Technical Note

RSL10 FAQ

常见问题解答

类别	内容
关键词	
摘要	

修订历史

版本	日期	原因
V1.0.00	2020/04/24	创建文档
V1.0.01	2021/12/28	增加 FAQ

目 录

1. 常见问题

1.1 RSL10 最小系统调试

1. RSL10 EVB 如何烧录固件？

答：查看开发板背面一个用作 Jlink 的芯片是否有焊接，如果有，则直接用 micro USB 线连接 PC 和 RSL10 EVB，然后从 IDE 或使用 J-Flash 或官方的 Flashloader 软件下载固件。如果 RSL10 EVB 背面没有焊接 Jlink 芯片，那么需要使用 J-link 仿真器和 RSL10 EVB 连接进行固件烧录，SWD 接口接线如图 1 所示，然后使用 USB 线给 EVB 供电，J-link 仿真器和 RSL10 EVB 上标有 GND 丝印的引脚共地连接，这样就可以开始下载程序了。

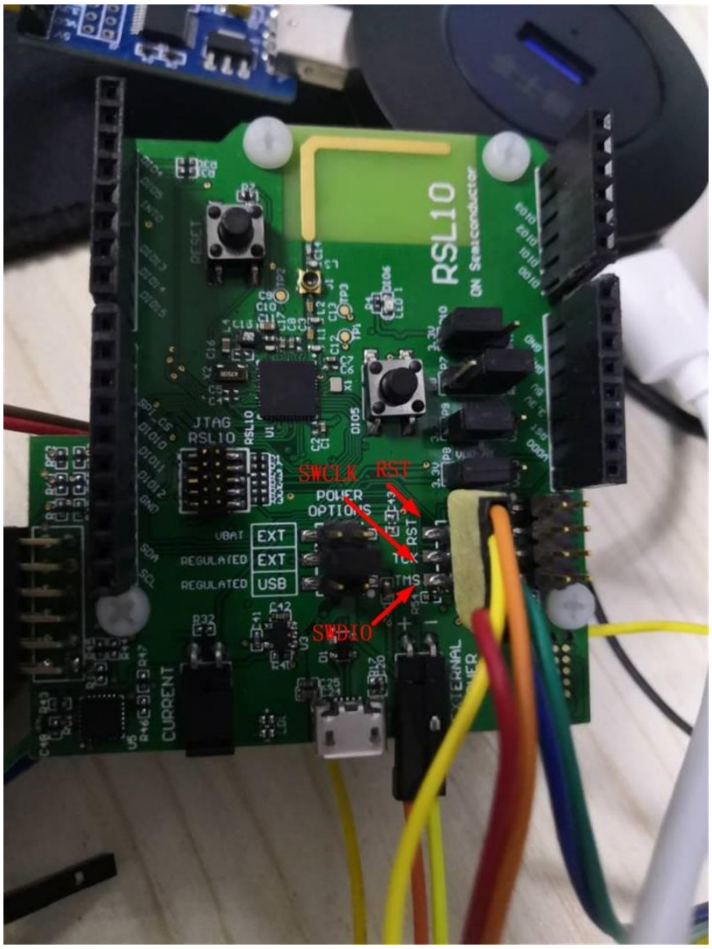


图 1 RSL10 EVB SWD 接口

2. 客户能够烧录固件到 RSL10 EVB 但却烧录不了固件到自己做的板子？

答：固件烧录是一个从 PC 到 J-link 仿真器再到 RSL10 芯片的数据传输过程，那么烧录不了固件，就可以分 PC、J-link 仿真器以及客户的 RSL10 板子三个部分进行问题排查。首先如果客户能够烧录固件到 RSL10 的 EVB，那么 PC 端方面 Jlink 的设置是设置正确的，Jlink 软件保持更新（v6.40 以上），，芯片型号也选择正确，那么接下来就是检查客户自己设计的板子和使用的 Jlink 仿真器。

客户板子的可以从 RSL10 最小系统原理图开始排查，先看原理图是否有按照官方给出的参考原理图来设计，原理图如图 2 所示，原理图上的网络标号是需要关注的，如果原理图

中引脚上有网络标号，需要注意是否该引脚被所连接的电路影响。两个晶振不需要外接电容电阻，直接和晶振引脚连接即可。晶振电路接电阻会影响固件下载。然后检查 SWD 接口有没有接上下拉电阻，一般 SWDIO 上拉，SWCLK 下拉。如果有不一样的，需改正。如果没错，那么检查板子供电是否正常，RSL10 VBAT 引脚是否有 3.3V 供电，VDDO 是否和 VBAT 电压相同。如果供电也正常，那么排查客户 RSL10 芯片及 48MHz 晶振等最小系统各部分元器件是否有虚焊问题，到这里客户板子 RSL10 最小系统部分电路就基本排查完毕了。如果最小系统的电路没问题，那么就剩下 Jlink 仿真器的问题，因为客户的 Jlink 基本都是在网上随便买的，网上 Jlink 仿真器的质量参差不齐，如果买到质量差的 Jlink，就很容易出现同个 Jlink 可以在 RSL10 EVB 可以下载程序但却没办法在客户板子上下载程序，原因就是大部分客户自己设计的板子可靠性没有 RSL10 EVB 那么高，而质量差的 Jlink 可靠性也差，所以就很容易导致无法在客户板子上烧录固件的问题，这时可以尝试更换几个 Jlink 试试以及 Jlink 的 USB 连接线以及和 SWD 接口的杜邦线来解决问题。

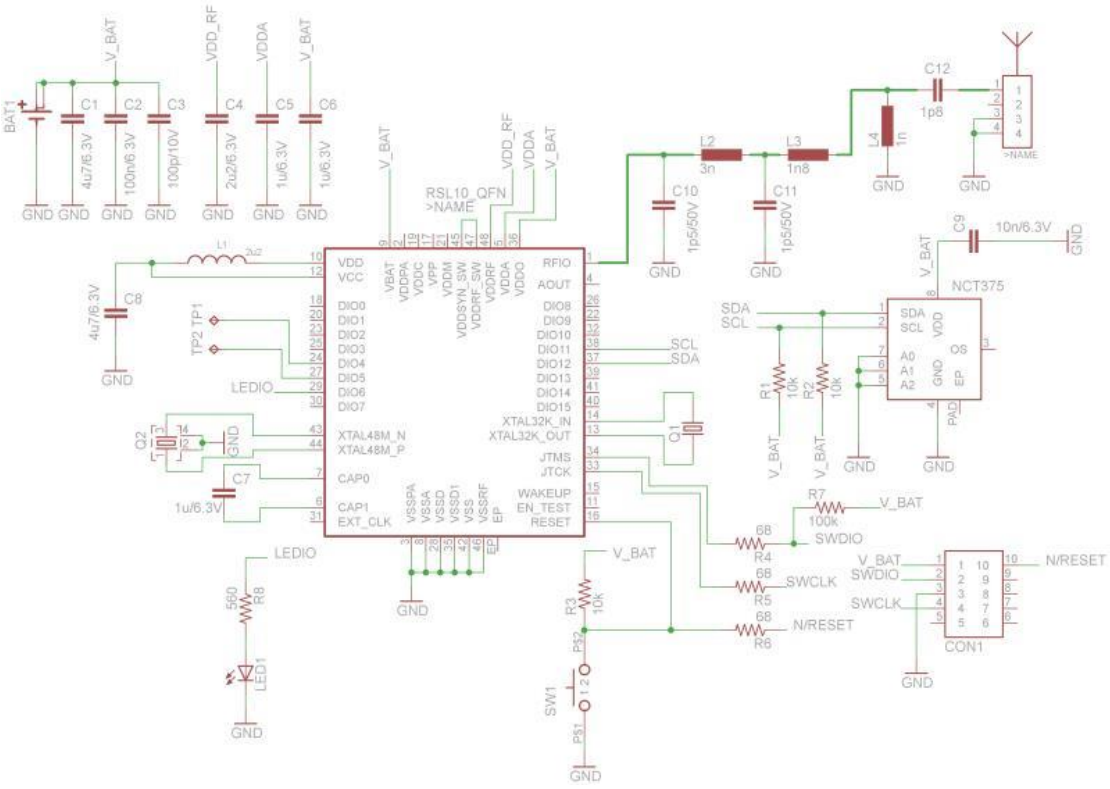


图 2　RSL10 最小系统原理图

3.　客户使用 RSL10 的睡眠模式，然后出现程序有时无法烧录问题？

答：RSL10 进入睡眠模式后会把 SWD 接口断开，这样会容易导致 SWD 接口无法控制芯片进行程序下载的问题，所以解决方法是需要客户在程序开头的初始化中保留和 SDK 一样的帮助固件烧录的程序段，如图 3 所示。然后把程序段中定义的“RECOVERY_DIO”对应的 DIO 引脚拉低后复位芯片，是芯片重新执行程序后停留在这段保护程序中，不执行其他功能，帮助下载程序。

```
    /* Test DIO12 to pause the program to make it easy to re-flash */
    DIO->CFG[RECOVERY_DIO] = DIO_MODE_INPUT | DIO_WEAK_PULL_UP |
                             DIO_LPF_DISABLE | DIO_6X_DRIVE;
    while (DIO_DATA->ALIAS[RECOVERY_DIO] == 0);
```

图 3　上电初始化时帮助烧录程序程序段

4. 客户板子下载 SDK 例程无法运行。

答：

1) 下载程序之后，程序不运行，需要看程序是否正确，如果程序有用到外部晶振，需要检查外部晶振电路是否正常，晶振是否有起振，如果晶振没起振，可能是晶振负载电容不匹配或者晶振有问题。

2) 查看客户板子 DIO12 引脚是否被拉低，因为大部分 SDK 例程都有把 DIO12 作为帮助程序烧录的引脚来使用，如果一直把 DIO12 拉低，在 RSL10 上电之后一检测到 DIO12 引脚一直为低电平，那么程序会一直停留在初始化函数中判别 DIO12 是否为低的 while 循环中，导致程序不会往下跑，这时需要电路把 DIO12 引脚不拉低或者修改程序把帮助程序烧录的 DIO12 引脚修改为其他 DIO 引脚。

5. 客户板子下载自己的程序无法运行。

答：下载程序之后，程序不运行，需要看程序是否正确，如果程序有用到外部晶振，需要检查外部晶振电路是否正常，晶振是否有起振，如果晶振没起振，可能是晶振负载电容不匹配或者晶振有问题。

1.2 RSL10 外设相关问题

1.2.1 DIO 使用问题

1. 为什么配置了 DIO13-DIO15 引脚做 SPI、GPIO 等用途时没有起作用？

答：DIO13-DIO15 默认是为 JTAG 调试接口，所以在配置这三个引脚为其他用途之前，需要在软件中先把 JTAG 功能关掉，如图 4 所示。

```
    /* Disable JTAG data on DIO14 and DIO15 */
    DIO_JTAG_SW_PAD_CFG->CM3_JTAG_DATA_EN_ALIAS = 0;
    /* Disable JTAG test on DIO13*/
    DIO_JTAG_SW_PAD_CFG->CM3_JTAG_TRST_EN_ALIAS = 0;
```

图 4　关闭 DIO13-DIO15 的默认 JTAG 功能

2. DIO_CFG 寄存器 DIO_MODE_GPIO_IN_0 和 DIO_MODE_GPIO_IN_1 的区别是什么？

答：DIO_MODE_GPIO_IN_0 和 DIO_MODE_GPIO_IN_1 这两个值都是用于设置 RSL10 的 DIO 为输入模式，由于当 DIO 配置为 GPIO 输入模式时，GPIO 引脚的电平可以通过 DIO_DATA 寄存器读取，也可以用 DIO_CFG 寄存器的第 0 位进行读取，因此才有 DIO_MODE_GPIO_IN_0 和 DIO_MODE_GPIO_IN_1 两个值用于在 DIO_CFG 寄存器标记 DIO 引脚的电平值，可使用这两个宏定义值与读取的引脚电平值进行比较。

3. 芯片启动时 DIO0-DIO15 引脚的默认电平是什么？

答：在芯片上电复位启动时，DIO0-DIO15 引脚的 DIO_CFG 寄存器配置都是默认内部

弱上拉的，IO Disable 模式，所以 DIO 引脚启动时默认都是高电平，如果在芯片启动进入 main 函数就把引脚初始化为下拉输入或者输出低电平时，至少需要在芯片启动约 3.2ms 之后生效（即芯片启动后 DIO 引脚由高电平到低电平需要约 3.2ms）

1.2.2 串口使用问题

1. 芯片串口寄存器是否有发送完成、接收完成标志位可以查询？

答：RSL10 芯片的串口做的十分的简单，UART 寄存器没有发送完成和接收完成的标志位，即无法通过寄存器的标志位进行串口的轮询发送和接收，但是 UART 的发送完成或接收到数据时可以触发 UART 中断，因此如果要使用轮询的方式进行串口收发，可以通过 UART 中断来软件模拟个标志位变量，轮询软件的标志位变量进行串口的轮询收发。

2. 如何在 freertos_ble_peripheral_server_bond 例程上增加 uart_cmsis_driver 的功能？

答：在 IDE 中打开 freertos_ble_peripheral_server_bond 例程的.retconfig 文件，然后如所示勾选 CMSIS Driver 的 USART 选项和 Device->Libraries 的 DMA 及 GPIO 选项，勾选后按“ctrl+s”对.retconfig 文件进行保存。

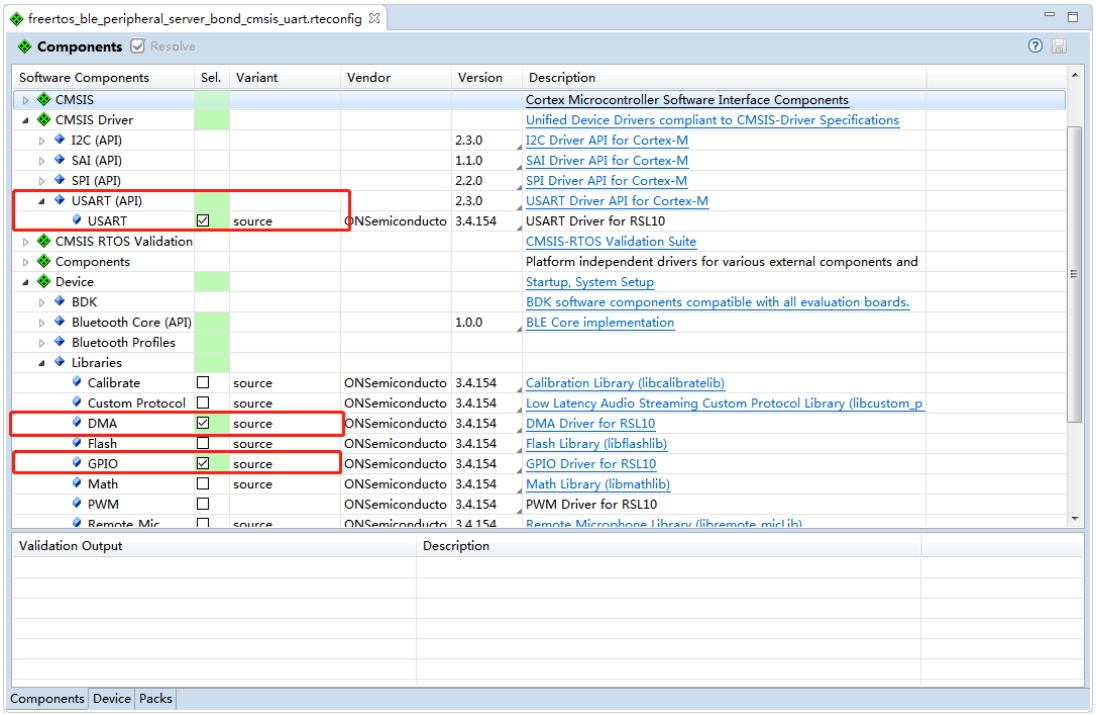


图 5 勾选 uart_cmsis_driver 的相关选项

然后打开位于“RTE/Device/RSL10”路径下的 RTE_Device.h 文件，分别将宏定义 RTE_DMA、RTE_GPIO 和 RTE_USART 的值都设置为 1，如图 6、图 7 和图 8 所示。至此 UART_CMSIS_Driver 就可以使用了。

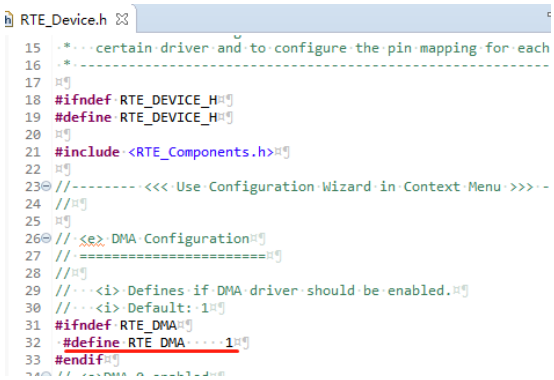


图 6　设置 RTE_DMA 宏为 1

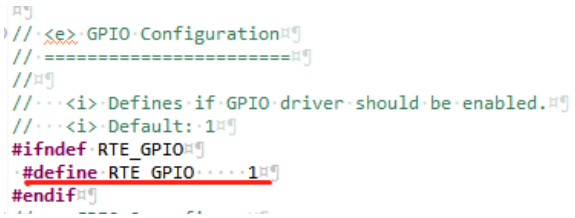


图 7　设置 RTE_GPIO 宏定义为 1

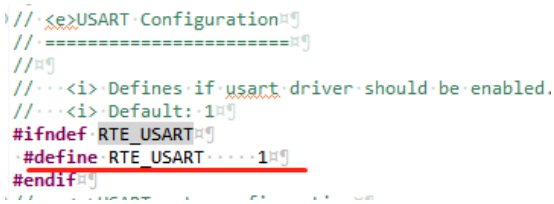


图 8　设置 RTE_USART 宏定义为 1

当使用 UART_CMSIS_Driver 时，需要先定义一个变量（这里命名为“p_uart_env”）获取 UART_CMSIS_Driver 的环境变量的地址“&Driver_USART0”，然后进行 UART_CMSIS_Driver 的初始化，如图 9 所示。

```
/* Driver_USART0 has defined in USART_RSLxx.c */
extern ARM_DRIVER_USART Driver_USART0;

/* Pointer of the uart cmsis driver */
static ARM_DRIVER_USART *p_uart_env;

void app_uart_init (void)
{
    /* Initialize usart driver structure */
    p_uart_env = &Driver_USART0;

    /* Initialize usart, register callback function */
    p_uart_env->Initialize(Usart_EventCallBack);
}
```

图 9　初始化 UART_CMSIS_Driver

初始化完成之后要使用 UART_CMSIS_Driver 收发数据，则调用 p_uart_env 的 Send 函数和 Receive 函数或者 Transfer 函数即可。

如需修改默认的波特率、串口引脚等配置信息，在 RTE_Device.h 文件的 USART 部分进行修改，如图 10 所示。

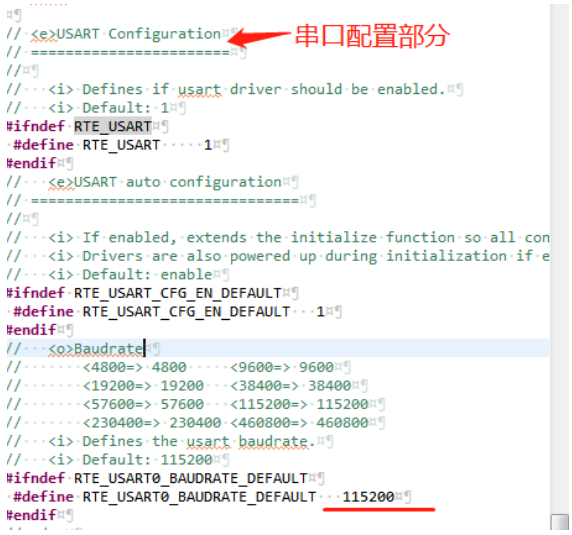


图 10　修改 UART_CMSIS_Driver 配置

1.2.3 DMIC 使用问题

1. 芯片的数字麦克风接口数量是多少？

答：DMIC 麦克风对外接口只有一个，但是 DMIC 外设有两个(DMIC0 和 DMIC1)，通过对外的一个 DMIC 接口接上两个数字麦克风（两个麦克风的 DATA 和 CLK 线分别短接，一个麦克风选择时钟上升沿输出数据，一个麦克风选择时钟下降沿输出数据），利用 DMIC0 和 DMIC1 分别在时钟的上升沿和下降沿对接口收到的数字麦克风进行采集，达到双麦拾音的效果。

1.2.4 ADC 外设

1. ADC 外设有多少个通道？

答：RSL10 芯片 ADC 是差分输入的，共有 8 个通道（通道 0~通道 7），每个通道都可以选择正极和负极的输入源，输入源如图 11 所示。

11.2.1 ADC Input Configuration

The purpose of the ADCs is to sample analog signals that are relevant to the user's application use cases—for example, the voltage associated with a potentiometer-based volume control, or a supply voltage for battery monitoring applications using the Bluetooth low energy battery service (BAS).

The signals measured by the ADC block are configured using the NEG_INPUT_SEL and POS_INPUT_SEL bit-fields from the ADC_INPUT_SEL register set. The negative and positive signals used for each differential measurement are selected from:

DIOs 0, 1, 2, 3	To use ADC functionality, the DIO must be configured as follows: 1. Set the IO_MODE field of the DIO_CFG register to 0x3F. 2. Set the PULL_CTRL field of the DIO_CFG register to 0x0. For more information about DIO configuration for ADCs, see Chapter 10, "Digital Input/ Output" on page 259.
AOUT	The analog test output signal. The signal provided to AOUT can be configured using the ACS_AOUT_CTRL_TEST_AOUT bit field from the ACS_AOUT_CTRL register, which allows the ADCs to additionally measure a number of other internal power supply outputs and status flags. For more information about the internal power supplies, see Section 5.3, "Internal Power Supply Voltages" on page 39.
VDDC	For more information about VDDC and its configuration, see Section 5.3.4, "Digital Supply Voltages" on page 46.
VBAT/2	A divided form of the battery supply voltage, measured through a fixed resistive divider which ensures that the measured value is in the expected range of the ADC for accurate measurement. To save power, the resistive divider can be configured to only be enabled when a conversion is taking place by setting the ADC_CFG_DUTY_DIVIDER bit-field from the ADC_CFG register to ADC_VBAT_DIV2_DUTY. For additional information about VBAT, see Section 5.2.1, "Battery Supply Voltage (VBAT)" on page 38.
GND	Ground (VSS) supply reference.

图 11　RSL10 ADC 外设可配置的输入源

通过以下配置就可以使 ADC_CHANNEL 指定的通道采集 VBAT/2 的功能，无需外接引脚连接电源时序 ADC 检测电压。

```
/* Configure ADC_CHANNEL input selection to VBAT/2 */
    Sys_ADC_InputSelectConfig(ADC_CHANNEL, ADC_POS_INPUT_VBAT_DIV2
|ADC_NEG_INPUT_GND);
```

ADC 外设有可选开启供电电源监控的功能（Power Supply Monitoring），电源监控功能固定使用 ADC 的通道 6 监控 VBAT/2 的电压值，固定使用 ADC 的通道 7 监控 VCC 的电压值。如果将 ADC 通道 6 配置为采集 VBAT/2 的电压值，开启 ADC 通道 6 采集中断，并且启用 ADC 电源监控功能监控 VBAT/2 的电压，开启电源监控中断，那么通道 6 可以触发 ADC 采集中断获取 VBAT/2 电压值和电源监控中断进行低电压报警。如果 ADC 通道 6 没有进行 ADC 配置，只配置 ADC 开启电源监控功能，那么通道 6 将只触发电源监控中断。ADC 各通道默认的正极输入源是 VBAT/2，负极输入源是 GND。

注：如果在 BLE 例程里面使用 ADC 外设，需要在 Sys_RFFE_SetTXPower 函数之后再进行应用程序的 ADC 初始化，因为 Sys_RFFE_SetTXPower 函数里面有使用 ADC 外设，所以如果在 Sys_RFFE_SetTXPower 函数之前就初始化 ADC 外设，Sys_RFFE_SetTXPower 函数里面会重新初始化一遍 ADC 外设，这样应用程序的 ADC 初始就无效了。

ADC 采集模式：

1、　Normal 模式：ADC 会依次在 8 个通道进行数据采集，每第八次采集产生中断（即 8 个通道轮询一次采集就产生一次中断），可以通过

ADC_BATMON_INT_ENABLE 寄存器指定是哪个通道采集完触发 ADC 中断。

2、 Continuous 模式：ADC 只对一个通道进行数据采集，ADC 中断在每次中断采集完成时触发，同样可以通过 ADC_BATMON_INT_ENABLE 寄存器指定是哪个通道采集完触发 ADC 中断。

ADC 值的计算：

在 Low-frequency mode 时，ADC 的采样结果都是 14-bits 的分辨率；在 High-Frequency mode 时，ADC 在采集速率小于 25kHz 的情况下，采样结果有 14-bits 的分辨率，ADC 在采样速率为 50kHz 的情况下，采样结果只有 8-bits 的分辨率，采样的值从 ADC_DATA_TRIM_CH*寄存器取出，该寄存器提供 14 位宽的无符号数值（0x000~0x3FFF）来代表电压值从 0-2.0V，超过这个范围的电压值都视为饱和。

ADC 外设在不同的分频系数下有不同的固有电压范围值，14 位的采样值将根据这些固有电压范围进行计算，固有电压范围如图 12 所示：

Table 22. ADC Sample Rate Configuration

Value	Setting	Fixed Divider	Configurable Division	Native Voltage Range (V)	Effective Division of SLOWCLK (Normal Mode)	Effective Division of SLOWCLK (Continuous Mode)
0x0	ADC_DISABLE	N/A	ADC Disabled	N/A	ADC Disabled	ADC Disabled
0x1	ADC_PRESCALE_200	10	Divide by 20	-0.125 to 2.125	Divide by 1600	Divide by 200
0x2	ADC_PRESCALE_400	10	Divide by 40	-0.063 to 2.063	Divide by 3200	Divide by 400
0x3	ADC_PRESCALE_640	10	Divide by 64	-0.031 to 2.031	Divide by 5120	Divide by 640
0x4	ADC_PRESCALE_800	10	Divide by 80	-0.016 to 2.016	Divide by 6400	Divide by 800
0x5	ADC_PRESCALE_1600	10	Divide by 160	-0.008 to 2.008	Divide by 12800	Divide by 1600
0x6	ADC_PRESCALE_3200	10	Divide by 320	-0.004 to 2.004	Divide by 25600	Divide by 3200
0x7	ADC_PRESCALE_6400	10	Divide by 640	-0.002 to 2.002	Divide by 51200	Divide by 6400
0x8	ADC_PRESCALE_20H	2	Divide by 10	-1.0 to 3.0	Divide by 160	Divide by 20
0x9	ADC_PRESCALE_40H	2	Divide by 20	-0.125 to 2.125	Divide by 320	Divide by 40
0xA	ADC_PRESCALE_80H	2	Divide by 40	-0.063 to 2.063	Divide by 512	Divide by 64
0xB	ADC_PRESCALE_128H	2	Divide by 64	-0.031 to 2.031	Divide by 640	Divide by 80
0xC	ADC_PRESCALE_160H	2	Divide by 80	-0.016 to 2.016	Divide by 1280	Divide by 160
0xD	ADC_PRESCALE_320H	2	Divide by 160	-0.008 to 2.008	Divide by 2560	Divide by 320
0xE	ADC_PRESCALE_640H	2	Divide by 320	-0.004 to 2.004	Divide by 5120	Divide by 640
0xF	ADC_PRESCALE_1280H	2	Divide by 640	-0.002 to 2.002	Divide by 10240	Divide by 1280

NOTE: For a typical SLOWCLK frequency of 1.00 MHz, the ADC samples all eight channels in Normal mode at a configurable rate between 6.25 kHz and 19.5 Hz.

图 12 RSL10 芯片 ADC 外设的采样配置

如当 ADC 的频率使用 ADC_PRESCALE_200，则固有电压范围是-0.125~2.125V，该频率下 ADC 是使用 14 位分辨率，因此 ADC_DATA 寄存器的值每增加 1，就代表增加了(2.125-(-0.125))/2^{14} = 0.0001373291015625V，所以如果 ADC 采集的电压值是 0V，那么寄存器取出的采样值应为 0.125/ 0.0001373291015625= 910.222 ≈ 910。

1.2.5 RTC 外设

1. RTC 外设及其中断如何使用？

答：RTC 外设是向下计数器，计数值到达 0 之后会自动从设置的起始值重新开始计数，使用 RTC 外设前需要先启动 32K 时钟，目前测试 RTC 外设只能正常触发

RTC_CLOCK_IRQHandler 中断，没办法正常使用 RTC_ALARM_IRQHandler 中断，使用 RTC_CLOCK_IRQHandler 中断时，需要将 RTC 的开始值设置为大于等于 32768 的值，最大是 32bit 全为 1（0xFFFFFFFF），RTC_CLOCK_IRQHandler 中断是固定 1s 触发一次中断的，只有当 RTC 的开始值设置为大于等于 32768 的值才能准确的 1s 触发一次。

1.3 频偏问题

1. 48MHz 晶振频偏调节

答：RSL10 晶振电路不需要外加负载电容，因为芯片内部负载电容可调，如果射频频偏过大，可以通过调节 48MHz 晶振端芯片内部负载电容值来实现频偏的改善，如图 13 所示。

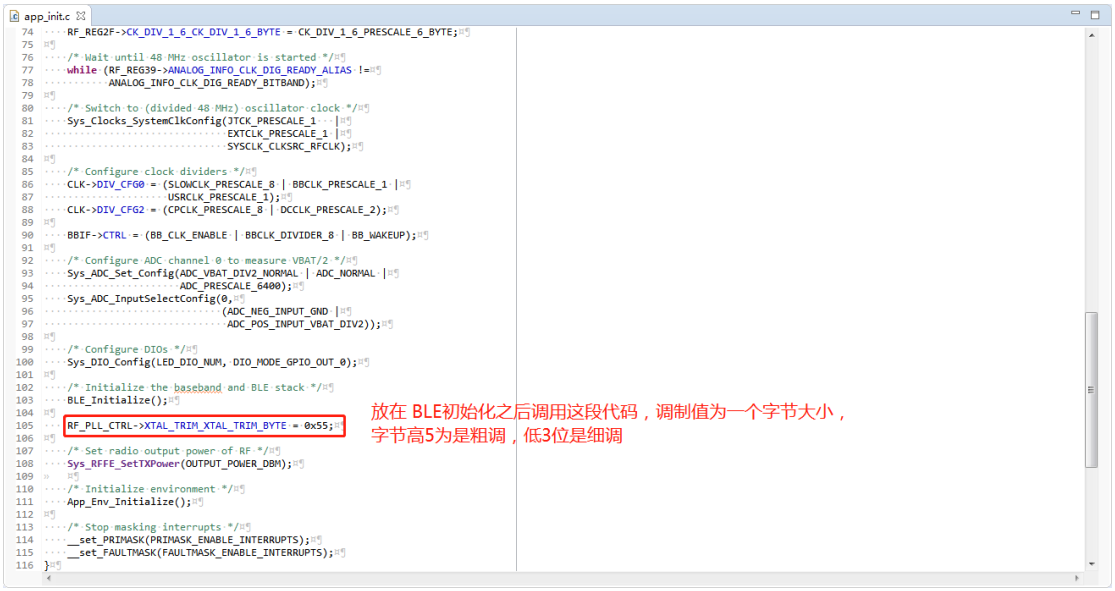


图 13 调整 RSL10 芯片 48MHz 晶振负载电容的代码

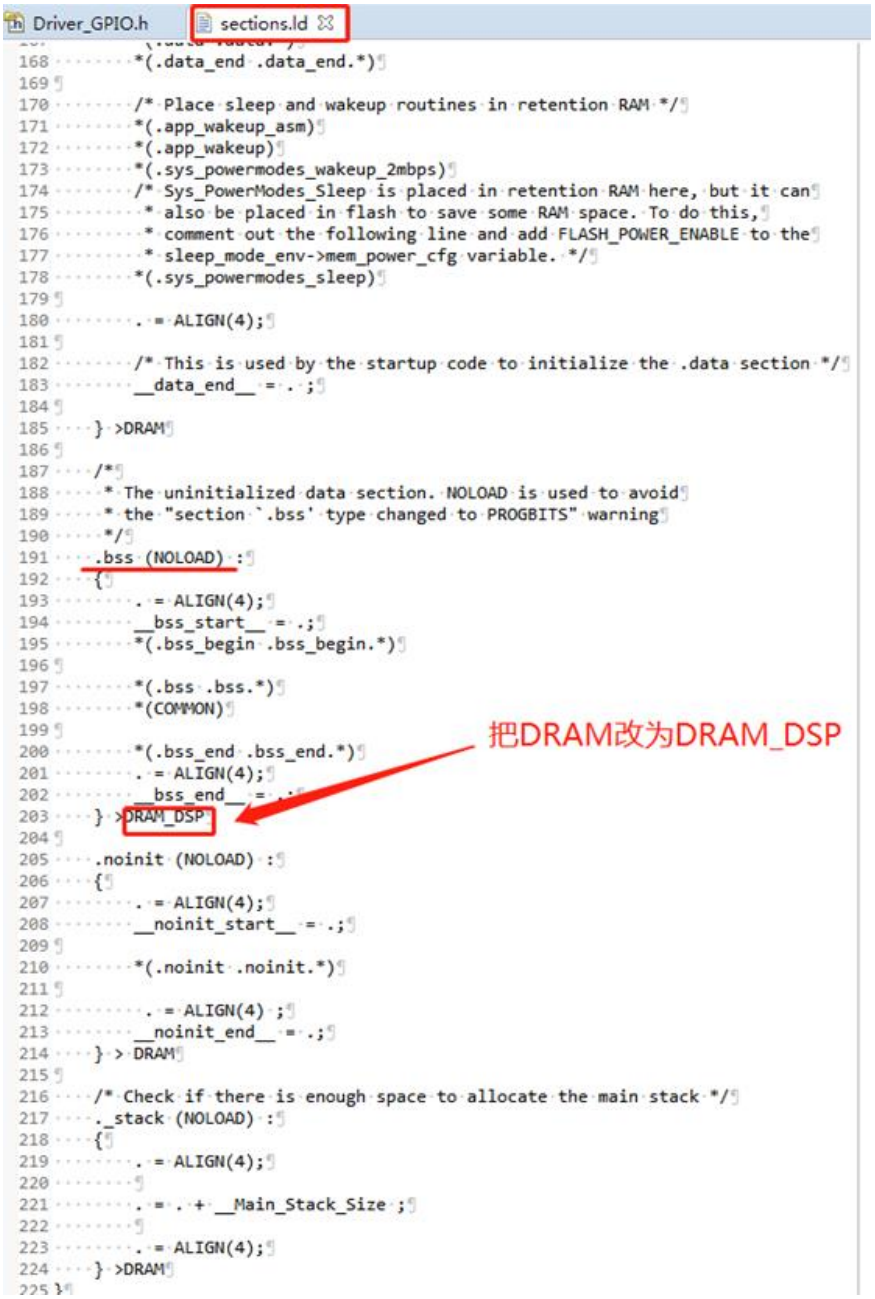
1.4 链接文件修改

1.4.1 全局变量移至 RAM_DSP 存放

由于 RSL10 芯片 DRAM 是 24KB 大小，如果用户程序的全局变量较多，DRAM 区域方不小，可以考虑将全局变量转移至 RAM_DSP 区域存放，RAM_DSP 有 48KB 大小。

1. 如何将全局变量 bss 段数据放到 DSP_RAM

答：找到工程使用的 ld 文件的对应路径，然后在该路径下打开 ld 文件进行修改，改好之后把工程编译生成的 debug 文件夹删除，然后再重新进行编译。修改 ld 文件的部分如图 14 所示，将 ld 文件里面的 bss 段由 DRAM 改为 DRAM_DSP。




```
        *(.data_end .data_end.*)

        /* Place sleep and wakeup routines in retention RAM */
        *(.app_wakeup_asm)
        *(.app_wakeup)
        *(.sys_powermodes_wakeup_2mbps)
        /* Sys_PowerModes_Sleep is placed in retention RAM here, but it can
         * also be placed in flash to save some RAM space. To do this,
         * comment out the following line and add FLASH_POWER_ENABLE to the
         * sleep_mode_env->mem_power_cfg variable. */
        *(.sys_powermodes_sleep)

        . = ALIGN(4);

        /* This is used by the startup code to initialize the .data section */
        __data_end__ = .;

    } >DRAM

    /*
     * The uninitialized data section. NOLOAD is used to avoid
     * the "section `.bss' type changed to PROGBITS" warning
     */
    .bss (NOLOAD) :
    {
        . = ALIGN(4);
        __bss_start__ = .;
        *(.bss_begin .bss_begin.*)

        *(.bss .bss.*)
        *(COMMON)

        *(.bss_end .bss_end.*)
        . = ALIGN(4);
        __bss_end__ = .;
    } >DRAM_DSP

    .noinit (NOLOAD) :
    {
        . = ALIGN(4);
        __noinit_start__ = .;

        *(.noinit .noinit.*)

        . = ALIGN(4) ;
        __noinit_end__ = .;
    } > DRAM

    /* Check if there is enough space to allocate the main stack */
    ._stack (NOLOAD) :
    {
        . = ALIGN(4);

        . = . + __Main_Stack_Size ;

        . = ALIGN(4);
    } >DRAM
}
```

图 14　修改链接文件的 bss 段数据存放到 DRAM_DSP 区域

1.5　ON IDE 相关问题

1.5.1 Debug 查看外设寄存器

1.　如何在 ON IDE 进入 debug 模式时查看 RSL10 芯片的外设情况？

答：在 debug 配置的时候，点击 SVD Path 一栏，按如图 15 所示的说明把路径给输入进去。设置完毕后点击“Debug”即可查看外设寄存器。

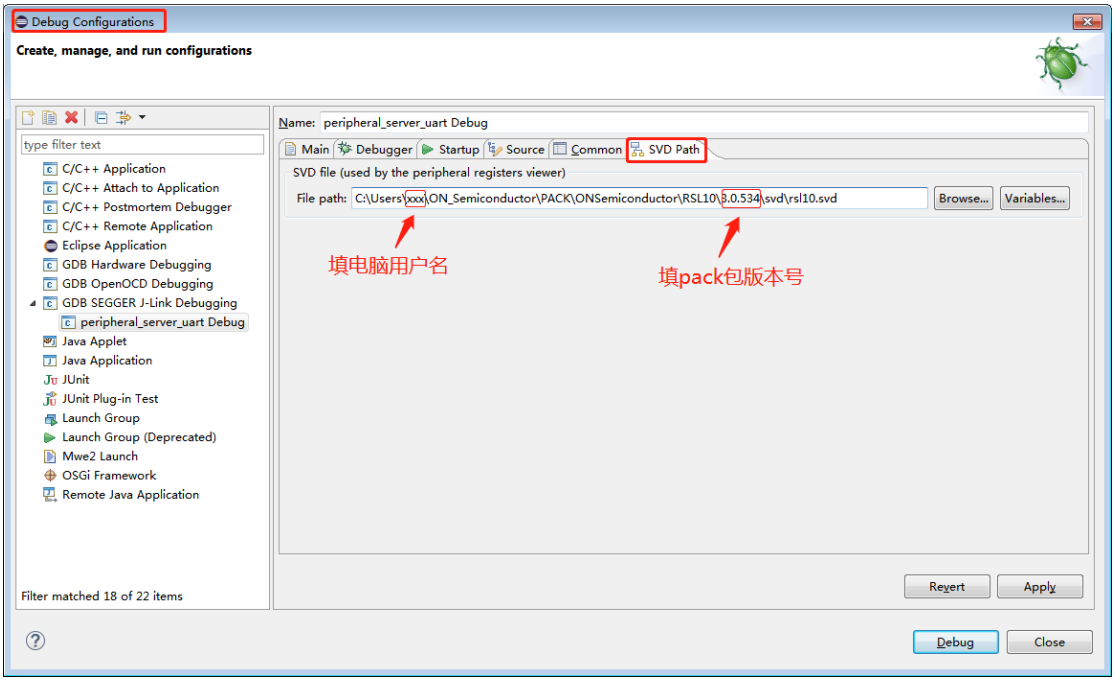


图 15　debug 配置项增加 SVD 设置，用于查看外设寄存器

1.5.2 Printf 输出浮点数设置

如图 16 所示，在 IDE 的对应工程设置中，打开“use float with nano printf”选项，之后将工程存放编译生成文件的“Debug”文件夹删除，再重新编译一下工程，编译后的固件对应的 printf 就可以输出浮点数了。

注：目前该功能在 fota 相关工程中无法使用。

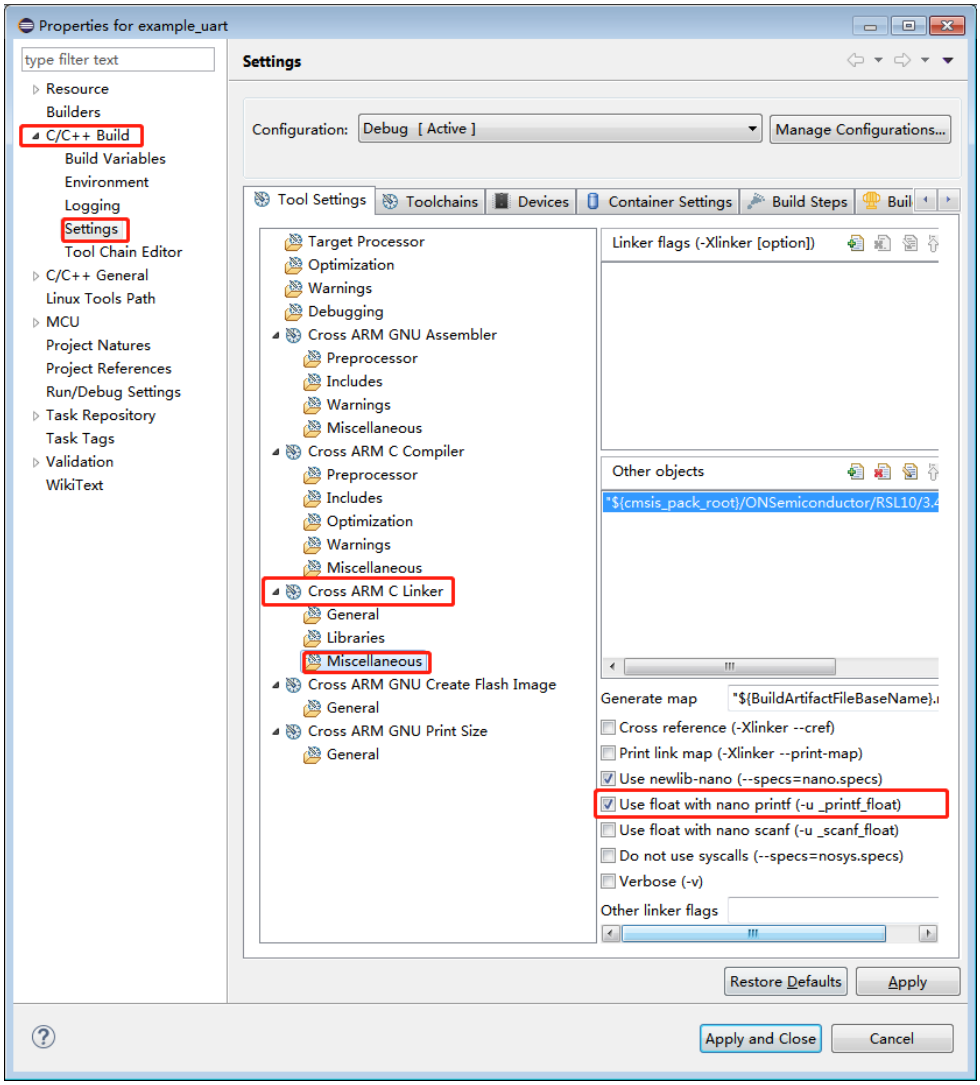


图 16　工程设置中勾选浮点数输出

1.5.3 添加头文件包含路径

如图 17 所示，在 IDE 的工程浏览框中鼠标右键点击要添加头文件的软件工程，然后选择“Properties”，然后在如图 18 所示位置进行头文件的添加。

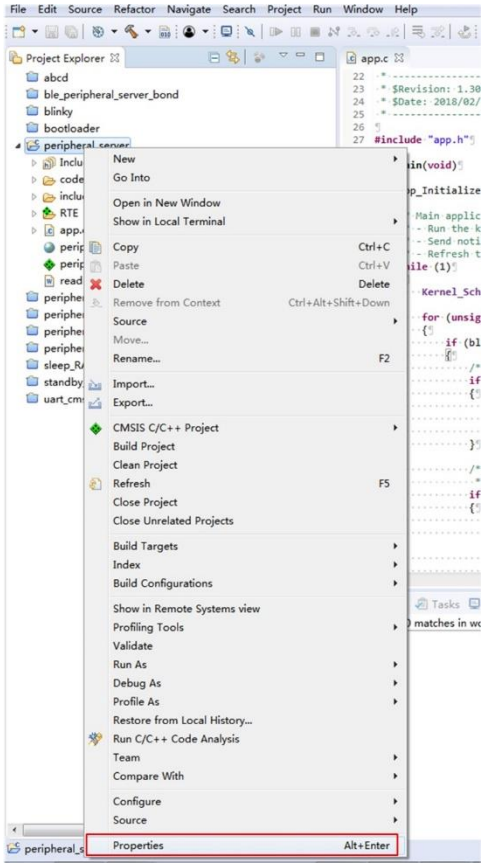


图 17　右键工程选择“Properties”

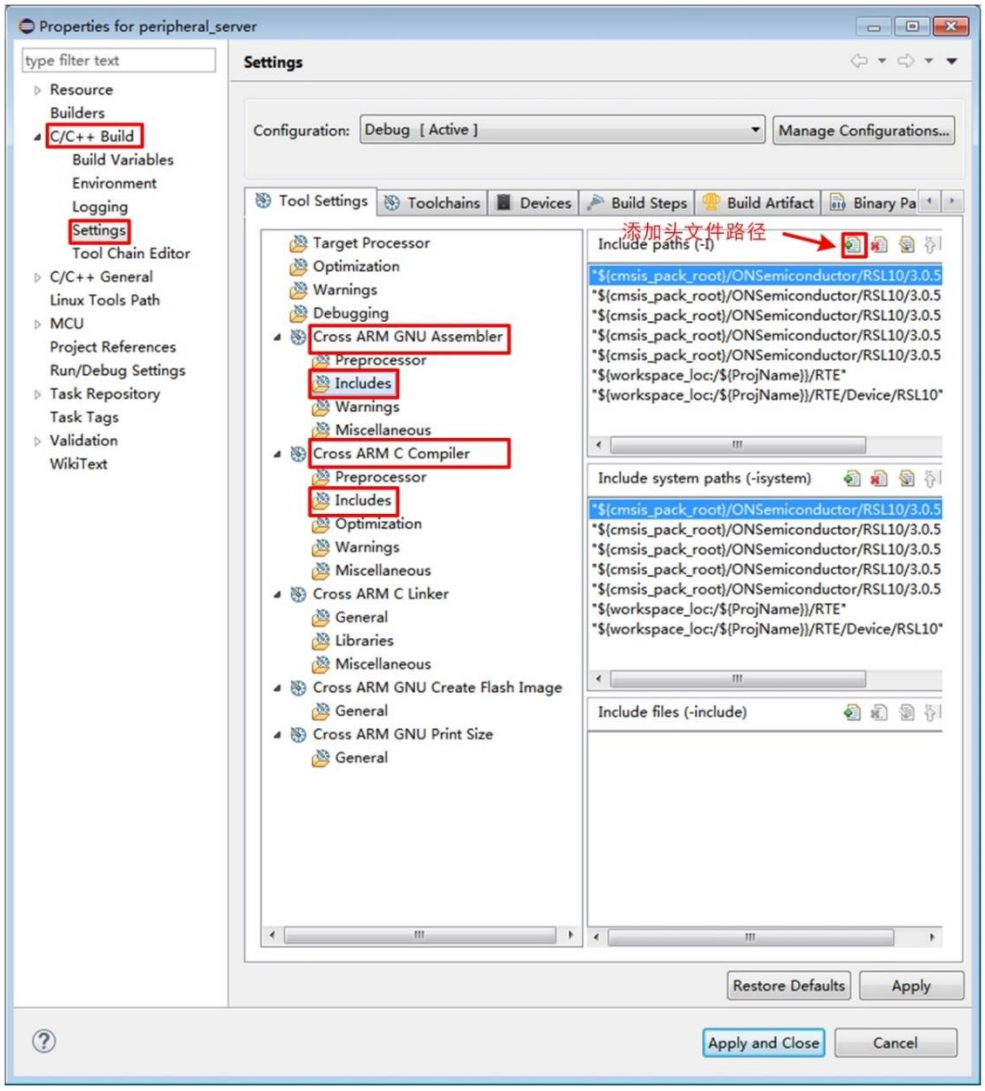
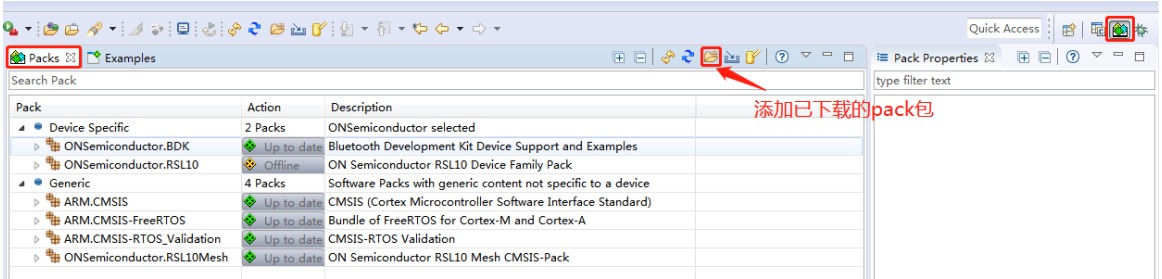


图 18　添加头文件包含路径

1.5.4 添加 PACK 包

如所示，在 IDE 中打开 CMSIS Pack Manager 按钮，然后在 pack 栏中点击添加 pack 包按钮然后浏览到已下载的 pack 路径下，将 pack 包添加安装到 IDE 中。



注：使用带 FreeRTOS 功能的例程时，如 freertos_ble_peripheral_server_bond、freertos_ble_central_client_bond 等，除了需要添加 RSL10 芯片的 pack 包外，还需要再安装 FreeRTOS 的 pack 包（FreeRTOS 的 pack 需要到 Keil 官网下载）。RSL10 芯片 pack 包和 FreeRTOS 的 pack 包版本对应关系如下：

1. ONSemiconductor.RSL10.3.4.154.pack ---------- ARM.CMSIS-FreeRTOS.10.2.0.pack

1.5.5 IDE 重新安装之后找不到 CMSIS PACK 插件

原因：旧 IDE 卸载之后没有把原来的安装路径的相关文件删除干净。

解决方法：卸载 IDE 后把原来的安装路径的相关文件和文件夹删除干净，再重新安装一次 IDE 即可（安装过程最好关闭拦截软件，防止拦截软件误删文件）。

1.6 蓝牙功能相关问题

1.6.1 设备名称

1. 如何修改蓝牙设备名称为中文名？

答：首先需要将中文专为 URL 编码格式，然后将转换成 URL 编码的名称值放到一个数组中，转换出来有多少个字节，广播名称长度就得对应多少个字节。

URL 编码工具可在网上搜索，这里放一个 URL 编码格式转换的参考网址：

https://tool.oschina.net/encode?type=4

比如现在想把设备名称设置为“你好”（不包含双引号），则到网址进行 URL 转换格式如图 19 所示。

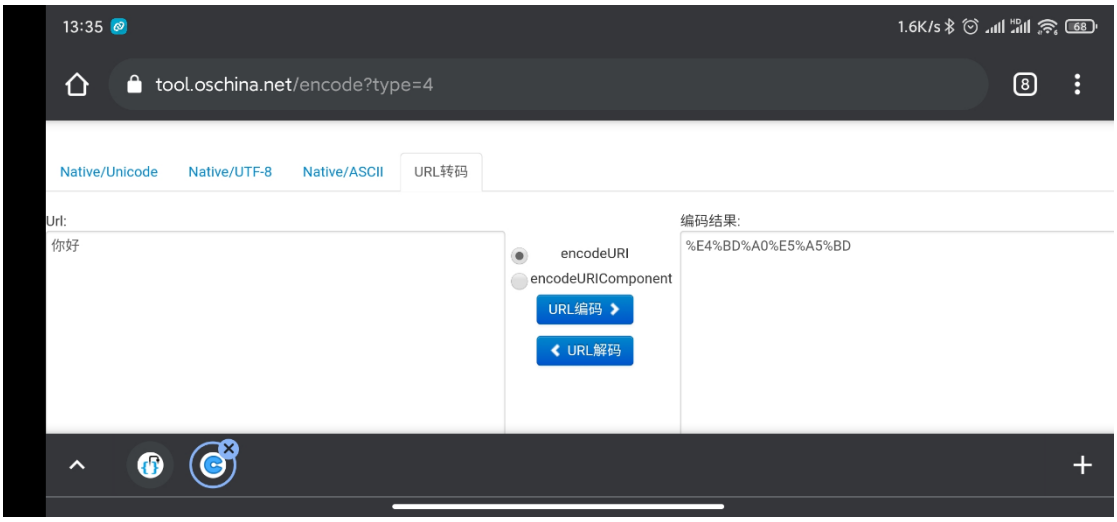


图 19 将中文名转换为 URL 编码格式

每一个%后面的数都是一个字节，每个汉字由 3 个字节组成。所以汉字“你好”转换为 URL 编码后则变为{0xe4,0xbd,0xa0,0xe5,0xa5,0xbd }，其中{0xe4,0xbd,0xa0}代表“你”，{0xe5,0xa5,0xbd}代表“好”。

在程序中以数组存储设备名称如图 20 所示：

```
/* URL编码表示中文：你好 */
uint8_t __name[6] = {0xe4,0xbd,0xa0,     //你
                     0xe5,0xa5,0xbd};    //好

/* Add as much of the device name as possible */
device_name_length = 6;
```

图 20 程序中以数组方式存储中文名的示例

1.6.2 获取连接设备的 RSSI 值

获取 RSSI 值的代码如图 21 所示。

```
static void gapc_get_rssi (uint8_t conidx)   /* GAPC_GET_INFO_CMD */
{
    struct gapc_get_info_cmd *cmd;

    cmd = KE_MSG_ALLOC(GAPC_GET_INFO_CMD, KE_BUILD_ID(TASK_GAPC, conidx),
                       KE_BUILD_ID(TASK_APP, conidx), gapc_get_info_cmd);
    cmd->operation = GAPC_GET_CON_RSSI;
    ke_msg_send(cmd);
}
```

图 21　RSL10 芯片获取连接链路 RSSI 值的代码

1.6.3 设置设备地址为随机私有可解析地址

以 ble_peripheral_server_bond 例程为例，在协议栈配置的时候需要在 app_config.h 文件的 devConfigCmd 结构体中将蓝牙地址类型设置为如图 22 所示的私有随机地址，然后在从机广播配置 advertiseCmd 结构体变量中如图 23 所示设置地址类型为随机地址，以上两个步骤操作完成后，广播时即可看到地址为可解析随机地址了。

gapm_task.h | app_config.c | app.h | *app_config.c | app.h

```
37  struct gapm_set_dev_config_cmd devConfigCmd =
38  {
39      .operation = GAPM_SET_DEV_CONFIG,
40      .role = GAP_ROLE_PERIPHERAL,
41      .renew_dur = GAPM_DEFAULT_RENEW_DUR,
42      .addr.addr = APP_BD_ADDRESS,
43      .irk.key = APP_IRK,
44      .addr_type = GAPM_CFG_ADDR_HOST_PRIVACY,//APP_BD_ADDRESS_TYPE,
45  #ifdef SECURE_CONNECTION
46      .pairing_mode = (GAPM_PAIRING_SEC_CON | GAPM_PAIRING_LEGACY),
47  #else
48      .pairing_mode = GAPM_PAIRING_LEGACY,
49  #endif
50      .gap_start_hdl = GAPM_DEFAULT_GAP_START_HDL,
51      .gatt_start_hdl = GAPM_DEFAULT_GATT_START_HDL,
52      .att_and_ext_cfg = GAPM_DEFAULT_ATT_CFG,
53      .sugg_max_tx_octets = GAPM_DEFAULT_TX_OCT_MAX,
54      .sugg_max_tx_time = GAPM_DEFAULT_TX_TIME_MAX,
55      .max_mtu = GAPM_DEFAULT_MTU_MAX,
56      .max_mps = GAPM_DEFAULT_MPS_MAX,
57      .max_nb_lecb = GAPM_DEFAULT_MAX_NB_LECB,
58      .audio_cfg = GAPM_DEFAULT_AUDIO_CFG,
59      .tx_pref_rates = GAP_RATE_ANY,
60      .rx_pref_rates = GAP_RATE_ANY
61  };
```

图 22　配置协议栈蓝牙地址类型为 host 控制的私有随机地址

```
struct gapm_start_advertise_cmd advertiseCmd =
{
    .op = {
        .code = GAPM_ADV_UNDIRECT,
        .addr_src = GAPM_GEN_RSLV_ADDR,//GAPM_STATIC_ADDR,
        .state = 0
    },
    .intv_min = GAPM_DEFAULT_ADV_INTV_MIN,
    .intv_max = GAPM_DEFAULT_ADV_INTV_MAX,
    .channel_map = GAPM_DEFAULT_ADV_CHMAP,
    .info.host = {
        .mode = GAP_GEN_DISCOVERABLE,
        .adv_filt_policy = ADV_ALLOW_SCAN_ANY_CON_ANY
        /* ADV_DATA and SCAN_RSP data are set in APP_BLE_Initialize() */
    }
};
```

图 23　配置广播的地址源是可解析的私有随机地址

随机地址的更新时间可以通过 app_config.h 文件中的 devConfigCmd 结构体的 renew_dur 成员进行设置，如图 24 所示，单位为秒，可设置的时间范围在 1~41400 秒。

```
struct gapm_set_dev_config_cmd devConfigCmd =
{
    .operation = GAPM_SET_DEV_CONFIG,
    .role = GAP_ROLE_PERIPHERAL,
    .renew_dur = GAPM_DEFAULT_RENEW_DUR,
    .addr.addr = APP_BD_ADDRESS,
    .irk.key = APP_IRK,
    .addr_type = GAPM_CFG_ADDR_HOST_PRIVACY,//APP_BD_ADD
#ifdef SECURE_CONNECTION
    .pairing_mode = (GAPM_PAIRING_SEC_CON | GAPM_PAIRING
#else
```

图 24　设置私有随机地址的更新时间

1.7　低功耗相关问题

1.7.1 RTC 唤醒

1. RTC 外设的中断都可以用于唤醒芯片低功耗的唤醒源吗？

答：在芯片进入 sleep 模式后，RTC 外设只有 RTC_ALARM 中断可以唤醒芯片，RTC_CLOCK 中断无法唤醒芯片。

2. RTC 外设在芯片进入 sleep 模式后是否会停止工作？

答：芯片进入 sleep 模式后，RTC 外设不会关闭。

1.7.2 定时唤醒

1. 芯片有多少种定时器可以作为唤醒源？

答：RSL10 芯片有 RTC 定时器和蓝牙基带外设的 BB_TIMER 定时器可以实现定时唤醒芯片。

1.7.3 BB_TIMER 定时器

1. BB_TIMER 在什么时候会启动？

答：只有当芯片通过“BLE_Power_Mode_Enter”函数进入低功耗的时候，BB_TIMER 才会启动开始计时，唤醒后 BB_TIMER 将停止计时。

2. BB_TIMER 的定时时间如何控制？

答：BB_TIMER 的最大定时时间可以通过“rwip.h”文件的“BLE_Sleep_MaxDuration_Set(int32_t maximum_value)”函数进行设置，该函数 maximum_value 参数的单位是 625us，该参数的最大值不能超过 32 位的 32.768KHz 频率的计数最大值，系统默认 BB_TIMER 的最大待机时间为 30s。

当芯片有蓝牙广播、蓝牙连接、蓝牙扫描等相关蓝牙时序活动正在进行，这时如果让芯片进入低功耗，BB_TIMER 的定时唤醒的时间将会被自动设置和蓝牙广播、连接等蓝牙时序相同以确保在准确的时刻唤醒芯片进行蓝牙通信。

1.8　SDK pack 版本 Bug

1.8.1 ONSemiconductor.RSL10.3.5.285.pack

1. RSSI 值获取问题：

在 1M PHY 的时候，获取的连接链路 RSSI 值会偏小，在 2M PHY 的时候，获取的连接链路 RSSI 值正常。

2. 免责声明

本着为用户提供更好服务的原则，广州立功科技股份有限公司（下称“立功科技”）在本手册中将尽可能地为用户呈现详实、准确的产品信息。但介于本手册的内容具有一定的时效性，立功科技不能完全保证该文档在任何时段的时效性与适用性。立功科技有权在没有通知的情况下对本手册上的内容进行更新，恕不另行通知。为了得到最新版本的信息，请尊敬的用户定时访问立功科技官方网站或者与立功科技工作人员联系。感谢您的包容与支持！

专业 . 专注成就梦想

Dreams come true with professionalism and dedication.

广州立功科技股份有限公司

更多详情请访问
www.zlgmcu.com

欢迎拨打全国服务热线
400-888-2705