
Freescale MQX ETHERNET BOOTLOADER User's Guide

Document Number:
MQXBLDUG Rev.1.4
07/2014



Chapter 1 Before You Begin	4
1.1 About This Book	4
1.2 Where to Look for More Information.....	4
1.3 Product Names.....	4
1.4 Typographic Conventions.....	4
Chapter 2 Using Ethernet Boot Loader	1
2.1 Ethernet Boot Loader at a glance	1
2.2 Ethernet boot loader flow	1
2.3 Boot loader Memory Maps.....	2
2.3.1 TWR-MCF54418 Memory Map.....	3
Chapter 3 Reference: Functions	4
3.1 _bootloader_init_table	4
3.2 _bootloader_check_image	4
3.3 _bootloader_del_image	4
3.4 _bootloader_del_table	4
3.5 _bootloader_store_image_data.....	5
3.6 _bootloader_list_image	5
3.7 _bootloader_set_default.....	5
3.8 _bootloader_get_timeout	5
3.9 _bootloader_set_timeout	6
Chapter 4 Reference: Commands.....	7
4.1 ipconfig.....	7
4.2 imgload	7
4.3 imglst	7
4.4 imgdel	7
4.5 tbldel	7
4.6 autoboot	7
4.7 imggo	8
4.8 imgtimeout.....	8
4.9 imgren	8
4.10 imgaddr.....	8
4.11 nandcheck	8
4.12 nandef	8
4.13 Reset	8
Chapter 5 MQX Bootloader Examples	9
5.1 Introduction	9
5.2 Preparing the setup	9
5.2.1 Software Setting up.....	9
5.2.2 Hardware Setting up	9
5.3 Preparing image file.....	10
5.3.1 Preparing image file for TWR-MCF54418.....	10
5.4 Running Boot loader.....	12

Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to <http://www.freescale.com/mqx>.

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
Rev. 1.0	01/2011	Draft coming with MQX 3.6
Rev. 1.1	04/2011	Prototype coming with MQX 3.7.0 official release
Rev. 1.2	06/2011	Update Bootloader commands and Bootloader example
Rev. 1.3	07/2011	Update Bootloader for Nor flash devices
Rev. 1.4	07/2014	Remove some unsupported platforms

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

© Freescale Semiconductor, Inc., 2014. All rights reserved.

Chapter 1 Before You Begin

1.1 About This Book

This book is a guide and reference manual for using the FSLMQX NAND Bootloader, which is a part of Freescale MQX Real-Time Operating System distribution.

1.2 Where to Look for More Information

- The release notes document accompanying the Freescale MQX release provides information that was not available at the time this user's guide was published.
- The *MQX User's Guide* describes how to create embedded applications that use the MQX RTOS.
- The *MQX Reference Manual* describes prototypes for the MQX API.

1.3 Product Names

In this book, we use BLD as the abbreviation for the Freescale MQX Bootloader Embedded File System.

1.4 Typographic Conventions

Throughout this book, we use typographic conventions to distinguish terms.

Font style	Usage	Example
Bold	Function families	The _io_mfs family of functions.
Bold	Function names	_io_mfs_install()
<i>Italic</i>	Data types (simple)	<i>uint32_t</i>
	Data types (complex)	See following example.
Constant-width	Code and code fragments	—
	Data types in prototype definitions	See following example.
	Directives	#include "mfs.h"
	Code and code fragments	
<i>Italic</i>	Filenames and path names	<i>part_mgr.h</i>
<i>Italic</i>	Symbolic parameters that you substitute with your values.	See following example.
UPPERCASE Italic	Symbolic constants	<i>MFS_NO_ERROR</i>

1.5 Other Conventions

1.5.1 Cautions

Cautions tell you about commands or procedures that could have unexpected or undesirable side effects or could be dangerous to your files or your hardware.

CAUTION	If an application calls read and write functions with the partition manager, the file system will be corrupted.
----------------	---

Chapter 2 Using Ethernet Boot Loader

2.1 Ethernet Boot Loader at a glance

Ethernet Boot Loader provides an easy and reliable way to load new user applications to the device that available Nand or Nor flash memory. After loaded, the new user applications will be able to run in the device. However, Ethernet boot loader support user to load more than one application to flash memory and specify booting image. The Ethernet boot loader needs an TFTP server running on PC.

This Ethernet boot loader was specifically written for several families of Freescale microcontrollers that have MQX RTOS support NAND or NOR flash driver and enough memory to store more than one applications.

Tested development boards:

- TWR-MCF54418 with NAND flash support

Tested operating systems:

- Windows XP Pro with Service Pack 2

Tested development tool:

- Code Warrior 10.6

This document intends to help you gain an insight into the Ethernet boot loader and capabilities to develop your own applications. The document targets to firmware application developers who would like to develop the applications using Ethernet boot loader.

2.2 Ethernet boot loader flow

The Boot loader is integrated with an application that performs the product's main functions. At beginning, the boot loader executes and does some simple checks to see which the application should start after a timeout value. The Boot loader should go to main application where user can setup own network and get image from server in first time. The Boot loader supports firmware image in S-record, Code warrior binary and raw binary file formats.

- S-record files are common ASCII files used to specify the program data stored in devices. Freescale's software tool chain called Code warrior generates S-record files and Code warrior binary files automatically when projects are compiled. S-record files have the extension .S19 and Code warrior binary files have the extension .bin.
- Raw binary file is the image file of flash memory.

After the image file has been transferred, the Boot loader will store in a specific area.

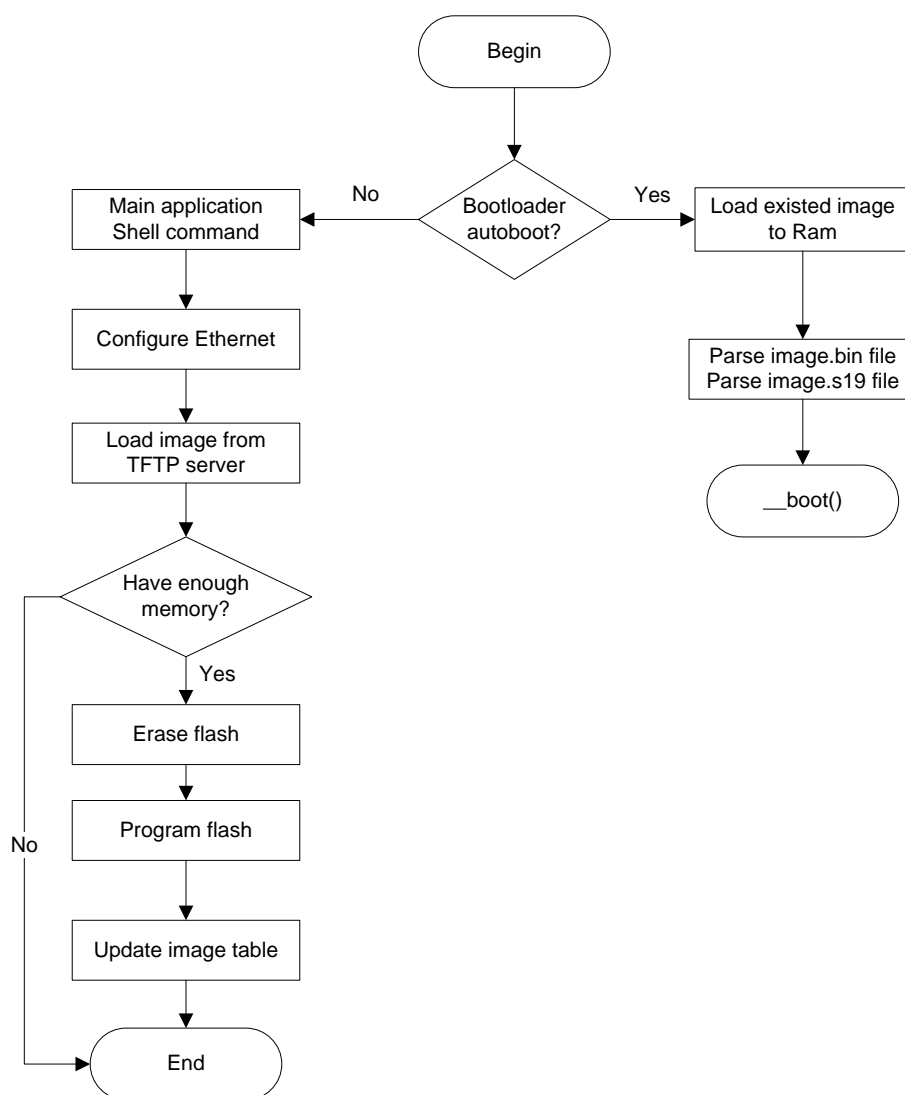


Figure 2-1 Boot loader functional follow chart

When entering to auto boot mode, the Boot loader will parse image file to Ram memory and execute booting from Ram.

2.3 Boot loader Memory Maps

The following sections show the memory maps for the different device examples. The memory map for both the boot loader and application are show to see how the RAM overlaps. Notice the application has access to all RAM and overlaps the bootloader RAM. All other memory sections are identical between the two.

2.3.1 TWR-MCF54418 Memory Map

Table 2-1 TWR-MCF54418 Boot loader and application memory map

Addresses	Boot loader	Application (Ram target only)
0x4000_0000 to 0x4000_0600	Vector table, due to the uboot the start address shifted by 0x100000	Reserved
0x4000_0600 to 0x4010_0000	Code + Const data (cached)	Reserved
0x4010_0000 to 0x4010_0600	Reserved	Vector table, due to the uboot the start address shifted by 0x100000
0x4010_0600 to 0x4300_0000	Reserved	Code + Const data (cached)
0x4300_0000 to 0x4400_0000	RW data cached	Reserved
0x4400_0000 to 0x4500_0000	Reserved	RW data cached
0x4500_0000 to 0x4600_0000	RW data uncached	Reserved
0x4600_0000 to 0x4800_0000	Reserved	RW data uncached

Chapter 3 Reference: Functions

3.1 `_bootloader_init_table`

Initialize image table

Synopsis

```
int32_t _bootloader_init_table(void)
```

Description

- Initialize image table. Create a blank table if does not exists.
- Calculate check sum all image. Image stored in NAND Flash may be deleted by another application, therefore it should calculate checksum before using.

Return Codes

MQX_OK – No error occurs

IO_ERROR – IO error occurs

3.2 `_bootloader_check_image`

Synopsis

```
bool _bootloader_check_image  
(     bool autoboot,  
      uint32_t index)
```

Description

Checking autoboot image

Return Codes

TRUE - If image index is set to autoboot image

FALSE - If image index not set to autoboot image

3.3 `_bootloader_del_image`

Synopsis

```
uint32_t _bootloader_del_image(uint32_t index)
```

Description

Delete image with index specified

Return Codes

MQX_OK – No error occurs

IO_ERROR – IO error occurs

3.4 `_bootloader_del_table`

Synopsis

```
int32_t _bootloader_del_table(void)
```

Description

Delete all images

Return Codes

MQX_OK – No error occurs

IO_ERROR – IO error occurs

3.5 `_bootloader_store_image_data`

Synopsis

```
int32_t _bootloader_store_image_data
(
    uint32_t    addr,
    uint32_t    size,
    unsigned char * buff,
    unsigned char * name
)
```

Description

Store image in NAND Flash. Bootloader find images was masked delete, overwrite this one if new image size equal or smaller than old image. Otherwise store new image in the end of table.

Addr - [IN] Start address refer

Size - [IN] Image size

Buff - [IN] Pointer to image data

Name - [IN] Name of image will be store in table.

Return Codes

MQX_OK - No error occurs

IO_ERROR - IO error occurs

3.6 `_bootloader_list_image`

Synopsis

```
void _bootloader_list_image(void)
```

Description

Print out all images stored in NAND Flash

Return Codes

NA

3.7 `_bootloader_set_default`

Synopsis

```
int32_t _bootloader_set_default(uint32_t index)
```

Description

Set image as autoboot image after bootloader startup with timeout value.

Return Codes

MQX_OK - No error occurs

IO_ERROR - IO error occurs

3.8 `_bootloader_get_timeout`

Synopsis

```
int32_t _bootloader_get_timeout(void)
```

Description

Get timeout value of autoboot image

Return Codes

-1 – No image found or no timeout value found.

3.9 _bootloader_set_timeout

Synopsis

```
int32_t _bootloader_set_timeout(uint32_t index, uint32_t
timeout)
```

Description

Set image timeout value

Return Codes

<i>MQX_OK</i>	– No error occurs
<i>IO_ERROR</i>	– IO error occurs

Chapter 4 Reference: Commands

4.1 ipconfig

This command used to setup Ethernet network. Flow the command instruction to set you board IP address.

4.2 imgload

This command used to load a file type .bin or .s19 from TFTP server and store to NAND flash immediately.

This image is set as auto boot when transfer success.

Usage

```
imgload <host> <source> <addr> [<name>]
```

<host> - TFTP server ip address

<source> - Source file name

<add> - Start address in Ram

[<name>] - Name of image will be stored in table (optional). Default source file name is stored in table.

Examples

```
imgload 10.207.215.40 hello.s19 0x400048f8 helloworld
```

4.3 imglst

This command has no argument. This command used to list all images in NAND flash.

Usage

This command doesn't have argument

4.4 imgdel

This command used to delete a image in memory of NAND flash.

Usage

```
imgdel <index>
```

<index> - Index of image to delete

4.5 tbdel

This command used to delete all images in memory NAND flash.

Usage

This command doesn't have argument

4.6 autoboot

This command used to select image auto boot when reset or power on.

Usage

```
autoboot [<index>]
```

<index> - Image index to boot.

4.7 imggo

This command jumps to image and runs application of this image.

Usage

```
imggo <index>  
    <index> - Image index in memory table.
```

4.8 imgtimeout

This command sets autoboot timeout expire in second.

Usage

```
atbtmo <index> <timeout>  
    <index> - Image index to set timeout  
    <timeout> - Time out in second value.
```

4.9 imgren

This command used to rename the image in table.

Usage

```
imgren <index> <name>  
    <index> - Image index to be renamed.  
    <name> - New image name.
```

4.10 imgaddr

This command used to set the boot address of image.

Usage

```
imgren <index> <address>  
    <index> - Image index to set boot address.  
    <name> - New image boot address.
```

4.11 nandcheck

This command used to check bad blocks of Nand flash

Usage

```
nandcheck
```

4.12 nandef

This command is used to force erase specific block of Nand flash

Usage

```
nandef <block>  
    <block> - Block force erased.
```

4.13 Reset

This command is used to soft reset the board.

Usage

```
reset
```

Chapter 5 MQX Bootloader Examples

5.1 Introduction

This chapter gives you insight on how to use the Boot Loader example. The example use TWR-MCF54418 board and CodeWarrior version 10.6. The following sections described in this chapter:

- Preparing the setup
- Preparing image file (CodeWarrior 10.6)
- Running the application

5.2 Preparing the setup

5.2.1 Software Setting up

This demo uses the following software:

- Code Warrior version 10.6
- Hyper terminal on Windows XP
- NetBurner TFTP Server Version 1.0

5.2.2 Hardware Setting up

Make connection as shown in this following picture

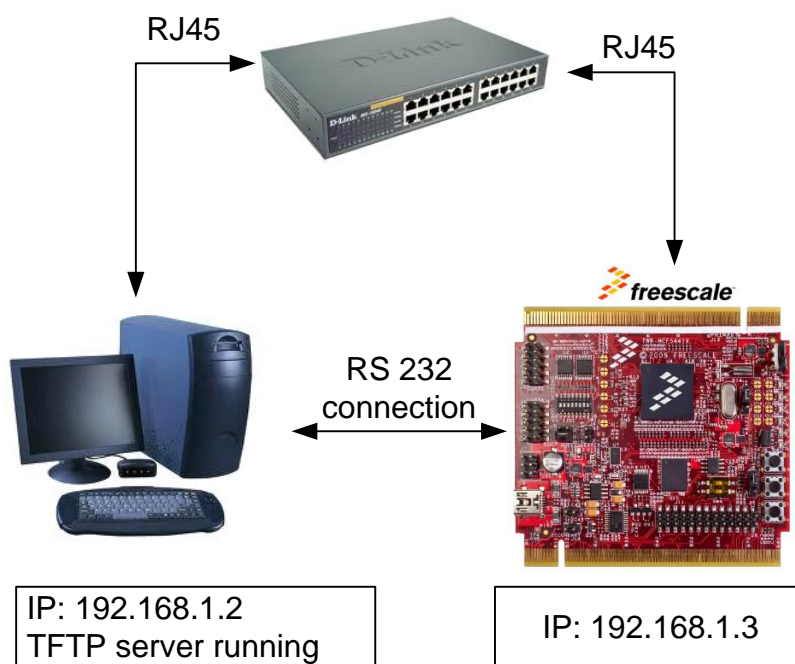


Figure 5-1 Hardware Setting up

This demo uses the following hardware:

- A personal computer
- A TWR-MCF54418 board and power supply for it

Follow these steps to setup the hardware

1. Connect the power supply to the board.
2. Connect to serial port of TWR-SER board.
3. Turn board power on.

5.3 Preparing image file

This section describes steps to create a MQX image, which will be loaded and executed in RAM by Boot loader.

5.3.1 Preparing image file for TWR-MCF54418

Step1. Build libraries of MQX by running

```
[MQX_PATH] \mqx\build\cw10\bsp_twrpcf54418
[MQX_PATH] \mqx\build\cw10\psp_twrpcf54418
[MQX_PATH] \mfs\build\cw10\mfs_twrpcf54418
[MQX_PATH] \rtcs\build\cw10\rtcs_twrpcf54418
[MQX_PATH] \shell\build\cw10\shell_twrpcf54418
```

Step2. Compile an application that will be run by Boot loader. For examples, we chose “Hello World”

```
[MQX_PATH] \mqx\examples\hello\build\cw10\hello_twrpcf54418
```

There should be chose Ram target for compiling because boot loader will copy image for Flash memory to Ram memory and execute program.

Step3. Configured the target settings

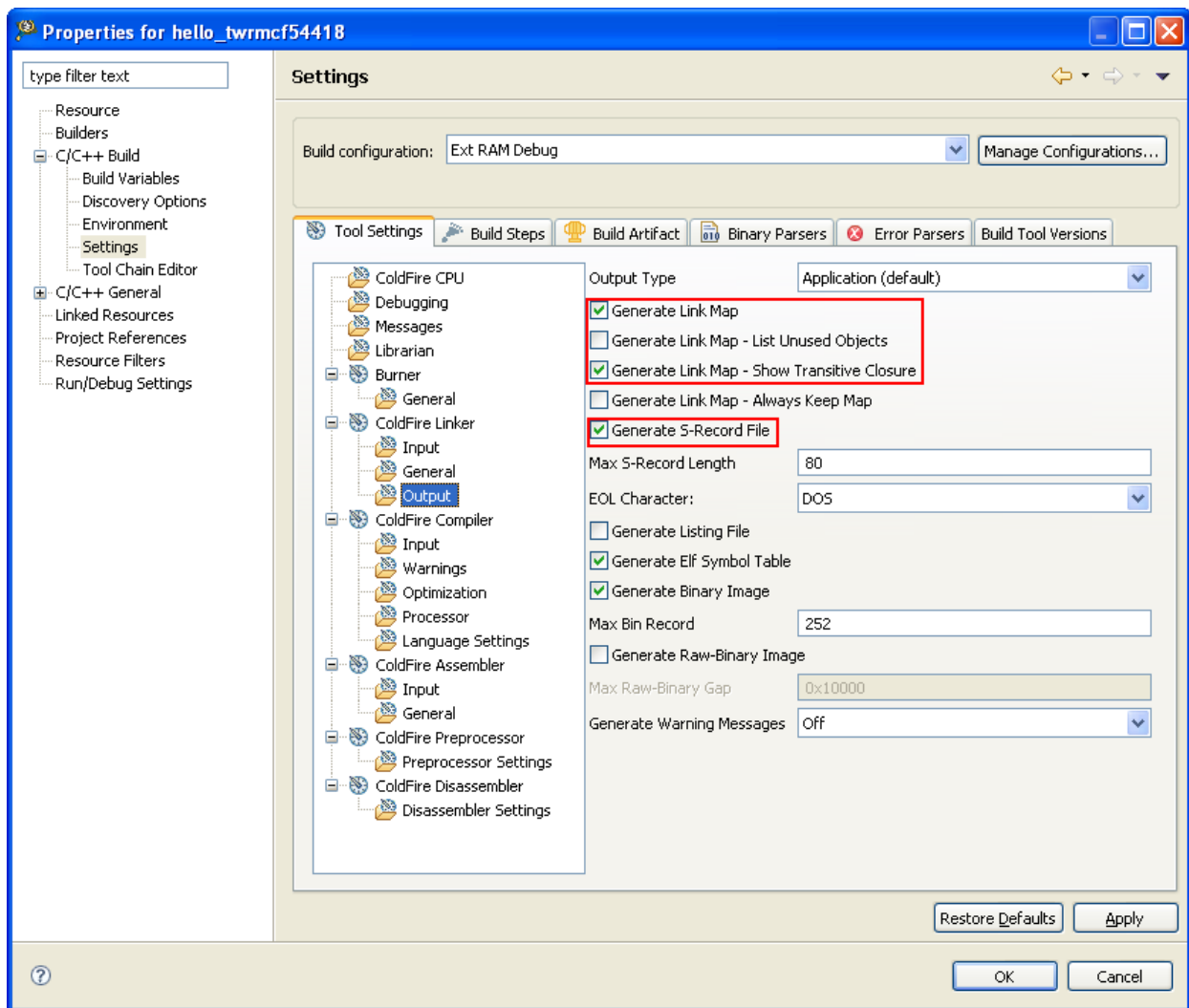


Figure 5-2 Target settings

Step4. Compile project

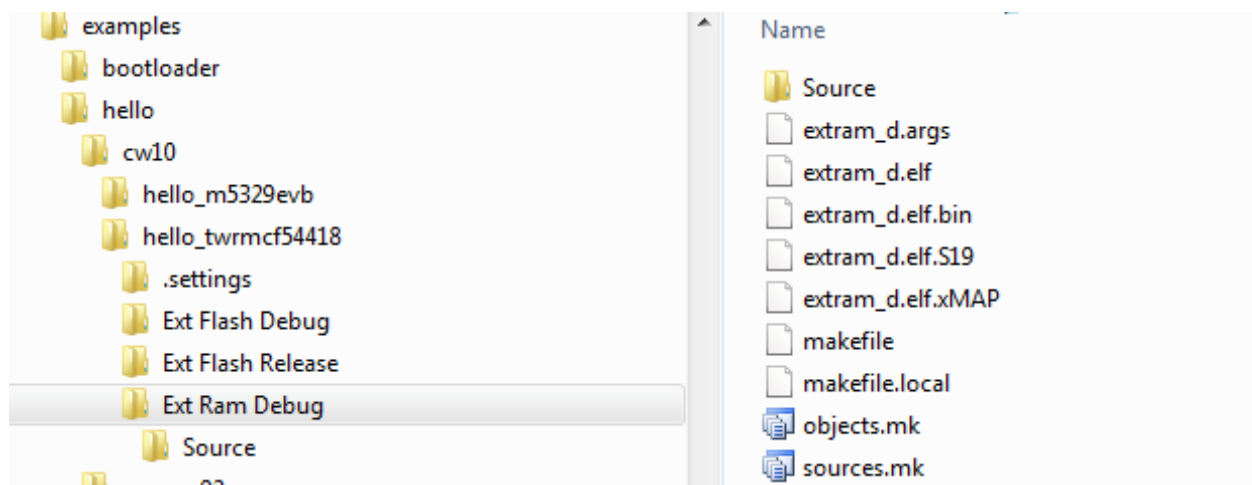


Figure 5-3 Generated files

Step5. Determine start address – Entry point in CodeWarrior setting.

Open `extram_d.elf.xMAP` by a text editor and find **boot** address. The format will be same with following line:

```
4010A60A 00000058 .text __boot(bsp_twrnmc54418_d.a boot_c.o )
```

0x4010A60A is start address, remember that will be a argument of *imgload* command

Step6. Copy `extram_d.elf.s19` to the TFTP root folder. For example

`D:\projects\tftp_root\twrnmc54418`

Rename `extram_d.elf.s19` to `helloworld_extram_d.s19`

5.4 Running Boot loader

This section describes step by step running Boot loader application with assumption your PC had installed Net Burner TFTP server or other kind of TFTP server.

Step1. Setting up “Search Directories” to the folder of image file, refer to section 5.3.1 Step 5.

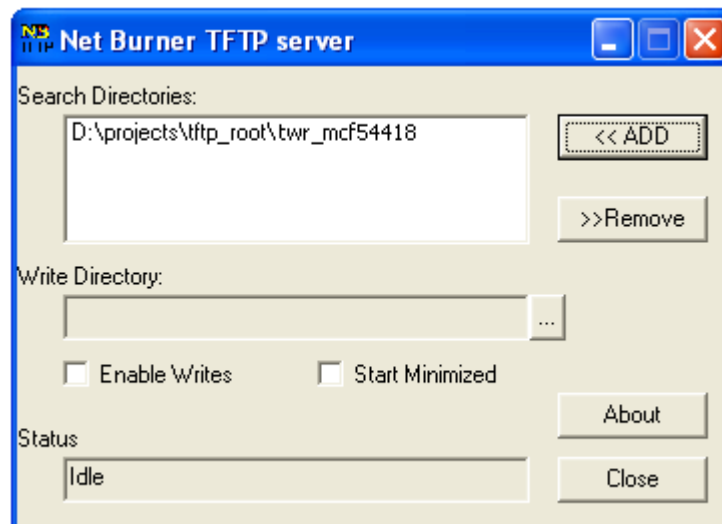


Figure 5-4 TFTP server settings

Step2. Manual configure your IP address as figure bellow:

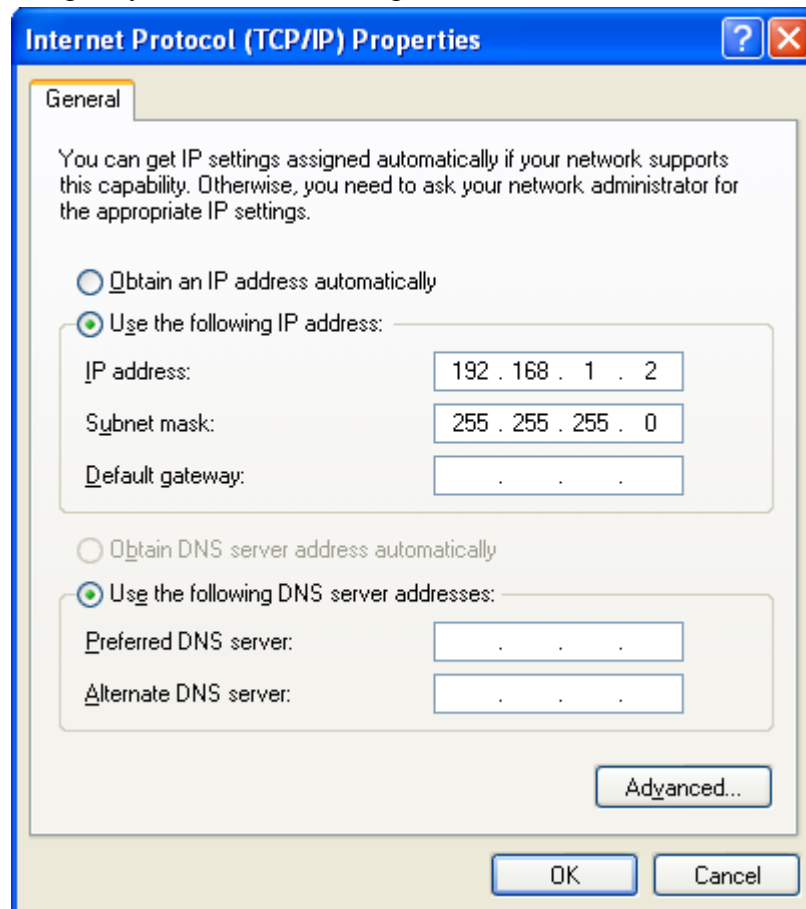


Figure 5-5 IP configure manual settings

Step3. Setting up HyperTerminal

To ensure that applications run correctly, the HyperTerminal is used on your computer to get events from the device that running the application. These steps are used to configure the HyperTerminal program:

1. Open HyperTerminal applications as shown in **Figure 5-6**

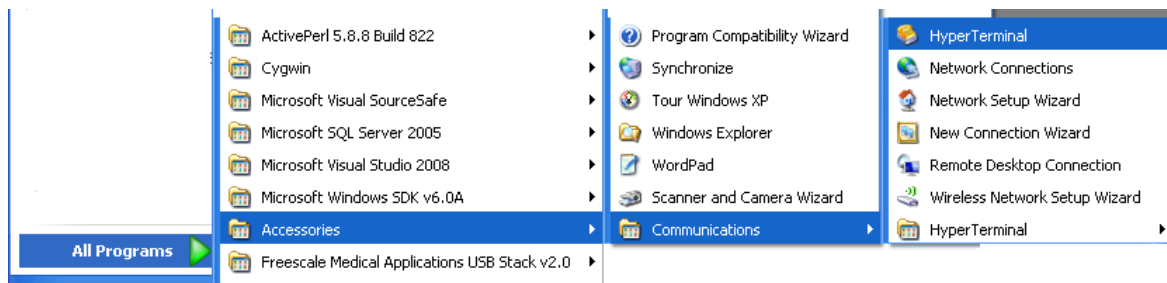


Figure 5-6 Launch HyperTerminal Application

2. The HyperTerminal opens as shown in *Error! Reference source not found.*. Enter the name of connection and click on the **OK** button.

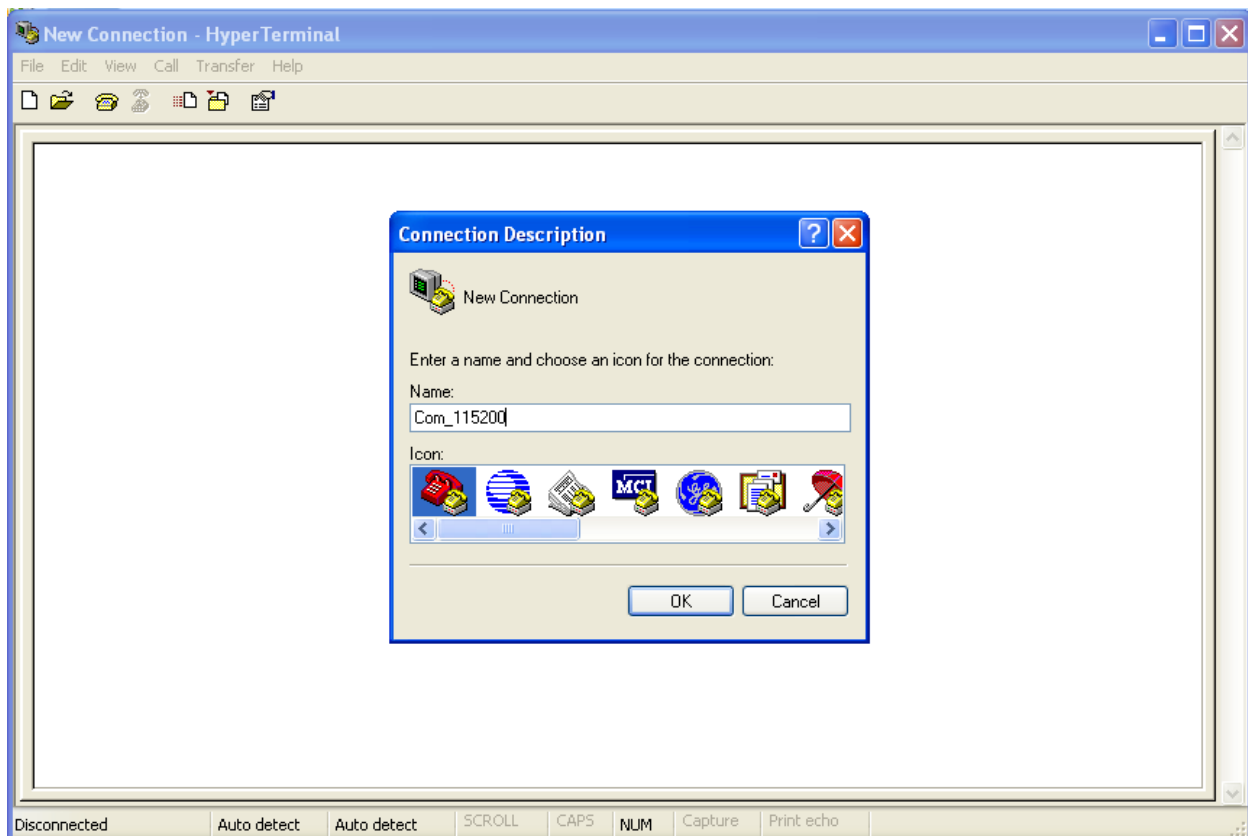


Figure 5-7 HyperTerminal GUI

3. The window shown in the **Figure 5-8** appears. Select the COM port appropriate with the COM port that you connect to the board.



Figure 5-8 Connect using COM8

4. In the next window set the communication baud rate to 115200, data length to 8, no parity, one stop bit and no flow control, then click **OK** to complete the HyperTerminal configuration. Configure the virtual COM port baud rate and other properties as shown in **Figure 5-9**

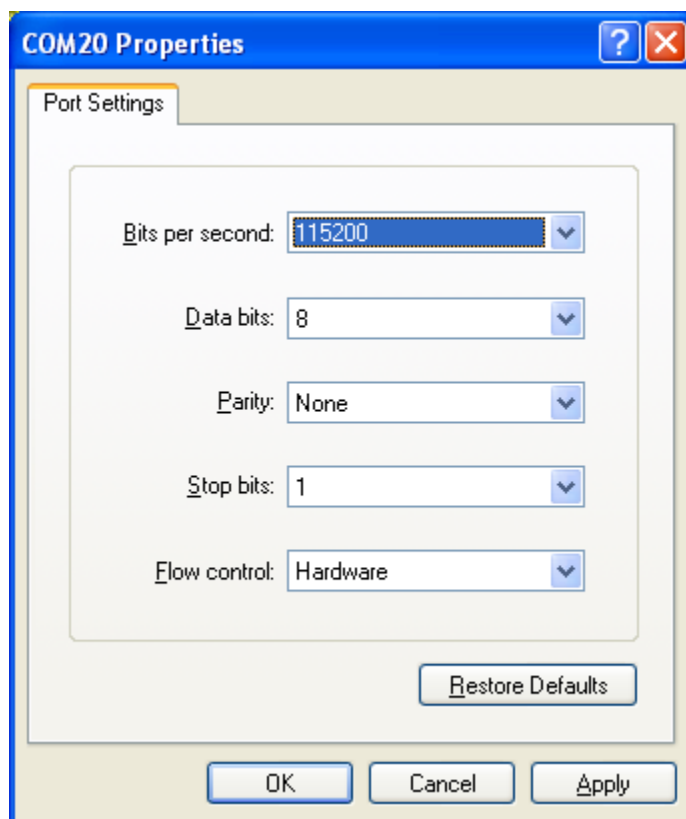


Figure 5-9 COM8 Properties

5. The HyperTerminal is configured now.

Step4. Run Boot loader application.

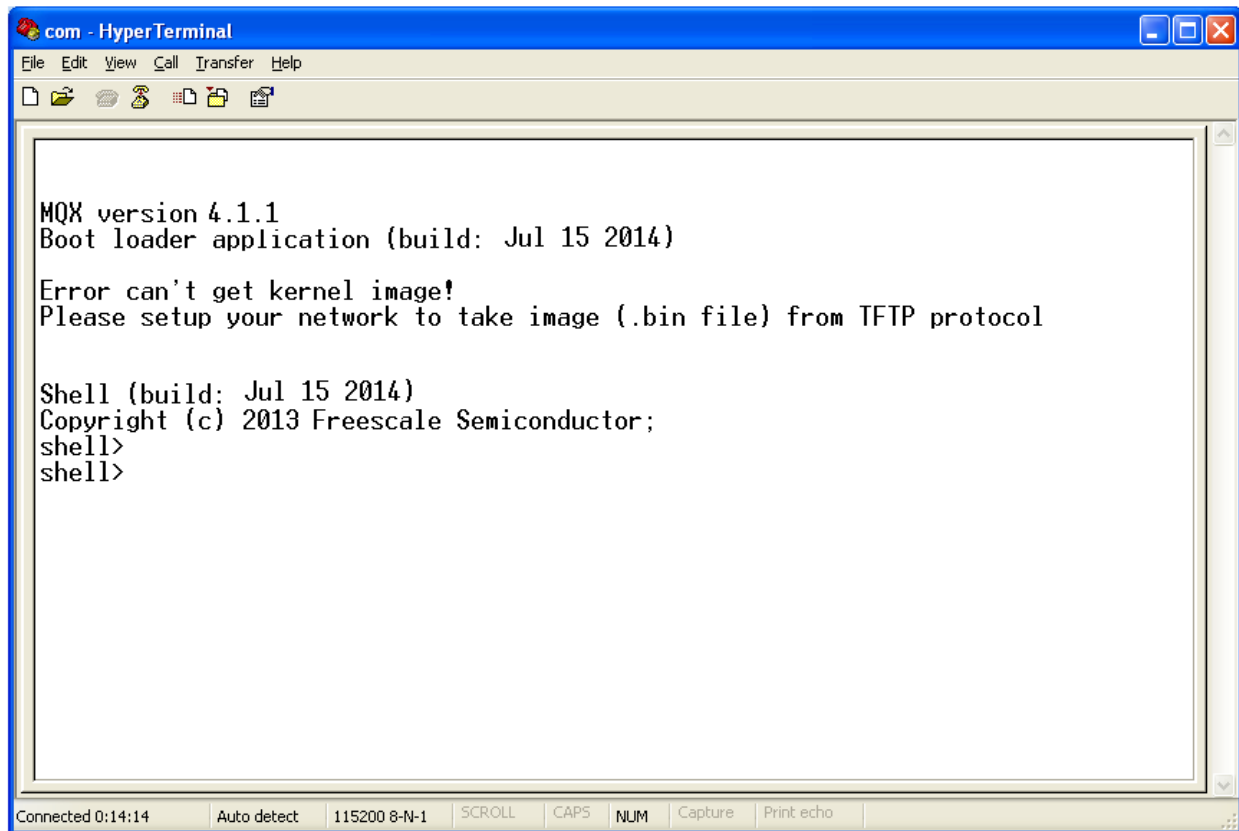


Figure 5-10 Bootloader started with no image

Step5. Configure board net IP address

```
shell> ipconfig 0 init
Ethernet device 0 initialization successful.
shell> ipconfig ip 192.168.1.3 255.255.255.0 192.168.1.1
Static IP4 bind successful.
shell>
```

Step5. Load image to Flash

```
shell> imgload 192.168.1.2 helloworld_extram_d.bin 0x10000 helloworld
```

```

com - HyperTerminal
File Edit View Call Transfer Help

Ethernet device 0 initialization successful.
shell> ipconfig ip 192.168.1.3 255.255.255.0 192.168.1.1
Static IP4 bind successful.
shell> imgload 192.168.1.2 helloworld_extram_d.bin 0x10000 helloworld

Downloading file helloworld_extram_d.bin from TFTP server: [192.168.1.2]
#####
Transfer 0x1f560 byte success
Write image to table Success
Listing images...
Table of stored images:
=====
| Index |          Name          | Start address |   Size   | Flash Addr | Auto |
| Timeout|                         |               |          |             |      |
=====
|   1    |          helloworld    | 0x00010000    | 0x0001f560 | 0x00f00000 |    1  |
|   5    |                         |               |          |             |      |
|        |                         |               |          |             |      |
=====
shell>

```

Connected 0:22:07 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

Figure 5-11 Image is stored successful in flash

Step6. Run image

```
shell> imggo 1
```

If the Boot loader application is flash to Nand Flash memory by using Ext Nand Download targets, you can boot image from hard reset.