ZHAOWEI

Shenzhen ZHAOWEI Machinery & Electronics Co., Ltd.

ZHAOWEI Industrial Park, No. 88, Yanhu Road, Yanchuan Community, Yanluo Street, Baoan District, Shenzhen City, Guangdong Province, China

· Shenzhen · Dongguan · Suzhou · Shanghai · Hong Kong · Germany · USA

Email: sales@szzhaowei.net
Tel: +86-0755-2732-2645
Web.: www.zwgearbox.com

ZHAOWEI | Stock Code 003021

ZWHAND
Dexterous Hand

ZWHAND-DM17 User Manual

Contents

ZWHAND Dexterous Hand User Manual ZHAOWEI

1 Important Notice

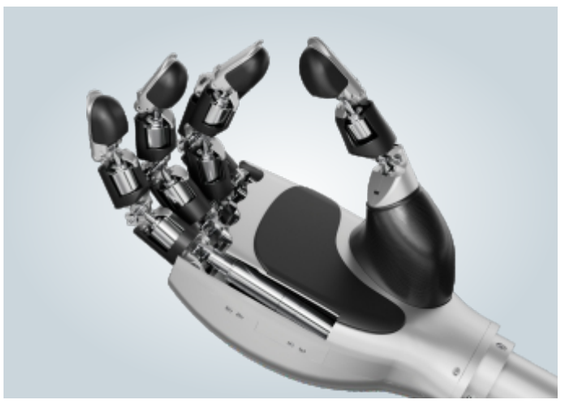
> Caution
>
> The ZWHAND bionic dexterous hand is not waterproof.

The interior of the ZWHAND dexterous hand is equipped with precision electronic components, power motors, and complex mechanical structures. Once these components are invaded by liquids, it will cause serious consequences such as short circuits and rusting of parts, which in turn will lead to irreversible damage to the equipment. Therefore, users must strictly ensure that no liquid penetrates into the interior of the dexterous hand during operation, and should avoid operating the device in environments with excessively high humidity or heavy dust pollution to prevent potential damage.

The ZWHAND dexterous hand uses high-strength aluminum alloy and high-quality stainless steel as the main skeleton materials to ensure structural stability and durability. However, users still need to pay special attention during use to ensure that the applied load does not exceed its design specifications (12N for the index finger, 5N for the other fingers). Any load exceeding the design bearing range may cause permanent deformation or even fracture of metal parts, which in turn will cause irrecoverable damage to the internal structure. In addition, accidents such as falling or being hit by heavy objects will also pose a serious threat to the mechanical structure and circuit system of the ZWHAND dexterous hand, possibly leading to functional failure.

2 Product introduction

The ZWHAND dexterous hand is equipped with 17 precision motion joints, each incorporating a high-performance motor, thus achieving 17 independent active degrees of freedom. With the built-in intelligent motor control algorithm, the dexterous hand can simulate various complex grasping movements of the human hand. Its typical application fields are extensive, including but not limited to robot end effectors, educational and scientific research equipment, and factory automation applications.



3 Product model, composition and dimensions

3.1 Model number

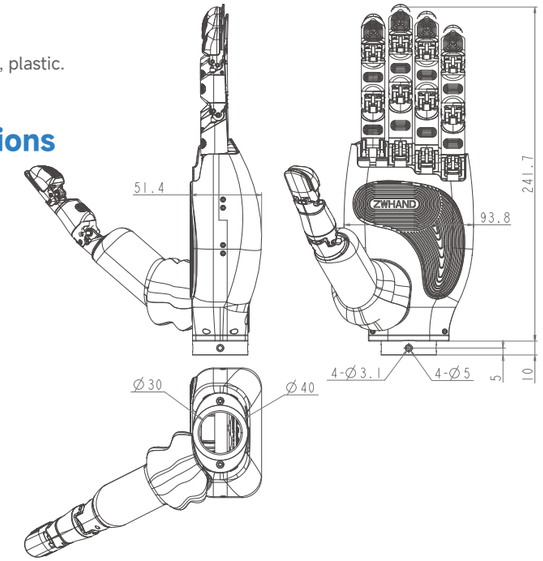
ZWHAND-DM17

17 active degrees of freedom, skin optional

3.2 Material

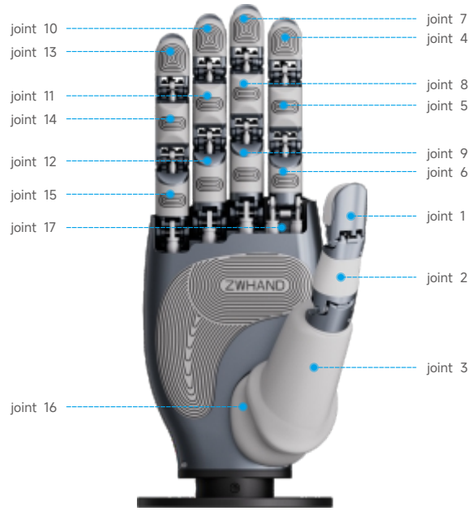
Aluminum alloy, stainless steel, silica gel, plastic.

3.3 Structural dimensions

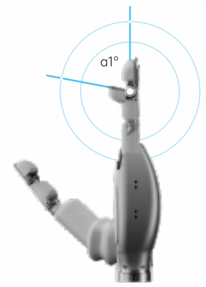
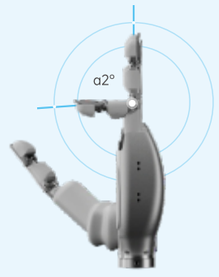


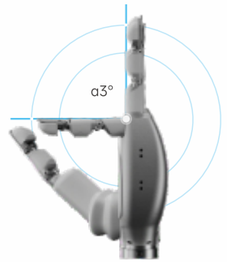
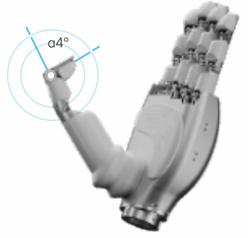
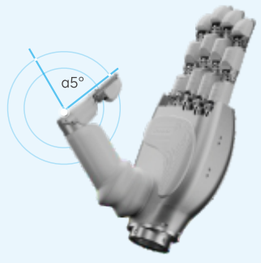
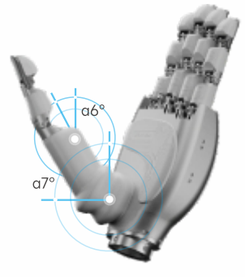
3.4 Motion space

The joint motion ranges of the dexterous hand are detailed as follows:



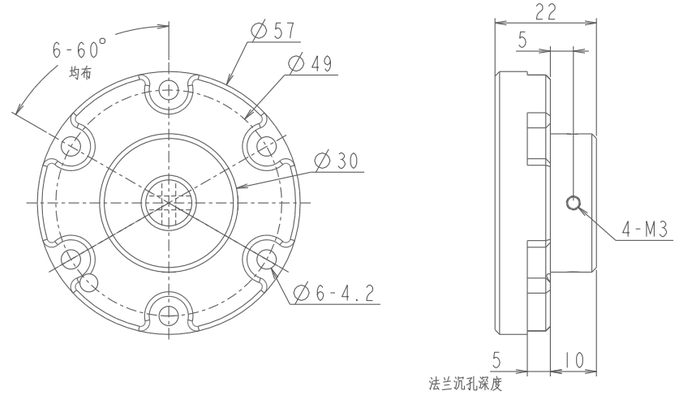
Category		Working Range (Angle)	
Thumb	joint 16	60	30-90
	joint 3	45	45-90
	joint 2	70	10-80
	joint 1	80	0-80
Index finger	joint 17	10	0-10
	joint 6	90	0-90
	joint 5	90	0-90
	joint 4	80	0-80
Middle finger	joint 9	90	0-90
	joint 8	90	0-90
	joint 7	80	0-80
Ring finger	joint 12	90	0-90
	joint 11	90	0-90
	joint 10	80	0-80
Llittle finger	joint 15	90	0-90
	joint 14	90	0-90
	joint 13	80	0-80

Joint	Diagram Reference	Range of Motion (ROM)
Four fingers – Distal Interphalangeal Joints		α1: 0-80°
Four fingers – Proximal Interphalangeal Joints		α2: 0-90°

Four fingers – Metacarpophalangeal Joints		α3: 0-90°
Index finger – MCP lateral abduction/adduction		β1: 0-10°
Thumb-Distal Interphalangeal Joint		α4: 0-80°
Thumb – Metacarpophalangeal Joint		α5: 10-80°
Thumb – Carpometacarpal (CMC) Joint		α6: 30-90° α7: 45-90°

3.5 Wrist connection dimensions

The ZWHAND is equipped with a standard mechanical interface as shown in the technical diagram. Customized interfaces can be provided to meet specific integration requirements upon request.



4 Performance Parameters

4.1 Load and Speed

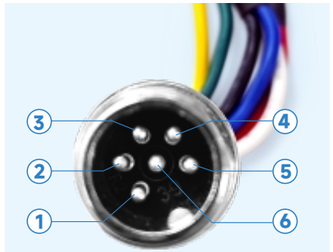
Parameter	Value
Minimum Time: Full Finger Extension to Full Flexion	0.45s
Minimum Time: Full Finger Flexion to Full Extension	0.45s
Minimum Time: Full Thumb Closure	0.38s
Minimum Time: Full Thumb Opening	0.38s
Maximum lifting weight (power grip)	3kg

4.2 Weigh

Parameter	Value
Total Weight with Wrist-Mounted Flange	1000 g

4.3 Communication Interface

The ZWHAND wrist integrates a high-performance motion control board featuring RS485/CANFD interfaces. This board offers a convenient 6-pin M12 male aviation connector option, which is securely mounted on the back of the wrist module. When facing the front of the male connector, the six signal pins are arranged clockwise as follows:



Pin Number	Pin Definition
1	485+/CAN_H
2	485-/CAN_L
3	GND
4	VCC
5	VCC
6	GND

4.4 Other Parameters

Parameter	Value
Operating Voltage	DC12V
Operating Current	5A
Standby Current	≤300mA
Operating Temperature	-10°C~40°C
Total Joints	17
Active DOF	17
Whole-Hand Grip Force	30N
Fingertip Pressure	5~12N

5 Communication Control Protocol

The communication protocol used by the ZWAHDN is a custom proprietary protocol operating over a CAN FD (Controller Area Network with Flexible Data-Rate) bus. Leveraging the multi-master peer-to-peer communication capability of CAN FD, the protocol utilizes predefined CAN identifiers (IDs) and custom data frame formats to enable command and data exchange between devices. This facilitates operations such as status reading and control of single or multiple finger joints.

5.1 Frame ID Forma

ID[10]	ID[9]	ID[8]	ID[7]	ID[6]	ID[5]	ID[4]	ID[3]	ID[2]	ID[1]	ID[0]
Message Direction				Device ID						

Message Direction:

0x00 Indicates a message from the control unit to the dexterous hand.

Any other value indicates a message from the dexterous hand to the control unit.

Device ID:

Default: 0x01,

Configurable range : 0x02 to 0xFE

5.2 Data Read/Write Operations

Read operations are used by the host to request specific status or parameter values from the dexterous hand. The host sends a read command frame over CAN FD using a designated read command CAN ID, which includes a target read instruction and data identifier defined in the data field format. Upon receiving and successfully verifying the frame, the dexterous hand will extract the requested data based on the identifier, package the relevant status or parameter information into another response frame (using the predefined response CAN ID), and transmit it back to the host system via the CAN FD bus.

Write/control operations refer to commands sent from the host (upper computer) to instruct the dexterous hand to perform actions or set parameters. The host sends a frame over the CAN FD bus containing the target control instruction and control parameters (e.g., joint angles, speed, etc., defined in the data field format), using the predefined write/control command CAN ID. After receiving and validating the frame, the dexterous hand will parse the instruction and the data segment contents, execute the corresponding action or apply the parameter settings, and send a response frame (using the predefined response or status feedback CAN ID) back to the host system via the CAN FD bus to indicate the execution status or result.

5.2.1 Data Write / Control Operations

Data write/control operations are executed using control instruction 0x10. The data frame format is as follows:

Data Frame Format: Instruction (0x10) + Function Code + Length + Data		
Request Frame Format	Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit) + Data (16-bit MSB)	
Byte	Parameter	Explanation
Byte0	Operation instructions	Instruction (Specifies the read or write function)
Byte1	function code	Function code (the specific operation function)
Byte2	length	Number of operations

Byte3 Byte4	data1	Data 1
Byte5 Byte6	Data2	Data 2
. . .	. . .	. . .
Byte(2 + 2*(n-1)+1) Byte(2 + 2*(n-1)+2)	Data N	Data N
Response Frame Format	Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit) + Data (16-bit MSB)	
Byte0	Operation instructions	Instruction (Specifies the read or write function)
Byte1	function code	Function code (the specific operation function)
Byte2	length	Number of operations
Byte3 Byte4	data1	Data 1
Byte5 Byte6	Data2	Data 2
. . .	. . .	. . .
Byte(2 + 2*(n-1)+1) Byte(2 + 2*(n-1)+2)	Data N	Data N

5.2.2 Data Read

Data read operations are executed using control instruction 0x04/0x06. The data frame format is as follows:

Data Frame Format: Instruction (0x06/0x04) + Function Code + Length + Data		
Request Frame Format	Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit)	
Byte	**Parameter**	**Explanation**
Byte0	Operation instructions	Instruction (Specifies the read or write function)
Byte2	function code	Function code (the specific operation function)
Byte3	length	Number of operations
Byte4 Byte5	data1	Data 1
Response Frame Format	Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit) + Data (16-bit MSB)	
Byte0	Operation instructions	Instruction (Specifies the read or write function)
Byte1	function code	Function code (the specific operation function)

Byte2	length	Number of operations
Byte3 Byte4	data1	Data 1
Byte5 Byte6	Data2	Data 2
Byte7 Byte8	Data3	. . .
. . .	. . .	Data N
Byte(2 + 2*(n-1)+1) Byte(2 + 2*(n-1)+2)	Data N	Instruction (Specifies the read or write function)

5.3 Dexterous Hand Functionality

5.3.1 Function Definitions

Functional logical address	Abbreviation	Read/Write Permission	Address width	Explanation
0x0000	HAND_ID	R/W	16 bit	Dexterous Hand ID
0x0001	BUS_BAUD_RATE	R/W	16 bit	Bus Baud Rate
0x0002	CLEAR_ERROR	R/W	16 bit	Clear Error
0x0003	POWER_OFF_SAVE_DTAT	R/W	16 bit	Save data on power-off
0x0005~0x0015	MOTOR_SPEED	R/W	16 bit	Config maximum operating speed for motors 1–17
0x0016~0x0026	MOTOR_CURRENT	R/W	16 bit	Config maximum operating current for motors 1–17
0x0027~0x0037	MOTOR_STOP	R/W	16 bit	Stop operation of motors 1–17
0x0038~0x0048	MOTOP_ABS_POS	R/W	16 bit	Set absolute positions for motors 1–17
0x0049~0x0059	MOTOP_INC_POS	R/W	16 bit	Set incremental positions for motors 1–17
0x005A~0x006A	MOTOR_ZERO	R/W	16 bit	Calibrate zero position for individual motors (1–17)
0x006B	ALL_STEP_MOTOR_ZERO	R/W	16 bit	Calibrate zero position for all stepper motors
0x006C	ALL_MOTOR_ZERO	R/W	16 bit	Calibrate zero position for all motors in the entire hand

5.3.1.1 HAND_ID Configure Dexterous Hand Identifier (Slave Device ID)

Description: This dexterous hand product supports online adjustment of the dexterous hand identifier (slave address). The currently supported identifier range is 0x01~0xFE. To adjust the identifier, simply modify the value of the HAND_ID register. Upon successful configuration, the hand will send a confirmation frame and modify the device ID before the start of the next frame cycle.

The new device ID is only valid for the current power cycle. To retain the updated ID, refer to 5.3.1.4 POWER_OFF_SAVE_DATA for saving configuration.

Identifier Configuration Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0x00	Function Code	uint8_t	0x00
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte7	0xXX	High Byte of HAND_ID	uint16_t	0x01~0xFE
Byte8	0xXX	Low Byte of HAND_ID		

Identifier Configuration Respond Format: Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit) + Data (16-bit MSB)

Byte	Data	Explanation	Data type	Range
Byte0	0xXX	Control Instruction	uint8_t	0x10
Byte1	0x10	Function Code (Write Multiple Holding Registers)	uint8_t	0x00
Byte2	0xXX	Function Code Length	uint8_t	0x01
Byte3	0xXX	Low byte of the previous HAND_ID		
Byte4	0xXX	High byte of the previous HAND_ID	uint16_t	0x0001

Example: Set Identifier to 0x02

Request	10 00 01 00 02
Respond	10 00 01 00 01

5.3.1.2 RS485 BUS_BAUD_RATE Configuration

Description: The dexterous hand allows online adjustment of the bus baud rate. Five predefined baud rate levels are available:

Level	Arbitration Phase	Data Phase	Sample Point
1	500k	500k	80%
2	500k	1000k	80%
3	500k	2000k	80%
4	1000k	5000k	80%
5	1000k	1000k	75%

To update the baud rate, write the desired level into the BUS_BAUD_RATE register. After successful configuration, the dexterous hand will acknowledge with a confirmation frame and apply the new settings before the next communication cycle.

The new baud rate setting is valid only for the current power cycle. To persist the configuration, refer to 5.3.1.4 POWER_OFF_SAVE_DATA.

Baud Rate Configuration Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0x01	Function Code	uint8_t	0x01
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte7	0xXX	High Byte of the BUS_BAUD_RATE level	uint16_t	0x01~0x05
Byte8	0xXX	Low Byte of the BUS_BAUD_RATE level		

Baud Rate Configuration Response Format: Instruction (0x10) + Start Function Code (8-bit) + Function Count (8-bit) + Data (16-bit MSB)

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0x01	Function Code (Write Multiple Holding Registers)	uint8_t	0x01
Byte2	0xXX	Function Code Length	uint8_t	0x01
Byte3	0xXX	High byte of the previous BUS_BAUD_RATE level	uint16_t	0x01~0x05
Byte4	0xXX	Low byte of the previous BUS_BAUD_RATE level		

Example: Set Baud Rate to 0x02

Request	10 00 01 00 02
Respond	10 00 01 00 01

5.3.1.3 CLEAR_ERROR

Description: This dexterous hand product supports online clearance of existing faults. To clear faults, simply write 1 to the CLEAR_ERROR bit.

CLEAR_ERROR Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0x01	Function Code	uint8_t	0x02
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte7	0xXX	High Byte of the BUS_BAUD_RATE level	uint16_t	0x01
Byte8	0xXX	Low Byte of the BUS_BAUD_RATE level		

No response

5.3.1.4 POWER_OFF_SAVE_DTAT

Description: This dexterous hand product requires saving the current relevant parameters before powering off to enable faster startup next time. To save data before shutdown, simply write the command 1 to the POWER_OFF_SAVE_DATA bit.

Power-off Data Save Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0x01	Function Code	uint8_t	0x03
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte7	0xXX	High Byte of the BUS_BAUD_RATE level	uint16_t	0x01
Byte8	0xXX	Low Byte of the BUS_BAUD_RATE level		

No response

5.3.1.5 MOTOR_SPEED

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. It supports limiting the maximum operating speed of one or multiple motors simultaneously. During operation, the motor's running speed will not exceed the defined threshold. The adjustable range for each motor's speed is 0–100 (unitless). The logical addresses of the MOTOR_SPEED parameters for the 17 motors are from 0x0005 to 0x0015. To set a speed limit, modify the corresponding MOTOR_SPEED parameter.

Single/Multiple Motor Speed Setting Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x0005~0x0015
Byte2	0xXX	Function Code Length	uint8_t	0x00-0x11
Byte3	0xXX	High Byte of the MOTOR_SPEED	uint16_t	0x0000~0x0064
Byte4	0xXX	Low Byte of the MOTOR_SPEED		
· · · · · · ·	0xXX	High Byte of the MOTOR_SPEED		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the MOTOR_SPEED		
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the MOTOR_SPEED	uint16_t	0x0000~0x0064

Example: Limit All Motor Speeds to 50% of Maximum

Request	01 10 00 05 00 11 22 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 A4 F0

5.3.1.6 MOTOR_CURRENT

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. It supports limiting the maximum operating current of one or multiple motors simultaneously to ensure safe operation. The adjustable range of maximum current for each motor is 0–100 (unitless). The logical addresses of the MOTOR_CURRENT parameters for the 17 motors are from 0x0016 to 0x0026. To set a current limit, modify the corresponding MOTOR_CURRENT control parameter.

MOTOR_CURRENT Setting Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x16~0x26
Byte2	0xXX	Function Code Length	uint8_t	0x00-0x11
Byte3	0xXX	High Byte of the MOTOR_CURRENT	uint16_t	0x0000~0x0064
Byte4	0xXX	Low Byte of the MOTOR_CURRENT		
· · · · · ·	0xXX	High Byte of the MOTOR_CURRENT	uint16_t	0x0000~0x0064
· · · · · ·	0xXX	Low Byte of the MOTOR_CURRENT		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the MOTOR_CURRENT	uint16_t	0x0000~0x0064
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the MOTOR_CURRENT		

Example: Limit all motors to a maximum of 50% current.

Request	01 10 00 16 00 11 22 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 00 32 15 B3

5.3.1.7 MOTOR_STOP

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. It supports stop control for one or multiple motors simultaneously. The logical addresses of the MOTOR_STOP parameters for the 17 motors range from 0x0027 to 0x0037. To stop a running motor, simply write 1 to the corresponding MOTOR_STOP control parameter.

Request Format for Single or Multiple Motors Stop Control: Instruction (0x10) + Function Code + Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x0027~0x0037
Byte2	0xXX	Function Code Length	uint8_t	0x00-0x11
Byte3	0xXX	High Byte of the Motor Stop Command	uint16_t	0x0000~0xFFFF
Byte4	0xXX	Low Byte of the Motor Stop Command		

· · · · · · ·	0xXX	High Byte of the Motor Stop Command	uint16_t	0x0000~0xFFFF
· · · · · · ·	0xXX	Low Byte of the Motor Stop Command		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the Motor Stop Command	uint16_t	0x0000~0xFFFF
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the Motor Stop Command		

Example: Control All Motors to Shut Down Simultaneously

Request	01 10 00 27 00 11 22 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 59 64

5.3.1.8 MOTOR_ABS_POS

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. It supports absolute position control for one or multiple motors simultaneously. After calibration, each motor's motion range is defined as 0–1000 (unitless). To control the motor to a specific absolute position within this range, simply modify the value of the MOTOR_ABS_POS parameter corresponding to the target motor.

Request Format for Single or Multiple Motors absolute position control: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x0038~0x0048
Byte2	0xXX	Function Code Length	uint8_t	0x00-0x11
Byte3	0xXX	High Byte of the MOTOR_ABS_POS	uint16_t	0x0001~0x03E8
Byte4	0xXX	Low Byte of the MOTOR_ABS_POS		
· · · · · · ·	0xXX	High Byte of the MOTOR_ABS_POS	uint16_t	0x0001~0x03E8
· · · · · · ·	0xXX	Low Byte of the MOTOR_ABS_POS		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the MOTOR_ABS_POS	uint16_t	0x0001~0x03E8
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the MOTOR_ABS_POS		

Example: Control the absolute position of motor No.15 to position 487.

Request	10 46 01 01 F4

5.3.1.9 MOTOR_INC_POS

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. It supports incremental position control for one or multiple motors simultaneously. (Incremental control means adjusting the position relative to the current position.) After calibration, each motor's motion range is defined as 0–1000 (unitless). To increment or decrement the motor position relative to its current position, modify the MOTOR_INC_POS parameter accordingly. A positive value increases the position, while a negative value decreases it.

Request Format for Single or Multiple Motors incremental position control: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x0049~0x0059
Byte2	0xXX	Function Code Length	uint8_t	0x00-0x11
Byte3	0xXX	High Byte of the MOTOR_INC_POS	uint16_t	0xFC18~0x03E8
Byte4	0xXX	Low Byte of the MOTOR_INC_POS		
· · · · · · ·	0xXX	High Byte of the MOTOR_INC_POS		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the MOTOR_INC_POS	uint16_t	0xFC18~0x03E8
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the MOTOR_INC_POS		

Example: Control motor No.15 to move positively by 500.

Request	10 57 01 01 F4

5.3.1.10 MOTOR_ZERO

Description: This dexterous hand product features 17 degrees of freedom, each corresponding to one motor. Initial use requires zero-position calibration for each motor. The logical addresses for motor zero calibration parameters (MOTOR_ZERO) for the 17 motors range from 0x005A to 0x006A. To calibrate one or more motors, simply write 1 to the corresponding MOTOR_ZERO parameter(s).

Request Format for Single or Multiple Motors zero-position calibration: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0xXX	Function Code	uint8_t	0x005A~0x006A
Byte2	0xXX	Function Code Length	uint8_t	0x0001~0x0011
Byte3	0xXX	High Byte of the MOTOR_ZERO command	uint16_t	0x0000~0xFFFF
Byte4	0xXX	Low Byte of the MOTOR_ZERO command		
· · · · · · ·	0xXX	High Byte of the MOTOR_ZERO command	uint16_t	0x0000~0xFFFF
· · · · · · ·	0xXX	Low Byte of the MOTOR_ZERO command		
Byte(2 + 2*(N-1)+1)	0xXX	High Byte of the MOTOR_ZERO command	uint16_t	0x0000~0xFFFF
Byte(2 + 2*(N-1)+2)	0xXX	Low Byte of the MOTOR_ZERO command		

Example: Calibrate the zero position of motor No.1

Request	10 5A 01 00 01

5.3.1.11 ALL_STEP_MOTOR_ZERO

Description: This dexterous hand product includes two types of motors: stepper motors and brushless motors. In addition to calibrating each motor individually, a simplified method is provided to calibrate all stepper motors at once. To do this, write 1 to the logical address of ALL_STEP_MOTOR_ZERO (0x006B).

ALL_STEP_MOTOR_ZERO Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0X6B	Function Code	uint8_t	0x6B
Byte2	0x00	Function Code Length	uint8_t	0x01
Byte4	0x00	High Byte of the ALL_STEP_MOTOR_ZERO command	uint16_t	0x0001
Byte5	0x01	Ligh Byte of the ALL_STEP_MOTOR_ZERO Command		

Example:Perform zero-position calibration for all stepper motors of the dexterous hand.

Request	10 6B 01 00 01

5.3.1.12 ALL_MOTOR_ZERO

Description: This dexterous hand product includes two types of motors: stepper motors and brushless motors. Besides individual motor zero-point calibration, a simplified method is available to calibrate all motors (both stepper and brushless) simultaneously. To perform full-hand calibration, simply write 1 to the logical address of ALL_MOTOR_ZERO (0x006C).

ALL_MOTOR_ZERO Request Format: Instruction (0x10) + Function Code +Length+ Data

Byte	Data	Explanation	Data type	Range
Byte0	0x10	Control Instruction	uint8_t	0x10
Byte1	0X6B	Function Code	uint8_t	0x6C
Byte2	0x00	Function Code Length	uint8_t	0x01
Byte4	0x00	High Byte of the ALL_MOTOR_ZERO command	uint16_t	0x0001
Byte5	0x01	Low Byte of the ALL_MOTOR_ZERO command		

Example: Perform zero-point calibration for all motors of the dexterous hand.

Request	10 6c 01 00 01

5.3.2 Dexterous Hand Input Register Definition

Functional logical address	Abbreviation	Read/Write Permission	Address width	Explanation
0x0000	INIT_OK	R	16-bit	Initialization success signal
0x0001	BOOT_VERSION	R	16-bit	BootLoader version
0x0002	HW_VERSION	R	16-bit	Hardware version
0x0003	VERSION	R	16-bit	Software version
0x0004	ERRCODE	R	16-bit	Error code
0x0005~0x000C	Reserve	R	16-bit	Reserved
0x000D	SYSVOLT	R	16-bit	System voltage value
0x0000E~0x001E	STALL_STATE	R	16-bit	Stall status
0x001F~0x002F	RT_Angle	R	16-bit	Real-time angle
0x0030-0x0040	Reserve	R	16-bit	Reserved
0x0041-0x0045	FTHUMBPRESS	R	16-bit	Fingertip pressure

5.3.2.1 INIT_OK

Description: The ZWHAND requires initialization after power-on to function properly.

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x00	Function Code	uint8_t	0x00
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x00	Function Code	uint8_t	0x00
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0x0001
Byte4	0xXX	Data Low Byte		

Example: Request Initialization Signal

Request	04 00 01
Response	04 00 01 00 01

5.3.2.2 BOOT_VERSION

Description: TThe dexterous hand product supports reading the bootloader version.

Request Format: Instruction (0x040) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x01	Function Code	uint8_t	0x01
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x01	Function Code	uint8_t	0x01
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		

Example: Request BOOTLOADER

Request	04 01 01
Response	04 01 01 00 01

5.3.2.3 HW_VERSION

Description: This smart handheld product supports reading hardware versions.

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x02	Function Code	uint8_t	0x02
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x02	Function Code	uint8_t	0x02
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		

Example: Request the hardware version

Request	04 02 01
Response	04 02 00 01

5.3.2.4 HW_VERSION

Description: This smart handheld product supports reading software versions.

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x03	Function Code	uint8_t	0x03
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x03	Function Code	uint8_t	0x03
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		

Example: Request the hardware version

Request	04 03 01
Response	04 03 01 00 01

5.3.2.5 Read System Error Status Code

Description: The ZWHAND supports reading error codes within the system

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x04	Function Code	uint8_t	0x04
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x04	Function Code	uint8_t	0x04
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		

Example: Request the hardware version

Request	04 04 01
Response	04 04 00 00

5.3.2.6 Read System Voltage

Description: The ZWHAND supports reading error codes within the system

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x0D	Function Code	uint8_t	0x0D
Byte2	0x01	Function Code Length	uint8_t	0x01

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0X04
Byte1	0x0D	Function Code	uint8_t	0x0D
Byte2	0x01	Function Code Length	uint8_t	0x01
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		

Example: With a 12V power supply, reading the system voltage returns 11.8V.

Request	04 0D 01
Response	04 0D 01 2E 6C

5.3.2.7 Read Motor Stall Status

Description: This dexterous hand supports reading the current motor stall status.

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x00E~0x1E
Byte2	0xXX	Function Code Length	uint8_t	0x01-11

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x00E~0x1E
Byte2	0xXX	Function Code Length	uint8_t	0x01-11
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0x0001
Byte4	0xXX	Data Low Byte		
...	...	...	...	...
Byte(2 + 2*(N-1)+1)	0xXX	Data N High Byte	uint16_t	0x0000-0x0001
Byte(2 + 2*(N-1)+2)	0xXX	Data N Low Byte		

Example: Read the stall status of 17 motors.

Request	04 0E 11
Response	04 0E 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

5.3.2.8 Read Real-Time Angle

Description: The ZWHAND supports reading the current motor Angle

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x1F~0x2F
Byte2	0xXX	Function Code Length	uint8_t	0x01-11

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x1F~0x2F
Byte2	0xXX	Function Code Length	uint8_t	0x01-11
Byte3	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte4	0xXX	Data Low Byte		
...	...	...	...	...
Byte(2 + 2*(N-1)+1)	0xXX	Data N High Byte	uint16_t	0x0000~0xFFF
Byte(2 + 2*(N-1)+2)	0xXX	Data N Low Byte		

Example: Read the Angle information of 17 motors

Request	04 30 11
Response	04 30 11 00 00 00 0D 00 00 00 00 00 00 00 C3 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 17 00 00 00 00

5.3.2.9 Read Fingertip Pressure

Description: The hand supports fingertip pressure readings. The corresponding register addresses are 0x0041 to 0x0045, representing fingertip pressure for the following fingers in order: Thumb, Index Finger, Middle Finger, Ring Finger, Little Finger.

Request Format: Instruction (0x04) + Function Code +Length

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x41-0x45
Byte2	0xXX	Function Code Length	uint8_t	0x01-0x05

Response Format: Instruction (0x04) + Function Code +Length+Data

Byte	Data	Explanation	Data type	Range
Byte0	0x04	Control Instruction	uint8_t	0x04
Byte1	0xXX	Function Code	uint8_t	0x41-0x45
Byte2	0xXX	Function Code Length	uint8_t	0x01-0x05
...	...	...	...	...
Byte(2 + 2*(N-1)+1)	0xXX	Data High Byte	uint16_t	0x0000-0xFFFF
Byte(2 + 2*(N-1)+2)	0xXX	Data Low Byte		

Example: Read the fingertip pressure information of 5 fingers.

Request	04 41 05
Response	04 41 05 00 00 00 00 00 00 00 00 00 11

6 Storage, Transportation and Operating Environment

Torage and Transportation Requirements: must be stored and transported in its original factory packaging	**Temperature:** -20°C ~ +60°C **Humidity:** Max Relative Humidity 85%
Operating Environment	**Temperature:** -10°C ~ +40°C **Humidity:** Max Relative Humidity 85%

ZWHAND Dexterous Hand User Manual ZHAOWEI