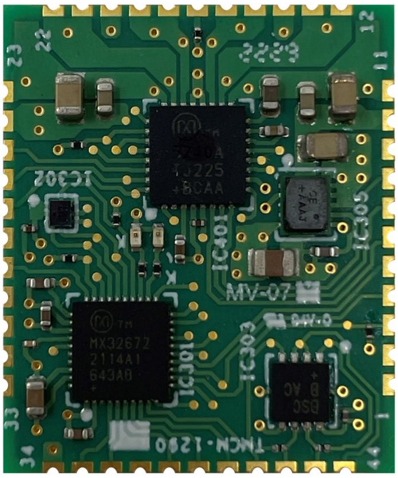
Module for Stepper

MODULE

TMCM-1290 Modbus Firmware Manual

Firmware Version V1.02 | Rev 0: 02/25

The TMCM-1290 is a single axis controller/driver module for 2-phase bipolar stepper motors. The TMCM-1290 Modbus firmware can control the module using the Modbus RTU protocol, supporting all features that can also be used from the TMCL direct mode and making use of the TMC5240 motion controller and motor driver. Dynamic current control, and quiet, smooth, and efficient operation are combined with StealthChop, DcStep, StallGuard2, and CoolStep features.



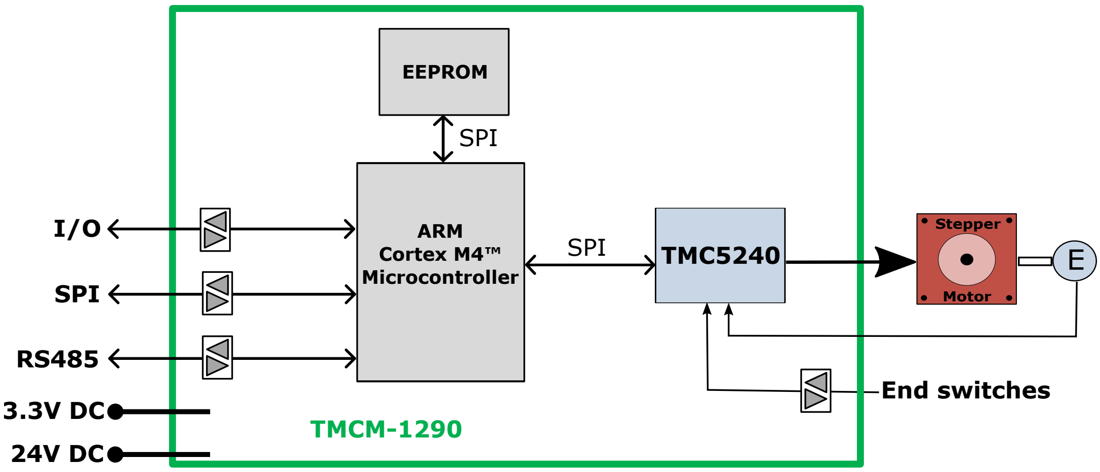
Features

- Single axis stepper motor control
- Supply voltage 24V DC
- Modbus RTU
- Host interface: RS485 or UART (TTL)
- Additional inputs and outputs
- EightPoint ramps
- CoolStep
- StallGuard2
- StealthChop
- ABN encoder interface

Applications

- Broad Market
- System Integrators
- Lab Automation
- Textile
- Packaging
- Life Sciences
- Semiconductor Handling
- Pumps and Motor Drives
- Multi-Axis Applications

Simplified Block Diagram



Read entire documentation.

Contents

1 Preface

This document specifies all Modbus registers used by the Trinamic TMCM-1290 motor control module with Modbus firmware. The Modbus firmware is desgined to fulfill the Modbus RTU standard. This manual assumes that the reader is already familiar with the basics of the Modbus RTU protocol, defined by the Modbus standard. Download the documents that define the Modbus standard free of charge from the website of the Modbus organization: https://www.modbus.org/. The Modbus firmware is built on the TMCL firmware. All TMCL axis parameters and user variables as well as other TMCL features are mapped to Modbus registers. This means that most TMCL features can be used through Modbus registers.

If necessary, it is always possible to turn the module into a TMCL module by loading the TMCM-1290 TMCL fimware through its host interface using the firmware update function of the TMCL-IDE.

1.1 General Features of this Modbus Implementation

Main Characteristics

- Communication according to Modbus RTU standard
- Communication interface: serial (RS485, RS232, or TTL UART)
- Supported transmission modes: RTU only
- Byte coding: 1 start bit, 8 data bits, no parity bit, one stop bit (8N1)
- Baud rates: 9600bps. . . 1000000bps (set using special Modbus register)
- Server addresses: 1. . . 247 (set using special Modbus register)
- Register address base: 0

Supported Modbus Functions

- 03_h: Read Holding Registers
- 04_h: Read Input Registers
- 05_h: Write Single Register
- 08_h: Diagnostics
- $0B_h$: Get Comm Event Counter
- 10_h: Write Multiple Registers
- 11_h: Report Server ID
- 16_h: Mask Write Register
- 17_h: Read/Write Multiple Registers

1.2 Abbreviations Used in this Manual

Abbreviations	
ADU	Application Data Unit
HDLC	High Level Data Link Control
HMI	Human Machine Interface
I/O	Input/Output
MB	Modbus Protocol
MBAP	Modbus Application Protocol
PDU	Protocol Data Unit
PLC	Programmable Logic Controller

Table 1: Abbreviations Used in this Manual

1.3 Firmware Update

The software running on the microprocessor consists of two parts, a bootloader and the Modbus firmware itself. Whereas the bootloader is installed during production and testing at TRINAMIC and remains untouched throughout the whole lifetime, the Modus firmware can easily be updated. Writing the value 1234_h to Modbus register $FF00_h$ turns the module into the bootloader mode. After that, the new firmware can be loaded into the module using the firmware update function of the TMCL-IDE.

2 Communication

This section gives a brief introduction to Modbus communication and the Modbus functions supported by this implementation. Refer to the Modbus standards (*Modbus Application Protocol* and *Modbus over Serial Line*) for more information. These can be downloaded from https://www.modbus.org/tech.php.

2.1 Data Representation

To access the process data, the Modbus protocol uses a set of 16-bit wide registers. Each register is numbered using a 16-bit wide index (0…65535). To access 32-bit wide data, two adjacent registers are used together for one value. These should then always be accessed together using the appopriate Modbus functions. Do not access only one half of a 32-bit value. The first of the two registers (the one with the lower index) is used for the least significant 16 bits.
On many PLCs, the register indices start at 1, whereas in the protocol (the PDUs), the register indices start at 0. This manual refers to PDU addresses (starting from 0). So, with some PLCs and HMIs, 1 has to be added to the register addresses used in this manual.

2.2 Modbus Functions

The **Modbus Application Protocol** defines a set of functions for accessing the process data. The TMCM-1290 Modbus firmware implements a subset of these.

2.2.1 03 (03_h): Read Holding Registers

This function reads one or more registers. In this implementation, it does the same as function 04 (Read Input Registers).

2.2.2 04 (04_h): Read Input Registers

Read one or more registers. In this implementation, it does the same as function 03 (Read Holding Registers).

2.2.3 06 (06_h): Write Single Register

This function writes to a single (16-bit) register. For writing a 32-bit value, use function 16 (Write Multiple Registers).

2.2.4 08 (08_h): Diagnostics

Use the diagnostic function to read the error counters or to reset the module. The following subfunctions are supported:

- 00_h: Return query data.
- 01_h: Restart communications option. With request data field set to FF_h 00_h, the module is reset. Setting the request data field to 00_h 00_h just resets the communication.
- 04_h: Force listen only mode. Use subfunction 01_h to terminate listen only mode.
- $0A_h$: Clear counters and diagnostic register.
- $0B_h$: Return bus message count.
- $0C_h$: Return bus communication error count.

- $0D_h$: Return bus exception error count.
- $0E_h$: Return server message count.
- $0F_h$: Return server no response count.
- 10_h: Return server NAK count.
- 11_h: Return server busy count.
- 12_h: Return bus character overrun count.

2.2.5 11 ($0B_h$): Get Comm Event Counter

Returns a status word (which is always 0 with the TMCM-1290 module) and the communication event counter. By fetching the current count before and after a series of messages, a client can determine whether the messages are handled normally by the server.

2.2.6 16 (10_h): Write Multiple Registers

This functions writes to a set of registers. Use this also to write 32-bit values.

2.2.7 17 (11_h): Report Server ID

Returns the module type (first two bytes of the reply) and the firmware version (third and fourth byte of the reply).

2.2.8 22 (16_h): Mask Write Register

Use this function to manipulate single bits of a 16-bit value by using an AND mask and an OR mask. All bits set to 1 in the AND mask are not changed, and all bits set to 0 in the AND mask take the value of the corresponding bit in the OR mask.

2.2.9 23 (17_h): Read/Write Multiple Registers

This function first writes a set of registers and then reads another set of registers.

3 Modbus Registers

3.1 Registers 0…511: TMCL Axis Parameters

All TMCL axis parameters (refer to the *TMCM-1290 TMCL Firmware Manual* for a list of all supported axis parameters) are mapped to Modbus registers 0 to 511. As most TMCL axis parameters are 32-bit values, each axis parameter is mapped to a pair of Modbus regsters. The first register of each pair is $0 + 2 \times Axis\_Parameter$. The following table shows some mapping examples.

Modbus Axis Parameter Mapping	
TMCL Axis Parameter	Modbus Register
0	0 and 1
1	2 and 3
2	4 and 5
…	…
…	…
…	…
255	510 and 511

Table 2: Modbus Axis Parameter Mapping

To read an axis parameter, read the corresponding Modbus register pair. To write to an axis parameter, write to the corresponding Modbus register pair. The first register of a register pair always contains the lower 16 bits of a 32-bit value.

3.2 Registers 512…1023: TMCL User Variables

All TMCL user variables (refer to the *TMCM-1290 TMCL Firmware Manual*) are mapped to Modbus registers 512…1023. As all TMCL user variables are 32-bit values, each user variable is mapped to a pair of Modbus registers. The first register of each pair is $512+2\times User\_Variable$. The following table shows some mapping examples.

Modbus User Variable Mapping	
TMCL User Variable	Modbus Register
0	512 and 513
1	514 and 515
2	516 and 517
…	…
…	…
…	…
255	1022 and 1023

Table 3: Modbus User Variable Mapping

To read a user variable, read the corresponding Modbus register pair. To write to a user variable, write to the corresponding Modbus register pair. The first register of a register pair always contains the lower 16 bits of a 32-bit value.

3.3 Register 1024: General Purpose Outputs

Each bit of register 1024 controls one general purpose output. The outputs are mapped to the register bits as follows.

Digital Outputs in Register 1024	
Port	Bit
GPIO0	0
GPIO1	1
GPIO2	2

3.4 Register 1025: General Purpose Inputs

Each bit of register 1025 shows the state of one general purpose input. The inputs are mapped to the register bits as follows.

Digital Inputs in Register 1025	
Port	Bit
GPIO0	0
GPIO1	1
GPIO2	2

3.5 Register 1026...1035: Analog Inputs

Modbus registers 1026...1035 contain the values of the analog inputs. They are mapped as follows.

Analog Inputs		
Port	Register	Range/Units
AIN0	1026	0...4095
Voltage	1034	[0.1V]
Temperature	1035	[°C]

3.6 Register 1280/1281: Move Absolute

Writing a 32-bit signed target position to register pair 1280/1281 executes an absolute move (like the MVP ABS command in TMCL) using the target position written to this register pair. Ramp parameters like acceleration, deceleration, and maximum positioning speed can be set using the corresponding axis parameters (which are mapped to Modbus registers 1...512).

3.7 Register 1282/1283: Move Relative

Writing a 32-bit signed target position to register pair 1282/1283 executes a relative move (like the MVP REL command in TMCL) using the target position written to this register pair. Ramp parameters like acceleration, deceleration, and maximum positioning speed can be set using the corresponding axis parameters (which are mapped to Modbus registers 1...512).

3.8 Register 1284/1285: Rotate

Writing a 32-bit signed target velocity to register pair 1284/1285 puts the drive into velocity mode and accelerates/decelerates the motor to the given target speed.

3.9 Register 1286: Hard Stop

Writing zero to register 1286 immediately stops the motor, without ramp.

3.10 Register 1287: Reference Search Start/Stop

Writing zero to register 1287 starts a reference search. Writing 1 to this register aborts a running reference search. The reference search can be configured using the corresponding axis parameters (refer to the *TMCM-1290 TMCL Firmware Manual* for more about the reference search).

3.11 Register 1288: Reference Search Status

This read-only register shows the status of a running reference search. It reads zero when the reference search is not running (or finished). A value greater than zero shows that a reference search is in progress.

3.12 Register 1296...1302: TMCL Tunnel

Using these registers, TMCL commands can be executed directly. The registers are mapped as follows:

- Register 1296: TMCL command opcode
- Register 1297: Type parameter (MSB) and Motor/Bank parameter (LSB)
- Register pair 1298/1299: Value parameter (32-bit signed)
- Register pair 1300/1301: Result (32-bit signed)
- Register 1302: Status

Write to registers 1296...1299 using just one **Write Multiple Registers** command. After the command is executed, register pair 1300/1301 contains the result of the command and register 1302 contains the status code.

3.13 Register 1304: Firmware Version

The MSB of this read-only register contains the major version number. The LSB contains the minor version number.

3.14 Register 1500: I/O Pull-up Resistors

Use this register to control the switchable pull-up resistors on GPIO0, GPIO1, and GPIO2. Each bit controls one pull-up resistor. Setting a bit to 1 switches on the pull-up resistor.

Pull-up Resistors	
Port	Bit
GPIO0	0
GPIO1	1
GPIO2	2

3.15 Register 1501: I/O Mode

This register contains a bit vector that selects the input or output mode for each general purpose input/output. Setting a bit to 0 selects the input mode and setting a bit to 1 selects the output mode.

I/O Mode	
Port	Bit
GPIO0	0
GPIO1	1
GPIO2	2

3.16 Register 1536: Server Address

This register sets the Modbus server address. The server address can be a value between 1 and 247. The factory default setting is 1. Changing this register does not take effect immediately. First, the new setting has to be stored using register 1538. The new setting then becomes effective after the next reset.

3.17 Register 1537: Baudrate

Use this register to set the baudrate of the serial interface. Changing this register does not take effect immediately. First, the new setting has to be stored using register 1538. The new setting then becomes effective after the next reset. The following baud rates can be set:

Serial Baudrate Settings	
Register 1537	Baudrate
0	9600
1	14400
2	19200
3	28800
4	38400
5	57600

Register 1537	Baudrate
6	76800
7	115200
8	230400
9	250000
10	500000
11	1000000

Table 4: Serial Baudrate Settings

The factory default setting is 115200bps.

3.18 Register 1538: Store Communication Settings

Writing the magic number 1234_h permanently stores any changes made to registers 1536 and 1537. This is necessary to make any changes of the communication parameters become effective after the next reset.

3.19 Register 65280 ($FF00_h$): Enter Bootloader Mode

Writing the magic number 1234_h to this register switches the module to bootloader mode. This is only necessary for performing a firmware update. Reading from this register is not possible.

4 Tables Index

5 Supplemental Directives

5.1 Producer Information

5.2 Copyright

5.3 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 ADI Trinamic or by other manufacturers, whose products are used or referred to in combination with ADI Trinamic's products and ADI Trinamic's product documentation.

This Modbus Firmware Manual 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.

5.4 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.

5.5 Disclaimer: Life Support Systems

ADI Trinamic/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 ADI Trinamic/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.

5.6 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 ADI Trinamic's/Trinamic Motion Control GmbH & Co. KG specific written consent.

ADI Trinamic/Trinamic Motion Control GmbH & Co. KG 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 ADI Trinamic/Trinamic Motion Control GmbH & Co. KG.
ADI Trinamic/Trinamic Motion Control GmbH & Co. KG conveys no patent, copyright, mask work right or other trademark right to this product. ADI Trinamic/Trinamic Motion Control GmbH & Co. KG assumes no liability for any patent and/or other trademark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

5.7 Collateral Documents and 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.analog.com.

6 Revision History

6.1 Firmware Revision

Version	Date	Description
V1.02	03/24	Initial version

Table 5: Firmware Revision

6.2 Document Revision

Version	Date	Description
Rev 0	02/25	Initial version

Table 6: Document Revision