STM32WB BLE 应用低功耗设计

关键字：STM32WB，BLE，低功耗

前言

功耗是物联网应用当中非常关键的因素，在开发的早期都会对功耗进行评估和测试。那么，如何使用 ST 提供的工具对动态功耗进行测量呢？针对 BLE 应用应当如何进行低功耗的设计呢？本篇跟大家一起聊聊该话题。

测量工具

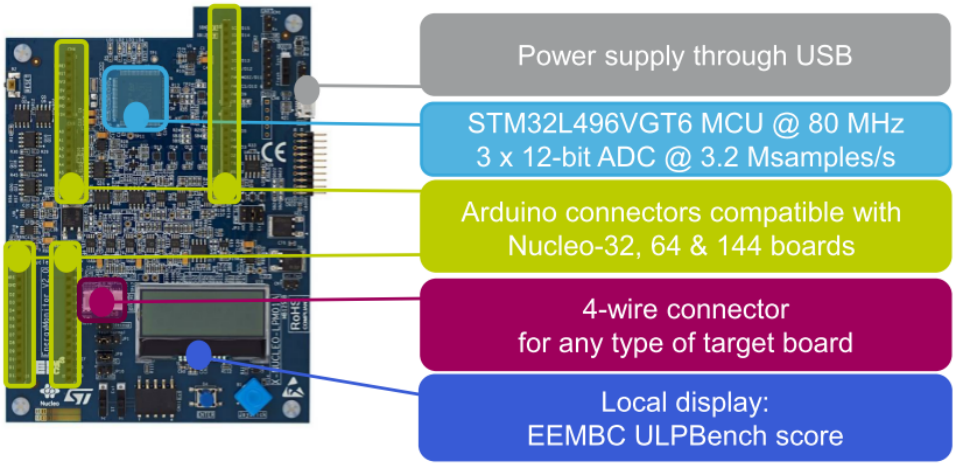
在本文中使用 CubeMonitor-Power + Power Shield 的组合工具对 STM32Nucleo 开发板的动态功耗进行测量。

CubeMonitor-Power 是 ST 提供的上位机软件，主要帮助用户动态的测量功耗。用户可以设定采样频率，采样时间，输入电压等。其界面如下：

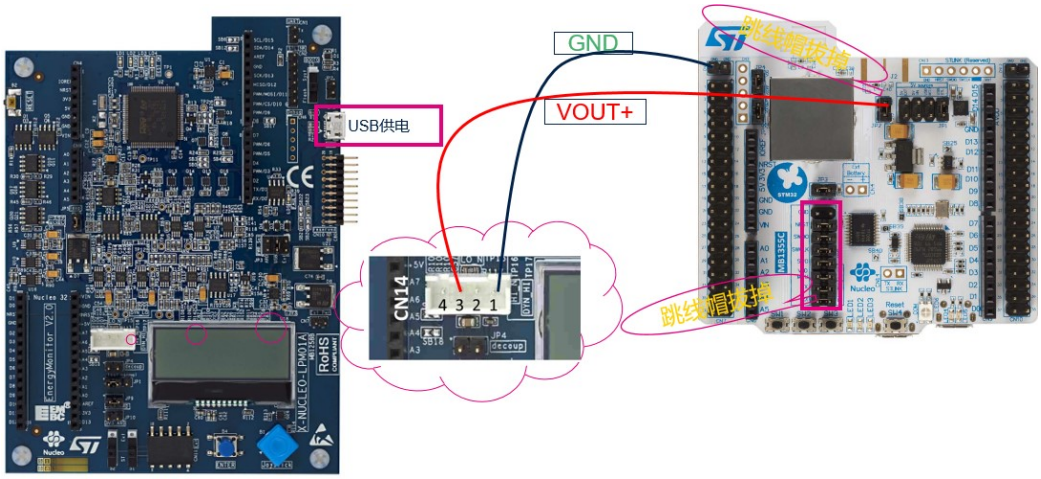
X-NUCLEO-LPM01A 是 ST 提供的一块功耗测量的评估板，它可以配合 CubeMonitor-Power 上位机软件，对目标板的动态功耗进行测量，方便开发者对功耗进行评估。其特性如下：

- 可编程电压源范围:1.8v~3.3v
- 静态测量
    - 电流范围：1nA~200mA

- 动态测量：
  - 电流范围：100nA~50mA
  - 100KHz 带宽，3.2Msps 采样率
  - 功耗测量范围：180nW~165mW



X-NUCLEO-LPM01A 与 STM32WB Nucleo 板的连接如下所示，完成连接后就可以通过上位机控制和测量 Nucleo 的功耗了。



低功耗模式

不同低功耗模式的区别主要如下：

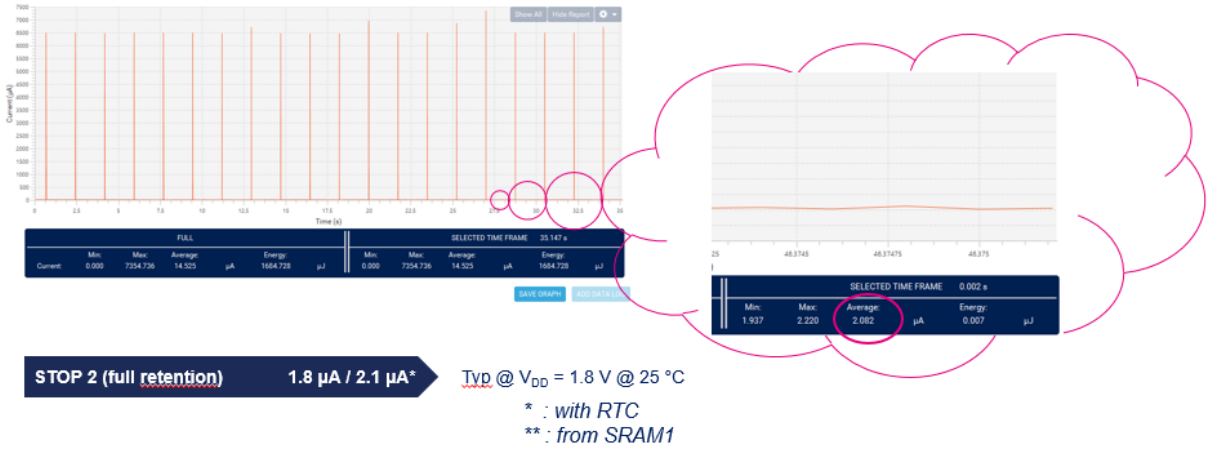
低功耗模式	唤醒源	唤醒后系统时钟	时钟影响
Sleep	中断/事件	与进入前一样	CPU 时钟关闭，对其他时钟或模拟时钟源无影响
LPRUN	清除 LPR 位	与进入前一样	无
LPSLEEP	中断/事件	与进入前一样	CPU 时钟关闭，对其他时钟或模拟时钟源无影响
Stop0/Stop1/Stop2	任意 EXTI， 特定外设事件	STOPWUCK=1， HSI16 STOPWUCK=0， MSI	所有时钟关闭，除了 LSI 和 LSE
Standby	WKUP 引脚,RTC 事件, LSECSS，NRST 引脚复位,IWDG 重置	HSI16	
Shutdown	WKUP 引脚,RTC 事件, NRST 引脚复位	MSI 4MHz	所有时钟关闭，除了 LSE

STM32WB 支持的低功耗模式非常多，首先需要根据应用情况选择一个适合的低功耗模式，能保持 BLE 连接的低功耗模式主要有 Sleep 和 STOP，所以选择 Sleep 模式和 STOP2 模式进行测量后结果如下：

Typ @ V_{DD} = 1.8 V @ 25 °C

* : *with RTC*

** : *from SRAM1*



Typ @ V_{DD} = 1.8 V @ 25 °C

* : *with RTC*

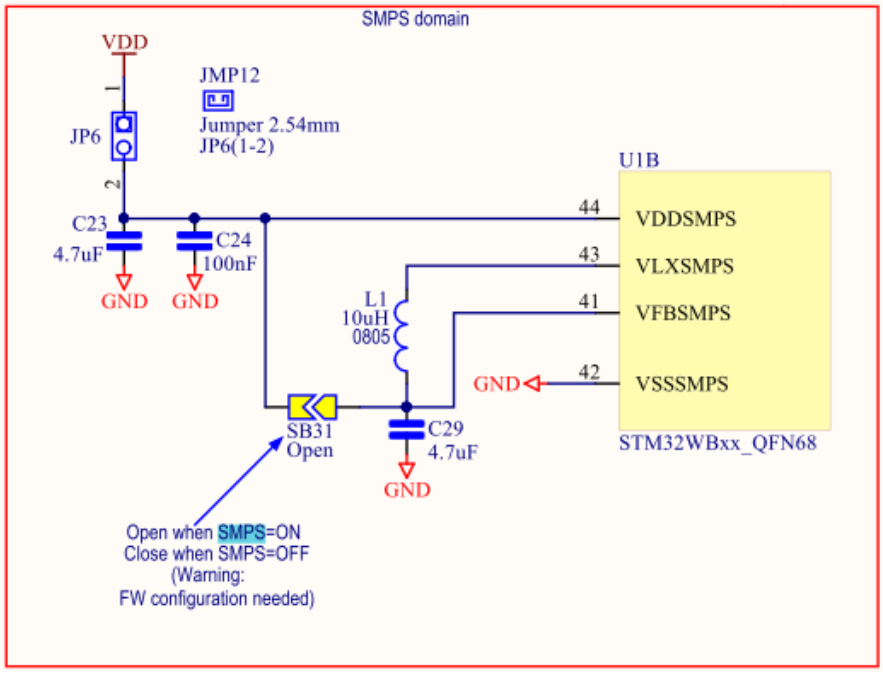
** : *from SRAM1*

图中红色脉冲代表的是设备的周期性广播，从测试结果来看 STOP2 模式比 Sleep 模式功耗低很多，所以应用在不在乎唤醒时间的情况下，应当尽可能进入 Stop2 模式。

SMPS

LDO 是一种线性电源，它的优势在于结构简单，电流纹波比较低，但电源效率偏低。

SMPS 是一种开关电源，它的优势在于电源效率高，损耗小，但由于频率较高会对周围设备造成一定的干扰，需要注意。



通过修改 Nucleo 板子上的 SB31 可以控制 LDO 还是 SMPS 供电：

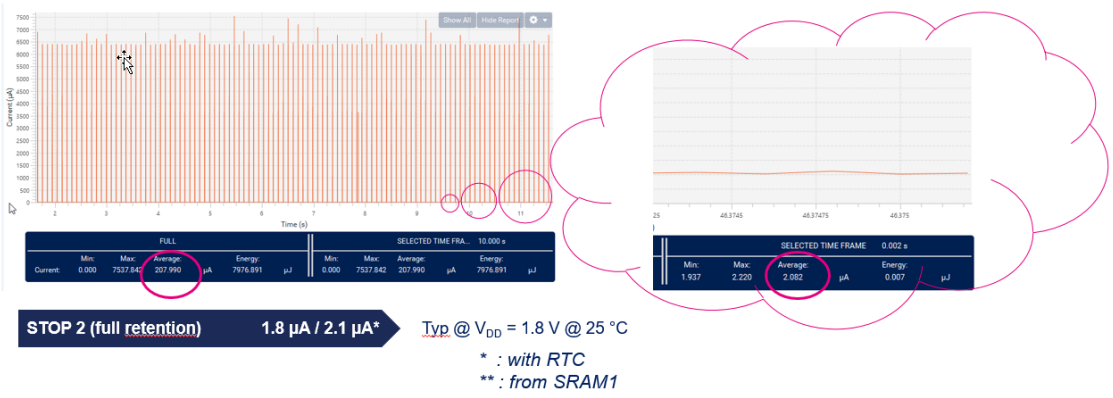
SB31 打开：SMPS 打开

SB31 关闭：SMPS 关闭

软件侧，打开 SMPS，修改 app_conf.h 代码如下：

```
#define CFG_USE_SMPS    1
```

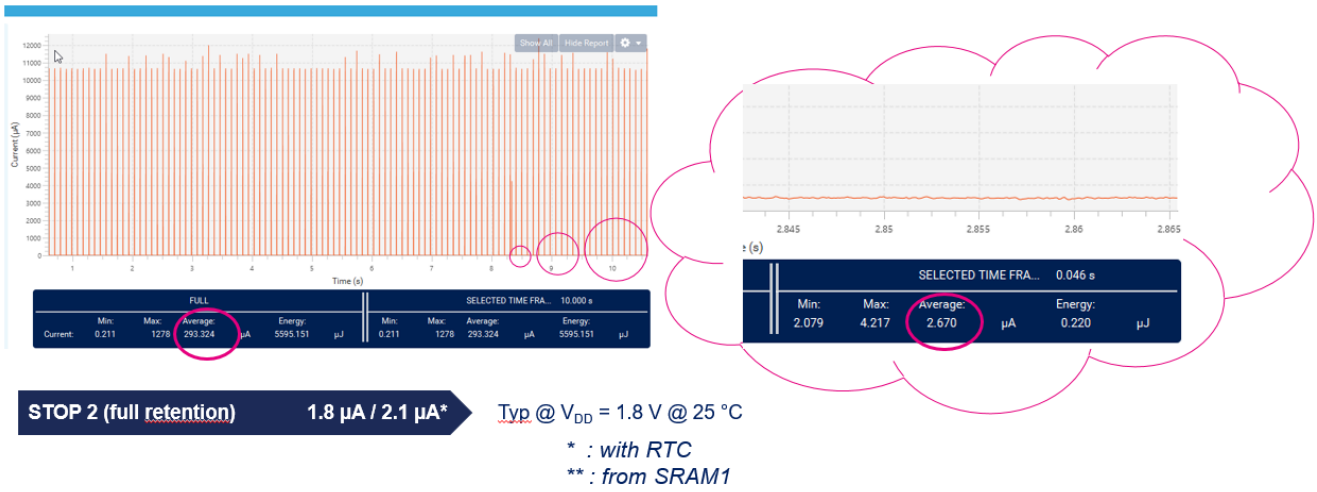
功耗测量结果如下：



关闭 SMPS，使用 LDO，修改 app_conf.h 代码：

```
#define CFG_USE_SMPS    0
```

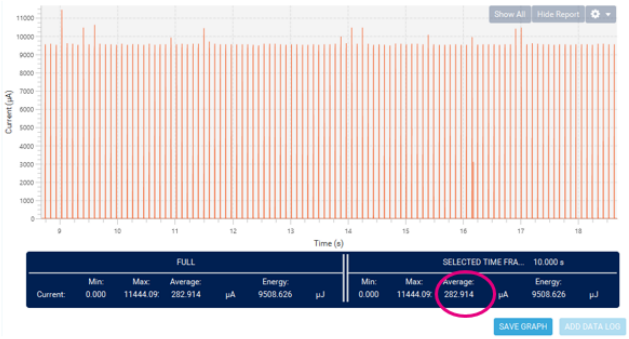
功耗测量结果如下：



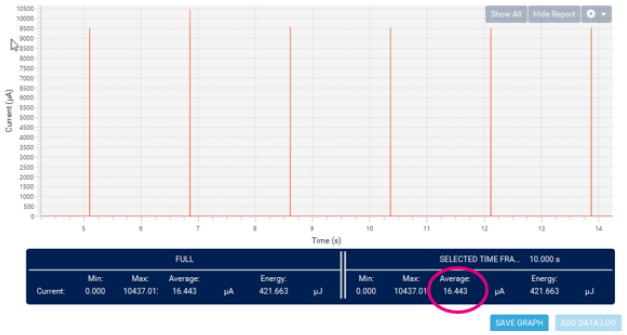
从以上图片可以看出，使用 SMPS 无论是平均电流还是低功耗状态下的电流都要更小，所以应当尽可能使用 SMPS。

广播参数

使用 BLE_HeartRate Demo 不需要做任何修改就可以测试不同的广播参数，该 demo 默认会先进行一段时间快速广播，然后再进入慢速广播。



广播间隔：
80ms~100ms



广播间隔：
1s~2.5s

从图中可以看出，广播间隔不同，平均功耗不同。

左侧图片的广播间隔为 80ms~100ms，平均电流为 282.914 uA。 右侧图片的广播间隔为 1s~2.5s，平均电流为 16.443 uA。

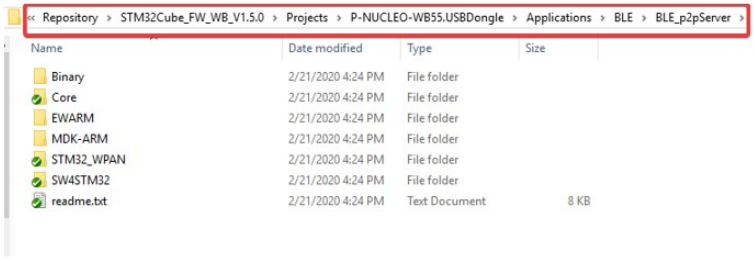
可以看出广播间隔的长短对功耗影响很大，所以在设计 BLE 应用的时候，应当考虑适当降低广播间隔。

为了保证尽快被对方设备发现，可以如 ST 心率 Demo 中的做法类似，首先先进入一段时间的快速广播，然后使用慢速广播，这样既可以保证开机时被发现的速度，又可以降低平均功耗。

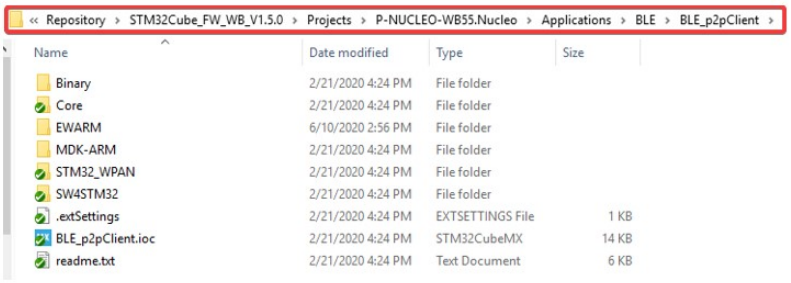
连接参数

测试修改连接参数，可以使用 ST 提供的 P2P Demo。注意：只有主机才能修改连接参数。

编译和烧录

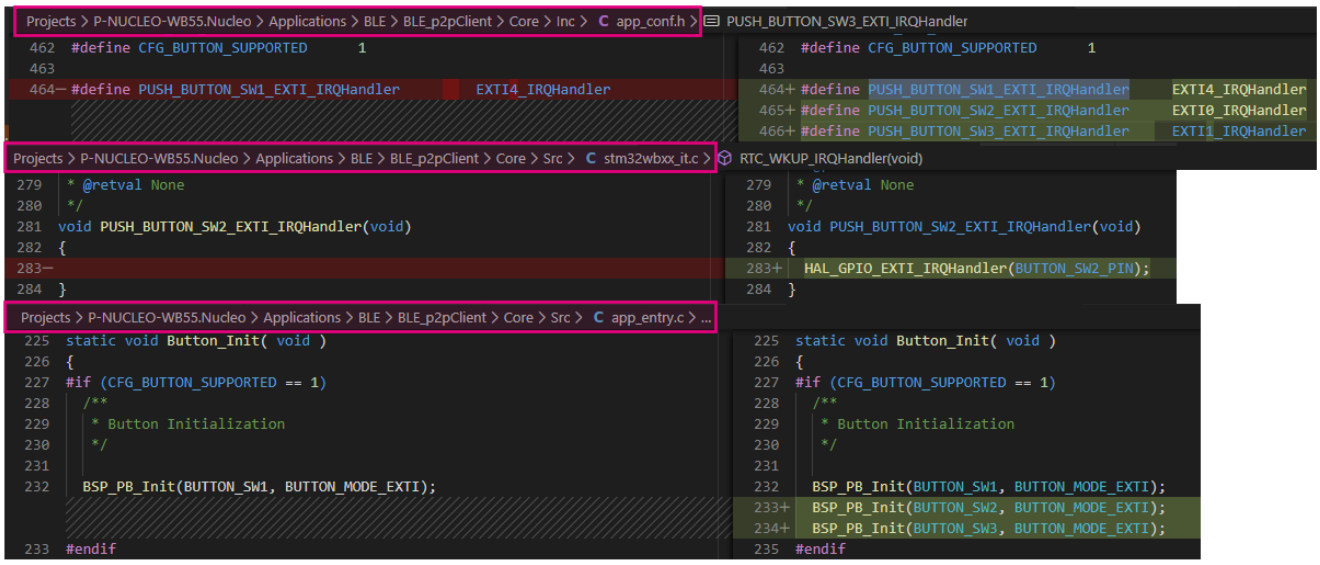


编译和烧录



通过 Button SW2 按键，切换并测试不同的连接参数，代码修改如下图：

打开 Button SW2 的中断：



关闭 trace 后，可以使能低功耗：

```
Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > Core > Inc > C app_conf.h > CFG_DEBUG_APP_TRACE
399   */
400- #define CFG_DEBUG_BLE_TRACE     1
401
402   /**
403    * Enable or Disable traces in applicatic
404    */
405- #define CFG_DEBUG_APP_TRACE     1
406

399   */
400+ #define CFG_DEBUG_BLE_TRACE     0
401
402   /**
403    * Enable or Disable traces in application
404    */
405+ #define CFG_DEBUG_APP_TRACE     0        You, a few seconds ago •
406
```

连接 dongle 后，按下 Button 后，切换不同的连接参数：

```
Projects > P-NUCLEO-WB55.Nucleo > Applications > BLE > BLE_p2pClient > STM32_WPAN > App > C app_ble.c > ...
657  void APP_BLE_Key_Button2_Action(void)
658  {
659  }

657  void APP_BLE_Key_Button2_Action(void)
658  {
659+   tBleStatus ret = BLE_STATUS_INVALID_PARAMS;
660+   static enum{
661+     CONN_FAST=0,
662+     CONN_SLOW=1
663+   }conn_mode;
664+   if(conn_mode == CONN_FAST){
665+     ret = aci_gap_start_connection_update(BleApplicationContext.BleApplicationContext_legacy.connectionHandle,
666+                                           0x64, 0x64, //Min&Max connect interval:125ms
667+                                           0x00, 0x0c80, //Connect latency:0, Supervision Timeout:32000ms
668+                                           0x00,0x64); //Min&Max CE Length:0~100*0.625ms
669+     conn_mode = CONN_SLOW;
670+   }else{
671+     ret = aci_gap_start_connection_update(BleApplicationContext.BleApplicationContext_legacy.connectionHandle,
672+                                           0x3E8, 0x3E8, //Min&Max connect interval:1250ms
673+                                           0x00, 0x0c80, //Connect latency:0, Supervision Timeout:32000ms
674+                                           0x00,0x64); //Min&Max CE Length:0~100*0.625ms
675+     conn_mode = CONN_FAST;
676+   }
677+   if (ret == BLE_STATUS_SUCCESS)
678+   {
679+     APP_DBG_MSG("Successfully aci_gap_start_connection_update()\n");
680+   }
681+   else
682+   {
683+     APP_DBG_MSG("aci_gap_start_connection_update() Failed , result: %d \n", ret);
684+   }
685  }
```

功耗测量结果如下：

连接间隔：
125ms

连接间隔：
1250ms

由于 BLE 在没有数据传输时，也会发送空包，所以降低连接间隔，可以降低平均功耗。

如图所示，左边是使用 125ms 连接间隔测试的结果，右边是使用 1250ms 连接间隔的测试结果，左边比右边的平均电流要高近 140uA。所以在应用中可以根据具体的应用使用合适的连接间隔来降低平均功耗。

总结

本文介绍了使用 STM32WB 设计 BLE 应用时，影响功耗的各方面的内容。低功耗设计的好坏，直接关系到产品的使用时长，往往是优秀产品的必备要素，需要认真掌握。

重要通知 – 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。