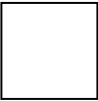


Table of Contents

- I. Introduction 5
 - A. Objective..... 5
- II. Setting-up the Host Computers..... 5
 - A. Tools..... 5
 - B. Preparation 6
- III. Preparing your Windows Host..... 12
 - A. Installing the USB serial driver for Windows..... 12
 - B. Installing and configuring Tera Term 14
- IV. Uboot..... 15
 - A. Uboot introduction..... 15
 - B. Preparation 15
 - C. Compiling the image 16
- V. Kernel 17
 - A. Linux Kernel Introduction..... 17
 - B. Preparation 17
 - C. Compiling the image and modules..... 18
- VI. Filesystem 21
 - A. Embedded Filesystem introduction 21
 - B. Preparation 21
 - B.1 Installing Softwares..... 21
 - B.2 Setup for NFS boot up 21
 - C. Compiling Tarball..... 23
 - D. Compiling UBIFS image 23
- VII. Flashing procedures..... 25
 - A. Type of flashing techniques 25
 - 1. Reflashing Method: "via OpenOCD" 25



	Software Development Kit v6.1 - NIMBUS		
			Page: 2/34
			A4



- 2. Reflashing Method : "Interactive" 29
- 2.1 Menu-based Reflashing..... 29
- 2.1 Reflashing Method: "via U-boot console" 30
- VIII. Software Features..... 31
- 1. LEDS Indicators..... 31
- 1.1 Controlling LED'S 31
- 1.2 Ethernet Ports status LED 32
- 2. CPUinfo..... 33
- 3. Test Scripts..... 33
- 4. Accessing uboot environment variables via userspace 34

Table of Figures

Figure 1: Uboot binary is ready 16

Figure 2: Kernel configuration..... 18

Figure 3: Save the configuration 19

Figure 4: Image is ready 20

Figure 5: Stop auto-boot 22

Figure 6: OpenOCD Files 25

Figure 7: U-boot reflashing complete..... 26

Figure 8: Reflashing the Plugcomputer (Filesystems and kernels)..... 27

Figure 9: Halt for 10 seconds before reflashing 27

Figure 10: Login..... 28

Figure 11: Interactive Reflashing 29

Figure 12: Reflash Manufacturing..... 30

Figure 13: CPUinfo 33



	Software Development Kit v6.1 - NIMBUS		
			Page: 4/34
			A4

B. Preparation

Note: Please be aware that most of the activities that shall be performed on the Linux host require "root privileges". Though this could be easily done, the developer should take precaution since he/she may accidentally modify the system files which could lead to unwanted problems.

❖ Installing Arm Compiler

- This should be installed in the developer's Linux Host Computer if he/she is planning to compile his/her own binaries.
- Copy the GCC compiler tarball "gcc.tar.bz2" from the DISC (/TOOLS.zip) and put it to the "(/home)" directory of your Linux Host. Decompress the tarball by issuing the following commands.

```
# cd /home
# tar -xvjpf gcc.tar.bz2
```

❖ Installing the USB-to-Serial Chip Driver for Linux

- Connect the mini-USB cable to the plugcomputer and connect it to the Linux Host Computer's USB port.
- Power-up the plugcomputer and issue "dmesg" on the terminal console

```
# dmesg
```

```
Usb 5-2: new full speed USB device using uhci_hcd and address 4
usb 5-2: configuration #1 chosen from 1 choice
ftdi_sio 5-2:1.0: FTDI USB Serial Device converter detected
usb 5-2: Detected FT2232C
usb 5-2: FTDI USB Serial Device converter now attached to ttyUSB0
ftdi_sio 5-2:1.1: FTDI USB Serial Device converter detected
usb 5-2: Detected FT2232C
usb 5-2: FTDI USB Serial Device converter now attached to
ttyUSB0
```

- Note: Some Linux Distributions like Ubuntu 10.10 Desktop edition already has built-in drivers for FTDI.
- If you need to manually install the FTDI driver, proceed with the instructions below:

Software Development Kit v6.1 - NIMBUS

- Copy the driver "ftdi_sio.tar.gz" from the SDK DISC (/TOOLS.zip) and put it to the "(/home/)" directory of your Linux PC Host
- Decompress the file tarball by issuing the following commands.

```
# cd /home/
# tar -xvzpf ftdi_sio.tar.gz
```

- Go to the directory created <ftdi_sio> and issue the commands

```
# cd ftdi_sio
# make
```

- Connect the Plugcomputer to the Linux PC Host using the mini-USB cable.
- Power-up the PlugComputer and check whether the device is recognized.

It should be able to see the following information after issuing the command below.

```
# dmesg
```

```
usb 5-2: new full speed USB device using uhci_hcd and address 4
usb 5-2: configuration #1 chosen from 1 choice
ftdi_sio 5-2:1.0: FTDI USB Serial Device converter detected
usb 5-2: Detected FT2232C
usb 5-2: FTDI USB Serial Device converter now attached to ttyUSB0
ftdi_sio 5-2:1.1: FTDI USB Serial Device converter detected
usb 5-2: Detected FT2232C
usb 5-2: FTDI USB Serial Device converter now attached to
ttyUSB0
```

- If the information above were not shown, add the USB device ID of the IONICS JTAG module (vendor=0x1c0c; product=0x0102) by issuing the command

```
# modprobe ftdi_sio vendor=0x1c0c product=0x0102
```

- Reconnect the USB cable of the JTAG to the Linux host and check whether the device is detected using the "dmesg" command.

- Make a shortcut for this method by creating a file named "ftdi_sio" at the "/etc/modprobe.d/" directory:

```
# vi /etc/modprobe.d/ftdi_sio
```

- Write the following to the file:

```
# options ftdi_sio vendor=0x1c0c product=0x0102
```

- Issue the command after every host reboot:

```
# modprobe ftdi_sio
```

❖ Installing Minicom

- Install minicom from linux repository by issuing the command

```
# apt-get install minicom
```

- Configure minicom after installation

```
# minicom -s
```

- Select "Serial Port Setup" from the menu and ensure that the following settings are loaded:

Serial Device	= /dev/ttyUSB0
Bps/Par/Bits	= 115200 8N1
HW Flow Ctrl	= No
SW Flow Ctrl	= No

- Select "Save setup as dfl" and "Exit", the developer should be able to see the serial console output after doing so. (Note: the developer may need to check the serial device location by issuing "dmesg" to your command line upon connecting the JTAG. Replace the "Serial Device" value accordingly.)



❖ Additional tools for Compiling

- Copy the "mkimage" binary from the DISC (/TOOLS.zip) and put it to the "/usr/bin" directory of your Linux PC Host.
- Some Linux distros don't have the curses.h header installed by default which is needed in making configuration menus. Install curses.h by typing the command:

```
# apt-get install libncurses5-dev
```

❖ Installing NFS and TFTP Server on your Linux Host

The following procedure will enable you to boot your plugcomputer via network connection. This is essential if you are planning to recompile your own filesystem binary for the plug.

- Setting-up the Network File System (NFS) Server.
 - ✓ Install the NFS through apt-get.

```
# apt -get install nfs-kernel-server nfs-common portmap
```

- ✓ Make a directory named i.e. "/home/myFilesystem".

```
# mkdir /home/myFilesystem
```

- ✓ Copy the filesystem tarball from the DISC (/FILESYSTEM/SOURCE /rootfs.tar.gz) that will be used for the NFS boot and put it to the "/home/myFilesystem" directory in your Linux PC Host.

- ✓ Decompress the rootfilesystem.

```
# cd /home/myFilesystem
# tar -xvzf rootfs.tar.gz
# rm rootfs.tar.gz
```

Software Development Kit v6.1 - NIMBUS			
		Page: 9/34	A4

- ✓ Open the "/etc/exports" file.

```
# vi /etc/exports
```

- ✓ Add the following lines to the "exports" file.

```
/home/myFilesystem/ * (rw, no_root_squash)
/tftpboot/ *(rw, no_root_squash)
```

- ✓ Activate the changes on "/etc/exports" by issuing the command:

```
# /etc/init.d/nfs-kernel-server restart
```

- ✓ Open the "/etc/network/interfaces" file.

```
# vi /etc/network/interfaces
```

- ✓ Edit and add a static IP address value:

```
iface eth0 inet static
address 192.168.0.3
netmask 255.255.255.0
gateway 192.168.0.1
```

➤ Setting-up the Trivial File Transfer Protocol (TFTP) Server

This should be installed if the developer is planning to boot the PlugComputer over the network.



Software Development Kit v6.1 - NIMBUS			
		Page: 10/34	A4

- ✓ Install the xinetd, tftpd, and tftp packages through apt-get

```
# apt-get install xinetd tftpd tftp
```

- ✓ Create file "/etc/xinetd.d/tftp" and put the following:

```
# vi /etc/xinetd.d/tftp
```

```
service tftp
{
    protocol          = udp
    socket_type       = dgram
    wait              = yes
    user              = nobody
    server            = /usr/sbin/in.tftpd
    server_args       = -s /tftpboot
    disable           = no
}
```

- ✓ Make a directory named "/tftpboot" at your Linux PC Host and change the permission and owner of the directory:

```
# mkdir /tftpboot
# chmod -R 777 /tftpboot
# chown -R nobody /tftpboot
```

- ✓ You may start the tftp through xinetd by issuing the command.

```
# /etc/init.d/xinetd restart
```

III. Preparing your Windows Host

Note: *The following instructions will prepare your Window's host in order to establish a serial communication with the PlugComputer.*

A. Installing the USB serial driver for Windows

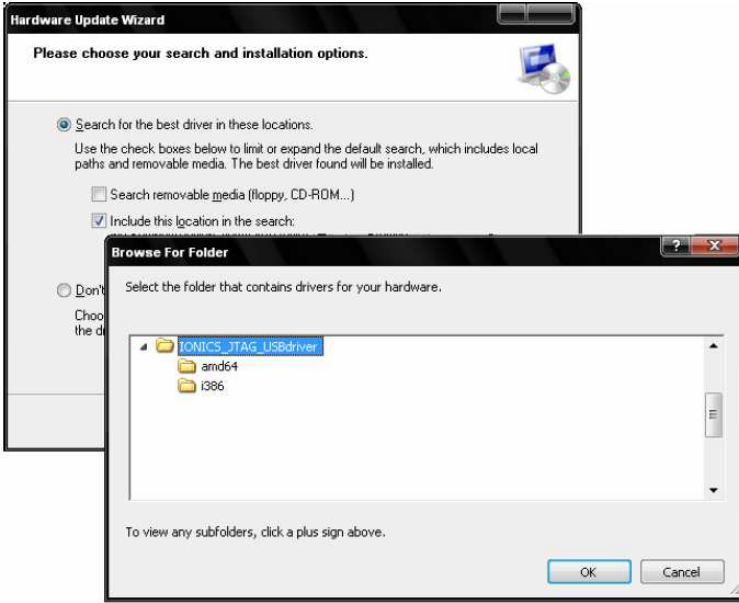
- Copy the "IONICS_JTAG_USBdriver.zip" from the DISC (/TOOLS.zip) to your Windows host.
- Unzip the file
- Connect the PlugComputer into the Windows host using the mini USB cable
- Power-up the PlugComputer.
- Please refer to the instructions below on how to install the driver.



- Install the new hardware by using the Hardware Wizard.
- Select "**Install from a list or specific location**" and click next.

	Software Development Kit v6.1 - NIMBUS		
			Page: 12/34
			A4





- Browse your windows host to locate the directory “IONICS_JTAG_USBdriver” and click **OK**.

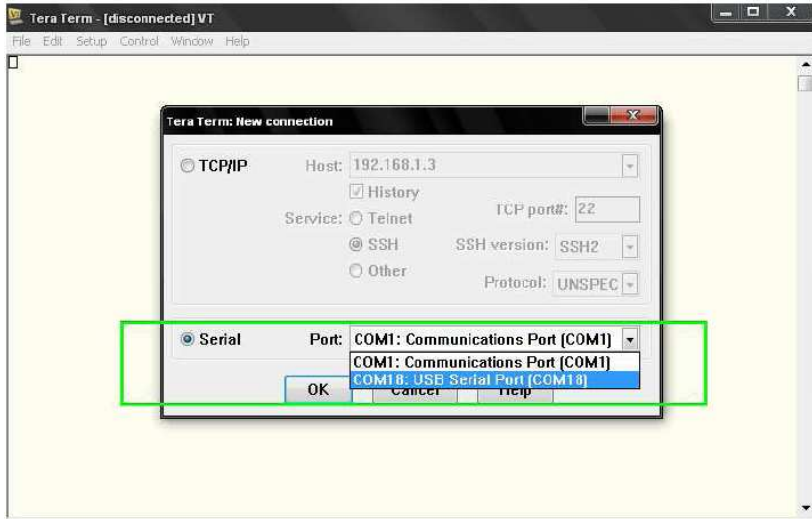


- Continue the installation when the prompt box appears.
- Repeat the installation procedure for **USB Serial Converter A**
- Repeat the installation procedure for **USB Serial Port**.

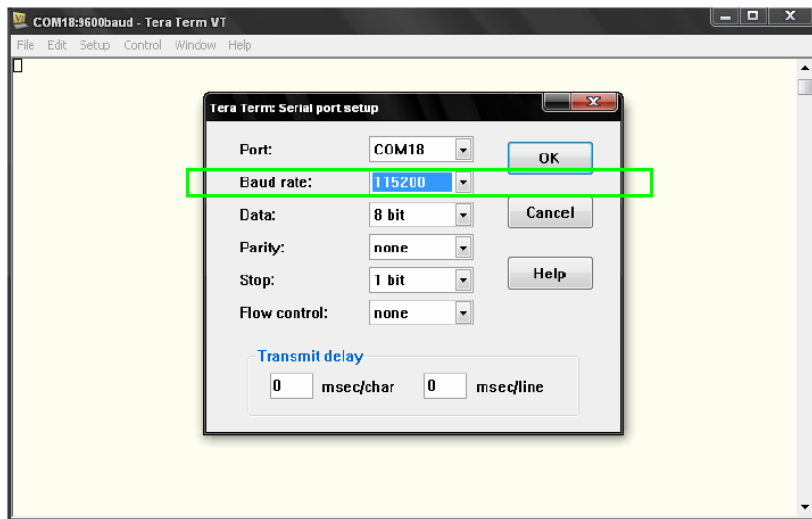
Software Development Kit v6.1 - NIMBUS			
		Page: 13/34	A4

B. Installing and configuring Tera Term

- Copy the “teraterm-4.60.exe” from the DISC (/TOOLS.zip) to your Windows host
- Install TeraTerm
- After installation, attach the PlugComputer to your Windows host and run TeraTerm
- Configure the terminal by following the instructions below:



- Open TeraTerm and select “**Serial**” connection
- Select the **USB Serial Port**



- Go to **Setup->Serial Port** and configure the baud rate to 115200.
- Click **OK** and you should be able to see the logs generated by the plugcomputer.

IV. Uboot

A. Uboot introduction

Is a universal boot loader for embedded boards based on PowerPC, ARM, MIP and several other processors, which can be installed in a boot ROM and used to initialize and test the hardware or to download and run OS and applications.

B. Preparation

- Make a working directory at your Linux Host:

```
# mkdir /home/uboot
```

- Copy u-boot source from the DISC (/BOOTLOADER/SOURCE/u-boot-marvell tar.bz2) into your Linux PC Host or download the source code from DENX

<http://git.denx.de/?p=u-boot/u-boot-marvell.git;a=summary>

- Untar the u-boot source:

```
# cd /home/uboot  
# tar xvjf u-boot-marvell.tar.bz2
```

- Copy and execute the uboot patch file from the DISC (/BOOTLOADER/PATCH/IONICS-NIME0-uboot.patch) into the u-boot source directory to update the codes to contain IONICS configurations.

- Check patch file before replacing:

```
# cd /home/uboot/ub-boot-marvell  
# patch -p1 --dry-run < IONICS-NIME0-uboot.patch
```

- Execute patch by issuing:

```
# patch -p1 < IONICS-NIME0-uboot.patch
```

Software Development Kit v6.1 - NIMBUS

C. Compiling the image

- Build the u-boot image:

```
# make clean
# make mrproper
# make sheevaplug_config
# make -s u-boot.kwb ARCH=arm
CROSS_COMPILE=/home/gcc/bin/arm-none-linux-gnueabi-
```

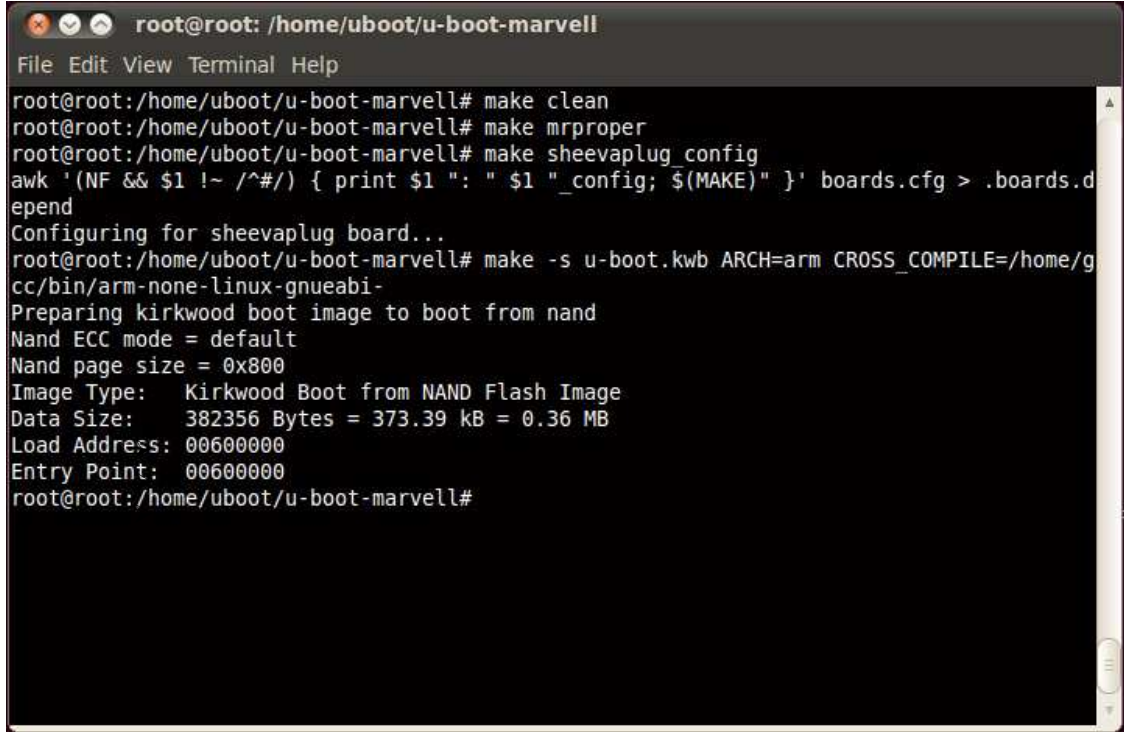


Figure 1: Uboot binary is ready

- Please see Chapter 7 (Flashing Procedures) on how to flash the compiled binary.

For Ubuntu users you may encounter an error regarding "MV_BOOTSIZE undefined" and may need to do the following changes to the shell before building the image:

```
# ls -l /bin/sh
# ln -sf /bin/bash /bin/sh
```


V. Kernel

A. Linux Kernel Introduction

Linux Kernel is the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system.

B. Preparation

- On the Linux PC Host create the directory "/home/kernel".

```
# mkdir /home/kernel
```

- Copy kernel source from the DISC (/KERNEL/SOURCE/linux-2.6.34.9.tar.bz2) into the "/home" directory of your Linux PC Host or download the source code from:

<http://www.kernel.org>

- Decompress the kernel source

```
# cd /home/kernel  
# tar xvjf linux-2.6.34.9.tar.bz2
```

- Copy and execute the kernel patch file from the DISC (/KERNEL/PATCH/IONICS-NIME0-kernel.patch) into the kernel source directory to update the codes to contain IONICS configurations.

- Check patch file before replacing:

```
# cd /home/kernel/linux-2.6.34.9  
# patch -p1 --dry-run < IONICS-NIME0-kernel.patch
```

- Execute patch by issuing:

```
# patch -p1 < IONICS-NIME0-kernel.patch
```

Software Development Kit v6.1 - NIMBUS

C. Compiling the image and modules

- Ensure that the source is in a clean state before starting the compile.

```
# make clean
# make mrproper
```

- Configure the kernel by copying the config file to .config

```
# cp arch/arm/configs/ionics_defconfig .config
```

- Enable additional kernel features thru menuconfig

```
# make ARCH=arm menuconfig
```

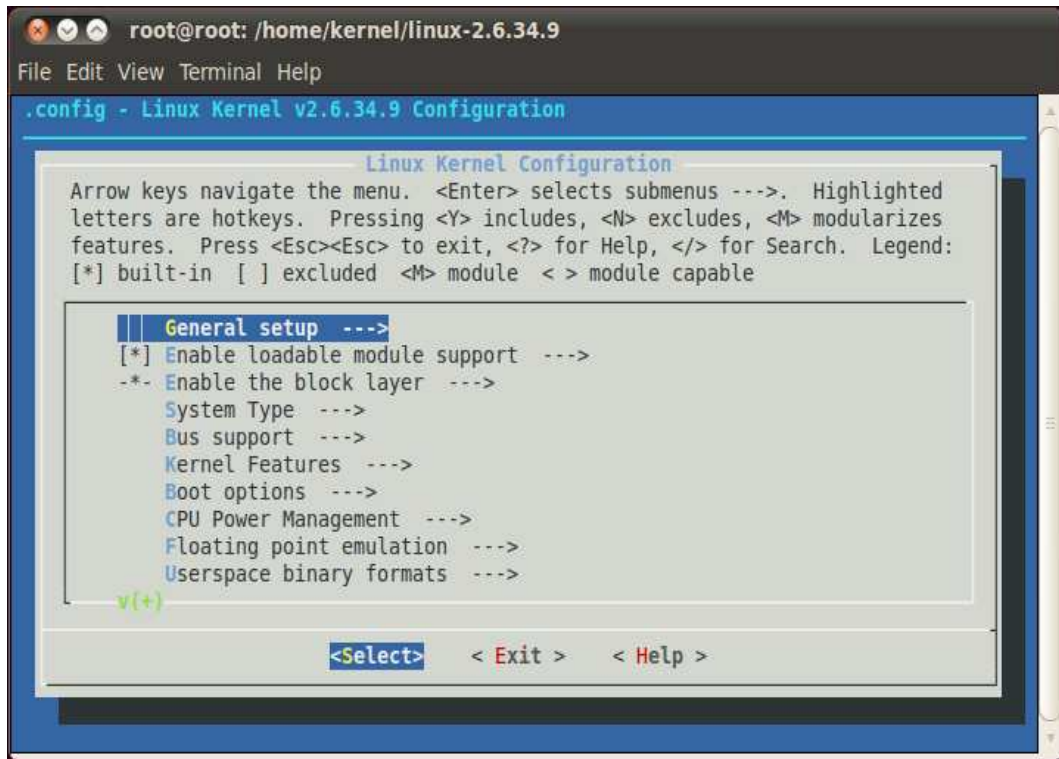


Figure 2: Kernel configuration

- Save the kernel configuration if you're done.

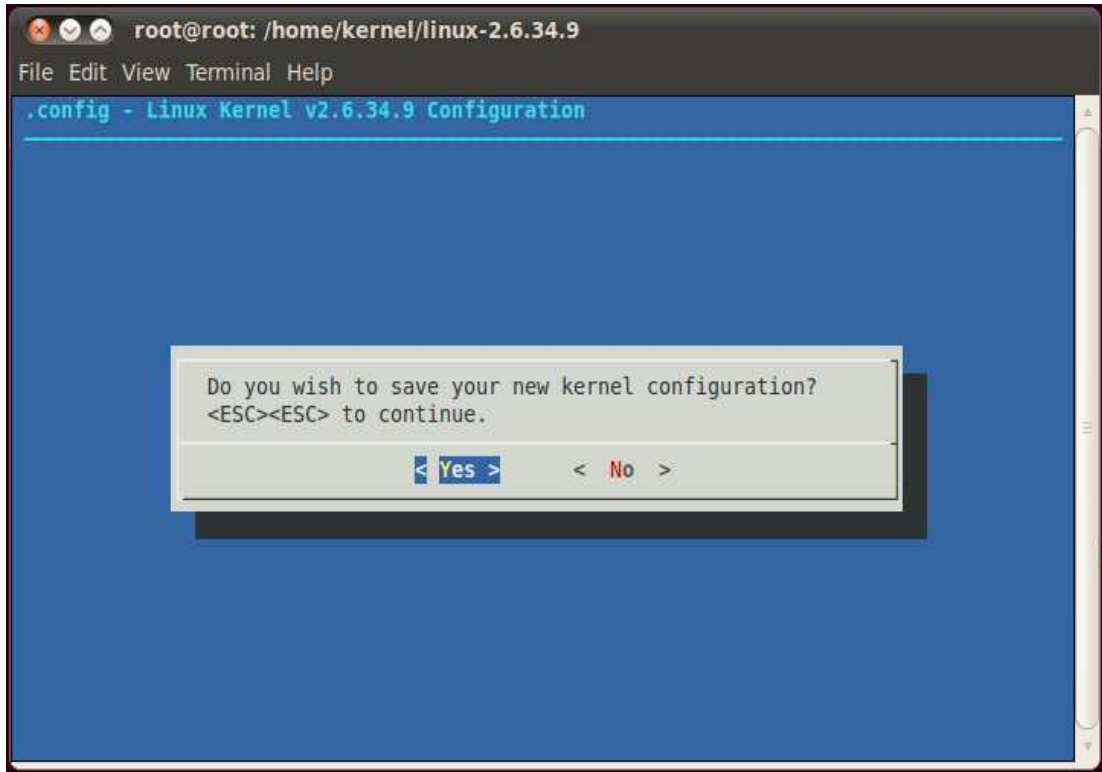


Figure 3: Save the configuration

- Compiling the kernel image:

```
# make -s uImage ARCH=arm CROSS_COMPILE=/home/gcc/bin/
arm-none-linux-gnueabi-
```



```

root@root: /home/kernel/linux-2.6.34.9
File Edit View Terminal Help
arch/arm/kernel/unwind.c:35:2: warning: #warning Change compiler or disable ARM_UNWIND
option.
drivers/pci/setup-bus.c: In function '_pci_bridge_assign_resources':
drivers/pci/setup-bus.c:706: warning: passing argument 1 of 'pdev_assign_resources_sort
ed' discards qualifiers from pointer target type
net/ipv4/netfilter/ip_tables.c: In function 'ipt_get_target_c':
net/ipv4/netfilter/ip_tables.c:213: warning: passing argument 1 of 'ipt_get_target' d
iscards qualifiers from pointer target type
net/ipv4/netfilter/ipt_ULOG.c: In function 'ipt_u_log_packet':
net/ipv4/netfilter/ipt_ULOG.c:214: warning: passing argument 1 of '_net_timestamp' dis
cards qualifiers from pointer target type
net/ipv4/netfilter/arp_tables.c: In function 'arpt_get_target_c':
net/ipv4/netfilter/arp_tables.c:239: warning: passing argument 1 of 'arpt_get_target' d
iscards qualifiers from pointer target type
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
Image Name: Linux-2.6.34.9
Created: Thu May 5 09:23:01 2011
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 2865344 Bytes = 2798.19 kB = 2.73 MB
Load Address: 0x00008000
Entry Point: 0x00008000
Image arch/arm/boot/uImage is ready
root@root: /home/kernel/linux-2.6.34.9#

```

Figure 4: Image is ready

- Copy the .config file back to the config file

```
# cp .config arch/arm/configs/ionics_defconfig
```

- Build the kernel modules

```
# make -s modules ARCH=arm CROSS_COMPILE=/home/gcc/bin/
arm-none-linux-gnueabi
```

```
# make -s modules_install ARCH=arm CROSS_COMPILE=/home/
gcc/bin/arm-none-linux-gnueabi INSTALL_MOD_PATH=<path>
```

```
# cd <path>
# tar -cvzpf modules.tar.gz lib/
```

- Please see Chapter 7 (Flashing Procedures) on how to re-flash your compiled image.

Software Development Kit v6.1 - NIMBUS			
		Page: 20/34	A4

VI. Filesystem

A. Embedded Filesystem introduction

Embedded file system which can be used with any type of storage device, is a high performance library that has been optimized for; minimum memory consumptions in RAM and ROM, high speed, and versatility.

B. Preparation

B.1 Installing Softwares

There are several ways on how the developer could install applications into the PlugComputer. One of basic ways is to connect the PlugComputer to the internet and access the open-source repositories.

Sample: installing "usbutils" to obtain "lsusb" command

```
# apt-get update  
# apt-get install <application>
```

B.2 Setup for NFS boot up

- Copy "uImage" from the DISC (/BINARIES.zip) to the "/tftpboot" directory of the Linux Host
- Access the plug computer via mini-USB cable
- Connect the plug computer and Linux PC Host to a network.
- Be sure to add an IP address that is a member of the network.
- Power up the PlugComputer access it via serial console. Stop auto-boot to access the uboot console.

```

swdesign3@root: ~
File Edit View Terminal Help
Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000 size 256MB
DRAM CS[1] base 0x10000000 size 256MB
DRAM Total size 512MB 16bit width
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB
Flash: 0 kB

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

USB 0: host mode
PEX 0: interface detected no Link.
Net: egiga0 [PRIME]
Hit any key to stop autoboot: 0
Marvell>>

```

Figure 5: Stop auto-boot

- Type the following command in the u-boot console to create the needed u-boot environment variables

```

> set ipaddr <enter static ipaddress for the plug>
> set serverip <enter ipaddress of the Linux PC Host>
> set image_name uImage
> set rootpath '/home/myFilesystem'
> set bootargs_root_nfs 'root=/dev/nfs rw'
> set bootcmd_nfs 'tftpboot 0x2000000 $(image_name); setenv
bootargs $(bootargs_root_nfs) $(mtdparts)
nfsroot=$(serverip): $(rootpath) ip=$(ipaddr): $(serverip);
bootm 0x2000000'
> saveenv

```

- Temporarily boot via NFS by issuing the following command

```

> run bootcmd_nfs

```

- After boot-up, you can now log-in using the following details

```
Login: root
Password: root
```

C. Compiling Tarball

- After finalizing the rootfs contents, boot-up your PlugComputer via network
- Please see Chapter VI B.2 "Setup for NFS boot up" for instructions on how to boot via network"
- Mount the NAND rootfs partition by issuing the following commands on the PlugComputer's linux console:

```
# ubiattach /dev/ubi_ctrl -m 3
# mount -t ubifs ubi0:rootfs /mnt
```

- Remove the modules

```
# rm -rf /mnt/lib/modules/<module name>
```

- Create the tarball

```
# cd /mnt
# tar -cvzpf ../rootfs.tar.gz *
```

- The resulting tarball would be located on the "/" directory

D. Compiling UBIFS image

- After finalizing the rootfs contents, boot-up your PlugComputer via network or USB stick.
- Please see Chapter VI B.2 "Setup for NFS boot up" for instructions on how to boot via network"
- Create "/home/myFilesytem/ubi.cfg" file and put the following:

```
# vi /home/myFilesytem/ubi.cfg
```

Software Development Kit v6.1 - NIMBUS

```
[ubifs]
mode=[ubi]
image=ubifs.img
vol_id=0
vol_size=256MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
```

- Mount the NAND rootfs partition by issuing the following commands on the PlugComputer's linux console:

```
# ubiattach /dev/ubi_ctrl -m 3
# mount -t ubifs ubi0:rootfs /mnt
```

- Remove the modules

```
# rm -rf /mnt/lib/modules/<>
```

- Create UBI image (This make take some time without progress display)

```
# mkfs.ubifs -v -r /mnt -m 2048 -e 129024 -c 4096 -o
ubifs.img -x zlib
```

```
# ubinize -o rootfs-ubi.img -m 2048 -p 128KiB -s 512
ubi.cfg
```

- The resulting ubifs image would be located on the "/" directory

VII. Flashing procedures

A. Type of flashing techniques

1. Reflashing Method: "via OpenOCD"

Warning: This will replace the current bootloader. This procedure may render the Plugcomputer unbootable. All identification (i.e. S/N, MAC address, etc) info will be lost. Do not do this procedure unless you know what you are doing

- Copy the OpenOCD installer tarball "OpenOCD.tar.bz2 " from the DISC (/TOOLS.zip) and extract it to the home directory "/home" of your Linux PC Host.

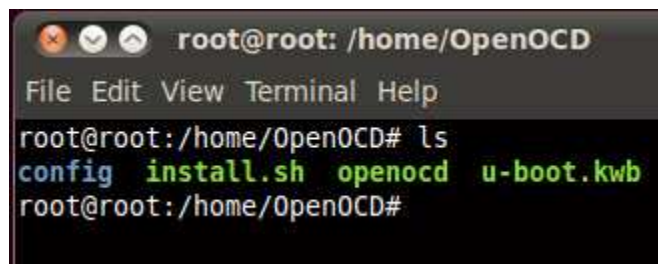
```
# cd /home
# tar -xvjf OpenOCD.tar.bz2
```

- Copy and unzip the binaries "BINARIES.zip" into a flash drive (uImage, rootfs-ubi.img, modules.tar.gz, initrd.usr, initrd.mfg, u-boot.kwb, checksum)

- Insert the flash drive into the Plugcomputer USB port

- Copy the uboot binary (u-boot.kwb) into the OpenOCD installer folder

```
# cp u-boot.kwb /home/OpenOCD/
# cd /home/OpenOCD/
# ls
```



A terminal window screenshot showing the directory listing of files in the /home/OpenOCD directory. The prompt is root@root: /home/OpenOCD. The terminal output shows: root@root:/home/OpenOCD# ls, followed by a list of files: config, install.sh, openocd, and u-boot.kwb. The prompt returns to root@root:/home/OpenOCD#.

Figure 6: OpenOCD Files

- Flash the firmware by issuing the command

```
# ./install.sh
```

- Wait until uboot reflashing is complete.

```

root@root: /home/OpenOCD
File Edit View Terminal Help

BUGS? Read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS

$URL: http://svn.berlios.de/svnroot/repos/openocd/trunk/src/openocd.c $
2000 kHz
dcc downloads are enabled
Info : JTAG tap: feroceon.cpu tap/device found: 0x20a023d3 (Manufacturer: 0x1e9, Part:
0x0a02, Version: 0x2)
Info : JTAG Tap/device matched
Error: unknown EmbeddedICE version (comms ctrl: 0x00000018)
Warn : no telnet port specified, using default port 4444
Warn : no gdb port specified, using default port 3333
Warn : no tcl port specified, using default port 6666
target state: halted
target halted in ARM state due to debug-request, current mode: Supervisor
cpsr: 0x000000d3 pc: 0xffff0000
MMU: disabled, D-Cache: disabled, I-Cache: disabled
0 0 1 0: 00052078
NAND flash device 'NAND 512MiB 3,3V 8-bit' found
successfully erased blocks 0 to 4 on NAND flash device 'NAND 512MiB 3,3V 8-bit'
wrote file u-boot.kwb to NAND flash 0 up to offset 0x0005d800 in 57.405437s
root@root:/home/OpenOCD#

```

Figure 7: Uboot reflashing complete

- In the Linux host issue the command below to access the plugcomputers console:

```
# minicom -c on
```

- The binaries in the thumbdrive will be automatically flashed to the plugcomputer i.e. filesystem and kernels



```
root@root: ~
File Edit View Terminal Help
DRAM: 512 MiB
NAND: 512 MiB
*** Warning - bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: egiga0
88E1116 Initialized on egiga0
Hit any key to stop autoboot: 0
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x60000 -- 100% complete.
Writing to Nand... done
(Re)start USB...
USB: Register 10011 NbrPorts 1
USB EHCI 1.00
scanning bus for devices... 2 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
reading uImage

2925320 bytes read
reading initrd.mfg
```

Figure 8: Reflashing the Plugcomputer (Filesystems and kernels)

- To access initrd.mfg console hit the keyboard else leave it as is to continue flashing

```
root@root: ~
File Edit View Terminal Help
Version: - 1.1
Flash Tool: - initrd.mfg
SW Package: - uImage, modules.tar.gz, rootfs-ubi.img, initrd.usr
Description: - this will perform complete NAND reflash!
              - will halt if there is any failure during checks

* Wait for 10 seconds to continue or press ENTER to stop *
=====

scsi 0:0:0:0: Direct-Access  Intuix  U3                6.17 PQ: 0 ANSI: 0 CCS
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] 997375 512-byte logical blocks: (510 MB/486 MiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Asking for cache data failed
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Asking for cache data failed
sd 0:0:0:0: [sda] Assuming drive cache: write through
   sda: sda1
sd 0:0:0:0: [sda] Asking for cache data failed
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

Figure 9: Halt for 10 seconds before reflashing

Software Development Kit v6.1 - NIMBUS			
		Page: 27/34	A4

- After reflashing, the Plugcomputer will automatically reboot. After boot-up, you can now log-in using the following details

Login: root
Password: root

```

root@root: ~
File Edit View Terminal Help
Starting remaining crypto disks...done.
Cleaning up ifupdown...
Setting up networking...
Loading kernel modules...done.
Checking file systems...fsck from util-linux-ng 2.17.2
done.
Mounting local filesystems...done.
Cleaning up temporary files...
Configuring network interfaces...done.
Cleaning up temporary files...
Setting kernel variables ...done.
INIT: Entering runlevel: 2
Using makefile-style concurrent boot in runlevel 2.
Starting enhanced syslogd: rsyslogd.
Starting periodic command scheduler: cron.
Starting OpenBSD Secure Shell server: sshdNET: Registered protocol family 10
ADDRCONF(NETDEV_UP): eth0: link is not ready
sshd (1037): /proc/1037/oom_adj is deprecated, please use /proc/1037/oom_score_
.

Debian GNU/Linux 6.0 debian ttyS0
debian login: root
Password: █

```

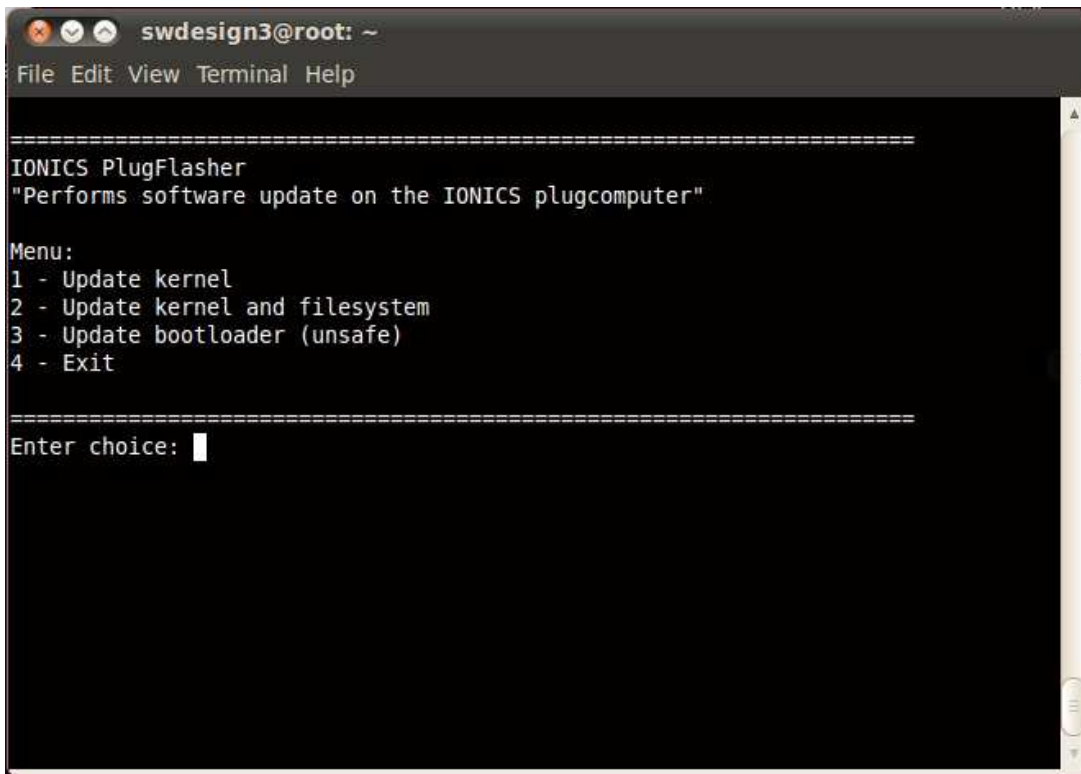
Figure 10: Login

2. Reflashing Method : "Interactive"

2.1 Menu-based Reflashing

- Copy and unzip the binaries "BINARIES.zip" into a flash drive (uImage, rootfs-ubi.img, modules.tar.gz, initrd.usr, initrd.mfg, u-boot.kwb, checksum)
- Power up the Plugcomputer
- Connect the flash drive to the Plugcomputers USB port
- Select different types of software updates from the menu by issuing:

```
# ./root/setup/plugflasher/reflash_interactive
```



```
swdesign3@root: ~
File Edit View Terminal Help

=====
IONICS PlugFlasher
"Performs software update on the IONICS plugcomputer"

Menu:
1 - Update kernel
2 - Update kernel and filesystem
3 - Update bootloader (unsafe)
4 - Exit

=====
Enter choice: █
```

Figure 11: Interactive Reflashing

2.1 Reflashing Method: "via U-boot console"

- Copy and unzip the binaries "BINARIES.zip" into a flash drive (uImage, rootfs-ubi.img, modules.tar.gz, initrd.usr, initrd.mfg, u-boot.kwb, checksum)
- Insert the flash drive into the Plugcomputer USB port
- Boot-up the PlugComputer and stop auto-boot to access the u-boot console
- Issue "run reflash_usr" (This will preserve the plug test information), else
- Issue "run reflash_mfg" (Alternate method but will erase all information on the NAND)

```

swdesign3@root: ~
File Edit View Terminal Help

Welcome to minicom 2.4

OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys

In:  serial
Out: serial
Err: serial
Net: egiga0
88E1116 Initialized on egiga0
Hit any key to stop autoboot: 0
Marvell>> AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
Unknown command 'AT' - try 'help'
Marvell>>
Marvell>> run reflash_mfg
  
```

Figure 12: Reflash Manufacturing

- Wait until Reflashing is done

VIII. Software Features

1. LEDS Indicators

1.1 Controlling LED'S

- Options
 - nand-disk
 - mmc0
 - timer
 - ide-disk
 - heartbeat
 - default-on
- LED's assignment
 - led1 - green
 - led2 - red

- To control LEDS issue in the Plugcomputers console:

Syntax:

```
# echo <options> > /sys/class/leds/<LED's assignment>/trigger
```

Sample:

```
# echo heartbeat > /sys/class/leds/led1/trigger
```

- You should be able to see the green LED blinking

1.2 Ethernet Ports status LED

The table below shows the NIMBUS' Ethernet port LED activities during different connection speed:

	Bootloader Version
	Marvell-Open Uboot
10 Mbps	orange LED only
100 Mbps	green and orange LEDs lights up
1000Mbps	green LED only

To verify the table above, the developer could force the connection speed to a different setting by doing the following:

- Force the ethernet connection speed to 100 Mbps (full-duplex) by
mii-tool -F 100baseTx-FD
- Force the ethernet connection speed to 100 Mbps (half-duplex) by
mii-tool -F 100baseTx-HD
- Force the ethernet connection speed to 10 Mbps (full-duplex) by issuing
mii-tool -F 10baseT-FD
- Force the ethernet connection speed to 10 Mbps (half-duplex) by issuing
mii-tool -F 10baseT-HD

Please be advised that the Ethernet port on the Plugcomputer still has auto-negotiating features. This means that when the unit is booted-up and is connected to a 1000 Mbps connection the green LED will light-up.

2. CPUinfo

- To see identification information about your Plugcomputer, issue:

```
# cat /proc/cpuinfo
```



```
root@root: /home/swdesign3
File Edit View Terminal Help
root@debian:~# cat /proc/cpuinfo
Processor       : Feroceon 88FR131 rev 1 (v5l)
BogoMIPS       : 1192.75
Features        : swp half thumb fastmult edsp
CPU implementer : 0x56
CPU architecture: 5TE
CPU variant     : 0x2
CPU part       : 0x131
CPU revision   : 1

Hardware       : PlugComputer
Manufacturer   : IONICS-EMS Inc.
Serial        : NB0500000091
Part          : 84SG100SGWZWBX
MAC           : 00:26:db:00:00:00
Model         : Nimbus
Revision      : E0
Firmware      : FIRMWARE-IONICS-NIME0-release-1.0-p0
Zwave API ver. : not installed
Zwave home ID : not installed
root@debian:~#
```

Figure 13: CPUinfo

- You should be able to see the identifications of your Plugcomputer, such as MAC address, Release package used, hardware model and etc.

3. Test Scripts

- `/root/setup/test/persistent_off` – moves the `75-persistent-net-generator.rules` from `/lib/udev/rules.d/` to `/lib/udev/rules.d/disabled/`; thus disabling the `persistent-net-generator`; for testing this script must be run; by default the system is `persistent_off` and should be tuned-on after test activities.
- `/root/setup/test/persistent_on` – moves the `75-persistent-net-generator.rules` from `/lib/udev/rules.d/disabled` to `/lib/udev/rules.d/`; thus enabling the `persistent-net-generator`; for deployment this script must be run
- `/root/setup/init_plugininfo` – mounts the `plugininfo` partition; `/plugininfo` directory contains all the plug test information.



4. Accessing uboot environment variables via userspace

- Create the file "/etc/fw_env.config" and put the following

```
# vi /etc/fw_env.config
```

```
/dev/mtd0 0x60000 0x20000 0x20000
```

- Printing uboot variable values

- Printing all environment variables at once

```
# fw_printenv
```

- Printing specific environment variables

Syntax:

```
# fw_printenv <environment variablename>
```

Sample1: Printing MAC address

```
# fw_printenv ethaddr
```

Sample2: Printing serial number

```
# fw_printenv ionicsplug_serial
```

- Setting uboot variables

- Syntax:

```
# fw_setenv <environment variablename> <value>
```

Sample 1: Setting serial number

```
# fw_setenv ionicsplug_serial NB0500000091
```

Sample 2: Setting Ethernet address

```
# fw_setenv ionicsplug_part 84STSDKSWJZW0XC
```

Software Development Kit v6.1 - NIMBUS