I. VI4300 上电工作流程

VI4300 正常上电后，初始化配置及正常工作的流程如下：

1. VI4300 硬件正常供电，等待电源稳定；
2. 拉低 RESETB 引脚进行复位，等待 10ms 后再将其置高；
3. 芯片处于正常工作状态后，先通过 IIC 接口（VI4300 的 IIC 做从机）进行 VI4300 内部寄存器的配置；
4. IIC 配置完成后，从 VI4300 的 SPI 接口输出测距点的 TOF 值（LSB 原始数据）和强度值（PEAK 值）；
5. 在标定后，通过标定系数的处理，得到最终实际距离值等信息。

II. 封装库的介绍

例程中已提供 VI4300 基于 stm32f031G6U6 的 Lib 库。

在该 Lib 库中，共声明了四个函数，依次为：

1. VI4300 初始化函数：
   功能： 对 VI4300 IIC 和 SPI 端口的配置，IIC 对寄存器进行配置等操作。
   函数原型：uint16_t CHIP_Reg4300Init(VI4300ParaType *vi4300_para)。
   返回值：=0，表示 IIC 配置正常；
   !=0，表示 IIC 配置失败。
   形参：VI4300ParaType *vi4300_para。相关参数包含：
   vi4300_para.clock_seclect：VI4300 的内部时钟线选择，default 配置值为 0x98；
   vi4300_para.set_pulse_width: trig 输出电脉宽配置，default 配置值为 0x94，电脉宽约 3.5ns；
   vi4300_para.select_spad_number: SPAD 使能，default 配置值为 0x0003ffff，表示 18 个 SPAD 全使能,每一位表示 1 个 SPAD 使能；
   vi4300_para.clock_value：计数输出方式选择，default 配置值为 0x60；
   vi4300_para.points ：积分次数值，输出点的频率=75M/208/(points+4)；
   vi4300_para.ma_buffer[0-7] ：MA 窗口配置，default 配置值为 0x0f；

2. VI4300 内部电信号模拟激光测距信号的配置：
   功能：VI4300 内部电信号模拟激光测距信号的配置
   函数原型：void CHIP_DelayLineTestInit(uint8_t delay_line_clk, uint8_t delay_line_value);
   返回值：无
   形参:delay_line_clk:delay_line_clk 时钟选择,default 配置值为 0x98,可选择值为 0x81、0x88、0x91、0x98；delay_line_value：delay_line 的配置值，default 配置值为 0x78,可选择值为 0x78-0x7f；

3. VI4300 选择 SPAD 开启/关闭配置：
   功能：18 个 SPAD，选择对应的 SPAD 打开或者关闭
   函数原型：void VI4300_SinglePixelConfig(uint32_t select_spad_number);
   返回值：无
   形参：select_spad_number：32bit 数据，低 18bit,每 1bit 表示 18 个 SPAD 对应的开启或者关闭状态，对应 bit 位=1，表示相应的 SPAD 打开，-0，表示相应的 SPAD 关闭。default 配置值为 0x0003ffff，18 个 SPAD 全开；

4. VI4300 激光脉冲脉冲宽度控制：

功能：配置 VI4300 的 Trig 脚输出的控制激光脉冲宽度

函数原型：void VI4300_SinglePixelConfig(uint32_t select_spad_number);

返回值：无

形参：trig_val：tirg 电信号输出脉冲宽度值，default 配置值为 0x94,可配置值为 0x90-0x97，值越大，电信号脉冲宽度越宽。

III. 库移植

上文中,已经介绍了 VI4300 寄存器相关配置的 Lib 库封装的函数。目前封装库的文件名为:VI4300_Lib.lib。工程访问地址下（如下图 1 中例程所示）。

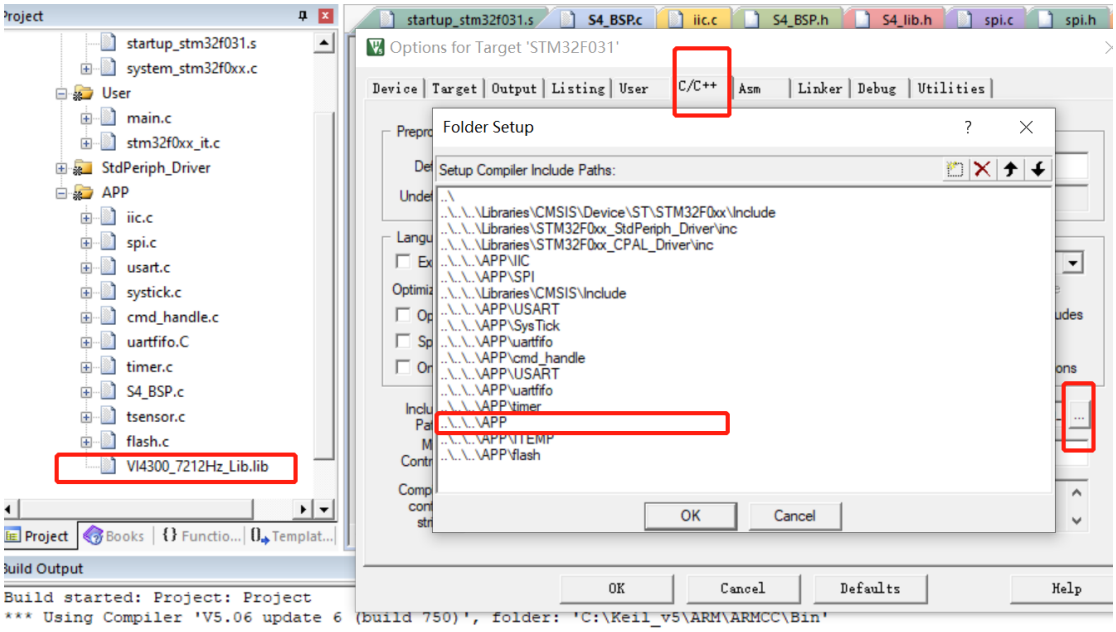

图 1 Lib 库在 keil 工程中的移植

IV. 例程介绍

提供的 VI4300 基于 STM32F031 的例程见附件中所示。

该例程实现了 7.2kHz 的点数据输出（实际输出频率为 1.8K，将输出数据 7.2K 进行了四分频输出，这是因为 7.2k 输出时，串口数据会导致上位机卡顿）。

同时，配合上位机，可实现 VI4300 模组的标定，单 Pixel 模式的读取等功能。

关于例程，主要从 VI4300 配置及数据输出，串口通信，温度补偿，VI4300 标定四个方面进行介绍：

1. VI4300 配置及数据输出
   a) 在 VI4300_BSP.c 文件中，函数 uint8_t VI4300_AllInit(void)为 VI4300 相关的初始化配置等，通过在主函数中调用该函数，可以实现对 VI4300 的所有初始化配置。
   b) 由于 VI4300 通过 SPI 输出数据（SPI 为 MASTER），所以为了准确实时的得到 VI4300 输出的测距点数据,MCU 采用 SPI_SLAVE_DMA 方案进行 VI4300 的获取。
   c) SPI_DMA 的 Buffer 命名为 DMA_RX_Buf[SPIOUTFIFO_SIZE]；其中该 Buffer 中，SPIOUTFIFO_SIZE 为一次 SPI_DMA 中断中采集数据的个数（4 个字节为一个点的数据，前两字节为 TOF 值，后两字节为 PEAK 值，数据位小端模式）

d) 在 SPI_DMA 中断中，g_SpecialCollectCount 变量用于单 Pixel 及 delayLine 相关处理时，TOF PEAK 值的采集等；g_SpiContinuFlag 用于判断是不是需要通过串口连续输出数据。如下图 2 所示。

```
123 void DMA1_Channel2_3_IRQHandler(void)
124 {
125   __packed uint16_t *tof_data_buffer = (uint16_t *)DMA_RX_Buf;
126   uint16_t tof_peak_buf[2] = {0};
127
128   if(DMA_GetFlagStatus(DMA1_IT_TC2) == SET)
129   {
130     DMA_ClearITPendingBit(DMA1_IT_TC2);
131     //原始TOF   = tof_data_buffer[0];
132     //PEAK      = tof_data_buffer[1];
133     if(g_SpecialCollectCount > 0) //校准状态的TOF采集
134     {
135       g_VI4300TofValue += tof_data_buffer[0];
136       g_VI4300PeakValue += tof_data_buffer[1];
137       g_SpecialCollectCount--;
138     }
139
140     if(g_SpiContinuFlag == 1) //连续读取
141     {
142       tof_peak_buf[0]=tof_data_buffer[0];
143       tof_peak_buf[1]=tof_data_buffer[1];
144       //直接存原始数据
145       spiinfifo_DataIn(tof_peak_buf);
146     }
147
148   }
149 }
```

图 2 SPI_DMA 中断的相关处理

e) SPI_DMA 采集的数据，在主函数的死循环中，有一个 TOF_AndAngleTransf ( ) 函数，会判断是按照原始数据(LSB)通过串口输出还是按照标定后的真实距离方式输出。当然，也有用于标定时，进行多次采集进行取平均的方式进行数据输出（调用函数原型为 uint16_t TOF_RealDistanceChange(uint16_t tof_value, uint16_t peak_value))。

f) 在函数 uint16_t TOF_RealDistanceChange(uint16_t tof_value, uint16_t peak_value)中，将 PEAK 值低于 PEAK 设定阈值的值认为是不可信数据，强制转换为最大值（默认 PEAK 不可信阈值为 20)。在转换为真实距离时，先进行了温度补偿，然后再通过标定的系数，将温度补偿后的 TOF PEAK 进行实际距离的转换。

2. 串口通信及通信协议

a) 在例程中，串口 TX 采用 DMA 输出，RX 采用中断采集的方式。

b) 默认上电时，直接进行 VI4300 TOF PEAK 值的输出，在主函数中，函数 DealDataFromUart1 ( ) 为串口接收数据的处理。

c) 串口接收数据的协议格式见附件<芯视界 DEMO 单点协议 20200702.docx>

3. 温度补偿

VI4300 的模组，温度对 VI4300 测距的结果是呈线性关系的，如下图 3 为实际测试的温度与 VI4300 输出的 TOF 原始值（LSB）的关系。所以只需要按照对应的线性关系，对温度进行补偿即可。注意：由于 VI4300 测距值，受芯片本身的温度影响，激光器发热导致的出光变慢，模组散热等很多因素影响，所以建议不同的模组结构进行实际温度曲线的测试补偿。

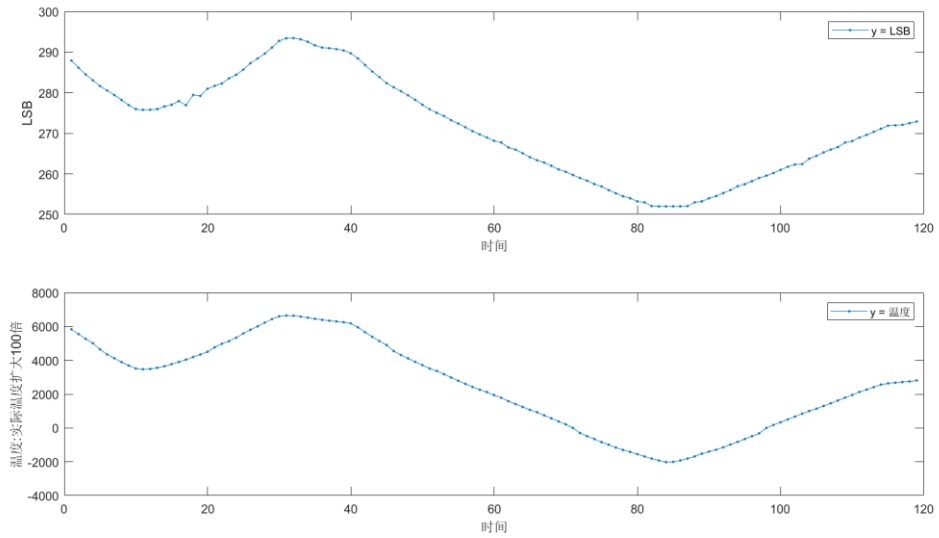

图 3 实际测试的温度与 VI4300 输出的 TOF 原始值（LSB）的关系

4. VI4300 标定

VI4300 模组近距离 0.1m-0.3m 由于光学盲区及 TX 和 RX 的 overlap 影响，距离偏差（bias）与强度值之间的线性关系与 0.3m 以后的的线性系数有明显区别。所以为了 0.1m-15m（实际目前只用到了 10m 即满足大多数场景的应用）满量程的测量距离的准确，进行了多段校正。

在进行标定之前，先将 VI4300 输出的 TOF 值转换为带有距离 Offset 的已经进行温度补偿过的数据，然后再进行标定。同时，在标定过程计标定完成后中，所有的温度均以开始标定时的温度作为基准温度。

目前标定标定分为 0.3m 以内,0.3-0.5m， 0.5m-1m， 1m 以外的共 4 段,8 个距离点(0.1m, 0.2m, 0.3m,0.4m,0.5m,1m,3m,8m）黑白（5%和 95%的标准反射率板）数据；每一段都是做分段线性处理，并通过 PEAK 值做了黑白的校正。校正原理为 0.3m（0.1m,0.2m,0.3m 三个距离的 6 个点数据）以内的标定点；0.3-0.5m（0.3m,0.4m,0.5m 三个距离的 6 个点数据）之间的标定点；0.5m-1m 两个距离的 4 个点数据；1m-8m（1m,3m,8m 三个距离 6 个点的数据）。每一段都进行标定点的线性拟合，以 PEAK 值作为校正的 X 轴坐标，Y 轴为校正的偏差值。该部分的处理是通过上位机实现的（线性拟合，单片机很难实现）。如下图 4 所示。

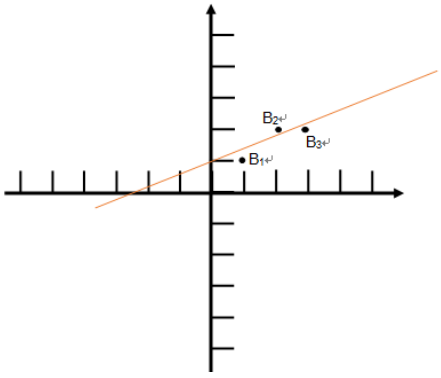

图 4 分段内点的线性拟合

上位机最终会计算出四段的每一段的系数及距离值的偏差以及 0.3m、0.5m、1m 的 TOF 阈值，然后将系数发送给 VI4300 模组，MCU 写进 Flash 中保存。

关于标定的具体处理，以下分为 MCU 部分及上位机相关的两部分介绍。

4.1 MCU 部分相关介绍

1) 得到上位机发送的新校准开始命令后，MCU 切换为芯视界协议的数据输出模式，此时输出的数据为进行了温度校正但带有 offset 的 mm 值距离及强度值输出的数据。

上位机发送的新校准开始的命令格式如下：

	帧头	命令	长度	地址	数据	校验
PC → Device	0x5A	0x00	0x0002	0x13	X	
Device → PC	0x5A	0x80	0x0002	0x13	X	

注：X=0x00，校准状态结束； X=0x01，校准状态开始

示例：

上位机发送：5A 00 02 00 13 01 E9（新校准开始命令）

MCU 回复 ：5A 80 02 00 13 01 69（新校准开始命令应答）

MCU 在收到上位机的新校准开始命令后，会做如下处理：

a) 把当前的温度记录下来，当做后面所有输出的温度校正但带有 offset 的 mm 值数据的温度补偿的基准系数。如下图 5 代码中的结构体中的校准时的温度变量 g_dataNeedSaved.distance_offset_temp。

b) 将数据的输出协议由科沃斯的 5A A5 + 距离值的数据更改为芯视界的数据输出协议。如下图 5 中的变量 g_dataNeedSaved.service_type。

芯视界的数据输出协议如下表所示：

	帧头	命令	长度	地址	数据	校验
Device → PC	0x5A	0x03	0x0005	0x0C	2-byte 距离值 + 2-byte 强度值	

例：5A 03 05 00 01 02 03 04 F1

5A	固定帧头
03	固定命令字
05 00	表示后面数据的长度
0C	数据主动上传地址
01 02	0x0201 = 十进制 512,表示距离值为 512mm
03 04	0x0403 = 十进制 1027，表示强度值为 1027
F1	从命令开始到数据区所有数据之和取反的低 8-bit 值

c) 置校准标志位，如下图 5 中的变量 g_CalibrationState。

d) 置连续上传数据的标志位，如下图 5 中的变量 g_SpiContinuFlag。

e) 回复上位机的命令（回复发命令见 4.1 中的示例）

```
case NEW_OFFSET_CALIBRATION_STATE_CMD: //新校准方案开始/结束
  if(pd[5] == 0x00) //校准结束
  {
    g_CalibrationState = 0;
  }
  else if(pd[5] == 0x01) //校准开始
  {
    g_dataNeedSaved.distance_offset_temp = JIZHUN_Get(g_temp);
    g_dataNeedSaved.service_type = 0x00; //先改为芯视界的协议
    g_CalibrationState = 1;
    g_SpiContinuFlag = 1;
  }
  i=0;
  _uart_send_pbuff[i++] = pd[5];
  UART_CmdAckSend(WRITE_CMD_ACK, NEW_OFFSET_CALIBRATION_STATE_CMD, _uart_
  break;
```

图 5 新校准开始命令的处理

2) 在完成了 1）中的任务后，MCU 会持续输出温度校正但带有 offset 的 mm 值及强度值数据。此时上位机会不断的根据滑台反馈的位置信息不断的处理对应的距离值数据。

3) 上位机将需要标定点的数据都获得并处理完成后，会发送新校准系数设置命令，将计算的系数等发送到 MCU 存储。

新校准系数设置命令协议如下表所示：

	帧头	命令	长度	地址	数据	校验
PC → Device	0x5A	0x00/0x01	0x003D/0x0001	0x12	X	
Device → PC	0x5A	0x80/0x81	0x003D	0x12	X	

注：X 数据格式为：

0.1-0.3m TOF 阈值	0.3-0.5m TOF 阈值	0.5-1m TOF 阈值	0.1-0.3m peak 阈值	0.3-0.5m peak 阈值	0.5-1m peak 阈值	1m 以外 peak 阈值	0.1-0.3m 系数 a	0.1-0.3m 系数 b	0.3-0.5m 系数 a	0.3-0.5m 系数 b	0.5-1m 系数 a	0.5-1m 系数 b
4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte
1m 以外系数 a	1m 以外系数 b											
4byte	4byte											

MCU 在新校准设置命令中，主要做了如下处理：

a) 将得到的系数数据存到结构体变量中。即变量 g_dataNeedSaved.calibration_buff 中的数据。如下图 6 所示。

b) 将输出的值转为修正所有 offset 后的真实 mm 值距离，即变量 g_dataNeedSaved.real_distance_LSB_select。如下图 6 所示。

c) 将以上信息存入 Flash 中。即函数 flash_SaveConfigerData();如下图 6 所示。

d) 将系数数据放到命令回复的缓冲区中，等待发送出去。即 for 循环中的处理。

e) 新系数设置命令的应答，协议的拼接等在 UART_CmdAckSend()函数中完成。如下图 6 所示。

f) 新系数设置命令的应答发送后，将数据设置为连续输出，即 g_SpiContinuFlag = 1;这是因为在现在的 Flash 中，会有 VI4300 的重新初始化部分，初始化后未进行数据的连续输出设置。如下

图 6 所示。

```
case NEW_OFFSET_CALIBRATION_CMD: //新校准方案
  if(pd[1] == 0x01) //写
  {
    for(j=0;j<15;j++)
    {
      g_dataNeedSaved.calibration_buff[j] = pd[5+4*j]
      +((pd[6+4*j]&0x000000ff)<<8)+((pd[7+4*j]&0x000000ff)<<16)
      +((pd[8+4*j]&0x000000ff)<<24);
    }
    g_dataNeedSaved.real_distance_LSB_select = 0; //转为实际距离输出

    flash_SaveConfigerData();
    g_CalibrationState = 0; //写入后，校准状态结束
    i = 0;
    for(j=0;j<15;j++)
    {
      _uart_send_pbuff[i++] = g_dataNeedSaved.calibration_buff[j];
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>8);
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>16);
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>24);
    }
    UART_CmdAckSend(WRITE_CMD_ACK, NEW_OFFSET_CALIBRATION_CMD, _uart_send_pbuff, i);
    delay_ms(10);
    g_SpiContinuFlag = 1; //连续输出
  }
  else //读命令
  {
    i = 0;
    for(j=0;j<15;j++)
    {
      _uart_send_pbuff[i++] = g_dataNeedSaved.calibration_buff[j];
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>8);
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>16);
      _uart_send_pbuff[i++] = (g_dataNeedSaved.calibration_buff[j]>>24);
    }
    UART_CmdAckSend(READ_CMD_ACK, NEW_OFFSET_CALIBRATION_CMD, _uart_send_pbuff, i);
  }
   break;
```

图 6 新校准设置命令的处理

4.2 上位机相关处理

1) 开启连续采集命令：

a) 开启采集命令（PC → MCU）：5A 01 02 00 08 00 F4；

b) MCU 应答完上位机数据后，模组主动发送 TOF&PEAK 值数据,数据格式如下：

	帧头	命令	长度	地址	数据	校验
Device → PC	0x5A	0x03	0x0005	0x0C	2-byte 距离值 + 2-byte 强度值	

2) 新校准开始命令：

(PC → MCU): 5A 00 02 00 13 01 E9

(MCU → PC): 5A 80 02 00 13 01 69

3) 获取校准点的值：

a) 与工装台建立串口通信：波特率：19200；数据位：8，停止位：1，偶检验；

b) PC 向工装复位命令；

c) 工装移动到 100mm 位置，放置黑色标定板，并开始采集 tof 和 peak，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_100, meanPeakB_100,（黑色标定板）；

d) 工装移动到 200mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_200, meanPeakB_200,（黑色标定板）；

e) 工装移动到 300mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_300, meanPeakB_300,（黑色标定板）；
f) 工装移动到 400mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_400, meanPeakB_400,（黑色标定板）；
g) 工装移动到 500mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_500, meanPeakB_500,（黑色标定板）；
h) 工装移动到 1000mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_1000, meanPeakB_1000,（黑色标定板）；
i) 工装移动到 3000mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_3000, meanPeakB_3000,（黑色标定板）；
j) 工装移动到 8000mm 位置，同样保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofB_800, meanPeakB_800,（黑色标定板）；
k) 工装停留在 8000mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_8000, meanPeakW_8000,（白色标定板）；
l) 工装移动在 3000mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_3000, meanPeakW_3000,（白色标定板）；
m) 工装移动在 1000mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_1000, meanPeakW_1000,（白色标定板）；
n) 工装移动在 500mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_500, meanPeakW_500,（白色标定板）；
o) 工装移动在 400mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_400, meanPeakW_400,（白色标定板）；
p) 工装移动在 300mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_300, meanPeakW_300,（白色标定板）；
q) 工装移动在 200mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_200, meanPeakW_200,（白色标定板）；
r) 工装移动在 100mm 位置，保存 2000 个点的 tof 和 peak 的均值，分别标记为：meanTofW_100, meanPeakW_100,（白色标定板）；

4) 数据预处理：
a) 校准部分分四段进行，故将保存的数据分为四组：（100mm、200mm、300mm），（300mm、400mm、500mm），（500mm，1000mm），（1000mm、3000mm、8000mm）；
b) 获取所有 TOF 偏差：tof 减去各自真实距离值（例如：100mm 黑偏差= meanTofB_100 - 100）；peak 值不变。

5) 获取校准参数：
校准参数包含：3 个 Tof 阈值，4 个 peak 阈值， 4 组拟合参数（A，B）
a) 3 个 TOF 阈值 tofThreshold[3]，其中：
tofThreshold[0]为第一组数据（100、200、300mm）TOF 的最大值（包含黑白）；
tofThreshold[1]为第二组数据（300、400、500mm）TOF 的最大值；
tofThreshold[2] 为第三组数据（500、1000mm）TOF 的最大值。
b) 4 个 peak 阈值 peakThreshold[4]，其中：
peakThreshold [0]为第一组数据（100、200、300mm）TOF 的最小值（包含黑白），然后再减去一个常数（暂定减去 10）；

peakThreshold [1]为第二组数据（300、400、500mm）TOF 的最小值，然后再减去一个常数（10）；

peakThreshold [2]为第三组数据（500、1000mm）TOF 的最小值，然后再减去一个常数（10）；

peakThreshold [3]为第四组数据（1000、3000、8000mm）TOF 的最小值，然后再减去一个常数（10）；

c) 第一组拟合参数（A[0],B[0]）：针对第一组数据（100、200、300）本组运算涉及到矩阵运算，为计算方便用到了 eigen 库；计算公式为

$$\theta = (X^T X)^{-1} X^T y \quad 。$$

其中 X 为 6x2 的矩阵, X 初始化时的第一列依次为：100mm 处黑色 meanPeakB_100，100mm 处白色 meanPeakW_100，200 处黑色 meanPeakB_200，200 处白色 meanPeakW_200，300 处黑色 meanPeakB_300，300 处白色 meanPeakW_300；X 的第二列全部设置为 1。

Y 为 6x1 的矩阵,Y 初始化时依次为：100mm 处黑色偏差，100mm 处白色偏差，200mm 处黑色偏差，200mm 处白色偏差，300mm 处黑色偏差，300mm 处白色偏差；

计算结果 Θ 为 2X1 的矩阵，A[0]= Θ(0,0), B[0]= Θ(1,0)。

Ps：计算是否正确可参考下图所示数据与结果：

```
meanTofB_100= 3042.83   meanPeakB_100= 151.2935;
meanTofW_100= 3042.83   meanPeakW_100= 151.2935;

meanTofB_200= 3141.19   meanPeakB_200= 134.2647458;
meanTofW_200= 3104.43   meanPeakW_200= 326.6861538;

meanTofB_300= 3231.057   meanPeakB_300= 154.266
meanTofB_300= 3197.566   meanPeakB_300=332.478;

拟合结果: A[0] = -0.2096   B[0] = 2969
```

d) 第二组拟合参数（A[1],B[1]）：针对第二组数据（300、400、500），处理方式同上，最后获取 A[1],B[1];

e) 第三组拟合参数（A[2],B[2]）：针对第三组数据（500，1000），处理方式同上，区别在于 X 此时为 4x2 的矩阵，最后获取 A[2],B[2];

f) 第四组拟合参数（A[3],B[3]）：针对第四组数据（1000，3000，8000），处理方式同(c)，同样获取 A[3],B[3]。

6) 将校准参数写入 MCU：

a) 经过步骤 5，共获取到 3 个 TOF 阈值 tofThreshold[3]，4 个 peak 阈值 peakThreshold[4]，4 组拟合参数（A[4],B[4]）；其中在发送前将 4 组拟合参数分别扩大 1000 倍；

b) 将阈值、拟合参数转换为小端模式下的 int32 类型；

c) 按照协议写入 MCU

	帧头	命令	长度	地址	数据	校验
PC → Device	0x5A	0x01	0x003D/0x0001	0x12	X	
Device → PC	0x5A	0x81	0x003D	0x12	X	

注：X 数据格式为：

0.1-0.3m TOF 阈值	0.3-0.5m TOF 阈值	0.5-1m TOF 阈值	0.1-0.3m peak 阈值	0.3-0.5m peak 阈值	0.5-1m peak 阈值	1m 以外 peak 阈值	0.1-0.3m 系数 a	0.1-0.3m 系数 b	0.3-0.5m 系数 a	0.3-0.5m 系数 b	0.5-1m 系数 a	0.5-1m 系数 b
4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte	4byte
1m 以外系数 a	1m 以外系数 b											
4byte	4byte											

上位机发送命令为：　5A 01 3D 00 12 + 15 个数据（60-byte） + 校验（1-byte）
MCU 回复：　5A 81 3D 00 12 + 15 个数据（60-byte） + 校验（1-byte）

V．上位机的使用

1. 打开标定上位机程序（VI4300_6.8.exe），界面如下图 7 所示。

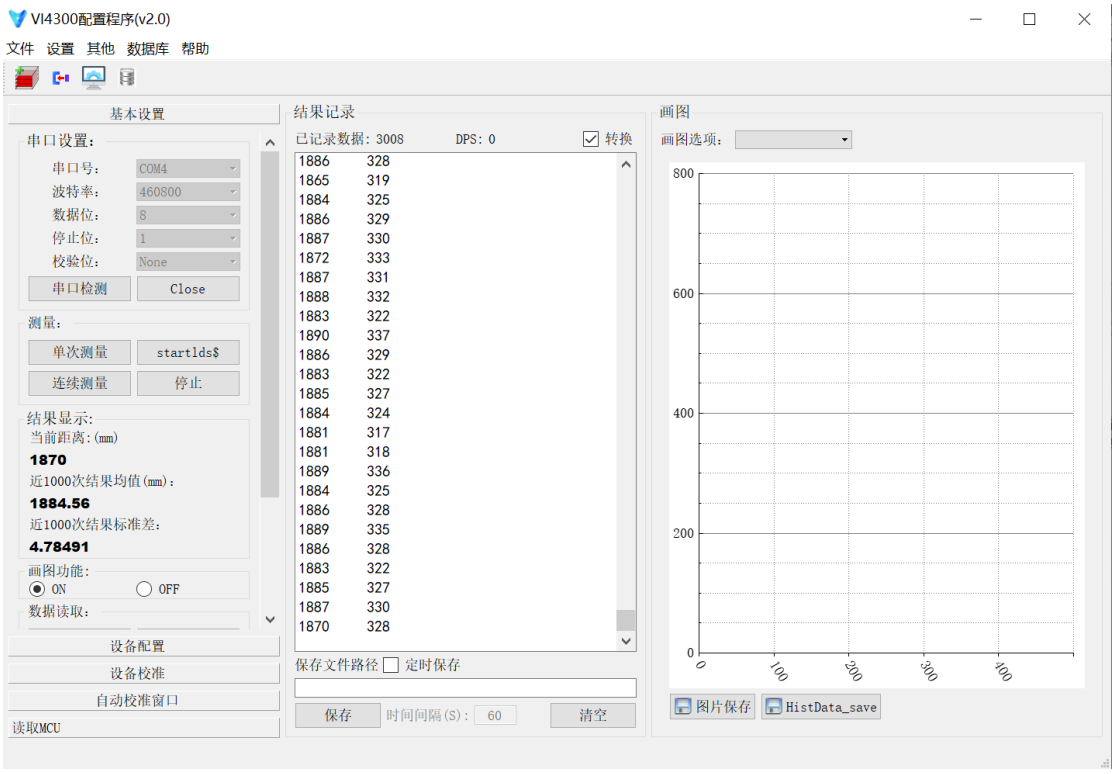

图 7 VI4300 上位机主界面

2. 点击左侧设置区按钮“基本设置”，下拉的界面如下图 8 所示。

图 8 基本设置界面

3. 点击上述图 8 中下拉菜单里的“串口检测”按钮，选择正确的串口号，然后点击“串口检测”右边的“Open”按钮，完成 VI4300 模组通讯连接。
4. 点击图 8 左侧设置区按钮“自动校准窗口”，下拉的界面如下图 9 所示，自动校准窗口可连接测试滑台（8m 导轨平台）的控制接口（芯视界自己搭建，该部分可根据自己情况选择使用）。但由于该部分因为设备的关系，客户可自己直接点击对应的距离处的不同反射率的按钮，通过手动的方式将测距值发送到上位机测试。需要注意的是，该处 8m 的白色按钮需要将前面 15 个点的数据都采集完才能点击采集，否则会导致标定数据错误，其余 15 个点的数据无先后顺序。如下图 9 所示。

图 9 自动校准窗口界面

5. 如 4 中所说，由于用户很难有匹配芯视界协议的滑轨完成自动标定，所以图 9 中的串口和复位按钮都无效。可以直接点击“开启自动校正”按钮，开始 VI4300 模组的校正。
6. 点击过“开启自动校正”按钮后，可看到数据区的数据回复很慢，在 14Hz 左右，这是因为此时处于校正状态，MCU 会自动将 TOF 和 PEAK 值进行多次取平均后输出，将黑色（5%反射率）目标板依次放置在距离 VI4300 模组 0.1m,0.2m,0.3m,0.4m,0.5m,1m,3m,8m 处,并且在每个距离处点击对应距离处的“黑”按钮；同理，将白色（95%反射率）目标板依次放置在距离 VI4300 模组 0.1m,0.2m,0.3m,0.4m,0.5m,1m,3m,8m 处,并且在每个距离处点击对应距离处的“白”按钮。最后标定 8m“白”按钮后，若标定成功，此时上位机会提示“标定完成”提示框。
7. 至此，标定完成。如下图 10 所示，点击“设备校准”菜单栏，在此页面下，可以实现最大量程和 PEAK 阈值的设置，其余的参数目前由于采用了新校准方案，无效。该命令有“设置”和“读取”两个按钮，用户可自行选择。

图 10 设备校准界面

8. 在设备配置界面，有波特率，输出方式为原始 LSB 或者真实距离等方案的选择，用户可自行选择。如下图 11 所示。

图 11 设备配置界面

9. 单 PIXEL 模式用于 VI4300 模组光学的调整。
   如下图 12 所示，在基本设置界面，有很多个按钮，其中红色框中的按钮，可用于单 PIXEL 模式的数据读取，该模式可以理解为，Vi430 接收到的光信号在 18 个宏像素上的 PEAK 值。

图 12 基本设置界面单 PIXEL 模式的读取相关操作

具体操作方法为：

将目标板移动到 10m 处（比如 95%反射率距离 VI4300 模组 10m），设定好单 PIXEL 定时读取时间（最快 100ms），然后点击“pixel 定时读取”按钮。此时右侧的文本框会定时输出 VI4300 的单 PIXEL 结果（3 行 6 列）。如下图 13 所示。光斑在中间处时，测距效果较好（如下图 13 所示，结果为光斑偏左，可以再往左调整一些）。

13	11	9	5	3	2
19	18	16	13	7	4
19	20	18	15	11	5

图 13 单 Pixel 结果

协议中单 PIXEL 的结果发送顺序如下表所示：

1 号	4 号	7 号	10 号	13 号	16 号
2 号	5 号	8 号	11 号	14 号	17 号
3 号	6 号	9 号	12 号	15 号	18 号

VI. 常见问题解答

1. VI4300 近距离能不能做到更高精度，现在近距离 LDS 建图可看到一段段弧线？
   这是由于 VI4300 的分辨率为 1.555cm，在此分辨率以内。很难实现快速采集数据的更高分辨率。

2. VI4300 的激光脉宽需要多少，光功率需要多少？
   Datasheet 中推荐电路，脉宽配置寄存器的值配置为 0x94,电脉宽 3.5ns，以芯视界的 8mm 模组为例，光脉宽（半宽）约 1.6-1.8ns，峰值光功率约 3W,该参数测距效果 10m 没有压力。