Trimming the S08P MCU

by: David Diaz Marin

1 Introduction

In case of S08P MCU family, the newest family from the 8-bit MCU's, it is recommended to trim the internal oscillator.

All MCU devices are factory programmed with a trim value in a reserved memory location. This trim value can be copied to the SCTRIM register during reset initialization. The factory trim value includes the FTRIM bit. For finer precision, the user can trim the internal oscillator in the application to take in account small differences between the factory test setup and actual application conditions.

The purpose of this document is to demonstrate the importance about trimming a microcontroller unit (MCU) showing the differences between an untrimmed and a trimmed device.

Contents

2 Device overview

The 8-bit S08P MCU family gives your designs more durability and reliability in both harsh industrial and user interface environments.

For more information please visit the 8-bit S08P MCU web site at:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=S08P

3 S08P Clock management

3.1 Internal clock source (ICS)

The internal clock source (ICS) module provides clock source options for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by an internal or external reference clock. The module can provide this FLL clock or the internal reference clock as a source for the MCU system clock, ICSCLK.

Whichever clock source is chosen, ICSCLK is the output from a bus clock divider, which allows a lower clock frequency to be derived.

One of the key features of the ICS Module is that the Frequency-locked loop (FLL) is **trimmable** for accuracy.

For more detailed information about the ICS, please refer to the Application Note AN3499:

http://www.freescale.com/files/microcontrollers/doc/app_note/AN3499.pdf

3.1.1 Clock distribution

The following figure shows a simplified clock connection diagram.

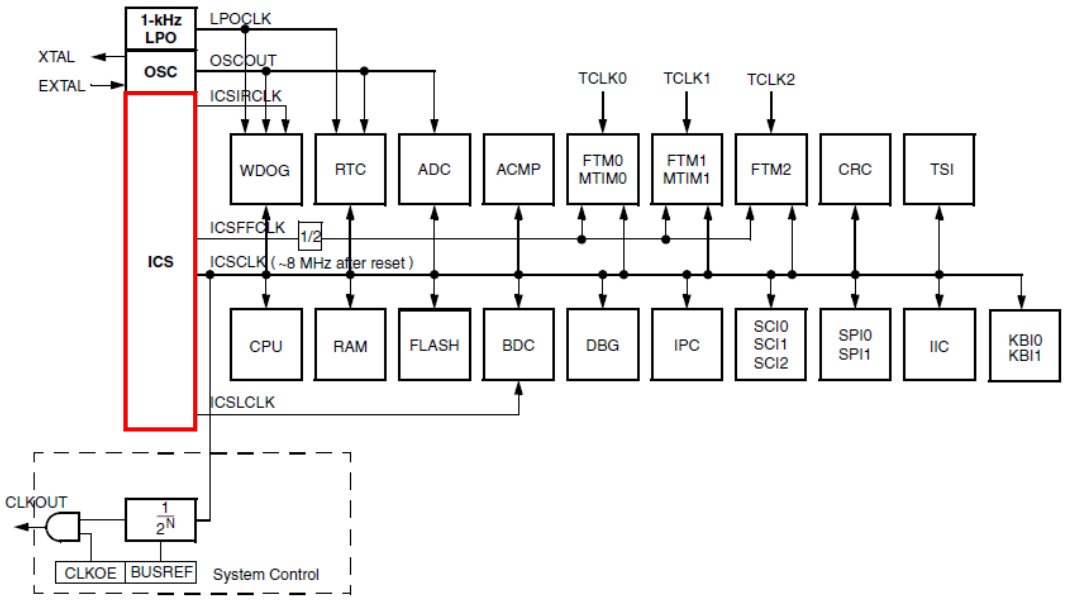


Figure 1. System Clock distribution diagram

3.1.2 Functional description

The ICS block diagram is shown in Figure 2.

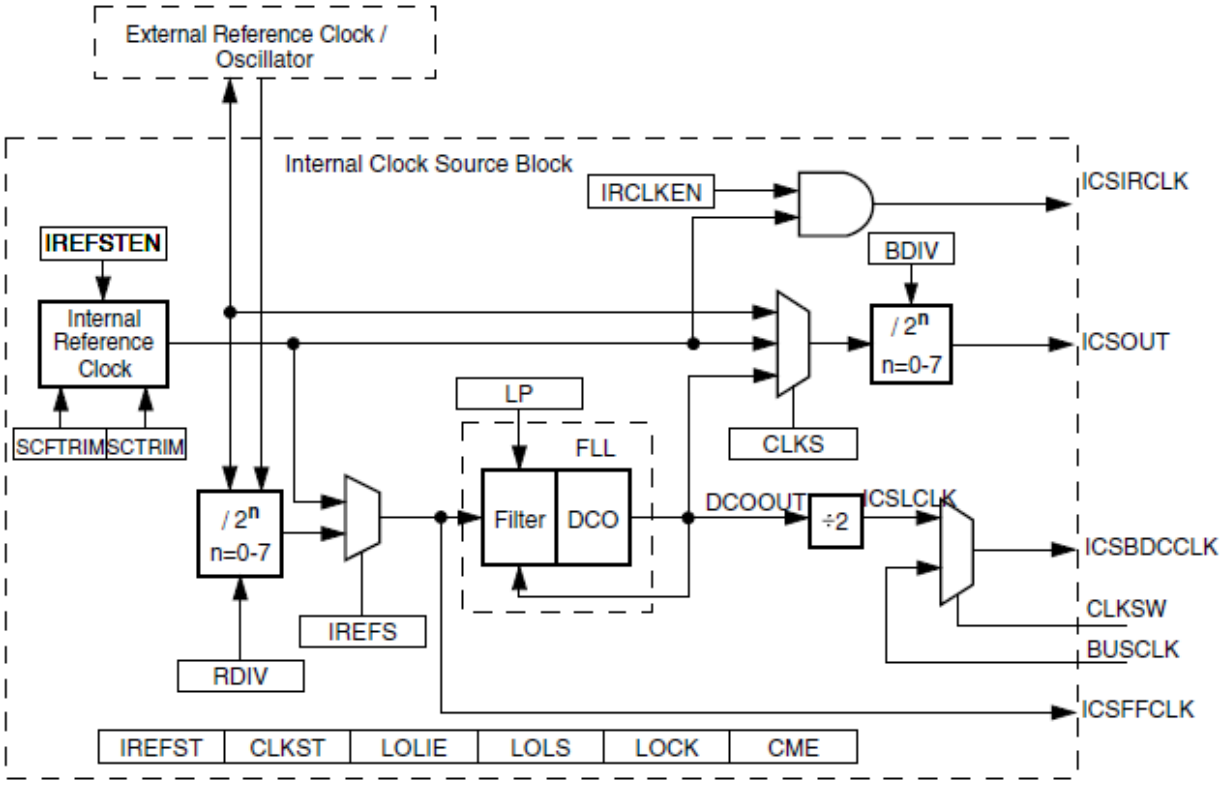


Figure 2. Internal clock source (ICS)

3.1.3 Registers

The bits involved in the trimming process are the SCTRIM and SCFTRIM from the ICS Control Registers, ICS_C3 and ICS_C4. At POR, the SCTRIM and SCFTRIM settings are reset to 0x80 and 0 respectively.

Please find more information about the ICS Registers at S08P Reference Manual, Chapter 8, from the following link:

http://www.freescale.com/files/microcontrollers/doc/ref_manual/MC9S08PT60RM.pdf

4 Demonstration

It is quite important to add the following code lines in order to set the system clock initialization:

```
//These code lines copy the trim value to the registers mentioned above:

if (*(unsigned char*)0xFF6F != 0xFF) { /* Test if the device trim value is stored on
the specified address */
     ICS_C3 = *(unsigned char*)0xFF6F;  /* Initialize ICSTRM register from a non
volatile memory */
     ICS_C4 = (unsigned char)((*(unsigned char*)0xFF6E) & (unsigned char)0x01); /*
Initialize ICSSC register from a non volatile memory */
   }


//After that, the clock source is initialized with the following configuration:

/* Initialization of the ICS control register 1 */
  ICS_C1 = 0x06;          /* Output of FLL is selected, RDIV=1 */
  /* Initialization of the ICS control register 2 */
  ICS_C2 = 0x20;          /* Divides selected clock by 2.*/

/* ICS_C4: LOLIE=0,CME=0 */

  ICS_C4 = 0x00;
```

To demonstrate, a PWM signal is generated at 1 KHz, using the bus clock as a source clock.

```
// FTM Initialization
 FTM2_SC = 0x48;  // Timer Overflow Interrupt Enable, Bus Clock/2
 FTM2_MOD = 4000; // PWM frequency is about 1 KHz

 // Configure FTM2 Channel 0
 FTM2_C0SC = 0x28; //Edge-aligned PWM (high-true pulses)
 FTM2_C0V = 2000;  // channel 0 set to 50%
```

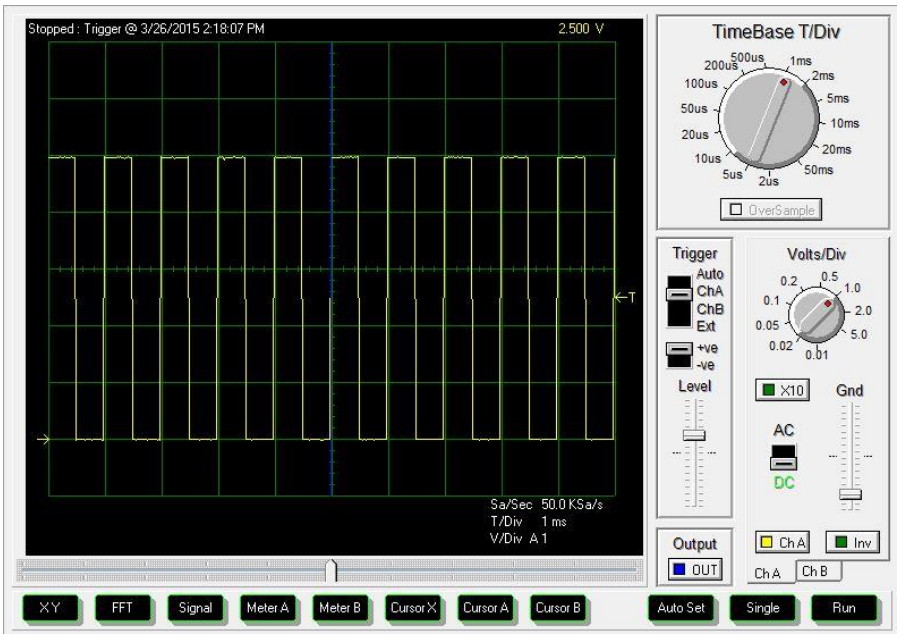


Figure 3. PWM signal at 1 KHz

If we do not use following code lines, the internal reference frequency will not be copied to the ICS registers, and therefore, the output PWM signal will be as the following image shows:

```
// if (*(unsigned char*)0xFF6F != 0xFF) { /* Test if the device trim value is stored
on the specified address */
//      ICS_C3 = *(unsigned char*)0xFF6F;  /* Initialize ICSTRM register from a non
volatile memory */
//      ICS_C4 = (unsigned char)((*(unsigned char*)0xFF6E) & (unsigned char)0x01); /*
Initialize ICSSC register from a non volatile memory */
//    }
```

Note:
In order to change the internal reference frequency, please refer to Chapter 5.

Please note that The FLL loop locks the frequency to the 512 times the internal reference frequency.

If we add the code mentioned and the Factory trim value, **32768 Hz**, is used as an internal reference frequency, we have the following results:

$$\frac{(Reference\ Frequency)(512)}{RDIV} = \frac{(32768)(512)}{1} = 16.77\ MHz$$

$$\frac{16.77\ MHz}{2} = 8.38\ MHz = Bus\ Clock$$

With the Bus clock running at 8.38 MHz and using the FTM code lines, the output PWM signal will be as the following image shows:

$$\frac{8.38\ MHz}{(4000)(2)} = 1048.576\ Hz$$

PWM output signal with the factory trim value as Internal Reference clock.

It is possible to change the internal reference clock in order to a finer PWM output signal.

PWM output signal with 31250 Hz as Internal Reference clock.

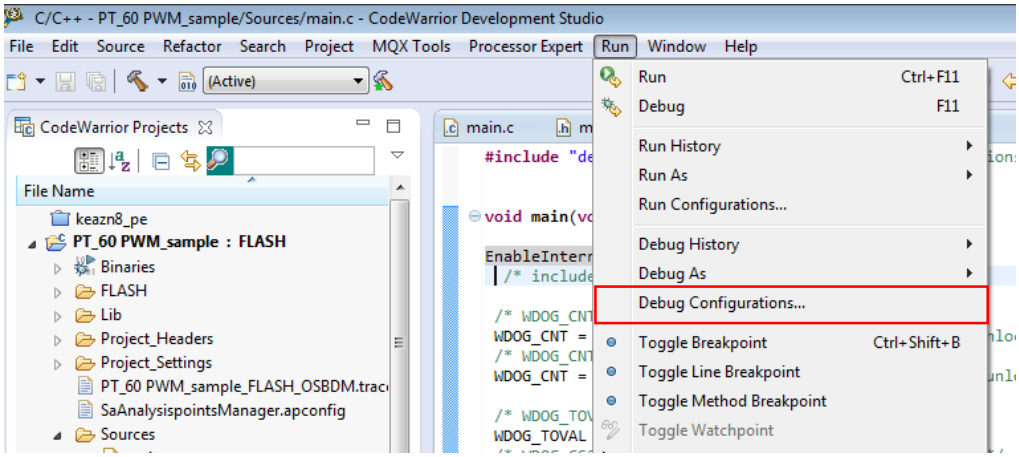
PWM output signal with 39062 Hz as Internal Reference clock.

5 Using CodeWarrior

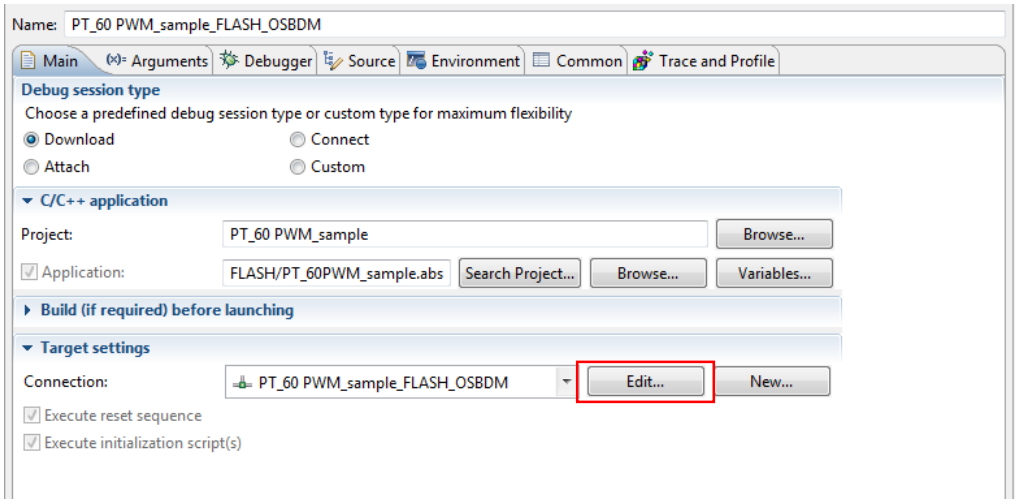
Once you have your project in CodeWarrior, it is possible to change the internal reference frequency.

Please follow the next steps:

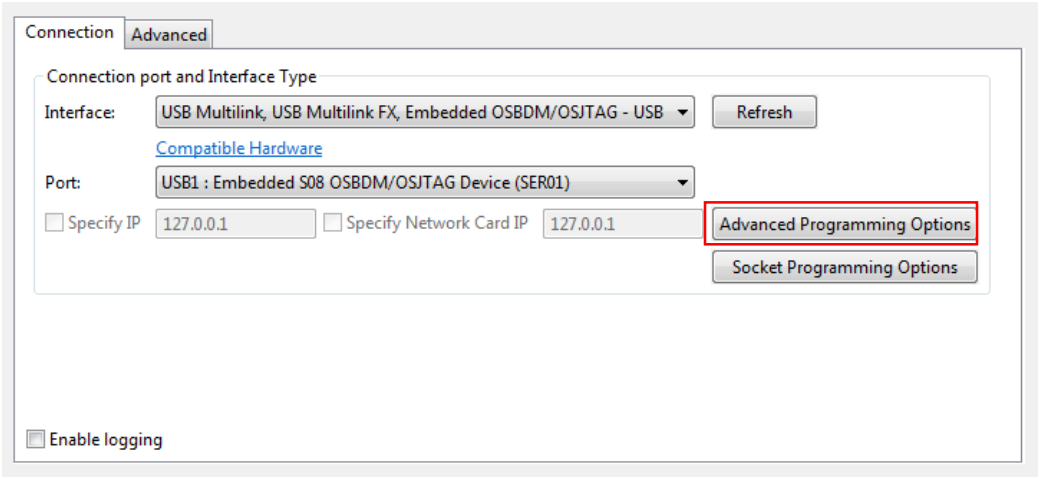
1. Click Debug Configurations from the Run tool.



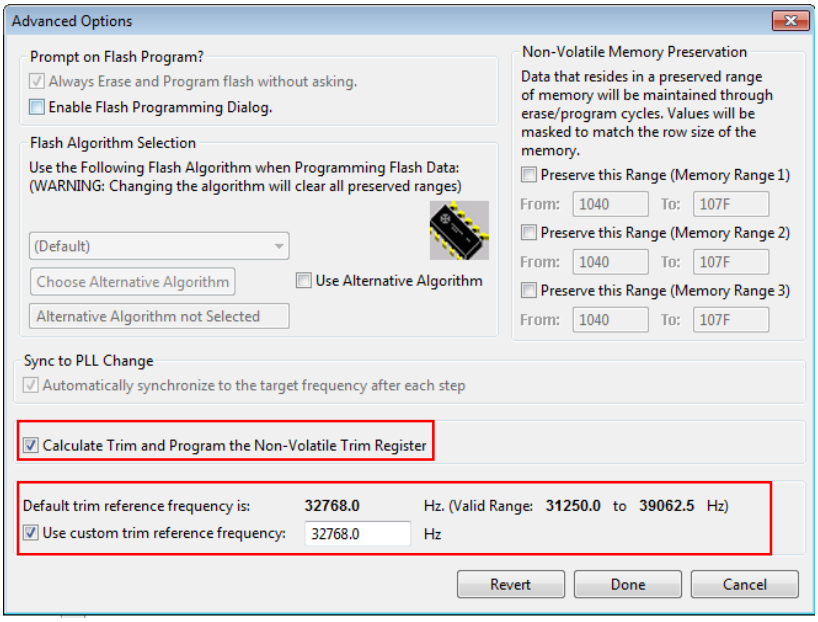
2. Click Edit in Target Settings.



3. Click Advanced Programming Options.



4. Make sure you select “Use custom trim reference frequency” in order to customize the trim frequency.



6 Conclusion

As it is mentioned, it is important to add the code lines which copy the internal reference frequency to the ICS registers.

The following table shows the results with different internal reference frequency values:

Reference Frequency	Bus Clock	PWM output (ideal)	PWM output (real)	% Error
32768 Hz *	8.38 MHz *	1.047 KHz *	1.051 KHz	0.38
31250 Hz	8 MHz	1 KHz	1.001 KHz	0.1
32768 Hz	8.38 MHz	1.047 KHz	1.045 KHz	0.19
39062 Hz	9.99 MHz	1.249 KHz	1.246 KHz	0.24

* Typical value if the code lines mentioned are not included.