

汕头市山海创新科技有限公司

无刷电机伺服控制器 用户编程手册

驱控一体，省心！省钱！



目录

1. 用户编程总述	1
2.编程指令说明	1
2.1 赋值指令	2
2.1.1 赋值指令	2
2.1.2 条件赋值指令	2
2.2 跳转指令	2
2.2.1 跳转指令	2
2.2.2 等于跳转指令	3
2.2.3 不等于跳转指令	3
2.2.4 小于跳转指令	4
2.2.5 大于跳转指令	4
2.2.6 小于等于跳转指令	4
2.2.7 大于等于跳转指令	5
2.3 运算指令	5
2.3.1 加指令	5
2.3.2 减指令	5
2.3.3 乘指令	6
2.3.4 除指令	6
2.3.5 取余指令	6
2.3.6 与指令	7
2.3.7 或指令	7
2.3.8 异或指令	7
2.3.9 逻辑与指令	8
2.3.10 逻辑或指令	8
2.4 堆栈指令	8
2.4.1 入栈指令	8
2.4.2 出栈指令	9
2.4.3 读栈指令	9
2.5 中断指令	9
2.5.1 启动中断指令	9
2.5.2 停止中断指令	10
2.5.3 中断返回指令	10
3.编程应用说明	10
3.1 外设检测及控制	10
3.2 电机控制参数设置及其控制	10
3.3 定时寄存器应用	11
3.4 通用寄存器应用	11
3.5 记忆寄存器应用	11
3.6 通讯应用	11
4. 编程软件使用说明	11



4.1 运行环境	11
4.2 软件界面与使用说明	11
4.2.1 菜单栏.....	12
4.2.2 常用功能操作区.....	13
4.2.3 程序文件编辑区.....	14
4.2.4 编译信号显示区.....	14
5. 免责声明	14

修订历史:

更新日期	修订作者	版本	主要修订摘要数
2020-05-31	EIX	V1.0	初版
2021-03-26	eix	V2.0	全新的编程界面

1. 用户编程总述

用户编程原理：用户编程为用户自定义电机运行过程及其控制逻辑的一系列步骤合集，每个步骤为一个指令语句。运行机理：系统为每条步骤指令语句分配一个唯一的地址，第一条指令语句为 1，依次递增，最后一条指令语句为 N，N 为用户程序的总步骤数量。控制器初始化后用户指令语句地址为 1，系统根据用户指令地址执行用户指令，执行完赋值、运算、堆栈、中断启动和中断停止指令后，用户指令语句地址自动加 1；当执行完跳转指令后，用户指令语句地址为跳转指令指定的用户指令地址；当执行条件跳转指令后（包含：<,>==,!=,>=,<=），如果符合条件，则用户指令语句地址为跳转指令指定的用户指令地址，若不符合条件，则用户指令语句地址自动加 1。如果在程序运行过程产生中断，则用户指令语句地址会跳转到中断入口地址执行，直到遇到中断返回指令用户指令语句地址跳转回中断前的地址。当用户指令地址大于 N 时，用户程序自然结束。

指令语句：指令语句由一个指令及其操作数组成，操作数数量根据不同指令数量不同，最少 0 个，最多 3 个。目前支持的指令为赋值指令（赋值，条件赋值），跳转指令（跳转，等于跳转，不等于跳转，小于跳转，大于跳转，小于等于跳转，大于等于跳转），运算指令（加、减、乘、除、取余），逻辑运算指令（与、或、异或、逻辑与、逻辑或），通过这些指令的组合，可以实现赋值、跳转、运算、条件判定、循环执行以及程序的模块化设计和调用。

指令操作对象：分为寄存器、数值和标签三种，寄存器包含通用功能寄存器，通用用户寄存器，外设寄存器，定时寄存器，普通寄存器，记忆寄存器，系统为每个寄存器分配一个地址，操作寄存器其实就是操作其对应的地址；数值为 32 位的符号数，即其范围为 -2,147,483,648~2,147,483,648。通过对寄存器的操作可实现控制器所有功能参数的设置，电机控制，端口的检测和控制，定时控制，数据临时存储，数据记忆保存。在部分指令中操作数可以做逻辑非运算。

每条指令占用 16 字节空间，程序信息占用 1 条指令的空间，控制器支持的总步骤数量=用户程序空间/16 - 1。例如：MSFOCD-5AL 用户程序空间为 8K，支持的总步骤数量=8*1024/16 - 1 = 511（步）。

程序编写及下载：用户程序编辑必须使用“无刷伺服控制器用户编程软件”进行编写，修改，阅读。编写后程序文件可保存在电脑上，其扩展名为*.mspdf。程序文件编译成功后自动生成用户程序代码文件，其扩展名为*.mspd，可在编译软件上直接下载到控制器，也可以通过“无刷电机伺服控制器配置软件”下载到控制器中。

2. 编程指令说明

本章节主要讲述外各种编程指令的应用规范，以及中断处理机制。在所有指令中，逻辑非运算规则为，对常数逻辑非时，如果数据为 0，逻辑非结果为 1，否则结果为 0；对寄存器逻辑非实际上是寄存器所保存数据的逻辑非运算，即寄存器所保存的数据为 0，逻辑非结果为 1，否则结果为 0。下面以 R?，D?，!?, B? 分别代码? 寄存

器的保存数值，常数？，？的逻辑非运算，标签？。

2.1 赋值指令

2.1.1 赋值指令

指令形式：A = B

指令说明：将 B 赋值给 A，A 必须是可写属性的寄存器，B 可以是寄存器也可以是常数。A 和 B 都可以进行逻辑非运算。RB 和 DB 的数值型号为 32 位符合号。

例子：R49 = D2000

指令语句意义：以速度闭环模式启动电机，电机速度锁定为 2000RPM

寄存器说明：R49 为速度闭环目标寄存器

2.1.2 条件赋值指令

指令形式：if C A = B

指令说明：当 C 为真时，将 B 赋值给 A，否则不进行任何操作。A、C 必须是寄存器，B 可以是寄存器也可以是常数。

A、B 和 C 都可以进行逻辑非运算。RB 和 DB 的数值型号为 32 位符号数。

例子：if R257 R49 = D2000

指令语句意义：当 IN1 为高电平的时候，以速度闭环模式启动电机，电机速度锁定为 2000RPM

例子：if !R257 R49 = D2000

指令语句意义：当 IN1 为低电平的时候，以速度闭环模式启动电机，电机速度锁定为 2000RPM

寄存器说明：R49 为速度闭环目标寄存器，R257 为 IN1 端口电平状态寄存器

2.2 跳转指令

2.2.1 跳转指令

指令形式：-> A

指令说明：无条件跳转至 A 步骤，A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间，BA 必须在程序中有定义。利用 A 为寄存器的跳转，可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算。

例子：-> start

指令语句意义：跳转到标签为 start 的指令语句

例子：-> R354

指令语句意义：跳转到普通寄存器 354 所储存数值的指令语句

例子: -> D10

指令语句意义: 跳转到第 10 行指令语句

寄存器说明: R354 为普通寄存器 1

2.2.2 等于跳转指令

指令形式: if B == C -> A

指令说明: 当 B 等于 C 时, 跳转至 A 步骤, A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间, BA 必须在程序中有定义。利用 A 为寄存器的跳转, 可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算, B 和 C 可以逻辑非运算。

例子: if R24 == D0 -> reverse

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到标签为 reverse 的指令语句

例子: if R24 == D0 -> R354

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到普通寄存器 354 所储存数值的指令语句

例子: if R24 == D0 -> D10

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到第 10 行指令语句

例子: if !R24 == D1 -> reverse

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到标签为 reverse 的指令语句

寄存器说明: R24 为正转限位开关 SQ1 的电平寄存器, R354 为普通寄存器 1

2.2.3 不等于跳转指令

指令形式: if B != C -> A

指令说明: 当 B 不等于 C 时, 跳转至 A 步骤, A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间, BA 必须在程序中有定义。利用 A 为寄存器的跳转, 可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算, B 和 C 可以逻辑非运算。

例子: if R24 != D1 -> reverse

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到标签为 reverse 的指令语句

例子: if R24 != D1 -> R354

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到普通寄存器 354 所储存数值的指令语句

例子: if R24 != D1 -> D10

指令语句意义: 当正向限位开关 SQ1 低电平的时候跳转到第 10 行指令语句

例子: if !R24 != D0 -> reverse

指令语句意义：当正向限位开关 SQ1 低电平的时候跳转到标签为 **reverse** 的指令语句

寄存器说明：R24 为正转限位开关 SQ1 的电平寄存器，R354 为普通寄存器 1

2.2.4 小于跳转指令

指令形式：if B < C -> A

指令说明：当 B 小于 C 时，跳转至 A 步骤，A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间，BA 必须在程序中有定义。利用 A 为寄存器的跳转，可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算，B 和 C 可以逻辑非运算。

例子：if R22 < R23 -> stop

指令语句意义：当 IN2 的电压小于 IN3 的电压的时候跳转到标签为 **stop** 的指令语句

例子：if R22 < D2000 -> R355

指令语句意义：当 IN2 的电压小于 2000mV 时候跳转到普通寄存器 355 所储存数值的指令语句

寄存器说明：R22 为 IN2 端口电压采样值寄存器，R355 为普通寄存器 2

2.2.5 大于跳转指令

指令形式：if B > C -> A

指令说明：当 B 大于 C 时，跳转至 A 步骤，A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间，BA 必须在程序中有定义。利用 A 为寄存器的跳转，可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算，B 和 C 可以逻辑非运算。

例子：if R22 > R23 -> stop

指令语句意义：当 IN2 的电压大于 IN3 的电压的时候跳转到标签为 **stop** 的指令语句

例子：if R22 > D2000 -> R355

指令语句意义：当 IN2 的电压大于 2000mV 时候跳转到普通寄存器 355 所储存数值的指令语句

寄存器说明：R22 为 IN2 端口电压采样值寄存器，R355 为普通寄存器 2

2.2.6 小于等于跳转指令

指令形式：if B <= C -> A

指令说明：当 B 小于等于 C 时，跳转至 A 步骤，A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间，BA 必须在程序中有定义。利用 A 为寄存器的跳转，可以实现不同情况跳转执行不同功能段代码。A 不可逻辑非运算，B 和 C 可以逻辑非运算。

例子：if R22 >= R23 -> stop

指令语句意义：当 IN2 的电压大于等于 IN3 的电压的时候跳转到标签为 **stop** 的指令语句

例子: if R22 >= D2000 -> R355

指令语句意义: 当 IN2 的电压大于等于 2000mV 时候跳转到普通寄存器 355 所储存数值的指令语句

寄存器说明: R22 为 IN2 端口电压采样值寄存器, R355 为普通寄存器 2

2.2.7 大于等于跳转指令

指令形式: if B >= C -> A

指令说明: 当 B 大于等于 C 时, 跳转至 A 步骤, A 可以是标签、寄存器也可以是常数。RA 或者 DA 值必须在 1 到总步骤数量 N 之间, BA 必须在程序中有定义。利用 A 为寄存器的跳转, 可以实现不同情况跳转执行不同功能段代码。

A 不可逻辑非运算, B 和 C 可以逻辑非运算。

例子: if R22 <= R23 -> stop

指令语句意义: 当 IN2 的电压小于等于 IN3 的电压的时候跳转到标签为 stop 的指令语句

例子: if R22 <= D2000 -> R355

指令语句意义: 当 IN2 的电压小于等于 2000mV 时候跳转到普通寄存器 355 所储存数值的指令语句

寄存器说明: R22 为 IN2 端口电压采样值寄存器, R355 为普通寄存器 2

2.3 运算指令

2.3.1 加指令

指令形式: A = B + C

指令说明: B 加上 C 的结果赋值给 A, A 必须是寄存器, B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子: R354 = R355 + R356

指令语句意义: 普通寄存器 2 的值加上普通寄存器 3 的值, 结果存在普通寄存器 1

例子: R354 = R354 + D1

指令语句意义: 普通寄存器 1 自增 1

寄存器说明: R354 为普通寄存器 1, R355 为普通寄存器 2, R356 为普通寄存器 3

2.3.2 减指令

指令形式: A = B - C

指令说明: B 减去 C 的结果赋值给 A, A 必须是寄存器, B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子: R354 = R22 - D1650

指令语句意义: IN2 的电压减去 1650mV, 差值保存在普通寄存器 1 中

例子: $R354 = D0 - R355$

指令语句意义: 普通寄存器 2 取反, 结果保存在普通寄存器 1 中

寄存器说明: R354 为普通寄存器 1, R355 为普通寄存器 2

2.3.3 乘指令

指令形式: $A = B * C$

指令说明: B 乘以 C 的结果赋值给 A, A 必须是寄存器, B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子: $R354 = R355 * D10$

指令语句意义: 普通寄存器 2 的值乘以 10, 结果保存在普通寄存器 1 中

例子: $R354 = R355 * R356$

指令语句意义: 普通寄存器 2 的值乘以普通寄存器 3 的值, 结果保存在普通寄存器 1 中

寄存器说明: R354 为普通寄存器 1, R355 为普通寄存器 2, R356 为普通寄存器 3

2.3.4 除指令

指令形式: $A = B / C$

指令说明: B 除以 C 的结果赋值给 A, A 必须是寄存器, B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数, 注意 RC 和 DC 不能为 0。A、B 和 C 都可以进行逻辑非运算。

例子: $R354 = R355 / D10$

指令语句意义: 普通寄存器 2 的值除以 10, 结果保存在普通寄存器 1 中

例子: $R354 = R355 / R356$

指令语句意义: 普通寄存器 2 的值除以普通寄存器 3 的值, 结果保存在普通寄存器 1 中

寄存器说明: R354 为普通寄存器 1, R355 为普通寄存器 2, R356 为普通寄存器 3

2.3.5 取余指令

指令形式: $A = B \% C$

指令说明: B 除以 C 的余数赋值给 A, A 必须是寄存器, B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数, 注意 RC 和 DC 不能为 0。A、B 和 C 都可以进行逻辑非运算。

例子: $R354 = R355 / D10$

指令语句意义: 普通寄存器 2 的值除以 10 的余数保存在普通寄存器 1 中

例子: $R354 = R355 / R356$

指令语句意义: 普通寄存器 2 的值除以普通寄存器 3 的值后的余数保存在普通寄存器 1 中

寄存器说明：R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.3.6 与指令

指令形式：A = B & C

指令说明：B 与上 C 的结果赋值给 A，A 必须是寄存器，B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子：R354 = R257 & R258

指令语句意义：IN2 电平状态与 IN3 电平状态与运算，结果保存在普通寄存器 1 中

例子：R354 = R355 & R356

指令语句意义：普通寄存器 2 的值与普通寄存器 3 的值与运算，结果保存在普通寄存器 1 中

寄存器说明：R257 为 IN1 电平状态寄存器，R258 为 IN2 电平状态寄存器，R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.3.7 或指令

指令形式：A = B | C

指令说明：B 或上 C 的结果赋值给 A，A 必须是寄存器，B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子：R354 = R257 | R258

指令语句意义：IN2 电平状态与 IN3 电平状态或运算，结果保存在普通寄存器 1 中

例子：R354 = R355 | R356

指令语句意义：普通寄存器 2 的值与普通寄存器 3 的值或运算，结果保存在普通寄存器 1 中

寄存器说明：R257 为 IN1 电平状态寄存器，R258 为 IN2 电平状态寄存器，R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.3.8 异或指令

指令形式：A = B ^ C

指令说明：B 异或上 C 的结果赋值给 A，A 必须是寄存器，B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子：R354 = R257 ^ R258

指令语句意义：IN2 电平状态与 IN3 电平状态异或运算，结果保存在普通寄存器 1 中

例子：R354 = R355 ^ R356

指令语句意义：普通寄存器 2 的值与普通寄存器 3 的值异或运算，结果保存在普通寄存器 1 中



寄存器说明：R257 为 IN1 电平状态寄存器，R258 为 IN2 电平状态寄存器，R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.3.9 逻辑与指令

指令形式：A = B && C

指令说明：B 逻辑与上 C 的和赋值给 A，A 必须是寄存器，B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子：R354 = R257 && R258

指令语句意义：IN2 电平状态与 IN3 电平状态逻辑与运算，结果保存在普通寄存器 1 中

例子：R354 = R355 && R356

指令语句意义：普通寄存器 2 的值与普通寄存器 3 的值逻辑与运算，结果保存在普通寄存器 1 中

寄存器说明：R257 为 IN1 电平状态寄存器，R258 为 IN2 电平状态寄存器，R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.3.10 逻辑或指令

指令形式：A = B || C

指令说明：B 逻辑或上 C 的和赋值给 A，A 必须是寄存器，B 和 C 可以是寄存器也可以是常数。RB、DB、RC 和 DC 的数值型号为 32 位符号数。A、B 和 C 都可以进行逻辑非运算。

例子：R354 = R257 || R258

指令语句意义：IN2 电平状态与 IN3 电平状态逻辑或运算，结果保存在普通寄存器 1 中

例子：R354 = R355 || R356

指令语句意义：普通寄存器 2 的值与普通寄存器 3 的值逻辑或运算，结果保存在普通寄存器 1 中

寄存器说明：R257 为 IN1 电平状态寄存器，R258 为 IN2 电平状态寄存器，R354 为普通寄存器 1，R355 为普通寄存器 2，R356 为普通寄存器 3

2.4 堆栈指令

用户堆栈深度 8，数据型号为 32 符号数，最后入栈的数据序号为 0，越先入栈的数据序号越大，先进的数据先出来，如果入栈数据超过 8 个，先入栈的数据丢弃。

2.4.1 入栈指令

指令形式：PUSH A

指令说明：将 A 推入栈，A 可以是寄存器也可以是常数。RA 和 DA 的数值型号为 32 位符号数。A 可以进行逻辑非运

算。

例子：PUSH R354

指令语句意义：普通寄存器 1 的值入栈

寄存器说明：R354 为普通寄存器 1

2.4.2 出栈指令

指令形式：POP A

指令说明：将出栈数据赋值给 A，A 必须是寄存器。A 可以进行逻辑非运算。

例子：POP R354

指令语句意义：堆栈值推出到普通寄存器 1

寄存器说明：R354 为普通寄存器 1

2.4.3 读栈指令

指令形式：A = STACK B

指令说明：将堆栈中序号为 B 的数据赋值给 A，A 必须是寄存器，B 可以是寄存器也可以是常数。RB 和 DB 的数值型号为 0~7。A 可以逻辑非运算，B 不可以逻辑非运算。

例子：POP R354

指令语句意义：堆栈值推出到普通寄存器 1

寄存器说明：R354 为普通寄存器 1

2.5 中断指令

可编程系统提供了系统异常中断、位置完成中断、时间定时到零中断，外设输入中断。中断优先级分为 1 和 2 两级，2 比 1 级别高，低优先级中断会被更高优先级中断所中断，即中断嵌套。如果新产生的中断优先级比当前的中断优先级低或者相等，后产生的中断将排队在先产生的中断后面，即先中断的任务执行完成后再执行后产生的中断任务，系统最多保存三个中断任务。如果新中断产生时，已经有三个中断未处理完，则不管当前中断的优先级，一律不响应。系统使用中断向量地址来代表不同的中断，具体定义详见附录 5.3。

2.5.1 启动中断指令

指令形式：IT A C -> B

指令说明：启动中断向量地址 A 的中断，中断处理任务地址为 B，中断的优先级为 C，A、C 必须是常数，B 可以是寄存器也可以是常数。DA 取值范围 0~12，DC 必须是 1 或者 2，RB 或者 DB 必须在 1 到总步骤数量 N 之间。A、B、C 都不可逻辑非运算。

例子: IT D1 D2 → Pdone

指令语句意义: 启动位置完成中断, 中断级别为 2, 中断处理地址为标签 Pdone 指令行

寄存器说明: D1 中断向量 1 为位置完成中断向量, D2 为中断优先级 2

2.5.2 停止中断指令

指令形式: STOP IT A

指令说明: 停止中断向量地址 A 的中断 A 必须是常数。DA 取值范围 0~12, A 不可逻辑非运算。

例子: STOP IT D1

指令语句意义: 停止位置完成中断

寄存器说明: D1 中断向量 1 为位置完成中断向量

2.5.3 中断返回指令

指令形式: RETI

指令说明: 中断返回, 用户指令地址返回中断前的地址。

3. 编程应用说明

本章节主要讲述外设检测及控制编程方法, 电机控制相关参数设置及电机控制编程方法, 定时寄存器应用及编程方法, 通用寄存器及记忆寄存器的应用及编程方法。下面以 MSFOCD-5AL 产品为例分项进行介绍。

3.1 外设检测及控制

外设检测及控制主要操作对象为外设寄存器及部分通用功能寄存器。包含模拟量 AD 采样值, IO 端口电平状态, PWM/频率/脉冲检测值, 端口功能设置寄存器。寄存器的读写属性由端口状态及寄存器功能确定, 如 AD 采样值寄存为只读, VO 端口开关量寄存器可读可写。可通过赋值指令检测外设端口状态, 设置输出状态, 利用条件指令进行判定跳转, 利用运算指令进行运算。

3.2 电机控制参数设置及其控制

电机控制参数设置及其控制主要操作对象为通用功能寄存器。包含实时状态寄存器, 电机控制寄存器, 系统参数配置寄存器, PI 参数配置寄存器。如要让电机以 1000 转速度反转, 只需要给速度闭环目标寄存器赋值-1000; 若要让电机向前移动 800 个脉冲位置, 只要将位置控制方式寄存器赋值为 1, 位置闭环控制目标位置寄存器赋值为 800; 若要让电机急停只需向电机停止寄存器赋值为 1。

3.3 定时寄存器应用

系统提供了 3 个 10 毫秒定时器和 2 个 1 秒定时器，计时单位分别为 10 毫秒和 1 秒，定时误差分别不大于 10 毫秒和 1 秒，最长定时时间达 68 年。当定时器对应的寄存器不为零时，定时器在每个计时单位时间会自动递减相应的寄存器值直到值为零。例如当要完成一个 2 秒定时，给 10 毫秒定时器 1 寄存器赋值 200，然后读取该寄存器的值，当其值为 0 时表示 2 秒时间到。

3.4 通用寄存器应用

系统提供了 16 个 32 位有符号整数通用寄存器，掉电数据清零，用于存放临时数据和中间数据，无需定义和申明，直接使用。如果要用到小数数据，可先将数据放大后再保存。

3.5 记忆寄存器应用

系统提供了 4 个 32 位有符号整数记忆寄存器，掉电后数据会保持，主要应用于需要长期保存或者统计的数据，如生产量统计，当前位置保存等。**注意：记忆寄存器写的时候必须在电机停止状态，否则会保存失败。**

3.6 通讯应用

在系统处理内置程序运行模式时，485 通讯只能配置控制参数，不能控制电机运行，此时可利用通用功能寄存器中的用户通用寄存器的读写实现通讯控制和状态反馈。例如，如果需要 485 通讯启动电机，可以将通用寄存器置为启动标志，用户程序查询到启动标志后运行启动电机的指令即可。

4. 编程软件使用说明

4.1 运行环境

运行环境：WinXP/Vista/Win7/Win10

4.2 软件界面与使用说明

编辑软件主页面如下图所示，分为菜单栏，程序编辑区，编译信息区，常用功能操作区。

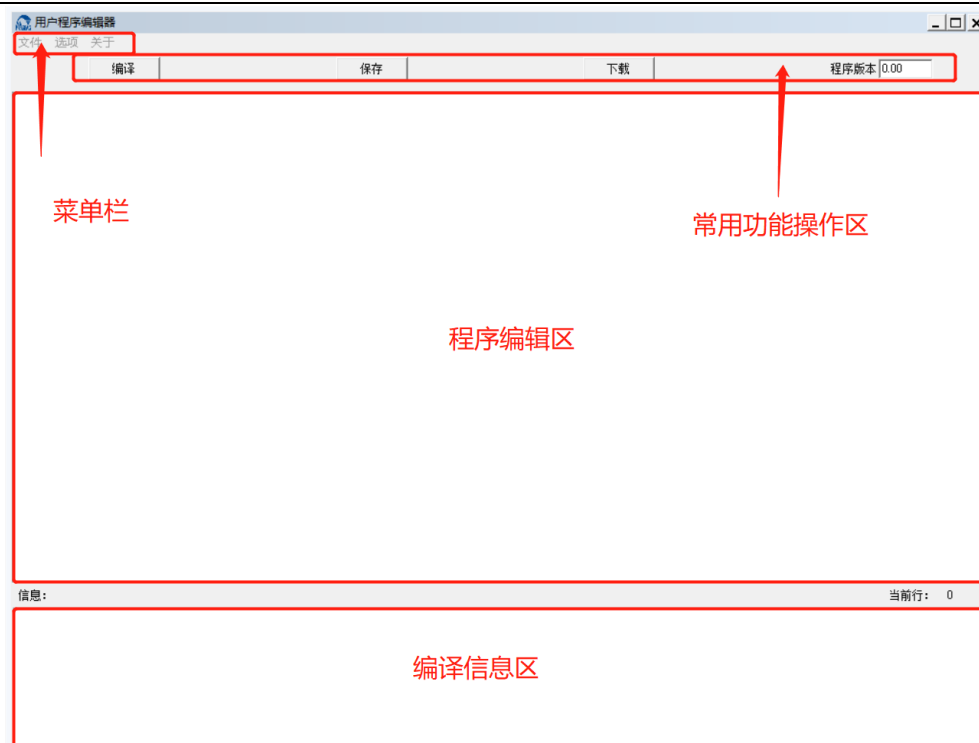


图 4.1 编程界面

4.2.1 菜单栏

菜单栏界面如图 4.2 所示。包含【文件】、【选项】和【关于】。



图 4.2 首行菜单栏

【文件】下拉菜单出现加载过程、保存过程和新过程界面如4.3所示。

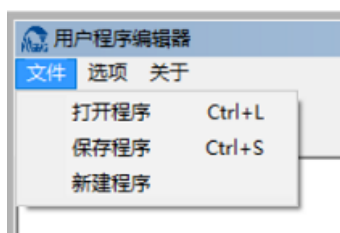


图 4.3 文件下拉菜单

打开程序：当需要阅读，编辑修改已有程序文件，可通过打开程序打开（程序文件扩展名为*.mispdf），快捷键Ctrl+L。

保存程序：当过程文件需要保存备份时，可通过保存程序保存在电脑上（程序文件扩展名为*.mispdf）。快捷键Ctrl+S。

新建过程：新建过程将会清除当前的程序文件，新建前保存好已有程序文件。

【选项】界面如图 4.4 所示，设置可进行程序代码下载设置及中英文界面切换。

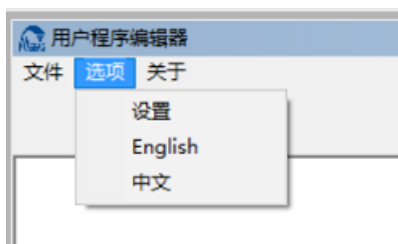


图 4.4 选项菜单

在选项-设置中设置程序代码下载的串口号，定义程序代码文件名称，读取控制器中用户程序代码及程序步骤数。



图 4.5 选项设置界面

【关于】界面如图 4.6 所示，单击出现软件版本信息和产品相关联系方式。



图 4.6 关于界面

4.2.2 常用功能操作区

操作功能包括：程序文件编译，程序文件保存，程序代码下载，程序版本号

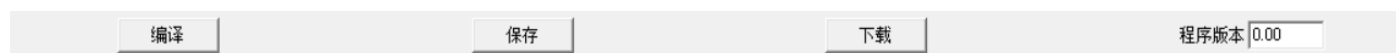


图 4.7 常用功能操作区

编译：对程序文件区域的程序文件进行编译，编译成功会自动生成程序代码文件（扩展名*.mspd）。编译信息在下面的编译信息区域显示。

保存：保存程序文件区域中的程序文件（扩展名*.mspd）。

下载：将编译生成的程序代码文件下载通过串口下载到控制器中，下载过程会弹出下载进度窗口，下载成功会有提示。

程序版本：编译的时候会将程序版本框在的版本号加载到程序代码文件中，格式为x.xx，x必须是0~9的数字。

4.2.3 程序文件编辑区

程序文件编辑区为一个纯文本的编程器，可对程序文件自由编辑，包含程序录入，删除，复制，剪切。程序文件必须遵守下面的规则：

- 1、每行只能写一条指令。
- 2、不能出现空行。每一行必须有一条合法的程序语句。
- 3、标签定义必须在每一行的最前面，用“:”结束，标签中不能包含空格，第一个程序文件中不能在不同行定义相同的标签。
- 4、每个操作命令或者操作数彼此之间必须用空格分开。
- 5、语句注释必须放在指令语句最后面，以“//”开始，不能跨行注释。

注：在程序文件编辑区右下角会显示当前编程区域光标所在的行号。

4.2.4 编译信号显示区

编译信息区显示编译的结果，如果成功则显示编译成功。如果程序文件编译有错，具体某一行错误，错误类型都会显示，每行指标一个错误点，点击提示错误信息，程序编辑器会对程序文件编辑区域中对应的错误指令行选中标示，方便找到错误点进行修改。

5. 免责声明

本文档提供相关产品的使用说明。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。并且，本产品的销售和 / 或使用我们不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。电机驱动模块及相关软件为商业级产品，本产品并非设计用于医疗、救生或维生等用途。我们可能随时对产品规格及产品描述做出修改，恕不另行通知。