See page 12 for quick start

Small Embedded Ethernet Converter

FMod-TCP (DB)

User Manual

Version 2.8

innovative engineering

Version: 2.8
Last revision: September 23, 2005
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 a medical, life-support or space product.

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/section_motion/support_motion_E.htm
e-mail: support@fiveco.ch

Table of Contents

Revision history

Revision	Date	Author	Note	Firmware version	Applet version	Win32 app version
1.3.2	02.02.04	AG	- First approved version	Since 3.0		
1.4	13.04.04	AG	- Added UART fixed settings definition and UART and I2C connectors' commercial reference.			
2.0	28.06.04	AG	- Updated whole manual for TCP0201	Since 5.1	Since 1.1	
2.1	07.07.04	AG	- Added PCB board dimensions. - Added comments to UART port and connectors pin out. - Correction made to layout of UART and I2C connectors. - Correction made in PORT DIRECTION register description and added default boot state.	Since 5.1	Since 1.1	
2.2	01.09.04	AG	- Added UDP port 7010. - Updated applet and win32 app description. - Updated default IP. - More explanations on UART use without flow ctrl.	Since 5.6	Since 1.5	
2.3	13.10.04	AG	- Correction to I2C pinning: SCL and SDA were inverted.	Since 5.6	Since 1.5	
2.4	19.10.04	AG	-Add more explanation for IP address reset jumper behavior.	Since 5.8	Since 1.5	
2.5	18.01.05	AG	-Add I2C scan and I2C speed change. -Add FMod-TCP DB -Change IP config method (quick start) -Add 4800bds for UART	Since 5.10	Since 1.6	Since 2.11
2.6	22.04.05	XG	- Add Memory organization - Add second UART to RS adapter	Same	Same	Same
2.7	20.07.05	AG	-Add I2C Read Write with ack command.	Since 5.13	Same	Since 2.13
2.8	26.09.05	AG	- Small text corrections - Add checksum function example - Add explanation about I2C	Same	Same	Same

1 Package and operating conditions

Starter Kit contents

The FMod-TCP "Starter kit" should contain:

- one FMod-TCP Server board
- one UART to RS232 (female connector) converter
- one UART to RS232 (male connector) converter
- one 5V Power supply
- one CD-Rom with dedicated software
- This manual

Operating conditions

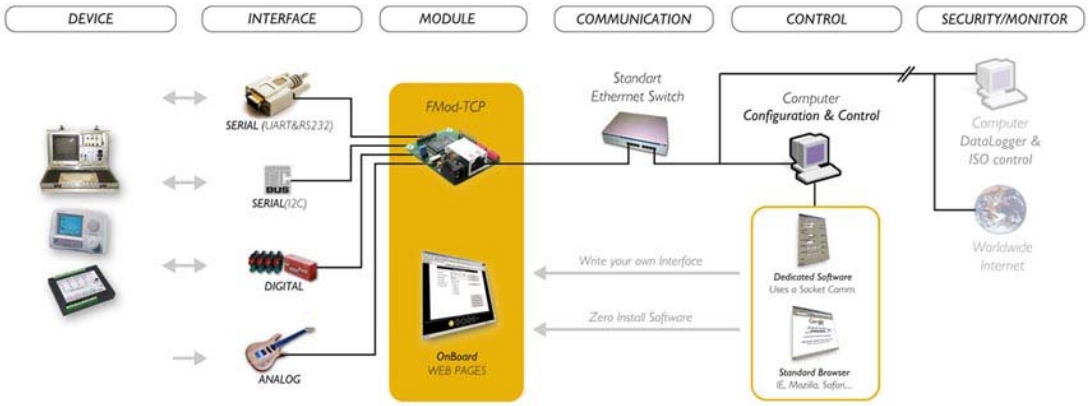
- Supply voltage : 5VDC ±10%
- Supply current : 100mA (50mA typical)
- Operating temperature : 0 – 70 °C
- I/O maximum current : 20mA (max 200mA for all I/Os together)

The RJ-45 connector is protected against ESD by a grounded varistor. The other components are not protected!

2 Overview

Applications

The FMod-TCP is a TCP/IP server board that allows manufacturers and system integrators to connect different devices such as home appliances, industrial sensors and industrial control systems directly to the Ethernet network, (10BaseT) and to remotely monitor & control those using standard protocols.



It can be accessed through a TCP socket connection from a computer or through a simple Web Page in a standard browser which can be directly loaded into the device (max 44kb). The module is delivered with a default web page including a Java Applet enabling the controlling of the card.

The connection between this device and the user's product can be done through the following interfaces:

- **UART** (up to 115200 bps with or without hardware flow control). (TCP port # 8000.)
- **Up to 19 I/O**. (TCP port # 8010 or UDP port # 7010.)
- **Up to 5 A/D converter**. (TCP port # 8010 or UDP port # 7010.)
- **I2C bus** (2 wires bus). (TCP port # 8010 or UDP port # 7010.)

Note:
Some examples of applications illustrating the use of the card with UART, I2C, I/O and A/D devices can be found on the FiveCo's website:

http://www.fiveco.ch/section_motion/tcp_db/real_tcp_E.htm

Software operating principle

The operating principle for PC softwares that must exchange data with an FMod-TCP card depends on which interface is used.

In case of **UART** use, the operation is really simple. Any byte sent to port TCP #8000 is simply redirected to the UART bus and any byte received from UART bus is redirected to the TCP connection.

In the other cases (**I/O, A/D and I2C**), the software has to use a dedicated protocol layer on top of the TCP Layer (see chapter 5**Erreur ! Source du renvoi introuvable.**). This protocol is Question & Answer oriented. The PC should send a Question, wait for the Answer and so on.

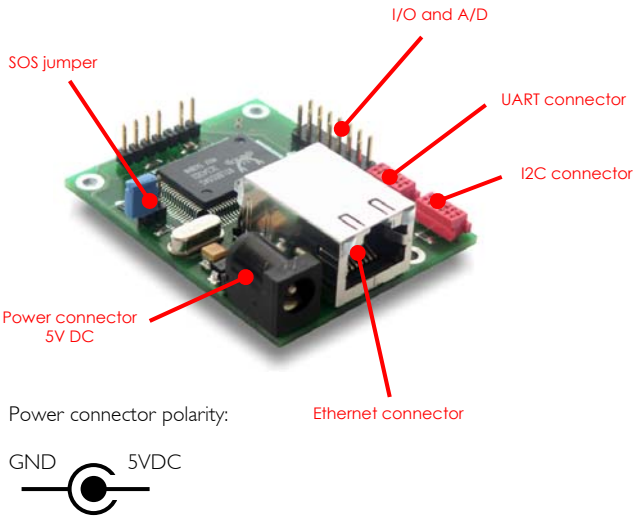
To configure the card's parameters and to access I/O and A/D features, the protocol uses an Internal Registers Access routine (see chapter 5 and 8).

The Java Applet example and the Borland C++ code sample available on the FiveCo's web site can help programmers get started with their development.

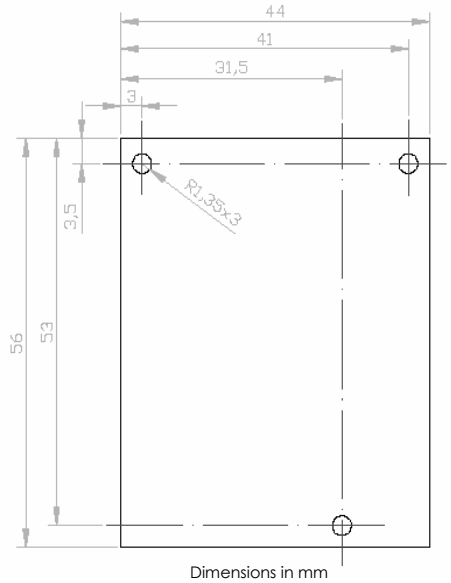
Hardware description

Two hardware versions exist. Both are OEM boards to be integrated inside the customer's product.

Cable version

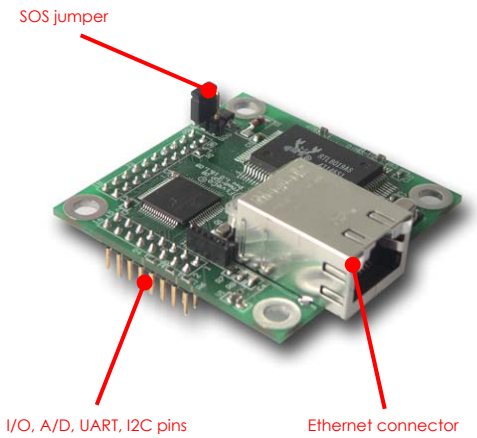


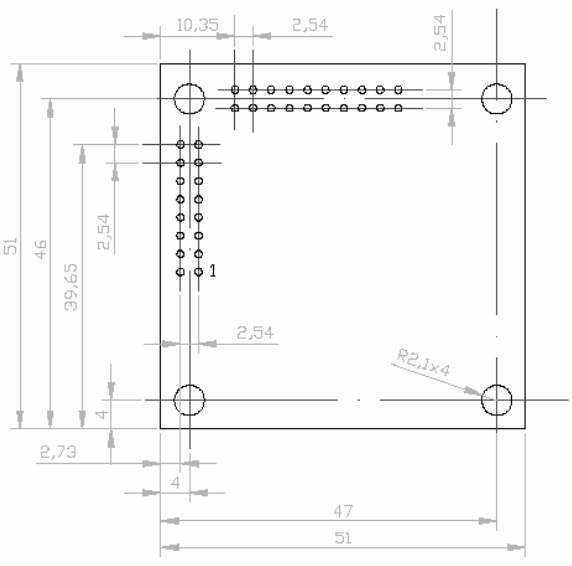
Power connector polarity:



Dimensions in mm

Daughter Board (DB) version



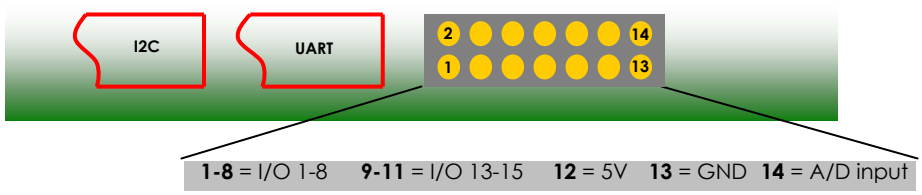


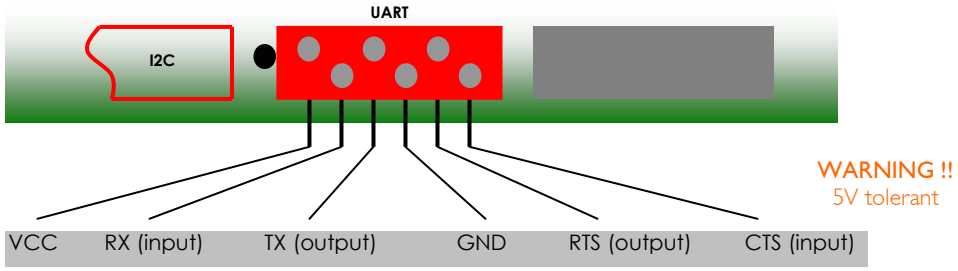
Dimensions in mm

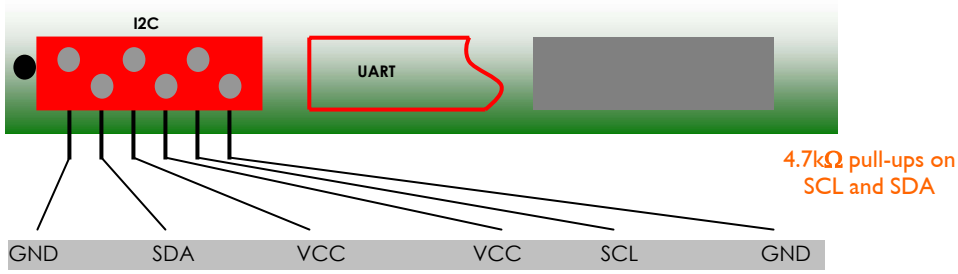
3 Inputs & Outputs

Beware:
Before connecting any device to the following connectors, unplug power from the FMod-TCP!

FMod-TCP



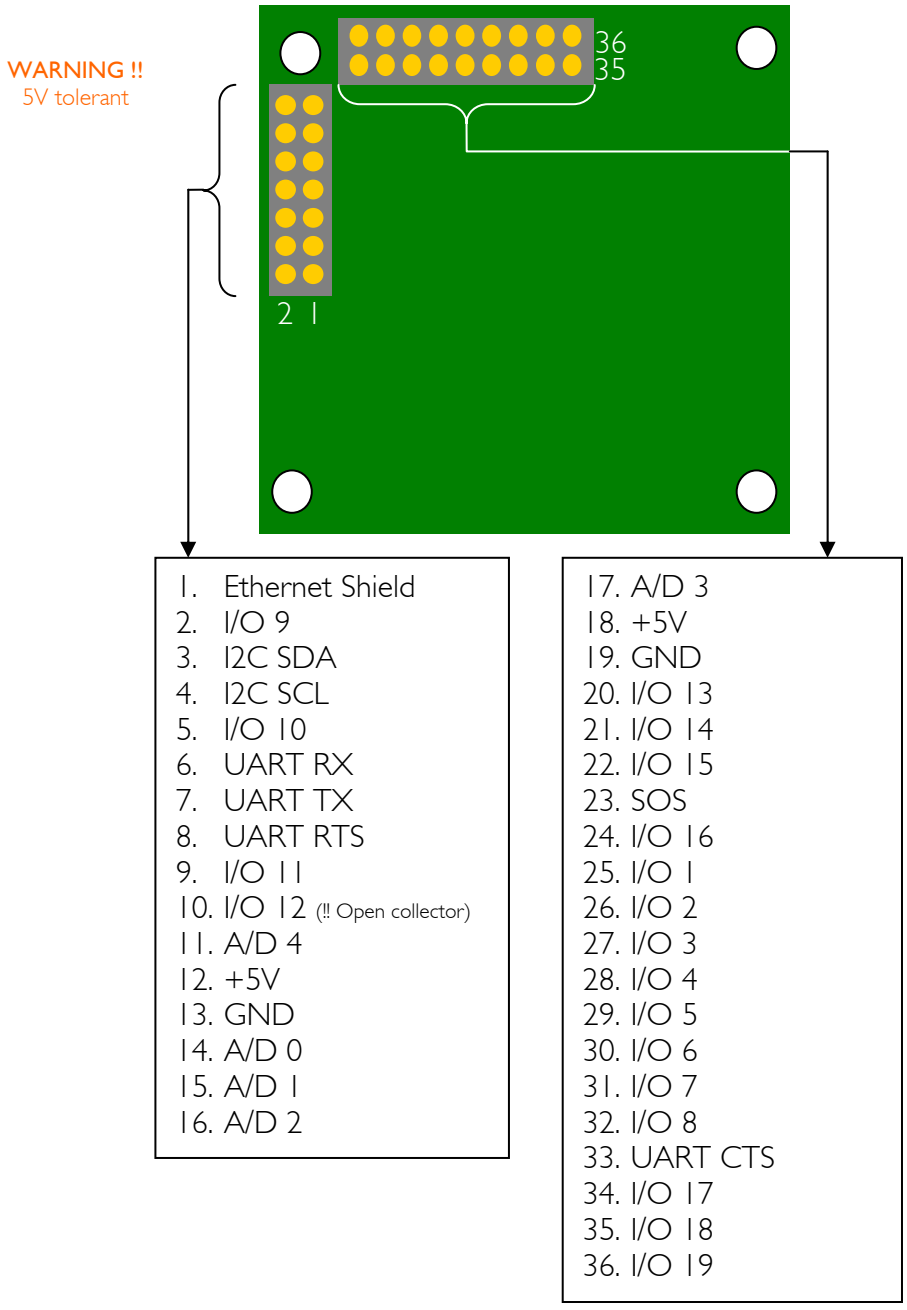




- The connectors for **I2C** and **UART** are from "Tyco-electronics" model "Micro-MaTch" (Manufacturer reference **# 7-215079-6**).
- Cable-side connectors' reference number is **# 7-215083-6**.

I/O are Schmitt Trigger type.

FMod-TCP DB



Warning:
The I/O 12 (pin 10) is an open collector input (standard output). Don't forget to add a pull-up if your application needs one. Other I/Os are Schmitt Trigger type.

SOS jumper

A jumper is dedicated to restore default IP address or factory settings.

Two cases can happen:

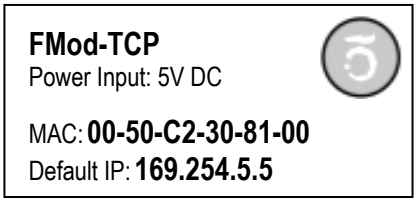
- If you plug it **while the card is running**, the IP address will be restored as soon as all TCP connections are closed. *Warning: you have to send Save Settings command to the card to save it into EEPROM.*
- If the **jumper is plugged during startup**, the default IP address and factory settings of all parameters will be restored AND saved into EEPROM. You do not need to send Save Parameters command in this case.

On the FMod-TCP DB, the SOS line is provided on the 2.54 pins connector to provide this feature to the main board.
This pin is driven high by a 10k pull-up resistor. Force it low to reset IP address.

4 Quick start

This section is intended to help users to quickly plug the module into their system and establish a connection between the computer and the card. Detailed information about hardware and software is described further in this document.

You can find the board's factory communication settings on the following label.



The Ethernet MAC Address is fixed and cannot be changed. The IP Address can be changed. The complete procedure is described below.

If the IP address has been changed, you can retrieve the default value by plugging the "SOS jumper" (See section 3).

Changing IP address

To easily change the factory IP address, use the Win32 software provided on the CD-Rom.

1. Plug your new card on your PC network.
2. Start the Win32 application.
3. Click on "File->Easy change IP address".
4. The software will scan the network and display a list of all FiveCo's devices found.
5. Select the MAC address corresponding to your new card.
6. If you have more than one network adapter on your PC, the software will ask you to select the one which is connected to the same network as the FMod-TCP.
7. The software will suggest a new IP address with the last byte left open. Choose a new IP (**Not already used on your network!!**) and click the "Change IP address" button.

That's it! The card has a new address and a new subnet mask (the same as your PC). They are automatically saved into EEPROM.

You can now connect the card with the Win32 software or open its web page by typing its new IP address into a web browser.

Notes:

- The IP address won't be changed if a TCP connection exists with the card.
- The protocol used to change the IP address is described later in this manual.

5 Controlling the FMod-TCP by TCP or UDP

General Information

All the board's parameters (configuration registers) and features can be accessed through a TCP or UDP port.
In addition, an HTTP-TCP port is available for web pages downloading and another TCP port for UART bus access.

Those ports are:

- TCP Port **#80** for HTTP communication.
- TCP Port **#8000** for UART transceiver.
- TCP Port **#8010** to access I/O registers (see chapter 8) and I2C bus.
- UDP Port **#7010** to access I/O registers (see chapter 8) and I2C bus.

With regards to TCP connections, the board allows up to **4 simultaneous connections**.

These ports are described below.

UART (TCP # 8000)

The UART bus of the microcontroller is accessible through the TCP port number 8000. The module acts simply as a transceiver for this port. Any byte sent from the network (ex: TCP-IP from a PC) to the module will be sent to the other side's UART bus, and vice versa. Thus there is no particular protocol dedicated to this feature. See later chapters to know how to change parameters such as baud rate and hardware flow control.
Note: This port supports only one user at a time.

UART fixed settings: *No parity / 1 Start Bit / 8 Data Bits / 1 Stop Bit*

Important note about baud rate greater than 9600bds:

Common TCP/IP stacks (on PC, Unix station ...) use a delay of 200ms for the acknowledgement of the TCP received data packets. This is done to reduce traffic on the network because TCP allows the acknowledgement of several packets at one time.

Unfortunately, the FMod-TCP module needs this acknowledgement to remove the data from its internal UART receive buffer (if no acknowledge is received from the PC, the module will resend those data).

So, with speeds greater than 9600bds, the buffer may be filled faster than data can be sent by TCP and part of those will be lost if no UART hardware flow control is used between the card and the UART device (CTS and RTS lines).

If you cannot use hardware flow control on UART bus and you have to get more than 250 bytes at one time, the solution is to reduce the TCP acknowledgement delay on your computer.

For Windows™ 2000/XP users, you can add/change the following value in the registry. **BEWARE**: *improperly done changes in the Windows registry can results in a system crash! Such changes are the user's full responsibility!*

Entry: *HKey_Local_Machine\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces**Interface-name***
Key: *TcpDelAckTicks*
Value: *DWORD with value 0x00*

The **interface-name** is the registry name (32 digits number) of your Ethernet card which you use to access the module.
The following web page describes this feature in details:

http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/regentry/58801.asp

Board parameters and I/O, A/D and I2C features (TCP # 8010 or UDP # 7010)

The main TCP port number 8010 or UDP port number 7010 can be used to change some important parameters of the module:

- **Baud rate and flow control**
- **TCP timeout value**
- **IP address**
- **Module name**

The user should use the Win32 application enclosed in the package or the default java applet loaded in the module to change those parameters. If the user wants to change the parameters by himself, the protocol is defined below.

This port is also used to **access I/O value and direction registers, A/D conversion result registers** (see page 36 for a complete description of those registers) **and I2C bus**. The I2C feature is described after "registers access feature".

The last feature accessible through this port is the "Easy IP config" that is used in the "Quick start" chapter of this document.

Registers access feature:

TCP/IP works in big endian: most significant byte first, followed by least significant byte.
The access to the data is done through an easy (6 byte header) protocol over TCP.

Structure of each packet:

1. **Function ID (2 bytes),**
2. **Transaction ID (2 bytes)**
3. **Length of the parameters (2 bytes)**
4. **Parameters (X byte)**
5. **Checksum (2 bytes) (described later in this chapter)**

The user (sender) defines the values of the Transaction IDs himself. The module that receives a command sends back an answer (for every command). The answer contains the same Transaction ID as the corresponding command sent. The user is also able to check execution of each command.

Read register(s) value command:

Byte#		Number of bits	Example
0x00	Read (0x0021)	16 bits	0x0021
0x02	TransactionID	16 bits	0x1B34
0x04	Number of registers to read (X)	16 bits	0x0001
0x06	X * Registers Addresses	X * 8 bits	0x02
0x06+X	Checksum	16 bits	0x...

The maximum number of registers that can be read at one time is almost 30. The answer sequence should not be greater than 180 bytes. If the number of registers is too big, the FMod-TCP will answer only with the value of some of them.

The module answers with the following sequence:

Byte#		Number of bits	Example
0x00	Read Answer (0x0023)	16 bits	0x0023
0x02	TransactionID (same as demand)	16 bits	0x1B34
0x04	Number of bytes in answer	16 bits	0x0019
0x06	Register address	8 bits	0x02
...	Register value	8–128 bits (16B)	0x12345
The two previous entries are replicated for every register that has been asked for reading			
...	Checksum	16 bits	0x...

Write register(s) value command

Byte#		Number of bits	Example
0x00	Write (0x0022)	16 bits	0x0022
0x02	TransactionID	16 bits	0x1B34
0x04	Number of bytes in command	16 bits	0x0003
0x06	Register Addresses	8 bits	0x02
0x07	Register value	8 – 64 bits	0x1234
The two previous entries are replicated for every register that has been asked for reading			
...	Checksum	16 bits	0x...

The max length of this sequence is 180 bytes.

The module answers with the following sequence:

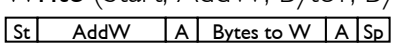
Byte#		Number of bits	Example
0x00	Write Answer (0x0024)	16 bits	0x0024
0x02	TransactionID (same as demand)	16 bits	0x1B34
0x04	0x0000	16 bits	0x0000
0x06	Checksum	16 bits	0x...

I2C feature

The Inter-IC bus, commonly known as the I²C bus, is a control bus that provides the communications link between integrated circuits in a system. Developed by Philips in the early 1980's, this simple two-wire bus has become the de facto worldwide standard for system control, finding its way into everything from temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters, CODECs, and microprocessors of all kinds.
You can find the I2C's specifications on the Philips web site at the following link: http://www.semiconductors.philips.com/

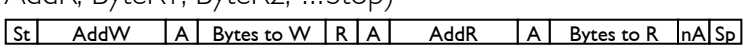
The I2C protocol can access a device by three different manners:

- **Write** (Start, AddW, Byte1, Byte2, ..., Stop)



- **Read** (Start, AddR, Byte1, Byte2, ..., Stop)

- **Read After Write** (Start, AddW, ByteW1, ByteW2, ..., ReStart, AddR, ByteR1, ByteR2, ...Stop)



To be able to do all of these 3 sequences, use this command sequence:

Byte#		Number of bits	Example
0x00	I2CRWwithAck (0x0007)	16 bits	0x0007
0x02	TransactionID	16 bits	0x1B34
0x04	LengthOfParameters (X + 3)	16 bits	0x0005
0x06	7 bits Address (bit 7 = 0)	8 bits	0x28
0x07	X (number of bytes to write)	8 bits	0x02
0x08	xBytes	X bytes	0xAF1D
....	Y (number of bytes to read)	8 bits	0x05
The four previous entries can be replicated to access the same or other I2C slaves within this command sequence.			
	Checksum	16 bits	0x...

If X = 0, the Read method is used.
If Y = 0, the Write method is used.
If X & Y ≠ 0, the Read after Write method is used.

The answer sequence is the following one:

Byte#		Number of bits	Example
0x00	I2CReadAnswer (0x0008)	16 bits	0x0008
0x02	TransactionID (same as demand)	16 bits	0x1254
0x04	Number of bytes in answer	16 bits	0x0005
0x06	Answer bytes	Y bytes	0x1A25...
...	Ack state of the I2C com.	1 byte	0x87
If the same or other I2C slave have been accessed in the command, the answer bytes and ack state is added here.			
...	Checksum	16 bits	0x...

The "Ack state" byte is composed of the following bits:

0	Address ack in write sequence	0 = No answer to this address 1 = ack received
1	Bytes written ack (each byte was acked)	0 = Bytes not acknowledged 1 = ack received
2	Address ack in read sequence	0 = No answer to this address 1 = ack received
3-6	Reserved	-
7	Must be always 1	1

The user can use these bits to check for the presence of his I2C devices and monitor hardware issues.

Note that the max length of those sequences is 180 bytes. **Pay close attention to building sequences that do not exceed this and not to ask too much byte in answer !**

The FMod-TCP translates automatically those sequences to I2C sequences. **It is mandatory that the sequence has to be transmitted within one TCP packet.** Otherwise, the FMod-TCP will ignore it.

Former I2C sequences

For backward compatibility, the following protocols are also available, but do not allow to confirm the acknowledge status of the I2C communication.

This protocol is not recommended for new designs!

Read sequence:

Byte#		Number of bits	Example
0x00	I2CRead (0x0001)	16 bits	0x0001
0x02	TransactionID	16 bits	0x1B34
0x04	LengthOfParameters (X + 3)	16 bits	0x0005
0x06	7 bits Address (bit 7 = 0)	8 bits	0x28
0x07	X (number of bytes to write)	8 bits	0x02
0x08	xBytes	X bytes	0xAF1D
....	Y (number of bytes to read)	8 bits	0x05
	Checksum	16 bits	0x...

If X = 0, the microcontroller uses the Read method instead of the Read after Write method.

The answer sequence is the following one:

Byte#		Number of bits	Example
0x00	I2CReadAnswer (0x0003)	16 bits	0x0003
0x02	TransactionID (same as demand)	16 bits	0x1254
0x04	Number of bytes in answer	16 bits	0x0005
0x06	Answer bytes	Y bytes	0x1A25
...	Checksum	16 bits	0x...

Note: The max number of bytes that can be read/write at one time is 64.

Write sequence:

Byte#		Number of bits	Example
0x00	I2CWrite (0x0002)	16 bits	0x0002
0x02	TransactionID	16 bits	0x2001
0x04	LengthOfParameters (X + 2)	16 bits	0x0003
0x06	Address	8 bits	0x28
0x07	X	8 bits	0x01
0x08	xBytes	X bytes	0x1A
	Checksum	16 bits	0x...

The answer sequence is the following one:

Byte#		Number of bits	Example
0x00	I2CWriteAnswer (0x0004)	16 bits	0x0004
0x02	TransactionID (same as demand)	16 bits	0x2001
0x04	Number of bytes in answer	16 bits	0x0000
0x06	Checksum	16 bits	0x...

The controller translates automatically those sequences to I2C sequences. It is mandatory that the sequence be transmitted within one TCP packet, otherwise the microcontroller will ignore it.

The calculation of the checksum is described later in this document.

Note: *The max number of bytes that can be read/write at one time is 64.*

I2C Bus scanning

The following command allows user to ask an I2C bus scanning to list which addresses answer with an acknowledge.

I2C scan sequence:

Byte#		Number of bits	Example
0x00	I2CScan (0x0005)	16 bits	0x0005
0x02	TransactionID	16 bits	0x2001
0x04	Number of addresses to scan	16 bits	0x0001
0x08	X Addresses	X bytes	0x1A
	Checksum	16 bits	0x...

The FMod-TCP answers with the following sequence:

Byte#		Number of bits	Example
0x00	I2CScanAnswer (0x0006)	16 bits	0x0006
0x02	TransactionID (same as demand)	16 bits	0x2001
0x04	Number of valid addresses	16 bits	0x0001
0x06	Valid addresses list	n bytes	0x1A
	Checksum	16 bits	0x...

Note:
If there is no address in the I2C Scan command, the FMod-TCP will scan all addresses between 1 and 127!

I2C speed change (advanced features)

The I2C bus speed can be changed at any time by changing the content of the I2CSPD parameter.

Beware:

1. Do not change I2C speed if it is not mandatory. Speed greater than 100kHz are not supported by all I2C slaves.
2. Do not change I2C speed during I2C communication.
3. The I2C controller does not match all I2C specifications at speed higher than 100kHz. This feature will therefore not work with all I2C slaves.

The value of the I2CSPD parameter must be computed with the following formula:

$$I2CSPD = \frac{10^7}{I2CSpeed_{wanted}} - 1$$

The I2C speed cannot be saved in EEPROM.

Easy IP address config (UDP # 7010)

A really useful feature of the UDP port #7010 is the "Easy IP config" one.

The user who wants to design his own software can use this feature to do a "quick start/install" method. Indeed, since this protocol uses a broadcast UDP packet, even if the device is not in the same subnet, it should receive its new IP address and subnet mask.

Procedure:
Send a UDP broadcast message (using a local or direct broadcast IP address) to your network (inside which the FMod-TCP is connected) with the following command:

Byte#		Number of bits	Example
0x00	Change IP fct (0x002A)	16 bits	0x002A
0x02	TransactionID	16 bits	0x0000
0x04	Length of params (0x000E)	16 bits	0x000E
0x06	FMod-TCP Mac Address	6 bytes	0x0050C2308101
0x0C	FMod-TCP new IP Address	4 bytes	0xC0A81064
0x10	FMod-TCP new SubnetMask	4 bytes	0xFFFF0000
0x14	Checksum	16 bits	0x...

If the FMod-TCP recognizes its MAC address, it will answer this command with a simple acknowledges and change its IP address and subnet mask IF NO TCP CONNECTION IS MADE TO THE BOARD.

Byte#		Number of bits	Example
0x00	Change IP fct ack (0x002B)	16 bits	0x002A
0x02	TransactionID	16 bits	0x0000
0x04	Length of params (0x0000)	16 bits	0x0000
0x14	Checksum	16 bits	0x...

Checksum calculation

This checksum is the same as the IP checksum.

Definition: sum of 1's complement of all 16 bits words of whole message (FiveCo packet) except checksum bytes.

Note: all values are unsigned!

Sequence:

1. Clear accumulator

Loop

x. Only if last word is not made of two bytes, the data byte is the upper byte (big endian)
2. Compute 1's complement of each 16bits word, result is 16bits
3. Convert last result from 16 bits to 32 bits, result is 32bits: 0x0000+last result
4. Add last result to the 32 bits accumulator

Try the Loop

5. Convert accumulator in two 16bits words
6. Add those two 16bits words, result is 16bits word.
7. If an overflow occurs with the last addition (Carry), add 1 to the last result.
8. Last result is the final result

Example (in hexadecimal):

```
 !0x0021  (0XFFDE) → 0x0000FFDE (Read)
+!0x1234  (0xEDCB) → 0x0001EDA9 (TransID)
+!0x0003  (0xFFFC) → 0x0002EDA5 (3 reg to read)
+!0x0A10  (0XF5EF) → 0x0003E394 (reg 0A,10,02)
+!0x02(00)(0XFDFF) → 0x0004E193
```

Note that in this case a last 00 is implicitly used. (02 → 02 00).

```
0x0004 + 0xE193 = 0xE197, (carry=0)
0xE197 + carry  = 0xE197
```

`Checksum = 0xE197`

Here is an example of a checksum calculation function in C:

```
int RetCheckSum(Byte* ByteTab, int Size)
{
    // This function returns the calculated checksum
    unsigned int     Sum=0;
    bool             AddHighByte=true;
    unsigned int     ChecksumCalculated;

    for(int i=0;i<Size;i++)
    {
        if(AddHighByte)
        {
            Sum+=((ByteTab[i])<<8)^0xFF00;
            AddHighByte=false;
        }
        else
        {
            Sum+=(ByteTab[i])^0x00FF;
            AddHighByte=true;
        }
    }
    if (AddHighByte==false)
    Sum+= 0xFF;

    ChecksumCalculated = ((Sum>>16)&0xFFFF)+(Sum&0xFFFF);

    ChecksumCalculated = ((ChecksumCalculated>>16)&0xFFFF)
                        +(ChecksumCalculated&0xFFFF);

    return ChecksumCalculated;
}
```

This function needs a Byte array (ByteTab) containing the command sequence and this array's length (Size) as input, it returns the checksum as an int.

6 Java Applet

A specific Java Applet is provided with the module to control any of its ports without having to write any specific code.

Overview

To connect to the http server on the card, simply open your web browser and type the IP address of the module. Example with default address:

"http://169.254.5.5"

The applet is downloaded from the module to your computer and runs as a local process (on your computer). You need to use an internet web browser that is compatible with Java 1.1.

Please note that on an MSWindows™ based computer, a few seconds delay can occur when you download the applet due to an MSWindows™ NetBios issue.

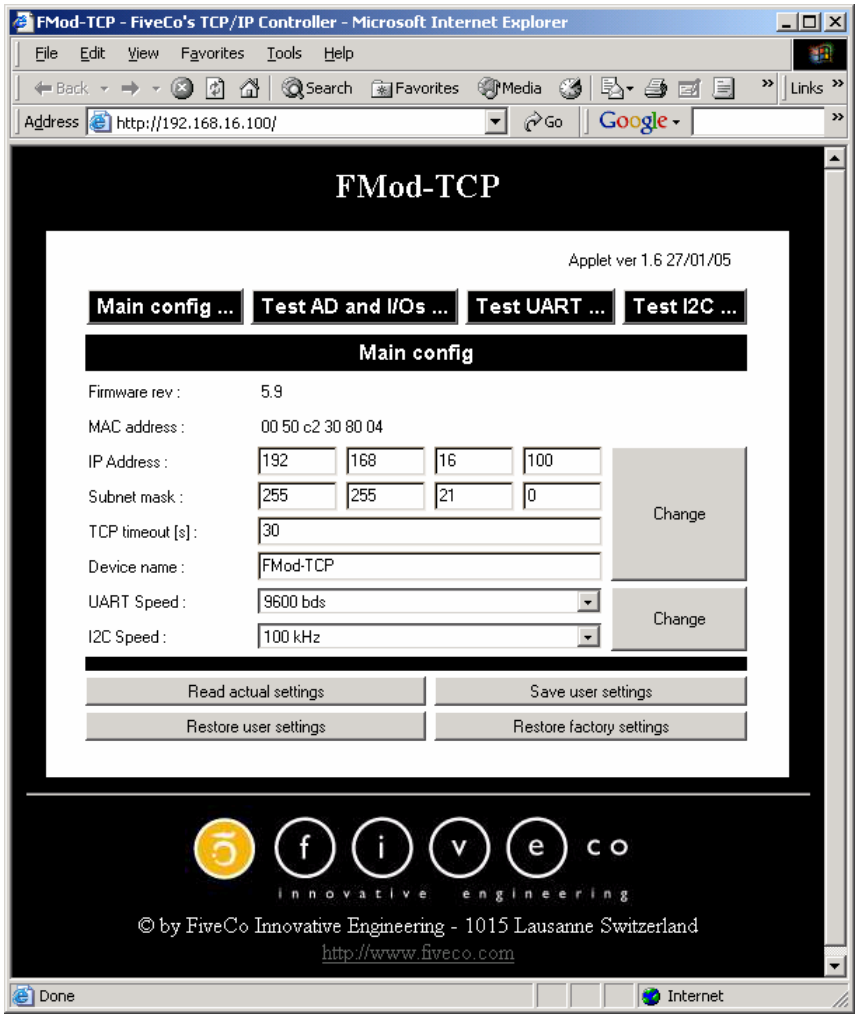
The navigation through the four panels of the applet is done through the menu bar:

Main Config

The Main panel shows the general information related to the module.

• The first part allows the user to change the main settings of the card. Don't forget to use the "Save user parameters" button to make changes permanent!

• The second part allows the user to save/restore user and factory parameters and to read the actual configuration of the device if it has been modified by another application.

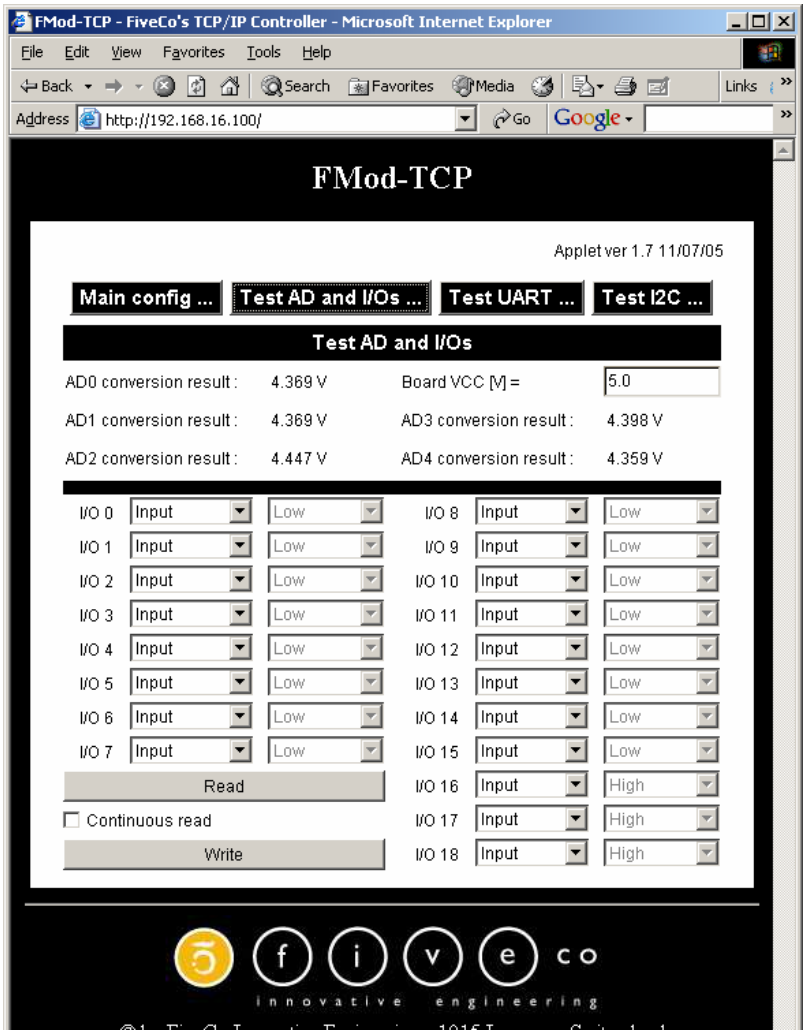


Test A/D and I/Os

This page can be used to access A/D and I/O ports.

The A/D value is regularly updated automatically (20 kHz). You can change the board Vcc value, used to convert data to a voltage, to be more accurate.

You can also change each I/O port direction and state.

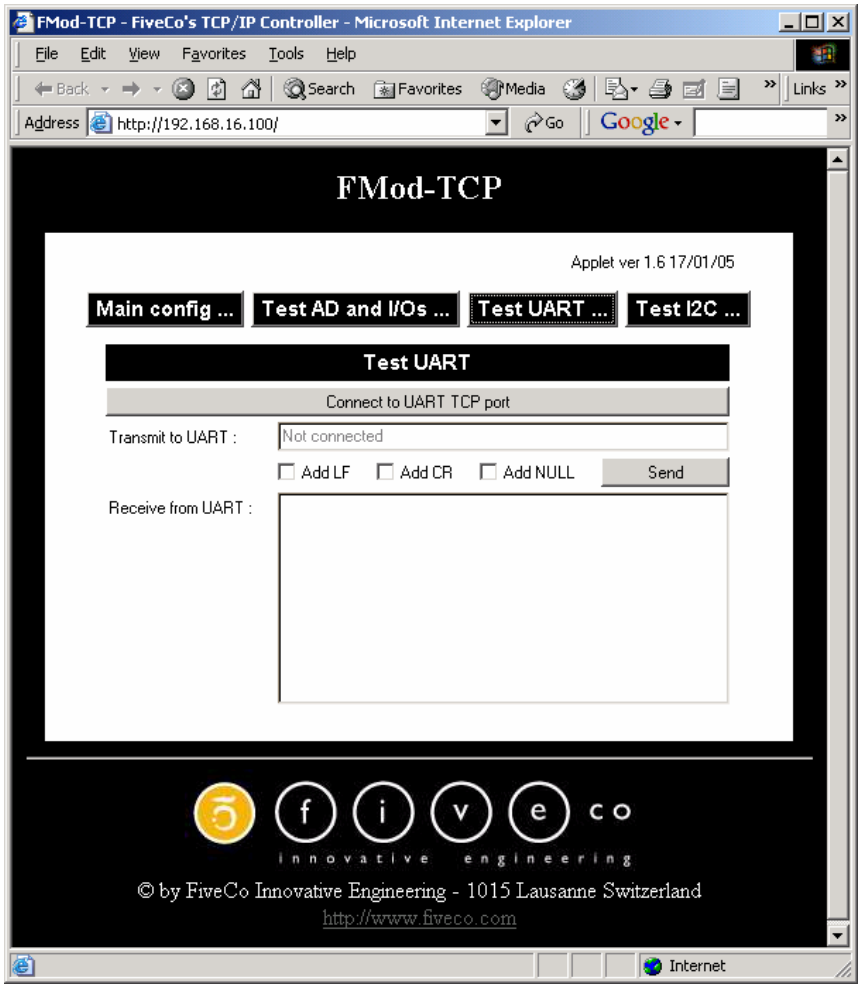


The "Continuous read" box allows reading continually the port for monitoring purposes. In other cases, use the "Read" button to read port direction and value and "Write" button to write port direction and port value.

Test UART

This page can be used to access UART bus.

If you click on the "Connect to UART TCP Port" you can send ASCII data to UART bus and receive data from it (only ASCII is visible in this application).



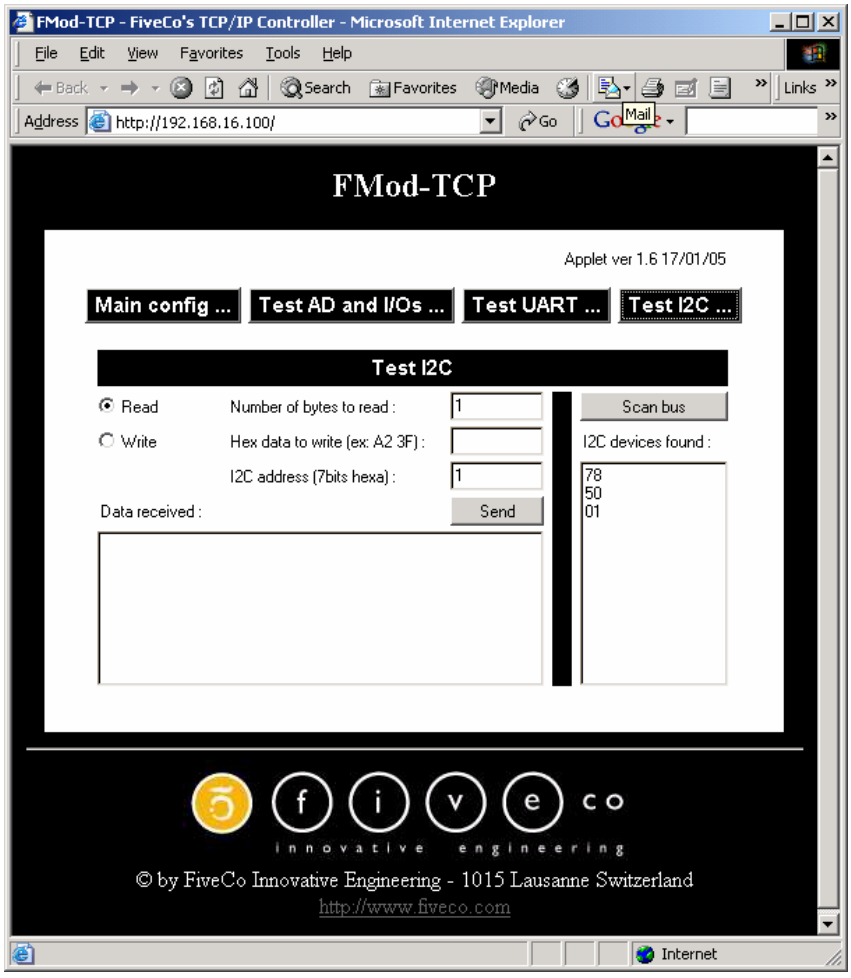
You can check "Add LF", "Add CR" and/or "Add Null" boxes to add a Line Feed, a Carriage Return and/or a Null Byte at the end of the ASCII chain sent to the device.

Test I2C

This page can be used to access I2C bus.

The panel is divided in two parts:

- The part on the right allows scanning the I2C bus to find available slave devices.
- The part on the left allows sending a read or a write command to a device on the I2C bus.

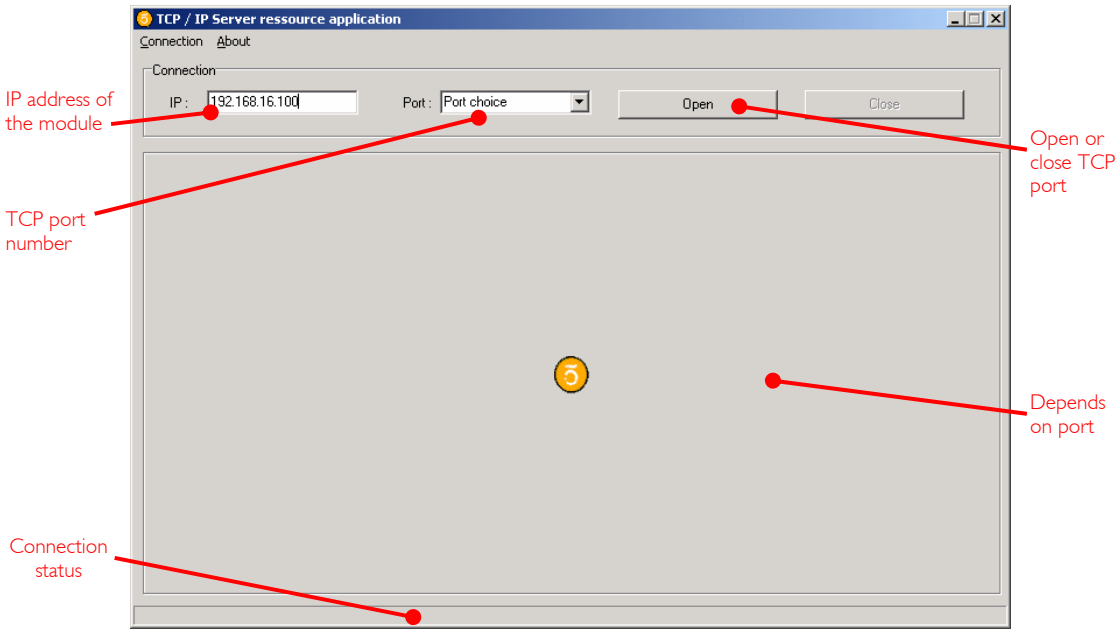


You should first check if you want a read or a write. Then you have to write hex data to write to the device in the "Hex data to write" field (if you checked the "Read" box, the FMod-TCP will use the read after write I2C feature).
After writing the I2C address (7bits hexa) and the number of bytes to read (if applicable), click on the "Send" button. The answer is displayed in hex in the "Data received" field.

7 Win32 Application

A specific application is provided with the module to control any of its ports without having to write a specific code.

Overview

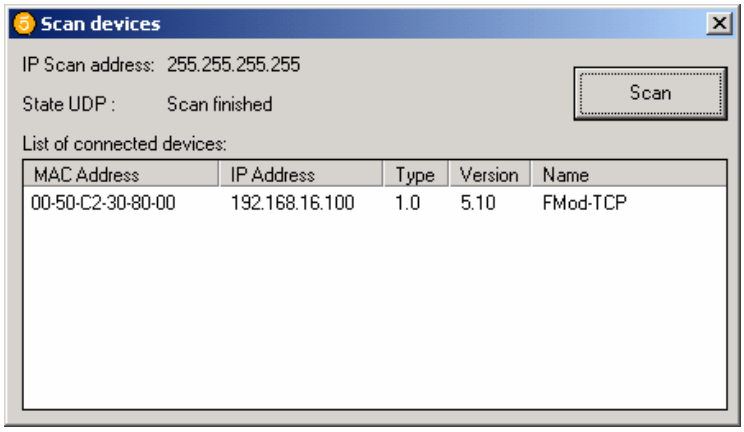


To open a TCP port on the module, the user has to set the correct IP address of the module, to choose the correct TCP port in the list and to click on the "Open" button. To close the port, simply click on the "Close" button.

The status bar displays the status of the connection:

- ***Connected***
- ***Disconnected***
- ***Error of connection*** (if connection was not established within 30s)

Note: *If you forgot the IP address of the module, you can use the "Scan network" feature of the "Connection" menu.*



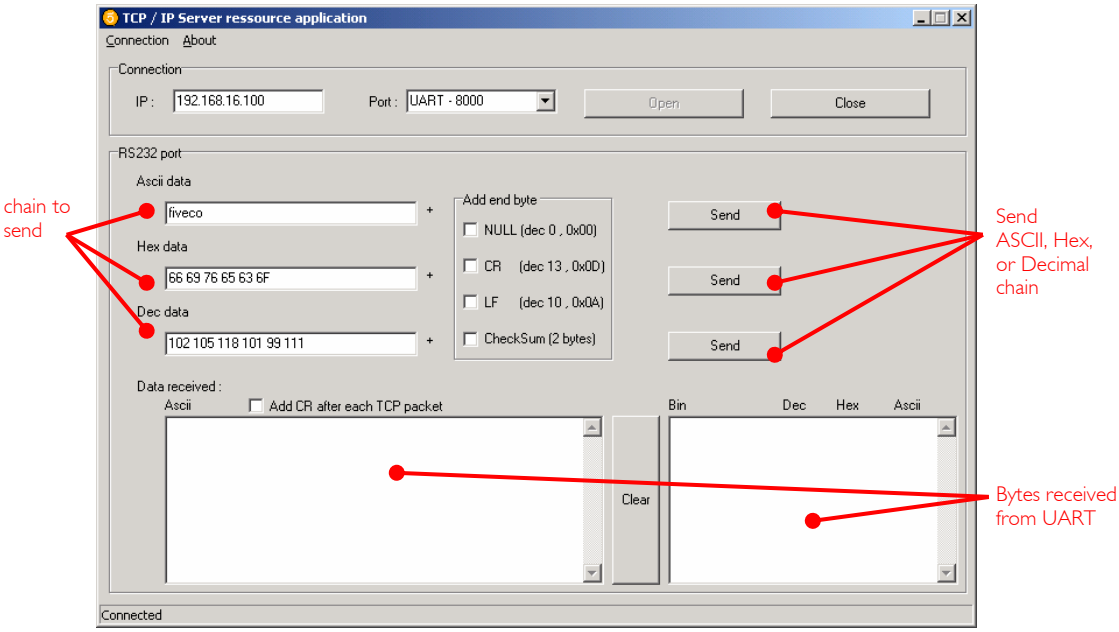
This application uses the IP address of your computer and its subnet mask to find the scan broadcast address. When you click on scan, a broadcast message will be sent to all boards in your subnet and answers will be listed.

WARNING: it only works with cards in your subnet!

How does it work?
When you click "Scan", the software simply sends a "Read registers" command to a broadcast address on UDP port number 7010 (see chapter about main port at page 15).

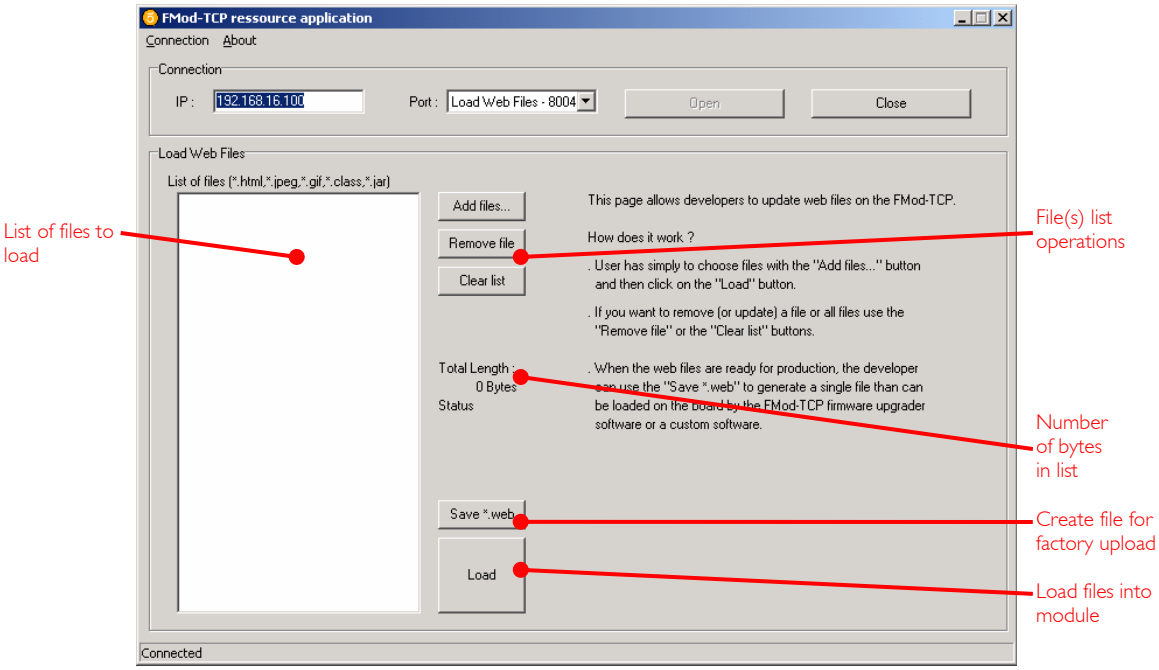
The broadcast address depends on the network subnet mask present on your PC.

UART interface



The UART interface is easy to use. Just write the ASCII, HEX or decimal string in the dedicated text box and click corresponding "Send" button. The received bytes are displayed into the "Data received" boxes (same data but different representation).

"Load web files" interface

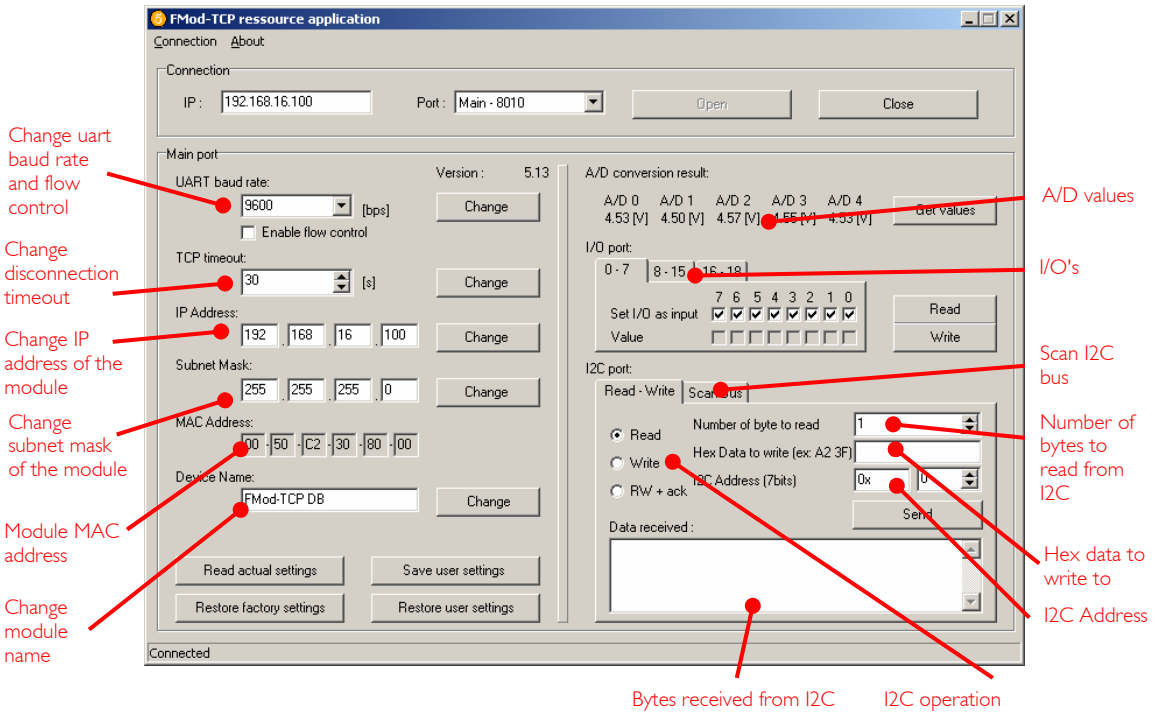


This option allows users to load their personal web files (html, java, jpeg, gif, txt ...) into the flash memory. A maximum of **44kB** is available for that application.

An example of pages and Default Java Applet code is available on the Start Kit CDRom or on the FiveCo's web page.

For factory web upload, the user can save a single file yourname.web. See FMod-TCP_WebPageUploading manual on the FiveCo web site.

Main port interface



The configuration interface allows the user to change some of the board's settings.

UART baud rate: The user can change the UART baud rate and enable or disable the hardware flow control (CTS/RTS).

TCP timeout: The user can change the number of seconds allowed before TCP port is disconnected. This feature avoids the problems due to the crash of a TCP client (PC).

IP address: The user can change the IP address of the module.

Subnet mask: The user can change the subnet mask of the module.

MAC address: Mac address of the module (cannot be changed).

Name: Module name.

The A/D values are simply obtained with a click on the "get value" button.

The I/Os interface allows users to choose the direction (Input or Output) of the 11 I/Os (19 for FMod-TCP DB) and their values if they are in output mode. When an I/O direction checkbox is checked, the input mode is enabled.

The I2C interface is an easy way to test the I2C connection between the module and the user's electronic. The user has to choose between an I2C Read or Write operation.

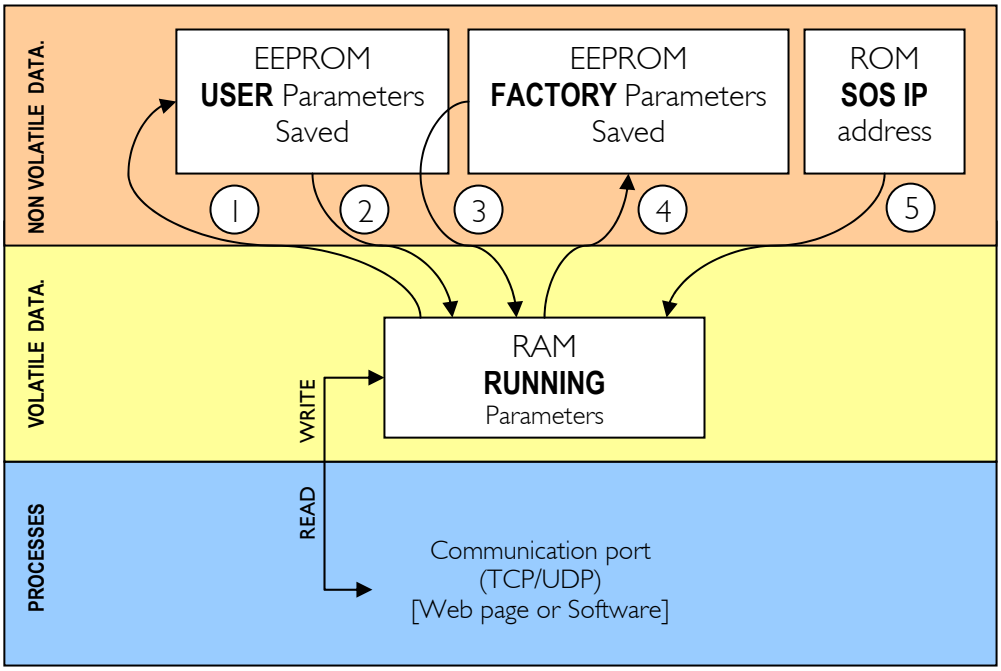
In **Write mode**, the bytes (in hexadecimal like 41 42 45 separated by spaces!) must be written in the dedicated line and the I2C address in the dedicated Address space (in hexadecimal or in decimal). Then click "Send".

In **Read mode**, the steps are the same except that the user must specify how many bytes have to be read. Data to write is optional (only used for ReadAfterWrite I2C method).

8 Registers management

Memory Organization

The user must know that a new register value sent through the communication port is loaded to the running parameters in RAM and used for the current process. All these parameters are lost upon power-down. It is required to save them to "User Parameters" or "Factory Parameters" with the corresponding function.



Action Number and description:

1	***SaveUserParameters*** (0x03) function
2	During standard power-up or calling ***RestoreUserParameters*** (0x04) function
3	***RestoreFactoryParameters*** (0x05) function
4 + 1	***SaveFactoryParameters*** (0x06) function [For integrators engineers only]
5	By setting "SOS Jumper" after power-up
3 + 5 + 1	By setting "SOS Jumper" during power-up

Full Register Description

List of registers

Address	Bytes	Name
General Information		
0x00 (00)	4	*TYPE*
0x01 (01)	4	*VERSION*
0x02 (02)	0 (fct)	*RESETCPU*
0x03 (03)	0 (fct)	*SAVEUSERPARAMETERS*
0x04 (04)	0 (fct)	*RESTOREUSERPARAMETERS*
0x05 (05)	0 (fct)	*RESTOREFACTORYPARAMETERS*
0x06 (06)	0 (fct)	*SAVEFACTORYPARAMETERS*
Communication		
0x10 (16)	4	*COMOPTIONS*
0x11 (17)	6	*ETHERNETMAC*
0x12 (18)	4	*IPADDRESS*
0x13 (19)	4	*SUBNETMASK*
0x14 (20)	1	*TCPTIMEOUT*
0x15 (21)	16	*MODULENAME*
0x16 (22)	1	*UARTCONFIG*
0x18 (24)	1	*I2CSPD*
0x1A (26)	1	*TCPCONNECTIONSOPENED*
External ports (some registers apply only to FMod-TCP DB)		
0x20 (32)	1	*IODIRECTION0*
0x21 (33)	1	*IOVALUE0*
0x22 (34)	2	*AD0VALUE*
0x23 (35)	2	*AD1VALUE*
0x24 (36)	2	*AD2VALUE*
0x25 (37)	2	*AD3VALUE*
0x26 (38)	2	*AD4VALUE*
0x27 (39)	1	*IODIRECTION1*
0x28 (40)	1	*IOVALUE1*
0x29 (41)	1	*IODIRECTION2*
0x2A (42)	1	*IOVALUE2*

TYPE

Register Address	Register Name	Function	Read/Write Control
0x00	*TYPE*	Product ID	Read only

Register Size	Register structure	
4 Bytes	Unsigned Int 16bits (HH-HL) *TYPE*	Unsigned Int 16bits (LH-LL) *MODEL*

Description:
Product identifier composed with a *Type* and *Model* number.
It defines which kind of peripheral it is.
Normally different modules *TYPE* are not software compatible.

Example:
TYPE = 0x00010001 means *Type*=1 (1= FMod-TCP), *Model* = 1

VERSION

Register Address	Register Name	Function	Read/Write Control
0x01	*VERSION*	Software ID	Read only

Register Size	Register structure	
4 Bytes	Unsigned Int 16bits (HH-HL) *Version*	Unsigned Int 16bits (LH-LL) *Revision*

Description:
Firmware identifier composed with a *Version* and *Revision* number.
Same *Version* with different *Revision* is backward compatible.

Example:
Firmware 0x0001000A = *Version* 1, *Revision* 10 is compatible with all earlier revisions of the same version (ver 1.0 to 1.9). However, it has new functionalities (which are deactivated by default) or code optimizations.

RESET CPU

Function Address	Function Name	Function	Read/Write Control
0x02	*RESETCPU*	Restart processor	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Reboots the card. The communication will be lost.

SAVE USER PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x03	*SAVEUSERPARAMETERS*	Saves all in EEPROM	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Saves the following parameters to user EEPROM space:

- 0x12 *IPADDRESS*
- 0x13 *SUBNETMASK*
- 0x14 *TCPTIMEOUT*
- 0x15 *MODULENAME*
- 0x16 *UARTCONFIG*

RESTORE USER PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x04	*RESTOREUSERPARAMETERS*	Restores saved values	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Restores the following parameters from user EEPROM space:

- 0x12 *IPADDRESS*
- 0x13 *SUBNETMASK*
- 0x14 *TCPTIMEOUT*
- 0x15 *MODULENAME*
- 0x16 *UARTCONFIG*

RESTORE FACTORY PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x05	*RESTOREFACTORYPARAMETERS*	Factory default	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Restores the following parameters from factory EEPROM space:

- 0x12 *IPADDRESS*
- 0x13 *SUBNETMASK*
- 0x14 *TCPTIMEOUT*
- 0x15 *MODULENAME*
- 0x16 *UARTCONFIG*

Note:
SAVEUSERPARAMETERS should be performed after this function in order to save restored parameters as user parameters.

SAVE FACTORY PARAMETERS

Function Address	Function Name	Function	Read/Write Control
0x06	*SAVEFACTORYPARAMETERS*	Save factory default	Write only

Register Size	Register structure	Unit
0 Byte	none	none

Description:
Saves the following parameters to factory EEPROM space:

- 0x12 *IPADDRESS*
- 0x13 *SUBNETMASK*
- 0x14 *TCPTIMEOUT*
- 0x15 *MODULENAME*
- 0x16 *UARTCONFIG*

Note:
This feature should only be used by a system integrator that would change the initial factory default settings.

COM OPTIONS

Register Address	Register Name	Function	Read/Write Control
0x10	*COMOPTIONS*	Communication options	Read/Write

Register Size	Register structure	Unit
4 Bytes	32 individual bits	none

Description:
This register is reserved for future use.

ETHERNET MAC

Register Address	Register Name	Function	Read/Write Control
0x11	*ETHERNETMAC*	Hardware network ID	Read only

Register Size	Register structure	Unit
6 Bytes	6 x Unsigned Bytes	none

Description:
A standard hardware unique identifier (worldwide) for each device on an Ethernet network.

Note:
If the user writes into this register, the MAC address will not be modified. This register is available only for informational purposes.

IP ADDRESS

Register Address	Register Name	Function	Read/Write Control
0x12	*IPADDRESS*	IP network ID	Read/Write

Register Size	Register structure	Unit
4 Bytes	4 x Unsigned Bytes	none

Description:
Network identifier used for TCP/IP and UDP/IP.
The values 255 (0xFF) and 0 (0x00) are reserved for broadcast and network addresses and should not be used in this register.

Notes:
The module will change for a new IP address only when all of its communications ports are closed.
Do not forget to use a *SAVEUSERPARAMETERS* command.

Default value:
169.254.5.5

Example:
For the IP=192.168.16.14 (0xC0, 0xA8, 0x10, 0x0E), write 0xC0A8100E to IPADDRESS.

SUBNET MASK

Register Address	Register Name	Function	Read/Write Control
0x13	*SUBNETMASK*	IP subnet mask	Read/Write

Register Size	Register structure	Unit
4 Bytes	4 x Unsigned Bytes	none

Description:
Network IP subnet mask used for TCP/IP and UDP/IP.

Notes:
The module will change for a new subnet mask only when all of its communications ports are closed.
Do not forget to use a *SAVEUSERPARAMETERS* command.

If you do not want to use subnets, use the following subnet mask when IP address byte 0 is:
>0 and <=127 : 255.0.0.0 (Class A addresses)
>127 and <=191 : 255.255.0.0 (Class B addresses)
>191 and <=223 : 255.255.255.0 (Class C addresses)

Default value:
255.255.0.0

Example:
For the IP=10.2.6.45 and subnet mask = 255.255.0.0:
IP address class = A → netID = 10, subNetID = 2 and hostID = 6.45

TCP TIMEOUT

Register Address	Register Name	Function	Read/Write Control
0x14	*TCPTIMEOUT*	Timeout for TCP connection	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	sec

Description:
The TCP timeout is a value (in seconds) after which the user will be disconnected if the board has not been accessed in the meantime.

If the value is 0, the TCP timeout is deactivated. In this case however, if the client crashes during connection, the communication will never be closed on the module's side! Because a maximum of 4 communications are allowed at the same time on the module, one of them will be blocked. If the client crashes four times, all of the 4 communications will be blocked and the module will have to be reset!

The timeout for each TCP/IP connection is reloaded when there is traffic through the port.

Default value:
30

Limitations:
Max value: 255

MODULE NAME

Register Address	Register Name	Function	Read/Write Control
0x15	*MODULENAME*	Module's ASCII name	Read/Write

Register Size	Register structure	Unit
16 Bytes	16 (only) x Unsigned Bytes (CHAR)	none

Description:
Name and/or description of the module.

Example:
For the name "Hello Module"; extend to 16 byte the name: "Hello Module"+5x space=16 Byte.
So write 0x48656C6C 6F204D6F 64756C65 20202020.

UART CONFIG

Register Address	Register Name	Function	Read/Write Control
0x16	*UARTCONFIG*	UART baud rate and flow control configuration	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
UART baud rate and flow control configuration.

Bits 0-2	Baud rate configuration : 0 : 4800 bds 1 : 9600 bps (default) 2 : 19200 bps 3 : 38400 bps 4 : 57600 bps 5 : 115200 bps 6 : 115200 bps 7 : 115200 bps
Bits 3-6	Reserved
Bit 7	Hardware flow control bit (0 = disabled, 1 = enabled)

Default value:
1 (9600 bps)

I2C SPeeD

Register Address	Register Name	Function	Read/Write Control
0x18	*I2CSPD*	I2C speed setting	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
I2C speed setting between ~39kHz and 1MHz.

The value of this parameter must be computed with the following formula based on the wanted speed:

$$I2CSPD = \frac{10^7}{I2CSpeed_{wanted}} - 1$$

Note:
Speeds greater than 100kHz have some limitations (see page 21).

Default value:
99 (100kHz)

Examples:
Most common speeds:
100kHz : I2CSPD = 99
400kHz : I2CSPD = 24
1MHz : I2CSPD = 9

TCP CONNECTIONS OPENED

Register Address	Register Name	Function	Read/Write Control
0x1A	*TCPCONNECTIONSOPENED*	Number of TCP connections that are opened	Read only

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Number of users connected to the card using TCP.
Value can be 0 to 4.

IO DIRECTION 0

Register Address	Register Name	Function	Read/Write Control
0x20	*IODIRECTION0*	I/O ports directions	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the direction of each of I/O #1 to 8 (0 = output, 1 = input):

Bit 0	I/O #1
Bit 1	I/O #2
Bit 2	I/O #3
Bit 3	I/O #4
Bit 4	I/O #5
Bit 5	I/O #6
Bit 6	I/O #7
Bit 7	I/O #8

Default:
b'11111111' (all input)

Example:
b'00000111' → I/O number 1, 2 and 3 are set as input and others as outputs.

I/O VALUE 0

Register Address	Register Name	Function	Read/Write Control
0x21	*IOVALUE0*	I/O ports value	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the value of each of the I/O ports configured as output in *IODIRECTION0* register:

Bit 0	I/O #1
Bit 1	I/O #2
Bit 2	I/O #3
Bit 3	I/O #4
Bit 4	I/O #5
Bit 5	I/O #6
Bit 6	I/O #7
Bit 7	I/O #8

Example:
If *IODIRECTION0* is set to b'00000111' → I/O number 1, 2 and 3 are set as input and bits 0, 1 and 2 in *IOVALUE0* register will not be used. You can set the output state for the other ports in this register.

AD0 VALUE

Register Address	Register Name	Function	Read/Write Control
0x22	*AD0VALUE*	Last AD0 conversion result	Read only

Register Size	Register structure	Unit
2 Bytes	Unsigned Int 10 bits left aligned.	none

Description:
The 6 less significant bits are not used.
Last value converted by A/D port #0. The result is between 0 and 1023. 0 means GND and 1023 means Vdd.

Notes:
The A/D converter has a 10 bits resolution (which means that only the 10 most significant bits of the A/D VALUE register make sense, others should be ignored).

The acquisition is done every 50μs (250μs on FMod-TCP DB) and is asynchronous with the read access of the A/D VALUE register. When you access this register, you get the last A/D conversion result that can be up to 50μs (250μs) old.

The voltage reference for that acquisition is the processor input voltage, which should be a regulated 5V. The accuracy of the conversion depends on that voltage's stability. The lowest error (best case) is a half of the LSB of the 10bits results.

AD1 VALUE (FMod-TCP DB only)

Register Address	Register Name	Function	Read/Write Control
0x23	*AD1VALUE*	Last AD1 conversion result	Read only

Register Size	Register structure	Unit
2 Bytes	Unsigned Int 10 bits left aligned.	none

Description:
The 6 less significant bits are not used.
Last value converted by A/D port #1. The result is between 0 and 1023. 0 means GND and 1023 means Vdd.

Notes:
The A/D converter has a 10 bits resolution (which means that only the 10 most significant bits of the A/D VALUE register make sense, others should be ignored).

The acquisition is done every 250µs and is asynchronous with the read access of the A/D VALUE register. When you access this register, you get the last A/D conversion result that can be up to 250µs old.

The voltage reference for that acquisition is the processor input voltage, which should be a regulated 5V. The accuracy of the conversion depends on that voltage's stability. The lowest error (best case) is a half of the LSB of the 10bits result.

AD2 VALUE (FMod-TCP DB only)

Register Address	Register Name	Function	Read/Write Control
0x24	*AD2VALUE*	Last AD2 conversion result	Read only

Register Size	Register structure	Unit
2 Bytes	Unsigned Int 10 bits left aligned.	none

Description:
The 6 less significant bits are not used.
Last value converted by A/D port #2. The result is between 0 and 1023. 0 means GND and 1023 means Vdd.

Notes:
The A/D converter has a 10 bits resolution (which means that only the 10 most significant bits of the A/D VALUE register make sense, others should be ignored).

The acquisition is done every 250µs and is asynchronous with the read access of the A/D VALUE register. When you access this register, you get the last A/D conversion result that can be up to 250µs old.

The voltage reference for that acquisition is the processor input voltage, which should be a regulated 5V. The accuracy of the conversion depends on that voltage's stability. The lowest error (best case) is a half of the LSB of the 10bits result.

AD3 VALUE (FMod-TCP DB only)

Register Address	Register Name	Function	Read/Write Control
0x25	*AD3VALUE*	Last AD3 conversion result	Read only

Register Size	Register structure	Unit
2 Bytes	Unsigned Int 10 bits left aligned.	none

Description:
The 6 less significant bits are not used.
Last value converted by A/D port #3. The result is between 0 and 1023. 0 means GND and 1023 means Vdd.

Notes:
The A/D converter has a 10 bits resolution (which means that only the 10 most significant bits of the A/D VALUE register make sense, others should be ignored).

The acquisition is done every 250µs and is asynchronous with the read access of the A/D VALUE register. When you access this register, you get the last A/D conversion result that can be up to 250µs old.

The voltage reference for that acquisition is the processor input voltage, which should be a regulated 5V. The accuracy of the conversion depends on that voltage's stability. The lowest error (best case) is a half of the LSB of the 10bits result.

AD4 VALUE (FMod-TCP DB only)

Register Address	Register Name	Function	Read/Write Control
0x26	*AD4VALUE*	Last AD4 conversion result	Read only

Register Size	Register structure	Unit
2 Bytes	Unsigned Int 10 bits left aligned.	none

Description:
The 6 less significant bits are not used.
Last value converted by A/D port #4. The result is between 0 and 1023. 0 means GND and 1023 means Vdd.

Notes:
The A/D converter has a 10 bits resolution (which means that only the 10 most significant bits of the A/D VALUE register make sense, others should be ignored).

The acquisition is done every 250µs and is asynchronous with the read access of the A/D VALUE register. When you access this register, you get the last A/D conversion result that can be up to 250µs old.

The voltage reference for that acquisition is the processor input voltage, which should be a regulated 5V. The accuracy of the conversion depends on that voltage's stability. The lowest error (best case) is a half of the LSB of the 10bits result.

IO DIRECTION 1

Register Address	Register Name	Function	Read/Write Control
0x27	*IODIRECTION1*	I/O ports directions	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the direction of each of I/O #9 to 16 (0 = output, 1 = input):

Bit 0	I/O #9 (FMod-TCP DB only)
Bit 1	I/O #10 (FMod-TCP DB only)
Bit 2	I/O #11 (FMod-TCP DB only)
Bit 3	I/O #12 (FMod-TCP DB only)
Bit 4	I/O #13
Bit 5	I/O #14
Bit 6	I/O #15
Bit 7	I/O #16 (FMod-TCP DB only)

Default value:
b'11111111' (all inputs)

Example:
b'00000111' → I/O number 9, 10 and 11 are set as inputs and others as outputs.

I/O VALUE 1

Register Address	Register Name	Function	Read/Write Control
0x28	*I/OVALUE1*	I/O ports value	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the value of each of the 8 I/O ports configured as output in *IODIRECTION1* register:

Bit 0	I/O #9 (FMod-TCP DB only)
Bit 1	I/O #10 (FMod-TCP DB only)
Bit 2	I/O #11 (FMod-TCP DB only)
Bit 3	I/O #12 (FMod-TCP DB only)
Bit 4	I/O #13
Bit 5	I/O #14
Bit 6	I/O #15
Bit 7	I/O #16 (FMod-TCP DB only)

Example:
If *IODIRECTION1* is set to b'00000111' → I/O number 9, 10 and 11 are set as input and bits 0, 1 and 2 in *IOVALUE1* register will not be used. The other I/Os' output states can be set in the register.

IO DIRECTION 2

Register Address	Register Name	Function	Read/Write Control
0x29	*IODIRECTION2*	I/O ports directions	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the direction of each of I/O #17 to 19 (0 = output, 1 = input):

Bit 0	I/O #17 (FMod-TCP DB only)
Bit 1	I/O #18 (FMod-TCP DB only)
Bit 2	I/O #19 (FMod-TCP DB only)
Bit 3	Reserved
Bit 4	Reserved
Bit 5	Reserved
Bit 6	Reserved
Bit 7	Reserved

Default value:
b'11111111' (all input)

Example:
b'00000101' → I/O number 17 and 19 are set as inputs and I/O number 18 as output.

I/O VALUE 2

Register Address	Register Name	Function	Read/Write Control
0x2A	*IOVALUE2*	I/O ports value	Read/Write

Register Size	Register structure	Unit
1 Byte	Unsigned Int 8 bits	none

Description:
Controls the value of each of the 8 I/O ports configured as output in *IODIRECTION2* register:

Bit 0	I/O #17 (FMod-TCP DB only)
Bit 1	I/O #18 (FMod-TCP DB only)
Bit 2	I/O #19 (FMod-TCP DB only)
Bit 3	Reserved
Bit 4	Reserved
Bit 5	Reserved
Bit 6	Reserved
Bit 7	Reserved

Example:
If *IODIRECTION2* is set to b'00000111' → I/O number 17 and 19 are set as inputs and bits 0 and 2 in *IOVALUE2* register will not be used. For I/O number 18, you can set its output state in this register.

Contact address :

FiveCo - Innovative Engineering
PSE-C
CH-1015 Lausanne
Switzerland
Tel: +41 21 693 86 71
Fax: +41 21 693 86 70

www.fiveco.ch
info@fiveco.ch

Updates, Support and further Information available on the following web page:

http://www.fiveco.com/section_engineering/products/tcpip/real_tcp_support_E.htm