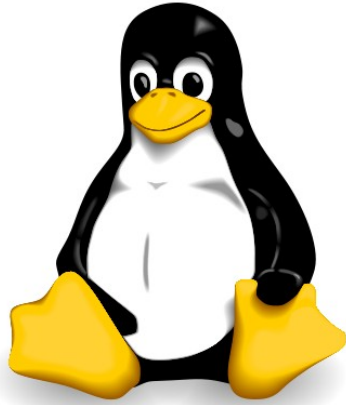


Toshiba / Topas910



Linux 2.6.26.5 – Stand alone package

1) Introduction

Linux is a fully open-source operating system which is on GPL v2 licence. This package contains a port of Linux 2.6.26.5 and a boot strap for [Toshiba Topas910](#).

The root file system can be chosen by the user. However, a pre-compiled and pre-package one, for Topas910, is available from [bplan GmbH](#).

The package is separated into two parts:

- Low level: A small boot code, stored at the beginning of the NOR flash, initializes the CPU and the board components. Then, it loads the user application from the NOR to the main memory and executes it.
- Linux Kernel: A fully stand alone port of Linux 2.6.26.5 to the Topas 910 board. It is stored into the NOR and loaded by the low level.

2) Contains and requirements.

Files list:

- patch_2.6.26.5_topas910.bz2: A patch to apply to the vanilla Linux kernel 2.6.26.5
- linux-2.6.26.5_topas910.tar.bz2: A "ready to use" Linux kernel 2.6.26.5 source tree including a valid configuration file
- config_linux_2.6.26.5_topas910: A valid Linux kernel configuration file
- lowlevel_topas910.tar.bz2: The low-level source code
- flash.bin.zip: A pre compiled flash image which contains the low-level boot code and a pre compiled Linux kernel

requirements:

In order to modify and build your own kernel, you first need a working GNU ARM tool chain. We highly recommend to use at least GCC 4.2 or higher and binutils 2.18 or higher.

A fully working development system (Linux installation with a pre-installed toolchain) is available for free, from bplan GmbH.

In this document, we will often assume the tool chain is named "arm-none-eabi-".

3) Quick-setup

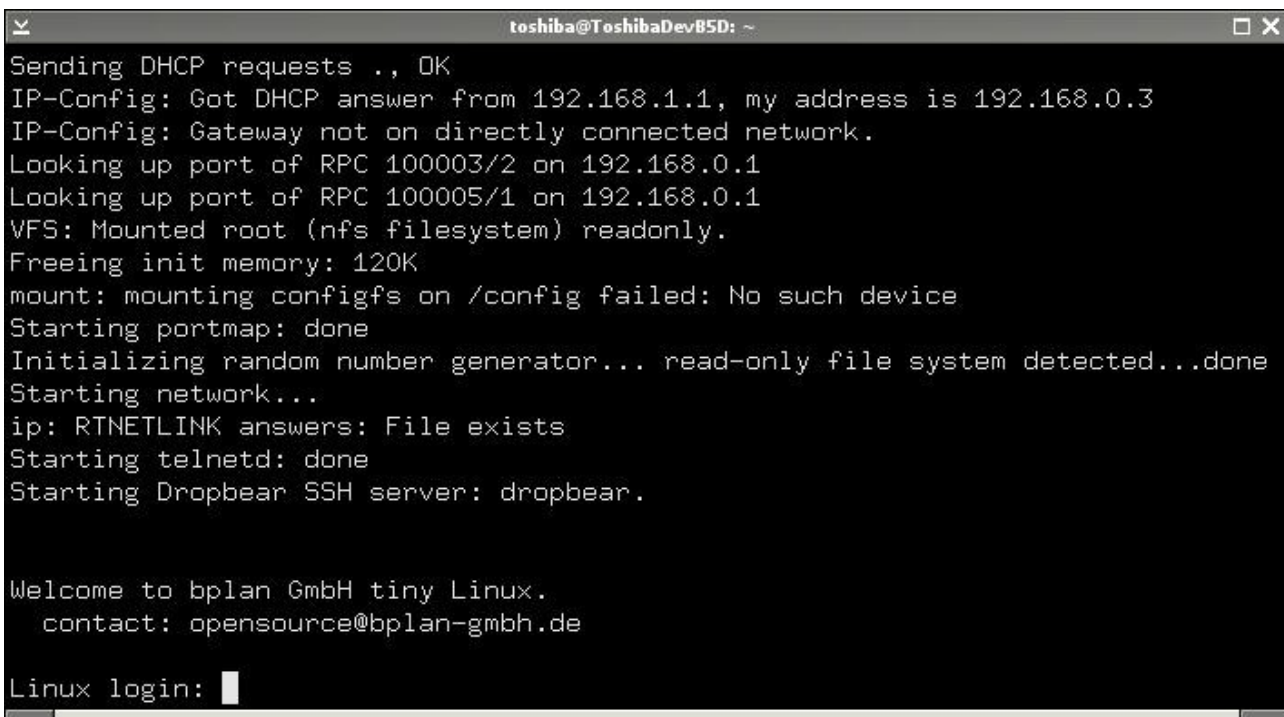
Burn the `flash.bin` image into the Topas 910 NOR flash using the JTAG recovery utility and issue a reset. Linux will start-up and try to mount an NFS root file system on `192.168.0.1:/srv/arm1`.



```
Select C:\WINNT\system32\cmd.exe - jlinkprogramtopas_a910 flash.bin
D:\JLinkDemo>jlinkprogramtopas_a910 flash.bin
SEGGER J-Link Toshiba TOPAS 910 boot loader restore.
Compiled May  2 2008 17:14:53
(c) 2008 SEGGER Microcontroller GmbH & Co. KG, www.segger.com
Solutions for real time microcontroller applications

Connecting...
JTAG speed: 8000 kHz
Core Id: 0x7926031
Reading CFI info...
Found CFI compliant flash device:
1 block region
256 sectors with 128 KB
CFI info read successfully.
Flash ID: 0x1227E
Loading data file...
Loading binary file... [flash.bin]
3145760 bytes of data loaded successfully.
Erasing Sectors...
Programming flash...
..... O.K. - Completed after 11.777 sec
.....
```

By the way, the Linux kernel is stored compressed into the Flash. During the decompression, the Topas910 band display will toggle. This is useful to prove that, at least, this process is in progress.



```
toshiba@ToshibaDev850: ~
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 192.168.1.1, my address is 192.168.0.3
IP-Config: Gateway not on directly connected network.
Looking up port of RPC 100003/2 on 192.168.0.1
Looking up port of RPC 100005/1 on 192.168.0.1
VFS: Mounted root (nfs filesystem) readonly.
Freeing init memory: 120K
mount: mounting configs on /config failed: No such device
Starting portmap: done
Initializing random number generator... read-only file system detected...done
Starting network...
ip: RTNETLINK answers: File exists
Starting telnetd: done
Starting Dropbear SSH server: dropbear.

Welcome to bplan GmbH tiny Linux.
  contact: opensource@bplan-gmbh.de

Linux login: █
```

4) The lowlevel

This is a small boot code which initialises the board components, loads an application from the NOR flash to memory and executes it.

You can change the application offset and size into the file "startmeup.S". Example with the default values:

```
#define APP_RAMLOCATION 0x41000000
#define APP_NORLOCATION 0x20040000
#define APP_SIZE      0x00200000
    NOR [0x20040000 - 0x00240000] -> Copy -> RAM 0x41000000

#define APP2_RAMLOCATION 0x43000000
#define APP2_NORLOCATION 0x20600000
#define APP2_SIZE      0x00300000
    NOR [0x20600000 - 0x20900000] -> Copy -> RAM 0x43000000
```

Once, the whole user application has been loaded, the boot code jumps to the first RAM location: APP_RAMLOCATION. In this case, a Linux kernel is copied at 0x41000000 and a Linux RAM Disk at 0x43000000. Then, the code jumps to 0x41000000 and Linux starts up.

The default toolchain is "arm-none-eabi-". If you have another one installed, edit the Makefile and change the toolchain option:



```
toshiba@ToshibaDevB5D: ~/src/linux/lowlevel
# Makefile
#
#####
CC = arm-none-eabi-gcc
LD = arm-none-eabi-ld
STRIP = arm-none-eabi-strip
OBJCOPY = arm-none-eabi-objcopy
ECHO = echo

TARGET = bootstart_tmpa910.bin
PLATFORM_PATH = ./platform/tmpa910

1,1 Top
```



Once you are done with your modification, type "make" to build the lowlevel. It will generate a file named "bootstart_tmpa910.bin" which has to be burn at offset 0x0 into the NOR Flash.

```
toshiba@ToshibaDevB5D: ~/src/linux/lowlevel
Assembling build/low_dm9000.o... Done
Assembling build/low_smc.o... Done
Assembling build/low_dmc.o... Done
Assembling build/low_clocks.o... Done
Assembling build/low_lcd.o... Done
Assembling build/low_misc.o... Done
Assembling build/minicon_common.o... Done
Assembling build/minicon.o... Done
Compiling build/low_sdhc.o... Done
Compiling build/low_misc_c.o... Done
Final Linking bootstart_tmpa910.prebin.elf.db... Done
Striping bootstart_tmpa910.prebin.elf... Done
Extracting bin bootstart_tmpa910.bin... Done
toshiba@ToshibaDevB5D:~/src/linux/lowlevel$
```

5) First build and boot option

Build:

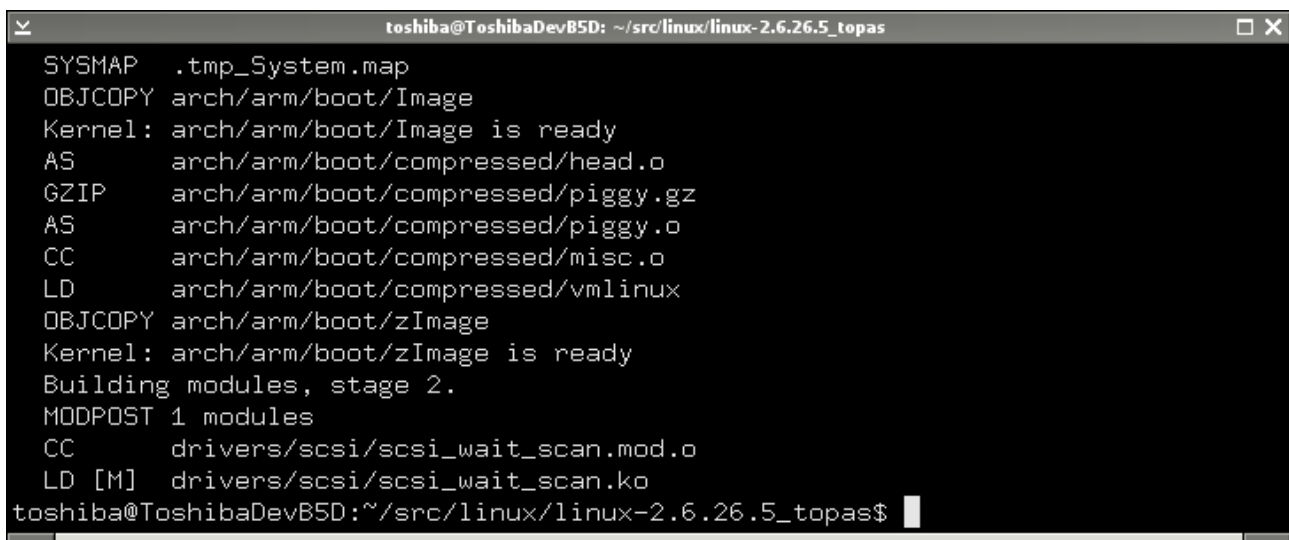
First, uncompress the tarball (linux-2.6.26.5_topas910.tar.bz2) somewhere. There is two options to give to the make command to build a valid ARM kernel:

- ARCH=arm: Force to compile for ARM. Otherwise, the build will be done for your own system.
- CROSS_COMPILE=arm-none-eabi-: Specify the tool chain to use. Here "arm-none-eabi-" is an example. You have to adapt it to your environment.

A valid command is:

```
> make ARCH=arm CROSS_COMPILE=arm-none-eabi-
```

After a while, the kernel is built. The file "arch/arm/boot/zImage" contains the compressed kernel and a small header to expand it. This file can be loaded into the TMPA910 memory and executed directly.



```
toshiba@ToshibaDevB5D: ~/src/linux/linux-2.6.26.5_topas
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy.gz
AS arch/arm/boot/compressed/piggy.o
CC arch/arm/boot/compressed/misc.o
LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
Building modules, stage 2.
MODPOST 1 modules
CC drivers/scsi/scsi_wait_scan.mod.o
LD [M] drivers/scsi/scsi_wait_scan.ko
toshiba@ToshibaDevB5D:~/src/linux/linux-2.6.26.5_topas$
```

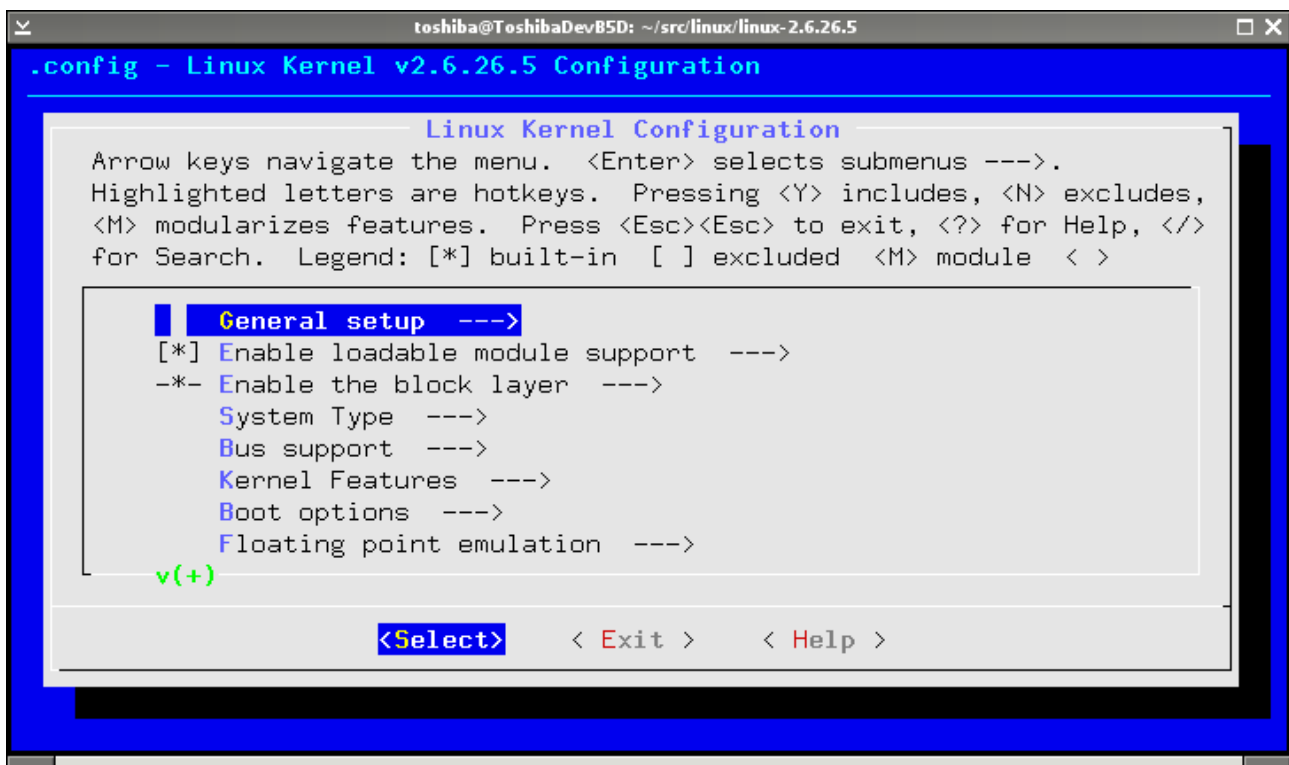
Kernel configuration:

A valid configuration is delivered with the Linux tarball. However, you will likely want to adapt the kernel options to your exact needs.

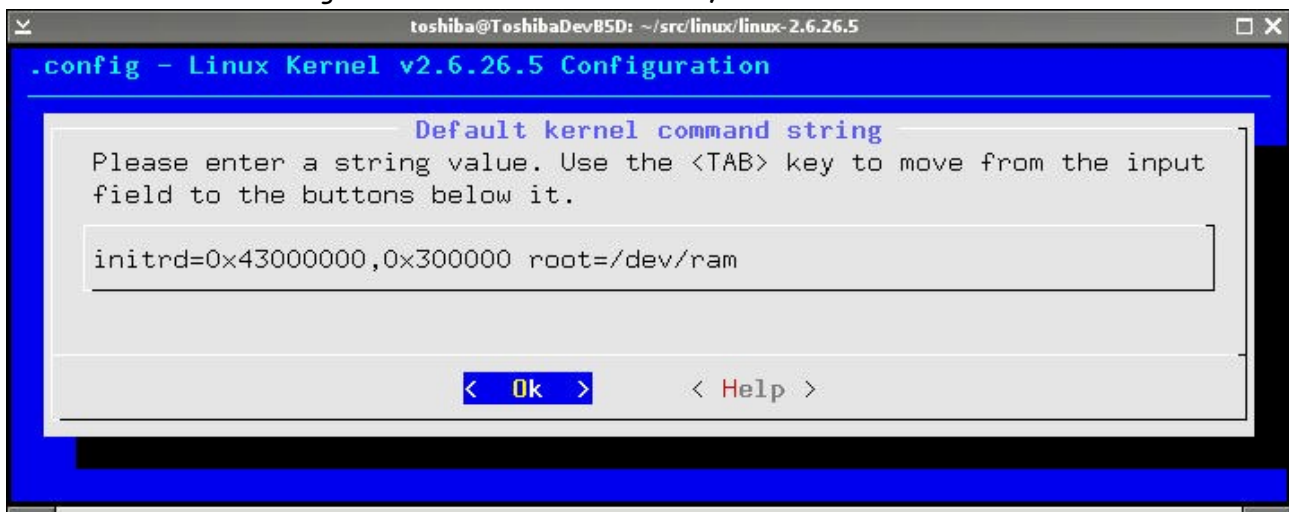
As an example, if you do not need the touch screen controller driver, you could disable the Linux input device system.

To change the Linux configuration enter the following command:

```
> make ARCH=arm CROSS_COMPILE=arm-none-eabi- menuconfig
```



You will likely want to change your boot option. For example, to use a RAM Disk image, located at 0x43000000 as root file system enter the following parameter into "Boot options -> Default kernel command string": "initrd=0x43000000,0x300000 root=/dev/ram"



6) Device support

- Interrupt controller: Provides interrupt to the Linux kernel
- Timer controller: The TMPA910 Timer 0 is used as time base for the Kernel
- Serial controller: Provides serial communication. This can be use for a serial console or to control external peripherals
- Davicom DM9000: Provides Ethernet network connection. It pops up as "eth0" into Linux.
- LCD Controller: Provides a 320x240 24bits frame buffer on the on- board display. This device is located at `/dev/fb0`, into the Linux file system
- Touch Screen: Provides user interaction. It is implemented as an Linux input device. It is at `/dev/input/event0`. We recommend to use the tslib to take full advantage of this device



7) FAQ

- I booted my system with the root file system provided on the bplan development environment. Telnet and SSH are working but I get no serial console login. Why ?

This root file system is configured to use the aura/hvc console. However, the standalone Linux use the serial port located at `"/dev/ttyS0"`. You have to edit the file `"/etc/inittab"` and replace `"hvc0"` by `"ttyS0"`.

- Linux reports only 63 MB of memory. However, there are 64 MB mounted on the board. Why is 1 MB missing?

We reserved the last mega bytes of memory [0x43f00000 - 0x44000000] for the frame buffer.

If you are not using the display and which to have access to the full RAM, disable the TMPA910 Frame buffer driver and edit `"include/asm/arm/arch/tmpa910/memory.h"`.

8) Links and resources

- Official Linux kernel:
<http://www.kernel.org>
- Toshiba TMPA910:
<http://www.toshiba-components.com/microcontroller/TMPA910.html>
- GNU development and toolchain:
<http://gcc.gnu.org/>
<http://www.gnu.org/software/binutils/>
- Official ARM info center:
<http://infocenter.arm.com/>



9) Contact

This software has been developed and is copyright bplan GmbH.

bplan Gesellschaft für Planung und Fertigung

elektrotechnischer Baugruppen mbH

Industriestraße 23

61449 Steinbach

Deutschland

Telefon: +49 (0) 6171/9187-0

Fax: +49 (0) 6171/9187-40

E-Mail: info@bplan-gmbh.de

Internet: www.bplan-gmbh.de



10) Licence

The Linux kernel is under GPLv2 licence:

<http://www.gnu.org/licenses/gpl-2.0.html>

The low level boot code is under BSD Licence:

Copyright (c) 2008, bplan GmbH

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the bplan GmbH nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.