# 1 DH5 ModbusAPI-

## 1.1 Class introduction

DH5ModbusAPI is a Python API based on the Modbus protocol and DH 5 device communication, used to control the initialization, parameter setting and status reading of 6-axis devices.

### 1.1.1 generic attribute

| Parameter name | Parameter functions |
|---|---|
| **port** | Serial port name, such as `COM6` |
| **modbus_id** | Modbus Device ID |
| **baud_rate** | Communication baud rate |
| **stop_bits** | Stop bit |
| **parity** | The verification method supports `N` (no verification), `E` (even verification) and `O` (odd verification) |
| **serial_connection** | Serial port connection object |

Table 1 Class initialization parameters

| Status name | State variable value |
|---|---|
| SUCCESS | 1: Successful execution |
| ERROR_CONNECTION_FAILED | 2: Connection error |
| ERROR_INVALID_RESPONSE | 3: No reply |
| ERROR_INVALID_COMMAND | 4: Input instruction error |

## 1.2 Class initialization

**__init__(port='COM6', modbus_id=1, baud_rate=115200, stop_bits=1, parity='N')**

Initialize the API instance.

parameter:

- `port` (str): Serial port name, default value is `COM6`.
- `modbus_id` (int): Modbus device ID, default value is `1`.
- `baud_rate` (int): Communication baud rate, default value is `115200`.
- `stop_bits` (int): Number of stop bits. The default value is `1`.
- `parity` (str): Verification method, supports `N`, `E` or `O`, the default value is `N`.

---

## 1.3 Serial port configuration and instruction sending

### 1.3.1 open_connection()

Function function:
Open the serial port connection.

returned value:
-Return when successfully opened: SUCCESS.
-If the operation fails, throw an exception and return the reason for failure.

### 1.3.2 close_connection()

Function function:
Close the serial port connection.

returned value:
-Return SUCCESS when successfully closed.

### 1.3.3 send_modbus_command(function_code, register_address, data=None, data_length=None)

Function function:
Send the Modbus command to the DH5 device and receive the response.

input parameter:

1. function_code: Modbus function code, supporting '0x03' (read register), '0x06' (write a single register), and '0x10' (write multiple registers).
2. register_address: Register address. See the DH5 Smart Hand User Manual.docx for supported register addresses
3. data (int or list, optional): The data to be written, the value corresponding to the register address.
4. data_length` (int, optional): The number of registers to read or write.

returned value:
-If successful, return the response data of the device.
-If the failure occurs, it returns the ERROR_INVALID_RESPONSE and ERROR_CRC_CHECK_FAILED error codes. Specific analysis is required.

---

### 1.3.4 _build_request(function_code,register_address, data_length=1, value=None, values=None)

Function:
Build the Modbus request message.

input parameter:
1. function_code (int): Function code.
2. register_address (int): Register address.
3. data_length` (int): The number of registers to read or write, default is `1`.
4. value (int, optional): The value when writing a single register.
5. values (list, optional): A list of values when writing to multiple registers.

returned value:
- `bytearray`: The request message to be built.

### 1.3.5 _calculate_crc(data)

Function:
Calculate the CRC checksum of the Modbus message.

input parameter:
1. `data` (bytes): The message data to calculate the CRC.

returned value:
- `int`: The calculated CRC checksum.

### 1.3.6 _parse_response(response, function_code)

Function:
Parse the response message returned by the device.

input parameter:
1. response (bytes): The received Modbus response message.
2. function_code (int): The function code used in the request.

returned value:
-If successful, return the parsed data.
-If the failure occurs, ERROR_INVALID_RESPONSE and ERROR_CRC_CHECK_FAILED error codes will be returned. Specific analysis is required.

### 1.3.7 set_config(modbus_id=None,baud_rate=None, stop_bits=None, parity=None)

Function:
Set serial communication configuration.

input parameter:
1. modbus_id (int, optional): Modbus device ID.
2. baud_rate (int, optional): baud rate.
3. stop_bits (int, optional): The number of stop bits.
4. parity (str, optional): Verification method.

## 1.4 Modbus Parameter configuration

### 1.4.1 set_uart_config(self,modbus_id=None, baud_rate=None, stop_bits=None, parity=None)

Function:
Configure the UART communication parameters and write them into the relevant registers of the Modbus device.

input parameter:
1. modbus_id (int, optional): Modbus from device ID.
2. baud_rate (int, optional): Communication baud rate.
3. stop_bits (int, optional): Stop bits.
4. parity (int, optional): Verification mode (for example: 0 means no verification, 1 means odd verification, 2 means even verification).

### 1.4.2 set_save_param(self, flag = None)

Function:
Set the flag that saves the parameters and write it to the Modbus register address 0x0300.

input parameter:
Flag (int, optional): The flag value to save parameters. For example:
-1 indicates that the current configuration is saved to permanent storage on the device (such as Flash).
-0 means not to save.

## 1.5 initialise

### 1.5.1 initialize(mode)

Function:
Initializes all 6-axis devices at once.

input parameter:
1. mode (int): Initialization mode:
2. `0b01`: Closed initialization.
- `0b10`: Open initialization.
- `0b11`: Initialize the total search trip.

returned value:
-Command execution results.

---

### 1.5.2 initialize_axis(axis, mode)

Function:
Initializes the axis to the specified mode.

input parameter:
1. `axis` (int): The axis number to initialize (1-6).
2. `mode` (int): Initialization mode:
3. `0b01`: Closed initialization.
- `0b10`: Open initialization.
- `0b11`: Find the total trip.

returned value:
-The result of the command execution.

### 1.5.3 check_initialization()

Function function:
Check the initialization status of all six axes.

returned value:
- `dict`: Initialization state of each axis:
  - ` "not initialized" `: Not initialized
  - ` "initialized" `: Initialized
  - ` "initializing" `: Initialization in prog-
  ress

## 1.6 Set parameter instructions

### 1.6.1 set_axis_position(axis, position)

Function function:
Set the position of the specified axis.

input parameter:
1. Axis (int): Axis number (1-6).
2. position (int): Target position.

** returned value:**
-Command execution result, SUCCESS.

### 1.6.2 set_axis_speed(axis, speed)

Function function:
Set the speed of the specified axis.

input parameter:
1. `axis` (int): Axis number (1-6).
2. `speed` (int): Target speed.

** returned value:**
-Command execution results.

### 1.6.3 set_axis_force(axis, force)

Function:
Set the force on the specified axis.

input parameter:
1. - `axis` (int): Axis number (1-6).
2. - `force` (int): Target force value.

** returned value:**
-Command execution results.

## 1.7 Get feedback parameter instructions

### 1.7.1 get_axis_position(axis)

function:
Get the position of the specified axis.

input parameter:
- `axis` (int): Axis number (1-6).

returned value:
-Location data or error information.

### 1.7.2 get_axis_speed(axis)

function:
Get the speed of the specified axis.

input parameter:
1. - `axis` (int): Axis number (1-6).

returned value:

-Speed data or error information.

### 1.7.3 get_axis_current(axis)

function:
Get the current of the specified axis.

input parameter:
- `axis` (int): Axis number (1-6).

returned value:
-Current data or error information.

## 1.8 error handling

### 1.8.1 get_history_faults()

Function function:
Obtain 21 historical faults.

returned value:
All historical troubleshooting information.

### 1.8.2 get_cur_faults()

Function function:
Obtain the fault status of the device.

returned value:
-Fault data or error information.

### 1.8.3 reset_faults()

Function function:
Reset the faulty status of the device.

returned value:
-Command execution results.

## 1.9 instance

```
Initialize an DH5ModbusAPI instance
api = DH5ModbusAPI(port='COM6', baud_rate=115200)

# interface
api.open_connection()

Initialize the device
mode = 0b10 # Open mode
status = api.initialize(mode)
if status == api.SUCCESS:
    Print ("Device initi-
alization successful!")
Else:
    Print ("Device initialization failed!")


Set the motion position of the axis
api.set_axis_position(2, 10)
api.set_axis_position(3, 10)
api.set_axis_position(4, 10)
api.set_axis_position(5, 10)
api.set_axis_position(1, 500)
api.set_axis_position(6, 500)

Set the speed
api.set_axis_speed(1, 10)
api.set_axis_speed(2, 10)
api.set_axis_speed(3, 10)
api.set_axis_speed(4, 10)
api.set_axis_speed(5, 10)
api.set_axis_speed(6, 10)

Set the force
api.set_axis_force(1, 100)
api.set_axis_force(2, 100)
api.set_axis_force(3, 100)
api.set_axis_force(4, 100)
api.set_axis_force(5, 100)
api.set_axis_force(6, 100)
```