# Mooresilicon Cygnus SDK BLE运行设计

© 2022 Mooresilicon All rights reserved.

本文档版权归TCL摩星半导体所有，受相关法律法规的保护。未经书面许可不得复制传播。

# 修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 作者 | 说明 |
| draft | 2022-04-15 | 区伟权 | 起草 |
| 1.0 | 2022-04-19 | 区伟权 | 初稿评审完成：  调整内存分配布局 |

目录

[Mooresilicon Cygnus SDK BLE运行设计 1](#_Toc100944822)

[修订历史 2](#_Toc100944823)

[目的 3](#_Toc100944824)

[范围 3](#_Toc100944825)

[概述 3](#_Toc100944826)

[设计约束 3](#_Toc100944827)

[零层设计 3](#_Toc100944828)

[模块分解 4](#_Toc100944829)

[APP 4](#_Toc100944830)

[HOST 5](#_Toc100944831)

[L2CAP 5](#_Toc100944832)

[GAP 5](#_Toc100944833)

[GATT 6](#_Toc100944834)

[Profiles 6](#_Toc100944835)

[HCI 6](#_Toc100944836)

[BLE Controller 7](#_Toc100944837)

[RWIP 9](#_Toc100944838)

[中断处理 10](#_Toc100944839)

[内存管理 10](#_Toc100944840)

[KERNEL 11](#_Toc100944841)

[事件处理 12](#_Toc100944842)

[消息处理 13](#_Toc100944843)

[RF 14](#_Toc100944844)

[任务运行设计 14](#_Toc100944845)

[内存分配 15](#_Toc100944846)

# 目的

SDK支持多任务系统，更好地适应产品多样性和业务多样性。集成FreeRTOS以获得操作系统多任务并发运行基础。BLE作为SDK的主要业务模块之一，需要做基于FreeRTOS的任务运行设计；同时考虑，简单产品的单进程兼容。有以下设计目标：

* 基于FreeRTOS的任务运行设计；
* 单进程运行设计兼容；
* BLE版本升级的快速移植；

# 范围

本文用于指导BLE模块基于FreeRTOS的任务运行设计。读者是相关软件开发工程师。

# 概述

芯片集成CEVA BLE IP，软件SDK集成CEVA BLE SW，配套支持BLE协议栈HOST + Controller 。可支持蓝牙相关IOT产品的应用开发。目前主要考虑产品应用是蓝牙遥控器。

# 设计约束

硬件约束：

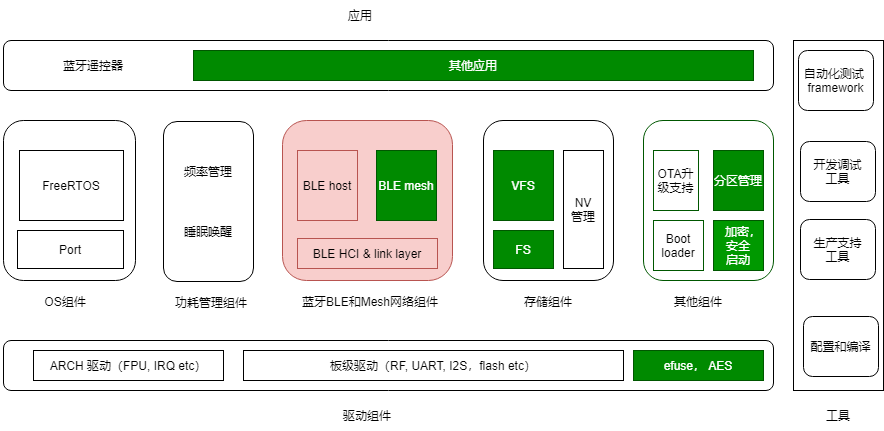
* 配套支持CEVA BLE IP；
* RAM 128Kbytes + RetRAM 4KBytes

软件约束：

* 基于FreeRTOS多任务开发；
* 基于当前CEVA1.1版本开发；

# 零层设计

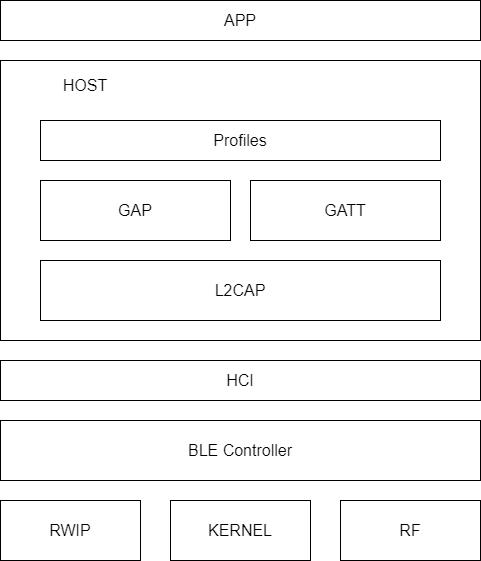
BLE作为SDK的组件之一，位置如下图所示。支持蓝牙BLE HOST + HCI &LL层协议栈实现。BLE Mesh功能暂不实现。对应用层提供API接口。应用层任务调用API接口，向BLE发送消息，实现蓝牙业务控制。BLE调用FreeRTOS的任务管理接口，实现多任务调度运行。



# 模块分解

BLE模块分解如下图所示。各模块做任务实例化，任务间使用消息通信。Controller LLD主要控制芯片IP，通过中断调度运行。

BLE KERNEL模块支持简单的任务调度，和任务间的消息通信。实际上是基于任务ID和消息ID的消息处理。各模块任务消息入队，调度周期消息出队，根据目的任务ID执行相应的消息处理函数。



## APP

实例化为任务运行，对应用层提供API接口。API接口实现为函数，向HOST发送相关控制消息。

BLE程序的主入口，执行main init和main loop循环事件调度。

## HOST

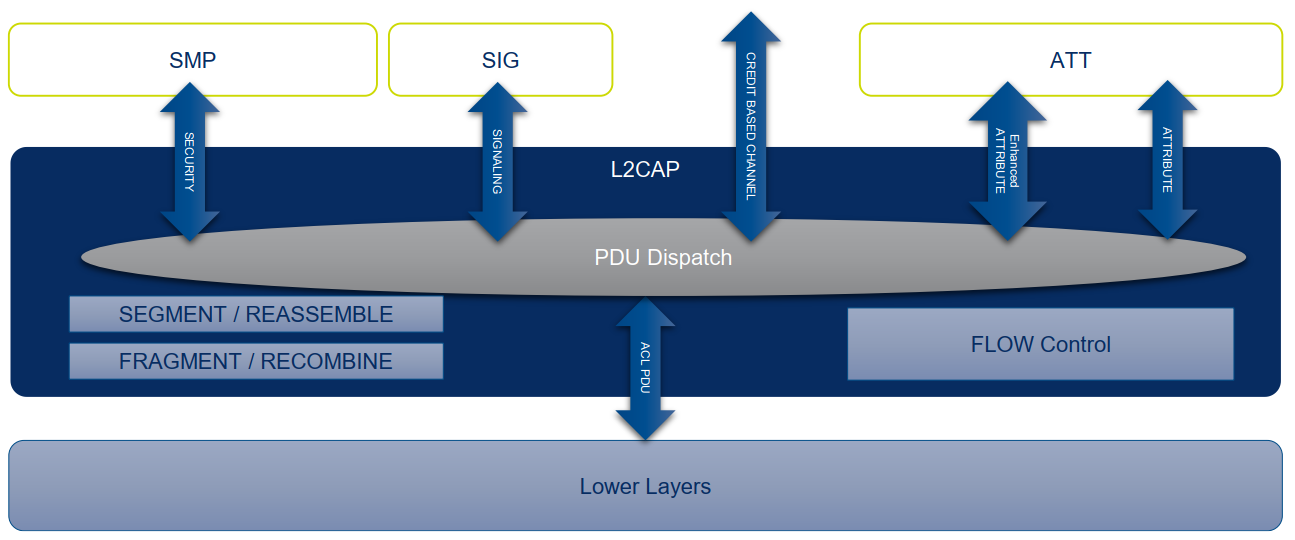
Host端协议栈实现，划分为四个子模块：

1. L2CAP Logical Link Control and Adaptation Protocol
2. GAP Generic Access Profile
3. GATT Generic Attribute Profile
4. Profiles

AHI是PC端应用通过AHI控制板端的HOST ，当前架构不涉及。

### L2CAP

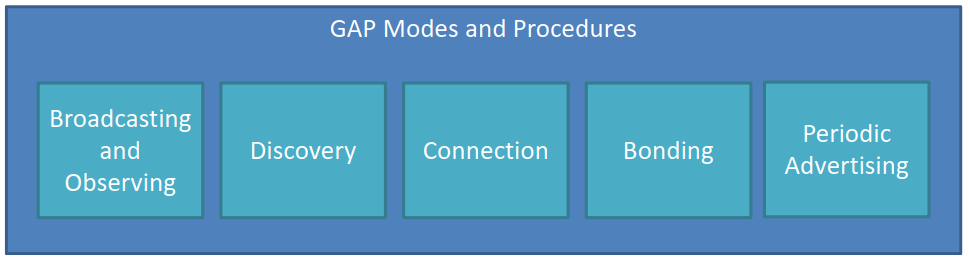
L2CAP协议层实现。负责BLE链路对端L2CAP建立通道和传输SDU 。



SIG（Special Interest Group）是L2CAP的子模块，实现Signaling protocol 。SIG保存设备支持的协议，服务和安全等级，由上层来注册和配置。

### GAP

用于与其它BLE设备建立连接。



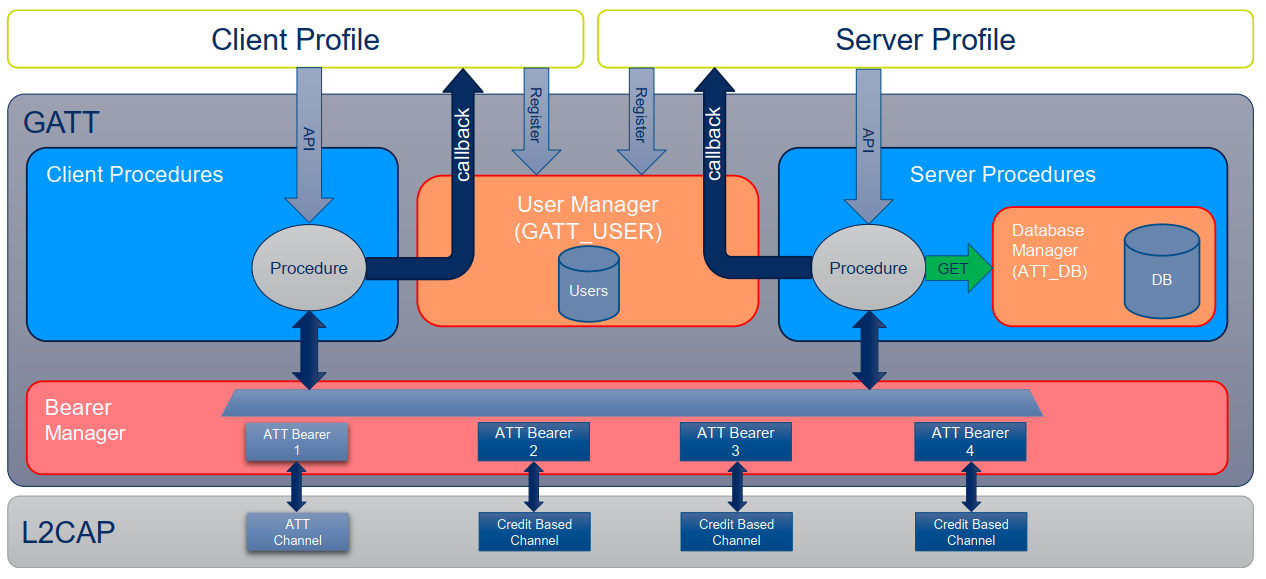
SMP是GAP的子模块，实现Security Manager Protocol，建立加密连接。

GAP对上层提供接口：

* LE ping；
* 协商报文长度；
* 协商速率；
* Profile任务管理，Client和Service的任务建立；

### GATT

ATT Attribute Protocol 是GATT的子模块。GATT通过ATT发现、读、写、获取Server属性，配置广播属性。对上层Profile做User管理。

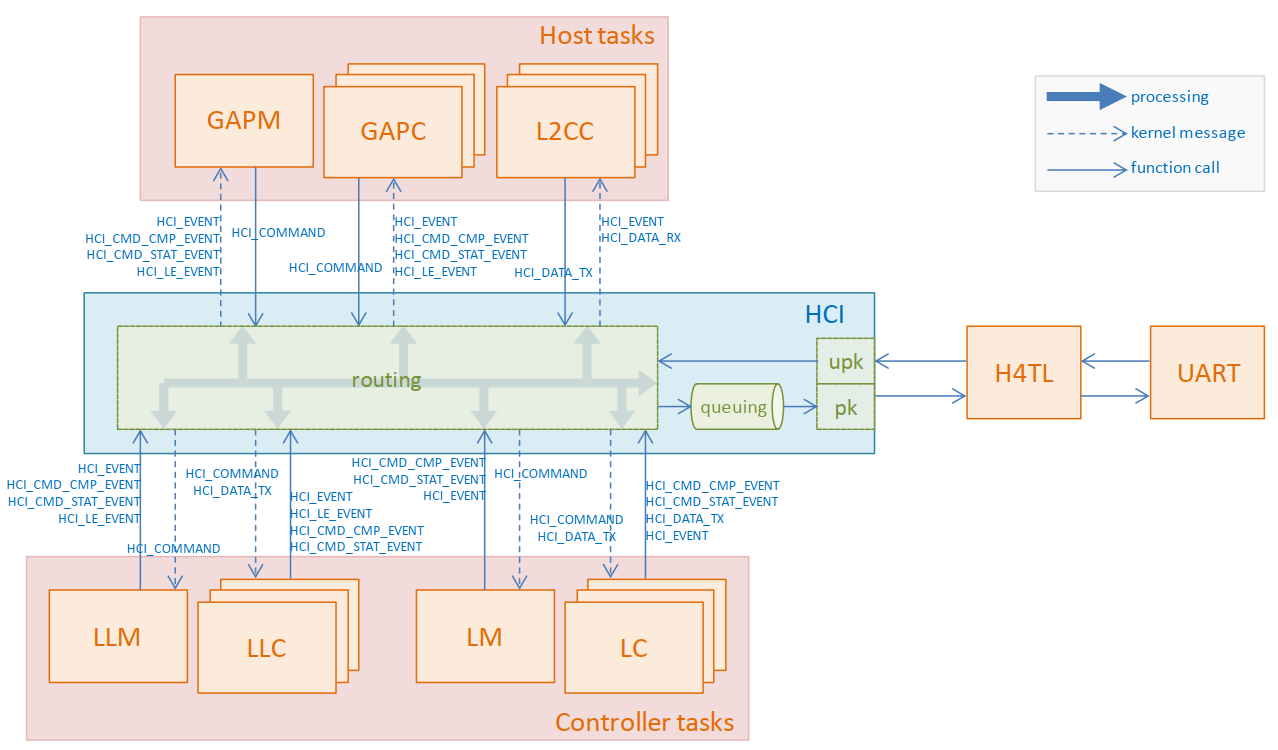


### Profiles

业务相关的Profile 。可以是Client，或者Server 。通过GAP建立任务和对端连接，通过GATT管理业务属性。

## HCI

Host Controller Interface ，Host和Controller之间的控制接口。Host和ControllerLLM/LLC部分都是task，它们之间通过HCI消息通信。

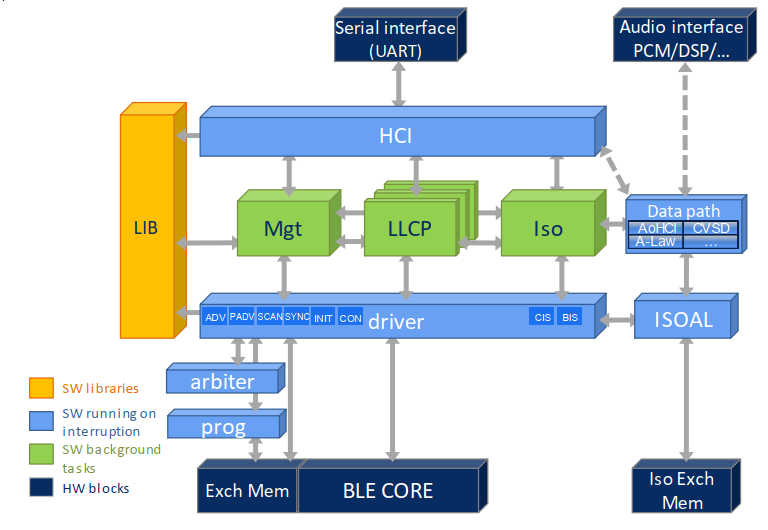


H4TL和UART用于与外部的Host，或者Controller通信，当前架构不涉及。

## BLE Controller

BLE链路层实现。支持以下功能：

* 设备发现，广播和扫描；
* 建立连接；
* 非连接数据的广播和侦听；
* 设备过滤，白名单和广播列表；
* 设备地址解析
* 多于10个的同步处理，master或者slave的连接/广播/扫描/初始化；
* Slave latency
* 异步数据的接收和发送；（数据长度最大251字节，速率1Mpbs，或者2Mbps）
* 信道分配；
* 加密；
* 多点和分散网（scatternet）的管理；
* BLE链路层的标准协议处理；
* 同步广播或者连接；



|  |  |
| --- | --- |
| 子模块 | 描述 |
| **RWBLE** | Main module of the RW-BLE SW; its first role is to manage the entire Controller SW for general procedures such as initialization, reset, deep sleep. It also provides an interface to the platform SW part (e.g. from system boot loader, interrupt handlers, etc…) |
| HCI | The Host Controller Interface component manages the HCI packets exchanges with Host. It redirects incoming packets to the right lower level component and provides API for sending HCI packet. The HCI component is described at [4]. |
| **Mgt** | The management task is referred as Link Layer Manager (LLM) in the implementation. It is a non-real time SW task. Its role is to manage non-connected operations. It handles all HCI commands except those associated to a specific connection. |
| **LLCP** | The task handling Link Layer Control Protocol is referred as Link Layer Controller (LLC) in the implementation. It is a multi-instantiated task, where each instance is associated to a BLE link with a peer device. It handles connection-oriented HCI commands and ACL data. It manages ACL data transmission/reception and implements the Link Layer Control Protocol (LLCP) procedures defined at BT standard specification part VI.B.5 [1], and the message sequence charts at VI.D [1]. |
| **Iso** | The task handling isochronous activities is referred as Link Layer Isochronous (LLI) in the implementation. It is a non-real time SW task. Its role is management of isochronous streams and groups with their corresponding data path. It manages life cycle of isochronous connection isochronous (CIS and CIG) or Broadcast Isochronous Stream and Group (BIS and BIG). |
| **Driver** | The Link Layer Driver module (LLD) is the RW-BLE Core driver, responsible of the real-time part of the Link Layer operations. It schedules the activities in collaboration with the scheduling arbiter. Then it performs the events thanks to the programming of the RW-BLE BaseBand Core. |
| **ISOAL** | Isochronous adaptation layerAdaptation Layer is a real-time SW task. It manages “fragmentation and recombination” or “segmentation and reassembly” of SDU to isochronous payload packet. It also manages the SDU interval and offset with Isochronous stream. |
| **Data Path** | Data path drivers is a container of plug-ins partially real-time responsible of handling SDU to transmit or received from an isochronous stream. It can be audio codecs, Audio over HCI, etc... With a simple interface, new driver can be added by customer. |
| **Arbiter** | The Arbiter is the central point for scheduling any air activity. It arbitrates the access to the HW resources between all the activities. It is described at [2]. |
| **Programmer** | The Programmer is the central point for programming any air activity from the drivers. It requests the HW to start new frames and route the out coming interrupts to the appropriate drivers. It is described at [2]. |

LLD driver与baseband core时间同步。基于625us时钟组织自己master连接；也可以作为slave跟踪其它的master连接，永久与master保持时钟同步。LLD从后台任务接收请求，在baseband core中断中执行处理。

LLM，LLC，和LLI处理上层协议部分，如HCI下发的命令、设备参数和配置，LLCP协商，和其它没有实时限制的工作。它们都实现为task，并运行在后台。

## RWIP

功能规格：

* 读取硬件版本；
* 获取baseband core时间，和下一个tick的时间；
* Half us timer，用于scheduling arbiter；
* Half Slot timer，用于scheduling alarm；
* 10 ms timer，用于kernel timer；
* 软件中断；
* 睡眠算法；
* 内存管理

### 中断处理

SCH Arbiter 服务于LLD，基于中断实现周期同步的收发帧和告警的事件处理。运行在中断上下文。中断明细如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 中断 | 描述 | 处理函数 | 处理描述 |
| IP\_TIMESTAMPTGT1INTSTAT\_BIT | General purpose timer interrupt - half us accuracy | rwip\_timer\_arb\_handler | handles arbiter timer target SCH基于此中断调度处理LLD事件，产生告警或者收发帧 |
| IP\_TIMESTAMPTGT2INTSTAT\_BIT | General purpose timer interrupt - half us accuracy | rwip\_timer\_alarm\_handler | handles alarm timer target SCH基于此中断处理LLD告警 |
| IP\_CLKNINTSTAT\_BIT | clock interrupt | rwip\_wakeup\_end | Handle end of wake-up Enable the core and check if some timer target has been reached. |
| IP\_TIMESTAMPTGT3INTSTAT\_BIT | General purpose timer interrupt - half us accuracy | rwip\_timer\_co\_handler | handles common timer target 产生TIMER事件 |
| IP\_SWINTSTAT\_BIT | SW interrupt | rwip\_sw\_int\_handler | 触发SCH调度 |
| IP\_FIFOINTSTAT\_BIT | FIFO interrupt | sch\_prog\_fifo\_isr | Call scheduling programmer SCH基于此中断处理收发帧 |
| IP\_SLPINTSTAT\_BIT |  | rwip\_wakeup | Handle wake-up Compute and apply the clock correction according to duration of the deep sleep. |
| IP\_CRYPTINTSTAT\_BIT | Encryption interrupt | rwip\_crypt\_isr\_handler | the crypto ISR handler 加解密完成中断，获取结果 |
| IP\_FINETGTINTSTAT\_BIT | General purpose timer interrupt - half slot accuracy | NA | Finetgt unused |

### 内存管理

BLE运行内存是全局变量数据管理。RAM总需求43Kbytes ，包括软件运行内存18Kbytes和与芯片IP共享运行的EM内存25Kbytes ；RetRAM需求是3.4Kbytes 。明细如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 描述 | block | size |
| KE\_MEM\_ENV | Memory allocated for environment variables | BT\_HEAP\_ENV\_SIZE\_ | 0 |
|  |  | BLE\_HEAP\_ENV\_SIZE\_ (ACT=12) | 3360 |
|  |  | HOST\_HEAP\_ENV\_SIZE\_ CONNECTION=3 ACT=12 CHNL=20 | 4348 |
| total |  |  | 7708 |
| KE\_MEM\_PROFILE | Memory allocated for profiles | RWIP\_HEAP\_PROFILE\_SIZE | 3072 |
| KE\_MEM\_KE\_MSG | Memory allocated for kernel messages | BT\_HEAP\_MSG\_SIZE\_ | 0 |
|  |  | BLE\_HEAP\_MSG\_SIZE\_ | 7874 |
|  |  | HOST\_HEAP\_MSG\_SIZE\_ CON=10 | 2048 |
| SW KE total |  |  | 9922 |
| KE\_MEM\_NON\_RETENTION | Non Retention memory block The current size show what is already known as not needing to be retained during deep sleep. 2 ECC computations simultaneously | RWIP\_HEAP\_NON\_RET\_SIZE | 656 |
| total |  |  | 18286 |
| BLE EM | The HW interface provided by RW-BLE Core is made of registers, a shared memory named “Exchange Memory” and interrupts. | RW-BLE-EM-Map.xlsx | 24158 |
| RAM total |  |  | 42444 |
| In rentention |  |  | 3424 |

## KERNEL

BLE自带的简易实时操作系统，支持规格：

* 事件管理；
* 消息管理；
* 定时器；
* 延迟处理；
* 事件调度器在main\_loop执行；
* 事件调度器检查事件是否非空，是则根据事件比特执行相应的事件处理函数；
* 任务管理，任务跟事件有绑定关系；

### 事件处理

事件是BLE SW的基础机制。各模块有处理的事件，和产生事件。在main\_loop执行事件调度，遍历事件标志，执行相应模块的事件处理。处理事件的模块负责清理事件标志。事件ID越大，优先级越高。最高支持优先级31 。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | 模块 | 处理函数 | 功能描述 |
| APP\_EVENT | APP | app\_start | 初始化向GAP发送配置命令 |
| KE\_EVENT\_DISPLAY | RWIP.display | display\_user\_action | 显示输出，如LCD；当前不支持 |
| KE\_EVENT\_RTC\_1S\_TICK | NA | NA | 不支持 |
| KE\_EVENT\_RSA\_SIGN | NA | NA | 不支持 |
| KE\_EVENT\_ECC\_MULTIPLICATION | security.ecc | ecc\_multiplication\_event\_handler | 生产Dhkey，LLM调用 |
| KE\_EVENT\_P192\_PUB\_KEY\_GEN | NA | NA | 不支持 |
| KE\_EVENT\_TRC | dbg | dbg\_trc\_ke\_evt\_cbk | DBG TRACE相关，不支持 |
| KE\_EVENT\_DJOB\_LP | djob | co\_djob\_lp\_evt\_handler | 低优先级延迟任务事件处理；HOST使用； |
| KE\_EVENT\_KE\_MESSAGE | KERNEL | ke\_task\_schedule | 内核任务消息调度处理；各模块的任务实例基于此事件进行消息通信； |
| KE\_EVENT\_DJOB\_HP | djob | co\_djob\_hp\_evt\_handler | 高优先级延迟任务事件处理；HOST使用； |
| KE\_EVENT\_AES\_END | RWIP.driver | rwip\_crypt\_evt\_handler | AES加解密结束事件；由中断触发，软件读取加解密数据返回 |
| KE\_EVENT\_TIMER | timer | co\_timer\_handler | 软件定时器 |
| KE\_EVENT\_ISOOHCI\_IN\_DEFER | DATA\_PATH.isoohci | isoohci\_in\_defer\_handler | 不支持 |
| KE\_EVENT\_ISOOHCI\_OUT\_DEFER | DATA\_PATH.isoohci | isoohci\_out\_defer\_handler | 不支持 |
| KE\_EVENT\_DJOB\_ISR | djob | co\_djob\_isr\_evt\_handler | 中断延迟服务，用于h4tl和isopcm；当前都不支持 |

### 消息处理

模块可以创建自己的task实例，可定义task需要处理的消息。Task发送消息，消息入队；内核任务消息调度处理，消息出队，根据消息ID，task id 找到对应的消息处理函数。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| task ID | 模块 | task实例数 | 消息 | 功能 |
| TASK\_LM | NA | NA | NA | BT Controller LM，不支持 |
| TASK\_LC | NA | NA | NA | BT Controller LC，不支持 |
| TASK\_LB | NA | NA | NA | BT Controller LB，不支持 |
| TASK\_LLM | LLM | 1 | KE\_MSG\_HANDLER\_TAB(llm) | 消息类型： LLM->LLM LLD->LLM HCI->LLM |
| TASK\_LLC | LLC | 12 | KE\_MSG\_HANDLER\_TAB(llc) | 消息类型： LLM->LLC LLC->LLC LLD->LLC HCI->LLC |
| TASK\_LLI | LLI | 1 | KE\_MSG\_HANDLER\_TAB(lli) | 不支持 |
| TASK\_DBG | dbg | 1 | KE\_MSG\_HANDLER\_TAB(dbg) | LL层Debug功能 |
| TASK\_DISPLAY | RWIP.display | 1 | display\_msg\_handler\_tab | 显示输出，如LCD；当前不支持 |
| TASK\_APP | APP | 1 | KE\_MSG\_HANDLER\_TAB(app) | 处理GAP消息 |
| TASK\_L2CAP | L2CAP | 1 | KE\_MSG\_HANDLER\_TAB(l2cap) | CMD/CFM消息处理 |
| TASK\_GATT | GATT | 1 | KE\_MSG\_HANDLER\_TAB(gatt) | CMD/CFM消息处理 |
| TASK\_GAPC | GAPC | 10 | KE\_MSG\_HANDLER\_TAB(gapc) |  |
| TASK\_GAPM | GAPM | 1 | KE\_MSG\_HANDLER\_TAB(gapm) |  |
| TASK\_PRF\_MIN TASK\_PRF\_MAX | Profiles | 10 | 各Profile的消息列表 KE\_MSG\_HANDLER\_TAB(x) | 由GAPM协议控制创建的Profile；GAPM发消息给APP，APP的消息处理创建对应的Profile； 共支持10个Profile； Profiles模块初始化为每个Profile创建1个task实例；GAPM发消处理给task绑定Profile的消息处理 |
| TASK\_AHI | AHI | 1 | KE\_MSG\_HANDLER\_TAB(ahi) | 当前架构不支持AHI |
| TASK\_GAF | AHI | 0 | NA |  |

## RF

射频驱动

# 任务运行设计

**BLE任务**，使用FreeRTOS任务管理接口创建。属于业务任务优先级。任务主入口是APP的blemain，完成初始化后，进入main\_loop事件调度处理。如果当前没有事件处理，则调用FreeRTOS任务调度接口主动释放CPU 。BLE任务进入就绪状态，等待下一次任务调度。

**应用层任务**，如按键任务。业务任务优先级。业务处理过程，调用BLE API ，产生消息事件。BLE任务被调度唤醒后，发现有消息事件待处理，则执行KERNEL的消息调度，找到HOST对应的task id处理该消息。如业务是把按键消息发往对端的，则HOST发消息到Controller。此处事件已全部处理完成，BLE任务再次释放CPU。

**BLE中断**，周期触发Controller LLD调度。消息由LLD控制，基于中断调度，周期同步发送到对端。蓝牙的同步周期是625us，为达到该性能要求，LLD调度在中断上下文处理，不考虑中断延迟任务处理。

运行过程如下图所示：

T1时刻，BLE任务运行，进行初始化；

T2时刻，BLE任务进入main\_loop，发现没有事件处理，主动释放CPU；

T3时刻，按键任务IO资源就绪，获取CPU调度运行；调用BLE API接口，产生事件；

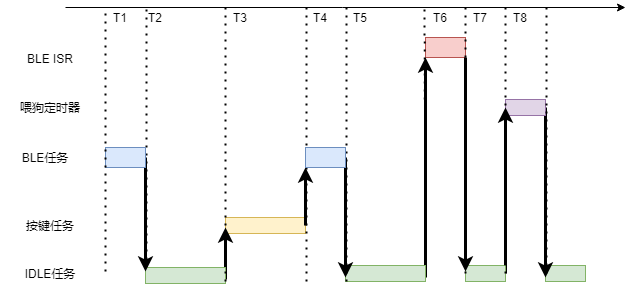
T4时刻，按键任务因IO资源再次阻塞释放CPU；BLE任务被调度，处理事件；

T5时刻，HOST按键业务发消息到Controller；事件处理完成，BLE任务再次释放CPU；无就绪任务，IDLE任务运行；

T6时刻，BLE周期中断触发ISR，同步传输数据帧到对端；

T7时刻，BLE周期中断结束，CPU回到IDLE任务；

T8时刻，喂狗定时器到时，执行喂狗操作；



**任务修改点**：

* 主进程创建BLE任务，任务处理进入BLE的main入口；
* main\_loop 在事件调度完成后，增加释放CPU；
* 释放CPU操作，使用宏定义，可编译控制。如果支持FreeRTOS则调用其任务调度接口主动释放CPU；

**单进程兼容**：

* 主进程调用BLE的main入口；
* 释放CPU的宏定义，编译控制，改为主动喂狗；

# 内存分配

总内存128Kbytes，BLE需求43Kbytes，RWIP预留8k，FreeRTOS需求40Kbytes，剩余37Kbytes给其它模块全局变量使用。