THE LEARNING LABS, INC.

Building Linux for the TLL6219 ARM Processor

Mark McDermott
7/31/2008

Table of Contents

Introduction

This document describes how to build a new 2.6.16 Linux kernel for the TLL6219 ARM processor. NOTE: This procedure is intended to run on X86 based Linux machines so that other applications such as BusyBox can be compiled to run as a dynamically linked object. The Linux will compile under Cygwin but BusyBox will not.

Installing the Kernel source code "Tarball" on an X86 Linux system

If you install the full sources, put the kernel tarball in a directory where you have permissions (eg. your home directory) and unpack it:

```
gzip -cd linux-2.6.16.tar.gz | tar xvf -
```

Do NOT use the /usr/src/linux area! This area has a (usually incomplete) set of kernel headers that are used by the library header files. They should match the library, and not get messed up by whatever the kernel-du-jour happens to be.

You can also upgrade between 2.6.xx releases by patching. Patches are distributed in the traditional gzip and the new bzip2 format. To install by patching, get all the newer patch files, enter the top level directory of the kernel source (linux-2.6.xx) and execute:

```
gzip -cd ../patch-2.6.xx.gz | patch -p1
```

(repeat xx for all versions bigger than the version of your current source tree, _in_order_) and you should be ok. You may want to remove the backup files (xxx~ or xxx.orig), and make sure that there are no failed patches (xxx# or xxx.rej) Unlike patches for the 2.6.x kernels, patches for the 2.6.x.y kernels (also known as the -stable kernels) are not incremental but instead apply directly to the base 2.6.x kernel. Please read Documentation/applying-patches.txt for more information.

Alternatively, the script patch-kernel can be used to automate this process. It determines the current kernel version and applies any patches found.

```
linux/scripts/patch-kernel linux
```

The first argument in the command above is the location of the kernel source. Patches are applied from the current directory, but an alternative directory can be specified as the second argument.

If you are upgrading between releases using the stable series patches (for example, patch-2.6.xx.y), note that these "dot-releases" are not incremental and must be applied to the 2.6.xx base tree. For example, if your base kernel is 2.6.16 and you want to apply the 2.6.16.3 patch,

you do not and indeed must not first apply the 2.6.16.1 and 2.6.16.2 patches. Similarly, if you are running kernel version 2.6.16.2 and want to jump to 2.6.16.3, you must first reverse the 2.6.16.2 patch (that is, patch -R) _before_ applying the 2.6.16.3 patch.

Make sure you have no stale .o files and dependencies lying around:

```
cd linux
make mrproper
```

You should now have the sources correctly installed.

Software Requirements

Compiling and running the 2.6.xx kernels requires up-to-date versions of various software packages. Consult Documentation/Changes for the minimum version numbers required and how to get updates for these packages. Beware that using excessively old versions of these packages can cause indirect errors that are very difficult to track down, so don't assume that you can just update packages when obvious problems arise during build or operation.

For the TLL6219 we will be using the CodeSourcery 4.3.2 version of gcc. Refer to the CodeSourcery Installation manual for details.

BUILD directory for the kernel

When compiling the kernel all output files will by default be stored together with the kernel source code. Using the option "make O=output/dir" allow you to specify an alternate place for the output files (including .config).

Example:

kernel source code: */usr/src/linux-2.6.16*
build directory: */home/name/build/kernel*

To configure and build the kernel use:

```
cd /usr/src/linux-2.6.16
make O=/home/name/build/kernel menuconfig
make O=/home/name/build/kernel
sudo make O=/home/name/build/kernel modules_install install
```

Please note: If the 'O=output/dir' option is used then it must be used for all invocations of make.

Configuring the kernel

Do not skip this step even if you are only upgrading one minor version. New configuration options are added in each release, and odd problems will turn up if the configuration files are not set up as expected. If you want to carry your existing configuration to a new version with minimal work, use "make oldconfig", which willonly ask you for the answers to new questions.

Alternate configuration commands are:

"make menuconfig" Text based color menus, radiolists & dialogs.
"make xconfig" X windows (Qt) based configuration tool.
"make gconfig" X windows (Gtk) based configuration tool.
"make oldconfig" Default all questions based on the contents of your existing ./.config file.
"make silentoldconfig"
Like above, but avoids cluttering the screen with questions already answered.

NOTES on "make config":
- having unnecessary drivers will make the kernel bigger, and can under some circumstances lead to problems: probing for a nonexistent controller card may confuse your other controllers

- A kernel with math-emulation compiled in will still use the coprocessor if one is present: the math emulation will just never get used in that case. The kernel will be slightly larger, but will work on different machines regardless of whether they have a math coprocessor or not.

- the "kernel hacking" configuration details usually result in a bigger or slower kernel (or both), and can even make the kernel less stable by configuring some routines to actively try to break bad code to find kernel problems (kmalloc()). Thus you should probably answer 'n' to the questions for "development", "experimental", or "debugging" features.

Compiling the kernel

Make sure you have at least gcc 3.2 available. For more information, refer to Documentation/Changes.

Please note that you can still run a.out user programs with this kernel. Do a "make" to create a compressed kernel image. It is also possible to do "make install" if you have lilo installed to suit the kernel makefiles, but you may want to check your particular lilo setup first.

To do the actual install you have to be root, but none of the normal build should require that. Don't take the name of root in vain.

If you configured any of the parts of the kernel as `modules', you will also have to do

"make modules_install".

Keep a backup kernel handy in case something goes wrong. This is especially true for the development releases, since each new release contains new code which has not been debugged. Make sure you keep a backup of the modules corresponding to that kernel, as well. If you are installing a new kernel with the same version number as your working kernel, make a backup of your modules directory before you do a "make modules_install".

Alternatively, before compiling, use the kernel config option "LOCALVERSION" to append a unique suffix to the regular kernel version. LOCALVERSION can be set in the "General Setup" menu.

In order to boot your new kernel, you'll need to copy the kernel image (e.g. .../linux/arch/arm/boot/zImage after compilation) to the place where your regular bootable kernel is found. (Refer to TLL6219 JFFS2 build document.

.config File

The kernel needs to be specially configured with “preemption” enabled. The standard .config file needs to look have the following settings:

```
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.16
# Mon Jul 21 10:01:51 2008
#
CONFIG_ARM=y
CONFIG_MMU=y
CONFIG_RWSEM_GENERIC_SPINLOCK=y
CONFIG_GENERIC_CALIBRATE_DELAY=y
```

```
#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
CONFIG_BROKEN_ON_SMP=y
CONFIG_LOCK_KERNEL=y
CONFIG_INIT_ENV_ARG_LIMIT=32
#
# Loadable module support
#
CONFIG_MODULES=y
CONFIG_MODULE_UNLOAD=y
CONFIG_OBSOLETE_MODPARM=y
CONFIG_KMOD=y
#
# IO Schedulers
#
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y
CONFIG_DEFAULT_AS=y
#
# Processor Type
#
CONFIG_CPU_32=y
CONFIG_CPU_ARM926T=y
CONFIG_CPU_32v5=y
CONFIG_CPU_ABRT_EV5TJ=y
CONFIG_CPU_CACHE_VIVT=y
CONFIG_CPU_COPY_V4WB=y
CONFIG_CPU_TLB_V4WBI=y
#
CONFIG_FAMILY_IMX=y
CONFIG_MACH_TLL_MX21=y
CONFIG_ARCH_IMX21=y
#
# Kernel Features
#
CONFIG_PREEMPT=y
CONFIG_NO_IDLE_HZ=y
CONFIG_AEABI=y
CONFIG_OABI_COMPAT=y
CONFIG_SELECT_MEMORY_MODEL=y
CONFIG_FLATMEM_MANUAL=y
CONFIG_FLATMEM=y
CONFIG_FLAT_NODE_MEM_MAP=y
CONFIG_SPLIT_PTLOCK_CPUS=4096
CONFIG_ALIGNMENT_TRAP=y
```