

USB1010 运动控制卡

软件使用说明书



北京阿尔泰科技发展有限公司

产品研发部修订



目 录

目 录	1
第一章 概述.....	3
第二章 USB1010 的功能和相关技术说明	5
2.1 定量驱动和连续驱动.....	5
2.1.1 定量脉冲输出驱动.....	5
2.1.2 连续脉冲驱动输出.....	6
2.2 速度曲线.....	6
2.2.1 定速驱动.....	6
2.2.2 直线加/减速驱动.....	7
2.2.3 非对称直线加减速驱动.....	8
2.2.4 S曲线加/减速驱动.....	9
2.2.5 脉冲宽度和速度的精度.....	10
2.3 位置管理.....	11
2.3.1 逻辑位置计数器和实际位置计数器.....	11
2.3.2 比较寄存器和软件限位.....	11
2.3.3 位置计数器的可变环形.....	12
2.4 插补.....	12
2.4.1 直线插补.....	12
2.4.2 圆弧插补.....	13
2.4.3 固定线速度.....	14
2.4.4 位模式插补.....	14
2.4.5 连续插补.....	15
2.4.6 步进插补.....	16
2.4.7 加减速驱动的插补.....	16
2.5 原点搜寻.....	18
2.6 中断.....	18
2.7 其它功能.....	18
2.7.1 外部信号控制的驱动操作.....	18
2.7.2 硬件限位(nLMTP(M)).....	18
2.7.3 伺服电机报警信号(nALARM).....	19
2.7.4 伺服电机到位信号 (nINPOS)	19
2.7.5 紧急停止.....	19
2.7.6 脉冲输出类型.....	19
2.7.7 驱动状态输出.....	19
2.7.8 通用I/O信号	20
第三章 库函数驱动程序的使用说明.....	21
3.1 函数调用举例(vc)说明	21
3.1.1 使用USB1010_InitLVDV, USB1010_StartLVDV定长、连续脉冲驱动函数启动电机进行定长驱动.....	21
3.1.2 使用USB1010_InitLVDV、USB1010_StartLVDV定长、连续脉冲驱动函数启动电机进行连续驱动.....	22
3.1.3 使用USB1010_InitLVDV、USB1010_Start2D函数, 启动两轴同时驱动.....	23
3.1.4 使用USB1010_InitCWInterpolation函数, 启动正反方向圆弧插补驱动.....	24

3.1.5 位插补举例.....	25
3.1.6 连续插补举例.....	25
3.1.7 外部信号控制的驱动函数.....	26
3.1.8 设置外部越限信号有效及停止方式.....	27
3.1.9 设置伺服马达输出到位有效.....	28
3.1.10 实际位置计数器例子.....	30
3.1.11 读RR状态寄存器的位状态.....	30
第四章 驱动函数库.....	33
4.1 驱动函数库函数列表.....	33
4.2 驱动函数库说明.....	35
4.2.1 设备对象管理函数.....	35
4.2.2 设置控制卡的基本参数函数.....	36
4.2.3 直线S曲线初始化、启动函数.....	39
4.2.4 直线插补初始化、启动函数.....	39
4.2.5 正反方向圆弧插补初始化、启动函数.....	40
4.2.6 位插补相关函数.....	41
4.2.7 连续插补相关函数.....	42
4.2.8 外部信号启动电机定长驱动、连续驱动.....	42
4.2.9 设置软件限位有效和无效.....	42
4.2.10 设置外部输入信号的有效和无效.....	43
4.2.11 减速函数设置.....	44
4.2.12 读电机状态: 逻辑计数器、实际位置计数器、当前速度、加/减速度.....	45
4.2.13 读状态寄存器的位状态.....	45
4.2.14 设置通用输出和读通用输入.....	46
第五章 硬件参数结构.....	48
5.1 公用参数介绍 (USB1010_PARA_DataList).....	48
5.2 直线和S曲线参数介绍 (USB1010_PAPA_LCDData).....	48
5.3 直线插补和固定线速度直线插补参数介绍 (USB1010_PAPA_LineData).....	49
5.4 正反方向圆弧插补参数介绍 (USB1010_PAPA_CircleData).....	49
5.5 状态寄存器RR0 参数介绍 (USB1010_PARA_RR0).....	50
5.14 状态寄存器RR1 参数介绍 (USB1010_PARA_RR1).....	50
5.15 状态寄存器RR2 参数介绍 (USB1010_PARA_RR2).....	51
5.16 状态寄存器RR3 参数介绍 (USB1010_PARA_RR3).....	51
5.17 状态寄存器RR4 参数介绍 (USB1010_PARA_RR4).....	51

第一章 概述

USB1010 是 USB 总线两轴伺服/步进电机运动控制卡，它以高频率脉冲串形式输出，控制伺服/步进电机的运动。该卡能精确地控制所发出的脉冲频率（电机速度）、脉冲个数（电机转角）及脉冲频率变化率（电机加速度），它能满足步进电机的各种复杂的控制要求。可对电机进行位置控制、插补驱动、加速/减速等控制。具有圆弧、直线插补功能。它含有丰富的，功能齐全的软件库函数资源。在 Windows9X/2000/XP 环境下，用户可直接使用我们为您提供的设备驱动程序函数接口；以最大方便地使您在 Visual C++、Visual Basic 及各种其他软件环境中使用本设备。以下是它的功能特点。

■ 独立 2 轴驱动

USB1010 可以分别控制 2 个马达驱动轴的运动。每个轴都可以进行定速驱动，直线加/减速驱动，S 曲线加/减速驱动等。2 轴性能相同。

■ 速度控制

输出的驱动速度范围是从 1PPS 到 4MPPS(pulses per second 脉冲/秒)。可以运行固定速度驱动，直线加/减速驱动，S 曲线加/减速驱动。加/减速驱动可以使用自动和手动 2 种操作方法。脉冲输出的频率最大误差 $\pm 0.1\%$ （在 CLK=16MHZ 时），驱动脉冲输出的速度可以在驱动中自由变更。

■ 非对称直线加/减速驱动

运行梯形加减速驱动时，加速度和减速度可以设定不同。

■ S 曲线加/减速驱动

每个轴可以用 S 曲线加/减速设定，可以运行对称 S 曲线和非对称 S 曲线。还可以设定为定长输出。加速变化率和减速变化率也可以设定不同，当希望运行对称 S 曲线时，用自动减速功能，当希望运行非对称 S 曲线运动时，要自己设手动减速点。此外，对于定量驱动，我们使用独特的方法避免在 S 曲线加/减速中发生三角波形。

■ 直线插补

可以进行直线插补驱动。插补坐标是从当前位置到-8,388,607~+8,388,607 之间。在整个指定的直线插补范围内，插补精度是 $\pm 0.5LSB$ 。插补速度范围从 1PPS 到 4MPPS。

■ 圆弧插补

可以进行圆弧插补。插补坐标范围是从当前位置到-8,388,607~+8,388,607 之间。在整个指定的圆弧曲线插补范围内，插补精度是 $\pm 0.5LSB$ 。插补速度范围从 1PPS 到 4MPPS。

■ 位模式插补

可以进行位模式插补。收到在高位 CPU 上计算的位模式插补数据后，可以用指定的驱动速度连续输出插补脉冲，用这种方式可以产生任何插补曲线。

■ 连续插补

直线插补→圆弧插补→直线插补→... 这样可以不停地运行每个插补接点的插补驱动，连续插补的最大驱动速度是 2MHz。

■ 步进插补

步进插补驱动是以逐个脉冲输出的方式执行的。有根据命令和根据外部信号 2 种方法。

■ 固定线速度控制

这是一种在插补驱动中保持插补轴合成速度的功能。2 轴同时输出脉冲时，第 2 轴可以设定为 1.414 倍脉冲周期。

■ 位置控制

每个轴都有 2 个 32 位位置寄存器，一个是在芯片内部管理驱动脉冲输出的逻辑位置计数器。另一个是管理从外部编码器来的脉冲的实际位置计数器。

■ 比较寄存器和软件限制功能

每个轴都有 2 个 32 位比较寄存器(COMP+,COMP-),用于跟逻辑位置计数器或者实际位置计数器的大小比较。在驱动时，可以从状态寄存器读出比较寄存器和逻辑/实际位置计数器之间的大小关系。大小关系有变化时，可以产生中断（但要设定中断有效）。并且，可以启动这 2 个比较寄存器作为软件限位。

■ 输入信号滤波器

USB1010 内部的每一个输入信号的输入端都装备积分型的滤波器。可以设定哪一个输入信号的滤波器功能变为有效或无效。滤波器的时间常数从 8 个种类里选择 1 个。

■ 由外部信号驱动

每个轴都可以用外部信号（nEXPP, nEXPM）进行+/-方向运行的定量驱动和连续驱动。这个功能在手动操作时，可以减轻 CPU 的负担。

■ 伺服马达的各种信号

USB1010 接受来自伺服马达驱动器的信号。如 2 相编码器信号，定位信号，报警信号等。

■ **实时监控功能**

在驱动中，可以实时读出逻辑位置计数器、实际位置计数器、加速度、加/减速状态（加速中、定速中、减速中）。

第二章 USB1010 的功能和相关技术说明

2.1 定量驱动和连续驱动

各轴的驱动脉冲输出一般使用正方向或负方向的定量驱动命令或者连续驱动命令。

2.1.1 定量脉冲输出驱动

定量脉冲驱动是以固定速度或加/减速度输出指定数量的脉冲。需要移动到确定的位置或进行确定的动作时，使用此功能。加/减速定量驱动如图 2.1.1 所示，输出脉冲的剩余数比加速累计的脉冲数少时就开始减速，输出指定的脉冲数后，驱动也结束。

进行加/减速的定量驱动，需要设定下列参数：

- 倍率 Multiple
- 加/减速度 Acceleration/ Deceleration
- 初始速度 StartSpeed
- 驱动速度 DriveSpeed
- 输出脉冲数 nPulseNum

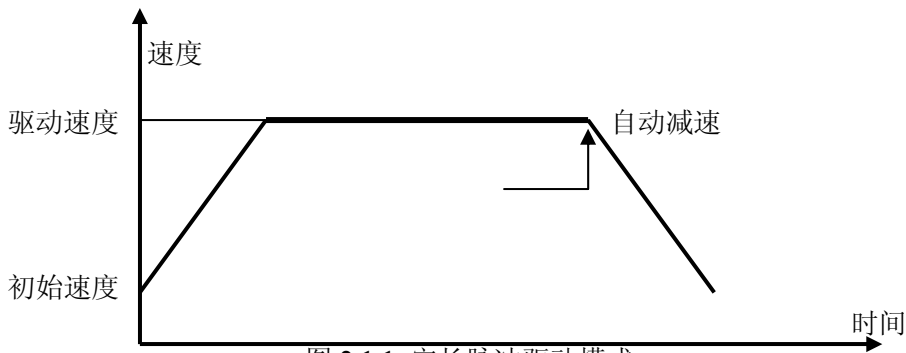


图 2.1.1 定长脉冲驱动模式

■ 在驱动中改变输出脉冲数

在定长脉冲驱动中，输出脉冲数是可以改变的。脉冲输出状况将如图 2.1.2 或图 2.1.3。加/减速驱动中，开始减速时，如果输出脉冲数有变更的话，重新开始加速（如图 2.1.3）。如果变更的输出脉冲数比已经输出的脉冲数要少的话，立即停止（图 2.1.4）。S 曲线减速时输出脉冲有变化的话，不能正确运行 S 曲线动作。

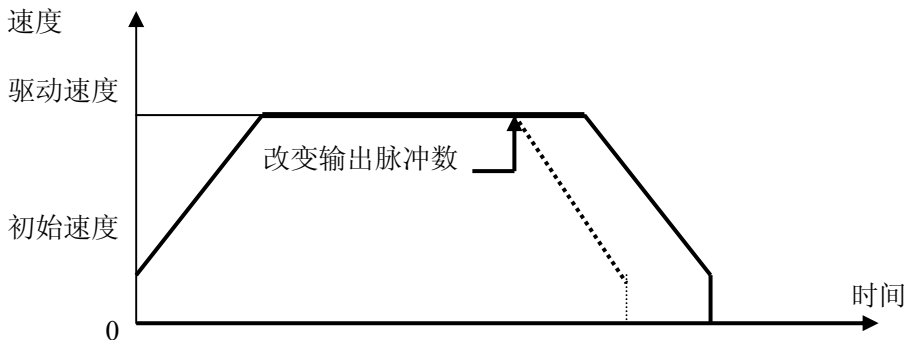


图 2.1.2 在匀速段增加脉冲数

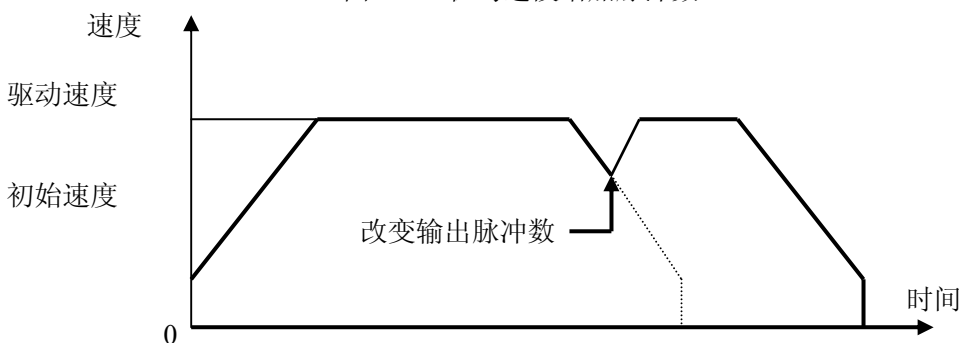


图 2.1.3 在降速段增加脉冲数

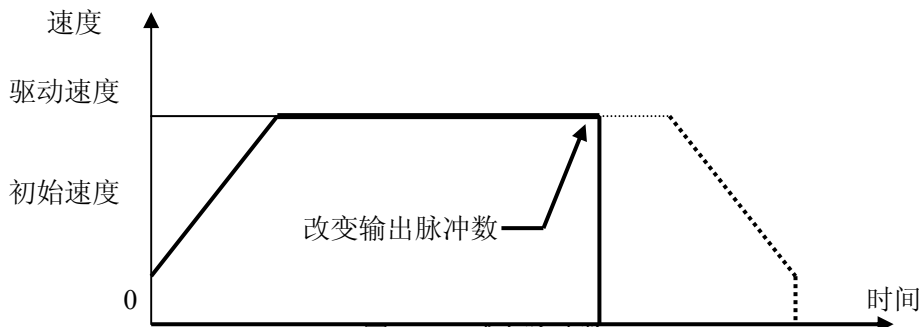


图 2.1.4 减少脉冲数

■ 加/减速驱动的缓冲值设置

用户在定长脉冲驱动情况下可以改变减速点，如图 2.1.1 所示。USB1010 将自动地计算出加/减速点，并且使加速段的脉冲数等于减速段的脉冲数。当为减速设置缓冲值 (shift pulses) 时，USB1010 将会因为缓冲值提前开始减速。减速完成后剩余的脉冲数(shift pulses)将会以初始速度输出，如图 2.1.5。USB1010 初始化时，缓冲脉冲数(shift pulses)的默认值为 8。S 曲线加/减速定量驱动中，如果驱动完毕速度降不到初始速度的话，要把加速计数器偏移值设定为适当的数值，以修正它的速度。

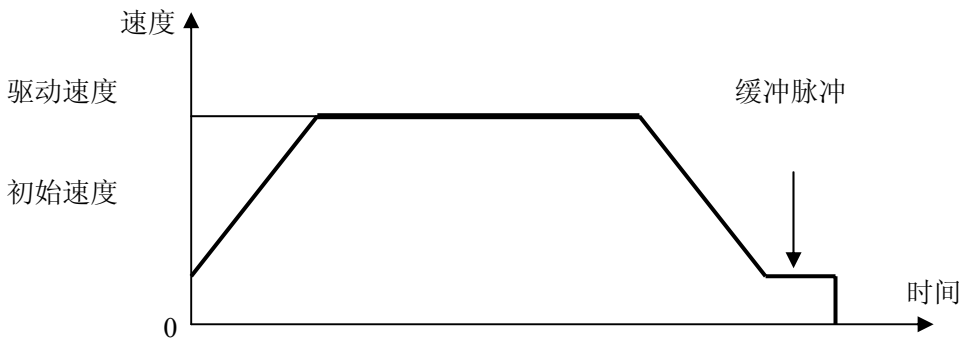


图 2.1.5 定长方式下的缓冲脉冲

2.1.2 连续脉冲驱动输出

当将 USB1010 卡的脉冲输出模式设置为连续驱动状态时，USB1010 将一直以特定的速度驱动脉冲输出直至接收到停止命令或是外部停止信号，如图 2.1.6 所示。

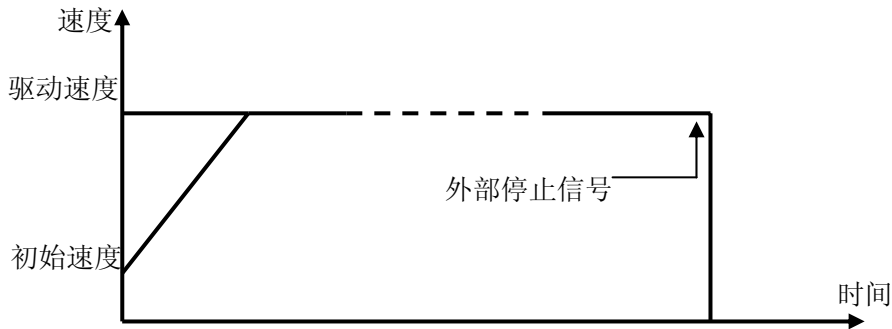


图 2.1.6 连续脉冲驱动

可用“减速至停 DecStop”和“立即停止 InstStop”等函数中断连续驱动脉冲；也可由外部信号使其制动。

2.2 速度曲线

各轴的驱动脉冲输出一般使用正/负方向的定量驱动或连续驱动命令。此外，以设定模式或设定参数来产生定速、直线加/减速、S 曲线加/减速的速度曲线。

2.2.1 定速驱动

定速驱动就是以一成不变的速度输出驱动脉冲及初始速度等于驱动速度。如果设定驱动速度小于初始速度，就没有加/减速驱动，而是定速驱动。使用搜寻原点、编码器 Z 相等信号时，找到信号后马上要立即停止的话，不必进行加/减速驱动，而是一开始就运行低速的定速驱动。如图 2.2.1。

为了定速驱动，需要设定下列参数（初始速度=驱动速度）：

- 倍率 Multiple

如图 2.2.4, 从实际初始速度 500 PPS 加速至 15 000 PPS, 时间为 0.3 S (秒)则

最高输出频率的倍数 Multiple=2 (初始化后默认值为 M=1)

初始速度 StartSpeed =500 PPS/M=250 PPS

驱动速度 DriveSpeed =15000PPS/M=7500 PPS

加速度 Acceleration =[(15000 - 500)PPS /0.3S]/M=24167 PPS/S, 实际取值 24125 PPS/S

减速度 Deceleration =[(15000 - 500)PPS /0.1S]/M=72500 PPS/S

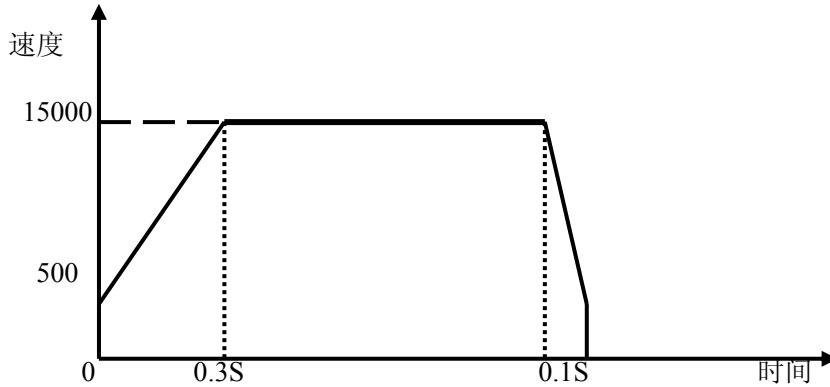


图 2.2.4 定长脉冲驱动模式加/减速度不同

注意: 当加速度>减速度时, 加速度和减速度比率要满足如下条件: 减速度>加速度 × (驱动速度) /4×10⁶

当加速度和减速度的比率越大, 则在减速时, 拖延的脉冲越多, 即减速点会前移。为了解决这个问题可以增大初始速度或设置加速计数器偏移来解决。比如加速度和减速度的比率为 10, 减速点大概会提前 10 个脉冲左右, 即会提前 10 个脉冲开始减速, 这样当减速完成达到初始速度时, 还会有 10 个左右的脉冲才会达到定长脉冲数。这 10 个脉冲会以初始速度输出直到输出定长脉冲数。如果初始速度是 1PPS/S 的话, 就要等 10 秒钟才会结束。把初始速度增大则等的时间会变小。或者设定加速计数器偏移为 0 或负值来调整减速点。

2.2.3 非对称直线加减速驱动

往垂直方向移动对象物时, 对对象物有重力加速度的负担, 所以在这样加速度和减速度不同的非对称直线加减速的定量驱动中, 最好变更上下移动的加速度和减速度。此 IC 可以运行自动减速, 事先不用设定手动减速点。图 2.2.5 是减速度比加速度大的例子, 图 2.2.6 是加速度比减速度大的例子。在这样的非对称的直线加减速中数据输出脉冲数 (P) 和各速度参数可以在 IC 内部计算减速开始点。

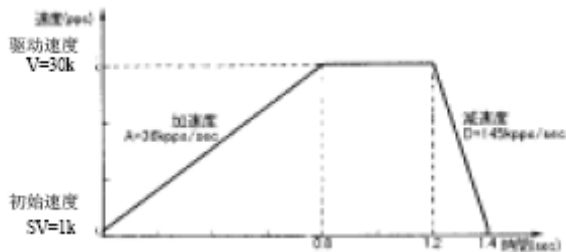


图 2.2.5 非对称直线加减速驱动(加速度<减速度)

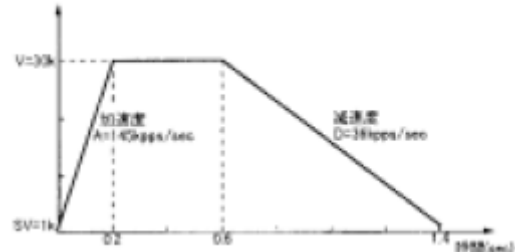


图 2.2.6 非对称直线加减速驱动(加速度>减速度)

在非对称直线加减速的定量驱动中需要运行自动减速的话, 为了把减速度设定数值变为有效, 要把 WR3 寄存器的 D1(DS~DE)设定为 1。为了使加减速定量驱动中自动运行减速, 把 WR3 寄存器的 D0(MANLD)设定为 0。

WR3/D1(DSNDE) 1 : 减速度设定值有效

WR3/D0(MANLD) 0 : 把加减速定量驱动中的减速运行自动

另外, 跟通常的直线加减速驱动一样, 需要设定下述的参数:

- 倍率 Multiple
- 加速度 Acceleration
- 减速度 Dcceleration
- 初始速度 StartSpeed
- 驱动速度 DriveSpeed
- 输出脉冲数 nPulseNum

注意:

- 加速度>减速度(图 2.2.6)时, 加速度和减速度的比率有下述的条件。

减速度 > 加速度 X (驱动速度 / 4x10⁶) (CLK=16MHz 时)

比如, 驱动速度 V=100KPPS 的话, 减速度 Dcceleration 要大于加速度 Acceleration 数值的 1/40, 不能小于

1/40。

- 加速度>减速度(图 2.2.6)时, 加速度 Acceleration 和减速度 Dcceleration 的比率越大延迟脉冲越多 (Acceleration/Dcceleration=10 倍时, 最大 10 脉冲左右。为了改善延迟脉冲问题, ①增加初始速度, ②在加速计数器偏移上设定负数值。

2.2.4 S 曲线加/减速驱动

驱动速度加/减速时, ART1010 线性地增加/减少加速度/减速度, 以产生 S 型速度曲线。S 曲线加/减速驱动如图 2.2.7 所示。

驱动开始加速时, 加速度以指定的加速度的增加率 (AccelerationK) 从 0 线性增加到指定的加速度值 (Acceleration), 因此, 这个速度曲线成为二次抛物线 (a 区间)。加速度达到指定数值 (Acceleration) 后保持此数值。这时速度曲线是直线型的, 速度在加速中 (b 区间)。目标速度及驱动速度和当前速度的差值比相应时间增加所增加的速度少时, 加速度趋向 0。当减速时和加速时一样, 减速度以指定的增加率 (AccelerationK) 增大到减速度值, 然后减速度保持一段时间不变, 最后减速度减少直到 0。这样具有部分固定加速度的加速为部分 S 曲线。

另一方面, 在 a 区间若在加速度达到指定数值 (Acceleration) 前, 目标速度 (DriveSpeed) 和当前速度的差值比相应时间增加所增加的速度少时, b 区间就消失, 只有 a 和 c 区间。这种没有固定加速度的加速称为完全 S 曲线加速。

同样, S 曲线加/减速运动也分为对称的和不对称的运动两种。当要运行对称的 S 曲线运动时, 要设定下列参数:

- 倍率 Multiple
- 加速度/减速度的变化率 AccelerationK
- 加速度 Acceleration
- 初始速度 StartSpeed
- 驱动速度 DriveSpeed
- 输出脉冲数 nPulseNum 定量驱动时使用

■ 参数设定例子 1 (对称的完全 S 曲线)

如右图所示, 是在 0.4 秒内用完全 S 曲线加速增加至 40KPPS 的例子。

首先, 在计算上不考虑初始速度 (把它当作 0)。因为是完全 S 曲线加速, 所以在 0.4 秒的 1/2 (0.2 秒) 把速度增加至 40KPPS 的 1/2 (20KPPS), 在剩下的 0.2 秒增加至 40KPPS。这时加速度线性的增加直至 0.2 秒。在右下图, 加速度 Acceleration = X (直线斜率) × t (时间), 已知初始速度是 0, 0.2 秒后为 20K, 而加速度又是速度的导数, 所以, $dv/dt = X \times t$, 计算得 $X = 1000K$, 0.2 秒的加速度是 $100,000 \times 0.2 = 200KPPS/SEC$, 加速度的增加率是 (AccelerationK) $200K/0.2 = 1,000KPPS/SEC^2$ 。

运行完全 S 曲线加/减速时, 速度取决于加/减速度的变化率, 所以为了避免产生部分 S 曲线, 加/减速度要设定 200KPPS/SEC 以上的数值。

- 倍率 Multiple = 10;
- 加/减速度的增加/减少率 AccelerationK = 100,000; $100,000 \times 10$ (倍率) = 1000K
- 加速度 Acceleration = 20,000; $20,000 \times 10$ (倍率) = 200K
- 初始速度 StartSpeed = 100; $100 \times 10 = 1000$
- 驱动速度 DriveSpeed = 4000; $4000 \times 10 = 40,000$

当要运行非对称的 S 曲线运动时, 要设定下列参数:

- 倍率 Multiple

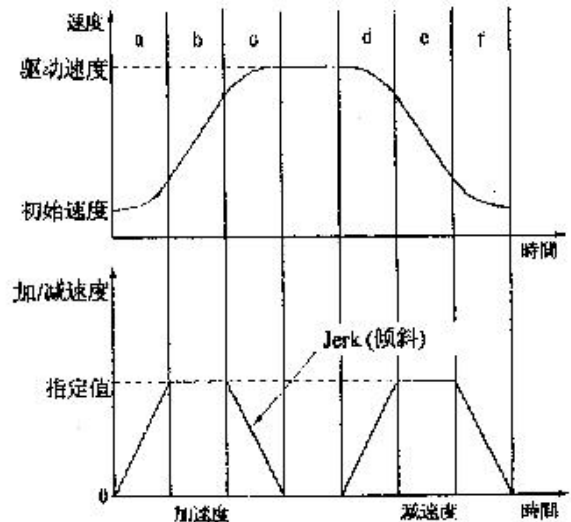
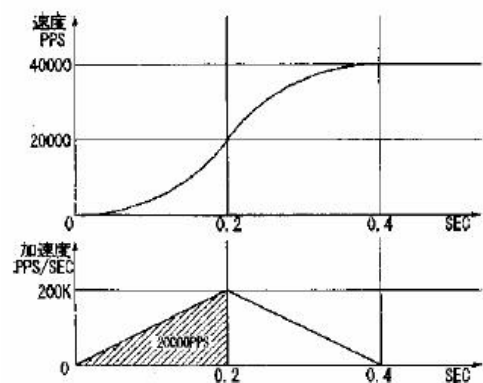


图 2.2.7 S 曲线加减速驱动



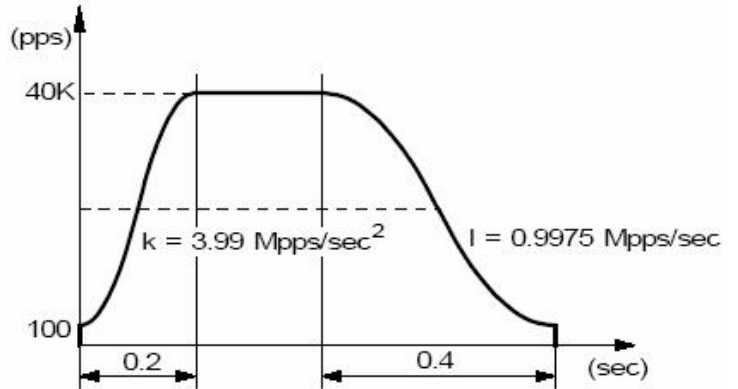
- 加速度的变化率 AccelerationK
- 减速度的变化率 DecIncRate
- 加速度 8000(必须设置为 8000)
- 减速度 8000(必须设置为 8000)
- 初始速度 StartSpeed
- 驱动速度 DriveSpeed
- 输出脉冲数 nPulseNum 定量驱动时使用

当运行非对称 S 曲线运动时不能自动减速，要手动设置减速点。

■ 参数设定例子 2（非对称的完全 S 曲线）

如右图所示,加速时在 0.2 秒内用完全 S 曲线从初始速度 100PPS 加速增加至 40KPPS, 减速时用 0.4 秒从 40K 减速到初始速度的例子。

首先, 因为是完全 S 曲线加速, 所以在 0.2 秒的 1/2 (0.1 秒) 把速度增加至 40KPPS 的 1/2 (20KPPS), 在剩下的 0.1 秒增加至 40KPPS。在右下图, 加速度 Acceleration =X (直线斜率) ×t (时间), 已知初始速度是 1000, 0.1 秒后为 20K, 而加速度又是速度的导数, 所以, dv/dt=X×t, 计算得 X=3.99M, 加速度的增加率是(AccelerationK)3.99MPPS/SEC²; 同理, 我们可以算出减速度变化率 (DecIncRate) 是 0.9975MPPS/ SEC²。则减速点是: nPulseNum - (初始速度+驱动速度)* (√驱动速度-初始速度) 具体程序请参考例子。

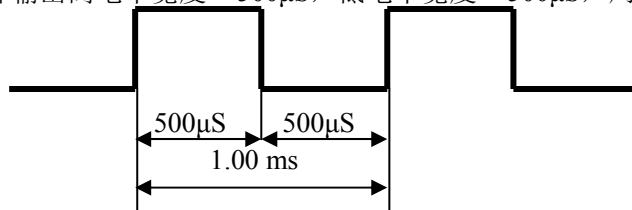


2.2.5 脉冲宽度和速度的精度

■ 驱动脉冲的脉冲比率

对于各轴正/负方向上的驱动脉冲, 由驱动速度决定的脉冲周期时间有±1SCLK (CLK = 16MHz 时为 ±125nSEC) 的误差, 并且基本上分布在高电平 50%和低电平 50%。举例如下图所示, 设定倍率为 1, 驱动速度 1000PPS, 驱动脉冲输出高电平宽度 =500μS, 低电平宽度 =500μS, 周期=1.0mS。

Multiple =1
StartSpeed =1000 PPS
DriveSpeed =1000 PPS



当处于加速时, 低电平脉冲长度小于高电平脉冲长度; 驱动速度将会提高; 反之, 当处于减速时, 低电平脉冲长度大于高电平脉冲长度; 驱动速度将会降低。

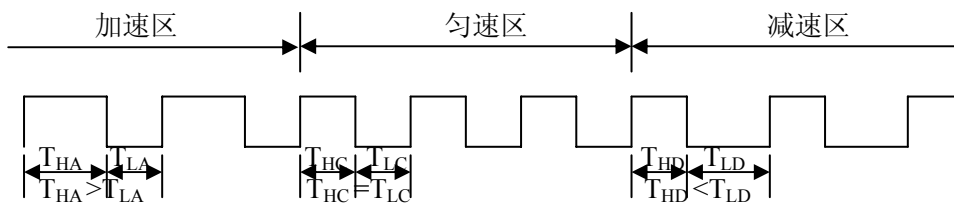


图 2.2.8 加/减速过程中脉冲宽度的比较

■ 驱动速度精度

因为 USB1010 的 CLK 时钟为 16MHz, 产生驱动脉冲的电路是由 IC 内 SCLK 来驱动的。SCLK 由时钟信号(CLK)2 分频而得。若 CLK 输入是标准的 16MHz, SCLK 就是 8MHz。要产生某个频率的驱动脉冲, 并且是没有抖动的均匀频率驱动脉冲, 它的频率周期只能是下表所示 SCLK 周期的整数倍。

USB1010 初始化后, 最高速度 (输出脉冲频率) 默认值为 8K。驱动速度越高, 精度越低。即使是驱动速度很高, USB1010 仍就能保持相对的精度; 驱动脉冲的精度仍在±0.1 之内。不会影响驱动电机的工作状态, 因为这个误差是会被电机系统的惯性吸收的。

最高输出脉冲频率可以通过 SetM 函数设置, 默认值 M=1, 默认最高输出脉冲频率 8K。

倍数 M	最高输出频率/PPS
M=1	MAX(V)=8000

M=2	MAX(V)=16,000
M=3	MAX(V)=24,000
...	...
M=500	MAX(V)=4,000,000

当 M 被设置为 500 时，最高输出频率为 4M。加/减速度也随着 M 的数值变化而变化。

2.3 位置管理

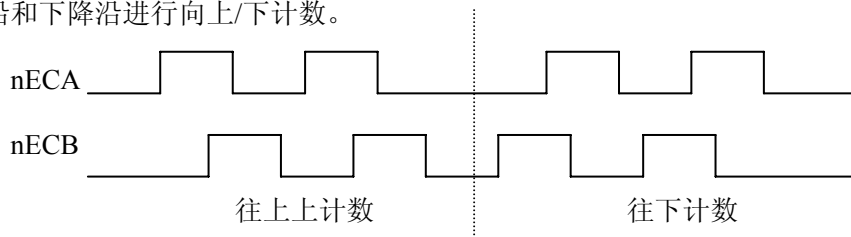
2.3.1 逻辑位置计数器和实际位置计数器

USB1010 对每一个轴都有一个逻辑位置计数器和实际位置计数器。逻辑位置计数器计数是计数 USB1010 卡发出的正/负方向输出脉冲。当发出一个正向脉冲时，计数器将自动加 1，当发出一个负向脉冲时，计数器将自动减 1。

实位计数器计数来自外部编码器的输入脉冲，输入信号可以设定为 2 相脉冲输入或上/下脉冲输入。

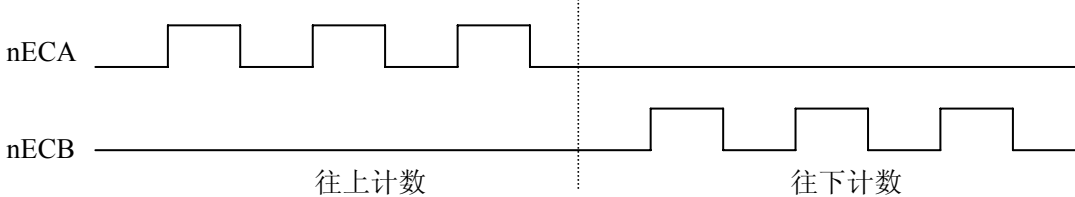
■ 2 相脉冲输入模式

当设定为 2 相脉冲输入后，按正逻辑当 A 相超前时向上计数，当 B 相超前就向下计数。由 2 个信号的上升沿和下降沿进行向上/下计数。



■ 下脉冲输入模式

nECA 作为向上计数输入，nECB 作为向下计数输入，由脉冲的上升沿计数。



USB1010 可以在任何时候写入或读出 2 个计数器的数据，我们提供了 USB1010_ReadLP, USB1010_ReadEP 函数分别 读出逻辑计数器和实位计数器的值。计数范围在-2,147,483,648~+2,147,483,647 之间。

2.3.2 比较寄存器和软件限位

USB1010 对每一个轴都有 2 个 32 位寄存器（上下限位寄存器 COMP+, COMP-）用来与逻辑位置计数器或实际位置计数器进行比较。把 2 个比较寄存器的比较对象设定为逻辑计数器还是实位计数器，可由函数指定，COMP+寄存器主要用来检测逻辑/实位计数器计数的上限。当逻辑/实位计数器的值大于 COMP+寄存器的值时，USB1010 的 RR1 寄存器的 D0 位就置 1。另一方面，COMP-寄存器用来检测逻辑/实位计数器某个范围的下限。当逻辑/实位计数器数值小于 COMP-寄存器的数值时，RR1 寄存器的 D1 位就置 1。

可以把 COMP+ 寄存器和 COMP- 寄存器用于正/负方向的软件限制来运行。我们提供 USB1010_SetPDirSoftwareLimit 函数供选择是逻辑寄存器还是实位寄存器，并设定正方向软件限位有效，在驱动中，如果逻辑/实位计数器的值大于 COMP+的值就执行减速停止，并且 RR2 寄存器的 D0 为 1；USB1010_SetMDirSoftwareLimit 用于设定反方向软件限位，并选择是逻辑寄存器还是实位寄存器，当执行负方向驱动命令并且逻辑/实位计数器的值小于 COPM+寄存器后，就会清除这个状态。同样情况适应于负方向的 COMP-。可以在任何时候调用 USB1010_SetPDirSoftwareLimit, USB1010_SetMDirSoftwareLimit 写 COMP+寄存器和 COMP-寄存器。复位时寄存器的值是任意的。

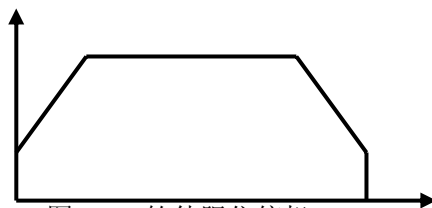


图 2.3.1 软件限位停机

2.3.3 位置计数器的可变环形

逻辑位置计数器及实位数器是 32 位长的上/下环形计数器,因此从 32 位长的最大数值 FFFFFFFFh 往+方向计数的话,最后计数计到 0。从 0 往-方向计数的话,最后计数计到 FFFFFFFFh。可变环功能是可以把这个环形计数器的最大数值任选设定,如果定位不是直线,而是旋转运动的话,用此功能控制位置很方便。要使可变环功能有效,把 WR3 寄存器的 D6 (RING) 位设定为 1。在 COMP+寄存器设定逻辑位置计数器的最大数值,在 COMP-寄存器设定实位计数器的最大数值。

比如,用 10,000 脉冲旋转一圈的话。

- ① 设定 WR3/D6 位为 1 把可变环功能就有效。
- ② 在 COMP+寄存器设定 9,999(270Fh—逻辑位置计数器的最大数值)。
- ③ 同时使用实位计数器的话,在 COMP-寄存器设定 9,999 (270Fh—逻辑位置计数器的最大数值)。

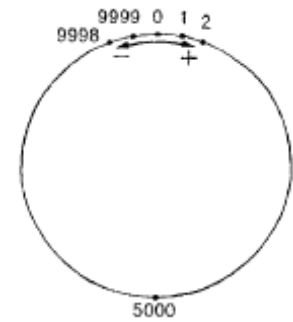
这时计数动作:

往+方向,上计数时、...→9,998→9,999→0→1→...

往-方向,下计数时、...1→0→9,999→9,998→...

注意:

- 每个轴都要设定可变环功能的有效/无效,但是不能分别设定逻辑位置计数器和实位计数器的有效/无效。
- 使可变环功能有效后,不能使用软件限制功能。



位置计数器的可变环最大值 9.999 操作

2.4 插补

USB1010 可实现 2 轴的直线插补,圆弧插补和位插补。插补运动是指 2 轴按照一定的算法进行联动,被控轴同时启动,并同时到达目标位置。对于直线插补,圆弧插补,位插补,最大驱动速度为 4 MPPS,连续插补最大驱动速度是 2MPPS。

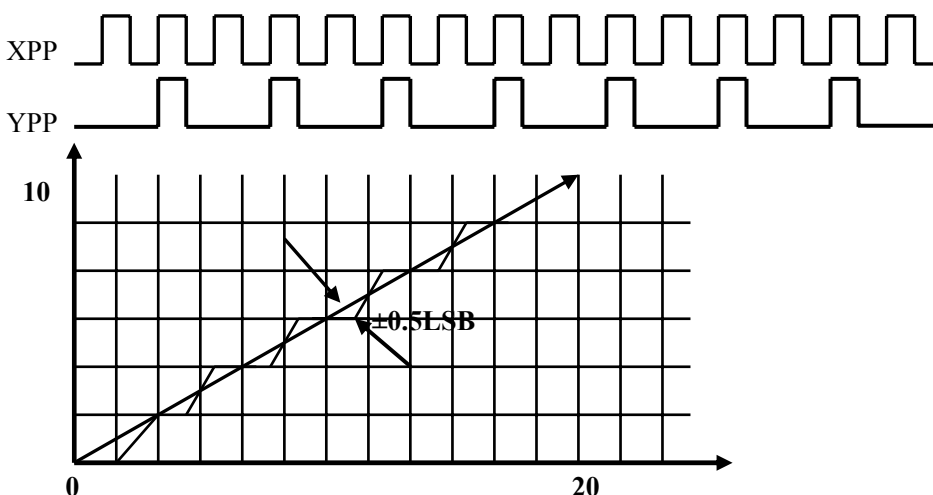
插补驱动时,每个驱动轴都能进行硬件限制和软件限制。在插补驱动中任何轴的限制有效,USB1010 停止插补。

在插补中,最长的移动距离的轴为“长轴”,另外一个轴为“短轴”。“长轴”输出一个均匀的脉冲序列,而“短轴”的驱动脉冲依赖于“长轴”和 2 轴之间的关系,有时候输出脉冲,有时候不输出脉冲。

2.4.1 直线插补

当选择 X, Y 轴直线插补时,从当前位置到相对位置 (X: +20, Y: +100) 如下图所示,从当前坐标执行直线插补,终点坐标由针对当前位置的相对数值设定。精确设定每个轴的输出脉冲数。在每个轴独立运行时,输出脉冲数设定为没有符号的数值。但是,在插补驱动时,用相对数值设定当前位置的终点坐标。如下图所示,对指定直线的位置的位置精度,在整个插补范围内有±0.5LSB。下图是直线插补驱动脉冲输出例子,在设定的终点数值中绝对值最大的是长轴。在插补驱动中,此轴一直输出脉冲,其它的轴是短轴,根据直线插补算术的结果,有时候输出脉冲,有时不输出脉冲。

直线插补的坐标范围是带符号的 24 位字长。插补范围为从各轴当前位置到-8,388,607~+8,388,607 之间(注意:不能设定-8,388,608)



2.4.2 圆弧插补

圆弧插补从当前位置开始，根据所指定的圆心和终点位置以及插补的方向(按顺时针或逆时针)来进行。坐标设定值是对当前坐标(始点)的相对值(并且是脉冲数)。图 2.4.1 说明了顺时针和逆时针插补的定义。

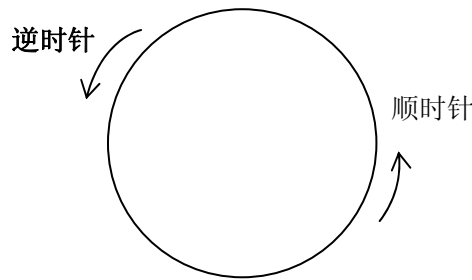


图 2.4.1

设定相对当前位置始点的圆弧中心坐标及终点坐标，在写入 CW 圆弧插补命令或 CCW 圆弧插补命令后执行圆弧插补。用当前坐标始点的相对值设定中心坐标及终点坐标。

CW 圆弧插补从当前坐标至终点坐标以顺时针方向绕中心坐标画圆弧。CCW 圆弧插补以逆时针方向绕中心坐标画圆弧。如果终点设为 (0, 0) 能画整个圆。

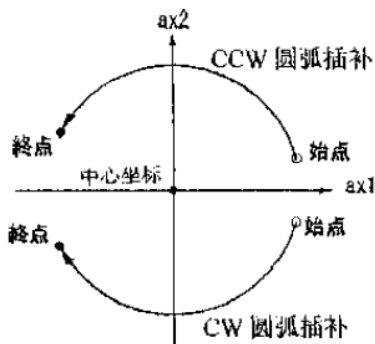


图 2.4.2

至于圆弧插补的算法，如图 2.4.3 所示，由 X 轴和 Y 轴定义一个平面，绕中心坐标把它分为 0~7 的 8 个象限。如图所示，在 0 象限的插补坐标 (X, Y) 上，Y 绝对值一直比 X 的绝对值小，绝对值小的轴为短轴，1、2、5、6 象限是 X 轴，0、3、4、7 象限是 Y 轴，短轴在这些象限之间一直输出驱动脉冲，长轴根据圆弧插补运算结果，有时输出脉冲，有时不输出脉冲。

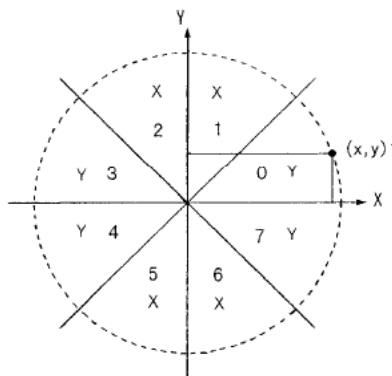


图 2.4.3

图 2.4.4 是在当前坐标上指定中心(-11,0)，终点(0, 0)，画半径 11 完整圆的例子。此外，图 2.4.5 是那个时候输出脉冲的例子。

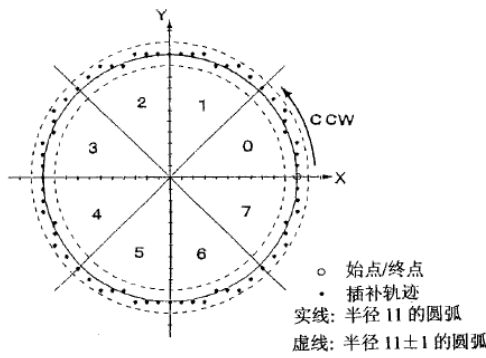


图 2.4.4

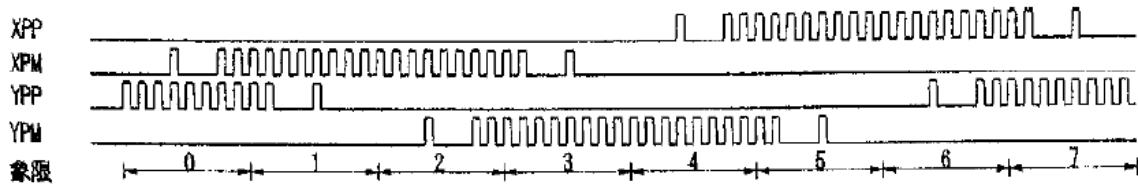
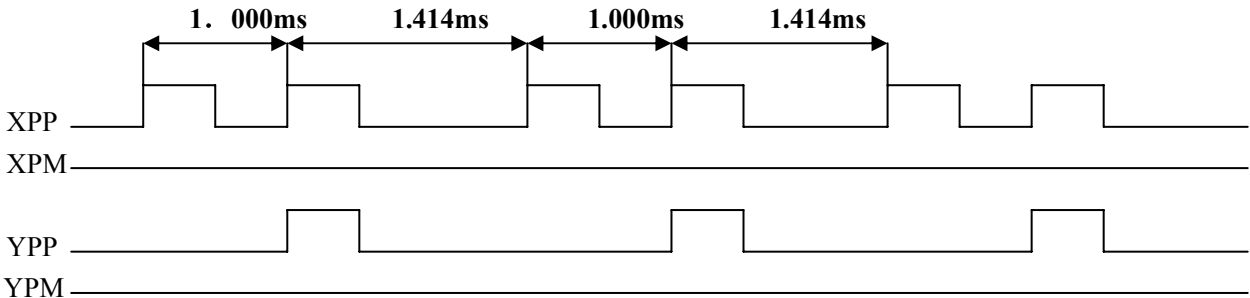


图 2.4.5

2.4.3 固定线速度

固定线速度控制是始终保持插补轴以合成速度运行的功能。每个轴根据主轴的基本脉冲输出驱动脉冲。如图所示，当在 X，Y 的 2 个轴上输出驱动脉冲时，移动的距离比只用 1 个轴的长 1.414 倍。因此，若需要保持 2 个轴始终一致的合成速度，其中 1 个轴的速度要设定为另 1 个轴当它作为单轴输出驱动脉冲速度的 1/1.414。

即如果设定初始速度是 1000PPS，驱动速度是 1000PPS，则插补时的合成速度始终是 1000PPS。例如设定插补终点为 (20, 10)，初始速度 1000PPS，驱动速度 1000PPS，脉冲输出如下图所示。



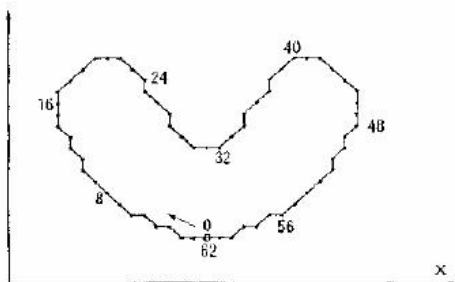
为了达到 (20, 10) 终点，必须是 X 轴输出两个脉冲，Y 轴输出一个脉冲。当 X 轴输出脉冲，Y 轴不输出时，X 轴脉冲频率为 1000PPS，当两轴同时都有脉冲时，两轴的脉冲周期都为 1.414ms，及频率为 $1000 * (\sqrt{2}/2) = 707$ ，则两轴的合成速度为 1000PPS，这样就保证了整个过程中的速度都为 1000PPS。

注意：2 个轴输出驱动脉冲时，脉冲周期延长到原来的 1.414 倍。驱动脉冲的高电平宽度是不变的，只有低电平在一个周期内比原有的宽 1.414 倍（驱动脉冲时要设定为正逻辑）。

2.4.4 位模式插补

位模式插补是把高位 CPU 计算的插补数据以数据包的方式接收后，以指定的驱动速度连续输出插补脉冲。在插补初始化时要事先指定主轴，副轴。

主轴正，负方向，副轴正，负方向要输出一个脉冲时，设定为 1，不输出脉冲时，设定为 0。这样设定的话，要画如图所示的轨迹，要输出的数据如下所示。



64 位	56	48	40	32	24	16	8	0
0100,0000	0000,0000	000,11111	1101,1011	1111,0110	1111,1110	0000,0000	0000,0000	:XPP
0111,1111	1111,0101	0000,0000	0000,0000	0000,0000	0000,0000	0010,1011	1111,1111	:XPM
0000,0000	0000,0000	0000,0000	1111,1111	0000,0000	0000,1111	1111,1111	1101,0100	:YPP
0000,1010	1111,1111	1111,1100	0000,0000	0011,1111	1100,0000	0000,0000	0000,0000	:YPM

BP1P 寄存器，BP1M 寄存器是从高位 CPU 写入位模式数据的 16 位寄存器，X 轴正方向的 16 位数据写入 BP1P 寄存器，X 轴负方向的数据写入 BP1M 寄存器，位模式插补开始后，从 D0 位依次输出驱动脉冲。

堆栈计数器 (SC) 是计算位模式数据存储量的计数器，能从 0 到 3 变化。给位数据堆栈写入一个 16 位数据，则增加 1。SC 为 3 的时候，表示位数据堆栈不能再补充数据。为 2 的时候，可以再补充一个 16 位数据。为 1 时，可以再补充 2 个 16 位数据。为 0 时，表示输出了所有数据，驱动结束。

在插补驱动开始后，随着驱动脉冲输出，SC 的数值以 3 → 2 → 1 减少，所以，可以重新写数据。要连续运行位模式插补的话，在 SC 为 2 或 1 时，要设定下一个数据。SC 数值从 2 变到 1 时，也可以要求中断高位 CPU，以写入数据。

■ 插补驱动速度的限制

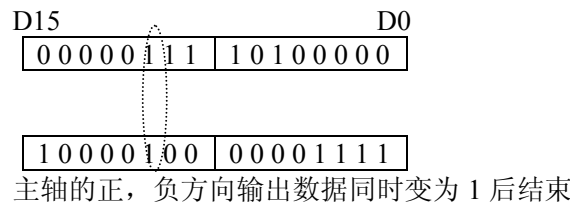
USB1010 的位模式插补驱动速度最高达 4MHz。但是，数据超过 48 位的话，CPU 要在插补驱动中补充数据，所以插补驱动速度将受到 CPU 建立模式数据所需时间的限制。

比如，对于 2 轴位模式插补，如果 CPU 花费在 64 位数据运算和数据建立的时间以及执行 BP 数据堆栈命令的时间是 100μs，则插补驱动速度在 $1 / (100\mu s / 16) = 160KPPS$ 以下。

■ 位模式插补的结束

位模式插补以下 2 种方法结束

① 当主轴 (X 轴) 正和负方向的输出位数据都为 1 时，位模式插补就结束。收到结束码后，堆栈计数器 (SC) 被强制为 0。所有遗留的位模式数据都无效。



② 停止数据写入

停止向 BP 数据堆栈写入数据，当所有的位模式数据作为驱动脉冲输出后，SC 为 0，结束插补驱动。

■ 用停止命令中断插补驱动

对运行位模式插补驱动的主轴写入立即停止或减速停止命令，插补驱动就停止。重新启动位插补，就继续为模式插补。如果以停止命令停止驱动而不需要再进行插补的话，用 USB1010_ClearBPData 函数把 BP 寄存器所遗留的数据都清除。

■ 用硬件限制和软件限制的停止

在插补驱动中任何轴有硬件限制，软件限制动作，插补驱动就停止。若不需要再运行插补，就一定要用 USB1010_ClearBPData 函数所有留下的数据清除。

位模式插补例子请参看《[位插补举例](#)》。

2.4.5 连续插补

连续插补是直线→圆弧插补→直线插补→……等等这样在每个插补节点之间不停地驱动，连续插补。在连续插补驱动中，如果设定了下一个插补参数并写入插补命令，就能执行连续的插补驱动。因此，在所有的插补节点中，从连续插补驱动开始至结束的时间必须长于设定下一个插补节点数据和发命令的时间。

RR0 寄存器的 D9 (CNEXT) 位用于连续插补。在插补驱动中这个位表示是否可写入下一个插补节点数据及插补命令，1 表示可以写入，0 表示不可以写入。当驱动停止时，该位为 0；插补驱动开始就变为 1，可以写入下一个插补节点的数据及插补命令。写入下一个插补节点的插补命令后变为 0 (不可以写入状态)。等到这个插补开始后，变为 1，可以写入再下一个插补节点的数据及插补命令。

■ 连续插补的注意事项

- 在每个插补节点上要在设定需要的数据后，发插补命令，请不要先送插补命令后送数据。
- 连续插补的驱动速度最高达 2MHz。
- 所有驱动插补节点的时间必须长于插补轴检错和设定下一个插补节点的数据及命令的时间。如果在设定下一个插补节点时驱动停止，RR0 寄存器的 D9 位为 0，这时在写入下一个插补节点的命令后就从暂停处继续插补。

- 连续插补中有圆弧插补时，圆弧插补终点的短轴数值也许会比真值偏差±1LSB，因此，为了避免累积每个节点的误差，事先要确认每个圆弧插补的终点，然后考虑怎么运行连续插补。

连续插补例子请参看《[连续插补举例](#)》。

2.4.6 步进插补

步进插补驱动的功能是以逐个脉冲的方式执行的。有根据命令和根据外部信号这 2 种方法。如果使用外部信号，就可以运行跟外部信号同步的插补驱动。步进插补时，插补主轴设定为定速驱动。从每个轴输出的驱动脉冲高电平宽度是 1/2 脉冲周期，这周期由在主轴设定的驱动速度而定。低电平宽度的增加直到下一个命令或外部信号的到来。下图是由外部信号驱动的步进插补的例子。例如把主轴的初始速度设定为 500PPS，把驱动速度设定为 500PPS 的定速驱动，那么，输出的驱动脉冲的高电平宽度是 1mSEC。



图 2.4.6 根据外部信号(EXPLSN)的步进插补例子(驱动速度: 500PPS)

■ 根据命令的步进插补

单步插补命令用于插补驱动的步进传送。在设置好根据命令步进插补，并设定好速度，插补数据（终点，圆心）启动插补命令后，当发一次步进插补命令则输出一个插补脉冲。以下是操作程序。

- ① 设置根据命令进行步进插补
- ② 插补驱动初始化。用相同数值设定插补主轴的初始速度和驱动速度
用相同数值设定初始速度和驱动速度将执行定速驱动，这个速度值要设定得比写入单步命令周期快。比如，要用最高 1mSEC 周期写入单步命令，就要把初始速度和驱动速度设定得比 1000PPS 快的数值

③ 启动插补命令

④ 写入单步插补命令

根据插补运算的结果，从各轴输出驱动脉冲。不断写入单步命令，直到插补驱动结束为止。若要在步进插补中停止，也可给主轴写立即停止命令（USB1010_InstStop），等待一个以上脉冲周期后，再写单步插补命令，驱动就停止。插补驱动结束后，写入的单步插补命令都无效。

■ 外部信号控制的步进插补

引脚 nMPLS 用作步进插补驱动的外部输入信号。nMPLS 信号通常是高电平。在插补步进模式时，步进插补由外部信号的下降沿启动。以下是操作程序。

- ① 设置根据外部信号启动步进插补
- ② 对插补驱动进行初始化，把初始速度和驱动速度设定相同值。
这个速度值和上节的命令控制一样。设定的速度值要比 nMPLS 的低电平脉冲周期快。

③ 启动插补驱动

④ 给 nMPLS 引脚输入低电平脉冲

脉冲下降 2~5CLK 后，插补驱动脉冲从各轴输出。EXPLS 的低电平脉冲宽度需要在 4CLK 以上。若要在步进插补中停止，就要给主轴写立即停止命令，等待一个以上驱动的脉冲周期后，再输入 MPLS 的低电平脉冲，驱动就停止。插补驱动结束后，向 MPLS 端输入的低电平脉冲无效。

注意：如果是由机械接点产生 MPLS 低电脉冲，要把滤波器功能变为有效，解除颤动。

2.4.7 加减速驱动的插补

插补一般用定速驱动，不过 USB1010 可以用直线加/减速驱动或 S 曲线加/减速驱动（只可做直线插补）运行插补。

在连续插补时为了实现加/减速驱动，使用减速有效命令和减速无效命令。在插补驱动时减速有效命令是使自动减速或手动减速变为有效，减速无效命令是使它变为无效。复位时，都是无效状态。在用加/减速单独运行插补驱动时，驱动之前一定要设定成减速有效状态。在驱动中写入减速有效命令，也不能变为有效。

■ 2 轴直线插补的加/减速驱动

在 2 轴直线插补中可以运行直线加/减速驱动及 S 曲线加/减速驱动减速，这时自动减速和手动减速都可以使用。使用手动减速时，把在终点坐标的各轴数值中绝对值最大的数值设定为 X 轴的手动减速点。比如：

运行 2 轴直线插补到终点 (X: -20000, Y: 60000)。假定减速时需要的脉冲数是 5000, Y 轴的终点绝对值比 X 轴大, 所以把 $60000-5000=55000$ 作为手动减速点设定 X 轴。

■ 圆弧插补, 位模式插补的加/减速驱动

在圆弧插补, 位模式插补中只能用手动减速的直线加/减速驱动, 不能使用 S 曲线加/减速驱动及自动减速。

在圆弧插补中不能用自动减速, 所以事先要设定手动减速点。半径 10,000 的圆通过从 0 至 7 象限, 在每一个象限上短轴一直输出脉冲, 所以短轴每一个象限输出 $10,000/\sqrt{2}=7,071$ 脉冲。因此在整个圆上从主轴输出的基本脉冲数是 $7,071 \times 8=56,568$ 。

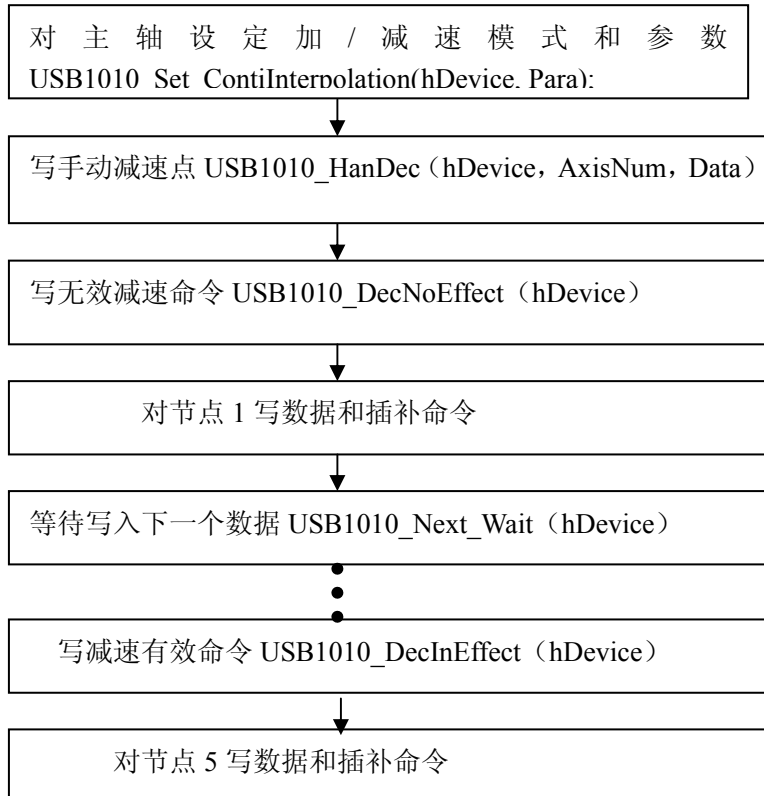
此外, 把初始速度设定为 500PPS, 在 0.3 秒之内用直线加速把驱动速度增加至 20,000PPS 的话, 加速度是 $(20,000-500)/0.3=65,000\text{PPS/S}$ 。加速时花费的脉冲数是右图的斜线部分面积 $(500+20,000) \times 0.3/2=3,075$ 。因此, 如果减速度和加速度一样, 手动减速点就设定为 $56,568-3,075=53,493$ 。

注意: 在固定线速度模式上, 不能使用这个算法。

■ 连续插补的加/减速驱动

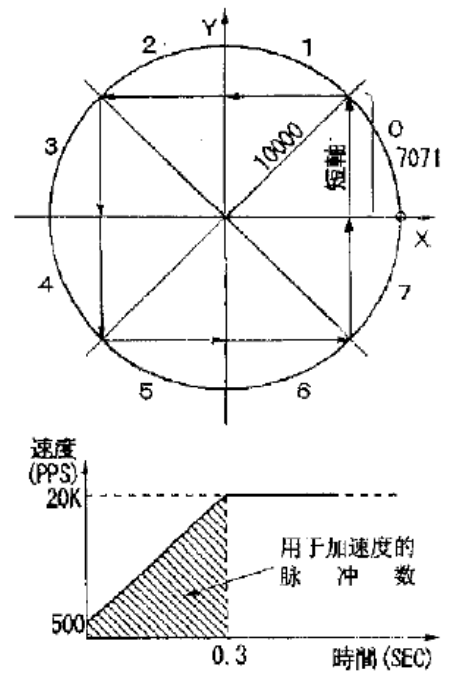
在连续插补中只能用手动减速的直线加/减速驱动, 不能用 S 曲线加/减速驱动及自动减速。在连续插补中要事先设定手动减速点。这个手动减速点设定在运行减速的最后一个节点上, 并设定从 X 轴输出的基本脉冲的数值。连续插补时先把减速设定为无效, 然后开始插补驱动。在要减速的最后一个节点上写入插补命令之前, 写入允许减速命令。开始最后一个节点的驱动时, 减速就有效。从最后一个插补节点开始计数主轴输出的基本脉冲数当大于手动减速点的数值时, 减速就开始。

比如, 从插补节点 1 至 5 的连续插补中, 在最后节点 5 上用手动减速的话, 有下述的程序。



● 由从节点 5 开始的基本脉冲数的数值来设定手动减速点。比如: 假定减速花费 2000 个脉冲, 在节点 5 上输出的基本脉冲的总脉冲数是 5000 的话, 则手动减速点就设定为 $5000-2000=3000$ 。

● 开始至停止一定要在 1 个节点内运行减速, 减速停止的最终插补节点需要从其 X 轴输出的基本脉冲总数要大于在减速中化费的脉冲数。



2.5 原点搜寻

■ 连续驱动的原点搜寻动作

把原点接近信号、原点信号、编码器 Z 相信号等安排在 nSTOP2~0。在各轴的 WR1 寄存器上设定各信号的有效/无效和逻辑电平。高速搜寻时，用加/减速连续驱动。当设定的有效信号处于激活电平时就减速停止。低速搜寻时，用定速连续驱动。当设定的有效信号处于激活电平时就立即停止。为了以加/减速连续驱动，除了输出脉冲数以外，都要设定和定量驱动一样的参数。

2.6 中断

发生中断有 2 种情况，其一是来自 X, Y 轴的中断，其二是位模式插补及连接插补时的中断。对 CPU 的中断信号只有一个 INTN 信号。

X, Y 轴产生的中断

中断位允许禁止	RR5 状态位	中断发生的条件
PBCM	D1 (PBCM)	当逻辑/实位计数器的值大于等于 COMP-的值时
PSCM	D2 (PSCM)	当逻辑/实位计数器的值小于 COMP-的值时
PSCP	D3 (PSCP)	当逻辑/实位计数器的值小于 COMP+的值时
PBCP	D4 (PBCP)	当逻辑/实位计数器的值大于等于 COMP+的值时
CEND	D5 (CEND)	在加/减速时，结束在定速区域输出脉冲
CSTA	D6 (CSTA)	在加/减速时，开始在定速区域输出脉冲
DEND	D7 (DEND)	驱动结束时

插补形成的中断

中断位的允许禁止	RR0 状态寄存器	中断发生的条件
连续插补的中断允许禁止位 (CIINT)	D9 (CNEXT)	在连续插补驱动中，可以写下一个插补节点的数据和插补驱动命令（写入下一个插补驱动命令后中断被清除）
位插补中断允许禁止位 (BPINT)	D14, 13 (BPS1, 0)	在位模式插补中堆栈计数值(SC)的数值从 2 变为 1，可以压入下一个 BP 数据（压栈 BP 数据后中断被清除）

2.7 其它功能

2.7.1 外部信号控制的驱动操作

此功能不是用命令，而是用外部信号来运行定量驱动，连续驱动。每个轴都有 nEXPP 和 nEXPM 的两个信号输入，nEXPP 信号用于正方向的驱动操作，nEXPM 信号用于负方向的操作驱动。我们提供 USB1010_SetOutEnableDV, USB1010_SetOutEnableLV 函数。

USB1010_SetOutEnableDV 用于设定外部控制定量驱动，当调用该函数设定好轴号，倍率，加速度，初始速度，驱动速度，输出脉冲数后，如果 nEXPP 引脚出现一个下降沿，则启动设定轴的正方向定长驱动，如果 nEXPM 引脚出现一个下降沿，则启动设定轴的反方向定长驱动。

USB1010_SetOutEnableLV 用于设定外部控制连续驱动，当调用该函数设定好轴号，倍率，加速度，初始速度，驱动速度后，如果 nEXPP 引脚保持低电平，则启动设定轴的正方向定长驱动。如果 nEXPM 引脚保持低电平，则启动设定轴的反方向定长驱动。（一旦引脚不是低电平了则停止连续驱动）

2.7.2 硬件限位(nLMTP(M))

硬件限位信号 (nLMTP (M)) 输入端用来终止脉冲输出。我们提供 USB1010_SetLMTEEnable 函数用来设定指定轴硬件限位信号有效，有效后是减速停止还是立即停止。调用该函数设定为 X 轴，立即停止，则当 XLMTTP 引脚出现低电平则电机立即停止。

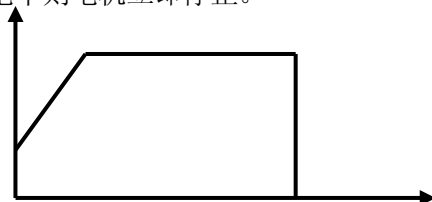


图 2.7.1 硬件限位停机

2.7.3 伺服电机报警信号(nALARM)

nALARM 输入信号接受从伺服马达驱动器的警报信号。设定为有效时，一直监视 nALARM 输入信号，若信号有效，RR2 寄存器的 D4 位为 1。若在驱动中，立即停止驱动。我们提供 USB1010_SetALARMEnable 函数，用来设定指定轴的 ALARM 信号有效(有效电平为低电平)。当调用函数设定 X 轴 ALARM 有效后，当 XALARM 引脚出现低电平，则电机停止驱动。

2.7.4 伺服电机到位信号 (nINPOS)

伺服马达定位完毕输入信号。我们提供 USB1010_SetINPOSEnable 函数用来设定指定轴 nINPOS 信号有效。当 nINPOS 有效后在驱动结束后，RR0 寄存器的 nDRV 位返回 0。

2.7.5 紧急停止

USB1010 有一个用于急停的输入端 SN1-19 (EMGN)。正常状态为高电平；当急停信号 EMGN 变为低电平时，所有轴将立即停止，各轴的 RR2 寄存器的 D5 (EMG) 位为 1。

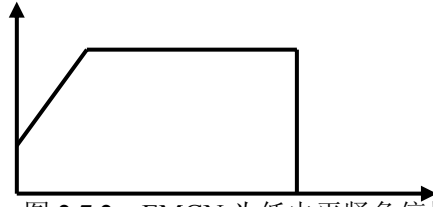


图 2.7.2 EMGN 为低电平紧急停机

2.7.6 脉冲输出类型

通过软件程序的设置，每个轴的驱动信号可设为两种输出方式之一：

(1) 独立 2 脉冲方式(CW/CCW 方式)：正向脉冲通过 nPP/PLS 输出，负方向脉冲通过 nPM/DIR 输出。

(2) 1 脉冲方式(CP/DIR 方式)：正负 2 个方向的驱动脉冲都通过输出信号 nPP/PLS 输出，脉冲方向通过 nPM/DIR 输出。

DIR 为高电平时，表示 nPP/PLS 输出的是正方向的驱动脉冲，DIR 为低电平时，表示 nPP/PLS 输出的是负方向的驱动脉冲。

脉冲输出类型表

脉冲输出方式	驱动方向	输出脉冲波形			
		CW (正)	CCW (负)	CP (脉冲)	DIR (方向)
CW/CCW 方式 (独立 2 脉冲方式)	正驱动方向	脉冲	低电平		
	负驱动方向	低电平	脉冲		
CP/DIR 方式 (1 脉冲方式)	正驱动方向			脉冲	高电平
	负驱动方向			脉冲	低电平

以上各控制输入输出点均有光电隔离。由外部的 DC24V 接入相应端子的供电，各点输入为低电平有效。X, Y 轴的脉冲输出和编码器的输入用高速光耦隔离。5V 供电。

2.7.7 驱动状态输出

每个轴的驱动状态输出至主状态寄存器 RR0 和每个轴的状态寄存器 nRR1 的每个位。此外，驱动状态输出可以做为信号输出。因为状态输出信号和通用输出信号共享引脚，所以当输出驱动状态时，置 WR3 寄存器的 D7 (OUTSL)位为 1。

在驱动中/驱动停止的状态反映 RR0 寄存器 D1, 0(n-DRV)位并由 nOUT4/DRIVE 信号输出。

此外，S 曲线加减速驱动中的加速度，减速度的增加/恒定/减少的状态也反映在 RR1 寄存器的 D5(AASND), D6(ACNST), D7(ADSND)位。并由 nOUT0/ACASND, nOUT1/ACDSND 信号输出。

驱动状态	驱动寄存器 (激活状态时: 1)	输出信号 (激活状态时: Hi)
------	---------------------	---------------------

驱动	RR0/D1, 0 (n-DRV)	nDRIVE
加速	nRR1/D2 (ASND)	nASND
定速	nRR1/D3 (CNST)	nCNST
减速	nRR1/D4 (DSND)	nDSND
加减速增加	nRR1/D7 (ADSND)	nACASND
加减速固定	nRR1/D6 (ACNST)	—
加减速减少	nRR1/D5 (AASND)	nACDSND

2.7.8 通用 I/O 信号

此IC 的每个轴都有nOUT3~0, nOTU7~4 的 8 个通用输出信号和nIN5~0 的 6 个通用输出信号。不过, nIN5~2 引脚和复数芯片插补的引脚合用, 所以使用此功能时, 就不能使用那些信号。nOUT7~4 引脚和位置比较输出及驱动状态输出的引脚合用, 所以使用这些输出时, 就不能使用那些信号。X 轴的XIN5~0 信号状态表示在RR4 寄存器的D13~8, Y 轴的YIN5~0 信号状态表示在RR5 寄存器的D13~8。低电平是 0, 高电平是 1。此外, nIN5~0 信号带有输入信号滤波功能, 请参考《[输入信号滤波器](#)》。

设定输出电平的数值后, X 轴的 XOUT7~0 信号输出至 WR4 寄存器 D7~0 的每个位, Y 轴的 YOUT7~0 信号输出至 WR4 寄存器 D15~8 的每个位。

设定为 0, 低电平被输出; 设定为 1, 高电平被输出。复位时, WR4 寄存器的每个位被子清除, 所有的输出变为低电平。通用输出信号可以作为马达驱动的励磁 OFF, 偏差计数器清除, 报警复位等。

在设定好 WR4 寄存器各位输出电平的数值后, 由 nOUT3~0 信号输出。使用 nOUT7~4 信号时, 通过 WR3 寄存器的 D7(OUTSL)设定为使用通用输出的模式。对 WR3 寄存器的 D11~8(OUT7~4)设定输出电平的数值后, 就输出。通用输出信号可以作为马达驱动器的励磁 OFF, 偏差计数器清除, 报警复位等。复位时, WR4 寄存器, nWR3 寄存器的每个位被清除, 所有的输出为低电平。

第三章 库函数驱动程序的使用说明

3.1 函数调用举例(vc)说明

■ 卡号定义:

当插入一块卡时, 卡号默认为 0 号卡, 插入两块卡, 从右(电源一侧)向左依次为 0 号卡, 1 号卡; 三块卡则依次为 0 号卡, 1 号卡, 2 号卡。依次类推。例:

```
// 该卡插入第一槽, 定义为 0 号, 该赋值语句用于获得该卡的句柄
HANDLE hDevice = USB1010_CreateDevice(0);
// 该卡插入第三槽, 定义为 1 号 (第二槽插入了其他卡), 该赋值语句用于获得该卡的句柄
HANDLE hDevice = USB1010_CreateDevice(1);
```

■ 初始化:

```
USB1010_InitDevice (hDevice); // 初始化卡
```

■ 调用函数举例

3.1.1 使用 USB1010_InitLVDV, USB1010_StartLVDV 定长、连续脉冲驱动函数启动电机进行定长驱动

以下是公共参数和直线 S 曲线参数的结构体

```
// 公用参数
#ifndef _USB1010_PAPA_DataList
typedef struct _USB1010_PAPA_DataList
{
    LONG Multiple; // 倍率 (1~500)
    LONG StartSpeed; // 初始速度(1~8000)
    LONG DriveSpeed; // 驱动速度(1~8000)
    LONG Acceleration; // 加速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG Deceleration; // 减速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG AccelerationK; // 加速度变化率(954~62500000)
} USB1010_PARA_DataList, *PUSB1010_PARA_DataList;
#endif

// 直线和 S 曲线参数
#ifndef _USB1010_PAPA_LCData
typedef struct _USB1010_PAPA_LCData
{
    LONG AxisNum; // 轴号 (X 轴 | Y 轴 | X、Y 轴)
    LONG LV_DV; // 驱动方式 (连续 | 定长 )
    LONG Mode; // 脉冲方式 (CW/CCW 方式 | CP/DIR 方式)
    LONG Line_Curve; // 运动方式 (直线 | 曲线)
    LONG Direction; // 运动方向 (正方向 | 反方向)
    LONG nPulseNum; // 定量输出脉冲数 (0~268435455)
} USB1010_PARA_LCData, *PUSB1010_PARA_LCData;
#endif
```

例: 要使 1 号卡的 Y 轴电机以 Pulse\DIR 方式; 直线加/减速; 倍率为 2; 加速度为 25000 PPS/s; 10 PPS(脉冲数/秒)初始速度; 5000 PPS 驱动速度; 输出脉冲数 50000; 正方向定长转动; 参考下例:

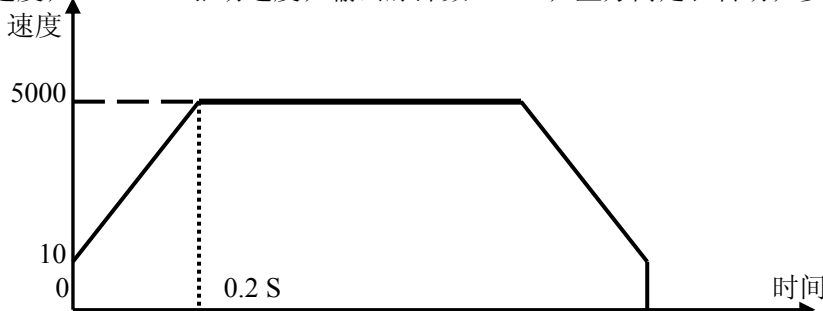


图 3.1.1

```
HANDLE hDevice = USB1010_CreateDevice(1); // 创建设备
USB1010_PARA_DataList DL; // 定义公共参数结构体
```

```

USB1010_PARA_LCDData LC; // 定义直线曲线参数结构体
LC.AxisNum = USB1010_YAXIS; // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
LC.LV_DV= USB1010_DV; // 驱动方式 USB1010_DV:定长驱动 USB1010_LV:连续驱动
LC.Mode = USB1010_CPDIR; // 脉冲方式 USB1010_CWCCW:CW/CCW 方式,
                          USB1010_CPDIR:CP/DIR 方式
LC.Line_Curve = USB1010_LINE; // 运动方式 USB1010_LINE:直线加/减速;
                              USB1010_CURVE:S 曲线加/减速)
DL.Multiple=2; // 倍率 (1~500)
DL.Acceleration = 1250; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
                       实际加速度=设置加速度(Acceleration)*倍率(Multiple)
DL.Deceleration = 1250; // 减速度(125~1000,000)(直线加减速驱动中加速度一直不变)
                       实际减速度=设置减速度(Deceleration)*倍率(Multiple)
DL.StartSpeed = 5; // 初始速度(1~8000)实际初始速度=倍率(Multiple) *设置的初始速度(StartSpeed)
DL.DriveSpeed=2500;// 驱动速度(1~8000)实际驱动速度=倍率(Multiple) *设置的驱动速度(DriveSpeed)
LC.nPulseNum = 50000; // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION; // 运动方向 USB1010_PDIRECTION: 正转,
                                     USB1010_MDIRECTION:反转
USB1010_InitLVDV( // 初始化连续,定长脉冲驱动
                  hDevice,
                  &DL,
                  &LC);
USB1010_StartLVDV( // 启动定长脉冲驱动
                  hDevice, // 设备句柄
                  LC.AxisNum);

```

3.1.2 使用 USB1010_InitLVDV、USB1010_StartLVDV 定长、连续脉冲驱动函数启动电机进行连续驱动

```

// 公用参数
#ifndef _USB1010_PAPA_DataList
typedef struct _USB1010_PAPA_DataList
{
    LONG Multiple; // 倍率 (1~500)
    LONG StartSpeed; // 初始速度(1~8000)
    LONG DriveSpeed; // 驱动速度(1~8000)
    LONG Acceleration; // 加速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG Deceleration; // 减速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG AccelerationK; // 加速度变化率(954~62500000)
} USB1010_PARA_DataList, *PUSB1010_PARA_DataList;
#endif
// 直线和 S 曲线参数
#ifndef _USB1010_PAPA_LCDData
typedef struct _USB1010_PAPA_LCDData
{
    LONG AxisNum; // 轴号 (X 轴 | Y 轴 | X、Y 轴)
    LONG LV_DV; // 驱动方式 (连续 | 定长 )
    LONG Mode; // 脉冲方式 (CW/CCW 方式 | CP/DIR 方式)
    LONG Line_Curve; // 运动方式 (直线 | 曲线)
    LONG Direction; // 运动方向 (正方向 | 反方向)
    LONG nPulseNum; // 定量输出脉冲数(0~268435455)
} USB1010_PARA_LCDData, *PUSB1010_PARA_LCDData;
#endif

```

例: 要使 0 号卡的 X 轴电机以 CW/CCW 方式; 直线加减速; 1 倍倍率; 加速度为 4000 PPS/s; 100PPS(脉冲数/秒)初始速度; 8000 PPS 驱动速度; 负方向连续运动, 可参考下例程序:

```

HANDLE hDevice =USB1010_CreateDevice(0);// 创建设备
USB1010_PARA_DataList DL; // 定义公共参数结构体
USB1010_PARA_LCDData LC; // 定义直线曲线参数结构体

```

```

LC.AxisNum = USB1010_XAXIS; // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
LC.LV_DV= USB1010_LV; // 驱动方式 USB1010_DV:定长驱动 USB1010_LV:连续驱动
LC.Mode = USB1010_CWCCW; // 脉冲方式 USB1010_CWCCW:CW/CCW方式,
                           USB1010_CPDIR:CP/DIR方式
LC.Line_Curve = USB1010_LINE;// 运动方式 USB1010_LINE:直线加/减速;USB1010_CURVE:S曲线加/
减(速)
DL.Multiple=1; // 倍率 (1~500)
DL.Acceleration = 4000; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
                        实际加速度=设置加速度(Acceleration)*倍率(Multiple)
DL.Deceleration = 4000; // 减速度(125~1000,000)(直线加减速驱动中加速度一直不变)
                        实际减速度=设置减速度(Deceleration)*倍率(Multiple)
DL.StartSpeed = 100 ; // 初始速度(1~8000), 实际初始速度=倍率(Multiple) *设置的初始速度(StartSpeed)
DL.DriveSpeed=8000;// 驱动速度(1~8000), 实际驱动速度=倍率(Multiple) *设置的驱动速度(DriveSpeed)
LC.Direction=USB1010_PDIRECTION;// 运动方向 USB1010_PDIRECTION: 正转,
                           USB1010_MDIRECTION:反转
USB1010_InitLVDV( // 初始化连续,定长脉冲驱动
                  hDevice,
                  &DL,
                  &LC);
USB1010_StartLVDV( // 启动定长脉冲驱动
                  hDevice, // 设备句柄
                  LC.AxisNum);

```

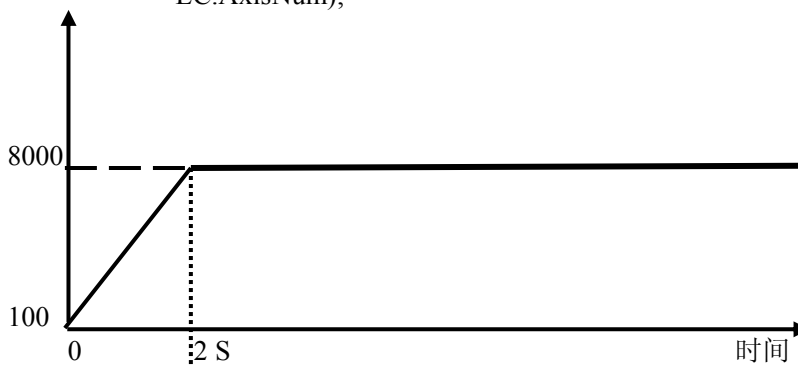


图 3.1.2

3.1.3 使用 USB1010_InitLVDV、USB1010_Start2D 函数，启动两轴同时驱动

例：需要设置 0 号卡的 X、Y 轴电机以 CW/CCW 方式，直线加减速，1 倍倍率，12500PPS/S 的加速度，100PPS（脉冲数/秒）的初速度，4000PPS/S 的驱动速度，100000 的脉冲数，正方向定长驱动。参考下例：

```

HANDLE hDevice =USB1010_CreateDevice(0);
USB1010_PARA_DataList DL[2];
USB1010_PARA_LCData LC[2];
USB1010_SetLP(hDevice, USB1010_ALLAXIS, 0); // 逻辑位置计数器清零
USB1010_SetEP(hDevice, USB1010_ALLAXIS, 0); // 实位计数器清零
USB1010_SetAccofst(hDevice, USB1010_ALLAXIS, 0);// 设置加速计数器
for(i=0; i<2; i++)
{
    LC[i].AxisNum = i; // 轴号(USB1010_XAXIS:X轴, USB1010_YAXIS:Y轴)
    LC[i].LV_DV= USB1010_DV; // 驱动方式 USB1010_DV:定长驱动, USB1010_LV:连续驱动
    LC[i].Mode = USB1010_CWCCW; // 模式 0: CW/CCW方式, 1: CP/DIR方式
    LC[i].Line_Curve = USB1010_LINE; // 直线曲线(0:直线加/减速; 1:S曲线加/减速)
    DL[i].Multiple=1; // 倍数 (1~500)
    DL[i].Acceleration = 12500; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
    DL[i].Deceleration = 12500; // 减速度(125~1000,000)(直线加减速驱动中加速度一直不变)
    DL[i].AccelerationK= 1000; // 加速度变化率(仅 S曲线驱动时有效)
    DL[i].StartSpeed = 1 ; // 初始速度(1~8000)
}

```



```

DL[i].DriveSpeed = 8000 ;      // 驱动速度 (1~8000)
LC[i].nPulseNum = 100000 ;    // 定量输出脉冲数(0~268435455)
LC[i].Direction = USB1010_PDIRECTION ; // 转动方向 USB1010_PDIRECTION: 正转,
                                   USB1010_MDirection:反转
USB1010_InitLVDV(           // 初始单轴化连续,定长脉冲驱动
    hDevice,
    &DL[i],
    &LC[i]);
}
USB1010_Start2D( hDevice);

```

3.1.4 使用 USB1010_InitCWInterpolation 函数，启动正反方向圆弧插补驱动

```

// 公用参数
#ifndef USB1010_PAPA_DataList
typedef struct _USB1010_PAPA_DataList
{
    LONG Multiple;           // 倍率 (1~500)
    LONG StartSpeed;        // 初始速度(1~8000)
    LONG DriveSpeed;        // 驱动速度(1~8000)
    LONG Acceleration;      // 加速度(125~1000000)，实际的取值应为 125 的整数倍
    LONG Deceleration;      // 减速度(125~1000000)，实际的取值应为 125 的整数倍
    LONG AccelerationK;     // 加速度变化率(954~62500000)
} USB1010_PARA_DataList, *PUSB1010_PARA_DataList;
#endif
// 正反方向圆弧插补参数
#ifndef USB1010_PAPA_CircleData
typedef struct _USB1010_PAPA_CircleData
{
    LONG ConstantSpeed;     // 固定线速度 (不固定线速度 | 固定线速度)
    LONG XCenter;           // X 轴圆心坐标 (脉冲数 -8388608~8388607)
    LONG YCenter;           // Y 轴圆心坐标 (脉冲数 -8388608~8388607)
    LONG XPulse;            // X 轴终点坐标 (脉冲数 -8388608~8388607)
    LONG YPulse;            // Y 轴终点坐标 (脉冲数 -8388608~8388607)
} USB1010_PARA_CircleData, *PUSB1010_PARA_CircleData;
#endif

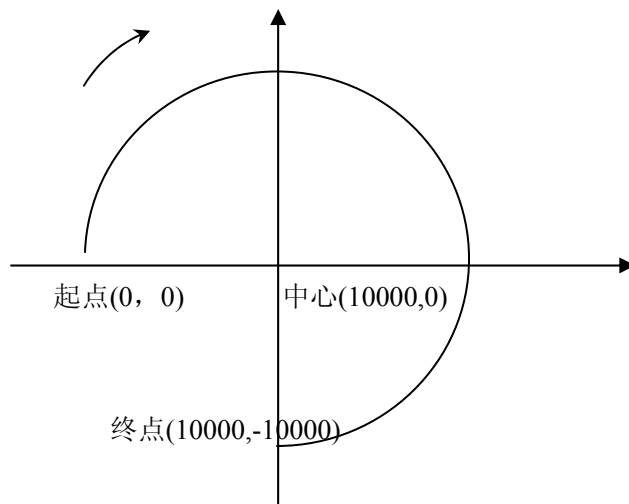
```

例：要使 0 号卡以 (X: 10000, Y: 0) 为圆心，以 (X: 0, Y: 0) 为终点，速度为 1000PPS (脉冲数/秒) 固定线速度进行正方向的圆弧插补。参考下例：

```

HANDLE hDevice =USB1010_CreateDevice(0); // 获得句柄号
USB1010_PARA_DataList DL;
USB1010_PARA_CircleData CD;
CD.ConstantSpeed = 1;           // USB1010_CONSTAND:固定，USB1010_NOCONSTAND: 不固定
CD.XCenter =10000;             // X 轴圆心坐标 (脉冲数)
CD.YCenter = 0;                // Y 轴圆心坐标 (脉冲数)
CD.XPulse =0;                  // X 轴终点坐标 (脉冲数)
CD.YPulse =0;                  // Y 轴终点脉冲数
DL.Multiple = 1;                // 倍数(1~500)
DL.Acceleration=5000;          // 加速度(125~1000000)
DL.StartSpeed = 1000;          // 初始速度(1~8000)
DL.DriveSpeed = 1000;          // 驱动速度(1~8000)
USB1010_InitCWInterpolation (hDevice, // 初始化圆弧插补
    &DL,
    &CD);
USB1010_StartCWInterpolation (hDevice); // 启动正方向圆弧插补
USB1010_StartCCWInterpolation (hDevice); // 启动反方向圆弧插补

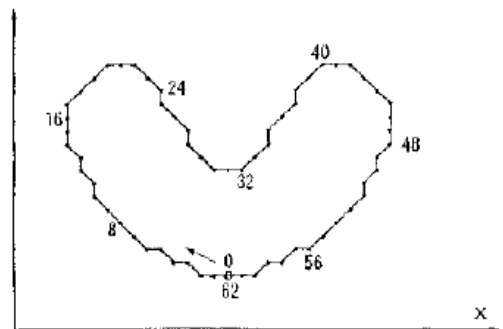
```



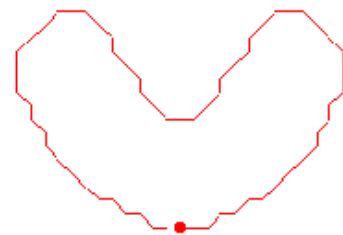
3.1.5 位插补举例

例：要运行如右图所示的轨迹，参考下例。

分析：要运行这样的轨迹，首先从图中算出每个轴要输出的脉冲。从 0 到 8，XPM 一直输出脉冲及 1111,1111，XPP 输出 0000,0000，YPP 输出 1101,0100 即 0XD4，YPM 输出 0000,0000，从 8 到 16，XPM 输出 0010,1011，XPP 不输出脉冲为 0000,0000，YPP 输出 1111,1111，YPM 不输出脉冲为 0000,0000，因此第一个 16 位数据为 BP1P 为 0X0，BP1M 为 0X2BFF，BP2P 为 0XFFD4，BP2M 为 0X0；其它数据一样。



```
USHORT nBitData[16] = {
    0x0000, 0x2BFF, 0XFFD4, 0x0000,
    0XF6FE, 0x0000, 0X000F, 0x3FC0,
    0x1FDB, 0x0000, 0x00FF, 0xFC00,
    0x4000, 0x7FF5, 0x0000, 0x0AFF};
HANDLE hDevice =USB1010_CreateDevice(0);
USB1010_PARA_DataList DL;
DL.Multiple = 1;
DL.Acceleration = 1250;
DL.StartSpeed = 1;
DL.DriveSpeed = 1;
USB1010_InitBitInterpolation ( // 初始化位插补参数
    hDevice, // 设备句柄
    &DL); // 公共参数结构体指针
```



```
USB1010_AutoBitInterpolation (hDevice, nBitData, 16); // 启动位插补子线程
```

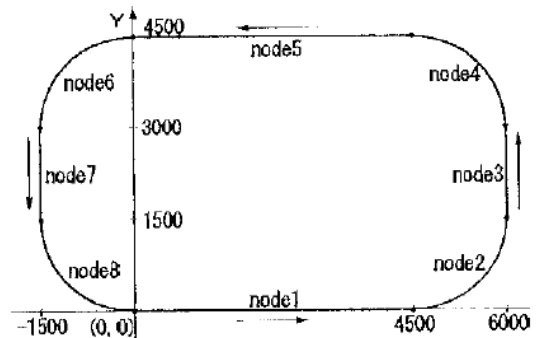
在我们的高级演示程序里画出的轨迹如右图所示。

3.1.6 连续插补举例

例 1：如右图所示，使用 0 号卡以 1 倍倍率，1000PPS 初始速度，1000PPS 驱动速度，运行如右图所示的轨迹。

分析：该轨迹总共有八段线段或圆弧组成。先直线，圆弧，直线，圆弧……对直线插补知道起点，终点，圆弧插补知道圆心坐标，终点坐标，所以很容易得到该轨迹。

```
HANDLE hDevice =USB1010_CreateDevice(0); // 获得句柄号
USB1010_PARA_DataList DL; // 公共参数结构体
USB1010_PARA_LineData LD; // 直线插补参数结构
LD.Line_Curve = USB1010_LINE; // USB1010_LINE:直线，USB1010_Curve:S 曲线
LD.ConstantSpeed = USB1010_CONSTAND; // USB1010_CONSTAND: 固定线速度，
USB1010_NOCONSTAND: 不固定线速度
```



```

DL.Multiple =1;           // 倍率 (1~500)
DL.Acceleration=1250;    // 加速度 (125~1000, 000)
DL.StartSpeed = 1000;    // 初始速度 (1~8000)
DL.DriveSpeed = 1000;    // 驱动速度 (1~8000)
LD.nXAxisPulseNum=4500;  // X 轴脉冲数
LD.nYAxisPulseNum= 0;    // Y 轴脉冲数
USB1010_SetLP(hDevice, USB1010_XAXIS, 0); // 清逻辑位置计数器
USB1010_SetLP(hDevice, USB1010_YAXIS, 0); // 清实际位置计数器
USB1010_InitLineInterpolation(           // 初始化直线插补运动
                                   hDevice, // 设备句柄
                                   &DL,
                                   &LD);

USB1010_StartLineInterpolation(hDevice); // 启动直线插补驱动
USB1010_NextWait( hDevice); // 等待写入下一个节点的参数和命令
USB1010_PARA_CircleData CD;
CD.ConstantSpeed = 1; // USB1010_CONSTAND:固定 USB1010_NOCONSTAND:不固定
CD.XCenter =0; // X 轴圆心坐标 (脉冲数)
CD.YCenter = 4500; // Y 轴圆心坐标 (脉冲数)
CD.XPulse =4500; // X 轴终点坐标 (脉冲数)
CD.YPulse =4500; // Y 轴终点脉冲数
USB1010_InitCWInterpolation(hDevice, // 初始化圆弧插补
                             &DL,
                             &CD);
USB1010_StartCWInterpolation( // 启动正方向圆弧插补
                             hDevice); // 设备号
USB1010_NextWait( hDevice); // 等待写入下一个节点的参数和命令
LD.nXAxisPulseNum=0;
LD.nYAxisPulseNum=4500;
USB1010_InitLineInterpolation( // 初始化直线插补运动
                             hDevice, // 设备句柄
                             &DL,
                             &LD);
USB1010_StartLineInterpolation(hDevice); // 启动直线插补驱动
重复上诉过程即可得到该轨迹。

```

3.1.7 外部信号控制的驱动函数

每个轴都有 nEXPP 和 nEXPM 2 个输入信号, nEXPP 用于正方向的驱动控制, nEXPM 用于反方向的驱动控制。在设定好轴号, 倍数, 加速度, 初始速度, 驱动速度, 输出脉冲数 (定量驱动时需要设定) 后, 由 nEXPP 和 nEXPM 启动电机。

例 1: 用外部信号 (nEXPP 或 nEXPM) 启动 0 号卡 Y 轴电机以 2 倍倍率, 25000PPS/SEC 的加速度, 200PPS 的初始速度, 16000PPS 的驱动速度, 直线加减速输出 100000 个脉冲定量驱动。

```

HANDLE hDevice =USB1010_CreateDevice(0); // 获得句柄号
USB1010_PARA_DataList DL;
USB1010_PARA_LCDData LC;
LC.AxisNum = USB1010_YAXIS;
LC.Line_Curve = USB1010_LINE;
LC.nPulseNum = 100000;
LC.Mode = USB1010_CWCCW;
DL.Multiple = 2;
DL.Acceleration = 12500;
DL.AccelerationK = 1000;
DL.StartSpeed = 100;
DL.DriveSpeed = 8000;
USB1010_InitLVDV(           // 初始化连续,定长脉冲驱动
                    hDevice, // 设备句柄

```



```

        &DL,           // 公共参数结构体指针
        &LC);         // 直线 S 曲线参数结构体指针
    USB1010_SetOutEnableDV( // 设置外部使能定量驱动(下降沿有效)
        hDevice,     // 设备句柄
        LC.AxisNum); // 轴号(USB1010_XAXIS:X 轴,USB1010_YAXIS:Y 轴)

```

运行程序后，当 nEXPP 引脚上出现低电平（nEXPP，nEXPM 在不加电平时为高电平），即有一个下降沿后，电机启动正方向的定量驱动，直到输完脉冲数停止。当 nEXPM 引脚上出现低电平，即有一个下降沿后，电机启动反方向的定量驱动，直到输完脉冲数停止。

例 2: 用外部信号 (nEXPP 或 nEXPM) 启动 0 号卡 X 轴电机以 1 倍倍率，12500PPS/SEC 的加速度，100PPS 的初始速度，8000PPS 的驱动速度直线连续驱动。

```

HANDLE hDevice =USB1010_CreateDevice(0); // 获得句柄号
USB1010_PARA_DataList DL;
USB1010_PARA_LCDData LC;
LC.AxisNum = USB1010_XAXIS;
LC.Line_Curve = USB1010_LINE;
LC.nPulseNum = 100000;           // 初始化要设置，但不会用到
LC.Mode = USB1010_CWCCW;       // 脉冲方式 USB1010_CWCCW: CW/CCW 方式,
                                USB1010_CPDIR: CP/DIR 方式

DL.Multiple = 1;                // 倍率
DL.Acceleration = 12500;        // 加速度
DL.AccelerationK = 1000;       // 加速度变化率
DL.StartSpeed = 100;           // 初始速度
DL.DriveSpeed = 8000;          // 驱动速度
USB1010_InitLVDV(              // 初始化连续,定长脉冲驱动
    hDevice,                    // 设备句柄
    &DL,                         // 公共参数结构体指针
    &LC);                        // 直线 S 曲线参数结构体指针
USB1010_SetOutEnableLV(        // 设置外部使能定量驱动(下降沿有效)
    hDevice,                    // 设备句柄
    LC.AxisNum);               // 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)

```

运行程序后，在 nEXPP 引脚上保持低电平期间，电机运行正方向连续驱动。在 nEXPM 引脚上保持低电平期间，电机运行反方向连续驱动。

3.1.8 设置外部超限信号有效及停止方式

例: 如果设置 0 号卡的 X 轴电机：硬件限制，减速停止。参考下例：

(当电机在运行正方向的驱动时，XLMTTP 引脚上为低电平时，电机立即减速停止。)

```

HANDLE hDevice =USB1010_CreateDevice(0); // 获得句柄号
USB1010_PARA_DataList DL;                // 定义公共参数结构体
USB1010_PARA_LCDData LC;                 // 定义直线曲线参数结构体
LC.AxisNum = USB1010_XAXIS;              // 轴号(USB1010_XAXIS:X 轴,USB1010_YAXIS:Y 轴)
LC.LV_DV= USB1010_LV;                    // 驱动方式 USB1010_DV:定长驱动 USB1010_LV:连续驱动
LC.Mode = USB1010_CWCCW;                 // 脉冲方式 USB1010_CWCCW: CW/CCW 方式,
                                USB1010_CPDIR: CP/DIR 方式
LC.Line_Curve = USB1010_LINE;            // 运动方式 USB1010_LINE:直线加/减速;
                                USB1010_CURVE: S 曲线加/减速)
DL.Multiple=1;                            // 倍率 (1~500)
DL.Acceleration = 2500;                   // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
DL.AccelerationK = 2000;                  // 加速度变化率(仅 S 曲线驱动时有效)
DL.StartSpeed = 100 ;                     // 初始速度(1~8000)
DL.DriveSpeed = 5000;                     // 驱动速度 (1~8000)
LC.nPulseNum = 10000 ;                    // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION;        // 运动方向 USB1010_PDirection: 正转,

```

```

                                USB1010_MDirection:反转
USB1010_InitLVDV(                // 初始化连续,定长脉冲驱动
    hDevice,
    &DL,
    &LC);
USB1010_SetLMTEnable(           // 设置外部超限信号的有效及停止方式
    hDevice,                     // 设备句柄
    LC.AxisNum,                 // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
    USB1010_DECSTOP);          // USB1010_DECSTOP:减速停止,
                                USB1010_SUDDENSTOP:立即停止
USB1010_StartLVDV(             // 启动定长脉冲驱动
    hDevice,                   // 设备句柄
    LC.AxisNum);

```

3.1.9 设置伺服马达输出到位有效

例 1: 设置 1 号卡 (第二块卡) 的 Y 轴电机 INPOS 有效, 直接调用 USB1010_SetINPOSEnable 函数。

```

HANDLE hDevice =USB1010_CreateDevice(1); // 获得句柄号
USB1010_PARA_DataList DL;                // 定义公共参数结构体
USB1010_PARA_LCDData LC;                 // 定义直线曲线参数结构体
LC.AxisNum = USB1010_YAXIS;              // 轴号(USB1010_XAXIS:X轴, USB1010_YAXIS:Y轴)
LC.LV_DV= USB1010_LV;                    // 驱动方式 USB1010_DV:定长驱动, USB1010_LV:连续驱动
LC.Mode = USB1010_CWCCW;                 // 脉冲方式 USB1010_CWCCW:CW/CCW 方式,
                                USB1010_CPDIR:CP/DIR 方式
LC.Line_Curve = USB1010_LINE;            // 运动方式 USB1010_LINE:直线加/减速;
                                USB1010_CURVE:S 曲线加/减速)
DL.Multiple=1;                           // 倍率 (1~500)
DL.Acceleration = 2500;                   // 加速度(125~1000,000)
DL.Deceleration = 2500;                   // 减速度(125~1000,000)
DL.StartSpeed = 10 ;                     // 初始速度(1~8000)
DL.DriveSpeed = 5000;                     // 驱动速度(1~8000)
LC.nPulseNum = 100000 ;                  // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION;        // 运动方向 USB1010_PDirection: 正转,
                                USB1010_MDirection:反转
USB1010_InitLVDV(                       // 初始化连续,定长脉冲驱动
    hDevice,
    &DL,
    &LC);
USB1010_SetINPOSEnable(                 // 设置伺服马达定位完毕输入信号有效
    hDevice,                             // 设备句柄
    LC.AxisNum);                         // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
USB1010_StartLVDV(                     // 启动定长脉冲驱动
    hDevice,                             // 设备句柄
    LC.AxisNum);

```

在 nINPOS 信号设置为有效时, 在驱动结束后, RR0 的驱动状态位仍然为 1, 当 nINPOS 引脚出现低电平 (即出现一个下降沿时), RR0 状态寄存器的状态位才返回为 0。比如上例, 在设置 Y 轴 INPOS 信号有效, 并启动电机后, RR0 的 D1 位及 YDRV 为 1, 电机结束驱动后仍然为 1, 在 YINPOS 引脚出现低电平 (即出现一个下降沿时) 该位才返回 0。

例 2: 需要设置一号卡 X 轴外部停止信号 0 有效, 直接调用 USB1010_SetStopEnable 函数。

```

HANDLE hDevice =USB1010_CreateDevice(1); // 获得句柄号
USB1010_PARA_DataList DL;                // 定义公共参数结构体
USB1010_PARA_LCDData LC;                 // 定义直线曲线参数结构体
LC.AxisNum = USB1010_XAXIS;              // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
LC.LV_DV= USB1010_LV;                    // 驱动方式 USB1010_DV:定长驱动 USB1010_LV:连续驱动

```



```

LC.Mode = USB1010_CWCCW; // 脉冲方式 USB1010_CWCCW: CW/CCW 方式,
                          // USB1010_CPDIR: CP/DIR 方式
LC.Line_Curve = USB1010_LINE; // 运动方式 USB1010_LINE: 直线加/减速;
                          // USB1010_CURVE: S 曲线加/减速)
DL.Multiple=1; // 倍率 (1~500)
DL.Acceleration = 2500; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
DL.AccelerationK = 2000; // 加速度变化率(仅 S 曲线驱动时有效)
DL.StartSpeed = 100 ; // 初始速度(1~8000)
DL.DriveSpeed = 5000; // 驱动速度 (1~8000)
LC.nPulseNum = 10000 ; // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION; // 运动方向 USB1010_PDIRECTION: 正转,
                          // USB1010_MDirection: 反转
USB1010_InitLVDV( // 初始化连续,定长脉冲驱动
    hDevice,
    &DL,
    &LC);
USB1010_SetStopEnable( // 设置外部停止信号有效
    hDevice, // 设备句柄
    LC.AxisNum, // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
    USB1010_IN0); // 停止号
USB1010_StartLVDV( // 启动定长脉冲驱动
    hDevice, // 设备句柄
    LC.AxisNum);

```

当 XIN0 信号出现低电平及出现一个下降沿时, 电机执行减速停止。

例 3: 1 号卡 X 轴设置伺服报警信号有效, 调用 USB1010_SetALARMEnable 函数。

```

HANDLE hDevice = USB1010_CreateDevice(1); // 获得句柄号
USB1010_PARA_DataList DL; // 定义公共参数结构体
USB1010_PARA_LCDData LC; // 定义直线曲线参数结构体
LC.AxisNum = USB1010_XAXIS; // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
LC.LV_DV = USB1010_LV; // 驱动方式 USB1010_DV: 定长驱动, USB1010_LV: 连续驱动
LC.Mode = USB1010_CWCCW; // 脉冲方式 USB1010_CWCCW: CW/CCW 方式,
                          // USB1010_CPDIR: CP/DIR 方式
LC.Line_Curve = USB1010_LINE; // 运动方式 USB1010_LINE: 直线加/减速;
                          // USB1010_CURVE: S 曲线加/减速)
DL.Multiple=1; // 倍率 (1~500)
DL.Acceleration = 2500; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
DL.AccelerationK = 2000; // 加速度变化率(仅 S 曲线驱动时有效)
DL.StartSpeed = 100 ; // 初始速度(1~8000)
DL.DriveSpeed = 5000; // 驱动速度 (1~8000)
LC.nPulseNum = 10000 ; // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION; // 运动方向 USB1010_PDIRECTION: 正转,
                          // USB1010_MDirection: 反转
USB1010_InitLVDV( // 初始化连续,定长脉冲驱动
    hDevice,
    &DL,
    &LC);
USB1010_SetALARMEnable( // 设置伺服报警信号有效
    hDevice, // 设备句柄
    LC.AxisNum); // 轴号(USB1010_XAXIS:X轴,USB1010_YAXIS:Y轴)
USB1010_StartLVDV( // 启动定长脉冲驱动
    hDevice, // 设备句柄
    LC.AxisNum);

```

当 XALARM 信号出现低电平及出现一个下降沿时, 电机执行立即停止。

3.1.10 实际位置计数器例子

实际位置计数器可以对外部信号进行计数，对 nECA 引脚信号进行向上计数，对 nECB 引脚信号进行向下计数。

例：我们把电机 X 轴正向输出脉冲引脚（XPP）接到 XECA 引脚，用 X 轴实际位置计数器对 X 轴脉冲计数，我们启动 X 轴正方向定长驱动，输入 10000 个脉冲后停止。参考下例：

```

LONG A,LP,EP,speed;
HANDLE hDevice = USB1010_CreateDevice(0); // 创建设备
USB1010_PARA_DataList DL; // 定义公共参数结构体
USB1010_PARA_LCData LC; // 定义直线曲线参数结构体
LC.AxisNum = USB1010_XAXIS; // 轴号(USB1010_XAXIS:X 轴,USB1010_YAXIS:Y 轴)
LC.LV_DV= USB1010_DV; // 驱动方式 USB1010_DV:定长驱动 USB1010_LV:连续驱动
LC.Mode = USB1010_CWCCW; // 脉冲方式 USB1010_CWCCW:CW/CCW 方式,
// USB1010_CPDIR:CP/DIR 方式
LC.Line_Curve = USB1010_LINE; // 运动方式 USB1010_LINE:直线加/减速;
// USB1010_CURVE:S 曲线加/减速)
DL.Multiple=1; // 倍率 (1~500)
DL.Acceleration = 1250; // 加速度(125~1000,000)(直线加减速驱动中加速度一直不变)
DL.Deceleration = 2500; // 减速度(125~1000,000)(直线加减速驱动中加速度一直不变)
DL.StartSpeed = 100 ; // 初始速度(1~8000)
DL.DriveSpeed = 8000; // 驱动速度(1~8000)
LC.nPulseNum = 50000; // 定量输出脉冲数(0~268435455)
LC.Direction = USB1010_PDIRECTION; // 运动方向 USB1010_PDIRECTION: 正转,
// USB1010_MDIRECTION:反转
USB1010_InitLVDV( // 初始化连续,定长脉冲驱动
    hDevice,
    &DL,
    &LC);
USB1010_SetEncoderSignalType( // 设置编码器输入信号类型
    hDevice, // 设备句柄
    LC.AxisNum, // 轴号(USB1010_XAXIS:X 轴,USB1010_YAXIS:Y 轴)
    1); // 设置编码器输入信号类型 0: 两相脉冲输入 1: 上/下脉冲输入
USB1010_StartLVDV(hDevice); // 启动定长、连续脉冲驱动
while(1)
{
    LP = USB1010_ReadLP(hDevice, 0X1);
    EP = USB1010_ReadEP(hDevice, 0X1);
    printf("实位计数器 = %ld ", EP);
    printf("逻辑位置计数器 = %ld \n", LP);
}
    
```

可以看到逻辑位置计数器输入的脉冲数和实际位置计数器的计数值。两个完全相同。

3.1.11 读 RR 状态寄存器的位状态

为了方便用户最大程度的利用板卡资源，我们提供了状态寄存器，可以用 USB1010_GetRR0Status 读状态寄存器。

■ **RR0 主状态寄存器**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	BPSC1	BPSC0	ZONE2	ZONE1	ZONE0	CNEXT	I-DRV	U-ERR	Z-ERR	Y-ERR	X-ERR	U-DRV	Z-DRV	Y-DRV	X-DRV

D3-0 n-DRV 表示每一个轴的驱动状态，该位为 1 时，表示此轴正在输出驱动脉冲。为 0 时，表示此轴已经结束驱动，当设置 nINPOS 信号有效时，当驱动结束后，nDRV 仍然为 1，当 nINPOS 信号出现一个下降沿则 nDRV 返回为 0。



- D7-4 n-ERR 表示每一个轴的出错状态。在每一个轴 RR2 寄存器的出错位 (D5~D0) 及 RR1 寄存器的错误结束位 (D15~D12) 中的任何位为 1, 该位就为 1。
- D8 I-DRV 表示插补驱动状态。该位为 1 时, 表示正在输出插补驱动脉冲。
- D9 CNEXT 表示可以写入连续插补的下一个数据。在连续插补驱动中, 该位为 1 后, 可以写入下一个节点的参数及插补命令。
- D12~10 ZONEm 在圆弧插补驱动中表示驱动所在的象限。

D12	D11	D10	当前驱动所在的象限
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

D14,D13 BPSC1,0 在位模式插补驱动中表示堆栈计数器 (SC) 的数值。

D14	D13	堆栈计数器的值 (SC)
0	0	0
0	1	1
1	0	2
1	1	3

位模式插补的驱动中, SC=3 的时候, 表示位数据堆栈不能再补充。SC=2 时, 可以给每一个轴补充 16 位; SC=1 时, 可以给每一个轴补充 16 位 2 次; SC=0 时, 表示输出了所有的位数据, 驱动结束。

例: 要读 RR0 寄存器的 X-DRV, Y-DRV 的状态;

```

USB1010_PARA_RR0 RR0;
USB1010_GetRR0Status(
    hDevice,          // 设备句柄
    &RR0);           // RR0 寄存器状态
printf("XDRV=%d ",RR0.XDRV);
printf("YDRV=%d ",RR0.YDRV);

```

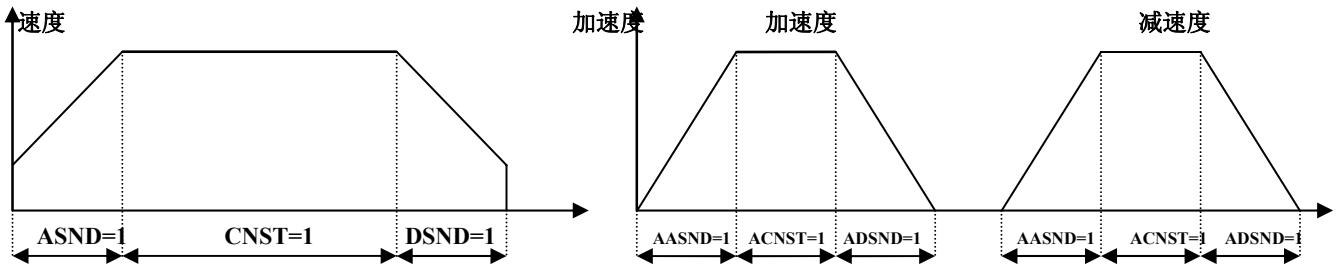
■ RR1 状态寄存器 1

X, Y 轴都有状态寄存器 1。在读 RR1 之前先要指定轴号。具体读哪一位, 方法同上。

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
EMG	ALARM	LMT-	LMT+	IN3	IN2	IN1	IN0	ADSMD	ACNST	AASND	DSND	CNST	ASND	CMP-	CMP+

- D0 CMP+ 表示逻辑/实位计数器和 COMP+寄存器的大小关系。
1: 逻辑/实位计数器 \geq COMP+寄存器
0: 逻辑/实位计数器 $<$ COMP+寄存器
- D1 CMP- 表示逻辑/实位计数器和 COMP—寄存器的大小关系。
1: 逻辑/实位计数器 \geq COMP—寄存器
0: 逻辑/实位计数器 $<$ COMP—寄存器
- D2 ASND 在加/减速驱动 (直线, S 曲线) 中加速时, 为 1
- D3 CNST 在加/减速驱动 (直线, S 曲线) 中定速时, 为 1。
- D4 DSND 在加/减速驱动 (直线, S 曲线) 中减速时, 为 1。
- D5 AASND 在 S 曲线加/减速驱动中, 加速度/减速度增加时, 为 1。
- D6 ACNST 在 S 曲线加/减速驱动中, 加速度/减速度不变时, 为 1。
- D7 ADSND 在 S 曲线加/减速驱动中, 加速度/减速度减少时, 为 1。
- D11~D8 IN3~0 根据外部减速停止信号 (nIN3~0) 驱动停止时, 为 1。
- D12 LMT+ 根据正方向限制信号 (nLMT+) 驱动停止时, 为 1。

- D13 LMT— 根据负方向限制信号 (nLMTM) 驱动停止时, 为 1。
- D14 ALARM 根据伺服马达警报信号 (nALARM) 驱动停止时, 为 1。
- D15 EMG 根据紧急停止信号 (EMGN) 驱动停止时, 为 1。



■ RR3, 4 输入寄存器 1, 2, 具体读哪一位, 方法同上。

输入寄存器 1, 2 表示每一个轴的输入信号状态, 输入信号在低电平时, 表示 0, 高电平时, 表示 1。不使用这些信号的功能时, 可以作为通用输入信号使用。具体读哪一位方法同上。

RR3

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
YALARM	YINPOS	YXXPM	YEXPP	YIN3	YIN2	YIN1	YIN0	XALARM	XINPOS	XEXPM	XEXPP	XIN3	XIN2	XIN1	XIN0

RR4

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
UALARM	UINPOS	UXXPM	UEXPP	UIN3	UIN2	UIN1	UIN0	ZALARM	ZINPOS	ZEXPM	ZEXPP	ZIN3	ZIN2	ZIN1	ZIN0

■ RR5 中断状态位。具体读哪一位, 方法同上。

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
						SYNC	HMEND	DEND	CSTA	CDEC	PBCP	PSMP	BSCM	PBCM	PULSE

- PULSE; 产生一个增量脉冲时
- PBCM; 逻辑/实际位置计数器的值大于等于 COMP-寄存器的值时
- PSCM; 逻辑/实际位置计数器的值小于 COMP-寄存器的值时
- PSCP; 逻辑/实际位置计数器的值小于 COMP+寄存器的值时
- PBCP; 逻辑/实际位置计数器的值大于等于 COMP+寄存器的值
- CDEC; 在加/减速时, 脉冲开始减速时
- CSTA; 在加/减速时, 开始定速时
- DEND; 驱动结束时
- HMEND; 自动原点搜索结束时
- SYNC; 同步产生的中断

某个中断源引发中断后, 此结构体的位就为 1 (前提是必须设置这个中断源有效), 中断输出信号为低电平, CPU 读出发生中断轴的 RR5 寄存器的状态位后, RR5 寄存器的位被清除为 0, 中断输出信号返回无效电平。

第四章 驱动函数库

4.1 驱动函数库函数列表

(每个函数省略了前缀“USB1010_”)

函数名	函数功能
1、设备对象管理函数	
CreateDevice	创建句柄
GetDeviceCount	获得设备总数
GetDeviceCurrentID	取得当前设备的物理 ID 号和逻辑 ID 号
GetDevVersion	获取设备固件及程序版本
ReleaseDevice	释放设备
ResetDevice	复位整个 USB 设备
2、设置控制卡的基本参数函数	
PulseOutMode	设置脉冲输出模式
PulseInputMode	设置脉冲输入模式
SetR	设置倍率(1-500)
SetA	设置加速度(1~1000,000)
SetDec	设置减速度(1~1000,000)
SetAK	设置加速度变化率(1-62500000)
SetSV	设置初始速度 (1~8000)
SetV	设置驱动速度 (1~8000)
SetP	设置定长脉冲数(0~268435455)
SetIP	设置插补终点脉冲数(-8388608~+8388607)
SetC	设置圆心坐标(脉冲数)(-8388608~+8388607)
SetLP	设置逻辑位置计数器(-2147483648~+2147483647)
SetEP	设置实位计数器(-2147483648~+2147483647)
SetAccofst	设置加速计数器偏移(-32768~+32768)
3、直线 S 曲线初始化、启动函数	
InitLVDV	初始化连续, 定长脉冲驱动
StartLVDV	启动连续, 定长脉冲驱动
Start2D	启动 2 轴同时驱动
4、直线插补初始化、启动函数	
InitLineInterpolation	初始化任意 2 轴直线插补驱动
StartLineInterpolation	启动任意 2 轴直线插补驱动
5、正反方向圆弧插补初始化、启动函数	
InitCWInterpolation	初始化正反方向圆弧插补驱动
StartCWInterpolation	启动正方向圆弧插补驱动
StartCCWInterpolation	启动反方向圆弧插补驱动
5、位插补相关函数	
InitBitInterpolation	初始化位插补参数
AutoBitInterpolation	自动执行位插补
ReleaseBitInterpolation	释放 BP 寄存器
SetBP	设置位插补数据

StartBitInterpolation	启动位插补
BPWait	等待位插补的下一组数据设定
ClearBPData	清除 BP 寄存器数据
6、连续插补相关函数	
NextWait	等待连续插补下一个节点写入数据命令允许
7、外部信号启动电机定长驱动、连续驱动	
SetOutEnableDV	外部控制定量驱动(下降沿有效)
SetOutEnableLV	外部控制连续驱动(保持低电平有效)
8、软件限位设置、清除	
SetPDirSoftwareLimit	设置正方向软件限位
SetMDirSoftwareLimit	设置反方向软件限位
ClearSoftwareLimit	清除软件限位
9、设置外部硬件限位、停止信号、报警信号、定位信号有效及清除	
SetLMTEnable	设置外部越限信号有效及停止方式
SetStopEnable	设置外部停止信号有效
SetStopDisable	设置外部停止信号无效
SetALARMEnable	设置伺服报警信号有效
SetALARMDisable	设置伺服报警信号无效
SetINPOSEnable	设置伺服马达定位完毕输入信号有效
SetINPOSDisable	设置伺服马达定位完毕输入信号无效
11、减速函数设置	
DecValid	减速有效
DecInvalid	减速无效
DecStop	减速停止
InstStop	立即停止
AutoDec	自动减速有效
HanDec	手动减速有效并设定手动减速点
12、读逻辑计数器、实位计数器、当前速度、加速度	
ReadLP	读逻辑计数器
ReadEP	读实位计数器
ReadCV	读当前速度
ReadCA	读当前加速度
13、读状态寄存器的位状态	
ReadRR	读 RR 寄存器
GetRR0Status	获得主状态寄存器 RR0 的位状态
GetRR1Status	获得状态寄存器 RR1 的位状态
GetRR2Status	获得状态寄存器 RR2 的位状态
GetRR3Status	获得状态寄存器 RR3 的位状态
GetRR4Status	获得状态寄存器 RR4 的位状态
14、设置通用输出和读通用输入	
SetOutput	设置通用输出电平
ReadTYPE	读通用输入信号电平

4.2 驱动函数库说明

4.2.1 设备对象管理函数

1. HANDLE CreateDevice(int DeviceID)

功能: 该函数负责创建 USB 设备对象, 并返回其设备对象句柄。

参数: DeviceID 设备 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 USB 设备时, 我们的驱动程序将以该设备的“基本名称”与 DeviceID 标识值为名称后缀的标识符来确认和管理该设备。比如若用户往 Windows 系统中加入第一个 USB1010 模板时, 驱动程序则以“USB1010”作为基本名称, 再以 DeviceID 的初值组合成该设备的标识符“USB1010-0”来确认和管理这第一个设备, 若用户接着再添加第二个 USB1010 模板时, 则系统将以“USB1010-1”来确认和管理第二个设备, 若再添加, 则以此类推。所以当用户要创建设备句柄管理和操作第一个 USB 设备时, DeviceID 应置 0, 第二个应置 1, 也以此类推。但默认值为 0。

返回值: 如果执行成功, 则返回设备对象句柄; 如果没有成功, 则返回错误码 INVALID_HANDLE_VALUE。由于此函数已带容错处理, 即若出错, 它会自动弹出一个对话框告诉您出错的原因。您只需要对此函数的返回值作一个条件处理即可, 别的任何事情您都不必做。

相关函数: [ReleaseDevice](#)

Visual C++程序举例

```

:
HANDLE hDevice; // 定义设备对象句柄
hDevice=CreateDevice ( 0 ); // 创建设备对象,并取得设备对象句柄
if(hDevice==INVALID_HANDLE_VALUE); // 判断设备对象句柄是否有效
{
    return; // 退出该函数
}
)
```

2. int GetDeviceCount(HANDLE hDevice)

功能: 取得 USB1010 设备的数量。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 返回系统中 USB1010 的数量。

3. int GetDeviceCurrentID(HANDLE hDevice, PLONG DeviceLgcID, PLONG DevicePhysID)

功能: 取得当前设备的物理 ID 号和逻辑 ID 号。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

DeviceLgcID 逻辑 ID 号。

DevicePhysID 物理 ID 号。

返回值: 返回当前设备的物理 ID 号和逻辑 ID 号。

相关函数: [CreateDevice](#)

4. BOOL GetDevVersion (HANDLE hDevice, PULONG pulFmwVersion, PULONG pulDriverVersion)

功能: 获取设备固件及程序版本。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pulFmwVersion 固件版本。

pulDriverVersion 驱动版本。

返回值: 返回当前设备的固件及程序版本。

相关函数: [CreateDevice](#)

5. BOOL ReleaseDevice(HANDLE hDevice)

功能: 释放设备对象所占用的系统资源及设备对象自身。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#)

需要注意的是, CreateDevice 必须和 ReleaseDevice 函数一一对应, 即当您执行了一次 CreateDevice 后, 再一次执行这些函数前, 必须执行一次 ReleaseDevice 函数, 以释放由 CreateDevice 占用的系统软硬件资源, 如 DMA 控制器, 系统内存等。只有这样, 当您再次调用 CreateDevice 函数时, 那些软硬件资源才可被再次使用。

6. BOOL Reset(HANDLE hDevice)

功能: 复位整个 USB 设备。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#)

4.2.2 设置控制卡的基本参数函数

1. BOOL PulseOutMode (HANDLE hDevice, LONG AxisNum, LONG Mode)

功能: 设置脉冲输出模式。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

Mode 脉冲输出模式 USB1010_CWCCW: CW/CCW 模式 USB1010_CPDIR: CP/DIR 模式。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

2. BOOL PulseInputMode(HANDLE hDevice, LONG AxisNum, LONG Mode)

功能: 设置脉冲输入模式。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

Mode 脉冲输出模式 USB1010_CWCCW: CW/CCW 模式 USB1010_CPDIR: CP/DIR 模式。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

3. BOOL SetR (HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置倍率, 范围是(1-500)。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

Data 加速度数据, 范围(125-1000,000)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际加速度 (PPS/SEC) = 设定的加速度 × 倍率。

4. BOOL SetA(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置指定轴加速度(Set acceleration), 范围(125~1000,000), 实际的加速度取值应为 125 的整数倍, 若设置值不为 125 的整数倍, 实际取值应为最接近 125 的整数倍的值。



参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Data 加速度数据, 范围(125~1000,000), 实际的加速度取值应为 125 的整数倍。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际加速度 (PPS/SEC) = 设定的加速度 × 倍率。

5. **BOOL SetDec(HANDLE hDevice, LONG AxisNum, LONG Data)**

功能: 设置指定轴减速度(Set deceleration), 范围(125~1000,000), 实际的减速度取值应为 125 的整数倍, 若设置值不为 125 的整数倍, 实际取值应为最接近 125 的整数倍的值。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Data 减速度数据, 范围(125~1000,000), 实际的减速度取值应为 125 的整数倍。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际减速度 (PPS/SEC) = 设定的减速度 × 倍率。

6. **BOOL SetAK (HANDLE hDevice, LONG AxisNum, LONG Data)**

功能: 设置指定轴加速度变化率, 范围(1~62500000)。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Data 加速度变化率数据, 范围(1~62500000)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际初始速度 (PPS) = 设定的初始速度 × 倍率。

7. **BOOL SetSV(HANDLE hDevice, LONG AxisNum, LONG Data)**

功能: 设置指定轴初始速度(Set start speed), 范围(1~8,000)。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Data 初始速度数据, 范围(1~8,000)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际初始速度 (PPS) = 设定的初始速度 × 倍率。

8. **BOOL SetV(HANDLE hDevice, LONG AxisNum, LONG Data)**

功能: 设置指定轴驱动速度(Set drive speed), 范围(1~8,000)。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Data 驱动速度数据, 范围(1~8,000)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 电机运行时的实际驱动速度 (PPS) = 设定的驱动速度 × 倍率。

9. **BOOL SetP(HANDLE hDevice, LONG AxisNum, LONG Data)**

功能: 设置指定轴定长脉冲数(Set pulse), 范围(0~268435455)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 定长脉冲数数据, 范围(0~268435455)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

10. BOOL SetIP (HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置插补终点脉冲数, 范围(-8388608~+8388607)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 定长脉冲数数据, 范围(-8388608~+8388607)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

11. BOOL SetC(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置指定轴圆弧圆心坐标 (脉冲数) (Set centre), 范围(-8388608~+8388607)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 圆心坐标数据, 范围(-8388608~+8388607)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

12. BOOL SetLP(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置指定轴逻辑位置计数器(Set Logic Position), 范围(-2147483648~+2147483647)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 逻辑位置计数器数据, 范围(-2147483648~+2147483647)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。
备注: 一般情况设定为 0。

13. BOOL SetEP(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置指定轴实际位置计数器 (Set Effect Position), 范围(-2147483648~+2147483647)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 实际位置计数器数据, 范围(-2147483648~+2147483647)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。
备注: 一般情况设定为 0。

14. BOOL SetAccofst(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 设置指定轴加速计数器偏移(Set Acceleration Counteroffset), 范围(0-65535)。
参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
 AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
 Data 加速计数器偏移数据, 范围(0-65535)。
返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。
备注: 在直线加减速驱动时不需要设定, 在 S 曲线驱动时根据情况设定。

4.2.3 直线 S 曲线初始化、启动函数

1. BOOL InitLVDV (HANDLE hDevice, PUSB1010_PARA_DataList pDL, PUSB1010_PARA_LCDData pLC)

功能: 初始化指定轴定长或连续驱动。

参数:

hDevice	设备对象句柄，它应由 CreateDevice 创建。
pDL	公共参数结构体指针，参数包括： Multiple 倍率，范围（1~500） StartSpeed 初始速度，范围（1~8000） DriveSpeed 驱动速度，范围（1~8000） Acceleration 加速度，范围（125~1000,000），实际的取值应为 125 的整数倍 Deceleration 减速度，范围（125~1000,000），实际的取值应为 125 的整数倍 AccelerationK 加速度变化率，范围(954~62500000)（仅 S 曲线驱动时设定，如果要加 速度和减速曲线不一样，还需要设定减速度变化率，这时加速度和减速度都必须设置为 8000）
pLC	直线和 S 曲线参数结构体指针，参数包括： AxisNum 轴号(USB1010_XAXIS:X 轴，USB1010_YAXIS:Y 轴)。 LV_DV 连续驱动、定长驱动选择 USB1010_LV:连续驱动； USB1010_DV:定长驱动 Mode 脉冲输出模式 USB1010_CWCCW:CW/CCW 模式； USB1010_CPDIR:CP/DIR 模式 Line_Curve 直线 S 曲线选择 USB1010_LINE:直线驱动；USB1010_Curve:S 曲线驱动 Direction 转动方向 USB1010_PDIRECTION:正方向转动； USB1010_MDIRECTION:反方向转动 nPulseNum 输出脉冲数，范围(0~268435455)（定长脉冲驱动时设定）

返回值: 若成功，则返回 TRUE，否则返回 FALSE。

备注: 在直线加减速度驱动时不需要设定加速度变化率，在 S 曲线驱动时必须设定，如果 S 曲线中加速和减速曲线不一样，就必须设置减速度变化率，并且加速度和减速度都要设置成 8000。调用例子请参看使用 USB1010_StartLVDV 定长、连续脉冲驱动函数启动电机。

2. BOOL StartLVDV(HANDLE hDevice, LONG AxisNum)

功能: 启动单轴直线 S 曲线驱动。

参数: hDevice 设备对象句柄，它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴，USB1010_YAXIS:Y 轴)。

返回值: 若成功，则返回 TRUE，否则返回 FALSE。

3. BOOL Start2D(HANDLE hDevice)

功能: 启动 2 轴同时驱动。

参数: hDevice 设备对象句柄，它应由 CreateDevice 创建。

返回值: 若成功，则返回 TRUE，否则返回 FALSE。

4.2.4 直线插补初始化、启动函数

1. BOOL InitLineInterpolation(HANDLE hDevice, PUSB1010_PARA_DataList pDL, PUSB1010_PARA_LineData pLD)

功能: 初始化直线插补。

参数: hDevice 设备对象句柄，它应由 CreateDevice 创建。
pDL 公共参数结构体指针，参数包括：

Multiple	倍率, 范围 (1~500)
StartSpeed	初始速度, 范围 (1~8000)
DriveSpeed	驱动速度, 范围 (1~8000)
Acceleration	加速度, 范围 (125~1000,000), 实际的取值应为 125 的整数倍
Deceleration	减速度, 范围 (125~1000,000), 实际的取值应为 125 的整数倍
AccelerationK	加速度变化率, 范围(954~62500000) (仅 S 曲线驱动时设定, 如果要加速度和减速曲线不一样, 还需要设定减速度变化率, 这时加速度和减速度都必须设置为 8000)
pLD	直线插补参数结构体指针, 参数包括:
Line_Curve;	运动方式 USB1010_LINE:直线 USB1010_CURVE: 曲线)
ConstantSpeed;	固定线速度 USB1010_NOCONSTAND 不固定线速度 USB1010_CONSTAND: 固定线速度
nXAxisPulseNum;	X 轴终点脉冲数 (-8388608~8388607)
nYAxisPulseNum;	Y 轴终点脉冲数 (-8388608~8388607)

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 在直线加减速驱动时不需要设定加速度变化率, 在 S 曲线驱动时必须设定。调用例子请参看使用 USB1010_LINEInterpolation 函数, 启动两轴直线插补驱动。

2. BOOL StartLineInterpolation (HANDLE hDevice)

功能: 启动直线曲线插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.5 正反方向圆弧插补初始化、启动函数

1. BOOL InitCWInterpolation (HANDLE hDevice, PUSB1010_PARA_DataList pDL, PUSB1010_PARA_CircleData pCD)

功能: 初始化正反方向圆弧插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pDL	公共参数结构体指针, 参数包括:
Multiple	倍率, 范围 (1~500)
StartSpeed	初始速度, 范围 (1~8000)
DriveSpeed	驱动速度, 范围 (1~8000)
Acceleration	加速度, 范围 (125~1000,000), 实际的取值应为 125 的整数倍
pCD	正反方向圆弧插补参数结构体指针, 参数包括:
ConstantSpeed	是否固定线速 USB1010_CONSTAND:固定 USB1010_NOCONSTAND: 不固定
XCenter	X 轴圆心坐标(脉冲数-8388608~8388607)
YCenter	Y 轴圆心坐标(脉冲数-8388608~8388607)
XPulse	X 轴终点坐标(脉冲数-8388608~8388607)
YPulse	Y 轴终点坐标(脉冲数-8388608~8388607)

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 圆弧插补一般使用直线加减速驱动, 不能使用 S 曲线驱动, 并且一般为定速驱动。当需要使用加减速驱动时, 必须设定手动减速点。自动减速无效。调用例子请参看使用 USB1010_InitCWInterpolation 函数, 启动任意两轴正方向圆弧插补驱动。

2. BOOL StartCWInterpolation (HANDLE hDevice)

功能: 启动正方向圆弧插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

**3. BOOL StartCCWInterpolation (HANDLE hDevice)**

功能: 启动反方向圆弧插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.6 位插补相关函数**1. BOOL InitBitInterpolation (HANDLE hDevice,
PUSB1010_PARA_DataList pDL)**

功能: 初始化位插补参数。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pDL 公共参数结构体指针, 参数包括:

Multiple 倍率, 范围 (1~500)

StartSpeed 初始速度, 范围 (1~8000)

DriveSpeed 驱动速度, 范围 (1~8000)

Acceleration 加速度, 范围 (125~1000,000), 实际的取值应为 125 的整数倍

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 位插补一般使用直线加减速驱动, 不能使用 S 曲线驱动, 并且一般为定速驱动。当需要使用加减速驱动时, 必须设定手动减速点。

**2. BOOL AutoBitInterpolation (HANDLE hDevice,
PUSHORT pBuffer,
UINT nCount)**

功能: 自动执行位插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pBuffer 位插补数组指针。

nCount 数据组数个数。

返回值: 该线程自动把数据数组里的位插补数据设置到位插补数据堆栈并执行位插补。若成功, 则返回 TRUE, 否则返回 FALSE。

3. BOOL ReleaseBitInterpolation (HANDLE hDevice)

功能: 释放位插补寄存器。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

**4. BOOL SetBP (HANDLE hDevice,
LONG BP1PData,
LONG BP1MData,
LONG BP2PData,
LONG BP2MData)**

功能: 设置位插补数据 BP1P, BP1M, BP2P, BP2M。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

BP1PData X 轴正方向驱动数据。

BP1MData X 轴反方向驱动数据。

BP2PData Y 轴正方向驱动数据。

BP2MData Y 轴反方向驱动数据。

返回值: 若成功, 则返回 TRUE, 设备句柄无效返回 FALSE。

5. BOOL StartBitInterpolation (HANDLE hDevice)

功能: 启动位插补。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

6. BOOL BPWait(HANDLE hDevice)

功能: 等待 BP 插补的下一个数据设定, 当堆栈计数器 (SC) 为 2 时就可以进行下一个数据设定。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 设备句柄无效返回 FALSE。

7. BOOL ClearBPData (HANDLE hDevice)

功能: 清除 BP 寄存器数据。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.7 连续插补相关函数

1. BOOL NextWait(HANDLE hDevice)

功能: 等待连续插补下一个数据, 当不能设定下一个节点数据及命令时, 一直等待, 能设定时自动跳出。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若不能设定下一个节点数据及命令时, 一直等待, 直到能设定时, 返回 TRUE, 设备句柄无效返回 FALSE。

4.2.8 外部信号启动电机定长驱动、连续驱动

1. BOOL SetOutEnableDV (HANDLE hDevice, LONG AxisNum)

功能: 外部控制定量驱动(下降沿有效, 平常为高电平), 当 nEXPP, nEXPM 引脚上有低电平, 即出现一个下降沿时, 启动电机按照设定好的参数进行定长脉冲驱动。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

2. BOOL SetOutEnableLV (HANDLE hDevice, LONG AxisNum)

功能: 外部控制连续驱动(保持低电平有效), 当 nEXPP, nEXPM 引脚上保持低电平期间, 启动电机按照设定好的参数进行连续驱动。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 调用例子请参看。

4.2.9 设置软件限位有效和无效

1. BOOL SetPDirSoftwareLimit (HANDLE hDevice, LONG AxisNum, LONG LogicFact, LONG Data)

功能: 设置正方向软件限位。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

LogicFact 逻辑位置计数器实际位置计数器选择 USB1010_Logic: 逻辑位置计数器;
USB1010_Fact: 实际位置计数器。

Data 限位脉冲数, 范围(-2147483648~+2147483647)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 当软件限位有效时, 就减速停止。

2. BOOL SetMDirSoftwareLimit (HANDLE hDevice, LONG AxisNum, LONG LogicFact,

**LONG Data)**

功能: 设置反方向软件限位。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
LogicFact 逻辑位置计数器实际位置计数器选择 USB1010_Logic: 逻辑位置计数器;
 USB1010_Fact:实际位置计数器。
Data 限位脉冲数, 范围(-2147483648~+2147483647)

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

备注: 当软件限位有效时, 就减速停止。

3. BOOL ClearSoftwareLimit (HANDLE hDevice, LONG AxisNum)

功能: 清除软件限位。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.10 设置外部输入信号的有效和无效

1. BOOL SetLMTEnable (HANDLE hDevice, LONG AxisNum, LONG StopMode)

功能: 设置正向外部超限信号有效及停止方式。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
StopMode 停止模式 (USB1010_DECSTOP: 减速停止; USB1010_SUDDENSTOP: 立即停止)

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

2. BOOL SetStopEnable (HANDLE hDevice, LONG AxisNum, LONG StopNum)

功能: 设置外部停止信号有效。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
StopNum 停止号(USB1010_STOP0: 停止信号 0, USB1010_STOP1: 停止信号 1)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

3. BOOL SetStopDisable (HANDLE hDevice, LONG AxisNum)

功能: 设置外部停止信号无效。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4. BOOL SetALARMEnable (HANDLE hDevice, LONG AxisNum)

功能: 设置伺服报警信号有效。

参数: **hDevice** 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

5. BOOL SetALARMDisable (HANDLE hDevice, LONG AxisNum)

功能: 设置伺服报警信号无效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

6. BOOL SetINPOSEnable (HANDLE hDevice, LONG AxisNum)

功能: 设置伺服马达定位完毕输入信号有效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

7. BOOL SetINPOSDisable (HANDLE hDevice, LONG AxisNum)

功能: 设置伺服马达定位完毕输入信号无效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.11 减速函数设置

1. BOOL DecValid (HANDLE hDevice)

功能: 减速有效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

2. BOOL DecInvalid (HANDLE hDevice)

功能: 减速无效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

3. BOOL DecStop (HANDLE hDevice, LONG AxisNum)

功能: 减速停止。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4. BOOL InstStop (HANDLE hDevice, LONG AxisNum)

功能: 立即停止。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

5. BOOL AutoDec(HANDLE hDevice, LONG AxisNum)

功能: 自动减速有效。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。



6. BOOL HanDec(HANDLE hDevice, LONG AxisNum, LONG Data)

功能: 手动减速有效并设定手动减速点, 范围(0~268435455)。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

4.2.12 读电机状态: 逻辑计数器、实际位置计数器、当前速度、加/减速度

1. LONG ReadLP(HANDLE hDevice, LONG AxisNum)

功能: 读逻辑计数器。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回逻辑位置计数器的值, 错误则返回 FALSE。

2. LONG ReadEP(HANDLE hDevice, LONG AxisNum)

功能: 读实际位置计数器。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回实际位置计数器的值, 错误则返回 FALSE。

3. LONG ReadCV(HANDLE hDevice, LONG AxisNum)

功能: 读当前速度。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回当前速度的值, 轴号错误返回 FALSE。

备注: 当前读出的速度, 范围在 (1~8000), 实际速度 = 读出的速度值×倍率

4. LONG ReadCA(HANDLE hDevice, LONG AxisNum)

功能: 读当前加速度。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回当前速度的值, 轴号错误返回 FALSE。

备注: 当前读出的加速度, 范围在 (1~8000), 实际加速度 = 读出的加速度值×125×倍率

4.2.13 读状态寄存器的位状态

1. LONG ReadRR (HANDLE hDevice, LONG AxisNum, LONG Num)

功能: 读 RR 寄存器。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。
AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。
Num 寄存器号。

返回值: 若成功, 则返回 RR 的值, 轴号错误返回 FALSE。

2. BOOL GetRR0Status(HANDLE hDevice, PUSB1010_PARA_RR0 pPara)

功能: 获得 RR0 寄存器的位状态。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pPara RR0 状态的参数结构体, 共有 15 个成员变量, 分别对应于 RR0 寄存器的各个状态位。
如果 pPara-> XDRV 为“1”则表示 X 轴正在输出脉冲。其他同理。

返回值: 若成功, 返回 TRUE, 其 pPara 中的值有效; 否则返回 FALSE, 其 pPara 中的值无效。

3. BOOL GetRR1Status(HANDLE hDevice, LONG AxisNum, PUSB1010_PARA_RR1 pPara)

功能: 获得 RR1 寄存器的位状态。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pPara RR1 状态的参数结构体, 共有 16 个成员变量, 分别对应于 RR1 寄存器的各个状态位。
如果 pPara-> CMPP 为“1”则表示逻辑/实位计数器 \geq COMP+。其他同理。

返回值: 若成功, 返回 TRUE, 其 pPara 中的值有效; 否则返回 FALSE, 其 pPara 中的值无效。

4. BOOL GetRR2Status (HANDLE hDevice, LONG AxisNum, PUSB1010_PARA_RR2 pPara)

功能: 获得 RR2 寄存器的位状态。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pPara RR2 状态的参数结构体, 共有 12 个成员变量, 分别对应于 RR2 寄存器的各个状态位。
如果 pPara-> ALARM 为“1”则表示外部伺服马达报警信号(nALARM)设置为有效并处于有效状态。
其他同理。

返回值: 若成功, 返回 TRUE, 其 pPara 中的值有效; 否则返回 FALSE, 其 pPara 中的值无效。

5. BOOL GetRR3Status (HANDLE hDevice, PUSB1010_PARA_RR3 pPara)

功能: 获得 RR3 寄存器的位状态。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pPara RR3 状态的参数结构体, 共有 16 个成员变量, 分别对应于 RR3 寄存器的各个状态位。
如果 pPara-> XIN0 为“1”则表示外部停止信号 XIN0 的电平为高电平。其他同理。

返回值: 若成功, 返回 TRUE, 其 pPara 中的值有效; 否则返回 FALSE, 其 pPara 中的值无效。

6. BOOL GetRR4Status (HANDLE hDevice, PUSB1010_PARA_RR4 pPara)

功能: 获得 RR4 寄存器的位状态。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

pPara RR4 状态的参数结构体, 共有 16 个成员变量, 分别对应于 RR4 寄存器的各个状态位。
如果 pPara-> ZIN0 为“1”则表示外部停止信号 ZIN0 的电平为高电平。其他同理。

返回值: 若成功, 返回 TRUE, 其 pPara 中的值有效; 否则返回 FALSE, 其 pPara 中的值无效。

4.2.14 设置通用输出和读通用输入

1. BOOL SetOutput (HANDLE hDevice, LONG XOut1, LONG XOut0, LONG YOut1, LONG YOut0)

功能: 设置通用输出电平。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

XOut1 X 轴 OUT1 1: XOUT1 为高电平 0: XOUT0 为低电平。



XOut0 X 轴 OUT0 1: XOUT1 为高电平 0: XOUT0 为低电平。

YOut1 Y 轴 OUT1 1: YOUT1 为高电平 0: YOUT0 为低电平。

YOut0 Y 轴 OUT0 1: YOUT1 为高电平 0: YOUT0 为低电平。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

2. LONG ReadTYPE (HANDLE hDevice, LONG AxisNum)

功能: 读通用输入信号电平。

参数: hDevice 设备对象句柄, 它应由 CreateDevice 创建。

AxisNum 轴号(USB1010_XAXIS:X 轴, USB1010_YAXIS:Y 轴)。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

第五章 硬件参数结构

5.1 公用参数介绍 (USB1010_PARA_DataList)

```
typedef struct _USB1010_PAPA_DataList
{
    LONG Multiple;           // 倍率 (1~500)
    LONG StartSpeed;        // 初始速度(1~8000)
    LONG DriveSpeed;        // 驱动速度(1~8000)
    LONG Acceleration;      // 加速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG Deceleration;      // 减速度(125~1000000), 实际的取值应为 125 的整数倍
    LONG AccelerationK;     // 加速度变化率(954~62500000)
} USB1010_PARA_DataList, *PUSB1010_PARA_DataList;
```

Multiple 倍率 (1~500)。

StartSpeed 初始速度(1~8000)。

DriveSpeed 驱动速度(1~8000)。

Acceleration 加速度(125~1000000), 实际的加速度取值应为 125 的整数倍, 若设置值不为 125 的整数倍, 实际取值应为最接近 125 的整数倍的值。例如, 设置加速度为 300, 倍率为 1, 则实际的加速度为 250。

Deceleration 减速度(125~1000000), 实际的减速度取值应为 125 的整数倍, 若设置值不为 125 的整数倍, 实际取值应为最接近 125 的整数倍的值。

AccelerationK 加速度变化率(954~62500000)。

5.2 直线和 S 曲线参数介绍 (USB1010_PAPA_LCData)

```
typedef struct _USB1010_PAPA_LCData
{
    LONG AxisNum;           // 轴号 (X 轴 | Y 轴 | X、Y 轴)
    LONG LV_DV;            // 驱动方式(连续 | 定长 )
    LONG Mode;             // 脉冲方式 (CW/CCW 方式 | CP/DIR 方式)
    LONG Line_Curve;       // 运动方式 (直线 | 曲线)
    LONG Direction;        // 运动方向 (正方向 | 反方向)
    LONG nPulseNum;        // 定量输出脉冲数(0~268435455)
} USB1010_PARA_LCData, *PUSB1010_PARA_LCData;
```

AxisNum 轴号选择, 取值如下表:

常量名	常量值	功能定义
USB1010_XAXIS	0x0	X 轴
USB1010_YAXIS	0x1	Y 轴
USB1010_XYAXIS	0x2	X、Y 轴

LV_DV 驱动方式选择, 取值如下表:

常量名	常量值	功能定义
USB1010_DV	0x0	定长驱动
USB1010_LV	0x1	连续驱动

Mode 脉冲输出方式选择, 取值如下表:

常量名	常量值	功能定义
-----	-----	------



USB1010_CWCCW	0x0	CW/CCW 方式
USB1010_CPDIR	0x1	CP/DIR 方式

Line_Curve 运动方式选择，取值如下表：

常量名	常量值	功能定义
USB1010_LINE	0x0	直线运动
USB1010_CURVE	0x1	S 曲线运动

Direction 运动方向选择，取值如下表：

常量名	常量值	功能定义
USB1010_MDIRECTION	0x0	反方向
USB1010_PDIRECTION	0x1	正方向

nPulseNum 定量输出脉冲数(0~268435455)。

5.3 直线插补和固定线速度直线插补参数介绍 (USB1010_PAPA_LineData)

```
typedef struct _USB1010_PAPA_LineData
{
    LONG Line_Curve;           // 运动方式 (直线 | 曲线)
    LONG ConstantSpeed;       // 固定线速度 (不固定线速度 | 固定线速度)
    LONG nXAxisPulseNum;      // X 轴终点脉冲数 (-8388608~8388607)
    LONG nYAxisPulseNum;      // Y 轴终点脉冲数 (-8388608~8388607)
} USB1010_PAPA_LineData, *PUSB1010_PAPA_LineData;
```

Line_Curve 运动方式选择，取值如下表：

常量名	常量值	功能定义
USB1010_LINE	0x0	直线运动
USB1010_CURVE	0x1	S 曲线运动

ConstantSpeed 固定线速度选择，取值如下表：

常量名	常量值	功能定义
USB1010_NOCONSTAND	0x0	不固定线速度
USB1010_CONSTAND	0x1	固定线速度

nXAxisPulseNum X 轴终点脉冲数 (-8388608~8388607)。

nYAxisPulseNum Y 轴终点脉冲数 (-8388608~8388607)。

5.4 正反方向圆弧插补参数介绍 (USB1010_PAPA_CircleData)

```
typedef struct _USB1010_PAPA_CircleData
{
    LONG ConstantSpeed;       // 固定线速度 (不固定线速度 | 固定线速度)
    LONG XCenter              // X 轴圆心坐标(脉冲数-8388608~8388607)
    LONG YCenter              // Y 轴圆心坐标(脉冲数-8388608~8388607)
    LONG XPulse               // X 轴终点坐标(脉冲数-8388608~8388607)
    LONG YPulse               // Y 轴终点坐标(脉冲数-8388608~8388607)
} USB1010_PAPA_CircleData, *PUSB1010_PAPA_CircleData;
```

ConstantSpeed 固定线速度选择，取值如下表：

常量名	常量值	功能定义
-----	-----	------

USB1010_NOCONSTAND	0x0	不固定线速度
USB1010_CONSTAND	0x1	固定线速度

- Center1 X 轴圆心坐标(脉冲数-8388608~8388607)。
- Center2 Y 轴轴圆心坐标(脉冲数-8388608~8388607)。
- Pulse1 X 轴终点坐标(脉冲数-8388608~8388607)。
- Pulse2 Y 轴轴终点坐标(脉冲数-8388608~8388607)。

5.5 状态寄存器 RR0 参数介绍 (USB1010_PARA_RR0)

```
typedef struct _USB1010_PARA_RR0
{
    UINT XDRV;           // X 轴的驱动状态 1: 正在输出脉冲 0: 停止驱动
    UINT YDRV;           // Y 轴的驱动状态 1: 正在输出脉冲 0: 停止驱动
    UINT XERROR;        // X 轴的出错状态 X 轴的 RR2 寄存器的任何一位为 1, 此位就为 1
    UINT YERROR;        // Y 轴的出错状态 Y 轴的 RR2 寄存器的任何一位为 1, 此位就为 1
    UINT IDRV;          // 插补驱动状态 1: 正处于插补模式 0: 未处于插补模式
    UINT CNEXT;         // 表示可以写入连续插补的下一个数据 1: 可以写入 0: 不可以写入
                        // 当设置连续插补中断使能后, CNEXT 为 1 表示产生了中断, 在中断程序写入
                        // 下一个插补命令后, 该位清零并且中断信号回到高电平
    UINT ZONE0;         // ZONE2、ZONE1、ZONE0 表示在圆弧插补驱动中所在的象限
    UINT ZONE1;         // 000 : 第 0 象限 001: 第 1 象限 010: 第 2 象限 011: 第 3 象限
    UINT ZONE2;         // 100 : 第 4 象限 101: 第 5 象限 110: 第 6 象限 111: 第 7 象限
    UINT BPSC0;         // BPSC1、BPSC0 表示在位插补驱动中堆栈计数器(SC)的数值
    UINT BPSC1;         // 00: 0 01: 1 10: 2 11: 3
                        // 设置位插补中断使能后, 当 SC 的值由 2 变为 1 时, 产生中断,
                        // 当向位插补堆栈写入新的数据或调用 USB1010_ClearInterruptStatus, 中断解
                        // 除。
} USB1010_PARA_RR0,*PUSB1010_PARA_RR0;
```

5.14 状态寄存器 RR1 参数介绍 (USB1010_PARA_RR1)

```
typedef struct _USB1010_PARA_RR1
{
    UINT CMPP;          // 表示逻辑/实位计数器和 COMP+ 寄存器的大小关系 1: 逻辑/实位计数器
                        // ≥COMP+ 0: 逻辑/实位计数器 < COMP+
    UINT CMPM;          // 表示逻辑/实位计数器和 COMP- 寄存器的大小关系 1: 逻辑/实位计数器 <
                        // COMP- 0: 逻辑/实位计数器 ≥ COMP-
    UINT ASND;          // 在加/减速驱动中加速时, 为 1
    UINT CNST;          // 在加/减速驱动中定速时, 为 1
    UINT DSND;          // 在加/减速驱动中减速时, 为 1
    UINT AASND;         // 在 S 曲线加/减速驱动中, 加速度/减速度增加时, 为 1
    UINT ACNST;         // 在 S 曲线加/减速驱动中, 加速度/减速度不变时, 为 1
    UINT ADSND;         // 在 S 曲线加/减速驱动中, 加速度/减速度减少时, 为 1
    UINT STOP0;         // 外部停止信号 STOP0 有效使驱动停止时, 为 1
    UINT STOP1;         // 外部停止信号 STOP1 有效使驱动停止时, 为 1
    UINT LMTP;          // 外部正方向限制信号(nLMTP)有效使驱动停止时, 为 1
    UINT LMTM;          // 外部反方向限制信号(nLMTM)有效使驱动停止时, 为 1
    UINT ALARM;         // 外部伺服马达报警信号(nALARM)有效使驱动停止时, 为 1
    UINT EMG;           // 外部紧急停止信号(EMGN)使驱动停止时, 为 1
} USB1010_PARA_RR1,*PUSB1010_PARA_RR1;
```



5.15 状态寄存器 RR2 参数介绍 (USB1010_PARA_RR2)

```
typedef struct _USB1010_PARA_RR2
{
    UINT SLMTP;           // 设置正方向软件限位后，在正方向驱动中，逻辑/实位计数器大于 COMP+寄存
器值时，为 1
    UINT SLMTM;           // 设置反方向软件限位后，在反方向驱动中，逻辑/实位计数器小于 COMP-寄存
器值时，为 1
    UINT HLMTP;           // 外部正方向限制信号(nLMTP)处于有效电平时，为 1
    UINT HLMTM;           // 外部反方向限制信号(nLMTM)处于有效电平时，为 1
    UINT ALARM;           // 外部伺服马达报警信号(nALARM)设置为有效并处于有效状态时，为 1
    UINT EMG;             // 外部紧急停止信号处于低电平时，为 1
} USB1010_PARA_RR2,*PUSB1010_PARA_RR2;
```

5.16 状态寄存器 RR3 参数介绍 (USB1010_PARA_RR3)

```
typedef struct _USB1010_PARA_RR3
{
    UINT XSTOP0;          // 外部停止信号 XSTOP0 的电平状态 1: 高电平 0: 低电平
    UINT XSTOP1;          // 外部停止信号 XSTOP1 的电平状态 1: 高电平 0: 低电平
    UINT EMG;             // 外部紧急停止信号 EMG 的电平状态 1: 高电平 0: 低电平
    UINT XEXPP;           // 外部正方向点动输入信号 XEXPP 的电平状态 1: 高电平 0: 低电平
    UINT XEXPM;           // 外部反方向点动输入信号 XEXPM 的电平状态 1: 高电平 0: 低电平
    UINT XINPOS;          // 外部伺服电机到位信号 XINPOS 的电平状态 1: 高电平 0: 低电平
    UINT XALARM;          // 外部伺服马达报警信号 XALARM 的电平状态 1: 高电平 0: 低电平
    UINT XIN0;            // 外部输入信号 XIN0 的电平状态 1: 高电平 0: 低电平
    UINT XIN1;            // 外部输入信号 XIN1 的电平状态 1: 高电平 0: 低电平
    UINT XLMTM;           // 外部正方向限制信号 XLMTM 的电平状态 1: 高电平 0: 低电平
    UINT XLMTM;           // 外部反方向限制信号 XLMTM 的电平状态 1: 高电平 0: 低电平
} USB1010_PARA_RR3,*PUSB1010_PARA_RR3;
```

5.17 状态寄存器 RR4 参数介绍 (USB1010_PARA_RR4)

```
typedef struct _USB1010_PARA_RR4
{
    UINT YSTOP0;          // 外部停止信号 YSTOP0 的电平状态 1: 高电平 0: 低电平
    UINT YSTOP1;          // 外部停止信号 YSTOP1 的电平状态 1: 高电平 0: 低电平
    UINT YEXPP;           // 外部正方向点动输入信号 YEXPP 的电平状态 1: 高电平 0: 低电平
    UINT YEXPM;           // 外部反方向点动输入信号 YEXPM 的电平状态 1: 高电平 0: 低电平
    UINT YINPOS;          // 外部伺服电机到位信号 YINPOS 的电平状态 1: 高电平 0: 低电平
    UINT YALARM;          // 外部伺服马达报警信号 YALARM 的电平状态 1: 高电平 0: 低电平
    UINT YIN0;            // 外部输入信号 YIN0 的电平状态 1: 高电平 0: 低电平
    UINT YIN1;            // 外部输入信号 YIN1 的电平状态 1: 高电平 0: 低电平
    UINT YLMTP;           // 外部正方向限制信号 YLMTP 的电平状态 1: 高电平 0: 低电平
    UINT YLMTM;           // 外部反方向限制信号 YLMTM 的电平状态 1: 高电平 0: 低电平
} USB1010_PARA_RR4,*PUSB1010_PARA_RR4;
```