**麦步科技有限公司**

**智能手表**

**表盘及应用开发API**

**V1.4.4**

**修订记录**

| 日期 | 版本 | 描述 | 作者 |
| --- | --- | --- | --- |
| 2015-08-14 | V1.0 | 创建版本V01 | 麦步科技有限公司 |
| 2015-10-19 | V1.1 | 增加：  增加获取OS信息相关接口。  增加状态栏插件。  增加打印日志接口。  修改：  删除block图层相关接口。 | 麦步科技有限公司 |
| 2016-01-15 | V1.2 | 修改：  蓝牙状态枚举值及命名 | 麦步科技有限公司 |
| 2016-01-28 | V1.3 | 增加：  与手机第三方APP通讯接口  请求Web数据接口  请求手机数据接口 | 麦步科技有限公司 |
| 2016-04-18 | V1.4 | 增加：  新增同步手表应用信息接口及注册同步手表应用回调接口。  新增注册通讯结果回调。  新增关闭振动接口。  新增读取程序编译打包时自带的文件数据。  新增获取系统时钟，从启动开始的毫秒数。  新增获取实时速度及其他信息。  新增获取农历数据接口。  修改：  maibu\_comm\_request\_phone接口，增加参数context\_size；增加type请求类型：ERequestPhoneSelfDefine，请求自定义数据。  ErequestPhoneMusicControl，音乐控制。 | 麦步科技有限公司 |
| 2016-04-25 | V1.4.1 | 修改：  接口maibu\_get\_lunar\_calendar第一个参数修改为：date\_time \*p\_date  增加:  date\_time结构体 | 麦步科技有限公司 |
| 2016-05-13 | V1.4.2 | 修改：  修改窗口ID的返回类型为int32\_t | 麦步科技有限公司 |
| 2016-06-13 | V1.4.3 | 增加：  新增几何类型：线段集合 | 麦步科技有限公司 |
| 2016-07-14 | V1.4.4 | 增加：  请求手机数据：自定义协议增加发送用户日志接口  直线、直线集、圆、多边形结构体说明 | 麦步科技有限公司 |

[1. 概况 7](#_Toc456271775)

[2. 接口 8](#_Toc456271776)

[2.1 APP通讯 8](#_Toc456271777)

[2.1.1 与手机第三方app通讯 8](#_Toc456271778)

[maibu\_comm\_send\_msg 8](#_Toc456271779)

[maibu\_comm\_register\_msg\_callback 8](#_Toc456271780)

[2.1.2 请求Web数据 8](#_Toc456271781)

[maibu\_comm\_request\_web 8](#_Toc456271782)

[maibu\_comm\_register\_web\_callback 9](#_Toc456271783)

[2.1.3 请求手机数据 9](#_Toc456271784)

[maibu\_comm\_request\_phone 9](#_Toc456271785)

[用户日志手表发送内容 11](#_Toc456271786)

[用户日志手机应答内容 11](#_Toc456271787)

[maibu\_comm\_register\_phone\_callback 12](#_Toc456271788)

[maibu\_comm\_sync\_watchapp 12](#_Toc456271789)

[maibu\_comm\_register\_watchapp\_callback 12](#_Toc456271790)

[maibu\_comm\_register\_result\_callback 13](#_Toc456271791)

[2.1.4 类型定义 13](#_Toc456271792)

[enum ECommResult 13](#_Toc456271793)

[enum ERequestPhone 13](#_Toc456271794)

[enum ESyncWatchApp 13](#_Toc456271795)

[CBCommMsg 14](#_Toc456271796)

[CBCommWeb 14](#_Toc456271797)

[CBCommPhone 14](#_Toc456271798)

[CBCommWatchApp 14](#_Toc456271799)

[CBCommResult 15](#_Toc456271800)

[typedef stuct SGpsAltitude 15](#_Toc456271801)

[2.2 APP服务 16](#_Toc456271802)

[2.2.1 时钟 16](#_Toc456271803)

[app\_service\_timer\_subscribe 16](#_Toc456271804)

[app\_service\_timer\_unsubscribe 16](#_Toc456271805)

[app\_service\_get\_datetime 16](#_Toc456271806)

[2.2.2 马达 16](#_Toc456271807)

[maibu\_service\_vibes\_pulse 16](#_Toc456271808)

[maibu\_service\_vibes\_off 17](#_Toc456271809)

[2.2.3 系统事件 17](#_Toc456271810)

[maibu\_service\_sys\_event\_subscribe 17](#_Toc456271811)

[2.2.4 系统通知 17](#_Toc456271812)

[maibu\_service\_sys\_notify 17](#_Toc456271813)

[2.2.5 类型定义 17](#_Toc456271814)

[AppServiceTimerCallback 18](#_Toc456271815)

[AppServiceSysEventCallback 18](#_Toc456271816)

[enum SysEventType 18](#_Toc456271817)

[typedef stuct NotifyParam 18](#_Toc456271818)

[enum VibesPulseType 19](#_Toc456271819)

[typedef struct date\_time 19](#_Toc456271820)

[2.3 APP资源 19](#_Toc456271821)

[2.3.1 读取资源 19](#_Toc456271822)

[res\_get\_user\_bitmap 19](#_Toc456271823)

[maibu\_read\_user\_file 20](#_Toc456271824)

[2.4 APP存储 20](#_Toc456271825)

[2.4.1 读写数据 20](#_Toc456271826)

[app\_persist\_create 20](#_Toc456271827)

[app\_persist\_read\_data 20](#_Toc456271828)

[app\_persist\_write\_data 21](#_Toc456271829)

[app\_persist\_write\_data\_extend 21](#_Toc456271830)

[app\_persist\_delete\_data 21](#_Toc456271831)

[app\_persist\_get\_data\_size 21](#_Toc456271832)

[2.5 APP界面 22](#_Toc456271833)

[窗口API列表 22](#_Toc456271834)

[图层API列表 23](#_Toc456271835)

[2.5.1 窗口 24](#_Toc456271836)

[app\_window\_create 24](#_Toc456271837)

[app\_window\_add\_layer 24](#_Toc456271838)

[app\_window\_click\_subscribe 24](#_Toc456271839)

[app\_window\_stack\_push 25](#_Toc456271840)

[app\_window\_stack\_pop 25](#_Toc456271841)

[app\_window\_update 25](#_Toc456271842)

[app\_window\_stack\_replace\_window 25](#_Toc456271843)

[app\_window\_stack\_get\_window\_by\_id 26](#_Toc456271844)

[app\_window\_set\_up\_button 26](#_Toc456271845)

[app\_window\_set\_down\_button 26](#_Toc456271846)

[app\_window\_timer\_subscribe 26](#_Toc456271847)

[app\_window\_get\_layer\_by\_id 27](#_Toc456271848)

[app\_window\_replace\_layer 27](#_Toc456271849)

[app\_window\_set\_current\_selected\_layer 27](#_Toc456271850)

[2.5.2 图层 28](#_Toc456271851)

[app\_layer\_create\_text 28](#_Toc456271852)

[app\_layer\_set\_text\_text 28](#_Toc456271853)

[maibu\_layer\_get\_text\_text 28](#_Toc456271854)

[maibu\_layer\_get\_text\_frame 28](#_Toc456271855)

[app\_layer\_create\_bitmap 29](#_Toc456271856)

[app\_layer\_set\_bitmap\_bitmap 29](#_Toc456271857)

[maibu\_layer\_get\_bitmap\_frame 29](#_Toc456271858)

[app\_layer\_create\_scroll 29](#_Toc456271859)

[app\_layer\_set\_scroll\_text\_key 30](#_Toc456271860)

[app\_layer\_create\_geometry 30](#_Toc456271861)

[app\_layer\_create\_multi\_menu 30](#_Toc456271862)

[app\_layer\_add\_multi\_menu\_item 30](#_Toc456271863)

[app\_layer\_add\_multi\_menu\_element 31](#_Toc456271864)

[app\_layer\_set\_multi\_menu\_selected 31](#_Toc456271865)

[app\_layer\_get\_multi\_menu\_selected 31](#_Toc456271866)

[app\_layer\_set\_multi\_menu\_item\_text 31](#_Toc456271867)

[app\_layer\_set\_multi\_menu\_item\_bitmap 32](#_Toc456271868)

[app\_layer\_set\_bg\_color 32](#_Toc456271869)

[app\_layer\_set\_selected\_status 32](#_Toc456271870)

[maibu\_layer\_set\_visible\_status 33](#_Toc456271871)

[2.5.3 插件 33](#_Toc456271872)

[app\_plug\_status\_bar\_create 33](#_Toc456271873)

[app\_plug\_status\_bar\_add\_time 33](#_Toc456271874)

[app\_plug\_status\_bar\_add\_battery 33](#_Toc456271875)

[app\_plug\_status\_bar\_add\_ble 34](#_Toc456271876)

[2.5.4 类型定义 34](#_Toc456271877)

[enum LayerType 34](#_Toc456271878)

[struct LayerBitmap 34](#_Toc456271879)

[struct LayerText 35](#_Toc456271880)

[struct LayerScroll 35](#_Toc456271881)

[struct LayerMultiMenu 35](#_Toc456271882)

[struct MultiMenuItem 35](#_Toc456271883)

[struct Element 36](#_Toc456271884)

[struct LayerGeometry 36](#_Toc456271885)

[struct Geometry 36](#_Toc456271886)

[enum GeometryType 36](#_Toc456271887)

[struct Click 37](#_Toc456271888)

[enum ButtonId 37](#_Toc456271889)

[ClickCallback 37](#_Toc456271890)

[struct GRect 37](#_Toc456271891)

[struct GPoint 38](#_Toc456271892)

[struct GSize 38](#_Toc456271893)

[struct GBitmap 38](#_Toc456271894)

[enum GAlign 38](#_Toc456271895)

[enum GColor 39](#_Toc456271896)

[struct GPathInfo 39](#_Toc456271897)

[enum FillType 39](#_Toc456271898)

[struct GFont 39](#_Toc456271899)

[AnimationType 40](#_Toc456271900)

[struct Line 40](#_Toc456271901)

[struct LineSet 40](#_Toc456271902)

[struct Circle 40](#_Toc456271903)

[struct Polygon 41](#_Toc456271904)

[2.6 OS系统信息 41](#_Toc456271905)

[2.6.1 接口 41](#_Toc456271906)

[maibu\_get\_user\_account 41](#_Toc456271907)

[maibu\_get\_os\_version 41](#_Toc456271908)

[maibu\_get\_watch\_id 41](#_Toc456271909)

[maibu\_get\_bootloader\_version 42](#_Toc456271910)

[maibu\_get\_sport\_data 42](#_Toc456271911)

[maibu\_get\_ accel\_data 42](#_Toc456271912)

[maibu\_get\_altitude 42](#_Toc456271913)

[maibu\_get\_pressure 43](#_Toc456271914)

[maibu\_get\_temperature 43](#_Toc456271915)

[maibu\_get\_ble\_status 43](#_Toc456271916)

[maibu\_get\_ble\_rssi 43](#_Toc456271917)

[maibu\_get\_battery\_percent 44](#_Toc456271918)

[maibu\_get\_system\_tick 44](#_Toc456271919)

[maibu\_get\_speed\_info 44](#_Toc456271920)

[maibu\_get\_lunar\_calendar 44](#_Toc456271921)

[maibu\_get\_phone\_type 45](#_Toc456271922)

[2.6.2 类型定义 45](#_Toc456271923)

[typedef struct SportData 45](#_Toc456271924)

[enum BLEStatus 45](#_Toc456271925)

[typedef struct SpeedInfo 46](#_Toc456271926)

[typedef struct SLunarData 46](#_Toc456271927)

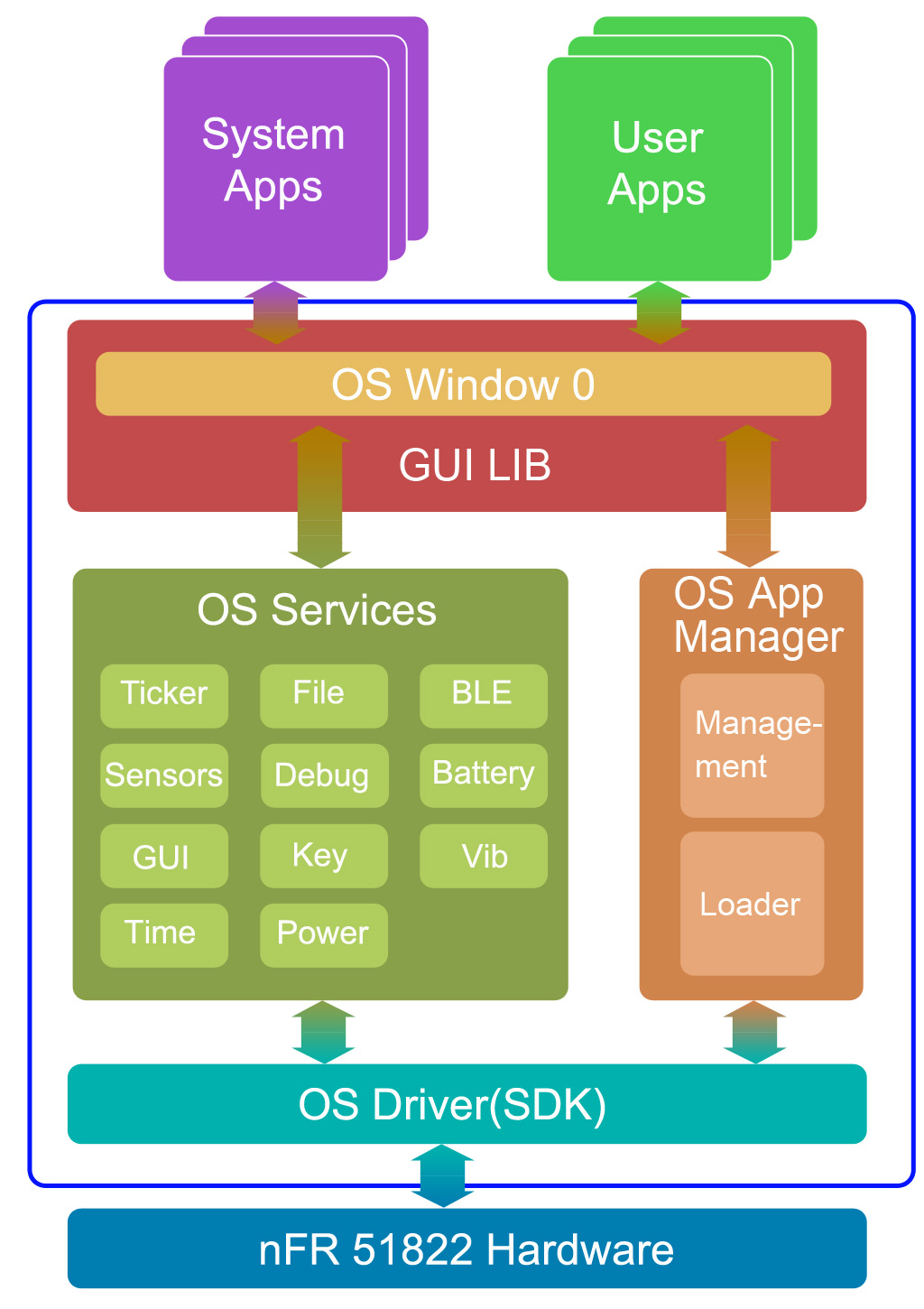
[enum PhoneType 46](#_Toc456271928)

[2.7 APP日志 47](#_Toc456271929)

[maibu\_clear\_log 47](#_Toc456271930)

[maibu\_print\_log 47](#_Toc456271931)

# 概况



说明：

麦步手表因为硬件的限制，无法做到像大型的系统一样资源基本无限制的使用，所以在编写APP的代码时请注意：

请尽量少使用全局变量，同时不要使用非静态的全局变量，系统分配给APP的内存大小为500字节，请一定注意内存的使用。

# 接口

## APP通讯

### 与手机第三方app通讯

#### maibu\_comm\_send\_msg

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_comm\_send\_msg(const char \*link\_id, const uint8\_t \*buff, uint16\_t size) |
| **功能说明** | 发送数据到手机第三方APP |
| **参数说明** | link\_id 手机第三方APP与手表APP通讯的link\_id，字符串格式，8字节  buff 数据缓存  size 数据长度 |
| **返回值** | 通讯ID，一次通讯唯一标识，表示已通讯，但不确保一定成功，可以使用接口maibu\_comm\_register\_result\_callback注册通讯结果回调，如果回调返回的通讯结果为成功，则表示上一次通讯成功，如果返回失败，用户可以在该回调中选择继续通讯。  如果通讯ID为0，则表示该应用之前调用过该接口，即上一次该类型的数据通讯还没有完成，发送失败，且系统不会回调用户注册的通讯结果接口。 |
| **参考** |  |
| **其他** | size的长度限制为200 |

#### maibu\_comm\_register\_msg\_callback

|  |  |
| --- | --- |
| **接口名** | void maibu\_comm\_register\_msg\_callback(CB[CommMsg](file:///C:\Users\hzy96\Desktop\Doc\MB_WATCH_SDK_API.doc#AppCommRecvCallback#AppCommRecvCallback) recv\_callback) |
| **功能说明** | 注册接受来自手机第三方APP数据的回调 |
| **参数说明** | recv\_callback 回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 请求Web数据

#### maibu\_comm\_request\_web

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_comm\_request\_web(const char \*url, const char \*filter\_param, int32\_t sync\_time) |
| **功能说明** | 请求WEB数据 |
| **参数说明** | url 请求链接  filter\_param 过滤参数，表示只接受过滤参数中key所对应的value值。不同的key使用英文逗号隔开，不需要过滤则设置为NULL。例如：过滤参数为“name,age,sex”三个key值。注意只返回第一次key对应的value值。  sync\_time 同步间隔时间，如果为0，则表示只请求一次，如果大于0，则表示间隔sync\_time秒请求一次，返回值与上一次不同时，系统会回调comm\_register\_web\_callback。 |
| **返回值** | 通讯ID，一次通讯唯一标识，表示已通讯，但不确保一定成功，可以使用接口maibu\_comm\_register\_result\_callback注册通讯结果回调，如果回调返回的通讯结果为成功，则表示上一次通讯成功，如果返回失败，用户可以在该回调中选择继续通讯。  如果通讯ID为0，则表示该应用之前调用过该接口，即上一次该类型的数据通讯还没有完成，发送失败，且系统不会回调用户注册的通讯结果接口。 |
| **参考** |  |
| **其他** | 注意url及filter\_param总长度不能超过200字节。sync\_time实际最小有效值为5秒。 |

#### maibu\_comm\_register\_web\_callback

|  |  |
| --- | --- |
| **接口名** | void maibu\_comm\_register\_web\_callback(CBCommWeb web\_callback) |
| **功能说明** | 注册收到来自WEB数据的回调 |
| **参数说明** | web\_callback回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 请求手机数据

#### maibu\_comm\_request\_phone

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_comm\_request\_phone(enum ERequest[Phone](#_enum_CommRequest) type, uint8\_t \*context, uint16\_t context\_size) |
| **功能说明** | 请求手机数据或控制手机某些功能 |
| **参数说明** | type 请求类型, context 根据不同请求类型，参数不同。  type=ERequestPhoneGPSAltitude， 请求GPS及海拔数据， context为NULL，context\_size为0  type=ERequestPhoneMusicControl， 发送音乐控制功能，例如切换歌曲调整音量  type=ERequestPhoneSelfDefine, 请求自定义数据，context为自定义命令  **音乐控制命令：**  数据长度1字节，  数据内容：   |  |  | | --- | --- | | 下一首 | 0x01 | | 上一首 | 0x02 | | 暂停 | 0x03 | | 调大音量 | 0x04 | | 调小音量 | 0x05 | | 播放进度 | 0x06 |   例如：切换到上一首歌，Context\_size = 1，Context: 0x02  Void send\_music\_control(void)  {  uint8\_t cmd = 0x02;  maibu\_comm\_request\_phone(ERequestPhoneMusicControl, &cmd, 1);  }  手机APP在切换歌曲后会下发音乐信息，包含歌手名、歌曲名、进度等。下发的回调  由maibu\_comm\_register\_phone\_callback()注册，  例如：  先注册回调maibu\_comm\_register\_phone\_callback(music\_data\_receive\_callback);  在music\_data\_receive\_callback回调中获取的数据格式如下：  **music\_data:**   |  |  |  | | --- | --- | --- | | 字段 | 占用字节数 | 描述 | | progress | 1 | 播放进度(0~100)  progress为必填选项 | | singer\_name\_size | 1 | 歌手名字长度  如果singer\_name\_size为0x00，则之后字段不存在 | | singer\_name\_value | 可变 | 歌手名字值 | | song\_name\_size | 1 | 歌曲名字长度 | | song\_name\_value | 可变 | 歌曲名字值 | | album\_name\_size | 1 | 专辑名字长度 | | album\_name\_value | 可变 | 专辑名字值 |   自定义协议基本格式   |  |  | | --- | --- | | 协议ID  （两个字节） | 协议具体数据内容或格式 |   协议ID列表   |  |  |  |  | | --- | --- | --- | --- | | 协议ID名 | ID值 | 方向 | 描述 | | get\_city\_by\_gps | 1 | 手表->手机  手机->手表 | 根据GPS获取城市 | | user\_log | 3 | 手表->手机  手机->手表 | 处理用户日志数据，可以在手机客户端查看日志。  iOS手机客户端：1.6.6及以上版本支持。  Android手机客户端：1.5.9及以上版本支持。 |   根据GPS获取城市协议手表应用请求数据内容   |  |  |  | | --- | --- | --- | | 字段 | 占用字节数 | 描述 | | longitude\_value | 16 | 经度值（字符串） | | latitude\_value | 16 | 纬度值（字符串），如果经度纬度都为空，则返回当前手机所在城市。 | |  |  |  |     根据GPS获取城市协议手机应答数据内容   |  |  |  | | --- | --- | --- | | 字段 | 占用字节数 | 描述 | | city\_id | 2 | 城市ID（整数） | | city\_name | 20 | 城市名（字符串） | | city\_pinyin | 20 | 城市拼音（字符串） | | city\_distance | 4 | 当前位置离城市距离（整数，单位km） |   例如：  请求当前手机所在城市时，context内容为如下，context\_size为34  0x01,0x00,  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00   用户日志手表发送内容  |  |  |  | | --- | --- | --- | | 字段 | 占用字节数 | 描述 | | display | 1 | 手机客户端显示日志内容的格式：   1. 字符串显示，即直接输出到日志模块。 2. 10进制显示，即每个字节转换为10进制数据显示，用逗号隔开。 3. 16进制显示，即每个字节转换为16进制数据，用逗号隔开。 | | data\_size | 2 | 数据长度 | | dat | 可变 | 数据内容 | |  |  |  |  用户日志手机应答内容  |  |  |  | | --- | --- | --- | | 字段 | 占用字节数 | 描述 | | status | 1 | 0 成功 | |  |  |  | |
| **返回值** | 通讯ID，一次通讯唯一标识，表示已通讯，但不确保一定成功，可以使用接口maibu\_comm\_register\_result\_callback注册通讯结果回调，如果回调返回的通讯结果为成功，则表示上一次通讯成功，如果返回失败，用户可以在该回调中选择继续通讯。  如果通讯ID为0，则表示该应用之前调用过该接口，即上一次该类型的数据通讯还没有完成，发送失败，且系统不会回调用户注册的通讯结果接口。 |
| **参考** |  |
| **其他** | 请求手机数据或控制手机某些功能 |

#### maibu\_comm\_register\_phone\_callback

|  |  |
| --- | --- |
| **接口名** | void maibu\_comm\_register\_phone\_callback(CBCommPhone phone\_callback) |
| **功能说明** | 注册请求或接受手机数据或控制手机某些功能的回调 |
| **参数说明** | phone\_callback 回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_comm\_sync\_watchapp

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_comm\_sync\_watchapp(enum E[SyncWatchApp](#CommAppSync) type, uint8\_t \*context, uint16\_t context\_size) |
| **功能说明** | 同步手表应用信息 |
| **参数说明** | type 同步类型, context 根据不同类型，参数不同。  type=ESyncWatchAppUpdateParam， 请求或上传手表应用参数  context为当前手表上某应用的参数，格式为字符串。  context\_size为参数大小。  如果context为空或者上传的参数与手机客户端参数不一致时，手机客户端会发送最新的参数至手表应用，手表应用需要使用接口maibu\_comm\_register\_watchapp\_callback注册接受手表应用同步的回调。 |
| **返回值** | 0 通讯失败  >0 通讯ID，一次通讯唯一标识，表示已通讯，但不确保一定成功，可以使用接口maibu\_comm\_register\_result\_callback注册发送结果回调，如果回调返回的通讯结果为成功，则表示上一次通讯成功，如果返回失败，用户可以在该回调中选择继续通讯。 |
| **参考** |  |
| **其他** | context\_size不能大于200字节 |

#### maibu\_comm\_register\_watchapp\_callback

|  |  |
| --- | --- |
| **接口名** | void maibu\_comm\_register\_watchapp\_callback(CBCommWatchApp watchapp\_callback) |
| **功能说明** | 注册同步手表应用的回调 |
| **参数说明** | watchapp\_callback 回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_comm\_register\_result\_callback

|  |  |
| --- | --- |
| **接口名** | void maibu\_comm\_register\_result\_callback(CB[CommResult](#_AppCommStatusCallback)  result\_callback) |
| **功能说明** | 注册通讯结果回调 |
| **参数说明** | result\_callback 回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 类型定义

#### enum ECommResult

|  |  |
| --- | --- |
| **接口名** | enum ECommResult |
| **功能说明** | 通讯结果 |
| **参数说明** | ECommResultSuccess 成功  ECommResultFail失败 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum ERequestPhone

|  |  |
| --- | --- |
| **接口名** | enum ERequestPhone |
| **功能说明** | 请求手机上数据类型 |
| **参数说明** | ERequestPhoneGPSAltitude 请求手机上GPS及海拔  ERequestPhoneSelfDefine，请求手机上自定义数据 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum ESyncWatchApp

|  |  |
| --- | --- |
| **接口名** | enum ESyncWatchApp |
| **功能说明** | 同步手表应用命令类型 |
| **参数说明** | ESyncWatchAppUpdateParam 手机下发应用配置参数到手表应用  ESyncWatchAppGetParam 手机请求手表应用的配置参数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### CB[CommMsg](#AppCommRecvCallback)

|  |  |
| --- | --- |
| **接口名** | typedef void(\*CBCommMsg )( const char \*link\_id, const uint8\_t \*buff, uint8\_t size) |
| **功能说明** | 接受来自手机第三方APP的数据 |
| **参数说明** | link\_id 手机APP的link\_id  buff 数据缓存  size 数据长度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### CBCommWeb

|  |  |
| --- | --- |
| **接口名** | typedef void(\*CBCommWeb )( const uint8\_t \*buff, uint16\_t size) |
| **功能说明** | 收到来自WEB的数据 |
| **参数说明** | buff 如果使用了filter\_param，则返回值是过滤后的WEB返回值，可以分别使用  maibu\_get\_json\_int及maibu\_get\_json\_str获取对应的键值。  size 数据缓存大小 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### CBCommPhone

|  |  |
| --- | --- |
| **接口名** | typedef void(\*CBCommPhone )( enum E[RequestPhone](#CommRequest) type, uint8\_t \*context, uint16\_t context\_size) |
| **功能说明** | 请求手机数据或控制手机某些功能 |
| **参数说明** | type 请求类型, context 根据不同请求类型，返回值不同。  type = ERequestPhoneGPSAltitude，则context需要转换为S[GpsAltitude](file:///C:\Users\hzy96\Desktop\Doc\MB_WATCH_SDK_API.doc#GpsAltitude#GpsAltitude)结构。  type=ERequestPhoneSelfDefine，则context为自定义数据格式。 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### CBCommWatchApp

|  |  |
| --- | --- |
| **接口名** | typedef void(\*CBCommWatchApp )( enum E[SyncWatchApp](#CommRequest) type, uint8\_t \*context, uint16\_t context\_size) |
| **功能说明** | 同步手表应用信息回调 |
| **参数说明** | type 同步类型, context 根据不同请求类型，返回值不同。  type = ESyncWatchAppUpdateParam ，则context为手机客户端下发的手表应用或表盘的配置参数，格式为字符串。应用或表盘需要完全保存context值。  type = ESyncWatchAppGetParam，则表明手机客户端需要获取手表上该应用或表盘  的当前配置信息，应用及表盘调用maibu\_comm\_sync\_watchapp返回配置信息。 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### CBCommResult

|  |  |
| --- | --- |
| **接口名** | typedef void(\*CBCommResult )(enum E[CommResult](#AppCommStatus) result, uint32\_t comm\_id, void \*context,) |
| **功能说明** | 通讯结果回调函数 |
| **参数说明** | result 通讯结果  comm\_id 通讯ID  context 保留参数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### typedef stuct SGpsAltitude

|  |  |
| --- | --- |
| **接口名** | typedef stuct SGpsAltitude |
| **功能说明** | GPS及海拔数据结构体 |
| **参数说明** | char lon[16] 经度  char lat[16] 纬度  int16\_t h\_accuracy 水平误差，单位米  int32\_t altitude 海拔高度， 单位米  int16\_t v\_accuracy 海拔误差，单位cm |
| **返回值** |  |
| **参考** |  |

## APP服务

### 时钟

#### app\_service\_timer\_subscribe

|  |  |
| --- | --- |
| **接口名** | uint8\_t app\_service\_timer\_subscribe(uint32\_t millis, [AppServiceTimerCallback](#AppServiceTimerCallback) timer\_callback , void\* context) |
| **功能说明** | 使用定时服务 |
| **参数说明** | millis 定时器任务执行间隔，单位毫秒  timer\_callback 定时回调函数  context 可变参数(在回调函数被调用时，context会作为参数传入) |
| **返回值** | 该应用申请的定时器ID |
| **参考** |  |
| **其他** |  |

#### app\_service\_timer\_unsubscribe

|  |  |
| --- | --- |
| **接口名** | void app\_service\_timer\_unsubscribe(uint8\_t timer\_id) |
| **功能说明** | 取消使用定时服务 |
| **参数说明** | timer\_id 定时器ID |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_service\_get\_datetime

|  |  |
| --- | --- |
| **接口名** | bool app\_service\_get\_datetime(date\_time\_t t) |
| **功能说明** | 获取系统时间 |
| **参数说明** | t 年月日时分秒 |
| **返回值** | 是否获取成功 |
| **参考** |  |
| **其他** | 星期从周日开始，对应序号为0 |

### 马达

#### maibu\_service\_vibes\_pulse

|  |  |
| --- | --- |
| **接口名** | void maibu\_service\_vibes\_pulse(enum VibesPulseType pulse\_type, uint8\_t times) |
| **功能说明** | 使用马达震动 |
| **参数说明** | type 马达震动类型  times 马达震动次数 |
| **返回值** |  |
| **参考** |  |
| **其他** | 当震动类型为VibesPulseTypeShort 时，times无需设定  受勿扰模式影响 |

#### maibu\_service\_vibes\_off

|  |  |
| --- | --- |
| **接口名** | void maibu\_service\_vibes\_off(void) |
| **功能说明** | 关闭振动 |
| **参数说明** |  |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 系统事件

#### maibu\_service\_sys\_event\_subscribe

|  |  |
| --- | --- |
| **接口名** | void maibu\_service\_sys\_event\_subscribe ([AppServiceSysEventCallback](#AppServiceSysEventCallback) sys\_event\_callback) |
| **功能说明** | 注册系统事件回调函数，如果有系统事件发生，回调用户函数并传递系统事件及参数 |
| **参数说明** |  |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 系统通知

#### maibu\_service\_sys\_notify

|  |  |
| --- | --- |
| **接口名** | void maibu\_service\_sys\_notify ([NotifyParam](#NotifyParam)  \*param) |
| **功能说明** | 供用户应用在后台运行时调用系统通知界面，显示应用提供的相关信息。 |
| **参数说明** | param 需要通知的内容参数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 类型定义

#### [AppServiceTime](#AppServiceTimerCallback)[rCallb](#AppServiceTimerCallback)[ack](#AppServiceTimerCallback)

|  |  |
| --- | --- |
| **接口名** | typedef void(\*[AppServiceTime](#AppServiceTimerCallback)[rCallback](#AppServiceTimerCallback))(date\_time\_t tick\_time, uint32\_t millis，void\* context ) |
| **功能说明** | 使用定时服务回调函数 |
| **参数说明** | tick\_time 当前时间结构  millis 系统启动的时间 单位毫秒  context 用户注册时写入的参数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### AppServiceSysEventCallback

|  |  |
| --- | --- |
| **接口名** | typedef void(\*AppServiceSysEventCallback)(enum [SysEventType](#_enum_SysEventType) type, void \*param ) |
| **功能说明** | 使用定时服务回调函数 |
| **参数说明** | type 系统事件类型  param事件参数，需要转换为对应结构体使用，如果是时间更改，则param为当前时间结构体指针struct date\_time |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum SysEventType

|  |  |
| --- | --- |
| **接口名** | enum SysEventType |
| **功能说明** | 系统事件类型 |
| **参数说明** | SysEventTypeTimeChange 时间更改 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### typedef stuct NotifyParam

|  |  |
| --- | --- |
| **接口名** | typedef stuct NotifyParam |
| **功能说明** | 通知参数 |
| **参数说明** | GBitmap bmp 通知界面中心显示的图片  char main\_title[20] 主标题  char sub\_title[30] 副标题  enum VibesPulseType pulse\_type 震动类型  int8\_t pulse\_time; //震动次数  void \*context 保留 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum VibesPulseType

|  |  |
| --- | --- |
| **接口名** | enum VibesPulseType |
| **功能说明** | 马达震动类型 |
| **参数说明** | VibesPulseTypeShort 短震动  VibesPulseTypeMiddle 中震动  VibesPulseTypeLong 长震动 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### typedef struct date\_time

|  |  |
| --- | --- |
| **接口名** | typedef struct date\_time |
| **功能说明** | 时间结构体 |
| **参数说明** | unsigned short int year; /\* [1901,-] 年\*/  unsigned char mon; /\* [1,12] 月\*/  unsigned char mday; /\* [1,31] 日 \*/  unsigned char wday; /\* [0,6] 星期\*/  unsigned char hour; /\* [0,23] 时\*/  unsigned char min; /\* [0,59] 分\*/  unsigned char sec; /\* [0,59] 秒\*/ |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

## APP资源

说明：使用图片时，请使用bmp格式位深度为1的图片。所有图片点阵（不是图片文件）的大小总和请不要超过200Kbytes。

### 读取资源

#### res\_get\_user\_bitmap

|  |  |
| --- | --- |
| **接口名** | uint8\_t res\_get\_user\_bitmap(uint32\_t key, GBitmap \*bitmap); |
| **功能说明** | 读取用户图片资源 |
| **参数说明** | key 资源ID  bitmap 图片结构体指针 |
| **返回值** | 实际读取的资源大小 |
| **参考** |  |
| **其他** |  |

#### maibu\_read\_user\_file

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_read\_user\_file(uint32\_t key, uint32\_t offset, uint8\_t \*buff, uint32\_t size) |
| **功能说明** | 读取程序编译打包时自带的文件数据 |
| **参数说明** | key 数据标识  offset 偏移量  buff 缓存  size 想要读取的数据大小 |
| **返回值** | 实际读取的数据大小 |
| **参考** |  |
| **其他** |  |

## APP存储

说明：

1. 读写数据时应该先create，如果之前已经create，则create立即返回成功。

2. 文件的KEY需要编写代码时自定义。

3. 文件小于4Kbytes的将至少占用4Kbytes的空间，外置存储中分配给所有应用使用的文件区域总大小是512Kbytes，建议尽量把零碎的数据存放到一个文件中。

### 读写数据

#### app\_persist\_create

|  |  |
| --- | --- |
| **接口名** | bool app\_persist\_create(const uint32\_t key , uint32\_t max\_file\_size) |
| **功能说明** | 以key创建一个大小为max\_file\_size的文件 |
| **参数说明** | key 数据标识  max\_file\_size 该key可以存储的大小 |
| **返回值** | 成功或者失败 |
| **参考** |  |
| **其他** |  |

#### app\_persist\_read\_data

|  |  |
| --- | --- |
| **接口名** | uint32\_t app\_persist\_read\_data(const uint32\_t key, uint32\_t offset, void \*buff, const uint32\_t size) |
| **功能说明** | 读取存储区的数据 |
| **参数说明** | key 数据标识  offset 读取的偏移量  buff 存储数据缓存  size 缓存大小 |
| **返回值** | 实际读取的数据大小 |
| **参考** |  |
| **其他** |  |

#### app\_persist\_write\_data

|  |  |
| --- | --- |
| **接口名** | uint32\_t app\_persist\_write\_data(const uint32\_t key, const void\* buff, const uint32\_t size) |
| **功能说明** | 写入数据到存储区 |
| **参数说明** | key 数据标识  buff 数据缓存  size 数据大小 |
| **返回值** | 实际写入的数据大小 |
| **参考** |  |
| **其他** |  |

#### app\_persist\_write\_data\_extend

|  |  |
| --- | --- |
| **接口名** | uint32\_t app\_persist\_write\_data\_extend(const uint32\_t key, const void\* buff, const uint32\_t size) |
| **功能说明** | 清除之前的数据，从头写入数据到存储区 |
| **参数说明** | key 数据标识  buff 数据缓存  size 数据大小 |
| **返回值** | 实际写入的数据大小 |
| **参考** |  |
| **其他** |  |

#### app\_persist\_delete\_data

|  |  |
| --- | --- |
| **接口名** | uint32\_t app\_persist\_delete\_data (const uint32\_t key) |
| **功能说明** | 删除存储区的数据 |
| **参数说明** | key 数据标识 |
| **返回值** | 删除数据的大小 |
| **参考** |  |
| **其他** |  |

#### app\_persist\_get\_data\_size

|  |  |
| --- | --- |
| **接口名** | uint32\_t app\_persist\_get\_data\_size(const uint32\_t key) |
| **功能说明** | 得到某个key当前的大小 |
| **参数说明** | key 数据标识 |
| **返回值** | 该key存储的大小 |
| **参考** |  |
| **其他** |  |

## APP界面



说明：

1. 一个应用最多创建3个窗口，有多层窗口需要可以使用窗口替换接口。
2. 一个窗口最多创建20个图层。不要创建过多。
3. 尽量少使用几何图层来实现复杂的图形，建议用位图图层代替。
4. 窗口中某图层的显示顺序和该图层添加到窗口中的顺序有关。
5. 如果在窗口中同一块区域有两个图层重叠，后添加的图层覆盖先添加的图层。
6. 窗口只能在前台运行。
7. 窗口及其图层的刷新频率建议不要过高，目前会影响运动数据。

### 窗口API列表

|  |
| --- |
| **Window \***[**app\_window\_create**](#maibu_window_create)**(void);**  创建窗口 |
| **int8\_t** [**app\_window\_add\_layer**](#maibu_window_add_layer)**(Window \*window, Layer \*layer);**  添加图层到窗口 |
| **void** [**app\_window\_click\_subscribe**](#maibu_window_click_subscribe)**(Window \*window, enum ButtonId button\_id, ClickCallback click\_callback);**  注册窗口按键回调事件 |
| **Int32\_t** [**app\_window\_stack\_push**](#maibu_window_stack_push)**(Window \*window);**  窗口放入窗口栈显示 |
| **void** [**app\_window\_stack\_pop**](#maibu_window_stack_pop)**(Window \*window);**  从窗口栈pop出窗口 |
| **void** [**app\_window\_update**](#maibu_window_update)**(Window \*window);**  更新指定窗口, 只有该窗口是窗口栈top窗口，才更新 |
| **int32\_t** [**app\_window\_stack\_replace\_window**](#maibu_window_stack_replace_window)**(Window \*old\_window, Window \*new\_window);**  替换窗口 |
| **Window \***[**app\_window\_stack\_get\_windo****w\_by\_id**](#maibu_window_stack_get_window_by_id)**(int32\_t id);**  根据窗口ID获取窗口栈中窗口句柄 |
| **void** [**app\_window\_set\_up\_button**](#maibu_window_set_up_button)**(Window \*window);**  设置向上按键,当需要向上选择图层时使用 |
| **void** [**app\_window\_set\_down\_button**](#maibu_window_set_down_button)**(Window \*window);**  设置向下按键, 当需要向下选择图层时使用 |
| **int8\_t** [**app\_window\_timer\_subscribe**](#maibu_window_timer_subscribe)**(Window \*window, uint32\_t millis, AppServiceTimerCallback timer\_callback, void \*context);**  窗口定时器 |
| **Layer \***[**app\_window\_get\_layer\_by\_id**](#maibu_window_get_layer_by_id)**(Window \*window, int8\_t id);**  根据图层ID获取某窗口中图层句柄 |
| **int8\_t**  [**app\_window\_replace\_layer**](#maibu_window_replace_layer)**(Window \*window, Layer \*old\_layer, Layer \*new\_layer);**  替换图层 |
| **void** [**app\_window\_set\_current\_selected\_layer**](#maibu_window_set_current_selected_layer)**(Window \*window, Layer \*layer);**  设置窗口中某图层被选择 |
|  |
|  |

### 图层API列表

|  |
| --- |
| **P\_Layer**  [**app\_layer\_create\_text**](#maibu_layer_create_text)**(LayerText \*layer\_text);**  创建文本图层 |
| **void** [**app\_layer\_set\_text\_text**](#maibu_layer_set_text_text)**(Layer \*layer, char \*text);**  设置文本图层文本内容 |
| **P\_Layer**  [**app\_layer\_create\_bitmap**](#maibu_layer_create_bitmap)**(LayerBitmap \*layer\_bitmap);**  创建位图图层 |
| **void** [**app\_layer\_set\_bitmap\_bitmap**](#maibu_layer_set_bitmap_bitmap)**(Layer \*layer, GBitmap \* bitmap);**  设置位图图层的位图 |
| **P\_Layer**  [**app\_layer\_create\_scroll**](#maibu_layer_create_scroll)**(LayerScroll \*layer\_scroll);**  创建滚动图层 |
| **void** [**app\_layer\_set\_scroll\_text\_key**](#maibu_layer_set_scroll_text_key)**(Layer \*layer, uint8\_t text\_key);**  设置滚动图层文本地址 |
| **P\_Layer** [**app\_layer\_create\_geometry**](#maibu_layer_create_geometry)**(LayerGeometry \*layer\_geometry);**  创建几何图层 |
| **P\_Layer** [**app\_layer\_create\_multi\_menu**](#maibu_layer_create_multi_menu)**(LayerMultiMenu \*layer\_multi\_menu);**  创建复杂菜单图层 |
| **int8\_t** [**app\_layer\_add\_multi\_menu\_item**](#maibu_layer_add_multi_menu_item)**(LayerMultiMenu \*lmm, MultiMenuItem \*mmi);**  添加菜单项 |
| **int8\_t** [**app\_layer\_add\_multi\_menu\_element**](#maibu_layer_add_multi_menu_element)**(MultiMenuItem \*mmi, Element \*e);**  添加某菜单项中元素 |
| **void** [**app\_layer\_set\_multi\_menu****\_selected**](#maibu_layer_set_multi_menu_selected)**(Layer \*layer, uint8\_t selected);**  设置菜单图层被选择项 |
| **int8\_t** [**app\_layer\_get\_multi\_menu\_selected**](#maibu_layer_get_multi_menu_selected)**(Layer \*layer);**  获取菜单图层被选择项 |
| **void** [**app\_layer\_set\_multi\_menu\_item\_text**](#maibu_layer_set_multi_menu_item_text)**(Layer \*layer, int8\_t item\_id, int8\_t element\_id, const char \*text);**  设置菜单图层菜单项文本 |
| **void** [**app\_layer\_set\_multi\_menu\_item\_bitmap**](#maibu_layer_set_multi_menu_item_bitmap)**(Layer \*layer, int8\_t item\_id, int8\_t element\_id, GBitmap \*bitmap);**  设置菜单图层菜单项图片 |
| **void** [**app\_layer\_set\_bg\_color**](#maibu_layer_set_bg_color)**(Layer \*layer, enum GColor color);**  设置背景颜色 |
| **void** [**app\_layer\_set\_selected\_status**](#maibu_layer_set_selected_status)**(Layer \*layer, uint8\_t status);**  设置图层可被选择状态 |
| **void** [**app\_layer\_set\_visible\_status**](#_app_layer_set_visible_status)**(Layer \*layer, bool status);**  设置图层可见状态 |
|  |

### 窗口

#### app\_window\_create

|  |  |
| --- | --- |
| **接口名** | [Wi](#Window)[ndow](#Window) \*app\_window\_create(void) |
| **功能说明** | 创建一个默认窗口 |
| **参数说明** |  |
| **返回值** |  |
| **参考** |  |
| **其他** | 屏幕上显示的任何数据，都必须建立在一个可见的窗口上。  只有创建了窗口，才能在窗口上添加图层。 |

#### app\_window\_add\_layer

|  |  |
| --- | --- |
| **接口名** | uint8\_t app\_window\_add\_layer([Window](#Window) \*window, [Layer](#Layer) \*layer) |
| **功能说明** | 给窗口添加图层 |
| **参数说明** | window 需要添加图层的窗口指针  layer 图层指针 |
| **返回值** | 成功：返回图层ID  失败：-1 |
| **参考** |  |
| **其他** | window上的所有显示内容，以图层为单位 |

#### app\_window\_click\_subscribe

|  |  |
| --- | --- |
| **接口名** | void app\_window\_click\_subscribe([Window](#Window) \*window, enum [ButtonId](#ButtonId) button\_id, [ClickCallback](#ClickCallback) click\_callback) |
| **功能说明** | 注册按键的回调函数 |
| **参数说明** | window 要注册按键事件的窗口指针  button\_id 按键类型ID  click\_callback 按键回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_window\_stack\_push

|  |  |
| --- | --- |
| **接口名** | Int32\_t app\_window\_stack\_push([Window](#Window) \*window) |
| **功能说明** | 把窗口放置窗口栈顶显示 |
| **参数说明** | window 需要放置的窗口 |
| **返回值** | 窗口ID |
| **参考** |  |
| **其他** | 如果窗口不属于前台运行程序，则不会显示。  如果应用在前台有定时更新界面的操作，务必保留该窗口ID，之后通过app\_window\_stack\_get\_window\_by\_id获取窗口指针，再调用其他相关接口。 |

#### app\_window\_stack\_pop

|  |  |
| --- | --- |
| **接口名** | void app\_window\_stack\_pop([Window](#Window) \*window) |
| **功能说明** | 从窗口栈pop出窗口 |
| **参数说明** | window 需要pop的窗口 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_window\_update

|  |  |
| --- | --- |
| **接口名** | void app\_window\_update(Window \*window) |
| **功能说明** | 如果window是当前窗口，则重新显示 |
| **参数说明** | window 窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** | 如果窗口不属于前台运行程序，则不会显示 |

#### app\_window\_stack\_replace\_window

|  |  |
| --- | --- |
| **接口名** | Int32\_t app\_window\_stack\_replace\_window(Window \*old\_window, Window \*new\_window) |
| **功能说明** | 使用新的窗口替换旧窗口 |
| **参数说明** | old\_window 旧窗口指针  new\_window 新窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_window\_stack\_get\_window\_by\_id

|  |  |
| --- | --- |
| **接口名** | Window \*app\_window\_stack\_get\_window\_by\_id(int32\_t id); |
| **功能说明** | 根据窗口ID获取窗口指针句柄 |
| **参数说明** | id 窗口ID |
| **返回值** | 窗口句柄  NULL |
| **参考** |  |
| **其他** |  |

#### app\_window\_set\_up\_button

|  |  |
| --- | --- |
| **接口名** | void app\_window\_set\_up\_button(Window \*window) |
| **功能说明** | 设置向上按键，当需要向上选择图层时使用，在注册的向上按键回调函数中使用。 |
| **参数说明** | window 窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_window\_set\_down\_button

|  |  |
| --- | --- |
| **接口名** | void app\_window\_set\_down\_button(Window \*window) |
| **功能说明** | 设置向下按键，当需要向下选择图层时使用，在注册的向下按键回调函数中使用。 |
| **参数说明** | window 窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_window\_timer\_subscribe

|  |  |
| --- | --- |
| **接口名** | int8\_t app\_window\_timer\_subscribe(Window \*window, uint32\_t millis, AppServiceTimerCallback timer\_callback, void \*context); |
| **功能说明** | 窗口定时器 |
| **参数说明** | window 窗口指针  mills 调用间隔  timer\_callback 回调函数  context 上下文参数 |
| **返回值** | 成功：返回定时器ID  失败：-1 |
| **参考** |  |
| **其他** | 窗口定时器在窗口销毁时，会自动注销。 |

#### app\_window\_get\_layer\_by\_id

|  |  |
| --- | --- |
| **接口名** | Layer \* app\_window\_get\_layer\_by\_id([Window](#Window) \*window, int8\_t id) |
| **功能说明** | 根据图层ID获取窗口中该图层指针 |
| **参数说明** | window 窗口指针  id 图层ID |
| **返回值** | 成功：返回图层指针  失败：NULL |
| **参考** |  |
| **其他** |  |

#### app\_window\_replace\_layer

|  |  |
| --- | --- |
| **接口名** | int8\_t app\_window\_replace\_layer([Window](#Window) \*window, Layer \*old\_layer, Layer \*new\_layer) |
| **功能说明** | 用新的图层替换窗口中某旧图层 |
| **参数说明** | window 窗口指针  old\_layer 旧图层指针  new\_layer 新图层指针 |
| **返回值** | 成功：图层ID，>=0  失败：-1 |
| **参考** |  |
| **其他** |  |

#### app\_window\_set\_current\_selected\_layer

|  |  |
| --- | --- |
| **接口名** | void app\_window\_set\_current\_selected\_layer([Window](#Window) \*window, int8\_t id) |
| **功能说明** | 设置当前窗口中被选择的图层ID |
| **参数说明** | window 窗口指针  id 图层ID |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 图层

#### app\_layer\_create\_text

|  |  |
| --- | --- |
| **接口名** | [Layer](#Layer) \*app\_layer\_create\_text(LayerText \*layer\_text) |
| **功能说明** | 创建一个文本图层 |
| **参数说明** | layer\_text 文本结构指针 |
| **返回值** | 成功：文本图层指针  失败：NULL |
| **参考** |  |
| **其他** | 文本图层只能显示一行文字 |

#### app\_layer\_set\_text\_text

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_text\_text(Layer \*layer, char \*text); |
| **功能说明** | 设置文本图层文本内容 |
| **参数说明** | layer\_text 文本图层指针  text 文本内容 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_layer\_get\_text\_text

|  |  |
| --- | --- |
| **接口名** | void maibu\_layer\_get\_text\_text(Layer \*layer, char \*text, uint16\_t text\_size); |
| **功能说明** | 设置文本图层文本内容 |
| **参数说明** | layer\_text 文本图层指针  text 存放文本内容buff  text\_size存放文本内容buff的大小 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_layer\_get\_text\_frame

|  |  |
| --- | --- |
| **接口名** | void maibu\_layer\_get\_text\_frame(Layer \*layer, GRect \*frame); |
| **功能说明** | 获取文本图层文本框架 |
| **参数说明** | layer\_text 文本图层指针  frame 框架结构体指针，保存框架值 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_create\_bitmap

|  |  |
| --- | --- |
| **接口名** | [Layer](#Layer) \*app\_layer\_create\_bitmap(LayerBitmap \*layer\_bitmap) |
| **功能说明** | 创建一个位图图层 |
| **参数说明** | layer\_bitmap 位图结构指针 |
| **返回值** | 成功：位图图层指针  失败：NULL |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_bitmap\_bitmap

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_bitmap\_bitmap(Layer \*layer, GBitmap \* bitmap); |
| **功能说明** | 设置一个位图图层显示的位图 |
| **参数说明** | layer 位图图层指针  bitmap 位图指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_layer\_get\_bitmap\_frame

|  |  |
| --- | --- |
| **接口名** | void maibu\_layer\_get\_bitmap\_frame(Layer \*layer, GRect \* frame); |
| **功能说明** | 获取一个位图图层的框架值 |
| **参数说明** | layer 位图图层指针  frame 位图框架结构体指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_create\_scroll

|  |  |
| --- | --- |
| **接口名** | P\_Layer app\_layer\_create\_scroll(LayerScroll \*layer\_scroll); |
| **功能说明** | 创建一个滚动图层 |
| **参数说明** | layer\_scroll 滚动结构指针 |
| **返回值** | 成功：滚动图层指针  失败：NULL |
| **参考** |  |
| **其他** | 滚动图层可以显示多行文字，并且具有翻页功能，最多显示5页。 |

#### app\_layer\_set\_scroll\_text\_key

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_scroll\_text\_key(Layer \*layer, uint8\_t text\_key) |
| **功能说明** | 设置滚动图层显示的文本KEY |
| **参数说明** | layer 图层指针  text\_key 文本key（通过存储接口把文本内容放在该key中） |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_create\_geometry

|  |  |
| --- | --- |
| **接口名** | [Layer](#Layer) \*app\_layer\_create\_geometry(LayerGeometry \*layer\_geometry) |
| **功能说明** | 创建一个几何图层 |
| **参数说明** | layer\_geometry 几何结构指针 |
| **返回值** | 成功：图层指针  失败：NULL |
| **参考** |  |
| **其他** | 在几何结构中可以添加多个几何图形。 |

#### app\_layer\_create\_multi\_menu

|  |  |
| --- | --- |
| **接口名** | [Layer](#Layer) \*app\_layer\_create\_multi\_menu(LayerMultiMenu \*layer\_multi\_menu) |
| **功能说明** | 创建一个菜单图层 |
| **参数说明** | layer\_multi\_menu 菜单结构指针 |
| **返回值** | 成功：图层指针  失败：NULL |
| **参考** |  |
| **其他** | 菜单图层中菜单项可以不等宽，可以自定义每个菜单项中显示什么类容及其排列。 |

#### app\_layer\_add\_multi\_menu\_item

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_add\_multi\_menu\_item(LayerMultiMenu \*lmm, MultiMenuItem \*mmi) |
| **功能说明** | 添加菜单项 |
| **参数说明** | lmm 菜单图层指针  mmi 菜单项元素指针 |
| **返回值** |  |
| **参考** |  |
| **其他** | 建议在一个菜单项，最多设置三个菜单项，如果要显示的菜单项过多，可以每次创建新的菜单图层（添加最新要显示的菜单项）替换旧的菜单图层。  菜单图层最后一个菜单项底部的Y轴坐标不能超过256，否则添加失败。 |

#### app\_layer\_add\_multi\_menu\_element

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_add\_multi\_menu\_element(MultiMenuItem \*mmi, Element \*e) |
| **功能说明** | 添加菜单项中元素 |
| **参数说明** | mmi 菜单项指针  e 菜单项中元素指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_multi\_menu\_selected

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_multi\_menu\_selected(Layer \*layer, uint8\_t selected) |
| **功能说明** | 设置菜单图层被选择项 |
| **参数说明** | layer 图层指针  selected 被选择菜单项序列 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_get\_multi\_menu\_selected

|  |  |
| --- | --- |
| **接口名** | int8\_t app\_layer\_get\_multi\_menu\_selected(Layer \*layer) |
| **功能说明** | 获取菜单图层被选择项 |
| **参数说明** | layer 图层指针 |
| **返回值** | 当前被选择的菜单项序列 |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_multi\_menu\_item\_text

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_multi\_menu\_item\_text(Layer \*layer, int8\_t item\_id, int8\_t element\_id, const char \*text); |
| **功能说明** | 设置菜单图层中某菜单项中某元素文本内容 |
| **参数说明** | layer 图层指针  item\_id 菜单项序列  element\_id 菜单项中元素序列  text 文本指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_multi\_menu\_item\_bitmap

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_multi\_menu\_item\_bitmap(Layer \*layer, int8\_t item\_id, int8\_t element\_id, GBitmap \*bitmap); |
| **功能说明** | 设置菜单图层中某菜单项中某元素位图 |
| **参数说明** | layer 图层指针  item\_id 菜单项序列  element\_id 菜单项中元素序列  bitmap 位图指针 |
| **返回值** | 成功 图层指针  失败 NULL |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_bg\_color

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_bg\_color(Layer \*layer, enum GColor color); |
| **功能说明** | 设置图层背景颜色 |
| **参数说明** | layer 图层指针  color 颜色 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_layer\_set\_selected\_status

|  |  |
| --- | --- |
| **接口名** | void app\_layer\_set\_selected\_status([Layer](#Layer) \*layer, uint8\_t status) |
| **功能说明** | 设置图层边界 |
| **参数说明** | layer 图层指针  status 选择状态  LAYER\_UNSELECTED 0 不能被选择  LAYER\_SELECTED 1 能被选择 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### maibu\_layer\_set\_visible\_status

|  |  |
| --- | --- |
| **接口名** | void maibu\_layer\_set\_visible\_status([Layer](#Layer) \*layer, bool status) |
| **功能说明** | 设置图层可见状态 |
| **参数说明** | layer 图层指针  status 可见状态  true 可见  false 隐藏 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 插件

#### app\_plug\_status\_bar\_create

|  |  |
| --- | --- |
| **接口名** | void app\_plug\_status\_bar\_create (Window \*window, GBitmap \*left\_bitmap, GBitmap \*right\_bitmap, const char \*text) |
| **功能说明** | 创建状态栏 |
| **参数说明** | window 状态栏所在窗口指针  left\_bitmap 左边显示的图片  right\_bitmap 右边显示的图片  text 中间显示的文本信息 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_plug\_status\_bar\_add\_time

|  |  |
| --- | --- |
| **接口名** | void app\_plug\_status\_bar\_add\_time (Window \*window) |
| **功能说明** | 状态栏中间显示时间 |
| **参数说明** | window 状态栏所在窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_plug\_status\_bar\_add\_battery

|  |  |
| --- | --- |
| **接口名** | void app\_plug\_status\_bar\_add\_battery (Window \*window) |
| **功能说明** | 状态栏右边显示电量 |
| **参数说明** | window 状态栏所在窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### app\_plug\_status\_bar\_add\_ble

|  |  |
| --- | --- |
| **接口名** | void app\_plug\_status\_bar\_add\_ble (Window \*window) |
| **功能说明** | 状态栏左边显示蓝牙状态 |
| **参数说明** | window 状态栏所在窗口指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

### 类型定义

#### enum [LayerType](#LayerType)

|  |  |
| --- | --- |
| **接口名** | enum [LayerType](#LayerType) |
| **功能说明** | 图层类型 |
| **参数说明** | LayerTypeBitmap, //位图  LayerTypeText, //文本  LayerTypeScroll, //滚动  LayerTypeGeometry, //几何  LayerTypeMultiMenu, //复杂菜单  LayerTypeStatusBar, //状态栏 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [LayerBitmap](#LayerBitmap)

|  |  |
| --- | --- |
| **接口名** | struct [LayerBitmap](#LayerBitmap) |
| **功能说明** | 位图图层结构体 |
| **参数说明** | GBitmap bitmap; //位图数据  GRect frame; //显示位图的矩形结构  enum GAlign alignment; //对齐方式 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [LayerText](#LayerText)

|  |  |
| --- | --- |
| **接口名** | struct [LayerText](#LayerText) |
| **功能说明** | 文本图层结构体 |
| **参数说明** | const char \*text; //文本数据  GRect frame; //显示文本的矩形结构  enum GAlign alignment; //对齐方式  uint8\_t font\_type:7; //字体类型  uint8\_t bound\_width:1; //边界宽度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [LayerScroll](#LayerScroll)

|  |  |
| --- | --- |
| **接口名** | struct [LayerScroll](#LayerScroll) |
| **功能说明** | 滚动图层结构体 |
| **参数说明** | GRect frame; //显示位图的矩形结构  uint16\_t key; //文本存放地址key  uint16\_t total\_bytes; //文本字节数  uint8\_t font\_type; //字体类型  uint8\_t gap; //行间  enum GColor color; //颜色  uint8\_t bound\_width; //边界宽度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct LayerMultiMenu

|  |  |
| --- | --- |
| **接口名** | struct LayerMultiMenu |
| **功能说明** | 菜单图层结构体 |
| **参数说明** | GRect frame; //显示菜单的矩形结构 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct MultiMenuItem

|  |  |
| --- | --- |
| **接口名** | struct MultiMenuItem |
| **功能说明** | 菜单项 |
| **参数说明** | GRect frame; //菜单项矩形结构指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct Element

|  |  |
| --- | --- |
| **接口名** | struct Element |
| **功能说明** | 菜单项中元素 |
| **参数说明** | void \*content; //元素类容  GRect frame; //存放元素矩形结构指针  uint8\_t type:1; //元素类型  uint8\_t font:7; //字体，如果是文字 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [LayerGeometry](#LayerGeometry)

|  |  |
| --- | --- |
| **接口名** | struct [LayerGeometry](#LayerGeometry) |
| **功能说明** | 几何图层结构体 |
| **参数说明** | uint8\_t num; //几何图形的个数  P\_Geometry \*p\_g; //几何图形指针数组 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [Geometry](#LayerGeometry)

|  |  |
| --- | --- |
| **接口名** | struct [Geometry](#LayerGeometry) |
| **功能说明** | 几何图形属性 |
| **参数说明** | enum GeometryType type; //几何类型  enum FillType fill\_type; //填充类型  enum GColor color; //颜色  void \*element; //构造要素，type类型不同，element为不同结构 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum [GeometryType](#GeometryType)

|  |  |
| --- | --- |
| **接口名** | enum [GeometryType](#GeometryType) |
| **功能说明** | 几何类型 |
| **参数说明** | GeometryTypeLine, //直线  GeometryTypePolygon, //多边形  GeometryTypeCircle, //圆  GeometryTypeLineSet //线段集合 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [Click](#Click)

|  |  |
| --- | --- |
| **接口名** | struct [Click](#Click) |
| **功能说明** | 按键结构体 |
| **参数说明** | [ButtonId](#ButtonId) id 按键类型  [ClickCallback](#ClickCallback) callback 对应的按键回调函数 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum [ButtonId](#ButtonId)

|  |  |
| --- | --- |
| **接口名** | enum [ButtonId](#ButtonId) |
| **功能说明** | 按键类型 |
| **参数说明** | [ButtonId](#ButtonId)Back 回退  [ButtonId](#ButtonId)Up 向上  [ButtonId](#ButtonId)Select选择  [ButtonId](#ButtonId)Down 向下 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### [ClickCallb](#ClickCallback)[ack](#ClickCallback)

|  |  |
| --- | --- |
| **接口名** | typedef void(\* [ClickCallback](#ClickCallback))( void \*context) |
| **功能说明** | 按键回调函数 |
| **参数说明** | context 按键传递的内容 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [GRect](#GRect)

|  |  |
| --- | --- |
| **接口名** | struct [GRect](#GRect) |
| **功能说明** | 矩形结构体 |
| **参数说明** | [GPoint](#GPoint) origin 矩形左上角坐标  [GSize](#GSize) size 矩形的高度和宽度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [GPoint](#GPoint)

|  |  |
| --- | --- |
| **接口名** | struct [GPoint](#GPoint) |
| **功能说明** | 坐标结构体 |
| **参数说明** | int16\_t x X轴位置  int16\_t y Y轴位置 |
| **返回值** |  |
| **参考** |  |
| **其他** | 坐标以屏幕的左上角为源点（0,0），向下为Y轴延伸，向右为X轴延伸 |

#### struct [GSize](#GSize)

|  |  |
| --- | --- |
| **接口名** | struct [GSize](#GSize) |
| **功能说明** | 大小结构体 |
| **参数说明** | int16\_t h 高度  int16\_t w 宽度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [GBitmap](#GBitmap)

|  |  |
| --- | --- |
| **接口名** | struct [GBitmap](#GBitmap) |
| **功能说明** | 位图数据结构体 |
| **参数说明** | uint8\_t height; //像素高度  uint8\_t width; //像素宽度 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum [GAlign](#GAlign)

|  |  |
| --- | --- |
| **接口名** | struct [GAlign](#GAlign) |
| **功能说明** | 显示方式 |
| **参数说明** | GAlignCenter 居中  GAlignTopLeft  左上  GAlignTopRight 右上  GAlignTop 顶部  GAlignLeft 靠左  GAlignBottom 底部  GAlignRight 靠右  GAlignBottomRight 右下  GAlignBottomLeft  左下 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum [GColor](#GColor)

|  |  |
| --- | --- |
| **接口名** | struct [GColor](#GColor) |
| **功能说明** | 颜色定义 |
| **参数说明** | GColorBlack 黑色  GColorWhite 白色 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [GPathInfo](#GPathInfo)

|  |  |
| --- | --- |
| **接口名** | struct [GPathInfo](#GPathInfo) |
| **功能说明** | 多边形坐标结构体 |
| **参数说明** | uint32\_t num\_points 坐标个数  [GPoint](#GPoint) \*points 坐标指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum [FillType](#FillType)

|  |  |
| --- | --- |
| **接口名** | enum [FillType](#FillType) |
| **功能说明** | 填充类型 |
| **参数说明** | FillArea 区域填充  FillOutline 边界填充 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct [GFont](#GFont)

|  |  |
| --- | --- |
| **接口名** | typedef void \* [GFont](#GFont) |
| **功能说明** | 字体结构 |
| **参数说明** | uint8\_t height; //字体高度像素  uint8\_t width; //字体宽度像素  uint8\_t type; //字体类型 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### [AnimationType](#AnimationType)

|  |  |
| --- | --- |
| **接口名** | enum [AnimationType](#AnimationType) |
| **功能说明** | 动画类型 |
| **参数说明** | AnimationType0 无动画  AnimationType1 待定  AnimationType2 待定 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct Line

|  |  |
| --- | --- |
| **接口名** | typedef struct Line |
| **功能说明** | 直线结构体 |
| **参数说明** | GPoint p0; //直线起点  GPoint p1; //直线终点 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct LineSet

|  |  |
| --- | --- |
| **接口名** | typedef struct LineSet |
| **功能说明** | 直线集合结构体 |
| **参数说明** | uint32\_t num\_lines; //直线个数  P\_Line lines; //直线数组指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct Circle

|  |  |
| --- | --- |
| **接口名** | typedef struct Circle |
| **功能说明** | 圆结构体 |
| **参数说明** | GPoint center; //圆心  uint8\_t radius; //半径 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### struct Polygon

|  |  |
| --- | --- |
| **接口名** | typedef struct Polygon |
| **功能说明** | 多边形结构体 |
| **参数说明** | uint8\_t num\_points; //坐标个数  GPoint \*points; //坐标指针 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

## OS系统信息

### 接口

#### maibu\_get\_user\_account

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_user\_account(char \*buff, int8\_t size) |
| **功能说明** | 获取与手表绑定的用户账户信息（目前是手机号码） |
| **参数说明** | buff 存储账户信息缓存  size 缓存大小 |
| **返回值** | 实际读取的账号信息大小 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_os\_version

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_os\_version(char \*buff, int8\_t size) |
| **功能说明** | 获取手表OS版本号 |
| **参数说明** | buff 存储版本号缓存  size 缓存大小 |
| **返回值** | 实际读取的版本号大小 |
| **参考** | 如：1.0.1\_alpha |
| **其他** |  |

#### maibu\_get\_watch\_id

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_watch\_id (char \*buff, int8\_t size) |
| **功能说明** | 获取手表ID |
| **参数说明** | buff 存储ID缓存  size 缓存大小 |
| **返回值** | 实际读取的ID大小 |
| **参考** | 如：NGDUXJASB1 |
| **其他** |  |

#### maibu\_get\_bootloader\_version

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_bootloader\_version (char \*buff, int8\_t size) |
| **功能说明** | 获取bootloader版本 |
| **参数说明** | buff 存储版本缓存  size 缓存大小 |
| **返回值** | 实际读取的大小 |
| **参考** | 如：1.0.1 |
| **其他** |  |

#### maibu\_get\_sport\_data

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_sport\_data ([SportData](#SportData) \*data, uint8\_t day) |
| **功能说明** | 获取运动数据 |
| **参数说明** | data 存储运动数据结构体指针  day 获取N天之前的运动数据，0 表示当天，1表示昨天，依次类推。系统最多保存最近30天的运动数据。 |
| **返回值** | 0 成功  >0错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_ accel\_data

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_accel\_data (int16\_t \*x, int16\_t \*y, int16\_t \*z) |
| **功能说明** | 获取传感器三轴数据 |
| **参数说明** | x X轴数据  y Y轴数据  z Z轴数据  某轴加速度为0时，该轴数值为2048左右。1g加速度变化为250左右。 |
| **返回值** | 0 成功  >0错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_altitude

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_altitude (float \*altitude, float \*accuracy) |
| **功能说明** | 获取海拔高度 |
| **参数说明** | altitude 海拔，单位米，保留小数点后1位  accuracy 精度，单位米，保留小数点后1位 |
| **返回值** | 0 成功  >0错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_pressure

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_pressure (float \*pressure) |
| **功能说明** | 获取气压 |
| **参数说明** | pressure 气压值，单位帕，保留小数点后1位 |
| **返回值** | 0 成功  >0错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_temperature

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_temperature (float \*temperature) |
| **功能说明** | 获取温度 |
| **参数说明** | temperature 温度值，单位摄氏度，保留小数点后1位 |
| **返回值** | 0 成功  >0错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_ble\_status

|  |  |
| --- | --- |
| **接口名** | enum [AppCommStatus](#AppCommStatus) maibu\_get\_ble\_status () |
| **功能说明** | 获取蓝牙状态 |
| **参数说明** |  |
| **返回值** | 蓝牙状态，见AppCommStatus |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_ble\_rssi

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_ble\_rssi (int8\_t \*rssi) |
| **功能说明** | 获取蓝牙信息强度 |
| **参数说明** | rssi 信号强度 一般在-120~+15，值越大信号越强 |
| **返回值** | 0 成功  >0 错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_battery\_percent

|  |  |
| --- | --- |
| **接口名** | int8\_t maibu\_get\_battery\_percent(int8\_t \*percent) |
| **功能说明** | 获取电量百分比 |
| **参数说明** | percent 电量百分比 |
| **返回值** | 0 成功  >0 错误码 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_system\_tick

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_get\_system\_tick() |
| **功能说明** | 获取系统时钟，从系统启动开始的毫秒数 |
| **参数说明** |  |
| **返回值** | >0 从系统启动开始的毫秒数 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_speed\_info

|  |  |
| --- | --- |
| **接口名** | uint32\_t maibu\_get\_speed\_info(SpeedInfo \*info) |
| **功能说明** | 获取实时速度及其他信息 |
| **参数说明** | 速度信息结构指针，见类型定义 |
| **返回值** | 返回实时速度 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_lunar\_calendar

|  |  |
| --- | --- |
| **接口名** | int maibu\_get\_lunar\_calendar(struct os\_date\_time \*p\_date, SLunarData \*p\_data) |
| **功能说明** | 根据公历日期获取农历日期数据 |
| **参数说明** | p\_date 公历日期结构体指针，如果为NULL，返回当天农历数据。  p\_data 农历数据结构体指针 |
| **返回值** | 0 成功  -1 失败 |
| **参考** |  |
| **其他** |  |

#### maibu\_get\_phone\_type

|  |  |
| --- | --- |
| **接口名** | enum PhoneType maibu\_get\_phone\_type(void) |
| **功能说明** | 获取与手表连接的手机类型 |
| **参数说明** |  |
| **返回值** | 手机类型，见PhoneType |
| **参考** |  |
| **其他** |  |

### 类型定义

#### typedef struct SportData

|  |  |
| --- | --- |
| **接口名** | typedef struct SportData |
| **功能说明** | 运动数据结构体 |
| **参数说明** | /\*当天计步数\*/  uint32\_t step;  /\*当天楼层\*/  uint32\_t floor;  /\*当天走路距离，单位厘米\*/  uint32\_t distance;  /\*消耗卡路里，单位千卡\*/  uint32\_t calorie;  /\*活动度\*/  uint32\_t activity; |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum BLEStatus

|  |  |
| --- | --- |
| **接口名** | enum BLEStatus |
| **功能说明** | 蓝牙状态 |
| **参数说明** | BLEStatusAdvertising 广播  BLEStatusConnected 被连接  BLEStatusUsing 使用中（发送数据）  BLEStatusClose 关闭 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### typedef struct SpeedInfo

|  |  |
| --- | --- |
| **接口名** | typedef struct SpeedInfo |
| **功能说明** | 速度信息结构体 |
| **参数说明** | /\*实时速度，单位：cm/s \*/  uint32\_t real\_time\_speed;  /\*距离上一次速度更新时间间隔\*/  uint32\_t interval;  /\*是否步行中, 0：停止或起步状态；1：步行中\*/  uint8\_t flag; |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### typedef struct SLunarData

|  |  |
| --- | --- |
| **接口名** | typedef struct SLunarData |
| **功能说明** | 农历数据结构体 |
| **参数说明** | char year[7]; //丙申  char zodiac[4]; //猴  char mon[7]; //正月  char day[7]; //十五  char solar\_term[7]; //谷雨  char festival[10]; //国庆节，注：公历农历节日二选一 |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

#### enum PhoneType

|  |  |
| --- | --- |
| **接口名** | enum PhoneType |
| **功能说明** | 手机类型枚举值 |
| **参数说明** | PhoneTypeUnknown,  PhoneTypeAndroid,  PhoneTypeIOS |
| **返回值** |  |
| **参考** |  |
| **其他** |  |

## APP日志

#### maibu\_clear\_log

|  |  |
| --- | --- |
| **接口名** | void maibu\_clear\_log(void) |
| **功能说明** | 清除日志信息 |
| **参数说明** |  |
| **返回值** |  |
| **参考** |  |
| **其他** | 系统会专门提供一块区域用来保存所有应用的日志信息，大小500字节，超过则会覆盖保存，建议单独调试某一应用时，先使用该接口清除。 |

#### maibu\_print\_log

|  |  |
| --- | --- |
| **接口名** | void maibu\_print\_log(char \*log) |
| **功能说明** | 写日志 |
| **参数说明** | log 要保存的日志信息buffer |
| **返回值** |  |
| **参考** |  |
| **其他** | 在应用中使用该接口保存日志信息后，需要在应用商店下载专门的debug应用去查看日志信息，log的长度不超过500字节。 |