1 DH5ModbusAPI - 中文文档

1.1 类简介

DH5ModbusAPI 是一个基于 Modbus 协议与 DH5 设备通信的 Python API，用于控制 6 轴设备的初始化、参数设置和状态读取。

1.1.1 类属性

参数名字	参数功能
port	串口名称，例如 `COM6`
modbus_id	Modbus 设备 ID
baud_rate	通信波特率
stop_bits	停止位
parity	校验方式，支持 `N` (无校验)、`E` (偶校验) 和 `O` (奇校验)
serial_connection	串口连接对象

表 1 类初始化参数

状态名称	状态变量值
SUCCESS	1 ：成功执行
ERROR_CONNECTION_FAILED	2 ：连接出错
ERROR_INVALID_RESPONSE	3 ：没有回复
ERROR_INVALID_COMMAND	4 ： 输入指令错误

1.2 类的初始化

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

初始化 API 实例。

参数：

- `port` (str): 串口名称，默认值为 `COM6`。
- `modbus_id` (int): Modbus 设备 ID，默认值为 `1`。
- `baud_rate` (int): 通信波特率，默认值为 `115200`。
- `stop_bits` (int): 停止位数量，默认值为 `1`。
- `parity` (str): 校验方式，支持 `N`、`E` 或 `O`，默认值为 `N`。

1.3 串口配置以及指令发送

1.3.1 open_connection()

函数功能：
打开串口连接。

返回值：
- 成功打开时返回 ：SUCCESS。
- 如果失败，抛出异常，返回失败原因。

1.3.2 close_connection()

函数功能：
关闭串口连接。

返回值：
- 成功关闭时返回 ：SUCCESS。

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

函数功能：
发送 Modbus 指令到 DH5 设备并接收响应。

输入参数：

1. function_code ： Modbus 功能码，支持 ‘0x03’(读寄存器) , ’0x06’(写单个寄存器) , ‘0x10’（写多个寄存器）。
2. register_address : 寄存器地址。支持的寄存器地址见 《DH5 灵巧手使用手册.docx》
3. data (int 或 list，可选) ： 要写入的数据，对应寄存器地址的值。
4. data_length`(int，可选) : 要读取或写入的寄存器数量。

返回值：
- 如果成功，返回设备的响应数据。
- 如果失败，会返回 ERROR_INVALID_RESPONSE 和 ERROR_CRC_CHECK_FAILED 错误代码。需要具体分析。

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

函数功能：
构建 Modbus 请求消息。

输入参数：
1. function_code(int): 功能码。
2. register_address(int): 寄存器地址。
3. data_length` (int): 要读取或写入的寄存器数量，默认为 `1`。
4. value (int，可选): 写单个寄存器时的值。
5. values (list，可选): 写多个寄存器时的值列表。

返回值：
- `bytearray`: 构建的请求消息。

1.3.5 _calculate_crc(data)

函数功能：
计算 Modbus 消息的 CRC 校验值。

输入参数：
1. `data` (bytes): 要计算 CRC 的消息数据。

返回值：
- `int`: 计算出的 CRC 校验值。

1.3.6 _parse_response(response, function_code)

函数功能：
解析设备返回的响应消息。

输入参数：
1. response (bytes): 接收到的 Modbus 响应消息。
2. function_code (int): 请求时使用的功能码。

返回值：
- 如果成功，返回解析的数据。
- 如果失败，会返回 ERROR_INVALID_RESPONSE 和 ERROR_CRC_CHECK_FAILED 错误代码。需要具体分析。

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

函数功能：
设置 串口通信配置。

输入参数：
1. modbus_id (int，可选) : Modbus 设备 ID。
2. baud_rate (int，可选): 波特率。
3. stop_bits (int，可选): 停止位数量。
4. parity (str，可选): 校验方式。

1.4 Modbus 参数配置

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

函数功能：
配置 UART 通信参数，并将其写入 Modbus 设备的相关寄存器。

输入参数:
1. modbus_id (int, 可选): Modbus 从设备 ID。
2. baud_rate (int, 可选): 通信波特率。
3. stop_bits (int, 可选): 停止位。
4. parity (int, 可选): 校验方式 (例如: 0 表示无校验，1 表示奇校验，2 表示偶校验)。

1.4.2 set_save_param(self, flag = None)

函数功能：
设置保存参数的标志位，并将其写入 Modbus 寄存器地址 0x0300。

输入参数:
flag (int, 可选): 保存参数的标志位值。例如：
- 1 表示保存当前配置到设备永久存储（如 Flash）。
- 0 表示不保存。

1.5 初始化

1.5.1 initialize(mode)

函数功能：
一键初始化所有 6 轴设备。

输入参数：
1. mode (int): 初始化模式：
2. `0b01`: 闭合初始化。
- `0b10`: 张开初始化。
- `0b11`: 初始化查找总行程。

返回值：
- 命令执行结果。

1.5.2 initialize_axis(axis, mode)

函数功能：
初始化指定轴到指定模式。

输入参数：
1. `axis` (int): 要初始化的轴编号（1-6）。
2. `mode` (int): 初始化模式：
3. `0b01`: 闭合初始化。
- `0b10`: 张开初始化。
- `0b11`: 查找总行程。

返回值：
- 命令执行结果。

1.5.3 check_initialization()

函数功能：
检查所有 6 轴的初始化状态。

返回值：
- `dict`: 每个轴的初始化状态：
  - `"not initialized"`: 未初始化
  - `"initialized"`: 已初始化
  - `"initializing"`: 初始化中

1.6 设置参数指令

1.6.1 set_axis_position(axis, position)

函数功能：
设置指定轴的位置。

输入参数：
1. axis (int): 轴编号（1-6）。
2. position (int): 目标位置。

返回值：
- 命令执行结果，SUCCESS。

1.6.2 set_axis_speed(axis, speed)

函数功能：
设置指定轴的速度。

输入参数：
1. `axis` (int): 轴编号（1-6）。
2. `speed` (int): 目标速度。

返回值：
- 命令执行结果。

1.6.3 set_axis_force(axis, force)

函数功能：
设置指定轴的力。

输入参数：
1. - `axis` (int): 轴编号（1-6）。
2. - `force` (int): 目标力值。

返回值：
- 命令执行结果。

1.7 获取反馈参数指令

1.7.1 get_axis_position(axis)

函数：
获取指定轴的位置。

输入参数：
- `axis` (int): 轴编号（1-6）。

返回值：
- 位置数据或错误信息。

1.7.2 get_axis_speed(axis)

函数：
获取指定轴的速度。

输入参数：
1. - `axis` (int): 轴编号（1-6）。

返回值：

- 速度数据或错误信息。

1.7.3 get_axis_current(axis)

函数：
获取指定轴的电流。

输入参数：
- `axis` (int): 轴编号（1-6）。

返回值：
- 电流数据或错误信息。

1.8 错误处理

1.8.1 get_history_faults()

函数功能：
获取 21 个历史故障。

返回值：
所有的历史故障信息。

1.8.2 get_cur_faults()

函数功能：
获取设备的故障状态。

返回值：
- 故障数据或错误信息。

1.8.3 reset_faults()

函数功能：
复位设备的故障状态。

返回值：
- 命令执行结果。

1.9 示例

```
# 初始化一个 DH5ModbusAPI 实例
api = DH5ModbusAPI(port='COM6', baud_rate=115200)

# 连接设备
api.open_connection()

# 初始化设备
mode = 0b10   # 打开模式
status = api.initialize(mode)
if status == api.SUCCESS:
    print("设备初始化成功！")
else:
    print("设备初始化失败！")


#  设置 轴的运动位置
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)

# 设置速度
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)

# 设置力
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)
```