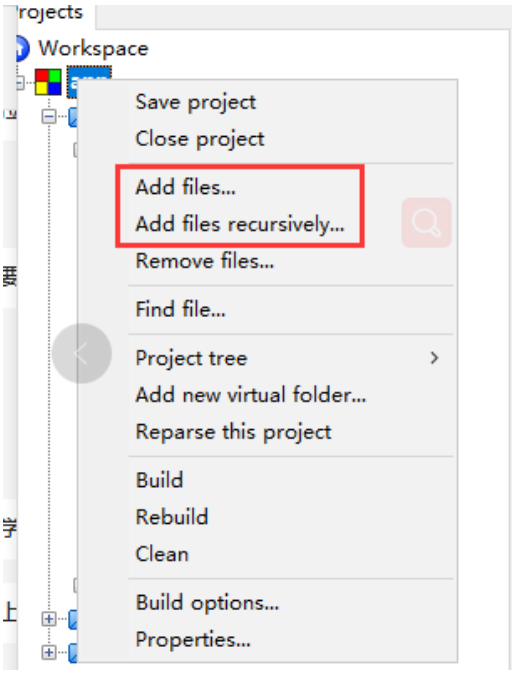
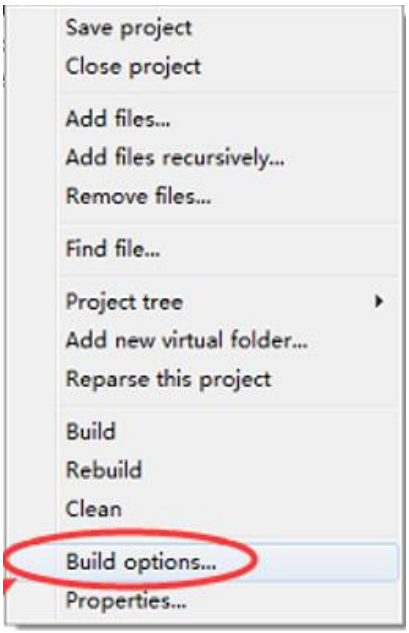
中科蓝讯 8918x 系列手表精简 SDK，移植注意事项

--- V1.0 版本

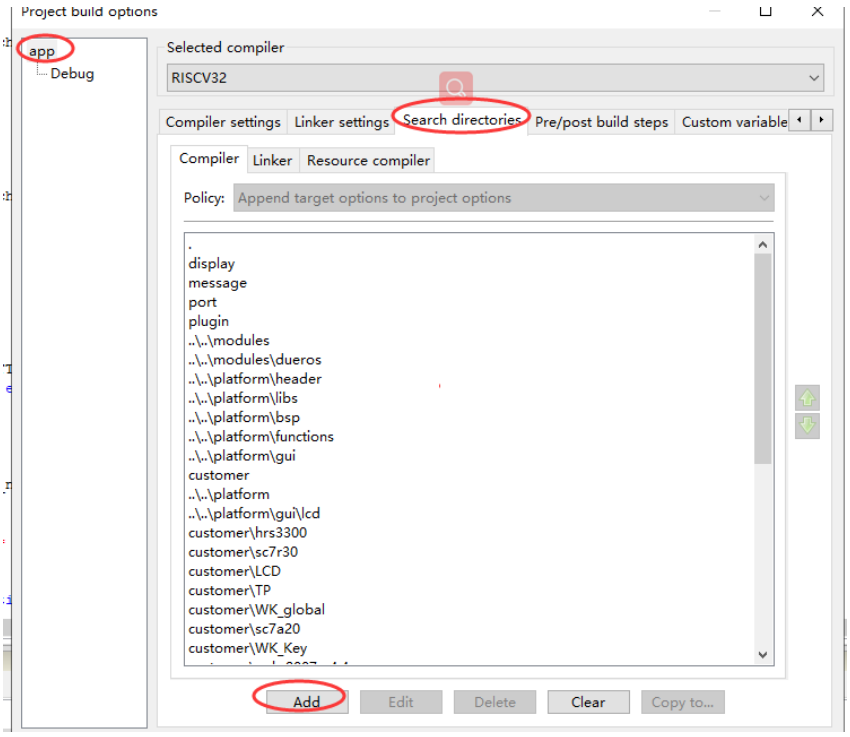
1. 如果往工程里添加代码
   a. 鼠标右键 app 工程，如下框图两个选项，第一个 Add files…是将指定文件添加到工程；第二个是 Add files recursively…将目录下所有代码文件添加到工程



   b. 包含文件引用路径方法
   右键 app -> 选择 Build options…



   按如下选择，添加路径进来即可



2. 常驻 RAM 函数

a) 哪些函数需要常驻在 RAM 里？

1.中断回调函数，及其被调用的子函数都需要常驻在 RAM 里，否则会导致死机复位等奇怪问题；

2.需要高效运行的代码，比如 5ms 定时回调函数，及其被调用的函数；

```
AT(.com_text.timer)
void usr_tmr5ms_thread(void)
{
    tmr5ms_cnt++;
```

b) 如何使用？

在函数上方加上 AT(.com_text.xxxx) ， xxxx 根据容易阅读命名

```
//timer tick interrupt(5ms)
AT(.com_text.timer)
void usr_tmr5ms_thread(void)
{
    tmr5ms_cnt++;
```

3. 函数的运行效率优化

建议客户非调用一次的函数，都集中放在 custom_text 段，举例：

AT(.custom_text.func)

Void func_test(void)

{...}

4. 变量定义相关的建议

全局变量建议放在 custom_buf 区里，举例：

char test[100] AT(.custom_buf.buff);

这样定义的 test 数组会放在额外分配的 RAM 里，否则没带 AT 指定的话，默认放在堆栈里。

5. 局部相关的建议

系统是基于 RTOS 的，所以线程的堆栈空间有限，建议不要定义特别大的局部变量，如果需要比较大的局部变量，可以调用 mem_malloc 分配吧。

6. 外部中断使用

在 downloader 工具配置中断 IO，和触发方式：

对应回调执行位置如下，客户可以在这里加入相关代码，建议这里处理一些短时的事件

```
AT(.com_text.timer)
void port_isr_do(u8 io_num)
{
    if(io_num == IO_PA7){
        if(WKUPEDG & (BIT(0) << 16))
        {
            WKUPCPND = (BIT(0) << 16);  //CLEAR PENDING
            //printf(strisr1);
        }
    } else if(io_num == IO_PB1){
        if(WKUPEDG & (BIT(1) << 16))
        {
            WKUPCPND = (BIT(1) << 16); //CLEAR PENDING
            //printf(strisr2);
        }
    }else if(io_num == IO_PB2){
        if(WKUPEDG & (BIT(2) << 16))
        {
            WKUPCPND = (BIT(2) << 16);  //CLEAR PENDING
            //printf(strisr3);
        }
    }else if(io_num == IO_PB3){
        if(WKUPEDG & (BIT(3) << 16))
        {
            WKUPCPND = (BIT(3) << 16); //CLEAR PENDING
            //printf(strisr4);
        }
    }else if(io_num == IO_PB4){
        if(WKUPEDG & (BIT(4) << 16))
        {
            WKUPCPND = (BIT(4) << 16); //CLEAR PENDING
            //printf(strisr5);
        }
    }else if(io_num == IO_PB5){
        if(WKUPEDG & (BIT(5) << 16))
        {
            WKUPCPND = (BIT(5) << 16); //CLEAR PENDING
            //printf(strisr6);
        }
    }else{
        if (WKUPEDG & (BIT(6) << 16))    //PORT INT
        {
            WKUPCPND = (BIT(6) << 16);   //CLEAR PENDING
            //printf(strisr7);
        }
    }
} « end port_isr_do »
```

7. 几个重要函数

a) 10ms 执行函数

每 10ms 执行一次，比如处理一些心率相关的事情

```
//仅唤醒的时候跑
AT(.text.sb.timer)
void thread_sb_10ms_callback(void)
{

}
```

a) 500ms 执行函数

每 500ms 执行一次，比如处理一些计步相关的事情

```
//休眠+唤醒一直跑
AT(.text.sb.timer)
void thread_sb_500ms_callback(void)
{
    rtc_sleep_process();
#if SC7A20_EN
    SL_SC7A20_PEDO_KCAL_WRIST_SLEEP_SWAY_ALGO();        //计步
#endif // SC7A20_EN
}
```

a) 5ms 执行函数

每 500ms 执行一次，比如处理一些计步相关的事情,为了效率，该函数及其子函数都要放在 com_text 区

```
//timer tick interrupt(5ms)
AT(.com_text.timer)
void usr_tmr5ms_thread(void)
{
```

a) 1ms 执行函数

每 1ms 执行一次，该函数是硬件定时器的回调，被调用的函数必须要放在 com_text 区

```
//void rtc_test(void);
//timer tick interrupt(1ms)
AT(.com_text.timer)
void usr_tmr1ms_isr(void)
{
```

8. BLE 相关的先参考 app.c 这个文件，里面有完整的例子