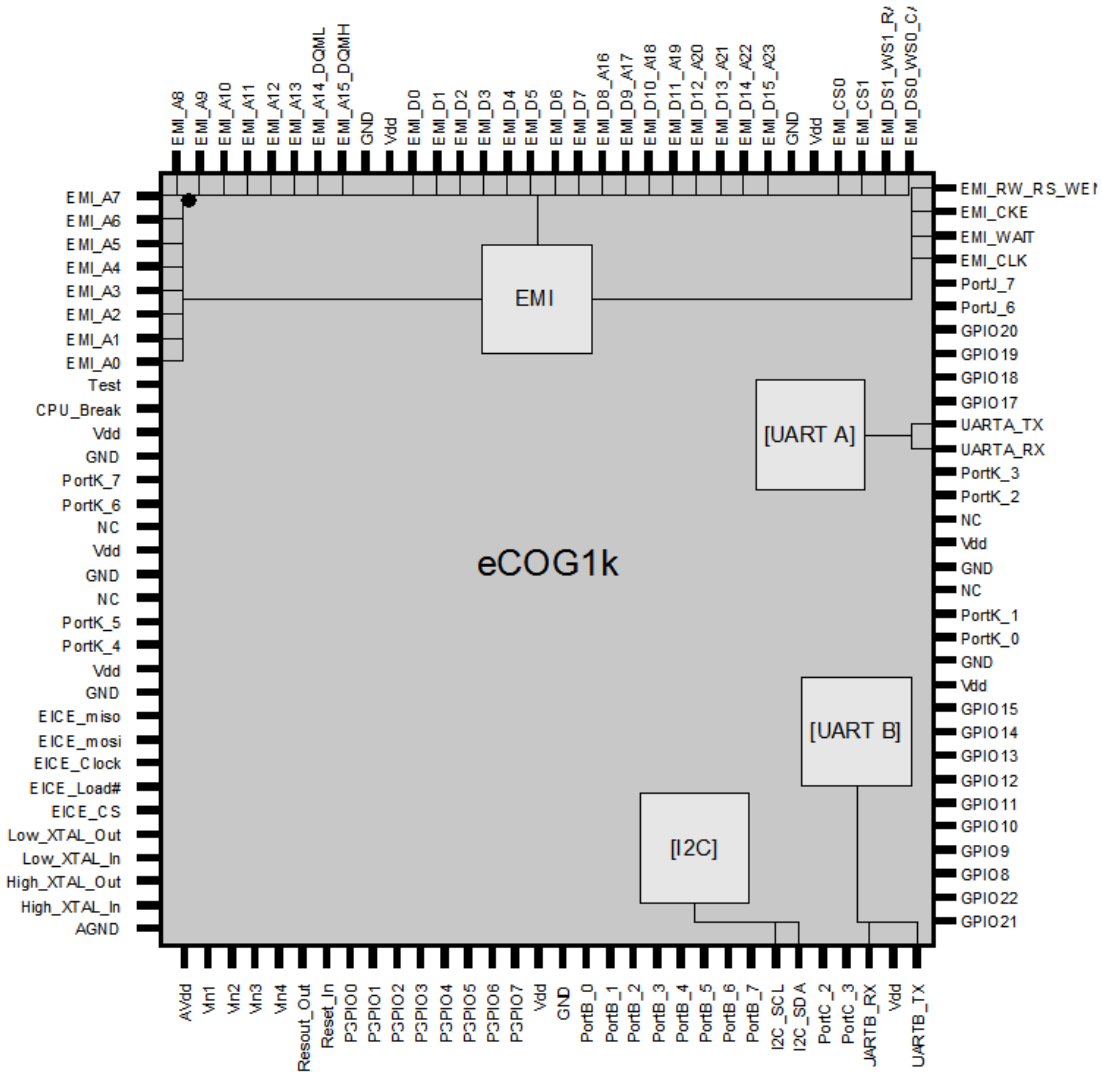




AN048 – Demonstrating TFTP with uIP V0.9

Version 1.0

This application note describes a simple implementation of a TFTP client for the uIP V0.9 TCP/IP networking stack, running on the eCOG1k microcontroller.



Confidential and Proprietary Information

©Cyan Technology Ltd, 2008

This document contains confidential and proprietary information of Cyan Technology Ltd and is protected by copyright laws. Its receipt or possession does not convey any rights to reproduce, manufacture, use or sell anything based on information contained within this document.

Cyan Technology™, the Cyan Technology logo and Max-eICE™ are trademarks of Cyan Holdings Ltd. CyanIDE® and eCOG® are registered trademarks of Cyan Holdings Ltd. Cyan Technology Ltd recognises other brand and product names as trademarks or registered trademarks of their respective holders.

Any product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Cyan Technology Ltd in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Cyan Technology Ltd shall not be liable for any loss or damage arising from the use of any information in this guide, any error or omission in such information, or any incorrect use of the product.

This product is not designed or intended to be used for on-line control of aircraft, aircraft navigation or communications systems or in air traffic control applications or in the design, construction, operation or maintenance of any nuclear facility, or for any medical use related to either life support equipment or any other life-critical application. Cyan Technology Ltd specifically disclaims any express or implied warranty of fitness for any or all of such uses. Ask your sales representative for details.



Revision History

Version	Date	Notes
V1.0	30/11/2006	First release

Contents

1	Introduction.....	5
2	Glossary	5
3	The TFTP Protocol	6
3.1	Overview.....	6
3.2	TFTP Packets.....	6
4	Modifications to the uIP TCP/IP Stack.....	7
5	Using the TFTP Client	8
	Appendix A File Summary.....	10

1 Introduction

TFTP (Trivial File Transfer Protocol) is a simple protocol for file transfer that has minimal capability and minimal overhead. TFTP is used only to read and write files from a remote server. It cannot list directories and has no provision for user authentication. The restricted feature set of TFTP as opposed to FTP makes it easier to implement and results in greatly reduced code size.

This application note describes a simple implementation of a TFTP client for the uIP V0.9 TCP/IP networking stack. Source code for this example is available for the eCOG1k Development Board V2.1 in the file AN048SW.zip.

2 Glossary

A table of abbreviations used in this document.

eCOG1k	Cyan Technology target micro controller
ICMP	Internet Control Message Protocol
RFC	Request For Comments
TFTP	Trivial File Transfer Protocol
TID	Transfer Identifier
UDP	User Datagram Protocol

3 The TFTP Protocol

3.1 Overview

TFTP runs on top of UDP and uses a system of timeouts and retransmissions to guarantee data delivery. TFTP supports five packet types, shown in Figure 1. RFC 1350 describes TFTP thus :-

A transfer begins with a request to read or write a file, which also serves to request a connection. If the server grants the request, the connection is opened and the file is sent in fixed length blocks of 512 bytes. Each data packet contains one block of data, and must be acknowledged by an acknowledgment packet before the next packet can be sent. A data packet of less than 512 bytes signals termination of a transfer. If a packet gets lost in the network, the intended recipient will timeout and may retransmit his last packet (which may be data or an acknowledgment), thus causing the sender of the lost packet to retransmit that lost packet. The sender has to keep just one packet on hand for retransmission, since the lock step acknowledgment guarantees that all older packets have been received. Notice that both machines involved in a transfer are considered senders and receivers. One sends data and receives acknowledgements, the other sends acknowledgements and receives data.

Most errors cause termination of the connection. An error is signalled by sending an error packet. This packet is not acknowledged, and not retransmitted (i.e., a TFTP server or user may terminate after sending an error message), so the other end of the connection may not get it. Therefore timeouts are used to detect such a termination when the error packet has been lost. Errors are caused by three types of events: not being able to satisfy the request (e.g., file not found, access violation, or no such user), receiving a packet which cannot be explained by a delay or duplication in the network (e.g., an incorrectly formed packet), and losing access to a necessary resource (e.g., disk full or access denied during a transfer).

TFTP recognizes only one error condition that does not cause termination, the source port of a received packet being incorrect. In this case, an error packet is sent to the originating host.

3.2 TFTP Packets

TFTP supports five packet types, where the packet type is defined by an opcode field:-

2 byte opcode	N bytes	1 byte	N bytes	1 byte
READ REQUEST (1)	FILENAME	0	MODE	0
WRITE REQUEST (2)	FILENAME	0	MODE	0
DATA (3)	BLOCK	Up to 512 bytes DATA BYTES		
ACKNOWLEDGE (4)	BLOCK			
ERROR (5)	ERROR CODE	ERROR MESSAGE	1 byte 0	

Figure 1. TFTP Packet Types

4 Modifications to the uIP TCP/IP Stack

When TFTP creates a connection, each end randomly creates a Transfer Identifier (TID) which is used to overwrite the UDP port number. For example :-

1. Host A sends a write request to host B, with the source port = TID A and the destination port = 69 (default).
2. Host B sends an acknowledge (with block number = 0) to host A with source port = TID B and the destination port = TID A.

The connection has now been established and the first data packet can be sent by host A with a block number = 1. For all subsequent packets, the hosts check that the source TID matches the previously agreed values.

The above behaviour presents a problem for the uIP v0.9 stack; the TIDs are not recognized as active UDP connections. To rectify this, line 801 of file *uIP.c* is modified as follows:-

Change from:

```
&& ((uip_udp_conn->rport == 0)
    || (UDPBUF->srcport == uip_udp_conn->rport))
```

to:

```
&& ((uip_udp_conn->rport == 0)
    || (UDPBUF->srcport == uip_udp_conn->rport)
    || (uip_udp_conn->rport == HTONS(TFTP_PORT)))
```

5 Using the TFTP Client

This section illustrates the use of the TFTP client with a telnet connection.

Note: As this is purely a demonstration, there is no file system and no file content is stored on the eCOG1k development board.

3. Install a TFTP server application on the host PC. A free TFTP server is available from Solarwinds. See <http://solarwinds.net>.
4. If the host PC has a firewall enabled, check that the telnet service is not blocked.
5. Connect the USB programming cable to the eCOG1k development board.
6. Connect the LAN port of host PC and the LAN port of the eCOG1k development board via a router. (The two may be connected directly only if a crossover cable is available.)
7. Set the telnet address by modifying UIP_IPADDR0-3 in file *uipopt.h*.
8. Set the default router address by modifying UIP_DRIPADDR0-3 in file *uipopt.h*.
9. In CyanIDE, rebuild the TFTP client application, then download and execute it using *Debug -> Run* from the main menu.
10. On the host PC, open a windows command shell and telnet to the eCOG1k development board. The following information is now displayed:-

```
uIP command shell
Type '?' for help
uIP-0.9 (Cyan) >
```

11. Enter '?' at the prompt and the list of available commands is displayed:-

```
uIP-0.9 (Cyan) > ?
Available commands:
stats - show uIP statistics
exit - exit shell
tftp - connect tftp server
file - set file size for put
put filename - tftp put
get filename - tftp get
? - show this help
uIP-0.9 (Cyan) >
```

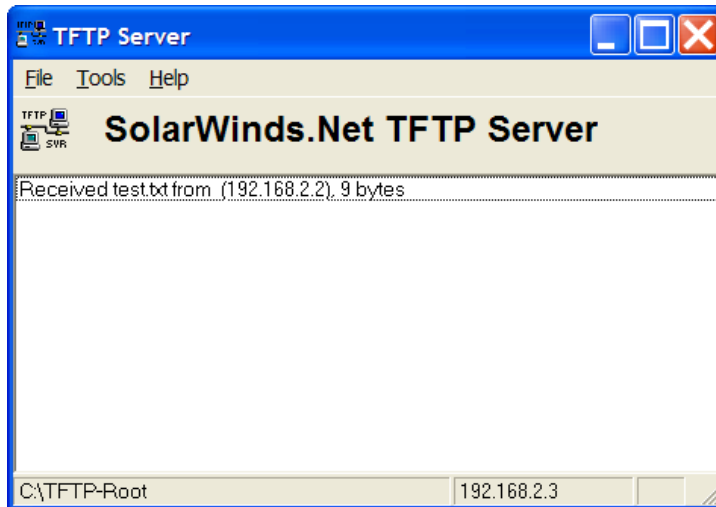
12. Connect to the TFTP server by entering **tftp** followed by the IP address of the TFTP server:-

```
uIP-0.9 (Cyan) > tftp 192.168.2.3
connect tftp server,
ip 192.168.2.3
uIP-0.9 (Cyan) >
```

13. To upload a file to the TFTP server, first set the file size and then transfer the file using **put**:-

```
uIP command shell
Type '?' for help
uIP-0.9 (Cyan) > tftp 192.168.2.3
connect tftp server,
ip 192.168.2.3
uIP-0.9 (Cyan) > f 9
set file size 9
uIP-0.9 (Cyan) > p test.txt
uIP-0.9 (Cyan) >
```

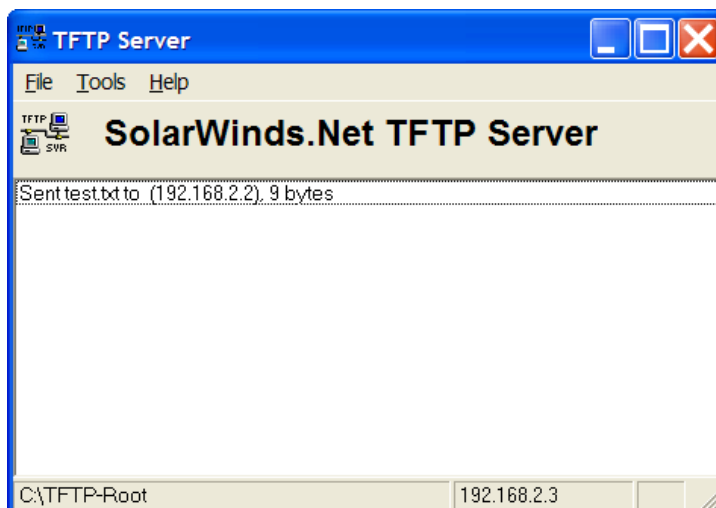
Note that the TFTP server shows that a file has been received:-



14. To retrieve a file from the TFTP server, enter 'g' followed by the filename:-

```
uIP-0.9 (Cyan) > g test.txt
uIP-0.9 (Cyan) >
```

Note that the TFTP server shows that a file has been sent:-



15. Finally, type **exit** to end the TFTP session:-

```
uIP-0.9 (Cyan) > exit
TFTP OK (9)
Connection to host lost.
C:\>
```

Appendix A File Summary

File	Description
irqutil.c	Interrupt mode library utility.
timeofday.c, timeofday.h	Basic time functions.
uip_arch.c	Architecture dependent uIP functions.
uipopt.h	uIP configuration options.
smc91c111.c, smc91c111.h, smc91c111_defs.h	Ethernet device driver.
uip.c, uip.h	uIP TCP/IP stack code.
uip_arp.c, uip_arp.h	uIP ARP implementation.
uip_arch.h	Architecture dependent uIP functions.
main.c	Main entry function.
putchar.c	Basic serial port output routine.
memb.c, mem.h	Memory block allocation.
telnetd.c, telnetd.h, apphdr.h	Telnet server.
telnetd_shell.c	Telnet server shell.
tftp.c, tftp.h	The TFTP application.