# Modbus-RTU

# 用户手册

深圳市杰美康机电科技有限公司

# 说明

本手册的所有内容，著作财产权归深圳市杰美康机电有限公司所有，未经深圳市杰美康机电有限公司许可，任何单位或个人不得随意仿制、拷贝、撰抄。本手册无任何形式的担保、立场表达或其它暗示。如有本手册所提到的产品的信息，所引起的直接或间接的资料流出，导致利益损失后果，深圳市杰美康机电有限公司与所属员工不承担任何责任。除此以外，本手册提到的产品及其资料仅供参考，内容如有更新，恕不另行通知。



深圳市杰美康机电有限公司

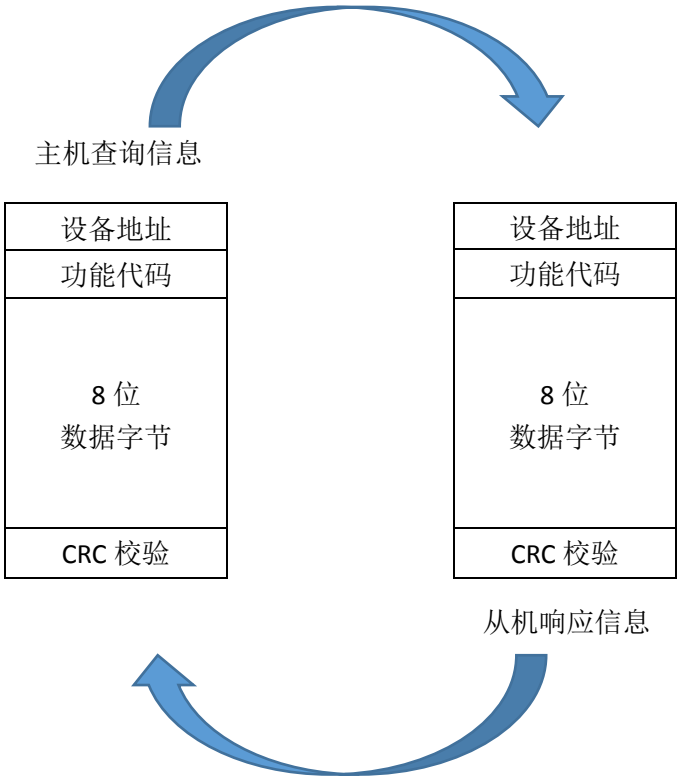
| 版本 | 编写 | 核准 |
| --- | --- | --- |
| V2.1 | 研发部 | 研发部 |

# 目录

# 1 Modbus-RTU 定义

Modbus 协议，由 MODICON 公司设计，是一允许主站和一个或多个从站共享数据的总线协议，数据由 16 位寄存器构成。主站可以 RW 单个寄存器或多个寄存器。本文讲述基于深圳市杰美康机电有限公司 ModBus 总线驱动器串口通信协议的使用说明，该通信协议完全遵循 Modbus-RTU 协议。

串口通讯端口是使用 RS-485 兼容的串行接口，定义了连接器，接线电缆，信号等级，传输波特率和奇偶校验。控制器通讯使用主从技术，即主机能启动数据传输，查询。而其他设备（从机）返回对查询做出的响应，或处理查询所要求的动作。主机设备应包括主从处理器和 PLC。从机包括伺服驱动器和步进驱动器。其主从查询-反馈机制如下所示：



# 2 Modbus-RTU 报文格式

Modbus-RTU 是一种主从技术，且 CRC 校验范围为从设备地址位到数据位；个功能码的详细报文格式，请见附录。Modbus-RTU 的消息帧如下：

| 地址域 | 功能码 | 数据 | CRC 校验码（2 个字节） |
|---|---|---|---|

# 3 Modbus-RTU 接线

Modbus-RTU 与标准的 RS-232 或 RS-485 有共同的物理层，可以配置 1~32 个从站地址；以拓扑结构构建 RS-422/485 网络，通常在最后的从站设备并联 120 欧姆的终端电阻。JMC 公司 Modbus-RTU 总线驱动器仅支持基于 RS-485 两线半双工的接线方式。

# 4 传输协议

## 消息处理：

作为从站，驱动器将等待接收主站发出的数据，当接收到数据后，判断是否为该轴数据并进行响应。当数据间超时 2s 未接收到下一帧数据，驱动器默认该从站数据收到错误数据，将前面的数据清除，重新接收新的数据。

## 消息帧结构：

“主站-从站”数据交换以从站地址开始，然后是功能代码、传输数据。 数据域的结构取决于使用的功能代码。消息帧结束时传送的是 CRC 校验码。

| 地址 | 功能 | 数据 | CRC 校验 |
|---|---|---|---|
| 1 字节 | 1 字节 | n 字节 | 2 字节 |

| | |
|---|---|
| 地址 | Modbus 从站地址 1~255 |
| 功能 | Modbus 功能码 |
| 数据 | Modbus 数据：寄存器地址，寄存器地址个数，寄存器数据 |
| CRC 校验 | 消息帧校验合 |

## 广播消息：

主站使用从站地址 0 对总线上的所有从站进行寻址。
广播消息仅允许与写 功能代码 0x06 和 0x10 相结合。
广播消息不需要从站的应答消息帧。

## 支持如下的 Modbus 功能码：

1) 0x03：读保持寄存器；
2) 0x06：写单个寄存器；
3) 0x10：写多个寄存器。

## 数据域 DATA：

数据域 DATA 用于传送功能代码特定数据，例如：
字节数、寄存器起始地址、读写地址数、数据值等

## CRC 校验：

消息帧的最后是由 2 个字节组成的 CRC 16 校验和。校验和是按如下多项式计算的：
$X^{16} + X^{15} + X^2 + 1$。先传输低位字节，然后传输高位字节。

## 异常响应：

当检测到主站请求消息错误时，比如接收寄存器地址非法，将置位功能码的最高位。
随后传输的是一个字节的错误代码，描述错误内容。

## 错误代码及其描述

| 错误代码 | 符合 ModBus 规范的定义 | 短描述 |
|---|---|---|
| 1 | 非法功能 | 功能代码非法 |
| 2 | 非法数据地址 | 从站具有非法的数据地址 |
| 3 | 非法数据值 | 从站具有非法的数据值 |
| 4 | 关联设备发生故障 | 从站出现内部错误 |
| 5 | 写只读寄存器 | 不能执行相应的写操作 |
| 6 | 忙，拒收消息 | 从站尚未准备好接收消息 |
| 7 | 读只写寄存器 | 不能执行相应的读操作 |
| 8 | 数据写入错误 | 字节个数与寄存器数不匹配 |
| 9 | 奇偶校验错误 | 奇偶校验发生错误（配置奇偶校验情况下） |
| 10 | CRC 校验错误 | CRC 校验发生错误，请求重发数据 |
| 11 | 读写的寄存器不存在 | 读写的寄存器地址不存在，无法完成 |

# 5 通讯协议

## 0x1001 错误寄存器

设备的内部错误将会映射到该寄存器。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 1001h | UNSIGNED16 | RO | 0 |

位 0: 常规错误
位 1: 电流错误
位 2: 电压错误
位 3: 温度报警
位 4: 通信错误
位 5: 位置超差 (仅限步进伺服和伺服驱动器)
位 6: 保留 (默认 0)
位 7: 电机缺相

## 0x1008 设备信息寄存器

设备厂商及具体型号

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 1008h | Vis-String | RO | JMC XXX |

## 0x1009 硬件版本号寄存器

该设备硬件版本号

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 1009h | Vis-String | RO | XXX |

## 0x1008 软件版本号寄存器

该设备内部软件版本号

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 1008h | Vis-String | RO | JMC XXX |

## 0x6000 格式存器

32 位寄存器存储格式，当 0x6000=0，32 位寄存器高 16bit 在前一个地址，低 16 位在后一个地址；当 0x6000=1，32 位寄存器低 16bit 在前一个地址，高 16 位在后一个地址；

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6000h | UNSIGNED16 | RO | 0 |

# 0x6040 控制字

驱动器的状态和运动的控制字。用于使能、禁止驱动器的电源和刹车输出，不同操作模式下启动和停止电机，清除错误报警等。

控制字的结构体说明

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6040h | UNSIGNED16 | WO | 0 |

控制字的位定义：

| 字节 | BIT | 位定义 | | | |
|---|---|---|---|---|---|
| | | 位置模式 | 速度模式 | 回零模式 | 转矩模式 |
| LSB | 0 | 0—>1：参数变量初始化（刹车关闭）。 | | | |
| | 1 | 0—>1：驱动器给电运行（刹车打开）。 | | | |
| | 2 | 0—>1：快速停止 | | | |
| | 3 | 0—>1：电机上电使能<br>1—>0：电机不使能 | | | |
| | 4 | 0—>1：位置采样 | 保留 | 开始回零 | 保留 |
| | 5 | 0：完成当前位置再执行下一次位置；<br>1：直接运行到下一次给定位置； | 保留 | 保留 | 保留 |
| | 6 | 0：绝对定位；　　1：相对定位； | 保留 | 保留 | 保留 |
| | 7 | 0—>1：错误复位清除 | | | |
| MSB | 8 | 0—>1：暂停<br>1—>0：正常运行 | | | |
| | 9 | 0：完成前一次位置停止再运行下一位置；<br>1：当前位置与下一位置间不停止； | 保留 | 保留 | 保留 |
| | 10 | 保留 | | | |
| | 11 | 保留 | | | |
| | 12 | 保留 | | | |
| | 13 | 保留 | | | |
| | 14 | 保留 | | | |
| | 15 | 保留 | | | |

控制字状态切换命令：

| 命令 | 控制字位 | | | | |
|---|---|---|---|---|---|
| | 错误复位 bit8 | 使能操作 bit3 | 快速停止 bit2 | 使能电压 bit1 | 初始化设备 bit0 |
| 关机 | 0 | X | 1 | 1 | 0 |
| 初始化设备 | 0 | 0 | 1 | 1 | 1 |
| 使能设备操作 | 0 | 1 | 1 | 1 | 1 |
| 快速停止 | 0 | X | 0 | 1 | X |
| 禁止使能 | 0 | 0 | 1 | 1 | 1 |
| 错误复位 | 0→1 | X | X | X | X |

# 0x6041 状态字

状态字只能被读取，反应当前驱动器的状态。

状态字结构体说明：

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6041h | UNSIGNED16 | RO | 0 |

状态字的位定义：

| 字节 | BIT | 位定义 | | | | | |
|---|---|---|---|---|---|---|---|
| | | 位置模式 | 速度模式 | | 回零模式 | | 转矩模式 |
| LSB | 0 | 0：未准备好初始化 | 1：准备好初始化 | | | | |
| | 1 | 0：初始化驱动器未完成 | 1：初始化驱动器完成 | | | | |
| | 2 | 0：电机未使能 | 1：电机上电使能 | | | | |
| | 3 | 0：驱动器正常 | 1：驱动器错误状态 | | | | |
| | 4 | 0：驱动器未工作 | 1：驱动器正常上电工作 | | | | |
| | 5 | 0：正常运行 | 1：快速停止 | | | | |
| | 6 | 0：正常运行 | 1：设备进入初始化状态 | | | | |
| | 7 | 0：正常运行 | 1：警告 | | | | |
| MSB | 8 | 0：正常运行 | 1：电机暂停运行 | | | | |
| | 9 | 0：电机未运行 | 1：电机运行标志 | | | | |
| | 10 | 0：未到位 | 0 | Bit8=0:未达目标速度<br>Bit8=1:减速 | 0 | Bit8=0:未达回零位置<br>Bit8=1:减速 | 保留 |
| | | 1：到位标志 | 1 | Bit8=0:达到目标速度<br>Bit8=1:速度为 0 | 1 | Bit8=0:达到回零位置<br>Bit8=1:速度为 0 | |
| | 11 | SW 机械原点限位标志 | | | | | |
| | 12 | 0: 不可设置新位置 | 0：速度不为 0 | | 0：回零未完成 | | 保留 |
| | | 1：可设置新位置 | 1：速度等于 0 | | 1：回零完成 | | |
| | 13 | 0：正常运行 | 0：未达最大加速度 | | 0：回零无错误 | | |
| | | 1：电机位置超差 | 1：达到最大加速度 | | 1：回零发生错误 | | |
| | 14 | CW 顺时针方向限位标志 | | | | | |
| | 15 | CCW 逆时针方向限位标志 | | | | | |

状态字指示设备状态：

| 状态字值（二进制） | State |
|---|---|
| XXXX XXXX X0XX 0000 | 设备未初始化 |
| XXXX XXXX X01X 0001 | 设备初始化 |
| XXXX XXXX X01X 0011 | 设备上电运行 |
| XXXX XXXX X01X 0111 | 设备使能 |
| XXXX XXXX X00X 0111 | 激活设备快速停止 |
| XXXX XXXX X0XX1111 | 设备发生错误报警 |
| XXXX XXXX X0XX1000 | 设备处于错误状态 |

## 0x605A 快停代码

快停代码决定在快速停止命令时停止的方式

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 605Ah | INTEGER16 | RW | 0 |

| 快停代码 | 执行操作 |
|---|---|
| 1 | 以当前减速度停止 |
| 2 | 以快速停止速度停止 |
| 3…32767 | 直接立即停止 |

## 0x6060 操作模式

操作模式用于选择相应的运动模式

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6060 | INTEGER16 | WO | 0 |

| 操作模式代码 | 运行模式 |
|---|---|
| 1 | 位置模式 |
| 3 | 速度模式 |
| 4 | 转矩模式（伺服）暂不支持 |
| 6 | 回零模式 |

该设备支持速度模式、位置模式以及回零模式三种模式。

## 0x6061 模式代码应答

模式代码应答表示当前的操作模式。返回值与相应的模式状态相关(index 6060h)。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6061 | INTEGER16 | RO | 0 |

## 0x6064 实际位置

当前位置表示位置模式下当前时刻的位置。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6064 | INTEGER32 | RO | 0 |

## 0x606C 实际速度

当前速度表示当前时刻的速度的大小

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 606C | INTEGER32 | RO | 0 |

单位 rps/min unit.

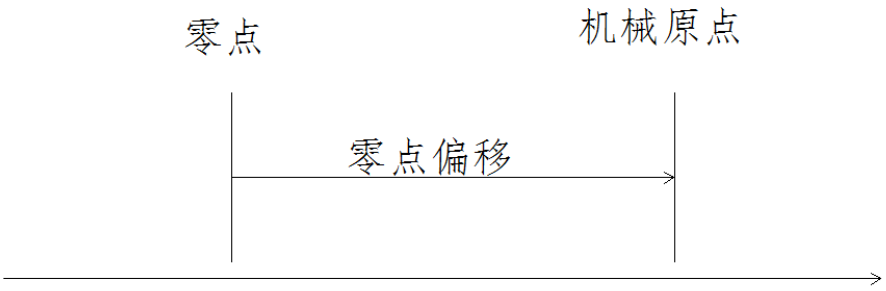
## 0x607A 目标位置

目标位置为在位置模式下主站设备给定驱动器应该移动的位置，与之相关的参数为目标速度，加速度以及减速度。目标位置跟不同的细分相关，可根据控制字的bit6看作计算或者相关量。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 607A | INTEGER32 | RW | 0 |

## 0x607C 零位偏移

零位偏移是指零点位置与机械原点的偏移位置，在找到机械原点以后相对于机械原点偏移一段距离后将所有的参数清零。这是下图所示。




| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 607C | INTEGER32 | RW | 0 |

## 0x6081 目标速度

目标速度为驱动器运行过程中匀速运行的速度

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6081 | INSIGNED32 | RW | 0 |

其设置值为实际值的 10 倍，例如：该值为 100，则实际速度值为 10r/s 即 600r/min

## 0x6083 加速度

电机运行的加速度

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6083 | UNSIGNED16 | RW | 0 |

其设置值为实际值的 10 倍，例如：该值为 100，则实际加速度值为 10 rps/s.s

## 0x6084 减速度

电机运行的减速度

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6084 | UNSIGNED16 | RW | 0 |

其设置值为实际值的 10 倍，例如：该值为 100，则实际减速度值为 10 rps/s.s

## 0x6085 快停减速度

该减速度用于快速停止减速，根据快速停止模式选择。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6085 | UNSIGNED16 | RW | 0 |

其设置值为实际值的 10 倍，例如：该值为 100，则实际减速度值为 10 rps/s.s

## 0x6098 回零模式

该对变量选择回零模式

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6098 | INTEGER16 | RW | 0 |

## 0x6099 回机械原点速度

回机械原点速度：

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 6099 | UNSIGNED32 | RW | 0 |

***注意:回零取值范围：0~100000，超出该范围会导致回零动作失败***

## 0x609A 回零加减速度

回零加减速度用于回零模式下的速度的加减速大小。

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 609A | UNSIGNED16 | RW | 0 |

其设置值为实际值的 10 倍，例如：该值为 100，则实际减速度值为 10 rps/s.s

## 0x609B 回零点速度

回零点速度：

| 寄存器 | 数据类型 | 访问权限 | 默认值 |
|---|---|---|---|
| 609B | UNSIGNED32 | RW | 0 |

***TIP:回零取值范围：0~100000，超出该范围会导致回零动作失败***

# 6 Modbus-RTU 例程

以下例程是基于杰美康 IHHS57-R 的 Modbus-RTU 通信协议。

## 6.1 位置模式

位置模式：由加速度、减速度、目标速度以及目标位置等参数实现点到点的运动控制模式。当所有的参数设置完成后，驱动器将会按照相应的参数运行，在运动过程中可以实现在一个点运行的过程中设置下一个点的位置，从而实现连续运动控制。

## 使能驱动器的操作

在驱动器上电或者复位后，驱动器处于禁止状态。向驱动器控制寄存器写入控制字 0x000F, 使设备进入操作使能状态。

## 使能位置模式

使能位置模式，需要先将 0X01 写入对象字典 6060h 中。从对象字典 6061h 中查看
当前的运动模式。
当一个点位正在运行，可以设置一个新的点位，当前一个点位运行完成后，直接运行第二个点位。

## 设置运行参数

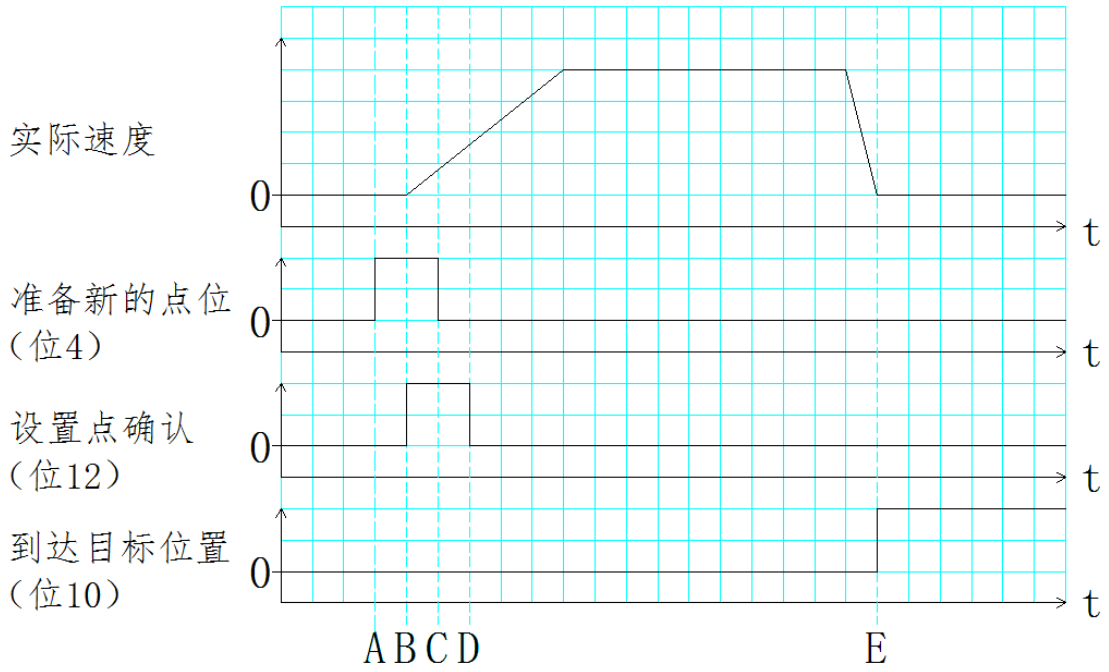
设置目标位置（607Ah）、速度（6081h）、加速度（6083h）、减速度（6084h）等参数。

## 启动/停止运行

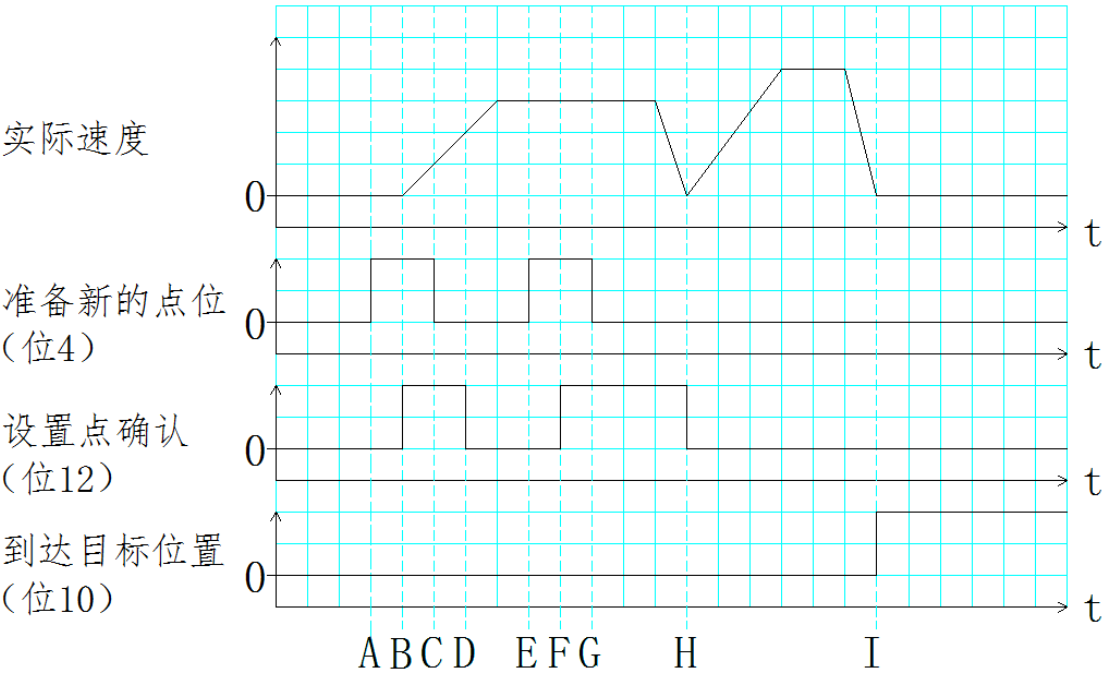
在写好位置运行的上述运行参数后，将控制字的 bit4 置 1，启动电机运转。在电机运转的过程中，将控制字的 bit8 置位，电机将暂停运转。

## 控制字的位

新的目标位置采集：在状态字 bit12 为 1 的情况下，将控制字的 bit4 由 0→1 跳变，将会采集当前位置值，状态字 bit12 为 0 的情况下，当前位置值将不会被采集。
如果第 9 位为 0，驱动器将会运行完成上一次设定的点位，并停止。如果第 9 位为 1，驱动器将按照当前速度完成前一次点位的运动，再转换为下一个点的运行速度运行到下一个点位。
将第 5 位置 1，新设置的位置将会直接生效，驱动器将会按照新的位置、新的速度立即运行。
第 6 位为 1，则运行的最终位置为前后两次位置之和，即相对定位。第 6 位为 0 时，则运行的最终位置为最新一次的位置，即绝对定位。

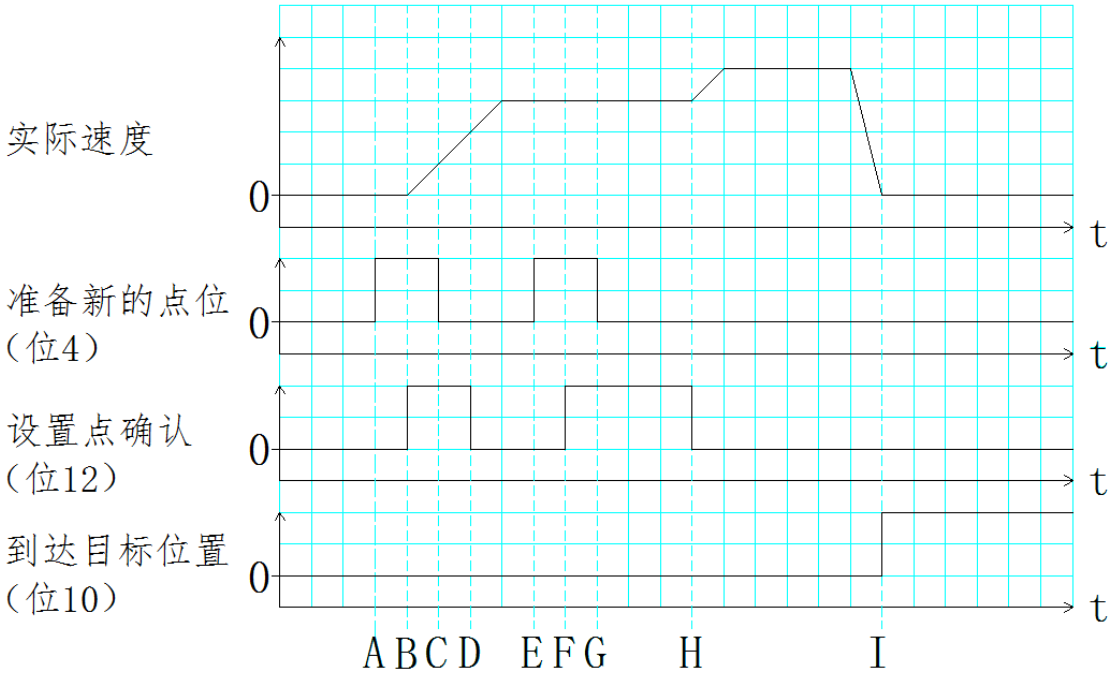



单点运动



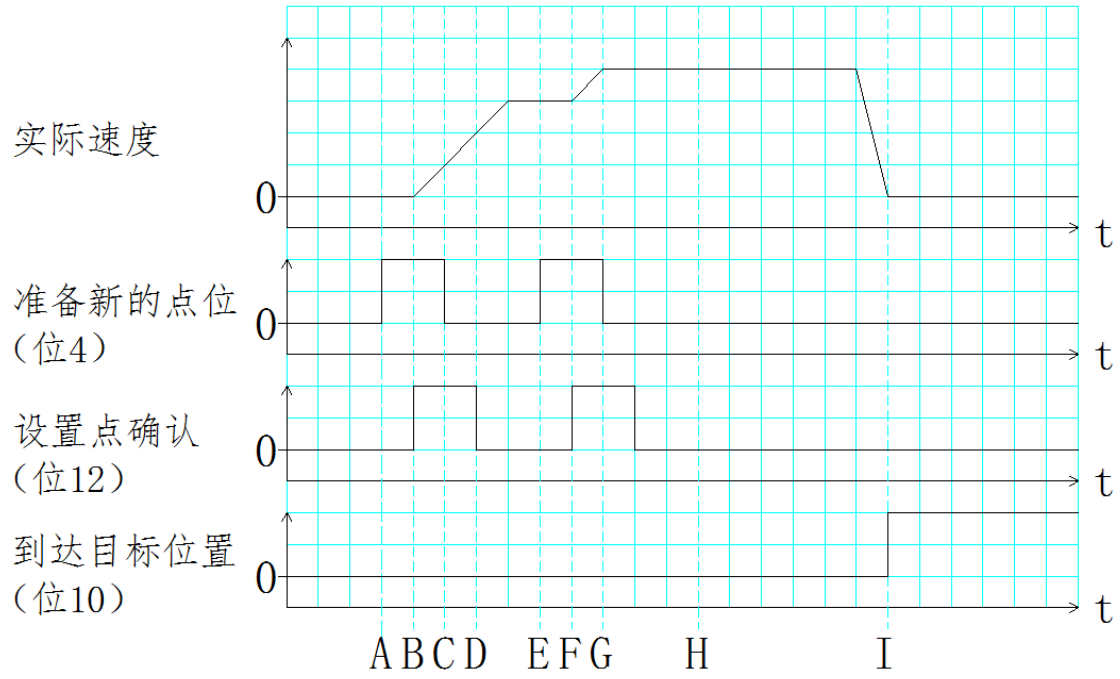

多点运动，在两点间会停止后再运行

在这种方式下控制字第 9 位和第 5 位均为 0，电机在两次运行的过程中将会停止。




多点运动，电机会连续不停的运动

在这种方式下，控制字的第 9 位为 1，第五位为 0，电机在到达第一个点前按照第一个点的速度匀速运行，在到达第一个点位以后按照第二个点的速度运行，期间电机不会停止。




多点运动，在设置好第二个点位后直接切换到第二个点的速度

在这种方式下，控制字的第 9 位置 1，并且第 5 位也置 1，电机会直接切换到第二个点运动速度，而不会完成第一个点的运动。电机运行的速度为连续运动。

# 编程说明

JMC’ ModBus-RTU 驱动器寄存器值：

| 寄存器地址 | 数值（实际值） | 类型 | 单位 | 说明 |
|---|---|---|---|---|
| 607Ah | 200000（200000） | INT32 | | 目标位置 |
| 6081h | 50（5） | INT32 | rps | 目标速度 |
| 6083h | 100（10） | UNINT16 | rps/s.s | 目标加速度 |
| 6084h | 100（10） | UNINT16 | Rps/s.s | 目标减速度 |
| 6085h | 100（10） | UNINT16 | Rps/s.s | 快停减速度 |

**注意**:16 位寄存器的地址数为 1,32 位寄存器的地址数为 2。32 位寄存器只能用 10 模式写入。

假设从站站号为 1 预写入控制字（0x6040 = 0x000F）,操作模式（0x6060 = 0x0001），目标速度（0x6081 = 0x00000064）,目标加速度(0x6083 = 0X0064),目标减速度（0x6084 = 0x0064），快停减速度（0x6085 = 0x0064）,目标位置（0x607A = 0x00030D40）。
可以用连续写入（0x10 功能码）模式，也可以用单个寄存器写入（0x06 功能码）模式。
连续写入模式如下：

写入报文：01 10 60 40 00 09 12 00 0F 00 01 00 00 00 64 00 64 00 64 00 64 00 03 0D 40 40 1C
反馈报文：01 10 60 40 00 09 1F DB

写入报文讲解如下：

| 报文： | 01 | 10 | 60 40 | 00 09 | 12 | 00 0F | 00 01 |
|---|---|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 总地址数 | 总的字节数 | 写入 0x6040 数据 | 写入 0x6060 数据 |

| 00 00 | 00 64 | 00 64 | 00 64 | 00 64 | 00 03 | 0D 40 | 40 1C |
|---|---|---|---|---|---|---|---|
| 写入 0x6081 高 16bit 数据 | 写入 0x6081 低 16bit 数据 | 写入 0x6083 数据 | 写入 0x6084 数据 | 写入 0x6085 数据 | 写入 0x607A 高 16bit 数据 | 写入 0x607A 低 16bit 数据 | CRC 校验 |

反馈报文讲解如下：

| 报文： | 01 | 10 | 00 09 | 1F DB |
|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 写入总地址个数 | CRC 校验 |

单个寄存器写入模式如下：
预写入目标加速度（0x6083）=100（0x0064）。
写入报文 1：01 06 60 83 00 64 67 C9
预写入目标减速度（0x6084）=100（0x0064）。
写入报文 2：01 06 60 84 00 64 D6 08

写入报文讲解：

| 报文 1： | 01 | 06 | 60 83 | 00 64 | 67 C9 |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 寄存器地址 | 写入数据 | CRC 校验 |

| 报文 2： | 01 | 06 | 60 7A | 00 03 | D6 08 |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 寄存器地址 | 写入数据 | CRC 校验 |

预写入目标位置(607A)=200000（0x00030D40），该寄存器数据类型为 INT32，只能以连续写入（0x10 模式）的模式写入数据,注意 0X6000 定义的模式不同，其 32 位寄存器写入数据的高地位区别，其报文如下：

若 0x6000=0：
写入报文：01 10 60 7A 00 02 04 00 03 0D 40 29 96
反馈报文：01 10 60 7A 00 02 7E 11
若 0x6000=1：
写入报文：01 10 60 7A 00 02 04 0D 40 00 03 9F 8F
反馈报文：01 10 60 7A 00 02 7E 11

写入报文讲解如下：

| 报文： | 01 | 10 | 60 7A | 00 02 | 04 | 00 03 | 0D 40 | 29 96 |
|---|---|---|---|---|---|---|---|---|
| 说名： | 地址 | 功能码 | 起始寄存器地址 | 总地址个数 | 总的字节数 | 写入 0x607A 高 16bit 数据 | 写入 0x607A 低 16bit 数据 | CRC 校验 |

反馈报文讲解如下：

| 报文： | 01 | 10 | 60 7A | 00 02 | 7E 11 |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 写入总地址个数 | CRC 校验 |

预读取目标位置寄存器（607A）,其报文如下:

若 0x6000=0：

读取报文：01 03 60 7A 00 02 FB D2

反馈报文：01 03 04 00 03 0D 40 0F 53

若 0x6000=1：

读取报文：01 03 60 7A 00 02 FB D2

反馈报文：01 03 04 0D 40 00 03 B9 4A

读取报文讲解如下：

| 报文： | 01 | 03 | 60 7A | 00 02 | FB D2 |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 总地址数 | 校验码 |

反馈报文讲解如下：

| 报文： | 01 | 03 | 04 | 00 03 | 0D 40 | B9 4A |
|---|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 总字节数 | 读取 0x607A 高 16bit 数据 | 读取 0x607A 低 16bit 数据 | 校验码 |

## 位置模式代码例程(0x6000=0)

*****给电机上电*****

01 06 60 40 00 01 57 DE “驱动器上电初始化”

01 06 60 40 00 03 D6 1F “驱动器正常运行，但电机未使能，松开刹车”

01 06 60 40 00 0F D6 1A “电机给电，处于可运行状态”

*****设置为位置模式*****

01 06 60 60 00 01 56 14

*****设置点位参数*****

01 10 60 81 00 02 04 00 00 00 0A 12 06 “设置速度为 1rps”

01 06 60 83 00 64 67 C9 “设置加速度为 10rps/s”

01 06 60 84 00 64 D6 08 “设置减速度为 10rps/s”

也可以 0x10 模式一次性写入点位参数

01 10 60 81 00 04 08 00 00 00 0A 00 64 00 64 10 52

*****单点运动*****

01 10 60 7A 00 02 04 00 03 0D 40 29 96 “设置目标位置为 200000”

01 06 60 40 00 1F D7 D6 “采集新位置并运行”

01 06 60 40 00 0F D6 1A “此处可先将控制字位 4 置 0，便于下一次快速采位”

*****多点运动，两点间需要停止电机，绝对定位*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 “设置速度为 5rps”

01 10 60 7A 00 02 04 00 03 0D 40 29 96 “设置目标位置为 200000”

01 06 60 40 00 1F D7 D6 “设置新位置”

01 06 60 40 00 0F D6 1A “此处可先将控制字位 4 置 0，便于下一次快速采位”

01 10 60 81 00 02 04 00 00 00 64 93 EA “设置第二段速度为 10rps”

01 10 60 7A 00 02 04 00 09 27 C0 17 54 “设置目标位置为 600000”

01 06 60 40 00 1F D7 D6 “设置新位置”

*****多点运动，两点间不需要停留*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 “设置速度为 5rps”

01 10 60 7A 00 02 04 00 03 0D 40 29 96 “设置目标位置为 200000”

01 06 60 40 02 5F D7 46 “设置新位置”

01 06 60 40 02 4F D6 8A “此处可先将控制字位 4 置 0，便于下一次快速采位”

01 10 60 81 00 02 04 00 00 00 64 93 EA “设置第二段速度为 10rps”

01 10 60 7A 00 02 04 00 09 27 C0 17 54 “设置目标位置为 600000”

01 06 60 40 02 5F D7 46 “设置新位置”

*****多点运动，直接改变运行速度*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 “设置速度为 5rps”

01 10 60 7A 00 02 04 00 03 0D 40 29 96 “设置目标位置为 200000”

01 06 60 40 00 7F D7 FE “设置新位置”

01 06 60 40 00 6F D6 32 “此处可先将控制字位 4 置 0，便于下一次快速采位”

01 10 60 81 00 02 04 00 00 00 64 93 EA “设置第二段速度为 10rps”

01 10 60 7A 00 02 04 00 09 27 C0 17 54 “设置目标位置为 600000”

01 06 60 40 02 7F D6 9E “设置新位置”

# 6.2 速度模式

## 速度模式介绍

速度模式：速度模式由目标速度（0x6081）、加速度(0x683)、减速度(0x6084)等参数确定。在运动的过程中可通过控制字的暂停位来暂停运动。

## 使能驱动器的操作

在驱动器上电或者复位后，驱动器处于禁止状态。向驱动器控制寄存器写入控制字 0x000F,使设备进入操作使能状态。在写入速度参数前可先将控制字位 8 置 1，暂停电机运行，在写入各项参数以后再使能运行操作。即给控制字（0x6040）写入 0x010F.

## 使能速度模式

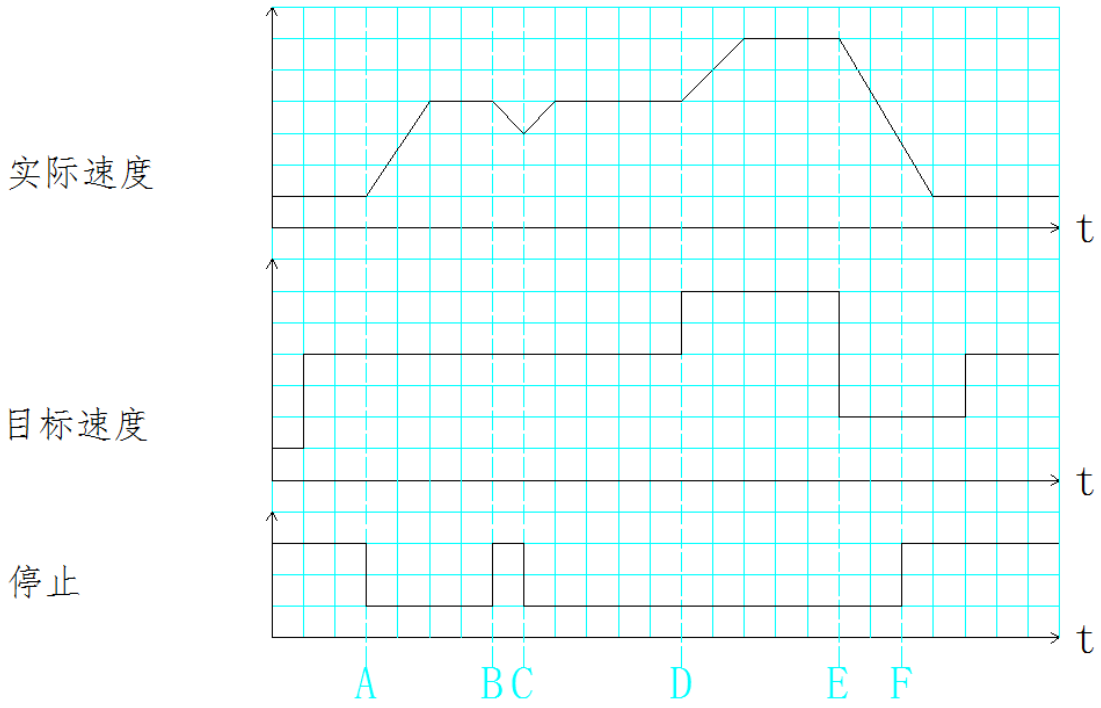
将 0x0003 写入对寄存器 0x6060h，使能速度模式。

## 设置运行参数

设置速度（0x6081h）、加速度（0x6083h）、减速度（0x6084h）等参数。

## 启动/停止

向控制字写 0x000F 启动电机运转。向控制字写 0x010F 停止电机运转。



# 速度模式代码例程(0x6000=0)

*****给电机上电*****

01 06 60 40 00 01 57 DE “驱动器上电初始化”

01 06 60 40 00 03 D6 1F “驱动器正常运行，但电机未使能，松开刹车”

01 06 60 40 00 0F D6 1A “电机给电，处于可运行状态”

*****设置为位置模式*****

01 06 60 60 00 03 D7 D5 “设置运行模式”

01 06 60 40 01 0F D7 8A “暂停电机运行”

*****设置点位参数*****

01 10 60 81 00 02 04 00 00 00 0A 12 06 “设置速度为 1rps”

01 06 60 83 00 64 67 C9 “设置加速度为 10rps/s”

01 06 60 84 00 64 D6 08 “设置减速度为 10rps/s”

也可以 0x10 模式一次性写入点位参数

01 10 60 81 00 04 08 00 00 00 0A 00 64 00 64 10 52

*****启动/停止运动*****

01 06 60 40 00 0F D6 1A “开始运行”

01 10 60 81 00 02 04 00 00 00 64 93 EA “设置速度为 10rps”

01 06 60 40 01 0F D7 8A “暂停运行”

## 6. 3 回零模式

在回零模式下，我们将通过三个限位开关进行回零操作，即 CW 限位，CCW 限位，以及原点限位（可以不用）。

### 使能驱动器的操作

在驱动器上电或者复位后，驱动器处于禁止状态。向驱动器控制寄存器写入控制字 0x000F, 使设备进入操作使能状态。

### 使能回零模式

将 0x06 写入对象字典 6060h，使能回零模式。

### 设置运行参数

需要设置回零方式（0x6098h）、回零速度（0x6099h）、以及回零的加减速度（0x609Ah），零位偏移量（0x607C），零位偏移速度（0x609B）等参数。

### 开始回零处理

在对寄存器 0x6098h 中选择回零的方式. 将控制字中的第 4 位翻转，开始执行回零操作。可以通过对状态字查看当前的状态。

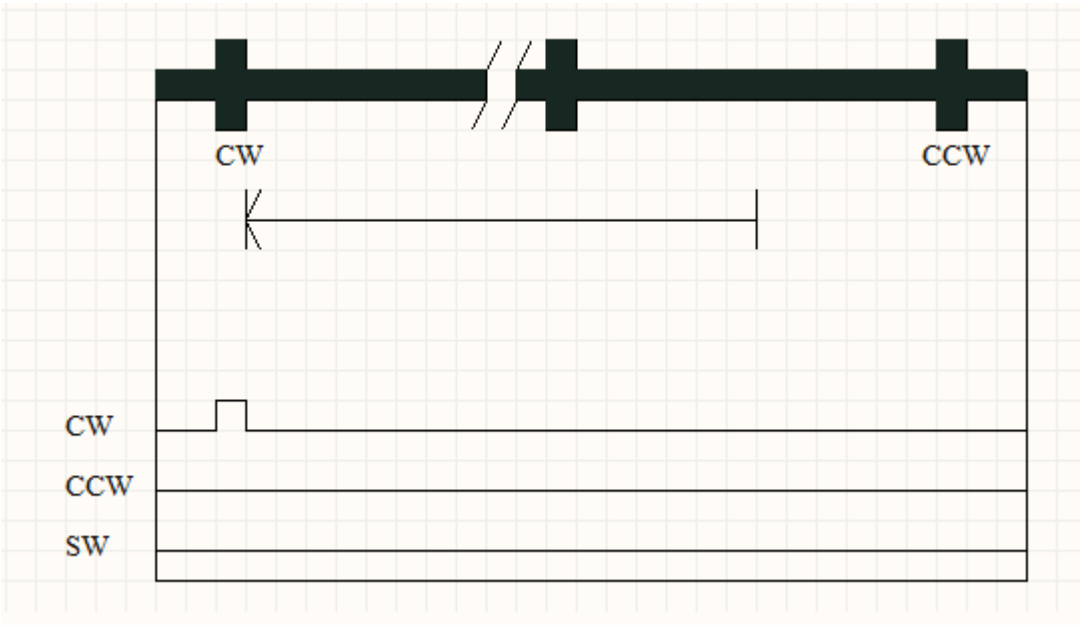
### 零位偏移

零位偏移就是机械原点与我们设置的零点的偏移距离，其偏移的方向可以是机械原点的左边或者右边。

# 回零运行框图

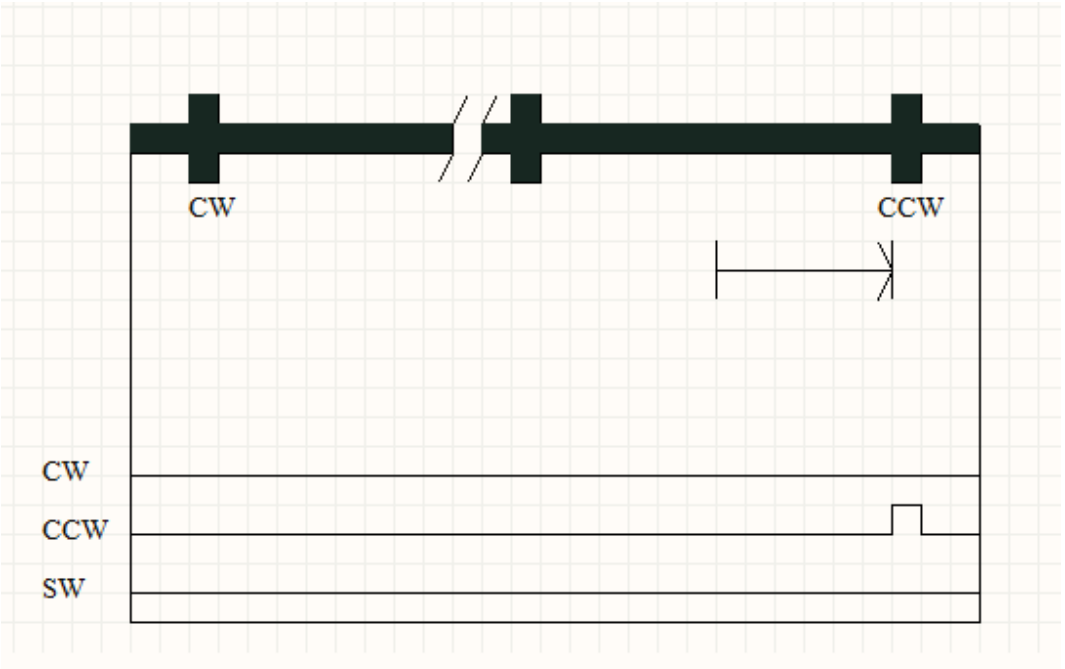
## 回零方式 1

向左运动，回到 CW 限位时停止运动。（回零最高速度为 300r/min）



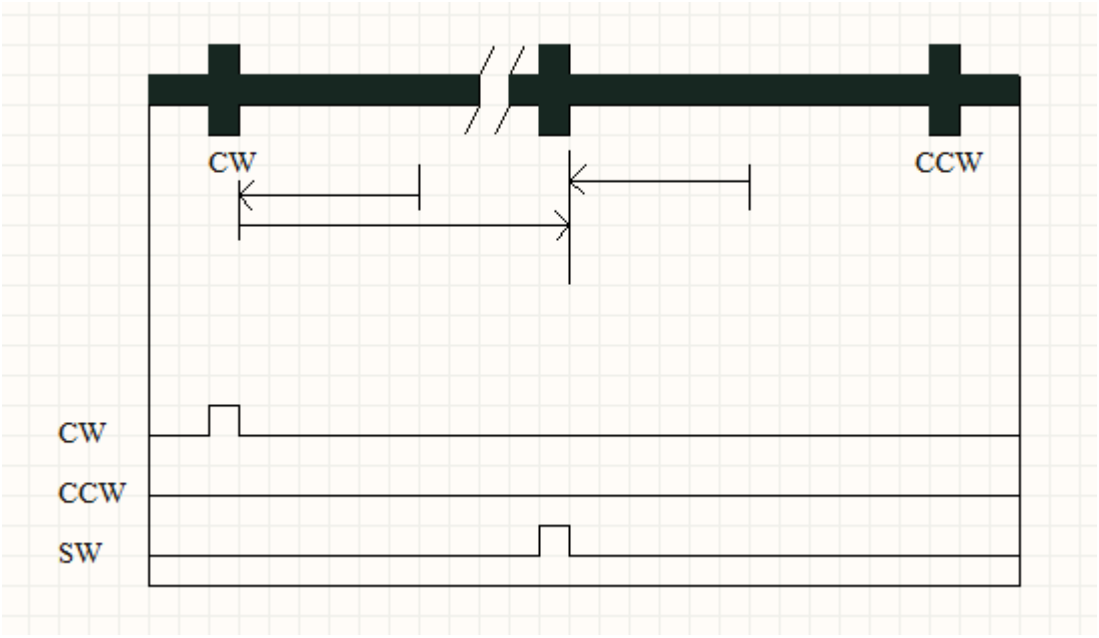

## 回零方式 2

向右运动，回到 CCW 限位时停止运动。（回零最高速度为 300r/min）



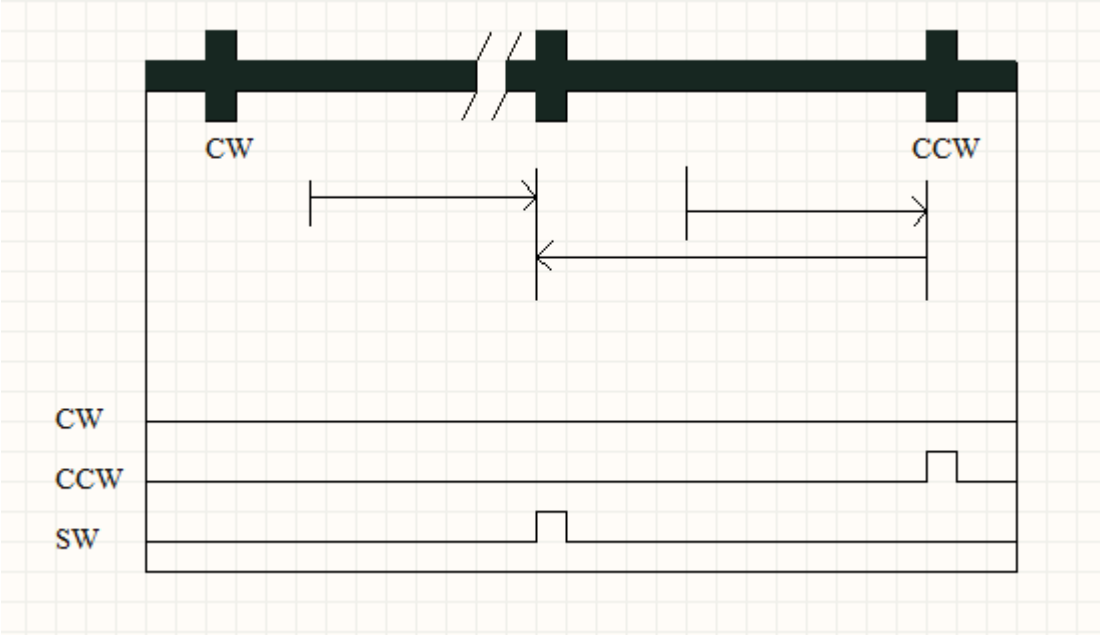
## 回零方式 3

向左运动，若到达 CW 限位，则向右运行（未触碰左限位则继续向左运行），到达机械原点限位时停止运动。（回零最高速度为 300r/min）



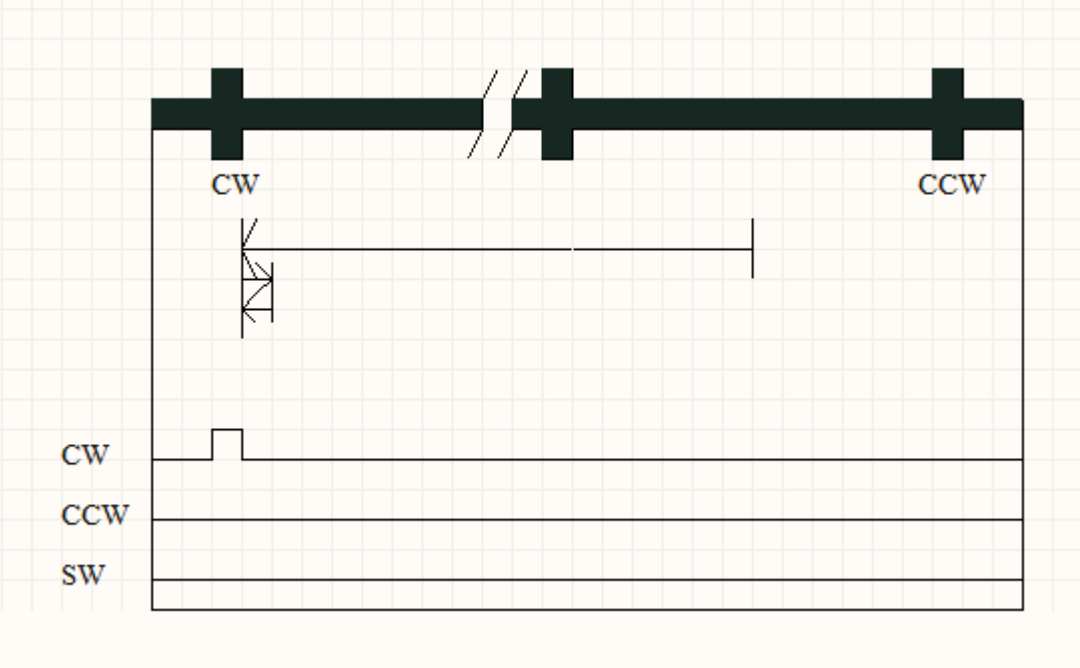

## 回零方式 4

向右运动，若到达 CCW 限位，则向左运行（未触碰右限位则继续向右运行），到达机械原点限位时停止运动。（回零最高速度为 300r/min）



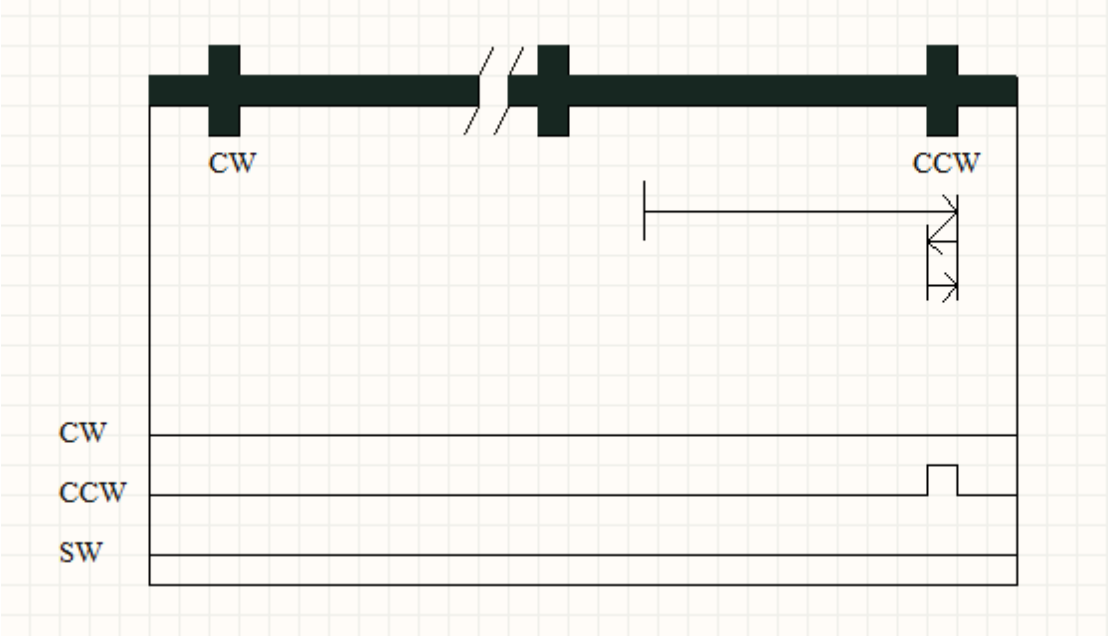
## 回零方式 5

高速向左遇到限位开关后快速降低，然后向右低速移动，过限位开关后低速向左回零。（回零最高速度为 1200r/min）



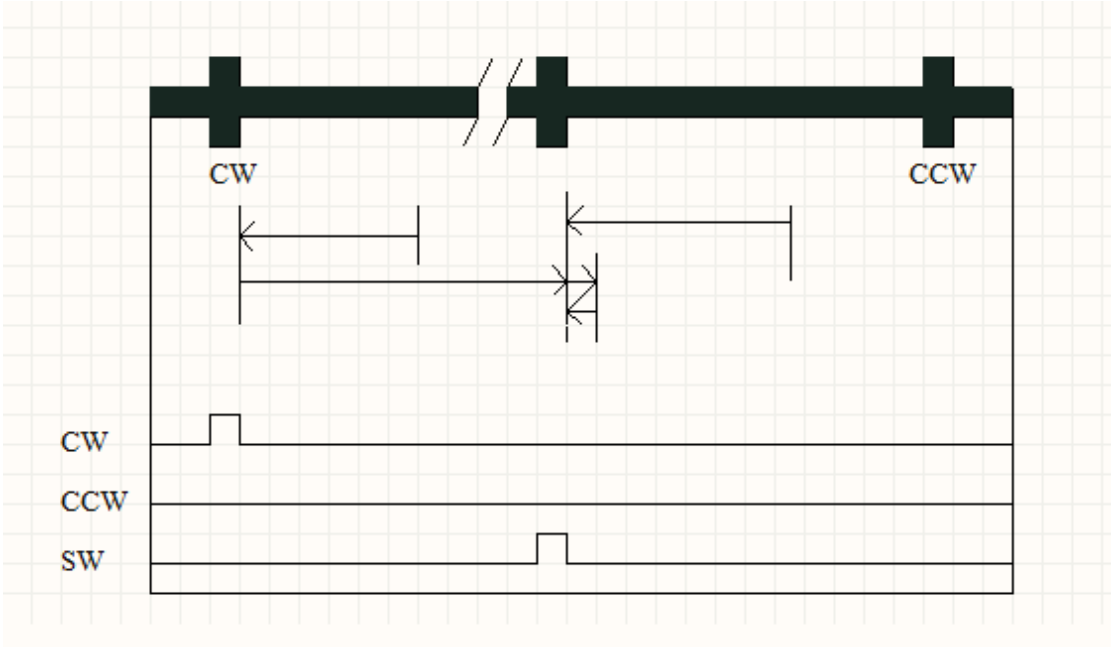

## 回零方式 6

高速向右遇到限位开关后快速降低，然后向左低速移动，过限位开关后低速向右回零。（回零最高速度为 1200r/min）



## 回零方式 7

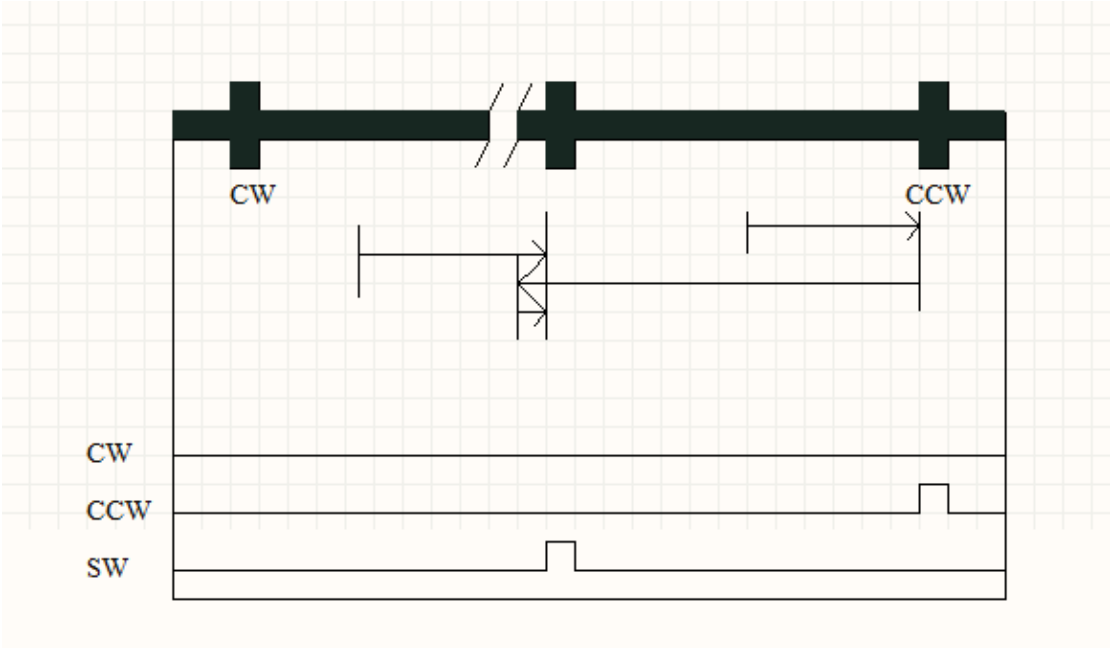
高速向左运行，如遇 CW 限位开关，则转换速度方向向右高速运行（若未遇到机械原点则继续向左运行），遇到机械原点限位开关后快速降低，然后向右低速移动，过限位开关后低速




向左回零。（回零最高速度为 1200r/min）

## 回零方式 8

高速向右运行，如遇 CW 限位开关，则转换速度方向向右高速运行（若未遇到机械原点则继续向左运行），遇到机械原点限位开关后快速降低，然后向右低速移动，过限位开关后低速向左回零。（回零最高速度为 1200r/min）



# 回零模式代码例程（0x6000=0）

*****给电机上电*****

01 06 60 40 00 01 57 DE　“驱动器上电初始化”

01 06 60 40 00 03 D6 1F　“驱动器正常运行，但电机未使能，松开刹车”

01 06 60 40 00 0F D6 1A　“电机给电，处于可运行状态”

*****设置为回零模式*****

01 06 60 60 00 06 17 D6　“设置回零模式”

*****设定回零方式*****

01 06 60 98 00 01 D7 E5　“选择回零方式 1”

*****设定回零参数*****

01 06 60 9A 03 E8 B7 5B　“设回零加速度为 100rps/s ”

01 10 60 99 00 02 04 00 00 00 64 93 40　“设回零速度为 10rps ”

01 10 60 9B 00 02 04 00 00 00 64 12 99　“设零位偏移速度速度为 10rps ”

01 10 60 7C 00 02 04 00 00 03 E8 5C 62　“设零位偏移量为 1000 ”

设定回方式以及设定回零参数的指令可以以 0x10 模式一次性写入：

01 10 60 7C 00 08 10 00 00 03 E8 00 01 00 00 00 64 03 E8 00 00 00 64 09 42

**** 启动/停止运行 ****

01 06 60 40 00 1F D7 D6　“开始运行”

01 06 60 40 01 1F D6 46　“暂停运行”

# 7 设备支持的功能码

## 7.1 简介

杰美康 ModBus-RTU 协议支持以下功能码：

| 功能 | 代码 | 描述 | 标记 |
|---|---|---|---|
| 读保持寄存器 | 0x03 | 读取多个字 | 最大字长度：63 字 |
| 写单个寄存器 | 0x06 | 写入单个字 | - |
| 写多个寄存器 | 0x10 | 写入多个字 | - |

## 7.2 读寄存器

请求

| | | |
|---|---|---|
| 地址码 | 1Byte | 0xXX |
| 功能码 | 1Byte | 0x03 |
| 起始地址 | 2Byte | 0xXXXX |
| 总地址数 | 2Byte | 1~126 |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

响应

| | | |
|---|---|---|
| 地址码 | 1Byte | 0xXX |
| 功能码 | 1Byte | 0x03 |
| 总字节数 | 1Byte | - |
| 返回数据 | N 个字节数据 | - |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

## 示例

请求

| 设备地址 | 功能码 | 起始地址 | 总地址数 | CRC 校验 |
|---|---|---|---|---|
| XX | 0x03 | Hi Lo | Hi Lo | Hi Lo |
| 1Byte | 1Byte | 2Byte | 2Byte | 2Byte |

备注: Hi=高字节，Li=低字节。

响应

| 设备地址 | 功能码 | 总字节数 | 第一个地址 | ..... | 最后一个地址 | CRC 校验 |
|---|---|---|---|---|---|---|
| XX | 0x03 | XX | Hi L0 | | Hi Lo | Hi Lo |
| 1Byte | 1Byte | 1Byte | 2Byte | | 2Byte | 2Byte |

备注: Hi=高字节，Li=低字节。

示例：读取 32 位寄存器（0x607Ah）的内容

若 0x6000 = 0：

读取报文：01 03 60 7A 00 02 FB D2

反馈报文：01 03 04 00 03 0D 40 0F 53

若 0x6000 = 1：

读取报文：01 03 60 7A 00 02 FB D2

反馈报文：01 03 04 0D 40 00 03 B9 4A

读取报文讲解如下：

| 报文： | 01 | 03 | 60 7A | 00 02 | FB D2 |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 总地址数 | 校验码 |

反馈报文讲解如下：

| 报文： | 01 | 03 | 04 | 00 03 | 0D 40 | 0F 53 |
|---|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 总字节数 | 地址 1 数据 | 地址 2 数据 | 校验码 |

## 7.3 写单个寄存器

请求

| 地址 | 1Byte | 0xXX |
|---|---|---|
| 功能码 | 1Byte | 0x06 |
| 寄存器地址 | 2Byte | 0xXXXX |
| 寄存器值 | 2Byte | 0xXXXX |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

响应

| 地址 | 1Byte | 0xXX |
|---|---|---|
| 功能码 | 1Byte | 0x06 |
| 寄存器地址 | 2Byte | 0xXXXX |
| 寄存器值 | 2Byte | 0xXXXX |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

备注：32 位寄存器不能用 06 功能码写，需要用 10 功能码写

示例：往设备 2 的 16 位寄存器（0x6060h）中写入 0x0003。

请求与响应数据：

| 报文： | 02 | 06 | 60 60 | 00 03 | XX XX |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 寄存器地址 | 寄存器数据 | 校验码 |

## 7.4 写多个寄存器

请求

| | | |
|---|---|---|
| 地址 | 1Byte | 0xXX |
| 功能码 | 1Byte | 0x10 |
| 寄存器起始地址 | 2Byte | 0xXXXX |
| 写入地址个数 | 2Byte | 0xXXXX |
| 写入字节数 | 1Byte | 0xXX |
| 写入 n 个数据值 | N*2 Byte | 0xXX |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

响应

| | | |
|---|---|---|
| 地址 | 1Byte | 0xXX |
| 功能码 | 1Byte | 0x10 |
| 寄存器起始地址 | 2Byte | 0xXXXX |
| 写入地址个数 | 2Byte | 0xXXXX |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

示例：

往 16 位寄存器目标加速度（6083h）以及目标减速度（6084h）中写入 0x0064。

若 0x6000 = 0：

写入报文：01 10 60 83 00 02 04 00 64 00 64 53 EC

反馈报文：01 10 60 83 00 04 AE 20

若 0x6000 =1：

写入报文：01 10 60 83 00 02 04 00 64 00 64 53 EC

反馈报文：01 10 60 83 00 04 AE 20

写入报文讲解如下：

| 报文： | 01 | 10 | 60 83 | 00 02 | 04 | 00 64 | 00 64 | 53 EC |
|---|---|---|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 总地址个数 | 总的字节数 | 写入 0x6083 数据 | 写入 0x6084 数据 | CRC 校验 |

反馈报文讲解如下：

| 报文： | 01 | 10 | 60 83 | 00 02 | 53 EC |
|---|---|---|---|---|---|
| 说明： | 地址 | 功能码 | 起始寄存器地址 | 写入总地址个数 | CRC 校验 |

## 7.5 异常功能码

响应

| | | |
|---|---|---|
| 地址 | 1Byte | 0xXX |
| 功能码 | 1Byte | 请求功能码\|0x80 |
| 异常代码 | 1Byte | 0~11 |
| CRC 校验代码“低字节” | 1Byte | 0xXX |
| CRC 校验代码“高字节” | 1Byte | 0xXX |

示例：

往设备 1 的 16 位寄存器（0x6090h）中写入 0x0003。发生异常，返回无该寄存器。

写入报文：01 06 60 90 00 03 D7 E6

返回报文：01 06 0B 03 A7

写入报文讲解如下：

| 报文： | 01 | 06 | 60 90 | 00 03 | D7 E6 |
|---|---|---|---|---|---|
| 说名： | 地址 | 功能码 | 寄存器地址 | 写入数据 | CRC 校验 |

反馈报文讲解如下：

| 报文： | 01 | 86 | 0B | 03 A7 |
|---|---|---|---|---|
| 说明： | 地址 | 异常功能码 | 写入寄存器不存在 | CRC 校验 |

# 8 寄存器列表

| 寄存器 | 访问权限 | 数据类型 | 说明 |
|---|---|---|---|
| 0x 1001 | RO | UNSIGNED16 | 内部错误将会映射到该寄存器 |
| 0x 1008 | RO | Via-String | 设备厂商及型号 |
| 0x 1009 | RO | Via-String | 设备硬件版本号 |
| 0x 100A | RO | Via-String | 设备软件版本号 |
| 0x 6000 | RW | UNSIGNED16 | 32 位数　据大小端格式寄存器 |
| 0x 605A | RW | INTEGER16 | 快速停止命令时停止的方式 |
| 0x 6040 | WO | UNSIGNED16 | 驱动器的状态和运动的控制字 |
| 0x 6060 | WO | INTEGER16 | 选择相应的操作模式 |
| 0x 6081 | RW | INTEGER32 | 匀速运行的速度 |
| 0x 6083 | RW | INTEGER16 | 电机运行的加速度 |
| 0x 6084 | RW | INTEGER16 | 电机运行的减速度 |
| 0x 6085 | RW | UNSIGNED16 | 快速停止减速度 |
| 0x 607A | RW | INTEGER32 | 位置模式驱动器移动的位置 |
| 0x 607C | RW | INTEGER32 | 零点位置与机械原点的偏移位置 |
| 0x 6098 | RW | INTEGER16 | 回零方式 |
| 0x 6099 | RW | UNSIGNED32 | 回机械原点速度 |
| 0x 609A | RW | UNSIGNED16 | 回零点加/减速度 |
| 0x 609B | RW | UNSIGNED32 | 回零点速度 |
| 0x 6041 | RO | UNSIGNED16 | 反应当前驱动器的状态 |
| 0x 6061 | RO | INTEGER16 | 当前的操作模式 |
| 0x 6064 | RO | INTEGER32 | 位置模式下当前时刻的位置 |
| 0x 606C | RO | INTEGER32 | 当前时刻的速度 |

TIP:如需使用 0x10 模式连续写入寄存器，写入的寄存器必须在上表中是连续的，如：0x6041,0x6061,0x6064 三个寄存器在上表中是连续的，所以可以用 0x10 模式一次性给这三个寄存器写入参数。

# 附录 1 CRC 校验

循环冗余校验 CRC 区为 2 字节，含一个 16 位二进制数据。由发送设备计算 CRC 值，并把计算值附在信息中，接收设备在接收信息时，重新计算 CRC 值，并把计算值与接收的在 CRC 区中实际值进行比较，若两者不相同，则产生一个错误。

CRC 开始时先把寄存器的 16 位全部置成“1”，然后把相邻 2 个 8 位字节的数据放入当前寄存器中，只有每个字符的 8 位数据用作产生 CRC，起始位，停止位和奇偶校验位不加到 CRC 中。
产生 CRC 期间，每 8 位数据与寄存器中值进行异或运算，其结果向右移一位(向 LSB 方向)，并用“0”填入 MSB，检测 LSB，若 LSB 为“1”则与预置的固定值异或，若 LSB 为“0”则不作异或运算。

重复上述处过程，直至移位 8 次，完成第 8 次移位后，下一个 8 位数据，与该寄存器的当前值异或，在所有信息处理完后，寄存中的最终值为 CRC 值。

产生 CRC 的过程：
1. 把 16 位 CRC 寄存器置成 FFFFH。
2. 第一个 8 位数据与 CRC 寄存器低 8 位进行异或运算，把结果放入 CRC 寄存器。
3. CRC 寄存器向右移一位，MSB 填零，检查 LSB。
4. (若 LSB 为 0)：重复 3，再右移一位。
   (若 LSB 为 1)：CRC 寄存器与 A001H 进行异或运算
5. 重复 3 和 4 直至完成 8 次移位，完成 8 位字节的处理。
6. 重复 2 至 5 步，处理下一个 8 位数据，直至全部字节处理完毕。
7. CRC 寄存器的最终值为 CRC 值。
8. 把 CRC 值放入信息时，高 8 位和低 8 位应分开放置。把 CRC 值放入信息中
发送信息中的 16 位 CRC 值时，先送低 8 位，后送高 8 位。
若 CRC 值为 1241(0001 0010 0100 0001)：

| 地址 | 功能码 | 数据个数 | 数据 1 | 数据 2 | 数据 3 | 数据 4 | 数据 5 | CRC 低位 | CRC 高位 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 41 | 12 |

例：
各种可能的 CRC 值，按两列装入，一列在 16 位 CRC 的高 8 位区，为(0-256 的)CRC 值
另一类为低 8 位区，为 CRC 的低位值。
用这种方法得到的 CRC 其执行速度快于计算缓冲器中每个新字符得到一个新 CRC 值的方法。
**注意：该函数内部交换 CRC 中的高/低字节，返回的 CRC 值 中，其字节已交换。**
因此，由功能码返回的 CRC 值，能直接放在信息中传送。

**例程：**

该函数的返回值 CRCC 为”unsigned short” 类型。

生成 CRC 的函数：

unsigned char *PMSG; 为生成 CRC 值，把指针指向含有二进制 的数据的缓冲器

unsigned short DataLen; 缓冲器中的字节数。

```
unsigned short CRC16(PMSG, DataLen)
{
    unsigned char uchCRCHi = 0xFF ; /*  初始化高字节*/
    unsigned char uchCRCLo = 0xFF ; /*  初始化低字节*/
    unsigned uIndex ; /*把 CRC 表*/
    while (DataLen—) /*通过数据缓冲器*/
    {
        uIndex = uchCRCHi ^ *PMSG++ ; /*计算 CRC*/
        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
        uchCRCLo = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

# 附录 2 Modbus/RTU16 位 CRC 校验例程

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Modbus
{
    public static class Utility
    {
        private static readonly ushort[] m_CrcTable =
        {
            0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
            0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
            0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
            0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
            0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
            0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
            0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
            0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
            0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
            0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
            0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
            0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
            0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
            0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
            0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
            0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
            0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
            0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
            0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
            0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
            0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
            0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
            0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
            0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
            0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
            0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
            0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
            0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
            0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
            0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
            0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
```

```
            0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
        };
        //计算纵向冗余校验
        //参数：”data”  用于 LRC 运算
        //返回值：LRC 计算结果
        public static byte CalculateLrc(byte[] data)
        {
            if (data == null)
            {
                throw new ArgumentNullException(“data”);
            }
            byte lrc = 0;
            foreach (byte b in data)
            {
                lrc += b;
            }
            lrc = (byte)((lrc ^ 0xFF) + 1);
            return lrc;
        }
        //计算周期冗余校验
        //参数：”data”  用于 CRC 运算
        //返回值：CRC 计算结果
        public static byte CalculateLrc(byte[] data)
        {
            if (data == null)
            {
                throw new ArgumentNullException(“data”);
            }
            ushort crc = ushort.MaxValue;
            foreach (byte b in data)
            {
                byte tableIndex = (byte)(crc ^ b);
                crc >>= 8;
                crc ^= m_CrcTable[tableIndex];
            }
        return BitConverter.GetBytes(crc);
        }
    }
}
```

以下是调用方法:

```
byte[] _Data = new byte[] { 0x18, 0x18};
byte[] _Crc = Modbus.Utility.CalculateCrc(_Data);
```

# 联系我们

**地址：深圳市宝安区留仙三路鸿威工业区 A 栋 2 楼**

**电话：0755-26509689　26502268**

**传真：0755-26509289**

**E-mail:info@jmc-motion.com**

**Http: //www.jmc-motion.com**