SIMTEC
ELECTRONICS

# IM2440D20 User Guide

**Simtec Electronics**

**B J Dooks**
**V R Sanders**

# IM2440D20 User Guide

Simtec Electronics
by B J Dooks and V R Sanders
Copyright © 2006 Simtec Electronics


- ARM® is a registered trademark of ARM Limited.
- LINUX® is a registered trademark of Linus Torvalds.
- UNIX® is a registerd trademark of The Open Group.
- All other trademarks are acknowledged.

The product described in this document is under continuous development and improvement. All descriptions of usage and particulars of the product are given in good faith by Simtec Electronics. However all warranties expressed or implied, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.


Revision History

| | | |
|---|---|---|
| Revision 1.00 | 10th April 2006 | VRS |

Initial Release.

# Table of Contents

IM2440D20 Development Board ..... xi
1. Overview ..... 1
1.1. Kit contents ..... 1
1.1.1. CD-ROM ..... 1
1.1.2. IM2440D20 module ..... 1
1.2. Development Tools ..... 2
1.3. Using the module ..... 2
1.4. Handling precautions ..... 2
2. Hardware Description ..... 3
2.1. Hardware overview ..... 3
2.2. Samsung S3C2440 SOC ..... 3
2.3. Memory ..... 4
2.3.1. EEPROM ..... 4
2.3.2. NAND boot flash ..... 4
2.3.3. SDRAM ..... 5
2.4. Power Supply ..... 5
2.5. Edge connector ..... 6
3. Bootloader ..... 11
3.1. Overview ..... 11
3.2. Getting Started ..... 11
3.2.1. Using hyperterm as a serial console ..... 12
3.2.2. Using minicom as a serial console ..... 13
A. Board Layout ..... 15
B. Mechanical drawing ..... 17

# List of Figures

2.1. Detailed block diagram of the IM2440D20 ..... 3
2.2. NAND flash to S3C2440 attachment ..... 5
2.3. SDRAM to S3C24440 attachment ..... 5
2.4. Edge connector placement ..... 6
3.1. Hyperterm settings window ..... 12
3.2. Hyperterm displaying ABLE output ..... 13
3.3. Minicom settings window ..... 13
A.1. EB675001DIP board layout top side ..... 15
A.2. EB675001DIP board layout bottom side ..... 15
B.1. IM2440D20 Mechanical Drawing ..... 17

# List of Tables

2.1. 200 way edge connector PL1 ..... 6

# List of Examples

3.1. Video display after starting ABLE on EB2410ITX ..... 11
3.2. Serial display after starting ABLE on EB2410ITX ..... 11

# IM2440D20 Development Board

**About this document.** This document describes the Simtec IM2440D20 integrated module which provides a flexible devlopemnt system for both experimentation and intergrator solutions.

**Intended Audience.** This document is aimed at experienced engineers.

**Related documents.** Some additional documents which may be useful:

Online resources [http://www.simtec.co.uk/products/IM2440D20/resources.html]
Connector and link pinouts. [http://www.simtec.co.uk/products/IM2440D20/files/pinlist.html]
Memory map and control registers. [http://www.simtec.co.uk/products/IM2440D20/files/mmap.html]
Mechanical Drawing [http://www.simtec.co.uk/products/IM2440D20/files/IM2440D20-Mechanical.pdf]

**Feedback.** Any suggestions, comments or corrections concerning this document are welcomed, please contact Simtec Electronics giving:

The document title
The document revision
A clear explanation of your comments and how they apply

# Chapter 1. Overview

This chapter describes:

- The kit contents
- Development tools
- Use of the module
- Handling precautions

## 1.1. Kit contents

The IM2440D20 is a comprehensive ARM computing platform. The Kit contains:

- The IM2440D20 user guide
- A CD-ROM containing development software and documents relevant to the IM2440D20
- The IM2440D20 module.

### 1.1.1. CD-ROM

The CD-ROM contains:

- A copy of all the freely available documentation, including this user guide.
- Datasheets for all major components used
- Debain Linux distribution
- x86 cross building toolchain for GNU/Linux
- ABLE bootloader

The toolchain contains a GCC compiler, assembler and linker suitable for cross compiling ARM binaries from an x86 machine running GNU/Linux.

### 1.1.2. IM2440D20 module

The IM2440D20 board (gold specification) has the following major components:

- Samsung S3C2440 ARM 920T SOC
- 128MB SDRAM
- 512MBit NAND Flash
- 8Kbit I2C EEPROM
- Three RS232 serial ports
- JTAG
- 30 GPIO or special function lines

- Full core power control

# 1.2. Development Tools

The development tools provided must be installed and run on a PC with a GNU/Linux Operating system (e.g. Debian, Ubuntu or Redhat distributions). The GCC toolchain provided creates executables that can be run on the IM2440D20. This toolchain is also required to build the kernel and Embedded Linux. Full details of building Linux are provided on the IM2440D20 resources page.

In order to run the development tools the host PC requires:

- 500MHz or faster processor
- Installed GNU/Linux distribution
- 128MB RAM
- 1GB of hard disk space (3GB if building uCLinux)
- CD-ROM Drive

In addition to the development tools, the module usually requires serial communications to access its bootloader and booted system. Most modern PC have serial ports, however increasingly only USB ports are provided, most typical USB->Serial converters appear to work with the IM2440D20.

# 1.3. Using the module

The IM2440D20 is a flexible system but does require a baseboard to be fully useful. With the addition of a carrier board, a suitable power supply and PC the Kit provides everything required to start producing the desired solution. The ABLE bootloader allows a flexible use of the IM2440D20. ABLE can start ARM Linux images from a variety of sources including a network interface. In this mode of usage the serial port is used as a console to communicate with the device.

# 1.4. Handling precautions

This module is intended for use either within a workshop/laboratory environment or as an integrator solution within a larger product. Because of this the IM2440D20 module is supplied without an enclosure. The lack of an enclosure means that standard electrostatic control procedures should be used when handling the board. When using the IM2440D20 outside an enclosure the following is recommended:

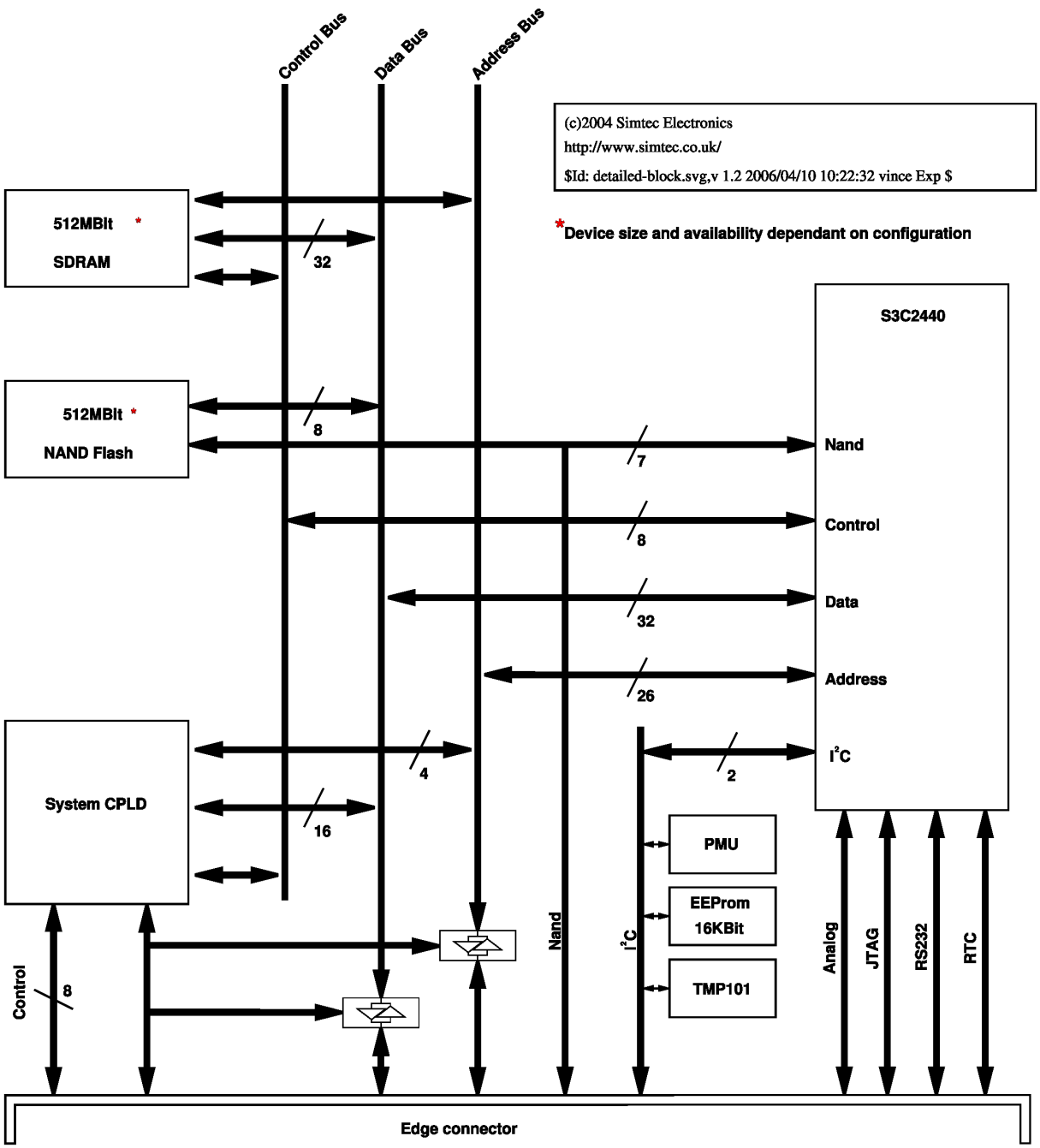
- Only hold the board by the edges
- Always use proper static handling equipment, as a minimum an earth strap

# Chapter 2. Hardware Description

## 2.1. Hardware overview

The IM2440D20 is a complete system on a module that includes a large number of Input/Output facilities. The module must be mounted in a carrier board to give access to the I/O.

**Figure 2.1. Detailed block diagram of the IM2440D20**




## 2.2. Samsung S3C2440 SOC

The S3C2440 is a 289 ball fine pitch Ball Grid Array (BGA) device from Samsung. The large number of peripherals within the device give flexibility to the user and the ability to use numerous I/O solutions without additional

controllers. The S3C2440 has a full MMU and high bandwidth memory system allowing a range of standard operating systems to be run. The S3C2440 has a Harvard architecture 16KByte Data/16KByte instruction cache which further improves performance. The extensive power management systems within the SOC allow for very flexible power control and with a typical power usage of 0.2W at 400MHz operating speeds contributes towards the IM2440D20 low overall power usage.

Simtec provide Debian GNU/Linux as standard which provides a full development environment identical in look and feel to an x86 install of Debian GNU/Linux.

The large number of peripherals and interfaces are summarised as follows:

- ARM920T 32-bit RISC CPU
- 32-bit mode (ARM) and/or 16-bit mode (Thumb)
- Built-in SDRAM external memory controller supports glueless connectivity to memory.
- External memory controller supporting external flash
- 24 external interrupt sources
- Four DMA channels with external access
- One 16-bit system timer
- Four 16-bit PWM channels
- 130 General purpose I/O ports with 24 external IRQ sources
- Eight 10-bit Analog channels
- Three UARTs with 64byte FIFOs
- Two SPI channels
- One I2C bus interface
- One I2S audio CODEC interface
- SD/MMC interface
- Real Time Clock (RTC)

For further details on the S3C2440A please consult the Samsung website [http://www.samsung.com/Products/Semiconductor/SystemLSI/MobileSolutions/MobileASSP/MobileComputing/S3C2440A/S3C2440A.htm]

# 2.3. Memory

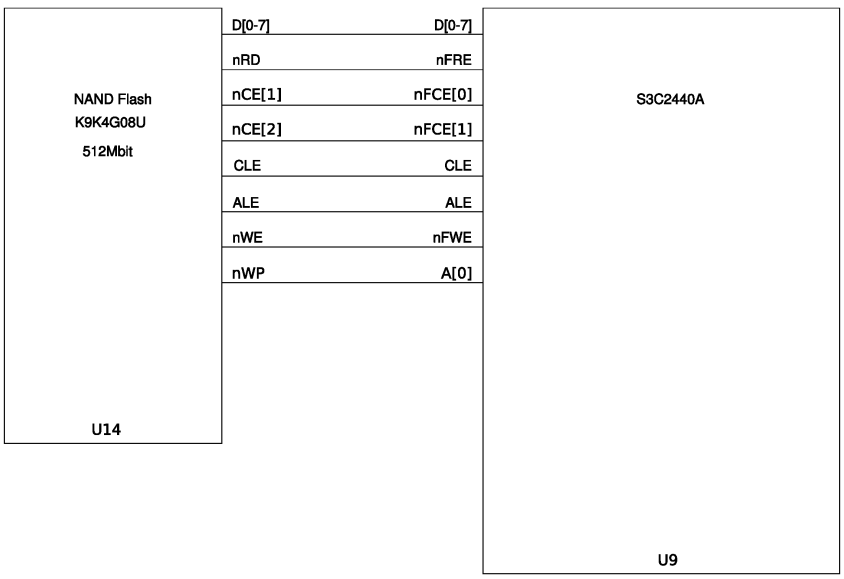
## 2.3.1. EEPROM

There is provision for a single EEPROM connected via the I2C bus. All specification boards have an 8KBit (1KB) device fitted as standard but up to 256KBit can be accommodated. The EEPROM is a seven bit addressed device at location 0xA0. The device is typically used to hold the non volatile settings in the ABLE bootloader.

## 2.3.2. NAND boot flash

The IM2440D20 has provision for a single NAND flash device. This device typically contains the bootloader and JFFS2 filesystem with a Linux image to boot, though it can contain anything the user requires. The flash is implemented as a single device, which is connected using the processors NAND bus.

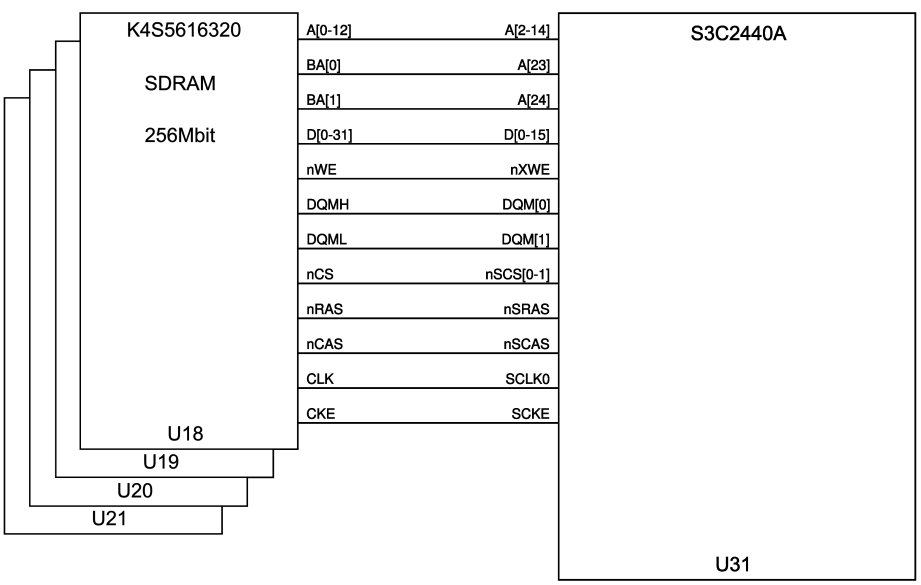
**Figure 2.2. NAND flash to S3C2440 attachment**




## 2.3.3. SDRAM

The IM2440D20 has provision for two or four SDRAM device with up to 512MBit capacity. Typically four 256MBit devices are fitted giving 128MBytes of storage. This memory is accessed using the integrated S3c2440 SDRAM controller.

**Figure 2.3. SDRAM to S3C24440 attachment**




# 2.4. Power Supply

The module is powered from four input pins (VIN) from which it generates all necessary internal voltages. This is achieved with a Texas Instruments TPS65011, this device is controlled from the CPU by using the I2C bus and is at address 0x24.

The nominal input operating voltages are 3.3 to 5V with a 5% tolerance. The absolute *maximum* ratings are 3.15V to 5.5V. The modules typical power consumption is 0.5W (150mA at 3.3V) when operating in a normal configuration at 400MHz.

The two VOUT pins provide output supply from the modules internal main power bus. The main bus is configured to default to 3.0V operation (this still lies within the typical 10% margin allowed for supplies on 3.3V devices) which saves power and gives a wider input operating voltage.

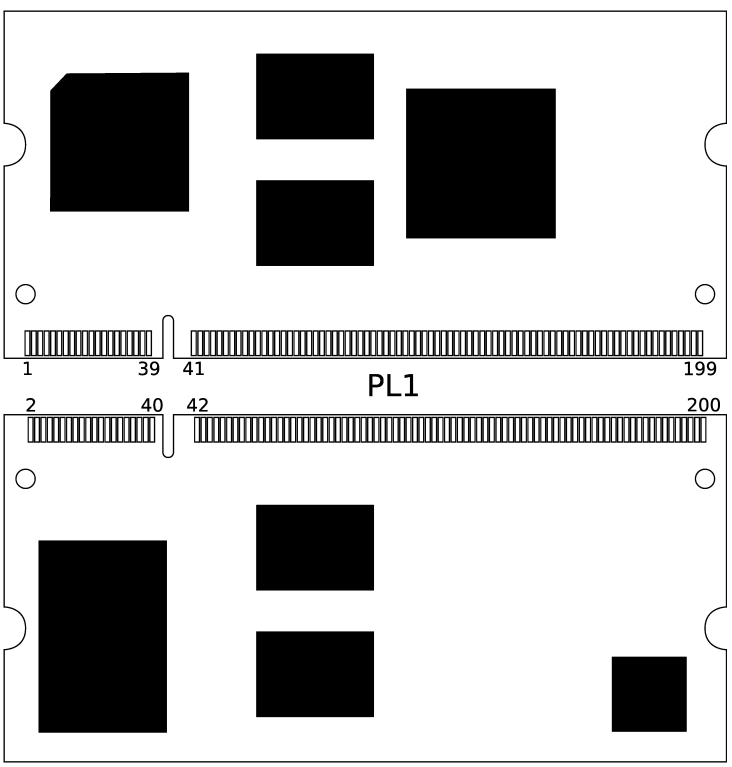
The VOUT output may be used to supply up to 300mA at 3.3V absolute maximum. The output must *not* be connected to the input supply or damage will occur. The output voltage will *never* exceed the input supply despite the output voltage selection.

The modules main supply voltage may be altered to one of 3.3, 3.0, 2.75 or 2.5V once the system is running, however correct module operations for settings below 3.0V are not guaranteed.

# 2.5. Edge connector

The IM2440D20 edge connections are JEDEC standard 200pin SODIMM compatible. The connector is keyed to 2.5V and any suitable SODIMM socket may be used.

### Figure 2.4. Edge connector placement




These connectors provide all the I/O signals to expand the use of the module. For full details the IM2440D20 Connector and link pinouts [http://www.simtec.co.uk/products/IM2440D20/files/pinlist.html] document should be consulted, this contains addition information and comments relevant to using this product.

### Table 2.1. 200 way edge connector PL1

| Description | Name | Pin | Pin | Name | Description |
|---|---|---|---|---|---|
| Output supply from module | VOUT | 1 | 2 | GND | Ground |
| Inverted read enable | nFRE | 3 | 4 | ALE | NAND address latch enable |
| NAND Inverted write enable | nFWE | 5 | 6 | CLE | NAND Command latch enable |
| Third NAND chip select | nFCE2 | 7 | 8 | R/nB | NAND ready/not busy |
| Data bus bit 0 | D0 | 9 | 10 | D16 | Data bus bit 16 |
| Data bus bit 1 | D1 | 11 | 12 | D17 | Data bus bit 17 |
| Data bus bit 2 | D2 | 13 | 14 | D18 | Data bus bit 18 |

| Description | Name | Pin | Pin | Name | Description |
|---|---|---|---|---|---|
| Data bus bit 3 | D3 | 15 | 16 | D19 | Data bus bit 19 |
| Data bus bit 4 | D4 | 17 | 18 | D20 | Data bus bit 20 |
| Data bus bit 5 | D5 | 19 | 20 | D21 | Data bus bit 21 |
| Data bus bit 6 | D6 | 21 | 22 | D22 | Data bus bit 22 |
| Data bus bit 7 | D7 | 23 | 24 | D23 | Data bus bit 23 |
| Data bus bit 8 | D8 | 25 | 26 | D24 | Data bus bit 24 |
| Data bus bit 9 | D9 | 27 | 28 | D25 | Data bus bit 25 |
| Data bus bit 10 | D10 | 29 | 30 | D26 | Data bus bit 26 |
| Data bus bit 11 | D11 | 31 | 32 | D27 | Data bus bit 27 |
| Data bus bit 12 | D12 | 33 | 34 | D28 | Data bus bit 28 |
| Data bus bit 13 | D13 | 35 | 36 | D29 | Data bus bit 29 |
| Data bus bit 14 | D14 | 37 | 38 | D20 | Data bus bit 20 |
| Data bus bit 15 | D15 | 39 | 40 | D31 | Data bus bit 31 |
| Key | | | | | |
| Ground | GND | 41 | 42 | GND | Ground |
| First inverted write byte enable | nWBE0 | 43 | 44 | BUF_EN | Buffer enable |
| Second inverted write byte enable | nWBE1 | 45 | 46 | BUF_DIR | Buffer direction |
| Third inverted write byte enable | nWBE2 | 47 | 48 | PWR_EN | Power good output |
| Forth inverted write byte enable | nWBE3 | 49 | 50 | nRESET | Inverted reset output |
| Inverted I/O wait signal | nWAIT | 51 | 52 | nCS1 | First inverted chip select |
| Inverted sixteen bit access select | nIOCS16 | 53 | 54 | nCS2 | Second inverted chip select |
| Inverted read select | nRD | 55 | 56 | nCS3 | Third inverted chip select |
| Inverted write select | nWR | 57 | 58 | nCS4 | Fourth inverted chip select |
| Ground | GND | 59 | 60 | nCS5 | Fifth inverted chip select |
| Address bus bit 0 | A0 | 61 | 62 | A15 | Address bus bit 15 |
| Address bus bit 1 | A1 | 63 | 64 | A16 | Address bus bit 16 |
| Address bus bit 2 | A2 | 65 | 66 | A17 | Address bus bit 17 |
| Address bus bit 3 | A3 | 67 | 68 | A18 | Address bus bit 18 |
| Address bus bit 4 | A4 | 69 | 70 | A19 | Address bus bit 19 |
| Address bus bit 5 | A5 | 71 | 72 | A20 | Address bus bit 20 |
| Address bus bit 6 | A6 | 73 | 74 | A21 | Address bus bit 21 |
| Address bus bit 7 | A7 | 75 | 76 | A22 | Address bus bit 22 |
| Address bus bit 8 | A8 | 77 | 78 | A23 | Address bus bit 23 |
| Address bus bit 9 | A9 | 79 | 80 | A24 | Address bus bit 24 |
| Address bus bit 10 | A10 | 81 | 82 | A25 | Address bus bit 25 |
| Address bus bit 11 | A11 | 83 | 84 | A26 | Address bus bit 26 |
| Address bus bit 12 | A12 | 85 | 86 | TCLK0 | Timer 0 output |
| Address bus bit 13 | A13 | 87 | 88 | TOUT0 | First PWM output |
| Address bus bit 14 | A14 | 89 | 90 | TOUT1 | Second PWM output |
| Ground | GND | 91 | 92 | TOUT2 | Third PWM output |
| First DMA acknowledge | DACK0 | 93 | 94 | TOUT3 | Fourth PWM output |
| First DMA request | DREQ0 | 95 | 96 | DACK1 | Second DMA acknowledge |
| I2S left right select | I2S_LRCLK | 97 | 98 | DREQ1 | Second DMA request |

| Description | Name | Pin | Pin | Name | Description |
|---|---|---|---|---|---|
| I2S clock | I2S_CLK | 99 | 100 | EINT0 | External interrupt 0 |
| I2S command/data select | I2S_CDCLK | 101 | 102 | EINT1 | External interrupt 1 |
| I2S serial data in | I2S_SDI | 103 | 104 | EINT2 | External interrupt 2 |
| I2S serial data out | I2S_SD0 | 105 | 106 | EINT3 | External interrupt 3 |
| SD card clock | SDCLK | 107 | 108 | EINT4 | External interrupt 4 |
| SD card command | SDCMD | 109 | 110 | EINT5 | External interrupt 5 |
| SD card first data line | SDDATA0 | 111 | 112 | EINT6 | External interrupt 6 |
| SD card second data line | SDDATA1 | 113 | 114 | EINT7 | External interrupt 7 |
| SD card third data line | SDDATA2 | 115 | 116 | EINT8 | External interrupt 8 |
| SD card fourth data line | SDDATA3 | 117 | 118 | EINT9 | External interrupt 9 |
| SPI bus master in slave out | SPIMISO | 119 | 120 | EINT10 | External interrupt 10 |
| SPI bus master out slave in | SPIMOSI | 121 | 122 | EINT11 | External interrupt 11 |
| SPI bus clock | SPICLK | 123 | 124 | EINT12 | External interrupt 12 |
| I2C bus clock | SCL | 125 | 126 | EINT13 | External interrupt 13 |
| I2C bus data | SDA | 127 | 128 | EINT14 | External interrupt 14 |
| Ground | GND | 129 | 130 | EINT15 | External interrupt 15 |
| Line end | LEND | 131 | 132 | EINT16 | External interrupt 16 |
| Video clock | VCLK | 133 | 134 | RTS1 | Second serial port request to send |
| Horizontal sync | HS | 135 | 136 | CTS1 | Second serial port clear to send |
| Vertical sync | VS | 137 | 138 | EINT19 | External interrupt 19 |
| Video data enable | DE/VM | 139 | 140 | CTS0 | First serial port clear to send |
| Video data 0 | VD0 | 141 | 142 | RTS0 | First serial port request to send |
| Video data 1 | VD1 | 143 | 144 | TX0 | First serial transmit |
| Video data 2 | VD2 | 145 | 146 | RX0 | First serial receive |
| Video data 3 | VD3 | 147 | 148 | TX1 | Second serial transmit |
| Video data 4 | VD4 | 149 | 150 | RX1 | Second serial receive |
| Video data 5 | VD5 | 151 | 152 | TX2 | Third serial transmit |
| Video data 6 | VD6 | 153 | 154 | RX2 | Third serial receive |
| Video data 7 | VD7 | 155 | 156 | VOUT | Output supply from module |
| Video data 8 | VD8 | 157 | 158 | GND | Ground |
| Video data 9 | VD9 | 159 | 160 | ADC0 | Analog input 0 |
| Video data 10 | VD10 | 161 | 162 | ADC1 | Analog input 1 |
| Video data 11 | VD11 | 163 | 164 | ADC2 | Analog input 2 |
| Video data 12 | VD12 | 165 | 166 | ADC3 | Analog input 3 |
| Video data 13 | VD13 | 167 | 168 | ADC4 | Analog input 4 |
| Video data 14 | VD14 | 169 | 170 | ADC5 | Analog input 5 |
| Video data 15 | VD15 | 171 | 172 | ADC6 | Analog input 6 |
| Video data 16 | VD16 | 173 | 174 | ADC7 | Analog input 7 |
| Video data 17 | VD17 | 175 | 176 | GND | Ground |
| Video data 18 | VD18 | 177 | 178 | DN0 | USB first channel data negative |
| Video data 19 | VD19 | 179 | 180 | DP0 | USB first channel data positive |
| Video data 20 | VD20 | 181 | 182 | DN1 | USB second channel data negative |
| Video data 21 | VD21 | 183 | 184 | DP1 | USB second channel data positive |
| Video data 22 | VD22 | 185 | 186 | nTRST | Inverted tap reset |

| Description | Name | Pin | Pin | Name | Description |
|---|---|---|---|---|---|
| Video data 23 | VD23 | 187 | 188 | TDI | Tap data input |
| Ground | GND | 189 | 190 | TMS | Tap mode select |
| Real time clock supply | VCC_RTC | 191 | 192 | TCK | Tap clock |
| Power supply | VIN | 193 | 194 | TDO | Tap data out |
| Power supply | VIN | 195 | 196 | nRESET | Inverted reset input |
| Power supply | VIN | 197 | 198 | GND | Power ground |
| Power supply | VIN | 199 | 200 | GND | Power ground |

# Chapter 3. Bootloader

## 3.1. Overview

The Simtec Electronics Advanced Boot Load Environment (ABLE) is a portable modular boot loader for use in applications where an OS must be retrieved and started. ABLE provides extended functionality providing modules for a command line, video consoles, serial consoles, network booting and numerous other facilities.

ABLE is a powerful tool and provides a very flexible environment useful for both development and deployment of systems. ABLE is a boot loader, not an Operating System this distinction can sometimes lead to misunderstandings about the capabilities provided by ABLE. A boot loader in this context is a self contained program which retrieves and starts execution of an Operating System. It does not execute user programs itself (all the CLI commands are built in) and does not provide services to an Operating System once started (PC BIOS perform this role).

The modular nature of ABLE allows the use of the same building blocks for every supported platform. The integration and omission of various modules allow for specific driver sets depending on the peripherals of a platform. The flexibility of this approach allows for a common familiar environment across all supported platforms while still supporting a complete feature set.

This chapter only provides a brief introduction to ABLE. Full documentation can be found in the ABLE user guide [http://www.simtec.co.uk/products/SWABLE/files/able-set/book_userguide.html].

## 3.2. Getting Started

When a platform is initially powered or a hard reset performed, the ABLE environment will be started and each component module will be loaded in turn. The last module loaded is the ABLE shell, which will present the user with a command line interface.

ABLE has the ability to use a combination of input and output sources to interact with a user. The default is to use all the input and output devices available. For example, on the EB2410ITX both the console serial port and the video display will be used to output and the serial port for input (future versions may support USB keyboards for input).

### Example 3.1. Video display after starting ABLE on EB2410ITX

```
selected all-wr for console write stream
selected all-rd for console read stream
DRAM: 128 Mb (134217728 bytes)
BAST: PMU version 1.02, ID 00:01:3d:00:01:6a
ABLE: 2.08 (s3c2410x) (vince@gerald) Fri Apr  8 16:35:26 BST 2005
hdc: TOSHIBA MK1003MAV: ATA PIO mode 4
hdc:Diagnosing disc drive: ok
(hdc) 1GB
(hd0) on ((hdc1):ext2)
(hd1) on (hdc2)
DM9000: dm0: r1, 00:01:3d:00:01:6a int phy, link ok, 100Mbit full duplex
NE2000: ne0: ISA/Generic, 00:01:3d:00:01:6b (EEPROM Invalid / Missing)
TMP101: I2C error (-2)
sys.autoshadow unset, automatically shadowing
>
```

### Example 3.2. Serial display after starting ABLE on EB2410ITX

```
SuperIO controller fitted
Initialising
Detecting SDRAM size
SDRAM: BANK6 size 04000000
```

```
SDRAM: BANK7 size 04000000
ABLE: 2.08 (s3c2410x) (vince@gerald) Fri Apr  8 16:35:26 BST 2005
Processor: Samsung S3C2410A (arm920)
System: Machine bast/s3c2410x, Linux id 0x014b
.S3C2410X RTC: 01:46:54, 00/01/2003
NAND: configured boot slot is 0 (card slot)
NAND: found Samsung K9F1208u0a [131072,32,512]
(flash0) on (nand0p1)
(flash1) on ((nand0p2):jffs2)
EEPROM: 24cXX, 1024 bytes, single byte addressed
(nvram0) on (24cxx)
sys.speed is unset, Setting CPU Speed to 266MHz
no configuration, defaulting to VGA
X/Y values invalid, configuring automatically
Chrontel CH7006 detected
screen mode is 640x480, ?Hz, ?Hz HSync
video: video size 300K
configuring ch7006: vga
selected all-wr for console write stream
selected all-rd for console read stream
DRAM: 128 Mb (134217728 bytes)
BAST: PMU version 1.02, ID 00:01:3d:00:01:6a
ABLE: 2.08 (s3c2410x) (vince@gerald) Fri Apr  8 16:35:26 BST 2005
hdc: TOSHIBA MK1003MAV: ATA PIO mode 4
hdc:Diagnosing disc drive: ok
(hdc) 1GB
(hd0) on ((hdc1):ext2)
(hd1) on (hdc2)
DM9000: dm0: r1, 00:01:3d:00:01:6a int phy, link ok, 100Mbit full duplex
NE2000: ne0: ISA/Generic, 00:01:3d:00:01:6b (EEPROM Invalid / Missing)
TMP101: I2C error (-2)
sys.autoshadow unset, automatically shadowing
>
```

The input devices are controlled by using the cons-read parameter and similarly the cons-write parameter controls which output devices are used.

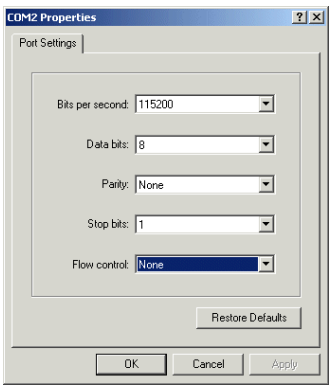
Typically the console serial port is used to interact with the ABLE CLI.

Unless the boot parameters are altered from their default settings the autoboot process will commence. To manually start an Operating System the command line must be used.

## 3.2.1. Using hyperterm as a serial console

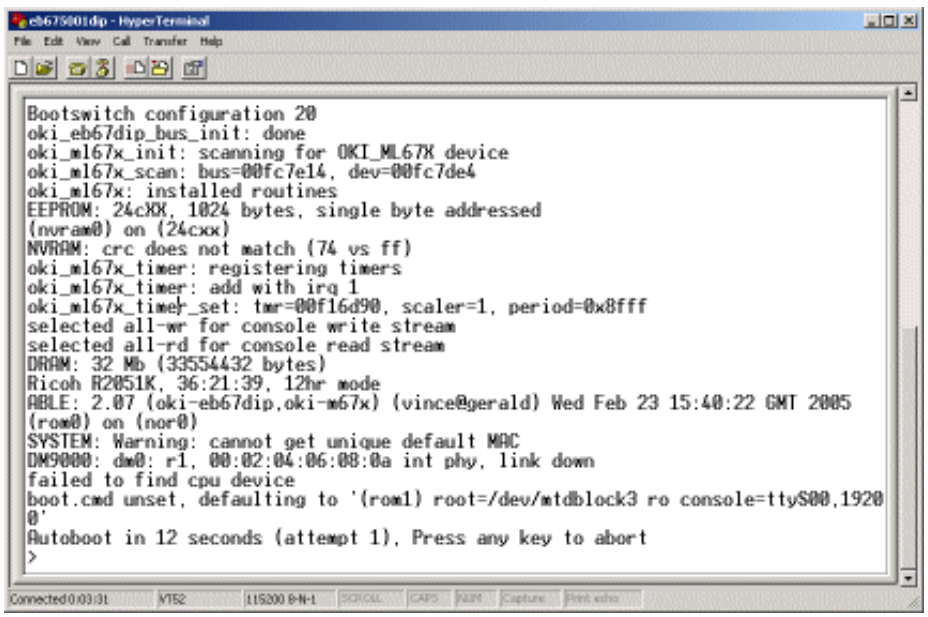
To access the serial console from windows the hyperterm program can be used. Identify which serial port the platform is connected to and ensure a note is made of the correct COM port, e.g. COM1 or COM2.

### Figure 3.1. Hyperterm settings window



Start HyperTerminal and create a new connection. When prompted for which modem to use, instead choose the appropriate COM port, as noted earlier. Then the appropriate settings for your platform (please refer to platform specific documentation) typically these settings are 115200 bits per second, 8 data bits, no parity, 1 stop bit and no flow control as shown in Figure 3.1, “Hyperterm settings window”.

### Figure 3.2. Hyperterm displaying ABLE output




Once the connection is established the output from ABLE should be seen in the hyperterm window as in Figure 3.2, “Hyperterm displaying ABLE output”

## 3.2.2. Using minicom as a serial console

To access the serial console from LINUX® the minicom program can be used. Identify which serial port the EB675001DIP is connected to and ensure a note is made of the correct device node, e.g. something like /dev/ttyS0 or /dev/ttyUSB0.

### Figure 3.3. Minicom settings window

```
Welcome to minicom 2.1
                        ┌──────────[Comm Parameters]──────────┐
OPTIONS: History Buf    │                                     │I18n
Compiled on Mar 29 2    │ Current: 115200 8N1                 │
                        │                                     │
Press CTRL-A Z for h    │   Speed          Parity          Data │
                        │                                     │
                        │ A: 300           L: None         S: 5 │
                        │ B: 1200          M: Even         T: 6 │
                        │ C: 2400          N: Odd          U: 7 │
                        │ D: 4800          O: Mark         V: 8 │
                        │ E: 9600          P: Space             │
                        │ F: 19200                       Stopbits │
                        │ G: 38400                         W: 1 │
                        │ H: 57600                         X: 2 │
                        │ I: 115200        Q: 8-N-1             │
                        │ J: 230400        R: 7-E-1             │
                        │                                     │
                        │                                     │
                        │ Choice, or <Enter> to exit? ▯       │
                        └─────────────────────────────────────┘

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.1     | VT102 |      Offline
```

Start minicom and ensure the correct settings are selected (Default is Ctrl-A p). These settings are 115200 baud, 8 data bits, no parity and 1 stop bit as shown in Figure 3.3, “Minicom settings window”. Obviously Minicom should be using the correct serial port as noted earlier.

# Appendix A. Board Layout

**Figure A.1. EB675001DIP board layout top side**

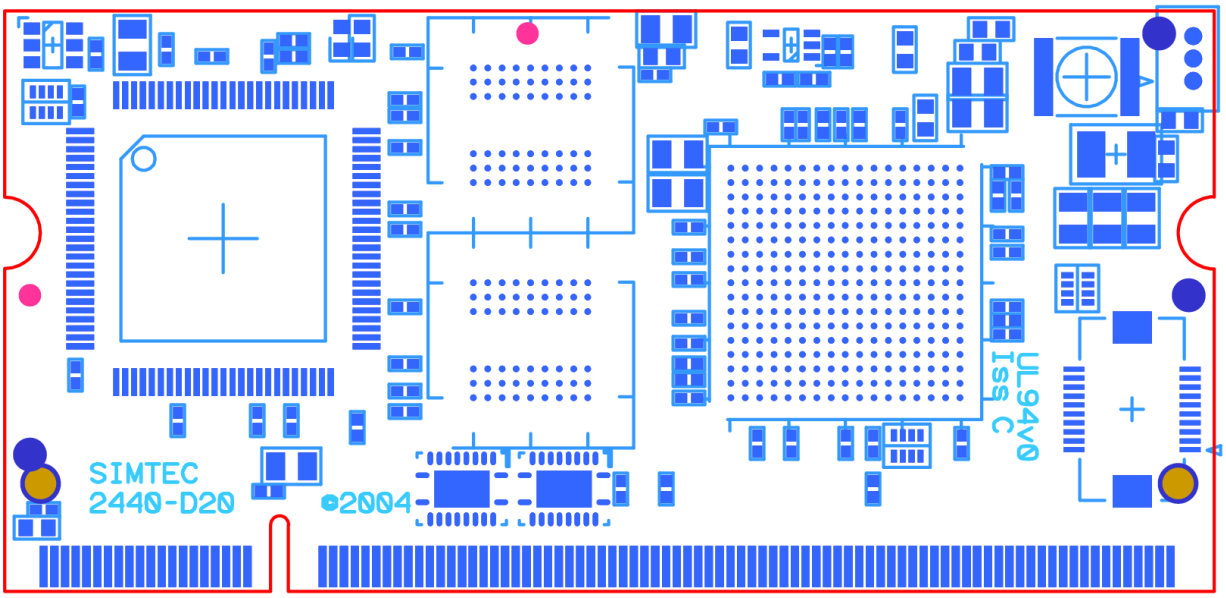
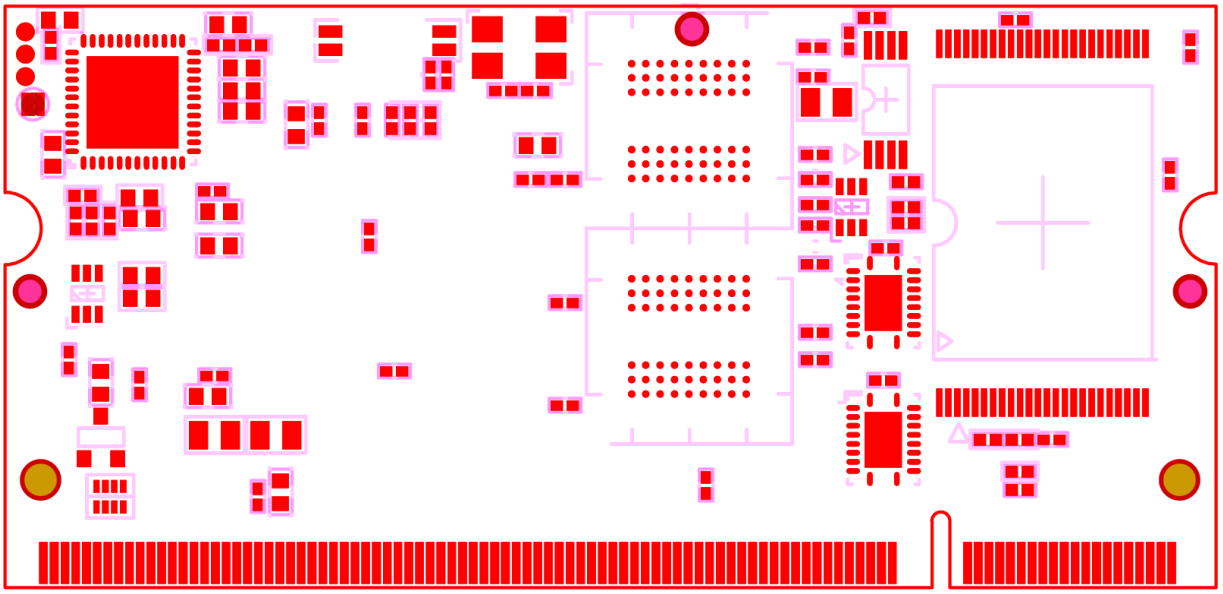
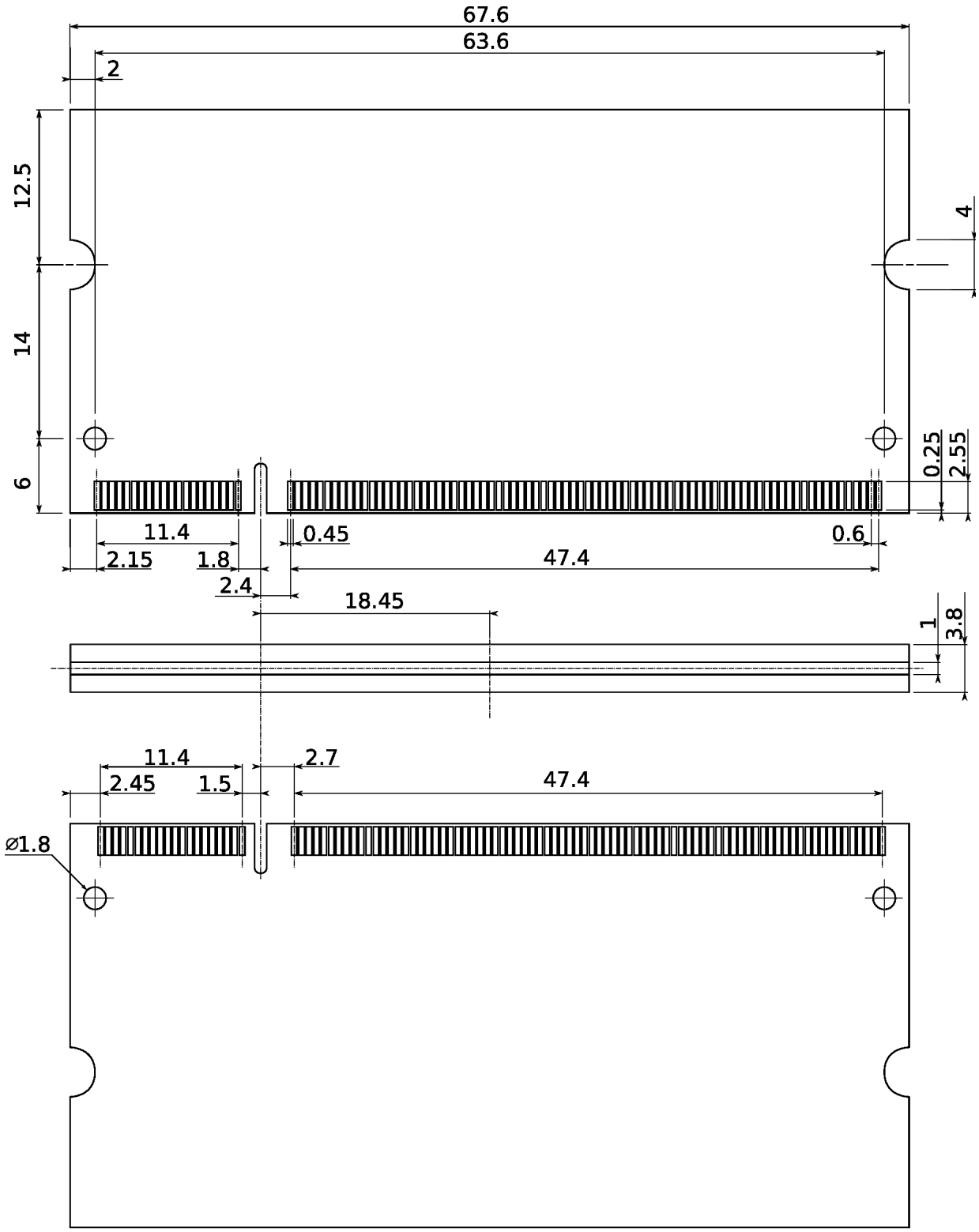



**Figure A.2. EB675001DIP board layout bottom side**



# Appendix B. Mechanical drawing

**Figure B.1. IM2440D20 Mechanical Drawing**




Scale 2:1
All measurements in mm
Version 1.1 - 8/9/2005

# Colophon

This Document was prepared in Docbook XML [http://www.docbook.org/] using the GNU emacs text editor. The source was combined with DocBook XSL Stylesheets [http://docbook.sourceforge.net/projects/xsl/] using an XSLT processor to produce output in various formats.

For web output the Saxon XSLT processor [http://saxon.sourceforge.net/] was used to convert the docbook XML directly to HTML.

For print output the Saxon XSLT processor [http://saxon.sourceforge.net/] was used to convert the docbook XML to Formatting Objects (FO) XML.

For general print documents the FO XML is converted to PDF and Postscript with the Apache project FOP [http://xmlgraphics.apache.org/fop/] utility.

For six by nine inch book output the Render X XEP digital typography tool was used to convert the FO XML to print ready PDF output. The URW Nimbus Sans font families were used to perform this typesetting.

The cover designs were developed in the GNU Image Manipulation Program [http://www.gimp.org/] (GIMP).