Development board PXA 250

version 1

June 22, 2003

Contents

1 Introduction **3**
- 1.1 Hardware Specification . . . 4
- 1.2 Software Specification . . . 4

2 Hardware **5**
- 2.1 Board Layout . . . 5
- 2.2 Connectors and jumpers . . . 5
- 2.3 Description of on-board devices . . . 8
  - 2.3.1 SDRAM memory 64 MB . . . 8
  - 2.3.2 Flash memory 16MB . . . 9
  - 2.3.3 Address map . . . 9
  - 2.3.4 Ethernet controller . . . 9
  - 2.3.5 AC'97 stereo audio codec AD1885 (optional) . . . 9
  - 2.3.6 GPIO Port Pin Assignments . . . 9
  - 2.3.7 PXA250 serial ports . . . 10
  - 2.3.8 PXA250 Initialization . . . 11
- 2.4 Switch and LEDs (optional) . . . 12
- 2.5 Flash erase jumper . . . 12
- 2.6 Power . . . 12
- 2.7 Cables . . . 12

3 Software & Development tools **13**
- 3.1 Programming flash via JTAG . . . 13
- 3.2 tftp server and client . . . 14
- 3.3 Cross Compiler installation . . . 14
- 3.4 Boot loader . . . 15
  - 3.4.1 Developmnet boot loader compilation . . . 15
  - 3.4.2 Production boot loader compilation . . . 15
- 3.5 Kernel compilation . . . 15
- 3.6 Creating file system . . . 16
  - 3.6.1 Flash memory map . . . 16

3.7 Mtd device 17
3.7.1 Creating root file system 18
3.7.2 JFFS2 file system 18
3.7.3 CRAMFS file system 18

4 Board using **19**
4.1 Connecting to the board 19
4.2 Flashing boot loader via JTAG 19
4.3 Flashing kernel and rootfs via Ethernet network 20

5 Examples **23**
5.1 Compiling application 23
5.2 “Hello World” 23
5.3 Test of GPIO Led 23
5.4 Simple serial port application 23
5.5 Simple client/server application 23

6 Resources **24**

Chapter 1

Introduction

This PXA250 board is intended for running embedded network applications. The PXA250 board is optimised for the development of highly efficient Internet devices, and for network infrastructure applications.

The PXA250 board is based on the new Intel XScale architecture. Intel XScale processor family increases efficiency and decreases processor power consumption. The Intel XScale micro architecture is based on the Intel Strong ARM technology.

Intel Strong ARM and Intel XScale are compatible with the ARM architecture, which in turn guarantees the compatibility of software solutions.

The PXA250 board includes Intel XScale processor, SDRAM and Intel StrataFlash memory, 2 Ethernet controllers,3 serial interfaces and 1 stereo audio output.

1.1 Hardware Specification

CPU Intel	XScale PXA 250
Clock	200 MHz
Memory	Flash 16MB SDRAM 64MB
On-Board Peripherals	1x asynchronous serial port, RS232 level (full handshaking) 1x asynchronous serial port, 3.3-5V level (RTS/CTS hadshaking only) 1x synchrnous serial port, 3.3-5V level, support SSPC, SSP and SPI protocols 2x Ethernet 10BASE-T port (chip CS8900a) JTAG interface
Audio (optional)	AD1885 AC'97 audio codec
Power Supply:	7-12V DC
Dimension	100 x 107 mm
Temperatures:	Commercial: 0C to +70C Industrial: $-40°C$ to $+85°C$ (without audio chip)

1.2 Software Specification

The PXA250 board is delivered with preinstalled boot loader and Linux kernel. This firmware support direct application loading and running via Ethernet.

This firmware can be changed via JTAG interface, development boot loader or through mtd device (on the running kernel).

The sources codes for development are on the CD Development tools for PXA250 Board.

Boot loader	proprietary for production Armboot for developing
Kernel	2.4.19 with patches for ARM, Xscale and proprietary hardware
Available Flash Memory	aprox. 9MB

Chapter 2

Hardware

2.1 Board Layout

The upper layer of board provide console port, reset button, three GPIO LED, two RJ-45 10BASE-T ports, four Ethernet LEDs , power connector and power LED. Figure 2.1 shows the upper layer.

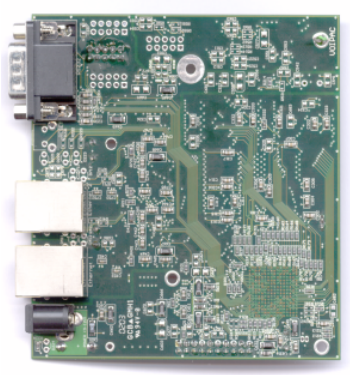
The bottom layer of board provide serial port (3-5V), flash lock jumper and JTAG interface. Figure 2.2 shows the bottom layer.

2.2 Connectors and jumpers

Power
7,5V DC (connector: 5.5x2.1mm, centre positive)

Console RS232
asynchronus serial port (V.28 voltage levels, FFUART)

Pin	Description	Pin	Description
1	SP0_DCD	6	SP0_DSR
2	SP0_RXD	7	SP0_RTS
3	SP0_TXD	8	SP0_CTS
4	SP0_DTR	9	SP0_RI
5	GND		



1	console port
2	reset button
3	GPIO LEDs
4	10BASE-T (eth1)
5	10BASE-T (eth0)
6	Ethernet LEDs
7	power connectors
8	power LED

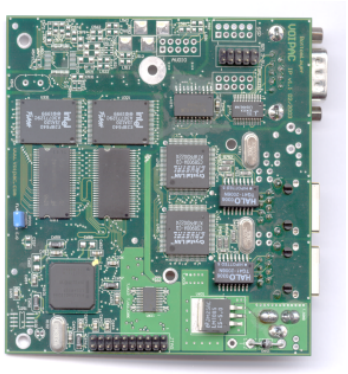
Figure 2.1: Upper layer

SP0 RS232
asynchronus serial port (V.28 voltage level, FFUART)

Pin	Description	Pin	Description
1	SP0_RXD	2	SP0_CTS
3	SP0_DCD	4	SP0_DSR
5	SP0_RI	6	SP0_TXD
7	SP0_RTS	8	SP0_DTR
9	GND	10	GND

SP1 3-5V
asynchronus serial port (BTUART), 5V tolerant I/O pins accept 5V,3.3V

Pin	Description	Pin	Description
1	VCC5	2	VCC3.3
3	SP1_BT_CTS	4	GND
5	SP1_BT_RTS	6	GND
7	SP1_BT_RXD	8	GND
9	SP1_BT_TXD	10	GND



1	serial port
2	flash lock
3	JTAG interface

Figure 2.2: Bottom layer

SSP
synchronous serial port, 5V tolerant I/O pins accept 5V,3.3V

Pin	Description	Pin	Description
1	VCC5	2	VCC3.3
3	SSP_FRM	4	GND
5	SSP_CLK	6	GND
7	SSP_TXD	8	GND
9	SSP_RXD	10	GND

JTAG interface
JTAG allows burning into the flash memory or debugging the processor.

Pin	Description	Pin	Description
1	VREF (VCC3.3)	2	VCC3.3
3	-TRST	4	GND
5	TDI	6	GND
7	TMS	8	GND
9	TCK	10	GND
11	RTCK	12	GND
13	TDO	14	GND
15	-RESET	16	GND
17	NC	18	GND
19	NC	20	GND

Audio (optional)
Connector AUDIO contains outputs of the Audio Codec.

Pin	Description	Pin	Description
1	LINE_IN_L	2	LINE_OUT_L
3	LINE_IN_R	4	LINE_OUT_R
5	AGND	6	MONO_OUT
7	MIC1_IN	8	HP_OUT_L
9	MIC1_REF	10	HP_OUT_R

Ethernet 1OBASE-T RJ45 connector:

Pin	Description	Pin	Description
1-2	TXD	3-6	RXD

Jumper

JP331	FLASH ERASE ENABLE (with jumper) FLASH ERASE DISABLE (without jumper)

LED

D221	connected to PXA250 GPIO3
D222	connected to PXA250 GPIO4
D223	connected to PXA250 GPIO5
D224	power LED
D411	Ethernet 0 LINK
D412	Ethernet 0 ACTIVE
D711	Ethernet 1 LINK
D712	Ethernet 1 ACTIVE

Switch

RESET	reset board

2.3 Description of on-board devices

2.3.1 SDRAM memory 64 MB

The board uses two 256Mbit SDRAM devices organised as one 32-Bit Bank. They support 100MHz operation.

2.3.2 Flash memory 16MB

The board uses two Intel 28F640J3 flash chips organised as one 32-Bit Bank.

2.3.3 Address map

The board used standard address map with the following modification:

Pin	Description	
CS0	(0-0x03FF.FFFF)	flash memory
CS1	(0x0400.0000-0x07FF.FFFF)	unused
CS2	(0x0800.0000-0x0BFF.FFFF)	Ethernet chip 1
CS3	(0x0C00.0000-0x0FFF.FFFF)	Ethernet chip 2
CS4	(0x1000.0000-0x13FF.FFFF)	unused
CS5	(0x1400.0000-0x17FF.FFFF)	unused

2.3.4 Ethernet controller

The Cirrius Logic CS8900a is used to provide 10Mbit Ethernet interface via twisted-pair.

2.3.5 AC'97 stereo audio codec AD1885 (optional)

The PXA250 implements a standart ac'97 Codec interface. An AD1885 AC'97 codec (or equivalent) allows this interface to transmit and receive analog audio data. The AD1885 is located at AC'97 input 0.

The AD1885 also integrates a Headphone Output Apmplifier and a Microphone Input Amplifier for ease of use.

2.3.6 GPIO Port Pin Assignments

GPIO (General purpose I/O pins) pin can be individually programmed as an output or an input. The following table describes the GPIO pins used by this board.

Pin	Description
0-2	unused
3 -5	USER LEDs D221,D222,D223
6 -12	unused
13	Ethernet reset
14	Ethernet 1 interrupt
15 -18	unused
19	Ethernet 2 interrupt
20	free pin for hardware configuration by connecting the R261(High) or R262(Low) Resistor through 4.7k
21 - 22	unused
23 - 27	SSP
28 - 31	AC'97
32	free pin for hardware configuration by connecting the R261(High) or R262(Low) Resistor through 4.7k
33	unused
34 - 41	UART 0
42 - 45	UART 1
46 - 47	IRDA
48 - 77	unused
78	Ethernet 1 chipselect
79	Ethernet 2 chipselect
80	unused

2.3.7 PXA250 serial ports

The PXA250 has three asynchronous serial ports (FFUART,BTUART and STUART) and one synchronous serial port (SSPC). The FFUART supports full handshaking, while the BTUART supports RTS/CTS only.

Only two UARTs of the processor are routed to the connectors: FFUART and STUART. The FFUART is shared by the SP0 (10-pin heder) and RS232 Connectors (DB-9 male) at V.28 power levels. The SP1 connector(10-pin header) connects to the BTUART and adopts for power levels of 3.3-5V.

The synchronous serial port (SSP) supports three protocols: National Semiconductor s Microwire, Texas Instruments Synchronous Serial Protocol, and Motorola's Serial Peripheral Interface. This port is routed to a 10-Pin Header SSP.

2.3.8 PXA250 Initialization

It is necessary to initialize certain internal registers before correct operation can begin. WARNING: Change to some register values may result in damaging behaviour of the hardware.

GPIO port control registers

These registers control the direction of the various GPIO port pins. The following table describes the control bits and their values that are required, to place the GPIO pins in the correct mode based upon the uPC usage for each GPIO.

```
GPIOSR0 (0x40E0.0018) = 0xFFFF.FFFF
GPIOSR1 (0x40E0.001C) = 0xFFFF.FFFF
GPIOSR2 (0x40E0.0020) = 0xFFFF.FFFF

GPCR0 (0x40E0.0024) = 0x0802.2080
GPCR1 (0x40E0.0028) = 0x0000.0000
GPCR2 (0x40E0.002C) = 0x0000.0000

GPDR0 (0x40E0.000C) = 0xCB82.A8D8
GPDR1 (0x40E0.0010) = 0xFCFF.AB83
GPDR2 (0x40E0.0014) = 0x0001.FFFF

GPIOAFR0L (0x40E0.0054) = 0x8000.0000
GPIOAFR0U (0x40E0.0058) = 0xA51A.8010
GPIOAFR1L (0x40E0.005C) = 0x5999A.9558
GPIOAFR1U (0x40E0.0060) = 0xAAA5.AAAA
GPIOAFR2L (0x40E0.0064) = 0x0000.5000
GPIOAFR2U (0x40E0.0068) = 0x0000.0002
```

Static memory controller configuration registers

The Static Chip Selects are setup to conform to the requirements of the various I/O devices. All timing is done using MCLK, which is 99.5MHz (cca 10nsec per clock). If lower frequency operation is desired, these values will need to be adjustned appropriately. The folloqing table lists recommended values for each of the static memory control registers.

```
MSC0 (0x4800.0008) = 0x2EF1.5AF0
MSC1 (0x4800.000C) = 0x3FF4.3FF4
MSC2 (0x4800.0010) = 0x0000.0000
```

SDRAM controller configuration registers

Attached to the PXA250 SDRAM Controller are two banks of two each, 8Mbit x 16 (or optionally 16Mbit x 16) SDRAM devices. In order for these devices to operate properly several SDRAM control registers must be set correctly. All timing is done using MCLK, which is 99.5MHz, or 10nsec per clock. If lower frequency operation is desired, these values will need to be adjusted appropriately. The following table lists recommended values.

```
MDCNFG (0x4800.0000) = 0x09A9.09A9
MDREFR (0x4800.0004) = 0x038F.F030
MDMRS (0x4800.0040) = 0x0022.0022
```

2.4 Switch and LEDs (optional)

The Board can accommodate the Reset Switch. The Switch may be connected if desired.

The power LED indicates that power is applied to the board.

Three user LEDs are driven from the GPTIO processor pins. A low level turns the LEDs on. It is necessary to connect the LED's to the following 330R Resistors (D221 and R221, D222 and R222, D223 and R223).

Two ethernet LEDs are used to provide status indication for each Ethernet. It is necessary to connect corresponding Resistors (D411 and R411, D412 and R412, D711 and R711, D712 and R712).

2.5 Flash erase jumper

Jumper JP331 must be inserted in case you want to erase anything from the flash memory.

2.6 Power

A standard 2.1mm DC jack is used to provide power the board. The center of jack is positiv. It is recommended to power the board by stabilised source 7.5V-12V.

2.7 Cables

Chapter 3

Software & Development tools

3.1 Programming flash via JTAG

The programming through JTAG interface is suitable for primary software burning (e.g. production or developer boot loader) into flash memory.

There are several software for programming FLASH through JTAG. We are tested Jflashmm from Intel (for Widows) and jtag from from OpenWinCE project (linux). Jflashmm is fast than jtag, but jtag is more reliable. We will use jtag in this guide. You need for compiling jtag install two source code from OpenWinCE: include and jtag-0.4. Use this command for instalation:

```
tar -xzv include.tar.gz
cd includeXX
configure
make
make install
tar -xzv jtagXX
cd jtag
configure
make
make install
```

Then connect JTAG to the board and to the parallel port on computer. Run software for programming through JTAG by command `jtag`.

Follow these steps for JTAG software initialisation (cable is connected to parallel port at address 0x378):

1.	`cable parallel 0x378 DLC5`	you have to determine type of JTAG cable
2.	`detect`	detect type of CPU
3.	`detectflash`	detect type of flash memory

You can use prepared script `cable` by command `jtag cable`. For reading/writing file from/to flash memory use this commands:

`readmem 0x0 0x20000 armboot.bin`
: to read data with length 0x2000 bytes from flash memory to file `boot_loader.bin`

`flashmem 0x0 armboot.bin`
: to write data from file `armboot.bin` to address 0x0

For help use command `help`. For exiting from jatg use command `quit`.

3.2 tftp server and client

On Debian Linux you can install tftp server (package `tftpd`) and client (package `tftp`). We have client for testing only. Data for tftp server are stored in `/boot` directory.

Now we can test our tftp server. We create file `test.bin` in `/boot`. Then we can attempt download it via tftp protcol.

```
# dd if=/dev/zero of=/boot/test.bin bs=1024 count=1
# tftp localhost
tftp> get test.bin
Received 1024 bytes in 0.4 seconds
tftp> quit
```

3.3 Cross Compiler installation

You can build your own tools form sources or you can use prebuild application from CD. The prebuild application come from www.arm.linux.co.org.

For installing prebuild tools use this commands:

```
mkdir /usr/local/arm
cp cross-X.X.tar.bz2 /usr/local/arm
cd /usr/local/arm
tar -xjf cross-X.X.tar.bz2
```

Then you can set PATH to binary executable in the shell enviroment.

```
PATH=$PATH:/usr/local/arm/X.X/bin
```

Now you can test compiler.

```
arm-linux-gcc hello_world.c -o hello_world
```

3.4 Boot loader

The boot loader intialize hardware before starting kernel. Development boot loader provide additional function as memory function (dump, copy and modification), file download (via serial line or network). Production boot loader initialize hardware and starting kernel.

You don't need copmile this boot loader, you can use precompiled binary image instead. This images are in the directory `boot` (production boot loader) and `armboot` (development boot loader).

The boot loaders placed on the address 0x0.We can flash bootloader by JTAG interface.

```
jtag cabel
>flashmem 0x0 armboot.bin
```

3.4.1 Developmnet boot loader compilation

To be done later.

3.4.2 Production boot loader compilation

To be done later.

3.5 Kernel compilation

The kernel source for Xscale is created by patching standard kernel with three patches. First patch is for ARM, second for Xscale architekture and third for board hardware. Firstly we have to prepare kernel sources and patches.

```
tar -xjf linux-2.4.19.tar.bz2
cp patch-2.4.19-rmk4.gz linux-2.4.19
cp diff-2.4.19-rmk4-pxa2.gz linux-2.4.19
cp diff-2.4.19-rmk4-pxa2-vpac.gz linux-2.4.19
mv linux-2.4.19 linux-2.4.19-rmk4-pxa2-vpac
cd linux-2.4.19-rmk4-pxa2-vpac/
```

Than we patch kernel.

```
gunzip patch-2.4.19-rmk4.gz
patch -p1 <patch-2.4.19-rmk4
gunzip diff-2.4.19-rmk4-pxa2.gz
```

```
patch -p1 <diff-2.4.19-rmk4-pxa2
gunzip diff-2.4.19-rmk4-pxa2-vpac.gz
patch -p1 <diff-2.4.19-rmk4-pxa2-vpac
```

We will use `gcc3.0` for kernel compilation. We can check `PATH` and gcc version setting:

```
# echo $PATH
... :/usr/local/arm/3.0/bin: ....
# arm-linux-gcc -v
...
gcc version 3.0
#
```

Now we can make kernel. We don't need make changes in menuconfig, we save configuration only.

```
make menuconfig
make dep
make zImage
```

And at the end we copy the image of kernel to `/boot/pxa` directory. The `/boot` directory is accessible via tftp protocol (on Debian).

```
mkdir /boot/pxa
cp arch/arm/boot/zImage /boot/pxa
```

You can find information about new kernel and patches on www.arm.linux.co.org.

3.6 Creating file system

3.6.1 Flash memory map

We use this configuratuion.

```
0x00000000-0x00040000 : "Bootloader"
0x00040000-0x00100000 : "Kernel"
0x00100000-0x00500000 : "Filesystem"
0x00500000-0x01000000 : "JFFS2"
```

This configurration is defined in `drivers/mtd/maps/lubbock.c`.

```
static struct mtd_partition lubbock_partitions[] = {
{
name: "Bootloader",
size: 0x00040000,
offset: 0,
mask_flags: MTD_WRITEABLE  /* force read-only */
},{
name: "Kernel",
size: 0x000c0000,
offset: 0x00040000,
},{
name: "Filesystem",
size: 0x00400000,
offset: 0x00100000
},{
name: "JFFS2",
size: MTDPART_SIZ_FULL,
offset: 0x00500000
}
};
```

3.7 Mtd device

Mtd devices (memory technology device) is intended for attending flash memory. Mtd devices are defined during the kernel compilation. You can use these devices in linux in our configuration:

Block Devices:

/dev/mtdblock0	boot loader
/dev/mtdblock1	linux kernel
/dev/mtdblock2	file system cramfs
/dev/mtdblock3	file system JFFS2

Character Devices:

/dev/mtd0	boot loader,rw
/dev/mtd1	boot loader,ro
/dev/mtd2	linux kernel,rw
/dev/mtd3	linux kernel,ro
/dev/mtd4	file system,rw
/dev/mtd5	file system,ro
/dev/mtd6	file system,rw
/dev/mtd7	file system,ro

You can operate a flash directly from running operating system.

Examples:

`eraseall /dev/mtd2`	erase LinuxKernel partition
`cat zImage >> /dev/mtd2`	write new LinuxKernel
`mount /dev/mtdblock2 -t jffs2 /`	mount root

3.7.1 Creating root file system

3.7.2 JFFS2 file system

3.7.3 CRAMFS file system

Chapter 4

Board using

4.1 Connecting to the board

We need two connection : via serial port and ethenet network. Serial port we use as console and network we use for downloading file to the board memory.

Serial console Use direct serial cable, connect it to RS232 on PXABoard and computer. Start terminal on computer (on Debian minicom, gtkterm) with cofiguration:

38400 baud rate
8 data bits
None Parity
1 Stop-Bit
None Flow Control

Ethernet network Use crossover ethernet cable or hub with normal direct cable.

Default address of board is set to `192.168.1.100`. The default password for user `root` is `root`.

4.2 Flashing boot loader via JTAG

First we have to copy developing boot loader `armboot` into flash memory. We can use `jtag`.

```
# jtag cable
jtag 0.4
...
Initializing Xilinx DLC5 JTAG Parallel Cable III on parallel port at 0x378
Device Id: 01001001001001100100000000010011
```

```
  Manufacturer: Intel
  Part:         PXA250
  Stepping:     B2
...
Device geometry definition:
        Device Size: 8388608 B (8192 KiB, 8 MiB)
        Flash Device Interface Code description: 0x0002 (x8/x16)
        Maximum number of bytes in multi-byte program: 32
        Number of Erase Block Regions within device: 1
        Erase Block Region Information:
                Region 0:
                        Erase Block Size: 131072 B (128 KiB)
                        Number of Erase Blocks: 64
jtag>flashmem 0x0 armboot.bin
Note: Supported configuration is 2 x 16 bit or 1 x 16 bit only
Manufacturer: Intel
Chip: 28F640J3A
program:

block 0 unlocked
erasing block 0: 0
addr: 0x0001416C
verify:
addr: 0x0001416C
Done.
jtag>quit
```

4.3 Flashing kernel and rootfs via Ethernet network

Now you see in terminal this dump after board start:

```
ARMboot 1.0.2 (Jun 18 2003 - 21:06:16)

ARMboot code: a1000000 -> a101759c
CPU: Intel XScale-PXA250 (ARM 5TE) revision B2
Clock: Mem=99.53MHz (*27), Run=199.07MHz (*2), Turbo=199.07MHz (*1.0,inactive)
DRAM Configuration:
Bank #0: a0000000 64 MB
Bank #1: a4000000 0 KB
```

```
Bank #2: a8000000 0 KB
Bank #3: ac000000 0 KB
Flash: 16 MB

Hit any key to stop autoboot:  0
Unknown command 'FIXME' - try 'help'
ArmBoot>
```

Firstly we set IP address board and tftp server. Default address is 192.168.1.100 for board and 192.168.1.111 for tftp server. We can change it, but it depend on your network enviroment.

```
ArmBoot> setenv ipaddr  192.168.1.100
ArmBoot> setenv serverip 192.168.1.111
ArmBoot> saveenv
Un-Protected 1 sectors
Saving Environment to Flash... done.
Protected 1 sectors
ArmBoot>
```

Then we erese flash for kernel (bank 1, sector 1 to 3).

```
ArmBoot> erase 1:1-3
Erase Flash Sectors 1-3 in Bank # 1:
Erasing sector  1 ... ok.
Erasing sector  2 ... ok.
Erasing sector  3 ... ok.
Done.
ArmBoot>
```

Now we can download kernel to the memory and then copy to the flash (0xa0000000 is in RAM, 0x40000 is flash and 0x30000 is length). Length is in long (4bytes), because the bus width is 32 bit.

```
ArmBoot> tftpboot 0xa0000000 pxa/zImage
ARP broadcast 1
eth addr: 00:50:04:e0:31:f3
TFTP from server 192.168.1.111; our IP address is 192.168.1.100
Filename 'pxa/zImage'.
Load address: 0xa0000000
Loading: #################################
done
```

```
Bytes transferred = 654788 (9fdc4 hex)
ArmBoot> cp 0xa0000000 0x40000 0x30000
Copy to Flash... 100% done.
ArmBoot>
```

We erese flash for rootfs (bank 1, sector 4 to 20),

```
ArmBoot> erase 1:4-20
Erase Flash Sectors 1-3 in Bank # 1:
Erasing sector  1 ... ok.
...
Erasing sector 20 ... ok.
Done.
ArmBoot>
```

download `/rootfs` to the memory and copy to the flash.

```
ArmBoot> tftpboot 0xa0000000 pxa/crfs-root.bin
ARP broadcast 1
eth addr: 00:50:04:e0:31:f3
TFTP from server 192.168.1.111; our IP address is 192.168.1.100
Filename 'voipac2/crfs-root.bin'.
Load address: 0xa0000000
Loading: #################################
done
Bytes transferred = 4075520 (3e3000 hex)
ArmBoot> cp 0xa0000000 0x100000 0x100000
Copy to Flash... 100% done.
```

Now we can start kernel by command `go 0x40000`. The default password for user `root` is `root`. You can connect via serial port again and now via ethernet network `ssh 192.168.1.100` too.

Chapter 5

Examples

5.1 Compiling application

5.2 “Hello World”

5.3 Test of GPIO Led

5.4 Simple serial port application

5.5 Simple client/server application

Chapter 6

Resources

The primary source about ARM architekture are web pages `www.arm.com`. There you can obtain Arm Reference Manual `www.arm.com/ARMARM`. The information about Xscale architekture are available on `developer.intel.com`.. Sources and manuals for `gcc`, `gdb`, `glib`+ and binary utilities are on `www.gnu.org`. Kernel souces and patches are on the page `www.arm.linux.org.uk`.