Order this document by
Preliminary—Not yet stocked

Advance Information

ATMC EVB

ATM Cell Processor Evaluation Board User's Manual

This manual provides information for ATMC EVB, Board Revision ENG (4/1/1998) Revised 11/5/1998

MOTOROLA

Table of Contents

Section 3
Operating Instructions

Section 4
Functional Description

Section 5
Registers Description

Section 6
Support Information

Appendix A
Programmable Logic Equations

Appendix B
Application Development Interface (ADI)

Appendix C
ADI Installation

List of Figures

List of Tables

1 General Information

1.1 Introduction

This document is a User's Manual for the Asynchronous Transfer Mode Cell Processor (ATMC) Evaluation Board (EVB). The manual contains operational and functional descriptions of the ATMC EVB, which is meant to be an evaluation tool for the Motorola MC92500/92501. The MC92500/92501 is an ATM cell processing device implementing ATM layer functions for BISDN. It can serve as a line card for core network switch.

The EVB enables system designers to develop applications for the ATMC. It supports the following applications:

- Software development
- Core network demonstration

1.2 Definitions, Acronyms, and Abbreviations

DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
EDO	Extended Data Out
Kbyte	1024 bytes
LSB	Least Significant Byte
lsb	least significant bit
Mbyte	1048576 bytes
MHz	1000000 cycles/s (hertz)
SIMM	Single In-line Memory Module
TBD	To Be Defined
ATMC	Asynchronous Transfer Mode Cell Processor
EVB	Evaluation Board
PHY	Physical Layer
SONET	Synchronous Optical Network
FPM	Fast Page Mode
Flash	Non volatile reprogrammable memory
ADI	Application Development Interface.

1.3 References

- Motorola MPC860PowerQUICC User's Manual
- Motorola ADI Board Specifications
- Motorola MC92500 or MC92501 Manual
- PMC-Sierra PM5346 S/UNI-155-LITE Manual

1.4 Board Specifications

Table 1-1 list the ATMC EVB specifications.

Table 1-1. ATMC EVB Specification

Characteristics	Specifications
Power Supply	+5 Vdc @ 3.2 A typical, TBD A maximum
Microprocessor	MPC860 or MPC860SAR
ATM Controller	MC92500 or MC92501
Total Address Space	4 Gbyte
Flash Memory Address Space	2 Mbyte, 32-bit wide, expandable up to 8 Mbyte
DRAM Memory Address Space	4 MByte, 36-bit wide SIMM (32-bit data, 4-bit parity), expandable up to 32 MByte
Operation Temperature	0°C–30°C
Dimensions: Height Width Thickness	 220 mm 233 mm 1.6 mm

1.5 ATMC EVB Features

- One, two, or four ATMC External Memory Banks—each bank up to 4 Mbyte of Fast SRAM 72-pin SIMM (1 Mbyte SIMM is supplied with the board)
- 4 K × 64 bits of CAM memory for External Address Compression
- Three sources of Received Ingress traffic—Slave UTOPIA Cell Generator, Fiber-optic ATM line, and External Slave UTOPIA port
- Two sources of Transmit Egress Switch traffic—Master UTOPIA Cell Generator and External Master UTOPIA Egress Switch port
- Three destinations for Transmit Egress traffic—Slave UTOPIA Recorder, Fiber-optic ATM line, and External Slave UTOPIA port
- Two destinations for Receive Ingress Switch traffic—Master UTOPIA Cell Recorder and External Master UTOPIA Ingress Switch port
- External PHY and SWITCH sources/destinations are completely transparent to the MC92500/MC92501
- Master/Slave Traffic Generators are user programmable
- Ingress/Egress interfaces are Octet/Cell-Based UTOPIA level 1 (level 2 is also supported for the MC92501)
- Fiber-optic port processes duplex 155.52 Mbit/s STS-3c/STM-1 or 51.84 Mbit/s STS-1 data streams
- Motorola MPC860PowerQUICC is used to control and operate the EVB (MPC860SAR is optional)
- MPC860 I/O Ports are transparent to the user
- 72-pin DRAM SIMM Socket for optional DRAM expansion—supports 4–32 Mbyte FPM or EDO DRAM SIMM with automatic DRAM SIMM identification (4 Mbyte 60 ns EDO SIMM is supplied with the board)
- 2–8 Mbyte of Flash SIMM, 5 V programmable, with automatic Flash SIMM identification (2 Mbyte 90 ns SIMM is supplied with the board)
- T.P. 10-Base-T Ethernet port via MC68160—EEST on SCC1 with Standby Mode
- RS-232 Serial port on SMC1 with Low-Power Option
- On-board Debug Port Controller with ADI
- Single 5 V power supply
- Hardware/Software reset and Abort push buttons, Status LEDs
- Software Option Switch provides 16 software options via BCSR

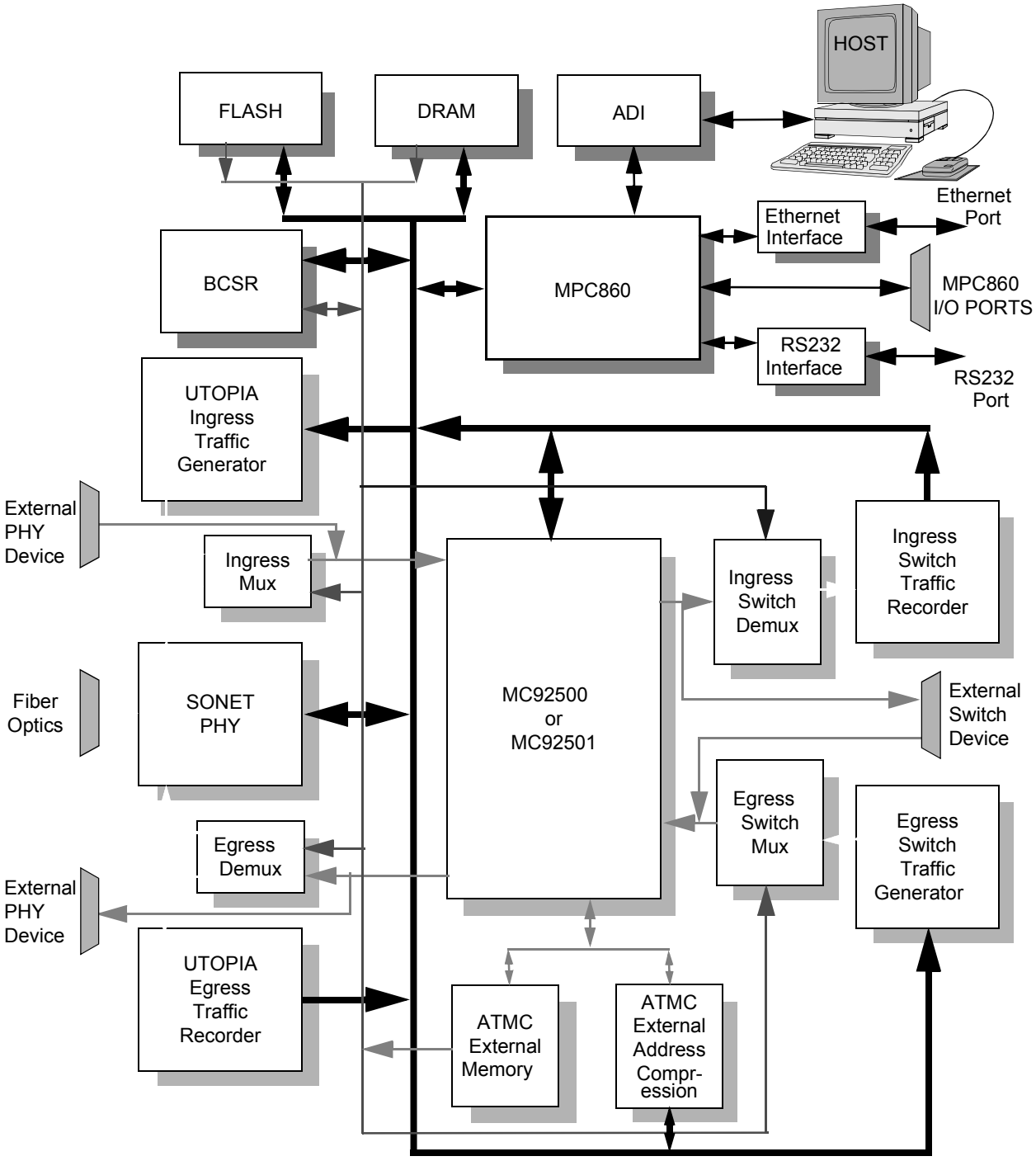


Figure 1-1. ATMC EVB Block Diagram

2

Hardware Preparation and Installation

2.1 Introduction

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

2.2 Unpacking Instructions

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

Because all electronic components are sensitive to the effects of electrostatic discharge (ESD) damage, correct procedures should be used when handling all components on this board and inside the supporting personal computer. Use the following procedures to minimize the likelihood of damage due to ESD:

– **Always handle all static-sensitive** components only in a protected area, preferably a lab with conductive (anti-static) flooring and bench surfaces.

– **Always use grounded wrist straps** when handling sensitive components.

– **Never remove components from** anti-static packaging until required for installation.

– **Always transport sensitive components in anti-static packaging.**

2.3 Hardware Preparations

To select the desired configuration and ensure proper operation of the ATMC EVB, changes of the DIP switch settings may be required before installation. The location of the switches, LEDs, DIP switches, and connectors is illustrated in **Figure 2-1**. The board has been factory tested and is shipped with DIP switches set as described in the following paragraphs. Parameters can be changed for the following conditions:

- ADI port address
- ADI port/SONET PHY Reference Clock Source
- MPC860 Clock Source
- ATMC $\overline{\text{IRQ}}$ allocation
- MC92500 or MC92501 DMA Requests
- MC92500 or MC92501 Word High/Low select
- STXCLK and SRXCLK On-board/Off-board source
- Fiber Optics ATM Line Rate

2.3.1 MC92501 Installation

Before verifying the hardware settings, the MC92501 chip must be installed. In case the board has no chip installed in it or it has to be re-installed or replaced, follow the chip installation instructions carefully.

1. Align the chip pin A1 to the marked corner on the board.
2. Press down on the socket and only then place the chip in the socket.
3. Hold down the chip while applying very light pressure on it with one finger.
4. While holding the chip down, release the pressure from the socket.
5. Visually verify that the chip is seated firmly in the socket.

NOTE: Applying too much pressure or no pressure at all on the chip will result in disconnected pins. Apply just enough pressure to hold the chip down.

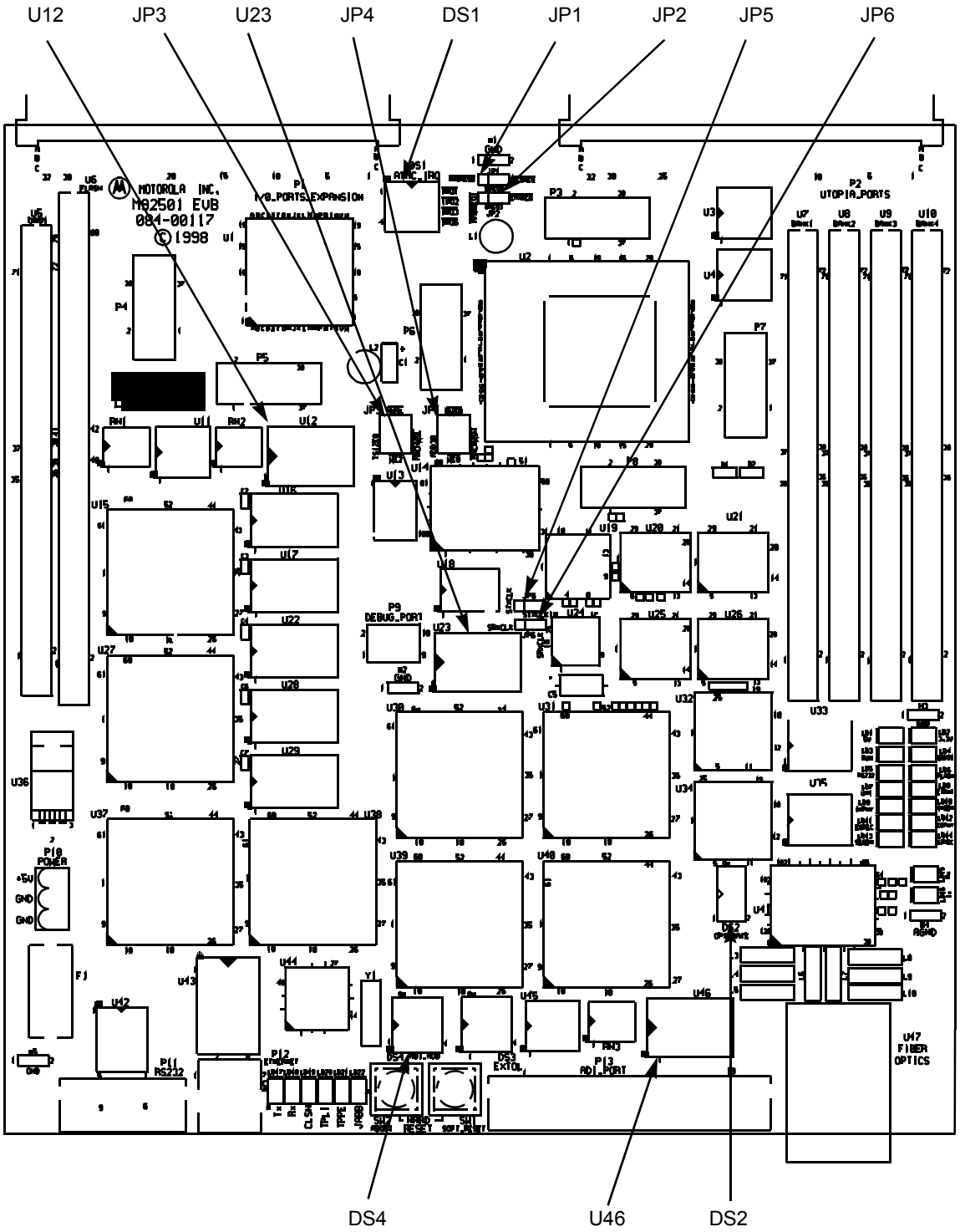


Figure 2-1. MC92501 EVB Top Side Components Location

2.3.2 ADI Port Address Selection

The ATMC EVB can have eight possible slave addresses set for its ADI port, thus allowing up to eight 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, and 3 in the DIP switch DS4. Switch 1 represents the most-significant bit of the address and switch 3 represents the least-significant bit. A switch in the ON state is a logical 1. In **Figure 2-2**, DS4 is configured for address 000.

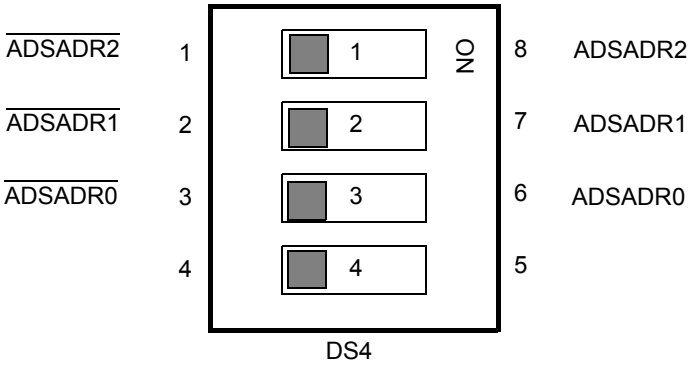


Figure 2-2. ADI Address Configuration—DS4

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
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

2.3.3 ADI Clock Source and SONET PHY Reference Clock—U46

U46 is used both for ADI clock and for the SONET PHY reference clock. It is a **5 V**-only clock generator in an 8- or 14-pin form-factor (refer to **Figure 2-3** for pin allocation). The SONET PHY restricts the clock value to 19.44 MHz.

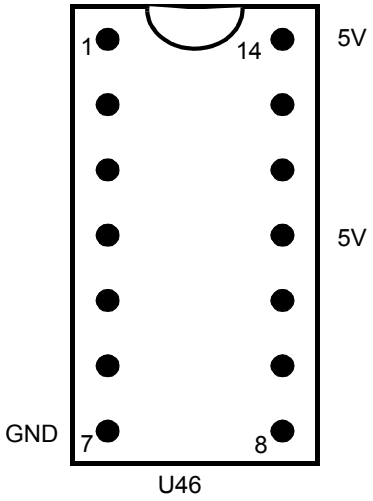


Figure 2-3. U46 Pin Diagram

2.3.4 MPC860 Clock Generator Replacement—U12

U12 is the clock source for the MPC860. It is configured to operate with 5 MHz clock generator. The clock generator is a 3.3 V-only, 5 MHz clock source in an 8- or 14-pin form-factor. **Figure 2-4** shows the pin allocation for U12 (a 4 MHz clock source is also supplied with the board for the purpose of running the software application that is supplied with the board).

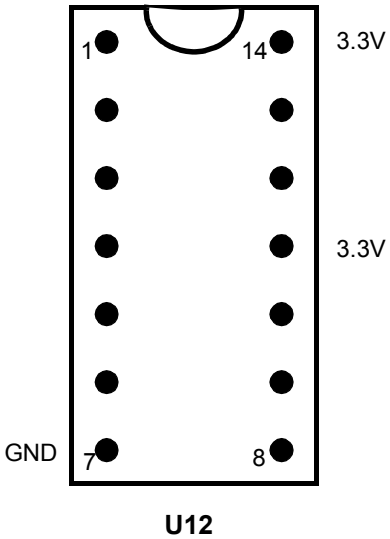


Figure 2-4. U12 Pin Diagram

2.3.5 Generators/Recorders Clock Generator Replacement—U23

U23 is the clock source for the Ingress Generator/Recorder and Egress Generator/Recorder. It is configured to operate with 25 MHz clock generator. The clock generator is a 3.3 V-only, 25 MHz clock source in an 8- or 14-pin form-factor. **Figure 2-5** shows the pin allocation for U23.

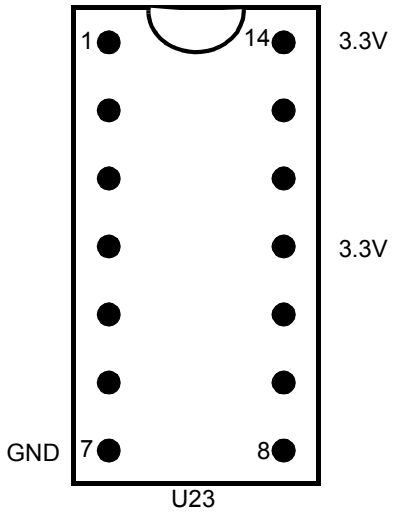


Figure 2-5. U23 Pin Diagram

2.3.6 ATMC $\overline{\text{IRQ}}$ Allocation—DS1

ATMC Interrupt Request can be allocated to four different interrupt levels in the MPC860. This is done with dip-switch DS1. Refer to **Figure 2-6** for details.

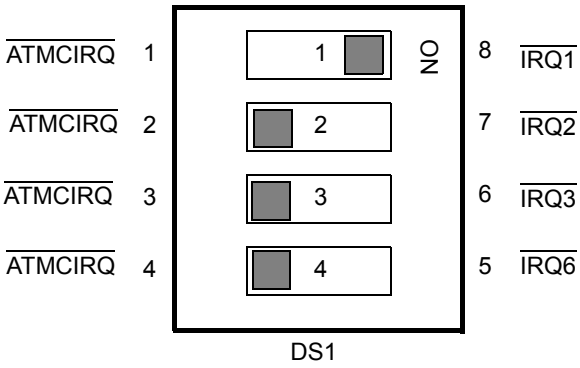


Figure 2-6. ATMC IRQ Settings—DS1

2.3.7 MC92500 or MC92501 DMA Requests Selection

The ATMC has three DMA Request signals: $\overline{\text{EMMREQ}}$, $\overline{\text{MCOREQ}}$ and $\overline{\text{MCIREQ}}$. The MPC860 has only two IDMA Request inputs, therefore some manipulation is needed on the three DMA request signals to transform them to two lines. In the MC92501, this is done internally so $\overline{\text{EMMREQ}}$ and $\overline{\text{MCOREQ}}$ are connected directly to the MPC860 IDMA inputs. If the MC92500 is used, the manipulation is done externally with programmable logic. The selection is done with the two jumpers shown in **Figure 2-7**.

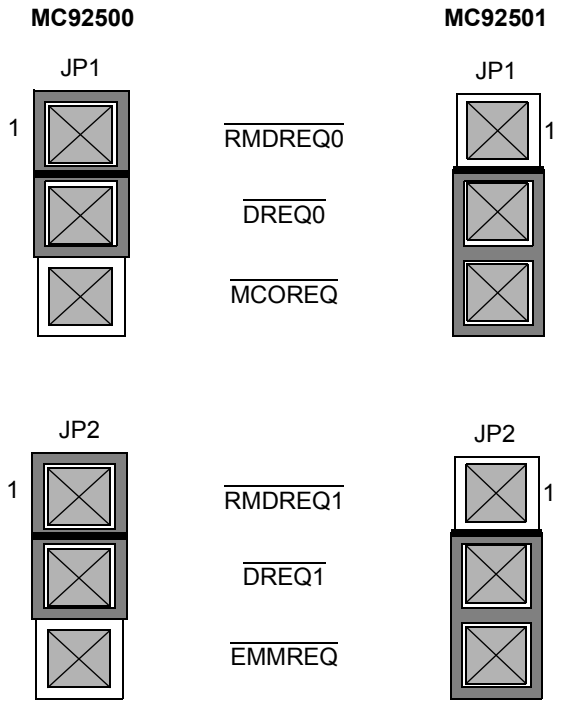


Figure 2-7. MPC860 IDMA Requests Configuration

2.3.8 Word High Select / Word Low Select Configuration

During External Memory access, the ATMC uses two input signals that indicate if the higher 16 bits or the lower 16 bits are being accessed. There are two options to determine which 16 bits are accessed—one for the MC92500 and one for the MC92501. Refer to **Figure 2-8** for details.

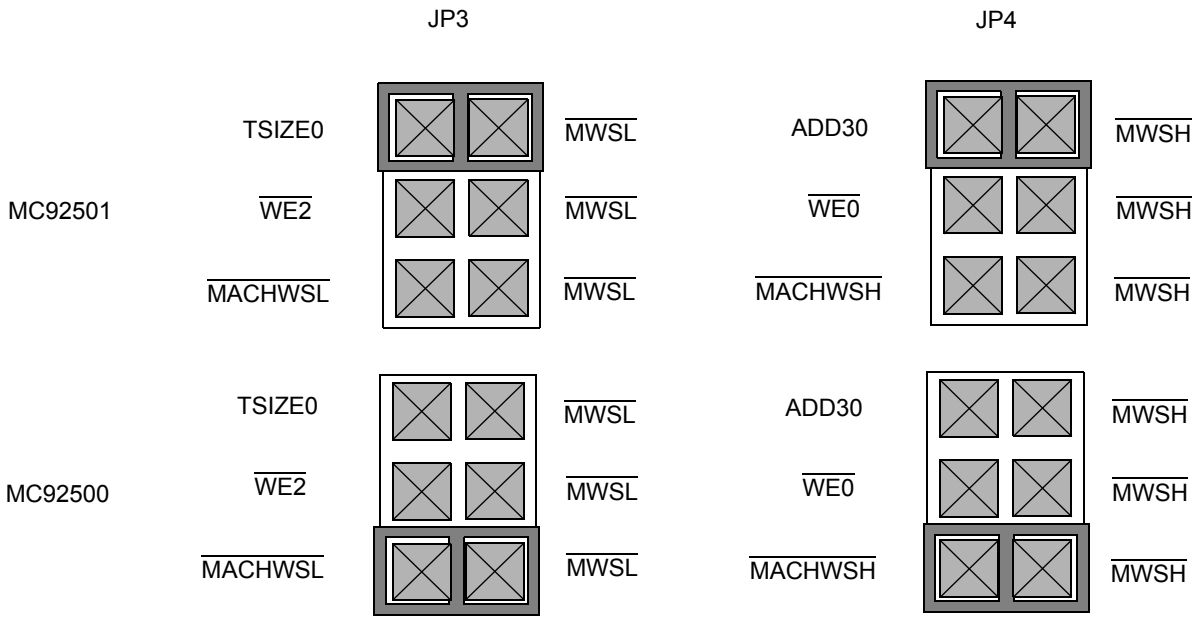


Figure 2-8. Word Select High/Low Configuration

NOTE: If a software application other than the one supplied with the board is used, the MC92501 default configuration is the same as the MC92500 default configuration. Actual functionality also depends on the configuration of the Word Select Signals Mode (WSSM) bit in the Microprocessor Configuration Register (MPCONR) in the ATMC chip. For additional information, refer to the MC92500 or MC92501 User's Manual.

2.3.9 Switch Ingress and Switch Egress Clock Source

The ATMC is driven by two different clocks on the switch side. Switch Egress and Switch Ingress both have individual clock sources. For each one the clock source can be on-board 25 MHz Clock Generator or off-board clock source. This configuration is set by two jumpers. Refer to **Figure 2-9** for details.

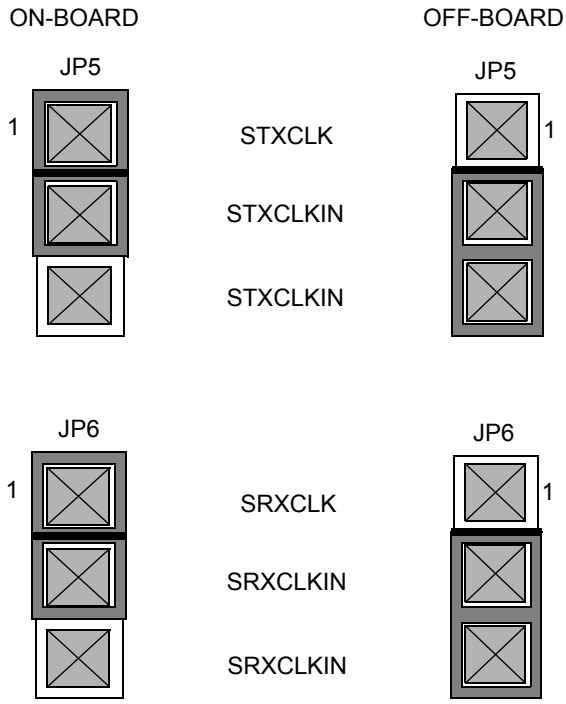


Figure 2-9. SRXCLK and STXCLK Sources

The clock source selection refers only to the ATMC. When off-board clock source is selected, only the ATMC receives that clock; the on-board logic is always driven by the on-board clock generator. The default configuration selects the on-board clock generator.

2.3.10 Fiber Optics ATM Line Rate Configuration

The board has a connection to a Fiber Optics line. The on-board hardware can receive two different rates: 155.52 Mbit/s and 51.84 Mbit/s when using a 19.44 MHz clock generator. Refer to **Figure 2-10** for details.

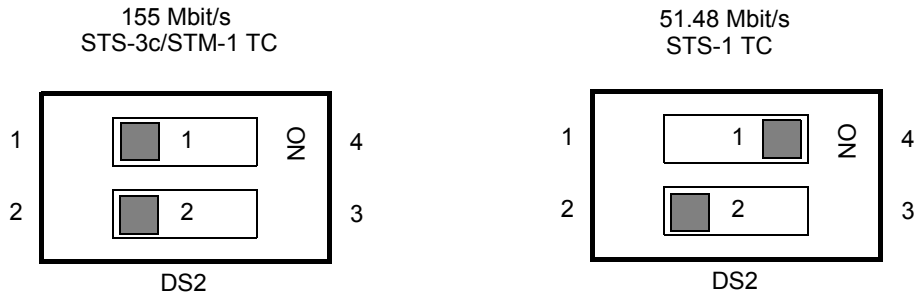


Figure 2-10. Fiber Optics Line Rate Configuration

NOTE: The default setting selects a 155 Mbit/s line rate. It is possible to replace the 19.44 MHz clock generator with a 6.48 MHz clock generator. In that case, there are two other line rates available: 25.92 Mbit/s STS-1 TC and 12.96 Mbit/s STS-1 TC. Replacing the clock generator also changes the Debug clock for the ADI. Refer to the PM5346 Data Sheet for further information.

2.4 Installation Instructions

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

- Host Controlled Operation
- Stand-Alone

2.4.1 Host Controlled Operation

In this configuration, the ATMC EVB is controlled by a host computer via the ADI through the Debug Port. This configuration allows for extensive debugging using the on-host debugger.

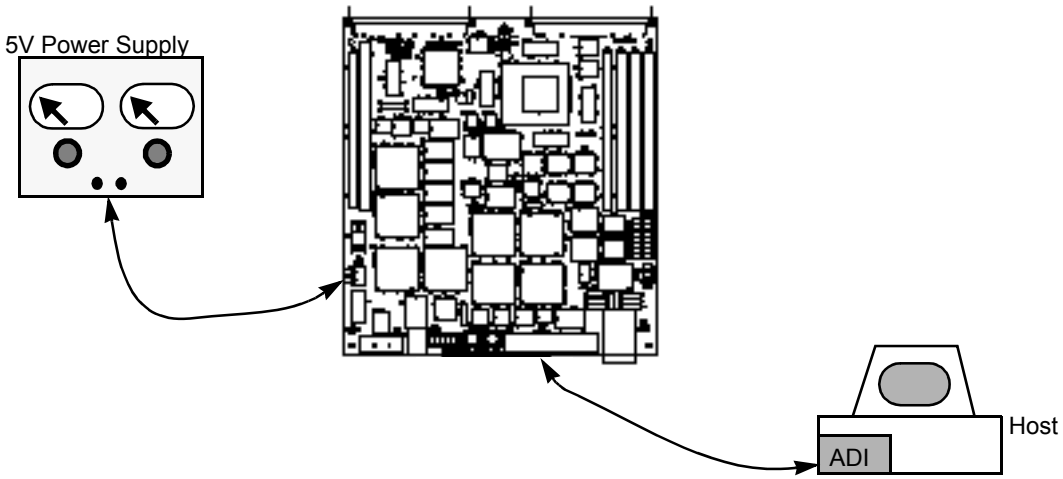


Figure 2-11. Host Controlled Operation Scheme

2.4.2 Stand Alone Operation

In this mode, the board operates using an application program residing in the board Flash memory and does not use the ADI/Debug Port. The board may connect to host via one of its other ports (e.g., RS-232 port, Ethernet port, etc.).

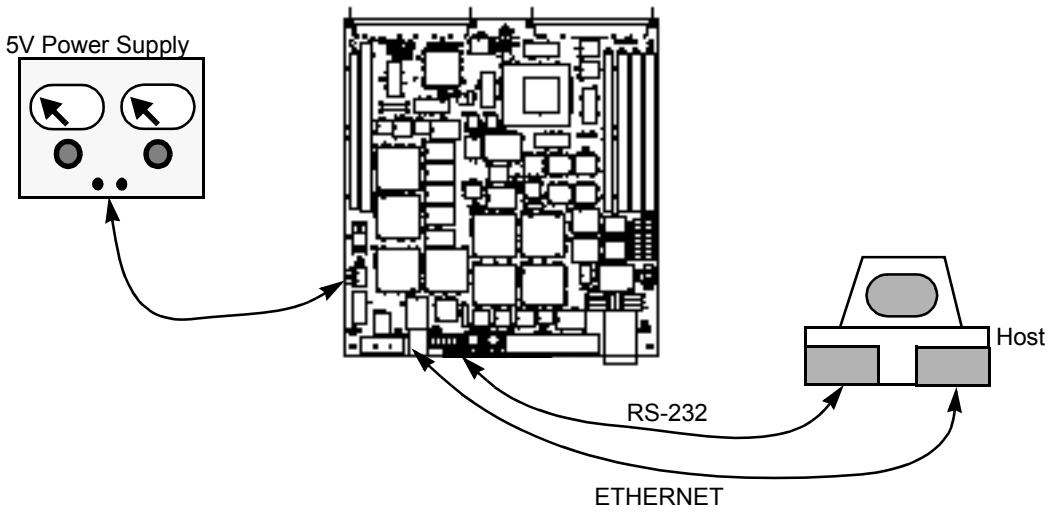


Figure 2-12. Stand Alone Configuration

2.4.3 +5 V Power Supply Connection

The ATMC EVB requires +5 V dc @ 5 A maximum power supply for operation. Connect the +5 V power supply to connector P10 as shown in **Figure 2-13**.

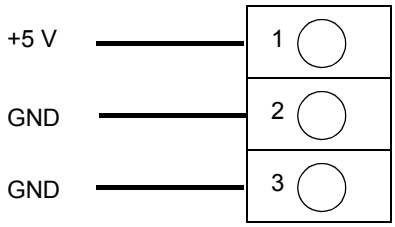


Figure 2-13. P10: +5 V Power Connector

P10 is a three-terminal block power connector with power plug. The plug is designed to accept recommended 14 to 22 AWG wires. To provide solid ground, two ground (GND) terminals are supplied. Connect both GND wires to the common of the power supply. Connect the +5 V to V_{CC} with a single wire.

NOTE: Since hardware applications may be connected to the board using the expansion connectors P1 or P2, consider this additional power consumption when evaluating a power supply connected to the ATMC EVB.

2.4.4 ADI Installation

For ADI installation on various host computers, refer to Appendix C.

2.4.5 Host Computer to ATMC EVB Connection

The EVB ADI interface connector P13 is a 37-pin, male, D-type connector. The connection between the EVB and the host computer uses a 37-line flat cable that is supplied with the ADI board. **Figure 2-14** shows the pin configuration of the connector.

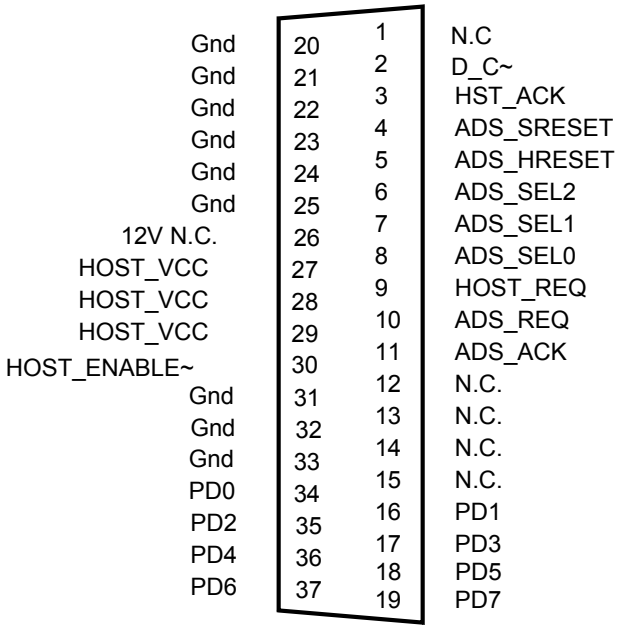
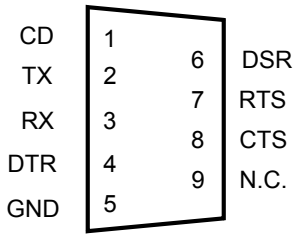


Figure 2-14. P13 ADI Port Connector

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

2.4.6 Terminal to MPC821/860ADS RS-232 Connection

The RS-232 connector P11 is a 9-pin, female, D-type connector as shown in **Figure 2-15**. The connector allows a 1:1 connection to a standard RS-232 serial port via a flat cable.



Note: The RTS line (pin 7) is not connected on the board.

Figure 2-15. P11 RS-232 Serial Port Connector

2.4.7 Memory Installation

The ATMC EVB is supplied with three types of Single-Inline Memory Modules (SIMMs):

- EDO DRAM SIMM
- Flash SIMM
- SRAM SIMM

To avoid shipment damage, these memories are packed aside rather than being installed in their sockets. Therefore, they should be installed on site. To install a SIMM, take it out of its package, place it diagonally in its socket (while the Flash SIMM is 80-pin SIMM, the DRAM and SRAM SIMMs are both 72-pin SIMMs and care must be observed in order not to confuse between them), and then twisted to a vertical position until the metal lock clips are locked—see **Figure 2-16**.

CAUTION

The memory SIMMs have alignment nibble near the # 1 pin. To prevent damage to the SIMM and the socket, make sure that the SIMM is aligned correctly before rotating it into position.

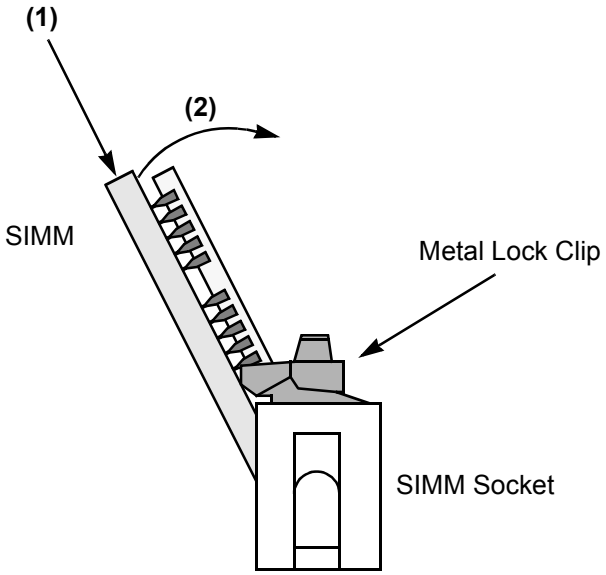


Figure 2-16. SIMM Installation

3

Operating Instructions

3.1 Introduction

This chapter provides information necessary to use the ATMC EVB 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 following sections describe the ATMC EVB switches and indicators.

3.2.1 SW1 and SW2 Switches

Table 3-1 describes the functions of toggle switches SW1 and SW2.

Table 3-1. SW1 and SW2 Functionality

SW1 Status	SW2 Status	Function
Not depressed	Not depressed	Normal operation
Depressed	Not depressed	The EVB sends a software reset to the MPC internal modules, maintaining MPC's configuration (clocks & chip-selects) and DRAM contents. The SW1 switch signal is debounced, and it is not possible to disable it by software. At the end of the Software Reset Sequence, the Software Reset Configuration is sampled and becomes valid.
Not depressed	Depressed	The EVB issues a level 0 interrupt to the MPC. If the ADS is in standalone mode (i.e., detached from a debug station), the user must provide a means of handling the interrupt since there is no resident debugger with the ATMC EVB. The SW2 switch signal is debounced, and can not be disabled by software.
Depressed	Depressed	A Hardware Reset is generated to the MPC. A Hardware Reset causes the MPC to lose all its configuration, including data stored in the DRAM; the MPC must be re-initialized following a Hardware Reset. At the end of the Hardware Reset sequence, the Hardware Reset Configuration stored in BCSR0 becomes valid.

3.2.2 Software Options Switch DS3

DS3 is a four-lever Dual In-line Position (DIP) Switch. This switch is connected over EXTOLI(0:3) lines, and since EXTOLI(0:3) lines are available at BCSR, software options may be manually selected according to DS3 state.

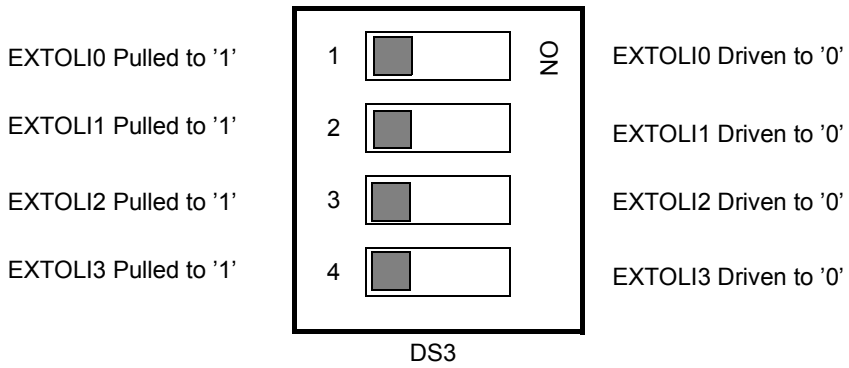


Figure 3-1. DS3 Description

3.2.3 GND Bridges

There are 4 GND bridges on the ATMC EVB. They are meant to assist general measurements and logic-analyzer connection.

CAUTION

To reduce the possibility of permanent damage to the ATMC EVB, use only insulated ground clips when connecting to a GND bridge.

3.2.4 LED Indicators

Table 3-2. LED Indicator Descriptions

Designator	Name	Color	Indication Description
LD1	5 V	Red	When LD1 is lit, the +5 V supply is present at P10.
LD2	3.3 V	Red	When LD2 is lit, the 3.3 V power bus is active.
LD3	RUN	Green	When LD3 is lit, the MPC is not in Debug mode (i.e., VFLS0 and VFLS1 = 0). It is important to remember that, if the VFLS0 and VFLS1 pins are programmed for alternative use (i.e., not as VFLS lines), this indication is meaningless.
LD4	DRAM ON	Green	When LD4 is lit, the DRAM is enabled in BCSR1. Therefore, any access made to CS1 (or CS2) hits on the DRAM. When it is dark, it indicates that the DRAM is disabled in BCSR1.
LD5	Port 1 ON	Green	LD5 is lit to designate that the RS-232 transceiver connected to SMC1, is active and communication via that medium is allowed. When dark, it designates that the transceiver is in shutdown mode, so SMC1 pins may be used off-board via the expansion connector P1.
LD6	Flash ON	Green	When LD6 is lit, the Flash module is enabled in the BCSR1 register (i.e., any access done to the CS0 address space hits the Flash memory). When it is dark, the Flash is disabled.
LD7	UNI ON	Green	When LD7 is lit, the ATMC EVB is configured to handle the User Network Interface (UNI) Switch Fabric. When dark, the board is configured to handle the Node Network Interface (NNI) Switch Fabric.
LD8	ETH ON	Green	When LD8 is lit, the Ethernet port transceiver, the MC68160 EEST connected to SCC1, is active. When it is dark, the EEST is in power down mode, enabling the use of SCC1 pins off-board via the expansion connector P1.
LD9	Ingress SONET PHY Enabled	Yellow	When LD9 is lit, the ATMC is receiving Ingress data from the on-board SONET PHY Optical Interface. When dark, the ATMC receives Ingress data from another source.
LD10	Ingress Genera-tor Enabled	Yellow	When LD10 is lit, the ATMC is receiving Ingress data from the on-board Ingress Generator. When dark, the ATMC receives Ingress data from another source.
LD11	Egress Recorder Enabled	Yellow	When LD11 is lit, the ATMC is transmitting Egress data to the on-board Egress Recorder. When dark, it designates that the ATMC transmits Egress data to another destination.
LD12	Egress SONET PHY Enabled	Yellow	When LD12 is lit, the ATMC is transmitting Egress data to the on-board SONET PHY Optical Interface. When dark, the ATMC transmits Egress data to another destination.

Table 3-2. LED Indicator Descriptions (Continued)

Designator	Name	Color	Indication Description
LD13	Switch Egress Genera-tor Enabled	Yellow	When LD13 is lit, the ATMC is receiving Egress data from the on-board Egress Generator. When dark, the ATMC receives Egress data from an off-board switch fabric.
LD14	Switch Ingress Recorder Enabled	Yellow	When LD14 is lit, it designates that the ATMC is transmitting Ingress data to the on-board Ingress Recorder. When dark, it designates that the ATMC transmits Ingress data to an off-board switch fabric.
LD17	Ethernet TX	Green	LD17 blinks whenever the EEST is transmitting data via the Ethernet port.
LD18	Ethernet RX	Green	LD18 blinks whenever the EEST is receiving data from the Ethernet port.
LD19	Ethernet CLSN	Red	LD19 blinks whenever a collision condition is detected on the Ethernet port (i.e., simultaneous receive and transmit).
LD20	Ethernet LIL	Yellow	LD20 lights to indicate good link integrity on the TP port. The LED is off when the link integrity fails.
LD21	Ethernet PLR	Red	LD21 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.
LD22	Ethernet JABB	Red	LD22 lights whenever a jabber condition is detected on the TP Ethernet port.

3.3 Memory Map

All accesses to ATMC EVB memory slaves are controlled by the MPC memory controller. Therefore, the memory map is reprogrammable to the desire of the user. After a Hardware Reset is performed by the debug station, the debugger checks to see the size, delay and type of the DRAM, SRAM, and Flash memory mounted on board and initializes the chip-selects accordingly. The DRAM and the Flash memory respond to any type of memory access (i.e., user/supervisory/, program/data and DMA). Following is a detailed memory map of the EVB. It is divided to sections according to the on-board elements.

3.3.1 MCP860 Internal Memory Map

Table 3-3. MPC860 Internal Memory Space

Address Range	Memory Type	Device Type	Port Size
$02200000–$02207FFF	MPC Internal MAP		32
$02208000–$027FFFFF	Empty Space		

Note: Refer to the MPC860 User's Manual for complete description of the MPC internal memory map.

3.3.2 CS0—Flash Memory

Table 3-4. CS0—Flash Memory Space

Address Range	Memory Type	Device Type			Port Size
$02800000–$029FFFFF	Flash SIMM	MCM29F020	MCM29F040	MCM29F080	32
$02A00000–$02BFFFFF			MCM29F040	MCM29F080	32
$02C00000–$02FFFFFF				MCM29F080	32
$03000000–$030FFFFF	Empty Space				

3.3.3 CS1—BCSR0–4

Table 3-5. CS1—BCSR0–BCSR4 Memory Space

Address Range	Memory Type	Device Type	Port Size
$02100000–$ 02107FFF	BCSR0–4[a]		32
$02108000–$021FFFFF	Empty Space		

a. The device appears repeatedly in multiples of its size (e.g., BCSR0 appears at memory locations $2100000, $2100014, $2100028, etc., while BCSR1 appears at $2100004, $2100018, $210002C, etc.)

3.3.4 CS2—DRAM SIMM Bank 1

Table 3-6. CS2—DRAM Bank 1 Memory Space

Address Range	Memory Type	Device Type				Port Size
$00000000–$003FFFFF	DRAM SIMM	MCM36100	MCM36200	MCM36400	MCM36800	32

3.3.5 CS3—DRAM SIMM Bank 2

Table 3-7. CS3—DRAM Bank 2 Memory Space

Address Range	Memory Type	Device Type				Port Size
$00400000–$007FFFFF	DRAM SIMM		MCM36200	MCM36400	MCM36800	32
$00800000–$00FFFFFF	DRAM SIMM			MCM36400	MCM36800	32
$01000000–$01FFFFFF	DRAM SIMM				MCM36800	32
$02000000–$020FFFFF	Empty Space					

3.3.6 CS4—SONET PHY

Table 3-8. CS4—PM5346 SONET PHY Memory Space

Address Range	Memory Type	Device Type	Port Size
$02300000–$02307FFF	PM5346 Internal Map[a]		8
$03108000–$031FFFFF	Empty Space		

a. Refer to the PM5346 User's Manual for complete description of the SONET PHY internal memory map.

3.3.7 CS5—ATMC

Table 3-9. CS5—MC92501 ATMC Memory Space

Address Range	Memory Type	Device Type	Port Size
$04000000–$07FFFFFF	MC92501 Internal Map[a]		32
$08000000–$FFFFFFFF	Empty Space		

a. Refer to the MC92501 User's Manual for complete description of the ATMC internal memory map.

3.3.8 CS6—General

Table 3-10. CS6—On Board Generator and Recorder Memory Space

Address Range	Memory Type	Device Type	Port Size
$03200000–$038FFFFF	Generators, Recorders	Receive Ingress Generator, Switch Ingress recorder, Transmit Egress recorder, Switch Egress generator.	32
$03900000–$03AFFFFF	Empty Space		

This memory space is divided into five sections.

3.3.8.1 DMA Request Control Register

Table 3-11. DMA Request Control Register Memory Space

Address Range	Memory Type	Device Type	Port Size
$03380000–$033FFFFF	REQ_CONTROL_REGISTER_ADDR	Determines the type of DMA request: MCIREQ MCOREQ EMMREQ	32

3.3.8.2 Receive Ingress Generator

Table 3-12. Receive Ingress Generator Memory Space

Address Range	Memory Type	Device Type	Port Size
$03200000–$0327FFFF	RPH_FIFO_ADDR	16 K x 16 bit FIFO	32
$03280000–$032FFFFF	RIL_CONTROL_REGISTER_ADDR	Ingress Generator Control Register	32
$03300000 - 0337FFFF	RIL_STATUS_REGISTER_ADDR	Ingress Generator Status Register	32

3.3.8.3 Transmit Egress Recorder

Table 3-13. Transmit Egress Recorder Memory Space

ADDRESS Range	Memory Type	Device Type	Port Size
$03400000–$0347FFFF	TPH_RAM_ADDR	128K x 24 bit SRAM.	32
$03480000–$034FFFFF	TIL_CONTROL_REGISTER_ADDR	Egress Recorder Control Register 1	32
$03500000–$0357FFFF	TPH_RUN_COUNT_REGISTER_ADDR	Egress Recorder Control Register 2	32
$03580000–$035FFFFF	TPH_STOP_COUNT_REGISTER_ADDR	Egress Recorder Control Register 3	32
$03600000–$0367FFFF	TIL_STATUS_REGISTER_ADDR	Egress Recorder Status Register	32

3.3.8.4 Switch Ingress Recorder

Table 3-14. Switch Ingress Recorder Memory Space

Address Range	Memory Type	Device Type	Port Size
$03680000–$036FFFFF	ISW_RAM_ADDR	128K x 16 bit SRAM.	32
$03700000–$0377FFFF	IIL_CONTROL_REGISTER_ADDR	Switch Ingress Recorder Control Register	32

3.3.8.5 Switch Egress Generator

Table 3-15. Switch Egress Generator Memory Space

Address Range	Memory Type	Device Type	Port Size
$03780000–$037FFFFF	ESW_FIFO_ADDR	16K x 16 bit FIFO	32
$03800000–$0387FFFF	EIL_CONTROL_REGISTER_ADDR	Switch Egress Generator Control Register	32
$03880000–$038FFFFF	EIL_STATUS_REGISTER_ADDR	Switch Egress Generator Status Register	32

3.3.9 CS7—CAM

Table 3-16. CS7—MCM69C232 CAM Memory Space

Address Range	Memory Type	Device Type	Port Size
$02400000–$02407FFF	MCM69C232 Internal Map[a]		16
$02408000–$027FFFFF	Empty Space		

a. Refer to the MCM69C232 User's Manual for complete description of the CAM internal memory map.

3.4 Programming the MPC860

The MPC provides the following functions on the board:

- DRAM Controller
- Chip Select generator
- UART (RS-232) for terminal or host computer connection
- Ethernet controller
- Flash Controller

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

NOTE: For more information, refer to the MPC860 User's Manual.

3.4.1 System Interface Unit (SIU) Configurations

These registers have to be programmed immediately after $\overline{\text{HRESET}}$.

Table 3-17. SIU Register Programming

Register	Init Value	Description
SIUMCR	$00632000	• Internal arbitration • External master arbitration priority = 0 • External arbitration priority = 0 • PCMCIA channel II pins = Debug pins • Debug Port on JTAG port pins • FRZ/$\overline{\text{IRQ6}}$ = $\overline{\text{IRQ6}}$ • Debug Register = locked • No parity for non-CS regions • DP0–3/$\overline{\text{IRQ3}}$–$\overline{\text{6}}$ pins = DP0–3 • $\overline{\text{R}}$eservation disabled • $\overline{\text{KR}}$/$\overline{\text{IRQ4}}$/SPK$\overline{\text{ROUT}}$ pin = $\overline{\text{IRQ4}}$ • $\overline{\text{BS}}$_A0–3 and $\overline{\text{WE}}$0–3 are driven just on their dedicated pins • $\overline{\text{GPL}}$_B5 disabled = $\overline{\text{BDIP}}$ enabled • $\overline{\text{GPL}}$_A/B2–3 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 prescaled.
TBSCR	$00C2	No interrupt level, reference match indications cleared, interrupts disabled, no freeze, time-base disabled.
RTCSC	$01C2	Interrupt request level—1, 32768 Hz source, second interrupt disabled, Alarm interrupt disabled, Real-time clock FREEZE, Real-time clock disabled.
PISCR	$0082	No level for interrupt request, Periodic interrupt disabled, clear status, interrupt disabled, FREEZE, periodic timer disabled.

3.4.2 Memory Controller Registers Programming

The memory controller on the MPC860 is initialized to 25 MHz operation (i.e., register programming is based on 25 MHz timing calculation except for refresh timer which is initialized to 16.67 Mhz, the lowest frequency at which the EVB may wake up).

Table 3-18. Memory Controller Initialization for 25 MHz

Register	Device Type	Init Value	Description
BR0	All Flash SIMMs supported.	$02800001	Base at $2800000, 32-bit port size, no parity, GPCM
OR0	MCM29F020-90	$FFE00D20	2 Mbyte block size, all types access, CS early deassert, 2 ws
	MCM29F040-90 SM732A1000A-9	$FFC00D20	4 Mbyte block size, all types access, CS early deassert, 2 ws
	MCM29F080-90 SM732A2000-9	$FF800920	8 Mbyte block size, all types access, CS early deassert, 2 ws, Timing relax
	MCM29F020-12	$FFE00D30	2 Mbyte block size, all types access, CS early deassert, 3 ws
	MCM29F040-12 SM732A1000A-12	$FFC00D30	4 Mbyte block size, all types access, CS early deassert, 3 ws
	MCM29F080-12 SM732A2000-12	$FF800930	8 Mbyte block size, all types access, CS early deassert, 3 ws
BR1	BCSR	$02100001	Base at $2100000, 32-bit port size, no parity, GPCM
OR1	BCSR	$FFFF8110	32 Kbyte block size, all types access, CS early deassert, 1 ws
BR2	All DRAM SIMMs Supported	$00000081	Base at $0, 32-bit port size, no parity, UPMA
OR2	MCM36100/200-60/70	$FFC00800	4 Mbyte block size, all types access, initial address multiplexing according to AMA.
	MCM36400/800-60/70 MT8/16D432/832X-6/7	$FF000800	16 Mbyte block size, all types access, initial address multiplexing according to AMA.
BR3[a]	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	4 Mbyte block size, all types access, initial address multiplexing according to AMA
	MCM36800-60/70 MT16D832X-6/7	$FF000800	16 Mbyte block size, all types access, initial address multiplexing according to AMA.
BR4	SONET PHY	$02300401	Base at $03100000, 8-bit port size, no parity, GPCM
OR4	PM5346	$FFFF8120	32 Kbyte block size, all types access, Internal $\overline{TA}$, SCY=2.

Table 3-18. Memory Controller Initialization for 25 MHz (Continued)

Register	Device Type	Init Value	Description
BR5	ATMC	$04000001	Base at $04000000, 32-bit port size, no parity, GPCM
OR5	MC92501	$FC000790 $FC000798	64 Mbyte block size, all type accesses, Internal $\overline{TA}$ (only for initialization), SCY=9,ACS=11. External $\overline{TA}$ (some accesses may take more than 15 clock cycles, so internal $\overline{TA}$ is used only for the initialization process of the ATMC).
BR6	General	$03200001	Base at $03200000, 32 bit port size, no parity, GPCM
OR6	On-board: • Receive Ingress Generator • Transmit Egress Recorder • Switch Ingress Recorder • Switch Egress Generator	$FF000120	16 Mbyte block size, all types access, Internal $\overline{TA}$.
BR7	CAM	$02400801	Base at $03B00000, 16 bit port size, no parity, GPCM
OR7	MCM69C232	$FFFF870C	32 Kbyte block size, all types access, External $\overline{TA}$, ACS=11, TRLX=1.
MPTPR	All DRAM SIMMs Supported	$0400	Divide by 16 (decimal)
MAMR	MCM36100-60/70	$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.
	MCM36200-60/70	$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.
	MCM36400-60/70 MT8D432X-6/7	$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.
	MCM36800-60/70 MT16D832-6/7	$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.

a. BR3 is not initialized for 36100 or 36400 DRAM SIMMs.

Table 3-19. UPMA Initialization for 60 ns DRAM @ 25 MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset in UPM		**0**	**8**	**18**	**20**	**30**	**3C**
Contents @ Offset +	0	$0FFFCC04	$0FFFCC24	$0FAFCC24	$0FAFCC04	$80FFCC84	$33FFCC07
	1	$08FFCC00	$08FFCC00	$08AFCC00	$08AFCC00	$13FFCC04	X
	2	$33FFCC47	$03FFCC4C	$3FBFCC47	$01AFCC48	$FFFFCC87	X
	3	X	$08FFCC00	X	$08AFCC44	$FFFFCC05	X
	4	X	$03FFCC4C	X	$0FAFCC08	X	
	5	X	$08FFCC00	X	$08AFCC44	X	
	6	X	$03FFCC4C	X	$0CAFCC08	X	
	7	X	$08FFCC00	X	$38BFCC46	X	
	8		$33FFCC47		$FFFFCC45	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 3-20. UPMA Initialization for 70 ns DRAM @ 25 MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset In UPM		**0**	**8**	**18**	**20**	**30**	**3C**
Contents @ Offset +	0	$0FFFEC04	$0FFFCC24	$0FAFCC04	$0FAFCC04	$C0FFCC84	$33FFCC07
	1	$08FFEC04	$0FFFCC04	$08AFCC00	$0CAFCC00	$01FFCC04	X
	2	$00FFEC00	$08FFCC00	$3FBFCC47	$01AFCC4C	$7FFFCC86	X
	3	$3FFFEC47	$03FFCC4C	X	$0CAFCC00	$FFFFCC05	X
	4	X	$08FFCC00	X	$01AFCC4C	X	
	5	X	$03FFCC4C	X	$0CAFCC00	X	
	6	X	$08FFCC00	X	$01AFCC4C	X	
	7	X	$03FFCC4C	X	$0CAFCC00	X	
	8		$08FFCC00		$31BFCC43	X	
	9		$33FFCC47		X	X	
	A		X		X	X	
	B		X		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

Table 3-21. UPMA Initialization for 60 ns EDO DRAM @ 25 MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset in UPM		**0**	**8**	**18**	**20**	**30**	**3C**
Contents @ Offset +	0	$0FFBCC04	$0FFBCC04	$0FEFCC04	$0FEFCC04	$80FFCC84	$33FFCC07
	1	$0CF3CC04	$09F3CC0C	$08AFCC04	$08AFCC00	$13FFCC04	X
	2	$00F3CC00	$09F3CC0C	$00AFCC00	$07AFCC48	$FFFFCC87	X
	3	$33F7CC47	$09F3CC0C	$0FBFCC47	$08AFCC48	$FFFFCC05	X
	4	X	$08F3CC00	X	$08AFCC48	X	
	5	X	$3FF7CC47	X	$39BFCC47	X	
	6	X	X	X		X	
	7	X	X	X		X	
	8		X			X	
	9		X			X	
	A		X		X	X	
	B		X		X	X	
	C		X		X		
	D		X		X		
	E		X		X		
	F		X		X		

Table 3-22. UPMA Initialization for 70 ns EDO DRAM @ 25 MHz

Cycle Type		Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
Offset In UPM		**0**	**8**	**18**	**20**	**30**	**3C**
Contents @ Offset +	0	$0FFBCC04	$0FFBEC04	$0FEFCC04	$0FEFCC04	$C0FFCC84	$33FFCC07
	1	$0CF3CC04	$08F3EC04	$08AFCC04	$08AFCC00	$01FFCC04	X
	2	$00F3CC00	$03F3EC48	$00AFCC00	$07AFCC4C	$7FFFCC86	X
	3	$33F7CC47	$08F3CC00	$0FBFCC47	$08AFCC00	$FFFFCC05	X
	4	X	$0FF3CC4C	X	$07AFCC4C	X	
	5	X	$08F3CC00	X	$08AFCC00	X	
	6	X	$0FF3CC4C	X	$07AFCC4C	X	
	7	X	$08F3CC00	X	$08AFCC00	X	
	8		$3FF7CC47		$37BFCC47	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		

3.5 Programming the ATMC

The ATMC wakes up after Power-On Reset in Maintenance mode with default setting of the registers. The ATMC does not assert DTACK as a default. To change the setting, the Microprocessor Configuration Register (MPCONR) has to be configured as shown in **Table 3-23**. This configuration is optional when external TA is desired. The user may change the setting at any moment (refer to the ATMC User Manual for details). This configuration applies both to the MC92500 and the MC92501.

Table 3-23. MPCONR Programming

Register	Init Value	Description
MPCONR	$04030129	Most Significant bit first (Big Endian), MWSH/A1 functions as A1 and MWSL/SIZE functions as SIZE (depends on JP3 and JP4), MREQ0 functions as MCIREQ (depends on JP1 and JP2), MREQ1 functions as MCOREQ (depends on JP1 and JP2), MREQ2 functions as EMMREQ (depends on JP1 and JP2), only DTACK0 is asserted, MDTACK is asserted with standard timing for Cell Insertion Register write and Maintenance Write Access, during General Register Read accesses MDTACK is asserted as soon as MDATA is valid, during Maintenance Read Access MDTACK is asserted at the MCLK rising edge following the falling edge at which MSEL is detected asserted, MDTACK is asserted during Cell Insertion Register Access, during Cell Extraction Register Access MDTACK is asserted in response to the assertion of MSEL.

3.6 Programming the PM5346

The S/UNI-155-LITE wakes up after reset with default configuration which qualifies to the board needs so no configuration is necessary (refer to the PM5346 User Manual for details).

4

Functional Description

4.1 Introduction

This section describes the design details of the various modules in the ATMC EVB.

4.2 MPC860x

Any chip of the MPC860 family can be installed. The MPC860 runs at frequencies from 15–50 MHz. It should be configured to run at 25 MHz internally and 25 MHz externally. The default configuration is 25 MHz internally and externally. The output clock of the MPC860 is restricted to 33 MHz which is the highest MC92501 frequency.

NOTE: The minimum MPC PLL frequency is 15 MHz. Below that, the Low-Power-Divider must be used, but this yields a CLKOUT that no longer has a 50% duty cycle, distorting the UPM timing.

4.3 Reset Sources

There are several reset sources on the board:

- Main Power On Reset
- Manual Soft Reset
- Manual Hard Reset
- Debug Port Soft Reset
- Debug Port Hard Reset
- MPC Internal Reset Sources

4.3.1 Main Power On Reset

The main power supply generates a Hard Reset and optionally, a Power-On (PON) Reset, when the power is initially supplied to the 3.3 V bus or there is a drop of voltage level over the bus. The reset is generated by a dedicated voltage detector made by Seiko the S-8052ANY-NH-X with detection range of 2.595 to 2.805 V. When regular Power-On Reset conditions exist, the $\overline{\text{HRESET}}$ signal of the MPC is asserted for a period of approximately 4 s. When $\overline{\text{HRESET}}$ is asserted, the Hard Reset Configuration is made available to the MPC. See **Section 4.4.2**.

4.3.2 Manual Soft Reset

To support resident application development and debuggers, a Soft Reset push-button is provided. Depressing that button, asserts the $\overline{\text{SRESET}}$ pin of the MPC, generating a Soft Reset sequence. This button is debounced to avoid spikes over the $\overline{\text{SRESET}}$ line. When $\overline{\text{SRESET}}$ is asserted, the Soft Reset Configuration is made available to the MPC (see **Section 4.4.3**).

4.3.3 Manual Hardware Reset

To support resident application development, a Hard Reset push-button combination is used. To generate a Hard Reset Sequence, press the Soft Reset push-button simultaneous with the ABORT push-button; this asserts the $\overline{\text{HRESET}}$ line. The button sharing saves board space and reduces design cost, but does not affect the functionality in any way.

4.3.4 MPC Internal Reset Sources

The correct operation of the internal reset sources of the MPC is facilitated because the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ lines of the MPC are open-drain and the on-board reset logic drives these lines with open-drain gates. As a rule, an internal reset source asserts $\overline{\text{HRESET}}$ and/or $\overline{\text{SRESET}}$ for a minimum time of 512 system clocks.

4.4 Reset Configuration

During reset sequences to their kinds, 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:

- Power-On Reset Configuration
- Hard Reset Configuration
- Soft Reset Configuration

4.4.1 Power-On Reset Configuration

Just before $\overline{\text{PORESET}}$ is deasserted by the external logic, the Power-On Reset Configuration, which includes the MODCK1 and MODCK2 pins, is sampled. These pins determine the clock operation mode of the MPC. One clock mode is supported within the board: 1:5 PLL operation via the on-board clock generator. In this mode, MODCK1 and MODCK2 are driven high during power on reset (the EVB uses the $\overline{\text{HRESET}}$ signal to drive these pins).

4.4.2 Hard Reset Configuration

During the Hard Reset Sequence, when $\overline{\text{RSTCONF}}$ pin is asserted, the data bus is sampled to acquire the MPCs Hard Reset Configuration. The Reset Configuration Word is driven by the BCSR0, using the defaults set during Power-On Reset. The BCSR0 drives the half configuration word (i.e., data bits D0–D15 in which the reserved bits are designated RSRVxx). If the Hard Reset Configuration must be changed from the default values, BCSR0 may be written with new values, which become valid after Hard Reset is applied to the board.

On the EVB, the $\overline{\text{RSTCONF}}$ line is always driven during Hard Reset (i.e., the MPCs internal

HARD reset configuration defaults can not be used during this time). To allow user programmable, full-word Hard Reset Configuration (i.e., D0–D31 lines being driven during Hard Reset), an option is provided for Flash memory driven Hard Reset Configuration. If this option is used, the desired Hard Reset Configuration Word is taken from the first word of the Flash memory. During Hard Reset, this word drives the data bus to set the desired configuration. To support this option, $\overline{CS0}$ of the MPC should be asserted during Hard Reset and the Address lines should be driven low. Selection of this option is done via BCSR1. See **Table 5-2**.

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

- **Arbitration**—Internal arbitration is selected
- **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.
- **Boot Disable**—Boot is enabled
- **Boot Port Size**—32-bit boot port size is selected
- **Initial Internal Space Base**—Immediately after Hard Reset, the internal space is located at $FF000000
- **Debug Pin Configuration**—PCMCIA port B pins become debug support pins
- **Debug Port Pins Configuration**—Debug port pins are on the JTAG port.
- **External Bus Division Factor**—1:1 internal to external clocks ratio is selected

4.4.3 Soft Reset Configuration

The rising edge of $\overline{SRESET}$ is used to configure the Development Port. Before the deassertion of $\overline{SRESET}$, DSCK is sampled to determine for Debug Mode enable/disable. After $\overline{SRESET}$ is deasserted, if Debug Mode is enabled, DSCK is sampled again for Debug Mode entry/non-entry.

NOTE: DSCK is configured at Hard Reset to reside on the JTAG port.

DSDI is used to determine the Debug Port Clock Mode and is sampled after the deassertion of $\overline{SRESET}$. With parts from the MPC8XX family, DSDI is sampled 3 system clock cycles prior to the deassertion of $\overline{SRESET}$ to determine the part configuration source: internal (default) or external via data bus.

The Soft Reset Configuration is provided by the Debug Port Controller U7 via the ADI I/F. When an ADI bundle is connected (i.e., a Debug station is connected), Debug Mode is always enabled, while immediate entry is determined by the Debug Station. When a bundle is not connected to the ADI port, or disconnected from the host computer, Debug Mode is disabled by means of pulling DSCK low via a pull-down resistor.

4.5 Local Interrupter

The only external interrupt applied to the MPC via its interrupt controller is the ABORT ($\overline{\text{NMI}}$), which is generated by a push-button SW2. When this button is depressed, the $\overline{\text{NMI}}$ input to the MPC is asserted (low). The purpose of this type of interrupt is to support the use of resident debuggers if any is made available to the EVB. All other interrupts to the MPC, are generated internally by the MPC peripherals and by the Debug Port. To support external (off-board) generation of an $\overline{\text{NMI}}$, the $\overline{\text{IRQ0}}$ line which drives the MPC $\overline{\text{NMI}}$, is driven by an open-drain gate. This allows for external hardware to also drive this line. If an external hardware indeed does so, it is compulsory that $\overline{\text{IRQ0}}$ is driven by an open-drain (or open-collector) gate.

4.6 Clock Generator

There is one way to clock the MPC on the ATMC EVB: 3–5 MHz Clock generator connected to CLK4IN input using the 1:5 PLL mode. This clock generator is 3.3-V powered.

4.7 SPLL Support

Since the SPLL requires quiet power supplies, GNDSYN and GNDSYN1 have a dedicated ground plane connected only in one point to the global ground plane of the board. Bypassing capacitor pairs of 0.1 µF and 0.01 µF are connected as close as possible between VDDSYN and GNDSYN. VDDSYN is filtered from the digital supply using a LC filter with a double pole at approximately 500 hz to provide satisfactory attenuation (i.e., approximately -45 dB @ 5 KHz) of switching regulators noise over PLL supply lines.

4.8 Buffering

Since the total capacitive load over the address lines of all local memory slaves is significant, two parallel sets of buffers are provided for address - a dedicated group for the Flash memory, generators, recorders and SONET PHY (U53, U54, U57, and U58) and a dedicated group for the DRAM (part of U11 and U66). Strobe lines are also buffered (U57, U65, and U11) while transceivers are provided for data (U51, U52, U55, U56). The data transceivers open only if there is an access to a valid board address (i.e., an address defined in a Chip Select region and enabled via BCSR1) or during Hard Reset Configuration (i.e., configuration word stored in Flash memory becomes active). That way data conflicts are avoided. The ATMC and the CAM are the only devices that are connected directly to the MPC860 external bus.

4.9 Chip Select Generator

The memory controller of the MPC is used as a chip select generator to access on-board and external memories, saving board area, reducing cost and power consumption, and increasing flexibility. The MPC chip selects are assigned to the various memories/registers on the EVB as follows:

- **CS0**—Flash memory
- **CS1**—BCSR
- **CS2**—DRAM Bank 1
- **CS3**—DRAM Bank 2 (if exists)
- **CS4**—SONET PHY
- **CS5**—ATMC
- **CS6**—Generators/Recorders
- **CS7**—CAM

4.10 DRAM

The ATMC EVB is supplied with 4 Mbyte of EDO DRAM, with an access time of 60 ns. The EVC also supports memory capacities ranging from 4 Mbyte with no parity up to 32 Mbyte with parity. The EVB supports the following Motorola memory devices:

- MCM36100AS60
- MCM36100AS70
- MCM36100ASG60
- MCM36100ASG70
- MCM36100ASH60
- MCM36100ASH70
- MCM36100ASHG60
- MCM36100ASHG70
- MCM36200AS60
- MCM36200AS70
- MCM36200ASG60
- MCM36200ASG70
- MCM36400AS60
- MCM36400AS70
- MCM36400ASG60
- MCM36400ASG70
- MCM36400ASH60
- MCM36800S60
- MCM36800S70
- MCM36800SG60
- MCM36800SG70
- MCM36100ASH70
- MCM36100ASHG60
- MCM36100ASHG70

Also supported are 5 V EDO SIMMs made by Micron:

- MT8D132M-6X (4 Mbyte)
- MT16D232M-6X (8 Mbyte)
- MT8D432M-6X (16 Mbyte)
- MT16D832M-6X (32 Mbyte)
- MT8D432M-7X
- MT16D832M-6X.

All DRAM configurations are supported via the Board Control and Status Register (BCSR) (i.e., DRAM size (4M to 32M) and delay (60/70 ns) 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 2-bank SIMM) region(s) (i.e., RAS and CAS signal generation during normal access—Single Read, Single Write, Burst Read, and Burst Write—as well as during refresh cycles, and the necessary address multiplexing, taking into account support for narrower bus widths, is performed using UPM1). $\overline{CS2}$ and $\overline{CS3}$ signals are buffered from the DRAM and each split to 2 to overcome the capacitive load over the DRAM SIMM RAS lines. The programming of UPM1 and other associated registers to perform that task is described in **Section 3.4.2**. The DRAM module may be enabled/disabled at any time by writing the $\overline{DRAMEN}$ bit in BCSR1. See **Table 5-2**.

4.10.1 DRAM Performance Figures

The performance figures for the DRAM as reflected from the initialization given in **Section 3.4.2** are shown in **Table 4-1** and in **Table 4-2**.

Table 4-1. Regular DRAM Performance in System Clock Cycles at 25 MHz

Operation	DRAM Delay [ns]	
	60	70
Single Read	3	4
Single Write	3	3
Burst Read	3,2,2,2	4,2,2,2
Burst Write	3,1,2,2	3,2,2,2
Refresh	21[1,2]	25[1]

Notes: **1.** Four-beat refresh burst
2. Not including arbitration overhead

Table 4-2. EDO DRAM Performance in System Clock Cycles at 25 MHz

Operation	DRAM Delay [ns]	
	60	70
Single Read	3	4
Single Write	2	3
Burst Read	3,1,1,1	4,1,2,2
Burst Write	2,1,1,1	3,2,2,2
Refresh	13[1,2]	13[1,2]

Notes: **1.** Four-beat refresh burst
2. Not including arbitration overhead

4.10.2 Refresh Control

The DRAM uses a CAS before RAS refresh, which is controlled by UPMA. The refresh logic is clocked by the BRG clock which is not influenced by the low-power divider. As shown in **Figure 4-1**, the BRG clock is divided twice, once by the PTP (Periodic Timer Prescaler) and again by another prescaler the PTA, and is dedicated for each UPM. If there is more than one DRAM bank, refresh cycles are performed for consecutive banks, and therefore, refresh should be made faster.

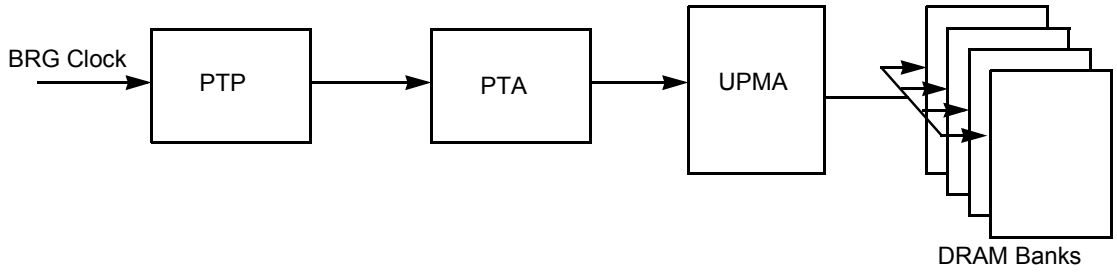


Figure 4-1. DRAM Refresh Scheme

The formula for calculation of the PTA is given below:

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

Where:

- PTA (Periodic Timer A) in MAMR is the value of the second divider.
- Refresh_Period is the time (usually in ms) 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 up to 16).
- Number_Of_Rows_To_Refresh is the number of rows in a DRAM bank.

- T_BRG is the cycle time of the BRG clock.
- MPTPR is the value of the periodic timer prescaler (2 to 64).
- Number_Of_Banks is the number of DRAM banks to refresh.

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

- Refresh_Period = 16 ms
- Number_Of_Beats_Per_Refresh_Cycle: on the ADS it is 4.
- Number_Of_Rows_To_Refresh = 1024
- T_BRG = 40 ns (1/2 system clock @ 50 Mhz)
- MPTPR arbitrarily chosen to be 8
- 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 ($61). The programming of the appropriate registers and UPM's memory, controlling this function, is shown in **Section 3.4.2**.

4.11 Flash Memory

The ATMC EVB supports Flash non-volatile SIMMs of the following types:

- MCM29F020 (2 Mbyte)
- MCM29F040 (4 Mbyte)
- MCM29F080 (8 Mbyte)

These devices are internally composed of 1, 2 or 4 banks of four Am29F040 devices. The Flash SIMM (U6) resides on an 80-pin SIMM socket. Also supported are SMART SM732A1000A 4 Mbyte (1M X 32) or SM732A2000 (2 X 1M X 32) SIMMs.

To minimize use of the MPC chip-select lines, only one chip-select line ($\overline{CS0}$) is used to select the Flash memory 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 installed on the board.

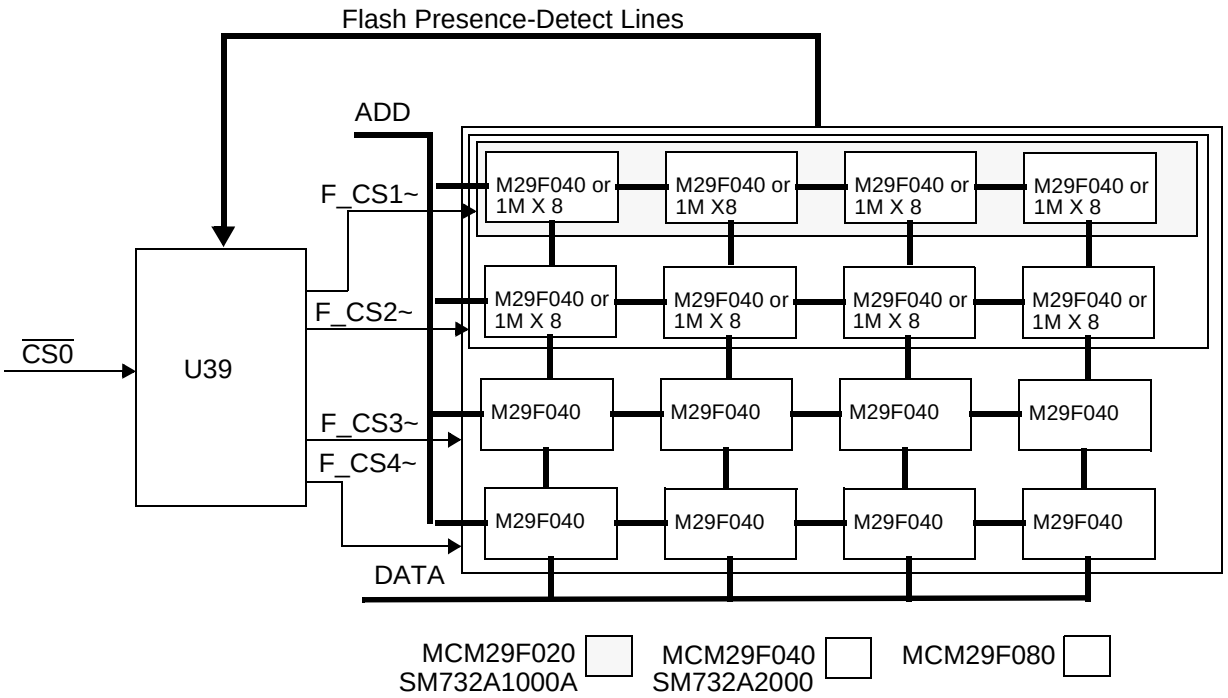


Figure 4-2. Flash SIMM Architecture

The programming of the associated registers is shown in **Section 3.4.2**. The Flash module may be disabled/enabled at any time by writing a 1 or 0, respectively, to the $\overline{\text{FlashEn}}$ bit in BCSR1. The access time of the Flash memory supplied with the ADS is 120 nsec, however, 90 nsec devices may be used. By reading the delay section of the Flash SIMM Presence-Detect lines, the debugger establishes (via OR0) the correct number of wait-states (considering 50 MHz system clock frequency). The Motorola parts which are built of MC29F0X0 devices are 5 V programmable (i.e., there is no need for an external programming voltage).

NOTE: A manufacturer specific dedicated programming algorithm should be implemented during Flash memory programming.

The SMART parts require application of 12V ± 0.5% programming voltage, however. No on-board programming of such devices is supported.

Flash memory control is done using the GPCM and a dedicated $\overline{\text{CS0}}$ region that controls the whole bank. During Hard Reset initializations, the debugger reads the Flash Presence-Detect lines via BCSR2 and decides how to program BR0 and OR0 in which the size and the delay of the region are determined. The performance of the Flash memory is shown in **Table 4-3**.

Table 4-3. Flash Memory Performance in System Clock Cycles at 25 MHz

Operation	Flash Delay [nsec]	
	90	120
Read/Write Access [Clocks]	4	5

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

4.12 Ethernet Port

An Ethernet port with a 10-Base-T interface is provided on the ATMC EVB. This port uses the SCC1of the MPC. A Motorola MC68160 EEST (Enhanced Ethernet Serial Transceiver) is used to mediate between the SCC and the Ethernet medium. To allow external use of SCC1, its pins connect to the expansion connector and the Ethernet transceiver may be Disabled/Enabled at any time by writing a 1 or a 0, respectively, to the $\overline{\text{EthEn}}$ bit in BCSR1.

The EEST is configured constantly to Twisted Pair I/F with automatic polarity correction enabled. There are three control lines that control the EEST function and are driven by the MPC parallel I/O lines:

- Twisted Pair Signal Quality Error Test Enable ($\overline{\text{TPSQEL}}$)—This active-low signal enables testing of the internal TP collision detect circuitry after each transmit to the TP media. It is connected to PC6 of the MPC and should be driven high during normal operation.

NOTE: After Hard Reset, $\overline{\text{TSPQEL}}$ wakes up tri-stated (high impedance). For proper operation, initialize it as an Output.

- Twisted Pair Full Duplex Mode Select ($\overline{\text{TPFLDL}}$)—This active low signal allows simultaneous transmit and receive over the twisted pair lines without indicated collision. This signal is connected to PC5 of the MPC and should be driven low during normal operation.

NOTE: After Hard Reset, $\overline{\text{TPFLDL}}$ wakes up tri-stated (high impedance). For proper operation, initialize it as an Output.

- Diagnostic Loopback (ETHLOOP)—This active high signal puts the EEST in diagnostic loopback mode, regardless of the I/F type it is configured to. This line is connected to PC4 of the MPC and should be driven low during normal operation.

NOTE: After Hard Reset, ETHLOOP wakes up tri-stated (high impedance). For proper operation, initialize it as an Output.

For additional information on the EEST refer to the *MC68160 Technical Data* document.

4.13 RS-232 Ports

To support user applications and to provide a convenient communication channel with a host computer, an RS-232 port is provided via the SMC1 port. This interface supports transmission rates as high as 19200 baud. The EVB uses a MC145707 transceiver which generates RS-232 levels internally using a single 5 V supply and is equipped with OE and shutdown mode. When the RS232EN_1 bit in BCSR1 is asserted, the transceiver is enabled. When deasserted, the transceiver is held in standby mode, in which the receiver outputs are tri-stated, enabling use of the associated SMC port pins off-board via the expansion connector. A standard 9-pin receptacle D-Type connector is used. It is designed to be directly connected (via a flat cable) to a standard IBM-PC like RS-232 connector.

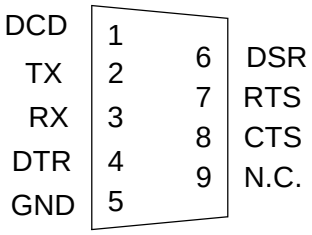


Figure 4-3. RS-232 Serial Port 1 or 2 Connector

In the list below, the directions I, O, and I/O are relative to the EVB (i.e., I means input to the board):

- **CD (O)**—Data Carrier Detect; this line is always asserted by the ATMC EVB.
- **TX (O)**—Transmit Data
- **RX (I)**—Receive Data
- **DTR (I)**—Data Terminal Ready: this signal may be used by the application running on the ATMC EVB to detect if a terminal is connected to the board.
- **DSR (O)**—Data Set Ready: this line is always asserted by the ATMC EVB.

NOTE: Since there are only three RS-232 transmitters available, DSR is connected to CD.

- **RTS (I)**—Request To Send: this line is not connected on the ATMC EVB.
- **CTS (O)**—Clear To Send: this line is always asserted by the ATMC EVB.

5

Registers Description

The EVB consists of several parts that must be initialized and configured before being used. The following sections explain how to configure the EVB and initialize its various elements. The ATMC should not be set to operate until all of these elements are initialized.

5.1 Board Control and Status Registers

Most of the hardware options on the EVB are controlled or monitored by the BCSR, which is a 32-bit wide read/write register (although only 16 bits are available). The BCSR is accessed via the MPC $\overline{CS1}$ region and in fact include five registers: BCSR0 to BCSR4. Since the minimum block size for a CS region is 32 Kbytes, BCSR0–BCSR4 are duplicated inside that region.

The following functions are controlled/monitored by the BCSR:

- MPC Hardware Reset Configuration
- Hardware Reset Configuration Source BCSR0/Flash Memory
- DRAM Type/Size and Delay Identification
- Flash Size/Delay Identification
- ATMC External Memory Banks Number/Size
- External (off-board) tools identification or software option selection switch (DS3) status.
- Board Revision code
- Ethernet port Enable/Disable
- RS-232 port Enable/Disable
- BCSR Enable/Disable
- Flash Module Enable/Disable
- DRAM Module Enable/Disable
- SONET PHY Ingress Port Enable/Disable
- SONET PHY Egress Port Enable/Disable
- Ingress Generator Enable/Disable
- Egress Recorder Enable/Disable
- Ingress Switch Recorder Enable/Disable
- Egress Switch Generator Enable/Disable
- UNI Switch Fabric Type Indication

5.1.1 BCSR Disable Protection Logic

The BCSR itself may be disabled in favor of off-board logic. To avoid accidental disabling of the BCSR (an event that requires a power down cycle to recover), protection logic is provided. The BCSR_EN bit resides in BCSR1. This bit wakes up (active low) during power-up and may not be changed unless the BCSR_EN_PROTECT bit in BCSR3 is 1. Writing a 1 to the BCSR_EN_PROTECT bit allows access to the BCSR_EN bit, but this is a one-shot opportunity to disable the BCSR. Immediately after any write to BCSR1, BCSR_EN_PROTECT is cleared and BCSR_EN is re-protected.

5.1.2 Hardware Reset Configuration Register BCSR0

BCSR0 is located at offset 0 in the BCSR space. It may be read or written at any time, provided the BCSR is not disabled. The system default values are written to BCSR0 during the main power-on reset sequence (i.e., when VDDH is applied to the MPC). During a Hardware Reset, data contained in BCSR0 is driven on the data bus to provide the Hardware Reset Configuration for the MPC, if the Flash_Configuration_Enable bit in BCSR1 is not active. Configuration software may write new values to BCSR0 at any time to change the MPC Hardware Reset Configuration data. The new values become valid when the next Hardware Reset is issued to the MPC regardless of the reset source. **Table 5-1** describes the BCSR0 bits.

Table 5-1. BCSR0 Bit Field Descriptions

Data Bus Bit	Mnemonic	Function	PON Def.	Att
0	ERB	**External Arbitration**—Determines the type of arbitration to be performed: • 0 = Internal • 1 = External	0	R/W
1	IP	**Interrupt Prefix**—Sets the value of the interrupt prefix: • 0 = $0xFFF00000 • 1 = $0	1	R/W
2	Reserved	Implemented[1]	0	R/W
3	BDIS	**Boot Disable**—Determines whether CS0 is Enabled for boot at Hardware Reset: • 0 = enabled • 1 = disabled	0	R/W
4–5	BPS0–BPS1	**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	0	R/W

Table 5-1. BCSR0 Bit Field Descriptions (Continued)

Data Bus Bit	Mnemonic	Function	PON Def.	Att
7–8	ISB0–ISB1	**Initial Space Base**—Determines the initial base address of the internal MPC memory map: • 00 = initial space at $0 • 01 = initial space at $0x00F00000 • 10 = initial space at $0xFF000000 • 11 = initial space at $0xFFF00000	01	R/W
9 - 10	DBGC(0:1)	**Debug Pins Configuration**—Determines the function of the PCMCIA channel II pins: • 00 = PCMCIA channel II pins • 01 = Watch-Points • 10 = Reserved • 11 = Show-cycle attribute pins (e.g., VFLS, VF, etc.)	11	R/W
11-12	DBPC(0:1)	**Debug Port Pins Configuration**—Determines the location of the Debug Port pins: • 00 = Debug Port pins are on the JTAG port • 01 = No Debug Port • 10 = Reserved • 11 = Debug Port is on PCMCIA channel II pins	00	R/W
13 - 14	EBDF(0:1)[2]	**External Bus Division Factor**—Determines the factor upon which the CLKOUT of the MPC external bus, is divided with respect to its internal MPC clock: • 00 = CLKOUT is GCLK2 divided by 1 • 01 = CLKOUT is GCLK2 divided by 2	01	R/W
15	Reserved	Not implemented[1]	0	R/W
16–31	Reserved	Not implemented	—	—

Notes: 1. May be read and written as any other fields and are presented at their associated data pins during Hardware Reset.
2. Applicable for MPC revision A or above. Otherwise have no influence.

5.1.3 Board Control Register 1 BCSR1

BCSR1 is located at offset 4 in the BCSR space. It may be read or written at any time, provided the BCSR is not disabled. The system default values are written to BCSR1 during the main power-on reset sequence (i.e., when VDDH is applied to the MPC). BCSR1 serves as a control register of the EVB. **Table 5-2** describes the BCSR1 fields.

Table 5-2. BCSR1 Bit Field Descriptions

Data Bus 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.	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.	0	R/W
2	ETHEN	**Ethernet Port Enable**—When asserted (low), the EEST connected to SCC1 is enabled. When deasserted (high) that EEST is in standby mode and all its system interface signals are tri-stated.	1	R/W
3	ING_SW_REC_EN	**Ingress Switch Recorder Enable**—When active (low), the on-board Ingress Switch Recorder is enabled.	1	R/W
4	FLASH_CFG_EN	**Flash Configuration Enable**—When this bit is asserted (low): • The Hardware Reset configuration held in BCSR0 is NOT driven on the data bus during Hardware Reset, and • Configuration data started at the first word of the Flash memory is driven to the data bus during Hardware Reset.[1]	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[2] bit on that register.	0	R
6	BCSR_EN	**BCSR Enable**—When this bit is active (low) the Board Control and Status Register is enabled on the local memory map. When inactive, the BCSR may not be read or written. This bit may be written with 1 only if CNT_REG_EN_PROTECT bit is deasserted (1). When the BCSR is disabled it still continues to configure the board according the last data held in it even during Hardware Reset.	0	R/W

Table 5-2. BCSR1 Bit Field Descriptions (Continued)

Data Bus Bit	Mnemonic	Function	PON Def.	Att.
7	IRS232EN	**IRS232 port 1 Enable**—When asserted (low) the RS-232 transceiver is enabled. When deasserted, the RS-232 transceiver is in standby mode and the relevant MPC communication port pins are available.	1	R/W
8	ING_GEN_EN	**Ingress Generator Enable**—When active (low), the on-board Ingress Generator is enabled.	1	R/W
9	ING_PHY_EN	**Ingress PHY Enable**—When active (low), the on_board Ingress SONET PHY is enabled.	1	R/W
10	EG_REC_EN	**Egress Recorder Enable**—When active (low), the on-board Egress Recorder is enabled.	1	R/W
11	EG_PHY_EN	**Egress PHY Enable**—When active (low), the on-board Egress SONET PHY is enabled.	1	R/W
12	Reserved	Not implemented	1	R/W
13	EG_SW_GEN_EN	**Egress Switch Generator Enable**—When active (low), the on-board Egress Switch Generator is enabled.	1	R/W
14	UNI	**User Network Interface**—When active (low), the board is interfaced to a switch fabric which handles UNI protocols. When inactive, the board is interfaced to an NNI switch type.	1	R/W
15–31	Reserved	Not implemented	—	—

Notes: **1.** Provided that this option is supported by the MPC by driving address lines low and asserting $\overline{CS0}$ during Hardware Reset.
2. It is written in BCSR3.

5.1.4 Board Control/Status Register 2 BCSR2

BCSR2 is a read-only status register located at offset 8 in the BCSR space. It may be read at any time, provided the BCSR is not disabled. **Table 5-3** describes the BCSR2 fields.

Table 5-3. BCSR2 Bit Field Descriptions

Data Bus Bit	Mnemonic	Function	PON Def	Att.
0–3	FLASH_PD4–1	**Flash Presence Detect 4–1**—These lines are connected to the Flash SIMM presence detect lines that encode the type of memory mounted in the Flash SIMM socket. There are three additional memory presence detect lines that encode the SIMM delay but appear in BCSR3. For the encoding of FLASH_PD4–1, see **Table 5-4**.	—	R
4	DRAM_EDO	**DRAM Is EDO**—When this bit is active (low), it indicates that the DRAM SIMM is capable of EDO burst read. When inactive, a non-EDO DRAM SIMM is used.	—	R
5–8	DRAM_PD4–1	**DRAM Presence Detect 4–1**—These lines are connected to the DRAM SIMM presence detect lines that encode the size and the delay of the memory mounted in the DRAM SIMM socket. For the encoding of DRAM_PD4–1, see **Table 5-5** and **Table 5-6**.	—	R
9–12	EXTTOLI0–3	**External Tools Identification 0–3**—These lines are intended to serve as tools' identifier or as software option selection. On board software may check these lines to detect the presence of various tools (hardware expansions) at the expansion connectors or the state of a dedicated 4-position DIP switch which resides over the same lines or a combination of both. Half of the available combinations is reserved while the other half is available to users' applications. For the external tools codes and their associated combinations, see **Table 5-7**.	—	R
13–31	Reserved	Not implemented.	—	—

Table 5-4. Flash Presence Detect Encoding

FLASH_PD4–1	Flash Type/Size
$0–$5	Reserved
$6	MCM29080—8 Mbyte SIMM by Motorola
$7	MCM29040—4 Mbyte SIMM by Motorola
$8	MCM29020—2 Mbyte SIMM by Motorola
$9	Reserved
$A	SM732A1000A/SM73218—4 Mbyte SIMM, by SMART Modular Technologies.
$B	SM732A2000/SM73228—8 Mbyte SIMM, by SMART Modular Technologies.
$C–$F	Reserved

Table 5-5. DRAM Presence Detect 2–1 Encoding

DRAM_PD2–1	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 5-6. DRAM Presence Detect 4–3 Encoding

DRAM_PD4–3	DRAM DELAY
00	Reserved
01	Reserved
10	70 ns
11	60 ns

Table 5-7. EXTOOLI0–3 Alignment

EXTTOOLI0–3	External Tool
0000–0111	Reserved
1000–1110	User Available
1111	Non Existent

5.1.5 Board Control/Status Register 3 BCSR3

BCSR3 is an additional control/status register that may be accessed at offset 0xC from the BCSR base address. BCSR3 gets its defaults during Power-On reset and may be read or written at any time, provided the BCSR is not disabled. **Table 5-8** describes the BCSR3 fields.

Table 5-8. BCSR3 Description

Data Bus Bit	Mnemonic	Function	PON Def	Att.
0–4	Reserved	Not implemented	—	—
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 inactive, 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	Not implemented	—	—
8	BREVN0	**Board Revision Number 0**—This is the Most Significant Bit of the Board Revision Number. Refer to **Table 5-9** for Board Revision Number information.	—	R
9–11	FLASH_PD7–5	**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. There are additional 4 presence detect lines which encode the SIMM Type but appear in BCSR2. For the encoding of FLASH_PD7–5, see **Table 5-10**.		
12	BREVN1	**Board Revision Number 1**—The second Most Significant Bit of the Board Revision Number. Refer to **Table 5-9** for Board Revision Number information.	—	R
13	ATMC_EVB	**ATMC Evaluation Board**—When this bit is set, the debugger (software) acknowledges the board as ATMC EVB for the purpose of loading the right version.	1	R
14-15	BREVN2–3	**Board Revision Number 2–3**—The two Least Significant Bits of the Board Revision Number. Refer to **Table 5-9** for Board Revision Number information.	—	R

Table 5-9. ATMC EVB Revision Number

Board Revision Number 0–3	ATMC EVB Revision
$0	ENG (Engineering)
$1	PILOT
$1–$F	Reserved

Table 5-10. Flash Presence Detect 7–5 Encoding

FLASH_PD7–5	Flash Delay [ns]
000	Not Supported
001	150
010	120
011	90
100–111	Not Supported

5.1.6 Board Control/Status Register 4 BCSR4

BCSR4 is a status register accessed at offset 0x10 from the BCSR base address. It is a read only register which may be read at any time, provided the BCSR is not disabled. **Table 5-11** describes the BCSR4 fields

Table 5-11. BCSR4 Description

Data Bus Bit	Mnemonic	Function	PON Def	Att.
0-3	EM_BANK1 3–0	**External Memory Bank1 Presence Detect 3–0**—These lines are connected to Bank1 SRAM SIMM presence detect lines which encode the type of SRAM SIMM mounted on Bank1 SRAM SIMM socket. For the encoding of the SRAM Presence Detect Lines, see **Table 5-12**.	—	R
4-7	EM_BANK2 3–0	**External Memory Bank2 Presence Detect 3–0**—These lines are connected to Bank2 SRAM SIMM presence detect lines which encode the type of SRAM SIMM mounted on Bank2 SRAM SIMM socket. For the encoding of the SRAM Presence Detect Lines, see **Table 5-12**.	—	R
8-11	EM_BANK3 3–0	**External Memory Bank3 Presence Detect 3–0**—These lines are connected to Bank3 SRAM SIMM presence detect lines which encode the type of SRAM SIMM mounted on Bank3 SRAM SIMM socket. For the encoding of the SRAM Presence Detect Lines see **Table 5-12**.	—	R
12-15	EM_BANK4 3–0	**External Memory Bank4 Presence Detect 3–0**—These lines are connected to Bank4 SRAM SIMM presence detect lines which encode the type of SRAM SIMM mounted on Bank4 SRAM SIMM socket. For the encoding of the SRAM Presence Detect Lines see **Table 5-12**.	—	R

Table 5-12. SRAM Presence Detect 3–0 Encoding

EM_BANKx 3–0	SRAM Type/Size
0000–1001	Reserved
1010	1 M × 32 bit; 20 ns or less access time—Cypress CYM1851 or Motorola MCM321024
1011	512 K × 32 bit; 20 ns or less access time—Cypress CYM1846 or Motorola MCM32515
1100	256 K × 32 bit; 20 ns or less access time—Cypress CYM1841A/C
1101–1110	Reserved
1111	No SRAM Module is installed

5.2 Debug Port Controller

The Debug Port of the MPC860 is implemented on-board and connects to the MPC via the Test Access Port (TAP) of the JTAG Boundary Scan Architecture system. Since the location of the Debug Port is determined via the Hardware Reset Configuration, it is important that the relevant configuration bits are not changed when working with the local Debug Port. 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, hardware is used to convert between the parallel and serial protocols. The MPC Debug Port is configured at Software Reset to Asynchronous Clock Mode (i.e., the Debug Port generates the Debug Clock DSCK, which is asynchronous with the MPC system clock).

When the ADI 37-lead cable is disconnected from the EVB connector, the Debug Port controller is disabled, allowing either the connection of an external Debug Port controller, or an independent software run (i.e., the MPC boots from the Flash memory to run user's application without Debug Port controller intervention). This feature is especially handy for running demos. The ADI interface supports up to 8 boards connected in parallel. Address selection is done by means of a 3-position DIP switch.

The Debug Port interface has two registers: a Debug Port Control/Status Register and a Debug Port Data Register. The Debug Port Control/Status Register performs interface related control/status functions, while the Debug Port Data Register serves as the parallel side of the Transmit/Receive shift register. The Control/Status Register is accessed when the D_C bit is low, and the Data Register is accessed when D_C is driven high by the host via the ADI port.

5.2.1 Debug Port Control/Status Register

The Debug Port Control/Status Register is an 8-bit register (bit 7 is the Most Significant Bit). **Table 5-13** describes the Debug Port Control/Status Register fields.

Table 5-13. Debug Port Control/Status Register

Data Bus Bit	Mnemonic	Function	Inter-face Reset Def	Att.
7	MPCRST	**MPC Reset**—This status bit indicates, when active (high), that either a software or a Hardware Reset is driven by the MPC.	—	R
6	TxError	**Transmit Error**—When this status only bit is active (high), the last transmission towards the MPC was cut by an internal PDA reset source. This bit is updated for each byte sent.	—	R
5	InDebug	**In Debug Mode**—When this status only bit is active (high), the MPC is in Debug mode. **Note:** For correct operation, the PCMCIA channel II pins must be configured as Debug pins (i.e, VFLS(0:1) signals are available).	—	R
4- 3	DebugClock-Freq	**Debug Clock Frequency Select**—This field controls a frequency divider that divides DSCK. For the division factors and relative DSCK frequency see **Table 5-14**.	00	R/W
2	StatusRequest	**Status Request**—When the host writes this bit active (low), the interface issues a status read request by asserting the ADS_REQ line to the host. When the host writes the control register with this bit deasserted, no status read request is issued. Upon interface reset, this bit wakes up active.	0	R/W
1	DiagLoopBack	**Diagnostic Loopback Mode**—When this control bit is active (low), the interface 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 interface data register is sampled back into the receive shift register). Using this bit allows to check the interface up to transmit and receive shift registers. Upon interface 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 interface reset, this bit wakes up active.	0	R/W

Table 5-14. DSCK Frequency Select

DebugClockFreq	DSCK Frequency in MHz
00	10
01	5
10	2.5
11	1.25

5.2.2 Standard MPCXXX Debug Port Connector Pin

The pins on the standard Debug Port connector, shown in **Figure 5-1** are the maximal group needed to support Debug Port controller for the MPC8XX families. Some of the pins are redundant for the MPC8XX family but are necessary for the MPC5XX family.

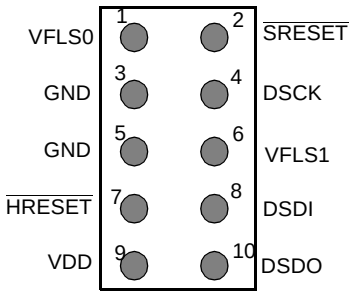


Figure 5-1. Standard Debug Port Connector

Table 5-15. Standard Debug Port Pin Descriptions

Pin Name	Description
VFLS0–1	These pins indicate to the Debug Port controller whether the MPC is in Debug Mode (both VFLS0 and VFLS1 = 1). These lines may serve alternate functions with the MPC but are needed for proper Debug Port operation.
HRESET	This is the Hardware Reset bidirectional signal of the MPC. When this signal is asserted (low), the MPC enters Hardware Reset sequence which includes Hardware Reset configuration. This signal is made redundant with the MPC8XX Debug Port controller since there is a Hardware Reset command integrated within the Debug Port protocol. However, the local debug port controller uses this signal for compatibility with MPC5XX existing boards and software.
SRESET	This is the Software Reset bidirectional signal of the MPC8XX. On the MPC5XX, it is an output. The Debug Port configuration is sampled and determined on the rising-edge of SRESET (for both processor families). The configuration sampling is divided into two 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. On the MPC8XX, SRESET is a bidirectional signal that may be driven externally to generate the Software Reset sequence. This signal is redundant for the MPC8XX Debug Port controller, since there is a Software Reset command integrated within the Debug Port protocol. However, the local Debug Port controller uses this signal for compatibility with MPC5XX existing boards and software.
DSDI	**Debug Port Serial Data In**—Via the DSDI signal, the debug port controller sends its data to the MPC. (See **Section 4.4.3** for its role during a Software Reset Configuration.)
DSCK	**Debug Port Serial Clock**—During asynchronous clock mode, the serial data is clocked into the MPC via the DSCK clock (i.e., DSDI must meat setup/hold time to/from rising edge of the DSCK). (See **Section 4.4.3** for its role during a Software Reset Configuration.)
DSDO	**Debug Port Serial Data Out**—DSDO is clocked out by the MPC via the Debug Port clock, in parallel (i.e., full duplex) 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 command (or data).

5.3 MPC860 DMA Requests

The ATMC has three DMA request signals (i.e., $\overline{\text{MCIREQ}}$, $\overline{\text{MCOREQ}}$ and $\overline{\text{EMMREQ}}$.) while the MPC860 has only two IDMA inputs. The following implementation enables the user to allocate any of the ATMC DMA requests to each of the two MPC860 IDMA inputs (see **Section 2.3.7** for details). The interface is implemented within a programmable logic device and is executed by writing to a control register.

5.3.1 DMA Request Control Register

The control register is a 32-bit register (32-bit port width) in which only 4 bits are used. **Table 5-16** describes the bit fields for the register.

Table 5-16. DMA Requests Control Register

Data Bus Bit	Mnemonic	Function	Def.	Att.
0–1	DREQ1CTL	**DMA Request 1 Control**—These bits determine the selected functionality for the MPC860 DREQ0 input. Any of the three DMA requests can show in DREQ0. For detailed options, see **Table 5-17**.	01	W
2–3	DREQ2CTL	**DMA Request 2 Control**—These bits determine the selected functionality for the MPC860 DREQ1 input. Any of the three DMA requests can show in DREQ1. For detailed options, see **Table 5-17**.	10	W
4–31	reserved	not implemented.	0	W

Table 5-17. DMA Requests Codes

DMA Request Type	Request Code
MCIREQ	00
MCOREQ	01
EMMREQ	10
NO-REQ	11

5.4 PHY Ingress Generator

The on-board Ingress Generator is made from a 16 K × 16-bit asynchronous FIFO, latches, and control logic. The FIFO is first loaded with an ATM traffic simulation file, and then the file is transmitted to the ATMC. The control logic uses a control register and a status register to control the operation of the generator. The user should first load the FIFO with data by issuing and writing to the proper address.

5.4.1 PHY Ingress Generator Control Register

The 32-bit PHY Ingress Generator Control Register uses only 3 bits. **Table 5-18** describes its fields.

Table 5-18. Ingress Generator Control Register

Bit	Mnemonic	Function	Def	Att
0	SRPIM	**Receive PHY Interface Mode**—When active (high), the generator is in run mode. When inactive (low), the generator loads the ATM traffic simulation file from MPC860.	0	W
1	$\overline{\text{SRFOE}}$	**Receive PHY Output Enable**—When active (low), the generator is connected to the ATMC. When inactive (high), the generator outputs are tri-stated (i.e., disconnected from the ATMC).	1	W
2	SRRST	**Receive PHY Reset**—When active (high), the generator is in RESET state (i.e., the FIFO read and write pointers are reset). When inactive (low), the generator is in Run Mode (depends on RPIM).	1	W
3–31	reserved	not implemented.	0	—

5.4.2 PHY Ingress Status Register

The 32-bit PHY Ingress Generator Status Register uses only 3 bits. **Table 5-19** describes its fields.

Table 5-19. Ingress Generator Status Register

Bit	Mnemonic	Function	Def	Att
0	RENG	**Receive PHY $\overline{\text{RXENB}}$ Deasserted**—When set (high), $\overline{\text{RXENB}}$ was deasserted at least once since last reset.	—	R
1	RSTP	**Receive PHY Stop**—When set (high), the generator is in a STOP state after being in TRANSMIT state.	—	R
2	RIDL	**Receive PHY Idle**—When set (high), the generator is in IDLE state.	—	R
3–31	reserved	not implemented	—	—

5.4.3 Ingress Traffic Emulation File Format

The file that is loaded to the FIFO emulates ATM traffic. The file includes all the control and data signals in UTOPIA format that the ATMC requires for proper operation. The data structure of a single line (16 bits) in the file is as follows:

Table 5-20. Ingress Traffic Emulation File Data Structure

Data Bus Bit	Mnemonic	Function
0–6	RXDATA7–1	**Receive Data Lines 7–1**—These lines contains the data.
7	RXDATA(0)	**Receive Data Line (0)**—This line is the lsb data line. It is used in conjunction with RX_ESC to generate control signals for the generator operation. Depending on RX_ESC and the state in which the generator is in and when RXDATA(0) is: • 1 = RETRANSMIT Command • 0 = STOP Command.
8–11	RXPHYID3–0	**Receive PHY Identification Number**—The ATMC supports up to 16 PHY sources and these 4 bits indicate the source that is currently connected to the ATMC. It is asserted simultaneously with the data. This feature supports only UTOPIA Level 1.
12	RXPRTY	**Receive PHY Parity**—When set (high), it indicates odd parity over the receive data lines. It is asserted simultaneously with the data.
13	$\overline{\text{B\_RXEMPTY}}$	**Buffered Receive Empty**—When active (low), the PHY source has no data available to the ATMC. It is used by the control logic to generate the $\overline{\text{RXEMPTY}}$ signal. When asserted, the data over the data lines in the next clock is ignored. When the interface is octet-based, $\overline{\text{B\_RXEMPTY}}$ can be asserted after every octet. When the interface is cell-based, it can be asserted only after the last octet of the cell. The generation of $\overline{\text{RXEMPTY}}$ takes one clock, therefore $\overline{\text{B\_RXEMPTY}}$ should be asserted one clock before $\overline{\text{RXEMPTY}}$ is expected to be asserted. $\overline{\text{RXEMPTY}}$ will be valid two clocks after the assertion of $\overline{\text{B\_RXEMPTY}}$.
14	RXSOC	**Receive Start Of Cell**—When active (high), the data on the data lines is the first octet of the cell. It should be asserted only with the first octet and deasserted with the second octet until the first octet of the next cell.
15	RX_ESC	**Receive Escape**—When active (high) and when the generator is in run mode and transmitting data to the ATMC, this bit in conjunction with RXDATA(0) will generate either a STOP or RETRANSMIT commands. It has to be asserted together with the RXDATA after the last octet of the last cell.

5.4.4 Ingress Generator State Machine

Ingress generator operation is based on a state machine. The generator state depends on the previous state and the control signals that are generated by the ATMC, the traffic emulation file, and the user. The state machine is activated only when the generator is moved to Run Mode. The algorithm of the state machine is shown in **Figure 5-2**.

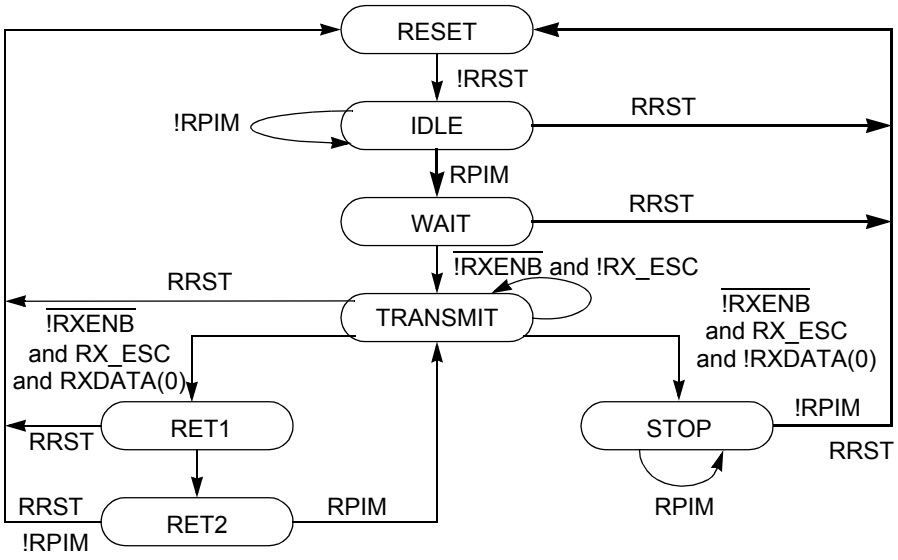


Figure 5-2. Ingress Generator State Machine

Table 5-21. Ingress Generator State Definitions

State	Definition
RESET	The read and write pointers of the FIFO are reset and point to the beginning. This state is accessible from every state by setting RRST (high) by either a board HRESET or by setting SRRST in the generator control register. The generator stays in this state until RRST is deasserted. It then moves to IDLE state.
IDLE	This is the state after the reset. The generator stays in this state if no signal is asserted. If RRST is detected, the generator moves back to RESET state, and moves to WAIT state only when the user asserts RPIM (i.e., the generator is loaded with traffic emulation data).
WAIT	The generator is enabled to start transmitting ($\overline{\text{RREN}}$ is activated, i.e., the FIFO is ready to output the data) and moves to TRANSMIT state without conditions. RRST detection moves the generator to the RESET state.
TRANSMIT	The generator waits for $\overline{\text{RXENB}}$. When detected, the generator transmits the data. Detection of RRST moves the generator to the RESET state. Detection of RX_ESC and RXDATA(0) active (high) while $\overline{\text{RXENB}}$ is asserted moves the generator to the RET1 state. Detection of RX_ESC and RXDATA(0) inactive (low) while $\overline{\text{RXENB}}$ is asserted moves the generator to the STOP state. While $\overline{\text{RXENB}}$ is asserted and RX_ESC is inactive (low), the generator stays in TRANSMIT state until it reaches the end of the FIFO. When changing to another state, the generator transmission is disabled ($\overline{\text{RREN}}$ is inactive).
STOP	The generator is halted. It stays in this state as long as RPIM is active (high) and RRST is inactive (low). When RPIM is deasserted (low), the generator moves to the RESET state.
RET1	This is a transfer state. The generator moves unconditionally to the RET2 state. If RRST is detected, it moves to the RESET state.
RET2	This is the RETRANSMIT state. If the generator is still in run mode (RPIM active) it moves back to the TRANSMIT state to the beginning of the FIFO. Detection of RRST or RPIM inactive moves the generator to the RESET state.

5.5 Switch Egress Generator

The on-board Switch Egress Generator is made from a 16 K× 16-bit asynchronous FIFO, latches, and control logic. The FIFO is first loaded with an ATM traffic simulation file, and then the file is transmitted to the ATMC. The control logic uses a control register and a status register to control the operation of the generator. The user should first load the FIFO with data by issuing the proper address and writing the data out to the FIFO.

5.5.1 Switch Egress Generator Control Register

The 32-bit Switch Egress Generator Control Register only uses 3 bits. **Table 5-22** describes is fields.

Table 5-22. Switch Egress Generator Control Register

Bit	Mnemonic	Function	Def	Att
0	SESIM	**Egress Switch Interface Mode**—When active (high), the generator is in run mode. When inactive (low), the generator loads the ATM traffic simulation file from MPC860.	0	W
1	$\overline{\text{SEFOE}}$	**Egress FIFO Output Enable**—When active (low), the generator is connected to the ATMC. When inactive (high), the generator outputs are tri-stated (i.e., disconnected from the ATMC).	1	W
2	SERST	**Egress Switch Reset**—When active (high), the generator is in reset mode, i.e. the FIFOs' read and write pointers are reset. When inactive (low), the generator is in run mode (depends on SESIM).	1	W
3–31	reserved	not implemented.	0	—

5.5.2 Switch Egress Generator Status Register

The 32-bit Switch Egress Generator Status Register uses only 2 bits. **Table 5-23** describes its fields.

Table 5-23. Switch Egress Generator Status Register

Bit	Mnemonic	Function	Def	Att
0	EIDL	**Egress Switch Generator Idle**—When set (high), the generator is in idle state waiting for SESIM.	—	R
1	ESTP	**Egress Switch Generator Stop**—When set (high), the generator is in a STOP state after being in a TRANSMIT state.	—	R
2–31	reserved	not implemented.	—	—

5.5.3 Egress Traffic Emulation File Format

The file that is loaded to the FIFO emulates ATM traffic. The file includes all the control and data signals in UTOPIA format that the ATMC requires for proper operation. The data structure of a single line (16 bits) in the file is as follows:

Table 5-24. Switch Egress Traffic Emulation File Format

Data Bus Bit	Mnemonic	Function
0–6	STXDATA7–1	**Switch Transmit Data Lines 7–1**—These are the switch Egress data lines.
7	STXDATA0	**Switch Transmit Data Line 0—**This line is the 8th data line. It is used in conjunction with EX_ESC to generate control signals for the generator operation. Depending on EX_ESC and the state in which the generator is in and when RXDATA(0) is: • 1 = RETRANSMIT Command • 0 = STOP Command.
8	ECCLV	**Egress Consider Cell Available**—Since STXCLAV is deasserted before the end of the cell, and the TRANSMIT state depends on STXCLAV, ECCLV is set (high) one octet before the end of the cell (cell based interface). In that case when STXCLAV is detected deasserted (low) and ECCLV is asserted one octet before the end of the cell, the generator will move to WAIT state at the end of the cell.
9–11	reserved	Not implemented
12	STXPRTY	**Egress Switch Generator Parity**—When set (high), it indicates even parity over the data lines. When set low, it indicates odd parity. The ATMC can be configured to either parity. It is asserted simultaneously with the data.
13	$\overline{\text{B\_STXENB}}$	**Buffered Egress Generator Enable**—When active (low), this bit indicates that the generator is transmitting data. When deasserted, it indicates it is the end of the cell, or that the switch outputs a blank (the switch has no octet ready for the ATMC). This bit is used to generate the $\overline{\text{STXENB}}$ and should be negated with the last octet of the cell (to indicate end of cell), or with the last octet before blank octets. It should be asserted one clock before the desired octet—either the first cell after a blank or the first octet of a new cell. $\overline{\text{STXENB}}$ will be valid two clocks after assertion of $\overline{\text{B\_STXENB}}$
14	STXSOC	**Switch Egress Start Of Cell**—When active (high), the data on the data lines is the first octet of the cell. It should be asserted only with the first octet and deasserted with the second octet until the first octet of the next cell.
15	EX_ESC	**Switch Egress Escape**—When active (high) and when the generator is in run mode and transmitting data to the ATMC, this bit in conjunction with RXDATA(0) will generate either a STOP or RETRANSMIT commands. It has to be asserted together with the RXDATA after the last octet of the last cell.

5.5.4 Switch Egress Generator State Machine

The Switch Egress Generator is based on a state machine. The state machine is very similar to the Ingress Generator state machine but with two additions. The interface is cell based and ECCLV is used. STXCLAV is deasserted at least 4 clocks before the end of the cell, therefore ECCLV is asserted one octet before the last octet of the cell. If STXCLAV is detected deasserted and ECCLV is asserted, the FIFO outputs the last octet and stops. The algorithm of the state machine is shown in **Figure 5-3**.

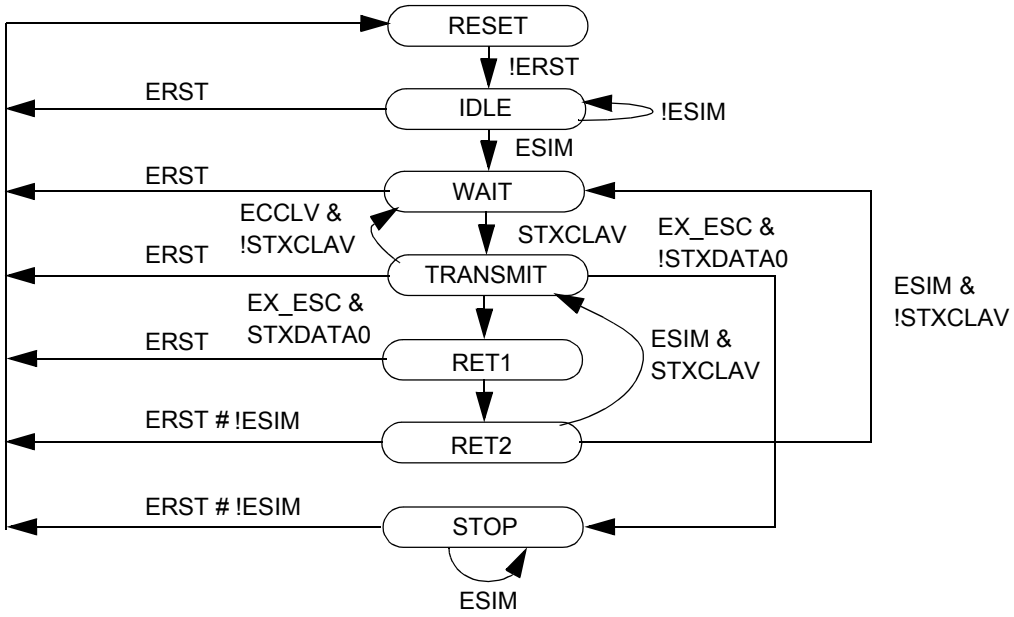


Figure 5-3. Switch Egress Generator State Machine

Table 5-25. Engress Generator State Definitions

State	Definition
RESET	The read and write pointers of the FIFO are reset and point to the beginning.This state is accessible from every state by setting ERST (high) either by a board HRESET or by setting SERST in the generator control register. The generator stays in this state until ERST is deasserted. It then moves to IDLE state.
IDLE	This is the state after the reset. The generator stays in this state if no signal is asserted. If ERST is detected, the generator moves back to the RESET state. It moves to the WAIT state only when the user asserts ESIM (i.e., the generator is loaded with the traffic emulation data).
WAIT	The generator is enabled to start transmitting ($\overline{EREN}$ is activated, i.e., the FIFO is ready to output the data) and moves to TRANSMIT state only when the ATMC asserts STXCLAV (the generator generates $\overline{STXENB}$ when moving from the WAIT state to the TRANSMIT state). ERST detection moves the generator to the RESET state.

Table 5-25. Engress Generator State Definitions (Continued)

State	Definition
TRANSMIT	The generator asserts STXENB and starts to transmit the data simultaneously. A complete cell is transmitted before the generator determines the next step. If at the end of a cell, ECCLV and STXCLAV are detected, the generator moves to the WAIT state. If EX_ESC and STXDATA0 are detected, then the generator moves to RET1 state. If EX_ESC and STXDATA0 are detected, then the generator moves to the STOP state. Detection of ERST at any time moves the generator to the RESET state.
STOP	The generator is halted. It stays in this state as long as ESIM is active (high) and ERST is inactive (low). When ESIM is deasserted (low), the generator moves to the RESET state.
RET1	This is a transfer state. The generator moves unconditionally to the RET2 state. If ERST is detected, it moves to the RESET state.
RET2	This is the RETRANSMIT state. If the generator is still in Run Mode (ESIM active) and STXCLAV is detected asserted, it moves back to the TRANSMIT state to the beginning of the FIFO. If ESIM is asserted but STXCLAV is detected deasserted, the generator moves to the WAIT state. Detection of ERST or ESIM inactive (low) moves the generator to the RESET state.

5.6 PHY Egress Recorder

The on-board Egress Recorder is made of 128K × 24 bit memory array. The recorder is loaded with the PHY Egress data in a cyclic manner. The recorder records the transmitted egress data until it is stopped by the user. Then the recorded data can be read by the MPC860. While recording, the on-board logic generates the addresses to the memory array. When the MPC860 reads the data stored in the memory array, it generates the addresses.

There are two modes of recording:

- **Slow Recorder**—The ATMC transmits the data faster then the recorder can receive it. This is to simulate an Egress PHY device that is full most of the time and therefore it has to assert TXFULL to control the flow of data. TXFULL is deasserted for a period determined by TFNP value in control register 2. TXFULL is asserted for a period determined by TFAP value in control register 3.
- **Fast Recorder**—The ATMC transmits the data slower then the recorder can receive. This is to simulate an Egress PHY device that is empty most of the time and can receive every cell that the ATMC transmits (cell based transmission). Since the recorder is recording continuously unless stopped, gaps will show in the recorder between cells when the ATMC has no cell to transmit.

5.6.1 Egress Recorder Control Registers

There are three 32-bit Egress Recorder Control Registers. **Table 5-26** through **Table 5-28** describe the register fields.

Table 5-26. Egress Recorder Control Register 1

Bit	Mnemonic	Function	Def	Att
0	TRST	**Transmit Reset**—Not used	0	W
1	TPIM	**Transmit Egress Interface Mode**—When active (high), the recorder is in run mode and it records the transmitted egress data. When in-active (low), the recorder is in setup mode and can be read by the MPC860.	0	W
2	TGPS	**Transmit Gaps allowed**—When active (high), the recorders' address is advanced even when transmission is suspended for any reason. It means that there will be gaps of no data in the recorder.	0	W
3	TFST	**Transmit Fast Mode**—When active (high) TFNP and TFAP are not used, TXFULL is always deasserted. When inactive, TFNP and TFAP are used.	1	W
4	TCLB	**Transmit Cell Based**—When active (high), the Egress Interface is cell based.	1	W
5–31	reserved	not implemented	0	—

Table 5-27. Egress Recorder Control Register 2

Bit	Mnemonic	Function	Def	Att
0–5	TFNP0–5	**Transmit TXFULL Deassertion Period**—The value (0–63) of this field determines the number of clock cycles that TXFULL is deasserted before assertion.	1	W
6–31	reserved	not implemented	0	—

Table 5-28. Egress Recorder Control Register 3

Bit	Mnemonic	Function	Def	Att
0–7	TFAP0–7	**Transmit TXFULL Assertion Period**—The value of this field (0-255) determines the number of clock cycles that TXFULL is asserted before deassertion when in Slow Recorder mode.	1	W
8–31	reserved	not implemented	0	—

5.6.2 Egress Recorder Status Register

A 32-bit Egress Recorder Status Register is defined, but not implemented.

5.6.3 Recorded Egress File Structure

The Egress Recorder stores data using the structure defined in **Table 5-29**.

Table 5-29. Egress Recorder Data Structure

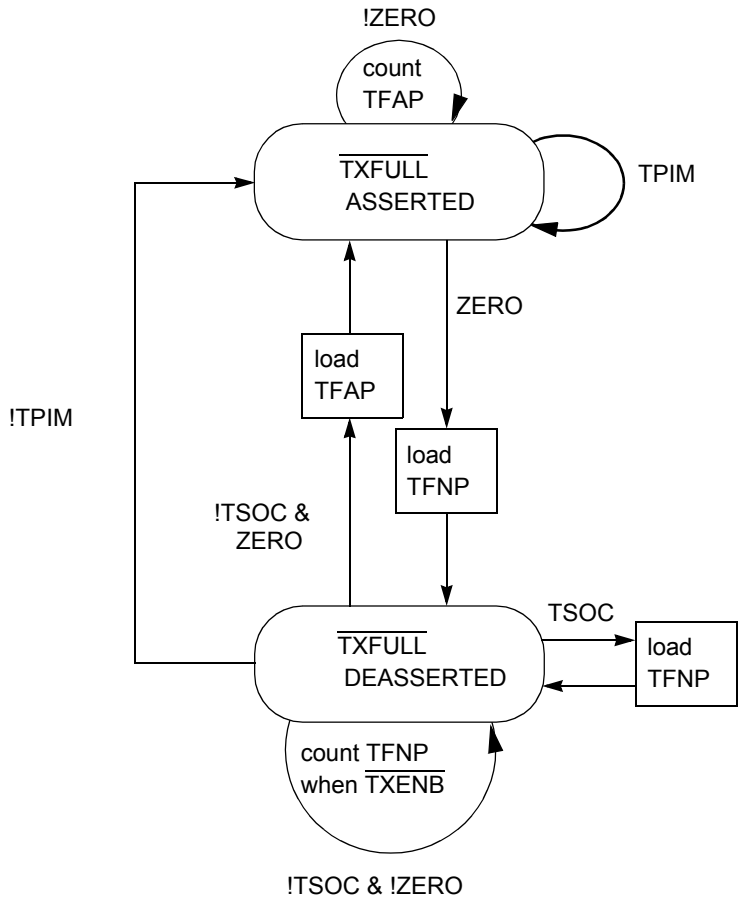
Bit	Mnemonic	Function
0–7	TXDATA7–0	**Transmit Data Lines 7–0**—These are the egress data lines.
8–10	reserved	These bits are not in use and are held low.
11–15	TXPHYID4–0	**Transmit PHY ID 4–0**—These bits show the PHY ID (link number) to which the transmission is directed. When using the MC92500 (UTOPIA 1), bits 3–0 are the PHY ID (only 16 links) and bit 4 is the PHY ID Valid bit. When using the MC92501 (UTOPIA 2), PHYID(3–0) show the PHY ID (16 links) and PHYID4 is the valid bit for the link.
16	TXPRTY	**Transmit Egress Parity**—It indicates the odd parity over the data lines.
17	TXSOC	**Transmit Egress Start Of Cell**—When active (high), the data on the data lines is the first octet of the cell. It should be active only with the first octet and inactive with the second octet until the end of the cell.
18	$\overline{\text{TXFULL}}$	**Transmit Egress PHY Buffer Full**—When active (low), it indicates that the Egress PHY Device is full and can't receive any more octets or cells (depends if the transmission is octet based or cell based respectively), therefore the ATMC suspends the egress transmission. If the transmission is cell based, the ATMC considers $\overline{\text{TXFULL}}$ only at the first octet of the cell and ignores it until the beginning of the next cell. When the transmission is octet based, the ATMC considers $\overline{\text{TXFULL}}$ on every octet it transmits. The transmission mode (cell based or octet based) is configured in one of the ATMC configuration registers.
19	$\overline{\text{TXENB}}$	**Transmit Egress Enable**—When active (low), this bit shows that the ATMC is transmitting valid data. The value of this bit depends on the $\overline{\text{TXFULL}}$ bit. When the ATMC detects $\overline{\text{TXFULL}}$ asserted, it deasserts $\overline{\text{TXENB}}$ and suspends transmission.
20–21	reserved	These bits are not in use and are held low.
22–23	TRADD(20–19)	**Transmit Egress Recorder Cycle Number**—Since the Egress Recorder runs in a cyclic manner, data is overwritten in the recorder. Every time the recorder reaches the end it goes back to the top and overwrites the data (unless stopped). These bits count the number of times the recorder started from the top (up to four times).

5.6.4 Transmit Egress Recorder State Machine

The Egress recorder operation is based on a state machine. As mentioned earlier, there are two modes:

- **Slow Recorder**—The ATMC transmits the data faster then the recorder can receive it. This is to simulate an Egress PHY device that is full most of the time and therefore it has to assert TXFULL to control the flow of data (octet based or cell based transmission). Thus, TXFULL is deasserted for a number of cycles determined by TFNP value in control register 2 and asserted for a number of cycles determined by TFAP value in control register 3.
- **Fast Recorder**—The ATMC transmits the data slower then the recorder can receive. This is to simulate an Egress PHY device that is empty most of the time and can receive every cell that the ATMC transmits (cell based transmission). Since the recorder is recording continuously unless stopped, gaps will show in the recorder between cells when the ATMC has no cell to transmit.

The algorithm of the state machine is shown in **Figure 5-4**.



** ZERO = (CNT==0)

** TSOC = !TXENB and TXSOC and TCLB

Figure 5-4. Transmit Egress Recorder State Machine

There are two distinguishable signals:

- **ZERO**—This state is active when either TFNP or TFAP has elapsed.
- **TSOC**—This state is active only at the first octet of a new cell when transmission is cell based.

There are two states:

- **$\overline{\text{TXFULL}}$ ASSERTED**—In this state, the recorder can not receive any data so the assertion of $\overline{\text{TXFULL}}$ suspends the transmission. The recorder is in this state when it is initialized (TPIM=0). When TPIM=0 the MPC860 can read the recorded data or set-up the recorder. When TPIM=1, the recorder is in run mode and can be in either state. When TPIM=1 this state means that the recorder can not receive an octet (octet based transmission) or a cell (cell based transmission). When entering this state the counter (CNT) is loaded with TFAP. The recorder asserts $\overline{\text{TXFULL}}$ for a number of clocks determined by the value of TFAP. Only at the end of that period (TFAP elapsed → ZERO active) the recorder moves to the next state. The next state is $\overline{\text{TXFULL}}$ deasserted so the counter (CNT) is loaded with TFNP.
- **$\overline{\text{TXFULL}}$ DEASSERTED**—In this state, $\overline{\text{TXFULL}}$ is deasserted, therefore the transmission is enabled. This state is valid only when TPIM=1. If TPIM=0, the recorder moves to $\overline{\text{TXFULL}}$ asserted state and transmission is suspended. When entering this state, the counter is loaded with TFNP. The recorder keeps $\overline{\text{TXFULL}}$ deasserted for a number of cycles that is determined by the value of TFNP. The counter that holds the TFNP value counts octets only when $\overline{\text{TXENB}}$ is asserted. When TFNP elapses, the recorder moves to $\overline{\text{TXFULL}}$ asserted state and the counter is loaded with TFAP. There are two options in this state - cell based and octet based transmission.

When in cell-based transmission, the recorder considers TSOC which is active only at the first octet of a cell. If TSOC is detected, the counter is re-loaded with TFNP. If Slow Recorder mode is desired, then TFNP should be 52 (for a total of 53 octets in a cell) since the counter is loaded after TSOC (the first octet) is detected. After the 53rd octet, the recorder asserts $\overline{\text{TXFULL}}$ and the transmission is suspended for TFAP cycles. If Fast Recorder mode is desired, then TFNP should be larger than 52 to enable the recorder to detect each TSOC.

When in octet-based transmission, the recorder does not consider TSOC. It records the data as long as TFNP does not elapse and $\overline{\text{TXENB}}$ is asserted. When TFNP elapses, the recorder suspends transmission and asserts $\overline{\text{TXFULL}}$. If Slow Recorder mode is desired, TFNP should be smaller than 53 to suspend the transmission regularly. If Fast recorder mode is desired, TFNP should be at least 53 to allow continuous recording of at least one cell and TFAP should be kept small to keep the transmission suspension period to a minimum.

Table 5-30 summarizes the options in the two states.

Table 5-30. *T*ransmit Egress Recorder State Machine Summary

Present State	Event	Next State	Action Taken	Fast Recorder Mode	Slow Recorder Mode	Cell Based	Octet Based
$\overline{\text{TXFULL}}$ Asserted	TPIM=0	$\overline{\text{TXFULL}}$ Asserted	*MPC860 reads the recorder *recorder setup.	—	—	—	—
$\overline{\text{TXFULL}}$ Asserted	TPIM=1 !ZERO	$\overline{\text{TXFULL}}$ Asserted	CNT = TFAP – 1	small TFAP	large TFAP	X	X
$\overline{\text{TXFULL}}$ Asserted	TPIM=1 ZERO	$\overline{\text{TXFULL}}$ deasserted	CNT = TFNP	TFNP > 53	TFNP = 52	X	—
$\overline{\text{TXFULL}}$ Asserted	TPIM=1 ZERO	$\overline{\text{TXFULL}}$ deasserted	CNT = TFNP	TFNP ≥ 53	TFNP < 52	—	X
$\overline{\text{TXFULL}}$ deasserted	TPIM=1 TSOC	$\overline{\text{TXFULL}}$ deasserted	CNT = TFNP	TFNP > 53	TFNP = 52	X	—
$\overline{\text{TXFULL}}$ deasserted	TPIM=1 TXENB !TSOC and !ZERO	$\overline{\text{TXFULL}}$ deasserted	CNT = TFNP – 1	—	—	X	X
$\overline{\text{TXFULL}}$ deasserted	TPIM=1 !TSOC and ZERO	$\overline{\text{TXFULL}}$ Asserted	CNT = TFAP	small TFAP	large TFAP	X	X

5.7 Switch Ingress Recorder

The on-board Switch Ingress Recorder is made of 128 K × 16-bit memory array. The recorder is loaded with the Switch Ingress data in a cyclic manner. The recorder records the received Ingress data continuously until it is stopped by the user. Then the recorded data can be read by the MPC860. While recording, the on-board logic generates the addresses to the memory array. When the MPC860 reads the data stored in the memory array, it generates the addresses. The operation of the recorder is based on the assumption that the switch is faster than the ATMC and therefore can receive any cell that the ATMC is outputting.

5.7.1 Control Register

There is one 32-bit wide control register. **Table 5-31** describes the register fields.

Table 5-31. Switch Ingress Recorder Control Register

Bit	Mnemonic	Function	Def	Att
0	IRST	**Switch Receive Reset**—Not used	0	W
1	ISIM	**Ingress Switch Interface Mode**—When active (high), the recorder is in Run Mode and records received Ingress data. When inactive (low), the recorder is in setup mode and can be read by the MPC860.	0	W
2	IFULL	**Ingress Switch is Full**—When active (high), the recorder suspends the operation by deasserting $\overline{\text{SRXENB}}$. When in-active, $\overline{\text{SRXENB}}$ is asserted if SRXCLAV is detected asserted.	0	W
3	IGPS	**Ingress Switch Gaps Allowed**—When active (high), the recorders' address is advanced even when transmission is suspended for any reason. It means that there will be gaps of no data in the recorder. When inactive, the address is advanced only when there is valid data.	0	W
4–31	reserved	not implemented.	0	—

5.7.2 Recorded Switch Ingress File Structure

Table 5-32 defines the Switch Ingress Recorder data structure.

Table 5-32. Switch Ingress Recorder Data Structure

Bit	Mnemonic	Function
0–7	SRXDATA7–0	**Switch Receive Data Lines 7–0**—These are the Ingress data lines.
8	SRXPRTY	**Switch Receive Ingress Parity**—It indicates the parity over the data lines: 0 = Odd parity and 1 = Even parity.
9	$\overline{\text{SRXENB}}$	**Switch Receive Ingress Enable**—The switch asserts $\overline{\text{SRXENB}}$ when it is ready to receive data from the ATMC. $\overline{\text{SRXENB}}$ is asserted only after the switch has detected SRXCLAV asserted.
10	SRXSOC	**Switch Receive Ingress Start Of Cell**—When active (high), the data on the data lines is the first octet of the cell. It should be active only with the first octet and inactive with the second octet until the end of the cell.
11	SRXCLAV	**Switch Receive Ingress Cell Available—**When active (high), the ATMC has a cell to transfer to the switch. The ATMC asserts SRXCLAV as long as it has cells to transfer to the switch. Since the transfer is cell based, SRXCLAV is asserted before the first octet and deasserted right after the last octet if no other cell is available.
12–13	reserved	These bits are not in use and are held low.
14–15	IRADD20–19	**Switch Receive Ingress Cycle Number**—The Switch Ingress Recorder runs in a cyclic manner and data is overwritten in the recorder from the top every time the recorder reaches the end unless stopped. These bits count the number of times the recorder started from the top (up to four times).

5.7.3 Switch Ingress Recorder State Machine

The Switch Ingress Recorder uses a very simple state machine that assumes that the switch is faster than the ATMC and therefore can receive any cell without delays. When the ATMC has cells to transfer to the switch, it asserts SRXCLAV. The recorder detects SRXCLAV and in return asserts SRXENB which indicates to the ATMC that it can start transferring the cells. When the ATMC has no more cells to transfer, it deasserts SRXCLAV and the recorder deasserts SRXENB in response. The deassertion of SRXCLAV suspends the recording unless gaps are allowed (the recorder keeps recording invalid data). The user can enforce the recorder to suspend recording by asserting IFULL. This deasserts SRXENB. The algorithm of the state machine is shown in **Figure 5-5**.

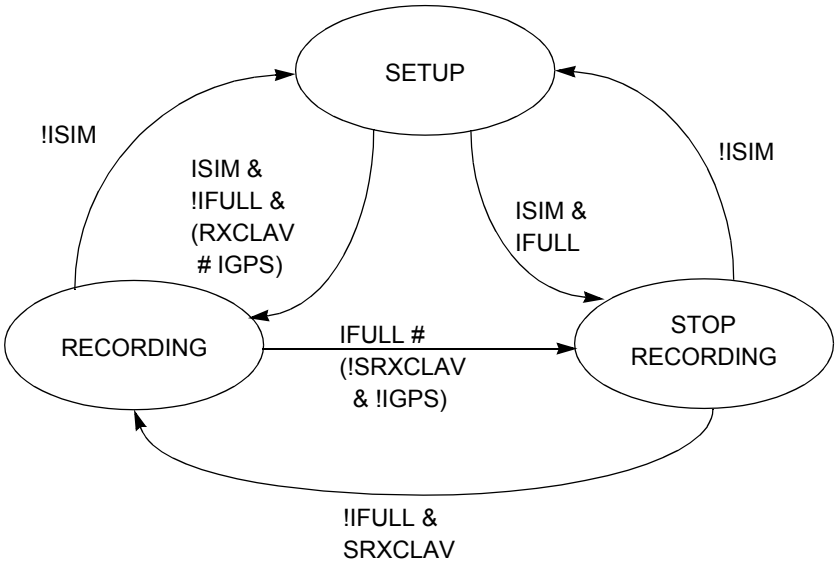


Figure 5-5. Switch Ingress Receive Recorder State Machine

- **SETUP STATE**—This is the initial state when ISIM = 0. In this state, the MPC860 can set up the recorder or read the data from the memory. When ISIM = 1 and IFULL is asserted, the recorder moves to stop recording state (enforced by IFULL). When ISIM = 1 and IFULL is not asserted and RXCLAV is asserted (ATMC has cells to transfer), the recording starts. If SRXCLAV is not asserted but IGPS is asserted, the recorder records invalid data.
- **STOP RECORDING STATE**—In this state the recorder is on but not recording because either IFULL is enforced or SRXCLAV is deasserted (ATMC has no cells to transfer) and IGPS = 0 (no gaps are allowed). When ISIM = 0, the recorder moves to Setup State. If IFULL is deasserted and SRXCLAV is detected asserted, the recorder moves to Recording State.
- **RECORDING STATE**—In this state, the recorder is recording data. If SRXCLAV is detected asserted, the recorder is recording valid data. If SRXCLAV is deasserted but IGPS is asserted (gaps allowed), the recorder is recording invalid data (gaps). When ISIM = 0, the recorder moves to Setup State. When either IFULL is enforced or SRXCLAV is deasserted (no cells to transfer) and IGPS is asserted (no gaps allowed), the recorder moves to Stop Recording State.

6

Support Information

6.1 Introduction

This section includes technical information needed for support, maintenance and connectivity to the ATMC EVB.

6.2 Interconnect Signals

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

- P1—Expansion connector
- P2—UTOPIA port connector
- P3, P4, P5, P6, P7, and P8—Logic Analyzer connectors
- P9—External Debug Port controller input
- P10—5 V Power In
- P11—RS-232 port
- P12—Ethernet port
- P13—ADI Port connector

6.2.1 Expansion Connector P1

The Expansion connector is a QUADS compatible Communication connector, for the benefit of those who developed communication tools for the M68360QUADS or M68360QUADS-040 boards. All SCC and I/O pins are routed to the same locations as they exist on the above boards. That way it is easy to migrate upward from the QUICC to the MPC860. P1 is a 96-pin, DIN 41612 receptacle connector.

Table 6-1. P1 Interconnect Signals

Pin No.	Signal Name	Att.	Description
A1	ETHRX	I/O	Ethernet port Receive Data. In fact PA15/RXD1. When the Ethernet port is disabled via BCSR1, may be used off-board.
A2	ETHTX	I/O	Ethernet port Transmit Data. In fact PA14/TXD1. When the Ethernet port is disabled via BCSR1, may be used off-board for any alternate function.
A3	PA13	I/O	PA13/RXD2—not used on the EVB
A4	PA12	I/O	PA12/TXD2—not used on the EVB
A5	PD11	I/O	PD11/RXD3—not used on the EVB
A6	PD10	I/O	PD10/TXD3—not used on the EVB
A7	PD9	I/O	PD9/RXD4—not used on the EVB
A8	PD8	I/O	PD8/TXD4—not used on the EVB
A9	ETHTCK	I/O	Ethernet Port Transmit Clock—in fact PA7/CLK1/TIN1/ L1RCLKA/BRGO1. When the Ethernet port is disabled via BCSR1, this signal may be used off-board for any alternate function.
A10	ETHRCK	I/O	Ethernet Port Receive Clock—in fact PA7/CLK2/$\overline{\text{TOUT1}}$/ BRGCLK1. When the Ethernet port is disabled via BCSR1, this signal may be used off-board for any alternate function.
A11	PA5	I/O	PA5/CLK3/TIN2/L1TCLKA/BRGOUT2—not used on the EVB
A12	PA4	I/O	PA4/CLK4/$\overline{\text{TOUT2}}$—not used on the EVB
A13	PA3	I/O	PA3/CLK5/TIN3/BRGOUT3—not used on the EVB
A14	PA2	I/O	PA2/CLK6/TOUT3/L1RCLKB/BRGCLK2—not used on the EVB
A15	PA1	I/O	PA1/CLK7/TIN4/BRGO4—not used on the EVB
A16	PA0	I/O	PA0/CLK8/TOUT4/L1TCLKB—not used on the EVB
A17	VCC	—	5 V Power line
A18	PA11	I/O	PA11/L1TXDB—not used on the EVB
A19	PA10	I/O	PA10/L1RXDB—not used on the EVB

Table 6-1. P1 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
A20	PA9	I/O	PA9/L1TXDA—not used on the EVB
A21	PA8	I/O	PA8/L1RXDA—not used on the EVB
A22	GND	—	Ground
A23	GND	—	Ground
A24	$\overline{\text{IRQ7}}$	I, L	Interrupt Request 7—the lowest priority interrupt request line—not used on the EVB
A25	$\overline{\text{IRQ6}}$	I/O	FRZ/$\overline{\text{IRQ6}}$—Freeze (Debug Mode) indication or Interrupt Request 6. Configured on the EVB as $\overline{\text{IRQ6}}$. Although not by EVB logic, this signal may be configured to an alternate function if $\overline{\text{IRQ6}}$ is not required.
A26	$\overline{\text{ETHEN}}$	O, L	Ethernet Port Enable—connected to BCSR1
A27	$\overline{\text{IRQ3}}$	I	$\overline{\text{CR}}$/$\overline{\text{IRQ3}}$—Cancel Reservation input or Interrupt Request line 3—pulled-up but otherwise unused on the EVB
A28	$\overline{\text{IRQ2}}$	I/O, L	$\overline{\text{RSV}}$/$\overline{\text{IRQ2}}$—Reservation output or Interrupt Request line 2 input—pulled-up but otherwise unused on the EVB
A29	$\overline{\text{IRQ1}}$	I, L	Interrupt Request 1—pulled-up but otherwise not used on the EVB
A30	$\overline{\text{NMI}}$	I, L	Non-Maskable Interrupt—in fact $\overline{\text{IRQ0}}$ of the MPC—driven by on-board logic by O.D. gate—may be driven off-board by O.D. gate only
A31	$\overline{\text{RS\_EN}}$	O,L	RS-232 Port Enable—connected to BCSR1
A32	GND	—	Ground
B1	PB31	I/O	PB31/$\overline{\text{SPISEL}}$/$\overline{\text{RRJECT1}}$—not used on the EVB
B2	PB30	I/O	PB30/SPICLK—not used on the EVB
B3	PB29	I/O	PB29/SPIMOSI—not used on the EVB
B4	PB28	I/O	PB28/SPIMISO/BRGO4—not used on the EVB
B5	PB27	I/O	PB27/I2CSDA/BRGO1—not used on the EVB
B6	PB26	I/O	PB26/I2CSCL/BRGO2—not used on the EVB
B7	RSTXD	I/O	RS-232 port Transmit Data—in fact PB25/SMTXD1. When the RS-232 port is disabled via BCSR1, this signal may be used off-board for any alternate function.
B8	RSRXD	I/O	RS-232 port Receive Data—in fact PB24/SMRXD1. When the RS-232 port is disabled via BCSR1, this signal may be used off-board for any alternate function.

Table 6-1. P1 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
B9	RSDTR	I/O	RS-232 port DTR signal—in fact PB23/SMSYN1/SDACK1. When the RS-232 port is disabled via BCSR1, this signal may be used off-board for any alternate function.
B10	PB22	I/O	PB22/SMSYN2/SDACK2—not used on the EVB
B11	PB21	I/O	PB21/SMTXD2/L1CLKOB—not used on the EVB
B12	PB20	I/O	PB20/SMRXD2/L1CLKOA—not used on the EVB
B13	E_TENA	I/O	Ethernet port Transmit Enable—in fact PB19/RTS1/L1ST1. When active, transmit is enabled via the MC68160 EEST. When the ethernet port is disabled via BCSR1, this signal may be used off-board for any alternate function.
B14	PB18	I/O	PB18/RTS2/L1ST2—not used on the EVB
B15	PB17	I/O	PB17/L1RQB/L1ST3—not used on the EVB
B16	PB16	I/O	PB16/L1RQA/L1ST4—not used on the EVB
B17	PB15	I/O	PB15/BRGO3—not used on the EVB
B18	PB14	I/O	PB14/RSTRT1—not used on the EVB
B19	GND	—	Ground
B20	DREQ0	I/O	IDMA request line 0
B21	DREQ1	I/O	IDMA request line 1
B22	PC13	I/O	PC13/L1RQB/L1ST3—not used on the EVB
B23	PC12	I/O	PC12/L1RQA/L1ST4—not used on the EVB
B24	E_CLSN	I/O	Ethernet Port Collision indication signal—in fact PC11/CTS1. When the Ethernet port is disabled via BCSR1, this signal may be used off-board for any alternate function.
B25	E_RENA	I/O	Ethernet Receive Enable—in fact PC10/CD1/TGATE1. Active when there is network activity. When the Ethernet port is disabled via BCSR1, this signal may be used off-board for any alternate function.
B26	PC9	I/O	PC9/CTS2—not used on the EVB
B27	PC8	I/O	PC8/CD2/TGATE2—not used on the EVB
B28	PC7	I/O	PC7/L1TSYNCB/SDACK2(/CTS3 for MPC860)—not used on the EVB

Table 6-1. P1 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
B29	TPSQEL	I/O	Twisted Pair Signal Quality Error Test Enable—in fact PC6/L1RSYNCB(/CD3 for MPC860). When active, a simulated collision state is generated within the EEST, so the collision detection circuitry within the EEST may be tested. Since after Hardware Reset, this line wakes up tri-stated, it should be initialized as an output and given the desired value.
B30	TPFLDL	I/O	Twisted Pair Full-Duplex—in fact PC5/L1TSYNCA/SDACK1 (/CTS4 for MPC860). When active, the MC68160 EEST is put into full-duplex mode, where, simultaneous receive and transmit are enabled. Since after Hardware Reset, this line wakes up tri-stated, it should be initialized as an output and given the desired value.
B31	ETHLOOP	I/O	Ethernet Port Diagnostic Loop-Back—in fact PC4/L1RSYNCA(/CD4 for MPC860). When active, the MC68160 EEST is configured into diagnostic Loop-Back mode, where the transmit output is internally fed back into the receive section. Since after Hardware Reset, this line wakes up tri-stated, it should be initialized as an output and given the desired value.
B32	GND	—	Ground
C1	VCC	—	5 V power line
C2	VCC	—	5 V power line
C3	VCC	—	5 V power line
C4	VCC	—	5 V power line
C5	VCC	—	5 V power line
C6	BRS_EN2	—	Not Connected
C7	GND	—	Ground
C8	GND	—	Ground
C9	GND	—	Ground
C10	GND	—	Ground
C11	GND	—	Ground
C12	GND	—	Ground
C13	GND	—	Ground
C14	GND	—	Ground
C15	PD15	I/O	PD15/L1TSYNCA—not used on the EVB
C16	PD14	I/O	PD14/L1RSYNCA—not used on the EVB

Table 6-1. P1 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
C17	PD13	I/O	PD13/L1TSYNCB—not used on the EVB
C18	PD12	I/O	PD12/L1RSYNCB—not used on the EVB
C19	PD7	I/O	$\overline{\text{PD7}}$/$\overline{\text{RTS3}}$—not used on the EVB
C20	PD6	I/O	MPC860 $\overline{\text{PD6/RTS4}}$—not used on the EVB
C21	VCC	—	5 V power line
C22	$\overline{\text{HRESET}}$	I/O, L	MPC Hardware Reset—driven by on-board logic and may be driven by off-board logic with Open-Drain gate only
C23	$\overline{\text{SRESET}}$	I/O, L	MPC Software Reset—driven by on-board logic and may be driven by off-board logic with Open-Drain gate only
C24	NC	—	Not Connected
C25	VCC	—	5 V power line
C26	PD3	I/O	PD3/$\overline{\text{RRJECT4}}$—not used on the EVB
C27	NC	—	Not Connected
C28	NC	—	Not Connected
C29	GND	—	Ground
C30	PD4	I/O	PD4/$\overline{\text{RRJECT3}}$—not used on the EVB
C31	GND	—	Ground
C32	PD5	I/O	PD5/$\overline{\text{RRJECT2}}$—not used on the EVB

6.2.2 UTOPIA Ports Connector P2

The UTOPIA Connector contains all the Ingress and Egress UTOPIA Ports and the Switch Ingress and Egress UTOPIA Ports. Through these lines, the user has direct access to the ATMC UTOPIA Ports for the purpose of connecting external devices and /or measurement equipment. For details on the functionality of the signals look in the ATMC data sheet. P2 is a 96-pin DIN 41612 receptacle connector.

Table 6-2. P2 Interconnect Signals

Pin No.	Signal Name	Att.	Description
A1	TXDATA0	O	Transmit Egress Data 0
A2	TXDATA1	O	Transmit Egress Data 1
A3	TXDATA2	O	Transmit Egress Data 2
A4	TXDATA3	O	Transmit Egress Data 3
A5	TXDATA4	O	Transmit Egress Data 4
A6	TXDATA5	O	Transmit Egress Data 5
A7	TXDATA6	O	Transmit Egress Data 6
A8	TXDATA7	O	Transmit Egress Data 7
A9	TXPHYID0	O	Transmit Egress PHY ID 0
A10	TXPHYID1	O	Transmit Egress PHY ID 1
A11	TXPHYID2	O	Transmit Egress PHY ID 2
A12	TXPHYID3	O	Transmit Egress PHY ID 3
A13	TXPHYID4	O	Transmit Egress PHY ID 4
A14	TXPRTY	O	Transmit Egress Parity
A15	$\overline{\text{TXFULL}}$	I	Transmit Egress Full
A16	TXSOC	O	Transmit Egress Start Of Cell
A17	$\overline{\text{TXENB}}$	O	Transmit Egress Enable
A18	GND	—	Ground
A19	STXDATA7	I	Switch Transmit Egress Data 7
A20	STXDATA6	I	Switch Transmit Egress Data 6
A21	STXDATA5	I	Switch Transmit Egress Data 5
A22	STXDATA4	I	Switch Transmit Egress Data 4
A23	STXDATA3	I	Switch Transmit Egress Data 3
A24	STXDATA2	I	Switch Transmit Egress Data 2
A25	STXDATA1	I	Switch Transmit Egress Data 1

Table 6-2. P2 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
A26	STXDATA0	I	Switch Transmit Egress Data 0
A27	STXPRTY	I	Switch Transmit Egress Parity
A28	STXCLAV	O	Switch Transmit Egress Cell Available
A29	STXSOC	I	Switch Transmit Egress Start Of Cell
A30	$\overline{\text{STXENB}}$	I	Switch Transmit Egress Enable
A31	STXCLKIN	I	Switch Transmit Egress Clock In
A32	GND	—	Ground
B1	GND	—	Ground
B2	ACLK	—	Ingress/Egress Clock
B3	GND	—	Ground
B4	GND	—	Ground
B5	VDD	O	3 V Power Line
B6	VDD	O	3 V Power Line
B7	GND	—	Ground
B8	VCC	O	5 V Power Line
B9	VCC	O	5 V Power Line
B10	GND	—	Ground
B11	GND	—	Ground
B12	NC	—	Not Connected
B13	NC	—	Not Connected
B14	NC	—	Not Connected
B15	NC	—	Not Connected
B16	NC	—	Not Connected
B17	NC	—	Not Connected
B18	NC	—	Not Connected
B19	NC	—	Not Connected
B20	NC	—	Not Connected
B21	NC	—	Not Connected
B22	NC	—	Not Connected
B23	GND	—	Ground

Table 6-2. P2 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
B24	GND	—	Ground
B25	GND	—	Ground
B26	GND	—	Ground
B27	GND	—	Ground
B28	VCC	O	5 V Power Line
B29	VCC	O	5 V Power Line
B30	GND	—	Ground
B31	VDD	O	3 V Power Line
B32	VDD	O	3 V Power Line
C1	RXDATA 0	I	Receive Ingress Data 0
C2	RXDATA 1	I	Receive Ingress Data 1
C3	RXDATA 2	I	Receive Ingress Data 2
C4	RXDATA 3	I	Receive Ingress Data 3
C5	RXDATA 4	I	Receive Ingress Data 4
C6	RXDATA 5	I	Receive Ingress Data 5
C7	RXDATA 6	I	Receive Ingress Data 6
C8	RXDATA 7	I	Receive Ingress Data 7
C9	RXPHYID 0	I	Receive Ingress PHY ID 0
C10	RXPHYID 1	I	Receive Ingress PHY ID 1
C11	RXPHYID 2	I	Receive Ingress PHY ID 2
C12	RXPHYID 3	I	Receive Ingress PHY ID 3
C13	RXPHYID 4	I	Receive Ingress PHY ID 4
C14	RXPRTY	I	Receive Ingress Parity
C15	$\overline{\text{RXEMPTY}}$	I	Receive Ingress Empty
C16	RXSOC	I	Receive Ingress Start Of Cell
C17	$\overline{\text{RXENB}}$	O	Receive Ingress Enable
C18	GND	—	Ground
C19	SRXDATA 7	O	Switch Receive Ingress Data 7
C20	SRXDATA 6	O	Switch Receive Ingress Data 6
C21	SRXDATA 5	O	Switch Receive Ingress Data 5

Table 6-2. P2 Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
C22	SRXDATA 4	O	Switch Receive Ingress Data 4
C23	SRXDATA 3	O	Switch Receive Ingress Data 3
C24	SRXDATA 2	O	Switch Receive Ingress Data 2
C25	SRXDATA 1	O	Switch Receive Ingress Data 1
C26	SRXDATA 0	O	Switch Receive Ingress Data 0
C27	SRXPRTY	O	Switch Receive Ingress Parity
C28	SRXCLAV	O	Switch Receive Ingress Cell Available
C29	SRXSOC	O	Switch Receive Ingress Start Of Cell
C30	$\overline{\text{SRXENB}}$	I	Switch Receive Ingress Enable
C31	SRXCLKIN	I	Switch Receive Ingress Clock In
C32	GND	—	Ground

6.2.3 Logic Analyzer Connectors P3–P8

There are six 43-pin Logic Analyzer Connectors P3–P8 used to interface with an HP Digital Analyzer. **Table 6-3** through **Table 6-8** list the pinouts for the Logic Analyzer Connectors.

Table 6-3. P3 MPC860 Control Interconnect Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA—not connected
5	J37	—	Not connected
6	$\overline{\text{TA}}$	I	Transfer Acknowledge
7	$\overline{\text{NMI}}$	I	Non Maskable Interrupt
8	$\overline{\text{DREQ1}}$	I	iDMA Request 1
9	$\overline{\text{IRQ1}}$	I	Interrupt Request 1
10	$\overline{\text{DREQ0}}$	I	iDMA Request 0
11	$\overline{\text{IRQ2}}$	I	Interrupt Request 2
12	$\overline{\text{MCIREQ}}$	I	Microprocessor Cell Insertion Request
13	$\overline{\text{IRQ3}}$	I	Interrupt Request 3
14	$\overline{\text{MCOREQ}}$	I	Microprocessor Cell Out (Extraction) Request

Table 6-3. P3 MPC860 Control Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
15	$\overline{\text{IRQ4}}$	I	Interrupt Request 4
16	$\overline{\text{EMMREQ}}$	I	External Memory Maintenance Request
17	$\overline{\text{IRQ6}}$	I	Interrupt Request 6
18	$\overline{\text{ATM\_CS}}$	I	ATMC Chip Select
19	$\overline{\text{IRQ7}}$	I	Interrupt Request 7
20	RD/$\overline{\text{WR}}$	I	Read/Write
21	$\overline{\text{HRESET}}$	I	Hard Reset
22	$\overline{\text{TA}}$	I	Transfer Acknowledge
23	$\overline{\text{SRESET}}$	I	Soft Reset
24	$\overline{\text{MWSH}}$	I	Microprocessor Word Select High
25	$\overline{\text{RSTCNF}}$	I	Reset Configuration
26	$\overline{\text{MWSL}}$	I	Microprocessor Word Select Low
27	$\overline{\text{PDRST}}$	I	Power On Reset
28	$\overline{\text{ATMCDTK}}$	I	ATMC Data Acknowledge
29	DSDI	I	Debug Serial Data In
30	$\overline{\text{CAM\_CS}}$	I	CAM Chip Select
31	DSCK	I	Debug Clock
32	$\overline{\text{CAM\_DTK}}$	I	CAM Data Acknowledge
33	DSD0	I	Debug Serial Data Out
34	$\overline{\text{EACEN}}$	I	External Address Compression Enable
35	TMS	I	Test Mode Select
36	$\overline{\text{G}}$	I	CAM Output Enable
37	$\overline{\text{TRST}}$	I	Test Reset
38	$\overline{\text{LH}}$/$\overline{\text{SH}}$	I	Load High/Start Match
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

Table 6-4. P4 Analyzer Address Interconnect Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA—not connected
5	$\overline{TA}$	I	Transfer Acknowledge
6	$\overline{TS}$	I	Transfer Start
7	$\overline{TA}$	I	Transfer Acknowledge
8	ADD16	I	ADDRESS 16
9	RD/$\overline{WR}$	I	Read/Write
10	ADD17	I	ADDRESS 17
11	TSIZE0	I	Transfer Size 0
12	ADD18	I	ADDRESS 18
13	TSIZE1	I	Transfer Size 1
14	ADD19	I	ADDRESS 19
15	J14	—	Not connected
16	ADD20	I	ADDRESS 20
17	ADD5	I	ADDRESS 5
18	ADD21	I	ADDRESS 21
19	ADD6	I	ADDRESS 6
20	ADD22	I	ADDRESS 22
21	ADD7	I	ADDRESS 7
22	ADD23	I	ADDRESS 23
23	ADD8	I	ADDRESS 8
24	ADD24	I	ADDRESS 24
25	ADD9	I	ADDRESS 9
26	ADD25	I	ADDRESS 25
27	ADD10	I	ADDRESS 10
28	ADD26	I	ADDRESS 26
29	ADD11	I	ADDRESS 11
30	ADD27	I	ADDRESS 27

Table 6-4. P4 Analyzer Address Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
31	ADD12	I	ADDRESS 12
32	ADD28	I	ADDRESS 28
33	ADD13	I	ADDRESS 13
34	ADD29	I	ADDRESS 29
35	ADD14	I	ADDRESS 14
36	ADD30	I	ADDRESS 30
37	ADD15	I	ADDRESS 15
38	ADD31	I	ADDRESS 31
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

Table 6-5. P5 MPC860 Control Interconnect Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA—not connected
5	J41	—	Not connected
6	SYSCLK	I	System Clock
7	$\overline{\text{F\_CS1}}$	I	Flash Chip Select 1
8	$\overline{\text{WE0}}$	I	Write Enable 0
9	$\overline{\text{F\_CS2}}$	I	Flash Chip Select 2
10	$\overline{\text{WE1}}$	I	Write Enable 1
11	$\overline{\text{F\_CS3}}$	I	Flash Chip Select 3
12	$\overline{\text{WE2}}$	I	Write Enable 2
13	$\overline{\text{F\_CS4}}$	I	Flash Chip Select 4
14	$\overline{\text{WE3}}$	I	Write Enable 3
15	$\overline{\text{F\_DE}}$	I	Flash Output Enable
16	$\overline{\text{BS\_A0}}$	I	Byte Select 0—UPMA
17	$\overline{\text{FLASHCS}}$	I	Flash Chip Select
18	$\overline{\text{BS\_A1}}$	I	Byte Select 1—UPMA
19	J42	—	Not connected
20	$\overline{\text{BS\_A2}}$	I	Byte Select 2—UPMA
21	J43	—	Not connected
22	$\overline{\text{BS\_A3}}$	I	Byte Select 3—UPMA
23	$\overline{\text{DRAMCS1}}$	I	DRAM Chip Select 1
24	$\overline{\text{RAS1}}$	I	Row Address Select 1—Single Bank DRAM SIMM
25	$\overline{\text{DRAMCS2}}$	I	DRAM Chip Select 2
26	$\overline{\text{RAS2}}$	I	Row Address Select 2—Single Bank DRAM SIMM
27	$\overline{\text{GEN\_CS}}$	I	General Chip Select
28	$\overline{\text{RAS100}}$	I	Row Address Select 1—Dual Bank DRAM SIMM
29	$\overline{\text{PHY\_CS}}$	I	PHY Chip Select
30	$\overline{\text{RAS200}}$	I	Row Address Select 2—Dual Bank DRAM SIMM

Table 6-5. P5 MPC860 Control Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
31	$\overline{\text{BCSR\_CS}}$	I	BCSR Chip Select
32	$\overline{\text{DRM\_W}}$	I	DRAM Write
33	$\overline{\text{TS}}$	I	Transfer Start
34	$\overline{\text{RD\_OE}}$	I	Read Output Enable
35	$\overline{\text{TEA}}$	I	Transfer Error Acknowledge
36	J19	—	Not connected
37	$\overline{\text{TA}}$	I	Transfer Acknowledge
38	SYSCLK	I	System Clock
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

Table 6-6. P6 Analyzer Data Bus Interconnect Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA—not connected
5	$\overline{\text{WE0}}$	I	Write Enable 0
6	$\overline{\text{RD\_OE}}$	I	Read Output Enable
7	DAT0	I	Data 0
8	DAT16	I	Data 16
9	DAT1	I	Data 1
10	DAT17	I	Data 17
11	DAT2	I	Data 2
12	DAT18	I	Data 18
13	DAT3	I	Data 3
14	DAT19	I	Data 19
15	DAT4	I	Data 4
16	DAT20	I	Data 20
17	DAT5	I	Data 5
18	DAT21	I	Data 21
19	DAT6	I	Data 6
20	DAT22	I	Data 22
21	DAT7	I	Data 7
22	DAT23	I	Data 23
23	DAT8	I	Data 8
24	DAT24	I	Data 24
25	DAT9	I	Data 9
26	DAT25	I	Data 25
27	DAT10	I	Data 10
28	DAT26	I	Data 26
29	DAT11	I	Data 11
30	DAT27	I	Data 27

Table 6-6. P6 Analyzer Data Bus Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
31	DAT12	I	Data 12
32	DAT28	I	Data 28
33	DAT13	I	Data 13
34	DAT29	I	Data 29
35	DAT14	I	Data 14
36	DAT30	I	Data 30
37	DAT15	I	Data 15
38	DAT31	I	Data 31
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

Table 6-7. P7 ATMC External Memory Data Bus Interconnect Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA—not connected
5	$\overline{\text{EMBSH0}}$	I	External Memory bank Select High 0
6	$\overline{\text{EMBSL0}}$	I	External Memory bank Select Low 0
7	EMDATA31	I	External Memory Data 31
8	EMDATA15	I	External Memory Data 15
9	EMDATA30	I	External Memory Data 30
10	EMDATA14	I	External Memory Data 14
11	EMDATA29	I	External Memory Data 29
12	EMDATA13	I	External Memory Data 13
13	EMDATA28	I	External Memory Data 28
14	EMDATA12	I	External Memory Data 12
15	EMDATA27	I	External Memory Data 27
16	EMDATA11	I	External Memory Data 11
17	EMDATA26	I	External Memory Data 26
18	EMDATA10	I	External Memory Data 10
19	EMDATA25	I	External Memory Data 25
20	EMDATA9	I	External Memory Data 9
21	EMDATA24	I	External Memory Data 24
22	EMDATA8	I	External Memory Data 8
23	EMDATA23	I	External Memory Data 23
24	EMDATA7	I	External Memory Data 7
25	EMDATA22	I	External Memory Data 22
26	EMDATA6	I	External Memory Data 6
27	EMDATA21	I	External Memory Data 21
28	EMDATA5	I	External Memory Data 5
29	EMDATA20	I	External Memory Data 20
30	EMDATA4	I	External Memory Data 4

Table 6-7. P7 ATMC External Memory Data Bus Interconnect Signals (Continued)

Pin No.	Signal Name	Att.	Description
31	EMDATA19	I	External Memory Data 19
32	EMDATA3	I	External Memory Data 3
33	EMDATA18	I	External Memory Data 18
34	EMDATA2	I	External Memory Data 2
35	EMDATA17	I	External Memory Data 17
36	EMDATA1	I	External Memory Data 1
37	EMDATA16	I	External Memory Data 16
38	EMDATA0	I	External Memory Data 0
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

Table 6-8. P8 ATMC External Memory Address/Control Signals

Pin No.	Signal Name	Att.	Description
1	NC	I	VDC—not connected
2	NC	I	SCL—not connected
3	GND	I	Ground
4	NC	I	SDA
5	ACLK4	I	ATMC Clock
6	SYSCLK	I	System Clock
7	SYSCLK	I	System Clock
8	EMADD17	I	External Memory ADDRESS 17
9	J72	—	Not connected
10	EMADD16	I	External Memory ADDRESS 16
11	J73	—	Not connected
12	EMADD15	I	External Memory ADDRESS 15
13	$\overline{\text{EMWR}}$	I	External Memory Write
14	EMADD14	I	External Memory ADDRESS 14
15	$\overline{\text{EMBSH3}}$	I	External Memory Bank Select High 3
16	EMADD13	I	External Memory ADDRESS 13
17	$\overline{\text{EMBSL3}}$	I	External Memory Bank Select Low 3
18	EMADD12	I	External Memory ADDRESS 12
19	$\overline{\text{EMBSH2}}$	I	External Memory Bank Select High 2
20	EMADD11	I	External Memory ADDRESS 11
21	$\overline{\text{EMBSL2}}$	I	External Memory Bank Select Low 2
22	EMADD10	I	External Memory ADDRESS 10
23	$\overline{\text{EMBSH1}}$	I	External Memory Bank Select High 1
24	EMADD9	I	External Memory ADDRESS 9
25	$\overline{\text{EMBSL1}}$	I	External Memory Bank Select Low 1
26	EMADD8	I	External Memory ADDRESS 8
27	$\overline{\text{EMBSH0}}$	I	External Memory Bank Select High 0
28	EMADD7	I	External Memory ADDRESS 7
29	$\overline{\text{EMBSL0}}$	I	External Memory Bank Select Low 0
30	EMADD6	I	External Memory ADDRESS 6

Table 6-8. P8 ATMC External Memory Address/Control Signals (Continued)

Pin No.	Signal Name	Att.	Description
31	EMADD21	I	External Memory ADDRESS 21
32	EMADD5	I	External Memory ADDRESS 5
33	EMADD20	I	External Memory ADDRESS 20
34	EMADD4	I	External Memory ADDRESS 4
35	EMADD19	I	External Memory ADDRESS 19
36	EMADD3	I	External Memory ADDRESS 3
37	EMADD18	I	External Memory ADDRESS 18
38	EMADD2	I	External Memory ADDRESS 2
39	GND	I	Ground
40	GND	I	Ground
41	GND	I	Ground
42	GND	I	Ground
43	GND	I	Ground

6.2.4 External Debug Port Controller Input Interconnect P9

The Debug Port connector P9 is a 10-pin header plug, signals of which are described in **Table 6-9**

Table 6-9. P9 Interconnect Signals

Pin No.	Signal Name	Att.	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. It also indicates whether the MPC is in Debug Mode. If not using the Debug Port, this signal may be configured for any alternate function.
2	$\overline{\text{SRESET}}$	I/O	Software Reset line of the MPC—Active-low, Open-Drain
3	GND	—	Ground
4	DSCK	I/O	Debug Serial Clock—over the rising edge, serial date is sampled by the MPC from DSDI signal. Over the falling edge, DSDI is driven towards the MPC and DSDO is driven by the MPC. The signal is configured on the MPC's JTAG port. This signal is an output when the Debug Port Controller is on the local MPC or when the ADS is a Debug Port Controller for a target system. It is an input when the ADI bundle is disconnected from the EVB.
5	GND	—	Ground
6	VFLS1	O	See VFLS0.
7	$\overline{\text{HRESET}}$	I/O	Hardware 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. This signal is an output when the Debug Port Controller is on the local MPC or when the EVB is a Debug Port Controller for a target system. It is an input when the ADI bundle is disconnected from the EVB.
9	V3.3	O	3.3 V 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. This signal is an output when the Debug Port Controller is on the local MPC or when the ADI bundle is disconnected from the EVB. It is an input when the ADS is a Debug Port Controller for a target system.

6.2.5 5 V Power Connector P10

The 5V power connector P10, is a 3-lead, two-part terminal block. The plug part is soldered to the Printed Circuit Board (PCB) and the receptacle is connected to the power supply to facilitate fast connection/disconnection of power and reduce physical stress on the solders to maintain a solid connection over time. **Table 6-10** lists the P10 pinout.

Table 6-10. P10 Interconnect Signals

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

6.2.6 RS-232 Port Connector P11

The RS-232 port connector P11 is a 9-pin, 90°, D-Type receptacle connector, signals of which are presented in **Table 6-11**.

Table 6-11. P11 Interconnect Signals

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

6.2.7 Ethernet Port Connector P12

The Ethernet connector on the ATMC EVB P12, is a Twisted-Pair (10-Base-T) compatible connector. Use is done with 90°, 8-pin, RJ45 connector, signals of which are described in **Table 6-12**.

Table 6-12. P12 Ethernet Port Interconnect Signals

Pin No.	Signal Name	Description
1	TPTX	Twisted-Pair Transmit Data positive output from the ATMC EVB
2	$\overline{\text{TPTX}}$	Twisted-Pair Transmit Data negative output from the ATMC EVB
3	TPRX	Twisted-Pair Receive Data positive input to the ATMC EVB
4	—	Not connected
5	—	Not connected
6	$\overline{\text{TPRX}}$	Twisted-Pair Receive Data negative input to the ATMC EVB
7	—	Not connected
8	—	Not connected

6.2.8 ADI Port Connector P13

The ADI port connector P13 is a 37-pin, 90°, D-Type plug connector, signals of which are described in **Table 6-13**.

Table 6-13. P13 ADI Port Interconnect Signals

Pin No.	Signal Name	Description
1	—	Not connected with this application
2	D_$\overline{\text{C}}$	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 Software Reset to the MPC
5	ADS_HRESET	When asserted (1) and the ADS is selected by the host and generates a Hardware Reset to the MPC
6	ADS_SEL2	ADI I/F address line 2 (MSB)
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

Table 6-13. P13 ADI Port Interconnect Signals (Continued)

Pin No.	Signal Name	Description
10	ADS_REQ	ADS Request output signal from the MPC821/860ADS to the host
11	ADS_ACK	ADS Acknowledge output signal from the MPC821/860ADS 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	$\overline{\text{HOST\_ENABLE}}$	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

6.3 ATMC EVB Parts List

The following is the part list of all the components that are used on the board.

Table 6-14. ATMC EVB Parts List

Reference Designation	Part Description	Mfr.	Part #
C1 C24 C134 C148 C166 C173 C177 C187 C190 C198 C199	Capacitor 10 µF, 20 V, 10%, SMD Size C, Tantalum	SIEMENS	B45196-H4475-K20
C2 C3 C4 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C25 C26 C27 C28 C29 C30 C31 C32 C33 C34 C35 C36 C37 C38 C39 C40 C41 C43 C44 C45 C46 C47 C48 C49 C50 C51 C52 C53 C54 C55 C56 C57 C58 C59 C60 C62 C55 C56 C57 C58 C59 C60 C62 C70 C71 C72 C73 C74 C75 C76 C77 C78 C79 C80 C81 C82 C83 C84 C85 C86 C87 C88 C89 C90 C91 C92 C93 C94 C95 C96 C97 C98 C99 C100 C101 C102 C103 C104 C105 C106 C107 C109 C110 C111 C112 C113 C114 C115 C116 C117 C118 C119 C120 C121 C122 C123 C124 C125 C126 C127 C128 C129 C130 C131 C132 C133 C135 C136 C137 C138 C139 C140 C141 C142 C143 C144 C145 C146 C147 C150 C151 C152 C153 C154 C155 C156 C157 C158 C159 C161 C162 C163 C165 C174 C175 C176 C178 C179 C180 C181 C182 C183 C185 C188 C189 C193 C194 C195 C196	Capacitor 0.1 µF SMD 1206 Ceramic	SIEMENS	B37872-K5104K
C5	Capacitor 22 µF, 25 V, 10%, SMD, Electrolyte	AVX	TAJD226K025RNJ
C42	Capacitor 5000 pF, 50 V, 10%, SMD 1206, Ceramic	AVX	AV12065C 502K A700J
C61 C168	Capacitor 100 µF, 10 V, 10%, SMD Size D, Tantalum	SIEMENS	B45196-H2107-K10
C108	Capacitor 0.01 µF SMD 1206 Ceramic	SIEMENS	B37872-K5103K

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
C149 C160	Capacitor, 0.47 µF, SMD, 1812 Ceramic	AVX	AV12065C 474K A700J
C164 C167 C169 C170 C171 C172 C186	Capacitor 10 nF, 50 V, 10%, NPO, SMD 1210, Ceramic	VITRAMNON	VJ1210A103KXAT
C184 C191	Capacitor 68 pF, 50 V, 5%, SMD 1206, Ceramic	SIEMENS	B37871-K5680J
C192	Capacitor 3900 pF, 50 V, 5%, COG, SMD 1210 Ceramic	SIEMENS	B37949-K5392J
C197	Capacitor 0.039 µF, 50 V, 5%, SMD 1206, Ceramic	SIEMENS	B37872-K5393J
CT1 CT2 CT3 CT4 CT5 CT6 CT7 CT8 CT9 CT10	Capacitor 100 pF, 50 V, 10%, SMD 1206, Ceramic	S+M	B37871-K5101-J62
D1	Diode SMD	Motorola	LL4004G
D2	Diode Pair, common cathode	Motorola	MBRD620CT
D3	Zener Diode, 5 V SMD	Motorola	1SMC5.0AT3
DS1 DS3 DS4	Dip-Switch, 4 × SPST, SMD	GRAYHILL	90HBW04S
DS2	Dip-Switch, 2 × SPST, SMD	GRAYHILL	90HBW02S
F1	Fuse, 5 A/250 V Miniature 5 × 20 mm, Fast-blow		
H1 H2 H3 H4 H5	Gnd Bridge, Gold Plated	PRECIDIP	999-11-112-10
JP1 JP2 JP5 JP6	Jumper Header, 3-pole with Fabricated Jumper	Molex	SDA-87156-03
JP3 JP4	Jumper Header, 3-pole × 2 with Fabricated Jumper	Samtec	TSM-10301-SDV-JH6
L1 L2	Inductor 8.2 mHy	BOURNS	PT12133
L3 L4 L5 L6 L7 L8 L9 L10	SM Beads, 1 µH	Rite Products	2775021447
LD1 LD2 LD19 LD21 LD22	Led Red SMD	SIEMENS	LS T670-HK
LD3 LD4 LD5 LD6 LD7 LD8 LD15 LD16 LD17 LD18	Led Green SMD	SIEMENS	LG T670-HK
LD9 LD10 LD11 LD12 LD13 LD14 LD20	Led Yellow SMD	SIEMENS	LY T670-HK
P1 P2	Connector 96-pin, receptacle, DIN 41612, 90°	ELCO	268477096002025
P3 P4 P5 P6 P7 P8	HP Logic Analyzer MICTOR38 Connector	AMP	2-767004-2

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
P9	Jumper Header, 5-pole × 2 with Fabricated Jumper	Samtec	TSM-10501-SDV-JH10
P10	Connector 3-pin, Power, Straight, with false insertion protection.	WB	8113S-253303353
P11	Connector 9-pin, receptacle, D-Type, 90°	KCC	DN-09-S-RCZ
P12	Connector 8-pin, RJ45 Receptacle, 90°	KCC	90015-8P8C
P13	Connector 37-pin, plug, D-Type, 90°	KCC	DN-37-P-RCZ
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R13 R14 R15 R16 R17 R19 R21 R23 R24 R25 R29 R30 R31 R32 R34 R35 R37 R39 R40 R41 R42 R43 R46 R47 R48 R49 R50 R51 R54 R55 R57 R59 R63 R66 R67 R70 R73 R76 R79 R83 R84 R85 R86 R87 R88 R89 R99 R100 R101 R102 R103 R104 R107 R112 R113 R114 R120 R121 R122 R123 R124 R133 R134 R135 R136 R137 R138 RT3 RT11	Resistor 10 KΩ, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 010K FC5
R11 R20 R38 R52 R53 R142	Resistor 1 KΩ, 5%, SMD 1206, 1/8 W	AVX	CR32 102F T
R12	Resistor 2 kΩ, 1%, SMD 1206, 1/8 W	BOURNS	CR1206 FX 2001E
R18 R26 R44 R58 R93	Resistor 4.7 KΩ, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 4K7 FCS
R22	Resistor 510 Ω, 1%, SMD 1206, 1/8 W	BOURNS	CR1206 JW 472E
R27	Resistor 47 KΩ, 1%, SMD 1206, 1/8 W	KYOCERA	CR32 473JT
R28 R33 R36 R45 R94 R95 R96 R105	Resistor 75 Ω, 5%, SMD 1206, 1/8 W	DRALORIC	CR1206 100 75RJ
R56 R97 R106 R115 R126	Resistor 100 Ω, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 100R FCS
R60	Resistor 10 Ω, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 10R FCS

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
R61 R62 R64 R65 R68 R69 R71 R72 R74 R75 R77 R78 R80 R81	Resistor 150 Ω, 5% SMD 1206, 1/8 W	BOURNS	CR1206 JW 151 E
R82 R90 R130 R131	Resistor 332 Ω, 5%, SMD 1206, 1/8 W	RODERSTEIN	D25 332R FC5
R91 R98 R109	Resistor 200 Ω, 5%, SMD 1206, 1/8 W	RODERSTEIN	D25 200R FC5
R92	Resistor 412 Ω, 5%, SMD 1206, 1/8 W	RODERSTEIN	D25412R FC5
R108 R116	Resistor 39.1 Ω, 1%, SMD 1206, 1/8 W	TYOHM	RMC 12061/8W 39E
R110 R111	Resistor 237 Ω, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 237R FCS
R117 R118 R119	Resistor 80.6 Ω, 1%, SMD 1206, 1/8 W	TYOHM	RMC 12061/8W 80E
R125	Resistor 0 Ω, SMD 1206, 1/8 W	TYOHM	RMC 1206 0E 1%
R127 R128 R129	Resistor 130 Ω, 5% SMD 1206, 1/8 W	BOURNS	CR1206 JW 131 E
R132	Resistor 294 Ω, 1%, SMD 1206, 1/8 W	TYOHM	RMC 1206 294E 1%
R139 R140 R141	Resistor 22 Ω, 5%, SMD 1206, 1/8 W	RODERSTEIN	D25 24R FCS
R143	Resistor 243 Ω, 1%, SMD 1206, 1/8 W	RODERSTEIN	D25 243R FCS
R144 R145 R146 R147 R148	Resistor 330 Ω, 5%, SMD 1206, 1/8 W	RODERSTEIN	D25 332R FC5
RN1 RN2	Resistor Network 75 Ω, 5%, 8 resistors, 16-pin	BOURNS	4816P 001 750J
RN3	Resistor Network 22 Ω, 5%, 8 resistors, 16-pin	DALE	SOMC 16 03 220J
RT1 RT2 RT4 RT5 RT6 RT7 RT10 RT12 RT13 RT14	Resistor 49.9 Ω, 1%, SMD 1206, 1/8 W	DRALORIC	S13G49R9FJB
RT8 RT9	Resistor 36.5 Ω, 1%, SMD 0603, 1/8 W	DRALORIC	S11G36R5FJ2
SW1	SPDT, push button, RED, Sealed	C & K	KS12R22-CQE

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
SW2	SPDT, push button, BLACK, Sealed	C & K	KS12R23-CQE
U1	MPC860SAR PowerQUICC 19 × 19 BGA Adapter 19 × 19 Socket Thru-hole	Motorola Andon Electronics (AEC)	PPC860SRZP50B 10-19-03-357-273-G04 10-19-03-357-274K-P31
U2	ATM Cell Processor Rev B 272-pin Prototype BGA Socket	Motorola CTI	MC92501GC 8227-272-1-00
U3 U50 U70 U77 U84	Octal Low Voltage CMOS Buffer	Motorola	MC74LCX541
U4 U48 U62 U71	Low Voltage CMOS Octal Transparent Latch	Motorola	MC74LCX573
U5	72-pin 4 Mbyte 60 ns DRAM SIMM 72-pin SIMM Socket	Motorola AMP	MCM36100EXP 822032-4
U6	80-pin 2 Mbyte 120 ns FLASH SIMM 80-pin SIMM Socket	Motorola AMP	MCM29020E 822032-5
U7 U8 U9 U10	72-pin 1 Mbyte 15 ns SRAM SIMM 72-pin Socket	Cypress AMP	CYM1841CP7 822032-4
U11 U53 U54 U57 U58 U60 U61 U65 U66 U67 U69 U73 U75 U79 U83	Low Voltage CMOS Octal Buffer	Motorola	MC74LCX244
U12	5 MHz Clock generator. 3.3 V, CMOS levels. 14-pin SMD Socket	MGR-Tech M-TRON Preci-Dip	MH14F AD5.00M 3.3 110-91-314-41-105
U13 U18	Low Voltage CMOS 18-bit Universal Bus Transceiver	National Semiconductor	74LCX16501
U14	4K × 64 bit Content-Addressable Memory	Motorola	MCM69C232
U15 U27 U30 U31 U37 U38 U39 U40	MACH220 Programmable Logic Device, 10 ns, PLCC 68-pin PLCC Socket	AMD AMP	MACH220-10JC 822280-1
U16 U17 U22 U28 U29	128K × 8-bit SRAM, 5 V, 15 ns	Motorola	MCM6226BBXJ15
U19	PAL16V8, Electrically Erasable, 5 ns, PLCC 20-pin PLCC Socket	AMD AMP	PALCE16V8H-5JC/5 822270-1

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
U20 U21 U25 U26	16K × 9-bit Asynchronous FIFO, 10-ns, PLCC	Cypress	CY7C462A-10JC
U23	25 MHz Clock generator, 3.3 V, CMOS levels. 14-pin SMD Socket	MEC - Modern Enterprise Corporation Preci-Dip	HXO-1255 25.000MHz 110-91-314-41-105
U24	Low Voltage PLL clock Driver	Motorola	MPC931FA
U32 U34	PAL22V10, Electrically Erasable, 5 ns, PLCC 28-pin PLCC Socket	AMD AMP	PALCE22V10H-5JC/5 822272-1
U33 U35	SPARE24		
U36	5 A, 3.3 V Low Dropout Voltage Regulator	Micrel	MIC29500BT
U41	Saturn User Network Interface (S/UNI) Lite, ATM PHY device for 155 Mbps Fiber-Optic Line	PMC-Sierra	PM5346-RC S/UNI LITE
U42	RS-232 Transceiver (3 × 3)	Motorola	MC145707DW
U43	10 Base-T Filter network	Pulse Engineering	PE-68026
U44	Enhanced Ethernet Serial Transceiver.	Motorola	MC68160FB
U45	Buffer Schmitt-Trigger	Motorola	MC74LS244D
U46	19.44 MHz Clock gen., 5 V 14-pin SMD Socket	COMCLK Preci-Dip	CM42AF/H 110-91-314-41-105
U47	ATM Multimode Fiber Transceivers for SONET OC-3/SDH STM-1	Hewlett Packard	HFBR-5205
U49 U59 U76 U85	Resistor Network 10 KΩ, 5%, 13 resistors, 14-pin	DALE	SOMC 14 01 103J
U51 U52 U55 U56	Low Voltage CMOS Octal Transceiver	Motorola	MC74LCX245D
U63	Voltage level detector. Range 2.595 V–2.805 V. O.D. output	Seiko	S-8052ANY-NH-X
U64	Schmitt-Trigger Hex Inverter	Motorola	MC74ACT14D
U68 U72 U74 U78 U82	Low-Voltage CMOS Octal D-Type Flip-Flop	Motorola	MC74LCX574D

Table 6-14. ATMC EVB Parts List (Continued)

Reference Designation	Part Description	Mfr.	Part #
U80 U81	Octal CMOS Buffer, TTL Levels	Motorola	MC74ACT541D
U86	Octal CMOS Buffer	Motorola	MC74AC541D
Y1	Crystal resonator, 20 MHz, SMD, Fundamental Oscillation mode, Frequency tolerance ±50 ppm, Drive-level—1 mW ±0.2 mW, Shunt capacitance—7pF Max., Load capacitance—32pF, Equivalent Series Resistance —50 Ω Max. Insulation Resistance— 500 MΩ at 100 Vdc	MEC - Modern Enterprise Corporation	HC-49/U-SM-3

A

Programmable Logic Equations

The ATMC EVB has 11 programmable logic devices on it. Use is done with MACH220-10 by AMD. These device support the following function on the EVB:

- U40 - Debug Port Controller.
- U38 - auxiliary board control functions, e.g,. buffers control, local interrupter, reset logic, etc.'.
- U39 - BCSR.
- U19 - logic equations for the CAM.
- U32 - logic equations for the Ingress Generator.
- U34 - logic equations for the Egress Generator.
- U31 - Ingress Generator equations.
- U30 - Egress Generator and Ingress Recorder equations.
- U37 - Egress Recorder equations.
- U15,U27 - Address Counter for the recorders.

A.1 U40—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 (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~ 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  'MPC860ADS Debug Port Controller.
        Originated for Ptec ADS, Shlomo Reches (MSIL) - October 26, 1993
        Modified for Pda ADS, Yair Liebman - (MSIL) - April 04, 1995
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************
U40 device 'mach220a';

"*****************************************************************************
"* #######
*
"* #        #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #         #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####      ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #          ##       #    #       #####   #  # #  ######  #            #    *
"* #         #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************


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

HstReq            PIN 20 ;          "Host to ADS, write pulse. (IN)

AdsAck            PIN 31 ISTYPE 'reg, buffer';  "ADS to host, write ack (OUT,3s)

AdsReq            PIN 2 ISTYPE 'reg, buffer';   "ADS to host, write signal. (OUT,3s)

HstAck            PIN 54;          "Host to ADS, write ack.   (IN)

AdsHardReset      PIN 50;          "Host to ADS, Hard reset.  (IN)

AdsSoftReset      PIN 17;          "Host to ADS, Soft reset.  (IN)

HstEn~            PIN 3;           "Host connected to ADS.    (IN)

HostVcc           PIN 49;          "Host to ADS, host is on.  (IN)
```

```
D_C~              PIN 51;          "Host to ADS, select data
                                        "or control access.        (IN)

AdsSel0,
AdsSel1,
AdsSel2           PIN 22, 21, 9;  "Host to ADS, card addr.    (IN)

AdsAddr0,
AdsAddr1,
AdsAddr2          PIN 7, 6, 5; "ADS board address switch. (IN)

AdsSelect~        NODE ISTYPE 'com, buffer'; "ADS selection indicator. (OUT)


"*************************************************************************
"* PDA pins. Including debug port.                                       *
"*************************************************************************
PdaHardReset~     PIN 40;        "Pda's hard reset input.  (I/O. o.d.)
PdaSoftReset~     PIN 65;        "Pda's soft reset output. (I/O. o.d.)
VFLS0, VFLS1      PIN 10, 11;    "Debug/Trap mode, report. (IN)
DSCK              PIN 48 istype 'com'; "Pda's debug port clock. (Out)
DSDI              PIN 47 istype 'com'; "Pda's debug serial data in (Out)
DSDO                  PIN 4;   " Pda's debug serial data output (In)
"*************************************************************************
"* Mach to ADI data bus.                                                 *
"*************************************************************************
PD7,
PD6,
PD5,
PD4,
PD3,
PD2,
PD1,
PD0               PIN 66,60,67,59,58,57,56,55;  "ADI data bus.(I/O)
"*************************************************************************
"* Clock gen pins.                                                       *
"*************************************************************************
DbgClk            PIN 16;                "Debug Clock input source. (IN)
DbgClkOut         PIN 33 istype 'com' ;        " to be connected to IClk. (out)

Clk2              PIN 14 istype 'reg, buffer'; "IClk divided by 2 (Out) (Out for testing,
                                                      "   may be node)
IClk              PIN 15;           " Connected to Clkout externally (In)
"*************************************************************************
"* Misc.
"*************************************************************************
Run               PIN 23 istype 'com';   "external indication

"*************************************************************************
"*   ###                                                                 *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*************************************************************************
"*************************************************************************
"* 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_HstReq          NODE istype 'reg';   "sync. host req.
DS_HstReq         NODE istype 'reg';  " double sync. host req.

S_D_C~            NODE istype 'reg, buffer';  " synchronized data/ control selection

S_HstAck          NODE istype 'reg, buffer'; " sync host ack
DS_HstAck         NODE istype 'reg, buffer'; " double sync host ack

BundleDelay1,
BundleDelay0      NODE istype 'reg, buffer';  "delay counter for bundle delay compensa-
tion
BndTmrExp         NODE istype 'com';           " terminal count for bundle delay timer.
PDOe              NODE istype 'com';               "Mach to ADI data OE.

PdaHardResetEn    NODE istype 'com';    " enables hard reset buffer.
PdaSoftResetEn    NODE istype 'com';    " enables soft reset buffer.

"*****************************************************************************
"* Tx Shift Register                                                         *
"*****************************************************************************
TxReg7,
TxReg6,
TxReg5,
TxReg4,
TxReg3,
TxReg2,
TxReg1,
TxReg0            NODE istype 'reg, buffer'; " Transmit latch and shift register


"*****************************************************************************
"* Tx Control Logic                                                          *
"*****************************************************************************
TxWordLen3,
TxWordLen2,
TxWordLen1,
```

```
TxWordLen0        NODE istype 'reg, buffer'; " Counter, counts (on fast clock,
                                             " to gain 1/2 clock resolution)
                                             " transmission length

TxWordEnd         NODE istype 'com';         " Terminal count, sets transmission length.
TxEn              NODE istype 'reg, buffer';    " Transmit Enable.
TxClkSns          NODE istype 'reg, buffer';  " transmit clock polarity

"*****************************************************************************
"* Rx Shift Register                                                         *
"*****************************************************************************
RxReg0            NODE istype 'reg, buffer'; " receive shift register and latch
"*****************************************************************************
"* Rx Control Logic                                                          *
"*****************************************************************************
DsdiEn            NODE istype 'reg';           " enables dsdi towards

"*****************************************************************************
"* ADI control & status register bits.                                       *
"*****************************************************************************
StatusRequest~    NODE istype 'reg, buffer';    "Status request
DebugEntry~       NODE istype 'reg, buffer';    "Debug enable after reset (L)
DiagLoopBack~     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~, PdaSoftReset~];

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

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

HOST_IS_ON = ((HstEn~==HOST_EN~_ACTIVE) & (HostVcc==HOST_VCC_ACTIVE));
HOST_IS_OFF = !HOST_IS_ON;

BOARD_IS_SELECTED = 0;
ADS_IS_SELECTED = (AdsSelect~.fb==BOARD_IS_SELECTED) ;

"Data_Cntrl~ 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~.fb==BOARD_IS_SELECTED) &
                              (DS_HstReq.fb ==HOST_REQ_ACTIVE) &
                   (AdsAck==!ADS_ACK_ACTIVE) &
                              (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_ADI_CONTROL = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                           (DS_HstReq.fb ==HOST_REQ_ACTIVE) &
                           (AdsAck==!ADS_ACK_ACTIVE) &
                           (D_C~==CONTROL) &
                           (S_D_C~.fb == CONTROL) &
                          (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_ADI_DATA = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                        (DS_HstReq.fb==HOST_REQ_ACTIVE) &
                        (AdsAck==!ADS_ACK_ACTIVE) &
                        (D_C~==DATA) &
                        (S_D_C~.fb ==DATA) &
                           (HstAck==!HOST_ACK_ACTIVE) );

HOST_WRITE_COMPLETE = ( (AdsSelect~.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;


IS_STATUS_REQUEST = (StatusRequest~.fb == STATUS_REQUEST);
DEBUG_MODE_ENTRY  = (DebugEntry~.fb == DEBUG_ENTRY);
IN_DIAG_LOOP_BACK = (DiagLoopBack~.fb == DIAG_LOOP_BACK);
IS_IN_DEBUG_MODE  = (InDebugMode.fb == IN_DEBUG_MODE);


"*****************************************************************************
"* 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~.fb==BOARD_IS_SELECTED) &
                  (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                  (AdsReq==ADS_REQ_ACTIVE) &
                  (HstReq==!HOST_REQ_ACTIVE) );

HOST_READ_ADI_DATA = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                       (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                       (HstReq==!HOST_REQ_ACTIVE) &
                           (AdsReq==ADS_REQ_ACTIVE) &
                       (D_C~==DATA) );

HOST_READ_ADI_CONTROL = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                          (DS_HstAck.fb==HOST_ACK_ACTIVE) &
                          (HstReq==!HOST_REQ_ACTIVE) &
                           (AdsReq==ADS_REQ_ACTIVE) &
                          (D_C~==CONTROL) );


ADS_SEND_STATUS = ((AdsSelect~.fb==BOARD_IS_SELECTED)&
                   (DS_HstReq.fb == !HOST_REQ_ACTIVE) &
                   (D_C~==CONTROL) &
                   (AdsAck==ADS_ACK_ACTIVE) &
                  IS_STATUS_REQUEST );

"*****************************************************************************
"* ADI Data Bus definitions
"*****************************************************************************
DATA_BUFFERS_ENABLE = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
```

```
                                        (HstAck==HOST_ACK_ACTIVE) &
                                        (HstReq==!HOST_REQ_ACTIVE) );

STATUS_WORD_ON_ADI_BUS = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                           (HstAck==HOST_ACK_ACTIVE) &
                           (HstReq==!HOST_REQ_ACTIVE) &
                           (D_C~==CONTROL) );

READ_DATA_WORD_ON_ADI_BUS = ( (AdsSelect~.fb==BOARD_IS_SELECTED) &
                            (HstAck==HOST_ACK_ACTIVE) &
                            (HstReq==!HOST_REQ_ACTIVE) &
                            (D_C~== DATA) );

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


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

!AdsSelect~ = 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 status

   PdaRst.clk = IClk;

   PdaRst := (!PdaHardReset~ # !PdaSoftReset~) &
                  (AdsSelect~.fb==BOARD_IS_SELECTED);   " synchronized inside.

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

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

   !PdaHardReset~ = H;

   PdaHardReset~.oe = PdaHardResetEn;  "open-drain

   PdaHardResetEn = ADS_IS_SELECTED & (AdsHardReset==ADS_HARD_RESET_ACTIVE);

   !PdaSoftReset~ = H;

   PdaSoftReset~.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~.clk = IClk;" synchronizing D_C~ selector
   S_D_C~ := D_C~;

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~ 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~ line is in control mode. It also may be read when StatusRequest~ bit
"* is active.
"*
"* Control register bits description:
"*
"* DebugEntry~: (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~: (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~: (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~ line is high (data) the Rx shift register contents is driven. If
"* D_C~ 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 pda is in debug mode,
"*             i.e., VFLS(0:1) lines are driven high.
"* 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~ 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;
   else  when ( STATUS_WORD_ON_ADI_BUS ) then
                    PD  =  [PdaRst.fb,  TxError.fb,  InDebugMode.fb,  DbgClkDivSel1.fb,
DbgClkDivSel0.fb,
                           StatusRequest~.fb,  DiagLoopBack~.fb, DebugEntry~.fb ];

   InDebugMode.clk = IClk;

   InDebugMode := VFLS1 & VFLS0;  "synchronized.

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

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

   when ( ADS_IS_SELECTED & !PdaSoftReset~ ) then
         DSCK = H;                          "debug mode enable
   else when ( ADS_IS_SELECTED & !TxEn.fb & PdaSoftReset~ ) then
              DSCK = !DebugEntry~.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~) then
```

```
        DSDI = H;
   else when (ADS_IS_SELECTED & !STATE_DSDI_ENABLED & PdaSoftReset~ ) then
        DSDI = L;
   else when (ADS_IS_SELECTED & STATE_DSDI_ENABLED) then
          DSDI = TxReg7.fb;
   else
        DSDI = L;



"*****************************************************************************
"* 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;

end dbg_prt7
```

A.2 U38—Auxiliary Board Control Functions

```
”*****************************************************************************
“* 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.
“*****************************************************************************

module cnt_reg1
```

```
title  'M92501 Board Control and Status Register.
        Originated for MPC860ADS, Yair Liebman (MSIL) - April 04, 1995
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************
U38 device 'mach220a';

"*****************************************************************************
"* #######
*
"* #         #    #   #####  ######  #####   #    #    ##    #         ####     *
"* #          #  #      #    #       #    #  ##   #   #  #   #        #         *
"* #####       ##       #    #####   #    #  # #  #  #    #  #         ####     *
"* #           ##       #    #       #####   #  # #  ######  #             #    *
"* #          #  #      #    #       #   #   #   ##  #    #  #        #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************


"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************
SYSCLK                      PIN 15;

RGPORIn                     PIN 49;
ResetConf~                  PIN 16;

BrdContRegCs~PIN 50;
BTA~                         PIN 17;
R_W~                         PIN 20;

A27                                 PIN 59;
A28                                 PIN 51;
A29                                 PIN 54;

D0                                  PIN 28;
D1                                  PIN 29;
D2                                  PIN 30;
D3                                  PIN 31;
D4                                  PIN 36;
D5                                  PIN 32;
D6                                  PIN 2;
D7                                  PIN 26;
D8                                  PIN 24;
D9                                  PIN 9;
D10                                 PIN 6;
D11                                 PIN 14;
D12                                 PIN 33;
D13                                 PIN 37;
D14                                 PIN 41;
D15                                 PIN 7;

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

```
FlashEn~                    PIN  44 istype 'reg,buffer';   " flash enable.
DramEn~                     PIN  55 istype 'reg,buffer';   " dram enable
EthEn~                      PIN  46 istype 'reg,buffer';   " ethernet port enable
FlashCfgEn~                 PIN  25 istype 'reg,buffer';   " flash configuration enable
CntRegEn~                   PIN  13 istype 'reg,buffer';   " control register access en-
able
RS232En1~                   PIN  48 istype 'reg,buffer';   " RS232 port 1 enable
HalfWord~                   PIN  38 istype 'reg,buffer';   " 32/16 bit dram operation se-
lect
GENRECEn~                   PIN   3 istype 'reg,buffer';   " one of the Generators/Record-
ers is
                                                " enabled.
IGGENEn~                    PIN   4 istype 'reg,buffer';   " Ingress Generator Enabled
IGPHYEn~                    PIN   5 istype 'reg,buffer';   " Ingress PHY Enabled
UNI~                        PIN  12 istype 'reg,buffer';    " EVB is connected to a UNI
Switch
EGPHYEn~                    PIN  39 istype 'reg,buffer';   " Egress PHY Enabled
EGRECEn~                    PIN  40 istype 'reg,buffer';   " Egress Recorder Enabled
SWIGREn~                    PIN  62 istype 'reg,buffer';   " Switch Ingress Recorder En-
abled
SWEGGEn~                    PIN  63 istype 'reg,buffer';   " Switch Egress Generator En-
abled
BCSR4Rd~                    PIN  64 istype 'com,buffer';    " BCSR #4 READ Cycle control
signal

"*****************************************************************************
"* Board Status Pins. Read only.
"*****************************************************************************
FlashPD7                    PIN 66;
FlashPD6                    PIN 65;
FlashPD5                    PIN 67;
FlashPD4                    PIN 43;
FlashPD3                    PIN 23;
FlashPD2                    PIN 22;
FlashPD1                    PIN 21;  " Flash presence detect lines

DramPD4                     PIN 10;
DramPD3                     PIN 60;
DramPD2                     PIN 56;
DramPD1                     PIN 45;  " Dram SIMM Identification pins

ExtToolI0                   PIN 57;
ExtToolI1                   PIN 58;
ExtToolI2                   PIN 47;
ExtToolI3                   PIN 11;  " External Tools Identification pins

"*****************************************************************************
"* Auxiliary Pins.
"*****************************************************************************

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************
```

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


ERB          NODE istype 'reg,buffer';   " External Arbitration
IP~          NODE istype 'reg,buffer';   " Interrupt Prefix in MSR
BDIS              NODE istype 'reg,buffer';   " Boot Disable
RSV2              NODE istype 'reg,buffer';   " reserved config bit 2
BPS0,
BPS1              NODE istype 'reg,buffer';   " Boot Port Size
RSV6              NODE istype 'reg,buffer';   " reserved config bit 6
ISB0,
ISB1              NODE istype 'reg,buffer';   " Internal Space Base
DBGC0,
DBGC1             NODE istype 'reg,buffer';   " Debug pins Config.
DBPC0,
DBPC1             NODE istype 'reg,buffer';   " Debug Port pins Config
EBDF0,
EBDF1   NODE istype 'reg,buffer';   " External Buse Division  Factor
RSV15   NODE istype 'reg,buffer';   " reserved config bit 15
DataOe             NODE istype 'com';        " data bus output enable on read.


"*****************************************************************************
"* Control & Status Registers write/read.
"*****************************************************************************

PDA_WRITE_REG0~  NODE istype 'com';
PDA_WRITE_REG1~  NODE istype 'com';
PDA_WRITE_REG2~  NODE istype 'com';

PDA_READ_CONF_REG~  NODE istype 'com';
PDA_READ_CONT_REG1~ NODE istype 'com';
PDA_READ_STAT_REG1~ NODE istype 'com';
PDA_READ_STAT_REG2~ NODE istype 'com';
PDA_READ_STAT_REG3~  NODE istype 'com';


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

CntRegEnProtect~ NODE istype 'reg,buffer';




"*****************************************************************************
"*  #####                                                                    *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####         *
"* #         #    #  ##   #  #          #     #  #   ##   #     #           *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #           *
"* #         #    #  #  # #       #     #    ######  #  # #     #           *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #           *
"*  #####    ####   #    #   ####      #    #    #  #    #     #           *
"*                                                                           *
"* ######                                                                   *
"* #     #  ######   ####   #         ##    #####                           *
```

```
"* #     #  #       #    #  #        #  #   #    #                          *
"* #     #  #####   #       #       #    #   #    #  #####                   *
"* #     #  #       #       #       ######  #####                           *
"* #     #  #       #    #  #       #    #  #   #                           *
"* ######   ######   ####   ######  #    #  #    #                          *
"*                                                                           *
"*    ##     #####     #     ####   #    #                                   *
"*   #  #      #       #    #    #  ##   #                                   *
"*  #    #     #       #    #    #  # #  #                                   *
"*  ######     #       #    #    #  #  # #                                   *
"*  #    #     #       #    #    #  #   ##                                   *
"*  #    #     #       #     ####   #    #                                   *
"*****************************************************************************
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,A28,A29];
Data = [D0..D15];


ConfigReg = [ERB,IP~,RSV2,BDIS,BPS0,BPS1,RSV6,ISB0,ISB1,DBGC0,DBGC1,DBPC0,
                                                          DBPC1,EBDF0,EBDF1,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
EBDF = [EBDF0,EBDF1];" External Bus Division Factor

ContReg = [FlashEn~,DramEn~,EthEn~,SWIGREn~,FlashCfgEn~,CntRegEnProtect~,CntRegEn~,
           RS232En1~,IGGENEn~,IGPHYEn~,EGRECEn~,EGPHYEn~,HalfWord~,SWEGGEn~,UNI~,0];

ReadConfReg = [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,EBDF0.fb,EBDF1.fb,RSV15.fb];

ReadContReg1 = [FlashEn~,DramEn~,EthEn~,SWIGREn~,FlashCfgEn~,CntRegEnProtect~.fb,
  CntRegEn~,RS232En1~,IGGENEn~,IGPHYEn~,EGRECEn~,EGPHYEn~,HalfWord~,SWEGGEn~,UNI~,0];

ReadStatReg1 = [FlashPD4,FlashPD3,FlashPD2,FlashPD1,0,DramPD4,DramPD3,
                DramPD2,DramPD1,ExtToolI0,ExtToolI1,ExtToolI2,ExtToolI3,0,0,0];

ReadStatReg2                          =                          [0,0,0,0,0,CntRegEnPro-
tect~.fb,0,0,0,FlashPD7,FlashPD6,FlashPD5,0,0,0,0];

DrivenContReg = [FlashEn~,DramEn~,EthEn~,SWIGREn~,FlashCfgEn~,CntRegEn~,
           RS232En1~,IGGENEn~,IGPHYEn~,EGRECEn~,EGPHYEn~,HalfWord~,SWEGGEn~,UNI~];

WideContReg = [FlashEn~,DramEn~,EthEn~,SWIGREn~,FlashCfgEn~,CntRegEnProtect~,
         CntRegEn~,RS232En1~,IGGENEn~,IGPHYEn~,EGRECEn~,EGPHYEn~,Half-
Word~,SWEGGEn~,UNI~];

StatRegIn = [FlashPD4,FlashPD3,FlashPD2,FlashPD1,0,DramPD4,DramPD3,
```

```
DramPD2,DramPD1,ExtToolI0,ExtToolI1,ExtToolI2,ExtToolI3,X,X,X];

FlashPdHigh = [FlashPD7,FlashPD6,FlashPD5];

"*****************************************************************************
"* 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 = ((ResetConf~ == RESET_CONFIG_ACTIVE) &
                                        (FlashCfgEn~ != FLASH_CFG_ENABLE));

"*****************************************************************************
"* Register Access definitions
"*****************************************************************************

CONFIG_REG_ADD = 0;
CONTROL_REG_ADD = 1;
STATUS_REG1_ADD = 2;
STATUS_CONTROL_REG2_ADD = 3;
STATUS_REG3_ADD = 4;

BCSR_WRITE_ACTIVE = 0;

PDA_WRITE_CONFIG_REG = (PDA_WRITE_REG0~ == BCSR_WRITE_ACTIVE);
PDA_WRITE_CONTROL_REG1 = (PDA_WRITE_REG1~ == BCSR_WRITE_ACTIVE);
PDA_WRITE_CONTROL_REG2 = (PDA_WRITE_REG2~ == BCSR_WRITE_ACTIVE);

BCSR_READ_ACTIVE = 0;
PDA_READ = (!BrdContRegCs~ & R_W~ & !CntRegEn~);

PDA_READ_CONFIG_REG = (PDA_READ_CONF_REG~ == BCSR_READ_ACTIVE);
PDA_READ_CONTROL_REG1 = (PDA_READ_CONT_REG1~ == BCSR_READ_ACTIVE);
PDA_READ_STATUS_REG1 = (PDA_READ_STAT_REG1~ == BCSR_READ_ACTIVE);
PDA_READ_STATUS_REG2 = (PDA_READ_STAT_REG2~ == BCSR_READ_ACTIVE);
PDA_READ_STATUS_REG3 = (PDA_READ_STAT_REG3~ == BCSR_READ_ACTIVE);

"*****************************************************************************
"* Config Reg 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;

CLKOUT_IS_GCLK2 = 0;
CLKOUT_IS_GCLK2_DIVIDE_2 = 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_FOR_SHOW;

DBPC_PON_DEFAULT = DEBUG_PORT_ON_JTAG;

EBDF_PON_DEFAULT = CLKOUT_IS_GCLK2;

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];
EBDF_DATA_BIT = [D13,D14];
RSV15_DATA_BIT = [D15];

"*****************************************************************************
"* Control Register 1 definitions.
"*****************************************************************************

HALF_WORD = 0;

ETH_ENABLED = 0;

DRAM_ENABLED = 0;

ETH_LOOP = 1;

TPSQEL = 0;

TP_FULL_DUP = 0;

CONT_REG_ENABLE = 0;

RS232_ENABLE_1 = 0;

FLASH_ENABLED = 0;

INGRESS_GENERATOR_ENABLED = 0;

INGRESS_PHY_ENABLED = 0;

EGRESS_RECORDER_ENABLED = 0;

EGRESS_PHY_ENABLED = 0;
```

```
SWITCH_INGRESS_RECORDER_ENABLED = 0;

SWITCH_EGRESS_GENERATOR_ENABLED = 0;

UNI_READY = 0;

"* FLASH_CFG_ENABLE = 0;  needed to be defined ealier

CNT_REG_EN_PROTECT = 0; " inadvertant write protect


"*********************************************
"******* Power On Defaults Assignments *********
"*********************************************
"@ifdef DRAM_8_BIT_OPERATION {
"
"        HALF_WORD_PON_DEFAULT = HALF_WORD;
"}
"@ifndef DRAM_8_BIT_OPERATION {
"
"        HALF_WORD_PON_DEFAULT = !HALF_WORD;
"}

ETH_ENABLE_PON_DEFAULT = !ETH_ENABLED;

DRAM_ENABLE_PON_DEFAULT = DRAM_ENABLED;

CONT_REG_ENABLE_PON_DEFAULT = CONT_REG_ENABLE;

RS232_ENABLE_1_PON_DEFAULT = !RS232_ENABLE_1;

FLASH_ENABLE_PON_DEFAULT = FLASH_ENABLED;

HALF_WORD_PON_DEFAULT = !HALF_WORD;

ING_GEN_ENABLE_PON_DEFAULT = !INGRESS_GENERATOR_ENABLED;

ING_PHY_ENABLE_PON_DEFAULT = !INGRESS_PHY_ENABLED;

EG_REC_ENABLE_PON_DEFAULT = !EGRESS_RECORDER_ENABLED;

EG_PHY_ENABLE_PON_DEFAULT = !EGRESS_PHY_ENABLED;

SW_ING_REC_ENABLE_PON_DEFAULT = !SWITCH_INGRESS_RECORDER_ENABLED;

SW_EG_GEN_ENABLE_PON_DEFAULT = !SWITCH_EGRESS_GENERATOR_ENABLED;

FLASH_CFG_ENABLE_PON_DEFAULT = !FLASH_CFG_ENABLE;

UNI_PON_DEFAULT = !UNI_READY;

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];
SW_ING_RECORDER_ENABLE_DATA_BIT = [D3];
FLASH_CFG_ENABLE_DATA_BIT = [D4];
CNT_REG_EN_PROTECT_DATA_BIT = [D5];
CONT_REG_ENABLE_DATA_BIT = [D6];
RS232_ENABLE_1_DATA_BIT = [D7];
ING_GENERATOR_ENABLE_DATA_BIT = [D8];
ING_PHY_ENABLE_DATA_BIT = [D9];
EG_RECORDER_ENABLE_DATA_BIT = [D10];
EG_PHY_ENABLE_DATA_BIT = [D11];
HALF_WORD_DATA_BIT = [D12];
SW_EG_GENERATOR_ENABLE_DATA_BIT = [D13];
UNI_DATA_BIT = [D14];

"*****************************************************************************
"* Control Register 2 definitions.
"*****************************************************************************


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


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

"*****************************************************************************
"* 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~ = !ResetConf~ #
                        RGPORIn;
```

```
}

PDA_WRITE_REG0~ = !(!BrdContRegCs~ & !BTA~ & !R_W~ & !A27 & !A28 & !A29 &
                                                                !CntRegEn~);
PDA_WRITE_REG1~ = !(!BrdContRegCs~ & !BTA~ & !R_W~ & !A27 & !A28 & A29 &
                                                                !CntRegEn~);
PDA_WRITE_REG2~ = !(!BrdContRegCs~ & !BTA~ & !R_W~ & !A27 & A28 & A29 &
                                                                !CntRegEn~);

PDA_READ_CONF_REG~ = !(!BrdContRegCs~ & R_W~ & !A27 & !A28 & !A29 & !CntRegEn~);
PDA_READ_CONT_REG1~ = !(!BrdContRegCs~ & R_W~ & !A27 & !A28 & A29 & !CntRegEn~);
PDA_READ_STAT_REG1~ = !(!BrdContRegCs~ & R_W~ & !A27 & A28 & !A29 & !CntRegEn~);
PDA_READ_STAT_REG2~ = !(!BrdContRegCs~ & R_W~ & !A27 & A28 & A29 & !CntRegEn~);
PDA_READ_STAT_REG3~ = !(!BrdContRegCs~ & R_W~ & A27 & !A28 & !A29 & !CntRegEn~);

ConfigReg.clk = SYSCLK;

state_diagram ERB
        state INTERNAL_ARBITRATION:
                if (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                        (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 (PDA_WRITE_CONFIG_REG &
                            (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 (PDA_WRITE_CONFIG_REG &
                            (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 (PDA_WRITE_CONFIG_REG &
                     (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 (PDA_WRITE_CONFIG_REG &
                     (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 (PDA_WRITE_CONFIG_REG &
                     (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 (PDA_WRITE_CONFIG_REG &
                     (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 (PDA_WRITE_CONFIG_REG &
```

```
                (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
                  (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 (PDA_WRITE_CONFIG_REG &
                  (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
          (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
                 (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 (PDA_WRITE_CONFIG_REG &
```

```
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 (PDA_WRITE_CONFIG_REG &
                (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 EBDF
         state CLKOUT_IS_GCLK2:
                  if (PDA_WRITE_CONFIG_REG &
                   (EBDF_DATA_BIT.pin == CLKOUT_IS_GCLK2_DIVIDE_2) &
                   (!KA_PON_RESET # (EBDF_PON_DEFAULT != CLKOUT_IS_GCLK2)) #
                   (KA_PON_RESET & (EBDF_PON_DEFAULT == CLKOUT_IS_GCLK2_DIVIDE_2)) )
                  then
                           CLKOUT_IS_GCLK2_DIVIDE_2
                  else
                           CLKOUT_IS_GCLK2;
         state CLKOUT_IS_GCLK2_DIVIDE_2:
                  if (PDA_WRITE_CONFIG_REG &
                   (EBDF_DATA_BIT.pin == CLKOUT_IS_GCLK2) &
                   (!KA_PON_RESET # (EBDF_PON_DEFAULT != CLKOUT_IS_GCLK2_DIVIDE_2)) #
                   (KA_PON_RESET & (EBDF_PON_DEFAULT == CLKOUT_IS_GCLK2)) )
                  then
                           CLKOUT_IS_GCLK2
                  else
                           CLKOUT_IS_GCLK2_DIVIDE_2;
"*****************************************************************************
state_diagram RSV15
         state !RSV15_ACTIVE:
                  if (PDA_WRITE_CONFIG_REG &
                   (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 (PDA_WRITE_CONFIG_REG &
                   (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;
"*****************************************************************************
"* Control Register.
"*****************************************************************************
equations

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


GENRECEn~.clk = SYSCLK;
GENRECEn~ := IGGENEn~.fb & SWEGGEn~.fb & EGRECEn~.fb & SWIGREn~.fb;

state_diagram FlashEn~
         state FLASH_ENABLED:
                  if (PDA_WRITE_CONTROL_REG1 &
                   (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 (PDA_WRITE_CONTROL_REG1 &
                 (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 (PDA_WRITE_CONTROL_REG1 &
                 (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 (PDA_WRITE_CONTROL_REG1 &
                 (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 (PDA_WRITE_CONTROL_REG1 &
                 (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 (PDA_WRITE_CONTROL_REG1 &
                 (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 UNI~
        state UNI_READY:
                if (PDA_WRITE_CONTROL_REG1 &
                 (UNI_DATA_BIT.pin == !UNI_READY) &
                 (!KA_PON_RESET # (UNI_PON_DEFAULT != UNI_READY)) #
                 (KA_PON_RESET & (UNI_PON_DEFAULT == !UNI_READY)) )
```

```
                then
                        !UNI_READY
                else
                        UNI_READY;
        state !UNI_READY:
                if (PDA_WRITE_CONTROL_REG1 &
                 (UNI_DATA_BIT.pin == UNI_READY) &
                 (!KA_PON_RESET # (UNI_PON_DEFAULT != !UNI_READY)) #
                 (KA_PON_RESET & (UNI_PON_DEFAULT == UNI_READY)) )
                then
                        UNI_READY
                else
                        !UNI_READY;
"*****************************************************************************
state_diagram FlashCfgEn~
        state FLASH_CFG_ENABLE:
                if (PDA_WRITE_CONTROL_REG1 &
                (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 (PDA_WRITE_CONTROL_REG1 &
                (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 (PDA_WRITE_CONTROL_REG2 &
                (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 (PDA_WRITE_CONTROL_REG2 &
                 (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)) #
                PDA_WRITE_CONTROL_REG1)            " any write to control reg 1
              then
                        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 (PDA_WRITE_CONTROL_REG1 &
                 (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 (PDA_WRITE_CONTROL_REG1 &
                 (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_ENABLE_1:
                if (PDA_WRITE_CONTROL_REG1 &
                 (RS232_ENABLE_1_DATA_BIT.pin == !RS232_ENABLE_1) &
                 (!KA_PON_RESET # (RS232_ENABLE_1_PON_DEFAULT != RS232_ENABLE_1)) #
                 (KA_PON_RESET & (RS232_ENABLE_1_PON_DEFAULT == !RS232_ENABLE_1)) )
                then
                        !RS232_ENABLE_1
                else
                        RS232_ENABLE_1;
        state !RS232_ENABLE_1:
                if (PDA_WRITE_CONTROL_REG1 &
                 (RS232_ENABLE_1_DATA_BIT.pin == RS232_ENABLE_1) &
                 (!KA_PON_RESET # (RS232_ENABLE_1_PON_DEFAULT != !RS232_ENABLE_1)) #
                 (KA_PON_RESET & (RS232_ENABLE_1_PON_DEFAULT == RS232_ENABLE_1)) )
                then
                        RS232_ENABLE_1
                else
                        !RS232_ENABLE_1;


"*************************************************************************
state_diagram IGGENEn~
        state INGRESS_GENERATOR_ENABLED:
```

```
                if (PDA_WRITE_CONTROL_REG1 &
                 (ING_GENERATOR_ENABLE_DATA_BIT.pin == !INGRESS_GENERATOR_ENABLED) &
                 (!KA_PON_RESET #
                 (ING_GEN_ENABLE_PON_DEFAULT != INGRESS_GENERATOR_ENABLED)) #
                 (KA_PON_RESET &
                 (ING_GEN_ENABLE_PON_DEFAULT == !INGRESS_GENERATOR_ENABLED)) )
                then
                        !INGRESS_GENERATOR_ENABLED
                else
                        INGRESS_GENERATOR_ENABLED;
        state !INGRESS_GENERATOR_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (ING_GENERATOR_ENABLE_DATA_BIT.pin == INGRESS_GENERATOR_ENABLED) &
                 (!KA_PON_RESET #
                 (ING_GEN_ENABLE_PON_DEFAULT != !INGRESS_GENERATOR_ENABLED)) #
                 (KA_PON_RESET &
                 (ING_GEN_ENABLE_PON_DEFAULT == INGRESS_GENERATOR_ENABLED)) )
                then
                        INGRESS_GENERATOR_ENABLED
                else
                        !INGRESS_GENERATOR_ENABLED;
"*****************************************************************************
state_diagram IGPHYEn~
        state INGRESS_PHY_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                (ING_PHY_ENABLE_DATA_BIT.pin == !INGRESS_PHY_ENABLED) &
                (!KA_PON_RESET#(ING_PHY_ENABLE_PON_DEFAULT != INGRESS_PHY_ENABLED)) #
                (KA_PON_RESET&(ING_PHY_ENABLE_PON_DEFAULT == !INGRESS_PHY_ENABLED)))
                then
                        !INGRESS_PHY_ENABLED
                else
                        INGRESS_PHY_ENABLED;
        state !INGRESS_PHY_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                (ING_PHY_ENABLE_DATA_BIT.pin == INGRESS_PHY_ENABLED) &
                (!KA_PON_RESET#(ING_PHY_ENABLE_PON_DEFAULT != !INGRESS_PHY_ENABLED))#
                (KA_PON_RESET & (ING_PHY_ENABLE_PON_DEFAULT == INGRESS_PHY_ENABLED)))
                then
                        INGRESS_PHY_ENABLED
                else
                        !INGRESS_PHY_ENABLED;
"*****************************************************************************
state_diagram EGPHYEn~
        state EGRESS_PHY_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (EG_PHY_ENABLE_DATA_BIT.pin == !EGRESS_PHY_ENABLED) &
                 (!KA_PON_RESET # (EG_PHY_ENABLE_PON_DEFAULT != EGRESS_PHY_ENABLED)) #
                 (KA_PON_RESET & (EG_PHY_ENABLE_PON_DEFAULT == !EGRESS_PHY_ENABLED)))
                then
                        !EGRESS_PHY_ENABLED
                else
                        EGRESS_PHY_ENABLED;
        state !EGRESS_PHY_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (EG_PHY_ENABLE_DATA_BIT.pin == EGRESS_PHY_ENABLED) &
                 (!KA_PON_RESET # (EG_PHY_ENABLE_PON_DEFAULT != !EGRESS_PHY_ENABLED))#
                 (KA_PON_RESET & (EG_PHY_ENABLE_PON_DEFAULT == EGRESS_PHY_ENABLED)) )
                then
                        EGRESS_PHY_ENABLED
```

```
                    else
                              !EGRESS_PHY_ENABLED;
"*****************************************************************************
state_diagram EGRECEn~
          state EGRESS_RECORDER_ENABLED:
                    if (PDA_WRITE_CONTROL_REG1 &
                     (EG_RECORDER_ENABLE_DATA_BIT.pin == !EGRESS_RECORDER_ENABLED) &
                     (!KA_PON_RESET #
                     (EG_REC_ENABLE_PON_DEFAULT != EGRESS_RECORDER_ENABLED)) #
                     (KA_PON_RESET &
                     (EG_REC_ENABLE_PON_DEFAULT == !EGRESS_RECORDER_ENABLED)) )
                    then
                              !EGRESS_RECORDER_ENABLED
                    else
                              EGRESS_RECORDER_ENABLED;
          state !EGRESS_RECORDER_ENABLED:
                    if (PDA_WRITE_CONTROL_REG1 &
                     (EG_RECORDER_ENABLE_DATA_BIT.pin == EGRESS_RECORDER_ENABLED) &
                     (!KA_PON_RESET #
                     (EG_REC_ENABLE_PON_DEFAULT != !EGRESS_RECORDER_ENABLED)) #
                     (KA_PON_RESET &
                     (EG_REC_ENABLE_PON_DEFAULT == EGRESS_RECORDER_ENABLED)) )
                    then
                              EGRESS_RECORDER_ENABLED
                    else
                              !EGRESS_RECORDER_ENABLED;
"*****************************************************************************
state_diagram HalfWord~
          state HALF_WORD:
                    if (PDA_WRITE_CONTROL_REG1 &
                     (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 (PDA_WRITE_CONTROL_REG1 &
                     (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 SWIGREn~
          state SWITCH_INGRESS_RECORDER_ENABLED:
                    if (PDA_WRITE_CONTROL_REG1 &
                     (SW_ING_RECORDER_ENABLE_DATA_BIT.pin ==
                                                  !SWITCH_INGRESS_RECORDER_ENABLED) &
                     (!KA_PON_RESET # (SW_ING_REC_ENABLE_PON_DEFAULT !=
                                                  SWITCH_INGRESS_RECORDER_ENABLED)) #
                     (KA_PON_RESET & (SW_ING_REC_ENABLE_PON_DEFAULT ==
                                                  !SWITCH_INGRESS_RECORDER_ENABLED)) )
                    then
                              !SWITCH_INGRESS_RECORDER_ENABLED
                    else
```

```
                        SWITCH_INGRESS_RECORDER_ENABLED;
        state !SWITCH_INGRESS_RECORDER_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (SW_ING_RECORDER_ENABLE_DATA_BIT.pin ==
                                          SWITCH_INGRESS_RECORDER_ENABLED) &
                 (!KA_PON_RESET # (SW_ING_REC_ENABLE_PON_DEFAULT !=
                                        !SWITCH_INGRESS_RECORDER_ENABLED)) #
                 (KA_PON_RESET & (SW_ING_REC_ENABLE_PON_DEFAULT ==
                                          SWITCH_INGRESS_RECORDER_ENABLED)) )
                then
                        SWITCH_INGRESS_RECORDER_ENABLED
                else
                        !SWITCH_INGRESS_RECORDER_ENABLED;
"*****************************************************************************
state_diagram SWEGGEn~
        state SWITCH_EGRESS_GENERATOR_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (SW_EG_GENERATOR_ENABLE_DATA_BIT.pin ==
                                          !SWITCH_EGRESS_GENERATOR_ENABLED) &
                 (!KA_PON_RESET # (SW_EG_GEN_ENABLE_PON_DEFAULT !=
                                          SWITCH_EGRESS_GENERATOR_ENABLED)) #
                 (KA_PON_RESET & (SW_EG_GEN_ENABLE_PON_DEFAULT ==
                                          !SWITCH_EGRESS_GENERATOR_ENABLED)) )
                then
                        !SWITCH_EGRESS_GENERATOR_ENABLED
                else
                        SWITCH_EGRESS_GENERATOR_ENABLED;
        state !SWITCH_EGRESS_GENERATOR_ENABLED:
                if (PDA_WRITE_CONTROL_REG1 &
                 (SW_EG_GENERATOR_ENABLE_DATA_BIT.pin ==
                                          SWITCH_EGRESS_GENERATOR_ENABLED) &
                 (!KA_PON_RESET # (SW_EG_GEN_ENABLE_PON_DEFAULT !=
                                          !SWITCH_EGRESS_GENERATOR_ENABLED)) #
                 (KA_PON_RESET & (SW_EG_GEN_ENABLE_PON_DEFAULT ==
                                          SWITCH_EGRESS_GENERATOR_ENABLED)) )
                then
                         SWITCH_EGRESS_GENERATOR_ENABLED
                else
                        !SWITCH_EGRESS_GENERATOR_ENABLED;
"*****************************************************************************
"* Read Registers.
"* All registers have read capabilty.
"*****************************************************************************
equations
        DataOe = (PDA_READ & PDA_READ_STAT_REG3~) # RESET_CONFIG_DRIVEN;
        Data.oe = DataOe;
"       Data.oe = ^hffff;

        when (PDA_READ_CONFIG_REG # RESET_CONFIG_DRIVEN) then
        Data = ReadConfReg;
"               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 (PDA_READ_CONTROL_REG1) then
       Data = ReadContReg1;
        else when (PDA_READ_STATUS_REG1) then
       Data = ReadStatReg1;
"               Data = [FlashPD4,FlashPD3,FlashPD2,FlashPD1,
"                       DramPdEdo~,DramPD4,DramPD3,DramPD2,DramPD1,
```

```
"                              ExtToolI0,ExtToolI1,ExtToolI2,ExtToolI3,PccVppG~,0,0];
         else when (PDA_READ_STATUS_REG2) then
        Data = ReadStatReg2;
"                Data = [0,0,0,0,0,0,0,0,0,FlashPD7,FlashPD6,FlashPD5,0,0,1,1];
                                                    " revision number 1 - rev - A

    BCSR4Rd~ = !PDA_READ_STATUS_REG3;
end cnt_reg1
```

A.3 U39—BCSR

```
"*****************************************************************************
"* 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.
"*****************************************************************************

module brdct101
title  'M92501 Board Misc. Control Functions.
        Originated for MPC860ADS, Yair Liebman (MSIL) - April 04, 1995
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************
U39 device 'mach220a';

"*****************************************************************************
"* #######
*
"* #        #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #         #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####      ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #          ##       #    #       #####   #  # #  ######  #            #    *
"* #         #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************


"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************

"*****************************************************************************
"* clock generator
"*****************************************************************************

SYSCLK            PIN 15;  " MPC860 clkout

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

```
"* Dram Associated Pins.
"*************************************************************************
A9                PIN 55;
A10               PIN 39;  " MPC860 address lines inputs (IN)

R_W~              PIN 23;

SizeDetect1       PIN 26;
SizeDetect0       PIN 20;  " dram simm size detect lines (IN)

HalfWord~         PIN 51; " dram port width selection from control register:
                          " '1' - 32 bit
                          " '0' - 16 bit
DramBank1Cs~      PIN 45;  " 1'st bank chip-select(IN, L)
DramBank2Cs~      PIN 46;  " 2'nd bank chip-select (IN, L)

DramEn~           PIN 54;  " Dram enable from control reg. (IN, H). Active
                           " high to support power control.

Ras1~             PIN 28 istype 'com';
Ras1DD~           PIN 30 istype 'com';
Ras2~             PIN 29 istype 'com';
Ras2DD~           PIN 31 istype 'com';" dram RAS lines.


"*************************************************************************
"* Flash Associated Pins.
"*************************************************************************
F_PD1             PIN 7;
F_PD2             PIN 65;
F_PD3             PIN 41;
F_PD4             PIN 25;

FlashCs~          PIN 49;  " flash bank chip-select
FlashEn~          PIN 50;  " flash enable from control reg.

FlashCs1~         PIN 12 istype 'com';" Flash bank1 chip-select
FlashCs2~         PIN 22 istype 'com';" Flash bank2 chip-select
FlashCs3~         PIN 57 istype 'com';" Flash bank3 chip-select
FlashCs4~         PIN 24 istype 'com';" Flash bank4 chip-select

FlashOe~          PIN 58 istype 'com';" Flash output enable.

"*************************************************************************
"* Control Register pins
"*************************************************************************
ContRegCs~        PIN 59;  " control register cs from pda
ContRegEn~        PIN 56;  " control register enable from control register.

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

Rst0              PIN 13;                " connected to N.C. of  Reset P.B.
Rst1              PIN 21;                " connected to N.O. of  Reset P.B.

!RegPORIn~        PIN 9;                " Regular Power On Reset In. (H)

HardReset~        PIN 48 istype 'com';    " Actual hard reset output (O.D.)
```

```
SoftReset~        PIN 40 istype 'com';    " Actual soft reset output (O.D.)
ResetConfig~      PIN 67 istype 'com';    " Drives the RSTCONF* signal of the 860.
DriveConfig~      PIN 63 istype 'com';" Drives configuration data to the 860

Abr0              PIN 10;                 " connected to N.C. of  Abort P.B.
Abr1              PIN 11;                 " connected to N.O. of  Abort P.B.

NMIEn             NODE istype 'com';   " enables T.S. NMI pin
NMI~              PIN 44 istype 'com';    " Actual NMI pin (O.D.)

"*************************************************************************
"* Power On Reset Configuration Support
"*************************************************************************
Modck2            PIN 60 istype 'com';" MODCK2 output
Modck1            PIN 66 istype 'com';" MODCK1 output

ModckOe           NODE istype 'com';" enables MODCKs towards PDA during
                                     " Hard Reset.

"*************************************************************************
"* Data Buffers Enables and Reset configuration support
"*************************************************************************
BTA~              PIN 6;                  " Buffered transfer Acknowledge
BTEA~             PIN 47;                 " Buffered Transfer Error Acknowledge.

FlashCfgEn~       PIN 17;              " flash configuration enable from control register.

UpperHalfEn~      PIN 3 istype 'com,invert';    " bits 0:15 data buffer enable
LowerHalfEn~      PIN 5 istype 'com,invert';    " bits 16:31 data buffer enable

"*************************************************************************
"* ATMC Chip Select
"*************************************************************************
ATMCCs~           PIN 2;            " ATMC Chip Select Input

"*************************************************************************
"* CAM Chip Select
"*************************************************************************
CAMCs~            PIN 33;                 " CAM Chip Select Input

"*************************************************************************
"* ATM Traffic Recorders & Generators Chip Select and Enable
"*************************************************************************
GENCs~            PIN 4;                  " GENERAL Chip Select Input
GENRECEn~         PIN 14;                 " Generators & Recorders Enabled Input

"*************************************************************************
"* SONET PHY Chip Select and Enable
"*************************************************************************
PHYCs~            PIN 64;                 " SONET PHY Chip Select Input
EGPHYEn~          PIN 43;                 " Egress PHY Enabled Input
IGPHYEn~          PIN 62;                 " Ingress PHY Enabled Input


"*************************************************************************
"* Transfer Acknowledge Generation
"*************************************************************************
ATMCDTACK~        PIN 32;                     " ATMC DTACK~ Input
CAMDTACK~         PIN 36;                     " CAM DTACK~ Input
```

```
"TA~              PIN 37  istype 'com,buffer';   Transfered Acknowledge output
TA~2              PIN 37 istype 'reg_D,buffer';


"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************

"*****************************************************************************
"* Reset & Interrupt Logic Pins.
"*****************************************************************************
RstDeb1           NODE istype 'com';    " reset push button debouncer
AbrDeb1           NODE istype 'com';    " abort push button debouncer

HardResetEn       NODE istype 'com';    " enables T.S. hard reset pin
SoftResetEn       NODE istype 'com';    " enables T.S. soft reset pin


ConfigHold2,
ConfigHold1,
ConfigHold0       node istype 'reg,buffer';" supplies data hold time for
                                           " hard reset configuration

ConfigHoldEnd     node istype 'com';

"*****************************************************************************
"* Transfer Acknowledge Generation
"*****************************************************************************
TA~1              node istype 'reg_D,buffer';


"*****************************************************************************
"* 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.;

"*****************************************************************************
"* SLOW_32K_LOCK = 1;
"*****************************************************************************
"* Signal groups
"*****************************************************************************
PdaAdd = [A9,A10];
DramCS~ =[DramBank2Cs~,DramBank1Cs~];
RAS = [Ras1~,Ras1DD~,Ras2~,Ras2DD~];
SD = [SizeDetect1,SizeDetect0];
FlashCsOut = [FlashCs4~,FlashCs3~,FlashCs2~,FlashCs1~];
Reset = [HardReset~,SoftReset~];
ResetEn = [HardResetEn,SoftResetEn];
Rst = [Rst1,Rst0];
Abr = [Abr1,Abr0];
Debounce = [RstDeb1,AbrDeb1];
DramCs = [DramBank2Cs~,DramBank1Cs~];
Cs = [ContRegCs~,FlashCs~,DramBank1Cs~,DramBank2Cs~,CAMCs~,ATMCCs~,PHYCs~,GENCs~];
LocDataBufEn = [UpperHalfEn~,LowerHalfEn~];
ModuleEn = [DramEn~,FlashEn~,ContRegEn~,GENRECEn~,EGPHYEn~,IGPHYEn~];
SyncReset = [SyncHardReset~,DSyncHardReset~];
RstCause = [Rst1,Rst0,Abr1,Abr0,RegPORIn~];
Stp = [BTA~];
Modck = [Modck2, Modck1];
ConfigHold = [ConfigHold2, ConfigHold1, ConfigHold0];
F_PD = [F_PD4, F_PD3, F_PD2, F_PD1];
"*****************************************************************************
"* Dram Declarations.
"*****************************************************************************
DRAM_ENABLE_ACTIVE = 0;

DRAM_ENABLED = (DramEn~ == DRAM_ENABLE_ACTIVE);

SIMM36100 = (SD == 0);
SIMM36200 = (SD == 3);
```

```
SIMM36400 = (SD == 2);
SIMM36800 = (SD == 1);

"IS_HALF_WORD = (HalfWord~ == 0);
IS_HALF_WORD = L;

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

FLASH_ENABLED = (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);

"*****************************************************************************
"* 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  = (RegPORIn~ == REGULAR_PON_RESET_ACTIVE);

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;
GEN_REC_ENABLE_ACTIVE = 0;
PHY_ENABLE_ACTIVE = 0;
GPL_ACTIVE = 0;

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

CONTROL_REG_ENABLED = (ContRegEn~ == CONTROL_REG_ENABLE_ACTIVE);

GENERATOR_RECORDER_ENABLED = (GENRECEn~ == GEN_REC_ENABLE_ACTIVE);

PHY_ENABLED = ((EGPHYEn~ == PHY_ENABLE_ACTIVE) #
               (IGPHYEn~ == PHY_ENABLE_ACTIVE));

FLASH_CONFIGURATION_ENABLED = (FlashCfgEn~ == FLASH_CONFIG_ENABLED_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 = !BTA~ & !FlashCs~ & R_W~;        " end of flash read cycle.
END_OF_OTHER_CYCLE = (!BTA~ & FlashCs~ #             " another access or
                          !BTA~ & !FlashCs~ & !R_W~); " flash write

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

"*****************************************************************************
"* Equations, state diagrams.                                                *
"*****************************************************************************
"*                                                                           *
"* #######                                                                   *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #        #    #  #    #   #  #      #       #    #    #  ##   #  #         *
"* #####    #    #  #    #  #    #     #       #    #    #  # #  #   ####     *
"* #        #  # #  #    #  ######     #       #    #    #  #  # #       #    *
"* #        #   #   #    #  #    #     #       #    #    #  #   ##  #    #    *
"* #######   ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                           *
"*****************************************************************************
"* Reset Logic
"*****************************************************************************
equations

Reset.oe = ResetEn;

Reset = 0;" open drain

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

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

```
HardResetEn = RstDeb1 & AbrDeb1 # REGULAR_POWER_ON_RESET;" both buttons are depressed

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


"*************************************************************************
"* Power On reset configuration
"*************************************************************************
equations

Modck.oe = ModckOe;

ModckOe = DRIVE_MODCK_TO_PDA;

Modck1 = H;
Modck2 = H;

"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) then ConfigHold := ConfigHold.fb +1;
else when (SyncHardReset~.fb & ConfigHoldEnd) 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; " drives configuration data on the bus.


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

```
NMI~.oe = NMIEn;

NMI~ = 0;" O.D.

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

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

SyncHardReset~.clk = SYSCLK;
DSyncHardReset~.clk = SYSCLK;

SyncHardReset~ := HardReset~;
DSyncHardReset~ := SyncHardReset~.fb;

SyncTEA~.clk = SYSCLK;
SyncTEA~ := BTEA~;

LocDataBufEn.oe = 3;


!UpperHalfEn~ = (!DramBank1Cs~ & DRAM_ENABLED #
                 !DramBank2Cs~ & (SIMM36200 # SIMM36800) & DRAM_ENABLED #
                 !FlashCs~  & FLASH_ENABLED #
                 !ContRegCs~ & CONTROL_REG_ENABLED #
                 !GENCs~ #
                 !PHYCs~ #
                 !ConfigHoldEnd)  &
                 (STATE_HOLD_OFF_CONSIDERED & (!HOLD_OFF_PERIOD) #STATE_NO_HOLD_OFF);

!LowerHalfEn~ = (!DramBank1Cs~ & DRAM_ENABLED & !IS_HALF_WORD #
                 !DramBank2Cs~ & (SIMM36200 # SIMM36800) &
                 !IS_HALF_WORD & DRAM_ENABLED #
                 !FlashCs~  & FLASH_ENABLED #
                 !GENCs~ #
                 !PHYCs~ #
                 !ConfigHoldEnd & 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~;
TD_FlashOe~ = DD_FlashOe~;
"* QD_FlashOe~ = TD_FlashOe~;
"* PD_FlashOe~ = QD_FlashOe~;

@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;

}

"*************************************************************************
"* 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_ENABLED;
!Ras2~ = !DramBank2Cs~ & DramBank1Cs~ & DRAM_ENABLED & (SIMM36200 # SIMM36800);

!Ras1DD~ = !DramBank1Cs~ & DramBank2Cs~ & DRAM_ENABLED;
!Ras2DD~ = !DramBank2Cs~ & DramBank1Cs~ & DRAM_ENABLED & (SIMM36200 # SIMM36800);

"*************************************************************************
"* Flash Chip Select
"*************************************************************************
equations

FlashCsOut.oe = ^hf;

!FlashCs1~ = FLASH_ENABLED & !FlashCs~ & FLASH_BANK1;
!FlashCs2~ = FLASH_ENABLED & !FlashCs~ & FLASH_BANK2 ;
!FlashCs3~ = FLASH_ENABLED & !FlashCs~ & FLASH_BANK3 ;
!FlashCs4~ = FLASH_ENABLED & !FlashCs~ & FLASH_BANK4 ;

FlashOe~.oe = H;

!FlashOe~ = FLASH_ENABLED & R_W~;

"*************************************************************************
"* TA~ Generation from CAMDTACK~ and ATMCDTACK~
"*************************************************************************
equations

TA~1.clk = SYSCLK;
TA~2.clk = SYSCLK;
```

```
TA~2.oe = (!CAMCs~ # !ATMCCs~);        " Open Drain emulation

TA~1 := (CAMCs~ & ATMCCs~) # (CAMDTACK~ & ATMCDTACK~) # !TA~2.fb;
TA~2 := (CAMCs~ & ATMCCs~) # TA~1.fb # !TA~2.fb;
"TA~ = DFF2.Q;


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

KeepPinsConnected = BTA~ ;


"*******************************************************************************
"* Test vectors                                                                *
"*******************************************************************************
"*                                                                             *
"* #######                                                                     *
"*    #     ######   ####    #####                                            *
"*    #     #       #          #                                              *
"*    #     #####    ####      #                                              *
"*    #     #            #     #                                              *
"*    #     #       #    #     #                                              *
"*    #     ######   ####      #                                              *
"*                                                                             *
"* #     #                                                                     *
"* #     #  ######   ####    #####   ####   #####    ####                     *
"* #     #  #       #    #     #    #    #  #    #  #                         *
"* #     #  #####   #          #    #    #  #    #   ####                     *
"*  #   #   #       #          #    #    #  #####        #                    *
"*   # #    #       #    #     #    #    #  #   #   #    #                    *
"*    #     ######   ####      #     ####   #    #   ####                     *
"*                                                                             *
"*******************************************************************************

@ifdef SIMULATION {

test_vectors 'RAS generator'

([DramEn~,SD,HalfWord~,DramCS~]->[Ras1~,Ras1DD~,Ras2~,Ras2DD~])
" check dran enable
 [   1   ,X ,   X     ,  2    ]->[  1  ,   1   ,  1  ,   1   ];"13
" 32 bit access, 1Meg, single bank, both cs asserted!
 [   0   ,0 ,   1     ,  0    ]->[  1  ,   1   ,  1  ,   1   ];"14
" 32 bit access, 1Meg, single bank, bank1 cs asserted
 [   0   ,0 ,   1     ,  2    ]->[  0  ,   0   ,  1  ,   1   ];"15
" 16 bit access, 1Meg, single bank, bank1 cs asserted
 [   0   ,0 ,   0     ,  2    ]->[  0  ,   1   ,  1  ,   1   ];"16
" 32 bit access, 4Meg, single bank, bank2 cs asserted
 [   0   ,2 ,   1     ,  1    ]->[  1  ,   1   ,  1  ,   1   ];"17
" 16 bit access, 1Meg, single bank, bank2 cs asserted
 [   0   ,2 ,   0     ,  1    ]->[  1  ,   1   ,  1  ,   1   ];"18
" 32 bit access, 1Meg, dual bank, bank1 cs asserted
 [   0   ,3 ,   1     ,  2    ]->[  0  ,   0   ,  1  ,   1   ];"19
" 16 bit access, 1Meg, dual bank, bank1 cs asserted
 [   0   ,3 ,   0     ,  2    ]->[  0  ,   1   ,  1  ,   1   ];"20
" 32 bit access, 4Meg, dual bank, bank2 cs asserted
```

```
 [   0   ,1 ,   1     ,  1    ]->[  1  ,   1   ,  0  ,   0   ];"21
" 16 bit access, 4Meg, dual bank, bank2 cs asserted
 [   0   ,1 ,   0     ,  1    ]->[  1  ,   1   ,  0  ,   1   ];"22

test_vectors 'reset and interrupt logic'

([Rst,Abr,KAPORIn~,RegPORIn~,HardReset~,SoftReset~]->[Debounce,ResetEn,HardReset~,SoftReset~,ResetConfig~,NMIEn,NMI~])
 [ 2 , 2 ,   1    ,   1    ,     1    ,     1    ]->[   0   ,   0   ,   Z    ,   Z    ,     1     ,  0  , Z  ]; "23
" soft reset generation
 [ 1 , 2 ,   1    ,   1    ,     1    ,     X    ]->[   2   ,   1   ,   Z    ,   0    ,     1     ,  0  , Z  ]; "24
" checking debouncer (switch ringing)
 [ 3 , 2 ,   1    ,   1    ,     1    ,     X    ]->[   2   ,   1   ,   Z    ,   0    ,     1     ,  0  , Z  ]; "25
" soft reset generation - ringing stopped
 [ 1 , 2 ,   1    ,   1    ,     1    ,     X    ]->[   2   ,   1   ,   Z    ,   0    ,     1     ,  0  , Z  ]; "26
" No Action
 [ 2 , 2 ,   1    ,   1    ,     1    ,     1    ]->[   0   ,   0   ,   Z    ,   Z    ,     1     ,  0  , Z  ]; "27
" nmi generation
 [ 2 , 1 ,   1    ,   1    ,     1    ,     1    ]->[   1   ,   0   ,   Z    ,   Z    ,     1     ,  1  , 0  ]; "28
" checking debouncer (switch ringing)
 [ 2 , 3 ,   1    ,   1    ,     1    ,     1    ]->[   1   ,   0   ,   Z    ,   Z    ,     1     ,  1  , 0  ]; "29
" nmi generation - ringing stopped
 [ 2 , 1 ,   1    ,   1    ,     1    ,     1    ]->[   1   ,   0   ,   Z    ,   Z    ,     1     ,  1  , 0  ]; "30
" hard reset generation
 [ 1 , 1 ,   1    ,   1    ,     X    ,     1    ]->[   3   ,   2   ,   0    ,   Z    ,     0     ,  0  , Z  ]; "31
" No Action
 [ 2 , 2 ,   1    ,   1    ,     1    ,     1    ]->[   0   ,   0   ,   Z    ,   Z    ,     1     ,  0  , Z  ]; "32
" keep alive power on reset
 [ 2 , 2 ,   0    ,   1    ,     X    ,     1    ]->[   0   ,   2   ,   0    ,   Z    ,     0     ,  0  , Z  ]; "33
" No Action
 [ 2 , 2 ,   1    ,   1    ,     1    ,     1    ]->[   0   ,   0   ,   Z    ,   Z    ,     1     ,  0  , Z  ]; "34
" regular power on reset
 [ 2 , 2 ,   1    ,   0    ,     X    ,     1    ]->[   0   ,   2   ,   0    ,   Z    ,     0     ,  0  , Z  ]; "35
" No Action
 [ 2 , 2 ,   1    ,   1    ,     1    ,     1    ]->[   0   ,   0   ,   Z    ,   Z    ,     1     ,  0  , Z  ]; "36


test_vectors 'data buffers enables'

([SYSCLK,HardReset~,TS~,Cs ,Stp,TEA~,ModuleEn,FlashCfgEn~,SD,HalfWord~,RstCause]->[Syn-
cReset,LocDataBufEn])
" hard reset asserted
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    X    ,     1      ]; "37
" hard reset asserted
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    0    ,     1      ]; "38
" hard reset negated - no change should occure
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    2    ,     1      ]; "39
" hard reset negated - now upper enable should negate
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "40
" hard reset asserted - flash configuration enabled, lower half should open also
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h7   ,     0     ,0 ,    1    ,  ^h2b  ]->[    1    ,     3      ]; "41
" hard reset asserted
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h7   ,     0     ,0 ,    1    ,  ^h2b  ]->[    0    ,     0      ]; "42
" hard reset negated - no change should occure
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h7   ,     0     ,0 ,    1    ,  ^h2b  ]->[    2    ,     0      ]; "43
" hard reset negated - now upper enable should negate
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h7   ,     0     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "44
" flash access, module disabled
 [  C   ,    1     , 0 ,^hb, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "45
" flash access, module enabled
 [  C   ,    1     , 0 ,^hb, 1 , 1  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "46
" TS negated
 [  C   ,    1     , 1 ,^hb, 1 , 1  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "47
" TA asserted
 [  C   ,    1     , 1 ,^hb, 1 , 1  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "48
" dram access, module disabled
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "49
" dram access, 1'st bank, module enabled, 32 bit, 1 bank simm
 [  C   ,    1     , X ,^hf, 1 , 1  ,  ^hf   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "50
" TS negated
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^hf   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "51
" TA asserted
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^hf   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "52
" dram access, 2'nd bank, module disabled
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "53
" negate CS
 [  C   ,    1     , X ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "54
" dram access, 2'nd bank, module enabled, 32 bit, 1 bank simm - buff en should not assert
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "55
" negate CS
```

```
 [  C   ,    1     , X ,^hf, 1 , 1  ,  ^hf   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "56
" dram access, 2'nd bank, module enabled, 32 bit, 2 bank simm - buff en should assert
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,3 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "57
" TS negated
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,1 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "58
" TA asserted
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,3 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "59
" dram access, 1'st bank, module enabled, 16 bit, 1 bank simm
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^hf   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "60
" TS negated
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^hf   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "61
" TA asserted
 [  C   ,    1     , X ,^hd, 1 , 1  ,  ^hf   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "62
" dram access, 2'nd bank, module disabled
 [  C   ,    1     , X ,^he, 3 , 1  ,  ^h7   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "63
" negate CS
 [  C   ,    1     , X ,^hf, 1 , 1  ,  ^h7   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "64
" dram access, 2'nd bank, module enabled, 32 bit, 1 bank simm - buff en should not assert
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "65
" negate CS
 [  C   ,    1     , X ,^hf, 1 , 1  ,  ^hf   ,     1     ,0 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "66
 " dram access, 2'nd bank, module enabled, 32 bit, 2 bank simm - buff en should not assert
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "67
" TS negated
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,1 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "68
" TA asserted
 [  C   ,    1     , X ,^he, 1 , 1  ,  ^hf   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "69
" Control register access, module disabled
 [  C   ,    1     , 0 ,^h7, 1 , 1  ,  ^h7   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "70
" Control register access, module enabled
 [  C   ,    1     , 0 ,^h7, 1 , 1  ,  ^h6   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "71
" TS negated
 [  C   ,    1     , 1 ,^h7, 1 , 1  ,  ^h6   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "72
" TA asserted
 [  C   ,    1     , 1 ,^h7, 1 , 1  ,  ^h6   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "73
" pcmcia access, module disabled
 [  C   ,    1     , 0 ,^hf, 1 , 1  ,  ^h7   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "74
" pcmcia access, module enabled, even byte
 [  C   ,    1     , 0 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "75
" TS negated
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "76
" TA asserted
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "77
" pcmcia access, module enabled, odd byte
 [  C   ,    1     , 0 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "78
" TS negated
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "79
" TA asserted
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "80
" pcmcia access, module enabled,  even & odd byte, hard reset ending
 [  C   ,    1     , 0 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "81
" TS negated
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     1      ]; "82
" hard reset asserted
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    1    ,     1      ]; "83
" hard reset asserted
 [  C   ,    0     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    0    ,     1      ]; "84
" hard reset negated
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    2    ,     1      ]; "85
" hard reset negated
 [  C   ,    1     , 1 ,^hf, 1 , 1  ,  ^h5   ,     1     ,3 ,    0    ,  ^h2b  ]->[    3    ,     3      ]; "86
" flash access, module enabled - TEA termination
 [  C   ,    1     , 0 ,^hb, 1 , 1  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "87
" TS negated
 [  C   ,    1     , 1 ,^hb, 1 , 1  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     0      ]; "88
" TEA asserted
 [  C   ,    1     , 1 ,^hb, 1 , 0  ,  ^h3   ,     1     ,0 ,    1    ,  ^h2b  ]->[    3    ,     3      ]; "89

}

end brdctl01
```

A.4 U19—Logic Equations for the CAM

```
"*****************************************************************************
"* In this file :
"* - Interface logic between CAM and ATMC.
"*****************************************************************************

module cam_atmc
title  'M92501 CAM/ATMC Interface Logic.
        Originated for M92501, Amitay Beler - (MSIL) - February 11, 1998'

"*****************************************************************************
"* Device declaration.                                                       *
"*****************************************************************************
U19 device 'P16V8';

"*****************************************************************************
"* #######                                                                   *
"* #       #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #        #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####     ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #         ##       #    #       #####   #  # #  ######  #            #    *
"* #        #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************


"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************

ACLKI~            PIN  1;
ACLK              PIN  2;

EACEn~            PIN  3;
EMWr~             PIN  4;

"*****************************************************************************
"* CAM/ATMC Interface Control Pins.
"*****************************************************************************
G~                PIN  17 istype 'com,buffer';   " CAM output enable on read
GQ~               PIN  12 istype 'com,buffer';   " Latch Output Enable for CAM write
MQOE~             PIN  13 istype 'com,buffer';   " Buffers Output Enable for CAM read
LH_SM~            PIN  18 istype 'com,buffer';   " Load high/Start Match
LHSM~             PIN  16 istype 'reg';       " Internal LH_SM~

ACLKO~            PIN  19 istype 'com,buffer';   " ACLK Inverted

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
```

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

"*******************************************************************************
"*  #####                                                                      *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####           *
"* #         #    #  ##   #  #          #     #  #   ##   #     #             *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #             *
"* #         #    #  #  # #       #     #    ######  #  # #     #             *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #             *
"*  #####    ####   #    #   ####      #    #    #  #    #     #             *
"*                                                                            *
"* ######                                                                     *
"* #     #  ######   ####   #          ##    #####                           *
"* #     #  #       #    #  #         #  #   #    #                          *
"* #     #  #####   #       #        #    #  #    #  #####                    *
"* #     #  #       #       #        ######  #####                           *
"* #     #  #       #    #  #        #    #  #   #                           *
"* ######   ######   ####   ######  #    #  #    #                          *
"*                                                                            *
"*    ##     #####     #     ####   #    #                                    *
"*   #  #      #       #     #    #  ##   #                                    *
"*  #    #     #       #     #    #  # #  #                                    *
"*  ######     #       #     #    #  #  # #                                    *
"*  #    #     #       #     #    #  #   ##                                   *
"*  #    #     #       #     ####   #    #                                    *
"*******************************************************************************

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

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

"*******************************************************************************
"* Power On Reset definitions
"*******************************************************************************

"*******************************************************************************
"* CAM Access definitions
"*******************************************************************************

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

equations

ACLKO~.oe = 1;
ACLKO~ = !ACLK;
```

```
LH_SM~.oe = 1;
LHSM~.clk = ACLKI~;
LHSM~ := (EACEn~ # EMWr~); " CAM write only when EACEn~ & EMWr~ are asserted
LH_SM~ = LHSM~.fb; " External LH_SM~ equals the internal due to PAL limitations
                   " on OE for registers
GQ~.oe = 1;
GQ~ = !LH_SM~.pin; " Latch output enable on CAM write

G~.oe = 1;
G~ = !(EACEn~ # EMWr~) # GQ~.pin; " CAM output enable on read

MQOE~.oe = 1;
MQOE~ = !EMWr~ # EACEn~; " Latch ATMC data bus on CAM write

"*****************************************************************************
"* TEST VECTORS
"*****************************************************************************
end cam_atmc
```

A.5 U32—Logic Equations for the Ingress Generator

```
"*****************************************************************************
"* In this file :
"* - Programmable logic for the Ingress Generator.
"*****************************************************************************

module ing_gen
title  'M92501 Ingress Generator Logic.
        Originated for M92501, Amitay Beler - (MSIL) - February 12, 1998'

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

U32 device 'P22V10C';

"*****************************************************************************
"* #######                                                                   *
"* #         #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #          #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####       ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #           ##       #    #       #####   #  # #  ######  #            #    *
"* #          #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************


"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************
```

```
ACLKI~              PIN  2;
ACLK                PIN  3;

RREn~               PIN  4;          " Ingress Generator Read Enable
RXENB~              PIN  5;          " Receive Enable

BWE0~               PIN 13;          " Buffered write Enable 0
GENCs~              PIN 16;          " Generators/Recorders Chip Select

BA8                 PIN  6;          " Buffered address Line 8
BA9                 PIN  7;          " Buffered address Line 9
BA10                PIN  9;          " Buffered address Line 10
BA11                PIN 10;          " Buffered address Line 11
BA12                PIN 11;          " Buffered address Line 12
BA13                PIN 12;          " Buffered address Line 13

"*****************************************************************************
"* Ingress Generator Control Pins.
"*****************************************************************************

RLATCHEn            PIN  18  istype 'com,buffer'; " Receive Latch Enable
RFRd~               PIN  19 istype 'com,buffer'; " Receive FIFO Read
RFWr~               PIN  27 istype 'com,buffer'; " Receive FIFO Write

ACLKO~              PIN  17 istype 'com,buffer'; " ACLK Inverted

"*****************************************************************************
"*   ###                                                                     *
"*    #      #     #   #####  ######  #####   #     #     ##     #         ####      *
"*    #      ##    #     #    #       #    #  ##    #    #  #    #        #          *
"*    #      # #   #     #    #####   #    #  # #   #   #    #   #         ####      *
"*    #      #  # #     #    #       #####   #  # #  ######  #             #     *
"*    #      #   ##     #    #       #   #   #   ##  #    #  #        #    #     *
"*   ###     #    #     #    ######  #    #  #    #  #    #  ######   ####      *
"*****************************************************************************

DFF1                NODE istype 'reg,buffer'; " D-FlipFlop
DFF2                NODE istype 'reg,buffer'; " D-FlipFlop

"*****************************************************************************
"*  #####                                                                    *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####         *
"* #         #    #  ##   #  #          #     #  #   ##   #     #           *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #           *
"* #         #    #  #  # #       #     #    ######  #  # #     #           *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #           *
"*  #####    ####   #    #   ####      #    #    #  #    #     #           *
"*                                                                          *
"* ######                                                                  *
"* #     #  ######   ####   #         ##    #####                          *
"* #     #  #       #    #  #        #  #   #    #                         *
"* #     #  #####   #       #       #    #  #    #  #####                  *
"* #     #  #       #       #       ######  #####                         *
"* #     #  #       #    #  #       #    #  #   #                          *
"* ######   ######   ####   ######  #    #  #    #                         *
"*                                                                          *
"*    ##     #####     #     ####   #    #                                 *
"*   #  #      #       #    #    #  ##   #                                 *
"*  #    #     #       #    #    #  # #  #                                 *
```

```
"*  ######     #       #    #    #  #  # #                                    *
"*  #     #    #       #    #    #  #   ##                                    *
"*  #     #    #       #    #     ####   #    #                               *
"*****************************************************************************

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

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

"*****************************************************************************
"* Power On Reset definitions
"*****************************************************************************

"*****************************************************************************
"* Access definitions
"*****************************************************************************

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

equations

ACLKO~.oe = 1;
ACLKO~ = !ACLK;

RFWr~.oe = 1;
RFWr~ = (BA8 # BA9 # !BA10 # BA11 # BA12 # GENCs~ # BWE0~);

DFF1.clk = ACLKI~;
DFF1.d := RXENB~;

DFF2.clk = ACLKI~;
DFF2.d := RREn~;

RFRd~.oe = 1;
RFRd~ = RLATCHEn.fb;
"RFRd~ = DFF1.fb # DFF2.fb # !ACLK;

RLATCHEn.oe = 1;
RLATCHEn = !DFF1.fb & !DFF2.fb & ACLK;

"*****************************************************************************
"* TEST VECTORS
"*****************************************************************************

end ing_gen
```

A.6 U34 Device ‘P22V10C’;

```
"* U34 - logic equations for the Egress Generator
"*****************************************************************************
"* In this file :
"* - Programmable logic for the Egress Generator.
"*****************************************************************************

module eg_gen
title  'M92501 Egress Generator Logic.
        Originated for M92501, Amitay Beler - (MSIL) - February 12, 1998'

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


"*****************************************************************************
"* #######                                                                   *
"* #         #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #          #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####       ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #           ##       #    #       #####   #  # #  ######  #            #    *
"* #          #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************

SYSCLKI~          PIN  2;          " SYSCLK Inverted In
SYSCLK            PIN  3;          " SYSCLK In

EREn~             PIN  4;          " Egress Generator Read Enable
ERFR              PIN  5;          "
STXCLAV           PIN  6;          " Switch transmit Cell Available

BWE0~             PIN 16;          " Buffered write Enable 0
GENCs~            PIN 27;          " Generators/Recorders Chip Select

BA8               PIN  7;          " Buffered address Line 8
BA9               PIN  9;          " Buffered address Line 9
BA10              PIN 10;          " Buffered address Line 10
BA11              PIN 11;          " Buffered address Line 11
BA12              PIN 12;          " Buffered address Line 12
BA13              PIN 13;          " Buffered address Line 13

"*****************************************************************************
"* Ingress Generator Control Pins.
"*****************************************************************************

ELATCHEn          PIN  18 istype 'com,buffer'; " Egress Latch Enable
EFRd~             PIN  19 istype 'com,buffer'; " Egress FIFO Read
EFWr~             PIN  26 istype 'com,buffer'; " Egress FIFO Write
```

```
SYSCLKO~          PIN  17 istype 'com,buffer'; " SYSCLK Inverted out

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ###### #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    # #       #    #    *
"*   ###    #    #     #    ######  #    #  #    # #    # ######   ####     *
"*****************************************************************************

DFF1              NODE     istype 'reg,buffer'; " D-FlipFlop
DFF2              NODE     istype 'reg,buffer'; " D-FlipFlop
DFF3              NODE     istype 'reg,buffer'; " D-FlipFlop

"*****************************************************************************
"*  #####                                                                    *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####          *
"* #         #    #  ##   #  #          #     #  #   ##   #     #            *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #            *
"* #         #    #  #  # #       #     #    ######  #  # #     #            *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #            *
"*  #####    ####   #    #   ####      #    #    #  #    #     #            *
"*                                                                           *
"* ######                                                                   *
"* #     #  ######   ####   #         ##    #####                           *
"* #     #  #       #    #  #        #  #   #    #                          *
"* #     #  #####   #       #       #    #  #    #  #####                   *
"* #     #  #       #       #       ######  #####                           *
"* #     #  #       #    #  #       #    #  #   #                           *
"* ######   ######   ####   ######  #    #  #    #                          *
"*                                                                           *
"*    ##     #####     #     ####   #    #                                  *
"*   #  #      #        #    #    #  ##   #                                  *
"*  #    #     #        #    #    #  # #  #                                  *
"*  ######     #        #    #    #  #  # #                                  *
"*  #    #     #        #    #    #  #   ##                                  *
"*  #    #     #        #     ####   #    #                                  *
"*****************************************************************************

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

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

"*****************************************************************************
"* Power On Reset definitions
"*****************************************************************************

"*****************************************************************************
"* Access definitions
"*****************************************************************************

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

```
"*                                                                              *
"* #######                                                                      *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #        #    #  #    #   #  #      #       #    #    #  ##   #  #         *
"* #####    #    #  #    #  #    #     #       #    #    #  # #  #   ####     *
"* #        #  # #  #    #  ######     #       #    #    #  #  # #       #    *
"* #        #   #   #    #  #    #     #       #    #    #  #   ##  #    #    *
"* #######   ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                              *
"********************************************************************************

equations

SYSCLKO~.oe = 1;
SYSCLKO~ = !SYSCLK;

EFWr~.oe = 1;
EFWr~ = BA8 # !BA9 # !BA10 # !BA11 # !BA12 # GENCs~ # BWE0~;

DFF1.clk = SYSCLKI~;
DFF1.d := STXCLAV;

DFF2.clk = SYSCLKI~;
DFF2.d := EREn~;

DFF3.clk = SYSCLKI~;
DFF3.d := ERFR;

ELATCHEn.oe = 1;
ELATCHEn = ((!DFF2.fb & DFF1.fb) # DFF3.fb) & SYSCLK;

EFRd~.oe = 1;
EFRd~ = ELATCHEn.fb;
" EFRd~ = !(((!DFF2.fb & DFF1.fb) # DFF3.fb) & SYSCLK);

"********************************************************************************
"* TEST VECTORS
"********************************************************************************

end eg_gen
```

A.7 U31 Device ‘MACH220a’;

```
"* U31 - Ingress Generator equations
"********************************************************************************
"* In this file :
"* - Programmable logic for the Ingress Generator.
"********************************************************************************

module rmach
title   'RPH Interface logic for ATMC_EVB Board
        Originated for ATMC EVB, Aviel Livay (MSIL) - April 04, 1996
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

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

```
"*****************************************************************************
"* #######
*
"* #         #     #   #####  ######  #####   #     #    ##    #         ####     *
"* #          #  #       #    #       #    #  ##    #   #  #   #        #         *
"* #####       ##        #    #####   #    #  # #   #  #    #  #         ####     *
"* #           ##        #    #       #####   #  # #  ######  #             #     *
"* #          #  #       #    #       #   #   #   ##  #    #  #        #    #     *
"* #######  #     #      #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                                          *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************

A_ACLK                      pin 15;
DFLT_CS_                    pin 16;
RESET_                      pin 17;
RW_                         pin 26;
ADD30                       pin 36;
SIZ0                        pin 37;
BWE0~                       pin 49;
RdOE~                       pin 50;

"*****************************************************************************
"* Address bits
"*****************************************************************************

BA8                         pin 2;
BA9                         pin 3;
BA10                        pin 4;
BA11                        pin 5;
BA12                        pin 6;
BA13                        pin 7;

"*****************************************************************************
"Data bits
"*****************************************************************************

BD0                         pin 9;
BD1                         pin 10;
BD2                         pin 11;
BD3                         pin 12;
BD4                         pin 13;
BD5                         pin 14;
BD6                         pin 21;
BD7                         pin 22;

"*****************************************************************************
" Ingress Generator signals
"*****************************************************************************

A_RXENB_                    pin 65;
IG_GEN_En~                  pin 20;
RX_ESC                      pin 66;
```

```
A_RXDATA0                   pin 67;
B_RXEMPTY_                  pin 64;
A_MSEL_                     pin 44;
DI_MSEL_                    pin 46;
RREN_                       pin 55 istype 'reg, buffer';
RFRT_                       pin 56 istype 'reg, buffer';
RFRS_                       pin 57 istype 'reg, buffer';
RFOE_                       pin 59 istype 'reg, buffer';
RSTP                        pin 23 istype 'reg, buffer';
RENG                        pin 24 istype 'reg, buffer';
RIDL                        pin 25 istype 'reg, buffer';
DREQ1_                      pin 33 istype 'com, buffer';
DREQ2_                      pin 32 istype 'com, buffer';
A_RXEMPTY_                  pin 58 istype 'com, buffer';
S0                          pin 41 istype 'reg, buffer';
S1                          pin 43 istype 'reg, buffer';
MWSH_                       pin 38 istype 'com, buffer';
MWSL_                       pin 39 istype 'com, buffer';
DO_MSEL_                    pin 45 istype 'com, buffer';
DDO_MSEL_                   pin 47 istype 'com, buffer';

"*****************************************************************************
" ATMC Request Signals
"*****************************************************************************

A_MCIREQ_                   pin 31;
A_MCOREQ_                   pin 30;
A_EMMREQ_                   pin 29;

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####    *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #        *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####    *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #   *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #   *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####    *
"*****************************************************************************

 READ_CYCLE                         node istype 'reg';
 FIFO_EMPTY_                        node istype 'reg';
 SRRST, SRPIM, SRFOE_               node istype 'reg';
 RRST, RPIM                         node istype 'reg';
 R_WAS_ENB                          node istype 'reg';
 DREQ10, DREQ11, DREQ20, DREQ21     node istype 'reg';

"*****************************************************************************
"*  #####                                                                    *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####          *
"* #         #    #  ##   #  #          #     #  #   ##   #     #            *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #            *
"* #         #    #  #  # #       #     #    ######  #  # #     #            *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #            *
"*  #####    ####   #    #   ####      #    #    #  #    #     #            *
"*                                                                           *
"* ######                                                                   *
"* #     #  ######   ####   #         ##    #####                           *
"* #     #  #       #    #  #        #  #   #    #                          *
"* #     #  #####   #       #       #    #  #    #  #####                   *
```

```
"* #     #  #       #       #       ######  #####                          *
"* #     #  #       #    #  #       #    #  #   #                          *
"* ######   ######   ####   ######  #    #  #    #                         *
"*                                                                          *
"*    ##     #####     #     ####   #    #                                  *
"*   #  #      #       #    #    #  ##   #                                  *
"*  #    #     #       #    #    #  # #  #                                  *
"*  ######     #       #    #    #  #  # #                                  *
"*  #    #     #       #    #    #  #   ##                                  *
"*  #    #     #       #     ####   #    #                                  *
"****************************************************************************

H, L, X, Z = 1, 0, .X., .Z.;
C, D, U, P = .C., .D., .U., .P.;
SIMULATION = 1;

D = [BD0,BD1,BD2,BD3,BD4,BD5,BD6,BD7];
ADDR = [BA8,BA9,BA10,BA11,BA12,BA13];

RPH_FIFO_ADDR = [0,0,1,0,0,X];
RIL_CONTROL_REGISTER_ADDR = [0,0,1,0,1,0];
GENERAL_STATUS_REGISTER_ADDR = [0,0,1,1,0,0];
REQ_CONTROL_REGISTER_ADDR = [0,0,1,1,1,0];

MCIRQ = [0,0];
MCORQ = [0,1];
EMMRQ = [1,0];
NORQ  = [1,1];

DREQ1CTL = [DREQ11, DREQ10];
DREQ2CTL = [DREQ21, DREQ20];

RIL_CONTROL_REGISTER = [SRPIM, SRFOE_, SRRST];

"****************************************************************************
" State Machine
"****************************************************************************

         RSTATE      = [RFRS_,RREN_,RFRT_,S1 ,S0 ];
         RESET_STATE = [ 0 , 1 , 1 , 0 , 0 ];
         IDLE_STATE  = [ 1 , 1 , 1 , 0 , 0 ];
         WAIT_STATE  = [ 1 , 0 , 1 , 0 , 0 ];
         TX_STATE    = [ 1 , 0 , 1 , 0 , 1 ];
         STOP_STATE  = [ 1 , 1 , 1 , 0 , 1 ];
         RET1_STATE  = [ 1 , 1 , 1 , 1 , 1 ];
         RET2_STATE  = [ 1 , 1 , 0 , 0 , 1 ];

State_Diagram RSTATE

State RESET_STATE:
         IF (RRST) THEN RESET_STATE
         ELSE IDLE_STATE;

State IDLE_STATE:
         IF (RRST) THEN RESET_STATE
         ELSE IF (RPIM) THEN WAIT_STATE
         ELSE IDLE_STATE;

State WAIT_STATE:
```

```
         IF (RRST) THEN RESET_STATE
         ELSE TX_STATE;

State TX_STATE:
         IF (RRST) THEN RESET_STATE
         ELSE IF (R_WAS_ENB & RX_ESC & A_RXDATA0) THEN RET1_STATE
         ELSE IF (R_WAS_ENB & RX_ESC & !A_RXDATA0) THEN STOP_STATE
         ELSE TX_STATE;

State RET1_STATE:
         IF (RRST) THEN RESET_STATE
         ELSE RET2_STATE;

State RET2_STATE:
         IF (RRST) THEN RESET_STATE
 ELSE IF (RPIM) THEN TX_STATE
 ELSE RESET_STATE;

State STOP_STATE:
         IF (RRST) THEN RESET_STATE
         ELSE IF (RPIM) THEN STOP_STATE
         ELSE RESET_STATE

State 0: GOTO RESET_STATE;
State 1: GOTO RESET_STATE;
State 2: GOTO RESET_STATE;
State 3: GOTO RESET_STATE;
State 4: GOTO RESET_STATE;
State 5: GOTO RESET_STATE;
State 6: GOTO RESET_STATE;
State 7: GOTO RESET_STATE;
State 8: GOTO RESET_STATE;
State 9: GOTO RESET_STATE;
State 10:GOTO RESET_STATE;
State 11:GOTO RESET_STATE;
"State 12:GOTO RESET_STATE;
State 13:GOTO RESET_STATE;
State 14:GOTO RESET_STATE;
State 15:GOTO RESET_STATE;
State 16:GOTO RESET_STATE;
State 17:GOTO RESET_STATE;
State 18:GOTO RESET_STATE;
State 19:GOTO RESET_STATE;
"State 20:GOTO RESET_STATE;
"State 21:GOTO RESET_STATE;
State 22:GOTO RESET_STATE;
State 23:GOTO RESET_STATE;
State 24:GOTO RESET_STATE;
"State 25:GOTO RESET_STATE;
State 26:GOTO RESET_STATE;
State 27:GOTO RESET_STATE;
"State 28:GOTO RESET_STATE;
"State 29:GOTO RESET_STATE;
State 30:GOTO RESET_STATE;
"State 31:GOTO RESET_STATE;


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

```
"*****************************************************************************
"* Access definitions
"*****************************************************************************

"*****************************************************************************
"* Equations,.                                                               *
"*****************************************************************************
"*                                                                           *
"* #######                                                                   *
"* #          ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #         #    #  #    #   #  #      #       #    #    #  ##   #  #         *
"* #####     #    #  #    #  #    #     #       #    #    #  # #  #   ####     *
"* #         #  # #  #    #  ######     #       #    #    #  #  # #       #    *
"* #         #   #   #    #  #    #     #       #    #    #  #   ##  #    #    *
"* #######    ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                           *
"*****************************************************************************

equations

RREN_.oe          = H;
RFRT_.oe          = H;
RFOE_.oe          = H;
RFRS_.oe          = H;
S0.oe     = H;
S1.oe     = H;
DREQ1_.oe         = H;
DREQ2_.oe         = H;
DO_MSEL_.oe       = H;
DDO_MSEL_.oe      = H;
A_RXEMPTY_.oe     = !IG_GEN_En~;  " generator initiates activity only when selected
RSTP.oe           = !DFLT_CS_ & (ADDR == GENERAL_STATUS_REGISTER_ADDR) & !RdOE~ & RESET_;
RENG.oe           = !DFLT_CS_ & (ADDR == GENERAL_STATUS_REGISTER_ADDR) & !RdOE~ & RESET_;
RIDL.oe           = !DFLT_CS_ & (ADDR == GENERAL_STATUS_REGISTER_ADDR) & !RdOE~ & RESET_;
MWSH_.oe          = H;
MWSL_.oe          = H;
ADD30.oe          = L;                  " I/O pin is configured as input only
SIZ0.oe           = L;                  " I/O pin is configured as input only

RREN_.clk         = A_ACLK;
RFRT_.clk         = A_ACLK;
RFOE_.clk         = A_ACLK;
RFRS_.clk         = A_ACLK;
RSTP.clk          = A_ACLK;
RIDL.clk          = A_ACLK;
RENG.clk          = A_ACLK;
RPIM.clk          = A_ACLK;
R_WAS_ENB.clk     = A_ACLK;
S0.clk            = A_ACLK;
S1.clk            = A_ACLK;
RRST.clk          = A_ACLK;
READ_CYCLE.clk    = A_ACLK;
FIFO_EMPTY_.clk   = A_ACLK;

SRRST.clk         = A_ACLK;
SRPIM.clk         = A_ACLK;
SRFOE_.clk        = A_ACLK;
DREQ10.clk        = A_ACLK;
```

```
DREQ11.clk        = A_ACLK;
DREQ20.clk        = A_ACLK;
DREQ21.clk        = A_ACLK;

"*****************************************************************************
"* Control Register Power-On Reset defaults
"*****************************************************************************

SRRST.ap          = !RESET_;                " SRRST = 1
SRPIM.ar          = !RESET_;                " SRPIM = 0
SRFOE_.ap         = !RESET_;                " SRFOE_ = 1
DREQ10.ar         = !RESET_;
DREQ11.ap         = !RESET_;                " DREQ1 = EMMREQ~
DREQ20.ap         = !RESET_;
DREQ21.ar         = !RESET_;                " DREQ2 = MCOREQ~

"*****************************************************************************
"* Control Register initialization
"*****************************************************************************

when (!DFLT_CS_ & (ADDR == RIL_CONTROL_REGISTER_ADDR) & !RW_ & !BWE0~)
         then
                  RIL_CONTROL_REGISTER := [BD0,BD1,BD2];
         else
                  RIL_CONTROL_REGISTER := RIL_CONTROL_REGISTER;

RRST := SRRST.fb;
RPIM := SRPIM.fb;
RFOE_ := SRFOE_.fb # IG_GEN_En~;

"*****************************************************************************
"* Status register
"*****************************************************************************

RSTP := (RSTATE == STOP_STATE);
RIDL := (RSTATE == IDLE_STATE);

when (!RFRS_)
         then
                  RENG := 0;
         else when (RPIM & A_RXENB_)
                  then
                           RENG := 1;
                  else
                           RENG := RENG.FB;

READ_CYCLE := !RREN_ & !A_RXENB_;

when (!RPIM)
         then
                  FIFO_EMPTY_ := 1;
         else when (READ_CYCLE)
                  then
                           FIFO_EMPTY_ := B_RXEMPTY_;
                  else
                           FIFO_EMPTY_ := FIFO_EMPTY_;

A_RXEMPTY_ = READ_CYCLE & FIFO_EMPTY_;
```

```
R_WAS_ENB := !A_RXENB_;

"*****************************************************************************
"* Request management
"*****************************************************************************

when ((ADDR == REQ_CONTROL_REGISTER_ADDR) & !BWE0~ & !DFLT_CS_)
         then
                  DREQ1CTL :=[BD0,BD1];
         else
                  DREQ1CTL := DREQ1CTL;

when ((ADDR == REQ_CONTROL_REGISTER_ADDR) & !BWE0~ & !DFLT_CS_)
         then
                  DREQ2CTL :=[BD2,BD3];
         else
                  DREQ2CTL := DREQ2CTL;

when (DREQ1CTL == EMMRQ)
         then
                  DREQ1_ = A_EMMREQ_;
         else when (DREQ1CTL == MCIRQ)
                  then
                           DREQ1_ = A_MCIREQ_;
                  else when (DREQ1CTL == MCORQ)
                           then
                                    DREQ1_ = A_MCOREQ_;
                           else
                                    DREQ1_ = H;

when (DREQ2CTL == EMMRQ)
         then
                  DREQ2_ = A_EMMREQ_;
         else when (DREQ2CTL == MCIRQ)
                  then
                           DREQ2_ = A_MCIREQ_;
                  else when (DREQ2CTL == MCORQ)
                           then
                                    DREQ2_ = A_MCOREQ_;
                           else
                                    DREQ2_ = H;

"*****************************************************************************
"* MSWH, MWSL generation
"*****************************************************************************

MWSH_ = SIZ0 & ADD30;
MWSL_ = SIZ0 & !ADD30;


DO_MSEL_ = A_MSEL_;
DDO_MSEL_ = DI_MSEL_;

end rmach
```

A.8 U30—Egress Generator and Ingress Recorder Equations

```
”*****************************************************************************
“* In this file :
“* - Programmable logic for the Egress Generator and Ingress Recorder.
“*****************************************************************************

module xmach
title   ‘ISW+ESW Interface logic for ATMC_EVB Board
        Originated for ATMC EVB, Aviel Livay (MSIL) - April 04, 1996
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

“*****************************************************************************
“* Device declaration.                                                       *
“*****************************************************************************

u30   device ‘mach220a’;

“*****************************************************************************
“* #######                                                                   *
“* #       #    #   #####  ######  #####   #    #    ##    #        ####     *
“* #        #  #      #    #       #    #  ##   #   #  #   #       #         *
“* #####     ##       #    #####   #    #  # #  #  #    #  #        ####     *
“* #         ##       #    #       #####   #  # #  ######  #            #    *
“* #        #  #      #    #       #   #   #   ##  #    #  #       #    #    *
“* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
“*****************************************************************************

“*****************************************************************************
“* Pins declaration.                                                         *
“*****************************************************************************
“* System i/f pins
“*****************************************************************************

RESET_                  pin 17;
SXCLK                   pin 15;
DFLT_CS_                pin 16;
RW_                     pin 20;
BA13                    pin 2;
BA12                    pin 3;
BA11                    pin 4;
BA10                    pin 5;
BA9                     pin 6;
BA8                     pin 7;
BD7                     pin 9;
BD6                     pin 10;
BD5                     pin 11;
BD4                     pin 12;
BD3                     pin 13;
BD2                     pin 14;
BD1                     pin 21;
BD0                     pin 22;
A_SRXCLAV               pin 25;
A_STXCLAV               pin 26;
EX_ESC                  pin 28;
A_STXDATA0              pin 29;
ECCLV                   pin 30;
```

```
B_STXENB_                   pin 56;
SW_EG_GEN_En~               pin 49;
SW_IG_REC_En~               pin 50;
RdOE~                       pin 51;
BWE0~                       pin 54;

"*****************************************************************************
"* Ingress Recorder OUTPUT PINS
"*****************************************************************************

 A_SRXENB_                  pin 31 istype 'reg, buffer';
 XEIDL                      pin 32 istype 'com, buffer';
 ISIM                       pin 33 istype 'reg, buffer';
 ISIM_                      pin 36 istype 'com, buffer';
 I_CNT_EN                   pin 37 istype 'com, buffer';
 IROE_                      pin 38 istype 'com, buffer';

"*****************************************************************************
"* Egress Generator OUTPUT PINS
"*****************************************************************************

 EREN_                      pin 39 istype 'reg, buffer';
 EFRT_                      pin 40 istype 'reg, buffer';
 EFRS_                      pin 41 istype 'reg, buffer';
 ESTP                       pin 23 istype 'reg, buffer';
 EIDL                       pin 24 istype 'reg, buffer';
 ERFR                       pin 44 istype 'reg, buffer';
 E0                         pin 45 istype 'reg, buffer';
 E1                         pin 46 istype 'reg, buffer';
 EFOE_                      pin 47 istype 'reg, buffer';
 A_STXENB_                  pin 57 istype 'com, buffer';

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####     *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************

"*****************************************************************************
"* Ingress Recorder Internal
"*****************************************************************************

 SIRST                      node istype 'reg';
 SISIM                      node istype 'reg';
 SIGPS                      node istype 'reg';
 SIFULL                     node istype 'reg';
 IGPS                       node istype 'reg';

"*****************************************************************************
"* Egress Generator Internals
"*****************************************************************************

 E_READ_CYCLE               node istype 'reg';
 E_FIFO_ENABLE_             node istype 'reg';
 SERST, SESIM, SEFOE_       node istype 'reg';
```

```
 ERST, ESIM                node istype 'reg';
 R_WAS_ENB                 node istype 'reg';

 "*****************************************************************************
"*  #####                                                                     *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####          *
"* #         #    #  ##   #  #          #     #  #   ##   #     #            *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #            *
"* #         #    #  #  # #       #     #    ######  #  # #     #            *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #            *
"*  #####    ####   #    #   ####      #    #    #  #    #     #            *
"*                                                                           *
"* ######                                                                    *
"* #     #  ######   ####   #         ##    #####                           *
"* #     #  #       #    #  #        #  #   #    #                          *
"* #     #  #####   #       #       #    #  #    #  #####                    *
"* #     #  #       #       #       ######  #####                           *
"* #     #  #       #    #  #       #    #  #   #                           *
"* ######   ######   ####   ######  #    #  #    #                          *
"*                                                                           *
"*    ##     #####     #     ####   #    #                                   *
"*   #  #      #       #     #    #  ##   #                                   *
"*  #    #     #       #     #    #  # #  #                                   *
"*  ######     #       #     #    #  #  # #                                   *
"*  #    #     #       #     #    #  #   ##                                  *
"*  #    #     #       #      ####   #    #                                  *
"*****************************************************************************

H,L,X,C,P,Z = 1,0,.X.,.C.,.P.,.Z.;

D = [BD0,BD1,BD2,BD3,BD4,BD5,BD6,BD7];
ADDR = [BA8,BA9,BA10,BA11,BA12,BA13];

ISW_RAM_ADDR = [0,1,1,0,1,X];
IIL_CONTROL_REGISTER_ADDR = [0,1,1,1,0,X];
ESW_FIFO_ADDR = [0,1,1,1,1,X];
EIL_CONTROL_REGISTER_ADDR = [1,0,0,0,0,X];
GENERAL_STATUS_REGISTER_ADDR = [1,0,0,0,1,X];

IIL_CONTROL_REGISTER = [SIRST, SISIM, SIFULL, SIGPS];
EIL_CONTROL_REGISTER = [SESIM, SEFOE_, SERST];
GENERAL_STATUS_REGISTER = [ESTP,EIDL];

"*****************************************************************************
"* Egress Generator State Machine
"*****************************************************************************

        ESTATE     = [ EFRS_, EREN_  , EFRT_  ,  ERFR   , E1, E0];
        RESET_STATE =[   0    ,   1   ,   1    ,   0    , 0 , 0 ];
        IDLE_STATE  =[   1    ,   1   ,   1    ,   0    , 0 , 0 ];
        TX_STATE    =[   1    ,   0   ,   1    ,   1    , 0 , 0 ];
        WAIT_STATE  =[   1    ,   0   ,   1    ,   0    , 0 , 0 ];
        STOP_STATE  =[   1    ,   1   ,   1    ,   0    , 0 , 1 ];
        RET1_STATE  =[   1    ,   1   ,   1    ,   0    , 1 , 1 ];
        RET2_STATE  =[   1    ,   1   ,   0    ,   0    , 0 , 0 ];


State_Diagram ESTATE
```

```
State RESET_STATE:
         IF (ERST) THEN RESET_STATE
         ELSE IDLE_STATE;

State IDLE_STATE:
         IF (ERST) THEN RESET_STATE
         ELSE IF (ESIM) THEN WAIT_STATE
         ELSE IDLE_STATE;

State TX_STATE:
         IF (ERST) THEN RESET_STATE
         ELSE IF (ECCLV & !A_STXCLAV) THEN WAIT_STATE
         ELSE IF (EX_ESC & A_STXDATA0) THEN RET1_STATE
         ELSE IF (EX_ESC & !A_STXDATA0) THEN STOP_STATE
         ELSE TX_STATE;

State WAIT_STATE:
         IF (ERST) THEN RESET_STATE
         ELSE IF (A_STXCLAV) THEN TX_STATE
         ELSE WAIT_STATE;

State RET1_STATE:
         IF (ERST) THEN RESET_STATE
  ELSE RET2_STATE;

State RET2_STATE:
         IF (ERST) THEN RESET_STATE
  ELSE IF (ESIM & A_STXCLAV) THEN TX_STATE
  ELSE IF (ESIM & !A_STXCLAV) THEN WAIT_STATE
         ELSE RESET_STATE;

State STOP_STATE:
         IF (ERST) THEN RESET_STATE
         ELSE IF (ESIM) THEN STOP_STATE
         ELSE RESET_STATE

State 0: GOTO RESET_STATE;
State 1: GOTO RESET_STATE;
State 2: GOTO RESET_STATE;
State 3: GOTO RESET_STATE;
State 4: GOTO RESET_STATE;
State 5: GOTO RESET_STATE;
State 6: GOTO RESET_STATE;
State 7: GOTO RESET_STATE;
State 8: GOTO RESET_STATE;
State 9: GOTO RESET_STATE;
State 10:GOTO RESET_STATE;
State 11:GOTO RESET_STATE;
State 12:GOTO RESET_STATE;
State 13:GOTO RESET_STATE;
State 14:GOTO RESET_STATE;
State 15:GOTO RESET_STATE;
State 16:GOTO RESET_STATE;
State 17:GOTO RESET_STATE;
State 18:GOTO RESET_STATE;
State 19:GOTO RESET_STATE;
State 20:GOTO RESET_STATE;
State 21:GOTO RESET_STATE;
State 22:GOTO RESET_STATE;
```

```
State 23:GOTO RESET_STATE;
"State 24:GOTO RESET_STATE;
State 25:GOTO RESET_STATE;
State 26:GOTO RESET_STATE;
State 27:GOTO RESET_STATE;
State 28:GOTO RESET_STATE;
State 29:GOTO RESET_STATE;
State 30:GOTO RESET_STATE;
State 31:GOTO RESET_STATE;
State 32:GOTO RESET_STATE;
State 33:GOTO RESET_STATE;
State 34:GOTO RESET_STATE;
State 35:GOTO RESET_STATE;
State 36:GOTO RESET_STATE;
State 37:GOTO RESET_STATE;
State 38:GOTO RESET_STATE;
State 39:GOTO RESET_STATE;
State 30:GOTO RESET_STATE;
State 31:GOTO RESET_STATE;
State 32:GOTO RESET_STATE;
State 33:GOTO RESET_STATE;
State 34:GOTO RESET_STATE;
State 35:GOTO RESET_STATE;
State 36:GOTO RESET_STATE;
State 37:GOTO RESET_STATE;
State 38:GOTO RESET_STATE;
State 39:GOTO RESET_STATE;
"State 40:GOTO RESET_STATE;
State 41:GOTO RESET_STATE;
State 42:GOTO RESET_STATE;
State 43:GOTO RESET_STATE;
"State 44:GOTO RESET_STATE;
State 45:GOTO RESET_STATE;
State 46:GOTO RESET_STATE;
State 47:GOTO RESET_STATE;
"State 48:GOTO RESET_STATE;
State 49:GOTO RESET_STATE;
State 50:GOTO RESET_STATE;
State 51:GOTO RESET_STATE;
State 52:GOTO RESET_STATE;
State 53:GOTO RESET_STATE;
State 54:GOTO RESET_STATE;
State 55:GOTO RESET_STATE;
"State 56:GOTO RESET_STATE;
"State 57:GOTO RESET_STATE;
State 58:GOTO RESET_STATE;
"State 59:GOTO RESET_STATE;
State 60:GOTO RESET_STATE;
State 61:GOTO RESET_STATE;
State 62:GOTO RESET_STATE;
State 63:GOTO RESET_STATE;

"*****************************************************************************
"*                                                                           *
"* #######                                                                   *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #        #    #  #    #   #  #      #       #    #    #  ##   #  #         *
"* #####    #    #  #    #  #    #     #       #    #    #  # #  #   ####     *
"* #        #  # #  #    #  ######     #       #    #    #  #  # #       #    *
```

```
"* #       #   #   #    #  #    #     #       #    #    #  #   ##  #    #    *
"* #######   ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                             *
"*******************************************************************************

equations

A_SRXENB_.oe = !SW_IG_REC_En~;      " Ingress recorder initiates only when selected
ISIM.oe = H;
ISIM_.oe = H;
I_CNT_EN.oe = H;
IROE_.oe = H;

"*******************************************************************************
"* Ingress recorder Control register Power-On Reset defaults
"*******************************************************************************

SISIM.ar = !RESET_; " SISIM = 0
SIFULL.ap = !RESET_; " SIFULL = 1
SIGPS.ar = !RESET_;                 " SIGPS = 0

A_SRXENB_.clk = SXCLK;
ISIM.clk = SXCLK;
IGPS.clk = SXCLK;
E_READ_CYCLE.clk = SXCLK;
E_FIFO_ENABLE_.clk = SXCLK;
SIRST.clk = SXCLK;
SISIM.clk = SXCLK;
SIGPS.clk = SXCLK;
SIFULL.clk = SXCLK;

"*******************************************************************************
"* Ingress recorder Control Register initialization
"*******************************************************************************

when ((ADDR == IIL_CONTROL_REGISTER_ADDR) & !RW_ & !BWE0~ & !DFLT_CS_)
         then
                  IIL_CONTROL_REGISTER := [BD0,BD1,BD2,BD3];
         else
                  IIL_CONTROL_REGISTER := IIL_CONTROL_REGISTER.FB;

ISIM := SISIM.FB;
IGPS := SIGPS.FB;
ISIM_ = !ISIM;
I_CNT_EN = A_SRXCLAV # IGPS.FB;
A_SRXENB_ := SIFULL.FB # !A_SRXCLAV;
IROE_ = !(ADDR == ISW_RAM_ADDR) # DFLT_CS_ # RdOE~ # SW_IG_REC_En~;

"*******************************************************************************
"* Egress generator
"*******************************************************************************

equations

A_STXENB.oe = !SW_EG_GEN_En~;  " Egress generator initiates only when selected
EREN_.oe = H;
EFRT_.oe = H;
EFOE_.oe = H;
EFRS_.oe = H;
```

```
ESTP.oe = !DFLT_CS_ & (ADDR == GENERAL_STATUS_REGISTER_ADDR) & !RdOE~ & RESET_;
EIDL.oe = !DFLT_CS_ & (ADDR == GENERAL_STATUS_REGISTER_ADDR) & !RdOE~ & RESET_;

"*****************************************************************************
"* Egress generator Control Register Power-On Reset Defaults
"*****************************************************************************

SERST.ap = !RESET_;                                    " SERST = 1
SESIM.ar = !RESET_;                                    " SESIM = 0
SEFOE_.ap = !RESET_;                                   " SEFOE = 1

EREN_.clk = SXCLK;
EFRT_.clk = SXCLK;
EFOE_.clk = SXCLK;
EFRS_.clk = SXCLK;
ESTP.clk = SXCLK;
ERFR.clk = SXCLK;
EIDL.clk = SXCLK;
ESIM.clk = SXCLK;
E0.clk = SXCLK;
E1.clk = SXCLK;
ERST.clk = SXCLK;
SERST.clk = SXCLK;
SESIM.clk = SXCLK;
SEFOE_.clk = SXCLK;

"*****************************************************************************
"* Egress generator Control Register initialization
"*****************************************************************************

when ((ADDR == EIL_CONTROL_REGISTER_ADDR) & !RW_ & !BWE0~ & !DFLT_CS_)
         then
                  EIL_CONTROL_REGISTER := [BD0,BD1,BD2];
         else
                  EIL_CONTROL_REGISTER := EIL_CONTROL_REGISTER;

ERST := SERST;
ESIM := SESIM;
EFOE_ := SEFOE_ # SW_EG_GEN_En~;

"*****************************************************************************
"* Egress generator Status Register
"*****************************************************************************

ESTP := (ESTATE == STOP_STATE);
EIDL := (ESTATE == IDLE_STATE);
XEIDL = EIDL;

E_READ_CYCLE := !EREN_.FB & A_STXCLAV # ERFR.FB;

when (!ESIM)
         then
                  E_FIFO_ENABLE_ := 1;
         else when (E_READ_CYCLE)
                  then
                           E_FIFO_ENABLE_ := B_STXENB_;
                  else
                           E_FIFO_ENABLE_ := E_FIFO_ENABLE_;
```

```
A_STXENB_ = !(E_READ_CYCLE & !E_FIFO_ENABLE_);

end xmach
```

A.9 U37—Egress Recorder Equations

```
"*****************************************************************************
"* In this file :
"* - Programmable logic for the Egress Recorder.
"*****************************************************************************

module tmach
title   'TPH logic for ATMC_EVB Board
        Originated for ATMC EVB, Aviel Livay (MSIL) - April 04, 1996
        Modified for ATMC EVB, Amitay Beler - (MSIL) - March 1st, 1998'

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

u37  device 'mach220a';

"*****************************************************************************
"* #######
*
"* #        #    #   #####  ######  #####   #    #    ##    #        ####     *
"* #         #  #      #    #       #    #  ##   #   #  #   #       #         *
"* #####      ##       #    #####   #    #  # #  #  #    #  #        ####     *
"* #          ##       #    #       #####   #  # #  ######  #            #    *
"* #         #  #      #    #       #   #   #   ##  #    #  #       #    #    *
"* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*****************************************************************************

"*****************************************************************************
"* Pins declaration.                                                         *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************

A_ACLK                  pin 15;
RESET_                  pin 17;
BA13                    pin 2;
BA12                    pin 3;
BA11                    pin 4;
BA10                    pin 5;
BA9                     pin 6;
BA8                     pin 7;
BD7                     pin 9;
BD6                     pin 10;
BD5                     pin 11;
BD4                     pin 12;
BD3                     pin 13;
BD2                     pin 14;
BD1                     pin 21;
BD0                     pin 22;
DFLT_CS_                pin 16;
RW_                     pin 20;
```

```
A_TXENB_                    pin 49;
A_TXSOC                     pin 50;
BWE0~                       pin 25;
EG_REC_En~                  pin 51;
RdOE~                       pin 24;

"*****************************************************************************
"* Output pins
"*****************************************************************************

TPIM                        pin 29 istype 'reg, buffer';
A_TXFULL_                   pin 30 istype 'reg, buffer';
COUNT0                      pin 31 istype 'com, buffer';
T_CNT_EN                    pin 32 istype 'com, buffer';
TPIM_                       pin 33 istype 'com, buffer';
TROE_                       pin 36 istype 'com, buffer';

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #        ####    *
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #        *
"*    #     # #  #     #    #####   #    #  # #  #  #    #  #        ####    *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #   *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #   *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####    *
"*****************************************************************************

RCNT7, RCNT6, RCNT5, RCNT4,
RCNT3, RCNT2, RCNT1, RCNT0          node istype 'reg, buffer';
SCNT7, SCNT6, SCNT5, SCNT4,
SCNT3, SCNT2, SCNT1, SCNT0          node istype 'reg, buffer';
CNT7, CNT6, CNT5, CNT4,
CNT3, CNT2, CNT1, CNT0              node istype 'reg, buffer';
TGPS, TFST, TCLB                    node istype 'reg, buffer';
STRST, STPIM, STGPS, STFST, STCLB   node istype 'reg, buffer';
TSOC                                node istype 'com';
ZERO                                node istype 'com';

 "*****************************************************************************
"*  #####                                                                    *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####          *
"* #         #    #  ##   #  #          #     #  #   ##   #     #            *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #            *
"* #         #    #  #  # #       #     #    ######  #  # #     #            *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #            *
"*  #####    ####   #    #   ####      #    #    #  #    #     #            *
"*                                                                           *
"* ######                                                                   *
"* #     #  ######   ####   #         ##    #####                           *
"* #     #  #       #    #  #        #  #   #    #                          *
"* #     #  #####   #       #       #    #  #    #  #####                   *
"* #     #  #       #       #       ######  #####                           *
"* #     #  #       #    #  #       #    #  #   #                           *
"* ######   ######   ####   ######  #    #  #    #                          *
"*                                                                           *
"*    ##     #####     #     ####   #    #                                  *
"*   #  #      #       #    #    #  ##   #                                  *
"*  #    #     #       #    #    #  # #  #                                  *
"*  ######     #       #    #    #  #  # #                                  *
```

```
"*  #    #     #       #    #    #  #   ##                                  *
"*  #    #     #       #     ####   #    #                                   *
"*****************************************************************************

H,L,X,C,P,Z = 1,0,.X.,.C.,.P.,.Z.;

RCNT = [RCNT7, RCNT6, RCNT5, RCNT4, RCNT3, RCNT2, RCNT1, RCNT0];
SCNT = [SCNT7, SCNT6, SCNT5, SCNT4, SCNT3, SCNT2, SCNT1, SCNT0];
CNT = [CNT7, CNT6, CNT5, CNT4, CNT3, CNT2, CNT1, CNT0];
ADDR = [BA8,BA9,BA10,BA11,BA12,BA13];
D = [BD0,BD1,BD2,BD3,BD4,BD5,BD6,BD7];
TIL_CONTROL_REGISTER = [STRST, STPIM, STGPS, STFST, STCLB];

TPH_RAM_ADDR                =        [0,1,0,0,0,X];
TIL_CONTROL_REGISTER_ADDR=           [0,1,0,0,1,X];
TPH_RUN_COUNT_REGISTER_ADDR=         [0,1,0,1,0,X];
TPH_STOP_COUNT_REGISTER_ADDR=        [0,1,0,1,1,X];
GENERAL_STATUS_REGISTER_ADDR=        [0,1,1,0,0,X];

"*****************************************************************************
"*                                                                           *
"* #######                                                                   *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####    *
"* #        #    #  #    #   #  #      #       #     #    #  ##   #  #        *
"* #####    #    #  #    #  #    #     #       #     #    #  # #  #   ####    *
"* #        #  # #  #    #  ######     #       #     #    #  #  # #       #   *
"* #        #   #   #    #  #    #     #       #     #    #  #   ##  #    #   *
"* #######   ### #   ####   #    #     #       #      ####   #    #   ####    *
"*                                                                           *
"*****************************************************************************

equations

TPIM.oe = H;
A_TXFULL_.oe = !EG_REC_En~;    " Egress Recorder initiates only when selected
COUNT0.oe = H;
T_CNT_EN.oe = H;
TPIM_.oe = H;
TROE_.oe = H;

TPIM.clk = A_ACLK;
A_TXFULL_.clk = A_ACLK;
TGPS.clk = A_ACLK;
TCLB.clk = A_ACLK;
TFST.clk = A_ACLK;
CNT.clk = A_ACLK;

STRST.clk = A_ACLK;
STPIM.clk = A_ACLK;
STGPS.clk = A_ACLK;
STFST.clk = A_ACLK;
STCLB.clk = A_ACLK;
RCNT.clk = A_ACLK;
SCNT.clk = A_ACLK;

"*****************************************************************************
"* Egress Recorder Control Register Power-On Reset Defaults
"*****************************************************************************
```

```
STRST.ap = !RESET_;                          " STRST = 1
STPIM.ar = !RESET_;                          " STPIM = 0
STGPS.ar = !RESET_;                          " STGPS = 0
STFST.ap = !RESET_;                          " STFST = 1
STCLB.ap = !RESET_;                          " STCLB = 1

"*****************************************************************************
"* Egress Recorder Control Registers initialization
"*****************************************************************************

when ((ADDR == TIL_CONTROL_REGISTER_ADDR) & !DFLT_CS_ & !BWE0~)
        then
                TIL_CONTROL_REGISTER := [BD0,BD1,BD2,BD3,BD4];
        else
                TIL_CONTROL_REGISTER := TIL_CONTROL_REGISTER;

when ((ADDR == TPH_RUN_COUNT_REGISTER_ADDR) & !DFLT_CS_ & !BWE0~)
        then
                RCNT := D;
        else
                RCNT := RCNT;

when ((ADDR == TPH_STOP_COUNT_REGISTER_ADDR) & !DFLT_CS_ & !BWE0~)
        then
                SCNT := D;
        else
                SCNT := SCNT;

TPIM := STPIM;
TPIM_ = !TPIM;
TGPS := STGPS;
TFST := STFST;
TCLB := STCLB;
T_CNT_EN = !A_TXENB_ # TGPS ;
TROE_ = !(ADDR == TPH_RAM_ADDR) # DFLT_CS_ # RdOE~ # EG_REC_En~;

CNT7 :=
        TPIM.FB &  TSOC & RCNT7 #
        TPIM.FB & !TSOC & ZERO & A_TXFULL_.FB & SCNT7 #
        TPIM.FB & !TSOC &  ZERO & !A_TXFULL_.FB & RCNT7 #
        TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT6 & !CNT5 & !CNT4 &
                                    !CNT3 & !CNT2 & !CNT1 & !CNT0 & !CNT7 #
        TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT6 & !CNT5 & !CNT4 & !CNT3 &
                                             !CNT2 & !CNT1 & !CNT0 & !CNT7 #
        TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT6 & !CNT5 &
                                !CNT4 & !CNT3 & !CNT2 & !CNT1 & !CNT0 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT6 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT5 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT4 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT3 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT2 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT7 #
        TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT7 ;
CNT6 :=
        TPIM.FB &  TSOC & RCNT6 #
        TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT6 #
        TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT6 #
        TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT5 & !CNT4 & !CNT3 &
                                             !CNT2 & !CNT1 & !CNT0 & !CNT6 #
```

```
         TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT6 & !CNT5 & !CNT4 & !CNT3 &
                                              !CNT2 & !CNT1 & !CNT0 & !CNT7 #
         TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT6 & !CNT5 &
                                 !CNT4 & !CNT3 & !CNT2 & !CNT1 & !CNT0 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT5 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT4 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT3 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT2 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT6 #
         TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT6 ;

CNT5 :=  TPIM.FB &  TSOC & RCNT5 #
         TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT5 #
         TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT5 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT4 & !CNT3 & !CNT2 &
                                                 !CNT1 & !CNT0 & !CNT5 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT4 & !CNT3 & !CNT2 & !CNT1 & !CNT0 &
!CNT5 #
         TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT4 & !CNT3 &
                                              !CNT2 & !CNT1 & !CNT0 & CNT5 #
         TPIM.FB & !TSOC & !ZERO  & CNT4 & CNT5 #
         TPIM.FB & !TSOC & !ZERO  & CNT3 & CNT5 #
         TPIM.FB & !TSOC & !ZERO  & CNT2 & CNT5 #
         TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT5 #
         TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT5 ;

CNT4 :=  TPIM.FB &  TSOC & RCNT4 #
         TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT4 #
         TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT4 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT3 & !CNT2 & !CNT1 & !CNT0 & !CNT4 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT3 & !CNT2 & !CNT1 & !CNT0 & !CNT4 #
         TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT3 &
                                                 !CNT2 & !CNT1 & !CNT0 & CNT4 #
         TPIM.FB & !TSOC & !ZERO  & CNT3 & CNT4 #
         TPIM.FB & !TSOC & !ZERO  & CNT2 & CNT4 #
         TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT4 #
         TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT4 ;

CNT3 :=  TPIM.FB &  TSOC & RCNT3 #
         TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT3 #
         TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT3 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT2 & !CNT1 & !CNT0 & !CNT3 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT2 & !CNT1 & !CNT0 & !CNT3 #
         TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT2 & !CNT1 & !CNT0 &
CNT3 #
         TPIM.FB & !TSOC & !ZERO  & CNT2 & CNT3 #
         TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT3 #
         TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT3 ;

CNT2 :=  TPIM.FB &  TSOC & RCNT2 #
         TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT2 #
         TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT2 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT1 & !CNT0 & !CNT2 #
         TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT1 & !CNT0 & !CNT2 #
         TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT1 & !CNT0 & CNT2 #
         TPIM.FB & !TSOC & !ZERO  & CNT1 & CNT2 #
         TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT2 ;

CNT1 :=  TPIM.FB &  TSOC & RCNT1 #
```

```
          TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT1 #
          TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT1 #
          TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT0 & !CNT1 #
          TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT0 & !CNT1 #
          TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & !CNT0 & CNT1 #
          TPIM.FB & !TSOC & !ZERO  & CNT0 & CNT1;

CNT0 :=   TPIM.FB &  TSOC & RCNT0 #
          TPIM.FB & !TSOC & ZERO  & A_TXFULL_.FB & SCNT0 #
          TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB & RCNT0 #
          TPIM.FB & !TSOC & !ZERO  & !A_TXFULL_.FB & !CNT0 #
          TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & !CNT0 #
          TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB  & CNT0 ;


A_TXFULL_ :=
          TPIM.FB &  TFST.FB #
          TPIM.FB &  TSOC #
          TPIM.FB & !TSOC & ZERO  & !A_TXFULL_.FB #
          TPIM.FB & !TSOC & !ZERO  & !A_TXENB_ & A_TXFULL_.FB #
          TPIM.FB & !TSOC & !ZERO  & A_TXENB_ & A_TXFULL_.FB ;

ZERO = (CNT == 0);
TSOC = !A_TXENB_ & A_TXSOC & TCLB.FB;
COUNT0 = CNT0;


end tmach
```

A.10 U27 and U15 Device ‘mach220a’;

```
U15,U27 - Address Counter for the recorders
”*************************************************************************
“* In this file :
“* - Programmable logic for the Address Counter of the Egress and Ingress Recorders.
“*************************************************************************

module addgen
title    ‘19 bit addgen implementation’
        Originated for ATMC EVB, Aviel Livay (MSIL) - April 04, 1996’

“*************************************************************************
“* Device declaration.                                                   *
“*************************************************************************

“*************************************************************************
“* #######
*
“* #        #    #   #####  ######  #####   #    #    ##    #        ####     *
“* #         # #       #    #       #    #  ##   #   #  #   #       #         *
“* #####      ##       #    #####   #    #  # #  #  #    #  #        ####     *
“* #          ##       #    #       #####   #  # #  ######  #            #    *
“* #         #  #      #    #       #   #   #   ##  #    #  #       #    #    *
“* #######  #    #     #    ######  #    #  #    #  #    #  ######   ####     *
“*************************************************************************

“*************************************************************************
```

```
"* Pins declaration.                                                          *
"*****************************************************************************
"* System i/f pins
"*****************************************************************************

CLK                         pin 15; " Clock in
CEN                         pin 16; " Count enable
MODE                        pin 17; " count mode

"***********************************************************
" MPC860 address lines
"***********************************************************
A20                         pin 28;
A19                         pin 26;
A18                         pin 25;
A17                         pin 24;
A16                         pin 23;
A15                         pin 22;
A14                         pin 21;
A13                         pin 14;
A12                         pin 13;
A11                         pin 12;
A10                         pin 11;
A9                          pin 10;
A8                          pin 9;
A7                          pin 7;
A6                          pin 6;
A5                          pin 5;
A4                          pin 4;
A3                          pin 3;
A2                          pin 2;

"***********************************************************
" Recorder address lines
"***********************************************************

RA2                         pin 29 istype 'reg, buffer';
RA3                         pin 30 istype 'reg, buffer';
RA4                         pin 31 istype 'reg, buffer';
RA5                         pin 32 istype 'reg, buffer';
RA6                         pin 33 istype 'reg, buffer';
RA7                         pin 36 istype 'reg, buffer';
RA8                         pin 37 istype 'reg, buffer';
RA9                         pin 67 istype 'reg, buffer';
RA10                        pin 39 istype 'reg, buffer';
RA11                        pin 40 istype 'reg, buffer';
RA12                        pin 41 istype 'reg, buffer';
RA13                        pin 43 istype 'reg, buffer';
RA14                        pin 44 istype 'reg, buffer';
RA15                        pin 45 istype 'reg, buffer';
RA16                        pin 46 istype 'reg, buffer';
RA17                        pin 47 istype 'reg, buffer';
RA18                        pin 58 istype 'reg, buffer';
RA19                        pin 55 istype 'reg, buffer';
RA20                        pin 57 istype 'reg, buffer';

"*****************************************************************************
"*   ###                                                                     *
"*    #     #    #   #####  ######  #####   #    #    ##    #       ####    *
```

```
"*    #     ##   #     #    #       #    #  ##   #   #  #   #       #         *
"*    #     # #  #     #    #####   #    #  # #  #  #    #   #        ####     *
"*    #     #  # #     #    #       #####   #  # #  ######  #            #    *
"*    #     #   ##     #    #       #   #   #   ##  #    #  #       #    #    *
"*   ###    #    #     #    ######  #    #  #    #  #    #  ######   ####     *
"*************************************************************************

TCLOW                              node istype 'com';    " count low

OLD_MODE                           node istype 'reg';

 "*************************************************************************
"*  #####                                                                 *
"* #     #   ####   #    #   ####    #####    ##    #    #   #####       *
"* #         #    #  ##   #  #          #     #  #   ##   #     #         *
"* #         #    #  # #  #   ####      #    #    #  # #  #     #         *
"* #         #    #  #  # #       #     #    ######  #  # #     #         *
"* #     #   #    #  #   ##  #    #     #    #    #  #   ##     #         *
"*  #####    ####   #    #   ####      #    #    #  #    #     #         *
"*                                                                       *
"* ######                                                                *
"* #     #  ######   ####   #         ##    #####                        *
"* #     #  #       #    #  #        #  #   #    #                       *
"* #     #  #####   #       #       #    #  #    #  #####                 *
"* #     #  #       #       #       ######  #####                        *
"* #     #  #       #    #  #       #    #  #   #                        *
"* ######   ######   ####   ######  #    #  #    #                       *
"*                                                                       *
"*    ##     #####     #     ####   #    #                               *
"*   #  #      #       #    #    #  ##   #                               *
"*  #    #     #       #    #    #  # #  #                               *
"*  ######     #       #    #    #  #  # #                               *
"*  #    #     #       #    #    #  #   ##                               *
"*  #    #     #       #     ####   #    #                               *
"*************************************************************************

COUNTH = [RA20, RA19, RA18, RA17, RA16, RA15, RA14, RA13, RA12];
COUNTL = [RA11, RA10, RA9, RA8, RA7, RA6 , RA5, RA4, RA3, RA2];

ADD = [A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10,
                               A9, A8 ,A7, A6, A5, A4, A3, A2];

ADDH = [A20, A19, A18, A17, A16, A15, A14, A13, A12];

ADDL = [A11, A10, A9, A8 ,A7, A6, A5, A4, A3, A2];

H,L,X,Z,C = 1, 0, .X., .Z., .C.;

ON = 1;
OFF = 0;

"*************************************************************************
"*                                                                       *
"* #######                                                               *
"* #         ####   #    #    ##     #####     #     ####   #    #   ####     *
"* #        #    #  #    #   #  #      #       #    #    #  ##   #  #         *
"* #####    #    #  #    #  #    #     #       #    #    #  # #  #   ####     *
"* #        #  # #  #    #  ######     #       #    #    #  #  # #       #    *
"* #        #   #   #    #  #    #     #       #    #    #  #   ##  #    #    *
```

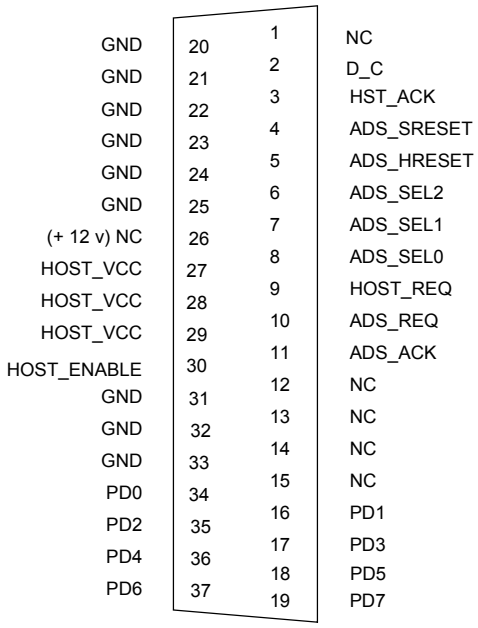
```
"* #######   ### #   ####   #    #     #       #     ####   #    #   ####     *
"*                                                                              *
"********************************************************************************

equations

WHEN (MODE == OFF)
         THEN
                  COUNTL := ADDL;
         ELSE WHEN ((MODE == ON) & (OLD_MODE.FB == OFF))
                  THEN
                           COUNTL := 0;
                  ELSE WHEN ((MODE == ON) & (OLD_MODE.FB == ON) & CEN)
                           THEN
                                    COUNTL := COUNTL + 1;
                           ELSE
                                    COUNTL := COUNTL;

TCLOW = (COUNTL == [1,1,1,1,1,1,1,1,1,1]);

WHEN (MODE == OFF)
         THEN
                  COUNTH := ADDH;
         ELSE WHEN ((MODE == ON) & (OLD_MODE.FB == OFF))
                  THEN
                           COUNTH := 0;
                  ELSE WHEN ((MODE == ON) & (OLD_MODE.FB == ON) & CEN & TCLOW)
                           THEN
                                    COUNTH := COUNTH + 1;
                           ELSE
                                    COUNTH := COUNTH;

COUNTH.C  = CLK;
COUNTH.OE = [H, H, H, H, H, H, H, H, H, H];
COUNTL.C  = CLK;
COUNTL.OE = [H, H, H, H, H, H, H, H, H, H];
OLD_MODE := MODE;
OLD_MODE.C = CLK;

"*************************************************************
" Test Vectors
"*************************************************************

TEST_VECTORS ( [CLK  , MODE  , ADDL, CEN] -> [COUNTL])
                [.C.  ,  0     ,  3   , X ] -> [3];
                [.C.  ,  0     ,  3   , X ] -> [3];
                [.C.  ,  0     ,  3   , X ] -> [3];
                [.C.  ,  1     ,  3   , X ] -> [0];
                [.C.  ,  1     ,  3   , 0 ] -> [0];
                [.C.  ,  1     ,  6   , 1 ] -> [2];
                [.C.  ,  1     ,  6   , 1 ] -> [3];
                [.C.  ,  0     ,  6   , X ] -> [6];
end addgen;
```

B

Application Development Interface (ADI)

B.1 Introduction

The ADI parallel port supplies parallel link from the MPCADS 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, and VMEbus computers and SUN-4 SPARC stations. It is possible to connect the ATMC EVB board to these computers provided that the appropriate software drivers are installed on them. Each ATMC EVB can have eight possible slave addresses set for its ADI port, enabling up to eight EVBs to be connected to the same ADI board. The ADI port connector is a 37-pin D-type plug connector. The connection between the ATMC EVB and the host computer is by a 37-line flat cable that is supplied with the ADI board. **Figure B-1** shows the pin configuration of the connector.



Note: Pin 26 on the ADI is connected to +12 v power supply, but it is not used in the MPC281ADS.

Figure B-1. ADI Port Connector

B.2 ADI Port Signal Description

The ADI port on the ATMC EVB was modified slightly to generate either a Hardware Reset or a Software Reset. This feature was added to comply with the MPC reset mechanism. In the list below, the directions I, O, and I/O are relative to the ATMC EVB (i.e., I means input to the EVB).

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.

- **ADS_SEL0–2 (I)**—These three input lines determine the slave address of the ATMC EVB being accessed by the host computer. One ADI can address up to eight EVBs.
- **ADS_SRESET (I)**—This input line is used to generate Software Reset for the MPC. When an ads is selected and this line is asserted by the host computer, a Software Reset will be generated to the MPC along with the Software 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 EVB, 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 Hardware Reset to the ATMC EVB 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 ATMC EVB write cycle.
- **ADS_ACK (O)**—This signal is the ATMC EVB response to the HOST_REQ signal, indicating that the board has detected the assertion of HOST_REQ.
- **ADS_REQ (O)**—This signal initiates an ATMC EVB to host write cycle.
- **HST_ACK (I)**—This signal serves as the host's response to the ADS_REQ signal.
- **HOST_VCC (I)**—These three lines are power lines from the host computer. In the ATMC EVB, these lines are used by the hardware to determine if the host computer is powered on.
- **PD0–7 (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.

NOTE: A complete description of the interface signals is provided in **Section 6.2.8**.

C

ADI Installation

This appendix describes the hardware installation of the ADI board into various host computers. The installation instructions cover the following host computers:

- PC-compatible computer
- SUN-4 (SBus interface)

C.1 PC-Compatible to ATMC EVB Interface

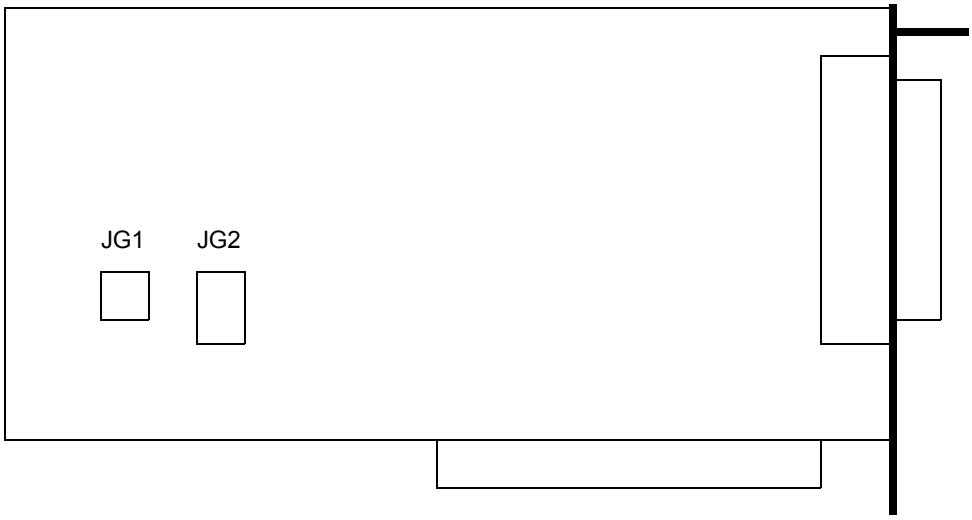
The ADI board should be installed in one of the PC-compatible system expansion slots. A single ADI can control up to eight ATMC EVBs. The ADI address in the computer is configured to be at I/O memory addresses $100–$102, but it may be reconfigured for an alternate address space.

CAUTION

Before removing or installing any equipment in a PC-compatible computer, turn the power off and remove the power cord.

1. Refer to the appropriate Installation and Setup manual of the PC-compatible computer for instructions and remove the computer cover.
2. Configure the ADI board address block JG1 to select a free I/O address space in the computer. **Figure C-1** shows the jumper location on the ADI board. 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.
   - — $200—This address is usually used for the game port.
   - — $300—This address is defined as a prototype port.

NOTE: The ADI board is factory configured for address decoding at $100–$102 in the PC-compatible I/O address map. These are undefined peripheral addresses.



Notes: **1.** Refer to **Figure C-2** to select the desired address configuration using JG1.
2. Leave JG2 unconnected.

Figure C-1. PC-Compatible ADI Jumper Location

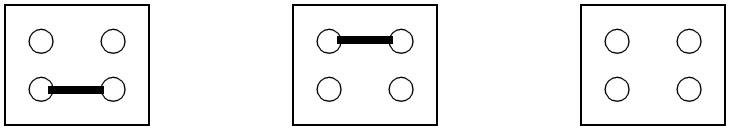


Figure C-2. JG1 Configuration Options

3. Install the ADI board by positioning its front bottom corner in the plastic card guide channel at the front of the PC-compatible 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.
4. 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.
5. Connect the 37-pin interface flat cable to the ADI board and secure.
6. Turn power on to the system unit and check for proper operation.

C.2 SUN-4 to ATMC EVB Interface

The SBus ADI board (shown in **Figure C-3**) should be installed in one of the SBus expansion slots in the Sun-4 SPARCstation computer. A single ADI can control up to eight MPC821/860ADS boards.

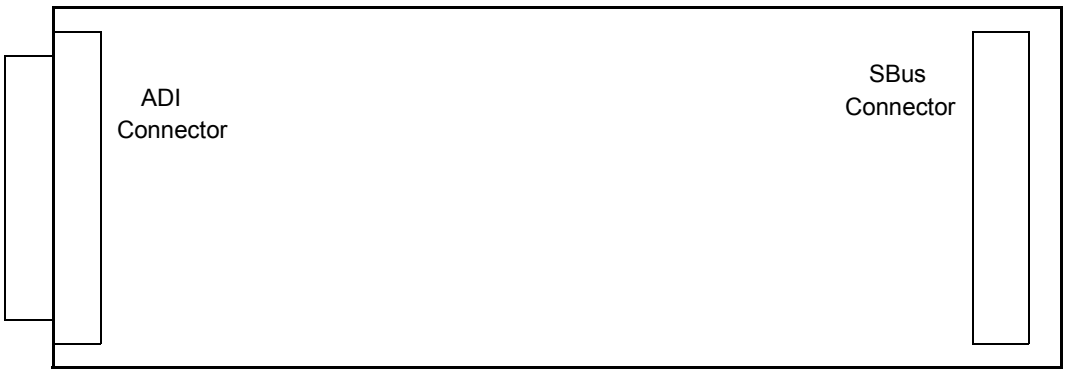


Figure C-3. SBus ADI Board

NOTE: 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.

CAUTION

Before removing or installing any equipment in a Sun-4 computer, ensure that you use proper procedure to shut your system and then turn the power off.

1. 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.

NOTE: The following instructions summarize those in the Sun manual:

2. Be sure to save all open files and then use the following steps to 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.

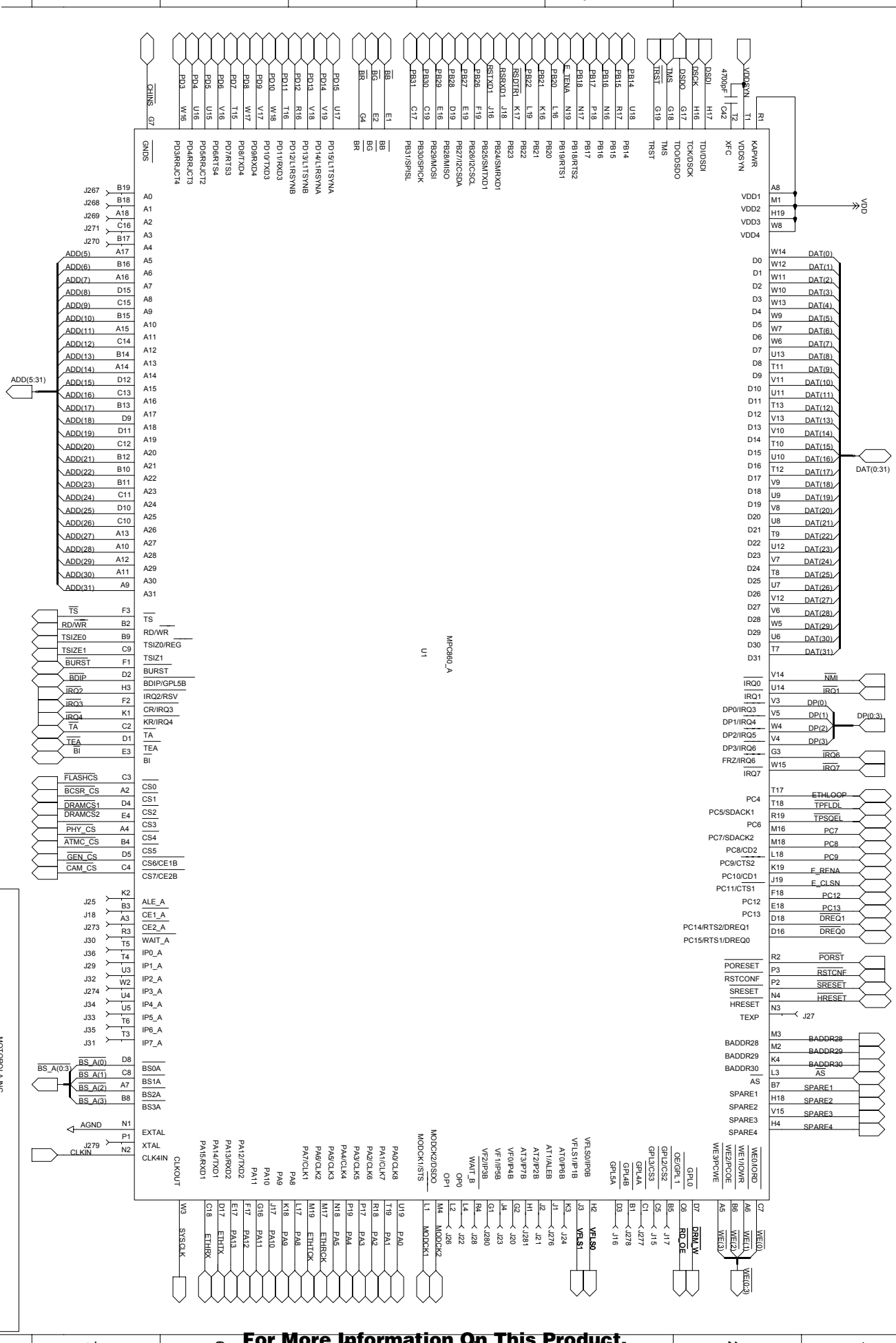
3. Turn off power to the system, but keep the power cord plugged in.
4. Open the system unit. Be sure to attach a grounding strap to your wrist and to the metal casing of the power supply.
5. Follow the instructions supplied with your system to gain access to the SBus slots.
6. Remove the SBus slot filler panel for the desired slot from the inner surface of the back panel of the system unit.

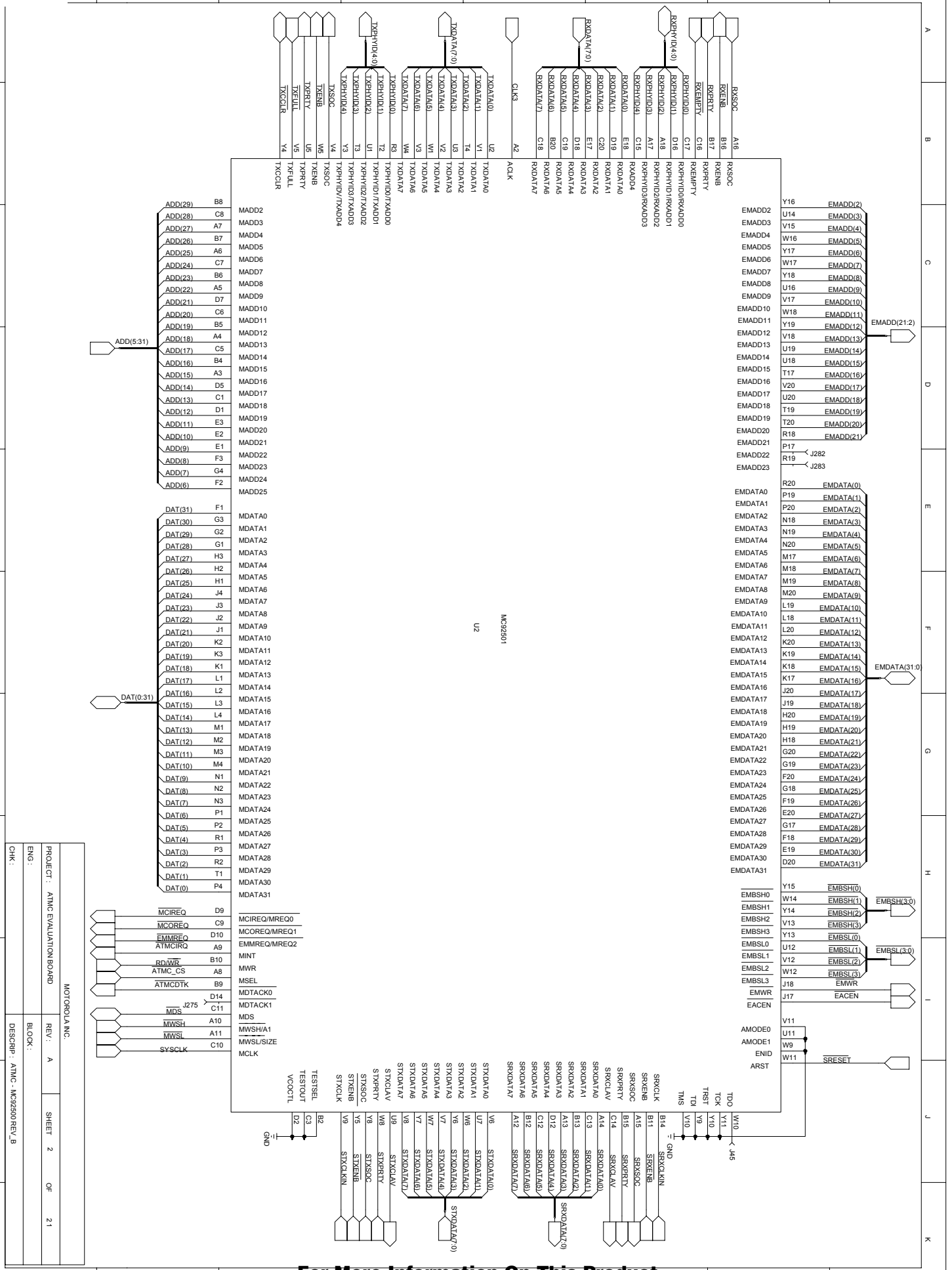
NOTE: The ADI board is a slave only board and thus will function in any available SBus slot.

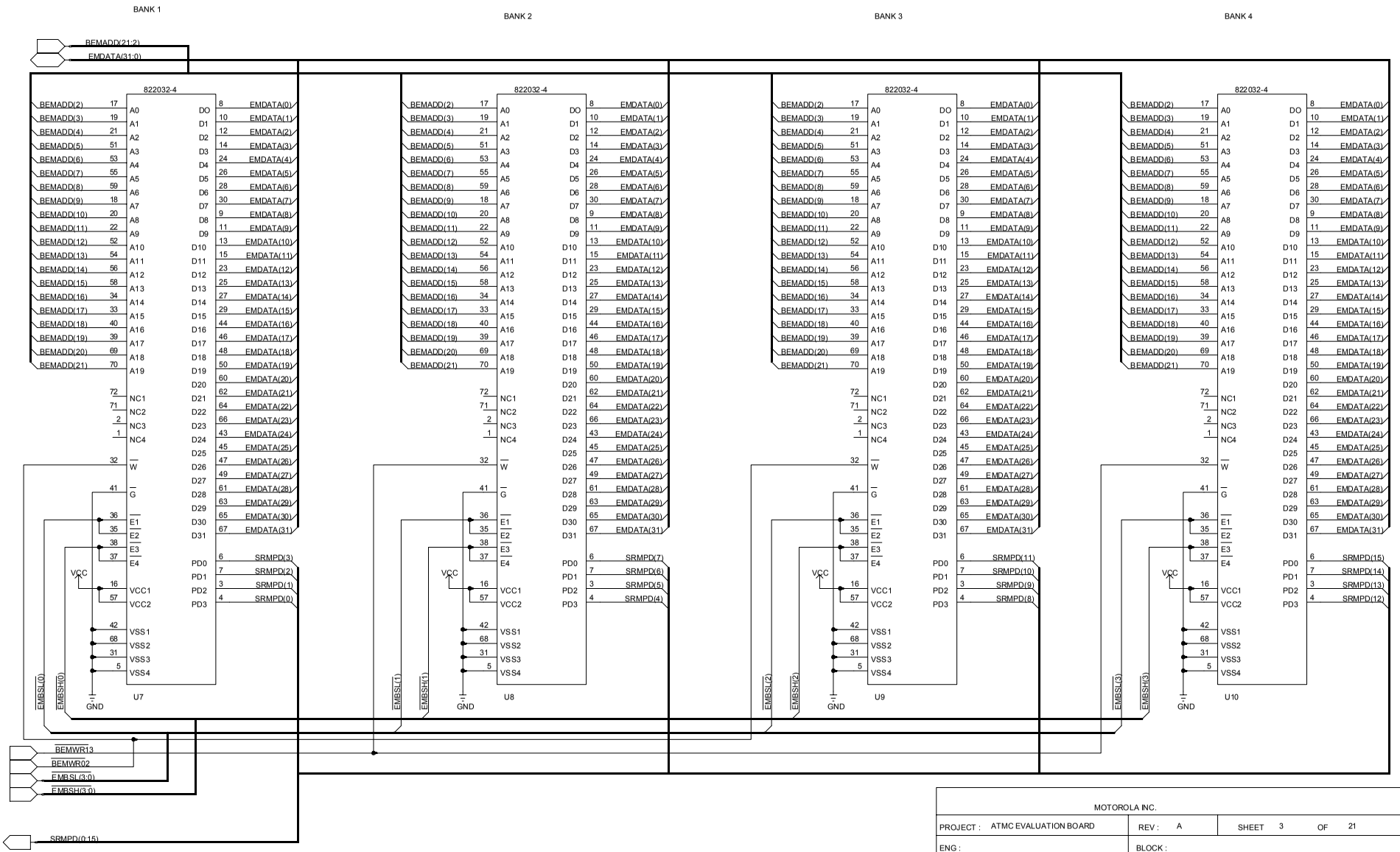
7. Slide the ADI board at an angle into the back panel of the system unit. Make sure that the mounting plate on the ADI board hooks into the holes on the back panel of the system unit.
8. Push the ADI board against the back panel and align the connector with its mate and gently press the corners of the board to seat the connector firmly.
9. Close the system unit.
10. Connect the 37-pin interface flat cable to the ADI board and secure.
11. Turn power on to the system unit and check for proper operation.

D

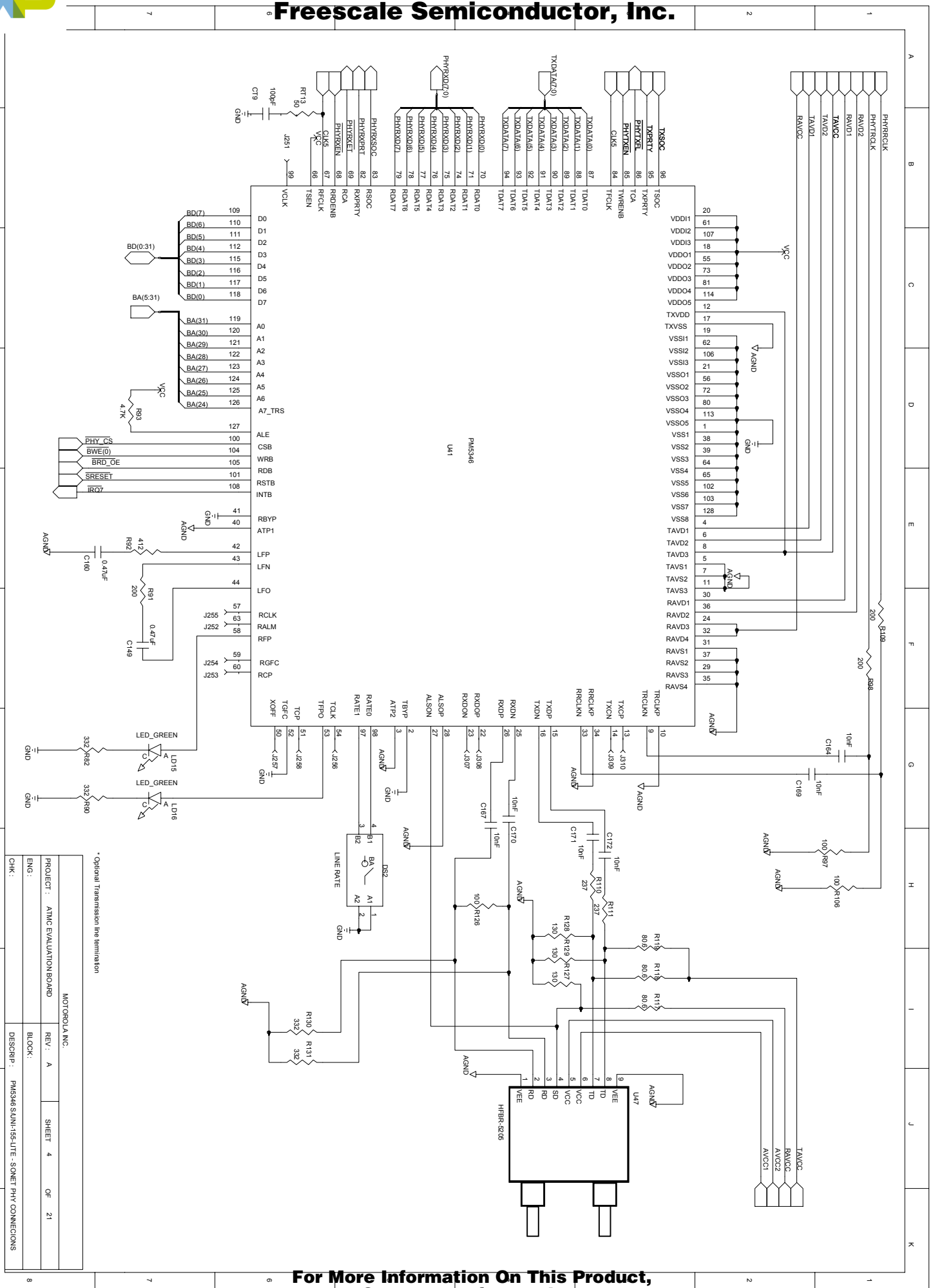
ATMC EVB Schematics

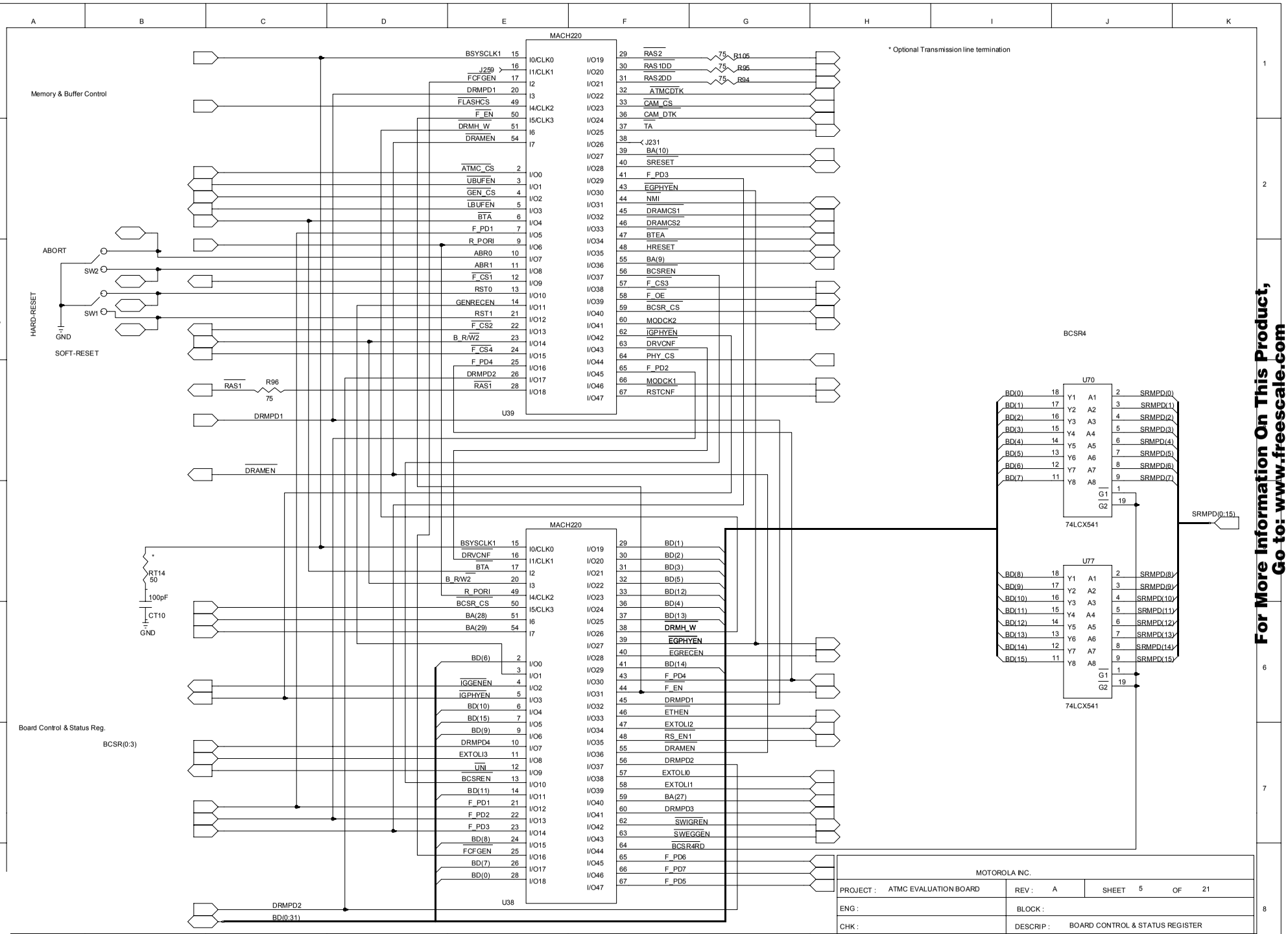


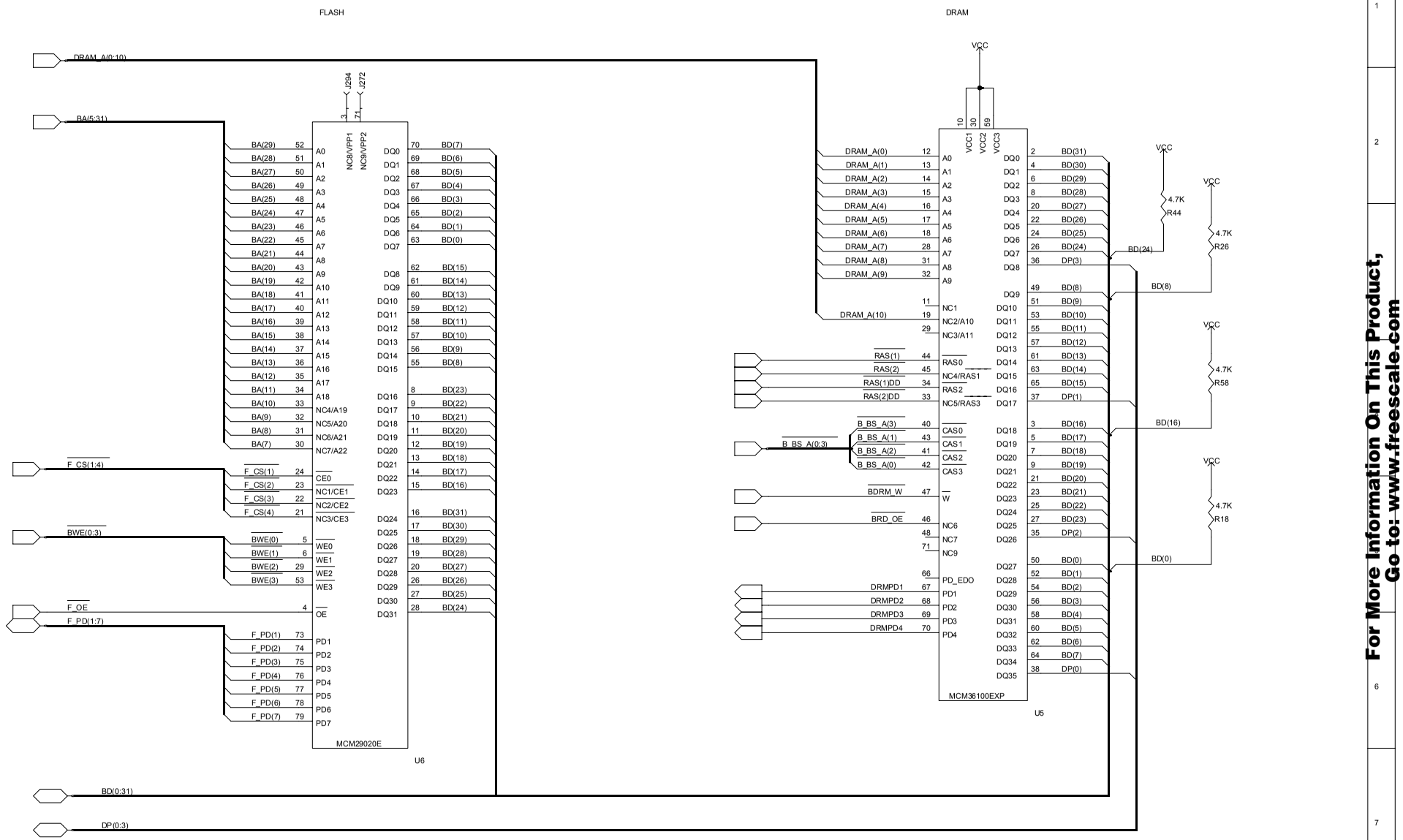




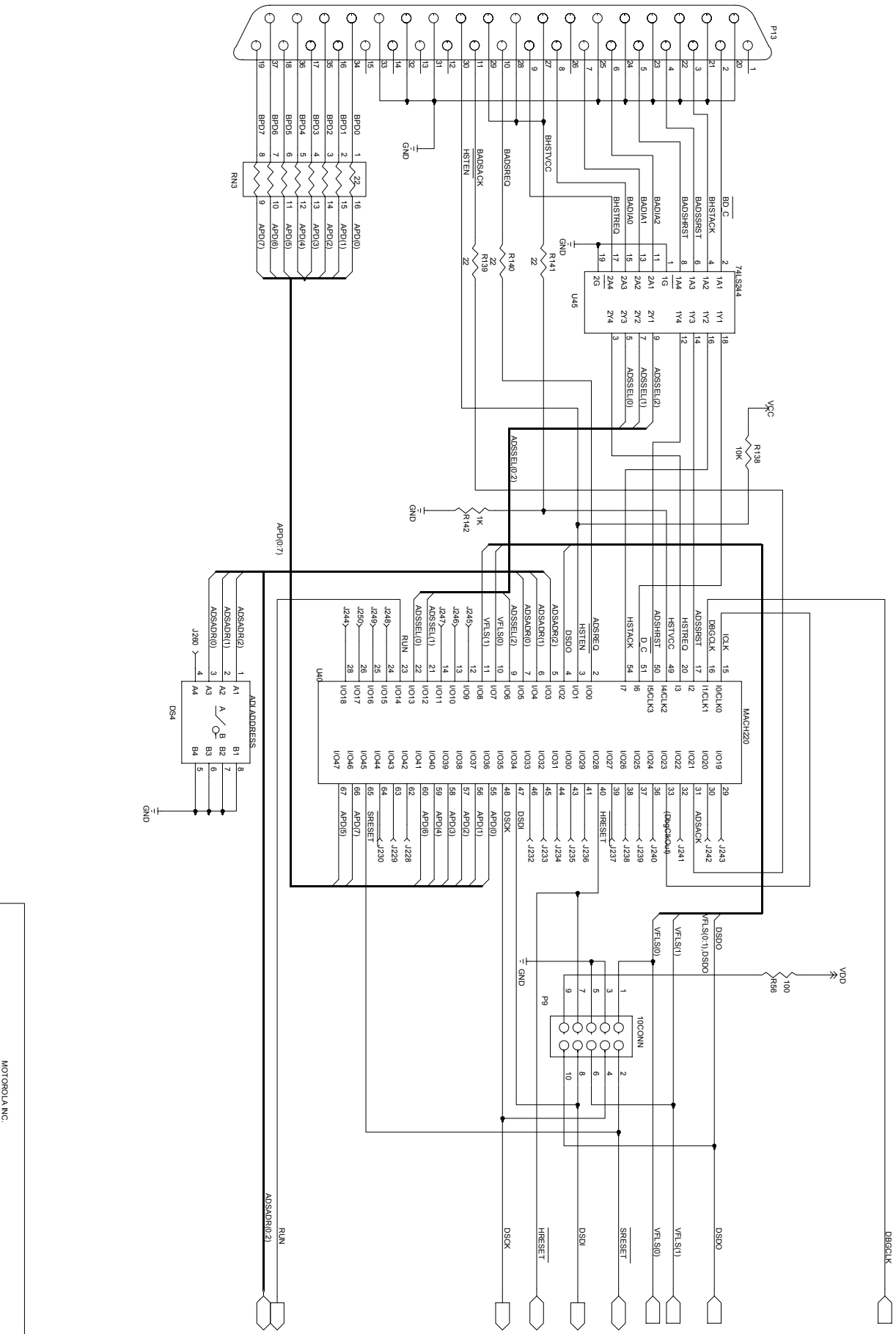
MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 3	OF 21
ENG :	BLOCK :		
CHK :	DESCRIP : ATMC EXTERNAL MEMORY BANKS		



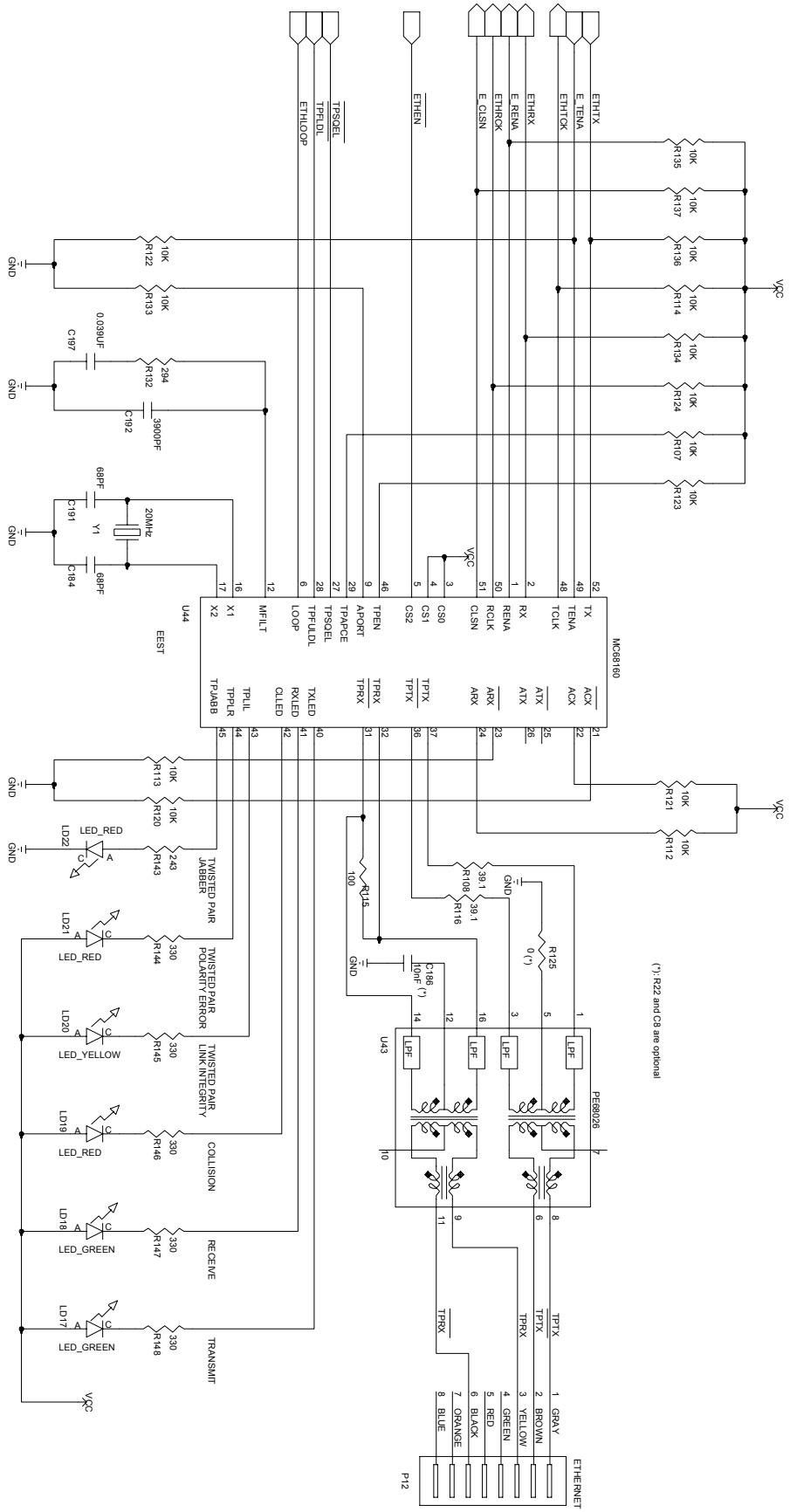


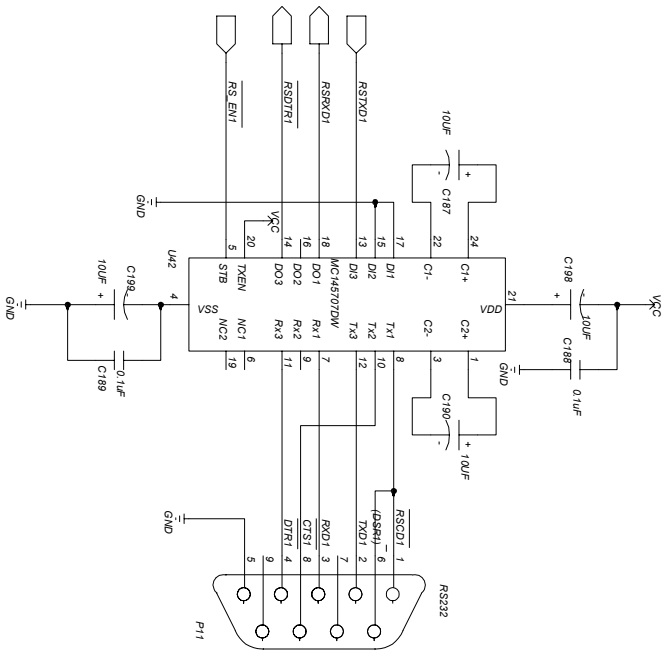


MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 6 OF 21	
ENG :	BLOCK :		
CHK :	DESCRIP : EDO DRAM & FLASH EPROM		



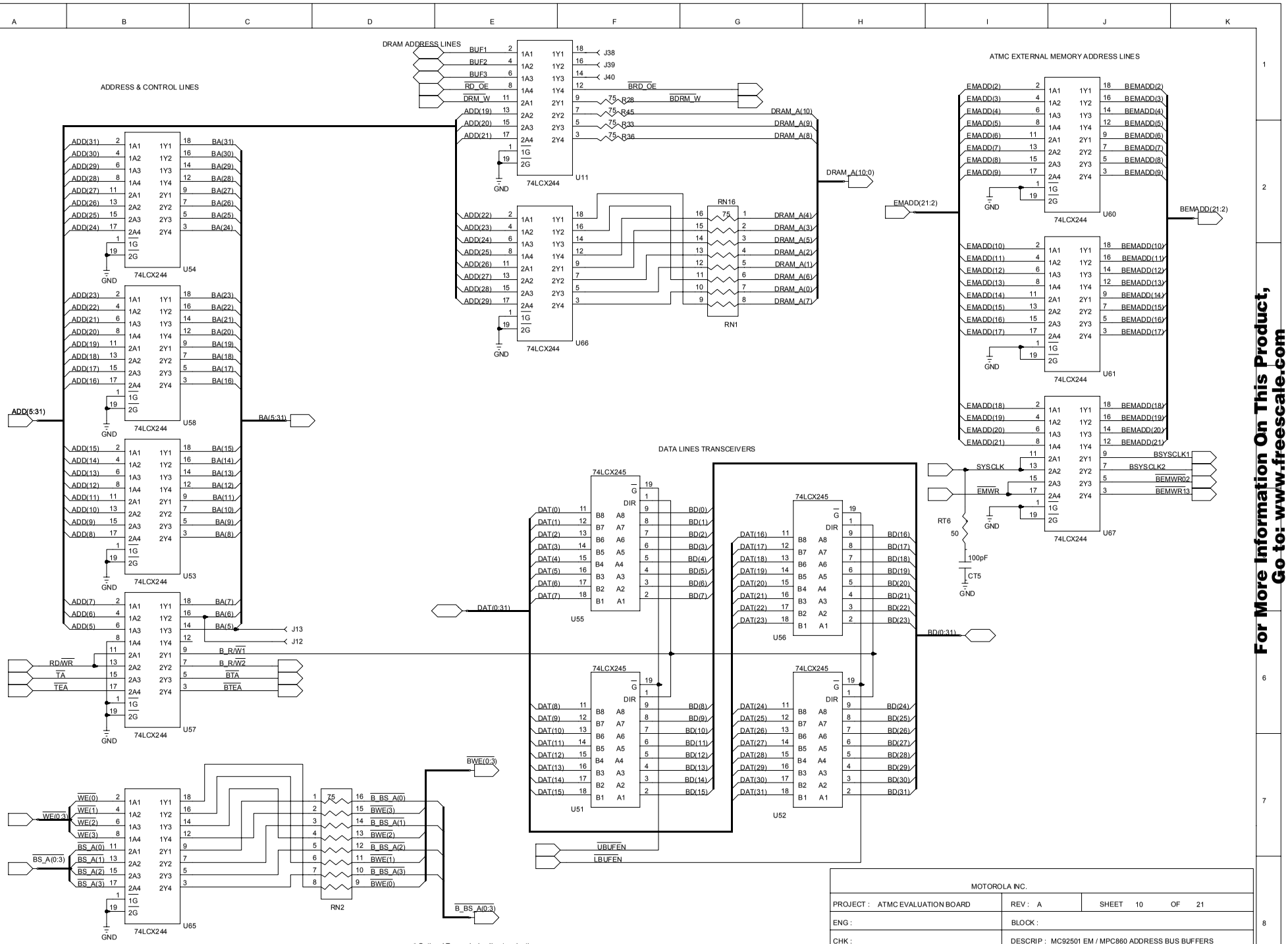
MOTOROLA INC.	
PROJECT : ATMC EVALUATION BOARD	REV: A SHEET 7 OF 21
ENG :	BLOCK :
CHK :	DESCRIP : ADI - DEBUG PORT CONNECTIONS

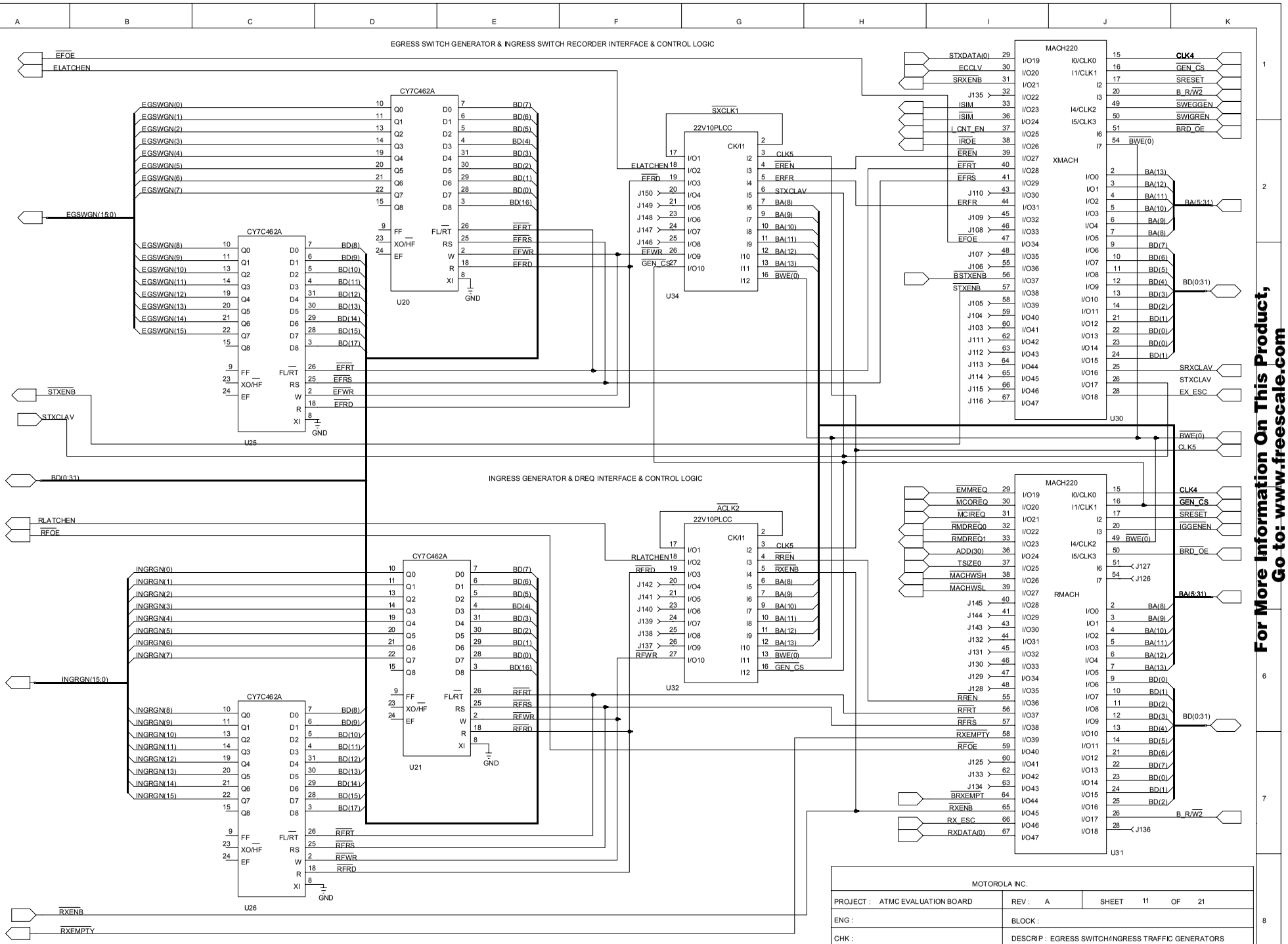


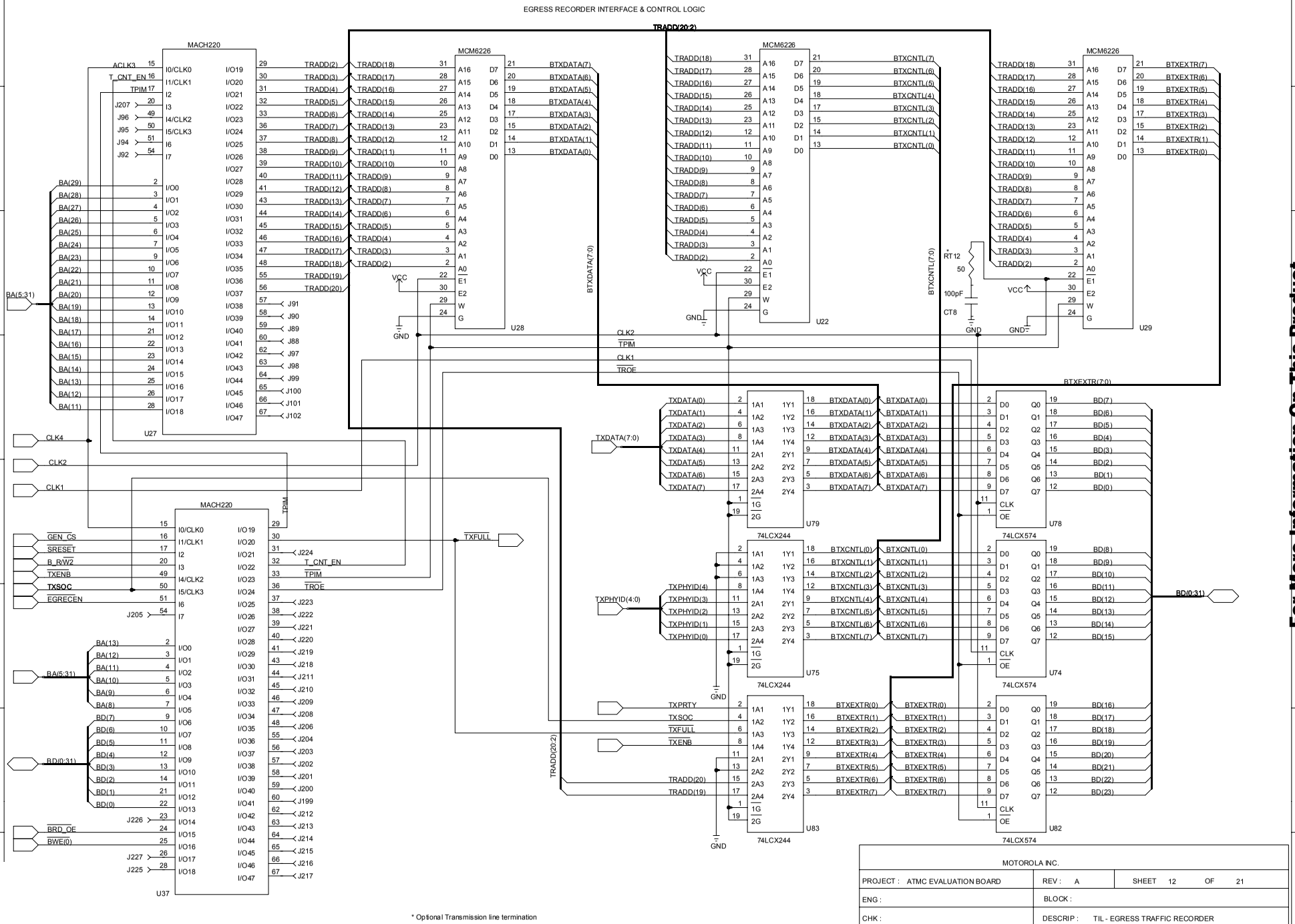


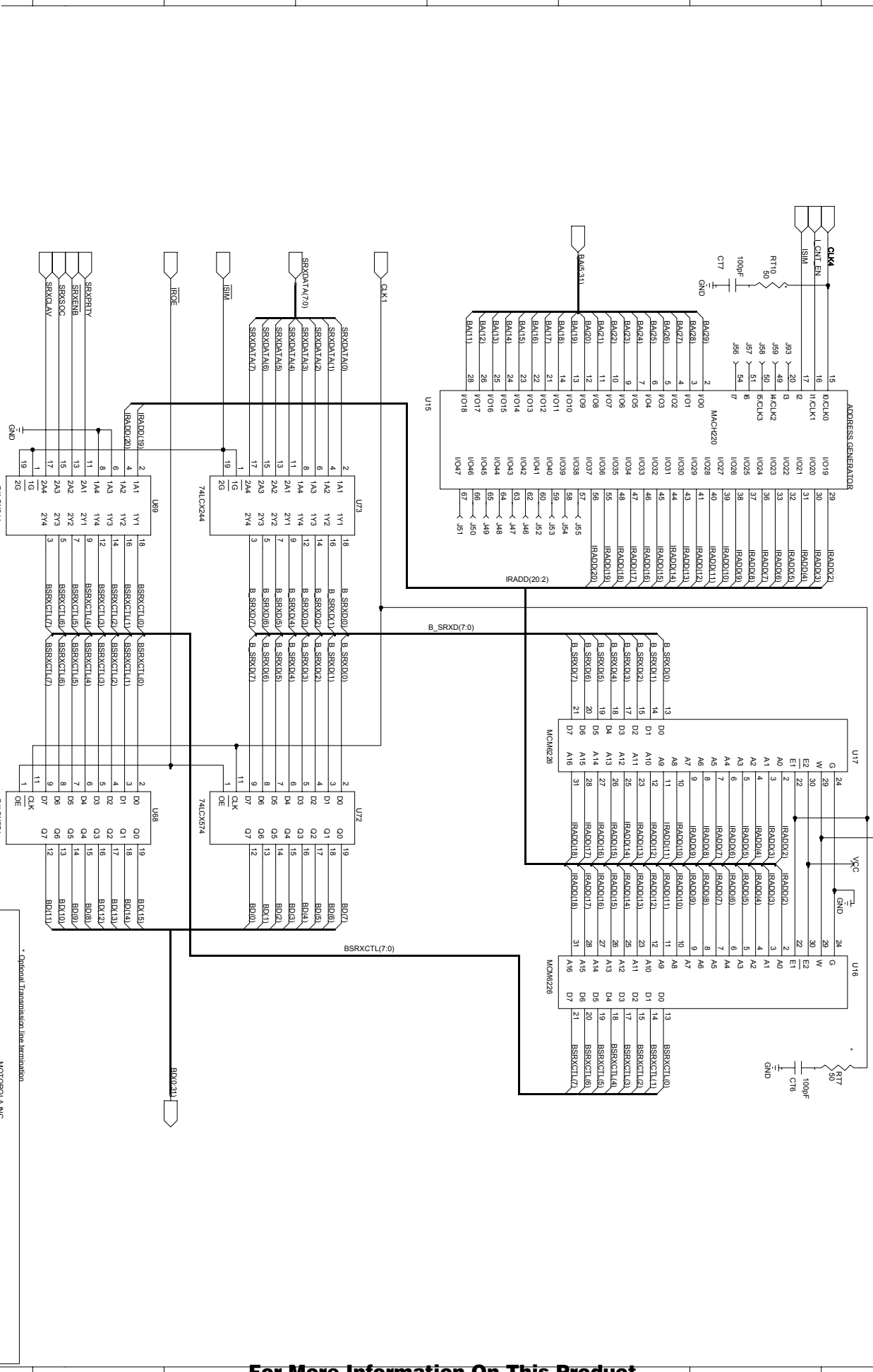
RS232 PORT

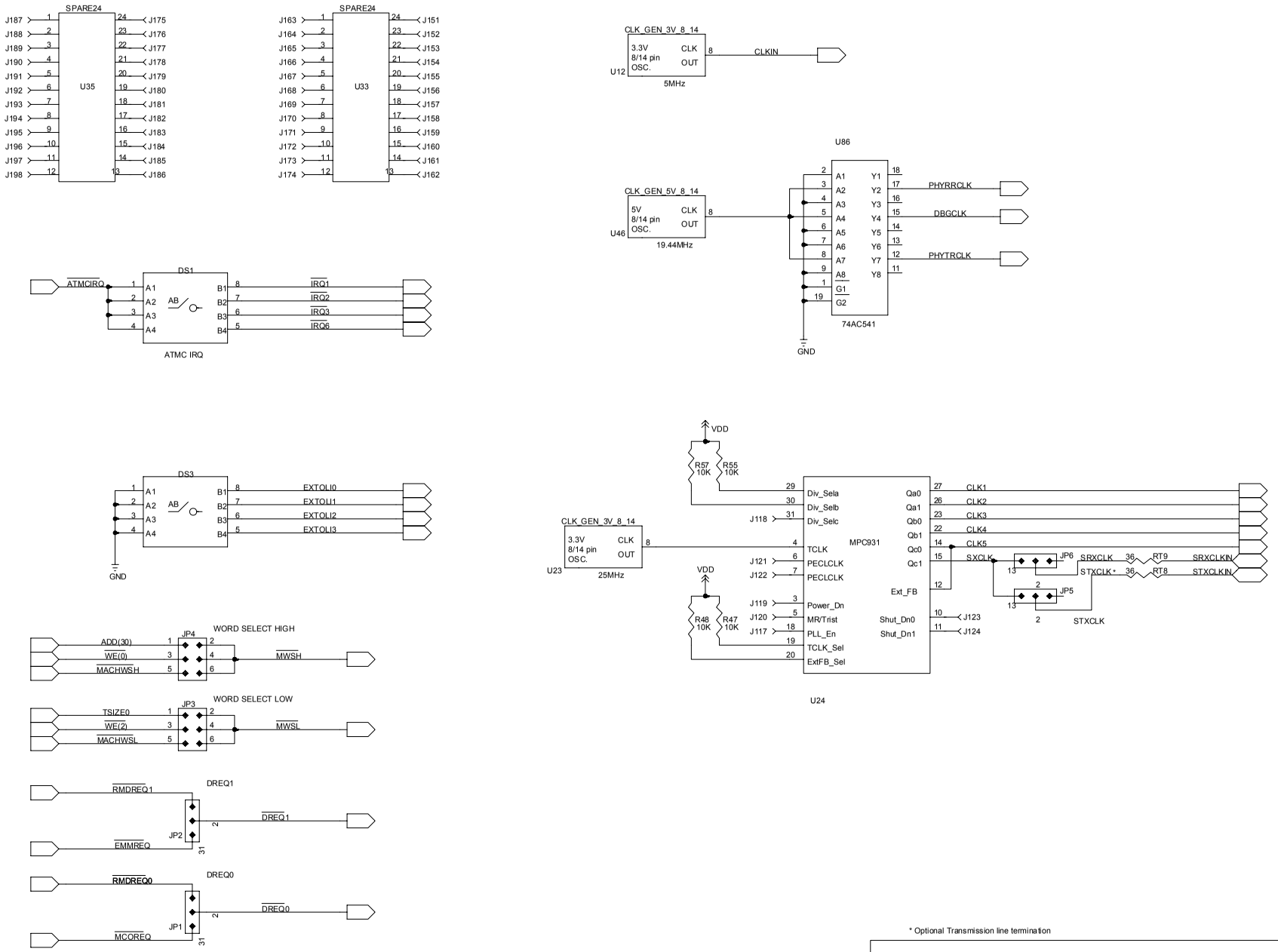
MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 9	OF 2 1
ENG :	BLOCK :		
CHK :	DESCRIP : RS232 PORTS		







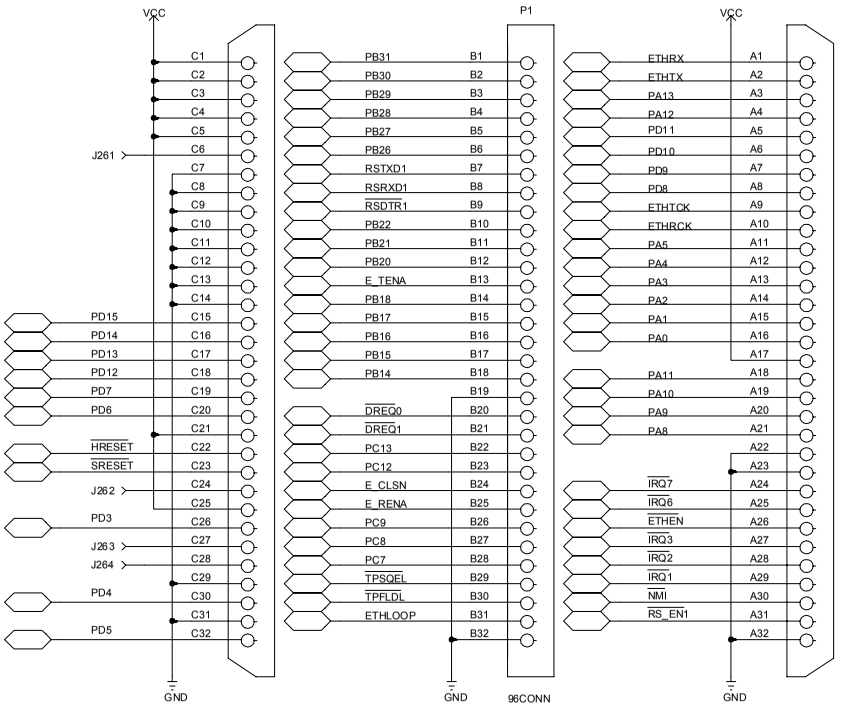




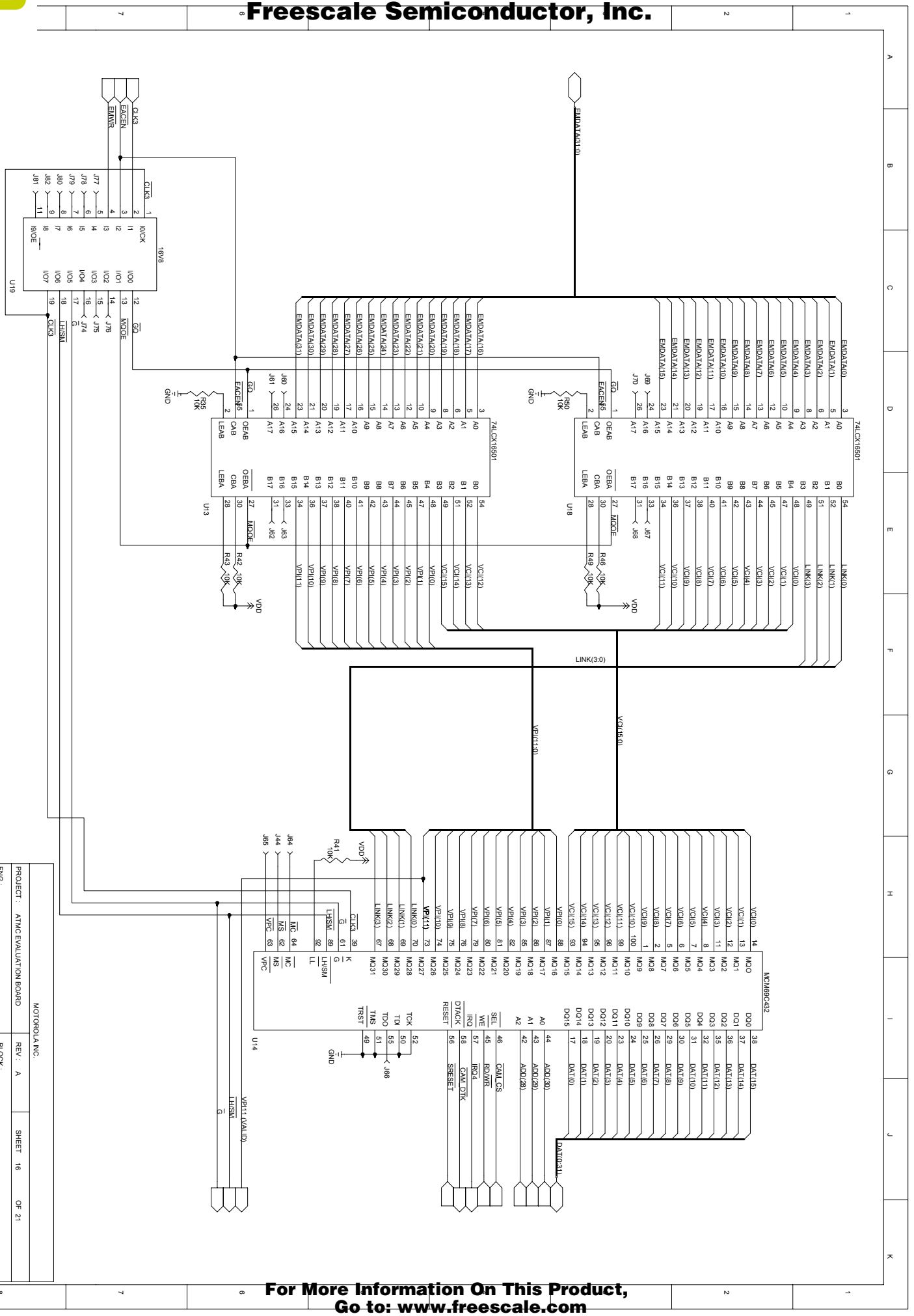
* Optional Transmission line termination

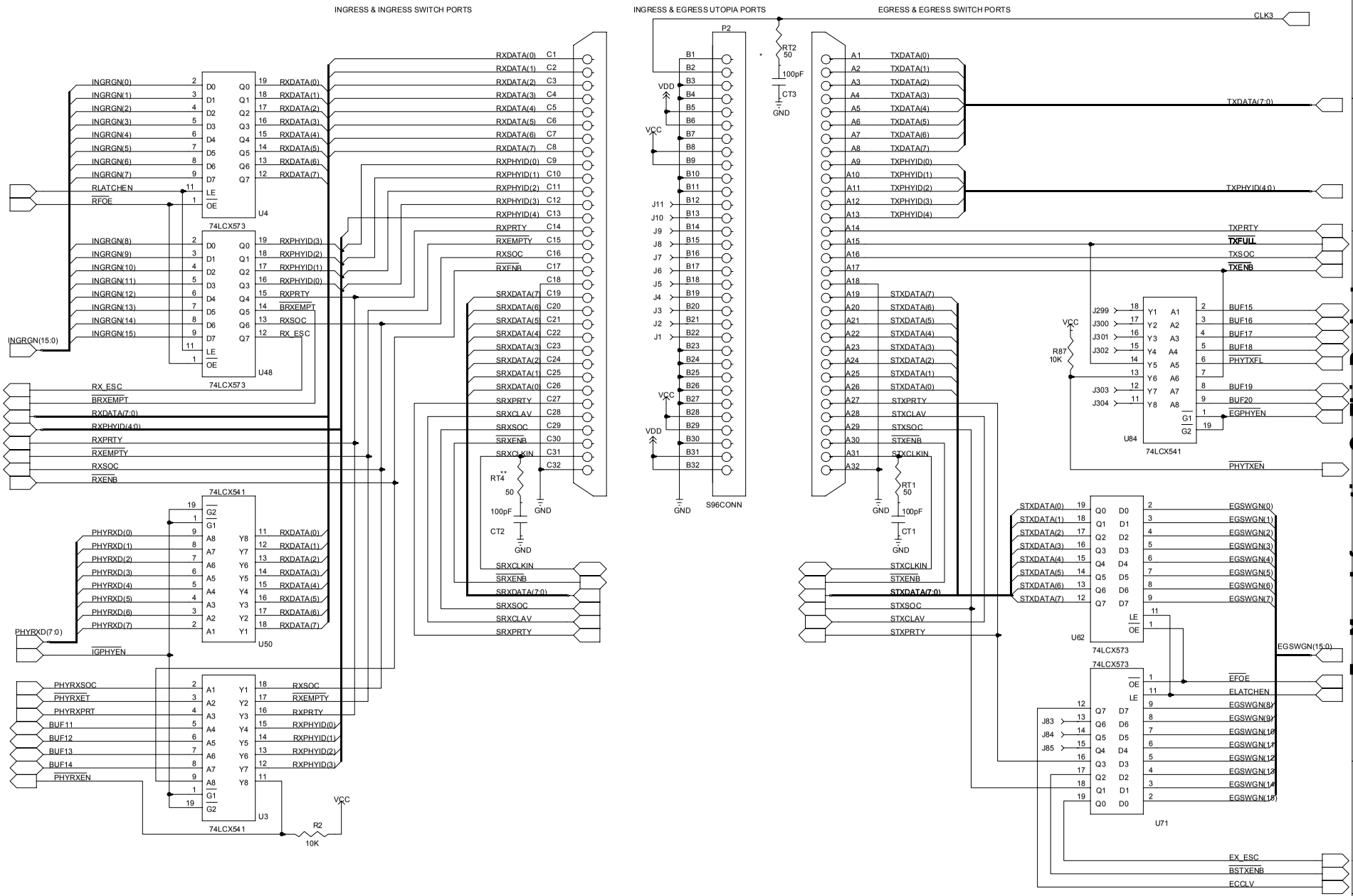
MOTOROLA INC.		
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 14 OF 21
ENG :	BLOCK :	
CHK :	DESCRIP : CLOCK GENERATORS, IRQ & ATMC SETTINGS	

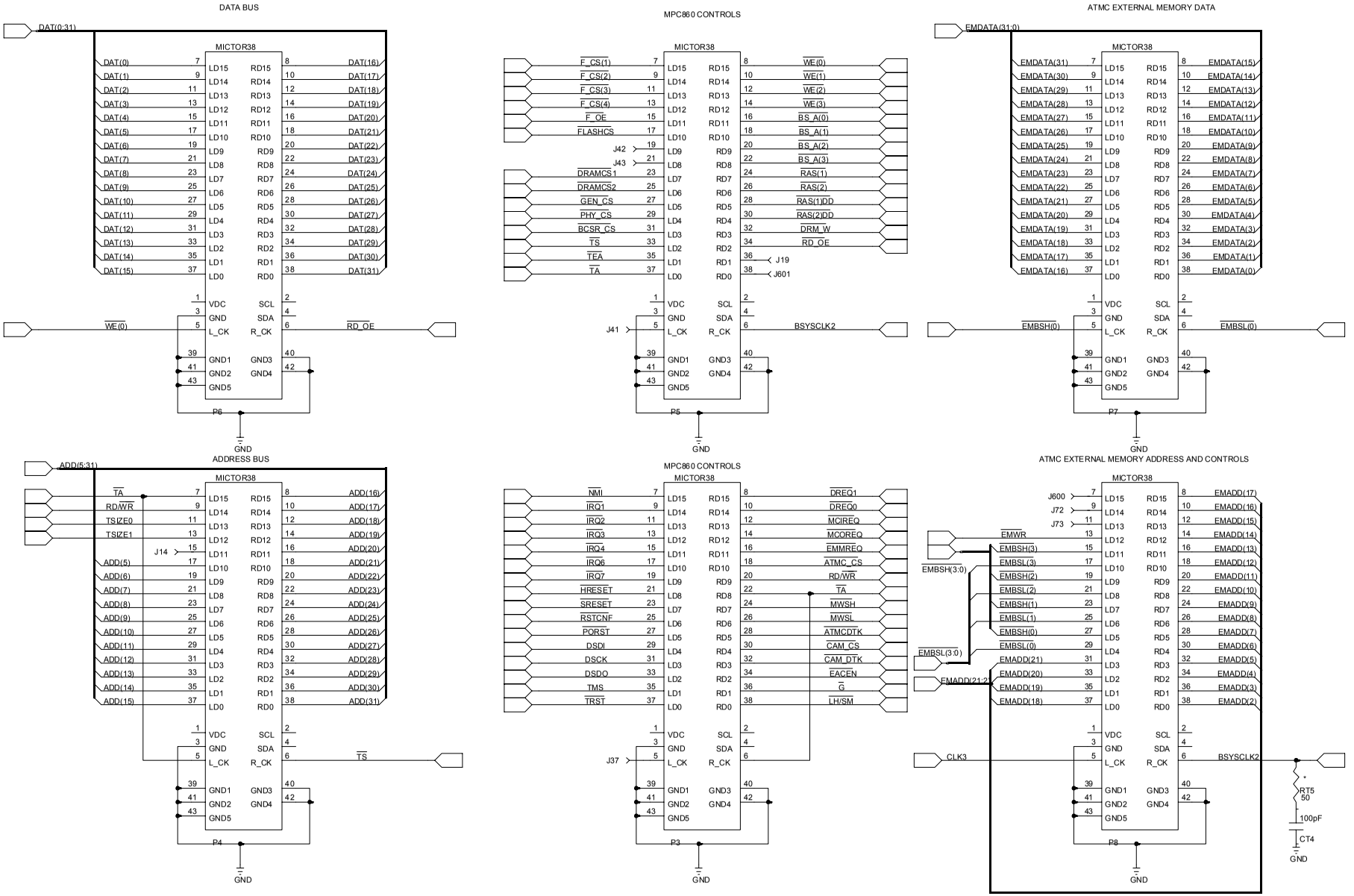
MPC860 PARALLEL I/O PORTS EXPANSION CONNECTOR



MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 15 OF 21	
ENG :	BLOCK :		
CHK :	DESCRIP : MPC860 I/O PORTS EXPANSION CONNECTOR		

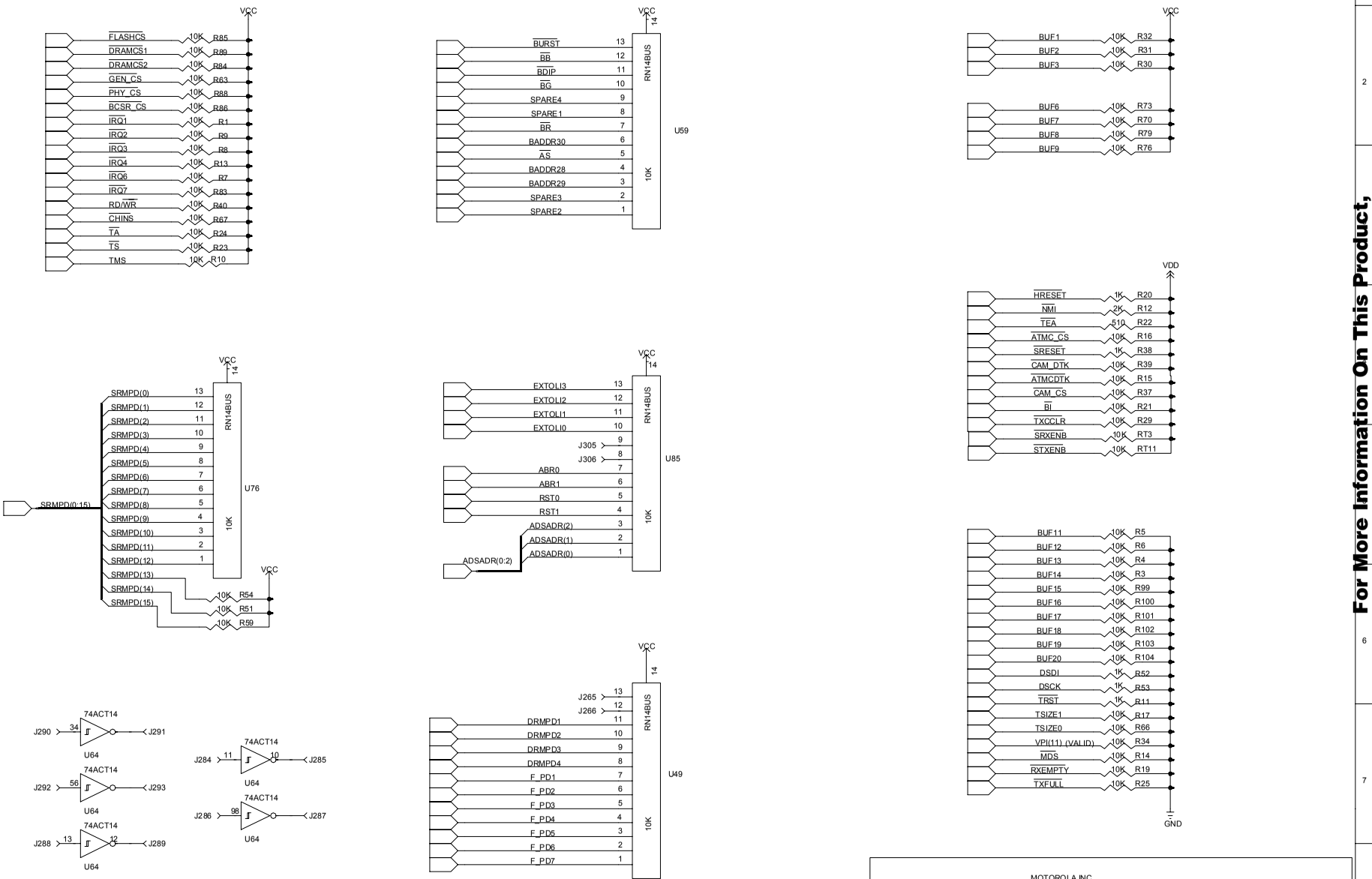




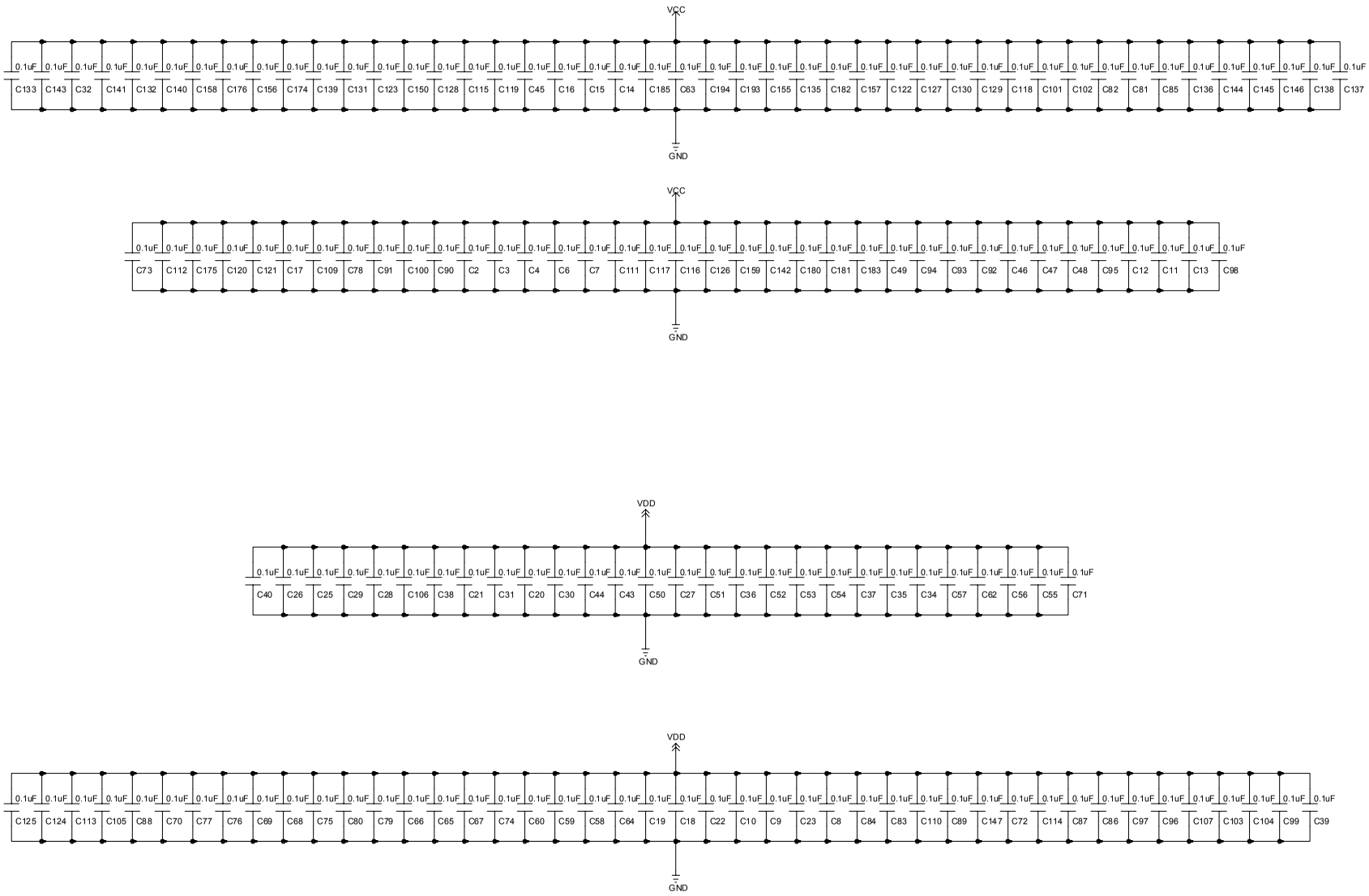


* Optional Transmition line termination

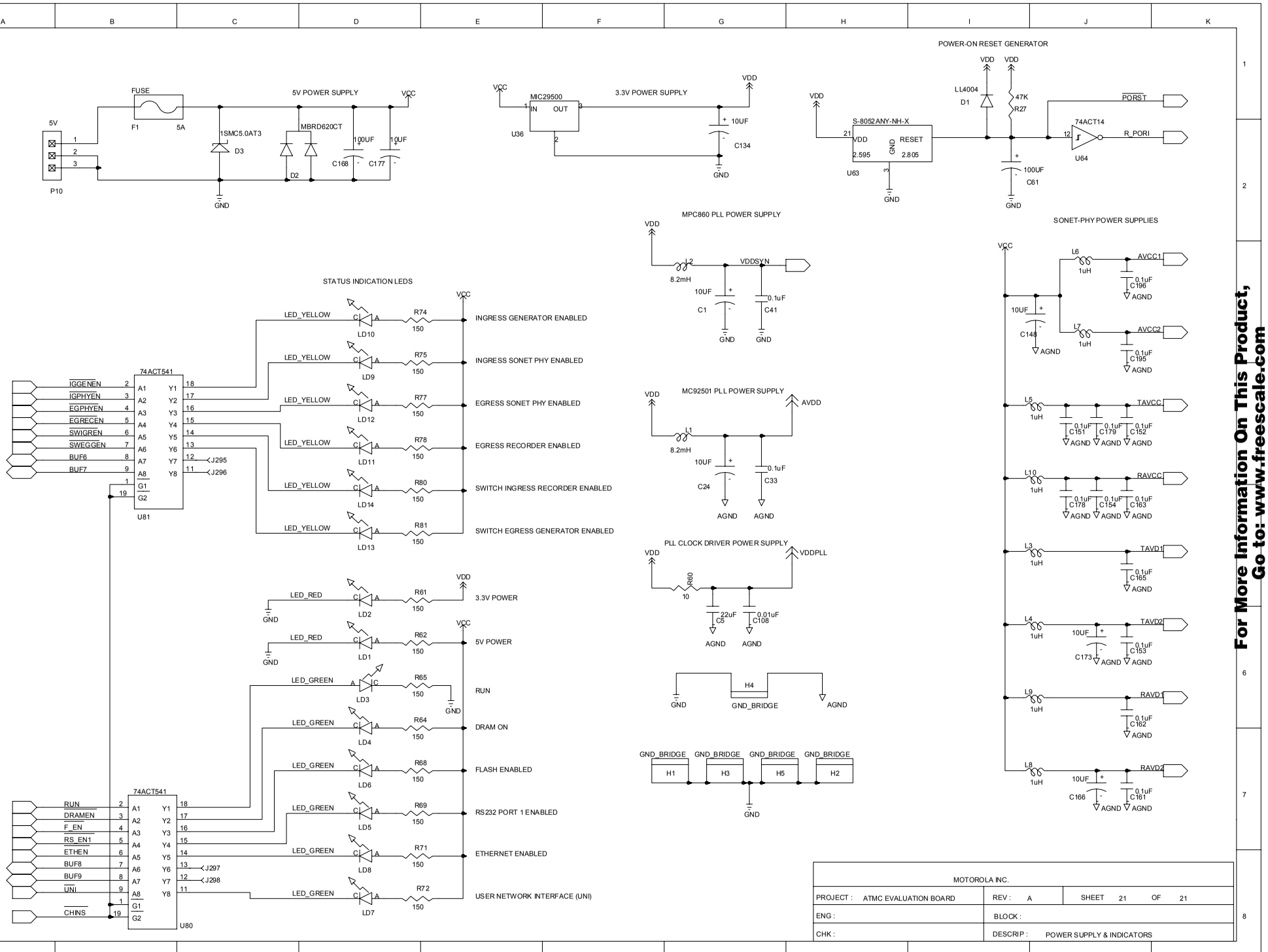
MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 18	OF 21
ENG :	BLOCK :		
CHK :	DESCRIP : HP DIGITAL ANALYZER ADAPTORS		



MOTOROLA INC.			
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 19 OF 21	
ENG :	BLOCK :		
CHK :	DESCRIP : PULL-UP & PULL-DOWN RESISTORS		

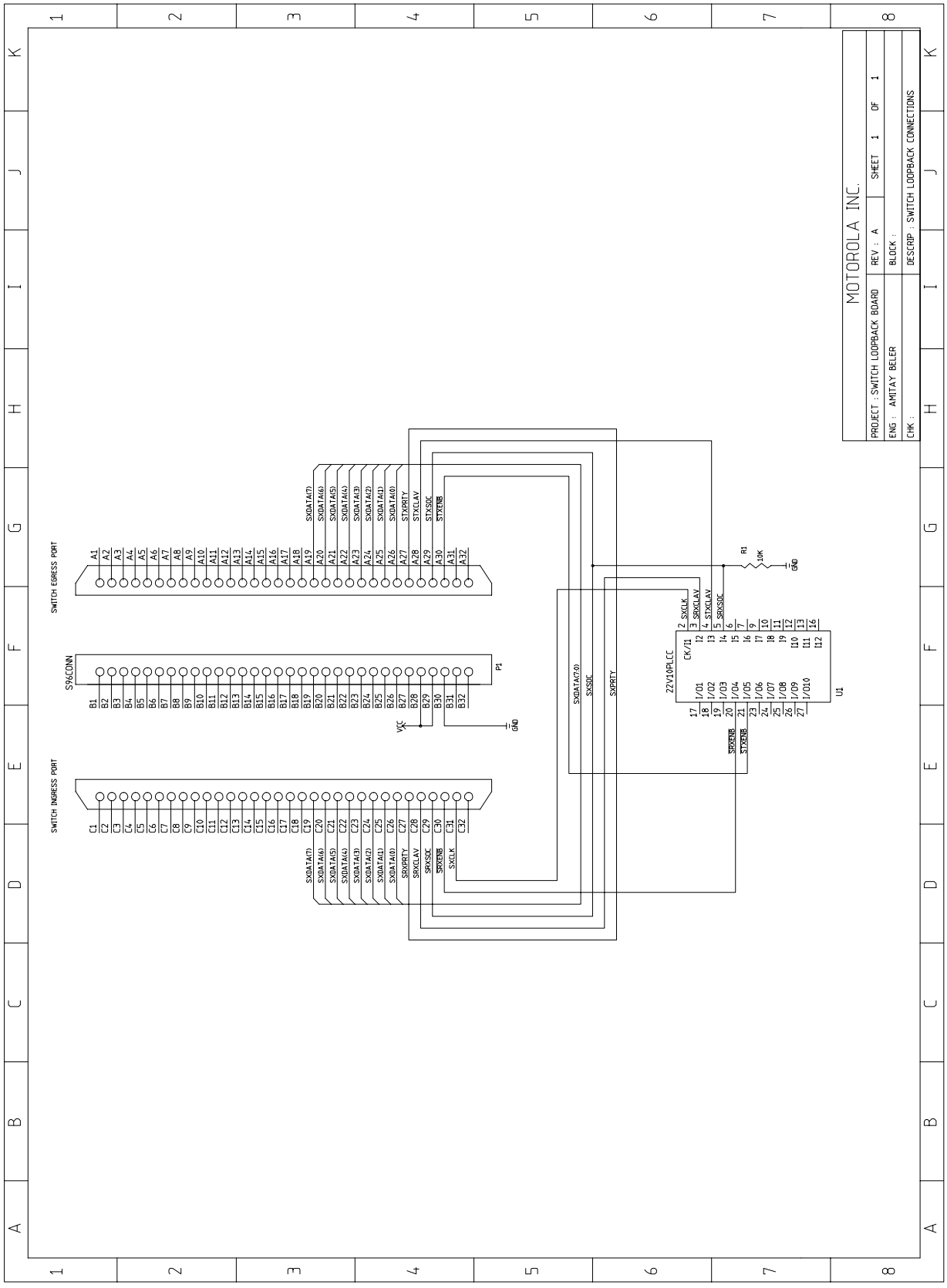


MOTOROLA INC.		
PROJECT : ATMC EVALUATION BOARD	REV : A	SHEET 20 OF 21
ENG :	BLOCK :	
CHK :	DESCRIP : DECCOUPLING CAPACITORS	



E

Loopback Connector Schematics



Index

C

C

D

E

F

G

H

I

J

R

S