Small Embedded Ethernet Converters

FMod-TCP family

Web Page Uploading Manual

Version 2.0

five co
innovative engineering

Version: **2.0**
Last revision: **August 15, 2011**
Printed in **Switzerland**

Trademarks
Windows® is a registered trademark of Microsoft Corporation.
Ethernet® is a registered trademark of Xerox Corporation.
Java® is a registered trademark of Sun Microsystem.
Philips® is a registered trademark of Koninklijke Philips Electronics N.V.
Borland® is a registered trademark of Borland Software Corporation.

Warning
This device is not intended to be used in medical, life-support or space products.

Any failure of this device that may cause serious consequences should be prevented by implementation of backup systems. The user agrees that protection against consequences resulting from device system failure is the user's responsibility. Changes or modifications to this device not explicitly approved by FiveCo will void the user's authority to operate this device.

Support
Web page: http://www.fiveco.ch/prod_products.php
e-mail: support@fiveco.ch

Revision history

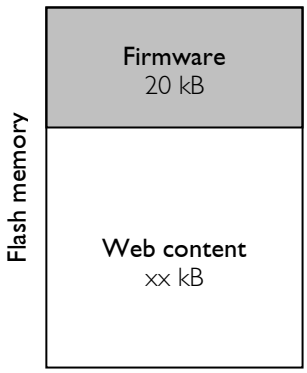
Revision	Date	Author	Note	Firmware version
1.0	20.12.04	AG	- First revision	Since 5.8
1.1	27.01.05	AG	- Corrections made to load process and add loadAnswerCommand	Since 5.8
1.3	26.09.05	AG	- Text and layout update	Since 6.00
1.4	10.01.08	AG	- Address changed	Same
2.0	14.08.11	AG	- Add 24 bits FAT	Since 7.0 (TCP DB) Since 2.0 (TCP BOX)

Table of Contents

1 Introduction

Hardware considerations

The FMod-TCP products family is based on a flash microcontroller with embedded flash memory. Since the firmware only uses a small amount of this memory, the remaining part is used to save web content. It is however limited to 44kBytes for the FMod-TCP DB and FMod-TCP BOX and to 108kBytes for the FMod-TCP BOX 2.



Software considerations

Since the available memory is limited, no real file system is provided and it is impossible to change each file separately. The whole content must be updated in one shot.

The steps of that updating process are the following:

- **Flash memory erasing.**
- **Web content preparation on the updater computer.**
- **Web content uploading on the FMod-TCP.**

The flash memory erasing and web-content uploading processes use the same protocol as the one used to access the internal registers but with different Function IDs. Please refer to the device user manual for detailed information about this protocol.

Note:
Since the firmware-upgrader uses the same flash region to temporarily save the new firmware, the previous web content will be destroyed by the firmware-upgrading process.

2 Flash erasing

The first step of that process consists in the erasing of the flash memory. The user must send the following command to the card:

Flash erasing command:

Byte#		Number of bits	Example
0x00	Flash erase (0x0031)	16 bits	0x0031
0x02	TransactionID	16 bits	0x0000
0x04	Number of parameters (0)	16 bits	0x0000
0x06	Checksum	16 bits	0xFFCE

The erasing process needs several seconds to be completed. Once done, the "FMod-TCP" module will answer with the "MemoryErased" ASCII chain in a Load File Answer packet.

Byte#		Number of bits	Example
0x00	Load file answer (0x0033)	16 bits	0x0033
0x02	TransactionID	16 bits	0x0000
0x04	Number of parameters (14)	16 bits	0x000E
0x06	"MemoryErased"	12 bytes	
0x12	LF CR	16 bits	0x0A0D
0x14	Checksum	16 bits	0xBC19

The user must wait for this answer before continuing the process!

3 Web content preparation

The web content that will be uploaded onto the FMod-TCP card has to be previously prepared based on the original web files (html, jpeg ...) from the host computer.

This content consists of two different parts:

- the **Files Table**
- the **Files Data**

The packet (bytes-chain) resulting from the concatenation of those two parts can be uploaded onto the card (see next chapter of this document).

Files Table

The "Files Table" provides the offset position and length of each file present in the Files Data part.

This table ends with a **"$$"** ASCII string.

There is two version of Files table available: the 24bits and the legacy 16bits. Both versions are understood by the latest firmware, but if the board used has more than 64kBytes available for the web data, the 24bits version should be used.

24bits version

The Files table begins with the "£FF24" ASCII chain.

Each file entry has the following structure:
"/" + [file name] + " " (space) + [file offset (24bits)] + [file size (24bits)].

The [file offset] is calculated from the beginning of the flash web area, so the "Files Table" length is included in this offset.

Example:

```
Address   00   02   04   06   08   0A   0C   0E       ASCII

  00000   A346 4632 342F 4643 4F43 5365 7175 656E £FF24/FCOCSequen
  00010   6365 722E 6A61 7220 0000 3201 AE10 2F69 cer.jar ..2.®./i
  00020   6E64 6578 2E68 746D 6C20 01AE 4200 01D4 ndex.html .®B..Ô
  00030   2424                                    $$
```

This sample "Files Table" shows 2 files:

	Offset	Length
FCOCSequencer.jar	0x000032 (50)	0x01AE10 (110096)
index.html	0x01AE42 (110146)	0x0001D4 (468)

16bits version

Each file entry has the following structure:
"/" + [file name] + " " (space) + [file offset (16bits)] + [file size (16bits)].

The [file offset] is calculated from the beginning of the flash web area, so the "Files Table" length is included in this offset.

Example:

```
Address   00   02   04   06   08   0A   0C   0E       ASCII

  00000   2F69 6E64 6578 2E68 746D 6C20 004A 01F9 /index.html .J..
  00010   2F6C 6F67 6F2E 6A70 6567 2002 4313 C92F /logo.jpeg .C../
  00020   4D6F 6475 6C65 5661 7269 6162 6C65 2E63 ModuleVariable.c
  00030   6C61 7373 2016 0C04 6F2F 5443 502E 636C lass ...o/TCP.cl
  00040   6173 7320 1A7B 5558 2424                ass .{UX$$
```

This sample "Files Table" shows 4 files:

	Offset	Length
index.html	0x004A (74)	0x01F9 (505)
logo.jpeg	0x0243 (579)	0x13C9 (5065)
ModuleVariable.class	0x160C (5644)	0x046F (1135)
TCP.class	0x1A7B (6779)	0x5558 (24848)

Files Data

The second part, "Files Data", is simply the content of each file concatenated one after the other, IN THE SAME ORDER AS THE "FILES TABLE"!

4 Uploading the web contents into the card

Since the flash memory is organized in blocks of 64 bytes, the downloading process uses the following protocol with 64 data bytes.

Load web data command:

Byte#		Number of bits	Example
0x00	Load (0x0032)	16 bits	0x0032
0x02	TransactionID	16 bits	0x0001
0x04	Number of parameters (70)	16 bits	0x0046
0x06	Data offset (multiple of 64)	24 bits	0x000040
0x09	Data length (always 64)	24 bits	0x000040
0x0C	64 data bytes	64 bytes	0x...
0x4C	Checksum	16 bits	0x????

The FMod-TCP answers each packet with a Load File Answer frame:

Byte#		Number of bits	Example
0x00	Load file answer (0x0033)	16 bits	0x0033
0x02	TransactionID	16 bits	0x0001
0x04	Number of parameters	16 bits	0x????
0x06	Command state		0x...
0x??	LF CR	16 bits	0x0A0D
0x??	Checksum	16 bits	0x????

The command state can be one of the following ASCII chains:

- **"BytesWritten"** if there is no error
- **"Not8BytesAligned"** if the command packet does not have 64 data bytes to write.
- **"FlashMemoryExceded"** if there is more than 44 or 108kBytes of data.

The user has to send a packet of 64 bytes, wait for an answer, and then send the next 64 bytes packets until all data has been transmitted.

If less than 64 bytes of data remain in the last packet, the remaining bytes can be filled with any values.

5 Code samples

Here is a C++ (Borland™) code sample to create a "Files Tables" (16bits) + "Files Data" bytes stream.

To prepare that stream, this code sample creates an object that represents the list (**LBFiles**) of all web files to be uploaded onto the module. This **LBFiles** list is a **CWWWFile** objects list. Each CWWWFile object contains the "FileName", the "FileData" and the "FileSize" of one specific file.

```
class CWWWFile
{
public: // User declarations
        __fastcall CWWWFile() ;
        __fastcall ~CWWWFile() ;

        AnsiString FileName ;        // File name
        TMemoryStream *FileStream ;  // File data
        int FileSize ;               // File size
};
```

There are also two global variables:

```
        int TotalFilesSize ;
        DynamicArray < Byte > FATBuf ;
```

The next function fills the final buffer (**FATBuf**) that will be uploaded onto the board, based on the file list **LBFile**s.

```
void TIPForm::FillFATBuf()
{
    // Fill FATBuf from the list LBFiles

    int DataBufIndex = 0 ;
    int FATBufIndex = 0 ;
    int FATSize = 0;
    CWWWFile *File ;
    DynamicArray < Byte > DataBuf ;



    //Size initialisation of DataBuf
    DataBuf.Length = TotalFilesSize ;

    for (int i=0; i<LBFiles->Count; i++)
    {
        File = (CWWWFile*) LBFiles->Items->Objects[i] ;
        FATSize += File->FileName.Length() + 6 ;
                            // for space + file pointer H/L and fileSize H/L
                            and /
    }
    FATSize += 2 ; // for $$ at the end of the File All Table



    // DataBuf will be added to end of FATBuf. It must be long enough.
    FATBuf.Length = FATSize + DataBuf.Length ;

    for (int i=0; i<LBFiles->Count; i++)
    {
        File = (CWWWFile*) LBFiles->Items->Objects[i] ;

        // Fill data buf
        File->FileStream->Seek(0,0) ;
        File->FileStream->Read(&DataBuf[DataBufIndex],File->FileStream->Size);

        // Fill FAT buf
        FATBuf[FATBufIndex++] = '/' ;
        for (int j=1; j<=File->FileName.Length(); j++)
            FATBuf[FATBufIndex++] = File->FileName[j] ;

        FATBuf[FATBufIndex++] = ' ' ;
        FATBuf[FATBufIndex++] = (Byte) ((FATSize + DataBufIndex)>>8&0xFF);
        FATBuf[FATBufIndex++] = (Byte) ((FATSize + DataBufIndex)&0xFF) ;
        FATBuf[FATBufIndex++] = (Byte) ((File->FileSize)>>8&0xFF) ;
        FATBuf[FATBufIndex++] = (Byte) ((File->FileSize)&0xFF) ;

        DataBufIndex += File->FileStream->Size ;
    }


    FATBuf[FATBufIndex++] = '$' ;
    FATBuf[FATBufIndex++] = '$' ;

    // Copy data buf to the end of FAT buf
    for (int k=0; k<DataBuf.Length; k++)
        FATBuf[FATBufIndex++] = DataBuf[k] ;

    // free Data buf memory
    DataBuf.Length = 0 ;
}
```

6 Example

Here is a byte stream example that can be downloaded onto the card (the 16bits Files Table is indicated in yellow):

```
Address   00   02   04   06   08   0A   0C   0E       ASCII

  00000   2F69 6E64 6578 2E68 746D 6C20 0012 01FA /index.html ...ú
  00010   2424 3C68 746D 6C3E 0D0A 3C68 6561 643E $$<html>..<head>
  00020   0D0A 3C74 6974 6C65 3E46 6976 6543 6F27 ..<title>FiveCo'
  00030   7320 5443 502F 4950 2043 6F6E 7472 6F6C s TCP/IP Control

  00040   6C65 723C 2F74 6974 6C65 3E0D 0A3C 2F68 ler</title>..</h
  00050   6561 643E 0D0A 0D0A 3C62 6F64 7920 6267 ead>....<body bg
  00060   636F 6C6F 723D 2223 3030 3030 3030 2220 color="#000000"
  00070   7465 7874 3D22 7768 6974 6522 206C 696E text="white" lin

  00080   6B3D 2223 3636 3636 3636 2220 766C 696E k="#666666" vlin
  00090   6B3D 2223 3636 3636 3636 2220 616C 696E k="#666666" alin
  000a0   6B3D 2223 3636 3636 3636 223E 0D0A 3C63 k="#666666">..<c
  000b0   656E 7465 723E 3C48 323E 464D 6F64 2D54 enter><H2>FMod-T

  000c0   4350 3C2F 4832 3E3C 703E 0D0A 0D0A 3C41 CP</H2><p>....<A
  000d0   5050 4C45 540D 0A09 636F 6465 093D 2022 PPLET...code.= "
  000e0   5443 502E 636C 6173 7322 0D0A 0977 6964 TCP.class"...wid
  000f0   7468 093D 2022 3630 3022 0D0A 0968 6569 th.= "600"...hei

  00100   6768 7409 3D20 2235 3030 220D 0A09 3E0D ght.= "500"...>.
  00110   0A3C 2F41 5050 4C45 543E 0D0A 0D0A 3C2F .</APPLET>....</
  00120   6365 6E74 6572 3E0D 0A0D 0A3C 6365 6E74 center>....<cent
  00130   6572 3E0D 0A3C 6872 3E0D 0A3C 696D 6720 er>..<hr>..<img

  00140   7372 633D 226C 6F67 6F2E 6A70 6567 2220 src="logo.jpeg"
  00150   7769 6474 683D 2233 3332 2220 6865 6967 width="332" heig
  00160   6874 3D22 3736 2220 626F 7264 6572 3D22 ht="76" border="
  00170   3022 3E3C 6272 3E0D 0A26 636F 7079 3B20 0"><br>..&copy;

  00180   6279 2046 6976 6543 6F20 496E 6E6F 7661 by FiveCo Innova
  00190   7469 7665 2045 6E67 696E 6565 7269 6E67 tive Engineering
  001a0   202D 2031 3031 3520 4C61 7573 616E 6E65 - 1015 Lausanne
  001b0   3C62 723E 0D0A 3C61 2068 7265 663D 2268 <br>..<a href="h

  001c0   7474 703A 2F2F 7777 772E 6669 7665 636F ttp://www.fiveco
  001d0   2E63 6822 3E68 7474 703A 2F2F 7777 772E .ch">http://www.
  001e0   6669 7665 636F 2E63 683C 2F41 3E0D 0A3C fiveco.ch</A>..<
  001f0   2F63 656E 7465 723E 0D0A 3C2F 626F 6479 /center>..</body

  00200   3E0D 0A3C 2F68 746D 6C3E 0D0A xxxx xxxx >..</html>..xxxx
  00210   xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxxxxxxxxxxxxxx
  00220   xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxxxxxxxxxxxxxx
  00230   xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxxxxxxxxxxxxxx
```

This bytes stream contains one html file.

	Offset	Length
index.html	0x0012 (18)	0x01FA (506)

The length of this byte stream is 9 x 64 bytes.

Contact address :

FiveCo - Innovative Engineering
Ch. de la Rueyre 116
CH-1020 Renens
Switzerland
Tel: +41 21 632 60 10
Fax: +41 21 632 60 11

www.fiveco.ch
info@fiveco.ch