BK1080 编程手册

简介：

BK1080 系列芯片提供了多种封装形式以及应用模式以满足各种场合不同应用。本文档总结了 BK1080 系列芯片的编程方法。本文档配合 Datasheet，Layout Guideline，FAQ 使用能使用户更好的使用 BK1080 系列。

1 目录

2 控制接口总线

BK1080 可以通过 2 线 I2C 模式或者 3 线 SPI 模式由一个 MCU 来控制，最快可支持高达 20MHz 的时钟信号。用户可以通过设置 MODE 脚（PIN 7）的高低电平来选择其中一种通信模式。当 MODE = 0 时，为 I2C 模式，当 MODE = 1 时，为 SPI 模式。（BK1080 XB、BK1080SB 这两种封装不支持 SPI 模式，只可用 I2C 模式）

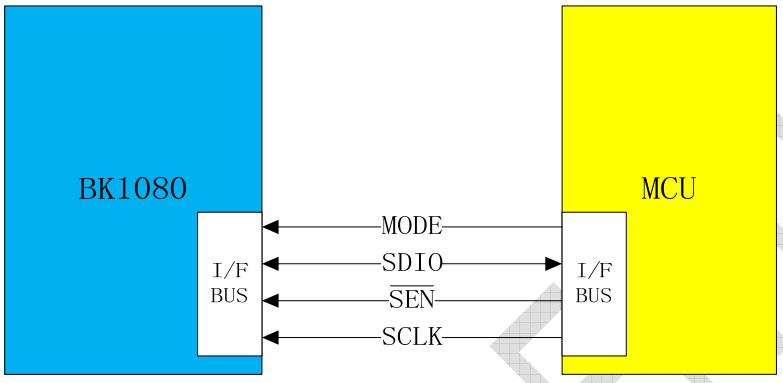

图 1 接口总线

不论是 I2C 模式还是 SPI 模式，BK1080 总是在 SCLK 信号的上升沿获取 SDIO 线上的数据，而在 SCLK 信号的下降沿输出数据到 SDIO 线上。所以对于 MCU 来说，必须在 SCLK 的下降沿向 SDIO 输出数据，在 SCLK 的上升沿后读取 SDIO 上的数据。

2.1 I2C 模式

在 I2C 模式下 SCLK 为时钟信号，SDIO 为数据信号，SEN 无效，此管脚可以接高电平或者悬空。

一个 I2C 读写操作由 Start 条件开始，到 Stop 条件结束。在 Start 之后，MCU 需要向 SDIO 上输出一个 8 位的 Device ID，BK1080 的 Device ID 为 0x80。

输出 Device ID 之后，MCU 继续向 SDIO 上输出一个 8 位的控制字，控制字由 7 位起始寄存器地址，以及一个读写位组成（读操作为 1，写操作为 0）。例如：起始寄存器地址为 0x03，需要从 Device 中读取数据，则 ControlWord=(0x03<<1+1)=0x07。

输出控制字之后，可以向 SDIO 数据需要写入的数据（写操作）或者从 SDIO 上获取数据（读操作）。

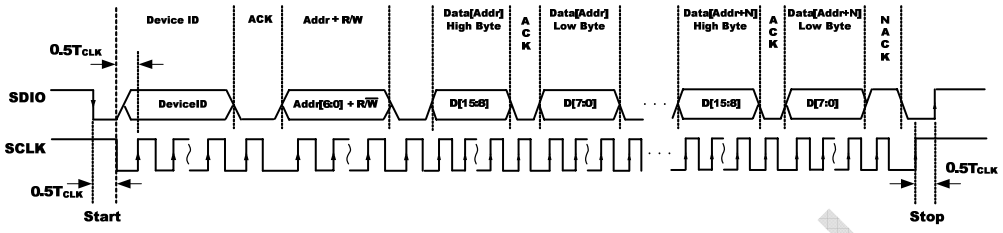

图 2 I2C 接口时序图

在 MCU 写数据时，每次写入一个字节之后，Device 会输出一个低电平 ACK 信号。在 MCU 读数据时，在每次读出一个字节后，MCU 必须输出一个 ACK 信号给 BK1080；而输出最后一个数据后，MCU 需要给出 NACK 信号给 BK1080。

2.2 3 线 SPI 模式

在 3 线 SPI 模式下 SCLK 为时钟信号，SDIO 为数据信号，SEN 为使能信号。

一个 SPI 读写操作从 SEN 信号的下降沿开始，到 SEN 信号的上升沿结束。SEN 拉低之后 MCU 需要向 SDIO 输出一个 8 位命令字。命令字由 7 位的起始寄存器地址和一个读写位组成，其计算方式同 I2C 的控制字。

输出控制字之后，可以向 SDIO 数据需要写入的数据（写操作）或者从 SDIO 上获取数据（读操作）。

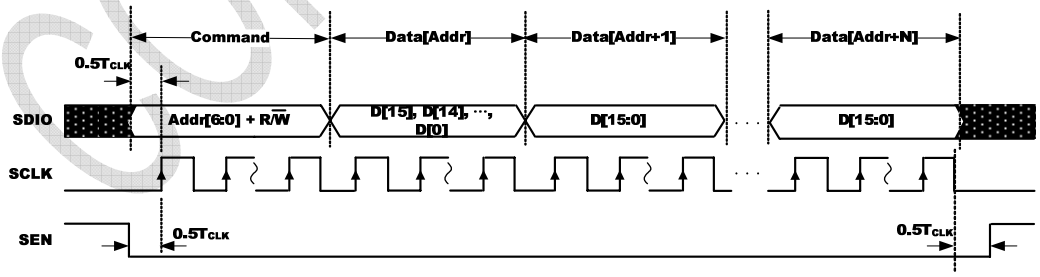

图 3 3 线接口时序图

3 状态控制

BK1080 有 5 种状态，分别为：复位初始化状态(Reset & Initial)，设置工作频点(Tune)，搜索电台(Seek)，工作状态(Working)和省电状态(Power down)。

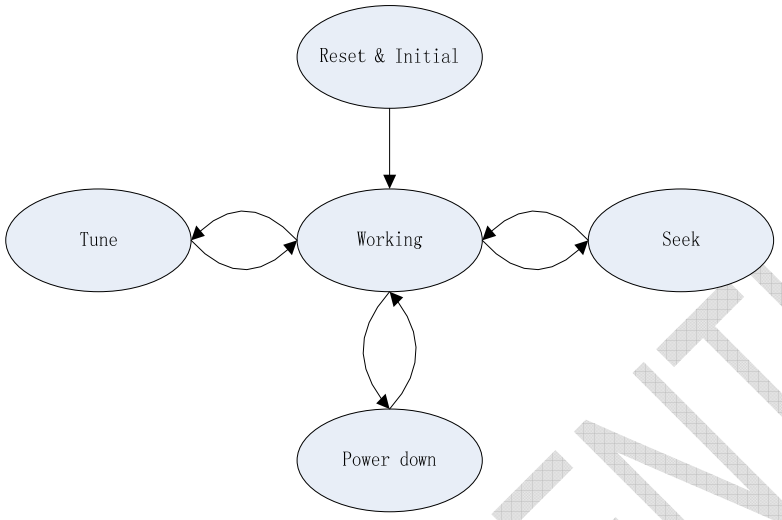

图 4 状态控制转换图

3.1 复位初始化

上电时，需要按如下顺序对 BK1080 进行操作：

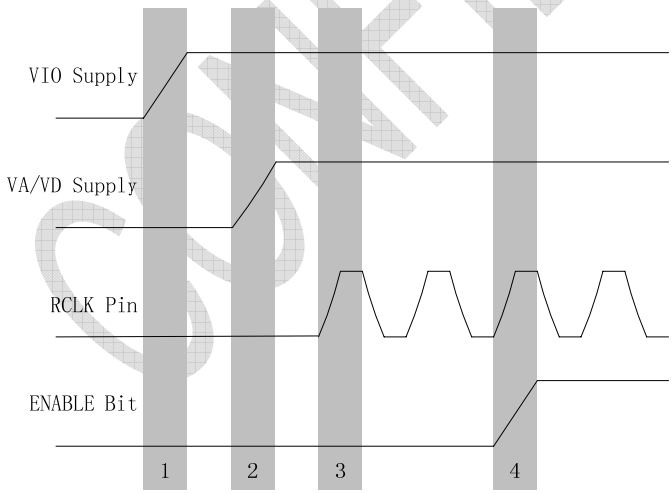

图 5 初始化顺序

提供 VIO 电压；

提供 VA 和 VD 电压（VA 和 VD 可以与 VIO 同时上电）；

提供 RCLK 时钟；

设置 ENABLE 为 1，DISABLE 为 0；

设置 BK1080 寄存器。

程序示例：

```
Supply VIO
Supply VD & VA
Supply RCLK clock
Mov 0x0201, 02h       //set ENALBE=1，DISABLE=0
//Set other registers
Mov 0x????, ??h
Mov 0x????, ??h
Mov 0x????, ??h
……
```

3.2 设置工作频点

设置工作频点前首先需要配置如下参数：

电台频率范围 BAND<1:0> (REG5 [7:6])；

00 = 87.5-108MHz

01 = 76-108MHz

10 = 76-90MHz

搜台步长 SPACE<1:0> (REG5 [5:4])；

00 = 200kHz

01 = 100kHz

10 = 50kHz

电台频道 CHAN<9:0> (REG3 [9:0])。

电台频道 CHAN[9:0]和电台频率 Freq 的换算公式如下：

CHAN = (Freq – $Freq_0$) / SPACE　　($Freq_0$ 为所选择 BAND 的起始频率)

假设设置 BAND = 00，即选择 87.5-108MHz，设置 SPACE = 01，即选择步长为 100kHz。如果需要设置频率为 101.7MHz 的话，那么设置 channel 值为 CHAN = (101.7-87.5)MHZ/100kHz = 0x8E。

设置完这些参数后，通过写寄存器将 TUNE(REG3[15])设置为 1，BK1080 将开始 Tune。当 Tune 结束后 STC(REG10[14])为将被置 1，且在 STC 中断被使能的状态下（STCIEN (REG4[14]) = 1，GPIO2<1:0> (REG4[3:2]) = 0x01），在 GPIO2 管脚（PIN 20）上会产生一个 5ms 的低电平脉冲。

当 TUNE 操作完成之后由软件读出 REG10 和 REG11 的值，从而得到 ST、RSSI 和 READCHAN 等参数值。

在下一次 Tune 操作开始前，需要把 TUNE(REG3[15])清 0，然后再次将其置 1，BK1080 才会开始另一次 Tune 操作。

程序示例：

```
Mov 0x0A1F, 05h      //set BAND = 00, set SPACE = 01
Mov 0x008E, 03h      //set CHAN = 0x8E, 设置频率为 101.7MHz
Mov 0x808E, 03h      //set Tune = 1
Wait 50ms
Read 0Ah, 0Bh//读取 RSSI 等参数
Stop Tune
```

3.3 搜索电台

BK1080 提供硬件搜台和软件搜台两种不同的搜台方式：硬件搜台是由 BK1080 的 Seek 功能来搜索，直到搜索到有效电台或者在整个频段搜索失败而退出；软件搜台是运用了 BK1080 的 Tune 功能，上层软件需要在每个频点上都 Tune 一次，直至找到有效电台。

硬件搜台的优点在于占用 MCU 资源更少，而软件搜台的优点是更加灵活，比如用于需要显示搜台进度的话可以使用这种搜台方式。

硬件搜台：

用户开始硬件搜台之前首先需要设置如下参数：

电台频率范围 BAND<1:0> (REG5 [7:6])

搜台步长 SPACE<1:0> (REG5 [5:4])

搜台模式（在频带边界循环搜索或停止搜索）SKMODE(REG2[10])

0：搜索到边界处循环，继续搜索

1：搜索到边界处停止搜索

搜索方向（向上搜索或向下搜索）SEEKUP(REG2[9])。

0：向下搜索

1：向上搜索

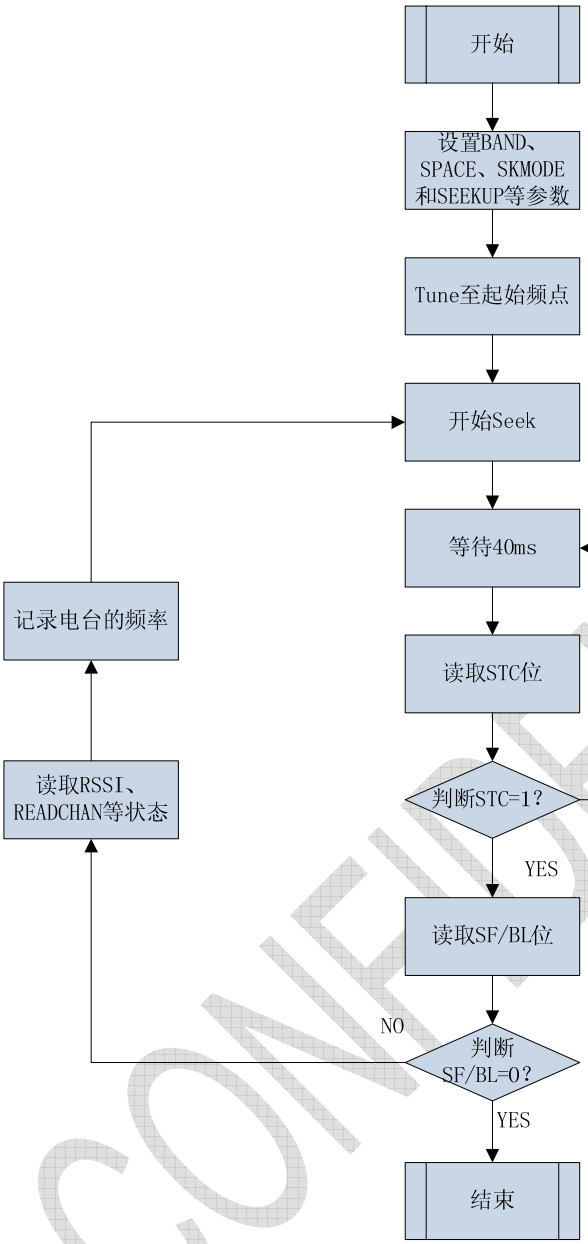

图 6 硬件搜台流程图

设置完这些参数后把 SEEK(REG2[8])置 1，BK1080 将开始 Seek。和 Tune 操作类似，当 BK1080 完成 Seek 操作之后，STC(REG10[14])会置 1，同时在 STC 中断使能的状态下 GPIO2 管脚会产生 5ms 的低电平脉冲。读出 SF/BL(REG10[13])的值，可以判断是否 Seek 成功。

Seek 操作完成后，用户可以读出 REG10 和 REG11 的值，从而得到 RSSI，ST 和 READCHAN 等信息。

程序示例：

Step 1

Mov 0x0A1F, 05h　　//set BAND = 00, set SPACE = 01

Mov 0x0601, 02h　　//set SKMODE = 1, SEEKUP = 1

Mov 0x8000, 03h　　//set CHAN = 0x00, 设置频率为 87.5MHz

Wait 50ms

Mov 0x0000, 03h　　//clear Tune bit

Step 2

Mov 0x0701, 02h　　//set SEEK = 1, 开始 seek

Wait STC=1 (polling mode) or GPIO2=0 (interrupt mode)　　//wait for seek complete

Read 0Ah, 0Bh//读取 RSSI 和 READCHAN 等参数

If SF/BL = 1, go to Step 3; else memorize READCHAN and go to Step 2.

Step 3

Stop seek

软件搜台：

由于软件搜台运用了 Tune 功能，所以其设置和 Tune 相同。每次 Tune 完成后需要读出寄存器的值来判断是否为真台。然后 Tune 下一个频点，循环进行，直至搜索完整个频段。

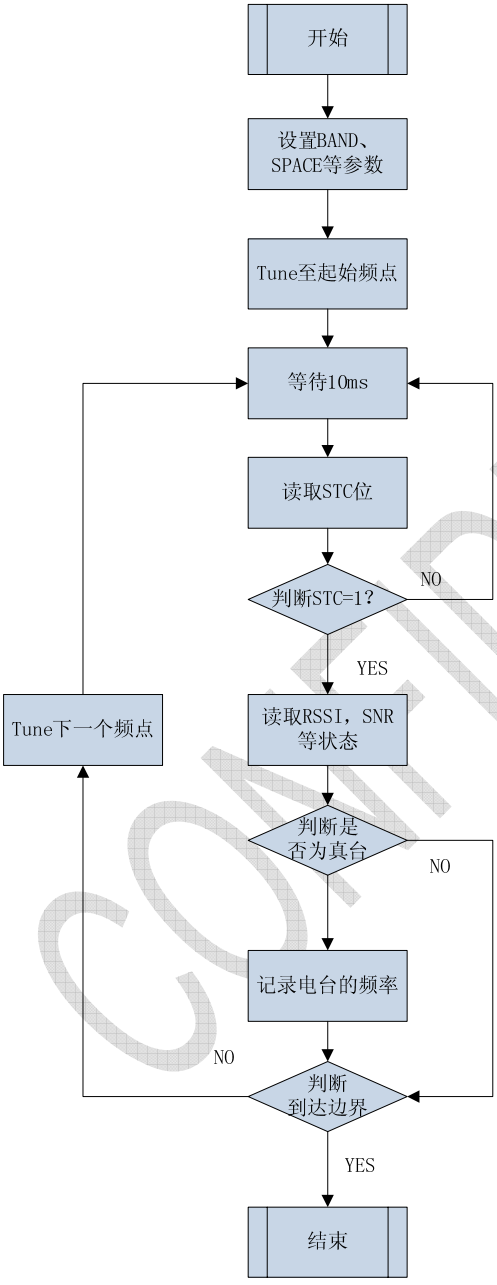

图 7 硬件搜台流程图

程序示例：

Step 1

Mov 0x0A1F, 05h　　//set BAND = 00, set SPACE = 01

CHAN = 0,　　//设置为起始频率

Step 2

Value = CHAN | 0x8000

Mov Value, 03h　　//set CHAN = 0x00, 设置频率为 87.5MHz

Wait 50ms

Mov 0x0000, 03h　　//clear Tune bit

Read 07h,0Ah,0Bh　　//读取 RSSI, SNR 和 Frequency Deviation 等参数，并判断是否为真台

If this is a real station, memorize READCHAN

CHAN = CHAN +1

If frequency is larger than band limit, go to Step 3

Value = CHAN | 0x8000

Go to Step 2

Step 3

Stop seek

3.4 省电状态

在不需要收听电台时，可以通过控制把 DISABLE(REG2[6])和 ENABLE(REG2[0])写为 1，使 BK1080 进入 power down 状态。此时 BK1080 只消耗极小的电流(Typical: 1.9uA)，但读写寄存器的操作依然可以进行。

当需要 BK1080 从省电状态恢复到工作状态时，可以把 DISABLE(REG2[6])写为 0，ENABLE(REG2[0])写为 1。然后重新初始化所有寄存器，BK1080 即可恢复工作。

程序示例：

Enter Power Down mode

Mov 0x0241, 02h　　//set DISABLE = 1

Exit from Power Down mode

　　Mov 0x0201, 02h　　//set DISABLE = 0, set ENABLE =1

//Set other registers

Mov 0x????, ??h

Mov 0x????, ??h

Mov 0x????, ??h

……

4 附录

4.1 移植应用

由于各家 FM 接受芯片的 ChipID 都不相同，所以可以根据所读到的 ChipID 来做到一套软件兼容多家 FM 接收芯片，具体方法如下：

a) 在代码中添加操作 BK1080 的函数，并在 C 文件开头添加函数声明

```
00060: bool BK1080_Intialization(void);
00061: void BK1080_SetVolumeLevel(uint8 level);
00062: uint16 BK1080_FreqToChan(uint16 frequency);
00063: uint8 BK1080_GetSigLvl( int16 curf );
00064: bool BK1080_IsChipValid(void);
00065: void BK1080_Mute(uint8 mute);
00066: void BK1080_SetFreq(int16 curFreq);
00067: uint8 BK1080_ValidStop(int16 freq);
00068: void BK1080_PowerOnReset(void);
00069: void BK1080_PowerOffProc(void);
00070: #ifdef BK1080_AUTO_SCAN_BY_CHIP
00071: uint8 BK1080_FmAutoScan (kal_uint16 *Freqs_buf,uint8 max_len,uint8 b_sort_by_RSSI);
00072: #endif
00073: #define FMCHANNELSMAX 30
```

b) 添加 BK1080 的初始化寄存器配置数组 kal_uint16 BK1080_Digital_Reg[]

```
00574: kal_uint16 BK1080_Digital_Reg[]=
00575: {
00576: 0x0008, //REG0
00577: 0x1080, //REG1
00578: 0x0201, //REG2
00579: 0x0000, //REG3
00580: 0x40C0, //REG4
00581: 0x0A1F, //REG5
00582: 0x002E, //REG6
00583: 0x02FF, //REG7
00584: 0x5B11, //REG8
00585: 0x0000, //REG9
00586: 0x411E, //REG10
00587: 0x0000, //REG11
00588: 0xCE00, //REG12
00589: 0x0000, //REG13
00590: 0x0000, //REG14
00591: 0x1000, //REG15
00592: 0x0010, //REG16
00593: 0x0000, //REG17
00594: 0x13FF, //REG18
00595: 0x9852, //REG19
00596: 0x0000, //REG20
00597: 0x0000, //REG21
00598: 0x0008, //REG22
00599: 0x0000, //REG23
00600: 0x51E1, //REG24
00601: 0x28DC, //REG25
00602: 0x2645, //REG26
00603: 0x00E4, //REG27
00604: 0x1CD8, //REG28
00605: 0x3A50, //REG29
00606: 0xEAF0, //REG30
00607: 0x3000, //REG31
00608: 0x0000, //REG32
00609: 0x0000, //REG33
00610: };
```

c) 在上层函数中添加调用 BK1080 的函数分支，根据开机初始化时获得的 ChipID 来判断，如果是 BK1080 的话，就运行 BK1080 的函数，是其它家 FM 接收芯片的话，就运行相应的函数。

5 更新记录

版本	更新内容	日期	作者
Rev.0.1	Initial draft	09-12-2009	JW，LFB