





目录

简介	6
开始使用Q驱动器	6
独立运行（程序驻留）或主机控制	7
单轴或多周主机控制	7
绪论	7
Q的含义	7
如何将指令放入队列	8
Q程序	8
命令	9
指令结构	10
缓存器和立即指令	10
缓存器指令类别	11
运行指令	11
伺服指令	11
配置指令	11
I/O指令	11
通讯指令	11
Q程序指令	11
寄存器指令	11
缓存器指令表	11
运行指令	11
伺服指令	12
配置指令	12
I/O指令	13
通讯指令	13
Q程序指令	13
寄存器指令	14
立即指令集	15
数据寄存器	16
只读寄存器	16
读/写寄存器	17
用户定义寄存器	17
非易失性数据存储单元	18
访问数据寄存器	18
下载	18
上传	18
写	18
读	18
数据寄存器操作	18

移位	18
增加/减少	19
计数	19
数学&逻辑	19
状态检测	19
主机串行连接	20
用RS-232 连接到PC	20
使用RS-422/485 可选模块连接到主机	20
用RS-232到RS-422转换器	21
用RS-232到RS-485转换器	21
USB到RS-485转换	21
连接编码器（需要可选的编码器反馈模块）	22
主机串行通讯协议	22
PR指令细节	22
Bit0-默认值（标准SCL）	22
Bit1-总是使用地址符	22
Bit2-Ack/Nak	23
Bit3-校验位	24
Bit4-RS-485 adaptor（MSST-S中不用）	24
Bit5-3位数字代码寄存器地址	24
PR指令举例	24
报警和状态代码	25
报警代码定义	25
状态代码定义	26
将报警和状态代码转化为2进制的工具	27
程序	27
开始编程	27
Q程序软件	28
段编辑	29
编写段和程序文件：.qsg和.qpr	29
开始键	29
串行通讯口	30
自动检测	30
波特率	31
驱动器信息和状态显示窗口	31
主机命令行	31
数据操作	32

驱动器 I/O 状态	32
指令序列示例	32
运行相对距离	32
运行绝对距离	33
运行到传感器信号有效	33
循环	34
跳转	34
调用	35
多任务处理	36

简介

感谢您购买鸣志公司的产品。我们期待我们的产品在性能、价格以及易于使用等方面为您的应用系统带来最佳的选择。

Q-Programmer 软件应用于鸣志驱动器产品，包括：MSST5-Q，MSST10-Q，MSST5-I，MSST10-I，MSSTAC6-Q，MSSTAC6-I以及STM23Q。本使用手册讲解怎样安装并使用 Q Programmer Windows 软件。

要查看关于Q系列驱动器的详细信息，请阅读产品附带的硬件使用手册。在随机附带的CD及我们的网站(<http://www.moons.com.cn>)上也有相应的硬件手册下载。

Q驱动器有如下功能：

- 程序驻留和串行主机命令控制。
- 数据寄存器
- 多任务处理
- 数学&逻辑功能
- RS-485通讯实现多个驱动器控制
- 可靠的通讯协议
- 模拟量输入
- 速度模式，点动
- 数据寄存器

Q Programmer有如下特性：

- 图形化编程界面
- 鼠标点击编程
- 即时数据寄存器监视
- 自动持续检测驱动器状态
- 主命令行用于输入指令
- 易于编程

开始使用Q驱动器

明在开始之前建议用户先熟悉本说明中的绪论，指令，数据寄存器篇章，这些章节写了使用Q驱动器的相关信息。绪论中描述了Q驱动器基本的结构和功能。指令部分描述了两 种不同类型的指令（立即和缓存器）和用法。数据寄存器部分列出了Q驱动器中的寄存器以寄存器以及如何访问。因为这份说明包含了针对不同应用使用Q驱动器的信息，您需要参考适合您应用的部分。

独立（程序驻留）或主机控制

这里您需要决定是否让驱动器运行驻留程序。如果是，您需要熟悉本说明中对Q Programmer软件的描述。Q Programmer用来为Q驱动器编写程序程序。如果您的应用不需要程序驻留，那么驱动器由主机控制器输入指令来实时控制，您需要选择串行连接-RS-232或RS-485并且熟悉本说明中主机串行通讯协议和串行通讯语言（SCL）。

在许多应用中Q驱动器会同时用到上面提到的俩点，即用主机实时控制和程序驻留，这里，您需要为您的主机控制器选择最佳的串行连接和通讯协议。通过用相兼容的编程电缆（每个驱动器配套）和RS-232连接，Q Programmer软件在基于WINDOWS操作环境下的PC中运行。您也可以选择RS-485进行通讯。

单轴或多轴主机控制

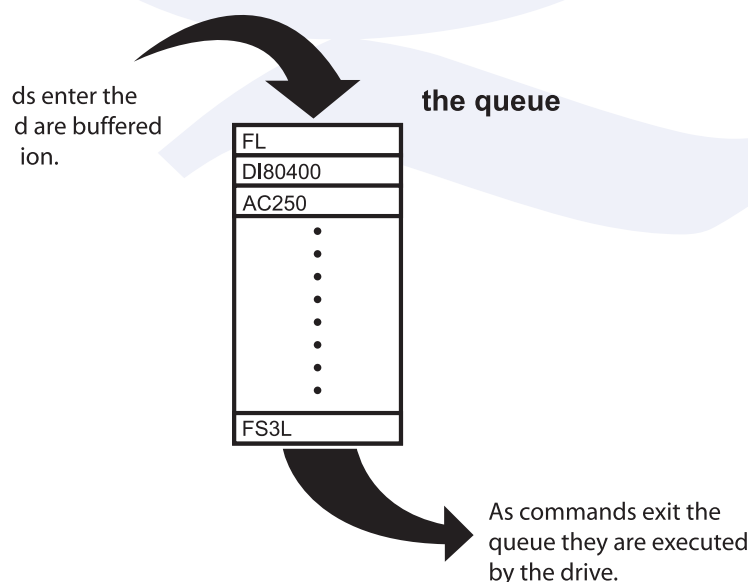
如果您想连接多于一个驱动器到主机控制器上并输入指令来控制，您可以选择2线RS-485或4线RS-485 串行通讯来连接多个驱动器到控制器串口上。而且RS-485电缆可以达到3000米通讯距离，RS-232只有150米。同时，您需要熟悉本说明中关于多轴通讯更多的信息。

绪论

Q的含义

Q驱动器中的Q来自单词queue。这是因为Q驱动器执行的命令是排入栈，或缓存器，储存在驱动器控制部分。定义：栈是条目的列表或序列，条目从栈的一边进入,从另一边出。对于我们的驱动器，栈中的条目就是指令，当指令到达栈的底部的时候控制器就会执行该指令。

栈存在于控制器不稳定的存储部分中，当电源关闭后留在栈中还没有执行的指令就会丢失。



指令如何进入栈中

指令通过连接驱动器的串口（从主控制器发出）或驱动器自己的非易失性存储单元进入栈中，最初，我们通常通过串口来输入指令（用RS-232 连接驱动器到PC的COM口,在Windows环境下的PC上运行Q-Programmer软件即可实现）。这里，指令还可以通过其他可编程设备来输入，例如HMIS，微型处理器或其他类型的控制器（用RS-232 或RS-485连接）。这些设备只需要遵循Q中的通讯协议来配置。

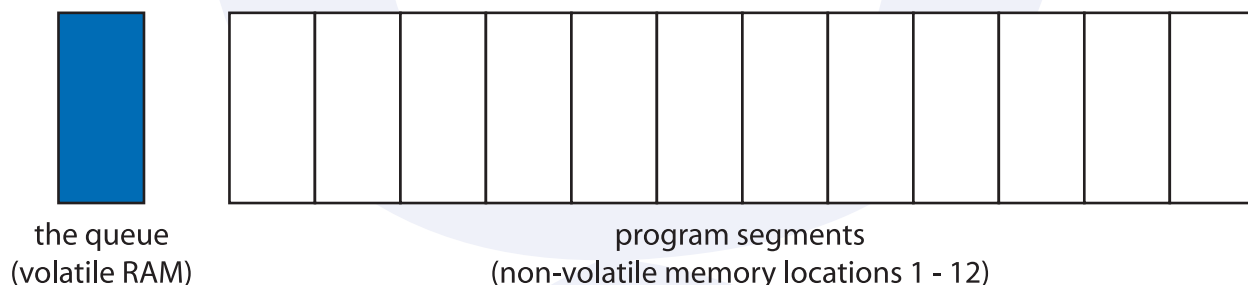
当串行通讯建立，我们可以输入指令到栈中，立即执行或缓存处理，或者通过栈把指令存到驱动器的非易失性存储器中，即进行编程，这样，存储的程序即使驱动器断电也不会丢失。

Q驱动器中的程序

一个指令由2个或3个ASCII码字符（只有字母是代表命令本身）构成。一些指令在初始的2或3个ASCII码字符后面还跟有其他参数（字母，数字，其他的ASCII字符）。这里有俩中基本的指令：立即和缓存。一个立即指令会被马上执行，不管目前栈中在它之前的指令的指令是什么。一个缓存指令会在栈中等到其他在栈中存贮的缓存指令执行完以后才会执行。

提示：栈中最多可以存储62条指令。

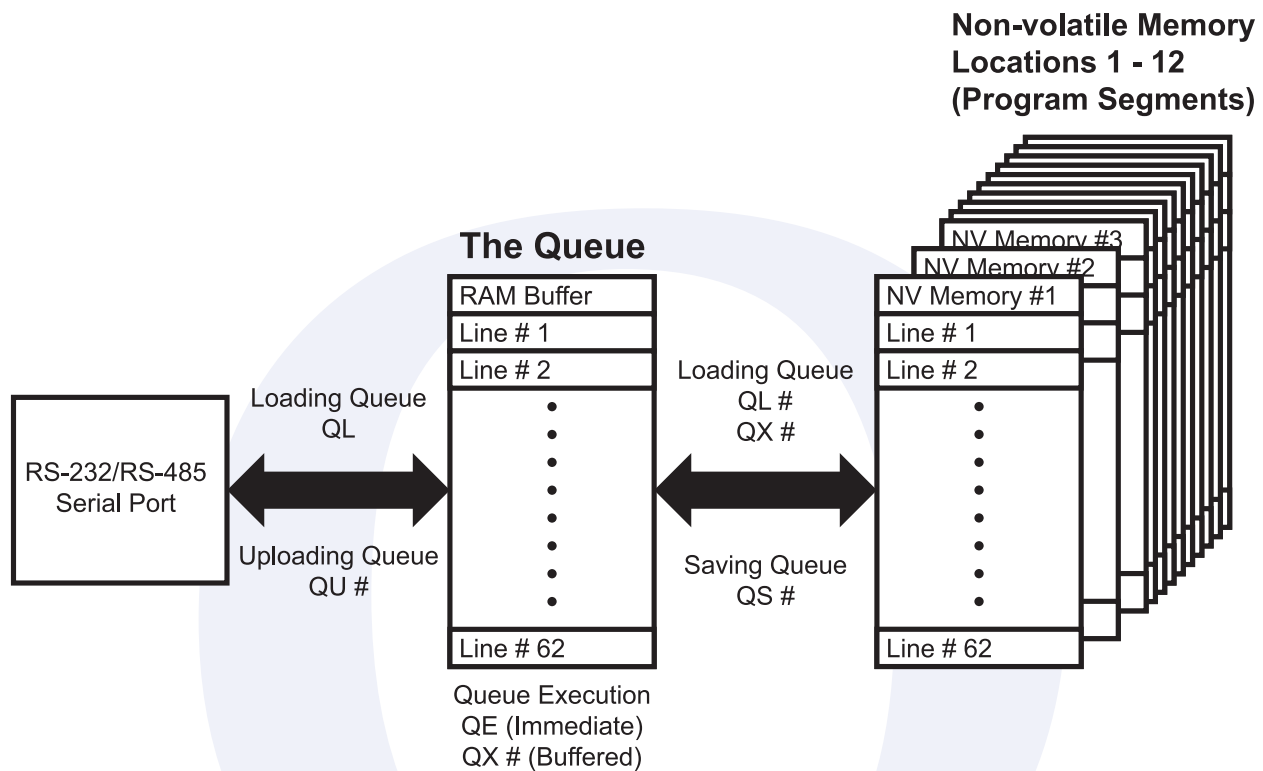
现在，虽然62条指令可以组合提供很多功能，但是在应用上还是有更多的需求。这里我们设计非易失性存储单元来存储更多的指令，然后把单元分解成和栈一样的大小。



每个驱动器中有12个非易失性存储单元，都可以用来存储指令，表现在Q-Program中就是有12个部分（segment），每个部分有62行，即一个栈的长度。每个部分中的指令可以马上调入到栈中。例如，我们开始程序先执行segment 1，在最后有个指令是调segment 2到栈中。这个调用过程仅需要125微秒，基本上可以忽略。

12个部分相互调用，每个部分有62行指令，所以现在可以有744个指令，可以实现更多功能的组合。

下面是指令怎么样实现指令从驱动器串口和驱动器非易失性存储单元到栈中的转换。



在上面的图片中，指令通过驱动器串口进入栈中，通过Load和Upload指令。Load指令可以将主控制器（PC，HMI，微型处理器，等等）的命令存储到栈中。Upload指令用来把栈中的指令返回到主控制器中。

同时，Load指令可以把在非易失性存储器中的指令调入到segment中，即进入栈中。保存（Save）可以把栈中的指令作为segment指令保存到非易失性存储单元中。

这样Segment中的指令可以按任意的顺序来调用，执行。这里有一些规则：Segment 1 永远是在驱动器上电后第一个被执行的段落。你可以认为segment 1 是自动执行的部分。同样，当使用中断功能，程序会默认执行segment 10。因此中断后如何动作的指令应该被写到segment 10 中。除了这两个规则，你可以任意调用segment中的程序。

命令

在绪论中介绍，指令是通过放入栈中执行。指令放入栈中有俩种渠道，驱动器串口和非易失性存储单元。在所有应用中，你首先需要编写指令到Q驱动器。然后，如果你的应用需要运行一个存储的程序，你需要存储指令到驱动器非易失性存储单元中。下面的内容描述了Q驱动器中指令的类型。

指令结构

Q驱动器中所有的指令由3个可能的部分构成：指令本身和俩个参数。一些指令没有参数，一些指令一定需要配参数，还有一些指令的功能取决于是否有参数。结构如下：

指令（参数1 ）（参数2 ）

指令一般由2个字母构成（有的是3个），参数一般由ASCII字符构成-字母，数字，其他的ASCII字符。例如：

FL

Feed to length指令 没有参数 立即执行

S01L

Set Output 指令 参数1="1", 参数2="L" 设置驱动器输出1为低电平

RL91000

寄存器Load指令 参数1="9", 参数2="1000" 用户定义的数据寄存器9写入值1000

缓存器和立即指令

两种类型的指令：缓存和立即。缓存指令在栈中按顺序存储，按存储的顺序执行。如果您向栈中写入两个缓存指令，那么第二个指令只有在第一个指令结束后才会执行。（在多任务处理部分有例外）

立即指令是立即被执行的，不管栈中其他执行的指令，就是说可以和缓存指令并行执行。

所有的指令都是这两种类型。

只有缓存指令可以用来编写程序。原因很简单，在驻留的程序中，指令是根据外部事件同时处理的。立即指令不需要外部事件的触发。

当使用主机控制器写入指令到Q驱动器中，立即和缓存指令都可以使用，这取决于功能是否复杂，需要何种控制。在一些应用中，立即指令和缓存指令都可以达到目的,这样您就需要做出最佳的选择。例如：

报警：AR（立即）AX（缓存）

停止：ST（立即）SM（缓存）

清除栈：SK（立即）QK（缓存）

寄存器装载：RL（立即）RX（缓存）

缓存指令类别

接下来是7个缓存指令的描述。

运行指令

伺服指令

配置指令

I/O指令

通讯指令

Q 程序指令

寄存器指令

运行指令

运行指令控制电机转轴的位置和速度，包括相对运行，绝对运行，条件运行，点动，停止运行，等等，进而影响到运行中的一系列参数：加速度，减速度，点动速度，最大加速度，速度，等等。

伺服指令

伺服命令用于伺服控制。包含调整PID环路，滤波器，和速度环路模块，还包括使能伺服系统。

配置指令

配置指令用来设置电流峰值和有效值，定义地址，设置限位和位置错误，编码器处理等等。

I/O指令

I/O指令设置驱动器的数字输入，模拟输入和数字输出。对于数字输入，指令包含等待输入，定义限位，滤波输入，定义报警输入，定义伺服使能输入等等。对于模拟输入，指令有滤波输入，设置偏移量，最大值和输入置零。对于数字输出，指令有报警，刹车和运动输出，还有设置通用输出口。

通讯指令

三个通讯指令设置通讯协议，调整波特率，设置传输延时。

Q 程序指令

在本类别中有三个子类别：Q，其他和等待。Q子类别中的指令用来写入，上传和保存指令进栈或出栈，分支，循环，调用子程序。其他子类别的指令用来实现多任务处理，中断，密码保护，等等。等待子类别有4个指令：等待输入，等待时间，等待寄存器延时，等待运动结束。

寄存器指令

寄存器指令用来对寄存器中的数据进行数学运算，从寄存器移入移出数据。计数，减1，加1，加减法，除法，乘法，与，或，寄存器数值比较，转移数据，读，写，等等。

缓存指令列表

运行指令

AC	加速度
AM	最大加速度
CJ	开始点动
DC	改变距离
DE	减速度
DI	距离
EG	微分
FC	运行到设定距离改变速度
FD	运行并检测两个传感器信号

FE	编码器跟随
FL	运行到设定距离
FM	带屏蔽距离的FS
FO	运行期间改变输出状态
FP	运行到绝对位置
FS	运行到传感器信号有效
FY	带安全距离的FS
HW	跟踪手轮
JA	点动加速度
JD	点动减速度
JE	点动使能
JM	点动模式
JS	点动速度
SH	找原点
SM	停止运动
SP	设置绝对位置
VC	改变速度
VE	速度
VM	最大速度

伺服指令

AX	报警复位
EP	编码器位置
KC	伺服滤波器
KD	微分增益
KE	微分滤波器
KF	速度前馈增益
KI	积分增益
KK	加速度前馈增益
KP	比例增益
KV	速度反馈增益
KW	速度反馈滤波器
MD	电机禁止
ME	电机使能
VI	速度模式积分增益
VP	速度模式比例增益

配置指令

CC	改变运行电流
CM	控制模式
CP	改变峰值电流
DA	定义地址
ER	编码器处理
PC	上电电流
PF	位置错误

PL	限位
PM	上电模式
PP	上电峰值电流
PT	位置时序
SA	保存所有参数

I/O指令

AF	模拟量滤波
AI	报警复位输入
AO	报警输出
AT	模拟量极限
AV	模拟偏移量
AZ	模拟量置零
BD	刹车释放延时
BE	刹车使能延时
BO	刹车输出
DL	定义限位
FI	输入滤波器
MO	运动输出
OI	输入
SI	伺服使能输入
SO	设置输出
SOY	设置输出扩展
TI	检测输入
WI	等待输入

通讯指令

BR	波特率
PR	通讯协议
TD	传输延时

Q程序指令

MT	多任务
OF	故障
OI	输入
PS	暂停
PW	密码
SS	输送串
WI	等待输入
WM	等待运时
WT	等待时间
WD	等待延时寄存器
QC	Q程序调用
QG	Q程序栈内跳转
QJ	Q程序栈间跳转

QK	清空栈
QR	Q循环
QU	Q程序上传
QX	Q程序下载，运行

寄存器指令

CR	寄存器比较
DR	数据寄存器比较
RC	寄存器计数
RD	寄存器减1
RI	寄存器加1
R+	寄存器加法
R-	寄存器减法
R*	寄存器乘法
R/	寄存器除法
R&	寄存器与
R 	寄存器或
RM	寄存器移动
RR	寄存器读
RW	寄存器写
RX	寄存器装载
SR	设置寄存器指示器
SV	设置寄存器值
TS	时间标记阅读

指令结构

AC	报警代码
AR	报警复位
BS	缓存状态
CE	错误
CS	改变速度
CT	继续
GC	电流指令
IA	即时模拟量值
IC	即时电流值反馈
ID	即时电流距离值
IE	即时编码器反馈
IF	即时版本反馈
IH	即时输出高电平
IL	即时输出低电平
IP	即时位置反馈
IO	即时输出状态反馈
IQ	即时电流值反馈
IS	输入状态反馈
ISX	扩展输入状态反馈
IT	即时温度反馈
IV	即时速度反馈
IU	即时总线电压反馈
IX	即时位置错误反馈
MN	模式代码
QE	Q程序执行
QL	Q程序载入
QS	Q程序保存
RE	复位或重新启动
RL	寄存器载入
RS	寄存器状态
RU	寄存器上传
RV	版本反馈
SC	状态十六进制反馈
SJ	停止点动
SK	停止&清楚缓存（栈）
ST	停止运行

数据寄存器

前边描述的许多指令用来传输数据到驱动器，这些指令设置的数据值被保存在驱动器的数据寄存器里。不断电的前提下，数据值会一直保存直到新的指令输入。

提示：一些寄存器会自动保存在非易失性寄存器中，在断电后也不会丢失。例如PC和PM指令。同时，你可以用RW指令将任意的数据寄存器写到非易失性寄存器中。

例如：DI指令，将数据存储在驱动器中等待后面的应用。这个指令将值写到D寄存器中，然后这个值被很多的指令调用，例如：FL，FS，CJ，等等。DI指令用来改变D寄存器。

RL指令也可以改变D寄存器的值（RL是立即指令，RX是缓存器指令），也可以载入值到D寄存器中。事实上，RL指令可以向任意可写入寄存器中载入值。

Q驱动器中有许多数据寄存器。下面试三种数据寄存器的描述：

只读寄存器

只读寄存器被设定为保存驱动器参数，配置和状态的寄存器，包括伺服功率放大需要的电流值，电机编码器位置，模拟输入，驱动器温度，内部总线电压，等等。用户不能改变这些寄存器的值，但可以监视。所有只读寄存器用小写字母表示。下面是只读寄存器的列表：

a.....	模拟量指令
b.....	栈的行数
c.....	电流指令
d.....	距离
e.....	编码器位置
f.....	报警代码
g.....	传感器位置
h.....	状态代码
i.....	驱动器输入
j.....	模拟量输入1
k.....	模拟量输入2
l.....	错误
m.....	控制模式
n.....	速度状态
o.....	位置状态
p.....	Segment编号
q.....	实际电流值
r.....	平均功率
s.....	状态代码
t.....	驱动器温度
u.....	总线电压
v.....	速度
w.....	速度前馈
x.....	位置错误
y.....	运行中
z.....	电机相错误

读/写寄存器

读/写寄存器设定为用户可设置参数的寄存器，包括加速度，速度，移动距离，持续电流，电流峰值，等等，用大写字母表示。

A	加速度
B	减速度
C	改变距离
D	距离
E	编码器位置
F	其他标记
G	电流反馈
H	保留
I	输入计数
J	速度反馈
K	点动加速度
L	点动减速度
M	最大加速度
N	持续电流
O	峰值电流
P	位置反馈
Q	保留
R	细分
S	第一步位置
T	总计数
U	改变速度
V	速度
W	时间标记
X	模拟量位置增益
Y	模拟量死区
Z	模拟量偏移值

用户定义寄存器

用户定义寄存器是未被定义的可读写寄存器，可以由用户自己定义以便实现更多的功能。这些寄存器由一位数字和其他ASCII码表示。

0
1
2
3
4
5
6
7
8
9

:	(冒号)
;	(分号)
<	(小于号)
=	(等号)
>	(大于号)
?	(问号)
(	(左括号)
\	(反斜线)
)	(右括号)
^	(shift+6)
_	(下划线)
,	(省略号)

非易失性数据寄存器

对于以上所描述的三种类型的数据寄存器，有125个非易失性寄存位置来存储数据寄存器的值。每个非易失性寄存器存储一个数据寄存器的数据，但并没有对应关系，所以您需要记住哪一个非易失性存储器中存的是什么样的数据。

进入数据寄存器

载入

进入数据寄存器靠载入数据到寄存器中和上传数据到寄存器中。载入一个数据寄存器靠从主机控制命令行写入或程序中的一行来写入。用RL指令从主机控制行载入寄存器，这个指令可以在任意时刻执行，即使驱动器正在执行程序，是立即指令。用RX指令从程序中一行载入一个寄存器，这是个缓存指令。

上传

上传数据寄存器只能靠主机控制行，不能从程序中。指令是RU，是立即指令，即可以在程序运行时执行。上传指令中，第二个参数可以使10个数据寄存器返回到主机控制器中。

写

写一个数据寄存器允许用户存储数据寄存器值到非易失性存储单元中。用RW指令来写入。在NV单元中有125个存储位置，并且不合任何数据寄存器由关联，所以用户需要知道数据寄存器存储在NV单元的什么地方。

读

读数据寄存器允许用户将当前NV中的数据移到数据寄存器中。RR (Register Read) 指令用来读寄存器，一般在程序中使用。

操作数据寄存器

移动

数据寄存器值可以从一个寄存器移到另一个中，通过RM指令。当执行RM指令时，原来寄存器中的数据会被保留。只读寄存器的内容可以被转移到读/写寄存器中和用户定义寄存器中。数据不可以转移到只读寄存器中，这样做不会有结构也不会有报错。

加1/减1

读/写和用户定义的寄存器可以做加1减1运算，通过RI（寄存器加）和RD（寄存器减）指令。加德幅度超过寄存器范围后数据会溢出。

计数

一个特殊的寄存器，I寄存器（计数寄存器）用来对被选择的数字输入状态改变的次数进行计数。I寄存器是读/写寄存器，可以使用用所有的寄存器功能包括数学功能和状态测试。

RC（计数）指令用来指定数字输入寄存器计数。有四种不同的输入状态可以选择而且对输入计数有不同的影响。当使用"高""低"状态，寄存器像一个125毫秒的计时器进行计数。边沿状态像"下降沿""上升沿"也被用来输入计数。（参见SCL语言中的RC指令）。

数学逻辑

数学逻辑功能可以使用在数据寄存器上。数学功能仅限整数值。一些数学功能还限制只能是16位的值。当数学功能每次只能进行一次操作。数学逻辑结果保存在累加寄存器"0"。这个寄存器是用户定义寄存器的一部分。数学功能包括加法，减法，乘法和除法。逻辑功能包括逻辑与和逻辑或。

状态检测

当使用程序的时候通常需要做一些状态检测判断来支持程序的执行。TR（检测寄存器）和CR（比较寄存器）两个指令用来操作数据寄存器判断状态。TR指令比较数据寄存器中已知的"第一个"值和"第二个"立即值。CR指令比较数据寄存器中的已知"第一个"的值和另一个寄存器中的"第二个"值。当使用TR和CR一个内部状态寄存器根据结果会被设置。结果可以是：

"True"	第一个值或正或负
"False"	第一个值为0
"Positive"	第一个为正
"Zero"	第一个值为0
"Greater Than"	第一个值大于第二个值
"Less Than"	第一个值小于第二个值
"Equal"	两个值相等
"Unequal"	两个值不等

提示：QJ（栈跳行）指令用以上的状态代码来实现跳转。

主机串行连接

用RS-232 连接到PC

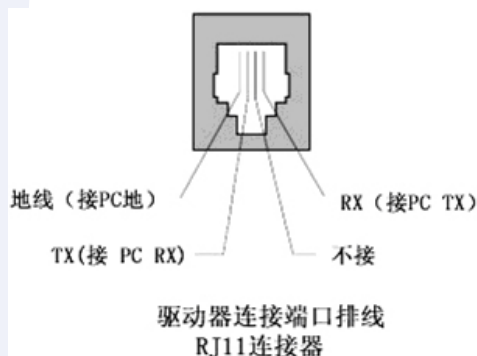
- 将PC 放在离驱动器2.5 米以内的地方
- 驱动器包装中配有一个黑色的适配器连接线。它一端是像电话线插头一样的水晶头，另一端是一个9 针的插头。将大的一端插在PC 的COM1 串口上，确定适配器边上的螺丝拧紧。

永远不要将电话线插头连接在驱动器上。驱动器连接线插头和电话线插头看起来是一样的，但两者的电压完全不兼容。

如果你电脑上的COM1串口已经被其他设备占用，你可以使用COM2口。在PC上，COM2口是一个25针的连接器，与黑色的适配器连接线并不匹配。这种情况下，那你需要一个25针到9针的适配器。

如果你的电脑上没有串口，你需要购买一个“USB 串口转换器”。我们推荐 CablesToGo的 Port Authority “USB Serial DB9 ”适配器（CablesToGo.com）以及 SewellDirectSW1301(SewellDirect.com)。

对于笔记本电脑，PCMCIA 转换卡也是一个很好的选择，我们推荐 SewellDirect的 [SSP-100](http://SewellDirect.com)。你也可以选择市场上供应的其他USB接口为9针RS-232接口的适配器来达到同样的目的。



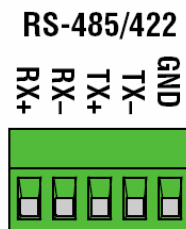
使用RS-422/485可选模块连接到主机

RS-422/485允许你将多个驱动器连接到一个主机(PC、PLC、HMI等)上，同时通讯距离也更长(通常可以达到300米)。但是你所连接的设备必须要有RS-422/485接口。

右图是接口图示，接线图在下一页给出。

我们推荐使用Category 5 cable(Cat-5, 双胶线),它被广泛地使用在电脑网络中，价格便宜、容易购买，便于测试。

ST驱动器通讯可以使用RS-422(全双工)或者RS-485(半双工)。连接方式可以为点对点(例如一台驱器和一台主机)或者多站式网络(一台主机和最多32台驱动器)。通常情况下，人们把RS-422和RS-485通称为RS-485通讯。



RS-422系统中数据的发送和接收分别使用不同的线缆。主机通过一对连接到驱动器的RX+和RX-端的线缆向驱动器发送信号，主机则通过另一对连接到驱动器TX+和TX-端的线缆接收由驱动器传回的信号。一个逻辑地线端可以用来保证所有驱动器共地。这个端口在内部与连接电源负极的端口相连，如果RS-422/485网络中的所有驱动器使用的是同一个电源，这个逻辑地端口可以不用连接。但是你仍然需要将其中一个驱动器的地线与主机的地线共地。

RS-422 RS-485

主机将不需要停止发送数据，这将使你的软件编写更为简单。

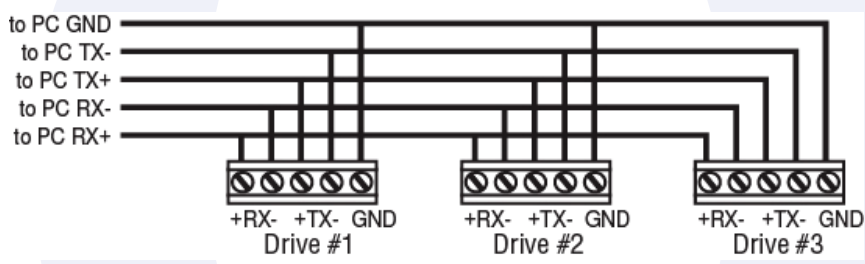
用RS-232到RS-422转换器

Jamesco Electronics(800-831-4242)的转换器型号 [117701](#) 可以将你电脑的RS-232端口转换为RS-422。这个适配器用于25Pin的串口。如果你的电脑上只有9Pin的串口，你需要购买Jemeco的转换电缆 [31722](#)。

适配器连接如图：

adaptor	drive
1	RX+
2	RX-
3	TX-
4	TX+

设置好分布式系统和TxON、RxON的开关。不要忘了接上各单元所用的直流电源适配器。



RS-422系统连线

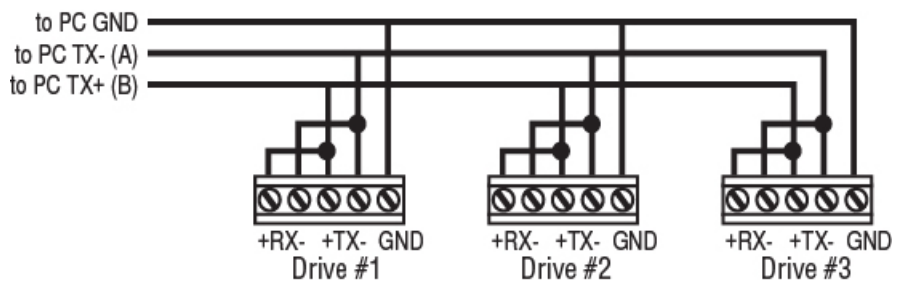
RS-485系统中数据的发送和接收使用同一对线缆，这样可能会造成一些问题。主机不仅需要在接收数据前停止发送数据，同时数据发送过程要很快，必须在其中一个驱动器开始应答前完成发送。ST系列驱动器需要设置一个可调的参数“transmit delay (发送延迟)”用以配合那些关闭发送较慢的主机。这个参数可以通过TD命令配置整个网络，也可以通过ST Configurator软件配置。在RS-422系统中不需要设置发送延迟。

用RS-232到RS-485转换器

Integrity Instruments 的转换器型号 [485-25E](#) 可以将你电脑上的RS-232端口转换成RS-485。将适配器的“B”针脚连接到ST驱动器的TX+和RX+端口，将“A”连接以驱动器的TX-和RX-端口。

USB到RS-485转换

[Byterunner](#) 的USB-COMi-M是USB到RS-485转换的很好选择。将SW1设为ON，SW2-4均设为OFF。在USB-COMi-M的螺纹连接端子侧，Pin1连接至RX+和TX+，Pin接地。DB-9不需要使用。

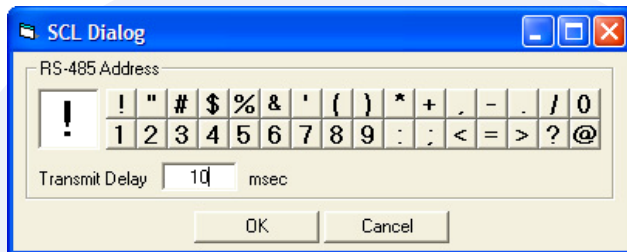


RS-485系统连线

编排地址

在连接整个系统前，你需要将每个驱动器单独连接到主机上以配置一个独一无二的地址，使用驱动器附带的编程电缆和ST Configurator 软件来完成。

将驱动器和你的PC 相连，运行ST Configurator 软件，最后给驱动器上电。如果你已经配置过这个驱动器，点击Unload 按钮上载该驱动器的设置。点击Motion 按钮，选择"SCL"模式或者"Q Programmer"模式。你将看到一个如下图所示的RS-485 地址面板，点击你选择的地址字符，你可以使用数字0..9 或者特殊字符 ! " # \$ % & ' () * + , - . / : ; < = > ? @。保证网络中的每一个驱动器都有一个独一无二的地址。



主机通讯协议

根据驱动器不同的应用，您可以调节驱动器串行通讯协议来满足最佳的应用，调节指令是PR。通常PR 只在配置驱动器时用一次，然后被作为启动参数保存起来。不过，它也可以随时用来改变串行通讯协议。PR指令靠发送5位二进制数工作，每一位代表通讯协议中的一个设置。这5位相当于5个开关，每个开关闭合或打开，对应不同的设置。

PR指令细节

下图是5位二进制数每一位对应的通讯协议。PR指令输入的值是对应于这个5位二进制数的十进制数。

Bit0-默认值(标准SCL)

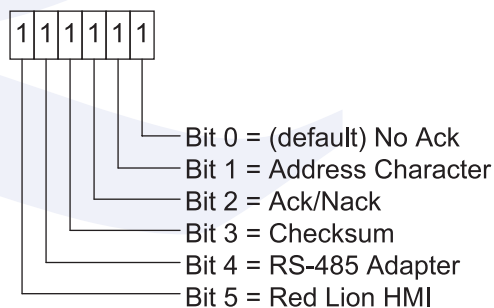
如果指令不要求返回值，则不会发送响应，也就是说，驱动器只有在指令要求返回值时才会发送响应。

发送数据举例：

```
Tx DI8000    设置距离为8000
Rx
Tx 1DI8000    地址为1的驱动器设置距离为8000
Rx
```

返回数据举例：

```
Tx DI        要求距离返回
Rx DI=8000    驱动器返回距离值
Tx 1DI       要求地址为1的驱动器返回距离值
Rx 1DI=8000   驱动器1返回距离值
```



Bit1-地址字符(总是发送地址字符)

当Bit1=1时，驱动器地址会一直随数据返回。

发送数据举例：

Tx	VE50	设置速度为50rps
Rx		
Tx	1VE50	1驱动器设置速度为50rps
Rx		

返回数据举例：

Tx	VE	速度返回
Rx	1VE=50	1驱动器返回速度值
Tx	1VE	1驱动器返回速度
Rx	1VE=50	1驱动器返回速度值

Bit2-Ack/Nak(总是发送应答字符)

这个指令要求驱动器对主机每次传输的指令做出回应，不管指令是否需要回应，即如果主机发送不需要返回数据的指令，驱动器仍然会返回下面字符中的一个：

"%"百分号表示驱动器接收命令并执行

"*"星号表示驱动器接收命令并存储到栈中

"?"问号表示驱动器接收指令时发生错误，后面可能会跟字符代表错误类型，例如：

- 1=指令传输超时
- 2=参数过长
- 3=参数过少
- 4=参数过多
- 5=参数超出取值范围
- 6=缓存器（栈）满
- 7=指令不能执行
- 8=程序正在运行

应答字符总是会输出到RS-232端口。如果是2线或4线的RS-485网络模式，应答字符将依下列情况输出。

如果指令里的起始位有地址符，将发送应答字符。

如果全局指令不要求返回数据，就不会发送应答字符。

如果指令里没有地址符，且要求返回数据，如果网络上有多个驱动器，每个驱动器都会返回数据，就会造成数据冲突。注意，在网络模式中，指令一定要带上地址，以避免冲突。

注意：在网络模式中尽量不要使用应答字符（%，*，?）作为地址符，以避免混淆。

正确指令举例：

指令	返回值	注释
D18000	%	驱动器全部响应，设置距离为8000，发送正确应答
1D18000	1%	地址为"1"的驱动器响应，设置距离为8000，发送正确应答

错误指令举例：

指令	返回值	注释
VE200	?5	驱动器全部响应，设置速度为200，发送错误应答，因为参数200溢出
1VE200	1?5	地址为"1"的驱动器响应，设置速度为200，发送错误应答，因为参数200溢出

寄存器指令举例：

指令	返回值	注释
AC10	*	驱动器全部响应，设置加速度为10转/秒 ² ，发送接受应答
1AC10	*	地址为"1"的驱动器响应，设置加速度为10转/秒 ² ，发送接受应答

Bit3-校验 (8位校验)

没有校验位。

Bit4-RS-485

允许主机通过驱动器232接口在485通讯网络上把驱动器用作RS-485转换器。当一开始时主机发送"~"到RS-232口，RS232和RS485口都会回复该指令。连接到RS485网络上的驱动器都会收到"~"指令。

不设置Bit4 (Bit4=0)，Q驱动器通常只会回应RS232口的任意处理命令，不论指令是从RS232还是RS485口接收。当使用Bit4后 (Bit4=1)，驱动器就会同时回应RS485口的指令，允许连接到一个驱动器的主机通过RS232口与RS485网络上的驱动器进行通讯。

提示：当Bit4=1，Bit2=1，主机会接受到回应包和确认包。例如：两个驱动器连接到RS-485 网络上，而且都设置了Bit4=1，Bit2=1，然后第一个驱动器通过RS232连接到主机，地址为1，另外一个驱动器地址为2。

发送数据：

Tx	~2DI8000	设置2驱动器距离为8000
Rx	2DI8000	1驱动器做出回应 (RS232和RS485)
Rx	2%	2驱动器回应接受命令并执行

回复数据：

Tx	~2DI	要求驱动器2回复距离值
Rx	2DI	1驱动器做出回应 (RS232和RS485)
Rx	2DI8000	2驱动器回复距离值

Bit5-三位数字代码寄存器地址

通常情况下，访问数据寄存器使用单字母、号码或其它的ASCII字符。设置Bit5后 (Bit5=1, PR=32)，所有的数据寄存器被改为使用3位数字代码 (000~074) 访问 (见 数据寄存器部分)。Bit5的设置，实现了RL指令和RU指令的使用，更实现了驱动器和HMI的兼容。

注意：不管Bit5是否设置，当驱动器返回数据时，数据寄存器总是用指定的字符表示。

RL指令举例：

指令	返回值	注释
RL017100	-	给寄存器017 ("A") 赋值100
RL017	RLA=100	驱动器返回加速度寄存器的值，寄存器用"A"表示

RU指令举例：

指令	返回值	注释
RU0174	RUA=100	
	RUB=150	
	RUC=140	
	RUD=210	

PR指令举例

例如：打开Ack/Nack (Bit2) 和红灯功能 (Bit5)

对应这个设置的5位二进制数为100100，十进制是36，因此，输入PR36到驱动器。

例如：打开RS485转换功能 (Bit4)

对应这个设置的5位二进制数为010000，十进制是16，输入PR16到驱动器完成配置。

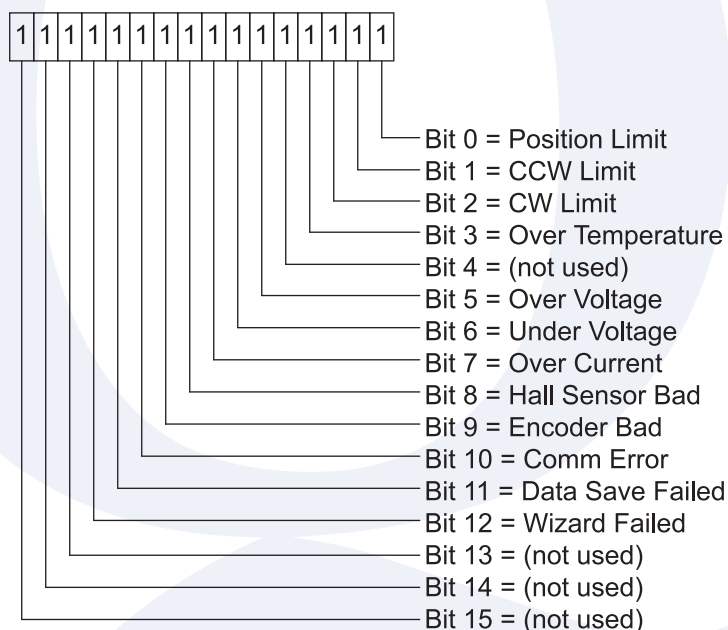
报警信号状态代码

Q驱动器有发送报警信号和状态代码返回到驱动器的能力。AL（报警代码）和SC（状态代码）指令可以用来在任意时间判断驱动器状态。如果一个驱动器报错或发送报警信号，AL指令允许主机找出是哪个或哪几个报警信号被触发。同样，SC指令允许主机找出哪个状态被触发，你可以根据状态代码提供的信息进行判断，例如判断驱动器是否在运行，到位，使能，在原点，和其他状态。报警信号和状态代码对初始化配置和装配伺服系统到你的机器上。

报错和状态代码是对应于16位二进制数的十六进制数，每一位代表一个意思，因此代码可以表示不止一个报警或状态情况的信息。

报警代码定义

下图是对每位的描述。例如，当Bit0=1表示伺服电机到位。同样的，如果Bit5=1，表示驱动器过压。当主机发出的AL指令时驱动器可以对所有位进行任意设置以获得报警或错误状态。



当主机发送AL指令，驱动器会真实反应对应于16位二进制数的十六进制数，该十六进制值表示报警代码，下面是对应的位：

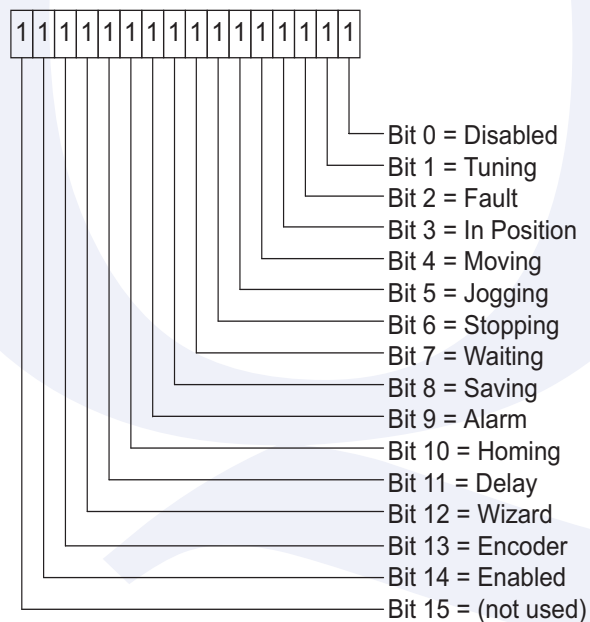
描述	十六进制值	位	注释
限位	0001	0	由PF指令设置
反向限位	0002	1	驱动器6号输入口开始工作
正向限位	0004	2	驱动器7号输入口开始工作
过温	0008	3	驱动器温度大于85摄氏度
不使用	0010	4	
过压	0020	5	总线电压大于58V
欠压	0040	6	总线电压小于18V

过流	0080	7	相电流大于20A
霍尔传感器出错	0100	8	霍尔传感器信号出错
编码器出错	0200	9	A或B信号不存在
通讯错误	0400	10	串行通讯错误
数据保存错误	0800	11	不能保存数据
向导失败	1000	12	时序向导失败
不使用	2000	13	
不使用	4000	14	
不使用	8000	15	

例如：当驱动器遇到正向限位（Bit2），有过压信号（Bit6），编码器信号出错（Bit9），对应的二进制数是0000001001000100，相等的十六进制数就是0244，因此，AL指令的返回值就是"AL=244"。

状态代码定义

下图是对状态代码的十六进制数描述。例如：当Bit1=1，驱动器处于非使能状态。同样，当Bit10=1，表示驱动器感应到原点信号（由SH-找原点指令定义）。主机发送SC指令后驱动器会对每一位随时进行设置并反馈。



当主机发送SC指令，驱动器会返回对应16位二进制数的十六进制数。

描述	十六进制数	位
非使能	0001	0
调整	0002	1
出错	0004	2
到位	0008	3
运动	0010	4
点动	0020	5
停止	0040	6

等待	0080	7
保存	0100	8
报警	0200	9
原点	0400	10
延迟	0800	11
向导	1000	12
编码器	2000	13
使能	4000	14
不使用	8000	15

例如：驱动器使能（Bit4），到位（Bit3），正在等待输入信号中（Bit7），对应该状态的二进制数为0100000010001000，相等的十六进制数十4088。因此，SC指令的返回值是"SC=4088"。

转化报警和状态代码为二进制数的工具

如果您使用的是安装Windows系统的PC机为主机，您可以使用Windows自带的计算器转化十六进制数为二进制数。

程序

这个部分描述了如何使用Q Programmer编程并存储到驱动器中。即使您的应用不需要程序驻留，熟悉Q Programmer对任意应用下的问题判断和解决仍然有所帮助。

准备

在程序中有12个段，编程中并不需要全部使用，但在程序中至少得有一个段。

对Q驱动器进行编程，最好可以对您的应用做出分段。例如，如果您的应用是找原点信号，进行绝对距离运动，然后进行点动，最后进行一系列增量运动，那么最好让一个段对应一段应用，也就是说，用一个段对找原点编程，一个段对绝对距离运动编程，一个段对点动编程，最后一个段对增量运动编程，这样就有4个段，另8个段是空白的。

注意第一个段默认是上电部分，所以在这里对一般的参数像加减速度，中断设置，寄存器值等其他不变的指令进行配置，通常也在这个部分对多任务处理进行设置。

要明确程序中的中断功能。例如：如果在一个中断输入信号后您想开始找原点（可以理解为复位），那么将这段中断程序写到第10段，默认为基于中断信号的段。同样，如果您想添加一些指令来纠正错误，您需要使用On Fault指令（适合写在第一段），指明当错误发生时程序跳转到那一段，然后在这一段中编写如何纠正错误。

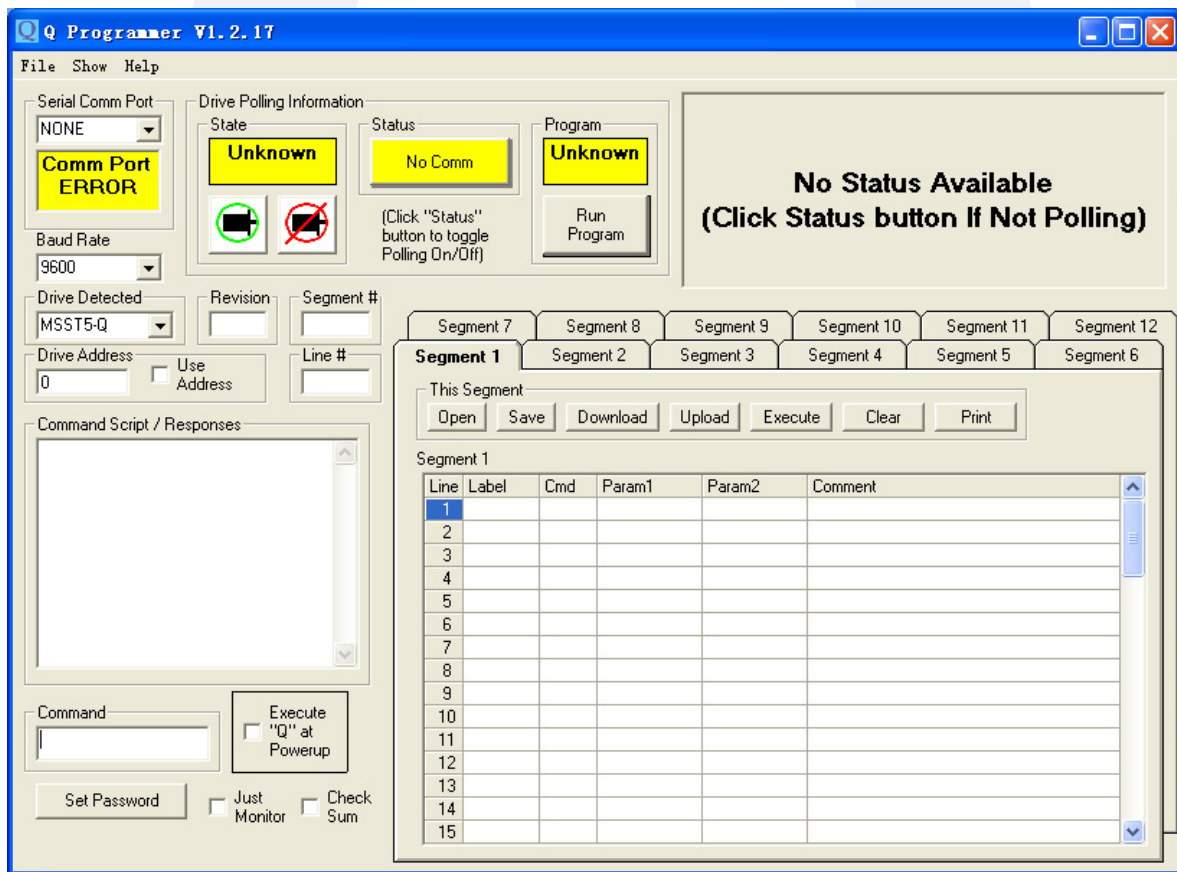
最后，将您的应用在逻辑功能上进行分段。您可以先对您所需功能的一段进行编程，然后进行下一个应用功能的编程，每一次实现一个功能，这样逐步进行，最终实现应用上的要求。

Q Programmer软件

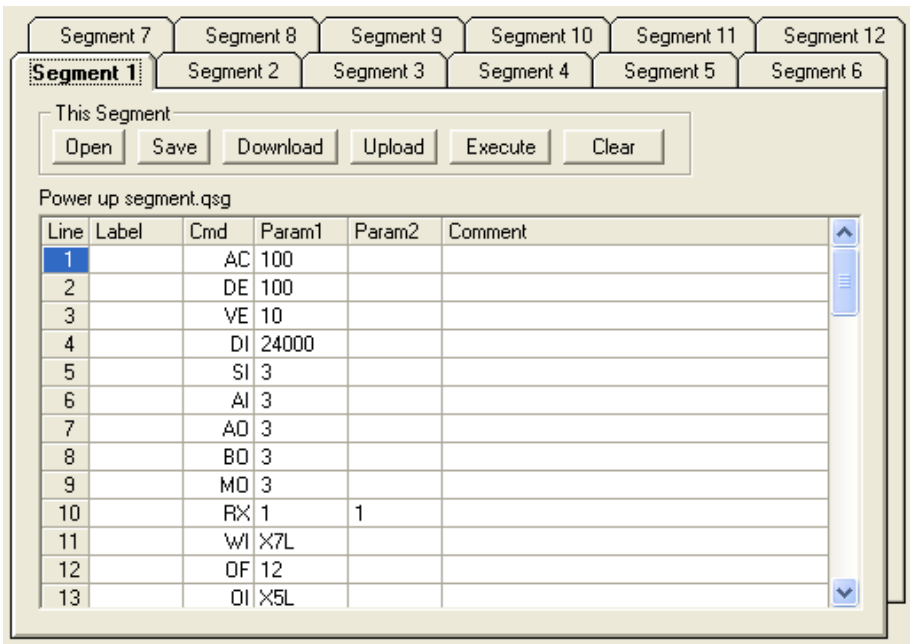
所有的Q驱动器可以通过串口或主机中断设备进行控制和编程。因为Q驱动器的通讯协议是基于简单串行通讯的，一个简单的终端（像SCL Utility）就可以进入Q驱动器的简单功能单元。但是，Q驱动器的功能范围超过了简单终端能实现的功能范围，所以Q Programmer软件用来帮助用户更好更有效率的使用Q驱动器。以下是Q Programmer主要的功能：

- 建立和配置于驱动器之间的串行通讯
- 不断通讯以检测驱动器状态
- 提供主机指令行直接发送指令到栈中
- 帮助用户编写，修改程序

下面是Q Programmer主屏幕



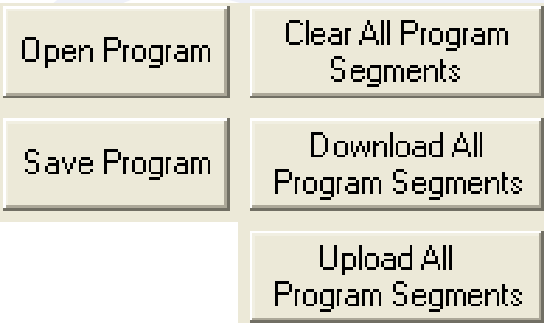
段编辑



上图中显示的是第一段的编写界面。上边有打开，保存，下载，上传，执行和清除按钮。每一个段中有62行来编写，每一行你可以定义标签，指令，参数（参数1，参数2）和一个注释。标签用作段中程序行的跳转。注释保存在硬盘上，但不保存在驱动器中。如果你下载一个带注释的段到驱动器中然后再上传，注释就会丢失。

段和程序文件：.qsg 和.qpr

当你使用段中的编写按钮用来打开，保存，下载，上传，和执行一个段时，你是对一个.qsg文件进行操作。在主控制界面上同样有打开/保存，下载/上传和执行按钮，但是是针对所有的段，即整个程序操作的。当使用这些按钮时我们是对.qpr文件进行操作，.qpr包括程序中所有的段，这对实际操作是由帮助的，可以大大的提高效率。



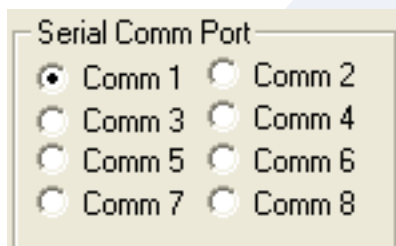
开始按钮

点开始按钮执行已经下载到Q驱动器中的程序的第一段。当选中"Execute Q at Power Up", 驱动器在上电后会自动执行保存的程序。START按钮在程序运行时会变成红色"STOP"。



串行口按钮

用户可以选择哪一个PC机串行口连接到Q驱动器上。当打开Q Programmer后, 会从1号口开始按顺序自动检测通讯接口。你可以选择使用哪个通讯口, 当设置后, 下次运行时Q Programmer会根据上次记忆来设置。



自动检测

Q Programmer会不断检测驱动器状态, 这有助于处理程序和应用中发生的问题。你可以使用Drive Idle按钮设置自动检测打开或关闭。



Comm Port状态显示是否在检测。当检测关闭, 窗口为绿色, 显示"Comm Port OPEN", 当检测打开, 窗口是紫色, 显示"Comm Port POLLONG"并闪烁。



自动检测开启时, 驱动器空闲按钮会

- 变绿, 显示"Program Running" (程序执行时)
- 变红, 显示"Drive Faulted" (驱动器出错并且不能运行)
- 变红, 显示"No Response" (和驱动器通讯丢失)
- 变黄, 显示"Faulted, Program Running" (驱动器出错但是可以运行)

如果自动检测关闭, 驱动器空闲按钮不会正确显示驱动器状态。

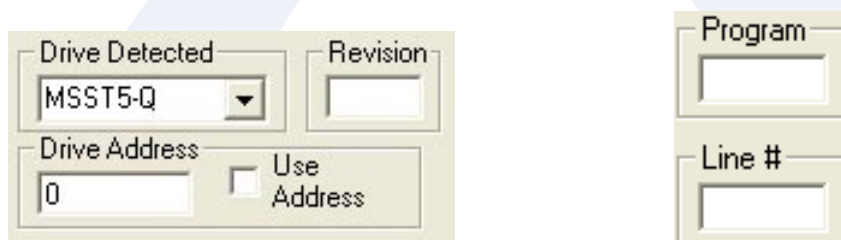
波特率

您可以从下拉菜单中选择一个波特率，有5种选择：9600，19200，38400，57600和115200。



驱动器信息和状态窗口

驱动器检测会告诉你连接到你电脑上的是哪一个驱动器，MSST-Q，MSST5-I，MSSTAC6-I，等等。固件版本会在REVISION一栏中显示，例如1.52。驱动器地址显示一个用户可定义的ASCII字符地址。使用RS485通讯时需要改变地址。Use Address一栏用来设置驱动器地址功能开启或关闭。

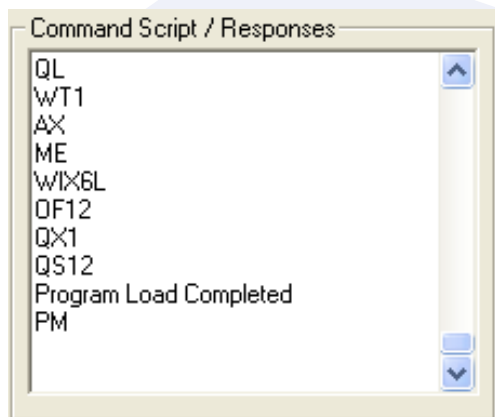


Program窗口显示当前运行的段，当程序运行时数值会随之改变。当没有程序运行或程序停止时窗口数值保持不变，您可以在这里观察到程序的运行。只有检测开启后窗口数值才会更新。

同时，当自动检测开启的时候，程序运行到哪一行在段中会有指示，这有助于程序的监测。

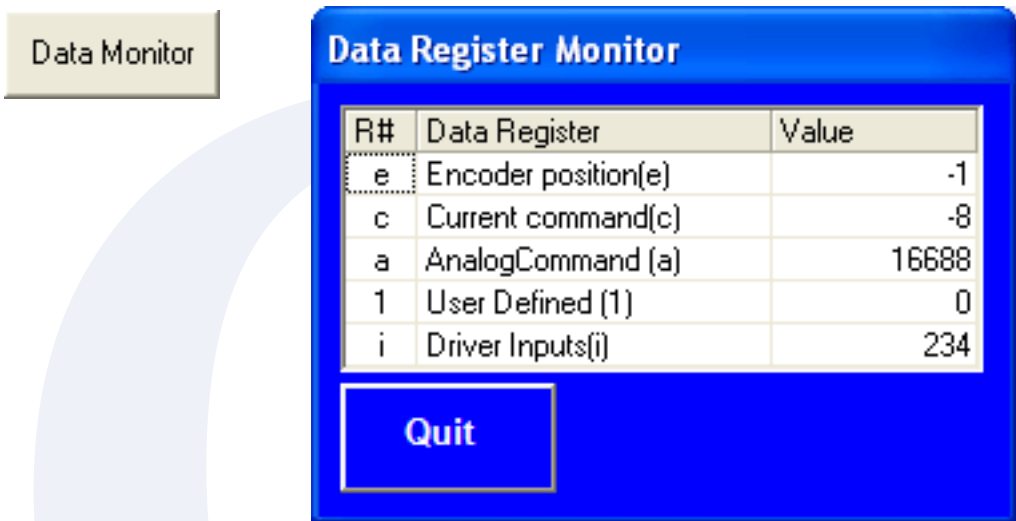
主机命令输入行

Q Programmer中包含主机控制命令行，允许我们立即监测指令和检查配置参数，并且不影响编写好的程序。在主机命令行中写程序不是下载到驱动器中，是直接写到栈中并操作。下载是一个非易失性功能：它保存程序中的段到非易失性存储器中。指令窗口显示发送到驱动器中的指令历史，包括直接写到主机指令行的，程序中的和单独得段中的指令。



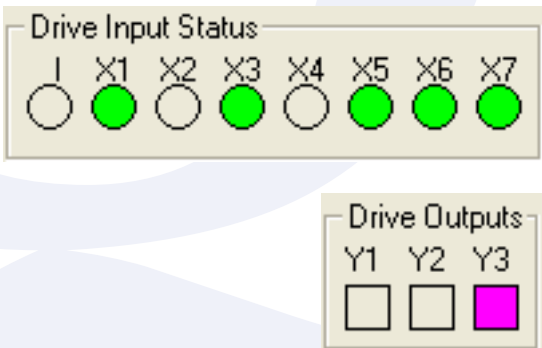
数据监视器

数据监视器按钮打开可以监视5个数据寄存器的窗口，自动检测开启后用户可以定义监视哪几个数据寄存器。例如在上图中，只读数据寄存器e, c, a, i和用户定义寄存器被选择监视。对按键 操作，您可以在所有寄存器中选择监视哪几个数据寄存器。



I/O口状态

驱动器数字输入和输出的状态在 Q Programmer右上角显示。同样，在自动检测开启后实现本功能。输入有效时显示绿灯，输出有效时显示紫灯。



指令序列示例

以下是对所有缓存器指令的举例。

运行相对距离

Segment 1				
Line	Label	Cmd	Param1	Param2
1		DI	80000	
2		AC	500	
3		DE	500	
4		VE	30	
5		FL		
6				
7				
8				
9				

FL (Feed to Length) 指令用于相对距离的运行，配合DI (距离)，AC (加速度)，DE (减速度)，VE (速度) 设定参数。方向由DI参数规定，DI8000表示正向旋转8000步，DI-8000表示反向旋转8000步。

上图是速度30转/秒，加减速500转/秒，正向运行80000步的程序。FL指令表示执行。FL指令前的指令顺序可以随意改变。

运行绝对距离

FP (Feed to Position) 指令用于绝对位置运动。使用FP指令时，配合DI (距离)，AC (加速度)，DE (减速度)，VE (速度) 设定参数，电机将根据驱动器内部的位置设定运行绝对距离，由DI参数设定。

上图表示电机运行到绝对位置4000步（正反转取决于运行前的电机位置），速度30转/秒，加减速500转/秒。另外一个与绝对距离运动有关的是SP (Set Position) 指令允许用户随时将电机位置零 (SP0)，或设置为一个任意值。例如SP5000指令将使电机相对当前位置 (SP0) 正向旋转2.5圈。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		DI	80000	
2		AC	500	
3		DE	500	
4		VE	30	
5		FL		
6				
7				
8				
9				

运行到传感器信号有效

FS (Feed to Sensor) 指令使电机以固定速度运行直到某个输入状态改变。当设定的某个输入改变状态时，电机减速到停止。参数由AC, DE, VE, DI指令设定。DI指令设定减速距离和减速方向，输入DI指令设定。DI指令设定减速距离和减速方向，输入状态可以设置为H(高)，L(低)，R(上升沿)，F(下降沿)。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		DI	4000	
2		AC	500	
3		DE	500	
4		VE	30	
5		FP		
6				
7				
8				
9				

上图表示电机以加速度500转/秒，速度30转/秒正向运行，当驱动器X6输入信号有效（低电平有效），电机开始减速到停止，减速距离由DI设定（4000步），减速度500转/秒（DE500）。

循环

两种方式来完成程序的循环。第一种使用QG (Queue Goto) 指令，设置QG参数使程序回到设定的行数。例如：在FL指令后是等待时间0.5秒 (WT0.5)，然后QG指令回到第一行。这样段中的程序会不断循环，每次在QG执行哪后回到第一行。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		DI	40000	
2		AC	500	
3		DE	500	
4		VE	20	
5		FL		
6		WT	0.5	
7		QG	1	
8				
9				

第二种方法是使用QR (Queue Repeat) 指令。QR指令指明跳转到哪一行和循环次数。例如下图中，QR指令表示跳转到第二行，参数为3，即循环次数由用户定义的3寄存器中的值决定，这里RX指令写入5到3寄存器，所以循环次数为5。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		RX	3	5
2		DI	40000	
3		AC	500	
4		DE	500	
5		VE	20	
6		FL		
7		WT	0.5	
8		QR	3	2
9				

跳转

程序跳转由QJ (Queue Jump) 指令实现，跳转和循环不同，跳转主要用于检测状态中，即跳转指令通常和TI (输入检测)，TR (寄存器检测)，CR (寄存器比较) 指令配合使用。

例如，现在有两种可能的运动，正向旋转，当输入信号5有效时 (低电平有效) 电机开始反向旋转。参数设置见下图，加速度300，减速度450，速度18.5，两种运动间有0.25秒的等待时间。然后，检测X5信号的状态，当X5信号有效时 (True)，程序跳转到第10行，开始反向旋转，如果X5信号为高时程序直接到第7行进行正向旋转，执行完正向运动后回到第一行进行循环。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		AC	300	
2		DE	450	
3		VE	18.5	
4		WT	0.25	
5		TI	X5L	
6		QJ	T	10
7		DI	50000	
8		FL		
9		QG	1	
10		DI	-50000	
11		FL		
12		QG	1	

调用

调用和调用子程序相似。QC (Queue Call) 指令允许用户结束一个段的程序，执行另一个段中的程序，然后回到执行调用指令的段。这样把需要多次循环的程序单独放到一个段中进行调用以减少循环的次数和程序的复杂度。例如：

Segment 1				
Line	Label	Cmd	Param1	Param2
1		AC	300	
2		DE	450	
3		VE	18.5	
4		DI	40000	
5		FL		
6		QC	2	
7		VE	1	
8		DI	4000	
9		FL		
10		QC	2	
11		QG	1	
12				

Segment 2				
Line	Label	Cmd	Param1	Param2
1		SO	Y1L	
2		WT	0.25	
3		SO	Y2L	
4		WT	0.25	
5		SO	Y2H	
6		WT	0.25	
7		SO	Y1H	
8		QC		
9				
10				

第二段中设置输出Y1低，等待0.25秒，设置输出Y2低，等待0.25秒。然后设置输出Y2高，等待0.25秒，设置输出Y1高，回到原始段。当第一段中程序运行到QC指令，调用第二段程序，执行完程序后回到第一段中调用行继续执行下面的执行，即第11行。这里，QC指令不加参数表示回到调用指令的原始段。

多任务处理

多任务处理指令允许程序执行运行指令时（FL，FP，CJ，FS等等）执行其他的指令，不需要等待前一个运行指令的结束。不使用多任务处理指令，Q驱动器通常执行指令会等待上一条指令的结束。例如，FL指令后是SO指令，那么驱动器在FL指令结束后才会设置输出。

当多任务处理开启，Q驱动器执行运行指令的同时会执行后面的指令。例如，像上面的FL，SO指令，执行多任务处理，驱动器开始运行并且立即执行输出设置，不需要等待FL指令执行完毕。多任务处理由MT指令设置，MT1-多任务处理开启，MT0-多任务处理关闭。

Segment 1				
Line	Label	Cmd	Param1	Param2
1		MT	0	
2		FL		
3		WT	0.50	
4		SO	Y1L	
5				
6				

Segment 1				
Line	Label	Cmd	Param1	Param2
1		MT	1	
2		FL		
3		WT	0.50	
4		SO	Y1L	
5				
6				

例如上图所示，当MT=1，驱动器执行FL指令，等待0.5秒后设置输出Y1为低电平，不需要等待FL指令结束后再等待0.5秒才输出低电平。

以上是一个最基本的例子，如果您尝试对您的驱动器进行编程，请尽量保证DI值足够大以观察不同的指令执行后的区别。

提示：因为电机不能同时执行两种运动，即使多任务处理开启后运动指令还是会有所妨碍。例如，多任务处理开启后程序中连续有两种运行指令，那么驱动器还是会等待第一条指令完成后执行下一条指令。

MOONS'

上海鸣志自动化设备有限公司

上海市闵行区闵北工业区鸣嘉路168号

201107

0086-21-52634688,62968715

0086-21-52634098,62968682

info@moons.com.cn

www.moons.com.cn

Q 用户指南 (Version Ac)

10/29/04