Application Note

AN2552/D
10/2003

Configuring the System and Peripheral Clocks in the MC9S12E128

By Steven Torres
8/16 Bit Systems Engineering
Austin, Texas

Introduction

The MC9S12E128 is a 16-bit FLASH-based microcontroller unit (MCU) that is designed for the low-cost general-purpose distribution market. It is the first in a series of new low-cost general-purpose products to address the UPS (uninterruptible power supply) and home appliance market. The MC9S12E128 has a rich set of modules including:

- CRG — Clocks and reset generator
- FLASH — 128 Kbytes FLASH
- SCI — Three asynchronous serial communication interface modules, all of which have infrared capability
- SPI — Synchronous serial peripheral interface
- IIC — Inter-IC bus
- PMF — 6-channel pulse-width modulator with fault protection and three 15-bit counters
- PWM — 6-channel pulse-width modulator with 8-bit counters or 3-channel pulse width modulator with 16-bit counters
- TIM — Three 4-channel timers with 16-bit counters
- ATD — 16-channel analog-to-digital converter with 10-bit resolution
- DAC — Two 1-channel digital-to-analog converters with 8-bit resolution

This application note details the MC9S12E128 CRG, which provides the internal clock signals for the HCS12 core and all peripheral modules. The discussion also reviews how the CRG output clocks interface with the rest of the MC9S12E128 modules/peripherals and how the clock source affects each module's clocking capability. This discussion describes in detail the systems clock's relationship to the configuration, initialization, and usage of the MCU peripherals.

This product incorporates SuperFlash® technology licensed from SST.

MC9S12E128 Clock Setup Helper

This application note has a companion Microsoft® Excel workbook, *MC9S12E128 Clock Usage Workbook,* available from the Freescale website (Freescale document number AN2552SW.zip). It contains worksheets that can be used to help configure the CRG, IIC, TPM, SCI, SPI, RTI, ATD, FLASH, BDM, PWM, and PMF.

The *MC9S12E128 Clock Usage Workbook* requires the user to enter clock/oscillator values and provides the user with clocking information for each module. The user can then select the clock rate, baud rate, or period value that is need for each module. The workbook displays the results of the selections in a summarized worksheet along with the register configuration values for each module. (A help sheet is provided.)

The registers values that are provided per module by the worksheet are:

- CRG — SYNR[5:0] and REFDV[3:0]
- FLASH — PRDIV8 and FDIV[5:0]
- SCI — SBR[12:0]
- SPI — SPPR[2:0] and SPR[2:0]
- IIC — IBC[2:0], IBC[5:3], and IBC[6:7]
- PMF —PWMPERx, PCKx[2:0], and PWMSCLx
- PWM — PMFMOD, PRSCx[1:0], and PMFDTM
- TIM — PR[2:0]
- ADC — PRS[4:0]
- RTI — RTR[3:0] and RTR[6:4]
- COP — CR[2:0]

NOTE: *With the exception of mask set errata documents, if any other Freescale document contains information that conflicts with the information in the device guide, the device guide should be considered to have the most current and correct data.*

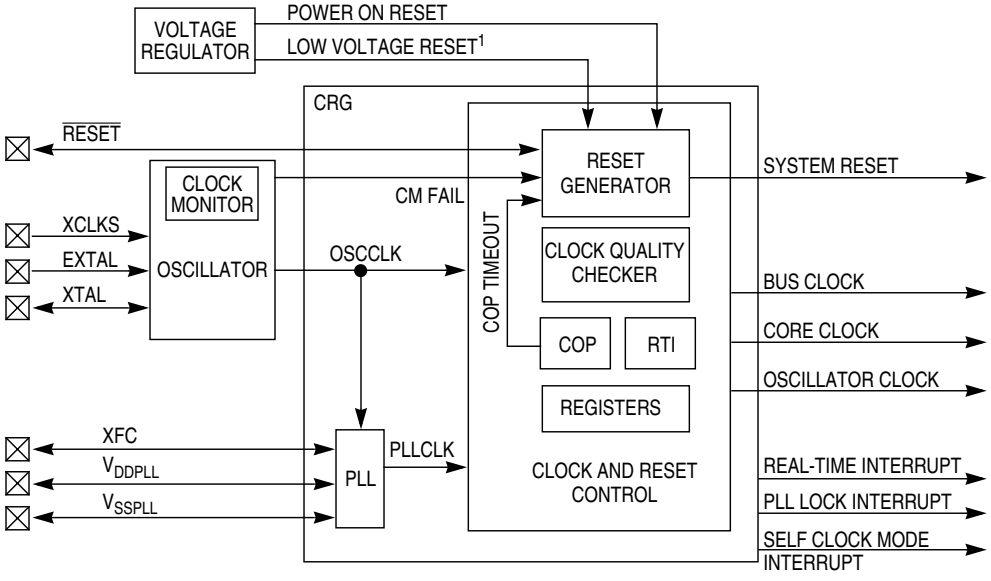
Microsoft® is a registered trademark of Microsoft Corporation in the U.S. and other countries.

CRG MCU System Blocks and Signals

The CRG module provides the system clock to the MC9S12E128 MCU. The CRG provides multiple selections for system clock sources, allowing the user to configure a system based on system cost, precision, and performance requirements.

This section addresses the features, behavior, and usage of the CRG module and describes its various modes of operation. This application note also describes the relationship between the oscillator (OSC) module and the CRG. **Figure 1** shows a functional view of the CRG module and highlights the CRG's internal and external interfaces/signals.

In the following paragraphs, a brief description of each of the CRG's system blocks and interfaces is provided. This application note focuses only on the parts of the CRG that control and impact the configuration, initialization, and usage of the MCU clocks. Since the discussion is primarily on clocks, the parts of the CRG that don't contribute to clocking, such as features that control reset and internal CRG interrupts, will not be described.



1 REFER TO DEVICE SPECIFICATION FOR AVAILABILITY OF THE LOW VOLTAGE RESET FEATURE.

Figure 1. System Clocks Block Diagram

CRG Module Features and Blocks

The CRG module consists of two major functional blocks:

- Clock and reset control block — This block (illustrated in **Figure 1**) consists of several subblocks including the system clock generator (SCG), computer operating properly (COP), system reset generator, and real-time interrupt (RTI). The clock and reset control block subblocks, especially the SCG, are key to the CRG's overall operation.
- Phase-locked loop (PLL) block — The PLL is used for frequency control. The PLL can be configured as a frequency multiplier to run the MCU from a different timebase than that of the incoming OSCCLK signal from the OSC module. **Figure 2** illustrates the internal components of the PLL.

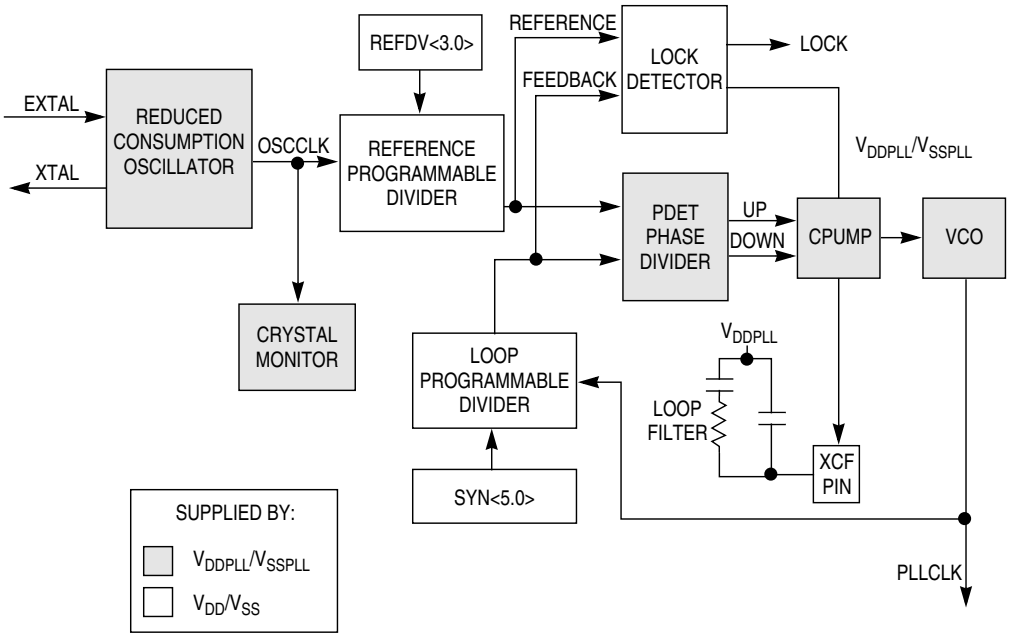


Figure 2. PLL Functional Diagram

System Clock Generator (SCG)

The system clock generator (SCG) is the primary component of the clock and reset control block that contributes to the clocking behavior of the CRG. The SCG subblock consists of several features including:

- Clock quality checker — Monitors the OSC module clock signal to determine whether the signal is lost or unusable.
- Clock PLL switch — Gates the MCU system clock to be sourced from either the CRG PLL block or OSC module. The user must configure this switch manually.
- CRG user registers — Configure, enable, and disable the CRG module clock behavior.

Figure 3 shows the how the clock signals are routed and gated through this part of the CRG module:

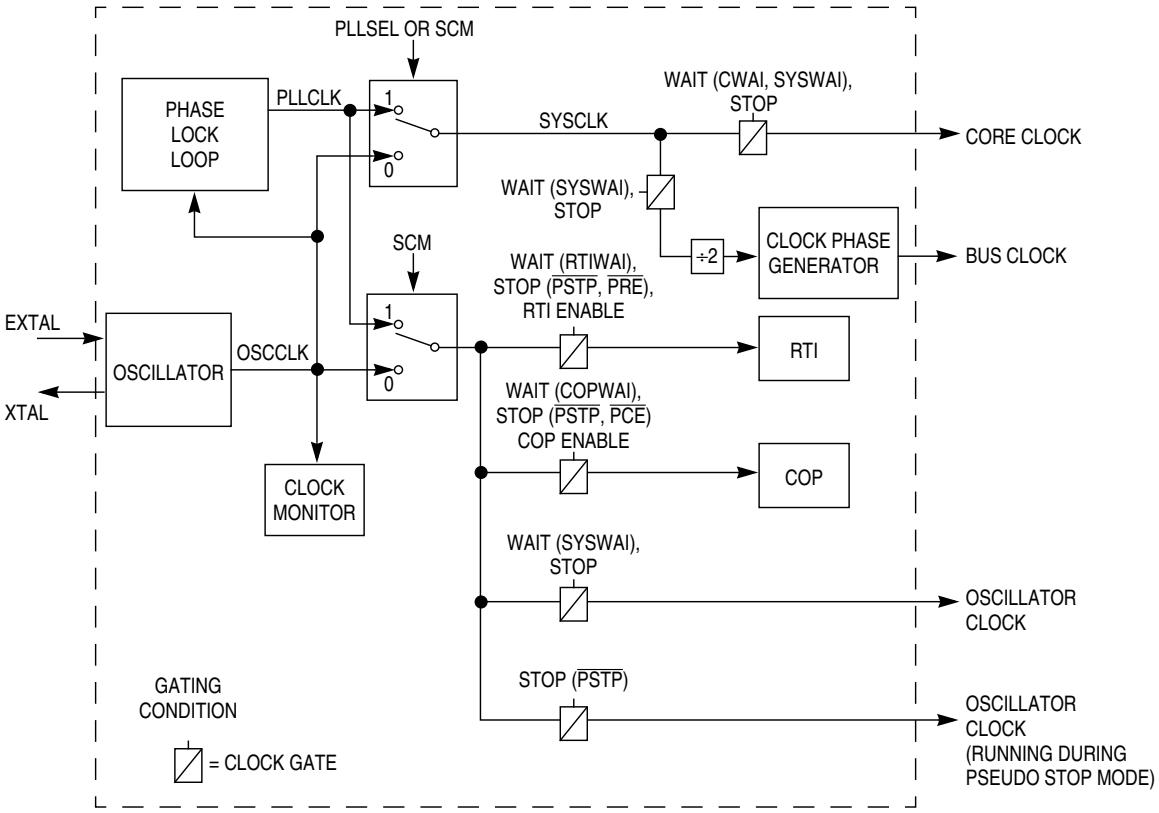


Figure 3. SCG Subblock Functional Block Diagram

PLL Block CRG Signals

The PLL has several internal and external interfaces as shown in Figure 1:

- XFC — Connects off chip. It requires a passive, second order external loop filter to eliminate the VCO input ripple. XFC is connected to V_{DDPLL} if not used.
- V_{DDPLL}, V_{SSPLL} — Connects off chip. V_{DDPLL} provides operating voltage while V_{SSPLL} provides a ground for a PLL circuitry external bypass.
- OSCCLK — Comes from the OSC module. It is the raw clock signal that originates from the external clock circuitry or a crystal oscillator amplifier.
- PLLCLK — An output of the PLL block. If the PLL is engaged, the PLLCLK signal is a function of the OSCCLK and the PLL frequency multipliers.

System Clock Generator (SCG) Signals

The SCG signals are itemized below. The name of each signal appears as shown in Figure 1.

- CM fail — Comes from the OSC module. Asserts when the clock monitor block of the OSC module detects a loss of clock source.
- OSCCLK — Comes from the OSC module. It is the raw clock signal that originates from the external clock circuitry or a crystal oscillator amplifier.
- PLLCLK — An output of the PLL block. If the PLL is engaged, the PLLCLK signal is a function of the OSCCLK and the PLL frequency multipliers.
- Bus clock — An output of the SCG block. The bus clock is equal to the main CPU clock divided by 2. The main CPU can be derived from the OSCCLK or PLLCLK depending on the clock switch, which is configured with the CRG registers.
- Core clock — An output of the SCG block. This is the frequency of operation of the CPU. It is derived from the OSCCLK or PLLCLK, depending on the clock switch.
- Oscillator clock — An output of the SCG block. Frequency is equal to OSCCLK.

Oscillator (OSC) Module Signals

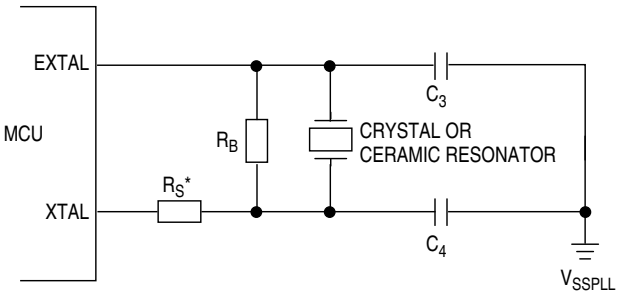
The OSC module is an integral component required for the operation of the CRG. The OSC provides a raw clock signal from which MCU clocking is derived. The primary OSC signal feed into the CGR is the OSCCLK signal. This signal is fed to both the PLL and the clock and reset control block (system clock generator subblock). The OSCCLK signal itself is based on the OSC module off-chip connections. The OSC module off-chip connections include:

- XCLKS — Connects off chip. It is an input signal which should be based on the type of oscillator or external clock circuitry used. XCLKS is pulled high when a Pierce oscillator or any type of external clock circuitry is used. XCLKS is pulled low when a Colpitts oscillator is used.

NOTE: *The configuration of the XCLKS may vary depending on the device. See the appropriate device guide for device-specific information.*

- EXTAL — Connects off chip. It is either an external clock or crystal oscillator amplifier input.
- XTAL — Connects off chip. It is the output of a crystal oscillator amplifier. When external clock circuitry is used, XTAL is not connected.

Figure 6, Figure 4, and Figure 5 illustrate the circuit recommendation for a Colpitts and Pierce oscillator and external clock circuitry.



*R_S CAN BE ZERO (SHORTED) WHEN USED WITH HIGHER FREQUENCY CRYSTALS.
REFER TO CRYSTAL MANUFACTURERS' DATA SHEET.

Figure 4. Pierce Oscillator

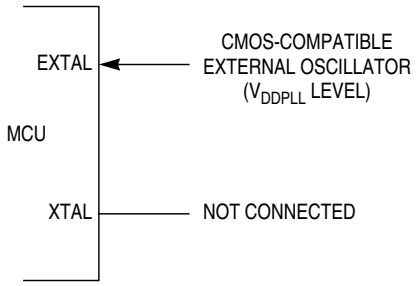
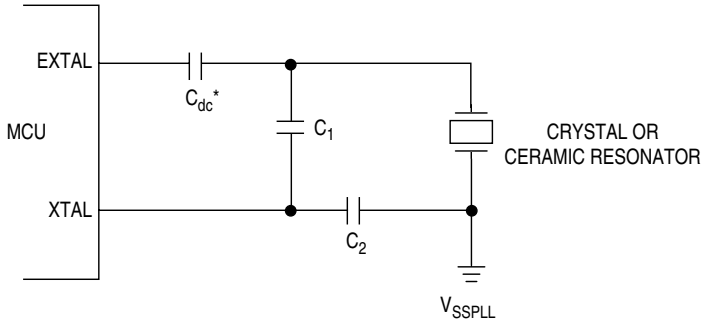


Figure 5. External Clock



*DUE TO THE NATURE OF TRANSLATED GROUND COLPITTS OSCILLATOR,
A dc VOLTAGE BIAS IS APPLIED TO THE CRYSTAL.
PLEASE CONTACT THE CRYSTAL MANUFACTURER FOR CYSTAL dc BIAS
CONDITIONS AND RECOMMENDED CAPACITOR VALUE C_{dc}

*COLPITTS CIRCUITS ARE NOT SUITABLE FOR OVERTONE RESONATORS AND CRYSTALS.

Figure 6. Colpitts Oscillator

CRG Registers Set

The CRG module provides a set of registers that allows the user to control the operation and behavior of the MCU in its various configurations and modes. Table 1 shows the complete register map of the CRG module.

Table 1. CRG Module Register Map

Address Offset	Use	Access
$_00	CRG synthesizer register (SYNR)	R/W
$_01	CRG reference divider register (REFDV)	R/W
$_02	Reserved	R/W
$_03	CRG flags register (CRGFLG)	R/W
$_04	CRG interrupt enable register (CRGINT)	R/W
$_05	CRG clock select register (CLKSEL)	R/W
$_06	CRG PLL control register (PLLCTL)	R/W
$_07	CRG RTI control register (RTICTL)	R/W
$_08	CRG COP control register (COPCTL)	R/W
$_09	Reserved	R/W
$_0A	Reserved	R/W
$_0B	CRG COP arm/timer reset (ARMCOP)	R/W

Because this application note addresses only the CRG features that impact clocking, only these relevant registers will be discussed in detail:

- SYNR register — Controls the multiplication factor of the PLL. (See **Configuring the PLLCLK Signal**.)
- REFDV register — Provides a finer granularity for the PLL multiplier steps. (See **Configuring the PLLCLK Signal**.)
- CLKSEL register — Configures and controls the clock behavior of the stop and wait modes of the MCU. The PLLSEL bit in this register controls whether the system clocks are derived from the PLLCLK or OSCCLK signal.
- PLLCTL register — Controls PLL functionality and other CRG functions. The PLLON bit in this register turns on the PLL circuitry.

Software Configuration PLL

This section will discuss the initialization of the PLL. Recall that the PLL can be configured as a frequency multiplier to run the MCU from a different timebase than that of the incoming OSCCLK signal.

To select the timebase from which the system clock (SYSCLK) will be derived, the PLLSEL bit of the CLKSEL register must be configured. SYSCLK can either be the OSCCLK signal coming from the OSC module or the PLLCKLK signal originating from the CRG PLL block. SYSCLK then becomes the source from which both the core and the bus clocks are computed. *Equation 1* and *Equation 2* show the relationship between the SYSCLK signal and the core and the bus clocks.

Equation 1:

$$f_{Core} = f_{SYSCLK}$$

Equation 2:

$$f_{Bus} = \frac{f_{SYSCLK}}{2}$$

The PLLSEL bit configures the CRG clock PLL switch. If PLLSEL bit is set, then the system clocks are derived from PLLCLK, but if the PLLSEL bit is cleared, SYSCLK is derived from OSCCLK. So, depending on the state of the PLLSEL bit, the bus clock can be calculated two different ways:

Equation 3: (In Equation 3, f_{PLL} is the frequency of the PLLCLK signal.)

if PLLSEL = 1, $f_{Core} = f_{PLL}$ and $f_{Bus} = \frac{f_{PLL}}{2}$

if PLLSEL = 0, $f_{Core} = f_{OSCCLK}$ and $f_{Bus} = \frac{f_{OSCCLK}}{2}$

f_{PLL} can be calculated from *Equation 4*.

Configuring the PLLCLK Signal

The value of PLLCLK signal is configured using these two CRG registers:

- CRG synthesizer register (SYNR) — 6-bit value that controls the multiplication factor of the PLL. If PLLSEL is asserted, the OSCCLK signal is multiplied by 2 × (SYNR + 1).
- CRG reference divider register (REFDV) — 4-bit value that provides a finer granularity for the PLL multiplier steps. If PLLSEL is asserted, the OSCCLK signal is divided by (REFDV + 1).

The SYNR and REFDV values together modify the incoming signal into the PLL block, OSCCLK. The combined contribution of these registers on the value of PLLCLK is shown in *Equation 4:*

$$f_{PLL} = \left[\frac{2 \times (SYNR + 1)}{REFDV + 1}\right] \times OSCCLK$$

Modes of the CRG

The CRG has five modes of operation:

- Run mode
- Wait mode
- Full stop mode
- Pseudo stop mode
- Self clock mode

Run Mode

Run mode is the normal operation mode for the MCU. All clocks are running and the MCU is fully functional when in run mode.

Wait Mode

Wait mode is a low-power mode of operation. It is entered when a WAI instruction is executed.

If wait mode is entered from normal operation while the PLL is enabled, wait mode will disable the PLL by deasserting the PLLSEL bit and the MCU will switch to OSCCLK clock source. When wait mode is exited, the software must manually re-enable the PLL by setting the PLLSEL bit (if PLL is required).

Depending on the configuration of the CLKSEL register, this mode can disable the system and core clocks. **Table 2** shows selected CLKSEL register bits that are relevant in wait mode and how these bits affect the MCU's configuration.

Table 2. CRG Clock Select Register (CLKSEL) Bits

	PLLWAI	CWAI	SYSWAI	RTIWAI	COPWAI	ROAWAI
PLL	Stopped	—	—	—	—	—
Core	—	Stopped	Stopped	—	—	—
System	—	—	Stopped	—	—	—
RTI	—	—	—	Stopped	—	—
COP	—	—	—	—	Stopped	—
Oscillator	—	—	—	—	—	Reduced(1)

1. Refer to the oscillator block user guide for availability of reduced oscillator amplitude.

There are several different methods to wake up and restart the MCU from wait mode:

- External reset — Resets the MCU to its reset state.
- Clock monitor reset — If the clock monitor is enabled (PLLCTL[CME = 1]), the MCU can leave wait mode when a loss of the clock is detected, which is indicated by an asserted clock monitor fail signal. How wait mode is exited depends on the state of the SCME bit. Recall that the SCME is the bit in the PLLCTL register that configures the MCU to either cause a clock monitor reset or forces the MCU into self-clock mode (when a crystal clock failure is detected).

  During a clock failure, if the SCME bit is asserted and the SCM interrupt is enabled (CRGINT[SCMIE =1]), the CRG generates an SCM interrupt, starts the clock quality checker, and continues with normal operation. If the SCME bit is not asserted and a loss of the clock is detected, a clock monitor fail reset is generated, which is similar an external reset.
- COP reset — If a COP time-out period ends without an update to the ARMCOP register in wait mode the CRG will generate a reset.
- SCM interrupt — If an SCM interrupt occurs in wait mode, wait mode will be exited and the MCU will continue in normal operation only if the SCME bit is asserted and the SCM interrupt is enabled (CTGINT[SCMIE = 1]). If the SCM interrupt is blocked (SCMIE = 0), the MCU will not wake up.
- Real-time interrupt (RTI) — If an RTI occurs in wait mode, the MCU exits from wait mode and starts normal operation. (Other interrupts will also exit the MCU from wait mode.)

Stop Modes

Stop mode is a low-power mode of operation that is entered when a stop instruction is executed.

Depending on the settings of the PCE and PRE bits in the CRG PLLCTL register and the PSTP bit in the CKLSEL register, all clocks can be stopped in stop mode. The list below provides some configuration details with the PCE, PRE, and PSTP bits.

- Pseudo stop bit (PSTP) — If this bit is set, the MCU enters pseudo stop mode when a STOP instruction is executed. In pseudo stop mode, the oscillator continues to run with a reduced amplitude.
- RTI enable during pseudo stop mode bit (PRE) — The PRE bit is set to allow the RTI to run in pseudo stop mode.
- COP enable during pseudo stop mode bit (PCE) — The PCE bit is set to allow the COP to run in pseudo stop mode.

If stop mode is entered from normal operation and the PLL is enabled, stop mode will disable the PLL by deasserting the PLLSEL bit in the CLKSEL register. With PLLSEL deasserted, the MCU switches to the OSCCLK clock source. When stop mode is exited in this case, the software must manually re-enable the PLL by setting the PLLSEL bit.

Pseudo Stop Mode

Pseudo stop mode is similar to wait mode, but in pseudo stop mode, the CRG requests any module, if capable, enter a power save mode. Pseudo stop mode is entered if the PSTP bit of the CLKSEL register is asserted. In this mode, the oscillator continues to run, but most of the system and core clocks are disabled. The methods to wake up from pseudo stop mode are the same as if waking up from wait mode (see **Wait Mode**).

If pseudo stop mode is entered from SCM, the CRG continues to check the clock quality signal until the clock check is successful and the PLL and the VREG remain enabled.

Full Stop Mode

If the PSTP bit is clear, the MCU enters full stop mode when a STOP instruction is executed and the oscillator is disabled. In this mode, the oscillator is disabled, which disables all system and core clocks. To wake up from full stop mode, the MCU requires:

- An external reset — Restores the MCU to its reset state.
- An external interrupt — Restarts all clocks. The CRG then continues to run in normal operation as long as the clock quality check is successful. If the clock quality check fails, then the CRG enters SCM. External Interrupts are facilitated by the $\overline{\text{XIRQ}}$ and $\overline{\text{IRQ}}$ pins.

If full stop mode is entered from SCM, the clock quality check is stopped.

Self Clock Mode

Self clock mode (SCM) is entered if both the CME and SCME bits of the PLLCTL register are asserted (CME = 1 and SCME = 1) and the clock monitor in the oscillator block detects a loss of clock. SCM remains active until the clock quality check detects a valid clock again. SCM can be entered due to long crystal start-up times.

In SCM, the bus and core clocks are derived from the VCO minimum operation frequency, f_{SCM}. The value of f_{SCM} can range from 2 MHz to 5 MHz depending on silicon process parametrics. An average value for f_{SCM} is 3 MHz. *Equation 5* illustrates the clocking relationships with f_{SCM}.

Equation 5:

$$f_{Core} = f_{SCM}$$

$$f_{osc} = f_{SCM}$$

$$f_{Bus} = \frac{f_{SCM}}{2}$$

If the MCU was clocked with the PLL prior to entering SCM, the PLLSEL bit is automatically cleared.

Software Configurations for Other Module Clocks

Figure 7 shows the dependence of the peripherals, core, and memory on the CRG clock outputs including the core, bus (IP_{Bus}), and oscillator clocks.

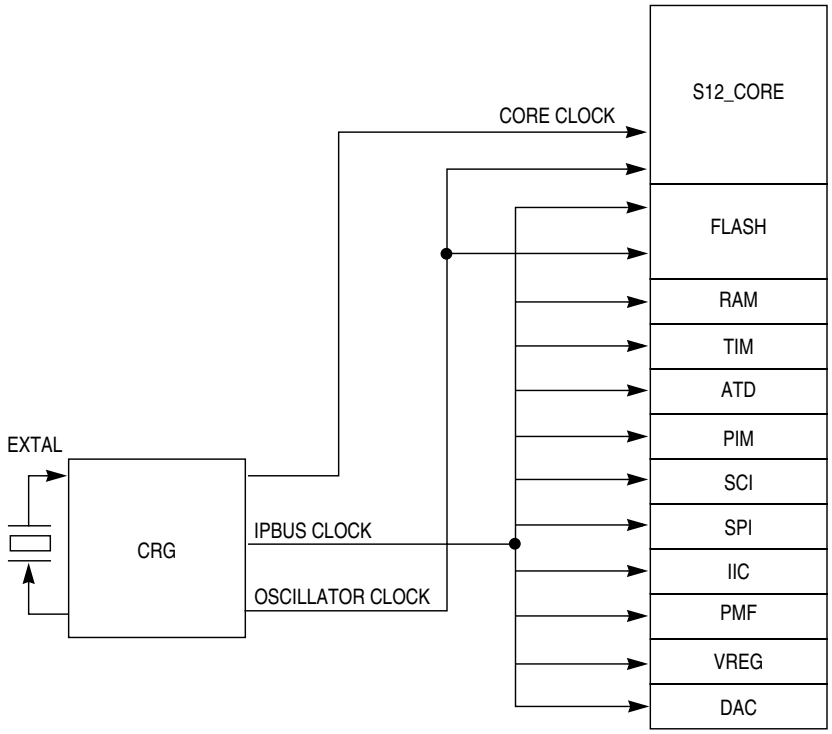


Figure 7. CRG Clocking Tree

The modules shown in Figure 7 are discussed in this section. A discussion of how each module uses the clock signals from the CRG is provided.

FLASH Clock

The oscillator clock is the input for the FLASH module. To perform FLASH program or erase operations, the internal FLASH clock frequency ($f_{FLASHCLK}$) must be configured to run between 150 kHz and 200 kHz.

The FLASH uses the PRDIV8 and FDIV[5:0] bits in the FCLKDIV register to divide down the oscillator clock to the required clock range. PRDIV8 is a 1-bit prescaler value that, if set, divides the oscillator clock by 8. If PRDIV8 is clear, the oscillator clock is directly fed into the FCLKDIV divider. The FCLKDIV

divider includes a 6-bit value (bits FDIV[5:0]) that generates a divisor from 1 to 64. The divisor is equal to (FDIV[5:0]+1). *Equation 6* is the resulting equation for the FLASH clock frequency (f_{FCLK}).

Equation 6:

$$f_{FLASHCLK} = \frac{f_{Bus}}{(8^{PRDIV8} \times (FDIV[5:0]+1))}$$

SCI Clock Baud Rate BR_{SCI}

The bus clock is the input for SCI modules. Three asynchronous SCIs are available on the E128. The SCI module version 3.x can be configured to operate in compliance with the IrDA SIR (IrDA) specification.

The SCI uses the SCIxBDH and SCIxBDL. These registers together provide a 13-bit field for SCI baud rate (BR_{SCI}) configuration. SBR[12:0] is used to represent the 13-bit SCI baud rate register value. SBR[12:0] can be set to any value from 0 to 8191. When SBR[12:0] = 0, the SCI baud rate generator is disabled, which reduces system current consumption. For all other SCI baud rate calculations, use *Equation 7*.

Equation 7:

$$\text{If IREN} = 0,\ BR_{SCI} = \frac{f_{Bus}}{(16 \times SBR[12:0])}$$

$$\text{If IREN} = 1,\ BR_{SCIIR} = \frac{f_{Bus}}{(32 \times SBR[12:1])}$$

The E128 SCI also can be operated in IrDA mode. In this mode, the SCI can modulate and demodulate narrow pulse-widths as defined by the IrDA SIR standard. IrDA mode is enabled by setting the IREN bit of the SCIBDH register. *Equation 7* shows that the BR_{SCIIR} calculation depends on the IREN bit.

SPI Clock Baud Rate

The synchronous serial peripheral interface (SPI) allows a duplex, synchronous serial communication between the MCU and peripheral devices. The only clock source for the SPI module is the bus clock.

SPI baud rate (BR_{SPI}) can be set using the SPI control register 2 (SPCR2). The SPCR2 register has two 3-bit values that contribute to an SPI baud rate divisor, SPPR[2:0] and SPR[2:0]. The first 3-bit value, identified by the SPPR[2:0] bits, can have a value ranging from 1 to 8. The second 3-bit value, identified by the SPR[2:0] bits, can have a value ranging from 2 to 256. The mathematical expression of the SPI baud rate divisor is $[(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}]$. Applying this expression to the bus clock yields a formula for the SPI baud rate:

Equation 8:

$$BR_{SPI} = \frac{f_{Bus}}{[(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}]}$$

IIC Clock

The IIC bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between two devices. The only clock source for the IIC module is the bus clock. The device is designed to operate at speeds up to 100 kbps.

The IIC baud rate is a function of the bus clock divided by the SCL divider. *Equation 9* shows how to generate the SCL divider and illustrates the IICF relationship to the SCL divider variables.

Equation 9

$$\text{SCL divider} = \text{mul} \times [2 \times \{2 + \text{SCL2tap} + ((\text{SCL\_TAP} - 1) \times \text{tap2tap})\}]$$

Each of the variables in the SCL divider (MUL, SCI2tap, tap2tap, and SCL_TAP can be derived from the IIC frequency divider register (IICF) IBCn bits.

The IIC baud rate is computed by *Equation 10*.

Equation 10:

$$\text{IIC baud rate} = \frac{f_{Bus}}{(\text{mul} \times [2 \times \{2 + \text{SCL2tap} + (\text{SCL\_TAP} - 1) \times \text{tap2tap})\}]}$$

where: $\text{MUL} = 2^{\text{IBC[7:6]}}$

$\text{tap2tap} = 2^{\text{IBC[5:3]}}$

$\text{SCL\_TAP} = 5 + \text{IBC[2:0]}$, if IBC[2:0] = 6, Tap = 12
if IBC[2:0] = 7, Tap = 15

$\text{SCL2tap} = (2^{\text{IBC[5:3]}} - 2)$, if IBC[5:3] = 0, SCL2tap = 4
if IBC[5:3] = 1, SCL2tap = 4
if IBC[5:3] = 2, SCL2tap = 6

TIM Clock Rate

The only clock source for the TIM module is the bus clock. The TIM clock feeds into the three 4-channel timers with 16-bit counters.

The TIM clock rate, $R_{DesiredTIM}$, is a function of the bus clock divided by the timer prescaler. The timer prescaler includes the PR[2:0] bits of the TSCR2 register. The clock rate is computed by *Equation 11*.

Equation 11:

$$R_{DesiredTIM} = \frac{f_{Bus}}{2^{(PR[2:0])}}$$

ATD Conversion Clock Frequency

The only clock source for the analog-to-digital (ATD) module is the bus clock. The ATD clock source feeds into a 16-channel ATD converter with 10-bit resolution and sets the ATD conversion clock frequency, $f_{DesiredATD}$. Depending on the power supply voltage applied to the ATD converter, the ATD conversion clock frequency should be in the range from 0.5 MHz to 2 MHz.

The ATD conversion clock frequency is a function of both the bus clock and an ATD prescaler value, PRS[4:0]. The ATD conversion clock frequency can be calculated using *Equation 12*.

Equation 12:

$$f_{DesiredATD} = \frac{f_{BUS}}{2 \times (PRS[4:0] + 1)}$$

The PRS[4:0] bits are found in the ATD control register 4 (ATDCTL4). PRS[4:0] allows 32 prescaler values to divide the bus clock. Based on the selected bus clock frequency and required system conversion times, the user must choose the prescaler value to achieve the required ATD conversion rate — keeping in mind the minimum and maximum ATD clock range.

RTI Time-Out Period

The real-time interrupt (RTI) function can be used to generate a hardware interrupt at a fixed periodic rate, $f_{DesiredRTI}$. The RTI runs with a gated oscillator clock, OSCCLK. The OSCCLK feeds into a RTI prescaler.

The RTI prescaler is controlled using the CRG RTI control register (RTICTL). This register controls the two RTI frequency divide rate components. The RTI prescaler rate select bits (RTR[6:4]) and the RTI modulus counter select bits (RTR[3:0]) are combined to determine the overall RTI frequency divide rate. Use *Equation 13* to calculate the RTI periodic rate.

Equation 13:

$$f_{DesiredRTI} = \frac{f_{Bus}}{((RTR[3:0] + 1) \times (2^{(9 + RTR[6:4])}))}$$

Note that if RTR[6:4] is equal to zero, the RTI is disabled. In SCM mode, the RTI operates using f_{SCM}.

COP Time-Out Period

The computer operating properly (COP) is a free-running watchdog timer. It enables the user to check whether a program is running and sequencing correctly. Like the RTI, the COP runs with a gated oscillator clock, OSCCLK. Three control bits, CR[2:0], in the COPCTL register allow the selection of seven COP timeout periods, $t_{DesiredCOP}$. Use *Equation 14* to calculate $t_{DesiredCOP}$. Note the special cases provided.

Equation 14:

$$t_{DesiredCOP} = t_{Bus} \times 2^{(12 + 2 \times CR[2:0])}$$

Special cases:

- if CR[2:0] = 6, $t_{DesiredCOP} = t_{Bus} \times 2^{(23)}$
- if CR[2:0] = 7, $t_{DesiredCOP} = t_{Bus} \times 2^{(24)}$

Note that if COP[2:0] is equal to zero, the RTI is disabled. In SCM mode, the RTI operates using f_{SCM}.

PWM Period

The only clock source for the PWM module is the bus clock. The PWM clock feeds into a 6-channel pulse width modulator with 8-bit counters. (Note that the PWM can be configured into a 3-channel pulse width modulator with 16-bit counters using the PWMCTL register concatenate channels bits.)

Four clocks are available to the PWM. Clock selection is facilitated using the PWM clock select register (PWMCLK). With some restrictions, each PWM channel is capable of selecting one of the clocks sources.

- Clock A — Can be used for channels 0, 1, 4, or 5. It is configured using the PWM prescaler register (PWMPRCLK).
- Clock B —Can be used for channels 2 or 3. It is configured using the PWM prescaler register (PWMPRCLK).
- Scaled clock A — Uses clock A as input clock, which is scaled by dividing the clock by two times the PWM scale A register (PWMSCLA). The PWMSCLA register is a user-programmable 8-bit value.
- Scaled clock B — Uses clock B as input clock, which is scaled by dividing the clock by two times the PWM scale B register (PWMSCLA). The PWMSCLB register is a user-programmable 8-bit value.

In the case of a PWM, a programmer is typically concerned with setting the period and the duty cycle. Most applications dictate several configuration requirements of a PWM operation. These include alignment, polarity, duty cycle, and period. The alignment, duty cycle, and polarity are quite easily set because they are not clock dependent. Please refer to the PWM specification for additional information about setting up these parameters.

The desired period, $t_{Desired}$, is a little more involved because it depends on the clock source and requires that both the selected clock source and PWMPERx registers are configured. The discussion that follows explains, in more detail, how to use the PWM register set to obtain the required PWM period. In the discussion, a suffix “x” indicates that either clock A or clock B can be used.

The desired period, $t_{Desired}$, is composed of many individual cycles of the selected clock source as shown in *Equation 15*.

Equation 15:

$$t_{Desired} = t_{SourceCLK} \times PWMPERx \times ANF$$

In the equation, $t_{SourceCLK}$ is the period of one cycle of the selected clock source, PWMPERx is an 8-bit value that is stored in the PWMPERx register, and ANF is a factor that depends on the alignment mode of the desired PWM output. The alignment mode of the PWM output is set by the PWMCAE register and can be left aligned (ANF = 1) or center aligned (ANF = 2).

Equation 15 shows that PWMPERx is the number of $t_{SourceCLK}$ periods required to achieve the desired period, $t_{Desired}$. For the equation, both $t_{SourceCLK}$ must be determined before the desired period, $t_{Desired}$, can be calculated.

With $t_{Desired}$ and bus clock frequency (f_{Bus}) known, $t_{SourceCLK}$ must be determined. $t_{SourceCLK}$, recall, is the period of the selected clock source and is a function of the bus clock frequency (f_{Bus}). The relationship between f_{Bus} and $t_{SourceCLK}$ is illustrated in *Equation 16*.

Equation 16:

$$t_{SourceCLK} = \frac{1}{(f_{Bus} \div (2^{PCKx[2:0]})) \times SCF} = \frac{2^{PCKx[2:0]}}{(f_{Bus}) \times SCF}$$

PCKx[2:0] is a prescaler that is used to generate longer periods. PCKx[2:0] is found in the PWMPRCLK register. In *Equation 16*, the scaled clock factor (SCF) adds the effect of using a scaled source clock, which generates even longer periods. If a scaled source clock is not used, SCF is equal to a value of one (SCF = 1). If a scaled source clock is used, SCF is equal to two times the value stored in the PWMSCLx register (SCF = 2 × PWMSCLx). Adding the effect of using a scaled source clock, the formula becomes *Equation 17*.

Equation 17:

$$t_{SourceCLKScaled} = \frac{(2^{PCKx[2:0]})}{(f_{Bus}) \times (2 \times PWMSCLx)}$$

PMF Period and Deadtime

The only clock source for the pulse-width modulator with fault protection (PMF) module is the bus clock. The PMF clock feeds into a 6-channel PMF and three 15-bit counters. As an alternative to operating as a clock, the PMF can also operate using an output control method that enables software to control the PMF outputs rather that the PMF timebase generators. Output control uses the OUTCTLx and OUTx registers. See the PMF block guide for more information.

The PMF and PWM modules have similar operational characteristics, but the configuration of their control registers is dissimilar. The PMF is considered an enhanced PWM.

Three clocks are available to the PMF. Each of these clocks can control a specific PWM channel pair depending on the PMF configuration. The list below details which clock each of the PMF channels is paired with when the multiple timebase setting is engaged.

- Clock A can be the clock source for PWM0 and PWM1
- Clock B can be the clock source for PWM2 and PWM3
- Clock C can be the clock source for PWM4 and PWM5

All PMF channels can be controlled with a single PMF clock, clock A. To select multiple timebase generators, as shown above, the MTG bit of the PMFCTL0 register must be asserted. If MTG bit is not asserted, all the PMF channels operate using a single timebase that will source the clock from clock A.

PMF programmers are also concerned with setting the period and the duty cycle. A more complete set of parameters includes setting alignment, polarity, duty cycle, and period. For the PMF, the programmer can also set up complementary mode, fault detection and behavior, deadtime, and others (see the PMF specification for a complete list). The PMF parameters that are not clock dependent are not described in this document.

The desired PMF period, $t_{Desired}$, requires that both the clock source and PMFMODx registers are configured. The discussion that follows explains how to use the PMF register set to obtain the required PMF settings. In the discussion below, a suffix “x” indicates that timebase A, B, or C can be used. Recall that in single timebase mode, all PMF channels use only timebase A.

The desired period ($t_{Desired}$) is composed of many individual cycles of the selected clock source as shown in the *Equation 18* below. In the equation, $t_{SourceCLK}$ is the period of one cycle of the selected clock source, PMFMODx is a 16-bit value that is stored in the PMFMODx register, and ANF is a factor that adds an adjustment to the equation depending on the alignment mode of the desired PMF output. The alignment mode of the PWM output is set by the PMFCTL0 register and can be edge aligned (ANF = 1) or center aligned (ANF = 2).

Equation 18:

$$t_{Desired} = t_{SourceClk} \times PMFMODx \times ANF$$

Equation 18 shows that $t_{SourceCLK}$ must be determined before $t_{Desired}$ can be calculated.

With $t_{Desired}$ and bus clock frequency (f_{Bus}) known, $t_{SourceCLK}$ must be determined. Recall that $t_{SourceCLK}$ is the period of the selected clock source

and is a function of f_{Bus}. The relationship between f_{Bus} and $t_{SourceCLK}$ is illustrated in *Equation 19*.

Equation 19.

$$t_{SourceClk} = \frac{1}{f_{Bus} \div 2^{PRSCx[1:0]}} = \frac{2^{PRSCx[1:0]}}{f_{Bus}}$$

PRSCx[1:0] is a prescaler that allows for configuring longer periods. PRSCx[1:0] is found in the PMFFQCx register.

For the PMF, the deadtime setting also is clock dependent. Deadtime is only relevant in the complementary channel operation of a pair of PWM channels. While in the complementary mode, the PWM pairs outputs that are inversions of each other and must not have the same output state. Adding deadtime to the output ensures that the output states are not the same by automatically inserting software-selectable activation delays into each pair of PWM outputs.

Deadtime is configured using the PMFDTMx register. The PMFDTMx register is a 12-bit value indicating the number of PWM clock cycles for the duration of the deadtime event. For deadtime, the desired amount of deadtime ($t_{DesiredDT}$) is governed by *Equation 20*.

Equation 20:

$$t_{DesiredDT} = t_{SourceClk} \times 2^{PRSCx[1:0]} \times (PMFDTMx - 1)$$

The value of PMFDTMx can be obtained from the equation above. An expression for the value of PMFDTMx can be obtained by rearranging *Equation 20* as shown in *Equation 21*:

Equation 21:

$$PMFDTMx = \frac{(t_{DesiredDT} \div t_{SourceClk}) + 1)}{2^{PRSCx[1:0]}}$$

Conclusion

The CRG is a highly flexible module on the HCS12 Family of microcontrollers. It provides multiple options for MCU clock sources. Because of the PLL, designers also can increase the clock speed for both the core and bus clocks. The flexibility of the CRG makes setting up the MCU's peripherals much simpler. Bus frequencies can be easily tailored for optimum performance of a specific module. These features allow designers a lot of flexibility when balancing cost, precision, and performance in an application.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com