Freescale Semiconductor
Application Note

Document Number: AN3792
Rev. 1.0, 06/2009

Computer Operating Properly (COP) Watchdog

Integrating the COP into MC1321x Synkro, BeeStack and 802.15.4 Applications

1 Introduction

This note describes how to integrate the Computer Operating Properly (COP) Watchdog into wireless, MC1321x applications that are using Freescale/802.15.4 based application stacks, such as Synkro, BeeStack and 802.15.4 MAC.

Users should be familiar with the Synkro www.freescale.com/synkro, and ZigBee Specifications and the 802.15.4 Standard available from www.zigbee.org ,www.ieee.org. Users should also be familiar with at least the basic operation of the Freescale Zigbee/802.15.4 devices. Refer to the appropriate Data Sheet or Reference Manual as needed for more information available at www.freescale.com/802154 or www.freescale.com/synkro.

Contents

© Freescale Semiconductor, Inc., 2009. All rights reserved.

2 Introduction to the COP Watchdog

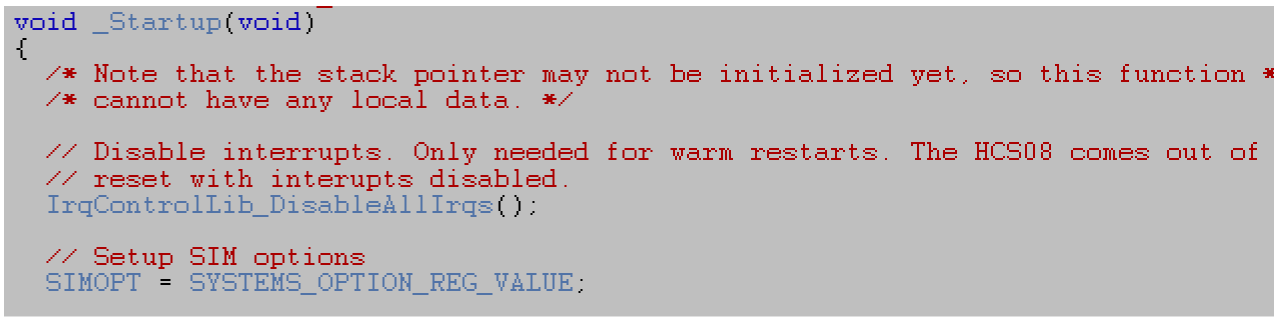
The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP timer periodically. If the application software gets lost and fails to reset the COP before it times out, a system reset is generated to force the system back to a known starting point.

The COP watchdog is enabled by the COPE bit in the “System Options Register (SOPT)”. The COP is reset by writing any value to the address of SRS.

The COPT bit in the SOPT register can be used to choose one of two time out periods; 2^{18} or 2^{13} of the bus rate clock. The longer of these time outs is recommended for applications detailed in this note.

2.1 Enabling the COP Watchdog in the Application

The application function ‘_Startup’ (located in the `crt0.c` source file) is the entry point for all applications using Freescale/802.15.4 based application stacks, such as Synkro, BeeStack and 802.15.4 MAC. This function can also initialize the SOPT register as shown in Figure 1. (in source code this register is called SIMOPT).

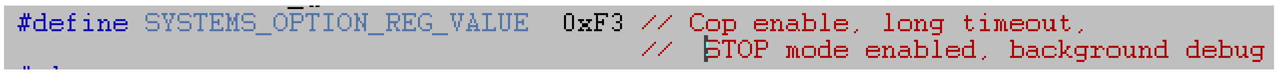


```
void _Startup(void)
{
  /* Note that the stack pointer may not be initialized yet, so this function *
  /* cannot have any local data. */

  // Disable interrupts. Only needed for warm restarts. The HCS08 comes out of
  // reset with interupts disabled.
  IrqControlLib_DisableAllIrqs();

  // Setup SIM options
  SIMOPT = SYSTEMS_OPTION_REG_VALUE;
```

Figure 1. COP Initialization in the SIMOPT Register

The source code macro ‘SYTEMS_OPTION_REG_VALUE’ is set in the source file `crt0.h`. The default value of this macro has the COP watchdog disabled.

To enable the COP with the long time out, modify the source file `crt0.h` to change this macro to that shown in Figure 2.

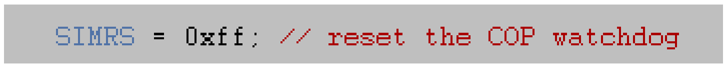


```
#define SYSTEMS_OPTION_REG_VALUE   0xF3 // Cop enable, long timeout,
                                        // STOP mode enabled, background debug
```

Figure 2. Systems Options Reg initialization value for COP enable with long time out

2.2 Resetting the COP Watchdog in the Application

Reset the COP watchdog by writing any value to the address of the read-only SRS register. In application source code this is accomplished as shown in Figure 3.



```
SIMRS = 0xff; // reset the COP watchdog
```

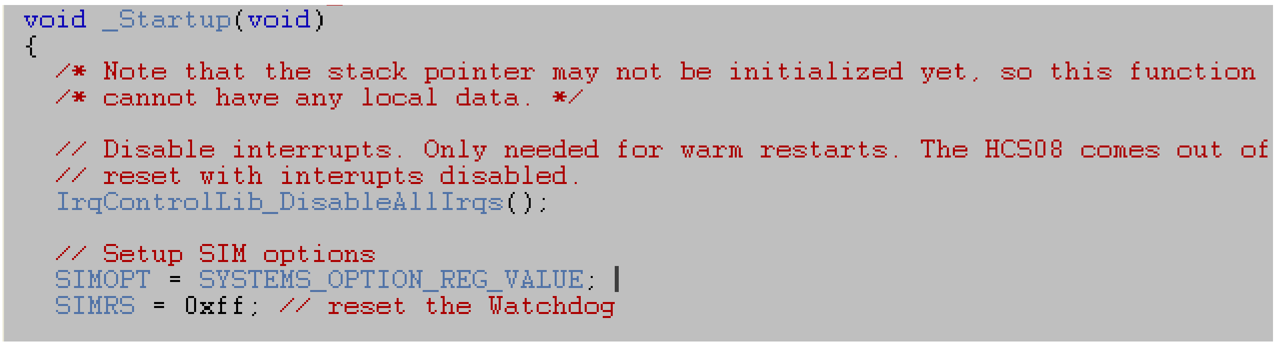
Figure 3. Resetting the COP Watchdog

3 Resetting the COP Watchdog in Application Code

The application must reset the COP watchdog periodically in order to avoid the COP timing out and performing a system reset.

3.1 Resetting the COP Watchdog in _Startup

The COP watchdog should be reset to guarantee its initial state right after its initialization in _Startup routine in the source code file `crt0.c` as in Figure 4.

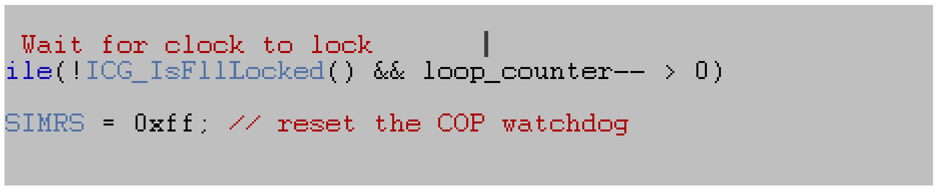


```
void _Startup(void)
{
  /* Note that the stack pointer may not be initialized yet, so this function
  /* cannot have any local data. */

  // Disable interrupts. Only needed for warm restarts. The HCS08 comes out of
  // reset with interupts disabled.
  IrqControlLib_DisableAllIrqs();

  // Setup SIM options
  SIMOPT = SYSTEMS_OPTION_REG_VALUE; |
  SIMRS = 0xff; // reset the Watchdog
```

Figure 4. Resetting the COP Watchdog in _Startup()

3.2 Resetting the COP Watchdog in the Internal Clock Generator (ICG) Module

During initialization, the ICG module synchronizes to the MC1321x clock in the source file `icg.c` routine ICG_Setup. To insure that the COP watchdog does not expire during this period, it must be reset within this loop, and shown in Figure 5.



```
 Wait for clock to lock        |
ile(!ICG_IsFllLocked() && loop_counter-- > 0)

SIMRS = 0xff; // reset the COP watchdog
```

Figure 5. Resetting the COP Watchdog in the ICG_Setup

3.3 Resetting the COP Watchdog in the Task Scheduler Loop

The task scheduler is the main loop for the application. The task scheduler routine, TS_Scheduler is located in the source file `TS_Kernel.c`. The application should reset the COP watchdog at the top of the task scheduler loop as shown in Figure 6.

```
void TS_Scheduler(void) {
  index_t activeTask;
  uint8_t ccr;
  event_t events;
  index_t i;
  index_t taskID;


  /* maTsTaskIDsByPriority[] is maintained in task priority order. If there */
  /* are fewer than the maximum number of tasks, the first gInvalidTaskID_c */
  /* marks the end of the table. */
  for (;;) {

    SIMRS = 0xff; // reset the COP watchdog
```

Figure 6. Resetting the COP Watchdog at the Top of the Task Scheduler Loop

3.4 Resetting the COP Watchdog in the Low Power Module

The COP watchdog should be reset before entering deep sleep mode. The application function PWR_HandleDeepSleep in the source code file `pwr.c` should be modified to reset the watchdog before calling PWRLib_MCUStop3, for each deep sleep mode.

Additionally, the COP watchdog should be reset after exiting deep sleep mode.

These changes are shown as an example for deep sleep mode 3 in Figure 7. Other deep sleep modes will require similar modifications.

```
  #elif (cPWR_DeepSleepMode==3)
    #if (cPWR_KBIWakeupEnable)

      PWRLib_MCU_WakeupReason.Bits.FromKBI = FALSE; // Clear any prior KBI wake

      PWRLib_RTIClockStart( cPWR_RTITickTime, DozeDuration);
      if (PWR_Stop3AndOff() != FALSE) {
        while ((PWRLib_MCU_WakeupReason.Bits.FromKBI == 0 ) && ( PWRLib_RTICloc

          SIMRS = 0xff; // reset the COP watchdog
          (void) PWRLib_MCUStop3();

        }
        PWRLib_RTIClockStop();
        #if (gTMR_EnableLowPowerTimers_d)
        /* Sync. the low power timers */
        TMR_SyncLpmTimers(MillisToSyncLpmTimers(notCountedTimeBeforeSleep));
        #endif /* #if (gTMR_EnableLowPowerTimers_d) */
        Res.AllBits = PWRLib_MCU_WakeupReason.AllBits;

        PWR_RunAgain();
        SIMRS = 0xff; // reset the COP watchdog
```

Figure 7. Resetting the COP Watchdog Before Entering and After Leaving Deep Sleep Mode

3.5 Resetting the COP Watchdog in the NVM/FlASH Module

The COP watchdog should be reset during flash accesses. The application function NvHalRead in the source code file `NV_FlashHal.c` should be modified to reset the watchdog while reading bytes from FLASH as shown in Figure 8.

```
/* FLib_MemCpy() can't handle a 16 bit size. */
while ( dstLen-- )
{
  SIMRS = 0xff; // reset the COP watchdog

  /* Copy one byte a the time of the given byte Length. */
  *pDst++ = *pSrc++;
}
```

Figure 8. Resetting the COP During FLASH Reads

Additionally, the COP watchdog should be reset inside the three wait loops within the NvHalExecuteFlashCmd routine as shown in Figure 9.

```
/* Wait for flash to be ready for a command. */
while ( !( FSTAT & FCCF ))
{
    SIMRS = 0xff; // reset the COP watchdog
    /* Do nothing for a while. */
}
```

Figure 9. Resetting the COP in Wait Loops in NvHalExecuteFlashCmd

NOTE

There are three of these wait loops in this routine.

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
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 Semiconductor 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 Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

Document Number: AN3792
Rev. 1.0
06/2009