Using hardware and software to make new stuff

Search

type, hit enter

Categories

- 3D Printing (1)
- Affinity Photo (1)
- Aide-memoir (1)
- Analog (1)
- Ansible (3)
- Dates to Remember (3)
- Docker (1)
- Electronics (144)
- ESP8266 (12)
- FPGA (4)
- Garden (1)
- General (8)
- Home Built CPU (6)
- Informatica (1)
- Internet of Things (8)
- iOS (2)
- KiCad (4)
- MSP430 (2)
- Netduino (49)
- NuttX (9)
- Photography (3)
- Pico (10)
- Raspberry Pi (17)
- Silverlight (6)
- Software Development (88)
- STM32 (5)
- STM8 (54)
- Tips (5)

Archives

- July 2024
- June 2024
- May 2024
- April 2024
- March 2024
- February 2024
- January 2024
- December 2023
- November 2023
- October 2023

- February 2020
- January 2020
- July 2019
- February 2018
- July 2017
- June 2017
- April 2017
- March 2017
- February 2017
- October 2016
- September 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- January 2016
- December 2015
- November 2015
- October 2015
- August 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- January 2013
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012

STM8S Beep Function

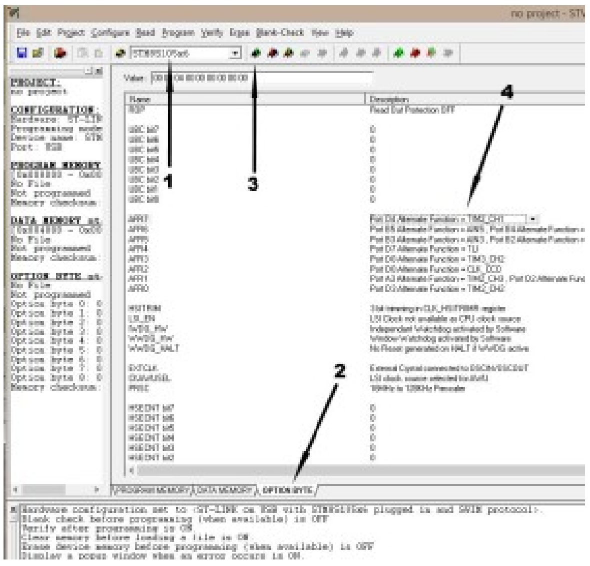
The beep function uses the 128 KHz LSI to generate a beep signal on the beep pin of the STM8S. This post demonstrates how to use this function.

Hardware

The signal is output on the beep pin. On the STM8S discovery board this is an alternative function on PD4 (port D, pin 4).

Setting the Alternative Function

The alternative functions are assigned to a pin by reprogramming the option bytes. This can be performed using the *ST Visual Programmer*. Start this application and read the option bytes from the STM8S:



no project - STV
File Edit Project Configure Read Program Verify Erase Blank-Check View Help
STM8S105x6
1
3
4
2
PROGRAM MEMORY
DATA MEMORY
OPTION BYTE

```
PROJECT:
no project

CONFIGURATION:
Hardware: ST-LINK
Programming mode
Device name: STM
Port: USB

PROGRAM MEMORY
[0x8000 - 0x00
No File
Not programmed
Memory checksum:

DATA MEMORY as
[0x4000 - 0x00
No File
Not programmed
Memory checksum:

OPTION BYTE as
No File
Not programmed
Option byte 0: 0
Option byte 1: 0
Option byte 2: 0
Option byte 3: 0
Option byte 4: 0
Option byte 5: 0
Option byte 6: 0
Option byte 7: 0
Option byte 8: 0
Memory checksum:
```

Value: 00 00 00 00 00 00 00 00

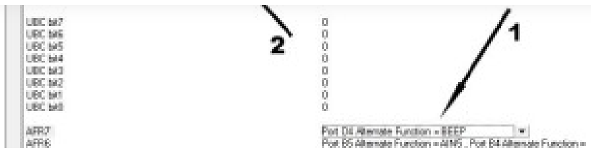
Name	Description
ROP	Read Out Protection OFF
UBC bit7	0
UBC bit6	0
UBC bit5	0
UBC bit4	0
UBC bit3	0
UBC bit2	0
UBC bit1	0
UBC bit0	0
AFR7	Port D4 Alternate Function = TIM2_CH1
AFR6	Port B5 Alternate Function = AIN5 , Port B4 Alternate Function =
AFR5	Port B3 Alternate Function = AIN3 , Port B2 Alternate Function =
AFR4	Port D7 Alternate Function = TLI
AFR3	Port D0 Alternate Function = TIM3_CH2
AFR2	Port D0 Alternate Function = CLK_CCO
AFR1	Port A3 Alternate Function = TIM2_CH3 , Port D2 Alternate Func
AFR0	Port D3 Alternate Function = TIM2_CH2
HSITRIM	3 bit trimming in CLK_HSITRIMR register
LSI_EN	LSI Clock not available as CPU clock source
IWDG_HW	Independant Watchdog activated by Software
WWDG_HW	Window Watchdog activated by Software
WWDG_HALT	No Reset generated on HALT if WWDG active
EXTCLK	External Crystal connected to OSCIN/OSCOUT
CKAWUSEL	LSI clock source selected for AWU
PRSC	16MHz to 128KHz Prescaler
HSECNT bit7	0
HSECNT bit6	0
HSECNT bit5	0
HSECNT bit4	0
HSECNT bit3	0
HSECNT bit2	0

```
Hardware configuration set to <ST-LINK on USB with STM8S105x6 plugged in and SWIM protocol>.
Blank check before programming (when available) is OFF
Verify after programming is ON
Clear memory before loading a file is ON
Erase device memory before programming (when available) is OFF
Display a popup window when an error occurs is ON.
```

Reading the options bytes using ST Visual Developer

1. Select the STM8S model. On the STM8S Discovery board this is STM8S105C6
2. Select the *OPTION BYTE* tab
3. Read the option bytes from the microcontroller
4. Check the value of the AFR7 bit

AFR7 needs to be set to *Port D4 Alternative Function = BEEP* on the STM8S Discovery board. This may be different on your STM8S so check the documentation for your microcontroller. To set the alternative function to



```
UBC bit7    0
UBC bit6    0
UBC bit5    0
UBC bit4    0
UBC bit3    0
UBC bit2    0
UBC bit1    0
UBC bit0    0

AFR7        Port D4 Alternate Function = BEEP
AFR6        Port B5 Alternate Function = AIN5 , Port B4 Alternate Function =
```

Writing new options bytes using ST Visual Developer

1. Select *Port D4 Alternative Function = BEEP* from the list of values in the alternative functions
2. Write the option bytes back to the microcontroller

The beep function should now be assigned to PD4.

Registers

The beep function is controlled by three registers:

- BEEP_CSR_BEEPSEL
- BEEP_CSR_BEEPDIV
- BEEP_CSR_BEEPEN

Beep Selection – BEEP_CSR_BEEPSEL

These bits set the multiplier for the Beep Divider.

BEEP_CSR_BEEPSEL	Frequency (KHz)
0	$f_{LSI} / (8 * BEEP_{DIV})$
1	$f_{LSI} / (4 * BEEP_{DIV})$
2 or 3	$f_{LSI} / (2 * BEEP_{DIV})$

Beep Divider – BEEP_CSR_BEEPDIV

Value for the prescalar divider.

$BEEP_{DIV}$ = 2 + BEEP_CSR_BEEPDIV

The documentation states that this should not be left set to the reset value (0x1f) and the value should be in the range 0 – 0x1e.

Beep Enable – BEEP_CSR_BEEPEN

Enable the beep function by setting this to 1, disable by setting this to 0.

Software

We start by providing a method for resetting the system clock to use the HSI:

```
//
//  This program demonstrates how to use the beep function on the STM8S
//  microcontroller.
//
```

```
//
#include <iostm8S105c6.h>
#include <intrinsics.h>

//--------------------------------------------------------------------------------
//
//  Setup the system clock to run at 16MHz using the internal oscillator.
//
void InitialiseSystemClock()
{
    CLK_ICKR = 0;                       //  Reset the Internal Clock Register
    CLK_ICKR_HSIEN = 1;                 //  Enable the HSI.
    CLK_ECKR = 0;                       //  Disable the external clock.
    while (CLK_ICKR_HSIRDY == 0);       //  Wait for the HSI to be ready for
    CLK_CKDIVR = 0;                     //  Ensure the clocks are running at
    CLK_PCKENR1 = 0xff;                 //  Enable all peripheral clocks.
    CLK_PCKENR2 = 0xff;                 //  Ditto.
    CLK_CCOR = 0;                       //  Turn off CCO.
    CLK_HSITRIMR = 0;                   //  Turn off any HSIU trimming.
    CLK_SWIMCCR = 0;                    //  Set SWIM to run at clock / 2.
    CLK_SWR = 0xe1;                     //  Use HSI as the clock source.
    CLK_SWCR = 0;                       //  Reset the clock switch control re
    CLK_SWCR_SWEN = 1;                  //  Enable switching.
    while (CLK_SWCR_SWBSY != 0);        //  Pause while the clock switch is b
}
```

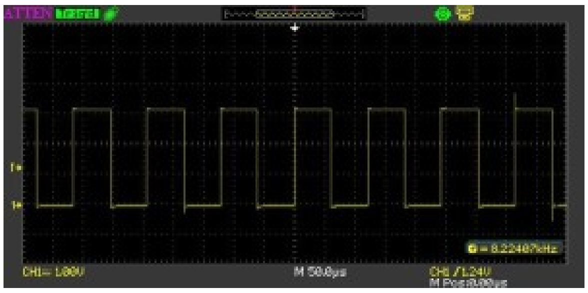
The next task is to configure the beep function:

```
//--------------------------------------------------------------------------------
//
//  Initialise the Beep function.
//
void InitialiseBeep()
{
    BEEP_CSR_BEEPEN = 0;     //  Turn off the beep.
    BEEP_CSR_BEEPSEL = 0;    //  Set beep to fls / (8 * BEEPDIV) KHz.
    BEEP_CSR_BEEPDIV = 0;    //  Set beep divider to 2.
    BEEP_CSR_BEEPEN = 1;     //  Re-enable the beep.
}
```

The final piece of code is the main program loop. This sets up the clock and the beep function:

```
//--------------------------------------------------------------------------------
//
//  Main program loop.
//
void main()
{
    //
    //  Initialise the system.
    //
    __disable_interrupt();
    InitialiseSystemClock();
    InitialiseBeep();
    while (1)
    {
        __halt();
    }
}
```

Compiling this code and deploying to the STM8S results in the following trace on the oscilloscope:



Beep Oscilloscope Output

As you can see, the output is not exactly 8KHz.

Conclusion

The documentation states that the beep function can generate 1KHz, 2KHz and 4KHz. By changing the values of the prescalar and the selection register it appears that you can also go as low as 500Hz and as high as 32KHz.

Tags: Electronics, Software Development, STM8, The Way of the Register

Tuesday, July 1st, 2014 at 1:00 am • Electronics, Software Development, STM8 • RSS 2.0 feed Both comments and pings are currently closed.

Comments are closed.

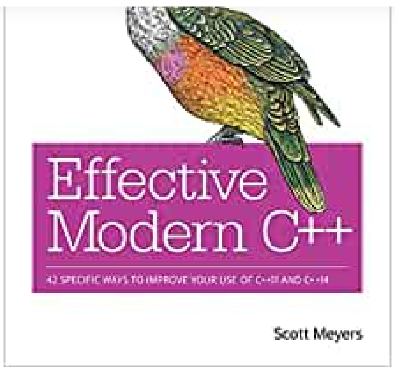
Pages

- About
- Making a Netduino GO! Module
- The Way of the Register
- Making an IR Remote Control
- Weather Station Project
- NuttX and Raspberry Pi PicoW

Support This Site

Buy me a coffee

Currently Reading


Effective Modern C++
42 SPECIFIC WAYS TO IMPROVE YOUR USE OF C++11 AND C++14
Scott Meyers