AC456
Application Note
Running uClinux on SmartFusion2 Advanced Development Kit

Microsemi
Power Matters.™

Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

51900456. 1.0 10/16

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Running uClinux on SmartFusion2 Advanced Development Kit Board

This application note describes a specific design example on how to boot uClinux on SmartFusion2 Advanced Development Kit using a U-boot loader.

2.1 Introduction

SmartFusion2 System-on-Chip (SoC) field programmable gate array (FPGA) devices integrate a fourth generation flash-based FPGA fabric and an ARM Cortex-M3 processor. This application note covers the compilation, booting of U-Boot and uClinux for SmartFusion2 Advanced Development Kit.

U-Boot is a multi-functional open-source bootloader that allows the developer to either load an operating system from different devices or configure a low-level platform. By power-on or reset, SmartFusion2 runs U-Boot firmware from the on-chip eNVM and completes the basic initialization by storing the volatile data to the embedded SRAM of the microcontroller. U-Boot configures the memory controller to access the external RAM (DDR3) and Flash memory (SPI), then copies the Linux image from Flash to RAM and jumps to the Linux Kernel entry point in RAM. It is possible to interrupt the U-Boot output sequence by hitting a key, before U-Boot starts relocating the Linux image to RAM. However, assuming no operator intervention, U-Boot proceeds to boot Linux up as soon as possible.

The uClinux Kernel is based on the Linux Kernel (from kernel.org) along with Microsemi additions (board support packages (BSP) and drivers). In general, the uClinux for SmartFusion2 follows normal ARM Linux processes for building and running. The uClinux Kernel is configured to mount a root file system in the external RAM using the initramfs file system. Initramfs is populated with the required files and utilities at the Kernel build time, Which are simply linked into the Linux image. The Linux image installed on the board provides a demonstration of the basic shell and network capabilities of the SmartFusion2 Advanced Development Kit.

2.1.1 Boot Sequence

The following figure shows the components involved in Linux boot sequence.

Figure 1 • **Boot Sequence**

2.1.1.1 Boot Loader

Boot loader is U-Boot and it performs the following functionalities:

- Hardware initializations
- Load and start Kernel

2.1.1.2 Linux Kernel

Kernel image is uClinux and it performs the following functionalities:

- Drivers initializations
- Mount root file system

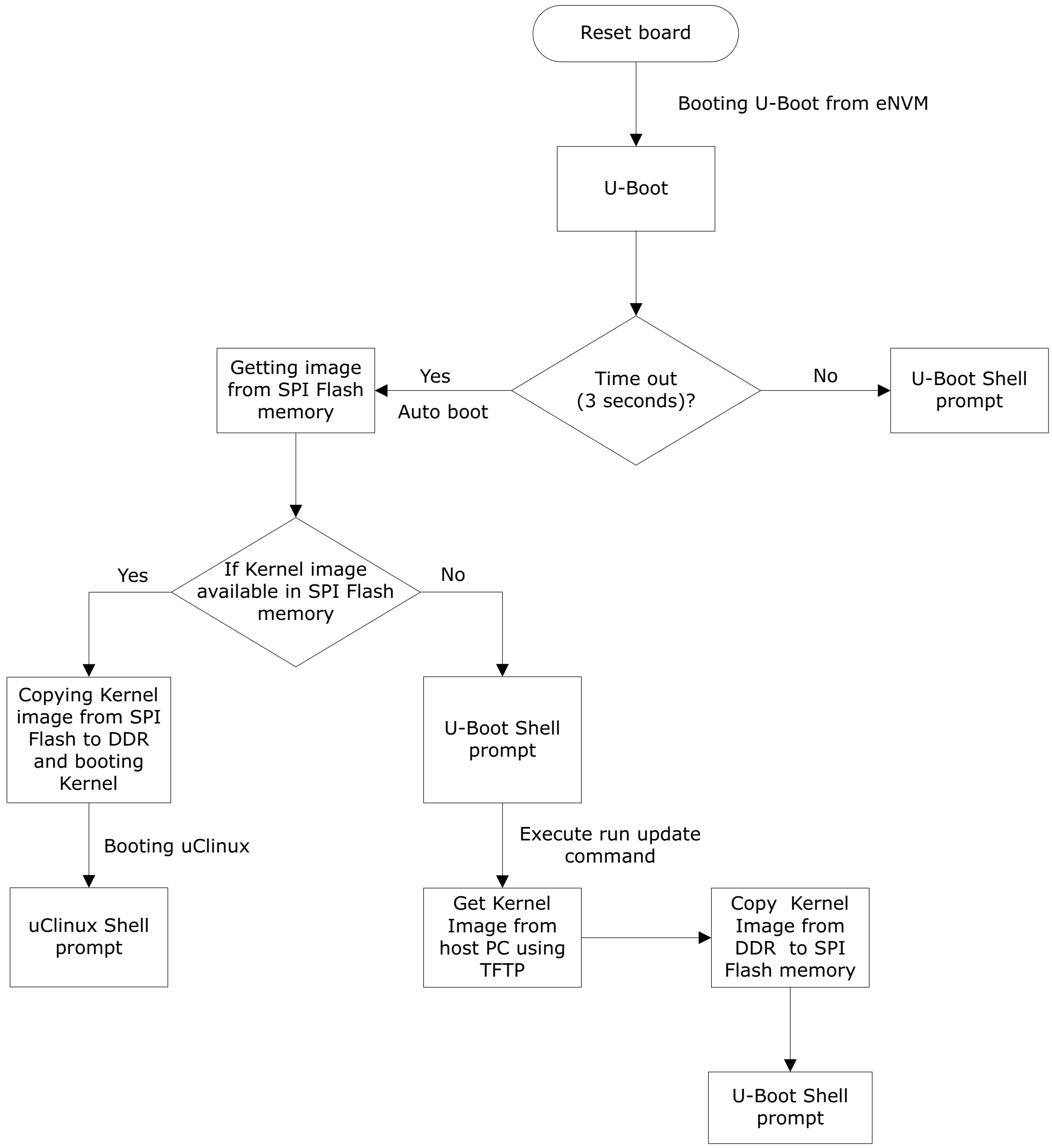
2.1.1.3 Application

The network application is configured to perform ping and http functionalities.

2.1.2 uClinux Booting Flow

The following figure shows the uClinux booting flow.

Figure 2 • **Booting uClinux Flow**



For more information on software functionalities, see Appendix: Software Functionality Supported Features of U-Boot and uClinux, page 26.

2.2 References

The following references are used in this document:

- *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*
- *UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide*

2.3 Design Requirements

The following table shows the design requirements.

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Advanced Development Kit: • 12 V adapter • FlashPro5 programmer • USB A to Mini-B cable	Rev A or later
Host PC or Laptop	RedHat and any Windows Operating System
Software Requirements	
Libero® System on Chip (SoC)	v11.7 SP1
FlashPro programming software	v11.7 SP1
USB to UART drivers	–
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	–

2.4 Design Description

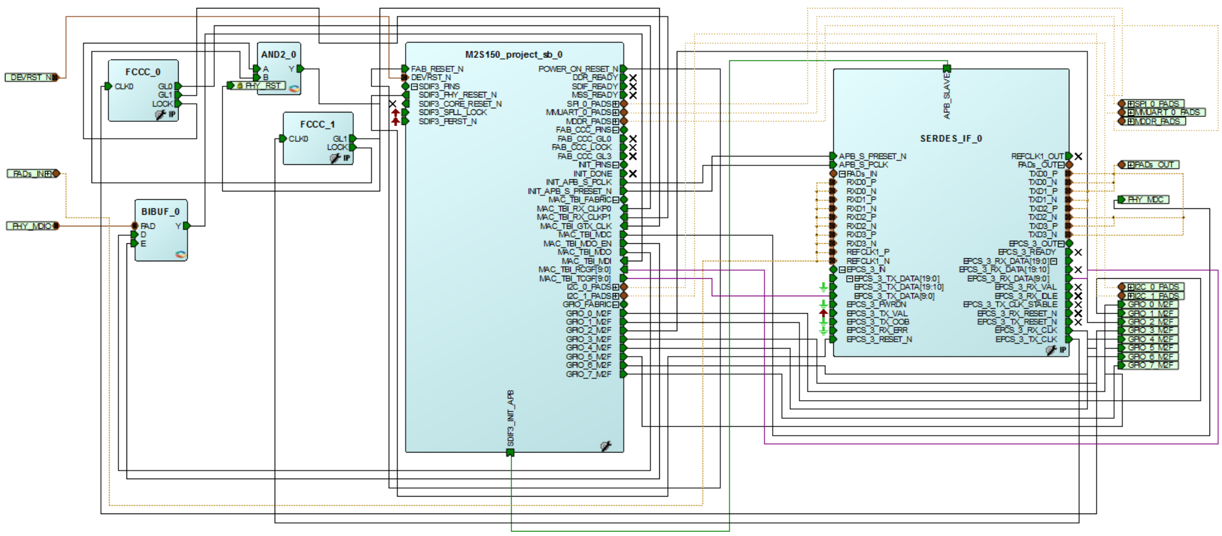
The design example in this application note uses MMUART_0, GPIO, eSRAM, DDR, and eNVM memory controllers. In this example, the system builder clock section is configured, as shown in Figure 3, page 5 to run the M3_CLK at 166 MHz, which drives the clock to the Cortex-M3 processor. Sourcecode of U-Boot, and uClinux along with a network application are compiled using the arm cross compiler.

2.5 Hardware Implementation

The hardware implementation involves configuring microcontroller subsystem (MSS), fabric, clocks, and oscillator using the system builder.

The following figure shows the top-level smart design of the application.

Figure 3 • **Libero Top-Level Design**

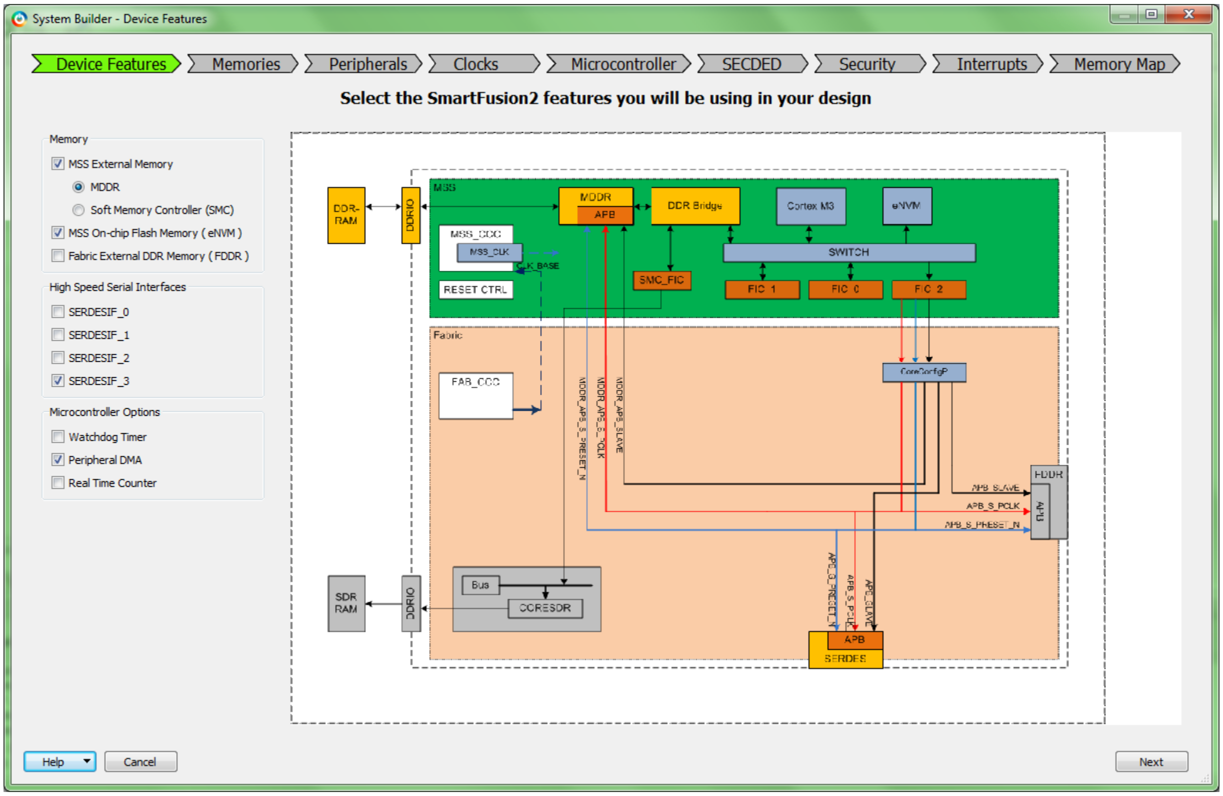


Libero hardware project uses the following SmartFusion2 MSS resources:

- SGMII Interface to Ethernet PHY
- SPI_0 for reading from/writing to SPI flash
- MDDR for DDR3 interface
- MMUART_0 is used as a console interface for the U-Boot and uCLinux software

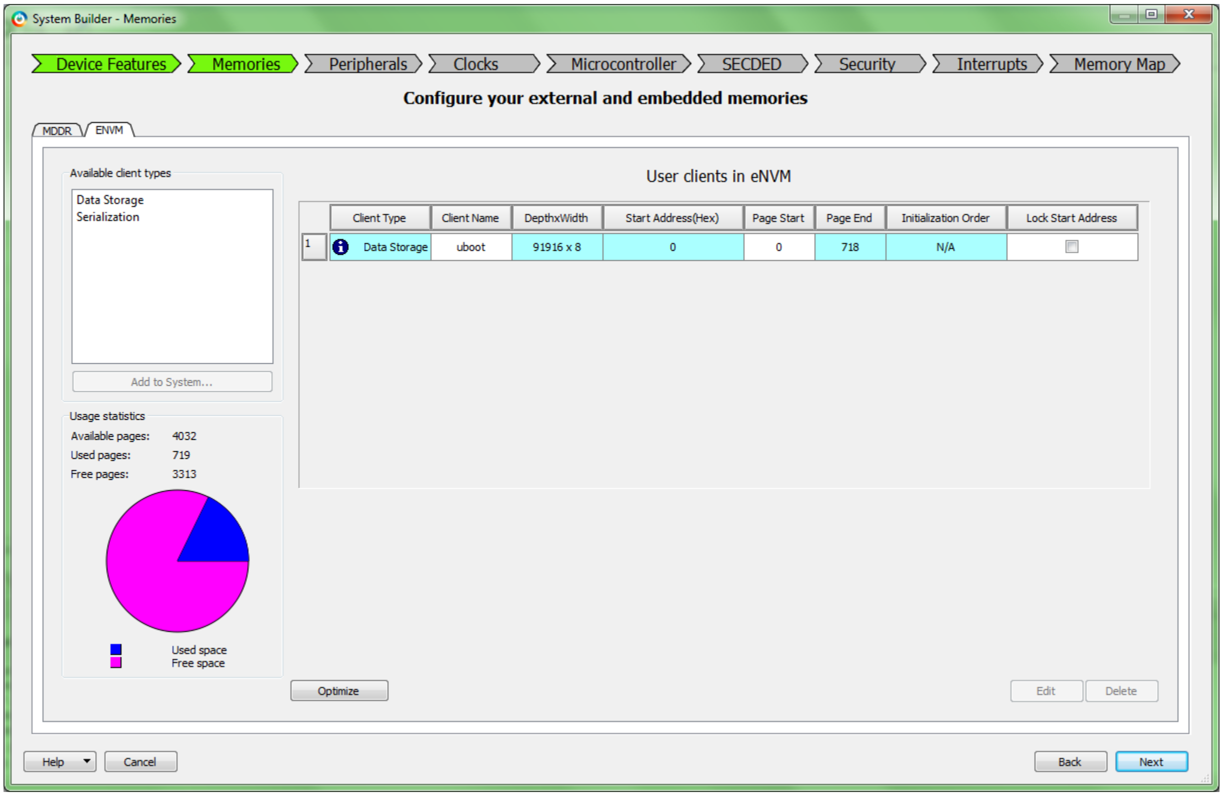
The following figures shows the MSS MDDR configuration settings. For more information on DDR configuration files, see Appendix: Design and Programming Files, page 25.

Figure 4 • **Select MDDR**



Add the eNVM user clients (`U-Boot.hex`) in eNVM configurator under the Memories tab.

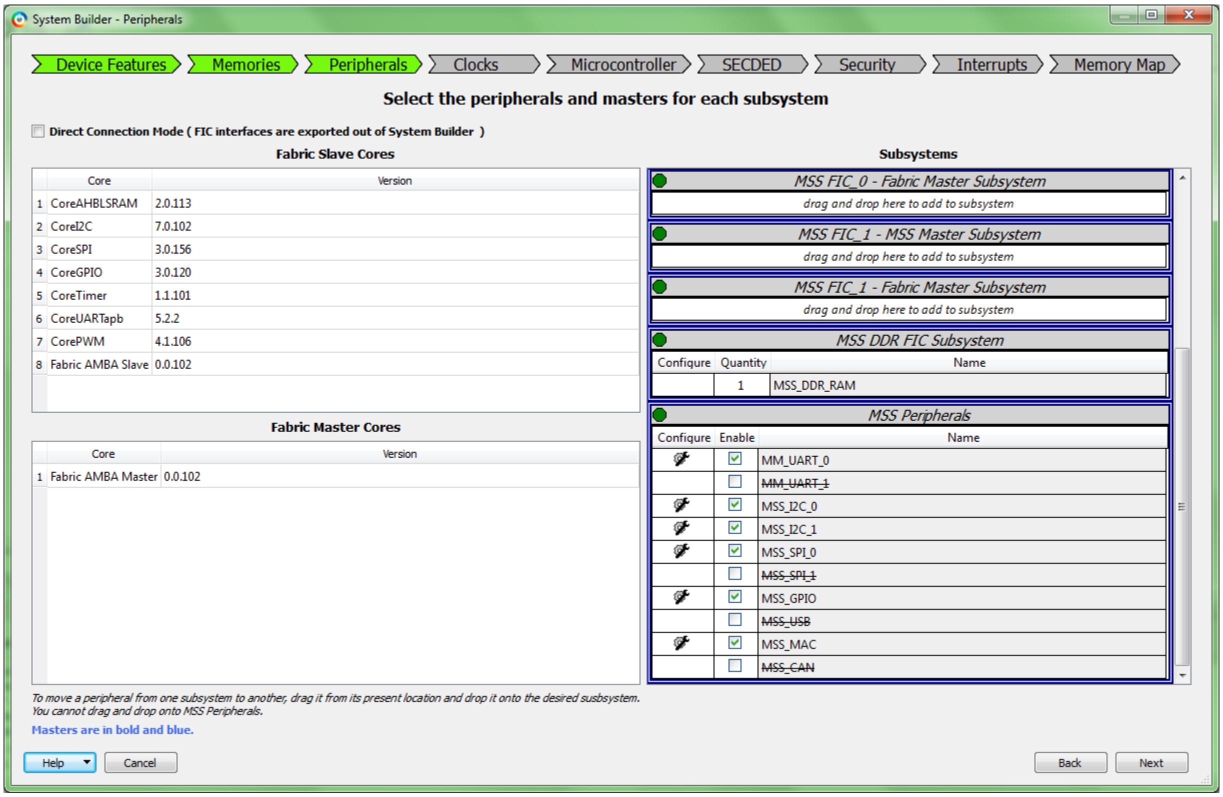
Figure 5 • **Memory Device Configuration**



Enable the following peripherals in the Peripherals configuration window.

- MM_UART_0
- MSS_I2C_0
- MSS_I2C_1
- MSS_SPI_0
- MSS_GPIO
- MSS_MAC

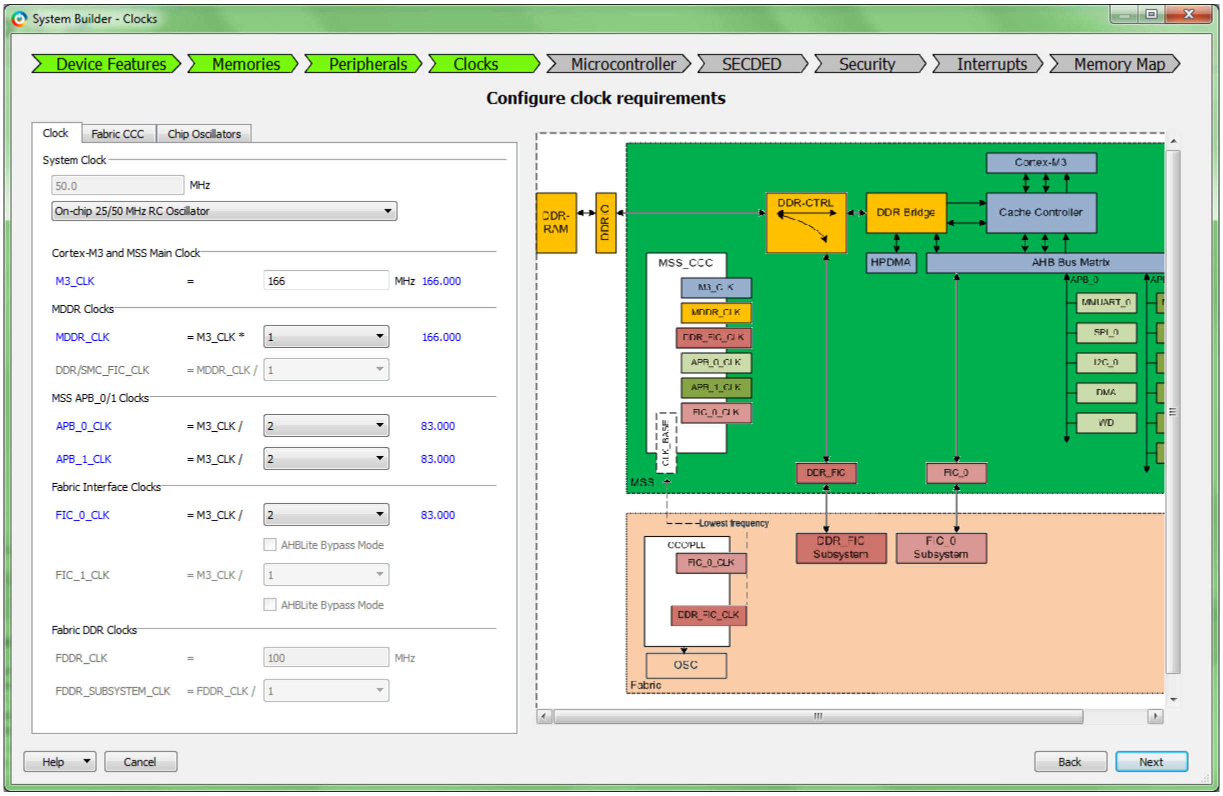
Figure 6 • **Peripherals Configuration**



The MMUART_0 is routed through the FPGA fabric to communicate with the serial terminal program. The MSS_CCC clock is sourced from the FCCC through the CLK_BASE port. The FCCC is configured to provide the 100 MHz clock using GL0.

The following figure shows the system clock configurations for the M3_CLK, MDDR_CLK, and APB_0_CLK/APB_1_CLK.

Figure 7 • **Lock Configuration**



2.6 Software Implementation

The following steps describe the compilation of the U-Boot loader and uClinux Kernel along with networking applications and also describes how to unpack the U-Boot and Linux source folder and set the environment variables required for U-Boot and Linux compilation.

U-Boot loader and uClinux source code are available under the `uboot-linux-source.zip` folder.

1. Create the sf2_linux directory under the home directory using the following command:
```
[@test]$ mkdir sf2_linux
```
2. Copy the uboot-linux-source.zip file to /home/sf2_linux directory.
3. Extract the U-Boot-linux-source.zip file.
```
[@test]$cd sf2_linux\
[@test:~/sf2_linux]$ unzip  uboot-linux-source.zip
```
4. Execute ls -l command to display the contents of the sf2_linux/uboot-linux-source directory.
```
[@test:~/sf2_linux]$ ls -l uboot-linux-source
total 24
drwxrwxr-x 12  test test 4096 May 19 12:51 A2F
-rwxrwxr-x  1    test test  313 Jan  5 12:34 ACTIVATE.sh
drwxrwxr-x 23  test test 4096 May 19 12:51 linux
drwxrwxr-x  5   test test 4096 May 19 12:51 projects
drwxrwxr-x  5   test test 4096 May 19 12:51 tools
drwxrwxr-x 31  test test 4096 May 19 12:51 u-boot
```

The following table shows the U-Boot Linux files and directories.

Table 2 • **U-Boot Linux Source Directories**

Directory	Description
A2F/	Directory with target components
A2F/busybox/	Busybox source and development tree
A2F/dropbear/	Dropbear secure shell (SSH) server source and development tree
A2F/net-snmp/	Source and development tree of the net-snmp package
A2F/uclibc	Source and development tree of the uClibc package
A2F/gdb-2011.03	Source and development tree of the GNU debugger package
A2F/hostapd-1.0	Source and development tree of the HostAP daemon package
A2F/wireless_tools	Source and development tree of the wireless tools package
A2F/libnl-3.2.11	Source and development tree of the netlink protocol library package
A2F/netperf-2.6.0	Source and development tree of the Netperf network benchmarking package
A2F/root	Pre-built target binaries ready for use on the target
U-Boot/	U-Boot source and development tree
linux/	Linux (uClinux) Kernel source and development tree
projects/	Sample projects (embedded applications)
tools/	Development tools
tools/bin/mkimage	Utility used by the Linux Cortex-M3 Kernel build process to create a bootable U-Boot Kernel image such as uImage
ACTIVATE.sh	Shell script to be performed in order to activate the Linux Cortex-M development environment on the host

Go to the uboot-linux-source folder and run the following activate script to set the environment variables.

```
[@test:~/sf2_linux/ uboot-linux-source]$ source  ACTIVATE.sh
```

5. ACTIVATE.sh, sets the following environment variables in the Linux development environment:
   - INSTALL_ROOT
   - CROSS_COMPILE
   - CROSS_COMPILE_APPS
   - PATH
   - MCU
6. Build U-Boot and uClinux along with a networking application, after setting the Linux development environment.

2.6.1 Compiling U-Boot

The following steps helps in compiling a U-Boot for the SmarFusion2 Advanced Development Kit:

1. Use the following command to go to the U-Boot source directory.:
   ```
   [@test:~/sf2_linux/ uboot-linux-source]$ cd projects/uboot
   ```
2. Select the SmartFusion2 Advanced Kit config file m2s-150-adk_config for U-Boot configuration and run the make command.
   ```
   [@test:~/sf2_linux/ uboot-linux-source]$ make m2s-150-adk_config
   ```
3. To compile the U-Boot source, execute the make command.
   ```
   [@test:~/sf2_linux/ uboot-linux-source]$ make
   ```
4. To generate `.hex` file after U-Boot compilation, execute make `u.boot.hex` command
   ```
   [@test:~/sf2_linux/ uboot-linux-source]$ make  u-boot.hex
   ```

`U-Boot.hex` file is used as an eNVM client in the Libero system builder.

Note: The CodeSourcery tools are 32-bit applications and need a compatible library to run on 64-bit Linux OS distributions. On the recent Fedora distributions, the compatibility package (glibc-2.17-14.fc19.i686) is installed by default. On Debian, you need to install it by running the command sudo apt-get install ia32-libs.

2.6.2 Compiling uClinux and Network Application

In the uboot-linux source folder, the Linux directory contains the uCLinux Kernel source code and the project directory contains the networking application.

Compiling the networking application in turn compiles the uClinux Kernel and generates the networking uImage.

1. Use the following command to go to the networking folder.
   ```
   [@test:~/sf2_linux/ uboot-linux-source]$ cd projects/ networking
   ```
2. To build the networking application, execute the make command.
   ```
   [@test:~/sf2_linux/ uboot-linux-source /projects/networking]$ make
   ```

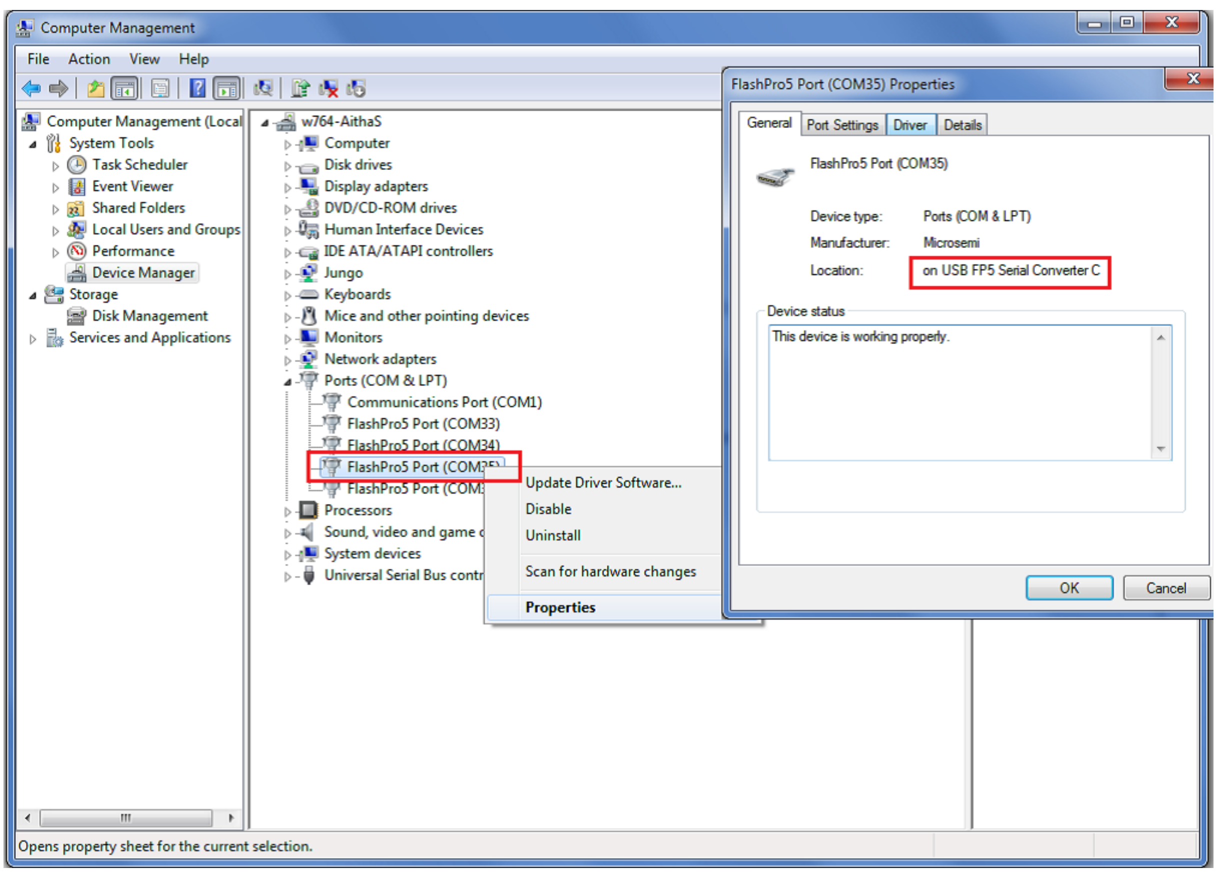
The networking.uImage is generated after successful compilation of networking project.

2.7 Setting Up the Demo Design

The following steps describe how to setup a demo design:

1. Connect the host PC (Windows) to the J33 connector using the USB A to mini-B cable. The USB to UART bridge drivers are automatically detected. If USB to UART bridge drivers are not installed, download and install the drivers from the following location: www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip
2. From the detected four COM ports, right-click any one of the COM ports and select **Properties**. Ensure to have the Location as **on USB FP5 Serial Converter C** in the **Properties** window.

Figure 8 • **Device Manager Window**



3. Connect the host PC to the J21 connector of the SmartFusion2 Advanced Development Kit board using an RJ45 cable.
4. Connect the jumpers on the SmartFusion2 Advanced Development Kit board, as shown in the following table.

CAUTION: Ensure that power supply switch **SW7** is switched OFF while connecting the jumpers on the SmartFusion2 Advanced Development Kit board.

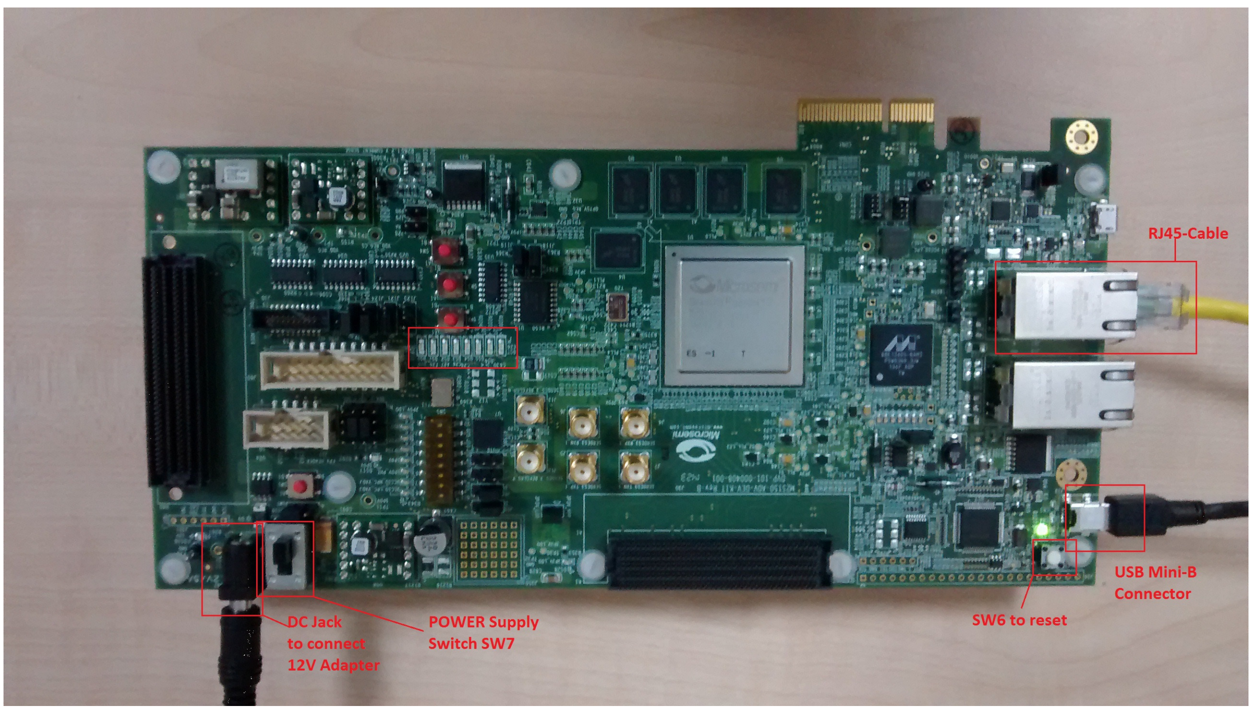
Table 3 • **SmartFusion2 SoC Advanced Development Kit Jumper Settings**

Jumper	Pin (From)	Pin (To)	Comments
J116, J353, J354, J54	1	2	These are the default jumper settings of the Advanced Kit board. Ensure these jumpers are set accordingly.
J123	2	3	
J124, J121, J32	1	2	JTAG programming through FTDI

5. Connect the power supply to the J42 connector on the SmartFusion2 Advanced Development Kit board.

The following figure shows the board setup for running the design on the SmartFusion2 Advanced Development Kit board.

Figure 9 • **SmartFusion2 Advanced Development Kit**



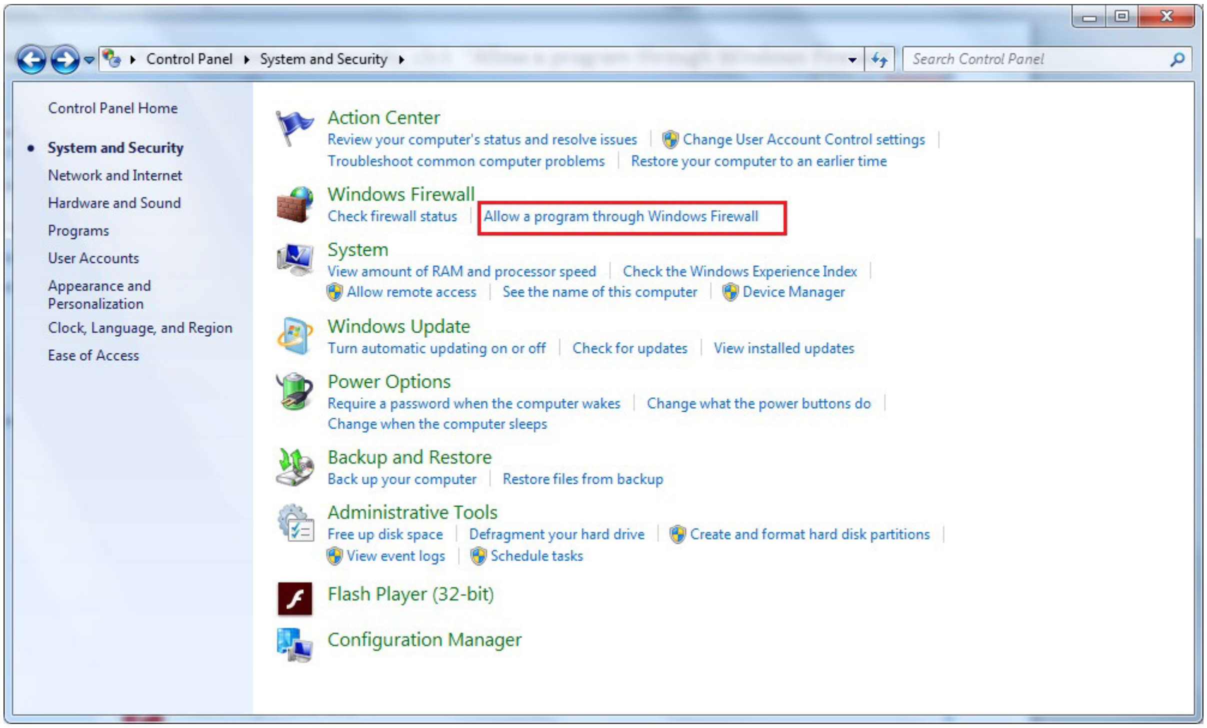
2.7.1 Setting Up the TFTP Server Application in Host PC

Execute the following steps to get the Linux image (networking.uImage) from the host PC, using the TFTP server and load it to the SPI flash memory.

1. Download and install the TFTP server application on the host PC using the following link: http://tftpd32.jounin.net/tftpd32_download.html
TFTP server application is allowed through Windows Firewall, follow the steps.

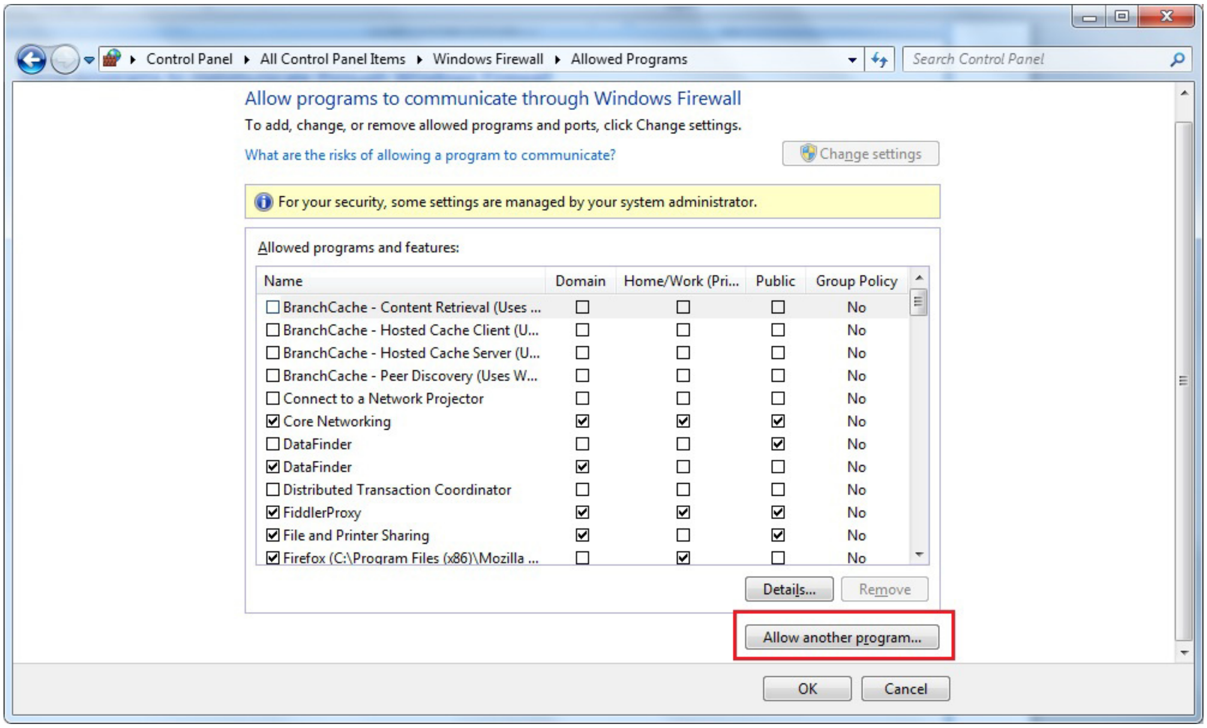
   a. Navigate to **Control Panel > System and Security**, click **Allow a program through Windows Firewall**.

Figure 10 • **System and Security Window**



b. Click **Change settings > Allow another program**…

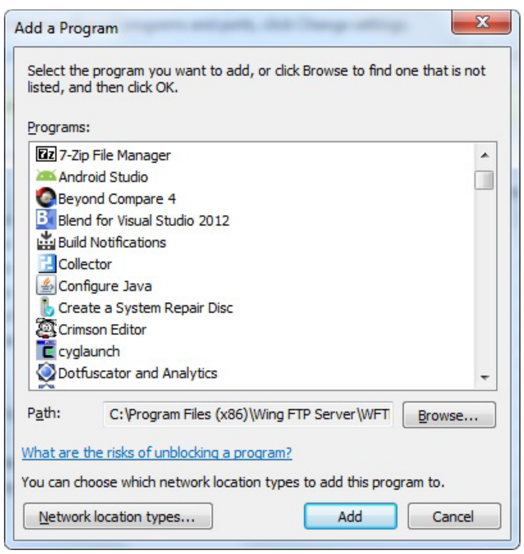
Figure 11 • **Choose Allow Another Program**



Add a Program window is displayed.

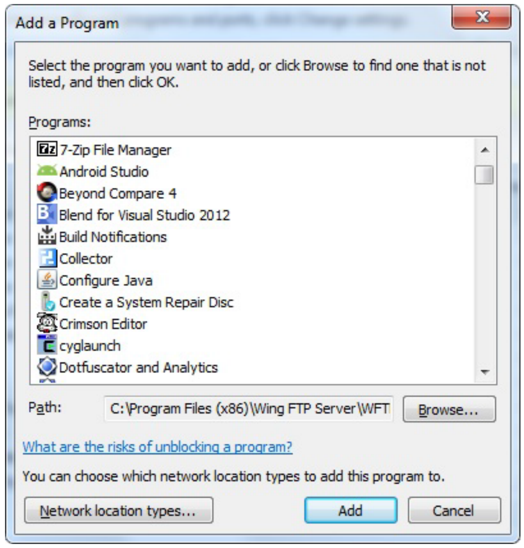
c. Click **Browse**... to install the TFTP server application.

Figure 12 • **Add a Program Window**



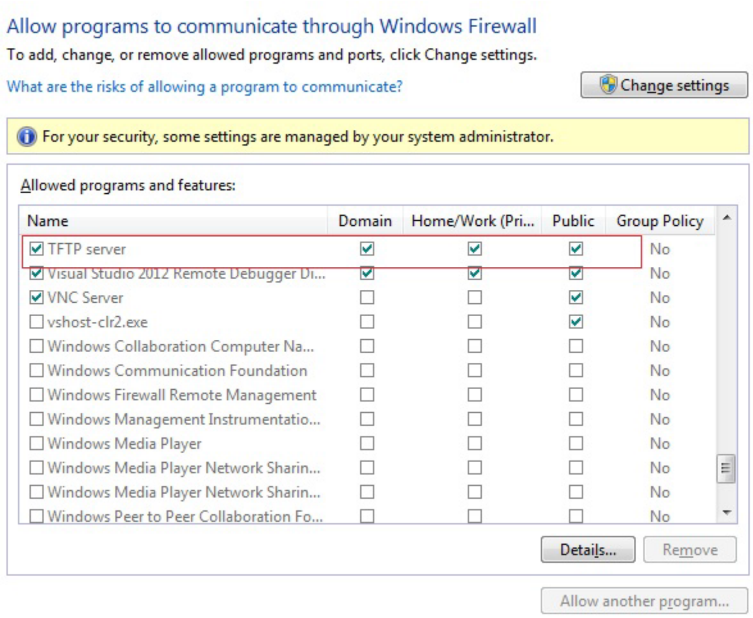
d. Click **Add**.

Figure 13 • **Adding TFTP Server Application**



e. Select **TFTP server** check box, as shown in the following figure.

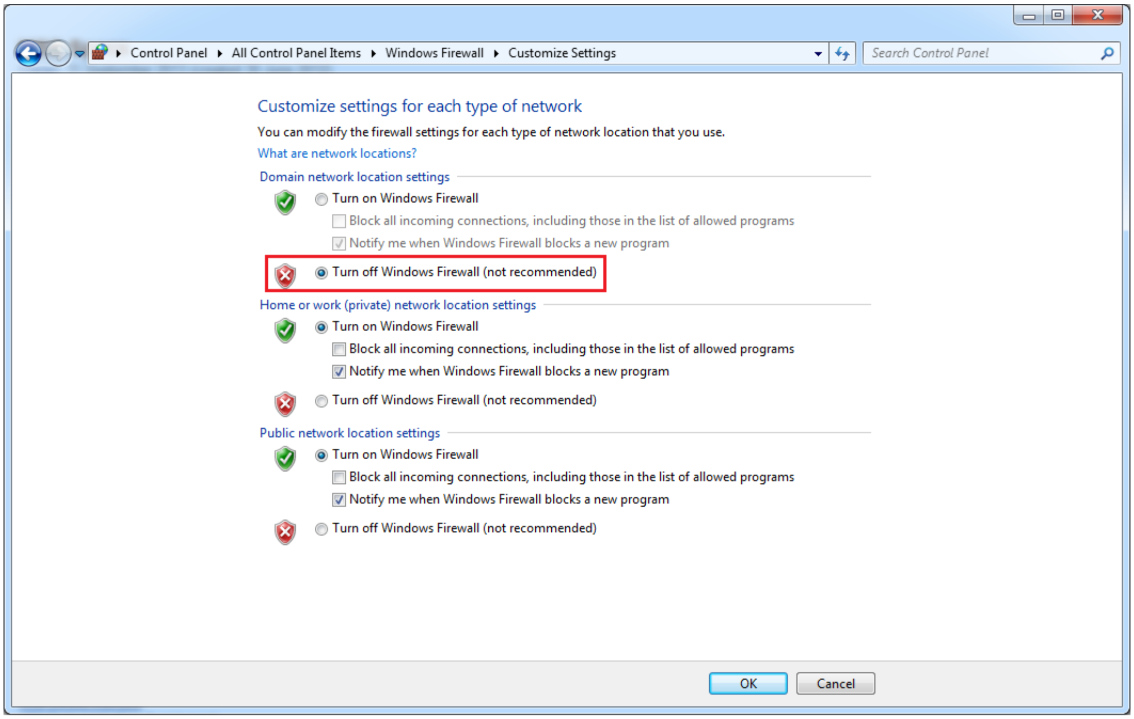
Figure 14 • **TFTP Server Application**



f. Click OK.

2. Navigate to **Control Panel > Windows Firewall > Turn Windows Firewall On or Off**. Select **Turn off Windows Firewall** under **Domain network location settings** and click **OK**.

Figure 15 • **Windows Firewall Settings**

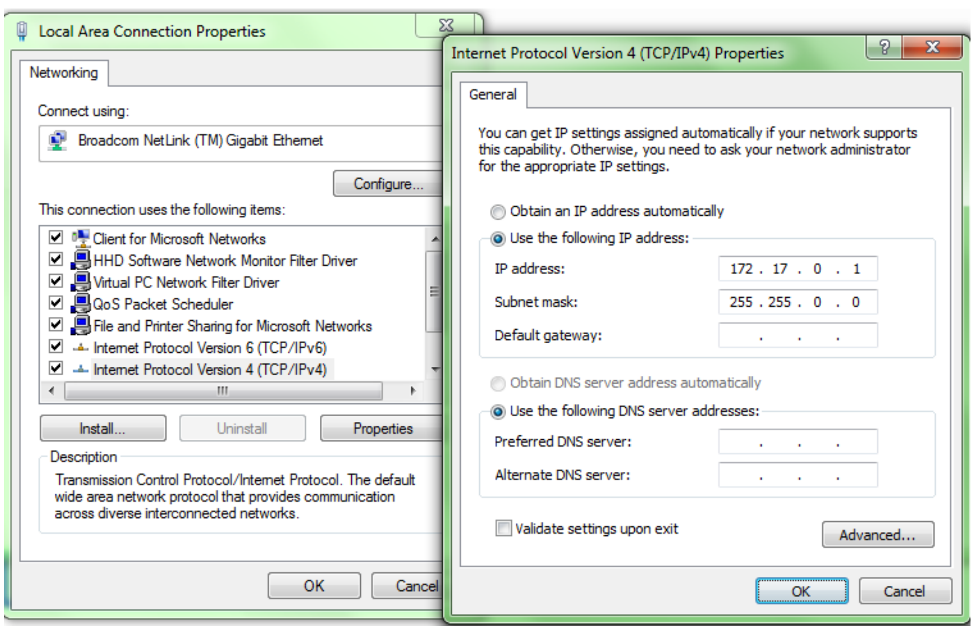


2.7.2 Configuring Host PC IP Address

As U-Boot configured in static IP address mode, 172.17.0.1 IP address is used to access the TFTP server. The following steps describe how to configure the host PC IP address:

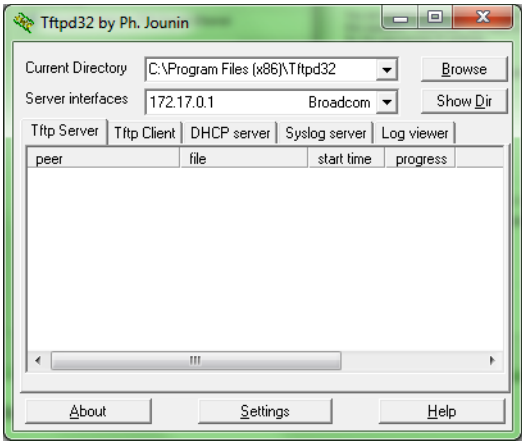
1. Configure the host PC TCP/IP as 172.17.0.1.

Figure 16 • **Host PC TCP/IP Settings**



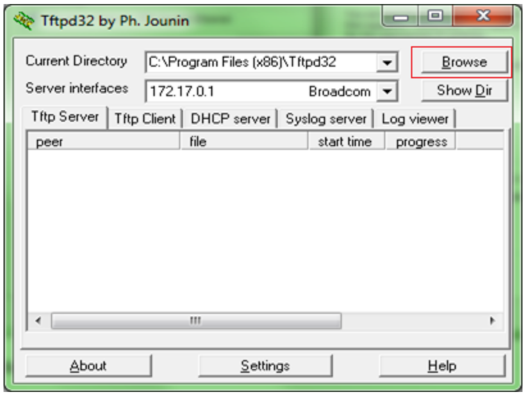
2. Invoke the Tftpd server application from **All Programs>Tftpd32>**, the TFTP server application is displayed.

Figure 17 • **Tftpd Server Application**



3. Click **Browse** and select the file networking .uImage. The default location of the image is: *<download_folder>\ Sf2_Running_uClinux_appnote_df \Linux_image*

Figure 18 • **Tftpd Server Application Browser Window**



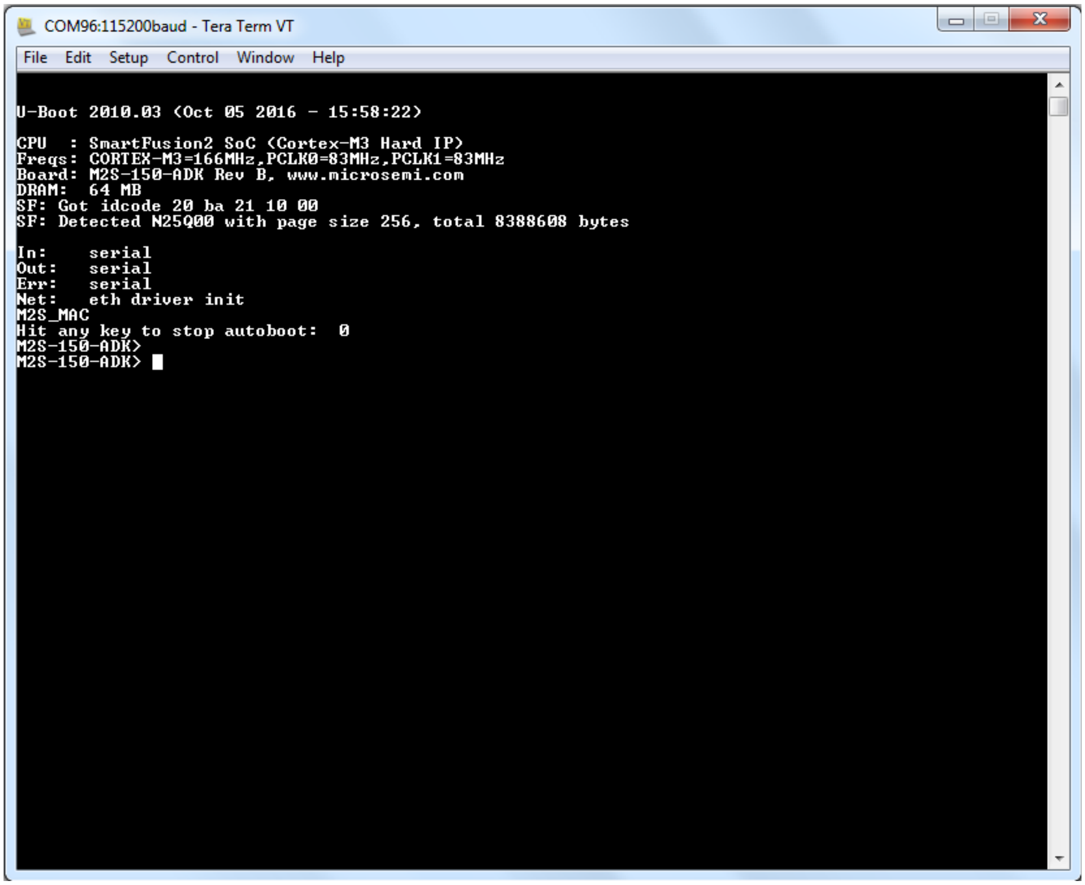
2.8 Running the Design

2.8.1 Booting U-Boot

The following steps describe how to boot the U-Boot:

1. Press **SW9** switch to reset the board after successful programming.
2. Cortex-M3 runs the U-Boot from the on-chip eNVM, press any key on the U-Boot prompt window within three seconds to stop auto boot.

Figure 19 • **U-Boot Prompt**

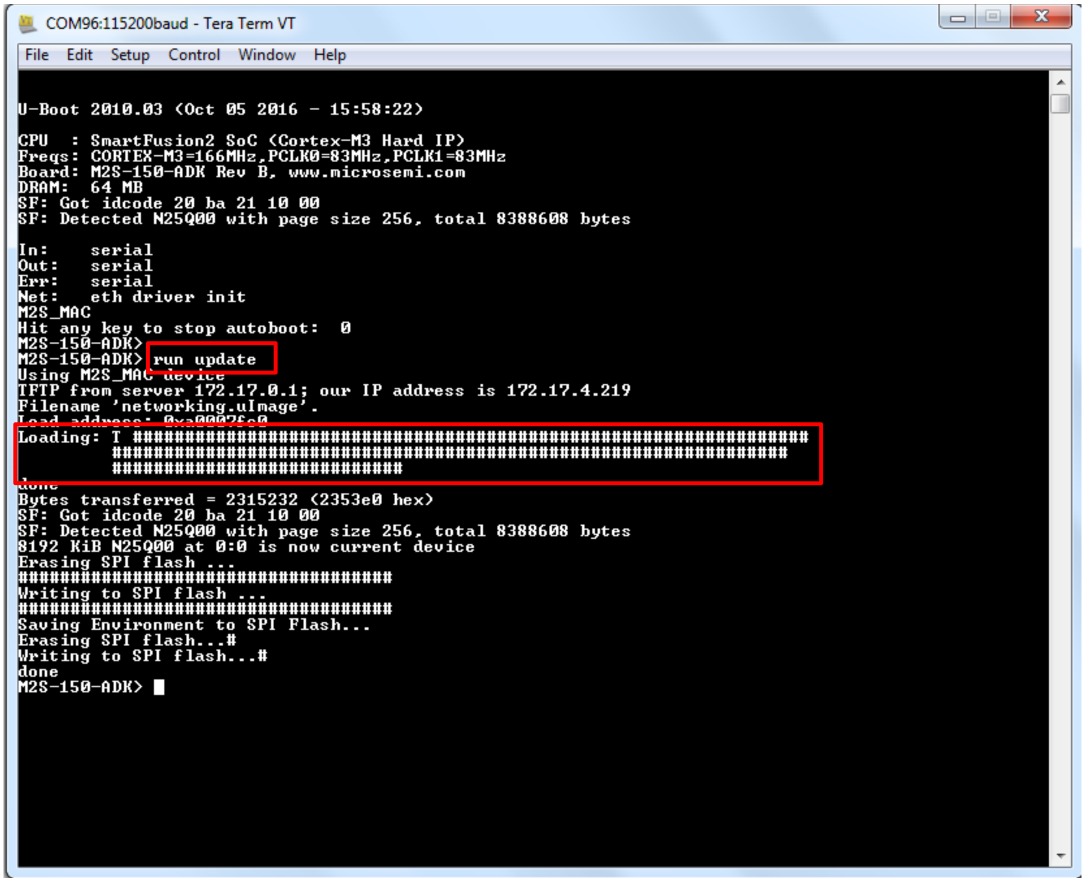


2.8.2 Programming Linux Kernel Image onto SPI Flash using TFTP Server Application

To copy networking.uImage from the host PC using TFTP server to device SPI flash memory, execute **run update** command.

```
M2S-150-ADK >run update
```

Figure 20 • **Run Update**



Linux Kernel image copies to SPI flash.

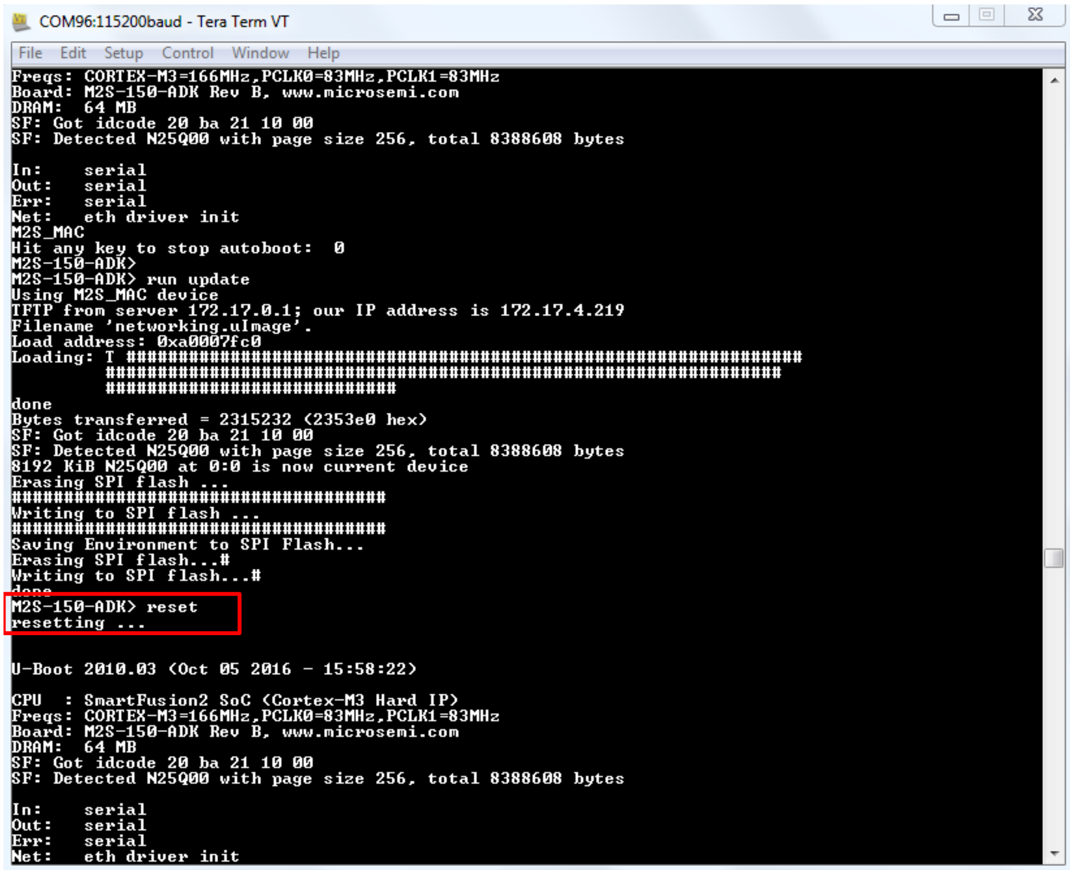
2.8.3 Booting the Linux Kernel

The following steps describe how to boot the Linux Kernel:

1. Execute the **reset** command on U-Boot prompt, to boot the Linux Kernel.

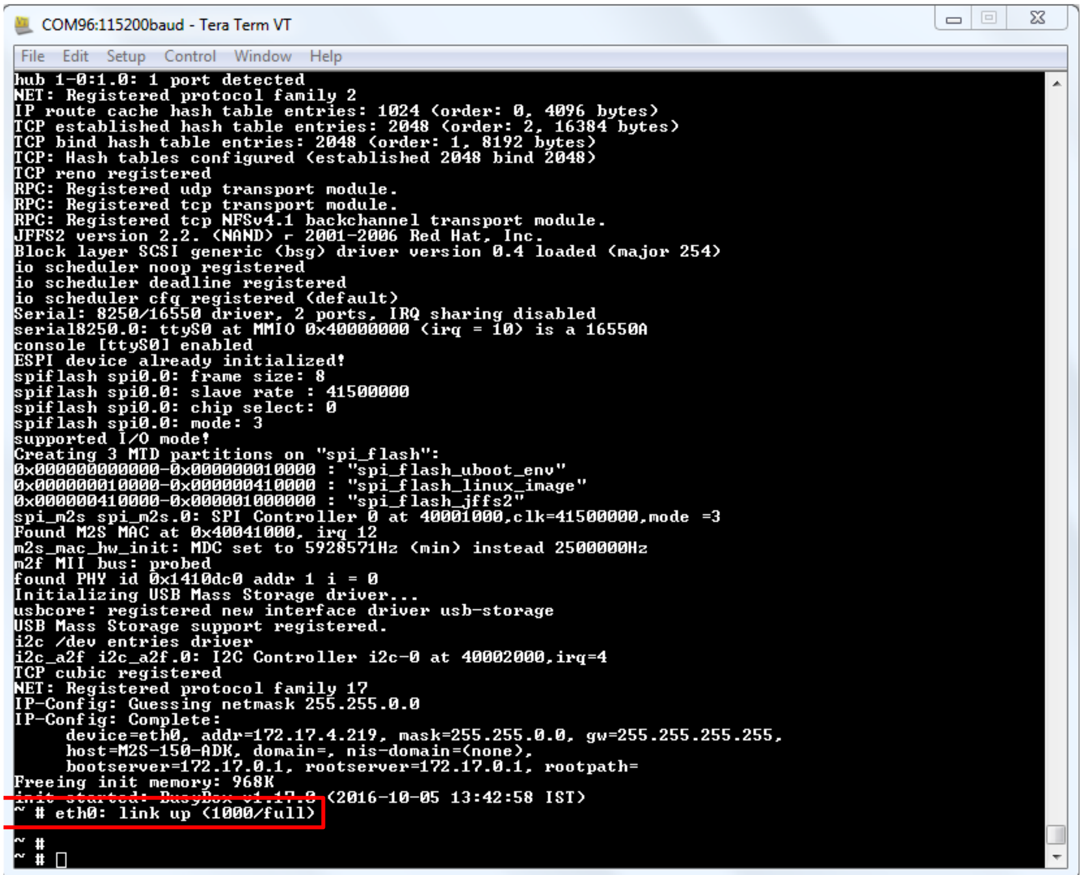
```
M2S-150-ADK > reset
resetting...
```

Figure 21 • **Resetting the Board**



After uClinux Kernel booting is completed, Linux Prompt appears, as shown in the following figure.

Figure 22 • **uClinux Prompt**



2.8.4 Running the Networking Applications

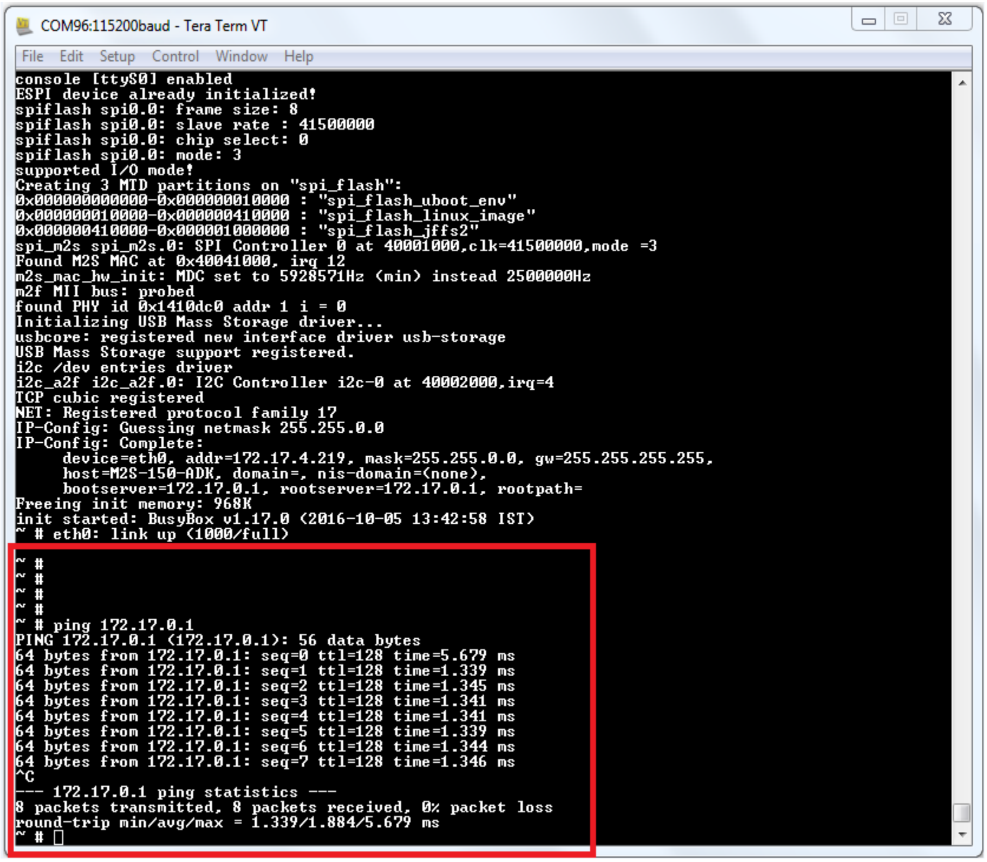
Ping and Http are the applications used to test the networking functionalities:

2.8.4.1 Ping

To ping the host PC, run the following command in uClinux prompt:

```
~ # ping 172.17.0.1
```

Figure 23 • **Ping Command**

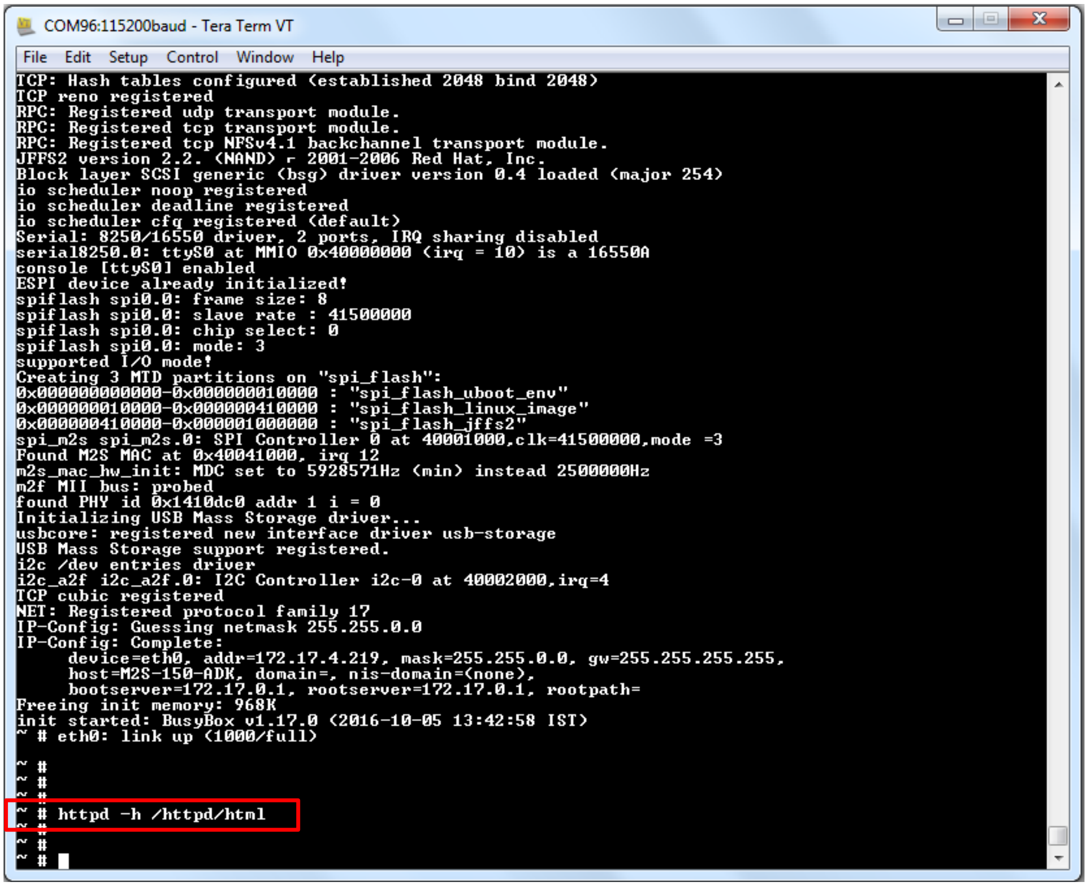


2.8.4.2 Http

To access a http page on the host PC, execute the steps:

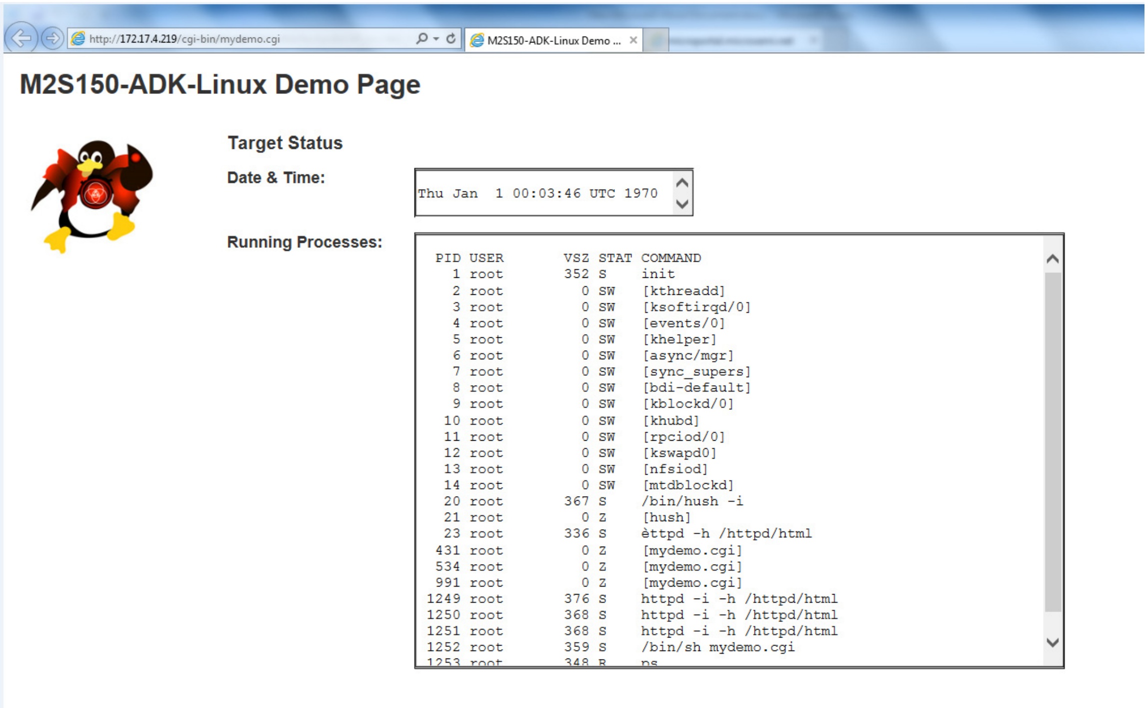
1. Run the following command in uClinux prompt.
   `~ # httpd -h /httpd/html`

Figure 24 • **Http Command**



2. Open the browser in the host PC and enter the board static IP address (172.17.4.219). The SmartFusion2 Linux Demo page is displayed.

Figure 25 • **SmartFusion2 Linux Demo Page**



2.9 Conclusion

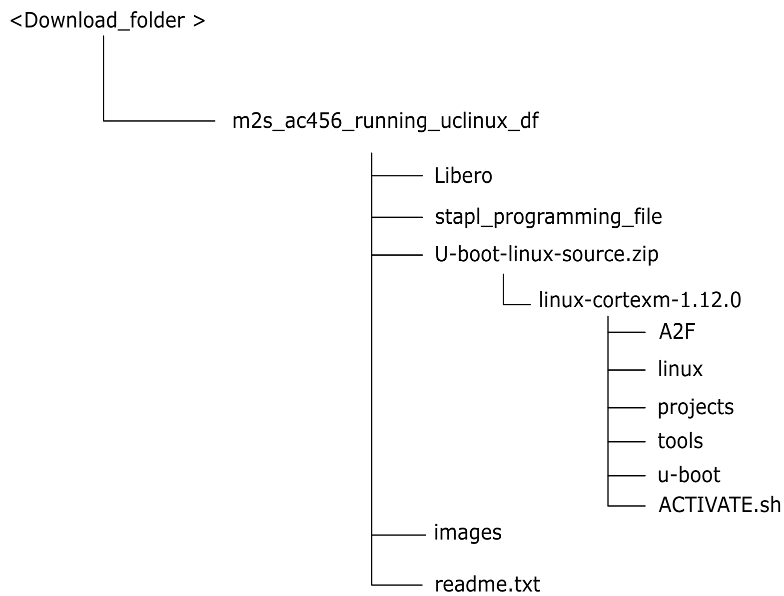
This application describes how to compile and boot uClinux and shows the networking applications on the SmartFusion2 Advanced Development Kit.

3 Appendix: Design and Programming Files

Download the SmartFusion2 design files from the following link on the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_ac456_running_uclinux_df

The following figure shows the top-level structure of the design files. Refer to the `readme.txt` file included in the design files for the directory structure and description

Figure 26 • **Demo Design Top-Level Structure**



.

4 Appendix: Software Functionality Supported Features of U-Boot and uClinux

4.1 U-Boot Firmware

Following are the functionalities of the U-Boot firmware:

- U-Boot v2010.03
- Target initialization from power-on/reset
- Runs from the internal eNVM (no external memory required for standalone operation)
- Serial console
- Ethernet driver for loading images to the target
- Device driver for built-in Flash (eNVM) and self-upgrade capability
- Device driver for storing environment and Linux images in external Flash
- Autoboot feature, allowing the boot of the OS images from Flash or other storage with no operator intervention
- Persistent environment in Flash for customization of target operation

4.2 uClinux

Following are the functionalities of uClinux firmware:

- uClinux Kernel v2.6.33
- Boot from compressed and uncompressed images
- Serial device driver and Linux console
- Ethernet device driver and networking (ping, NFS, Telnet, FTP, ntpd, and so on)
- busybox v1.17
- POSIX pthreads
- Process-to-Kernel and process-to-process protection using the memory protection unit (MPU) of the SmartFusion2 core
- Hardened exception handling
- Loadable Kernel modules
- SSH daemon
- Web server
- SPI controller master-mode device driver
- I2C (master slave device drivers)
- GPIO device driver
- Device driver for the embedded