

Understanding S08P Internal Clock Source

By: William Jiang

1 Introduction

The S08P family integrates an internal clock source module (ICS) which provides clock source options for the system. The clock source can be from an internal reference clock or external reference clock. To use an external crystal or resonator as the reference clock, the external oscillator module (XOSC) must be enabled. With both the ICS module and the XOSC module, the family can provide six operational modes and one stop mode to meet different application requirements.

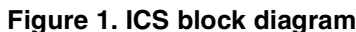
This application note describes the use of both modules as well as the ICS operation mode switch with the sample code provided.

2 Overview of ICS

The following figure shows the ICS module block diagram:

Contents

1. Introduction	1
2. Overview of ICS	1
2.1. ICS Module	2
3. Overview of XOSC	3
4. ICS Operation Mode	4
5. Conclusion	9
6. References	9
7. Glossary	10



The ICS module provides several clock source options for S08P devices. The module contains a frequency-locked loop (FLL) that is controllable by either an internal or external reference clock.

The internal reference clock has nine trim bits available, and can be trimmed from 31.25 kHz to 39.0625 kHz. The typical trim value at the factory is 32.768 kHz, 31.25 kHz, or 39.0625 kHz, depending on the different devices. The module can select a clock from the FLL or bypass the FLL as a source of the MCU system clock. The selected clock source is passed through a reduced bus divider controlled by BDIV to generate ICSOUT, which allows a lower output clock frequency to be derived. The FLL loop locks the frequency to the fixed 512 multiplied by the reference frequency. Assume the FLL reference frequency (ICSFFCLK) is F_{ref} , then the FLL output frequency after lock (F_{dcoout}) is $(F_{ref} * 512)$.

In addition, the FLL reference frequency must be within the range of 31.25 kHz to 39.0625 kHz. This can be fulfilled via ICS_C1[RDIV] bits and the XOSC ICS_OSCSC[RANGE] bit. The digitally-controlled oscillator (DCO) is optimized for 16 MHz to 20 MHz frequency range. The total deviation of DCO output from trimmed frequency for over fixed voltage and temperature range of 0 to 70° C can be within $\pm 1\%$.

In Figure1, ICSOUT is the bus clock provided to the core and all peripherals. ICSFFCLK is the fixed reference clock, and it must be no more than 1/4 of the ICSOUT frequency to be valid. This clock can be used as a FlexTimer clock source. ICSIRCLK is internal reference clock output, and can be selected as the clock source to the WDOG module. The frequency of ICSIRCLK can be trimmed via ICS_C3[SCTRIM] and ICS_C4[SCFTRIM] bits. Writing a larger value to the trim bits slows down the ICSIRCLK frequency, while writing a smaller value to the trim bits speeds up the ICSIRCLK frequency.

ICSLCLK is derived from the DCO, and is a BDC alternate clock, which can be used by development tools to speed up BDC communications in systems where the bus clock is slow. ICSBDCCLK is the actual BDC clock. CORECLK is the core clock, and in this family it is the same as ICSOUT.

3 Overview of XOSC

The external oscillator (XOSC) module allows an external crystal, ceramic resonator, or other external clock sources to produce the external reference clock. The oscillator is a Pierce-type oscillator. The output of the XOSC module can be used as the reference of ICS to generate a system bus clock, and/or a clock source of watchdog (WDOG), real-time counter (RTC), and/or analog-to-digital (ADC) modules. It supports an external clock from 4 MHz to 20 MHz in high range mode (high frequency mode) and 32 kHz crystals in low range mode (low frequency mode) as shown in the following table.

Table 1. XOSC Modes

ICS_OSCSC[RANGE]	ICS_OSCSC[HGO]	Mode	Frequency Range	Connection
0	0	Low frequency, low power	32 kHz	Crystal Connection 1
0	1	Low frequency, high gain	32 kHz	Crystal Connection 2
1	0	High frequency, low Power	4 MHz to 20 MHz	Crystal Connection 3
1	1	High frequency, high gain	4 MHz to 20 MHz	Crystal Connection 2

The possible crystal/resonator connections are shown in the following figures.

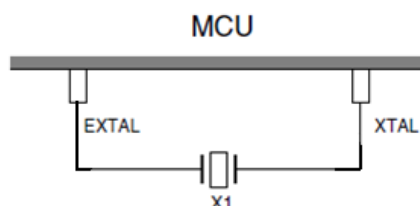


Figure 2. Crystal connection 1

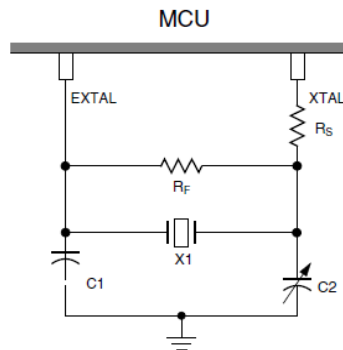


Figure 3. Crystal connection 2

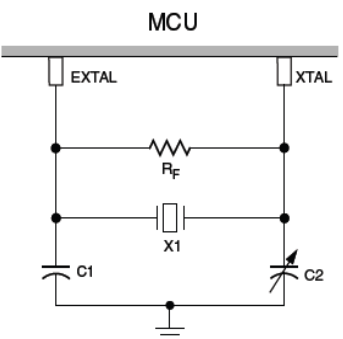


Figure 4. Crystal connection 3

The recommended component values for R_s , R_f , C_1 and C_2 can be found in the device data sheet.

In addition, the XOSC also supports direct input from an external clock in bypass mode ($ICS_OSCSC[OSCEN] = 0$, and $ICS_OSCSC[OSCOS] = 0$) as shown in the figure below:

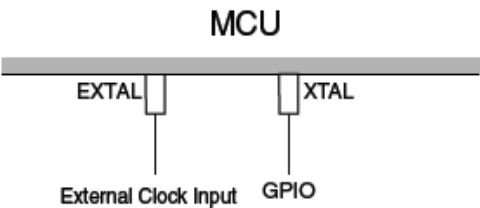


Figure 5. Clock connection

4 ICS Operation Mode

The ICS has seven modes of operation: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop. Figure 6 shows the seven states of the ICS as a state diagram. The arrows indicate the allowed movements between the states. Out of reset, it is in FEI mode. To enter one of other modes, the arrow must be followed.

When in low power mode (FBELP or FBILP), FLL is disabled, so the BDC alternate clock ICSLCLK is not active and will not be used as BDC clock source for BDC communications (for example, when the

BDM debugger is used). In this case, the MCU bus clock must be selected as the BDC clock for BDC communications.

When FLL is engaged (FEx) or bypassed (FBx), the reference clock to FLL must be configured within 31.25 kHz to 39.0625 kHz.

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1;
- IREFSTEN bit is written to 1.

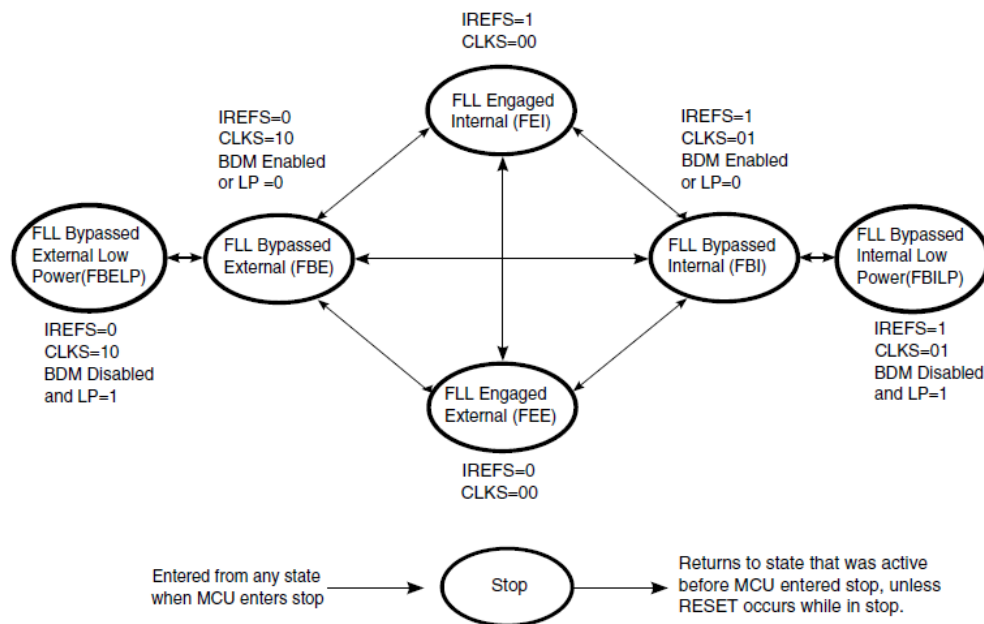


Figure 6. ICS operation mode

The following code fragment shows how to transition to clock mode from FEI to FEE mode.

```

void FEI_to_FEE(void)
{
    /* enable OSC with high gain, high range and select oscillator output as OSCOUT */
    /*
    ICS_OSCSC = ICS_OSCSC_OSCEN_MASK
    | ICS_OSCSC_OSCSTEN_MASK          /* enable stop */
    #if defined(CRYST_HIGH_GAIN)
    | ICS_OSCSC_HGO_MASK /* Rs must be added and be large up to 200K */
    #endif
    #if (EXT_CLK_CRYST >= 4000)
    | ICS_OSCSC_RANGE_MASK
    #endif
    | ICS_OSCSC_OSCOS_MASK; /* for crystal only */
    #if defined(IAR)
    asm(

```

ICS Operation Mode

```

        "nop \n"
        "nop \n"
    );
#elif defined(__MWERKS__)
    asm{
        nop
        nop
    };
#endif

    /* wait for OSC to be initialized
    *
    */
    while(!(ICS_OSCSC & ICS_OSCSC_OSCINIT_MASK));

    /* divide down external clock frequency to be within 31.25K to 39.0625K
    *
    */
    #if (EXT_CLK_CRYST == 8000)
        /* 8MHz */
        ICS_C1_RDIV = 3; /* now the divided frequency is 8000/256 = 31.25K */
    #elif (EXT_CLK_CRYST == 4000)
        /* 4MHz */
        ICS_C1_RDIV = 2; /* now the divided frequency is 4000/128 = 31.25K */
    #elif (EXT_CLK_CRYST == 16000)
        /* 16MHz */
        ICS_C1_RDIV = 4; /* now the divided frequency is 16000/512 = 31.25K */
    #elif (EXT_CLK_CRYST == 20000)
        /* 20MHz */
        ICS_C1_RDIV = 4; /* now the divided frequency is 20000/512 = 39.0625K */
    #elif (EXT_CLK_CRYST == 32)
        ICS_C1_RDIV = 0;
    #else
        #error "Error: crystal value not supported!\n";
    #endif

    /* change FLL reference clock to external clock */
    ICS_C1_IREFS = 0;
    /* wait for the reference clock to be changed to external */
    #if defined(IAR)
        asm(
            "nop \n"
            "nop \n"
        );
    #elif defined(__MWERKS__)
        asm{
            nop
            nop
        };
    #endif

    while(ICS_S & ICS_S_IREFST_MASK);

    /* wait for FLL to lock */
    while(!(ICS_S & ICS_S_LOCK_MASK));

    if(((ICS_C2 & ICS_C2_BDIV_MASK)>>5) != 0)

```

```

    {
        ICS_C2_BDIV = 0;
    }

    /* clear Loss of lock sticky bit */
    ICS_S |= ICS_S_LOLS_MASK;
}

```

The following code fragment shows how to transition to clock mode from FEE to FEI mode:

```

void FEE_to_FEI(void)
{
    ICS_C1_IREFS = 1; /* select internal reference for FLL */

    /* wait for the reference clock to be changed */
#ifdef IAR
    asm(
        "nop \n"
        "nop \n"
    );
#elif defined(__MWERKS__)
    asm{
        nop
        nop
    };
#endif
    while(!(ICS_S & ICS_S_IREFST_MASK));

    /* wait for FLL to lock */
    while(!(ICS_S & ICS_S_LOCK_MASK));

    /* clear Loss of lock sticky bit */
    ICS_S |= ICS_S_LOLS_MASK;

    if(((ICS_C2 & ICS_C2_BDIV_MASK)>>5) != 0)
    {
        ICS_C2_BDIV = 0;
    }
}

```

The following code fragment shows how to transition to clock mode from FEI to FBE mode:

```

void FEI_to_FBE(void)
{
    /* assume external crystal is 8Mhz or 4MHz
    *
    */
    /* change clock source to external reference clock */
    ICS_C1_IREFS = 0;
    ICS_C1_CLKS = 2;
    ICS_C2_LP = 0;
    /* wait for the reference clock to be changed */
#ifdef IAR
    asm(
        "nop \n"
        "nop \n"
    );

```

ICS Operation Mode

```
#elif defined(__MWERKS__)
    asm(
        nop
        nop
    );
#endif

while(ICS_S_CLKST != 2);
while(ICS_S_IREFST);

/* now external clock is the system clock
 *
 */
if(((ICS_C2 & ICS_C2_BDIV_MASK)>>5) != 0)
{
    ICS_C2_BDIV = 0;
}
/* now system/bus clock is external clock
 *
 */
/* clear Loss of lock sticky bit */
ICS_S |= ICS_S_LOLS_MASK;
}
```

The following code fragment shows how to transition to clock mode from FBE to FEI mode:

```
void FBE_to_FEI(void)
{
    ICS_C1_IREFS = 1; /* select internal reference clock */
    ICS_C1_CLKS = 0; /* select the internal clock as clock source */

    /* wait for the clock source to be changed */
#ifdef IAR
    asm(
        "nop \n"
        "nop \n"
    );
#endif
#ifdef __MWERKS__
    asm(
        nop
        nop
    );
#endif

while(ICS_S_CLKST);
while(ICS_S_IREFST != 1);

/* now system clock source is internal clock
 *
 */
if(((ICS_C2 & ICS_C2_BDIV_MASK)>>5) != 0)
{
    ICS_C2_BDIV = 0;
}
/* clear Loss of lock sticky bit */
ICS_S |= ICS_S_LOLS_MASK;
}
```


The following code fragment shows how to transition to clock mode from FEI to FBI mode:

```
void FEI_to_FBI(void)
{
    /* change clock source to internal reference clock */
    ICS_C1_CLKS = 1;
    ICS_C2_LP = 0;
    /* wait for the reference clock to be changed */
#ifdef IAR
    asm(
        "nop \n"
        "nop \n"
    );
#elif defined(__MWERKS__)
    asm{
        nop
        nop
    };
#endif
    while(ICS_S_CLKST !=1);

    /* now internal reference clock is the system clock
     *
     */
    if(((ICS_C2 & ICS_C2_BDIV_MASK)>>5) != 0)
    {
        ICS_C2_BDIV = 0;
    }
    /* now system/bus clock is around 32KHz
     *
     */
    /* clear Loss of lock sticky bit */
    ICS_S |= ICS_S_LOLS_MASK;
}
```

The following code fragment shows how to transition to clock mode from FBI to FBILP mode:

```
void FBI_to_FBILP(void)
{
    ICS_C2_LP = 1; /* enter low power mode */
}
```

5 Conclusion

This application note describes S08P internal clock source related modules, and introduces the external clock connections and ICS operation mode switch with example code fragments to help users understand the internal clock source.

6 References

The following are a few useful references regarding this document.

- MC9S08PT60/32 Reference Manual
- MC9S08PT60 Data Sheet

7 Glossary

This is a compilation of commonly used abbreviations used in this document.

BDC	Background Debug Controller
BDM	Background Debug Mode
ICS	Internal Clock Source module
XOSC	External Oscillator

**How to Reach Us:**

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo and Freescale, the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number: AN4763
Rev. 0
8/2013

