COP8 External Sleep Timer

National Semiconductor
Application Note 1091
Scott O. Campbell
June 1998

1.0 ABSTRACT

Many microcontroller applications require the system to wake up periodically to perform some function (such as remote meter reading) while the vast majority of time can be spent in some low-power mode to conserve power (often from a battery). Therefore, a sleep mode of several seconds at ultra-low power consumption is useful in these applications.

Normal operating current for a typical COP8 ranges from 2 to 10 mA depending on the supply voltage and clock speed. The COP8 feature family of microcontrollers has two built-in power saving modes, IDLE and HALT. The IDLE mode allows the COP8 to go into a low-power mode where supply current is reduced to approximately 1 mA for 2^{12} instruction cycles (4.1 ms at 10 Mhz). This is not a very long time for many applications that only need to awake every few seconds. Also, the power savings is not enough to be useful in battery powered applications. However, knowing the exact duration of the IDLE period is very useful for many applications.

The COP8 HALT mode actually stops the internal clock and reduces supply current to a small leakage current, typically less than 1 µA. Another advantage of HALT mode is that the supply voltage (V_{CC}) can be reduced to 2.0V without altering the state of the machine. However, in order to exit the HALT mode, an external signal must be supplied. This makes it difficult to know the duration of time spent in the HALT mode in most applications.

This application note describes how to combine the best features of the IDLE and HALT mode to create a low-cost External Sleep Timer. This timer consists of an external RC network controlled by the COP8 to create a sleep period of several seconds. During the sleep period, the COP8 is in the HALT mode, keeping power consumption to a minimum. Prior to entering the sleep mode, the timer may be calibrated so that the sleep period may be accounted for when waking up.

2.0 INTRODUCTION

The Multi-Input Wakeup (MIWU) feature of the COP8 is the key to making this application work. This feature is one of several methods that may be used to wakeup the device from either the HALT or IDLE modes. The MIWU feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the device to exit the HALT or IDLE mode. The selection is done through the register WKEN. The register WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the register WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the device will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user must clear the pending flags before attempting to enter the HALT mode.

3.0 THEORY OF OPERATION

Please refer to the schematic shown in *Figure 1* and the timing diagram in *Figure 2* for the following discussion. The basic concept of the sleep timer is to slowly charge and discharge a capacitor (C1) while the COP8 is in the HALT mode. The charging voltage is supplied by port L2 through resistor R1. Port L3 is placed into a high impedance input mode for the duration of the sleep timer operation. The time required to charge C1 is determined by the time constant of R1 and C1. After C1 is charged, port L2 is placed into a high impedance state to allow C1 to discharge. The discharge path is R2 in parallel with R1 and the path that L2 provides to ground in parallel with the path that L3 provides to ground. The time required for C1 to discharge is determined by the time constant of (R2 || R1 + L2 || L3) and C1. The voltage across the capacitor is monitored by port L3 which is configured as a Multi-Input Wakeup. When the voltage across C1 reaches the trigger threshold of pin L3 (which is configured as either positive-going or negative-going), the COP8 exits the HALT mode and resumes code execution. Alternating cycles of charge and discharge may be repeated as long as the user wishes to remain in sleep mode. An 8-bit counter may be used to simplify the task of repeating the loop for several cycles. The COP8 only wakes up long enough to decrement the counter and reverse the charge/discharge cycle.

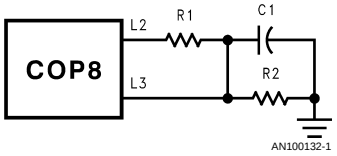

FIGURE 1. Basic Circuit

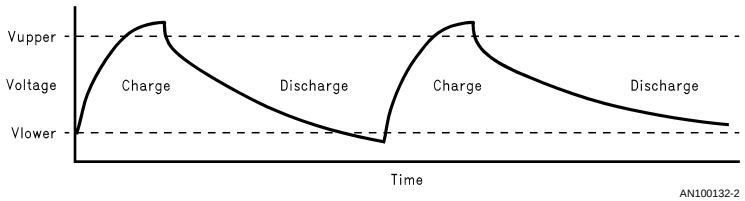

FIGURE 2. C1 Charge/Discharge Cycles

4.0 IMPLEMENTATION

The first thing to consider when selecting components is the desired sleep period. For this example, a sleep period of 10 seconds is chosen. This means that one complete charge/discharge cycle should take approximately 10 seconds. It must be remembered however, that this is very approximate based on the tolerance of capacitor C1, leakage currents on port L, logic input level variations, ambient temperature fluctuations, and the voltage levels on the RC network if V_{CC} is unregulated. This is where self-calibration becomes useful. It will be described later.

It should be noted that the voltage developed across C1 remains within the Schmitt trigger hysteresis window of L3 during the charge/discharge cycle. Based on COP8 characterization data for a V_{CC} of 5V a typical input level for a logic high on Port L is 2.8V and a typical input level for a logic low is 2.2V. Therefore the hysteresis window of 0.6V is only about 34% of the first time constant when charging or discharging C1. This means that we must select the RC circuit to have a time constant of about 3 times our desired 10 second sleep period, or 30 seconds. Remember, as V_{CC} decreases, so does the hysteresis window (as a function of V_{CC}), and so does the sleep period. Note that the Schmitt trigger thresholds can vary from device to device depending on the transistor thresholds of the particular lot that the COP8 device was processed. The logic high threshold can vary from 2.3V to 2.9V and the logic low threshold from 1.9V to 2.4V at 25˚C and a V_{CC} of 5V. Even wider variations may be seen at extreme temperatures.

To simplify the process of selecting component values, C1 is chosen to be 10 µF. This allows a fairly small tantalum capacitor to be used which keeps the required PCB space to a minimum. Also, resistor R2 should be at least 10 times the value of R1 to ensure that C1 can be fully charged by port L2. One sleep period is the sum of the charge and discharge time constants (R1*C1) and (R2 || (R1+L2) || L3)*C1. In addition, the minimum input impedance of the L port pins is on the order of 6 MΩ. Therefore the value of R1 can be determined by the following formula:

(R1*10e-6) + (R1*10 || R1+6M || 6M)*10e-6 = 30 seconds

Substituting the common resistor value of 680 kΩ for R1 yields a time constant of 28.4 seconds. Then 34% of 28.4 is 9.6 seconds which is quite close to our original value of 10 seconds. Don't forget that R2 is 10 x R1 or 6.8 MΩ.

5.0 SOFTWARE DESCRIPTION

The sleep timer can be implemented with one short subroutine. It is described in the flowchart in *Figure 3* and the code listing in section 6. If knowing the exact period of the sleep timer is important to your application, a calibration routine is described in the flowchart in *Figure 4* and the code is listed in section 7.

The sleep timer subroutine first calls the calibration routine before entering the sleep mode. This way, subsequent sleep periods can be accounted for if desired. Next, pin L3 is checked to ensure it is a logic low before beginning to charge C1. Then pin L2 is configured to charge C1 and pin L3 is configured to detect when C1 reaches the input logic high level and wake the device. The device is then placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). It is during this time that C1 is charged. Note that it is necessary to program two NOP instructions following the set HALT mode instruction. These NOP instructions are necessary to allow clock resynchronization following the HALT mode. When the voltage on C1 reaches the logic high threshold of port L3, the device exits the HALT mode. Now that C1 has been charged, port L2 is put into a hi-Z state, allowing C1 to slowly discharge while the COP8 goes back into HALT mode. Port L3 is now configured for detecting when C1 reaches the input logic low level. Please refer to the COP8 data book for details on the operation of the HALT mode.

The calibration routine executes one complete charge/discharge cycle of the sleep timer to determine the actual elapsed time at the ambient temperature and supply voltage of the system. This value is then saved and later used by the sleep routine to accumulate the amount of time in the sleep mode. Note that the calibration routine first gets the capacitor voltage into the hysteresis window of port L3 before beginning the charge/discharge cycle. The calibration routine can be run as often as the user feels necessary to keep the sleep timer accurate. The accuracy of the calibration routine is determined by the user since it uses a timer interrupt routine to determine the sleep period. The example included here uses a resolution of 1 second.

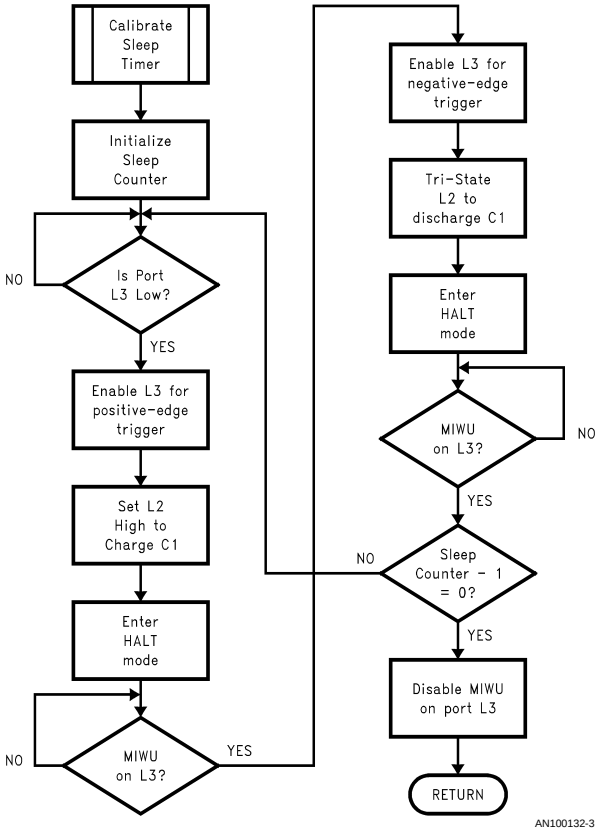

FIGURE 3. Sleep Timer Flowchart

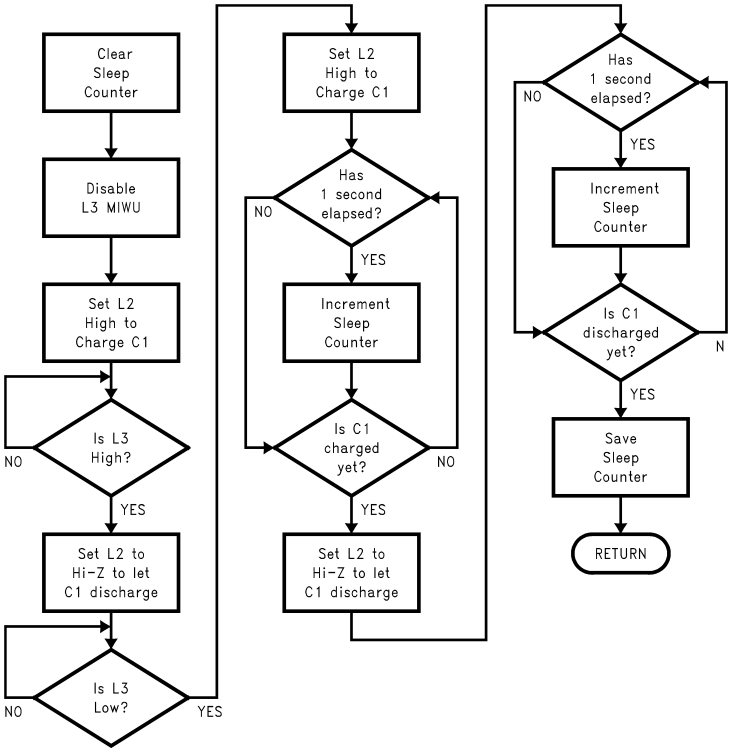

FIGURE 4. Calibration Flowchart

6.0 SLEEP TIMER CODE LISTING

```
;***************************************************************************
;  Sleep
;  Implements Sleep Timer using external RC circuit on Ports L2 and L3
;***************************************************************************
;
     .INCLD  COP888CF.INC

;*****************************************************
; RAM Allocation (96 bytes: 0x10 - 0x6F)
; Note: Stack starts at 0x6F and grows downward
;
     .SECT   DATA, RAM
bSleepTime      : .DSB  1          ; Calibration result

;*****************************************************
;   REGISTER DEFINITIONS, (12 bytes: 0xF0 to 0xFB)
;   For use with DRSZ REG and LD REG,# instructions
;
      .SECT   REGISTERS, REG
rSleepCounter   : .DSB  1          ; Sleep-Timer Counter

;*****************************************************
;
      .SECT   SLEEP, ROM
      .EXTRN  Calibrate
      .PUBLIC Sleep, bSleepTime, rSleepCounter
;
; CONSTANT DECLARE
```

```
;     Bit definitions on port L
      SLEEP_OUT   = 2             ; used to charge C1
      SLEEP_IN    = 3             ; used to monitor C1
;
;*****************************************************
;
Sleep:
;
; Turn off any external peripherals here...
;
      LD    rSleepCounter, #16    ; Initialize Sleep Counter
SLEEP1:
      IFBIT SLEEP_IN,  PORTLP     ; Ensure L3 is low
      JP    SLEEP1                ; Wait until C1 discharged
SLEEP2:
      RBIT  SLEEP_IN,  WKEN       ; Disable MIWU
      RBIT  SLEEP_IN,  WKEDG      ; Trigger on positive edge
      RBIT  SLEEP_IN,  WKPND      ;Clear pending flag
      SBIT  SLEEP_IN,  WKEN       ; Enable MIWU
      SBIT  SLEEP_OUT, PORTLD     ; Begin Charging C1
      SBIT  SLEEP_OUT, PORTLC
      SBIT  7, PORTGD             ; Enter HALT mode
;
; Wait here until MIWU occurs
;
      NOP                         ; 2 NOP's required after HALT
      NOP
      RBIT  SLEEP_IN,  WKEN       ; Disable MIWU
      SBIT  SLEEP_IN,  WKEDG       ; Trigger on negative edge
      RBIT  SLEEP_IN,  WKPND      ; Clear pending flag
      SBIT  SLEEP_IN,  WKEN       ; Enable MIWU
      RBIT  SLEEP_OUT  PORTLC      ; Allow C1 to discharge
      RBIT  SLEEP_OUT, PORTLD
      SBIT  7, PORTGD             ; Enter HALT mode
;
; Wait here until MIWU occurs
;
      NOP                         ;2 NOP's required after HALT
      NOP
;
; If other MIWU's are enabled, they should be checked here
;
      DRSZ  rSleepCounter         ; Decrement Sleep Counter
      JP    SLEEP1                ; Repeat loop until counter = 00

SLEEP99:
      RBIT  SLEEP_IN,  WKEN       ; Disable MIWU on L3
     RBIT  SLEEP_IN,  WKPND       ; Clear pending flag
      RBIT  SLEEP_OUT, PORTLC      ; Put L2 in Hi-Z mode
      RBIT  SLEEP_OUT, PORTLD
;
; Can calculate total Sleep time here by multiplying number of loops
; by the value stored in bSleepTime
;
      RET
```

7.0 CALIBRATION CODE LISTING

```
;***********************************************************************
; CALIBRATE
; Calibrates the Sleep Timer to within 1 second resolution
; Call before SLEEP routine for best accuracy
;***********************************************************************
;
      .INCLD      COP888CF.INC
      .SECT       SLEEP, ROM
      .PUBLIC     Calibrate
      .EXTRN      bSleepTime, rSleepCounter
```

```
;
; CONSTANT DECLARE
;     Bit definitions on port L
      SLEEP_OUT  = 2               ; Sleep Timer output pin
      SLEEP_IN   = 3               ; Sleep Timer input pin
;
;*****************************************************************************
;
Calibrate:
      LD    rSleepCounter, #00     ; Clear Sleep Counter
      RBIT  SLEEP_IN,  PORTLD      ; Make L3 a Hi-Z input
      RBIT  SLEEP_IN,  PORTLC
      RBIT  SLEEP_IN,  WKEN        ; Disable MIWU on L3
      RBIT  SLEEP_IN,  WKPND       ; Clear pending flag
      SBIT  SLEEP_OUT, PORTLC      ; Make L2 an output
      SBIT  SLEEP_OUT, PORTLD      ; Set it high to charge C1
CAL10:
      IFBIT SLEEP_IN   PORTLP      ; Is L3 high?
      JP    CAL15                  ;  Yes,break out of loop
      JP    CAL10                  ;  No, keep checking
CAL15:
      RBIT  SLEEP_OUT, PORTLC      ; Make L2 a Hi-Z input...
      RBIT  SLEEP_OUT, PORTLD      ; to allow C1 to discharge
CAL20:
      IFBIT SLEEP_IN,  PORTLP      ; Check L3
      JP    CAL20                  ; Wait until C1 hits MIWU threshold
      SBIT  SLEEP_OUT, PORTLC      ; Set L2 high
      SBIT  SLEEP_OUT, PORTLC      ; Begin Calibration Cycle, Charge C1
CAL30:
      IFBIT fbSecond,  fTimers     ; Has 1 second elapsed?
      JP    CAL40                  ;  Yes, increment counter
      JP    CAL50                  ;  No, continue
CAL40:
      RBIT  fbSecond,  fTimers     ; Clear flag
      LD    B, #rSleepCounter      ; Point to Sleep Counter
      LD    A, [B]                 ; Get count
      INC   A                      ; Increment it
      X     A, [B]                 ; Save it
CAL50:
      IFBIT SLEEP_IN,  PORTLP      ; Is C1 charged yet?
      JP    CAL60                  ;  Yes, now discharge it
      JP    CAL30                  ;  No, continue loop
CAL60:
      RBIT  SLEEP_OUT, PORTLC      ; Make L2 a Hi-Z input...
      RBIT  SLEEP_OUT, PORTLD      ; to allow C1 to discharge
CAL70:
      IFBIT fbSecond,  fTimers     ; Has 1 second elapsed?
      JP    CAL80                  ;  Yes, increment counter
      JP    CAL90                  ;  No, continue
CAL80:
      RBIT  fbSecond,  fTimers     ; Clear flag
      LD    B, #rSleepCounter      ; Point to Sleep Counter
      LD    A, [B]                 ; Get count
      INC A                        ; Increment it
      X     A, [B]                 ; Save it
CAL90:
      IFBIT SLEEP_IN,  PORTLP      ; Is C1 discharged yet?
      JP    CAL70                  ;  No, keep checking
      LD    B, #rSleepCounter      ; Yes, get Sleep Counter
      X     A, [B]
      LD    B, #bSleepTime
      X     A, [B]                 ; Save it in bSleepTime
CALEXIT:
      RET
```

8. 0 CONCLUSION

The external sleep timer described in this application note provides a method of conserving power by putting the COP8 into a low-power HALT mode for several seconds. Minimal I/O and software is required to implement this function. Depending on the application, the designer can vary the sleep period for optimal results. The accuracy of the components is shown to be non-critical since the timer can be calibrated before use. The accuracy of the calibration scheme is also up to the designer, since the resolution of the timer is under software control.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507