Linux-c6x-2.0-beta2 User Guide

Introduction

This document will guide you through initial setup of the EVM, updating the board with new bootloader and Linux software, and will walk you through initial use of the software. This document has been validated with the 2.0 Beta2 release of Linux-c6x / Linux-MCSDK.

The release notes are available from linux-c6x.org Files [1]

Getting Started

The goal of this section is to help the user run Linux-c6x kernel on the EVMS.

On a Windows host, install [tera term [2]] serial port program.

On a Linux host, install and configure the minicom serial port program.

You will also need to install the version of CCS recommended in the release notes.

Finally you will need to download the binaries for your selected boards and endian mode. Little endian is highly recommended.

The binaries are available from linux-c6x.org file area [3].

Hardware setup for c64x+ EVMs

Please pick up the c64x binary tarballs from [linux-c6x File download Area [3]]. There are separate tarballs for little and big endian. Untar the appropriate tarball.

ENDIAN Switches

Device	Switch	Pin	Little Endian	Big Endian
C6474	SW5	5	OFF	ON
C6472	SW1	8	ON	OFF
DSK6455	SW3	1	OFF	ON
C6474L	SW4	5	ON	OFF
C6457	SW3	8	ON	OFF

Boot Mode Settings

C6457

- For both Big and Little Endian Boot, SW4 pins 1-4 should represent master I2C Boot Mode.
- SW4 pins 1-4 should be 1000.
- The c6457 EVM has a NAND and the bootloader supports both a TFTP boot mode and a NAND boot mode
- SW3 pin 1 switches between TFTP boot mode and NAND boot mode.
  - For TFTP boot mode SW3 pin 1 should be OFF
  - For NAND boot mode SW3 pin 1 should be ON
- These settings are depicted below

C6457	Switch	Pin 1	Pin 2	Pin 3	Pin 4
Master I2C Boot Mode	SW4	ON	OFF	OFF	OFF
No Boot Mode	SW4	OFF	OFF	OFF	OFF
TFTP Boot	SW3	OFF			
NAND Boot	SW3	ON			

C6474L

- For both Big and Little Endian Boot, SW4 pins 1-4 should represent master I2C Boot Mode.
- SW4 pins 1-4 should be 1000.
- The C6474L EVM has a UART port and J13 jumper 3-5, 4-6 are connected
- The c6474l EVM has a NAND and the bootloader supports both a TFTP boot mode and a NAND boot mode
- SW3 pin 1 switches between TFTP boot mode and NAND boot mode.
  - For TFTP boot mode SW3 pin 1 should be OFF
  - For NAND boot mode SW3 pin 1 should be ON
- These settings are depicted below

C6474L	Switch	Pin 1	Pin 2	Pin 3	Pin 4
Master I2C Boot Mode	SW4	ON	OFF	OFF	OFF
No Boot Mode	SW4	OFF	OFF	OFF	OFF
TFTP Boot	SW3	OFF			
NAND Boot	SW3	ON			

C6472

- For Little and Big Endian Boot, SW2[1-4] should be 0100, i.e. SW2 should be in master I2C Boot Mode
- The C6472 EVM has a UART port and J10 jumper 3-5, 4-6 are connected
- The c6472 EVM has a NAND and the bootloader supports both a TFTP boot mode and a NAND boot mode
- SW1 pin 1 switches between TFTP boot mode and NAND boot mode.
  - For TFTP boot mode SW1 pin 1 should be OFF
  - For NAND boot mode SW1 pin 1 should be ON
- These settings are depicted below

C6472	Switch	Pin 1	Pin 2	Pin 3	Pin 4
Master I2C Boot Mode	SW2	OFF	ON	OFF	OFF
No Boot Mode	SW2	OFF	OFF	OFF	OFF
TFTP Boot	SW1	OFF			
NAND Boot	SW1	ON			

C6474

- For both Big and Little Endian Boot, SW5 pins 1-4 should represent master I2C Boot Mode.
- SW5 pins 1-4 should be 1000.
- There seems to be a discrepancy between the docs and the actual switch on the EVM. So to boot of the EEPROM, the SW5 pins 1-4 should be 0111
- The C6474 supports only TFTP boot
- These settings are depicted below

C6474	Switch	Pin 1	Pin 2	Pin 3	Pin 4
Master I2C Boot Mode	SW5	OFF	ON	ON	ON
No Boot Mode	SW5	ON	ON	ON	ON

DSK6455

- For both Big and Little Endian boot SW3 pin 2 and 4 should be 1, and SW3 pin 5 must be 0.
- The DSK6455 Supports only TFTP Boot
- To initially flash the bootloader, SW3 should be in NO BOOT Mode and SW3 pins 1, 2, 3 and 4 should all be 0.

Steps to flash the Bootloader

- First keep the EVM in NO BOOT mode
- In the case of DSK6455 and C6472 while flashing big endian boot loader

```
    * configure the Endian switch to Little.
```

- For all other platforms Endian switch need to reflect correct Big/Little depending on bootloader image being updated
- Then connect to the EVM
- Program Load the EEPROM writer BUT do not execute the program
- Data Load the bootloader to memory at location 0x800000
- Now execute the EEPROM writer. Please wait till you get the following message on the console

```
I2C write complete, reading data
```

```
I2C read complete, comparing data
```

```
Data compare passed
```

- In the case of DSK6455 and C6472 while flashing big endian boot loader following additional steps would be required

```
 * Configure the Endian switch back to Big
 * Disconnect the EVM from CCS
 * Reset the EVM
 * Connect the EVM to CCS
```

- As part of the release we are also releasing the GEL file(i2cConfig.gel) that is used to initialize the EVM
- Load the I2C parameter writer through CCS and execute the program.
- On the CCS console, the user will be asked to execute the appropriate function for the EVM being flashed
- There are separate functions in the GEl file to initialize all the 7 EVMs supported in the release
- Please take care to execute the correct initialization function.
- Take care to change the switched appropriately depending on the Endian and the boot mode

Steps to flash a Kernel Binary to the NAND

- We will be using the kernel NAND write utility to write a kernel to the NAND
- Please follow the steps below to write the kernel binary to the NAND flash

flash_eraseall /dev/mtd2

nandwrite -p /dev/mtd2 vmlinux-c64xx.bin

- Please take care to flash the appropriate kernel binary to the NAND
- This procedure assumes that the kernel binary is present in the filesystem

Hardware setup for c66x EVMs

Programming the EVM

Please pick up the c66x binary tarballs from [linux-c6x File download Area [3]]. There are separate tarballs for little and big endian. Untar the appropriate tarball.

Please Note

- When programming NAND and IBL for big endian, use little endian switch settings instead.
- When testing the big endian kernel, use big endian and SW3 pin 1 need to be ON

Switch Settings

Make sure the EVM dip switches are kept as below when programming the EVM for both little and big endian

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	**Off**	On	On	On
SW4	On	On	On	On
SW5	On	On	On	On
SW6	On	On	On	On

Set the Environment Variables

Please make sure the below environment variables needs to be set. Otherwise there could be some unexpected behavior experienced.

1. Set the DSS_SCRIPT_DIR environment variable (Mandatory)

Example: export DSS_SCRIPT_DIR=/opt/ti/ccsv5/ccs_base_5.0.3.00028/scripting/bin

2. Set the PROGRAM_EVM_TARGET_CONFIG_FILE environment variable for using the DVD provided ccxml files OR user ccxml files. (Optional)

Example: export PROGRAM_EVM_TARGET_CONFIG_FILE=/opt/configs/evm667xl/my_evm667xl.ccxml

This step is required only if you are using an emulator other than the one that came with your board. If you wish to use such an external emulator you will need to set ROGRAM_EVM_TARGET_CONFIG_FILE. If this environment variable is not set, the DSS script will use the default ccxml files that support the following emulators.

3. xds100 inbuilt (evm667xl-linuxhost.ccxml)

4. xds560 mezzanine card (evm667xle-linuxhost.ccxml)

Please note that depending on the emulator selected the restore image time may vary. For example, if xds100 inbuilt emulator is selected, the entire process may take over 60 minutes. If xds560 mezzanine card emulator is selected, the process may take about 10 minutes

DSS Script Arguments

program_evm$ $DSS_SCRIPT_DIR\dss.sh program_evm.js [tmdx|tmds]evm[c](6678|6670)l[x][e][-le|-be]

tmdx: TMDX type EVM

tmds: TMDS type EVM

c: Not used, for backward compatibility

6678: C6678 device

6670: C6670 device

l: Low cost EVM

x: EVM supports encryption

e: EVM uses 560 Mezzanine Emulator daughter card

le: Little Endian

be: Big Endian

Image Identifier

“nand,nor,eeprom50,eeprom51”: User can include one or more image identifier to program the image, if no image identifier specified, the script will program all the images.

Example:

```
$ $DSS_SCRIPT_DIR/dss.sh program_evm.js evm6678l-le “eeprom51,nand”
```

This will write the little endian nand.bin image to C6678 low cost EVM using XDS 100 emulator or

```
$ $DSS_SCRIPT_DIR%/dss.sh program_evm.js evm6670le-le
“eeprom50,eeprom51”
```

This will write the little endian eeprom50.bin and eeprom51.bin images to C6670 low cost EVM using XDS 560 Mezzanine emulator. If switched to big endian, eeprom51 need to be written since IBl is big endian.

Executing the DSS script

1. cd “program_evm” directory

2. Please note that PROGRAM_EVM_TARGET_CONFIG_FILE is not a mandatory environment variable, if not set it uses the default ccxml files as below.

3. Using the DSS Script batch file, run the “program_evm.js” script command from program_evm directory.

Example:

/program_evm>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6678L-LE

This will write all the little endian images to C6678 low cost EVM using XDS 100 emulator.

/program_evm>$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDXEVM6670LE-LE

This will write all the little endian images to C6670 low cost EVM using XDS 560 Mezzanine emulator.

DIP Switch Settings to Boot from NAND

For TMDXEVM6678L EVM, to boot Linux from NAND flash, make sure the Dip switches on the EVM are set as follows:-

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off
SW4	On	Off	On	On
SW5	On	On	On	Off
SW6	On	On	On	On

For TMDXEVM6670L EVM, to boot Linux from NAND flash, make sure the Dip switches on the EVM are set as follows:-

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off
SW4	On	**Off**	On	On
SW5	On	On	On	Off
SW6	On	On	On	On

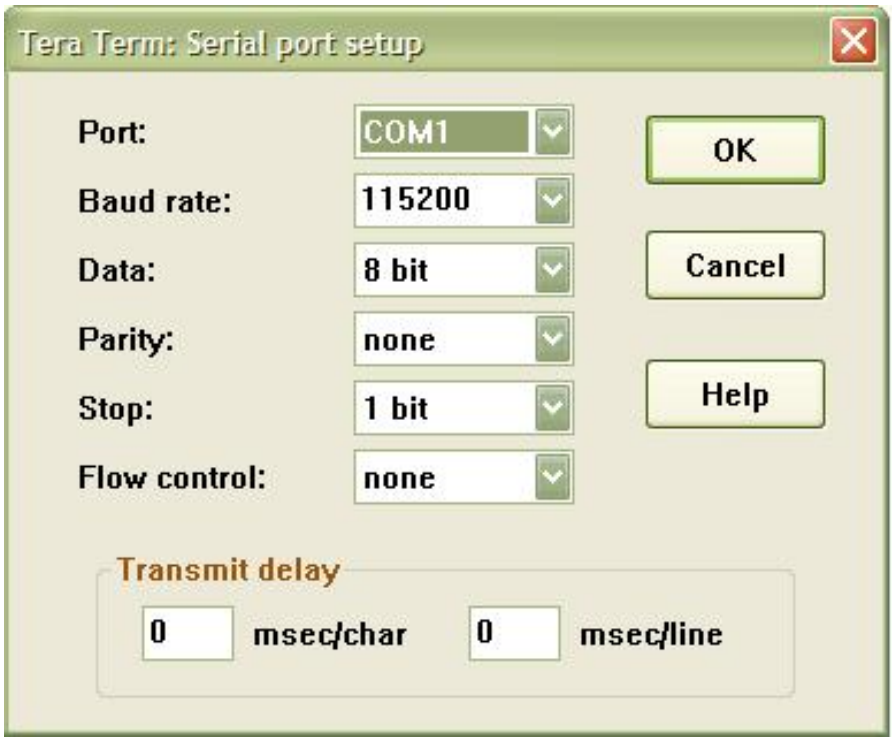
DIP Switch Settings to Boot from TFTP

For the TMDXEVM6678L EVM and TMDXEVM6670L EVM, to boot Linux kernel from a TFTP server, make sure the Dip switches on the EVM are set as follows:-

SWITCH	Pin1	Pin2	Pin3	Pin4
SW3	Off	Off	On	Off
SW4	On	On	**Off**	On
SW5	On	On	On	Off
SW6	On	On	On	On

Serial Port Setup

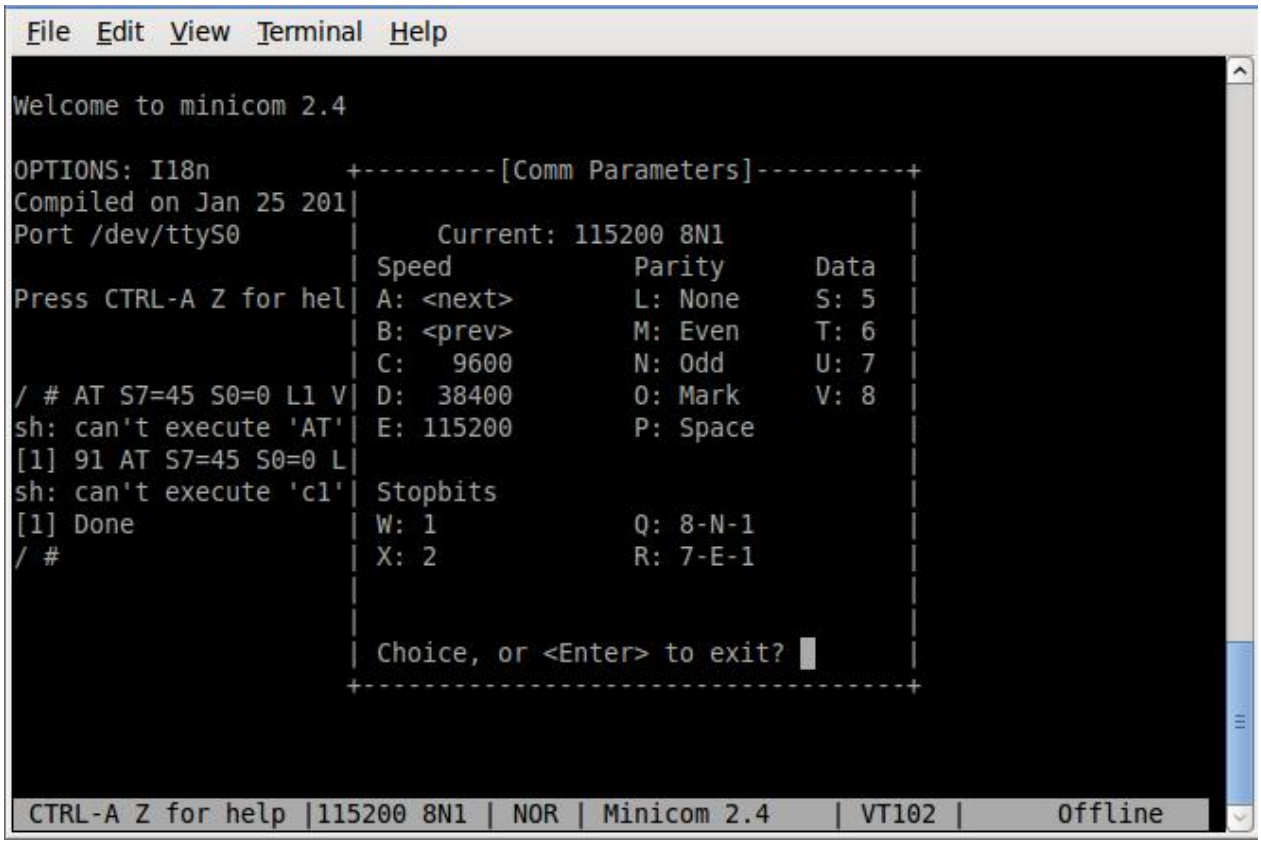
Connect one end of the RS232 Serial cable provided in the box to the serial port of the Host PC. Connect the other end to the on board RS 232 port (3 pin socket) on the EVM. If Host is running Windows OS, start Tera Term and configure the serial port settings as follows:-



If Host PC is running Linux OS, start minicom

> sudo minicom -s

Choose Serial Port Setup from the menu and then E for for comm parameters. Choose parameters as follows:-



Set serial device used by minicom to match the serial port on the Host PC connected to the EVM.

Choose Serial Port Setup from the menu and then A for Serial Device.

Save the configuration and run the minicom

> sudo minicom

Booting with Linux

Power the EVM using the power supply provided. The board should now boot up Linux kernel from NAND flash. A sample boot up log is provided below for TMDXEVM6678L EVM.

```
Linux version 2.6.34-evmc6678.el-20110504
(root@gtcs13.gt.design.ti.com) (gcc version
4.5.1 (Sourcery G++ Lite 4.5-124) ) #1 Wed July 27 11:14:47 EDT 2011
Designed for the EVMC6678 board, Texas Instruments.
CPU0: C66x rev 0x0, 1.2 volts, 1000MHz
Initializing kernel
Built 1 zonelists in Zone order, mobility grouping on.  Total pages:
130048
Kernel command line: console=ttyS0,115200 initrd=0x80400000,0x300000
ip=dhcp rw
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory available: 512760k/520340k RAM, 0k/0k ROM (784k kernel code,
200k data)
SLUB: Genslabs=7, HWalign=128, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Hierarchical RCU implementation.
RCU-based detection of stalled CPUs is enabled.
NR_IRQS:288
Console: colour dummy device 80x25
Calibrating delay loop... 999.42 BogoMIPS (lpj=1998848)
Mount-cache hash table entries: 512
C64x: 8 gpio irqs
NET: Registered protocol family 16
SGMII init complete
bio: create slab <bio-0> at 0
Switching to clocksource TSC64
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
```

```
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
Freeing initrd memory: 3072k freed
JFFS2 version 2.2. (NAND) (SUMMARY)  Â© 2001-2006 Red Hat, Inc.
ROMFS MTD (C) 2007 Red Hat, Inc.
msgmni has been set to 1007
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
Serial: 8250/16550 driver, 1 ports, IRQ sharing disabled
serial8250.0: ttyS0 at MMIO 0x2540000 (irq = 276) is a 16550A
console [ttyS0] enabled
brd: module loaded
loop: module loaded
at24 1-0050: 131072 byte 24c1024 EEPROM (writable)
uclinux[mtd]: RAM probe address=0x803da3c0 size=0x0
Creating 1 MTD partitions on "RAM":
0x000000000000-0x000000000000 : "ROMfs"
mtd: partition "ROMfs" is out of reach -- disabled
Generic platform RAM MTD, (c) 2004 Simtec Electronics
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB
1,8V 8-bit)
Scanning device for bad blocks
Creating 2 MTD partitions on "davinci_nand.0":
0x000000000000-0x000001000000 : "kernel"
0x000001000000-0x000004000000 : "filesystem"
davinci_nand davinci_nand.0: controller rev. 2.5
m25p80 spi0.0: n25q128 (16384 Kbytes)
Creating 1 MTD partitions on "spi_flash":
0x000000000000-0x000001000000 : "test"
spi_davinci spi_davinci.0: Controller at 0x20bf0000
spi_davinci spi_davinci.0: Operating in interrupt mode using IRQ 182
pktgen 2.72: Packet Generator for packet performance testing.
TCP cubic registered
NET: Registered protocol family 17
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 0.0.0.0, my address is 158.218.100.184
IP-Config: Complete:
     device=eth0, addr=158.218.100.184, mask=255.255.255.0,
gw=158.218.100.2,
     host=158.218.100.184, domain=am.dhcp.ti.com, nis-domain=(none),
     bootserver=0.0.0.0, rootserver=0.0.0.0, rootpath=
Freeing unused kernel memory: 140K freed
starting pid 17, tty : '/etc/rc.sysinit'

Starting system...
```

```
Mounting proc filesystem: done.
Mounting other filesystems: done.
Starting mdev
Setting hostname 158.218.100.184: done.
Bringing up loopback interface: done.
Starting inetd: done.

eth0      Link encap:Ethernet  HWaddr 90:D7:EB:07:12:93
          inet addr:158.218.100.184  Bcast:158.218.100.255
Mask:255.255.255.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1708 (1.6 KiB)  TX bytes:1180 (1.1 KiB)
          Interrupt:48

System started.

starting pid 54, tty '/dev/console': '/bin/sh'
/ #
```

Configuring IP address

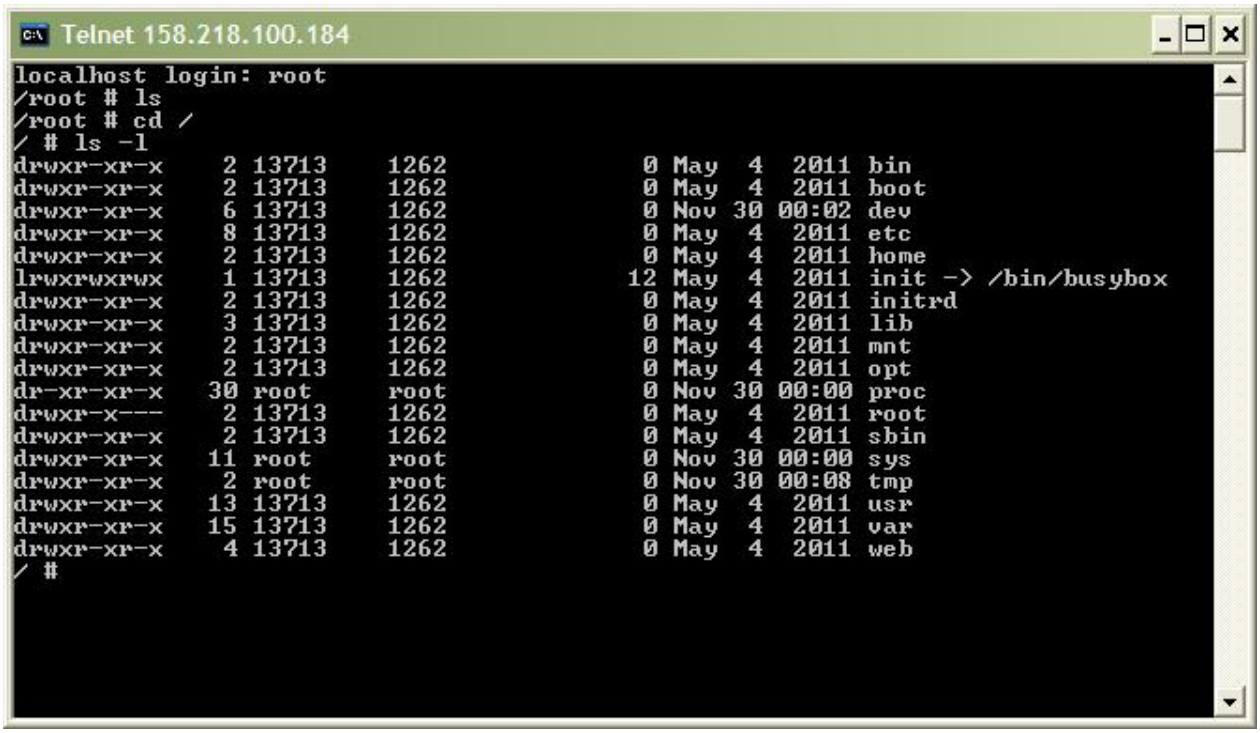
The EVM has Linux kernel image (with initramfs) programmed into the flash. By default, the Linux boot command line parameter is configured to assign IP address through DHCP. So typically user may connect the EVM Ethernet port to a LAN that has a DHCP server or connect it to a Ethernet switch port with a Windows or Linux Host PC connected to the same switch. In the former case IP address will get assigned automatically during boot up. If DHCP server is absent, during boot up DHCP requests time out after 2 attempts and boot prompt will be shown to the serial port console. To assign an IP address manually to eth0, user can use the ifconfig command from the serial console.

For example to assign 158.218.100.184 to eth0, user type

>ifconfig eth0 158.218.100.184 netmask 255.255.255.0 up

Telnet to the EVM

User will be able to telnet (user id root and no password) to the EVM using the IP address assigned to the eth0 interface. A sample telnet session is shown below:-



Transferring files to the filesystem on EVM

Some of the options are

1) tftp

From the console, user will be able to tftp files to the filesystem using the command

tftp -g -r min-root-c6x.cpio.gz 158.218.100.179

Where 158.218.100.179 is the IP address of the tftp server. min-root-c6x.cpio should be copied to the folder configured in the tftp server.

2) ftp

FTP server is enabled in the busybox. So from the Host machine, files can be transferred to the EVM using ftp command. The ftp server path in the target is set to /var/local. So files can be put or get from this folder.

Following commands are known to work in the ftp session:-

- cd
- get
- mget
- put
- mput

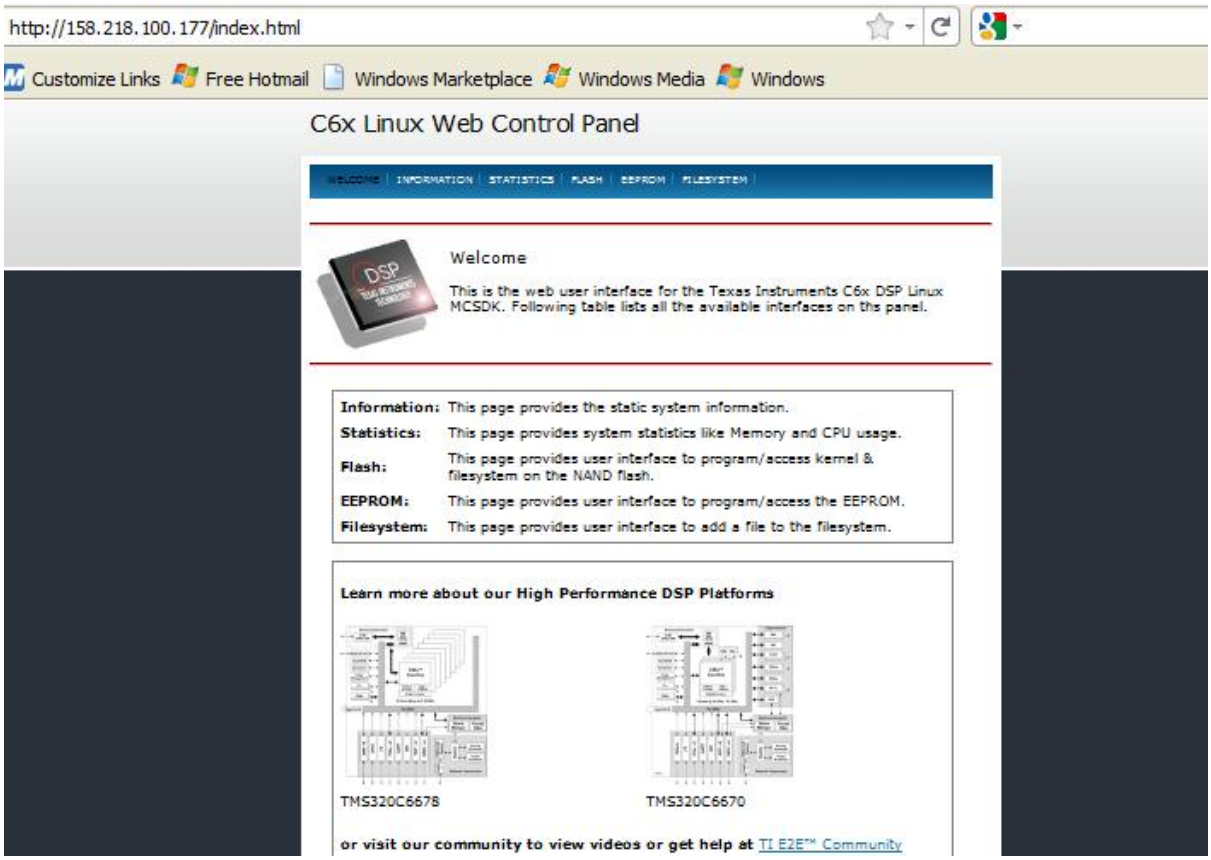
Sample session is shown below:

3) Using nfs

Refer instructions at [NFS Server Setup [4]]

Out of box Demo for c66x - c6x Linux Web control panel

Once the Linux Kernel is booted up, user will be able to access the Web control panel by pointing the browser to the eth0 IP address. A sample screenshot of the web control panel welcome page is shown below:-



Apart from the Welcome page, Web control panel implemented in this release has following menus:-

Information

This provides information about the Linux version running on the board, CPU info, mount information, network interfaces etc.

System Statistics

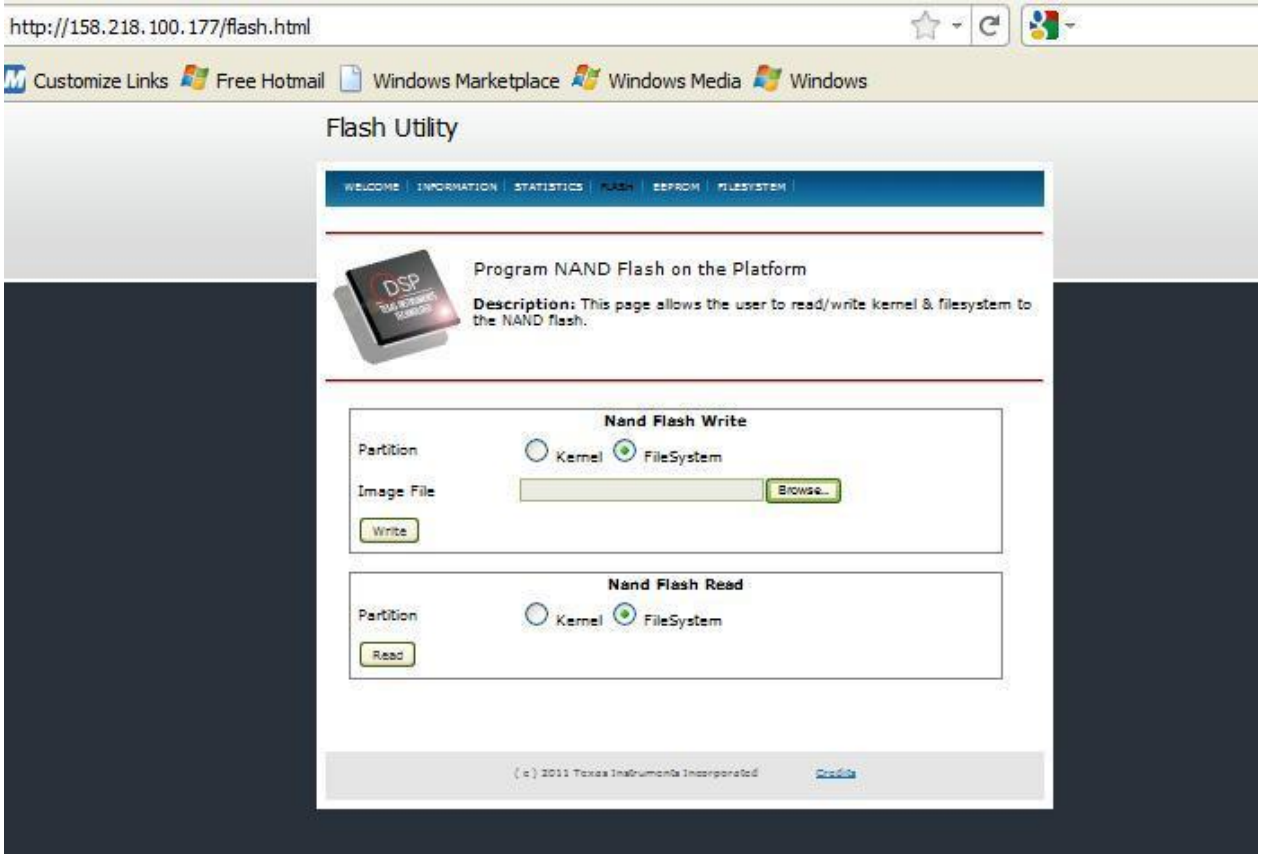
This provides information about memory and CPU usage

Flash Utility - Program NAND Flash on the platform

This page provides following capabilities to user:-

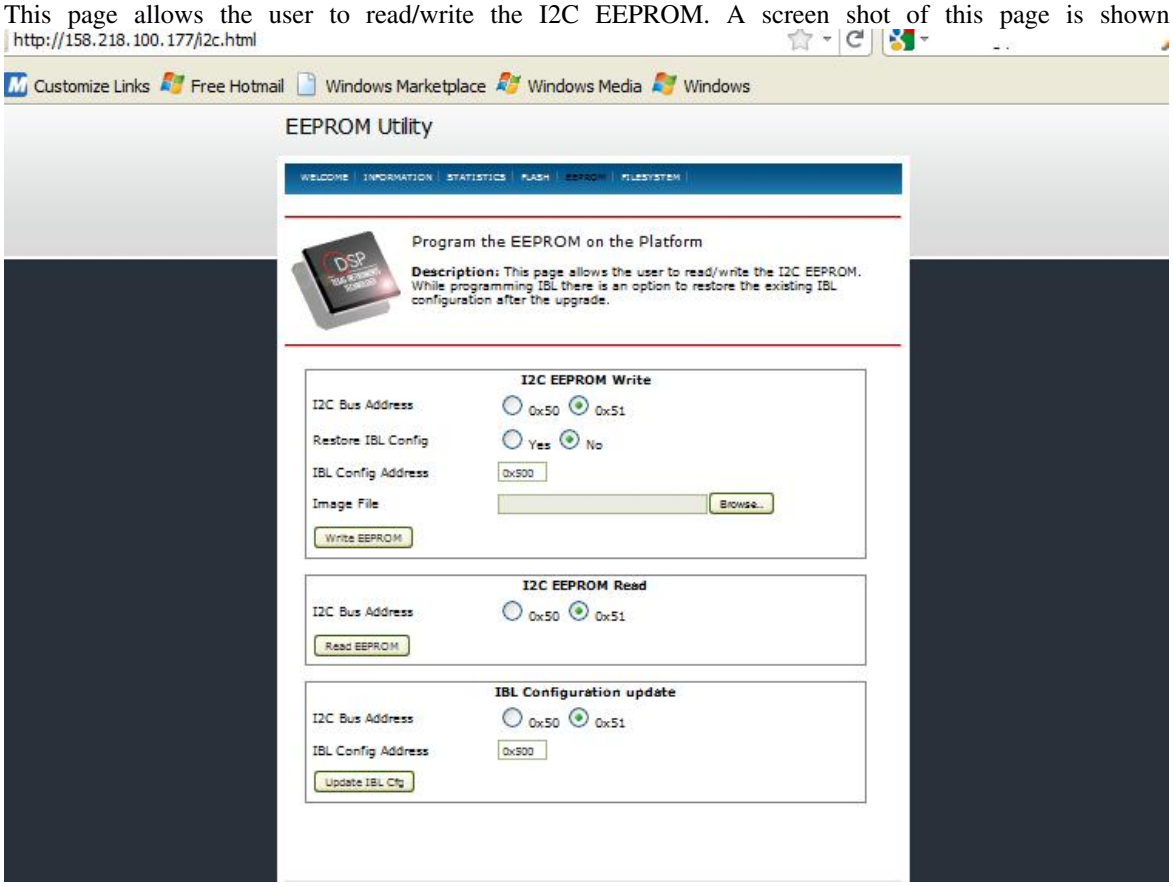
- NAND Flash Write. Allow user to Program the Kernel and Filesystem to the NAND flash.
- NAND Flash Read. Allow user to save the Kernel and Filesystem image to the Host PC.

A screen shot of this page is shown below:-



EEPROM Utility - Program the EEPROM on the platform

This page allows the user to read/write the I2C EEPROM. A screen shot of this page is shown below.



This provides following capabilities to user:-

- I2C EEPROM Write

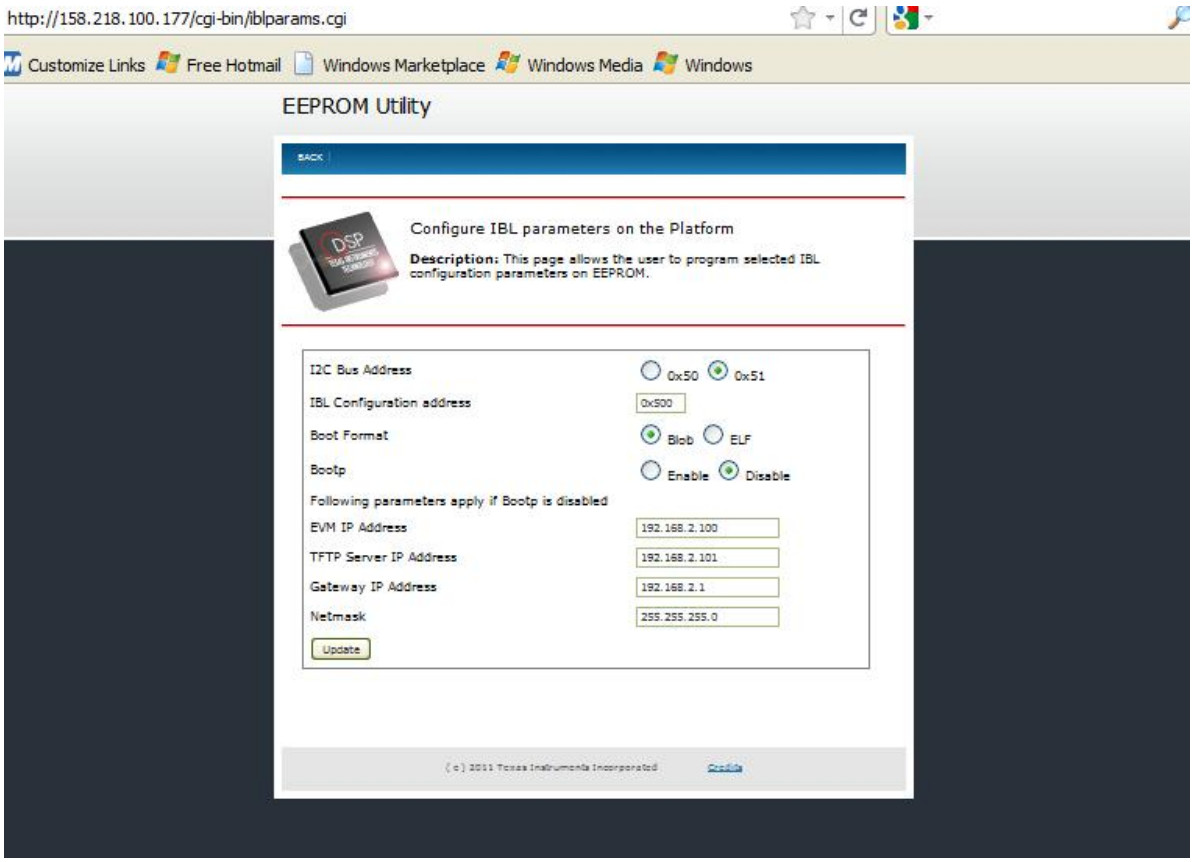
The EEPROM at i2c address 0x50 is by default programmed with POST image and at address x51 is programmed with IBL image. User will be able to program either of these using this interface. While programming the IBL image, there is an option to restore the existing IBL configuration.

- I2C EEPROM Read

This interface allows the user to save 64K of EEPROM data from bus address 0x50 or 0x51 to the Host PC.

- IBL Configuration update

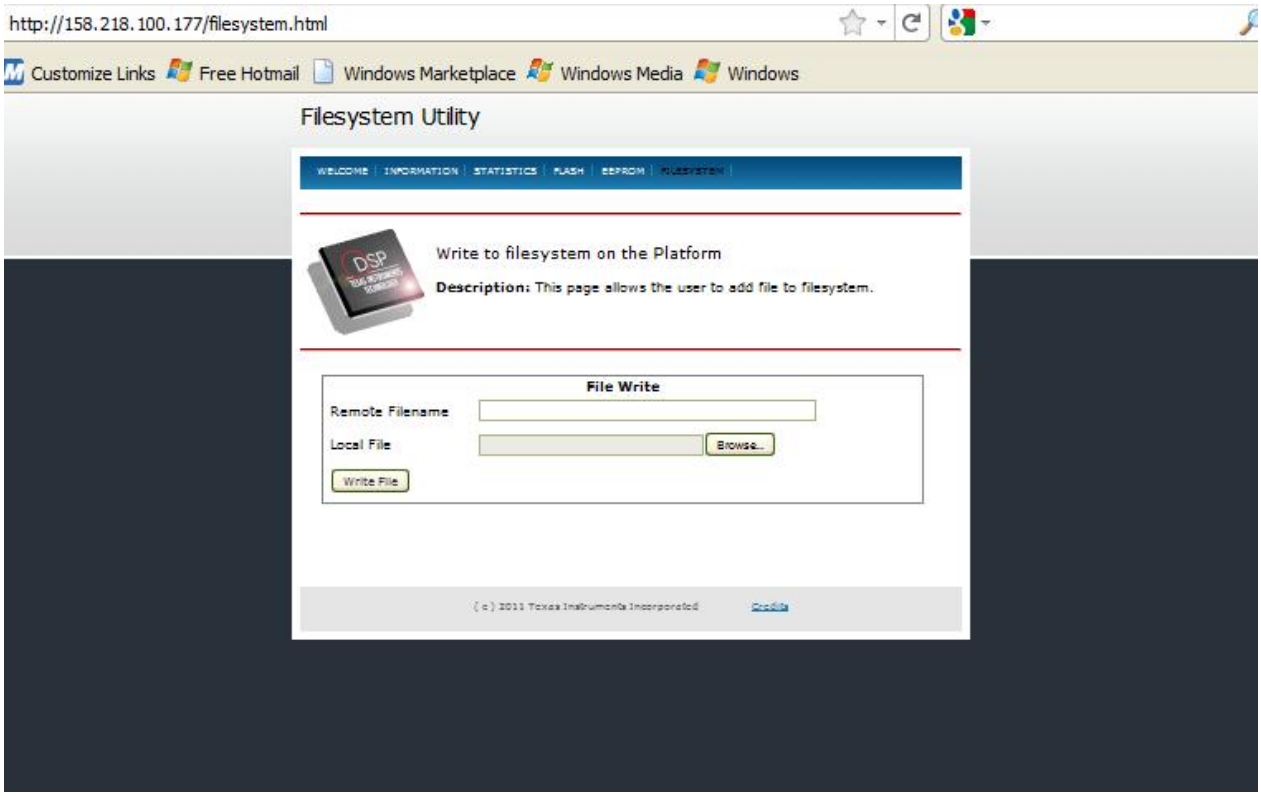
This interface allows the user to update the IBL configuration parameters. By default the configuration parameters are stored at offset 0x500 from the start of EEPROM. This is normally not changed by the user and has to match with the offset in the IBL program. Currently only TFTP boot parameters are configurable through this page. A Screenshot of the configuration update page is shown below.



Boot format refers to the format of the image downloaded by IBL which could be a binary blob or an ELF executable.

Filesystem Utility

This page allows the user to copy a file from the Host PC to the target filesystem on the EVM. A screen shot is shown below.



Building Linux kernel and filesystem from source

A tar ball of the source code is provided at the ti.com Software Download Site [5] or the latest Linux MCSDK release can be downloaded from linux-c6x.org File Download Area [3]

The steps to build kernel and filesystem from source code is given at linux-c6x MCSDK 2.0 Beta 2 Release Notes [6]

Support

Please send any support related questions to the TI e2e forum [7]

References

[1] http://linux-c6x.org/files/releases/linux-c6x-2.0-beta2/linux-c6x-2.0-beta2-release-notes.pdf
[2] http://ttssh2.sourceforge.jp/index.html.en
[3] http://linux-c6x.org/files/releases/linux-c6x-2.0-beta2/
[4] http://www.linux-c6x.org/wiki/index.php/How_to_setup_an_NFS_filesystem%3F
[5] http://software-dl.ti.com/sdoemb/sdoemb_public_sw/linux_mcsdk/02_00_BETA_2/index_FDS.html
[6] http://linux-c6x.org/wiki/index.php/Linux-c6x_2.0-Beta2_Release#Configuration_and_Build
[7] http://e2e.ti.com/support/embedded/f/354.aspx

Article Sources and Contributors

Linux-c6x-2.0-beta2 User Guide *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=111692 *Contributors*: BillMills, Elvitalobo, Raghunambiath, Spaulraj

Image Sources, Licenses and Contributors

File:Endian-table.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Endian-table.JPG *License*: unknown *Contributors*: Spaulraj
File:C6457-switch.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C6457-switch.JPG *License*: unknown *Contributors*: Spaulraj
File:C6474L-switch.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C6474L-switch.JPG *License*: unknown *Contributors*: Spaulraj
File:C6472-switch.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C6472-switch.JPG *License*: unknown *Contributors*: Spaulraj
File:C6474dial-switch.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C6474dial-switch.JPG *License*: unknown *Contributors*: Spaulraj
File:C66x-def.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C66x-def.JPG *License*: unknown *Contributors*: Spaulraj
File:evmc6678l-dip-switches.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Evmc6678l-dip-switches.JPG *License*: unknown *Contributors*: Mkaricheri
File:evmc6670l-dip-switches.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Evmc6670l-dip-switches.JPG *License*: unknown *Contributors*: Mkaricheri
File:C66x-tftp.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:C66x-tftp.JPG *License*: unknown *Contributors*: Spaulraj
File:tera-term.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Tera-term.JPG *License*: unknown *Contributors*: Mkaricheri
File:minicom.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Minicom.JPG *License*: unknown *Contributors*: Mkaricheri
file:telnet.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Telnet.JPG *License*: unknown *Contributors*: Mkaricheri
file:ftp.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Ftp.JPG *License*: unknown *Contributors*: Mkaricheri
file:Demo-web-control_panel.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Demo-web-control_panel.JPG *License*: unknown *Contributors*: Mkaricheri
file:nand-flash-2.0-alpha2.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Nand-flash-2.0-alpha2.JPG *License*: unknown *Contributors*: Mkaricheri
file:eeprom-utility-2.0-alpha2.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Eeprom-utility-2.0-alpha2.JPG *License*: unknown *Contributors*: Mkaricheri
file:ibl-config-update-2.0-alpha2.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Ibl-config-update-2.0-alpha2.JPG *License*: unknown *Contributors*: Mkaricheri
file:filesystem-2.0-alpha2.JPG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Filesystem-2.0-alpha2.JPG *License*: unknown *Contributors*: Mkaricheri