目 录

BC5602 HT32 Standard Demo Code 使用手册

简介

Holtek 推出全新高效能 2.4GHz Tx/Rx 射频芯片 BC5602，BM5602-60-1 是一款基于 2.4G IC BC5602 开发的、高性能的 2.4G RF GFSK 收发模块，此模块支持双向数据传输，可广泛应用于 2.4GHz 频段的无线应用，应用领域包含各式开关遥控、办公室自动化以及智能居家无线控制应用。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。

BC5602 工作电压 1.9V~3.6V，可程序设定发射功率-10dBm~+6dBm；高接收灵敏度-98dBm@125Kbps，接收电流 17mA@250/500Kbps。具备低休眠电流 0.5μA，快速唤醒晶振的 Middle Sleep 模式，以及自动收发(ATR：Auto-Transmit-Receive)功能。

本篇文章将介绍如何使用 M0+系统开发板(BCE-GENTrx32-001)搭配 BM5602-60-1 模块配合多样化的 RF 接收/发射 Demo code 对 BC5602 进行各别功能的操作，可以让使用者针对 BC5602 的各别功能做了解，并有一个整体的系统架构。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。首先将介绍如何架设开发平台。

相关 BC5602 的技术资料请参考 Holtek 官网：https://www.holtek.com.cn/productdetail/-/vg/BC5602。

图 1、BM5602-60-1 模块

应用领域

各式开关遥控、办公室自动化以及智能居家无线控制应用。

开发平台说明

本开发平台使用以 Holtek 32-bit Arm® Cortex®-M0+单片机 HT32F52352 为主控制器的系统开发板 BCE-GENTrx32-001，搭配上 BC5602 模块 BM5602-60-1，完整平台结构如下图所示：

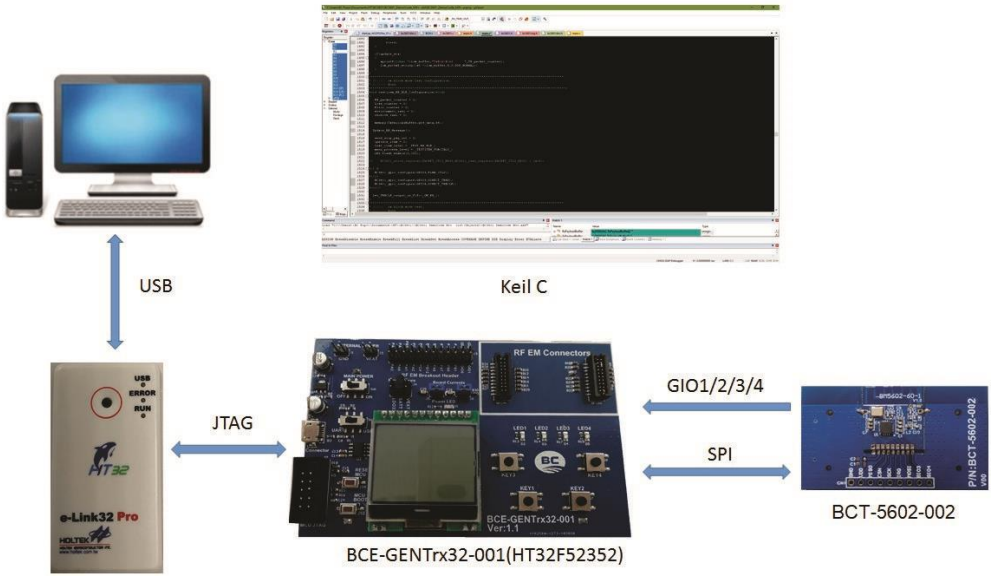


图 2、系统架构图

HT32F52352 是 Arm® Cortex®-M0+微控制器，使用者需要先在电脑上安装 keil uVisionIDE 接口，搭配 Holtk 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，方便使用者来操作，如下图所示。

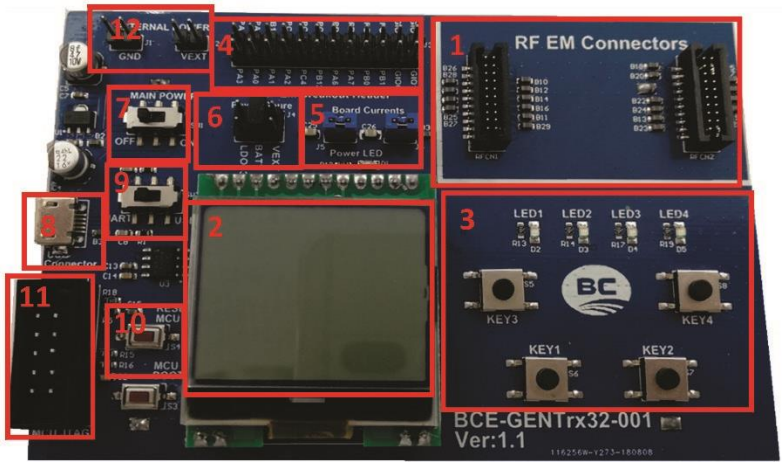


图 3、系统开发板组成

包含了：

1. 射频模块接口，是射频发射/接收装置连接处，在此范例则安装上 BM5602-60-1。

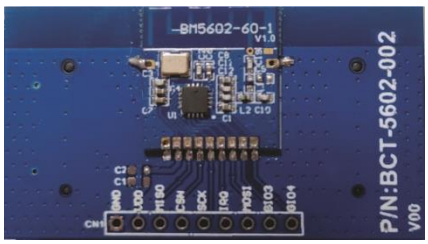


图 4、BCT-5602-002

2. 液晶显示器(支持 128×64 点)，使用方法请参考本文范例程序，内含此显示器控制程序库。
3. 指示灯：LED×4 及按键×4，使用者在程序开发时，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，其余由使用者定义。

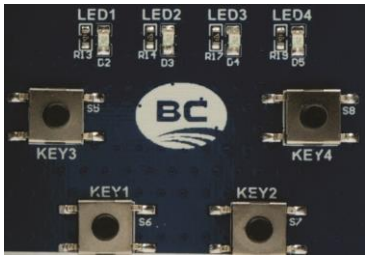


图 5、LED & KEY

4. I/O 接口，BCE-GNTrx32-001 的 I/O 与 BM5602-60-1 的对应引脚，详细 I/O 如下图所示：

BCE-GENTrx32-001	BM5602-60-1
PA3	CSN
PA0	SCK
PA1	SDIO
PA2	GIO1
PC4	GIO2
GIO2	GIO3
GIO3	GIO4

图 6、I/O 接口

5. MCU 和 BC5602 模块板电源电流检测点。
6. 系统电源选择，如图 7，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 9)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。

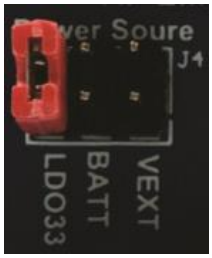


图 7、电源选择

7. 总电源开关，左拨(OFF)为关闭电源，右拨为开启电源。

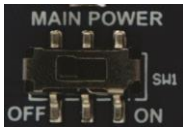


图 8、电源开关

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。选择 UART 时，将动态吐出 log，可由电脑串口软件观察 BC5602 状态。
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 9、外部电源接点

开发前须知

BC5602 范例程序目前是使用 Keil C 开发，以下将介绍如何使用 Keil C 软件去编译与下载 BC5602 范例程序。

范例程序种类

建置在 BCE-GENTrx32-001 开发平台上 BC5602 范例程序共有 7 种，分别为：TX Carry、PER Mode、SB Mode、ESB Mode、DPL、DYN_ACK 与 ACK PLD，有关各操作模式详细描述可参考 BC5602 Datasheet。

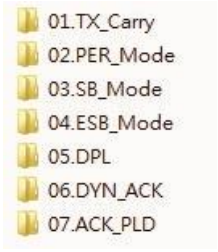


图 10、范例程序

1. TX Carry：该程序提供单一 RF 载波(无数据与调变)输出。
2. PER (Packet Error Rate) Mode：该程序分为 PTX 端与 PRX 端，持续传输 32-byte 长度封包，并显示 PER 在 PTX 端。
3. SB (ShockBurst) Mode：该程序分为 PTX 端与 PRX 端，单击发送一笔单向 32-byte 长度封包。
4. ESB (Enhanced ShockBurst) Mode：该程序分为 PTX 端与 PRX 端，单击发送一笔双向 32-byte 长度封包。
5. DPL (Dynamic Payload Length)：该程序分为 PTX 端与 PRX 端，单击发送一笔双向长度可调整的封包。
6. DYN_ACK (Dynamic ACK)：该程序分为 PTX 端与 PRX 端，单击发送一笔单向 32-byte 封包或双向 32-byte 封包。
7. ACK PLD (PRX acknowledge with payload)：该程序分为 PTX 端与 PRX 端，PRX 端收到后会回复 ACK+Payload 32-byte 封包。

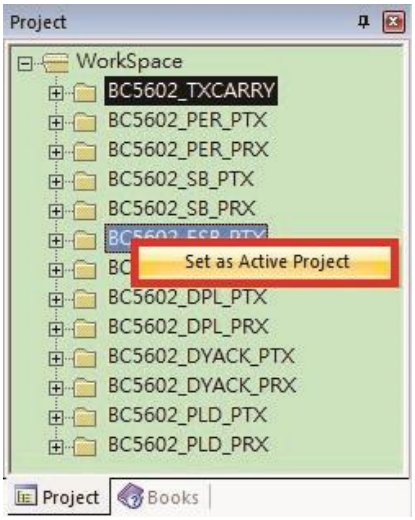
开发平台硬件安装

需准备 e-Link32 Pro 1 台、BEC-GENTrx32-001 开发板 1 块和 BM5602-60-1 模块 1 块，分别将 BM5602-60-1 安装至 BEC-GENTrx32-001 上，并把 e-Link32 Pro 连接头安装至 BEC-GENTrx32-001 后，开启 Keil C 即可开始研发。

图 11、安装开发平台硬件示意图

挑选范例程序

可以在每一个 Project 上按下鼠标右键选择要 Active Project。



范例程序简介

本范例共分七大类 13 个 Project，接下来会由参数介绍开始，循序介绍每组范例的操作流程与使用方式。

参数设定

每个 Project 的 folder 中都会有一个 main.c，每一个 main.c 开头均有参数设定，方便使用者可以一目了然此范例需要设定的参数为何。

```
///************************ RF parameters Setup value *********************************/
#define _DEFAULT_ADDWidth          AW_5BYTE      //AW_3BYTE / AW_4BYTE / AW_5BYTE
#define _DEFAULT_CRCSET            CRC_16BIT     //0:CRC_OFF / 1:CRC_8BIT / 2:CRC_16BIT
#define _DEFAULT_RF_Channel        2442          //2400~2485MHz
#define _DEFAULT_RF_Power          P6dBm         //N5dBm / P0dBm / P5dBm / P6dBm
#define _DEFAULT_DATA_RATE         DR125KBPS     //DR125KBPS / DR250KBPS / DR500KBPS
#define _DEFAULT_PKT_Length        32            //1~32
/* set FEATURE register */
#define _DEFAULT_DYN_ACK           DIS_DYN_ACK   //EN_DYN_ACK / DIS_DYN_ACK
#define _DEFAULT_ACK_PAY           DIS_ACK_PAY   //EN_ACK_PAY / DIS_ACK_PAY
#define _DEFAULT_DPL               DIS_DPL       //EN_DPL / DIS_DPL
/* set pipe dynamic length */
#define _DEFAULT_DPL_P0            DIS_DPL_P0    //EN_DPL_P0 / DIS_DPL_P0
#define _DEFAULT_DPL_P1            DIS_DPL_P1    //EN_DPL_P1 / DIS_DPL_P1
#define _DEFAULT_DPL_P2            DIS_DPL_P2    //EN_DPL_P2 / DIS_DPL_P2
#define _DEFAULT_DPL_P3            DIS_DPL_P3    //EN_DPL_P3 / DIS_DPL_P3
#define _DEFAULT_DPL_P4            DIS_DPL_P4    //EN_DPL_P4 / DIS_DPL_P4
#define _DEFAULT_DPL_P5            DIS_DPL_P5    //EN_DPL_P5 / DIS_DPL_P5
/* set pipe active */
#define _DEFAULT_PIPE0             DIS_P0        //EN_P0 / DIS_P0
#define _DEFAULT_PIPE1             DIS_P1        //EN_P1 / DIS_P1
#define _DEFAULT_PIPE2             DIS_P2        //EN_P2 / DIS_P2
#define _DEFAULT_PIPE3             DIS_P3        //EN_P3 / DIS_P3
#define _DEFAULT_PIPE4             DIS_P4        //EN_P4 / DIS_P4
#define _DEFAULT_PIPE5             DIS_P5        //EN_P5 / DIS_P5
/* set RX pipe auto ack */
#define _DEFAULT_AAP0              DIS_AAP0      //EN_AAP0 / DIS_AAP0
#define _DEFAULT_AAP1              DIS_AAP1      //EN_AAP1 / DIS_AAP1
#define _DEFAULT_AAP2              DIS_AAP2      //EN_AAP2 / DIS_AAP2
#define _DEFAULT_AAP3              DIS_AAP3      //EN_AAP3 / DIS_AAP3
#define _DEFAULT_AAP4              DIS_AAP4      //EN_AAP4 / DIS_AAP4
#define _DEFAULT_AAP5              DIS_AAP5      //EN_AAP5 / DIS_AAP5
/* set auto repeat counter & delay timer */
#define _DEFAULT_ARD               ARD2750       //
#define _DEFAULT_ARC               DIS_ARC       //
/* set rx pipe payload width */
#define DEFAULT_RX_PW_P0           32            //1~32Bytes
#define DEFAULT_RX_PW_P1           32            //1~32Bytes
#define DEFAULT_RX_PW_P2           32            //1~32Bytes
#define DEFAULT_RX_PW_P3           32            //1~32Bytes
#define DEFAULT_RX_PW_P4           32            //1~32Bytes
#define DEFAULT_RX_PW_P5           32            //1~32Bytes
///************************************************************************************/
```

图 12、参数清单

参数说明如下：

1. _DEFAULT_ADDWidth：Address 长度(byte)选择，共有 3、4、5 个 byte 可供选择。
2. _DEFAULT_CRCSET：CRC 开关与选择 8-bit 或 16-bit CRC 格式。
3. _DEFAULT_RF_Channel：射频频率选择，设定范围 2400~2485MHz。
4. _DEFAULT_RF_Power：RF 输出功率设定，有-5、0、+5、+6dBm，四种可供选择。
5. _DEFAULT_DATA_RATE：数据传输速率设定，有 125K、250K、500Kbps，三种可供选择。
6. _DEFAULT_PKT_Length：封包长度 1~32-byte。
7. _DEFAULT_DYN_ACK：使能“写 TX FIFO (No-Auto-ACK 模式)”命令。
8. _DEFAULT_ACK_PAY：带 Payload 的 PRX 应答功能使能。
9. _DEFAULT_DPL：动态 Payload 长度(Dynamic payload length)功能使能。
10. _DEFAULT_DPL_P0~_DEFAULT_DPL_P5：pip0~pipe5 的动态 Payload 长度控制。
11. _DEFAULT_PIPE0~_DEFAULT_PIPE5：使能 RX pipe0~pipe5。

12. _DEFAULT_AAP0~_DEFAULT_AAP5：使能 pipe0~pipe5 Auto-ACK 模式。
13. _DEFAULT_ARD：自动重发延迟时间 250μs~4000μs。
14. _DEFAULT_ARC：自动重发次数 1~15 次。
15. DEFAULT_RX_PW_P0~DEFAULT_RX_PW_P5：pipe0~pipe5 中 RX Payload 的字节数。

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 BC5602 基本功能初始化(BC5602 接口配置、BC5602 寄存器初始化)。
2. 按键动作：读取判断按键状态，KEY2 为开始与结束功能。KEY1 和 KEY3 在不同的范例程序会有不同的功用。
3. 检查 IRQ 状态：这边将检查 BC5602 硬件 IRQ 引脚状态。如果为低电平，表示 IRQ 旗标立起来，随即会将目前 IRQ 状态存储到寄存器，并将 BC5602 IRQ 状态复归。
4. 程序主循环(BC560x Program)：TX Carry、PER Mode、SB Mode、ESB Mode、DPL、DYN_ACK 和 ACK PLD 七大范例项目，后续会逐一介绍

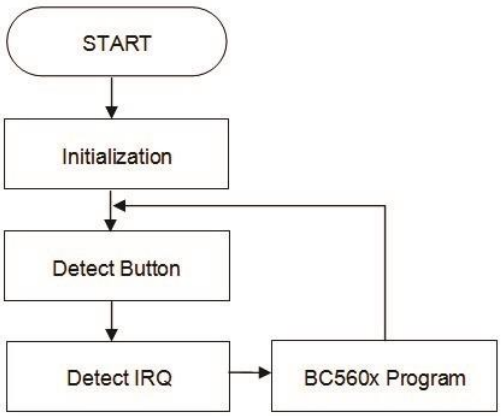


图 13、主程序流程图

TX Carry

提供 RF Single Tone (无数据与调变)输出，按下 KEY2 会持续输出载波直到再次按下 KEY2 停止。也可以使用 KEY1 和 KEY3 调整频率 UP 和 DOWN。

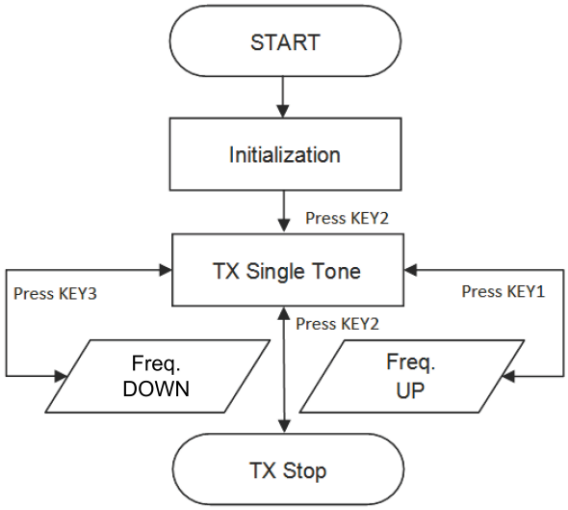


图 14、TX Carry 流程图

PER Mode

由 PTX 端连续发出 PN9 封包长度 32-byte 给 PRX 端，当 PTX 端收到 PRX 端发射的 ACK 封包即完成一次传输动作。且会在 PTX 端动态显示目前的 PER。需设定参数如下：

```
/* set pipe active */
#define _DEFAULT_PIPE0          EN_P0       //EN_P0 / DIS_P0
#define _DEFAULT_PIPE1          DIS_P1      //EN_P1 / DIS_P1
#define _DEFAULT_PIPE2          DIS_P2      //EN_P2 / DIS_P2
#define _DEFAULT_PIPE3          DIS_P3      //EN_P3 / DIS_P3
#define _DEFAULT_PIPE4          DIS_P4      //EN_P4 / DIS_P4
#define _DEFAULT_PIPE5          DIS_P5      //EN_P5 / DIS_P5
/* set RX pipe auto ack */
#define _DEFAULT_AAP0           EN_AAP0     //EN_AAP0 / DIS_AAP0
#define _DEFAULT_AAP1           DIS_AAP1    //EN_AAP1 / DIS_AAP1
#define _DEFAULT_AAP2           DIS_AAP2    //EN_AAP2 / DIS_AAP2
#define _DEFAULT_AAP3           DIS_AAP3    //EN_AAP3 / DIS_AAP3
#define _DEFAULT_AAP4           DIS_AAP4    //EN_AAP4 / DIS_AAP4
#define _DEFAULT_AAP5           DIS_AAP5    //EN_AAP5 / DIS_AAP5
```

操作方式：按下 KEY2 即开始 PER Mode，直到下次按下 KEY2 结束 PER Mode。

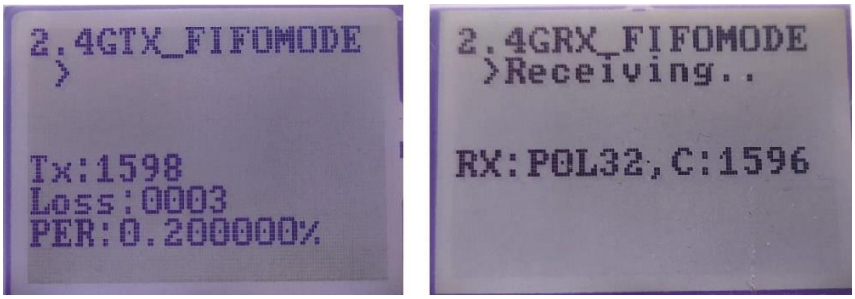


图 15、PER 显示 LCM 数据

SB Mode (单向传输)

由 PTX 端每次发出一笔 PN9 封包长度 32-byte 给 PRX 端。

操作方式：按下 KEY2 即开始 SB Mode，期间 PTX 端每单击 KEY1 即会发送一笔 PN9 封包长度 32Byte 给 RRX 端，直到再次按下 KEY2 结束 SB Mode。

ESB Mode (双向传输)

由 PTX 端每次发出 PN9 封包长度 32-byte 给 PRX 端，当 PTX 端收到 PRX 端发射的 ACK 封包即完成一次传输动作。

操作方式：按下 KEY2 即开始 ESB Mode，期间 PTX 端每单击 KEY1 即会发送一笔 PN9 封包长度 32Byte 给 RRX 端，直到再次按下 KEY2 结束 ESB Mode。

设定参数部分与 PER Mode 相同。

DPL

由 PTX 端每次可发射 PN9 封包动态长度 0~32-byte 给 PRX 端。

操作方式：按下 KEY2 即开始 DPL，期间 PTX 端每单击 KEY1 即会发送一笔 PN9 封包长度+1 Byte 给 RRX 端；每单击 KEY3 即会发送一笔 PN9 封包长度-1 Byte 给 RRX 端，直到再次按下 KEY2 结束 DPL。

需设定参数如下：

```
/* set FEATURE register */
#define _DEFAULT_DYN_ACK          DIS_DYN_ACK   //EN_DYN_ACK / DIS_DYN_ACK
#define _DEFAULT_ACK_PAY          DIS_ACK_PAY   //EN_ACK_PAY / DIS_ACK_PAY
#define _DEFAULT_DPL              EN_DPL        //EN_DPL / DIS_DPL
/* set pipe dynamic length */
#define _DEFAULT_DPL_P0           EN_DPL_P0     //EN_DPL_P0 / DIS_DPL_P0
#define _DEFAULT_DPL_P1           DIS_DPL_P1    //EN_DPL_P1 / DIS_DPL_P1
#define _DEFAULT_DPL_P2           DIS_DPL_P2    //EN_DPL_P2 / DIS_DPL_P2
#define _DEFAULT_DPL_P3           DIS_DPL_P3    //EN_DPL_P3 / DIS_DPL_P3
#define _DEFAULT_DPL_P4           DIS_DPL_P4    //EN_DPL_P4 / DIS_DPL_P4
#define _DEFAULT_DPL_P5           DIS_DPL_P5    //EN_DPL_P5 / DIS_DPL_P5
/* set pipe active */
#define _DEFAULT_PIPE0            EN_P0         //EN_P0 / DIS_P0
#define _DEFAULT_PIPE1            DIS_P1        //EN_P1 / DIS_P1
#define _DEFAULT_PIPE2            DIS_P2        //EN_P2 / DIS_P2
#define _DEFAULT_PIPE3            DIS_P3        //EN_P3 / DIS_P3
#define _DEFAULT_PIPE4            DIS_P4        //EN_P4 / DIS_P4
#define _DEFAULT_PIPE5            DIS_P5        //EN_P5 / DIS_P5
/* set RX pipe auto ack */
#define _DEFAULT_AAP0             EN_AAP0       //EN_AAP0 / DIS_AAP0
#define _DEFAULT_AAP1             DIS_AAP1      //EN_AAP1 / DIS_AAP1
#define _DEFAULT_AAP2             DIS_AAP2      //EN_AAP2 / DIS_AAP2
#define _DEFAULT_AAP3             DIS_AAP3      //EN_AAP3 / DIS_AAP3
#define _DEFAULT_AAP4             DIS_AAP4      //EN_AAP4 / DIS_AAP4
#define _DEFAULT_AAP5             DIS_AAP5      //EN_AAP5 / DIS_AAP5
```

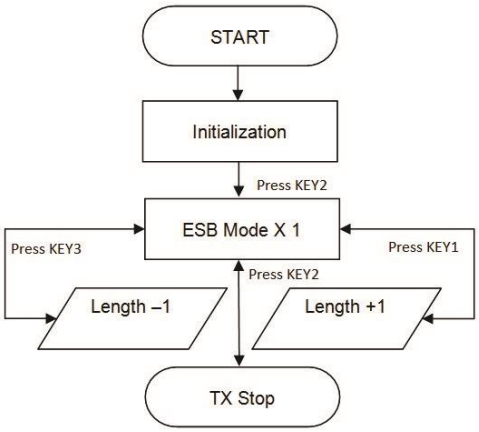


图 16、DPL 流程图

DYN_ACK

为了方便使用者随时切换 SB Mode (单向传输)与 ESB Mode (双向传输)，只要设定 EN_DYN_ACK 后即可使用两种 Strobe 命令：WRITE_NACKPLD_CMD (0x13)和 WRITE_ACKPLD_CMD (0x11)，轻易切换单向或是双向传输。

```
/* set FEATURE register */
#define _DEFAULT_DYN_ACK        EN_DYN_ACK      //EN_DYN_ACK / DIS_DYN_ACK
#define _DEFAULT_ACK_PAY        DIS_ACK_PAY     //EN_ACK_PAY / DIS_ACK_PAY
#define _DEFAULT_DPL            DIS_DPL         //EN_DPL / DIS_DPL
```

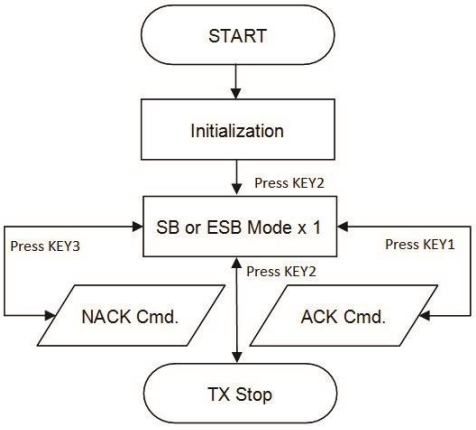


图 17、DYN_ACK 流程图

ACK PLD

当 PRX 端有信息想要回传给 PTX 端时，可以通过回传 ACK 时将数据带上。需要将 EN_ACK_PAY 和 EN_DPL 均设定才可以动作正常。

```
/* set FEATURE register */
#define _DEFAULT_DYN_ACK        DIS_DYN_ACK     //EN_DYN_ACK / DIS_DYN_ACK
#define _DEFAULT_ACK_PAY        EN_ACK_PAY      //EN_ACK_PAY / DIS_ACK_PAY
#define _DEFAULT_DPL            EN_DPL          //EN_DPL / DIS_DPL
```

请注意：1. 请在 PRX 端收到 RX_DR IRQ 前就使用 Strobe 命令：写 Pipe 0 ACK Payload 命令 (0x18) 将需要回传的数据填入；若没有及时填入，那么就只会回传 ACK。等待下次 PRX 端收到 RX_DR IRQ 时，数据才会与 ACK 一起传送至 PTX 端。

2. 当 PTX 端收到 PRX 端回传的数据后，PRX 端在下一笔收到 PTX 的封包时，除了 RX_DR IRQ 会使能外，额外还会多一个 TX DS(ACKPAY) IRQ 使能(表示 PTX 有收到 PRX 端的回传数据)。

BC5602 API 函数介绍

API 列表(51 项)与其功能，请参考范例程序的 BC5602.h 档案：

命令名称	功能说明
BC5602SoftwareReset()	软件复位
BC5602RegisterBank(regbk)	选择寄存器 Bank
BC5602DeepSleepMode()	深度睡眠模式设置
BC5602MiddleSleepMode()	中度睡眠模式设置
BC5602LightSleepMode()	轻度睡眠模式设置
BC5602StandbyMode()	待机模式设置
BC5602TransmitterMode()	TX 模式触发
BC5602ReceiveMode()	RX 模式触发
BC5602TxFIFOFlush()	TX FIFO Flush
BC5602RxFIFOFlush()	RX FIFO Flush
BC5602WriteTxPayload(pbuf,len)	将数据以带 Auto-ACK 的方式写入 TX FIFO
BC5602WriteTxNAckPayload(pbuf,len)	将数据以带 No-Auto-ACK 的方式写入 TX FIFO
BC5602WriteRxAckPayload(pipe,pbuf,len)	将 pbuf 的 len 个 Byte 的数据写入回传 Ack 的封包中
BC5602ReadRxPayload(pbuf,len)	从 RX Payload 中读取 len 个 Byte 长度的数据存入 pbuf
BC5602GetIRQLineStatus(void)	获取当前 IRQ 引脚的状态
BC5602RegisterConfigure(void)	将 BC5602 的特殊寄存器设定为默认值
BC5602CrystalSetup(u8 xo_il,u8 xo_trim)	设置晶振电流模式和内部电容校准值
BC5602WaitCrystalReady(void)	等待晶振稳定
BC5602SetPrimaryMode(u8 op_mode)	设置 BC5602 的工作模式
BC5602GetOperationMode(void)	读取 BC5602 当前的操作模式状态
BC5602SetCRCMode(u8 crc_mode)	设置 BC5602 CRC 格式
BC5602SetIRQMode(u8 int_source, u8 irq_state)	控制 BC5602 IRQ (TX_DS, RX_DR, MAX_RT)使能/除能
BC5602GetClearIRQFlags(void)	清除中断控制寄存器 1 标志位，并返回清除前的数值
BC5602SetAddressWidth(u8 aw)	设置 BC5602 地址长度
BC5602GetAddressWidth(void)	读取 BC5602 地址长度
BC5602SetAddress(const u8 adr,const u8 *adrbuf)	设置 PTX、PRX PIPE0、PIPE1、PIPE2、PIPE3、PIPE4、PIPE5 通道地址
BC5602GetAddress(const u8 adr,u8 *adrbuf)	读取 PTX、PRX PIPE0、PIPE1、PIPE2、PIPE3、PIPE4、PIPE5 通道地址
BC5602SetDataRate(u8 datarate)	设置 BC5602 数据传输速率
BC5602OpenPipe(u8 pipe_num, u8 auto_ack)	控制 pipe 开启、Auto-Ack 使能/除能

命令名称	功能说明
BC5602ClosePipe(u8 pipe_num)	控制 pipe 关闭
BC5602SetAutoRetr(u8 retr, u8 delay)	设置 BC5602 自动重发次数和重发延迟时间
BC5602SetRFChannel(channel)	设置 BC5602 通信频率
BC5602SetOutputPower(u8 power)	设置 BC5602 输出功率
BC5602GetIRQFlags(void)	读取 BC5602 中断控制寄存器 1 的 Bit4~Bit6 的数值
BC5602ClearIRQFlags(source)	清除 BC5602 中断控制寄存器 1 的标志位
BC5602SetRxPayloadWidth(pipe_num,width)	设置 BC5602 接收 pipe 的 Payload 长度
BC5602GetRxPayloadWidth(pipe_num)	读取 BC5602 接收 pipe 的 Payload 长度
BC5602GetAutoRetrStatus(void)	读取重发控制寄存器的数值
BC5602GetPacketLostCtr(void)	读取于同一频率丢包的次数
BC5602GetFIFOStatus(void)	读取 TX FIFO 和 RX FIFO 的状态
BC5602SetupFeature(setup)	选择 BC5602 Feature
BC5602SetupDynamicPayload(setup)	使能 pipe 的动态长度功能
BC5602ReadRxPayloadWidth(void)	回应当下 RX FIFO 的数据长度
BC5602SetupCE(setup)	使能/除能控制 BC5602
BC5602GetAddressRSSI(void)	读取地址匹配时的接收信号强度值
BC5602GetMeasurementRSSI(void)	读取当前接收信号强度值
BC5602StrobeCommand(u8 cmd)	发送 Strobe 命令
BC5602ReadRegister(u8 regs)	读取寄存器的数值
BC5602WriteRegister(u8 regs,u8 data)	对寄存器写入数值
BC5602ReadMultibyteRegister(u8 cmd,u8 *pbuf,u8 length)	读取寄存器连续的 n 个长度的数据
BC5602WriteMultibyteRegister(u8 cmd,u8 *pbuf,u8 length)	对寄存器连续写入 n 个长度的数据

表 1. BC5602 API 列表

以下针对模块的 API 进行详细说明：

API 名称	**BC5602SoftwareReset()**
API 作用	软件复位
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，会进行软件复位。
API 名称	**BC5602RegisterBank(regbk)**
API 作用	选择 BC5602 寄存器 Bank
输入参数	REGS_BANK0 / REGS_BANK1 / REGS_BANK2 / REGS_BANK3
输出参数	—
程序详解	范例程序中，输入参数可直接使用 REGS_BANK0 / REGS_BANK1 / REGS_BANK2 / REGS_BANK3 对此 API 操作。 1. REGS_BANK0 (0)，使用后，BC5602 的寄存器组被设定为 Bank 0。 2. REGS_BANK1 (1)，使用后，BC5602 的寄存器组被设定为 Bank 1。 3. REGS_BANK2 (2)，使用后，BC5602 的寄存器组被设定为 Bank 2。 4. REGS_BANK3 (3)，使用后，BC5602 的寄存器组被设定为 Bank 3。
API 名称	**BC5602DeepSleepMode()**
API 作用	控制 BC5602 进入深度休眠模式

输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入深度休眠模式。
API 名称	**BC5602MiddleSleepMode()**
API 作用	控制 BC5602 进入中度休眠模式
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入中度休眠模式。
API 名称	**BC5602LightSleepMode()**
API 作用	控制 BC5602 进入轻度休眠模式
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入轻度休眠模式。
API 名称	**BC5602StandbyMode()**
API 作用	控制 BC5602 进入待机模式
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入待机模式。
API 名称	**BC5602TransmitterMode()**
API 作用	控制 BC5602 进入发射模式
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入发射模式。
API 名称	**BC5602ReceiveMode()**
API 作用	控制 BC5602 进入接收模式
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，BC5602 会进入接收模式。
API 名称	**BC5602TxFIFOFlush()**
API 作用	清除 TX FIFO 的数据
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，会清除 BC5602 TX FIFO 的数据。
API 名称	**BC5602RxFIFOFlush()**
API 作用	清除 RX FIFO 的数据
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，会清除 BC5602 RX FIFO 的数据。
API 名称	**BC5602WriteTxPayload(pbuf,len)**
API 作用	将 pbuf 的 len 个 Byte 的数据以带 Auto-Ack 的方式写入 TX FIFO
输入参数 1	数据存储区

输入参数 2	数据长度(1~32)
输出参数	—
程序详解	注：输入参数 2 的最大长度不能超过 32 个 Byte。
API 名称	**BC5602WriteTxNAckPayload(pbuf,len)**
API 作用	将 pbuf 的 len 个 Byte 的数据以带 No-Auto-Ack 的方式写入 TX FIFO
输入参数 1	数据存储区
输入参数 2	数据长度(1~32)
输出参数	—
程序详解	注：输入参数 2 的最大长度不能超过 32 个 Byte。
API 名称	**BC5602WriteRxAckPayload(pipe,pbuf,len)**
API 作用	将 pbuf 的 len 个 Byte 的数据写入回传 Ack 的封包中
输入参数 1	pipe0 (0) / pipe1 (1) / pipe2 (2) / pipe3 (3) / pipe4 (4) / pipe5 (5)
输入参数 2	数据存储区
输入参数 3	数据长度(1~32)
输出参数	—
程序详解	注：输入参数 3 的最大长度 len 不能超过 32 个 Byte。
API 名称	**BC5602ReadRxPayload(pbuf,len)**
API 作用	从 RX Payload 中读取 len 个 Byte 长度的数据存入 pbuf
输入参数 1	数据存储区
输入参数 2	读取的数据长度(1~32)
输出参数	—
程序详解	注：输入参数 2 的最大长度 len 不能超过 32 个 Byte。
API 名称	**BC5602GetIRQLineStatus(void)**
API 作用	获取当前 IRQ 引脚的状态
输入参数	—
输出参数	TRUE / FALSE
程序详解	范例程序中，执行此 API 后，若 BC5602 曾经产生了中断(未复位)，则该函数会返回 TRUE，否则，则返回 FALSE。
API 名称	**BC5602RegisterConfigure(void)**
API 作用	将 BC5602 的特殊寄存器设定为默认值
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API，会对 BC5602 的特定寄存器进行配置，为了提升 RF 性能。
API 名称	**BC5602CrystalSetup(u8 xo_il,u8 xo_trim)**
API 作用	设置晶振电流模式和内部电容校准值
输入参数 1	TRUE / FALSE
输入参数 2	0~31
输出参数	—
程序详解	范例程序中，参数 1 可使用 TRUE / FALSE 对此 API 进行操作。 1. TRUE (1)，使用后，使能晶振低电流模式。 2. FALSE (0)，使用后，除能晶振低电流模式。 参数 2 可校准晶振内部电容负载值，具体数值请参考 Datasheet。

API 名称	**BC5602WaitCrystalReady(void)**
API 作用	等待晶振稳定
输入参数	—
输出参数	—
程序详解	范例程序中，执行此 API 后，会去判断 BC5602 的晶振是否稳定输出，若一直无法稳定，将导致持续执行此函数，且无法离开。
API 名称	**BC5602SetPrimaryMode(u8 op_mode)**
API 作用	设置 BC5602 的工作模式
输入参数	PRIM_PTX / PRIM_PRX
输出参数	—
程序详解	范例程序中，输入参数可直接使用 PRIM_PTX / PRIM_PRX 对此 API 操作。 1. 输入参数 PRIM_PTX (0)，使用后，BC5602 被设置为 PTX 设备。 2. 输入参数 PRIM_PRX (1)，使用后，BC5602 被设置为 PRX 设备。
API 名称	**BC5602GetOperationMode(void)**
API 作用	读取 BC5602 当前的模式状态
输入参数	—
输出参数	0H / 1H / 2H / 3H / 4H / 5H / 6H
程序详解	范例程序中，执行此 API 后，该 API 可返回 BC5602 当前的模式状态： 0H: 深度休眠模式 1H: 中度休眠模式 2H: 轻度休眠模式 3H: 待机模式 4H: 发射模式 5H: 接收模式 6H: 校准模式
API 名称	**BC5602SetCRCMode(u8 crc_mode)**
API 作用	设置 BC5602 CRC 格式
输入参数	CRC_OFF / CRC_8BIT / CRC_16BIT
输出参数	—
程序详解	范例程序中，输入参数可直接使用 CRC_OFF / CRC_8BIT / CRC_16BIT 对此 API 操作。 1. 输入参数 CRC_OFF (0)，使用后，BC5602 CRC 会被除能。 2. 输入参数 CRC_8BIT (1)，使用后，BC5602 CRC 格式被设置 CRC8=X^8+X^2+X+1。 3. 输入参数 CRC_8BIT (2)，使用后，BC5602 CRC 格式被设置 CRC16=$X^{16}+X^{12}+X^5+1$。
API 名称	**BC5602SetIRQMode(u8 int_source, u8 irq_state)**
API 作用	控制 BC5602 IRQ (TX_DS, RX_DR, MAX_RT)使能/除能
输入参数 1	IRQ_MAXRT / IRQ_TXDS / IRQ_RXDR / IRQ_ALL
输入参数 2	ENABLE / DISABLE
输出参数	—
程序详解	范例程序中，参数 1 可使用 IRQ_MAXRT / IRQ_TXDS / IRQ_RXDR / IRQ_ALL，参数 2 可使用 ENABLE / DISABLE 对此 API 操作。 1. 输入参数 1 的 IRQ_MAXRT (4)，如果参数 2 是 DISABLE，则 BC5602 检测达到最大重发次数将不会产生中断，否则立即产生 MAX_RT 中断。 2. 输入参数 1 的 IRQ_TXDS (5)，如果参数 2 是 DISABLE，则 BC5602 检测发送完成将不会产生中断，否则立即产生 TX_DS 中断。

	3. 输入参数 1 的 IRQ_RXDR (6)，如果参数 2 是 DISABLE，则 BC5602 检测接收到数据将不会产生中断，否则立即产生 RX_DR 中断。 4. 输入参数 1 的 IRQ_ALL (0xFF)，如果参数 2 是 DISABLE，则 BC5602 的 IRQ_MAXRT、TX_DS、RX_DR 的中断都会被除能，否则都会使能。
API 名称	**BC5602GetClearIRQFlags(void)**
API 作用	清除中断控制寄存器 1 标志位，并返回清除前的数值
输入参数	—
输出参数	中断控制寄存器 1 (04H)的值
程序详解	范例程序中，执行该 API 后，该 API 会清除 BC5602 中断控制寄存器 1 (04H)的 Bit 4 (MAX_RT)、Bit 5 (TX_DS)、Bit 6 (RX_DR)当前的状态，并返回清除前寄存器的值。
API 名称	**BC5602SetAddressWidth(u8 aw)**
API 作用	设置 BC5602 地址长度
输入参数	AW_3BYTE / AW_4BYTE / AW_5BYTE
输出参数	—
程序详解	范例程序中，参数可直接使用 AW_3BYTE / AW_4BYTE / AW_5BYTE 对此 API 操作。 1. 输入参数 AW_3BYTE (1)，使用后，BC5602 地址长度被设置为 3 个 Byte。 2. 输入参数 AW_4BYTE (2)，使用后，BC5602 地址长度被设置为 4 个 Byte。 3. 输入参数 AW_5BYTE (3)，使用后，BC5602 地址长度被设置为 5 个 Byte。
API 名称	**BC5602GetAddressWidth(void)**
API 作用	读取 BC5602 地址长度
输入参数	—
输出参数	1 / 2 / 3
程序详解	执行该 API 后，该 API 可返回 BC5602 当前设置的地址长度。 1. 输出参数返回 1，当前 BC5602 地址长度被设置为 3 个 Byte。 2. 输出参数返回 2，当前 BC5602 地址长度被设置为 4 个 Byte。 3. 输出参数返回 3，当前 BC5602 地址长度被设置为 5 个 Byte。
API 名称	**BC5602SetAddress(const u8 adr, const u8 *adrbuf)**
API 作用	设置 PTX、PRX PIPE0、PIPE1、PIPE2、PIPE3、PIPE4、PIPE5 通道的地址
输入参数 1	TRXADDR / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5
输入参数 2	输入地址数据
输出参数	—
程序详解	范例程序中，执行此 API 后，会将参数 2 传入的地址数据设置成参数 1 通道的通信地址。参数 1 可直接使用 TRXADDR / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5，对此 API 操作。 1. 输入参数 1 的 TRXADDR (0)，参数 2 是设置 PTX、PRX PIPE0 的地址。 2. 输入参数 1 的 PIPE1 (1)，参数 2 是设置接收 PIPE1 的地址。 3. 输入参数 1 的 PIPE2 (2)，参数 2 是设置接收 PIPE2 的地址。 4. 输入参数 1 的 PIPE3 (3)，参数 2 是设置接收 PIPE3 的地址。 5. 输入参数 1 的 PIPE4 (4)，参数 2 是设置接收 PIPE4 的地址。 6. 输入参数 1 的 PIPE5 (5)，参数 2 是设置接收 PIPE5 的地址。
API 名称	**BC5602GetAddress(const u8 adr,u8 *adrbuf)**
API 作用	读取 PTX、PRX PIPE0、PIPE1、PIPE2、PIPE3、PIPE4、PIPE5 通道地址
输入参数 1	TRXADDR / PIPE1 / PIPE2 /PIPE3 / PIPE4 / PIPE5
输入参数 2	存储地址的数据
输出参数	—

程序详解	范例程序中，执行此 API 后，可读取对应的通信通道(参数 1)至地址存储区(参数 2)，输入的参数 1 可直接使用 TRXADDR / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5，对此 API 操作。 1. 输入参数 1 的 TRXADDR (0)，参数 2 是存储 PTX、PRX PIPE0 接收地址。 2. 输入参数 1 的 PIPE1 (1)，参数 2 是存储接收 PIPE1 地址。 3. 输入参数 1 的 PIPE2 (2)，参数 2 是存储接收 PIPE2 地址。 4. 输入参数 1 的 PIPE3 (3)，参数 2 是存储接收 PIPE3 地址。 5. 输入参数 1 的 PIPE4 (4)，参数 2 是存储接收 PIPE4 地址。 6. 输入参数 1 的 PIPE5 (5)，参数 2 是存储接收 PIPE5 地址。
API 名称	**BC5602SetDataRate(u8 datarate)**
API 作用	设置 BC5602 数据传输速率
输入参数	DR500KBPS / DR250KBPS / DR125KBPS
输出参数	—
程序详解	范例程序中，输入参数可直接使用 DR500KBPS / DR250KBPS / DR125KBPS 对此 API 操作。 1. 输入参数的 DR500KBPS (0)，BC5602 传输速率被设置为 500Kbps。 2. 输入参数的 DR250KBPS (1)，BC5602 传输速率被设置 250Kbps。 3. 输入参数的 DR125KBPS (2)，BC5602 传输速率被设置为 125Kbps。
API 名称	**BC5602OpenPipe(u8 pipe_num, u8 auto_ack)**
API 作用	控制 pipe 开启、Auto-Ack 使能 / 除能
输入参数 1	PIPE0 / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5 / PIPEALL
输入参数 2	ENABLE / DISABLE
输出参数	—
程序详解	输入参数 1 可直接使用 PIPE0 / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5 / PIPEALL，输入参数 2 可直接使用 ENABLE / DISABLE 对此 API 操作。 1. 输入参数的 PIPE0 (0)，此时 PIPE0 通道将被开启，如果输入参数 2 是 DISABLE，则 PIPE0 通道的 Auto-ACK 功能则被除能，否则 Auto-ACK 功能会使能。 2. 输入参数的 PIPE1 (1)，此时 PIPE1 通道将被开启，如果输入参数 2 是 DISABLE，则 PIPE1 通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。 3. 输入参数的 PIPE2 (2)，此时 PIPE2 通道将被开启，如果输入参数 2 是 DISABLE，则 PIPE2 通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。 4. 输入参数的 PIPE3 (3)，此时 PIPE3 通道被开启，如果输入参数 2 是 DISABLE，则 PIPE3 通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。 5. 输入参数的 PIPE4 (4)，此时 PIPE4 通道被开启，如果输入参数 2 是 DISABLE，则 PIPE4 通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。 6. 输入参数的 PIPE5 (5)，此时 PIPE5 通道被开启，同时如果输入参数 2 是 DISABLE，则 PIPE5 通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。 7. 输入参数的 PIPEALL (0xFF)，此时所有通道被开启，如果输入参数 2 是 DISABLE，则所有通道的 Auto-ACK 功能被除能，否则 Auto-ACK 功能会使能。
API 名称	**BC5602ClosePipe(u8 pipe_num)**
API 作用	控制 pipe 关闭
输入参数	PIPE0 / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5 / PIPEALL
输出参数	—

程序详解	范例程序中，输入参数可直接使用 PIPE0 / PIPE1 / PIPE2 / PIPE3 / PIPE4 / PIPE5 / PIPEALL 对此 API 操作。 1. 输入参数 PIPE0 (0)，执行后，此时 PIPE0 通道被关闭。 2. 输入参数 PIPE1 (1)，执行后，此时 PIPE1 通道被关闭。 3. 输入参数 PIPE2 (2)，执行后，此时 PIPE2 通道被关闭。 4. 输入参数 PIPE3 (3)，执行后，此时 PIPE3 通道被关闭。 5. 输入参数 PIPE4 (4)，执行后，此时 PIPE4 通道被关闭。 6. 输入参数 PIPE5 (5)，执行后，此时 PIPE5 通道被关闭。 7. 输入参数 PIPEALL (0xFF)，执行后，此时所有通道被关闭。
API 名称	**BC5602SetAutoRetr(u8 retr, u8 delay)**
API 作用	设置 BC5602 自动重发次数和重发延迟时间
输入参数 1	0~15
输入参数 2	ARD_DLY250 / ARD_DLY500 / ARD_DLY750 / ARD_DLY1000 / ARD_DLY1250 / ARD_DLY1500 / ARD_DLY1750 / ARD_DLY2000 / ARD_DLY2250 / ARD_DLY2500 / ARD_DLY2750 / ARD_DLY3000 / ARD_DLY3250 / ARD_DLY3500 / ARD_DLY3750 / ARD_DLY4000
输出参数	—
程序详解	范例程序中，参数 1 可使用 0~15 对自动重发次数进行设置，参数 2 可使用 ARD_DLY250~ARD_DLY4000 对重发延迟时间 250μs~4000μs 进行选择，具体详细数据请查阅 Datasheet 中 PTX 重发控制寄存器 (13H)的介绍。
API 名称	**BC5602SetRFChannel(channel)**
API 作用	设置 BC5602 通信频率
输入参数	2~85
输出参数	—
程序详解	范例程序中，输入参数可使用 2~85 对通信频率进行设置，执行后，BC5602 的通信频率会被设置为(2400+channel)MHz。
API 名称	**BC5602SetOutputPower(u8 power)**
API 作用	设置 BC5602 输出功率
输入参数	N5dBm / P0dBm / P5dBm / P6dBm
输出参数	—
程序详解	范例程序中，输入参数可直接使用 N5dBm / P0dBm / P5dBm / P6dBm 对此 API 进行操作。 1. 输入参数 N5dBm (0)，执行后，BC5602 的输出功率被设置为-5dBm。 2. 输入参数 P0dBm (1)，执行后，BC5602 的输出功率被设置为 0dBm。 3. 输入参数 P5dBm (2)，执行后，BC5602 的输出功率被设置为 5dBm。 4. 输入参数 P6dBm (3)，执行后，BC5602 的输出功率被设置为 6dBm。
API 名称	**BC5602GetIRQFlags(void)**
API 作用	读取 BC5602 中断控制寄存器 1 的 Bit 4~Bit 6 的数值
输入参数	—
输出参数	10H / 20H / 40H
程序详解	范例程序中，执行此 API 后，能把中断控制寄存器 1 (04H) Bit 4~Bit 6 的状态读取出来。 1.若输出参数是 10H，说明产生 MAX_RT 中断。 2.若输出参数是 20H，说明产生 TX_DS 中断。 3.若输出参数是 40H，说明产生 RX_DR 中断，可从 RX FIFO 读取数据。
API 名称	**BC5602ClearIRQFlags(source)**
API 作用	清除 BC5602 中断控制寄存器 1 标志位
输入参数	_MASK_MAXRT_ / _MASK_TXDS_ / _MASK_RXDR_

输出参数	—
程序详解	范例程序中，执行此 API 后，输入对应的参数可清除对应的中断。 1. 输入参数的_MASK_MAXRT_ (10H)，会清除 MAX_RT 的中断。 2. 输入参数的_MASK_TXDS_ (20H)，会清除 TX_DS 的中断。 3. 输入参数的_MASK_RXDR_ (40H)，会清除 RX_DR 的中断。
API 名称	**BC5602SetRxPayloadWidth(pipe_num,width)**
API 作用	设置 BC5602 接收 pipe 的 Payload 长度
输入参数 1	RX_PIPE0 / RX_PIPE1 / RX_PIPE2 / RX_PIPE3 / RX_PIPE4 / RX_PIPE5
输入参数 2	0~32
输出参数	—
程序详解	注：当对应的 pipe 动态长度功能开启时，此设定长度将不会被使用。
API 名称	**BC5602GetRxPayloadWidth(pipe_num)**
API 作用	读取 BC5602 接收 pipe 的 Payload 长度
输入参数	RX_PIPE0 / RX_PIPE1 / RX_PIPE2 / RX_PIPE3 / RX_PIPE4 / RX_PIPE5
输出参数	数值
程序详解	范例程序中，执行此 API 后，能返回一个数值，此数值是输入通信通道设置的 Payload 长度。
API 名称	**BC5602GetAutoRetrStatus(void)**
API 作用	读取重发控制寄存器的数值
输入参数	—
输出参数	数值
程序详解	范例程序中，执行此 API 后，能读取出重发控制寄存器的值，High nibble 代表在同一频率丢失的封包次数，Low nibble 代表重发的封包次数。
API 名称	**BC5602GetPacketLostCtr(void)**
API 作用	读取于同一频率丢包的次数
输入参数	—
输出参数	丢失的封包次数
程序详解	范例程序中，执行此 API 后，能从重发控制寄存器读取出在同一频率丢失的封包次数。
API 名称	**BC5602GetFIFOStatus(void)**
API 作用	读取 TX FIFO 和 RX FIFO 的状态
输入参数	—
输出参数	数值
程序详解	范例程序中，执行此 API 后，能读取出 FIFO 寄存器的值，此寄存器只有 Bit 0、Bit 1、Bit 4、Bit 5 有效，这个四个 bit 具体的的含义如下： 1. Bit 0：若为 1，RX FIFO 为空，否则 RX FIFO 有数据。 2. Bit 1：若为 1，RX FIFO 已满，否则 RX FIFO 未满。 3. Bit 4：若为 1，TX FIFO 为空，否则 TX FIFO 有数据。 4. Bit 5：若为 1，TX FIFO 已满，否则 TX FIFO 未满。
API 名称	**BC5602SetupFeature(setup)**
API 作用	选择 BC5602 Feature
输入参数	数值
输出参数	—
程序详解	范例程序中，执行此 API，输入参数只需要设置 Bit 0、Bit 1、Bit 2、Bit 7，每个 Bit 的作用如下：

	1. Bit 0：使能 PTX“以带 No-Auto-ACK 的方式写数据到 TX FIFO”的命令 2. Bit 1：使能回传 ACK 中带 Payload 数据的功能 3. Bit 2：使能动态 Payload 长度功能 4. Bit 7：NO_ACK 位功能控制
API 名称	**BC5602SetupDynamicPayload(setup)**
API 作用	使能 pipe 的动态长度功能
输入参数	0x01 / 0x02 / 0x04 / 0x08 / 0x10 / 0x20 对应 pipe0 / pipe1 / pipe2 / pipe3 / pipe4 / pipe5
输出参数	—
程序详解	范例程序中，输入参数为一个 6-bit 的数据，Bit0~Bit5 代表 pipe0~pipe5，若设置为 1，则开启该 pipe 的动态长度功能，设置为 0，则除能该 pipe 的动态长度功能。
API 名称	**BC5602ReadRxPayloadWidth(void)**
API 作用	回应当下 RX FIFO 的数据长度
输入参数	—
输出参数	数据长度
程序详解	范例程序中，执行此 API 后，此 API 会输出当下 RX FIFO 的数据长度。
API 名称	**BC5602SetupCE(setup)**
API 作用	使能/除能控制 BC5602
输入参数	CE / ~CE
输出参数	—
程序详解	范例程序中，输入参数可使用 CE / ~CE 对此 API 操作。 1. CE (1)，若输入参数是 CE，则使能控制 BC5602。 2. ~CE (0)，若输入参数是~CE，则除能控制 BC5602。
API 名称	**BC5602GetAddressRSSI(void)**
API 作用	读取地址匹配时的接收信号强度值
输入参数	—
输出参数	信号强度值
程序详解	范例程序中，执行此 API 后，会返回一个数值，此数值为 BC5602 读取到地址匹配时瞬间的接收信号强度。
API 名称	**BC5602GetMeasurementRSSI(void)**
API 作用	读取当前接收信号强度值
输入参数	—
输出参数	信号强度值
程序详解	范例程序中，执行此 API 后，会返回一个数值，此数值为 BC5602 读取到数据时的接收信号强度。
API 名称	**BC5602StrobeCommand(u8 cmd)**
API 作用	发送 Strobe 命令
输入参数	Strobe 命令
输出参数 1	—
程序详解	具体详细的 Strobe 命令请查阅 Datasheet 中 Strobe 命令表的介绍。
API 名称	**BC5602ReadRegister(u8 regs)**
API 作用	读取寄存器的数值
输入参数	寄存器的地址
输出参数	返回读取到寄存器的值

程序详解	输入参数是寄存器的地址，具体的寄存器地址请查阅 Datasheet 寄存器的介绍。执行此 API 后，可通过此函数读取输入寄存器的数值。
API 名称	**BC5602WriteRegister(u8 regs,u8 data)**
API 作用	对寄存器写入数值
输入参数	寄存器的地址
输入参数	需要写入的数值
输出参数	—
程序详解	输入参数是寄存器的地址，具体的寄存器地址请查阅 Datasheet 寄存器的介绍。
API 名称	**BC5602ReadMultibyteRegister(u8 cmd,u8 *pbuf,u16 length)**
API 作用	读取寄存器的 n 个长度的数据
输入参数 1	寄存器地址
输入参数 2	存储读取出的数据
输入参数 3	需要读取的长度
输出参数	—
程序详解	输入参数 1 是寄存器的地址，具体的寄存器地址请查阅 Datasheet 寄存器的介绍。
API 名称	**BC5602WriteMultibyteRegister(u8 cmd,u8 *pbuf,u16 length)**
API 作用	对寄存器写入 n 个长度的数据
输入参数 1	寄存器地址
输入参数 2	需写入的数据
输入参数 3	需要写入的长度
输出参数	—
程序详解	输入参数 1 是寄存器的地址，具体的寄存器地址请查阅 Datasheet 寄存器的介绍。

结论

本文介绍了如何使用 BC5602 相关的七大项应用，后续使用者于程序编辑上亦可以通过芯通取得简易的范例程序。

参考资料

参考文件 BC5602 Datasheet。

如需进一步了解，敬请浏览 HOLTEK 官方网站 http://www.holtek.com.cn。

版本及修改说明

日期	作者	Issue 发行、修订说明
2020.04.23	徐鸿文(Harry Hsu)	First Version