

VCC 系列控制器 DLL 开发说明

拟制	日期	版本
	2019-2-18	T1

修改记录:

修改人员	日期	SDK 版本	修改内容
	2022. 5. 25	V1.0.0.6	1、 轴状态获取函数增加回零运动状态bit位 2、 增加读实时位置函数 3、 增加设置运动加速度函数
	2022. 11. 18	V1.0.011	1、 轴状态增加伺服使能状态、驱动器报警状态和光栅尺报警状态。
	2023. 05. 27	V1.0.0.15	1、 光源控制指令中增加激光笔控制字节内容说明 2、 增加控制器型号查询函数

目录

目录	3
概述	6
DLL 说明	6
使用步骤	6
DLL 方法概览	6
返回值说明	8
DLL 方法解析	10
1. int LH3500_initial(void)	10
2. int LH3500_close(void)	10
3. int LH3500_get_count_status(unsigned int *status)	10
4. int LH3500_get_motion_status(unsigned int *status)	11
5. int LH3500_get_axis_status(unsigned int *status)	12
6. int LH3500_get_profile(double *acc_time,double *max_speed)	13
7. int LH3500_get_position(double *x_scale,double *y_scale,double *z_scale, double *u_scale);	14
8. int LH3500_get_pulse_count(long *count)	14
9. int LH3500_set_light(unsigned int *light)	14
10. int LH3500_get_joystick_Msg(int *message)	14
11. int LH3500_go_home(int axis,double search_speed)	15
12. int LH3500_single_move_to(int axis,double speed,double target_pos)	15
13. int LH3500_XY_move_to(double speed,double *target_pos)	16
14. int LH3500_XYZ_move_to(double speed,double *target_pos)	16
15. int LH3500_arc_move_xy(double x_center, double y_center, double line_speed, double angle)	17
16. int LH3500_arc_move_yz(double y_center, double z_center, double line_speed, double angle)	17
17. int LH3500_arc_move_zx(double z_center, double x_center, double line_speed, double angle)	17

18.	int LH3500_jog_start(int axis,double jog_speed_max)	17
19.	int LH3500_change_speed(int axis,double curnt_speed)	18
20.	int LH3500_jog_stop(void).....	18
21.	int LH3500_disable_joystick(void).....	18
22.	int LH3500_enable_joystick(double *low_speed,double *high_speed)	18
23.	int LH3500_get_joy_comeback_position(double *scale_pos)	19
24.	int LH3500_stop(int axis,int mode)	19
25.	int LH3500_set_inposition_precision(double *precision)	19
26.	int LH3500_set_motion_probe(unsigned int probe_set)	19
27.	int LH3500_set_joystick_comeback(double comeback_speed,double comeback_distance).....	20
28.	int LH3500_touch_move(double touch_speed, double comeback_speed,double search_radius, double *target_pos).....	20
29.	int LH3500_get_probe_capture(double *x_scale,double *y_scale,double *z_scale)	21
30.	int LH3500_get_input(unsigned int *input)	21
31.	int LH3500_set_output(unsigned int output).....	21
32.	int LH3500_get_function_describe(unsigned int *function_describe)	22
33.	int LH3500_get_scale_resolution(double *resolution)	22
34.	int LH3500_set_position (unsigned short axis, double pos)	22
35.	int LH3500_get_soft_limit(int axis,double *pos_limit,double *neg_limit)	22
36.	int LH3500_get_inposition_precision(double *precision)	22
37.	int LH3500_get_firmware(char *firmware).....	23
38.	int LH3500_get_last_err(unsigned int *err_code).....	23
39.	int LH3500_super_command(unsigned int ch, int Slen, int Rlen, unsigned char *ucIn, unsigned char *ucOut)	23
40.	int LH3500_log_enable(bool enableVal)	23
41.	int LH3500_get_scale_picture(double *x_scale, double *y_scale, double *z_scale)	23
42.	int LH3500_clear_motion_err().....	23

43.	int LH3500_jog_mode(int val)	23
44.	int LH3500_flyingshot_set_position(unsigned short nums, unsigned short axes, double *pos)	24
45.	int LH3500_flyingshot_get_position(unsigned short *nums, unsigned short *axes, double *pos)	25
46.	int LH3500_flyingshot_clear_position()	25
47.	int LH3500_arc_move_xyz(double x_start, double y_start, double z_start, double x_mid, double y_mid, double z_mid, double x_end, double y_end, double z_end, double synVel, unsigned int circleType)	26
48.	int LH3500_flyingshot_set_basicpara(unsigned short outputValid, unsigned short outputReflection)	26
49.	int LH3500_flyingshot_set_pulse_width(unsigned short pulseWidth)	26
50.	int LH3500_flyingshot_set_range(float rangeVal)	26
51.	int LH3500_get_serialno(char* serialVal);	26
52.	int LH3500_compare_password(char* pwd, unsigned short len_pwd) 26	
53.	int LH3500_get_position_realtime(double* pos)	27
54.	int LH3500_set_acceleration(int axis, double value)	27
55.	int LH3500_get_controller_type(unsigned int* type)	27

概述

规定VCC3500控制器与控制端的通讯方式、LH3500 DLL的使用方法

DLL 说明

控制器通讯方式：100M以太网全双工通讯

开发系统配置：64 位操作系统

开发语言：C++

使用步骤

步骤 1：准备 LH3500.h、LH3500.lib、LH3500.dll；

步骤 2：在工程中引用头文件 LH3500.h 和 LH3500.lib，并把 LH3500.dll 放到程序运行目录下；

步骤 3：在程序中使用 LH3500_initial 与控制器建立连接；

步骤 4：开始其他操作；

DLL 方法概览

序号	名称	作用	备注
1	int LH3500_initial(void)	与控制器建立连接, 并初始化控制器参数	
2	int LH3500_close(void)	断开与控制器的数据连接	
3	int LH3500_get_count_status(unsigned int *status)	读取计数模块状态字	
4	int LH3500_get_motion_status(unsigned int *status)	读取运动模块状态字	
5	int LH3500_get_axis_status(unsigned int *status)	读各轴状态	
6	int LH3500_get_profile(double *acc_time, double *max_speed)	读取调试软件设定的各轴最大速度和加速度	
7	int LH3500_get_position(double *x_scale, double *y_scale, double *z_scale, double *u_scale)	读取光栅尺位置	
8	int LH3500_get_pulse_count(long *count)	读取各轴脉冲计数器	
9	int LH3500_set_light(unsigned int *light)	设置光源亮度	
10	int LH3500_get_joystick_Msg(int *message)	读取操纵杆信息	
11	int LH3500_go_home(int axis, double search_speed)	回原点运动	
12	int LH3500_single_move_to(int axis, double speed, double target_pos)	单轴点位运动	
13	int LH3500_XY_move_to(double speed, double *target_pos)	XY 直线插补到指定位置	
14	int LH3500_XYZ_move_to(double speed, double *target_pos)	XYZ 直线插补到指定位置	
15	int LH3500_arc_move_xy(double x_center, double y_center, double line_speed, double angle)	在 XOY 平面作圆弧插补运动	
16	int LH3500_arc_move_yz(double y_center, double z_center, double line_speed, double angle)	在 YOZ 平面作圆弧插补运动	
17	int LH3500_arc_move_zx(double z_center, double x_center, double line_speed, double angle)	在 ZOX 平面作圆弧插补运动	
18	int LH3500_jog_start(int axis, double	示教运动启动	

	jog_speed_max)		
19	int LH3500_change_speed(int axis,double curnt_speed)	示教运动变速	
20	int LH3500_jog_stop(void)	停止示教运动	
21	int LH3500_disable_joystick(void)	禁止操纵杆控制	
22	int LH3500_enable_joystick(double *low_speed,double *high_speed)	允许操纵杆控制,同时指定各轴高/低档速设置	
23	int LH3500_get_joy_comeback_position(double *scale_pos)	读取操纵杆方式探针采点反弹完毕瞬间光栅尺坐标位置	
24	int LH3500_stop(int axis,int mode)	停止指令	
25	int LH3500_set_inposition_precision(double *precision)	设置各轴定位精度	
26	int LH3500_set_motion_probe(unsigned int probe_set)	探针功能设置	
27	int LH3500_set_joystick_comeback(double comeback_speed,double comeback_distance)	设置摇杆采点回退参数	
28	int LH3500_touch_move(double touch_speed, double comeback_speed,double search_radius, double *target_pos)	探针采点运动指令	
29	int LH3500_get_probe_capture(double *x_scale,double *y_scale,double *z_scale)	读取探针锁存值	
30	int LH3500_get_input(unsigned int *input)	读通用输入口状态	
31	int LH3500_set_output(unsigned int output)	写通用输出口状态	
32	int LH3500_get_function_describe(unsigned int *function_describe)	读取 LH3500 功能配置符	
33	int LH3500_get_scale_resolution(double *resolution)	读取各轴光学尺分辨率	
34	int LH3500_set_position (unsigned short axis, double pos)	重置光栅尺位置	
35	int LH3500_get_soft_limit(int axis,double *pos_limit,double *neg_limit)	读取软件限位坐标(相对于机械零位)	
36	int LH3500_get_inposition_precision(double *precision)	读取定位精度设置	
37	int LH3500_get_firmware(char *firmware)	读取固件程序版本	
38	int LH3500_get_last_err(unsigned int *err_code)	读取最后一个出错信息代码	
39	int LH3500_super_command(unsigned int ch, int Slen, int Rlen, unsigned char *ucIn, unsigned char *ucOut)	超级指令	
40	int LH3500_log_enable(bool enableVal)	日志使能	
41	int LH3500_get_scale_picture(double *x_scale, double *y_scale, double *z_scale)	获取温度补偿后的坐标值	
42	int LH3500_clear_motion_err()	清除错误标记	
43	int LH3500_jog_mode(int val)	jog 模式设置	
44	int LH3500_flyingshot_set_position(unsigned short	飞拍--设定位置	

	nums, unsigned short axes, double *pos)		
45	int LH3500_flyingshot_get_position(unsigned short *nums, unsigned short *axes, double *pos)	飞拍--回读位置	
46	int LH3500_flyingshot_clear_position()	飞拍--清除位置	
47	int LH3500_arc_move_xyz(double x_start,double y_start, double z_start, double x_mid,double y_mid, double z_mid, double x_end,double y_end, double z_end, double synVel, unsigned int circleType)	空间三点圆弧	
48	int LH3500_flyingshot_set_basicpara(unsigned short outputValid,unsigned short outputReflection)	飞拍--配置飞拍	
49	int LH3500_flyingshot_set_pulse_width(unsigned short pulseWidth)	飞拍--设置输出脉宽	
50	int LH3500_flyingshot_set_range(float rangeVal)	飞拍--设置输出范围	
51	int LH3500_get_serialno(char* serialVal)	获取控制器序列号	
52	int LH3500_compare_password(char* pwd, unsigned short len_pwd)	比较加密信息	
53	LH3500_API int __stdcall LH3500_get_position_realtime(double* pos)	读取实时当前坐标	
54	LH3500_API int __stdcall LH3500_set_acceleration(int axis, double value)	设置轴加速度	

返回值说明

所有的 DLL 方法的返回值都是 int 类型，具体含义如下表

返回值说明表

返回值	说明	备注
0	操作成功	
1	Flash 错误: NAND_STATUS_PASSED	
2	Flash 错误: NAND_STATUS_FAILED	
8	Flash 错误: NAND_STATUS_NOT_FOUND	
16	Flash 错误: NAND_STATUS_DEVBUSY	
31	网络连接异常	
32	Flash 错误: NAND_STATUS_DEVWRPROTECT	
64	Flash 错误: NAND_STATUS_WAITTIMEOUT	
128	Flash 错误: NAND_STATUS_NOT_FOUND	
256	Flash 错误: NAND_STATUS_READ_ECC_ERROR_CORRECTED	
512	Flash 错误: NAND_STATUS_READ_ECC_UNCORRECTABLE_ERROR	
513	Flash 错误: 地址错误	
514	Flash 错误: 数据长度错误	
515	Flash 错误: 坏块错误	
516	Flash 错误: 校验错误	
4096	缓冲区队列为空	
4097	缓冲区队列已满	
4098	缓冲区队列忙碌	

4099	缓冲区队列空闲	
4100	缓冲区队列操作成功	
4101	缓冲区队列操作错误	
4102	网络通信数据长度错误	
4103	网络通信数据长度超限	
4104	网络通信超时	
4105	网络通信数据校验错误	
4106	网络通信通讯帧头错误	
4110	任务线程操作失败	
4115	该命令不支持, 请升级控制器固件版本	
4131	机器坐标系空闲状态	
4133	机器坐标系处于运动状态	
4135	机器坐标系任务已经执行完成	
4352	机器轴空闲状态	
4354	机器轴处于运动状态	
4356	该轴不存在	
4357	轴未使能	
4358	控制模式错误	
4359	正限位报警	
4360	负限位报警	
4361	伺服未使能	
4362	目标位置距离太小	
4368	位置跟随超差报警	
4372	驱动器报警	
4373	软限位报警	
4384	回零运动运行中	
4385	回零运动执行完成	
4386	回零运动负限位搜索错误	
4387	回零运动正限位搜索错误	
4388	回零运动原点信号异常	
4389	回零运动索引信号异常	
4390	回零运动轴未使能	
4391	机器未回零	
4392	回零运动偏移错误	
4393	回零运动捕获成功	
4394	回零运动捕获成功且运动结束	
4416	急停输入有效	
4417	编码器 A 相报警	
4418	编码器 B 相报警	
4419	编码器 C 相报警	
4420	限位输入已有效, 请先移动机器脱离限位	
4421	正限位未使能	
4422	负限位未使能	

4432	探针已触发	
4433	探针异常触发	
4435	测量运动搜索失败	
4436	自动探测回退结束后未到达目标位置	
4437	自动探测回退结束后探针未复位	
4448	不支持该参数, 请升级控制器固件版本	
4449	参数写入 FPGA 失败	
4464	DLL 参数错误, 不能为 0 值	
4465	DLL FIFO 错误	
4466	DLL FIFO 缓存已满错误	
4467	DLL FIFO 缓存已空错误	
4468	DLL FIFO 缓存可执行帧错误	
4480	正向软限位报警	
4481	负向软限位报警	
4512	圆弧半径不能为 0	
4513	圆弧旋转方向不匹配	
4514	圆弧两点距离大于直径	
4515	圆弧两点不在同一平面上	
4516	圆弧平面取值不匹配	
4517	圆弧两点到圆心距离不相等, 圆弧错误	
4518	圆弧弦高误差不匹配	
4519	空间圆弧取值不匹配, 为一条线	
4520	圆弧计算圆心角错误	

DLL 方法解析

1. int LH3500_initial(void)

函数功能:

该函数试图与控制器建立通讯连接, 如果通讯建立, 将初始化控制器。如果返回值非0, 请查返回值说明表获得返回值含义。

2. int LH3500_close(void)

函数功能:

断开与控制器的数据连接等操作。

3. int LH3500_get_count_status(unsigned int *status)

函数功能:

读取计数模块状态字; 该函数用来检测探针数据锁存标记、飞拍数据锁存标记等。

参数说明:

计数模块状态字表

位编号	位定义	取值说明	Word
Bit0	探针锁存标记	如果该位为 1, 表明对应锁存事件发生, 调用相应的捕获值读取函数可读取到光学尺位置锁存值, 读取后该位自动清 0; 探针、飞拍各自具有独立的一级深度的锁存	0x0001
Bit1	飞拍锁存标记		0x0002
Bit2	保留		0x0004

		器：以探针为例：假设探针触发，在未读取探针锁存值之前探针再次触发，则第一次触发时的位置数据被新的捕获位置覆盖。	
Bit3-bit7	保留	保留，读为 0	
Bit8	探针状态	1：探针未触发； 0—探针已触发	0x0100
Bit9	保留	保留，读为 0	
Bit10	保留	保留，读为 0	
Bit11	保留	保留，读为 0	
Bit12	飞拍过冲标记	1：飞拍 锁存数据未读取之前第 2 个移动拍照点到来； 0：未发生过冲	0x1000
Bit13-15	保留	保留，读为 0	

另请参考： LH3500_get_probe_capture

4. int LH3500_get_motion_status(unsigned int *status)

函数功能：

读取运动模块状态字。运动控制的重点在于运动模块状态字的理解,运动模块状态字含义丰富，是运动控制状态之窗，可以说，对运动控制理解和灵活运用的熟练程度与软件开发质量、控制精度以及控制器使用效果直接相关。如果测量软件有运行日志记录功能，在异常出现时该状态字应该被记录，作为硬件异常诊断的有力依据。如果不具备，应保留一个菜单在必要时弹出该状态字内容。

参数说明：

运动模块状态字表

位编号	位定义	取值说明	16 进制值
Bit0	所有轴停止标记	0-有轴尚处于运动中， 1-所有轴运动均已停止； 该位由 bit4-bit7“逻辑与”操作而来	0x0001
Bit1	出错标记	0-无出错信息或出错信息已读取， 1-有新的出错信息发生	0x0002
Bit2	保留	保留	
Bit3	操纵杆使能状态	1：操纵杆已使能， 0-操纵杆已被禁止	0x0008
Bit4	X 轴状态	1：停止状态， 0—运动状态	0x0010
Bit5	Y 轴状态	1：停止状态， 0—运动状态	0x0020
Bit6	Z 轴状态	1：停止状态， 0—运动状态	0x0040
Bit7	U 轴状态	1：停止状态， 0—运动状态	0x0080
Bit8	测量模式	1： 3D 测量模式， 0-2D 测量模式	0x0100
Bit9	探针使能	1：探针已使能， 0-探针未使能	0x0200
Bit10	紧急停止开关状态	1—紧急停止开关已触发， 0—未触发	0x0400
Bit11	自动探测异常标记	1：自动探测异常， 0：自动探测无异常	0x0800
Bit12	操纵杆采点异常标记	1：操纵杆采点异常， 0：操纵杆采点无异常	0x1000
Bit13	探针状态	1—安装就绪， 0—探针未安装或者探针已触发	0x2000
Bit14	探针异常触发标志	1—探针使能后，在非探针采点运动（点位运动、插补运动、 jog 运动）时触发，当探针复位后该标记位自动清 0 0—未出现异常触发现象	0x4000

Bit15	采点运动标记	0- 无探针采点运动或探针采点运动已结束 1- 探针采点运动正在进行中	0x8000
-------	--------	--	--------

Bit0: 用来判断直线插补运动是否完成，或者用来检测所有轴是否均已停止运动，它由bit4-bit7逻辑与操作而来。

Bit1: 出错标记，当API函数操作错误、系统错误（见pms 4000_get_last_err所获得出错代码信息）时，该标记位置1。调用pms 4000_get_last_err读取出错代码后该标记为清0。

BIT3: 操纵杆使能状态：成功使能后该标记为1，操纵杆禁止后该标记被清0。

BIT4: X轴停止状态标记位，当X轴停止运动（包括JOG运动change_speed到0）时该标记为1，运动时为0。

BIT5~7: Y/Z/U轴停止状态标记位，作用与X轴相同。

BIT9: 探针使能标记，探针被使能为1，探针被禁止为0。

BIT10: 紧急停止开关状态，紧急停止开关触发后为1，复位后为0。紧急停止开关触发后，将产生如下影响：

- A. 立即停止输出脉冲信号，停止所有轴运动。
- B. 结束探针测量自动采点线程。
- C. 终止JOG运动（相当于调用pms 4000_stop）。
- D. 终止单轴、多轴点位运动。
- E. 终止回原点运动线程。

BIT11: 自动探测异常标记，成功调用LH3500_touch_move函数后该标记清0，机台将朝向工件运动，在接触到工件后减速、并回退到调用LH3500_touch_move函数的位置，一个完整的探针测量运动成功结束，该标记保持为0；当异常发生时，采点运动结束后该标记为1，异常现象和原因请参考LH3500_get_touch_flag以及LH3500_get_last_err。异常处理：用操纵杆或JOG运动将探针移到安全的区域，排除问题后再进行测量。LH3500_clear_motion_err可清除该标记。

BIT12: 操纵杆采点异常标记，正常情况下，使用操纵杆进行探针采点编程时，探针碰撞后将自动回退，当异常发生时该标记被置1。异常处理：用操纵杆或JOG运动将探针移到安全的区域，排除问题后再进行测量。LH3500_clear_motion_err可清除该标记。

BIT13: 探针状态，当探针未安装、探针触发时该标记为0，探针已安装且就绪状态时该标记为1，实时的反映了探针当前的状态。

BIT14: 探针异常触发标志，在进行单轴、多轴点到点运动、JOG运动时，如果探针触发则被控制器判定为“探针异常触发”，该标志位置1，这些运动被立即停止，探针复位后该标志位自动清0。LH3500_clear_motion_err也可清除该标记。探针异常触发后如果探针使能，应禁止探针后下达运动指令使探针离开工件。

BIT15: 自动探测标志，成功调用LH3500_touch_move函数后该位置1，机台自动探测；采点成功并回退到调用LH3500_touch_move的位置后，该位置清0；异常出现时所有轴立即停止运动，新的touch_move指令成功下达时该标志清0。见BIT11说明

5. int LH3500_get_axis_status(unsigned int *status)

函数功能：

读取X/Y/Z/U轴的轴状态字

参数说明：

无论控制器是否具备U轴，status所指向的数组长度均不得小于4。

status[0]: X轴状态字

status[1]: Y轴状态字

status[2]: Z轴状态字

status[3]: U轴状态字

轴状态字表

位编号	位定义	取值说明	16 进制值
Bit0-4	保留	保留	
Bit5	正限位状态	0-未触发,1-已触发	0x0020
Bit6	负限位状态	0-未触发,1-已触发	0x0040
Bit7	原点开关状态	0-未触发,1-已触发	0x0080
Bit8	伺服状态	0-未上伺服, 1-上伺服	0x0100
Bit9	软限位状态	0-未触发, 1-已触发	0x0200
Bit10	运动状态	0-运动中, 1-已停止	0x0400
Bit11	回原点标记	0-未回原点或者回原点失败, 1-已成功回原点	0x0800
Bit12	回零运动状态	0-已停止, 1-回原点运动中	0x1000
Bit13	驱动器报警	0-无异常, 1-驱动器报警	0x2000
Bit14	编码器报警	0-无异常, 1-编码器报警	0x4000
Bit15	跟随误差超差	0-未出现跟随误差超差, 1-跟随误差超过设定值	0x8000

该寄存器对运动控制器的理解和使用比较重要，各 BIT 介绍如下：

Bit0-4、bit8-9、bit12-13：保留未用，读为 0。

Bit5 / Bit6 / Bit7：该位实时反映正限位 / 负限位 / 原点开关状态； 0 表示未触发，1 表示已触发。

Bit8：轴上伺服状态，0-未上伺服，1-上伺服。

Bit9：轴软限位状态，0-未触发软限位报警，1-已触发软限位报警。

Bit10：停止状态为 1，运动状态为 0。轴的运动控制命令下达后该位立即清 0，如果运动距离非常短（为 1 个运动当量或者为 0），运动状态也会保持为 0 若干微秒，尽管应用软件未必能捕捉到。因此应用软件发送移动距离不为 0 的运动命令后，最短时间内查询运动状态，即使移动还没有发生，但读取的运动状态也一定为“运动”状态。

Bit11：0—未回原点或者回原点失败；1—已成功回原点。控制器上电后为 0；LH3500_go_home 函数调用后首先清该位为 0，然后执行找原点运动。运动结束后应测试该位以确定找原点是否成功。

LH3500_initial 和 LH3500_close 函数对该位无影响，因此测量软件启动、关闭对该标记位无影响。若软件启动 LH3500_initial 成功后，调用该函数检测到 bit11 为 1，表明机器已回原点，可以立即投入使用，缩短软件从启动到投入正式使用的时间，提高仪器工作效率。

Bit12：0—回零运动已停止；1—回零运动中。

Bit13：0-驱动器工作正常；1-驱动器报警。

Bit14：控制器上电后为 0，若当前轴启动回原点，机台在正负限位开关之间往返一次仍然未找到 RI，则该标记被置位，同时 Last_err 将记录错误代码。

Bit15：控制器上电后为 0，当前轴点位运动时跟随误差超过设定值，当前轴将立即停止，同时该位置 1，在执行新的任务前需要调用 LH3500_clear_motion_err()清除错误。

6. int LH3500_get_profile(double *acc_time, double *max_speed)

函数功能：

读取各轴最大速度和最小加减速时间。加速度等于最大速度除以最小加减速时间。各轴最大速度一般情况下不建议测量软件来设定，因为测量软件无法得知该机台的最大运动速度、最大加速度；力合精密提供调试软件“SetupTool.exe”，该软件可设定机台稳定工作所能达到的各项最大参数，这包括最大速度、最大加速度（加速度=最大速度 / 加速时间）、脉冲

分辨率、安全行程等参数。

参数说明:

无论控制器是否具备U轴控制功能， `max_speed`所指向的数组长度不得小于4。

`acc[0]`: X轴最小加减速时间，单位S

`acc[1]`: Y轴最小加减速时间，单位S

`acc[2]`: Z轴最小加减速时间，单位S

`acc[3]`: U轴最小加减速时间，单位S

`max_speed [0]`: X轴最大速度，单位mm/S

`max_speed [1]`: Y轴最大速度，单位mm/S

`max_speed [2]`: Z轴最大速度，单位mm/S

`max_speed [3]`: U轴最大速度，单位X/S

7. `int LH3500_get_position(double *x_scale, double *y_scale, double *z_scale, double *u_scale);`

函数功能:

读取X / Y / Z/U轴光学尺位置（名义值）；控制器电源开启后该位置无意义，成功回原点
后该位置为相对原点（光学尺RI）位置。

参数说明:

单位为毫米。

8. `int LH3500_get_pulse_count(long *count)`

函数功能:

获取当前各轴脉冲计数

参数说明:

`count [0]`: X轴脉冲计数

`count [1]`: Y轴脉冲计数

`count [2]`: Z轴脉冲计数

`count [3]`: U轴脉冲计数

9. `int LH3500_set_light(unsigned int *light)`

函数功能:

设置表面光、底光、同轴光亮度；

参数说明:

`Light[0]`——`Light[39]`为表面光源1-40相亮度，表面光源不足40相的，未使用的应设为0。

`Light[40]`:轮廓光； `Light[41]`:同轴光； **`Light[42]`: 激光笔（串口控制通道），写1点亮激光笔，写0关闭激光笔， `Light[43]`保留。**

无论控制器具备多少分区表面光控制能力， `Light`数组长度不得小于44。

光源亮度分200级可调，参数取值范围为0—200，超过200将导致参数出错返回。

10. `int LH3500_get_joystick_Msg(int *message)`

函数功能:

读取操纵盒按钮状态信息。无论控制器是否处于操纵杆运动模式，该函数均有效。

参数说明:

`message`指向的数组长度为5，小于5将导致指针访问出界。

`Message[0]`—`[2]`: 未定义

Message[3]: 键盘状态码

Message[4]: 未定义

操纵杆状态表

位编号	按键定义	操作说明	对应 16 进制数
0	未定义		
1	KEY_GOTO	拐点设置	0x0002
2	KEY_F1	F1 功能键	0x0004
3	未定义		
4	未定义		
5	未定义		
6	KEY_DELETE_POINT	删除探针采点	0x0040
7	未定义		
8	未定义		
9	KEY_PROBE_OVERRIDE	探针忽略/使能	0x0200
10	未定义		
11	未定义		
12	KEY_F2	F2 功能键③	0x1000
13	KEY_F3	F3 功能键③	0x2000

11. int LH3500_go_home(int axis, double search_speed)

函数功能:

令指定轴回原点，对于x / y / z轴，回原点即为找光学尺索引信号（又称零窗）；该轴状态bit11（回原点标记）清0；然后移动到预设限位开关，最后开始找索引信号，找到索引信号瞬间将位置设为0且轴状态bit11被置1，减速停止。

Go_home成功执行的条件:

1. 当前轴处于停止状态
2. 如果探针已使能，探针应该处于未触发状态

对于U轴，回原点实际上是找负限位开关，找到负限位开关后低速离开限位，在限位信号消失瞬间立即停止，并且脉冲计数器清零。

参数说明:

axis: 轴编号， 0-X轴， 1-Y轴， 2-Z轴,3-U轴;

search_speed: 由软件根据机器行程和状态设定回零速度，默认20~30mm/s。

12. int LH3500_single_move_to(int axis, double speed, double target_pos)

函数功能:

命令指定轴以绝对坐标方式定位到光学尺位置为target_pos。

对于U轴控制变倍镜头，设置目标位置为需要设定的放大倍数的10倍，控制器按照调试的参数运动到指定的目标位置，完成自动调整放大倍数的功能。

请注意： U轴传动机构多为采用齿轮方式的传动机构，若U轴位置控制系统为步进马达，或者伺服马达，但位置反馈元件与电机同轴相连，则传动系统必存在间隙，在运动方向需反向时需重新回零，以消除间隙。

如果探针已使能，但调用该函数时探针处于触发状态，该操作将被忽略并返回“操作错误”。对于XYZ轴而言，目标位置为名义值，等于光学尺计数脉冲个数乘以分辨率，如果您的机

台进行了线性补偿，需将终点位置换算为名义值后传递给该函数。另请参考 LH3500_get_scale。

Single_move_to成功执行的条件：

1. 当前轴已处于停止状态
2. 如果探针已使能，探针应该处于未触发状态
3. 如果操纵杆已使能，操纵杆应处于未扳动状态

参数说明：

axis: 轴编号， 0-X轴， 1-Y轴， 2-Z轴,3-U轴；

speed: 对于X / Y / Z轴，单位为mm / s， 当该速度大于控制器安装软件整定的最大速度时，控制器采用安装软件设定的最大速度；对于U轴，单位为：10*脉冲/s。

target_pos[0]/target_pos[1]/target_pos[2]/ target_pos[3]: 运动到终点时光学尺坐标位置， 绝对坐标方式， 单位为： mm，或设定的坐标单位。

13. int LH3500_XY_move_to(double speed, double *target_pos)

函数功能：

令 XY 轴按照直线插补运动规律运动到某处。

直线插补运动状态查询请参考 LH3500_get_motion_status。

XY_move_to 成功执行的条件：

1. 当前轴已处于停止状态
2. 如果探针已使能，探针应该处于未触发状态
3. 如果操纵杆已使能，操纵杆应处于未扳动状态

参数说明：

speed:直线插补运动矢量速度。当各轴分速度大于控制器所保存的最大速度时，控制器自动压缩矢量速度曲线； 该特性使得机器能在保证可靠运行的前提下使仪器工作效率最高。

target_pos[0]:X 轴终点绝对位置

target_pos[1]:Y 轴终点绝对位置

14. int LH3500_XYZ_move_to(double speed, double *target_pos)

函数功能：

令 XYZ 轴按照直线插补运动规律运动到某处。

直线插补运动状态查询请参考 LH3500_get_motion_status，运动模块状态字

BIT0 最适合用来判断直线插补运动是否完毕， BIT14 作为辅助判断标记。

XYZ_move_to 成功执行的条件：

1. 当前轴已处于停止状态
2. 如果探针已使能，探针应该处于未触发状态
3. 如果操纵杆已使能，操纵杆应处于未扳动状态

参数说明：

speed:直线插补运动矢量速度。当计算出来某一轴的分速度超过该轴的最大速度值时，各轴的运动速度将等比例降低，确保各轴安全运动，且运动轨迹的精度保持不丢失。

举例说明： Z 轴运动速度一般远低于 X/Y 轴， X/Y 轴运动距离比 Z 轴大时，仪器以比大的矢量速度运动，确保效率。当 Z 轴运动距离比较大时，Z 轴的理论分速度超过了 Z 轴最大速度，各轴运动速度将自动降低，确保 Z 轴能可靠运行。 VCC3500 的多轴直线插补采用确保效率和运行可靠的“聪明”控制策略。

target_pos[0]:X 轴终点绝对位置

target_pos[1]:Y 轴终点绝对位置

target_pos[2]:Z 轴终点绝对位置

15. int LH3500_arc_move_xy(double x_center, double y_center, double line_speed, double angle)

函数功能:

XOY平面圆弧插补运动。相当于G代码G17

参数说明:

cent_pos1: 对应圆弧中心点坐标X轴相对起点坐标;

cent_pos2: 对应圆弧中心点坐标Y轴相对起点坐标;

line_speed: 圆弧运动线速度; 单位: mm/s

rotary_angle: 旋转角度和方向, 单位: 度; 取值范围: -360--360; 大于0时, 逆时针旋转, 值小于0时, 顺时针旋转。

16. int LH3500_arc_move_yz(double y_center, double z_center, double line_speed, double angle)

函数功能:

YOZ 平面圆弧插补运动。相当于 G 代码 G19

参数说明:

y_center: 对应圆弧中心点坐标 Y 轴相对起点坐标;

z_center: 对应圆弧中心点坐标 Z 轴相对起点坐标;

line_speed: 圆弧运动线速度; 单位: mm/s

rotary_angle: 旋转角度和方向, 单位: 度; 取值范围: -360--360; 大于 0 时, 逆时针旋转, 值小于 0 时, 顺时针旋转。

17. int LH3500_arc_move_zx(double z_center, double x_center, double line_speed, double angle)

函数功能:

ZOX 平面圆弧插补运动。相当于 G 代码中的 G18

参数说明:

z_center: 对应圆弧中心点坐标 Z 轴相对起点坐标;

x_center: 对应圆弧中心点坐标 X 轴相对起点坐标;

line_speed: 圆弧运动线速度; 单位: mm/s

rotary_angle: 旋转角度和方向, 单位: 度; 取值范围: -360--360; 大于 0 时, 逆时针旋转, 值小于 0 时, 顺时针旋转。

18. int LH3500_jog_start(int axis, double jog_speed_max)

函数功能:

使指定轴开始 JOG 模式运动, 该函数指定了本次启动的 jog 运动所允许的最大速度。该函数调用之后, LH3500_jog_stop 调用之前, 可连续调用 LH3500_change_speed 改变运动速度和方向。jog_speed_max 通常等于 LH3500_get_profile 读取到的最大运行速度。

Jog 功能特别适用于使用鼠标控制机台运动的情形。

假如首次启动 JOG 运动, 或者先前启动的 JOG 运动已被 LH3500_stop 停止, 需要再次启动 JOG 运动时, 必须具备如下条件:

1. 操纵杆已被 disable 或者操纵杆已使能但没有被扳动;

2. 没有一个轴处于单轴、多轴点到点运动状态；
 3. 如果探针已被使能，探针应该处于未触发状态；
- 如果某轴已经处于 jog 运动，其它轴当然具备 jog 运动条件。
- 如果探针已使能，且调用该函数时探针处于触发状态，该操作将被忽略并返回“操作出错”。

参数说明：

axis: 轴编号， 0-X 轴， 1-Y 轴， 2-Z 轴,3-U 轴；

jog_speed_max: 期望的最大 JOG 运动速度绝对值。

19. int LH3500_change_speed(int axis, double curnt_speed)

函数功能：

jog_start 函数调用之后、 LH3500_stop 调用之前，连续不断调用该函数以达到改变 jog 运动速度和方向的目的。

该函数执行的条件：指定轴已使用 jog_start 设置好最大速度并启动 jog 运动。

参数说明：

axis: 轴编号， 0-X 轴， 1-Y 轴， 2-Z 轴,3-U 轴；

curnt_speed: 目标速度。该参数为正时，机台朝正向运动，为负时朝负向运动，该参数数值和方向突然变化时，控制器将会使速度按照梯形曲线加减速变化；该参数绝对值大于 jog_speed_max 时，速度限制为 jog_speed_max。调用后轴移动速度将一直保持，直到限位被触发。如果 LH3500_change_speed(n,0)，轴将处于“速度为 0 的 jog 状态”，除非使用 LH3500_jog_stop 命令。

20. int LH3500_jog_stop(void)

函数功能：

令所有处于 JOG 状态的轴减速停止。该函数任何时候调用、重复调用都不会报错。该函数一定要与 jog_start 配套使用，比如：在鼠标左键按下时调用 LH3500_jog_start，在鼠标左键松开时调用 LH3500_jog_stop。Change_speed 到 0 可以使轴处于停止运转状态，但轴仍然处于“速度为 0 的 JOG”运动状态，使用 jog_stop 之前，发送运动命令将会拒绝执行并以出错返回。

21. int LH3500_disable_joystick(void)

函数功能：

通知控制器，无论任何轴处于任何状态，扳动操纵杆都不要让机台产生运动。该操作常用于：

1. 用户编制的自动测量程序启动前。
2. 自动对焦或其它不希望操作者通过操纵杆控制机台运动的情形。

无论操纵杆是否使能，LH3500_disable_joystick 多次调用该函数无影响。

22. int LH3500_enable_joystick(double *low_speed, double *high_speed)

函数功能：

通知控制器，允许操纵杆控制机台运动。控制器连接 IMO 手操器时，可通过面板上的 Unlock 键解锁操纵盒，从而使能操纵杆。该函数执行成功的条件：

1. 操纵盒已校正并且已连接到控制器。
2. 调用该函数时操纵盒处于未被扳动状态。

允许操纵盒控制后，操纵盒上高低速切换按钮才会被响应。其他按钮不受操纵杆是否使能

的限制。

操纵杆使能后，如果所有轴空闲，扳动操纵杆，机台移动且速度由操纵杆控制。所有轴停止后，可以执行点位运动、JOG 运动、探针采点运动；如果这些运动正在进行，操纵杆将暂时屏蔽，直到所有轴停下来。为了避免测量程序自动运行时，操纵杆产生干扰（比如运动的停顿瞬间），自动运行启动前应该用 LH3500_disable_joystick 禁止操纵杆，在需要人工干预的地方或者测量程序运行完毕再使能操纵杆。扳动操纵杆移动机台，当探针触发时机台将自动减速、停止，并按照设定的回退速度和回退距离回退。另请参考:LH3500_disable_joystick、LH3500_touch_move。

参数说明:

low_speed 和 high_speed 参数保留；该速度由用户在“调试软件”界面设置。

23. int LH3500_get_joy_comeback_position(double *scale_pos)

函数功能:

读取操纵杆方式探针采点，反弹完毕瞬间光栅尺坐标位置。该位置结合探针与工件发生接触时的坐标位置，可用于判断采点法矢量、生成 IJK。应用软件准备接受操纵杆示教采点时——比如测量软件上鼠标点击了圆柱测量图标，使用 LH3500_get_motion_status 查询 motion_status 的 bit2。无论是否有记录，建议使用该函数一次，以清除不必要的记录。随后周期性（建议 100ms）查询新记录是否发生。

参数说明:

scale_pos: 对应数组长度为 4，scale_pos[0]: X 轴坐标；scale_pos[1]: Y 轴坐标；scale_pos[2]: Z 轴坐标

24. int LH3500_stop(int axis, int mode)

函数功能:

令指定轴停止运动，通常用来中止一个已经下达的任何运动指令，但如果是 JOG 运动，还需要使用 LH3500_jog_stop 命令。

参数说明:

axis: 轴编号，0-X 轴，1-Y 轴，2-Z 轴,3-U 轴；

mode: 停止模式:

0—减速停止，如果正在高速运动，该函数调用后机台将滑行一段之后停止；

1—立即停止，该函数返回后该轴便处于停止状态，由于立即停止会导致产生较大冲击，可能导致机台损坏，如果机台比较大运行速度比较快，非紧急情况不要使用立即停止。

25. int LH3500_set_inposition_precision(double *precision)

函数功能:

设置各轴定位精度，机器运动位置误差小于定位精度时，定位完成标志置 1。

参数说明:

precision[0]\precision[1]\precision[2]: 分别是 XYZ 轴定位完成窗口大小，数值不小于光学尺分辨率 2 倍。precision[3]必需保留；单位 mm

26. int LH3500_set_motion_probe(unsigned int probe_set)

函数功能:

使能、忽略探针。探针使能与屏蔽在控制器中可以设置，并存储在控制器 flash 内。探针使能后，单轴点到点运动、多轴直线插补和圆弧插补、回原点运动以及 JOG 运动时触发探

针控制器将立即停止。因此，在使用探针交换架或类似操作时，需暂时忽略探针，直到测头交换完毕再次使能探针。

参数说明:

probe_set: 0-禁止探针, 1-使能探针

27. int LH3500_set_joystick_comeback(double comeback_speed, double comeback_distance)

函数功能:

设置操纵杆采点反弹参数。该参数已通过控制器调试软件设置并保存于控制器之内。测量软件可覆盖此设置,但测量软件的设置不会被保存,电源重新启动将恢复调试软件设定值。

参数说明:

comeback_speed: 探针采点后反弹速度, 单位 mm/s

comeback_distance: 反弹距离, 单位 mm。如果操纵杆从距离碰撞点较远距离 (大于 comeback_distance), 采点后反弹距离为该参数设定值, 如果距离不大于该参数, 反弹运动到采点起始点。该特性对于直径较小的孔洞测量非常方便, 避免了呆板的按照 comeback_distance 反弹导致的“碰壁”现象。

28. int LH3500_touch_move(double touch_speed, double comeback_speed, double search_radius, double *target_pos)

函数功能:

启动自动探测采点运动。探针采点运动用于测量软件执行已编好的测量程序, 获取一个碰撞点坐标。使用 LH3500_XYZ_move_to 定位到碰撞点附近, 然后调用该函数启动采点运动。

成功执行的条件:

1. 探针已经使能并且已就绪
2. 所有轴处于停止状态。
3. 操纵杆已禁止。

该函数成功调用后发生如下事情:

1. 运动模块状态字 Bit11 (自动探测失败标记) 设置为 0
2. 运动模块状态字 bit15 设置为 1
3. 控制器使机器从调用该函数的地方开始, 以 touch_speed 朝“理论碰撞点”运动, 在此过程中接触到工件后, 探针将减速停止然后回退到调用该函数的位置, 采点运动成功结束。如果到达理论碰撞点后探针没有接触到工件, 将继续向前运动 (搜索工件), 如果超出“理论碰撞点”的距离大于“搜索半径”, 机台将停止运动, 采点运动结束; 运动模块状态字 Bit11 将被置 1, 同时 Last_err 代码被设置。

无论成功结束还是异常结束, 运动模块状态字 bit15 都会清零。

参数说明:

touch_speed: 采点速度, 单位 mm/s, 通常为 0.5-20mm/s

comeback_speed: 回退速度, 单位 mm/s; 通常为 5-200mm/s

search_radius: 搜索半径, 由于待测工件尺寸偏差和夹具精度原因, 自动测量时理论碰撞点总与实际碰撞点存在偏差, 因此探针需要在理论碰撞点之后一定范围内 (搜索半径) 搜索工件。

target_pos[0]: X 轴理论碰撞点坐标, 单位 mm

target_pos[1]: Y 轴理论碰撞点坐标, 单位 mm

target_pos[2]: Z 轴理论碰撞点坐标, 单位 mm

29. int LH3500_get_probe_capture(double *x_scale, double *y_scale, double *z_scale)

函数功能:

读取探针锁存的 X / Y / Z 坐标。当 get_count_status 检测到计数模块状态字 bit0 (探针锁存标记) 为 1 时, 说明探针有更新新的锁存数据, 探针忽略状态时被触发, 不会锁存数据。

参数说明:

x_scale: X 轴光学尺位置 (名义值), 单位为 mm。
y_scale: Y 轴光学尺位置, 同 X 轴
z_scale: Z 轴光学尺位置, 同 X 轴

30. int LH3500_get_input(unsigned int *input)

函数功能:

读取输入口电平状态。VCC3200 和 VCC4500 控制器硬件设计最多 11 个通用输入 IO (IN0~10), VCC3500 控制器最多可以使用 5 路输入信号 (IN0~4)。

参数说明:

输入口状态位定义表

位编号	位定义	取值说明	16 进制值
Bit0	通用输入 0	0—低电平, 1—高电平	0x0001
Bit 1	通用输入 1	0—低电平, 1—高电平	0x0002
Bit 2	通用输入 2	0—低电平, 1—高电平	0x0004
Bit 3	通用输入 3	0—低电平, 1—高电平	0x0008
Bit 4	通用输入 4	0—低电平, 1—高电平	0x0010
Bit 5	通用输入 5	0—低电平, 1—高电平	0x0020
Bit 6	通用输入 6	0—低电平, 1—高电平	0x0040
Bit 7	通用输入 7	0—低电平, 1—高电平	0x0080
Bit 8	通用输入 8	0—低电平, 1—高电平	0x0100
Bit 9	通用输入 8	0—低电平, 1—高电平	0x0200
Bit 10	通用输入 9	0—低电平, 1—高电平	0x0400
Bit 11- Bit 31	保留		

31. int LH3500_set_output(unsigned int output)

函数功能:

设置控制器 IO 输出口状态。输出口可用于声光提示设备等, 还可以用来实现在线测量时输出握手信号给其他设备。输出口电路为三极管集电极开路输出, 控制器电源开启后三极管处于截止状态。VCC3200 和 VCC4500 控制器硬件设计最多可以控制 11 个通用输出口 (OUT0~10)。VCC3500 最多可以控制 6 路输出 (OUT0~5)。

参数说明:

output 状态位定义表

位编号	位定义	取值说明	16 进制值
Bit 0	Out 0	0—输出导通, 负载加电, 1—输出截止, 负载断电	0x0001
Bit 1	Out 1	0—输出导通, 负载加电, 1—输出截止, 负载断电	0x0002
Bit 2	Out 2	0—输出导通, 负载加电, 1—输出截止, 负载断电	0x0004
Bit 3	Out 3	0—输出导通, 负载加电, 1—输出截止, 负载断电	0x0008
Bit 4	Out 4	0—输出导通, 负载加电, 1—输出截止, 负载断电	0x0010

Bit 5	Out 5	0—输出导通，负载加电， 1—输出截止，负载断电	0x0020
Bit 6	Out 6	0—输出导通，负载加电， 1—输出截止，负载断电	0x0040
Bit 7	Out 7	0—输出导通，负载加电， 1—输出截止，负载断电	0x0080
Bit 8	Out 8	0—输出导通，负载加电， 1—输出截止，负载断电	0x0100
Bit 9	Out 9	0—输出导通，负载加电， 1—输出截止，负载断电	0x0200
Bit 10	Out 10	0—输出导通，负载加电， 1—输出截止，负载断电	0x0400
Bit 11- Bit 31	Reserved	0—输出导通，负载加电， 1—输出截止，负载断电	

32. int LH3500_get_function_describe(unsigned int *function_describe)

函数功能：获取功能说明

33. int LH3500_get_scale_resolution(double *resolution)

函数功能：

获取各轴光学尺分辨率

参数说明：

resolution [0]: X 轴光学尺分辨率

resolution [1]: Y 轴光学尺分辨率

resolution [2]: Z 轴光学尺分辨率

resolution [3]: Z 轴光学尺分辨率

34. int LH3500_set_position (unsigned short axis, double pos)

函数功能：

重置各轴光学尺位置值；回原点标记由于该函数会改变机械坐标，原来基于原点的工件坐标系将失效，已做好的测量程序可能失效，因此需慎用。该函数调用后，回零位标记（axis_status bit11）将被清 0，软件限位和软件减速功能失效，机台最大移动速度将被限制为设定最大速度的四分之一。

参数说明：

axis: 轴编号，0-X 轴，1-Y 轴，2-Z 轴,3-U 轴；

pos: 轴光学尺计数设置，单位为 mm

35. int LH3500_get_soft_limit(int axis,double *pos_limit,double *neg_limit)

函数功能：

获取指定轴正、负限位开关相对于原点的位置，即获取轴安全运动范围。

参数说明：

axis: 轴编号，0-X 轴，1-Y 轴，2-Z 轴,3-U 轴；

pos_limit/neg_limit: 正、负软件限位坐标位置，单位 mm

36. int LH3500_get_inposition_precision(double *precision)

函数功能：

获取各轴定位精度，单位 mm。

参数说明：

precision [0]: X轴定位精度

precision [1]: Y轴定位精度

precision [2]: Z轴定位精度

precision [3]: U轴定位精度

37. int LH3500_get_firmware(char *firmware)

函数功能:

获取固件版本号

参数说明:

返回固件的版本号

38. int LH3500_get_last_err(unsigned int *err_code)

函数功能:

当 LH3500_get_motion_status(unsigned int *status)读取到运动模块状态字 bit1 为 1（对应 16 进制数为 0x0002）时，说明有出错现象产生且未被读取。该函数用于读取最后一次的出错信息代码。该函数调用后模块状态字 bit1 自动清 0。

参数说明:

err_code: 错误信息代码，具体含义请对照“返回值说明表”。

39. int LH3500_super_command(unsigned int ch, int Slen, int Rlen, unsigned char *ucIn, unsigned char *ucOut)

函数功能: 获取超级用户权限

40. int LH3500_log_enable(bool enableVal)

函数功能:

使能日志记录

参数说明:

enableVal 为 true--记录操作日志，false--不记录操作日志

41. int LH3500_get_scale_picture(double *x_scale, double *y_scale, double *z_scale)

函数功能:

读取温度补偿后 X / Y / Z 轴光学尺位置;

参数说明:

单位为 mm。

42. int LH3500_clear_motion_err()

函数功能:

清除控制器当前出现的错误;

参数说明:

没有参数。

43. int LH3500_jog_mode(int val)

函数功能:

设置控制器 jog 运动的模式;

参数说明:

val 为 0 时, jog 模式为不连续调速模式; val 为 1 时, jog 模式为连续调速模式。默认值为 0。

44. int LH3500_flyingshot_set_position(unsigned short nums, unsigned short axes, double *pos)

函数功能:

设置飞拍触发的目标位置, 最多可以同时设置 1000 组位置点。

参数说明:

nums 表示点位个数;

axes 表示需要采集的轴的掩码;

pos 表示采集点位坐标的数组, 数组长度为 nums * 3, 并且(x,y,z)为一组顺序排列;

备注: 如采集输出点位 2 个, 三轴全采集, 则 nums 为 2, axes 为 7(111), pos 的数据为[x0,y0,z0,x1,y1,z1];

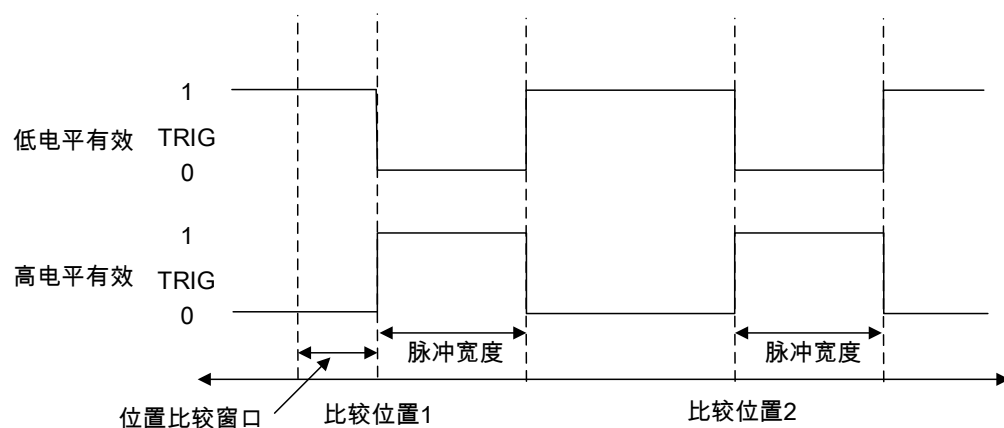
采集输出点位 2 个, 采集 x 和 y 数据, 则 nums 为 2, axes 为 3(110), pos 的数据为[x0,y0,0,x1,y1,0]

飞拍功能说明:

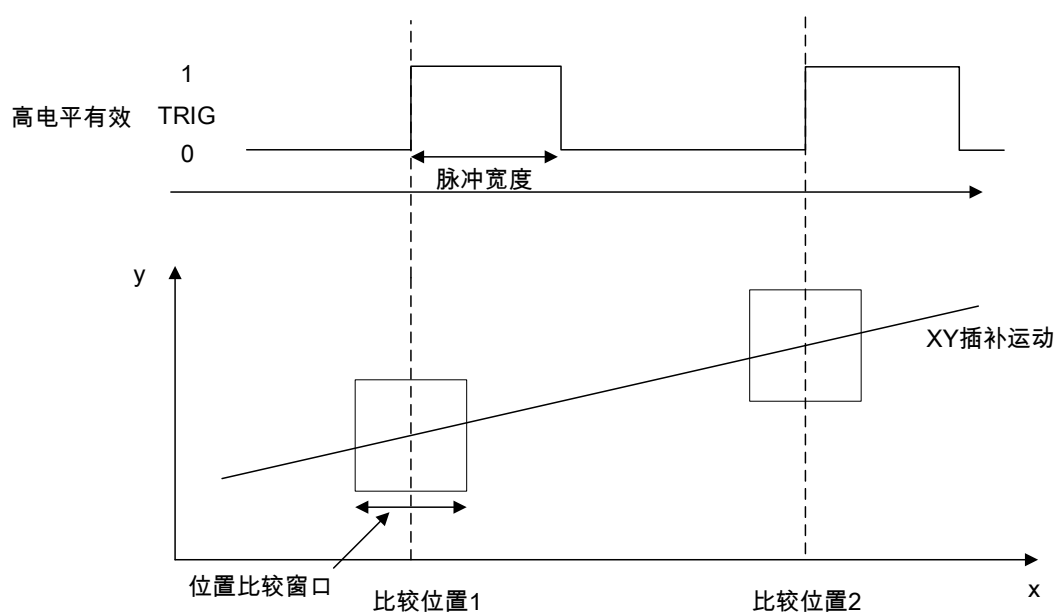
飞拍是 VCC3500 的高级功能(VCC3200 不支持)。它是在硬件位置比较单元中执行的。VCC3500 的硬件位置比较功能基于当前编码器位置生成数字输出 IO 触发信号。当编码器位置与预置的“比较”寄存器匹配时, “TRIG”信号输出电平跳变, 并持续设定的触发电平保持时间。这个功能完全在硬件中执行, 不需要软件干预(当然它是在软件中可设置的)。比较功能中唯一的延迟是硬件门电路延迟(在任何机械系统中可以忽略不计), 所以这提供了一个非常精确的比较功能。

位置比较功能是在 ASIC 的软件配置硬件寄存器中实现。软件配置赋予功能灵活性; 硬件电路使功能的速度和准确性。

这个功能对视觉飞拍测量很有用。该测量可以设置几个位置, 将并预先存入控制器缓存中。然后开始单点对点移动, 或 XY、XYZ 直线插补移动, 使机床通过预设位置移动。当机器进入控制器预置的比较窗口时, 比较单元会切换输出 IO, 触发相机拍照。位置比较三角函数如下图所示:



a. 单轴比较模式



b. XY 比较模式

45. `int LH3500_flyingshot_get_position(unsigned short *nums, unsigned short *axes, double *pos)`

函数功能:

获取运动过程中飞拍触发时实际捕获的位置数据。

参数说明:

`nums` 表示当前采集到的点位个数;

`axes` 表示采集轴的掩码;

`pos` 表示采集点位坐标的数组, 传入的数组长度要大于或等于 `pms4000_flyingshot_set_position` 设置的输出点位个数*3,;

备注: 如采集到的点位 2 个, 三轴全采集, 则 `nums` 为 2, `axes` 为 7(111), `pos` 的数据为 `[x0,y0,z0,x1,y1,z1]`;

采集到的点位 2 个, 采集 x 和 y 数据, 则 `nums` 为 2, `axes` 为 3(110), `pos` 的数据为 `[x0,y0,0,x1,y1,0]`

46. `int LH3500_flyingshot_clear_position()`

函数功能:

清除设置的飞拍触发点位数据

参数说明:

无参数;

47. int LH3500_arc_move_xyz(double x_start, double y_start, double z_start, double x_mid, double y_mid, double z_mid, double x_end, double y_end, double z_end, double synVel, unsigned int circleType)

函数功能:

空间三点圆弧运动

参数说明:

x_start、y_start、z_start 为圆弧起点坐标,
x_mid、y_mid、z_mid 为圆弧中间任意点坐标,
x_end、y_end、z_end 为圆弧终点坐标;
synVel 为圆弧运动线速度
circleType 表示曲线类型, 0 为圆弧, 1 为整圆

48. int LH3500_flyingshot_set_basicpara(unsigned short outputValid, unsigned short outputReflection)

函数功能:

设置飞拍比较输出的基础参数。

参数说明:

outputValid 表示比较输出的有效电平, 0 为低电平有效, 1 为高电平有效;
outputReflection 表示比较输出映射的模式, (0-无映射, 1-X 轴, 2-Y 轴, 3-Z 轴, 4-XY 轴, 5-XYZ 轴)

49. int LH3500_flyingshot_set_pulse_width(unsigned short pulseWidth)

函数功能:

设置飞拍输出脉宽。

参数说明:

pulseWidth 表示比较输出的脉宽, 单位 0.1 毫秒, 范围 0-6500;

50. int LH3500_flyingshot_set_range(float rangeVal)

函数功能:

设置飞拍位置比较窗口, 机器运动位置在设定的拍照位置附近的比较窗口范围内时, 控制器开始计算, 在最近的位置输出触发信号, 同时锁存拍照时的实际位置。

参数说明:

rangeVal 表示位置比较的窗口范围, 单位: mm。

51. int LH3500_get_serialno(char* serialVal);

函数功能:

获取控制器序列号。

参数说明:

serialVal 表示用于存储返回的序列号信息的数组, 长度为 32;

52. int LH3500_compare_password(char* pwd, unsigned short len_pwd)

函数功能:

比较加密信息，成功则允许运动操作，否则禁止运动。

参数说明:

pwd 表示密码数组;

len_pwd 表示密码数组长度 (长度小于 1000 字节);

53.int LH3500_get_position_realtime(double* pos)

函数功能:

获取实时位置数据。

参数说明:

Pos: 当前实时机器坐标数据,数组指针, 最长 4 个 double 数据,

pos[0]:X 轴机器当前坐标, 单位 mm;

pos[1]:Y 轴机器当前坐标, 单位 mm;

pos[2]:Z 轴机器当前坐标, 单位 mm;

pos[3]:U 轴机器当前坐标, 单位 mm;

54.int LH3500_set_acceleration(int axis, double value)

函数功能:

设置机器运动加速度(单位:mm/s²), 设置值立即生效, 但不永久保存在控制器中。

参数说明:

axis: 要设置加速度的轴号: 0~3;

value: 设置的加速度值 (单位: mm/s²);

55.int LH3500_get_controller_type(unsigned int* type)

函数功能:

获取控制器型号。

参数说明:

type: 控制器型号代号:

0- VCC3200,

1- VCC3500,

2- VCC4500。