Freescale Semiconductor
User Guide

Rev. 1, 08/2009

S12XE IPLL Calculator

Suitable to use with the S12XE, XF, XS, and S12P, HY, HA families

by: Michael Galda
Freescale Roznov CSC (TIC Team)

1 Introduction

The S12XE (S12XF, S12XS, S12P, S12HY) family of MCU's include an internal phase-locked loop (IPLL) frequency multiplier with an internal filter as a part of the S12XECRG (or S12CPMU) module. The purpose of the PLL is to generate an internal timebase from the external resonator signal or from the internal-reference clock (S12P and S12HY only). The IPLL allows the internal timebase (usually called the bus clock) to be generated at higher or lower frequency than the oscillator signal. The usage of a low-frequency resonator facilitates lower power consumption in low-power modes (pseudo stop mode). The usage of a cheaper low-frequency resonator with PLL to increase the internal MCU bus clock instead of usage of high-frequency external (canned) oscillator, may reduce the final costs of design. The PLL adds more flexibility in order to generate a wide range of internal MCU bus clock frequencies from the fixed external resonator frequency. The user can easily switch between the bus frequencies, depending on application performance or power consumption requirements.

Contents

freescale™ semiconductor

The IPLL allows to set an appropriate internal MCU bus clock, in order to achieve correct timing for the internal MCU modules (like SCI, timer modules, and so on), even if the external resonator frequency is not suitable for proper timing. The IPLL (unlike the PLL module on S12(X)) doesn't require any external filter components.

NOTE

Sections referring to S12P are generally valid also for the S12HY family, due to module compatibility.

2 IPLL Register Setting Description

The SYNR, REFDV, and POSTDIV registers are responsible for the PLL frequency settings.

The following rules, equations, and frequency limitations are considered by the IPLL calculator utility to achieve maximum stability and shortest PLL locking time.

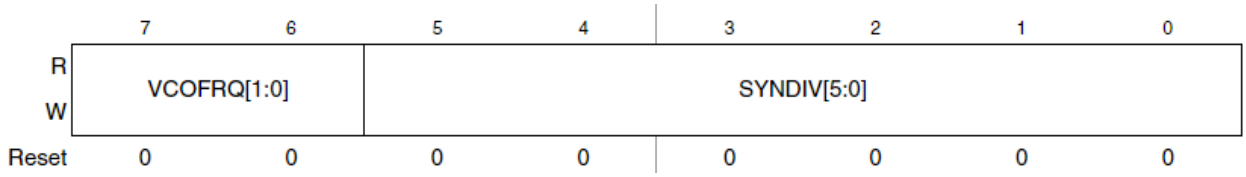

Figure 1. The S12XE(XS) SYNR / S12P CPMUSYNR Register

The VCOFRQ[1:0] bits are used to configure the VCO gain for optimal stability and lock time. For correct IPLL operation, the VCOFRQ[1:0] bits have to be selected according to the actual target VCOCLK frequency, as shown in the following table:

Table 1. VCO Frequency Selection

VCOCLK Frequency Ranges	VCOFRQ[1:0]
32 MHz <= fvco <= 48 MHz	00
48 MHz < fvco <= 80 MHz	01
Reserved	10
80 MHz < fvco <=120 MHz	11

Setting the VCOFRQ[1:0] bits wrongly can result in a non-functional IPLL (no locking and/or insufficient stability).

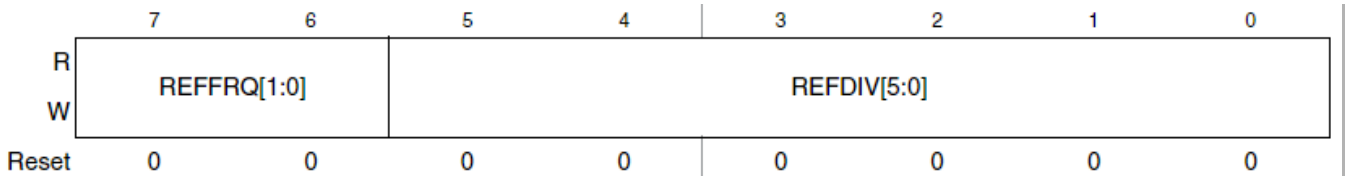

Figure 2. The S12XE(XS) CRG Reference Divider Register (REFDIV)

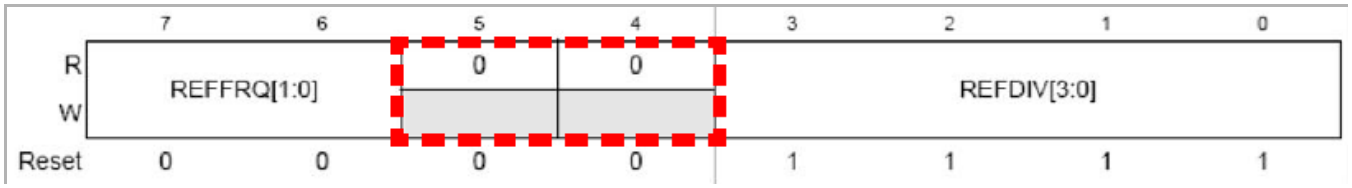

Figure 3. The S12P CPMU Reference Divider Register (CPMUREFDIV)

The REFFRQ[1:0] bits are used to configure the internal IPLL filter for optimal stability and lock time. For correct IPLL operation, the REFFRQ[1:0] bits have to be selected according to actual REFCLK frequency as shown in the following table:

Table 2. Reference Clock Frequency Selection

REFCLK Frequency Ranges	REFFRQ[1:0]
1 MHz <= fref <= 2 MHz	00
2 MHz < fref <= 6 MHz	01
6 MHz < fref <= 12 MHz	10
fref > 12 MHz	11

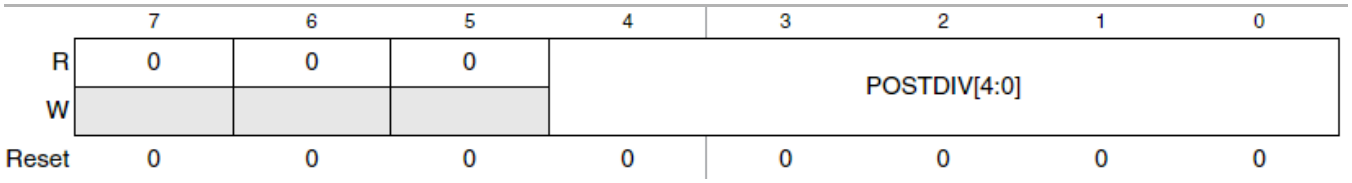

Figure 4. The S12XE(XS) CRG / S12P CPMU Postdiv Register

NOTE

If POSTDIV = $00, the f_{PLL} is identical to f_{VCO} (divide by one).

2.1 PLL Setting Equations

2.1.1 Equations Valid for S12XE

$$f_{VCO} = 2 \times f_{OSC} \times \frac{(SYNDIV + 1)}{(REFDIV + 1)} \qquad \textit{Eqn. 1}$$

$$f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)} \qquad \textit{Eqn. 2}$$

$$f_{PLL} = \frac{f_{VCO}}{(2 \times POSTDIV)} \qquad \textit{Eqn. 3}$$

If PLL is selected (PLLSEL = 1):

$$f_{BUS} = \frac{f_{PLL}}{2}$$

Eqn. 4

2.1.2 Formulas Valid for S12P (S12HY)

If PLL is locked (LOCK = 1), then:

$$f_{PLL(CRG)} = \frac{f_{VCO}}{(1 + POSTDIV)}$$

Eqn. 5

If PLL is not locked (LOCK = 0), then:

$$f_{PLL(CPMU)} = \frac{f_{VCO}}{4}$$

Eqn. 6

If the external oscillator is enabled (OSCE = 1) — PLL engaged external mode (PEE), then:

$$f_{REF} = \frac{f_{OSC}}{(REFDIV + 1)}$$

Eqn. 7

If the external oscillator is disabled (OSCE = 0) — PLL Engaged Internal Mode (PEI), then:

$$f_{REF} = f_{IRC1M} = 1.000MHz$$

Eqn. 8

If PLL is selected (PLLSEL = 1), then:

$$f_{BUS} = \frac{f_{PLL}}{2}$$

Eqn. 9

Table 3. Frequency Limitations for S12XE, XS, S12P, and S12HY

	S12XE(XF)		S12XS		S12P (S12HY)		–	NOTE
frequency	min.	max.	min.	max.	min.	max.	–	
$f_{OSC\ (LCP)}$	4	16	4	16	4	16	MHz	loop-controlled pierce
$f_{OSC\ (FSP)}$	2	40	2	40	-	–	MHz	full-swing pierce
$f_{OSC\ (ext.)}$	2	50	2	50	-	–	MHz	external square clock
f_{BUS}	0.5	50	0.5	40	0.5	32	MHz	
f_{REF}	1	40	1	40	1	40	MHz	
f_{VCO}	32	120	32	120	32	64	MHz	

NOTE: f_{OSC} is limited to 4–16 MHz in LCP mode, 2–40 MHz in FSP mode, and 2–50 MHz for external clk.

NOTE

The following rules help to achieve optimum stability and shortest lock time possible:

- Use the lowest possible f_{VCO} / f_{REF} ratio (SYNDIV value).
- Use the highest possible REFCLK frequency f_{REF}.

3 The IPLL Calculator Application

The S12XE IPLL filter calculator application has been written in free Borland Turbo C++ Explorer IDE, available at http://www.turboexplorer.com/cpp.

Calculator is run by executing the file S12XE_IPLL_Calc.exe on a PC, running a Windows™ XP OS. The application window shown in Figure 5 is presented.

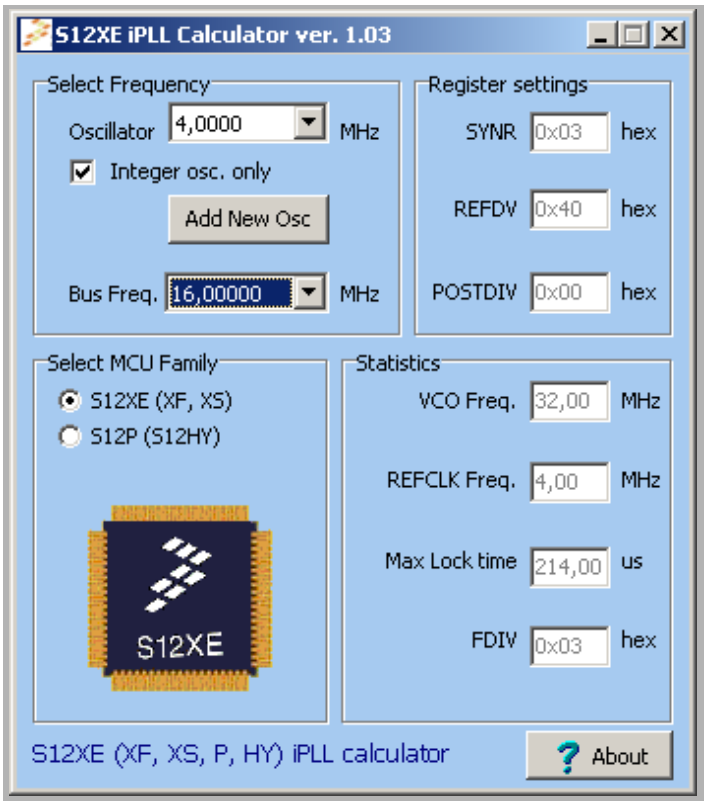

Figure 5. The S12XE IPLL Calculator

3.1 Data Entry

The following data is required in order to calculate the correct register values.

3.1.1 The MCU Family

Since there are some differences between S12XE, XS, S12P, and S12HY register settings and frequency limitations (See Section 2.1, “PLL Setting Equations”), the user has to select the appropriate MCU family by the radio button.

3.1.2 Oscillator Frequency

The oscillator frequency in units of MHz is picked from the Oscillator listbox. A PC mouse or keyboard input can be used. The basic set of the most often manufactured resonator frequencies in range 2–50 MHz are stored in the internal database. Only the integer frequency values are available in the list by default. Uncheck the checkbox window to show the complete list of all manufactured and user-defined frequencies from the internal database. The user can add a new oscillator frequency by clicking the Add New Osc button. The user-defined oscillator frequency will be stored in the myosc.cfg file in the application directory. The newly-added oscillator frequency will appear at the end of the Oscillator list after restarting the calculator application. Depending on the selected MCU family, only the valid and suitable oscillator frequencies are shown in the Oscillator listbox window. See Table 3 for more details.

NOTE

If the S12P (S12HY) MCU family is selected and 1.000 MHz internal reference clock (IRC) is picked, f_{REF} is fixed to f_{IRC1M}.

$$f_{REF} = f_{IRC1M} = 1.000 MHz \qquad \textit{Eqn. 10}$$

3.1.3 Bus Frequency

The required MCU-bus frequency is selected from the Bus Freq listbox window.

Depending on the selected MCU family, only the bus frequencies from the allowed bus ranges are shown in the Oscillator listbox window. See Table 3 for more details.

3.1.4 Output Results

If any bus frequency is picked, the correct register settings are calculated by the application, allowed f_{VCO} and f_{REF} ranges are considered by the application. The SYNR, REFDV, and POSTDIV register values are displayed in hexadecimal format in the appropriate boxes.

The VCO frequency, reference frequency, and maximum PLL lock time are calculated and displayed in the Statistic window.

The FDIV (FCLKDIV) register value is important for proper internal flash state-machine timing, in case the internal flash or emulated EEPROM is used by the embedded application. Correct FDIV is chosen from the appropriate lookup table, depending either on the selected oscillator frequency (S12XE, XF, XS), or the internal bus frequency (S12P, S12HY). Correct flash clock frequency range is checked.

The PLL locking time is defined as follows:

Eqn. 11

$$t_{lock}[\mu s] = 150 + \frac{256}{f_{REF}[MHz]}$$

4 PLL Software Initialization Examples

4.1 S12XE PLL Init Examples

4.1.1 Basic S12XE PLL Init

```
//---------------------------------------------------------------------------
// **** S12XE PLL_init example ****
//---------------------------------------------------------------------------
void PLL_init(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  PLLCTL = 0B00000001; // CME=0,PLLON=0,FM1=0,FM2=0,FSTWKP=0,PRE=0,PCE=0,SCME=1
  CLKSEL = 0B00000011; // PLLSEL=0,PSTP=0,PLLWAI=0,RTIWAI=1,COPWAI=1
  SYNR = synr;         // Set the multiplier register
  REFDV = refdv;       // Set the divider register
  POSTDIV = postdiv;   // Set the post divider register
  PLLCTL_PLLON = 1;    // Enable the Phase Lock Loop
  while(!CRGFLG_LOCK); // Wait till the PLL VCO is within tolerance
  CLKSEL_PLLSEL = 1;   // Select clock source from PLLCLK
  //ECLKCTL_NECLK=0;     // Enable the BusClk output at ECLK pin
}
//---------------------------------------------------------------------------
```

4.1.2 S12XE PLL Init with Timeout and Status Checking

```
//-----------------------------------------------------------------------------
// **** S12XE PLL_init example with status checking and timeout ****
// return 0 - OK
//        1 - Error - PLL not locked
//-----------------------------------------------------------------------------
unsigned char PLL_init(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  unsigned int timeout=0xffff; // aux. var. to make small SW delay
  PLLCTL = 0B00000001; // CME=0,PLLON=0,FM1=0,FM2=0,FSTWKP=0,PRE=0,PCE=0,SCME=1
  CLKSEL = 0B00000011; // PLLSEL=0,PSTP=0,PLLWAI=0,RTIWAI=1,COPWAI=1
  SYNR = synr;         // Set the multiplier register
  REFDV = refdv;       // Set the divider register
  POSTDIV = postdiv;   // Set the post divider register
  PLLCTL_PLLON = 1;    // Enable the Phase Lock Loop
  // Wait till the PLL VCO is within tolerance
  while((!CRGFLG_LOCK)&&( timeout-- != 0));
  if(timeout == 0)     // PLL didn't lock for some reason
    return(1);         // return error
  CRGFLG = 0x10;       // Ensure clearing of LOCKIF flag
  CRGINT_LOCKIE = 1;   // Enable PLL lock interrupt - to know if it loses clock
  CLKSEL_PLLSEL = 1;   // Select clock source from PLLCLK
  if(CLKSEL_PLLSEL != 1) // will only be set if the PLL was still locked
    return(1);         // return error if loss of lock
  return(0);           // else return OK
}
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// PLL_LOCK_ISR
// Triggered when PLL lock status changed (locked / unlocked)
//-----------------------------------------------------------------------------
#pragma CODE_SEG NON_BANKED
interrupt 28 void PLL_LOCK_ISR(void)
{
  CRGFLG = 0x10;       // Clear LOCKIF flag
  if(CRGFLG_LOCK == 0)
  {
    //do something here
  }
  else
  {
    //do something here
  }
}
#pragma CODE_SEG DEFAULT
//-----------------------------------------------------------------------------
```

4.2 S12P PLL Init Examples

4.2.1 S12P PLL Init in PEI Mode

```
//---------------------------------------------------------------------------
// **** S12P PLL_init in PEI mode ****
// - PLL Engaged, Internal Reference Clock used
//---------------------------------------------------------------------------
void PLL_init_PEI(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  CPMUSYNR = synr;      // Set the multiplier register
  CPMUREFDIV = refdv;   // Set the ref. divider register
  CPMUPOSTDIV = postdiv;// Set the post divider register
  while(!CPMUFLG_LOCK); // Wait till the PLL VCO is within tolerance (PLL locked)
  //now the PLL has been locked and fpll = fvco / (POSTDIV + 1)
  //ECLKCTL_NECLK=0;    // Enable the BusClk output at ECLK pin
}
//---------------------------------------------------------------------------
```

4.2.2 S12P Basic Init in PEE Mode

```
//---------------------------------------------------------------------------
// **** S12P PLL_init in PEE mode ****
// - PLL Engaged, External Reference Clock used
//---------------------------------------------------------------------------
void PLL_init_PEE(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  CPMUSYNR = synr;      // Set the multiplier register
  CPMUREFDIV = refdv;   // Set the ref. divider register
  CPMUPOSTDIV = postdiv;// Set the post divider register

  //if(OSCE = 0) then fref = fIRC1M
  //if(OSCE = 1) then fref = fosc / (REFDIV + 1)
  CPMUOSC_OSCE = 1; //enable external oscillator OSCE

  //Wait for the UPOSC bit to be set, indicating the oscillator start up
  while(CPMUFLG_UPOSC == 0);

  while(!CPMUFLG_LOCK); // Wait till the PLL VCO is within tolerance (PLL locked)
  //now the PLL has been locked and fpll = fvco / (POSTDIV + 1)
  //ECLKCTL_NECLK=0;    // Enable the BusClk output at ECLK pin
}
//---------------------------------------------------------------------------
```

4.2.3 S12P Init in PEE Mode with Timeout & Status Checking

```
//-----------------------------------------------------------------------
// **** S12P PLL_init in PEE mode with status checking and timeout ****
// - PLL Engaged, External Reference Clock used
// return 0 - OK
//        1 - Error - PLL not locked, oscillator start up error
//-----------------------------------------------------------------------
unsigned char PLL_init_PEE(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  unsigned int timeout=0xffff; // aux. var. to make small SW delay
  unsigned char i;       // aux. var.
  CPMUSYNR = synr;       // Set the multiplier register
  CPMUREFDIV = refdv;    // Set the ref. divider register
  CPMUPOSTDIV = postdiv;// Set the post divider register

  //if external oscillator is disabled (OSCE = 0) then fref = fIRC1M
  //if external oscillator is enabled (OSCE = 1) then fref = fosc / (REFDIV + 1)
  CPMUOSC_OSCE = 1; //enable external oscillator OSCE

  //Wait for the UPOSC bit to be set, indicating the oscillator start up
  while((!CPMUFLG_UPOSC)&&(timeout-- != 0)){
    for(i=0;i<20;i++){asm nop;}
  }// total timeout delay > ~ 100ms
  if(timeout == 0)
    return(1);           // Oscillator doesn't started properly, return error

  // Wait till the PLL VCO is within tolerance
  timeout=0xffff;
  while((!CPMUFLG_LOCK)&&( timeout-- != 0));
  if(timeout == 0)       // PLL didn't lock for some reason
    return(1);           // return error
  CPMUFLG = 0x10;        // Ensure clearing of LOCKIF flag
  CPMUINT_LOCKIE = 1;    // Enable PLL lock interrupt - to know if it loses clock
  CPMUCLKS_PLLSEL = 1;   // Select clock source from PLLCLK
  if(CPMUCLKS_PLLSEL != 1) // will only be set if the PLL was still locked
    return(1);           // return error if loss of lock
  return(0);             // else return OK
  //now the PLL has been locked and fpll = fvco / (POSTDIV + 1)
  //ECLKCTL_NECLK=0;     // Enable the BusClk output at ECLK pin
}
//-----------------------------------------------------------------------
```

4.2.3.1 PLL Lock Interrupt Service Routine

```
//-----------------------------------------------------------------------------
// PLL_LOCK_ISR
// Triggered when PLL lock status changed (locked / unlocked)
//-----------------------------------------------------------------------------
#pragma CODE_SEG NON_BANKED
interrupt 28 void PLL_LOCK_ISR(void)
{
  CPMUFLG = 0x10;       // Clear LOCKIF flag
  if(CPMUFLG_LOCK == 0)
  {
    //do something here
  }
  else
  {
    //do something here
  }
}
#pragma CODE_SEG DEFAULT
//-----------------------------------------------------------------------------
```

4.2.4 S12P Init in PBE Mode

First of all, we have to set a valid PEE mode, and then we can switch to PBE mode.

To enter the PBE mode from PEI (PEE) mode take the following steps:

1. Make sure the PLL configuration is valid: Program the reference divider (REFDIV[3:0] bits) to divide down the oscillator frequency if necessary.
2. Enable the external oscillator (OSCE bit).
3. Wait for the oscillator to start up (UPOSC = 1).
4. Select the oscillator clock as bus clock (PLLSEL = 0).

```
//-----------------------------------------------------------------------------
// **** S12P CLK_init in PBE mode ****
// - PLL Bypassed, External Oscillator Clock signal used
//-----------------------------------------------------------------------------
void CLK_init_PBE(unsigned char synr, unsigned char refdv, unsigned char postdiv)
{
  //CPMUSYNR = synr;      // Set the multiplier register - optional
  //CPMUREFDIV = refdv;   // Set the ref. divider register - optional
  //CPMUPOSTDIV = postdiv;// Set the post divider register - optional

  //if (OSCE = 0) then fref = fIRC1M
  //if (OSCE = 1) then fref = fosc / (REFDIV + 1)
  CPMUOSC_OSCE = 1; //enable external oscillator OSCE

  //Wait for the UPOSC bit to set, indicating the oscillator start up
  while(CPMUFLG_UPOSC == 0);

  //Set the PLLSEL to 0. System clock derived from OSC now.(fbus = fosc/2)
  CPMUCLKS_PLLSEL = 0;
  //ECLKCTL_NECLK=0;    // Enable the BusClk output at ECLK pin
}
//-----------------------------------------------------------------------------
```

4.2.5 Important Notes for S12P PLL Init

Writing into the CPMUSYNR or CPMUREFDIV will unlock the PLL.While PLL is unlocked, $f_{pll} = f_{vco}/4$ to protect the system from high-core clock frequencies during the PLL stabilization time. If PLL is locked, then $f_{pll} = f_{vco} / (POSTDIV + 1)$.

Since the adaptive spike filter uses the VCOCLK (from PLL) to continuously qualify the external oscillator clock, losing the PLL lock status (LOCK = 0) means losing the oscillator status information as well (UPOSC = 0).

The impact of losing the oscillator status in PBE mode is as follows:

- The MSCAN module, which can be configured to run on the oscillator clock, may need to be reconfigured.
- The PLLSEL is set automatically and the bus clock is switched back to the PLL clock.
- Application software needs to be prepared to deal with the impact of losing the oscillator status at any time.

If external oscillator is disabled (OSCE = 0), then $f_{ref} = f_{IRC1M}$.

If external oscillator is enabled (OSCE = 1), then $f_{ref} = f_{osc} / (REFDIV + 1)$.

5 References

- *MC9S12XEP100 Reference Manual, MC9S12XEP100RMV1.pdf*
- *MC9S12XS256 Reference Manual, MC9S12XS256RMV1.pdf*
- *MC9S12P Reference Manual, MC9S12P128.pdf*
- *Comparsion of the S12XS CRG Module with S12P CPMU Module, AN3622.pdf*

6 Glossary

Table 4. Glossary Table

Term	Definition
Resonator	Common term used for the external ceramic resonator, quartz crystal, or active oscillator clock (canned oscillator).
Oscillator	External or internal reference clock source, used by MCU.
External oscillator	external clock reference source
f_{OSC}	frequency of oscillator
f_{BUS}	frequency of internal MCU bus clock
f_{VCO}	frequency of PLL voltage-controlled oscillator
f_{REF}	PLL reference frequency
PEI	PLL engaged — Internal 1 MHz reference clock is used by PLL to derive MCU bus clock (S12P, S12HY only).
PEE	PLL engaged — External reference clock is used by PLL to derive MCU bus clock (S12P, S12HY only).
PBE	PLL bypassed — External reference clock is directly used for MCU timing.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number:
Rev. 1
08/2009

freescale™
semiconductor