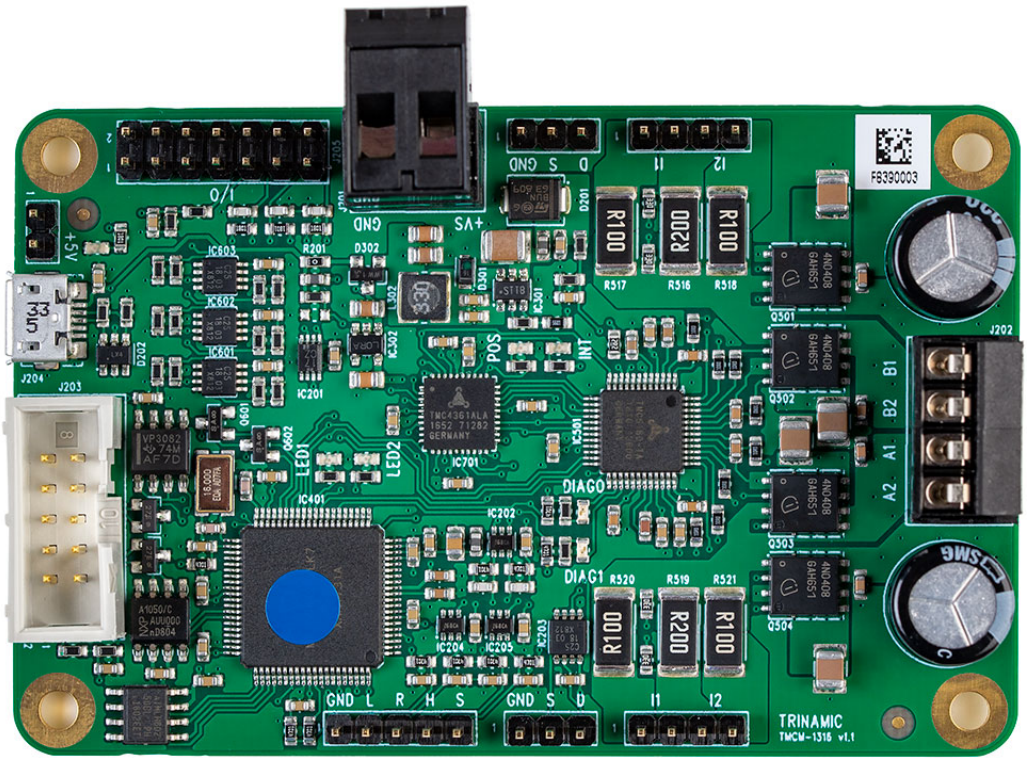
OPEN SOURCE FIRMWARE

TMCM-1316 TMCL™-LITE

Revision V1.00 | Document Revision V1.00 • 2018-OCT-02

TMCL-LITE is an open source firmware for the TMCM-1316 module, provided as C source code with Doxygen documentation. It implements a subset of the TMCL direct mode commands. With the help of TMCL-LITE it is possible to write an own, customized firmware that can use the entire hardware of the TMCM-1316 module. For communicating with a host either the TMCL protocol can be used, or other protocols can be implemented.



Download newest version at: www.trinamic.com

Read entire documentation.

TRINAMIC
MOTION CONTROL

Contents

1 Introduction **4**
1.1 Installing TMCL-LITE . . . 4
1.2 Compiling TMCL-LITE . . . 4

2 The Boot Loader **4**
2.1 Compiling for Use with the Boot Loader . . . 5
2.2 Compiling for Use with other Programmers . . . 5

3 Structure of the Source Code **5**
3.1 Doxygen Documentation . . . 5
3.2 Files . . . 6

4 Functions of the Library **6**
4.1 TMC4361 Motion Controller Functions . . . 7
4.1.1 InitTMC4361 . . . 7
4.1.2 WriteTMC43xxBytes . . . 7
4.1.3 WriteTMC43xxInt . . . 7
4.1.4 ReadTMC43xxInt . . . 7
4.1.5 TMC43xxSetBits . . . 7
4.1.6 TMC43xxClearBits . . . 7
4.1.7 TMC43xxWriteBits . . . 7
4.1.8 PeekTMC43xxEvents . . . 7
4.1.9 ReadAndClearTMC43xxEvents . . . 7
4.1.10 HardStop . . . 7
4.1.11 ResetTMC43xx . . . 7
4.1.12 GetHomeInput . . . 8
4.2 TMC5160 Stepper Motor Driver Functions . . . 8
4.2.1 InitMotorDrivers . . . 8
4.2.2 WriteTMC5160Datagram . . . 8
4.2.3 WriteTMC5160Int . . . 8
4.2.4 ReadTMC5160Int . . . 8
4.2.5 Set5160... functions . . . 8
4.2.6 Get5160... functions . . . 8
4.2.7 Read5160State . . . 8
4.2.8 Disable5160 . . . 8
4.2.9 Enable5160 . . . 8
4.3 System Timer Functions . . . 9
4.3.1 InitSysTimer . . . 9
4.3.2 GetSysTimer . . . 9
4.4 SPI Functions . . . 9
4.4.1 InitSPI . . . 9
4.4.2 ReadWriteSPI . . . 9
4.5 RS485 Functions . . . 9
4.5.1 InitRS485 . . . 9
4.5.2 WriteRS485 . . . 10
4.5.3 ReadRS485 . . . 10
4.5.4 SetUARTTransmitDelay . . . 10
4.5.5 CheckUARTTimeout . . . 10
4.6 USB Functions . . . 10
4.6.1 InitUSB . . . 10
4.6.2 DeInitUSB . . . 10
4.6.3 GetUSBCmd . . . 10

4.6.4 SendUSBReply . . . 10
4.7 CAN Functions . . . 11
4.7.1 InitCan . . . 11
4.7.2 CanGetMessage . . . 11
4.7.3 CanSendMessage . . . 11
4.8 I/O Functions . . . 11
4.8.1 InitIO . . . 11
4.8.2 DisableInterrupts . . . 11
4.8.3 EnableInterrupts . . . 12
4.8.4 GetStallState . . . 12
4.8.5 SetMotorCurrent . . . 12
4.9 EEPROM Functions . . . 12
4.9.1 WriteEepromByte . . . 12
4.9.2 ReadEepromByte . . . 12
4.9.3 WriteEepromBlock . . . 12
4.9.4 ReadEepromBlock . . . 12
4.10 System Control Functions . . . 12
4.11 TMCL Interpreter . . . 12

5 Figures Index 13

6 Tables Index 14

7 Supplemental Directives 15
7.1 Producer Information . . . 15
7.2 Copyright . . . 15
7.3 Trademark Designations and Symbols . . . 15
7.4 Target User . . . 15
7.5 Disclaimer: Life Support Systems . . . 15
7.6 Disclaimer: Intended Use . . . 15
7.7 Collateral Documents & Tools . . . 16

8 Revision History 17
8.1 Software Revision . . . 17
8.2 Document Revision . . . 17

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 stealthRocker is provided as open source software. With the TMCL-LITE firmware TRINAMIC invites you to develop own program structures and specific commands which perfectly match to your needs. Share and discuss your ideas with other users and make use of TRINAMICs unique features like coolStep, stallGuard2 and also closed loop control. Together with our customers we intend to newly create a great deal of motion control firmware.

The TMCM-1316 module comes with a pre-installed complete TMCL firmware with a wide range of commands. But this firmware is not open source and hence cannot be modified. The firmware running on the microprocessor of the TMCM-1316 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. In order to use TMCL-LITE this firmware can simply be installed instead of the standard TMCL firmware using the firmware update function of the TMCL-IDE.

1.1 Installing TMCL-LITE

Before using TMCL-LITE it is necessary to download the program to your TMCM-1316 module. The appropriate PC tool for downloading TMCL-LITE is the firmware update tool of the TMCL-IDE. So you will need to have the TMCL-IDE installed on your PC. Open the firmware update tool in the IDE and load the hex file. Files used for updating the firmware of TRINAMIC modules using the TMCL-IDE must be in HEX format.
Updating the firmware can be done via USB interface or the 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 GNU Tools for ARM Embedded distribution of the GNU C compiler that can be downloaded at https://developer.arm.com/open-source/gnu-toolchain/gnu-rm. As the distributions offered there normally do not include a Make tool you will also need to download a copy of the GNU Make tool (if you do not already have one).
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 GCC 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 stealthRockerMiniTMCL.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 normally remains on the module. With this boot loader it is possible to download the firmware to the TMCM-1316 module 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 TMCM-1316 module is in boot loader mode, LED1 and LED2 are on. For

downloading a HEX file to the stealthRocker please use the firmware update function of the TMCL-IDE V3.0.20 or higher. In the Firmware Update Tool, first select the appropriate port then click the Load button to select the desired HEX file and finally click the Update button to start the download process. If the download process cannot be started because the firmware in the stealthRocker does not react on any TMCL commands or implements a different protocol than the TMCL protocol the boot loader can also be activated by shorting the SWCLK and SWDIO pads. After switching on with these two lines shorted the TMCM-1316 module stays in boot loader mode (LED1 and LED2 are on), and the Firmware Update function of the TMCL-IDE can be used as usual. The SWCLK and SWDIO pads can be found on the bottom side of the PCB, as shown in figure 1.

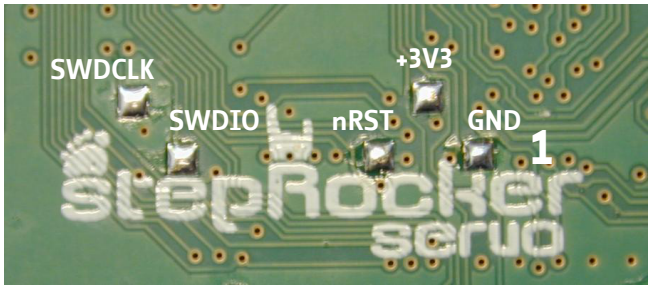


Figure 1: TMCM-1316 Programming Pads

2.1 Compiling for Use with the Boot Loader

The boot loader resides in the first 32KB of the microcontroller FlashROM. This means that the application firmware has to be linked at the start address 0x8000. 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 0x8000. 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 0x8000 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 (for example to enable source level debugging). This needs to be a programmer that supports SWD programming (e.g. the Segger J-LINK programmer). Such programmers can be connected through the soldering pads on the bottom side of the TMCM-1316 PCB (figure 1). There is also the stealthRocker programming adaptor board. It is provided as open source hardware, too. When using such programmers, the TMCM-1316 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_LOADER=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, too.

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 MK20DX128 microcontroller is also provided with TMCL-LITE. Table 1 contains a short list of the most important files. Please see the automatically generated documentation for more information. For every C file there is also a header file which defines all public functions implemented in that file.
The LIB directory contains the library for the MK20DX128 microcontroller (as source code), and the DOCS directory 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).

Most Important Source Files	
Name	Contents
stealthRocker.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
CAN.c	CAN interface access functions
USB.c	USB 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
TMC5160.c	TMC5160 access functions
TMC4361.c	TMC4361 access functions
stealthRocker.h	General definitions
Commands.h	TMCL command definitions
Makefile	Make file for controlling compiling and linking via GNU Make
MK20DX128.ld	Linker script file for linking at address 0 (for use with SWD programmers)
MK20DX128-TMCM.ld	Linker script file for linking at address 0x8000 (for use with the boot loader)

Table 1: Most Important Source Files

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 TMC4361 Motion Controller Functions

4.1.1 InitTMC4361

This function initializes the TMC4361 and must be called once before any other TMC4361 functions are being used. Please note that the TMC4361 library itself needs SPI communication, so the SPI must have been initialized before using any TMC4361 library functions.

4.1.2 WriteTMC43xxBytes

This is a low level function for writing data to a TMC4361 register (a 32 bit value split up into four bytes).

4.1.3 WriteTMC43xxInt

This is a low level function for writing data to a TMC4361 register (32 bit value).

4.1.4 ReadTMC43xxInt

This is a low level function for reading data from a TMC4361 register.

4.1.5 TMC43xxSetBits

This function sets specified bits in a specified register. Bits that are set in the bit mask will also be set in the register. All other bits in the register remain untouched.

4.1.6 TMC43xxClearBits

This function clears specified bits in a specified register. Bits that are set in the bit mask will be cleared in the register, too. All other bits in the register remain untouched.

4.1.7 TMC43xxWriteBits

This function writes to specified bits in a specified register. The value given will be copied to the specified register, limited to the bits specified by start and size. All other bits remain unchanged.

4.1.8 PeekTMC43xxEvents

This function reads the TMC4361 event register, without clearing the event bits (by setting the event clear mask to 0xffffffff before reading the event register).

4.1.9 ReadAndClearTMC43xxEvents

This function reads the TMC4361 event register and clears all event bits specified by EventMask (all bits that are set there).

4.1.10 HardStop

This function stops the motor immediately, without using any deceleration ramp.

4.1.11 ResetTMC43xx

This function resets the TMC4361 (by pulling its reset pin low for some clock periods).

4.1.12 GetHomeInput

This function reads the TMC4361 home input. This works using a little trick: Bit 17 and bit 16 in the REFERENCE_CONF register must be set and the XHOME register must be set to INT_MAX. The InitTMC4361 function does these initializations.

4.2 TMC5160 Stepper Motor Driver Functions

The file TMC5160.c provides a set of functions for accessing all TMC5160 registers in a convenient way. Software copies are kept for all TMC5160 registers which makes it possible to read back all settings. In order to make this feature work properly it is important to access the TMC5160 only through the functions of this library and not directly. This section contains a list of all functions implemented in the file TMC5160.c.

4.2.1 InitMotorDrivers

This function initializes the TMC5160 and must be called once before any other TMC5160 functions are being used. Please note that the TMC5160 library itself needs SPI communication, so the SPI must have been initialized before using any TMC5160 library functions.

4.2.2 WriteTMC5160Datagram

This is a low level function for writing data to a TMC5160 register (a 32 bit value split up into four bytes).

4.2.3 WriteTMC5160Int

This is a low level function for writing data to a TMC5160 register (32 bit value).

4.2.4 ReadTMC5160Int

This is a low level function for reading data from a TMC5160 register.

4.2.5 Set5160... functions

For every motor driver parameter of the TMC5160 there is a function for setting it. The names of all these functions start with "Set5160".

4.2.6 Get5160... functions

Every TMC5160 parameter that can be set using a Set5160... function can be read back using the appropriate Get5160... function. Please note that some of the TMC5160 registers cannot be read back, but the Get5160 functions return the software copies kept in this library.

4.2.7 Read5160State

This function reads the motor driver status register of the TMC5160 and decodes the data.

4.2.8 Disable5160

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

4.2.9 Enable5160

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 (approximately 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 TMC4361 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 TMC4361 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. This section descibes the functions implemented in the RS485 library.

4.5.1 InitRS485

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

RS485 Baud Rates	
InitRS485 Parameter	Baud Rate
0	9600
1	14400
2	19200
3	28800
4	38400
5	57600
6	76800

InitRS485 Parameter	Baud Rate
7	115200
8	230400

Table 2: RS485 Baud Rates

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 USB Functions

The file USB.c contains functions for accessing the USB interface with the help of the USB protocoll stack supplied by the manufacturer of the MK20DX128 microcontroller.

4.6.1 InitUSB

This functions initializes the USB interface and the USB protocol stack. It needs to be called at the beginning of the program, after the I/O ports have been initialized.

4.6.2 DeInitUSB

This function switches off the USB interface and so detaches the module from USB. It only needs to be called before entering the boot loader.

4.6.3 GetUSBCmd

This function tries to read nine bytes from the USB interface (one TMCL command). If there are more than nine bytes in the USB receive buffer then these bytes will be discarded.

4.6.4 SendUSBReply

This function sends nine byte (one TMCL reply) via the USB interface.

4.7 CAN Functions

The file CAN.c provides functions for using the CAN interface. Before using CAN functions please be sure that your TMCM-1316 is fitted with a CAN transceiver.

4.7.1 InitCan

This function initializes the CAN interface. It also sets the CAN bit rate and one or two CAN IDs on which the CAN interface will listen. Only CAN telegrams with these ID will be copied to the receive buffer. The following CAN bit rates can be set:

CAN Bit Rates	
InitCan Parameter	Bit Rate
1	10 kBit/s
2	20 kBit/s
3	50 kBit/s
4	100 kBit/s
5	125 kBit/s
6	250 kBit/s
7	500 kBit/s
8	1MBit/s

Table 3: CAN Bit Rates

4.7.2 CanGetMessage

This function checks if there is a CAN message in the receive buffer and reads the message then.

4.7.3 CanSendMessage

This function puts a CAN message into the transmit buffer. It will then be put onto the CAN bus by the CAN interface.

4.8 I/O Functions

The file IO.c contains functions for initializing all I/O ports of the CPU. The functions implemented in this file are described in this section.

4.8.1 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.

4.8.2 DisableInterrupts

This function disables all interrupts.

4.8.3 EnableInterrupts

This function re-enables all interrupts.

4.8.4 GetStallState

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

4.8.5 SetMotorCurrent

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

4.9 EEPROM Functions

The file Eeprom.c contains functions for accessing the AT25128 EEPROM. Please note that the EEPROM functions need the SPI, so the SPI must have been initialized before using any EEPROM function. This section describes all functions of the EEPROM library.

4.9.1 WriteEepromByte

This function writes a single byte to an EEPROM location.

4.9.2 ReadEepromByte

This function reads a single byte from an EEPROM location.

4.9.3 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).

4.9.4 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).

4.10 System Control Functions

The file SysControl.c contains functions that implement some higher level TMCL functionality like stop on stall, automatic reset of stop events, closed loop initialization and torque mode. It contains the function SystemControl() that must be called regularly from the main loop.

4.11 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 commands. It fetches commands from the USB interface, the RS485 interface and the CAN interface. Then it executes them and sends back the reply through the interface from where the command came. Supported TMCL commands are: ROL, ROR, MST, MVP, SAP, GAP, GetVersion, SoftwareReset.

5 Figures Index

1 TMCM-1316 Programming Pads . . . 5

6 Tables Index

1 Most Important Source Files 6
2 RS485 Baud Rates 10
3 CAN Bit Rates 11
4 Software Revision 17
5 Document Revision 17

7 Supplemental Directives

7.1 Producer Information

7.2 Copyright

TRINAMIC owns the content of this user manual in its entirety, including but not limited to pictures, logos, trademarks, and resources. © Copyright 2018 TRINAMIC. All rights reserved. Electronically published by TRINAMIC, Germany.

Redistributions of source or derived format (for example, Portable Document Format or Hypertext Markup Language) must retain the above copyright notice, and the complete Datasheet User Manual documentation of this product including associated Application Notes; and a reference to other available product-related documentation.

7.3 Trademark Designations and Symbols

Trademark designations and symbols used in this documentation indicate that a product or feature is owned and registered as trademark and/or patent either by TRINAMIC or by other manufacturers, whose products are used or referred to in combination with TRINAMIC's products and TRINAMIC's product documentation.

This Open Source Firmware is a non-commercial publication that seeks to provide concise scientific and technical user information to the target user. Thus, trademark designations and symbols are only entered in the Short Spec of this document that introduces the product at a quick glance. The trademark designation /symbol is also entered when the product or feature name occurs for the first time in the document. All trademarks and brand names used are property of their respective owners.

7.4 Target User

The documentation provided here, is for programmers and engineers only, who are equipped with the necessary skills and have been trained to work with this type of product.

The Target User knows how to responsibly make use of this product without causing harm to himself or others, and without causing damage to systems or devices, in which the user incorporates the product.

7.5 Disclaimer: Life Support Systems

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 document is believed to be accurate and reliable. However, no 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.

7.6 Disclaimer: Intended Use

The data specified in this user manual is intended solely for the purpose of product description. No representations or warranties, either express or implied, of merchantability, fitness for a particular purpose

or of any other nature are made hereunder with respect to information/specification or the products to which information refers and no guarantee with respect to compliance to the intended use is given.

In particular, this also applies to the stated possible applications or areas of applications of the product. TRINAMIC products are not designed for and must not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (safety-Critical Applications) without TRINAMIC's specific written consent.

TRINAMIC products are not designed nor intended for use in military or aerospace applications or environments or in automotive applications unless specifically designated for such use by TRINAMIC. TRINAMIC conveys no patent, copyright, mask work right or other trade mark right to this product. TRINAMIC assumes no liability for any patent and/or other trade mark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

7.7 Collateral Documents & Tools

This product documentation is related and/or associated with additional tool kits, firmware and other items, as provided on the product page at: www.trinamic.com.

8 Revision History

8.1 Software Revision

Version	Date	Author	Description
V1.00	2018-OCT-02	OK	First open source version.

Table 4: Software Revision

8.2 Document Revision

Version	Date	Author	Description
V1.00	2018-OCT-02	OK	First release version.

Table 5: Document Revision