常见问题(FQA)

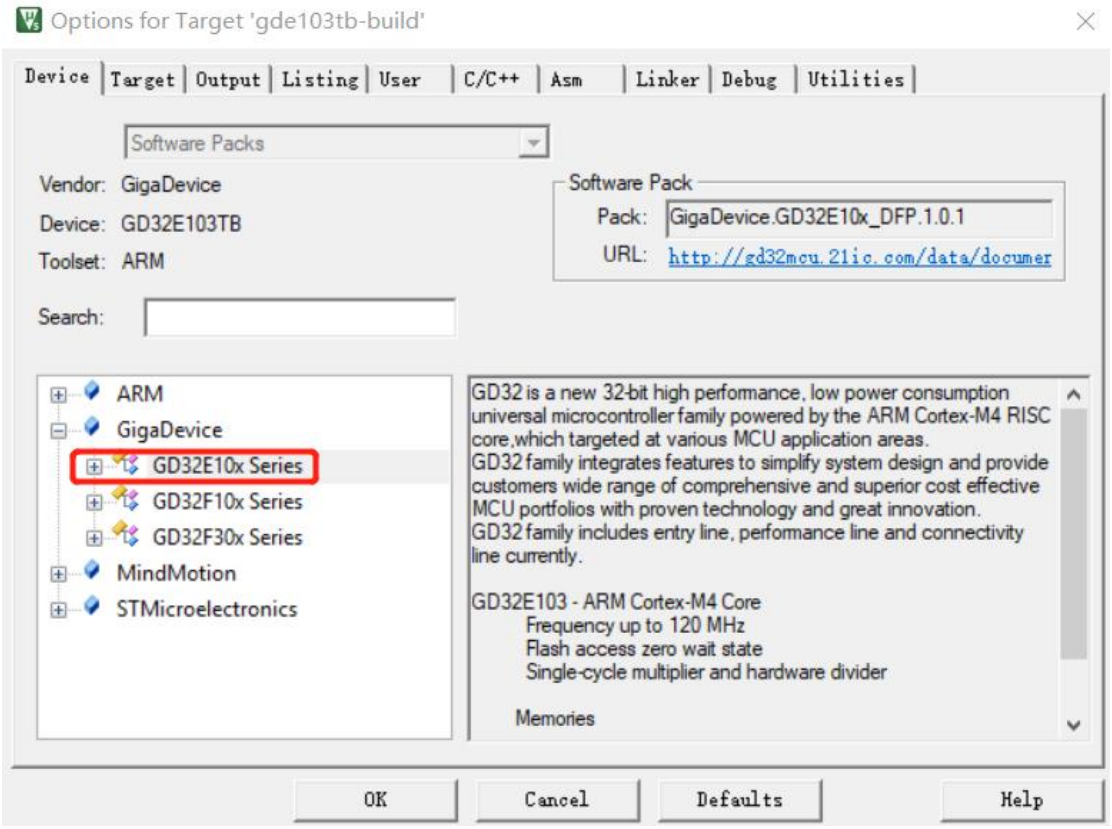
东信源芯微电子有限公司

1. keil 中找不到 GD MCU 的支持包

GD 官网资料下载链接：http://www.gd32mcu.com/cn/download?kw=&lan=cn

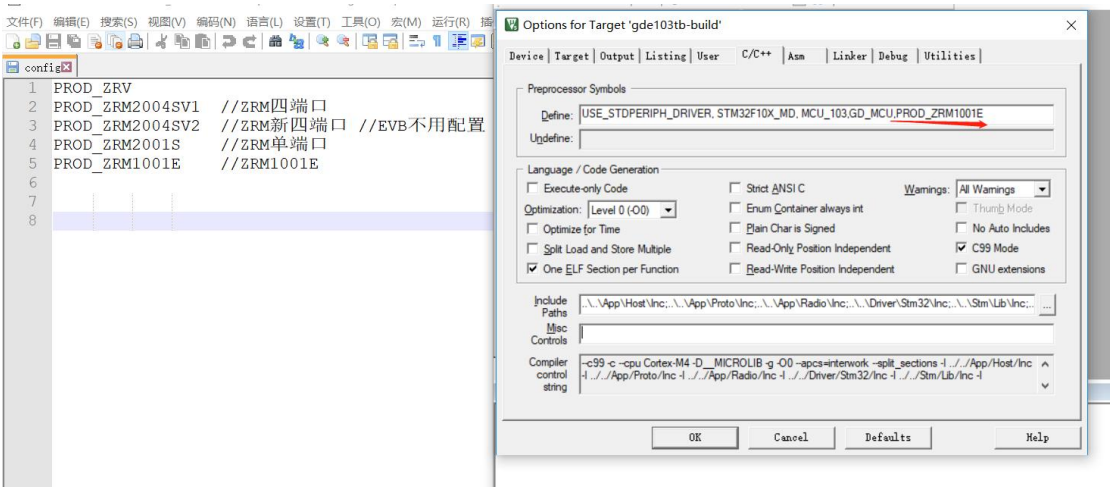
例如：GD32F1 系列 keil 支持包

安装完成后，重新打开 keil，显示如下：



2. SDK 编译时，如何指定产品类型

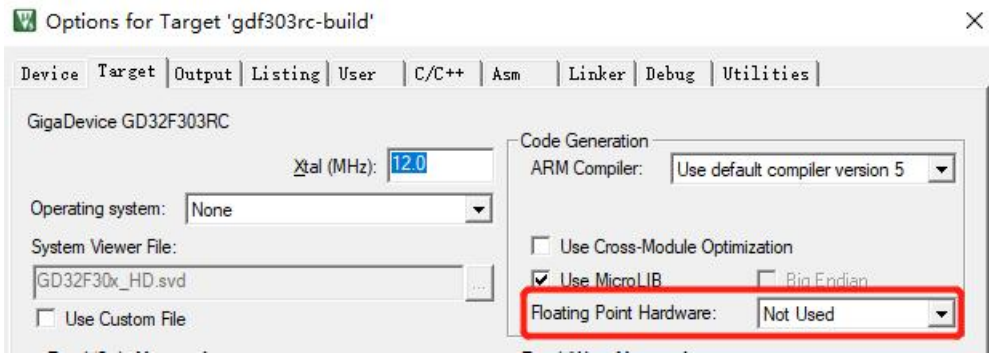
根据 config 文件中的产品宏定义，找出产品对应的类型。在 keil define 中进行设置。例如，产品型号为 ZRM1001E，设置如下：



3. SDK 运行时，出现 HardFault_Handler 错误，解决方法。

```
/**
  * @brief  This function handles Hard Fault exception.
  * @param  None
  * @retval None
  */
void HardFault_Handler(void)
{
  /* Go to infinite loop when Hard Fault exception occurs */
  while (1)
  {
  }
}
```

使用 GD 的 MCU，Keil 软件配置需要设置 Floating Point Hardware 为 not used



4. 如何调试 MCU 与 RFID 芯片的 SPI，验证 SPI 工作正常？

SDK 源码中关于 SPI 通信的测试程序。

```
//用于测试SPI与RFID芯片的通信是否正常
void rfid_spi_test(void)
{
    uint8_t data = 0;

    rfes_pcb_power_up();                          //打开供电
    GPIO_WriteBit(GPIOA, GPIO_Pin_1, Bit_RESET);  //RFID芯片复位
    mcu_delay_us(5000);
    GPIO_WriteBit(GPIOA, GPIO_Pin_1, Bit_SET);
    mcu_delay_us(1000);
    if (rf_powerup_reg() != 0)                    //对芯片进行上电操作
    {
        printf("rf power up fail\n");
    }

    spi_read_reg(16138, &data, 1);
    //printf("data = [%x]\n", data);    //如果data=0x0f,表示spi read正常
    spi_write_reg(16138, 0x08);
    mcu_delay_us(1000);
    spi_read_reg(16138, &data, 1);
    //printf("data = [%x]\n", data);    //如果data=0x08,表示spi write正常
} « end rfid_spi_test »
```

main 函数中调用 rfid_spi_test 即可，如下图：

```
int main(void)
{
    mcu_init();           //MCU硬件相关初始化
    rfid_lib_init();      //RFID lib库初始化
    printf("Board Initial OK!!!\n");
    mcu_adc_init();       //ADC初始化
    printf("Parameter Initial OK!!!\n");
    mcu_ext_int_init(); //外部中断初始化
    sw_timer_init();      //定时器初始化
    ucmd_reader_config_init();  //全局配置初始化
    rfid_spi_test();      //测试SPI与RFID芯片通信
    cmd_parser();         //命令解析
}
```

5. 使用 GD 库 SPI 操作时需要注意的问题

SDK 软件中使用的是 ST 的库，操作 SPI 时，直接操作相关寄存器，没有使用 ST 封装 SPI 的相关函数，如下图：

```
/* SPI发送数据 */
static uint16_t spi_send_data(uint16_t data)
{
    while ((SPI1->SR & SPI_I2S_FLAG_TXE) == (uint16_t)RESET);
    SPI1->DR = data;
    while ((SPI1->SR & SPI_I2S_FLAG_RXNE) == (uint16_t)RESET);
    return (SPI1->DR);
}
```

如果使用 GD 的库，不要直接使用 GD 封装 SPI 的相关函数，自己重新封装函数，直接调用寄存器操作。

注意：调用 GD 封装的 SPI 函数，会导致读取 RFID 芯片寄存器耗时较长，可能会影响标签盘点流程中的时序。

6. 天线检测问题：只连接了一根天线，上位机检测显示连接了 4 根天线。

```
#ifdef PROD_ZRM2004SV1
double g_threshold = 1.0f;
#elif PROD_ZRV
double g_threshold = 3.0f;
#elif PROD_ZRM2004SV2
double g_threshold =-5.6f;
#endif
```

可以调整天线检测阈值 g_threshold，根据新硬件，重新进行适配。

7. 板子遇到 q 值不收敛问题，请问有什么优化建议吗？软件上如果没有问题，硬件上有什么需要检查和注意的吗？

可能是通道的性能比较差，通过两种方式可以确认下：

第一种是用仪器测试下下在最大输出功率下的灵敏度

第二在最大输出功率下的接收口的自干扰功率大小

8. select 命令的最小参数长度为什么是 8，协议中是 9 个字节？

```
// ISO18000-63/EPC commands
const ucmd_item_t g_arr_cmds_iso[] =
{
    {UCMD_CMD_SETISO_COILPRM,          2,  UCMD_OPT_FIXLEN,    ucmd_set_iso_coil_handler},
    {UCMD_CMD_GETISO_COILPRM,          0,  UCMD_OPT_FIXLEN,    ucmd_get_iso_coil_handler},
    {UCMD_CMD_INVENTORYISO_CONTINUE,5,UCMD_OPT_FIXLEN | UCMD_OPT_AIR,       ucmd_inventory_iso_continue_handler},
    {UCMD_CMD_INVENTORYISO_STOP,    0,  UCMD_OPT_FIXLEN,                        ucmd_null_handler},
    {UCMD_CMD_READISO_TAG,             9,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_read_iso_tag_handler},
    {UCMD_CMD_WRITEISO_TAG,            10, UCMD_OPT_AIR,                       ucmd_write_iso_tag_handler},
    {UCMD_CMD_LOCKISO_TAG,             6,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_lock_iso_tag_handler},
    {UCMD_CMD_KILLISO_TAG,             4,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_kill_iso_tag_handler},
    {UCMD_CMD_SELECTCMDISO_TAG,     3,  UCMD_OPT_AIR,                          ucmd_select_iso_tag_handler},
    {UCMD_CMD_ISO_MIN_SENSE,           4,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_min_sens_handler},
    {UCMD_CMD_MULTI_SET_RF_PRM,     6,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_multi_set_rf_prm_handler},
    {UCMD_CMD_MULTI_GET_RF_PRM,     1,  UCMD_OPT_FIXLEN | UCMD_OPT_AIR,     ucmd_multi_get_rf_prm_handler},
    {UCMD_CMD_MULTI_SET_SELECT,     8,  UCMD_OPT_NULL,                      ucmd_multi_set_select_handler},
    {UCMD_CMD_MULTI_GET_SELECT,     1,  UCMD_OPT_FIXLEN,                    ucmd_multi_get_select_handler},
    {UCMD_CMD_MULTI_SET_QUERY,      4,  UCMD_OPT_FIXLEN,                    ucmd_multi_set_query_handler},
    {UCMD_CMD_MULTI_GET_QUERY,      1,  UCMD_OPT_FIXLEN,                    ucmd_multi_get_query_handler}
};
```

1.3.3.13 HSURM_SET_SELPRM (设置 Select 指令参数)

该命令用于设置模块的指定协议的Select指令参数，包括“MemBank”、“Target”、“Action”、“Pointer”、“Length”、“Mask”、“Truncate”。各参数的具体应用及含义详见协议说明

➢ **命令格式**

HEAD	FC	IL	PAYLOAD								CHECK
			Protocal	Target	Truncate	Action	Membank	Pointer	Length	Mask	
0xBD	0x0087	1Byte	1Byte	1Byte	1Byte	1Byte	1Byte	2Bytes	1Byte	N-Bytes	1Byte

最少是 8 个字节，只要大于等于 8 字节都是合法的，select 有可能是不带 mask。

9. 标签命令里面 UCMD_OPT_AIR 表示主时钟 100Mhz，是什么含义？

这个宏最早做芯片研发的时候定义的，现在已经不用了，除了那个固定参数的宏，用于接收命令长度检查，其他的没有实际意义的。

10. 目前 inventory 只有 RSSI，是否支持相位的识别？

目前这个芯片不支持相位，下一版芯片会支持

11. 上位机上如何配置 session？

盘点标签页面下进行设置，如下如：

红色框里的盘点，默认 session 为 0，如下图

12. 6C 协议里面只有一个 preamble，SDK 中 dig_set_long_preamble 和 dig_set_short_preamble 长前导和短前导指的是什么？

```
ret_ctrl_e iso_query_command(query_param_t *param)
{
    uint8_t cmd_code=0;
    cmd_buf_t txcmd;
    ret_ctrl_e ret;

    dig_set_fifo_max_level(8);
    dig_set_rxlen(16);
    dig_auto_rx(1);
    txcmd.bits_cnt=0;
    txcmd.bytes_cnt=0;
    cmd_code=0x08;
    common_make_bits_sequence(&txcmd,(uint8_t *)&cmd_code,4);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->dr,1);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->blcoding,2);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->trext,1);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->sel,2);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->session,2);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->target,1);
    common_make_bits_sequence(&txcmd,(uint8_t *)&param->Q_value,4);

    dig_crc_ctrl(RF_CRC_TX5);

    dig_clr_fifo();
    dig_set_long_preamble();
    dig_start_send();
    dig_set_txlen(txcmd.bytes_cnt*8+txcmd.bits_cnt);
    ret=common_fifo_multi_bytes_write(txcmd.cmd_buf,txcmd.bytes_cnt*8+txcmd.bits_cnt);
    if(ret)
    {
        return ret;
    }
    ret=rfid_irq_wait_resp(g_rdsm_status,RF_INT_STDCMD_OVER);
    if(ret)
    {
        return ret;
    }
    dig_set_short_preamble();
    return RFID_RET_SUCCESS;
} « end iso_query_command »
```

query 命令前面的部分叫长前导码，其他命令的头部都是短前导码。

dig_set_long_preamble 设置长前导码后，通过 dig_start_send 发送出去，dig_set_short_preamble 再还原设置为短前导码(长前导和短前导是控制同一个寄存器，所以需要设置回去)

13. rf_auxadc_lna_pkdet_result 回波损耗实现机制？

回波检测有两种机制：

内部实现：通过配置芯片寄存器，由芯片检测的，也就是通过 rf_auxadc_lna_pkdet_result 实现。

外部实现：通过 mcu 的 adc 计算来的（SDK 中没有实现）。

具体使用那种方式取决于你们采用那种模块型号的参考设计。

内部的不是对数线性，范围也不够宽，精度和外部比还是有点差距。

14. 不接天线时候，回波损耗为什么很大？

不接天线的话回波损耗绝对值会比较小，问题原因可能是板子上的功率检测器的测量电路有问题。

15. 关于回波损耗的计算、回波损耗在什么范围内，说明阻抗匹配是合适的？

```
/* 回波损耗测试,使用芯片内部adc */
static int16_t ucmd_reader_backpower_test(uint8_t count)
{
    float backpower=0.0,tmp=0.0;
    uint8_t power = g_rf_config_parameter.txpower;
    float txpower=(float)power;
    int16_t data=0xffff;

    rf_rfpll_freq_cfg(g_tmp_freq);                  //设置锁相环的工作频率
    power_txpower_cfg(power);                       //设置功率
    rfid_cw_on();                                   //开启载波
    rf_auxadc_sample_enable();                      //使能auxadc，用于jammer测试
    backpower=rf_auxadc_lna_pkdet_result(count); //通过芯片adc测试反射功率
    if(abs(backpower)<40.0)
    {
        tmp=(txpower-(backpower+10.0))*10.0;
        if(tmp<0)
        {
            tmp=tmp-0.5f;
        }
        else
        {
            tmp=tmp+0.5f;
        }
        data=(int16_t)tmp;
    }
```

1）红框中的+10，这个公式是拟合公式，10 是根据芯片特性验证出来的固定参数。

2）回波损耗越大越好，具体是一个相对的范围（驻波比，尽量小，回损尽量大，这样反射回来能量就少。接收效果也好）

3）回波损耗和天线环境有关系，测试时，使用相同的天线进行测试，回波损耗值越大，效果越好。

16. 在关闭载波状态 standby 模式和 shutdown 模式 功耗差距大概有多少？

一般 shutdown 模式下，我们给芯片断电，即功耗基本为 0，standby 模式下功耗大概是 600mw~800mw

17. 关于芯片的功耗模式说明一下？

芯片分为三个模式：standby，txrx，shutdown

txrx 模式：在该模式下进行盘点即标签操作，功耗会高

standby 模式:该模式会关闭芯片内部分电路，sjc，模拟电路等。在该模式下部分配置依然保存

shutdown 模式：给芯片断电，所有配置，上电后需要重新配置

三个模式的功耗情况：txrx > standby > shutdown

状态切换示例代码

从 shutdown 到 txrx：

```
rfes_shutdown_to_standby();
rfes_standby_to_txrx();
dig_base_init();
rf_sjc_init();
rf_base_dc_meas();
ucmd_reader_config_set();
rfes_curr_status_set(RFES_STATUS_TXRX);
```

从 txrx 到 shutdown：

```
rfes_standby_to_shutdown();
rfes_curr_status_set(RFES_STATUS_SHUTDOWN);
```

从 standby 到 txrx：

```
rfes_standby_to_txrx();
rf_sjc_init();
rf_base_dc_meas();
power_txpower_cfg(power_get_cur_pwr());
fhss_hopping_freq_config();
rfes_curr_status_set(RFES_STATUS_TXRX);
```

从 txrx 到 standby：

```
rfes_txrx_to_standby();
rfes_curr_status_set(RFES_STATUS_STANDBY);
```

18. 关于温度检测和功率补偿的校准方案

基准温度为 36 度
36 - 71 度，每增加 4 度， scal 增加 1
大于 71 度，每增加 2 度， scal 增加 1
36 度以下，每降低 5 度 scal 减 1

我们是这么做的，实际中也不是很准确，仅供参考。

19. 下面红框中的代码是否可以去掉？

```
l152:            break;
l153:        default:
l154:            if(common_status_outtime_check(12000))
l155:            {
l156:                g_rdsm_status=MAIN_SM_STATUS_END;
l157:            }
l158:            //printf("default Status=%d\n",g_rdsm_status);
l159:            break;
l160:        } «end switch g_rdsm_status »
l161:    } «end while 1 »
```

不能去掉，如果去掉，当状态修改为中断程序执行的状态，而中断不来的情况下，程序会死锁，无法退出。这是规避一些异常情况的。正常情况下不会不来中断。

20. SDK 接口说明文档中，dig_isr_reg_clean、dig_data_isr_clean、dig_data_isr_manul_clean，三个函数的使用有什么区别？

dig_isr_reg_clean
void dig_isr_reg_clean(void)
清除中断寄存器，中断掩码寄存器，中断数据寄存器
无

dig_isr_reg_clean， 一般初始化的时候需要把所有中断清除一下

void dig_data_isr_clean(void)
清除中断数据寄存器

dig_data_isr_clean 盘点过程中，发送命令、接收数据都都需要清除的，这个等同于 dig_data_isr_get（）

void dig_data_isr_manul_clean(void)

清除中断寄存器

dig_data_isr_manul_clean 清除中断寄存器，一般强制结束盘点时，清除一下

21. 触发 common_inventory_statis_outtime_check 的超时保护之后，QueryRep 帧可以正常发送，但是无法接收到芯片的 RN16 回复，请问触发超时后，在不重启 Query 的前提下，如何操作才能使芯片恢复正常呢？

common_inventory_statis_outtime_check 里面检查有两个目的：

1.当一定时间内没有盘点的新标签，就结束盘点。目的是提高盘点效率，没有必要按照碰撞算法一直到盘点结束。

2.限制了每一个 query 周期的最大时间，就是担心再某些情况下碰撞算法跑飞，永远出不来。

从上面的目的来讲，我们都是加强系统的稳定性的。但这些功能不是必要的。如果觉得多余或者误导你们的思路可以去掉。去掉也不影响正常盘点。

超时后怎么操作能恢复正常？我们认为，既然已经出现超时，那我们就需要重新跳频，重新初始化，开始下一个 query 周期。如果你不想发 query 只是想继续盘点，那这个功能就没有必要用了，可以直接删除掉。你非要再超时的时候想办法恢复继续盘点，那需要检查一下，为什么会超时？根据超时的原因去解决。超时的原因，有多种，可能是碰撞算法异常，环境改变等等。

22. 功率配置的三个参数的取值范围以及推荐取值区间？

```
rf_pa_gain_cfg(5);
rf_modulator_gain_cfg(2,7);
dig_dbb_gain_cfg(83);
```

tx_mod_gain 取值如下：

```
tx_mod_gain=0    max-14   db
tx_mod_gain=1    max-8    db
tx_mod_gain=2    max-6    db
tx_mod_gain=3    max-3    db
tx_mod_gain=4    max-6    db
tx_mod_gain=5    max-3    db
tx_mod_gain=6    max-2    db
tx_mod_gain=7    max(8)   db
*/
```

tx_pa_gain 取值如下：

```
/*
 tx_pa_gain=0    max-24    db
 tx_pa_gain=1    max-18    db
 tx_pa_gain=2    max-12    db
 tx_pa_gain=3    max-6    db
 tx_pa_gain=4    max-3.5    db
 tx_pa_gain=5    max(12.5)    db
*/
```

scale 值对增益的影响关系如下

Y = (scale/128) * x

y 为输出的数字信号幅值

x 为数字信号原始幅度

scale 的值不要超过 140，否则会波形畸变

23. SDK 接口对于 Q 值传参检查数值为 0 或者大于 15，Q 设置为 4；Q 值不可以设置为 0 吗？

Q 值可以设置为 0，Q 值为 0 应用于单张标签的场景。

24. 一条 read 指令，在默认配置下（FCC 下），理论最多可返回 50 条 read 的结果，这样设计是否合理，是否我们的固件设计里，可以尝试最多读取 50 次，如果 50 次全部失败，返回错误，如果有一次成功就停止反复读取，这样设计更合理一些，否则用户使用的时候，一条指令的响应会是长度不定、次数不定。

我们当前对读的设计是，指定单张标签读取时，如果失败，会跳频继续读，直到一次读取成功，整个读操作退出。如果未指定标签读取，也就是多标签读取时，不管标签成功或者失败，都会继续下次读取，直到所有频点都跳一次，而且标签信息不会重复上报，固件中有过滤机制。

25. read 读取响应中 STATUS 和 TagStatus 有什么关系？

- 响应格式及状态字节

HEAD	FC	IL	STATUS	PAYLOAD								CHECK
				TagStatus	Antenna	CRC	PC	UIILength	UII	WordCount	Data	
0xBD	0x005E	1Byte	1Byte	1Byte	1Byte	2Byte	2Bytes	1Byte	N Bytes	1Bytes	N-Byte	1Byte

STATUS：　表示模块成功接收到标签响应数据
TagStatus：　表示标签返回的操作状态

只有上面 2 个标志同时成功，才表明读取成功。

26. dig_set_fifo_min_level(3)表示什么含义？

```
ret_ctrl_e common_fifo_multi_bytes_write(uint8_t *p_buf, uint16_t bitcnt)
{
    uint16_t bytecnt;
    uint16_t cnt = 0;
    ret_ctrl_e ret;

    bytecnt = (bitcnt+0x07) >> 3;

    if(bytecnt>RF_FIFO_MAX_WRITELEN)
    {
        dig_data_isr_manul_clean();
        dig_set_fifo_min_level(3);
        dig_intr_mask(RF_INT_MASK&(~(RF_INT_FIFO_ALMOST_EMPTY_EDGE_PL)));

        while(bytecnt > RF_FIFO_MAX_WRITELEN)
        {
            dig_fifo_write(&p_buf[cnt], RF_FIFO_MAX_WRITELEN);
            ret =rfid_irq_wait_resp(g_rdsm_status,RF_INT_FIFO_ALMOST_EMPTY_EDGE_PL);
            if (ret)
            {
                return ret;
            }
```

芯片的发送机，从 FIFO 中取数据，当取的小于等于 3 个字节时，发出将要空的中断（RF_INT_FIFO_ALMOST_EMPTY_EDGE_PL），提示应用层继续给 FIFO 中写入数据。

27. 以下各种情况下如何产生外部中断？

1. 在合适的时间范围内，单一标签返 RN16,且射频芯片接收到正确的 RN16;
   RN16 成功接收到，会产生接收完成中断（RF_INT_RX_END_PL）

2. 在合适的时间范围内，由于标签冲突，多个标签同时返回 RN16,导致无法解出任何 RN16;
   rn16 无法解析，会产生 preamble 错误中断（RF_PREAMBLE_CHECK_FAIL_PL）或 pilot 错误中断（RF_INT_PILOT_FAIL_PL）

3. 在合适的时间范围内，尽管标签冲突，但是射频芯片仍然解出一个错误的 RN16;
   rn16 如果能成功解析出数据，这个时候无法判断 RN16 的正确性，只能发送 ACK 走下面的流程，如果 RN16 不正确，标签不会回应 EPC

4. 场内无标签，rx 端信号幅值均在解调阈值之下，相当于在有效的返回时间段内无 RN16 信号；
   rn16 空闲情况下，会产生 preamble 错误中断（RF_PREAMBLE_CHECK_FAIL_PL）或 pilot 错误中断（RF_INT_PILOT_FAIL_PL）

5. 射频芯片向空口发送 query 失败；

等待发送结束中断（RF_INT_STDCMD_OVER），如果发送 query 失败，等待会出现超时错误

6. 是否还有其他情况下返回外部中断呢？

发送完成中断 RF_INT_STDCMD_OVER

接收完成中断 RF_INT_RX_END_PL

FIFO 将空中断 RF_INT_FIFO_ALMOST_EMPTY_EDGE_PL

FIFO 将满中断 RF_INT_FIFO_ALMOST_FULL_EDGE_PL