Modbus function codes description

for Nano/Pico/Micro/27E positioner

Preface

This document requires at least basic knowledge of Modbus protocol and Modbus over serial line. If there is any misunderstandings, refer to the following web page:

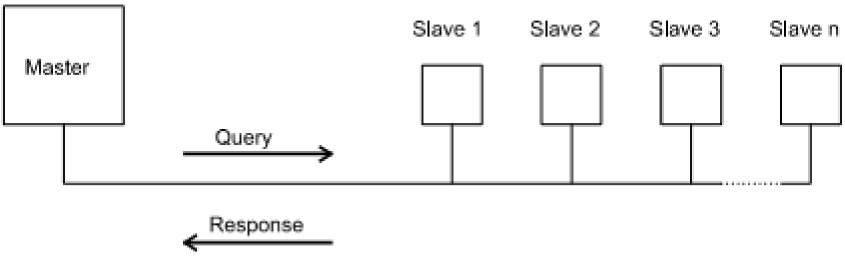
www.modbus.org

and document:

www.modbus.org/docs/Modbus_over_serial_line_V1.pdf

Modbus properites

Mode:	RTU (remote terminal unit)
Physical layer:	RS485, 19200 bps, 8bits, even parity, 1 stop
Addressing:	Valid addresses are 1-64. 0 is broadcast address (don't use it for changing device address when multiple devices are connected to bus).



Query
Response

Address field	Function code	Data	CRC (or LRC)

Overview

Protocol has three data types UByte8, Int32 and Float32(IEEE754).
Data formating is big endian.
Pico and 27E support only motor A.
Micro and 27E positioners don't support folowing function codes :

- Read/Write motor stop time A/B (0x90, 0x92, 0x91, 0x93)
- Read/Write endswitch error detect start position (0xF7, 0xF9, 0xF8, 0xFA)
- Write homing timeout (0x6F)
- Write U supply measurement factor (0x63)
- Write I motor measuring factor (0x65)
- Write I motor detection current (0x80, 0x82)

ACK – ackowledge code

According to Modbus protocol, Slave must reply on every Master's string. If command can not be executed due to different reasons (but was correctly recepted), error ACK code is sent back as a reply.

- if OK, reply contains equal function code byte (0xxx xxxx) + »ok« data byte
- if ERR, reply contains function code byte with MSB bit set to '1' (1xxx xxxx), due to protocol + error message byte, which describe what is wrong.

ACK=OK	ACK byte:	0x00	function code was executed
ACK=UNRECOGNIZED_COMMAND	ACK byte:	0x01	function code does not exist
ACK=VALUE_OUT_OF_LIMIT	ACK byte:	0x02	data was not accepted, out of limit
ACK=NOT_USED_DURING_REF	ACK byte:	0x03	function code can not be used now

Function codes

General information

Read tracker status

Function code: 0x20
Data: None

Reply: Function code: 0x20 or 0xA0
Data: 8 bytes:

- Int32 – Status
- Int32 – Extended status
-

Clear tracker error status

Function code: 0x21
Data: None

Reply: Function code:0x21 or 0xA1
Data: 1 byte – ACK

Set address

Slave responds with old address!

Function code: 0x22
Data: UByte8

Reply: Function code:0x22 or 0xA2
Data: 1 byte – ACK

Set address by serial

To be used with broadcast addres 0

Function code: 0x79
Data: Int32 – SN1

Reply: Function code:0x22 or 0xA2
Data: 1 byte – ACK

Read Usupply

Function code: 0x23
Data: None

Reply: Function code: 0x23 or 0xA3
Data: Float32[V]

Read Imotor

Function code: 0x24

Reply: Function code: 0x24 or 0xA4

Data: None

Data: Float32[A]

Read serial numbers

Function code: 0x25
Data: None

Reply: Function code: 0x20/0xA0
Data: 16 bytes:

- Int32 - SN1
- Int32 - SN2
- Int32 - SN3
- Int32 - SN4

Read software version

Nano responds with 3.xxx and pico with 2.xxx

Function code: 0x26
Data: None

Reply: Function code: 0x20/0xA0
Data: Float32 – application version

Read bootloader version

Function code: 0x29
Data: None

Reply: Function code: 0x20/0xA0
Data: 16 bytes:

- Int32 - version
- Int32 - Devtype(2=pico,3=nano)
- Int32 - Hw revision
- Int32 – Minimum application version

Read events

Events are cleared after readout

Function code: 0x2A
Data: None

Reply: Function code: 0x20/0xA0
Data: Int32 – events:

- Bit0 – Homing A finished
- Bit1 – Homing B finished
- Bit2 – Resistance measurement complete
- Bit3 – Resistance measurement canceled

Read error A/B vars

Last recorded values when error occur

Function code: 0x2B,0x2C
Data: None

Reply: Function code: 0x20/0xA0
Data: 32 bytes:

- Float32 - Current[I]
- Float32 - Voltege[V]
- Float32 – Position[mm/deg]
- Int32 – not used
- Int32 – not used
- Int32 – not used
- Int32 – not used
- Int32 – not used

Fast read general info

Function code: 0x78
Data: None

Reply: Function code: 0x20/0xA0
Data: 104 bytes:

- Float32 – Application version
- Int32 – Status
- Int32 – Extended status
- Int32 – Devtype(2=pico, 3=nano)
- Int32 – Bootlader version
- Int32 – CRC error count
- Int32 – not used
- Float32 – Supply voltage[V]
- Float32 – Motor current[I]
- Float32 – Remain position A[mm/deg]
- Float32 – Motor position A[mm/deg]
- Float32 – Motor destination A[mm/deg]
- Float32 – Motor current limit A[A]
- Int32 – not used

- Int32 – not used
- Int32 – not used
- Int32 – not used
- Float32 – Remain position B[mm/deg]
- Float32 – Motor position B[mm/deg]
- Float32 – Motor destination B[mm/deg]
- Float32 – Motor current limit B[A]
- Int32 – not used
- Int32 – not used
- Int32 – not used
- Int32 – not used
- Int32 – not used

Status description

Status byte is 4 bytes long (32 flags). It holds various states of tracker condition. Others are described below:

Note: ERR_ means error flag in status register. In this case the red led will light and moving is disabled until staus register is cleared. It goes for a mechanical problem. SF_ means status flag, which just inform the user about internal functions which are in operation.

Flag name:	*Bit Nr.*	*Description:*
---- errors ----		
ERR_OVERCURRENT_MOTOR_A	*0*	*Motor A has exceed Imotor limitation for 5 times*
ERR_HALL_A	*1*	*There is no position feedback signal A (after 5 tries), but motor current is present.*
ERR_TOOLONG_REF_A	*2*	*Moving Ato reference exceed maximum time. See »reference timeout« parameter.*
ERR_CABLE_A	*3*	*While moving there is no current nor feedback signal A. Possible cause is in disconnected cable.*
ERR_OVERCURRENT_MOTOR_B	*4*	*Motor B has exceed Imotor limitation for 5 times*
ERR_HALL_B	*5*	*There is no position feedback signal B (after 5 tries), but motor current is present.*
ERR_TOOLONG_REF_B	*6*	*Moving B to reference exceed maximum time. See »reference timeout« parameter.*
ERR_CABLE_B	*7*	*While moving there is no current nor feedback signal B. Possible cause is in disconnected cable.*
SF_HALL_WIRING_ERROR_A	*23*	*Hall sensor count A !=B*
SF_HALL_WIRING_ERROR_B	*24*	*Hall sensor count A !=B*
---- status flags ----		
SF_POWER_FAILURE	*8*	*Positioner reset has occured from the last status clear*
SF_BUTTON_PRESSED	*9*	*Button was pressed from last status clear*
SF_NO_MODBUS	*10*	*MODBUS timeout occured from the last status clear*
SF_MOVING_OUT_A	*11*	*Motor A is moving out*
SF_MOVING_IN_A	*12*	*Motor A is moving in*
SF_MOVING_REF_CLR_A	*13*	*Motor A is executing command »go to reference – with clear position«.*
SF_MOVING_REF_NOCLR_A	*14*	*Motor A is executing command »go to reference – without clear position«*
SF_MOVING_OUT_B	*15*	*Motor B is moving out*
SF_MOVING_IN_B	*16*	*Motor B is moving in*
SF_MOVING_REF_CLR_B	*17*	*Motor B is executing command »go to reference – with clear position«.*
SF_MOVING_REF_NOCLR_B	*18*	*Motor B is executing command »go to reference – without clear position«*
SF_ENDSW_A_LO_PRESSED	*19*	*end switch pressed A - LO*
SF_ENDSW_A_HI_PRESSED	*20*	*end switch pressed A - HI*
SF_ENDSW_B_LO_PRESSED	*21*	*end switch pressed B - LO*

SF_ENDSW_B_HI_PRESSED	*22*	*end switch pressed B – HI*

Extended status description

Extended status byte is 4 bytes long (32 flags). It holds various states of tracker condition. Others are described below:

---- status flags ----

ESF_MOVE_OUT_ERR_C	*26*	*'1' if motor was moving out when error occured otherwise '0'*
ESF_MOVE_OUT_ERR_B	*27*	*'1' if motor was moving out when error occured otherwise '0'*
ESF_MOVE_OUT_ERR_A	*28*	*'1' if motor was moving out when error occured otherwise '0'*

---- warning flags ---

EFS_VOLTAGE_TO_LOW	*18*	*Voltage was under 17V when trying to move motors*
EFS_LINE_RES_MEASURING	*22*	*Measuring resistance*
EFS_MOTOR_CUTOFF	*23*	*Voltage under 17V when motor was moving(motor is stoped)*
EFS_LOCKED	*25*	*Positioner is locked (will not move)*
EFS_UNDERVOLTAGE	*29*	*Voltage lower than 20V*
EFS_OVERVOLTAGE	*30*	*Voltage is higher than 32V*
EFS_BUTTON_STUCK	*31*	*One or both buttons on positioner is/are stucked*

---- error flags ----

EFS_END_SWB_FAIL	*19*	*End switch was pressed when position > End switch error detect*
EFS_END_SWA_FAIL	*20*	*End switch was pressed when position > End switch error detect*
EFS_LINE_RESISTANCE_HIGH	*21*	*Line Resistance to high*

Motor positioning

Write destination A,B

Function code: 0x35,0x36
Data: Float32[mm/deg]

Reply: Function code:0x31 or 0xB1
Data: 1 byte – ACK

Go to reference A,B

Function code: 0x37,0x38
Data: None

Reply: Function code:0x31 or 0xB1
Data: 1 byte – ACK

Stop motor

Function code: 0x30
Data: None

Reply: Function code:0x30 or 0xB0
Data: 1 byte – ACK

Read enabled motors

Function code: 0x39
Data: None

Reply: Function code:0x31 or 0xB1
Data: Int32 – Axis status:

- Bit0 – A motor enabled
- Bit1 – B motor enabled

Write enable/disable motors

Function code: 0x40
Data: Int32 – Enable/Disable motor:

- Bit0 – Select motor A
- Bit1 – Select motor B
- Bit16 – Enable/Disable motor A
- Bit17 – Enable/Disable motor B

Reply: Function code:0x31 or 0xB1
Data: 1 byte – ACK

Read hall invert

Function code: 0x39
Data: None

Reply: Function code:0x31 or 0xB1
Data: Int32 – Hall invert state:

- Bit0 – Motor A halls inverted
- Bit1 – Motor B halls inverted

Write hall invert

Function code: 0x40
Data: Int32 - Hall invert state:

- Bit0 – normal(0)/invert(1) Motor A halls
- Bit1 – normal(0)/invert(1) Motor B halls

Reply: Function code:0x31 or 0xB1
Data: 1 byte – ACK

Configuration

Read min range A/B

Function code: 0x50,0x52	Reply:	Function code: 0x50 or 0xD0
Data: None	Data:	Float32[mm/deg]

Write min range A/B

Function code: 0x51,0x53	Reply:	Function code: 0x51 or 0xD1
Data: Float32[mm/deg]	Data:	1 byte – ACK

Read max range A/B

Function code: 0x54,0x56	Reply:	Function code: 0x54 or 0xD4
Data: None	Data:	Float32[mm/deg]

Write max range A/B

Function code: 0x55,0x57	Reply:	Function code: 0x55 or 0xD5
Data: Float32[mm/deg]	Data:	1 byte – ACK

Read zero offset A/B

Function code: 0xFB,0xFD	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Float32[mm/deg]

Write zero offset A/B

Function code: 0xFC,0xFE	Reply:	Function code: 0x55 or 0xD5
Data: Float32[mm/deg]	Data:	1 byte – ACK

Read gear ratio A/B

Function code: 0x5C,0x5E	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Float32[imp/(mm or deg)]

Write gear ratio A/B

Function code: 0x5D,0x5F	Reply:	Function code: 0x55 or 0xD5
Data: Float32[imp/(mm or deg)]	Data:	1 byte – ACK

Read motor stop time A/B

Function code: 0x90,0x92	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Float32[ms]

Write motor stop time A/B

Function code: 0x91,0x93	Reply:	Function code: 0x55 or 0xD5
Data: Float32[ms]	Data:	1 byte – ACK

Read I motor limitation A/B

Function code: 0x58,0x5A	Reply:	Function code: 0x58 or 0xD8
Data: None	Data:	Float32[A]

Write I motor limitation A/B

Function code: 0x59,0x5B	Reply:	Function code: 0x59 or 0xD9
Data: Float32[A]	Data:	1 byte – ACK

Read I motor inrush ratio A/B

Inrush ratio is multiplied with I motor limitation to get max. inrush current

Function code: 0x71,0x73	Reply:	Function code: 0x58 or 0xD8
Data: None	Data:	Float32

Write I motor inrush ratio A/B

Function code: 0x70,0x72	Reply:	Function code: 0x59 or 0xD9
Data: Float32	Data:	1 byte – ACK

Read I motor inrush time A/B

Function code: 0x75,0x77	Reply:	Function code: 0x58 or 0xD8
Data: None	Data:	Float32[ms]

Write I motor inrush time A/B

Function code: 0x74,0x76	Reply:	Function code: 0x59 or 0xD9
Data: Float32[ms]	Data:	1 byte – ACK

Read I motor detection current A/B

Function code: 0x81,0x83	Reply:	Function code: 0x58 or 0xD8
Data: None	Data:	Float32[A]

Write I motor detection current A/B

Function code: 0x80,0x82	Reply:	Function code: 0x59 or 0xD9
Data: Float32[A]	Data:	1 byte – ACK

Read endswitch error detect start position A/B

Function code: 0xF7,0xF9	Reply:	Function code: 0x58 or 0xD8
Data: None	Data:	Float32[mm/deg]

Write endswitch error detect start position A/B

Function code: 0xF8,0xFA	Reply:	Function code: 0x59 or 0xD9
Data: Float32[mm/deg]	Data:	1 byte – ACK

Read modbus timeout position A/B

Function code: 0x66,0x68	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Float32[mm/deg]

Write modbus timeout position A/B

Function code: 0x67,0x69	Reply:	Function code: 0x55 or 0xD5
Data: Float32[mm/deg]	Data:	1 byte – ACK

Read modbus timeout

Function code: 0x6A	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Int32[s]

Write modbus timeout

Function code: 0x6B	Reply:	Function code: 0x55 or 0xD5
Data: Int32[s]	Data:	1 byte – ACK

Read modbus timeout ID delay

*Modbus timeout ID delay * Slave Address is added to modbus timeout*

Function code: 0x6C	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Int32[s]

Write modbus timeout ID delay

Function code: 0x6D	Reply:	Function code: 0x55 or 0xD5
Data: Int32[s]	Data:	1 byte – ACK

Read homing timeout

Function code: 0x6E	Reply:	Function code: 0x55 or 0xD5
Data: None	Data:	Int32[s]

Write homing timeout

Function code: 0x6F	Reply:	Function code: 0x55 or 0xD5
Data: Int32[s]	Data:	1 byte – ACK

Read U supply measuring factor

Function code: 0x62	Reply:	Function code: 0x62 or 0xE2
Data: None	Data:	Float32(Micro default=28)

Write U supply measurement factor

Function code: 0x63	Reply:	Function code: 0x63 or 0xE3
Data: Float32	Data:	1 byte – ACK

Read I motor measuring factor

Function code: 0x64	Reply:	Function code: 0x64 or 0xE4
Data: None	Data:	Float32(Micro default=28)

Write I motor measuring factor

Function code: 0x65	Reply:	Function code: 0x65 or 0xE5
Data: Float32	Data:	1 byte – ACK

Sat-Control d.o.o.
Poženik 10
SI-4207 Cerklje an Gorenjskem

Tel:	+386 4 28 162 19
Fax:	+386 4 28 162 12
Email:	development@sat-control.si