GOSUNCN 高新兴物联

Welink your smart

ME3616

WelinkOpen API编程指南

Version: V1.1

Date: 2018-05-31

NB-IoT Module

Website: www.gosuncnwelink.com

E-mail: welink@gosuncn.com

修订记录

Version	Date	Note
V1.0	2018-04-18	初始版本
V1.1	2018-05-31	更新函数描述表格 添加 pwm，i2c，alarm，spi，iot，wefota API 接口说明 删除 rtc API 接口说明

目录

1 简介

1.1 范围

本文描述了模块产品ME3616支持的WelinkOpen API接口，该接口可以使用的户应用程序直接访问平台提供的服务。

1.2 读者

该文档适用于使用API接口进行WelinkOpen开发的软件开发人员，并需要读者对FreeRTOS编程有一定程度的了解。

1.3 说明

在该文档中涵盖的使用外设的API接口中，如使用不当会返回相应的错误码，错误码的含义在具体接口介绍中已表明意义，更详细信息还请参考GSDK API头文件注释。

2 FreeRTOS API介绍

本章包含FreeRTOS系统函数API接口，包含TASK/QUEUE/EVENT/TIMER/LIST等相关API函数介绍，旨在使户更好的使用该RTOS的相关功能。

2.1 Task相关函数

2.1.1 eTaskConfirmSleepModeStatus

Function	eSleepModeStatus eTaskConfirmSleepModeStatus (void)		
描述	该函数为确认睡眠模式状态。		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	返回睡眠模式。 Typedef enum eSleepModeStauts		
	Type	**参数**	**描述**
	void	eAbortSleep	终止进入睡眠模式
	void	eStandardSleep	进入睡眠模式,不会持续时间比预期的空闲时间长
	void	eNoTasksWaiting Timeout	有任务等待超时,所以它是安全的进入睡眠模式,只能通过外部中断退出

2.1.2 eTaskGetState

Function	eTaskState eTaskGetState(TaskHandle_t xTask)		
描述	该函数为获取任务的状态。		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTask	需要获得状态的任务
返回值	返回任务状态。 Typedef enum eTaskState		
	Type	**参数**	**描述**

	void	eRuning	一个任务是查询本身的状态,所以必须运行态
	void	eReady	正在查询的任务是在阅读或等待阅读列表
	void	eBlocked	任务查询处于阻塞状态
	void	eSuspend	查询的任务是在挂起状态,或处于阻塞状态无限时间
	void	eDeleted	被查询的任务已被删除,但其TCB尚未释放

2.1.3 *pcTaskGetTaskName

Function	char *pcTaskGetTaskName(TaskHandle_t xTaskToQuery)		
描述	该函数为获取任务描述内容。		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTaskToQuery	任务的句柄
返回值	一个指针，指向任务描述字符串。		

2.1.4 *pvTaskIncrementMutexHeldCount

Function	void *pvTaskIncrementMutexHeldCount(void)		
描述	该函数为将任务控制块中的成员变量uxMutexesHeld加1，表示当前任务获取到一个互斥信号量		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	无		

2.1.5 ulTaskNotifyTake

Function	uint32_t ulTaskNotifyTake(BaseType_t xClearCountOnExit, TickType_t xTicksToWait)
描述	该函数为获取通知。

参数	Type	参数	描述
	BaseType_t	xClearCountOnExit	如果该参数设置为pdFALSE，则API函数xTaskNotifyTake()退出前，将任务的通知值减1；如果该参数设置为pdTRUE，则API函数xTaskNotifyTake()退出前，将任务通知值清零
	TickType_t	xTicksToWait	因等待通知而进入阻塞状态的最大时间
返回值	返回任务的当前通知值，为0或者为调用API函数xTaskNotifyTake()之前的通知值减1。		

2.1.6 uxTaskGetBottomOfStack

Function	UBaseType_t uxTaskGetBottomOfStack(TaskHandle_t xTaskHandle)		
描述	该函数为获取栈底		
参数	Type	参数	描述
	TaskHandle_t	xTaskHandle	任务句柄
返回值	返回栈底		

2.1.7 uxTaskGetNumberOfStack

Function	UBaseType_t uxTaskGetNumberOfStack(TaskHandle_t xTaskHandle)		
描述	该函数为获取任务状态的个数		
参数	Type	参数	描述
	TaskHandle_t	xTaskHandle	任务句柄
返回值	返回uxCurrentNumberOfTasks		

2.1.8 uxTaskGetStackHighWaterMark

Function	UBaseType_t uxTaskGetStackHighWaterMark(TaskHandle_t xTask)		
描述	该函数为获取任务堆栈最大使用深度		
参数	Type	参数	描述

	TaskHandle_t	xTask	任务句柄
返回值	返回最小剩余堆栈空间，以字节为单位		

2.1.9 uxTaskGetSystemState

Function	UBaseType_t uxTaskGetSystemState(TaskStatus_t *const pxTaskStatusArray, const UBaseType_t uxArraySize, uint32_t *const pulTotalRunTime)		
描述	该函数为获取任务系统状态		
参数	**Type**	**参数**	**描述**
	TaskStatus_t	pxTaskStatusArray	指向TaskStatus_t类型的结构体数组
	UBaseType_t	uxArraySize	pxTaskStatusArray数组的大小
	uint32_t	pulTotalRunTime	运行总时间
返回值	被填充的TaskStatus_t结构体数量。		

2.1.10 uxTaskGetTaskNumber

Function	UBaseType_t uxTaskGetTaskNumber(TaskHandle_t xTask)		
描述	该函数为获取任务的任务号		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTask	需要获取任务号的任务
返回值	xTask任务号		

2.1.11 uxTaskPriorityGet

Function	UBaseType_t uxTaskPriorityGet(TaskHandle_t xTask)		
描述	该函数为获取任务的优先级		
参数	**Type**	**参数**	**描述**

	TaskHandle_t	xTask	需要处理的任务
返回值	xTask的优先级		

2.1.12 uxTaskResetEventItemValue

Function	TickType_t uxTaskResetEventItemValue(void)		
描述	该函数为重置事件列表的正常值，因此可用于队列和信号量		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	事件列表值		

2.1.13 vTaskClearTaskRunTimeCounter

Function	void vTaskClearTaskRunTimeCounter(void)		
描述	该函数为清除任务运行的节拍计数		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	无		

2.1.14 vTaskDelay

Function	void vTaskDelay(const Tick_type_t xTicksToDelay)		
描述	该函数为延时任务		
参数	**Type**	**参数**	**描述**
	Tick_type_t	xTicksToDelay	时间数量，调用任务应该锁住的时间片周期
返回值	无		

2.1.15 vTaskDelayUntil

Function	void vTaskDelayUntil(TickType_t *const pxPreviousWakeTime, const TickType_t xTimeIncrement)		
描述	该函数为延时一个任务到指定的时间		
参数	Type	参数	描述
	TickType_t	pxPreviousWakeTime	保存了任务上一次离开阻塞态（被唤醒）的时刻。这个时刻被作用作一个参考点来计算该任务下一次离开阻塞的时刻
	TickType_t	xTimeIncrement	循环周期时间
返回值	无		

2.1.16 vTaskDelete

Function	void vTaskDelete(TaskHandle_t xTaskToDelete)		
描述	该函数为从RTOS实时内核管理中移除任务		
参数	Type	参数	描述
	TaskHandle_t	xTaskToDelete	处理要删除的任务
返回值	无		

2.1.17 vTaskEndScheduler

Function	void vTaskEndScheduler(void)		
描述	该函数为停止实时内核运行，所有创建的任务将自动删除，并且多任务（优先级或合作式）将停止		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.1.18 vTaskGetRunTimeStats

Function	void vTaskGetRunTimeStats(char *pcWriteBuffer)		
描述	该函数为获取任务运行时间		
参数	Type	参数	描述
	char	pcWriteBuffer	任务的运行时间信息会写入这个缓冲区
返回值	无		

2.1.19 vTaskList

Function	void vTaskList(char *pcWriteBuffer)		
描述	该函数为获取所有任务详情		
参数	Type	参数	描述
	char	pcWriteBuffer	任务的信息会写入这个缓冲区
返回值	无		

2.1.20 vTaskMissedYield

Function	void vTaskMissedYield(void)		
描述	该函数为设置xYieldPending为真		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.1.21 vTaskPriorityInherit

Function	void vTaskPriorityInherit(TaskHandle_t const pxMutexHolder)		
描述	该函数为修改拿锁任务的优先级。		
参数	Type	参数	描述
	TaskHandle_t	pxMutexHolder	该函数比较当前任务（拿锁失败，阻塞等待）和

			拿锁任务优先级，进行优先级继承处理
返回值	无		

2.1.22 vTaskPrioritySet

Function	void vTaskPrioritySet(TaskHandle_t xTask, UBaseType_t uxNewPriority)		
描述	该函数为设置任务优先级		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTask	需要设置优先级的任务
	UBaseType_t	uxNewPriority	任务需要设置的优先级
返回值	无		

2.1.23 vTaskResume

Function	void vTaskResume(TaskHandle_t xTaskToResume)		
描述	该函数为唤醒挂起的任务。		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTaskToResume	就绪的任务的句柄
返回值	无		

2.1.24 vTaskSetTaskNumber

Function	void vTaskSetTaskNumber(TaskHandle_t xTask, const UBaseType_t uxHandle)		
描述	该函数为设置任务号		
参数	**Type**	**参数**	**描述**
	TaskHandle_t	xTask	任务句柄
	UBaseType_t	uxHandle	任务号
返回值	无		

2.1.25 vTaskSetTimeOutState

Function	void vTaskSetTimeOutState(TimeOut_t * const pxTimeOut)		
描述	该函数为设置超时状态，用于设置初始状态。		
参数	Type	参数	描述
	TimeOut_t	pxTimeOut	指向一个结构体的指针，该结构体用来保存确定超时是否发生的必要信息
返回值	无		

2.1.26 vTaskStartScheduler

Function	void vTaskStartScheduler(void)		
描述	该函数为启动内核实时处理，在调用之后，内核已经控制执行的任务。		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.1.27 vTaskStepTick

Function	void vTaskStepTick(const TickType_t xTickToJump)		
描述	该函数为更改内核的嘀嗒计数		
参数	Type	参数	描述
	TickType_t	xTickToJump	计数值
返回值	无		

2.1.28 vTaskSuspend

Function	void vTaskSuspend(TaskHandle_t xTaskToSuspend)		
描述	该函数为挂起任务		
参数	Type	参数	描述
	TaskHandle_t	xTaskToSuspend	处理需要挂起的任务
返回值	无		

2.1.29 vTaskSuspendAll

Function	void vTaskSuspendAll(void)		
描述	该函数为挂起所有活动的实时内核，同时允许中断		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.1.30 vTaskSwitchContext

Function	void vTaskSwitchContext(void)		
描述	该函数为完成选择下一个运行的任务		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.1.31 xTaskCheckForTimeOut

Function	BaseType_t xTaskCheckForTimeOut(TimeOut_t *const pxTimeOut, TickType_t * const pxTicksToWait)		
描述	该函数为检测阻塞时间是否超过总超时时间。		
参数	Type	参数	描述

	TimeOut_t	pxTimeOut	指向一个结构体的指针。该结构体保存确定超时是否发生的必要信息
	TickType_t	pxTicksToWait	TickType_t指针，指向的变量保存总超时时间
返回值	pdTRUE -超时。 pdFALSE -未超时。		

2.1.32 xTaskGenericCreate

Function	signed portBASE_TYPE xTaskGenericCreate(pdTASK_CODE pxTaskCode, const signed char * const pcName, unsigned short usStackDepth, void *pvParameters, unsigned portBASE_TYPE uxPriority, xTaskHandle *pxCreatedTask, portSTACK_TYPE *puxStackBuffer, const xMemoryRegion * const xRegions)		
描述	该函数为用来建立一个任务		
参数	**Type**	**参数**	**描述**
	pdTASK_CODE	pxTaskCode	指向任务的入口函数，任务必须执行并且永不返回（即无限死循环）
	char	pcName	描述任务的名字。这个参数不会被FreeRTOS 使用。其只是单纯地用于辅助调试。识别一个具有可读性的名字总是比通过句柄来识别容易得多。应用程序可以通过定义常量config_MAX_TASK_NAME_LEN 来定义任务名的最大长度——包括‘\0’结束符。如果传入的字符串长度超过了这个最大值，字符串将会自动被截断
	unsigned short	usStackDepth	当任务创建时，内核会分为每个任务分配属于任务自己的唯一状态。usStackDepth 值用于告诉内核为它分配多大的栈空间。这个值指定的是栈空间可以保存多少个字(word)，而不是多少个字节(byte)。比如说，如果是32 位宽的栈空间，传入的usStackDepth 值为100，则将会分配400 字节的栈空间(100 * 4bytes)。栈深度乘以栈宽度的结果千万不能超过一个size_t 类型变量所能表达的最大值。
	void	pvParameters	任务函数接受一个指向void 的指针(void*)。pvParameters 的值即是传递到任务中的值

	portBASE_TYPE	uxPriority	指定任务优先级。优先级的取值范围可以从最低优先级0 到最高优先级(configMAX_PRIORITIES – 1)
	xTaskHandle	pxCreatedTask	用于传出任务的句柄。这个句柄将在API 调用中对该创建出来的任务进行引用，比如改变任务优先级，或者删除任务。如果应用程序中不会用到这个任务的句柄，则pxCreatedTask 可以被设为NULL
	portSTACK_TYPE	puxStackBuffer	堆栈缓冲区
	xMemoryRegion	xRegions	缓存区
返回值	1. pdTRUE表明任务创建成功。 2. errCOULD_NOT_ALLOCATE_REQUIRED_MEMORY由于内存堆空间不足，FreeRTOS 无法分配足够的空间来保存任务结构数据和任务栈，因此无法创建任务		

2.1.33 xTaskGetCurrentTaskHandle

Function	TaskHandle_t xTaskGetCurrentTaskHandle(void)		
描述	该函数为获取当前任务句柄		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	当前任务（调用该函数的任务）的句柄		

2.1.34 xTaskGetSchedulerState

Function	BaseType_t xTaskGetSchedulerState(void)		
描述	该函数为获取调度器状态		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	返回值是以下常量之一： taskSCHEDULER_NOT_STARTED（未启动）。 taskSHCEDULER_RUNNING（正常运行）。 taskSCHEDULER_SUSPENDED（挂起）。		

2.1.35 xTaskGetTickCount

Function	TickType_t xTaskGetTickCount(void)		
描述	该函数为获取系统节拍次数		
参数	Type	参数	描述
	void	void	无
返回值	返回从vTaskStartScheduler函数调用后的系统时钟节拍次数。		

2.1.36 xTaskIncrementTick

Function	BaseType_t xTaskIncrementTick(void)		
描述	该函数为系统中断服务程序会调用该函数来完成主要的工作		
参数	Type	参数	描述
	void	void	无
返回值	如果该函数返回值为真（不等于pdFALSE），说明处于就绪态任务的优先级比当前运行的任务优先级高。		

2.1.37 xTaskNotify

Function	BaseType_t xTaskNotify(TaskHandle_t xTaskToNotify, uint32_t ulValue, eNotifyAction eAction)		
描述	该函数为发送通知		
参数	Type	参数	描述
	TaskHandle_t	xTaskToNotify	被通知的句柄
	uint32_t	ulValue	通知更新值
	eNotifyAction	eAction	指明更新通知值的方法
返回值	参数eAction为eSetValueWithoutOverwrite时，如果被通知任务还没取走上一个通		

	知，又接收到了一个通知，则这次通知值未能更新并返回pdFALSE，否则返回pdPASS。

2.1.38 xTaskNotifyWait

Function	BaseType_t xTaskNotifyWait(uint32_t ulBitsToClearOnEntry, uint32_t ulBitsToClearOnExit, uint32_t *pulNotificationValue, TickType_t xTicksToWait)		
描述	该函数为等待通知		
参数	**Type**	**参数**	**描述**
	uint32_t	ulBitsToClearOnEntry	在使用通知之前，先将任务的通知值与参数ulBitsToClearOnEntry的按位取反值按位与操作。设置参数ulBitsToClearOnEntry为0xFFFFFFFF(ULONG_MAX)，表示清零任务通知值
	uint32_t	ulBitsToClearOnExit	在函数xTaskNotifyWait()退出前，将任务的通知值与参数ulBitsToClearOnExit的按位取反值按位与操作。设置参数ulBitsToClearOnExit为0xFFFFFFFF(ULONG_MAX)，表示清零任务通知值
	uint32_t	pulNotificationValue	用于向外回传任务的通知值。这个通知值在参数ulBitsToClearOnExit起作用前将通知值拷贝到*pulNotificationValue中。如果不需要返回任务的通知值，这里设置成NULL
	TickType_t	xTicksToWait	因等待通知而进入阻塞状态的最大时间。时间单位为系统节拍周期。宏pdMS_TO_TICKS用于将指定的毫秒时间转化为相应的系统节拍数
返回值	如果接收到通知，返回pdTRUE，如果API函数xTaskNotifyWait()等待超时，返回pdFALSE。		

2.1.39 xTaskPriorityDisinherit

Function	BaseType_t xTaskPriorityDisinherit(TaskHandle_t const pxMutexHolder)		
描述	该函数为恢复优先级		
参数	Type	参数	描述
	TaskHandle_t	pxMutexHolder	任务句柄
返回值	pdTRUE -需要任务切换。 pdFALSE -不需切换		

2.1.40 xTaskRemoveFromEventList

Function	BaseType_t xTaskRemoveFromEventList(const List_t * const pxEventList)		
描述	该函数为将任务从队列中移除		
参数	Type	参数	描述
	List_t	pxEventList	队列列表
返回值	pdTRUE -移除成功 pdFALSE -移除失败		

2.1.41 xTaskRemoveFromUnorderedEventList

Function	BaseType_t xTaskRemoveFromUnorderedEventList(ListItem_t *pxEventListItem, const TickType_t xItemValue)		
描述	该函数为从无序的列表中删除节点		
参数	Type	参数	描述
	ListItem_t	pxEventListItem	事件列表
	TickType_t	xItemValue	节点值
返回值	pdTRUE -移除成功 pdFALSE -移除失败		

2.1.42 xTaskResumeAll

Function	BaseType_t xTaskResumeAll(void)		
描述	该函数为恢复所有挂起的任务		
参数	Type	参数	描述
	void	void	无
返回值	无		

2.2 Queue相关函数

2.2.1 ucQueueGetQueueType

Function	uint8_t ucQueueGetQueueType(QueueHandl_t xQueue)		
描述	该函数为获取队列类型		
参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄
返回值	返回队列类型		

2.2.2 uxQueueGetQueueNumber

Function	UBaseType_t uxQueueGetQueueNumber(QueueHandle_t xQueue)		
描述	该函数为获取队列号		
参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄
返回值	返回队列号		

2.2.3 uxQueueMessagesWaiting

Function	UBaseType_t uxQueueMessagesWaiting(const QueueHandle_t xQueue)
描述	该函数为获取队列入队信息

参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄
返回值	返回队列中存储的信息数目		

2.2.4 uxQueueSpacesAvailable

Function	UBaseType_t uxQueueSpacesAvailable(const QueueHandle_t xQueue)		
描述	该函数为获取队列的空闲数目		
参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄
返回值	返回队列的空闲数目		

2.2.5 vQueueAddToRegistry

Function	void vQueueAddToRegistry(QueueHandle_t xQueue, const char *pcQueueName)		
描述	该函数为队列注册		
参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄
	char	pcQueueName	分配队列的名字
返回值	无		

2.2.6 vQueueDelete

Function	void vQueueDelete(QueueHandle_t xQueue)		
描述	该函数为删除队列并释放分配给队列的内存		
参数	Type	参数	描述
	QueueHandl_t	xQueue	队列句柄

返回值	无

2.2.7 vQueueSetQueueNumber

Function	void vQueueSetQueueNumber(QueueHandle_t xQueue, UBaseType_t uxQueueNumber)		
描述	该函数为设置队列号		
参数	**Type**	**参数**	**描述**
	QueueHandl_t	xQueue	队列句柄
	UBaseType_t	uxQueueNumber	要设置的队列号
返回值	无		

2.2.8 vQueueUnregisterQueue

Function	void vQueueUnregisterQueue(QueueHandle_t xQueue)		
描述	该函数为解除注册，从队列注册表中移除指定的队列		
参数	**Type**	**参数**	**描述**
	QueueHandl_t	xQueue	队列句柄
返回值	无		

2.2.9 vQueueWaitForMessageRestricted

Function	void vQueueWaitForMessageRestricted(QueueHandle_t xQueue, TickType_t xTicksToWait)		
描述	该函数为把任务加入到阻塞链表中		
参数	**Type**	**参数**	**描述**
	QueueHandl_t	xQueue	队列句柄
	TickType_t	xTicksToWait	阻塞时间

返回值	无

2.2.10 xQueueAddToSet

Function	BaseType_t xQueueAddToSet(QueueSetMemberHandle_t xQueueOrSemaphore, QueueSetHandle_t xQueueSet)		
描述	该函数为将队列添加到队列集中		
参数	**Type**	**参数**	**描述**
	QueueSetMemberHandle_t	xQueueOrSemaphore	处理的队列或信号量被添加到队列
	QueueSetHandle_t	xQueueSet	所创建的队列集句柄
返回值	pdTRUE -添加成功 pdFALSE -添加失败		

2.2.11 xQueueCreateCountingSemaphore

Function	QueueHandle_t xQueueCreateCountingSemaphore(const UBaseType_t uxMaxCount, const UBaseType_t uxInitialCount)		
描述	该函数为创建计数信号量		
参数	**Type**	**参数**	**描述**
	UBaseType_t	uxMaxCount	最大计数值，当信号到达这个值后，就不再增长了
	UBaseType_t	uxInitialCount	创建信号量时的初始值
返回值	返回队列句柄		

2.2.12 xQueueCreateMutex

Function	QueueHandle_t xQueueCreateMutex(const uint8_t ucQueueType)		
描述	该函数为创建互斥量		
参数	Type	参数	描述
	uint8_t	ucQueueType	队列类型
返回值	返回队列句柄		

2.2.13 xQueueCreateSet

Function	QueueSetHandle_t xQueueCreateSet(const UBaseType_t uxEventQueueLength)		
描述	该函数为创建队列集，队列集也是一个队列，只是其存储区放置的是队列的指针，队列集的作用是让一个任务可以同时从几个队列中获取数据或者阻塞在其中等待数据。队列集很少使用。		
参数	Type	参数	描述
	UBaseType_t	uxEventQueueLength	所有被包含队列的数据个数总和
返回值	返回队列句柄		

2.2.14 xQueueGenericCreate

Function	QueueHandle_t xQueueGenericCreate(const UBaseType_t uxQueueLength, const UBaseType_t uxItemSize, const uint8_t ucQueueType)		
描述	该函数为队列创建函数		
参数	Type	参数	描述
	UBaseType_t	uxQueueLength	队列项数目
	UBaseType_t	uxItemSize	每个队列项的大小
	uint8_t	ucQueueType	queueQUEUE_TYPE_BASE：表示队列 queueQUEUE_TYPE_SET：表示队列集合 queueQUEUE_TYPE_MUTEX：表示互斥量

			queueQUEUE_TYPE_COUNTING_SEMAPHORE：表示计数信号量 queueQUEUE_TYPE_BINARY_SEMAPHORE：表示二进制信号量 queueQUEUE_TYPE_RECURSIVE_MUTEX：表示递归互斥量
返回值	返回队列句柄		

2.2.15 xQueueGenericReceive

Function	BaseType_t xQueueGenericReceive(QueueHandle_t xQueue, void * const pvBuffer,TickType_t xTicksToWait, const BaseType_t xJustPeeking)		
描述	该函数为通用接收信号量函数		
参数	**Type**	**参数**	**描述**
	QueueHandle_t	xQueue	队列句柄
	void	pvBuffer	指向一个缓冲区，用于拷贝接收到的列表项
	TickType_t	xTicksToWait	要接收的项目队列为空时，允许任务最大阻塞时间。如果设置该参数为0，则表示队列为空也立即返回。阻塞时间的单位是系统节拍周期，宏portTICK_RATE_MS可辅助计算真实阻塞时间。如果INCLUDE_vTaskSuspend设置成1，并且阻塞时间设置成portMAX_DELAY，将会引起任务无限阻塞（不会有超时）
	BaseType_t	xJustPeeking	一般默认是pdFALSE
返回值	成功接收到列表项返回pdTRUE，否则返回pdFALSE		

2.2.16 xQueueGenericReset

Function	portBASE_TYPE xQueueGenericReset(xQueueHandle xQueue, portBASE_TYPE xNewQueue)

描述	该函数为队列重置，如果是新队列则需要初始化队列中的链表。如果是对已有队列进行复位则需要检查是否有任务正在等待向队列中发送数据，如果有则将将任务从等待链表中移除，同时进行任务调度。		
参数	**Type**	**参数**	**描述**
	xQueueHandle	xQueue	队列句柄
	portBASE_TYPE	xNewQueue	当前队列是否是新队列
返回值	pdPASS		

2.2.17 xQueueGenericSend

Function	BaseType_t xQueueGenericSend(QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait, const BaseType_t xCopyPosition)		
描述	该函数为向队列投递队列项		
参数	**Type**	**参数**	**描述**
	xQueueHandle	xQueue	队列句柄
	void	pvItemToQueue	指针，指向要入队的项目。要保存到队列中的项目字节数在队列创建时就已确定。因此要从指针pvItemToQueue指向的区域拷贝到队列存储区域的字节数也已确定
	TickType_t	xTicksToWait	指向的区域拷贝到队列存储区域的字节数
	BaseType_t	xCopyPosition	如果队列满，任务等待队列空闲的最大时间。如果队列满并且xTicksToWait被设置成0，函数立刻返回。时间单位为系统节拍时钟周期，因此宏portTICK_PERIOD_MS可以用来辅助计算真实延时值。如果INCLUDE_vTaskSuspend设置成1，并且指定延时为portMAX_DELAY将引起任务无限阻塞（没有超时）
返回值	队列项入队成功返回pdTRUE，否则返回errQUEUE_FULL		

2.2.18 xQueueGiveMutexRecursive

Function	BaseType_t xQueueGiveMutexRecursive(QueueHandle_t xMutex)		
描述	该函数为释放递归信号量函数		
参数	Type	参数	描述
	QueueHandle_t	xMutex	信号量句柄
返回值	成功返回pdPASS 失败返回pdFAIL		

2.2.19 xQueueRemoveFromSet

Function	BaseType_t xQueueRemoveFromSet(QueueSetMemberHandle_t xQueueOrSemaphore, QueueSetHandle_t xQueueSet)		
描述	该函数为从队列中删除一个RTOS队列或信号量集		
参数	Type	参数	描述
	QueueSetMemberHandle_t	xQueueOrSemaphore	队列的处理或信号从队列中删除
	QueueSetHandle_t	xQueueSet	队列的处理组的队列或信号量
返回值	成功返回pdPASS 失败返回pdFAIL		

2.2.20 xQueueSelectFromSet

Function	BaseType_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait)		
描述	该函数为查看和获取队列集的数据存储区中的数据		
参数	Type	参数	描述
	QueueSetHandle_t	xQueueSet	队列集句柄

	TickType_t	xTicksToWait	阻塞超时参数
返回值	成功返回pdTRUE 失败返回pdFALSE		

2.2.21 xQueueTakeMutexRecursive

Function	BaseType_t xQueueTakeMutexRecursive(QueueHandle_t xMutex, TickType_t xTicksToWait)		
描述	该函数为获取递归信号量函数		
参数	**Type**	**参数**	**描述**
	QueueHandle_t	xMutex	信号量句柄
	TickType_t	xTicksToWait	阻塞超时参数
返回值	成功返回pdPASS 失败返回pdFAIL		

2.3 Event Groups相关函数

2.3.1 uxEventGroupGetNumber

Function	UBaseType_t uxEventGroupGetNumber(void *xEventGroup)		
描述	该函数为获取事件组号		
参数	**Type**	**参数**	**描述**
	void	xEventGroup	事件组句柄
返回值	返回事件组号		

2.3.2 vEventGroupClearBitsCallback

Function	void vEventGroupClearBitsCallback(void *pvEventGroup, const uint32_t ulBitsToClear)
描述	该函数为清零特定的bit的callback函数

参数	Type	参数	描述
	void	pvEventGroup	事件组句柄
	uint32_t	ulBitsToClear	需要清除的bit
返回值	无		

2.3.3 vEventGroupDelete

Function	void vEventGroupDelete(EventGroupHandle_t xEventGroup)		
描述	该函数为删除以前使用xEventGroupCreate()创建的事件组		
参数	Type	参数	描述
	EventGroupHandle_t	xEventGroup	事件组句柄
返回值	无		

2.3.4 vEventGroupSetBitsCallback

Function	void vEventGroupSetBitsCallback(void *pvEventGroup, const uint32_t ulBitsToSet)		
描述	该函数为设置特定的bit的callback函数		
参数	Type	参数	描述
	void	pvEventGroup	事件组句柄
	uint32_t	ulBitsToSet	需要设置的bit
返回值	无		

2.3.5 xEventGroupClearBits

Function	EventBits_t xEventGroupClearBits(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToClear)
描述	该函数为清除特定的bit

参数	Type	参数	描述
	EventGroupHandle_t	xEventGroup	事件组句柄
	EventBits_t	uxBitsToClear	需要清零的bit，可以为多个bit
返回值	Handler内部存储的bits值，也可以用来判断成功被清零的bit		

2.3.6 xEventGroupCreate

Function	EventGroupHandle_t xEventGroupCreate(void)		
描述	该函数为创建事件组		
参数	Type	参数	描述
	void	void	无
返回值	如果成功建立事件组，则会返回事件组的句柄（指针），如果内存堆没有足够的内存则会返回NULL（创建失败）		

2.3.7 xEventGroupSetBits

Function	EventBits_t xEventGroupSetBits(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToSet)		
描述	该函数为设定特定的bit		
参数	Type	参数	描述
	EventGroupHandle_t	xEventGroup	事件组句柄
	EventBits_t	uxBitsToSet	需要设定的bit，可以为多个bit
返回值	仍然保持为1的bit		

2.3.8 xEventGroupSync

Function	EventBits_t xEventGroupSync(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToSet, const EventBits_t uxBitsToWaitFor, TickType_t xTicksToWait)		
描述	该函数为多个task间进行同步		
参数	**Type**	**参数**	**描述**
	EventGroupHandle_t	xEventGroup	事件组句柄
	EventBits_t	uxBitsToSet	本task进行设置的bit
	EventBits_t	uxBitsToWaitFor	等待的bit组合
	TickType_t	xTicksToWait	超时时间
返回值	当函数退出时，等待的bit组合的真实值		

2.3.9 xEventGroupWaitBits

Function	EventBits_t xEventGroupWaitBits(EventGroupHandle_t xEventGroup, const EventBits_t uxBitsToWaitFor, const BaseType_t xClearOnExit, const BaseType_t xWaitForAllBits, Ticktype_t xTicksToWait)		
描述	该函数为等待特定的bit		
参数	**Type**	**参数**	**描述**
	EventGroupHandle_t	xEventGroup	事件组句柄
	EventBits_t	uxBitsToWaitFor	等待的bit，可以有多个bit
	BaseType_t	xClearOnExit	退出时是否清零等待的比特位
	BaseType_t	xWaitForAllBits	等待的bit，全部为1时才会退出
	Ticktype_t	xTicksToWait	等待超时时间
返回值	当函数退出时，等待的bit真实值		

2.4 Timers相关函数

2.4.1 *pcTimerGetTimerName

Function	const char *pcTimerGetTimerName(TimerHandle_t xTimer)		
描述	该函数为获取软件定时器名		
参数	Type	参数	描述
	TimerHandle_t	xTimer	定时器句柄
返回值	返回软件定时器名		

2.4.2 *pvTimerGetTimerID

Function	void *pvTimerGetTimerID(const TimerHandle_t xTimer)		
描述	该函数为用于返回使用函数xTimerCreate创建的软件定时器ID		
参数	Type	参数	描述
	TimerHandle_t	xTimer	定时器句柄
返回值	定时器ID（类型转换至uint32_t后可判断哪个定时器触发）		

2.4.3 xTimerCreate

Function	TimerHandle_t xTimerCreate(const char * const pcTimerName, const TickType_t xTimerPeriodInTicks, const UBaseType_t uxAutoReload, void * const pvTimerID, TimerCallbackFunction_t pxCallbackFunction)		
描述	该函数为创建软件定时器		
参数	Type	参数	描述
	char	pcTimerName	定时器名字，用于调试目的，方便识别不同的定时器
	TickType_t	xTimerPeriodInTicks	定时器周期，单位系统时钟节拍
	UBaseType_t	uxAutoReload	选择周期模式还是单次模式，若参数为pdTRUE，则表示选择周期模式，若参数为pdFALSE，则表示选择单次模式
	void *	pvTimerID	定时器 ID，为uint32_t类型，需转换为void *传入。

	TimerCallbackFunction_t	pxCallbackFunction	定时器回调函数
返回值	创建成功返回定时器的句柄，由于 FreeRTOSCongfig.h 文件中定义的 heap 空间不足，或者定时器周期设置为 0，会返回 NULL		

2.4.4 xTimerGenericCommand

Function	BaseType_t xTimerGenericCommand(TimerHandle_t xTimer, const BaseType_t xCommandID, const TickType_t xOptionalValue, BaseType_t * const pxHigherPriorityTaskWoken, const TickType_t xTicksToWait)		
描述	该函数为内部工具函数，用来统一管理对软件定时器的命令。 这里采用的是命令队列的操作，命令会被封装后放入命令队列。但是这里并不会对命令进行处理，命令的处理是守护task的工作。 这样做的好处是简化处理:软件定时器是由守护task进行模拟的，假如有一个软件定时器要被修改状态(启动，停止)，函数调用后不会立即生效，只有守护task被调度后才会生效		
参数	**Type**	**参数**	**描述**
	TimerHandle_t	xTimer	定时器句柄
	BaseType_t	xCommandID	命令ID
	TickType_t	xOptionalValue	可选值
	BaseType_t	pxHigherPriorityTaskWoken	NULL
	TickType_t	xTicksToWait	指定时间
返回值	pdFAIL -失败 pdPASS -成功		

2.4.5 xTimerIsTimerActive

Function	void *pvTimerGetTimerID(const TimerHandle_t xTimer)		
描述	该函数为用于判定定时器是否处于激活状态		
参数	**Type**	**参数**	**描述**

	TimerHandle_t	xTimer	定时器句柄
返回值	pdFALSE -没有激活 pdTRUE -激活		

2.4.6 xTimerPendFunctionCall

Function	BaseType_t xTimerPendFunctionCall(PendedFunction_t xFunctionToPend, void *pvParameter1, uint32_t ulParameter2, TickType_t xTicksToWait)		
描述	该函数为用于ROTS守护进程任务执行函数		
参数	**Type**	**参数**	**描述**
	PendedFunction_t	xFunctionToPend	定时器服务/守护进程的函数来执行任务
	void	*pvParameter1	回调函数的第一个参数的值。参数有一个void *类型允许它被用来传递任何类型。例如,整数类型可以铸造一个void *,或void *可以用来指向一个结构。
	uint32_t	ulParameter2,	回调函数的第二个参数值
	TickType_t	xTicksToWait	调用这个函数将导致一个消息被发送到定时器队列守护进程的任务。xTicksToWait调用任务的时间应该一直处于封锁状态(不使用任何处理时间)空间可用定时器队列上如果队列被发现是满的。（队列的长度由FreeRTOSConfig.h 中 的configTIMER_QUEUE_LENGTH值）
返回值	如果消息成功发送到RTOS计时器守护进程返回pdPASS，否则返回pdFALSE。		

2.5 List相关函数

2.5.1 uxListRemove

Function	UBaseType_t uxListRemove(ListItem_t * const pxItemToRemove)		
描述	该函数为从列表中删除任务		
参数	Type	参数	描述
	ListItem_t	pxItemToRemove	删除节点
返回值	无		

2.5.2 vListInitialise

Function	void vListInitialise(List_t * const pxList)		
描述	该函数为链表初始化		
参数	Type	参数	描述
	List_t	pxList	链表句柄
返回值	无		

2.5.3 vListInitialiseItem

Function	void vListInitialiseItem(ListItem_t * const pxItem)		
描述	该函数为在准备插入新节点前，对新节点进行初始化，主要是检测新节点是否可用		
参数	Type	参数	描述
	ListItem_t	pxItem	链表新节点
返回值	无		

2.5.4 vListInsert

Function	void vListInsert(List_t * const pxList, ListItem_t * const pxNewListItem)
描述	该函数为将节点按照升序插入到链表的合适的位置，前提是链表原本就是按照升序排序的。算法很简单，就是从第一个元素（xListEnd.pxNext）开始往后遍历，直到找

	到合适的位置，然后将节点插进去。		
参数	**Type**	**参数**	**描述**
	List_t	pxList	链表句柄
	ListItem_t	pxNewListItem	新节点
返回值	无		

2.5.5 vListInsertEnd

Function	void vListInsertEnd(List_t * const pxList, ListItem_t * const pxNewListItem)		
描述	该函数为把一个节点插入到链表尾部		
参数	**Type**	**参数**	**描述**
	List_t	pxList	链表句柄
	ListItem_t	pxNewListItem	新节点
返回值	无		

3 用户数据业务API介绍

本章包含数据业务域面向客户端的接口和常见类型，客户可参考该章节提供的API接口来建立自己的socket通信等业务。

3.1 Socket相关函数

3.1.1 lwip_accept

Function	int lwip_accept(int s, sockaddr *addr, socklen_t *addrlen)		
描述	该函数为TCP服务器监听到客户端请求之后，调用accpet()函数去接收请求。		
参数	**Type**	**参数**	**描述**
	int	s	服务器的socket描述字
	sockaddr	addr	客户端的socket地址
	socklen_t	addrlen	socket的地址长度
返回值	0 --成功 -1 --失败		

3.1.2 lwip_bind

Function	int lwip_bind(int s, const strcut sockaddr *name, socklen_t namelen)		
描述	该函数为把一个地址组中的特定地址绑定给socket，一般在用作服务器时使用该函数。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	sockaddr	name	要绑定给socketfd的协议地址
	socklen_t	namelen	地址的长度
返回值	0 --成功 -1 --失败		

3.1.3 lwip_close

Function	int lwip_close(int s)		
描述	该函数用来关闭socket，使相应socket描述字的引用计数-1，当引用计数为0的时候，触发TCP客户端向服务器发送终止连接请求。		
参数	Type	参数	描述
	int	s	socket描述字
返回值	0 --成功 -1 --失败		

3.1.4 lwip_connect

Function	int lwip_connect(int s, const struct sockaddr *name, socklen_t namelen)		
描述	该函数为连接某个socket		
参数	Type	参数	描述
	int	s	客服端的socket描述字
	sockaddr	name	服务器的socket地址
	socklen_t	namelen	socket地址长度
返回值	0 --成功 -1 --失败		

3.1.5 lwip_fcntl

Function	int lwip_fcntl(int s, int cmd, int val)		
描述	该函数为根据描述字来操作文件的特性，目前只支持F_GETFL和F_SETFL执行，只有O_NONBLOCK标志可执行		
参数	Type	参数	描述
	int	s	socket描述字
	int	cmd	操作命令

	int	val	供命令使用的参数
返回值	返回值与命令有关，如果出错，所有命令都返回-1，如果成功返回某个其他值。		

3.1.6 lwip_getpeername

Function	int lwip_getpeername(int s, struct sockaddr *name, socklen_t *namelen)		
描述	该函数为获取远端地址信息		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	sockaddr	name	地址信息
	socklen_t	namelen	地址长度
返回值	0 ----成功 -1 ---失败		

3.1.7 lwip_getsockname

Function	int lwip_getsockname(int s, struct sockaddr *name, socklen_t *namelen)		
描述	该函数为从已连接的socket中获取本地地址信息。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	sockaddr	name	地址信息
	socklen_t	namelen	地址长度
返回值	0 ---成功 -1 ---失败		

3.1.8 lwip_getsockopt

Function	int lwip_getsockopt(int s, int level, int optname, void *optval, socklen_t *optlen)		
描述	该函数为获取socket的选项的值。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	int	level	协议栈选项，包括SOL_SOCKET，IPPROTO_TCP和IPPRPTP_IP
	int	optname	需要修改的选项名
	void	optval	修改值地址
	socklen_t	optlen	修改值长度
返回值	返回0表示成功。		

3.1.9 lwip_ioctl

Function	int lwip_ioctl(int s, long cmd, void *argp)		
描述	该函数为控制I/O设备，提供了一种获得设备信息和向设备发送控制参数的手段。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	long	cmd	函数定义的所有操作
	void	argp	指针的类型依赖与cmd参数
返回值	-1 ---错误		

3.1.10 lwip_listen

Function	int lwip_listen(int s, int backlog)		
描述	该函数为监听socket。		
参数	**Type**	**参数**	**描述**

	int	s	要监听的socket描述字
	int	backlog	相应的socket可以排队的最大链接个数
返回值	0 ---成功 -1 ---失败		

3.1.11 lwip_read

Function	int lwip_read(int s, void *mem, size_t len)		
描述	该函数为读取socket内容。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	void	mem	缓冲区
	size_t	len	缓冲区长度
返回值	返回接收到的数据大小，-1表示出错，0表示远端已关闭连接。		

3.1.12 lwip_recv

Function	int lwip_recv(int s, void *mem, size_t len, int flags)		
描述	该函数为接收数据包，tcp专用。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	void	mem	接收缓存地址
	size_t	len	接收缓存最大空间
	int	flags	接收选项
返回值	返回接收到的数据，-1表示接收错误，0表示目标地址已经传输完毕并关闭连接。		

3.1.13 lwip_recvfrom

Function	int lwip_recvfrom(int s, void *mem, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen)		
描述	该函数为接收数据包，udp专用。socket类型必须是SOCK_DRAM类型，由于没有连接，所以recvfrom函数增加两个与连接有关的参数。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	void	mem	接收缓存地址
	size_t	len	接收缓存最大空间
	int	flags	接收选项
	sockaddr	from	存放发送方IP地址和端口
	socklen_t	fromlen	结构体长度
返回值	返回接收到的数据数，-1表示接收错误，0表示目标地址已经传输完毕并关闭连接。		

3.1.14 lwip_select

Function	int lwip_select(int maxfdp1, fd_set *readset, fd_set *writeset, struct timeval *timeout)		
描述	该函数为挂起当前线程，等待特定的事件发生或定时器过期，本函数可以指定4类特定事件：read，write，exception和超时。		
参数	**Type**	**参数**	**描述**
	int	maxfdp1	指集合中所有文件描述符的范围，即所有文件描述符的最大值加1，不能错
	fd_set	readset	指向fd_set结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的读变化的，即我们关心是否可以从这些文件中读取数据了，如果这个集合中有一个文件可读，select就会返回一个大于0的值，表示有文件可读；如果没有可读的文件，则根据timeout参数再判断是否超时，若超出timeout的时间，select返回0，若发生错误返回负值。可以传入NULL值，表示不

			关心任何文件的读变化
	fd_set	writeset	指向fd_set结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的写变化的，即我们关心是否可以向这些文件中写入数据了，如果这个集合中有一个文件可写，select就会返回一个大于0的值，表示有文件可写，如果没有可写的文件，则根据timeout参数再判断是否超时，若超出timeout的时间，select返回0，若发生错误返回负值。可以传入NULL值，表示不关心任何文件的写变化
	timeval	timeout	设置超时时间
返回值	返回socket ID表示事件有相应，0表示超时，-1表示出错。		

3.1.15 lwip_send

Function	int lwip_send(int s, const void *data, size_t size, int flags)		
描述	该函数为socket发送数据包，tcp专用。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	void	data	要发送的数据指针
	size_t	size	要发送的数据长度
	int	flags	发送选项
返回值	返回实际发送的字节长度，-1表示发送不成功。		

3.1.16 lwip_sendto

Function	int lwip_sendto(int s, const void *data, size_t size, const struct sockaddr *to, socklen_t tolen)
描述	该函数为socket发送数据包，udp专用。

参数	Type	参数	描述
	int	s	socket描述字
	const void	data	要发送的数据指针
	size_t	size	要发送的数据长度
	sockaddr	to	目标地址的结构体
	socklen_t	tolen	结构体长度
返回值	返回实际发送的字节长度，-1表示发送不成功。		

3.1.17 lwip_setsockopt

Function	int lwip_setsockopt(int s, int level, int optname, const void *optval, socklen_t optlen)		
描述	该函数为用来改变socket的模式		
参数	Type	参数	描述
	int	s	socket描述字
	int	level	协议栈选项，包括SOL_SOCKET，IPPROTO_TCP和IPPRPTP_IP
	int	optname	需要修改的选项名
	void	optval	修改值地址
	socklen_t	optlen	修改值长度
返回值	标志打开或关闭某个特征的二进制选项。		

3.1.18 lwip_shutdown

Function	int lwip_shutdown(int s, int how)		
描述	该函数为提供更大的权限控制socket的关闭过程。		
参数	Type	参数	描述
	int	s	socket描述字

	int	how	How仅能为0，1和2这三个值
返回值	0表示停止接收当前数据并拒绝以后的数据接收。 1表示停止发送数据并丢弃未发送的数据。 2是0和1的合集。		

3.1.19 lwip_socket

Function	int lwip_socket(int domain, int type, int protocol)		
描述	该函数为服务器根据地址类型(ipv4,ipv6)，socket类型，协议创建socket。		
参数	**Type**	**参数**	**描述**
	int	domain	协议族，常用的有AF_INET、AF_INET6、AF_LOCAL、AF_ROUTE其中AF_INET代表使用ipv4地址
	int	type	socket类型，常用的socket类型有，SOCK_STREAM、SOCK_DGRAM、SOCK_RAW、SOCK_PACKET、SOCK_SEQPACKET等
	int	protocol	协议。常用的协议有，IPPROTO_TCP、IPPTOTO_UDP、IPPROTO_SCTP、IPPROTO_TIPC等
返回值	返回大于等于0的整数作为socket ID，如果出错返回-1。		

3.1.20 lwip_write

Function	int lwip_write(int s, const void *data, size_t size)		
描述	该函数为向socket写入内容，其实就是发送内容。		
参数	**Type**	**参数**	**描述**
	int	s	socket描述字
	void	data	缓冲区
	size_t	size	缓冲区长度
返回值	返回实际发送的数量，-1表示出错。		

3.2 Netdb相关函数

3.2.1 lwip_freeaddrinfo

Function	void lwip_freeaddrinfo(struct addrinfo *ai)		
描述	该函数为释放地址信息。		
参数	**Type**	**参数**	**描述**
	addrinfo	ai	指向由getaddrinfo返回的第一个addrinfo结构
返回值	无		

3.2.2 lwip_getaddrinfo

Function	void lwip_getaddrinfo(const char *nodename, const char *servname, const struct addrinfo *hints, struct addrinfo **res)		
描述	该函数为获取地址信息。		
参数	**Type**	**参数**	**描述**
	char	nodename	一个主机名或者地址串
	char	servname	服务器名，可以是十进制的端口号，也可以是第一的服务器名称
	addrinfo	hints	可以一个空指针也可以是一个指向某个addrinfo结构体的指针，调用者在结构体中填入关于期望返回的信息类型的暗示。例如：如果指定的服务器既支持TCP也支持UDP，那么调用者可以把hits结构中的ai_socketype成员设置成为SOCK_DGRAM使的返回的仅仅适用于数据报套接字的信息
	addrinfo	res	通过res指针返回一个指向addrinfo的结构体链表指针
返回值	无		

3.2.3 hostent *lwip_gethostbyname

Function	struct hostent *lwip_gethostbyname(const char *name)		
描述	该函数为用域名或主机名获取地址。		
参数	Type	参数	描述
	char	name	域名与主机名
返回值	返回hostent 结构指针，失败返回NULL。		

3.2.4 lwip_gethostbyname_r

Function	int lwip_gethostbyname_r(const char *name, struct hostent *ret, char *buf, size_t buflen, struct hostent **result, int *h_errnop)		
描述	该函数为可重入的通过主机名获取地址。		
参数	Type	参数	描述
	char	name	域名与主机名
	hostent	ret	预分配的结构体，存储结果
	char	buf	预分配的缓冲区存储更多数据
	size_t	buflen	缓冲区大小
	hostent	result	指向hostent指针，将result设置1为成功，0为失败
	int	h_errnop	指针指向一个int型存储error信息
返回值	返回0成功，非0失败		

3.3 Ip4 addr相关函数

3.3.1 ip4_addr_isbroadcast_u32

Function	u8_t ip4_addr_isbroadcast_u32(u32_t addr, const struct netif *netif)		
描述	该函数为确定一个地址是否为网络接口上的广播地址。		
参数	Type	参数	描述
	u32_t	addr	被检查的地址
	netif	netif	检查网络接口的地址
返回值	如果是广播地址返回非零。		

3.3.2 ip4_addr_netmask_valid

Function	u8_t ip4_addr_netmask_valid(u32_t netmask)		
描述	该函数为检查一个子网掩码是否有效。		
参数	Type	参数	描述
	u32_t	netmask	检查子网掩码IPv4的网络掩码
返回值	如果网络掩码有效为1，无效为0。		

3.3.3 ip4_addr_aton

Function	int ip4_addr_aton(const char *cp, ip4_addr_t *addr)		
描述	该函数为将一个字符串IP地址转换为一个32位的网络序列IP地址。		
参数	Type	参数	描述
	char	cp	输入参数cp包含ASCII表示的IP地址
	ip_addr_t	addr	输出参数addr是将要用新的IP地址更新的结构
返回值	如果成功返回非零，如果输入地址不正确则返回零。		

3.3.4 *ip4_addr_ntoa

Function	char *ip4_addr_ntoa(const ip4_addr_t *addr)		
描述	该函数为将网络地址转换成为”.”点隔的字符串。		
参数	Type	参数	描述
	ip4_addr_t	addr	网络地址指针
返回值	成功返回一个字符指针，否则的话，返回NULL。		

3.3.5 *ip4_addr_ntoa_r

Function	char *ip4_addr_ntoa_r(const ip4_addr_t *addr, char *buf, int nuflen)		
描述	该函数ip4_addr_ntoa一样,但可重入。		
参数	Type	参数	描述
	ip4_addr_t	addr	网络地址指针
	char	buf	目标缓冲区
	int	nuflen	缓冲区长度
返回值	成功返回一个字符指针，否则的话，返回NULL。		

3.3.6 ipaddr_addr

Function	u32_t ipaddr_addr(const char *cp)		
描述	该函数为将一个点分十进制的IP转换成一个长整型数。		
参数	Type	参数	描述
	char	cp	字符串，一个点分十进制的IP地址
返回值	如果正确执行将返回一个无符号长整型数，如果传入的字符串不是一个合法的IP地址，将返回INADDR_NONE。		

3.4 IOT相关函数

3.4.1 gsdk_ril_set_link_iot

Function	gsdk_status_t gsdk_ril_set_link_iot(char *imei_data, int lifetime)		
描述	该函数为注册iot平台。		
参数	**Type**	**参数**	**描述**
	char	*imei_data	
	int	lifetime	
返回值	成功返回0，失败返回非0。错误码见下表7-1		

3.4.2 gsdk_ril_del_link_iot

Function	gsdk_status_t gsdk_ril_del_link_iot(void)		
描述	该函数为去注册iot平台。		
参数	**Type**	**参数**	**描述**
	void	空	无
返回值	成功返回0，失败返回非0。错误码见下表7-1		

3.4.3 gsdk_ril_send_data_to_iot

Function	gsdk_status_t gsdk_ril_send_data_to_iot(char *user_data)		
描述	该函数为向iot平台发送数据。		
参数	**Type**	**参数**	**描述**
	char	*user_data	发送数据指针
返回值	成功返回0，失败返回非0。错误码见下表7-1		

3.5 WEFOTA相关函数

3.5.1 gsdk_ril_fota_settv(char *fotatv)

Function	gsdk_status_t gsdk_ril_fota_settv(char *fotatv)		
描述	该函数为设置目标版本号		
参数	Type	参数	描述
	char	*fotatv	目标版本号
返回值	成功返回0，失败返回非0。错误码见下表7-1		

3.5.2 gsdk_ril_fota_ctr(void)

Function	gsdk_status_t gsdk_ril_fota_ctr(void)		
描述	该函数为启动升级		
参数	Type	参数	描述
	void	空	无
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4 模块外围接口API介绍

本章规定了ME3616模块外围接口驱动API接口，用于管理和使用模块外围接口资源，比如uart，gpio，flash外设等。

4.1 模块UART驱动相关函数

4.1.1 gsdk_uart_open

Function	gsdk_status_t gsdk_uart_open(uart_port_t port, uart_config_t *config, gsdk_handle_t *phuart)		
描述	该函数为初始化uart硬件和基本功能。		
参数	**Type**	**参数**	**描述**
	uart_port_t	port	指定初始化uart端口
	uart_config_t	config	指定uart硬件初始化参数
	gsdk_handle_t	phuart	客户使用句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.1.2 gsdk_uart_close

Function	void gsdk_uart_close(gsdk_handle_t huart)		
描述	该函数为关闭uart端口。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	huart	客户使用句柄
返回值	无		

4.1.3 gsdk_uart_read

Function	int gsdk_uart_read(gsdk_handle_t huart, uint8_t *data, int len, int timeout_ms)		
描述	该函数为接收uart端口数据。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	huart	客户使用句柄
	uint8_t	data	接收数据指针
	int	len	接收长度
	int	timeout_ms	操作超时值
返回值	接收数据字节数		

4.1.4 gsdk_uart_write

Function	int gsdk_uart_write(gsdk_handle_t huart, const uint8_t *data, int len, int timeout_ms)		
描述	该函数为通过uart端口发送数据。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	huart	客户使用句柄
	uint8_t	data	发送数据指针
	int	len	发送数据长度
	int	timeout_ms	操作超时值
返回值	发送数据字节数		

4.1.5 gsdk_uart_ioctl

Function	int gsdk_uart_ioctl(gsdk_handle_t huart, uart_ioctl_code_t code, void *param, int param_len)		
描述	该函数为uart的ioctl功能，在调用前必须首先调用gsdk_uart_open()。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	huart	客户使用句柄
	uart_ioctl_code_t	code	Ioctl命令符
	void	param	命令参数
	int	param_len	参数长度
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.2 uart驱动相关数据结构

4.2.1 enum uart_port_t

uart端口号，可配置的端口号

数据成员

Type	参数	描述
uart_port_t	UART_0	UART端口0
uart_port_t	UART_1	UART端口1
uart_port_t	UART_2	UART端口2
uart_port_t	UART_3	UART端口3
uart_port_t	UART_MAX	UART 端口最大值

4.2.2 struct uart_config_t

uart端口参数配置项

数据成员

Type	参数	描述

uart_baudrate_t	baudrate	波特率配置
uart_word_length_t	word_length	数据位配置
uart_stop_bit_t	stop_bit	停止位配置
uart_parity_t	parity	校验位配置

4.3 flash驱动函数

4.3.1 gsdk_flash_open

Function	gsdk_status_t gsdk_flash_open(uint32_t base, uint32_t size, gsdk_handle_t *phflash)		
描述	该函数为初始化flash硬件和基本功能。		
参数	**Type**	**参数**	**描述**
	uint32_t	base	flash起始地址
	uint32_t	size	flash的大小
	gsdk_handle_t	phflash	客户使用句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.3.2 gsdk_flash_close

Function	void gsdk_flash_close(gsdk_handle_t hflash)		
描述	该函数为关闭flash硬件和基本功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hflash	客户使用句柄
返回值	无		

4.3.3 gsdk_flash_erase

Function	gsdk_status_t gsdk_flash_erase(gsdk_handle_t hflash, uint32_t start_address, flash_block_t block_type)		
描述	该函数为初始化flash擦除功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hflash	客户使用句柄
	uint32_t	start_address	擦除flash的起始地址
	Flash_block_t	block_type	擦除flash的块大小
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.3.4 gsdk_flash_read

Function	gsdk_status_t gsdk_flash_read(gsdk_handle_t hflash, uint32_t start_address, uint8_t *buffer, uint32_t length)		
描述	该函数为flash读功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hflash	客户使用句柄
	uint32_t	start_adderss	读flash的起始地址
	uint8_t	buffer	保存读取数据的位置
	uint32_t	length	读取flash的长度
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.3.5 gsdk_flash_write

Function	gsdk_status_t gsdk_flash_write(gsdk_handle_t hflash, uint32_t address, const uint8_t *data, uint32_t length)

描述	该函数为flash写功能。在地址第一次被写之前必须被擦除。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hflash	客户使用句柄
	uint32_t	address	写flash的起始地址
	const uint8_t	data	被写入的源数据
	uin32_t	length	数据长度
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.4 flash驱动相关数据结构

4.4.1 enum flash_block_t

flash块大小参数配置项

数据成员

Type	参数	描述
flash_block_t	FLASH_BLOCK_4K	块大小为4K
flash_block_t	FLASH_BLOCK_32K	块大小为32K
flash_block_t	FLASH_BLOCK_64K	块大小为64K

4.5 gpio驱动函数

4.5.1 gsdk_gpio_open

Function	gsdk_status_t gsdk_gpio_open(gpio_pin_t pin, gpio_config_t *config, gsdk_handle_t *pgpio)		
描述	该函数为初始化gpio硬件和基本功能。		
参数	**Type**	**参数**	**描述**
	gpio_pin_t	pin	配置的引脚号

	gpio_config_t	config	引脚配置参数
	gsdk_handle_t	pgpio	客户使用句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.5.2 gsdk_gpio_close

Function	void gsdk_gpio_close(gsdk_handle_t hgpio)		
描述	该函数为关闭gpio硬件和基本功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hgpio	客户使用句柄
返回值	无		

4.5.3 gsdk_gpio_read

Function	gsdk_status_t gsdk_gpio_read(gsdk_handle_t hgpio, gpio_data_t *pdata)		
描述	该函数为gpio读功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hgpio	客户使用句柄
	gpio_data_t	pdata	读取引脚数据指针
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.5.4 gsdk_gpio_write

Function	gsdk_status_t gsdk_gpio_write(gsdk_handle_t hgpio, gpio_data_t data)
描述	该函数为gpio写功能。

参数	Type	参数	描述
	gsdk_handle_t	hgpio	客户使用句柄
	gpio_data_t	data	写引脚数据指针
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.5.5 gsdk_gpio_toggle

Function	gsdk_status_t gsdk_gpio_toggle(gsdk_handle_t hgpio)		
描述	该函数为gpio切换功能。		
参数	Type	参数	描述
	gsdk_handle_t	hgpio	客户使用句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.5.6 gsdk_gpio_interrupt_enable

Function	gsdk_status_t gsdk_gpio_interrupt_enable(gsdk_handle_t hgpio, int enable)		
描述	该函数为gpio中断使能函数。		
参数	Type	参数	描述
	gsdk_handle_t	hgpio	客户使用句柄
	int	enable	使能/禁止
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.6 gpio驱动相关数据结构

4.6.1 enum gpio_pin_t

gpio 引脚选择 参数项。

数据成员

Type	参数	描述
gpio_pin_t	GPIO_PIN_0	0号引脚
gpio_pin_t	GPIO_PIN_1	1号引脚
...	...	...
gpio_pin_t	GPIO_PIN_36	36号引脚
gpio_pin_t	GPIO_PIN_MAX	可配引脚最大值

4.6.2 struct gpio_config_t

gpio 引脚配置 参数项。

数据成员

Type	参数	描述
gpio_direction_t	direction	输入/输出方向
gpio_pull_t	pull	gpio数据类型
gpio_int_t	int_mode	中断模式
uint32_t	debounce_time	硬件防反跳的时间,以毫秒为单位
gpio_callback_t	callback	gpio callback函数 typedef void (*gpio_callback_t)(void * user_data)

4.6.3 enum gpio_data_t

gpio 数据类型参数项。

数据成员

Type	参数	描述
gpio_data_t	GPIO_DATA_LOW	gpio数据： 低（0）
gpio_data_t	GPIO_DATA_HIGH	gpio数据：高（1）
gpio_data_t	GPIO_DATA_MAX	Gpio数据类型最大值

4.7 pwr驱动函数

4.7.1 gsdk_sleeplock_open

Function	gsdk_status_t gsdk_sleeplock_open(const char *handle_name, gsdk_hadle_t *psl)		
描述	该函数为设置一个sleep handle来控制系统睡眠状态。		
参数	**Type**	**参数**	**描述**
	char	handle_name	sleep handle的字符类型名称
	gsdk_hadle_t	psl	sleep句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.7.2 gsdk_sleeplock_acquire

Function	gsdk_status_t gsdk_sleeplock_acquire(gsdk_hadle_t *psl)		
描述	该函数为防止系统进入睡眠或深度睡眠模式。		
参数	**Type**	**参数**	**描述**
	gsdk_hadle_t	psl	sleep句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.7.3 gsdk_sleeplock_release

Function	gsdk_status_t gsdk_sleeplock_release(gsdk_hadle_t *psl)		
描述	该函数为打开特定的睡眠锁和允许单片机进入睡眠模式时解锁睡眠锁。		
参数	**Type**	**参数**	**描述**
	gsdk_hadle_t	psl	sleep句柄

返回值	成功返回0，失败返回错误码，错误码见下表7-1。

4.7.4 gsdk_sleeplock_close

Function	gsdk_status_t gsdk_sleeplock_close(gsdk_hadle_t *psl)		
描述	该函数为释放sleep handle		
参数	**Type**	**参数**	**描述**
	gsdk_hadle_t	psl	sleep句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1。		

4.7.5 gsdk_sleep_islocked

Function	gsdk_lc_t gsdk_sleep_islocked(void)		
描述	该函数为检查在系统睡眠时是否有特定的锁		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	返回true表示有特定级别的锁在hold。		

4.7.6 gsdk_wakup_in_open

Function	gsdk_lc_t gsdk_wakup_in_open(gsdk_handle_t *pwk, wakeup_in_eint_callback_t user_callback, void *user_data)		
描述	该函数为打开wakeup_in唤醒功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	*pwk	操作句柄
	wakeup_in_eint_callback	user_callback	用户回调函数
	void	*user_data	回调函数参数

返回值	成功返回0，失败返回错误码，错误码见下表7-1。

4.7.7 gsdk_wakup_in_close

Function	void gsdk_wakeup_in_close(gsdk_handle_t *pwk)		
描述	该函数为关闭wakeup_in唤醒功能		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	*pwk	操作句柄
返回值	无返回值。		

4.8 adc驱动函数

4.8.1 gsdk_adc_read

Function	int gsdk_adc_read(gsdk_handle_t padc, adc_channel_t adc_channel, uint32_t *adc_volt)		
描述	该函数为打开一个rtc handle。		
参数	**Type**	**参数**	**描述**
	gsdk_hadle_t	padc	adc句柄
	adc_channel_t	channel	adc通道号
	uint32_t	adc_volt	返回的电压值会保存在这个参数中
返回值	返回电压值		

4.8.2 gsdk_adc_open

Function	int gsdk_adc_open(gsdk_handle_t *padc)		
描述	该函数为打开adc功能。		
参数	Type	参数	描述
	gsdk_hadle_t	padc	adc句柄
返回值	成功返回0，失败返回-1		

4.8.3 gsdk_adc_close

Function	int gsdk_adc_close(gsdk_handle_t padc)		
描述	该函数为关闭adc功能。		
参数	Type	参数	描述
	gsdk_hadle_t	padc	adc句柄
返回值	成功返回0，失败返回-1		

4.9 IIC驱动函数

4.9.1 gsdk_i2c_master_init

Function	int gsdk_i2c_master_init(gsdk_port_t i2c_port, i2c_frequency_t i2c_freq)		
描述	该函数初始化i2c master功能。		
参数	Type	参数	描述
	gsdk_port_t	i2c_port	i2c端口
	i2c_frequency_t	i2c_freq	i2c传输速率
返回值	成功返回0，失败返回非0。		

4.9.2 gsdk_i2c_write

Function	int gsdk_i2c_write(uint8_t slave_addr, uint8_t *data, uint16_t len)

描述	该函数i2c写操作功能。		
参数	**Type**	**参数**	**描述**
	uint8_t	slave_addr	从设备地址
	uint8_t	*data	发送数据的指针
	uint16_t	len	发送数据的长度
返回值	成功返回0，失败返回非0。		

4.9.3 gsdk_i2c_read

Function	int gsdk_i2c_read(uint8_t slave_addr, uint8_t *data, uint16_t len)		
描述	该函数i2c读操作功能。		
参数	**Type**	**参数**	**描述**
	uint8_t	slave_addr	从设备地址
	uint8_t	*data	读取数据的指针
	uint8_t	len	读取数据的长度
返回值	成功返回0，失败返回非0		

4.10 ALARM驱动函数

4.10.1 gsdk_alarm_open

Function	int gsdk_alarm_open(gsdk_handle_t *phalarm, gsdk_alarm_config_t *config)		
描述	该函数打开alarm定时器功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	*phalarm	alarm句柄
	gsdk_alarm_config_t	*config	alarm配置参数结构指针
返回值	成功返回0，失败返回非0。		

4.10.2 gsdk_alarm_close

Function	int gsdk_alarm_close(gsdk_handle_t halarm)		
描述	该函数关闭alarm定时器功能。		
参数	Type	参数	描述
	gsdk_handle_t	halarm	alarm句柄
返回值	成功返回0，失败返回非0。		

4.10.3 gsdk_alarm_start

Function	int gsdk_alarm_start(gsdk_handle_t halarm)		
描述	该函数启动alarm定时器功能。		
参数	Type	参数	描述
	gsdk_handle_t	halarm	alarm句柄
返回值	成功返回0，失败返回非0		

4.10.4 gsdk_alarm_stop

Function	int gsdk_alarm_stop(gsdk_handle_t halarm)		
描述	该函数停止alarm定时器功能。		
参数	Type	参数	描述
	gsdk_handle_t	halarm	alarm句柄
返回值	成功返回0，失败返回非0		

4.11 PWM驱动函数

4.11.1 gsdk_pwm_get_duty_cycle

Function	gsdk_status_t gsdk_pwm_get_duty_cycle(gsdk_handle_t *hpwm, uint32_t *duty_cycle)		
描述	该函数为获得占空比。		
参数	Type	参数	描述
	gsdk_handle_t	*hpwm	操作句柄
	uint32_t	*duty_cycle	获得的值被存储在该参数中
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.11.2 gsdk_pwm_get_frequency

Function	gsdk_status_t gsdk_pwm_get_frequency(gsdk_handle_t*hpwm, uint32_t *frequency)		
描述	该函数为获取频率周期。		
参数	Type	参数	描述
	gsdk_handle_t	*hpwm	操作句柄
	uint32_t	*frequency	获得的值被存储在该参数中
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.11.3 gsdk_pwm_open

Function	gsdk_status_t gsdk_pwm_open(pwm_channel_t channel, pwm_config_t *config, gsdk_handle_t *hpwm)		
描述	该函数为打开pwm功能。		
参数	Type	参数	描述
	pwm_channel_t	channel	通道号
	pwm_config_t	*config	配置参数
	gsdk_handle_t	*hpwm	操作句柄

返回值	成功返回0，失败返回非0。错误码见下表7-1

4.11.4 gsdk_pwm_start

Function	gsdk_status_t gsdk_pwm_start(gsdk_handle_t *hpwm,)		
描述	该函数为启动pwm功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hpwm	操作句柄
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.11.5 gsdk_pwm_stop

Function	gsdk_status_t gsdk_pwm_stop(gsdk_handle_t hpwm)		
描述	该函数为停止pwm功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hpwm	操作句柄
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.11.6 gsdk_pwm_close

Function	gsdk_pwm_close(gsdk_handle_t hpwm)		
描述	该函数为关闭pwm。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	hpwm	操作句柄
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.12 SPI驱动函数

4.12.1 gsdk_spi_open

Function	gsdk_status_t gsdk_spi_open(gsdk_handle_t *pspi, spi_master_config_t *spi_config)		
描述	该函数为打开spi功能。		
参数	Type	参数	描述
	gsdk_handle_t	*pspi	操作句柄
	spi_master_config_t	*spi_config	配置参数
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.12.2 gsdk_spi_transmit_and_receive

Function	gsdk_status_t gsdk_spi_transmit_and_receive(gsdk_handle_t pspi, uint8_t *tx_data, uint8_t *rx_data, int tx_size, int rx_size, int timeout)		
描述	该函数为spi发送或接收数据		
参数	Type	参数	描述
	gsdk_handle_t	pspi	操作句柄
	uint8_t	*tx_data	发送数据的指针
	uint8_t	*rx_data	接受数据的指针
	int	tx_size	发送数据长度
	int	rx_size	接受数据长度
	int	timeout	超时时间
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.12.3 gsdk_spi_cs_gpio_set

Function	gsdk_status_t gsdk_spi_cs_gpio_set(gsdk_handle_t pspi, spi_master_cs_gpio_t cs_level)		
描述	该函数为设置cs引脚等级为SPI_MASTER_CS_GPIO_MODE。		
参数	Type	参数	描述
	gsdk_handle_t	pspi	操作句柄
	spi_master_cs_gpio_t	cs_level	cs引脚的模式

返回值	成功返回0，失败返回非0。错误码见下表7-1

4.12.4 gsdk_spi_close

Function	gsdk_status_t gsdk_spi_close(gsdk_handle_t pspi)		
描述	该函数为关闭spi功能。		
参数	**Type**	**参数**	**描述**
	gsdk_handle_t	pspi	操作句柄
返回值	成功返回0，失败返回非0。错误码见下表7-1		

4.13 开机原因函数

4.13.1 gsdk_sys_get_boot_mode

Function	gsdk_boot_mode_t gsdk_sys_boot_mode(gsdk_rtc_wakeup_mode_t *rtc_mode)		
描述	该函数为获取开机原因。		
参数	**Type**	**参数**	**描述**
	gsdk_rtc_wakeup_mode_t	rtc_mode	获取开机原因句柄
返回值	返回开机原因，开机原因见下表		

第一阶段开机原因列表信息	
GSDK_BOOT_FIRST_UP	0
GSDK_BOOT_DEEP_SLEEP	1
GSDK_BOOT_DEEPER_SLEEP	2
GSDK_BOOT_SYS_RESET	3
GSDK_BOOT_WDT_HW_RESET	4
GSDK_BOOT_WDT_SW_RESET	5
GSDK_BOOT_FORCED_SHUT_DOWN	6

GSDK_BOOT_MODE_INVALID	7

第二阶段开机原因列表信息	
GSKD_BOOT_RTC_TC_WAKEUP	10
GSDK_BOOT_RTC_EINT_WAKEUP	11
GSDK_BOOT_RTC_ALARM_WAKEUP	12
GSDK_BOOT_POWERKEY_WAKEUP	13
GSDK_BOOT_INVALID_WAKEUP	0xFF

5 WelinkOpen AT TOK API介绍

本章规定了ME3616模块WelinkOpen开发中使用的AT TOK相关API接口，使用该类API接口来处理接收到的AT命令字符串。

5.1 AT TOK函数

5.1.1 at_tok_start

Function	int at_tok_start(char **p_cur)		
描述	该函数为找到第一个“：”的位置。		
参数	**Type**	**参数**	**描述**
	char	p_cur	输入字符串
返回值	如果不是一个有效的响应字符串返回-1，成功返回0。		

5.1.2 at_tok_nextint

Function	int at_tok_nextint(char **p_cur，int *p_out)		
描述	该函数为用于获取返回的AT数据对应指针位置后的首个Integer数据，并将指针指向数据之后。		
参数	**Type**	**参数**	**描述**
	char	p_cur	输入字符串
	int	p_out	输出存放在p_out
返回值	成功返回0，失败返回-1。		

5.1.3 at_tok_nexthexint

Function	int at_tok_nexthexint(char **p_cur，int *p_out)		
描述	该函数为获取hex数据，同样会将指针后移。		
参数	Type	参数	描述
	char	p_cur	输入字符串
	int	p_out	输出存放在p_out
返回值	成功返回0，失败返回-1。		

5.1.4 at_tok_nextbool

Function	int at_tok_nextbool(char **p_cur，int *p_out)		
描述	该函数为获取的是boolean数据，同样会将指针后移。		
参数	Type	参数	描述
	char	p_cur	输入字符串
	int	p_out	输出存放在p_out
返回值	成功返回0，失败返回-1。		

5.1.5 at_tok_nextstr

Function	int at_tok_nextstr(char **p_cur，int *p_out)		
描述	该函数为获取的是String数据，同样会将指针后移。		
参数	Type	参数	描述
	char	p_cur	输入字符串
	int	p_out	输出存放在p_out
返回值	成功返回0，失败返回-1。		

5.1.6 at_tok_hasmore

Function	int at_tok_hasmore(char **p_cur)		
描述	该函数为判断相应参数结束没有，即是不是到达数据尾部，不会进行指针操作。		
参数	Type	参数	描述
	char	p_cur	输入字符串
返回值	如果有更多参数返回1，如果没有返回0		

6 WelinkOpen GSDK RIL API介绍

本章规定了ME3616模块WelinkOpen的Ril相关API接口，用于帮助客户使用和管理AT相关的业务活动。

6.1 Ril接口函数

6.1.1 gsdk_ril_init

Function	gsdk_status_t gsdk_ril_init(void)		
描述	该函数为初始化ril环境。		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.2 gsdk_ril_close

Function	void gsdk_ril_close(void)		
描述	该函数为关闭ril环境，是gsdk_ril_init的反操作。		
参数	**Type**	**参数**	**描述**
	void	void	无
返回值	无		

6.1.3 gsdk_at_command

Function	gsdk_status_t gsdk_at_command(const char *command, at_response_t **pp_response)

描述	该函数为发送AT指令。		
参数	**Type**	**参数**	**描述**
	char	command	at command
	at_response_t	pp_response	at命令的 response值
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.4 gsdk_at_command_singleline

Function	gsdk_status_t gsdk_at_command_singleline(const char *command, const char *response_prefix, at_response_t **pp_response)		
描述	该函数为发送AT指令，只返回一行中间结果。		
参数	**Type**	**参数**	**描述**
	char	command	at command
	char	response_prefix	该at command的at response的前缀
	at_response_t	pp_response	at命令的 response值
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.5 gsdk_at_command_numeric

Function	gsdk_status_t gsdk_at_command_numeric(const char *command, at_response_t **pp_response)		
描述	该函数为发送AT指令，返回的中间结果全部为数字。		
参数	**Type**	**参数**	**描述**
	char	command	at command
	char	pp_response	at命令的 response值
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.6 gsdk_at_command_multiline

Function	gsdk_status_t gsdk_at_command_multiline(const char *command, const char *response_prefix, at_response_t **pp_response)		
描述	该函数为发送AT指令，返回的中间结果多行。		
参数	Type	参数	描述
	char	command	at command
	char	response_prefix	该at command的at response的前缀
	at_response_t	pp_response	at命令的 response值
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.7 gsdk_atport_open

Function	gsdk_status_t gsdk_atport_open(gsdk_handle_t *hport)		
描述	该函数为打开at port功能。		
参数	Type	参数	描述
	gsdk_handle_t	*hport	atport句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.8 gsdk_atport_close

Function	gsdk_status_t gsdk_atport_close(gsdk_handle_t hport)		
描述	该函数为关闭at port功能。		
参数	Type	参数	描述
	gsdk_handle_t	hport	atport句柄
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.9 gsdk_atport_read

Function	int gsdk_atport_read(gsdk_handle_t hport, uint8_t *data, int len, int timeout_ms)
描述	该函数为atport读取功能。

参数	Type	参数	描述
	gsdk_handle_t	hport	atport句柄
	uint8_t	*data	读取数据的指针
	int	len	读取数据长度
	int	timeout_ms	超时时间
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.10 gsdk_atport_write

Function	int gsdk_atport_write(gsdk_handle_t hport, uint8_t *data, int len, int timeout_ms)		
描述	该函数为atport写操作功能。		
参数	Type	参数	描述
	gsdk_handle_t	hport	atport句柄
	uint8_t	*data	写数据的指针
	int	len	写数据长度
	int	timeout_ms	超时时间
返回值	成功返回0，失败返回错误码，错误码见下表7-1		

6.1.11 gsdk_atport_status

Function	gsdk_atport_status_t gsdk_atport_status(gsdk_handle_t hport)		
描述	该函数为获取atport状态。		
参数	Type	参数	描述
	gsdk_handle_t	hport	atport句柄
返回值	成功返回1，失败返回0。		

7 错误码表

表7-1 错误码列表

错误码列表信息	
GSKD_ERROR_UNINITIALIZED	-8
GSDK_ERROR_HARDWARE	-7
GSDK_ERROR_INTERNAL	-6
GSDK_ERROR_TIMEOUT	-5
GSDK_ERROR_DEVICE_BUSY	-4
GSDK_ERROR_INSUFFICIENT_RESOURCE	-3
GSDK_ERROR_INVALID_PARAMETER	-2
GSDK_ERROR	-1