Dotpeenator®

BY SERMATEK

INTEGRATION GUIDE · MODBUS TCP

Modbus Register Guide.

Register map and frame-level examples for connecting Dotpeenator® dot peen marking machines to PLCs and host computers over Modbus TCP: file selection, mark content, position and size, marking control and I/O.

PROTOCOL	PORT	UNIT ID	ENCODING
Modbus TCP	502	1	UTF-8

In status names and event identifiers (for example `Printing`, `PrintFinished`, `OnPrintStart`), the word *print* refers to marking on the Dotpeenator® machine. These identifiers are shown exactly as the controller uses them.

1 Overview

The Dotpeenator® marking machine communicates with external master stations (PLCs, host computers, etc.) via the Modbus protocol. The machine (slave) acts as a Modbus server and provides four data areas for the master to read and write:

DATA AREA	TYPE	ADDRESS RANGE	QUANTITY	ACCESS
Coils	Bit	0 ~ 8	9	Read/Write
Discrete Inputs	Bit	0 ~ 11	12	Read only
Input Registers	16-bit word	0 ~ 9	10	Read only
Holding Registers	16-bit word	0 ~ 65534	65535	Read/Write

DEFAULT COMMUNICATION PARAMETERS

PARAMETER	VALUE
Protocol	Modbus TCP
Port	502
Slave address (Unit ID)	1

PARAMETER	VALUE
Character encoding	UTF-8

FUNCTION CODES

FUNCTION CODE	OPERATION	DATA AREA
`0x01`	Read Coils	Coils
`0x02`	Read Discrete Inputs	Discrete Inputs
`0x03`	Read Holding Registers	Holding Registers
`0x04`	Read Input Registers	Input Registers
`0x05`	Write Single Coil	Coils
`0x06`	Write Single Holding Register	Holding Registers
`0x0F`	Write Multiple Coils	Coils
`0x10`	Write Multiple Holding Registers	Holding Registers

2 Register Modes

The controller supports two string storage modes for Holding Registers, selectable in the Modbus module settings window of the controller software.

MODE	DESCRIPTION
Direct (single-byte)	Each byte is stored in one register using only the low 8 bits; a zero-value register terminates the string
Packed (two-byte)	Every two bytes are packed into one register (first byte in the low 8 bits, second byte in the high 8 bits); a zero-value register terminates the string

Both modes use the same register address layout; only the way strings are encoded inside the registers differs. The default is **Direct** mode, and this document mainly uses that mode in its explanations.

3 Holding Register Address Map

Holding Registers are the core data area, used for file operations, marking control and reading/writing mark data.

ADDRESS RANGE	PURPOSE	REGISTERS
0 ~ 99	File name (without extension)	100
100	Reserved	1
101	Marking control command	1
102 ~ 999	Reserved	898
1000 ~ 1099	Mark 0 data area	100
1100 ~ 1199	Mark 1 data area	100

ADDRESS RANGE	PURPOSE	REGISTERS
…	…	…
1000+i×100 ~ 1099+i×100	Mark i data area	100
…	…	…
10900 ~ 10999	Mark 99 data area	100
11000 ~ 65534	Reserved	54535

LAYOUT OF EACH MARK DATA AREA (100 REGISTERS)

OFFSET	ADDRESS (MARK I)	PURPOSE	REGISTERS
+0 ~ +91	1000+i×100 ~ 1091+i×100	Mark content (text)	92
+92 ~ +93	1092+i×100 ~ 1093+i×100	X position (CDAB float)	2
+94 ~ +95	1094+i×100 ~ 1095+i×100	Y position (CDAB float)	2
+96 ~ +97	1096+i×100 ~ 1097+i×100	Width W (CDAB float)	2
+98 ~ +99	1098+i×100 ~ 1099+i×100	Height H (CDAB float)	2

4 Direct Mode (Single-Byte) in Detail

4.1 Mark Content Registers

> **Address:** `1000+i×100 ~ 1091+i×100` (92 registers per mark)

In Direct mode, each byte of the string is stored in the low 8 bits of one register; the high 8 bits are always zero. A zero-value register must follow the last byte as a terminator.

ENCODING RULES

1. Encode the mark content to a byte array using UTF-8 (or the configured encoding)
2. Store each byte in one register (value range `0x0001 ~ 0x00FF`)
3. Append a zero-value register after the last byte as a terminator
4. Fill the remaining registers with 0

EXAMPLE: MARK 0 CONTENT IS "ABC"

UTF-8 bytes: `0x41 0x42 0x43`

REGISTER ADDRESS	VALUE (HEX)	MEANING
1000	0x0041	'A'
1001	0x0042	'B'
1002	0x0043	'C'
1003	0x0000	Terminator
1004 ~ 1091	0x0000	Unused

Modbus frame example — write mark 0 content "ABC" (FC 0x10)

REQUEST

00 01	00 00	00 0F	01	10	03 E8	00 04	08	00 41	00 42	00 43	00 00
TID	PID	Len	Unit	FC	Addr	Qty	BC	Reg0	Reg1	Reg2	Reg3
	0	15	1	0x10	1000	4	8	'A'	'B'	'C'	Term

RESPONSE

00 01	00 00	00 06	01	10	03 E8	00 04
TID	PID	Len	Unit	FC	Addr	Qty
	0	6	1	0x10	1000	4

EXAMPLE: MARK 1 CONTENT IS "HELLO"

UTF-8 bytes: `0x48 0x65 0x6C 0x6C 0x6F` . Mark 1 base address = 1000 + 1×100 = **1100**.

REGISTER ADDRESS	VALUE (HEX)	MEANING
1100	0x0048	'H'
1101	0x0065	'e'
1102	0x006C	'l'
1103	0x006C	'l'
1104	0x006F	'o'
1105	0x0000	Terminator
1106 ~ 1191	0x0000	Unused

Modbus frame example — write mark 1 content "Hello" (FC 0x10)

REQUEST

00 02	00 00	00 13	01	10	04 4C	00 06	0C	00 48	00 65	00 6C	00 6C	00 6F	00 00
TID	PID	Len	Unit	FC	Addr	Qty	BC	Reg0	Reg1	Reg2	Reg3	Reg4	Reg5
	0	19	1	0x10	1100	6	12	'H'	'e'	'l'	'l'	'o'	Term

RESPONSE

00 02	00 00	00 06	01	10	04 4C	00 06
TID	PID	Len	Unit	FC	Addr	Qty
	0	6	1	0x10	1100	6

EXAMPLE WITH MULTI-BYTE CHARACTERS: MARK 0 CONTENT IS "ÇELIK"

UTF-8 bytes: `0xC3 0x87 0x65 0x6C 0x69 0x6B` (6 bytes — "Ç" takes 2 bytes)

REGISTER ADDRESS	VALUE (HEX)	MEANING
1000	0x00C3	UTF-8 byte 1 of 'Ç'
1001	0x0087	UTF-8 byte 2 of 'Ç'
1002	0x0065	'e'
1003	0x006C	'l'
1004	0x0069	'i'
1005	0x006B	'k'
1006	0x0000	Terminator