中科蓝讯
bluetrum

BT8918 手表
SDK 开发指南

VERSION：0.0.2

Declaration

Bluetrum reserves the right to make changes without further notice to any products herein to improve reliability, function or design.

For further information on the technology, product and business term, please contact Bluetrum Company.
For sales or technical support, please send email to the address:
Sales: sales@bluetrum.com
Technical: project@bluetrum.com

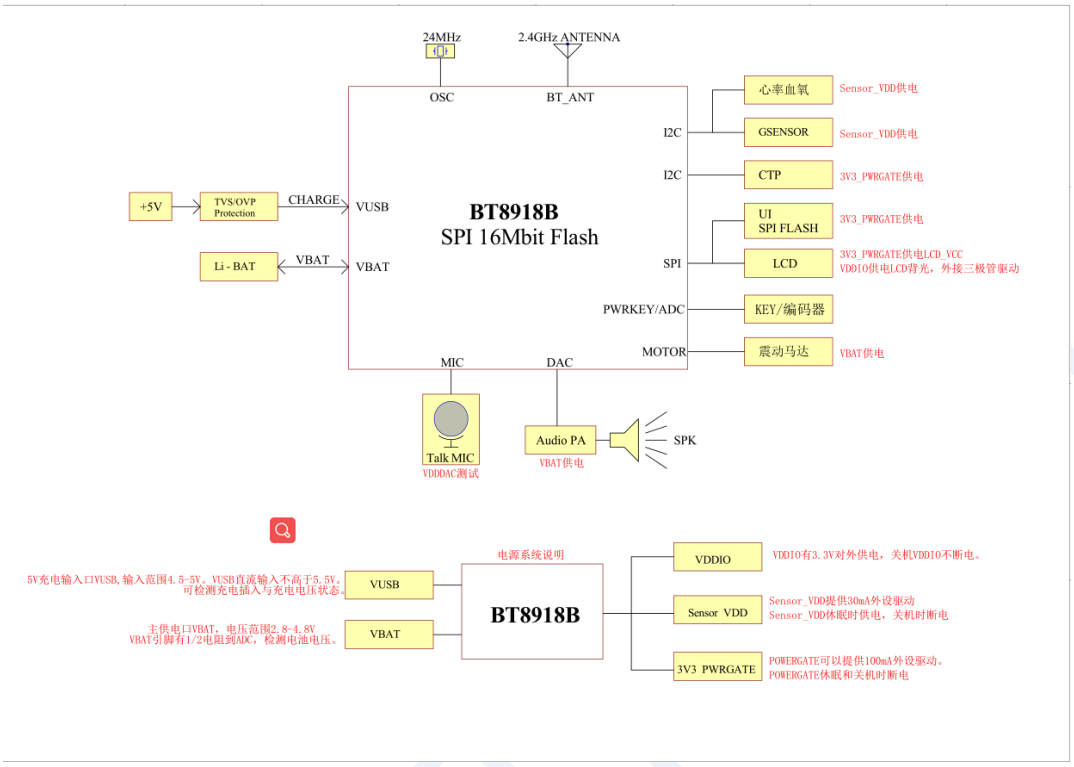
Revision History

Date	Version	Comments	Revised by
2022/2/18	0.0.1	初版 SDK 说明文档	Wilson

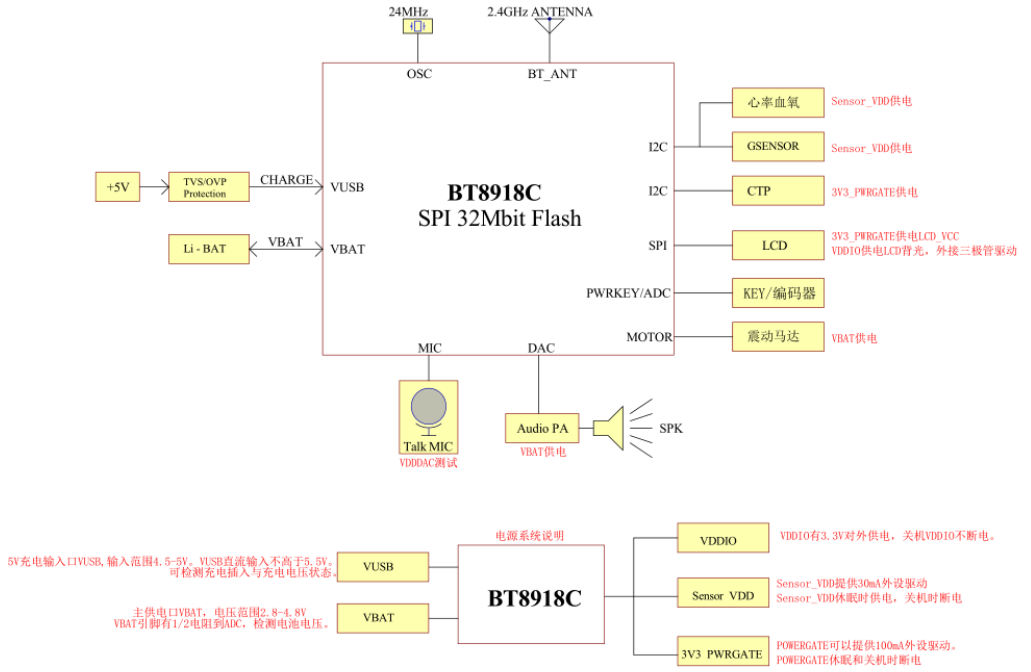
目 录

REVISION HISTORY 1
目录 2
1 硬件框架 3
2 开发板介绍 4
3 SDK 开发指南 5
3.1 综述 5
3.2 任务架构 6
3.3 GUI 介绍 7
3.4 GUI 使用示例 8
3.5 SPI 使用指南 9
3.6 IIC 使用指南 10
3.7 功耗数据 10
4 开发工具 11
4.1 DOWNLOADER 图像转换/烧录 11
3.1.1 内/外部资源烧录 11
3.1.2 单独外部资源烧录 12
3.1.3 图像格式转换： 13
4.2 点阵字库生成器 14
附录 15

1 硬件框架

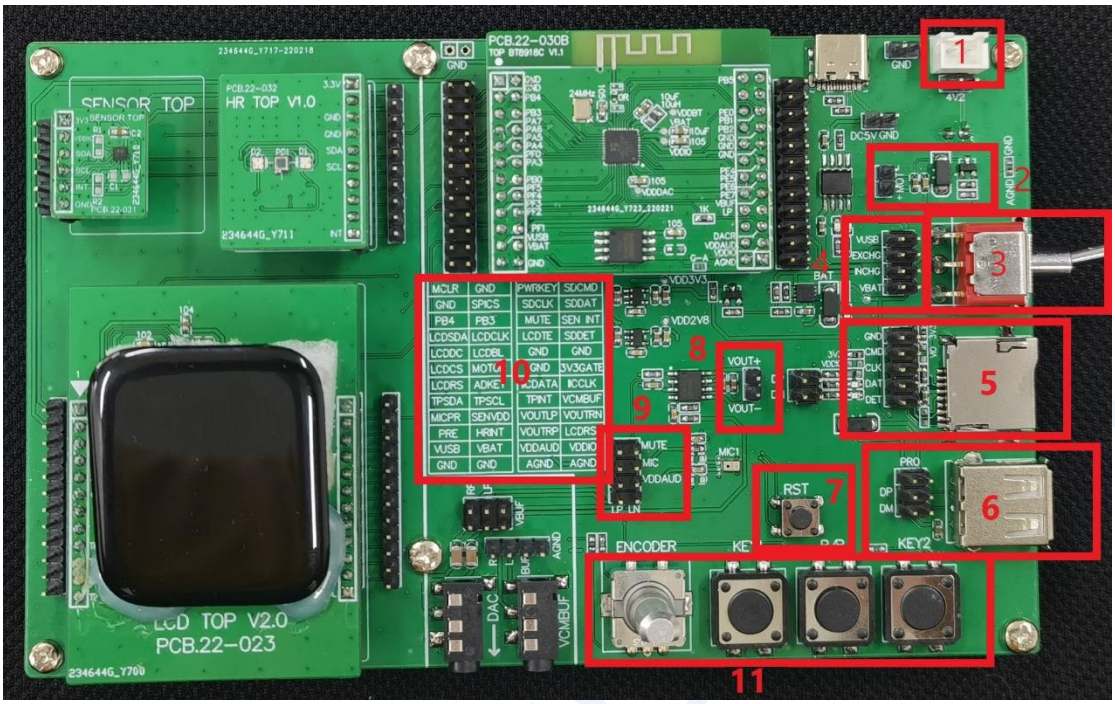


BT8918B



BT8918C

2 开发板介绍



1.电池接口。

2.震动马达接口。

3.电池开关。

4.电流测试接口/供电选择。测试电流时，将电流表串接在 VBAT/VUSB 插针上，可以测试对应 SITE x 的电流；不需要测试电流时则将对应的短接帽插上。a)开发板自带了 HX4056 电池管理芯片，电池亏电时，通过 TYPE C 供电 5V，短接 EXCHG x 进行充电；b)若使用蓝讯 IC 充电功能时，则把对应的 VUSB x / INCHG x /VBAT x 短接，进行充电。

5.TF 卡接口。需要调试 TF 卡时，将插针短接。

6.USB 接口。调试 USB 功能和下载时，短接 DP 插针。

7.复位按键 RST。此按键控制芯片 VBAT 的供电，按键按下时，对应的 VBAT x 断电；松开按键后，VBAT x 重新上电，实现复位的作用。

8.喇叭接口。

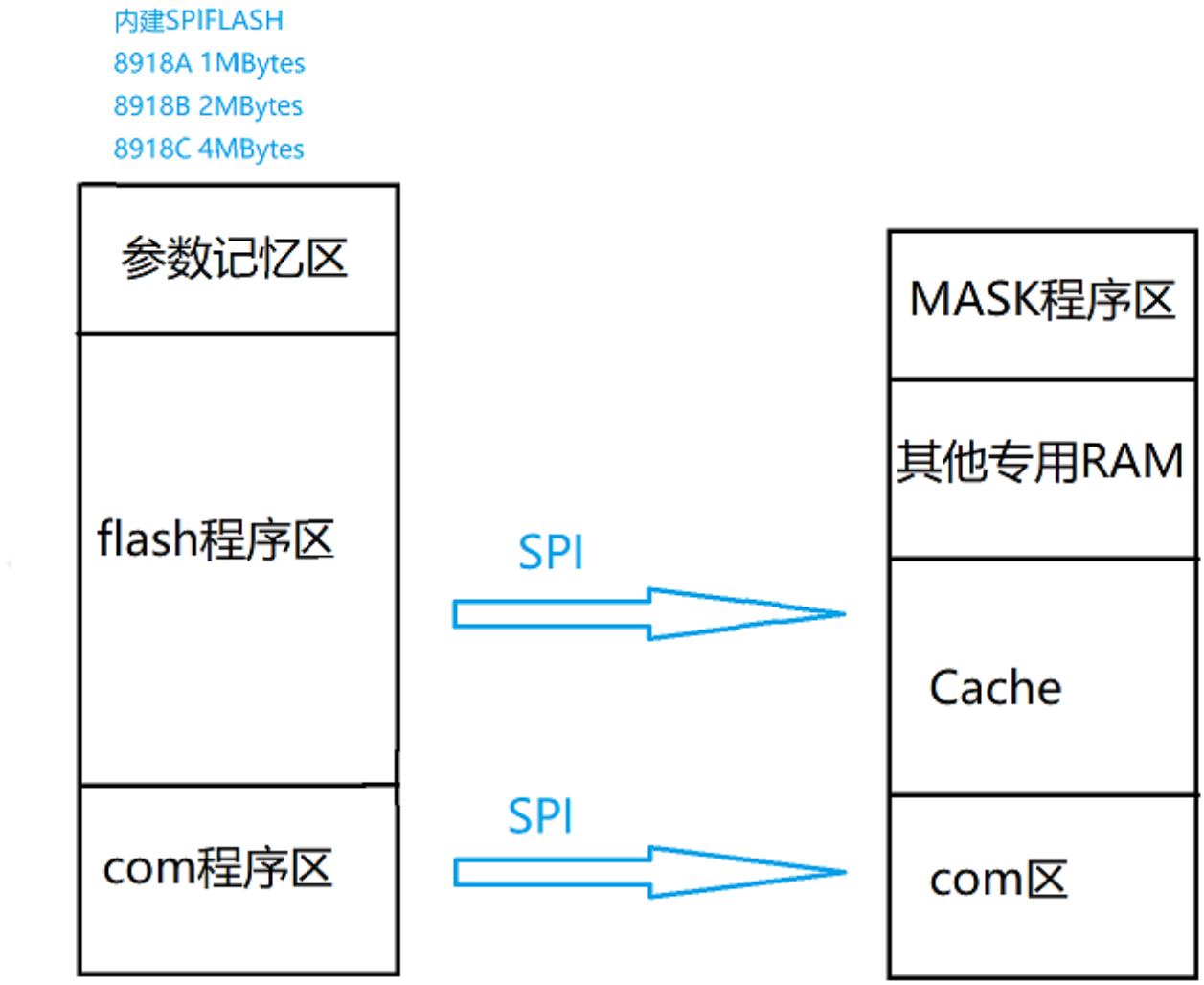
9.MIC 通路选择/PA MUTE。a)需要使用 MIC 功能时，选择 MIC 插针短接，同时短接 VDDAUD 插针；（V2.0 和 V2.1 版本开发板请用杜邦线交叉短接，后续版本会做修改）b)使用到开发板的 PA 功放时，短接 MUTE 插针，实现对 PA 的控制。

10.IC 核心板丝印对照区。此区域与核心板插针一一对应，一般地，核心板芯片的功能引脚均在顶板标有丝印，用户可进行功能对照。

11.按键及编码器，可以通过软件自定义按键功能。

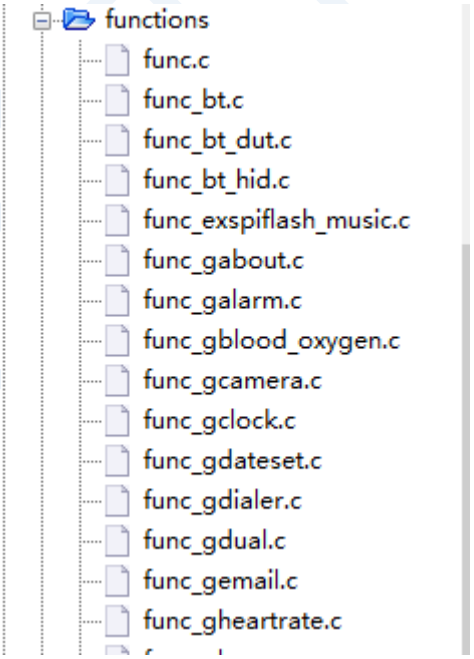
3 SDK 开发指南

3.1 综述



BT8918 系列采用最近比较流行的 RISC-V(32 位)开源内核架构+国产 RT-Thread 操作系统，其中 8918A 内建 1Mbytes Flash，8918B 内建 2Mbytes Flash，8918C 内建 4Mbytes FLASH。SDK 对休眠进行深度优化，目前休眠连接功耗低至 199ua，大大提高整机的续航，BT8918 主频最高 160MHz，配套自主研发的 GUI ，ROM 使用情况：标准 sdkflash 代码为 460KB，RAM 使用情况 GUI 约占 30K ram，剩余 40K ram 可供用户自主开发；同时拥有 2 组 SPI 总线，其中一组内部 flash 使用，一组 LCD 使用，支持普通单线 SPI，双线 DSPI，不支持 QSPI，SPI 总线频率最大 80M；IIC TP、GSENSOR 已接入 SDK，到手即可使用。

3.2 任务架构



目前 SDK 采用的是任务式的流程，每个界面就是一个 func 任务，使用方法同蓝讯任意 SDK 相同，只需要对单独的一个任务进行开发，即可轻松整合到手表功能中。

3.3 GUI 介绍

```
///widget
void widgets_init(u8 *buf, int buf_size);                       //初始化控件Buffer
void widgets_clear(void);                                       //清除控件Buffer
void *widget_get_head(void);                                    //获取第一个控件的地址
void *widget_creat(void *parent, u8 type);                      //创建控件, parent:父控件, type:控件类型
void widget_free(void *widget, u8 type);                        //清除控件, widget:控件, type:控件类型
void widget_set_pos(void *widget_ptr, int x, int y);            //设置控件位置, widget:控件, x:横坐标, y:纵坐标
void widget_set_pos_wait(void *widget_ptr, int x, int y);       //等待刷完本帧后才设置一个控件位置, 用于快速更新坐标, widget:控件, x:横坐标, y:纵坐标
void widget_set_pos_centre(void *widget_ptr, int x, int y);     //设置控件位置并居中, widget:控件, x:横坐标, y:纵坐标
void widget_set_visible(void *widget_ptr, bool visible);        //设置控件是否可视化, widget_ptr:控件, visible:1->可视, 0->不可视
void widget_set_transparent(void *widget_ptr, bool transparent);//设置控件是否透明, widget_ptr:控件, transparent:1->透明, 0->不透明

//设置页的触摸回调函数, page_ptr:页控件, press_callback:按下事件回调, release_callback:抬起事件回调, move_callback:滑动事件回调, para:按钮编号
void widget_page_set_callback(void *page_ptr, void *press_callback, void *release_callback, void *move_callback);

//设置控件位置及大小, widget_ptr:控件, x:横坐标, y:纵坐标;宽, 高设置不同于控件大小时, 会自动进行缩放
void widget_set_location(void *widget_ptr, int x, int y, int width, int height);
//设置控件位置及大小并居中, widget_ptr:控件, x:横坐标, y:纵坐标;宽, 高设置不同于控件大小时, 会自动进行缩放
void widget_set_location_centre(void *widget_ptr, int x, int y, int width, int height);

///widget line
void widget_set_color(void *widget_ptr, int color);             //设置线段颜色, widget_ptr:控件, color:rgb565颜色
//设置线段位置及大小并居中, widget_ptr:控件, x:起始横坐标, y:起始纵坐标, x1:结束横坐标, y1:结束纵坐标;
void widget_set_line_location(void *widget_ptr, int x, int y, int x1, int y1);
```

```
///widget image
void *load_res_img(u32 addr);                                   //加载图片资源
void free_all_res(void);                                        //清除所有的图片资源
void widget_image_free_res(void *img_ptr);                      //清除指定的图片资源
void widget_image_set_rotation(void *img_ptr, u8 rotation);     //设置图片翻转,img_ptr:图片控件,rotation:ROT_HFLIP水平翻转,ROT_VFLIP垂直翻转
//设置图片资源中截取位置及宽高, img_ptr:图片控件, p_img:图片资源结构体, x:图片资源截取的起始横坐标, y:图片资源截取的起始纵坐标;宽, 高
void widget_image_set(void *img_ptr, const void *p_img, int x, int y, int width, int height);

///widget button
//设置按钮控件资源中截取位置及宽高, btn_ptr:按钮控件, p_icon:图片资源结构体, x:图片资源截取的起始横坐标, y:图片资源截取的起始纵坐标;宽, 高
void widget_button_set_img(void *btn_ptr, const void *p_icon, int x, int y, int width, int height);
//设置按钮控件的触摸回调函数, btn_ptr:按钮控件, press_callback:按下事件回调, release_callback:抬起事件回调, move_callback:滑动事件回调, para:按钮编号
void widget_button_set_callback(void *btn_ptr, void *press_callback, void *release_callback, void *move_callback, int para);

///widget text
void widget_text_clear(void);                                   //清除文本控件
//设置文本控件初始化, str_ptr:文本控件, a_addr:ascii的flash起始地址, g_addr:gbk2312的flash起始地址, u_addr:utf2gbk的flash起始地址, type_size:字体高度24
void widget_text_set_res(void *str_ptr, u32 a_addr, u32 g_addr, u32 u_addr, u32 type_size);
//设置文本控件位置及颜色, str_ptr:文本控件, content:文本字符串, x:横坐标, y:纵坐标;front_color:文本颜色, back_color:背景色
void widget_text_set_content(void *str_ptr, char *content, int x, int y, int front_color, int back_color);

///widget rectangle
//设置矩形控件属性, widget:矩形控件, x0:横坐标, y0:纵坐标,宽, 高, color:rgb565颜色, padding:1->填充, 0->不填充
void widget_rectangle_set(void *widget, s16 x0, s16 y0, u16 wid, s16 hei, int color, bool padding);

///widget number
void widget_num_free(widget_num_t *num);                        //清除指定的数字控件
//创建数字控件, parent:父控件, num_addr:数字图片资源地址, number_cnt:需要显示数字的数量
widget_num_t *widget_num_create(void *parent_ptr, u32 num_addr, u16 number_cnt);
//创建数字控件, num_ptr:数字控件, content:数字字符串, x:横坐标, y:纵坐标, 宽高
void widget_num_set_content(void *num_ptr, const char *content, int x, int y, int width, int height);
```

```
///widget number
void widget_num_free(widget_num_t *num);                        //清除指定的数字控件
//创建数字控件, parent:父控件, num_addr:数字图片资源地址, number_cnt:需要显示数字的数量
widget_num_t *widget_num_create(void *parent_ptr, u32 num_addr, u16 number_cnt);
//创建数字控件, num_ptr:数字控件, content:数字字符串, x:横坐标, y:纵坐标, 宽高
void widget_num_set_content(void *num_ptr, const char *content, int x, int y, int width, int height);

///compo list
void *compo_list_create(s16 dc_width, s16 dc_height);           //创建菜单列表, dc_width:屏幕宽, dc_height:屏幕高
void *compo_list_item_create(void *parent);                     //创建菜单列表(图标+文本), parent:父控件
void compo_list_set_lineheight(int line_height);                //设置菜单列表每一项的高
void compo_list_set_iconsize(int icno_size);                    //设置菜单列表每一图标的高
void compo_list_set_color(int color);                           //设置菜单列表文本(单色图)的颜色

//设置菜单图标资源中截取位置及宽高, item:菜单列表控件(图标+文本), p_img:图片资源结构体,x:图片资源截取的起始横坐标, y:图片资源截取的起始纵坐标;宽,高
void compo_list_item_set_icon(void *item, const void *p_img, int x, int y, int width, int height);
//设置菜单文本图片资源中截取位置及宽高, item:菜单列表控件(图标+文本), p_img:图片资源结构体,x:图片资源截取的起始横坐标, y:图片资源截取的起始纵坐标;宽,高
void compo_list_item_set_text(void *item, const void *p_img, int x, int y, int width, int height);

///compo star
void *compo_star_create(s16 dc_width, s16 dc_height);           //创建蜂窝菜单, dc_width:屏幕宽, dc_height:屏幕高
void compo_star_set_icon(int icon_size);                        //设置蜂窝图标每一项高
void compo_star_set_focus(int x, int y);                        //设置蜂窝组件中心焦点偏移
void compo_star_update(void);                                   //重新计算蜂窝坐标

bool gui_res_wait(void);                                        //等待本帧结束
```

在 api_gui.h 中，SDK 内部提供了丰富的 GUI 接口，支持创建页，按钮，图像，线段，矩形，文本，时间，数字，菜单，蜂窝界面等控件，并且支持对这些控件进行翻转缩放处理，目前 SDK 已经对界面过渡动画进行了处理，用户只需关心子界面的开发即可。

3.4 GUI 使用示例

```
AT(.text.func.msg)
void *lcd_msg_init(int x, int y)
{
    //创建msg页面
    void *page_ptr = widget_creat(NULL, WGT_TYPE_PAGE);

    //创建矩形空间，父控件为msg页面
    void *rect_ptr = widget_creat(page_ptr, WGT_TYPE_RECT);

    //设置矩形为msg页面黑底图
    widget_rectangle_set(rect_ptr, 0, 0, LCD_WIDTH_SIZE, LCD_HEIGHT_SIZE, COLOR_BLACK, 1);

    //创建任务文本控件，父控件为msg页面
    void *text = widget_creat(page_ptr, WGT_TYPE_TXT);

    //初始化任务文本控件字库资源
    widget_text_set_res(text, UI_BUF_FONT_ASCII_BIN, UI_BUF_FONT_GBK2312_BIN, UI_BUF_UTF2GBK_BIN, 24);

    //设置任务文本控件显示内容"msg"、坐标、前景色、背景色
    widget_text_set_content(text, "Msg", TEXT_DESC_X, TEXT_DESC_Y, TEXT_DESC_COLOR, COLOR_BLACK);

    //创建右上角时钟显示，父控件为msg页面
    gui_clock_init(page_ptr);

    //创建ancs标题、内容文本控件，父控件为msg页面
    gmsg_cb.text_title = widget_creat(page_ptr, WGT_TYPE_TXT);
    gmsg_cb.text_message = widget_creat(page_ptr, WGT_TYPE_TXT);

    //初始化ancs标题、内容文本控件字库资源
    widget_text_set_res(gmsg_cb.text_title, UI_BUF_FONT_ASCII_BIN, UI_BUF_FONT_GBK2312_BIN, UI_BUF_UTF2GBK_BIN, 24);
    widget_text_set_res(gmsg_cb.text_message, UI_BUF_FONT_ASCII_BIN, UI_BUF_FONT_GBK2312_BIN, UI_BUF_UTF2GBK_BIN, 24);

    //设置任务文本控件显示内容、坐标、前景色、背景色
    widget_text_set_content(gmsg_cb.text_title, gmsg_cb.title, 10, 44, COLOR_RED, COLOR_BLACK);
    widget_text_set_content(gmsg_cb.text_message, gmsg_cb.messages, 10, 70, TEXT_DESC_COLOR, COLOR_BLACK);

    //设置msg页面坐标、大小
    widget_set_location(page_ptr, x, y, LCD_WIDTH_SIZE, LCD_HEIGHT_SIZE);

    //设置msg页面可视化
    widget_set_visible(page_ptr, true);

    //设置msg触摸回调函数
    widget_page_set_callback(page_ptr, NULL, &gmsg_touch_release_callback , &gmsg_touch_move_callback);

    //通知刷新界面
    gui_refresh();

    return page_ptr;
}
```

3.5 SPI 使用指南

以下是 LCD、内外部 Flash API 函数:

```
//LCD API
void spi_lcd_init(void);                                                    //SPI LCD初始化,宏GUI_SELECT选择对应型号
u32 lcd_id_read(void);                                                      //SPI LCD读取ID
void lcd_dma_send(uint16_t *addr, uint16_t len);                            //LCD SPI发送,addr:buf首地址 len:字节长度
void lcd_set_window(uint16_t x1,uint16_t y1,uint16_t x2,uint16_t y2);       //LCD设置窗口大小,x1:x轴起始位置 x2:x轴结束位置 y1:y轴起始位置 y2:y轴结束位置
void spil_channel_select(u8 type);                                          //SPI MISO复用LCD IO,type: 1 LCD, 0 flash

//内置FLASH API
void os_spiflash_erase(u32 addr);                                           //内部 SPI FLASH 4K擦除,addr:擦除首地址
void os_spiflash_read_res(void *buf, u32 addr, uint len);                   //内部 FLASH读函数,主要用于读取内部资源文件 buf:读取数据buf首地址 addr:目的地址 len:字节长度
void os_spiflash_program(void *buf, u32 addr, u32 len);                     //内部 FLASH写函数,buf:写入数据buf首地址 addr:写入地址 len:字节长度

//外置FLASH API
void spi_flash_init(void);                                                  //SPI FLASH初始化
u32 spi_flash_id_read(void);                                                //SPI FLASH读取ID
void spi_flash_erase(u32 addr);                                             //SPI FLASH 4K擦除,addr:擦除首地址
void spi_flash_read(void *buf, u32 addr, u32 len);                          //SPI FLASH CPU读函数, buf:读取数据buf首地址 addr:目的地址 len:字节长度 波特率55M以下
void spiflash1_fast_read(void *buf, u32 addr, u32 len);                     //SPI FLASH CPU读函数, buf:读取数据buf首地址 addr:目的地址 len:字节长度 波特率55M以上
void spiflash1_dma_read(void *buf, u32 addr, u32 len);                      //SPI FLASH DMA读函数, buf:读取数据buf首地址 addr:目的地址 len:字节长度 波特率55M以下
void spiflash1_dma_fast_read(void *buf, u32 addr, u32 len);                 //SPI FLASH DMA读函数, buf:读取数据buf首地址 addr:目的地址 len:字节长度 波特率55M以上
void spiflash1_db_dma_fast_read(void *buf, u32 addr, u32 len);              //SPI FLASH DMA双线读函数, buf:读取数据buf首地址 addr:目的地址, len:字节长度, 波特率55M以上
void spi_flash_dma_write(void *buf, u32 addr, u32 len);                     //SPI FLASH DMA写函数, buf:写入数据buf首地址, addr:写入地址, len:字节长度
```

设置 SPI 波特率:

SPIBAUD = SYS_CLK_SEL /(1 + SPIFALSH_DIV)

```
#define SPIFALSH_DIV                    1           //0:不分频 1:1分频, 计算方式: SPIBAUD = SYS_CLK_SEL /(1 + SPIFALSH_DIV)
#define SPILCD_DIV                      2           //0:不分频 1:1分频, 计算方式: SPIBAUD = SYS_CLK_SEL /(1 + SPIFALSH_DIV)
```

设置 LCD 型号:

```
#define GUI_SELECT                      GUI_LCD_7789V3          //GUI Display Select
#define UART0_PRINTF_SEL                PRINTF_PB3              //选择UART打印信息输出IO，或关闭打印信息输出


#define GUI_LCD                         (DISPLAY_LCD | 0x00)            //LCD点阵屏(128*64)
#define GUI_LCD_9306                    (DISPLAY_LCD | 0x01)            //LCD点阵屏(240*280)    for watch
#define GUI_LCD_9306V2                  (DISPLAY_LCD | 0x02)            //LCD点阵屏(240*280)    for watch 1.69 inch
#define GUI_LCD_3023                    (DISPLAY_LCD | 0x03)            //LCD点阵屏(128*160)    for watch
#define GUI_LCD_7789                    (DISPLAY_LCD | 0x04)            //LCD点阵屏(240*240)    for watch
#define GUI_LCD_7789V3                  (DISPLAY_LCD | 0x05)            //LCD点阵屏(240*280)    for watch
#define GUI_LCD_7789S                   (DISPLAY_LCD | 0x06)            //LCD点阵屏(240*240)    for watch
#define GUI_LCD_9307                    (DISPLAY_LCD | 0x07)            //LCD点阵屏(240*240)    for watch
```

3.6 IIC 使用指南

IIC API 函数:

```
void bsp_i2c_init(void);                          //软硬I2C初始化
void bsp_sw_i2c_tx_ack(void);                     //I2C发送带ack
bool bsp_sw_i2c_rx_ack(void);                     //I2C接收带ack
void bsp_sw_i2c_tx_nack(void);                    //I2C接收不带ack
void bsp_sw_i2c_start(void);                      //I2C发送起始信号
void bsp_sw_i2c_stop(void);                       //I2C发送停止信号
void bsp_sw_i2c_tx_byte(uint8_t dat);             //I2C发送1Bytes
uint8_t bsp_sw_i2c_rx_byte(void);                 //I2C接收1Bytes
```

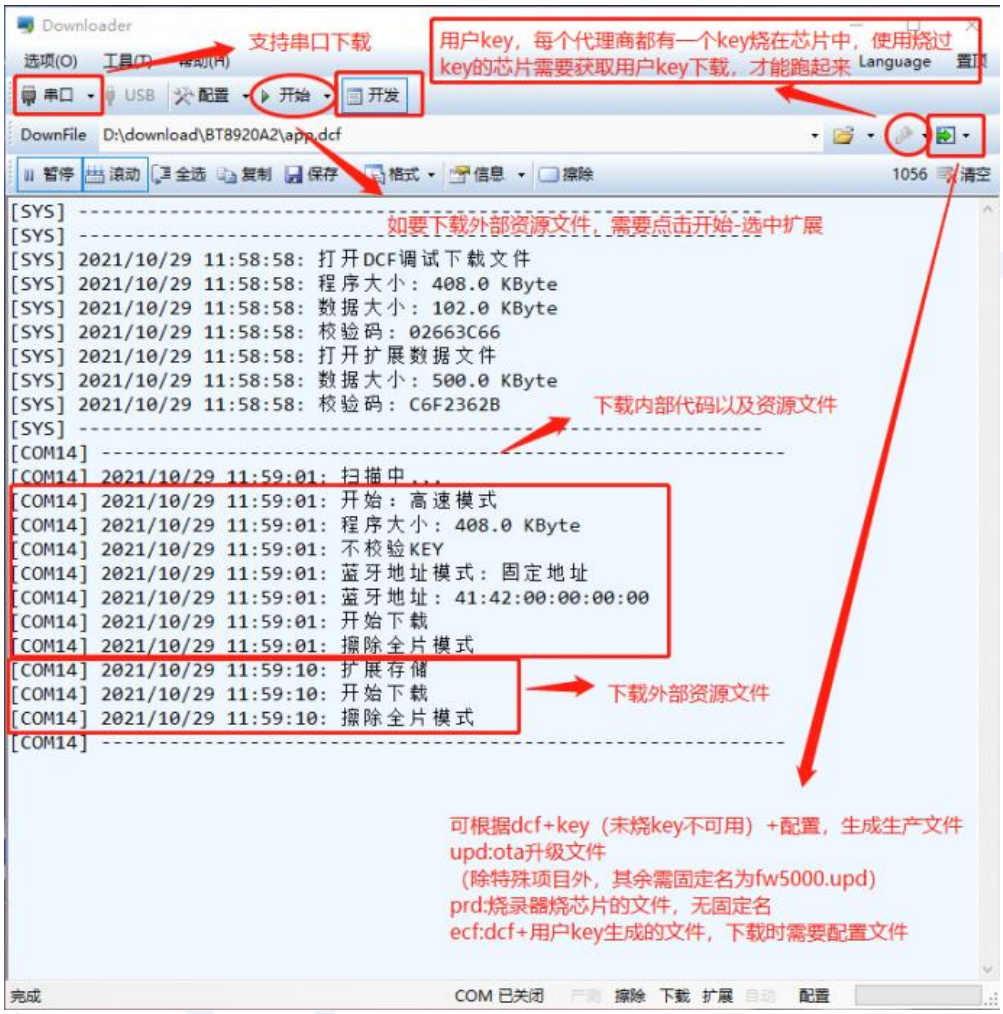
3.7 功耗数据

电池电压: 4.166V
硬件版本: 8918 手表开发板
测试手机: Iphone 12
SDK 版本: S3570

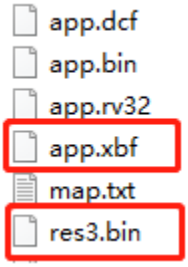
测试项	静置 1 分钟平均电流
双模未连接休眠	290
双模连接休眠	230
单 BLE 连接休眠(关 bt 可发现)	195

4 开发工具

4.1 Downloader 图像转换/烧录



3.1.1 内/外部资源烧录



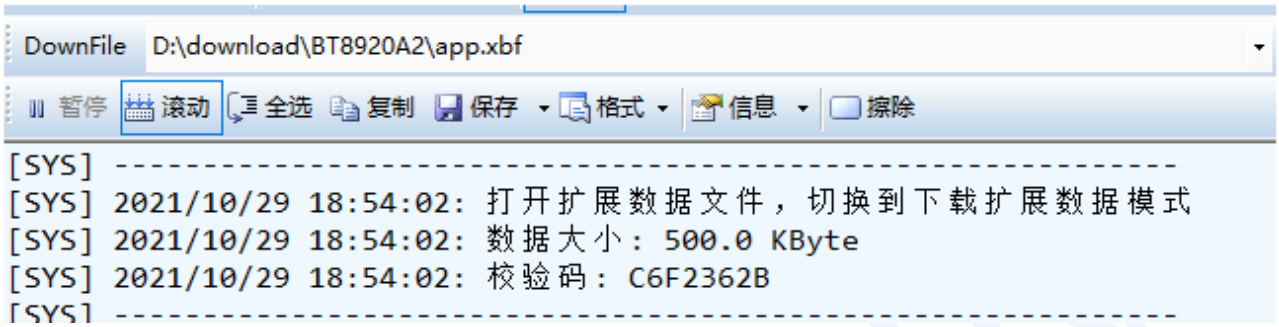
内部资源文件需要命名为:将 res3.bin 放置到 app.dcf 同级目录，编译后合入到 app.dcf 中。

外部资源文件需要命名为:app.xbf，然后放置到 app.dcf 同级目录下，downloader 选择“开始”- 勾选“扩展”。

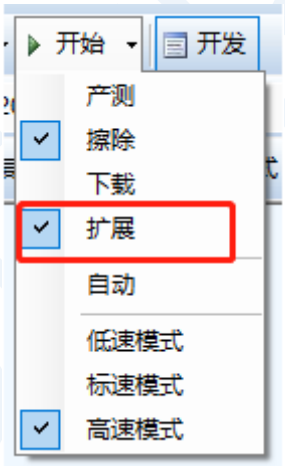
3.1.2 单独外部资源烧录

工具有 2 种方式支持单独烧录外部资源文件：

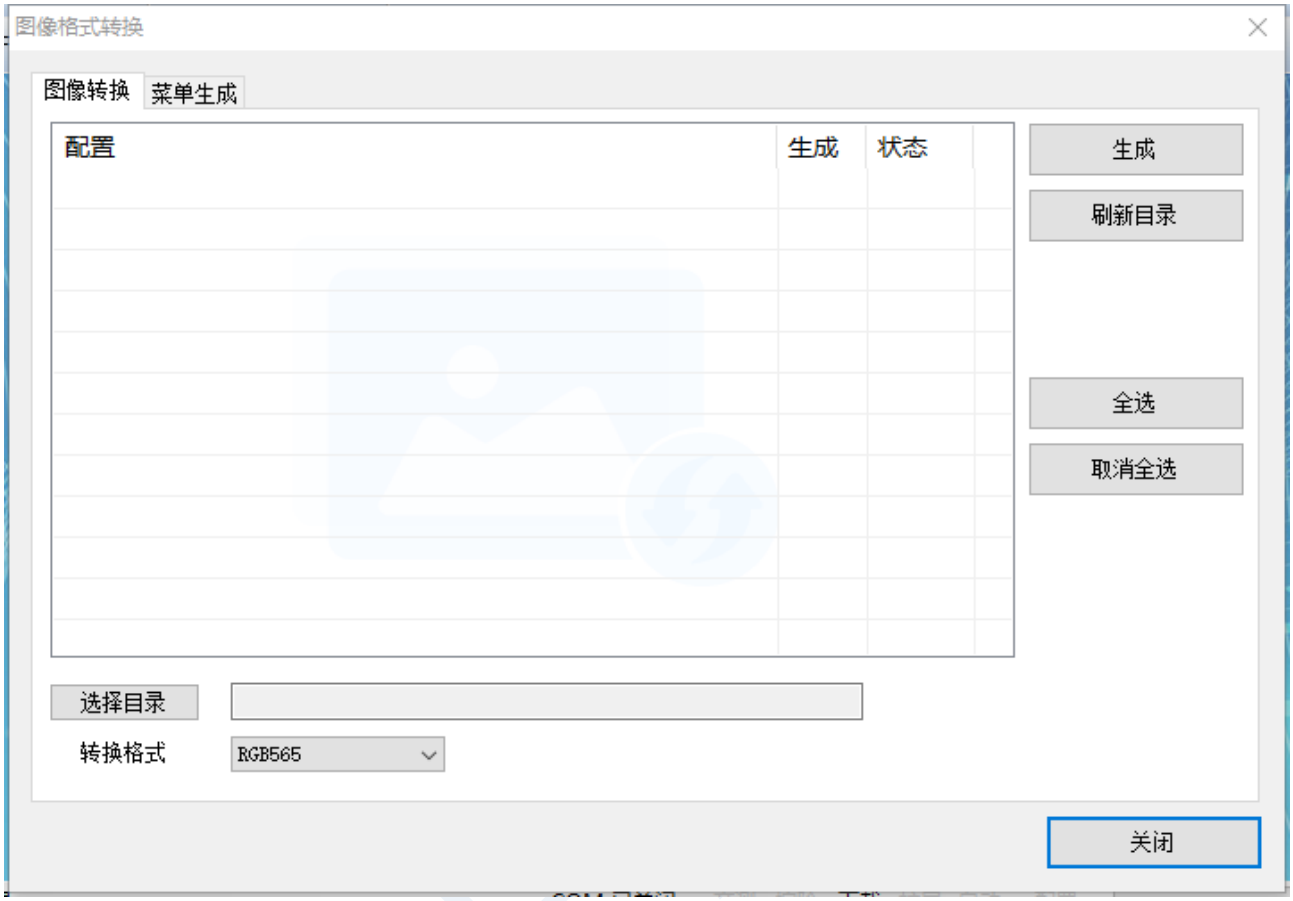
1、直接用 downloader 打开 xbf 文件，点击开始即可：



2、app.xbf，然后放置到 app.dcf 同级目录下，downloader 选择“开始”- 勾选“扩展”，反勾选“下载”，即可单独下载 xbf 文件：

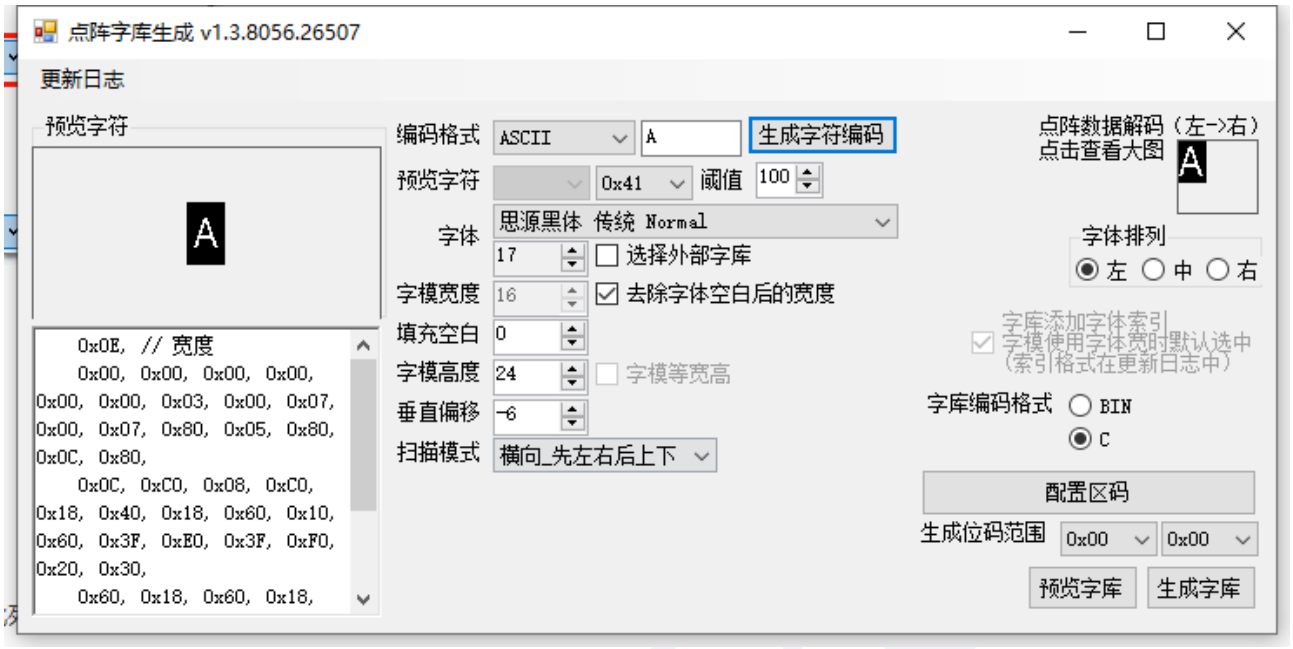


3.1.3 图像格式转换：



可以通过 downloader 工具-图像格式转换进入，转换后生成 bin 文件，然后放置到 SDK 中 app\projects\watch\Output\bin\ui 目录下，编译会生成 ui.h，里面找到对应的 bin 文件，在代码中使用图片资源即可

4.2 点阵字库生成器



可以通过此工具生成需要的字库 bin，然后放置到 SDK 中 app\projects\watch\Output\bin\ui 目录下，编译会生成 ui.h，里面找到对应的 bin 文件，在代码中使用字库资源即可

附 录