Section 1

Overview

1.1 Scope

The AT91EB01 Evaluation Board enables real-time code development and evaluation. It supports the AT91M40400 family with its various memory options.

This guide focuses on the AT91 Evaluation Board as an evaluation and demonstration platform.

1. Section 1 provides an overview.
2. Section 2 describes the selection of the debugging system.
3. Section 3 details the set-up of the development board.
4. Section 4 contains the memory maps.
5. Section 5 gives the circuit description.

Following are 2 appendices covering configuration links and memory data, and schematics including pin connectors.

1.2 Deliverables

The evaluation board is supplied with a DB9 plug to DB9 socket straight through serial cable to connect the target evaluation board to a PC. There is also a bare power lead with a 2.1 mm jack on one end for connection to a bench power supply.

To use the evaluation board the user needs to provide a host computer and a debugging system, both of which are described below.

1.3 System Requirements

The host computer must be running the debugger in the ARM® Software Development Toolkit, version 2.1x.

If an earlier version of the Software Development Toolkit is being used, it is strongly recommended to upgrade to the most recent version. Other software toolkits for the ARM, available from third parties, are not discussed in this guide. If a third-party toolkit is being used, use this guide alongside the documentation supplied with the toolkit.

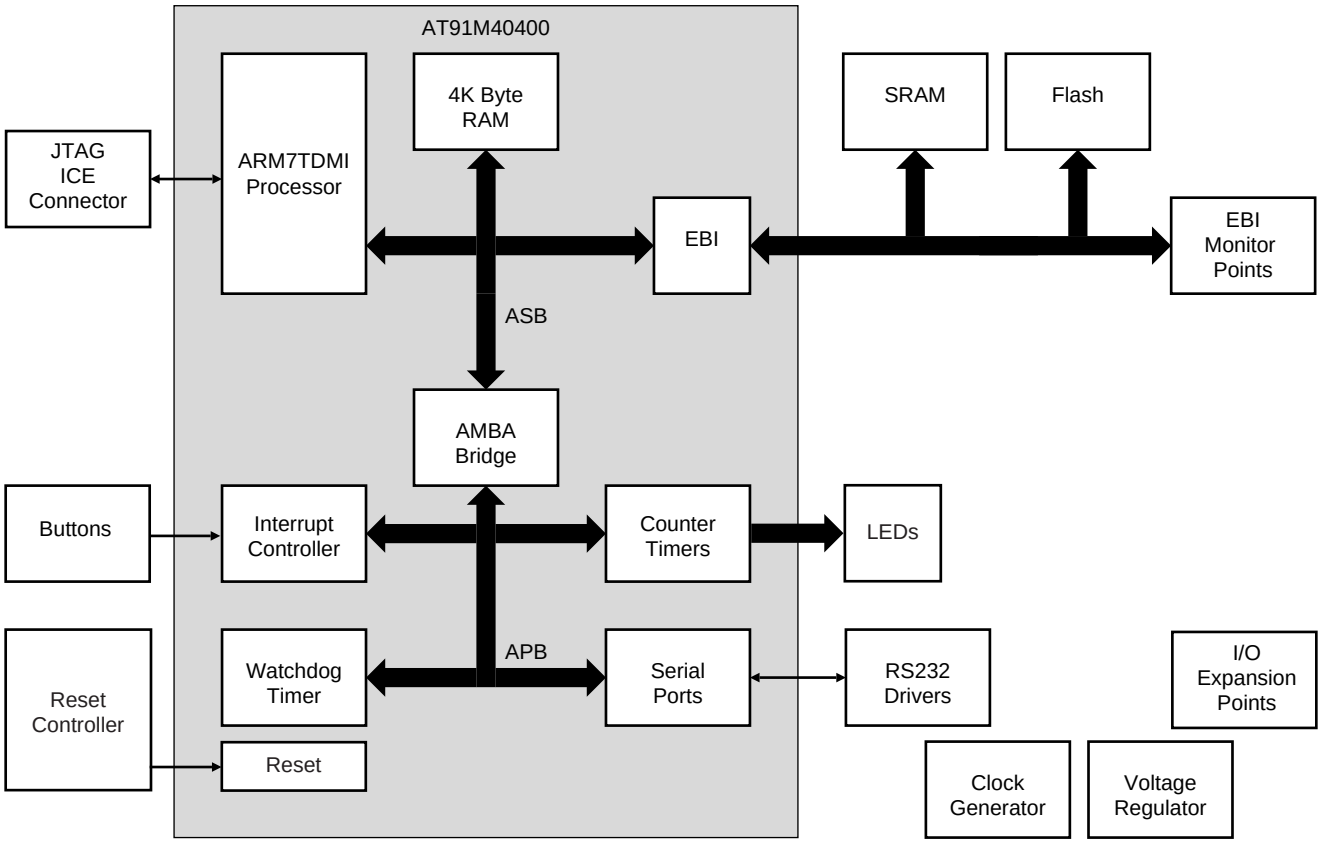
1.4 The AT91EB01 Evaluation Board

The board consists of an AT91M40400 together with several peripherals:

- Two serial ports
- Reset button
- Three Applicative Buttons (FIQ, TIOB0, IRQ0)
- Three LEDs (TIOA0, TIOA1, TIOB0)
- 512K bytes 16-bit SRAM (upgradable to 2048K bytes)
- 128K bytes 16-bit Flash (of which 64K bytes is available for user software)
- 20-pin JTAG interface connector

If required, user defined peripherals can also be added to the board. See “Appendix A - Configuration Links and SRAM Bank 1” on page 27 for details.

Figure 1. AT91EB01 Block Diagram



Section 2

The Debugging System

The number of debugging systems available for the AT91EB01 is rapidly increasing. This section presents the characteristics and advantages of several of these debugging systems. If a debugging system has already been chosen, proceed to “Setting up the AT91EB01 Evaluation Board” on page 7.

2.1 Choosing a Debugging System

The four debugging systems available for software applications developed to run on an AT91 microcontroller described in this document are:

- The ARMulator
- EmbeddedICE
- Multi-ICE
- Angel Debug Monitor (with or without EmbeddedICE or Multi-ICE)

2.1.1 The ARMulator

The ARMulator is a software emulator of the ARM7TDMI processor. It supports both the ARM and Thumb® instruction sets. The ARMulator enables the development of software before any hardware is available. It does this by providing an environment for the development of AT91-targeted software on non-ARM-based host systems. A target system is not necessary to use the ARMulator. The ARMulator is included in the ARM Software Development Toolkit.

2.1.2 EmbeddedICE

EmbeddedICE is a JTAG-based, non-intrusive, debugging system for ARM7TDMI processors. EmbeddedICE provides the interface between a debugger and the ARM7TDMI core of the AT91M40400 on the evaluation board, using the JTAG port for communication.

EmbeddedICE provides the user with:

- Real-time address and data-dependent breakpoints (including ROM)
- Single stepping
- Full access and control of the ARM core
- Full memory access (read and write)
- Full I/O system access (read and write)

EmbeddedICE also enables the embedded microprocessor to access host system peripherals, such as screen display, keyboard input, and disk drive storage.

EmbeddedICE is an extension to the architecture of the ARM7TDMI RISC processor, and provides the ability to debug cores that have been deeply embedded into systems. It consists of:

- A set of debug extensions to the ARM7TDMI processor
- The EmbeddedICE macrocell to provide access to the extensions from the outside world
- The ARM EmbeddedICE interface, to provide communication between the EmbeddedICE macrocell and the debugger resident on the host computer

The EmbeddedICE macrocell is the integrated on-chip logic that provides debug support for the ARM7TDMI processor. The EmbeddedICE macrocell is programmed in serial through the Test Access Port (TAP) controller on the ARM7TDMI, via the JTAG interface. This is usually achieved via the EmbeddedICE interface. The EmbeddedICE macrocell consists of two real-time watchpoint units, together with a control and status register. One or both watchpoint units can be programmed to halt the execution of instructions by the ARM7TDMI.

The ARM EmbeddedICE interface is a JTAG protocol conversion unit. This translates the debug protocol messages generated by the debugger into JTAG signals that can be sent to the EmbeddedICE macrocell, and vice versa. The interface can be connected to the host computer using either the built-in serial port, or the built-in serial and parallel ports. The serial port is used for bidirectional transfers, but the parallel port is only used to download code. Using the parallel port increases the maximum download speed with EmbeddedICE to approximately 20K bytes/s. This is considerably greater than using the serial ports alone (even at 38400 baud), and so is the recommended method of using EmbeddedICE (host permitting).

Although EmbeddedICE is generally non-intrusive, there are two exceptions:

- When an ARM debugger is started, it attempts to find out the state of the target microcontroller. To do this, it halts the target and inspects the state of the ARM registers. This, however, can be considered non-intrusive if the debugging session begins after the debugger has been started.
- Watchpoints on structures or arrays that are larger than one word may cause the target microcontroller to halt execution when writes occur close to the watchpointed area. EmbeddedICE will restart execution transparently to the user, but this may still cause problems if the application is real-time.

Note: The Debug Comms Channel is currently not supported on the AT91EB01 board.

2.1.3 Multi-ICE

Multi-ICE is the latest ICE-compatible debug solution from Advanced RISC Machines. It comprises an interface unit which connects between the parallel port of a PC and the JTAG interface and software to allow an ARM debugger to communicate with the interface unit.

The Multi-ICE system is very similar to EmbeddedICE except that the Embedded ICE interface unit contains a processor and software, whereas the Multi-ICE interface unit contains no processor and the interface software runs on the host PC. There are a number of advantages to this approach - a smaller, lighter interface unit with a lower power consumption (it can be target powered), easier updating of software, more flexible interfaces.

Features of Multi-ICE:

- Stored/Automatic configuration

- Networked connections
- High-performance code download
- Low-voltage target operation
- Small, lightweight unit improves ease of use
- Additional power supply not normally required (powered from target)
- Easy-to-use parallel port connection
- Software configurable JTAG rate
- Easy software upgrades - 100% host-based software
- Extra user input/output bits for user-supplied drivers

2.1.4 The Angel Debug Monitor

Note: To aid readability, the term “Angel” is used to mean the Angel Debug Monitor on target throughout this user guide.

Angel is a program that enables development and debugging of applications running on the AT91 microcontrollers. Angel communicates with a debugger that can handle the Angel communications protocol, such as the ARMSD, the ARM Debugger for Windows.

Angel can debug applications running in both ARM and Thumb state on the AT91 Evaluation Board. A typical Angel system has two main components that communicate via a physical link, such as a serial cable:

Debugger: This runs on the host computer. It gives instructions to Angel and displays the results obtained from it. The debugger could be ARMSD, the ARM Debugger for Windows.

Angel: This runs alongside the application being debugged on the target platform.

The ARM Software Development Toolkit features two versions of Angel:

- a full version for use on development hardware
- a minimal version for use on production hardware.

Only the full version is provided with the AT91EB01.

Angel uses a debugging protocol called the Angel Debug Protocol (ADP), which supports multiple channels. It requires control over the ARM’s exception vectors, but can share these with your applications.

Angel can be used to:

- evaluate existing application software on real hardware (as opposed to emulation of the hardware)
- develop software applications on the AT91 Evaluation Board
- bring into operation new hardware that includes an ARM processor
- port operating systems to the AT91 microcontroller

These activities require some understanding of how Angel’s components work together. Some involve modification of Angel itself, such as porting operating systems. For more detailed information, see the ARM Software Development Toolkit User Guide (ARM DUI 0040) and the ARM Software Development Toolkit Reference Guide (ARM DUI 0041).

The section “Using Angel” on page 19 gives details on the use of Angel on the AT91EB0x.

2.1.5 How EmbeddedICE and Multi-ICE differs from a Debug Monitor

A debug monitor, such as Angel, is an application that runs on the board in conjunction with the user application, and requires some resource to be available for it on the board. When the board powers up, Angel installs itself by initializing the vector table so that it takes control of the board when an exception occurs. Communication coming in from the host causes an interrupt, halting the user application and calling the appropriate code within Angel. Angel then returns to the user application. This can complicate matters if the application also requires access to interrupts. Similarly, if the application requests some form of I/O to the host, this is implemented within the application using a software interrupt (SWI) instruction that is dealt with by Angel's SWI handler. This means that Angel requires non-volatile memory to store the debug monitor code, RAM to store its data, and control over the exception vectors to enable it to gain control of the ARM7TDMI while the user's application is running.

EmbeddedICE and Multi-ICE, on the other hand, require no such resource. Rather than existing as an application on the board, they work by using a combination of the debug hardware on the core and the interface that handles communication between the core's debug hardware and the host. EmbeddedICE and Multi-ICE have been designed so that debugging via the JTAG port is as non-intrusive as possible:

- The target microcontroller does not require special hardware to support debugging (the debug extensions and the EmbeddedICE macrocell are all that are necessary).
- The target microcontroller system does not require memory to be set aside for debugging nor special software to be incorporated to allow debugging.

Note: Although EmbeddedICE and Multi-ICE require no memory on the target to operate, the target still requires some memory to execute the application code: An ICE does not provide a memory emulation.

EmbeddedICE and Multi-ICE may pollute the exception vectors if semihosting and/or vector catch are enabled.

EmbeddedICE and Multi-ICE interact with the target processor by halting execution and then forcing the processor into debug state to access data required by the debugger via the JTAG interface. This results in the processor being halted for significant periods of time, which may cause problems when you are debugging real-time systems.

Angel is invoked by an SWI instruction and does not halt the processor. Instead, control is passed from the application code to the debug monitor. This may reduce the amount of dead time when data is accessed on behalf of the debugger. Furthermore, interrupts can be disabled for the minimum time needed by the debugger, allowing real-time application interrupts to be serviced.

Section 3

Setting up the AT91EB01 Evaluation Board

3.1 Electrostatic Warning

The AT91EB01 Evaluation Board is shipped in protective packaging. The board must not be subjected to high electrostatic potentials. A grounding strap or similar protective device should be worn when handling the board. Avoid touching the component pins or any other metallic element.

3.2 Requirements

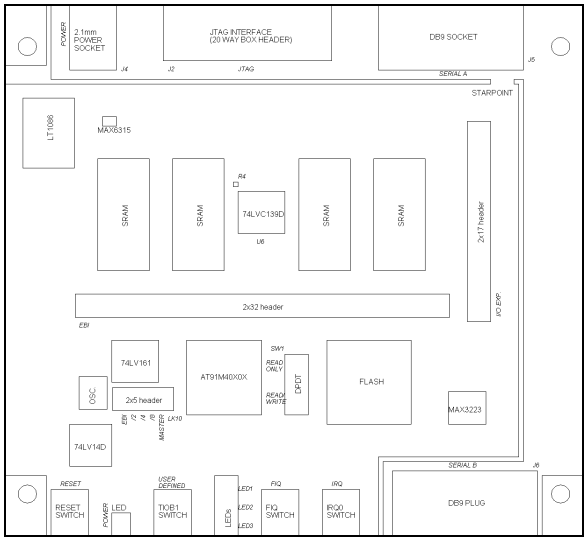
Requirements in order to set up the AT91EB01 Evaluation Board are:

- the host computer running the ARM Software Development Toolkit (a time-limited evaluation copy is provided with the Kit version of the AT91EB01)
- the AT91EB01 Evaluation Board itself
- suitable connection between the host and the development board (see the following sections)
- DC power supply capable of supplying 7.5V to 9V @ 500 mA (not supplied)

3.3 Layout

Following is the layout of the AT91EB01 evaluation board.

Figure 2. Layout of the AT91EB01 Evaluation Board

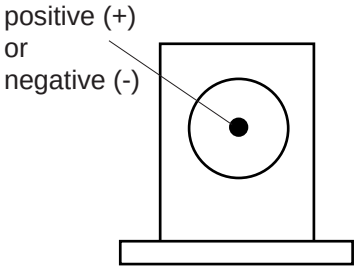


3.4 Jumper Settings

LK2 is used to select the clock frequency and by default is set to 16,384 MHz. For more information about this jumper and also the various surface mount links see “Appendix A - Configuration Links and SRAM Bank 1” on page 27.

3.5 Powering Up the Board

DC power is supplied to the board via the 2.1mm socket (J4) shown below in Figure 1. The polarity of the power supply is not critical. The minimum voltage required is 7V.



2.1 mm connector

The board has a voltage regulator providing +3.3V. The regulator allows the input voltage to be from 7V to 9V. When you switch the power on, the red LED marked "POWER" will light up. If it does not, switch off and check the power supply connections.

3.5.1 Measuring Current Consumption on the AT91M40400

The board is designed so as to generate the whole power supply of the AT91 device, and only the AT91 device through the wirelink WL2. This feature enables measurements to be made of the current consumption of the AT91 device.

3.6 Getting Started with the AT91EB01

In the following, it is assumed that:

- the ARM Software Development Toolkit has been installed,
- there are no default options for the APM tools, and
- the directory “Bin\Config” of the ARM setup is empty (except “readme.txt”).

3.6.1 Testing the Board

Connect the straight cable between Serial A and Serial B. With the FIQ button pressed down, power up the board, or generate a reset. The 3 LEDs (green, amber, red) light up. Release the FIQ. The 3 LEDs go off and then blink once.

Press the TIOB1 button. The red LED lights up. Release the TIOB1. The red LED goes off. If the board has 512 Kbytes SRAM, the green LED blinks twice. If the board has 2048 Kbytes SRAM, the green LED blinks four times.

Press the FIQ button. The amber LED lights up. Release the FIQ. The amber LED goes off and the green LED blinks four times. If the green LED blinks twice and then the red LED blinks twice, the straight cable is not connected.

Press the IRQ0 button. The amber and red LEDs light up. Release the IRQ0. The LEDs go off and the green LED blinks twice.

3.6.2 Building the Software Library

Copy the AT91 library (complete folder and sub-folder) onto your hard disk in a directory referred to in the following description as <MyFolderAT91>.

Copy/move the files from the directory “<MyFolderAT91>\Template” in the directory “Template” of the ARM setup.

Start the ARM Project Manager (APM).

Open the project “at91_l16” in the directory “<MyFolderAT91>\Library” and build the 16-bit THUMB library.

Open the project “at91_l32” in the directory “<MyFolderAT91>\Library” and build the 32-bit ARM library.

3.6.3 Use with the Angel Debugger

Build the project for the Angel Debug System variant.

- Open the project “led_blink.apj” in the directory “<MyFolderAT91>\Examples\led_blink”.
- Select the variant “EB01SramAngel”, and build the application.
- Connect the AT91EB01 on your host computer.
- Connect the COM port of the PC with the SERIAL A port of the AT91EB01 using the straight cable.
- Check the SW1 switch is on the “LOWER_MEM” position.
- Supply power to the AT91EB01: the amber LED lights up.
- Configure the debugger to work with Angel:
- Start the ARM Debugger for Windows (ADW) in Armulate Mode.
- Activate the menu “Options – Configure Debugger”.
- If not already done, add the “remote_a.dll” target environment to the debugger.
- Select the target environment “remote_a.dll” and configure it:
  - “Serial” Remote Connection
  - Heartbeat enabled
  - COMx port at 38400 baud
- Select OK: the debugger starts communicating with Angel, displaying a small window with transmit and receive counters.
- The connection with Angel is established. In the console window is displayed:

```
  Angel Debug Monitor (serial) 1.04 (Advanced RISC Machines SDT 2.11a) for
AT91EB0x(1.01)
  Angel Debug Monitor rebuilt on Sep. 14 1998 at 14:52:22
  Serial Rate: 38400
```

Load the Image and start the application:

- Activate the menu “File – Load Image…“.
- Select the file “led_blink.axf“ in the directory ”<MyFolderAT91>\Examples\led_blink\EB01SramAngel”: the debugger loads the image, displaying a window with transmits and receives counter.
- Select “Execute – Go” or press F5: the debugger stops on the main function, displaying the C source file.
- Select “Execute – Go” or press F5 again: the console window continuously displays “Loop 0, Loop 1, …” and the amber and red LEDs are blinking.

3.6.4 Use with Embedded ICE

Build the project for the Embedded ICE Debug System variant.

- Open the project “led_blink.apj” in the directory “<MyFolderAT91>\Examples\led_blink”.
- Select the variant “EB01SramICE”, and build the application.
- Connect the AT91EB01 on your host computer.

- Connect the COM port of the PC to the DB09 connector of the Embedded ICE Box with the crossed cable (not delivered with the AT91EB01).
- Connect the parallel port of the PC to the DB25 connector of the Embedded ICE Box with the parallel cable (not delivered with the AT91EB01).
- Connect the 14-pin connector of the Embedded ICE Box to the 20-pin JTAG connector of the AT91EB01 through the Embedded ICE to the JTAG target connector.
- Check the SW1 switch is on the “LOWER_MEM” position.
- Supply power to the AT91EB01 and to the Embedded ICE: the amber LED and the red LED on the EmbeddedICE light up.
- Configure the debugger to work with Embedded ICE:
- Start the ARM Debugger for Windows (ADW) in Armulate Mode
- Activate the menu “Options – Configure Debugger”.
- If not already done, add the “remote_a.dll” target environment to the debugger.
- Select the target environment “remote_a.dll” and configure it:
  - “Serial/Parallel” Remote Connection
  - Heartbeat enabled
  - LPTx port and COMx port at 115200 bauds
- Select OK: the debugger starts communicating with the Embedded ICE, displaying a small window with transmit and receive counters.
- The connection with Embedded ICE is established. In the console window is displayed :

```
EmbeddedICE Manager (ADP, ARM7TDMI) 2.04 (Advanced RISC Machines SDT 2.11)
```

Define the top of memory

- Activate the menu “View – Debugger Internals … “
- Double click on the “top_of_memory” parameter and redefine it to be “0x0220 0000”.

Load the Image and start the application:

- Activate the menu “File – Load Image…”.
- Select the file “led_blink.axf” in the directory “<MyFolderAT91>\Examples\led_blink\EB01SramICE”: the debugger loads the image, displaying a window with transmits and receives counter.
- Select “Execute – Go” or press F5: the debugger stops on the main function, displaying the C source file.
- Select “Execute – Go” or press F5 again: the console window continuously displays “Loop 0, Loop 1,…” and the amber and red LEDs are blinking.

3.6.5 Use with the Multi-ICE Debugger

The procedure is the same as for EmbeddedICE (see “Use with Embedded ICE” on page 9) except that the Multi-ICE Debugger for Windows (MDW) is used in place of ADW. The mode is automatic with Multi-ICE; the configuration of the debugger is useless.

3.6.6 Programming and Executing in Flash

Open the project “led_blink.apj” in the directory “<MyFolderAT91>\Examples\ led_blink”.

Select the variant “EB01Flash”, and build the application.

Power up the AT91EB01 and establish a debugger (any one) with the target. Refer to the sections above.

Execute the programming sequence (as described in the section “Using the Flash Downloader” on page 19):

```
$top_of_memory=0x02200000
load <MyFolderAT91>\Tools\at91eb01\flash.li <MyFolderAT91>\Exam-
ples\led_blink\EB01Flash\led_blink.axf 0x01010000
g
```

The flash programmer displays the following in the Console Window :

```
**** AT91EB01 Flash Programming Utility ****
Trying to identify Flash at base address (0x1010000)
Flash recognised : AT29LV1024
Input file is  :   <filename>
Load address is:   0x1010000
Programming flash ("." = 1 sector)
.................
........
Flash written and verified successfully
```

Start the application:

- Exit from the debugger
- Set the SW1 switch on the UPPER_MEM position
- Reset the AT91EB01

The amber and red LEDs blink.

3.7 Booting with the Flash

3.7.1 Flash Memory Organization

The Flash memory is an AT29LV1024 (64K x 16). It is arbitrarily located at the address 0x01000000 by the boot software (see “Programmer’s Model” on page 23). It has been split in two different spaces :

- 0x01000000 up to 0x0100FFFF = boot and Angel software
- 0x01010000 up to 0x0101FFFF = application software

The switch SW1 allows the signal “SW_A16” to be set (see Schematic 3 “External Bus Interface”) which drives the bit A15 of the flash part:

- “LOWER MEM” uses the bit A16 of the AT91M40400. The entire flash can be reached by the AT91M40400, and the boot software at location 0x01000000 is executed at the reset.
- “UPPER MEM” uses the Vcc. Only the upper 64K can be reached, at the location defined by the application software executed at the reset.

It is important to note that the examples provided by the AT91 library set up the EBI with:

- Flash at address 0x01000000
- SRAM at address 0x02000000

Nevertheless, when SW1 is set with “UPPER_MEM”, these addresses are under user application responsability.

At delivery, the flash is programmed with the following software:

- Boot from 0x01000000 up to 0x01001FFF,
- Angel from 0x01002000 up to 0x0100FFFF, and
- Demo application from 0x01010000 up to 0101FFFF.

3.7.2 Flash Write Access

The upper 64K of the Flash can be overwritten by using the Flash downloader (see “Using the Flash Downloader” on page 19) whatever the position of the switch SW1. This can also be done by using Angel rather than the Embedded ICE.

The lower 64K are write protected whatever the position of the switch SW1. This is to prevent the boot and Angel software stored in the lower 64K bytes from being erased.

Nevertheless, it is always possible to make this space unprotected by setting a jumper or a link on the footprint J7.

Note: If the lower 64K are not protected, the user must be especially careful. Even though one of the features of the boot is the ability to restore Angel, it cannot be saved itself. Once it is overwritten, the only way to restore the Flash is to use an Embedded ICE.

3.7.3 The Boot Software

The boot software is started at the reset if SW1 is switched to ”LOWER MEM”.

It first initializes the EBI (see “Programmer’s Model” on page 23), then executes the REMAP procedure (see the AT91M40400 datasheet), and then checks the state of the buttons:

- If the button FIQ is pressed, all the LEDs are lit, and the Functional Test Software (FTS) is activated
- If the button TIOB1 is pressed, the red LED is lit, and the SRAM downloader is activated
- If neither TIOB1 nor FIQ are pressed, the amber LED is lit and Angel is activated.

The code sources of the boot software are included in the AT91 library in the folder “<MyFolderAT91>\Tools\BootEb01”.

3.7.4 The Functional Test Software (FTS)

The FTS is a part of the boot software which allows testing of the AT91EB01. It is started by the boot if the FIQ button is pressed at reset. At start of the FTS, the LEDs 1, 2 and 3 light up. The user must then release the FIQ button. The LEDs switch off, and then blink once.

The FTS then waits for one of the following user actions on the buttons:

- If the TIOB1 button is pressed, the memory size is checked and the green LED blinks:
  - 1 = 256K bytes
  - 2 = 512K bytes
  - 3 = 1024K bytes
  - 4 = 2048K bytes
- If the FIQ button is pressed, the USART are tested:
  - Transfer one character from USART 0 to USART 0 (loopback mode)
  - Transfer one character from USART 1 to USART 1 (loopback mode)
  - Transfer one character from USART 0 to USART 1 (normal mode)
  - Transfer one character from USART 1 to USART 0 (normal mode)

For each case, the green or the red LED blinks once following the positive or negative result of the test. Note that the 2nd and 3rd tests can succeed only if the straight serial cable (provided) is connected between SERIAL A and SERIAL B.

- If the IRQ0 button is pressed, the buses of the SRAM are tested:

  - Address bus
  - Data bus

  In each case, the green or the red LED blinks once following the positive or negative result of the test.

3.7.5 The SRAM Downloader

The SRAM downloader is a part of the boot software and allows an application in the SRAM to be loaded at the address 0x02000000, then activated. It is started by the boot if the button TIOB1 is pressed at reset.

The procedure is as follows:

1. Connect the AT91EB01 to the host PC using either the straight serial cable (provided) on the serial A, or a crossed serial cable (not provided) on the serial B.
2. Set the clock frequency (LK2) to the desired frequency. Note that this directly determines the baud rate for the download (cf. table below).
3. Generate a power on or a RESET with the TIOB1 pressed down.Wait for the red LED to light up and then release the TIOB1 button.
4. Start the BINCOM utility, located in the folder “<MyFolderAT91>\Tools”, on the host computer: Select the port and the baud rate for communications, then open the file to download and send it. Wait for the end of the transfer.
5. Press the button IRQ0 to terminate the download. The control is switched to the address 0x02000000.

If the downloaded program is Angel, start the ARM debugger (first exit BINCOM). Refer to the section “Communicating with Angel” on page 17 for more details.

The following table shows the relationship between the clock rate on the board, and the serial baud rate:

Table 1. Clock Rate vs. Serial Baud Rate

Frequency	Serial A	Serial B
16 MHz	57600	19200
32 MHz (MASTER)	115200	38400

Note that if the debugger is started through ICE while the SRAM downloader is on, then both serial channels are enabled.

3.7.6 The Angel Monitor

The Angel monitor is located in the flash from 0x01002000 up to 0x0100FFFF. It is started by the boot if neither TIOB1 nor FIQ are pressed at reset. Refer to the section “Communicating with Angel” on page 17 and “Using Angel” on page 19 for more details.

The latest version of the Angel debugger is included in the AT91 library in the folder “<MyFolderAT91>\Tools“, file “Angeleb01.exf”. The file “Angeleb01.txt” contains the corresponding prompt printed in the Console window at Angel startup.

The Angel on the AT91EB01 can be upgraded regardless of the version programmed on it. Refer to the section “Using the Flash Downloader” on page 19 for more details.

Note that if the debugger is started through ICEe while the Angel monitor is on, the Advanced Interrupt Controller (AIC) and the USART channel are enabled.

3.8 Communicating with EmbeddedICE

The following elements are required:

- AT91EB01 Evaluation Board
- EmbeddedICE interface unit with its 14-way ribbon cable

- EmbeddedICE to Multi-ICE adapter (supplied with the EmbeddedICE interface)
- 9-pin RS-232 cable (not supplied with the AT91EB01)
- 25-pin parallel cable (optional, not supplied with the AT91EB01)
- Two DC power supplies, 7.5V - 9V @ 500 mA (not supplied)

3.8.1 Connecting the EmbeddedICE Unit

To communicate between the host computer and the target system using the EmbeddedICE interface:

1. Connect the serial cable between the 9-pin serial port on the EmbeddedICE interface and the serial communications port of the host PC.
2. Connect the parallel port cable between the 25-pin parallel port connector on the EmbeddedICE interface and the printer port on the host PC. (This is not essential, but will speed up download.)
3. Connect the EmbeddedICE interface to an external DC power supply, 500 mA or greater (not supplied).
4. Note that the interface has diode protection to prevent reversed supplies from damaging the board. For correct operation, the 2.1 mm power connector should have the positive supply connected to the center pin.
5. Power up the EmbeddedICE interface and press the red reset button.
6. On releasing the reset button, the LED on the unit lights up. If the LED immediately blinks off once for a fraction of a second, the software version is Remote Debug Protocol (RDP) compatible, not Angel Debug Protocol (ADP) compatible.
7. Power up the evaluation board as detailed above.
8. Connect the EmbeddedICE interface unit to the target evaluation board using a 14-way ribbon JTAG cable and the EmbeddedICE to Multi-ICE adapter. The JTAG connection on the evaluation board is located on the same edge as the “Host PC” DB9 connector and the DC-power connector and is labeled “JTAG” (J2). Refer to the Layout diagram above.

3.8.2 Configuring the ARM Debugger for Windows to Use EmbeddedICE

To configure the ARM Debugger for Windows (ADW) to access a remote target rather than the ARMulator:

1. Start the ADW (not via the ARM Project Manager). In its default state, a message similar to the following is displayed in the console window:

   ```
   ARMULATOR 2.0 [Jan 6 1997]ARM7DTMI, 15.0MHz, 4Gb, Dummy MMU, Soft  Angel
   1.4,[Angel SWIs, Demon SWIs],FPE, Profiler, Tracer, Little endian
   ```

2. Choose “Configure Debugger...” from the Options menu. The Debugger Configuration property sheet is displayed.
3. Click on the “Target” tab.
4. Select “Remote_A” for ADP-compatible EmbeddedICE units.

   Note: If you are using an older RDP version of the EmbeddedICE unit (the LED blinks on-off-on at reset),

5. Add the configuration file `Arm21x\Bin\remote_d.dll` for the Target Environment and select “Remote_D”. *Alternatively, upgrade your EmbeddedICE ROM (see http://www.arm.com for the latest version).*
6. Click on “Configure...”
7. If you are using both serial and parallel cables, select “Serial/Parallel from Remote Connection”. If you are using only a serial cable, select “Serial”.
8. Select the serial and parallel communication ports, and the serial line speed. (The AT91 Evaluation Board operates at up to 38400 bps.)
9. Click on “OK”.

10. To select little-endian mode, click on the Debugger tab, and click on “Little” in the Endian selection.
11. Click on “OK”.

The debugger is restarted. The Restarting dialog box is displayed. Rapidly changing numbers are shown, indicating reading and writing to the target.

Note: If the target board does not respond (the numbers in the Restarting dialog box do not change), ADP times out within approximately five seconds and defaults back to ARMulator.

When the configuration has completed successfully, a message similar to the following is displayed in your host console window:

```
EmbeddedICE Manager (ADP ARM7TDI) 2.01 (Advanced RISC Machines)
```

When you start EmbeddedICE, it automatically defaults to a particular configuration according to which version of the interface software is installed:

- versions up to 1.03 default to ARM7DI
- versions after this default to ARM7TDI

If any of the following messages are displayed when you start up the debugger or reload an image, you must reconfigure the EmbeddedICE:

```
Error during initialization:
Recoverable error in RDI initialization
Target processor not stopped
```

Note: Ensure that the selected configuration is correct, otherwise problems may arise when you run a program.

To configure the EmbeddedICE interface:

1. Choose “Configure EmbeddedICE” from the Options menu.
2. Do one of the following:

   - Choose the name from the list and type in the version. You can usually type any for the version. Click on “Select”.

   - Load a new agent: click on “Load Agent...”, select the required agent file (usually found in *ARM21x\Angel\Images\iceagent*), and then click on “Open”.

   - Load a new configuration file: click on “Load Config...”, select the required configuration file, and then click on “Open”. The required configuration data is usually contained within the agent itself, so this option is seldom used.
3. Click on “OK”.

When the debugger closes down, the remote debugging option settings are stored in memory. Thus at the next use of the debugger, the options may be recalled automatically.

3.8.3 Configuring the armsd Debugger to Use EmbeddedICE

When initializing armsd using EmbeddedICE, it is necessary to tell armsd the endianness of the target. This contrasts with running Angel, or using ARMulator, where there is a default endianness (little-endian).

To configure for a little-endian target using EmbeddedICE version 2.0 or later, the standard serial and parallel ports, and the default linespeed (9600 bps), enter:

```
armsd -li -adp -port s,p
```

To use serial port 2 (rather than serial port 1), and line speed of 38400, enter:

```
armsd -li -adp -port s=2,p -linespeed 38400
```

where:

`-li`	is the little-endian switch
`-adp`	is the switch to allow communication using the ARM Debug Protocol required by Angel

`-port s=2,p` indicates that the serial port is port 2 on the host computer, and the parallel port is the standard port. The serial and parallel ports can be shown numerically (1 or 2) or be the host device name required

`-linespeed 38400` sets the baud rate for serial communications between the host computer and the Development System.

If you are using a version of EmbeddedICE earlier than 2.00 (that uses RDP rather than ADP communication), enter:

```
armsd -li -serpar
```

When you enter the correct command, the host displays the debugger's opening banner, similar to:

```
A.R.M. Source-level Debugger vsn 4.46 (Advanced RISC Machines SDT 2.10)[Dec13
1996] EmbeddedICE Manager (ADP ARM7TDI) 2.01 (Advanced RISC Machines)
```

The following commands enable you to access EmbeddedICE configuration and agent facilities from within armsd:

`listconfig` lists the configurations known to the debug agent. Note that the first configuration listed is the default for the version of the agent in use

`selectconfig` *`name version`* specifies the target configuration to use

where:

`name` indicates the configuration to use, typically ARM7DI or ARM7TDI

`version` indicates the version to be used:

- `any` accepts any version (default)
- `n` must use version n
- `n+` must use version n or later

The highest numbered version meeting the version constraint is used.

`loadagent` *`file`* downloads the replacement agent software into EmbeddedICE and starts it in RAM, rather than using the one in NUM. See the “ARM Software Development Toolkit User Guide” (ARM DUI 0040) for further details

`loadconfig` *`file`* specifies the file containing configuration data to be loaded. Note that this option is seldom needed because the required configuration data is usually contained within the agent itself

For further information on using EmbeddedICE, see the “ARM Software Development Toolkit User Guide (ARM DUI 0040)”.

3.9 Communicating with Multi-ICE

The following is required:

- AT91EB01 Evaluation Board
- Multi-ICE interface unit with its 20-pin ribbon cable
- 25-pin parallel cable
- DC power supply, 7.5V - 9V @ 500 mA (not supplied)

3.9.1 Connecting the Multi-ICE Unit

To communicate between the host computer and the target system using the Multi-ICE interface:

1. Connect the parallel port cable between the 25-pin parallel port connector on the Multi-ICE interface and the printer port on the host PC.
2. Connect the Multi-ICE interface unit to the target evaluation board using the 20-way ribbon JTAG cable. The JTAG connection on the evaluation board is located on the same edge as the “Host PC” DB9 connector and the DC-power connector and is labeled “JTAG” (J2).
3. Power up the evaluation board as described above.

3.9.2 Configuring the Multiprocessor Debugger for Windows to Use Multi-ICE

To configure the Multiprocessor Debugger for Windows (MDW) to access a remote target rather than the ARMulator:

1. Start Portmap.
2. Start the Multi-ICE Server.
3. When the Multi-ICE Server has loaded, click on the auto-configure icon. This should detect the ARM7TDMI RISC core. If not, manually select it - refer to the Multi-ICE manual.
4. Load MDW.
5. Choose “Configure Debugger...” from the Options menu. The Debugger Configuration property sheet is displayed.
6. Click on the “Target” tab.
7. If this is the first time using Multi-ICE, click on “add” and select multi-ice.dll. If not jump to step 9.
8. Click on “OK”.
9. Select “Multi-ICE”.
10. Click on “Configure...”
11. In the “Location” box, enter “localhost” if the host PC is the PC you are using or, if another PC is the host, enter the Host PC’s network name.
12. Check that the correct core is shown in the “Processor Driver Selection” box.
13. Enter a name for the “Connection name”. The user’s name is a good idea.
14. Click on “OK”.
15. Select little-endian mode (the AT91 Evaluation Board is little-endian only), click on the Debugger tab, and click on “Little” in the Endian selection.
16. Click on “OK”.

The debugger is restarted. The Restarting dialog box is displayed. Rapidly changing numbers are shown, indicating reading and writing to the target.

Note: If the target board does not respond (the numbers in the Restarting dialog box do not change), ADP times out within approximately five seconds and defaults back to ARMulator.

Before running the application, set the variable:

```
$top_of_memory=0x02200000
```

This can be done using the “Command” window or the “Debugger Internals” window.

3.10 Communicating with Angel

Note: System development using Angel is subject to special considerations. These are documented in “Application Note 43: Design Considerations when using Angel Debug Monitors” (ARM DAI 0043).

3.10.1 Serial Host Connection

Connect the host computer to the Serial A port USART on the AT91 Evaluation Board using a 9-pin RS-232 cable. Power up the board (see “Powering Up the Board” on page 8).

3.10.2 Configuring the ARM Debugger for Windows to Use Angel

To configure the ARM Debugger for Windows (ADW) to access Angel rather than the ARMulator:

1. Start the ADW (not via the ARM Project Manager). In its default state, a message similar to the following is displayed:

   ```
   ARMULATOR 2.0 [Jan 6 1997]ARM7DTMI, 15.0MHz, 4Gb, Dummy MMU, Soft  Angel
   1.4,[Angel SWIs, Demon SWIs],FPE, Profiler, Tracer, Little endian
   ```

2. Choose “Configure Debugger...” from the Options menu.
3. Click on the “Target” tab.
4. Select “Remote_A”.
5. Click on “Configure...” and select “Serial”.
6. Select the serial communication port and line speed. (The AT91 Development Board operates at up to 57600 bps).
7. Click on “OK”.

The debugger is restarted. The Restarting dialog box is displayed. Rapidly changing numbers are shown, indicating reading and writing to the target.

Note: If the target board does not respond (the numbers in the Restarting dialog box do not change), ADP times out within approximately five seconds and defaults back to ARMulator. When the configuration has completed successfully, a message similar to the following is displayed in the host console window:

```
  Angel Debug Monitor (serial) 1.04 (Advanced RISC Machines SDT 2.11a) for
AT91EB0x(1.01)
  Angel Debug Monitor rebuilt on Sep. 14 1998 at 14:52:22
  Serial Rate: 38400
```

3.10.3 Configuring the armsd Debugger to Use Angel

To operate armsd with Angel using a serial connection, type:

```
armsd -li -adp -port s=1 -linespeed 38400
```

where:

`-li`	is the little endian switch
`-adp`	is the switch to allow communication using ARM Debug Protocol required by Angel
`-port s=1`	indicates that the serial port is port 1 on the host computer.Type the serial port number (1 or 2) or the host device name
`-linespeed 38400`	sets the baud rate for serial communications between the host computer and the Development System

When this command is entered, the host displays the debugger's opening banner, which will be similar to:

```
A.R.M. Source-level Debugger vsn 4.46 (Advanced RISC Machines SDT2.10) [Dec
13 1996] Angel Debug Monitor for AT91 (serial) 1.04 (Advanced RISC Machines
SDT 2.11a)
```

Note: The options available for armsd can be viewed by typing armsd -h at the command line.

3.11 Using Angel

Angel has been ported on the AT91EB01 so that the majority of features of the AT91M40400 are available to the application. The Angel Debug Monitor is intrusive. Therefore, this implies some limitations on the use of the resources:

- USART 0 is reserved for communications with the debugger.
- A Flash space is used for code saving.
- A SRAM space is used for code running, data and stacks of Angel.
- Care must be taken with the stacks and the interrupt handling.

3.11.1 Memory Mapping

Angel is stored in the Flash from address 0x0100 2000, and occupies with the boot the entire lower 64K bytes of the Flash. When started, Angel automatically copies itself into the SRAM at address 0x0200 8000. This allows faster execution than if running from the Flash; moreover, Angel can be used by the Flash programmer.

Stacks and variables needed by Angel are based from 0x0200 0000 up to 0x0200 7FFF. As 64K bytes are reserved for Angel and its future versions, it is mandatory to link any application running in SRAM at addresses higher than 0x0201 8000. Otherwise, the download of the image will overwrite Angel's code, and communications between Angel and the debugger will be lost.

3.11.2 Interrupt Handling

The Angel's interrupt handling uses the vectoring feature of the AT91's Advanced Interrupt Controller. This implies that the ARM Interrupt vector (at address 0x18) is set with the instruction "ldr pc, [pc, #&-F20]" in order to access AIC_IVR.This is assumed by Angel itself.

If the user wants to use Interrupt Vectoring, he just has to store the handler addresses in the AIC_SVR, and to set the AIC_SMR.

If he does not want to use this feature, he has to replace the ARM vector, then in the new defined interrupt handling, to test if the interrupt which has occurred comes from the USART 0 (Interrupt source number 2) and, in this case, branch at address saved in AIC_SVR[2].

3.11.3 Stacks Setup

As Angel needs to manage interrupts, it initializes the Interrupt stack pointer (r14_irq). This register cannot be modified by the application without disagreement on the Angel communications.

Other stack pointers are also initialized by default by the Angel startup sequence. Particularly, user mode stack is initialized at the top of the memory. As this last differs depending on the version of the AT91EB01, it is fixed at 0x0204 0000.

3.12 Using the Flash Downloader

3.12.1 Flash Downloading with ADW & MDW

Downloading into Flash overwrites any existing image already stored there. This may be a version of Angel, or an application program. Ensure that the download completes successfully before you reset or power off the board.

The flash downloader for the AT91EB01 is based on the flash downloader utility provided with the ARM Software Development Toolkit. With Angel, it works only if Angel is running from RAM, and not from the flash that is to be programmed. Alternatively EmbeddedICE or Multi-ICE can be used rather than Angel.

The flash downloader is included by the AT91 library in the folder "<MyFolderAT91>\Tools\FlashPgm\at91eb01", file "flash.li". This file can be used with Angel as well as with EmbeddedICE or Multi-ICE.

The download can be done by different ways :

- with an ICE (Embedded or Multi).
- with Angel started in flash by the boot and auto-copied in SRAM before execution (refer to the section Memory Mapping in “Using Angel” on page 19).
- with Angel downloaded and started by the SRAM downloader (see “The SRAM Downloader” on page 13). The file to download is “<MyFolderAT91>\Tools\Angeleb01.axf”.

Angel is started by pressing the button IRQ0, then the connection with the debugger can be established.

Once the connection is established between the debugger on the PC and Angel/ICE on the target, following steps must be performed to download a new image in flash :

1. Set the top of the memory for the debugger (ADW & MDW only):
   `$top_of_memory=0x02200000`
2. Load the flash programmer and set the arguments :
   `load <MyFolderAT91>\Tools\FlashPgm\at91eb01\flash.li <filename> <address>`

where:

`filename`	is the file name (including the path) to download and program
`address`	is the address where the file is programmed (generally 0x01010000)

3. Activate the debugger:
   `g`

The flash programmer displays the following in the Console Window :

```
**** AT91EB01 Flash Programming Utility ****
Trying to identify Flash at base address (0x1010000)
Flash recognised : AT29LV1024
Input file is  :   <filename>
Load address is:   0x1010000
Programming flash ("." = 1 sector)
.................
........
Flash written and verified successfully
```

The sequence described above can be automatically executed by using a command file. Build a file `<command_file>` with the commands :

```
$top_of_memory=0x02200000
load <MyFolderAT91>\Tools\FlashPgm\at91eb01\flash.li <filename>
<address>
g
```

and then execute the file by entering the command:

```
ob <command_file>
```

Another way is by using the item “Flash Download” in the menu “File” of the debugger. For this, the file “flash.li” must be previously copied in the current directory or in the “bin” directory of the ARM setup. The arguments are `<filename>` and `<address>`.

In the case where the boot software has been overwritten, and before any use of the flash downloader, the EBI must be programmed :

EBI_CSR0:	`0xFFE00000=0x01002535`
EBI_CSR1:	`0xFFE00004=0x02002121`
REMAP done:	`0xFFE00020=1`
2M bytes per CS:	`0xFFE00024=6`

A command file called “ebi_eb0x” which performs the default EBI configuration is available in “<MyFolderAT91>\Tools”.

3.12.2 Upper 64K Flash Downloading

The upper 64K Flash can be used to program then execute an application software (see the section “Booting with the Flash” on page 11).

To have the application software executed at reset, the switch SW1 must be set with the position “UPPER MEM”. Therefore, the image must be generated with a “Read only” space based at the address 0x01000000.

To download the application software in Flash, the switch SW1 must be set with the position “LOWER MEM”. The argument `<address>` of the `<load>` command must be 0x01010000.

The table below summarizes the address of the upper 64K Flash following the action:

Action	Button SW1	Address
Downloading	UPPER MEM	0x01010000
Executing	LOWER MEM	0x01000000

Note: The address 0x0100000 for the “Read Only” space has been chosen so as to differentiate it from the address used by the boot software. Since the RESET is done by the application itself, any address above 0x00400000 (and allowed by the EBI) can be chosen.

3.12.3 Lower 64K Flash Downloading

The lower 64K of the Flash can be downloaded (e.g. to update an Angel version). To do so, the switch SW1 must be set with the position “LOWER MEM”, and the lower 64K must be unprotected by setting a jumper or a link on the footprint J7 (see section “Booting with the Flash” on page 11). In any case, when an attempt to download the lower 64K is done, the Flash downloader asks for a confirmation.
Extra caution is required when the lower 64K are not protected. If the download fails or if the Flash is overwritten:

- Angel can be restored by using the SRAM downloader (see “Angel Upgrade” on page 21)
- the boot can be restored only by using an ICE (Embedded or Multi)

3.12.4 Angel Upgrade

Angel can be upgraded in the lower part of the flash. For this, it is mandatory to set a jumper on J7 in order to allow the writes to the lower part of the flash (see section “Lower 64K Flash Downloading” on page 21).

Execute the sequence described in the section “Flash Downloading with ADW & MDW” on page 19. The command is :

```
load <flash> <angel> 0x01002000
```

where:

`flash` is the file “Tools\FlashPgm\at91eb01\flash.li” including the path
`angel` is the file “Tools\Angeleb01.axf” including the path
`0x01002000` is the address where Angel is started by the boot

Section 4

Programmer's Model

4.1 Memory Map

The memory map can easily be changed by reprogramming the external bus interface. This section details the default memory map that is set up by the Angel debug monitor.

Table 2. Memory Map

Name	Address	Size
Peripheral Devices	HFFFFFFFF HFFD00000	3M bytes
undefined		
External Memory [7]	H7FFFFFFF H70000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [6]	H6FFFFFFF H60000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [5]	H5FFFFFFF H50000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [4]	H4FFFFFFF H40000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [3]	H3FFFFFFF H30000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [2]	H2FFFFFFF H20000000	1, 4, 16 or 64M bytes - not used
undefined		
External Memory [1]	H0FFFFFFF H02000000	256K bytes to 2048K bytes 16-bit SRAM - 4M bytes page size
undefined		

Table 2. Memory Map (Continued)

Name	Address	Size
External Memory [0]	H01FFFFFF H01000000	128K bytes 16-bit Flash - 1M byte page size
undefined		
On-chip RAM (reboot)	H00300FFF H00300000	4K bytes
Reserved On-chip device	H002FFFFF H00200000	1M byte
Reserved On-chip device	H001FFFFF H00100000	1M byte
On-chip RAM (normal) or External ROM (reboot)	H00000FFF H00000000	4K bytes

Note: For the peripheral memory map, refer to the AT91M40400 datasheet.

Section 5

Circuit Description

5.1 AT91EB01 Top Level

The top level schematic in “Appendix B - Schematics” on page 29 shows the blocks in the system. Each block is described in the appropriate section below.

5.2 AT91M40400 Processor

This schematic shows the AT91M40400. The footprint is for a 100-pin TQFP package.

Wirelink (WL2) can be removed by the user to allow measurement of the current demand by the microcontroller.

5.3 I/O Expansion

The I/O Expansion connector makes available to the user the General Purpose I/O (GPIO) lines, VDD and Ground. Surface mount links 5, 6, 7, 8 and 9 are used to select between the I/O lines being used by the evaluation board or by the user via the I/O expansion connector. The connector is not fitted at the factory; however, the user can fit any 17 x 2 connector on a 0.1" (2.54 mm) pitch.

5.4 External Bus Interface

This schematic shows one AT29LV1024-15JC 128K bytes 16-bit Flash and four 512K x 8 SRAM devices.

Note: The AT91EB01 is fitted with four 128K x 8 SRAM devices.

The schematic also shows the Bus Expansion connector which, like the I/O Expansion connector, is not fitted at the factory. The user can fit any 32 x 2 connector on a 0.1" (2.54 mm) pitch to gain access to the data, address, chip select, read/write, oscillator output, and wait state pins. Pin 8 on this connector can be used to apply an external clock frequency to the board assuming the clock select jumper is fitted accordingly (see Appendix A). VDD and Ground are also available on the connector.

Switch 1 shown on this schematic is used to select either Read only access of all locations in Flash or allow Read/Write access to the top 64K bytes. This is to prevent the Angel Debug Program which is stored in the lower 64K bytes from being erased.

5.5 Power, Crystal Oscillator and Clock Distribution

The system clock is derived from a single 32.768 MHz crystal oscillator. This is divided by a 4-bit binary counter to give alternate clock frequencies of 32.768 MHz divided by 2, 4 or 8. The system clock frequency is selected by fitting a jumper link in one position of the link field (LK2) and details of this can be found in Appendix A. One position in LK2 selects an external oscillator to be applied via the Expansion Bus Interface.

The Voltage Regulator provides 3.3V to the board and will light the red POWER LED when operating. Power can be applied via the 2.1 mm connector to the regulator in either polarity because of the diode rectifying circuit. The regulators can tolerate supply transients to 30V although they will shut down without damage if they overheat.

5.6 JTAG Interface, Reset, Interrupts & LEDs

An ARM-standard 20-pin box header (J2) is provided to enable connection of an EmbeddedICE interface (using an EmbeddedICE to Multi-ICE adapter) or Multi-ICE interface to the JTAG inputs on the AT91. This allows code to be developed on the board without the use of system resources such as memory and serial ports.

The IRQ, FIQ and TIOB0 switches are debounced and buffered. A supervisory circuit has been included in the design to detect and consequently reset the board when the 3.3V supply voltage drops below 3.08V. It also provides a debounced reset signal.

The schematic also shows LEDs 1, 2 and 3 which are for general purpose use and are connected to timer (or I/O) pins TIOA0/P1, TIOA1/P4 and TIOB0/P2.

5.7 Serial Interface

Two 9-way D-type connectors (J5/6) are provided for serial port connection. Serial port A (J5) is used primarily for Host PC communication and is a DB9 Female connector. TXD and RXD are swapped so that a straight through cable can be used. CTS and RTS are connected together as is DCD, DSR and DTR.

Serial port B (J6) is a DB9 male connector with TXD and RXD obeying the standard RS-232 pinout. Apart from TXD, RXD and ground, the other pins are not connected.

RS-232 level conversion is provided by a MAX3223 device (U13) and associated bulk storage capacitors.

5.8 Layout Drawing

The layout diagram schematic shows an approximate floorplan for the board. This has been designed to give the lowest board area, whilst still providing access to all testpoints, links and switches on the board.

The size is approximately 4.5 inches square with four mounting holes, one at each corner, into which feet are attached. The board has two signal layers and two power planes.

Section 6

Appendix A - Configuration Links and SRAM Bank 1

6.1 Jumpers (LK2)

The LK2 jumper is used to select the clock frequency. The frequency options are 32,768 MHz, 32.768 MHz divided by 2, 4 or 8, or an external clock applied via the Expansion Bus interface.

LK2

1	3	5	7	9
2	4	6	8	10
EBI	**MAST**	**/2**	**/4**	**/8**

6.2 Surface Mount Links (LK1, LK3-9)

By adding the I/O expansion and the Bus Expansion Connectors the user can add own peripherals to the evaluation board. These peripherals may require more I/O lines than available while the board is in its default state. Extra I/O lines can be made available by disabling some of the on-board peripherals. This is done using the surface mount links detailed below. If these links need to be changed then a fine-tipped soldering iron is required. All links are 0R 0805 resistors unless stated otherwise.

LK1 should be left unaltered unless a memory upgrade is required. If this is the case, see the section headed “Increasing Memory Size” on page 28.

LK1	Function
A-C	A18 is used as the chip select for SRAM bank1. Should be in this position for 128K x 8 SRAMs
B-C	A20 is used as the chip select for SRAM bank1. Should be in this position for 512K x 8 SRAMs

LK3	Function
A-C	Alternate boot mode NOT selected
B-C	Alternate boot mode selected

Note: Use a 10K resistor

LK4	Function
A-C	RXD lines are set tri-state on the RS-232 transceiver so that the receiver signals to AT91 can be used for external I/O via the I/O connector
B-C	Receive pins on AT91 used by on board serial ports

Note: Use a 10K resistor

LK5	Function
A-C	TXD0 connects to I/O connector
B-C	TXD0 connects to RS-232 Transceiver

LK6	Function
A-C	AT91 IRQ0 pin connects to I/O connector
B-C	AT91 IRQ0 input is from IRQ switch

LK7	Function
A-C	AT91 FIQ pin connects to I/O connector
B-C	AT91 FIQ input is from FIQ switch

LK8	Function
A-C	TXD1 connects to I/O connector
B-C	TXD1 connects to RS-232 Transceiver

LK9	Function
A-C	TIOB1 connects to I/O connector
B-C	AT91 TIOB1 input is from TIOB1 switch

6.3 Increasing Memory Size

The AT91EB01 is supplied with four 128K x 8 byte SRAM memories. If, however, the user needs more than 512K bytes of memory, the devices can be replaced with four 512K x 8 3.3V 15/20 ns SRAMs giving in total 2048K bytes. If this is done, LK1 must be moved to position B-C.

Note: The user must either have 128K x 8 SRAMs **or** 512K x 8 SRAMs fitted in Bank 0 and, if desired, Bank 1. The two SRAM sizes cannot be mixed.

Section 7

Appendix B - Schematics

The following schematics are appended:

Figure 3. AT91M40400 Evaluation Board

Figure 4. PCB Layout

Figure 5. External Bus Interface

Figure 6. JTAG Interface, Reset, Interrupts and LEDs

Figure 7. I/O Expansion

Figure 8. Power, Crystal Oscillator and Clock Distribution

Figure 9. AT91M40400 in TQFP Package

Figure 10. Serial Interface

The pin connectors are indicated on the schematics:

- J1 = EBI Expansion - External Bus Interface (Figure 5)
- J2 = JTAG Interface - JTAG Interface, Reset, Interrupts & LEDs (Figure 6)
- J3 = I/O Expansion - I/O Expansion (Figure 7)
- J5 = Serial A - Serial Interface (Figure 10)
- J6 = Serial B - Serial Interface (Figure 10

Figure 3. AT91M40400 Evaluation Board

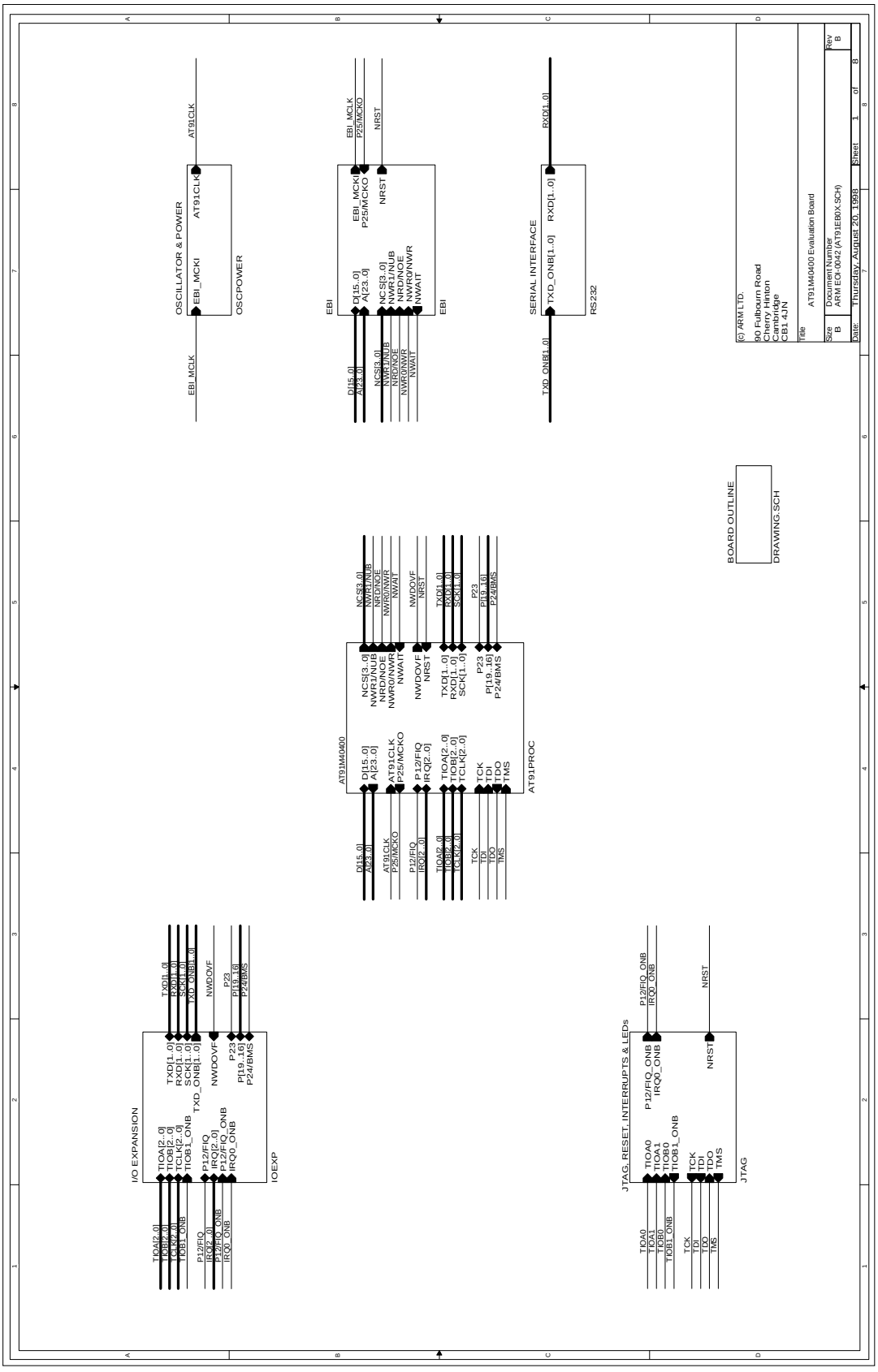


Figure 4. PCB Layout

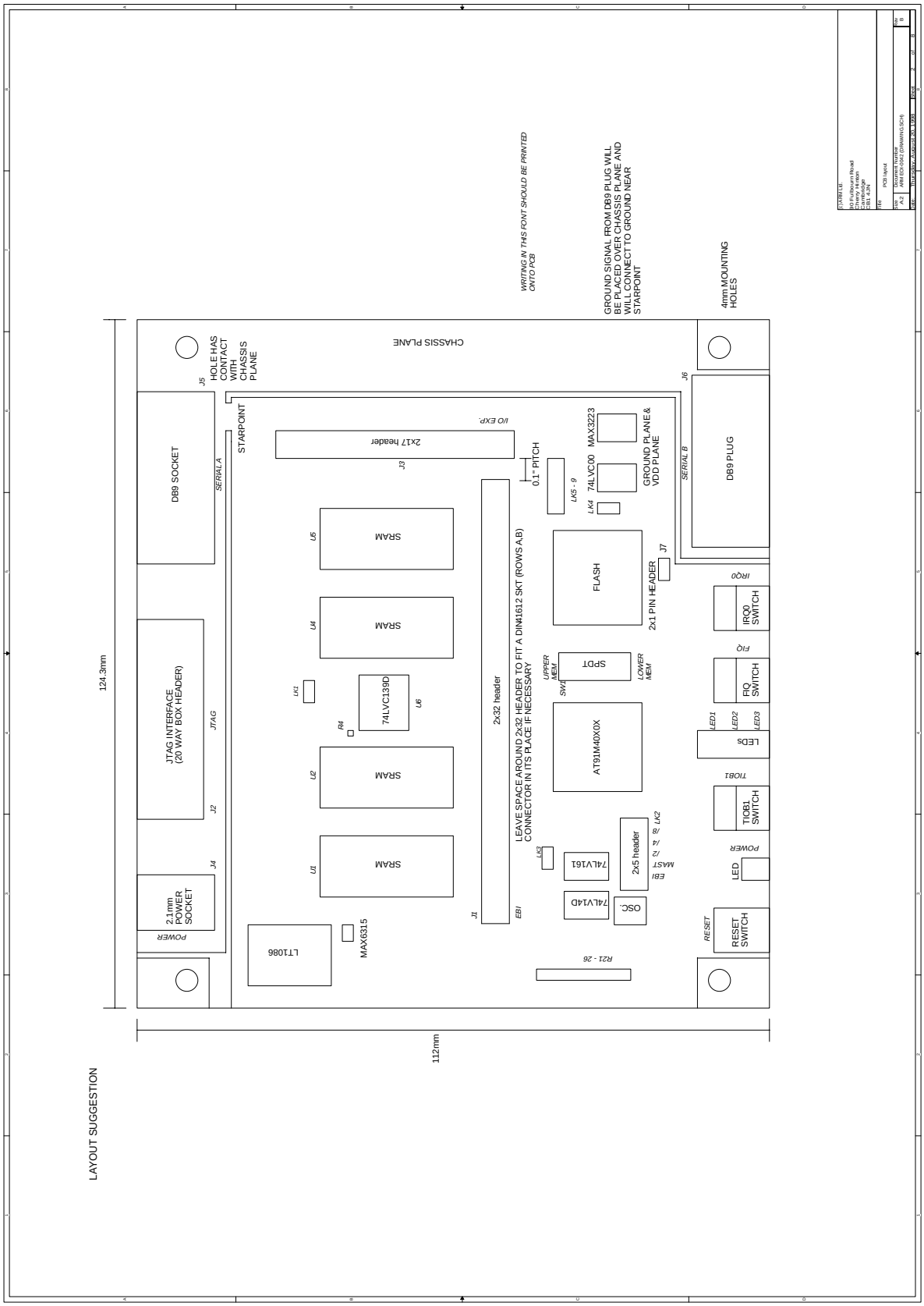


Figure 5. External Bus Interface

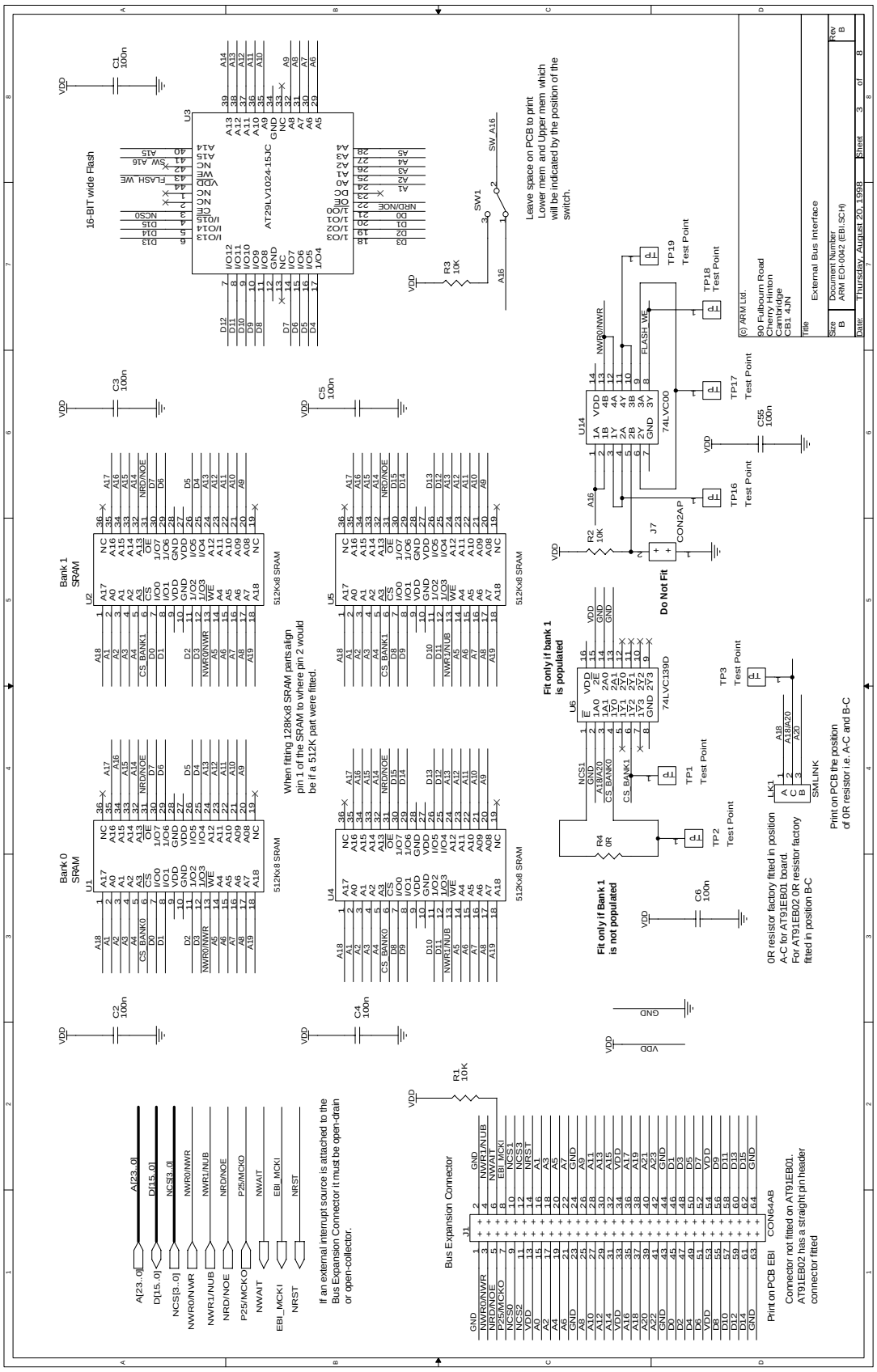


Figure 6. JTAG Interface, Reset, Interrupts and LEDs

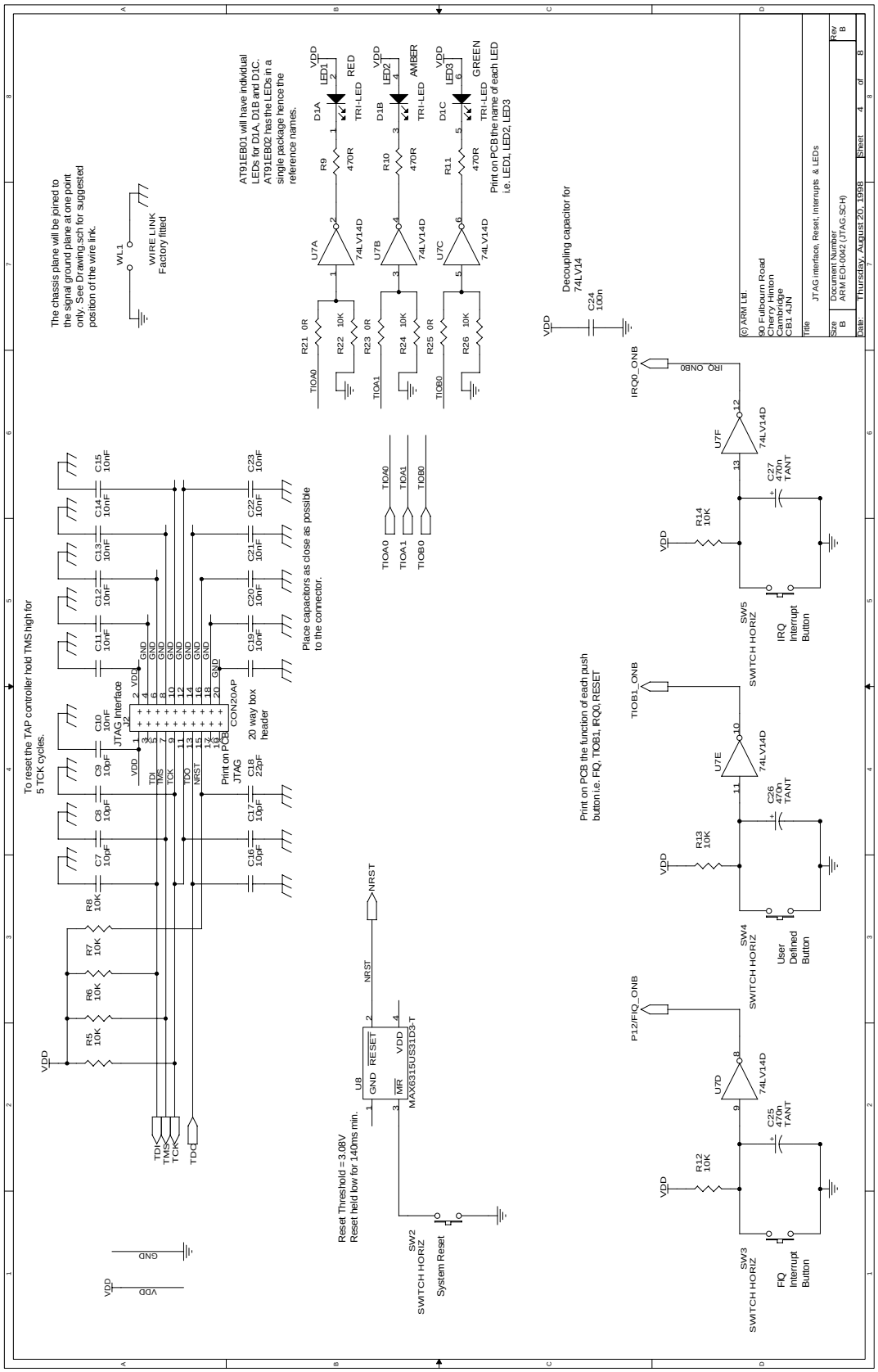


Figure 7. I/O Expansion

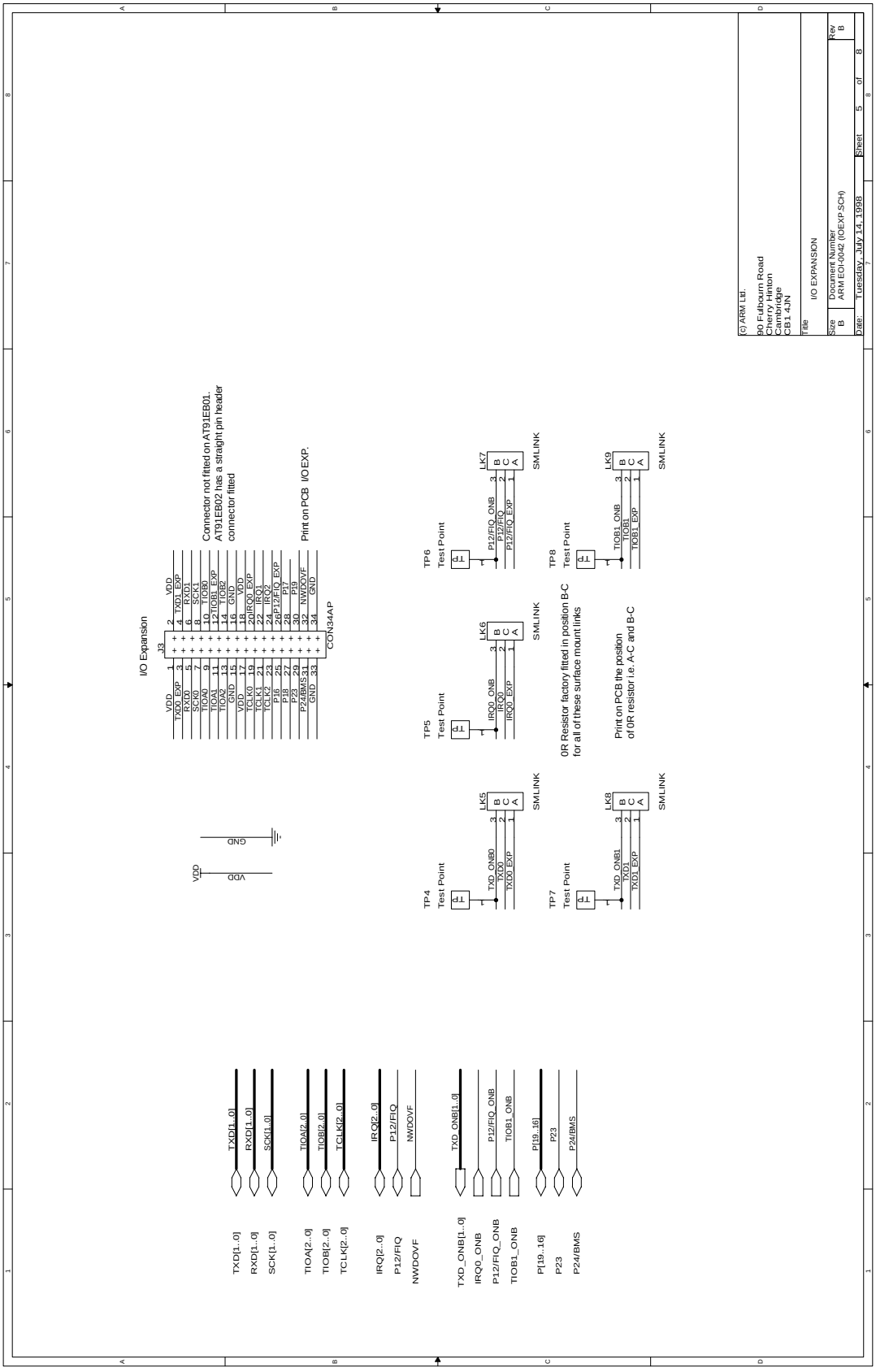


Figure 8. Power, Crystal Oscillator and Clock Distribution

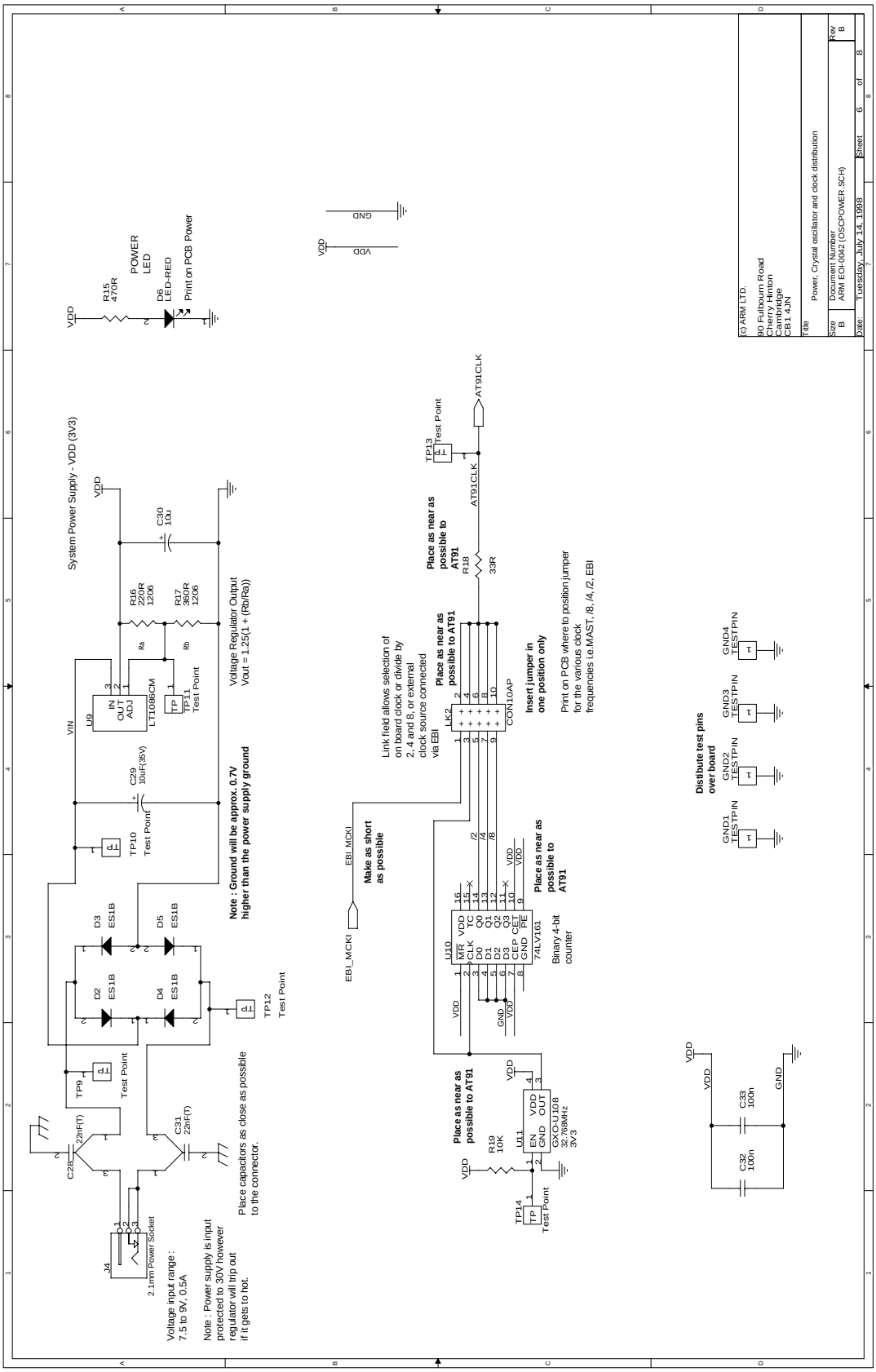


Figure 9. AT91M40400 in TQFP Package

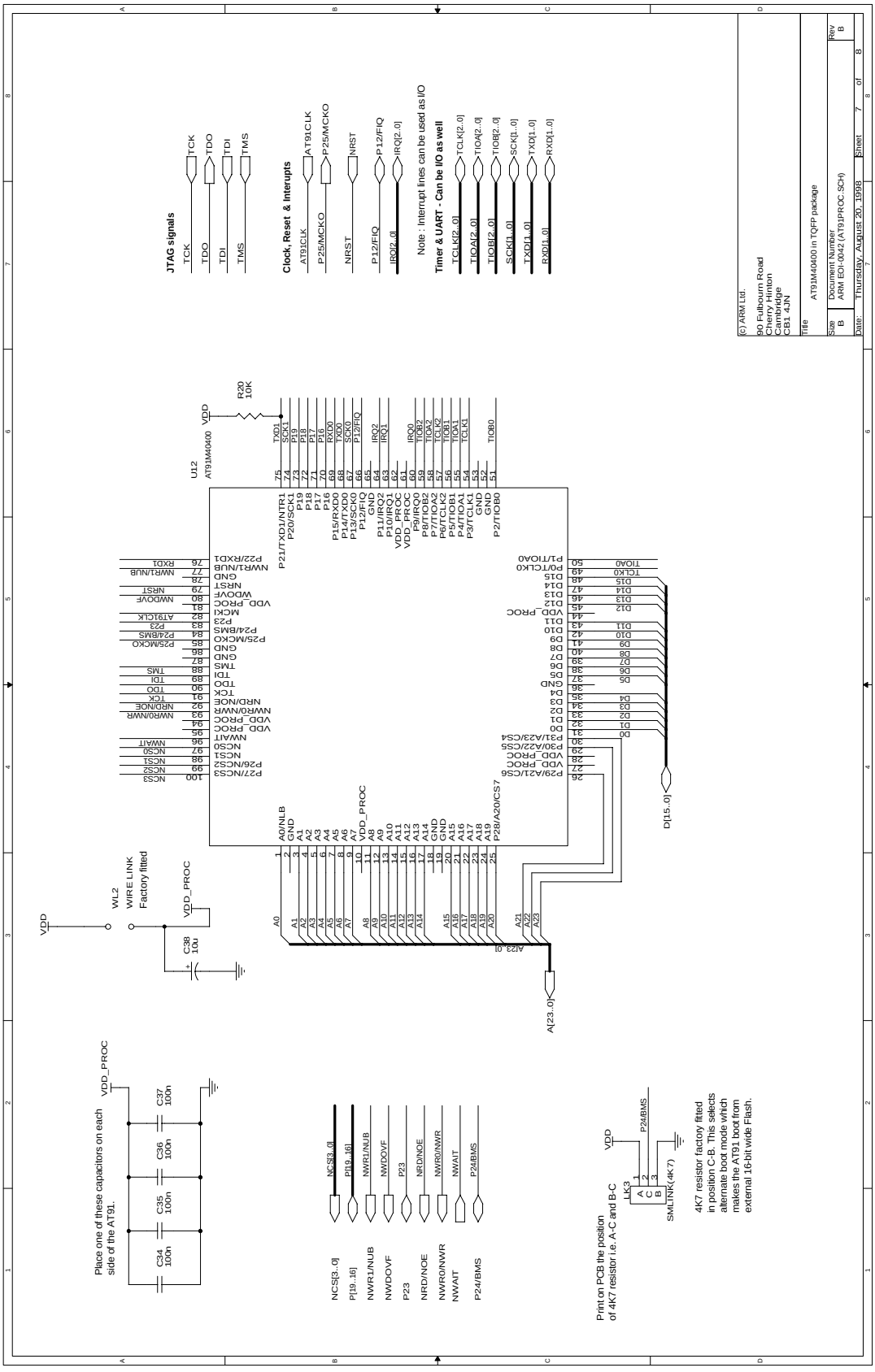


Figure 10. Serial Interface