Modbus® Protocol

Included files:

- Modbus Protocol.qcp
- Modbus CRC.xls

The Modbus® protocol may be implemented in either an ASCII format or RTU format. QuickSilver has implemented the RTU format. The RTU format uses an 8 bit Binary communications, with an idle period between characters serving as the inter-frame indicator. The inter-frame idle period is defined as being at least four (4) character time periods between messages, with the receiving stations being able to consider an idle time of as short as 1.5 character periods as a valid inter-frame gap, while requiring recognition of the inter-frame gap after 3.5 character periods of idle.

The data stream then consists of the Address field (8 bits), the Function field (8 bits), followed by any data required by the selected function, followed by 2 bytes of CRC (cyclic redundancy code) sent low byte, then high byte.

The Modbus protocol also optionally supports both 1 or 2 stop bits and supports odd, even and no parity. QuickSilver supports only no parity and 2 stop bits prior to SD32. SilverDust Revision 32 (SD32) and higher support all combinations of stop bits and parity. See Modicon for detailed description of the Modbus protocol.

Modbus/TCP is supported on some controllers. See QCI-AN028 Modbus TCP.

Modbus® is a registered trademark of Modicon.

Requirements

QuickControl® 4.0 Service Release 2 or higher. Note, QuickControl® 4.6 is recommended to allow programming (i.e. downloading and debugging) using the Modbus protocol.

SilverDust

SilverDust™ firmware revision 02+. NOTE: Additional Modbus functions added on latter revisions. Newer firmware revision requirements denoted on feature by SD nn, where nn specifies revision number.

SilverNugget

SilverNugget™firmware revision 46+, series 7+. See the table below for an example how standard firmware revisions cross to Modbus firmware revisions.

Standard Firmware	Modbus Firmware
46-1	46-7
46-2	46-8
46-3	46-9
46-4	46-A
46-5	46-B

The SilverNugget firmware revisions supporting Modbus do not support the Position Compare (PCP) command or the QCI 9-Bit Binary protocol.

To specify the Modbus configuration at time of order, specify an 'M' in the "Controller" section of the part number. For example:

Standard Configurations (non-Modbus configuration)
QCI-N2-E3-04-EE
QCI-N2-E1-01-BB04

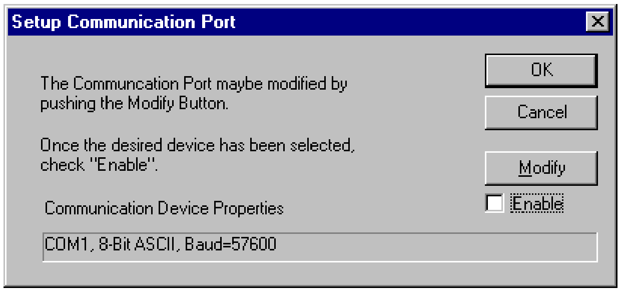
Modbus Configurations
QCI-N2-M3-04-EE
QCI-N2-M1-01-BB04

Configuring SilverDust (SD31) for Modbus®

QuickControl 4.6 supports Modbus as one of its communication protocols thus allowing a device to be programmed and debugged using the Modbus protocol.

Configure QuickControl For Modbus

1) Setup->Comm Port



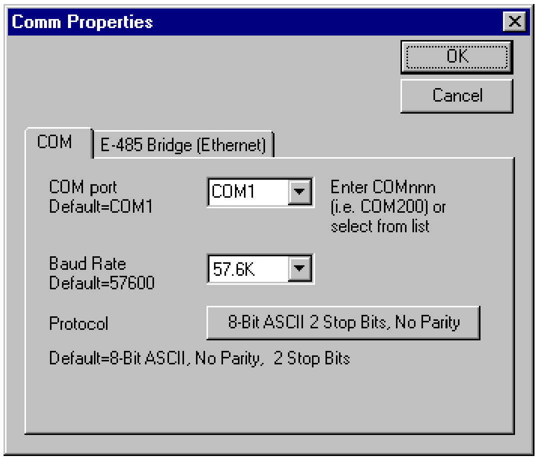
The Communcation Port maybe modified by pushing the Modify Button.

Once the desired device has been selected, check "Enable".

Communication Device Properties: COM1, 8-Bit ASCII, Baud=57600

□ Enable

2) Press Modify



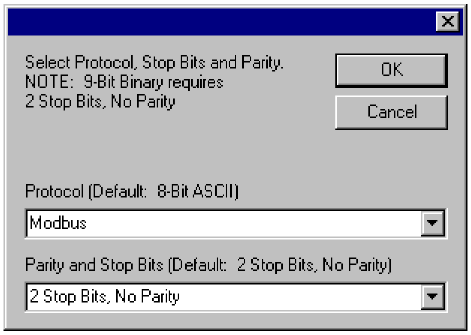
COM port Default=COM1: COM1 — Enter COMnnn (i.e. COM200) or select from list

Baud Rate Default=57600: 57.6K

Protocol: 8-Bit ASCII 2 Stop Bits, No Parity

Default=8-Bit ASCII, No Parity, 2 Stop Bits

3) Select a COM port and press the Protocol button.



Select Protocol, Stop Bits and Parity.
NOTE: 9-Bit Binary requires 2 Stop Bits, No Parity

Protocol (Default: 8-Bit ASCII): Modbus

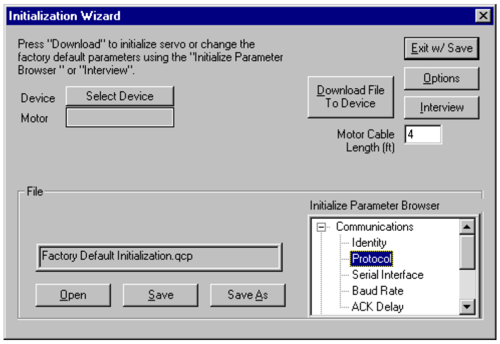
Parity and Stop Bits (Default: 2 Stop Bits, No Parity): 2 Stop Bits, No Parity

4) Select Modbus, 2 Stop Bits, No Parity
Press OK, OK, OK

Initialize Device For Modbus

1) Tools->Initialization Wizard

2) From the Initialize Parameter Browser, select Communications->Protocol


Initialization Wizard

Press "Download" to initialize servo or change the factory default parameters using the "Initialize Parameter Browser " or "Interview".

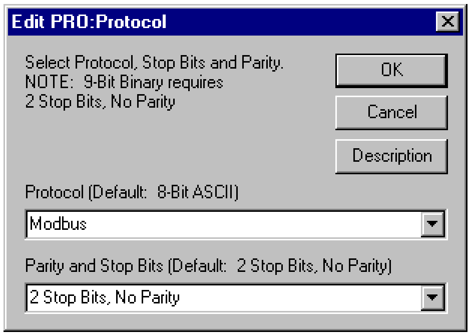
Device: Select Device

Motor:

Motor Cable Length (ft): 4

File: Factory Default Initialization.qcp

Initialize Parameter Browser
- Communications
  - Identity
  - Protocol
  - Serial Interface
  - Baud Rate
  - ACK Delay


Edit PRO:Protocol

Select Protocol, Stop Bits and Parity.
NOTE: 9-Bit Binary requires
2 Stop Bits, No Parity

Protocol (Default: 8-Bit ASCII): Modbus

Parity and Stop Bits (Default: 2 Stop Bits, No Parity): 2 Stop Bits, No Parity

3) Select Modbus, 2 Stop Bits, No Parity. Press OK

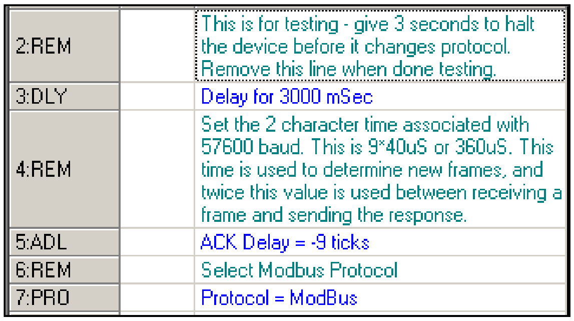
4) Press "Download File…". Most likely, QuickControl will prompt you through the Unknown Device Wizard to establish communications. Follow the prompts. After the initialization program is downloaded, use Save As to save your modified file.

QuickControl will now use Modbus to communicate with the device allowing you to use all of QuickControl's powerful programming and debug features including Control Panel, Register Watch, Single Step, and Trace.

Tip: Use Register Watch's Monitor feature to monitor the serial bus between a host (i.e. HMI) and the device(s). This feature updates the Register Watch registers by passively monitoring the serial bus for read register type commands (i.e Modbus funct 3).

Configuring SilverNugget for Modbus®

Besides the normal initialization commands, there are two commands required to put the QuickSilver device into Modbus protocol. Please see the program "Modbus Protocol.qcp" for an example. The two required commands are ACK Delay (ADL) and Protocol (PRO). An exert from the program is shown on the right.



2:REM		This is for testing - give 3 seconds to halt the device before it changes protocol. Remove this line when done testing.
3:DLY		Delay for 3000 mSec
4:REM		Set the 2 character time associated with 57600 baud. This is 9*40uS or 360uS. This time is used to determine new frames, and twice this value is used between receiving a frame and sending the response.
5:ADL		ACK Delay = -9 ticks
6:REM		Select Modbus Protocol
7:PRO		Protocol = ModBus

1) Run Initialization Wizard to establish communications using the defaults.

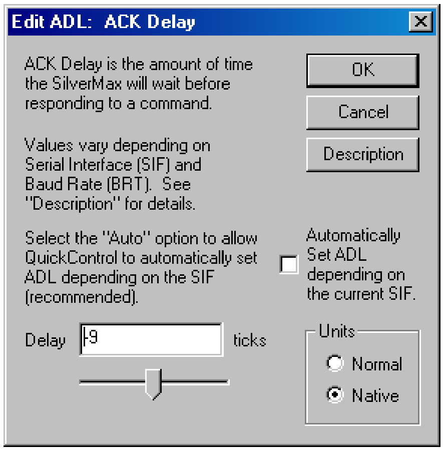
2) Download the provided program “Modbus Protocol.qcp” to configure the device for Modbus.

Please note, once the Modbus program is executed (i.e. the servo restarts), QuickControl will loose communications with the servo because the servo will switch to Modbus protocol. To get QuickControl communications back you can either run the Unknown Device Wizard or modify the Modbus program to switch out of Modbus protocol (PRO command) in response to some external event (i.e. input , register value,..).

SilverLode Commands Used For Modbus

ACK Delay (ADL)

The Modbus Inter-frame idle time is configured on the QuickSilver device by means of the ACK Delay (ADL) command.   NOTE: Normal resolution for the ADL is 120uSec. For a 40uSec resolution, use a negative Delay Count (see ADL for more details).



Edit ADL: ACK Delay

ACK Delay is the amount of time the SilverMax will wait before responding to a command.

Values vary depending on Serial Interface (SIF) and Baud Rate (BRT). See "Description" for details.

Select the "Auto" option to allow QuickControl to automatically set ADL depending on the SIF (recommended).

□ Automatically Set ADL depending on the current SIF.

Delay -9 ticks

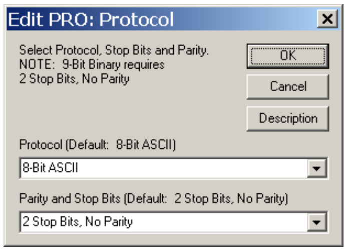
Units: ○ Normal ● Native

Baud	ACK periods	Periods
Rate	x 120uS	x 40uS
300	612	-1834
600	306	-917
1200	153	-459
2400	77	-230
4800	39	-115
9600	20	-58
19200	10	-29
38400	5	-15
57600	4	-10
76800	3	-8
115200	2	-5

The provided table may be used to set the inter-frame idle time. This corresponds to approximately two (2) character periods. Following the receipt of a valid frame, the execution of the command is delayed by two character periods to verify that the frame was, indeed, completed. The response, if any, is delayed a further 2 character periods to provide the required 4 character inter-frame idle period between frames.

Protocol (PRO)

The Modbus protocol is selected by means of the Protocol (PRO) command, with the Mode parameter set to 2. When using QuickControl simply select Modbus as follows.



Edit PRO: Protocol

Select Protocol, Stop Bits and Parity.
NOTE: 9-Bit Binary requires
2 Stop Bits, No Parity

Protocol (Default: 8-Bit ASCII): 8-Bit ASCII

Parity and Stop Bits (Default: 2 Stop Bits, No Parity): 2 Stop Bits, No Parity

NOTE: The Identity (IDT) command is used to set the device Address. Note that only the unit ID is used, as group ID is not supported in Modbus. Note that Modbus broadcast address is address 0 rather than address 255 for the QuickSilver protocols. Valid Modbus addresses are 0 to 247.

NOTE: The SilverLode is currently configured for 1 start bit, 8 data bits, no parity, 2 stop bits. SilverDust Revision 32 and higher support configurable parity and stop bits.

NOTE: Modbus is limited to no higher than 115k bits per second

Modbus Functions Implemented

A limited number of the Modbus Functions are implemented. These include Function 03, read holding registers, and Function 16, preset multiple registers. Revision 31 and higher of SilverDust also includes additional functions 05 (Force Single Coil), 06 (Preset Single Register), 16 (Preset multiple registers) has been extended to allow up to 8 Modbus registers (4 SilverDust Registers) to be updated at a time (limited to register 10-199 for multiple register access), 22 (Mask Write Registers), 23 Read/Write Registers. Note: Funct 23 has also been enhanced to allow any SilverDust command to be accessed via Modbus (see Encapsulated Manufacturer Communication for more details.

Address Translation

The SilverLode registers have been mapped onto the Modbus register set into what is referred to in the Modbus literature as the ***4xxxx*** bank. The SilverLode registers have been mapped such that the low word of Register 0 is mapped to Modbus physical address 1000 or logical address 1001 within the ***4xxxx*** bank.

Modbus requires an offset of one (1) between the logical address presented to users via (most) user interfaces, and the physical address transmitted across the interface. Thus entering a value of 1001 into a Modbus interface or controller will result in an address value of 1000 physically being sent over the interface). The addresses given point to the low 16 bit word of the SilverLode register. The high word of the corresponding SilverLode register is found by adding 1 to the address of the low word.

Information Registers		
	Modbus	Modbus
Register	Physical	Logical
0	1000	1001
1	1002	1003
2	1004	1005
3	1006	1007
4	1008	1009
5	1010	1011
6	1012	1013
7	1014	1015
8	1016	1017
9	1018	1019

User Registers		
	Modbus	Modbus
Register	Physical	Logical
10	1020	1021
11	1022	1023
12	1024	1025
13	1026	1027
14	1028	1029
15	1030	1031
16	1032	1033
17	1034	1035
18	1036	1037
19	1038	1039
20	1040	1041
21	1042	1043
22	1044	1045
23	1046	1047
24	1048	1049
25	1050	1051
26	1052	1053
27	1054	1055
28	1056	1057
29	1058	1059
30	1060	1061
31	1062	1063
32	1064	1065
33	1066	1067
34	1068	1069
35	1070	1071
36	1072	1073
37	1074	1075
38	1076	1077
39	1078	1079
40	1080	1081

Special Registers		
	Modbus	Modbus
Register	Physical	Logical
200	1400	1401
201	1402	1403
202	1404	1405
203	1406	1407
204	1408	1409
205	1410	1411
206	1412	1413
207	1414	1415
208	1416	1417
209	1418	1419
210	1420	1421
211	1422	1423
212	1424	1425
213	1426	1427
214	1428	1429
215	1430	1431
216	1432	1433
217	1434	1435
218	1436	1437
219	1438	1439
220	1440	1441
221	1442	1443
222	1444	1445
223	1446	1447
224	1448	1449
225	1450	1451
226	1452	1453
227	1454	1455
228	1456	1457
229	1458	1459
230	1460	1461

* Note: SilverDust units extend this same addressing for all registers. The same calculation holds:
Physical address = 1000 + 2 * Register number
Logical address = 1001 + 2 * Register number

Read Holding Registers – Function 03

Description

The Read Holding Registers command returns the data in the selected registers to the Modbus host. All of the data to be transferred in the response frame is sampled simultaneously, allowing the changing values of various 32bit SilverLode registers to be captured synchronously. The address field is translated to obtain the corresponding SilverLode register.

The address range referenced by the command is not allowed to cross the SilverLode register boundaries. Single 16 bit word transfers may address either the high or low word of the SilverLode register, while double word transfers must address only the low word of the register so as to only transfer to or from one SilverLode register per command. *

* REV31-1x up to 8 Modbus registers (4 user registers) may be read, as long as they are accessing SilverLode user registers 10 to 199, inclusive. Other registers are still only accessible one at a time.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Read Holding Registers	Modbus 03 (0x03)	Address (Physical)	U16	1000-1099, 1400-1467 (1000-1510 SilverDust)
		Number of 16 bit words	U16	1-2 (1-8 REV31-1x)

Example

Read the information in SilverLode registers 10 (Modbus logical address 1021-1022 physical address 1020-1021.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	3	0x03
Address (High Byte)	U16	1020	0x03
(Low Byte)			0xFC
Number (High Byte)	U16	2	0x00
(Low Byte)			0x02
CRC (Low Byte)	U8		0x07
(High Byte)	U8		0x3E

Note that the CRC is calculated according to the Modbus CRC calculation (see Calculation CRC).

Response
(Assuming Register 10 contains a value of 1000)

Parameter	**Type**	**Decimal**	**Hex**
Unit Address	U8	16	0x10
Function	U8	3	0x03
Byte Count	U8	4	0x04
Low Word (High Byte)	U16	1000	0x03
(Low Byte)			0xE8
High Word(High Byte)	U16	0	0x00
(Low Byte)			0x00
CRC (Low Byte)	U8		0xFB
(High Byte)	U8		0x8C

Preset Multiple Registers – Function 16

Description

The Preset Multiple Registers command allows the Modbus host to write to up to two* 16-bit Modbus registers (one SilverLode 32-bit registers). The write operation is performed to all updated registers simultaneously (no other operation will take place between the actual data transfer), so 32 bit values can be safely transferred.

The address field is translated to obtain the corresponding SilverLode register. The address range referenced by the command is not allowed to cross SilverLode register boundaries.

* REV31-1x allows up to 7 Modbus registers may be written (3 and one half user registers), as long as they are accessing SilverLode user registers 10 to 199, inclusive. Other registers are still only accessible one at a time.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Write Holding Registers	Modbus 16 (0x10)	Address (Physical)	U16	1000-1099, 1400-1467 (1000-1510 SilverDust)
		Number of 16 bit words	U16	1-2
		Byte count of Data	U8	2,4
		Data (instance for each Modbus register)	U16	0 to 65535

Example

Write the information into SilverLode registers 10 (Modbus logical address 1021-1022 physical address 1020-1021).

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	16	0x10
Address (High Byte)	U16	1020	0x03
(Low Byte)			0xFC
Number (High Byte)	U16	2	0x00
(Low Byte)			0x02
Byte Count	U8	4	0x04
Low Word (High Byte)	U16	1000	0x03
(Low Byte)			0xE8
High Word(High Byte)	U16	0	0x00
(Low Byte)			0xE0
CRC (Low Byte)	U8		0xB8
(High Byte)	U8		0xCC

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC).

Response

(assuming Register 10 contains a value of 1000)

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	16	0x10
Address (High Byte)	U16	1020	0x03
(Low Byte)			0xFC
Number (High Byte)	U16	2	0x00
(Low Byte)			0x02
CRC (Low Byte)	U8		0x82
(High Byte)	U8		0xFD

Force Single Coil – Function 5 (SD 26)

Description

The Force Single Coil command allows direct control of I/O via Modbus. An “ON” command sets the bit low, an “OFF” sets the bit high. Note: IO1-7 need to be preconfigured as outputs via the Configure I/O (CIO) prior to the force coil command being received. I/O 101-116 and 201-203 are only available on units supporting these I/O.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Write Holding Registers SN n/a SD 26	Modbus 5 (0x05)	IO number (Physical)	U16	1-7, 101-116, 201-203
		Forced Data	U16	0x0000 = Off 0xFF00=On

Example

Set SilverLode IO 101 ON (Low):

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	5	0x05
Coil	U16	101	0x00
			0x65
Number (High Byte)	U16	0XFF00	0xFF
(Low Byte)		(ON=LOW)	0x00
CRC (Low Byte)	U8		0x9F
(High Byte)	U8		0x64

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC).

Response

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	5	0x05
Coil	U16	101	0x00
			0x65
Number (High Byte)	U16	0XFF00	0xFF
(Low Byte)		(ON=LOW)	0x00
CRC (Low Byte)	U8		0x9F
(High Byte)	U8		0x64

Preset Single Register – Function 6 (SD 26)

Description

The Preset Single Register command allows a single register to be written. Similar to Function 16, but to a single (16 bit) Modbus register.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Write Holding Registers SN n/a SD 26	Modbus 6 (0x06)	Address (Physical)	U16	1000-1510
		Data	U16	0 to 65535

Example

Write a decimal 10 to user Register 30, low word.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	6	0x06
Physical Address	U16	1060	0x04
			0x24
Data (High Byte)	U16	10 = 0x000A	0x00
(Low Byte)			0x0A
CRC (Low Byte)	U8		0x4B
(High Byte)	U8		0xB7

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC). ** = calculated value

Response

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	6	0x06
Physical Address	U16	1060	0x04
			0x24
Data (High Byte)	U16	10 = 0x000A	0x00
(Low Byte)			0x0A
CRC (Low Byte)	U8		0x4B
(High Byte)	U8		0xB7

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC). ** = calculated value

Mask Write Register – Function 22 (SD 26)

Description

The Mask Write register command provides the ability to clear or set any combination of bits in a single 16 bit (Modbus) register. The data in the register is ANDed with the AND mask value, and then ORed with the OR mask value, and then stored back to the register. This may be advantageously used to set and clear bits in the extended IO word, as well as other registers.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Write Holding Registers SN n/a SD 26	Modbus 22 (0x16)	Address (Physical)	U16	1000-1510
		AND mask	U16	0 to 65535
		OR mask	U16	0 to 65535

Example

Clear bit 0 and set bit 1 in lower word of user Register 30 (Modbus 1060 physical address)

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	22	0x16
Physical Address	U16	1060	0x04
			0x24
AND mask (High Byte)	U16	0xFFFE	0xFF
(Low Byte)			0xFE
OR mask (High Byte)	U16	0x0002	0x00
(Low Byte)			0x02
CRC (Low Byte)	U8		0x97
(High Byte)	U8		0x60

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC). ** = calculated value

Response

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	22	0x16
Physical Address	U16	1060	0x04
			0x24
AND mask (High Byte)	U16	0xFFFE	0xFF
(Low Byte)			0xFE
OR mask (High Byte)	U16	0x0002	0x00
(Low Byte)			0x02
CRC (Low Byte)	U8		0x97
(High Byte)	U8		0x60

Read/Write Registers – Function 23 (SD 26)

Description

Read/Write registers implements an atomic read/write operation to the selected registers (16 or 32 bit operation, 32 bit must be aligned to a SilverLode register). The current value of the register is first sampled, and then the forced data is written to the selected register. The sampled value is returned. The Read and Write registers may be the same register or different registers. The read count (1 or 2 for 16 bit or 32 bit operation) is also independent from the write count value.

Function Info

Function Name	Function Type/Num	Parameters	Param Type	Parameter Range
Write Holding Registers SN n/a SD 26	Modbus 23 (0x17)	Read Address (Physical)	U16	1000-1510
		Read Count	U16	1 or 2
		Write Address (Physical)	U16	1000-1510
		Write Count	U16	1 or 2
		Byte Count of Data	U8	2 or 4
		Data (instance for each Modbus write register)	U16	0 to 65535

Example

Read Register 30 (Modbus 1060 physical address), Write 255 to Register 31. Assume Register 31 held 327690.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Read Physical Address	U16	1060	0x04
			0x24
Read Count	U16	2	0x00
			0x02
Write Physical Address	U16	1062	0x04
			0x26
Write Count	U16	2	0x00
			0x02
Byte Count of Data	U8	4	0x04
Write Data (High byte)	U16	0x00FF	0x00
Low Word (Low byte)			0xFF
Write Data (High byte)	U16	0x0000	0x00
High Word (Low byte)			0x00
CRC (Low Byte)	U8		0x98
(High Byte)	U8		0x0A

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC). ** = calculated value

Response

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count	U8	4	0x04
Read Data (High byte)	U16	10	0x00
Low Word (Low byte)			0x0A
Read Data (High byte)	U16	5	0x00
High Word (Low byte)			0x05
CRC (Low Byte)	U8		0x18
(High Byte)	U8		0x27

Note that the CRC is calculated according to the Modbus CRC calculation(see Calculation CRC). ** = calculated value

Encapsulated Manufacturer Communication – Function 23 (SD 31)

Function 23, the Read Register, Write Register command has been specially overloaded to allow the transmission of binary formatted SilverLode commands and the reception of the binary response though the facilities of this common Modbus command.

Function 23 checks the Read Physical Address field for the special address 20803 (20804 logical) and the Write Physical Address field for the address 19780 (19781 logical). The Read Count, which will hold the response to the command, must be between in the range of 0 to 9 words, while the Write Count must be one for the command plus one for each word of parameters (2 for each long parameter). The Byte Count of Data is simply the Write Count doubled.

The user may choose to either always read the maximum 9 word response, or must determine how many words of response are needed for the given command. The first word of the response has two byte fields. The high byte is the number of bytes returned by the SilverLode binary command processor, ranging from 0 (for an acknowledge) to 16, always an odd number. The low byte of the first word the first byte of response, the SilverLode command which is responding. The following data are the response data from the command followed by any extra padding words. The byte count in the upper byte of the first response word should be used to parse the received data, as the extra Read Registers requested beyond those needed to respond contain garbage (unknown) data, as needed to respond with the requested read count.

The binary command embedded is of the same format as that used for the 9-bit binary communications, except that the address and byte count fields are not embedded, but are rather the Modbus address and the Write Register word count fields.

Any SilverLode command may be sent to the unit in this manner, including immediate and program commands. The unit may also thus have programs downloaded via the Modbus. The firmware download procedure, however, will not work over Modbus.

Note: SilverLode Command errors are returned in the SilverLode status words, not as Modbus errors. Only Encapsulation errors (not properly encapsulating the command, such as wrong number of bytes with respect to number of registers, or wrong CRC values, wrong register address, etc.) will be reported as Modbus errors.

Examples: Poll Command (POL)

This is command 0, and may respond with either an ACK, represented by 0 bytes of data (if the status word is zero), or with 1 word of status. As the first embedded response word is the byte count in the upper byte and the poll command in the lower byte , at least two read registers must be selected to get the range of responses. (Note: Poll with Response (POR), command 27, is probably easier to use, as it always returns the status word. See below.)

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Read Physical Address	U16	20803	0x51
			0x43
Read Count	U16	2	0x00
			0x02
Write Physical Address	U16	19780	0x4D
			0x44
Write Count	U16	0	0x00
			0x00
Byte Count of Data	U8	0	0x00
CRC (Low Byte)	U8		0x6E
(High Byte)	U8		0xC1

Response – status word = 0x2000

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count (Modbus)	U8	4	0x04
Bytes returned/ SilverLode Command	U16	03*256+0	0x03
			0x00
Status Word	U16	0x2000	0x20
			0x00
CRC (Low Byte)	U8		0xE1
(High Byte)	U8		0xA2

Response – status word = 0 – Standard ACK response

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count (Modbus)	U8	4	0x04
Bytes returned/ SilverLode Command	U16	0*256+1	0x00
			0x01
Xx – junk data	U16	0xFFFF	0xFF
			0xFF
CRC (Low Byte)	U8		0xA8
(High Byte)	U8		0x56

Example: Poll with Response (POR)

This is command 27, which always responds with 1 word of status. As the first embedded response word is the byte count in the upper byte and the poll command in the lower byte , at least two read registers must be selected to get the range of responses.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Read Physical Address	U16	20803	0x51
			0x43
Read Count	U16	2	0x00
			0x02
Write Physical Address	U16	19780	0x4D
			0x44
Write Count	U16	1	0x00
			0x01
Byte Count of Data	U8	2	0x02
SilverLode Command	U16	27	0x00
			0x1B
CRC (Low Byte)	U8		0x0D
(High Byte)	U8		0xC7

Response – status word

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count (Modbus)	U8	4	0x04
Bytes returned/ SilverLode Command	U16	03*256 + 27	0x03
			0x1B
Status Word	U16	0x0000	0x00
			0x00
CRC (Low Byte)	U8		0x88
(High Byte)	U8		0x65

Example: Velocity Mode Immediate (VMI)

This is command 15, with 32bit Acceleration and Velocity parameters, and two 16 bit stop parameters. This example will ramp to 1000 RPM. The normal response should be an ACK.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Read Physical Address	U16	20803	0x51
			0x43
Read Count	U16	2	0x00
			0x02
Write Physical Address	U16	19780	0x4D
			0x44
Write Count	U16	7	0x00
			0x07
Byte Count of Data	U8	14	0x0E
SilverLode Command	U16	15	0x00
			0x0F
Acceleration	U32	200,000	0x00
			0x03
			0x0D
			0x40
Velocity	U32	1000 RPM * 536871= 536871000	0x20
			0x00
			0x00
			0x58
Stop State	U16	0	0x00
			0x00
Stop Word	U16	0	0x00
			0x00
CRC (Low Byte)	U8		0x38
(High Byte)	U8		0xBF

Response – ACK (Bytes returned = 0)

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count (Modbus)	U8	4	0x04
Bytes returned/ SilverLode Command	U16	00*256 + 27	0x00
			0x0F
Room for NAK, Junk for ACK	U16	0x0003	0x00
			0x03
CRC (Low Byte)	U8		0x88
(High Byte)	U8		0x24

Example – Read Register command (RRG)

This is command 12, which responds with two words of data for each register read, 1 to 4 registers at a time. This example will read current position, register 1. The read count will be 3 registers, one for the byte count/register and two more for the returned data.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Read Physical Address	U16	20803	0x51
			0x43
Read Count	U16	3	0x00
			0x03
Write Physical Address	U16	19780	0x4D
			0x44
Write Count	U16	2	0x00
			0x02
Byte Count of Data	U8	4	0x04
SilverLode Command	U16	12	0x00
			0x0C
Register number	U16	1	0x00
			0x01
CRC (Low Byte)	U8		0x00
(High Byte)	U8		0x01

Response – Register 01 value = 411

Note: The data is returned High word, Low word – the same as in standard binary (9-bit) format.

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	23	0x17
Byte Count (Modbus)	U8	6	0x06
Bytes returned/ SilverLode Command	U16	05*256 + 12	0x05
			0x1C
High Word of Register	U16	0	0x00
			0x00
Low Word of Register	U16	411	0x01
			0x9B
CRC (Low Byte)	U8		0xB1
(High Byte)	U8		0xB5

Error Responses

Timeout recovery

The Modbus protocol specifies that there are to be no responses to broadcast messages, it also specifies that there are to be no responses to frames that are unrecognizable nor those with bad CRC calculations. An incoming frame is also terminated if there is an idle period on the serial communications line exceeding 3.5 character periods. (Note: the detection period is allowed to be as short as 1.5 character times.) The incoming frame is also terminated if additional characters are detected following the frame prior to the 1.5 to 3.5 character silence period that is required at the end of frame. The master must do a time-out recovery if the response is not heard within the expected time period.

Error Codes

If the command frame was recognized as properly formatted, but the commanded unit is unable to perform the requested function as specified, then an Error Response is generated. The error response frame returns the Unit Address, the original Function code with bit 7 also set (that is, the original function code or'ed to 0x80), an error code (specified below), and the CRC.

Exception codes

01 = illegal function
02 = illegal data address
03 = illegal data value
04 = slave device failure
05 = Acknowledge (response delayed)
06 = Slave device busy (try again later)
07 = NAK - unable to process function code 13 or 14 operations
08 = Memory parity error (memory accessed bad)

Example error response for function 16 to an illegal data address

Parameter	Type	Decimal	Hex
Unit Address	U8	16	0x10
Function	U8	128+16=144	0x90
Error Code	U8	02	0x02
CRC (Low Byte)	U8		0x9D
(High Byte)	U8		0xC4

Calculating CRC

The procedure for calculating the CRC is as follows:

The CRC is calculated using a 16 bit register.

```
CRC = 0xFFFF
For each Byte of serial stream
    CRC=CRC XOR  Byte
    Do 8 times
        Shift CRC Right
        If Least Significant Bit (LSB) of CRC = 1
            CRC=CRC XOR 0xA001
        Endif
    Next
Next Byte
```

The resulting checksum is sent or received low byte, then high byte

In C++

```
    // Calculate CRC
    // byteStream[] is a BYTE array holding the Modbus packet of length numBytes
    // count is an int holding the number
    WORD crc = 0xFFFF;
    for(int x=0;x< numBytes;x++){
        crc = crc ^ byteStream[x]; // XOR
        for(int y=0;y<8;y++){
            crc>>1;
            if(crc & 0x1)
                crc = crc ^ 0xA001;
        }
    }
    byteStream [numBytes ++] = LOBYTE(crc);
    byteStream [numBytes ++] = HIBYTE(crc);
```

NOTE: There are also Modbus references on the web that describe a faster calculation method involving the use of pre-computed look-up tables that may speed processing.