BC3602 开发板应用范例

文件编号：AN0542SC

简介

Holtek 推出双向无线 FSK/GFSK 高效能射频芯片 BC3602，适合在 1GHz 以下免执照 ISM Band(300MHz~960MHz)应用；IC 整合高功率 PA、频率合成器及数字解调功能，精简外围电路，射频特性符合 ETSI/FCC 规范。

BC3602 工作电压为 2.0V~3.6V，可程序设定发射功率，最高达+13dBm；高灵敏度接收能力，最高传输速率达 250kbps；具 ATR 自动收发(Auto Transmit Receive)功能，内置高精度低功耗振荡器及省电模式自我唤醒收发功能，适合低功耗电池及 IoT 产品需求，可广泛应用于智能居家/安防、汽车防盗器、工业/农业控制器等等无线双向应用产品。

本篇将介绍使用 BC3602 API 在 M0+系统开发板上进行功能操作。文中将介绍如何架设环境和程序编译，而所附范例程序，可让用户了解 BC3602 所提供多样 RF 接收/发射功能；使用者可通过本文介绍，选择适合自己应用情境，进而开发出无线产品。

开发平台说明

开发平台使用 Holtek 32-bit Arm® Cortex®-M0+ Microcontroller：HT32F52352 为主控制器，系统开发板 BCE-GENTrx32-001，搭配上 BC3602 模块化开发板 BCT-360x-001+BCM-3602-X01，可组成完整平台结构如下图所示。

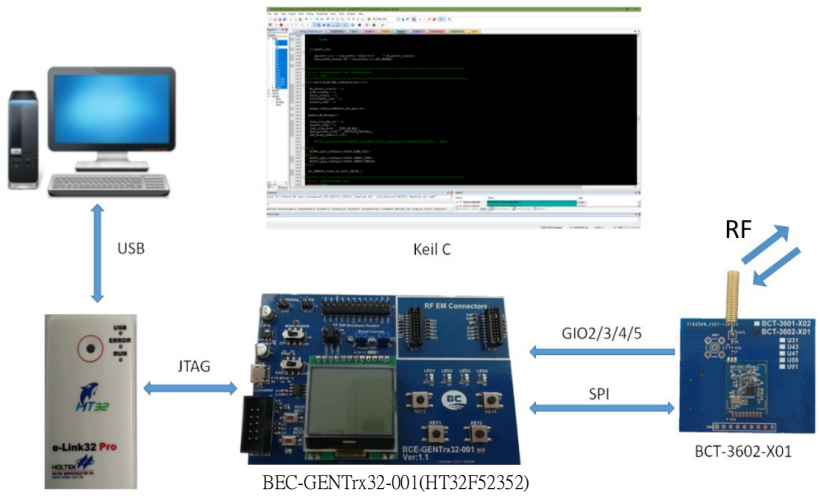

图 1. 系统架构图

HT32F52352 是 Arm® M0+微控制器，用户需要先在计算机上安装 keil uVisionIDE 接口，搭配 Holtek 所开发 e-Link32 pro，通过 JTAG 接口，对 HT32F52352 编辑程序(F/W)，进一步了解 HT32F52352 请参考网址： https://www.holtek.com.cn/productdetail/-/vg/HT32F52342-52。

系统开发板介绍

BCE-GENTrx32-001 开发板提供了良好操作接口，方便用户来操作，如下图所示。

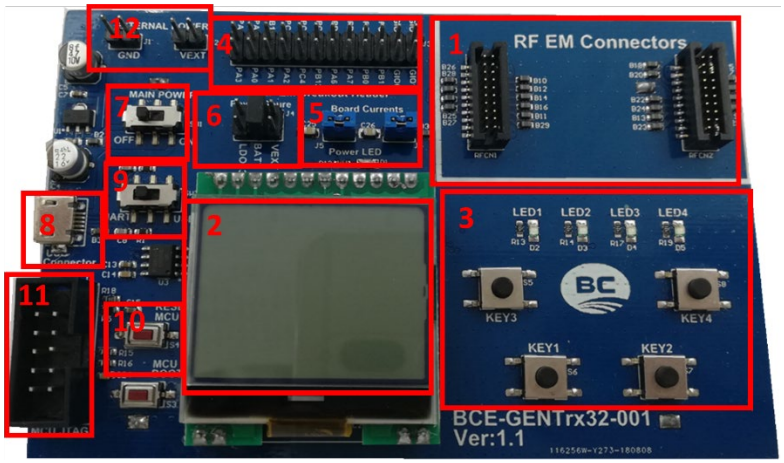

图 2. 系统开发板组成

包含了：

1、射频模块接口是射频发射/接收装置连接处，此范例则安装上 BCT-360x-001+BCM-3602-X01。

图 3. BCT-3602-X01 板

2、液晶显示器(支持 128×64 点)，使用方法请参考本文范例程序，内含显示器控制程序库。

3、指示灯：LED×4 及按键×4，用户在程序开发时，可利用其作指示及输入功能。在本范例程序中按键为 KEY2=Enter 功能，其余由用户定义。

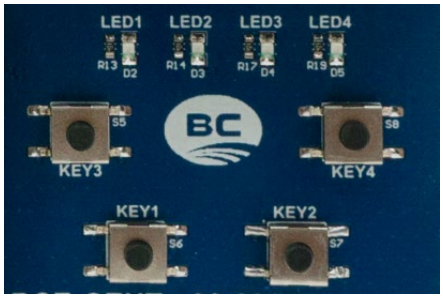

图 4. LED & KEY

4、I/O 接口包括 HT32F52352 部份 I/O 及 BC3602 的 GIO3 跟 GIO4，详细 I/O 如下图所示。

BCE-GENTrx32-001	BCM-3602-X01
PA3	CSN
PA0	SCLK
PA1	SDIO
PA2	GIO1
PC4	GIO2
GPIO2	GIO3
GPIO3	GIO4

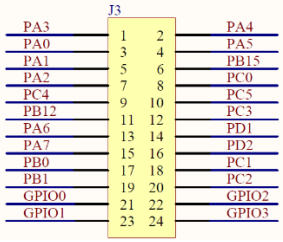

图 5. I/O 接口

5、MCU 和 BC3602 模块板电源电流检测点。

6、系统电源选择，如图 6，跳线(Jumper)于左侧 LDO33 处，表示电源由 USB 口输入；跳线于中间 BATT 处，代表电源使用电池座(板背：两颗 1.5V AA 电池)；跳线于右侧 VEXT 处，则是使用外部电源接点(如图 6)供电，注意若是使用外部电源接点输入电压，电压不得超 3.6V。

图 6. 电源选择

7、总电源开关，左拨(OFF)为关闭电源，右拨为开启电源。

图 7. 电源开关

8、Micro USB 接口，可用来作为系统电源输入(电源选择 LDO33)。

9、UART/USB 接口选择。

10、系统复位键。

11、JTAG 接口，可配合 IDE 接口仿真程序及下载程序用。

12、外部电源接点。

图 8. 外部电源连接

范例程序使用方式介绍

BC3602 范例程序目前是使用 Keil C 这套软件开发，以下将介绍如何使用 Keil C 软件去编译与下载 BC3602 范例程序。

挑选范例程序

建置在 BCE-GENTrx32-001 开发平台上 BC3602 范例程序共有 6 种分别为：TX Carry、Simple FIFO、Extend FIFO、PER Mode、ATR Mode 和 ARK Mode，有关各操作模式详细描述可参考 BC3602 Datasheet。

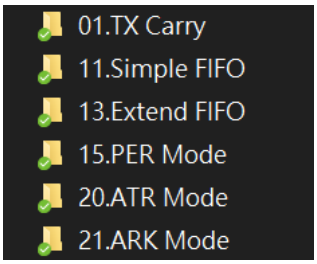

图 9. 范例程序

1、TX Carry：该程序提供单一 RF 载波(无数据与调变)输出。

2、Simple FIFO：该程序分为 TX(传送)与 RX(接收)，提供使用者传送/接收 1~64 byte 长度数据。

3、Extend FIFO：该程序分为 TX(传送)与 RX(接收)，提供使用者传送/接收 1~255 byte 长度数据。

4、PER Mode：该程序分为 Master(主端)与 Slave(从端)，此程序建立在 Simple FIFO 上做双向收发，提供使用者传送和接收 1~64 byte 长度数据。

5、ATR Mode：该程序分为 WOT(间歇传送)与 WOR(间歇接收)，提供使用者间歇自动传送/接收 1~64 byte 长度数据而不需要 MCU 介入。

6、ARK Mode：该程序分为 ARS(自动重传)与 AAK(自动应答)，提供使用者自动重传(无数据)/自动应答(无数据)而不需要 MCU 介入。

硬件安装

需准备 1 台 e-Link32 Pro、2 块 BEC-GENTrx32-001 开发板和 2 块 BCT-3602-X01 模块，分别将 BCT-3602-X01 模块安装至 BEC-GENTrx32-001 开发板上，并把 e-Link32 Pro 连接头安装至 BEC-GENTrx32-001 开发板后开机等待烧录。

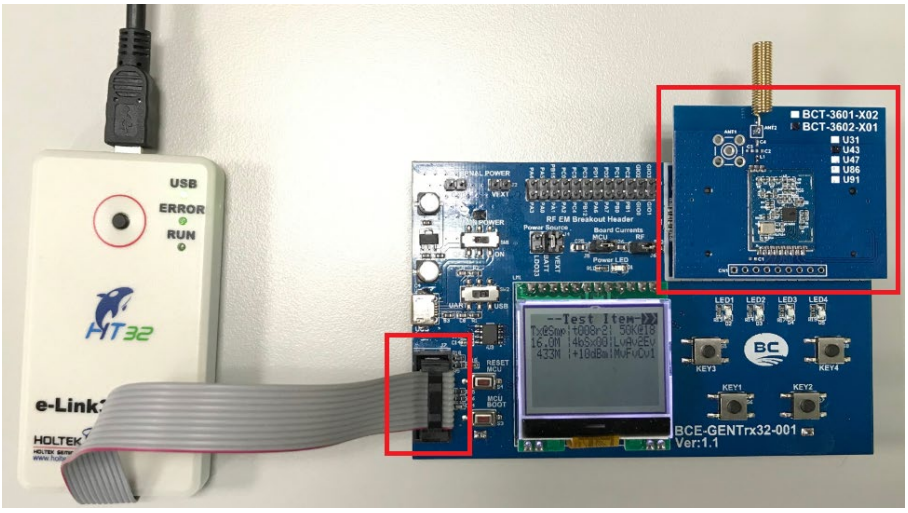

图 10. 安装硬件示意图

关于 e-Link32 Pro 请至 Holtek 官网参考如何使用：https://www.holtek.com.cn/ice。

烧录流程

开启程序项目文件，请使用所附范例程序下..\BC3602_DemoCode_M0+\Example Code\15.PER Mode\Master\BC3602_DemoCode_M0+.uvproj (以 PER 作为范例)后，依下列顺序操作：

1、点选“Project\Options for target 'xxxxx' ”，或点击如下图所示，开启配置窗口。

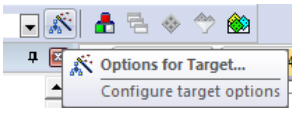

图 11. 项目设定示意图 1

2、开启“除错(Debug)”分页。

3、在连结设备选择“CMSIS-DAP Debugger”。

4、点选“设定(Settings)”开启链接设备设定窗口。

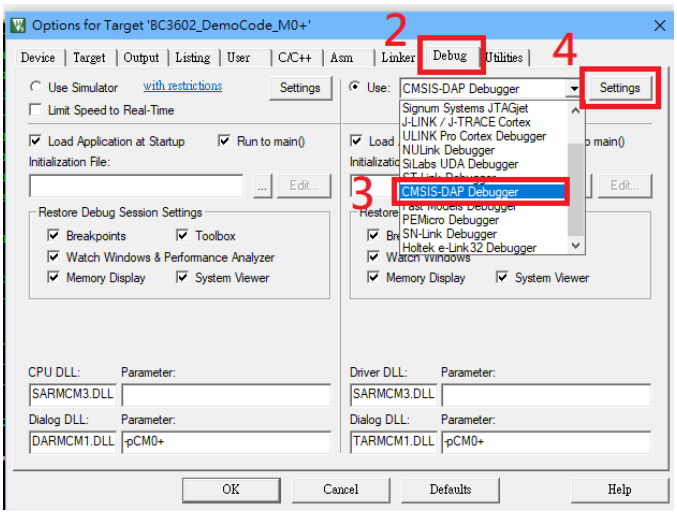

图 12. 项目设定示意图 2

5、开启“除错(Debug)”分页。

6、除错接口适配器请选择“Holtek CMSIS-DAP”。

7、勾选“SWJ”并选择“SW”。

8、检查是否有读到 IDCODE，如果没有出现，请检查 USB 线连接或是 e-Link 驱动程序。

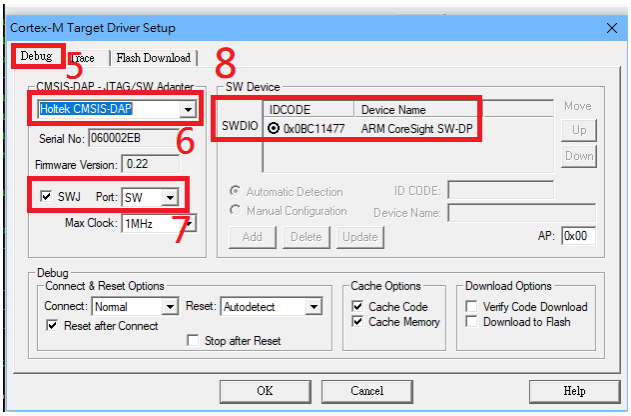

图 13. 项目设定示意图 3

9、点选“Rebuild”重新编译。

10、点选“Download”下载程序至 BEC-GENTrx32-001 开发板。

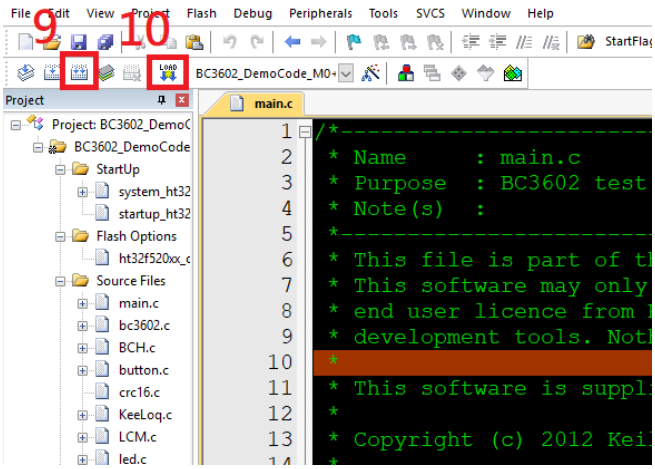

图 14. 项目设定示意图 4

11、 在范例程序中 15.PER Mode 下有 Master 与 Slave 两个文件夹，将其分别烧录至两张开发板后，重新上电，其中 a 和 b 会显示各程序参数状态、按下 c 和 d 则可以代表开始与停止。如下图，可以看到分别有左边为 Master 模式与右边为 Slave 模式板子，Master 板开机后按下 KEY2 后则会传输 N 次封包(再按一次将会停止)，面板上 TX 计数器会显示目前发射封包次数；Slave 板开启电源后无需按下任何按键即处在 RX 模式等待接收信号。当按下 Master 板子上按键后，Slave 面板上 RX 计数器将会显示接收到多少封包，并且回复对应封包给 Master。

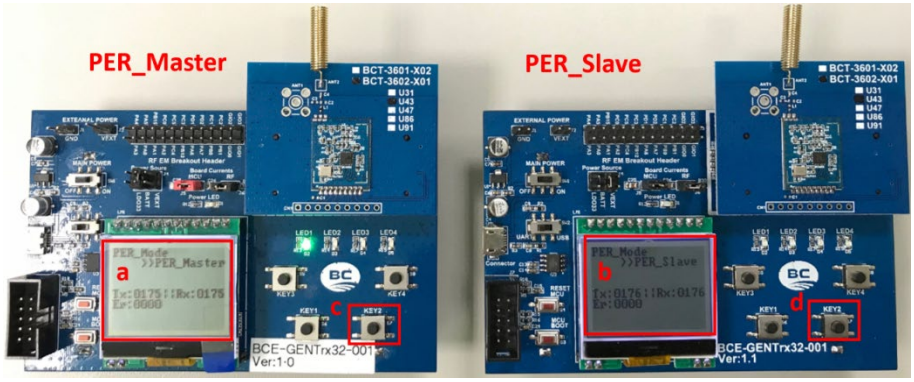

图 15. PER 操作示意图

范例程序简介

此范例程序分为 6 类共 11 支程序，接下来我们将分别介绍各范例所使用功能。主程序及各个执行项目流程，我们将分开说明。

挑选参数

```
//********************MAIN MENU**********************

#define DEFAULT_RF_Band           _RF_Band_433MHz_        //433MHz
#define DEFAULT_RF_Frequency      433.92                  //433.92MHz
#define DEFAULT_TX_Preamble       4                       //1~256       4 = 4 byte
#define DEFAULT_RX_Preamble       _RX_Preamble_2B_        //2 byte
#define DEFAULT_SyncWidth         _Syncword_4B_           //4 byte
#define DEFAULT_DeviceID          0xab00111111            //0~0xFFFFFFFFFF //40bit
#define DEFAULT_TX_Power          _TX_Power_13dBm_        //+13 dBm
#define DEFAULT_DATA_RATE         _DataRate_50K_          //50K bps

#define DEFAULT_Man_EN            Disable                 //Manchester code enable
#define DEFAULT_FEC_EN            Disable                 //FEC enable
#define DEFAULT_CRC_EN            Enable                  //CRC field enable
#define DEFAULT_CRCFMT            1                       //CRC format selection
#define DEFAULT_Tra_EN            Enable                  //Trailer enable

#define DEFAULT_WHT_EN            Disable                 //Data whitening enable
#define DEFAULT_WHTFMT            0                       //whitening format selection
#define DEFAULT_WHTSEED           54                      //whitening Seed selection

#define DEFAULT_PLLEN_EN          Disable                 //Payload length field enable
#define DEFAULT_PLHAC_EN          Disable                 //Payload header address corre
#define DEFAULT_PLHLEN            Disable                 //1~2byte Payload header lengt
#define DEFAULT_PLH_EN            Disable                 //Payload header field enable

#define DEFAULT_PKT_Length        64                      //1~64

#define DEFAULT_PKT_Ext_Length    180                     //1~255
#define DEFAULT_Margin_Length     Margin_32byte           //FIFO length margin selection
//*****************************************************************
```

图 16. 参数清单

选择并开启范例程序后，打开程序 main.h 后可以看到 MAIN MENU 里面参数，这些参数与传输速度、距离、频段、链路强健性等等有关，使用者须理解这些参数间彼此影响才能挑选出较为适合使用参数。参数说明如下。

1. DEFAULT_RF_Band：射频频段选择，共有 315、433、470、868、915MHz 五种选项可以选择，并请按照模块板上所使用频段来设定。
2. DEFAULT_RF_Frequency：射频频率选择，需依照各个频段的区间来去设定频率。
3. DEFAULT_TX_Preamble：选择 TX 封包里 Preamble 数量(byte)，数量为 1~256 个 byte 可供选择。
4. DEFAULT_RX_Preamble：选择 RX 需要侦测封包 Preamble 数量(byte)，数量为 1、2、4 个 byte 三种可供选择。
5. DEFAULT_SyncWidth：Syncword 长度(byte)选择，共有 4、6、8 个 byte 可供选择。
6. DEFAULT_DeviceID：设备传输序列码(40-bit)，此序列码将会影响传收与接收配对，请务必将把要一起传送与接收设备设置为同一个序列码。
7. DEFAULT_TX_Power：RF 输出功率设定，有 0、+5、+10、+13dBm，四种可供选择。
8. DEFAULT_DATA_RATE：数据传输速率设定，有 2K、5K、10K、25K、50K、125K、

250Kbps，七种可供选择。

9. DEFAULT_Man_EN：Manchester 编码方式开关，功能详情请参照 Datasheet 或 API 说明章节。
10. DEFAULT_FEC_EN：FEC 前向纠错编码开关，功能详情请参照 Datasheet 或 API 说明章节。
11. DEFAULT_CRC_EN：CRC 开关，功能详情请参照 Datasheet 或 API 说明章节。
12. DEFAULT_CRCFMT：CRC 编码公式选择，功能详情请参照 Datasheet 或 API 说明章节。
13. DEFAULT_Tra_EN：Trailer 编码开关，功能详情请参照 Datasheet 或 API 说明章节。
14. DEFAULT_WHT_EN：WHT 功能开关，功能详情请参照 Datasheet 或 API 说明章节。
15. DEFAULT_WHTFMT：WHT 编码公式选择，功能详情请参照 Datasheet 或 API 说明章节。
16. DEFAULT_WHTSEED：WHT 编码种子，功能详情请参照 Datasheet 或 API 说明章节。
17. DEFAULT_PLLEN_EN：封包长度信息开关，功能详情请参照 Datasheet 或 API 说明章节。
18. DEFAULT_PLHAC_EN：Address0 功能开关，功能详情请参照 Datasheet 或 API 说明章节。
19. DEFAULT_PLHLEN：Address 长度选择，功能详情请参照 Datasheet 或 API 说明章节。
20. DEFAULT_PLH_EN：Header 开关，功能详情请参照 Datasheet 或 API 说明章节。
21. DEFAULT_PKT_Length：除了 Extend 和 Carry 模式外所有模式封包长度选择，长度可以设置为 1~64byte。
22. DEFAULT_PKT_Ext_Length：Extend 模式封包长度选择，长度可以设置为 1~255byte。
23. DEFAULT_Margin_Length：Extend 模式下 Margin 检测门坎，提供 4、8、16、32byte，四种可供选择。

主程序流程

1、初始化：这里分为两种初始化：MCU 功能初始化(CKCU、GPIO、LED、Button、LCM、BFTM、UART)和 BC3602 基本功能初始化(SPI、Crystal、VCO、LIRC、GIO Current…)。

2、等待计数器：主循环将会以 1 毫秒时间检测按钮是否有被改变，若是按钮有被改变，系统将会改变按钮状态，通知系统按钮被按下，以便下一个步骤判断。

3、读取按键：此步骤将会读取按键状态，范例程序只定义了 KEY2 作为开始与结束功能。

4、检查 IRQ 状态：这边将检查 BC3602 硬件 IRQ 引脚状态是否为低准位。如果为低准位，表示现在 IRQ 旗标立起来，将目前 IRQ 状态储存到寄存器并将 BC3602 IRQ 状态复归。

5、程序主循环(BC360x Program)：根据程序内状态执行该通信状态。可选状态有 TX Carry 模式、Simple FIFO 模式、Extend FIFO 模式等等，文章后面会为您一一介绍。

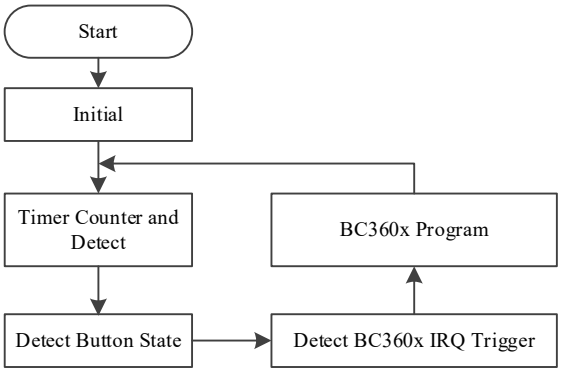

图 17. 主程序流程图

TX Carry 模式

该程序提供单一 RF 载波(无数据与调变)输出，此程序将会持续输出载波直到主循环将它停止。

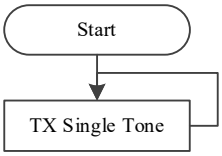

图 18. TX Carry 流程图

Simple FIFO 模式

该程序提供封包 1~64 byte 数据长度供用户传送/接收，此程序将会持续传送/接收直到主循环将他停止。

- **TX 模式**

Simple FIFO TX 模式流程主要分为两个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ、封包长度等信息，复位 TX FIFO 指针后将数据写入 TX FIFO 中并且开启 TX 传送数据。

步骤 1：等待 TX 完成后 IRQ 信号进来，并复位 TX FIFO 指标回到步骤 0 执行 TX。

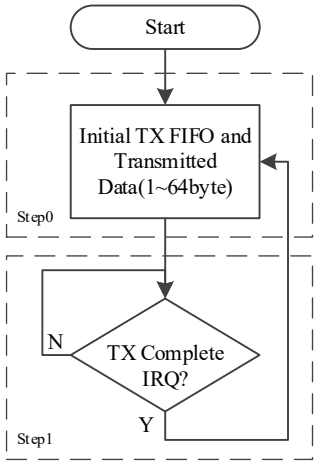

图 19. Simple FIFO_TX 流程图

- **RX 模式**

Simple FIFO RX 模式流程主要分为两个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ 和复位 RX FIFO 指标，并且开启 RX 聆听数据。

步骤 1：等待 RX 接收完成/失败 IRQ 信号进来，并开始判断收进来数据是错误还是正确，但无论正确与否，都会将 RX FIFO 指标清空，并开启 RX 模式继续执行。

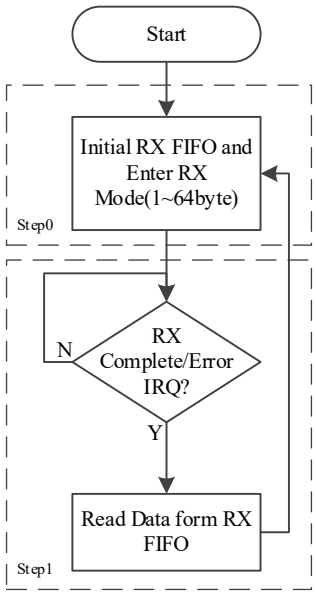

图 20. Simple FIFO_RX 流程图

Extend FIFO 模式

该程序提供封包 1~255 byte 数据长度供用户传送/接收，此程序将会持续传送/接收直到主循环将他停止。

- **TX 模式**

Extend FIFO TX 模式流程主要分为三个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ 和 Margin 长度等设定。

步骤 1：复位 TX FIFO 指针，并且开启 TX 传送数据。

步骤 2：等待 TX 完成和 Margin IRQ 信号进来；若是 TX 完成 IRQ 立起，程序将回到步骤 1 执行；若是 Margin IRQ 立起，代表 MCU 需要进一步再填入 FIFO 数据，并再次传送数据直到 TX 完成 IRQ 立起。

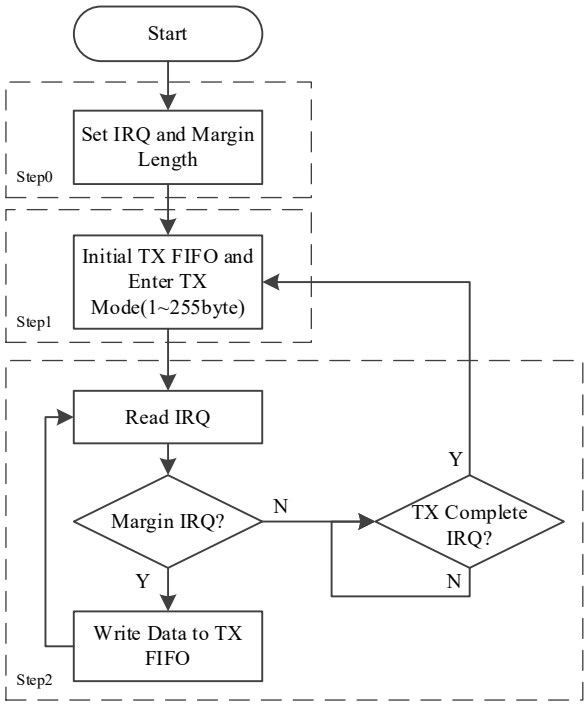

图 21. Extend FIFO_TX 流程图

- **RX 模式**

Extend FIFO RX 模式流程主要分为三个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ 和 Margin 长度等设定。

步骤 1：复位 RX FIFO 指针并且开启 RX 聆听数据。

步骤 2：等待 RX 完成/失败和 Margin IRQ 信号进来；若是 RX 完成/失败 IRQ 立起，程序将回到步骤 1 执行；若是 Margin IRQ 立起，代表 MCU 需要进一步再读取出 FIFO 数据并再次聆听数据直到 RX 完成/失败 IRQ 立起。

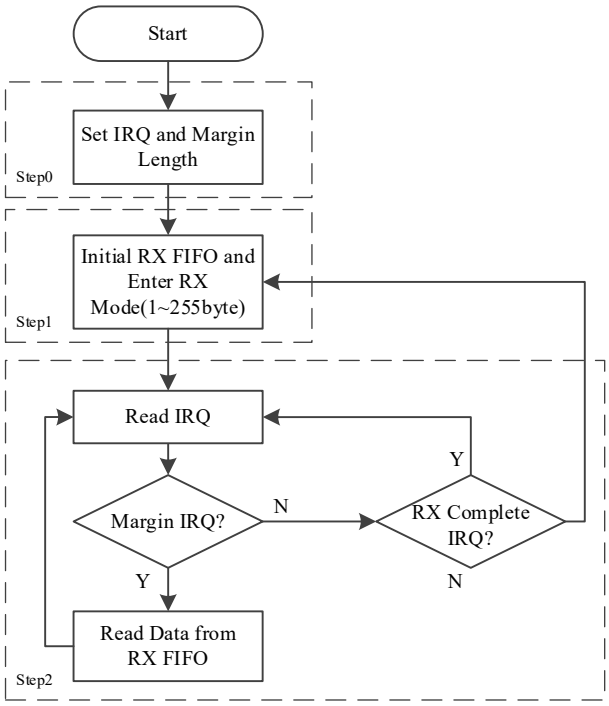

图 22. Extend FIFO_RX 流程图

PER 模式

该程序提供封包 1~64 byte 数据长度供用户双向收发(建立在 Simple FIFO 模式下)，此程序将会持续双向收发，直到主循环将他停止。

- **Master 模式**

PER_Master 模式流程主要分为三个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：开启 TX 完成、RX 接收和 RX 接收失败 IRQ 功能，设置封包数据与长度并且将 TX、RX FIFO 指标复位后，开启 TX 功能将数据传出。

步骤 1：等待 TX 完成 IRQ 信号进来后，马上复位 RX FIFO 指针，并且开启 RX 模式收取数据。

步骤 2：等待 RX 接收完成/失败 IRQ 信号进来，并开始判断收进来数据是错误还是正确，但无论正确与否，都会将回到步骤 0 执行。

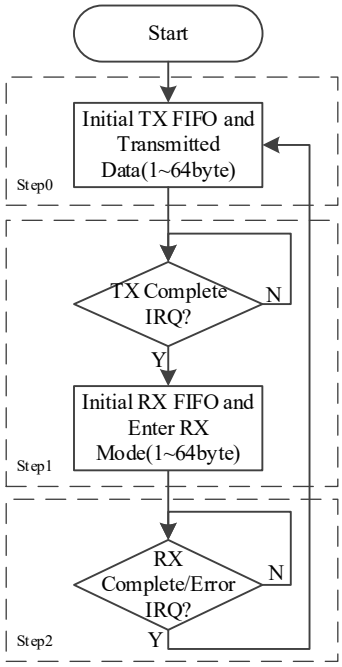

图 23. PER_Master 模式流程图

- **Slave 模式**

PER_Slave 模式流程主要分为四个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：开启 TX 完成、RX 接收和 RX 接收失败 IRQ 功能，将 RX FIFO 指针复位后，开启 RX 功能等待数据进来。

步骤 1：等待 RX 接收完成/失败 IRQ 信号进来，并开始判断收进来数据是错误还是正确，但无论正确与否，都会到下个步骤。

步骤 2：设置封包数据与长度，并且将 TX FIFO 指标复位，并开启 TX 功能将数据传出。

步骤 3：等待 TX 完成 IRQ 信号进来后，将回到步骤 0 执行 RX 功能。

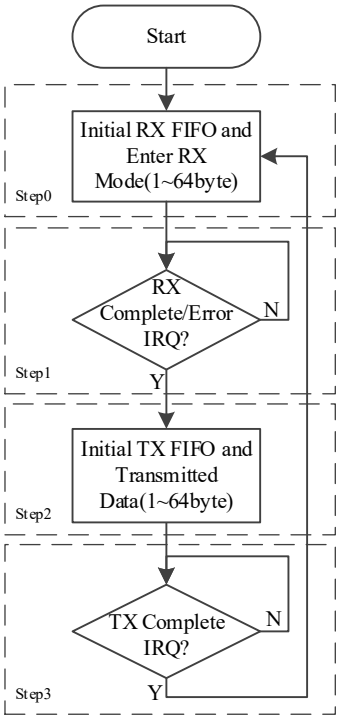

图 24. PER_Slave 流程图

ATR 模式

该程序提供封包 1~64 byte 数据长度供用户，让 BC3602 自动间歇性传送/接收，此程序将会持续自动间歇性传送/接收，直到主循环将它停止。

- **WOT 模式**

ATR WOT 模式流程主要分为三个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ 和封包长度等信息，并且发送 Simple FIFO 模式封包。

步骤 1：等待 TX 完成 IRQ 信号进来后复位 TX FIFO 指标，并且设定 WOT 相关参数后，进入(下 IDLE 指令)WOT 模式。

步骤 2：等待 WTM 时间 IRQ 信号进来，并在此步骤重复执行。

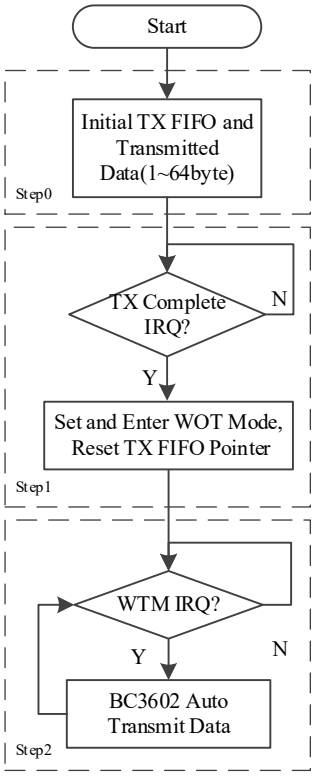

图 25. ATR_WOT 流程图

注：步骤 0 与步骤 1 前半部是为了能让第一笔传送数据让 WOR 可以同步才设置 TX Simple FIFO 模式，单纯 WOT 模式在步骤 1 后半部和步骤 2。

- **WOR 模式**

ATR WOR 模式流程主要分为四个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需要 IRQ 和封包长度等信息，并且进入 Simple FIFO RX 模式聆听数据进来。

步骤 1：等待 RX 接收完成/失败 IRQ 信号进来后，复位 RX FIFO 指标，并且设定 WOR 相关参数后进入(下 IDLE 指令)WOR 模式。

步骤 2：等待 WTM 时间 IRQ 信号进来，进入下一个步骤。

步骤 3：等待 RX 完成/失败和 WTM 时间 IRQ 信号进来；若是 RX 完成/失败 IRQ 立起，将复位 RX FIFO 指标，并重新进入(下 IDLE 指令)WOR 模式继续等待下一个 IRQ 状态；若是 WTM 时间 IRQ 立起，代表 BC3602 在 WOR RX 开启时间内，并无收到任何数据，此时程序将继续等待下一个 IRQ 状态。

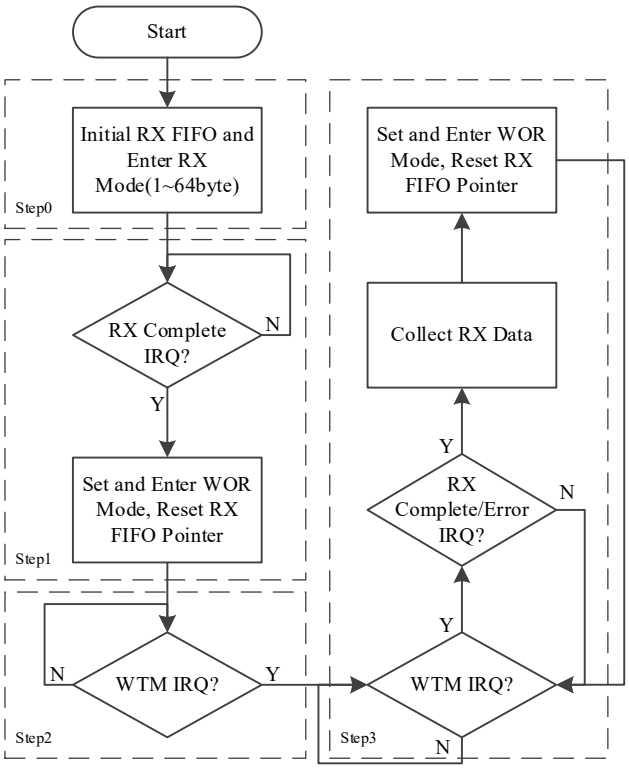

图 26. ATR_WOR 流程图

注：此处 RX Simple FIFO 是为了准确接收到第一笔接收数据让 WOR 可以同步 WOT 模式才设置，单纯 WOR 模式在步骤 1 后半部和步骤 3 判断 RX 完成/失败和 WTM 时间 IRQ。

ARK 模式

该程序提供封包 1~64 byte 数据长度(Simple FIFO TRX)供使用者，让 BC3602 传送/接收数据后进入 ARS(自动重传)/AAK(自动应答)模式，此程序将会持续传送/接收直到主循环将它停止。

- **ARS 模式**

ARK ARS 模式流程主要分为三个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：初始化所需 IRQ 和设定 ARS 模式相关参数并使能。

步骤 1：复位 TX FIFO 指标、PID 计数器更新和设置 TX 封包数据长度后进入(下 TX 指令)TX 和 ARS 模式。

步骤 2：等待 TX 完成和 ARK FAIL IRQ 信号进来；若是 TX 完成 IRQ 立起，代表 ARS 模式成功；若是 ARK FAIL IRQ 立起，则代表 ARS 模式下，并无收到任何有效响应；无论收到成功或是失败，程序都将会回到步骤 1 重新传送数据。

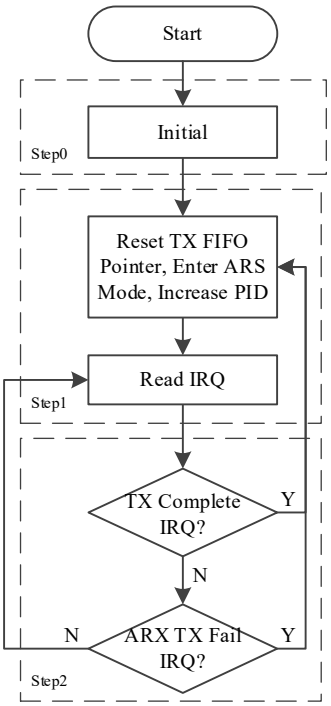

图 27. ARK_ARS 流程图

- **AAK 模式**

ARK AAK 模式流程主要分为两个区块，下列将讲解各个区块内部流程与动作方式。

步骤 0：始化所需 IRQ、复位 RX FIFO 指针、设定 AAK 模式相关参数，并进入(下 RX 指令)RX 和 AAK 模式。

步骤 1：待 RX 成功 IRQ 信号进来，若是 RX 成功代表 BC3602 收到更新过 PID 正确封包，此时将复位 RX FIFO 指针并且继续聆听数据进来直到中止(下 Light Sleep 指令)AAK 模式。

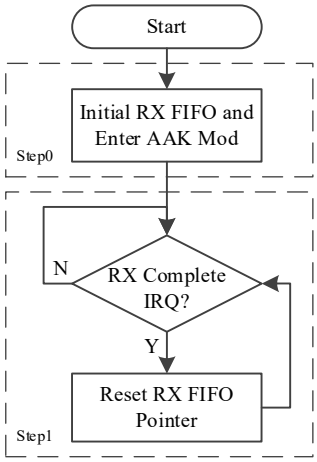

图 28. ARK_AAK 流程图

BC3602 API 函式介绍

范例程序中含 BC3602 API，主要分为三大类：命令类、一般类及自动收发类，下面便介绍各函式使用方式。

命令类

此类指令主要是 BC3602 内建快捷指令函式。

void BC3602_SoftwareReset (void)

输出：无。

输入：无。

功能：将 BC3602 复位。

说明：将 BC3602 寄存器、状态复位(有些寄存器数值并不能通过此命令复位，详情请参考 Datasheet 每一个 Bank 开头描述)。

void BC3602_DeepSleepMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入深度睡眠模式。

说明：BC3602 当前状态在 Light Sleep 和 IDLE 模式下才能执行此函式，使 BC3602 进入深度睡眠模式；请参考下图或 Datasheet 中 State Machine 章节；在深度睡眠模式下，BC3602 只对 SPI 有反应。

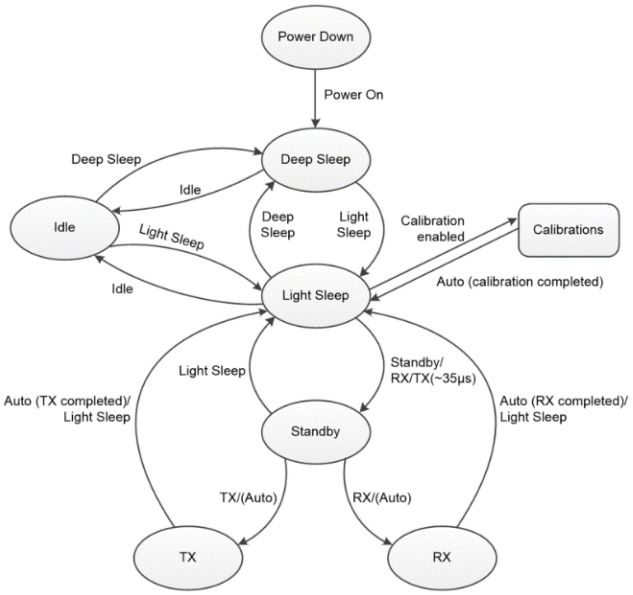

图 29. FIFO Mode State Diagram

void BC3602_IdleMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入空闲模式。

说明：BC3602 当前状态在 Light Sleep 和 Deep Sleep 模式下才能执行此函式，使 BC3602 进入空闲模式；请参考图 29 或 Datasheet 中 State Machine 章节。

void BC3602_LightSleepMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入浅层睡眠模式。

说明：BC3602 当前状态除了 Power Down 模式下不能进入浅层睡眠模式，其余状态执行此函式，都能使 BC3602 进入浅层睡眠模式；请参考图 29 或 Datasheet 中 State Machine 章节。

void BC3602_StandbyMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入待机模式。

说明：BC3602 当前状态除了 Light Sleep 模式下才能进入待机模式，其余状态执行此函式，都不能使 BC3602 进入待机模式;请参考图 29 或是 Datasheet 中 State Machine 章节。

void BC3602_TransmitterMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入发射模式。

说明：BC3602 当前状态除了 Light Sleep 模式下才能进入发射模式，其余状态执行此函式，都不能使 BC3602 进入发射模式，一旦在 Light Sleep 模式下执行此函式，BC3602 会先进入 Standby 模式自动调整后再进入发射模式，进入发射模式后，BC3602 会自动将 TX FIFO 内资料自动发射出去，使用者可以等待 IRQ 发生确保发射成功；请参考图 29 或是 Datasheet 中 State Machine 章节。

void BC3602_ReceiveMode (void)

输出：无。

输入：无。

功能：使 BC3602 进入接收模式。

说明：BC3602 当前状态除了 Light Sleep 模式下才能进入接收模式，其余状态执行此函式，都不能使 BC3602 进入接收模式，一旦在 Light Sleep 模式下执行此函式，BC3602 会自动进入 Standby 模式自动调整后再进入接收模式，进入接收模式后，BC3602 会把接收资料存入 RX FIFO 中，使用者可以等待 IRQ 发生确保接收成功或失败；请参考

图 29 或 Datasheet 中 State Machine 章节。

void BC3602_ResetRxFifoPointer (void)

输出：无。

输入：无。

功能：复位 BC3602 接收 FIFO 指标。

说明：BC3602 接收 FIFO 指标为自动累加，使用者并不能直接指定或控制；使用者在接收新封包前请务必执行此函式，确保 BC3602 接收指针处于开始位置。

void BC3602_ResetTxFifoPointer (void)

输出：无。

输入：无。

功能：复位 BC3602 发射 FIFO 指标。

说明：BC3602 发射 FIFO 指标为自动累加，使用者并不能直接指定或是控制；使用者在发射新封包前请务必执行此函式确保 BC3602 发射指针处于开始位置。

void BC3602_ReadRxPayload (pbuf,len)

输出：无。

输入：pbuf：读出接收数据暂存矩阵、len：读取接收数据长度。

功能：读取 BC3602 RX FIFO 数据(读取 len 长度数据给 pbuf)。

说明：在 RX 接收 IRQ 立起后，需通过此函式读取 BC3602 接收数据。

void BC3602_WriteTxPayload (pbuf,len)

输出：无。

输入： pbuf：写入发射资料矩阵、len：写入发射数据长度。

功能：写入 BC3602 TX FIFO 数据(写入 len 长度数据 pbuf 给 TX FIFO)。

说明：将要传送数据，需通过此函式写入 FIFO 寄存器。

一般类

此类指令可针对 BC3602 内部各项参数设定函式。

void BC3602_InterfaceConfigure (void)

输出：无。

输入：无。

功能：初始化 MCU 与 BC3602 沟通接口(SPI、GIO)。

说明：建立 MCU 与 BC3602 间 SPI(硬件/软件 SPI，依 MCU 不同而有变动，请至 BC3602.h 档案里设定)，并且将 GIO 引脚(GIO1~GIO4)全数设定成输入引脚。

```
#define  RF_SPI          HT_SPI1
#define  RF_SPI_CFGR     HT_AFIO->GPACFGR[0]
#define  RF_SPI_PORT     HT_GPIOA
#define  _RF_CSN_        3
#define  _RF_SCK_        0
#define  _RF_MOSI_       1
#define  _RF_MISO_       2
#define  RF_CSN          (1UL << _RF_CSN_)
#define  RF_SCK          (1UL << _RF_SCK_)
#define  RF_MOSI         (1UL << _RF_MOSI_)
#define  RF_MISO         (1UL << _RF_MISO_)
#define  RF_SPI_CFGR_MK ((2UL << (_RF_CSN_*4))|(2UL << (_RF_SCK_*4))|(2UL << (_RF_MOSI_*

#define RF_CSN_LOW     (RF_SPI_PORT->DOUTR &= ~RF_CSN)
#define RF_CSN_HIGH    (RF_SPI_PORT->DOUTR |= RF_CSN)
#define RF_SCK_LOW     (RF_SPI_PORT->DOUTR &= ~RF_SCK)
#define RF_SCK_HIGH    (RF_SPI_PORT->DOUTR |= RF_SCK)
#define RF_SPI_ACTIVE RF_SPI->CR0 |= SPI_SEL_ACTIVE
#define RF_SPI_INACTIVE RF_SPI->CR0 &= SPI_SEL_INACTIVE
```

图 30. 软件 SPI 引脚定义

```
/*-------------------------------------------------------------------------*/
/*   MCU & BC3602 interface Configure                                      */
/*-------------------------------------------------------------------------*/
void BC3602_InterfaceConfigure(void)
{
   RF_SPI_CFGR &= ~((15UL << (_RF_CSN_*4))  |
                    (15UL << (_RF_SCK_*4))  |
                    (15UL << (_RF_SDIO_*4)) );
   RF_SPI_PORT->DIRCR |= (RF_CSN | RF_SCK | RF_SDIO);   /* CSN/SCK/SDIO for output mode */
   RF_SPI_PORT->ODR &= ~(RF_CSN | RF_SCK | RF_SDIO);    /* CMOS output */
   RF_SPI_PORT->INER |= RF_SDIO;                        /* input enable */
   RF_SPI_PORT->PUR |= RF_SDIO;                         /* pull-high enable */
  #if (_IRQ_ENABLE_ == 1)
   RF_IO_CFGR &= ~(15UL << (_RF_IRQ_*4));               /* RF_IRQ to GPIO mode */
   RF_IO_PORT->DIRCR &= ~RF_IRQ;                        /* IRQ for input mode */
   RF_IO_PORT->INER |= RF_IRQ;                          /* input enable */
   RF_IO_PORT->PUR |= RF_IRQ;                           /* pull-high enable */
  #endif
   RF_CSN_HIGH;
   RF_SCK_LOW;
   RF_SDIO_LOW;

#if (_SPI_ARCHITECTURE_ != 0)
  /* SPI Configuration */
   RF_SPI->CR1 =  SPI_MASTER |  SPI_DATALENGTH_8 |
                  SPI_SEL_SOFTWARE | SPI_SELPOLARITY_LOW |
                  SPI_FIRSTBIT_MSB | 0x0100;      /* CPOL=1, CPHA=1 */
   /*-- Enable or Disable the specified SPI interrupt --*/
   RF_SPI->IER = 0x00000000;
   /*-- SPIx FIFO Control Register Configuration --*/
   RF_SPI->FCR = SPI_FIFO_DISABLE;
   /*-- SPIx Clock Prescaler Register Configuration -*/
   RF_SPI->CPR = ((CKCU_GetPeripFrequency(CKCU_PCLK_SPI1)/_SPI_SPEED_)-1)/2; /* fSCK=fPCLK/(2*(CP + 1)) */
   /* Enable or Disable the SEL output for the specified SPI peripheral.*/
   RF_SPI->CR0 = 0x00000008;
   /* Enable the selected SPI peripheral */
   RF_SPI->CR0 |= 0x00000001;
   /* configure CSN/SCK/MOSI/MISO as GPIO  */
   AFIO_GPxConfig(GPIO_PA,RF_CSN|RF_SCK|RF_MOSI|RF_MISO,AFIO_FUN_SPI);
#endif
```

图 31. 沟通接口初始化 API

void BC3602_LircCalibartion (void)

输出：无。

输入：无。

功能：LIRC 校调。

说明：使 BC3602 自动校调 LIRC，并且等待校正完毕。

void BC3602_VcoCalibartion (void)

输出：无。

输入：无。

功能：VCO 校调。

说明：使 BC3602 自动校调 VCO，并且等待校正完毕。

void BC3602_CrystalReady (void)

输出：无。

输入：无。

功能：等待晶振稳定。

说明：等待 BC3602 XCLK 是否准备就绪，若是晶振无法起振，则会停在此函式无法往下执行。

void BC3602_GioConfigure (u8 gio,u8 fun)

输出：无。

输入：gio：欲操作 GIO 引脚(编号)、fun：指定 GIO 功能。

功能：设定 BC3602 GIO 功能。

说明：BC3602 有 GIO1~GIO4 总共 4 只脚可以使用，每根 GIO 可设定为不同种功能；需注意 GIO1 和 GIO2 功能比较少，GIO3 和 GIO4 功能较为齐全(详情请至 BC3602 Datasheet 查询)。

范例：BC3602_GioConfigure(GIO2, INT_REQUEST);将 BC3602 GIO2 设定为 IRQ 功能引脚。

```
/* GIO function */
enum
{
   INPUT_MODE = 0,          /* GPIO input mode */
   SPI_SDO_FUN,             /* SPI SDO function */
   DIRECT_TRXD,             /* Direct mode TX/RD data input/output */
   DIRECT_TXD,              /* Direct mode TX data input */
   DIRECT_RXD,              /* Direct mode RX data output */
   INT_REQUEST,             /* Interrupt request output */
   LOSC_INPUT,              /* LOSC input */
   TBCLK_OUTPUT = 8,        /* Direct mode TX bit clock output */
   RBCLK_OUTPUT,            /* Direct mode RX bit clock output */
   FCLK_OUTPUT,             /* Frequency clock output */
   LOSC_OUTPUT,             /* LOSC output */
   EPA_CTLO,                /* External PA control output */
   ELNA_CTLO,               /* External LNA control output */
   DIRECT_TRBCLK            /* Direct mode Tx/Rx bit clock output */
};
```

```
enum
{
   GIO1 = 0,
   GIO2,
   GIO3,
   GIO4
};
```

图 32. GIO 引脚(编号)及功能编号

void BC3602_IrqConfigure (u8 irq_en)

输出：无。

输入：irq_en：欲开启 IRQ 功能。

功能：设定 BC3602 IRQ 功能开关。

说明：设定 BC3602 IRQ 功能开关，包含 TX 成功发射、RX 接收完成、校正完成、RX 事件接收、RX 接收 CRC 错误、FIFO 门坎、ATR 时间和 ARK 发射失败功能(详情请至 BC3602 Datasheet 查询)。

范例：BC3602_IrqConfigure(TXCMPIE_EN | RXCMPIE_EN | RXERRIE_EN);将 TX 完成、RX 接收完成和 RX 接收失败 IRQ 功能开启。

```
/* IRQ2 */

#define TXCMPIE_EN     0x01        //TX Complete IRQ Enable
#define RXCMPIE_EN     0x02        //RX Complete IRQ Enable
#define CALCMPIE_EN    0x04        //Calibration Complete IRQ Enable
#define RXDETIE_EN     0x08        //RX Event Detected IRQ Enable
#define RXERRIE_EN     0x10        //RX Error IRQ Enable
#define FIFOLTIE_EN    0x20        //FIFO Low Threshold IRQ Enable
#define ATRCTIE_EN     0x40        //ATR Cycle Timer IRQ Enable
#define ARKTFIE_EN     0x80        //ARK TX Failure IRQ Enable
```

图 33. IRQ 功能编号

void BC3602_AgcConfigure(u8 en)

输出：无

输入：en:AGC 功能开关

功能：设定 AGC 相关功能

说明：开启/关闭 AGC 功能并设定相关参数

void BC3602_CrystalConfigure (void)

输出：无。

输入：无。

功能：设定晶振负载电容值(C-Load)。

说明：设定晶振负载电容值为 16pF(0x41);若要设其他值可用 BC3602_WriteRegister 函式。

void BC3602_PreambleConfigure (u16 tx_preamble,u8 rx_preamble)

输出：无。

输入： tx_preamble ： TX Preamble 数量、rx_preamble：RX Preamble 数量(编号)。

```
enum
{
  _RX_Preamble_1B_ = 0,
  _RX_Preamble_2B_,
  _RX_Preamble_4B_,
};
```

图 34. RX Preamble 数量(编号)

功能：设定 BC3602 在传送与接收门坎 Preamble 数量(byte)。

说明：设定传送 Preamble 数量(1~256 byte)与接收门坎 Preamble 数量(1、2、4 byte)。

范例：BC3602_PreambleConfigure (8,_RX_Preamble_4B_);将 BC3602 TX Preamble 设置为 8byte，RX Preamble 设为 4byte。

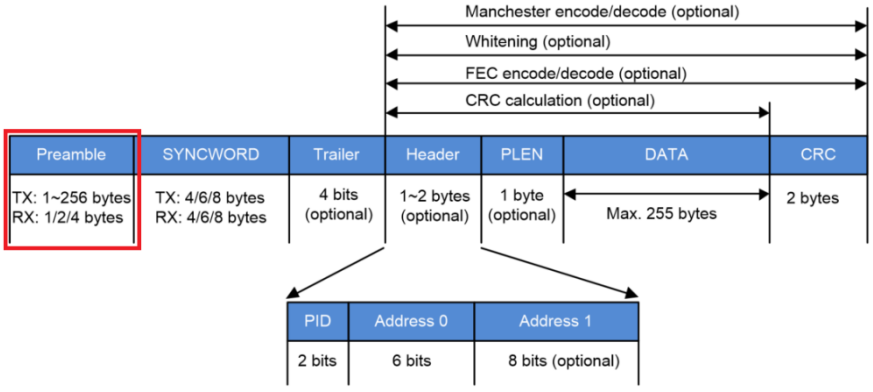

图 35. RX Preamble 数量(编号)

void BC3602_SyncwordConfigure (u8 length, int64_t id)

输出：无。

输入：length：TX/RX Syncword 长度(编号)、id：设备序列编码(40-bit)。

功能：计算并设定 Syncword 长度和 Syncword。

```
enum
{
  _Syncword_4B_ = 0,
  _Syncword_6B_,
  _Syncword_8B_,
};
```

```
//********************MAIN MENU**********************

#define DEFAULT_RF_Band           _RF_Band_433MHz_
#define DEFAULT_TX_Preamble       8
#define DEFAULT_RX_Preamble       _RX_Preamble_4B_
#define DEFAULT_SyncWidth         _Syncword_4B_
#define DEFAULT_DeviceID          0xab00111111
#define DEFAULT_TX_Power          _TX_Power_10dBm_
#define DEFAULT_DATA_RATE         _DataRate_50K_
```

图 36. Syncword 长度(编号) 、设备序列码

说明：程序内提供一设备序列码，此序列码(必须填入 5byte 长度)经过此 API 后将会换成另外一组 Syncword 供 BC3602 使用。

范例：BC3602_SyncwordConfigure (_Syncword_4B_,DEFAULT_DeviceID);将 Syncword 长度设置为 4 byte，并以这组序列码经 BCH 计算后写入对应 Syncword 位置。

注意事项：当选择 Syncword 为 4-byte 长度时，API 将会撷取设备 ID 后面 20-bit 作为转换来源使用；当选择 Syncword 为 6-byte 长度时，API 将会撷取设备 ID 后面 25-bit 作为转换来源使用；当选择 Syncword 为 8-byte 长度时，API 将会撷取设备 ID 全部 40-bit 作为转换来源使用。

例子：

选择 4-byte 长度，设备序列码 0x123456789A，系统将会取 0x000006789A(20-bit)。

选择 6-byte 长度，设备序列码 0x123456789A，系统将会取 0x000056789A(25-bit)。

选择 8-byte 长度，设备序列码 0x123456789A，系统将会取 0x123456789A(40-bit)。

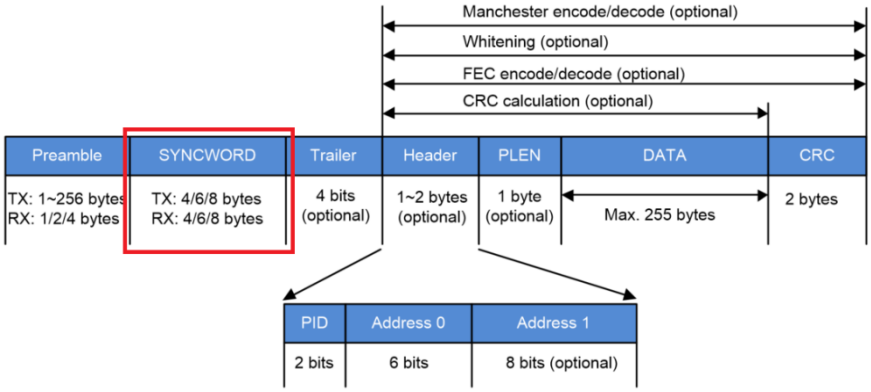

图 37. Packet Format - Syncword

void BC3602_HeaderConfigure (u8 PLLEN_EN, u8 PLHAC_EN, u8 PLHLEN, u8 PLH_EN)

输出：无。

输入：PLLEN_EN：封包长度信息开关、PLHAC_EN：延长 address 功能、PLHLEN ：Header 长度设定、PLH_EN ： Header 开关。

功能：设定 BC3602 Header 相关参数。

说明：BC3602 封包支持 Header 与封包长度功能，可以通过 PLH_EN 和 PLLEN_EN 分别设定 Header(与 address 相关)与 PLEN(是否将长度信息藏在封包里)开关，而 PLH_EN 使能后可进一步设定 PLHLEN(address 长度选择)和 PLHAC_EN(延长 address 可以设定硬件 address 或是当成一般数据)。

范例： BC3602_HeaderConfigure (Enable, Enable, Enable, Enable);将 BC3602 Header 相关功能全部使能。

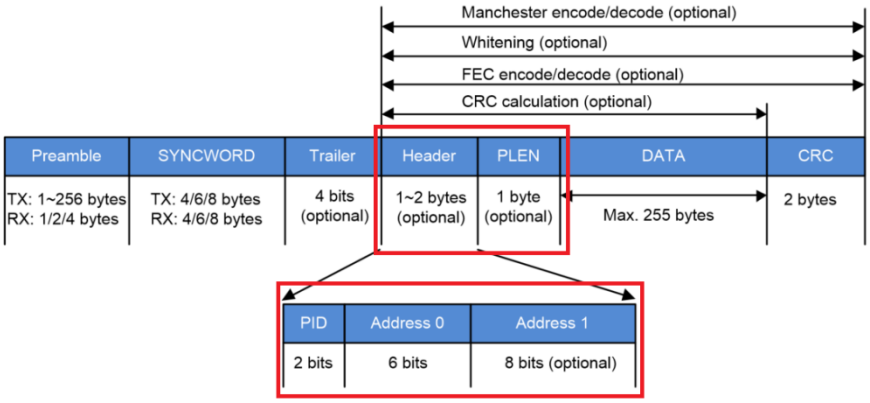

图 38. Packet Format - Header and PLEN

void BC3602_ManchesterConfigure (u8 man_EN)

输出：无。

输入： man_EN： Manchester 编码开关。

功能：启动/关闭 BC3602 数据以曼彻斯特编码方式发射/接收。

说明：BC3602 封包支持曼彻斯特编码；开启曼彻斯特编码可让 RF 在空气中传递更为强健，但传输时间多两倍。

范例：BC3602_ManchesterConfigure (Enable)；将 BC3602 曼彻斯特功能使能。

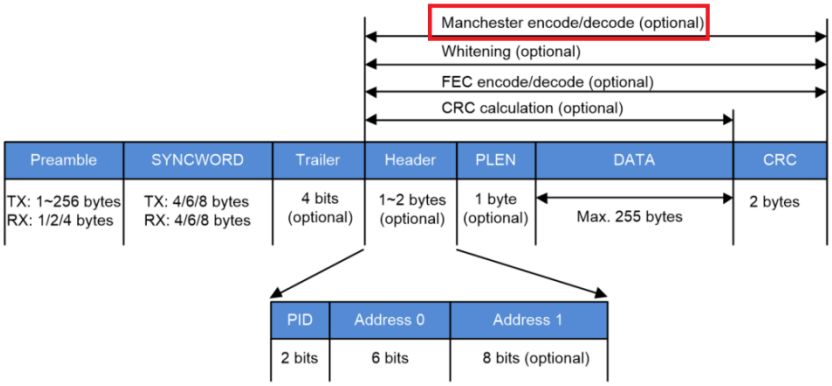

图 39. Packet Format - Manchester

void BC3602_FecConfigure (u8 fec_EN)

输出：无。

输入：fec_EN：前向纠错编码开关。

功能：启动/关闭 BC3602 数据以简易 7-4 汉明编码方式发射/接收。

说明：BC3602 封包支持简易 7-4 汉明码纠错功能;开启 fec_EN 设置前向纠错功能，采用 7-4 汉明码，数据量大 7/4 倍。

范例：BC3602_FecConfigure (Enable)；将 BC3602 前向纠错功能使能。

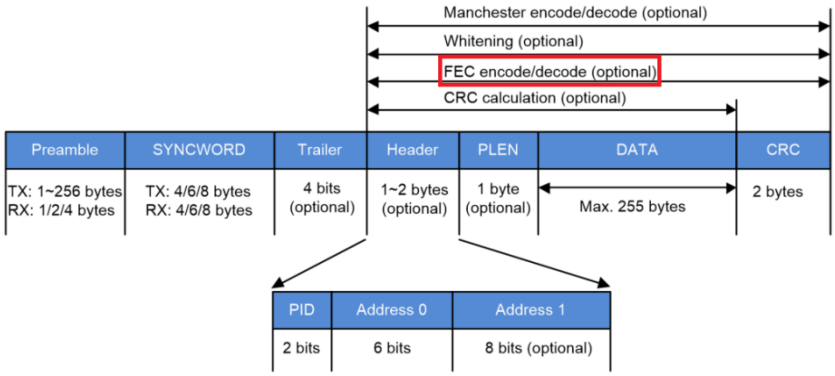

图 40. Packet Format - FEC

void BC3602_Crc_Configure (u8 crc_EN, u8 crcfmt)

输出：无。

输入：crc_EN：2-byte CRC 编码开关、crcfmt：CRC 公式选择。

功能：设定 BC3602 封包 CRC 相关功能。

说明：BC3602 封包支持循环冗余校验功能；开启 crc_EN 可让 BC3602 在封包末端设置 CRC 检查码，通过 crcfmt 选择 CRC 计算公式。

范例：BC3602_Crc_Configure (Enable,1)；将 BC3602 CRC 功能使能，并选择 CRC 公式为 1。

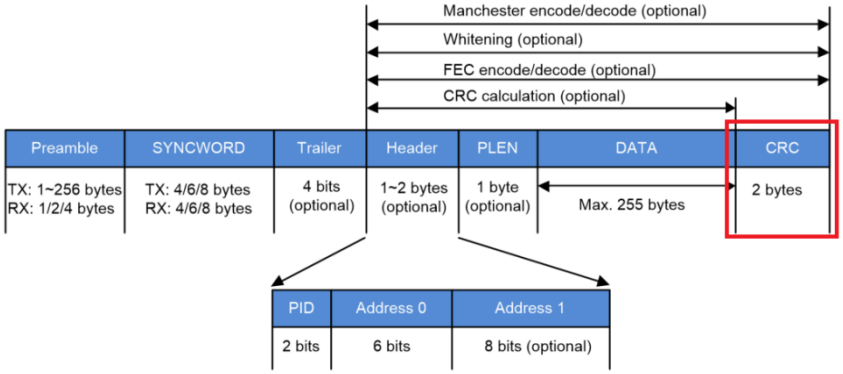

图 41. Packet Format - CRC

void BC3602_TrailerConfigure (u8 tra_EN)

输出：无。

输入：tre_EN：Trailer 功能开关。

功能：启动/关闭 BC3602 封包 Trailer 编码是否需要。

说明：开启 tre_EN 可以在 Syncword 后设置 Trailer 以便和 BC3601 沟通。

范例：BC3602_TrailerConfigure (Enable)；将 Trailer 功能使能。

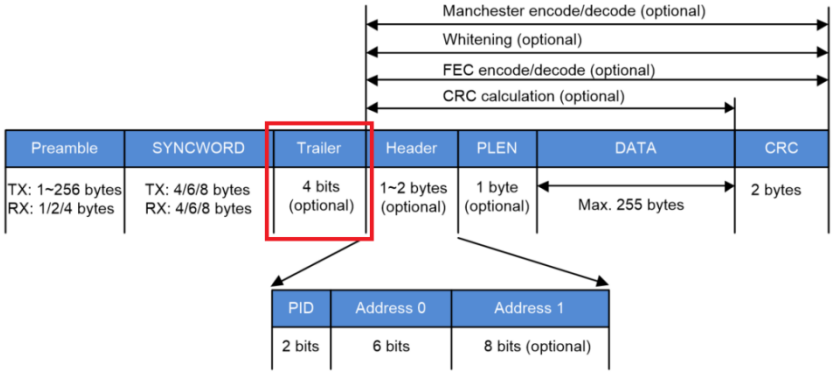

图 42. Packet Format – Trailer

void BC3602_WhiteningConfigure (u8 wht_EN, u8 whtfmt, u8 wht_seed)

输出：无。

输入：wht_EN：whitening 功能开关、whtfmt：whitening 公式选择、wht_seed：whitening 种子选择。

功能：设定 BC3602 whitening 编码相关功能。

说明：BC3602 支持数据白化功能；开启 wht_EN 可以让数据经过 XOR 计算后传出，以达到白化目的、whtfmt 可选择公式(请参照 Datasheet)、wht_seed 则可以选择欲加密数列种子(此种子数列可填入数值范围为 1~255，填 0 为无效数值)。

范例：BC3602_WhiteningConfigure (Enable, 0,54)；将 BC3602 whitening 相关功能使能，分别选择 whitening 公式为 0 后，并且选择第 54 号序列种子。

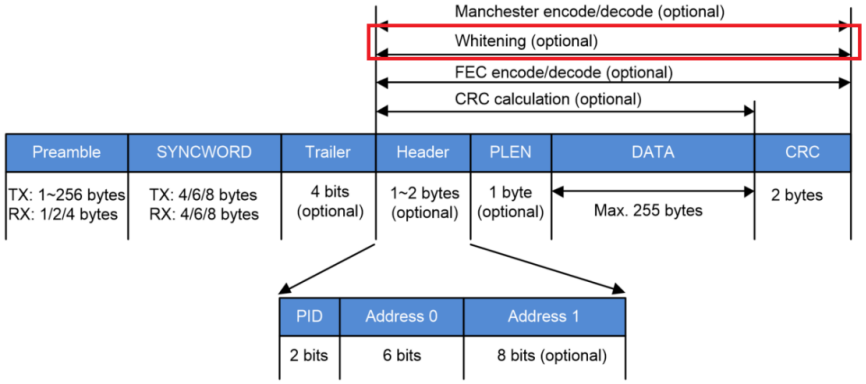

图 43. Packet Format - Whitening

void BC3602_FrequencyConfigure (float frequ)

输出：无。

输入：frequ：设置 BC3602 射频频率(MHz)。

功能：设置 BC3602 欲发送/接收射频频率。

说明：BC3602 支持射频频段(频率范围)有 315、433、470、868 及 915MHz，以用户所需要频率去设置(每个频段都有可以调整范围，该范围数值请参考 Datasheet)。

范例：BC3602_FrequencyConfigure (433.92)；将 BC3602 射频频率设置为 433.92MHz。

void BC3602_PowerConfigure (u8 pwr)

输出：无。

输入：pwr：发射功率(编号)。

```
enum
{
  _TX_Power_0dBm_ = 0,
  _TX_Power_5dBm_,
  _TX_Power_10dBm_,
  _TX_Power_13dBm_,
};
```

图 44. TX 发射功率(编号)

功能：设置 BC3602 欲发送射频功率。

说明：目前函式支持+0、+5、+10 和+13dBm 射频功率。

范例：BC3602_PowerConfigure (_TX_Power_10dBm_)；设定 BC3602 发射功率为 433MHz 频段下+10dBm。

void BC3602_DataRateConfigure (u8 dr)

输出：无。

输入：dr：发射/接收数据速度(编号)。

```
enum
{
  _DataRate_2K_ = 0,
  _DataRate_5K_,
  _DataRate_10K_,
  _DataRate_25K_,
  _DataRate_50K_,
  _DataRate_125K_,
  _DataRate_250K_,
};
```

图 45. 发射/接收数据速度(编号)

功能：设置 BC3602 欲发送/接收数据速度。

说明：目前支持 2K、5K、10K、25K、50K、125K 和 250K 数据速度。

范例：BC3602_DataRateConfigure (_DataRate_50K_)；设定 BC3602 传送/接收数据速率为 50K。

void parameter_initialization(void)

输出：无。

输入：无。

功能：初始化 BC3602 所有射频参数设定。

说明：此函式功能为依参数列表(请至 BC3602.h 档案里设定，图 46)设定值，将 BC3602 所有 RF 参数设定完成。

```
//*******************MAIN MENU*********************

#define DEFAULT_RF_Band           _RF_Band_433MHz_        //433MHz
#define DEFAULT_TX_Preamble       8                       //1~256      8 = 8 byte
#define DEFAULT_RX_Preamble       _RX_Preamble_4B_        //4 byte
#define DEFAULT_SyncWidth         _Syncword_4B_           //4 byte
#define DEFAULT_DeviceID          0xab00111111            //0~0xFFFFFFFFF //40bit
#define DEFAULT_TX_Power          _TX_Power_10dBm_        //+10 dBm
#define DEFAULT_DATA_RATE         _DataRate_50K_          //50K bps

#define DEFAULT_Man_EN            Enable                  //Manchester code enable
#define DEFAULT_FEC_EN            Enable                  //FEC enable
#define DEFAULT_CRC_EN            Enable                  //CRC field enable
#define DEFAULT_CRCFMT            1                       //CRC format selection
#define DEFAULT_Tra_EN            Enable                  //Trailer enable

#define DEFAULT_WHT_EN            Disable                 //Data whitening enable
#define DEFAULT_WHTFMT            0                       //whitening format selection
#define DEFAULT_WHTSEED           54                      //whitening Seed selection

#define DEFAULT_PLLEN_EN          Enable                  //Payload length field enable
#define DEFAULT_PLHAC_EN          Enable                  //Payload header address corre
#define DEFAULT_PLHLEN            Enable                  //1~2byte Payload header lengt
#define DEFAULT_PLH_EN            Enable                  //Payload header field enable

#define DEFAULT_PKT_Length        64                      //1~64

//*****************************************************************
```

图 46. RF 参数清单

u8 BC3602_ReadRegister (u8 regs)

输出：1 byte 数据(代表 BC3602 现在想读出寄存器数值)。

输入：regs：欲读出寄存器位置。

功能：查看 BC3602 该寄存器数值。

void BC3602_WriteRegister (u8 regs,u8 data)

输出：无。

输入：regs：欲写入寄存器位置、data：欲写入寄存器数值。

功能：写入 BC3602 该寄存器数值。

u8 BC3602_GetIrqState ()

输出：1 byte 数据(代表 BC3602 现在 IRQ 状态，可参考 Datasheet IRQ3 寄存器)。

输入：无。

功能：查看 BC3602 此时 IRQ 状态。

说明：通过此 API 查看 BC3602IRQ 状态，进而得知 BC3602 完成事件触发。

```
/* IRQ3 */
enum
{
  _b_TX_COMPLETE_IRQ_ = 0,
  _b_RX_COMPLETE_IRQ_,
  _b_CALIBRATION_COMPLETE_,
  _b_RX_CRY_PRMBL_SYNC_IRQ_,
  _b_RX_FAIL_IRQ_,
  _b_TX_RX_MEG_IRQ_,
  _b_WTX_WAKEUP_IRQ_,
  _b_ARK_TX_FAIL_IRQ_,
};

#define _TX_COMPLETE_IRQ_          (0x01 << _b_TX_COMPLETE_IRQ_)
#define _RX_COMPLETE_IRQ_          (0x01 << _b_RX_COMPLETE_IRQ_)
#define _CALIBRATION_COMPLETE_     (0x01 << _b_CALIBRATION_COMPLETE_)
#define _RX_CRY_PRMBL_SYNC_IRQ_    (0x01 << _b_RX_CRY_PRMBL_SYNC_IRQ_)
#define _RX_FAIL_IRQ_              (0x01 << _b_RX_FAIL_IRQ_)
#define _TX_RX_MEG_IRQ_            (0x01 << _b_TX_RX_MEG_IRQ_)
#define _WTX_WAKEUP_IRQ_           (0x01 << _b_WTX_WAKEUP_IRQ_)
#define _ARK_TX_FAIL_IRQ_          (0x01 << _b_ARK_TX_FAIL_IRQ_)
```

图 47. IRQ 编号

void BC3602_ClearIrqFlag (u8 sts)

输出：无。

输入：IRQ 事件(可参考 Datasheet IRQ3 寄存器)。

功能：清除指定 IRQ 事件 IRQ 状态。

说明：当 BC3602 触发事件发生时，IRQ 状态将会立起，此时使用者必须将该状态清除才能等待下一个状态发生。

u8 BC3602_GetModeState (void)

输出：1 byte 数据(代表 BC3602 现在状态，可参考 Datasheet STA1 寄存器)。

输入：无。

功能：查看 BC3602 现在状态(State machine)。

说明：BC3602 运作模式(State machine)转换有固定的流程，请务必遵照图 29 操作。

u8 BC3602_ReadReceiveRSSI (void)

输出：1 byte 数据(RSSI 值，可参考 Datasheet RSSI4 寄存器)。

输入：无。

功能：查看 BC3602 现在接收到正确 Syncword 封包 RSSI 数值。

说明：RSSI 数值有分为两种：环境与接收封包，此 API 读取到为接收到正确 Syncword 封包 RSSI 数值。

u8 BC3602_ReadEnvironmentRSSI (void)

输出：1 byte 数据(RSSI 值，可参考 Datasheet RSSI3 寄存器)。

输入：无。

功能：查看 BC3602 现在环境中 RSSI 数值。

说明：RSSI 数值有分为两种：环境与接收封包，此 API 读取到为环境中 RSSI 数值，请确保呼叫此 API 时候 BC3602 是处在 RX 模式当中。

命令自动收发类

此类指令则是为了能方便使用 BC3602 中自动收发功能所制定。

void BC3602_SetAutoCycleMode (u8 md,bool en)

输出：无。

输入：md：欲使用 ATR 操作模式(编号)、en：选择该模式开关。

功能：开启/关闭 WOT、WOR、WTM 模式。

说明：开启/关闭 WOT、WOR、WTM 模式，但是启动该模式必须要下达 IDLE 指令才能使模式正式运行。

```
/* ATR1 */
enum
{
  _ATR_WTX_ = 0,
  _ATR_WRX_,
  _ATR_WTM_
};
```

图 48. ATR 操作模式(编号)

void BC3602_SetAutoCycleTimer (u32 tm)

输出：无。

输入：tm：ATR 循环时间。

功能：设定 ATR 模式下循环时间。

说明：WOT、WOR、WTM 功能都有一个循环时间，此参数提供这些模式设定循环时间，时间单位为 μs。

void BC3602_SetAutoRxActivePeriod (u32 tm)

输出：无。

输入：tm：WORRX 开启时间。

功能：设定 WOR 开启 RX 时间长度。

说明：设定 WOR 开启 RX 时间长度，该时间单位为 μs。需注意时间单位虽然为 μs，但芯片本身是以 250μs 为一个基本单位，设定值需超过 250μs。

范例：BC3602_ SetAutoRxActivePeriod (740)；该设定值小于 750 大于 500，实际运作时间会是 500μs。

void BC3602_SetAutoRxExtendPeriod (u32 tm)

输出：无。

输入：tm：WOR RX 延长时间。

功能：设定 WOR 延长开启 RX 时间长度。

说明：设定 WOR 延长开启 RX 时间长度，该时间单位为 μs。延长开启时间是在正常 RX 开启时，收到 Preamble 后 RX 开启时间便会加上 tm 设定值，此机制可以有效防止过长 RX 开启时间导致耗能问题；该参数时间单位为 250μs。

void BC3602_EnableATRCTM (bool flag)

输出：无。

输入：flag：ATR 时间连续性控制。

功能：设定 ATR 时间连续性。

说明：当设定该参数为 0 时，为单一模式，内部定时器会在每个 ATR 转换时自动重新计时；当设定参数为 1 时，为连续模式，计数器将只会在收到 IDLE 命令时开始计时，且会在 ATR_EN=0 或 ATRCTM=0 结束计时。

void BC3602_SetATRCT_Timer (u32 tm)

输出：无。

输入：tm：内部计数器时间。

功能：设定内部计数器时间。

说明：设定内部计数器时间，时间单位为 μs。

void BC3602_EnableARK (bool en)

输出：无。

输入：en：选择 ARK 模式开关。

功能：开启/关闭 ARK 模式。

说明：开启/关闭 ARK 模式，但是启动该模式必须要分别下达 TX/RX 指令才能使 ARS/AAK 模式正式运行。

void BC3602_SetARKRXAP_Timer (u32 tm)

输出：无。

输入：tm：ARSRX 开启时间/AAKRX 间隔时间。

功能：设定 ARSRX 开启时间或是 AAK RX 间隔时间。

说明：该时间单位为 μs。需注意时间单位虽然为 μs，但芯片本身是以 250μs 为一个基本单位，设定值需超过 250μs，不足 250μs 向下取整数。

范例：BC3602_SetARKRXAP_Timer (740)，该设定值小于 750 大于 500，实际运作时间会是 500μs。

void BC3602_SetAutoRetryNumber (u8 counter)

输出：无。

输入：counter：ARS 重传次数。

功能：设定 ARS 重传次数。

说明：当 ARS 模式中 RX 并无收到有效封包，ARS 模式将会再重新传送上一次传送封包，直到该 RX 有接收到正确封包或是该参数最大值+1。

void BC3602_IncPID (void)

输出：无。

输入：无。

功能：递加 PID 数值。

说明：设定递增 PID 参数在 ARK 模式尤为重要，此参数能用来判断 ARK 传送封包是否为重复封包，在使用此功能前须先将 Header 功能开启(请参照寄存器 PLH_EN 或是 BC3602_HeaderConfigure)。

结论

本文介绍了 Holtek 针对 BC3602 无线收发器所设计韧体开发板，通过此开发板可较轻松来了解此芯片，进一步来开发出您所需产品。

参考资料

参考文件 BC3602 Datasheet。

如需进一步了解，敬请浏览 Holtek 官方网站 www.holtek.com.cn。

版本及修改信息

日期	作者	发行	修改信息
2022.03.07	何信智	V1.30	附件程序更新
2021.09.17	何信智	V1.20	附件程序更新
2020.07.31	何信智	V1.10	附件程序更新及相关说明修改
2019.10.07	吴柏颖	V1.00	第一版

免责声明

本网页所载的所有数据、商标、图片、链接及其他数据等 (以下简称「数据」)，只供参考之用，合泰半导体 (中国) 有限公司及其关联企业 (以下简称「本公司」) 将会随时更改数据，并由本公司决定而不作另行通知。虽然本公司已尽力确保本网页的数据准确性，但本公司并不保证该等数据均为准确无误。本公司不会对任何错误或遗漏承担责任。

本公司不会对任何人士使用本网页而引致任何损害 (包括但不限于计算机病毒、系统故障、数据损失) 承担任何赔偿。本网页可能会连结至其他机构所提供的网页，但这些网页并不是由本公司所控制。本公司不对这些网页所显示的内容作出任何保证或承担任何责任。

责任限制

在任何情况下，本公司并不须就任何人由于直接或间接进入或使用本网站，并就此内容上或任何产品、信息或服务，而招致的任何损失或损害负任何责任。

管辖法律

以本公司所在地法律为准据法，并以本公司所在地法院为第一审管辖法院。

免责声明更新

本公司保留随时更新本免责声明的权利，任何更改于本网站发布时，立即生效。