ARM Development Board
ARM7TDMI Version

Hardware Reference Guide

Document number: ARM DUI 0017C
Issued: March 1997

ENGLAND
ARM
90 Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN
UK
Telephone: +44 1223 400400
Facsimile: +44 1223 400410
Email: info@armltd.co.uk

GERMANY
ARM
Otto-Hahn Str. 13b
85521 Ottobrunn-Riemerling
Munich
Germany
Telephone: +49 89 608 75545
Facsimile: +49 89 608 75599
Email: info@armltd.co.uk

JAPAN
ARM
KSP West Bldg, 3F 300D, 3-2-1 Sakado
Takatsu-ku, Kawasaki-shi
Kanagawa
213 Japan
Telephone: +81 44 850 1301
Facsimile: +81 44 850 1308
Email: info@armltd.co.uk

USA
ARM
Suite 5
985 University Avenue
Los Gatos
CA 95030 USA
Telephone: +1 408 399 5199
Facsimile: +1 408 399 8854
Email: info@arm.com

World Wide Web address: http://www.arm.com

Proprietary Notice

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Ltd shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Trademarks

ARM, and the ARM Powered logo are registered trademarks of ARM Ltd.
EmbeddedICE is a trademark of ARM Limited.

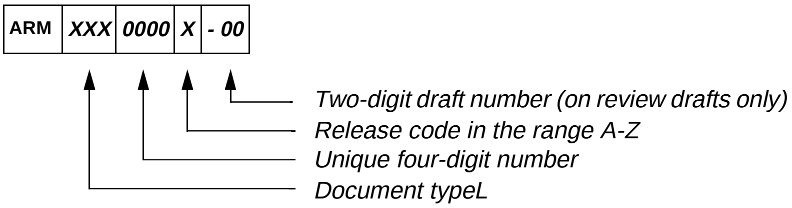
Windows 95 is a registered trademark of Microsoft Corporation.

Windows NT is a trademark of Microsoft Corporation.

Key

Document Number

This document has a number which identifies it uniquely. It is displayed on every page.


ARM
XXX
0000
X
- 00
Two-digit draft number (on review drafts only)
Release code in the range A-Z
Unique four-digit number
Document typeL

Document Status

This describes the document’s confidentiality and information status, and is shown at the bottom of each page.

Confidentiality status is one of:

ARM Confidential	Distributable to ARM staff and NDA signatories only
Named Partner Confidential	Distributable to the above and to the staff of named partner companies only
Partner Confidential	Distributable within ARM and to staff of all partner companies
Open Access	No restriction on distribution

Information status is one of:

Advance	Information on a potential product
Preliminary	Current information on a product under development
Final	Complete information on a developed product

Change Log

Issue	Date	By	Change
A	June 96	KTB	Created
B	Nov 96	LG	Updated
C	March 97	BJH	Updated to reflect creation of TDS User Guide

Contents

Contents

1 Introduction

This manual provides hardware reference information on the ARM Development Board.

For information on connecting the board to a host computer and using the software development tools, please refer to the companion manual *Target Development System User Guide (ARM DUI 0061)*.

Introduction

1.1 Using this Manual

Chapter 1	is this introduction
Chapter 2	introduces the ARM Development Card
Chapter 3	describes the circuits of the ARM Development Card
Chapter 4	describes how to expand the ASB
Chapter 5	describes how to expand the APB
Chapter 6	describes the EmbeddedICE interface
Chapter 7	describes the logic analyser interface
Chapter 8	describes the test interface
Chapter 9	describes how to program the APB FPGA
Chapter 10	describes how to program the MACH and PAL devices
Appendix A	provides detailed circuit schematics of the board
Appendix B	provides detailed circuit schematics of the daughter board
Appendix C	is an index of the programmable devices
Appendix D	is a summary of the switches, jumpers and links
Appendix E	is a mechanical drawing of the ARM Development Card

1.1.1 Related Documentation

You may find it useful to refer to the following documents:

ARM IHI-0001	AMBA Specification
ARM DUI 0014	HP ARM Inverse Assembler User Guide
ARM DDI-0041	AMBA Arbiter
ARM DDI-0042	AMBA Decoder
ARM DDI-0043	AMBA Test Interface Controller
ARM DDI-0047	AMBA Interrupt Controller
ARM DDI-0048	AMBA Reset and Pause
ARM DDI-0049	AMBA Timer APB Peripheral
ARM DDI-0051	AMBA Reset Controller
ARM DUI 0061	Target Development System user Guide
ARM DDI 0062	Reference Peripherals Specification

1.2 Conventions

This manual employs typographic conventions intended to improve its ease of use.

`code`	code which you need to enter, or which is provided as an example

1.3 Useful Contacts

1.3.1 Contacting ARM

Further information is available from ARM.

All schematics (ORCAD), PLD and VHDL binary files and latest release notes are available from our world wide web servers at:

`http://www.arm.com`

If you require PDL descriptions or have difficulty accessing our web page, please email:

`upgrades@arm.com`

1.3.2 Component data sheets

Contact points for component data sheets are as follows:

XR16C552 Exar (Startech) serial and parallel port chip

UK distributor:

Farnell Electronic Components Ltd.	Tel: +44 113 2310160
	`http://www.exar.com`

MACH and PALCE AMD programmable logic devices

UK distributors:

Kudos Thame Ltd.	Tel: +44 1734 351010
Avnet Access Ltd.	Tel: +44 1462 480888
	`http://www.amd.com`

XC4005 Xilinx FPGA

UK distributor:

Microcall Ltd.	Tel: +44 1844 261939
Avnet Access Ltd.	Tel: +44 1462 480888
	`http://www.xilinx.com`

VG-468 Vadem PC card controller

UK distributor:

MMD	Tel: +44 1734 633700
	`http://www.vadem.com`

1.3.3 Information on chips

A useful site for chip information is:

`http://www.xs4all.nl/~ganswijk/chipdir`

1.4 Glossary

Some of the terms used in this manual may be unfamiliar to you. This section explains some of the more important ones.

ARM7TDMI The ARM7TDMI test chip is an example of an ARM processor macrocell that is suitable for use on the ARM Development Card. See the *ARM7TDMI Data Sheet* (ARM DDI 0029) for more information.

CPLD A *complex programmable logic device* (*CPLD*) is usually a collection of PAL-type devices in a single package. The AMD MACH device is an example of a CPLD.

EmbeddedICE This is the additional hardware that is provided by debuggable ARM processors to aid debugging. The EmbeddedICE macrocell is fully described in the *ARM7TDMI Data Sheet* (ARM DDI 0029). The EmbeddedICE macrocell is controlled via the JTAG test access port, using an EmbeddedICE interface. This is an extra piece of hardware that allows software tools to debug code running on a target processor.

FPGA A *field-programmable gate array* (*FPGA*) is a type of *programmable logic device* (*PLD*). The ARM Development Card is fitted with one FPGA manufactured by Xilinx. You can change the functionality of this device if the appropriate design tools are available. Xilinx sells an appropriate tool set which interfaces to a variety of front-end systems which may be based on schematics or hardware description languages such as VHDL. See also *LCA*.

ICE An *in-circuit emulator* (*ICE*), is a device that aids debugging of hardware and software. ARM debuggable processors such as the ARM7TDMI have extra hardware called *EmbeddedICE* to assist this process.

JTAG This is a serial-like test port provided on many large silicon chips such as the ARM7TDMI.

LCA A *logic cell array* (*LCA*) is a type of *programmable logic device* (*PLD*) also known as a *field-programmable gate array* (*FPGA*).

MACH A MACH device is a example of a *complex programmable logic device* (*CPLD*). The ARM Development Card uses a number of MACH210 and MACH230 devices. Based on electrically erasable (*EE*) technology, they are reprogrammable. Using appropriate software (such as PALASM), the function of these devices may be changed by reprogramming in a standard programmer.

NISA *NISA* (*not-ISA*) is ARM's description of the bus that connects the *Advanced System Bus* (*ASB*) to some standard peripheral devices such as the serial/parallel ports and PC card controller. It is a subset of the *Industry Standard Architecture* (*ISA*) bus found in most IBM compatible PCs.

PAL

A *programmable array logic* (*PAL*) device is a example of a *programmable logic device* (*PLD*). The PAL used on the ARM Development Card is a PALCE22V10. This has up to 22 inputs, ten outputs and ten programmable macrocells. As it is based on electrically erasable (EE) technology, it is reprogrammable. Using appropriate software (such as PALASM), the function of this device may be changed by reprogramming in a standard programmer.

PCMCIA

The *Personal Computer Memory Card Association* (*PCMCIA*) produces a specification that details an interface suitable for connecting small boards (the size of credit cards) to larger host systems. The name PCMCIA is generally used to describe these cards, but its use has been superseded by the term PC card.

PLD

A *programmable logic device*. See also PAL and FPGA.

PALASM

A *programmable array logic assembler* (*PALASM*) is a low-cost, proprietary logic description language produced by *Advanced Micro Devices* (*AMD*) for their range of PLDs and CPLDs. It has been used extensively in the design of the ARM Development Card.

PLL

A *phase-locked loop* (*PLL*) usually comprises a voltage controller oscillator, programmable divider, frequency comparator, and an integrator. These components allow a programmable frequency clock to be generated. This is locked to and stabilised by a reference clock input.

On the ARM Development Card, a single component performs this function. A reference crystal at 14.318MHz is used, and with three programmable inputs, the device is able to generate 8 output frequencies from about 4–50MHz.

VHDL

VHDL is a hardware description language suitable for the simulation and synthesis of logic circuits. The design for the FPGA on the ARM Development Card was completed using VHDL and synthesis tools from Compass. Xilinx tools were used to place and route the design.

2 Board Overview

This chapter describes each of the main blocks of the ARM Development Board.

Board Overview

2.1 Overview of the ARM Development Board

The ARM Development Board is a platform that is suitable for code development and exploration of embedded ARM processors. It is a convenient means of evaluating the Advanced RISC Machines' Thumb-aware (ARM7T) family of RISC processors.

The ARM Development Board has been designed to conform to the *Advanced Microcontroller Bus Architecture* (*AMBA*) specification. This specification defines an on-chip communications standard for designing high performance 32- and 16-bit embedded microcontrollers. A convenient way to view the ARM Development Board is as a microcontroller design in discrete components. This means that it is possible to observe bus transactions and peripheral accesses using standard test equipment. Thus, a typical microcontroller design can be easily observed and prototyped.

Because the processor in the system is little more than ARM core it is possible to use an in-circuit emulator (ICE). This enables a system design to be tested and debugged at the processor level. In addition processors with EmbeddedICE capabilities can be debugged directly using the EmbeddedICE interface. The ARM Development Board also has a parallel port and two serial ports that allow it to be connected to a variety of hosts. Using a monitor program supplied with the board, the user can download and run code in collaboration with the ARM Software Development Toolkit.

The ARM Development Board shows how to design a system based on the AMBA specification, comprising a multi-master system bus (ASB) and a low-power peripheral bus (APB). While on-chip techniques may differ, the main system modules and their interconnect have been preserved.

The following are useful reference documents. You should refer to these to understand the functionality of AMBA modules.

- *AMBA Specification(ARM IHI 0001)*
- *Reference Peripherals Specification(ARM DDI 0062)*

2.1.1 Using ARM resources in your design

This manual contains both the circuit description (and schematics) of the ARM Development Board and a description of programmable logic devices used.

The programmable logic equations and schematics can be obtained from ARM for use in your own designs. These are provided to help you design your prototype target hardware systems.

Both hardware and software are provided as tutorial aids and demonstrate techniques rather than an optimal implementation. Please feel free to use the schematics and programmable logic equations provided as a basis for your own system designs.

2.2 An Overview of the Board

A typical AMBA system comprises a processor connected to an *Advanced System Bus* (*ASB*) with a bridge to the slower, low-power *Advanced Peripheral Bus* (*APB*). The main system blocks are shown in ***Figure 2-1: Overview of the ARM Development Board*** on page 2-4:

- AMBA bus master comprising an ARM processor and PLD
- AMBA system modules, arbiter and decoder
- On-chip (synchronous SRAM) memory
- SRAM block
- EPROM or FLASH block
- DRAM block
- Test interface
- APB bridge
- APB slaves, timer, interrupt controller
- ASB expansion connectors
- APB expansion connectors
- NISA bus bridge
- PC card (PCMCIA) block
- Serial and parallel port block

2.2.1 Board architecture

A convenient way to view the ARM Development Board is as a sample microcontroller with its support peripherals constructed from discrete devices. The bus master, system modules, APB bridge and peripherals, on-chip RAM and external bus interfaces form the heart of a microcontroller. Additional peripherals such as PC card (PCMCIA) and serial and parallel ports may also be incorporated or interfaced to externally.

Each functional block is constructed from separate programmable logic devices (PLDs). This enables you to observe the system interactions using standard test equipment such as a logic analyser. The expansion connectors provide a way of interfacing additional circuitry to the ARM Development Board and also provide convenient hook-up points for a logic analyser.

Refer to ***Chapter 4, Expanding and Monitoring the ASB*** and ***Chapter 5, Expanding and Monitoring the APB*** for further information.

Memory types

A typical system might provide some of the following memory types:

- SRAM
- EPROM
- FLASH
- DRAM

Examples of all of these can be found on the board.

Each memory type has its own controller. An ideal system might have a single external bus interface (EBI). In this implementation the EBI is distributed into separate memory controllers.

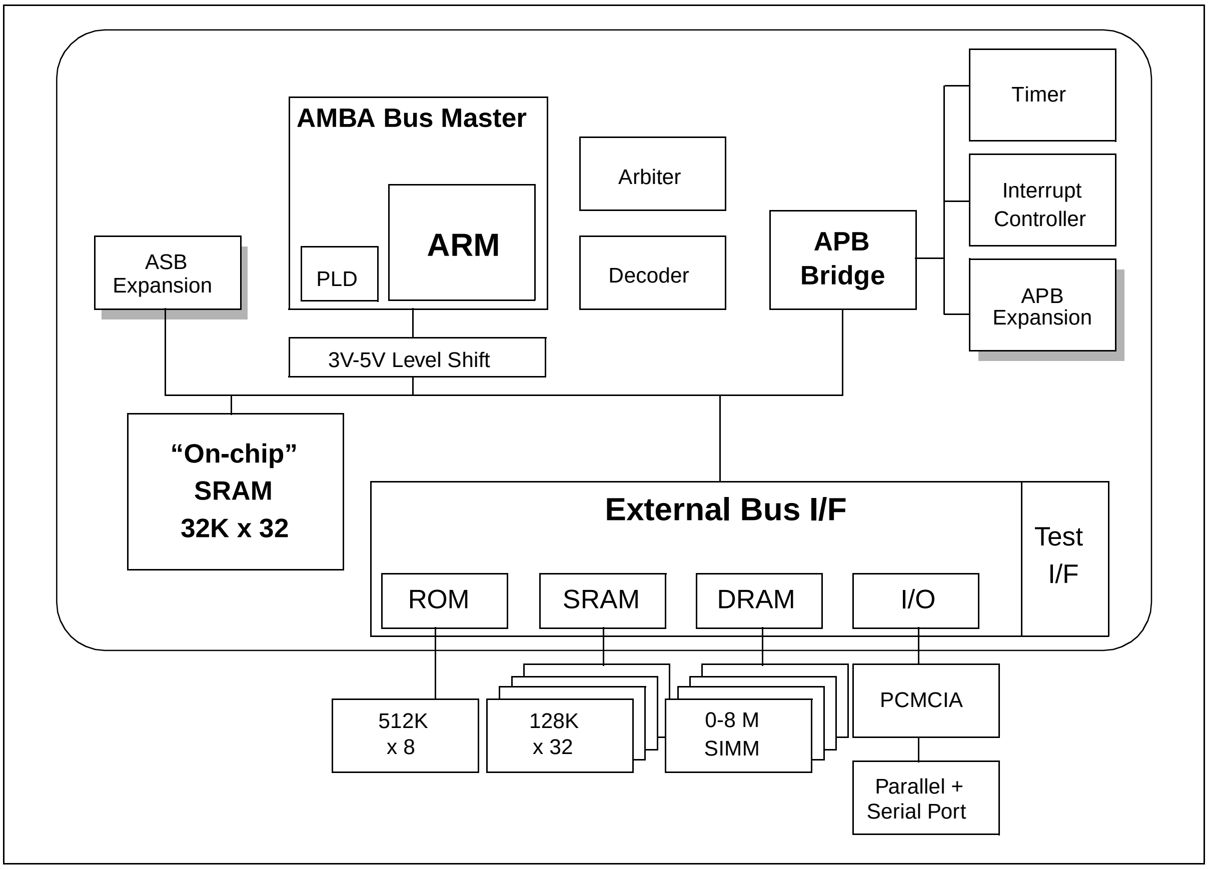


Figure 2-1: Overview of the ARM Development Board

3 Circuit Descriptions

This chapter describes the circuits of the ARM Development Board.

Circuit Descriptions

3.1 Overview of Schematics

The board has been designed to allow an AMBA bus master (such as an ARM7TDMI test chip) to be mounted on daughter board. The daughter board is an integral part of the ARM Development Board, although the daughter board supplied could be replaced with another AMBA master, such as an in-circuit emulator.

3.1.1 Master board circuits

The master board design comprises 22 schematics as listed below.

1	Board outline drawing	DRAWING.SCH
2	Top-level diagram	CHAMP.SCH
3	Power supply	POWER.SCH
4	Crystal oscillator and clock distribution	OSC.SCH
5	ASB slaves	ASBSLAVE.SCH
6	“On-chip” memory (synchronous SRAM)	ONCHIP.SCH
7	EPROM/FLASH ASB slave	EPROM.SCH
8	DRAM ASB slave	DRAM.SCH
9	SRAM ASB slave	SRAM.SCH
10	APB and NISA bridge	ASBNISA.SCH
11	NISA bus peripherals	NISABUS.SCH
12	Serial and parallel ports	SUPERIO.SCH
13	PC card interface	PCMCIA.SCH
14	PC card connectors and power supply	CARDCON.SCH
15	APB slaves	APBSLAVE.SCH
16	APB expansion connectors	APBEXP.SCH
17	APB buffers	APBBUF.SCH
18	Memory address and data buffers	MEMBUF.SCH
19	Test interface controller and connectors	TIC.SCH
20	Master header connectors and level convertors	MASTER.SCH
21	System modules (arbiter and decoder)	SYSMODS.SCH
22	ASB expansion connector	ASBEXP.SCH

3.1.2 Configuring the board

The board is configurable through the use of links, jumpers and switches. Each of these is described in detail in the following subsections. In addition, there is a summary of links and switches in ***Appendix D, Summary of Jumpers and Links***. Also, the *Target Development System User Guide (ARM DUI 0061)* contains information on configuring the ARM Development Board.

3.1.3 Daughter board circuits

The daughter board schematics for the supplied system are included in ***Appendix B, Daughter Board Schematics***.

The daughter board (or header) is connected to the ARM Development Board by four 60-way connectors. This allows different bus masters to be connected, including in-circuit emulators.

The design comprises seven schematics as listed below.

Note *There are two versions of the processor schematic depending upon whether you have a QFP or PGA packaged part on the board.*

1	Board outline drawing	DRAWING.SCH
2	Top-level diagram	CHAMPQFP.SCH
3	Header connectors	CPUHEAD.SCH
4	Logic analyser connectors	LAPODS.SCH
5	AMBA bus master veneer	AMBAPLD.SCH
6	Processor in QFP package	PROCQFP.SCH
7	Processor in PGA package	PROCPGA.SCH
8	EmbeddedICE interface	EICE.SCH

3.2 ARM Development Board

The top-level schematic is illustrated in ***A.1 Card Outline Drawing*** on page A-2, and shows the main blocks of the design. The blocks are interconnected by the ASB signals prefixed **B_**, such as **B_A[31:0]**, **B_D[31:0]** and **B_WAIT**. There are two ASB bus masters, the ARM chip mounted on a header card and the test interface controller (TIC).

The ASB system modules (the system arbiter and decoder) can also be seen.

There is a block called ASB expansion which details the physical connectors. This allows the ASB to be monitored by a logic analyser, or allows external circuitry to be attached

The oscillator block describes the system clock generation and distribution, and the power supply block describes the power input and regulation.

There are a number of ASB slaves which are described in ***3.2.3 ASB Slaves*** on page 3-7.

3.2.1 Power Supply

This schematic is shown in ***A.3 Power Supply*** on page A-4.

Two green LEDs marked (5V) and (3V3) light up when power is connected to the board.

Note *Take care when connecting up power to this board as there is no protection for incorrectly wired supplies. If the LEDs fail to light, switch off immediately and check the connections.*

The board is designed to function at 5V so that high-speed programmable logic devices can be used. The ARM processor is a 3.3V component and so needs to be protected from high logic levels. This is accomplished through use of level-convertor ICs. A 3.3V supply is generated on board from a 5V supply for use by the ARM processor and the synchronous SRAM (a 3.3V part with 5V tolerant I/O).

Power to the board is supplied through a PC-style 12-way connector. This allows a PC power supply to be connected directly, and this will provide all the requirements of the board. The board consumes 2–5A at 5V depending upon the amount of DRAM fitted and the clock frequency used. If preferred a bench power supply can be used instead.

Note *Some PC power supplies can trip out if very low current is taken, so you can insert a load across the +12V supply has been made. If this is a requirement, connect a resistance across the pads marked (JP2).*

The connector (J1) contacts are rated at 2A, so if current consumption is low, it is only necessary to connect to pin 2 (+5V) and pin 5 (GND). Pin 3 (+12V) need only be connected if the PC card (PCMCIA) interface is to be used.

An LM317 voltage regulator (U1) is used to generate the 3.3V supply required by the ARM processor. You can decouple this from the regulator by removing a wire link (JP1) if required.

3.2.2 Crystal Oscillator and Clock Distribution

This schematic is shown in ***A.4 Crystal Oscillator and Clock Distribution*** on page A-5.

A standard 14.318MHz crystal oscillator (X1) is connected to a phase-locked loop (PLL) based clock-generation device (U5), which generates all the system clocks. These are:

SYSCLK	distributed to system
SYSCLK2X	double rate system clock
CLK32MHZ	32MHz clock for NISA bus (primary)
CLK24MHZ	24MHz clock for NISA bus (alternative to **CLK32MHZ**)
COMMCLK	1.843MHz for serial port UART

System clocks

The system clocks, either on-board or external, are distributed to the rest of the board via three low-skew clock buffer devices (U3,U4 and U6). Each clock output is serially terminated and drives one load only.

If other system clock frequencies are needed (up to a maximum of 25MHz), an external source can be applied in place of the **SYSCLK** and **SYSCLK2X** outputs from (U5). Both must be provided and they must be phase-aligned.

External clocks

The external clocks, **EXTSCLK** and **EXTSCLK2X**, should be connected to plugs (PL1 and PL2). Optional 47R resistors can be fitted (R36 and R37) to terminate the clock inputs if required. To select external clocks instead of the on-board clocks, you have to move the surface mount links (LK2 and LK3) to the B-C position.

Double-rate clock

The double-rate clock **SYSCLK2X** is used by the synchronous SRAM to allow single cycle memory accesses. This is a departure from the AMBA bus methodology, but used in order to simulate fast “on-chip” memory.

Serial port (UART)

The 1.843MHz COMMCLK drives the serial port (UART) baud rate generator directly.
In addition, this clock is divided down by the PAL (U2) to provide a refresh signal (**REFCLK**) at 64kHz for the DRAM controller.

NISA bus devices

The **NISACLK** is used to drive the NISA bus devices. This is a 32MHz clock signal, derived from **CLK32MHZ**, selected by a surface mount link (LK1). The **CLK24MHZ** output from (U5) is not normally used.

Circuit Descriptions

Unused clock signals

Two clock signals are not used **NISACLK1** and **B_CLK8**. These are available as monitor points (V1 and V2), as shown in ***A.2 Top-level Diagram*** on page A-3.

Clock frequencies

The frequency of **SYSCLK** and **SYSCLK2X** can be controlled by three inputs (**CLKSEL[2:0]**). A switch (S1) is used to control these select lines via a PAL (U2), which is used to prevent inappropriate clock frequencies being selected.

Ref	Position	Name	Option	Description
S1	1	SEL0	on/off	see table below
	2	SEL1	on/off	see table below
	3	SEL2	on/off	see table below
	4	SEL3	on/off	see table below

Table 3-1: S1

	Switch position			Frequency (MHz	
SEL3	**SEL2**	**SEL1**	**SEL0**	**SYSCLK**	**SYSCLK2X**
on	on	on	on	4	8
on	on	on	off	8	16
on	on	off	on	16	32
on	on	off	off	20	40

Table 3-2: Switch positions

3.2.3 ASB Slaves

This schematic is shown in ***A.5 ASB Slaves*** on page A-6, and shows the following ASB slaves and associated circuitry:

- DRAM controller
- Synchronous SRAM controller
- SRAM controller
- EPROM/FLASH controller
- Memory address and data buffers
- APB and NISA bus bridge

Each memory controller is described in the appropriate section. The memory address and data buffers are shared by the these controllers as follows:

- **B_D** is driven onto **M_D** when **nOEMD** is driven LOW
- **M_D** is driven onto **B_D** when **nOEBD** is driven LOW

The SRAM controller and the DRAM controller both drive **nOEMD** and **nOEBD**, which are implemented as open-collector active low signals.

M_A is generated by sampling **B_A** on the falling edge of **B_CLK**, and is used by the SRAM and EPROM slaves.

Address and data latches for the APB and NISA buses are implemented separately (see ***3.2.15 APB Buffers*** on page 3-22).

Link (LK4) is used to select the endianism of the board. The default (link out) is little-endian. If the link is inserted the **BIGEND** signal goes high and this is used by the SRAM, synchronous SRAM and DRAM controllers to enable big-endian style writes. There is no support for EPROMs that have been programmed big-endian. The **BIGEND** signal is also routed to the ARM processor.

3.2.4 “On-Chip” Memory (Synchronous SRAM)

The “on-chip” memory (synchronous SRAM) schematic is shown in ***A.6 “On-chip” Memory (Synchronous SRAM)*** on page A-7.

A typical AMBA system might comprise some fast “on-chip” memory and various memory controllers for SRAM, DRAM and EPROM. On the board, a synchronous SRAM device has been used to simulate “on-chip” memory. By running the device with a double-rate clock, single-cycle memory access can be achieved.

The controller is implemented as an ASB slave in a fast PAL (U8). The memory device is a 32K x 32-bit pipelined synchronous SRAM capable of being clocked at up to 66MHz. Because it is a pipelined device, the data is available to be read two clock cycles after the address is latched. In an ARM system, the data needs to be available in the clock cycle following the address. Therefore, by running the device at twice the system clock frequency, the data is read out in the correct system cycle.

Figure 3-1: ASB Sync SRAM Timing Diagram shows the clock relationship and control signals.

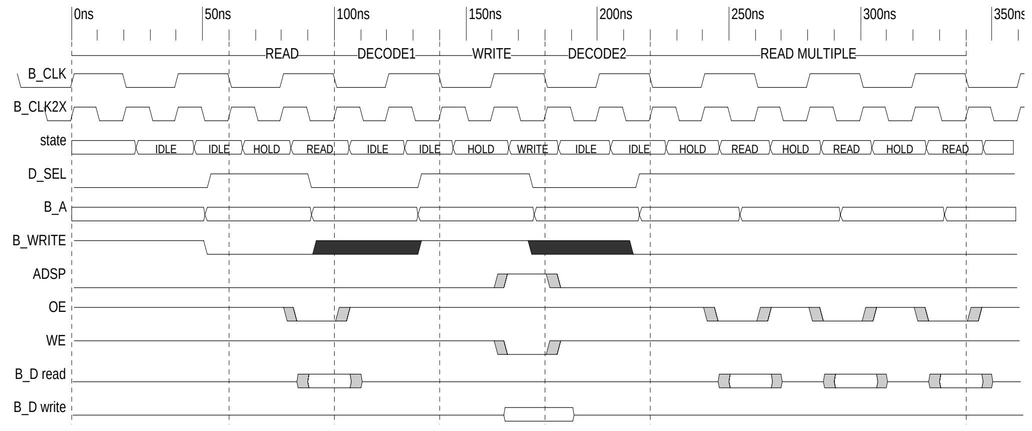


Figure 3-1: ASB Sync SRAM Timing Diagram

3.2.5 EPROM/FLASH ASB Slave

This schematic is shown in ***A.7 EPROM/FLASH ASB Slave*** on page A-8.

The EPROM/FLASH subsystem is implemented in two devices:

- one drives the memory strobes and interfaces with ASB (U10)
- the other provides data path steering (U11)

The board contains two sockets:

- an 32-pin DIL socket (U12) into which EPROM or FLASH devices up to 512Kx8 (4MB) can be fitted
- a 44-pin PLCC socket (U13) into which EPROM or FLASH devices up to 256Kx16 (4MB) can be fitted

Therefore, there is one 8-bit wide and one 16-bit wide socket, but only one device may be driven at a time. Links on the board select:

- whether an 8- or 16-bit device is driven
- whether it is EPROM or FLASH
- the number of bus cycles required to access it

Link field LK6 has the following link positions:

Position	Name	Description	Options	Default
1	CYC1	Number of cycles	see table below	out
2	CYC0	Number of cycles	see table below	in

3	EPROM	Selects EPROM or FLASH	out = EPROM
			in = FLASH
				in

4	SEL8BIT	Selects 8- or 16-bit device	out = 8-bit
			in = 16-bit
				out

Table 3-3: LK6 link positions

Clock cycles

The number of cycles is either 2, 3, 4 or 5. The cycle time must be carefully selected, taking into account the system clock frequency and the device speed grade.

For example, for a system clock frequency of 20MHz, cycle time = 50ns. ***Table 3-4: Pulse widths for settings of CYC[1:0]*** on page 3-10 shows pulse widths for various settings of **CYC[1:0]**.

Note *The write-enable strobe length (for FLASH only), is always the number of cycles minus one.*

CYC1	CYC0	Cycles	Pulse width(ns)		
			CE	OE	WE
in	in	2	100ns	100ns	50ns
in	out	3	150ns	150ns	100ns
out	in	4	200ns	200ns	150ns
out	out	5	250ns	250ns	200ns

Table 3-4: Pulse widths for settings of CYC[1:0]

For an ATMEL AT29C040A-15 (512K x 8) 5V only FLASH PEROM, tWP = 90ns, and tCE = 150ns, so select **CYC[1:0]** = [out,in] = 3cycles.

Connecting external peripherals to the EPROM/FLASH controller

As well as supporting EPROM and FLASH devices, you can use the controller to drive simple 8- or 16-bit peripherals that need memory type strobes such as:

- chip enable
- output enable
- write enable

To connect such a peripheral to the ARM Development Board, disconnect any EPROM or FLASH devices and wire up the peripheral to the appropriate socket using a transition header. If you need to support EPROM or FLASH and a peripheral, some modification of the controller may be necessary.

An alternative is to implement a similar controller and data-path router on a daughter card and attach it to the ASB using the 20-way headers provided. Refer to ***Chapter 4, Expanding and Monitoring the ASB*** for further information.

3.2.6 DRAM ASB Slave

This schematic is shown in ***A.8 DRAM ASB Slave*** on page A-9.

The DRAM subsystem comprises:

- an ASB slave controller (U14)
- two SIMM sockets (SK1 and SK2) which can be fitted with a variety of memory modules

The DRAM controller supports one or two 72-pin SIMM slots and uses automatic SIMM presence-detection for contiguous memory configuration. The DRAM controller provides:

- fast page-mode burst-mode sequential-access support
- byte, half-word and word transaction support
- 256-byte boundary page-mode cycle break-up
- DRAM refresh (using CAS-before-RAS Refresh mode)
- automatic module-size reconfiguration
- support for 4MB, 8MB and 16MB SIMM modules

Note *Although up to 64MB of DRAM can be fitted, the default memory map allows for 16MB. If you require more DRAM, modify the system decoder to allow access to the range required.*

The DRAM controller drives the address and control paths for the DRAM subsystem. The datapath is shared with the SRAM subsystem.

DRAM SIMMs supported

The DRAM controller supports 72-pin modules built of 4Mb and 16Mb technology DRAM devices, with a RAS Access Time specified at 70ns or faster.

Note *Where two SIMM modules are fitted, these must be of the same size and configuration. Where only one SIMM is fitted then this should only be fitted to slot A.*

The following size DRAM SIMM modules are supported:

4MB	32x1M	72-pin	70ns	(single-sided module, 8 x 1Mx4)
4MB	36x1M	72-pin	70ns	(single-sided module, 9 x 1Mx4)
8MB	32x2M	72-pin	70ns	(double-sided module, 2 x 8 x 1Mx4)
8MB	36x2M	72-pin	70ns	(double-sided module, 2 x 9 x 1Mx4)
16MB	32x4M	72-pin	70ns	(single-sided module, 8 x 4Mx4)
16MB	36x4M	72-pin	70ns	(single-sided module, 9 x 4Mx4)
32MB	32x8M	72-pin	70ns	(double-sided module, 2 x 8 x 4Mx4)
32MB	36x8M	72-pin	70ns	(double-sided module, 2 x 9x 4Mx4)

These SIMMs are standard commodity parts for desktop computers and workstations, and byte parity is not required or used. The basic memory size for all these modules is determined by the two presence-detect bits (PD1, PD0) with the addition of two pull-up resistors on the board. Only Slot A presence-detect is used.

3.2.7 SRAM ASB Slave

This schematic is shown in ***A.9 SRAM ASB Slave*** on page A-10.

The SRAM subsystem comprises:

- an ASB slave controller (U16)
- four 20ns 128KBx8 SRAM devices (U15,U17,U18 and U19)

The controller uses this memory to emulate two logical banks of 8-, 16- or 32-bit wide SRAM. Although there is one 8-bit wide device connected to each byte lane, the controller simulates narrow memory systems by inserting the correct number of wait states.

DIP switches

Slow SRAM can also be simulated through the use of DIP switches to control the number of bus cycles required for a memory access. Two-, three-, four-, and five-cycle memory can be emulated. DIP switches are also used to determine the memory width.

Switch	Name	Description	Options	Default
1	B0CYC0	Bank 0 number of cycles	see table below	on
2	B0CYC1	Bank 0 number of cycles	see table below	on
3	B0SIZ0	Bank 0 size (8,16,32-bit)	see table below	on
4	B0SIZ1	Bank 0 size (8,16,32-bit)	see table below	off
5	B1CYC0	Bank 1 number of cycles	see table below	on
6	B1CYC1	Bank 1 number of cycles	see table below	on
7	B1SIZ0	Bank 1 size (8,16,32-bit)	see table below	on
8	B1SIZ1	Bank 1 size (8,16,32-bit)	see table below	off

Table 3-5: DIP switch positions

The controller partitions the memory space into two logical banks of 256KB. The banks are called bank 0 and bank 1 and each has individual select lines for size and speed.

DIP switch (S2) has the following positions, where # is the bank number, 0 or 1

B#CYC1	B#CYC0	Cycles
on	on	2
on	off	3
off	on	4
off	off	5

B#SIZ1	B#SIZ0	Size
on	on	8-bit
on	off	16-bit
off	on	32-bit
off	off	32-bit

Table 3-6: S2 switch positions

The default configuration emulates 2-cycle 32-bit memory in both banks.

Many different configurations are possible. For example:

bank 0	8-bit 5 cycle (250ns @ 20MHz) memory (EPROM emulation)
bank 1	16-bit 2 cycle (100ns @ 20MHz) memory (standard SRAM).

Switch	Name	Position
1	B0CYC0	off
2	B0CYC1	off
3	B0SIZ0	on
4	B0SIZ1	on
5	B1CYC0	on
6	B1CYC1	on
7	B1SIZ0	off
8	B1SIZ1	on

Table 3-7: Switches

3.2.8 APB and NISA Bridge

This schematic is shown in ***A.10 APB and NISA Bridge*** on page A-11, and shows the following blocks:

- APB address and data buffers
- APB slave block
- NISA (not-ISA) bus peripherals block
- APB expansion block
- APB and NISA (not-ISA) bridge device

The Advanced Peripheral Bus (APB) connects to the ASB through the address and data buffers and the bridge which is implemented in (U20). In addition, this bridge generates signals required to interface to the ISA-type peripherals. This is not a full implementation of the ISA bus, hence not-ISA (NISA). The NISA bus peripherals comprise the PC card (PCMCIA) controller and the serial and parallel I/O device.

The bridge chip is responsible for generating all the APB control signals and enabling the address and data latches. The NISA bus shares the address and data latches with the APB bus. The bridge detects accesses to APB and NISA address space and acts accordingly.

Link (LK7) is used to select the width of the **P_STB** signal. The default (link out) is a strobe of two system clock cycles. If the link is inserted the **P_STB** signal is asserted for one system clock cycle only.

Note *The APB peripherals are not guaranteed to function correctly if the link is inserted and the system clock frequency is above 20MHz. Use this link with care.*

3.2.9 NISA Bus Peripherals

The NISA bus peripherals schematic is shown in ***A.11 NISA Bus Peripherals*** on page A-12, and shows the following blocks.

- serial and parallel I/O port block
- PC card (PCMCIA) interface block

Details of these blocks can be found in the following sections.

3.2.10 Serial and Parallel Ports

The serial and parallel ports schematic is shown in ***A.12 Serial and Parallel Ports*** on page A-13.

The serial and parallel I/O subsystem is based around an ST16C552 device. This is a dual asynchronous receiver and transmitter with 16-byte transmit and receive FIFO and a bi-directional Centronics type parallel printer port. This device is pin and functionally compatible with the VL16C552 and the WD16C552. In order to program the device you are advised to obtain a data sheet for one of these devices. Please contact ARM if you have any problems obtaining the datasheet.

The XR16C552 (U21) drives two serial ports (A and B) through two RS232 level shifters (U22 and U23). To connect to the serial ports you will need a cable that terminates in a 9-pin D socket. The pinout is compatible with a standard PC serial port and is shown ***Table 3-8: Serial port pinout***.

Pin	Function	Direction
1	DCD	in
2	RxD	in
3	TxD	out
4	DTR	out
5	GND	power
6	DSR	in
7	RTS	out
8	CTS	in
9	RI	in

Table 3-8: Serial port pinout

The ST16C552 also drives a Centronics-type parallel port. To connect to the parallel port you will need a cable that terminates in a 25-way D plug. The pinout is compatible with a standard PC parallel port and is shown in ***Table 3-9: Parallel port pinout*** on page 3-16.

Pin	Function	Direction
1	STROBE	i/o
2	DATA[0]	i/o
3	DATA[1]	i/o
4	DATA[2]	i/o
5	DATA[3]	i/o
6	DATA[4]	i/o
7	DATA[5]	i/o
8	DATA[6]	i/o
9	DATA[7]	i/o
10	ACK	in
11	BUSY	in
12	PE	in
13	SLCT	in
14	AUTOFD	i/o
15	ERROR	in
16	INIT	i/o
17	SLCTIN	i/o
18	GND	power
19	GND	power
20	GND	power
21	GND	power
22	GND	power
23	GND	power
24	GND	power
25	GND	power

Table 3-9: Parallel port pinout

Additional features

Some additional features have been added to the board to allow the parallel port to drive some LEDs and read switches.

DIP switch (S3) connects to bits 0-3 of the parallel port when the four lower bits of the link field (LK11) are jumpered.

Similarly the four yellow LEDs marked PP0–3 are connected to bits 4–7 of the parallel port when the upper four bits of the link field (LK11) are jumpered. This information is summarized in ***Table 3-10: LED’s and read switches***t:

Position	Bit	Connects to
1-2	0	S3 switch 1
3-4	1	S3 switch 2
5-6	2	S3 switch 3
7-8	3	S3 switch 4
9-10	4	LED PP0
11-12	5	LED PP1
13-14	6	LED PP2
15-16	7	LED PP3

Table 3-10: LED’s and read switches

Note *When using the parallel port to connect to external devices, you are advised to remove all the jumpers of link field (LK11). If you need to drive some LEDs and use an external device, you should consider driving the LEDs connected to the PC card (PCMCIA) controller.*

The ST16C552 can be configured through the use of two links:

- INT TYPE
- DIRN (LK8 and LK9)

LK8 selects the interrupt type, either latched mode (out) or ACK mode (in).

Latched mode
A falling edge on the ACK pin is latched and causes a parallel port interrupt. This interrupt is cleared by reading the appropriate status register. Latched mode is the default.

ACK mode
The ACK pin is connected directly to the parallel port interrupt line. If the parallel port ACK line is not connected externally, it is possible to make ACK pulse low by pressing the “momentary action” switch, SW1 (which has a black cap). To enable this function, the link ENABLE INT (LK10) must be inserted. If the link is out, pressing the switch has no effect.

Note *When using the parallel port to connect to external devices, you are advised to remove the jumper from ENABLE INT (LK10).*

L9 (The DIRN link) is connected to the BIDEN pin on the ST16C552. You can program the parallel port to be input or output under software control, but the method differs depending upon the state of the BIDEN pin. The default is link out which means that BIDEN is high and the direction is programmed by writing to the parallel port control register. Please refer to the 16C552 data sheet for further information.

Ref	Name	Description	Options	Default

LK8	INT TYPE	latched or ACK mode	out = latched
			in = ACK
				out

LK9	DIRN	parallel port direction	BIDEN select	out

LK10	ENABLE INT	enable switch interrupt	out = disabled
			in = enabled
				out

Table 3-11: Link summary

3.2.11 PC Card Interface

This schematic is shown in ***A.13 PC Card Interface*** on page A-14.

The PC card (PCMCIA) I/O subsystem is based around a Vadem VG-468 (U25) device. This a PC card socket controller which is designed to connect to a PC ISA bus. In this case, it is driven by the APB and NISA bridge. The controller supports two PC card sockets and the associated voltage switching devices.

In order to program the controller, you are advised to obtain a data sheet from the manufacturer. Please contact ARM if you have any problems obtaining a data sheet. It is a complex device and a description of its internal function is outside the scope of this document.

In combination with the voltage switching devices described in the next section the ARM Development Board is able to support cards that require a +5V supply and a programming voltage (VPP) of +5V or +12V.

Note *No support is provided for +3.3V PC cards.*

There are two yellow LEDs (D11 and D12) attached to the VG-486 labelled PCA and PCB. Logically, one LED is associated with each of the PC card sockets A and B. These LEDs are connected to the GPIO pins of the controller and so can be switched on an off by writing to an appropriate register in the device.

3.2.12 PC Card Connectors and Power Supply

The PC card connectors and power supply schematic is shown in ***A.14 PC Card Connecters and Power Supply*** on page A-15.

This schematic shows two voltage switching devices (U26 and U27) and two PC card socket connectors (SK4A and SK4B). The two socket connectors form one physical device, housing two card slots. The upper slot is A and the lower slot is B. The connectors are driven directly from the PC card controller, which is also responsible for determining the card voltage.

To function correctly, the MIC2560 (U26 and U27) requires supplies at +3.3V (VDD), +5V (VCC), and +12V (VPP). If any of these supplies are not connected, the supply presented to the card will be incorrect. The card VCC voltage depends upon two inputs, VCC5EN and VCC3EN. With VCC3EN tied HIGH through a surface mount link (LK12 and LK13), VCC5EN controls the supply.

Driving this signal LOW sets the card VCC voltage at +5V. Driving the signal HIGH puts the power supply pins into a high impedance state, effectively cutting off the supply to the card.

VG-468 signal name MIC2560-1 signal name	nVCCEN VCC5EN	- VCC3EN
VCC Supply5V	0	1
HIGH Z	1	1

The card VPP voltage depends upon two inputs:

- EN0
- EN1

By driving these signals HIGH and LOW the supply can be switched between +5V, +12V, GND and high impedance. See the table on the right for more information.

VG-468 signal name MIC2560-1 signal name	VPP1EN EN1	VPP2EN EN2
VPP Supply0V	0	0
5V	0	1
12V	1	0
HIGH Z	1	1

Note *The signal names for slot A are prefixed by A_.*
The signal names for slot B are prefixed by B_.

3.2.13 APB Slaves

This schematic is shown in ***A.15 APB Slaves*** on page A-16.

All the APB slaves are implemented in a single Xilinx field programmable gate array (FPGA). The XC4005 (U29) is programmed to provide the following functions:

- two 16-bit counter/timers with pre-scale
- interrupt controller
- reset and pause controller

Each of these functions is selected by accessing the appropriate address space. In addition to **P_SELCT**, **P_SELIC** and **P_SELRPC**, which are the select lines for the functions above, there is also a **P_SELEX** line which can be used to select user implemented functions. If you wish to reprogram the FPGA for your own use then contact ARM for VHDL descriptions of this device.

FPGA configuration

The FPGA is configured at power-up by a serial PROM (U28). The configuration can be downloaded from a workstation using a special download cable connected to header (J2). This procedure is detailed in ***Chapter 9, Programming the APB FPGA***.

Link field (LK16) is used to tell the FPGA whether it is to be programmed from the serial PROM or by download cable.

Position	Name	Description	Options	Default
1	INIT	not used	do not connect	out
2	MODE0	Number of cycles	out = cable, in = PROM	in
3	MODE1	Selects EPROM or FLASH	out = cable, in = PROM	in
4	MODE2	Selects 8 or 16-bit device	out = cable, in = PROM	in

Table 3-12: LK16

When the FPGA is successfully configured, the green LED marked “FPGA OK” lights up. If it does not light up, check that:

- the serial PROM (U28) and the links MODE0–2 are inserted
- the device has been configured by download cable

FPGA functionality

You can program the FPGA to have different functionality by replacing the serial PROM (U28) with an alternative device.

Note *The FPGA has just four outputs:* **ARMnFIQ**, **ARMnIRQ**, **STANDBY** *and* **REMAP**. *If you choose to change the function of the FPGA you must ensure that these outputs have the following default values:*

Output	Description	Default
ARMnFIQ	connects to nFIQ on processor	HIGH
ARMnIRQ	connects to nIRQ on processor	HIGH
STANDBY	puts system into standby state	LOW
REMAP	selects normal or reset memory map	AS REQUIRED

Table 3-13: FPGA outputs

See *Target Development System User Guide (ARM DUI 0061)* for further details on the **REMAP** signal.

3.2.14 APB Expansion Connecters

This schematic is shown in ***A.16 APB Expansion Connecters*** on page A-17.

This schematic shows six 20-way box headers (POD1-6) which can be used to expand or monitor the APB. POD6 is spare and can be used to connect external devices to signals on the board.

For expansion devices, there are four interrupt pins available:

- **nINTAPB[2:0]**
- **nFIQSRC**

These are all active low inputs to the interrupt controller, with on-board pull-up resistors.

Note *The* **nFIQSRC** *pin is also connected to the ASB expansion connectors, so in order to share the signal line, you must make drivers open-collector. To select external devices* **P_SELEX** *is available.*

If you are planning to build external expansion devices for the APB, refer to ***Chapter 5, Expanding and Monitoring the APB*** for further details.

3.2.15 APB Buffers

The APB buffers schematic is shown in ***A.17 APB Buffers*** on page A-18.

This schematic shows two 16-bit address buffers(U32 and U33) and two 16-bit data buffers (U30 and U31) that are used to connect the ASB to the APB. These devices are controlled by the APB and NISA bridge.

3.2.16 Memory Address and Data Buffers

This schematic is shown in ***A.18 Memory Address and Data Buffers*** on page A-19.

The schematic shows one 16-bit address latch (U34) and two 16-bit data buffers (U35 and U36) that are used to connect the ASB to the memory devices. These devices are controlled by the SRAM and DRAM ASB slaves.

3.2.17 Test Interface Controller and Connectors

This schematic is shown in ***A.19 Test Interface Controller and Connecters*** on page A-20.

The test interface comprises:

- a controller (U37), which is an ASB bus master
- four bi-directional transceivers (U38–41)
- two 20-way box headers (TEST1 and TEST2), which form the external connection.

Enabling the test interface

To enable the test interface, you must insert the link (LK17) marked “USE TIC”. If this link is inserted, it is important to drive the test bus signals **T_REQA**, **T_REQB** and **T_CLK**. Otherwise, the test interface controller may become the default bus master and the ARM processor will not be able to gain control of the ASB.

In normal operation the test interface is not used and LK17 should be left out.

Ref	Name	Description	Options	Default
LK17	USETIC	enable the TIC	out=disable, in=enable	out

Table 3-14: LK17

For full details of the test interface refer to the *AMBA Specification* (ARM IHI 001).

3.2.18 Master Header Connectors and Level Convertors

This schematic is shown in ***A.20 Master Header Connecters and Level Converters*** on page A-21.

Master header connectors

The ARM processor, mounted on a daughter card, is connected to the board using four 60-way connectors (PL6–9), as shown on this schematic. A number of the connector pins are not used, as these are reserved for future expansion.

Level convertors

Because the board functions at 5V and the processor is a 3.3V component, level convertors (U42–53) are provided to prevent high signal levels causing damage. These level convertors are constructed from “Quickswitch” buffers. These devices have very low propagation delay (less than 250ps) and appear as a 5 ohm resistor when switched on.

The output voltage for an input voltage equal to the supply is approximately 1V below the supply. A resistor/diode network is used to provide a supply of 4.3V, so that high input levels are clamped to 3.3V when driven out of the device.

3.2.19 System Modules (Arbiter and Decoder)

This schematic is shown in ***A.21 System Modules (Arbiter and Decoder)*** on page A-22. The schematic shows the two ASB system modules:

- the arbiter
- the decoder

Arbiter

The arbiter (U54) is responsible for deciding which bus master gains control of the ASB. In addition, this device also controls the system reset **B_RES[2:0]** lines.

Two surface mount links (LK14 and LK15) are provided to enable the expansion bus request lines (**A_REQ001** and **A_REQ002**). In normal operation these lines are tied LOW through the links, but if you connect a bus master to the ASB, you need to enable one or other of these request lines by moving the appropriate link to the A-C position.

Pressing SW2 (the switch with a red cap) causes a full system reset.

Decoder

The decoder (U55) is responsible for driving the ASB select lines (**D_SELxxx**) and is vital to the correct operation of the board. This device can be reprogrammed to implement alternative memory maps if required.

The decoder functions in two distinct states:

- normal
- reset

On power-up, the board is by default in the reset configuration. In this state the EPROM or FLASH can be found at the bottom of the address map. If the remap register is written to, the **REMAP** signal goes HIGH and the decoder switches to the normal memory map where there is SRAM at the bottom.

If you are bringing up a system where there is no EPROM or FLASH, you may want to disable this feature. Link (LK18) is provided for this purpose. In the default position (in), the **REMAP** input to the decoder is driven by the **REMAP** and pause controller implemented in the APB FPGA. If this link is removed, the **REMAP** signal is always high, and the board starts up with SRAM at the bottom of the address map.

Ref	Name	Description	Options	Default
LK18	REMAP	driven or always high	out=high, in=driven	in

Table 3-15: LK18

3.2.20 ASB Expansion Connectors

The ASB expansion connectors schematic is shown in ***A.22 ASB Expansion Connecters*** on page A-23.

This schematic shows six 20-way box headers (POD7–12) which can be used to expand or monitor the ASB.

For expansion devices there are three interrupt pins available **nINTASB[1:0]** and **nFIQSRC**. These are all active LOW inputs to the interrupt controller, with on-board pull-up resistors. Note however that **nFIQSRC** is also connected to the APB expansion connectors, so in order to share the signal line make drivers open-collector. To select external devices two signals **D_SELASB[1:0]** are available. To enable these lines drive the signals **nENASB[1:0]** LOW.

If you are planning to build external expansion devices for the ASB, refer to ***Chapter 4, Expanding and Monitoring the ASB*** for further details.

3.3 ARM7TDMI Processor Daughter Board

This schematic is shown in ***B.2 Top-level Diagram*** on page B-3.

This AMBA bus master daughter card comprises an ARM7TDMI test chip and a PLD that converts it into an AMBA master. It is connected to the main board through four 60-way connectors and provides headers to which a logic analyser may be connected. The JTAG interface on the test chip is brought out to a header to which an EmbeddedICE interface may be connected.

The top-level schematic shows the following blocks:

- processor (QPF or PGA)
- header card connectors
- AMBA bus master veneer
- logic analyser connectors
- EmbeddedICE interface connector

3.3.1 Processor in QFP package

This schematic is shown in ***B.6 Processor in QFP Package*** on page B-7.

This schematic shows the ARM7TDMI test chip in a QFP package. A number of inputs are tied to default values through resistors.

To learn more about the ARM7TDMI refer to the documents:

- *ARM7TDMI Data Sheet* (ARM DDI 0029) and
- *ARM7TDMI Test Chip Appendix* (ARM DXI 0022).

3.3.2 Processor in PGA package

This schematic is shown in ***B.7 Processor in PGA Package*** on page B-8.

This schematic shows the ARM7TDMI test chip in a PGA package. It is identical in all other respects to the QFP packaged device.

3.3.3 Header card connectors

This schematic is shown in ***B.3 Header Connecters*** on page B-4.

This schematic shows four 60-way connectors (SK1–4) which are used to attach the daughter card to the main board. A number of the connector pins are not used, these are reserved for future expansion.

Refer to section ***3.2.18 Master Header Connectors and Level Convertors*** to see which ASB signals the processor is connected to.

3.3.4 AMBA bus master veneer

This schematic is shown in ***B.5 AMBA Bus Master Veneer*** on page B-6.

In order to turn an ARM processor (such as the ARM7TDMI test chip) into an AMBA bus master an AMBA veneer is required. This function is performed by the MACH215 device (U2). Because this is a 5V part it is necessary to level shift the outputs so that output high voltages do not damage the processor. The two level convertors (U3 and U4) are constructed from “Quickswitch” buffers. These devices have very low propagation delay (less than 250ps) and appear as a 5 ohm resistor when switched on. The output voltage for an input voltage equal to the supply is approximately 1V below the supply. A resistor/diode network (R32 and D1) is used to provide a supply of 4.3V, so that high input levels are clamped to 3.3V when driven out of the device.

There are four surface mount links (LK1–4) which operate as follows. BMEN0 (LK1) and BMEN1 (LK2) are configuration inputs to the processor and AMBA veneer device. These links should always be connected A–C as their functionality is reserved for future use.

OLDTC (LK3) allows older revisions of the ARM7TDMI test chip to be used in this system. Before revision 1 of the ARM7TDMI test chip some of the AMBA signals were not available. This link configures the AMBA veneer to provide these functions on behalf of the processor. By default this link is in the B–C position as all production headers will be fitted with revision 1 or higher ARM7TDMI devices.

The GRANT SELECT link (LK4) is used to configure the way in which the processor is granted and enabled onto ASB. By default this link is in position A–C and should not be moved.

3.3.5 Logic analyser connectors

This schematic is shown in ***B.4 Logic Analyser Connecters*** on page B-5.

Six 20-way box headers (POD1–6) are provided to allow connection of Hewlett Packard 20 pin (HP 01650-63203) pods suitable for use with HP1650B-series logic analysers. These connectors can also be used for expansion purposes. Using a logic analyser it is possible to observe the processor cycles and if required disassemble ARM instruction mnemonics to trace program execution. This procedure and the POD pin assignments are covered in detail in ***Chapter 7, The Logic Analyser Interface***.

3.3.6 EmbeddedICE interface

This schematic is shown in ***B.8 EmbeddedICE Interface*** on page B-9.

This schematic shows the EmbeddedICE JTAG connector (PL1). It is a 14-way box header, compatible with the ARM EmbeddedICE interface. To connect this device to an EmbeddedICE unit you will need a short 14-way IDC cable which is supplied with that interface.

For further information on the EmbeddedICE interface refer to ***Chapter 6, The EmbeddedICE Interface***.

4 Expanding and Monitoring the ASB

This chapter describes how to expand and monitor the ASB.

Expanding and Monitoring the ASB

4.1 Expanding the ASB

Note *Please refer to the AMBA Specification (ARM IHI 0001) for a detailed description of the signals mentioned in this section.*

4.1.1 Headers and pinout

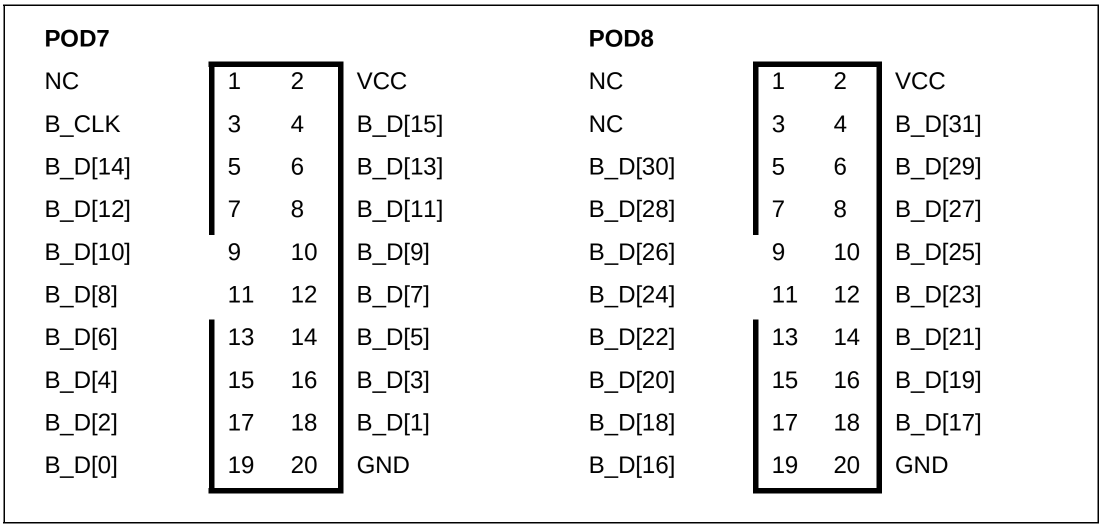
The ASB expansion interface comprises six 20-way box headers, horizontally mounted along the top edge of the development card. The headers are numbered POD7 to POD12.

Each header has a pinout that is compatible with Hewlett Packard HP1650B series logic analyser pods (HP01650-63203). Unconnected pins are labeled NC. These pins can be used to connect other signals to the header if required.

Note *If the logic analyser pods described are used, they supply +5V on pin 1, so this should not be connected to any output on the board. The logic analyser pods have signal inputs on pins 4–19 and a trigger input on pin 3.*

The pods are assigned as follows:

POD7:	low ASB data bus	**B_D[15:0]**	trigger **B_CLK**
POD8:	high ASB data bus	**B_D[31:16]**	no trigger input
POD9:	low ASB address bus	**B_A[15:0]**	trigger **nB_CLK**
POD10:	high ASB address bus	**B_A[31:16]**	no trigger input
POD11:	ASB control signals		no trigger input
POD12:	ASB control signals		no trigger input



POD7				POD8			
NC	1	2	VCC	NC	1	2	VCC
B_CLK	3	4	B_D[15]	NC	3	4	B_D[31]
B_D[14]	5	6	B_D[13]	B_D[30]	5	6	B_D[29]
B_D[12]	7	8	B_D[11]	B_D[28]	7	8	B_D[27]
B_D[10]	9	10	B_D[9]	B_D[26]	9	10	B_D[25]
B_D[8]	11	12	B_D[7]	B_D[24]	11	12	B_D[23]
B_D[6]	13	14	B_D[5]	B_D[22]	13	14	B_D[21]
B_D[4]	15	16	B_D[3]	B_D[20]	15	16	B_D[19]
B_D[2]	17	18	B_D[1]	B_D[18]	17	18	B_D[17]
B_D[0]	19	20	GND	B_D[16]	19	20	GND

Figure 4-1: Pods 7 and 8

POD9				POD10			
NC	1	2	VCC	NC	1	2	VCC
nB_CLK	3	4	B_A[15]	NC	3	4	B_A[31]
B_A[14]	5	6	B_A[13]	B_A[30]	5	6	B_A[29]
B_A[12]	7	8	B_A[11]	B_A[28]	7	8	B_A[27]
B_A[10]	9	10	B_A[9]	B_A[26]	9	10	B_A[25]
B_A[8]	11	12	B_A[7]	B_A[24]	11	12	B_A[23]
B_A[6]	13	14	B_A[5]	B_A[22]	13	14	B_A[21]
B_A[4]	15	16	B_A[3]	B_A[20]	15	16	B_A[19]
B_A[2]	17	18	B_A[1]	B_A[18]	17	18	B_A[17]
B_A[0]	19	20	GND	B_A[16]	19	20	GND

Figure 4-2: Pods 9 and 10

POD11				POD12			
NC	1	2	VCC	NC	1	2	VCC
NC	3	4	D_SELASB[1]	NC	3	4	NC
D_SELASB[0]	5	6	B_ERROR	D_SELSRAM	5	6	D_SELSSRAM
B_LAST	7	8	B_WAIT	D_SELROM	7	8	D_SELDRAM
B_LOCK	9	10	B_RES[2]	D_SELNISA	9	10	D_SELAPB
B_RES[1]	11	12	B_RES[0]	nFIQSRC	11	12	nENASB[1]
B_PROT[1]	13	14	B_PROT[0]	nENASB[0]	13	14	nINTASB[1]
B_TRAN[1]	15	16	B_TRAN[0]	nINTASB[0]	15	16	A_GNT002
B_SIZE[1]	17	18	B_SIZE[0]	A_GNT001	17	18	A_REQ002
B_WRITE	19	20	GND	A_REQ001	19	20	GND

Figure 4-3: Pods 11 and 12

4.1.2 List of signals

The following list describes each signal.

Signal	Description
B_CLK	AMBA system clock
nB_CLK	AMBA system clock inverted
B_RES[2:0]	AMBA reset signals
B_D[31:0]	ASB data bus
B_A[31:0]	ASB address bus
B_WAIT	ASB wait response
B_LAST	ASB last response
B_ERROR	ASB error response
B_LOCK	ASB locked transfers
B_PROT[1:0]	ASB protection control
B_TRAN[1:0]	ASB transfer type
B_SIZE[1:0]	ASB transfer size
B_WRITE	ASB transfer direction
D_SELASB[1:0]	ASB expansion select signals
D_SELSSRAM	ASB select SSRAM controller
D_SELSRAM	ASB select SRAM controller
D_SELDRAM	ASB select DRAM controller
D_SELROM	ASB select EPROM/FLASH controller
D_SELAPB	ASB select APB bridge
D_SELNISA	ASB select NISA bridge
nINTASB[1:0]	interrupt sources, active low, level sensitive
nFIQSRC	fast interrupt source, active low, level sensitive

Table 4-1: ASB signals

Signal	Description
nENASB[1:0]	enable ASB expansion slaves
A_REQ001	ASB expansion request signal 1
A_REQ002	ASB expansion request signal 2
A_GNT001	ASB expansion grant signal 1
A_GNT002	ASB expansion grant signal 2

Table 4-1: ASB signals (Continued)

4.2 Building an ASB Master Expansion Board

To build an ASB master that connects to the motherboard, you need some or all of the signals listed in ***Table 4-1: ASB signals***. The following are particularly important

A_REQ001	request to system arbiter for bus ownership
A_REQ002	request to system arbiter for bus ownership
A_GNT001	grant from system arbiter of bus ownership
A_GNT002	grant from system arbiter of bus ownership

4.2.1 Surface mount links

By default, **A_REQ001** and **A_REQ002** are tied LOW on the motherboard through surface mount links. When implementing expansion bus masters, the surface mount links must be moved to connect the arbiter request lines to the expansion header. Refer to section ***3.2.19 System Modules (Arbiter and Decoder)*** on page 3-24 for more information on this.

4.2.2 Interrupts

A master can issue interrupts to the system CPU through the following:

nINTASB[1:0]	interrupt sources
nFIQSRC	fast interrupt source

Note *An expansion master cannot receive interrupts from other devices in the system.*

4.3 Building an ASB Slave Expansion Board

To build an ASB peripheral or slave that connects to the motherboard, you need some of the following signals.

B_CLK	or **nB_CLK** if other clock edge is required
B_RES[2:0]	only **B_RES[1]** is usually required
B_D	a number of data bits
B_A	a number of address bits
B_WRITE	transfer direction
B_SIZE[1:0]	transfer size
D_SELASB[1:0]	select signal reserved for expansion slaves

4.3.1 Slaves

Normally the system decoder asserts **B_WAIT**, **B_ERROR** and **B_LAST** on behalf of the absent slaves. If the slave needs to drive these signals:

nENASB[1:0]	must be driven or tied LOW

In this case, the slave must drive the following signals or the system cannot function correctly:

B_WAIT	waits for the master
B_ERROR	signals a slave error
B_LAST	signals the last transfer in a burst

If the slave generates interrupts:

nINTASB[1:0]	one or more of these IRQ sources
nFIQSRC	possibly the FIQ source

4.4 ASB Timing on the ARM Development Board

The AMBA specification does not specify bus timing, as this depends upon the technology used. For expansion on the ARM Development Board, it is important to have some timing guidelines. To assist with this, the following timings have been defined:

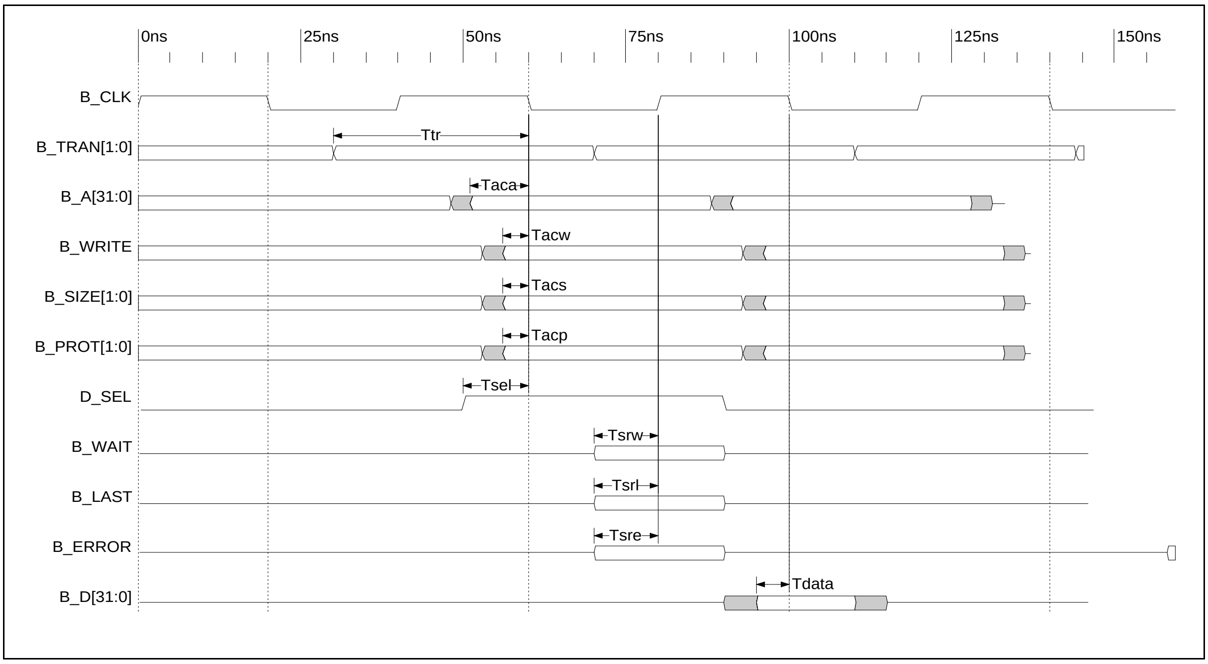


Figure 4-4: ASB timings on the ARM Development Board

Parameter	Description	Min	Typ	Max
Ttr	**B_TRAN** setup to **B_CLK** falling	13.5		
Taca	**B_A** setup to **B_CLK** falling	5		
Tacw	**B_WRITE** setup to **B_CLK** falling	5		
Tacs	**B_SIZE** setup to **B_CLK** falling	5		
Tacp	**B_PROT** setup to **B_CLK** falling	5		
Tsrw	**B_WAIT** setup to **B_CLK** rising	8		
Tsrl	**B_LAST** setup to **B_CLK** rising	8		
Tsre	**B_ERROR** setup to **B_CLK** rising	8		
Tdata	**B_D** setup to **B_CLK** falling	5		

Table 4-2: Sample timings

5 Expanding and Monitoring the APB

The ARM Development Board implements an APB with full 32-bit address and data buses. In a typical system, these buses may well be narrower; the APB slaves only use 16 data lines and nine address lines. Full 32-bit support is provided for flexibility when expanding the APB.

This chapter describes how to expand and monitor the APB.

Expanding and Monitoring the APB

5.1 APB Expansion Interface

Note *Please refer to the AMBA Specification (ARM IHI 0001) for a detailed description of the signals mentioned in this section.*

5.1.1 Headers and pinout

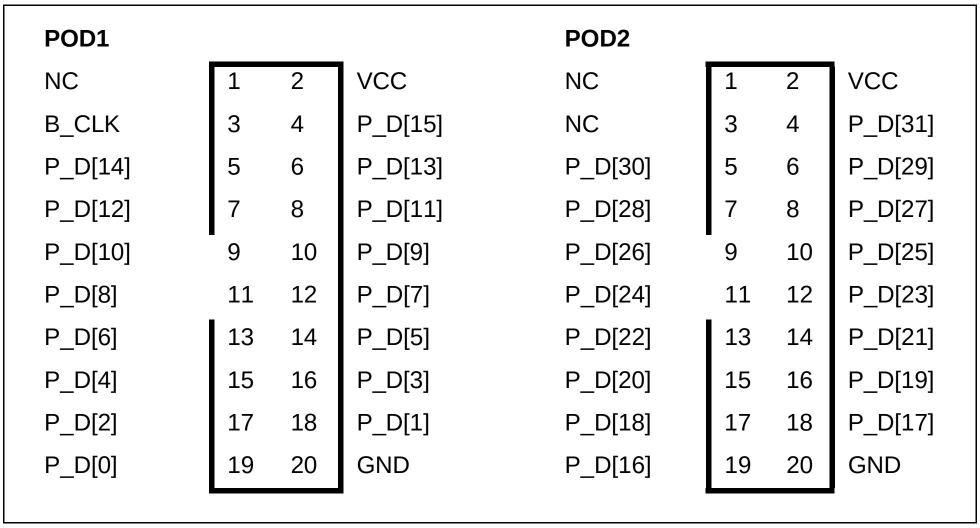
The APB expansion interface comprises six 20-way box headers horizontally mounted along the bottom edge of the development card. The headers are numbered POD1 to POD6.

Each header has a pinout that is compatible with Hewlett Packard HP1650B series logic analyser pods (HP01650-63203). Unconnected pins are labelled NC. These pins can be used to connect other signals to the header if required.

Note *If the logic analyser pods described are used, they supply +5V on pin 1, so this should not be connected to any output on the board. The logic analyser pods have signal inputs on pins 4–19 and a trigger input on pin 3.*

The pods are assigned as follows:

POD1:	low APB data bus	**P_D[15:0]**	trigger **B_CLK**
POD2:	high APB data bus	**P_D[31:16]**	no trigger input
POD3:	low APB address bus	**P_A[15:0]**	trigger **nB_CLK**
POD4:	high APB address bus	**P_A[31:16]**	no trigger input
POD5:	APB control signals		trigger **P_STB**
POD6:	unassigned, for future expansion		



POD1				POD2			
NC	1	2	VCC	NC	1	2	VCC
B_CLK	3	4	P_D[15]	NC	3	4	P_D[31]
P_D[14]	5	6	P_D[13]	P_D[30]	5	6	P_D[29]
P_D[12]	7	8	P_D[11]	P_D[28]	7	8	P_D[27]
P_D[10]	9	10	P_D[9]	P_D[26]	9	10	P_D[25]
P_D[8]	11	12	P_D[7]	P_D[24]	11	12	P_D[23]
P_D[6]	13	14	P_D[5]	P_D[22]	13	14	P_D[21]
P_D[4]	15	16	P_D[3]	P_D[20]	15	16	P_D[19]
P_D[2]	17	18	P_D[1]	P_D[18]	17	18	P_D[17]
P_D[0]	19	20	GND	P_D[16]	19	20	GND

Figure 5-1: Pods 1 and 2

POD3				POD4			
NC	1	2	VCC	NC	1	2	VCC
nB_CLK	3	4	P_A[15]	NC	3	4	P_A[31]
P_A[14]	5	6	P_A[13]	P_A[30]	5	6	P_A[29]
P_A[12]	7	8	P_A[11]	P_A[28]	7	8	P_A[27]
P_A[10]	9	10	P_A[9]	P_A[26]	9	10	P_A[25]
P_A[8]	11	12	P_A[7]	P_A[24]	11	12	P_A[23]
P_A[6]	13	14	P_A[5]	P_A[22]	13	14	P_A[21]
P_A[4]	15	16	P_A[3]	P_A[20]	15	16	P_A[19]
P_A[2]	17	18	P_A[1]	P_A[18]	17	18	P_A[17]
P_A[0]	19	20	GND	P_A[16]	19	20	GND

Figure 5-2: Pods 3 and 4

POD5				POD6			
NC	1	2	VCC	NC	1	2	VCC
P_STB	3	4	NC	NC	3	4	NC
NC	5	6	NC	NC	5	6	NC
NC	7	8	nINTAPB[2]	NC	7	8	NC
nINTAPB[1]	9	10	nINTAPB[0]	NC	9	10	NC
B_RES[1]	11	12	B_RES[2]	NC	11	12	NC
nFIQSRC	13	14	B_RES[0]	NC	13	14	NC
P_SELIC	15	16	P_SELRC	NC	15	16	NC
P_SELCT	17	18	P_SELEX	NC	17	18	NC
P_WRITE	19	20	GND	NC	19	20	GND

Figure 5-3: Pods 5 and 6

5.1.2 List of signals

The following list describes each signal.

Signal	Description
B_CLK	AMBA system clock
nB_CLK	AMBA system clock inverted
B_RES[2:0]	AMBA reset signals
P_D[31:0]	APB data bus
P_A[31:0]	APB address bus
P_STB	APB strobe
nINTAPB[2:0]	interrupt sources, active low, level sensitive
nFIQSRC	fast interrupt source, active low, level sensitive
P_SELIC	APB select signal for interrupt controller
P_SELRC	APB select signal for reset/pause controller
P_SELCT	APB select signal for counter/timers
P_SELEX	APB select signal for expansion device
P_WRITE	APB read/write signal

Table 5-1: APB signals

5.2 Building an APB Slave Expansion Board

To build an APB peripheral or slave that connects to the motherboard, you need the following signals:

P_D	a number of data bits
P_A	a number of address bits
P_STB	strobe signal
P_WRITE	read/write signal
P_SELEX	select signal reserved for expansion devices

5.2.1 System clock and reset

If the application requires a system clock and reset:

B_RES[2:0]	only **B_RES[2]** is usually required
B_CLK	or **nB_CLK** if other clock edge is required

5.2.2 Interrupts

If the slave generates interrupts:

nINTAPB[2:0]	one or more of these IRQ sources
nFIQSRC	possibly the FIQ source

5.3 APB Timing on the ARM Development Board

The AMBA specification does not specify bus timing, as this depends upon the technology used. For expansion on the ARM Development Board it is important to have some timing guidelines. To assist with this, the following timings have been defined:

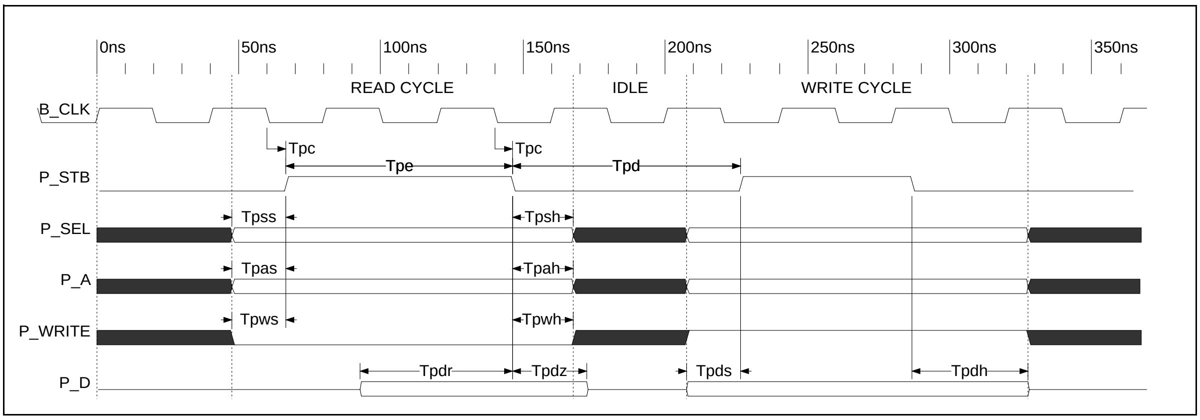


Figure 5-4: APB timings on the ARM Development Board

This shows a read and write cycle, separated by an idle cycle. If multiple reads or writes occur, they are not separated by an idle cycle, and this gives the minimum Tpd of 40ns at 25MHz.

5.3.1 P_STB signal

Figure 5-4: APB timings on the ARM Development Board shows that the **P_STB** signal lasts for two **B_CLK** cycles. This is because the APB slaves are implemented in a Xilinx FPGA, which is a relatively slow device.

If the system clock frequency is set at 20MHz or below, **P_STB** need only last for one **B_CLK** cycle. By inserting a link the APB bridge is instructed to shorten the **P_STB** HIGH pulse to one **B_CLK** cycle. Refer to section ***3.2.8 APB and NISA Bridge*** on page 3-14 for details.

To implement slower APB slaves, it is necessary to reprogram the APB bridge MACH chip so that additional wait states are inserted. Refer to ***Chapter 10, Programming the MACH and PAL Devices*** for further details of this procedure.

Parameter	Description	Min	Typ	Max
Tpe	**P_STB** high pulse width (enabled time @ 25MHz)		80	
Tpd	**P_STB** low pulse width (disabled time @ 25MHz)		40	
Tpss	**P_SEL** setup to **P_STB** rising	21		
Tpas	**P_A** setup to **P_STB** rising	21		
Tpws	**P_WRITE** setup to **P_STB** rising	21		
Tpsh	**P_SEL** hold from **P_STB** falling			19
Tpah	**P_A** hold from **P_STB** falling			19
Tpwh	**P_WRITE** hold from **P_STB** falling			19
Tpds	**P_D** setup to **P_STB** rising (write)	19		
Tpdh	**P_D** hold from **P_STB** falling (write)			41
Tpdr	**P_D** setup to **P_STB** falling (read)	54		
Tpdz	**P_D** hold from **P_STB** falling (read)			26
Tpc	**B_CLK** falling to **P_STB** falling			6.5

Table 5-2: Sample timings

6 The EmbeddedICE Interface

This chapter describes how the EmbeddedICE interface connects to the ARM Development Board.

The EmbeddedICE Interface

6.1 EmbeddedICE Interface

A debuggable ARM processor (such as ARM7TDMI) on the ARM Development Board can be controlled through its JTAG port using the EmbeddedICE interface.

The EmbeddedICE interface is packaged separately from the ARM Development Board. The host computer requires the ARM Software Development Toolkit to communicate with the EmbeddedICE interface: the *ARM Software Development Toolkit Reference Manual* (*ARM DUI 0020*) explains how to use the tools with the EmbeddedICE interface.

This chapter describes the physical connection between the EmbeddedICE interface and the ARM Development Board. For details on how to connect the system components together refer to the *Target Development System User Guide (ARM DUI 0061)*.

If you have the Angel Debug Monitor resident in the on-board FLASH then you do not need to use the EmbeddedICE interface.

6.1.1 EmbeddedICE interface connector

The interface connector PL1 (shown below), is mounted on the header card. It is a 14-way box header as shown in ***Figure 6-1: EmbeddedICE interface connector PL1 (viewed from above)*** below.

This plug is connected to the EmbeddedICE interface module using a short 14-way IDC cable with IDC sockets at each end.

Note *The signals on this interface are at 3.3V, so care should be taken if the JTAG port is connected to any 5V system.*

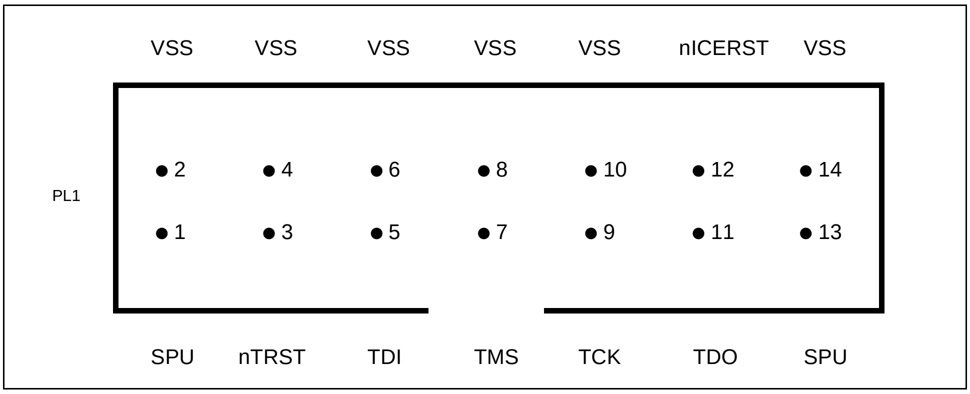


Figure 6-1: EmbeddedICE interface connector PL1 (viewed from above)

The connector pins are shown in ***Table 6-1: EmbeddedICE interface connector pins***.

Pin descriptions

The functions of the connector pins are summarised in the following table:

Pin	Name	Function
1	**SPU**	System powered up, pin connected to Vdd through a 33R resistor
3	**nTRST**	Test reset, active low
5	**TDI**	Test data in
7	**TMS**	Test mode select
9	**TCK**	Test clock
11	**TDO**	Test data out
12	**nRSTOUT**	unused
13	**SPU**	As pin 1
2, 4, 6, 8, 10, 14	**VSS**	System ground

Table 6-1: EmbeddedICE interface connector pins

7 The Logic Analyser Interface

This chapter describes the features of the ARM Development Card that facilitate code development and debugging.

In most cases, you can download code and debug it using the ARM toolkit, in combination with either a debug monitor resident on the board or an EmbeddedICE interface. However, you sometimes need to use a logic analyser so you can debug code in real time.

The Logic Analyser Interface

7.1 ARM HP Inverse Assembler

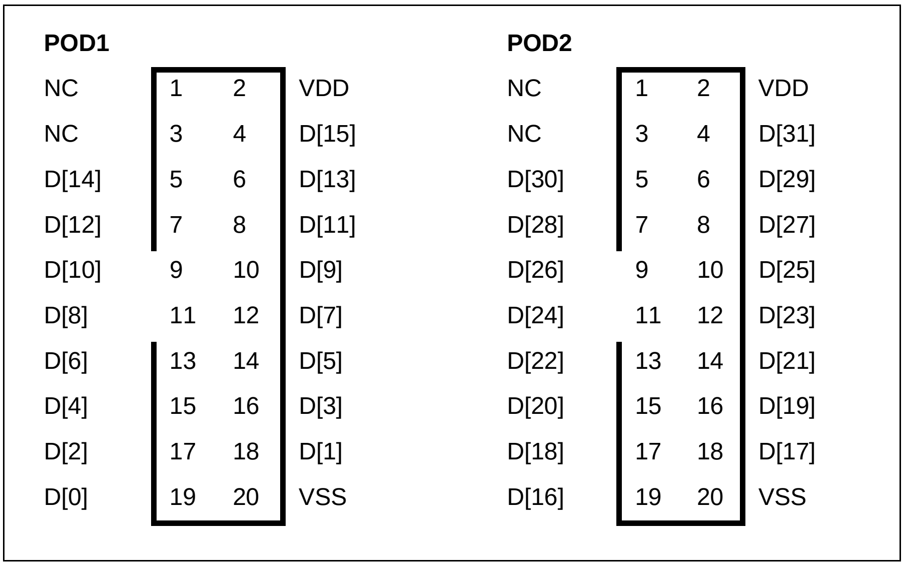
ARM have developed an inverse assembler for use with HP logic analyzers to enable disassembly of the code. A set of six 2-way box headers are provided on the ARM Development Board for this purpose (POD 7-12). A further set is mounted on the ARM TDMI Daughter Board. See the *ARM HP Inverse Assembler User Guide (ARM DUI 0062)* for more information on the inverse assembler.

7.1.1 Connector and pinout

To enable you to observe processor cycles and signals on the ASP or APB buses, a further six 20-way box headers are mounted along two sides of the AMBA master daughter card. The headers are numbered POD1 to POD6 and they connect directly to the ARM processor. Each header has a pinout that is compatible with Hewlett Packard HP1650B series logic analyser pods (HP01650-63203).

The pods are assigned as follows:

POD1:	low data bus	**D[15:0]**	no trigger input
POD2:	high data bus	**D[31:16]**	no trigger input
POD3:	low address bus	**A[15:0]**	no trigger input
POD4:	high address bus	**A[31:16]**	no trigger input
POD5:	bus control signals		trigger **MCLK**
POD6:	bus control signals		trigger **ECLK**



POD1				**POD2**			
NC	1	2	VDD	NC	1	2	VDD
NC	3	4	D[15]	NC	3	4	D[31]
D[14]	5	6	D[13]	D[30]	5	6	D[29]
D[12]	7	8	D[11]	D[28]	7	8	D[27]
D[10]	9	10	D[9]	D[26]	9	10	D[25]
D[8]	11	12	D[7]	D[24]	11	12	D[23]
D[6]	13	14	D[5]	D[22]	13	14	D[21]
D[4]	15	16	D[3]	D[20]	15	16	D[19]
D[2]	17	18	D[1]	D[18]	17	18	D[17]
D[0]	19	20	VSS	D[16]	19	20	VSS

Figure 7-1: Pods 1 and 2

POD3				POD4			
NC	1	2	VDD	NC	1	2	VDD
NC	3	4	P_A[15]	NC	3	4	A[31]
A[14]	5	6	P_A[13]	A[30]	5	6	A[29]
A[12]	7	8	P_A[11]	A[28]	7	8	A[27]
A[10]	9	10	P_A[9]	A[26]	9	10	A[25]
A[8]	11	12	P_A[7]	A[24]	11	12	A[23]
A[6]	13	14	P_A[5]	A[22]	13	14	A[21]
A[4]	15	16	P_A[3]	A[20]	15	16	A[19]
A[2]	17	18	P_A[1]	A[18]	17	18	A[17]
A[0]	19	20	VSS	A[16]	19	20	VSS

Figure 7-2: Pods 3 and 4

POD5				POD6			
NC	1	2	VDD	NC	1	2	VDD
MCLK	3	4	nEXEC	ECLK	3	4	NC
PIPEF	5	6	nM[1]	NC	5	6	NC
nM[0]	7	8	nOPC	NC	7	8	NC
MAS[1]	9	10	MAS[0]	nM[4]	9	10	nM[3]
nRW	11	12	SEQ	nM[2]	11	12	DBGACK
nMREQ	13	14	ABORT	nTRANS	13	14	LOCK
nIRQ	15	16	nFIQ	TRAN[1]	15	16	TRAN[0]
nRESET	17	18	PWAIT	CPA	17	18	CPB
TBIT	19	20	VSS	nCPI	19	20	VSS

Figure 7-3: Pods 5 and 6

7.1.2 List of signals

The following list describes each signal.

Signal	Description
D[31:0]	data bus
A[31:0]	APB address bus
MCLK	system clock (equivalent to **B_CLK**)
PIPEF	pipeline full
nM[4:0]	processor mode
MAS[1:0]	transfer size (equivalent to **B_SIZE[1:0]**)
nRW	read/write (equivalent to **B_WRITE**)
nMREQ	memory request
nFIQ	fast interrupt request
nIRQ	interrupt request
nRESET	processor reset (equivalent to **B_RES[1]**)
TBIT	Thumb bit
nEXEC	instruction not executed
nOPC	opcode fetch
SEQ	sequential address
ABORT	memory abort
PWAIT	processor wait
ECLK	external clock output
nTRANS	processor user mode
TRAN[1:0]	processor **BC[1:0]** (equivalent to **B_TRAN[1:0]**)
CPA	co-processor absent
CPB	co-processor busy
nCPI	co-processor instruction
VDD	+3.3V
VSS	system ground

Table 7-1: Logic analyser pod signals

8 The Test Interface

This chapter describes how to use the test interface.

The Test Interface

8.1 Introducing the Test Interface

The test interface comprises:

- a controller
- a number of bidirectional latched transceivers that control flow between the test bus and ASB

The test interface is the default ASB master. This means that if no other master is requesting the bus, the arbiter grants access to the test interface controller (TIC).

Using the test bus interface, you can gain control of the ASB in real time and perform manufacturing test and in-circuit diagnostics. On a circuit board this is not usually a problem, but the ARM Development Board is a board-level implementation of a typical ASIC, where it would not be possible to examine the internal connections.

Note *Refer to the AMBA Specification (ARM IHI 0001) for details of the test port provided.*

8.1.1 External signals

The following external signals are defined:

T_CLK	test clock	input
T_REQA	test bus request A	input
T_REQB	test bus request B	input
T_ACK	test acknowledge	output
T_D[31:0]	test bus	bi-directional
B_D[31:0]	ASB data bus	bi-directional
B_A[31:0]	ASB address bus	bi-directional

8.2 Connecting External Equipment to the Test Bus

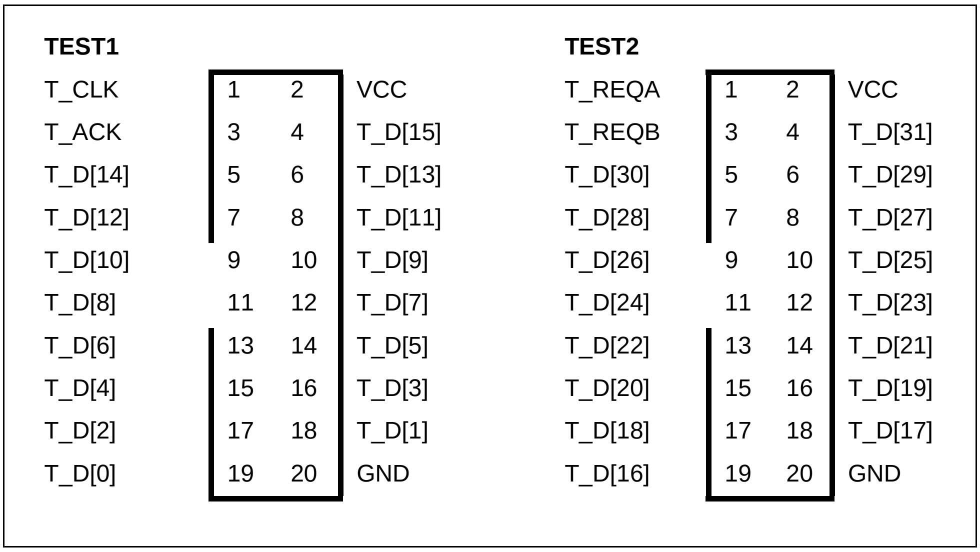
To drive the test interface on the ARM Development Board some external equipment is required. This is typically a parallel I/O device offering 36 separate input/output lines.

8.2.1 Headers and pinout

The equipment is connected to the board via two 20-way box headers mounted on the right edge of the development card. These are called:

- TEST1
- TEST2

The connector pins are assigned as follows:



TEST1				TEST2			
T_CLK	1	2	VCC	T_REQA	1	2	VCC
T_ACK	3	4	T_D[15]	T_REQB	3	4	T_D[31]
T_D[14]	5	6	T_D[13]	T_D[30]	5	6	T_D[29]
T_D[12]	7	8	T_D[11]	T_D[28]	7	8	T_D[27]
T_D[10]	9	10	T_D[9]	T_D[26]	9	10	T_D[25]
T_D[8]	11	12	T_D[7]	T_D[24]	11	12	T_D[23]
T_D[6]	13	14	T_D[5]	T_D[22]	13	14	T_D[21]
T_D[4]	15	16	T_D[3]	T_D[20]	15	16	T_D[19]
T_D[2]	17	18	T_D[1]	T_D[18]	17	18	T_D[17]
T_D[0]	19	20	GND	T_D[16]	19	20	GND

Figure 8-1: Connector pin assignments

8.2.2 Test vectors

ARM has code that runs on a host computer and drives an I/O board. This code requires a test vector file as input. The test vector file is written to provide stimulus and check data that is read back from the test bus.

The program has been written to be hardware-independent, so you only have to write a few functions to interface to your chosen system.

If you need to use the test interface and would like to take advantage of this code then please contact ARM for assistance.

8.3 Test Interface Interconnections

The following diagram explains the interconnections:

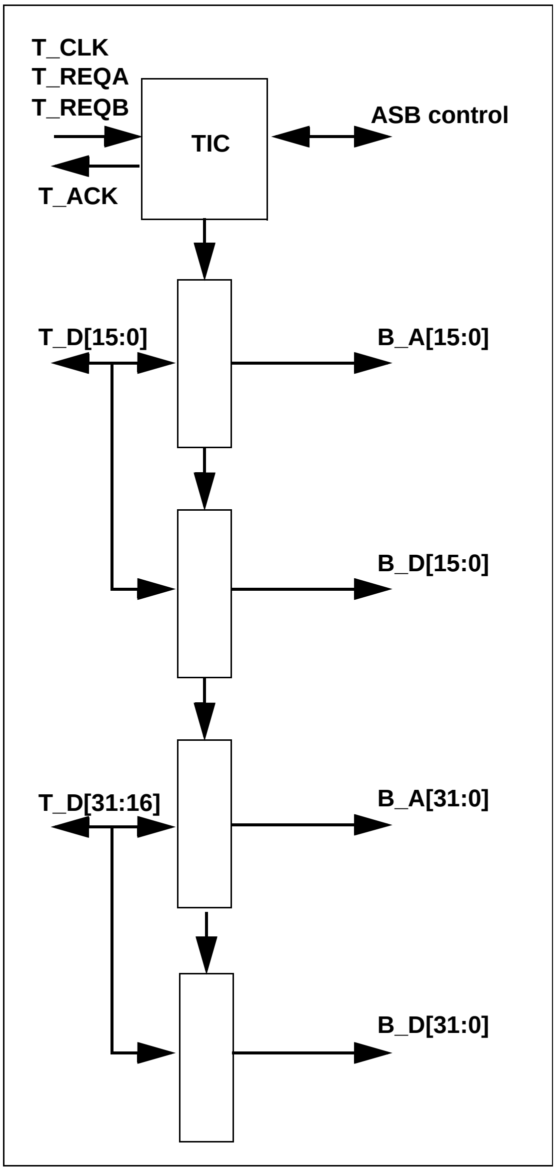


Figure 8-2: Test interface interconnections

9 Programming the APB FPGA

This chapter describes how to program the *field programmable gate array* (*FPGA*) on the ARM Development Board.

Programming the APB FPGA

9.1 Introduction

The ARM Development Board has one Xilinx 4000 series *field programmable gate array* (*FPGA*).

The FPGA is an array of *configurable logic blocks* (*CLBs*) and *in/out blocks* (*IOBs*). The interconnection between CLBs and IOBs, and their function is configured by programming SRAM cells.

Programming typically occurs on power-up and takes a few milliseconds. On power-up, the FPGA reads its mode pins (**MODE[2:0]**). These determine how the internal SRAM is to be programmed.

9.1.1 APB slaves and standard peripherals

The FPGA is used to implement a number of APB slaves. These are as follows:

- interrupt controller (11 IRQ sources and one FIQ source)
- two 16-bit counter/timers with 8 bit prescalers
- reset and pause (standby) controller

This is a set of standard peripherals common to most AMBA systems. These peripherals are mapped into the APB memory area, which is any address between 0x0A000000 and 0x0BFFFFFF.

Further information about these peripherals can be found in the *Reference Peripherals Specification (ARM DDI 0062)*.

The base addresses shown in ***Table 9-1: Base addresses*** have been implemented on the board:

Address	Name
0x0A000000	Interrupt Controller base (ICBase)
0x0A800000	Counter Timer base (CTBase)
0x0B000000	Reset and Pause Controller base (RPCBase)
0x0B800000	Expansion (spare for user functions)

Table 9-1: Base addresses

For a full system memory map, please refer to the *Target Development System User Guide (ARM DUI 0061)*.

9.2 Interrupt Controller

The interrupt controller differs from the standard design in that it has an extra register which allows the source of the FIQ interrupt to be selected as any of the 15 IRQ sources (1–15) or an external FIQ source pin.

9.2.1 Selecting the FIQ source

To select the FIQ source, write to the following 4-bit register:

```
ICBase + 0x114    r/w    FIQ_source_register
```

Program this register with any value from 0x0 to 0xF.

Source 0 is the external FIQ source which is an active low, level-sensitive input on the ASB and APB expansion connectors. This input is pulled up on the ARM Development Board.

9.2.2 Bit allocation

The table on the right shows the bit assignment in the IRQ interrupt controller:

Bit	Interrupt Source
0	Unused
1	Soft interrupt
2	COMMRX from processor
3	COMMTX from processor
4	Timer 1 (internal)
5	Timer 2 (internal)
6	PC card slot A
7	PC card slot B
8	Serial port A
9	Serial port B
10	Parallel port
11	ASB expansion 0
12	ASB expansion 1
13	APB expansion 0
14	APB expansion 1
15	APB expansion 2

9.3 Using the APB FPGA in Your Own Designs

With care, the APB FPGA can be used to implement additional logic. This section describes how to reprogram the device. Note that this is not a trivial task. You should adopt a VHDL design methodology and use the default configuration as a starting point.

VHDL for the default configuration is available; details can be found in ***Appendix C, Summary of Programmable Devices***.

9.3.1 Configuring the FPGA

The mode pins of the FPGA can be individually set HIGH or LOW by inserting jumpers onto the MODE pins of link field (LK16). In the default configuration these pins are linked, pulling **MODE[2:0]** LOW and configuring the FPGA for serial master mode. In this mode, the FPGA reads its configuration from a serial PROM.

MODE[2:0]	Name	Comment
000	master serial	Default, use serial PROM
001	master parallel up	not available on this board
010	reserved	not available on this board
011	master parallel down	not available on this board
100	reserved	not available on this board
101	peripheral	not available on this board
110	reserved	not available on this board
111	slave serial	Use download cable

Table 9-2: Configuring the FPGA

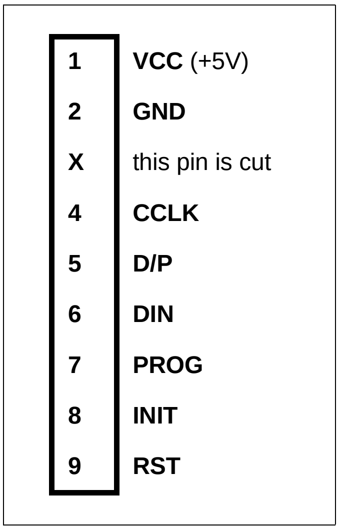
The serial PROM is an 8-pin DIL packaged device that can be programmed using a standard device programmer. The appropriate data file is generated using Xilinx proprietary tools.

9.3.2 Connecting the XChecker Cable

If the **MODE[2:0]** pins are not linked, the FPGA is in master-slave mode. In this mode the FPGA expects to be configured using the XChecker download cable.

The XChecker cable must be connected to the special 9-pin header provided on the ARM Development Board. The individual wires of the XChecker cable are labelled to aid the connection.

The pins are shown in ***Figure 9-1: Download cable pin connections***:



1	**VCC** (+5V)
2	**GND**
X	this pin is cut
4	**CCLK**
5	**D/P**
6	**DIN**
7	**PROG**
8	**INIT**
9	**RST**

Figure 9-1: Download cable pin connections

Preparing a Bit File

To configure an FPGA using the XChecker cable, a configuration bit file is required. This is generated by the Xilinx XACT place and route tools using the `makebits` program. This program can be run on any appropriate logic cell array (`*.lca`) file.

After you place and route the Xilinx tools, write a file with a `*.lca` extension. To generate a bit file type:

```
makebits -t filename.lca
```

where:

`-t` ties down unused interconnect internally

This generates a file called *`filename`*`.bit`.

9.3.3 Downloading a Configuration

To download a configuration:

1. Connect the XChecker download cable to a serial port on the host computer.
2. Connect the other end to the ARM Development Board (see ***9.3.2 Connecting the XChecker Cable*** on page 9-5).
3. Send the bit file to the FPGA using the XChecker download cable by typing the following command on the host machine:

   ```
   xchecker filename.bit
   ```
4. Follow the on screen prompts. If the program returns an error, check the power supply to the board and the integrity of the connectors.

When the download is successful the xchecker program reports:

```
DONE signal went high
```

and the LED marked “FPGA OK” lights up because it is connected to the FPGA **DONE** signal.

9.3.4 Configuration Using a Serial PROM

When a design has been tested—usually using the download cable—the configuration can be programmed into a PROM. A PROM inserted into the board initializes the FPGA on power-up.

To generate a PROM data file suitable for programming a device, use the Xilinx `makeprom` program. This is a graphical tool that allows PROM files to be generated in a variety of formats.

1. Select the format (for example, MCS) and PROM size required.
2. Using the menus, load the bit file from address 0 upwards. This can then be saved to disk.
3. Using a device programmer, load the PROM data file into memory. The serial PROMs required are Xilinx parts XC17128D–PD8C. These are 5V devices in an 8-pin DIL package.

Note *This part has a programmable reset polarity bit. Various device programmers handle programming of this bit in different ways. Some manual intervention is required to ensure that the device has active low reset polarity. When these bits have been written to, the device can be programmed.*

4. Program the PROM and insert it into the 8-pin DIL socket (U24) provided.
5. Ensure that the **MODE[2:0]** pins are connected using jumper sockets.
6. Power the board and observe the LED marked “FPGA OK”. If this LED does not light up, power down the board and recheck.

10 Programming the MACH and PAL Devices

This chapter briefly describes the methods for programming the MACH and PAL devices.

Programming the MACH and PAL Devices

10.1 Reprogramming a Device

Much of the programmable logic on the ARM Development Board is implemented in AMD MACH and PALCE type devices. These devices are based on electrically erasable (EE) technology so they can be reprogrammed to add or change functionality if required. This should only be attempted if you have a full and detailed understanding of the design.

There are many and varied reasons why you might want to make modifications. For example:

- One reason why you might want to do this is to change the address map. In this case, you program the decoder.
- Alternatively, you might want to change the priority of bus masters, in which case you alter the arbiter.
- If you want to fit slower SRAM, you need to modify the SRAM controller to prevent unsuitable switch positions crashing the system.

ARM can assist with making such modifications. See ***Appendix C, Summary of Programmable Devices*** for information on all the PLD filenames and how to obtain them from ARM.

10.1.1 Design files

Reprogramming a MACH or PALCE device on the ARM Development Board requires a design file. This design file comprises a list of logic equations in a hardware description language (HDL). Various tools are available for this, such as:

ABEL	from Data I/O
PLDesigner	from Minc
PALASM	from AMD

All the designs for the ARM Development Board were completed using PALASM which is a low-cost proprietary tool from the device vendor AMD.

If you use a different PLD design tool then it should not be too difficult to modify the PALASM description to suit your front end. No special constructs have been used, so the process should be straightforward.

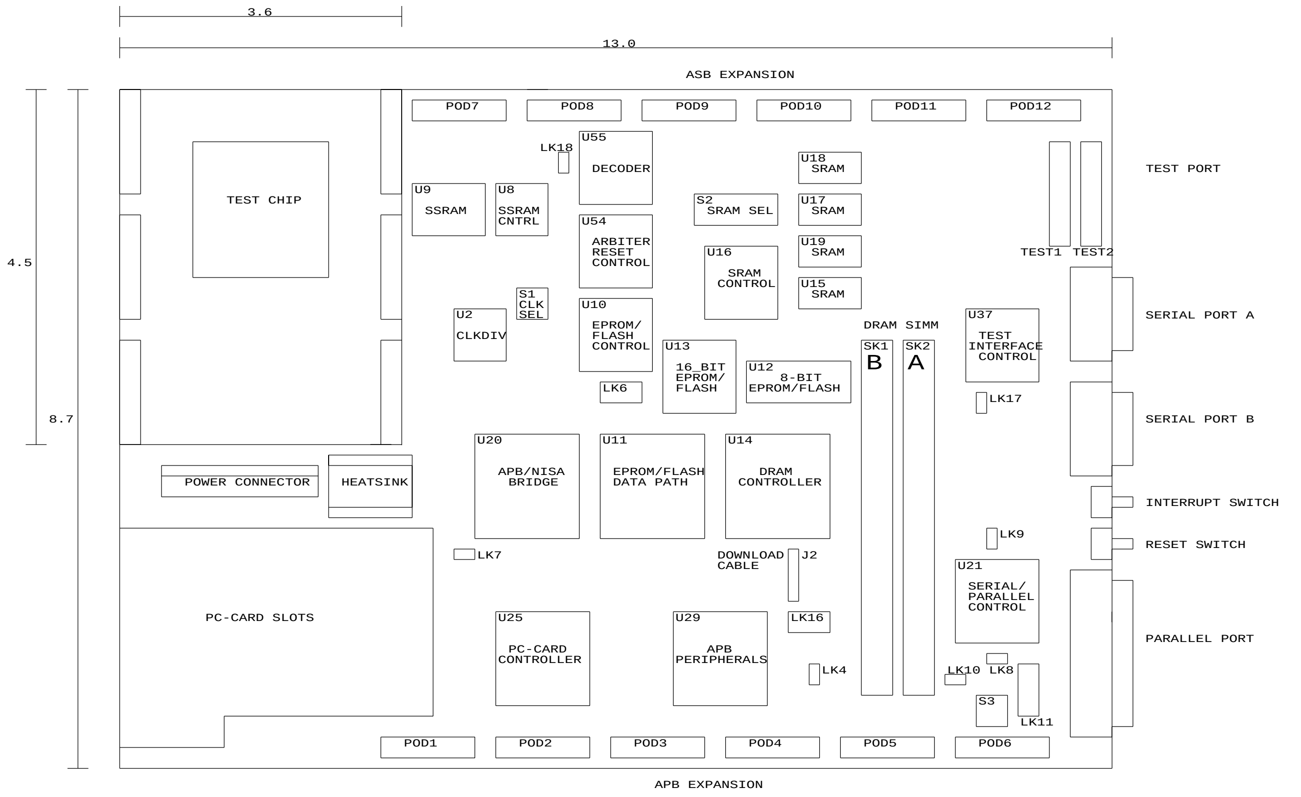
10.1.2 Preparing a .JED File and Programming a Device

A tool such as PALASM turns a design written in an HDL into a fuse map. This fuse map is described in a JEDEC standard file format. The file, usually called `filename.jed`, can be downloaded into the target device using a logic programmer.

A Board Schematics

The board design comprises 22 schematics as listed below.

A.1 Card Outline Drawing



SWITCHES

S1 CLOCK FREQUENCY SELECT
S2 SRAM SELECT
S3 PARALLEL PORT SWITCHES

2-PIN LINKS

LK4 BIGEND
LK7 P_STB WIDTH
LK8 INTERRUPT TYPE
LK9 DIRECTION
LK10 ENABLE INTERRUPT
LK17 USE TEST INTERFACE
LK18 REMAP

LINK FIELDS

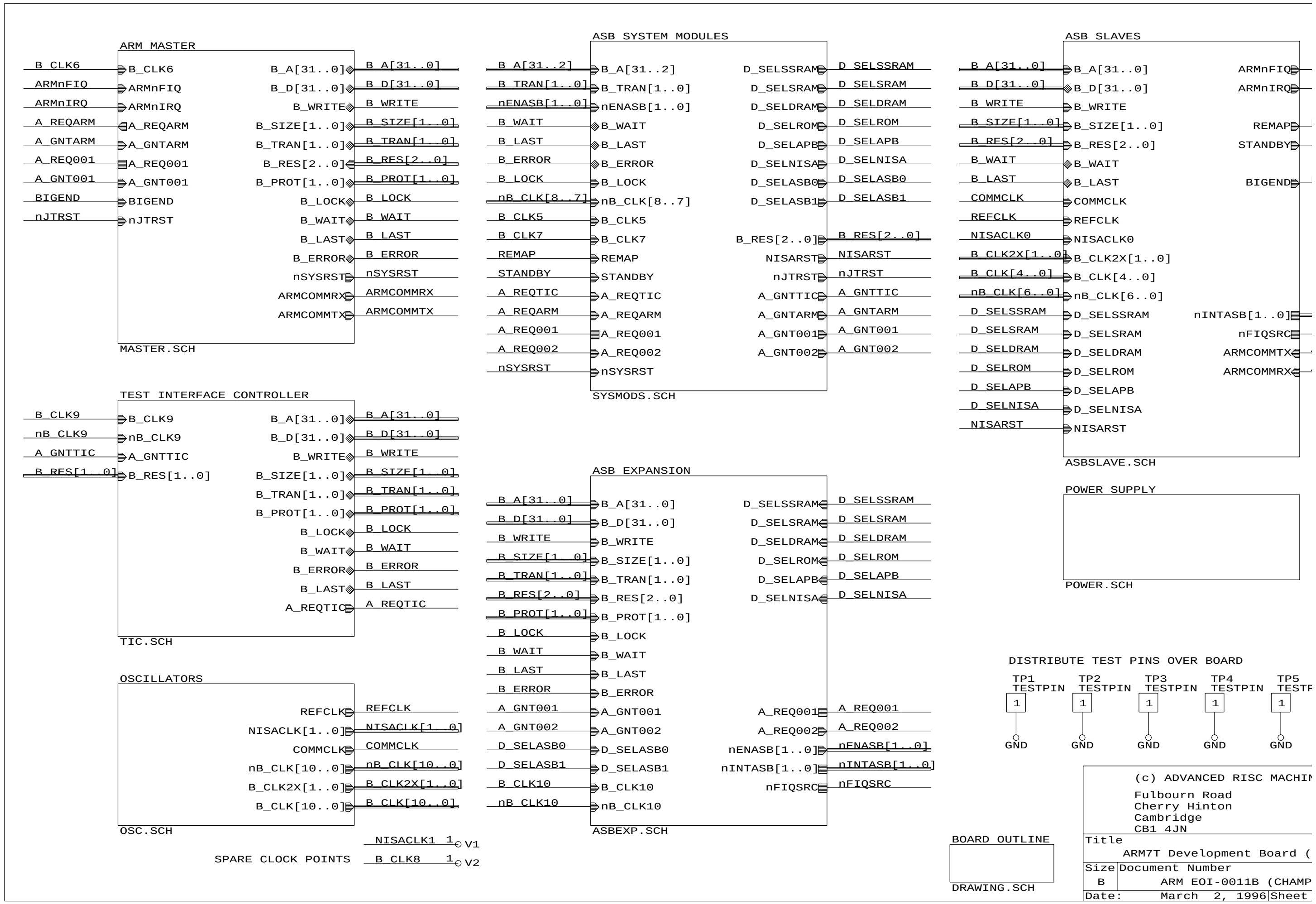
LK6 EPROM/FLASH SELECT
LK11 PARALLEL PORT LINKS
LK16 FPGA LINKS

ALL DIMENSIONS IN INCHES

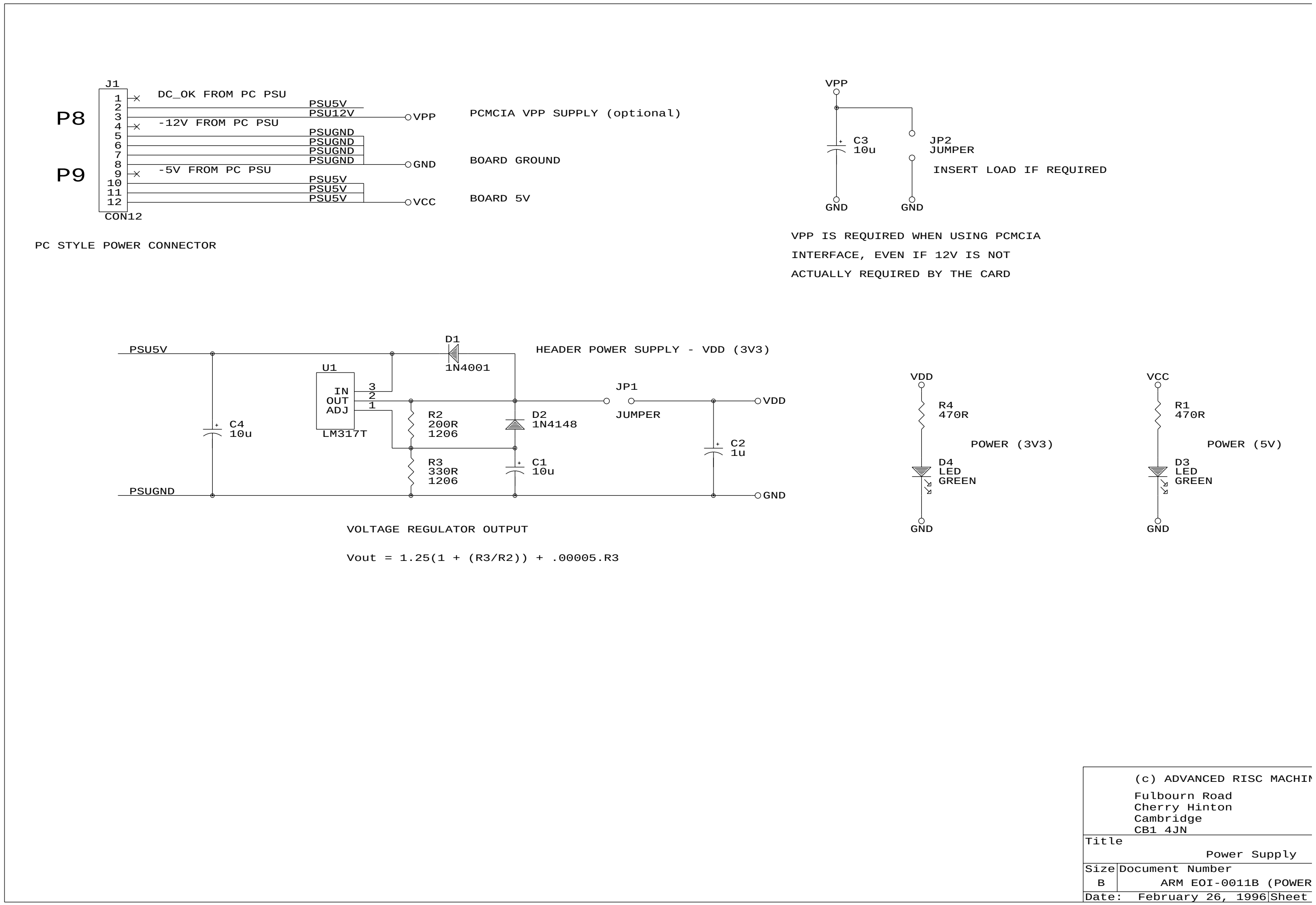
(C) ADVANCED RISC MAC
FULBOURN ROAD
CHERRY HINTON
CAMBRIDGE
CB1 4JN

Title		
	Card Outline Drawi	
Size	Document Number	
B	EOI-0011B (DRAWING.	
Date:	March 8, 1996	Sheet

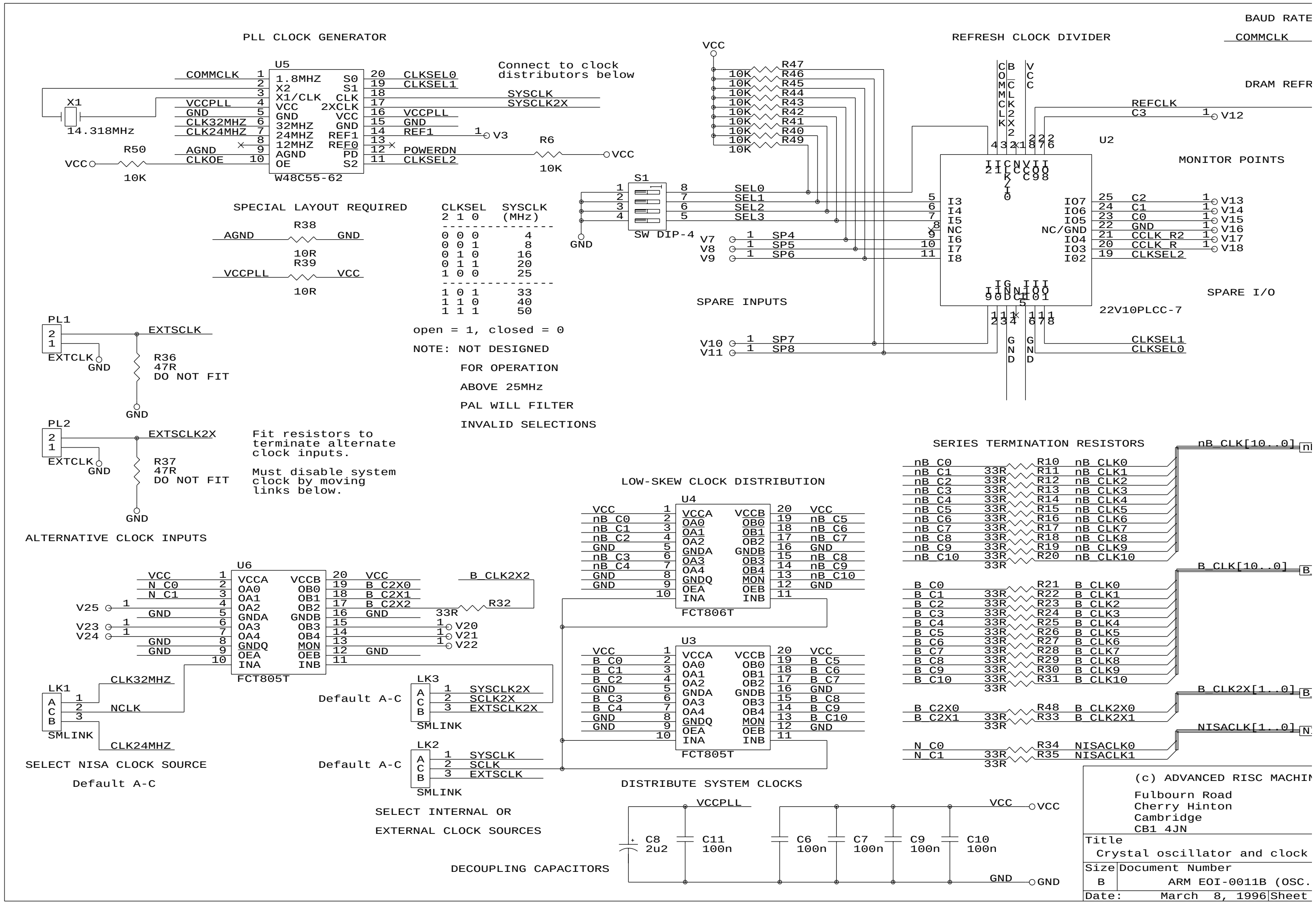
A.2 Top-level Diagram



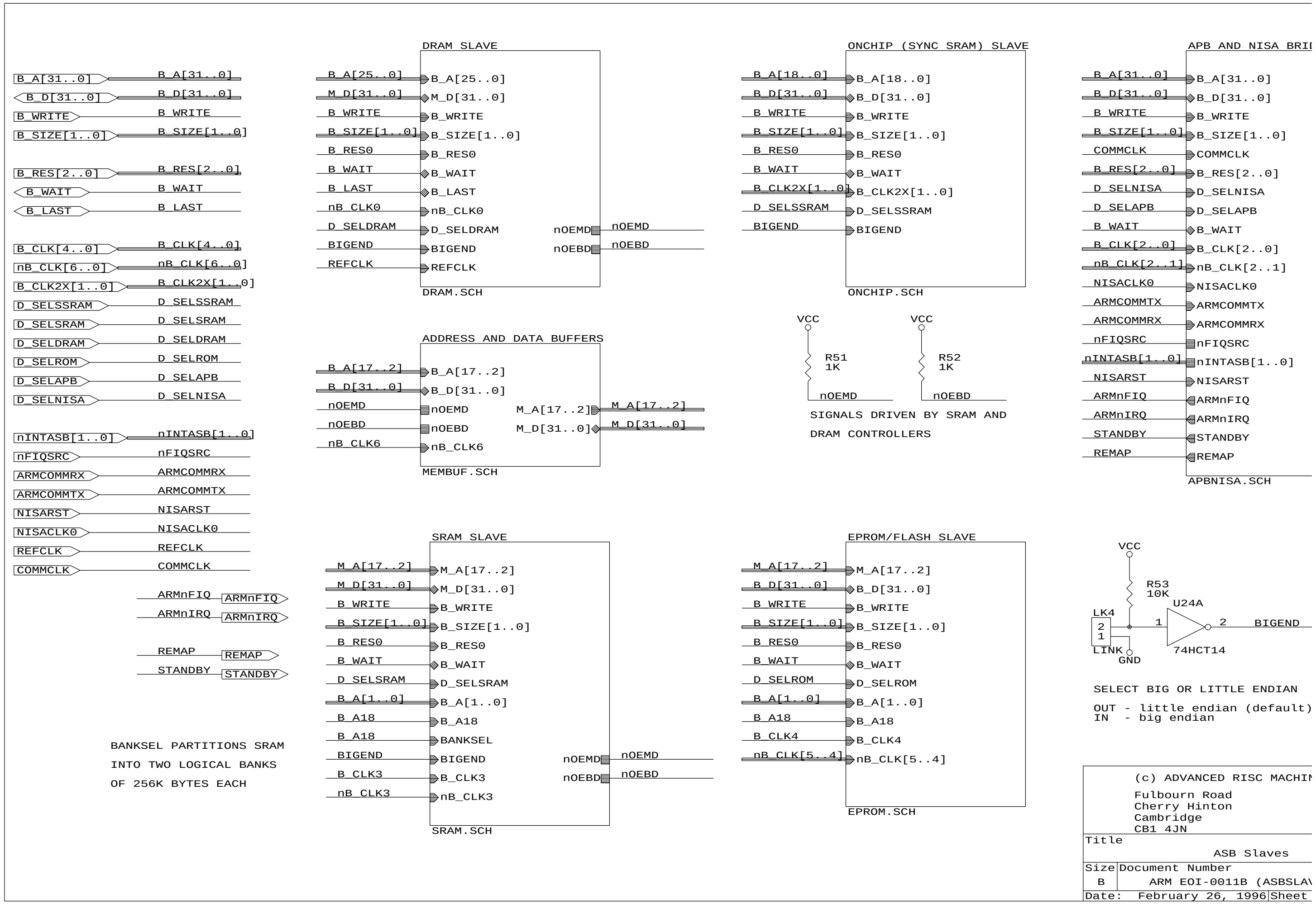
A.3 Power Supply



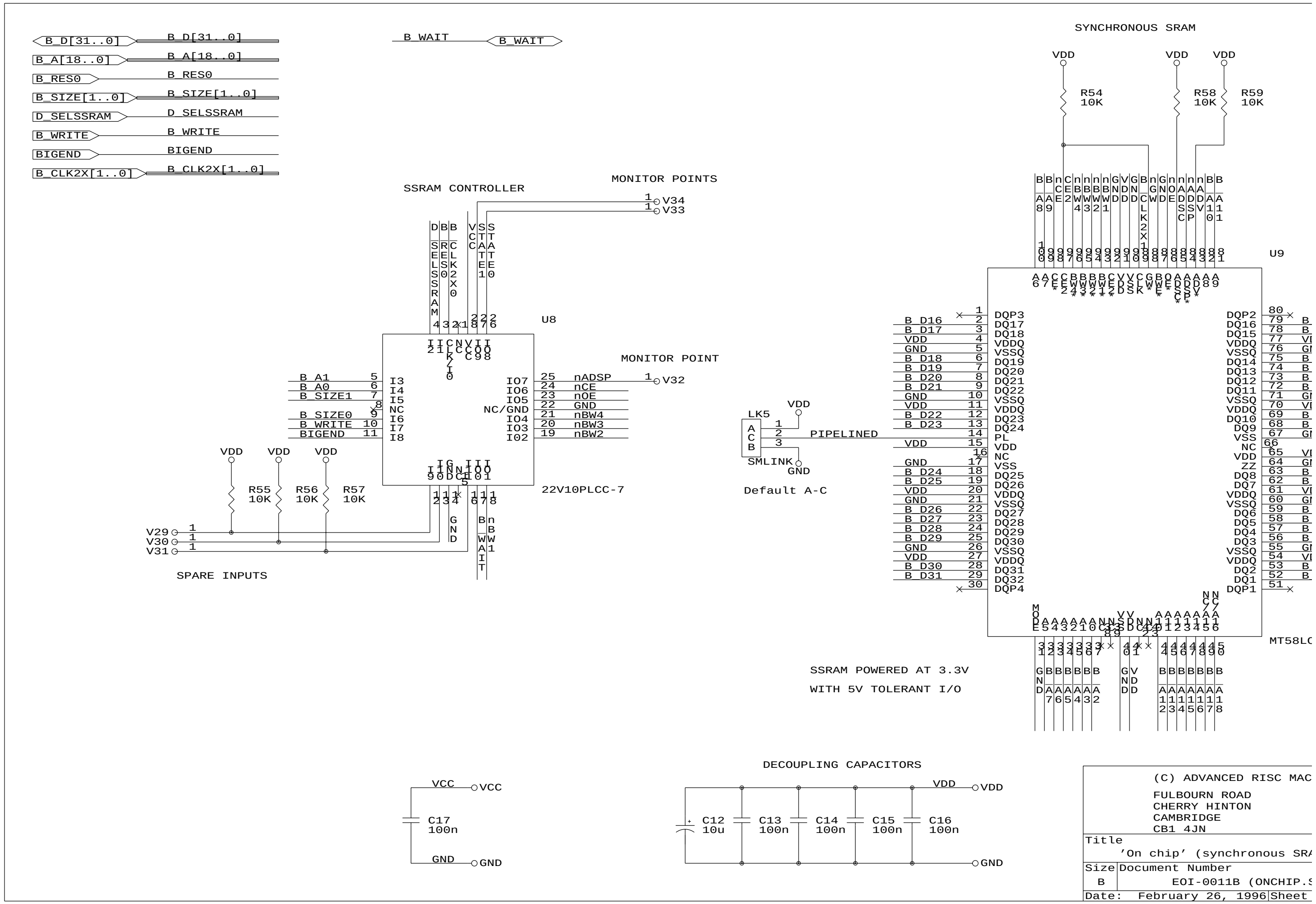
A.4 Crystal Oscillator and Clock Distribution



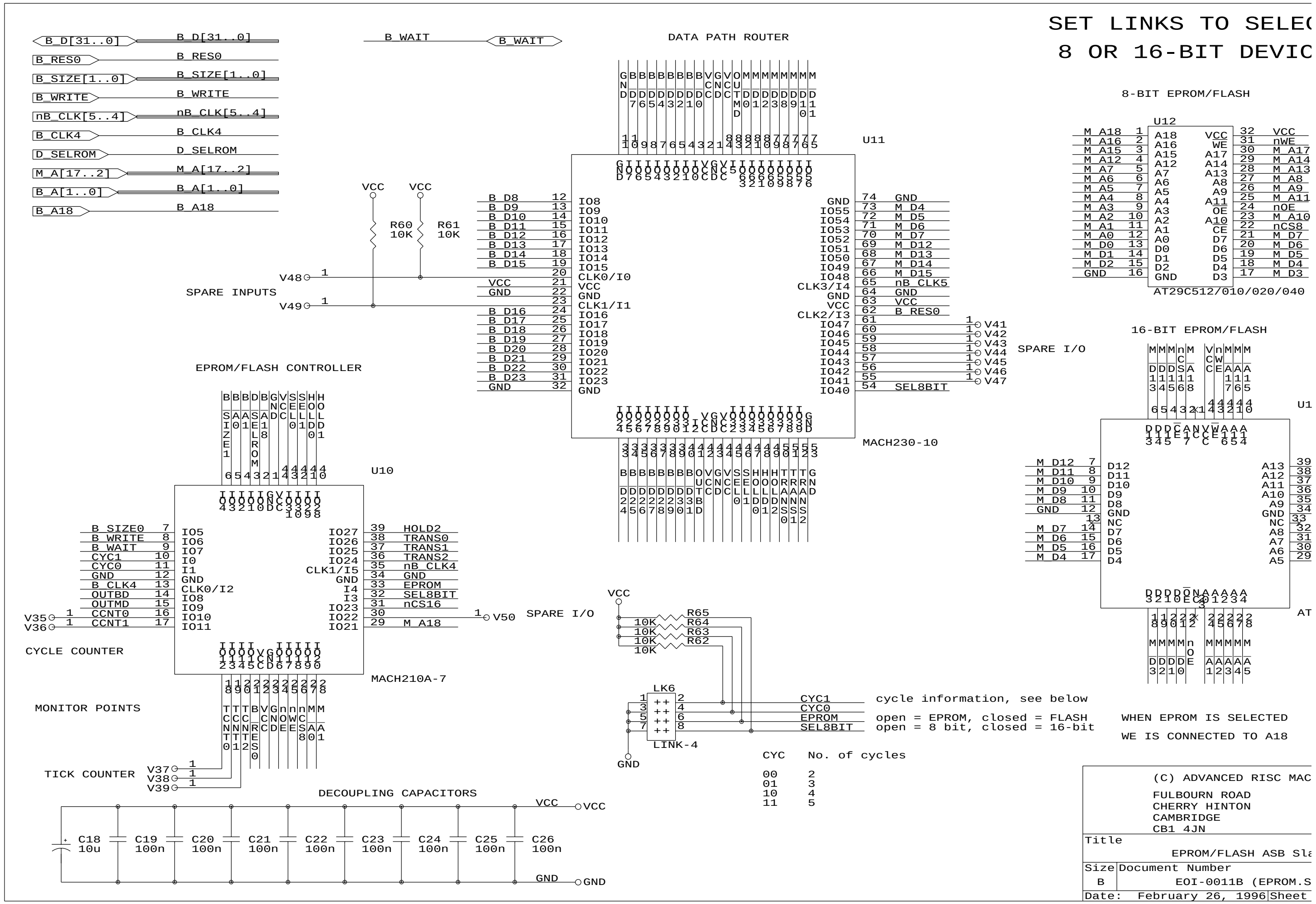
A.5 ASB Slaves



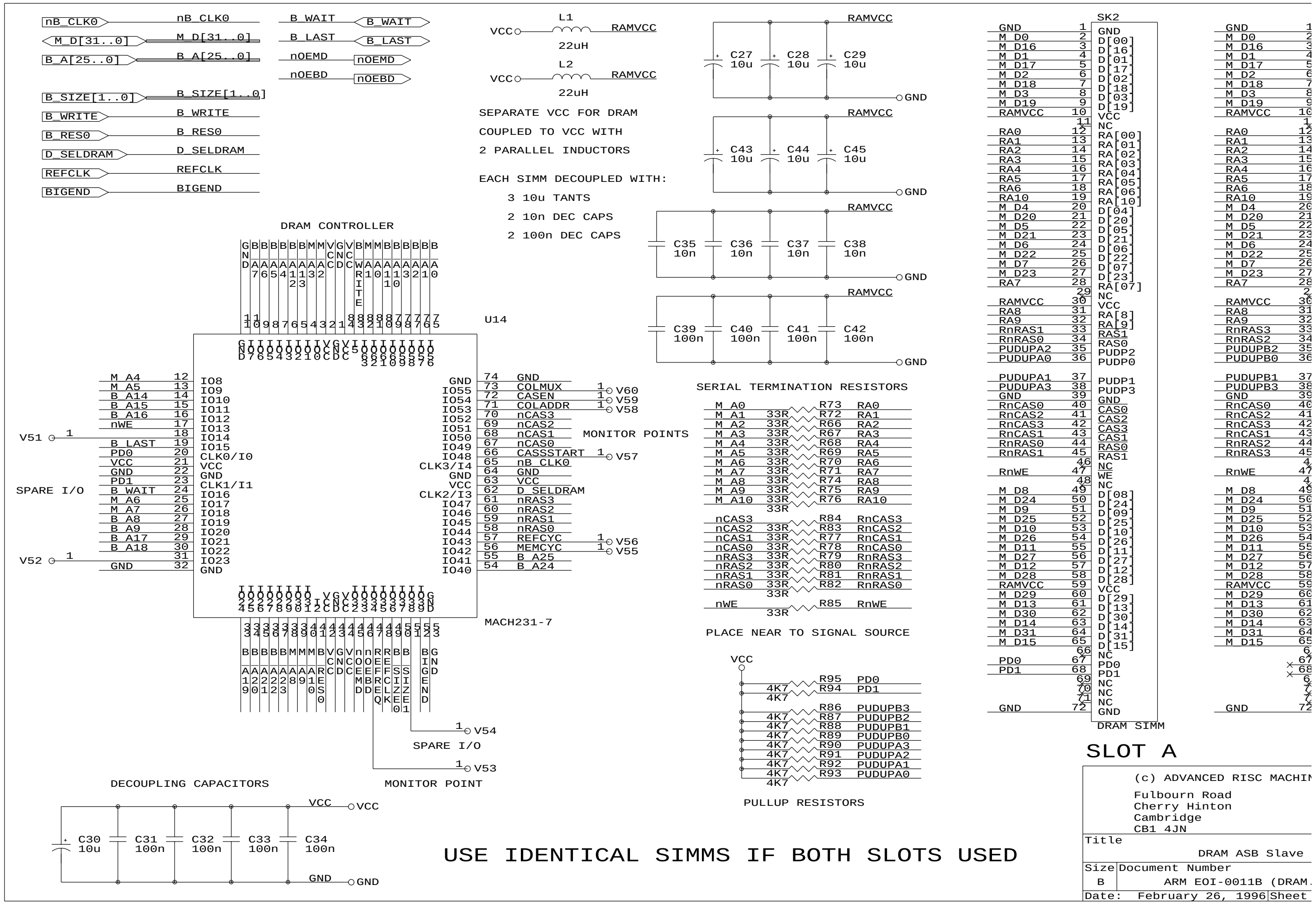
A.6 “On-chip” Memory (Synchronous SRAM)



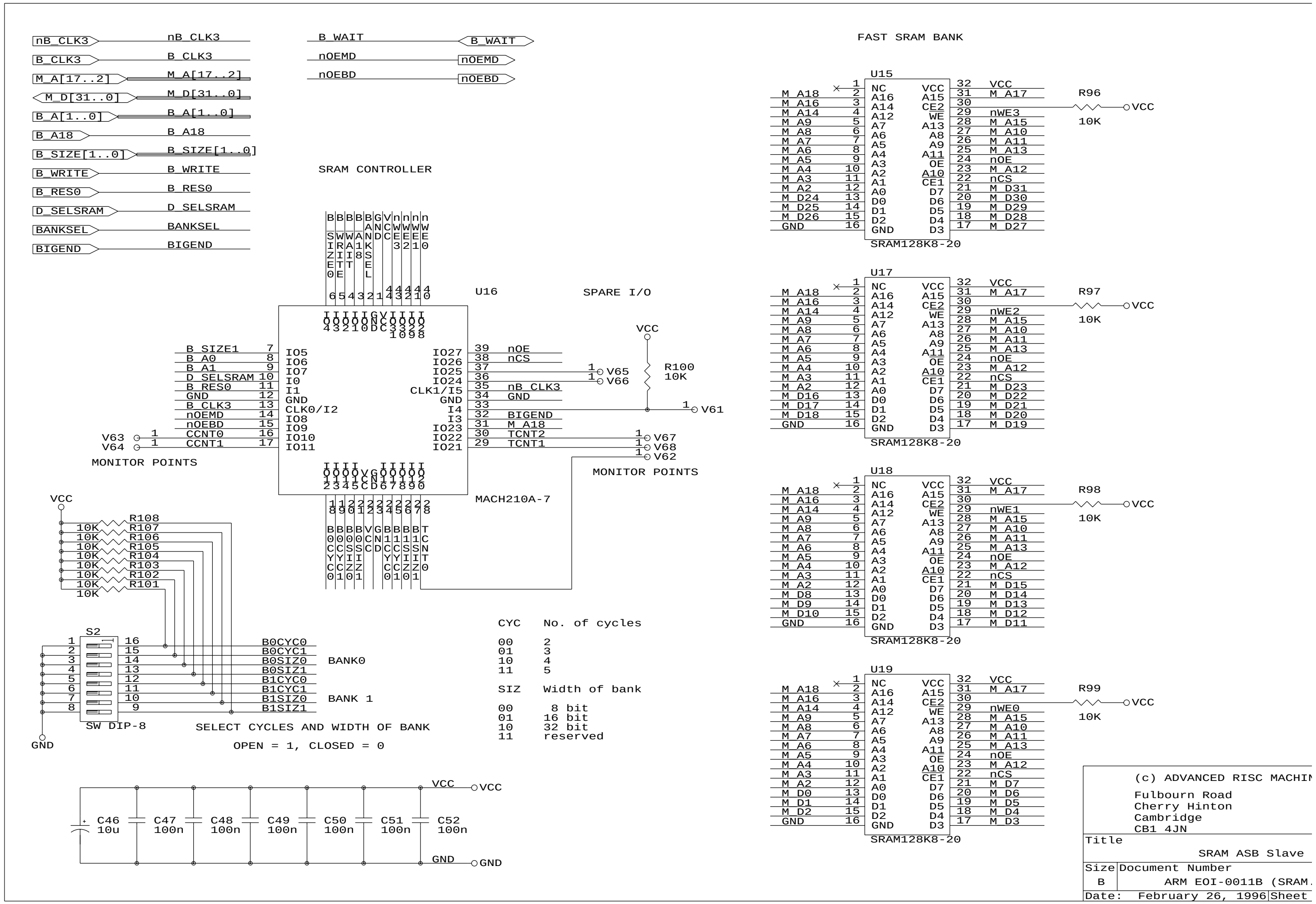
A.7 EPROM/FLASH ASB Slave



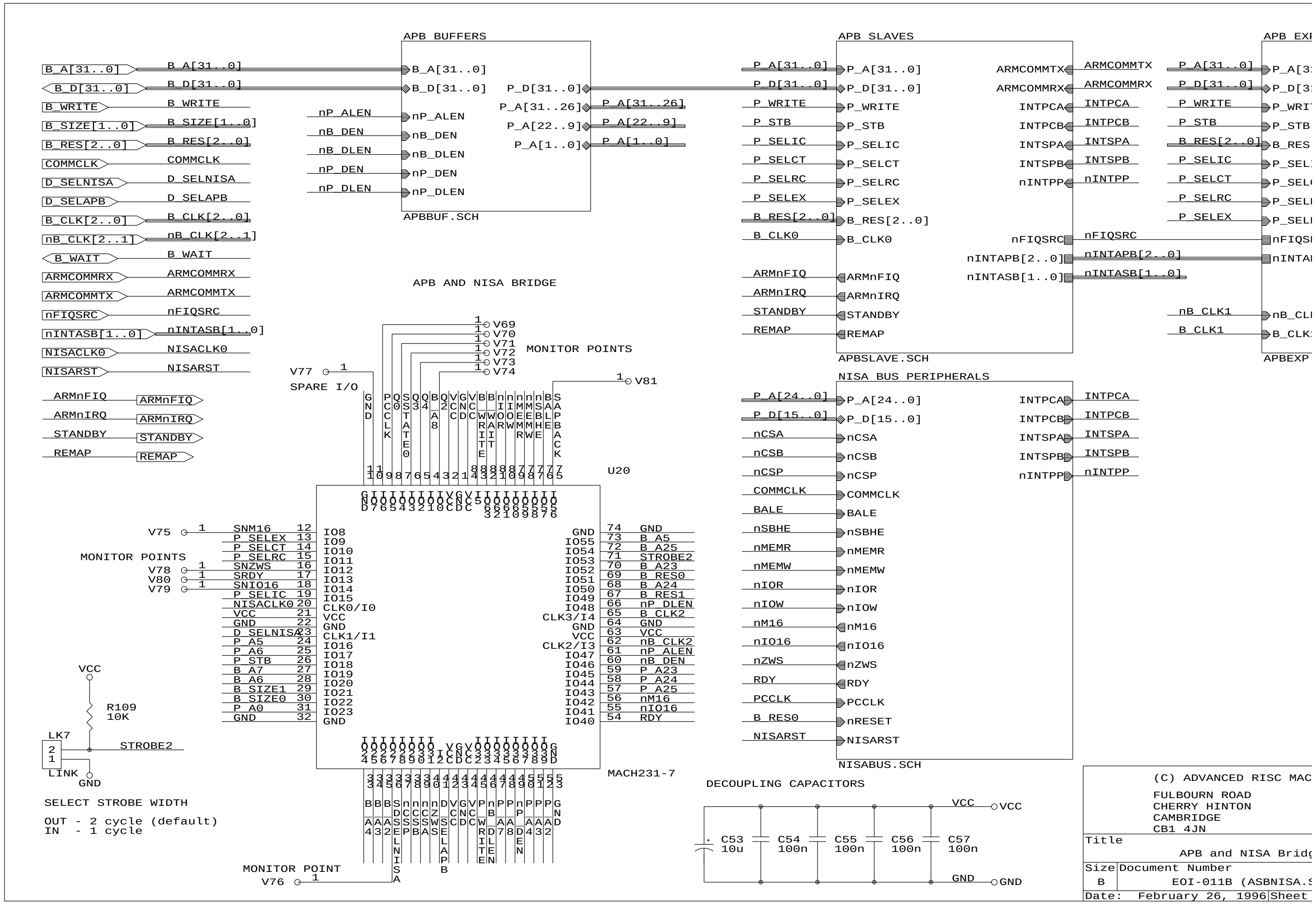
A.8 DRAM ASB Slave



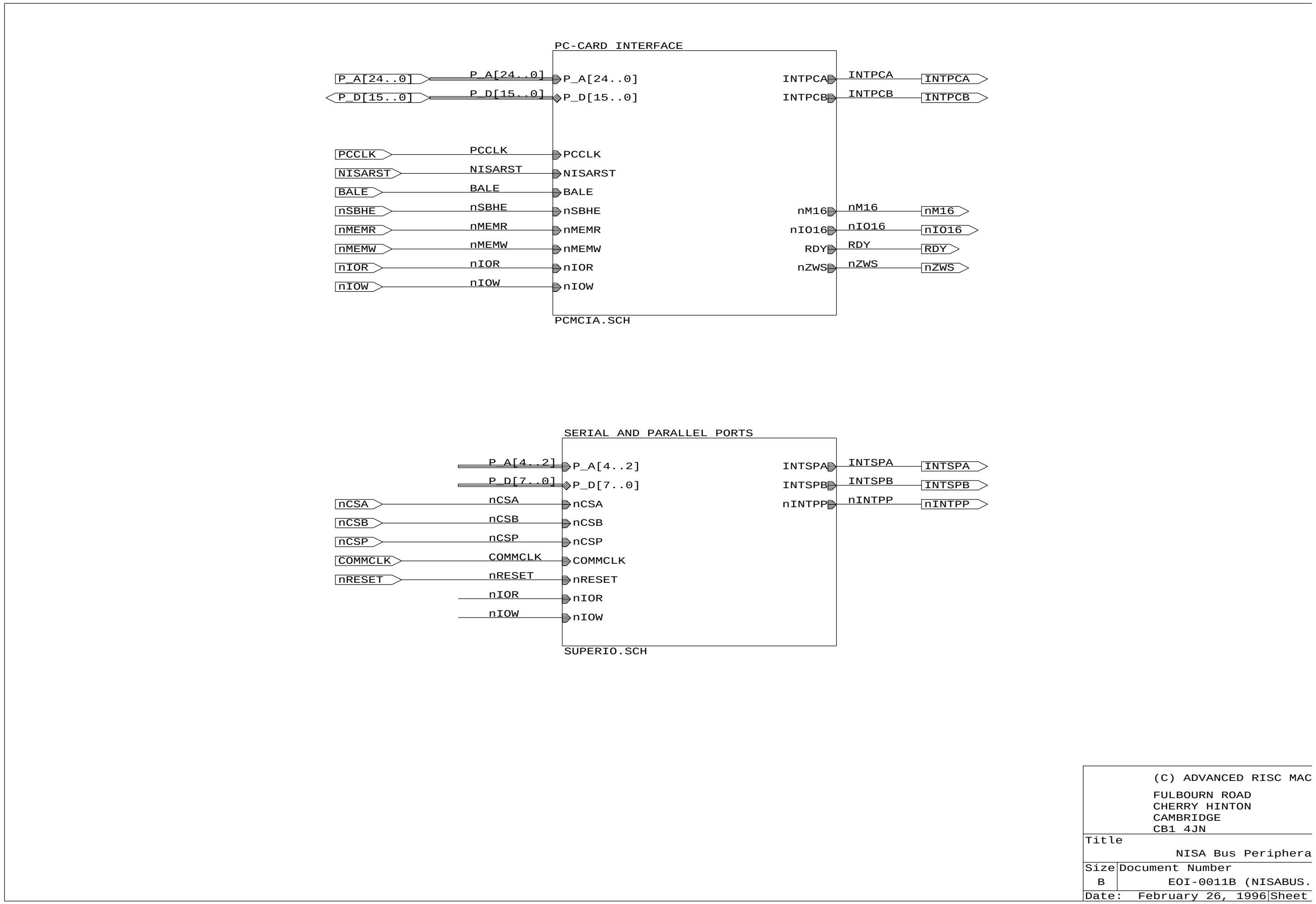
A.9 SRAM ASB Slave

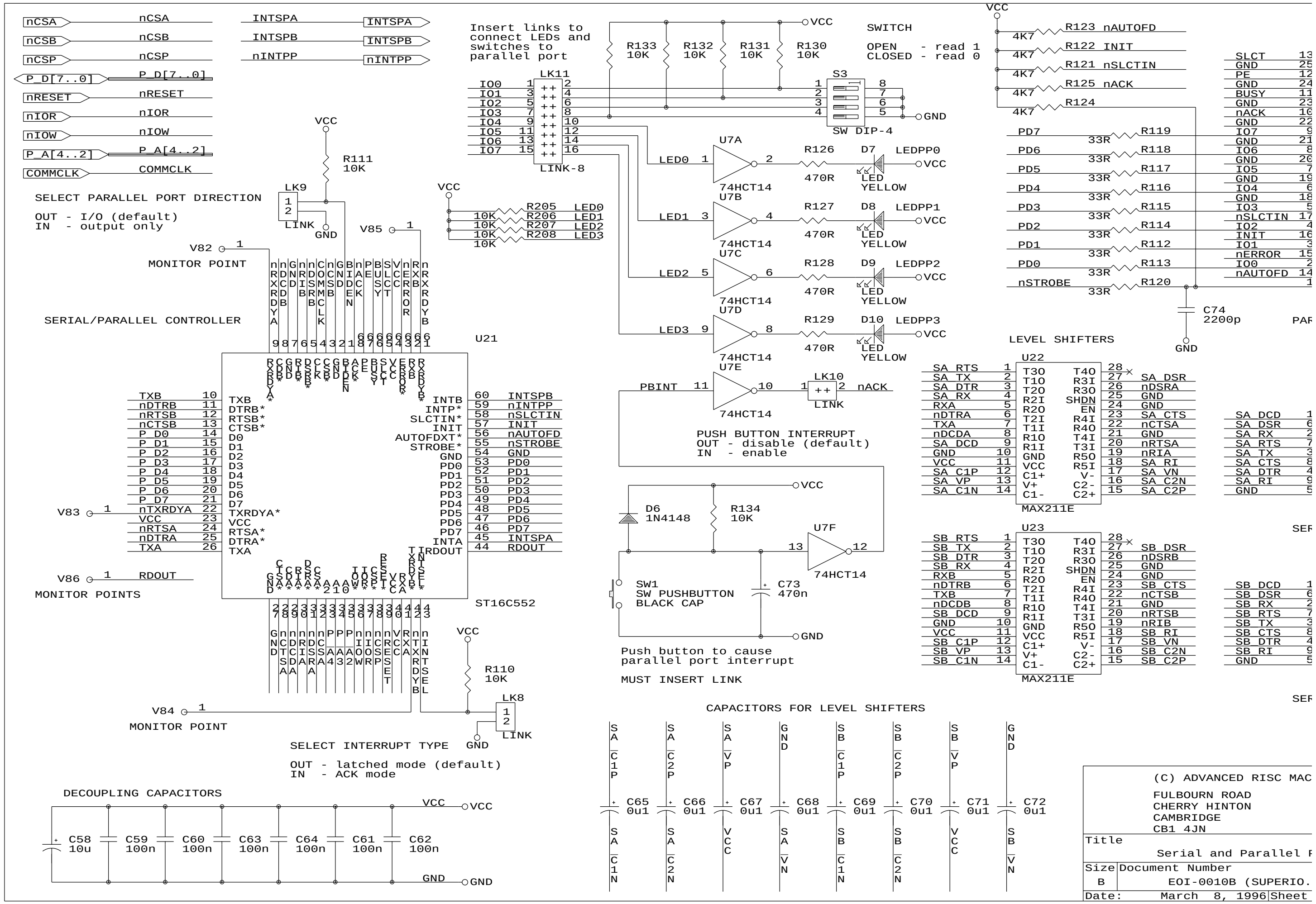


A.10 APB and NISA Bridge

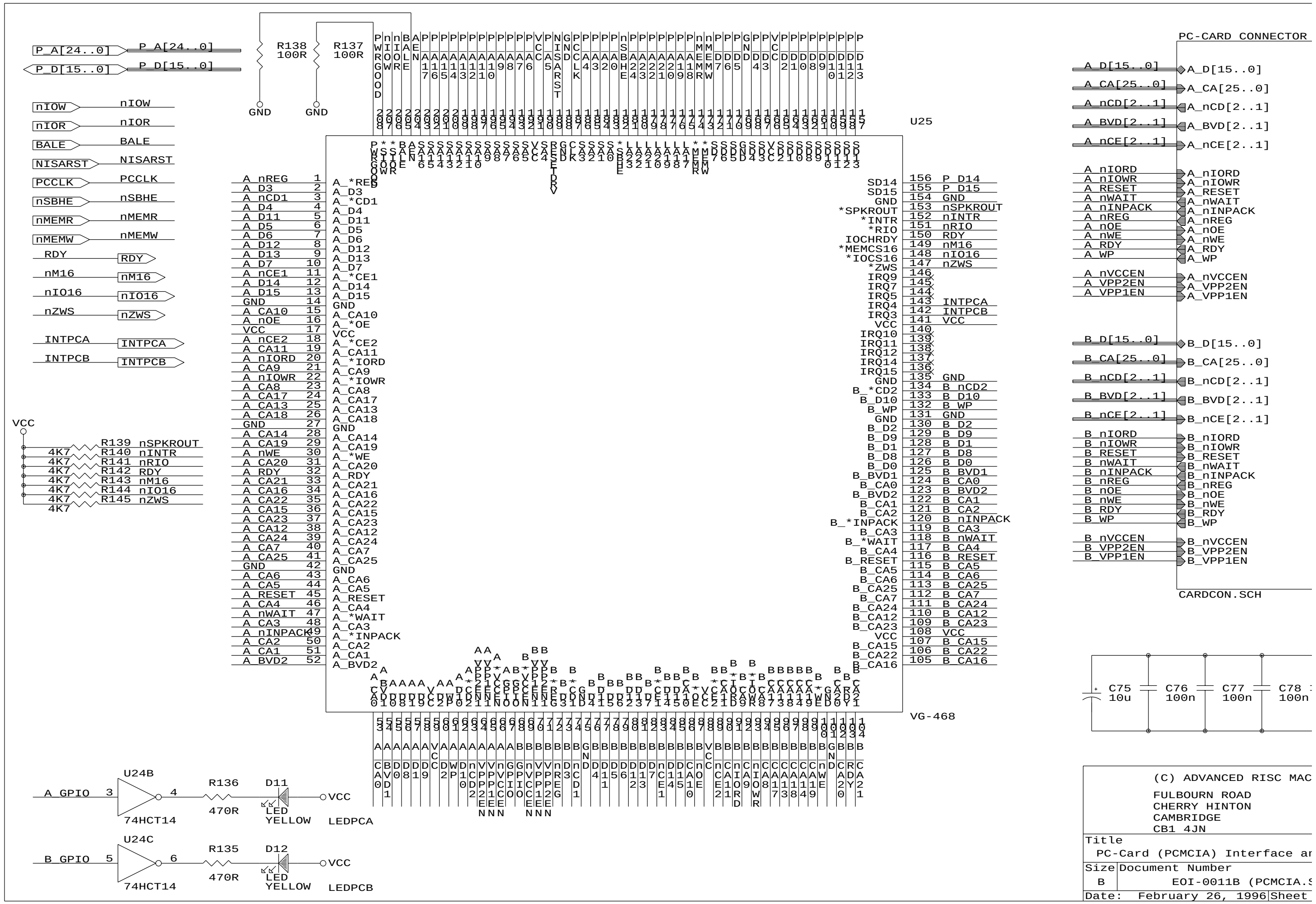


A.11 NISA Bus Peripherals

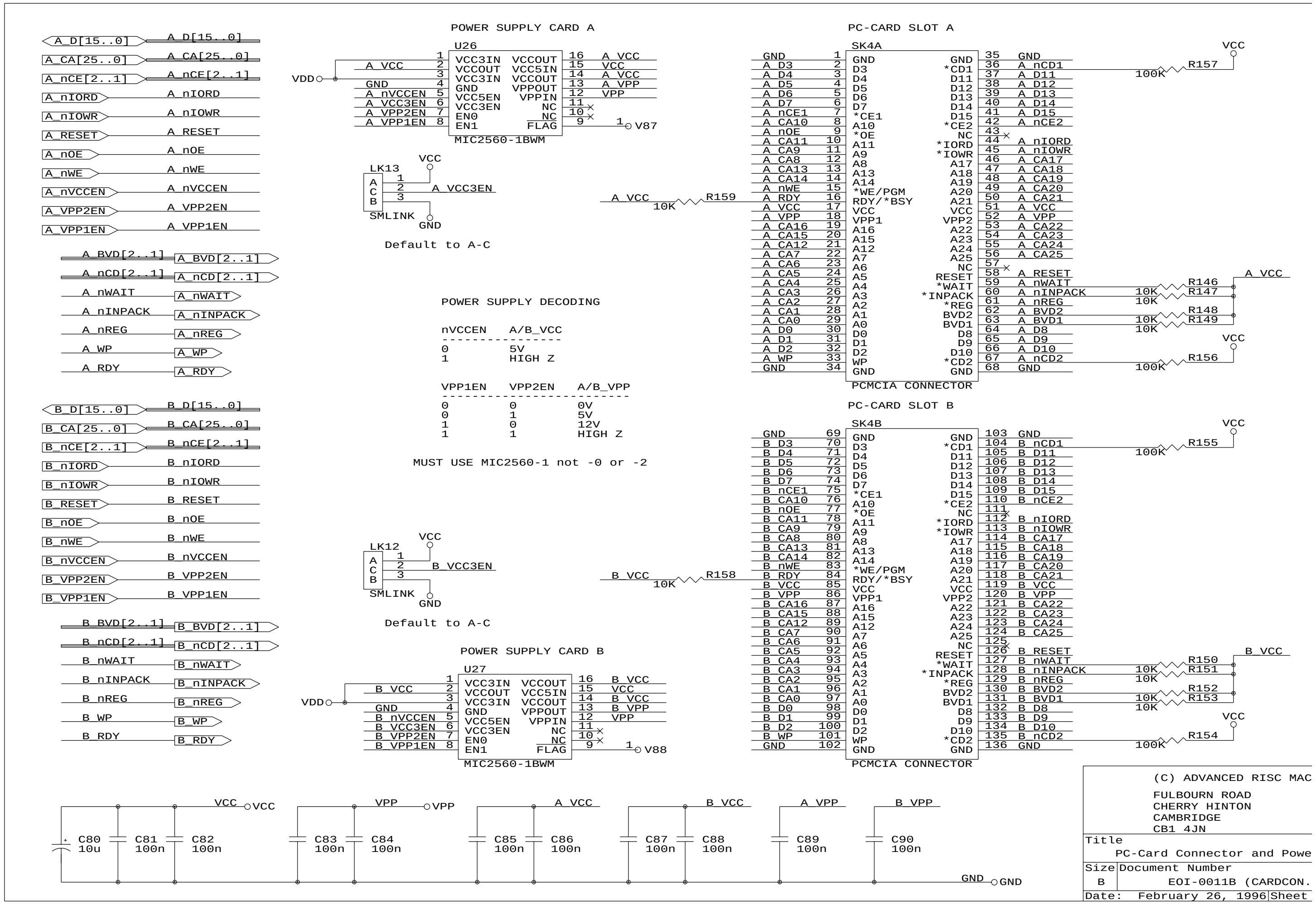



nCSA
nCSB
nCSP
P_D[7..0]
nRESET
nIOR
nIOW
P_A[4..2]
COMMCLK
INTSPA
INTSPB
nINTPP
Insert links to connect LEDs and switches to parallel port
VCC
R133 10K
R132 10K
R131 10K
R130 10K
SWITCH
OPEN - read 1
CLOSED - read 0
S3
SW DIP-4
GND
LK11
LINK-8
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
R111 10K
LK9
LINK
SELECT PARALLEL PORT DIRECTION
OUT - I/O (default)
IN - output only
V82
V85
MONITOR POINT
R205
R206
R207
R208
10K
LED0
LED1
LED2
LED3
U7A
U7B
U7C
U7D
U7E
U7F
74HCT14
R126
R127
R128
R129
470R
D7
D8
D9
D10
LEDPP0
LEDPP1
LEDPP2
LEDPP3
LED YELLOW
SERIAL/PARALLEL CONTROLLER
U21
ST16C552
TXB
nDTRB
nRTSB
nCTSB
P_D0
P_D1
P_D2
P_D3
P_D4
P_D5
P_D6
P_D7
nTXRDYA
VCC
nRTSA
nDTRA
TXA
V83
V86
RDOUT
MONITOR POINTS
DTRB*
RTSB*
CTSB*
TXRDYA*
RTSA*
DTRA*
INTB
INTP*
SLCTIN*
INIT
AUTOFDXT*
STROBE*
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
INTA
IRDOUT
INTSPB
nINTPP
nSLCTIN
nAUTOFD
nSTROBE
R110 10K
LK8
V84
SELECT INTERRUPT TYPE
OUT - latched mode (default)
IN - ACK mode
PBINT
LK10
nACK
PUSH BUTTON INTERRUPT
OUT - disable (default)
IN - enable
D6 1N4148
R134 10K
SW1
SW PUSHBUTTON
BLACK CAP
C73 470n
Push button to cause parallel port interrupt
MUST INSERT LINK
DECOUPLING CAPACITORS
C58 10u
C59 100n
C60 100n
C63 100n
C64 100n
C61 100n
C62 100n
CAPACITORS FOR LEVEL SHIFTERS
C65 0u1
C66 0u1
C67 0u1
C68 0u1
C69 0u1
C70 0u1
C71 0u1
C72 0u1
4K7
R123 nAUTOFD
R122 INIT
R121 nSLCTIN
R125 nACK
R124
33R
R119
R118
R117
R116
R115
R114
R112
R113
R120
nSTROBE
C74 2200p
LEVEL SHIFTERS
U22
U23
MAX211E
SA_RTS
SA_TX
SA_DTR
SA_RX
RXA
nDTRA
TXA
nDCDA
SA_DCD
SA_C1P
SA_VP
SA_C1N
SA_DSR
nDSRA
SA_CTS
nCTSA
nRTSA
nRIA
SA_RI
SA_VN
SA_C2N
SA_C2P
SB_RTS
SB_TX
SB_DTR
SB_RX
RXB
nDTRB
TXB
nDCDB
SB_DCD
SB_C1P
SB_VP
SB_C1N
SB_DSR
nDSRB
SB_CTS
nCTSB
nRTSB
nRIB
SB_RI
SB_VN
SB_C2N
SB_C2P
SLCT
PE
BUSY
nACK
nSLCTIN
INIT
nERROR
nAUTOFD
PAR
SER
(C) ADVANCED RISC MAC
FULBOURN ROAD
CHERRY HINTON
CAMBRIDGE
CB1 4JN
Title
Serial and Parallel P
Size Document Number
B
EOI-0010B (SUPERIO.
Date: March 8, 1996 Sheet

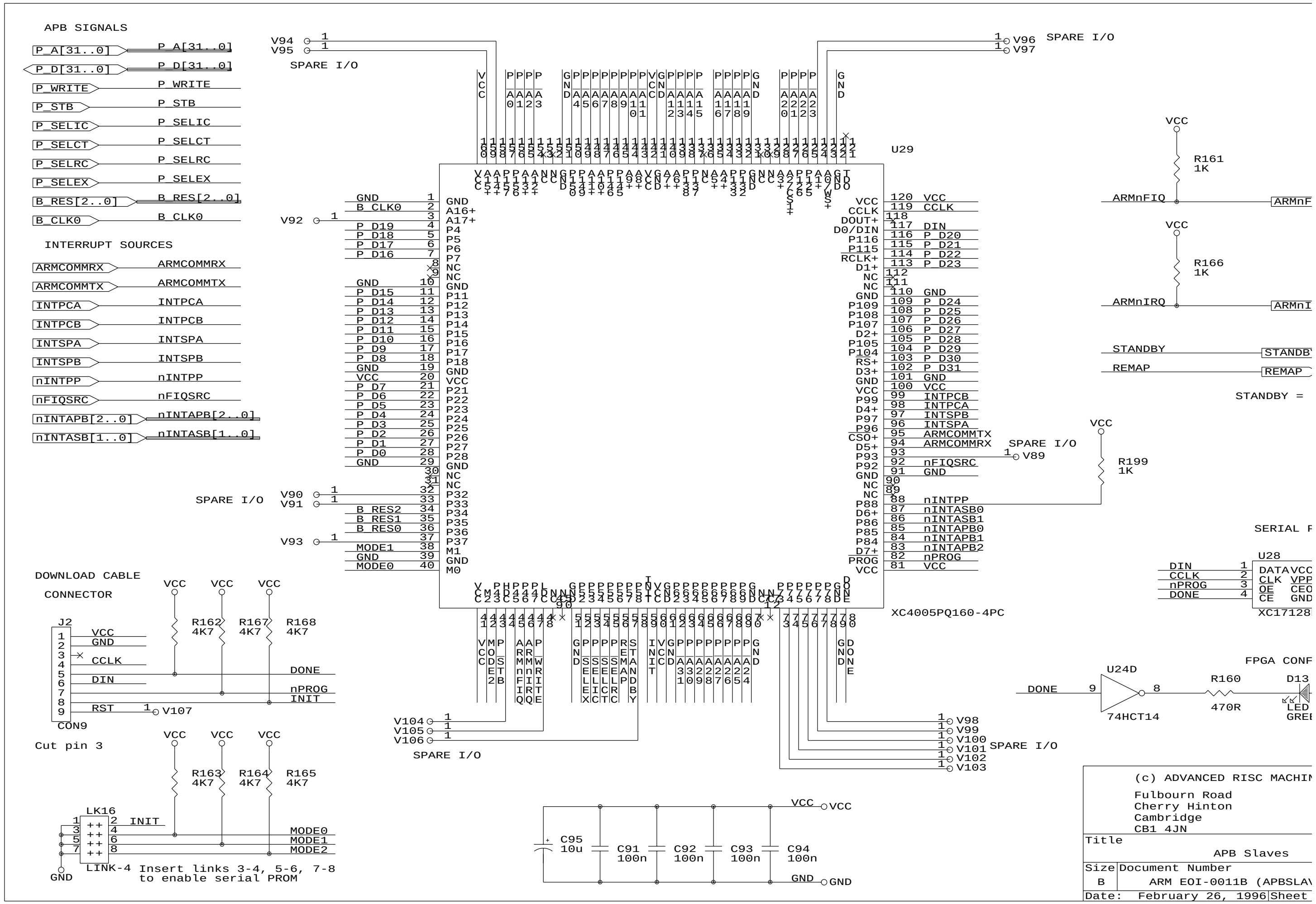
A.13 PC Card Interface



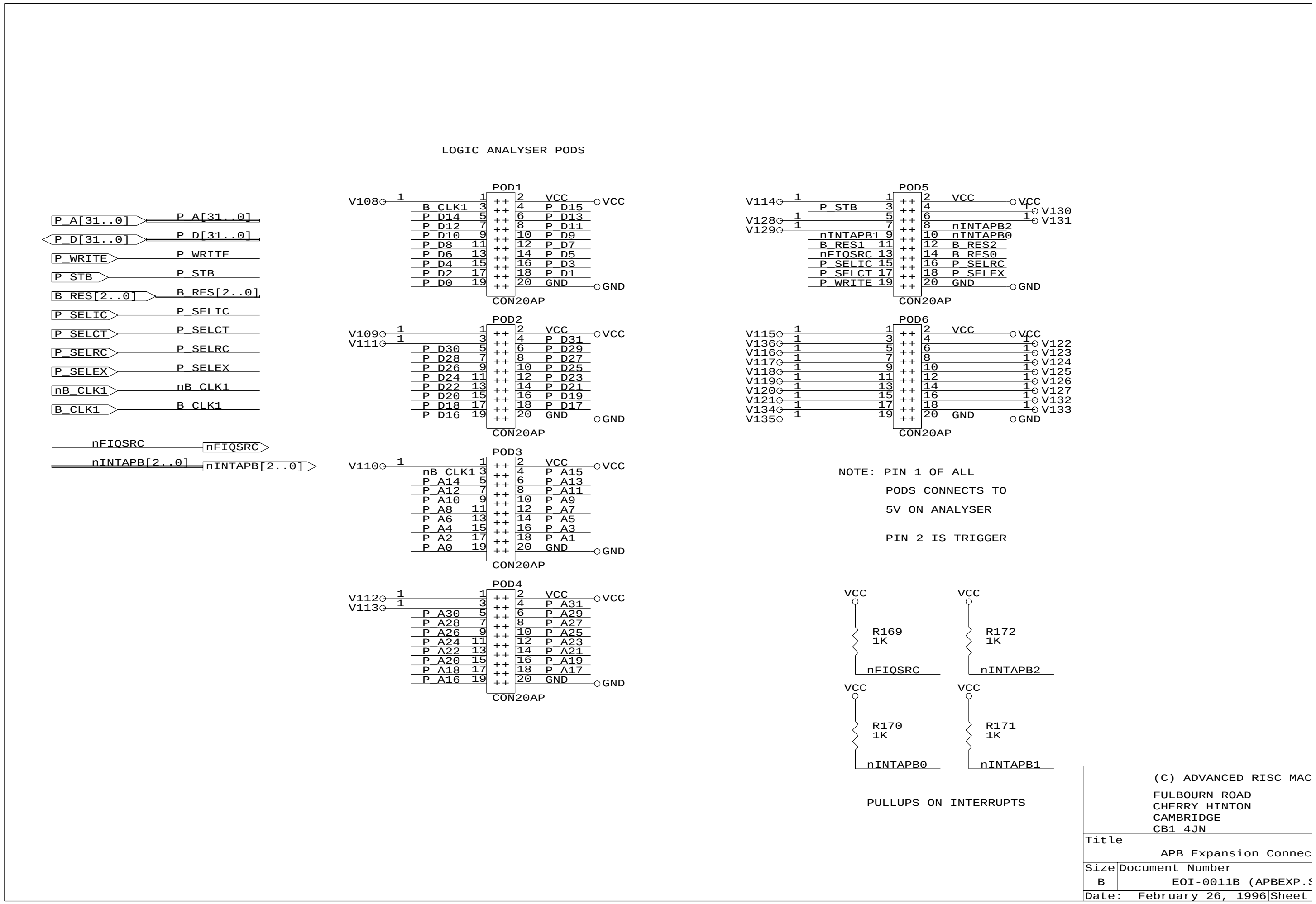
A.14 PC Card Connecters and Power Supply



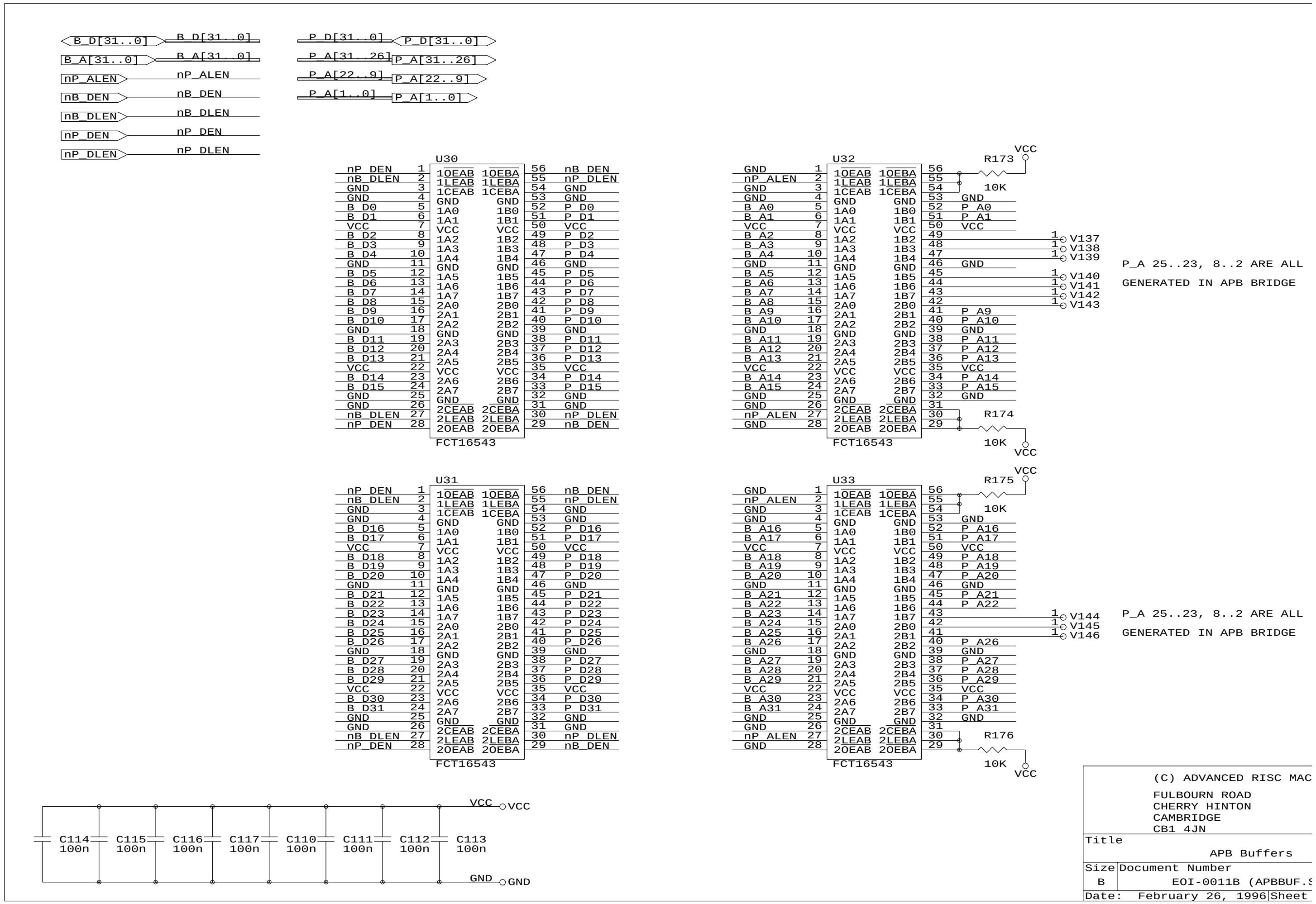
A.15 APB Slaves



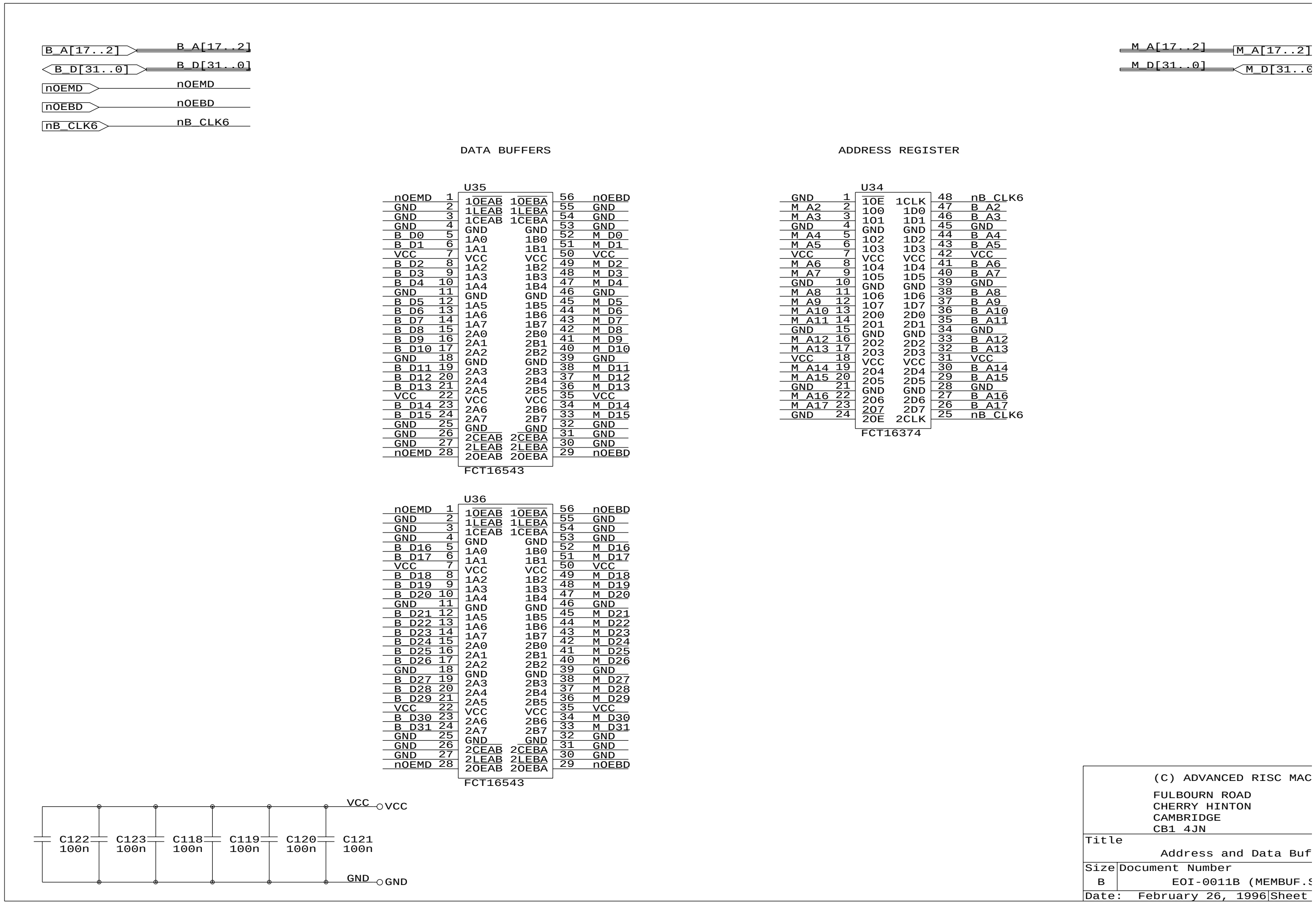
A.16 APB Expansion Connecters



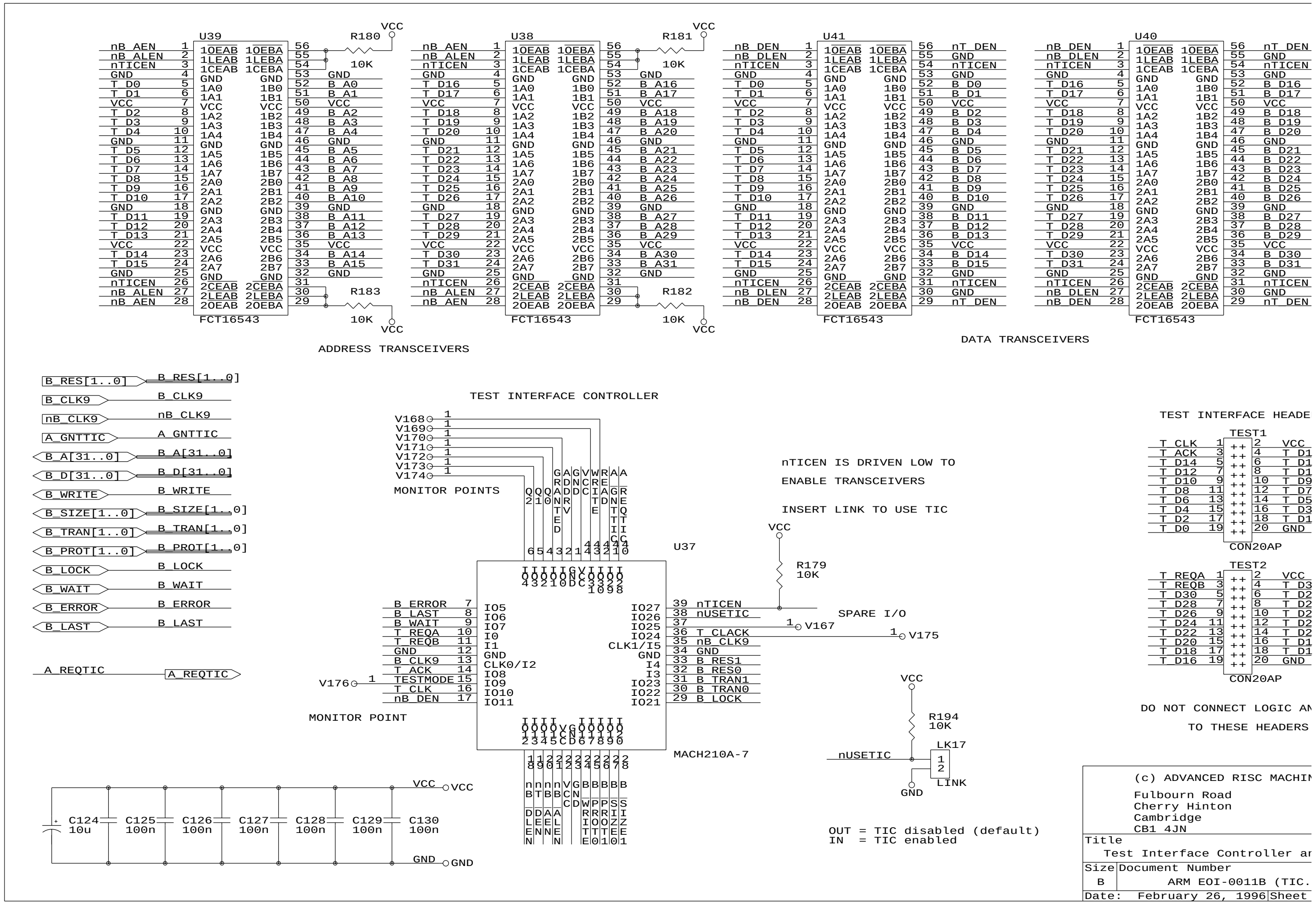
A.17 APB Buffers



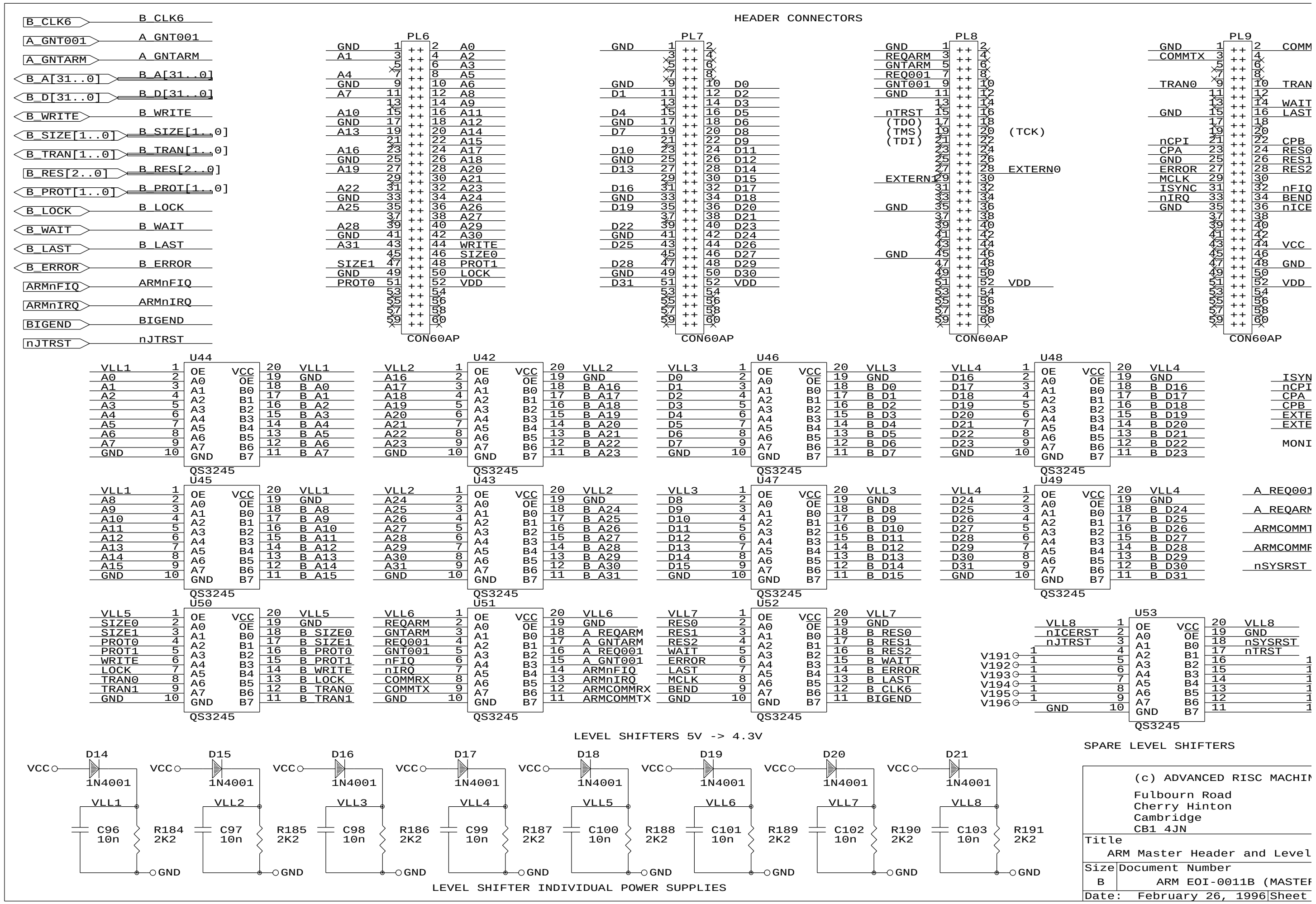
A.18 Memory Address and Data Buffers



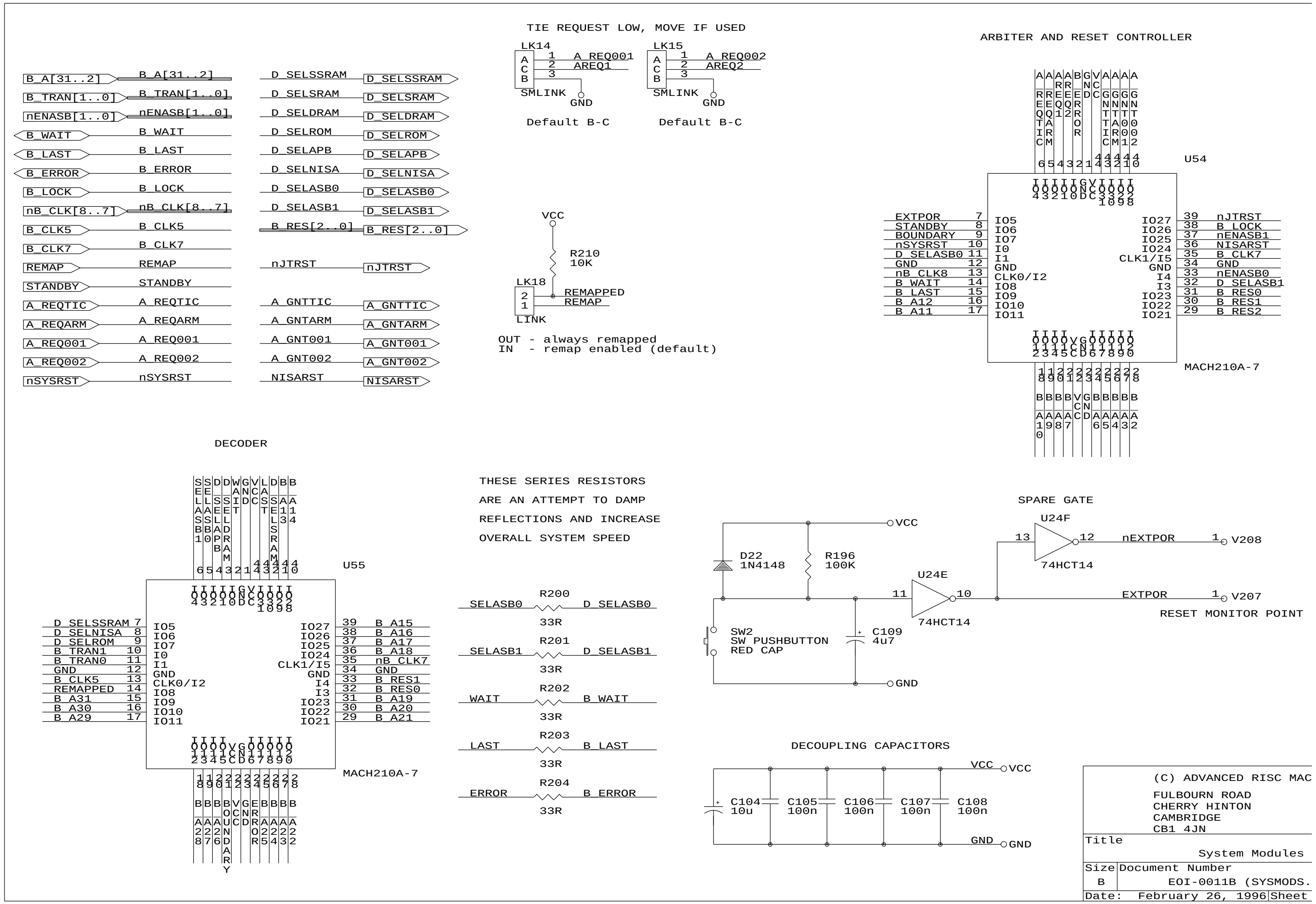
A.19 Test Interface Controller and Connecters



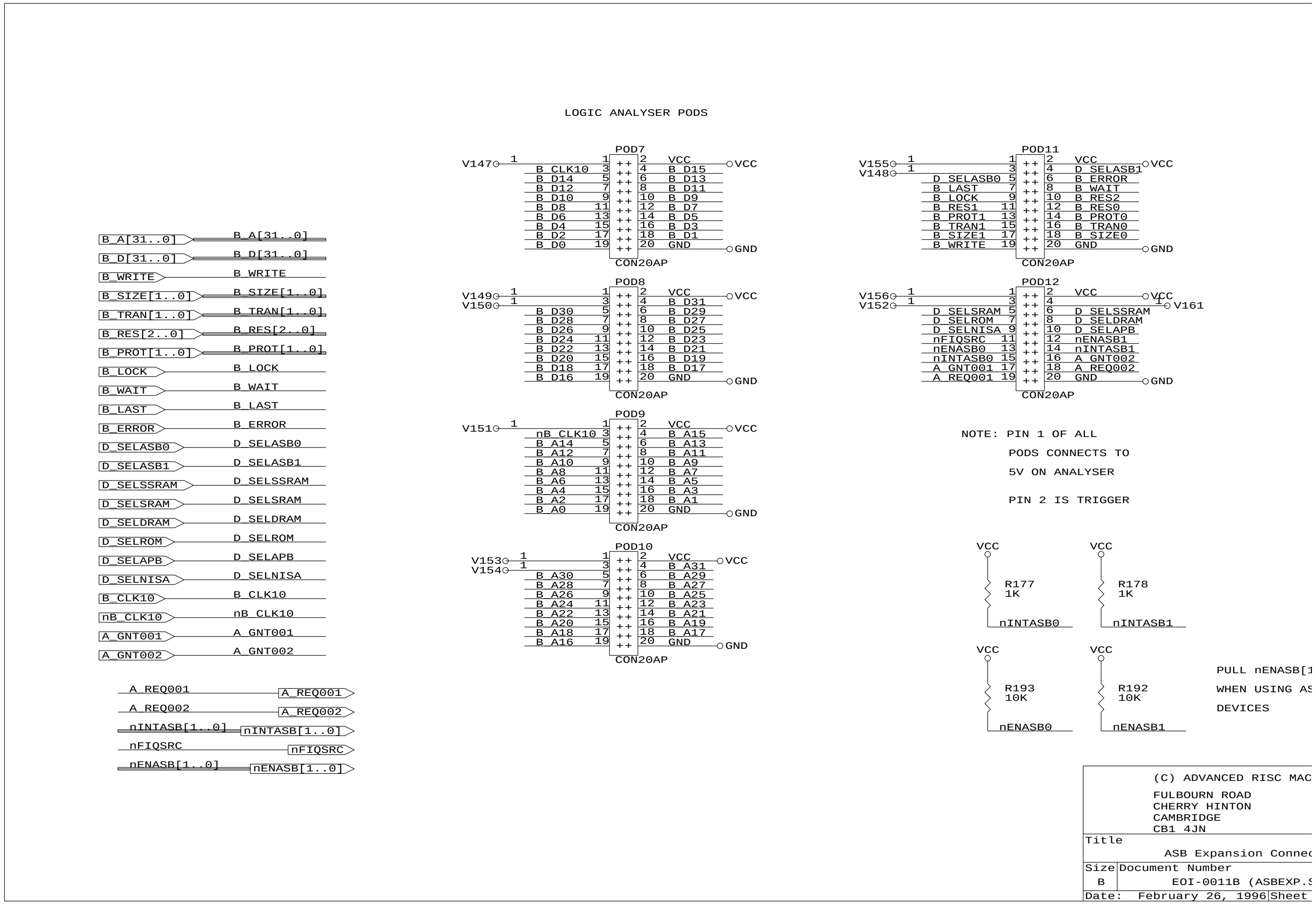
A.20 Master Header Connecters and Level Converters



A.21 System Modules (Arbiter and Decoder)



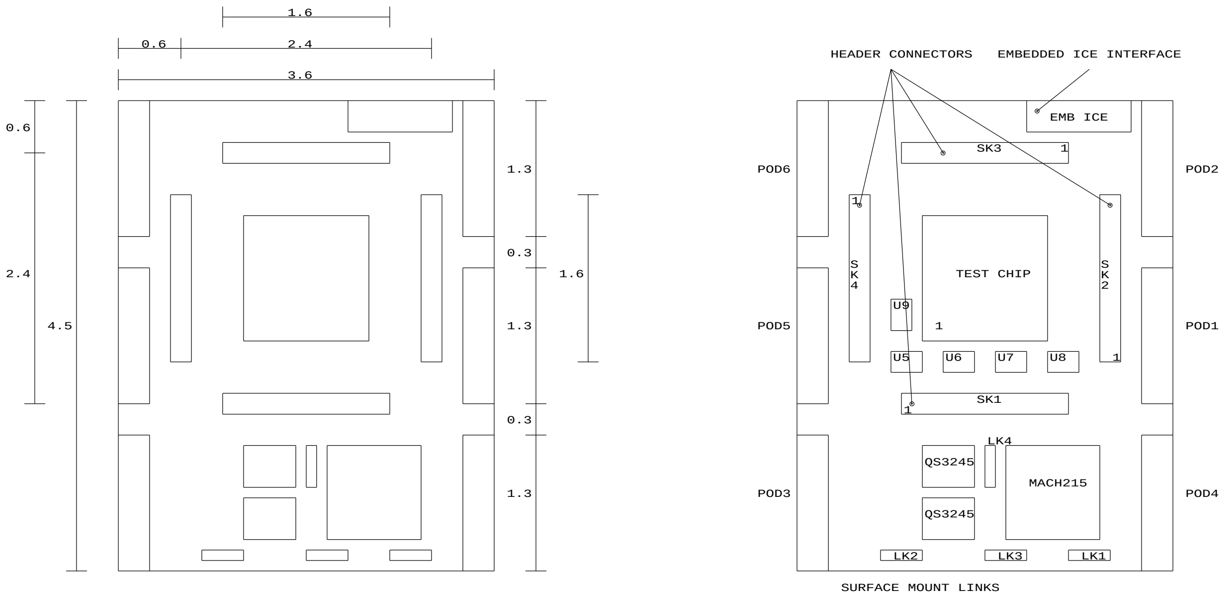
A.22 ASB Expansion Connecters



B Daughter Board Schematics

The daughter board design comprises the seven schematics as listed below. There are two versions of the processor schematic according to whether you have a QFP or PGA packaged part on the board.

B.1 Card Outline Drawing

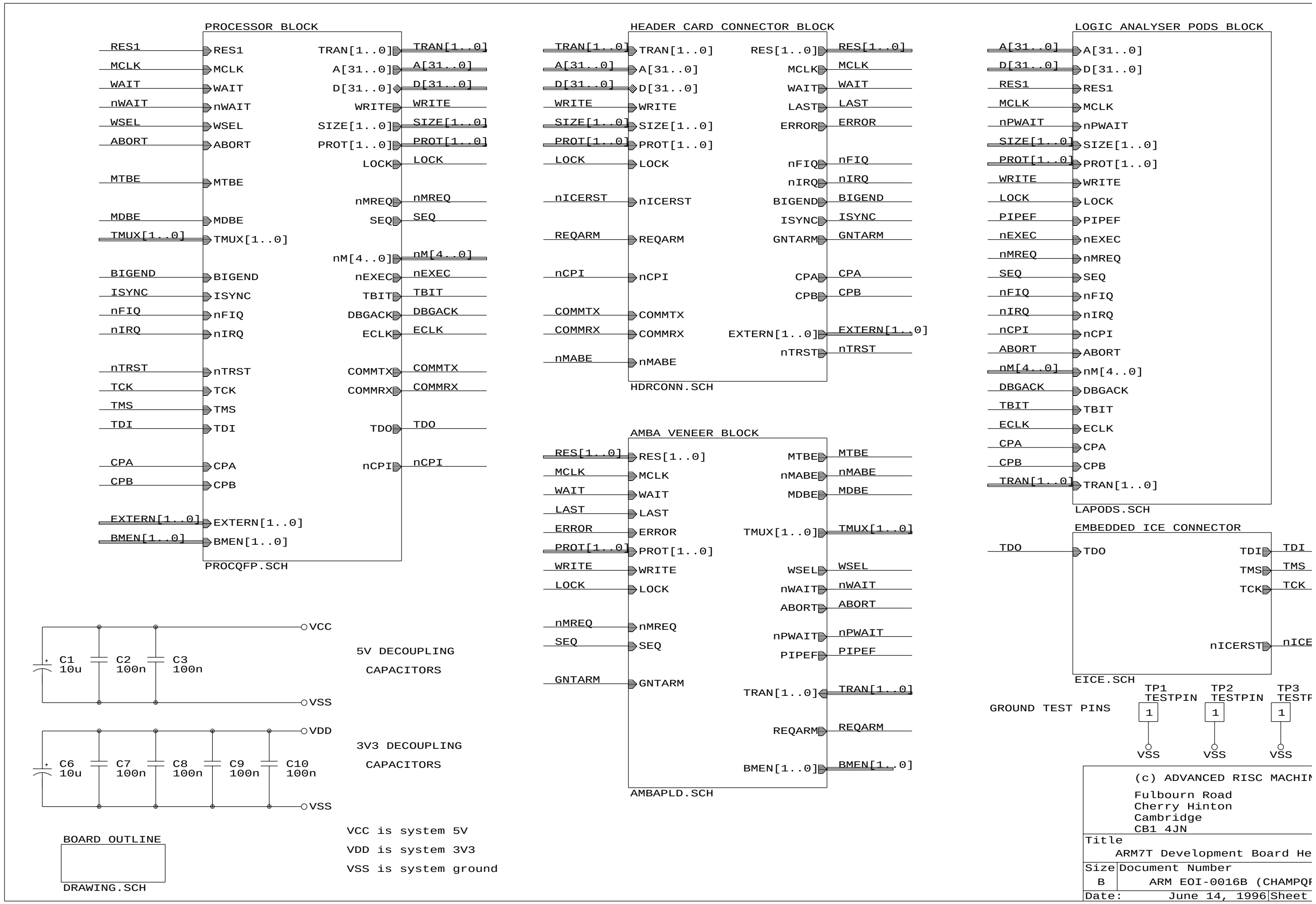


ALL DIMENSIONS IN INCHES

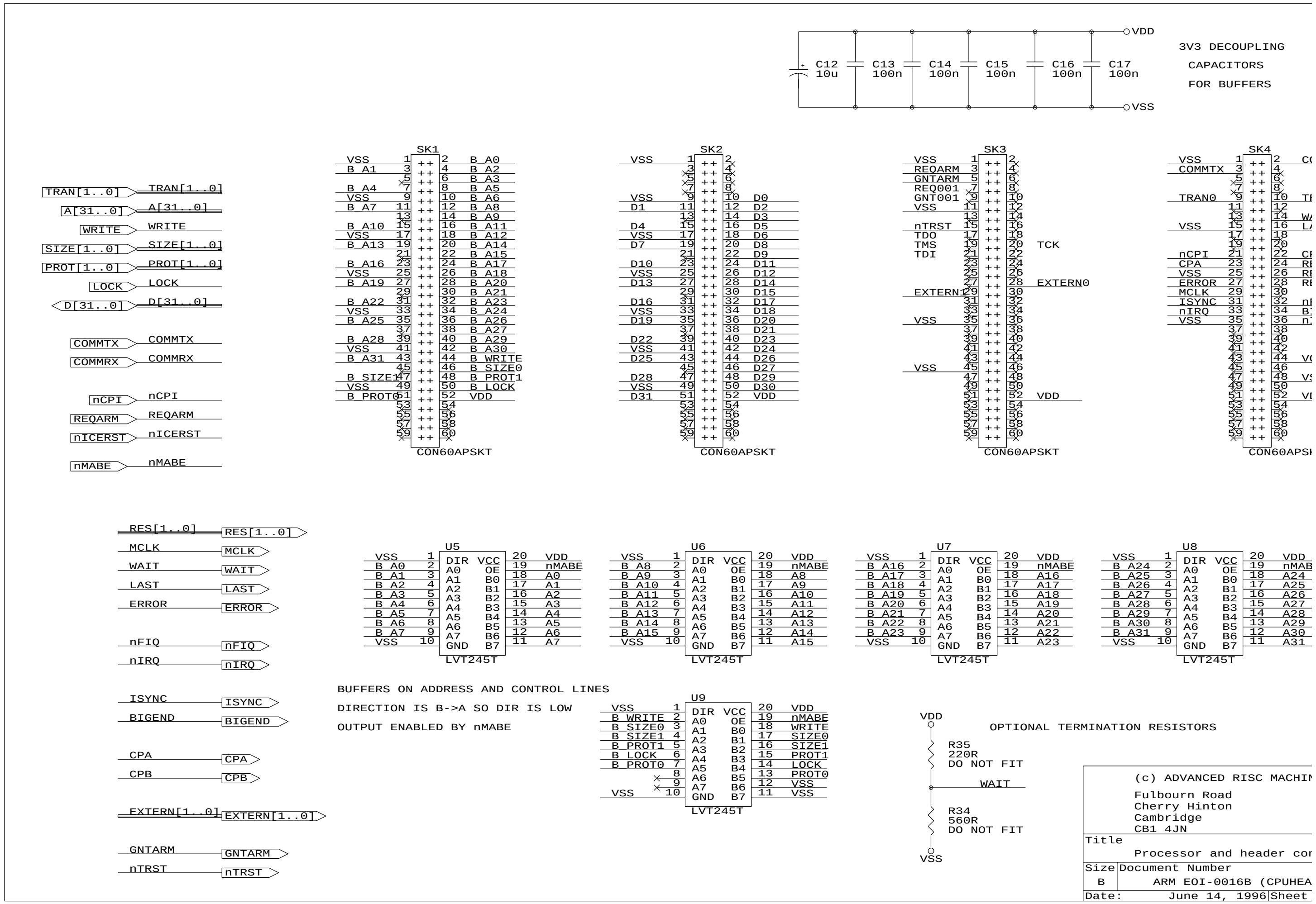
(C) ADVANCED RISC MAC
FULBOURN ROAD
CHERRY HINTON
CAMBRIDGE
CB1 4JN

Title	Header Card Outline Dr
Size	Document Number
B	EOI-0016B (DRAWING.
Date:	June 14, 1996 Sheet

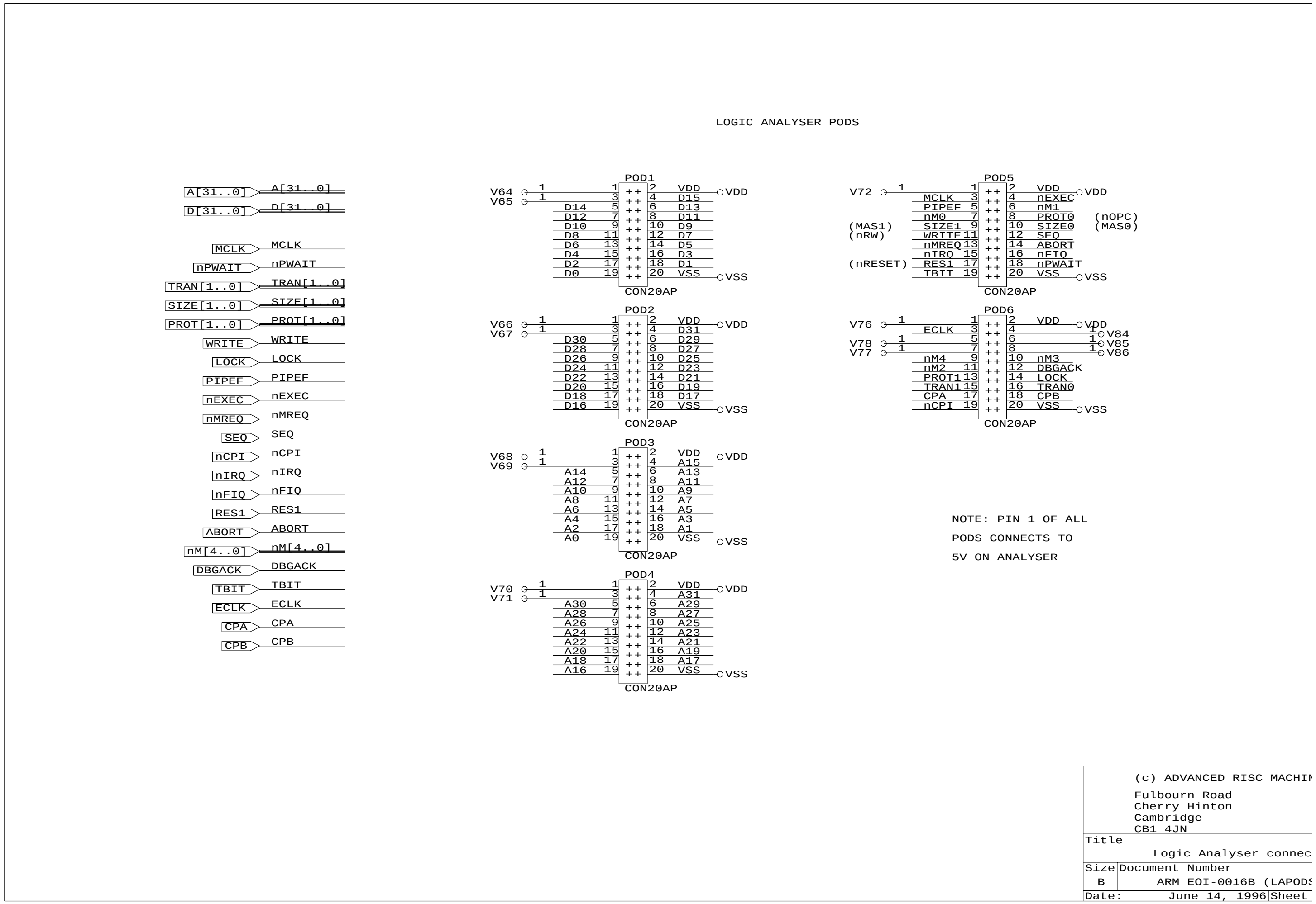
B.2 Top-level Diagram

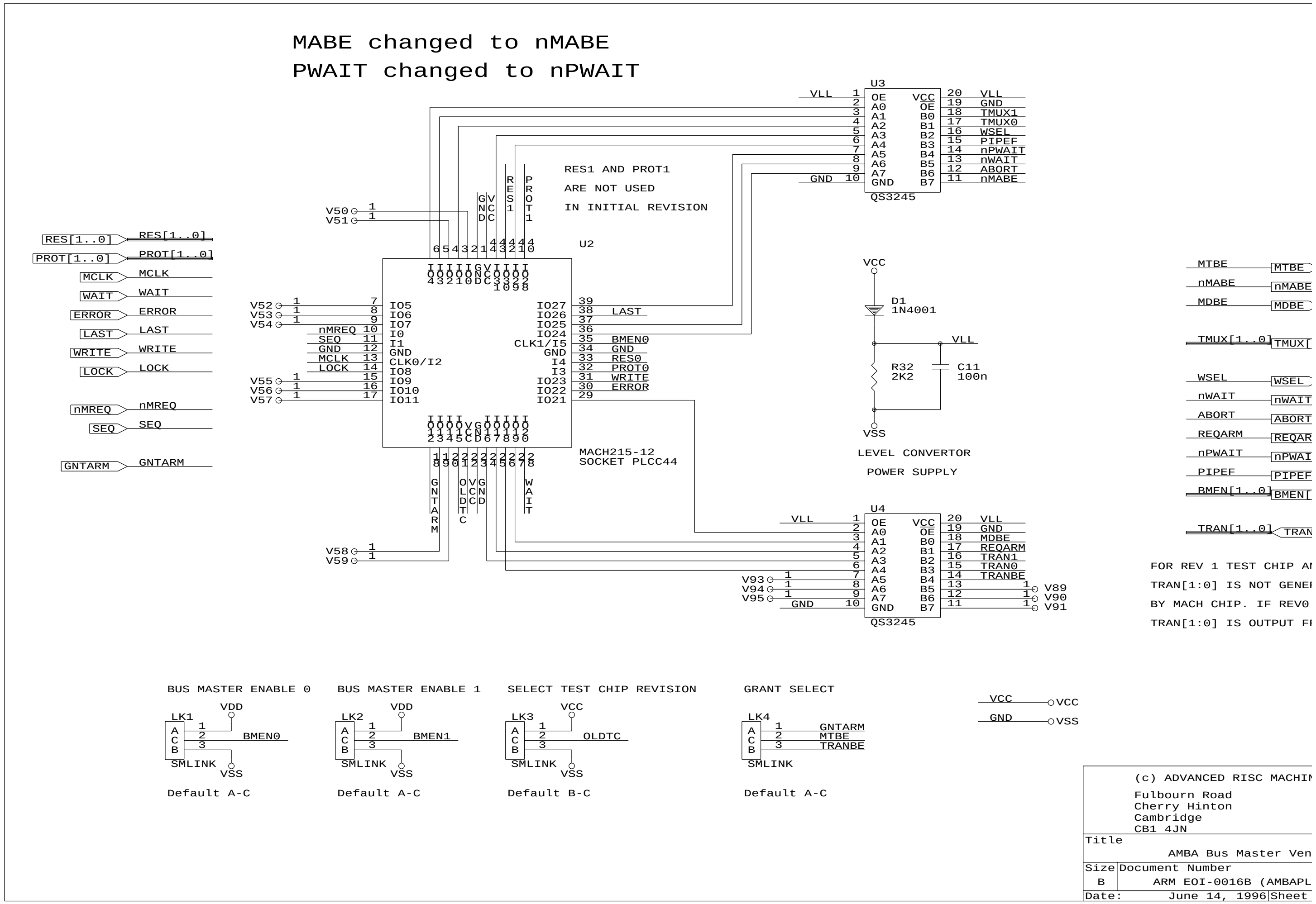


B.3 Header Connecters

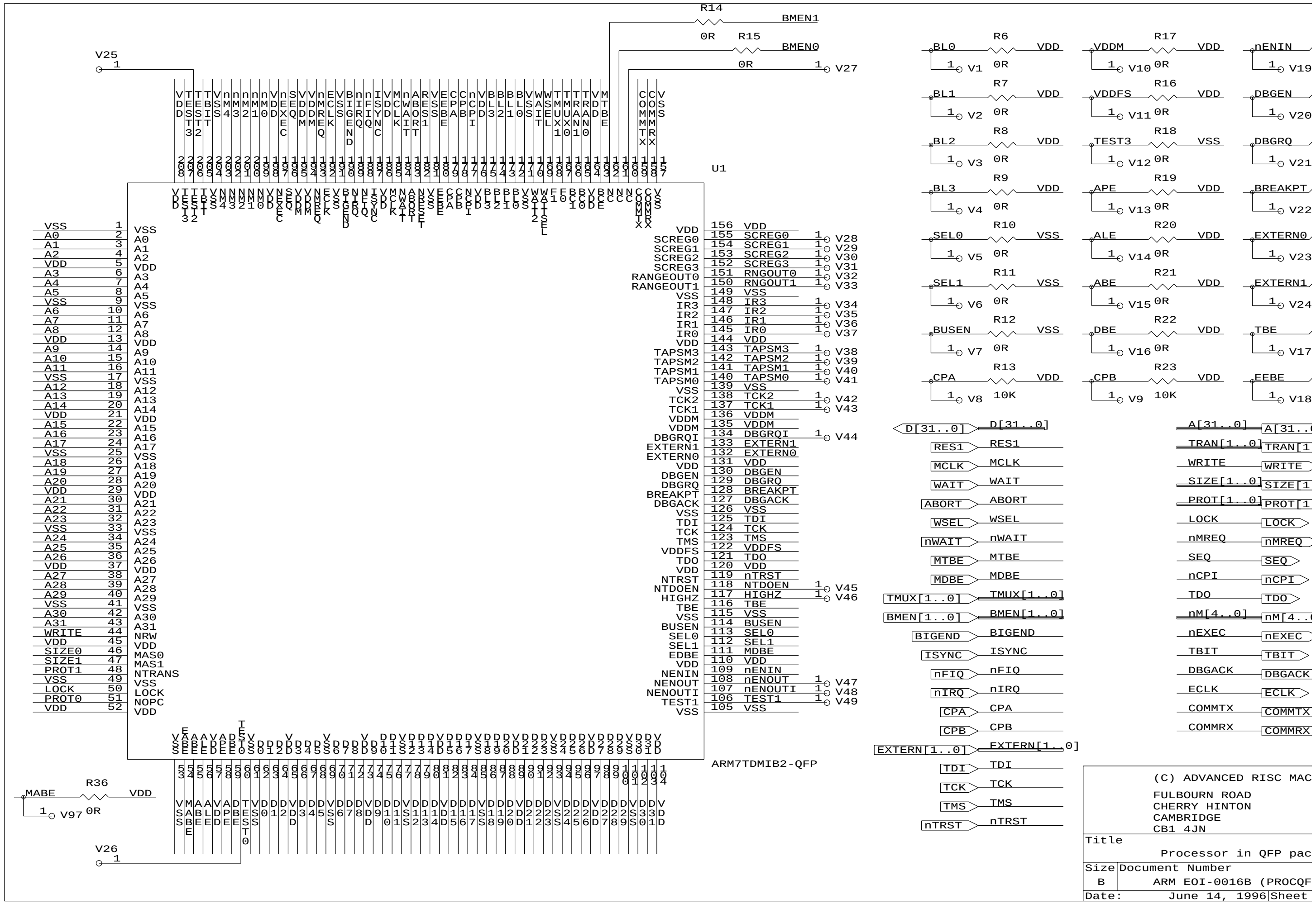


B.4 Logic Analyser Connecters

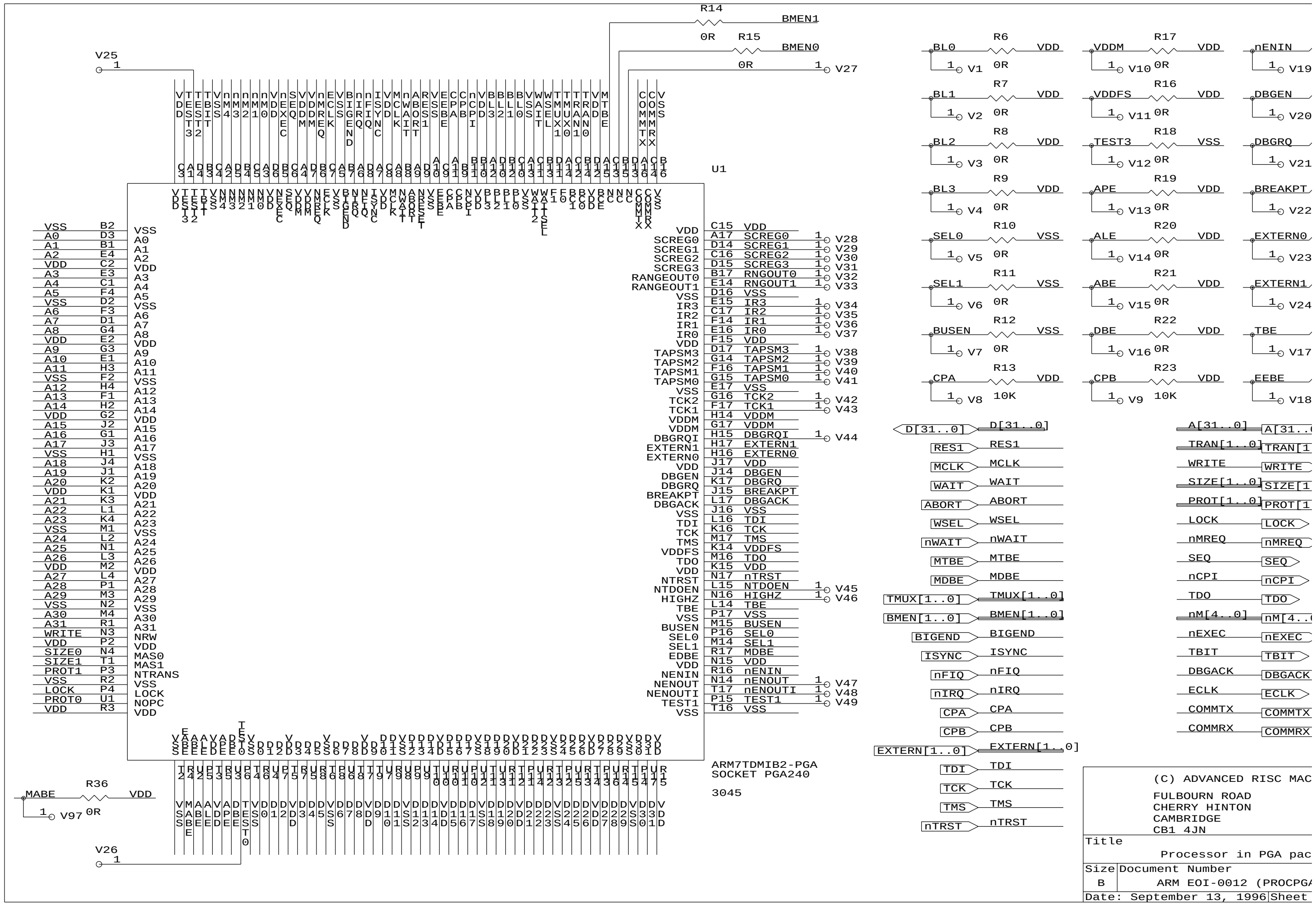



MABE changed to nMABE
PWAIT changed to nPWAIT
U3
VLL
OE A0 A1 A2 A3 A4 A5 A6 A7 GND
VCC OE B0 B1 B2 B3 B4 B5 B6 B7
VLL GND TMUX1 TMUX0 WSEL PIPEF nPWAIT nWAIT ABORT nMABE
QS3245
RES1 AND PROT1
ARE NOT USED
IN INITIAL REVISION
V50
V51
U2
RES[1..0]
PROT[1..0]
MCLK
WAIT
ERROR
LAST
WRITE
LOCK
nMREQ
SEQ
GNTARM
V52
V53
V54
V55
V56
V57
IO5 IO6 IO7 IO I1 GND CLK0/I2 IO8 IO9 IO10 IO11
IO27 IO26 IO25 IO24 CLK1/I5 GND I4 I3 IO23 IO22 IO21
LAST
BMEN0
GND
RES0
PROT0
WRITE
ERROR
MACH215-12
SOCKET PLCC44
GNTARM
OLDTC
VCC
GND
WAIT
V58
V59
VCC
D1
1N4001
VLL
R32
2K2
C11
100n
VSS
LEVEL CONVERTOR
POWER SUPPLY
U4
VLL GND MDBE REQARM TRAN1 TRAN0 TRANBE
V93
V94
V95
V89
V90
V91
QS3245
MTBE
nMABE
MDBE
TMUX[1..0]
WSEL
nWAIT
ABORT
REQARM
nPWAIT
PIPEF
BMEN[1..0]
TRAN[1..0]
FOR REV 1 TEST CHIP AN
TRAN[1:0] IS NOT GENEI
BY MACH CHIP. IF REV0
TRAN[1:0] IS OUTPUT FI
BUS MASTER ENABLE 0
LK1
VDD
BMEN0
SMLINK
VSS
Default A-C
BUS MASTER ENABLE 1
LK2
VDD
BMEN1
SMLINK
VSS
Default A-C
SELECT TEST CHIP REVISION
LK3
VCC
OLDTC
SMLINK
VSS
Default B-C
GRANT SELECT
LK4
GNTARM
MTBE
TRANBE
SMLINK
Default A-C
VCC
VCC
GND
VSS
(c) ADVANCED RISC MACHIN
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN
Title
AMBA Bus Master Ven
Size Document Number
B ARM EOI-0016B (AMBAPL
Date: June 14, 1996 Sheet

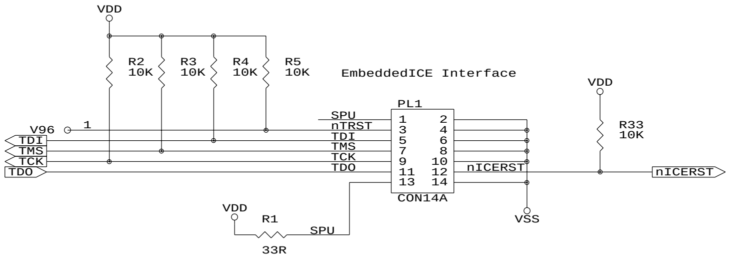
B.6 Processor in QFP Package



B.7 Processor in PGA Package



B.8 EmbeddedICE Interface



nTRST is driven by the system at power up only.
To reset the TAP controller hold TMS high for
5 TCK cycles. To cause a full system reset drive
nICERST low.

(c) ADVANCED RISC MACHIN
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

Title
EmbeddedICE Interfa

Size	Document Number
B	ARM EOI-0016B (EICE.

Date: June 14, 1996 Sheet

C Summary of Programmable Devices

This appendix summarizes the available programmable devices.

Summary of Programmable Devices

C.1 Programmable Devices

This appendix lists the available programmable devices. If you would like to use these designs, contact ARM for help.

C.1.1 Board devices

The board contains twelve programmable devices.

Ref	ARM Name	Ident.	Device
U2	CLKDIV	EFI-0017	PALCE22V10-7
U8	SSRAMC	EFI-0018	PALCE22V10-7
U10	EPROMC	EFI-0019	MACH210A-7
U11	EPROMDP	EFI-0020	MACH230-10
U12	ANGELDM	EFI-0021	8-bit EPROM/FLASH
U13	not fitted	EFI-0022	16-bit EPROM/FLASH
U14	DRAMC	EFI-0023	MACH231-7
U16	SRAMC	EFI-0024	MACH210A-7
U20	APBIF	EFI-0025	MACH231-7
U28	APBPER[1]	EFI-0026	XC17128D serial PROM
U37	TIC	EFI-0027	MACH210A-7
U54	ARBRES8	EFI-0028	MACH210A-7
U55	DECODER	EFI-0029	MACH210A-7

Table C-1: Programmable devices

Note [1] *U28 is not re-programmable.*

C.1.2 Daughter board device

The daughter board contains one programmable device.

Ref	ARM Name	Ident.	Device
U2	CPU4	EFI-0030	MACH215-12

Table C-2: The daughter board programmable device

C.1.3 MACH and PALCE

All MACH and PALCE devices can be reprogrammed. The functionality of these devices was designed using PALASM.

The PALASM source is available from ARM, as described in ***1.3 Useful Contacts*** on page 1-3.

C.1.4 FPGA

The FPGA is programmed by serial PROM (U28). To reprogram the device you need to replace this PROM with a new one.

The FPGA design was completed using VHDL and synthesised using the Compass synthesizer. The VHDL source is also available from ARM.

D

Summary of Jumpers and Links

This appendix summarises the jumpers and links.

Summary of Jumpers and Links

D.1 Overview

The ARM Development Card is configurable through the use of links, jumpers and switches. Each of these is described in detail in ***Chapter 3, Circuit Descriptions***. This section summarises that information.

Surface mount links are used where an option should only be changed for a special purpose. You may need to move these links if you are modifying the ARM Development Card to add additional hardware.

Standard 2-pin links are used for infrequently used options. You should only insert or remove the jumpers if you are sure of the effect.

DIP switches are used where the you may frequently want to make changes. You cannot stop the ARM Development Card functioning by altering the switch positions.

The default positions are denoted by a *.

D.2 Surface Mount Links

Ref	Name	Position	Description
LK1	NISA clock select	A * C	32MHz 24MHz
LK2	SYSCLK source	A * C	internal external
LK3	SYSCLK2X source	A * C	internal external
LK5	pipelined SSRAM	A * C	pipelined non-pipelined
LK12	PC card B VCC3EN	A * C	high low
LK13	PC card A VCC3EN	A * C	high low
LK14	AREQ1 select	A C *	**A_REQ001** low
LK15	AREQ2 select	A C *	**A_REQ002** low

Table D-1: Surface mount links

D.3 Standard 2-pin Links

Ref	Name	Position	Description
LK4	BIGEND	out * in	little-endian big-endian
LK7	P_STB WIDTH	out * in	2-cycle **P_STB** 1-cycle **P_STB**
LK8	INT TYPE	out * in	latched mode ACK mode

LK9	DIRN	out *
		in
			BIDEN=1
			BIDEN=0

LK10	ENABLE INT	out * in	disable switch interrupt enable switch interrupt
LK17	USETIC	out * in	disable test interface enable test interface
LK18	REMAP	out in *	**REMAP** high **REMAP** driven

Table D-2: Standard 2-pin links

Summary of Jumpers and Links

D.4 Link Fields

CYC1	CYC0	Cycles
in	in	2
in	out	3
out	in	4
out	out	5

Table D-3: Cycle selection

Ref	Position	Name	Option	Description
LK6	1	CYC1	out * in *	see Table D-3 see Table D-3
	2	CYC0	out * in	see Table D-3 see Table D-3
	3	EPROM	out in *	select EPROM select FLASH
	4	SEL8BIT	out * in	8-bit device 16-bit device
LK11	1		out * in	use parallel port PP bit0 to S3-1
	2		out * in	use parallel port PP bit1 to S3-2
	3		out * in	use parallel port PP bit2 to S3-3
	4		out * in	use parallel port PP bit3 to S3-4
	5		out * in	use parallel port PP bit4 to LED PP0
	6		out * in	use parallel port PP bit5 to LED PP1
	7		out * in	use parallel port PP bit6 to LED PP2
	8		out * in	use parallel port PP bit7 to LED PP3
LK16	1	INIT	out * in	no function do not connect
	2	MODE0	out in *	use download cable use serial PROM
	3	MODE1	out in *	use download cable use serial PROM
	4	MODE2	out in *	use download cable use serial PROM

Table D-4: LK6, LK11, and LK16

D.5 DIP Switches

B#CYC1 [1]	B#CYC0 [1]	Cycles	
on	on	2	*
on	off	3	
off	on	4	
off	off	5	

Table D-5: S2 Switch positions

B#SIZ1[1]	B#SIZ0[1]	Size	
on	on	8-bit	
on	off	16-bit	
off	on	32-bit	*
off	off	32-bit	

Table D-6: S2 Switch positions

[1] # is the bank number: 0 or 1

Ref	Position	Name	Option	Description
S1	1	SEL0	on/off	see Table D-8
	2	SEL1	on/off	see Table D-8
	3	SEL2	on/off	see Table D-8
	4	SEL3	on/off	see Table D-8
S2	1	B0CYC0	on/off	see Table D-5
	2	B0CYC1	on/off	see Table D-5
	3	B0SIZ0	on/off	see Table D-6
	4	B0SIZ1	on/off	see Table D-6
	5	B1CYC0	on/off	see Table D-5
	6	B1CYC1	on/off	see Table D-5
	7	B1SIZ0	on/off	see Table D-6
	8	B1SIZ1	on/off	see Table D-6
S3	1	S3-1	on/off	toggle parallel port bit 0
	2	S3-2	on/off	toggle parallel port bit 1
	3	S3-3	on/off	toggle parallel port bit 2
	4	S3-4	on/off	toggle parallel port bit 3

Table D-7: S!, S2, and S3

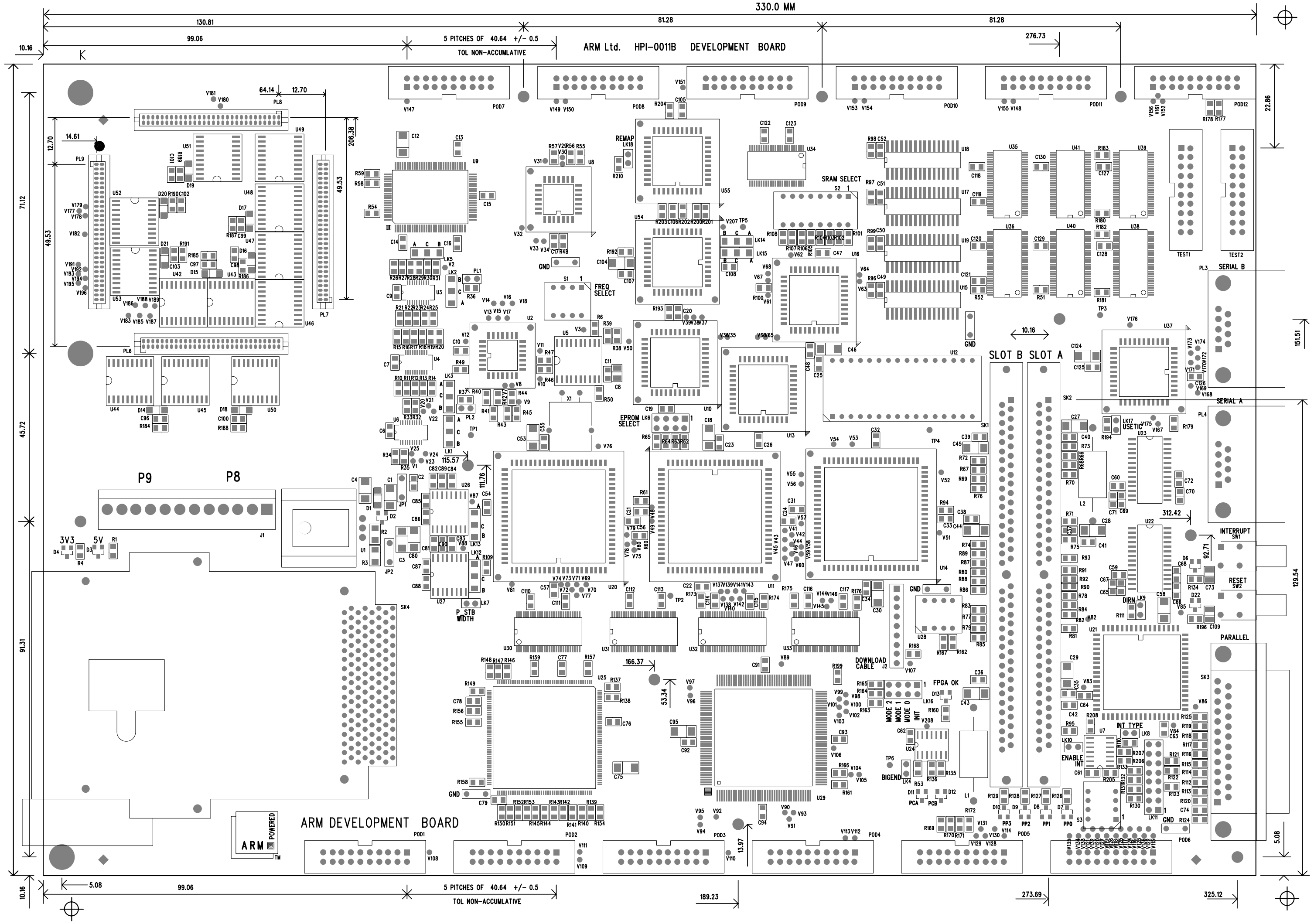
	Switch position			Frequency (MHz)	
SEL3	SEL2	SEL1	SEL0	SYSCLK	SYSCLK2X
on	on	on	on	4	8
on	on	on	off	8	16
on	on	off	on	16	32
on	on	off	off	20	40

Table D-8: S1 switch positions

E Mechanical Information

This appendix shows a mechanical drawing of the ARM Development Card with dimensions to help you to build add-on hardware.

Mechanical Information


330.0 MM
130.81
81.28
81.28
99.06
5 PITCHES OF 40.64 +/- 0.5
TOL NON-ACCUMLATIVE
276.73
10.16
ARM Ltd. HPI-0011B DEVELOPMENT BOARD
POD7
POD8
POD9
POD10
POD11
POD12
64.14
12.70
206.38
14.61
12.70
71.12
49.53
49.53
22.86
REMAP
SRAM SELECT
FREQ SELECT
EPROM SELECT
TEST1
TEST2
SERIAL B
SERIAL A
SLOT B
SLOT A
10.16
151.51
GND
P9
P8
3V3
5V
115.57
111.76
45.72
91.31
USETIC
312.42
INTERRUPT
RESET
92.71
129.54
PARALLEL
P_STB WIDTH
166.37
53.34
DOWNLOAD CABLE
FPGA OK
MODE 2
MODE 1
MODE 0
INIT
BIGEND
INT TYPE
ENABLE INT
DIRN
PCA
PCB
ARM DEVELOPMENT BOARD
ARM POWERED
POD1
POD2
POD3
POD4
POD5
POD6
13.97
5.08
5.08
99.06
5 PITCHES OF 40.64 +/- 0.5
TOL NON-ACCUMLATIVE
189.23
273.69
325.12
10.16