[GPRS及GPRS C SDK 从零开始文档](https://github.com/Ai-Thinker-Open/GPRS-C-SDK/blob/master/doc/gprs\_start\_from\_scratch\_zh-cn.md)

===========

在开始之前，请保证开发环境和下载软件已经安装好并且可以正常使用，

参考[GPRS C SDK 开发环境搭建](./compile\_environment\_zh-cn.md)

和[GPRS 下载调试说明](./download\_debug\_tool\_zh-cn.md)

## (一) SDK 目录结构

| 目录 | 描述 |

| --- | --- |

|app | 程序主目录，应用代码放在这里|

|build | 编译生成的目录、中间文件 |

|demo | 一些例程 |

|doc | 一些SDK相关的文档，Markdown格式，可在github在线阅读，也可用相关软件打开阅读，官网Wiki的压缩包会提供`html`格式的文档，可以直接用浏览器打开阅读

|hex | 最后产生的可烧录文件 |

|include | SDK文件目录 |

|init | 系统初始化的目录，可以不用理会，不建议改动 |

## （二）例程

在`demo`目录下有各种例程，可以先从简单的GPIO开始

### GPIO 引脚使用

\* \*\*编译一个看看现象\*\*：`./build.sh demo gpio`,然后下载到开发板，第一次要先烧录`hex`目录下较大的文件`\*\*\_B\*\*.lod`,后面只需烧录小文件``\*\*\_flash.lod`

烧录完成后开发板的LED会闪烁

\* \*\*代码\*\*： 整个demo代码量很少，如下：

```

#include <api\_os.h>

#include <api\_hal\_gpio.h>

#include "demo\_gpio.h"

#define GPIO\_PIN\_LED\_BLUE GPIO\_PIN27

#define GPIO\_Main\_TASK\_STACK\_SIZE (1024 \* 2)

#define GPIO\_Main\_TASK\_PRIORITY 1

HANDLE gpioTaskHandle = NULL;

void Gpio\_MainTask(VOID \*pData)

{

static GPIO\_LEVEL ledBlueLevel = GPIO\_LEVEL\_LOW;

GPIO\_config\_t gpioLedBlue = {

.mode = GPIO\_MODE\_OUTPUT,

.pin = GPIO\_PIN\_LED\_BLUE

};

GPIO\_Init(gpioLedBlue);

while(1)

{

ledBlueLevel = (ledBlueLevel==GPIO\_LEVEL\_HIGH)?GPIO\_LEVEL\_LOW:GPIO\_LEVEL\_HIGH;

Trace(1,"gpio\_main toggle:%d",ledBlueLevel);

GPIO\_SetLevel(gpioLedBlue,ledBlueLevel); //Set level

OS\_Sleep(1000); //Sleep 1 s

}

}

void gpio\_Main(void)

{

gpioTaskHandle = OS\_CreateTask(Gpio\_MainTask ,

NULL, NULL, GPIO\_Main\_TASK\_STACK\_SIZE, GPIO\_Main\_TASK\_PRIORITY, 0, 0, "main Task");

OS\_SetUserMainHandle(&gpioTaskHandle);

}

```

\* \*\*入口函数\*\*：入口函数为`gpio\_Main`，

\* 其中`gpio\_Main`是入口函数，名字必须是`demo文件夹名\_Main()`

在`gpio\_Main()`中必须创建一个主要任务，没有用过多线程和操作系统的也没关系，这里只用了最简单的函数，`OS\_CreateTask`创建了一个任务，第一个参数是任务的函数名，`GPIO\_Main\_TASK\_STACK\_SIZE`和`GPIO\_Main\_TASK\_PRIORITY`分别是任务堆栈大小以及任务优先级（数值越小优先级越高）

\* 然后是`OS\_SetUserMainHandle`，这个函数必须要调用一次，很重要。它主要是告诉SDK 应用程序的主任务是哪一个，这样SDK底层才知道给谁发送事件（比如SDK发现没有插卡，需要发送`API\_EVENT\_ID\_NO\_SIMCARD`事件给主任务

\* \*\*GPIO使用\*\*：创建好了主任务，就可以在主任务中操作GPIO了，这里创建的任务函数为：`Gpio\_MainTask()`

\* 定义一个结构体`GPIO\_config\_t gpioLedBlue`，设置引脚和模式

```

gpioLedBlue mode = GPIO\_MODE\_OUTPUT; //输出模式

gpioLedBlue pin = GPIO\_PIN\_LED\_BLUE; //引脚为GPIO27脚

```

\* 然后使用`GPIO\_Init()`函数初始化GPIO

\* 接下来就可以调用`GPIO\_SetLevel()`在`while(1){}`中来设置GPIO输出的电平高低啦

\* 需要注意的是，在目前的版本SDK中，最好在while(1)中加入一个`OS\_Sleep(1)`，否则可能导致下载程序比较麻烦！如何解决这个问题在下载调试的文档里说了

当然GPIO还有其它功能，比如作为输入，作为中断引脚，在例程中被注释了，可以自己试试哦

### 最小系统、操作系统(OS)使用

相比于GPIO，OS的例程的主任务不再是直接在`while(1){}`中直接操作，而是在主任务中监听来自SDK的事件，事件的定义以及参数的含义在`api\_event.h`中定义和说明。需要做其它事情可以再创建一个或者多个任务

\* \*\*应用最基本结构\*\*

主任务应该按照如下方式写：

```

void EventDispatch(API\_Event\_t\* pEvent);

void socket\_MainTask(void \*pData)

{

API\_Event\_t\* event=NULL;

while(1)

{

if(OS\_WaitEvent(socketTaskHandle, &event, OS\_TIME\_OUT\_WAIT\_FOREVER))

{

EventDispatch(event);

OS\_Free(event->pParam1);

OS\_Free(event);

}

}

}

```

其中`OS\_WaitEvent`是阻塞等待事件到来，暂时不支持非阻塞（即第三个参数只能为`OS\_TIME\_OUT\_WAIT\_FOREVER`）

然后在`EventDispatch`函数中处理事件，处理完后释放内存，这里`OS\_Free`会自动判断是否合法

```

void EventDispatch(API\_Event\_t\* pEvent)

{

switch(pEvent->id)

{

case API\_EVENT\_ID\_NO\_SIMCARD:

Trace(10,"!!NO SIM CARD%d!!!!",pEvent->param1);

break;

case API\_EVENT\_ID\_NETWORK\_REGISTERED\_HOME:

case API\_EVENT\_ID\_NETWORK\_REGISTERED\_ROAMING:

Trace(2,"network register success");

break;

default:

break;

}

}

```

这里主要处理了两个事件，

一个是没有读到SIM卡事件，`param1`参数代表了卡号，在`api\_event.h`中有说明

另一个是注册网络成功事件，分别是成功注册到本地网络和漫游网络，注册成功后会接收到这个事件，并打印调试信息，调试信息需要使用coolwatcher的tracer功能查看，如何调试参见下载调试文档

\* OS的其它功能

包括发送事件、信号量等可以参见头文件

### socket 连接服务器

要使用socket，需要做以下几步：

\* 1. \*\*注册到基站网络\*\*

开机会自动注册，注册完成后会参数`API\_EVENT\_ID\_NETWORK\_REGISTERED\_HOME`或者`API\_EVENT\_ID\_NETWORK\_REGISTERED\_ROAMING`事件

\* 2. \*\*附着到基站网络\*\*

注册成功后，使用`Network\_StartAttach()`进行附着，附着成功后触发`API\_EVENT\_ID\_NETWORK\_ATTACHED`事件

\* 3. \*\*激活GPRS上下文\*\*

附着成功后，使用如下语句进行上下文激活

```

Network\_PDP\_Context\_t context = {

.apn ="cmnet",

.userName = "" ,

.userPasswd = ""

};

Network\_StartActive(context);

```

其中，`context`即为定义的上下文，设置正确的apn以及其它信息，这里使用的移动物联网卡，只需设置apn为`cmnet`即可

激活成功后会触发`API\_EVENT\_ID\_NETWORK\_ACTIVATED`事件，到此模块就具备GPRS联网能力了

\* 4. \*\*创建socket连接并通信\*\*

参见例程

调用socket相关的API建立通信即可：

```

int fd = Socket\_TcpipConnect(TCP,SERVER\_IP,SERVER\_PORT);

Socket\_TcpipWrite(fd,buffer,length);

length = Socket\_TcpipRead(fd,buffer,length);

```

目前这几个API是基于事件的API，及建立连接成功后会产生`API\_EVENT\_ID\_SOCKET\_CONNECTED`，接收到数据后会产生`API\_EVENT\_ID\_SOCKET\_RECEIVED`事件等，可以在socket例程中查看，

后续版本会更新阻塞方法，使用起来更加简单，在一些简单的应用中使用起来更加方便

### 其它模块的使用请参照例程