823;
"*****************************************************************************
state_diagram Modem_Audio~
  state MODEM:
    if (MPC_WRITE_BCSR_4 &
        (MODEM_FUNC_SEL_DATA_BIT.pin == !MODEM) &
         (!KA_PON_RESET # (MODEM_FUNC_SEL_PON_DEFAULT != MODEM)) #
         (KA_PON_RESET & (MODEM_FUNC_SEL_PON_DEFAULT == !MODEM)) ) then
        !MODEM
    else
        MODEM;
  state !MODEM:
    if (MPC_WRITE_BCSR_4 &
        (MODEM_FUNC_SEL_DATA_BIT.pin == MODEM) &
         (!KA_PON_RESET # (MODEM_FUNC_SEL_PON_DEFAULT != !MODEM)) #
         (KA_PON_RESET & (MODEM_FUNC_SEL_PON_DEFAULT == MODEM)) ) then
        MODEM
    else
```

```
        !MODEM;
"*******************************************************************************
"*******************************************************************************
" External Read Registers' Chip-Selects
"*******************************************************************************
"*******************************************************************************
equations

Bcsr2_3Cs~.oe = 3;

!Bcsr2Cs~ = MPC_READ_BCSR_2;
!Bcsr3Cs~ = MPC_READ_BCSR_3;

"*******************************************************************************
"*******************************************************************************
"* Read Registers.
"* All registers have read capabilty.
"*******************************************************************************
"*******************************************************************************
equations
  DataOe = MPC_READ_BCSR_0 #
           MPC_READ_BCSR_1 #
           MPC_READ_BCSR_4 #
       RESET_CONFIG_DRIVEN;
  Data.oe = DataOe;
  "Data.oe = ^hffff;

  when (MPC_READ_BCSR_0 #
      RESET_CONFIG_DRIVEN) then
      Data = [ERB.fb,IP~.fb,RSV2.fb,BDIS.fb,BPS0.fb,BPS1.fb,RSV6.fb,
          ISB0.fb,ISB1.fb,DBGC0.fb,DBGC1.fb,DBPC0.fb,DBPC1.fb,
          RSV13.fb,RSV14.fb,RSV15.fb];
  else when (MPC_READ_BCSR_1) then
      Data = ReadBcsr1;
  else when (MPC_READ_BCSR_4) then
         "Data = [UsbFethEn~,UsbSpeed,UsbVcc0, UsbVcc1, VideoOn~,VideoExtClkEn,
         "        VideoRst~,SignaLamp~,EthLoop,TPFLDL~,TPSQEL~,Modem_Audio~,0,0,0,0];
                        Data = [EthLoop,TPFLDL~,TPSQEL~,SignaLamp~,UsbFethEn~,UsbSpeed,UsbVcc0,
                                UsbVcc1, VideoOn~,VideoExtClkEn,VideoRst~,ModemEn~,Modem_Audio~,0,0,0];
"*******************************************************************************
"*******************************************************************************
"* Additional Equations from brd_ctl13
"*******************************************************************************
"*******************************************************************************

"********************
"* Flash Chip Select
"********************

equations

FlashCsOut.oe = ^hf;

!FlashCs1~ = FLASH_ACTIVE & !FlashCs~ & FLASH_BANK1;
!FlashCs2~ = FLASH_ACTIVE & !FlashCs~ & FLASH_BANK2 ;
!FlashCs3~ = FLASH_ACTIVE & !FlashCs~ & FLASH_BANK3 ;
!FlashCs4~ = FLASH_ACTIVE & !FlashCs~ & FLASH_BANK4 ;

FlashOe~.oe = H;
```

```
!FlashOe~ = FLASH_ACTIVE & R_W~;

"*****************************************************************************
"* Dram Address lines.
"* These lines are conencted to the dram high order address lines A9 and A10
"* (if available). These lines change value according to the dram size and
"* port size.
"* The dram size is encoded from the presence detect lines (see definitions
"* above) and the port size is determined by the control register.
"*****************************************************************************
equations

DramAdd.oe = 3;

when (!IS_HALF_WORD # IS_HALF_WORD & (SIMM36400 # SIMM36800)) then
  DramAdd9 = A20;
else
  DramAdd9 = A30;

when ( (SIMM36400 # SIMM36800) & !IS_HALF_WORD) then
  DramAdd10 = A19;
else when ( (SIMM36400 # SIMM36800) & IS_HALF_WORD) then
  DramAdd10 = A30;
else
  DramAdd10 = 0;

"*****************************************************************************
"* RAS generation.
"* Since the dram simm requires RAS signals to be split due to high capacitive
"* load and to allow 16 bit operation. When working with 16 bit port size,
"* the double drive RAS signals are disabled.
"*****************************************************************************
equations

RAS.oe = ^hf;

!Ras1~ = !DramBank1Cs~ & DramBank2Cs~ & DRAM_ACTIVE;
!Ras2~ = !DramBank2Cs~ & DramBank1Cs~ & DRAM_ACTIVE & (SIMM36200 # SIMM36800);

!Ras1DD~ = !DramBank1Cs~ & DramBank2Cs~ & DRAM_ACTIVE;
!Ras2DD~ = !DramBank2Cs~ & DramBank1Cs~ & DRAM_ACTIVE & (SIMM36200 # SIMM36800);

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

Reset.oe = ResetEn;

Reset = 0;" open drain

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

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


HardResetEn = RstDeb1.fb & AbrDeb1.fb " both buttons are depressed;
      # REGULAR_POWER_ON_RESET;
```

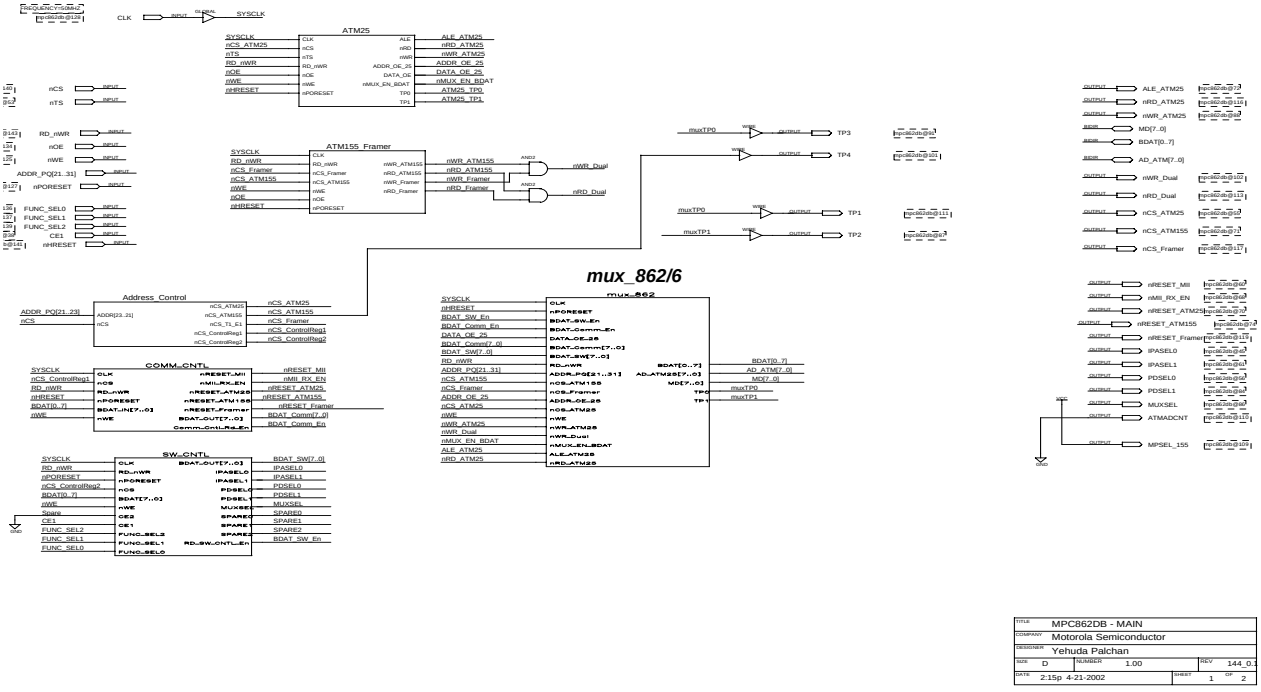
```
SoftResetEn = RstDeb1.fb & !AbrDeb1.fb;" only reset button depressed

"*****************************************************************************
"* Power On reset configuration
"*****************************************************************************
equations

Modck.oe = ModckOe;

ModckOe = DRIVE_MODCK_TO_PDA;

Modck2 = L;
@ifndef SLOW_32K_LOCK {

  Modck2 = ModIn;" support for 1:513 (32KHz crystal) or
  Modck1 = ModIn;" 1:5 (5MHz clock gen.) via CLK4IN
}

@ifdef SLOW_32K_LOCK {

  Modck2 = !ModIn;" support for 1:1 or 1:5 from CLK4IN only
  Modck1 = H;" no support for 32K oscillator.
}

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

ResetConfig~.oe = H;
DriveConfig~.oe = H;
"* Configuration hold counter. Since the rise time of the HARD RESET signal
"* is relatively slow, there is a need to provide a hold time for reset
"* configuration.

ConfigHold.clk = SYSCLK;

when (SyncHardReset~.fb & !ConfigHoldEnd.fb) then ConfigHold := ConfigHold.fb +1;
else when (SyncHardReset~.fb & ConfigHoldEnd.fb) then ConfigHold := ConfigHold.fb;
else when (!SyncHardReset~.fb) then ConfigHold := 0;

ConfigHoldEnd = (ConfigHold.fb == HARD_CONFIG_HOLD_VALUE); " terminal count

!ResetConfig~ = !HardReset~;" drives RSTCONF~ to pda

!DriveConfig~ = !ConfigHoldEnd.fb; " drives configuration data on the bus.

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

NMI~.oe = NMIEn;

NMI~ = 0;" O.D.

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

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

SyncHardReset~.clk = SYSCLK;
DSyncHardReset~.clk = SYSCLK;

SyncHardReset~ := HardReset~;
DSyncHardReset~ := SyncHardReset~.fb;
SyncTEA~.clk = SYSCLK;
SyncTEA~ := TEA~;

LocDataBufEn.oe = 3;


!UpperHalfEn~ = (!DramBank1Cs~ & DRAM_ACTIVE #
     !DramBank2Cs~ & (SIMM36200 # SIMM36800) & DRAM_ACTIVE #
     !FlashCs~  & FLASH_ACTIVE #
     !BrdContRegCs~ & CONTROL_REG_ENABLED #
     !PccCE1~ & PCC_ENABLED #
     !PccCE2~ & PCC_ENABLED #
     !ConfigHoldEnd.fb)  &
                 (STATE_HOLD_OFF_CONSIDERED & !(HOLD_OFF_PERIOD) #
               STATE_NO_HOLD_OFF)
        # !cs5 # !cs6 # !cs7;  "add haim

!LowerHalfEn~ = (!DramBank1Cs~ & DRAM_ACTIVE & !IS_HALF_WORD #
     !DramBank2Cs~ & (SIMM36200 # SIMM36800) &
          !IS_HALF_WORD & DRAM_ACTIVE #
     !FlashCs~  & FLASH_ACTIVE #
     !ConfigHoldEnd.fb & FLASH_CONFIGURATION_ENABLED) &
        (STATE_HOLD_OFF_CONSIDERED & !(HOLD_OFF_PERIOD) #
         STATE_NO_HOLD_OFF);

"*****************************************************************************
"* local data buffers disable (data contention protection)
"*****************************************************************************
equations

HoldOffConsidered.clk = SYSCLK;

D_FlashOe~ = FlashOe~;
DD_FlashOe~ = D_FlashOe~.fb;
TD_FlashOe~ = DD_FlashOe~.fb;
"* QD_FlashOe~ = TD_FlashOe~.fb;
"* PD_FlashOe~ = QD_FlashOe~.fb;

@ifdef DEBUG {
equations

HoldOffConsidered := HOLD_OFF_CONSIDERED;

}

@ifndef DEBUG {

state_diagram HoldOffConsidered
  state NO_HOLD_OFF:
```

```
      if (END_OF_FLASH_READ & DSyncHardReset~.fb) then
         HOLD_OFF_CONSIDERED
      else
         NO_HOLD_OFF;
   state HOLD_OFF_CONSIDERED:
      if (END_OF_OTHER_CYCLE # !DSyncHardReset~.fb) then
         NO_HOLD_OFF
      else
         HOLD_OFF_CONSIDERED;

}

"*****************************************************************************
"* pcc data buffers enable
"*****************************************************************************
equations

PccDataBufEn.oe = 3;

!PccEvenEn~ = (!PccCE1~ # !PccCE2~) & PCC_ENABLED & !HARD_RESET_ASSERTED &
         (STATE_HOLD_OFF_CONSIDERED & !(HOLD_OFF_PERIOD) #
          STATE_NO_HOLD_OFF);
!PccOddEn~ = (!PccCE1~ # !PccCE2~) & PCC_ENABLED & !HARD_RESET_ASSERTED &
         (STATE_HOLD_OFF_CONSIDERED & !(HOLD_OFF_PERIOD) #
          STATE_NO_HOLD_OFF);

"*****************************************************************************
"* pcc data buffers direction
"*****************************************************************************
equations

PccR_W~.oe = H;

PccR_W~ = R_W~;

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

KeepPinsConnected = TA~ ;


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

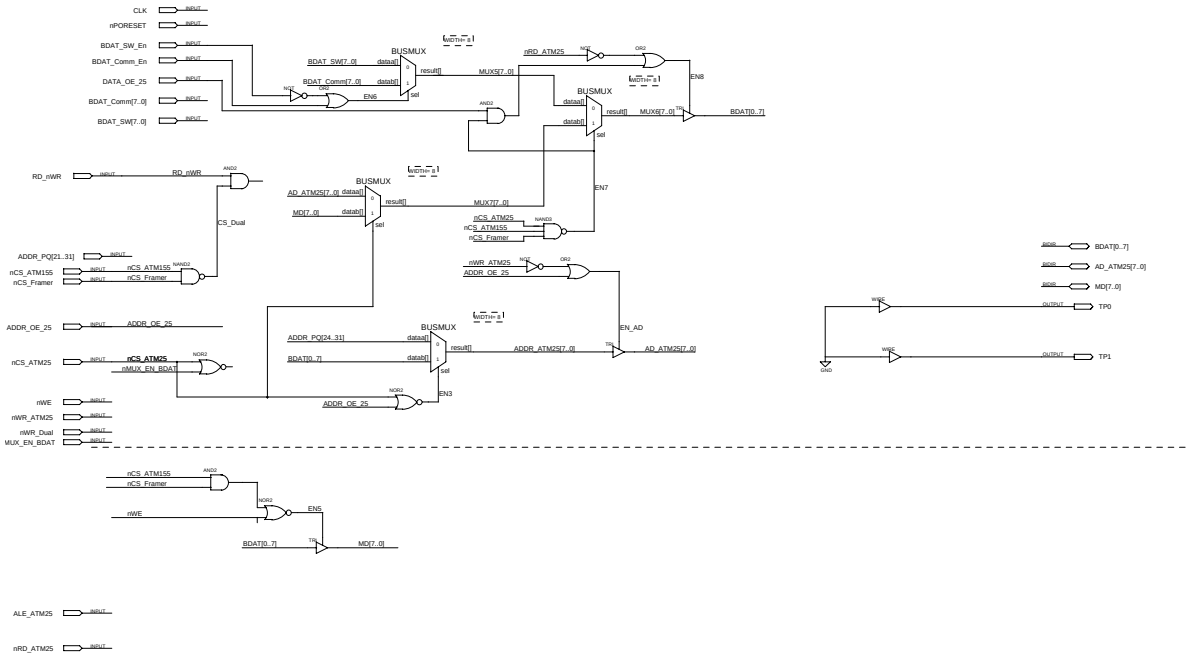
Support Information

2•0•3 BCSR5 & ATM & Fast-Ethernet Control Logic.



Support Information

2•0•4 Block called nux_862/6



TITLE	MPC862DB - MUX_862
COMPANY	Motorola Semiconductor
DESIGNER	Yehuda Palchan
SIZE	D
NUMBER	1.00
REV	144_0.1
DATE	11:17a 12-03-2001
SHEET	2 OF 2

Support Information

This module is one of the Altera internal module called:

```
 module ATM25(
CLK ,
nCS,
nTS,
RD_nWR,
nOE,
nWE,
nPORESET ,
ALE,
nRD,
nWR,
ADDR_OE_25,
DATA_OE,
nMUX_EN_BDAT,
TP0,
TP1
  );

/*****************************************************/
/* Port Declaration  */
/*****************************************************/

input  CLK; /*system clock*/
inputnCS;/*Chip Select */
inputnTS;/*PQ Transfer Start*/
inputRD_nWR; /*PQ Rd/Wr*/
inputnOE;/*PQ Output Enable*/
inputnWE;/*PQ Write Enable*/
input  nPORESET;/*PQ Power On Reset*/

outputALE;/*ATM25 Address Latch Enable*/
outputnRD;/*ATM25 Read*/
outputnWR;/*ATM25 Write*/
outputADDR_OE_25;/*Address to ATM25 Output Enable*/
output DATA_OE;/*BDAT Output Enable*/
output nMUX_EN_BDAT;/*MUX Write BDAT to ATM25*/
output TP0;
output TP1;



reg DATA_OE_Reg;
reg ADDR_OE; //ADDR lines to ATM25 Output Enable

dff D_ALE(.d(!nTS), .clk(CLK), .q(delay_ALE));

always
begin
//Generate ATM25 controls

   ADDR_OE = !nTS || delay_ALE;
//Send Address to ATM25 for an EXTENDED nTS signal//Send Data to ATM25 in whole WR cycle
DATA_OE_Reg = !nOE;
```

```
end

assign DATA_OE = DATA_OE_Reg;
assign ADDR_OE_25 = ADDR_OE; //address oe + hold time.
assignALE = !nTS;
assign nRD = !(!ADDR_OE && !nCS && !nOE);//Assert nRD only one CLK after ALE is done
assign nWR = !(!ADDR_OE && !nCS && !nWE);//Assert nWR only after ALE is done;
assign  nMUX_EN_BDAT = delay_ALE;//MUX Enable BDAT Write to ATM25
assignTP0= delay_ALE;
assignTP1= delay_ALE;


endmodule //ATM25
```

This module is one of the Altera internal module called:

```
module ATM155_Framer(
CLK,
RD_nWR,
nCS_Framer,
nCS_ATM155,
nWE,
nOE,
nWR_ATM155,
nRD_ATM155,
nWR_Framer,
nRD_Framer,
nPORESET);

//Port Declaration
input CLK; //PQ Bus Clock 50MHz
input RD_nWR; //PQ Bus Signal
input nCS_Framer; //T1_E1 Framer Chip Select
input nCS_ATM155; //ATM155 Chip Select
input nWE;//PQ Bus Signal
input nOE;//PQ Bus Signal
input nPORESET;//Power On Reset
output nWR_ATM155;//Write ATM155 command
output nRD_ATM155;//Read  ATM155 command
output nWR_Framer;//Write Framer command
output nRD_Framer;//Read Framer command


assign nWR_ATM155 = nCS_ATM155 || nWE;
assign nRD_ATM155 = nCS_ATM155 || nOE;
assign nWR_Framer = nCS_Framer || nWE;
assign nRD_Framer = nCS_Framer || nOE;

endmodule //ATM155_Framer
```

Support Information

This module is one of the Altera internal module called:

```
module Address_Control (
ADDR,
nCS,
nCS_ATM25,
nCS_ATM155,
nCS_T1_E1,
nCS_ControlReg1,
nCS_ControlReg2);
//Port Declaration
input [23:21] ADDR; //PQ Address
input nCS;//PQ nCS5

output nCS_ATM25;
output nCS_ATM155;
output nCS_T1_E1;
output nCS_ControlReg1;
output nCS_ControlReg2;

assign nCS_ATM25 = !((!nCS)&&(ADDR==0));
assign nCS_ATM155 = !((!nCS)&&(ADDR==1));
assign nCS_T1_E1 = !((!nCS)&&(ADDR==2));
assign nCS_ControlReg1 = !((!nCS)&&(ADDR==3));
assign nCS_ControlReg2 = !((!nCS)&&(ADDR==4));


endmodule //Address_Control
```

This module is one of the Altera internal module called:

```
module COMM_CNTL(
CLK ,
nCS,
RD_nWR,
nPORESET ,
BDAT_IN,
nWE,
nRESET_MII,
nMII_RX_EN,
nRESET_ATM25,
nRESET_ATM155,
nRESET_Framer,
BDAT_OUT,
Comm_Cntl_Rd_En
  );

/*****************************************************/
/* Port Declaration  */
/*****************************************************/

input  CLK; /*system clock */
inputnCS;/*Chip Select  */
inputRD_nWR; /*PQ Rd/Wr*/
```

```
input  nPORESET;/*PQ Power On Reset*/
input [7:0]BDAT_IN;/*PQ Data Bus*/
input nWE;/*PQ Data Bus Write Enable*/

output nRESET_MII;//Reset MII
output nMII_RX_EN;//RX Enable MII
output nRESET_ATM25;//Reset ATM25
output nRESET_ATM155;//Reset ATM155
output nRESET_Framer;//Reset E1_T1 Framer
output [7:0] BDAT_OUT;/*PQ Data Bus*/
output Comm_Cntl_Rd_En; /*Enable BDAT Output TRI*/

reg [7:0] Comm_Cntl_Reg;//Communication Control Register

reg Comm_Cntl_Wr;//Write Comm Reg
reg Comm_Cntl_Rd;//Read Comm Reg

always
begin
Comm_Cntl_Wr = (!nCS)&&(!nWE);
Comm_Cntl_Rd = (!nCS)&&(RD_nWR);
end

always @(posedge CLK)
begin
if (!nPORESET)
begin
Comm_Cntl_Reg = 0;
end
else if (Comm_Cntl_Wr)
begin
Comm_Cntl_Reg = BDAT_IN;
end
end

assign Comm_Cntl_Rd_En = Comm_Cntl_Rd;
assign BDAT_OUT = Comm_Cntl_Reg;
assign nRESET_MII = !(Comm_Cntl_Reg[0] || (!nPORESET));
assign nMII_RX_EN = !(Comm_Cntl_Reg[1] || (!nPORESET));
assign nRESET_ATM25 = !(Comm_Cntl_Reg[2] || (!nPORESET));
assign nRESET_ATM155 = !(Comm_Cntl_Reg[3] || (!nPORESET));
assign nRESET_Framer = !(Comm_Cntl_Reg[4] || (!nPORESET));

endmodule //COMM_CNTL
```

This module is one of the Altera internal module called:

```
module SW_CNTL(
CLK,
RD_nWR,
nPORESET,
nCS,
BDAT,
```

```
nWE,
BDAT_OUT,
IPASEL0,
IPASEL1,
PDSEL0,
PDSEL1,
MUXSEL,
SPARE0,
SPARE1,
SPARE2,
RD_SW_CNTL_En,
CE2,
CE1,
FUNC_SEL2,
FUNC_SEL1,
FUNC_SEL0
);

// Port Declaration
inputCLK;//Bus Clock
inputRD_nWR;//PQ Bus Signal
inputnPORESET;//Power On Reset
input nCS;//Chip Select
input [7:0] BDAT;//PQ Data Bus
input nWE;//PQ Bus signal Write Enable
input CE2;
inputCE1;
inputFUNC_SEL2;
inputFUNC_SEL1;
inputFUNC_SEL0;

output [7:0] BDAT_OUT;//PQ Data Bus
output IPASEL0;
outputIPASEL1;
outputPDSEL0;
outputPDSEL1;
outputMUXSEL;
outputSPARE0;
outputSPARE1;
outputSPARE2;
output RD_SW_CNTL_En;// Control Enable BDAT output

reg [7:0] SW_Control_Reg;//Switch Control Register 7 bit wide
reg WR_SW_CNTL;//Write SW_CNTL command
reg RD_SW_CNTL;//Read SW_CNTL command

always
begin
//WR_SW_CNTL= (!nCS)&&(!RD_nWR);
RD_SW_CNTL= (!nCS)&&(RD_nWR);
end

always @(negedge nWE or negedge nPORESET)
begin
```

```
if (!nPORESET)
begin
SW_Control_Reg =0;
end
else if (!RD_SW_CNTL)
 begin
SW_Control_Reg[0] = CE1;
SW_Control_Reg[1] = CE2;
SW_Control_Reg[2] = FUNC_SEL0;
SW_Control_Reg[3] = FUNC_SEL1;
SW_Control_Reg[4] = FUNC_SEL2;
 end
end

assign RD_SW_CNTL_En = RD_SW_CNTL;
assign BDAT_OUT = SW_Control_Reg;
assign IPASEL0 = (!FUNC_SEL0 && !FUNC_SEL1 && !FUNC_SEL2)  ||
                 (FUNC_SEL0 && !FUNC_SEL1 && !FUNC_SEL2) ||
                 (FUNC_SEL0 && !FUNC_SEL1 && FUNC_SEL2);
assign IPASEL1 = (!FUNC_SEL0 && !FUNC_SEL1 && !FUNC_SEL2) ||
                (!FUNC_SEL0 && !FUNC_SEL1 && FUNC_SEL2) ||
                (!FUNC_SEL0 &&  FUNC_SEL1 && FUNC_SEL2) ||
                (FUNC_SEL0 && !FUNC_SEL1 && FUNC_SEL2);
assign PDSEL1  = !((!FUNC_SEL0 && !FUNC_SEL1 && !FUNC_SEL2) ||
               (!FUNC_SEL0 && !FUNC_SEL1 && FUNC_SEL2));

assign PDSEL0  = (FUNC_SEL0 && FUNC_SEL1 && FUNC_SEL2 ||
                 FUNC_SEL0 && FUNC_SEL1 && !FUNC_SEL2);
assign MUXSEL  = !((!FUNC_SEL0 && FUNC_SEL1 && !FUNC_SEL2) ||
               (!FUNC_SEL0 && FUNC_SEL1 && FUNC_SEL2)); //0; //SW_Control_Reg[4];
assign SPARE0  = SW_Control_Reg[5];
assign SPARE1  = SW_Control_Reg[6];
assign SPARE2  = SW_Control_Reg[7];

endmodule //SW_CNTL
```

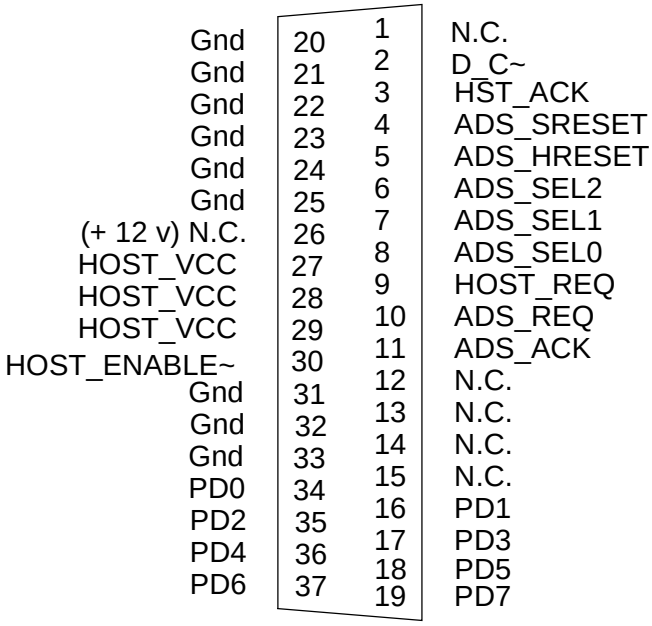
APPENDIX C - ADI I/F

The ADI parallel port supplies parallel link from the MPC86xADS to various host computers. This port is connected via a 37 line cable to a special board called ADI (Application Development Interface) installed in the host computer. Four versions of the ADI board are available to support connection to IBM-PC/XT/AT, MAC II, VMEbus computers and SUN-4 SPARC stations. It is possible to connect the MPC281ADS board to these computers provided that the appropriate software drivers are installed on them.

Each MPC281ADS can have 8 possible slave addresses set for its ADI port, enabling up to 8 MPC281ADS boards to be connected to the same ADI board.

The ADI port connector is a 37 pin, male, D type connector. The connection between the MPC281ADS and the host computer is by a 37 line flat cable, supplied with the ADI board. FIGURE A-1 below shows the pin configuration of the connector.

FIGURE A-1 ADI Port Connector



NOTE: Pin 26 on the ADI is connected to +12 v power supply, but it is not used in the MPC281ADS.

C•1 ADI Port Signal Description

The ADI port on the MPC281ADS was slightly modified to generate either hard reset or soft reset. This feature was added to comply with the MPC's reset mechanism.

In the list below, the directions 'I', 'O', and 'I/O' are relative to the MPC86xADS board. (I.E. 'I' means input to the MPC86xADS)

NOTE:

Since the ADI was originated for the DSP56001ADS some of its signals throughout the boards it was used with, were designated with the prefix "ADS". This convention is kept with this design also.

- ADS_SEL(0:2) - 'I'

These three input lines determine the slave address of the MPC86xADS being accessed by the host computer. Up to 8 boards can be addressed by one ADI board.

- ADS_SRESET - 'I'

  This input line is used to generate Soft Reset for the MPC. When an ads is selected and this line is asserted by the host computer, Soft Reset will be generated to the MPC along with the Soft Reset configuration applied during that sequence.

- HOST_ENABLE~ - 'I'

  This line is always driven low by the ADI board. When an ADI is connected to the MPC86xADS, this signals enabled the operation of the debug port controller. Otherwise the debug port controller is disabled and its outputs are tri-stated.

- ADS_HRESET - 'I'

  When a host is connected, this line is used in conjunction with the addressing lines to generate a Hard Reset to the MPC86xADS board. When this signal is driven in conjunction with the ADS_SRESET signal, the ADI I/F state machines and registers are reset.

- HOST_REQ -'I'

  This signal initiates a host to MPC86xADS write cycle.

- ADS_ACK - 'O'

  This signal is the MPC86xADS response to the HOST_REQ signal, indicating that the board has detected the assertion of HOST_REQ.

- ADS_REQ - 'O'

  This signal initiates an MPC86xADS to host write cycle.

- HST_ACK - 'I'

  This signal serves as the host's response to the ADS_REQ signal.

- HOST_VCC - 'I' (three lines)

  These lines are power lines from the host computer. In the MPC86xADS, these lines are used by the hardware to determine if the host computer is powered on.

- PD(0:7) - 'I/O'

These eight I/O lines are the parallel data bus. This bus is used to transmit and receive data from the host computer.

Support Information

APPENDIX D - ADI Installation

D•1 INTRODUCTION

This appendix describes the hardware installation of the ADI board into various host computers.

The installation instructions cover the following host computers:

1) IBM-PC/XT/AT
2) SUN - 4 (SBus interface)

D•2 IBM-PC/XT/AT to MPC86xADS Interface

The ADI board should be installed in one of the IBM-PC/XT/AT motherboard system expansion slots. A single ADI can control up to eight MPC86xADS boards. The ADI address in the computer is configured to be at I/O memory addresses 100-102 (hex), but it may be reconfigured for an alternate address space.

CAUTION

BEFORE REMOVING OR INSTALLING ANY EQUIPMENT IN THE IBM-PC/XT/AT COMPUTER, TURN THE POWER OFF AND REMOVE THE POWER CORD.

D•2•1 ADI Installation in IBM-PC/XT/AT

Refer to the appropriate Installation and Setup manual of the IBM-PC/XT/AT computer for instructions on removing the computer cover.

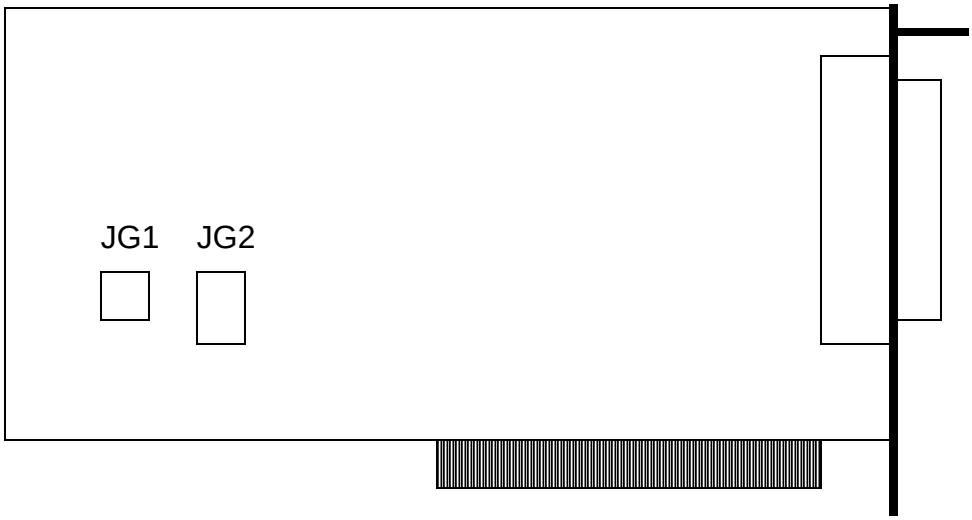
The ADI board address block should be configured at a free I/O address space in the computer. The address must be unique and it must not fall within the address range of another card installed in the computer.

The ADI board address block can be configured to start at one of the three following addresses:

- $100 - This address is unassigned in the IBM-PC
- $200 - This address is usually used for the game port
- $300 - This address is defined as a prototype port

The ADI board is factory configured for address decoding at 100-102 hex in the IBM-PC/XT/AT I/O address map. These are undefined peripheral addresses.

FIGURE B-1 Physical Location of jumper JG1 and JG2



NOTE: Jumper JG2 should be left unconnected.

The following figure shows the required jumper connection for each address configuration. Address 0 hex is not recommended, and its usage might cause problems.

FIGURE B-2 JG1 Configuration Options

To properly install the ADI board, position its front bottom corner in the plastic card guide channel at the front of the IBM-PC/XT/AT chassis. Keeping the top of the ADI board level and any ribbon cables out of the way, lower the board until its connectors are aligned with the computer expansion slot connectors. Using evenly distributed pressure, press the ADI board straight down until it seats in the expansion slot.

Secure the ADI board to the computer chassis using the bracket retaining screw. Refer to the computer Installation and Setup manual for instructions on reinstalling the computer cover.

D•3 SUN-4 to MPC86xADS Interface

The ADI board should be installed in one of the SBus expansion slots in the Sun-4 SPARCstation computer. A single ADI can control up to eight MPC86xADS boards.

Support Information

CAUTION

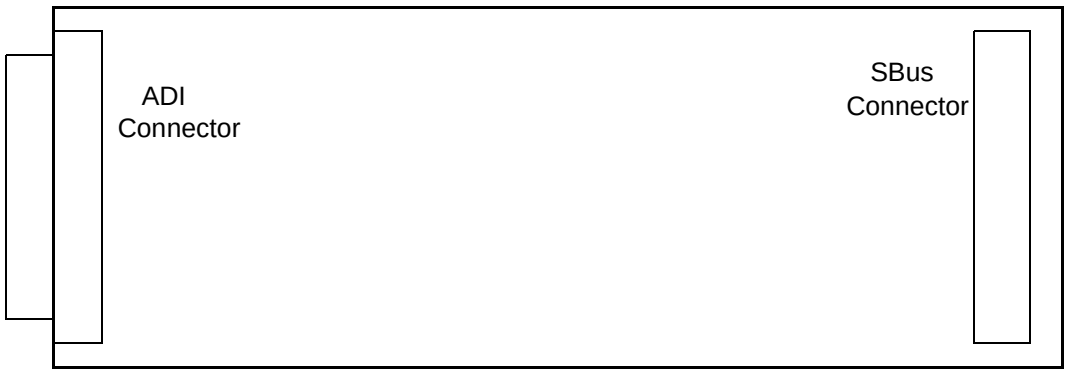
BEFORE REMOVING OR INSTALLING ANY EQUIPMENT IN THE SUN-4 COMPUTER, TURN THE POWER OFF AND REMOVE THE POWER CORD.

D•3•1 ADI Installation in the SUN-4

There are no jumper options on the ADI board for the Sun-4 computer. The ADI board can be inserted into any available SBus expansion slot on the motherboard.

Refer to the appropriate Installation and Setup manual for the Sun-4 computer for instructions on removing the computer cover and installing the board in an expansion slot.

FIGURE B-3 ADI board for SBus



Following is a summary of the Instructions in the Sun manual:

1. Turn off power to the system, but keep the power cord plugged in. Be sure to save all open files and then the following steps should shut down your system:
   - hostname% /bin/su
   - Password: mypasswd
   - hostname# /usr/etc/halt
   - wait for the following messages.

     Syncing file systems... done

     Halted

     Program Terminated

     Type b(boot), c(continue), n(new command mode)
   - When these messages appear, you can safely turn off the power to the system unit.
2. Open the system unit. Be sure to attach a grounding strap to your wrist and to the metal casing of the power supply. Follow the instructions supplied with your system to gain access to the SBus slots.
3. Remove the SBus slot filler panel for the desired slot from the inner surface of the back panel of the system unit. Note that the ADI board is a slave only board and thus will function in any available SBus slot.
4. Slide the ADI board at an angle into the back panel of the system unit. Make sure that the mounting plate on the ADI board hooks into the holes on the back panel of the system unit.

5. Push the ADI board against the back panel and align the connector with its mate and gently press the corners of the board to seat the connector firmly.
6. Close the system unit.
7. Connect the 37 pin interface flat cable to the ADI board and secure.
8. Turn power on to the system unit and check for proper operation.