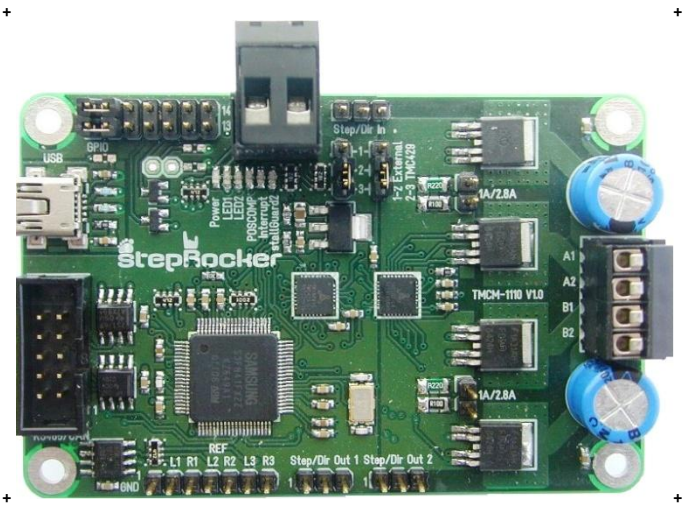
Open Source Firmware V1.00

TMCL-LITE FIRMWARE MANUAL



stepRocker

1-axis stepper controller/driver
1A RMS / 2.8A RMS
24V DC (10… 30V DC)
USB, RS485

TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

www.trinamic.com

TRINAMIC
MOTION CONTROL

Table of Contents

1 Introduction

TMCL-LITE offers basic commands which can be used in direct mode, only. Commands for standalone use of the module are not included, but can be created. This special TMCL™ firmware for the stepRocker has open source codes. With the TMCL-LITE firmware TRINAMIC invites you to think over own program structures and specific commands which perfectly match to your needs. Share and discuss your ideas with other users in our forum and make use of TRINAMICs unique features coolStep™, stallGuard2™, and dcStep™. Together with our customers we intend to newly create a great deal of motion control firmware.

The module comes with a preinstalled complete TMCL™ version with a big range of commands. This version is without source codes and thus cannot be modified. The firmware running on the microprocessor of the TMCM-1110 stepRocker consists of two parts, a boot loader and the firmware itself. The boot loader is installed during production and testing at TRINAMIC and remains normally untouched on the module. For using TMCL-LITE the provided firmware has to be changed.

1.1 Installing TMCL-LITE

Before using TMCL-LITE it is necessary to download the program to your stepRocker. The appropriate PC tool for downloading TMCL-LITE is the *stepRocker software installer*. Please download and start the installer on your PC. Afterwards load the hex file in the installer. For updating the firmware of the stepRocker it must be hex-format.

Updating the firmware can be done via USB interface or RS485 interface. Further, TRINAMIC recommends the *Segger J-Link programmer* for faster access and debugging functions.

1.2 Compiling TMCL-LITE

For compiling the TMCL-LITE source code we recommend to use the GNU C compiler for ARM and Cortex microcontrollers. The source code has been compiled with the Codesourcery G++ Lite distribution of the GNU C compiler that can be downloaded at www.codesourcery.com. The Lite version of this GCC distribution only contains command line tools. Of course also other GNU C compiler distributions can be used, but this has not been tested by Trinamic. Also commercial C compilers for ARM/Cortex microcontrollers can be used but in this case it might be necessary to change parts of the source code.
When using the Codesourcery G++ Lite command line tools, compiling and linking will be contolled by the makefile provided with the source code. To compile, just open a command line window, change to the directory where the code and the makefile are located and then type `make hex`. Then all files that have to be compiled will be compiled automatically. When compiling has been successful all the resulting object files will be linked to an ELF file and a HEX file will be generated from this ELF file.
If you wish to recompile all files (also those that have not been changed), type `make clean` before you type `make hex`. This will erase all output files before recompiling the entire source code.
All compiler and linker output files (also the final HEX file named stepRockerMiniTMCL.hex) will be put into the directory ROM_RUN (which will be created if not already there).

2 The Boot Loader

The boot loader is installed during production and testing at TRINAMIC and remains normally on the module. With this boot loader it is possible to download the firmware to the stepRocker via the USB port or via the RS485 port (we recommend using the USB port as downloading through RS485 takes quite a long time).
Whenever the stepRocker is in boot loader mode, LED1 and LED2 are on.
For downloading a HEX file to the stepRocker either use the stepRocker Software Installer (stepRockerBoot.exe) or the Install OS function of the TMCL-IDE V2.03 or higher.
In the Software Installer, first select the appropriate port then click the Load button to select the desired HEX file and finally click the Start button to start the download process.
If the download process cannot be started because the firmware in the stepRocker does not react on any TMCL commands or implements a different protocol than the TMCL protocol the boot loader can also be activated by plugging on two jumpers as shown in Figure 1. After switching on with the two jumpers plugged on, the stepRocker stays in boot loader mode (LED1 and LED2 are on), and the stepRocker Software Installer or the Install OS function of the TMCL-IDE can be used as usual.

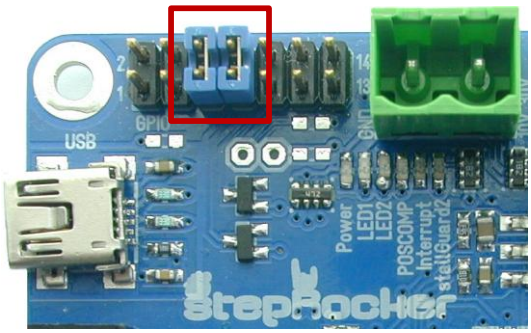


Figure 1 Achtivating the boot loader with jumpers

2.1 Compiling for Use with the Boot Loader

The boot loader resides in the first 16KB of the microcontroller FlashROM. This means that the application firmware has to be linked at the start address 0x4000. This is controlled by the linker script provided with the source code. In the makefile the line "BOOT_LOADER=TMCM-BL" (line 75) activates the linker script that makes the linker putting the code at start address 0x4000.
When changing this line (or when making other changes to the makefile) please do not forget to execute `make clean` before executing `make hex` so that the entire code gets re-compiled and re-linked.
When using a different compiler than GCC please make sure that the code will start at address 0x4000 when the boot-loader is to be used.

2.2 Compiling for Use with other Programmers

Instead of the boot loader other programmers can also be used to directly program the FlashROM of the microcontroller (e.g. to enable source level debugging). This must be a programmer that supports SWD programming (e.g. the Segger J-LINK programmer or the IAR SM-Link programmer). Such programmers can be connected either through the soldering pads on the bottom side of the stepRocker PCB.
When using such programmers, the stepRocker boot loader will be erased. This means that only the application firmware resides in the FlashROM, starting at address 0. For this case the code has to be linked at address 0 also. To achieve this, comment out the line "BOOT_LOADE=TMCM-BL" (line 75) in the makefile and recompile the code (as after every change to the makefile, execute `make clean` before executing `make hex`).
When using other compilers than GCC please make sure that the code will start at address 0.

3 Structure of the Source Code

3.1 Doxygen Documentation

The source code also comes with an automatically generated documentation (generated using the DOXYGEN tool) that shows the overall structure and some explanation for all functions. Please make use of that documentation also. It is provided in HTML format. Just double click the INDEX.HTM file that can be found in the "docs" directory. You can then browse through this automatically generated documentation.

3.2 Files

The TMCL-LITE source code consists of several C files and header files. A makefile and linker script files for the GNU linker are also provided. Furthermore, the programming library for the Samsung S3FN41 microcontroller is also provided with TMCL-LITE.
Here is a short list of the most important files, please see the automatically generated documentation for more information.

Name	Contents
stepRocker.c	Basic MCU initialization, main() function, main loop
Commands.c	TMCL interpreter supporting basic TMCL commands
Eeprom.c	EEPROM access functions
Globals.c	Global variables
IO.c	I/O port access functions
RS485.c	RS485 interface access functions
SPI.c	SPI bus access functions (needed by some other modules)
SysControl.c	Motor monitoring functions
SysTick.c	1ms system tick timer functions
TMC262.c	TMC262 access functions
TMC429.c	TMC429 access functions
stepRocker.h	General definitions
Commands.h	TMCL command definitions
Makefile	Make file for controlling compiling and linking via Make
s3fn41f.ld	Linker script file for linking at address 0 (for use with SWD programmers)
S3fn41f-tmcm.ld	Linker script file for linking at address 0x4000 (for use with the boot loader)

For every C file there is also a header file which defines all public functions implemented in that file.

The directory "lib" contains the MCU library from Samsung (as source code), and the directory "docs" contains the automatically generated HTML documentation. The directory ROM_RUN will be created automatically and then contains all compiler and linker output files (and also the generated HEX file).

4 Functions of the Library

This chapter contains an overview of all functions which are implemented in the library. Please see also the Doxygen documentation and the source code itself.

4.1 TMC429 Motion Controller Functions

The file TMC429.c contains functions necessary for accessing the TMC429 motion controller IC. Here is a list of functions implemented in this file:

4.1.1 Init429

This function initializes the TMC429 and must be called once before any other TMC429 functions are being used. Please not that the TMC429 library itself needs SPI communication, so the SPI must have been initialized before using any TMC429 library function.

4.1.2 ReadWrite429

This function handles the low-level communication with the TMC429 and is used by most of the other functions of the TMC429 library. In most cases there is no need to use this function directly.

4.1.3 Read429Int

This function reads a value from a 24 bit register of the TMC429 (e.g. a position register) and sign-extends the value into a 32 bit integer.

4.1.4 Read429Short

This function reads a value from a 10..12-bit register of the TMC429 (e.g. the velocity registers) and sign-extends the value into a 32 bit integer.

4.1.5 Read429Bytes

This function reads a TMC429 register and puts the result into an array of three bytes. It also returns the TMC429 status byte.

4.1.6 Read429SingleByte

This function reads a TMC429 register and returns the desired byte.

4.1.7 Read429Status

This function returns the TMC429 status byte. If only the status byte is needed this function should be used because it only uses one 8 bit SPI transfer which is slightly faster.

4.1.8 Write429Short

This function writes an integer value into a 10..12 bit register of the TMC429 (e.g. velocity registers).

4.1.9 Write429Int

This function writes an integer value into a 24 bit register of the TMC429 (e.g. position registers).

4.1.10 Write429Bytes

This function writes an array of three bytes into a TMC429 register.

4.1.11 Write429Datagram

This function writes three single bytes into a TMC429 register.

4.1.12 Write429Zero

This function writes zero into a TMC429 register.

4.1.13 SetAMax

This function sets the acceleration parameter and also re-calculates the PMUL and PDIV values according to the new acceleration and the pulse_div and ramp_div settings.

4.1.14 Set429RampMode

This function changes the ramp mode of an axis.

4.1.15 Set429SwitchMode

This function changes the end switch mode of an axis.

4.2 TMC262 Stepper Motor Driver Functions

The file TMC262.c provides a set of functions for accessing all TMC262 register in a convenient way. Software copies are kept for all TMC262 registers which makes it possible to read back all settings. In order to make this feature work properly it is important to access the TMC262 only through the functions of this library and not directly. Here is a list of all functions implemented in the file TMC262.c:

4.2.1 InitMotorDrivers

This function initializes the TMC262 and also the software copies of the TMC262 registers kept in this library. It must be called once at the beginning of the program. Please note that the TMC262 library needs SPI access to the TMC262, so the SPI must have been initialized before using any function of the TMC262 library.

4.2.2 Set262... functions

For every parameter of the TMC262 there is a function for setting it. The names of all these functions start with "Set262".

4.2.3 Get262... functions

Every TMC262 parameter that can be set using a Set262... function can be read back using the appropriate Get262... function. Please note that the TMC262 registers cannot be read back, but the Get262 functions return the software copies kept in this library.

4.2.4 Read262State

This function reads the status register of the TMC262 and decodes the data according to the last selected TMC262 read back mode (which can be selected using the function Set262DriverReadSelect). With each call this function re-writes one of the TMC262 configuration registers. So it is recommended to call this function on a regular basis to ensure that the TMC262 registers always contain the correct data.

4.2.5 Disable262

This function disables the motor driver (by setting the TOff time to zero).

4.2.6 Enable262

This function re-enables the motor driver (by setting the TOff time back to the value last set).

4.3 System Timer Functions

The file SysTick.c contains functions for using the system tick timer of the CPU as an 1ms timer. This is useful for example for timing delays. It contains the following functions:

4.3.1 InitSysTimer

This function initializes and starts the system tick timer. It then runs as a millisecond counter.

4.3.2 GetSysTimer

This function returns the value of the millisecond timer. It starts at zero after calling InitSysTimer and rolls over after 4294967295 milliseconds (= 49.7 days).

4.4 SPI Functions

The file SPI.c contains functions for using the serial peripheral interface (SPI). This is needed by the TMC429 library, the TMC262 library and the EEPROM library.

4.4.1 InitSPI

This function initializes the SPI. It must be called before any SPI access takes place (so before any functions of the TMC429 library, the TMC262 library or the EEPROM library are being used).

4.4.2 ReadWriteSPI

This function sends and receives one byte via SPI. It is the low level SPI function used by all other libraries that need SPI access.

4.5 RS485 Functions

The file RS485.c contains functions for accessing the RS485 interface. It implements an interrupt driven send and receive buffer and automatically switches between sending and receiving. Here is a list of function implemented in this library:

4.5.1 InitRS485

This function initializes the RS485 interface and sets the baud rate. The following baud rates can be set:

InitRS485 parameter	Baud rate
0	9600
1	14400
2	19200
3	28800
4	38400
5	57600
6	76800
7	115200
8	230400

4.5.2 WriteRS485

This function writes a character to the send buffer of the RS485 interface. Sending takes place in background (interrupt driven).

4.5.3 ReadRS485

This function tries to read a character from the RS485 interface. It returns TRUE if a character has been read or FALSE when the receive buffer is empty.

4.5.4 SetUARTTransmitDelay

This function sets the additional delay between receiving the last character and sending the next character.

4.5.5 CheckUARTTimeout

This function returns the state of the RS485 timeout flag and resets it.

4.6 I/O Functions

The file IO.c contains functions for initializing all I/O ports of the CPU. Here is a list of functions implemented in this file:

4.6.1 InitIO

This function initializes all I/O ports that will not be initialized by other initialization functions. It should be called at the very beginning of the program.

4.6.2 DisableInterrupts

This function disables all interrupts.

4.6.3 EnableInterrupts

This function re-enables all interrupts.

4.6.4 GetStallState

This function returns the state of the SG_TEST pin of the TMC262.

4.6.5 SetMotorCurrent

This function sets the motor current (by calling the appropriate TMC262 library function). It is just a wrapper for the Set262StallGuardCurrentScale function.

4.7 EEPROM Functions

The file Eeprom.c contains functions for accessing the AT25128 EEPROM. Please not that the EEPROM functions need the SPI, so the SPI must have been initialized before using any EEPROM function. Here is a list of functions implemented in this library:

4.7.1 WriteEepromByte

This function writes a single byte to an EEPROM location.

4.7.2 ReadEepromByte

This function reads a single byte from an EEPROM location.

4.7.3 WriteEepromBlock

This function copies a memory block to the EEPROM. There are no size limitations for the memory block (except the EEPROM size which is 16384 bytes).

4.7.4 ReadEepromBlock

This function copies data from the EEPROM into a memory block. There are no size limitations (except the EEPROM size which is 16384 bytes).

4.8 System Control Functions

The file SysControl.c contains functions that implement some higher level TMCL functionality: automatic switching to standby current, freewheeling, stop on stall etc.
It contains the function SystemControl() that must be called regularly from the main loop.

4.9 TMCL Interpreter

The file Commands.c contains a small TMCL interpreter. The function InitTMCL() must be called before the main loop begins, and the function ProcessCommand() must be called regularly from the main loop in order to run the TMCL interpreter.
This TMCL interpreter only supports direct mode. It fetches commands from the RS485 interface, executes them and sends back the replies through the RS485 interface. It supports the TMCL commands ROL, ROR, MST, MVP, SAP, GAP, GetVersion and SoftwareReset.

5 Life Support Policy

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this data sheet is believed to be accurate and reliable. However neither responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties, which may result from its use.

Specifications are subject to change without notice.

6 Revision History

6.1 Firmware revision

Version	Date	Author OK - Olav Kahlbaum	Description
1.00	2011-NOV-22	OK	First open source version

6.2 Document revision

Version	Date	Author SD – Sonja Dwersteg	Description
1.00	2012-FEB-06	SD	Initial version

7 References

[TMCM-1110] TMCM-1110 Hardware Manual
[TMCM-1110] TMCM-1110 Firmware Manual (TRINAMICs complete TMCL™ version)
[TMC262] TMC262 Datasheet
[TMC429] TMC429 Datasheet
[TMCL-IDE] TMCL-IDE User Manual
[USB-2-485] USB-2-485 Manual

Please refer to www.trinamic.com.