GH30x 驱动库移植指南

版本：**3.5**

发布日期：**2022-07-14**

深圳市汇顶科技股份有限公司

商标声明

GOODIX和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX 对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经 GOODIX 书面批准，不得将 GOODIX 的产品用作生命维持系统中的关键组件。在 GOODIX 知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦 B 座 13 层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档主要介绍如何使用 GH30x example 文件、驱动库文件，完成 GH30x 在对应平台的移植，以帮助开发者快速实现 GH30x 心率（HR）、血氧饱和度（SpO2）、心率变异性（HRV）和佩戴检测（NADT）等功能。

读者对象

本文适用于以下读者：

- GH30x 用户
- 软件开发工程师
- FAE 工程师

版本说明

本手册为第 7 次发布，对应的软件版本号为 V3.5。

修订记录

版本	日期	修订内容
1.0	2019-07-03	首次发布
2.0	2020-07-20	根据 example 重构文档
3.0	2021-04-29	重构文档，增加驱动 API 说明
3.1	2021-05-14	修改 MCU 本地测试部分，为 TestLib V1.1.0 后版本
3.2	2021-09-03	修改增加部分驱动库 API 说明
3.3	2021-09-23	修改部分驱动库 API 说明，增加平滑策略说明
3.4	2022-03-21	修正描述，删除多余描述，整体格式修改
3.5	2020-07-14	新增部分内容，更新部分 API 说明

目录

1 简介

GH30x 驱动库为 GH30x 系列芯片（如 GH300、GH301、GH3011、GH3018）的应用开发提供驱动支持，包含丰富的应用 API、配置文件、硬件驱动接口以及 lib 文件等，可帮助应用开发者快速集成心率、血氧及佩戴检测功能，缩短产品开发周期。

1.1 驱动库架构

如图 1-1 所示，GH30x 驱动库的应用软件架构可分为五层：

- 应用处理层（gh30x_process）：包括应用功能启动流程及中断处理流程，可供上层用户应用调用。
- 配置层：包括 gh30x_reg_array、gh30x_comm_pkg 和 gh30x_ctrl 等配置文件，用于配置 GH30x 芯片的寄存器、初始化参数、以及算法开关等。并且，应用处理层可通过配置层调用硬件驱动接口层的 API。
- 硬件驱动接口层（GH30x Port）：包括 GH30x 芯片的硬件驱动接口，用于访问、操作 GH30x 的硬件模块，如 IIC、GPIO、TIMER 等。
- 驱动库层（hbd_ctrl.lib&hbd_communicate.lib）：包含 GH30x 芯片的功能驱动程序及协议处理程序，并提供丰富的 API 接口，满足用户的应用需求。
- 算法调用层（gh30x_algo_hook）：包括算法调用的注册函数接口，用于回调算法，管理算法功能配置，内存分配，输入原始数据以及获取运算结果等。

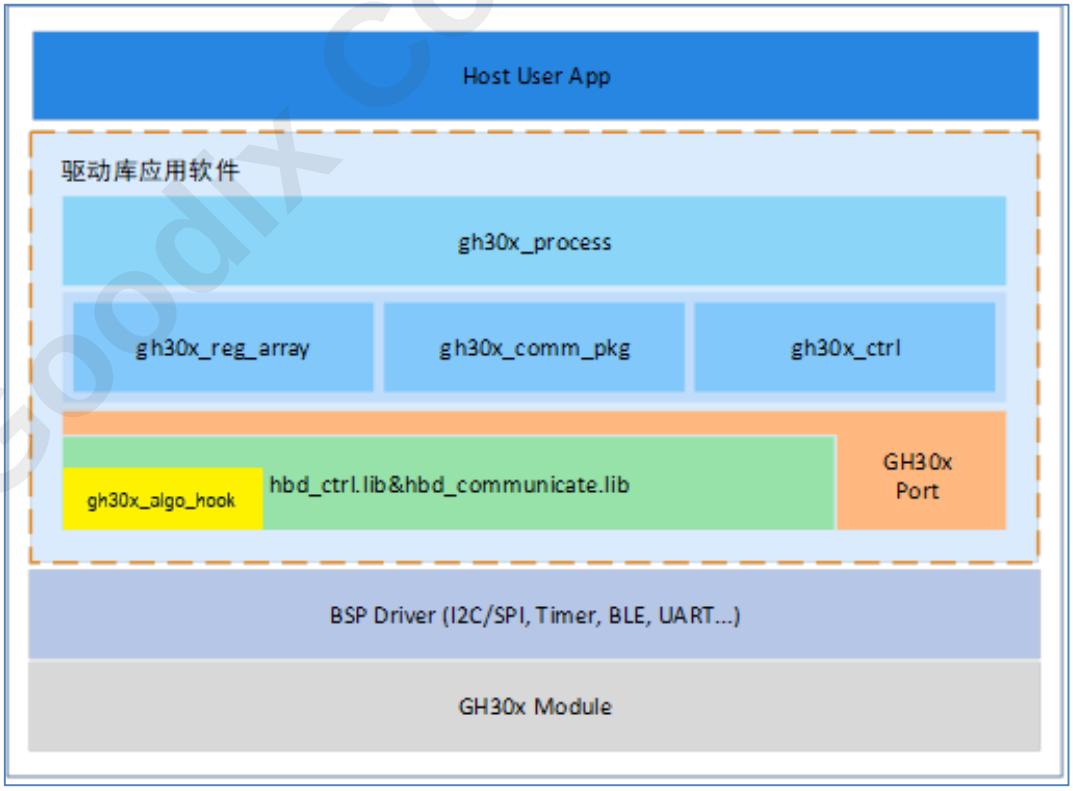


图 1-1 GH30x 驱动库应用软件架构

说明：

- 用户应用层（Host User App）：用户应用逻辑模块，对驱动库进行调用，获取驱动库返回的信息（如心率值、血氧值等），并对这些信息进一步处理。

- BSP 驱动层（BSP Driver）：用户平台的硬件接口驱动，可将功能接口注册到驱动库层，供驱动库操作 GH30x 芯片。
- GH30x 硬件层（GH30x Module）：GH30x 芯片的硬件模组。

1.2 驱动库应用流程

GH30x 驱动库应用流程主要包括三个子流程：上电初始化、G-sensor 中断处理、GH30x 模组中断处理。如下图所示：

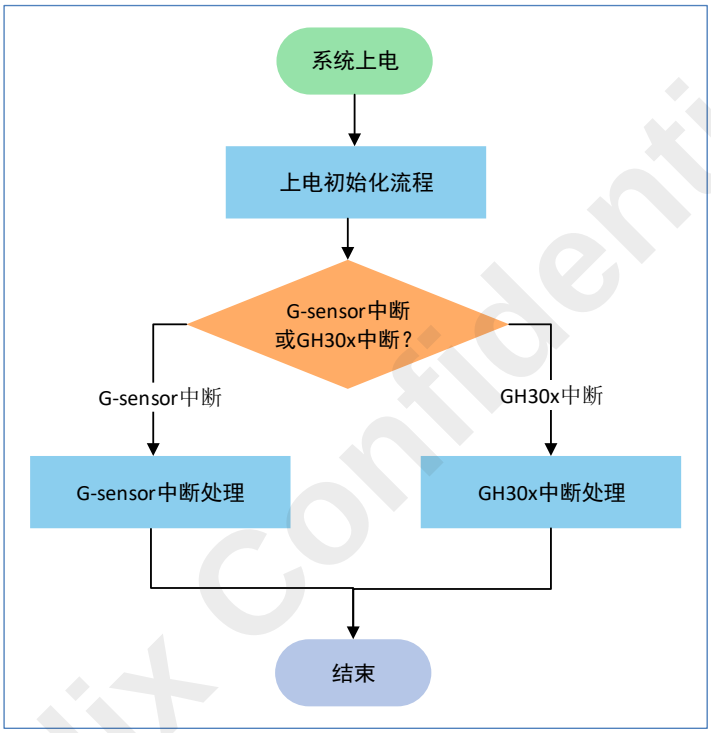


图 1-2 驱动库应用流程图

1. 上电初始化

系统上电后，先调用软件 init 接口先初始化 GH30x 模组，再初始化 G-sensor，具体流程如下图所示：

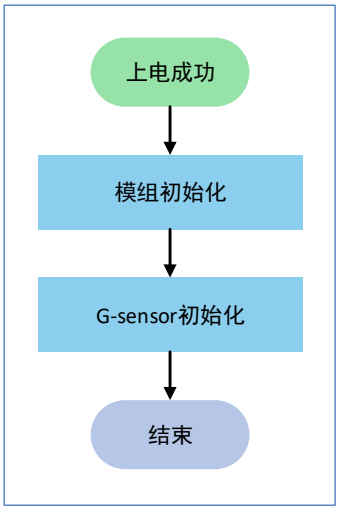


图 1-3 上电及初始化流程

2. G-sensor 中断处理

初始化完成后，可通过 G-sensor 中断处理流程读取 G-sensor 数据，进行动作检测，具体流程如下图所示：

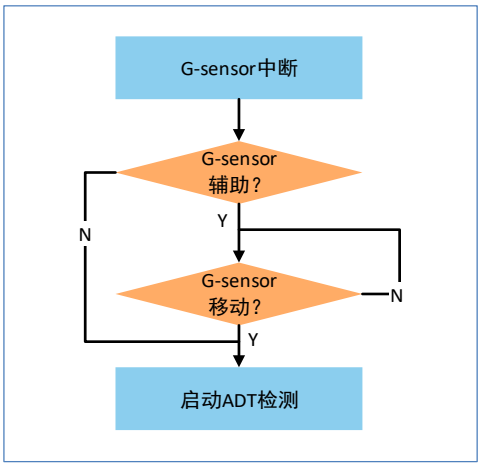


图 1-4 G-sensor 中断处理流程

3. GH30x 模组中断处理

完成 GH30x 模组的初始化、并调用对应的启动功能接口之后，通过 GH30x 模组中断处理流程，实现心率、血氧、佩戴检测功能，具体流程如下图所示：

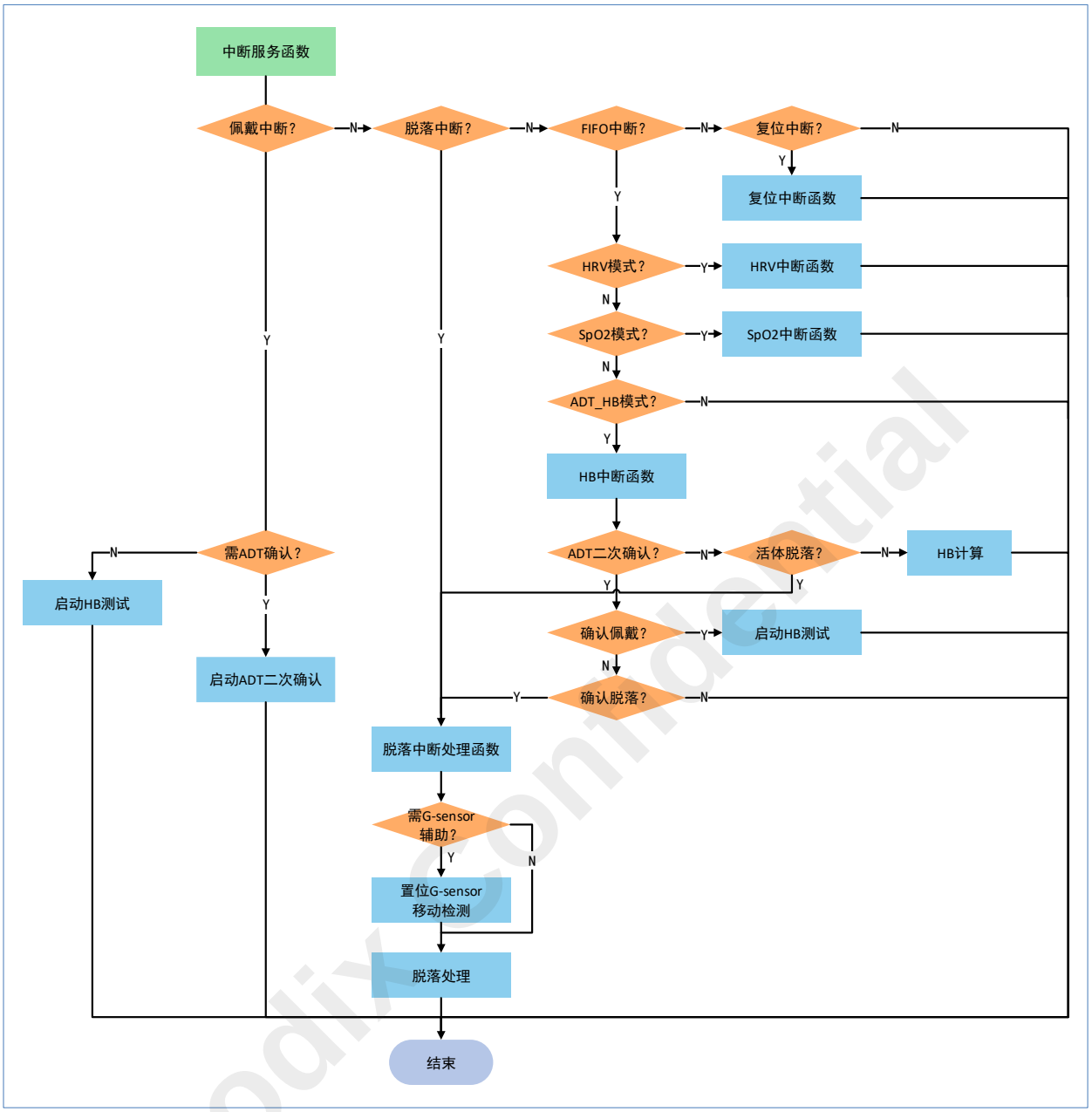
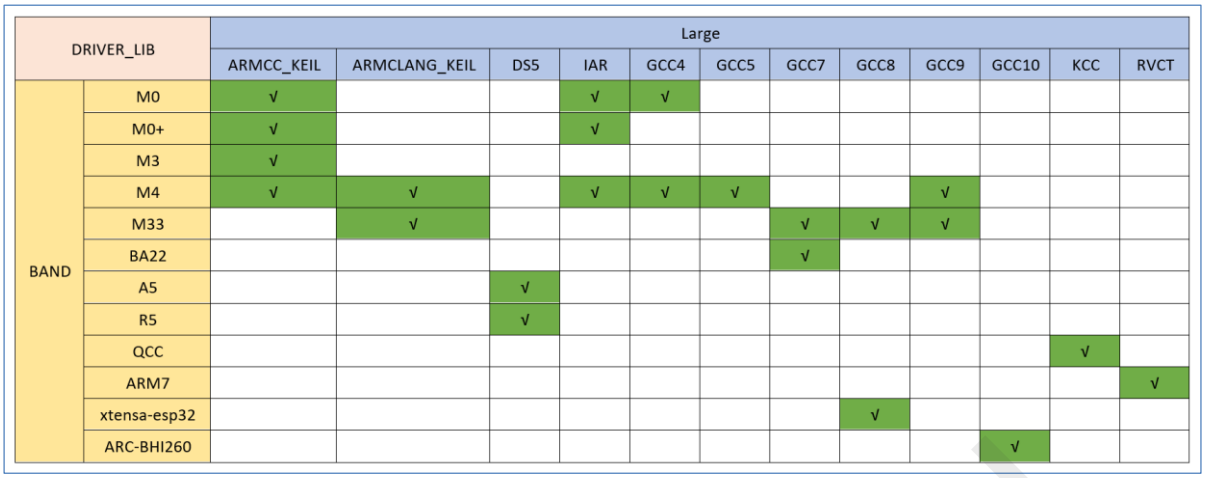


图 1-5 GH30x 模组中断处理流程

1.3 支持平台

GH30x 应用软件需配合 GH30x 驱动库使用，目前驱动库支持的控制器及编译器平台如下图所示：



DRIVER_LIB		Large										
		ARMCC_KEIL	ARMCLANG_KEIL	DS5	IAR	GCC4	GCC5	GCC7	GCC8	GCC9	GCC10	KCC	RVCT
BAND	M0	√			√	√							
	M0+	√			√								
	M3	√											
	M4	√	√		√	√	√			√			
	M33		√					√	√	√			
	BA22							√					
	A5			√									
	R5			√									
	QCC											√	
	ARM7												√
	xtensa-esp32								√				
	ARC-BHI260										√		

图 1-6 驱动库支持的控制器与编译器平台

如需支持其他平台，需要提供对应平台的编译工具链、工程编译选项等。

1.4 驱动库文件

GH30x 驱动库包含各种 example 头文件及源文件、库文件等，具体描述参见下表：

表 1-1 GH30x 驱动库文件说明

文件类型	文件名称	描述	备注
头文件（.h）	gh30x_example.h	对外接口声明头文件，提供给用户上层应用调用	
	gh30x_example_config.h	example 功能支持配置头文件	
	gh30x_example_common.h	examplee 公共头文件，包含 example 通用的宏和常量定义，以及函数声明等	用户可无需关注
	gh30x_algo_hook.h	算法调用的头文件，与算法头文件共同使用	
	goodix_mem.h	算法内存管理头文件	
	goodix_type.h	算法数据类型头文件	
	goodix_hba.h	心率算法头文件	
	goodix_spo2.h	血氧算法头文件	
	goodix_hrv.h	心率变异性算法头文件	
	hbd_communicate.h	Host 通信库的头文件	
	hbd_ctrl.h	驱动算法库的头文件	
	app_cmd_send.h	蓝牙（BLE）透传数据库的头文件	
	systemTest.h	Local Test 产测库的头文件	
源文件（.c）	gh30x_example_port.c	example 功能移植接口源文件，包含各接口函数的实现代码	用户需根据平台进行移植适配
	gh30x_example_reg_array.c	初始化配置以及不同模式的寄存器配置	
	gh30x_example_comm_pkg.c	BLE/UART 数据打包函数接口定义	用户无需关注
	gh30x_example_ctrl.c	控制函数接口定义	
	gh30x_example_process.c	应用流程控制函数定义	
	gh30x_algo_hook.c	算法调用函数定义，与算法库文件共同使用	
库文件（.lib）	hbd_communicate_KEIL5_M4-fp.lib	Host 通信库（以 Cortex-M4 内核为例）	必须包含
	hbd_ctrl_KEIL5_M4-fp.lib	驱动库（以 Cortex-M4 内核为例）	

文件类型	文件名称	描述	备注
	common_cortexM4_keil-armcc-5.06.lib	算法公共库	可选，用户可根据实际量产测试需求添加
	hba_cortexM4_keil-armcc-5.06.lib	心率算法库	
	spo2_cortexM4_keil-armcc-5.06.lib	血氧算法库	
	HRV_KEIL5_M4-fp.lib	心率变异性算法库	
	NADT_KEIL5_M4-fp.lib	活体检测算法库	
	TestLib.lib	Local Test 产测库	
	lib_app_cmd_send.lib	透传库	

一般情况下，移植 GH30x 驱动库时，用户只需关注这 4 个文件：*gh30x_example_config.h*、*gh30x_example.h*、*gh30x_example_port.c*、*gh30x_example_reg_array.c*。

1.4.1 gh30x_example_config.h

gh30x_example_config.h 文件为 example 功能配置头文件，包含各种应用功能、芯片配置、初始化、算法功能等配置参数的宏定义，具体描述参见下表：

表 1-2 *gh30x_example_config.h* 文件配置说明

宏定义	描述
__GH30X_COMMUNICATION_INTERFACE__	芯片 IIC/SPI 接口选择（二选一），需实现 *gh30x_example_port.c* 文件中对应接口函数
__GH30X_IRQ_PLUSE_WIDTH_CONFIG__	芯片中断脉冲宽度配置为 255 μs 使能
__PLATFORM_DELAY_US_CONFIG__	注册平台延时 μs 函数使能，需在 *gh30x_example_port.c* 实现平台 μs 延时函数 hal_gh30x_delay_us()
__GS_SENSITIVITY_CONFIG__	G-sensor 归一化精度配置，需与用户平台中的 G-sensor 精度相对应
__RETRY_MAX_CNT_CONFIG__	最大启动重试次数
__RESET_REINIT_CNT_CONFIG__	初始化重试次数
__ALGO_CALC_WITH_DBG_DATA__	计算接口输出 Debug 数据使能
__ALGO_CALC_DBG_BUFFER_LEN__	接收 Debug 数据的缓存 Buffer 长度
__FIFO_INT_TIMEOUT_CHECK__	FIFO 中断 Timeout 监控使能 用户平台可能会出现漏中断时必须使能。使能该功能后，用户需实现 *gh30x_example_port.c* 中的相关接口
__HB_DET_SUPPORT__	心率功能使能
__HB_FIFO_THR_CNT_CONFIG__	心率功能 FIFO 中断阈值配置
__HBA_ENABLE_WEARING__	心率算法中的佩戴检测算法使能
__HB_START_WITH_GSENSOR_MOTION__	使能 G-sensor 触发启动心率应用流程 使能后需支持 G-sensor 动作检测，可配置硬件 G-sensor 动作检测，或使用软件模拟动作检测
__HB_NEED_ADT_CONFIRM__	心率功能流程 ADT 确认流程使能 如果使能该配置，ADT 检测到佩戴后，软件算法会再次确认是否为佩戴
__HB_ADT_CONFIRM_GS_AMP__	ADT 确认流程中的 G-sensor 幅度阈值配置
__HB_ADT_CONFIRM_CHECK_CNT__	ADT 确认流程中的总计算点数配置
__HB_ADT_CONFIRM_THR_CNT__	ADT 确认流程中的超过点数阈值配置

宏定义	描述
__SPO2_DET_SUPPORT__	血氧功能使能
__SPO2_FIFO_THR_CNT_CONFIG__	血氧功能 FIFO 阈值配置
__SPO2_GET_ABN_STATE__	血氧出值异常提示
__HRV_DET_SUPPORT__	HRV 功能使能
__HRV_FIFO_THR_CNT_CONFIG__	HRV 功能 FIFO 阈值配置
__USE_GOODIX_APP__	GHealth App 对接支持使能。若要上传 MCU 模式的 Debug 数据，则需开启 __ALGO_CALC_WITH_DBG_DATA__
__NEW_DATA_MULTI_PKG_NUM__	BLE 单次发送多包数据数量配置 配置多包发送支持时，需要用户平台的 BLE 支持长包发送
__BLE_PKG_SIZE_MAX__	BLE 发送的最长包的长度配置 该参数一般对齐 BLE MTU，并受限于手机支持的数据包长度
__BLE_MCU_PKG_BUFFER_MAX_LEN__	BLE 发送的 MCU 模式数据的 Buffer 长度 Buffer 长度需大于以下公式的计算值： (DBG_MCU_PKG_RAW_FRAME_LEN * __ALGO_CALC_DBG_BUFFER_LEN__) + MCU_PKG_SPO2_ALGO_RESULT_LEN
__UART_WITH_GOODIX_TOOLS__	UART 对应的 Goodix 量产测试工具使能
__SYSTEM_TEST_SUPPORT__	Local Mode 测试模式使能
__SYSTEM_TEST_DATA_CNT_CONFIG__	Local Mode 测试数据点数配置
__EXAMPLE_DEBUG_LOG_LVL__	调试信息等级配置
__EXAMPLE_LOG_DEBUG_SUP_LEN__	调试信息的最大字符串长度
__HR_SPO2_DET_SUPPORT__	红外动态心率与血氧功能使能
__GET_RAWDATA_WITHOUT_ALGO_HANDLE	仅获取 rawdata，应用于 hb_only 于 hr_spo2 模式
_GSENSOR_ASYN_HANDLE_ENABLE_	Gsensor 数据异步对齐功能，仅用于 GH30X 与 Gsensor 分离的场景
__HBD_DOUBLE_CORE_ENABLE__	应用于双核运行，驱动与算法分离使能
__FACTORY_DET_SUPPORT__	测试模式使能
__ADT_DET_SUPPORT__	ADT 模式使能
__HBD_HB_ALGORITHM_ENABLE__	心率算法使能
__HBD_HRV_ALGORITHM_ENABLE__	心率变异性算法使能
__HBD_SPO2_ALGORITHM_ENABLE__	血氧算法使能
__HBD_NADT_ALGORITHM_ENABLE__	活体检测算法使能
__HBD_ALGORITHM_USE_DYN_MEM__	动态分配算法内存使能

1.4.2 gh30x_example.h

gh30x_example.h 文件为 example 应用的头文件，包含应用功能启动与停止、初始化等对外的函数接口，具体描述参见下表：

表 1-3 *gh30x_example.h* 文件说明

宏定义/函数声明	描述
gh30x_module_init	GH30x 模组初始化函数，上电/重新上电时必须调用
gh30x_module_start	GH30x 模组启动函数，根据模式启动 启动模式定义参见 EMGh30xRunMode
gh30x_module_start_without_adt	GH30x 模组启动函数，根据模式启动 启动模式定义参见 EMGh30xRunMode，调用接口时不会进行硬件 ADT 检测

宏定义/函数声明	描述
gh30x_module_stop	GH30x 模块停止函数
ble_module_recv_data_via_gdcs	BLE 接收数据处理函数，在用户平台接收 GHeath App 数据的函数中调用
uart_module_recv_data	UART 接收处理函数，在用户平台的 Goodix 量产工具 UART 数据接收函数中调用
gh30x_module_system_test_otp_check	系统测试 OTP 函数
gh30x_module_system_test_os_start	系统测试 OS 函数
gh30x_check_mem_buf_enable	算法内存校验使能函数

1.4.3 gh30x_example_port.c

gh30x_example_port.c 文件包括 GH30x 芯片的硬件驱动接口，用于访问、操作 GH30x 的硬件模块，具体描述参见下表：

表 1-4 *gh30x_example_port.c* 文件说明

函数类型	函数名称	描述	备注
GH30x 通信接口函数	hal_gh30x_i2c_init	初始化 IIC 模块	IIC 与 SPI 通信方式二选一，且这些函数要求用户必须实现 **说明：** IIC、SPI 数据格式要求，请参考对应的 GH30x 数据手册
	hal_gh30x_i2c_write	IIC 写操作	
	hal_gh30x_i2c_read	IIC 读操作	
	void hal_gh30x_spi_init	初始化 SPI 模块	
	uint8_t hal_gh30x_spi_write	SPI 写操作	
	uint8_t hal_gh30x_spi_read	SPI 读操作	
	void hal_gh30x_spi_cs_set_low	设置 SPI 片选信号为低电平	
	void hal_gh30x_spi_cs_set_high	设置 SPI 片选信号为高电平	
G-sensor 驱动函数	gsensor_drv_init	G-sensor 初始化函数	用户可根据应用需求实现
	gsensor_drv_enter_normal_mode	G-sensor 配置为普通模式，可直接读取 x、y、z 寄存器获取数据	
	gsensor_drv_enter_fifo_mode	G-sensor 配置为 FIFO 模式，可通过 G-sensor FIFO 获取一组 x、y、z 数据	
	gsensor_drv_enter_motion_det_mode	G-sensor 配置为动作检测模式，检测动作输出中断	
	gsensor_drv_get_fifo_data	从 G-sensor FIFO 获取最新一组 x、y、z 数据，并缓存到 G-sensor 的软件 Buffer 中，缓存到软件 Buffer 中的数据对应为 GH30x 产生中断后的前 1s 数据	
	gsensor_drv_get_data	从 G-sensor 中获取 x、y、z 数据，获取的数据为最新一帧数据	
	gsensor_drv_int1_handler	G-sensor 中断处理函数	
延时函数	hal_gh30x_delay_us	μs 级延时函数	用户必须实现，且用户平台 SPI 需根据 Goodix 要求实现
用户平台 IO 中断初始化函数	hal_gh30x_int_init	用户平台与 GH30x INT 管脚连接的 IO 的外部中断初始化函数	GH30x 中断必须实现，G-sensor 中断

函数类型	函数名称	描述	备注
	hal_gsensor_int1_init	用户平台与 G-sensor GINT_FIFO 管脚连接的 IO 的外部中断初始化函数	可根据应用需求实现
用户应用处理算法结果与事件函数	handle_hb_mode_result	用户应用处理心率算法结果函数	用户必须实现，且根据使能的模式实现
	handle_spo2_mode_result	用户应用处理血氧算法结果函数	
	handle_hrv_mode_result	用户应用处理心率变异性算法结果函数	
	handle_wear_status_result	用户应用处理佩戴事件结果函数	
	handle_system_test_otp_check_result	用户应用处理量产测试 OTP 测试结果函数	
	handle_system_test_os_result	用户应用处理量产测试 OS 测试结果函数	
BLE 通信函数	ble_module_send_heartrate	通过标准心率服务发送心率值	对接用户应用心率 App 时实现
	ble_module_add_rr_interval	添加 RR 间期值到标准心率服务	
	ble_module_send_data_via_gdcs	通过自定义的 GHealth App 服务发送数据	用于对接 GHealth App，用户必须实现 **说明：** 关于自定义服务，可参考 2.3.2.1 新增蓝牙服务
定时器函数	ble_module_repeat_send_timer_start	BLE 重复发送数据定时器启动函数	BLE 重复发送数据定时器，需发送 MCU 数据时实现。定时间隔取决于 BLE 稳定性，推荐 50 ~ 100 ms
	ble_module_repeat_send_timer_stop	BLE 重复发送数据定时器停止函数	
	ble_module_repeat_send_timer_init	BLE 重复发送数据定时器初始化函数	
	hal_gh30x_fifo_int_timeout_timer_start	FIFO 中断函数监控定时器启动函数	FIFO 中断函数监控定时器，用户必须实现。 定时间隔为 FIFO 中断间隔加 80 ms@25 Hz 、 20 ms@100 Hz
	hal_gh30x_fifo_int_timeout_timer_stop	FIFO 中断函数监控定时器停止函数	
	hal_gh30x_fifo_int_timeout_timer_init	FIFO 中断函数监控定时器初始化函数	
UART 通信函数	uart_module_send_data	用户平台 UART 发送数据	对接 Goodix 量产工具/Dongle 工具时需实现
通信命令处理函数	handle_goodix_communicate_cmd	处理 Goodix 工具通信命令	处理 Goodix 工具命令时需实现
日志函数	example_dbg_log	用户平台输出 Log 函数	用户必须实现

1.4.4 gh30x_example_reg_array.c

gh30x_example_reg_array.c 包含各应用功能的配置数组。

表 1-5 *gh30x_example_reg_array.c* 文件说明

数组名	描述
hb_adt_confirm_reg_config	ADT 二次佩戴确认功能的配置

数组名	描述
hb_reg_config_array	心率功能的配置
spo2_reg_config_array	血氧功能的配置
hrv_reg_config_array	心率变异性功能的配置

2 例程移植说明

本章以 GH3011 为例介绍例程移植，其它型号（如 GH300、GH301、GH3018）可根据实际情况参考。

2.1 基本功能移植

本节以下列主控平台为例，介绍 GH30x 驱动库基本功能（如 BLE 透传功能）的移植，以帮助用户快速入门驱动库移植的操作流程。

- 控制器内核：Cortex-M4
- 集成开发环境：Keil

说明：

更多应用功能移植或调试，请参考 2.2 应用方案介绍和 2.3 应用调试。

2.1.1 移植驱动库到工程

拷贝 GH30x 驱动库文件至用户应用示例工程目录下，再打开 Keil 示例工程，若在“sensor_ctrl”group 下查看到添加的驱动库文件（如图 2-1 所示），即表示驱动库成功移植到工程中。

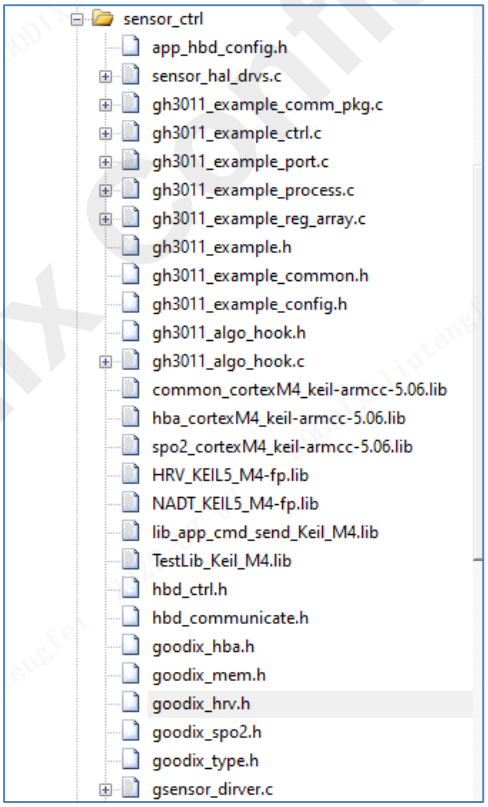


图 2-1 算法库移植到 keil 工程图

说明：

- 不同编译环境下的 *hbd_communicate_KEIL5_M4-fp.lib*、*hbd_ctrl_KEIL5_M4-fp.lib*、*common_cortexM4_keil-armcc-5.06.lib*、*hba_cortexM4_keil-armcc-5.06.lib*、*spo2_cortexM4_keil-armcc-5.06.lib*、*HRV_KEIL5_M4-fp.lib*、*NADT_KEIL5_M4-fp.lib*、*lib_app_cmd_send_keil_cortexM4.lib*、*TestLib.lib*（如需产测功能）以及库对应头文件的添加方法可能有差异。
- Goodix 提供了一个驱动库示例工程，可方便用户移植时参考。

2.1.2 调通通信功能

用户可按照如下步骤，调通主控平台与 GH30x 模组间的通信功能：

1. 根据平台实现精准的 μs 级延时函数。

GH30x 的上电时序（如图 2-2 所示）明确地规定了上电后需要完成的初始化工作及精准的 μs 级延时。在 *gh30x_example_port.c* 文件中，用户需实现精准的 μs 级延时函数 hal_gh30x_delay_us()，具体实现方法可能因平台差异稍有不同。

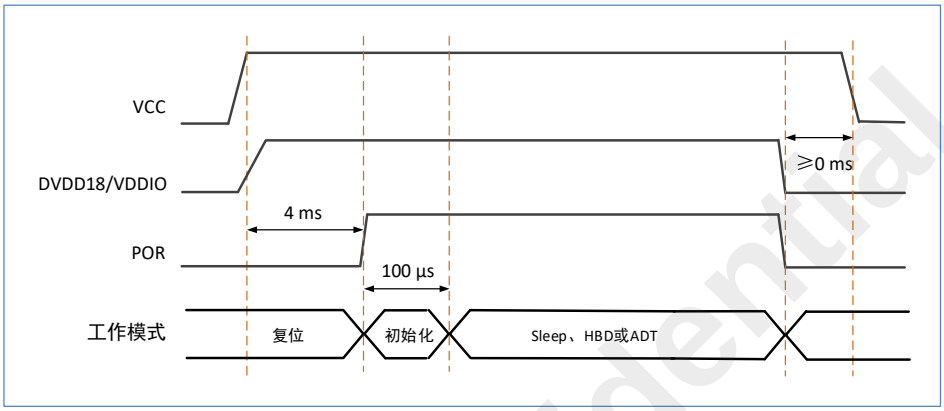


图 2-2 GH30x 上电时序图

2. 设置 IIC 通信。

(1) 设置 IIC 管脚：将 IIC_EN 管脚配置为 IIC，并设置内部上拉（若外部硬件增加了上拉，可无需使用内部上拉）、IIC 通信速率以及 IIC 数据传输方式等。

(2) 实现 IIC 读写函数：在 *gh30x_example_port.c* 文件中，实现 IIC 写函数 hal_gh30x_i2c_write()和 IIC 读函数 hal_gh30x_i2c_read()。IIC 读函数需先利用 IIC 写函数写入一个数据，再进行读操作。对 GH30x 的所有操作均通过 IIC 读写函数实现，因此，需保证这两个函数的准确性。

(3) 验证 IIC 通信：在应用工程中调用 gh30x_module_init()函数，并利用 UART 打印信息验证 IIC 通信是否正常。

- 如果 IIC 读写函数不正常或者 IIC 管脚设置不正确，UART 会打印出类似“gh30x init error[*]”的 Log 信息。
- 如果 IIC 读写正常并且 IIC 管脚设置正确，UART 会打印出类似“gh30x module init ok”的 Log 信息。
- 直接读取寄存器 0x0028 的值，判断是否为 0x31。若寄存器值为 0x31，则表示 IIC 通信正常。

2.1.3 修改配置文件

如果用户应用是参照 Goodix 参考设计推荐值（参考对应的 GH30x 应用设计指南）进行设计的，可直接使用默认的配置文件。否则，需根据实际的应用设计及配置，修改配置文件 *gh30x_example_reg_array.c*，具体操作步骤如下：

1. 使用 EVK PC Tool 生成配置文件，如下图所示：

图 2-3 EVK 工具生成的配置文件图

2. 按照下表所示的对应关系，将 EVK PC Tool 生成的配置文件中的数组内容替换到 *gh30x_example_reg_array.c* 文件中对应的驱动配置数组中。

表 2-1 EVK PC Tool 生成的配置文件与驱动配置的对应关系

配置文件	驱动配置（*gh30x_example_reg_array.c*）
_HRV_20220322200857.conf	hrv_reg_config_array
_穿戴确认 ADTConfirm_20220322200857.conf	hb_adt_confirm_reg_config
_血氧 SPO2_20220322200857.conf	spo2_reg_config_array
_心率 HB_20220322200857.conf	hb_reg_config_array

2.1.4 添加用户代码

为实现 INT 中断、BLE 透传功能，用户需在相应的函数中添加用户代码。

- INT 中断（设置 INT 为中断管脚）

在 *gh30x_example_port.c* 文件的 INT 管脚的中断处理函数 hal_gh30x_int_handler()中，调用 gh30x_int_msg_handler()函数进行 INT 管脚的中断处理，即可通过算法计算输出心率值。

```
// gh30x int handler
void hal_gh30x_int_handler(void)
{
    gh30x_int_msg_handler();
}
```

提示：

gh30x_int_msg_handler()的实现示例代码，位于 *gh30x_example_comm_pkg.c* 文件。

- BLE 数据透传

实现 BLE 透传，用户需要完成两部分工作：添加服务和实现 Timer 定时器函数。

  - 添加服务

在主控平台（BLE 设备）的 BLE Profile 中，按如下配置新建一个服务，以实现 BLE 设备与 GHealth App 的蓝牙通信。

    - 服务 UUID："0000190e-0000-1000-8000-00805f9b34fb"

- TX_UUID："00000003-0000-1000-8000-00805f9b34fb"

数据 Tx Characteristic UUID，BLE 连接成功后，等待接收 BLE 设备端的数据。

- RX_UUID："00000004-0000-1000-8000-00805f9b34fb"

数据 Rx Characteristic UUID，BLE 连接成功后，可发送数据到 BLE 设备端。

服务添加后，需检查设备所有 UUID，包括 service UUID 下面的特性，不能与 Goodix UUID 重复，并需在 ble_module_recv_data_via_gdcs()函数中调用 gh30x_app_cmd_parse()函数，用于处理接收的 BLE 数据。

```
/// recv data via through profile
void ble_module_recv_data_via_gdcs(uint8_t *data, uint8_t length)
{
    gh30x_app_cmd_parse(data, length);
}
```

提示：

gh30x_app_cmd_parse()的实现示例代码，位于 *gh30x_example_comm_pkg.c* 文件。

用户还需在 ble_module_send_data_via_gdcs()函数中添加用户代码，以将心率数据通过 BLE 发送到手机端。

```
// send string via through profile
uint8_t ble_module_send_data_via_gdcs(uint8_t string[], uint8_t length)
{
    uint8_t ret = GH30X_EXAMPLE_OK_VAL;
    // code implement by user
    gus_tx_data_send(0, string, length);
    return ret;
}
```

◦ 实现 Timber 定时器函数

MCU 发送数据时，需实现 BLE 重复发送数据的定时器，定时间隔取决于 BLE 稳定性，推荐设置为 50 ms ~ 100 ms。

用户需在 *gh30x_example_port.c* 文件中，实现下列 Timer 函数：

- ble_module_repeat_send_timer_start：BLE 重复发送数据定时器启动函数。
- ble_module_repeat_send_timer_stop：BLE 重复发送数据定时器停止函数。
- ble_module_repeat_send_timer_init：BLE 重复发送数据定时器初始化函数。

2.1.5 验证测试

上述操作完成后，用户可按以下步骤验证驱动库移植是否成功：

1. 编译示例工程，并将生成的应用固件文件下载至 BLE 设备。
2. 重启设备，运行程序。
3. 在手机端，使用蓝牙调试 App（GRToolbox）扫描并连接设备，查看是否发现添加的 GH30x 服务与

特征。具体操作，可参考 2.3.2.5 验证蓝牙透传功能。

4. 蓝牙透传功能验证成功后，使用应用 App（GHealth App） 连接设备，开启心率测试功能。如果在 App 界面查看到心率波形图、心率值，即表示 GH30x 驱动库移植成功。

说明：

- Andriod 手机：支持蓝牙 4.0 及以上版本。
- 主控设备：支持蓝牙 4.0 及以上版本。
- 蓝牙调试 App：可使用 Goodix 提供的 GRToolbox App （Andriod），下载地址：https://product.goodix.com/zh/product/category/ble。
- 应用 App：Goodix 提供的 GHealth App，可联系 Goodix 技术支持或代理商获取。

2.2 应用方案介绍

2.2.1 应用框图

GH3011 需要与蓝牙主板上的 MCU 进行数据交互，并搭配 G-sensor 使用（G-sensor 用于辅助心率检测算法，心率算法运行在主控 MCU 上）。绿光 LED 检测心率，红光 LED 加红外 LED 实现血氧检测，可选配红外 LED 进行佩戴检测。

GH3011 与 MCU 的通信方式可以选择 SPI 或 IIC 通信。另外，GH3011 提供一路 INT 中断信号，用于提示心率数据 Ready 和其他状态。在供电上，需要外部提供 VLED 及 VCC 供电电源。

GH3011 有 3 个 LED 驱动电流通道，同一个驱动通道上可并联多个 LED，型号必须相同，驱动通道的选择需要满足表 2-2 所示要求。

表 2-2 GH3011 LED 电流驱动通道选择

驱动通道	“血氧+心率”应用		“心率”应用	
	LED 光源	分配功能	LED 光源	分配功能
LED_DRV0	红外光	佩戴检测、血氧、夜间心率检测	红外光	佩戴检测、夜间心率检测
LED_DRV1	红光	血氧	绿光	心率
LED_DRV2	绿光	心率	绿光	

GH3011 典型应用的系统框图，如图 2-4（“血氧+心率”应用）和图 2-5（“心率”应用）所示。

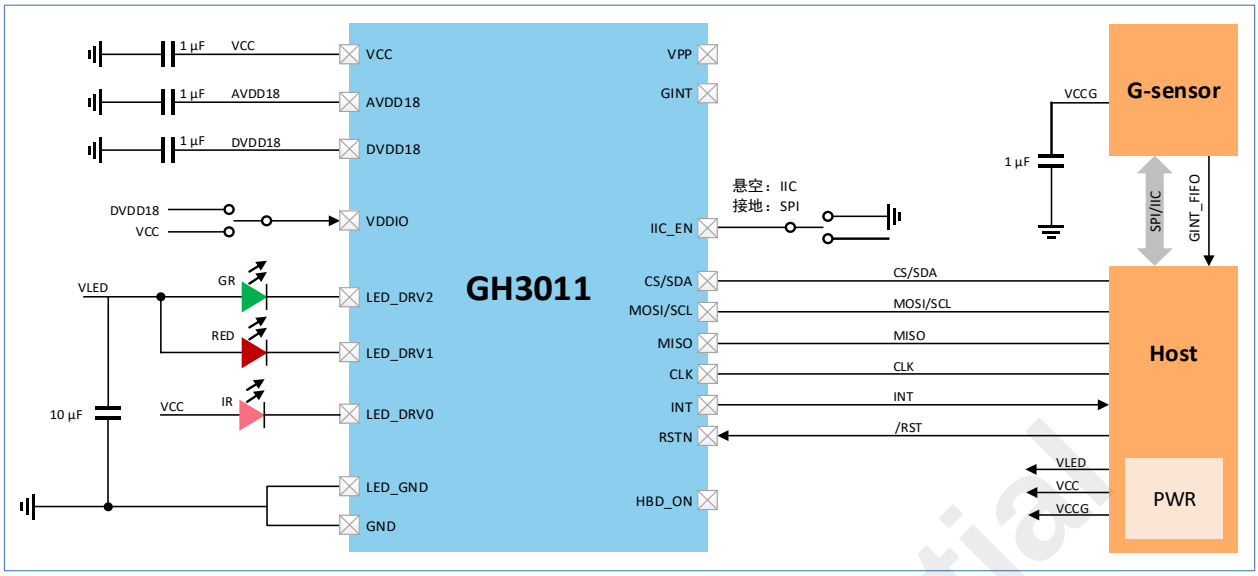


图 2-4 GH3011 典型心率及血氧检测应用方案系统框图

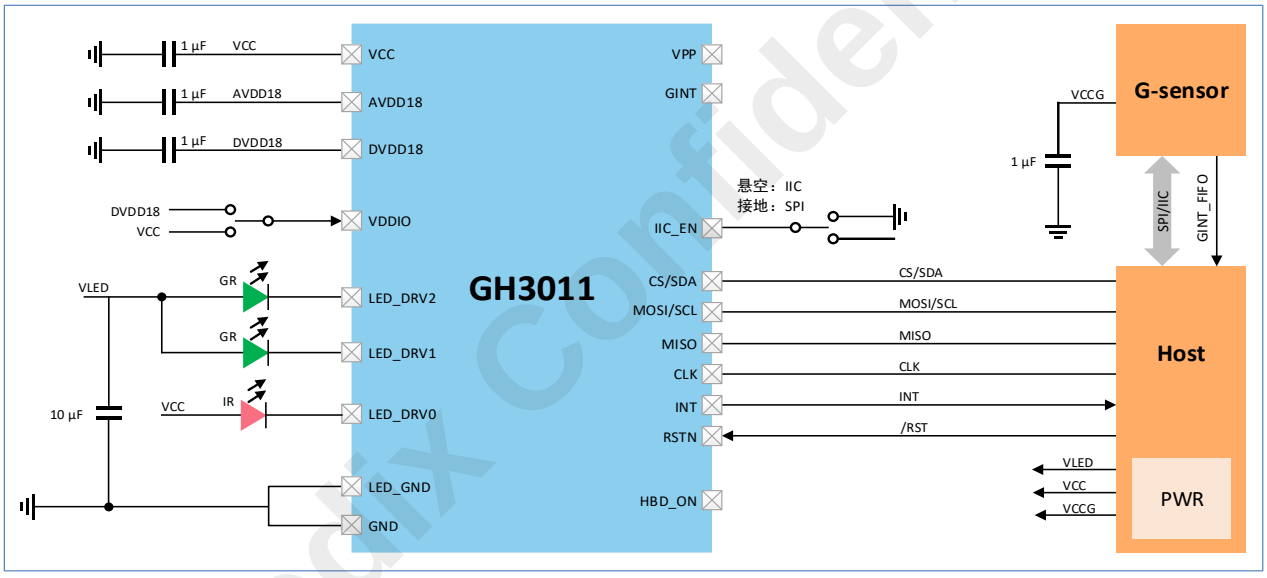


图 2-5 GH3011 典型心率检测应用方案系统框图（不支持血氧）

2.2.2 佩戴模块

佩戴检测包括 4 个阶段：G-sensor 移动侦测、红外 ADT 检测、红外+G-sensor 二次确认以及 HR-PPG 活体脱落检测。

1. G-sensor 移动侦测（可选项）

用户可根据实际应用，选择是否启用 G-sensor 移动侦测功能。若启用该功能，则需执行以下操作：

(1) 使能 G-sensor 移动侦测功能

在 *gh30x_example_config.h* 中，设置宏定义__HB_START_WITH_GSENSOR_MOTION__为 1，使能 G-sensor 移动侦测功能。

(2) 实现 G-sensor 移动侦测功能

在 *gh30x_example_port.c* 中，实现 G-sensor API 函数 gsensor_drv_enter_motion_det_mode()，可选择硬件或软件方式。

- 硬件方式

利用 G-sensor 中断函数 hal_gsensor_drv_int1_handler()来完成。例如，G-sensor 每 20 ms~40 ms 产生一个中断，只需计算相邻两个中断的 G-sensor 数据，其计算公式如下：

$$|\sqrt{x1^2+y1^2+z1^2}-\sqrt{x2^2+y2^2+z2^2}| \geq 30 \quad (0.05g\ @512Lsb/g)$$

其中，x1、y1、z1 为上一个中断的 ACC；x2、y2、z2 为当前中断的 ACC。

当计算的差值大于等于 30 时，即可调用函数 gh30x_start_adt_with_mode()，进入佩戴检测第 2 阶段。

- 软件方式

当 G-sensor 无法在几十毫秒内执行中断函数时，需增加一个软件定时器（定时时间为 30 ms）实现移动侦测。每 30 ms 读取一次 G-sensor 的实时数据，再将连续两次的数据按照上面的公式计算，满足差值大于等于 30 要求时，即可调用函数 gh30x_start_adt_with_mode()，进入佩戴检测第 2 阶段。

G-sensor 移动侦测的具体流程如下：

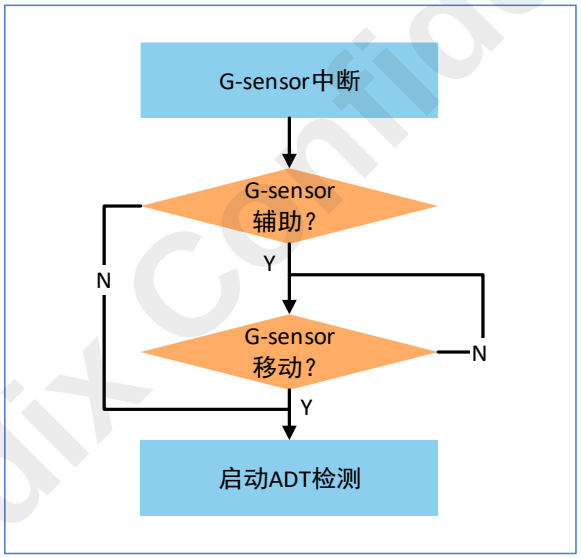


图 2-6 G-sensor 移动侦测流程

2. 红外 ADT 检测

会调用 API gh30x_start_func_with_mode()函数中的 gh30x_adt_confirm_start()函数，启动 ADT 二次确认。可通过配置宏定义__HB_NEED_ADT_CONFIRM__决定是否需启动第 3 阶段的红外+G-sensor 二次确认。

红外 ADT 检测的具体流程如下：

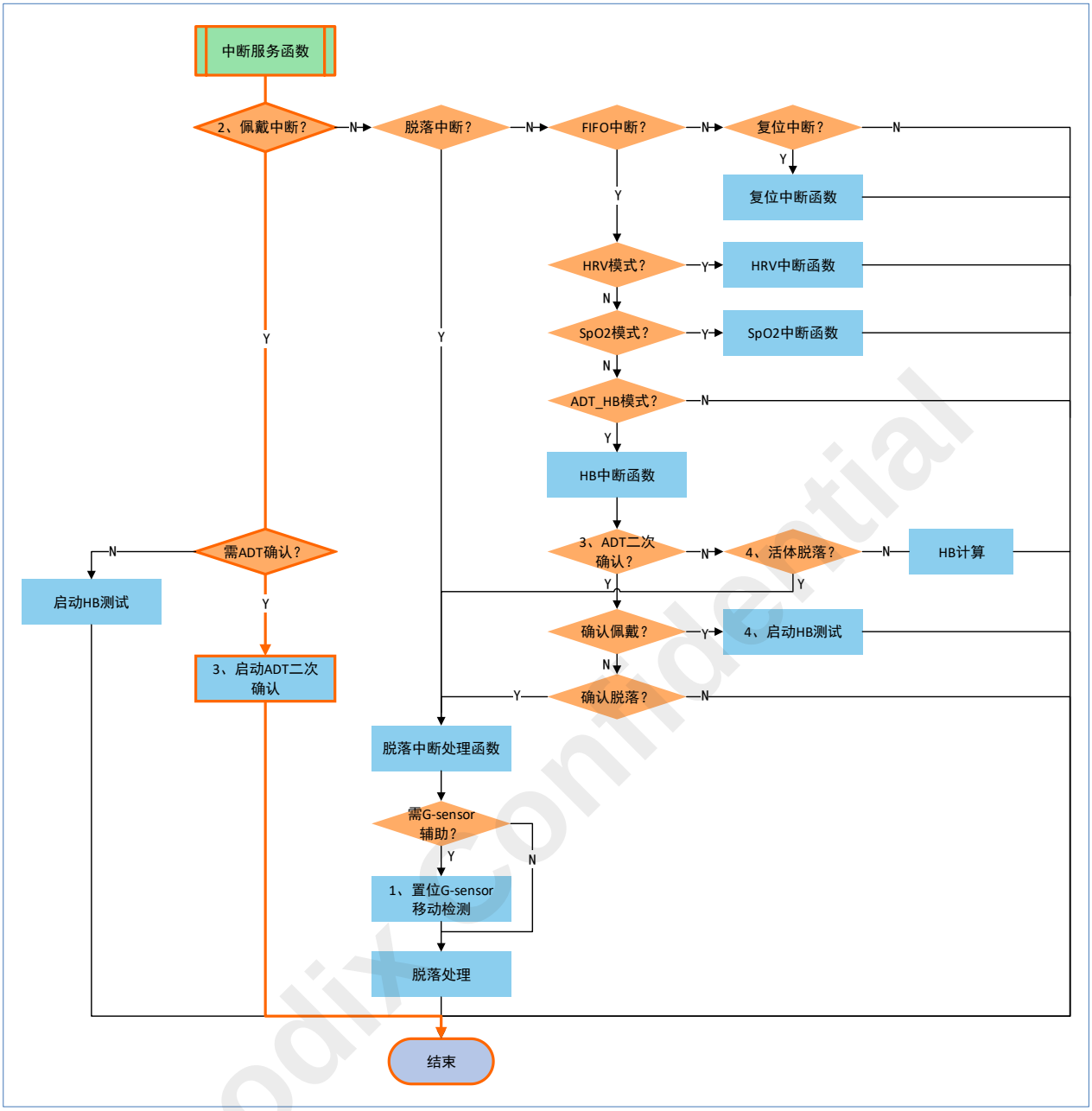


图 2-7 红外 ADT 检测流程

3. 红外+G-sensor 二次确认（可选项）

如果宏定义__HB_NEED_ADT_CONFIRM__配置为 1，则在第 2 阶段调用 gh30x_adt_confirm_start()函数后的 FIFO 中断中处理 ADT 二次确认的逻辑。在 FIFO 中断函数中，根据函数 HBD_AdtConfirmCalculateByFifoIntDbgOutputData()的返回值判断是否进入第 4 阶段，或者进入脱落状态。

该阶段处理的主要原理为：利用红外 PPG 和 G-sensor 的数据进行判断。当红外 PPG 超过一定阈值，并且 G-sensor 的抖动数据个数超过设定的阈值时，则认为人体在进行佩戴的动作，即可调用 gh30x_hb_start()进入第 4 阶段。此时，可基本确认佩戴已经成功，第 4 阶段的任务则是确认是否为活体。

红外+G-sensor 二次确认的具体流程如下：

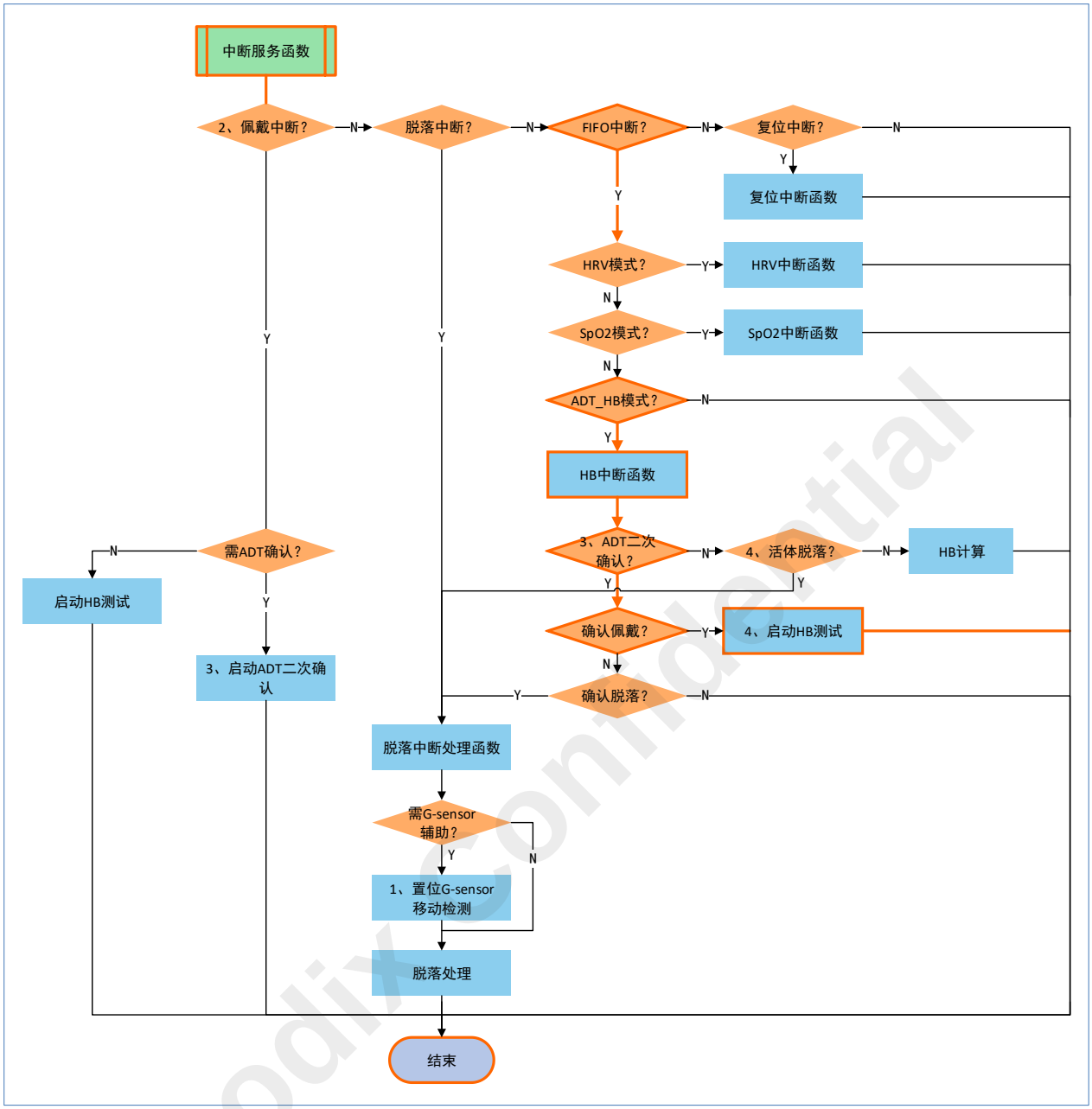


图 2-8 红外+G-sensor 二次确认流程

4. HR-PPG 活体脱落检测

调用 gh30x_hb_start()后，在 FIFO 中断函数中，根据接口 HBD_HbCalculateByFifoIntEx ()的返回参数 wearing_state 的值判断是否为活体脱落。若不是活体脱落，则继续心率测试并输出心率值；若是活体脱落，则进入脱落处理（返回第 1 阶段，重新判断佩戴）。

说明：

目前，仅心率测试模式支持活体检测功能，且该功能在整个心率计算过程中一直有效。

HR-PPG 活体脱落检测的具体流程如下：

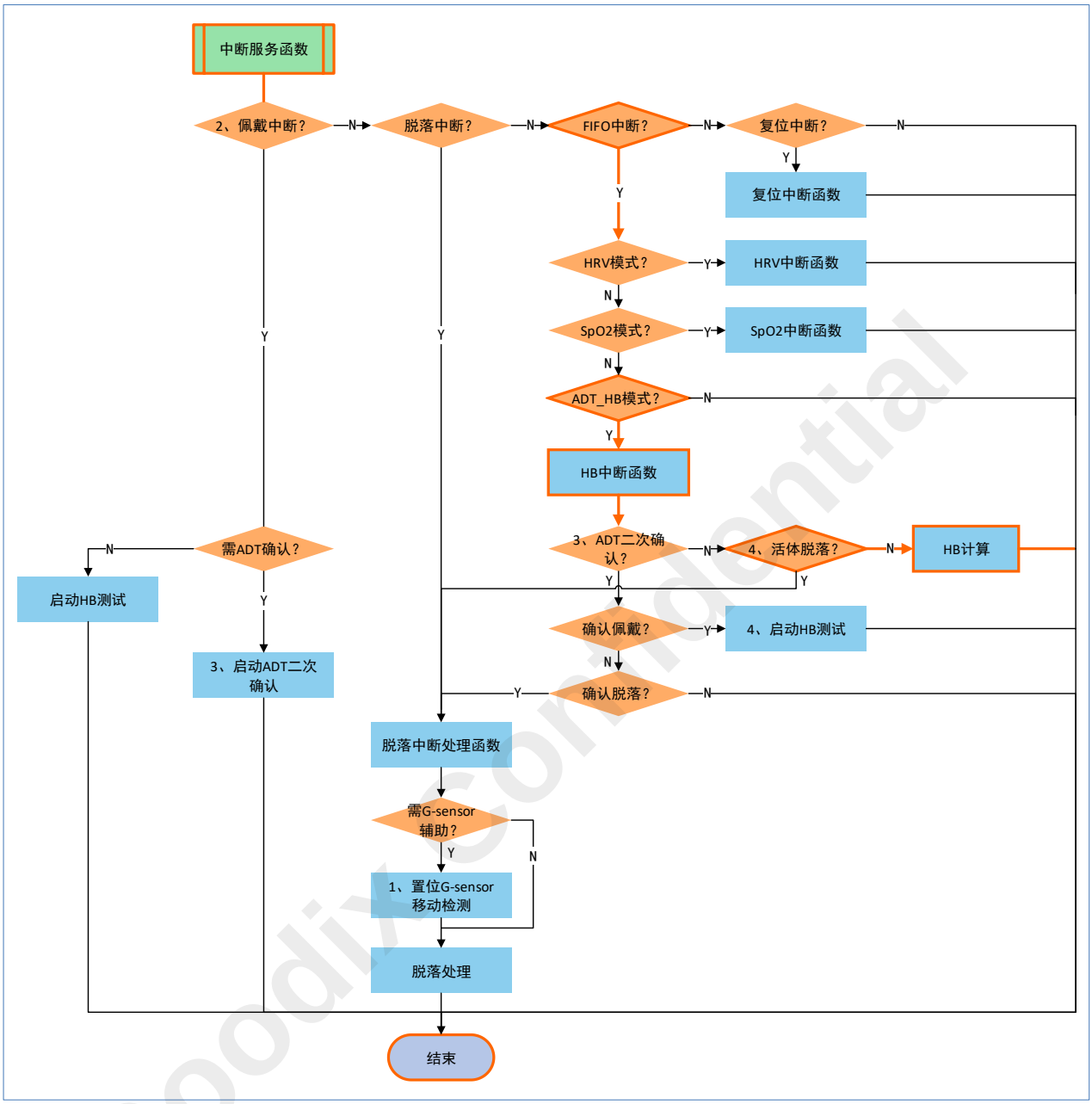


图 2-9 HR-PPG 活体脱落检测流程

2.2.3 心率模块

心率模组上电后，需先调用 gh30x_module_init()函数初始化心率模组和 G-sensor，方可启动心率检测。

函数说明：

- gh30x_module_start：支持 ADT 检测的启动测试，根据参数启动 HB、SpO2、HRV 的功能。
- gh30x_module_start_without_adt：不支持 ADT 检测的启动测试，根据参数启动 HB、SpO2、HRV 的功能。
- gsensor_drv_get_fifo_data：获取 g-sensor 的 FIFO 数据，G-sensor 的数据可用于心率、血氧、HRV 的计算。

该函数需用户自己实现，其实现思路为：一般 FIFO 中断为 1s 钟一次，则在这 1s 内产生的 G-sensor 数据都需要传给算法计算。

- 心率模式下，采样率为 25 Hz，则需至少传入 25 组 G-sensor 数据给心率计算函数。
- 血氧模式下，采样率为 100 Hz，则需至少传入 100 组 G-sensor 数据给心率计算函数。若 G-sensor 采样率达不到 100 Hz，则需进行插值处理，以确保数据不少于 100 组。
- HRV 模式下，采样率为 200 Hz，则需至少传入 200 组 G-sensor 数据给心率计算函数。如果 G-sensor 采样率达不到 200 Hz，则需要进行插值处理，以确保数据不少于 200 组。

静态心率测试：初始化完成后，可在 G-sensor 中断处理函数 hal_gsensor_drv_int1_handler()中，添加佩戴检测的逻辑（参考 2.2.2 佩戴模块），根据佩戴检测结果，启动 HB 测试。或者应用直接调用函数 gh30x_module_start（GH30X_RUN_MODE_HB）启动 ADT 佩戴检测并测试心率，还可调用函数 gh30x_module_start_without_adt（GH30X_RUN_MODE_HB）跳过 ADT 直接启动心率测试。

运动心率测试：运动心率是指用户系统设置了不同的运动模式，并测试对应模式下的心率，且不同运动模式的心率特征会有差异。因此心率算法需匹配相应的运动模式。启用心率检测的运动模式时，需在 gh30x_algo_hook.c 文件里，将 goodix_hba_init()的入参 stHbaConfig.scence 配置为对应的运动场景；切换心率检测的运动模式时，需要重新调用 gh30x_module_start（GH30X_RUN_MODE_HB）或者 gh30x_module_start_without_adt（GH30X_RUN_MODE_HB）；在回调到算法的 goodix_hba_init()前，需要将函数入参 stHbaConfig.scence 配置为对应的运动场景，切换后的心率检测运动模式才能生效，此时心率也会重新开始测试。

说明：

切换运动模式时，在修改运动场景配置后需重新启动心率测试。

进入和退出运动模式的流程如下：

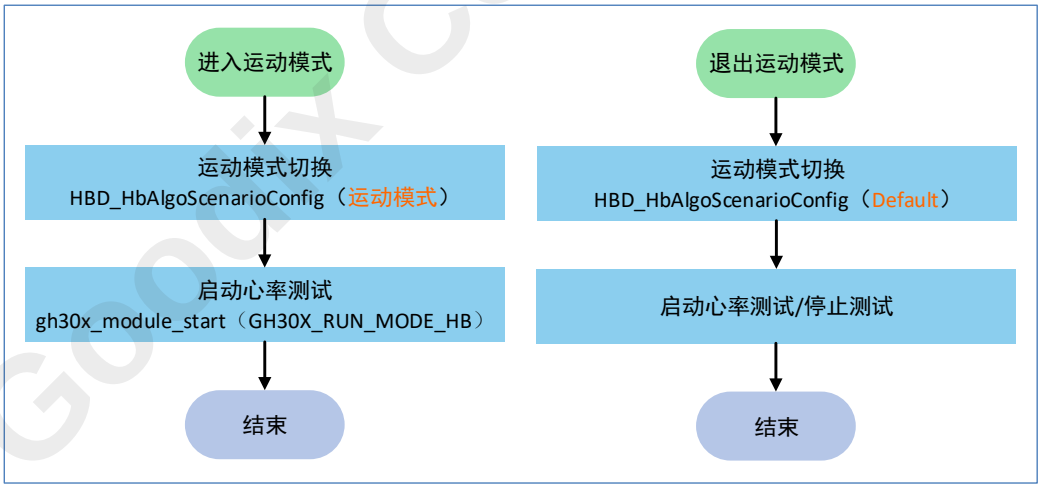


图 2-10 进入和退出运动模式流程

心率模式启动后，会在心率的中断函数 gh30x_fifo_evt_hb_mode_calc()中输出心率值，函数 HBD_HbCalculateByFifoIntEx()的输出参数 hb_value 即为心率值，可在函数 handle_hb_mode_result()中将心率值传递给应用层。

2.2.4 血氧模块

血氧测试流程跟心率测试流程相似，启动调用的函数一样，调用 gh30x_module_start（GH30X_RUN_MODE_SPO2）或者 gh30x_module_start_without_adt（GH30X_RUN_MODE_SPO2），只是传入的参数不同；血氧的中断函数为 gh30x_fifo_evt_spo2_mode_calc()，血氧模式可以同时输出心率值和血氧值，计算函数 HBD_Spo2CalculateByFifoIntEx ()的输出参数 hb_value 和 spo2_value 分别为心率值和血氧值，可在

函数 handle_spo2_mode_result 中将对应的值传给应用层。

此外，血氧测试流程还可调用 gh30x_module_start（GH30X_RUN_MODE_HR_SPO2）或者 gh30x_module_start_without_adt （GH30X_RUN_MODE_HR_SPO2），其中断函数为 gh30x_fifo_evt_hr_spo2_mode_calc（），此模式下的心率出值较上一种模式更可靠，性能更好，但空间占用更大，请根据实际情况选择。

2.2.5 HRV 模块

HRV 的测试流程跟心率、血氧的差异不大，启动调用的函数也一样，调用 gh30x_module_start（GH30X_RUN_MODE_HRV）或者 gh30x_module_start_without_adt（GH30X_RUN_MODE_HRV），只是传入的参数不同；HRV 的中断函数为 gh30x_fifo_evt_hrv_mode_calc()，计算函数为 HBD_HbWithHrvCalculateByFifoIntDbgDataEx()，计算结果存储在数组 RRvalueArry 中，可在函数 handle_hrv_mode_result()中将对应的值传给应用层。

2.3 应用调试

2.3.1 G-sensor 调试

2.3.1.1 G-sensor 要求

为保证动态心率算法的正常工作，G-sensor 必须满足以下 4 项参数要求：

表 2-3 G-sensor 要求

参数	要求	备注
精度	512LSB/g	若 G-sensor 精度不是 512LSB/g，则需进行精度归一化处理。传递给心率算法的精度必须为 512LSB/g。
量程与跳点	量程≥ ±4g，无跳点	建议量程设置为±4g
噪声	不大于 25LSB	
采样周期	≥ GH30x 的采样周期	心率算法要求送入的数据量为前 1s 的 G-sensor 数据，并要求 G-sensor 的采样频率为 25 Hz。 例如，若 G-sensor 的采样频率为 25 Hz，则送入的数据量为 25 个；若 G-sensor 的采样频率为 50 Hz，则送入的数据量为 50。 如果 25 Hz 采样频率送入的数据量不足 25 个，需对 G-sensor 数据进行补值。

2.3.1.2 G-sensor 驱动调试

为调试 G-sensor 驱动，用户需在 *gh30x_example_prot.c* 文件中实现 G-sensor 初始化、通用模式、FIFO 模式、动作检测模式等接口函数。

- G-sensor 初始化：gsensor_drv_init（必须实现）

  gsensor_drv_init()函数用于初始化 G-sensor 参数，要求用户必须实现。

```
int8_t gsensor_drv_init(void)
{
    int8_t ret = GH30X_EXAMPLE_OK_VAL;
    gsensor_drv_motion_det_mode = false;
    // code implement by user
    /* if enable all func equal 25Hz, should config > 25Hz;
    but if enable have 100hz, should config to > 100hz. if not, feeback to GOODIX!!!
```

```
    */
    return ret;
}
```

说明：

若用户已在其他模块实现了该函数，可替换 *gh30x_example_prot.c* 中的示例代码。

- 通用模式：gsensor_drv_enter_normal_mode

  gsensor_drv_enter_normal_mode()函数用于使能 G-sensor 低功耗模式。在该模式下，G-sensor 不会自动采集数据。

```
void gsensor_drv_enter_normal_mode(void)
{
    // code implement by user
    gsensor_drv_motion_det_mode = false;
}
```

- FIFO 模式：gsensor_drv_enter_fifo_mode

  gsensor_drv_enter_fifo_mode()函数用于使能 G-sensor 测量模式。在该模式下，G-sensor 将测量的数据存放在 FIFO 中。通过切换 gsensor_drv_enter_normal_mode 和 gsensor_drv_enter_fifo_mode，可降低功耗。若要和其他功能共用 G-sensor，仅需保证能够按时、按量传入 G-sensor 数据到算法接口即可。

```
void gsensor_drv_enter_fifo_mode(void)
{
    // code implement by user
    gsensor_drv_motion_det_mode = false;
}
```

- 动作检测接口：gsensor_drv_enter_motion_det_mode

  gsensor_enter_motion_det_mode()函数用于使能 G-sensor 动作检测模式。在该模式下，G-sensor 将进行运动检测。若检测到运动超过设定阈值，则将调用中断服务，触发 GH30x 检测。

```
void gsensor_drv_enter_motion_det_mode(void)
{
    // code implement by user
    gsensor_drv_motion_det_mode = true;
}
```

说明：

若应用逻辑无需检测功能，则可在 *gh30x_example_config.h* 文件中，将__HB_START_WITH_GSENSOR_MOTION__设置为“0”，且无需实现该函数。

- G-sensor 获取寄存器数据、FIFO 数据的接口：
  - 获取寄存器数据：gsensor_drv_get_data（一般用于透传）
  - 获取 FIFO 数据：gsensor_drv_get_fifo_data（用于 MCU 端传送 G-sensor 数据给算法，必须实现）。

2.3.1.3 G-sensor 应用场景

G-sensor 主要应用于佩戴检测和动态心率检测。

- 佩戴状态检测

  佩戴状态检测流程包括：佩戴检测和脱落检测，如图 2-11 所示。

  在未佩戴状态下，由 G-sensor 动作触发佩戴检测；在佩戴状态下，通过活体检测判断是否脱落。

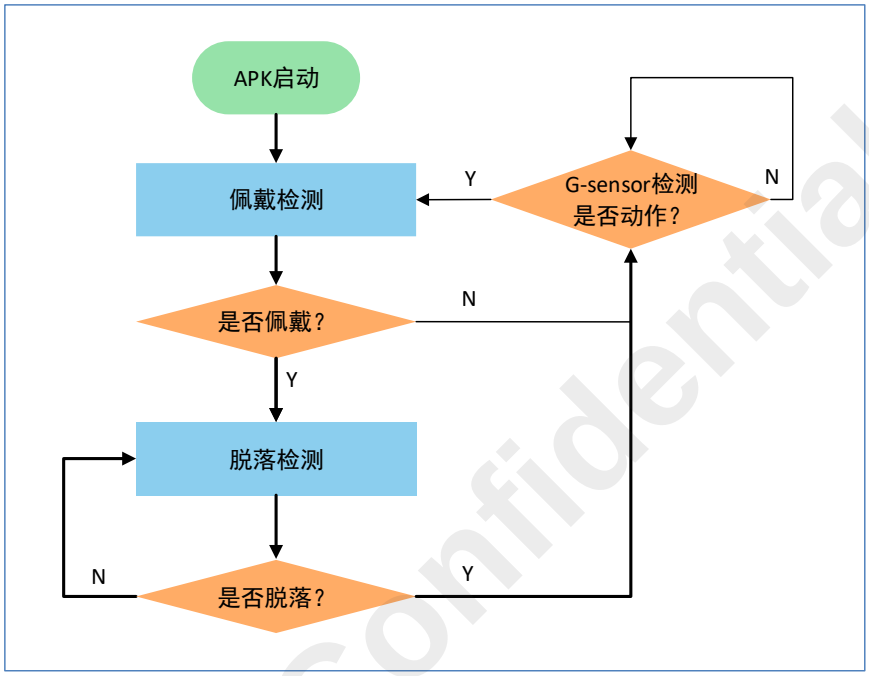


图 2-11 佩戴状态检测流程

1. 佩戴检测

   佩戴检测包含接近检测和佩戴确认：

   (1) 接近检测以 5 Hz 的频率采集红外 PPG 数据。

   (2) 佩戴确认采样频率为 25 Hz。通过比较 G-sensor 幅度/PPG 与阈值，判断是否为佩戴状态。

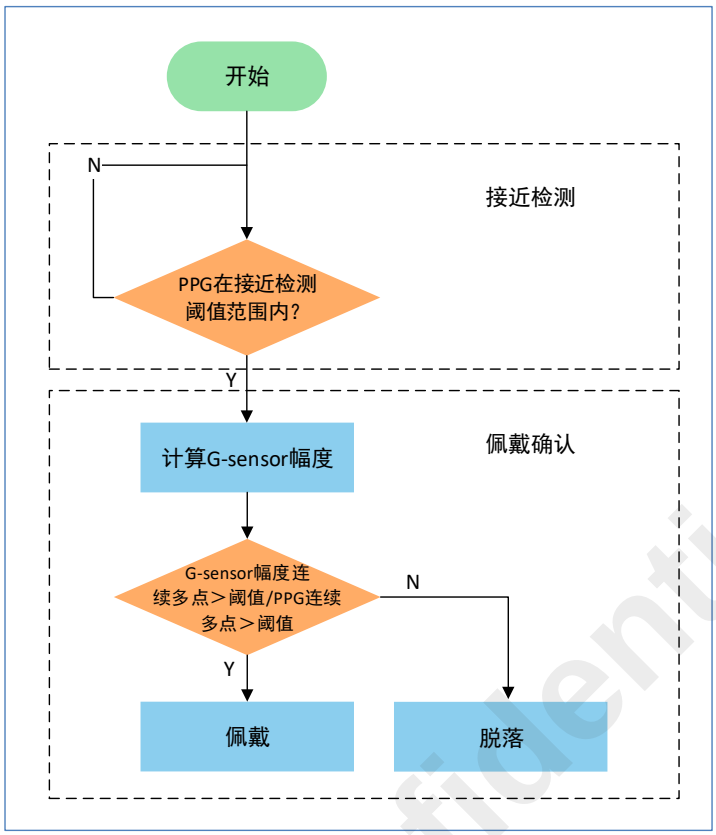


图 2-12 接近检测和佩戴确认流程

2. 脱落检测

通过判断 G-sensor 加速度是否变化，或 PPG 信号是否正常，进行脱落检测。

检测周期为 1s。

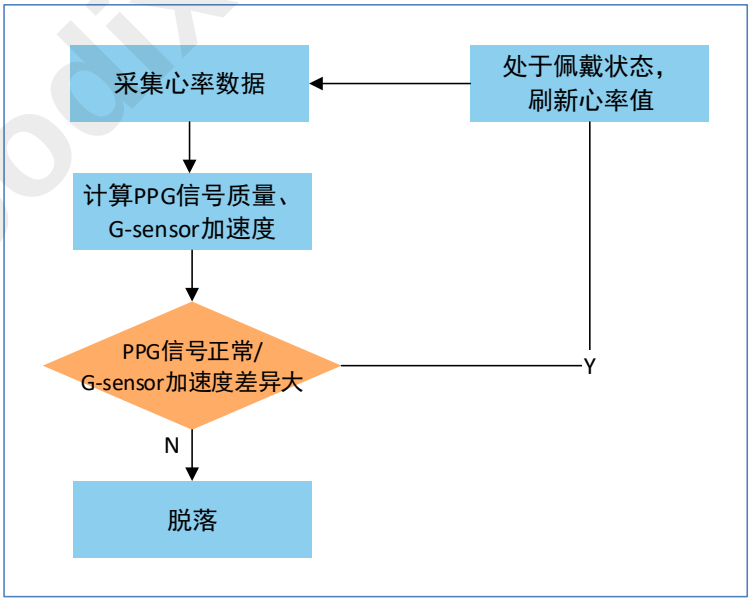


图 2-13 脱落检测流程

- 心率检测

心率算法将利用 G-sensor 数据进行运动的判断和去除运动噪声等。

2.3.1.4 G-sensor 应用设置

参考表 2-3，心率算法对 G-sensor 的精度要求为 512 LSB/g，量程≥±4g。若不符合要求，则需进行调整。G-sensor 数据宽度的计算公式为：

2^x = LSB *满量程（比如±4g 的满量程是 8）　　（1）

由上述公式，可计算出 G-sensor 的最小数据宽度为 512*8 (±4) = 4096，即 2^12。因此，G-sensor 数据位宽为 12-bit。

当 G-sensor 精度小于 512LSB/g 或量程小于 4g 时，需根据 G-sensor 实际情况配置参数或更换 G-sensor；当 G-sensor 位宽大于 12-bit 时，需进行归一化处理，以使得 G-sensor 精度满足算法要求。

在 *hbd_ctrl.h* 中，定义了 G-sensor 精度的预处理宏。

```
typedef enum

{

    HBD_GSENSOR_SENSITIVITY_512_COUNTS_PER_G = 0,

    HBD_GSENSOR_SENSITIVITY_1024_COUNTS_PER_G,

    HBD_GSENSOR_SENSITIVITY_2048_COUNTS_PER_G,

    HBD_GSENSOR_SENSITIVITY_4096_COUNTS_PER_G,

    HBD_GSENSOR_SENSITIVITY_8192_COUNTS_PER_G,

} EM_HBD_GSENSOR_SENSITIVITY;
```

G-sensor 数据经过精度预处理后，再传入心率算法。

```
GS8    HBD_HbCalculateByFifoIntEx(ST_GS_DATA_TYPE    stGsAxisValue[],    GU16    usGsDataNum,
EM_HBD_GSENSOR_SENSITIVITY    emGsensorSensitivity,    GS32    nRawdataOut[][6],    GU16
*pusRawdataOutLen, ST_HB_RES *pstHbRes)
```

用户可在 *gh30x_example_config.h* 中修改 G-sensor 精度的宏定义。

例如，若 G-sensor 实际精度为 2048LSB/g，则可按下列代码设置宏定义：

```
/// gsensor sensitivity normalized

#define __GS_SENSITIVITY_CONFIG__          (HBD_GSENSOR_SENSITIVITY_2048_COUNTS_PER_G)
```

或者，在送入算法前对 G-sensor 数据右移 2 位处理。

G-sensor 数据对齐

心率算法获取 G-sensor 数据的目的是消除运动干扰等。因此，G-sensor 数据和 GH30x 的采样需保持同步。心率算法的处理逻辑为：主控接收到 GH30x 中断后，即给算法传入 GH30x 的 1s 数据。因此，为了获取同步的 G-sensor 数据，主控接收到 GH30x 中断后，需从 G-sensor Buffer 中读取前 1s 的数据。

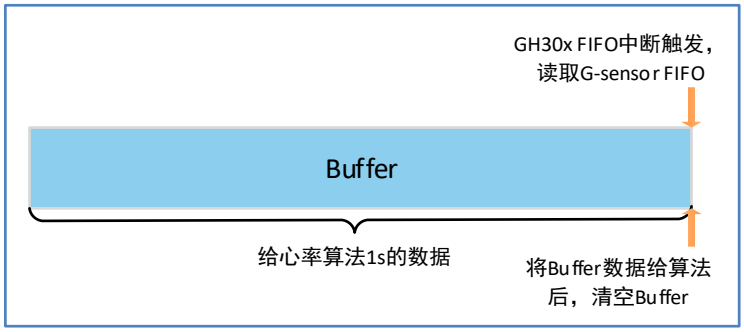


图 2-14 G-sensor 数据对齐

用户可通过定义一个软件 Buffer，实现 G-sensor 数据对齐。当 GINT_FIFO 管脚（主控设备与 G-sensor 连接的中断管脚）连接或断开时，定义的软件 Buffer 结构有所不同。

- 连接 GINT_FIFO

  在连接 GINT_FIFO 的情况下，用户需定义一个软件 Buffer，其结构如下图所示。

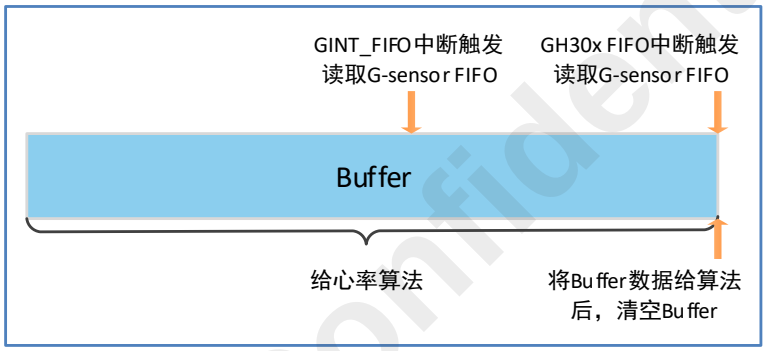


图 2-15 Buffer 结构图

  中断处理包括两种：GINT_FIFO 中断和 GH30x FIFO 中断。

  - GINT_FIFO 中断触发读取 G-sensor FIFO：

    读取 G-sensor FIFO 数据，写入 Buffer，改变 Index。

  - GH30x FIFO 中断触发读取 G-sensor FIFO：

    读取 G-sensor FIFO 数据，写入 Buffer，改变 Index。

  将主控从 Buffer 读取的数据发送给 HBD_HbCalculateByFifoInt 进行计算，Index 设置为 0。

说明：

上述为推荐方案，具体实现可参考示例工程以及示例代码。

- 断开 GINT_FIFO：

  在断开 GINT_FIFO 管脚的情况下，用户需定义一个软件 Ring Buffer，其结构如下图所示。

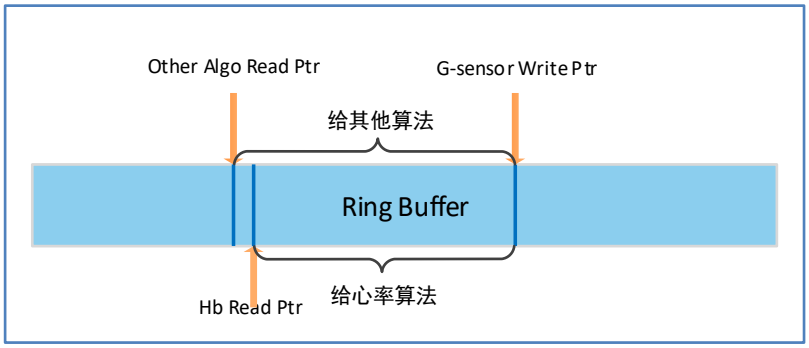


图 2-16 Ring Buffer 结构图

说明：

如果 G-sensor 数据还有其他应用也可使用该方式，一个写指针，不同应用对应不同读取指针，对应的应用获取的数据为对应读指针与写指针间的数据。

中断处理包括两种：定时读取 G-sensor FIFO 的中断，GH30x FIFO 中断。

- 定时器中断触发读取 G-sensor FIFO：

  (1) 读取 G-sensor FIFO 数据，写入 Ring Buffer，改变 G-sensor write ptr。

  (2) 若存在其他应用，主控先将 Other App Read Ptr 和 G-sensor Write Ptr 之间的数据发送给其他应用，再将 Other App Read Pter 赋值为 G-sensor write ptr。

- GH30x FIFO 中断触发读取 G-sensor：

  (1) 读取 G-sensor FIFO 的数据，写入 Ring Buffer，改变 G-sensor write ptr。

  (2) 读取 Ring Buffer 数据，将 Hb read ptr 赋值为 G-sensor write ptr，读取的数据会发送给 HBD_HbCalculateByFifoIntEx()函数进行计算。

时间轴示例参见图 2-17。其中，定时时间为 500 ms，G-sensor 采样率为 50 Hz。

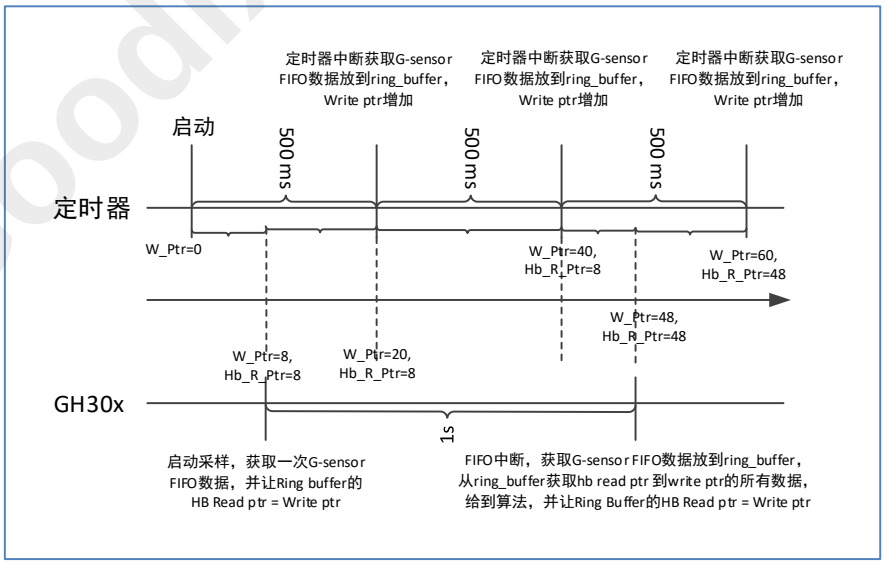


图 2-17 时间轴示例

2.3.1.5 G-sensor 应用配置

脱落后再次进行佩戴检测时，用户可通过配置 *gh30x_example_config.h* 中的宏定义 __HB_START_WITH_GSENSOR_MOTION__，设置佩戴检测是否需要通过 G-sensor 触发。

如果该宏定义配置为 1，即使能 G-sensor 触发，则用户需实现 gsensor_drv_enter_motion_det_mode()函数，并注册 G-sensor 触发回调函数 hal_gsensor_drv_int1_handler()。

```
/// whether need gsensor motion before start adt detect
#define __HB_START_WITH_GSENSOR_MOTION__                    (1)
```

一般情况下，开始调试时可能还未实现 G-sensor 功能，因此建议将宏定义配置为 0 或者忽略再次佩戴的效果。

2.3.2 蓝牙透传调试

对于具有蓝牙功能的应用，可利用 App 软件，将主控设备的数据通过蓝牙（BLE）传输到手机，进行蓝牙透传功能调试。

说明：

- Andriod 手机：支持蓝牙 4.0 及以上版本。
- 主控设备：支持蓝牙 4.0 及以上版本。
- 蓝牙调试 App：可使用 Goodix 提供的 GRToolbox App （Andriod），下载地址：https://product.goodix.com/zh/product/category/ble。
- 应用 App：Goodix 提供的 GHealth App，可联系 Goodix 技术支持或代理商获取。

下面章节将介绍蓝牙透传调试的具体操作步骤。

2.3.2.1 新增蓝牙服务

在蓝牙主控芯片的 BLE GATT 初始化代码中，添加一个自定义服务和两个特征，具体要求如下：

- 自定义服务 UUID："0000190e-0000-1000-8000-00805f9b34fb"

  GHealth App 专用 BLE 服务，用于传输算法数据（设备与手机间的通信）

- 发送特征 UUID："00000003-0000-1000-8000-00805f9b34fb"

  GHealth App 发送特征，用于设备向手机上传算法相关数据。

- 接收特征 UUID："00000004-0000-1000-8000-00805f9b34fb"

  GHealth App 接收特征，用于手机向设备下发算法相关命令。

对于满足要求（参考 1.3 支持平台）的蓝牙主控平台，均可添加上述自定义服务与特征。

为方便操作，用户可直接将下列代码拷贝到对应的 GATT 初始化函数中。

```
#define GH30X_SERVICE_UUID {0xFB, 0x34, 0x9B, 0x5f, 0x80, 0x00, 0x00, 0x80, 0x00, 0x10, 0x00, 0x00, 0x0E, 0x19, 0x00, 0x00}
#define GH30X_TX_UUID {0xFB, 0x34, 0x9B, 0x5f, 0x80, 0x00, 0x00, 0x80, 0x00, 0x10, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00}
#define GH30X_RX_UUID {0xFB, 0x34, 0x9B, 0x5f, 0x80, 0x00, 0x00, 0x80, 0x00, 0x10, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00}
```

代码添加完成后，可利用蓝牙调试 App（GRToolbox）进行测试验证。若在服务列表中，发现自定义的服务和特征（如图 2-18 所示），即表示成功新增蓝牙服务。

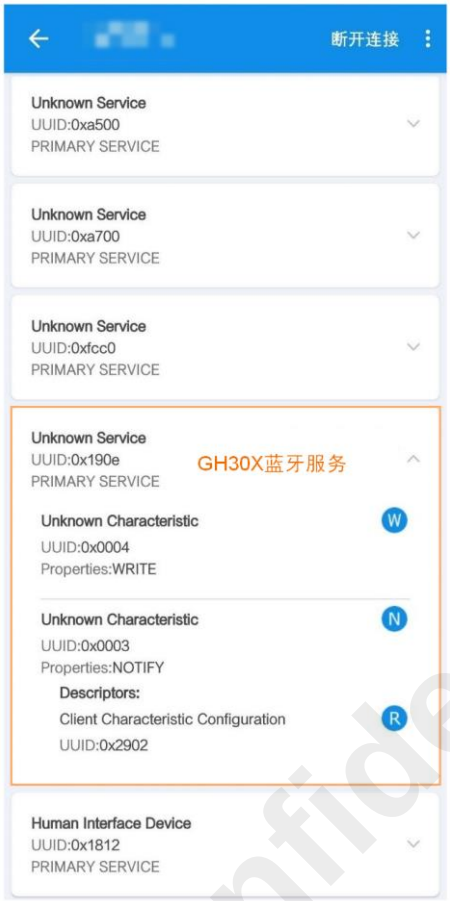


图 2-18 服务初始化界面

2.3.2.2 注册蓝牙接收数据回调函数

GHealth App 连接设备后，会寻找所需的蓝牙服务和特征；成功找到后，再通过 GHealth App 接收特征下发命令；此时，BLE 主控软件将触发通知事件。

用户代码需调用蓝牙数据解析函数 ble_module_recv_data_via_gdcs()，如下所示：

```
void ble_module_recv_data_via_gdcs(uint8_t *data, uint8_t length)
{
    gh30x_app_cmd_parse(data, length);
}
```

说明：

- 函数声明位于 *gh30x_example.h*，需确保将该文件加入工程并参与编译。
- 函数实现代码位于 *gh30x_example_port.c*，需确保该文件加入工程头文件的搜索目录。

2.3.2.3 注册蓝牙发送数据回调函数

设备接收 GHealth App 发送的数据后，需进行回复。此时，算法库会自动调用注册的蓝牙发送函数，再进行数据上传操作。在 GH30x 算法库初始化阶段，调用 gh30x_comm_pkg_init()函数将 BLE 发送函数注册到算法库中。

当注册的蓝牙发送函数需发送数据时，可调用 *gh30x_example_port.c* 文件中的 ble_module_send_data_via_gdcs()函数，其示例代码如下：

```
uint8_t ble_module_send_data_via_gdcs(uint8_t data[], uint8_t length)
{
```

```
    uint8_t ret = GH30X_EXAMPLE_OK_VAL;
    // code implement by user
    ble_tx_data_send(data, length);
    return ret;
}
```

用户需根据实际情况，添加蓝牙主控平台的发送数据函数的调用（替换示例代码中的绿色语句）。

2.3.2.4 注册 MCU 模式发送数据定时器函数

在蓝牙 MCU 模式下，需使用一个定时器作为事件驱动发送数据。在 GH30x 算法库初始化阶段，调用 gh30x_comm_pkg_init()函数初始化定时器。

用户需在 *gh30x_example_port.c* 文件的 ble_module_repeat_send_timer_init()函数中添加用户代码，以实现定时器初始化。定时器建议配置为 100 毫秒触发，且蓝牙连接间隔小于定时器触发时间 100 毫秒。

```
void ble_module_repeat_send_timer_init(void)
{
    // code implement by user
    // must register func ble_module_repeat_send_timer_handler as callback
     /* should setup 100ms timer and ble connect interval should < 100ms
     */
}
```

另外，用户还需实现开启定时器、关闭定时器这两个函数，并且在定时器触发函数中增加发送数据函数的调用，参考代码如下：

```
void ble_module_repeat_send_timer_start(void)

{
    // code implement by user
}
void ble_module_repeat_send_timer_stop(void)
{
    // code implement by user
}
void TIMER_IRQHandler(void)
{
    ble_module_repeat_send_timer_handler();
}
```

2.3.2.5 验证蓝牙透传功能

上述操作完成后，可使用蓝牙调试 App（以 GRToolbox App 为例）验证蓝牙透传功能。

1. 将设备正常佩戴在合适位置，并使用 GRToolbox 连接设备。
2. 当 GRToolbox 成功发现服务后，进入服务列表页面。找到 GHealth App 服务，然后打开发送特征的通知功能，如图 2-19 的图 1 所示。
3. 使用接收特征写入四字节的十六进制数据“0xC022452C”，并发送给主控设备，如图 2-19 的图 2 所示。随后，主控设备的绿灯将被点亮并闪烁。

4. 使用发送特征接收数据，先接收到上一步发送的数据“0xC022452C”，随后持续接收数据，如图 2-19 的图 3 所示，即表示蓝牙透传功能已正常工作。

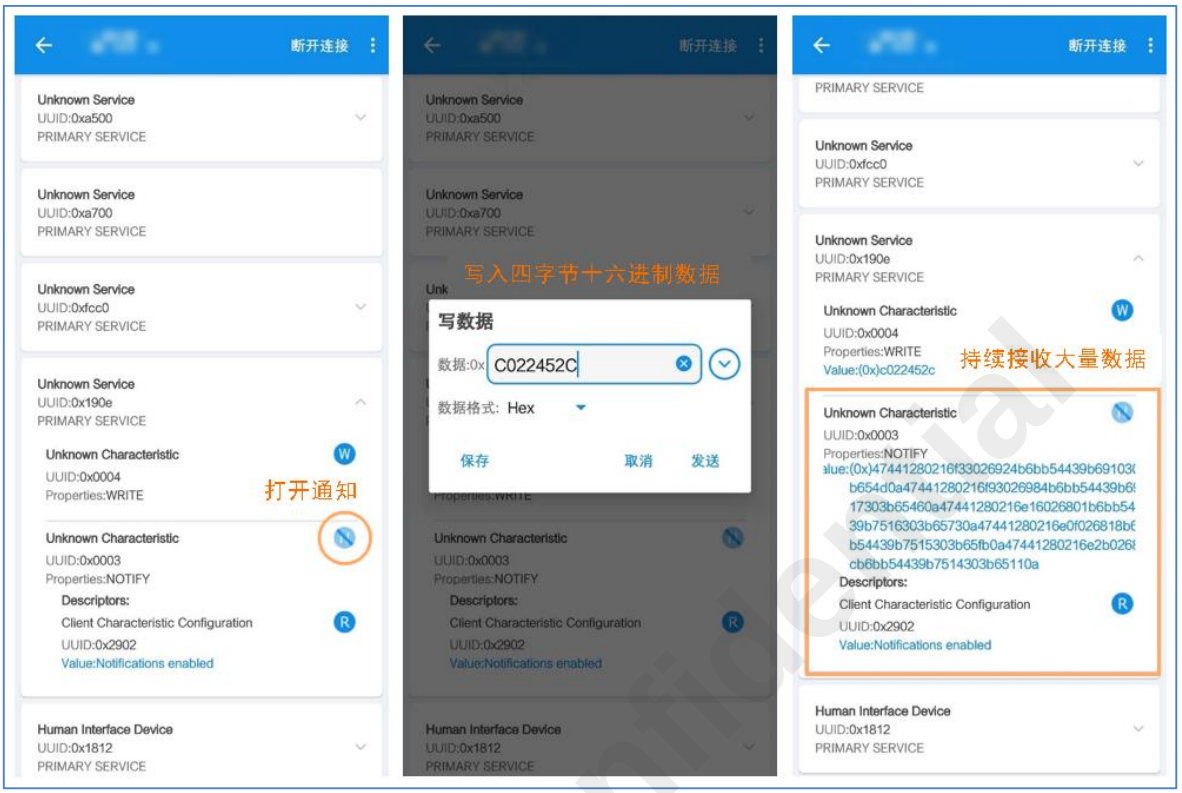


图 2-19 蓝牙透传功能验证 App 界面

在蓝牙透传 MCU 模式下，使用设备固件中的算法库执行运算。算法执行完毕后，将运算结果和参与运算的 PPG、ACC 源数据打包，再通过蓝牙发送至手机端。通常每秒开启一轮蓝牙发送，并需在下次蓝牙发送开启前，完成所有数据的发送：心率模式每秒发送 650 字节左右，血氧模式每秒发送 2600 字节左右。蓝牙透传 MCU 模式，需配合 2.1.4 添加用户代码初始化的定时器一起使用。

说明：

App 仅作为数据接收者，不会对设备固件运行产生影响，可以最大限度地做到与终端用户使用效果一致。

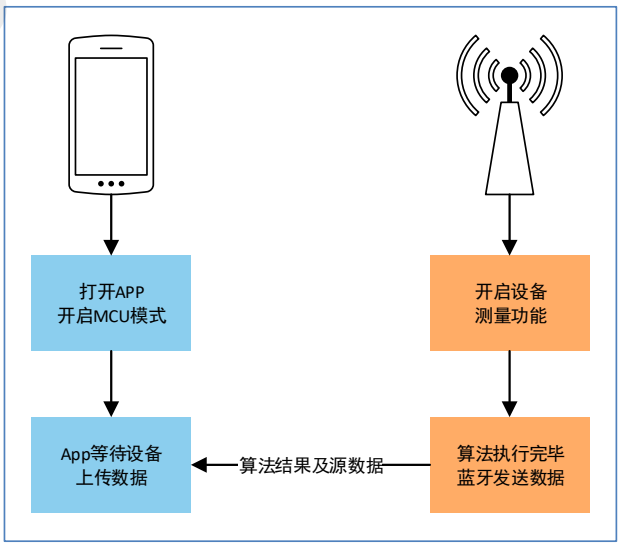


图 2-20 蓝牙透传调试应用模式

2.3.2.6 配置蓝牙透传功能

- __USE_GOODIX_APP__：使用蓝牙透传调试功能连接 GHealth App 时，需开启该宏定义。用户可在调试版本中开启该宏定义，在发布版本中关闭它。
- __ALGO_CALC_WITH_DBG_DATA__：使用蓝牙透传 MCU 模式时，需开启此宏定义，并与__USE_GOODIX_APP__宏定义配套使用。
- __NEW_DATA_MULTI_PKG_NUM__：使用蓝牙透传功能在一次调用数据发送函数时拼包发送多个数据包，该宏定义数量加 1 为拼包数量，默认值为 4，表示每次拼 5 包数据，共计 5*22 = 110 字节。
- __BLE_PKG_SIZE_MAX__：使用蓝牙发送最长数据包定义，一般对齐蓝牙连接后更新的 MTU，该定义与手机蓝牙强相关，需根据手机支持情况再合理设置。
- __BLE_MCU_PKG_BUFFER_MAX_LEN__：蓝牙透传 MCU 模式下的数据缓冲区长度，其长度值应大于该公式计算值：(DBG_MCU_PKG_RAW_FRAME_LEN * __ALGO_CALC_DBG_BUFFER_LEN__) + MCU_PKG_SPO2_ALGO_RESULT_LEN。
- DBG_MCU_MODE_PKG_LEN：蓝牙 MCU 模式数据包个数，保持默认不修改。
- DBG_MCU_PKG_RAW_FRAME_LEN：蓝牙 MCU 模式数据帧长度，保持默认不修改。
- DBG_MCU_PKG_HEADER_LEN：蓝牙 MCU 模式数据包头长度，保持默认不修改。

注意：

1. 使用蓝牙透传调试功能时，请确认蓝牙连接间隔（Connection Interval）设置是否可以满足数据传输使用。App 模式心率应用要求连接间隔小于 40 毫秒，血氧应用要求连接间隔小于 10 毫秒；MCU 模式心率应用要求每秒可以发送 650 字节数据，MCU 血氧应用要求每秒可以发送 2600 字节数据。
2. 使用蓝牙 MCU 模式，需要确认蓝牙 MTU 长度，避免应用层单次发送的数据长度大于蓝牙最大单次发送的数据长度，造成堵塞影响蓝牙芯片正常工作。

2.3.3 工程模式调试

工程模式的测试，一般是在采用 MCU 本地测试和 MT 测试（或者整机测试）中二选一，MCU 本地产测是不是必须的，视用户实际应用情况而定。

2.3.3.1 MCU 本地测试

测试准备

如果用户确定使用 MCU 本地测试，测试之前需完成以下准备工作：

- 硬件设备：GH30x 模组、乳白色测试头、黑色测试头。
- 产测库：GH30x 产测库（（TestLib v1.1.0 及以上版本）
- 使用 MCU 本地产测功能时，需关注的使能开关和测试接口包括：
  - __SYSTEM_TEST_SUPPORT__：MCU 本地产测使能。
  - gh30x_systemtest_start：启动 MCU 本地产测的接口函数。

一般情况下，MCU 本地测试包括放置白色测试头的光路灵敏度测试和放置黑色吸光硅胶头的漏光测试。由于漏光和灵敏度测试的物理要求不一样，所以可利用物理按键触发漏光和灵敏度测试事件。

测试步骤

测试准备工作完成后，可按如下步骤开始测试：

1. 在 *gh3011_example_config.c* 中，将宏定义__SYSTEM_TEST_SUPPORT__配置为 1，使能 MCU 本地产测功能。
2. 在 main 函数中调用 gh30x_systemtest_start(EMGh30xTestItem mode)函数，并选择测试模式：
   - HBDTEST_TESTIEM_COMM 通信测试
   - HBDTEST_TESTIEM_OTP OTP 测试
   - HBDTEST_TESTIEM_CTR 光路灵敏度测试（包括噪声测试）
   - HBDTEST_TESTIEM_LEAK 漏光测试
   - HBDTEST_ONLYTEST_CTR_LEAK 光路灵敏度测试和漏光测试
   - HBDTEST_TEST_ALL_TEST 通信测试、OTP 测试、光路灵敏度测试、漏光测试
3. 在 *gh3011_example_reg_array.c* 中，配置 ledmask 选择 LED 0-2 进行测试，ledmask[0]- ledmask[2]分别对应 LED 0-1，相应 LED 置 1，即表示使用该 LED 进行测试。通过配置 led0std/led1std/led2std，可设置对应 LED 的测量阈值和其他默认值，若无需默认值，则置 0 即可。在测试过程中，如有测试不通过，则会立即停止测试，可以通过打印信息查看 CTR、LEAK、测试不通过的 LED 等结果。可参考如下的 HBDTEST_ROMATSTRes 结构体定义或头文件 *systemTest.h*。

   结构体 HBDTEST_ROMATSTRes：
   - _flag：若该变量置为 1，则产测功能会去计算_leadRatio
   - _CTR：CTR 测试结果
   - _leak：漏光测试结果
   - _leakRatio：Leak/CTR
   - _noise：噪声测试结果

HBDTEST_ROMATSTRes 结构体为测试计算结果，也可在 *gh3011_example_reg_array.c* 设置相关默认值。若_flag 为 0，则 leakRatio 无意义，将一直为 0，如：只进行漏光测试模式；若在 *gh3011_example_reg_array.c* 里设置默认值，当_flag 为 1 时，_CTR 不能设置为 0。

```
HBDTEST_ROMAStd: _ctrStd: min ctr
_leakStd: max leak
        _ratio: max leak/ctr ratio
        _noiseStd: max noise
```

HBDTEST_ROMAStd 结构体为相关阈值，可在 *gh3011_example_reg_array.c* 中设置。

```
HBDTEST_ROMAConfigParam: _ledResisLst:the resistance in config
         _ledCurrLst:the current in config
```

HBDTEST_ROMAConfigParam 结构体为相关配置参数。

gh3011_example_port.c 中的 handle_system_test_result()函数为结果处理函数，参数 test_res 为测试结果，具体定义为：

- <0=> ok 测试成功

- <1=> order err 操作错误
- <2=> comm read err 通信读操作错误
- <3=> comm write err 通信写操作错误
- <4=> otp read err OTP 操作错误
- <5=> ctr not pass CTR 测试不通过
- <6=> rawdata not pass Rawdata 测试不通过
- <7=> noise not pass 噪声测试不通过
- <8=> leak not pass 漏光测试不通过
- <9=> leakratio not pass Leak/CTR 测试不通过
- <10=> resource error 源错误

参数 led_num 只在出现测试失败时有意义，对应测试不通过的 LED，测试通过则第二个参数始终为 0，无意义。

进行光路灵敏度测试和漏光测试之前，可以设置停顿时间用来放置测试头，停顿时间可在 *gh3011_example_port.c* 中的 handle_before_system_os_test()函数进行设置，该函数需在光路灵敏度测试和漏光测试开始之前被调用。

若测试模式为 HBDTEST_TEST_ALL_TEST，LED 0-2 全部置 1 进行测试时，将按照“通信测试→OTP 测试→LED0 光路灵敏度测试、LED1 光路灵敏度测试、LED2 光路灵敏度测试→LED0 漏光测试、LED1 漏光测试、LED2 漏光测试”的顺序依次进行，其余模式和 LED 选择以此类推。光路灵敏度测试和漏光测试将分别测试 100 次，测试次数可在 *gh3011_example_config.c* 中__SYSTEM_TEST_DATA_CNT_CONFIG__设置，且必须小于等于 100。

2.3.3.2 UART 测试

工程模式下的 MT 测试，即使用 FPM02 工具板进行 UART 测试。该工具板的 UART 接口定义如下所示：

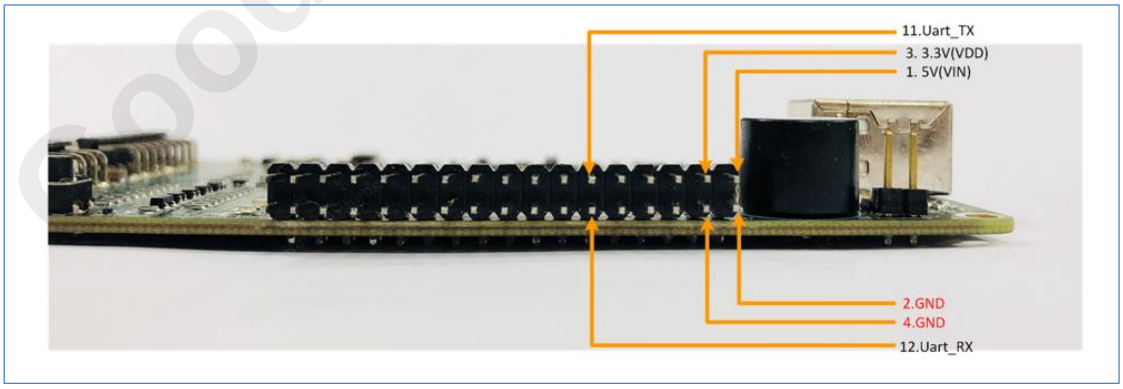


图 2-21 UART 通信接口(J6)定义（侧视图）

各管脚的具体描述，如表 2-4 所示：

表 2-4 UART 通信接口(J6)管脚描述

管脚序号	管脚名称	描述
1	5V (VIN)	5 V 供电电源，根据测试设备需求选择使用
2	GND	连接模组的 GND 网络
3	3.3V (VDD)	3.3 V 供电电源，根据测试设备需求选择使用
4	GND	连接模组的 GND 网络

管脚序号	管脚名称	描述
11	I2C1_SDA	复用 UART Tx，连接模组 UART_TX 网络
12	I2C1_SCL	复用 UART_Rx，连接模组 UART_RX 网络

当使用 UART 测试时，需根据实际应用需求，实现 *gh30x_example_port.c* 文件中的 uart_module_recv_data()函数和 uart_module_send_data()函数。除这两个函数以外，还需实现 UART 功能，包括管脚设置、UART 功能设置等。

uart_module_recv_data 函数主要用于从 UART 接收控制命令用于返回数据及控制 GH30x 功能；uart_module_send_data 函数主要用于发送数据给外部，以便于测试分析。

2.4 其他应用说明

2.4.1 漏中断处理

由于主控平台可能会出现漏中断的情况，用户可按如下步骤，添加漏中断的处理方案：

1. 在 *gh30x_example_config.h* 中，通过配置宏定义__FIFO_INT_TIMEOUT_CHECK__，使能于 FIFO 中断 Timeout 监控功能。该宏定义配置为 1 时，表示使能 FIFO 漏中断使能机制；配置为 0 时，则不开启。
2. 在 *gh30x_example_port.c* 中，实现以下三个函数（必须实现）：FIFO 中断函数监控定时器，定时间隔为 FIFO 中断间隔加 80 ms@25Hz、20 ms@100Hz。
    - hal_gh30x_fifo_int_timeout_timer_start：FIFO 中断函数监控定时器启动函数：使能定时器中断。
    - hal_gh30x_fifo_int_timeout_timer_stop：FIFO 中断函数监控定时器停止函数：禁用定时器中断。
    - hal_gh30x_fifo_int_timeout_timer_init：FIFO 中断函数监控定时器初始化函数：初始化定时器资源，包括定时器模式、定时时长、定时器中断处理事件等。

说明：

- 根据需求在 *gh30x_example_config.h* 文件中使能__FIFO_INT_TIMEOUT_CHECK__。
- 根据平台资源，实现上述函数。

图 2-22 为 FIFO 漏中断定时器程序的处理流程以及漏中断之后的处理。

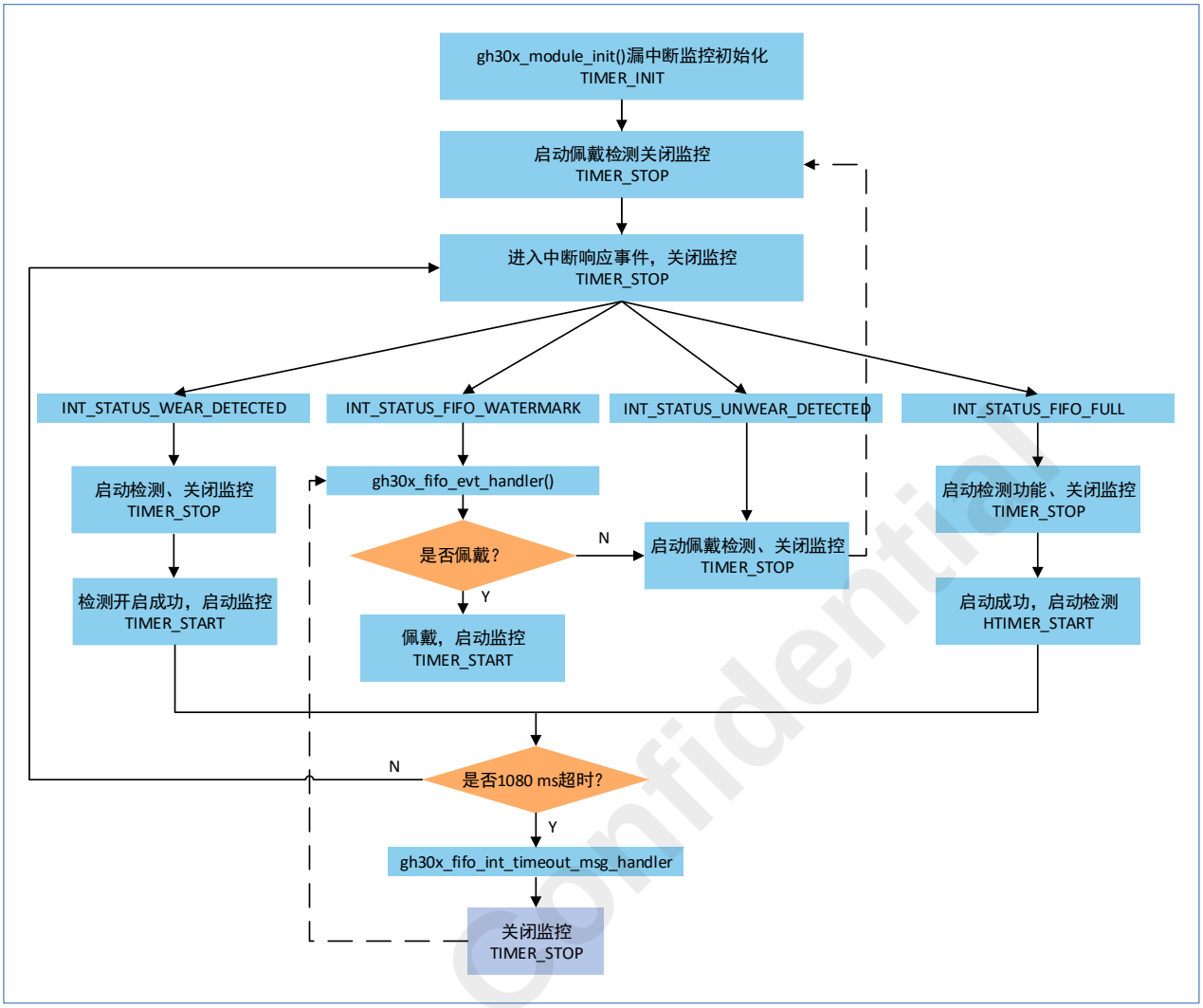


图 2-22 FIFO 漏中断定时器程序处理流程

```
TIMER_INIT: hal_gh30x_fifo_int_timeout_timer_init;
TIMER_START: hal_gh30x_fifo_int_timeout_timer_start;
TIMER_STOP: hal_gh30x_fifo_int_timeout_timer_stop;
```

2.4.2 平滑策略

平滑策略可应用在全天心率检测情况下，改善心率相差较大的突变问题：当上一次心率不为 0 且与当前心率值相差 35 以上时，取上一次心率和当前心率的平均值。

handle_hb_mode_result()函数的示例代码位于 *gh30x_example_port.c* 文件。

2.4.3 算法内存分配

静态内存的算法 Buffer 开辟在静态存储区。

动态内存的算法 Buffer 通过 malloc 申请在堆区，且用户可通过接口 HBD_GetMemRequired 获取算法所需内存大小。

2.4.4 功能及算法裁剪

SDK 可对应用功能及算法通过宏开关进行灵活裁剪，只有保证应用功能及算法同时开启时，才能正常应用。控制心率，血氧，心率变异性以及其他测试模式的应用功能宏开关定义如下：

```
/// hb detect support
#define __HB_DET_SUPPORT__                  (1)
/// spo2 detect support
#define __SPO2_DET_SUPPORT__                (1)
/* hb and spo2 */
#define __HR_SPO2_DET_SUPPORT__             (0)
/// hrv detect support
#define __HRV_DET_SUPPORT__                 (1)
/// get rawdata only support
#define __GET_RAWDATA_ONLY_SUPPORT__        (0)
/// factory detect support
#define __FACTORY_DET_SUPPORT__             (0)
/// adt detect support
#define __ADT_DET_SUPPORT__                 (0)
/// system test moudle support
#define __SYSTEM_TEST_SUPPORT__             (0)
```

SDK 也可灵活裁剪心率，血氧，HRV，活体检测以及佩戴确认算法，其中佩戴确认开源，其他库宏定义如下：

```
#define __HBD_HB_ALGORITHM_ENABLE__         (1)
#define __HBD_HRV_ALGORITHM_ENABLE__        (1)
#define __HBD_SPO2_ALGORITHM_ENABLE__       (1)
#define __HBD_NADT_ALGORITHM_ENABLE__       (1)
#define __HB_NEED_ADT_CONFIRM__             (1)
```

2.4.5 驱动库与算法库分离功能

驱动库与算法库可分离运行，控制宏定义：

```
/* driver lib and algo lib separate run enable ,like double core enable */
#define __HBD_DOUBLE_CORE_ENABLE__          (0)
```

开启此宏定义后，开发者可在中断处理函数中调用驱动库函数获取一次中断周期的原始数据，中间不会进行算法计算。开发者可灵活调用算法接口，不再与驱动耦合，算法接口调用可参考 *gh3011_algo_hook.c* 文件。仅 Lib_hba_v0.7.8.0 驱动库版本及以后版本支持此功能。

3 驱动库 API 说明

GH30x 驱动库提供了丰富的 API，可供用户应用层调用。

各 API 函数的具体描述，参见下列表格。

表 3-1 HBD_AlgoRegisterHookFunc 函数

函数原型	void HBD_AlgoRegisterHookFunc(GS32 (*pAlgoMemoryInitHookFunc)(GS32* error_code), void (*pAlgoMemoryDeinitHookFunc)(void), void (*pAlgoInitHookFunc)(EMFunctionID function_id,GU8 buffer_len,GS32 *algo_param_buffer), void (*pAlgoCalculateHookFunc)(EMFunctionID function_id,ST_ALGO_CALCULATE_INFO *algo_calc_info,ST_ALGO_RESULT *algo_result), void (*pAlgoDeinitHookFunc)(EMFunctionID function_id));
功能描述	算法注册函数
输入参数	pAlgoMemoryInitHookFunc, 算法内存初始化函数 pAlgoMemoryDeinitHookFunc, 算法内存去初始化函数 pAlgoInitHookFunc 算法功能初始化函数 pAlgoCalculateHookFunc 算法功能计算函数 pAlgoDeinitHookFunc 算法功能去初始化函数
输出参数	无
返回值	无

表 3-2 HBD_SetI2Crw API 函数

函数原型	GS8 HBD_SetI2cRW(GU8 uchI2cIdLowTwoBitsSelect, GU8 (*pI2cWriteFunc)(GU8 uchDeviceId, const GU8 uchWriteBytesArr[], GU16 usWriteLen), GU8 (*pI2cReadFunc)(GU8 uchDeviceId, const GU8 uchCmddBytesArr[], GU16 usCmddLen, GU8 uchReadBytesArr[], GU16 usMaxReadLen))
功能描述	注册 IIC/SPI 读写函数
输入参数	uchI2cIdLowTwoBitsSelect 地址设置，使用 EM_HBD_I2C_ID_SEL 枚举体定义 *pI2cWriteFunc 写函数 *pI2cReadFunc 读函数
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_PARAMETER_ERROR 参数异常

表 3-3 HBD_SetDelayUsCallback API 函数

函数原型	void HBD_SetDelayUsCallback(void (*pDelayUsFunc)(GU16 usUsec))
功能描述	注册微秒延时函数
输入参数	*pDelayUsFunc 延时函数
输出参数	无
返回值	无

表 3-4 HBD_LoadNewRegConfigArr API 函数

函数原型	GS8 HBD_LoadNewRegConfigArr(const ST_REGISTER stRegConfigArr[], GU16 usRegConfigLen)
功能描述	加载配置表。通常在启动测试前调用该函数
输入参数	StRegConfigArr[] 配置表数组 usRegConfigLen 数组长度（非字节长度）
输出参数	无
返回值	HBD_RET_OK 加载成功 HBD_RET_COMM_ERROR 加载失败

表 3-5 HBD_CommunicationInterfaceConfirm API 函数

函数原型	GS8 HBD_CommunicationInterfaceConfirm(void)
功能描述	检查 IIC/SPI 通信接口是否正常
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_COMM_ERROR 芯片通信异常 HBD_RET_COMM_NOT_REGISTERED_ERROR 未注册芯片通信接口

表 3-6 HBD_SimpleInit API 函数

函数原型	GS8 HBD_SimpleInit(ST_HBD_INIT_CONFIG_TYPE *stHbdInitConfigParam)
功能描述	初始化驱动
输入参数	*stHbdInitConfigParam　　初始化 ADT 配置
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_PARAMETER_ERROR 参数异常 HBD_RET_COMM_ERROR 芯片通信异常 HBD_RET_COMM_NOT_REGISTERED_ERROR 未注册芯片通信接口

表 3-7 HBD_HbDetectStart API 函数

函数原型	GS8 HBD_HbDetectStart(void)
功能描述	启动心率测试
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-8 HBD_InearEpWearDetectStart API 函数

函数原型	GS8 HBD_InearEpWearDetectStart(void)
功能描述	启动耳道耳机佩戴检测
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口 HBD_RET_GENERIC_ERROR 未启动佩戴检测功能，请先调用 HBD_EnableWearing 接口

表 3-9 HBD_InearEpDetectRecover API 函数

函数原型	GS8 HBD_InearEpDetectRecover(void)
功能描述	恢复耳道耳机佩戴检测。硬件调光失败后，调用该函数。
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口 HBD_RET_GENERIC_ERROR 未启动佩戴检测功能，请先调用 HBD_EnableWearing 接口

表 3-10 HBD_SoftWearDetectStart API 函数

函数原型	GS8 HBD_SoftWearDetectStart(void)

功能描述	启动软件佩戴检测
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口 HBD_RET_GENERIC_ERROR 未启动佩戴检测功能，请先调用 HBD_EnableWearing 接口

表 3-11 HBD_SoftWearDetectRecover API 函数

函数原型	GS8 HBD_SoftWearDetectRecover(void)
功能描述	恢复软件佩戴检测
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口 HBD_RET_GENERIC_ERROR 未启动佩戴检测功能，请先调用 HBD_EnableWearing 接口

表 3-1 HBD_HrvDetectStart API 函数

函数原型	GS8 HBD_HrvDetectStart(void)
功能描述	启动 HRV 测试
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-13 HBD_HbWithHrvDetectStart API 函数

函数原型	GS8 HBD_HbWithHrvDetectStart(void)
功能描述	启动心率加 HRV 测试
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-14 HBD_Stop API 函数

函数原型	GS8 HBD_Stop(void)
功能描述	停止运行，芯片进入休眠模式
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-15 HBD_AdtWearDetectStart API 函数

函数原型	GS8 HBD_AdtWearDetectStart(void)
功能描述	启动一次 ADT 佩戴检测
输入参数	无
输出参数	无

返回值	HBD_RET_OK 成功 HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-16 HBD_AdtWearContinuousDetectStart API 函数

函数原型	GS8 HBD_AdtWearContinuousDetectStart(void)
功能描述	启动持续的 ADT 佩戴检测，即在中断产生后仍继续进行 ADT 检测，检测模式（佩戴检测或脱落检测）会自动翻转
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit 接口

表 3-17 HBD_IsAdtWearDetectHasStarted API 函数

函数原型	GU8 HBD_IsAdtWearDetectHasStarted(void)
功能描述	检查是否已启动 ADT 佩戴检测
输入参数	无
输出参数	无
返回值	0：未启动 1：已启动

表 3-18 HBD_GetIntStatus API 函数

函数原型	GU8 HBD_GetIntStatus(void)
功能描述	获取中断类型
输入参数	无
输出参数	无
返回值	0：芯片复位中断 1：新数据就绪中断 2：达到 FIFO 阈值中断 3：满 FIFO 中断 4：佩戴中断 5：脱落中断 6：无效中断

表 3-19 HBD_HbCalculateByNewdataIntDbg API 函数

函数原型	GS8 HBD_HbCalculateByNewdataIntDbg(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 *pusRRvalue)
功能描述	用于新数据就绪中断处理的心率计算接口，支持 Rawdata 上传功能
输入参数	*stGsAxisValue　G-sensor 数据 emGsensorSensitivity　G-sensor 灵敏度
输出参数	*puchHbValue　心率值 *puchWearingState　佩戴状态 *puchWearingQuality　已废弃，但不能为空指针 *puchVoiceBroadcast　已废弃，但不能为空指针 *pusRRvalue　已废弃，但不能为空指针
返回值	0：出值未更新 1：出值已更新

表 3-20 HBD_HbCalculateByNewdataIntDbgEx API 函数

函数原型	GS8 HBD_HbCalculateByNewdataIntDbgEx(ST_GS_DATA_TYPE *stGsAxisValue,

	EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, ST_HB_RES *pstHbRes)
功能描述	用于新数据就绪中断处理的心率计算接口，支持 Rawdata 上传功能
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pstHbRes 见 ST_HB_RES
返回值	0：出值未更新 1：出值已更新

表 3-21 HBD_HbCalculateByFifoInt API 函数

函数原型	GU8 HBD_HbCalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 *pusRRvalue)
功能描述	用于 FIFO 中断处理的心率计算接口
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*puchHbValue 心率值 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 *pusRRvalue 已废弃，但不能为空指针
返回值	0：出值未更新 1：出值已更新

表 3-22 HBD_HbCalculateByFifoIntDebugOutputData API 函数

函数原型	GU8 HBD_HbCalculateByFifoIntDebugOutputData(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 *pusRRvalue, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	用于 FIFO 中断处理的心率计算接口，且输出 Rawdata
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*puchHbValue 心率值 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 *pusRRvalue 已废弃，但不能为空指针 *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	0：出值未更新 1：出值已更新

表 3-23 HBD_HbCalculateWithLvlByFifoInt API 函数

函数原型	GU8 HBD_HbCalculateWithLvlByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchAccuracyLevel, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 *pusRRvalue)

功能描述	FIFO 中断处理的心率计算接口，支持置信度输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*puchHbValue 心率值 *puchAccuracyLevel 置信度，取值为 1~3，值越大置信度越高 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 *pusRRvalue 已废弃，但不能为空指针
返回值	0：出值未更新 1：出值已更新

表 3-24 HBD_HbCalculateByFifoIntEx API 函数

函数原型	GS8 HBD_HbCalculateByFifoIntEx(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen, ST_HB_RES *pstHbRes)
功能描述	FIFO 中断处理的心率计算接口，支持置信度和 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pstHbRes 见结构体 ST_HB_RES
返回值	0：出值未更新 1：出值已更新

表 3-25 HBD_HbCalculateWithLvlByFifoIntDebugOutputData API 函数

函数原型	GS8 HBD_HbCalculateByFifoIntEx(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen, ST_HB_RES *pstHbRes)
功能描述	FIFO 中断处理的心率计算接口，支持置信度和 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*puchHbValue 心率值 *puchAccuracyLevel 置信度，取值为 1~3，值越大置信度越高 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 *pusRRvalue 已废弃，但不能为空指针 *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	0：出值未更新 1：出值已更新

表 3-26 HBD_HbWithHrvCalculateByNewdataInt API 函数

函数原型	GS8 HBD_HbWithHrvCalculateByNewdataInt(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 usRRvalueArr[4], GU8 *puchRRvalueCnt, GU8 *puchHrvConfidentLvl)
功能描述	新数据就绪中断处理的心率和 HRV 计算接口
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*puchHbValue 心率值 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 usRRvalueArr[4] RRI 值，1 秒至多 4 个 RRI *puchRRvalueCnt RRI 数量 *puchHrvConfidentLvl RRI 置信度。0：不可信；1：可信
返回值	0：出值未更新 1：出值已更新

表 3-27 HBD_HbWithHrvCalculateByNewdataIntEx API 函数

函数原型	GS8 HBD_HbWithHrvCalculateByNewdataIntEx(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, ST_HB_RES *pstHbRes, ST_HRV_RES *pstHrvRes)
功能描述	新数据就绪中断处理的心率和 HRV 计算接口
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pstHbRes 见 ST_HB_RES *pstHrvRes 见 ST_HRV_RES
返回值	0：出值未更新 1：出值已更新

表 3-28 HBD_HbWithHrvCalculateByFifoIntDbgData API 函数

函数原型	GS8 HBD_HbWithHrvCalculateByFifoIntDbgData(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchHbValue, GU8 *puchAccuracyLevel, GU8 *puchWearingState, GU8 *puchWearingQuality, GU8 *puchVoiceBroadcast, GU16 usRRvalueArr[4], GU8 *puchRRvalueCnt, GU8 *puchHrvConfidentLvl, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的心率和 HRV 计算接口，支持置信度和 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*puchHbValue 心率值 *puchAccuracyLevel 置信度，1 至 3，越大置信度越高 *puchWearingState 佩戴状态 *puchWearingQuality 已废弃，但不能为空指针 *puchVoiceBroadcast 已废弃，但不能为空指针 usRRvalueArr[4] RRI 值，1 秒最多 4 个 RRI *puchRRvalueCnt RRI 数量

	*puchHrvConfidentLvl RRI 置信度，0 不可信，1 可信 *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	0：出值未更新 1：出值已更新

表 3-29 HBD_HbWithHrvCalculateByFifoIntDbgDataEx API 函数

函数原型	GS8 HBD_HbWithHrvCalculateByFifoIntDbgDataEx(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, ST_HB_RES *pstHbRes, ST_HRV_RES *pstHrvRes, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的心率和 HRV 计算接口，支持置信度和 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pstHbRes 见 ST_HB_RES *pstHrvRes 见 ST_HRV_RES
返回值	0：出值未更新 1：出值已更新

表 3-30 HBD_EnableWearing API 函数

函数原型	void HBD_EnableWearing(GF32 fDirectionArr[3])
功能描述	使能软件佩戴检测
输入参数	fDirectionArr[3] 已废弃，可以为空指针
输出参数	无
返回值	无

表 3-31 HBD_HbAlgoScenarioConfig API 函数

函数原型	GS8 HBD_HbAlgoScenarioConfig(EM_HBA_SCENES uchScenario)
功能描述	设置心率测试场景。下次启动测试时生效
输入参数	uchScenario 场景值，请参考 5.1 心率场景定义
输出参数	无

表 3-32 HBD_HrvCalculateByNewdataInt API 函数

函数原型	GS8 HBD_HrvCalculateByNewdataInt(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr)
功能描述	新数据就绪中断处理的 HRV 计算接口
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI
返回值	RRI 个数

表 3-33 HBD_HrvCalculateByFifoInt API 函数

函数原型	GS8 HBD_HrvCalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr)
功能描述	FIFO 中断处理的 HRV 计算接口
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI

返回值	RRI 个数

表 3-34 HBD_HrvCalculateByFifoIntDbgRawdata API 函数

函数原型	GS8 HBD_HrvCalculateByFifoIntDbgRawdata(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的 HRV 计算接口，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	RRI 个数

表 3-35 HBD_HrvCalculateByFifoIntEx API 函数

函数原型	GS8 HBD_HrvCalculateByFifoIntEx(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, ST_HRV_RES *pstHrvRes, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的 HRV 计算接口，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rrawdata 的 Buffer 区长度（单位为帧）
输出参数	*pstHrvRes 见 ST_HRV_RES
返回值	RRI 个数

表 3-36 HBD_HrvCalculateByNewdataInt API 函数

函数原型	GS8 HBD_HrvCalculateByNewdataInt(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr)
功能描述	新数据就绪中断处理的 HRV 计算接口，支持置信度输出
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI *puchConfidentLvl RRI 置信度，0：不可信；1：可信
返回值	RRI 个数

表 3-37 HBD_HrvCalculateWithLvlByFifoInt API 函数

函数原型	GS8 HBD_HrvCalculateWithLvlByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr, GU8 *puchConfidentLvl)
功能描述	FIFO 中断处理的 HRV 计算接口，支持置信度输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI *puchConfidentLvl RRI 置信度，0：不可信；1：可信
返回值	RRI 个数

表 3-38 HBD_HrvCalculateWithLvlByFifoIntDbgRawdata API 函数

函数原型	GS8 HBD_HrvCalculateWithLvlByFifoIntDbgRawdata(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU16 *pusHrvRRValueArr, GU8 *puchConfidentLvl, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的 HRV 计算接口，支持置信度和 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pusHrvRRValueArr RRI 值，1 秒最多 4 个 RRI *puchConfidentLvl RRI 置信度，0：不可信；1：可信 *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	RRI 个数

表 3-39 HBD_InearEpWearDetectCalculateByNewdataInt API 函数

函数原型	GU8 HBD_InearEpWearDetectCalculateByNewdataInt(ST_GS_DATA_TYPE *stGsAxisValue, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchWearState)
功能描述	新数据就绪中断处理的耳道耳机佩戴检测
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*puchWearState 佩戴状态，1：佩戴，2：未佩戴
返回值	0：出值未更新 1：出值已更新

表 3-40 HBD_InearEpWearDetectCalculateByFifoInt API 函数

函数原型	GU16 HBD_InearEpWearDetectCalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchWearState)
功能描述	FIFO 中断处理的耳道耳机佩戴检测
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*puchWearState 佩戴状态，1：佩戴，2：未佩戴
返回值	0：出值未更新 1：出值已更新

表 3-41 HBD_InearEpWearDetectByFifoIntDbgRawdata API 函数

函数原型	GU16 HBD_InearEpWearDetectByFifoIntDbgRawdata(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *puchWearState, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的耳道耳机佩戴检测，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*puchWearState 佩戴状态，1：佩戴，2：未佩戴 *pusRawdataOutLen 实际输出的 Rawdata 帧数

返回值	0：出值未更新 1：出值已更新

表 3-42 HBD_GetFifoCntHasRead API 函数

函数原型	GU8 HBD_GetFifoCntHasRead(void)
功能描述	获取上次读取的 FIFO 数据帧数
输入参数	无
输出参数	无
返回值	上次读取的 FIFO 数据帧数

表 3-43 HBD_ChipReset API 函数

函数原型	GS8 HBD_ChipReset(void)
功能描述	复位芯片
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_COMM_NOT_REGISTERED_ERROR 未注册芯片通信接口

表 3-44 HBD_SetIrqPluseWidth API 函数

函数原型	GS8 HBD_SetIrqPluseWidth(GU8 uchIrqPluseWidth)
功能描述	设置中断脉冲宽度。下次启动测试时生效
输入参数	uchIrqPluseWidth　　脉冲宽度，不能为 0
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_PARAMETER_ERROR 参数错误

表 3-45 HBD_FifoConfig API 函数

函数原型	GS8 HBD_FifoConfig(GU8 uchMode, EM_HBD_FUNCTIONAL_STATE emFifoEnable)
功能描述	配置各测试模式下的 FIFO 开关，下次启动测试时生效
输入参数	uchMode　测试模式。0：HRV；1：心率和血氧 emFifoEnable　　0：禁用 FIFO；1：使能 FIFO
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_PARAMETER_ERROR 参数错误

表 3-46 HBD_SetFifoThrCnt API 函数

函数原型	GS8 HBD_SetFifoThrCnt(GU8 uchMode, GU16 usFifoCnt)
功能描述	配置各测试模式下的 FIFO 中断阈值，下次启动时测试生效
输入参数	uchMode　测试模式。0：HRV；1：心率；5：血氧；其它值：无效 usFifoCnt　　产生 FIFO 中断的阈值（单位为帧）
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_PARAMETER_ERROR 参数错误

表 3-47 *HBD_GetHbdVersion API 函数

函数原型	GS8 * HBD_GetHbdVersion(void)
功能描述	获取驱动库版本号字符串
输入参数	无
输出参数	无
返回值	版本号字符串

表 3-48 *HBD_GetHbaVersion API 函数

函数原型	GS8 * HBD_GetHbaVersion(void)
功能描述	获取心率算法版本号字符串
输入参数	无
输出参数	无
返回值	版本号字符串

表 3-49 *HBD_GetSpo2Version API 函数

函数原型	GU8 * HBD_GetSpo2Version (void)
功能描述	获取血氧算法版本号字符串
输入参数	无
输出参数	无
返回值	版本号字符串

表 3-50 HBD_GetSpo2AbnormalState API 函数

函数原型	GU8 HBD_GetSpo2AbnormalState(void)
功能描述	获取血氧出值状态
输入参数	无
输出参数	无

表 3-51 HBD_SpO2DetectStart API 函数

函数原型	GS8 HBD_SpO2DetectStart (void)
功能描述	启动血氧检测
输入参数	无
输出参数	无
返回值	HBD_RET_OK：成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，没有开 LED HBD_RET_NO_INITED_ERROR 未初始化，需要调用 HBD_SimpleInit

表 3-52 HBD_Spo2CalculateByFifoInt API 函数

函数原型	GU8 HBD_Spo2CalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *pusSpo2Value, GU8 *puchSpo2ConfidentLvl, GU8 *puchHbValue, GU8 *puchHbConfidentLvl, GU16 *pusHrvRRVal1,GU16 *pusHrvRRVal2,GU16 *pusHrvRRVal3,GU16 *pusHrvRRVal4, GU8 *puchHrvConfidentLvl, GU8 *puchHrvcnt,GU16 *pusSpo2RVal, GU8 *puchWearingState)
功能描述	FIFO 中断处理的血氧检测
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusSpo2Value 血氧值 *puchSpo2ConfidentLvl 血氧置信度，取值为 0 ~ 100，值越大置信度越好 *puchHbValue 心率值 *puchHbConfidentLvl 心率置信度 *pusHrvRRVal1 1 秒内 RRI 第一个间隔 *pusHrvRRVal2 1 秒内 RRI 第二个间隔 *pusHrvRRVal3 1 秒内 RRI 第三个间隔 *pusHrvRRVal4 1 秒内 RRI 第四个间隔 *puchHrvcnt 1 秒内 RRI 数量 *pusSpo2RVal 血氧 R 值

	*puchWearingState 佩戴状态
返回值	0：出值未更新 1：出值已更新

表 3-53 HBD_Spo2CalculateByNewdataInt API 函数

函数原型	GU8 HBD_Spo2CalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *pusSpo2Value, GU8 *puchSpo2ConfidentLvl, GU8 *puchHbValue, GU8 *puchHbConfidentLvl, GU16 *pusHrvRRVal1,GU16 *pusHrvRRVal2,GU16 *pusHrvRRVal3,GU16 *pusHrvRRVal4, GU8 *puchHrvConfidentLvl, GU8 *puchHrvcnt,GU16 *pusSpo2RVal, GU8 *puchWearingState)
功能描述	用于新数据就绪中断处理的血氧检测
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	*pusSpo2Value 血氧值 *puchSpo2ConfidentLvl 血氧置信度，取值为 0 ~ 100，值越大置信度越高 *puchHbValue 心率值 *puchHbConfidentLvl 心率置信度 *pusHrvRRVal1 1 秒内 RRI 第一个间隔 *pusHrvRRVal2 1 秒内 RRI 第二个间隔 *pusHrvRRVal3 1 秒内 RRI 第三个间隔 *pusHrvRRVal4 1 秒内 RRI 第四个间隔 *puchHrvcnt 1 秒内 RRI 数量 *pusSpo2RVal 血氧 R 值 *puchWearingState 佩戴状态
返回值	0：值未更新 1：出值已更新

表 3-54 HBD_Spo2CalculateByFifoIntDbgRawdata API 函数

函数原型	GU8 HBD_Spo2CalculateByFifoIntDbgRawdata(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *pusSpo2Value, GU8 *puchSpo2ConfidentLvl, GU8 *puchHbValue, GU8 *puchHbConfidentLvl, GU16 *pusHrvRRVal1,GU16 *pusHrvRRVal2, GU16 *pusHrvRRVal3,GU16 *pusHrvRRVal4, GU8 *puchHrvConfidentLvl, GU8 *puchHrvcnt,GU16 *pusSpo2RVal, GU8 *puchWearingState, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	用于 FIFO 中断处理的血氧检测，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pusSpo2Value 血氧值 *puchSpo2ConfidentLvl 血氧置信度，取值为 0 ~ 100，值越大置信度越高 *puchHbValue 心率值 *puchHbConfidentLvl 心率置信度 *pusHrvRRVal1 1 秒内 RRI 第一个间隔 *pusHrvRRVal2 1 秒内 RRI 第二个间隔 *pusHrvRRVal3 1 秒内 RRI 第三个间隔 *pusHrvRRVal4 1 秒内 RRI 第四个间隔

	*puchHrvcnt 1 秒内 RRI 数量 *pusSpo2RVal 血氧 R 值 *puchWearingState 佩戴状态 *pusRawdataOutLen 实际输出的 Rawdata 帧数
返回值	0：值未更新 1：出值已更新

表 3-55 HBD_Spo2CalculateByFifoIntDbgRawdataInnerUse API 函数

函数原型	GS8 HBD_Spo2CalculateByFifoIntDbgRawdataInnerUse(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GU8 *pusSpo2Value, GU8 *puchSpo2ConfidentLvl, GU8 *puchHbValue, GU8 *puchHbConfidentLvl, GU16 *pusHrvRRVal1,GU16 *pusHrvRRVal2,GU16 *pusHrvRRVal3,GU16 *pusHrvRRVal4, GU8 *puchHrvConfidentLvl, GU8 *puchHrvcnt,GU16 *pusSpo2RVal, GU8 *puchWearingState, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen, GU8 *puchValidLvl)
功能描述	用于 FIFO 中断处理的血氧检测，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pusSpo2Value 血氧值 *puchSpo2ConfidentLvl 血氧置信度 0-100，越大越好 *puchHbValue 心率值 *puchHbConfidentLvl 心率置信度 *pusHrvRRVal1 1 秒内 RRI 第一个间隔 *pusHrvRRVal2 1 秒内 RRI 第二个间隔 *pusHrvRRVal3 1 秒内 RRI 第三个间隔 *pusHrvRRVal4 1 秒内 RRI 第四个间隔 *puchHrvcnt 1 秒内 RRI 数量 *pusSpo2RVal 血氧 R 值 *puchWearingState 佩戴状态 *pusRawdataOutLen 实际输出的 Rawdata 帧数 *puchValidLvl 出值情况
返回值	0：出值未更新 1：出值已更新

表 3-56 HBD_Spo2CalculateByFifoIntEx API 函数

函数原型	GS8 HBD_Spo2CalculateByFifoIntEx(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen, ST_SPO2_RES *pstSpo2Res)
功能描述	用于 FIFO 中断处理的血氧检测，支持 Rawdata 输出
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 Rawdata 的 Buffer 区 *pusRawdataOutLen 保存 Rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pstSpo2Res 见 ST_SPO2_RES

返回值	0：出值未更新 1：出值已更新

表 3-57 HBD_AdtConfirmConfig API 函数

函数原型	void HBD_AdtConfirmConfig(GU16 usAdtConfirmGsThrVal, GU8 uchAdtConfirmGsCalcThrCntMax, GU8 uchAdtConfirmGsCalcThrCnt)
功能描述	配置 ADT 确认功能
输入参数	usAdtConfirmGsThrVal G-sensor 运动阈值 uchAdtConfirmGsCalcThrCntMax 计算帧数 uchAdtConfirmGsCalcThrCnt 确认佩戴计数阈值
输出参数	无
返回值	无

表 3-58 HBD_AdtConfirmStart API 函数

函数原型	GS8 HBD_AdtConfirmStart(void)
功能描述	启动 ADT 确认功能
输入参数	无
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit

表 3-59 HBD_AdtConfirmCalculateByFifoInt API 函数

函数原型	GU8 HBD_AdtConfirmCalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity)
功能描述	新数据就绪中断处理的 ADT 确认计算接口
输入参数	*stGsAxisValue G-sensor 数据 emGsensorSensitivity G-sensor 灵敏度
输出参数	无
返回值	0：计算中 0x11：已佩戴 0x12：未佩戴

表 3-60 HBD_AdtConfirmCalculateByFifoInt API 函数

函数原型	GU8 HBD_AdtConfirmCalculateByFifoInt(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity)
功能描述	FIFO 中断处理的 ADT 确认计算接口
输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度
输出参数	无
返回值	0：计算中 0x11：已佩戴 0x12：未佩戴

表 3-61 HBD_AdtConfirmCalculateByFifoIntDbgOutputData API 函数

函数原型	GU8 HBD_AdtConfirmCalculateByFifoIntDbgOutputData(ST_GS_DATA_TYPE stGsAxisValue[], GU16 usGsDataNum, EM_HBD_GSENSOR_SENSITIVITY emGsensorSensitivity, GS32 nRawdataOut[][6], GU16 *pusRawdataOutLen)
功能描述	FIFO 中断处理的 ADT 确认计算接口，支持 Rawdata 输出

输入参数	stGsAxisValue[] G-sensor 数据 usGsDataNum G-sensor 数据帧数 emGsensorSensitivity G-sensor 灵敏度 nRawdataOut[][6] 保存 rawdata 的 Buffer 区 *pusRawdataOutLen 保存 rawdata 的 Buffer 区长度（单位为帧）
输出参数	*pusRawdataOutLen 实际输出的 rawdata 帧数
返回值	0：计算中 0x11：已佩戴 0x12：未佩戴

表 3-62 HBD_StartHBDOnly API 函数

函数原型	GS8 HBD_StartHBDOnly(GU16 usSampleRate, GU8 ucEnableFifo, GU16 ucFifoTHR)
功能描述	纯采集功能启动，即仅启动数据采集，不进行算法处理。
输入参数	usSampleRate 采样率 ucEnableFifo 1：使能 FIFO；0：禁用 FIFO ucFifoTHR FIFO 中断阈值
输出参数	无
返回值	HBD_RET_OK 成功 HBD_RET_LED_CONFIG_ALL_OFF_ERROR 配置异常，未打开 LED HBD_RET_NO_INITED_ERROR 未初始化，需调用 HBD_SimpleInit

表 3-63 HBD_ SetLedCurrent API 函数

函数原型	GS8 HBD_SetLedCurrent(GF32 fLed0Current, GF32 fLed1Current);
功能描述	纯采集功能下，设置电流接口。
输入参数	fLed0Current led0 电流 fLed1Current led1 电流
输出参数	无
返回值	HBD_RET_PARAMETER_ERROR: 超过可设置的电流阈值 HBD_RET_GENERIC_ERROR: 此模式不能设置电流 HBD_RET_OK: 设置电流成功

表 3-64 HBD_ GetLedCurrent API 函数

函数原型	GS8 HBD_GetLedCurrrent(GF32 *pLed0Current, GF32 *pLed1Current);
功能描述	纯采集功能下，获取电流接口。
输入参数	fLed0Current led0 电流指针 fLed1Current led1 电流指针
输出参数	无
返回值	HBD_RET_GENERIC_ERROR: 此模式下不能获取电流 HBD_RET_OK: 获取成功

表 3-65 HBD_GetRawdataByFifoInt API 函数

函数原型	GU8 HBD_GetRawdataByFifoInt(GU16 ucBufLen, GS32 nRawdataOut[][2], GU16 *pucRealDataLen)
功能描述	用于 FIFO 中断处理的获取纯采集功能的 FIFO 数据
输入参数	ucBufLen 保存数据的 Buffer 帧长度，1 帧为两个 PPG 数据 nRawdataOut[] 保存数据的 Buffer，数组大小应为 ucBufLen*2 *pucRealDataLen 实际输出的数据帧数
输出参数	无
返回值	0：获取成功 1：软件调光失败

表 3-66 HBD_GetRawdataByNewDataInt API 函数

函数原型	GU8 HBD_GetRawdataByNewDataInt(GU32 *ppg1, GU32 *ppg2)
功能描述	新数据就绪中断处理的纯采集功能接口
输入参数	*ppg1　PPG1 数据 *ppg2　PPG2 数据
输出参数	无
返回值	0：获取成功 1：软件调光失败

表 3-67 HBD_CheckMemEnable API 函数

函数原型	void HBD_CheckMemEnable(EM_HBD_FUNCTIONAL_STATE uchCheckMemEnable)
功能描述	使能内存检测
输入参数	uchCheckMemEnable 0：关闭 1：使能
输出参数	无

表 3-68 HBD_EnableHBAlgo API 函数

函数原型	void HBD_EnableHBAlgo(EM_HBD_FUNCTIONAL_STATE emState)
功能描述	心率算法开关
输入参数	emState　见结构体 EM_HBD_FUNCTIONAL_STATE HBD_FUNCTIONAL_STATE_DISABLE：关闭 HBD_FUNCTIONAL_STATE_ENABLE：开启
输出参数	无
返回值	无

4 常见问题及注意事项

1. 更换寄存器数组在哪个文件？对应模式的寄存器数组是哪些？

   gh30x_example_reg_array.c 文件；

   - hb_adt_confirm_reg_config：心率 ADT 检测寄存器数组
   - hb_reg_config_array：心率检测寄存器数组
   - spo2_reg_config_array：血氧检测寄存器数组

   需要进行寄存器数组更改时，目前只需关注以上数组的更改，常见的修改频率的寄存器地址为：0x0016；对应的频率为：0x51e(25HZ)、0x0147(100HZ)。

2. 算法对于 G-sensor 的要求？

   G-sensor 的传入数量一定需要大于 Rawdata 的数量，否则，心率计算接口等会吐出 0 值；确保 G-sensor 的频率大于或等于心率 IC 的采样率，并确保时间轴上对齐。具体可参考 2.3.1 G-sensor 调试。

3. 怎么确认通信正常？

   不包含 Goodix 驱动库，可按以下步骤操作：

   (1) 实现 IIC/SPI 接口函数。

   (2) IIC 地址为 0x28（8 bits），按照 IIC 发送命令协议发送 0XDD 0xDD 0xC0 命令，延时 1 ms 后，按照 IIC 读操作协议从地址为 0x0028 的寄存器中读取 2 个字节，若得到值为 0x0031，则通信接口验证成功。

   (3) 按照 SPI 发送命令协议发送 0xC0 命令，延时 1 ms 后，按照 SPI 读操作协议从地址为 0x0028 的寄存器中读取 2 个字节，若读取的值为 0x0031，则通信接口验证成功。

   包含 Goodix 驱动库时，可按以下步骤操作：

   (1) 实现 IIC/SPI 接口函数。

   (2) 调用 Goodix 驱动库的 HBD_SetI2CRW/ HBD_SetSPIRW 函数将 IIC 接口注册到库中。

   (3) 调用 Goodix 驱动库的 HBD_CommunicationInterfaceConfirm 接口，若返回 HBD_RET_OK，则通信接口验证成功。

4. 对 IIC 速率有什么要求？

   最大支持 400 kHz，一般使用 400 kHz。

   当使用模拟 IIC 的时候，可能会存在读取 IIC 数据不及时的问题，所以建议使用硬件 IIC。

5. 中断触发周期与模式是什么？

   正常工作情况下，中断触发周期是 1s；中断触发方式设置为上升沿触发。

   默认的采样率设置为：心率检测 25 Hz，血氧检测 100 Hz，当芯片内数据达到 FIFO 水线的时候，则会触发中断信号，以下是设置心率水线的宏，默认配置无需改动。

```
#define __HB_FIFO_THR_CNT_CONFIG__ (25)   //心率
```

6. 怎么获取原始数据？

   以从心率计算接口获取为例：

```
HBD_AdtConfirmCalculateByFifoIntDbgOutputData(gsensor_soft_fifo_buffer,
gsensor_soft_fifo_buffer_index, __GS_SENSITIVITY_CONFIG__, (GS32
(*)[DBG_MCU_MODE_PKG_LEN])dbg_rawdata_ptr, &dbg_rawdata_len);
```

调用纯数据采集接口，也可以从纯获取 PPG 接口获取：handle_getrawdata_mode_result。

获取的原始数据为 24 bits，实际的 PPG 数据为 bit0 ~ bit16。

7. 怎么配置心率算法场景？
   详见 **2.2.3 心率模块章节**
8. 驱动库接口调用说明

   驱动库接口调用的注意事项如下：

   (1) 驱动库的接口有些是不能重入的，重入会导致系统异常。

   比如算法计算相关的接口，列举如下：

   - HBD_HbCalculateByFifoIntEx()
   - HBD_AdtConfirmCalculateByFifoIntDbgOutputData()
   - HBD_Spo2CalculateByFifoIntEx()
   - HBD_HbWithHrvCalculateByFifoIntDbgDataEx()

   比如硬件读写操作接口，列举如下：

   - gh30x_i2c_write_exchange_to_spi()
   - gh30x_i2c_read_exchange_to_spi()

   (2) 驱动库的某些接口在退出之前不能被其他接口打断，否则会引起系统异常。

   例如下列算法接口：

   HBD_AdtWearDetectStart()

   - HBD_AdtConfirmStart()
   - HBD_HbDetectStart()
   - HBD_WearStateConfirmStart()
   - HBD_HrvDetectStart()
   - HBD_SpO2DetectStart()

   (3) 中断函数不能被打断，否则修改了全部变量，导致整个流程错乱。

   比如 G-sensor 中断函数、心率中断函数：

   - hal_gsensor_drv_int1 handler()
   - gh30x_int_msg handler()

全局变量 gsensor_drv_motion_det_mode，在 Gsensor 中断函数中判断状态，在多个函数中赋值，如果发生冲突会导致逻辑错误。

说明：

心率的所有接口调用，推荐采用单任务方式，心率的所有接口调用都在一个任务中执行，中断也建议通过消息发给单线程任务来执行中断函数体；对于多任务的调用，需对部分接口加互斥锁。

9. 睡眠状态下，异常脱落问题优化说明

睡眠标志传入前出现误脱落：活体检测识别为活体后，将活体脱落的阈值降低；

睡眠标志传入后出现对物不脱落：若出现脱落情况，需清除应用层睡眠标记，避免下次进入活体检测后，处于睡眠模式，导致对物不脱落情况发生。.

5 附录

5.1 心率场景定义

心率模式可以设置场景以提高准确性，每次在 gh30x_module_start 前调用 GH30X_HBA_SCENARIO_CONFIG 即可完成设置，参数范围 0 ~ 23，具体描述如下：

表 5-1 心率场景参数说明

参数	说明
0	Default（由算法内部识别处理）
1	Routine（日常生活）
2	Indoor running（室内跑步）
3	Indoor walking（室内步行）
4	Stair activity（上下楼梯）
5	Outdoor running（户外跑步）
6	Outdoor walking（户外步行）
7	Tranquillization（静息）
8	Rest（休息）
9	Short of breath（憋气）
10	Indoor cycling（室内自行车）
11	Outdoor cycling（室外自行车）
12	Bicycle motocross（室外自行车越野）
13	High heart rate running（高心率跑步）
14	Combination running of treadmill（跑步机组合跑）
15	High intensity exercise combination（高强度运动组合）
16	Traditional strength training combination（传统力量训练组合）
17	Step-on testing（台阶测试）
18	Ball game（球类运动）
19	Aerobics（健身操）
20	Sleep（睡眠）
21	Jump rope（跳绳）
22	Cordless jump rope（无绳跳绳）
23	Swimming（游泳）