Mindman

DRIVER

CM20 series

Instruction Manual

Closed-loop stepping motor driver

Version 202303

Contents

1	Specifications	3
2	Preparations	4
3	Connector Pin Assignments	4
4	Input schematic diagram	9
5	Output schematic diagram	10
6	Timing chart	11
7	Indicator lights	11
8	Dimension (Unit: mm)	12
9	Parameters	13
10	MODBUS Transaction	26
11	Data Encoding	28
12	Public Function Code Definition	28
13	MODBUS master node protocol principle	36
14	MODBUS addressing rules	36
15	Master / Slave communication time diagram	37
16	RTU transmission mode	37
17	CRC checking	39
18	Two-Wire MODBUS Definition	42
19	Appendix. Wiring Diagram Example of MITSUBISHI FX3U Series	43

1 Specifications

Item	Description	Notes
Model Number	CM20	
Input Power Voltage	DC24V-48V	
Max. Output Current	4.5A (peak)	
Motor	2-phase bipolar stepper motor with encoder	
Driver Method	PWM constant current driving	
Interface	Input • PUL/DIR (Also available as digital input) • 5 digital inputs • Encoder input (A, B, Z) Output • 4 digital outputs • Encoder output (Differential A, B, Z)	Free assignment for every input/output via communication except Encoder output
Detail of Digital Inputs	/SV ON (Motor Enabled) /RESET (Alarm reset) /START (Motor start/stop) /JOG (Motor Jogging) /HOME (Homing)	
Detail of Digital Outputs	/IN POTISION /ALARM	
LED Indicators	Status, Alarm	2 indicator lights
Communication I/F	RS485 with maximum 32 axis available	Modbus RTU protocol Default Baud: l9200bps (Adjustable by parameter)
Control Mode	Position Control	Positioning with pulse command or via RS485
Dimension (mm)	W86.6 x D145.9 x H28.4	Connectors excluded
Weight	About 350g	Connectors excluded
Operating temperature/humidity	0-45℃, less than 85%RH	No condensation
Storage temperature/humidity	0-85℃, less than 85%RH	No condensation
Atmosphere	Avoid the corrosive gases	

Specifications of the Motor

Model		□20	□25	□28	□35	□42	□56
Series	-	BM					
Drive Method	-	Bi-Polar					
Number of Phases	-	2					
Current per Phase	A	0.6	1	1	1.5	2	3
Holding Torque	N.m	0.036	0.085	0.085	0.28	0.51	1.53
Rotor Inertia	g·cm²	2.9	8	8	40	75	490
Weights	g	70	120	120	300	400	1150
Insulation Resistance	Mohm	100 MIN.(at 500VAC)					
Insulation Class	-	Class B (130℃)					
Operating Temperature	℃	0~50					
Encoder	Incremental Optical Encoder	6,400 PPR	9,600 PPR	9,600 PPR	12,800 PPR	16,000 PPR	16,000 PPR

2 Preparations

Please complete below items before power up the driver.

2.1 Connections

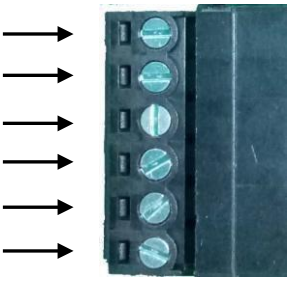
Please wire securely while see the connector pin assignment table below.

A. CN1: Connect a power supply to a motor
   Please connect a power supply to the motor properly. Especially note that the driver might be damaged if the user connects output terminals of the motor to the power supply. Please use the wire which is 20 AWG at least.
B. CN2: To an encoder
C. CN3: To an interface signal
   Please distribute necessary digital input/output signals. These common inputs/outputs are opto-isolated (except Differential Encoder Output signal).
   Please prepare the isolated power input (+24V) additionally.
D. CN4: RS485 wiring
   Please build the connection with the RJ45 connectors.

3 Connector Pin Assignments

3.1 CN1 (Power & Motor)

Pin.	Signal name
6	Motor B-
5	Motor B+
4	Motor A-
3	Motor A+
2	Power GND
1	Power V+ (DC24V or48V)

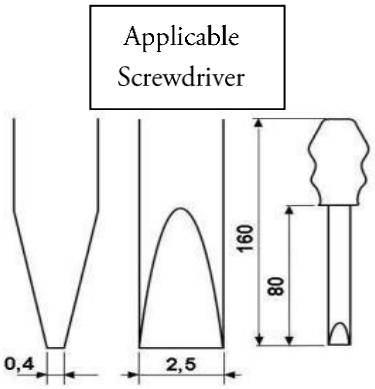


Note: Please carefully observe voltage and polarities. Do NOT connect the power backwards.

Applicable wire size: AWG20~AWG16 (stranded wire)

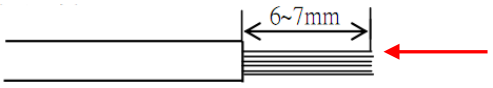
Please use a bladed type screwdriver (0.4*2.5mm size) for tightening the screws on the terminal block. (e.g. SZS 0.4*2.5 VDE – 1205037 made by Phoenix Contact)

Tightening torque: 0.22~0.25N·m (2.3kgf·cm~2.5kgf·cm)



Wiring:

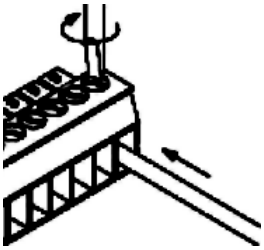
A. Insulation stripping length: 6~7mm



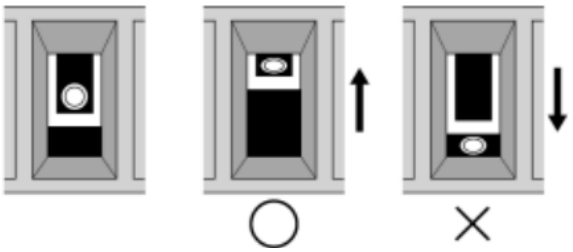
No advanced soldering (It may cause a connection error)

B. Insert the wires into the terminal block, and then tighten the screws.
Applicable tightening torque: 0.22~0.25N·m (2.3kgf·cm~2.5kgf·cm)

Notes



- Please strip the wire carefully to avoid nicking, cutting, or damaging the strands.
- While inserting the wires into the terminal block, please do not twist the stripped wires. Also, the wires which are stripped too long may cause a short circuit because it allows an extra bare area after they've been inserted into the terminal block.
- Please direct insert the wires into the terminal block without soldering to avoid the disconnection due to the vibration.
- Please do not increase extra stress on the wires.
- Please turn the screwdriver clockwise when fasten down the screw of the terminal block to avoid a loose contact.



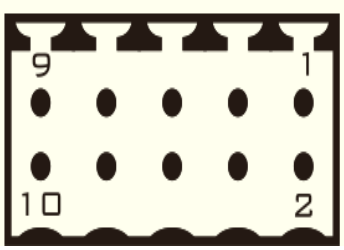
Turning clockwise Turning counterclockwise

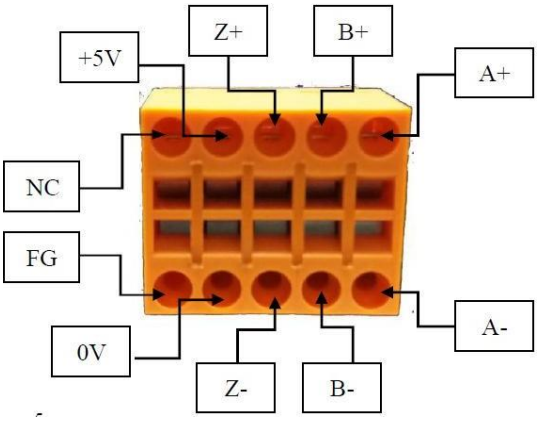
3.2 CN2 (Encoder IN)

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

Front view of encoder connector

Note: please carefully observe voltage and polarities. Do NOT connect the power backwards.

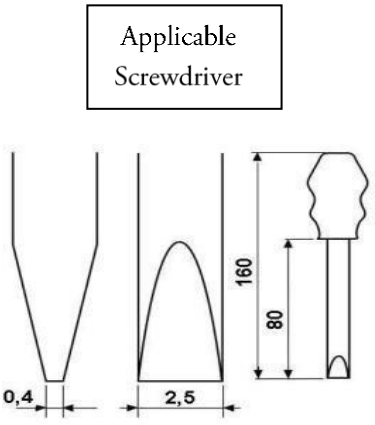




Applicable wire size: AWG28~AWG18 (Stranded wire)

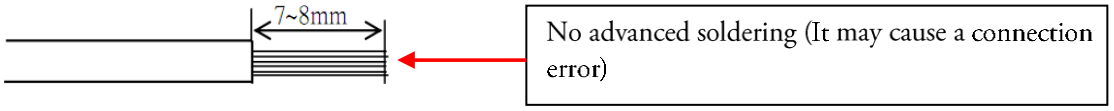
Easy and convenient insertion thanks to push-in connection

Please use a bladed type screwdriver (0.4*2.5mm size) as a tool. (e.g. SZS 0.4*2.5 VDE – 1205037 made by Phoenix Contact)



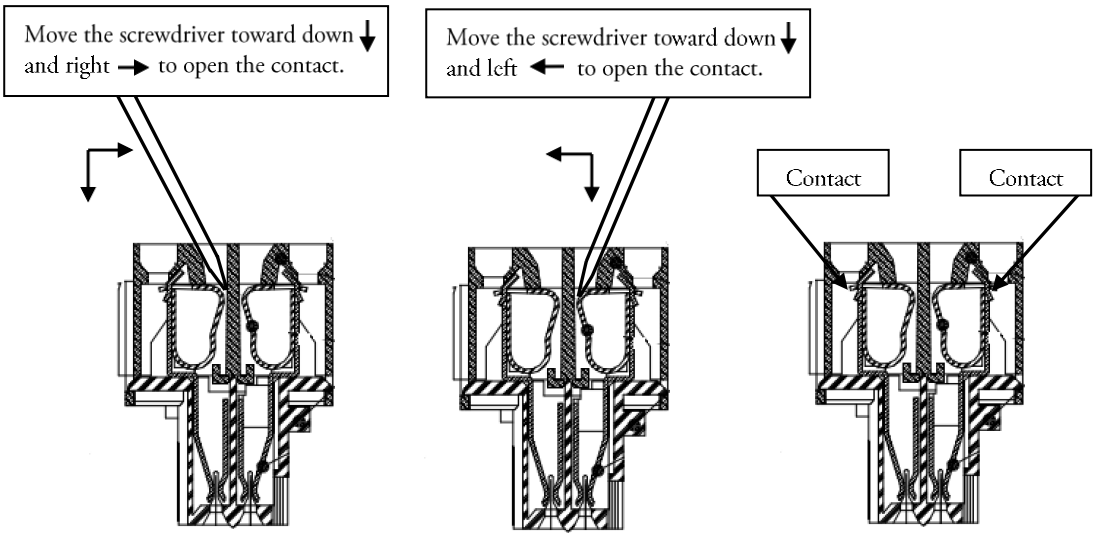
Wiring:

A. Insulation stripping length: 7~8mm

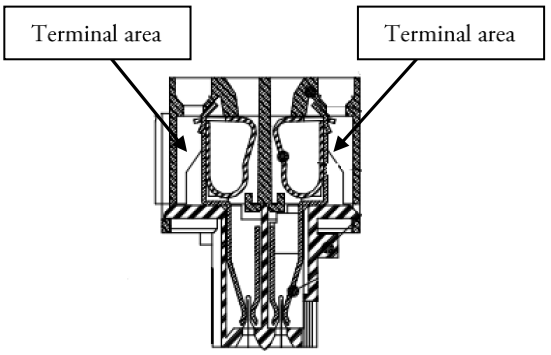


Convenient insertion, thanks to direct Push-in connection technology.

B. Use the screwdriver to open the contacts.



C. Insert the stripped end of each wire fully into its intended terminal. Ensure that no bare wire strands extends from the terminal. And then remove the screwdriver.



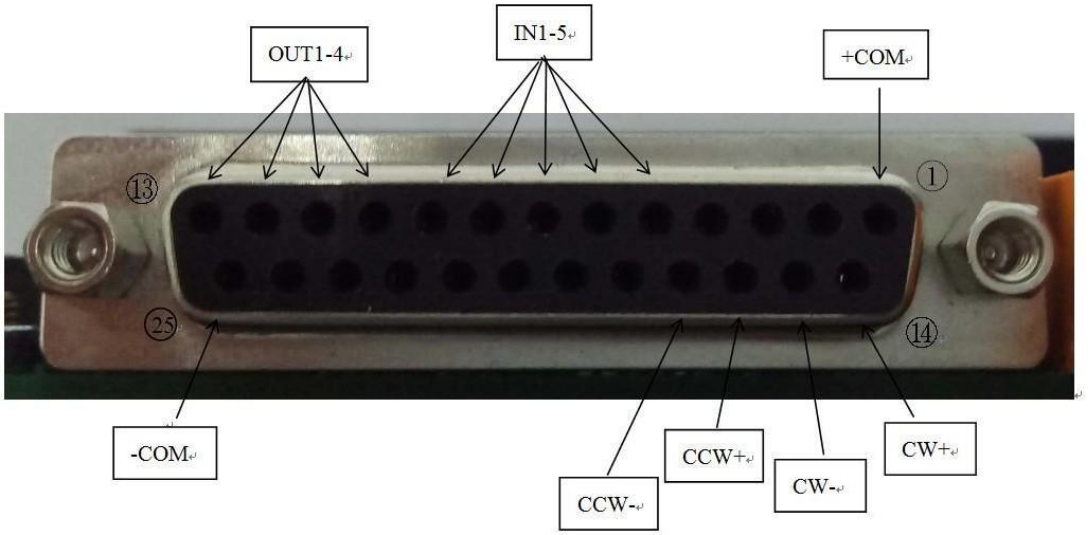
Notes

- Please strip the wire carefully to avoid nicking, cutting, or damaging the strands.
- While inserting the wires into the terminal block, please do not twist the stripped wires. Also, the wires which are stripped too long may cause a short circuit because it allows an extra bare area after they've been inserted into the terminal block.
- Please direct insert the wires into the terminal block without soldering to avoid the disconnection due to the vibration.
- Please do not increase extra stress on the wires.
- Please use the proper screwdriver to open the contacts of the terminal block to avoid damaging its spring.

3.3 CN3 (I/O)

No	Color	Name	No	Color	Name	No	Color	Name
1	Brown/Black	+COM (24V)	10	White/Red	OUTPUT 1	19	Blue	Encoder A-
2	Red/Black	NC	11	L. Blue/Black	OUTPUT 2	20	Purple	Encoder B+
3	Orange/Black	NC	12	L. Green/Black	OUTPUT 3	21	Gray	Encoder B-
4	Yellow/Black	NC	13	Pink	OUTPUT 4	22	White	Encoder Z+
5	Green/Black	INPUT 1	14	Brown	CW+ / Pulse+	23	Black	Encoder Z-
6	Blue/Black	INPUT 2	15	Red	CW- / Pulse-	24	L. Blue	FG
7	Purple/Black	INPUT 3	16	Orange	CCW+ / Dir+	25	L. Green	-COM (0V)
8	Gray/Black	INPUT 4	17	Yellow	CCW- / Dir-			
9	White/Black	INPUT 5	18	Green	Encoder A+			

Front view of the I/O female connector.

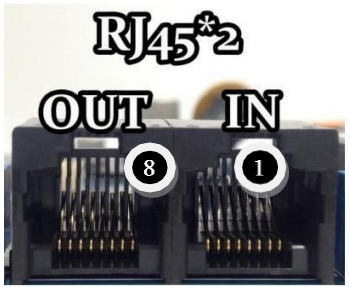


3.4 CN4 (IN) / CN5 (OUT) (RS485)

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

RJ45 type x 2

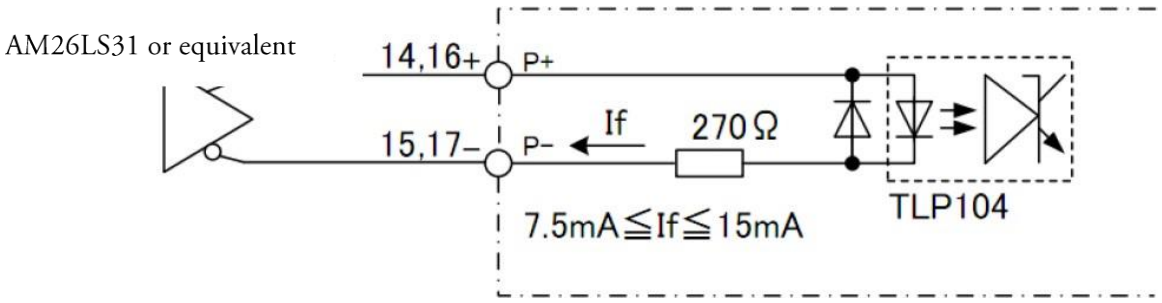
Front view of the socket



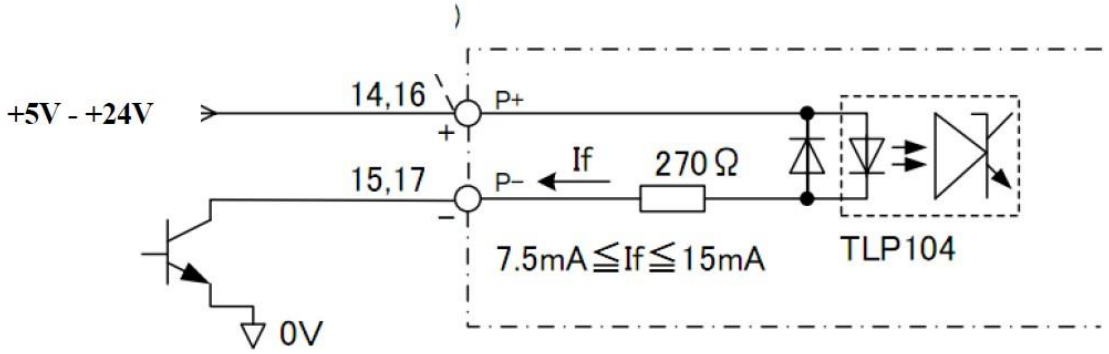
Important Note:
In the case of connecting several drivers, please enable the terminal resistor in the last device by making a short circuit between its Pin3 & Pin8, and Pin6 & Pin7 of CN5.
Ps. Terminal resistor cannot be wired with CN4.

4 Input schematic diagram

A. Pulse command input circuit (Differential / Line driver)

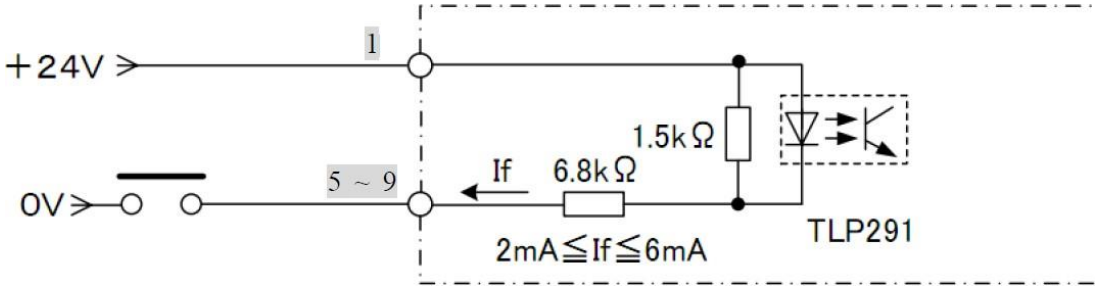


B. Pulse command input circuit (Open collector)

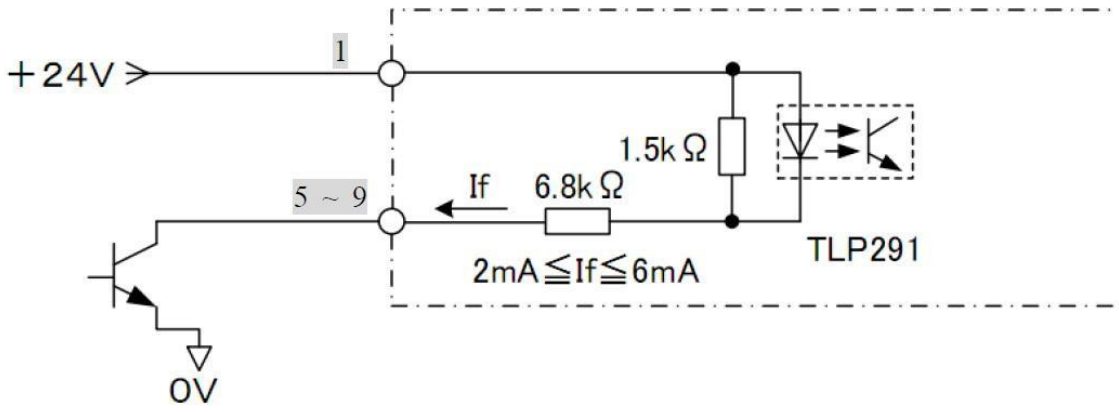


Note: CM-20 is compatible both with +5V & +24V signal, therefore, the user doesn't need to connect a resistor in series additionally.

C. Sensor, digital input circuit (Contacts)

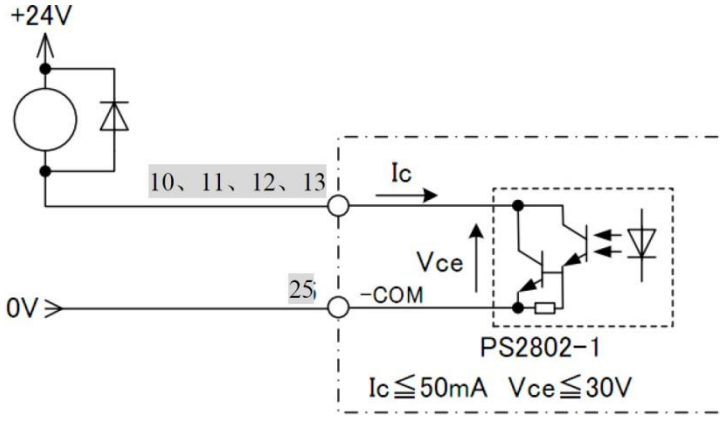


D. Sensor, digital input circuit (Open collector)



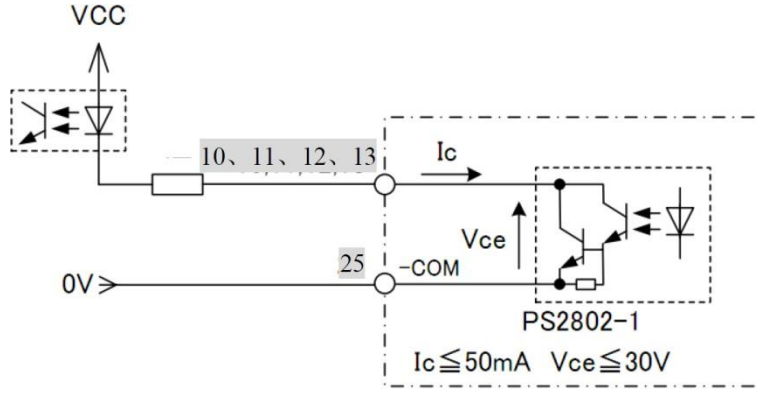
5 Output schematic diagram

A. Digital output circuit (Relay)

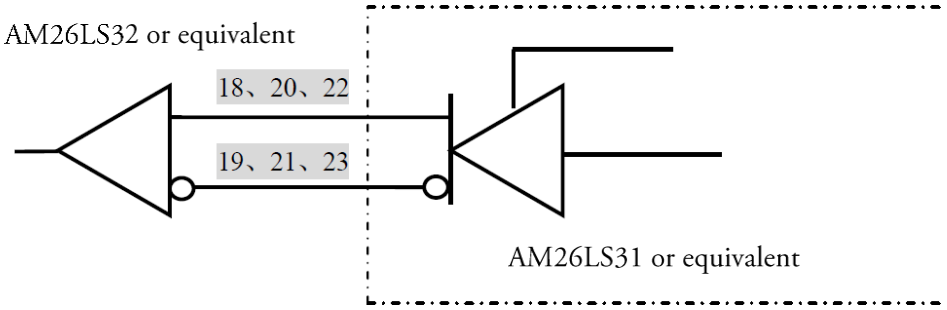


Important note: Connecting a diode (1N4007 or equivalent) on both side of the relay additionally is required when the user is connecting a relay.

B. Digital output circuit (Photo-coupled)

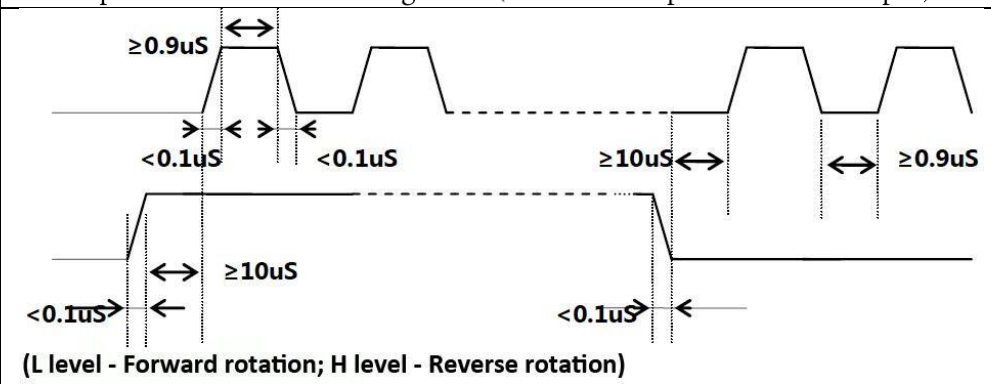
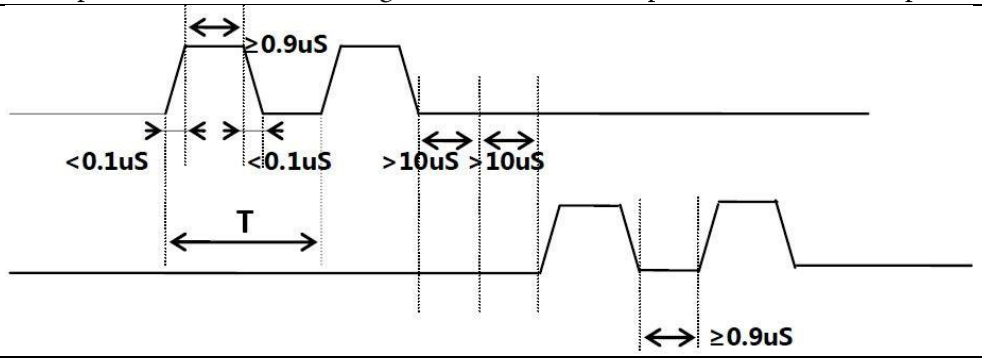


C. Encoder output circuit (Differential/ Line driver)



Important note: Since Encoder output is NOT opto-coupled, please make sure that the wiring is correct before power-up the driver. Also make sure that there is no short circuit connected with +24V power of CN3 to avoid damaging the master device and the driver.

6 Timing chart

	The input waveform and timing chart (One-clock input or Pul/Dir input)
STEP Input DIR Input	
	The input waveform and timing chart (Two-clock input or CW/CCW input)
STEP Input (Forward) DIR Input (Reverse)	

7 Indicator lights

A. Status

To indicate the status of the driver, the indicator may flick (Low level for 0.5 second & High level for 0.5 second). A flickering sequence is ended by high level for 2 seconds and repeats.

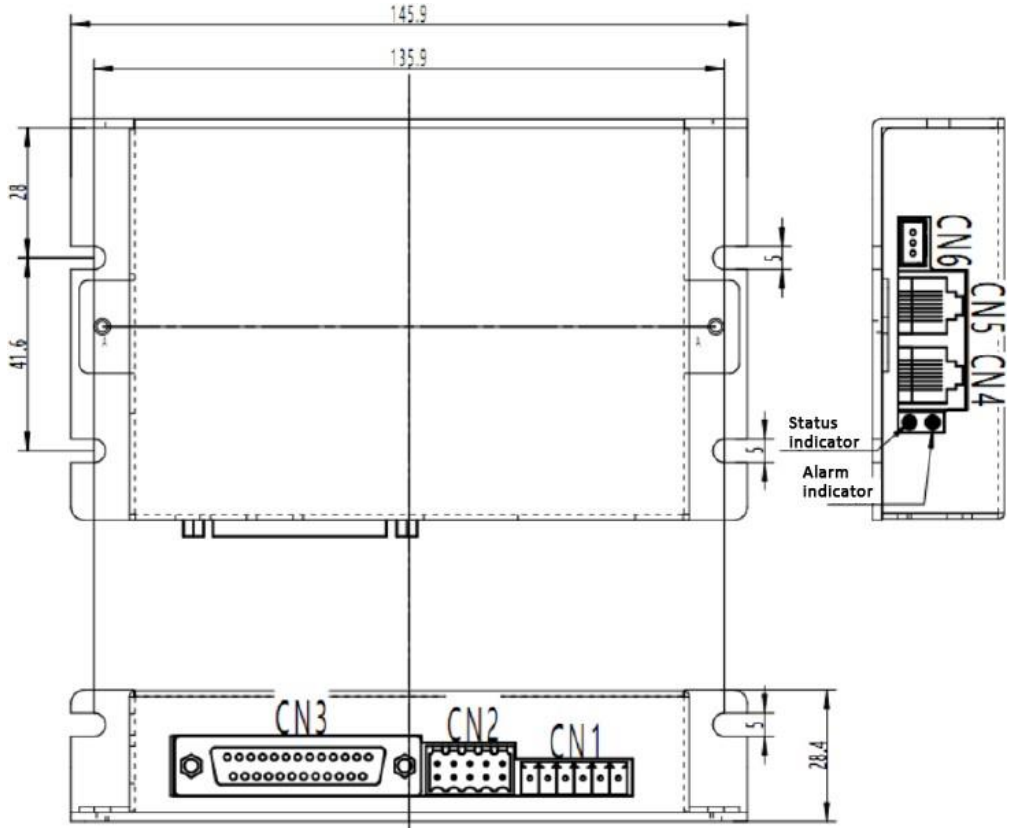
Status	Green Indicator	Communication Code	Description
Stopped	Flickering	2	The motor is enabled but not rotating.
Working	Steady light	3	The driver is working
Disabled	Flickering	1	The motor is disabled and in a free state.

B. Alarm

To indicate the status of the driver, the indicator may flick (Low level for 0.5 second & High level for 0.5 second). A flickering sequence is ended by high level for 2 seconds and repeats.

Alarm type	Red Indicator	Communication Code	Description
Over current	Flashing once	10	Over current/the drvier was out of order
Disconnected motor	Flashing twice	11	Motor has been disconnected from the driver.
Over voltage	Flashing 3 times	14	Input voltage is over 60V.
Low voltage	Flashing 4 times	13	Input voltage is under 18V.
Position error	Flashing 5 times	25 or 26	25: The position error is larger than tolerance set by the user. 26: Overload, 150% motor current had been output continuously for more than 2 second.
TBD	TBD	TBD	

8 Dimension (Unit: mm)



Name of parts

Symbol	Usage
CN1	Power supply / motor connector
CN2	Encoder connector
CN3	Interface connector
CN4	RS485 connector (IN)
CN5	RS485 connector (OUT)
CN6	Reserved

9 Parameters

9.1 Basic status of the controller (Sort 1)

adr	word	Parameter	Description	Range/Unit
0100	1	Motor current	Current value of motor current	0.1%/A
0101	1	Input voltage	Current input voltage	1%/V
0104	2	Micro-step subdivisions	To set micro-step subdivisions	Ppr
0106	1	Pulse method	1:1-clock input, 2: 2-clock input	1-2
0108	1	Alarm code	To indicate which kind of alarm is triggered. Please refer to 1-2 for details. "0" means no error/alarm occurred.	-
0109	1	Status	Driver status. Please refer to 1-1 for details	-
0110	1	Hardware version	To display hardware version of the driver	-
0111	1	Software version	To display software version of the driver	-
0117	2	Position	Target position	pulse
0119	1	Actual speed	-	0.01rps
0126	1	Actual position	Current position during operation	pulse
0174	1	Specify executing program section via I/O	-	-
0176	1	Program error No.	-	-
0178	1	Program No.		

9.2 Basic parameters (Sort 02)

adr	word	Parameter	Description	Range/Unit
0201	l	Motor direction	To determine motor rotating direction and encoder direction: If Bit1 = 0, encoder direction is the same; if Bit1 = 1, encoder direction changes. If Bit0 = 0, rotating direction is the same; if Bit0 = 1, rotating direction changes. Configured setting will be available after a power cycle.	0-1
0202	1	Activated edge of pulse signal	To specify the activated edge of pulse signal 0: Falling edge 1: Rising edge Configured setting will be available after a power cycle.	0-5
0213	l	Target current for automatic reduction	To specify the current applied when the motor is stopped. (Available in Open-loop mode)	l0%-120%
0217	l	Motor control	0: Open-loop control 1: Closed-loop control (Default = 1)	0- 1
0224	1	Micro-step emulation	The smaller value, the smoother motor motion. Side effect: A delay/lag	1-700
0234	1	Digital Filter	Filter coefficient of input pulse. The bigger value means less input frequency.	l - 15
0241	l	Input current	To set the current	l 00-4500 0.lA-4.5A
0242	2	Micro-Step subdivision	To specify the pulses per motor revolution	200-102400 Ppr
0244	1	Pulse input method	1: PUL & DIR (1 clock input) 2: CW & CCW (Double clock input)	l -2
0245	1	Time of current reduction	To specify the delay time after the motor is stopped and then the current reduction starts.(Available in Open-loop mode)	1-32767 US
0296	l	Control method	0: External pulse (Default) 1: Internal pulse (A power cycle is required after modifying this parameter)	0- 1

0298	1	Communication address	Default value is 1	1~255
0299	2	Baud for serial communication	Default value is 19200	1600~115200

9.3 Closed loop control related parameters (Sort 04)

adr	word	Parameter	Description	Range/Unit
0246	1	Encoder resolution	Resolution = Encoder counts x 4	200~65535
0247	2	Width of in-position pulse	The in-position signal outputs when the motor is approaching target position (Default value=0)	1~1000 Encoder resolution
0251	1	Velocity loop Kp	Velocity loop Kp	0~30000
0252	1	Velocity loop Ki	Velocity loop Ki	0~30000
0255	1	Position loop Kp	Position loop Kp	0~30000
0258	1	Threshold for in-position	The unit is based on encoder resolution	0~30000 Encoder resolution

9.4 Control parameters (Sort 05)

adr	word	Parameter	Description	Range/Unit
0301	1	Starting speed	Default value: 100	1~2000 0.01~20rps
0302	1	Stopping speed	Default value: 100	1~2000 0.01~20rps
0303	1	Acceleration	Default value: 100	5~10000 rps^2
0304	1	Deceleration	Default value: 100	5~10000 rps^2
0305	1	Homing mode	To specify the method when executing homing. 0: CW homing (Default) 1: CCW homing 2: Plus Limit 3: Minus Limit 8: Plus direction to Z phase 9: Minus direction to Z phase	0~10
0306	1	Constant moving speed	Default value:1000	1~5000 0.01~50rps
0307	1	Moving speed in speed control mode	In speed mode, the motion direction would be the same with the direction of speed. Default value:1000	-5000~5000 -50~50rps
0308	1	JOG speed	Default value:1000	1~5000 0.01~50rps
0309	1	Homing speed	Default value:1000	1~5000 0.01~50rps

0310	1	Moving speed after reached Home	Moving speed after reached Home Default value:1000	l -5000 0.01-50rps
0311	2	Offset amount of homing	Default value: 0	-2000000000•2000000000 pulse
0313	2	Output pulse	Operation method In absolute positioning mode: Move to specified target position In relative positioning mode: Move with a specified amount Default value: 0	-2000000000•2000000000 pulse
0317	2	Plus software limit	Default value: 2000000000 Note: N/A during Homing sequence	-2000000000•2000000000 pulse
0319	2	Minus software limit	Default value: -2000000000 Note: N/A during Homing sequence	-2000000000•2000000000 pulse
0321	2	Set current position	Default value: 0	-2000000000•2000000000 pulse
0323	1	Control commands	0: Null (Default) 1: Perform an absolute move toward to specified position. The moving direction is defined by specifying + or — for moving amount, not by specifying + or — for moving speed. The target position can be changed even when the motor is in motion. 2: Perform a relative move with a specified distance. The moving direction is defined by specifying + or — for moving amount, not by specifying + or — for moving speed. It's NOT available for changing moving amount when the motor is in motion. 3: Speed mode 4: +Jog 5: -Jog 6: Deceleration Stop 7: Emergency Stop 8: Set current position. Setting is only available when the motor is stopped. 12: Homing 13: Reset alarm 14: Check the programming data 15: Save the programming data	0~29

<table>
<tr><td></td><td></td><td></td><td>16: Start the programming data
17: Pause the programming data
18: End the programming data</td><td></td></tr>
<tr><td>0324</td><td>1</td><td>Internal control switch</td><td><table><tr><td>Byte</td><td>Bit 1</td><td>Bit 0</td></tr><tr><td>Function</td><td>-limit</td><td>+limit</td></tr></table>
1: Enable
0: Disable (Default setting)

Configured setting will be available after a power cycle.</td><td>0-1</td></tr>
</table>

9.5 Input assignments (Sort 06)

adr	word	Parameter	Description	Range/Unit
0400	1	IN1 assignment	0: Null (Default) 1: Perform an absolute move toward to specified position. The moving direction is defined by specifying + or — for moving amount, not by specifying + or — for moving speed. The target position can be changed even when the motor is in motion. 2: Perform a relative move with a specified distance. The moving direction is defined by specifying + or — for moving amount, not by specifying + or — for moving speed. It's NOT available for changing moving amount when the motor is in motion. 3: Speed mode 5: -JOE 6: Deceleration Stop 7: Emergency Stop 8: Set current position. Setting is only available when the motor is stopped. 9: +Limit 10: -Limit 11: Home signal 12: Perform Homing 13: Reset alarm 14: Check the programming data 15: Save the programming data 16: Start the programming data	0-30

			17: Pause the programming data 18: End the programming data 20: Enable (Servo ON) 25: Assign Bit 0 to I/O ports for selecting programming data 26: Assign Bit 1 to I/O ports for selecting programming data 27: Assign Bit 2 to I/O ports for selecting programming data 28: Assign Bit 3 to I/O ports for selecting programming data 29: Assign Bit 4 to I/O ports for selecting programming data	
0401	l	IN2 assignment	Please refer to “Description” column of 0400 for details. (Default: 0)	0-30
0402	1	IN3 assignment	Please refer to “Description” column of 0400 for details. (Default: 0))	0-30
0403	l	IN4 assignment	Please refer to “Description” column of 0400 for details. (Default: 0)	0-30
0404	l	INS assignment	Please refer to “Description” column of 0400 for details. (Default: 0)	0-30
0405	1	IN6 assignment (CCW port)	Please refer to “Description” column of 0400 for details. (Default: 0) (Unavailable in external pulse mode)	0-30
0406	1	IN7 assignment (CW port)	Please refer to “Description” column of 0400 for details. (Default: 0) (Unavailable in external pulse mode)	0-30
0410	l	Pseudo port IN1	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400)	0-1
0411	l	Pseudo port IN2	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400)	0-1
0412	1	Pseudo port IN3	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400)	0-1
0413	l	Pseudo port IN4	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400)	0-1
0414	l	Pseudo port INS	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400)	0-1

0415	l	Pseudo port IN6	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400) (Unavailable in external pulse mode)	0- 1
0416	l	Pseudo port IN7	0: OFF (Default) 1: ON (Trigger the action specified from the list of 0400) (Unavailable in external pulse mode)	0- 1

9.6 Output assignments (Sort 07)

<table>
<tr><th>adr</th><th>word</th><th>Parameter</th><th>Description</th><th>Range/Unit</th></tr>
<tr><td>0420</td><td>1</td><td>OUT1 assignment</td><td>100: Common port
101: Alarm output:
The signal is output when there's no alarm. It won't be output when an alarm occurred (Default)
102: In-position signal
103: Servo ON/OFF state signal:
The signal is output in a Servo OFF state and it won't be output when the motor is enabled.</td><td>100-104</td></tr>
<tr><td>0421</td><td>1</td><td>OUT2 assignment</td><td>Please refer to "Description" column of 0420 for details. (Default:100)</td><td>100-104</td></tr>
<tr><td>0422</td><td>1</td><td>OUT3 assignment</td><td>Please refer to "Description" column of 0420 for details. (Default:100)</td><td>100-104</td></tr>
<tr><td>0423</td><td>l</td><td>OUT4 assignment</td><td>Please refer to "Description" column of 0420 for details. (Default:100)</td><td>100-104.</td></tr>
<tr><td>0428</td><td>l</td><td>Common digital output control</td><td>Output function (Default:100)
<table><tr><td>Byte</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>Output</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr></table></td><td></td></tr>
<tr><td>0430</td><td>1</td><td>Logic of digital outputs</td><td>Logic for output ports
<table><tr><td>Byte</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>Output</td><td>OUT4</td><td>OUT3</td><td>OUT2</td><td>OUT1</td></tr></table></td><td></td></tr>
</table>

9.7 Programming position mode (Sort 08)

Address range for programming control: 1024- 1536. Up to 256 points of data is available.

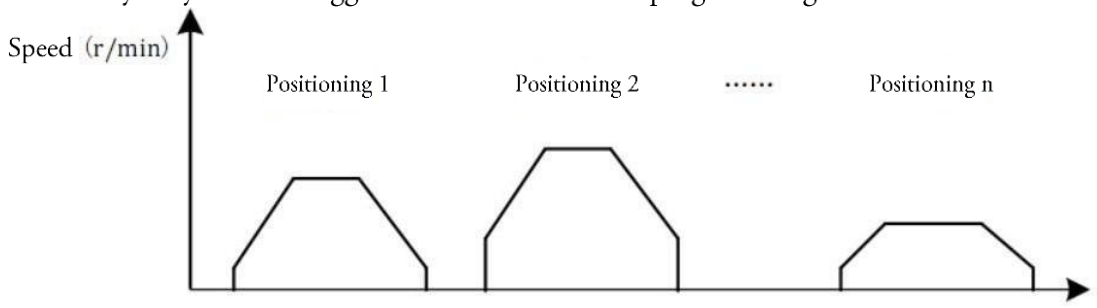
Format of programming control commands

Cmd code	word	Parameter	Description	Range/Unit
1	2	Absolute move	Parameter 1: Target position Default value: 0	-2147483647~ 2147483647 pulse

<table>
<tr><td>2</td><td>2</td><td>Relative move</td><td>Parameter 1: Moving distance/amount
Default value: 0</td><td>-2147483647~2147483647 pulse</td></tr>
<tr><td>3</td><td>3</td><td>Move with constant speed</td><td>Parameter 1: Moving speed
Default value: 0</td><td>-5000~5000
-50~50rps</td></tr>
<tr><td>6</td><td>0</td><td>Decelerated stop</td><td>-</td><td>-</td></tr>
<tr><td>8</td><td>2</td><td>Position setting</td><td>Parameter 1: Set position
Default value: 0</td><td>-2000000000~2000000000 pulse</td></tr>
<tr><td>12</td><td>1</td><td>Homing</td><td>Parameter 1: Homing method
Default value: 0
Please refer to the content stated in 0305 for details</td><td>0~10</td></tr>
<tr><td>51</td><td>1</td><td>Start speed</td><td>Default value: 100</td><td>1~2000
0.01~20rps</td></tr>
<tr><td>53</td><td>1</td><td>Stopping speed</td><td>Default value: 100</td><td>1~2000
0.01~20rps</td></tr>
<tr><td>54</td><td>1</td><td>Position setting speed</td><td>Default value: 1000</td><td>1~5000
0.01~50rps</td></tr>
<tr><td>61</td><td>1</td><td>Acceleration</td><td>Default value: 100</td><td>5~10000
rps^2</td></tr>
<tr><td>62</td><td>1</td><td>Deceleration</td><td>Default value: 100</td><td>5~10000
rps^2</td></tr>
<tr><td>65</td><td>2</td><td>Awaiting the jump</td><td>Par 3 (high 8 bits) – Par 2 (low 8 bits) / Par 1 (low 16 bits)
Par 1: Awaiting until the next command started to be executed. If the setting value is 0, then the device will keep awaiting until an I/O signal enabled
Par 2: The target of jumping once the
<table>
<tr><td>Bit7</td><td>Bit6</td><td>Bit5</td><td>Bit4</td></tr>
<tr><td>Matching status</td><td colspan="3">Specify the input port for matching (1~7)</td></tr>
<tr><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>Matching status</td><td colspan="3">Specify the input port for matching (1~7)</td></tr>
</table></td><td>-</td></tr>
<tr><td>66</td><td>2</td><td>Jump to</td><td>Par 1 (high 16 bits) / Par 2 (low 16 bits)
Par 1: Loop counts
Par 2: Jump target address</td><td>-</td></tr>
<tr><td>68</td><td>1</td><td>Common output ports</td><td>Par 1: Status of the output port</td><td>-</td></tr>
<tr><td>100</td><td>0</td><td>End the programming control</td><td>Adding the end code at the end of every section</td><td>-</td></tr>
</table>

Users set several steps in Programming position mode for programming positioning control which would be executed in order automatically by triggering a defined external I/O. The users can specify such as moving distance or acceleration/deceleration for every step and store these steps in EEPROM.

And then they only need to trigger the I/O to start these programming motions.



Programming positioning mode

Port configurations for every step of programming data

Bit4	Bit3	Bit2	Bit1	Bit0	Step
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

Programming-related assignments to I/O Ports

1. 25-29 can be used for determining a certain programming step: By assigning Bit 0 – Bit 4 to I/O ports.
   By assigning 15 to input ports: for starting the programming control.
2. Choose the required programming functions for I/O ports.

Example:
Assign 25 (Bit 0) for IN1 port
Assign 26 (Bit 1) for IN3 port
Assignments for IN1~7 are available based on user's needs.

IN3 Bit1	IN1 Bit0	Step
0	0	1
0	1	2
1	0	3
1	1	4

Note: "1" in the graph above means "Activated signal"
The signal of programming assignments should be completed for at least 20msec before the Start signal.

Examples: Writing / Checking / Saving
Note: All message elements are hexadecimal.

Example 1. Set/Write a programming parameter

Command 1, current step#0: Set Constant moving speed as 1000, i.e. 10rps

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

① Address 0x1
② Writing MODBUS command "0x10"
③ Communication address 0x400 (=1024 in decimal)
④ 2 data
⑤ 4 bytes
⑥ Data 1: Specify "Constant moving speed" command (0x0036 = 54 in decimal)
⑦ Data 2: Specify value 0x03E8 for Constant moving speed (=1000 in decimal)
⑧ CRC check

Command 2, current step#1: Set a relative positioning with 10000 pulses

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

① Address 0x1
② Writing MODBUS command "0x10"
③ Communication address 0x402 (=1026 in decimal)
④ 3 data
⑤ 6 bytes
⑥ Data 1: Set Relative positioning command as 0x0002 (=2 in decimal)
⑦ Data 2: Specify value 0x2710 to move with pulses (=10000 in decimal)
⑧ CRC check

Command 3, current step#2: Waiting for 1000ms

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

① Address 0x1
② Writing MODBUS command "0x10"
③ Communication address 0x405 (=1029 in decimal)
④ 3 data
⑤ 6 bytes
⑥ Data 1: Specify value 0x0041 for Relative positioning command (=65 in decimal)
⑦ Data 2: Convert data from "03 E8 00 03" to "00 03 03 E8"
A B C

*Note: Little-Endian byte order is applied for 4-bytes data.

A: System default value is 0. Please do not change the setting.

B: Waiting for jumping to #3, current command step is #2

C: Specify a value 0x03E8 for Waiting time (=1000ms in decimal)

⑧ CRC check

Command 4, current step#3: Perform a Relative moving repeatedly for 10 times.

01	10	04 08	00 03	06	00 42	00 01 00 0A	DB 92
①	②	③	④	⑤	⑥	⑦	⑧

① Node address 0x1

② MODBUS command “0x10”

③ Communication Address 0x408 (=1032 in decimal)

④ 3 data

⑤ 6 bytes

⑥ Data 1: Specify a value 0x0042 for Relative positioning command (= 66 in decimal)

⑦ Data 2: Convert data from 00 01 00 0A to <u>00 0A</u> <u>00 01</u>

A B

*Note: Little-Endian byte order is applied for 4-bytes data.

A: Specify 10 times of Jumping (0xA = 10 in decimal)

B: Jump to #1 and perform a relative positioning again

⑧ CRC check

Command 5, current step #4: End of the motion sequence

01	06	04 0B	00 64	F8 D3
①	②	③	④	⑤

① Node address 0x1

② MODBUS command “0x06”

③ Communication Address 0x40B (= 1035 in decimal)

④ End of the data (0x64 = 100 in decimal)

⑤ CRC check

Example 2. Check programming parameter

01	06	01 43	00 0E	F8 26
①	②	③	④	⑤

① Node address 0x1

② MODBUS command “0x06”

③ Communication Address 0x0143 (= 323 in decimal, communication command setting)

④ Data: Check programming data (0xE =14 in decimal)

⑤ CRC check

3. Save a programming parameter

*Note: Please check the data before saving it. If not, the data may not be saved properly.

01 06 01 43 00 OF 39 E6

① ② ③ ④ ⑤

① Node address 0x1

② MODBUS command "0x06"

③ Communication address 0x0143 (=323 in decimal, communication command setting)

④ Data: Save programming data (0xF =15 in decimal)

⑤ CRC check

Protocol description

The MODBUS protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers. The mapping of MODBUS protocol on specific buses or network can introduce some additional fields on the application data unit (ADU).

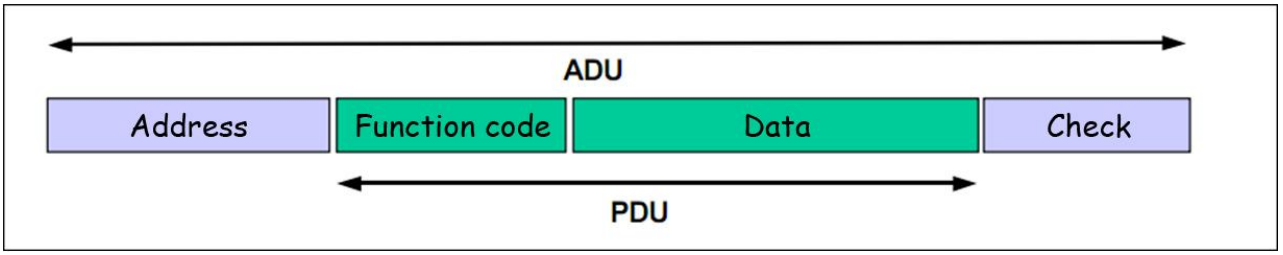


Figure 1. General MODBUS frame

The size of the MODBUS PDU is limited by the size constraint inherited from the first MODBUS implementation on Serial Line network (max. RS485 ADU = 256 bytes).

Therefore:

MODBUS PDU for serial line communication = 256 - Server address (1 byte) - CRC (2 bytes) = 253 bytes.

Consequently:

RS232 / RS485 ADU = 253 bytes + Server address (1 byte) + CRC (2 bytes) = 256 bytes.

The MODBUS protocol defines three PDUs. They are:

- MODBUS Request PDU, mb_req_pdu
- MODBUS Response PDU, mb_rsp_pdu
- MODBUS Exception Response PDU, mb_excep_rsp_pdu

The mb_req_pdu is defined as:

mb_req_pdu = {function_code, request_data}, where
function_code = [1 byte] MODBUS function code,
request_data = [n bytes] This field is function code dependent and usually contains information such as variable references, variable counts, data offsets, sub-function

codes etc.

The mb rsp pdu is defined as:

mb rsp_pdu = (function code, response_data), where

function code = [1 byte] MODBUS function code

response_data = [n bytes] This field is function code dependent and usually contains information such as variable references, variable counts, data offsets, sub-function codes, etc.

The mb excep rsp_pdu is defined as:

mb excep rsp_pdu = (exception-function code, request data), where

exception-function code = [1 byte] MODBUS function code + 0x80

exception code = [1 byte] MODBUS Exception Code Defined in table ”MODBUS Exception Codes” (see Section 12 below)

10 MODBUS Transaction

10.1 Define MODBUS Transaction

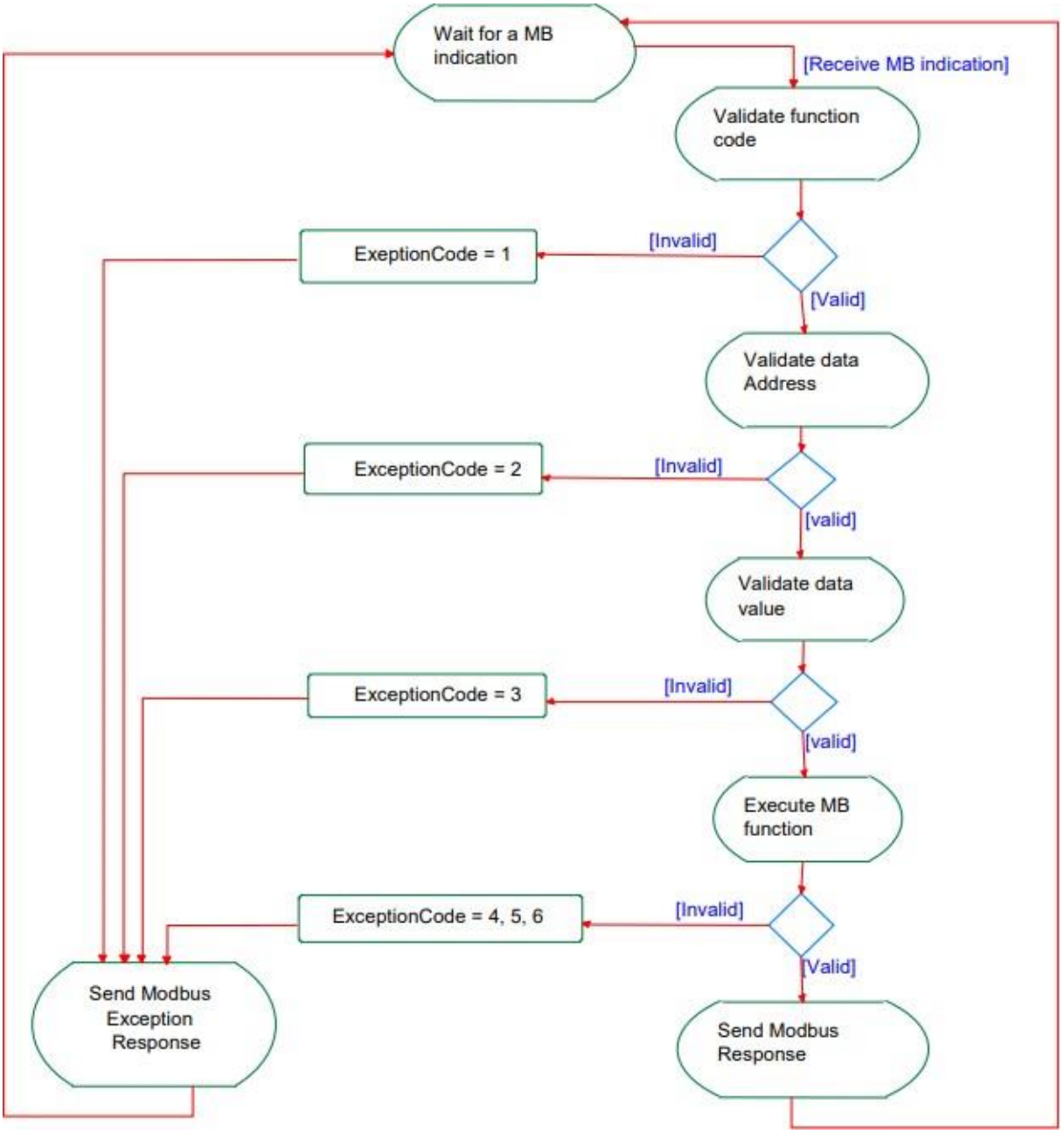


Figure 2. MODBUSTransaction state diagram

Once the request has been processed by a server, a MODBUS response using the adequate MODBUS server transaction is built. Depending on the result of the processing two types of response are built:

- A positive MODBUS response:
  The response function code = the request function code

- A MODBUS Exception response (see section 12):

The objective is to provide to the client relevant information concerning the error detected during the processing;
The exception function code = the request function code + 0x80;
An exception code is provided to indicate the reason of the error.

10.2 MODBUS: General response

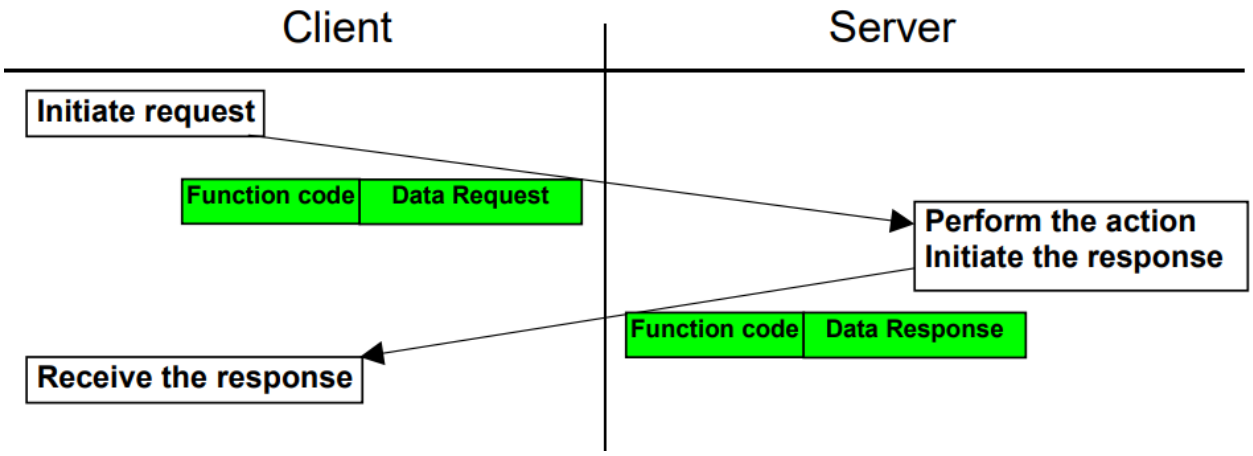


Figure 3. MODBUS transaction (error free)

10.3 MODBUS: General exception response

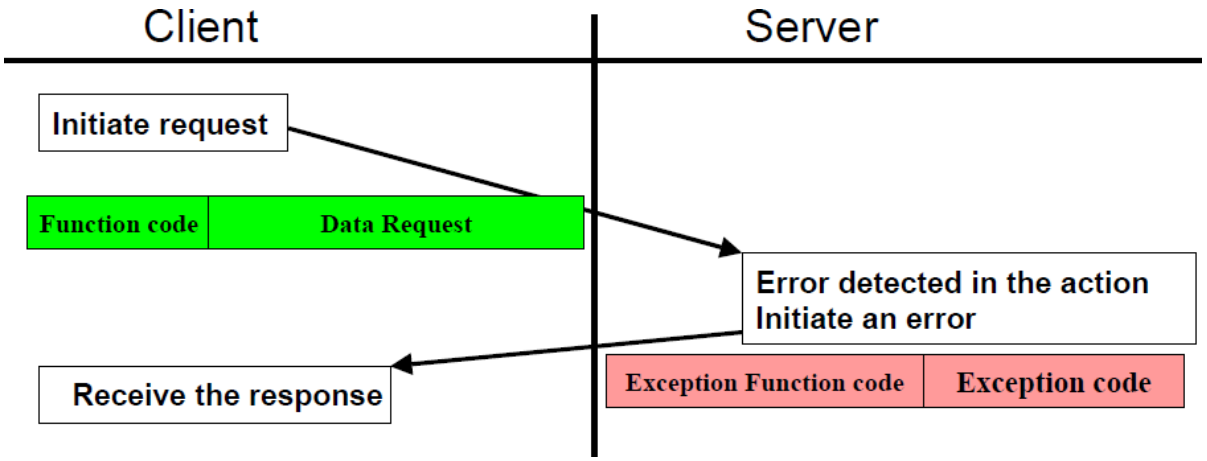


Figure 4. MODBUS transaction (exception response)

11 Data Encoding

MODBUS uses a ‘big-Endian’ representation for addresses and data items.

12 Public Function Code Definition

				Function codes		
				Code	Sub code	(hex)
Data Access	Bit access	Physical discrete inputs	Read discrete inputs	02		02
		Internal bits or Physical coils	Read coils	01		01
			Write single coil	05		05
			Write multiple coils	15		0F
	16 bits access	Physical input registers	Read input register	04		04
		Internal registers or Physical output registers	Read holding registers	03		03
			Write single register	06		06
			Write multiple registers	16		10
			Read/Write multiple registers	23		17
			Mask write register	22		16
	File record access		Read file record	20	6	14
			Write file record	21	6	15
Diagnostics			Read device identification	43	14	2B

The frequently used function codes based on communication needs are highlighted in above table.

03 (0x03): Read holding registers
06 (0x06): Write single register
16 (0x10): Write multiple registers

12. 1 03 (0x03): Read holding registers

This function code is used to read the contents of a contiguous block of holding registers in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU Registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first bye contains the high order bits and the second contains the low order bits.

Request

Function code	1 byte	0x03
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	1 to 125 (0x7 D)

Response

Function code	1 byte	0x03
Byte count	1 byte	2xN*
Register value	N*x 2 bytes	

*N = Quantity of registers

Error

Error code	1 byte	0x03
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to read registers 108 — 110:

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	03	Function	03
Starting Address Hi	00	Byte Count	06
Starting Address Lo	6B	Register value Hi (108)	02
No. of Registers Hi	00	Register value Lo (108)	2B
No. of Registers Lo	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 shown as the two byte values of 02 2B hex, or 555 decimal. The contents of registers 109-110 are 00 00 and 00 64 hex, or 0 and 100 decimal, respectively.

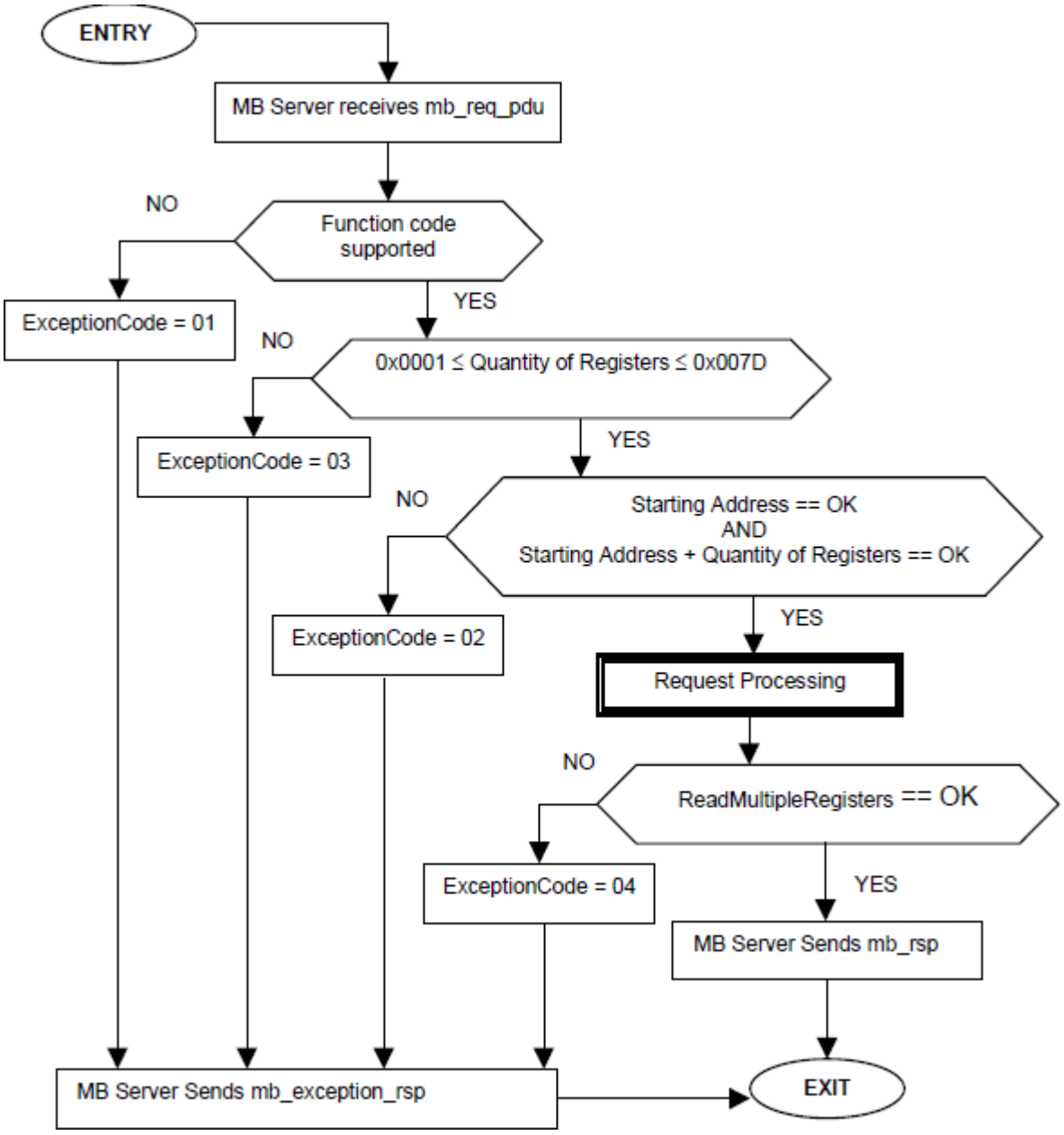


Figure 5. Read holding registers state diagram

12.2 06 (0x06): Write single register

This function code is used to write a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0. The normal response is an echo of the request, returned after the register contents have been 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

Error

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

Here is an example of a request to write register 2 to 00 03 hex:

Request		Response	
Field	(Hex)	Field	(Hex)
Function	06	Function	06
Register address Hi	00	Register address Hi	00
Register address Lo	01	Register address Lo	01
Register value Hi	00	Register value Hi	00
Register value Lo	03	Register value Lo	03

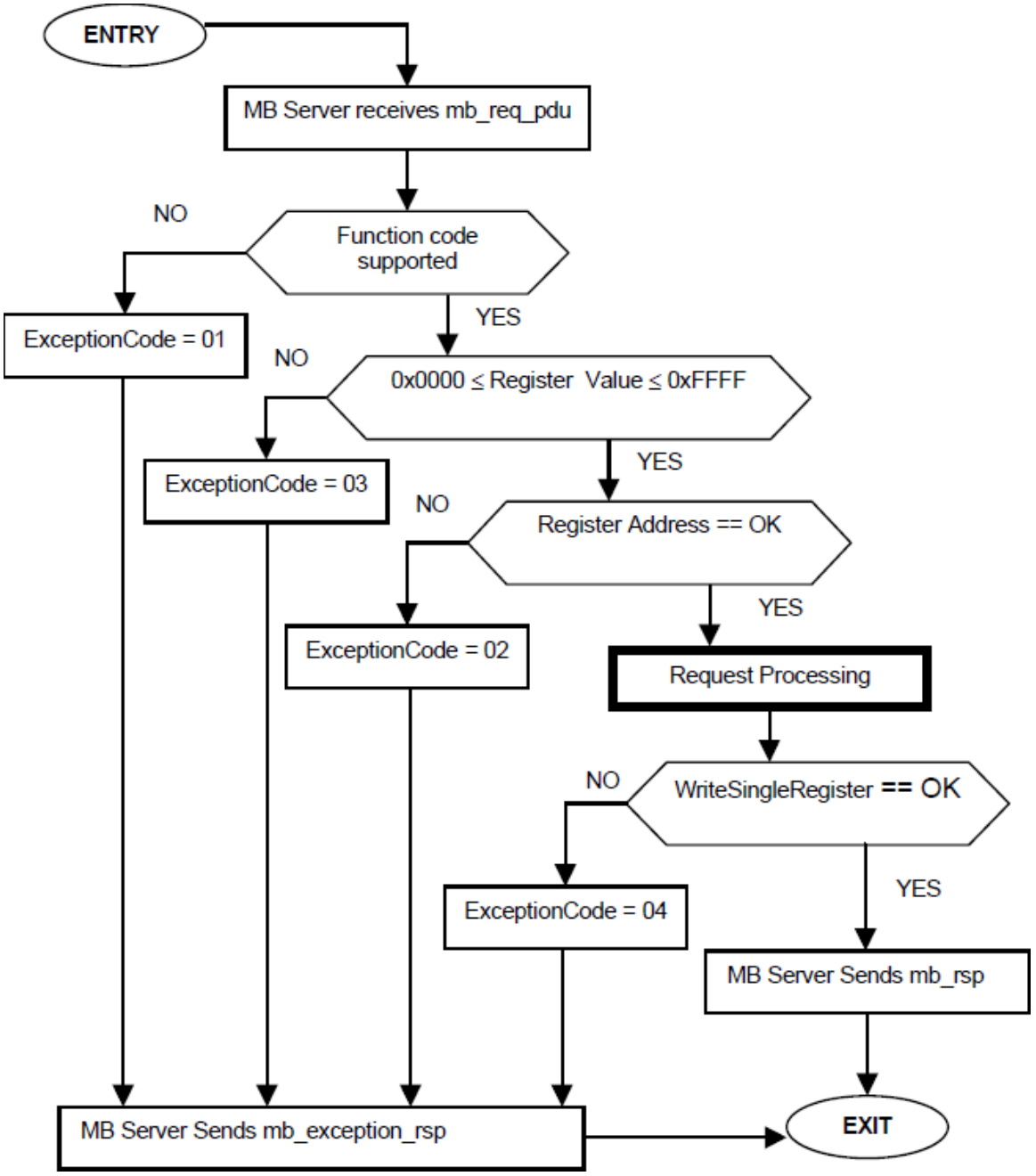


Figure 6. Write single register state diagram

12.3 16 (0xl0): Write multiple registers

This function code is used to write a block of contiguous registers (1 to 120 registers) in a remote device. The requested written values are specified in the request data field. Data is packed as two bytes per register. The normal response returns the function code, starting address, and quantity of registers written.

Request

Function code	1 byte	0x10
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	0x0001 to 0x0078
Byte count	1 byte	2 x N*
Registers value	N*x 2 bytes	value

*N = Quantity of registers

Response

Function code	1 byte	0x10
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	1 to 123 (0x07B)

Error

Error code	1 byte	0x10
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to write two registers starting at 2 to 00 0A and 01 02 hex:

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	10	Function	10
Starting address Hi	00	Starting address Hi	00
Starting address Lo	01	Starting address Lo	01
Quantity of registers Hi	00	Quantity of registers Hi	00
Quantity of registers Lo	02	Quantity of registers Lo	02
Byte count	04		
Registers value Hi	00		
Registers value Lo	0A		
Registers value Hi	01		
Registers value Lo	02		

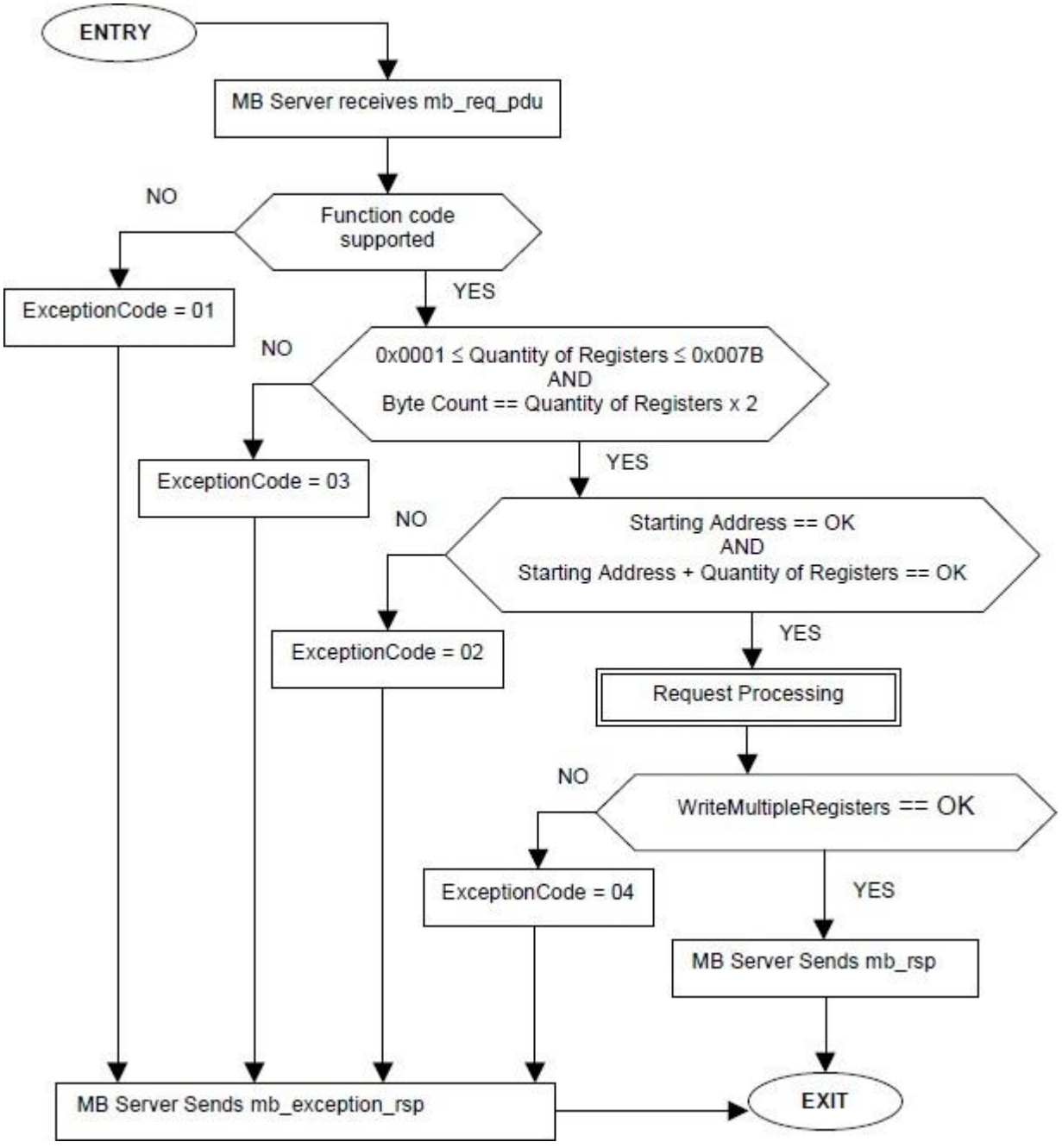


Figure 7. Write multiple registers state diagram

A listing of exception codes begins on the next page.

MODBUS Exception codes		
Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, if a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 “Illegal Data Address”.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the driver was attempting to perform the requested action.
05	ACKNOW-LEDGE	Specialized use in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a Poll Program Complete message to determine if processing is completed.
06	SLAVE DEVICE BUSY	Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long—duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server (or slave) attempted to read record file, but detected a parity error in the memory. The client (or master) can retry the request, but service may be required on the server (or slave) device.

0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways indicates that no response was obtained from the target device. Usually means that the device is not present on the network.
0C	TRANSMISSION TIME-OUT BETWEEN BYTES	If the interval between the last byte and the next byte in the same frame sent is greater than 1.5 character times, an error will occur and the current frame will be deleted. An error code is sent if no data is received within 3.5 character times. For continuous data transmission, an error code will be sent when there's no data is received within 3.5 character times. In this case, the previously received data will be discarded.
0D	THE INTER-FRAME DELAY IS LESS THAN 3.5T	An error occurs once another message frame was transmitted within 3.5 character times after a successful reception of the previous message frame An error code returns when there was no data received after a 3.5 character times expired. For continuous data transmission, an error code will be sent when there's no data is received within 3.5 character times. In this case, the previously received data will be discarded.

13 MODBUS master node protocol principle

The master node issues a MODBUS request to the slave nodes in two modes:

13. 1 **In unicast** mode, the master addresses an individual slave. After receiving and processing the request, the slave returns a message (a 'reply') to the master.

In that mode, a MODBUS transaction consists of 2 messages: a request from the master, and a reply from the slave. Each slave must have a unique address (from 1 to 247) so that it can be addressed independently from other nodes

13.2 In broadcast mode, the master can send a request to all slaves.

No response is returned to broadcast requests sent by the master. The broadcast requests are necessarily writing commands. All devices must accept the broadcast for writing function. The address 0 is reserved to identify a broadcast exchange.

14 MODBUS addressing rules

The Modbus addressing space comprises 256 different addresses.

0	1-47	48-255
Broadcast address	Slave individual addresses	Reserved

The address 0 is reserved as the broadcast address. All slave nodes must recognize the broadcast address. The Modbus master node has no specific address, only the slave nodes must have an address. This address must be unique on a Modbus serial bus.

Note: Communication address = Value of DIP switches +1 (The address of the driver cannot be 0)

15 Master / Slave communication time diagram

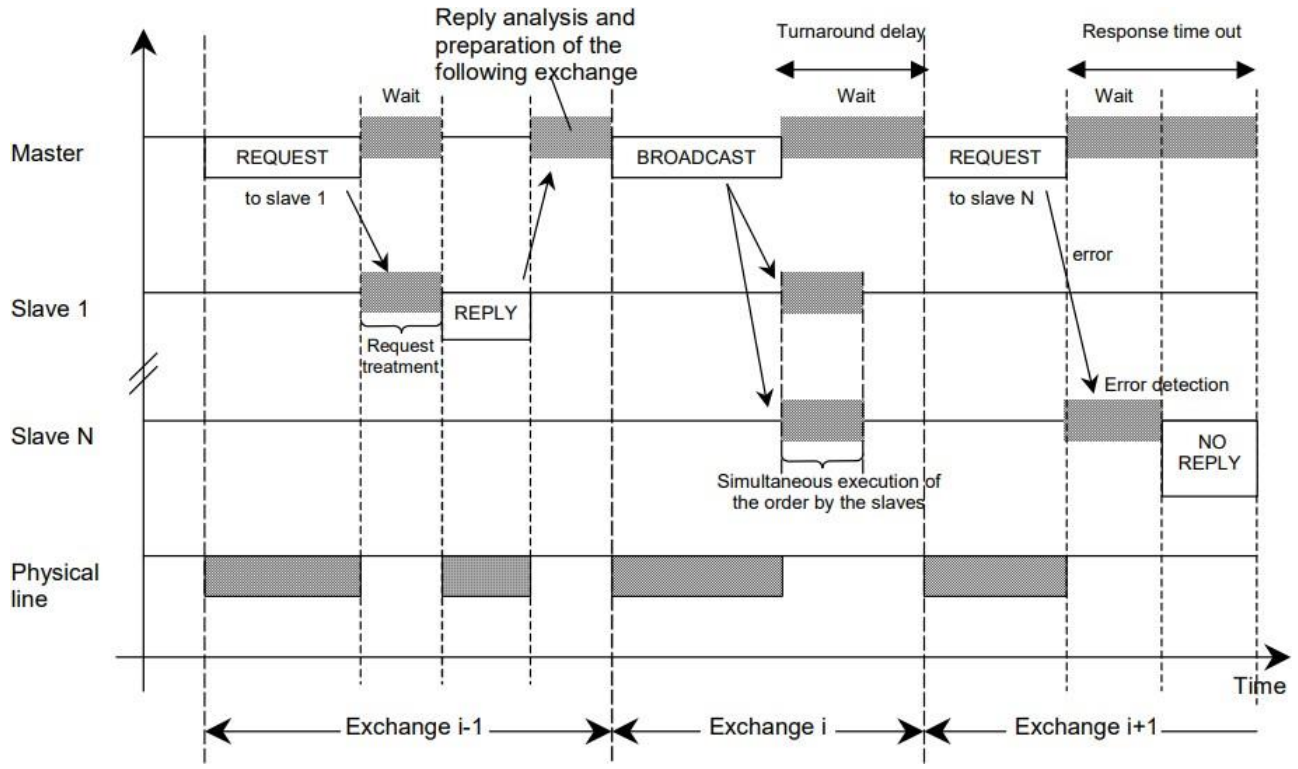


Figure 8. Master / Slave scenario time diagram

16 RTU transmission mode

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

Coding System: 8-bit binary. Each 8-bit byte in a message contains two 4–bit hexadecimal characters. (0–9， A–F)

Bits per byte: 1 start bit
8 data bits, least significant bit sent first
1 bit for parity completion
1 stop bit

Modbus message RTU framing

A MODBUS message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message, and to know when the message is completed. Partial messages must be detected and errors must be set as a result. In RTU mode, message frames are separated by a silent interval of at least 3.5 character times. In the following sections, this time interval is called t3,5.

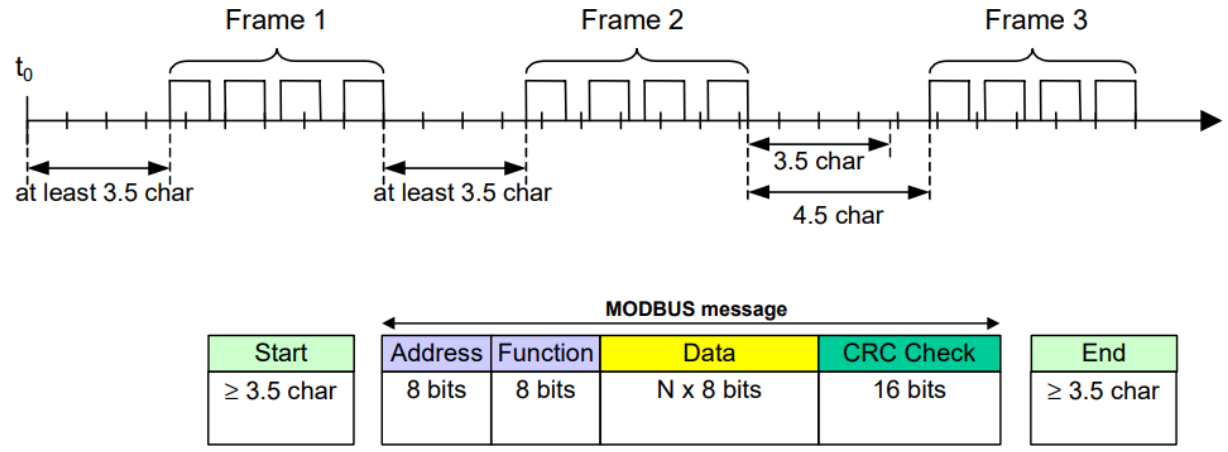


Figure 9. RTU message frame

The entire message frame must be transmitted as a continuous stream of characters. If a silent interval of more than 1.5 character times occurs between two characters, the message frame is declared incomplete and should be discarded by the receiver.

Baud rate	Interval of message	Spaces between bytes	Response time-out	Transition 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 checking

The CRC field contains a 16-bit value implemented as two 8-bit bytes. The CRC field is appended to the message as the last field in the message. When this is done, the low-order byte of the field is appended first, followed by the high-order byte. The CRC high-order byte is the last byte to be sent in the message.

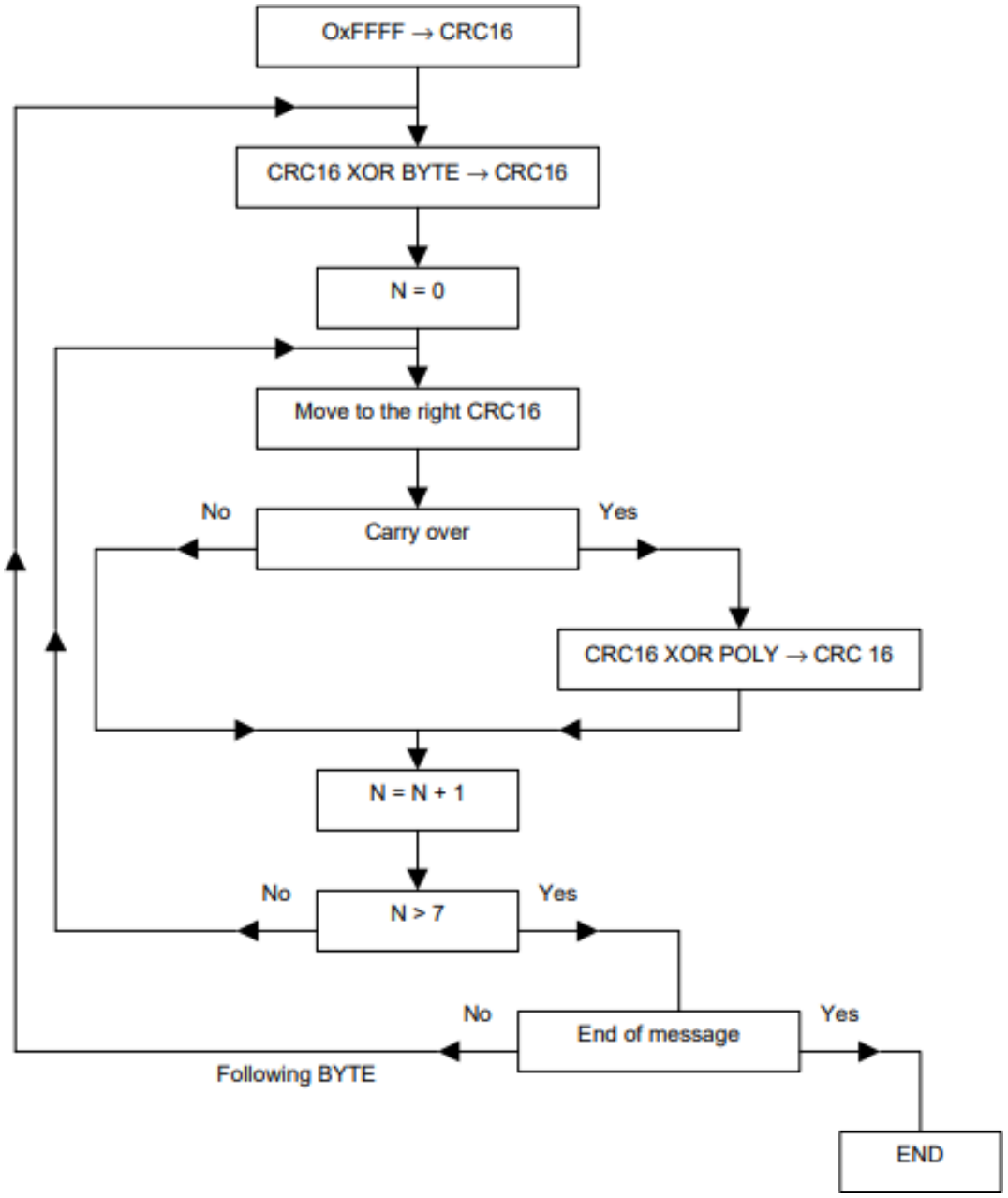


Figure 10. Calculation algorithm of the CRC 16

XOR = Exclusive or
N = Number of information bits
POLY = Calculation polynomial of the CRC 16 = 1010 0000 0000 0001
(Generating polynomial = $1 + x_2 + x_{15} + x_{16}$)
In the CRC 16, the 1st byte transmitted is the least significant one.

The function takes two arguments:
unsigned char *puchMsg; A pointer to the message buffer containing binary data to be used for generating the CRC,
unsigned short usDataLen; The quantity of bytes in the message buffer.

CRC Generation Function

```
unsigned short CRC16 (puchMsg, usDataLen)        /* will be returned to CRC with unsigned short */

unsigned char *puchMsg ;                         /* message to calculate CRC upon */
unsigned short usDataLen ;                       /* quantity of bytes in message */


    unsigned char uchCRCHi = 0xFF ;              /* high byte of CRC initialized "/
    unsigned char uchCHRLo = 0xFF ;              /* low byte of CRC initialized */
    unsigned uIndex ;                            /* will index into CRC lookup table */

    while (usDataLen--)                          /* pass through message buffer */

        uIndex = uchCRCLo ^ *puchMsgg+ + ;       /* calculate CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex);
        uchCRCHi = auchCRCLo[uIndex] ;

    return (uchCRCHi << 8 | uchCRCLo) ;
```

High-Order Byte Table

/* Table of CRC values for 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, 0 x00, 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-Order Byte Table

/* Table of CRC values for 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, 0x1 D, 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, 0x8 D, 0x4D, 0x4C, 0x8C, 0x44, 0x84, 0x85, 0x45,
```

0x87, 0x47, 0x46, 0x86 , 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40
};

18 Two-Wire MODBUS Definition

A MODBUS solution over serial line should implement a "Two-Wire" electrical interface in accordance with EIA/TIA-485 standard.

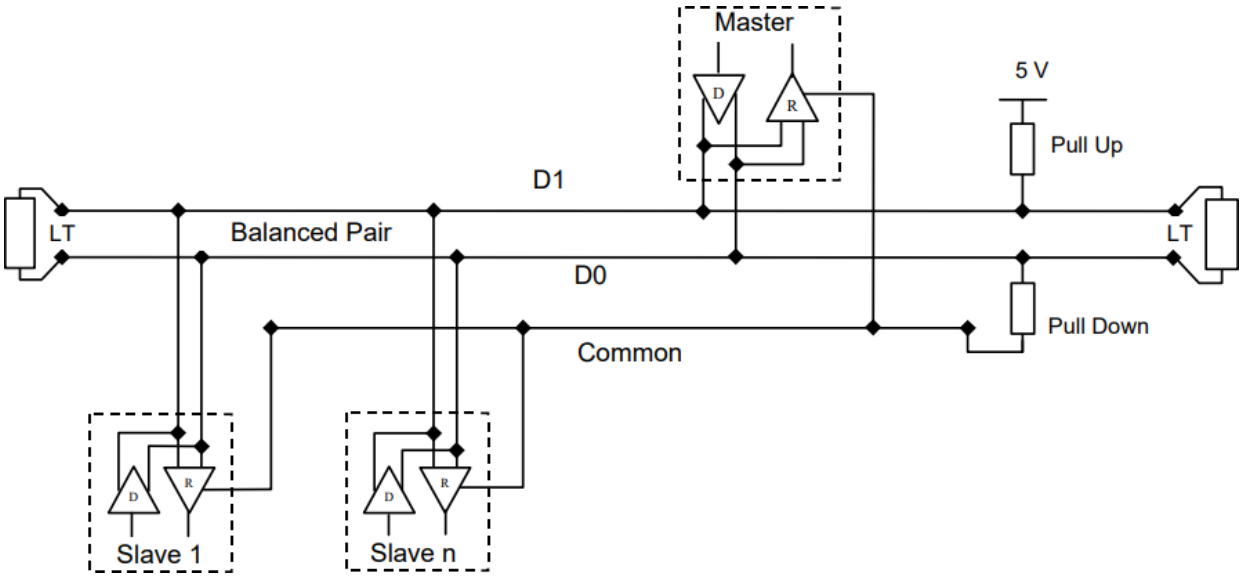
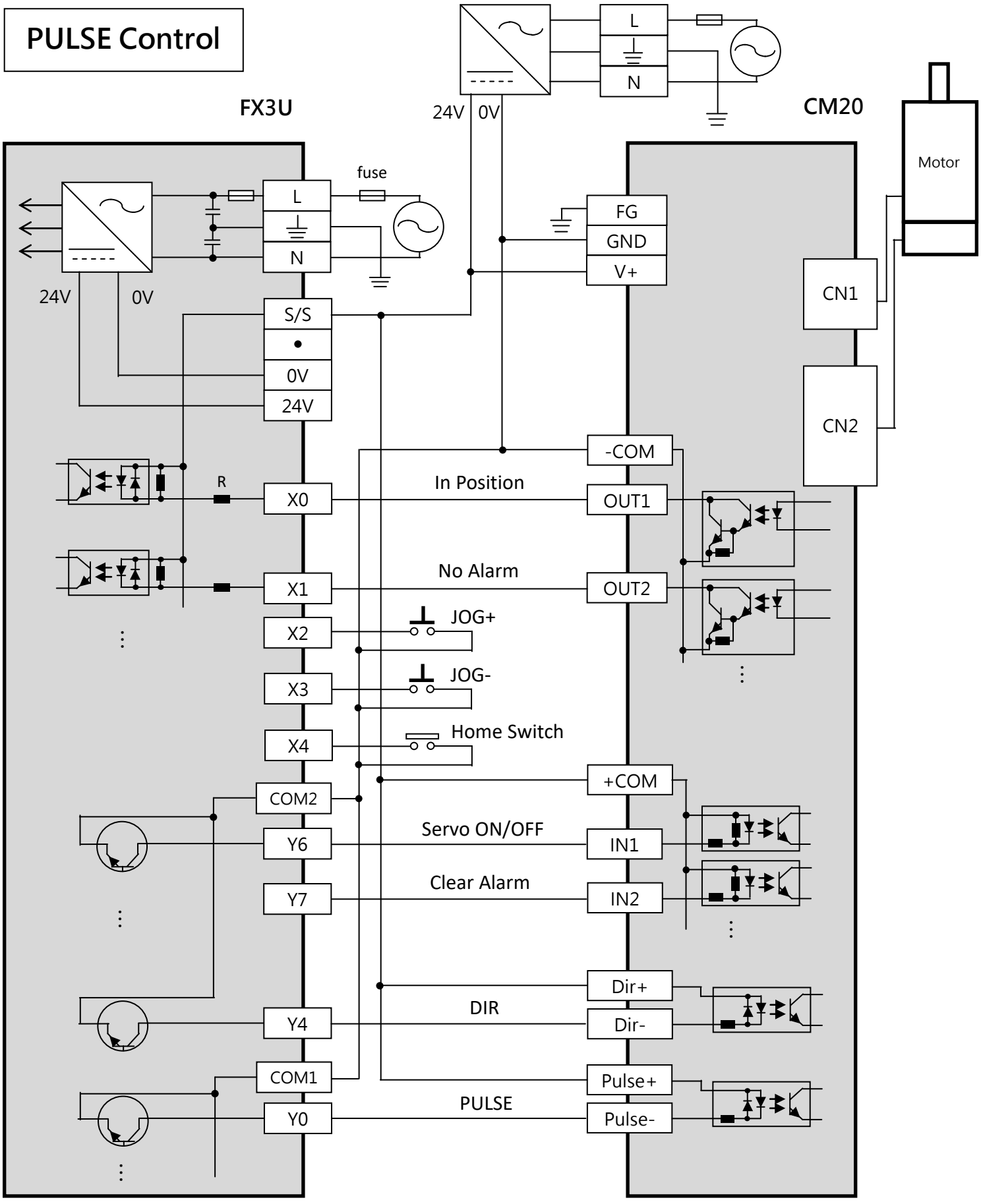
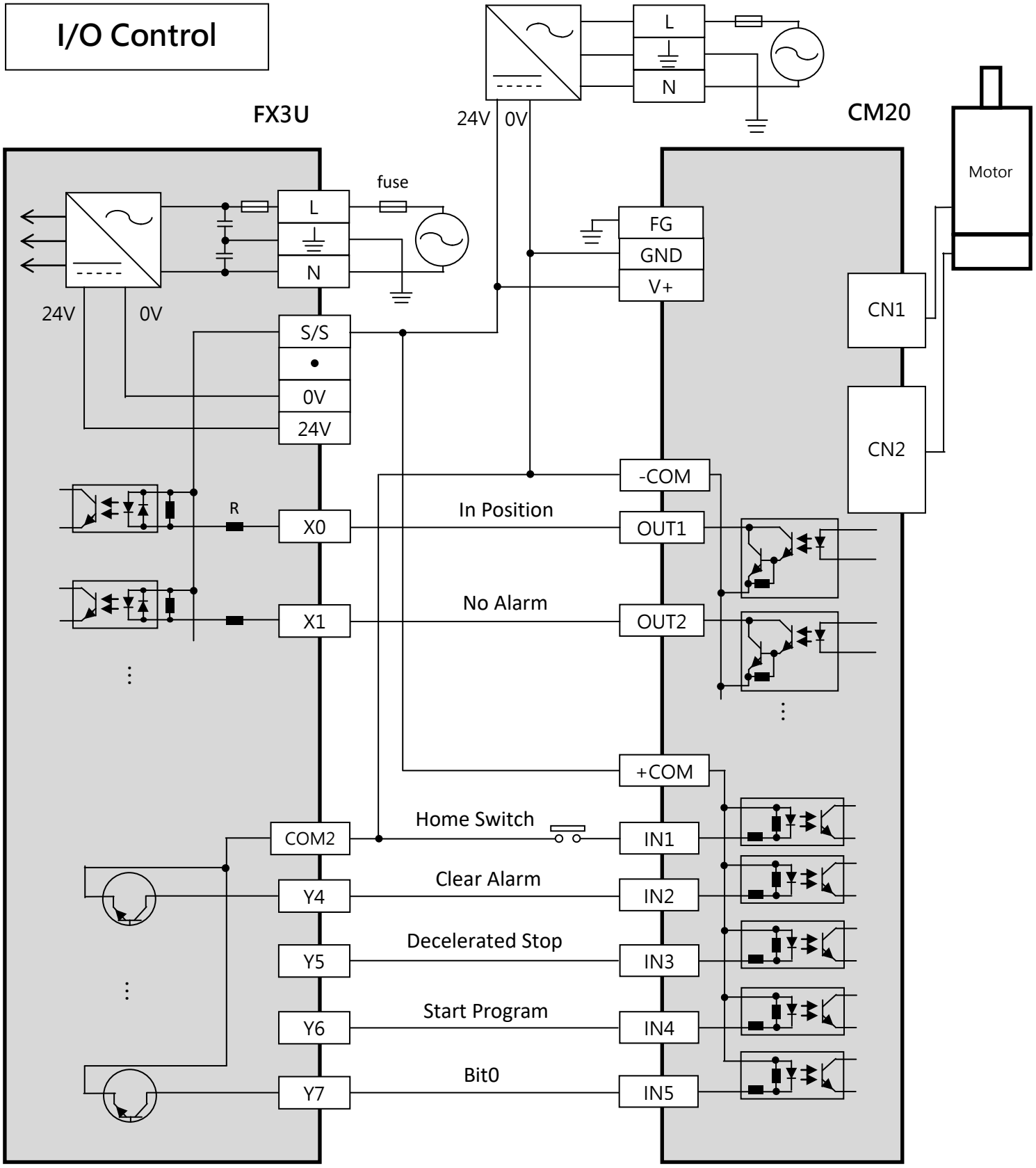


Figure 11. General 2-Wire Topology

Appendix. Wiring Diagram Example of MITSUBISHI FX3U Series



I/O Control


FX3U
CM20
L
N
24V
0V
fuse
S/S
0V
24V
FG
GND
V+
Motor
CN1
CN2
-COM
+COM
R
X0
X1
In Position
No Alarm
OUT1
OUT2
COM2
Y4
Y5
Y6
Y7
Home Switch
Clear Alarm
Decelerated Stop
Start Program
Bit0
IN1
IN2
IN3
IN4
IN5