SW Tools Solutions for TI ARM(+DSP)

Linux by Degrees
(TMDSOSKL137)

Part I

(Using DaVinci_PSP_03.20.00.14 Community Linux on DA830/OMAP-L137/AM1808 and CCSv4.x/CCSv5.x)

July 19, 2011

Joe George
(with material stolen from all over)
Agenda

Operating Systems – Tools vs. Target

- DaVinci Tool Summary
- Development System Configurations
  - Standard DaVinci Linux (2 PC’s -> 1 WinXP PC + 1 VMWare Linux PC) from Bootcamp
  - Joe DaVinci WinXP (1 PC -> WinXP + VMWare Linux on 1 PC) – uses PumpKIN TFTP and NFSAxe Servers in WinXP
  - Minimal WinXP (1 PC -> WinXP Only)
  - Joe Optimum DaVinci (2 PC’s -> 1 WinXP PC + 1 WinxXP PC with VMWare Linux-Optional)
- New Joe Optimum DaVinci System Configuration (2 PC’s)
- New Joe Minimal DaVinci System Configuration (1 PC)

- TI SW Packages (PSP, SDK, etc.)
- Community Linux Installation Linux and WinXP
- OMAP-L137 EVM Easy Standalone Boot (USB Stick for ulimage and RAMDISK Filesystem)
- Tool chain Installation
- Build and Run “Hello World”
- Build and Run ulimage kernel
- Build and Run “GPIO” in Linux
- Building in VMWare Linux while booting Linux and Debugging (Linux Aware) with CCSv4/CCSv5
- Debug ARM9 and DSP simultaneously with CCSv4/CCSv5 Linux Aware
- Standalone EVM Block Diagram (USB Stick for ulimage and SD/MMC Card Filesystem)
- Format SD/MMC Card for Target Filesystem
- Other
- Various Hints
- Montavista Archive
- Marketing Backup
If Target development tools run under Windows, why use Linux as Tools O/S? But CCSv4/CCSv5 Stop-mode Linux kernel debug can run on Linux or WinXP.
<table>
<thead>
<tr>
<th>Tools O/S</th>
<th>ARM Target O/S</th>
<th>DSP Target O/S</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC running development tools</strong></td>
<td><strong>Target H/W (OMAP-L1 ARM+DSP)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Traditional TI DSP</strong></td>
<td>Windows (CCS)</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIOS (Linux - Virtuallogix)</td>
</tr>
<tr>
<td><strong>DaVinci ARM/Linux (Catalog ARM for Video)</strong></td>
<td>Linux (Red Hat 3 or 4, Suse) (and WinXP, too)</td>
<td>Linux (MontaVista?)</td>
</tr>
<tr>
<td></td>
<td>xDM/Codec Engine (DSP/BIOS)</td>
<td></td>
</tr>
<tr>
<td><strong>OMAP-L1 ARM+DSP (Catalog ARM+DSP)</strong></td>
<td>Linux (Ubuntu 9.04/10.04) and WinXP (CCS?)</td>
<td>Linux (Community) and SYS/BIOS on ARM?</td>
</tr>
</tbody>
</table>
Joe has been looking at most “cost effective” and traditional dsp options, some GUI and mostly WinXP-based. But lately using VMWare Linux images and CCSv4/CCSv5 has been most efficient.
Standard DaVinci System Configuration (from bootcamp)

- **Windows PC**
  - Ethernet port
- **Linux “Tools” PC (VMware)**
- **OMAP-L137 EVM**
  - eth0:
- **TeraTerm/ HyperTerminal**
- **RS-232**
- **Samba**

- **RS-232 is physical connection to U-Boot**
- **Use to interrupt DVEVM standalone boot**
- **Configure U-Boot modes by setting/saving environment variables**

**Ethernet provides physical connection for booting**
**U-Boot loads Kernel from “Tools” PC into DDR2 memory using TFTP**
**Filesystem is accessed via NFS protocol**

After Make
No need to download to target
• Got uImage kernel to boot off WinXP using Pumpkin TFTP server
• Got NFSAxe to act as NFS Server on WinXP
• Got MV-Pro to build uImage kernel in WinXP in Bash and DevRocket (with some path modification)
• Used DevRocket Debug on WinXP and GDBserver (with –attach PID) on DVEVM to build/debug decode (AAC)
• Got ARM and DSP open in CCS (though SD drivers seems more stable then BH) with Linux running.

If LAN then got Corp network access but some delay and certain local resources (i.e. DevRocket Target filesystem might not work)
If WAN/Internet then local router with MAC address filtering isolates from Corp network, but tough to use Outlook, SNAP servers, and other network resources (turn off DHCP, though turning back on will require a Router hard reset).
Minimal WinXP DaVinci System Configuration

- Makes you wonder if you could get rid of the Linux Tools completely for the Catalog
  ARM+DSP programmer

For Bash hello - p. 4-10 of SPRUE66.pdf
PATH="/opt/montavista/pro/devkit/arm/v5t_le/bin:/opt/montavista/pro/bin:/opt/montavista/common/bin:$PATH"
cd /home/a0321791/workdir/filesys/opt/hello
arm_v5t_le-gcc hello.c -o hello
./hello

For Bash kernel uImage - p. 4-11 of SPRUE66.pdf
PATH="/opt/montavista/pro/devkit/arm/v5t_le/bin:/opt/montavista/pro/bin:/opt/montavista/common/bin:$PATH"
AND
PATH="/opt/mv_pro_4.0/montavista/pro/devkit/arm/v5t_le/bin:/opt/mv_pro_4.0/montavista/pro/bin:/opt/mv_pro_4.0/montavista/common/bin:$PATH"
cd /home/a0321791/workdir/lsp/ti-davinci
make ARCH=arm CROSS_COMPILE=arm_v5t_le- uImage
Joe Optimum DaVinci System Configuration (2 PC’s)

- Local router with MAC address filtering isolates ONLY “Linux Box” and EVM from TI network
- WinXP PC (regular laptop) on TI network AND can access DVEVM with RS-232 terminal (and uBoot)
- WinXP PC can be (hibernated and) removed daily without messing up the local system/IP address
- Got uImage kernel to boot off WinXP TFTP server (PumpKIN) with uBoot from WinXP PC
- Got MV-Pro to build uImage kernel in WinXP in Bash and DevRocket (some path modification)
- Used DevRocket on WinXP and GDBserver on DVEVM to build/debug decode (AAC) demo program
- VMware Linux optional for testing
Getting to like using VMWare in Linux most of the time since you can do a standalone boot with USB stick (FAT32) having ulimage and a RAMDISK

Avoids the need for making an SD/MMC card (EXT2//3) which is useful after you are up and running (and default for Android, Qt, various SDK’s etc.)

But I wonder if we can get more packages running with USB Stick and RAMDISK

NFS/TFTP good for work in your cube/lab but not easily portable
• Makes you wonder if you could get rid of the Linux Tools completely for the Catalog ARM+DSP programmer

For Bash hello - p. 4-10 of SPRUE66.pdf
PATH="/opt/montavista/pro/devkit/arm/v5t_le/bin:/opt/montavista/pro/bin:/opt/montavista/common/bin:$PATH"
cd /home/a0321791/workdir/filesys/opt/hello
arm_v5t_le-gcc hello.c -o hello
./hello

For Bash kernel ulmage - p. 4-11 of SPRUE66.pdf
PATH="/opt/montavista/pro/devkit/arm/v5t_le/bin:/opt/montavista/pro/bin:/opt/montavista/common/bin:$PATH"
AND
PATH="/opt/mv_pro_4.0/montavista/pro/devkit/arm/v5t_le/bin:/opt/mv_pro_4.0/montavista/pro/bin:/opt/mv_pro_4.0/montavista/common/bin:$PATH"
cd /home/a0321791/workdir/lsp/ti-davinci
make ARCH=arm CROSS_COMPILE=arm_v5t_le- uImage
Most Key SW ARM only, ARM+DSP, DSP Only starts here (need to register TI&ME):


<table>
<thead>
<tr>
<th>Target Content Infrastructure Product Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BIOS Platform Support Packages</strong></td>
</tr>
<tr>
<td><strong>DSP/BIOS and 375/BIOS</strong></td>
</tr>
<tr>
<td><strong>DSP/BIOS BIEUSB Product</strong></td>
</tr>
<tr>
<td><strong>DSP/BIOS Utilize Product</strong></td>
</tr>
<tr>
<td><strong>Digital Video Software Development Kits (DVSDK)</strong></td>
</tr>
<tr>
<td><strong>DSP Link and Syslink</strong></td>
</tr>
<tr>
<td><strong>Graphics SDK</strong></td>
</tr>
<tr>
<td><strong>EDMA3 Low-level Driver</strong></td>
</tr>
<tr>
<td><strong>Interprocessor Communication (IPC)</strong></td>
</tr>
<tr>
<td><strong>Linux Platform Support Packages</strong></td>
</tr>
<tr>
<td><strong>Multimedia Framework Products (MFP)</strong></td>
</tr>
<tr>
<td><strong>Network Developer's Kit (SDK)</strong></td>
</tr>
<tr>
<td><strong>RTOS / XDC tools</strong></td>
</tr>
<tr>
<td><strong>Software Development Kits (SDKx)</strong></td>
</tr>
</tbody>
</table>

- **DSPBIOS/SYSBIOS PSP's and other packages**
- **Combined SDK Nirvana (OMAP-L138 and above)**
- **Linux PSP**
# Software Development Kits (official from TTO)

<table>
<thead>
<tr>
<th>S/W Dev’l Kit</th>
<th>Description</th>
<th>Processor(s)</th>
</tr>
</thead>
</table>
| Linux PSP SDK          | Small Linux Distro supporting TI ARM devices                                | ✓ OMAP35, AM35, AM18  
|                        |                                                                            | ✓ OMAP-L1                                         |
|                        |                                                                            | ✓ DM644x, DM6467, DM3xx                          |                                                    |
| “DVSDK”                | TI provided libraries, examples, demos                                        | ✓ All TI SOC’s: ARM, DSP, ARM+DSP                  |
|                        | Codec Engine (VISA), DSPlink, Codecs/Algos (XDM), BIOS, XDC, Linux utilities, etc. | ✓ Obviously, not all devices require all the s/w components |                                                    |
| Code Gen Tools         | **Linux GNU Compiler** (CodeSourcery)                                        | All TI ARM and DSP devices where appropriate       |
|                        | **C6000 DSP Compiler** (TI)                                                  |                                                   |
| Graphics SDK           | Graphix SVSGX development kit                                                | ✓ OMAP3515, OMAP3530  
|                        | OPENGL ES / VG demos, drivers, targetfs, Getting Started Guide              | ✓ AM3517                                         |

- PSP is a TI specific acronym that represents the name of the group inside of Texas Instruments which “owns” the kernel and driver development activities: Platform Support Package team

- Wireless SDK is available independently of these other kits to support the TI WLxxxx Bluetooth/WiFi devices
TI SW Offerings (Unofficial for all TI processors)

- Stellarisware = driverlib + usblib + graphicslib + 3rd party stuff
  - driverlib ~= CSL (old DSP Chip Support Library which now some EP has registerCSL)
  - 3rd party stuff = lwip (Ethernet) + CAN stacks + fatfs (FAT32) + FreeRTOS? + etc.

- Sitaraware hopefully ~= Stellarisware (but more challenging with more CPU's, ARM9, Cortex-A8, etc. and more serious peripherals)

- PSP = DSP/BIOS (mostly BIOS5) + IOM drivers (higher end like EDMALLD, McASP, McBSP, codec but also low end like SPI, IIC, UART, GPIO) + rCSL (some EP) + RTFS (FAT32, some EP) + BIOSUSB (MSC, CDC, HID, some EP) + etc.

- LSP = Linux + PSP
  - Linux = drivers + menuconfig + kernel + everything including the kitchen sink (don't fight it. Just accept it).
  - Note that LSP Primus/Freon often has PSP. But for OMAP3 you mostly don't get PSP.

- SYS/BIOS6 = kernel (for MSP430 all the way up to Cortex-A8, but in general NO PSP, drivers, etc.) PSP for DSP/BIOS6 Primus/Freon has higher and lower end IOM drivers PSP for SYS/BIOS6 for ARM9 has only lower end IOM drivers (SPI, IIC, GPIO, etc.)

- RTSC (Real-Time Software Components) – TI-SB solution for make. Originally for Codec Engine but also used for SYS/BIOS6, GRACE, PSP, C6Flo, etc

- O/S = kernel + device drivers + (+ build utilities?)

- O/S is resource management, security/robust, inertia...
Community Linux Install – Linux and WinXP

This is what you get with your EVM:

For CL Install – Linux box or VMWare –
(External from http://software-dl.ti.com/dsps/dsps_public_sw/psp/LinuxPSP/DaVinci_03_20/03_20_00_14/index_FDS.html )

But you also need Target Filesystem:

And Arm cross development environment/cross compiler:

Much of the details in the next few slides come from:
http://processors.wiki.ti.com/index.php/Training#OMAP.E2.84.A2.2FDaVinci.E2.84.A2.2FSitara.E2.84.A2.2FIntegra.E2.84.A2_System Integration using Linux Workshop
and
## Community Linux Install – Linux and WinXP

DaVinci-PSP-SDK-03.20.00.14.gz (Linux Support Package which builds uImage)

Nice summary of the files in the package is here:

http://processors.wiki.ti.com/index.php/Community_Linux_PSP_for_OMAP-L1#PSP_Overview

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>..</code></td>
<td>DaVinci-PSP-SDK-#.###.###</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td>`</td>
<td>--`</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
</tbody>
</table>

Files in red will be used in the next few slides
Easy Standalone EVM Block Diagram (USB Stick for uImage and RAMDISK Filesystem)

Switch on power

Boot uImage and RAMDISK from FAT32 USB stick (USB0 microUSB connector)

Figure 1-1, Block Diagram OMAP-L137 EVM

Use Teraterm or Hyperterminal to talk to uboot
(115k*-N-1 – >No flow control)

http://support.spectrumdigital.com/boards/evmomapl137/revd
Boot switches

- Select boot switches from:

Booting from SPI Flash
Set the SW2 switch on the DSK board as follows. (X indicates the setting is 'don't care')

<table>
<thead>
<tr>
<th>Pin#</th>
<th>7 2 1 0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>0 1 0 1 X</td>
</tr>
</tbody>
</table>

Setup the EVM in "emulation debug" mode by setting SW2 switch as follows:

For revisions A and B:

<table>
<thead>
<tr>
<th>Pin #</th>
<th>7 2 1 0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>1 1 1 1 1</td>
</tr>
</tbody>
</table>

For all revisions after B:

<table>
<thead>
<tr>
<th>Pin #</th>
<th>7 2 1 0 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>1 1 1 1 0</td>
</tr>
</tbody>
</table>
Booting Linux – ROM to Kernel

1. RBL
   ROM

2. UBL
   (dsp-spi-ais.bin
    ubl-spi.bin)
   Internal RAM

3. U-Boot
   DDR2/SDRAM

   Kernel
   DDR2/SDRAM

Device

SPI) Flash

UBL (dsp-spi-ais.bin
ubl-spi.bin)

U-Boot
Linux
Kernel

DDR2/SDRAM

U-Boot
Linux
Kernel

UBL (dsp-spi-ais.bin
ubl-spi.bin)
Linux Boot Process

Power On

- ARM assembly code
- Passes args to Linux (bootargs)
- Boot from USB stick
- Can even turn on DSP

Boot Loader

U-Boot

Linux Kernel

Linux Kernel

Initrd (optional)

Initialize hardware

Initial RAM Disk *(we use this when no SD/MMC card)*

System

/sbin/init – 1st process exe by kernel

Login console

Login Prompt

Looking at doing what we say ...
Easy Standalone EVM (USB Stick) – Boot

- How do you Flash uboot into the EVM?: Turn on power and in Teraterm check for USB boot:

  • For the Code Composer way (uses spiflash_writer.out), Goto: http://processors.wiki.ti.com/index.php/Restoring/Flashing_OMAP-L137_Bootloaders#Flashing_the_EVM

  • For the UART/teraterm way (uses uFlash/ sfh_OMAP-L137.exe) go to: http://processors.wiki.ti.com/index.php/Serial_Boot_and_Flash_Loading_Utility_for_OMAP-L137#Serial_Flasher_Options

  In both cases you need:
  - dsp-spi-ais.bin – DSP Bootloader image
  - ubl-spi.bin – ARM9 bootloader image
  - u-boot.bin - uboot image

- Copy to USB Stick root directory (FAT32):
  • Type:
    - uImage – kernel
    - ramdisk-base.gz – RAMDISK
    - Hello
    - gpio_test.ko

- Turn on power and in Teraterm check for USB boot:
  • Type:
    - u-boot $ help
    - U-Boot > help usb
    - usb reset - reset (rescan) USB controller
    - usb stop [f] - stop USB [f]=force stop
    - usb tree - show USB device tree
    - usb info [dev] - show available USB devices
    - usb storage - show details of USB storage devices
    - usb dev [dev] - show or set current USB storage device
    - usb part [dev] - print partition table of one or all USB storage devices
    - usb read addr blk# cnt - read `cnt' blocks starting at block `blk#'
      to memory address `addr'
    - U-Boot > help usb
Easy Standalone EVM (USB Stick) – Boot

• Boot in Teraterm:
  • Type:
    u-boot $ setenv bootargs console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M mem=32M ip=192.168.1.111:192.168.1.1:255.255.255.0::off; usb reset;fatload usb 0:1 0xC0700000 uImage;fatload usb 0:1 0xc1180000 ramdisk-base.gz;bootm

• Login with root:
  • Type:

The Angstrom Distribution arago ttyS2

Arago 2009.03 arago ttyS2

arago login: root
root@arago:~# cd /
root@arago:/# ls
bin    etc    linuxrc    mnt    sys    var
boot   home   lost+found proc   tmp
dev    lib     media   sbin    usr
root@arago:/#

The output should be: Hello World!
VMWare Linux – Install Code Gen Tools

- Install Tools chain (in VMWare Ubuntu 9.04):
  - Type:
    host $ sudo dpkg-reconfigure -plow dash
    host $./arm-2009q1-203-arm-none-linux-gnueabi.bin

- Add paths in Terminal shell by:
  - Type:
    host $ export PATH="/root/CodeSourcery/Sourcery_G++_Lite/bin:$PATH"
    host $ source ~/.bashrc
  - Test Path by typing:
    host $ printenv
    PATH="/root/CodeSourcery/Sourcery_G++ Lite/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/X11R6/bin"
VMWare Linux – Build “Hello World” (Steps)

• Choose your directory:
  • Type:
    host $ cd /home/root1/joetemp/OMAPL137/Hello

• Create a file called main_OMAPL137.c with the following contents:
  • Type:
    ```
    #include <stdio.h>
    int main() {
    printf("Buongiorno DaVinci! - Built by VMWare Ubuntu 9.04\n");
    return 0;
    }
    ```

• Build:
  • Type:
    host $ arm-none-linux-gnueabi-gcc main_OMAPL137.c -o hello

• Perform the following steps on the target board. You may use either the target's console window or a telnet session.
  • Type:
    target $ cd /opt/hello

• Run Hello:
  • Type:
    target $ ./hello

  The output should be: Hello World (in Italian)!
Edit .c file

```c
#include <stdio.h>

int main() {
    printf("Buongiorno DaVinci! - Built by VMWare Ubuntu 9.04\n");
    return 0;
}
```

Build on host $`

```bash
chmod +x main_OMAPL137.c
./main_OMAPL137
```

Run on target $`

```bash
./main_OMAPL137
```
VMWare Linux – Build uImage kernel

Untar LSP:
Type:
host $ tar zxvf DaVinci-PSP-SDK-03.20.00.14.gz
host $ cd /home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/kernel
host $ tar zxvf linux-03.20.00.14.tar.gz
host $ cd /home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/kernel/linux-03.20.00.14

Make default configuration for uImage (comes from):
Type:
host $ make da830 omapl137_defconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf

Build uImage by:
Type:
host $ make uImage ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf
(If you get a mkimage error, cp mkimage to /usr/local/bin for uImage)

Get uImage from:
Type:
target $ cd
/home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/kernel/arch/arm/boot

Ubuntu_904_50Gb
VMware Player
CD-ROM (IDE 1:0)
CD-ROM (IDE 2:0)
Microsoft Windows XP Pro SP3
Ethernet Sound Adapter

To direct output to this virtual machine, press Ctrl-G.
- Invoke Menuconfig with make:
  - Type:
    ```
    make menuconfig ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
    ```

In Ubuntu 9.04 You might need:
(Terminal Font 6+?) and libncurses5-dev in Synaptic package manager

Check here for various driver options:

http://processors.wiki.ti.com/index.php/GSG:_Building_Software_Components_for_OMAP-L1/AM1x#Driver_configuration_in_the_Linux_kernel
VMWare Linux – Build GPIO

• Choose your directory:
  • Type:
    
    host $ cd /home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/examples

• Untar Examples:
  • Type:
    
    host $ tar zxvf examples.tar.gz
cd /home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/examples/examples-03.20.00.14/gpio

• Modify gpio_test.c:
  • Add:
    
    //jg 3/28/11
    int cpu_is_davinci_da830j=1;
  • Edit (Search and replace):
    
    //jg 3/28/11
    // if(cpu_is_davinci_da830()) {
    if(cpu_is_davinci_da830j) {

• Build:
    
    host $ make -C /home/root1/joetemp/OMAPL137/DaVinci-PSP-SDK-03.20.00.14/src/kernel/linux-03.20.00.14 M=`pwd` ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-

• Perform the following steps on the target board. You may use either the target's console window or a telnet session.
  • Type:
    
    target $ cd /media/sda1/OMAPL137/gpio
target $ insmod gpio_test.ko

• Flip (Boot) SW2-Pin 7:
  • See:
    
    target $ Testing gpio 87 (connected to boot pin S2-7)
The current state of S2-7 pin is OFF.
    Waiting for the pin to be on..
    .. Done OMAP-L137
target $
• Flip (Boot) SW2-Pin 7:
  • See:

  root@arago:~# cd /media/sda1/OMAPL137/gpio
  root@arago:/media/sda1/OMAPL137/gpio# insmod gpio_test.ko

  Testing gpio OMAP-L137 87 (connected to boot pin S2-7)
  The current state of S2-7 pin is OFF.
  Waiting for the pin to be on..
  .. done OMAP-L137
  root@arago:/media/sda1/OMAPL137/gpio#

  The cursor is now available

VMware Linux – Build GPIO
(Screenshot)