BC3603 开发板应用范例

文件编号：AN0652SC

简介

Holtek 推出全新双向无线 OOK/GFSK 高效能射频芯片 BC3603，适合在 1GHz 以下免执照 ISM Band(300MHz~960MHz)应用；IC 集成高功率 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。

BC3603 工作电压为 1.8V~3.6V，可程序设定发射功率，最高达+20dBm；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，内置高精准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 BC3603 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 BC3603 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。

环境准备

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-GENTrx32-002，搭配 BC3603 模块化开发板 BCT-3603-X01+BM3603-0x-1，完整平台结构如下图所示。

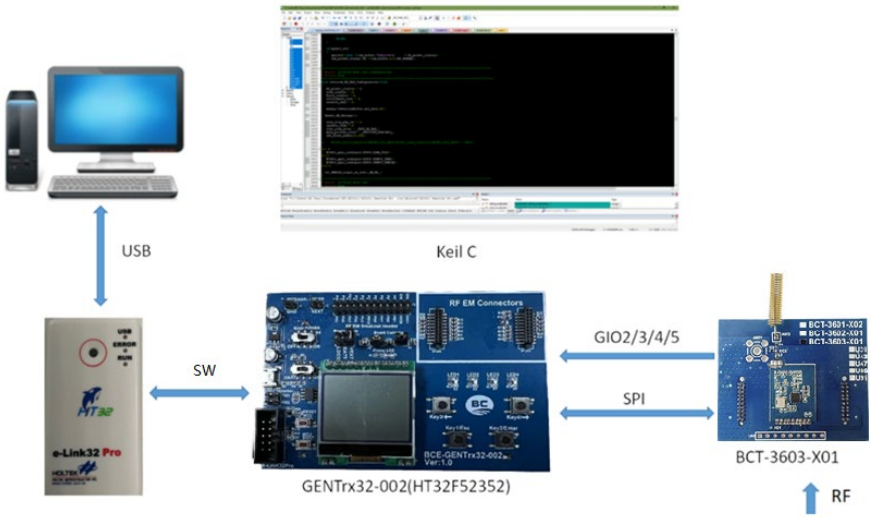

图 1. 系统架构图

HT32F52352 是 Arm® M0+单片机，用户需要先在计算机上安装 keil µVisionIDE 接口，搭配 Holtek 所开发 e-Link32 Pro，通过 SW 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-GENTrx32-002 开发板提供了良好操作接口，方便用户操作，如下图所示。

图 2. 系统开发板组成

包含了：

1. 射频模块接口为射频发射/接收设备连接处，此范例则安装上 BCT-3603-X01+BM3603-0x-1。

图 3. BCT-3603-X01 + BM3603-0x-1

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. 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 BC3603 的 GIO1~GIO3，详细 I/O 如下图所示。

BCE-GENTrx32-002	BM3603-0x-1
PA3	CSN
PA0	SCK
PA1	SDIO
PA2	GIO1
PC4	GIO2
GPIO2	GIO3

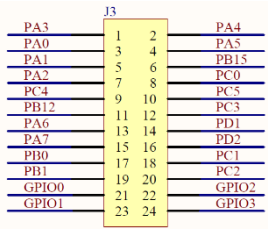

图 5. I/O 接口

5. MCU 和 BC3603 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 端口输入；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 8)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. Power Source Select

7. 总电源开关，左拨(OFF)为关闭电源，右拨为开启电源。

圖 7. Power Switch

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. SW 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，可用来作为系统电源输入(电源选择 VEXT)。

图 8. External Power Connect

使用指南

BC3603 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 BC3603 范例程序。

挑选范例程序

建置在 BCE-GENTrx32-002 开发平台上 BC3603 范例程序共有 7 种分别为：TX Carry、Simple FIFO、Extend FIFO、PER Test、WOT Mode、WOR Mode 和 ARK Mode，有关各工作模式详细描述可参考 BC3603 Datasheet。

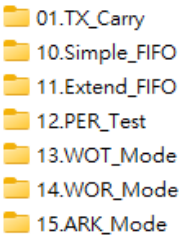

图 9. 范例程序

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 65~255 bytes 长度数据。
4. PER Test：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. WOT Mode(间歇传送)：该程序提供用户间歇自动传送 1~64 byte 长度数据而不需要 MCU 介入。
6. WOR Mode(间歇接收)：该程序提供用户间歇自动接收 1~64 byte 长度数据而不需要 MCU 介入。
7. ARK Mode：该程序分为 TX(传送)与 RX(接收)，提供使用者自动重发/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 BCE-GENTrx32-002 开发板和 2 块 BCT-3603-X01 模块，分别将 BCT-3603-X01 模块安装至 BCE-GENTrx32-002 开发板上，并把 e-Link32 Pro 连接头安装至 BCE-GENTrx32-002 开发板后开机等待烧录。

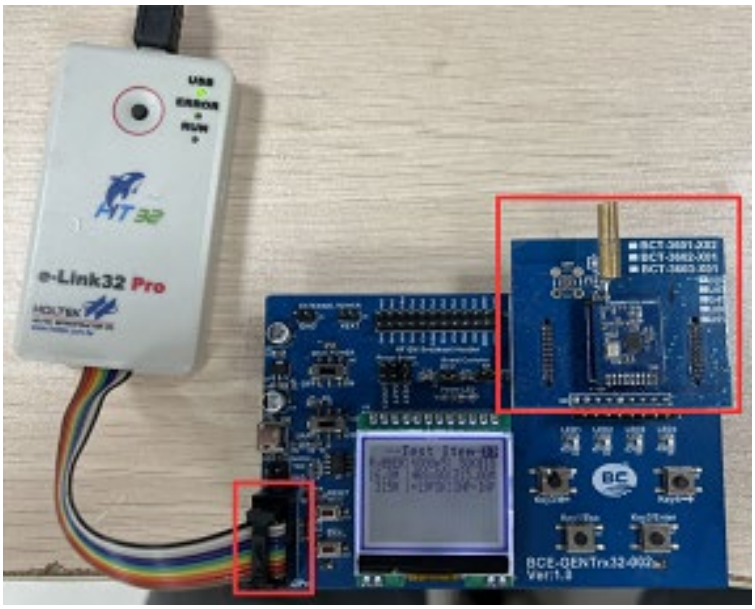

图 10. 安装硬件示意图

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..\BC3603_DemoCode_M0+\Example Code\12.PER_Test\12.1.Master\BC3603_M0+_DemoKit.uvprojx (以 PER 作为范例)后，依下列顺序操作：

1. 点选“Project\Options for Target 'xxxxx'”，或点击如下图标，开启配置窗口。

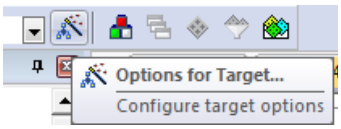

图 11. 项目设定示意图 1

2. 开启“除错(Debug)”分页。
3. 在连接设备选择“CMSIS-DAP Debugger”。
4. 点选“设定(Settings)”开启连结设备设定窗口。

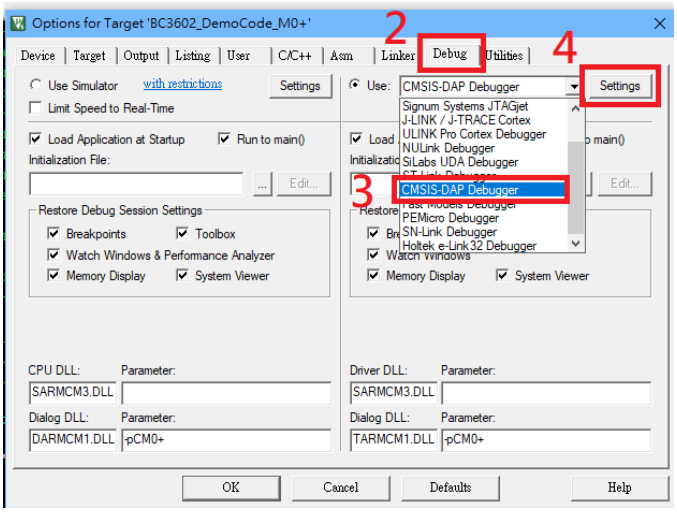

图 12. 项目设定示意图 2

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-Link32 Pro 驱动程序。

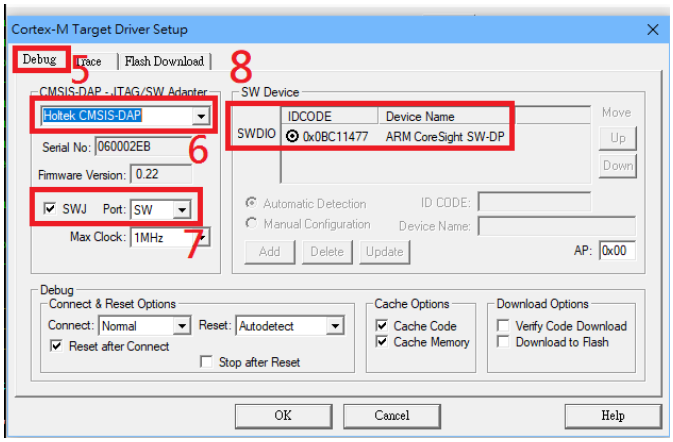

图 13. 项目设定示意图 3

9. 点选“Rebuild”重新编译。
10. 点选“Download”下载程序至 BCE-GENTrx32-002 开发板。

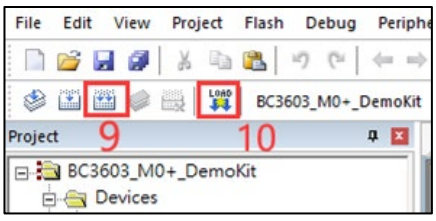

图 14. 项目设定示意图 4

11. 完成 PER_Master 的烧录后，再依上述步骤，在另一张开发板中烧录 12.PER_Test 下的 PER_Slave 程序。之后按压板子上的复位键，使两张开发板重新上电。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 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 板开机后按下 KEY2 即会进入 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。

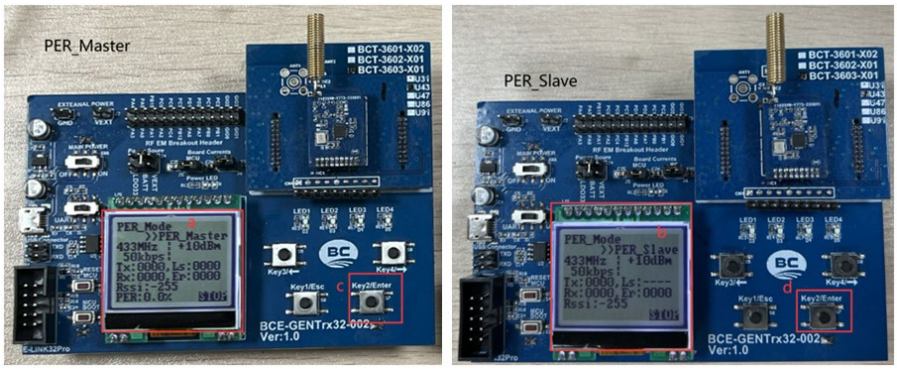

图 15. PER 操作示意图

范例程序简介

此范例程序分为 7 类共 14 支程序，接下来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将分开说明。

RF 相关参数配置

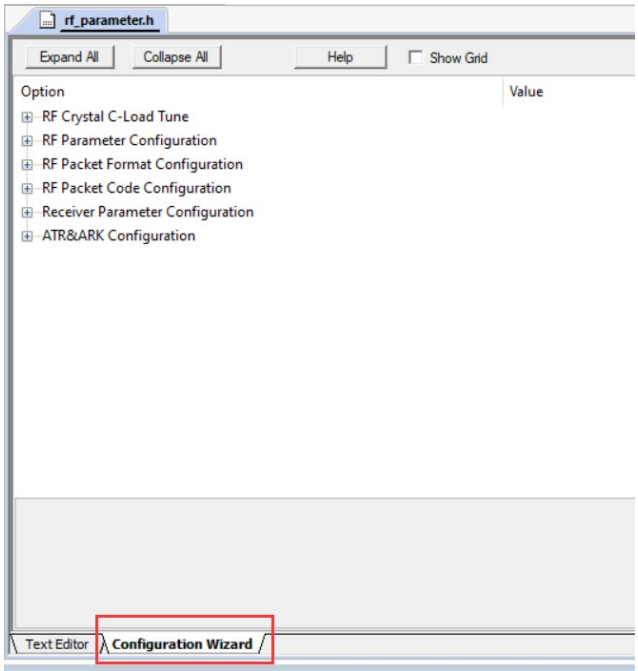

图 16. 参数清单

选择并开启范例程序后，先对程序进行编译。编译后打开 rf_parameter.h 档，在 Configuration Wizard 页面中可以看到各类参数选项，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. RF Crystal C-Load Tune

(1) Coarse Tune：晶振负载电容值粗调，可设定范围 0~3。

(2) Fine Tune：晶振负载电容值细调，可设定范围 0~31。

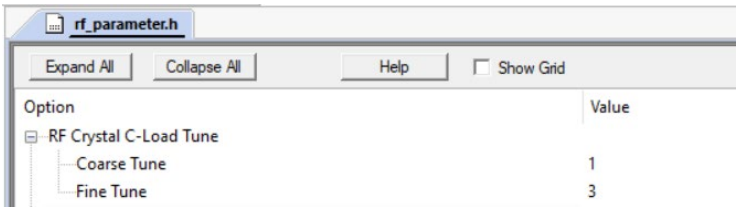

图 17. 晶振 C-Load 设定

2. RF Parameter Configuration

(1) RF Frequency

- Frequency 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
- Frequency：射频频率设定，根据所选频段，设定具体的工作频率，可设定范围 300000000~999000000

(2) RF TX Power：RF 发射功率设定，有 0、+10dBm、+13dBm、+17dBm、+19dBm，五种可供选择。

(3) RF Data Rate：数据传输速率设定，有 2Kbps、10Kbps、50Kbps、125Kbps、250Kbps，五种可供选择。

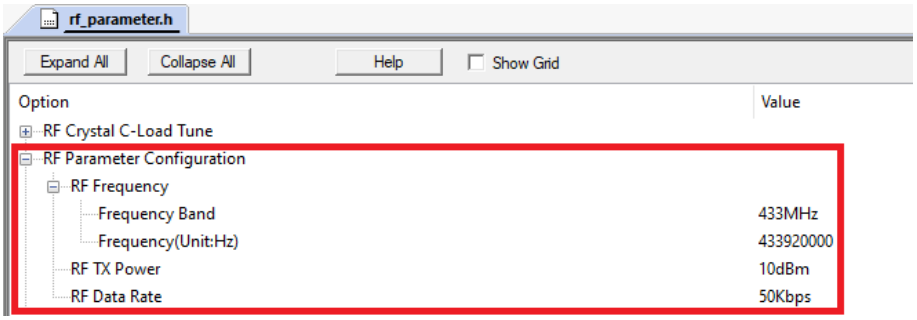

图 18. RF 参数设定

3. RF Packet Format Configuration

(1) TX Preamble Pattern

- Type Select：Preamble 模式选择，有 SYNCWORD MSB、1-byte Pattern、2-byte Pattern，三种可供选择
- Pattern Value：Pattern 值设定，可设定范围 0x0000~0xFFFF
- Length：Preamble 的长度设定，可设定范围 1~255

(2) Payload Data Length：Payload 数据长度设定，可设定范围 1~64。

(3) SYNC Pattern

- SYNCWORD Field Length：SYNCWORD 长度(byte)选择，共有 1、2、3、4、5、6、7、8 个 byte 可供选择

  注：SYNCWORD 的值需通过 SYNC_L Value 和 SYNC_H Value 两个 4-byte 的变量存放，若 SYNCWORD 长度在 4-byte 及以下，则只需对 SYNC_L Value 写值，否则超出 4-byte 的部分则需写入 SYNC_H Value。
- SYNC_L Value：SYNCWORD 低位值设定，可设定范围 0x00000000~0xFFFFFFFF
- SYNC_H Value：SYNCWORD 高位值设定，可设定范围 0x00000000~0xFFFFFFFF
- SYNCWORD 由 BCH 计算：SYNCWORD 计算开关

(4) Trailer Field Enable：Trailer 功能开关。

(5) Header Field Enable

- Header Length：Header 长度选择，有 1-byte、2-byte 两种可供选择
- Header Address：Header 地址设定，可设定范围 00000~16383

(6) Length Field Enable：Length 功能开关。

(7) CRC Field Enable

- Format：CRC 格式选择，有 CCIT-16、IBC-16 两种可供选择
- Inverted Enable：反向功能开关
- Byte Transmission order：Byte 传输方式选择，有 High Byte First、Low Byte First 两种可供选择
- Bit Transmission order：Bit 传输方式选择，有 MSB First、LSB First 两种可供选择
- CRC Seed：CRC Seed 值设定，可设定范围 0x0000~0xFFFF

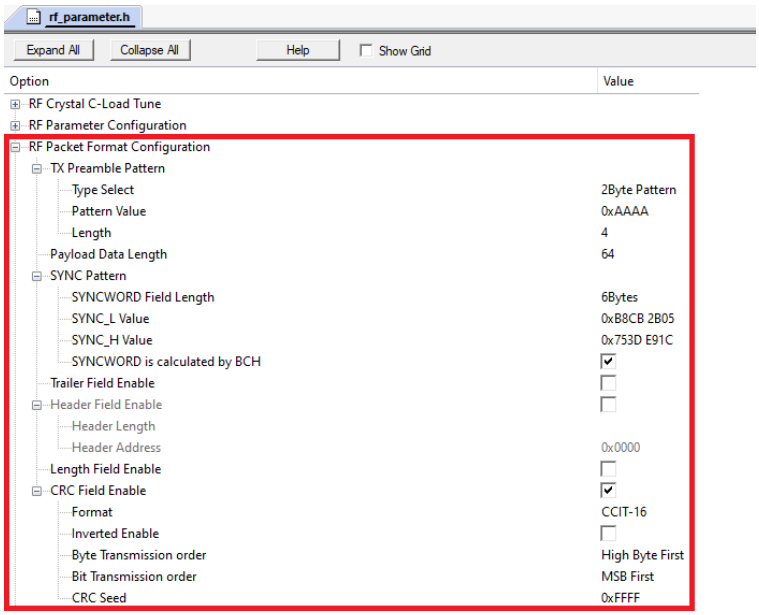

图 19. 封包格式设定

4. RF Packet Code Configuration

(1) Manchester Code Enable：Manchester 编码功能开关。

(2) Hamming Code (FEC) Enable：Hamming 编码功能开关。

(3) Examine Header Address Enable：Examine Header Address 功能开关。

(4) Whitening Code Enable

- Format：Whitening Code 格式选择，有 360X、PN7、PN9-CCIT、PN9-IBM 可供选择
- Whitening Seed (12 bit)：Whitening Seed 值设定，可设定范围 0x000~0x1FF

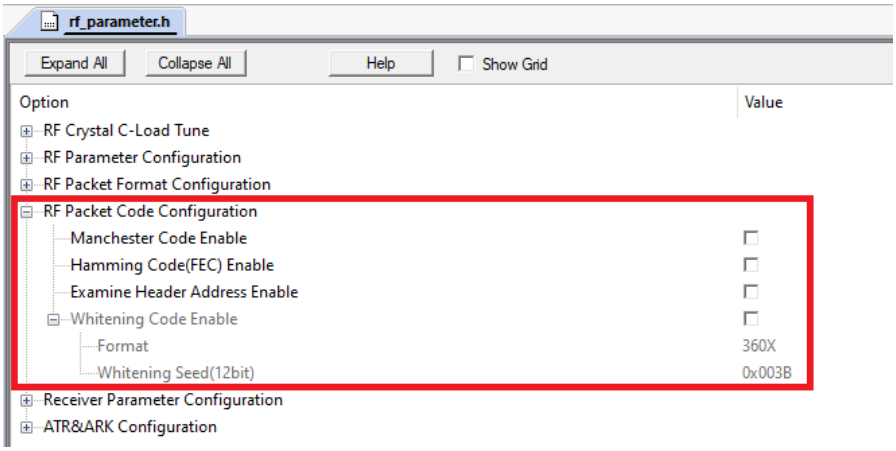

图 20. 封包编码功能设定

5. Receiver Parameter Configuration

Continue Mode Enable：连续接收模式开关。

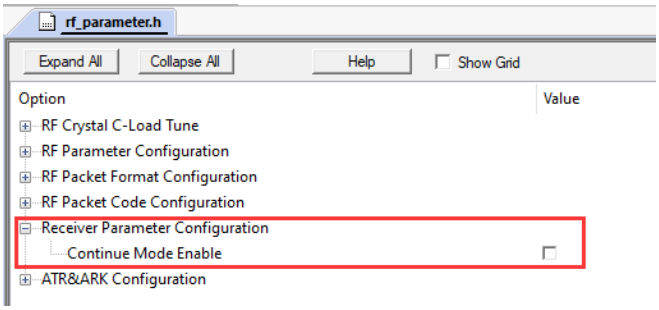

图 21. 接收参数设定

6. ATR&ARK Configuration

(1) ATR/ARK Time unit：ATR/ARK 时间选择，有 250μs、1ms 可供选择。

(2) Automatic Acknowledge(ARK) Enable

- Auto-ACK RX waiting time (unit:ms)：ARK RX 等待时间设定
- Auto resend times：自动重发次数设定，可设定范围 0~15

(3) Automatic TX/RX(ATR) Enable

- ATR clock frequency selection：ATR 时钟频率选择，有 2048Hz、4096Hz、8192Hz、32768Hz 可供选择

- ATR cycle period (unit:ms)：ATR 周期时间设定
- ATR RX active period time (unit:ms)：ATR RX 使能周期时间设定
- ATR RX active extend period time (unit:ms)：ATR RX 使能扩展周期时间设定

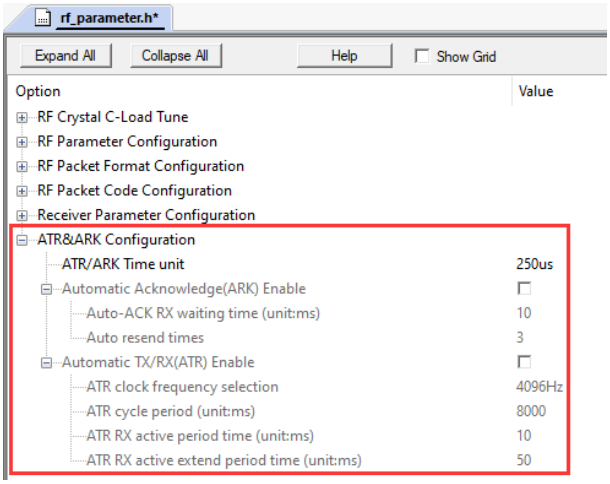

图 22. ATR&ARK 功能设定

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 BC3603 基本功能初始化(SPI、Crystal、VCO、LIRC、GIO Current…)。
2. 等待计数器：主循环将会每 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 BC3603 硬件 IRQ 引脚状态是否为低电平。如果为低电平，表示此时 IRQ 标志立起来，将目前 IRQ 状态储存到寄存器并将 BC3603 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。

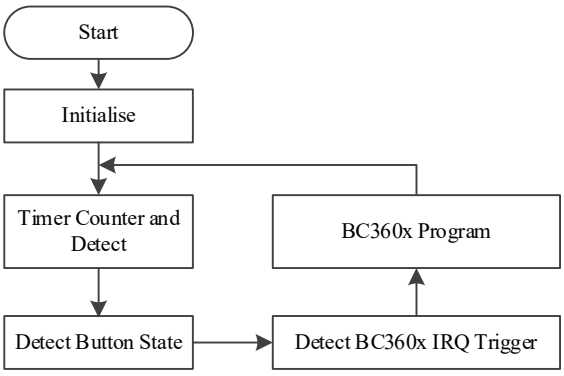

图 23. 主程序流程图

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。

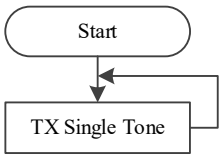

图 24. TX Carry 流程图

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/接收直到主循环将它停止。

- **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 IRQ 信号进来，并复位 TX FIFO 指针回到步骤 0 执行 TX。

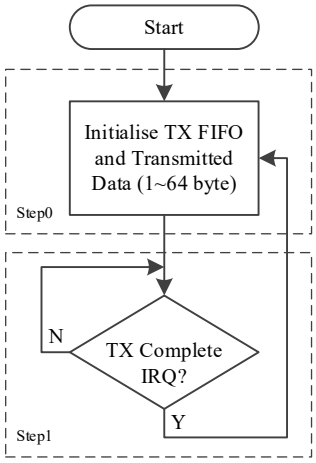

图 25. Simple FIFO_TX 流程图

- **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 指针，并且开启 RX 接收数据。

  步骤 1：等待 RX 接收完成/错误 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 指针清空，并开启 RX 模式继续执行。

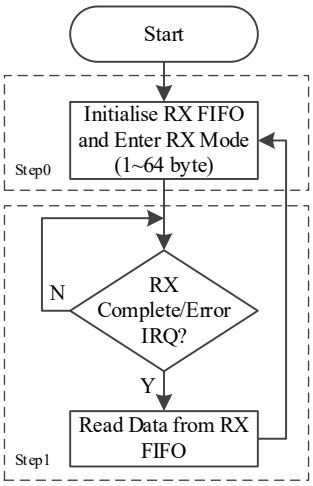

图 26. Simple FIFO_RX 流程图

Extend FIFO 模式

该程序提供封包 65~255 bytes 数据长度供用户传送/接收，此程序将会持续传送/接收直到主循环将它停止。

- **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。

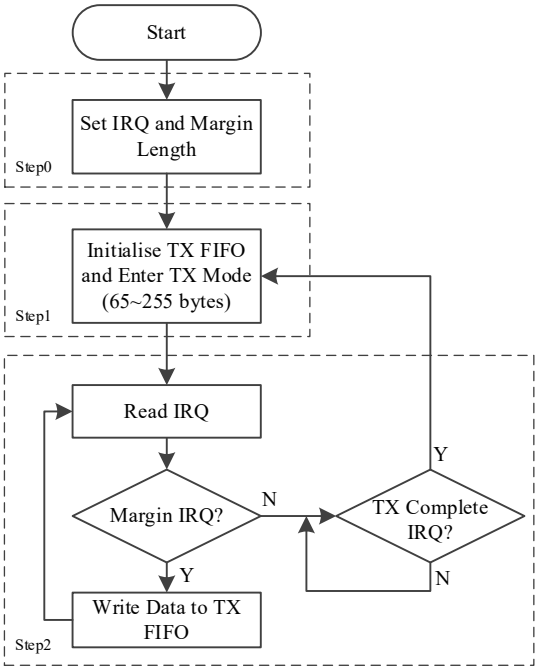

图 27. Extend FIFO_TX 流程图

- **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 接收数据。

步骤 2：等待 RX 完成/错误和 Margin IRQ 信号进来；若是 RX 完成/错误 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 数据并再次接收数据直到 RX 完成/错误 IRQ 立起。

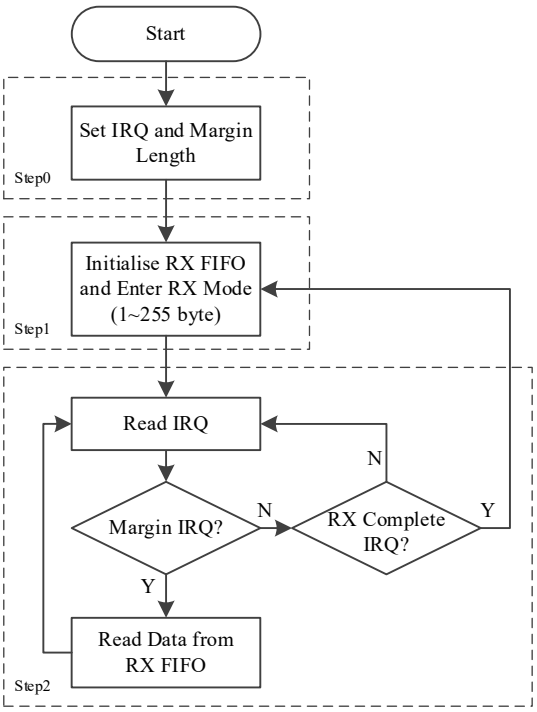

图 28. Extend FIFO_RX 流程图

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，直到主循环将它停止。

- **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 接收错误 IRQ 功能，设定封包数据与长度，并且将 TX、RX FIFO 指针复位后，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/错误 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。

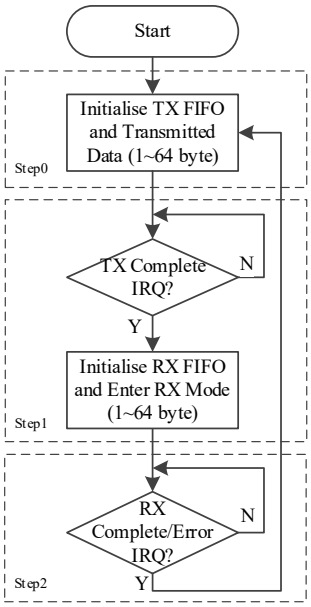

图 29. PER_Master 模式流程图

- **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 接收错误 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/错误 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：设定封包数据与长度，并且将 TX FIFO 指针复位，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。

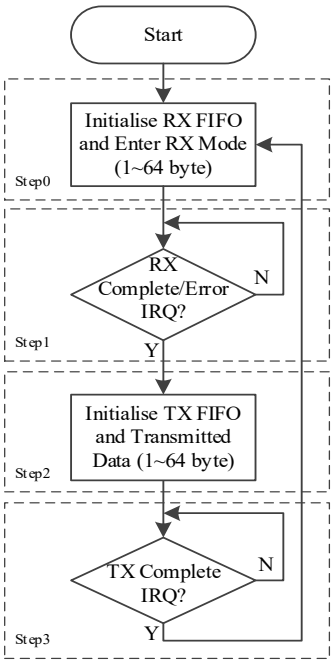

图 30. PER_Slave 流程图

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 指针，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 信号进来后发送数据，并在此步骤重复执行。

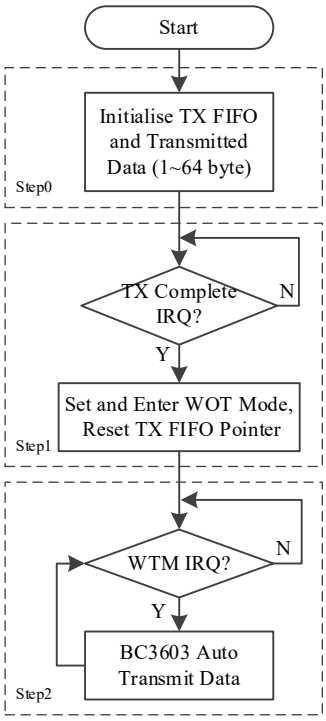

图 31. ATR_WOT 流程图

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 可以同步才设定 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 模式接收数据进来。

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

步骤 2：等待 IRQ 信号出现，进入下一个步骤。

步骤 3：判断是否为 RX 事件检测 IRQ 信号。若是 RX 事件检测 IRQ 立起，则继续判断 RX 完成/错误 IRQ 是否立起，若 RX 完成/错误 IRQ 立起，则将复位 RX FIFO 指针，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 BC3603 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。

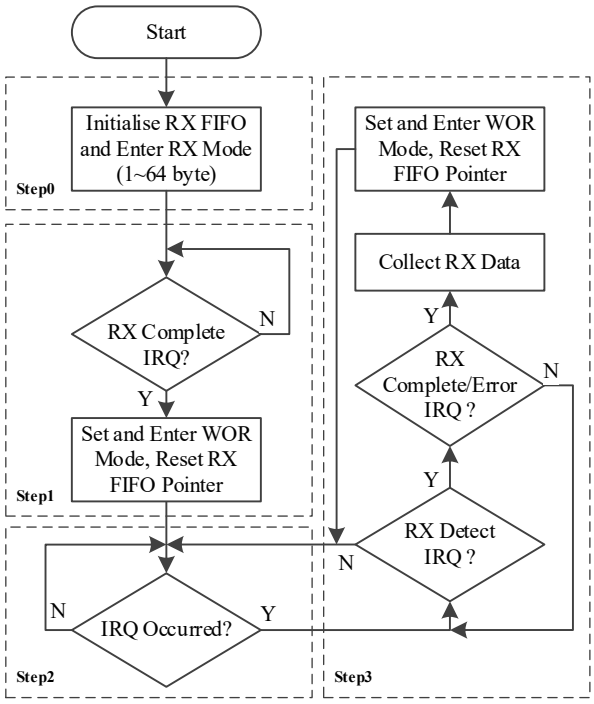

图 32. ATR_WOR 流程图

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 模式才设定，单纯 WOR 模式在步骤 1 后半部和步骤 3 判断 RX 事件检测及 RX 完成/错误 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 BC3603 传送/接收数据后进入 ARS(自动重发)/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 指针、PID 计数器更新和设定 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应答；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。

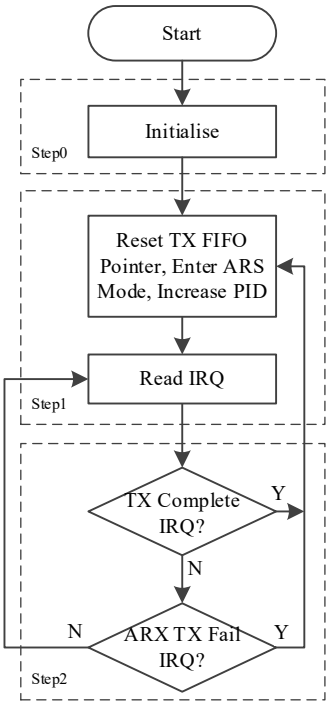

图 33. ARK_ARS 流程图

- **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：初始化所需 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：等待 RX 成功 IRQ 信号进来，若是 RX 成功代表 BC3603 收到更新过 PID 正确封包，此时将复位 RX FIFO 指针并且继续接收数据进来直到中止(下 Light Sleep 指令)AAK 模式。

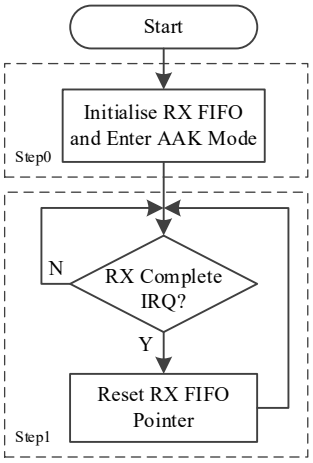

图 34. ARK_AAK 流程图

BC3603 API 函数介绍

范例程序中含 BC3603 API，主要分为三大类：命令类、一般类及命令自动收发类，下面便介绍各函数使用方式。

命令类

此类指令主要是 BC3603 内建快捷指令函数。

void BC3603_SoftwareReset(void)

输出：无

输入：无

功能：将 BC3603 复位

说明：范例程序中，执行此函数后，BC3603 寄存器、状态将会被复位(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 BC3603_RegisterBank(RF_BANK_Enum Bank)

输出：无

输入：Bank 为 BANK 区(编号)

功能：切换要访问的 BANK 区域

说明：范例程序中，执行此函数后，将会切换要访问 BANK 区域

```
typedef enum
{
  REGS_BANK0  =(u8)0,              /* register bank 0 */
  REGS_BANK1  =(u8)1,              /* register bank 1 */
  REGS_BANK2  =(u8)2,              /* register bank 2 */
  REGS_BANK3  =(u8)3               /* register bank 3 */
}  RF_BANK_Enum;
```

void BC3603_EnterDeepSleep(void)

输出：无

输入：无

功能：使 BC3603 进入深度休眠模式

说明：BC3603 当前状态在 Light Sleep 和 IDLE 模式下才能执行此函数，执行后会使 BC3603 进入深度休眠模式；请参考下图或 Datasheet 中 State Machine 章节；在深度休眠模式下，BC3603 只对 SPI 有反应

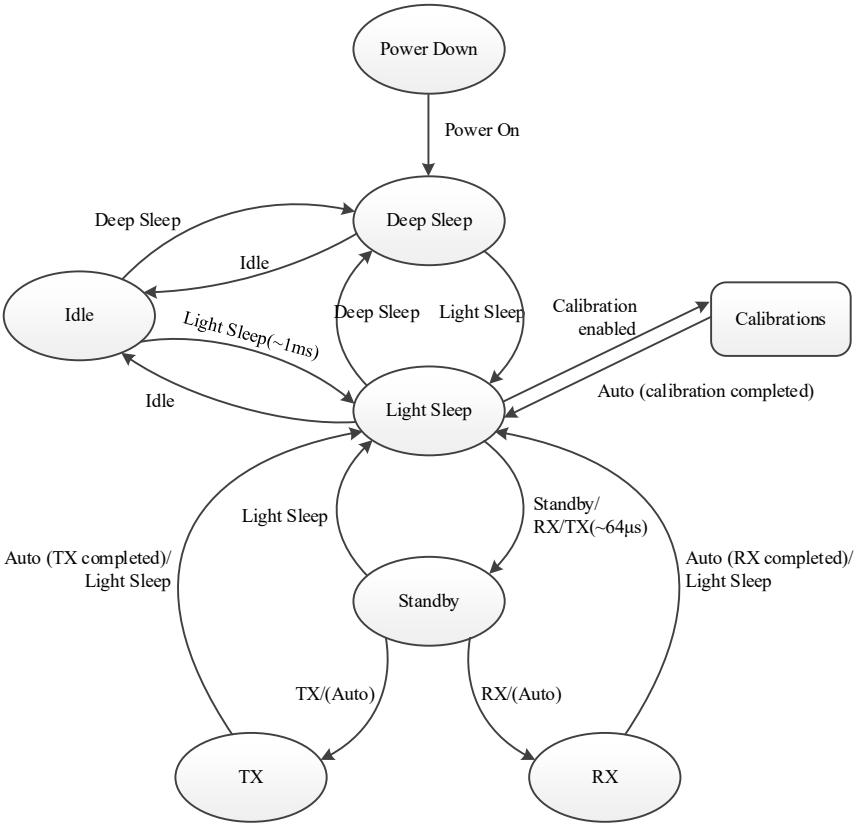

图 35. FIFO Mode State Diagram

void BC3603_EnterIdleMode(void)

输出：无

输入：无

功能：使 BC3603 进入空闲模式

说明：BC3603 当前状态在 Light Sleep 和 Deep Sleep 模式下才能执行此函数，执行后会使 BC3603 进入空闲模式；请参考图 35 或 Datasheet 中 State Machine 章节

void BC3603_EnterLightSleep(void)

输出：无

输入：无

功能：使 BC3603 进入浅层休眠模式

说明：BC3603 当前状态除了 Power Down 模式下不能进入浅层休眠模式，其余状态执行此函数都能使 BC3603 进入浅层休眠模式；请参考图 35 或 Datasheet 中 State Machine 章节

void BC3603_EnterStandby(void)

输出：无

输入：无

功能：使 BC3603 进入待机模式

说明：BC3603 只有在 Light Sleep 模式下执行此函数，才能进入待机模式，其余状态执行此函数都不能使 BC3603 进入待机模式；请参考图 35 或是 Datasheet 中 State Machine 章节

void BC3603_EnterTransmit(u8 FSAddr)

输出：无

输入：FSAddr 为 FIFO 初始地址

功能：使 BC3603 进入发射模式

说明：BC3603 只有在 Light Sleep 模式下执行此函数，才能进入发射模式，其余状态执行此函数都不能使 BC3603 进入发射模式，一旦在 Light Sleep 模式下执行此函数，BC3603 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，BC3603 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 35 或是 Datasheet 中 State Machine 章节

void BC3603_EnterReceiver(void)

输出：无

输入：无

功能：使 BC3603 进入接收模式

说明：BC3603 只有在 Light Sleep 模式下执行此函数，才能进入接收模式，其余状态执行此函数都不能使 BC3603 进入接收模式，一旦在 Light Sleep 模式下执行此函数，BC3603 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，BC3603 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 35 或 Datasheet 中 State Machine 章节

void BC3603_ResetRxFIFOPosition(void)

输出：无

输入：无

功能：复位 BC3603 接收 FIFO 指针

说明：执行此函数后，BC3603 接收 FIFO 指针会回到起始位置，BC3603 接收 FIFO 指针为自动累加，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数，确保 BC3603 接收 FIFO 指针处于起始位置

void BC3603_ResetTxFIFOPosition(void)

输出：无

输入：无

功能：复位 BC3603 发射 FIFO 指针

说明：执行此函数后，BC3603 发射 FIFO 指针会回到起始位置，BC3603 发射 FIFO 指针为自动累加，使用者并不能直接指定或控制；使用者在发射新封包前请务必执行此函数，确保 BC3603 发射 FIFO 指针处于起始位置

u8 BC3603_ReadReceiverPayload(u8 *Payload)

输出：RX 数据长度

输入：Payload 为指向 RX 数据储存 buffer 的指针变量

功能：读取 BC3603 RX FIFO 数据

说明：在 RX 接收 IRQ Flag 立起后，需运行此函数读取 BC3603 接收数据

void BC3603_WriteTransmitPayload(u8 *Payload,u8 Length)

输出：无

输入：Payload 为指向 TX 数据储存 buffer 的指针变量，Length 为 TX 数据长度

功能：写入 BC3603 TX FIFO 数据

说明：执行该函数首先会对 PID 进行累加，然后会根据 Length 配置 RF 的 TX 数据长度，并将该长度的数据从 Payload 所指数据区内写入到 RF 内部 FIFO 之中，实现 RF TX 前所有数据准备工作

void BC3603_SetSyncWord(u8 *SyncWord,u8 Length)

输出：无

输入：SyncWord 为指向 SYNCWORD 长度的指针变量，Length 为最大 SYNCWORD 长度

功能：设定 SYNCWORD 长度

说明：运行此函数来设定需要的 SYNCWORD 长度，若设定的 SYNCWORD 长度超过最大长度，则按最大长度进行设定

void BC3603_GetSyncWord(u8 *SyncWord,u8 Length)

输出：无

输入：SyncWord 为指向 SYNCWORD 长度的指针变量，Length 为最大 SYNCWORD 长度

功能：读取 SYNCWORD 长度

void BC3603_WriteTxFIFO(u8 *FIFO,u8 Length)

输出：无

输入：FIFO 为指向 TX 数据储存 buffer 的指针变量，Length 为 TX 数据长度

功能：将要传送数据时，需运行此函数将数据写入 FIFO 寄存器

一般类

此类指令可针对 BC3603 内部各项参数设定函数。

ErrStatus BC3603_Configures(RF_InitTypeDef *RF_InitStruct)

输出：参数设定结果标志位

输入：RF_InitStruct 为指向 RF_InitTypeDef 结构体的指针变量

功能：初始化 BC3603 所有射频参数设定

说明：运行此函数后，BC3603 所有 RF 参数会按照参数清单(rf_parameter.h 档案)进行设定

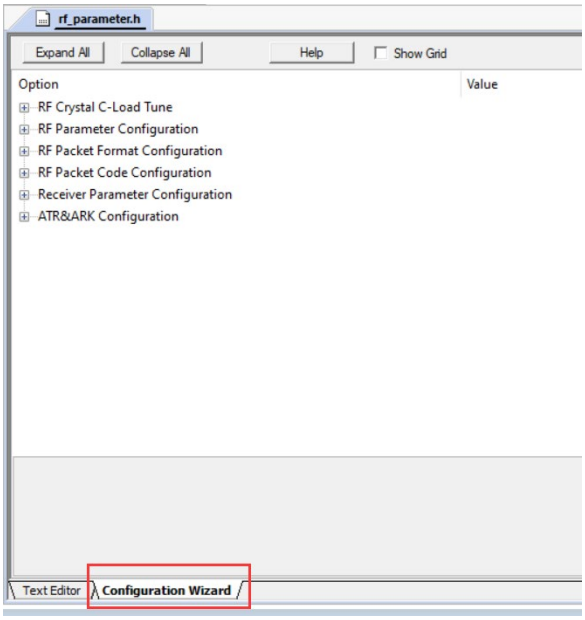

图 36. 参数设定界面

ErrStatus BC3603_LIRCCalibration(void)

输出：LIRC 校准结果标志位

输入：无

功能：LIRC 校准

说明：运行此函数后，将自动校准 LIRC 并且等待校正完毕

ErrStatus BC3603_VCOAutoCalibration(void)

输出：VCO 校准结果标志位

输入：无

功能：VCO 校准

说明：运行此函数后，将自动校准 VCO 并且等待校正完毕

ErrStatus BC3603_WaitCrystalReady(void)

输出：晶振校准结果标志位

输入：无

功能：等待晶振稳定

说明：运行此函数后，将确认 BC3603 XCLK 是否准备就绪，若是晶振无法起振，则会停在此函数无法往下执行

void BC3603_SetGIOFunction(RF_GIOS_Enum GIO,RF_GIOF_Enum Fun)

输出：无

输入：GIO 为欲操作 GIO 引脚(编号)，Fun 为指定 GIO 功能(编号)

功能：设定 BC3603 GIO 功能

说明：各 GIO 端口的功能将由此函数进行设定；BC3603 有 GIO1~GIO3 总共 3 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 功能较为齐全(详情请至 BC3603 Datasheet 查询)

```
typedef enum
{
  RF_GIO1 =(u8)0,
  RF_GIO2 =(u8)1,
  RF_GIO3 =(u8)2,
  RF_GIO4 =(u8)3
} RF_GIOS_Enum;
```

```
typedef enum
{
  RF_GIO_INPUT    =(u8)0x00,          /* GPIO input mode */
  RF_GIO_MISO     =(u8)0x01,          /* 4-wire SPI data output */
  RF_GIO_TRXD     =(u8)0x02,          /* Direct mode Tx/Rx data input/output */
  RF_GIO_IRQ      =(u8)0x05,          /* interrupt request output */
  RF_GIO_LOSCI    =(u8)0x06,          /* ATR clock external input */
  RF_GIO_TRXDMO   =(u8)0x07,          /* Tx/Rx data monitor output */
  RF_GIO_TBCLK    =(u8)0x08,          /* Tx clock output */
  RF_GIO_RBCLK    =(u8)0x09,          /* Rx(recovery) clock output */
  RF_GIO_FCLKO    =(u8)0x0A,          /* Frequency clock output */
  RF_GIO_LIRCO    =(u8)0x0B,          /* internal LIRC clock output */
  RF_GIO_EPACO    =(u8)0x0C,          /* External PA control output */
  RF_GIO_ELNACO   =(u8)0x0D,          /* External LNA control output */
  RF_GIO_TRBCLK   =(u8)0x0E,          /* Direct mode Tx/Rx bit clock output */
  RF_GIO_OSCCTL   =(u8)0x0F           /* External OSC control output */
} RF_GIOF_Enum;
```

void BC3603_IRQConfig(u8 IrqBitMap,ControlStatus NewState)

输出：无

输入：IrqBitMap 为 IRQ 功能选择(编号)，NewState 为 Enable/Disable

功能：设定 BC3603 IRQ 功能并使能

说明：通过运行此函数设定 BC3603 IRQ 功能，包含 TX 成功发射、RX 接收完成、校正完成、RX 事件接收、RX 接收错误、FIFO 门槛、ATR 时间和 ARK 发射失败功能(详情请至 BC3603 Datasheet 查询)

```
/* Definitions of Interrupt control register */
#define IRQ_TXCMPIE             (u8)(1 << 0)
#define IRQ_RXCMPIE             (u8)(1 << 1)
#define IRQ_CALCMPIE            (u8)(1 << 2)
#define IRQ_RXDETIE             (u8)(1 << 3)
#define IRQ_RXERRIE             (u8)(1 << 4)
#define IRQ_FIFOLTIE            (u8)(1 << 5)
#define IRQ_ATRCTIE             (u8)(1 << 6)
#define IRQ_ARKTFIE             (u8)(1 << 7)
```

void BC3603_SetCrystalCload(u8 xshift,u8 xtim)

输出：无

输入：xshift 为负载电容粗调值，xtim 为负载电容细调值

功能：设定晶振负载电容值(C-Load)

void BC3603_SetPacketField(RF_FieldTypeDef *Field)

输出：无

输入：Field 为指向欲设定的参数类型(编号)的指针变量

功能：设定 BC3603 封包相关参数

说明：运行此函数，对需要的封包参数进行设定，包括 trailer、syncword、CRC field 等

```
typedef struct
{
  RF_PMPATWD_Enum   PreambleClass;     /* preamble class select */
  u16               PreamblePattern;   /* preamble pattern */
  RF_SYNCLEN_Enum   SyncWord;          /* syncword length 1~8bytes */
  ControlStatus     Trailer;           /* payload trailer field enable/disable */
  RF_PLHF_Enum      Head;              /* payload header field enable/disable/length */
  ControlStatus     Length;            /* payload length field enable/disable */
  RF_CRCFMT_Enum    CRC;               /* payload CRC field enable/disable/type */
  RF_CRCPOL_Enum    CRCInv;            /* CRC data Inverse */
  RF_BYTEOD_Enum    CRCByteOD;         /* CRC byte Transmission order */
  RF_BITOD_Enum     CRCBitOD;          /* CRC bit Transmission order */
  u16               CRCSeed;           /* CRC seed */
} RF_FieldTypeDef;
```

void BC3603_FrequencyConfig(RF_BAND_Enum Band,u32 Frequency)

输出：无

输入：Band 为 BC3603 工作频段，Frequency 为具体工作频率

功能：设定 BC3603 欲发送/接收射频频率

说明：通过此函数设定工作频段&具体频率；BC3603 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定(每个频段都有可以调整范围，该范围数值请参考 Datasheet)

void BC3603_SetTxPower(RF_BAND_Enum Band,RF_TXPWR_Enum Power)

输出：无

输入：Band 为 BC3603 工作频段，Power 为发射功率(编号)

功能：设定 BC3603 发射功率

说明：通过此函数设定发射功率；目前函数支持 0dBm、+10dBm、+13dBm、+17dBm 和 +19dBm

```
typedef enum
{
  TXPOWER0DBM    =(u8)0,      /* Tx Power 0dBm */
  TXPOWER10DBM   =(u8)1,      /* Tx Power 10dBm */
  TXPOWER13DBM   =(u8)2,      /* Tx Power 13dBm */
  TXPOWER17DBM   =(u8)3,      /* Tx Power 17dBm */
  TXPOWER19DBM   =(u8)4       /* Tx Power 19dBm */
} RF_TXPWR_Enum;
```

void BC3603_DataRateConfig(RF_BAND_Enum Band,RF_DATARATE_Enum DataRate)

输出：无

输入：Band 为 BC3603 工作频段，DataRate 为发射/接收数据速度(编号)

功能：设定 BC3603 发射/接收速率

说明：通过此函数设定发射/接收速率；目前支持 2Kbps、10Kbps、50Kbps、125Kbps 和 250Kbps 数据速度

```
typedef enum
{
  DATARATE2K     =(u8)0,      /* data rate 2Kbps */
  DATARATE10K    =(u8)1,      /* data rate 10Kbps */
  DATARATE50K    =(u8)2,      /* data rate 50Kbps */
  DATARATE125K   =(u8)3,      /* data rate 125Kbps */
  DATARATE250K   =(u8)4       /* data rate 250Kbps */
} RF_DATARATE_Enum;
```

ErrStatus BC3603_DeInit(void)

输出：初始化结果标志位

输入：无

功能：初始化 MCU 与 BC3603 沟通接口(SPI、GIO)，初始化寄存器值

说明：运行此函数后，将初始化所有寄存器的值，并建立 MCU 与 BC3603 间 SPI 通信，设定各 GIO 口的功能

u8 BC3603_ReadRegister(u8 Addr)

输出：1 byte 数据(代表 BC3603 此时欲读出寄存器数值)

输入：Addr 为欲读出的寄存器地址

功能：查看 BC3603 寄存器数值

void BC3603_WriteRegister(u8 Addr,u8 Data)

输出：无

输入：Addr 为欲写入的寄存器地址，Data 为欲写入的寄存器数值

功能：写入 BC3603 寄存器数值

FlagStatus BC3603_GetIRQFlagStatus(u8 FlagBitMap)

输出：IRQ 事件的状态

输入：IRQ 事件(编号)(可参考 Datasheet IRQ3 寄存器)

功能：查看 BC3603 此时 IRQ 状态

说明：通过运行此函数查看 BC3603 IRQ 状态，进而得知 BC3603 完成哪些事件触发

void BC3603_ClearIRQFlag(u8 FlagBitMap)

输出：无

输入：IRQ 事件(编号)(可参考 Datasheet IRQ3 寄存器)

功能：清除指定 IRQ 事件 IRQ 状态

说明：当 BC3603 触发事件发生时，IRQ 状态将会立起，需通过运行此函数将该状态清除才能等待下一个状态发生

```
/* Definitions of Interrupt status register */
#define IRQ_TXCMPIF         (u8)(1 << 0)
#define IRQ_RXCMPIF         (u8)(1 << 1)
#define IRQ_CALCMPIF        (u8)(1 << 2)
#define IRQ_RXDETIF         (u8)(1 << 3)
#define IRQ_RXERRIF         (u8)(1 << 4)
#define IRQ_FIFOLTIF        (u8)(1 << 5)
#define IRQ_ATRCTIF         (u8)(1 << 6)
#define IRQ_ARKTFIF         (u8)(1 << 7)
```

RF_OMST_Enum BC3603_GetOperationState(void)

输出：1 byte 数据(代表 BC3603 此时状态，可参考 Datasheet STA1 寄存器)

输入：无

功能：查看 BC3603 此时状态(State Machine)

说明：BC3603 运作模式(State Machine)转换有固定的流程，请务必遵照图 35 操作

u8 BC3603_GetSYNCRSSIValue(void)

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

输入：无

功能：查看 BC3603 此时接收到正确 SYNCWORD 封包 RSSI 数值

说明：RSSI 数值有分为两种：环境与接收，运行此函数读取接收到正确 SYNCWORD 封包的 RSSI 数值

u8 BC3603_GetRSSIValue(void)

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

输入：无

功能：查看 BC3603 此时环境中 RSSI 数值

说明：RSSI 数值有分为两种：环境与封包，此函数读取到为环境中 RSSI 数值，运行此函数时需确认 BC3603 是处在 RX 模式当中

void BC3603_SetPacketCodec(RF_CodecTypeDef *Codec)

输出：无

输入：Codec 为指向编码功能类型的指针变量

功能：使能各项编码功能

说明：通过运行此函数配置编码功能，包含曼彻斯特编码使能、汉明码使能、数据白化使能等

```
typedef struct
{
  ControlStatus     Manchester;              /* manchester code enable/disable */
  ControlStatus     HammingCode;             /* hamming code enable/disable */
  ControlStatus     HeadAddrExamine;         /* examine header address */
  RF_WHTFMT_Enum    WhiteningCodec;          /* whitening codec type */
  u16               WhiteningSeed;           /* whitening seed */
} RF_CodecTypeDef;
```

void BC3603_ExtendFIFOConfig(RF_FFMGS_Enum FFMgs,ControlStatus NewState)

输出：无

输入：FFMgs 为 FIFO 长度(编号)，NewState 为 Enable/Disable

功能：使能 Extend FIFO 功能，配置长度边距检测 byte 数

说明：在需要使用 Extend FIFO 模式时，通过此函数来使能并配置 FIFO 长度边距检测功能

```
typedef enum
{
  FFMG4BYTE    =(u8)0,      /* FIFO length margin 4bytes */
  FFMG8BYTE    =(u8)1,      /* FIFO length margin 8bytes */
  FFMG16BYTE   =(u8)2,      /* FIFO length margin 16bytes */
  FFMG32BYTE   =(u8)3       /* FIFO length margin 32bytes */
} RF_FFMGS_Enum;
```

void BC3603_InfiniteFIFOControl(RF_FFMGS_Enum FFMgs,ControlStatus NewState)

输出：无

输入：FFMgs 为 FIFO 长度(编号)，NewState 为 Enable/Disable

功能：使能 Infinite FIFO 功能，配置长度边距检测 byte 数

说明：在需要使用 Infinite FIFO 模式时，通过此函数来使能并配置 FIFO 长度边距检测功能

```
typedef enum
{
  FFMG4BYTE    =(u8)0,      /* FIFO length margin 4bytes */
  FFMG8BYTE    =(u8)1,      /* FIFO length margin 8bytes */
  FFMG16BYTE   =(u8)2,      /* FIFO length margin 16bytes */
  FFMG32BYTE   =(u8)3       /* FIFO length margin 32bytes */
} RF_FFMGS_Enum;
```

FlagStatus BC3603_GetInfiniteFIFOStatus(void)

输出：FFINF_EN 的状态

输入：无

功能：读取上面 Infinite FIFO 模式的使能状态

FlagStatus BC3603_GetPowerOnFlag(void)

输出：上电标志位状态

输入：无

功能：读取上电标志位

void BC3603_ClearPowerOnFlag(void)

输出：无

输入：无

功能：清除上电标志位

void BC3603_SetFSYCKPrescaler(RF_FSYCK_Enum CKPRE,ControlStatus NewState)

输出：无

输入：CKPRE 为分频倍数，NewState 为 Enable/Disable

功能：设定 GIOx 在晶振频率输出时的分频功能

说明：GIOx 引脚可配置为输出晶振频率的功能，可根据需要对输出的频率进行分频，有 1、2、4、8 共 4 个分频系数供选择

```
typedef enum
{
  FSYCK_DIV1      =(u8)0,      /* FSYCK = XCLK divide 1 */
  FSYCK_DIV2      =(u8)1,      /* FSYCK = XCLK divide 2 */
  FSYCK_DIV4      =(u8)2,      /* FSYCK = XCLK divide 4 */
  FSYCK_DIV8      =(u8)3       /* FSYCK = XCLK divide 8 */
} RF_FSYCK_Enum;
```

void BC3603_SetReceiverDetectEvent(RF_RXDETS_Enum DETS)

输出：无

输入：DETS 为标志位的检测类型(编号)

功能：设定 RXDETS 标志位的检测类型

说明：RX 检测类型有 Carry 和 SYNCWORD 两种选择(可参考 Datasheet IRQ1 寄存器)

```
typedef enum
{
  DET_CARRY       =(u8)0,      /* RX detect event of carry */
  DET_SYNCWORD    =(u8)2       /* Rx detect event of sync-word */
} RF_RXDETS_Enum;
```

FlagStatus BC3603_GetReceiverErrorStatus(u8 FlagBitMap)

输出：接收错误标志位状态

输入：FlagBitMap 为错误标志位

功能：查询 RX 是否发生错误

说明：通过运行此函数，读取标志位 IRQ_RXCRCF、IRQ_RXFFOW 和 IRQ_RXTO，查看 RX 是否发生错误(可参考 Datasheet IRQ1 寄存器)

u8 BC3603_GetReceiverErrorFlag(void)

输出：标志位读取结果

输入：无

功能：读取接收错误标志位

说明：通过运行此函数，读取 IRQ_RXCRCF、IRQ_RXFFOW 和 IRQ_RXTO 三个标志位并返回结果

void BC3603_SetIRQPolarity(RF_IRQPOL_Enum POR,RF_IRQPOL_Enum CPOR)

输出：无

输入：POR 为 IRQPOR 极性选择，CPOR 为 IRQCPOR 极性选择

功能：设定 IRQ 信号和 IRQ 标志位清零极性

void BC3603_SetPreambleLength(u8 Length)

输出：无

输入：Length 为前导码长度

功能：设定封包前导码长度

void BC3603_SetTransmitLength(u8 Length)

输出：无

输入：Length 为发送数据长度

功能：设定发送数据长度

void BC3603_SetReceiverLength(u8 Length)

输出：无

输入：Length 为接收数据长度

功能：设定接收数据长度

u8 BC3603_GetReceiverLength(void)

输出：接收数据长度

输入：无

功能：读取接收数据长度

void BC3603_SetPacketID(u8 PID)

输出：无

输入：PID 为 PID 数值

功能：设定封包格式中的 PID，详情参考 Datasheet 中 PKT2 寄存器

u8 BC3603_GetPacketID(void)

输出：PID 数值

输入：无

功能：读取封包格式中的 PID，详情参考 Datasheet 中 PKT2 寄存器

void BC3603_SetHeaderAddress(u16 PLHA)

输出：无

输入：PLHA 为 PLHA 数值

功能：设定封包格式中的 PLH 有效载荷头码地址

u16 BC3603_GetHeaderAddress(void)

输出：PLHA 数值

输入：无

功能：读取封包格式中的 PLH 有效载荷头码地址，PLHA 和 PLHEA

命令自动收发类

此类指令则是为了能方便使用 BC3603 中自动收发功能所制定。

void BC3603_ATRKTimeUnit(RF_ATRTU_Enum Unit)

输出：无

输入：Unit 为 ATRK 单位时间，可有 250μs/1ms 选择

功能：设定 ATR 单位时间

说明：执行该函数会变更 ATR 的单位时间

```
typedef enum
{
  ATRTU_250US       = (u8)0,        /* unit:250us */
  ATRTU_1MS         = (u8)1         /* unit:1ms */
} RF_ATRTU_Enum;
```

void BC3603_ATRConfig(ATR_InitTypeDef *ATR_InitStruct)

输出：无

输入：ATR_InitStruct 为指向欲使用的 ATR 工作模式的指针变量

功能：配置 ATR 各项参数设定

```
typedef struct
{
  RF_ATRM_Enum          Mode;
  RF_ATRCKS_Enum        ClockSource;     /* internal LIRC clock or external clock */
  RF_ATRCKD_Enum        Prescaler;       /* Clock Prescaler Selection */
  u32           ClockFrequency;          /* external clock source frequency(unit:Hz) */
  u32           CycleCounter;            /* cycle period time(unit:us) */
  RF_ATRWDLY_Enum       WakeUpDelayTime;
} ATR_InitTypeDef;
```

void BC3603_ATRControl(ControlStatus NewState)

输出：无

输入：NewState 为 Enable/Disable

功能：开启/关闭 ATR 功能

void BC3603_ARKControl(ControlStatus NewState)

输出：无

输入：NewState 为 Enable/Disable

功能：开启/关闭 ARK 功能

void BC3603_ARKRepeatPeriod(u8 Repeat,u32 Period)

输出：无

输入：Repeat 为 TX 自动重发次数数值，Period 为 RX 自动应答时间数值

功能：设定 TX 自动重发次数及 RX 自动应答时间

u32 BC3603_ATRConvertTimeUnit(u32 Period,u32 Max,float Unit)

输出：寄存器配置值

输入：Period 为时间，Max 为最大值(参考如下)，Unit 为单位时间

功能：根据单位时间来转换计算寄存器配置值

说明：执行该函数将根据 Unit 单位时间来计算 Period 时间内时钟需要计数多少次，并且会以 Max 值作为限制最大输出

```
/**
 * @brief maximum value for ATR and ARK timer
   */
#define ATR_CYCLE_MAX           (65535)
#define ATR_RXACTIVE_MAX        (2047)
#define ATR_RXEXTEND_MAX        (65535)
#define ARK_REPEAT_MAX          (15)
#define ARK_RXACTIVE_MAX        (255)
```

void BC3603_SetWORPeriod(u32 RxActive,u32 RxExtend)

输出：无

输入：RxActive 为 ATRRXAP 数值，RxExtend 为 ATRRXEP 数值

功能：设定 RX 唤醒之后的工作时间(WOR 模式)

u16 BC3603_GetATRCycleValue(void)

输出：ATR 工作周期

输入：无

功能：读取 ATR 工作周期，详情参考 Datasheet 中 ATR2 和 ATR3 寄存器

u16 BC3603_GetATRCycleCounter(void)

输出：ATR 工作周期计数值

输入：无

功能：读取 ATR 工作周期计数值，详情参考 Datasheet 中 ATR9 和 ATR10 寄存器

void BC3603_UpdateATRCycleCounter(u16 NewCounter)

输出：无

输入：NewCounter 为 ATR 工作周期数值

功能：更新 ATR 工作周期计数值，详情参考 Datasheet 中 ATR9 和 ATR10 寄存器

结论

本文介绍了 Holtek 针对 BC3603 无线收发器所设计的固件开发板，通过此开发板可较轻松地了解此芯片，从而进一步开发出用户所需产品。

参考资料

参考文件 BC3603 Datasheet。

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

版本及修订说明

日期	作者	发行	修订说明
2023.04.25	李健钊	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。