Linux-c6x-2.0-GA 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 GA release (2.0.0.63) of Linux-c6x / Linux-MCSDK.

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

The latest version of this user's guide for the 2.x versions can be found on the linux-c6x.org Wiki [2]

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 [3]] serial port program.

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

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

A couple of the older C64x board require a low cost external emulator to be compatible with a Linux Host. See the board specific information below.

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 [4].

Hardware setup for c64x+ EVMs

Please pick up the c64x binary tar files from [linux-c6x File download Area [4]]. There are separate tar files for little and big endian. Untar the appropriate tar file and then inside the product directory invoke the bootblob all command. An example follows:

```
 mkdir ~/my-linux-c6x
 cd    ~/my-linux-c6x
 wget http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/
linux-c6x-2.0.0.63-c64le-bin.tar.gz
 tar xvzf linux-c6x-2.0.0.63-c64le-bin.tar.gz
 cd linux-c6x-2.0.0.63/product
 ./bootblob all
 cd ../program_evm
```

Boot Mode Settings

C6457

- The C6457 EVM has a NAND and the bootloader supports both a TFTP and a NAND boot modes
- SW4 selects the chip/ROM boot mode
- SW3 pin 1 is used by IBL to select between TFTP and NAND boot mode.
- These settings are depicted below

C6457	Switch	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin8
Master I2C Boot Mode (IBL)	SW4	**On**	Off	Off	Off	Off	Off	Off	Off
No Boot Mode (JTAG)	SW4	Off	Off	Off	Off	Off	Off	Off	Off
TFTP Boot (for IBL)	SW3	Off	Off	Off	Off	Off	Off	Off	**E**
NAND Boot (for IBL)	SW3	**On**	Off	Off	Off	**On**	Off	Off	**E**

E == On for little endian and Off for big endian

C6474L

This is the C6474L EVM from eInfochips with a single C6474 chip.

- 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 TFTP and NAND boot modes
- SW4 selects the chip/ROM bootmode
- SW3 pin 1 is used by IBL to select between TFTP and NAND boot modes
- These settings are depicted below

C6474L	Switch	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin8
Master I2C Boot Mode (IBL)	SW4	**On**	Off	Off	Off	**E**			
No Boot Mode (JTAG)	SW4	Off	Off	Off	Off	**E**			
TFTP Boot (for IBL)	SW3	Off	Off	Off	Off	**On**	Off	Off	Off
NAND Boot (for IBL)	SW3	**On**	Off	Off	Off	**On**	Off	Off	Off

E == On for little endian and Off for big endian

C6472

- 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 TFTP and NAND boot modes
- SW2 selects the chip/ROM boot mode
- SW1 pin 1 is used by IBL to select between TFTP and NAND boot modes
- These settings are depicted below

C6472	**Switch**	**Pin 1**	**Pin 2**	**Pin 3**	**Pin 4**	**Pin 5**	**Pin 6**	**Pin 7**	**Pin8**
Master I2C Boot Mode (IBL)	SW2	Off	**On**	Off	Off	**On**			
No Boot Mode (JTAG)	SW2	Off	Off	Off	Off	**On**			
TFTP Boot (for IBL)	SW1	Off	Off	Off	Off	Off	**On**	**On**	**E**
NAND Boot (for IBL)	SW1	**On**	Off	Off	Off	Off	**On**	**On**	**E**
	SW3	Off	Off						

E == On for little endian and Off for big endian

C6474

This is the original C6474 EVM from Spectrum Digital with dual C6474 chips.

- The C6474 supports only TFTP boot
- SW5 controls the chip/ROM boot mode.
- **This board requires an external XDS100v2 emulator**
- These settings are depicted below

Note that the board documents show logic levels of the configuration not switch settings. A switch On == logic 0 and switch Off == logic 1.

C6474	**Switch**	**Pin 1**	**Pin 2**	**Pin 3**	**Pin 4**	**Pin 5**	**Pin 6**	**Pin 7**	**Pin 8**
Master I2C Boot Mode (IBL)	SW5	**Off**	On	On	On	**E**	**Off**	**Off**	**Off**
No Boot Mode (JTAG)	SW5	On	On	On	On	**E**	**Off**	**Off**	**Off**
	SW1	On	**Off**	**Off**	On				

E == Off for little endian and On for big endian
Note that this is different than most of the other boards

DSK6455

This is the cheaper EVM with a single 6455 chip. The EVM6455 should also be compatible if the daughter card with the second 6455 is disconnected.

- The DSK6455 Supports only TFTP Boot
- SW3 selects the chip/ROM boot mode
- **This board requires an external XDS100v2 emulator**
- *CCS on a Linux host does not work with the on board emulation on this board*
- These settings are depicted below

DSK6455	**switch**	**Pin 1**	**Pin 2**	**Pin 3**	**Pin 4**	**Pin 5**	**Pin 6**	**Pin 7**	**Pin 8**
Master I2C Boot Mode (IBL)	SW3	**E**	**On**	Off	**On**	Off	Off	Off	Off
I2C Slave Boot Mode (JTAG)	SW3	**E**	Off	**On**	**On**	Off	Off	Off	Off

E == Off for little endian and On for big endian
Note that this is different than most of the other boards

Steps to flash the Bootloader

- With power off, put the EVM in the JTAG mode specified in the board specific tables above
- The Endian switch needs to reflect the endianness depending on the bootloader image being updated
- Set the appropriate environment variables as described in Set the Environment Variables
- Power on the EVM and wait for the emulator to initialize and complete USB enumeration
  - This is about 10 seconds for an XDS100 and 45 seconds for an XDS560
- Run program_evm to write the IBL image to the EEPROM

```
cd ~/my-linux-c6x/program_evm
$DSS_SCRIPT_DIR/dss.sh program_evm.js evm6472l-le
```

- You will get the following message on the console on completion.

```
 I2C Bus address is = 80
 I2C writing has started
 Please be Patient
 I2C write complete, reading data
 I2C read complete, comparing data
 Data compare passed
```

- For c64x devices, program_evm does not support writing to NAND flash. Follow the steps below to program the NAND flash
- This procedure updates only eeprom50; ignore all warnings about not updating other memories.

Steps to update NAND content

The C6472, C6457 and C6474L EVMs have a NAND and NAND boot is supported on these EVMs

- Update the bootloader using the above procedure if not already done
- Use the bootblob templates to create the initramfs-min bootblob image and the image you wish to write to NAND
  - Below we will assume we are writing the same initramfs-min image to NAND

```
cd ~/my-linux-c6x/product
./bootblob all
```

- Boot the initramfs-min bootblob via TFTP
- Transfer the nand content to the board's filesystem using ftp or some other method

```
host$ ftp 158.218.100.184
root
type bin
hash
put evmc6472-initramfs-min.el-linux-c6x-2.0.0.63.bin
quit
```

- Connect to the target via telnet to serial port
- On the target, use the MTD utilities to update the NAND

```
cd /var/local
flash_eraseall /dev/mtd2
nandwrite -p /dev/mtd2 evmc6472-initramfs-min.el-linux-c6x-2.0.0.63.bin
```

Please take care to flash the appropriate kernel binary to the NAND.
See below sections for more information on serial port, telnet, and transfering files to the EVM.

Hardware setup for c66x EVMs

Programming the EVM

Please pick up the c66x binary tar files from [linux-c6x File download Area [4]]. There are separate tar files for little and big endian. Untar the appropriate tar file and then inside the product directory invoke the bootblob all command. An example follows:

```
 mkdir ~/my-linux-c6x
 cd    ~/my-linux-c6x
 wget http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/
linux-c6x-2.0.0.63-c66le-bin.tar.gz
 tar xvzf linux-c6x-2.0.0.63-c66le-bin.tar.gz
 cd linux-c6x-2.0.0.63/product
 ./bootblob all
 cd ../program_evm
```

Please Note

- When programming I2C, NAND, or NOR using program_evm, we always use little endian switch settings (even if programming big endian content)
- When booting IBL & kernel, we use the correct endian for the programed content (SW3 pin 1 ON == big; OFF == little endian)

Switch Settings

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

EVM667x Program_evm

Switch	Pin 1	Pin 2	Pin 3	Pin 4
SW3	**Off**	On	On	On
SW4	On	On	On	On
SW5	On	On	On	On
SW6	On	On	On	On

Note that the above only applies to running the program_evm script which use writers that operate in little endian mode. To load and debug your programs directly using JTAG without running IBL you would set the endian switch for the desired mode:

EVM667x JTAG operation

Switch	Pin 1	Pin 2	Pin 3	Pin 4
SW3	**E**	On	On	On
SW4	On	On	On	On
SW5	On	On	On	On
SW6	On	On	On	On

E == Off for little endian and On for big endian

Set the Environment Variables

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

Set the DSS_SCRIPT_DIR environment variable (Mandatory)

Example:

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

If you are NOT using an emulator that came with your board, you will have to point the PROGRAM_EVM_TARGET_CONFIG_FILE environment variable to an appropriate ccmxl file.

Custom emulator example ***(not common)***:

```
export
PROGRAM_EVM_TARGET_CONFIG_FILE=/opt/configs/evm667xl/my_evm667xl.ccxml
```

Please note that depending on the emulator selected the restore image time may vary. For example, if on board xds100 emulator is selected, the entire process may take 45 to 60 minutes. If xds560 mezzanine card emulator is selected, the process may take 4 to 10 minutes.

If you are using the on board XDS100, you may wish to update just the bootloader in I2C, boot the initramfs-demo bootblob and use the web control panel to update the flash. This procedure requires a working DHCP & TFTP server but is much faster than using just the XDS100. Updating just the bootloader takes a couple of 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 (default)

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.

Update the CCS GEL files

The program_evm/gel directory contains updated GEL files that must be installed into CCS. See the README.txt in that same directory for instructions.

If you use a version of CCS greater than CCSv5.0.3.00028, check the version numbers at the top of the two gel files (one from CCS and one from program_evm) and use the latest version.'

If you don't update the GELs, program_evm has a 10-20% change of silently programming the wrong data into I2C, NAND, and/or NOR when using TMX EVMs

Executing the DSS script

Using the DSS Script, run the “program_evm.js” script command from program_evm directory and supply the name of your board. Append a -be to program big endian content.

Example:

```
cd ~/my-linux-c6x/program_evm
$DSS_SCRIPT_DIR/dss.sh program_evm.js evm6678l
```

The above will write all the little endian images to C6678 low cost EVM using the on board XDS 100 emulator.

```
cd ~/my-linux-c6x/program_evm
$DSS_SCRIPT_DIR/dss.sh program_evm.js evm6670le-be eeprom51
```

The above will write just the big endian (**-be** suffix) bootloader image (**eeprom51**) to C6670 low cost EVM using XDS 560 Mezzanine emulator (**e** in evm6670le).

Note: The Linux-MCSDK or linux-c6x.org version of program_evm only updates the IBL bootloader in IC2 (eeprom51 for C66) and the kernel partition of the NAND (nand). It intentional skips the other half of the I2C (eeprom50 for C66x) and the SPI NOR flash (nor). You may safely ignore and warnings about not having the files to update these memories. If you wish to program all the memories on your EVM, see the program_evm on the DVD that came with your EVM.

DIP Switch Settings to Boot from NAND

For EVM6678L and EVM6670L, to boot Linux from NAND flash, make sure the Dip switches on the EVM are set as follows:-

EVM667xL NAND BOOT

Switch	**Pin 1**	**Pin 2**	**Pin 3**	**Pin 4**
SW3	**E**	**Off**	On	**Off**
SW4	On	**Off**	On	On
SW5	On	On	On	**Off**
SW6	On	On	On	On

E == Off for little endian and On for big endian

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:-

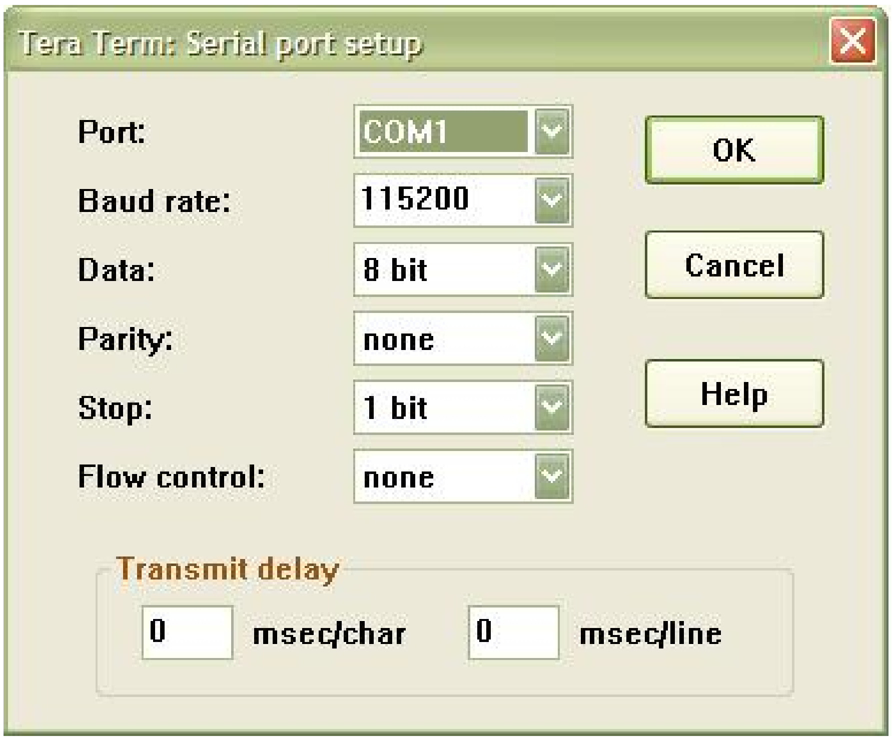
EVM667xL TFTP BOOT

Switch	**Pin 1**	**Pin 2**	**Pin 3**	**Pin 4**
SW3	**E**	**Off**	On	**Off**
SW4	On	On	**Off**	On
SW5	On	On	On	**Off**
SW6	On	On	On	On

E == Off for little endian and On for big endian

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:-

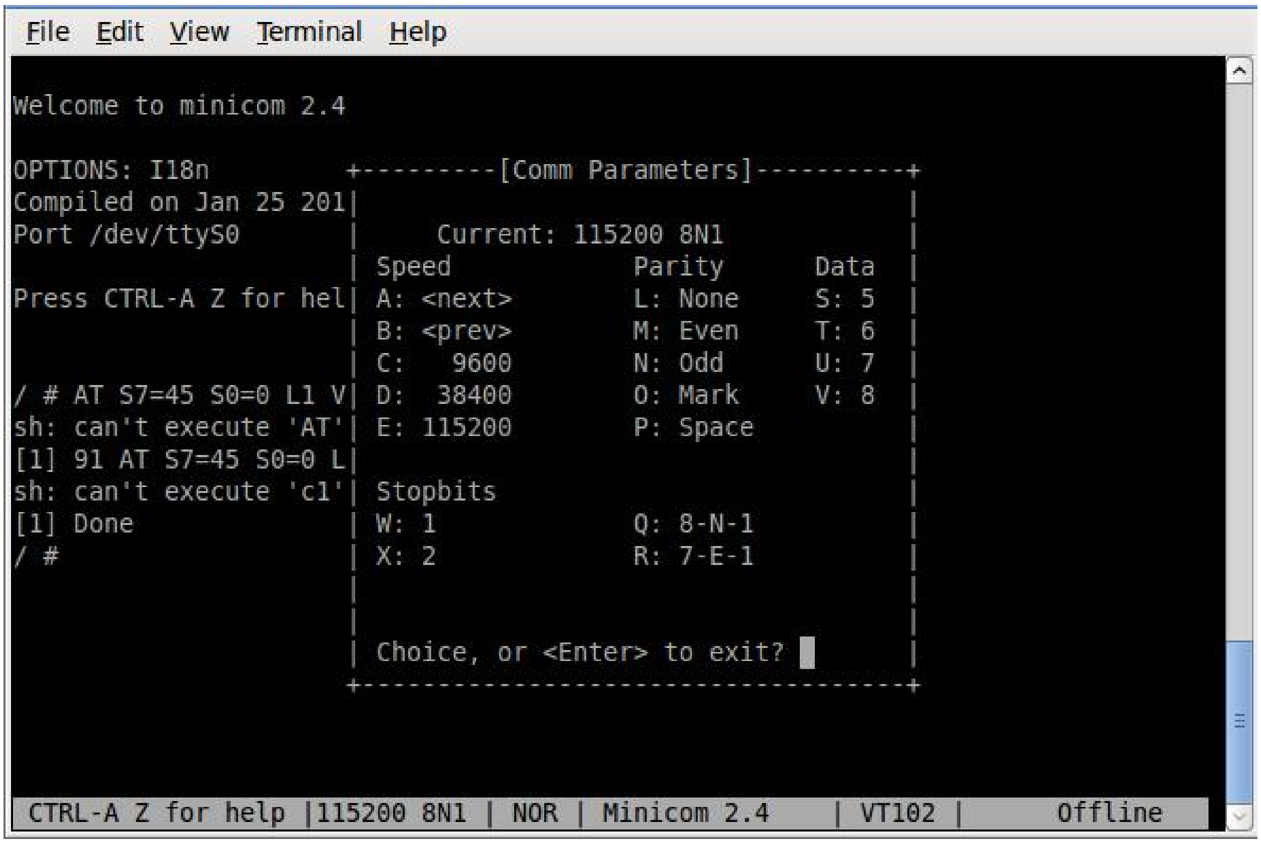


```
Port:           COM1
Baud rate:      115200
Data:           8 bit
Parity:         none
Stop:           1 bit
Flow control:   none

Transmit delay
0 msec/char     0 msec/line
```

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:-



```
Welcome to minicom 2.4

OPTIONS: I18n         +---------[Comm Parameters]----------+
Compiled on Jan 25 201|                                    |
Port /dev/ttyS0       |     Current: 115200 8N1            |
                      | Speed            Parity      Data  |
Press CTRL-A Z for hel| A: <next>        L: None     S: 5  |
                      | B: <prev>        M: Even     T: 6  |
                      | C:   9600        N: Odd      U: 7  |
/ # AT S7=45 S0=0 L1 V| D:  38400        O: Mark     V: 8  |
sh: can't execute 'AT'| E: 115200        P: Space          |
[1] 91 AT S7=45 S0=0 L|                                    |
sh: can't execute 'c1'| Stopbits                           |
[1] Done              | W: 1             Q: 8-N-1          |
/ #                   | X: 2             R: 7-E-1          |
                      |                                    |
                      |                                    |
                      | Choice, or <Enter> to exit?        |
                      +------------------------------------+

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4    | VT102 |      Offline
```

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.

```
IBL: PLL and DDR Initialization Complete
IBL Result code 00
IBL: Booting from NAND
Linux version 2.6.34-evmc6678.el-linux-c6x-2.0.0.63
(ubuntu@cm-build.linux-c6x.org) (gcc version 4.5.1 (Sourcery CodeBench
Lite 4.5-124) ) #1 Sat Dec 10 07:09:40 UTC 2011
Designed for the EVMC6678 board, Texas Instruments.
CPU0: C66x rev 0x0, 1.2 volts, 1000MHz
Initializing kernel
physical RAM map changed by user
Built 1 zonelists in Zone order, mobility grouping on.  Total pages:
65024
Kernel command line: console=ttyS0,115200 rw mem=256M ip=dhcp
initrd=0x80400000,0x500000
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory available: 250776k/258156k RAM, 0k/0k ROM (793k kernel code,
201k 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: 9 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: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
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: 5120k freed
JFFS2 version 2.2. (NAND) (SUMMARY)  �© 2001-2006 Red Hat, Inc.
ROMFS MTD (C) 2007 Red Hat, Inc.
msgmni has been set to 499
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=0x803e4760 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 3 MTD partitions on "davinci_nand.0":
0x000000000000-0x000000004000 : "bootconfig"
0x000000004000-0x000001000000 : "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
keystone_netcp keystone_netcp.0: firmware: using built-in firmware
keystone-pdsp/qmss_pdsp_acc48_le.fw
keystone_netcp keystone_netcp.0: firmware: using built-in firmware
keystone-pdsp/pa_pdsp_default.fw
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 192.168.1.102, my address is
```

```
192.168.1.87
IP-Config: Complete:
     device=eth0, addr=192.168.1.87, mask=255.255.255.0,
gw=255.255.255.255,
     host=tic6678-0d1206, domain=, nis-domain=(none),
     bootserver=192.168.1.102, rootserver=192.168.1.102, 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 tic6678-0d1206: done.
Bringing up loopback interface: done.
Starting inetd: done.

eth0      Link encap:Ethernet  HWaddr 90:D7:EB:0D:12:06
          inet addr:192.168.1.87  Bcast:192.168.1.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 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:692 (692.0 B)  TX bytes:1180 (1.1 KiB)
          Interrupt:48

System started.

starting pid 66, tty '/dev/console': '/bin/sh'
/ #
```

Configuring IP address

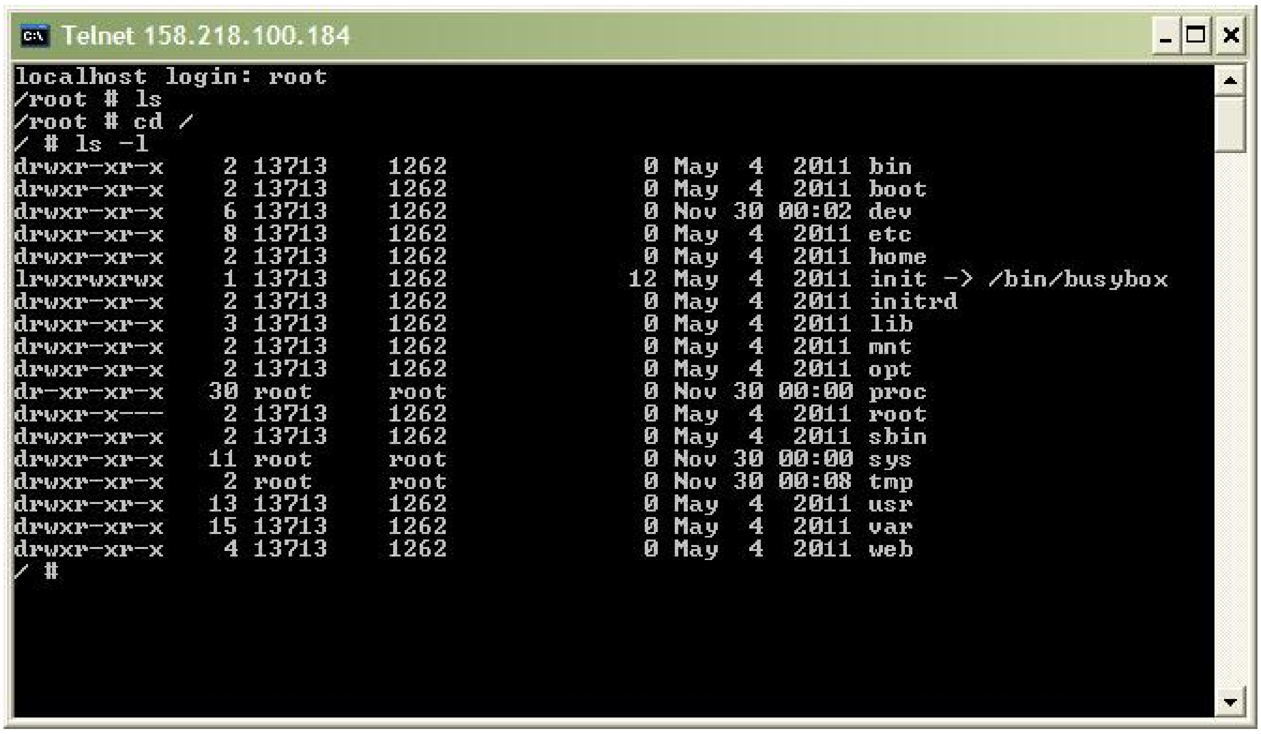
The EVM has Linux kernel image (with initramfs) programmed into the flash. By default, the Linux boot command line parameter is configured to assign IP address through DHCP. So typically user may connect the EVM Ethernet port to a LAN that has a DHCP server or connect it to a Ethernet switch port with 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:-



```
localhost login: root
/root # ls
/root # cd /
/ # ls -l
drwxr-xr-x    2 13713    1262             0 May  4  2011 bin
drwxr-xr-x    2 13713    1262             0 May  4  2011 boot
drwxr-xr-x    6 13713    1262             0 Nov 30 00:02 dev
drwxr-xr-x    8 13713    1262             0 May  4  2011 etc
drwxr-xr-x    2 13713    1262             0 May  4  2011 home
lrwxrwxrwx    1 13713    1262            12 May  4  2011 init -> /bin/busybox
drwxr-xr-x    2 13713    1262             0 May  4  2011 initrd
drwxr-xr-x    3 13713    1262             0 May  4  2011 lib
drwxr-xr-x    2 13713    1262             0 May  4  2011 mnt
drwxr-xr-x    2 13713    1262             0 May  4  2011 opt
dr-xr-xr-x   30 root     root             0 Nov 30 00:00 proc
drwxr-x---    2 13713    1262             0 May  4  2011 root
drwxr-xr-x    2 13713    1262             0 May  4  2011 sbin
drwxr-xr-x   11 root     root             0 Nov 30 00:00 sys
drwxr-xr-x    2 root     root             0 Nov 30 00:08 tmp
drwxr-xr-x   13 13713    1262             0 May  4  2011 usr
drwxr-xr-x   15 13713    1262             0 May  4  2011 var
drwxr-xr-x    4 13713    1262             0 May  4  2011 web
/ #
```

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.

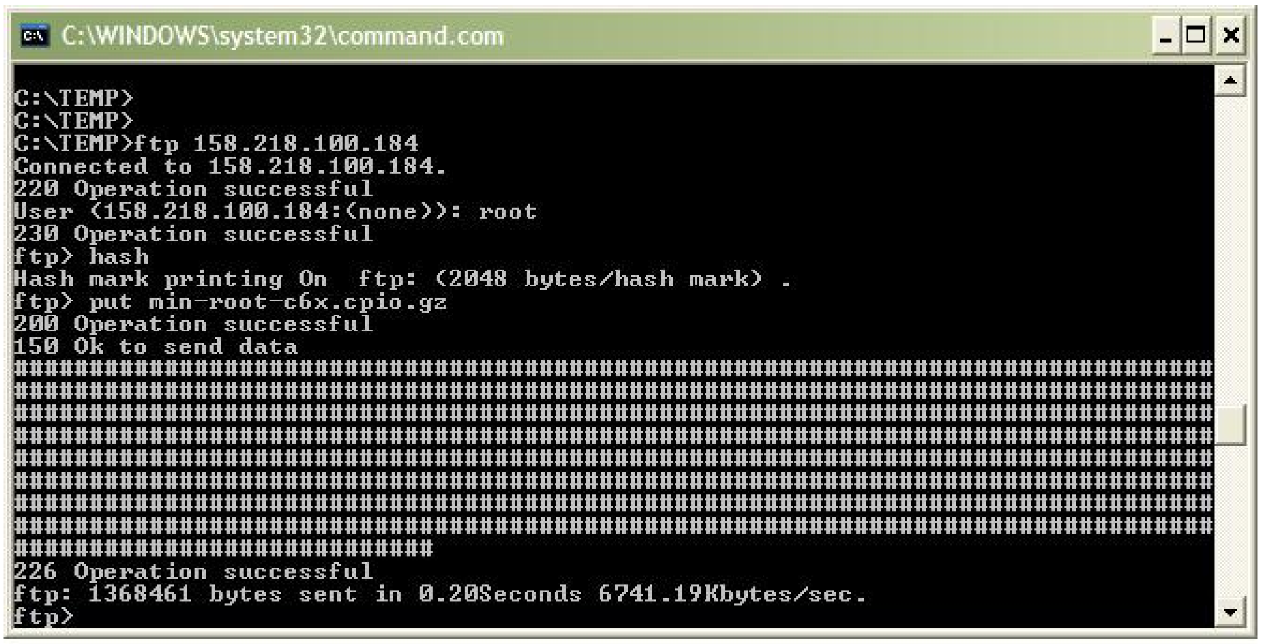
2) ftp

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:



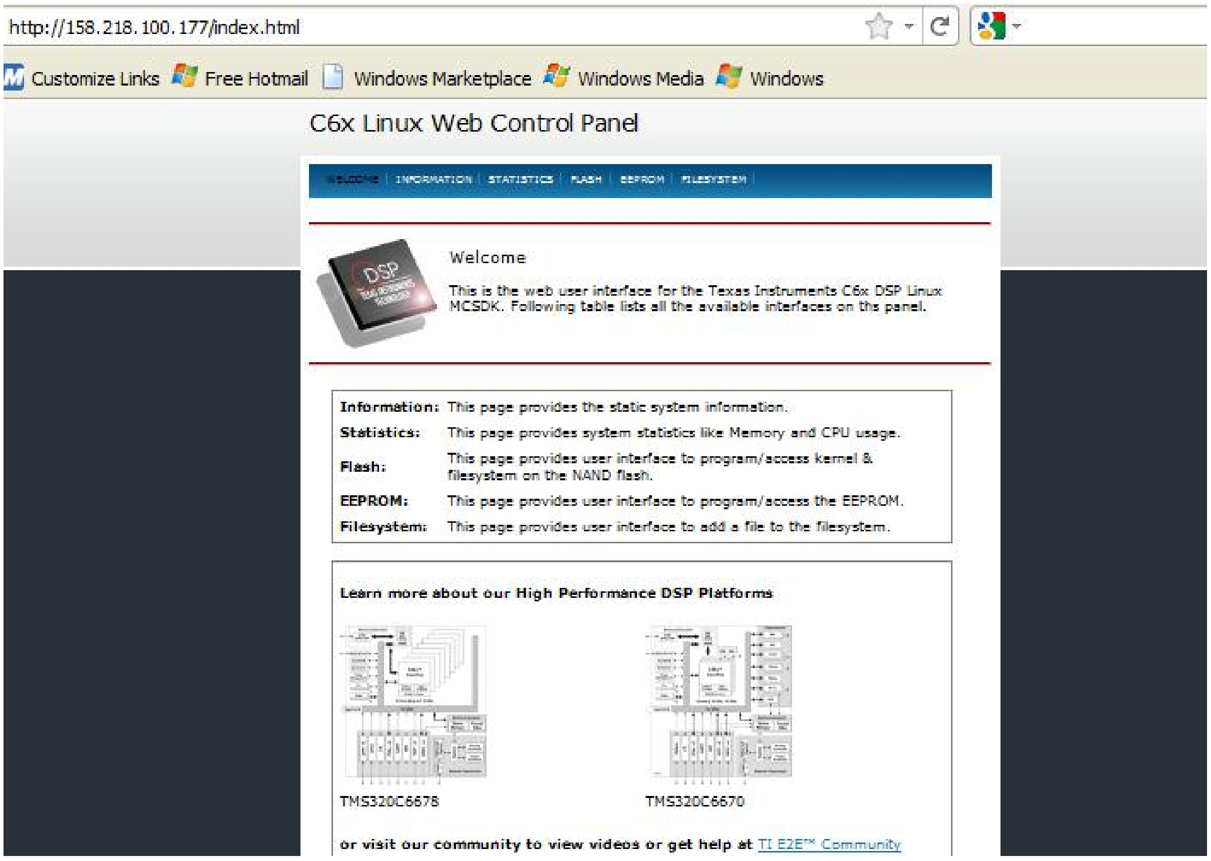
```
C:\WINDOWS\system32\command.com

C:\TEMP>
C:\TEMP>
C:\TEMP>ftp 158.218.100.184
Connected to 158.218.100.184.
220 Operation successful
User (158.218.100.184:(none)): root
230 Operation successful
ftp> hash
Hash mark printing On  ftp: (2048 bytes/hash mark) .
ftp> put min-root-c6x.cpio.gz
200 Operation successful
150 Ok to send data
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
############################
226 Operation successful
ftp: 1368461 bytes sent in 0.20Seconds 6741.19Kbytes/sec.
ftp>
```

3) Using nfs

Refer instructions at [NFS Server Setup [5]]

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:-



C6x Linux Web Control Panel

Welcome

This is the web user interface for the Texas Instruments C6x DSP Linux MCSDK. Following table lists all the available interfaces on the panel.

Information:	This page provides the static system information.
Statistics:	This page provides system statistics like Memory and CPU usage.
Flash:	This page provides user interface to program/access kernel & filesystem on the NAND flash.
EEPROM:	This page provides user interface to program/access the EEPROM.
Filesystem:	This page provides user interface to add a file to the filesystem.

Learn more about our High Performance DSP Platforms

TMS320C6678 TMS320C6670

or visit our community to view videos or get help at TI E2E™ Community

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

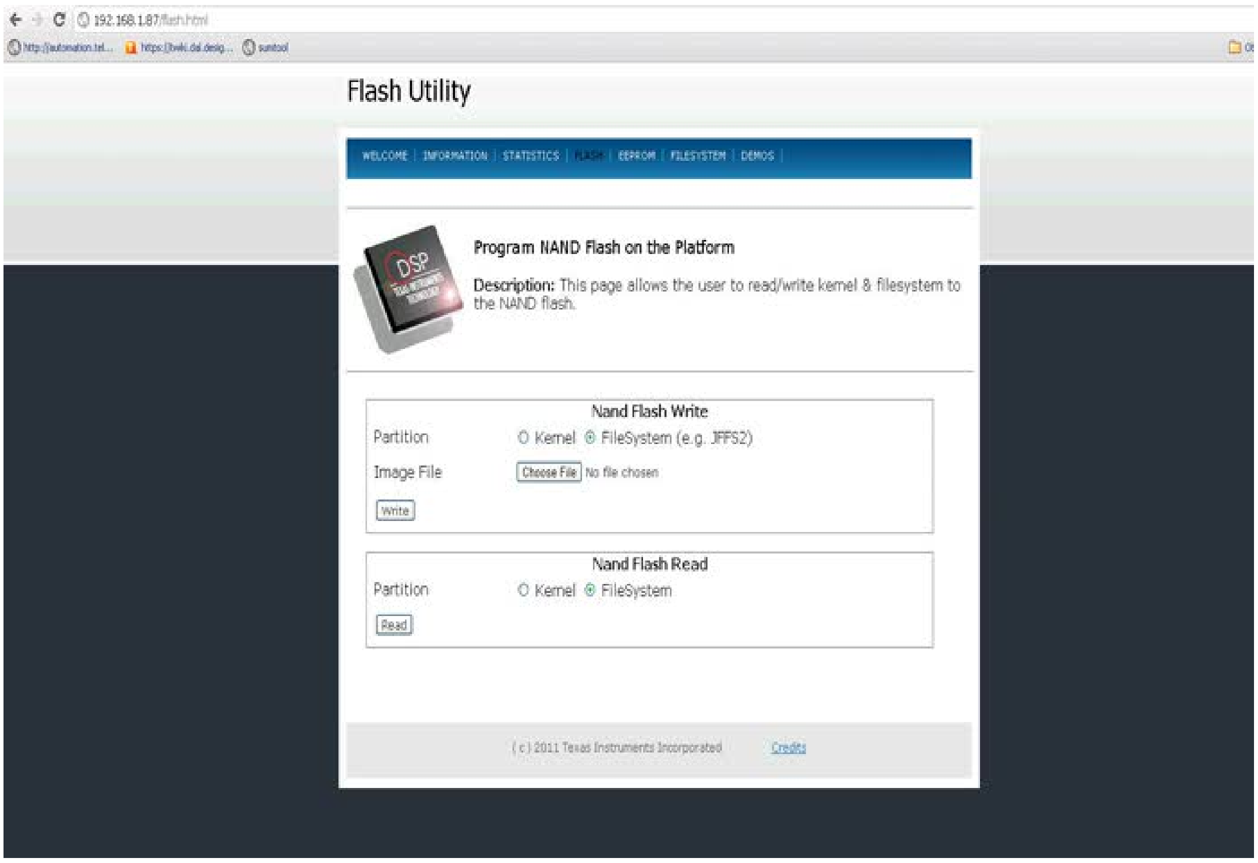
This provides information about memory and CPU usage

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:-


192.168.1.87/flash.html
WELCOME | INFORMATION | STATISTICS | FLASH | EEPROM | FILESYSTEM | DEMOS
DSP

Flash Utility

Program NAND Flash on the Platform

Description: This page allows the user to read/write kernel & filesystem to the NAND flash.

Nand Flash Write

Partition	Kernel [✓ FileSystem (e.g. JFFS2)]
Image File	Choose File No file chosen

Write

Nand Flash Read

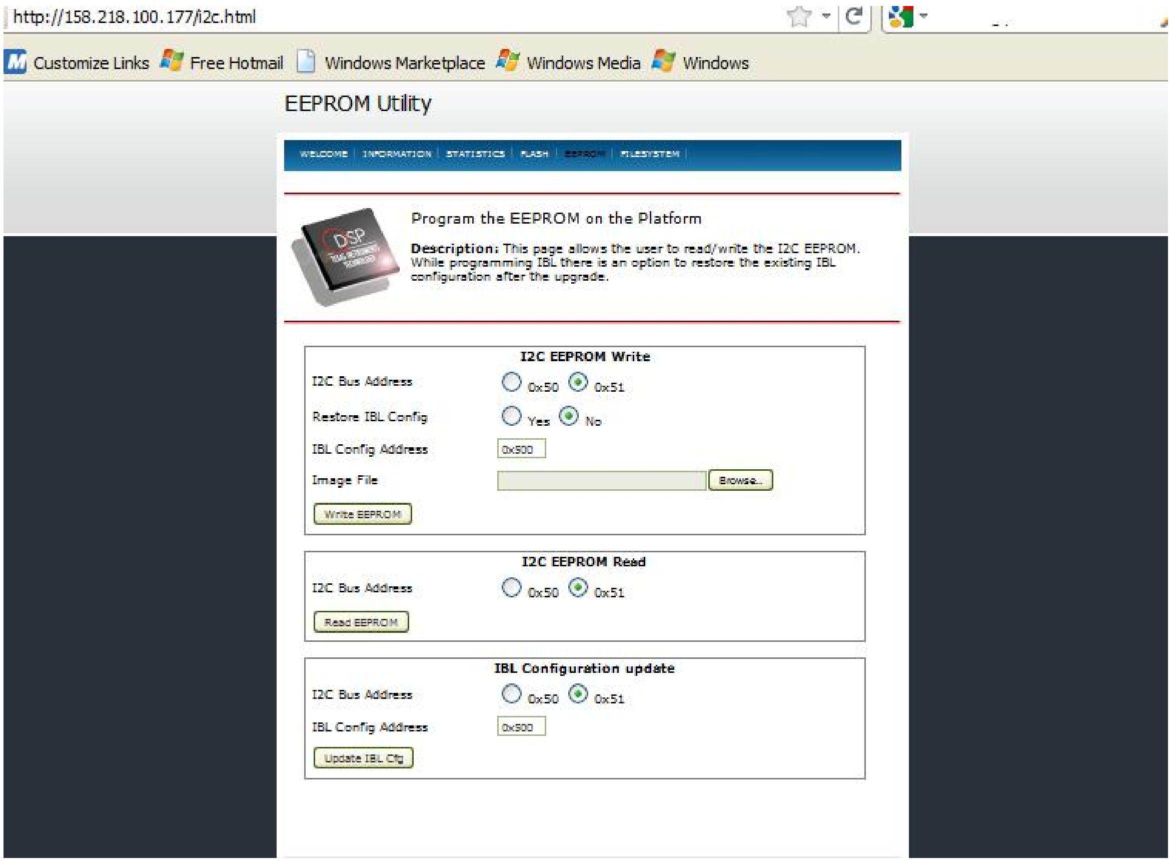
Partition	Kernel [✓ FileSystem]

Read

(c) 2011 Texas Instruments Incorporated Credits

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.



EEPROM Utility

Program the EEPROM on the Platform

Description: This page allows the user to read/write the I2C EEPROM. While programming IBL there is an option to restore the existing IBL configuration after the upgrade.

I2C EEPROM Write

I2C Bus Address	0x50 [✓ 0x51]
Restore IBL Config	Yes [✓ No]
IBL Config Address	0x500
Image File	Browse...

Write EEPROM

I2C EEPROM Read

I2C Bus Address	0x50 [✓ 0x51]

Read EEPROM

IBL Configuration update

I2C Bus Address	0x50 [✓ 0x51]
IBL Config Address	0x500

Update IBL Cfg

This provides following capabilities to user:-

- I2C EEPROM Write

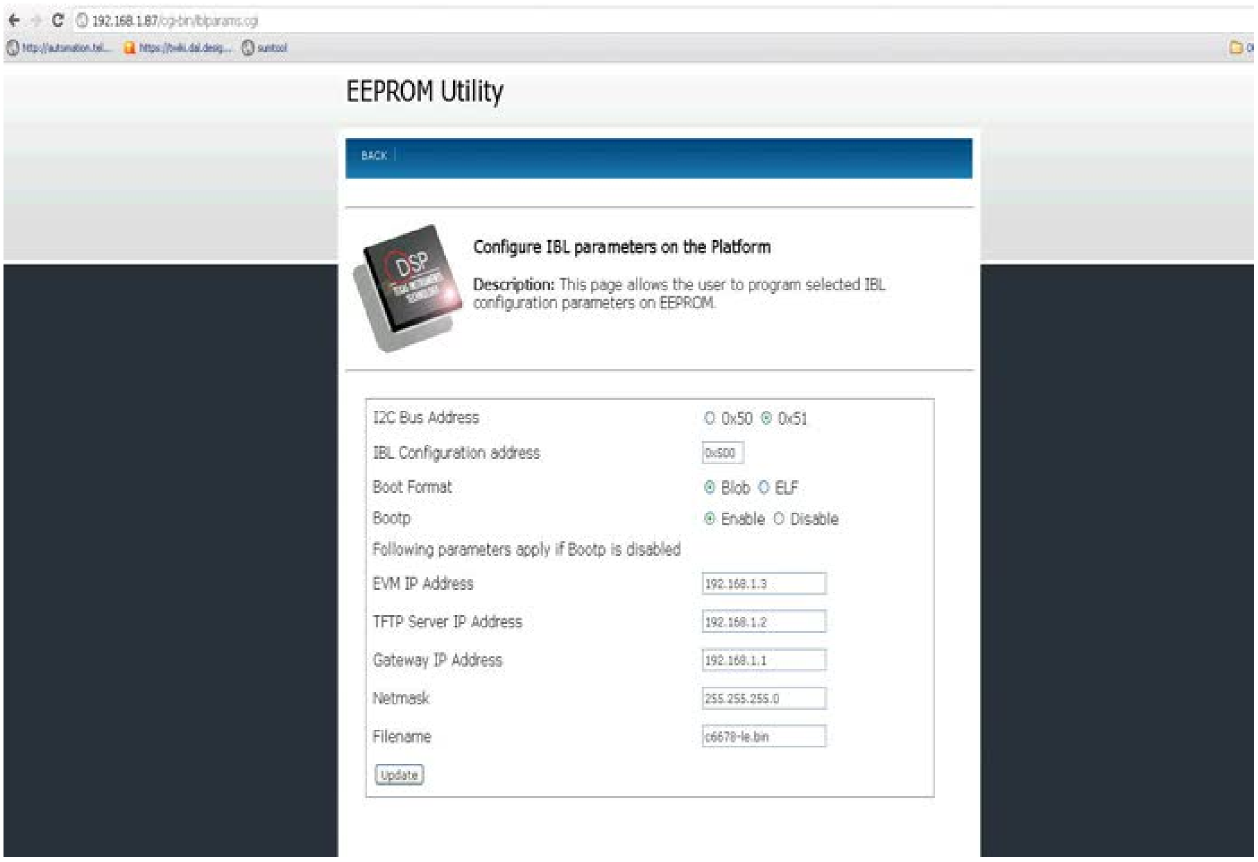
The EEPROM at i2c address 0x50 is by default programmed with POST image and at address x51 is programmed with IBL image. User will be able to program either of these using this interface. While programming the IBL image, there is an option to restore the existing IBL configuration.

- 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.



EEPROM Utility

Configure IBL parameters on the Platform

Description: This page allows the user to program selected IBL configuration parameters on EEPROM.

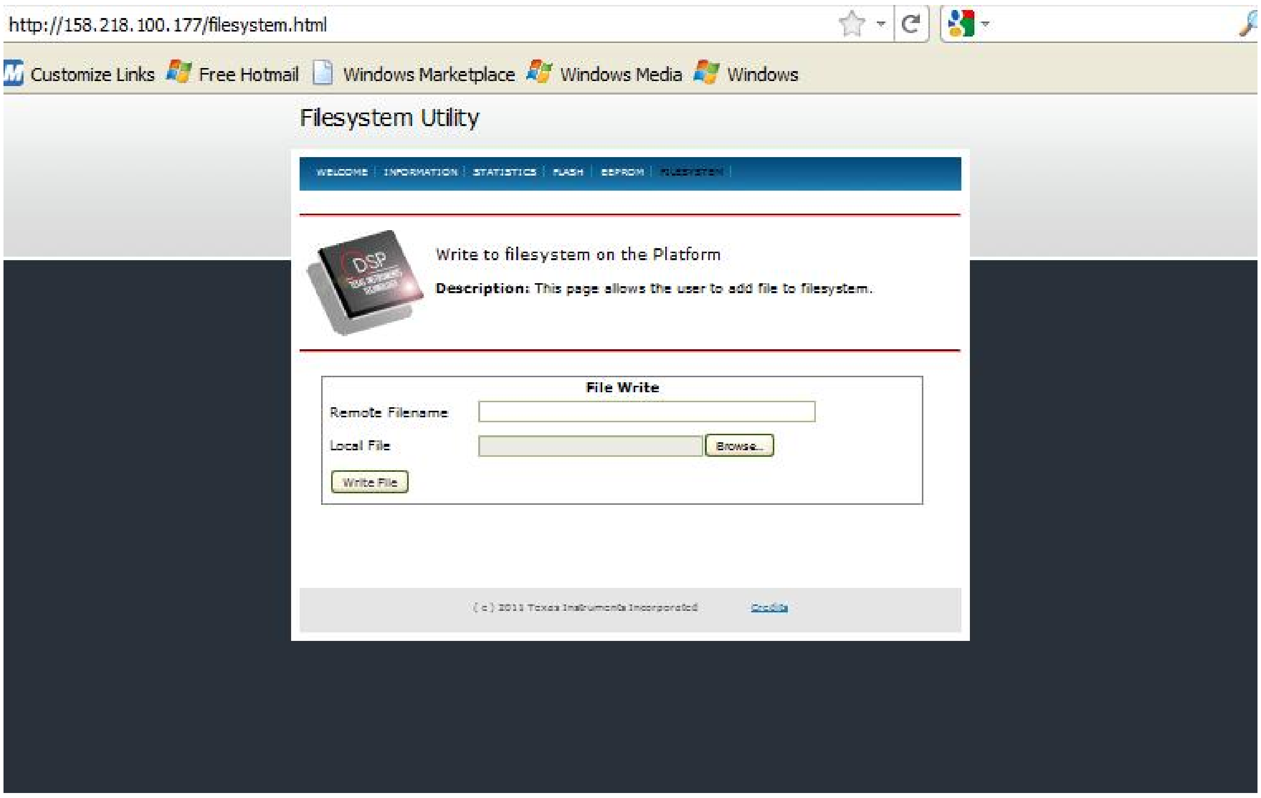
I2C Bus Address	0x50 [✓ 0x51]
IBL Configuration address	0x500
Boot Format	[✓ Blob] ELF
Bootp	[✓ Enable] Disable
Following parameters apply if Bootp is disabled	
EVM IP Address	192.168.1.3
TFTP Server IP Address	192.168.1.2
Gateway IP Address	192.168.1.1
Netmask	255.255.255.0
Filename	c6678-le.bin

Update

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.



http://158.218.100.177/filesystem.html

Filesystem Utility

WELCOME | INFORMATION | STATISTICS | FLASH | EEPROM | FILESYSTEM |

Write to filesystem on the Platform

Description: This page allows the user to add file to filesystem.

File Write

Remote Filename

Local File

(c) 2011 Texas Instruments Incorporated   Credits

Building Linux kernel and filesystem from source

A tar ball of the source code is provided at the ti.com Software Download Site [6] or the latest Linux MCSDK release can be downloaded from linux-c6x.org File Download Area [4]

The steps to build kernel and filesystem from source code is given in the Release Notes [7]

Support

Please send any support related questions to the TI e2e forum [8]. Please use the tag **Linux-c6x** on all questions.

References

[1] http://linux-c6x.org/files/releases/linux-c6x-2.0-ga/linux-c6x-2.0.0.63-release-notes.pdf
[2] http://linux-c6x.org/wiki/index.php/Linux-c6x_Users_Guide_2.x
[3] http://ttssh2.sourceforge.jp/index.html.en
[4] http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/
[5] http://www.linux-c6x.org/wiki/index.php/How_to_setup_an_NFS_filesystem%3F
[6] http://software-dl.ti.com/sdoemb/sdoemb_public_sw/linux_mcsdk/02_00_00_63/index_FDS.html
[7] http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/linux-c6x-2.0.0.63-release-notes.pdf
[8] http://e2e.ti.com/support/embedded/f/354.aspx

Article Sources and Contributors

Linux-c6x-2.0-GA User Guide *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=126072 *Contributors*: BillMills, Spaulraj

Image Sources, Licenses and Contributors

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:L-c6x-flash-698x476.jpeg *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:L-c6x-flash-698x476.jpeg *License*: unknown *Contributors*: BillMills
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:L-c6x-eeprom-698x476.jpeg *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:L-c6x-eeprom-698x476.jpeg *License*: unknown *Contributors*: BillMills
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