

**ParamMate使用说明**

目录

[更新说明 3](#_Toc8129)

[一、 简介 4](#_Toc32398)

[二、 程序移植 5](#_Toc6137)

[2.1文件移植 5](#_Toc7213)

[2.2接口移植 5](#_Toc2148)

[三、 SDK说明 6](#_Toc23226)

[3.1宏定义说明 6](#_Toc23160)

[3.2枚举说明 6](#_Toc1009)

[3.2.1参数读写类型枚举 6](#_Toc9881)

[3.2.2数据类型枚举 7](#_Toc21976)

[3.2.3示波线条类型枚举 7](#_Toc25889)

[3.2.4图像类型枚举 7](#_Toc30758)

[3.3结构体说明 8](#_Toc29863)

[3.3.1参数控件结构体 8](#_Toc13797)

[3.3.2示波控件结构体 8](#_Toc1940)

[3.3.2图传控件结构体 8](#_Toc25921)

[3.4函数说明 9](#_Toc23362)

[3.4.1复位窗口 9](#_Toc30360)

[3.4.2复位窗口 9](#_Toc27964)

[3.4.3创建参数控件 9](#_Toc11595)

[3.4.4创建参数通道 10](#_Toc18267)

[3.4.5创建示波控件 10](#_Toc22501)

[3.4.6创建示波通道 11](#_Toc457)

[3.4.7创建图传控件 11](#_Toc19079)

[3.4.8发送参数控件数据 12](#_Toc21052)

[3.4.9发送示波控件数据 12](#_Toc13765)

[3.4.10发送图传控件数据 12](#_Toc31201)

[3.4.11回传数据处理函数 13](#_Toc1365)

[四、 使用方法 14](#_Toc12243)

[4.1上位机使用方法 14](#_Toc3385)

[4.1.1上位机布局介绍 14](#_Toc21522)

[4.1.2 MainWindow介绍 15](#_Toc16726)

[4.1.3图传控件介绍 16](#_Toc22350)

[4.1.4示波控件介绍 17](#_Toc21132)

[4.1.5参数控件介绍 18](#_Toc26124)

[4.2下位机使用方法 18](#_Toc9681)

[4.2.1下位机初始化流程 18](#_Toc6633)

[4.2.2下位机发送数据流程 20](#_Toc31961)

# 更新说明

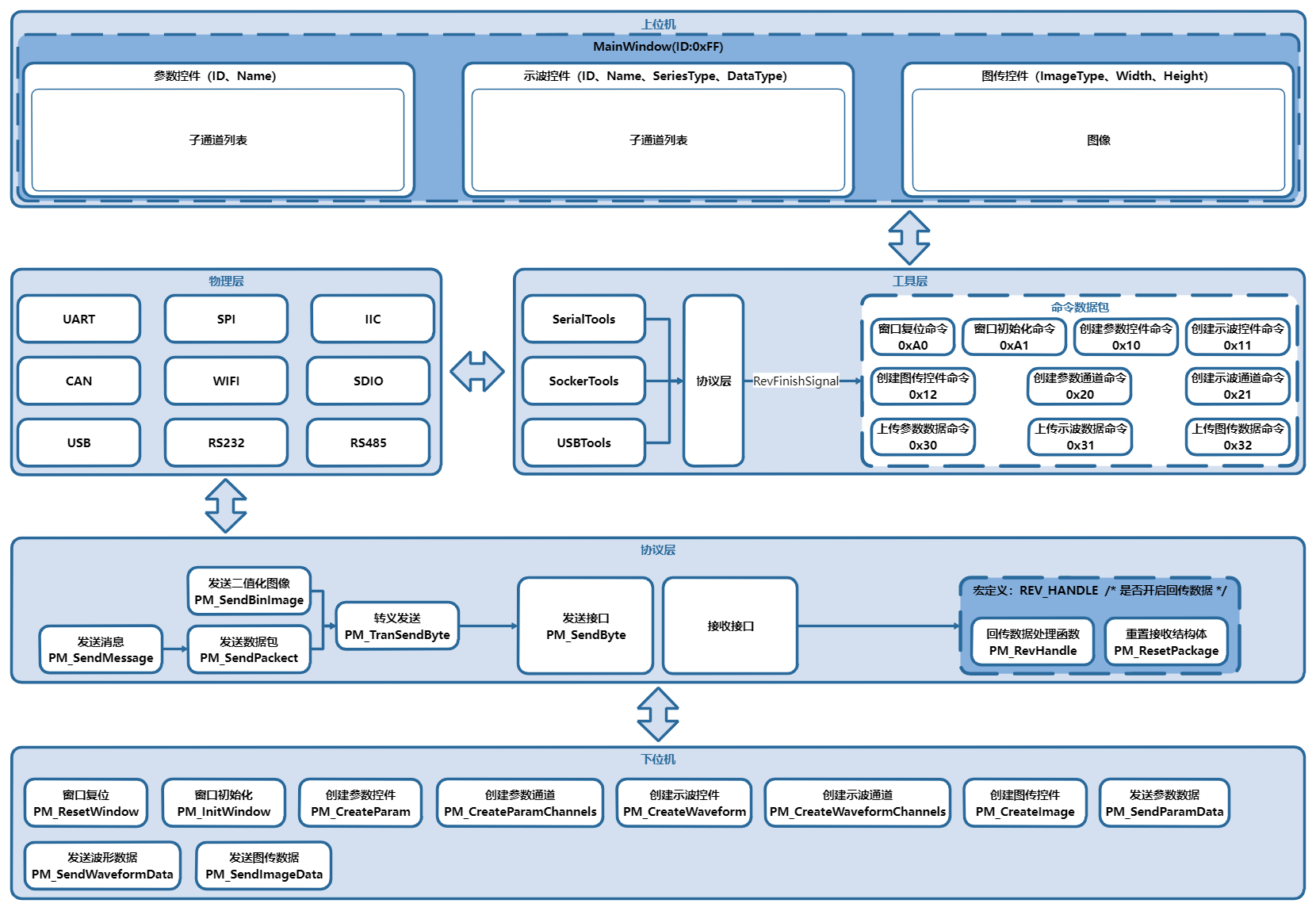
|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 时间 | 备注 |
| V1.0.0 | 满心欢喜 | 20220812 | 初始版本 |

# 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。



# 程序移植

## 2.1文件移植

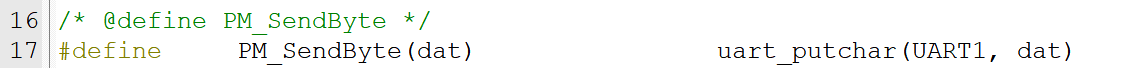
①将ParamMate.lib、PM\_CommProt.h、ParamMate.h三个文件放入工程路径下任意位置（若为逐飞驱动库可将其放在Project\CODE文件夹下）。

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。



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）。

# 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) |

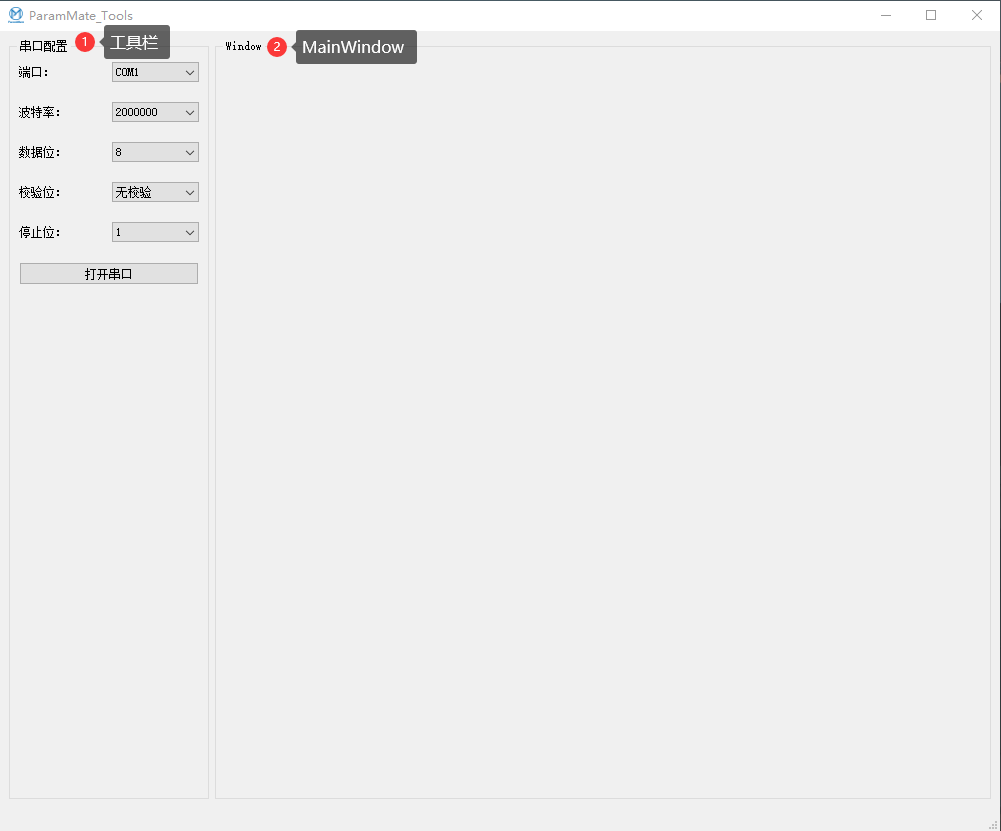
## **3.4.11回传数据处理函数**

|  |  |
| --- | --- |
| **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) |

# 使用方法

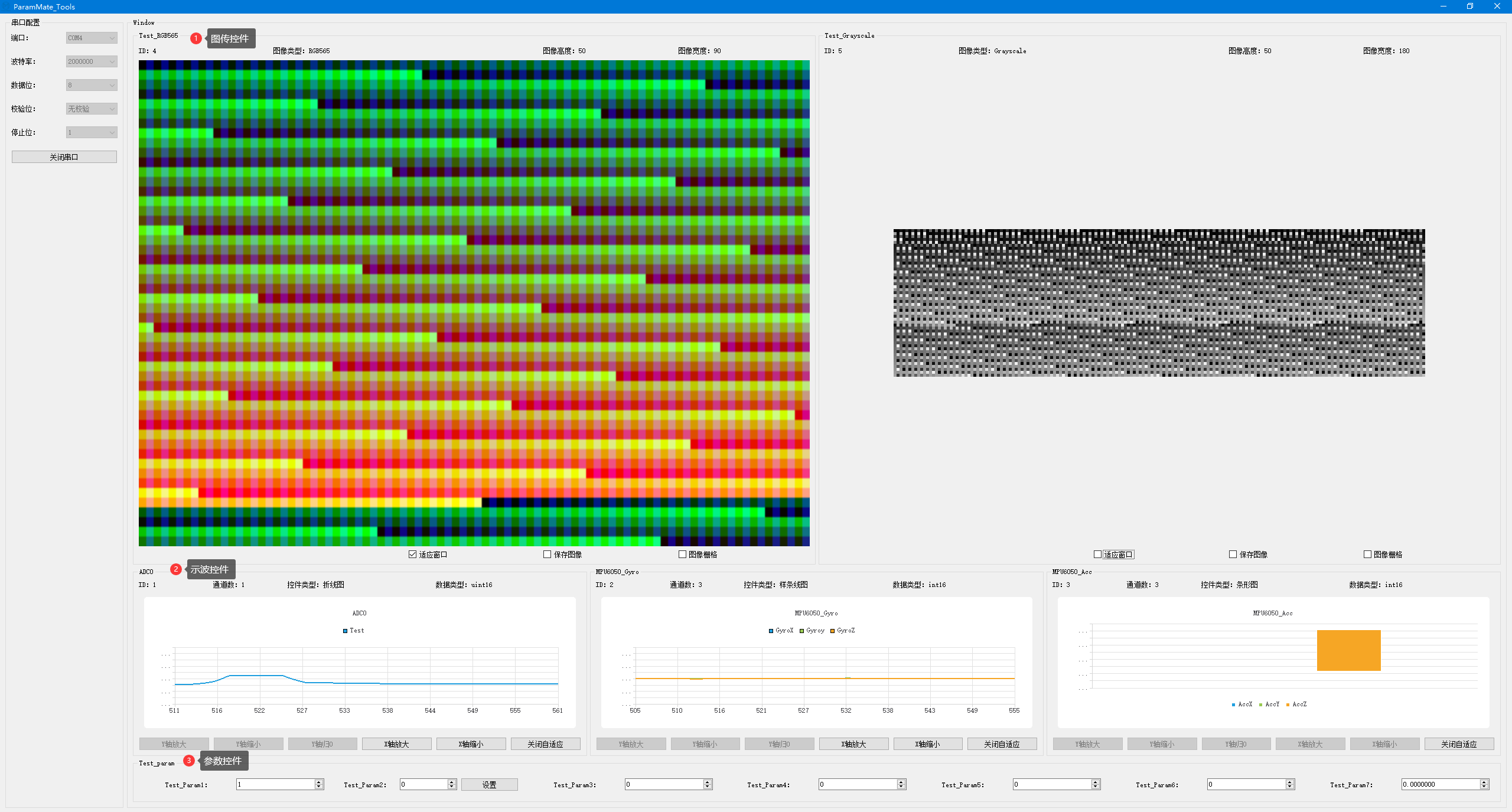
# 4.1上位机使用方法

### 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。



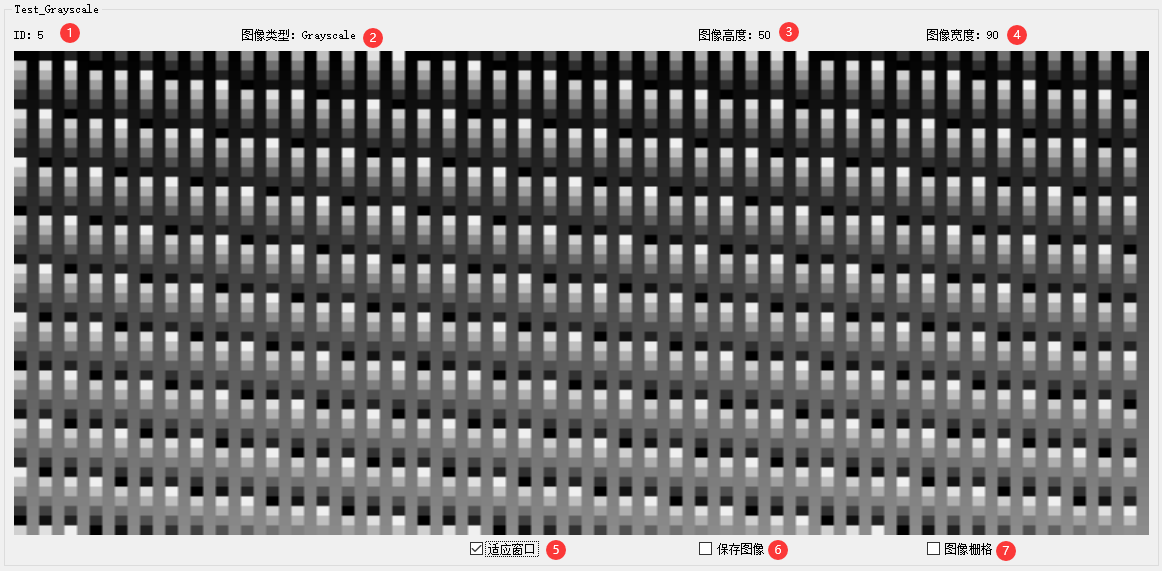
①图传控件、②示波控件、③参数控件

### 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。

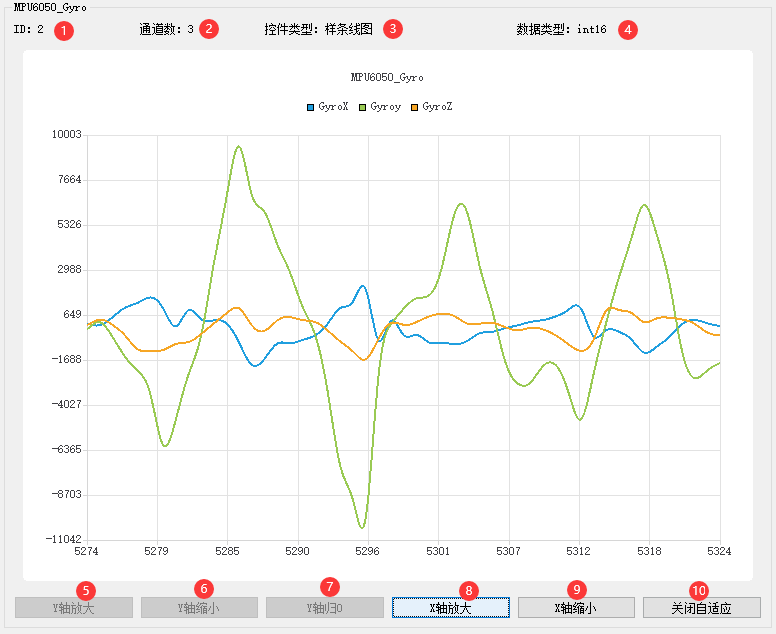
⑤适应窗口：勾选该选项则将图片铺满控件，否则将以原始比例显示。

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。**

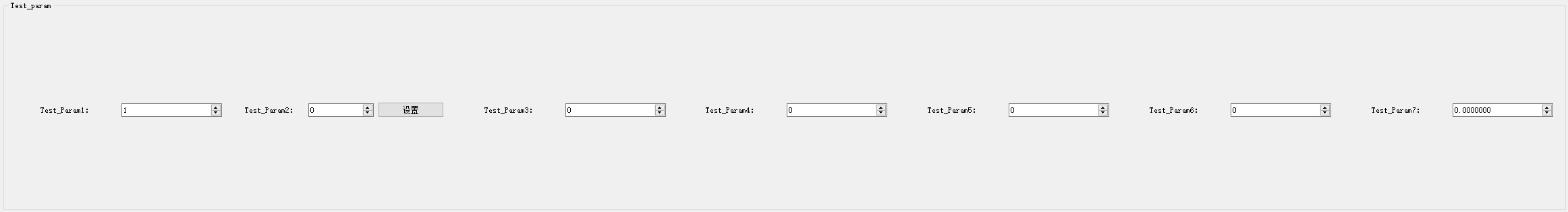
⑧X轴放大：X轴坐标放大。

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);

}