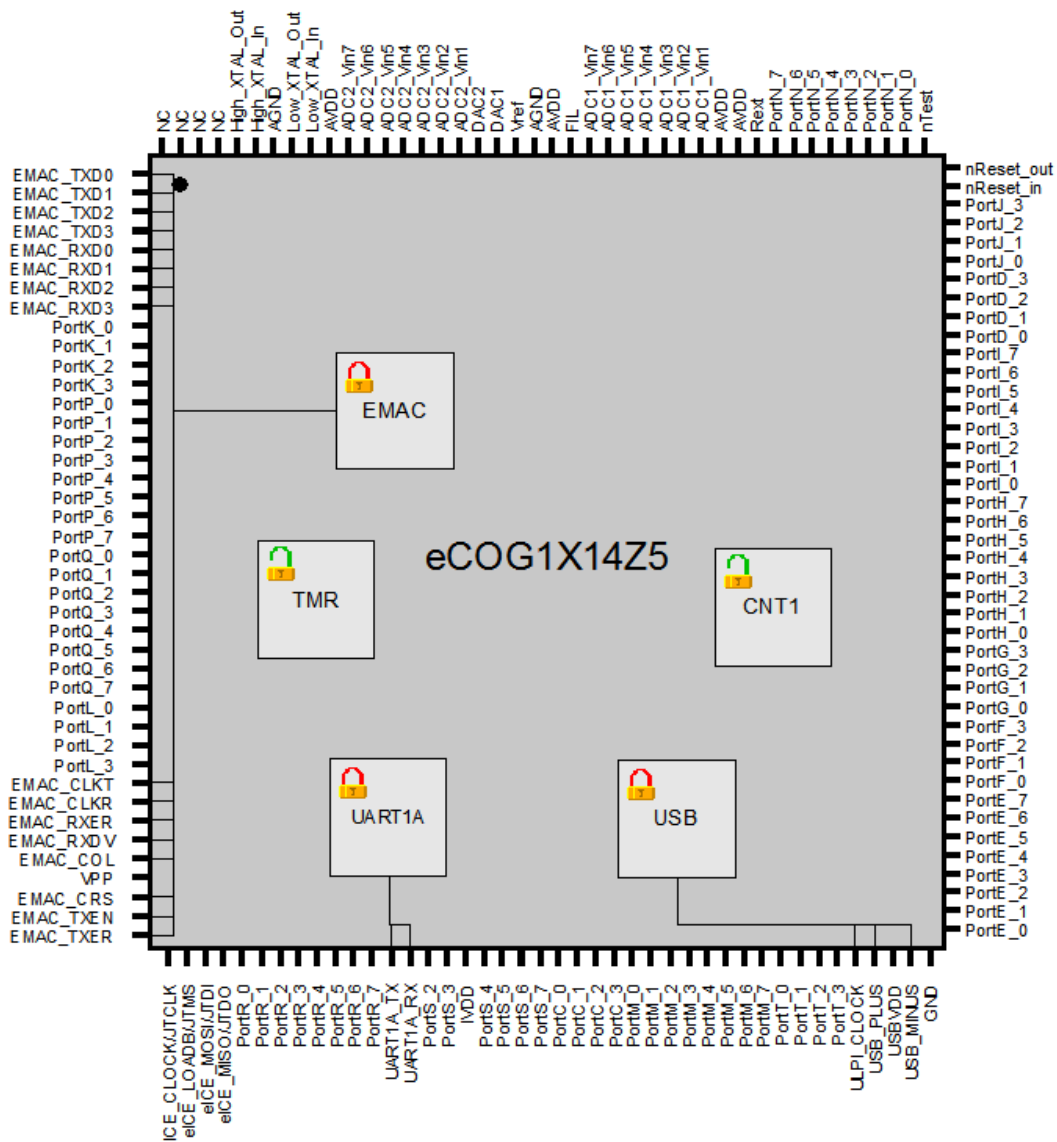


AN068 – eCOG1X FTP Server with USB File Storage

Version 1.0

This application note describes using the eCOG1X microcontroller as an FTP Server (FTP Daemon) with a USB Mass Storage Device (such as a flash memory stick) containing the file system to be accessed.



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	06/11/2007	First release

Contents

1	Introduction	5
2	Glossary	5
3	Overview.....	5
4	Requirements	5
4.1	FAT File System Support	5
4.2	uIP V1.0 and EMAC Library Update.....	5
4.3	USB Power	6
5	Using the Application	6
5.1	IP Address	6
5.2	Filename restriction	6
5.3	Implemented Commands	6
5.4	User Names and Passwords	6
5.5	FTP Clients.....	6
5.6	Simultaneous Access	7
5.7	Debug Information	7
5.8	Split Output Packets.....	7
5.9	USB Performance.....	7
6	Application Structure	8
7	Memory Use	9

1 Introduction

This application note describes using the eCOG1X microcontroller as an FTP Server (FTP Daemon) with a USB Mass Storage Device (such as a flash memory stick) containing the file system to be accessed.

2 Glossary

A table of abbreviations and terms used in this document.

eCOG1X	Cyan Technology target microcontroller.
EOF	End of File. Used as a marker in the FAT table.
FAT	File Allocation Table.
UART	Universal Asynchronous Receiver/Transmitter.
FTP	File Transfer Protocol (ref. RFC 959)

3 Overview

This application note demonstrates how to combine several of the major functional blocks of the eCOG1X microcontroller to provide an embedded FTP server that can be used to access files stored on a USB Mass Storage device, such as a flash memory stick.

4 Requirements

This project was developed using CyanIDE V1.4.2.

4.1 FAT File System Support

The software for this application note must be used in conjunction with the library software from application note AN042, "FAT File System Support for the eCOG1k and eCOG1X", contained in the zip file <AN042 FAT Libraries.zip>.

To install the FAT file system libraries:

- Download the zip file <AN042 FAT Libraries.zip> from the Cyan web site and open it.
- Click the *Extract* icon, or select *Actions->Extract* from the main menu.
- Check that the *Use folder names* option in the dialogue box is ticked.
- Set the *Extract to:* directory to the CyanIDE peripheral libraries directory. Usually this is <C:\Program Files\Cyan Technology\CyanIDE\libraries>.
- Click the *Extract* button to extract the files to the CyanIDE peripheral libraries directory.

The zip file contains two copies of the FAT library files, one for eCOG1k and one for eCOG1X. For this example, only the files for the eCOG1X are required.

4.2 uIP V1.0 and EMAC Library Update

The versions of the uIP V1.0 TCP/IP stack and EMAC Library used by this application note have been updated from those supplied with the CyanIDE V1.4.2 install. To obtain the updated libraries, download the library software used in conjunction with application note AN057, "uIP V1_0 TCP-IP Stack for eCOG1k and eCOG1X", contained in the zip file <AN057LIB.zip>.

Extract the library software file <AN057LIB.zip> to the CyanIDE install directory, usually <C:\Program Files\Cyan Technology\CyanIDE>. This updates the files in the eCOG1X EMAC peripheral library and in the uIP V1.0 library directories.

4.3 USB Power

When acting as a Host, the eCOG1X must supply the VBus power to the peripheral. On the eCOG1X Development Board, this depends on which USB interface is used.

When the external ULPI PHY connection on S6 is used, the MAX5008 power supply is capable of supplying up to 125mA, which is sufficient for most flash memory based USB drives.

For the internal USB PHY connection on S5, the power supply output from the MAX3355 is capable of only 8mA, which is sufficient for OTG devices but not for most USB flash drives. To use this connection with a USB flash drive, it is recommended that a link is placed on the board to connect the USB VBus power directly to the +5V supply. For example, connect J33 pin 1 (VBus on the internal USB connector) and J18 pin 2 (+5V supply).

5 Using the Application

Two project files are supplied, one configured to use the Internal USB PHY on the eCOG1X, connected to the USB Socket S5 on the eCOG1X Development Board, and the other is configured to use the external ULPI PHY, connected to the USB Socket S6 on the eCOG1X Development Board. When choosing which PHY to use, please read the USB Power section above.

5.1 IP Address

This application note example can be used either with a DHCP Server to allocate an IP address dynamically to the web server, or with a static IP address.

The software is supplied with the configuration set to use DHCP to acquire an IP address.

To change the software to use a fixed IP address, open the project, and then open the **uip-conf.h** header file in the Headers sub folder. At the start of this file is a definition `UIP_CONF_FIXEDADDR` that is defined as '0'. Change this to '1', and set the IP address and net mask values in the definitions immediately following it.

5.2 Filename restriction

Because the Fat File System Library does not support long file names, all the names of the files and directories must be in the 8.3 (DOS) format.

5.3 Implemented Commands

The following FTP Commands have been implemented:

USER, QUIT, NOOP, PASV, PORT, LIST, RETR, STOR, PWD, XPWD, CWD, CDUP, TYPE, ABOR, MODE, STRU, DELE, MKD, XMKD, RMD, XRMD and HELP.

5.4 User Names and Passwords

The application does not currently implement any password control on the access to the server, and accepts any username as valid.

5.5 FTP Clients

The Microsoft Windows™ operating systems are supplied with a command line FTP Client that can be invoked from the command prompt.

In addition, most Web Browsers can be used as a “read only” FTP Client, by entering the address “FTP://<IP Address>” in the address box. The contents of the FTP Server can then be browsed in an Explorer window.

5.6 Simultaneous Access

As supplied, the server can maintain three simultaneous accesses from FTP Clients. This limit is defined by the `NUM_FTP_DAEMONS` definition at the start of the *FTPD.C* file.

Note that each Daemon requires up to two TCP Connections, one for the command and the other for the data. The maximum number of simultaneous TCP connections is defined by the `UIP_CONF_MAX_CONNECTIONS` definition in the *uip_conf.h* file. As supplied, this is set to ten.

5.7 Debug Information

If a serial terminal (such as HyperTerminal) is connected to the serial port P1, at 9600 baud, 8 data bits, No parity and 1 stop bit, then debug information can be seen indicating when the server has acquired an IP address (if DHCP is used), and when a USB Mass Storage Device is attached and detached.

Note that if the Full Debug build is used, then the significant amount of debug information sent to the serial port significantly reduces the performance of the application. To mitigate this, increase the baud rate of the debug serial port.

5.8 Split Output Packets

This project makes use of the ability of uIP V1.0 to split large output packets into two, so as to achieve greater data throughput when sending data to client with TCP/IP stack using the "Nagle Algorithm" of delayed acknowledge.

The Nagle Algorithm reduces network traffic by delaying the acknowledge packets and using one single packet to acknowledge all the packets in a time interval. The algorithm is specified in RFC 896.

When uIP is used to send data to a client using this algorithm, this can significantly reduce the data throughput, as uIP can only handle a single packet at a time, and must wait for the acknowledge to come back before it can move on to send another packet.

By splitting large packets (that are the maximum segment size) into two smaller packets, the client TCP/IP stack sends the acknowledge immediately on reception of the second packet, rather than waiting for the delay. This significantly improves the data throughput when sending data from uIP to a client using this algorithm, such as the Microsoft Windows™ operating systems. Typically, the data throughput is increased by an order of magnitude.

To use this split packet method, the application calls a different uIP polling function from its main loop. This function is `uip_poll_with_arp_split_output()`, instead of the `uip_poll_with_arp()` function used previously.

Note that this is applicable only to applications that send large data packets of the maximum segment size, and has no effect on applications that send packets smaller than the maximum segment size. The effect of this can be seen by comparing the rate of transfer of files, which are sent in packets of the maximum segment size, with the speed of a directory listing, which is sent one line at a time.

5.9 USB Performance

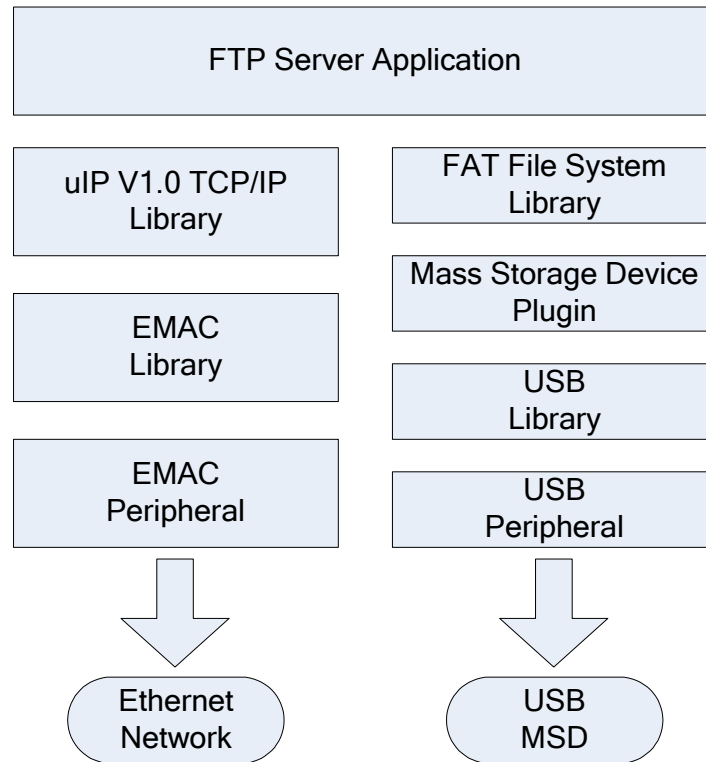
During the development of this application it has become apparent that significant variations exist in the performance of different makes of USB flash drives. As the main function of this application is to transfer data to and from the USB flash drive, then this has a significant impact on the overall performance of the application.

It is recommended that a range of USB flash drives are tested with the application, to find out which has the best performance in the target system.

Cyan cannot recommend any specific make or model of flash drive, as the volatility and turnover of this type of high volume consumer product means that any such information rapidly would become out of date.

6 Application Structure

The application is built on a series of libraries in the following fashion.



The main application is the FTP Server itself. It handles call-backs from the uIP library, which are generated when requests come in from FTP Clients across the network. These requests are then interpreted by the application, and the FAT File System library is used to access the USB Mass Storage Device. This is then passed back to the client through the uIP Library and across the network. Data then flows through the TCP Data Connection, either from the USB Mass Storage Device to the Ethernet Network or vice versa, through the application.

7 Memory Use

The application uses the following amounts of memory:

Code: 62K Bytes of Flash

Constants: 3K Bytes of Flash

Variables: 15K Bytes of Internal RAM

Note that as this application uses the USB Peripheral, a maximum of 20K Bytes of Internal RAM are available to the application, as 4K bytes are used internally by the USB peripheral.

The RAM requirements can be reduced by reducing the size of the `UIP_CONF_BUFFER_SIZE` constant defined in the `uip-conf.h` header file. This results in a three-fold reduction in the RAM usage as this defines the size of three buffers, one used by the uIP TCP/IP Library, plus one transmit buffer and one receive buffer used by the EMAC Library. Thus, if the value of the constant `UIP_CONF_BUFFER_SIZE` is reduced to 500 bytes (from its original value of 1518 bytes), then the RAM requirements of the application is reduced to 12K bytes, a saving of approximately 3K bytes.