

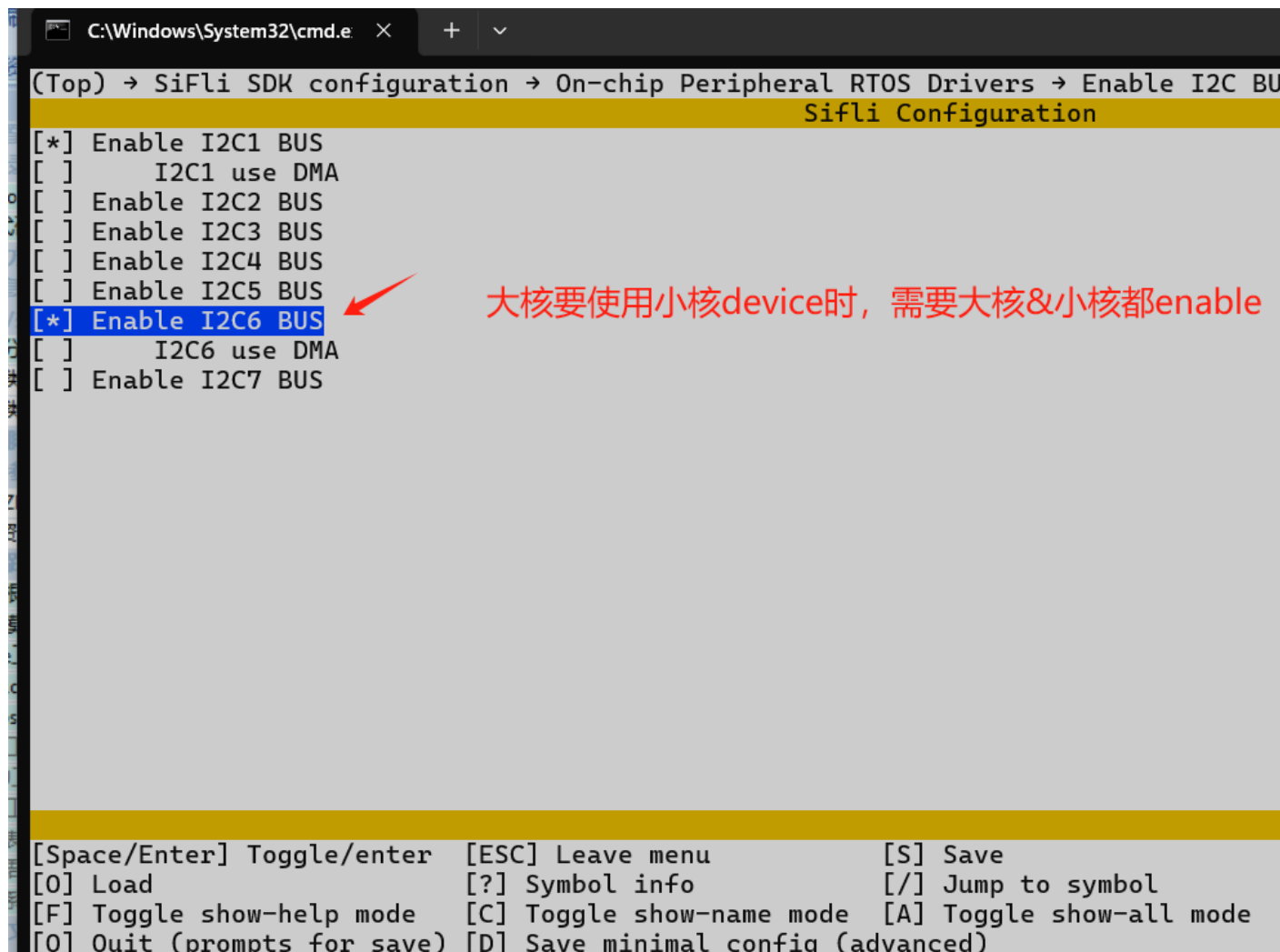
设备应用问题

1. 大小核 device 配置注意事项

1.1. 大核要用小核上的 device 设备时，大小核必须都配置为打开。不然初始化后，小核会关掉 device。

例：spi3/pwm5 都是小核的设备，在大核使用时，需要同步在小核也配置为打开。

lcpu 目前代码是看该模块是不是开了，如果没开回去关掉这个模块电进行省电。但是 lcpu 跟 hcpu 又分开的，所以会导致 lcpu 把挂载在自身的 device 关掉了。大核来使用出现异常。



```
C:\Windows\System32\cmd.e  X  +  v

(Top) → SiFli SDK configuration → On-chip Peripheral RTOS Drivers → Enable I2C BU
Sifli Configuration
[*] Enable I2C1 BUS
[ ] I2C1 use DMA
[ ] Enable I2C2 BUS
[ ] Enable I2C3 BUS
[ ] Enable I2C4 BUS
[ ] Enable I2C5 BUS
[*] Enable I2C6 BUS
[ ] I2C6 use DMA
[ ] Enable I2C7 BUS

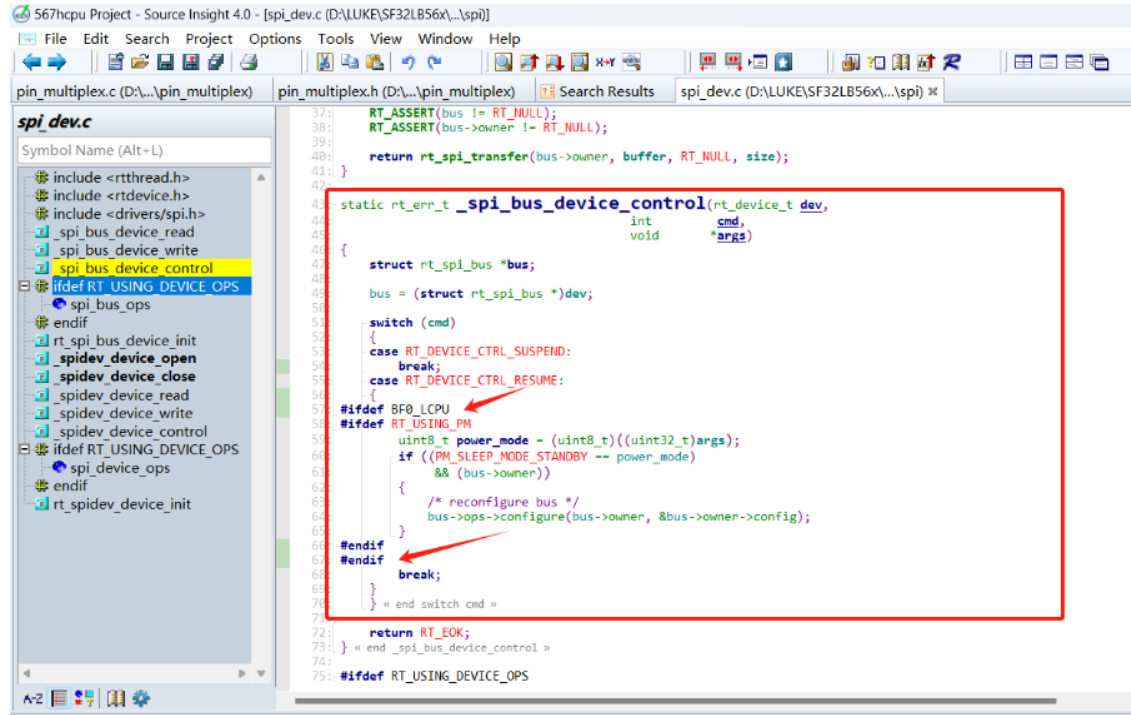
[Space/Enter] Toggle/enter  [ESC] Leave menu  [S] Save
[O] Load  [?] Symbol info  [/] Jump to symbol
[F] Toggle show-help mode  [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save)  [D] Save minimal config (advanced)
```

2. 同一个 device 在大小核都要应用时，大核睡眠后，小核唤醒会读写失败。

例：spi3/I2C5

验证了几个方式，大核不用是就反注册，但是再用的时候要在注册一次。

最后还是采用了 resume 只小核跑的时候在配置。56x 只存在大核睡了小核在跑的情景。不存在小核睡了，大核是唤醒状态。



2. 睡眠唤醒注意事项

2.1. 52x 系列

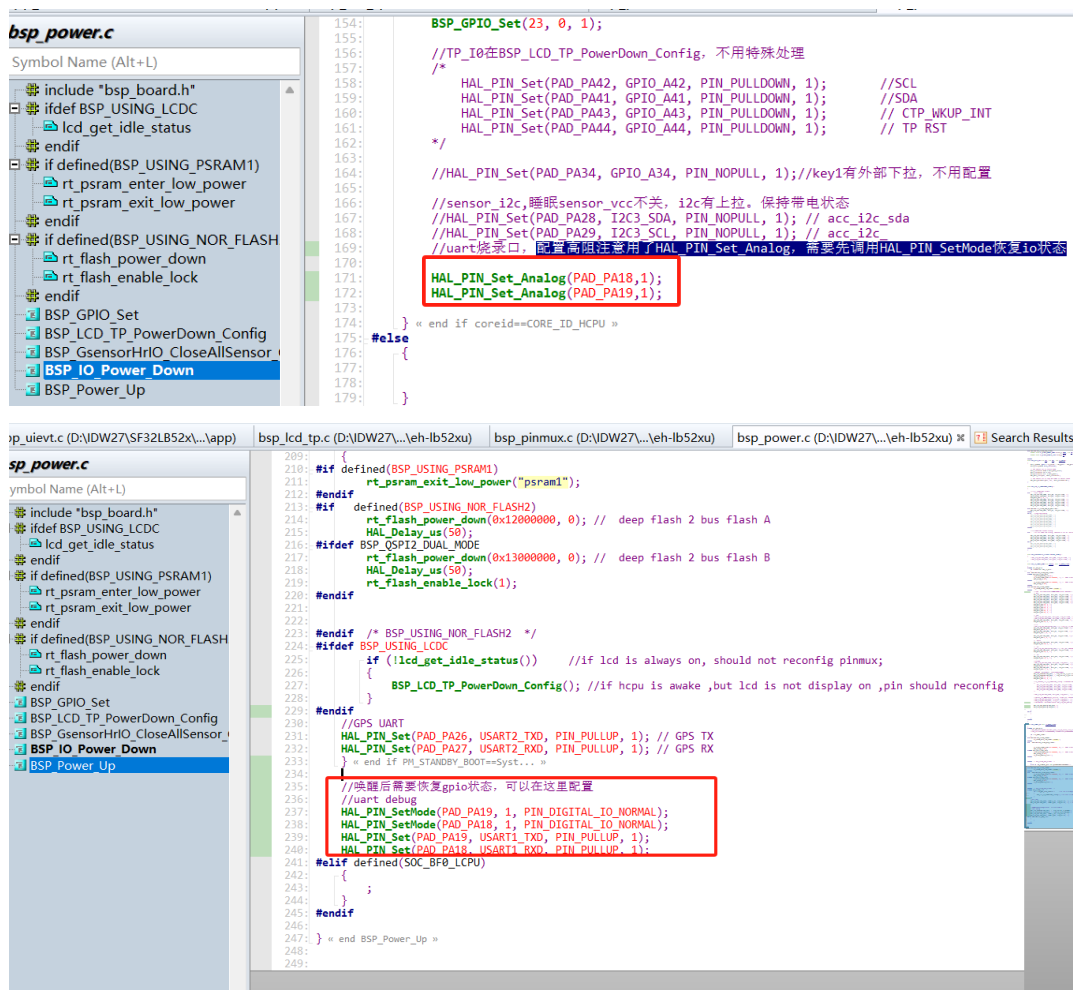
2.1.1. 52x 配置高阻注意用了 HAL_PIN_Set_Analog，需要先调用

HAL_PIN_SetMode 恢复 io 状态

52x 睡眠唤醒后不会去恢复 bsp_pinmux 里的 pin 脚状态。需要在代码里主动恢复

统一管理可以在睡眠前可以在 BSP_IO_Power_Down 配置睡眠的 pin 脚状态。

唤醒后可以在 BSP_Power_Up 里恢复。



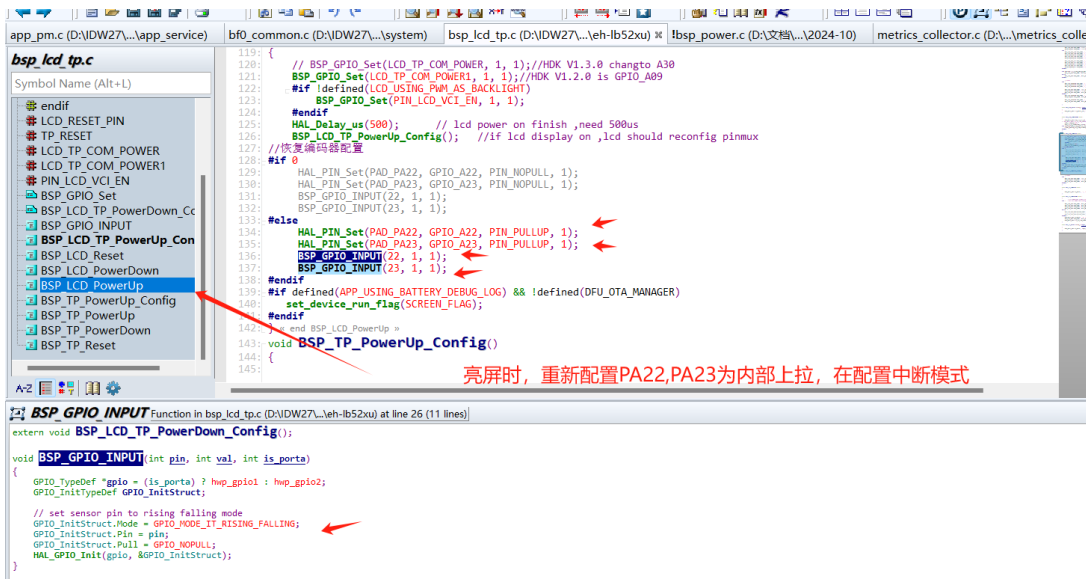
2.1.2. 52x 中断脚在睡眠前改了状态，要在唤醒前恢复中断模式。

不然无法使用。

例：

无感编码器睡眠后必须关闭上拉电源。一般客户为了节省成本，使用的内部上拉。

睡眠时就会配置为内部下拉，输出 0。避免漏电。用 `HAL_PIN_Set` 配置下拉后，改变了中断状态。需要在唤醒后配置。



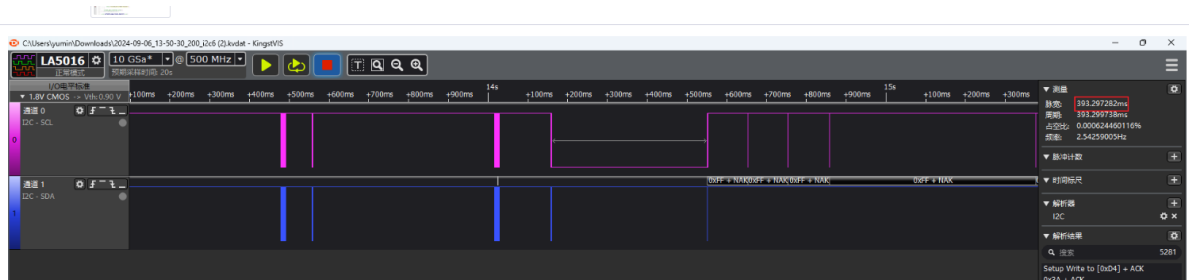
3. I2C 使用遇到过的问题

3.1. 线程优先级导致的 I2C 数据 err

问题原因：i2c 读写的线程优先级被更高的优先级打断，导致数据异常

解决方法：

1. 把 I2C 读写有线程优先级提高
2. 配置 DMA 方式读写也能规避改问题，但是 DMA 通道有限。要看下有没有其他设备在用



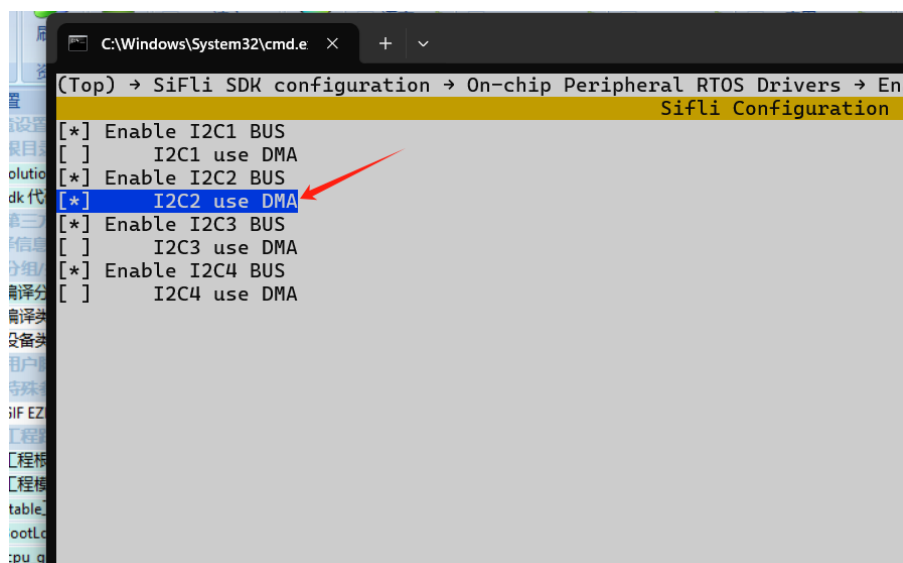
从波形看，就是I2C发送数据的时候，被更高优先级的线程或者中断打断了，导致I2C数据断了393ms，导致外设无ack，要解决这个问题。

可以尝试把Gsensor i2c中断读写线程的优先级高于nodic lcpu收发线程的优先级，看看还会出现如上图的被打断现象。例如如下的sensor线程：

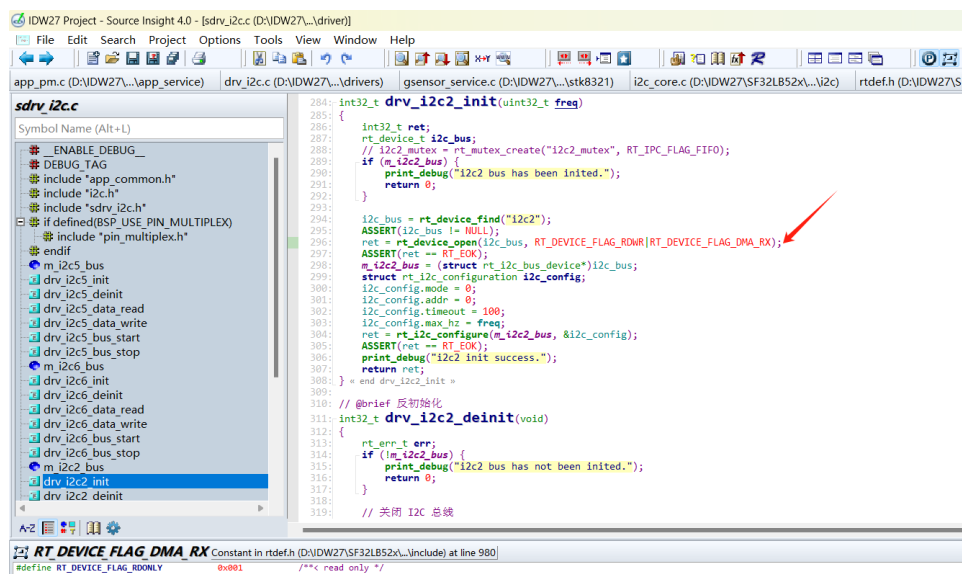
```
384:
385: static void sc7a20_algo_taskflow_entry(void *parameter)
386: {
387:     while(1)
388:     {
389:         if(RT_EOK == rt_sem_take(sc7a20_sem, RT_WAITING_FOREVER))
390:         {
391:             SL_SC7A20_Watch_Int_Fun();
392:             //LOG_I("SL_MCU_WAKE_CLOSE_SC7A20_INT_AND_TIME");
393:         }
394:     }
395: }
396:
397: void sc7a20_algo_task_init(void)
398: {
399:     ...
400:     ...occe_tash = rt_thread_create("accetask",
401:                                     sc7a20_algo_taskflow_entry, RT_NULL,
402:                                     SC7A20_TASK_STACK_SIZE,
403:                                     SC7A20_TASK_THREAD_PRIORITY, 10);
404:     if (acce_tash != RT_NULL)
405:         rt_thread_startup(acce_tash);
406:     ...
407:     ...
408:     ...LOG_I("sc7a20task thread create failed.");
409:     return;
410: }
411: ...sc7a20_sem = rt_sem_create("acce_sem", 0, RT_IPC_FLAG_FIFO);
412: if (sc7a20_sem == RT_NULL)
413: {
414:     LOG_E("sc7a20_sem create failed.");
415: }
416: ...sc7a20_gpio_interrupt_init();
417: ...end_sc7a20_algo_task_init...
418:
419:
420: void sc7a20_algo_task_deinit(void)
```

3.2. I2C 设备配置 dma

1) menuconfig 打开



2) 注册 I2C 设备 open 时需要|上 dma



3) 检查对应 I2C 的 dma 底层是否有配置 dma 通道，检查 dma_config.h

像是 52x 的 solution 代码，默认只给 I2C2 配置了 dma 通道。其他 I2C 通道没有配置。

配置新增是需要注意，I2C_DMA_REQUEST 是对应的物理寄存器地址，在芯片上是固定的。

每个外设不一样，需要看芯片手册。其他的几个参数都是对应的 channel，这个只要没被其他的设备使用都可以配置。

52x DMA req table←

req_sel	DMAC1	DMAC2
0	mpi1	usart4_tx
1	mpi2	usart4_rx
2	/	usart5_tx
3	i2c4	usart5_rx
4	usart1_tx	/
5	usart1_rx	/
6	usart2_tx	btim3
7	usart2_rx	btim4
8	gptim1_update	/
9	gptim1_trigger	/
10	gptim1_cc1	/
11	gptim1_cc2	/
12	gptim1_cc3	/
13	gptim1_cc4	/
14	btim1	/
15	btim2	/
16	atim1_update	/
17	atim1_trigger	/
18	atim1_cc1	/
19	atim1_cc2	/
20	atim1_cc3	/
21	atim1_cc4	/
22	i2c1	/
23	i2c2	/
24	i2c3	/
25	atim1_com	/
26	usart3_tx	/
27	usart3_rx	/
28	spi1_tx	/
29	spi1_rx	/
30	spi2_tx	/
31	spi2_rx	/
32	i2s1_tx	
33	i2s1_rx	
34	/	
35	/	
36	pdm1_rx_l	
37	pdm1_rx_r	
38	gpadc	
39	adc0	
40	adc1	

56x DMA req table←

req_sel	DMAC1	DMAC2
0	mpi1	usart4_tx
1	mpi2	usart4_rx
2	mpi3	usart5_tx
3	i2c4	usart5_rx
4	usart1_tx	usart6_tx
5	usart1_rx	usart6_rx
6	usart2_tx	btim3
7	usart2_rx	btim4
8	gptim1_update	gptim3_update
9	gptim1_trigger	gptim3_trigger
10	gptim1_cc1	gptim3_cc1
11	gptim1_cc2	gptim3_cc2
12	gptim1_cc3	gptim3_cc3
13	gptim1_cc4	gptim3_cc4
14	btim1	gptim5_update
15	btim2	gptim5_trigger
16	atim1_update	spi3_tx
17	atim1_trigger	spi3_rx
18	atim1_cc1	spi4_tx
19	atim1_cc2	spi4_rx
20	atim1_cc3	mpi5
21	atim1_cc4	i2c5
22	i2c1	i2c6
23	i2c2	i2c7
24	i2c3	gptim4_update
25	atim1_com	gptim4_trigger
26	usart3_tx	gptim4_cc1
27	usart3_rx	gptim4_cc2
28	spi1_tx	audadc_ch0/gptim4_cc3
29	spi1_rx	audadc_ch1/gptim4_cc4
30	spi2_tx	gpadc
31	spi2_rx	/
32	i2s1_tx	
33	i2s1_rx	
34	sci_tx	
35	sci_rx	
36	pdm1_rx_l	
37	pdm1_rx_r	
38	pdm2_rx_l	
39	pdm2_rx_r	
40	/	
41	dac0	
42	dac1	
43	gptim2_update	
44	gptim2_trigger	
45	gptim2_cc1	
46	audprc_tx_out_ch1	
47	audprc_tx_out_ch0	
48	audprc_tx_ch3	
49	audprc_tx_ch2	
50	audprc_tx_ch1	

3.3. 软件算法调用 I2C dma 读数据

- 1) 用 dma 读书时需要注意，需要配置一个全局的静态数组。如果缓存数据的地址不是固定的会导致 dma 搬数后，应用层返回的是 0 值。

```
watch > app > driver > stk8327_jdo > C:\drv_stk8327.c > drv_gsensor_stk8327_fifo_data_get(yz_t *xyz, uint8_t *count)
350 uint8_t fifo_TEXT[1] = {0};
351 uint8_t fifo_p_data[6] = {0};
352 void drv_gsensor_stk8327_fifo_data_get(yz_t *xyz, uint8_t *count)
353 {
354     uint8_t fifo_len;
355     uint32_t i;
356     uint8_t data[sizeof(yz_t)*ACCD_TOTAL_FIFO_LENGTH] = {0};
357     uint8_t *p_data;
358     int32_t err = 0;
359
360     ASSERT(p_xyz);
361     i = *count;
362     ASSERT(i > 0 && i <= 32);
363
364     drv_gsensor_bus_start();
365     // 读取数据长度
366     // stk8327_register_read(0x9C, &fifo_len);
367     stk8327_register_read(0x9C, &fifo_TEXT);
368     // print_debug("fifo_len222 = %d", fifo_len);
369     print_debug("fifo_len222 = %d", fifo_TEXT[0]);
370     fifo_len = fifo_TEXT[0];
371     // 限制最大长度为32
372     fifo_len &= 0x7f;
373     print_debug("fifo_len222 = %d", fifo_len);
374     if(ACCD_TOTAL_FIFO_LENGTH < fifo_len)
375     {
376         fifo_len = ACCD_TOTAL_FIFO_LENGTH;
377     }
378
379     uint8_t val = 0;
380     // print_debug("fifo_len=%d, i=%d", fifo_len, i);
381
382     p_data = data;
383     memset(p_xyz, 0, sizeof(yz_t)*ACCD_TOTAL_FIFO_LENGTH);
384
385     if (fifo_len > 0)
386     {
387         drv_gsensor_register_data_read(STK8327_ADDR, 0x3F, fifo_p_data, 6*fifo_len);
388         for (i=0; i<fifo_len; i++)
389         {
390             //err = drv_gsensor_register_data_read(STK8327_ADDR, 0x3F, fifo_p_data, 6);
391             //rt_kprintf("%d %d %d\r\n", p_data[i]);
392             //val = 0;
393         }
394     }
395 }
```

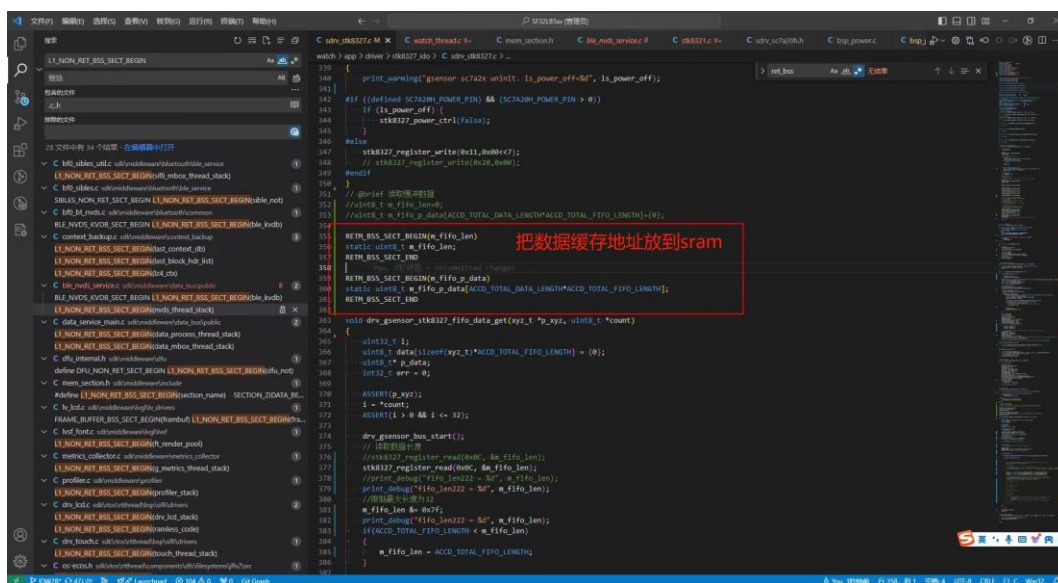
ACCD_TOTAL_FIFO_LENGTH 第 5 项

连续数组大小忘记改了

指针改为静态数组

for循环读改为连读

2) 软件运行频繁，速率高时对数组的加载位置也有影响。默认注册的静态数组是在 psram，如果读数据频繁，数据多。在 psram 搬运可能来不及，会出现上层拿到的数据部分为 0。解决方法，把注册的静态数组放在 sram 里。这么做会让 dma 搬运速度提升，避免速度不够出现的 dma 搬运没完成。



4.GPIO 配置 FAQ

4.1. gpio 中断怎么配置，中断唤醒怎么配置

solution 代码里可以参考 button_service.c 跟 battery_comm.c

```

165: static void GpsEnableWake(void)
166: {
167:     #ifdef BSP_USING_PM
168:     ...int8_t wakeup_pin;
169:     ...uint16_t gpio_pin;
170:     ...GPIO_TypeDef* gpio;
171:     #endif /* BSP_USING_PM */
172:     ...log_debug("...");
173:     ...//加载中断
174:     ...rt_pin_mode(WAKE_PIN, PIN_MODE_INPUT);
175:
176:     #ifdef BSP_USING_PM
177:     ...gpio = GET_GPIO_INSTANCE(WAKE_PIN);
178:     ...gpio_pin = GET_GPIOx_PIN(WAKE_PIN);
179:
180:     #if (WAKE_PIN > 96)
181:     ...wakeup_pin = HAL_LPAON_QueryWakeupPin(gpio, gpio_pin);
182:     #else
183:     ...wakeup_pin = HAL_HPAON_QueryWakeupPin(gpio, gpio_pin);
184:     #endif
185:     ...RT_ASSERT(wakeup_pin >= 0);
186:
187:     ...pm_enable_pin_wakeup(wakeup_pin, AON_PIN_MODE_DOUBLE_EDGE);
188:     #endif /* BSP_USING_PM */
189:
190:     ...rt_pin_attach_irq(WAKE_PIN, PIN_IRQ_MODE_RISING_FALLING, um_loda_wake_irq_handler, NULL);
191:     ...rt_pin_irq_enable(WAKE_PIN, 1);
192: } //end GpsEnableWake
193:

```

中断唤醒，PB脚在大核使用
可以直接唤醒大核。但是在小核使用
只能唤醒小核，要同步消息订阅去唤醒大核

pin脚注意区分大小核

唤醒使能

唤醒后，正常执行中断函数

4.2. 配置高阻

The screenshot shows the 'bsp_power.c' file in an IDE. The left pane lists symbols, and the right pane shows the code. The bottom pane shows the 'HAL_PIN_Set Analog' function definition. A red arrow points to the 'pad' parameter in the function signature.

```

271:     if (is_deep_sleep)
272:     {
273:     #if defined(BSP_USING_PSRAM1)
274:     rt_psram_enter_low_power("psram1");
275:     #endif
276:     }
277:     if (flag_hcpu_powerdown_standby)
278:     {
279:     BSP_Psram_Flash_IO_PowerStandby_Config(); //reconfig pinmux
280:     }
281:     } // end if coreid==CORE_ID_HCPU
282:
283:     #else
284:     {
285:     #if 1
286:     HAL_PIN_Set_Analog(PAD_PB13, 0);
287:     HAL_PIN_Set_Analog(PAD_PB15, 0);
288:     //uart4
289:     HAL_PIN_Set_Analog(PAD_PB16, 0);
290:     HAL_PIN_Set_Analog(PAD_PB17, 0);
291:
292:     HAL_PIN_Set_Analog(PAD_PB18, 0);
293:
294:     HAL_PIN_Set(PAD_PB19, GPIO_B19, PIN_PULLDOWN, 0);
295:

```

HAL_PIN_Set Analog Function in b0_hal_pinmux.c (D:\gtx12\hal) at line 976 (31 lines)

```

/**
 * @brief Set pin for analog function, fix for ROM patch, avoid pin_const update.
 * @param pad: physical pin
 * @param hcpu: 1: pin for hcpu; 0: pin for lcpu
 * @retval -1 if invalid, otherwise 0
 */
// for SF32LB55X pin function is fixed to 10 for analog function, PS/PE should be 0(NOPULL).
#define PIN_ANALOG_FUNC 0xf
int HAL_PIN_Set_Analog(int pad, int hcpu)
{
    volatile uint32_t *pin;
    uint32_t val;
    #ifdef hwp_0br
    if (!hcpu && pad >= PAD_PB80)

```

- 1) “pad” 注意 PA,PB 不要搞错，PA 是 HCPU，PB 是 LCPU。
- 2) hcpu，这里是区分大小核的，PA 脚是大核的 PIN 脚，这里是 1. PB 脚是小核的 PIN 脚，要用 0.

3.faq

1. I2C 报错怎么办
 - 1) 检查外部上拉，或者内部上拉。确认上拉是否异常。外部上拉是否匹配，I2C 配置的速率，外设是否支持。

例：跑 400k 速率，外部上拉用的 4.7K 或者 10k。I2C 实际测量出来的速率会不对。

2) 检查外设电源，是否异常。

例：经常又同学的外设电源是跟其他地方共用的，导致电源被关。外设断电，导致 I2C 访问异常。

3) 上逻辑分析仪

逻辑分析仪，是调试驱动必备的。能分析具体数据。

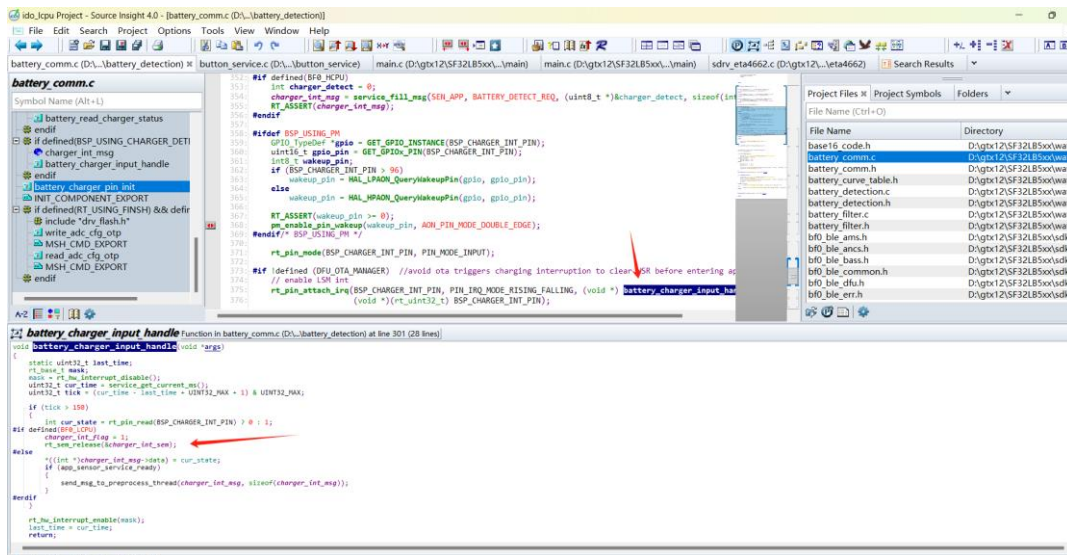
2. uart 报错怎么办

由于 uart fifo 只有 1 字节。所有用 RX 中断方式会出现接收漏掉的情况。

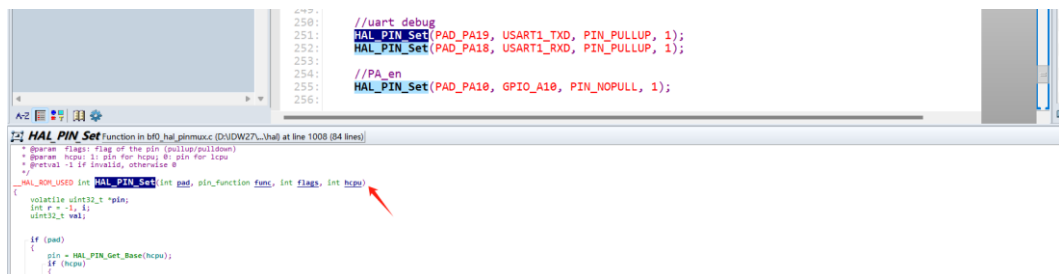
需要用 RX_DMA 来接收。

3. 中断唤醒后，怎么马上又睡眠了。应该怎么唤醒后去做用户逻辑

唤醒后要看中断处理逻辑，没事件需要处理。系统进入 idle 一段时间就会睡眠



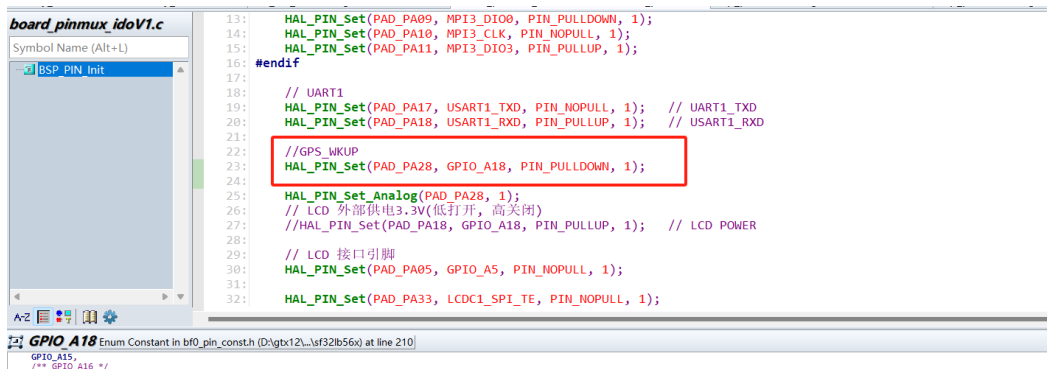
4. 为什么配置了 pin 脚不生效



1) “pad” 注意 PA,PB 不要搞错，PA 是 HCPU，PB 是 LCPU。经常有同学搞错。PB 能在 HCPU 直接调用。PA 不能直接在 LCPU 调用。

2) func，注意对照 pinmux，config 表看下是否支持，注意是否配置错了。

例：PA28,对应的是 GPIO_A28，不要复制，粘贴忘记改了。



3) flags，内部上下拉状态。有外部上下拉的，不需要再配置内部上下拉了。这里对于通讯接口（I2C/SPI）统一推荐使用外部上拉，芯片内部上拉不可调节。大概在 18K 左右，速率高的时候会出现数据异常

4) hcpu，这里是区分大小核的，PA 脚是大核的 PIN 脚，这里是 1. PB 脚是小核的 PIN 脚，要用 0.

经常有同学复制，粘贴忘记改了。

5. 功耗优化（算法&数据搬运到 PSRAM&SRAM）

1) 算法先从 nor_flash 移到 psram (+RO)。编译后，可以搜相关函数。看地址是不是变 0x6 开头

2) 算法数据全移到 SRAM (+RW+ZI) 编译后，可以搜相关函数。看地址是不是变 0x2 开头

3) 常用算法移动到 SRAM (+CODE)（验证驱动&算法相关代码搬运到 sram 处理，功耗有降低（爱都 IDW27/SIFLI523）平均功耗 30ua）.对于有功耗优化诉求的客户都可以尝试下

Line 235648: m_sensor_thread_mq		0x2005d488	Data	4	app_sensor_thread.o(.bss..L_MergedGlobals)
Line 235649: [Anonymous Symbol]	数据在SRAM	0x2005d488	Section	0	app_sensor_thread.o(.bss..L_MergedGlobals)
Line 235650: m_sensor_thread		0x2005d48c	Data	4	app_sensor_thread.o(.bss..L_MergedGlobals)
Line 235651: m_sys_timer_max_interval_count		0x2005d490	Data	4	app_sensor_thread.o(.bss..L_MergedGlobals)
Line 235652: m_sys_timer		0x2005d494	Data	4	app_sensor_thread.o(.bss..L_MergedGlobals)
Line 239260: [Anonymous Symbol]		0x600131ac	Section	0	app_sensor_thread.o(.text.app_send_msg_to_sensor_thread)
Line 239261: __arm_cp_0_1		0x60014020	Number	4	app_sensor_thread.o(.text.app_send_msg_to_sensor_thread)
Line 239262: [Anonymous Symbol]		0x60014024	Section	0	app_sensor_thread.o(.text.app_sensor_load_function)
Line 239264: app_sensor_thread_entry		0x6001405d	Thumb Code	96	app_sensor_thread.o(.text.app_sensor_thread_entry)
Line 239264: app_sensor_thread_entry		0x6001405d	Thumb Code	96	app_sensor_thread.o(.text.app_sensor_thread_entry)
Line 239264: app_sensor_thread_entry		0x6001405d	Thumb Code	96	app_sensor_thread.o(.text.app_sensor_thread_entry)
Line 239265: [Anonymous Symbol]		0x6001405c	Section	0	app_sensor_thread.o(.text.app_sensor_thread_entry)
Line 239265: [Anonymous Symbol]		0x6001405c	Section	0	app_sensor_thread.o(.text.app_sensor_thread_entry)
Line 239266: [Anonymous Symbol]		0x600140bc	Section	0	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239266: [Anonymous Symbol]		0x600140bc	Section	0	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239267: __arm_cp_4_0		0x60014140	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239267: __arm_cp_4_0		0x60014140	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239268: __arm_cp_4_2		0x60014150	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239268: __arm_cp_4_2		0x60014150	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239269: __arm_cp_4_3		0x60014154	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239269: __arm_cp_4_3		0x60014154	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239270: __arm_cp_4_4		0x60014158	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)
Line 239270: __arm_cp_4_4		0x60014158	Number	4	app_sensor_thread.o(.text.app_sensor_thread_init)

D:\DW27\SF32LB52x\watch\si\template\config\hcpu\linker_script\link_flash_523.sct2024/6/26 13:17:31

234app_mainmenu_layout_cellular.o (+CODE)

235lvsf_trans_anim.o (+CODE)

236#endif

237#endif

238ido_algo*.o (+R0)

239sdrv*.o (+R0)

240*.o (sys_timer_handle)

241*.o (ido_algo_activity_result)

242'ido_alg_m33_V1_0_42'lib (+R0)

243'gsensor*.o (+R0)

244app_sensor_thread.o (+R0)

245}

246

247RW_FSRAM_RET2 +0

248

D:\文档\WeChat Files\wxid_j3h4k76orgeu22\FileStorage\File\2024-06\link_flash_523.sct2024/6/20 14:57:

234app_mainmenu_layout_cellular.o (+CODE)

235lvsf_trans_anim.o (+CODE)

236#endif

237#endif

238ido_algo*.o (+R0)

239sdrv*.o (+R0)

240*.o (sys_timer_handle)

241*.o (ido_algo_activity_result)

242'ido_alg_m33_V1_0_42'lib (+R0)

243app_sensor_thread.o (+R0)

244'gsensor*.o (+R0)

245}

246

247

248RW_FSRAM_RET2 +0

249

link_flash_523.sct [Beyond Compare]2024/6/26 13:17:31

284

285RW_FSRAM_SAVE_DATA +0{

286.NBY (+RW +ZI)

287}

288

289

290#ifdef BSP_USING_FSRAM1

291ScatterAssert((ImageLength(RW_FSRAM_NON_RET) + ImageLength(RW_FSRAM_NAND) + ImageLength(RW_FSRAM

292#else

293ScatterAssert((ImageLength(RW_FSRAM_NON_RET) + ImageLength(RW_FSRAM_NAND) + ImageLength(RW_FSRAM

294#endif

295#endif

296

297RW_IRAM0_HFSYS_RAM0_BASE UNINIT { ; ZI data, not retained

298//The previous 128KB is the retention area

299audio_server.o (.bss.audio_server_stack)

300audio_server.o (.bss.bt_downvoice_stack)

301media_player_server.o (.bss.ffmpeg_thread_stack)

302*.o (.l1_non_ret_bss_uhog_be_ram_buf)

303#if defined (SOLUTION_RES_USING_NAND)

304drv_spi_nand.o (.l2_ret_bss_nand_buf)

305nand.o (.bss.dhara_meta_cache)

306#endif

307bts2_app_av_snk.o (.bss.displayback_thread_stack)

308*.o (STACK) ; ISR stack

309*.o (.l1_non_ret_bss_drv_lcd_stack)

310*.o (.bss.sensor_thread_stack)

311*.o (.bss.idle_thread_stack)

312*.o (.l1_non_ret_bss_bg_thread_stack)

313*.o (.l1_non_ret_bss_timer_thread_stack)

314*.o (.l1_non_ret_bss_touch_thread_stack)

315*.o (.l1_non_ret_bss_watch_thread_stack)

316*.o (.l1_non_ret_bss_frambuf)

317*.o (.l1_non_ret_bss_app_sram_non_ret_cache)

318*.o (.l2_non_ret_bss_ft_render_pool)

319

320#ifdef BSP_USING_FSRAM

321*.o (.l2_non_ret_data_*)

322

323

324

D:\文档\WeChat Files\wxid_j3h4k76orgeu22\FileStorage\File\2024-06\link_flash_523.sct2024/6/20 14:57:

285

286RW_FSRAM_SAVE_DATA +0{

287.NBY (+RW +ZI)

288}

289

290

291#ifdef BSP_USING_FSRAM1

292ScatterAssert((ImageLength(RW_FSRAM_NON_RET) + ImageLength(RW_FSRAM_NAND) + ImageLength(RW_FSRAM

293#else

294ScatterAssert((ImageLength(RW_FSRAM_NON_RET) + ImageLength(RW_FSRAM_NAND) + ImageLength(RW_FSRAM

295#endif

296#endif

297

298RW_IRAM0_HFSYS_RAM0_BASE UNINIT { ; ZI data, not retained

299//The previous 128KB is the retention area

300audio_server.o (.bss.audio_server_stack)

301audio_server.o (.bss.bt_downvoice_stack)

302media_player_server.o (.bss.ffmpeg_thread_stack)

303*.o (.l1_non_ret_bss_uhog_be_ram_buf)

304#if defined (SOLUTION_RES_USING_NAND)

305drv_spi_nand.o (.l2_ret_bss_nand_buf)

306nand.o (.bss.dhara_meta_cache)

307#endif

308bts2_app_av_snk.o (.bss.displayback_thread_stack)

309*.o (STACK) ; ISR stack

310*.o (.l1_non_ret_bss_drv_lcd_stack)

311*.o (.bss.sensor_thread_stack)

312*.o (.bss.idle_thread_stack)

313*.o (.bss.ido_sensor_stack)

314*.o (.l1_non_ret_bss_bg_thread_stack)

315*.o (.l1_non_ret_bss_timer_thread_stack)

316*.o (.l1_non_ret_bss_touch_thread_stack)

317*.o (.l1_non_ret_bss_watch_thread_stack)

318*.o (.l1_non_ret_bss_frambuf)

319*.o (.l1_non_ret_bss_app_sram_non_ret_cache)

320*.o (.l2_non_ret_bss_ft_render_pool)

321

322#ifdef BSP_USING_FSRAM

323*.o (.l2_non_ret_data_*)

324

325

326

D:\DW27\SF32LB52x\watch\sim\template\config\hcpu\linker_script\link_flash_523.sct2024/6/26 13:17:31 <pre> 407 ; for PSRAM clock changes 408 hf0_hal_rom.o (+RW +ZI) 409 bsp_init.o (+RW +ZI) 410 411 *.o (non_ret) ; non-retention section 412 *.o (.data.SystemCoreClock) 413 *.o (.l1_non_ret_bss_ft_render_pool) 414 *.o (.l1_non_ret_bss_ft_render_pool) 415 *.o (.l1_non_ret_bss_ft_render_pool) 416 *.o (.l1_non_ret_bss_ft_render_pool) 417 *.o (.l1_non_ret_bss_ft_render_pool) 418 *.o (.l1_non_ret_bss_ft_render_pool) 419 *.o (.l1_non_ret_bss_ft_render_pool) 420 *.o (.l1_non_ret_bss_ft_render_pool) 421 *.o (.l1_non_ret_bss_ft_render_pool) 422 *.o (.l1_non_ret_bss_ft_render_pool) 423 *.o (.l1_non_ret_bss_ft_render_pool) 424 *.o (.l1_non_ret_bss_ft_render_pool) 425 *.o (.l1_non_ret_bss_ft_render_pool) 426 *.o (.l1_non_ret_bss_ft_render_pool) 427 *.o (.l1_non_ret_bss_ft_render_pool) 428 *.o (.l1_non_ret_bss_ft_render_pool) 429 *.o (.l1_non_ret_bss_ft_render_pool) 430 *.o (.l1_non_ret_bss_ft_render_pool) 431 *.o (.l1_non_ret_bss_ft_render_pool) 432 *.o (.l1_non_ret_bss_ft_render_pool) 433 *.o (.l1_non_ret_bss_ft_render_pool) 434 *.o (.l1_non_ret_bss_ft_render_pool) 435 *.o (.l1_non_ret_bss_ft_render_pool) 436 *.o (.l1_non_ret_bss_ft_render_pool) 437 *.o (.l1_non_ret_bss_ft_render_pool) 438 *.o (.l1_non_ret_bss_ft_render_pool) 439 *.o (.l1_non_ret_bss_ft_render_pool) 440 *.o (.l1_non_ret_bss_ft_render_pool) 441 *.o (.l1_non_ret_bss_ft_render_pool) 442 *.o (.l1_non_ret_bss_ft_render_pool) </pre>	D:\文档\WeChat Files\wxid_j3h4k76orgeu22\FileStorage\File\2024-06\link_flash_523.sct2024/6/20 14: <pre> 409 ; for PSRAM clock changes 410 hf0_hal_rom.o (+RW +ZI) 411 bsp_init.o (+RW +ZI) 412 413 *.o (non_ret) ; non-retention section 414 *.o (.data.SystemCoreClock) 415 *.o (.l1_non_ret_bss_ft_render_pool) 416 *.o (.l1_non_ret_bss_ft_render_pool) 417 *.o (.l1_non_ret_bss_ft_render_pool) 418 *.o (.l1_non_ret_bss_ft_render_pool) 419 *.o (.l1_non_ret_bss_ft_render_pool) 420 *.o (.l1_non_ret_bss_ft_render_pool) 421 *.o (.l1_non_ret_bss_ft_render_pool) 422 *.o (.l1_non_ret_bss_ft_render_pool) 423 *.o (.l1_non_ret_bss_ft_render_pool) 424 *.o (.l1_non_ret_bss_ft_render_pool) 425 *.o (.l1_non_ret_bss_ft_render_pool) 426 *.o (.l1_non_ret_bss_ft_render_pool) 427 *.o (.l1_non_ret_bss_ft_render_pool) 428 *.o (.l1_non_ret_bss_ft_render_pool) 429 *.o (.l1_non_ret_bss_ft_render_pool) 430 *.o (.l1_non_ret_bss_ft_render_pool) 431 *.o (.l1_non_ret_bss_ft_render_pool) 432 *.o (.l1_non_ret_bss_ft_render_pool) 433 *.o (.l1_non_ret_bss_ft_render_pool) 434 *.o (.l1_non_ret_bss_ft_render_pool) 435 *.o (.l1_non_ret_bss_ft_render_pool) 436 *.o (.l1_non_ret_bss_ft_render_pool) 437 *.o (.l1_non_ret_bss_ft_render_pool) 438 *.o (.l1_non_ret_bss_ft_render_pool) 439 *.o (.l1_non_ret_bss_ft_render_pool) 440 *.o (.l1_non_ret_bss_ft_render_pool) 441 *.o (.l1_non_ret_bss_ft_render_pool) 442 *.o (.l1_non_ret_bss_ft_render_pool) </pre>
--	--