1 Introduction

This document describes how to use an Ethernet bootloader

provided by Freescale. This Ethernet bootloader is supported

in different platforms, such as Kinetis, MCF52259, and

MCF51CN128. This application note is focused on the Kinetis

family, which has an ARM Cortex-M4 core. The software and

main tests were performed on a K60N512. It explains how to

modify the .lcf and .icf link files of CW and IAR IDEs

respectively, in order to add the Ethernet bootloader support

for your application.

2 Overview

A bootloader is a small piece of software that allows the user

to download/update either code or data into flash memory.

The communication layer between the MCU and the host,

which contains the new application binary, might differ

depending on the use case. This communication interface

could be SPI I2C, Ethernet, and USB, etc. For this scenario,

use an Ethernet communication interface. The bootloader

provided will work as a TFTP client and the DHCP feature

will be used

This ethernet bootloader implements a TFTP server on the Hercules microcontroller to upload the firmware to board over ethernet controller attached using any TFTP client.

2 Supported Features

• Integration of the CPDMA based EMAC driver for RM46/RM48/RM57/TMS570LS/TMS570LC devices

• lwIP (lightweight IP) 1.4.1 TCP/IP stack ported for the above devices' EMAC driver

• By default, the DHCP support is enabled. However, the integration supports both DHCP and Static IP addressing.

• Integration with HALCoGen v04.07.01 release

• The integrated application demonstrates web server applications on RM46/RM48/RM57/ TMS570LS/TMS570LC devices.

• By default, the software is configured for executing from flash.

• Diagnostic and debug messages are printed on the JTAG SCI Port. The following are the settings for the console:

– Baud Rate: 9600

– Data: 8 bit

– Parity: None

– Stop bit: 2

– Flow Control: None

NOTE: Since the MAC Address is part of the binary image, all of the devices programmed with these binaries and connected to the same DHCP server will be assigned the same IP address. The default MAC address is 00:08:EE:03:A6:6C.

3 Get the Software

The latest lwIP Demo software version 04.00.00 can be downloaded from: <http://git.ti.com/hercules_examples/hercules_examples/trees/master/Application/LwIP>.

The previous version 00.03.00 can be downloaded from: http://softwaredl.ti.com/hercules/hercules\_public\_sw/HALCoGen\_EMAC\_lwIP-00.03.00-installer.exe. Compared to 00.03.00 version, the new version has the following updates:

• Code generated with HALCoGen release 04.07

• Added LAUNCHXL2 570LC43x and LAUNCHXL2 RM57x support The latest active web server demo software version 1.1.0 can be downloaded from: http://git.ti.com/hercules\_examples/hercules\_examples/trees/master/Application/ActiveWebserver. Compared to the previous 1.0.0 version, the new one updated with HALCoGen release 04.07.01 supports LAUNCHXL2 570LC43x and LAUNCHXL2 RM57x.

4. Configuring EMAC and MDIO Using HALCoGen GUI for the lwIP Demo

NOTE: The lwIP demo software package includes all the CCS projects and source code for various Hercules platforms. See the installation for more details. Those projects can be imported into CCS and built to run as it is.

The following sequence explains how to get the working driver code for the EMAC and MDIO modules using HALCoGen for the lwIP demo, which is a static web server demo. This is for informative purposes only.

4.1 RM46x, RM48x and TMS570LSx HDK

1. Under the ‘Driver Enable’ tab, enable EMAC Driver and SCI2 Driver.

2. Under ‘VIM RAM’, add the names of the ISRs for EMAC Transmit and Receive Interrupts (Channels 77 and 79, respectively).

3. Enable these interrupts under the ‘VIM Channel 64-95’ tab.

4. Under the ‘PLL’ tab, change the multiplier for both PLLs to a value of 120, such that the output frequency in both cases is 160.00 MHz

5. Under the ‘GCM’ tab, change the value of the VCLKA4 divider to 2, such that the output of VCLKA4( or VCLKA4\_DIVR\_EMAC in case of RM46x/TMS570LS12x devices) is 40.00 MHz

6. Under the ‘PINMUX’ tab, enable RMII or MII (depending on the board that you are using - Most HDKs support MII by default, while control CARDs are designed for RMII), MDIO(G3) and MDCLK(V5).

7. Under the ‘EMAC’ tab, change the EMAC address to 00:08:EE:03:A6:6C, which is the default MAC address. Change the physical address to 1.

8. Generate the system initialization and HAL Code.

4.2 TMS570LC43x and RM57x HDK

1. Under the ‘Driver Enable’ tab, enable EMAC Driver and SCI1 Driver.

2. Under ‘VIM RAM’ add the names of the ISRs for EMAC Transmit and Receive Interrupts (Channels 77 and 79, respectively).

3. Enable these interrupts under the ‘VIM Channel 64-95’ tab.

4. Under the ‘PLL’ tab, change the multiplier for both PLLs to a value of 150, such that the output frequency in both cases is 300.00 MHz.

5. Under the ‘GCM’ tab, change the value of the VCLK1, VCLK2 and VCLK3 Dividers to 1 and VCLKA4 Divider to 2, such that the output of VCLKA4\_DIV is 37.50 MHz.

6. Under the ‘PINMUX’ tab, enable RMII/MII, under Pin Muxing. Under Input Muxing, enable MDIO(G3), MII\_COL(F3), MII\_CRS(B4), MII\_RX\_DV(B11), MII\_RX\_ER(N19), MII\_RXCLK(K19), MII\_RXD[0], MII\_RXD[1], MII\_RXD[2], MII\_RXD[3], MII\_TX\_CLK.

7. Under the ‘EMAC’ tab, change the EMAC address to the correct address (the default one in the example is mentioned above). The physical address is 1 by default.

8. Generate the system initialization and HAL Code.

4.3 RM57x Launchpad (LAUNCHXL2 RM57x)

1. Navigate to the folder ''. Copy the RM57x folder to a folder named '/LAUNCHXL2-RM57'.

2. Start a HALCoGen session.

3. From HALCoGen, open the project '/LAUNCHXL2-RM57/HALCoGen-RM57x/HALCoGenRM57x.hcg'.

4. Make the following changes to the HALCoGen Project:

a. RM57L843ZWT/Driver Enable Tab. Enable the GIO Driver:

i. Check "Enable GIO Driver".

ii. Confirm GIO, SCI1, and EMAC drivers are enabled.

b. RM57L843ZWT/ECLK Tab. Provide a 25 MHz Clock to the PHY:

i. In the ECLK Pin Mode Group, change the ECLK pin Mode to ECLK.

ii. In the ECLK pin Group, make sure DIR is checked.

iii. In the ECLK Functional Configuration group, change Divider to 3 so that ECPCLK is 25 MHz.

iv. Also check the "Continue on suspend" button.

c. PINMUX/Pin Muxing Tab. Change the MII and MDIO interface pins to their default locations:

i. Make sure MII is checked in the Enable/Disable Peripherals Group at the top of the PINMUX. This is necessary to put the MAC in MII mode, even though it will also move the MII pins to their alternate location, which have to be undone manually.

ii. Uncheck the MII and MDIO signals on Balls A14, B4, B11, D19, E18, F3, G3, G19, H18, H19, J18, J19, K19, N19, P1, R2 and V5. These rows should now be blank, although you can select non-Ethernet functions, if desired.

iii. Change the selection on balls T4, U7 to the default functions (from MII\_RX\_AVCLK4 to MII\_RXCLK, and from MII\_TX\_AVCLK4 to MII\_TX\_CLK). The PHY will provide these clocks to the MAC on the launchpad.

NOTE: List Conflicts show total conflicts 2 for Ball T4 and U7 - just ignore this.

d. PINMUX/Input Pin Muxing Tab. Change all of the input MII and MDIO signals to the Default (left Column) states: i. MDIO=F4, MII\_COL=W4, MII\_CRS=V4, MII\_RX\_DV=U6, MII\_RX\_ER=U5, MII\_RXCLK=T4, MII\_RXD[0]=U4, MII\_RXD[1]=T3, MII\_RXD[2]=U3, MII\_RXD[3]=V3, MII\_TX\_CLK=U7

e. GIO/Port A Tab. GIOA[3] and GIOA[4] need to be driven high, to release the PHY from reset and power down. i. Check the "DIR" box for Bit 3 and Bit 4. ii. Change DOUT to '1' for Bit 3 and Bit 4.

f. Save your modified HALCoGen Project. g. Press "F5" or Choose File → Generate Code from The HALCoGen menu. h. Exit HALCoGen.

5. Launch Code Composer Studio.

6. Import the project you just modified:

a. From the Project Menu, choose "Import CCS Projects".

b. Select 'search-directory'.

c. Browse to the folder ''.

d. From the list of projects available to import - select "Build-RM57x".

7. Make a copy of the original project, naming it "Build-LAUNCHXL2-RM57x":

a. In CCS's Project Explorer, select the project "Build-RM57x".

b. Use CTRL-C, CTRL-V or the context menu to copy/paste the project.

c. For the new project, use the name "Build-LAUNCHXL2-RM57x". You can create a new folder '/LAUNCHXL2-RM57/Build-LAUNCHXL2-RM57x' for this.

8. Change the HALCoGen Project included by your new project:

a. Select your new project, 'Build-LAUNCHXL2-RM57x', in CCS's Project Explorer.

b. If not expanded, expand the tree under this project.

c. Select the folder "HALCoGen-RM57x". The folder icon should indicate that it is a linked resource.

d. From the context-menu, select "Properties" for the folder "HALCoGen-RM57x".

e. With "Resource" selected, press "Edit" and change the location to: PROJECT\_LOC\..\HALCoGenRM57x. f. Make sure the Resolved Location displayed matches the HALCoGen project that you edited in step 7.

g. You can do the same steps to resolve the locations for the "example" and "lwIP-1.4.1" folders if your CCS project was created in a different location.

9. Change the PHY Id to match the DP83630 Precision PHYTER on the Launchpad:

a. From your "Build-LAUNCHXL2-RM57L" CCS project, navigate to and open for editing "HALCoGenRM57x\include\HL\_phy\_dp83640.h".

b. Change the last "USER CODE" block in the header file, so that it reads:

/\* USER CODE BEGIN (2) \*/

/\* @todo @fixme: This is a dirty hack, but it minimizes changes for now \*/

#undef DP83640\_PHY\_ID

#define DP83640\_PHY\_ID (0x20005CE1u)

/\* USER CODE END \*/

10. Add Code to initialize GIOA[3] and GIOA[4] (To release the PHY):

a. From your "Build-LAUNCHXL2-RM57L" CCS project, navigate to and open for editing "HALCoGenRM57x\source\HL\_sys\_main.c".

b. Add an include directive for "HL\_gio.h", so that the first USER CODE block reads:

/\* USER CODE BEGIN (1) \*/

#include "HL\_gio.h"

extern void EMAC\_lwIP\_Main (uint8\_t \* emacAddress);

/\* USER CODE END \*/

c. Add a call to gioInit() from main() so that the third user block reads:

/\* USER CODE BEGIN (3) \*/

gioInit();

EMAC\_lwIP\_Main(emacAddress);

/\* USER CODE END \*/

11. Update PINMUX file:

a. From your "Build-LAUNCHXL2-RM57L" CCS project, navigate to and open for editing "HALCoGenRM57x\source\HL\_pinmux.c".

b. Change PINMUX\_BALL\_R4\_ to PINMUX\_BALL\_R4\_GIOB\_3 pinMuxReg->PINMUX[9] = PINMUX\_BALL\_R4\_GIOB\_3 | PINMUX\_BALL\_N17\_EMIF\_nCS\_0 | PINMUX\_BALL\_L17\_EMIF\_nCS\_2;

12. Rebuild your project.

13. Load your project onto the Launchpad and test it out.

4.4 TMS570LC43 Launchpad (LAUNCHXL2 570LC43x)

The instructions discussed in the previous RM57x Launchpad section apply to port lwIP demo on TMS570LC43, except copying '/ TMS570LC43x' as the starting point.

**6.1.6 TFTP server demo**

The TFTP server is a file transfer application that needs a remote TFTP client. The file is

transferred to the Hercules microcontroller located on the Hercules HDK or Launchpad.

The TFTP server waits for a request from a remote TFTP client. The HDK or Launchpad

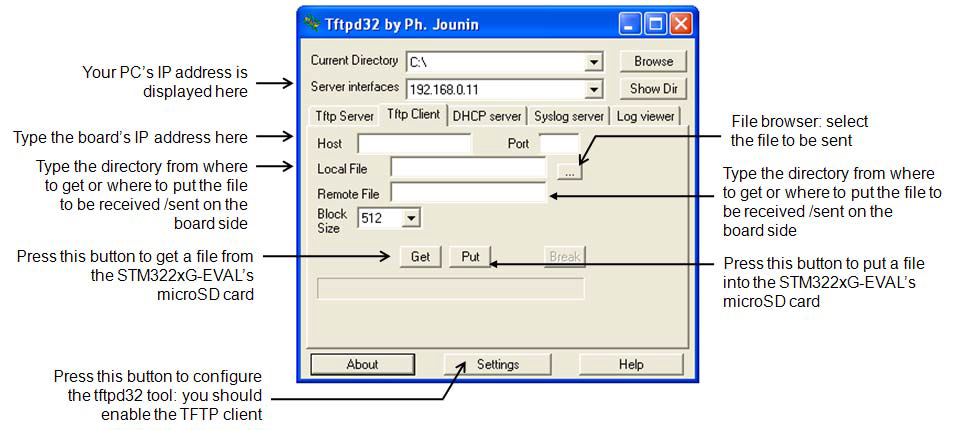
must be connected to a remote PC to upload a file. To do this, a TFTP client must be installed on the remote PC. This can be done by using the tftpd32 tool, which can be found at http://tftpd32.jounin.net

In order to test the tftpserver demo, follow these steps:

1. Be sure of the correct dip switch (S2) settings in the HDK board.

6. On the remote PC, open the TFTP client (for example, TFTPD32), and configure the TFTP server address (host address in TFTPD32).

7. Start transferring files to the Hercules microcontroller located on the HDK board.



The Ethernet bootloader using TFTP is widely used in embedded applications that require a firmware upgrade capability.

TFTP is a simple file transfer protocol that works on top of the UDP transport layer. It is intended to be used on a LAN environment. It is based on a client/server architecture, where a client requests a file transfer (read or write operation) from a file server.

In this case the server only processes write requests from a PC TFTP client, so a simple TFTP server is implemented on top of the LwIP stack.

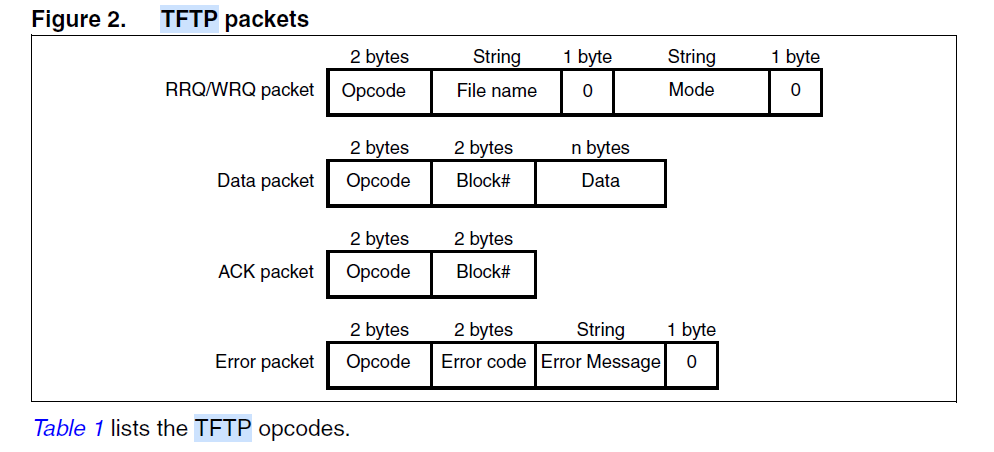
**2 IAP using TFTP**

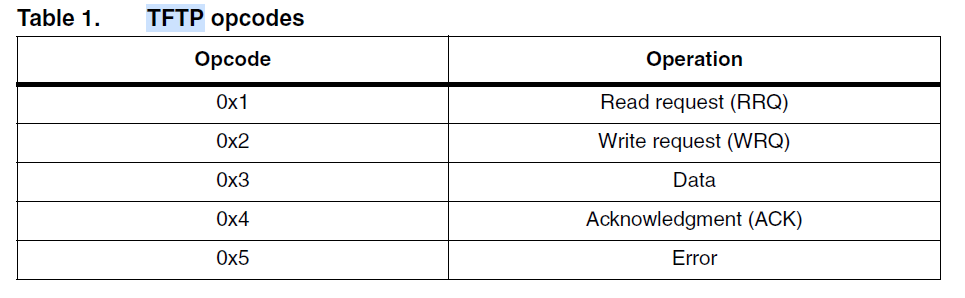
**2.1 TFTP overview**

TFTP is a simple file transfer protocol that works on top of UDP. A file transfer is initiated from a TFTP client, which sends a Read or Write request to a TFTP server. When the server acknowledges the request, the file data transfer starts. The data is sent in fixed size blocks (for example in blocks of 512 bytes).

Each transferred data block must be acknowledged by the recipient before the next block can be sent. The acknowledge mechanism is based on the block number sent with each data block. A data block with less than the fixed block size indicates the termination of the file transfer.

*Figure 2* describes the format of the various TFTP packets:





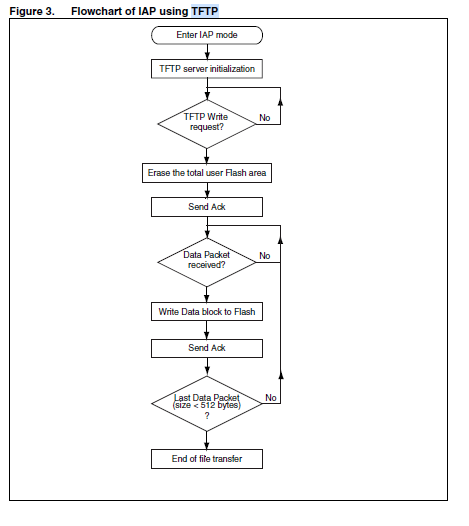
**2.2 Implementing IAP for the STM32F2x7 using TFTP**

This IAP implementation consists of a TFTP server on top of the LwIP TCP/IP stack. This server responds to file WRITE requests received from a remote TFTP client (PC). TFTP READ requests are ignored.

Instead of writing received files to a file system, which is normally what TFTP is used for, the server writes the received data blocks into the MCU Flash (in the user Flash area).

*Note: In this implementation, the data block size is fixed to 512 bytes.*

*Figure 3* provides an overview of the IAP operation using TFTP.



**4 Environment**

**4.1 MAC and IP address settings**

The MAC and IP address are defined in the *main.c* file.

The default MAC address is fixed to: 00:00:00:00:00:02

The IP address can be set either as a static address or as a dynamic address, assigned by

a DHCP server. The default static IP address is: 192.168.0.10

You can select DHCP mode by enabling USE\_DHCP in the *main.h* file.

Note that if you choose to configure the IP address by DHCP and the application does not

find a DHCP server on the network to which it is already connected, the IP address is then

automatically set to the static address (192.168.0.10).

**4.2 Jumper settings on the HDK board**

In order to run the software, configure the HDK board as shown in *Table 2*.

Select between MII and RMII configuration in the main.h file in the project\inc folder.

For example, to select the RMII mode:

//#define MII\_MODE

#define RMII\_MODE

For MII mode, the PHY clock can be taken from the external crystal or from the STM32 via

the MCO pin if both MII\_MODE and PHY\_CLOCK\_MCO are defined in the main.h file.

*Note: 1 In RMII mode, it is not possible to use MCO to output the 50 MHz clock to PHY due to the*

*PLL limitation explained in chapter 2.6.5 of STM32F20x & STM32F21x Errata sheet*

*(ES0005). In such a case, it is possible to provide the 50 MHz clock by soldering a 50 MHz*

*oscillator (ref SM7745HEV-50.0M or equivalent) on the U3 footprint located under CN3 and*

*also by removing the jumper on JP5. This oscillator is not provided with the board. For more*

*details, please refer to STM3220G-EVAL evaluation board User manual UM1057.*

*2 Throughout this document, “STM322xG-EVAL board” refers to STM3220G-EVAL and*

*STM3221G-EVAL boards.*

