Porting Linux 2.6.9

to the PXA270 based development platform

Intel Glencoe Beyond3G

Final Technical Report

as part of a

research collaboration between

Intel

and

Carnegie Mellon University

Author: Matthias Ihmig, m.ihmig@mytum.de
Last Change: 05/11/2005

Table of Contents

Porting Linux 2.6.9 1
1. Introduction 4
2. The Linux Boot Loader BLOB 5
2.1. Defaults and configuration values 5
2.1.1. Clock settings 5
2.1.2. Flash partition table 6
2.1.3. Other default settings 7
2.2. How to build BLOB 7
2.3. How to flash BLOB to the Glencoe board 8
2.3.1. Using jflashmm 8
2.3.2. How to flash BLOB, kernel and rootfs to Glencoe board using usbdnet 8
2.4. How to debug blob with the Intel XDB debugger 9
3. A Linux Kernel for the Glencoe board 10
3.1. PXA27x specific adaptions and patches 10
3.1.1. AC97 audio and touchscreen 11
3.1.2. Flash drivers (MTD&JFFS2) 11
3.1.3. MultiMediaCard, SecureDigital card 11
3.1.4. Compactflash interface 12
3.1.5. Serial ports (FFUART, BTUART, STUART) 12
3.1.6. USB host and gadget support 12
3.1.7. Wireless updates 12
3.2. Downloading and patching the Glencoe kernel 13
3.3. Compiling, uploading and running the kernel 13
3.4. Debugging the kernel 14
3.4.1. Using the kgdb patches 14
3.4.2. Using the Intel JTAG cable 14
3.4.2.1. Modifications to the Glencoe developer board 14
3.4.2.2. Using the XDB debugger (SDTool Suite 2.0 Beta22.0_beta) 15
4. Glencoe specific peripherals 16
4.1. SyChip Wireless LAN CompactFlash adapter 16
4.2. WM9712 sound codec 16
4.3. Sahara USB Bluetooth module 17
4.4. Fastap keyboard 17
4.4.1. Default mapping for FastAp keypad 17
4.4.2. Remapping scancodes with setkeycodes 18
4.4.3. Using the keyboard driver 18
4.5. Xircom/Intel GEM GSM/GPRS module 19
4.5.1. Controlling the GEM module through GPIO lines 19
4.5.2. Using the GEM module as serial modem 19
4.6. Access to Glencoe's peripherals (LEDs, backlight) 20
5. The root filesystem 21

5.1. /etc/ 21
5.2. /dev/ 22
5.3. /proc/ 22
5.4. /tmp/ 23
5.5. Software overview and capabilities walkthrough 23
5.5.1. Overview over installed software 23
5.5.2. A few system utilities 24
5.5.3. CF/PCMCIA hardware 25
5.5.4. USB Host 25
5.5.5. USB gagdet support 25
5.5.6. Serial port 26
5.5.7. Bluetooth 26
5.5.8. Networking 28
5.5.9. Console, file system and multimedia utilities 28
A.1. Installing the arm toolchain for cross compiling 30
A.2. Creating an environment for remote debugging 30
A.3. HowTo setup USBDNET 31
A.4.1. Installing USB Ethernet on host 31
A.4.2. Configuring USB Ethernet 31
A.4.3. Using Kernel 2.6 on host side with usbnet 32
A.5. Mounting MTD & JFFS2 images on a PC system 32
B. Links of Interest 33

Revision History

Date	Author	Changes
22 Dec 05	Matthias Ihmig	Intermediate report (BLOB & linux-2.4.21
11 May 05	Matthias Ihmig	Release to Intel

1. Introduction

The “Glencoe” board uses – like Mainstone – the Intel Xscale CPU PXA270. An existing Linux kernel with patches available for the “Mainstone” board has been used as the base system.

In the beginning of the porting process, the functionality of the 2.6 kernel based patches for the PXA270 was very limited, so the first goal was to get the 2.4.21 kernel running on Glencoe.

As it turned out in the end of 2004, this kernel version `2.4.21-rmk1-pxa1-intc3` had no PCMCIA drivers for the PXA27x, the MMC/SD modules were still buggy and due to the outdated kernel version, there is no longer active support on the arm-linux mailing list (see [2]).

In the meantime, a PXA270 patchset appeared for the 2.6.9 kernel on which future work could be based, so the switch to the 2.6.9 kernel was made for the Glencoe board. In the end of January, which was pretty late in our porting process, the Mainstone group also released a number of patches based on the 2.6.9 kernel. As these patches mostly concerned general PXA support and were not Mainstone specific, this code was also integrated and partly adapted.

Following is a more detailed overview over the porting status, hints and examples for the bootloader, the kernel and the root filesystem.

2. The Linux Boot Loader BLOB

As Linux boot loader, the “Boot Loader Object” from the LART project originally available from [3] has been used. The Mainstone enabled version is also available from [1].

For the Glencoe board, the Mainstone enabled version 2.0.5-pre3 has been used. It is available from here: ftp://ftp.arm.linux.org.uk/pub/armlinux/people/xscale/mainstone/ 01-27-2005/src/blob/blob-xlli-snapshot-123004.tgz

A new board name “glencoe” has been added to BLOB, which activates and includes all parts and changes necessary to work with the Glencoe board.

A new architecture GLENCOE has been defined with ARCH_NUMBER(4061) in include/blob/linux.h.

Some bug fixes, like the reboot functionality, has been added and can be also integrated into the Mainstone tree.

To add Glencoe support to the Mainstone sources, apply blob-xscale-mih1-glencoe.patch:

```
$ tar xfvz blob-xlli-snapshot-123004.tgz
$ cd blob-xscale
$ patch -p1 < ../blob-xscale-mih1-glencoe.patch
```

2.1. Defaults and configuration values

The assembler code in src/blob/xlli/glencoe/ takes care of basic initialization that needs to be done before the part of BLOB written in C can start, like register settings for correct memory and cpu clock.

2.1.1. Clock settings

The hardware switches used on the Mainstone board are not available for Glencoe. They were used to **determine clock speed and multiplier settings for the PXA27x CPU**. These settings can now be made **by changing** the .equ values of **xlli_CLOCK_SWITCH_*** at the end of src/blob/xlli/glencoe/xlli_Glencoe_defs.inc. In order to tune the clock settings to the requested speed, edit the corresponding variables in that file.

Be aware that wrong settings may damage the CPU permanently due to overclocking (see Mainstone documents for more details on switch settings).

Other hardware initialization (e.g. switching on LCD and background light) is done in src/blob/xlli/glencoe/start.s and in src/blob/main.c.

The default settings are as following:

```
blob> clockinfo
Run Mode clock: 247.00MHz (*19)
Turbo Mode clock: 494.00MHz (*2.0, active)
Memory clock: 123.50MHz (Alt=0, SDCLK[0]=/4, SDCLK[1]=/2)
System bus clock: 123.50MHz
```

2.1.2. Flash partition table

The file `include/blob/arch/glencoe.h` contains **board specific definitions** like **console serial port**, **FLASH and RAM memory layout, default initialization values** for GPIO registers used later in C part of BLOB.

Memory layout (a.k.a. Flash Partition Table):

	within FLASH	**within RAM**	**Partition Length**
BLOB (“`blob`”)	0x0000 0000	0xa020 0000	256 kB = 0x0004 0000
Kernel (“`zImage`”)	0x0004 0000	0xa000 8000	2 MB = 0x0020 0000
JFFS2 FS (“`rootfs`”)	0x0024 0000	0xa050 0000	29.5MB = 0x01d8 0000
Params (“param”)	0x01fc 0000	0xa026 0000	256 kB = 0x0004 0000

NOTE: `rootfs` is also sometimes referred to as `ramdisk` within BLOB. In the Glencoe case, this is not an initial ramdisk (initrd), as the kernel can boot directly from the FLASH root filesystem.

The file `src/blob/glencoe.c` contains a hardcoded flash partition table and other board specific functions.

Default Flash partition table (for sizes, see table above) and Glencoe specific FLASH write&erase wrapper functions.

The flash partition table can be stored in a special parameter partition in the Flash, but default values are hardcoded as shown above for improved usability. This allows for a quick installation without the need to manually create a partition table.

However, this hardcoded table is only used when no other partition table is found. It is possible to put a partition table in the “param” partition without further code modifications (untested). Again, this is not necessary for operation.

N. B.: f modifying the hardcoded partition table in blob in `include/blob/arch/glencoe.h` and `src/blob/glencoe.c`, it should also be made available to the kernel by either specifying an appropriate kernel parameter or by adapting the hardcoded **default partition table in the Linux kernel** in `drivers/mtd/maps/pxa27x-flash.c`.

The **format for the partition table in the kernel command line** is as follows:

```
mtdparts=<mtddef>[;<mtddef]
<mtddef> := <mtd-id>:<partdef>[,<partdef>]
<partdef> := <size>[@offset][<name>][ro]
<mtd-id> := unique id used in mapping driver/device (here:pxa27x-flash)
<size> := standard linux memsize OR "-" to denote all remaining space
<name> := (NAME)
```

Due to the way Linux handles the command line, no spaces are allowed in the partition definition, including mtd id's and partition names.

Example: (equiv. to existing flash partition table):

```
mtdparts=pxa27x-flash:256k@0k(blob),2M@256k(zImage),30208k@2304k(rootfs),-(param)
```

The **maximum amount of available flash memory** in Mbyte can be defined through

- in blob: `include/blob/arch/glencoe.h: #define FULL_FLASHMEM_LEN`
- in the kernel: `make xconfig`
  in `MTD`

```
    -> Mapping drivers for chip access
          -> CFI Flash device mapped on PXA27x boards
                -> Window size of Flash memory
```

The partition size for blob, kernel and param is kept constant, while rootfs gets the remaining amount of available flash memory.

The parameter partition `Param` is unused right now. In future extensions, values like CPU and Memory clock settings or a partition table may be stored here. Note that the block erase size within the parameter partition is smaller than in the rest of the Flash memory (see L18 flash specification sheet for details). To circumvent problems from the beginning, it is advised to exclude this area from the rootfs partition.

The **static partition table in the kernel** is defined in `drivers/mtd/maps/glencoe.c`

2.1.3. Other default settings

- **console serial port interface:** default serial settings are STUART port with 115200 baud, 8N1.
- **Default** initialization values for **server and client IP address** in case of using usbd network support are

Server:	192.168.1.100
Glencoe:	192.168.1.101

2.2. How to build BLOB

Depending on the Linux environment and the tool chain installation, the following commands can be put into a script (see the script `myblobmake` in the blob package) and used to compile blob.

```
export myarmdir=$HOME/arm/arm-linux-3.4.2-oe
export myarmlinux=$HOME/arm/linux-2.6.9-intc1-mih1
export WANT_AUTOMAKE=1.8
export WANT_AUTOCONF=2.5
export CC=$myarmdir/bin/arm-linux-gcc
export OBJCOPY=$myarmdir/bin/arm-linux-objcopy
export LD=$myarmdir/bin/arm-linux-ld
export STRIP=$myarmdir/bin/arm-linux-strip
export AR=$myarmdir/bin/arm-linux-ar
export RANLIB=$myarmdir/bin/arm-linux-ranlib
export PATH=$myarmdir/arm-linux/bin:$myarmdir/lib/gcc-lib/arm-linux/3.3.2:$PATH

        make -f Makefile.cvs
        make -f Makefile.cvs

        ./configure --host=arm-linux --with-board=glencoe --enable-xlli --enable-
        usbdnet --enable-network --with-linux-prefix=$myarmlinux --enable-
        maintainer-mode --enable-memtest --enable-zimage –with-commands=all

        make
```

The flashable blob binary can then be found in `src/blob/blob`.

2.3. How to flash BLOB to the Glencoe board

2.3.1. Using `jflashmm`

For the first board bring-up, when the FLASH is empty and no bootloader is installed, BLOB can be uploaded using JTAG with a corresponding cable and utility.

The bootloader must be located at the very beginning of the FLASH memory, using bulbcx.dat as configuration file.

```
        jflashmm bulbcx blob
```

In Linux, jflashmm can be used through WINE, which is slower, but works. Add the following lines to .wine/config and make sure, no parallel port module shows up in /proc/ioports:

```
[ports]
"read"  = "0x378-0x37a"
"write" = "0x378-0x37a"
```

Note: The bootloader can also be flashed to Glencoe through the Intel JTAG cable and the XDB Debugger. Use `plugin\intelxsc\flash\configs\NBMMNS2BVS_led.fcf` as flash configuration file.

2.3.2. How to flash BLOB, kernel and rootfs to Glencoe board using `usbdnet`

Later, when a previous instance of BLOB is running, files such as a new version of BLOB, the Linux kernel or a root filesystem can be uploaded through serial connection or by using usbdnet/TFTP. Note that there are no drivers for the compact flash subsystem available, therefore using a flash card or an ethernet CF card is not yet possible with blob on a PXA270.

For downloading files through the serial port, the command `download <file> <partition>` can be used. Files must be uuencoded for uploading. If receive errors occur, additional delay between characters may have to be included, as the serial routines are not interrupt-based.

For speed and reliability, it is better to use the built-in usb-client module. This configures the Glencoe board as a USB gadget, i.e. the board appears as USB network devices when plugged into a host PC via a simple USB cable. See below for instructions on how to install usbdnet.

To download a file into RAM through USB, enter

`blob> tdownload blob`	for updating blob, or
`blob> tdownload kernel zImage`	for the Linux kernel (zImage), or

```
blob> tdownload rootfs ramdisk
```
for root filesystem (jffs2 image),

plug in the USB cable and manually configure usb0 on host by running

```
$ ifconfig usb0 192.168.1.100
```

When usb0 is correctly setup as a network device, and USB hotplugging services and TFTP are installed on the host machine, downloading automatically starts a few seconds after inserting the USB cable without the need to run `ifconfig` manually.

When finished downloading, the file can be stored into flash using

`blob> flash blob`	for blob, or
`blob> flash kernel zImage`	for Linux kernel (zImage), or
`blob> flash rootfs ramdisk`	for root filesystem (jffs2 image),

The kernel can be started with the `boot` command.

When Linux is already running, the kernel can also be updated in a quicker way by downloading the zImage and copying to /dev/mtdblock1

```
$ wget host/tftp/kernel
$ cp kernel /dev/mtdblock1
```

NOTE: When flashing an unpadded rootfs (i.e. smaller than the partition size), be sure to erase the partition completely to avoid old “ghost” files.

2.4. How to debug blob with the Intel XDB debugger

Compiled blob with debugging enabled:

```
./configure --host=arm-linux --with-board=glencoe --enable-xlli --enable-
usbdnet --enable-network --with-linux-prefix=$myarmlinuxpath --enable-
maintainer-mode --enable-memtest --enable-zimage --with-commands=all –enable-
blob-debug

make
```

Make the complete blob source directory available to the Windows machine with the XDB debugger, e.g. by sharing through samba or simply by copying the complete directory.

Then create the bd-file with the symbol database:

```
blob-2.0.5-pre3-glencoe05\src\blob> dwarf2bd.exe blob-elf32
```

and load it with File|Load into the XDB debugger.

You may be asked to locate start.s afterwards. Locate it:

```
blob-2.0.5-pre3-glencoe05\src\blob\xlli\glencoe\start.s
```

3. A Linux Kernel for the Glencoe board

Originally the porting began based on the 2.4.21 Linux kernel version from http://www.kernel.org and existing patches (Russell's arm-linux patches, Nicolas' pxa patches and Intel ICG PCG/SE Linux Team's intc3 patches) as a base system for own modifications and extension necessary for the Glencoe board.

In December 2004, the switch to linux-2.6.9 was made. To get the best possible working system, patches and updates from different sources have been used. See below for a more detailed overview over patches, what they do and where they come from.

The Glencoe patchset consists of updates, changes and merges from Toby Churchill (esp. SD card support), Liam Girdwood (sound and touchscreen) and backports from later kernels or updated CVS versions (MTD, JFFS2, ALSA).

For this purpose, a **new machine type `ARCH_GLENCOE`** has been included in the kernel sources. The actual mach-types number assigned to `ARCH_GLENCOE` is not yet officially registered at the arm-linux.org.uk project. It is defined in `arch/arm/tools/mach-types`.

The file `arch/arm/mach-pxa/glencoe.c` contains board specific initializations like general init functions, setting of power status flags, configuring and powering the **serial ports** for console and debug output and **additional IRQ setup**.

An adapted module for using `HEARTBEAT_LED` and `SYS_BUSY_LED` on the Glencoe board is available in `arch/arm/mach-pxa`. Power management routines in `pm.c` are different from Mainstone board and have been corrected for the 2.4 kernel.

The board dependent flash initialization functions along with the **flash partition table** are included in `drivers/mtd/maps/glencoe.c`. See the flash chapter in the blob section for more details on the flash.

Settings for the **LCD display** framebuffer driver are contained in `drivers/video/pxafb.h`.

In `include/asm-arm/arch-pxa/glencoe.h` **GPIO pin variables** like LEDs are defined.

3.1. PXA27x specific adaptions and patches

Due to bugs or incompatibilities in the existing Mainstone patches, updates, code merges and backports from later kernel versions have been necessary for the Glencoe board. Following is a more detailed overview over the changes that were made for the PXA27x subsystems and their drivers.

The Mainstone patches add support for SRAM, framebuffer overlay, camera interface (included but not yet adapted for Glencoe), RTC and dynamic frequency and voltage management, some of which needed modifications to also work with Glencoe.

See ftp://ftp.arm.linux.org.uk/pub/armlinux/people/xscale/mainstone/02-25-2005/README.txt for more details on the Mainstone patches.

3.1.1. AC97 audio and touchscreen

Both OSS (Open Sound System OSS) and ALSA (Advanced Linux Sound Architecture) are included, but with different restrictions.

- The AC97 **OSS** drivers in `sound/oss` are mainly based on Liam Girdwoods version from September 2004 with own changes and backports from later kernel versions to improve behaviour with MMC/SD cards. In the current state with OSS, playback from MMC card is possible. But recording audio after an MMC/SD card has been inserted is not possible due to a hardware problem with the PXA270.
- The AC97 **ALSA** driver support for the PXA are based on a CVS snapshot from the ALSA tree (http://www.alsa-project.org/) from 04/03/2005 which have been merged into the Glencoe kernel.

The AC97 ALSA driver in `sound/arm/` and `sound/pci/ac97/` have a workaround implemented, so recording audio to MMC/SD is now possible with ALSA.

This version of ALSA has known issues with power management, which are fixed in 2.6.12. But due to API changes, backporting the 2.6.12 ALSA drivers is more difficult than completely switching to a newer kernel version.

The **touchscreen drivers** work only in OSS right now – for ALSA, they are work in progress by Liam Girdwood. An MCP framework by Nicolas Pitre (http://lists.arm.linux.org.uk/pipermail/linux-arm-kernel/2005-March/027990.html) is included, which supposedly works with the ALSA and the UCB1x00 touchscreen, but not yet with the WM97xx. This is most probably resolved within the next three months.

As the sound chip drivers are backports from later kernels, which use different functions for memory access, a few functions concerning dma memory mapping were backported to linux-2.6.9.

3.1.2. Flash drivers (MTD&JFFS2)

The mtd drivers from the original 2.6.9 kernel had some bugs which could destroy the root filesystem when the end of the flash was reached and erase sizes were mixed like with the L18 flash.

To fix this, an updated version of the MTD and JFFS2 layer has been merged into the kernel.
The merged code comes from a CVS snapshop of the MTD tree from 03/20/2005 (http://www.linux-mtd.infradead.org/) and now supports additional flash devices.

Also a few changes have been made to drivers/mtd/map/pxa27x-flash.c to enable easier adaption to changed flash size. See the flash chapter in the blob section for more details on the flash.

3.1.3. MultiMediaCard, SecureDigital card

Two main changes were made to the MMC drivers, backport from 2.6.11 to get a number of bugfixes and adding support for SD cards. The SD code was written by Ian Molton and was merged from Toby Churchill's patches available here: http://www.toby-churchill.org/balloon/patches/patch-linux-2.6.9-rc3-tcl1-20041008.gz

Be aware that there is a hardware bug in the PXA causing IRQ conflicts between MMC and AC97, breaking recording audio after an MMC has been inserted.

Also note that after a software reboot (a hardware reset works fine), the MMC card needs to be re-inserted manually before it will be recognized.

3.1.4. Compactflash interface

The compactflash interface is similar to the PCMCIA interface, where the main differences are in the supported voltage and pin-outs. A CF interface driver has been written for Glencoe, based on Mainstone's PCMCIA interface drivers. The existing userland tools like cardctl and cardmgr can be used.

3.1.5. Serial ports (FFUART, BTUART, STUART)

The serial port for initial output before the actual message console is activated can now be set in the kernel config in the architectures section in “Initial UART output port”.

This affects the output during uncompressing the kernel and debug and other outputs of the kernel messages.

Devices:

```
FFUART <--> /dev/ttyS0
BTUART <--> /dev/ttyS1
STUART <--> /dev/ttyS2
```

A bug using `printascii()` and `printk()` concurrently has been corrected in the 2.4. kernel. This bug occurred when a special combination of enabled debugging outputs was turned on and resulted in blocking the IRQ subsystem. This problem did not occur in linux-2.6.9.

3.1.6. USB host and gadget support

USB has been adapted for power-up, remaining untested in the 2.4.21 kernel version, but was running successful in the 2.6.9 kernel version. A USB keyboard and USB mouse were recognized correctly and worked flawlessly on the Glencoe board. Note that revisions of Glencoe up to 1.1 need an external power supply for USB host to work. See the chapter in the rootfs section on how to use it.

3.1.7. Wireless updates

The Wireless Extension kernel interface has been updated from v16 to v17 using patches originating here: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

For easier support of Prism wireless chips, the HostAp driver modules have also been included into the kernel tree. Later updates to drivers and tools are available on http://hostap.epitest.fi/

3.2. Downloading and patching the Glencoe kernel

The base for the Glencoe patchset is derived from the kernel available here:
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.9.tar.gz

and the Mainstone patches available here:
ftp://ftp.arm.linux.org.uk/pub/armlinux/people/xscale/mainstone/02-25-2005/src/kernel/patch-2.6.9-intc1

The Glencoe patches are also available combined in a single file: linux-2.6.9-intc1-mih1.patch.

To patch the mainline kernel for Glencoe support, do the following:

```
$ tar xfvz linux-2.6.9.tar.gz
$ cd linux-2.6.9
$ patch -p1 < ../patch-2.6.9-intc1
$ patch -p1 < ../linux-2.6.9-intc1-mih1.patch
```

3.3. Compiling, uploading and running the kernel

To **change the default config** for the kernel, edit `arch/arm/def-configs/glencoe_defconfig` .

Among other things, the **default kernel command line** can be set here with `CONFIG_CMDLINE`:

```
CONFIG_CMDLINE="root=/dev/mtdblock2 noirqdebug console=ttyS2,115200 mem=64M"
```

Disabling irqdebug suppresses “IRQ: nobody cared” messages, which happen due to updates in the IRQ subsystem (http://lists.arm.linux.org.uk/pipermail/linux-arm-kernel/2004-November/025278.html)

To **build the kernel** from sources, a script like the following can be used.

```
export PATH=$HOME/arm/arm-linux-3.4.2-oe/bin:$PATH
export INSTALL_MOD_PATH=/tftp/fs

export WANT_AUTOMAKE=1.8
export WANT_AUTOCONF=2.5
export ARCH=arm
export CROSS_COMPILE=arm-linux-

make glencoe_defconfig && make oldconfig
make && make modules_install && cp arch/arm/boot/zImage /tftp/kernel
```

To change the configuration, run `make xconfig` after `make glencoe_defconfig`

To **transfer the kernel** from the Linux machine to the Glencoe board, TFTP over usbdnet can be used. The last cp command assumes an existing tftp directory. See previous section for details how to download a file using BLOB and see instructions below on how to setup `usbdnet`.

A root filesystem is not necessary just for booting up the kernel to see if it works. However to do anything useful, e.g. get a login console, a rootfs is required.

After downloading, boot the kernel with

```
      blob> boot
```

Optional parameters can be appended:

```
      blob> boot root=/dev/mtdblock2 console=ttyS2,115200 noirqdebug mem=64M debug
```

3.4. Debugging the kernel

3.4.1. Using the kgdb patches

It is possible to remotely debug kernel over serial port. This is a nice option for emergencies, but it is limited to compiled-in kernel modules. The extension for loading modules later did not work. Also very slow

The following description is based on work with the linux-2.6.9 version of the kernel.

Apply linux-2.6.9-pxa-kgdb.patch:

```
linux-2.6.9-kgdb $ patch -p1 < kgdb-pxa-2.6.9-intc1-mih1.patch
```

This patch is not included in `linux-2.6.9-intc1-mih1.patch`.

In “General setup”, activate KGDB support, KGDB over serial port, “Console messages over KGDB” and “Enable SysRq-G command”.

Selecting a debugging serial port in the kernel config automatically excludes this port from the serial driver to circumvent conflicts. This is defined in `drivers/serial/pxa.c`

Note: The KGDB patch does not work with the precompiled toolchain from the 01-27-2005 Mainstone patches. Use the old precompiled toolchain from 02-28-2004 instead.

Using GDB: `arm-linux-gdb vmlinux`

Using DDD: `ddd --debugger arm-linux-gdb vmlinux`

```
(gdb) target remote /dev/ttyS0
```

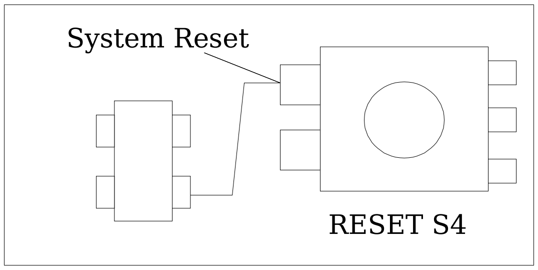
For more information about debugging software, see:
http://www.dsplinux.net/bsp/CadenuxArmUsersGuide.html

3.4.2. Using the Intel JTAG cable

A better possibility for debugging is the use of Intel's XDB Debugger. It allows simulation as well as in-circuit debugging through the JTAG interface. The MEMEC Insight cable used for flashing does not work with the XDB debugger.

3.4.2.1. Modifications to the Glencoe developer board

The existing Glencoe JTAG interface does not directly work with Glencoe. The following modifications are necessary:



- **nTRST can not drive the SystemRST line**
  - **Remove R144** to separate nTRST and SystemRST lines (1cm above the PXA)
  - Solder a wire to the Reset Switch connector pin 2 that connects to pin 3 of U22 (see figure)
- **The JTAG 3.3V line cannot power the Intel JTAG cable**
  - Solder a wire to the lower end of R33 (3.3V_SB) and connect the Intel JTAG cable power pin here.

Intel JTAG cable pinout:

Pin	Function
1 and 2	VCC 3.3V (both pins need to be connected to VCC)
3	nTRST
5	TDI
7	TMS
9	TCK
13	TDO
15	System Reset
20	GND

3.4.2.2. Using the XDB debugger (SDTool Suite 2.0 Beta22.0_beta)

After starting the XDB Debugger, select Project File:

```
xdb\configurations\jtaglinux\nbmmns2bvs.xsf

[Common Settings]
Interactive mode
Target: JTAG CPU
Working Directory: c:\linux
Batch File: .\configurations\jtaglinux\nbmmns2bvs.xdb
Batch Arguments: c:\linux\vmlinux

[CPU - JTAG]
Target Connection: lpt1:1
JTAG Device: Intel(R) JTAG Cable
Scan-chain: NBMMNS2BVS
Monitor at: 0xfffff000
```

When the symbolfile (.BD) has been generated and loaded, the debugger is ready and the breakpoint at &start_kernel reached, run the batch file:

```
./Example/ocdxs/batch/exoff_linux.xdb
```

This is necessary to enable debugging after enabling the MMU.

4. Glencoe specific peripherals

4.1. SyChip Wireless LAN CompactFlash adapter

The Prism3 based WLAN card seems to have problems with the existing Prism firmware. Several combinations of available Primary and Secondary (STA) firmware versions have been tested. Some didn't work at all, while some resulted in limited support.

The linux-wlan-ng drivers from AbsoluteValue Systems (http://www.linux-wlan.com/) broke after receiving an IP over DHCP and didn't recover.

The HostAp drivers, which also support a number of Prism chipsets, resets the card automatically after a Timeout occurred and in general seemed more stable. For this reason, and for the possibility to act as Wireless Access Point, the HostAp driver was used.

The recurring card timeouts remain, and during load occur at about every 30s. According to reports from different people using Prism cards in Linux, who have similar card timeouts, the only solution to them was updating the card firmware. But despite usting the latest firmware (PRI 1.1.4 and STA 1.8.3) the problems remain.

For the PCMCIA system, this card is defined in `/etc/pcmcia/hostap_cs.conf` in the root filesystem.

Mini-howto on Flashing Intersil Prism Chipsets, also includes links to different firmware versions can be found here: http://linux.junsun.net/intersil-prism/

In order to rule out Glencoe- and software problems, another CF card was used, which turned out to work reliably without any dropout or other issues.

This card is a 802.11b Linksys WCF12 Compactflash card and works with the orinoco driver.

4.2. WM9712 sound codec

On Glencoe Revision 1.2 it appears that the GEM beeper is connected to the beeper output of the sound codec, which outputs a lot of noise as soon as it is running.

When using OSS, it cannot be muted once unmuted. To keep OSS from activating the channel e.g. by using a mixer like aumix, this channel can be disabled completely in the kernel config: `Sound->Open Sound System->PXA Audio->Disable mixer initialization for SPEAKER`. This workaround is implemented in `sound/oss/ac97_codec.c`

For more information about AC97 sound and touchscreen, see the chapter in the kernel section.

For ALSA, the OSS-compatibility modules are also loaded, so legacy applications using the OSS interface still work with ALSA.

4.3. Sahara USB Bluetooth module

The Sahara USB Bluetooth module uses the National semiconductor chipset LMX9814SB which is already supported in the existing Bluetooth stack bluez (http://www.bluez.org/).

Interestingly enough, it seemed that actual connections were only possible about 30% of the time. Even name queries often failed. This behaviour did occur in both Linux and Windows, leading to rather suspect the hardware.

Further tests with another Bluetooth module revealed that this problem was not caused by driver problems, as this module worked reliably – it is by BlueTake with a CSR BT radio chipset.

4.4. Fastap keyboard

A kernel driver for the FastAp serial keyboard can be found in `drivers/input/keyboard/fastap_serial.c`.

On Glencoe, the alphanumeric keys are connected to a serial port (B9600, 8n1)

To compile the kernel driver, activate CONFIG_KEYBOARD_FASTAPSERIAL in the Input Device Drivers section.

Despite the name, it also takes care of the PXA27x's DKPIN input.

4.4.1. Default mapping for FastAp keypad

The **default mapping** can be changed in `drivers/input/keyboard/fastap_serial.c`. The array `fastap_keycode` maps the serial interfaced keys to keycodes, the array `fastap_directkey` the DKIN keys to keycodes.

The printed characters on the pad are mapped to a default US keyboard map. These keycodes can be looked up in `include/linux/input.h`.

The **DirectInput keys** are mapped as follows:

- Top left --> Left SHIFT key
- Top right --> Left ALT key
- Green phone --> Left CTRL key
- Red phone --> ESC
- Up/Down/Left/Right keys are mapped to cursor keys

The **lowest row of keys** is mapped as follows:

- Home symbol --> QT_MENU (65)
- Arrow Up --> QT_SEL (116)
- Blue dot --> KEY_INSERT (110)
- Arrow Up-Left --> QT_BACK (67)

Additionally, there are two types of **special keypand mappings** implemented:

- Shift + Number --> Fn key
- Alt + Cursor key --> Home/End/PgUp/PgDown

4.4.2. Remapping scancodes with setkeycodes

Scancodes of the keys interfaced through the serial port can be remapped on the fly with

```
      $ setkeycodes <scancode in hexadecimal> <keycode in decimal>.
```

FastAp Scancode Table: (see `drivers/input/keyboard/fasta_serial.c`)

```
0x00-a  01-1    02-b  03-~! 04-c  05-2    06-d    07-|?    08-e     09-3     0A-f
0x10-g  11-4    12-h  13-^$ 14-i  15-5    16-j    17-_&    18-k     19-6     1A-l
0x20-m  21-7    22-n  23-+: 24-o  25-8    26-p    27--;    28-q     29-9     2A-r
0x30-s  31-*    32-t  33-"' 34-u  35-0    36-v    37-%,    38-w     39-#     3A-x
0x40-<. 41-=tab 42->@ 43-[( 44-y  45-del  46-z    47-])    48-{/    49-enter 4A-)\
0x50-home       52-ar.up    54-lspace     56-rspace        58-blue  5A-esc
```

Keycodes: see #define's in `include/linux/input.h`

Example: Map the blue key to “-” (val (KEY_MINUS) = 12)

```
      $ setkeycodes 58 12
```

4.4.3. Using the keyboard driver

To connect keypad with serial port, use the `inputattach` tool. The original sourcecode is available from http://linuxconsole.sourceforge.net/quick.html in the CVS **ruby** directory

For compiling hints, look at: http://agendawiki.com/cgi-bin/aw.pl?DatacompSerialKeyboard

To patch an unpatched version of inputattach.c, use

```
      $ patch -p2 < inputattach-fastap.patch
```

or use the supplied inputattach.c file.

To use the FastAp driver in an unpatched kernel, use

```
      linux-2.6.9 $ patch -p1 < fastap_serial-keybaord.patch
```

This patch is already included in linux.2.6.9-intc1-mih1.patch.

The FastAp device can be attached any time with

```
      $ inputattach -ftap /dev/ttyS1 &
```

This is currently done automatically in `/etc/init.d/inputattach`.

4.5. Xircom/Intel GEM GSM/GPRS module

4.5.1. Controlling the GEM module through GPIO lines

To acces and control the GEM module's pins for power control, the `/proc` interface is used.
The corresponding driver can be found in `drivers/misc/xircom_gem.c`
To include it in the kernel, in menu “Misc devices”, select "Xircom GEM Module support”

- A write of '0' de-asserts the pin (i.e. Sets it to HIGH for a negated output and vice versa)
- A write of '1' asserts the pin, delays 250ms and deasserts pin
- A write of '2' asserts the pin

Examples:

- Power down: `$ echo 1 > /proc/driver/xircom_gem/power_down`
- Wake up: `$ echo 1 > /proc/driver/xircom_gem/wake_up`
- Reset: `$ echo 1 > /proc/driver/xircom_gem/reset`
- Get sleep status: `$ cat /proc/driver/xircom_gem/asleep`

4.5.2. Using the GEM module as serial modem

The GEM module is connected to FFUART, which can be accessed through `/dev/ttyS0` as standard serial port.

4.6. Access to Glencoe's peripherals (LEDs, backlight)

For accessing and controlling Glencoe peripherals like LEDs, LCD backlight and SIM/MMC/Compactflash card detection pins, a new module has been written that enables this access through the /proc filesystem. It can be found in drivers/misc/glencoe_periph.c.

Writing 1 asserts the pin, 0 de-asserts it.

Examples:

Input (query state, returns 1 when asserted):

```
$ cat /proc/driver/glencoe_periph/gpio0_state
$ cat /proc/driver/glencoe_periph/sim_inserted
$ cat /proc/driver/glencoe_periph/mmc_inserted
$ cat /proc/driver/glencoe_periph/cf1_inserted
$ cat /proc/driver/glencoe_periph/cf2_inserted
```

Output (set state):

```
$ echo 1 > /proc/driver/glencoe_periph/green_led_on
$ echo 1 > /proc/driver/glencoe_periph/blue_led_on
$ echo 1 > /proc/driver/glencoe_periph/backlight_on
$ echo 1 > /proc/driver/glencoe_periph/lcd_on
```

Set LCD backlight brightness (value between 0-1023):

```
$ echo 300 > /proc/driver/glencoe_periph/lcd_pwm
```

Please note that setting the LEDs only works when the Timer and CPU LEDs are deactivated (disable General setup->Timer and CPU usage LEDs)

The utility `pxa27xregs` can also be used to directly access the GPIO pins through direct register access

5. The root filesystem

The rootfs image from Mainstone [1] provides the base system for the Glencoe board. The most important modifications are:

- created missing device nodes in `/dev`
- extended `/etc/inittab` to spawn login shell on `STUART`
- changed `/etc/fstab` to match Glencoe Flash filesystem
- support for USB mass storage device: created `fstab` entries, directories and `/dev` nodes
- improved startup system for easier configuration
- added a number of software tools to enable Linux kernel functionality

The cross-compiler with the basic libraries in /lib is arm-linux-gcc-3.4.2 from openembedded. See cross-compiler section for details.

To switch to a new toolchain, copy the shared object libraries and links from the cross-compiler directory:

```
$ cp -a arm-linux/lib/ /rootfs/lib/
$ arm-linux-strip -s /rootfs/lib/*
```

Be aware that binaries depend on these libraries and may not work without being recompiled (libc !).

As almost every kernel driver needs a userland utility to configure and use special functions, this chapter gives an overview of this mini-Linux-distribution.

To create the glencoe rootfs:

```
# tar xfvz glencoe-rootfs-05-03-05.tar.gz
# mkfs.jffs2 –pad=0x01d80000 -U -e 0x40000 -d rootfs/ -o /tftp/rootfs.bin
```

or unpadded:

```
# mkfs.jffs2 -U -e 0x40000 -d rootfs/ -o /tftp/rootfs.bin
```

5.1. /etc/

The following section gives an overview overview over the files in /etc

For system boot-up and initialization, the busybox initialization tool `init` loads /etc/inittab for information on which logins to spawn.

/etc/init.d/00rcS is the system init file for further initialization and is called by the previous init tool. This script does some file system clean-up, mounts the file systems, general initialization and calls:

> **/etc/init.d/01rc.init**, which is a simple replacement for the runlevels directory in bigger distributions.

and

/etc/init.d/02.rc.local, for small additional things, like loading libraries

Additional init scripts that can be added to 01rc.init are found /etc/init.d/

Configuration files, e.g. APM or Bluetooth, can be found in /etc/default and /etc/sysconfig.

The **network configuration** for pcmcia devices is configured in `/etc/pcmcia/network.opts`. Change this file, e.g. to switch from DHCP to fixed IP addresses.

The prism firmware files are placed in /etc/pcmcia/prism_fw and come from http://hostap.epitest.fi/

Of special interest may be the directories in /etc/apm/ as scripts placed here allow easy handling of **suspend/resume** events.

The files in this directory come from different sources. Most of them belong to the different packages listed below. Some script files have been newly created, others come from the familiar distribution 0.8.2, available at http://familiar.handhelds.org/releases/v0.8.2/install/download.html

5.2. /dev/

These entries are a selection of the available devices files:

- /dev/input/event0: FastAp keyboard (device usually not required)
- /dev/input/event1: touch device if touchscreen driver loaded after `init.d/inputattach`
- /dev/input/touchscreen: links to event1, which is the touchscreen device
- /dev/input/mice: USB mouse
- /dev/snd/: ALSA sound devices
- /dev/ipmc: DVFM device
- /dev/root: points to the root filesystem (in our case /dev/mtdblock2)
- /dev/mtd*: character based access to flash device
- /dev/mtdblock*: cached block devices access to flash for filesystems
- /dev/ttyS0-2: PXA serial ports (FFUART, BTUART, STUART)
- /dev/mmca#: MMC/SD partitions
- /dev/hd*: Compact Flash memory partitions
- /dev/sd*: SCSI partitions (e.g. used for USB storage)
- /dev/loop*: loopback device
- /dev/rfcomm*: Bluetooth serial port
- /dev/ttygserial: serial device for g_serial USB gadget module

Special files for special hardware may still be missing in the /dev directory. Documentation/devices.txt in the kernel directory gives an overview of all registered device numbers.

5.3. /proc/

Here are a few examples of information about the system itself. Try and see:

```
# cat /proc/mtd
# cat /proc/cpuinfo
# tail /proc/kmsg
# cat /proc/interrupts
# cat /proc/input/devices
```

5.4. /tmp/

The contents of /tmp are only stored in SDRAM and lost after a reboot. The size is limited by the amount of available RAM (e.g. use `top` for more information)

5.5. Software overview and capabilities walkthrough

5.5.1. Installed software

- Busybox 1.0, (standard linux tools), http://www.busybox.net/
- BASH 2.05a, (for a nicer shell and bash scripts), http://www.gnu.org/software/bash/bash.html
- tar 1.13, (busybox's tar is broken), http://www.gnu.org/software/tar/tar.html
- dhcpcd v.1.3.22-pl4, http://www.phystech.com/download/dhcpcd.html
- strace 4.4, http://sourceforge.net/project/showfiles.php?group_id=2861
- ser2net 2.2, http://sourceforge.net/projects/ser2net
- fbv 1.0b (framebuffer picture viewer), http://s-tech.elsat.net.pl/fbv/
- HostAp-Driver 0.3.7, http://www.hostap.epitest.fi/
- module-init-tools 3.1, ftp://ftp.kernel.org/pub/linux/kernel/people/rusty/modules/
- ncurses 5.4, http://ftp.gnu.org/pub/gnu/ncurses/ncurses-5.4.tar.gz
- PCMCIA tools 3.2.8, http://pcmcia-cs.sourceforge.net/
- picocom 1.4 (serial terminal), http://efault.net/npat/hacks/picocom/
- sqlite 3.2.1, http://www.hwaci.com/sw/sqlite/sqlite-3.2.1.tar.gz
- Wireless Tools 27, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html
- iptables 1.3.1, http://www.netfilter.org/downloads.html
- zlib 1.2.2, http://www.gzip.org/zlib/
- aumix 2.8 (OSS mixer), http://jpj.net/~trevor/aumix.html
- madplay 0.15.2b, (MP3), http://sourceforge.net/project/showfiles.php?group_id=12349
- sox 12.17.7 (SOund eXchange), http://sox.sourceforge.net/

- ALSA sound tools 1.0.9rc2, http://www.alsa-project.org/download.php
- tslib and utilities CVS 03/2005, cvs@pubcvs.arm.linux.org.uk/mnt/src/cvsroot co tslib
- libjpeg v6b, ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz

Downloaded and compiled through openembedded:

- apmd-3.3.2-r2, http://www.worldvisions.ca/~apenwarr/apmd/
- bluez-libs-2.11, bluez-utils-2.11, http://bluez.sourceforge.net/
- bridge-utils-1.0.4, http://bridge.sourceforge.net/
- dbus-0.22, http://dbus.freedesktop.org/
- dropbear-0.44 (ssh), http://matt.ucc.asn.au/dropbear/
- hostap-utils-0.3.7, http://hostap.epitest.fi/
- hotplug-20040920, http://linux-hotplug.sourceforge.net
- libpng-1.2.5, http://www.libpng.org/
- libusb-0.1.8, http://libusb.sourceforge.net/
- links-2.1pre14 (webbrowser), http://atrey.karlin.mff.cuni.cz/~clock/twibright/links/
- mc-4.6.0, http://www.ibiblio.org/mc/
- mtd-utils-20050217, http://www.linux-mtd.infradead.org/
- ppp-2.4.1, http://www.samba.org/ppp
- usbutils-0.70, http://usb.cs.tum.edu/

A few additional demo scripts to demonstrate Linux on Glencoe are in `/root/demo/` (linked to `/usr/local/bin/`)

5.5.2. A few system utilities

By default, a telnet and ssh daemon is started, allowing remote login through a network connection. The default root password is empty, but can be changed with `passwd root`.

- `flash-blob`, `flash-kernel`, `flash-rootfs`: downloads a file over http to ram and burns it to flash afterwards. This allows later system updates on-the-fly, which are quicker than through USB.
  Needs /`usr/local/bin/wgetfile` to work, so please configure this file before using!
- Suspend to RAM:
  ```
  # apm –suspend
  ```
  For more details on power management, see
  ftp://ftp.arm.linux.org.uk/pub/armlinux/people/xscale/mainstone/02-25-2005/README.txt
- change core speed: (patches and utilities are original Mainstone – for details, please see

ftp://ftp.arm.linux.org.uk/pub/armlinux/people/xscale/mainstone/02-25-2005/README.txt
`# dvfm 11 4 0 2 0` (285 Mhz)

- inspect and change PXA27x registers:
  `# pxa27xregs`
- synchronize clock via ntp server:
  `# rdate_sync_time_now`
- Without running system logger, get kernel messages continuously:
  `# tail -f /proc/kmsg`
- List files opened by cardmgr
  `# lsof | grep cardmgr`
- Where is <program>
  `# which <program>`

The current timezone is determined by the link /etc/localtime and can be configured as follows:

```
# rm /etc/localtime
# ln -s /usr/share/zoneinfo/America/New_York /etc/localtime
```

or any other timezone file in /usr/share/zoneinfo.

5.5.3. CF/PCMCIA hardware

Compact flash cards (e.g. Ethernet and Wireless LAN cards) are configured and loaded on insertion.

To list the inserted devices, run

```
[root@Linux /]#cardctl ident
```

Storage hardware like an IBM microdrive or Compactflash memory card is recognized. A fstab entry exists for the first partition and mounts /dev/hda1 to /mnt/cf upon manual mount like

```
[root@Linux /]#mount /mnt/cf
```

To shutdown CF/PCMCIA hardware without removing it (e.g. to save power), either run

```
[root@Linux /]#/etc/init.d/pcmcia stop
```

or use the utiliy `cardctl`.

5.5.4. USB Host

A few drivers have already been compiled in the kernel. Among them are a USB mouse driver, keyboard driver for an external USB keyboard, Bluetooth and memory card drivers for USB card readers.

USB drives are accessed through the `/dev/sdx[#]` devices.

An entry in /etc/fstab exists for a USB Memory Stick. To mount it, run

```
[root@Linux /]#mount /mnt/usb
```

5.5.5. USB gagdet support

For the USB client functionality, four USB gadget modules have been included:
g_ether, g_file_storag, g_serial and gadgetfs.

For information on how to use them, please look here:
http://www.linux-usb.org/gadget/

The following two examples have been taken from there.

Setting up an USB ethernet device:

```
bash# : this implicitly loads the usb controller driver, and helps
bash# : hosts use host_addr in config databases (win32 registry etc)
bash# modprobe g_ether host_addr=00:dc:c8:f7:75:05 dev_addr=00:dd:dc:eb:6d:f1
bash#
bash# : connect the device.
bash# : then use dhcp, zcip or static configuration:
bash# ifconfig usb0 192.168.1.101
bash# ip addr show usb0
2: usb0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:dd:dc:eb:6d:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.105/8 brd 10.255.255.255 scope global usb0
bash#
bash# : traffic will flow when the usb host is ready too
bash#
```

The linux kernel driver usbnet in linux-2.6.10 and higher has RNDIS/g_ether support. A windows driver is also available (http://www.davehylands.com/linux/gumstix/usbnet/linux.inf)

Setting up a USB serial connection

On Glencoe: (ignore warnings)

```
# modprobe g_serial
# picocom /dev/ttygserial
```

On Host (Gentoo Linux):

```
# modprobe usbserial vendor=0x0525 product=0xA4A6
# picocom /dev/usb/tts/0
```

5.5.6. Serial port

Each serial line can be controlled through stty:

```
# stty -F /dev/ttyS0 115200 crtscts cs8
# echo -n -e "atdt 4121234567;\n\r" > /dev/ttyS0
```

`ser2net` opens a local serial port and allows access through a telnet session over the network.

A sample configuraion, which opens `/dev/ttyS0` and a local port 2001 exists in `/etc/ser2net.conf`

On Glencoe:

```
# ser2net
```

On Remote host:

```
      $ telnet glencoe 2001
```

For **local serial port access**, picocom, a small terminal, is also installed:

```
      # picocom -b115200 /dev/ttyS0
```

The script `gemterm` is a shortcut for the above line

To exit picocom, press CTRL+A, CTRL+Q

To issue a voice call, enter the following string in picocom:

```
      atdt 4121234567;
```

A shortcut to **initiate a voice connection**:

```
      # gemdial 4121234567
```

5.5.7. Bluetooth

The bluetooth drivers are configured in /etc/default/bluetooth and /etc/bluetooth. Currently the Bluetooth pin is static and can be changed in /etc/bluetooth/pin. This allows scripted connections with Bluetooth devices witouth user interaction.

For detailed information on how to use the BlueZ bluetooth stack, have a look at http://www.bluez.org and http://www.hanscees.com/bluezhowto.html

Examples:

- List information about the Bluetooth module:

```
 # hciconfig -a
 hci0:   Type: USB
 BD Address: 00:08:F4:10:11:11 ACL MTU: 192:8  SCO MTU: 64:8
 UP RUNNING PSCAN ISCAN
 RX bytes:95 acl:0 sco:0 events:11 errors:0
         TX bytes:43 acl:0 sco:0 commands:11 errors:0
         Features: 0xff 0xff 0x0f 0x00 0x00 0x00 0x00 0x00
         Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
         Link policy:
         Link mode: SLAVE ACCEPT
         Name: 'localhost.localdomain-0'
         Class: 0x000000
         Service Classes: Unspecified
         Device Class: Miscellaneous,
         HCI Ver: 1.1 (0x1) HCI Rev: 0x110 LMP Ver: 1.1 (0x1) LMP Subver: 0x110
         Manufacturer: Cambridge Silicon Radio (10)
```

- Scan for devices:

```
# hcitool scan
Scanning ...
   00:05:16:5F:02:4C       smart_bt1_2
```

- Get information about device:

```
# hcitool info  00:05:16:5F:02:4C
Requesting information ...
        BD Address:  00:05:16:5F:02:4C
        Device Name: smart_bt1_2
```

```
          LMP Version: 1.2 (0x2) LMP Subversion: 0x570
          Manufacturer: Cambridge Silicon Radio (10)
          Features: 0xff 0xff 0x8f 0x78 0x18 0x18 0x00 0x80
                  <3-slot packets> <5-slot packets> <encryption> <slot offset>
                  <timing accuracy> <role switch> <hold mode> <sniff mode>
                  <park state> <RSSI> <channel quality> <SCO link> <HV2 packets>
                  <HV3 packets> <u-law log> <A-law log> <CVSD> <paging scheme>
                  <power control> <transparent SCO> <broadcast encrypt>
                  <enhanced iscan> <interlaced iscan> <interlaced pscan>
                  <inquiry with RSSI> <AFH cap. slave> <AFH class. Slave>
                  <AFH cap. master> <AFH class. master> <extended features>
```

- Ping bluetooth device:
```
# l2ping  00:05:16:5F:02:4C
Ping: 00:05:16:5F:02:4C from 00:08:F4:10:11:11 (data size 20) ...
44 bytes from 00:05:16:5F:02:4C id 0 time 27.19ms
44 bytes from 00:05:16:5F:02:4C id 1 time 31.83ms
44 bytes from 00:05:16:5F:02:4C id 2 time 27.94ms
3 sent, 3 received, 0% loss
```
- Open BT serial connection:
  # rfcomm connect 0 `00:05:16:5F:02:4C &`
  `# echo “atdt12345” > /dev/rfcomm0` or
  `# picocom /dev/rfcomm0`
- Close BT connection:
```
# fg
CTRL-C
```

5.5.8. Networking

Here are some helpful networking commands. Run them with `--help` option or look up in manpages/google for more detailed information on how to use them.

- Wireless tools: `iwconfig, iwpriv, iwspy, iwlist`
  `# iwconfig eth1 essid CMU` (set essid to CMU)
- Bridge between networks: `brctl`
- Firewall and NAT: `iptables, echo 1 > /proc/sys/net/ipv4/ip_forward`
  For a simple NAT configuration, see /usr/local/bin/iptables_nat_example
- download a file (HTTP, FTP): `wget`
- browse the internet:
  `# links <URL>`
  Press ESC for Menu
- Tiny webserver, serving root directory:
  `# httpd -h /`
  On host, open in browser: http://glencoe/root/img1.jpg
- Use dhcp to get an IP address and gateway:
  `# dhcpcd eth0`
- Show NIC ips:
  `# ifconfig | grep -E \(encap\)\|\(inet\ addr\) | awk 'BEGIN { FS=" "}`

```
        { print $1 " " $2 " " $3 }'
        # showip
```

or short

More information about Linux and Networking can be found in the HOWTOs on http://www.tldp.org/. Here are a few links:

http://www.tldp.org/HOWTO/BRIDGE-STP-HOWTO/index.html

http://www.tldp.org/HOWTO/DHCP/index.html

Quick pppd connection:

```
      # /usr/sbin/pppd /dev/ttyS0 115200 debug connect "/usr/sbin/chat -v   ''
ATD5555555  CONNECT  ''  ogin:  <yourusername>  assword:  <yourpassword>"
```

When the GEM module returns NO CARRIER, data calls are not activated for the current SIM card.

More information is in this document, which shows how to connect your Linux PC to a PPP server, how to use PPP to link two LANs together and provides one method of setting up your Linux computer as a PPP server. The document also provides help in debugging non-functional PPP connections: http://www.tldp.org/HOWTO/PPP-HOWTO/

5.5.9. Console, file system and multimedia utilities

The following are a few sample commands concerning console and multimedia.

All framebuffer programs like fbv or qpe should use an unused virtual terminal. So switch the console before you start these type of programs with:

```
      # chvt 3
```

Show all images in /root:

```
      [root@Linux /]# chvt 3
      [root@Linux /]# cd /root
      [root@Linux /root]# fbv *
```

Use space to switch, n to rotate

Switch from OSS to ALSA:

Edit `/etc/init.d/01rc.init`, comment out line with OSS and enable line with ALSA

```
      # mcedit /etc/init.d/01rc.init
```

Reboot afterwards

Show OSS capabilities:

```
      # dsp_info
```

Start OSS mixer:

```
      # aumix
```

Start ALSA mixer:

```
      # alsamixer
```

Use “,” to mute or unmute channels, cursor keys, numbers, ... to navigate and settings

Play an MP3:

```
      # madplay /mnt/card/MyMusic.mp3
```

`sox` can be used to manipulate wave files (format conversion, effects, ...)

Record audio file in ALSA: (remember to check the mixer settings in advance)

```
        # arecord -f cd sound.wav
        Press CTRL+C to end recording
```

Play audio file in ALSA:

```
        # aplay sound.wav
```

Calibrate touchscreen:

```
        # ts_calibrate
```

Test touchscreen:

```
        # ts_test
```

Load new console font:

```
        # gunzip -c /usr/share/consolefonts/default8x9.psfu.gz  | loadfont
```

Shortcut scripts:

```
        # loadfont-big
        # loadfont-small
```

Change cursor to red:

```
        # echo -e '\033[?17;0;64c' > /dev/tty0
```

or simply

```
        # redcursor
```

Change timeout for switching off LCD:

```
        # setconsoleblankingtime <minutes>
```

A. Appendix - Related HowTo's

This is some information which are indirectly related to BLOB, the Linux kernel or Linux in a broader sense.

A.1. Installing the arm toolchain for cross-compiling

Unpacking the pre-compiled PXA27x arm-linux toolchain from [1] along with the necessary shell variable exports is sufficient to compile code. However in some circumstances, some required libraries and includes cannot be found if they are not installed to `/usr/local/arm-linux`.

As workaround, create a symbolic link to the toolchain:

```
      ln -s /home/matthias/arm/arm-linux /usr/local/arm-linux
```

Also, be careful when switching toolchains as updating the root filesystem libraries may be necessary due to ABI changes. Unmatched libraries can e.g. cause problems with floating point.

But even with the corresponding system libraries, the wireless tools kept having problems with floating point. Another bug in the Mainstone toolchain was the missing create_module function, which breaks module_init_tools.

As there were problems with the Mainstone toolchains (both arm-linux-gcc-3.4.3 and arm-linux-gcc-3.3.2), a switch to the openembedded toolchain was made.

The toolchain can be built following the instructions here: http://openembedded.org/cgi-bin/moin.cgi/GettingStarted

This builds arm-linux-gcc-3.4.2 and was used for all steps in the process (blob, kernel and rootfs binaries)

A.2. Creating an environment for remote debugging

Build gdb for arm-linux:

To build the x86 host cross debugger:

```
Download gdb-5.3
./configure --host=i386-linux --target=arm-linux --build=i386-linux
make
```

Build kgdb enabled kernel.
Read Documentation/arm/kgdb.
Add "console=ttyKGDB" to boot command line.
Enable KGDB support and sub-options under Kernel hacking

Sequence:
ARM: boot kgdb kernel with console=ttyKGDB
x86: gdb-5.3/gdb/gdb vmlinux
x86: start debugging in gdb with "target remote /dev/ttyS0"

set breakpoints, single step, etc.

On PC:

```
# stty -F /dev/ttyS0 115200
# /src/gdb-5.3/gdb/gdb vmlinux
```

In GDB:

```
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
0xc0021ac8 in breakpoint () at kgdb-stub.c:1121
1121            BREAKPOINT();
warning: shared library handler failed to enable breakpoint
(gdb) cont
Continuing.
```

To debug applications on ARM, gdbserver can be used. Configure and compile gdbserver with the following command:

```
    ./configure arm-linux
```

A.3. HowTo setup USBDNET

This HowTo is mainly extracted from the Mainstone README file in [1].

A.4.1. Installing USB Ethernet on host

```
      # tar xzf usbdnet.tar.gz
      # cd usbdnet
      # make
      # rmmod usbnet
      # insmod usbdnet.o
```

NOTE: Remove kernel module "`usbnet.o`" before installing kernel module "`usbdnet.o`".

A.4.2. Configuring USB Ethernet

After resetting the Glencoe board, enter BLOB command line.

Now configure the USB Ethernet.

- Reset USB
- Execute "tftp" server: `$ /usr/sbin/in.tftpd -l -s /tftp/`
- connect the USB cable when the prompt displays.

```
      blob> tftp filename
              **** Plug-in USB cable & config usbdnet now *****"
```

- Configure USB Ethernet device on host

```
      $ ifconfig -a
```

You now find an Ethernet interface named "usb0". Configure it with

```
$ ifconfig usb0 192.168.1.100
```

NOTE: if eth1 is already configured as 192.168.1.100, please disable the eth1.

```
$ ifconfig eth1 down
```

BLOB has following default configuration

* IP address	: 192.168.1.101
* Default TFTP server	: 192.168.1.100

To change the configuration, use "ifconfig" on the BLOB command line. Note the new configuration is not permanent before BLOB as it does not have a flash area to store the configuration.

It helpful to create a new ethernet device config for usb0, so that the interface is started and set up automatically whenever the usb cable is plugged in. When tftp is running, the download starts automatically after a few seconds.

A usb hotplug script in /etc/hotplug/usb/usbdnet can also help to automate this procedure:

```
#! /bin/bash
echo "usbdnet hotplug script running"
typeset -i num
num=`ifconfig | grep usb0 | wc -l`
if [ $num -eq 0 ] ; then
echo "Loading usbdnet"
ifconfig usb0 192.168.1.100 netmask 255.255.255.255
fi
```

A.4.3. Using Kernel 2.6 on host side with `usbnet`

The `usbdnet` driver only works with the 2.4 kernel version. For newer kernels, the driver usbnet within the kernel source tree is intended to work with BLOB:
`linux-2.6.9/drivers/usb/net/usbnet.c`

A.5. Mounting MTD & JFFS2 images on a PC system

The jffs2 root filesystem can be mounted on a PC for preliminary modifications as follows:

```
modprobe mtdcore
modprobe jffs2
modprobe mtdram total_size=16384 erase_size=256
modprobe mtdchar
modprobe mtdblock
mknod /dev/mtdblock0 b 31 0
dd if=ramdisk of=/dev/mtdblock0
mount -t jffs2 /dev/mtdblock0 fs
```

After modifications, the image can be written back to a file:

```
umount fs
dd if=/dev/mtdblock0 of=ramdisk
```

B. Links of Interest

[1] Mainstone-based Linux kernel 2.6.9-intc1, with tools
http://ftp.linux.org.uk/pub/linux/arm/people/xscale/mainstone/01-27-2005

[2] ARM-Linux mailing list
http://lists.arm.linux.org.uk/

[3] Linux Boot Loader Object BLOB
http://www.lart.tudelft.nl/lartware/blob/

[4] Porting device drivers to the 2.6 kernel, with information about DMA, I/O, IRQ, ...
http://lwn.net/Articles/driver-porting/

[5] Free Electrons online Training and Articles about Embedded Linux
http://free-electrons.com/training/
http://free-electrons.com/articles/

[6] EmbeddedLinuxInterfacing Wiki pages with additional links
http://www.embeddedlinuxinterfacing.com/ewiki/index.php?id=WikiTopPage

[7] PTXdist – Userland Configuration Tool
http://www.pengutronix.de/software/ptxdist_en.html

[8] Cross-Referencing Linux (quick code browsing)
http://lxr.linux.no/ident?a=arm

[9] How to compile a linux kernel for ARM
http://www.arm.linux.org.uk/docs/kerncomp.php

[10] Toby Churchill's Balloon patchset
http://husaberg.toby-churchill.com/balloon

[11] Openembedded support forum
http://www.oesf.org/forums

[12] Setup OPIE
http://dev-bywater.media.mit.edu/wiki/borglab/OpieSetup

[13] Qtopia2.1
http://doc.trolltech.com/qtopia2.1/html/index.html

[14] Patch collection for ARM cross-compiling
http://www.hughes-family.org/~craig/trunk/sources/