




VER 1.0 **Technical Manual**

DS-CLS10-FRS4



Table of Contents

 [Click to return to table of contents](#)

1. Main Specifications.....	3
2. Get Ready.....	4
2.1 Wiring.....	4
3. The Connector Specifies the Table.....	4
3.1 CN1 (power supply).....	4
3.2 CN2 (motor wiring).....	4
3.3 CN3 (holding brake output).....	5
3.4 CN4 (encoder input).....	5
3.5 CN5 (I/O).....	7
3.6 CN6 (IN) / CN7 (OUT)(RS485).....	7
3.7 SW1 (Set the switch).....	8
4. Input Loop Diagram.....	9
5. Output Loop Diagram.....	11
6. LED Indicator.....	12
7. Form Factor (mm).....	14
8. Control Parameters.....	14
8.1 Controller Basic Status (Class 01).....	14
8.2 Basic Parameter Settings (Class 02).....	15
8.3 Closed-Loop Parameter Setting (Class 04).....	15
8.4 Control Parameters (Class 06).....	16
8.5 Input Block Designation (Class 06).....	17
8.6 Output Block Designation (Class 07).....	18
8.7 Multi-Segment Position Mode (Class 08).....	19
9. Message Format.....	23
10. MODBUS Transactions.....	25
10.1 Definition of MODBUS Transactions.....	25
10.2 MODBUS Responds Normally.....	26
10.3 MODBUS Exception Response.....	26
11. Data Encoding.....	26
12. Definition of Public Function Code and Description of Function Code.....	27
12.1 03 (0x03) Read hold registers.....	27
12.2 06 (0x06) Write a single register.....	29
12.3 16 (0x10) Write multiple registers.....	31
13. MODBUS Master Node Working Mode.....	34
14. MODBUS Address Rules.....	34
15. Master/Slave Communication Timing Diagram.....	35
16. RTU Transmission Mode.....	36
17. CRC Check.....	37
18. Line-MODBUS Definition.....	41

1. Main Specifications



Project	Content	Note
Model	DS-CLS10-FRS4	
Input supply voltage	DC 24V~72V	
Maximum output current	6.0A (0-peak)	
Control object motors	2-phase bipolar stepper motor with encoder	
Drive mode	PWM constant current drive	
Communication interface	Input <ul style="list-style-type: none">• Pulse, direction input (configurable as digital input)• Number input 7• Encoder inputs (A, B, Z) Output <ul style="list-style-type: none">• 3 digital outputs• Encode the signal output (Differential A, B, Z)	With the exception of the fixed encoder output, all other inputs/outputs can be freely configured via communication
Enter the details digitally	/SV ON (Servo On) /RESET (Alarm reset) /START (Motor start / stop) /JOG (The motor jogs) /HOME (Back to zero)	
Digital output details	/IN POTISION /ALARM	
LED indication	Status, fault	
Communication I/F	RS485, up to 30 nodes	MODBUS RTU protocol, baud rate: 19200bps (preset) or according to convention
Control method	Location control mode	Positioning according to pulse and RS485 communication
	Speed control mode	Digital instructions
Form factor (mm)	156 (L) × 97 (W) × 33.5 (H)	Terminal blocks are not included
weight	About 376g	Terminal blocks are not included
Operating temperature / humidity	0~45°C, 85%RH or less	Prevents condensation
Save the temperature	-10~70°C, 85% or less	Prevents condensation
Ambient gases	Protection against corrosive gases	

2. Get Ready



* Be sure to do the following before turning on the power.

2.1 Wiring

1. Be sure to make sure that you refer to the Description Connector Designation Table for wiring.
2. **CN1** : power supply
Please use AWG#20 or above.
3. **CN2** : Motor wiring
4. **CN3** : Holding brake output
5. **CN4** : Encoder wiring
6. **CN5** : Wiring of interface signals
Configure the necessary digital input and digital output signals. The general-purpose inputs / outputs are isolated by optocouplers. Please prepare the power supply (+24V) for the interface.



Notes

The encoder signal is differential output and is not isolated with an optocoupler

7. **CN6** : Wiring for RS485 communication Please use RJ45 connectors.
8. **CN7** : Wiring for RS485 communication Please use an RJ45 connector.
9. **SW1** : Eight-step DIP switch, node setting

3. The Connector Specifies the Table



3.1 CN1 (power supply)

Terminal number	Icon	Pin.	Signal name
CN1		2	power supply V+ (DC24V~72V)
		1	power supply GND


Pay attention to the polarity of the power supply when wiring

Wire specifications: AWG20~AWG16 (multi-stranded wire)

3.2 CN2 (motor wiring)

Terminal number	Icon	Pin.	Signal name
CN2		4	Electric machine A+
		3	Electric machine B+
		2	Electric machine A-
		1	Electric machine B-

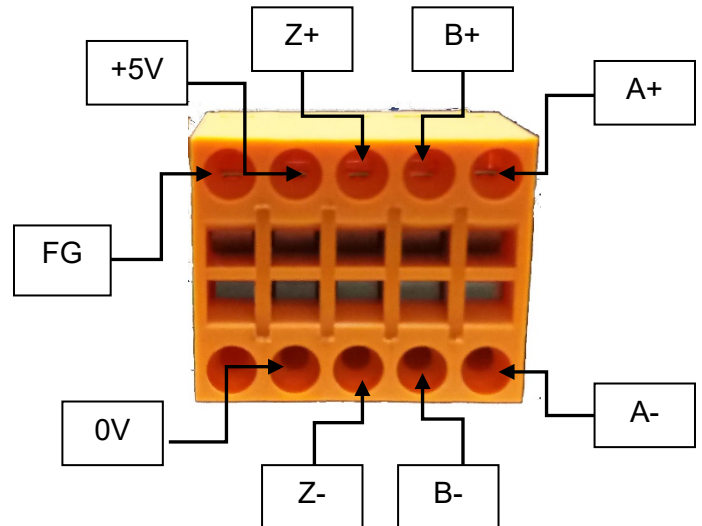
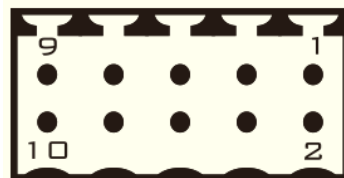
3.3 CN3 (holding brake output)

Terminal number	Icon	Pin.	Signal name
CN3		2	BRK+ brake output is positive
		1	BRK - Negative brake output

3.4 CN4 (encoder input)

Pin.	Signal name	Pin.	Signal name
1	A+	2	A-
3	B+	4	B-
5	Z+	6	Z-
7	+5V	8	0V
9	FG	10	NC

Diagram



Pay attention to the polarity of the encoder power supply when wiring

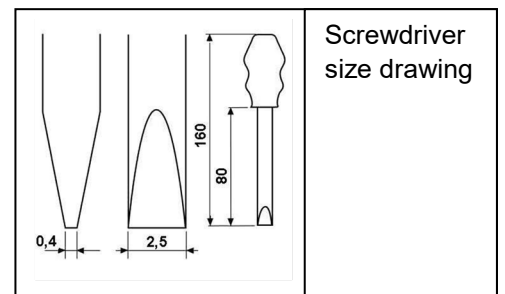
Wire specifications: AWG28~AWG18 (multi-stranded wire)

- The terminals are spring-retracted and front-wired, making it easy to operate with a dedicated screwdriver.

- ♦ Use special tools to tighten the terminal blocks
When tightening the terminals, use a screwdriver with a blade width of 0.4×2.5.

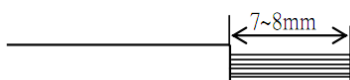


Phoenix Contacts screwdriver
(Product number: 1205037, model SZS 0. 4×2.5)



- ♦ Wiring method:

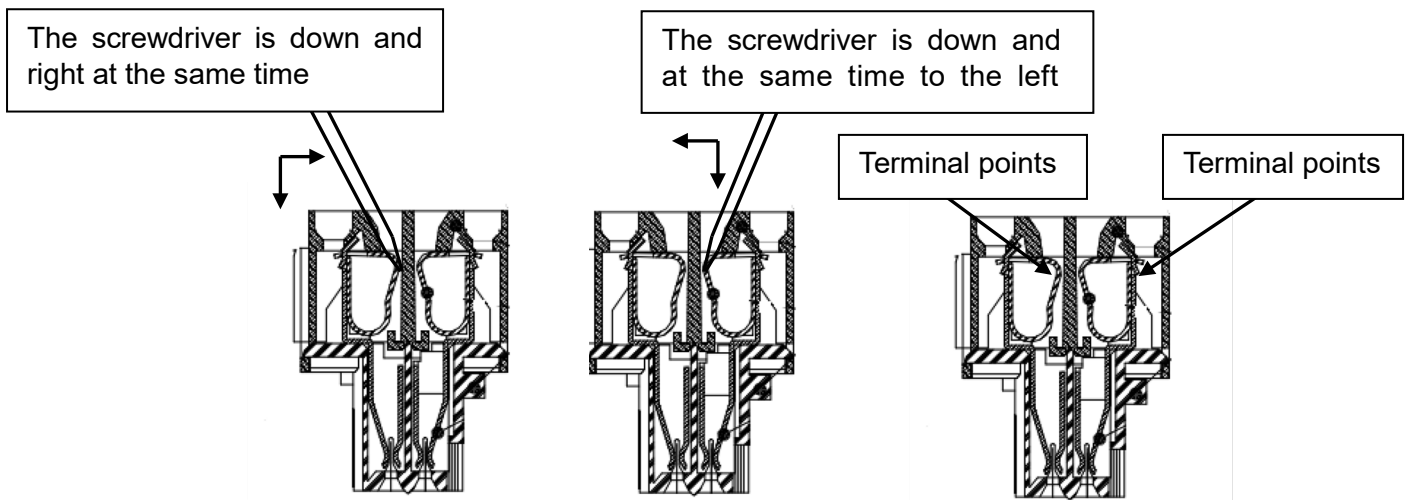
- ① Stripping length : 7 ~ 8mm



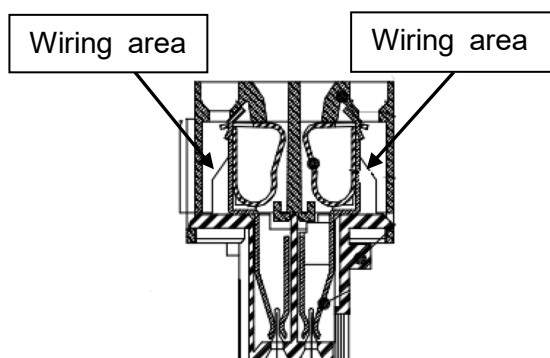
Do not put a layer of solder on the thread.
(May result in failure to wire properly)

The terminals are pulled back spring connections and are connected on the front for easy operation

- ② You can open the terminal points with a standard screwdriver.



- ③ Insert the wire into the wiring area and remove the screwdriver. The wires are automatically connected.



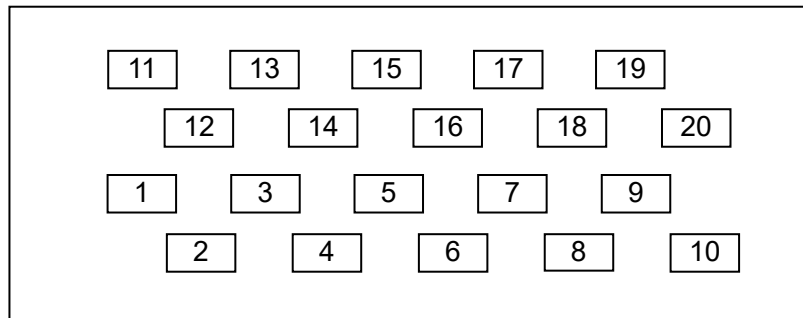
Caution

Observe the following items and be careful not to break the line.

- * When stripping the cladding, do not damage the core.
- * When wiring, be careful not to kink the core wire, and the core wire must not leak to avoid causing a short circuit of the wire.
- * Please connect the core wire directly, do not solder. Otherwise, the wire may be broken due to vibration.
- * After wiring, no pressure should be applied to the wires.
- * A screwdriver of the specified size and equivalent type must be used, otherwise there is a risk of damaging the terminal spring.

3.5 CN5 (I/O)

ICON :



Pin.	Signal name	Pin.	Signal name	Pin.	Signal name
1	COM (IN)	8	IN6-	15	Encoder A+
2	IN1	9	IN7+	16	Encoder A-
3	IN2	10	IN7-	17	Encoder B+
4	IN3	11	OUT1	18	Encoder B-
5	IN4	12	OUT2	19	Encoder Z+
6	IN5	13	OUT3	20	Encoder Z-
7	IN6+	14	COM (OUT)		

	Notes	Pin15~20 is the encoder output (differential output), which is optional and needs to be indicated when ordering
--	--------------	---

3.6 CN6 (IN) / CN7 (OUT) (RS485)

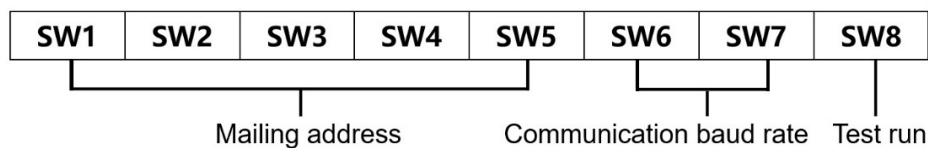
Pin.	Signal name	Pin.	Signal name
1	NC	2	GND
3	A Input (RS485)	4	NC
5	NC	6	B Input (RS485)
7	Termination resistance (CN5)	8	Termination resistance (CN5)

Standard product: RJ45 type × 2

Look at the position of each pin from the perspective facing the insertion



3.7 SW1 (Setting Switch)



3.7.1 Mailing Address

Users can control up to 30 DS-CLS10-FRS4 drivers simultaneously using the RS-485 bus. The drive communication address setting adopts a 5-digit DIP switch,

The address setting range is 1-32, where address 32 is reserved for the system, when the drive address setting is greater than 31, it needs to be set and saved using the upper debugging software.

And the switch needs to be set to OFF (default is 1).

i

Notes

1) One controller can control up to 30 DS-CLS10-FRS4 drives simultaneously via RS-485 bus.

2) The communication address setting of each drive must be unique, otherwise it will cause communication errors.

DIP switch					Physical address (DEC)	Displays the address (HEX)
SW1	SW2	SW3	SW4	SW5		
ON	ON	ON	ON	ON	1	01H
ON	ON	ON	ON	OFF	2	02H
ON	ON	ON	OFF	ON	3	03H
ON	ON	ON	OFF	OFF	4	04H
ON	ON	OFF	ON	ON	5	05H
ON	ON	OFF	ON	OFF	6	06H
ON	ON	OFF	OFF	ON	7	07H
ON	ON	OFF	OFF	OFF	8	08H
ON	OFF	ON	ON	ON	9	09H
ON	OFF	ON	ON	OFF	10	0AH
ON	OFF	ON	OFF	ON	11	0BH
ON	OFF	ON	OFF	OFF	12	0CH
ON	OFF	OFF	ON	ON	13	0DH
ON	OFF	OFF	ON	OFF	14	0EH
ON	OFF	OFF	OFF	ON	15	0FH
ON	OFF	OFF	OFF	OFF	16	10H
OFF	ON	ON	ON	ON	17	11H
OFF	ON	ON	ON	OFF	18	12H
OFF	ON	ON	OFF	ON	19	13H
OFF	ON	ON	OFF	OFF	20	14H

OFF	ON	OFF	ON	ON	21	15H
OFF	ON	OFF	ON	OFF	22	16H
OFF	ON	OFF	OFF	ON	23	17H
OFF	ON	OFF	OFF	OFF	24	18H
OFF	OFF	ON	ON	ON	25	19H
OFF	OFF	ON	ON	OFF	26	1AH
OFF	OFF	ON	OFF	ON	27	1BH
OFF	OFF	ON	OFF	OFF	28	1C H
OFF	OFF	OFF	ON	ON	29	1D H
OFF	OFF	OFF	ON	OFF	30	1E H
OFF	OFF	OFF	OFF	ON	31	1F H
OFF	OFF	OFF	OFF	OFF	customize	customize

3.7.2 Communication Baud Rate

DIP switch		baud rate (bps)
SW6	SW7	
ON	ON	4800
ON	OFF	9600
OFF	ON	19200
OFF	OFF	38400

3.7.3 Test Run

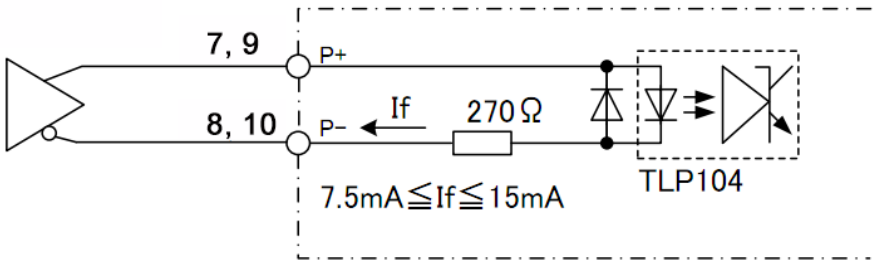
The pilot run function is used to verify the performance of the drive. When the power is off, set the SW8 switch to ON. Then power on in the state of no pulse input, dial the SW8 DIP switch from ON to OFF, and then dial ON gear from OFF gear after 1 second, that is, start the trial run function (the motor cycles forward and reverse movement at a speed of 1 revolution/second).

4. Input Loop Diagram



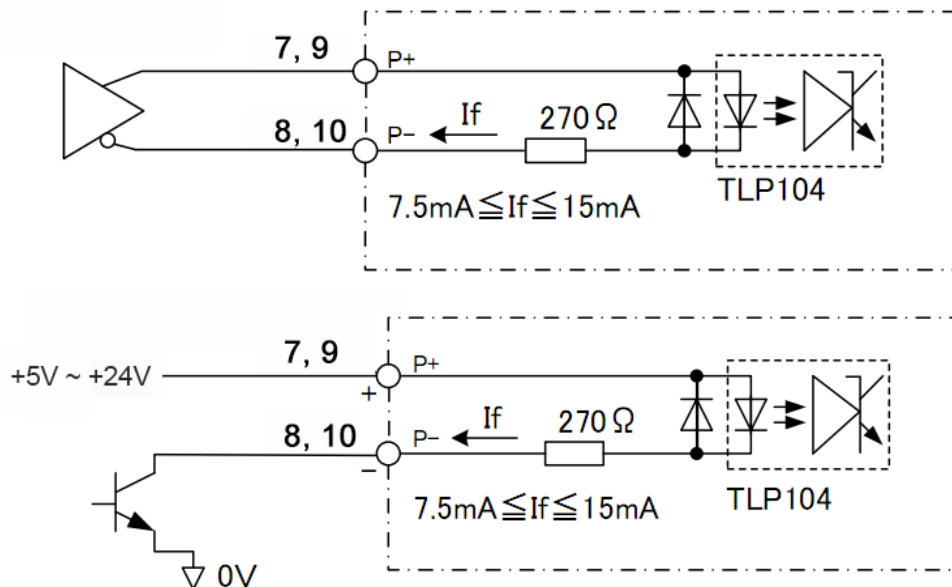
4.1 Command Pulse Input Loop (differential drive)

AM26LS31 equivalent



4.2 Command Pulse Input Loop (collector)

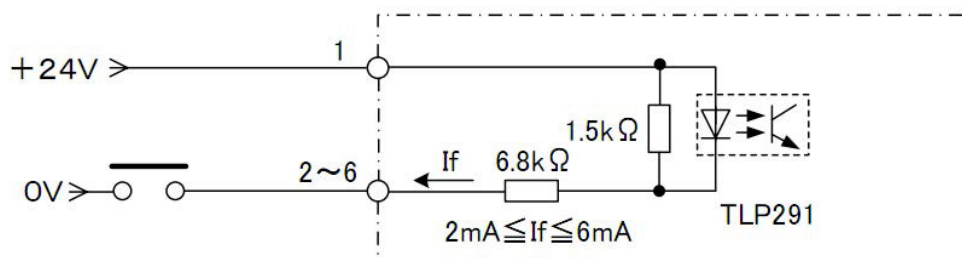
AM26LS31 equivalent



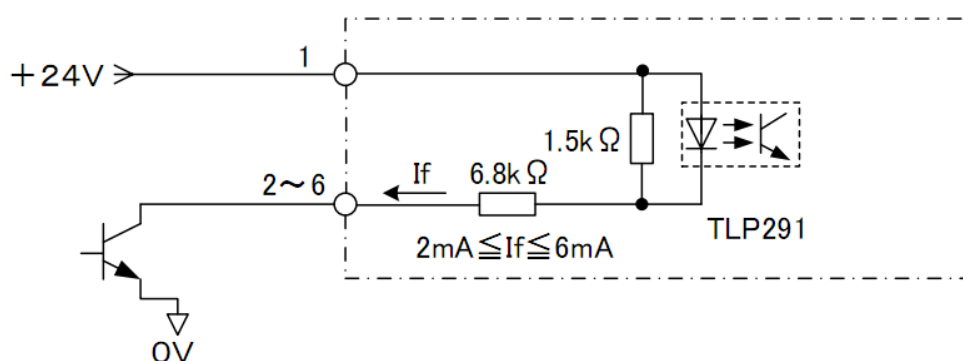
Notes

This product is compatible with +5V/+24V signal and there is no need to connect current limit resistor in serial when 24V input.

4.3 Sensor, Digital Input Loop (contacts)



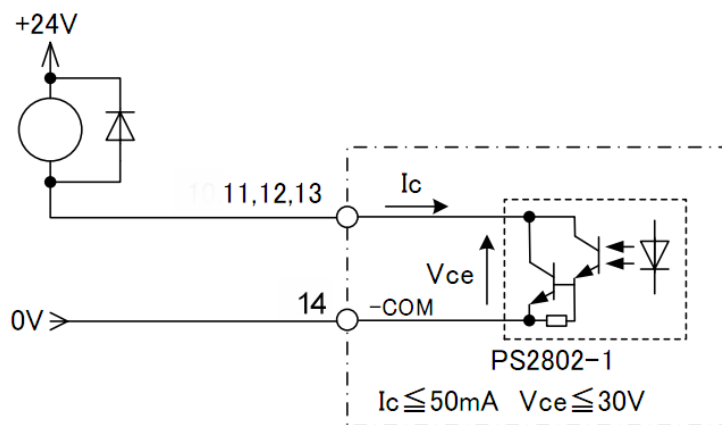
4.4 Sensor, Digital Input Loop (collector output)





5. Output Loop Diagram

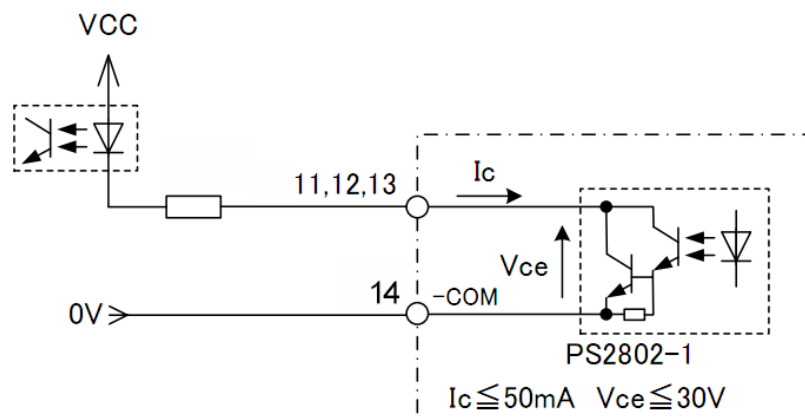
5.1 Digital Output Loop (relay connection)



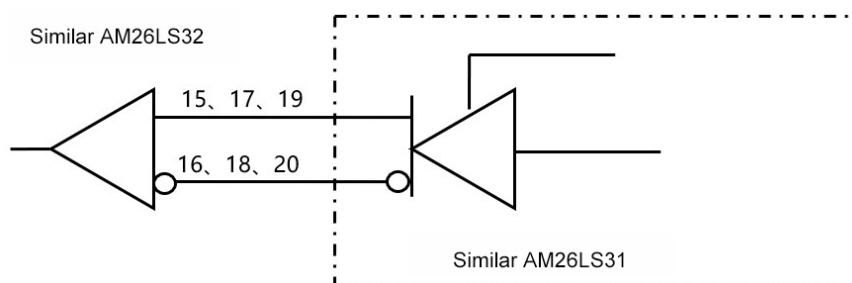
Notes

When the relay is connected, it is required to connect diodes at both ends of the relay (such as IN4000 series)

5.2 Digital Output Loop (optocoupler connection)



5.3 Differential Output Loop (encoder output)



Notes

The encoder output has no optocoupler isolation, please confirm again whether the wiring is correct and whether there is a short circuit before powering on, so as to avoid introducing the 24V power supply on the port and damaging the host computer and driver.

6. LED Indicator



6.1 Status display

Display	Description
	Motor rotation display The light is on when the motor rotates and turns off when it stops
	Device enabled state The device enabling light is on, and the device disabling light is off
	Displayed in the command input The light is on in the command input
	CONNECT display The light is on in CONNECT

6.2 The Site Number is displayed

The site number is displayed verbatim, ending with H, and only the status is displayed after the connect connection is successful.

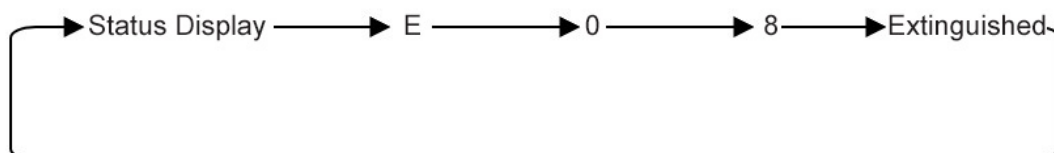
Example : Site number : 45H



6.3 Alarm display

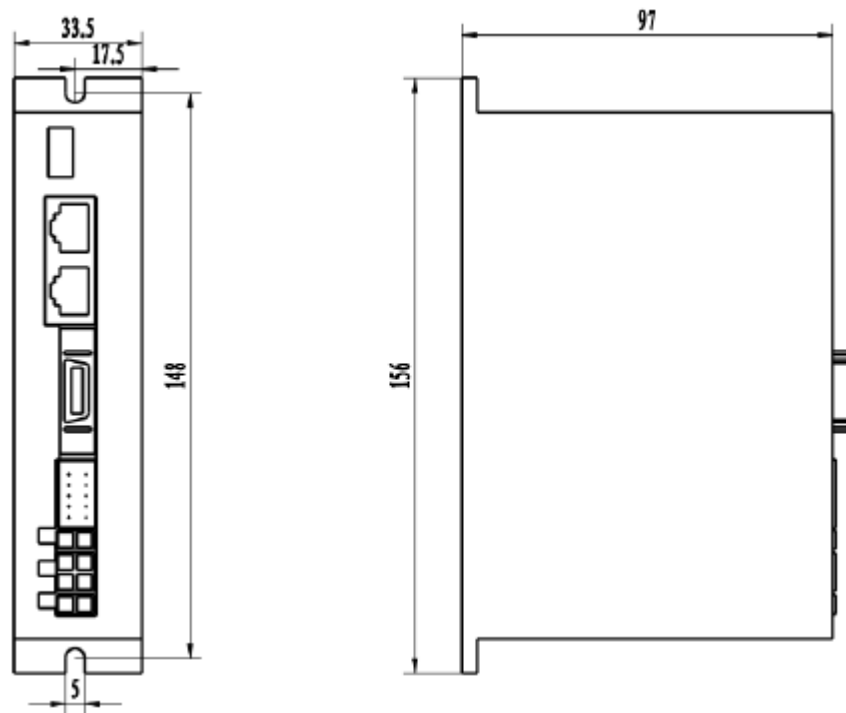
The alarm code is verbatim and flashes and ends with E

Example : Alarm code E8



Function	Alarm code	Alarm/warning (Hex/Dec)	Description
Motor overcurrent	Alarm	AH (10)	Motor phase current overcurrent or drive failure
The motor is out of phase	Alarm	bH (12)	The motor is not connected
Spare	Alarm	CH (13)	Factory reserved
Undervoltage	Alarm	dH (14)	The power input is less than 18V
Overvoltage	Alarm	EH (15)	The power input is greater than 60V
Superheating	Alarm	FH (16)	The driver radiator temperature reaches above 85°C
MOS transistor driver voltage failure	Alarm	10H (17)	MOS transistor driver voltage failure
Spare	Alarm	11H (18)	Factory reserved
Spare	Alarm	12H (19)	Factory reserved
Spare	Alarm	13H (20)	Factory reserved
Abnormal EEPROM data writing	Alarm	14H (21)	EEPROM data write exception
Overspeed error	Alarm	18H (24)	The motor runs faster than the maximum system value
Position out of tolerance	Alarm	19H (25)	Position out of tolerance
Current overload	Alarm	1AH (26)	Current overload
Encoder error	Alarm	1BH (27)	The encoder is wired incorrectly
EEPROM data read exception	Warn	100H (256)	EEPROM data read exception
The bus voltage is unstable	Warn	200H (512)	The bus voltage is unstable
Emergency stop	Warn	400H (1024)	Emergency stop
Positive limit	Warn	800H (2048)	On positive limits or super positive and soft limits
Negative limit	Warn	1000H (4096)	On the negative limit or above the negative soft limit
Return to origin failed	Warn	2000H (8192)	Return to origin failed

7. Form Factor (mm)



8. Control Parameters



Notes

Communication parameters are unofficial version, and some parameters are fixed and not set open

8.1 Controller Basic Status (Class 01)

adr	word	content	recount	range / unit
0100	1	Motor current	Real-time motor current value	0.1%A
0101	1	Input voltage	Current input voltage	1%V
0104	2	Set up segments	Set breakdown values	ppr
0106	1	Pulse mode	1 is pulse + direction mode, 2 is double pulse mode	1-2
0108	1	Failure code	The alarm code, see 1-2, shows "0" as no fault	-
0109	1	Running status	Drive operating status, see 1-1	-
0110	1	Hardware version	Drive hardware version	-
0111	1	Software version	Drive software version	-
0117	2	Current location	Target location	pulse
0119	1	Actual speed display	-	0.01rps

0126	2	Physical location	Run a live location	pulse										
0174	1	IO Select Multi-segment Run Paragraph	-	-										
0176	1	Multi-segment writing error No	-	-										
0178	1	Multi-stage operation No	-	-										
0135	1	Enter the port status	<table border="1"> <tr> <td>Data bits</td><td>Bit7</td><td>.....</td><td>Bit1</td><td>Bit0</td></tr> <tr> <td>Enter the port</td><td>IN7</td><td>.....</td><td>IN2</td><td>IN1</td></tr> </table>	Data bits	Bit7	Bit1	Bit0	Enter the port	IN7	IN2	IN1	
Data bits	Bit7	Bit1	Bit0										
Enter the port	IN7	IN2	IN1										
0136	1	Output port status	<table border="1"> <tr> <td>Data bits</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr> <tr> <td>Output port</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr> </table>	Data bits	Bit3	Bit2	Bit1	Bit0	Output port	OUT4	OUT3	OUT2	OUT1	
Data bits	Bit3	Bit2	Bit1	Bit0										
Output port	OUT4	OUT3	OUT2	OUT1										

8.2 Basic Parameter Setting (Class 02)

adr	word	content	recount	range/unit
0201	1	Motor direction switching	Select the direction in which the motor runs	0~1
0213	1	Half-flow ratio	Stop current ratio (active in open-loop mode)	10%~120%
0217	1	Motor control mode	0: Open loop 1: Closed loop Default: 1	0~1
0224	1	Angular filtering	The smaller the value, the smoother the motor runs, but the higher the latency	1~700
0234	1	Digital filtering	The filter coefficient of the input pulse, the larger the value, the lower the input frequency response	1~15
0241	1	Input current	Set the current	100~4500 0.1A~-4.5A
0242	2	Set up segments	Number of pulses per revolution	200~102400ppr
0244	1	Pulse mode	1: Pulse + direction mode 2: Double pulse mode	1~2
0245	1	Half-flow time	Time delay time to enter half-flow after motor stops operation (active in open-loop mode)	1~32767ms
0296	1	Operating mode selection	0: External pulse 1: Internal pulse Default: 0 Note: After the function is modified, the power needs to be turned off and restarted	0~1
0298	1	Mailing address	Default: 1	1~255
0299	2	Communication baud rate	Default: 19200	1600~115200

8.3 Closed-Loop Parameter Setting (Class 04)

adr	word	content	recount	range/unit
0246	1	Encoder resolution	Resolution = Number of encoder lines x 4	200~65535
0247	2	Pulse width in place	Reach the target position close to the distance, output the signal in place Default: 0	1~1000 Encoder resolution
0251	1	Speed ring Kp	Speed ring Kp	0~30000
0252	1	Speed ring Ki	Speed ring Ki	0~30000

0255	1	Location ring Kp	Location ring Kp	0~30000
0258	1	Location out-of-tolerance threshold	In units of encoder resolution	0~30000 Encoder resolution

8.4 Control Parameters (Class 05)

adr	word	content	recount	range/unit
0301	1	Startup frequency	Default: 100	1~2000 0.01~20rps
0302	1	Stop frequency	Default: 100	1~2000 0.01~20rps
0303	1	acceleration	Default: 100	5~10000rps ²
0304	1	Deceleration	Default: 100	5~10000rps ²
0305	1	Return to origin mode	Return to origin mode, 0: Clockwise back to origin 1: Return counterclockwise to the origin	0~1
0306	1	Fixed-length running speed	Default: 1000	1~5000 0.01~50rps
0307	1	Speed mode runs speed	In speed mode, the direction of operation coincides with the direction of speed Default: 1000	-5000~5000 -50~50rps
0308	1	Jog running speed	Default: 1000	1~5000 0.01~50rps
0309	1	Return to origin running speed	Default: 1000	1~5000 0.01~50rps
0310	1	Return to the origin peristaltic speed	The speed of operation after hitting the origin Default: 1000	1~5000 0.01~50rps
0311	2	Return to origin offset	Default: 0	-2000000000~ 2000000000 pulse
0313	2	Output pulse	Run the trip Absolute position mode: Runs to a specified location Relative Position Mode: Run the set offset stroke Default: 0	-2000000000~ 2000000000 pulse
0317	2	Positive and soft limits	Default: 2000000000 Note: The process of returning to the origin is not valid	-2000000000~ 2000000000 pulse
0319	2	Negative soft limit	Default: -2000000000 Note: The process of returning to the origin is not valid	-2000000000~ 2000000000 pulse
0321	2	Sets the current location	Default: 0	-2000000000~ 2000000000 pulse
0323	1	Control commands	0: Empty 1: Absolute operation, running to the set distance, the running direction is determined by the distance plus or minus, the speed plus and minus values are invalid, and the modification of the target position during operation is effective 2: Relative operation, run with set distance and running speed, the running direction is determined by the distance plus or minus, the speed plus and minus value is invalid, and the modification of the	0~29

moving distance during operation is invalid
 3: Speed mode
 4: Positive jogging
 5: Reverse jogging
 6: Decelerate and stop
 7: Emergency stop
 8: Set the current position, only when the motor stops
 12: Return to the original point
 13: Alarm clearance
 14: Multi-segment data verification
 15: Multi-segment data saving
 16: Multi-segment data starts
 17: Multi-segment data paused
 18: End of multi-segment data
 Default: 0

0324	1	Internal control switch	Data bits	Bit1	Bit0	0-65535
			function	Negative soft limit	Positive and soft limits	
1: Turn on the function, 0: Turn off the function Default: 0						
0327	1	Number of multiple paragraphs	Default: 1			1~32
0328	1	Multi-segment selection	Default: 0 Note: If the IO port is configured with the multi-segment selection feature, the IO configuration multi-segment selection takes precedence			0~31

8.5 Input Block Designation (Class 06)

adr	word	content	recount	range /unit
0400	1	IN1 function selection	0: Empty	0~30
			1: Absolute operation, running to the set distance, the running direction is determined by the distance plus or minus, the speed plus and minus values are invalid, and the modification of the target position during operation is effective	
0400	1	IN1 function selection	2: Relative operation, run with set distance and running speed, the running direction is determined by the distance plus or minus, the speed plus and minus value is invalid, and the modification of the moving distance during operation is invalid	0~30
			3: Speed mode	
			4: Positive jogging	
			5: Reverse jogging	
			6: Decelerate and stop	
			7: Emergency stop	
			8: Set the current position, only when the motor stops	
			9: Positive limit	
			10: Negative limit	
			11: Origin signal	
			12: Return to the original point	
			13: Alarm clearance	
			14: Multi-segment data verification	

			15: Multi-segment data saving 16: Multi-segment data starts 17: Multi-segment data paused 18: End of multi-segment data 20: Enablement 25: Select Bit0 for the IO port configuration multi-segment 26: The IO port is configured with multiple segments to select Bit1 27: The IO port is configured with multiple segments to select Bit2 28: The IO port is configured with multiple segments to select Bit3 29: IO port configuration multi-segment selection Bit4 Default: 0	
0401	1	IN2 function selection	The setting content is the same as IN1 (default: 0)	0~30
0402	1	IN3 function selection	The setting content is the same as IN1 (default: 0)	0~30
0403	1	IN4 function selection	The setting content is the same as IN1 (default: 0)	0~30
0404	1	IN5 feature selection	The setting content is the same as IN1 (default: 0)	0~30
0405	1	IN6 Function Selection (CCW Port)	The setting content is the same as IN1 (default: 0) (When the external pulse is off, the port function fails)	0~30
0406	1	IN7 Function Selection (CW Port)	The setting content is the same as IN1 (default: 0) (When the external pulse is off, the port function fails)	0~30
0429	1	Universal numeric input logic		
0410	1	Pseudo communication setting IN1	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration)	0~1
0411	1	Pseudo communication setting IN2	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration)	0~1
0412	1	Pseudo communication setting IN3	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration)	0~1
0413	1	Pseudo communication setting IN4	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration)	0~1
0414	1	Pseudo communication settings IN5	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration)	0~1
0415	1	Pseudo communication setting IN6	0: OFF (initial value 0)	0~1
0416	1	Pseudo communication settings IN7	0: OFF (initial value 0) 1: ON (action that triggers IN1 configuration) (When the external pulse is used, the pseudo-communication port function fails)	0~1

8.6 Output Block Designation (Class 07)

adr	word	content	recount	range/unit
			100: Generic port 101: Alarm output function: There is an output signal when there is no alarm, and no output signal when there is an alarm.	
0420	1	OUT1 function selection	102: Signal in place 103: Enable control output: There is an output signal when offline, and no output signal when enabled. (Default value: 101)	100~104

0421	1	OUT2 function selection	The setting content is the same as OUT 1 (default: 100)	100~104										
0422	1	OUT3 function selection	The setting content is the same as OUT 1 (default: 100)	100~104										
0423	1	OUT 4 feature selection	The setting content is the same as OUT 1 (default: 100)	100~104										
0428	1	Universal digital output control	Output port function selects 100 <table border="1"> <tr> <td>Data bits</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr> <tr> <td>Output port</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr> </table>		Data bits	Bit3	Bit2	Bit1	Bit0	Output port	OUT4	OUT3	OUT2	OUT1
Data bits	Bit3	Bit2	Bit1	Bit0										
Output port	OUT4	OUT3	OUT2	OUT1										
0430	1	Digital output logic	Corresponds to the output port logic <table border="1"> <tr> <td>Data bits</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr> <tr> <td>Output port</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr> </table>		Data bits	Bit3	Bit2	Bit1	Bit0	Output port	OUT4	OUT3	OUT2	OUT1
Data bits	Bit3	Bit2	Bit1	Bit0										
Output port	OUT4	OUT3	OUT2	OUT1										

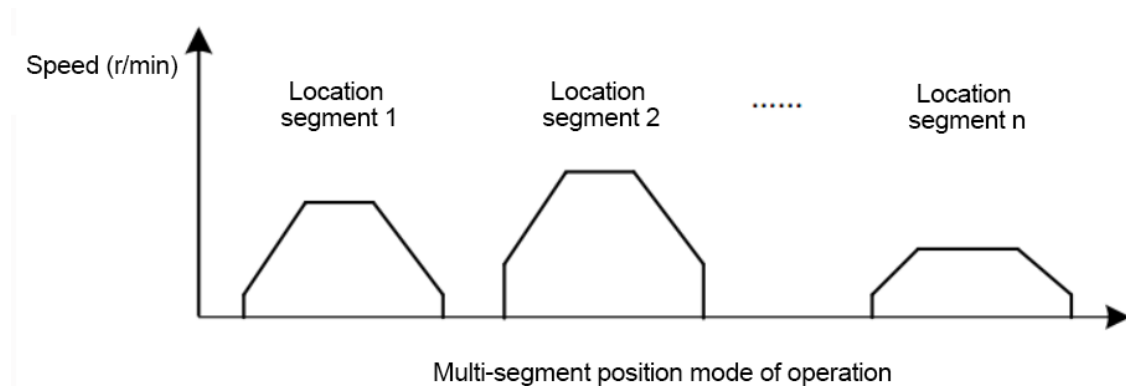
8.7 Multi-segment Position Mode (Class 08)

The multi-segment address range is 1024~1536, and up to 256 data can be set

Multi-segment command format

Command code	word	content	recount	range/unit
1	2	Absolutely run	Parameter 1: Running position Default: 0	-2147483647~ 2147483647 pulse
2	2	Relative operation	Parameter 1: Running position Default: 0	-2147483647~ 2147483647 pulse
51	1	Startup speed	Default: 100	1~2000 0.01~20rps
53	1	Stop speed	Default: 100	1~2000 0.01~20rps
54	1	Fixed length speed	Default: 1000	1~5000 0.01~50rps
61	1	acceleration	Default: 100	5~10000rps ²
62	1	Deceleration	Default: 100	5~10000rps ²
65	2	Wait for the jump	A (high 8 bits) / B (low 8 bits) / C (low 16 bits), A: fixed at 0 / B: jump address / C: Waiting time	-
66	2	Jump sequence	A (high 16 bits) / B (low 16 bits), A: number of cycles / B: jump address	-
100	1	End of multiple segments	Each paragraph should end with an end code	-

The multi-segment position mode function is a working method that combines multiple position segments in a certain order, triggers the movement through an external IO signal, and completes a series of position segment actions. This function can be regarded as a multi-segment combination of position mode, the user can store the description parameters of several position segments such as acceleration and deceleration, pulse number, etc. in advance in EEPROM, and only need to provide a trigger signal to complete the work when these position segments need to be enabled, and its working process description is shown in the figure below.



Port selection corresponds to multiple segments

Bit4	Bit3	Bit2	Bit1	Bit0	Location segment
0	0	0	0	0	1
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
...
1	1	1	0	1	30
1	1	1	1	0	31
1	1	1	1	1	32

IO Select the port

1. Input port configuration multi-segment selection function 25~29: IO port configuration multi-segment selection Bit0~Bit4

Input port configuration multi-segment start function 15: multi-segment data start

2. Port selection corresponds to multiple segments

Example: IN1 port function configuration 25, Bit0

IN3 port function configuration 26, Bit1

The IN1~ IN7 function can be configured according to the requirements

IN3	Bit1	IN1	Bit0	Location segment
0		0		1
0		1		2
1		0		3
1		1		4

i

Notes

"1" in the table indicates a valid hold signal
The segment selection signal needs to be completed more than 20ms in advance of the start signal

Example: Writing, validating, and saving multi-segment parameters *Note: The data in the example is expressed in base 16

1. Multi-segment parameter settings

[Command 1] the current line number 0: the fixed length speed is set to 1000, that is, 10rps,

<u>01</u>	<u>10</u>	<u>04 00</u>	<u>00 02</u>	<u>04</u>	<u>00 36</u>	<u>03 e8</u>	<u>21 DF</u>
①	②	③	④	⑤	⑥	⑦	⑧

- ① : Mailing address 0x1
- ② : MODBUS WRITE COMMAND 0x10
- ③ : Mailing address 0x400 (decimal means 1024)
- ④ : Write 2 pieces of data
- ⑤ : Write 4 bytes
- ⑥ : Data 1, fixed-length speed command 0x0036 (decimal means 54)
- ⑦ : Data 2, fixed length speed value 0x03E8 (decimal means 1000)
- ⑧ : CRC check

[Command 2] current line number 1: relative operation, running distance 10000 pulses

<u>01</u>	<u>10</u>	<u>04 02</u>	<u>00 03</u>	<u>06</u>	<u>00 02</u>	<u>27 10 00 00</u>	<u>20 CB</u>
①	②	③	④	⑤	⑥	⑦	⑧

- ① : 0x1 mailing address
- ② : MODBUS write command 0x10
- ③ : Mailing address 0x402 (decimal means 1026)
- ④ : Write 3 data
- ⑤ : Write 6 bytes
- ⑥ : Data 1, relative to the command 0x0002 (decimal means 2)
- ⑦ : Data 2, parameters: running pulse value 0x2710 (decimal means 10000)
- ⑧ : CRC verification

[Command 3] current line number 2: wait 1000ms

<u>01</u>	<u>10</u>	<u>04 05</u>	<u>00 03</u>	<u>06</u>	<u>00 41</u>	<u>03 E8 00 03</u>	<u>1F DE</u>
①	②	③	④	⑤	⑥	⑦	⑧

- ① : Mailing address 0x1
- ② : MODBUS write command 0x10
- ③ : Mailing address 0x405 (decimal means 1029)

- ⑧ : CRC check

⑨ : CRC check

page | 22

2. Multi-segment parameter checking

<u>01</u>	<u>06</u>	<u>01 43</u>	<u>00 0E</u>	<u>F8 26</u>
①	②	③	④	⑤

- ① : 0x1 mailing address
- ② : MODBUS write command 0x06
- ③ : Mailing address 0x0143 (decimal means 323, write communication command)
- ④ : Data multi-segment data check 0xE (decimal representation 14)
- ⑤ : CRC check

3. Multi-segment parameter saving

*Note: Data can only be saved after the data verification is successful, otherwise the data can not be saved normally

<u>01</u>	<u>06</u>	<u>01 43</u>	<u>00 0F</u>	<u>39 E6</u>
①	②	③	④	⑤

- ① : 0x1 mailing address
- ② : MODBUS write command 0x06
- ③ : Mailing address 0x0143 (decimal means 323, write communication command)
- ④ : Data Multi-segment data check 0xF (decimal representation 15)
- ⑤ : CRC check

9. Message Format



The MODBUS protocol defines a protocol data unit (PDU) independent of the underlying communication layer and the MODBUS protocol on the RS485 physical layer is mapped on the application data unit (ADU).

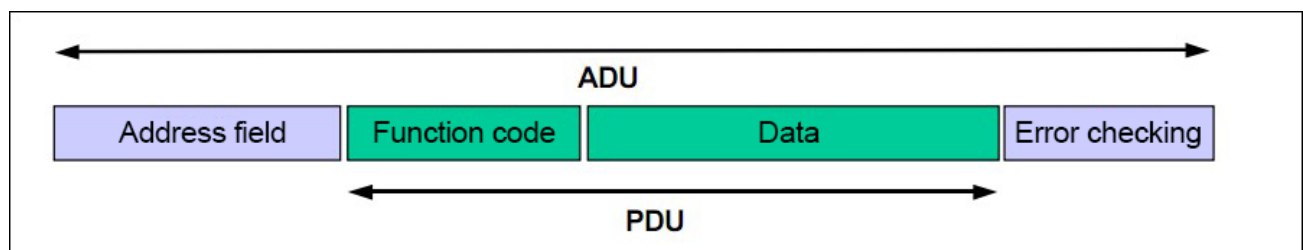


Figure 1 : Generic MODBUS frames

The length constraint performed by the first MODBUS on the serial link limits the MODBUS PDU size (maximum RS485ADU=256 bytes).

Therefore, for serial link communication, MODBUS PDU=256-server address (1 byte)-CRC (2-by te)=253 bytes.

Thereby:

RS232 / RS485 ADU = 253 bytes + server address (1 byte) + CRC (2 bytes) = 256 bytes. The MODBUS protocol defines three types of PDUs. They are:

- a. MODBUS requests PDU, mb_req_pdu
- b. MODBUS responds to PDU, mb_rsp_pdu
- c. MODBUS abnormally responds to PDU, mb_excep_rsp_pdu

Definition mb_req_pdu as:

mb_req_pdu = { function_code, request_data}, where
 function_code - [1 byte] MODBUS function code
 request_data - [n bytes], this field is related to function codes and typically includes information such as variable references, variables, data offsets, subfunction codes, and so on.

Definition mb_rsp_pdu as:

mb_rsp_pdu = { function_code, response_data}, where
 function_code - [1 byte] MODBUS function code
 response_data - [n bytes], this field is related to function codes and typically includes information such as variable references, variables, data offsets, subfunction codes, and so on.

Definition mb_excep_rsp_pdu as:

mb_excep_rsp_pdu = { function_code, request_data}, where
 function_code - [1 byte] MODBUS function code + 0x80
 exception_code - [1 byte], the MODBUS exception code is defined in the following table.

10. MODBUS Transactions



10.1 Definition of MODBUS Transactions

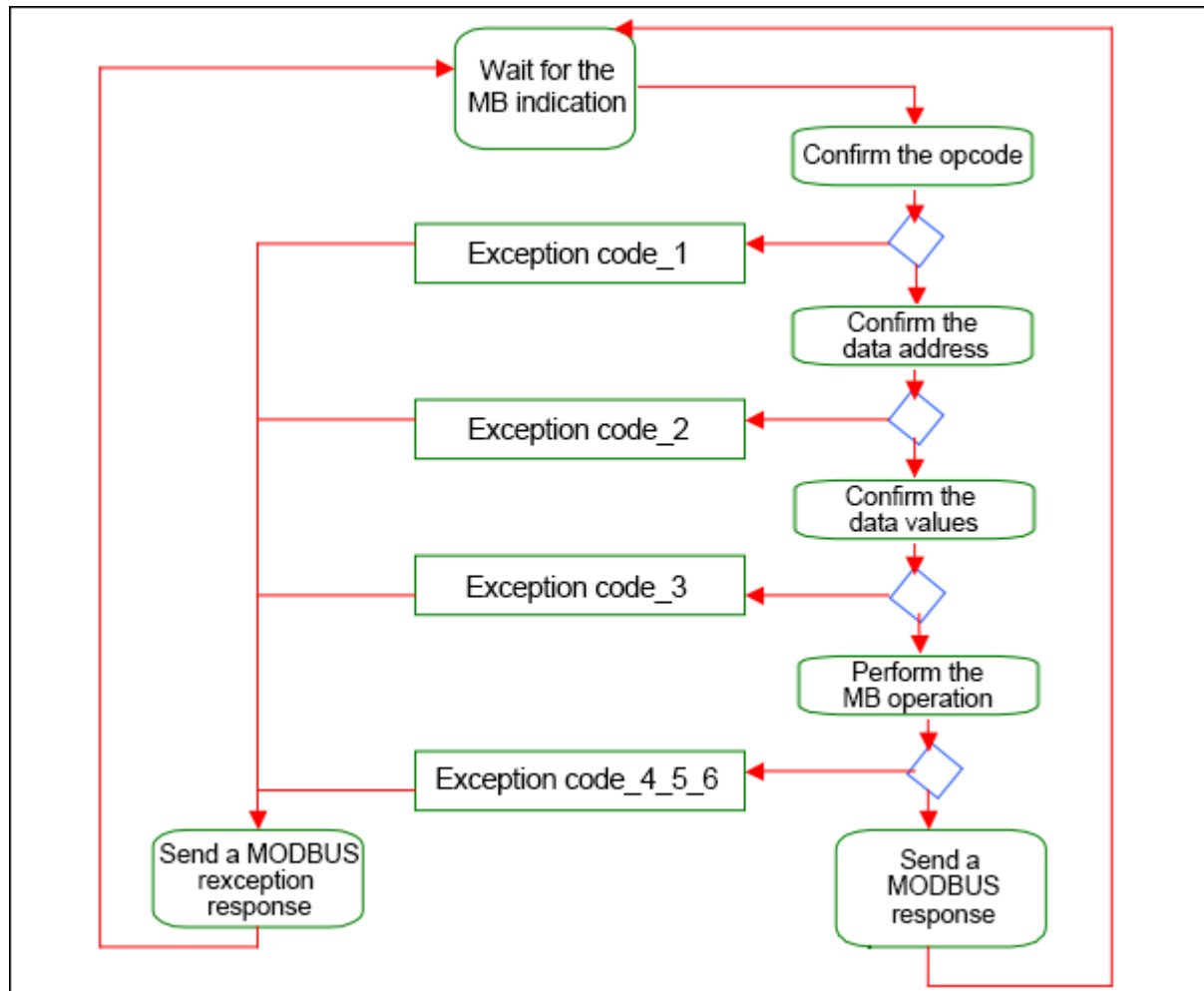


Figure 2: A state diagram of MODBUS transactions

Once the server processes the request, establish a MODBUS response using the appropriate MODBUS server transaction.

Based on the processing results, two types of responses can be established:

1. A MODBUS normal response :

Response Function Code = Request Function Code

2. A MODBUS EXCEPTION RESPONSE :

a. Used to provide the client with information related to the errors found in the process of processing

b. Response function code = request function code + 0x80

c. Provide an exception code to indicate the cause of the error.

10.2 MODBUS Responds Normally

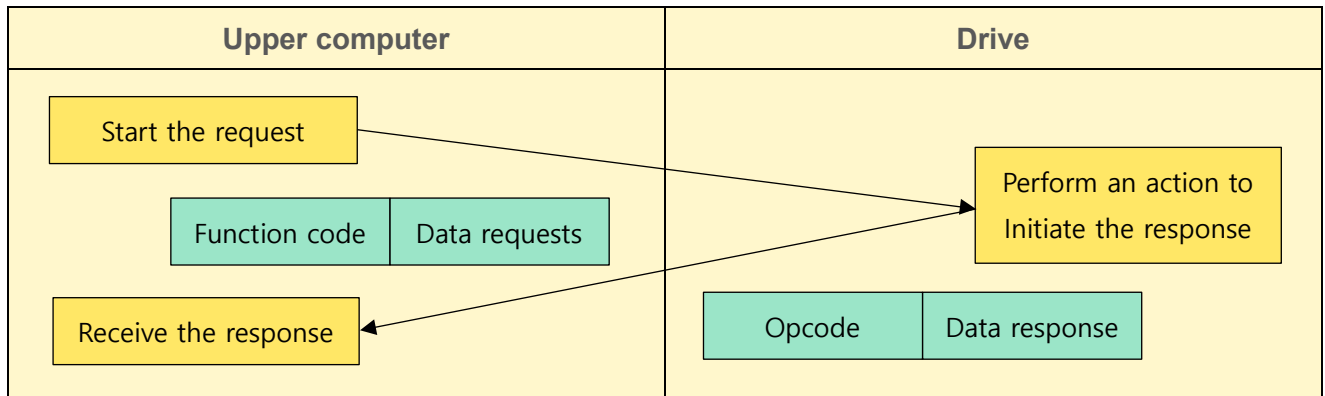


Figure 3: MODBUS transaction processing (error-free)

10.3 MODBUS exception response

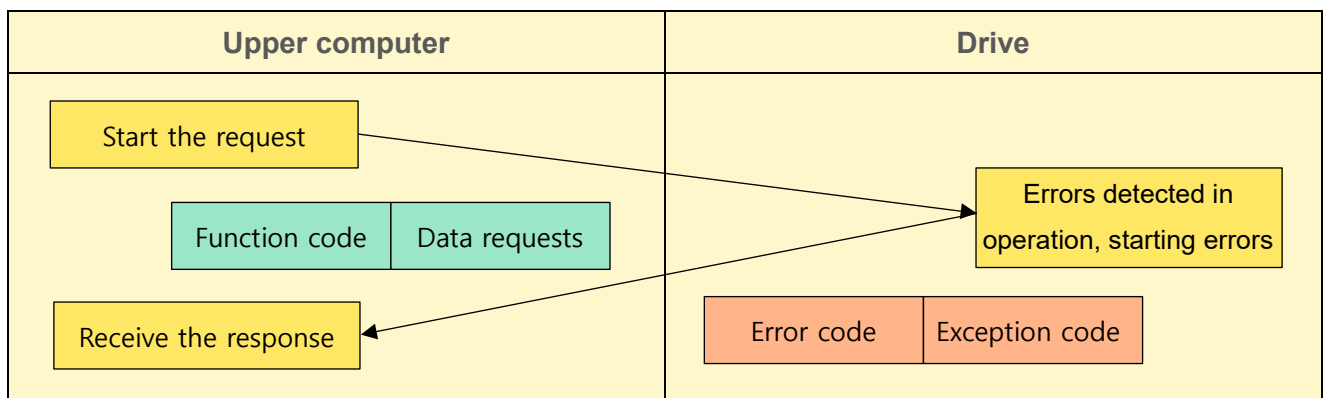


Figure 4: MODBUS transaction processing (exception response)

11. Data Encoding



MODBUS uses a big-Endian specification (i.e. high bits first, low bits last) to represent addresses and data items.

12. Definition of Public Function Code and Description of Function Code



				Function code		
				code	Subcode	Hexadecimal
Data access	Bit access	Physical discrete input	Read input discrete	02		02
		Internal bits or physical coils Enter memory Physical discrete input	Read the coil	01		01
			Write a single coil	05		05
			Write multiple coils	15		0F
	16 Bit access	Internal bits or physical coils	Read input registers	04		04
		Enter memory	Read multiple registers	03		03
			Write a single register	06		06
			Write multiple registers	16		10
			Read/write multiple registers	23		17
			Masked write registers	22		16
			File record access	Read the file record	20	6
	Write a file record	21		6	15	
Encapsulate the interface			Read the device identification code	43	14	2B

The common function codes used by STAP according to communication needs are the yellow part of the above table

03 (0x03) Read hold register

06 (0x06) Write a single register

16 (0x10) Write multiple registers

12.1 03 (0x03) Read Hold Registers

In a remote device, use this function code to read the contents of contiguous blocks of keep-registers. The requesting PDU states the starting register address and number of registers. Address registers from scratch. Therefore, addressing registers 1-16 is 0-15.

Divide the register data in the response packet into two bytes per register, and adjust the binary content directly in each byte.

For each register, the first byte includes the high bit, and the second byte includes the low bit.

Request

Function code	1 byte	0x03
Start address	2 byte	0x0000 to 0xFFFF
Number of registers	2 byte	1 to 125 (0x7D)

Response

Function code	1 byte	0x03
Number of bytes	1 byte	2 x N*
Register value	N* x 2 bytes	

*N = Number of registers

Mistake

Error code	1 byte	0x83
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of requesting read registers 108-110:

Request		Response	
Domain name	(hexadecimal)	Domain name	(hexadecimal)
function	03	function	03
High start address	00	Number of bytes	06
Low start address	6B	Register value Hi(108)	02
High register number	00	Register value Lo(108)	2B
Low register number	03	Register value Hi(109)	00
		Register value Lo(109)	00
		Register value Hi(110)	00
		Register value Lo(110)	64

The contents of register 108 are represented as two hexadecimal byte values 02 2B, or decimal 555. The contents of registers 109-110 are expressed as hexadecimals 00 00 and 00 64, or decimal 0 and 100, respectively

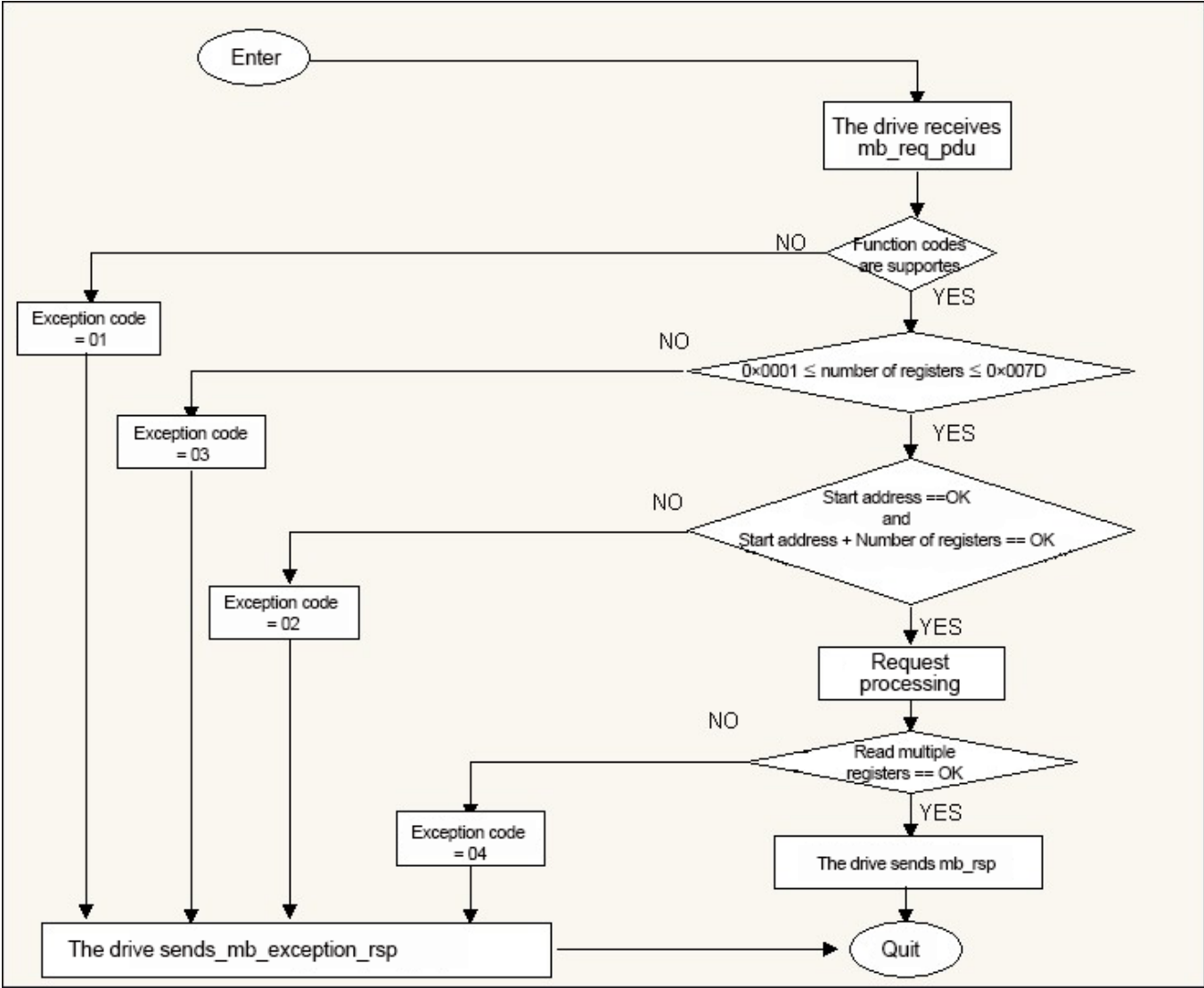


Figure 5: Read the state diagram of the hold-and-register register

12.2 06 (0x06) Write a Single Register

In a remote device, use this function code to write a single hold register. The request PDU states the address of the register being written. Address registers from scratch. Therefore, address register 1 is 0. A normal response is a reply to a request that is returned after the contents of the register are written.

Request

Function code	1 byte	0x06
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

Response

Function code	1 byte	0x06
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

*N = Number of registers

Mistake

Error code	1 byte	0x86
Exception code	1 byte	01 or 02 or 03 or 04

Here is an instance of request to write hexadecimal 00 03 to register 2:

Request		Response	
Domain name	(hexadecimal)	Domain name	(hexadecimal)
function	06	function	06
Register address Hi	00	Output address Hi	00
Register address Lo	01	Output address Lo	01
Register value Hi	00	Output value Hi	00
Register value Lo	03	Output value Lo	03

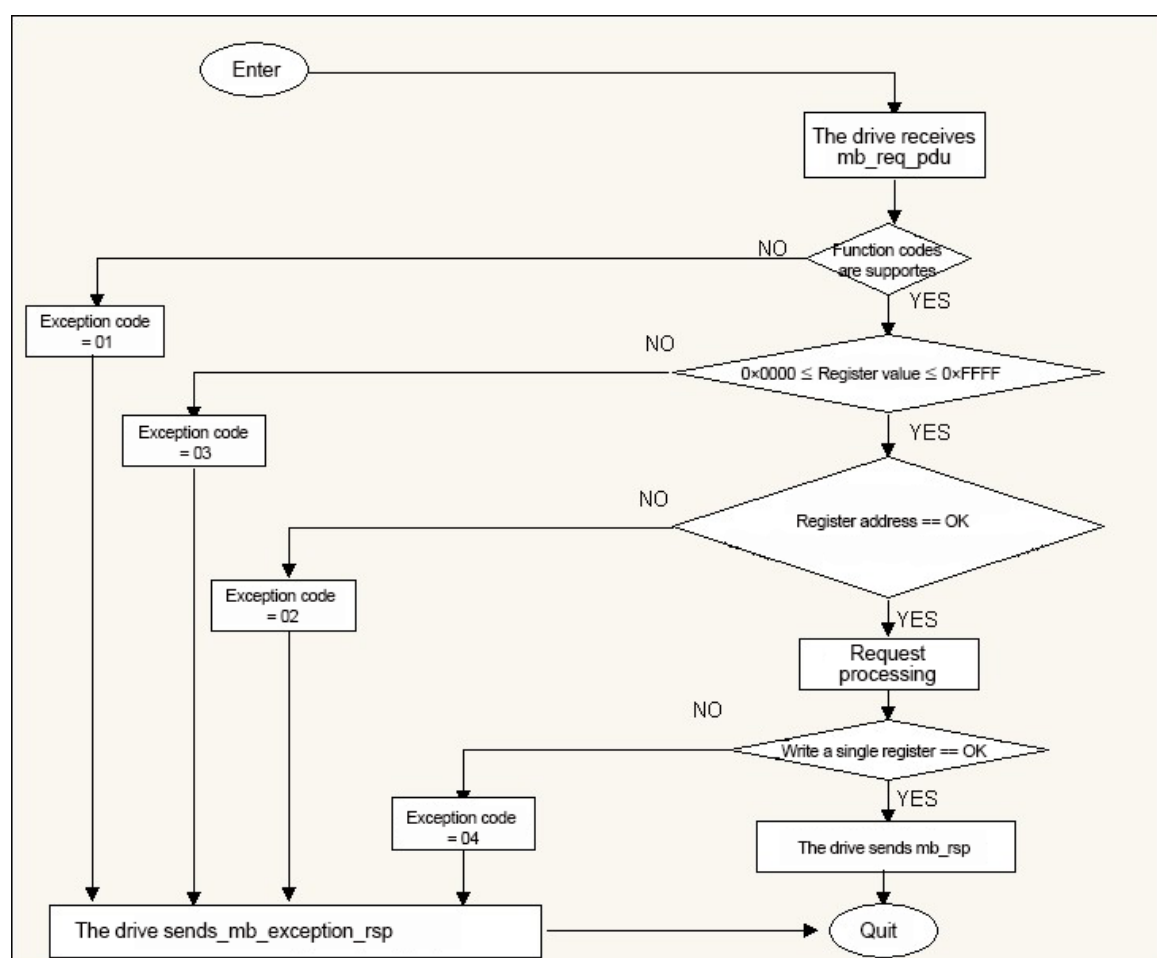


Figure 6 : Write a single register state diagram

12.3 16 (0x10) Write Multiple Registers

In a remote device, use this function code to write blocks of continuous registers (1 to about 120 registers).

The value written by the request is described in the request data field. Each register divides the data into two bytes.

The normal response returns the function code, start address, and number of registers to be written.

Request

Function code	1 byte	0x10
Start address	2 bytes	0x0000 to 0xFFFF
Number of registers	2 bytes	0x0001 to 0x0078
Number of bytes	1 byte	2×N*
Register value	N× 2 bytes	value

Response

Function code	1 byte	0x10
Start address	2 bytes	0x0000 to 0xFFFF
Number of registe	2 bytes	1 to 123(0x7B)

*N = Number of registers

Mistake

Error code	1 byte	0x90
Exception code	1 byte	01 or 02 or 03 or 04

Here is an instance of a request to write hexadecimal 00 0A and 01 02 to two registers starting with 2:

Request		Response	
Domain name	(hexadecimal)	Domain name	(hexadecimal)
function	10	function	10
Start address Hi	00	Start address Hi	00
Start address Lo	01	Start address Lo	01
Number of registers Hi	00	Number of registers Hi	00
Number of registers Lo	02	Number of registers Lo	02
Number of bytes	04		
Register value Hi	00		
Register value Lo	0A		
Register value Hi	01		
Register value Lo	02		

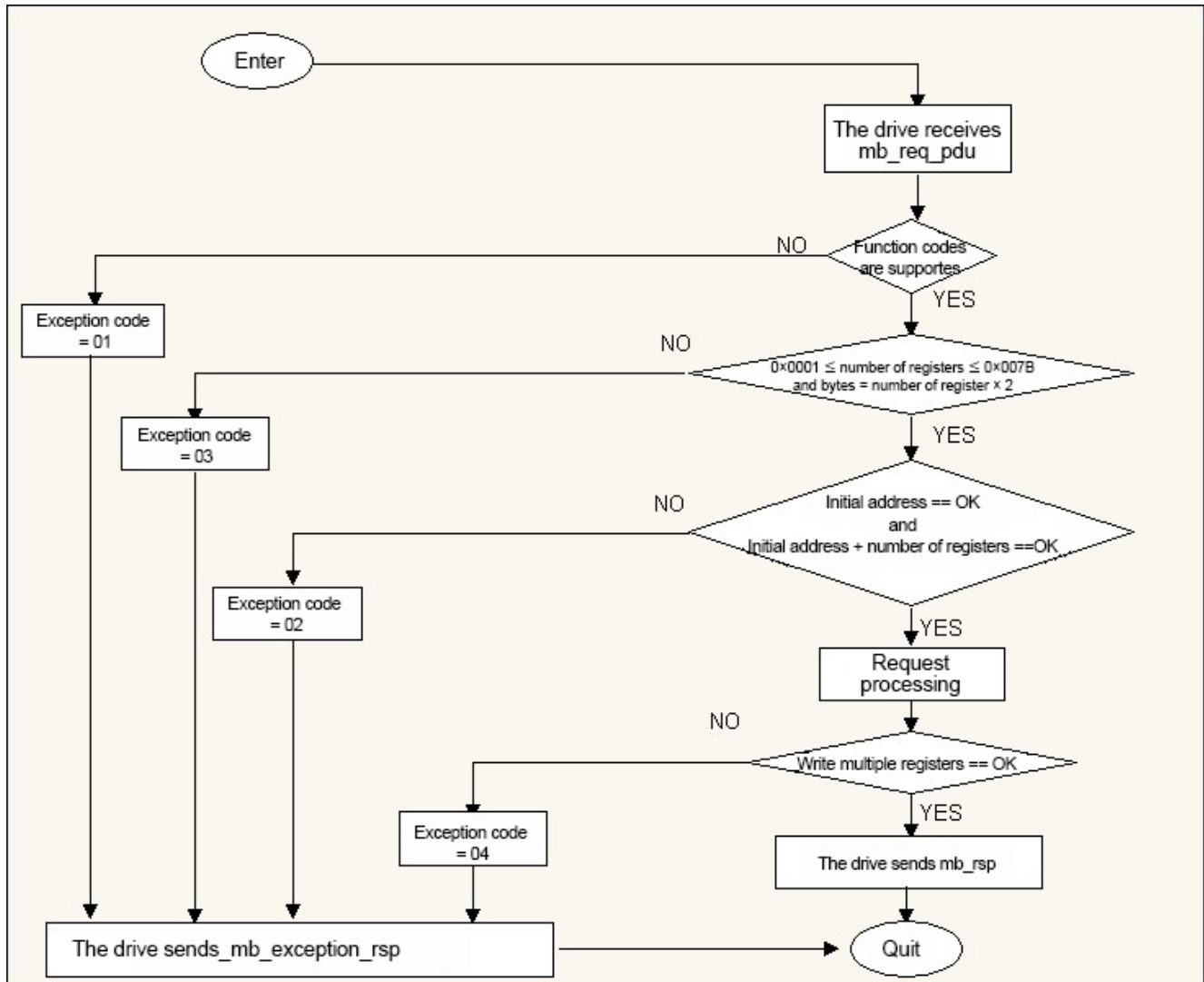


Figure 7: Writing a multi-register state diagram

List of exception codes:

MODBUS exception code		
Code	Name	Meaning
01	Illegal features	For the server (or slave), the function code received in the query is not an allowable operation. This may be because the function code is only applicable to the new device and is not possible in the selected unit. At the same time, it is also pointed out that the server (or slave) handles such a request in an error state, for example: because it is not configured and asks to return a register value.
02	Illegal data address	The data address received in the inquiry is not permissible to the server (or slave). In particular, the combination of reference number and transmission length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 will succeed, and a request with offset 96 and length 5 will result in exception code 02.

03	Illegal data values	The value included in the query is not permissible to the server (or slave). This value indicates a failure in the remaining structure of the combined request, for example, the implied length is incorrect. It does not mean that, because the MODBUS protocol does not know the significance of any special value of any particular register, the data item in the register that is committed for storage has a value that is not expected by the application.
04	Slave equipment failure	An unrecoverable error occurs while the drive is trying to perform the requested operation.
05	Confirm	Use with programming commands. The server (or slave) has already accepted the request and is processing it, but it takes a long duration for these operations. Returning this response prevents timeout errors from occurring on the client (or master). The client (or master) can continue to send polling program completion messages to determine whether processing is complete.
06	The slave is busy	Use with programming commands. The server (or slave) is processing program commands for long durations. When the server (or slave) is idle, the user (or master) should retransmit the message later.
08	Store parity errors	Used with function codes 20 and 21 and reference type 6 to indicate that the extent does not pass the consistency check. The server (or slave) attempted to read the log file, but found a parity error in memory. The client (or master) can resend the request, but can request service on the server (or slave) device.
0A	Gateway paths are not available	Used with a gateway to indicate that the gateway cannot assign an internal communication path from an input port to an output port for processing requests. This usually means that the gateway is misconfigured or overloaded.
0B	The gateway target device response failed	Used with a gateway to indicate that no response is obtained from the target device. Usually means that the device is not on the network.
0C	Send timeout between bytes	If the time between the previous byte and the next byte sent is greater than 1.5 characters in the same frame of data sent, an error occurs, the current frame of data is discarded, waiting for 3.5 characters to not receive the data sending error code, if there is data sending until 3.5 characters are not received and then send the error code, the data received in between is discarded.
OD	Sending between frames is less than the minimum interval	If a frame of data is successfully received and the data is received within less than 3.5 characters, an error occurs, waiting for the data transmission error code not received within 3.5 characters, if there is data transmission all the time, it is necessary to wait until there is no data received within 3.5 characters and then send the error code, and the data received in between them is discarded.

13. MODBUS Master Node Working Mode



The master node makes Modbus requests to child nodes in two modes:

a. In unicast mode, the master node accesses a child node at a specific address, and after the child node receives and processes the request, the child node returns a packet (a reply) to the master node. In this pattern, a Modbus transaction consists of 2 packets: a request from the master node and a reply from a child node.

Each child node must have a unique address (1 to 247) so that it can be addressed independently from other nodes.

b. In broadcast mode, the master node sends requests to all child nodes.

No reply is returned for the request broadcast by the master node. Broadcast requests are typically used to write commands. All devices must accept write capability for broadcast mode. Address 0 is specifically used to represent broadcast data.

14. MODBUS Address Rules



The Modbus addressing space has 256 different addresses.

0	1-47	48-255
broadcast address	Child node separate address	retain

Address 0 is reserved as the broadcast address.

All child nodes must recognize the broadcast address.

Modbus masters have no addresses, only child nodes must have an address. The address must be unique on the Modbus serial bus.



Notes

Correspondence address = value of DIP switch + 1 (drive address cannot be 0)

15. Master / Slave Communication Timing Diagram

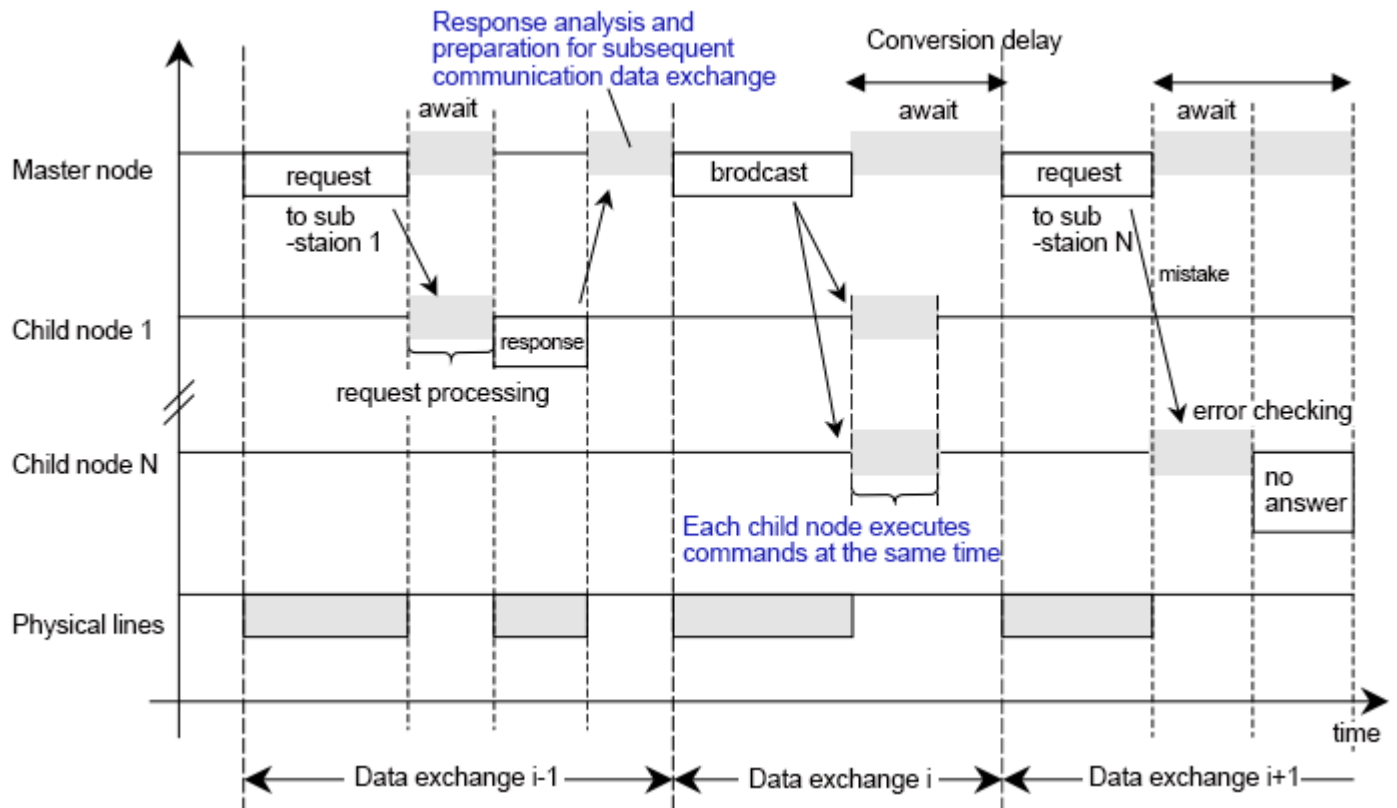


Figure 8: Master/slave communication timing diagram for various scenarios

16. RTU Transmission Mode



The format of each byte (11 bits) in RTU mode is:

Encoding system: 8-bit binary

Each 8-bit byte in the message contains two 4-digit hexadecimal characters (0–9, A–F)

Bits per Byte: 1 starting bit

8 data bits, the least significant bit is sent first

1 bit as parity

1 Stop bit

Modbus message RTU frame

The Modbus message is constructed by the transmitting device as a frame with known start and end tags. This allows the device to receive a new frame at the beginning of the message and to know when the message ends. Incomplete messages must be detectable and error flags must be set as a result. In RTU mode, message frames are distinguished by idle intervals of at least 3.5 characters in length. In the following sections, this timeframe is called $t_{3.5}$.

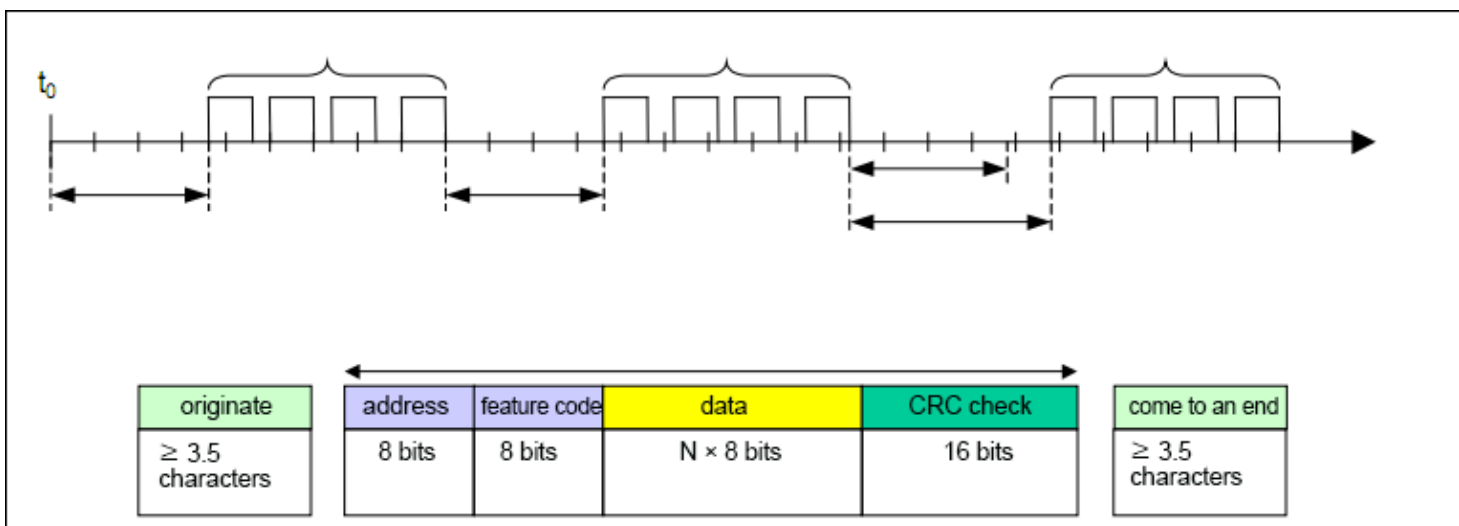


Figure 9: RTU message frame

The entire message frame must be sent in a continuous stream of characters.

If the idle interval between two characters is greater than 1.5 character time, the message frame is considered incomplete and should be discarded by the receiving node.

baud rate	Message frame free separator	Spaces between bytes	Response timed out	Conversion delay
≥19200 bps	≥2ms	≤0.8ms	1s	200ms
14400 bps	≥2.7ms	≤1.1ms	1s	200ms
9600 bps	≥4ms	≤1.7ms	1s	200ms
4800 bps	≥8ms	≤3.4ms	1s	200ms
2400 bps	≥16ms	≤6.8ms	1s	200ms

17. CRC Check



CRC contains a 16-bit value consisting of two 8-bit bytes.

The CRC field is appended to the message as the last domain of the message. After calculation, the low byte is appended first, followed by the high byte. The CRC high byte is the last subsection of the message sent.

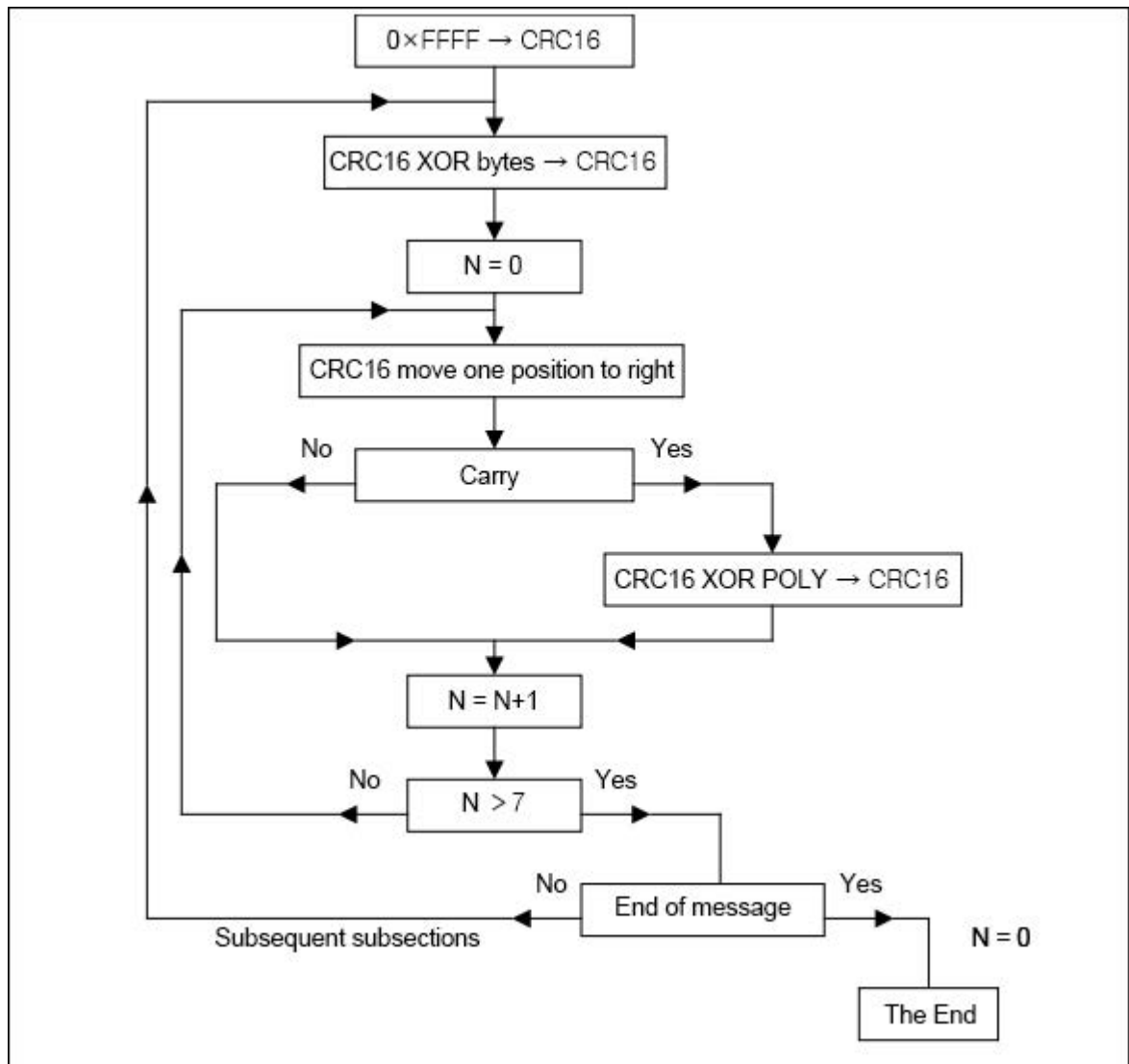


Figure 10: CRC 16 calculation algorithm

XOR = XOR

N = byte of information bits

POLY = CRC 16 Polynomial Computation = 1010 0000 0000 0001

(Generate a polynomial = 1 + x¹⁷ + x¹⁶ + x¹⁵ + x¹⁴ + x¹³ + x¹² + x¹¹ + x¹⁰ + x⁹ + x⁸ + x⁷ + x⁶ + x⁵ + x⁴ + x³ + x² + x + 1)

In CRC 16, the first byte sent is the low byte.

The function takes two parameters:

unsigned char *puchMsg; A pointer to a buffer containing the binary datagram used to generate the CRC

unsigned short usDataLen; The number of bytes in the message buffer

CRC generation function

unsigned short CRC16 (puchMsg,usDataLen) /* The function is returned as an unsigned short type CRC */

unsigned char *puchMsg ; /* The message used to calculate the CRC */

unsigned short usDataLen ; /* The number of bytes in the message */

```
{
    unsigned char uchCRCHi = 0xFF ; /* High-byte initialization of the CRC */
    unsigned char uchCRCLo = 0xFF ; /* Low-byte initialization of the CRC */
    unsigned ulIndex ; /* CRC queries table indexes */
    while (usDataLen--) /* Complete the entire packet buffer */
    {
        ulIndex = uchCRCLo ^ *puchMsgg++ ; /* Calculate the CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[ulIndex] ;
        uchCRCHi = auchCRCLo[ulIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-byte tables

/* The CRC value of the high-order byte */

```
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1,
0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

```

0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00,
0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1,
0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81,
0x40
};

```

Low-byte table

```

/* The CRC value of the low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB,
0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE,
0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2,
0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
0x36, 0xF6, 0xF7,

```

```
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E,  
0xFE, 0xFA, 0x3A,  
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B,  
0x2A, 0xEA, 0xEE,  
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27,  
0xE7, 0xE6, 0x26,  
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,  
0x63, 0xA3, 0xA2,  
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD,  
0x6D, 0xAF, 0x6F,  
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8,  
0xB9, 0x79, 0xBB,  
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4,  
0x74, 0x75, 0xB5,  
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,  
0x50, 0x90, 0x91,  
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94,  
0x54, 0x9C, 0x5C,  
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59,  
0x58, 0x98, 0x88,  
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D,  
0x4D, 0x4C, 0x8C,  
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,  
0x41, 0x81, 0x80,  
0x40  
};
```


18. Line-MODBUS Definition



MODBUS solutions on serial links should implement a "2-wire" electrical interface in accordance with the EIA/TIA-485 standard.

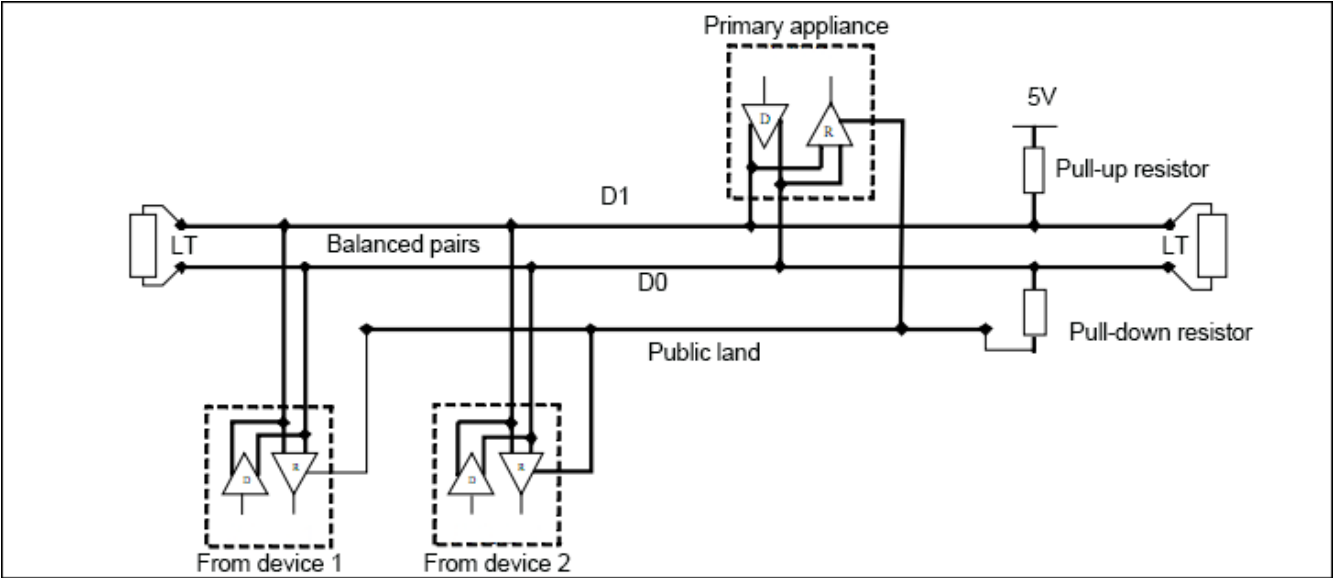


Figure 11: 2-Line general topology



+86-0519-8517 7825



+86-0519-8517 7807



Building 1#,355 Longjin Road, Changzhou
Economic Development Zone, Jiangsu, China



www.dingsmotion.com

International Customer

Person in Charge :

Daniel Jang

daniel@dingsmotion.com

Building 1#, 355 Longjin Road,
Changzhou Economic Development
Zone, Jiangsu, China

+86-519-85177826, 85177827

North America Customer

Person in Charge :

Nicolas Ha

sales@dingsmotionusa.com

335 Cochrane Circle Morgan Hill,
CA 95037

+1-408-612-4970

China Customer

Person in Charge :

Sweet Shi

info@dingsmotion.com

Building 1#, 355 Longjin Road,
Changzhou Economic Development
Zone, Jiangsu, China

+86-519-85177826, 85177827