ParamMate 使用说明

目录

更新说明 3
一、 简介 4
二、 程序移植 5
2.1 文件移植 5
2.2 接口移植 5
三、 SDK 说明 6
3.1 宏定义说明 6
3.2 枚举说明 6
3.2.1 参数读写类型枚举 6
3.2.2 数据类型枚举 7
3.2.3 示波线条类型枚举 7
3.2.4 图像类型枚举 7
3.3 结构体说明 8
3.3.1 参数控件结构体 8
3.3.2 示波控件结构体 8
3.3.2 图传控件结构体 8
3.4 函数说明 9
3.4.1 复位窗口 9
3.4.2 复位窗口 9
3.4.3 创建参数控件 9
3.4.4 创建参数通道 10
3.4.5 创建示波控件 10
3.4.6 创建示波通道 11
3.4.7 创建图传控件 11
3.4.8 发送参数控件数据 12
3.4.9 发送示波控件数据 12
3.4.10 发送图传控件数据 12
3.4.11 回传数据处理函数 13
四、 使用方法 14
4.1 上位机使用方法 14
4.1.1 上位机布局介绍 14
4.1.2 MainWindow 介绍 15
4.1.3 图传控件介绍 16
4.1.4 示波控件介绍 17
4.1.5 参数控件介绍 18
4.2 下位机使用方法 18
4.2.1 下位机初始化流程 18
4.2.2 下位机发送数据流程 20

更新说明

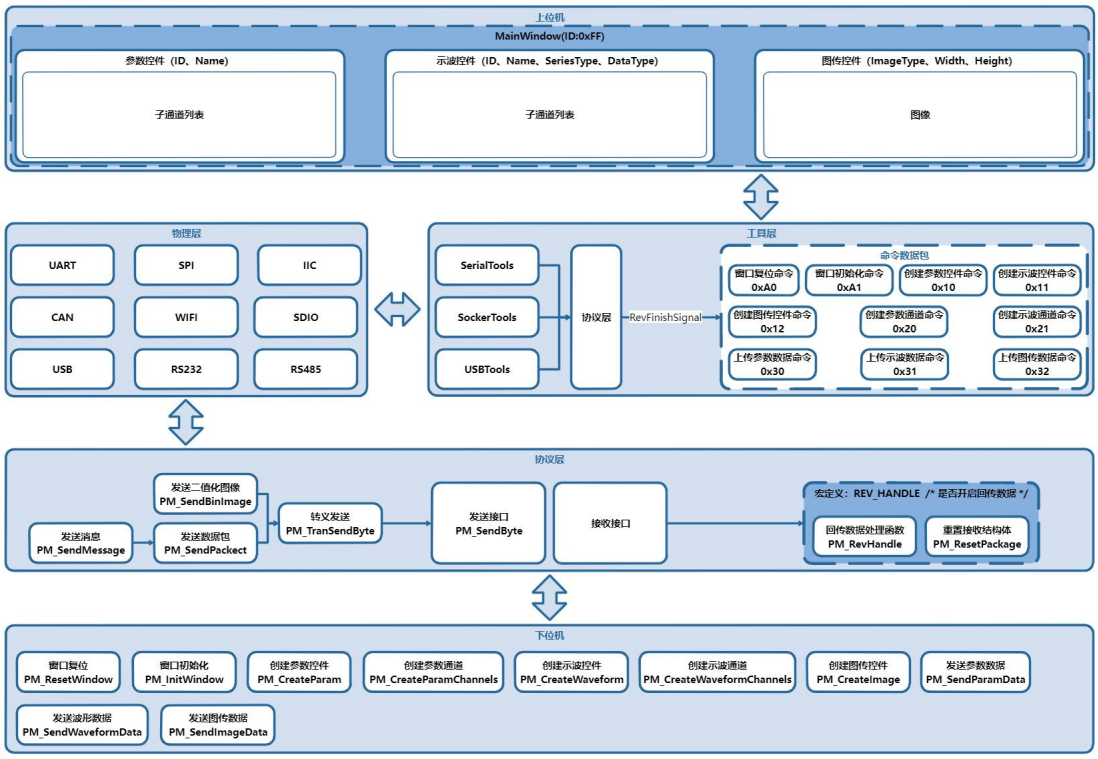
版本	作者	时间	备注
V1.0.0	满心欢喜	20220812	初始版本
V1.1.0	满心欢喜	20220817	将 QtSerialPort 替换为 pyserial，添加串口工具配置文件

一、简介

ParamMate 是一个集调参、示波、图传的多功能调试助手。采取模块化的设计思路将各个功能以控件的形式拼装组合使用，用户可以根据下位机的不同初始化语句来 DIY 上位机页面组成而无需像传统调试助手一样需要两头配置（配置下位机之后还要再上位机进行相应配置），这样可以大大提升调试效率。

ParamMate 采用自定义通信协议，可承载于多种通信方式如：串口、IIC、SPI、UART、USB、CAN、SDIO、WIFI 等，这样无需关心具体数据传输是如何实现的只需要将通信数据完整传输到上位机即可。就像计算机网络中应用层无需关心物理层是如何实现传输的。封装好的通信协议栈，用户只需要将数据 IO 接口配置好，就可以实现调用 SDK 实现所有功能。

ParamMate 最大支持一个 32 通道调参控件、三个 8 通道示波控件、两个图传控件。支持读模式、写模式、读写模式调参，支持折线图、样条线图、条形图示波，支持二值化图、灰度图、RGB565 彩图等图传。



二、程序移植

2.1 文件移植

①将 ParamMate.lib、PM_CommProt.h、ParamMate.h 三个文件放入工程路径下任意位置（若为逐飞驱动库可将其放在 Project\CODE 文件夹下）。

②打开工程将 ParamMate.lib、PM_CommProt.h、ParamMate.h 添加到工程当中。

③将 PM_CommProt.h、ParamMate.h 文件路径添加至工程头文件路径（若为逐飞驱动库放入 CODE 文件夹下无需此操作）。

2.2 接口移植

①打开 PM_CommProt.h 文件将 PM_SendByte(dat)宏定义修改为自己的 UART 输出数据函数。

```
16 /* @define PM_SendByte */
17 #define     PM_SendByte(dat)                uart_putchar(UART1, dat)
```

②若需要参数控件下传数据还需适配 PM_RevHandle 函数，若不需要无需适配。

将 PM_RevHandle 函数放入串口接收中断中将参数控件结构体指针与接收的数据传入函数即可（若不放在中断服务函数也可以，只要确保每次 UART 接收数据可以正确传入函数即可）。

三、SDK 说明

在 SDK 使用过程中创建的控件结构体为 Malloc 生成，会占用部分静态区内存，可能会导致 Malloc 申请失败，请在创建控件之后判断一下控件地址是否为 PM_NULL，若为 PM_NULL 则创建失败。

所有的控件结构体存储着控件的所有信息，创建成功返回的结构体指针建议存储在全局变量，也存在局部变量中，但请注意变量的作用域。

3.1 宏定义说明

名称	数值	描述
REV_HANDLE	1	是否开启回传数据
PM_MAIN_WINDOW_ID	255	主窗口 ID
PM_MAX_NAME_LEN	32	名称最大字节数
PM_MAX_PARAM	1	最大支持参数控件数
PM_MAX_WAVEFORM	3	最大支持示波控件数
PM_MAX_IMAGE	2	最大支持图传控件数
PM_MAX_PARAMCANNELS	32	参数最大通道数
PM_MAX_WAVEFORMCANNELS	8	示波最大通道数
PM_MAX_WIDTH	188	图像最大宽度
PM_MAX_HEIGHT	120	图像最大高度

3.2 枚举说明

3.2.1 参数读写类型枚举

RWMode_Type		
名称	数值	描述
R_Type	1	只读类型
W_Type	2	只写类型
RW_Type	3	读写类型

3.2.2 数据类型枚举

Data_Type		
名称	数值	描述
uint8_Type	0	uint8 型
uint16_Type	1	uint16 型
uint32_Type	2	uint32 型
int8_Type	3	int8 型
int16_Type	4	int16 型
int32_Type	5	int32 型
float_Type	6	float 型

3.2.3 示波线条类型枚举

Series_Type		
名称	数值	描述
LineSeries_Type	0	折线图
SplineSeries_Type	1	样条线图
BarSeries_Type	2	条形图

3.2.4 图像类型枚举

Image_Type		
名称	数值	描述
Binarization_Type	0	二值化图
Grayscale_Type	1	灰度图
RGB565_Type	2	RGB565 彩图

3.3 结构体说明

3.3.1 参数控件结构体

PM_Param_t		
类型	名称	描述
PM_uint8	ID	ID
PM_uint8	Channels	参数通道数
PM_uint8	DataType[PM_MAX_PARAMCANNELS]	参数数据类型列表
void*	DataAddrList[PM_MAX_PARAMCANNELS]	参数数据地址列表

3.3.2 示波控件结构体

PM_Waveform_t		
类型	名称	描述
PM_uint8	ID	ID
PM_uint8	DataType	数据类型
PM_uint8	Channels	示波通道数
void*	DataAddrList[PM_MAX_WAVEFORMCANNELS]	示波数据地址列表

3.3.2 图传控件结构体

PM_Image_t		
类型	名称	描述
PM_uint8	ID	ID
PM_uint8	ImageType	图像类型
PM_uint8	Height	图像高度
PM_uint8	Width	图像宽度
void*	DataAddr	图像数据地址

3.4 函数说明

3.4.1 复位窗口

void PM_ResetWindow(void)	
名称	PM_ResetWindow
描述	复位窗口
参数	void
返回值	void
示例	PM_ResetWindow()

3.4.2 复位窗口

void PM_InitWindow(void)	
名称	PM_InitWindow
描述	初始化窗口
参数	void
返回值	void
示例	PM_InitWindow()

3.4.3 创建参数控件

PM_Param_t* PM_CreateParam(PM_uint8 ID, const PM_int8* Name)	
名称	PM_CreateParam
描述	创建参数控件
参数	ID　　控件 ID（保证全局唯一性）
参数	Name　　控件名称（标题）
返回值	参数控件结构体指针
示例	PM_CreateParam(0x00, "Test_param")

3.4.4 创建参数通道

PM_err PM_CreateParamChannels(PM_Param_t* PM_Param, const PM_int8* Name, RWMode_Type ModeType,Data_Type DataType, void* DataAddr)	
名称	PM_CreateParamChannels
描述	创建参数通道
参数	PM_Param　　参数控件指针
参数	Name　　数据名称
参数	ModeType　　读写类型
参数	DataType　　数据类型
参数	DataAddr　　数据地址
返回值	错误码　PM_EOK 为创建成功　PM_EFULL 为创建失败
示例	PM_CreateParamChannels(PM_Param,"Test1",RW_Type,int8_Type, &test1)

3.4.5 创建示波控件

PM_Waveform_t* PM_CreateWaveform(PM_uint8 ID, const PM_int8* Name, Series_Type SeriesType, Data_Type DataType)	
名称	PM_CreateWaveform
描述	创建示波控件
参数	ID　　控件 ID（保证全局唯一性）
参数	Name　　控件名称（标题）
参数	SeriesType　控件类型：示波线条类型枚举
参数	DataType　　数据类型
返回值	示波控件结构体指针
示例	PM_CreateWaveform(0x00,"Test_Waveform",SplineSeries_Type,uint16_Type)

3.4.6 创建示波通道

PM_err PM_CreateWaveformChannels(PM_Waveform_t* PM_Waveform, const PM_int8* Name, void* DataAddr)	
名称	PM_CreateWaveformChannels
描述	创建示波通道
参数	PM_Waveform　　示波控件结构体指针
参数	Name　　数据名称
参数	DataAddr　　数据地址
返回值	错误码　PM_EOK 为创建成功　PM_EFULL 为创建失败
示例	PM_CreateWaveformChannels(PM_Waveform,"Test", &Test)

3.4.7 创建图传控件

PM_Image_t* PM_CreateImage(PM_uint8 ID, const PM_int8* Name, Image_Type ImageType, PM_uint8 Height, PM_uint8 Width, void* DataAddr)	
名称	PM_CreateImage
描述	创建图传控件
参数	ID　　控件 ID（保证全局唯一性）
参数	Name　　控件名称（标题）
参数	ImageType　图像类型：详见图像类型枚举
参数	Height　　图像高度
参数	Width　　图像宽度
参数	DataAddr　　图像地址
返回值	图传控件结构体指针
示例	PM_CreateImage(0x00,"TestImage1",Binarization_Type,60,90,image)

3.4.8 发送参数控件数据

void PM_SendParamData(PM_Param_t* PM_Param)	
名称	PM_SendParamData
描述	发送参数控件数据
参数	PM_Param　　参数控件结构体
返回值	void
示例	PM_SendParamData(PM_Param)

3.4.9 发送示波控件数据

void PM_SendWaveformData(PM_Waveform_t* PM_Waveform)	
名称	PM_SendWaveformData
描述	发送示波控件数据
参数	PM_Waveform 示波控件结构体
返回值	void
示例	PM_SendWaveformData(PM_Waveform)

3.4.10 发送图传控件数据

void PM_SendImageData(PM_Image_t* PM_Image)	
名称	PM_SendImageData
描述	发送图传控件数据
参数	PM_Image　　图传控件结构体
返回值	void
示例	PM_SendImageData(PM_Image)

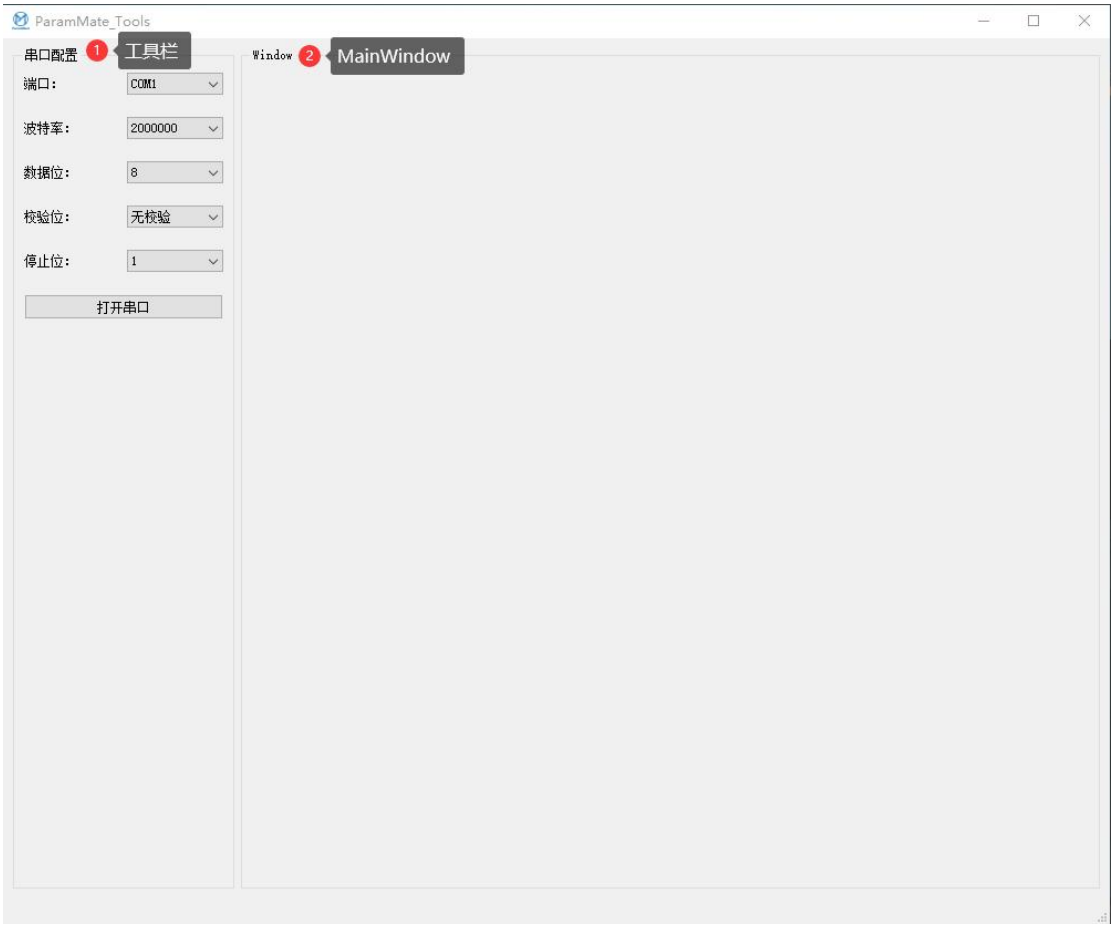
3.4.11 回传数据处理函数

void PM_RevHandle(PM_Param_t* Param, PM_uint8 dat)	
名称	PM_RevHandle
描述	回传数据处理函数
参数	Param　参数控件结构体指针
参数	dat　　接收字节
返回值	void
示例	PM_RevHandle(PM_Param, dat)

四、使用方法

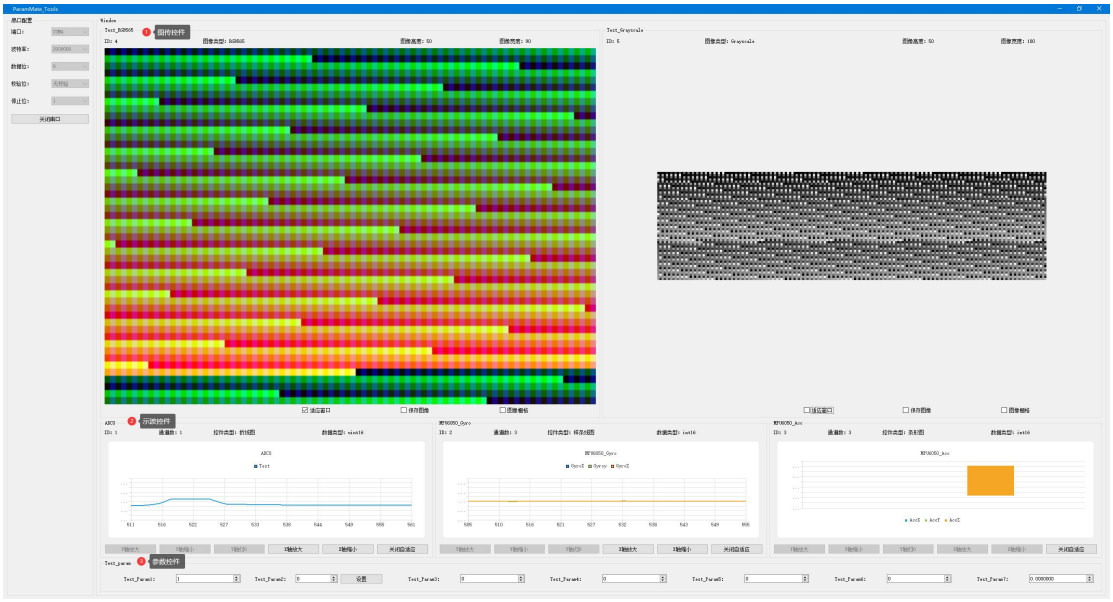
4.1 上位机使用方法

4.1.1 上位机布局介绍



①工具栏：主要为与物理层连接的工具，负责传输与下位机通信的数据，初代版本为串口工具。

②Window 栏：MainWindow 栏各个空间的容器，ID 默认为 0xFF。



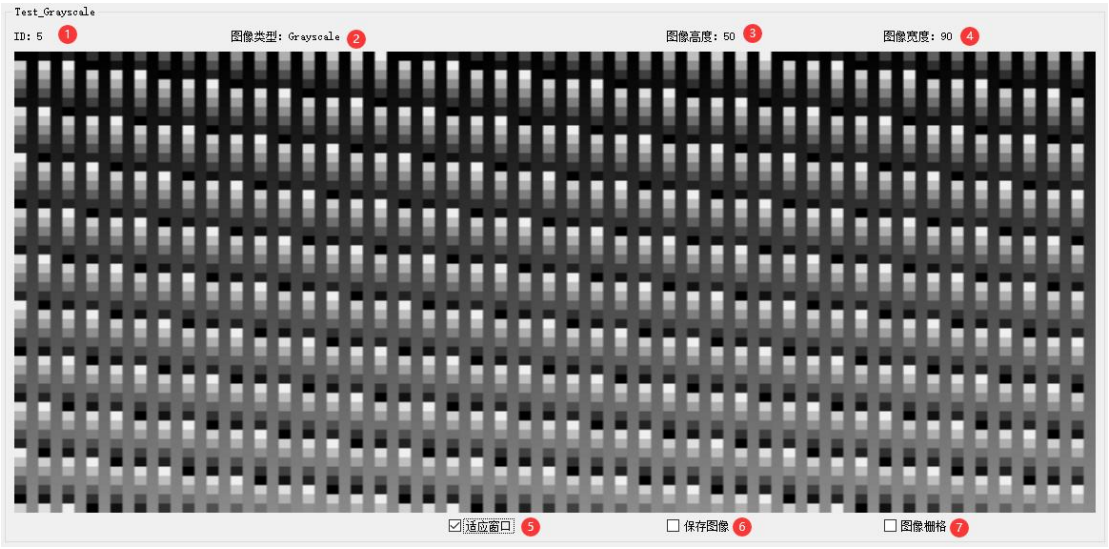
①图传控件、②示波控件、③参数控件

4.1.2 MainWindow 介绍

MainWindow 为所有控件的母容器，下位机创建的所有的控件均添加到该容器内，默认 ID 为 0xFF。

MainWindow 内所有控件均可自由缩放。

4.1.3 图传控件介绍



①控件 ID 号：由下位机初始化获得。

②图像类型：由下位机初始化获得。

③图像高度：由下位机初始化获得。

④图像宽度：由下位机初始化获得。

⑤适应窗口：勾选该选项则将图片铺满控件，否则将以原始比例显示。

⑥保存图像：勾选该选项将自动保存所有接收到的图像，**鼠标左键双击图像也可保存图像**。图像保存路径为以控件名称命名的文件夹下。

⑦图像栅格：将单位像素框选。

4.1.4 示波控件介绍

①控件 ID 号：由下位机初始化获得。

②通道数：由下位机创建的通道总数。

③控件类型：由下位机初始化获得。

④数据类型：由下位机初始化获得。

⑤Y 轴放大：在非自适应模式下 Y 轴放大。

⑥Y 轴缩小：在非自适应模式下 Y 轴缩小。

⑦Y 轴归 0：在非自适应模式下 Y 轴中心归 0，**双击控件背景也可 Y 轴中归 0。**

⑧X 轴放大：X 轴坐标放大。

⑨X 轴缩小：X 轴坐标缩小。

⑩关闭自适应：开关自适应模式按键。

示波控件默认开启自适应模式，该模式会将数据完整的显示在窗口内。

4.1.5 参数控件介绍

单个参数通道是由通道名称和输入框组成的，若通道为只写模式（W_Type）还会有一个设置按钮。

强烈推荐若无需修改通道数值请设置为“**只读模式**”，若无需知道通道数值请设置为“**只写模式**”。因为读写模式是双向传输的在上位机写的时候下位机同时在上传，很有可能会造成修改失败的情况。

4.2 下位机使用方法

4.2.1 下位机初始化流程

①复位 MainWindow 控件：调用 **PM_ResetWindow** 函数，发送窗口复位命令。

②根据需求创建控件并添加通道：

调用 **PM_CreateParam** 函数，创建示波控件，将其返回的参数控件结构体指针保存到全局变量，判断其是否为 PM_NULL，若为 PM_NULL 则创建失败。其他控件同理。

调用 **PM_CreateParamChannels**，创建示波通道判断其返回的错误码是否为 PM_EOK，若为 PM_EOK 则创建成功。其他通道同理。

③初始化 MainWindow 控件：调用 **PM_InitWindow** 函数，发送窗口初始化命令。

示例程序：

```
PM_Param_t* PM_Param;
PM_Waveform_t* PM_Waveform[3];
PM_Image_t* PM_Image[2];
uint16 adc;
int8 test1 = 1;
int16 test2 = 0;
int32 test3 = 0;
uint8 test4 = 0;
uint16 test5 = 0;
uint32 test6 = 0;
float test7 = 0.0;
uint16 Image[50][90] = {0};

void Window_init(void)
{
    /* 复位 MainWindow 控件 */
    PM_ResetWindow();

    /* Test_param 控件 */
    PM_Param = PM_CreateParam(0x00, "Test_param");
    PM_CreateParamChannels(PM_Param, "Test_Param1",R_Type, int8_Type, &test1);
    PM_CreateParamChannels(PM_Param, "Test_Param2",W_Type, int16_Type, &test2);
    PM_CreateParamChannels(PM_Param, "Test_Param3",RW_Type, int32_Type, &test3);
    PM_CreateParamChannels(PM_Param, "Test_Param4",RW_Type, uint8_Type, &test4);
    PM_CreateParamChannels(PM_Param, "Test_Param5",RW_Type, uint16_Type, &test5);
    PM_CreateParamChannels(PM_Param, "Test_Param6",RW_Type, uint32_Type, &test6);
    PM_CreateParamChannels(PM_Param, "Test_Param7",RW_Type, float_Type, &test7);

    /* ADC0 控件 */
    PM_Waveform[0] = PM_CreateWaveform(0x01, "ADC0", LineSeries_Type, uint16_Type);
    PM_CreateWaveformChannels(PM_Waveform[0], "Test", &adc);

    /* MPU6050_Gyro 控件 */
    PM_Waveform[1] = PM_CreateWaveform(0x02, "MPU6050_Gyro", SplineSeries_Type, int16_Type);
    PM_CreateWaveformChannels(PM_Waveform[1], "GyroX", &mpu_gyro_x);
    PM_CreateWaveformChannels(PM_Waveform[1], "Gyroy", &mpu_gyro_y);
    PM_CreateWaveformChannels(PM_Waveform[1], "GyroZ", &mpu_gyro_z);

    /* MPU6050_Acc 控件 */
    PM_Waveform[2] = PM_CreateWaveform(0x03, "MPU6050_Acc", BarSeries_Type, int16_Type);
    PM_CreateWaveformChannels(PM_Waveform[2], "AccX", &mpu_acc_x);
    PM_CreateWaveformChannels(PM_Waveform[2], "AccY", &mpu_acc_y);
    PM_CreateWaveformChannels(PM_Waveform[2], "AccZ", &mpu_acc_z);

    /* Test_RGB565 控件 */
    PM_Image[0] = PM_CreateImage(0x04, "Test_RGB565", RGB565_Type, 50, 90, Image);

    /* Test_Grayscale 控件 */
    PM_Image[1] = PM_CreateImage(0x05, "Test_Grayscale", Grayscale_Type, 50, 90, Image);

    /* 初始化 MainWindow */
    PM_InitWindow();
}
```

4.2.2 下位机发送数据流程

①在需要发送数据的地方调用 **PM_SendWaveformData** 函数，即可将控件数据发送到上位机。

其他控件同理。

示例程序：

```
while(1)
{
    gpio_toggle(C0);

    /* 更新数据 */
    adc = adc_convert(ADC_1, ADC1_CH00_A00);
    get_accdata();
    get_gyro();

    /* 发送数据 */
    PM_SendParamData(PM_Param);
    PM_SendWaveformData(PM_Waveform[0]);
    PM_SendWaveformData(PM_Waveform[1]);
    PM_SendWaveformData(PM_Waveform[2]);
    PM_SendImageData(PM_Image[0]);
    PM_SendImageData(PM_Image[1]);

    /* 延时 */
    systick_delay_ms(20);
}
```