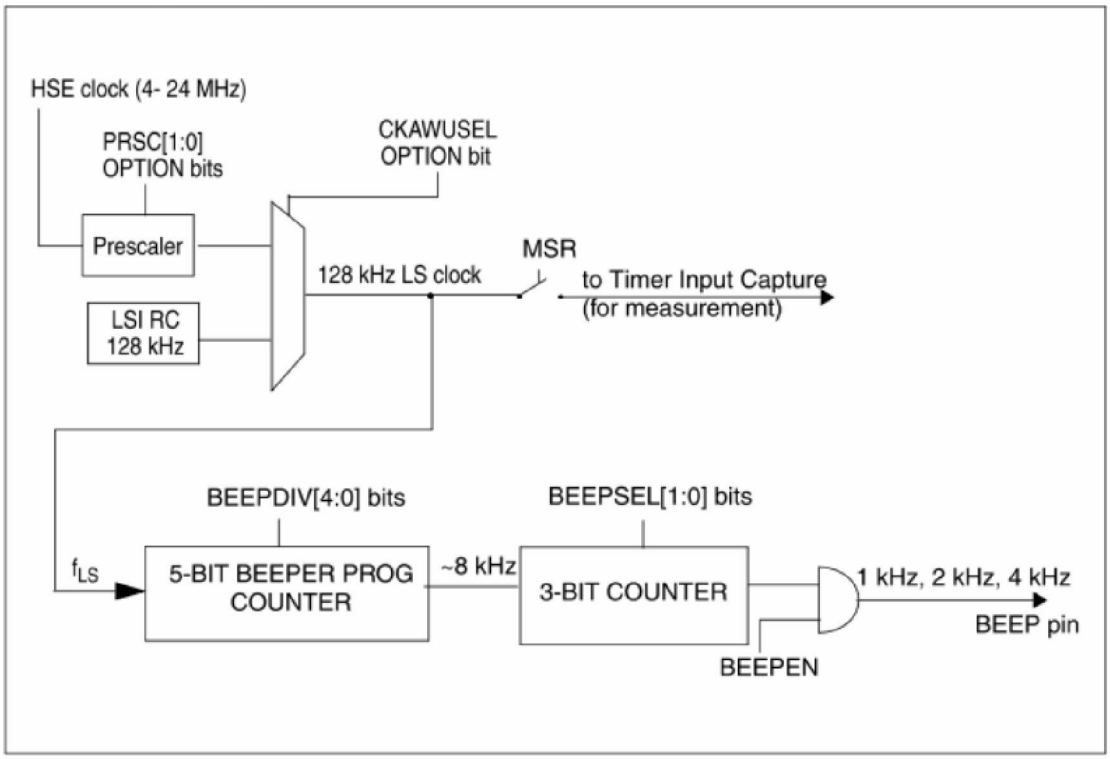
例程五 蜂鸣器

蜂鸣器接口是 STM8S 特有的一个模块，它产生一个特定的频率的方波来驱动无源蜂鸣器。

下面介绍一下 Beep 的功能模块

当LS时钟工作在128kHz时可产生频率为1kHz，2 kHz或者是4 kHz的蜂鸣信号。

图23 蜂鸣器功能图



13.2 功能描述

13.2.1 蜂鸣器操作

为了使用蜂鸣功能，按顺序执行如下的步骤：

1. 根据 *13.2.2*中描述的方法确定BEEPDIV[4:0]的值来校准LS时钟的频率；
2. 通过写BEEP_CSR的 BEEPSEL[1:0] 位来选择1 kHz, 2 kHz 或 4 kHz 的输出频率；
3. 置位BEEP_CSR的 BEEPEN 位来使能LS的时钟源；

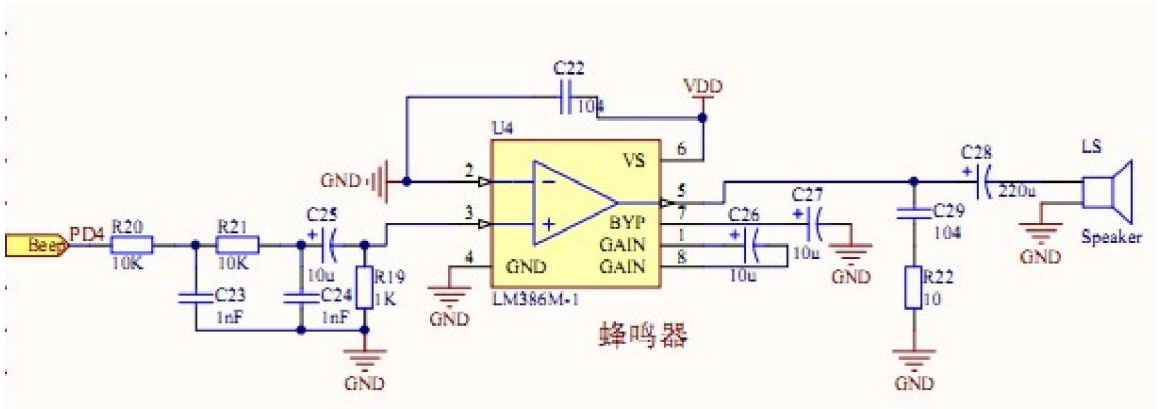
注意： 预分频计算器仅仅在当BEEPDIV[4:0]的值不同于复位值0x1F时才开始运行。

13.2.2 蜂鸣器校准

该步骤可以用来校准LS 128 kHz 的时钟以便达到标准的1 kHz, 2 kHz 或 4 kHz 频率输出采用如下的步骤：

1. 测量LSI的时钟频率(请参考*12.2.3*)
2. 采用如下方法计算$BEEP_{DIV}$的值，这里 A 和 x 是 $f_{LS}/8$ (kHz) 的整数和小数部分值：
   当 x 小于或者等于 A/(1+2*A)时，$BEEP_{DIV}$ = A-2 ；
   否则$BEEP_{DIV}$ = A-1
3. 将$BEEP_{DIV}$值写入到BEEP_CSR的BEEPDIV[4:0] 位。

以上的功能描述来自 STM8 寄存器.pdf 中的第 96 页
看完理论设置之后就一下我们的风驰电子 STM8 开发板的硬件设计。我们是用 LM386 音频放大器来驱动蜂鸣器的，这样处理的音质会好很多，这里先设置 STM8S207RB 里面的 Beepd 的功能模块，稍后几个例程将会介绍如何设计一个音乐符声音。
看下我们的硬件设计



现从我们的主函数看起

```
int main(void)
{
  /* Infinite loop */

  /*设置内部时钟16M为主时钟*/

  CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
   /*!<Set High speed internal clock  */
  Set_Beep_OptionByte();
  Beep_Init(BEEP_FREQUENCY_2KHZ);
  BEEP_LSICalibrationConfig(LSI_128kHz);

  while (1)
  {
    /*  添加你的代码  */


  }
}
```

Set_Beep_OptionByte()这个函数非常重要，是设置 Beep 的功能，在默认情况下，是没有 Beep 这个功能的，Beep 是备选功能重映射的，只有激活它才有 Beep 的功能。看下它的函数原型：

```
void Set_Beep_OptionByte(void)
{
  uint16_t Beep_Option_status;/*记录激活备选功能Beep的状态*/
  Beep_Option_status=FLASH_ReadOptionByte(Beep_OptionAdd);
  /*Beep_Option_status的最高位为1激活了Beep,否则不激活*/
  if(!(Beep_Option_status&0x8000))
  {
    FLASH_ProgramOptionByte(Beep_OptionAdd, (uint8_t)(Beep_Option_status|0x8000));
    /*向Beep_OptionAdd  0x4803 置1, 激活了Beep*/
  }
}
```

这里就是备选功能重映射设置，在这里粗略谈一下，后面的例程会有专门的详细的讲解。大家也可以参考“STM8SXX 中文手册.pdf”中的第 26、27 页。

表6　选项字节

Addr.	Option name	Option byte no.	Option bits								Factory default setting
			7	6	5	4	3	2	1	0	
4800h	Read-out protection (ROP)	OPT0	ROP[7:0]								00h
4801h	User boot code(UBC)	OPT1	UBC[7:0]								00h
4802h		NOPT1	NUBC[7:0]								FFh
4803h	Alternate function remapping (AFR)	OPT2	AFR7	AFR6	AFR5	AFR4	AFR3	AFR2	AFR1	AFR0	00h
4804h		NOPT2	NAFR7	NAFR6	NAFR5	NAFR4	NAFR3	NAFR2	NAFR1	NAFR0	FFh

AFR7 被选功能重映射选项7

0：端口D4备选功能为TIM2_CH1

1：端口D4备选功能为BEEP

Beep_Init(BEEP_FREQUENCY_2KHZ)是初始化 Beep，并设置产生 2kHz 的方波驱动蜂鸣器。

函数原型：

```
void Beep_Init(BEEP_Frequency_TypeDef BEEP_Frequency)
{
  BEEP_Init(BEEP_Frequency);
  CLK_LSICmd(ENABLE);
  BEEP_Cmd(ENABLE);
}
```

BEEP_Init(BEEP_Frequency)设置 Beep 管脚的输出频率

函数原型：

```
/**
  * @brief  Initializes the BEEP function according to the specified parameters
  * @param   BEEP_Frequency Frequency selection.
  * can be one of the values of @ref BEEP_Frequency_TypeDef.
  * @retval None
  * @par Required preconditions:
  * The LS RC calibration must be performed before calling this function.
  */
void BEEP_Init(BEEP_Frequency_TypeDef BEEP_Frequency)
{

    /* Check parameter */
    assert_param(IS_BEEP_FREQUENCY_OK(BEEP_Frequency));

    /* Set a default calibration value if no calibration is done */
    if ((BEEP->CSR & BEEP_CSR_BEEPDIV) == BEEP_CSR_BEEPDIV)
    {
        BEEP->CSR &= (uint8_t)(~BEEP_CSR_BEEPDIV); /* Clear bits */
        BEEP->CSR |= BEEP_CALIBRATION_DEFAULT;
    }

    /* Select the output frequency */
    BEEP->CSR &= (uint8_t)(~BEEP_CSR_BEEPSEL);
    BEEP->CSR |= (uint8_t)(BEEP_Frequency);
}
```

CLK_LSICmd(ENABLE) 把 LSI 时钟打开，复位后默认是打开，这里设置为了更好说明 Beep 的工作原理

函数原型：

```
/**
  * @brief  Enables or disables the Internal Low Speed oscillator (LSI).
  * @param    NewState new state of LSIEN, value accepted ENABLE, DISABLE
  * @retval None
  */
void CLK_LSICmd(FunctionalState NewState)
{

    /* Check the parameters */
    assert_param(IS_FUNCTIONALSTATE_OK(NewState));

    if (NewState != DISABLE)
    {
        /* Set LSIEN bit */
        CLK->ICKR |= CLK_ICKR_LSIEN;
    }
    else
    {
        /* Reset LSIEN bit */
        CLK->ICKR &= (uint8_t)(~CLK_ICKR_LSIEN);
    }

}
```

BEEP_Cmd(ENABLE)使能 Beep 功能模块，同时也把 LSI 的硬件置位，见下文的说明

函数原型：

```
/**
  * @brief  Enable or disable the BEEP function.
  * @param   NewState Indicates the new state of the BEEP function.
  * @retval None
  * @par Required preconditions:
  * Initialisation of BEEP and LS RC calibration must be done before
  */
void BEEP_Cmd(FunctionalState NewState)
{
    if (NewState != DISABLE)
    {
        /* Enable the BEEP peripheral */
        BEEP->CSR |= BEEP_CSR_BEEPEN;
    }
    else
    {
        /* Disable the BEEP peripheral */
        BEEP->CSR &= (uint8_t)(~BEEP_CSR_BEEPEN);
    }
}
```

LSIEN：低速内部振荡器使能
由软件置位或清除。如果LSI为必需的，则硬件将该位置1，例如：
- 当时钟源切换至LSI时(参见寄存器CLK_SWR)
- 当LSI被指定为时钟输出源(CCO)时(参见寄存器CLK_CCOR)
- 当BEEP被使能时(寄存器BEEP_CSR的位BEEPEN=1)
- 当LSI测量被使能时(寄存器AWU_CSR的位MSR=1)

当LSI被指定为主时钟源/CCO时钟源/AWU/IWDG的时钟源时，该位不能被清除。
0：关闭低速内部振荡器
1：打开低速内部振荡器

BEEP_LSICalibrationConfig(LSI_128kHz)是蜂鸣器校正，让管脚输出标准的 1K，2K，4K 的输出。
函数原型

```
/**
  * @brief  Update CSR register with the measured LSI frequency.
  * @par Note on the APR calculation:
  * A is the integer part of LSIFreqkHz/4 and x the decimal part.
  * x <= A/(1+2A) is equivalent to A >= x(1+2A) and also to 4A >= 4x(1+2A) [F1
  * but we know that A + x = LSIFreqkHz/4 ==> 4x = LSIFreqkHz-4A
  * so [F1] can be written :
  * 4A >= (LSIFreqkHz-4A)(1+2A)
  * @param   LSIFreqHz Low Speed RC frequency measured by timer (in Hz).
  * @retval None
  * @par Required preconditions:
  * - BEEP must be disabled to avoid unwanted interrupts.
  */
void BEEP_LSICalibrationConfig(uint32_t LSIFreqHz)
{

    uint16_t lsifreqkhz;
    uint16_t A;

    /* Check parameter */
    assert_param(IS_LSI_FREQUENCY_OK(LSIFreqHz));

    lsifreqkhz = (uint16_t)(LSIFreqHz / 1000); /* Converts value in kHz */
```

```
    /* Calculation of BEEPER calibration value */

    BEEP->CSR &= (uint8_t)(~BEEP_CSR_BEEPDIV); /* Clear bits */

    A = (uint16_t)(lsifreqkhz >> 3U); /* Division by 8, keep integer part only */

    if ((8U * A) >= ((lsifreqkhz - (8U * A)) * (1U + (2U * A))))
    {
        BEEP->CSR |= (uint8_t)(A - 2U);
    }
    else
    {
        BEEP->CSR |= (uint8_t)(A - 1U);
    }
}
```

实验效果：

蜂鸣器发出固定频率(1K，2K，4K)的声音。

风驰电子祝您学习愉快～～～!!!!!