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

1 Revision History 1

1.1 Revision 1.0 1

2 Running uClinux on SmartFusion2 Advanced Development Kit Board 2

2.1 Introduction 2

2.1.1 Boot Sequence 2

2.1.2 uClinux Booting Flow 3

2.2 References 3

2.3 Design Requirements 4

2.4 Design Description 4

2.5 Hardware Implementation 5

2.6 Software Implementation 9

2.6.1 Compiling U-Boot 10

2.6.2 Compiling uClinux and Network Application 11

2.7 Setting Up the Demo Design 11

2.7.1 Setting Up the TFTP Server Application in Host PC 13

2.7.2 Configuring Host PC IP Address 16

2.8 Running the Design 18

2.8.1 Booting U-Boot 18

2.8.2 Programming Linux Kernel Image onto SPI Flash using TFTP Server Application 19

2.8.3 Booting the Linux Kernel 20

2.8.4 Running the Networking Applications 22

2.9 Conclusion 24

3 Appendix: Design and Programming Files 25

4 Appendix: Software Functionality Supported Features of U-Boot and uClinux 26

4.1 U-Boot Firmware 26

4.2 uClinux 26

Figures

Figure 1 Boot Sequence 2
Figure 2 Booting uClinux Flow 3
Figure 3 Libero Top-Level Design 5
Figure 4 Select MDDR 6
Figure 5 Memory Device Configuration 7
Figure 6 Peripherals Configuration 8
Figure 7 Lock Configuration 9
Figure 8 Device Manager Window 11
Figure 9 SmartFusion2 Advanced Development Kit 12
Figure 10 System and Security Window 13
Figure 11 Choose Allow Another Program 14
Figure 12 Add a Program Window 14
Figure 13 Adding TFTP Server Application 15
Figure 14 TFTP Server Application 15
Figure 15 Windows Firewall Settings 16
Figure 16 Host PC TCP/IP Settings 16
Figure 17 Tftpd Server Application 17
Figure 18 Tftpd Server Application Browser Window 17
Figure 19 U-Boot Prompt 18
Figure 20 Run Update 19
Figure 21 Resetting the Board 20
Figure 22 uClinux Prompt 21
Figure 23 Ping Command 22
Figure 24 Http Command 23
Figure 25 SmartFusion2 Linux Demo Page 24
Figure 26 Demo Design Top-Level Structure 25

Tables

Table 1 Design Requirements 4
Table 2 U-Boot Linux Source Directories 10
Table 3 SmartFusion2 SoC Advanced Development Kit Jumper Settings 12

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.

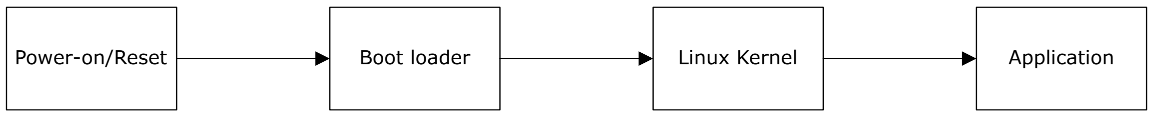
U-Boot is a multi-functional open-source bootloader that allows the developer to either load an operating system from different devices or configure a low-level platform. By power-on or reset, SmartFusion2 runs U-Boot firmware from the on-chip eNVM and completes the basic initialization by storing the 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

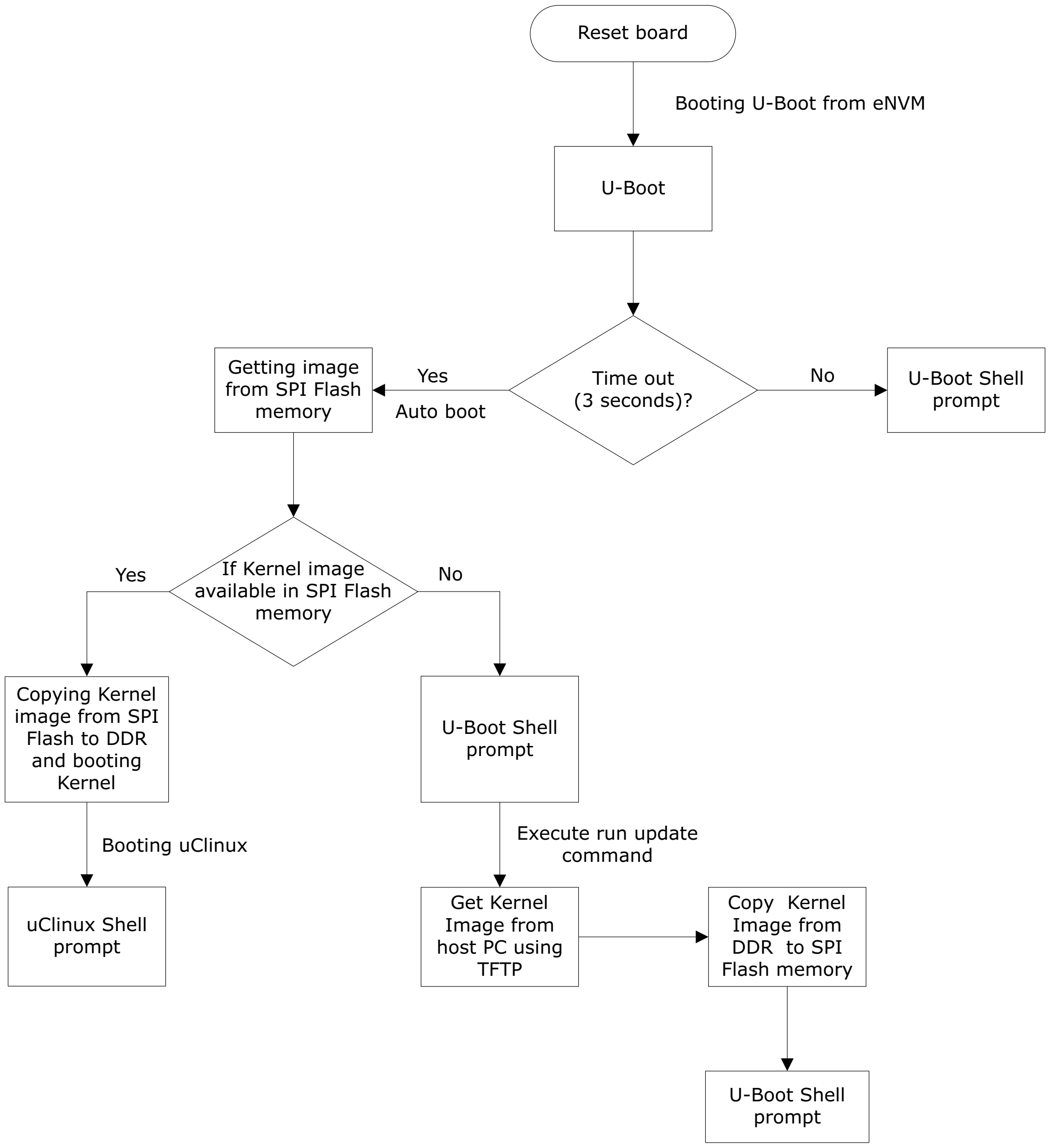
2.1.1.3 Application

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**


Reset board
Booting U-Boot from eNVM
U-Boot
Time out (3 seconds)?
Yes
Auto boot
No
Getting image from SPI Flash memory
U-Boot Shell prompt
If Kernel image available in SPI Flash memory
Yes
No
Copying Kernel image from SPI Flash to DDR and booting Kernel
U-Boot Shell prompt
Booting uClinux
Execute run update command
uClinux Shell prompt
Get Kernel Image from host PC using TFTP
Copy Kernel Image from DDR to SPI Flash memory
U-Boot Shell prompt

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

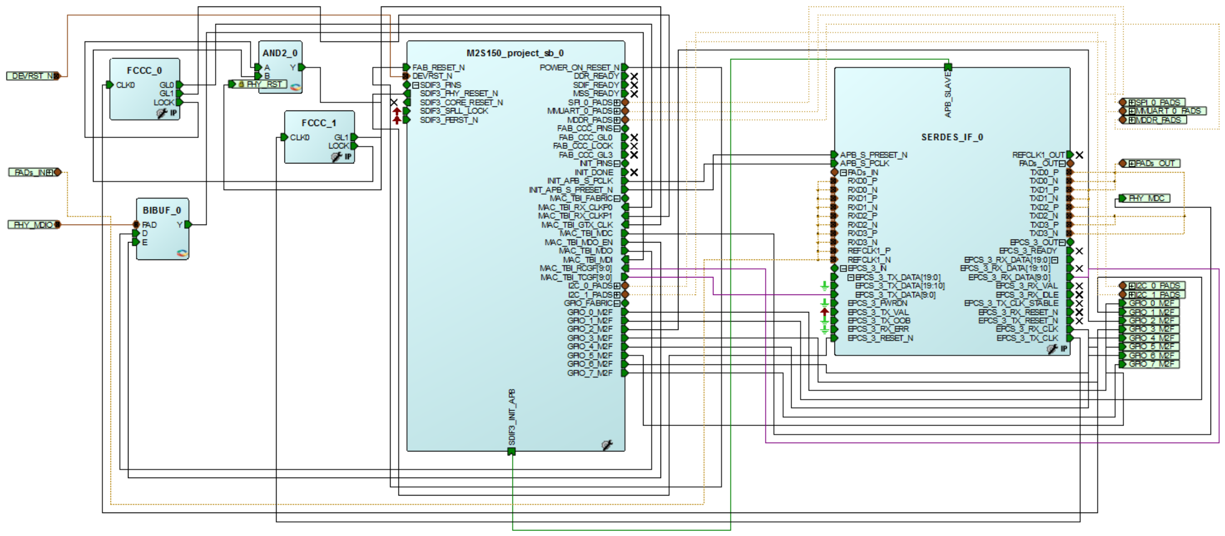
The design example in this application note uses MMUART_0, GPIO, eSRAM, DDR, and eNVM memory controllers. In this example, the system builder clock section is configured, as shown in Figure 3, page 5 to run the M3_CLK at 166 MHz, which drives the clock to the Cortex-M3 processor. Sourcecode of U-Boot,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

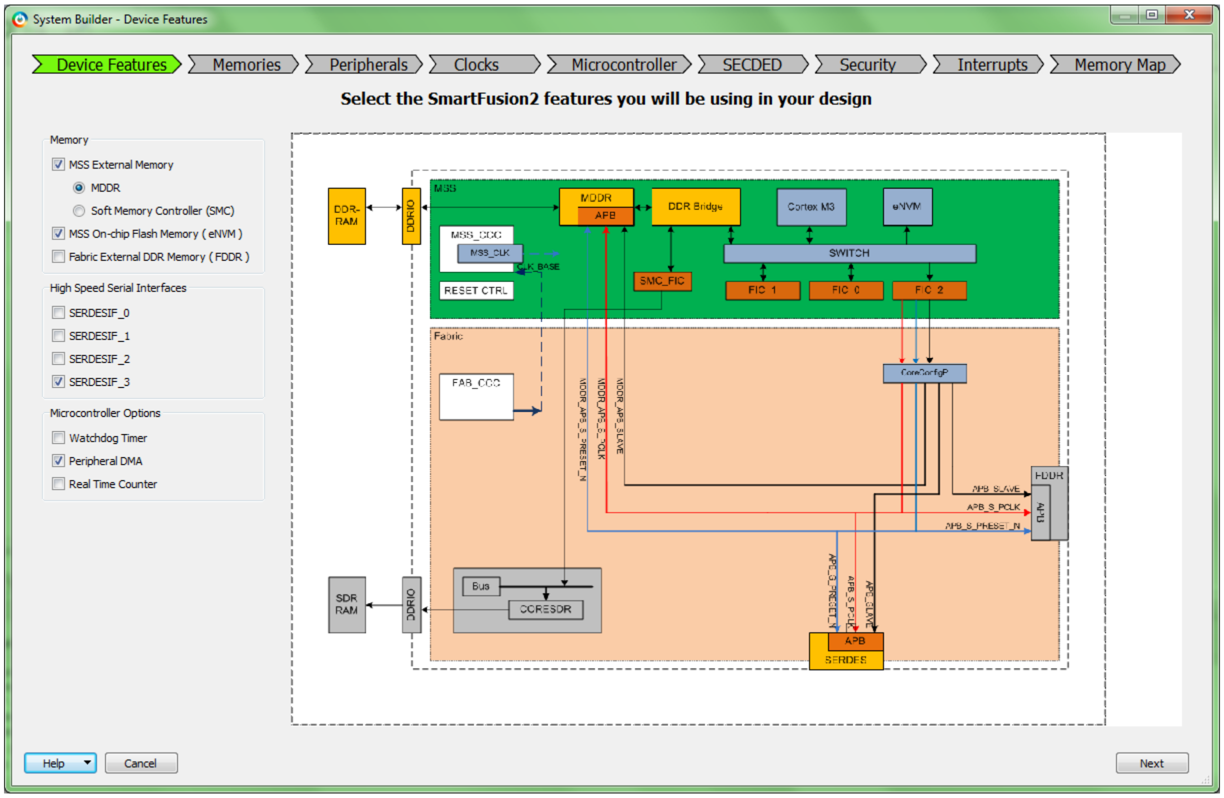


Libero hardware project uses the following SmartFusion2 MSS resources:

- 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**



System Builder - Device Features

Select the SmartFusion2 features you will be using in your design

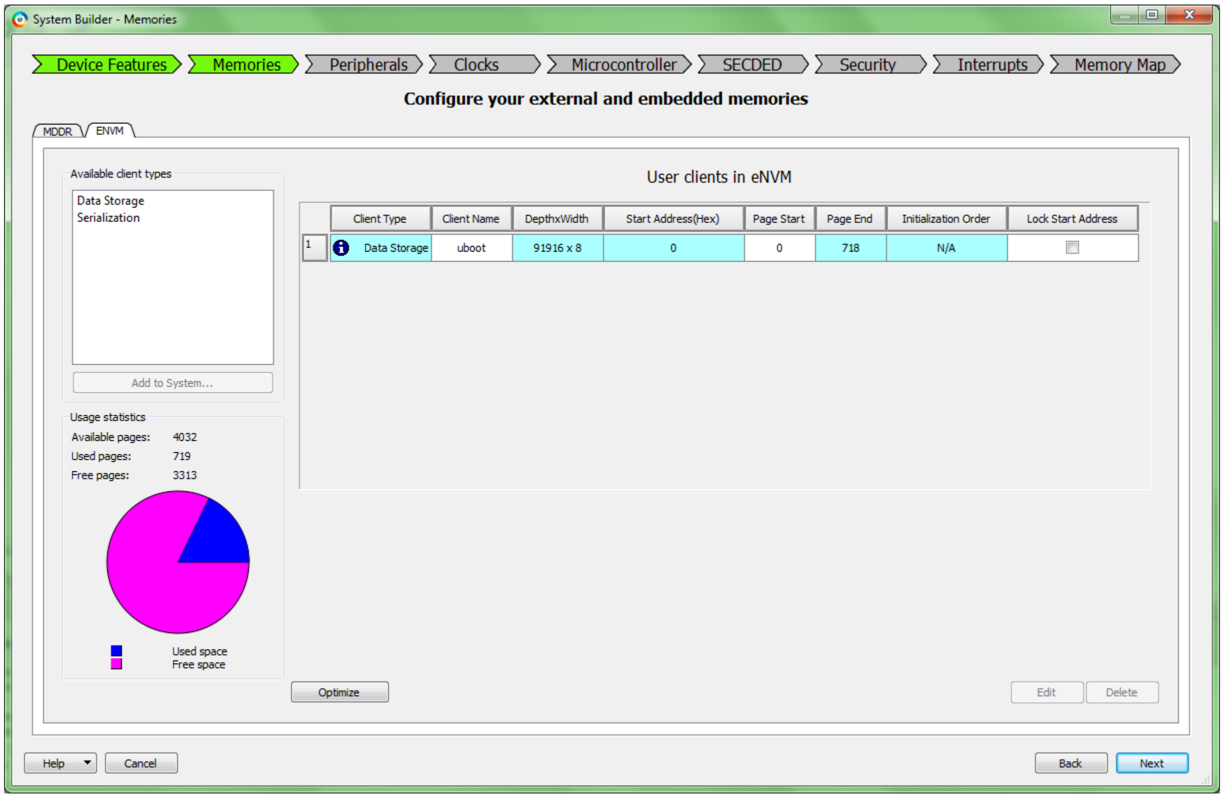
Memory
- [✓] MSS External Memory
  - [● MDDR]
  - Soft Memory Controller (SMC)
- [✓] MSS On-chip Flash Memory (eNVM)
- [ ] Fabric External DDR Memory (FDDR)

High Speed Serial Interfaces
- [ ] SERDESIF_0
- [ ] SERDESIF_1
- [ ] SERDESIF_2
- [✓] SERDESIF_3

Microcontroller Options
- [ ] Watchdog Timer
- [✓] Peripheral DMA
- [ ] Real Time Counter

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

Figure 5 • **Memory Device Configuration**



Usage statistics

Available pages:	4032
Used pages:	719
Free pages:	3313

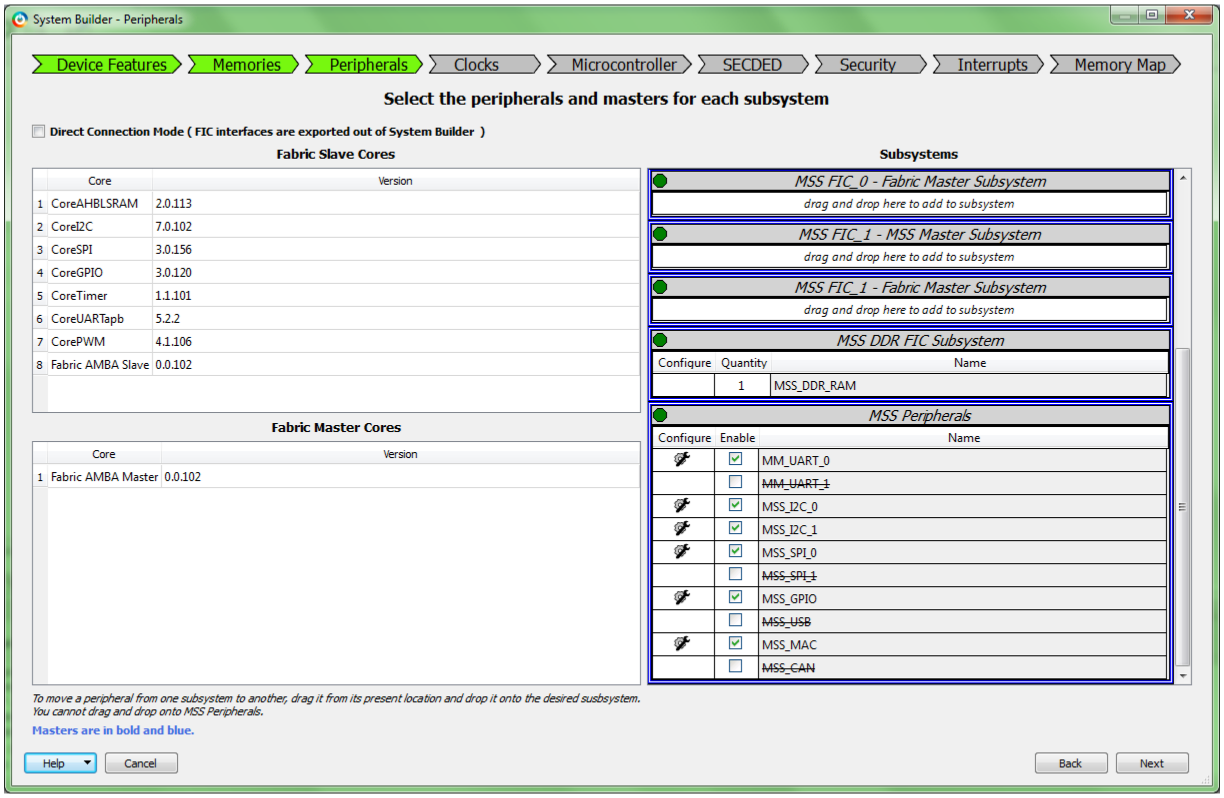
User clients in eNVM

	Client Type	Client Name	DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order	Lock Start Address
1	Data Storage	uboot	91916 x 8	0	0	718	N/A	☐

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**



System Builder - Peripherals

Device Features > Memories > Peripherals > Clocks > Microcontroller > SECDED > Security > Interrupts > Memory Map

Select the peripherals and masters for each subsystem

☐ **Direct Connection Mode (FIC interfaces are exported out of System Builder)**

Fabric Slave Cores

	Core	Version
1	CoreAHBLSRAM	2.0.113
2	CoreI2C	7.0.102
3	CoreSPI	3.0.156
4	CoreGPIO	3.0.120
5	CoreTimer	1.1.101
6	CoreUARTapb	5.2.2
7	CorePWM	4.1.106
8	Fabric AMBA Slave	0.0.102

Fabric Master Cores

	Core	Version
1	Fabric AMBA Master	0.0.102

Subsystems

MSS FIC_0 - Fabric Master Subsystem

drag and drop here to add to subsystem

MSS FIC_1 - MSS Master Subsystem

drag and drop here to add to subsystem

MSS FIC_1 - Fabric Master Subsystem

drag and drop here to add to subsystem

MSS DDR FIC Subsystem

Configure	Quantity	Name
	1	MSS_DDR_RAM

MSS Peripherals

Configure	Enable	Name
🔧	✓	MM_UART_0
	☐	~~MM_UART_1~~
🔧	✓	MSS_I2C_0
🔧	✓	MSS_I2C_1
🔧	✓	MSS_SPI_0
	☐	~~MSS_SPI_1~~
🔧	✓	MSS_GPIO
	☐	~~MSS_USB~~
🔧	✓	MSS_MAC
	☐	~~MSS_CAN~~

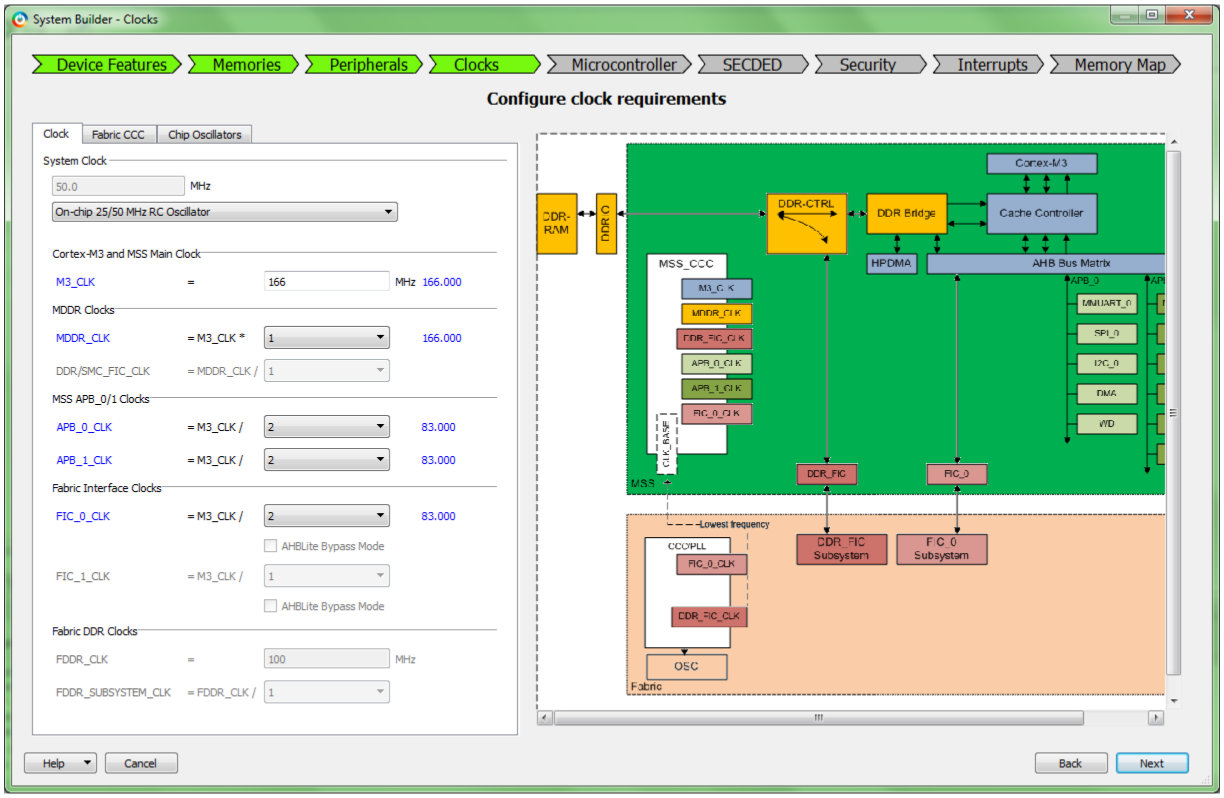
To move a peripheral from one subsystem to another, drag it from its present location and drop it onto the desired susbsystem.
You cannot drag and drop onto MSS Peripherals.

Masters are in bold and blue.

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**



System Clock
50.0 MHz
On-chip 25/50 MHz RC Oscillator

Cortex-M3 and MSS Main Clock
M3_CLK = 166 MHz 166.000

MDDR Clocks
MDDR_CLK = M3_CLK * 1 166.000
DDR/SMC_FIC_CLK = MDDR_CLK / 1

MSS APB_0/1 Clocks
APB_0_CLK = M3_CLK / 2 83.000
APB_1_CLK = M3_CLK / 2 83.000

Fabric Interface Clocks
FIC_0_CLK = M3_CLK / 2 83.000
AHBLite Bypass Mode
FIC_1_CLK = M3_CLK / 1
AHBLite Bypass Mode

Fabric DDR Clocks
FDDR_CLK = 100 MHz
FDDR_SUBSYSTEM_CLK = FDDR_CLK / 1

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
```

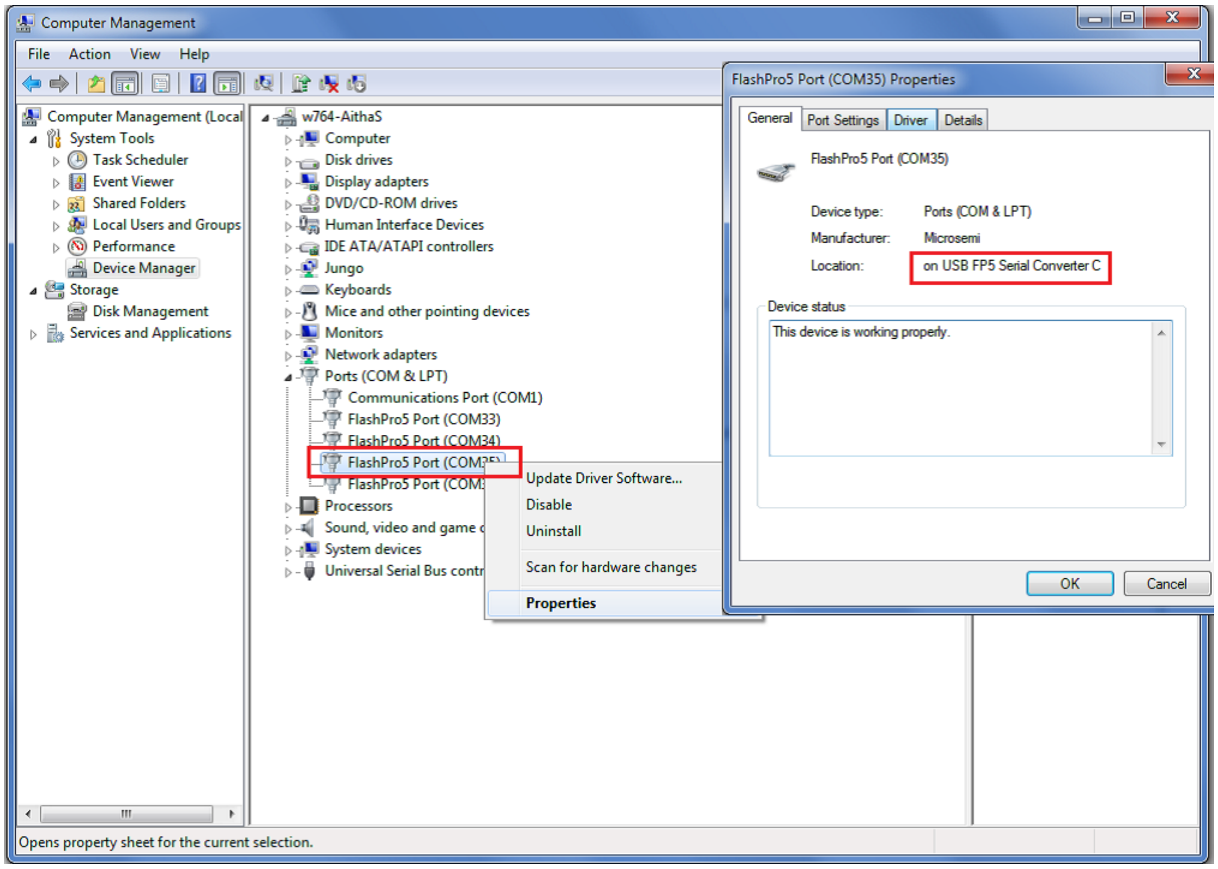
The networking.uImage is generated after successful compilation of networking project.

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**



FlashPro5 Port (COM35)

Device type: Ports (COM & LPT)
Manufacturer: Microsemi
Location: on USB FP5 Serial Converter C

Device status
This device is working properly.

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.

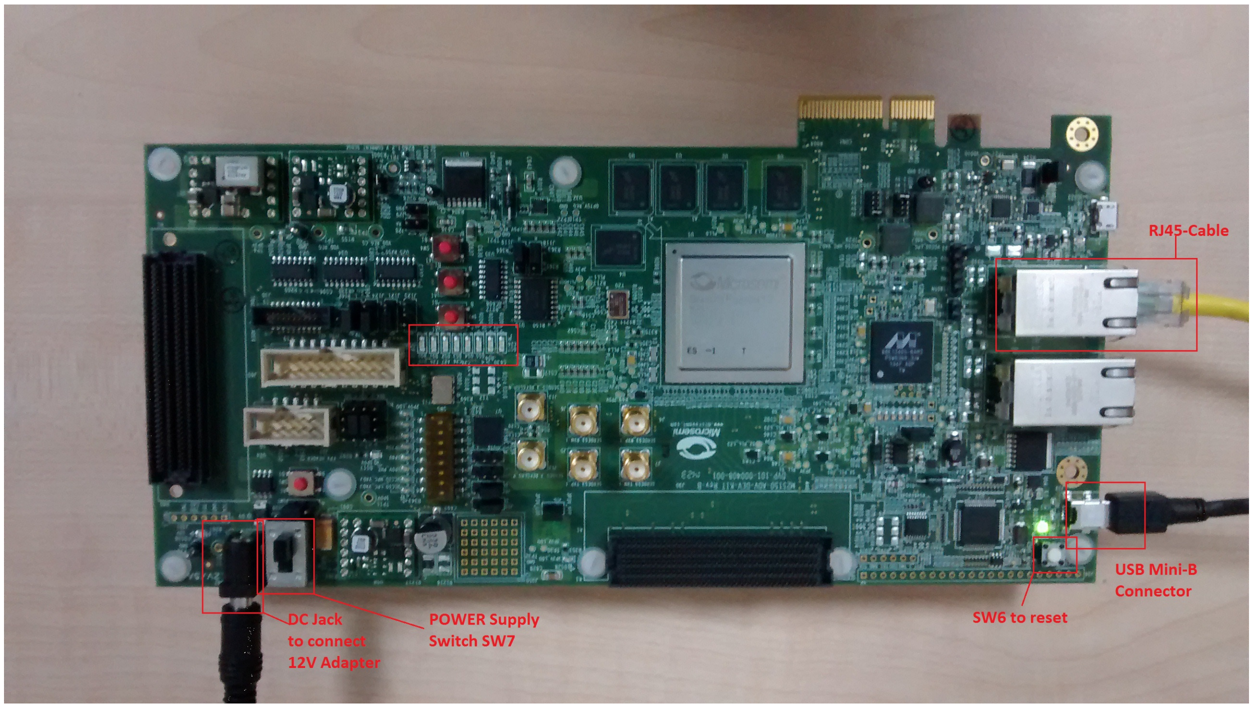
Table 3 • **SmartFusion2 SoC Advanced Development Kit Jumper Settings**

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**



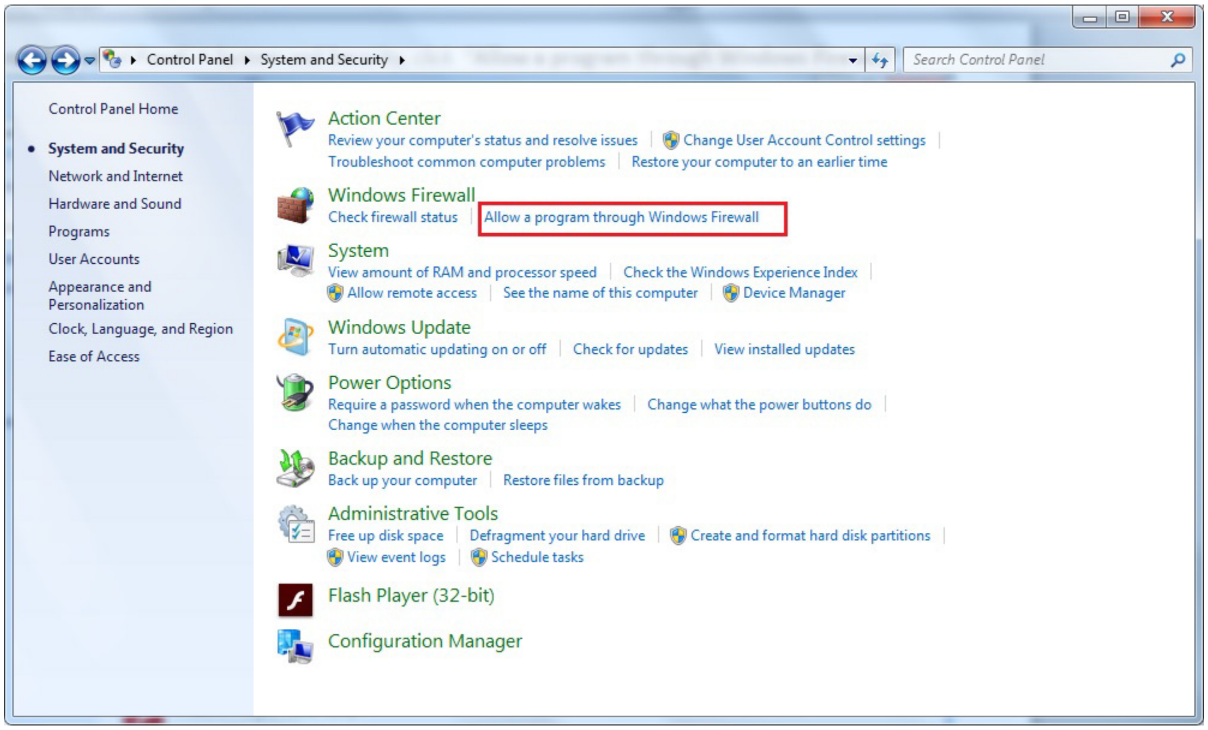
2.7.1 Setting Up the TFTP Server Application in Host PC

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.

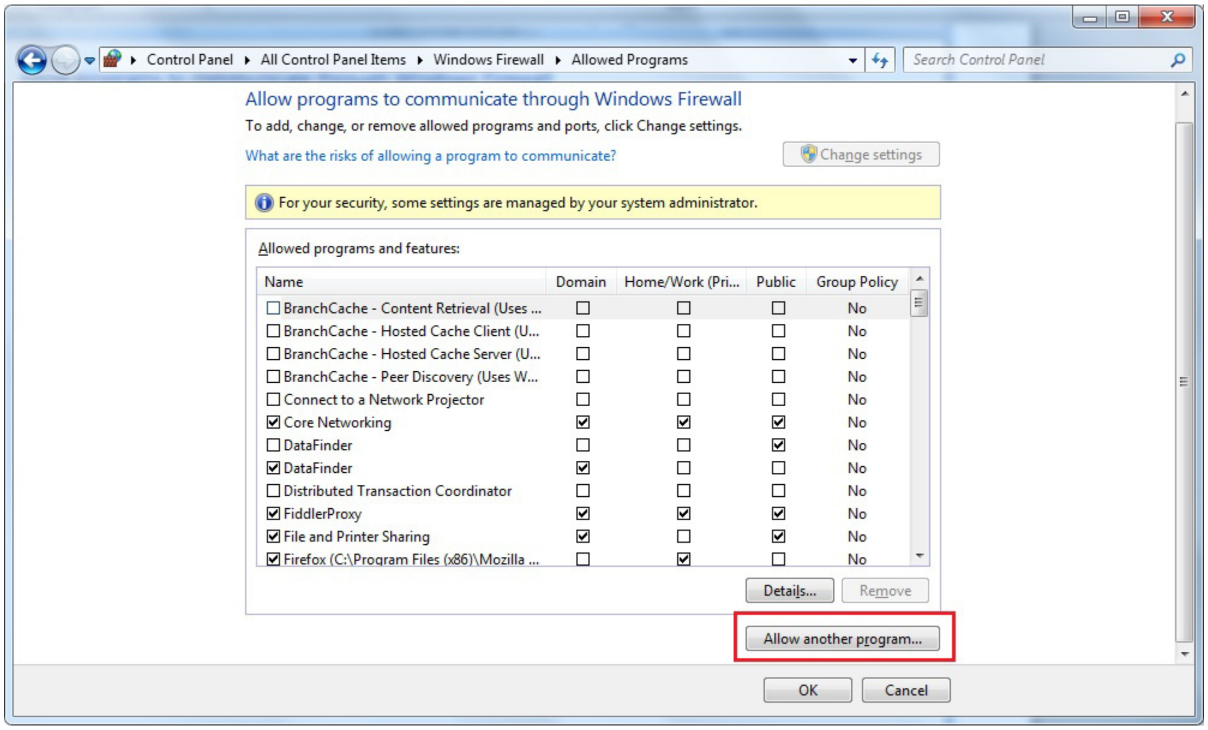
   a. Navigate to **Control Panel > System and Security**, click **Allow a program through Windows Firewall**.

Figure 10 • **System and Security Window**


Control Panel ▸ System and Security ▸
Search Control Panel
Control Panel Home
System and Security
Network and Internet
Hardware and Sound
Programs
User Accounts
Appearance and Personalization
Clock, Language, and Region
Ease of Access
Action Center
Review your computer's status and resolve issues | Change User Account Control settings | Troubleshoot common computer problems | Restore your computer to an earlier time
Windows Firewall
Check firewall status | Allow a program through Windows Firewall
System
View amount of RAM and processor speed | Check the Windows Experience Index | Allow remote access | See the name of this computer | Device Manager
Windows Update
Turn automatic updating on or off | Check for updates | View installed updates
Power Options
Require a password when the computer wakes | Change what the power buttons do | Change when the computer sleeps
Backup and Restore
Back up your computer | Restore files from backup
Administrative Tools
Free up disk space | Defragment your hard drive | Create and format hard disk partitions | View event logs | Schedule tasks
Flash Player (32-bit)
Configuration Manager

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

Figure 11 • **Choose Allow Another Program**



Control Panel ▸ All Control Panel Items ▸ Windows Firewall ▸ Allowed Programs

Allow programs to communicate through Windows Firewall

To add, change, or remove allowed programs and ports, click Change settings.

What are the risks of allowing a program to communicate?

For your security, some settings are managed by your system administrator.

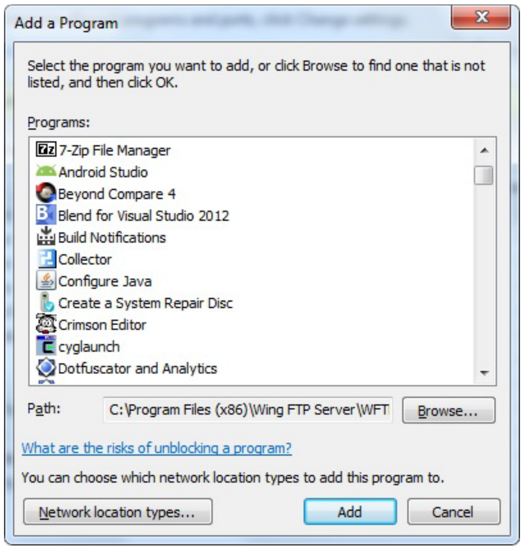
Allowed programs and features:

Name	Domain	Home/Work (Pri...	Public	Group Policy
☐ BranchCache - Content Retrieval (Uses ...	☐	☐	☐	No
☐ BranchCache - Hosted Cache Client (U...	☐	☐	☐	No
☐ BranchCache - Hosted Cache Server (U...	☐	☐	☐	No
☐ BranchCache - Peer Discovery (Uses W...	☐	☐	☐	No
☐ Connect to a Network Projector	☐	☐	☐	No
☑ Core Networking	☑	☑	☑	No
☐ DataFinder	☐	☐	☑	No
☑ DataFinder	☑	☐	☐	No
☐ Distributed Transaction Coordinator	☐	☐	☐	No
☑ FiddlerProxy	☑	☑	☑	No
☑ File and Printer Sharing	☑	☐	☑	No
☑ Firefox (C:\Program Files (x86)\Mozilla ...	☐	☑	☐	No

Add a Program window is displayed.

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

Figure 12 • **Add a Program Window**



Add a Program

Select the program you want to add, or click Browse to find one that is not listed, and then click OK.

Programs:

7-Zip File Manager
Android Studio
Beyond Compare 4
Blend for Visual Studio 2012
Build Notifications
Collector
Configure Java
Create a System Repair Disc
Crimson Editor
cyglaunch
Dotfuscator and Analytics

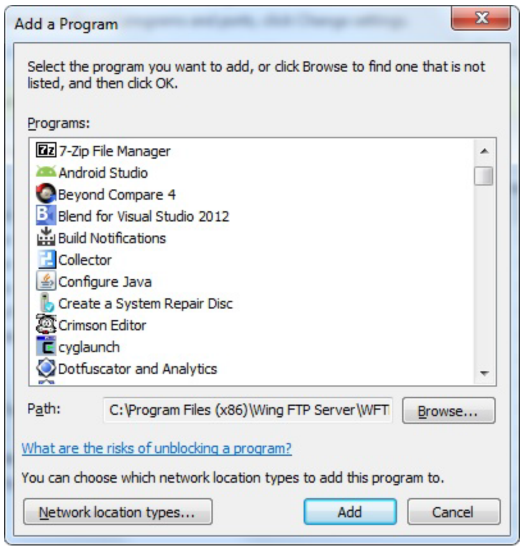
Path: C:\Program Files (x86)\Wing FTP Server\WFT

What are the risks of unblocking a program?

You can choose which network location types to add this program to.

d. Click **Add**.

Figure 13 • **Adding TFTP Server Application**



Add a Program

Select the program you want to add, or click Browse to find one that is not listed, and then click OK.

Programs:

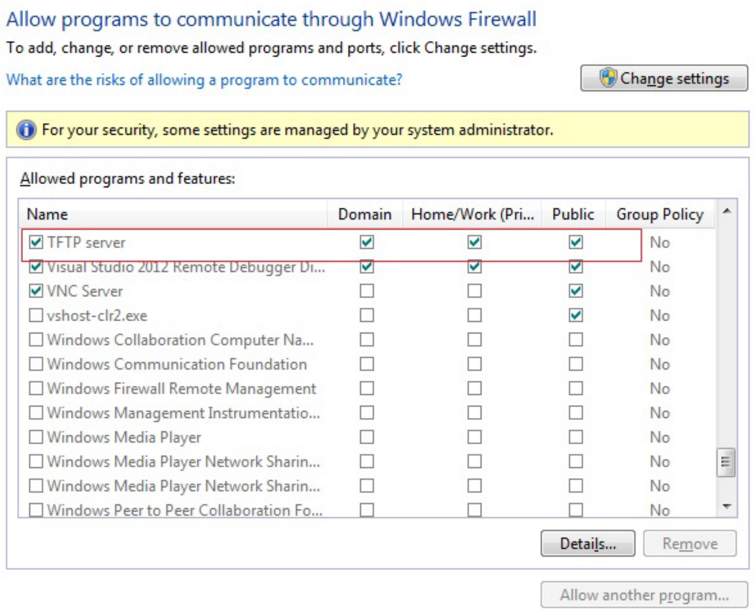
- 7-Zip File Manager
- Android Studio
- Beyond Compare 4
- Blend for Visual Studio 2012
- Build Notifications
- Collector
- Configure Java
- Create a System Repair Disc
- Crimson Editor
- cyglaunch
- Dotfuscator and Analytics

Path: C:\Program Files (x86)\Wing FTP Server\WFT

You can choose which network location types to add this program to.

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

Figure 14 • **TFTP Server Application**



Allow programs to communicate through Windows Firewall

To add, change, or remove allowed programs and ports, click Change settings.

For your security, some settings are managed by your system administrator.

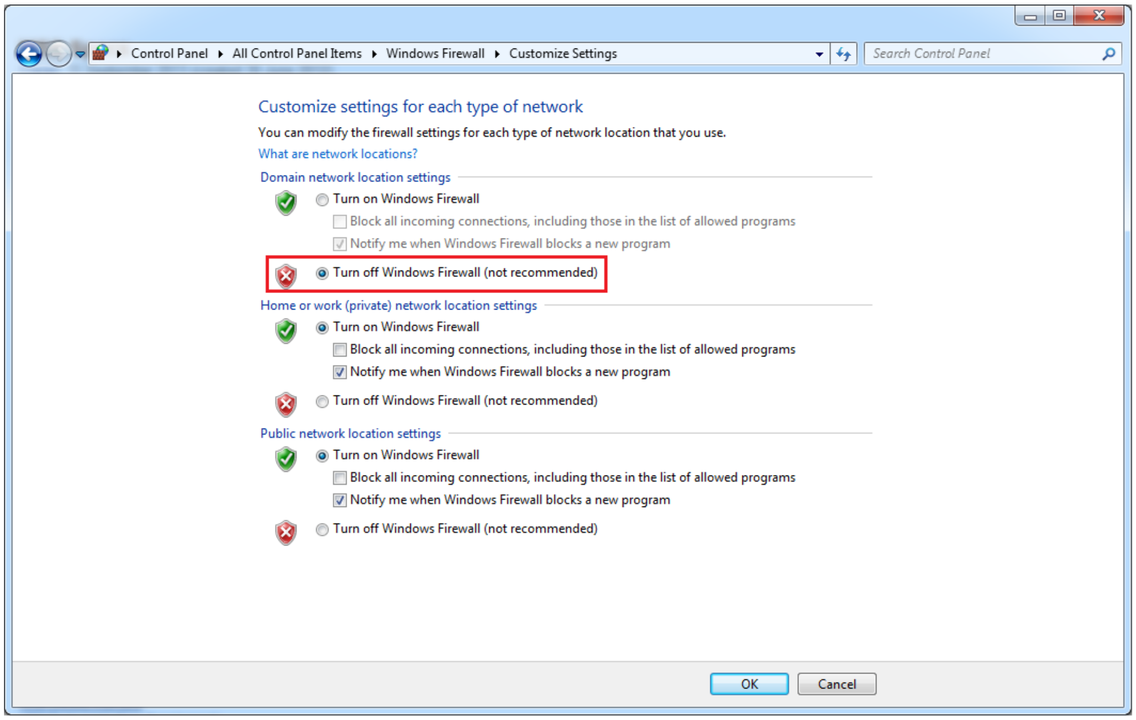
Allowed programs and features:

Name	Domain	Home/Work (Pri...	Public	Group Policy
☑ TFTP server	☑	☑	☑	No
☑ Visual Studio 2012 Remote Debugger Di...	☑	☑	☑	No
☑ VNC Server	☐	☐	☑	No
☐ vshost-clr2.exe	☐	☐	☑	No
☐ Windows Collaboration Computer Na...	☐	☐	☐	No
☐ Windows Communication Foundation	☐	☐	☐	No
☐ Windows Firewall Remote Management	☐	☐	☐	No
☐ Windows Management Instrumentatio...	☐	☐	☐	No
☐ Windows Media Player	☐	☐	☐	No
☐ Windows Media Player Network Sharin...	☐	☐	☐	No
☐ Windows Media Player Network Sharin...	☐	☐	☐	No
☐ Windows Peer to Peer Collaboration Fo...	☐	☐	☐	No

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**



Control Panel ▸ All Control Panel Items ▸ Windows Firewall ▸ Customize Settings

Customize settings for each type of network

You can modify the firewall settings for each type of network location that you use.

What are network locations?

Domain network location settings

- ○ Turn on Windows Firewall
  - ☐ Block all incoming connections, including those in the list of allowed programs
  - ☑ Notify me when Windows Firewall blocks a new program
- ● Turn off Windows Firewall (not recommended)

Home or work (private) network location settings

- ● Turn on Windows Firewall
  - ☐ Block all incoming connections, including those in the list of allowed programs
  - ☑ Notify me when Windows Firewall blocks a new program
- ○ Turn off Windows Firewall (not recommended)

Public network location settings

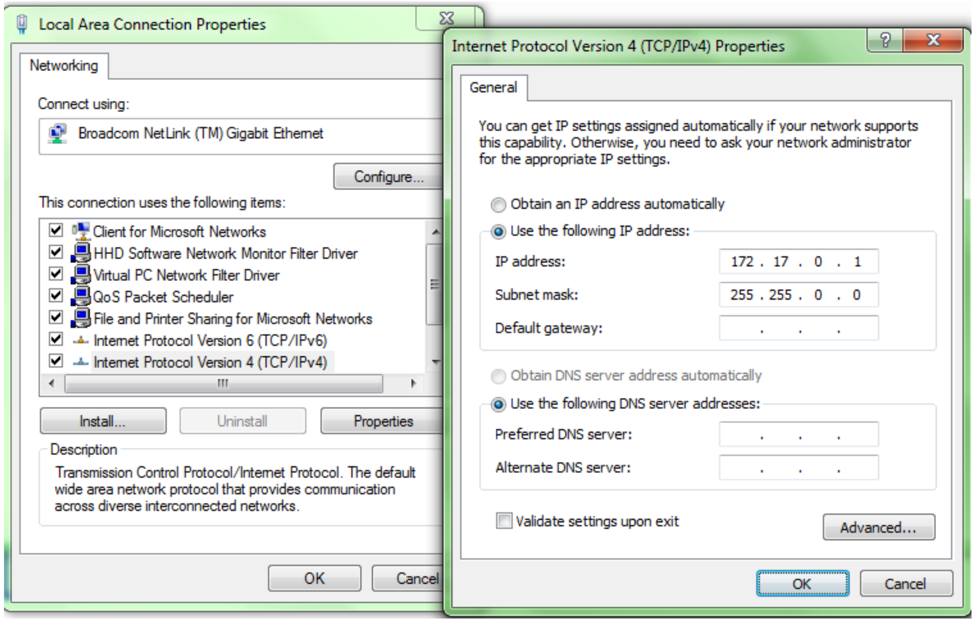
- ● Turn on Windows Firewall
  - ☐ Block all incoming connections, including those in the list of allowed programs
  - ☑ Notify me when Windows Firewall blocks a new program
- ○ Turn off Windows Firewall (not recommended)

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**



Local Area Connection Properties

Networking

Connect using:

Broadcom NetLink (TM) Gigabit Ethernet

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ HHD Software Network Monitor Filter Driver
- ☑ Virtual PC Network Filter Driver
- ☑ QoS Packet Scheduler
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

- ○ Obtain an IP address automatically
- ● Use the following IP address:

IP address:	172 . 17 . 0 . 1
Subnet mask:	255 . 255 . 0 . 0
Default gateway:	. . .

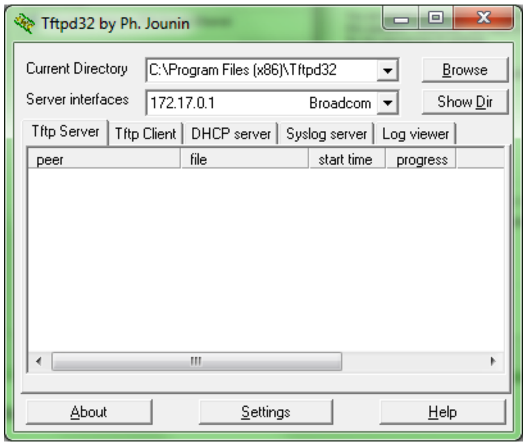
- ○ Obtain DNS server address automatically
- ● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

☐ Validate settings upon exit

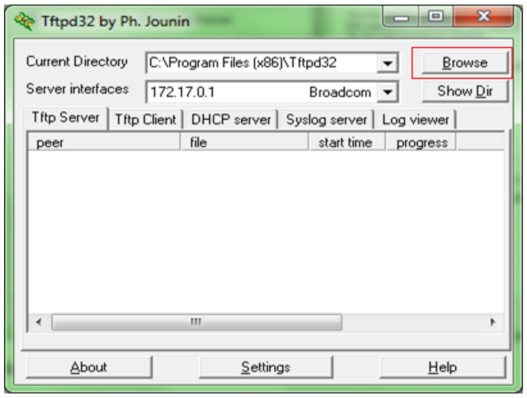
2. Invoke the Tftpd server application from **All Programs>Tftpd32>**, the TFTP server application is displayed.

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**



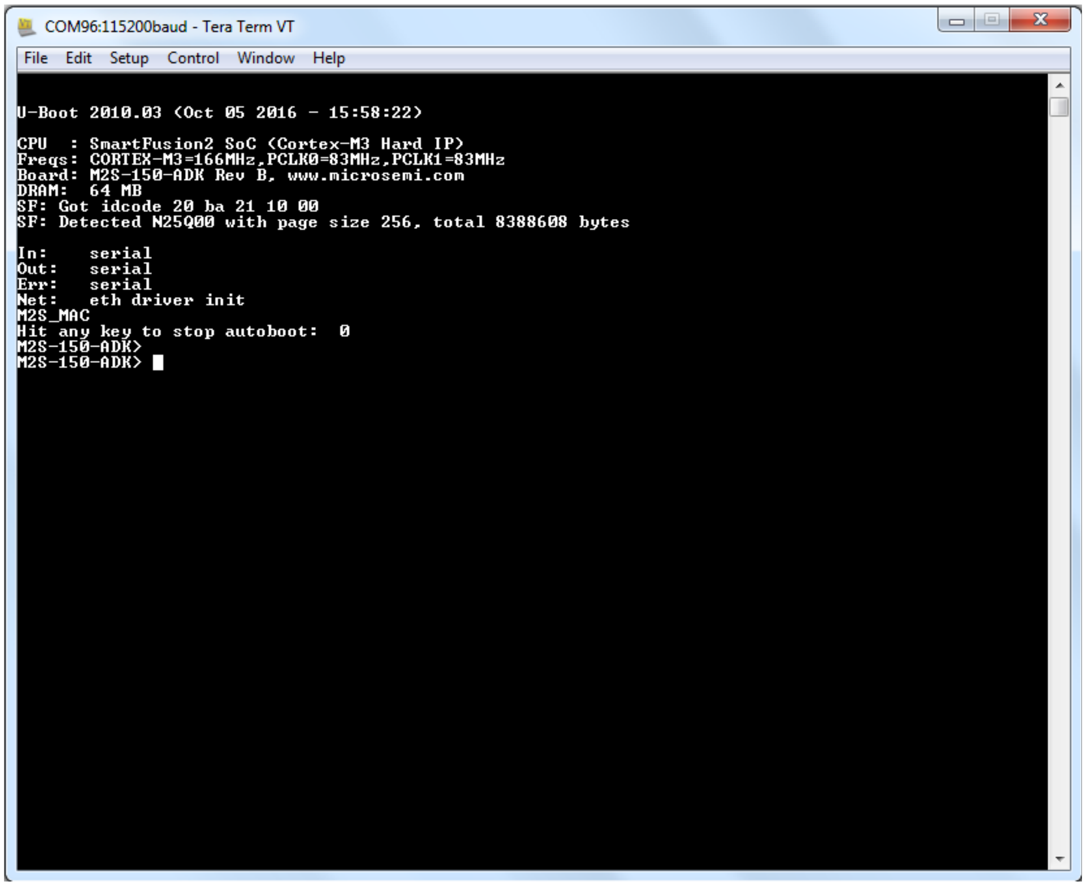
2.8 Running the Design

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**


COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

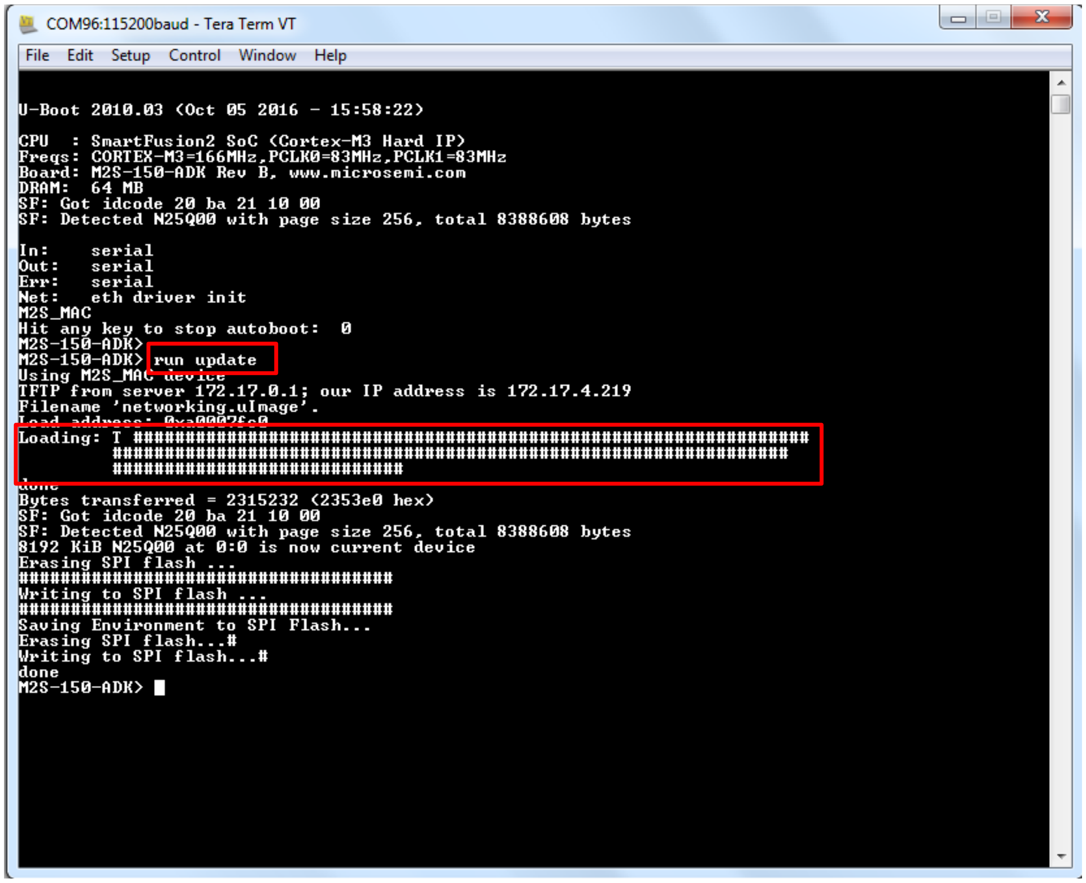
```
U-Boot 2010.03 (Oct 05 2016 - 15:58:22)

CPU  : SmartFusion2 SoC (Cortex-M3 Hard IP)
Freqs: CORTEX-M3=166MHz,PCLK0=83MHz,PCLK1=83MHz
Board: M2S-150-ADK Rev B, www.microsemi.com
DRAM:  64 MB
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes

In:    serial
Out:   serial
Err:   serial
Net:   eth driver init
M2S_MAC
Hit any key to stop autoboot:  0
M2S-150-ADK>
M2S-150-ADK>
```

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**



COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
U-Boot 2010.03 (Oct 05 2016 - 15:58:22)

CPU  : SmartFusion2 SoC (Cortex-M3 Hard IP)
Freqs: CORTEX-M3=166MHz,PCLK0=83MHz,PCLK1=83MHz
Board: M2S-150-ADK Rev B, www.microsemi.com
DRAM:  64 MB
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes

In:    serial
Out:   serial
Err:   serial
Net:   eth driver init
M2S_MAC
Hit any key to stop autoboot:  0
M2S-150-ADK>
M2S-150-ADK> run update
Using M2S_MAC device
TFTP from server 172.17.0.1; our IP address is 172.17.4.219
Filename 'networking.uImage'.
Load address: 0xa0007fc0
Loading: T #################################################################
           #################################################################
           ############################
done
Bytes transferred = 2315232 (2353e0 hex)
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes
8192 KiB N25Q00 at 0:0 is now current device
Erasing SPI flash ...
####################################
Writing to SPI flash ...
####################################
Saving Environment to SPI Flash...
Erasing SPI flash...#
Writing to SPI flash...#
done
M2S-150-ADK>
```

Linux Kernel image copies to SPI flash.

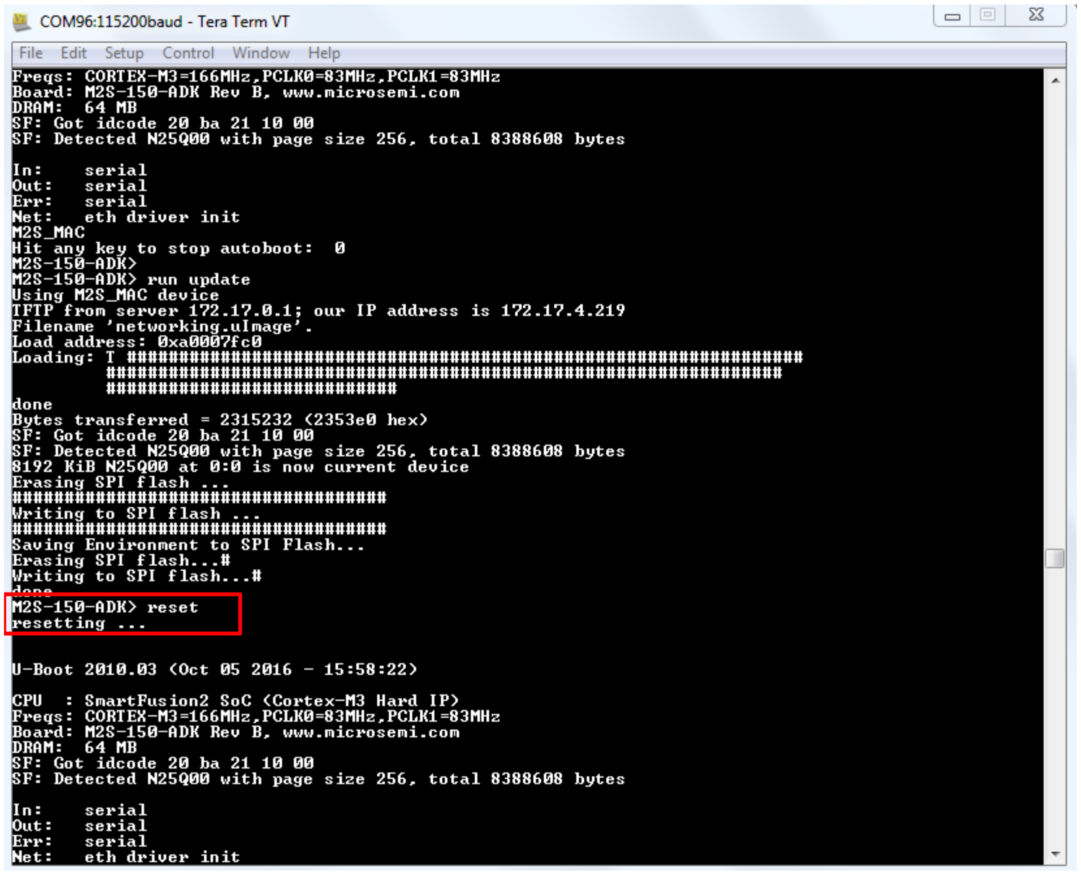
2.8.3 Booting the Linux Kernel

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**

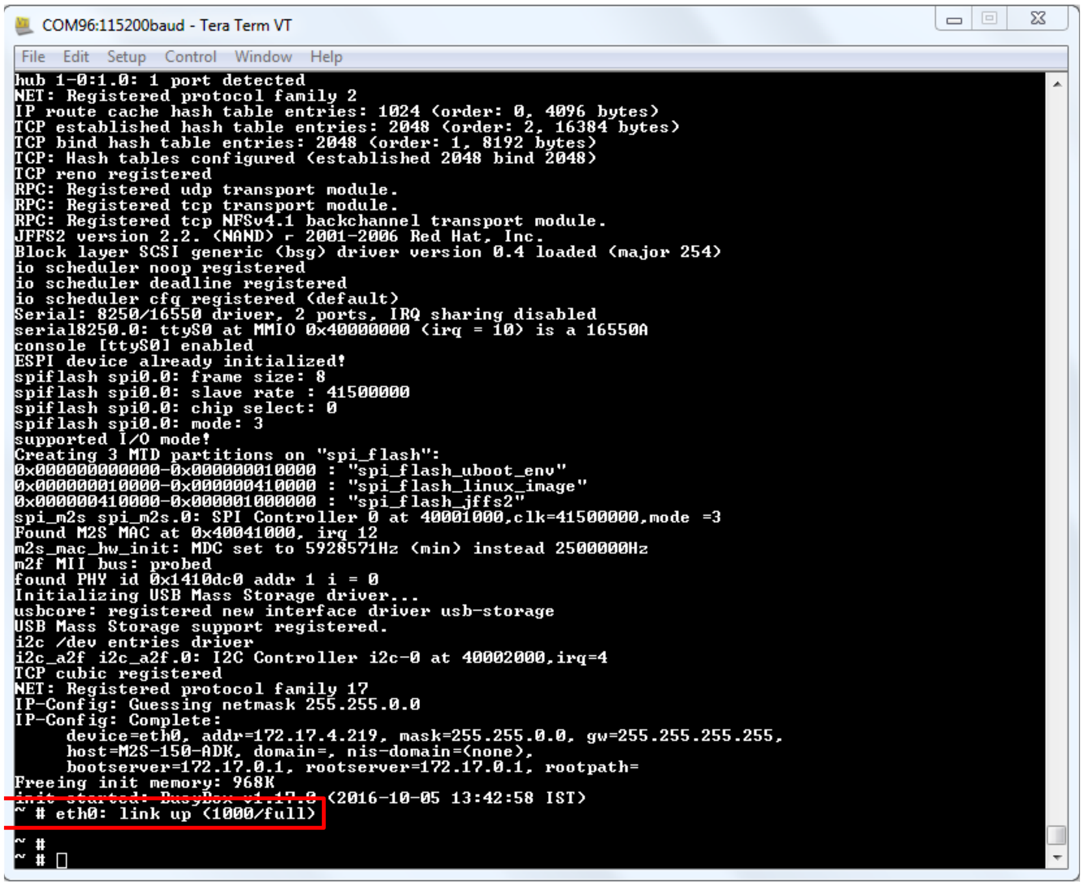


COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Freqs: CORTEX-M3=166MHz,PCLK0=83MHz,PCLK1=83MHz
Board: M2S-150-ADK Rev B, www.microsemi.com
DRAM:  64 MB
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes

In:    serial
Out:   serial
Err:   serial
Net:   eth driver init
M2S_MAC
Hit any key to stop autoboot:  0
M2S-150-ADK>
M2S-150-ADK> run update
Using M2S_MAC device
TFTP from server 172.17.0.1; our IP address is 172.17.4.219
Filename 'networking.uImage'.
Load address: 0xa0007fc0
Loading: T #################################################################
         #################################################################
         ############################
done
Bytes transferred = 2315232 (2353e0 hex)
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes
8192 KiB N25Q00 at 0:0 is now current device
Erasing SPI flash ...
####################################
Writing to SPI flash ...
####################################
Saving Environment to SPI Flash...
Erasing SPI flash...#
Writing to SPI flash...#
done
M2S-150-ADK> reset
resetting ...


U-Boot 2010.03 (Oct 05 2016 - 15:58:22)

CPU   : SmartFusion2 SoC (Cortex-M3 Hard IP)
Freqs: CORTEX-M3=166MHz,PCLK0=83MHz,PCLK1=83MHz
Board: M2S-150-ADK Rev B, www.microsemi.com
DRAM:  64 MB
SF: Got idcode 20 ba 21 10 00
SF: Detected N25Q00 with page size 256, total 8388608 bytes

In:    serial
Out:   serial
Err:   serial
Net:   eth driver init
```

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

Figure 22 • **uClinux Prompt**


COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
hub 1-0:1.0: 1 port detected
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
JFFS2 version 2.2. (NAND) r 2001-2006 Red Hat, Inc.
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, 2 ports, IRQ sharing disabled
serial8250.0: ttyS0 at MMIO 0x40000000 (irq = 10) is a 16550A
console [ttyS0] enabled
ESPI device already initialized!
spiflash spi0.0: frame size: 8
spiflash spi0.0: slave rate : 41500000
spiflash spi0.0: chip select: 0
spiflash spi0.0: mode: 3
supported I/O mode!
Creating 3 MTD partitions on "spi_flash":
0x000000000000-0x000000010000 : "spi_flash_uboot_env"
0x000000010000-0x000000410000 : "spi_flash_linux_image"
0x000000410000-0x000001000000 : "spi_flash_jffs2"
spi_m2s spi_m2s.0: SPI Controller 0 at 40001000,clk=41500000,mode =3
Found M2S MAC at 0x40041000, irq 12
m2s_mac_hw_init: MDC set to 5928571Hz (min) instead 2500000Hz
m2f MII bus: probed
found PHY id 0x1410dc0 addr 1 i = 0
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
i2c /dev entries driver
i2c_a2f i2c_a2f.0: I2C Controller i2c-0 at 40002000,irq=4
TCP cubic registered
NET: Registered protocol family 17
IP-Config: Guessing netmask 255.255.0.0
IP-Config: Complete:
     device=eth0, addr=172.17.4.219, mask=255.255.0.0, gw=255.255.255.255,
     host=M2S-150-ADK, domain=, nis-domain=(none),
     bootserver=172.17.0.1, rootserver=172.17.0.1, rootpath=
Freeing init memory: 968K
init started: BusyBox v1.17.0 (2016-10-05 13:42:58 IST)
~ # eth0: link up (1000/full)

~ #
~ # 
```

2.8.4 Running the Networking Applications

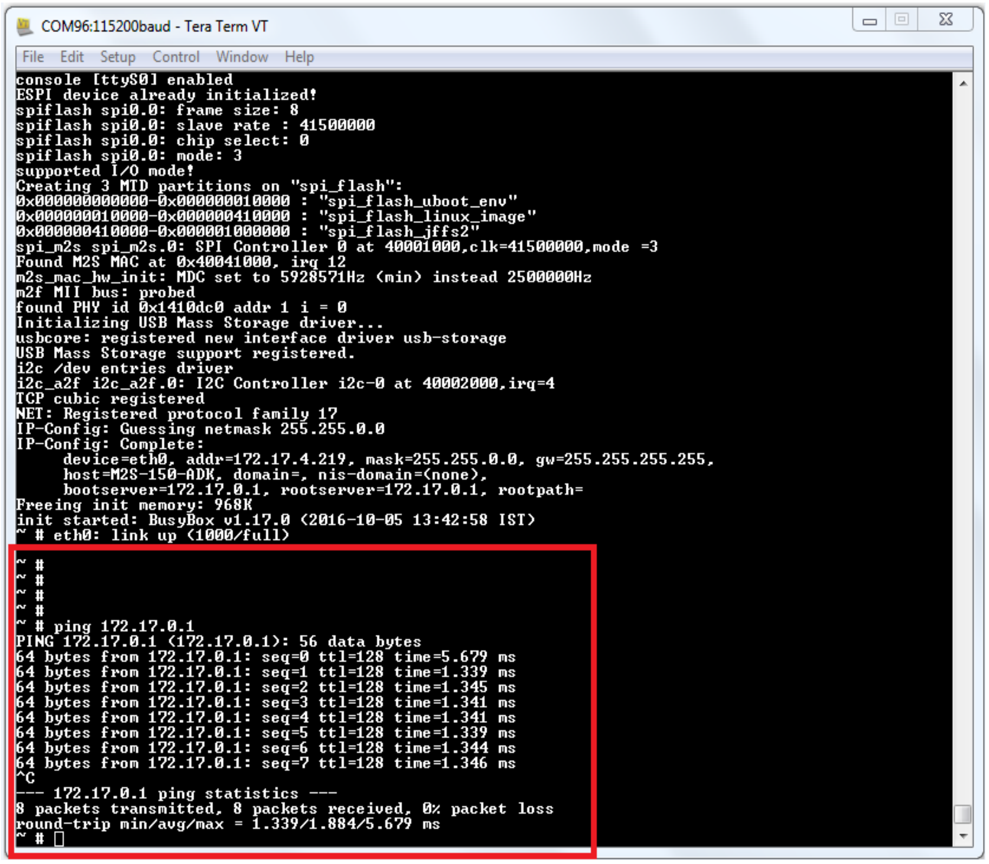
Ping and Http are the applications used to test the networking functionalities:

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**


COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

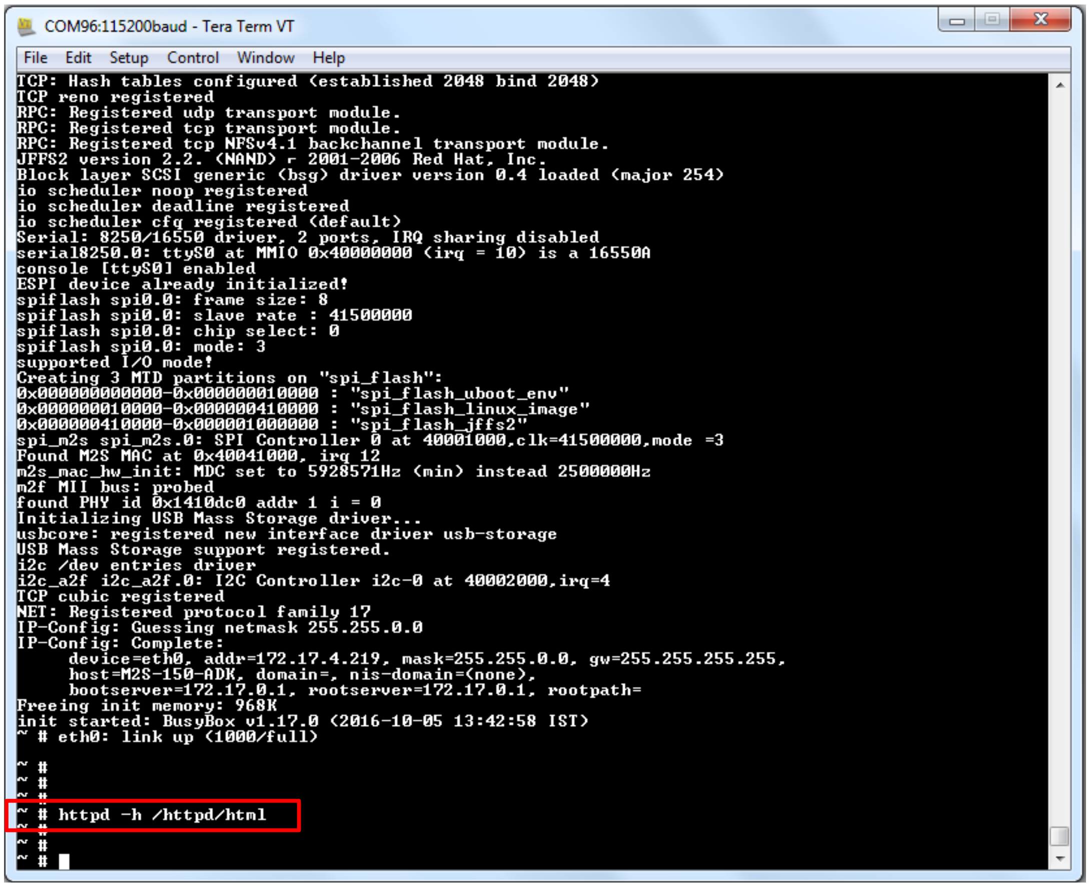
```
console [ttyS0] enabled
ESPI device already initialized!
spiflash spi0.0: frame size: 8
spiflash spi0.0: slave rate : 41500000
spiflash spi0.0: chip select: 0
spiflash spi0.0: mode: 3
supported I/O mode!
Creating 3 MTD partitions on "spi_flash":
0x000000000000-0x000000010000 : "spi_flash_uboot_env"
0x000000010000-0x000000410000 : "spi_flash_linux_image"
0x000000410000-0x000001000000 : "spi_flash_jffs2"
spi_m2s spi_m2s.0: SPI Controller 0 at 40001000,clk=41500000,mode =3
Found M2S MAC at 0x40041000, irq 12
m2s_mac_hw_init: MDC set to 5928571Hz (min) instead 2500000Hz
m2f MII bus: probed
found PHY id 0x1410dc0 addr 1 i = 0
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
i2c /dev entries driver
i2c_a2f i2c_a2f.0: I2C Controller i2c-0 at 40002000,irq=4
TCP cubic registered
NET: Registered protocol family 17
IP-Config: Guessing netmask 255.255.0.0
IP-Config: Complete:
     device=eth0, addr=172.17.4.219, mask=255.255.0.0, gw=255.255.255.255,
     host=M2S-150-ADK, domain=, nis-domain=(none),
     bootserver=172.17.0.1, rootserver=172.17.0.1, rootpath=
Freeing init memory: 968K
init started: BusyBox v1.17.0 (2016-10-05 13:42:58 IST)
~ # eth0: link up (1000/full)

~ #
~ #
~ #
~ #
~ # ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=128 time=5.679 ms
64 bytes from 172.17.0.1: seq=1 ttl=128 time=1.339 ms
64 bytes from 172.17.0.1: seq=2 ttl=128 time=1.345 ms
64 bytes from 172.17.0.1: seq=3 ttl=128 time=1.341 ms
64 bytes from 172.17.0.1: seq=4 ttl=128 time=1.341 ms
64 bytes from 172.17.0.1: seq=5 ttl=128 time=1.339 ms
64 bytes from 172.17.0.1: seq=6 ttl=128 time=1.344 ms
64 bytes from 172.17.0.1: seq=7 ttl=128 time=1.346 ms
^C
--- 172.17.0.1 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 1.339/1.884/5.679 ms
~ #
```

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**

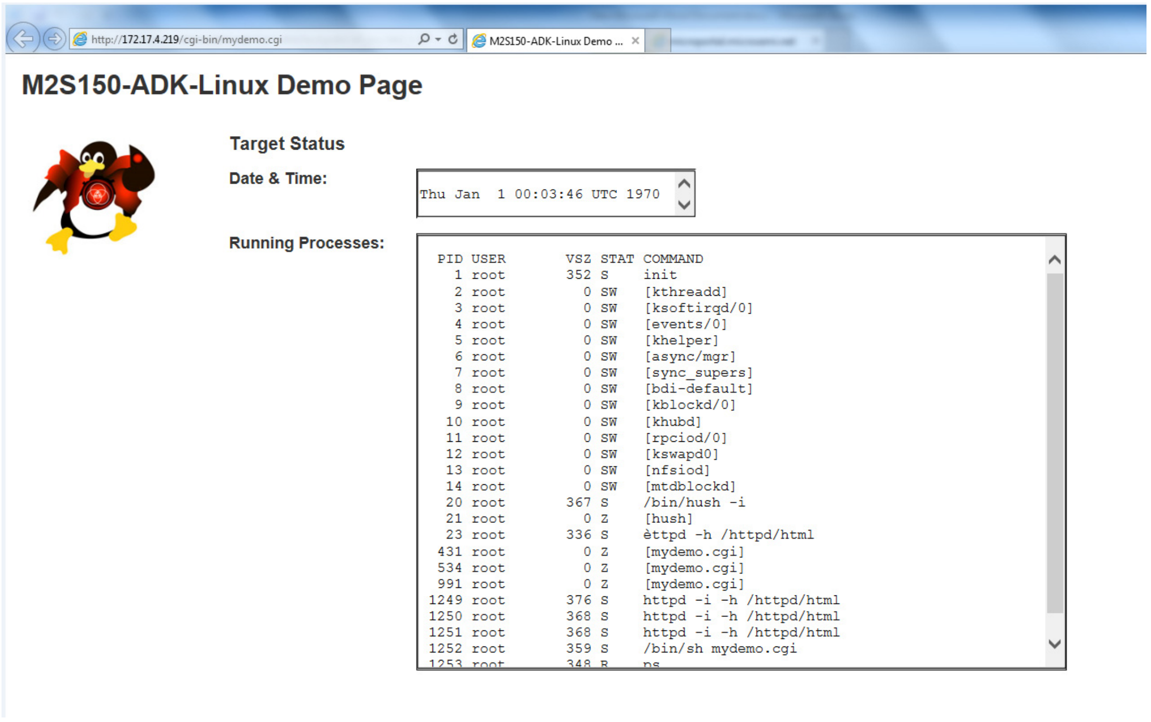


COM96:115200baud - Tera Term VT

```
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
JFFS2 version 2.2. (NAND) ┌ 2001-2006 Red Hat, Inc.
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, 2 ports, IRQ sharing disabled
serial8250.0: ttyS0 at MMIO 0x40000000 (irq = 10) is a 16550A
console [ttyS0] enabled
ESPI device already initialized!
spiflash spi0.0: frame size: 8
spiflash spi0.0: slave rate : 41500000
spiflash spi0.0: chip select: 0
spiflash spi0.0: mode: 3
supported I/O mode!
Creating 3 MTD partitions on "spi_flash":
0x000000000000-0x000000010000 : "spi_flash_uboot_env"
0x000000010000-0x000000410000 : "spi_flash_linux_image"
0x000000410000-0x000001000000 : "spi_flash_jffs2"
spi_m2s spi_m2s.0: SPI Controller 0 at 40001000,clk=41500000,mode =3
Found M2S MAC at 0x40041000, irq 12
m2s_mac_hw_init: MDC set to 5928571Hz (min) instead 2500000Hz
m2f MII bus: probed
found PHY id 0x1410dc0 addr 1 i = 0
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
i2c /dev entries driver
i2c_a2f i2c_a2f.0: I2C Controller i2c-0 at 40002000,irq=4
TCP cubic registered
NET: Registered protocol family 17
IP-Config: Guessing netmask 255.255.0.0
IP-Config: Complete:
     device=eth0, addr=172.17.4.219, mask=255.255.0.0, gw=255.255.255.255,
     host=M2S-150-ADK, domain=, nis-domain=(none),
     bootserver=172.17.0.1, rootserver=172.17.0.1, rootpath=
Freeing init memory: 968K
init started: BusyBox v1.17.0 (2016-10-05 13:42:58 IST)
~ # eth0: link up (1000/full)

~ #
~ #
~ #
~ # httpd -h /httpd/html
~ #
~ #
~ #
```

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**


http://172.17.4.219/cgi-bin/mydemo.cgi
M2S150-ADK-Linux Demo ...

M2S150-ADK-Linux Demo Page

Target Status

Date & Time:

```
Thu Jan  1 00:03:46 UTC 1970
```

Running Processes:

```
  PID USER       VSZ STAT COMMAND
    1 root       352 S    init
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]
    4 root         0 SW   [events/0]
    5 root         0 SW   [khelper]
    6 root         0 SW   [async/mgr]
    7 root         0 SW   [sync_supers]
    8 root         0 SW   [bdi-default]
    9 root         0 SW   [kblockd/0]
   10 root         0 SW   [khubd]
   11 root         0 SW   [rpciod/0]
   12 root         0 SW   [kswapd0]
   13 root         0 SW   [nfsiod]
   14 root         0 SW   [mtdblockd]
   20 root       367 S    /bin/hush -i
   21 root         0 Z    [hush]
   23 root       336 S    èttpd -h /httpd/html
  431 root         0 Z    [mydemo.cgi]
  534 root         0 Z    [mydemo.cgi]
  991 root         0 Z    [mydemo.cgi]
 1249 root       376 S    httpd -i -h /httpd/html
 1250 root       368 S    httpd -i -h /httpd/html
 1251 root       368 S    httpd -i -h /httpd/html
 1252 root       359 S    /bin/sh mydemo.cgi
 1253 root       348 R    ps
```

2.9 Conclusion

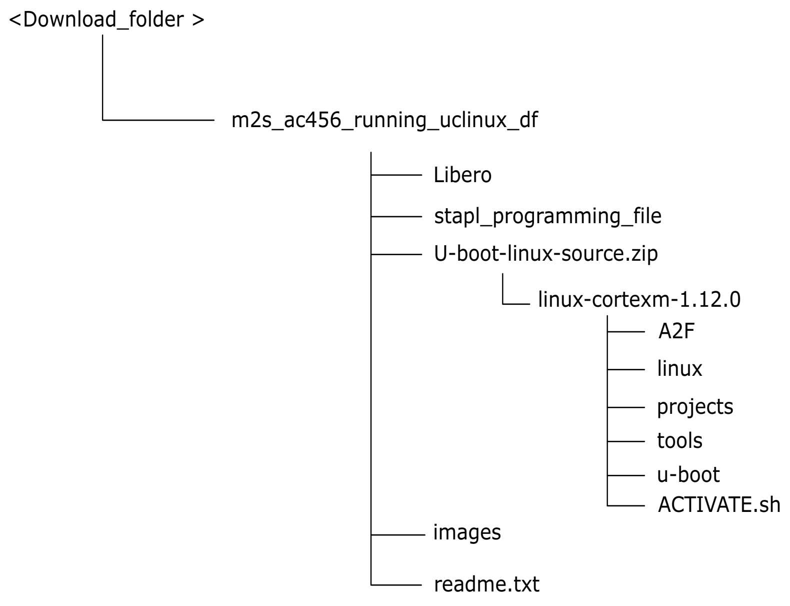
This application describes how to compile and boot uClinux and shows the networking applications on the SmartFusion2 Advanced Development Kit.

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