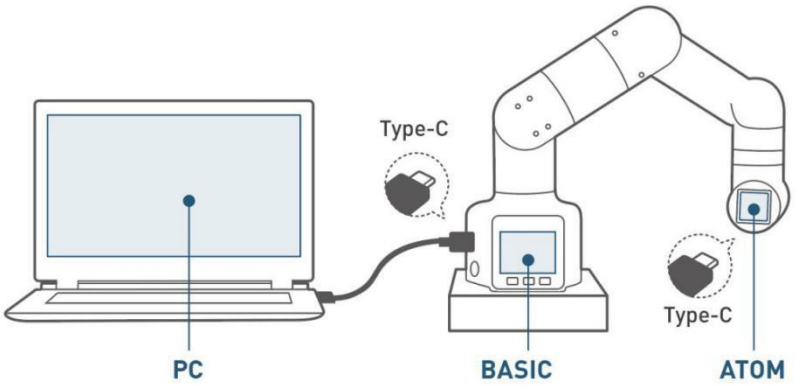
Development Manual

myCobot



Arduino 创客！ Maker!	Arduino IDE + M5Stack Lib 库 + MycobotBasic Lib 库	各类程序自定义 All Exapmles	atomMain
通信协议 – USB/TxRx0(G1/G3)	通信协议阅读 Read Protocol	Transponder文件	atomMain

Language:English
Version:V 2021.01.12

Copyright Declaration

No unit or individual may extract, compile, translate or reproduce any contents of this manual (eg: technical documentation,software,etc.), nor disseminate in any form (including materials and publications) without the written permission of Shenzhen Elephant Robotics Technology Co., Ltd. (hereinafter referred to as "Elephant Robotics").

In addition, the product information and related resources mentioned in this manual are for reference only and the contents are subject to change without notice.

Except as expressly stated in this manual, nothing in this manual should be construed as any warranty or guarantee by the Elephant Robotics of personal loss, damage to property, or fitness for a particular purpose.

All rights reserved!

Content

1. About Manual

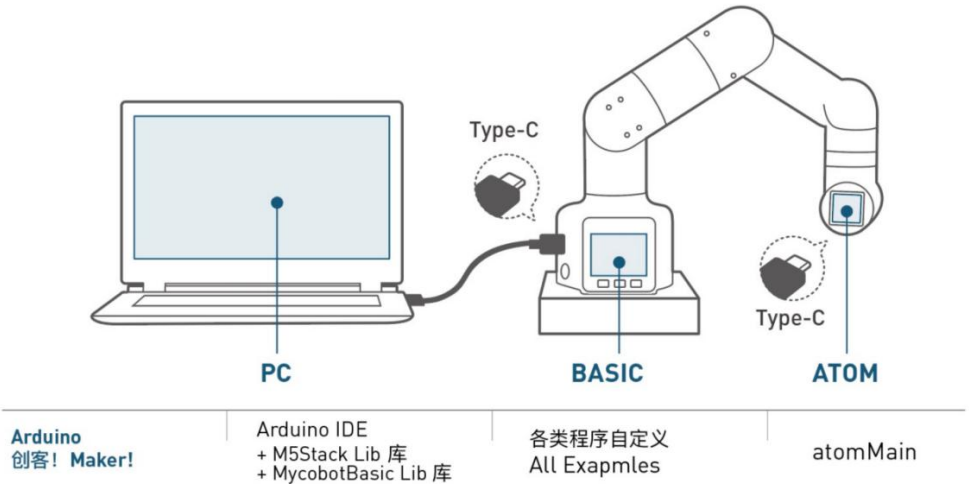
This manual is applicable to all series of MyCobot.

Library download, routine download, firmware download, etc.
Please visit the following links: https://github.com/elephantrobotics/myCobot

For product details and related products, please visit the following links:
https://www.elephantrobotics.com/en/myCobot-en/

2. Arduino API

Arduino API need to use and burn the following program



2.1 System Status (in development)

2.2 Overall Status

powerOn();

- Function: atom power on (open by default)
- Return Value: none

powerOff();

- Function: atom power off
- Return Value: none

isPoweredOn();

- Function: atom status inquiry, return Atom link status
- Return Value: power on is TRUE, power off is FALSE

setFreeMove();

- Function:All joints close torsion output
- Return Value: none

2.3 MDI Mode and Robot Control (Manual Data Input)

getAngles();

- Function: Read all joint angles, when used one Angles should be defined to receive data that was read. Angles are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 angles to store Angle variables, it is used in the same way as arrays.
- Return Value: Angles type of array

writeAngle(int joint, float value, int speed);

- Function: Send a single joint Angle
- Parameter Specification:
  Joint Number= joint, range from 1 to 6
  Specified Angle Value= value, range approximately from -170°~ + 170°
  Specified Speed= value, range from 0~100
- Return Value: none

writeAngles(Angles angles, int speed);

- Function: Synchronize joint angles, send joint angles at the same time. Specified Angles is a container with a capacity of 6 data, can be viewed as an array. Use a for loop to assign values, or assign values separately.
- Parameter Specification:
  Angles[0] = Specified Angle, Angles[2] = Specify Angle,range from 0 – 90 (the value range should be the same as writeAngle) unit°
  Movement Speed = speed, range from 0 – 100 unit°
- Return Value: none

getCoords();

- Function: Read x,y,z,rx,ry,rz of the end of myCobot, a Coords tempcoords should be defined when used to received angles that was read. Coords are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 tempcoords to store Angle variables, it is used in the same way as arrays.
- Return Value: An array of type COORDS. You need to define variables of type Coords.

writeCoord(Axis axis, float value, int speed);

- Function: Send the specific value of the individual coordinate parameters x/y/z, the ends are going to move in a single direction.
- Parameter Specification:
  Value of Moving Path Coordinate = value range from -300 – 300 (The position coordinates of axis=Axis::X,aixs=Axis::Y and axis=Axis::Z are respectively X,Y,Z, the units would be mm. Position coordinate value range is not uniform, axis=Axis::RX, aixs=Axis::RY and axis=Axis::RZ are respectively RX,RY,RZ ranging from-180°~180°, if the value is beyond the range it will return the clue "inverse kinematics no solution")
  Specified Speed =speed range from 0~100 unit %
- Return Value: none

writeCoords(Coords coords, int speed);

- Function: Send the specified coordinate parameter, the type of parameter is Coords, need to declares a variable of type COORDS, it is used in the same way as arrays.

- Parameter Specification:
  coords[0] = X, coords[1] = Y, coords[2] = Z,
  X,Y,Z range from -300-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm
  RX,RY,RZ range from -180~180
  Specified Speed = speed, range from 0~100unit %
- Return Value: none

checkRunning();

- Function: Check whether the equipment is in motion
- Return Value: In motion is TRUE， on the contrary it's FALSE

setEncoder(int joint, int encoder);

- Function: Set a single joint to rotate to a specified potential vaule
- Parameter Specification:
  Joint Number = joint, range from 1-6
  Potential Vaule of Servo Motor = encoder, range from 0-4096 (The range should be positively related to the range of each joint)
- Return Value: none

getEncoder(int joint);

- Function:Get the specified joint potential vaule
- Parameter Specification:
  Servo Motor Number = joint range from 1-6
- Return Value: int type， range from 0-4096

setEncoders(Angles angleEncoders, int speed);

- Function: Set the six joints to run synchronously to the specified position
  Parameter Specification:
  Need to define a variableof type Angles: angleEncoders, it is used in the same way as arrays. Assign a value to the array angleEncoders, values range from 0 to 4096 (The range should be positively related to the range of each joint)
- , the length range of the array is 6.
  Specified Speed = speed, range from 0~100unit %
- Return Value: none

2.4 JOG Mode

jogAngle(int joint, int direction, int speed);

- Function: Control the movement of a single joint in one direction
- Parameter Specification:
  Joint/Servo Motor Number = joint, range from 1-6
  Direction of Joint Motion = Direction, range from -1/1
  Specified Speed = speed, range from 0~100unit %
- Return Value: none

jogCoord(Axis axis, int direction, int speed);

- Function: Control myCobot moves in one direction in Cartesian space
- Parameter Specification:

Direction Selection = axis, range from X,Y,Z,RX,RY,RZ
Direction of Joint Motion = Direction, range from -1/1
Specified Speed = speed, range from 0~100unit %

- Return Value: none

jogStop();

- Function: Stops the specified direction of motion that has started
- Return Value: none

pause();

- Function: Program pause
- Return Value:none

resume();

- Function: Program continues to run
- Return Value: none

stop();

- Function: Program stop
- Return Value: none

2.5 Running Status and Settings

getSpeed();

- Function: read the current running speed
- Return Value: int tape, range from 0-100, unit %

setSpeed(int percentage);

- Function: set the running speed
- Parameter Specification: percentage, range from 0-100, unit %

getFeedOverride();

- Function: read FeedOverride
- Return Value: float type of the array

sendFeedOverride(float feedOverride);

- Function: send FeedOverride

getAcceleration();

- Function: read acceleration
- Return Value: floattype of the array

setAcceleration(float acceleration);

- Function: set acceleration
- Parameter Specification: acceleration range from 0-100 （Value range is not defined）

getJointMin(int joint);

- Function: read the joint minimal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6

- Return Value: float type of the array

getJointMax(int joint);

- Function: read the joint maximal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: float type of the array

setJointMin(int joint, float angle);

- Function: set the joint minimal limit Angle
- Parameter Specification: joint/servo motor number = joint, range from 1-6
- Return Value: none

setJointMax(int joint, float angle);

- Function: set the joint maximal limit Angle
- Parameter Specification: joint/servo motor number = joint, range from 1-6
- Return Value: none

2.6 Joint Servo Control

isServoEnabled(int joint);

- Function: Check whether the joint is properly connected
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Normal links return TRUE, on the contrary return FALSE

isAllServoEnabled();

- Function: Check whether all joins are properly connected
- Return Value: Normal links return TRUE, on the contrary return FALSE

getServoData(int joint, byte data_id);

- Function: Read the data of the specified address of joint
- Parameter Specification:
  Joint Number = joint, range from 1-6
  Data Address = data_id. , refer to the following Figure for address
  Return Value: refer to the following Figure for the value range

Address	Function	Value Range	Initial Value	Analytical Value
20	Led alarm conditions	0-254	0	The corresponding bit set to 1 is on, flashing alarm; The corresponding bit set to 0 is off, flashing alarm;
21	Position loop P proportionality coefficient	0-254	123 joint value is 8 456 joint value is 5	Control the proportional coefficient of the servo
22	Position loop D differential coefficient	0-254	123 joint value is 20 456 joint value is 13	Control the differential coefficient of the servo
23	Position loop I integral coefficient	0-254	0	Control the integral coefficient of the servo
24	Minimal start torque	0-1000	0	Set the minimal output start torque of the servo , set 1000 = 100% * stall torque

setServoData(byte servo_no, byte servo_state, byte servo_data);

- Function: Set the data of the specified address of servo
- Parameter Specification:
  Servo Number= servo_no, range from 1-6
  Servo State = servo_state, refer to the above Figure for address
  Servo Data = servo_data, refer to the above Figure for the value range
- Return Value: none

setServoCalibration(int joint);

- Function: Calibrate the current position of the joint to zero Angle, the corresponding potential value is 2048
- Parameter Specification: joint number = joint, range from 1 – 6

setPinMode(byte pin_no, byte pin_mode);

- Function: set atom specified pin mode
- Parameter Specification:
  Pin Number = pin_no, range from:19, 22, 23, 26, 32, 33
  Pin mode = pin_mode, range from:0, 1
- Return Value: none

2.7 Atom IO Control

setLEDRGB(byte r, byte g, byte b);

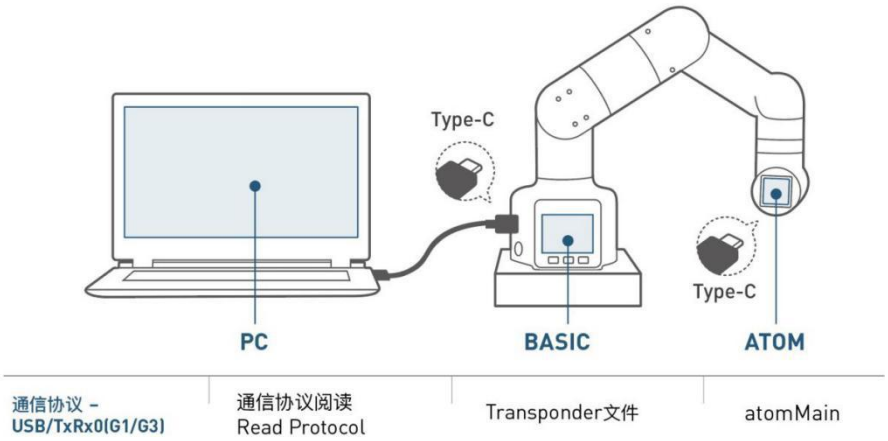
- Function:set the color of the RGB lights of atom
- Parameter Specification:
  Parameter of Red Light = r, range from 0x00 – 0xFF;
  Parameter of Green Light = g, range from 0x00 – 0xFF;
  Parameter of Blue Light = b, range from 0x00 – 0xFF;
- Return Value: none

setGripper(int data);

- Function: Set the jaw opening and closing
- Parameter Specification: data 0 is open, 1 is close

3. Communication protocols and data structures

Note: When use a communication protocol to communicate directly you need to burn Transponder in Basic and the latest version of AtomMain in Atom.



3.1 USB Communication Settings

- Bus Interface: USB Type-C
- Baud Rate: 115200
- Date Bits: 8
- Parity Check:none
- Stop Bit: 1

3.2 Command Frame Description & Single Instruction Parsing

The main BASIC sends data to the Atom, and the Atom parses the data after receiving it. For example, an instruction containing Return Value will be returned to the BASIC within 500ms.

Type	Data Description	Data Length	Description
Command frame	First byte 0	1	The frame head identification, 0xfe
	First byte1	1	The frame head identification, 0xfe
	Data length byte	1	Different instructions correspond to different lengths of data
	Command byte	1	Depends on different commands
Data frame	Data	0-16	Command attached data, depends on different commands
End frame	End the byte	1	Stop bit, 0xfa

3.3 Single Instruction Parsing

1). atom power on

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x10
Data[4]	End frame	0xfa

2). atom power off

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x11
Data[4]	End frame	0xfa

3). atom status inquiry

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x12
Data[4]	End frame	0xfe

4).read angle (read position information)

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x20
Data[4]	End frame	0xfa

5).returns a data

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Servo number	Servo_no
Data[3]	Angle in a low position	Angle_low
Data[4]	Angle in a high position	Angle_high
Data[5]	End frame	0xfa

6). send individual angles

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x06
Data[3]	Instruction frame	0x21
Data[4]	Joint number	Joint_no
Data[5]	Angle of rotation	Angle
Data[6]	Specified speed	Sp
Data[7]	End frame	0xfa

7). send all angles

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x0f
Data[3]	Instruction frame	0x22
Data[4]	Angle 1 is in low _byte	Angle1_low
Data[5]	Angle 1 is in high_byte	Angle1_high
Data[6]	Angle 2 is in low_byte	Angle2_low
Data[7]	Angle 2 is in high_byte	Angle2_high
Data[8]	Angle 3 is in low_byte	Angle3_low
Data[9]	Angle 3 is in high_byte	Angle3_high
Data[10]	Angle 4 is in low_byte	Angle4_low
Data[11]	Angle 4 is in high_byte	Angle4_high
Data[12]	Angle 5 is in low_byte	Angle5_low
Data[13]	Angle 5 is in high_byte	Angle5_high
Data[14]	Angle 6 is in low_byte	Angle6_low
Data[15]	Angle 6 is in high_byte	Angle6_high
Data[16]	Specified speed	Sp
Data[17]	End frame	0xfa

8). read coordinate

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x23
Data[6]	End frame	0xfa

9).send individual coordinates

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x06
Data[3]	Instruction frame	0x24
Data[4]	Specified coordinate	X/y/z/rx/ry/rz
Data[5]	Specified xyz/rxryrz Is in a low parameters	Xyz/ rxryrz_low
Data[6]	Specified xyz/rxryrz Is in a high parameters	Xyz/rxryrz_high
Data[7]	Specified speed	Sp
Data[8]	End frame	0xfa

10).send all coordinates

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x10
Data[3]	Instruction frame	0x25
Data[4]	Specified x is in a low coordinate	X_low
Data[5]	Specified x is in a high coordinate	X_high
Data[6]	Specified y is in a low coordinate	Y_low
Data[7]	Specified y in a high coordinate	Y_high
Data[8]	Specified z is in a low coordinate	Z_low
Data[9]	Specified z in a high coordinate	Z_high
Data[10]	Specified rx is in a low coordinate	Rx_low
Data[11]	Specified rx is in a high coordinate	Rx_high
Data[12]	Specified ry is in a low coordinate	Ry_low
Data[13]	Specified ry in a high coordinate	Ry_high
Data[14]	Specified rz is in a low coordinate	Rz_low
Data[15]	Specified rz is in a high coordinate	Rz_high
Data[16]	Specified speed	Sp
Data[17]	Mode	Mode
Data[18]	End frame	0xfa

11).whether to reach the point

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe

Data[2]	Data length frame	0x15
Data[3]	Instruction frame	0x2a
Data[4]	Joint1 is in a low position	Joint1_low
Data[5]	Joint1 is in a high position	Joint1_high
Data[6]	Joint2 is in a low position	Joint2_low
Data[7]	Joint2 is in a high position	Joint2_high
Data[8]	Joint3 is in a low position	Joint3_low
Data[9]	Joint3 is in a high position	Joint3_high
Data[10]	Joint4 is in a low position	Joint4_low
Data[11]	Joint4 is in a high position	Joint4_high
Data[12]	Joint5 is in a low position	Joint5_low
Data[13]	Joint5 is in a high position	Joint5_high
Data[14]	Joint6 is in a low position	Joint6_low
Data[15]	Joint6 is in a high position	Joint6_high
Data[16]	Coordinate	Coords
Data[17]	End frame	0xfa

12).check for movement

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x2b
Data[4]	End frame	0xfa

13).joint control

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x05
Data[3]	Instruction frame	0x30
Data[4]	Joint number	Joint
Data[5]	Joint direction	Direction
Data[6]	Specified speed	Sp
Data[7]	End frame	0xfa

14).coordinate control

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x05
Data[3]	Instruction frame	0x33
Data[4]	Specified coordinates	X1 y2 z3 rx4 ry5 rz6
Data[5]	Joint direction	Direction
Data[6]	Specified speed	Sp
Data[7]	End frame	0xfa

15).jog stop

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe

Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x34
Data[4]	End frame	0xfa

16).send potential value

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x05
Data[3]	Instruction frame	0x3a
Data[4]	Joint number	Joint
Data[5]	Encoder is in a Low position	Encoder_low
Data[6]	Encoder is in a High position	Encoder_high
Data[7]	End frame	0xfa

17).get potential value

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x3b
Data[4]	Joint angle	Joint angle
Data[5]	End frame	0xfa

18).send potential value of 6 joints

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x15
Data[3]	Instruction frame	0x3c
Data[4]	Angle 1 is in low _byte	Angle1_low
Data[5]	Angle 1 is in high_byte	Angle1_high
Data[6]	Angle 2 is in low_byte	Angle2_low
Data[7]	Angle 2 is in high_byte	Angle2_high
Data[8]	Angle 3 is in low_byte	Angle3_low
Data[9]	Angle 3 is in high_byte	Angle3_high
Data[10]	Angle 4 is in low_byte	Angle4_low
Data[11]	Angle 4 is in high_byte	Angle4_high
Data[12]	Angle 5 is in low_byte	Angle5_low
Data[13]	Angle 5 is in high_byte	Angle5_high
Data[14]	Angle 6 is in low_byte	Angle6_low
Data[15]	Angle 6 is in high_byte	Angle6_high
Data[16]	Specified speed	Sp
Data[17]	End frame	0xfa

19).read speed

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x40
Data[4]	End frame	0xfa

20).set speed

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x34
Data[4]	Specified speed	Sp
Data[5]	End frame	0xfa

21).read feedoverride

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x42
Data[6]	End frame	0xfa

22).read acceleration

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x44
Data[4]	End frame	0xfa

23).read the minimum angle of the joint

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x4a
Data[4]	Joint number	Joint
Data[5]	End frame	0xfa

24).read the maximum angle of the joint

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x4b
Data[4]	Joint number	Joint_no
Data[5]	End frame	0xfa

25).check the connection

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x50
Data[4]	Servo number	Servo_no
Data[5]	End frame	0xfa

26).check whetherjoints are fully powered on

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x51
Data[4]	End frame	0xfa

27).read the status of servos

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x04
Data[3]	Instruction frame	0x53
Data[4]	Servo number	Servo_no
Data[5]	Data	Data_id
Data[6]	End frame	0xfa

28).set servos zero

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x55
Data[4]	Servo number	Servo_no
Data[5]	End frame	0xfa

29).brake single motor

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x56
Data[4]	Servo number	Servo_no
Data[5]	End frame	0xfa

30).set atom pin mode

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x04
Data[3]	Instruction frame	0x60
Data[4]	Pin number	Pin_no
Data[5]	Pin mode	Pin_mode

Data[6]	End frame	0xfa

31).program pause

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x26
Data[4]	End frame	0xfa

32).program continue to run

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x27
Data[6]	End frame	0xfa

33).program stop

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x28
Data[4]	End frame	0xfa

34).set servos state

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x05
Data[3]	Instruction frame	0x52
Data[4]	Servo number	Servo_no
Data[5]	Data	Data_id
Data[6]	Servo state	Servo_data
Data[7]	End frame	0xfa

35).free mode (track recording & learn by hand)

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x02
Data[3]	Instruction frame	0x13
Data[4]	End frame	0xfa

36).set the color of lights of atom

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x05
Data[3]	Instruction frame	0x6a
Data[4]	R	R

Data[5]	G	G
Data[6]	B	B
Data[7]	End frame	0xfa

37).set the angle of jaw

Data field	Description	Data
Data[0]	Identify the frame	0xfe
Data[1]	Identify the frame	0xfe
Data[2]	Data length frame	0x03
Data[3]	Instruction frame	0x66
Data[4]	Jaw opening and closing	States
Data[5]	End frame	0xfa

4.Contact Us

If you have any need for help, please contact us shown as following.

Shenzhen Elephant Robotics Technology Co., Ltd

Address: B7, Yungu Innovative Industrial Park 2, Nanshan, Shenzhen, China

Email: support@elephantrobotics.com

Phone: +86(0755)-8696-8565 (working day 9:30-18:30)

Website: www.elephantrobotics.com