Freescale Semiconductor
Application Note

AN2882
Rev. 1, 5/2005

Using Real-Time Interrupt on HCS12 Microcontrollers

By Amin Morales
RTAC Americas
Mexico 2004

Introduction

This document is intended to serve as a quick reference for an embedded engineer to get the real-time interrupt (RTI) module up and running for any HCS12 MCU. Basic knowledge about the functional description and configuration options will give the user a better understanding on how the RTI module works. This application note provides examples which demonstrate one use of the RTI module for the HCS12 Family of microcontrollers. The examples mentioned are intended to be modified to suit the specific needs for any application.

The example CodeWarrior project files are available as AN2882SW.zip from http://freescale.com.

Description

The RTI is a sub-system of the clock and reset generator module (CRG) shown in Figure 1. The RTI can be used to generate a hardware interrupt at a periodic rate. If enabled (by setting RTIE=1), this interrupt will occur at the rate selected by the RTICTL register.

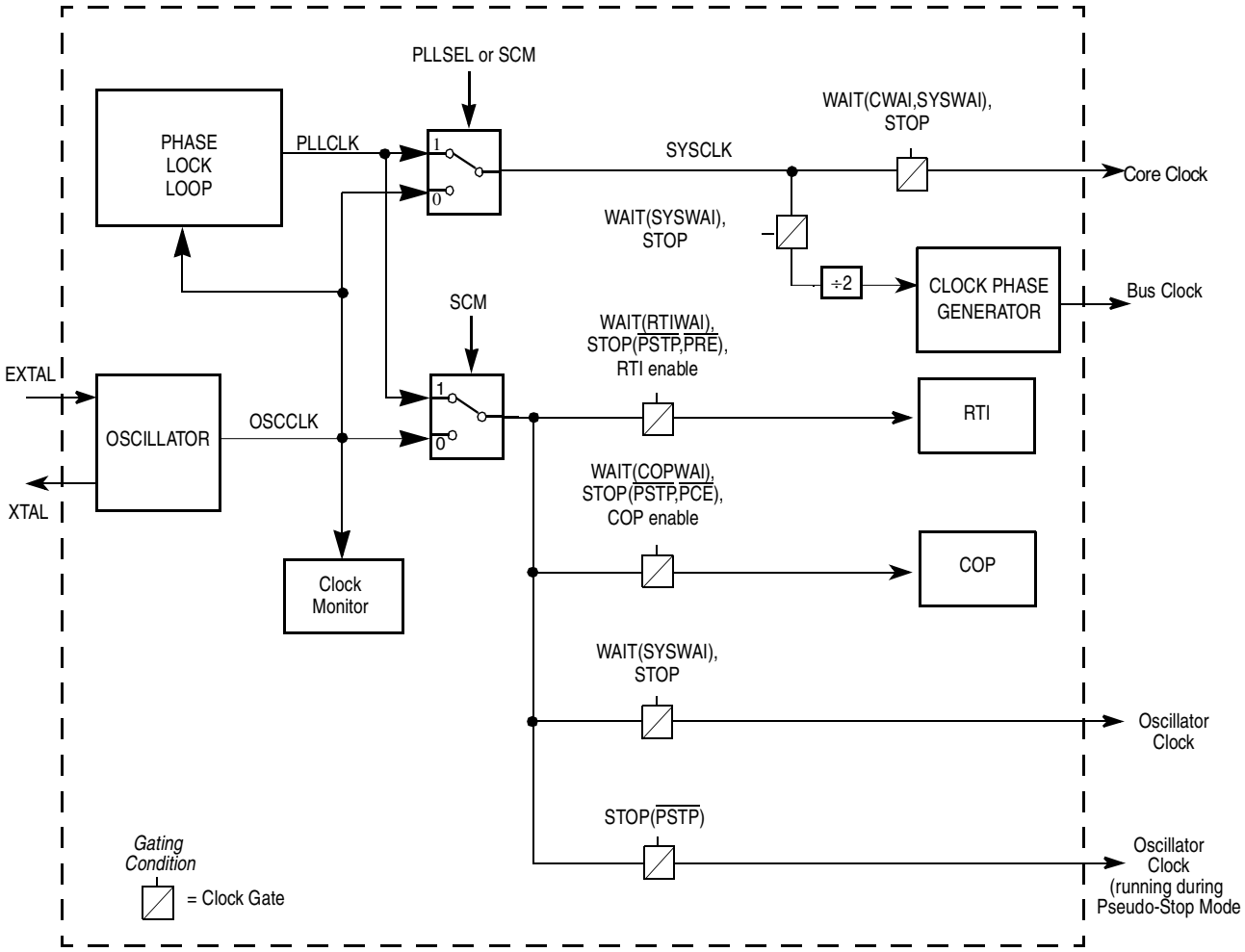

Figure 1. System Clocks Generator

The RTI normally uses the oscillator clock as its clock source (OSCCLK), as shown in Figure 2. At the end of the RTI time-out period, the RTIF flag is set and a new RTI time-out period starts immediately.

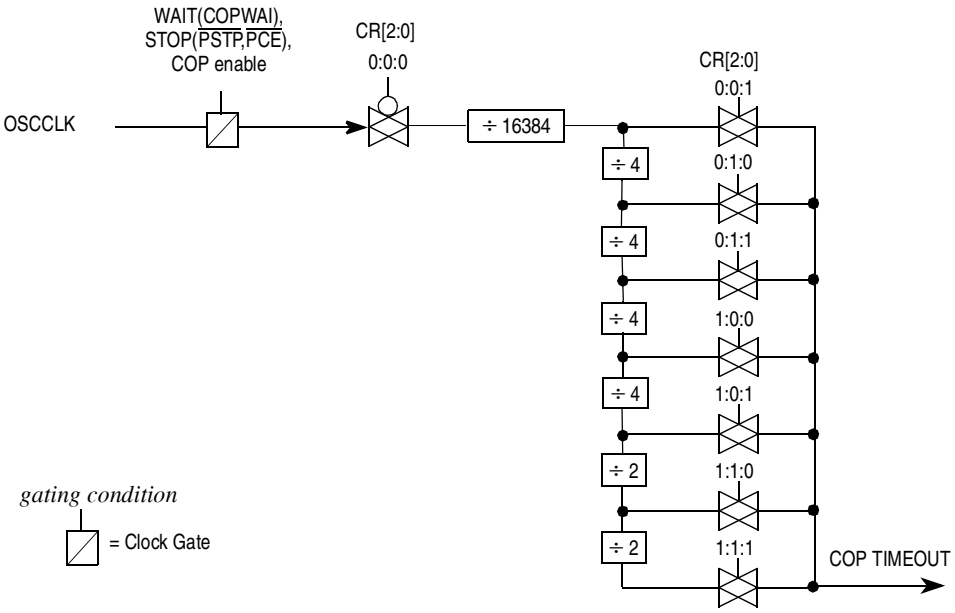

Figure 2. Clock Chain for RTI

Writing to the RTI control register restarts the RTI time-out period. If the PRE (enable during pseudo stop) bit is set, the RTI will continue to run in pseudo-stop mode. This feature can be used for periodic wakeup from pseudo stop if the RTI interrupt is enabled.

RTI Control and Configuration Registers

This section describes the control and configuration registers of the RTI. More information is available in the CRG block guide, Freescale document number S12CRGV4. Only the bits which influence functionality of the RTI module will be described in the following text.

CRG Flags Register (CRGFLG)

This register provides CRG status bits and flags.

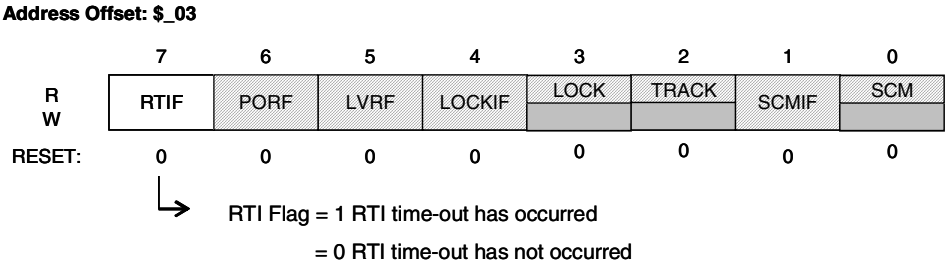

Figure 3. CRGFLG Register

CRG Interrupt Enable Register (CRGINT)

This register enables CRG interrupt requests.

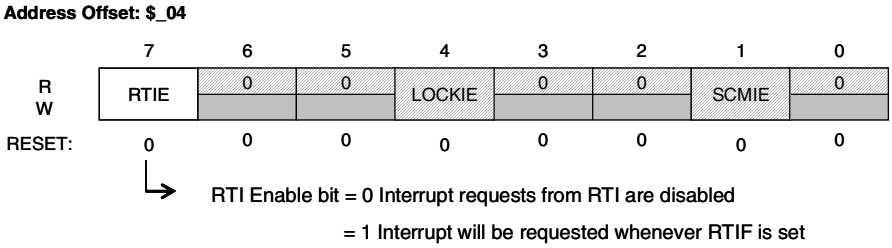

Figure 4. CRGINT Register

CRG Clock Select Register (CLKSEL)

This register controls CRG clock selection. Refer to Figure 1 for more details on the effect of each bit.

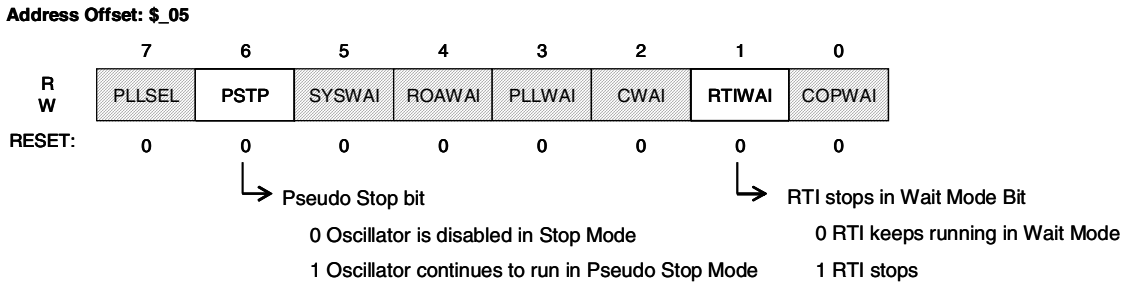

Figure 5. CLKSEL Register

CRG PLL Control Register (PLLCTL)

This register controls the PLL functionality.

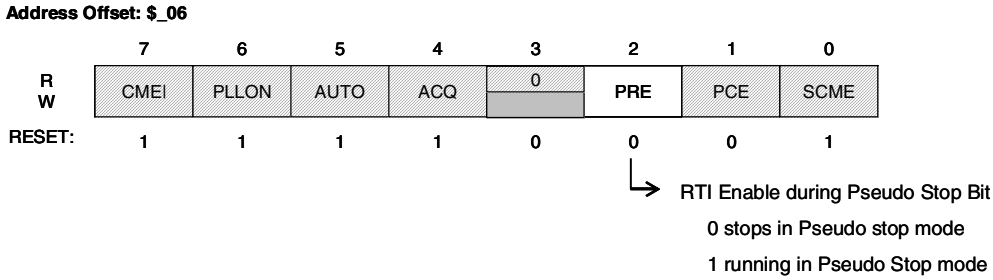

Figure 6. PLLCTL Register

CRG RTI Control Register (RTICTL)

This register selects the timeout period for the real-time interrupt.

NOTE

A write to this register initializes the RTI counter.

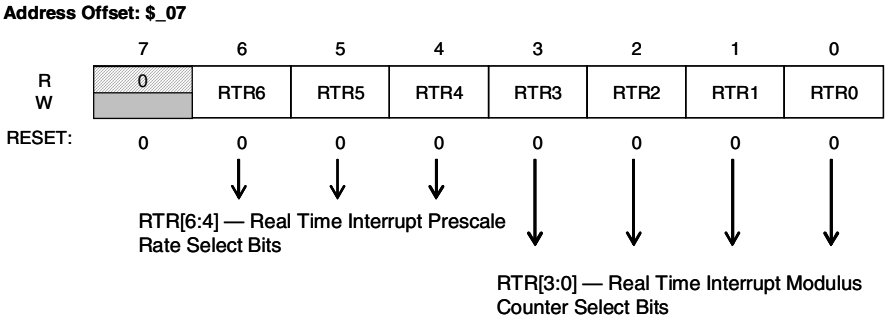

Figure 7. RTICTL Register

Table 1. RTI Frequency Divide Rates

RTR[3:0]	RTR[6:4] =							
	000 (OFF)	001 (2^{10})	010 (2^{11})	011 (2^{12})	100 (2^{13})	101 (2^{14})	110 (2^{15})	111 (2^{16})
0000 (±1)	OFF*	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
0001 (±2)	OFF*	$2x2^{10}$	$2x2^{11}$	$2x2^{12}$	$2x2^{13}$	$2x2^{14}$	$2x2^{15}$	$2x2^{16}$
0010 (±3)	OFF*	$3x2^{10}$	$3x2^{11}$	$3x2^{12}$	$3x2^{13}$	$3x2^{14}$	$3x2^{15}$	$3x2^{16}$
0011 (±4)	OFF*	$4x2^{10}$	$4x2^{11}$	$4x2^{12}$	$4x2^{13}$	$4x2^{14}$	$4x2^{15}$	$4x2^{16}$
0100 (±5)	OFF*	$5x2^{10}$	$5x2^{11}$	$5x2^{12}$	$5x2^{13}$	$5x2^{14}$	$5x2^{15}$	$5x2^{16}$
0101 (±6)	OFF*	$6x2^{10}$	$6x2^{11}$	$6x2^{12}$	$6x2^{13}$	$6x2^{14}$	$6x2^{15}$	$6x2^{16}$
0110 (±7)	OFF*	$7x2^{10}$	$7x2^{11}$	$7x2^{12}$	$7x2^{13}$	$7x2^{14}$	$7x2^{15}$	$7x2^{16}$
0111 (±8)	OFF*	$8x2^{10}$	$8x2^{11}$	$8x2^{12}$	$8x2^{13}$	$8x2^{14}$	$8x2^{15}$	$8x2^{16}$
1000 (±9)	OFF*	$9x2^{10}$	$9x2^{11}$	$9x2^{12}$	$9x2^{13}$	$9x2^{14}$	$9x2^{15}$	$9x2^{16}$
1001 (±10)	OFF*	$10x2^{10}$	$10x2^{11}$	$10x2^{12}$	$10x2^{13}$	$10x2^{14}$	$10x2^{15}$	$10x2^{16}$
1010 (±11)	OFF*	$11x2^{10}$	$11x2^{11}$	$11x2^{12}$	$11x2^{13}$	$11x2^{14}$	$11x2^{15}$	$11x2^{16}$
1011 (±12)	OFF*	$12x2^{10}$	$12x2^{11}$	$12x2^{12}$	$12x2^{13}$	$12x2^{14}$	$12x2^{15}$	$12x2^{16}$
1100 (±13)	OFF*	$13x2^{10}$	$13x2^{11}$	$13x2^{12}$	$13x2^{13}$	$13x2^{14}$	$13x2^{15}$	$13x2^{16}$
1101 (±14)	OFF*	$14x2^{10}$	$14x2^{11}$	$14x2^{12}$	$14x2^{13}$	$14x2^{14}$	$14x2^{15}$	$14x2^{16}$
1110 (±15)	OFF*	$15x2^{10}$	$15x2^{11}$	$15x2^{12}$	$15x2^{13}$	$15x2^{14}$	$15x2^{15}$	$15x2^{16}$
1111 (±16)	OFF*	$16x2^{10}$	$16x2^{11}$	$16x2^{12}$	$16x2^{13}$	$16x2^{14}$	$16x2^{15}$	$16x2^{16}$

Algorithms and Software Examples

The flow diagrams in Figure 8 show the RTI configuration routine and the RTI interrupt service routine algorithms, which are implemented in the example code provide in this application note (AN2882SW.zip).

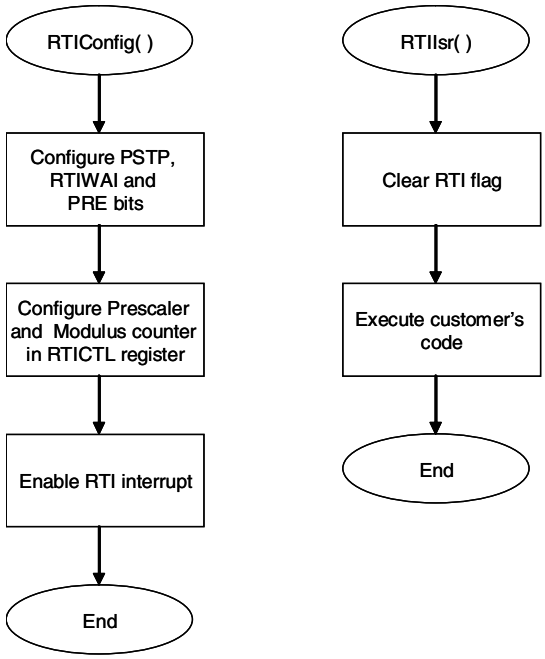

Figure 8. Configuring RTI and RTI Interrupt Service Function

Example Code

The example code provided in the following shows the use of the RTI module to manage events in time. In this particular code, a pin on the PORTP is toggled in 512 ms intervals. The RTI frequency is set to 976.56 Hz, which means 1.024 ms interrupt periods. The code consists of two functions: RTIConfig( ), which configures the RTI to the desired frequency, and RTIIsr( ), which executes every time the RTI interrupt period of 1.024 ms elapses. The interrupt service routine, RTIIsr(), maintains a software counter and toggles a port pin every 500 RTI interrupt periods. A square wave signal with a frequency of 0.97656 Hz can therefore be observed on the port pin. The RTI frequency is based on a clock of 8 MHz, which comes from the external crystal oscillator.

```
/**
 * Copyright (c) 2004, Freescale Semiconductor
 * Freescale Willy Note
 *
 * File name      : main.c
 * Project name: RTI Demo Software
 *
 * Author           : Amin Morales
 * Department   : RTAC Americas
 *
```

```
 * Description    : Configures RTI frequency at 976.56Hz and the
 *                          RTI Isr toggles a port pin each 500mS
 *
 * History           :
 * 07/09/2004     : Release. (A19258)
 */

#include <hidef.h>

/* PORTP definitions */
#define PTP      (*((volatile unsigned char*)(0x0258)))
#define DDRP     (*((volatile unsigned char*)(0x025A)))
/* RTI definitions */
#define CRGINT   (*((volatile unsigned char*)(0x0038)))
#define CRGFLG   (*((volatile unsigned char*)(0x0037)))
#define RTICTL   (*((volatile unsigned char*)(0x003B)))
/*Global variables*/
unsigned int rticounter;


#pragma CODE_SEG __NEAR_SEG NON_BANKED
/*
 * RTIIsr: Interrupt Service routine for the RTI.
 *  Clear RTI flag
 *  After 500 RTI Interrupts (~500 ms) toggle PORTP
 *  Reset RTI counter to restart cycle
 *
 * Parameters: None
 *
 * Return : None
 */
interrupt void RTIIsr(void) {

    CRGFLG = 0x80;  /* clear RTIF bit */

    if(rticounter == 500) {
        rticounter = 0;
        PTP = ~PTP;
    }
    else {

        rticounter++;
    }
    return;
}

#pragma CODE_SEG DEFAULT
/*
 * RTIConfig: Setup of the RTI interrupt frequency, adjusted to get
 *  1.024 ms with 8 MHz crystal oscillator
 *  RTI frequency = 8MHz/(8x2^10) = 976.5625 Hz
 *  Period of time between interrupts = 1/976.5625 Hz = 1.024 ms
 *
 * Parameters: None
 *
 * Return : None
 */
```

```
void RTIConfig(void){

    RTICTL = 0x17;  /* set RTI prescaler */
    CRGINT = 0x80;  /* enable RTI interrupts */

    return;
}

void main(void){

    DDRP = 0x80;          /*Configure pin 7 in PORT P as output*/
    PTP  = 0x80;            /*Set high pin 7 in PORT P*/

    rticounter = 0;          /*Initialize RTI counter*/
    RTIConfig();            /*Configure RTI Interrupt frequency*/

    EnableInterrupts;

    for (;;)
        ; /* Empty Body */
}
```

Conclusion

The RTI module is available on all HCS12 derivatives: A, B, C, D, DB, DJ, DG, DP, DT, E, H, KG, KT, NE. The example code provided demonstrates an easy way to configure the RTI to generate an interrupt in 1.024-ms periods and toggle a port pin every 512 milliseconds.

Considerations

Find these and other useful resources on the Freescale Semiconductor home page: http://www.freescale.com.

- One important consideration when programming the RTI is the frequency of the clock source because this will impact the RTI frequency.
- MC9S12DJ256 derivative was used to generate the RTI DemoSoftware.
- The RTI DemoSoftware code was developed in CodeWarrior 12 version 3.1
- Refer to CRG block guide, document number S12CRGV4, for more information on RTI.

This page is intentionally blank.

This page is intentionally blank.

This page is intentionally blank.

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

AN2882
Rev. 1, 5/2005

Freescale Semiconductor Internal Use Only

Preliminary

freescale™ semiconductor