G-Sensor Algorithm Interface

（对应算法版本3.9.0）

**1. 算法概述**

RTK（Realtek）的GSA算法（G-Sensor Algorithm），使用三轴加速计来对运动、健康进行监测。具体功能包括：

（1）计步算法。记录日常走路、跑步的步数，以及运动类型、路程、卡路里消耗。

（2）睡眠监测。监测全天的睡眠状态，包括“清醒、浅睡、深睡”。

（3）动作监测。捕捉特定的手势动作，包括“抬腕（抬/放）、转腕、摇一摇”。

算法接口定义在rtk\_gsa.h中，并以lib形式（health\_algorithm.lib）提供了具体实现。算法版本可通过如下接口获得：

**typedef struct {**

**uint8 major;**

**uint8 minor;**

**uint16 revision;**

**} gsa\_ver\_t;**

**gsa\_ver\_t\* rtk\_gsa\_get\_ver(void);**

**2. 算法初始化**

算法初始化或重置时，需要传入一些必要的信息，包括用户信息、G-Sensor信息、回调函数。

**void rtk\_gsa\_init(**

**usr\_prof\_t\* prof,**

**gsa\_gs\_inf\_t\* gs\_inf,**

**gsa\_cbs\_t\* cbs);**

**2.1 用户信息**

用户信息包括身高、体重、年龄、性别，身高体重的精度是二进制小数点后1位。

**typedef struct {**

**uint32 reserve : 5;**

**uint32 weight : 10; // Q1, kg**

**uint32 heigh : 9; // Q1, cm**

**uint32 age : 7;**

**uint32 gender : 1; // 0-female, 1-male**

**} usr\_prof\_t;**

**2.2 G-Sensor信息**

G-Sensor信息包括X/Z轴的方向、采样率。根据右手定律，确定2个轴的方向便能知道第3轴，故这里仅需配置X/Z轴。Z轴一般与表面垂直，X/Y轴一般在表平面内，故共有2\*4=8种情况。

G-Sensor采样率ODR表示平均每秒采样次数，精度为二进制小数点后10位。为保证精度，请计算足够长时间段（3分钟以上）内的平均每秒采样数。考虑到Sensor的个体差异，建议在运行中计算实际的ODR，并重置算法。

**typedef enum {**

**GS\_XPOS\_RIGHT,**

**GS\_XPOS\_DOWN,**

**GS\_XPOS\_LEFT,**

**GS\_XPOS\_UP**

**} gs\_xpos\_e;**

**typedef enum {**

**GS\_ZPOS\_UP,**

**GS\_ZPOS\_DOWN**

**} gs\_zpos\_e;**

**typedef struct {**

**gs\_xpos\_e xpos;**

**gs\_zpos\_e zpos;**

**uint32 odr; // Q10**

**} gsa\_gs\_inf\_t;**



**2.3 回调函数**

算法会尽量实时地把结果反馈给SDK，而SDK需要为计步、睡眠、手势三个模块注册回调函数。

**typedef void (\*gsa\_pedo\_cb) (gsa\_pedo\_info\_t\*);**

**typedef void (\*gsa\_sleep\_cb) (gsa\_sleep\_stt\_e, uint32);**

**typedef void (\*gsa\_act\_cb) (gsa\_act\_type\_e, uint16);**

**typedef bool (\*gsa\_wear\_chk) (void);**

**typedef void (\*gsa\_sleep\_stat) (uint16);**

**typedef struct {**

**gsa\_pedo\_cb pedo\_cb;**

**gsa\_sleep\_cb sleep\_cb;**

**gsa\_act\_cb act\_cb;**

**gsa\_wear\_chk wear\_chk;**

**gsa\_sleep\_stat sleep\_stat;**

**} gsa\_cbs\_t;**

计步算法返回实时的运动状态、步数、路程、卡路里、速度、步频，请注意返回单位和精度。睡眠算法返回睡眠状态，以及该状态的起始时间与当前时间的偏移（分钟）。手势算法返回捕捉到的动作，包括抬腕（第二参数的0/1分别对应抬/放）、转腕、摇一摇（后两者的第二参数为0）。 wear\_chk检查佩戴状态（非必须实时，不确定时先返回TRUE），如不支持请置为NULL。如要细致分析睡眠问题，需注册函数sleep\_stat，它返回每分钟的活动值。请通过APP把这些log保存下来，以供开发人员分析（release版本置NULL）。

**typedef enum {**

**SPORT\_MODE\_WALK,**

**SPORT\_MODE\_RUN,**

**SPORT\_MODE\_INVALID**

**} gsa\_sport\_mode\_e;**

**typedef struct {**

**gsa\_sport\_mode\_e mode;**

**uint16 steps;**

**uint16 distance; // Q4, cm**

**uint16 speed; // Q8, km/h**

**uint16 frequency; // Q6, steps/m**

**uint16 calories; // Q2, cal**

**} gsa\_pedo\_info\_t;**

**typedef enum {**

**GSA\_SLEEP\_OFF,**

**GSA\_SLEEP\_DEEP,**

**GSA\_SLEEP\_SLEEP,**

**GSA\_SLEEP\_WAKE**

**} gsa\_sleep\_stt\_e;**

**typedef enum {**

**GSA\_ACT\_LIFT,**

**GSA\_ACT\_TWIST,**

**GSA\_ACT\_WAVE**

**} gsa\_act\_type\_e;**

**3. 算法运行/控制**

SDK将3轴的加速度传给算法状态机，便能驱动算法运行。算法目前接受的数据格式是int16\*3的数组，数据精度为：2^9等于加速度1G。

**void rtk\_gsa\_fsm(int16\* gsa\_accs);**

依据活动量的睡眠判断无法满足实时性，如果用户想立刻获得之前的状态，可以发起一次主动唤醒。该接口把缓存的状态都为“清醒”，需谨慎使用。

**void rtk\_gsa\_sleep\_wakeup(void);**

手环马达的震动会导致G-Sensor数据出现波动，从而对睡眠算法造成干扰。在马达打开之前以及关闭之后，需要通知算法库，以便采取适当的屏蔽策略。

**void rtk\_gsa\_motor\_state(bool is\_on);**

手势算法的开关接口，可控制是否捕捉特定动作（目前抬腕、转腕捆绑在一起，需同时置否）。算法初始化时，手势算法会全部关闭，需重新手动控制开关。

**void rtk\_gsa\_act\_switch(gsa\_act\_type\_e type, bool turn\_on);**