

Programming the ROBO TX Controller

Part 1: PC programming

"PC_Programming_RoboTXC"
package and documentation

V1.5 dated 4/24/2012

References:

Description	Version	Date
ROBO TX Controller firmware	1.30	3/19/2012
ftMscLib.dll library	1.5.11	2/19/2012
ROBO Pro	3.1.3	3/26/2012

MSC Vertriebs GmbH
Design Center Aachen
Pascalstr. 21
52076 Aachen, Germany

Contents

1	Introduction.....	3
1.1	System requirements	3
1.2	Terminology	3
2	Software structure on the PC	4
2.1	Low-level API (COM port level)	5
2.2	High-level API (library level)	5
3	Installing drivers on the PC	6
3.1	Installing the USB driver	6
3.2	Installing Bluetooth	6
3.3	Initial USB and Bluetooth connection test	6
4	Testing using RoboTxTest.exe	8
4.1	Installation	8
4.2	Connecting to the ROBO TX Controller	8
4.3	Diagnostics and testing using the ROBO TX Controller	9
4.4	File upload and remote shell	13
4.5	Firmware Update	15
5	Description of the ftMscLib.dll library	16
6	Bluetooth Messaging API	17
6.1	Network topology, active and passive connections	17
6.2	Bluetooth messaging in online mode	18
7	I ² C interface	20
7.1	I ² C hardware connection	20
8	Demo applications written in C	21
9	Demo applications written in C++	22
9.1	Example: two controllers and Bluetooth messaging	23
10	Demo applications written in C#	25
11	Multi-controller operation.....	26
12	Updating the ROBO TX Controller firmware.....	27
13	Low-level API (COM port level)	28
13.1	Fish.X1 interface protocol	28
13.2	Remote shell.....	30
13.3	Xmodem file transfer.....	31
14	Document change history	33

1 Introduction

This document describes how to program the fischertechnik ROBO TX Controller from your PC and is intended primarily for PC programmers who do not work with the ROBO Pro fischertechnik programming interface, but want to create their own PC programs for the ROBO TX Controller instead.

USB or Bluetooth are the available interfaces for the PC. In both cases, a virtual COM port is available on the PC for communication. The USB interface represents the USB standard *Communication Device Class (CDC)*. The Bluetooth interface uses the *Serial Port Profile (SPP)*.

1.1 System requirements

Programming the ROBO TX Controller, as described in this document and insofar as it is supported by the supplied library, requires a Microsoft Windows operating system (2000, XP, Vista). Standard Microsoft development tools such as Microsoft Visual C++ or Microsoft Visual Studio were also used to create the supplied demo projects.

Other than programming under Microsoft Windows, the ROBO TX Controller can basically also be programmed using other computer operating systems that support the standard interfaces used (USB CDC, Bluetooth SPP).

A full-speed USB host port on the PC and a Bluetooth interface (integrated in the PC or via an external adapter, such as a Bluetooth USB stick) are also required.

1.2 Terminology

ROBO TX Controller: the name of the fischertechnik robotics controller itself. It is also abbreviated as **TX-C**.

Firmware: the software on the ROBO TX Controller and functions as the operating system, device driver, logger and boot loader.

Robotics program: the software component that can run as a loadable, executable robotics application on the ROBO TX Controller. An existing executable firmware on the TX-C is of course required.

2 Software structure on the PC

The ROBO TX Controller features two physical interfaces for connection to the PC: USB and Bluetooth. In both cases, COM port drivers are available and can be used as standard serial interfaces (RS-232) on the PC software side. However, only one of the two interfaces, USB or Bluetooth, can be active at any point in time.

The low-level interface is the particular COM port itself. When using the ftMscLib.dll library, an alternative high-level API is available that reproduces the functionality of the ROBO TX Controller in convenient function calls. The following figure illustrates this once again graphically:

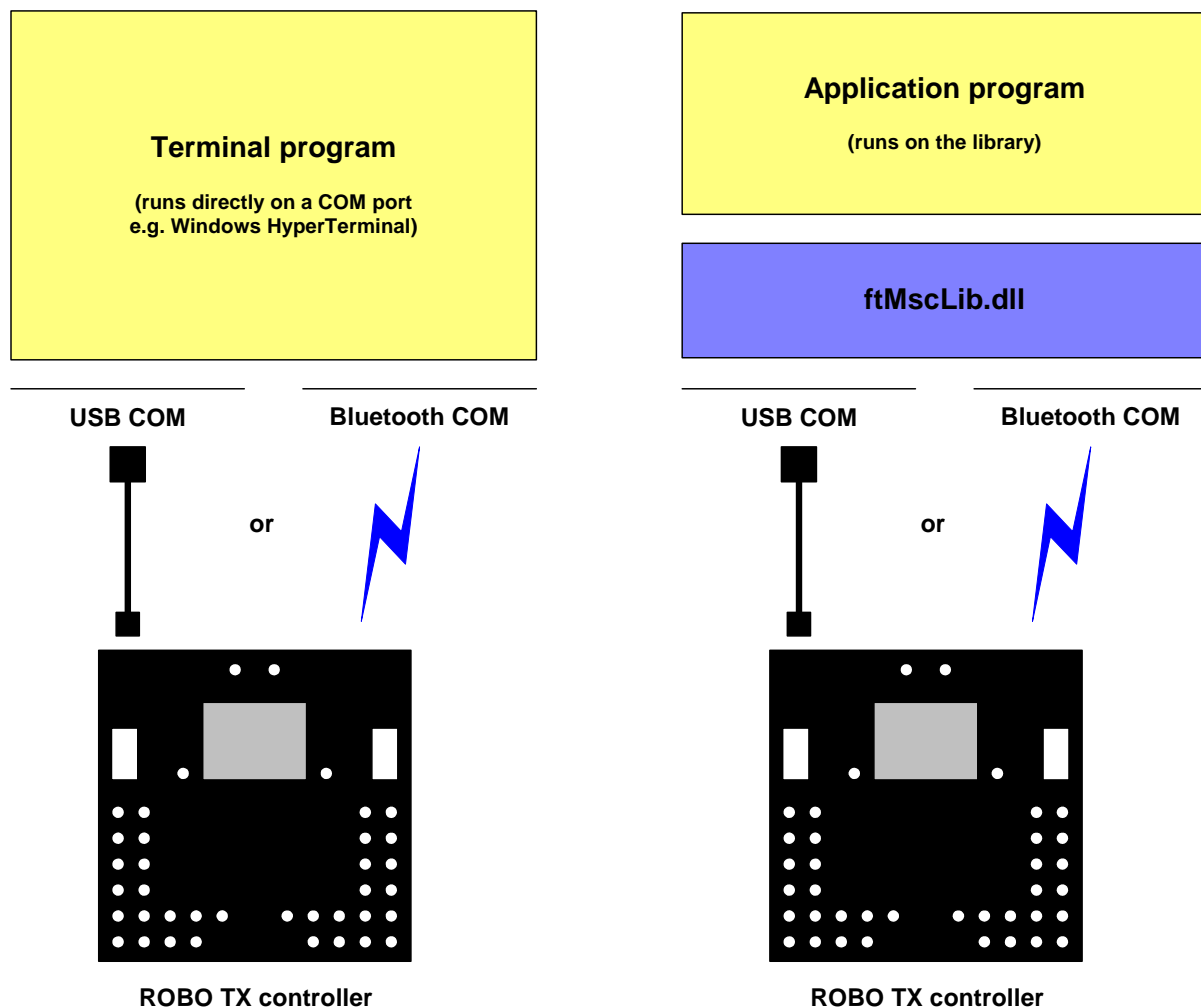


Fig. 1: Interfaces from the ROBO TX Controller to the PC

2.1 Low-level API (COM port level)

The low-level API is the COM port itself. In this case, the ftMscLib.dll library is not used, but directly accessed on the ROBO TX Controller via the COM port. There are three sub-interfaces:

- Remote shell
- File transfer interface (Xmodem)
- X1 protocol interface

The firmware on the ROBO TX Controller automatically detects which interface is being used. If the incoming data have a particular structure (Xmodem or X1), then the relevant service is used. All other incoming data are interpreted as remote shell commands. The remote shell and file transfer interface can be used with a standard terminal program such as Windows HyperTerminal.

A more detailed description of the different low-level API sub-interfaces can be found in chapter 13.

2.2 High-level API (library level)

The high-level API is represented by the ftMscLib.dll library functions. The latter uses the different sub-interfaces of the low-level API internally to facilitate programming of the ROBO TX Controller in compiled function calls. The relatively complex X1 protocol in particular disappears under an essentially simpler interface.

A description of the library (i.e. the high-level API) can be found in chapter 5. More detailed information on all available API functions can be found in the separate **Windows_Library_ftMscLib.pdf** document.

3 Installing drivers on the PC

3.1 Installing the USB driver

The board currently running is connected to the PC using a USB cable (if necessary, disconnect and then reconnect the USB cable). Windows then reports the presence of a new USB device and searches for a driver. The installation wizard launches automatically. When prompted for "Use Windows Update", choose "No, not this time". On the next screen, choose "Install from a list or specific location " and point to the supplied file `\USB_Driver\ROBO_TX_Controller.inf`. A new COM port device should then appear in the Device Manager with the description "fischertechnik USB ROBO TX Controller". It is recommended that you make note of the COM port number that is automatically assigned to it by Windows (see Device Manager).

3.2 Installing Bluetooth

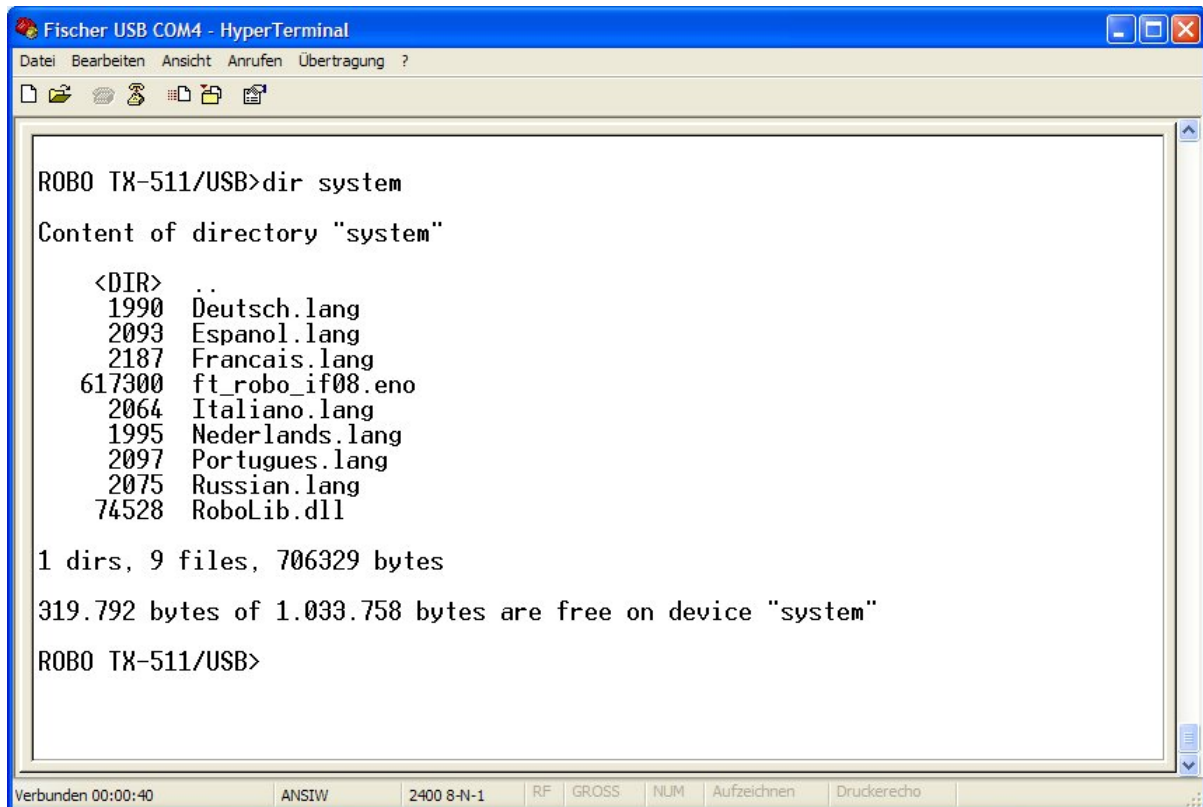
The ROBO TX Controller currently running can be detected by the PC Bluetooth service at any time as long as this property ("Bluetooth device discoverable") is not switched off in the relevant settings. The controller appears in the search window on the PC with its host name, which is shown on the display (e.g. "ROBO TX-511"). A connection to the PC is made by using the main PIN "1234". From this moment on, a fixed COM port is assigned to the ROBO TX Controller and the Bluetooth connection is made automatically from the PC as soon as the relevant COM port is accessed.

The procedure for installation and PC connection with the TX-C may differ depending on the Bluetooth protocol stack and the Bluetooth hardware used on the PC. In any case, follow the installation instructions of the particular manufacturer.

3.3 Initial USB and Bluetooth connection test

An initial USB or Bluetooth communication test with the board can be carried out using a simple terminal program (e.g. HyperTerminal, ZOC or TeraTerm). Select the right COM port number for the USB or Bluetooth driver and set any baud rate desired (does not matter).

If the connection could be made, you will see in the terminal program the monitor console of the 4NetOS operating system, which is running on the ROBO TX Controller. A command line prompt should now appear each time <ENTER> is pressed. By entering "dir", you can continue to view the contents of the "system", "flash" or "ramdisk" virtual disks:



```
Fischer USB COM4 - HyperTerminal
Datei Bearbeiten Ansicht Anrufen Übertragung ?

ROBO TX-511/USB>dir system
Content of directory "system"

<DIR>
1990 Deutsch.lang
2093 Espanol.lang
2187 Francais.lang
617300 ft_robo_if08.eno
2064 Italiano.lang
1995 Nederlands.lang
2097 Portugues.lang
2075 Russian.lang
74528 RoboLib.dll

1 dirs, 9 files, 706329 bytes
319.792 bytes of 1.033.758 bytes are free on device "system"
ROBO TX-511/USB>
```

Verbunden 00:00:40 ANSIW 2400 8-N-1 RF GROSS NUM Aufzeichnen Druckerecho

You can view additional commands by entering "?" or "help". More detailed descriptions are available in section 13.2.

4 Testing using RoboTxTest.exe

4.1 Installation

The Windows-based program **RoboTxTest.exe** is a simple program you can use to diagnose and test the functionality of the ROBO TX Controller. The communication of **RoboTxTest** with the controller is based on the same **ftMscLib.dll** PC library that is used by the ROBO Pro GUI. The integrated test window known as "Interface Test" in ROBO Pro also offers similar functionality with a narrower scope. In the case of **RoboTxTest** in particular, it is important to mention the remote shell interface implemented for the ROBO TX Controller, which is not offered by the ROBO Pro software.

A special installation for the diagnostics tool and library is not required if all files from the CD have already been copied to the computer. The **RoboTxTest.exe** executable and the associated **ftMscLib.dll** just have to be placed together in the same folder (in this case: **\RoboTxTest**). The **RoboTxTest.exe** diagnostics tool (not the DLL) was created as a .NET program and therefore requires that the installation of the .NET Framework version 2.0 or later on the relevant Windows PC.

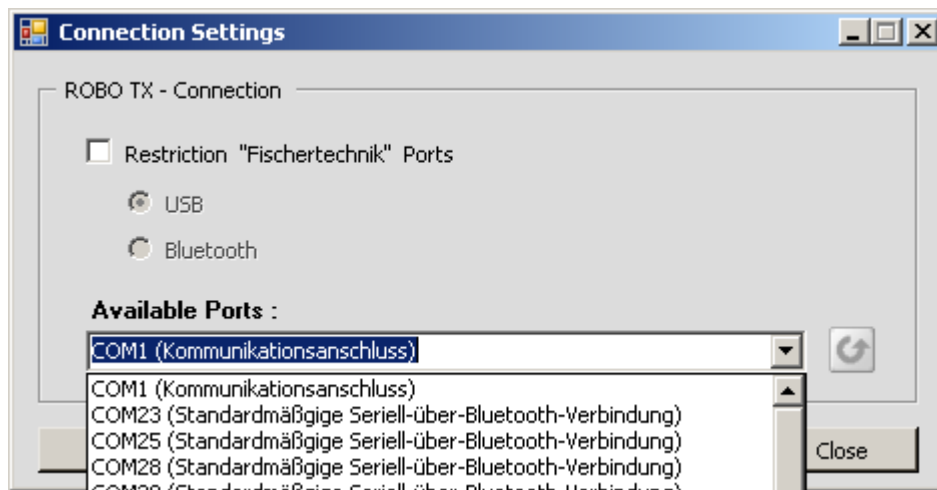
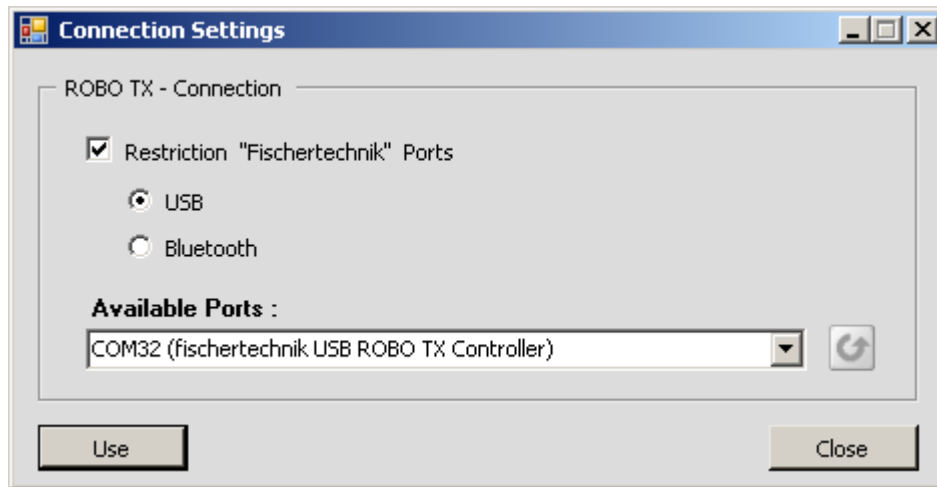
If the .NET environment is not yet installed on your computer, installation of the environment is offered automatically the first time you run **RoboTxTest.exe**. An Internet connection is required for the installation.

Every time the program is called, **RoboTxTest.exe** creates a log file called **ftlib.log** which can be used to diagnose errors if something goes wrong right away. Any existing log file is renamed to **ftLib.bak**.

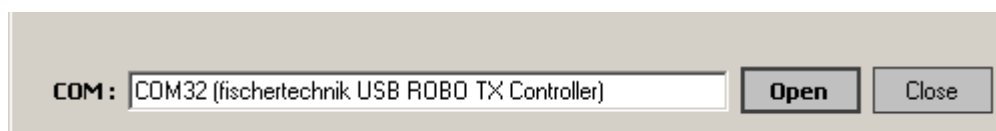
4.2 Connecting to the ROBO TX Controller

You can use the **RoboTxTest.exe** program to open a connection to the ROBO TX Controller via a (virtual) COM port to be selected (the COM port number is not restricted), which in turn uses the USB or Bluetooth interface via the relevant Windows device driver. You can select the COM port either by using the **[Connection]** menu on the menu bar or by clicking on the COM port description field at the bottom. A new window will then open, which offers a choice of available COM ports once the type of connection is set. If the 'Fischertechnik Ports' restriction has been removed, all available COM ports will be listed. The list of COM ports will then be generated automatically according to the selection criteria and will be shown if you click on the button.

Selecting a COM port:



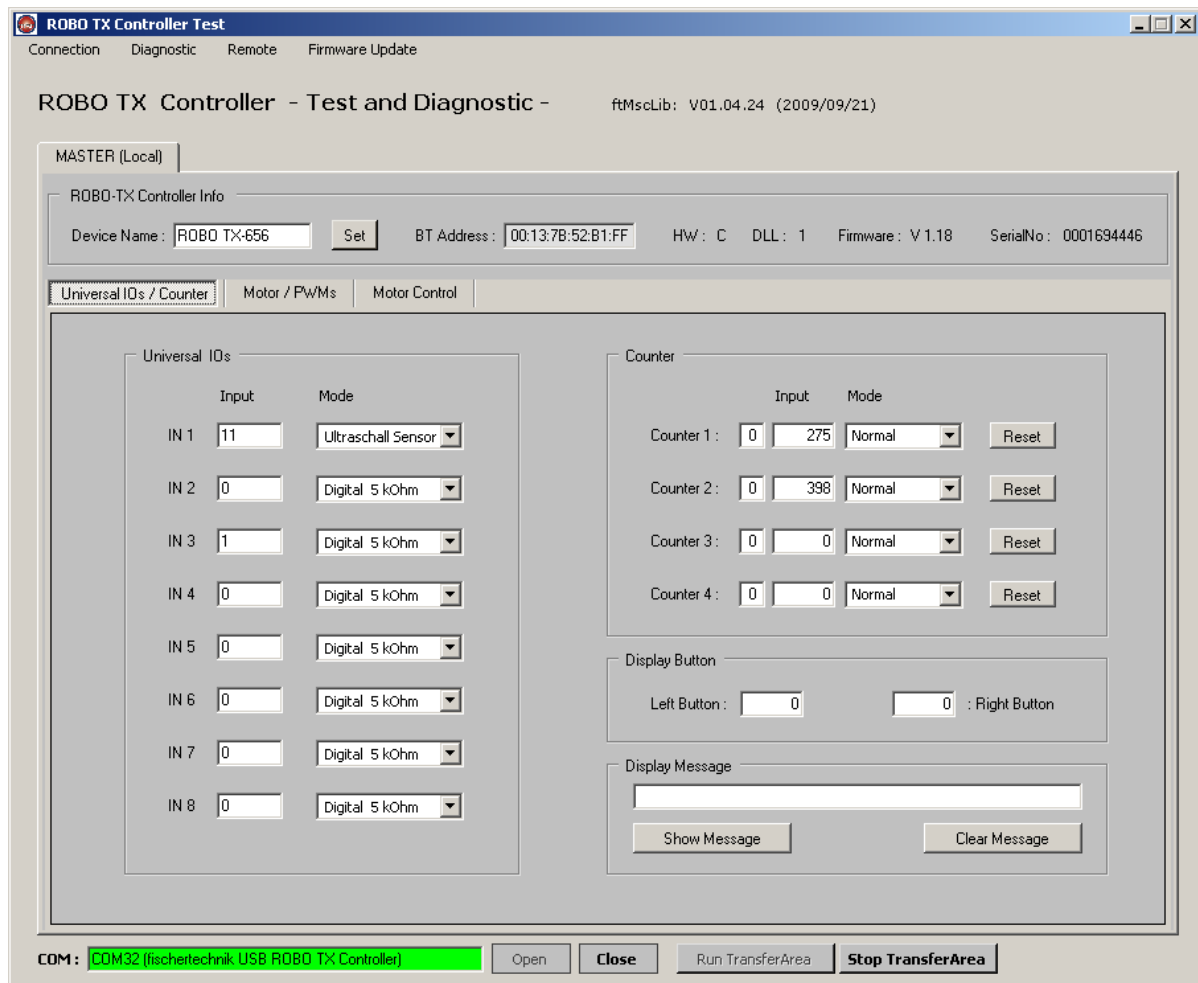
After selecting a COM port and confirming with the **[Use]** button, the selected port is displayed in the text box of the main window and can then be opened. Clicking the **[Close]** button closes the connection again.



4.3 Diagnostics and testing using the ROBO TX Controller

After connecting successfully, the text box turns green. The ROBO TX Controller's key information is read out and output in the upper section of the 'ROBO-TX Controller Info'

screen. In addition, the relevant number of connected slave interfaces (extension boards), which is obtained from the information read out, is displayed on another tab. If no slaves are connected, only the local interface (Master) appears as a tab. If you are running multiple ROBO TX Controllers, please also read chapter 11.



Clicking the **[Run TransferArea]** button starts communication with the ROBO TX Controller and the current values are displayed.

The following information or controller statuses are displayed on the **'Universal IOs / Counter'** tab:

- Values that are present at the eight universal inputs (IN1 through IN8) according to the input configuration
- Current counter status of the four counter inputs (C1 through C4)
- Time measurement when the two selection push-button switches are pressed on the controller (ms)

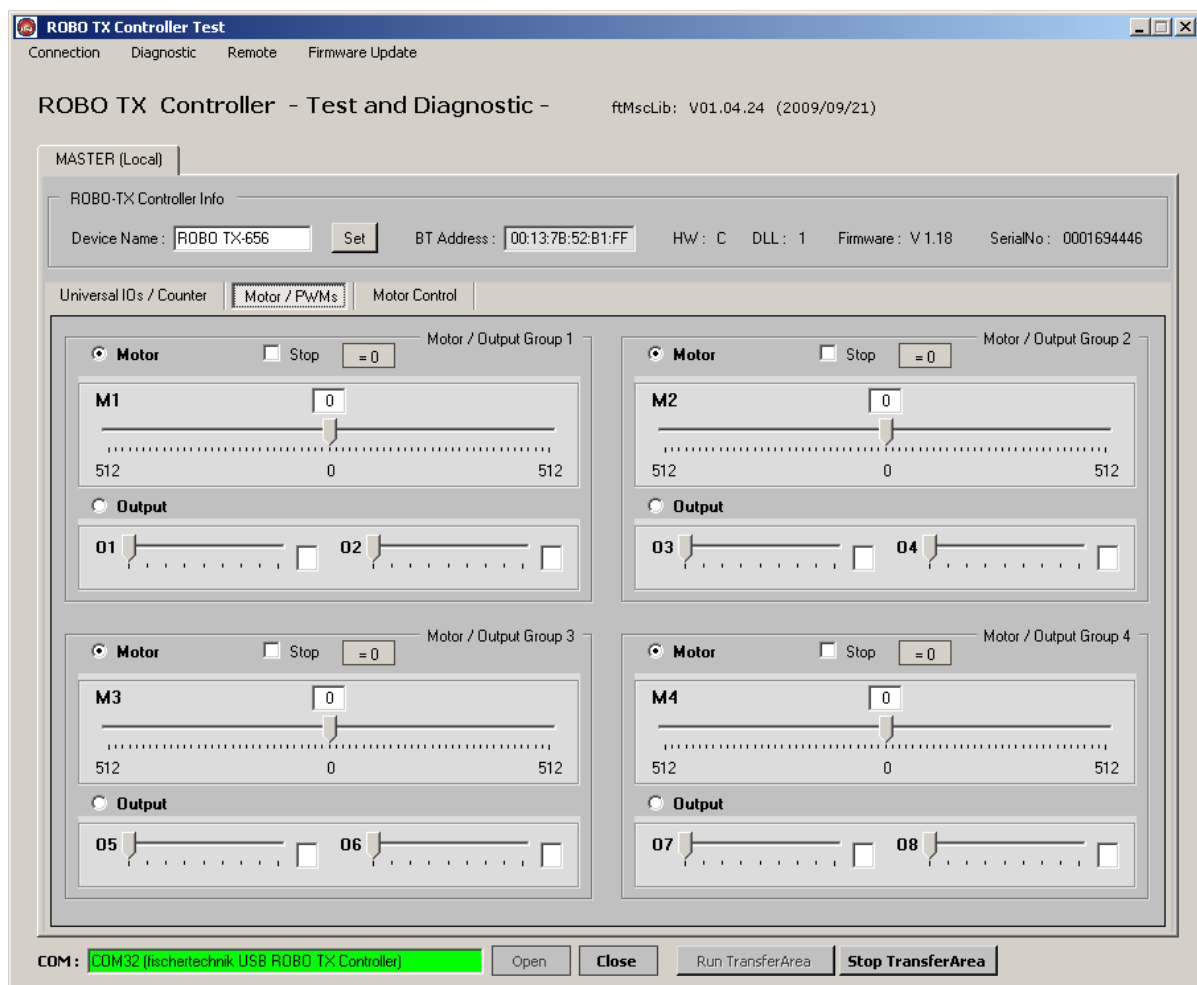
Displaying the universal inputs: The existing values can be influenced by selecting the relevant display mode. The values of a voltage or resistance measurement are displayed separately for each universal input. The output can be optionally displayed digitally (1/0). If a value overrun is present at a universal input, the string *"overrun"* appears in the output field.

Displaying the counter inputs: The current counter status appears in the fields for the four digital counter inputs (counter 1 through 4). The counter can be reset by clicking the **[Reset]** button. A corresponding reset command is sent to the ROBO TX Controller.

Displaying the time measurement of depressed selection push-button switches: For the two selection push-button switches on the ROBO TX Controller, the time is output here as long as these push-button switches remain depressed on the controller. The time displayed is output in milliseconds (ms).

A message with a maximum of 64 characters can be output on the ROBO TX Controller display using the **'Display Message'** option. To do this, edit a message in the line provided and click the **[Show Message]** button. The message is sent to the controller and shown on the display. Multiple messages can be sent consecutively, but only the last message sent will be displayed. By pressing the right selection push-button switch, the most recent message is deleted. However, the **[Clear Message]** button deletes all messages on the controller.

If you select the **'Motor / PWMs'** view, you will have the following display options:

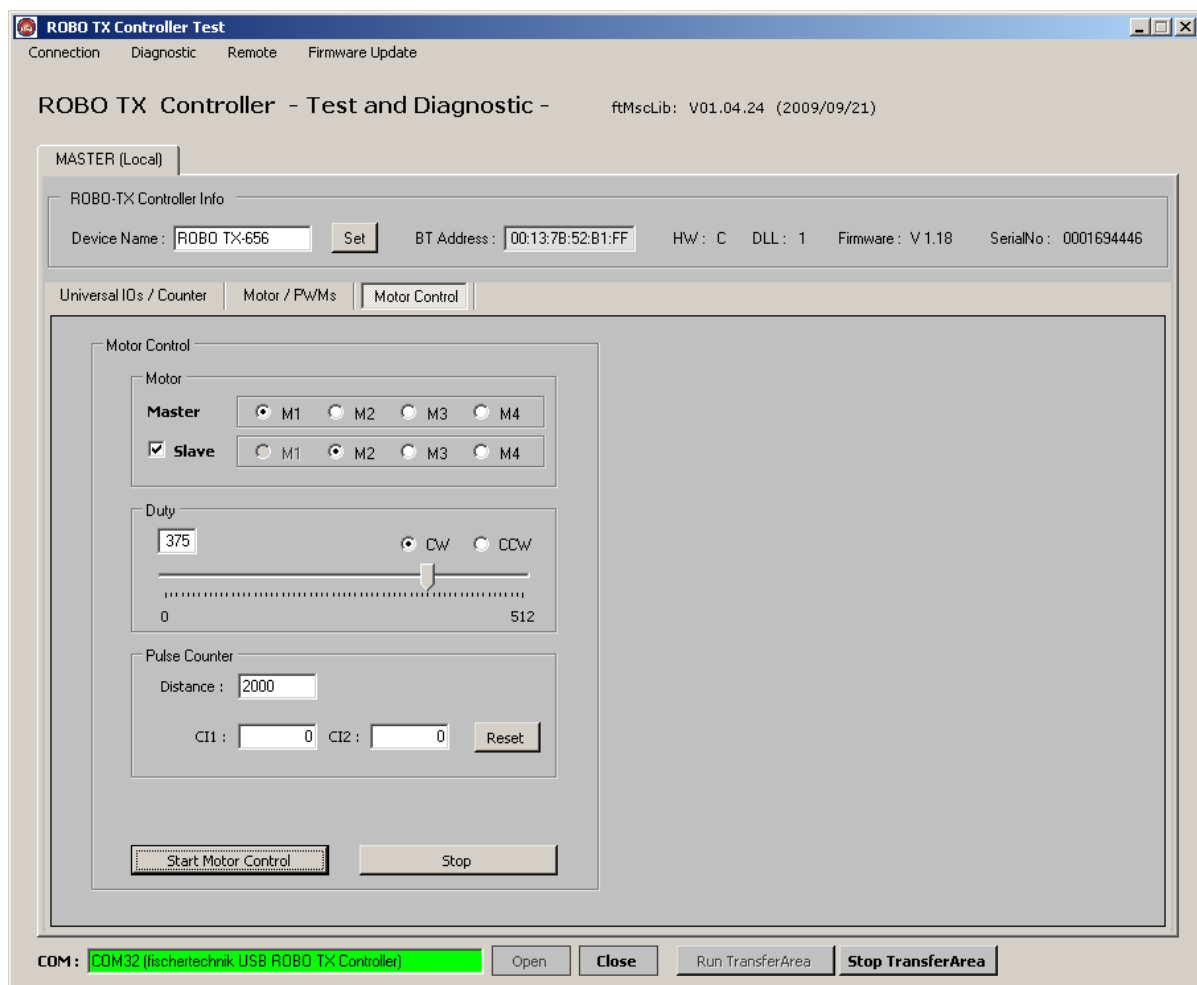


Here you can control the four motors or the eight PWM outputs individually. Either a motor or the corresponding pair of PWM outputs is controlled by independent sliders.

Motor Control: For the four motors M1 through M4, the direction of rotation and the motor speed can be controlled by moving the slider left or right. The speed can be set in fine PWM increments from 0 to 512. The [**Stop**] option allows a motor to stop momentarily while maintaining the speed setting. The "=0" option moves the motor temporarily and precisely resets it to the speed of 0 (standstill).

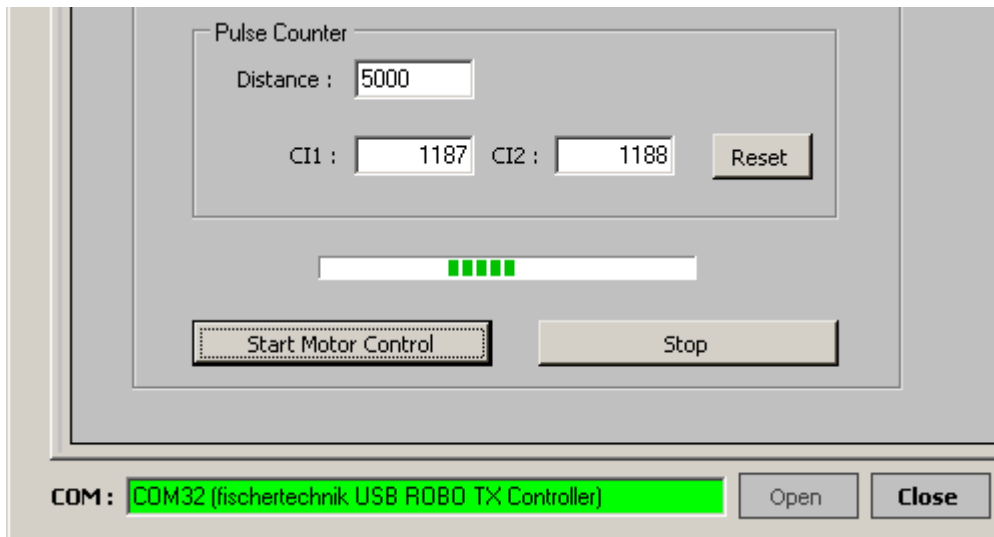
If you switch from the '**Motor**' option to the '**Output**' option, a change in the configuration is automatically sent to the interface, which deactivates the motors, thus treating the two outputs of a motor group as independent outputs. The particular PWM outputs of a motor group can now be controlled in increments of 0 to 8 using the separate sliders. The slider for the motor of a group is thus disabled.

Synchronization of the motors can be tested and carried out via the following settings on the '**Motor Control**' tab.



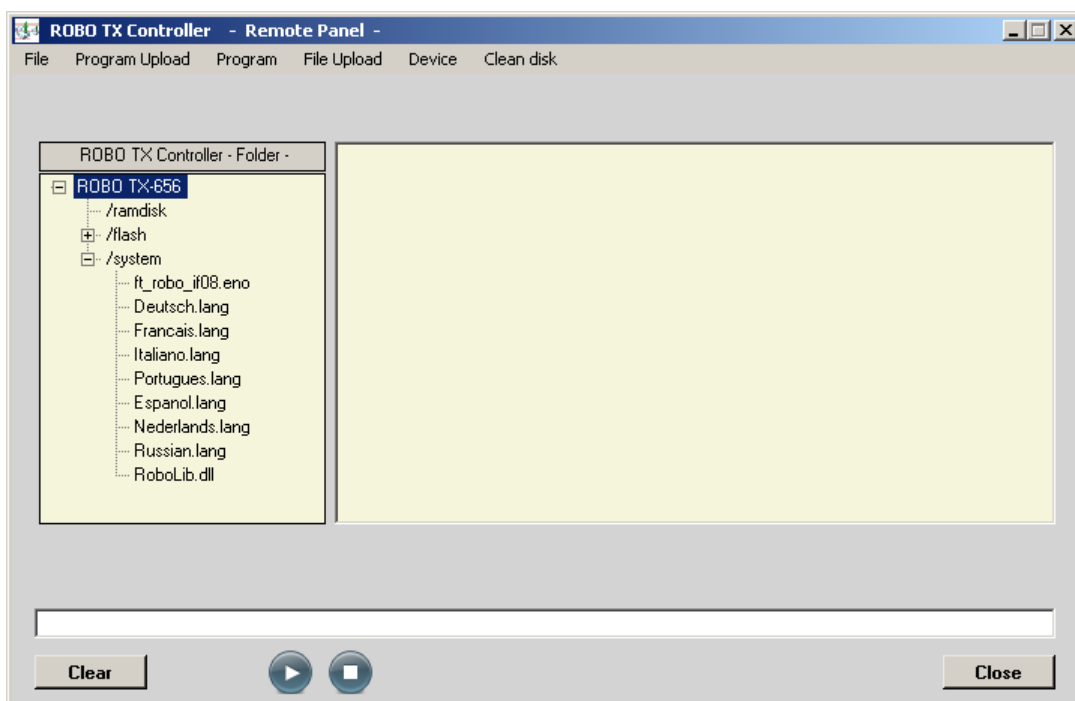
First select a master motor and assign a slave motor to it (optional). Within the '**Duty**' setting you can set the speed to be driven as well as the direction of rotation of the motors. By entering the distance, the application is told when the target of the motor synchronization process is reached. The two counter inputs record the momentary counter status during synchronization. The [**Reset**] button is used to reset these counters even while motor synchronization is taking place. The target is then of course reached a bit later. The [**Start Motor Control**] button starts active motor synchronization, the [**Stop**] button stops

synchronization temporarily until it is activated again by pressing [**Start Motor Control**]. Starting the synchronization process always causes the counter statuses to be reset. If the motor(s) has reached the predefined target, the motor(s) is stopped momentarily. During synchronization a looping progress bar is displayed which symbolizes the active synchronization process.



4.4 File upload and remote shell

You can access the following additional window via the [**Remote**] menu:

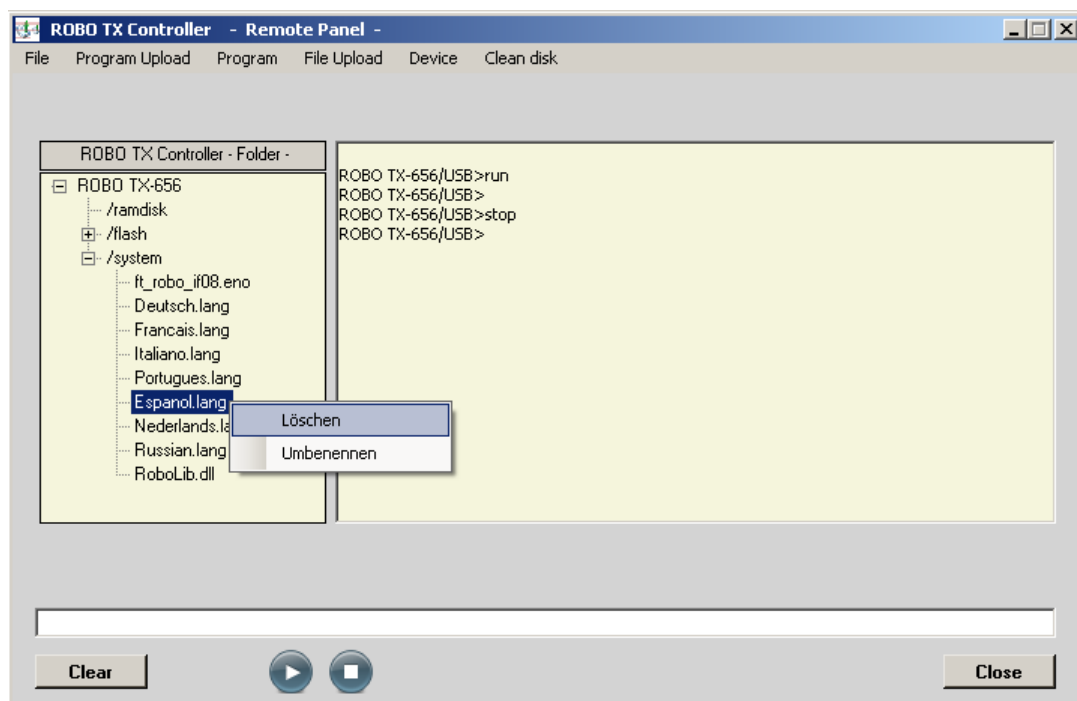


The window is divided into a 'ROBO TX Controller Folder' section and a section for console output (remote shell). In the area on the left, the contents of the available disks '/ramdisk', '/flash' and '/system' are output in a hierarchical structure. The disks can be opened by clicking on the "+" sign and the files currently stored there are displayed.

The **[File]** menu selection can be used to select any file that is to be loaded to the interface. The selection of a directory is stored in the "robotest.ini" INI file and is available again the next time the call is made. After selecting a file, the file information is output in the console window. The **[File Upload]** menu is used to activate the uploading of the selected file to the ROBO TX Controller file system. All available disks are available as the "target disk". Only the flash disk stores files permanently so that they are available again the next time the controller is switched on.

Only executable programs can be loaded to the relevant target disk using the **[Program Upload]** menu. Programs can be started or stopped using the commands **[Run]** and **[Stop]** from the input line or via the **[Program]** menu.

Files on the disk can either be deleted via the console command 'del' or by clicking the **"Delete"** context menu (right-click with mouse) on the file name in the left window. A file can also be renamed using the context menu.



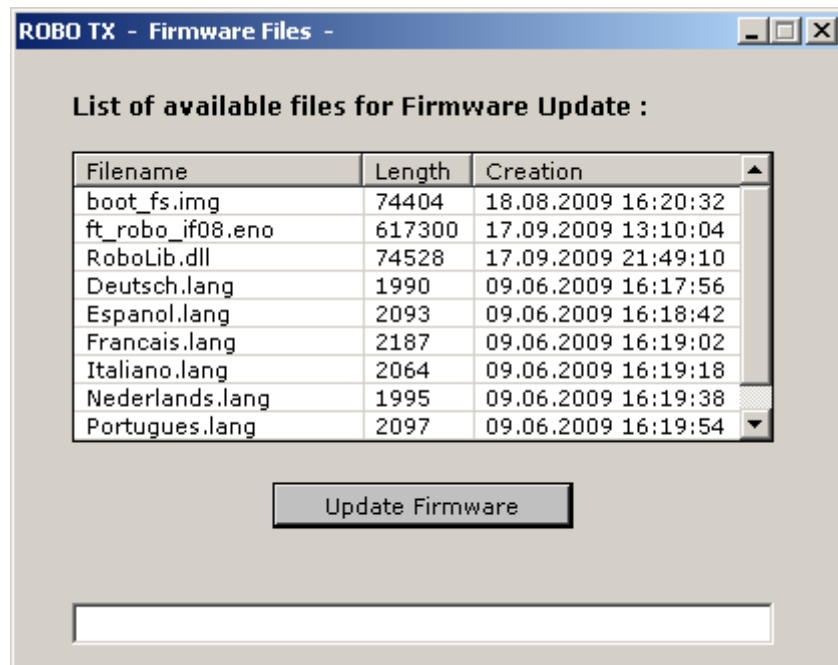
The contents of the two disks /ramdisk and /flash can be deleted completely using the **[Clean disk]** menu. Entering the corresponding console command "**clean_disk /ramdisk**" or "**clean_disk /flash**" in the command line carries out the same action.

The bottom input line is used to input console commands. The ROBO TX Controller output on the command set is output in the upper right window. You can output all available console commands of the ROBO TX Controller using the "?" command. The **[Clear]** button deletes the contents of the output window on the right.

The "Remote Panel" console corresponds to what you would receive or see using a terminal program (e.g. Windows HyperTerminal) when directly accessing the ROBO TX Controller via a COM port.

4.5 Firmware Update

The [Firmware Update] menu is used to update the firmware, including the required language files, from any directory. After selecting a directory containing the necessary files for the software update (in this case: subfolder /**Firmware**), the following window appears with a list of files available for the firmware update.



Clicking on the [Update Firmware] button starts the firmware update. The progress bar that then appears shows the progression of time for the update. The steps, or commands, executed during the firmware update are output in the bottom output line. The update may occasionally appear to halt briefly. Then the commands '**flush_disk flash**' and '**flush_disk system**' are carried out on the ROBO TX Controller and take a while to complete. If the firmware updates successfully, the message 'Firmware Update finished...' appears in the bottom output line. The window can then be closed again and the firmware version previously transmitted should be on the ROBO TX Controller after it is switched off and then back on.

5 Description of the ftMscLib.dll library

Like the "FtLib" library of the ROBO interface (by Knobloch), the Windows "ftMscLib" library supports online mode communication with the new ROBO TX Controller.

The closest equivalent function calls are defined to minimize the cost of implementing the new library. The error codes from the previous ftLib.h file were used as possible return codes and have been expanded to include new ones.

The Windows library was created using Microsoft Visual C++ Studio 6.0 and is available in the following versions:

Static library (WIN32 library .lib)

- | | |
|--------------------------------------|--|
| - FtMscLib_Static_LIBCMT_Release.lib | Multithreaded (/MT link option) |
| - FtMscLib_Static_LIBCMTD_Debug.lib | Debug Multithreaded (/MTd link option) |

Dynamic library (.dll):

- | | |
|--|--|
| - ftMscLib.dll, ftMscLib.lib | Multithreaded DLL (/MD link option) |
| - ftMscLib_debug.dll, ftMscLib_debug.lib | Debug Multithreaded DLL (/MDd link option) |

For the multimedia timer calls within the library, the WIN32 library module (Winmm.lib) has been linked.

A more detailed description of the library and a complete list of all library calls (high-level API) can be found in the `Windows_Library_ftMscLib.pdf` document.

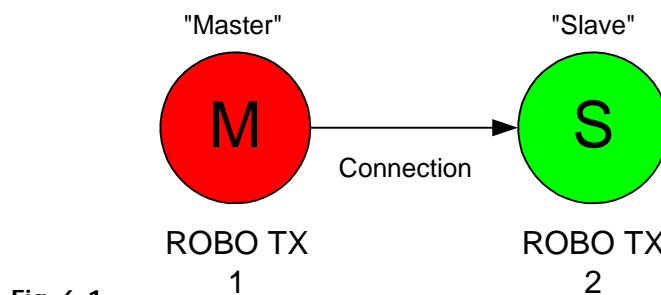
6 Bluetooth Messaging API

In addition to the option of a PC host connection via Bluetooth, the ROBO TX Controllers can also exchange short text messages controlled by the program via Bluetooth. Actions can be triggered from one controller to another or even programs on multiple controllers can be synchronized remotely during operation.

To facilitate this are a number of API functions that control the connection (active or passive) from one controller to another (or multiple) controllers and make it possible to send and receive messages.

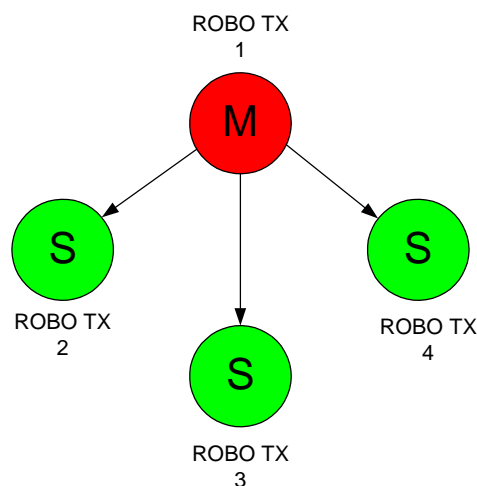
6.1 Network topology, active and passive connections

If, for instance, two ROBO TX Controllers are to exchange messages with each other, one is always the MASTER and the other is the SLAVE. The MASTER is the one that initiates the connection (active, connect) and the SLAVE is the one that waits to be connected by the MASTER (passive, listen):



If a connection is terminated, for instance because the two controllers are each on moving models and the models have been moved too far away from each other (out of remote range if there are more than approx. 10 meters between them), it is again the responsibility of the MASTER to re-establish the connection. The SLAVE, which registers the terminated connection, does nothing else except to wait for the MASTER to reconnect.

If there are more than two controllers in the Bluetooth network, there is still only one MASTER and the rest are SLAVES:



In this example, the transmission of the message from ROBO TX 2 to ROBO TX 3 is routed via the MASTER (ROBO TX 1). This is taken care of in the application program.

6.2 Bluetooth messaging in online mode

If you want to run Bluetooth messaging in online mode, there are a few things to be aware of. The connection from the host PC to the Bluetooth messaging MASTER should not also simultaneously be a Bluetooth connection, since the host PC will otherwise attempt to take over the role of the MASTER. The controller with the role of the MASTER should always be connected by the host PC via a USB connection. A connection from the host PC to the controller with the role of SLAVE for Bluetooth messaging can, however, be made without difficulty because in this case no change in roles is anticipated. A SLAVE can also accept connections from different MASTERS as long as the SLAVE itself does not attempt to set up Bluetooth connections (which would turn it into the MASTER).

Below are a few examples:

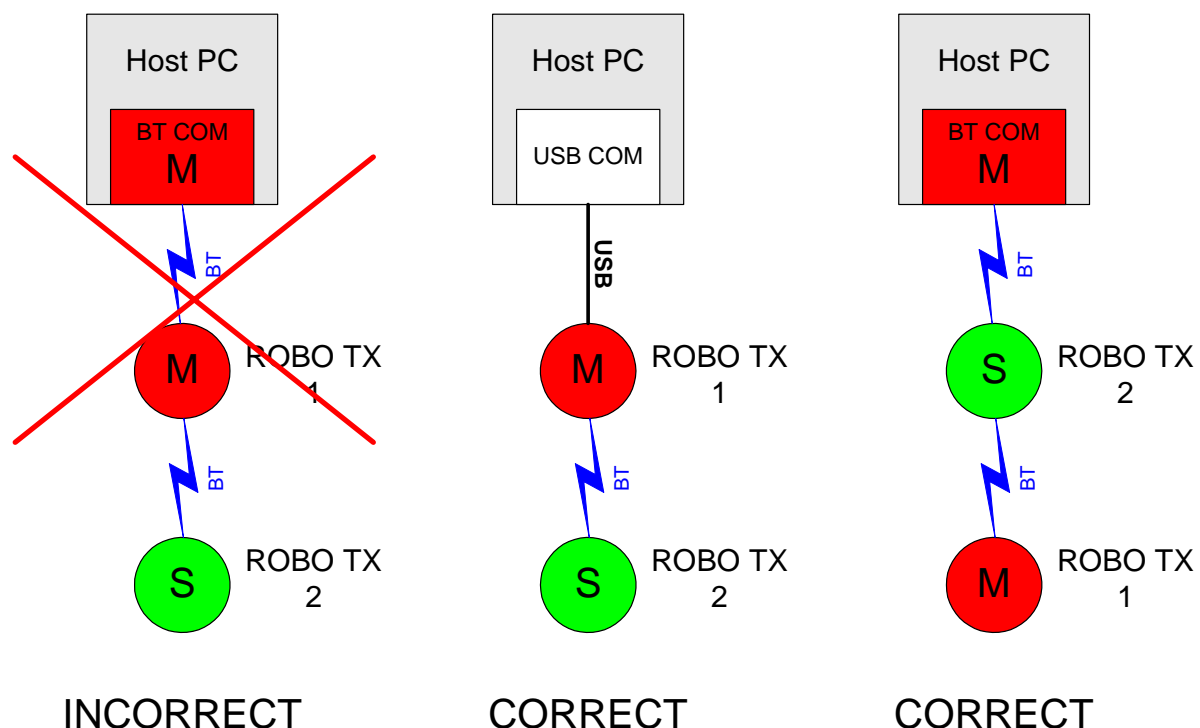


Fig. 6-3

This connection limitation is from the Bluetooth *Piconet* specification, which is defined as a topology with one MASTER and 1-n SLAVES. On the other hand, a network connection in which multiple *piconets* are linked to each other is called a *scatternet*. In this case, some participants are the MASTER and the SLAVE at the same time. However, this is not supported by the ROBO TX Controller firmware.

IMPORTANT NOTE

If inadvertently at some point the master changes or a controller takes on a dual role (MASTER and SLAVE simultaneously), the connection can by all means still be made. However, it is highly likely that there will be problems sending and receiving messages. Some transmission directions will not work.

If you want to run multiple ROBO TX Controllers simultaneously from one PC in online mode, the following setup can be considered, while still maintaining the master/slave principle (Piconet rules):

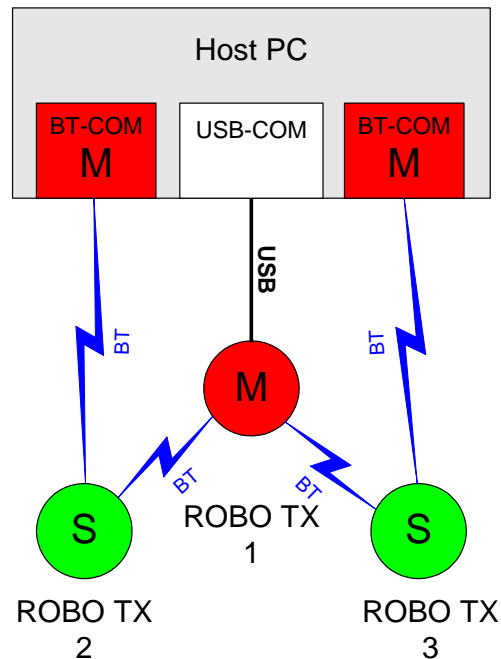


Fig. 6-4

The possibility of working with multiple PCs in online mode in particular is of course always an option (only one USB connection can be used per PC). This option is recommended particularly for novices, since otherwise it may be easy to lose oversight of the Bluetooth connections, Bluetooth addresses and COM ports in the case of complex scenarios:

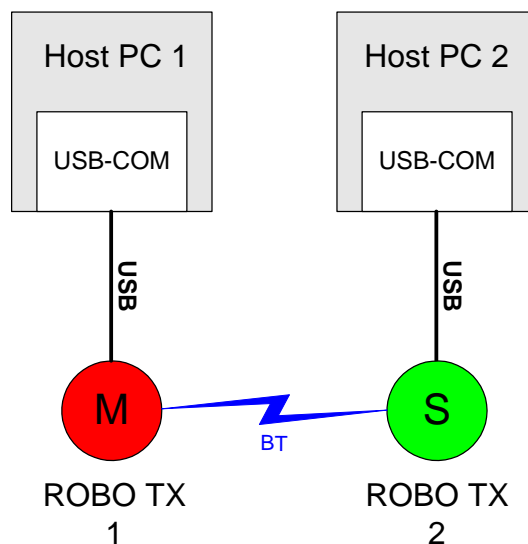


Fig. 6-5

7 I²C interface

The I²C bus (also called IIC or TWI) is a simple 2-wire bus introduced by Philips Semiconductor for inter-chip communication. It allows for the simple and affordable exchange of small to medium amounts of data over short distances.

Data bits are transmitted synchronously to a clock signal on the I²C bus. Two clock speeds are available by default:

"Standard" 100 KHz clock
 "Fast" 400 KHz clock

Most I²C devices today work with 400 KHz. Using the existing function API (see the separate **Windows_Library_ftMscLib.pdf** document), I²C write and read operations can be carried out in which the clock speed can be set for each device.

A I²C device can extend the clock cycle (clock stretching). This is automatically detected and supported by the firmware. The callback functions in this case return later, if applicable.

7.1 I²C hardware connection

The ROBO TX Controller includes a I²C master interface on the EXT2 extension port. There is also a 5 V output on this connector, over which a maximum of 150 mA can be drawn. If an attempt is made to draw more current than that, the 5 V output is switched off by a protective circuit.

The data and clock wires feature a 2.2 KOhm pull-up resistor inside the ROBO TX Controller.

The internal circuitry of the ROBO TX Controller is shown here:

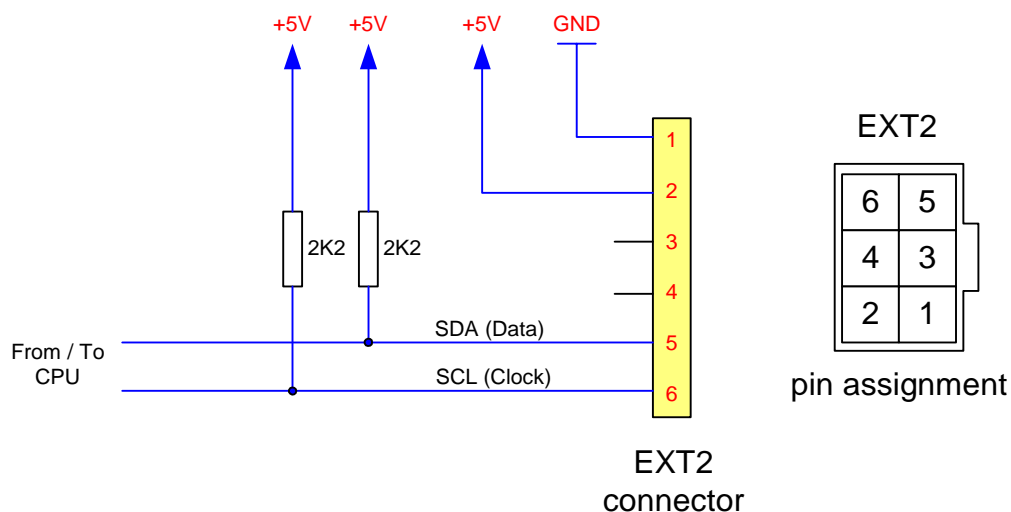


Fig. 7-1: I²C internal wiring in the ROBO TX Controller

Since the I²C interface is a bus structure, multiple I²C devices can be connected to the same bus at the same time. Addressing is done via the respective I²C device addresses, which must differ in this case.

8 Demo applications written in C

The subfolder \Demo_Static_Lib_C contains source code examples for C programs using the static version of the library. It is therefore possible to integrate ROBO TX Controller library functions in one and the same executable. The examples are also available as MS Visual C++ V 6.0 projects.

The examples show simple applications for controlling the ROBO TX Controller:

LightRun

An example of using the individual outputs O1 through O8 to control lights, for instance.
The program quits automatically after running briefly.

MotorRun

Simple motor control (varying speeds) at output M1.
The program quits automatically after running briefly.

StopGo

Demo program for controlling an encoder motor. Counter pulses are read and, if the counter status ≥ 1000 , the program quits. The motor is started (=1) or stopped (=0) via digital input I8.
The program quits automatically after reaching the counter status of 1000.

TrafficLights

Simple traffic light control on O1 – O3.
The program quits automatically after running briefly.

During creation, all demos access files in the shared subdirectories \Common\Lib and \Common\Inc.

The executables created are started at the command line.

Example for the call:

Start the program LightRun by calling RUN.BAT in the directory
\Demo_Static_Lib_C\LightRun\Release

The USB COM port used by the ROBO TX Controller is automatically detected.

9 Demo applications written in C++

The \Demo_Dll_C++ subdirectory contains the source code examples for C++ programs that use the ftMscLib.dll dynamic library. The examples are also available as MS Visual C++ V 6.0 projects.

The examples show simple applications for controlling the ROBO TX Controller:

LightRun – lights

An example of using the individual outputs O1 through O8 to control lights, for instance.
The program quits automatically after running briefly.

MotorStop – ultrasonic distance sensor and encoder motor

Demo program for controlling an encoder motor at M1 (encoder output at C1 respectively). In this case the value of an ultrasonic distance sensor is evaluated at I1. Without "seen" obstacles, the motor M1 turns. If the measured value is below the limit of 15, the motor is stopped. Each time the motor stops, the counter status reached is output by the counter input C1 after starting again. The counter status of the counter pulse is then reset to 0 for the next measurement.
The program quits when <Ctrl> + <C> are pressed on the computer.

WarningLight – ultrasonic distance sensor and light

Demo program for controlling a lamp on O8 with distance measurement via an ultrasonic distance sensor at I1. The light flashes at varying speed intervals, depending on the distance measured via the ultrasonic distance sensor.
The program quits when <Ctrl> + <C> are pressed on the computer.

MotorEx_2M_Master – synchronizes two encoder motors

Demo program for controlling two encoder motors at M1 and M2 (encoder outputs at C1 and C2 respectively) and for synchronized operation of the two motors. Both motors run back and forth cyclically every 200 steps. If the motor is braked while running (e.g. by using mechanical resistance), the other motor adapts its speed accordingly.
The program quits when <Ctrl> + <C> are pressed on the computer.

MotorEx_Ext1 – operates encoder motor on extension controller

Demo program for controlling an encoder motor at M1 (encoder output at C1 respectively) on an extension module (slave extension 1). The motor runs back and forth cyclically every 200 steps. The master module is the one connected to the PC.
The program quits when <Ctrl> + <C> are pressed on the computer.

StopGoBtButtonPart and **StopGoBtMotorPart** – Bluetooth messaging

See section 9.1 further down for a detailed description.

I2cTemp – I²C temperature sensor DS1631

Demo program for controlling a DS1631 external temperature sensor with the I²C interface (Conrad Electronic part. no. 19 82 98).

After a brief initialization sequence, the current temperature value is read out every 0.5 seconds and is displayed on the computer monitor.

You can press any button to quit the program.

Note: other Conrad C control series sensors can also be used and connected directly to the ROBO TX Controller. The I²C connection is compatible.

I2cTpa81 – I²C thermopile array infrared sensor TPA81

Demo program for controlling an external TPA81 thermopile array sensor with the I²C interface (available, for instance, from www.roboter-teile.de). This provides non-contact temperature control as well as easy locating of heat-emitting objects thanks to its array structure (8 pixels).

After a brief initialization sequence, the value of the ambient temperature as well as all 8 measuring points (pixels) are output on the computer monitor. If you move the thermal source in front of the lens, you can see the value with the pixel that has the thermal source in focus rise. The display is updated every 0.5 seconds.

You can press any button to quit the program.

During creation, all demos access files in the shared subdirectories \Common\Lib and \Common\Inc.

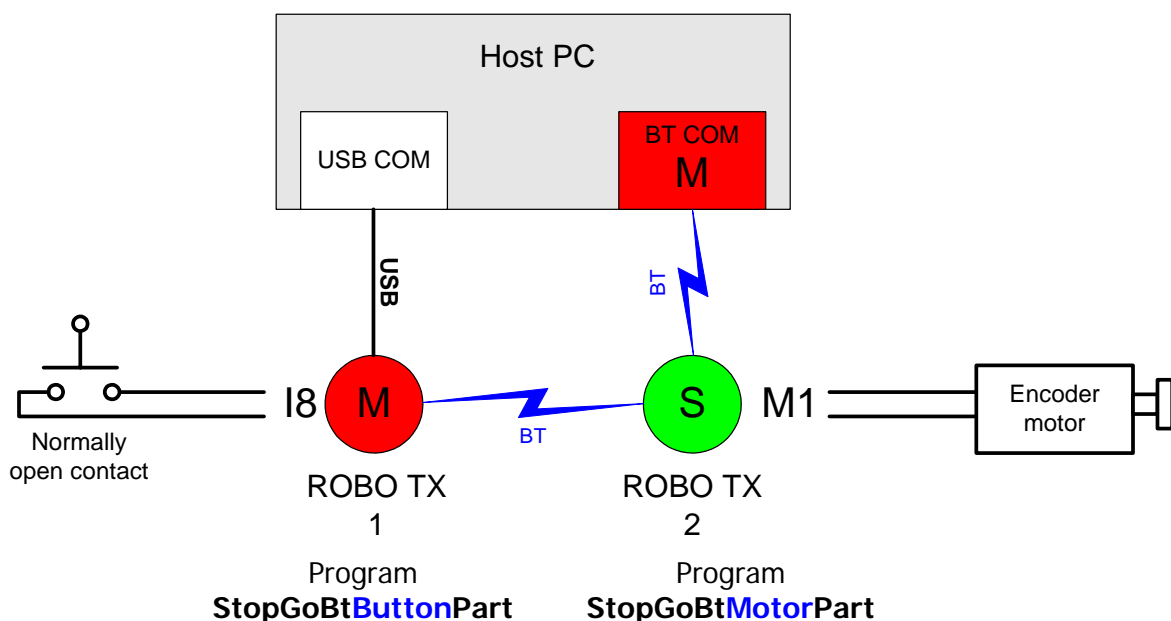
The executables created are started at the command line. Example for the call:

Start the program LightRun by calling RUN.BAT in the directory
 \Demo_D11_C#\LightRun\Release

The USB COM port used by the ROBO TX Controller is automatically detected.

9.1 Example: two controllers and Bluetooth messaging

The following example demonstrates the exchange of Bluetooth messages between two controllers using programs written in C. In this example, the following structure is required:



The MASTER controller (ROBO TX 1, on the left in the figure), is connected to the PC via the USB port; the SLAVE controller (ROBO TX 2, on the right in the figure) is connected via Bluetooth. The SLAVE controller must be assigned a COM port number on the PC. This is entered in the RUN_BT.BAT file in the directory Demo_Dll_C++\StopGoBtMotorPart\Release\ (e.g. COM111). This makes it possible for the SLAVE controller to run in online mode via Bluetooth.

The StopGoBtButtonPart program runs on the MASTER controller, which actively establishes the Bluetooth connection; the StopGoBtMotorPart program runs on the SLAVE controller, which waits for a Bluetooth connection.

Before starting the programs, the Bluetooth addresses of the controllers must be entered into the BT_addr.c file in the directory Common\C\ as follows (example only):

```
UCHAR8 bt_address_table[BT_CNT_MAX][BT_ADDR_LEN] =
{
    {0x00, 0x13, 0x7B, 0x53, 0x10, 0xE7}, // Bluetooth address of ROBO TX 1 (MASTER)
    {0x00, 0x13, 0x7B, 0x52, 0xB2, 0x11}, // Bluetooth address of ROBO TX 2 (SLAVE)
};
```

Next, the programs need to be regenerated so that the correct Bluetooth addresses are used in them.

The programs are then started as follows:

First the StopGoBtMotorPart program in the directory Demo_Dll_C++\ StopGoBt MotorPart\Release\ is started by calling RUN_BT.BAT.

Next, the StopGoBtMotorPart program in the directory Demo_Dll_C++\StopGoBtButtonPart\Release\ is started by calling RUN.BAT.

If you now press push-button switch I8 on the ROBO TX 1 (MASTER), the StopGoBtButtonPart program sends a Bluetooth message to ROBO TX 2 (SLAVE) and the motor on this controller begins to turn. When you release the button, the motor stops. The StopGoBtMotorPart program on ROBO TX 2 sends the position of the encoder motor back to ROBO TX 1 via a Bluetooth message. If the 1000 position is reached, the StopGoBtButtonPart program stops and the Bluetooth connection between ROBO TX 1 and ROBO TX 2 is terminated.

10 Demo applications written in C#

The \Demo_Dll_C# subdirectory contains the source code example for C# programs that use the ftMscLib.dll dynamic library. The example is available as a MS Visual Studio 2008 project.

The example shows simple a application for controlling the ROBO TX Controller:

MotorStop ultrasonic distance sensor, encoder motor and menu buttons

Demo program for controlling an encoder motor at M1. In this case the value of an ultrasonic distance sensor is evaluated at I1. Without "seen" obstacles, the motor M1 turns. If the measured value is below the limit of 15 cm, the motor is stopped. Each time the motor stops, the counter status reached is output by the counter input C1 after starting again. The counter status of the counter pulse is then reset to 0 for the next measurement.

The program is terminated by using <Ctrl>+<C> on the computer or by pressing any push-button switch on the ROBO TX Controller.

Start the program by calling RUN.BAT in the directory
\Demo_Dll_C#\MotorStop\MotorStop\bin\Release\

The USB COM port used by the ROBO TX Controller is automatically detected.

11 Multi-controller operation

To set up larger models, it may be necessary to connect multiple ROBO TX Controllers. This is possible using what are called "extension ports". The connection is made both physically as well as at the command level through a master/slave structure. In a chain of connected interfaces, there is always one master and up to 8 slaves, which are controlled remotely by the master.

Each TX-C has 2 extension ports that represent a serial expansion bus. Only one 6-pin ribbon cable (female connector) is required to make a connection from one interface to the adjacent interface. The data connection is thus made from the master to each of the slaves, as shown below.

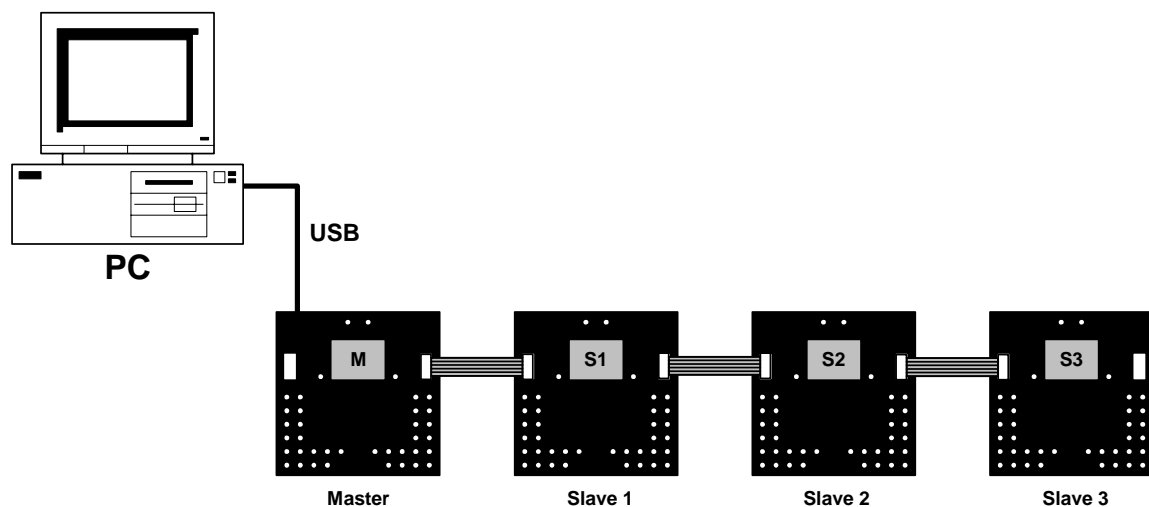


Fig. 2: Linking interfaces for multi-interface operation

The expansion bus is an RS-485 interface with taps on each interface. Since each interface has two physical connections and the expansion bus is through-connected between the two connections to the interface, simple wiring of adjacent interfaces is possible without having to use a T-bus connector, still achieving a traditional bus structure.

Communication on the expansion bus is also master/slave oriented. The master polls all connected slaves cyclically and can thereby exchange I/O information. To ensure that each slave can be controlled individually and separately from the master, each slave must have its own unique address. The mode (master or slave) as well as the slave address can be configured using the "Settings" menu.

The slaves are controlled by the master in online mode in the same way as if controlling a single board from a PC in online mode. The master in this case is the only one that can communicate with the PC. If a request or command from the PC is made for one of the slaves in online mode, this request is routed to the respective slave by the master.

12 Updating the ROBO TX Controller firmware

There are several ways to update the ROBO TX Controller firmware:

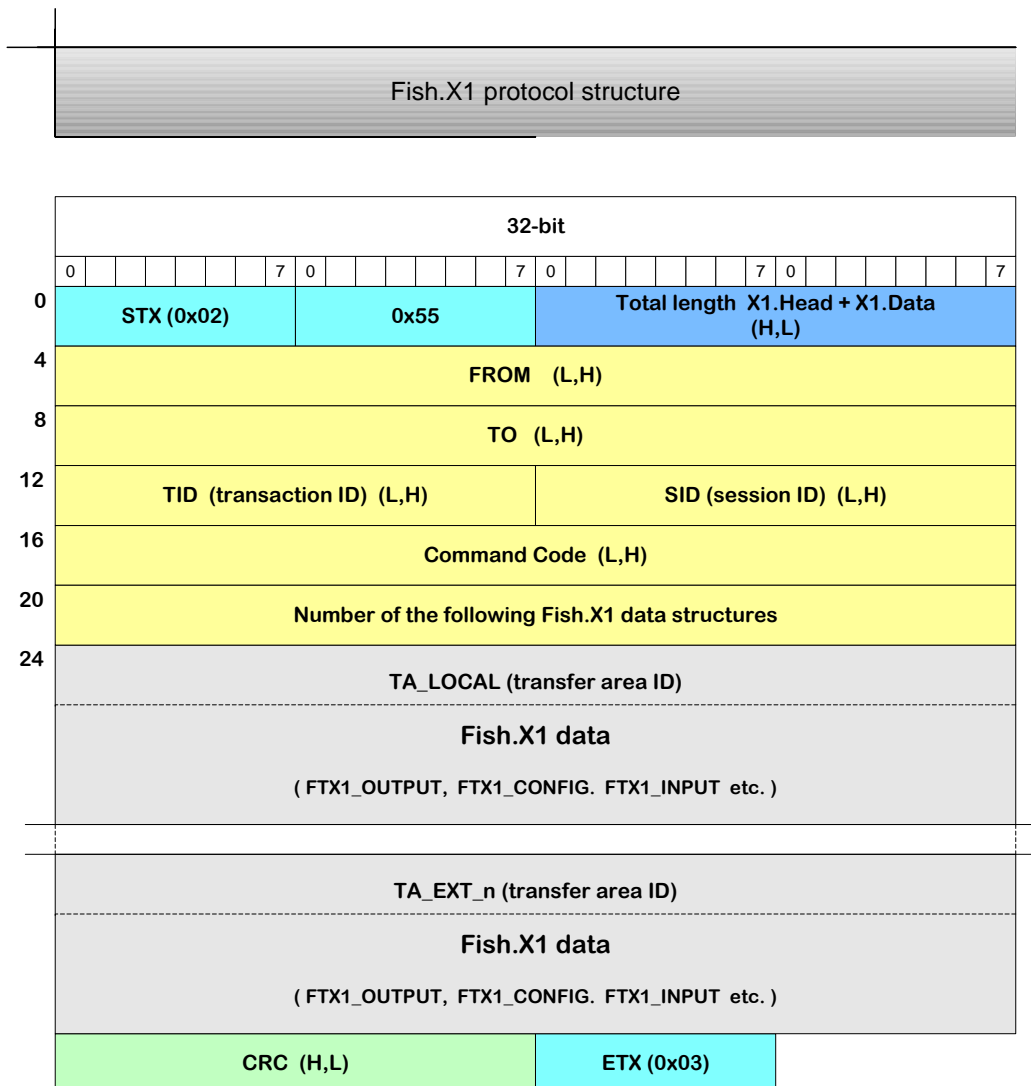
- Via ROBO Pro (usually offered automatically the first time a connection is made, as long as an update is necessary)
- Using the "Firmware Update" function in RoboTxTest (see section 4.5). In this case, refer to the subdirectory **/Firmware** supplied with this package.
- From a self-written PC application with the help of the "RoboTxFwUpdate" library function (refer also to the **Windows_Library_ftMscLib.pdf** document).
- Using the Repair tool (see the separate package with the relevant description)

13 Low-level API (COM port level)

The low-level API uses the particular COM port of a ROBO TX Controller itself (USB or Bluetooth). The three sub-interfaces are described below.

13.1 Fish.X1 interface protocol

The following describes the structure and contents of a Fish.X1 data packet between the PC library and the ROBO TX Controller.



- Fish.X1 frame (start, end)
- Length of data packet (Fish.X1 Header + n * Fish.X1 data structure(s))
- Fish.X1 Header
- Fish.X1 payload (controller ID (local, ext.) + X1.data structure)
- CRC (calculation across data length, Fish.X1 Header and Fish.X1 data)

Data packet description and contents

Offset	Length	Contents and meaning
0	1	STX (0x02) Marks the beginning of a Fish.X1 data packet with the following byte
1	1	0x55
2	2	Length (H,L) of a data packet Consists of the header length (20 bytes) and the payload length.
4	4	Bus address, 4 bytes (L,H), <FROM> (data packet transmitter) Possible values: BUS_ADR_WILDCARD=1, BUS_ADR_MASTER=2, BUS_ADR_SLAVE1=3 ... BUS_ADR_SLAVE8=10
8	4	Bus address, 4 bytes (L,H), <TO> (data packet receiver) Possible values, see above
12	2	Transaction ID (TID) , 2 bytes (L,H) Each data packet is marked by the transmitter with a sequential number; the response to a request is returned by the message receiver with the same TID.
14	2	Session ID (SID) , 2 bytes (L,H) The ROBO TX interface internally manages a session ID, which is reassigned every time there is a change in task from local to remote or remote to local. For example, the session ID is reassigned by the firmware if a timeout occurs during communication. The transmitter can then respond accordingly.
16	4	Fish.X1 command code (tele-code), 4 bytes (L,H) Defines a unique request or reply code. The command code thus describes the supplied data structure in the payload.
20	4	Number of subsequent Fish.X1 data structures, 4 bytes (L,H). (= n)
24	4	Transfer area ID , 4 bytes (L,H) The ID specifies for which ROBO TX Controller, and thus for which transfer area subsection, the subsequent data structure is addressed. The data structure specified via the command code is updated from the individual transfer area. The interfaces can be configured as master (always local) or as slaves (to a master). A corresponding ID is specified here via the transfer area ID: TA_LOCAL=0, TA_EXT_1=1, TA_EXT_2= 2, etc. The master (TA_LOCAL) manages the IO values of the individual slaves in its transfer area and can thereby assign the current IO data using the share memory ID.
28	m	Message payload, dependent upon the transmitted data structure. The payload, together with the share memory ID, can also have a length of 0, typically a request or a reply to a request. The structures defined in the header file ROBO_TX_FW.H are set as the payload. The command code identifies the supplied data structure.
24 + n*(4+m)	2	CRC, 2 bytes (H,L) The CRC is calculated from byte 2 (data length) to the end of the payload.
24 + 2 n*(4+m)	1	ETX (0x03) Marks the end of a Fish.X1 message

13.2 Remote shell

The remote shell (or remote console) makes it possible to carry out file operations in the ROBO TX Controller file system as well as to call up other information. The remote shell commands are all ASCII based and can be input directly from a terminal program such as Windows HyperTerminal. Usually there is one response for each command.

If there is an active connection to the ROBO TX Controller, a prompt appears when the <RETURN> key is pressed which displays the host name of the activated ROBO TX Controller and the interface.

Example: **ROBO TX-511\USB>** for USB
 or: **ROBO TX-511\Bluetooth>** for Bluetooth

The remote shell commands are as follows:

Command	Description
?	Outputs the list of valid commands
help	Outputs the list of valid commands
version	Outputs the current version number of the firmware
get_ser_num	Outputs the ROBO TX Controller serial number
get_board_ver	Outputs the hardware revision ID
get_eeprom	Outputs a structured representation of the board data
clean_disk	Deletes unnecessary files from the specified disk
erase_disk	Erases the specified disk of the ROBO TX Controller ¹⁾
erase_flashdisks	Erases all flash disks of the ROBO TX Controller ¹⁾
erase_boot	Erases the bootstrap of the ROBO TX Controller ¹⁾
erase_phase0	Erases the boot loader of the ROBO TX Controller ¹⁾
dir	Displays the directory contents
type	Outputs the contents of a text file
del	Deletes the specified file
copy	Copies the specified file
format	Formats the specified disk
ren	Renames the specified file
xrecv	Activates the file receiving process on the ROBO TX Controller
load	Loads a ROBO program from one of the disks into the program memory so that it can be executed.
run	Starts the ROBO program that is currently in the program memory.
stop	Stops the currently running ROBO program.

- 1) These commands are only to be used when you know exactly what you are doing. You could otherwise cause a problem that would prevent the ROBO TX Controller from starting. If this happens, you can repair the firmware using the Repair Kit.

13.3 Xmodem file transfer

Using the Xmodem file transfer interface, you can transfer files (including ROBO programs) from the computer to the ROBO TX Controller. This is not intended to be used the other way around (to transfer data from the TX-C to the computer).

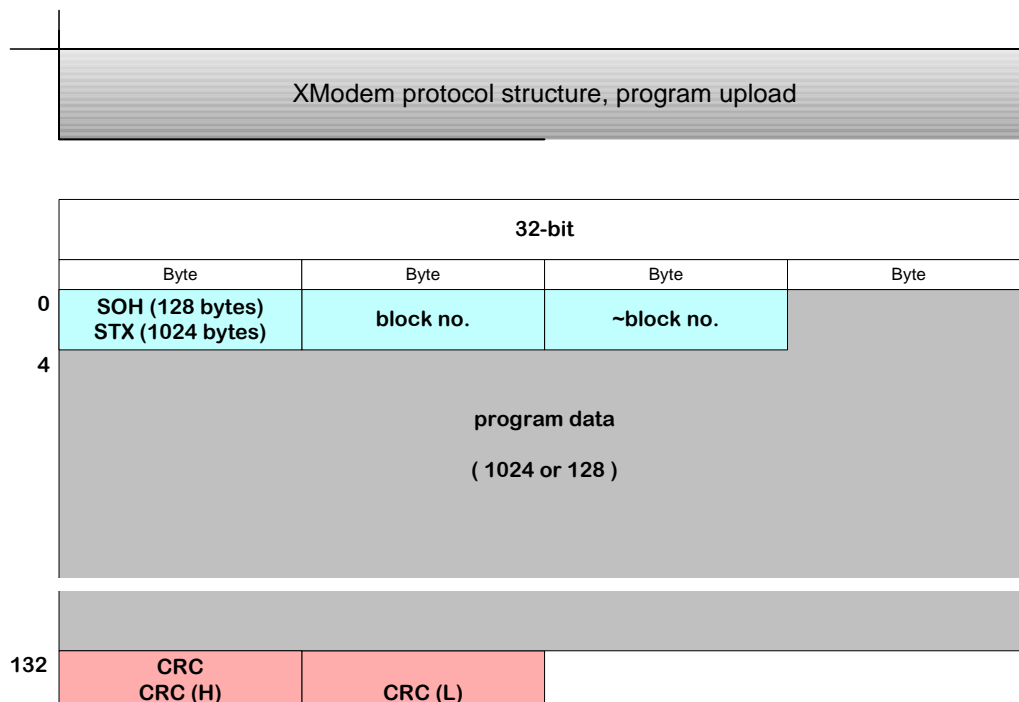
To download files, the Xmodem protocol is used with a fixed packet length and controls the secured transfer of data. The Xmodem protocol runs block-by-block, where the data to be transferred is distributed into blocks of equal size. The sizes of the blocks are always 1029 or 1028, or 133 or 132, depending on the length of the CRC, and are filled with any applicable characters. Acknowledgments of the transmitted data blocks consist of a single characters.

The download is triggered by the receiver (ROBO TX Controller), which sends a NAK or a 'C' in response to the request:

```
xrcv /<device>/"<file>" <length>
```

. In the case of a NAK, the length of the CRC is specified with a byte; if the receiver sends a 'C' as confirmation, the CRC length is declared with 2 bytes. A checksum received without errors is confirmed by the receiver with ACK (acknowledge). If an error occurs, the block transmitted is rejected with NAK (negative acknowledge) and then up to ten attempts are made to resend it. The end of the download is indicated by the transmitter with EOT (end of transmission). This must also be confirmed using ACK. The checksum is the arithmetic sum of all data bytes, ignoring the 16-bit overflow.

The file name specified in the request is to be limited using the " character and can therefore also contain spaces. Moreover, the length of the file must be specified in the request. Entering <device> specifies on which target disk the file is to be saved. The RAM disk and flash disk are available as target disks; the string "ramdisk" or "flash" must be specified in this case.



Description and contents of a data packet during upload (Xmodem protocol)

Offset	Length	Contents and meaning
0	1	SOH (0x01), start of heading Input when the block size is 128 bytes STX (0x02), start of text Input when the block contains 1024 bytes
1	1	Current block number, starting with 0
2	1	Ones' complement of the block number
3	128 or 1024	Data block consisting of the data of the file or program to be transmitted
131	1	CRC, arithmetic checksum via data bytes or CRC (H) high value of checksum
132	1	CRC (L), low value of checksum

In addition to the NAK and ACK acknowledgments, the receiver can also respond to a data block using CAN (= cancel). The data transfer must be interrupted by the transmitter and stopped using a closing EOT.

14 Document change history

Version	Date	Author	Other comments
1.0	10/2/2009	Entire team	Initial version
1.1	10/9/2009	Peter Duchemin	Section 4.5 and chapter 10: Information on the supplied /Firmware subdirectory Section 11.1, comment added to figure showing structure. Missing "Number" field (Offset 20) added to table. Section 11.2, footnote added under table
1.2	12/11/2009	Peter Duchemin	Only version numbers on coversheet were adapted
1.4	12/23/2011	Peter Duchemin	Added section covering the Bluetooth Message API, added more examples to include a Bluetooth messaging demo.
1.5	4/24/2012	Peter Duchemin	Added section covering the I ² C interface, added more examples to include two demos using I ² C sensors