Implementing a Realtime Clock

Relevant Devices

This application note applies to the following devices:

C8051F000, C8051F001, C8051F002, C8051F005, C8051F006, C8051F007, C8051F010, C8051F011, and C8051F012.

Introduction

The purpose of this note is to provide an example of how to add a real-time clock (RTC) feature to a C8051F00x or C8051F01x device. Example software is included at the end of this note.

Key Points

- The external oscillator can be used to drive a crystal for the RTC while the system clock uses the high-frequency internal oscillator.
- The system clock can be derived from the internal or external oscillator, and can change sources without compromising the accuracy of the RTC.
- The RTC uses Timer 2, which is configured to increment on falling edges of an external input.
- Comparator 0 is used to convert the crystal waveform to a square wave.

Overview

Real-time clocks are used in many embedded applications to record the time at which an event occured, a pressure sensor was activated, or an ADC reading was taken, for example. Currently there are off-the-shelf components that contain a small crystal time base coupled with simple logic that have standardized interfaces for connecting to the I2C, SPI, or parallel port of a microcontroller.

This application note describes how to implement the function of a real-time clock inexpensively by using a C8051Fxxx device, a small 32 kHz watch crystal, and a few passive components.

Because the CPU overhead and resource requirements of the RTC are very small, this functionality can easily be added to an existing 8051-based system.

In this design, a 32 kHz watch crystal is connected to the external oscillator of the C8051 device. The output signal from the crystal oscillator is conditioned by one of the internal analog comparators and fed into a timer input. The timer is configured in auto-reload mode to generate an interrupt at a periodic rate, one-tenth second in this example. The interrupt service routine for the timer updates a series of counters for seconds, minutes, hours, and days.

Hardware Description

A schematic of the hardware is shown in Figure 1. This design uses an external 32kHz watch crystal as the time base for the RTC. This crystal is connected between the XTAL1 and XTAL2 pins of the device. Note that the external oscillator's crystal driver can be enabled while the CPU core is operating from the internal oscillator.

The XTAL2 output is fed into the (+) input of an on-chip analog comparator (Comparator 0). A low-pass filtered version of the XTAL2 signal is fed to the (-) input of the comparator to provide the DC bias level at which to detect the transitions of the oscillating signal. The corner frequency of this filter, where R = 1 MΩ and C = 0.022 µF, is substantially below the frequency of oscillation.

The output of the on-chip comparator is routed to an external GPIO pin (CP0, determined by the crossbar) and connected to the input signal of Timer 2 (T2, also determined by the crossbar). Timer 2 increments once for each falling edge detected at the T2 input.

Timer 2 is configured in 16-bit auto-reload mode to generate an interrupt every 3200 counts, or once every tenth of a second. The interrupt handler for Timer 2 updates a series of counters for tenths of seconds, seconds, minutes, hours, and days.

The default mode of the RTC implementation assumes that the CPU system clock (SYSCLK) is derived from the high-speed internal oscillator. When the system clock is changed to use the external 32kHz source, for example to save power, Timer 2 is switched by the software to use SYSCLK as its time base. Synchronizing the clock

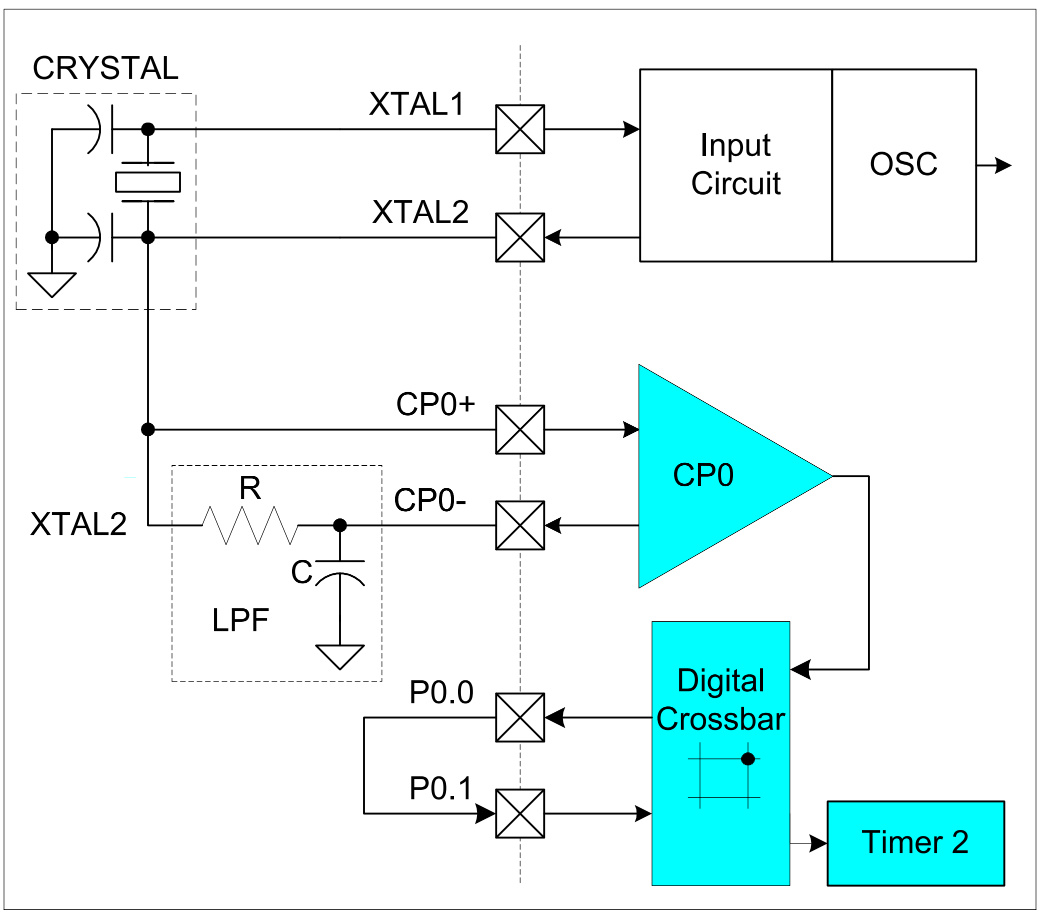

Figure 1. Connection Diagram

switching inside the RTC interrupt handler ensures no loss of accuracy.

Crossbar Configuration

The connection between internal digital peripherals and the GPIO pins is handled by the crossbar. In this design, the crossbar routes the CP0 output and T2 input to GPIO pins P0.0 and P0.1, respectively. It is important to note that the specific port pins used will change if peripherals with a higher crossbar priority are enabled (see AN001). Crossbar setup is accomplished with the following statements:

```
; enable CP0 outpput
mov XBR0, #80h
; enable T2 input
mov XBR1, #20h
; enable crossbar and weak pull-
; ups
mov XBR2, #40h
```

Oscillator Configuration

Refer to AN002 for details on configuring external oscillator. The following statement configures and enables the external oscillator for use with a 32 kHz crystal.

```
; enable external oscillator
; in 'crystal' mode; XFCN = 001
; for a 32kHz crystal
mov OSCXCN, #61h
```

Once configuration is complete, the external oscillator must be checked for stability before enabling the timer. The XTLVLD bit (OSCXN.7) is set when the crystal is running and stable. Software polls the XTLVLD bit before enabling Timer 2:

```
; wait until the external osc.
; is stable
WAIT:
mov ACC, OSCXCN
jnb ACC.7, WAIT

; enable Timer 2
setb TR2
```

Comparator Configuration

The Comparator 0 setup involves setting the positive and negative hysteresis and enabling the comparator. The comparator hysteresis can be configured in the comparator control register CPT0CN. Since the voltage of the XTAL2 signal will be fairly large (500 mV to 3 V), the CP0 hysteresis can be set high to provide noise immunity. The hysteresis is set and the comparator is enabled with the following statements:

```
; set CP0 hysteresis 10mV/10mV
mov CPT0CN, #0Ah

; enable CP0
orl CPT0CN, #80h
```

Timer Configuration

When the CPU system clock (SYSCLK) is derived from the high-frequency internal oscillator, Timer 2 is configured in auto-reload mode to count falling edges on the external signal T2. Timer 2 is configured with the following statement:

```
mov T2CON, #02h
```

We must also set the initial and reload values for Timer 2. The initial value is the value loaded into Timer 2 before it is enabled, and the reload value, held in RCAP2H (high byte) and RCAP2L (low byte), is loaded into Timer 2 after an overflow. The initial and reload values, which are identical, are determined by the precision required of the real-time clock. This design implements precision of a tenth of a second; therefore, Timer 2 is set to overflow every tenth of a second, or every 3200 counts of the 32 kHz time base. We set the COUNT value to 3,200, and set the reload values in the RCAP2 registers with the following commands:

```
;set T2 reload high byte
mov RCAP2H, #HIGH(-COUNT)

;set T2 reload low byte
mov RCAP2L, #LOW(-COUNT)
```

When Timer 2 overflows, it will be reloaded to overflow in another 3200 counts, and it will generate an interrupt. The program will vector to the Timer 2 interrupt service routine every tenth of a second to increment the counters. Because the interrupt service routine is short and is only called once every tenth of a second, CPU utilization is remarkably low.

Once Timer 2 is configured, its interrupt must be enabled with the following statement:

```
; enable Timer 2 interrupt
setb ET2
```

Timer 2 is enabled after all other timer configuration is complete by setting its run bit:

```
; start Timer 2
setb TR2
```

System Clock Switching

The default configuration of this RTC example assumes that the CPU system clock (SYSCLK) is derived from the high-speed internal oscillator. If SYSCLK is derived instead from the external oscillator, for power savings, the configuration for Timer 2 must be changed to use SYSCLK as the time base because signals at T2 can have a maximum frequency of SYSCLK / 4 in order to be properly detected.

The process for changing the system clock is as follows:

1. Stop the timer (TR2 = ‘0’).
2. Change timer time base.
3. Change SYSCLK time base.
4. Add correction factor to timer’s counter.
5. Start the timer (TR2 = ‘1’).

In order to guarantee that no external clock edges are missed, the SYSCLK should be updated in the RTC’s interrupt service routine.

The system clock can be changed by setting either SET_EXT_OSC (to change to the external oscillator) or SET_INT_OSC (to change to the internal oscillator) to ‘1’. These bits are used as flags in the Timer 2 ISR to permit changing of the system clock without sacrificing RTC accuracy. Details are given in the software description at the end of this report.

Software Description

This section contains a description of the software flow. The program listing begins on page 6.

Main Function

The MAIN function is used to configure the crossbar, external oscillator, comparator, and timer. First we setup the external crystal by enabling the external oscillator and setting the power factor bits.

The crossbar setup and CP0 setup values described above are then loaded, and then each are enabled. The crystal must be settled before Timer 2 is enabled. When the crystal is settled, the XTLVLD bit is set by hardware, and the program moves past the WAIT loop. At the end of the MAIN function the RTC_INIT function is called, Timer 2 is enabled, and global interrupts enabled.

RTC Initialization Function

The RTC_INIT function is used to reset the counter values and to configure Timer 2. This function can be used as a reset for the RTC. After clearing the counter values, the initial value for Timer 2 is set to the COUNT value as described in the configuration section. The COUNT value is also loaded into the reload registers (RCAP2H & RCAP2L). Timer 2 is then set to increment on external input edges, and the Timer 2 interrupt is enabled.

Timer Interrupt Service Routine

The Timer 2 ISR is called each time Timer 2 overflows (once every tenth of a second). When the ISR is called, it first clears the Timer 2 interrupt flag (TF2). The ISR then checks for overflows in all of the counters, starting with the tenths counter. If the tenths counter is at 9, it is reset to 0 and the seconds are checked for an overflow. Similarly, if the seconds are at 59, they are reset to 0, and the minutes are checked. The hours and days are checked in the same fashion. The counter is incremented, and then the oscillator selection bits (SET_EXT_OSC and SET_INT_OSC) are checked.

Oscillator Selection

If the SET_EXT_OSC bit is set, the bit is cleared, and the program jumps to the EXT_OSC label. First, OSCICN is checked--if the system clock is already using the external oscillator, the ISR exits. If not, Timer 2 is disabled to avoid any miscounts during the system clock switch. CKCON is setup so that the Timer 2 input clock is the system clock divided by one. Timer 2 is then set to increment on the system clock, and the Timer 2 counter register is updated to compensate for missed ticks during the SYSCLK transition. Between the system clock switch and the Timer 2 re-enable, Timer 2 misses 5 ticks. The correction value, EXT_COR, is set to 5; this value is added to the Timer 2 register before the system clock is switched to the external oscillator. After the switch, Timer 2 is enabled again, and the ISR exits.

If the SET_INT_OSC bit is set, the bit is cleared and the program jumps to the INT_OSC label. OSCICN is checked first to make sure the system clock is not already using the internal oscillator. If it is not, Timer 2 is disabled for the clock switch. The internal oscillator is selected as the system clock, and then the correction value, COR_INT is added to the Timer 2 register. In this case, 3 ticks are missed during the switch. COR_INT, which is set to 3, is added to Timer 2. The external input pin is selected as the Timer 2 input, and Timer 2 is enabled. The ISR then exits to wait for another overflow.

Counter Access

The tenths/seconds/minutes/etc counters can be accessed by calling the SAVE routine. The SAVE routine first saves the current state of the Timer 2 interrupt flag in the Carry bit and then disables the Timer 2 interrupt so that no interrupts occur during the save. Disabling the interrupt does no harm here because the interrupt will be enabled again at the end of the SAVE routine. If an interrupt is generated during the SAVE routine, it will be serviced as soon as the Timer 2 interrupt is enabled again. After ET2 is cleared, each counter is saved (TENTHS into STORE_T, SECONDS into STORE_S, etc). The interrupt flag is restored, and the function returns to its caller.

Software Example

```
;-----------------------------------------------------------------------------
;   CYGNAL, INC.
;
;
;   FILE NAME        : RTC_1.asm
;   TARGET DEVICE    : C8051F0xx
;   DESCRIPTION      : Software implementation of a real-time clock
;
;   AUTHOR           : JS
;
;   Software implementation of a real-time clock using a 32KHz crystal oscillator.
;   This program uses the crystal driver, XTAL2 to drive Comparator 0.  The positive
;   comparator input is from XTAL2, and the negative input is an averaged version of
;   XTAL2.  The averaging is done by a low pass filter. The output of Comparator 0
;   is routed to the Timer 2 input (T2).
;
;   Timer 2 is configured in auto-reload mode, and is set to trigger on
;   the external input pin connected to the Comparator 0 output.
;
;   This code assumes the following:
;
;   (1)    An external oscillator is connected between XTAL1 and XTAL2
;   (2)    A low pass averaging filter is connected bewteen XTAL2 and CP0-
;   (3)    XTAL2 is routed to CP0+
;   (4)    CP0 output is routed to Timer 2 input through the port pins assigned
;          by the crossbar
;
;   For a 32KHz crystal, the low pass filter consists of a 0.022uF capacitor and a
;   1 Mohm resistor.
;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------
; EQUATES
;-----------------------------------------------------------------------------

   $MOD8F000

; Count value: This value is used to define what is loaded into timer 2 after each
; overflow.The count value is 3200, meaning the timer will count 3200 ticks before an
; overflow.  Used with the 32KHz crystal, this means the timer will overflow every
; tenth of a second.

   COUNT     EQU    3200d                      ; count value

; Compensation factors for system clock switching used to update Timer 2 after a
; system clock change

   EXT_COMP  EQU    5d
   INT_COMP  EQU    3d

;-----------------------------------------------------------------------------
; VARIABLES
;-----------------------------------------------------------------------------

DSEG
```

```
   org 30h

   TENTHS:   DS     1                           ; counts tenths of seconds
   SECONDS:  DS     1                           ; counts seconds
   MINUTES:  DS     1                           ; counts minutes
   HOURS:    DS     1                           ; counts hours
   DAYS:     DS     1                           ; counts days

   STORE_T:  DS     1                           ; storage byte for tenths,
                                                ; used by SAVE routine
   STORE_S:  DS     1                           ; storage byte for seconds
   STORE_M:  DS     1                           ; minutes
   STORE_H:  DS     1                           ; hours
   STORE_D:  DS     1                           ; days

BSEG

   org    00h

SET_EXT_OSC: DBIT   1                           ; flag to change system clock
                                                ; to external osc
SET_INT_OSC: DBIT   1                           ; flag to change system clock
                                                ; to internal osc

;-----------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-----------------------------------------------------------------------------

CSEG

; Reset Vector
   org    00h
   ljmp   MAIN

; Timer 2 ISR Vector
   org    2Bh
   ljmp   T2_ISR                                ; Timer 2 ISR

;-----------------------------------------------------------------------------
; MAIN PROGRAM
;-----------------------------------------------------------------------------

   org    0B3h

MAIN:

   mov    OSCXCN, #61h                          ; enable external oscillator
                                                ; in ‘crystal’ mode for a
                                                ; 32kHz watch crystal

   mov    WDTCN, #0DEh                          ; disable watchdog timer
   mov    WDTCN, #0ADh

   ; Setup Crossbar
   mov    XBR0, #80h                            ; enable CP0 output
   mov    XBR1, #20h                            ; enable T2 input
   mov    XBR2, #40h                            ; enable crossbar

   ; Setup Comparator 0
```

```
   mov    CPT0CN, #08h                       ; set positive hysteresis to 10mV
   orl    CPT0CN, #02h                       ; set negative hysteresis to 10mV
   orl    CPT0CN, #80h                       ; enable CP0

   acall  RTC_INIT                           ; Initialize RTC and Timer 2

WAIT:

   mov    ACC, OSCXCN                        ; wait until the external
                                             ; oscillator is steady
   jnb    ACC.7, Wait                        ; by checking the XTLVLD bit
                                             ; in OSCXCN

   setb   TR2                                ; turn on Timer 2 (starts RTC)

   setb   EA                                 ; enable global interrupts

   jmp    $                                  ; spin forever

;-----------------------------------------------------------------------------------
;  Initialization Subroutine
;-----------------------------------------------------------------------------------

RTC_INIT:

   ; Clear all counters
   mov    TENTHS, #0
   mov    SECONDS, #0
   mov    MINUTES, #0
   mov    HOURS, #0
   mov    DAYS, #0

   ; Setup Timer2 in auto-reload mode to count falling edges on external T2

   mov    TH2, #HIGH(-COUNT)                 ; set initial value for timer 2
   mov    TL2, #LOW(-COUNT)

   mov    RCAP2H, #HIGH(-COUNT)              ; set reload value for timer 2
   mov    RCAP2L, #LOW(-COUNT)

   mov  T2CON, #02h                          ; configure Timer 2 to increment
                                             ; falling edges on T2
   setb   ET2                                ; enable Timer 2 interrupt
   ret

;-----------------------------------------------------------------------------------
; Timer 2 ISR
;-----------------------------------------------------------------------------------

T2_ISR:
   clr    TF2                                ; clear overflow interrupt flag
   push   PSW                                ; preserve PSW (carry flag)
   push   ACC                                ; preserve ACC

   ; Check for overflows
   mov    A, TENTHS
   cjne   A, #9d, INC_TEN                    ; if tenths less than 9, jump
                                             ; to increment
   mov    TENTHS, #0                         ; if tenths = 9, reset to zero,
```

```
                                              ; and check seconds

   mov    A, SECONDS
   cjne   A, #59d, INC_SEC                    ; if seconds less than 59, jump
                                              ; to increment
   mov    SECONDS, #0                         ; if seconds = 59, reset to zero,
                                              ; and check minutes

   mov    A, MINUTES
   cjne   A, #59d, INC_MIN                    ; if minutes less than 59, jump
                                              ; to increment
   mov    MINUTES, #0                         ; if minutes = 59, reset to zero,
                                              ; and check hours

   mov    A, HOURS
   cjne   A, #23d, INC_HOUR                   ; if hours less than 23, jump
                                              ; to increment
   mov    HOURS, #0                           ; if hours = 23, reset to zero,
                                              ; and check days

   inc    DAYS                                ; DAYS will roll over after 255

   jmp    CHECK_OSC                           ; jump to check for oscillator
                                              ; change request

;Increment counters----------------------------------------------------------------

INC_TEN:
   inc    TENTHS                              ; increment tenths counter
   jmp    CHECK_OSC                           ; jump to check for oscillator
                                              ; change request

INC_SEC:
   inc    SECONDS                             ; increment seconds counter
   jmp    CHECK_OSC                           ; jump to check for oscillator
                                              ; change request

INC_MIN:
   inc    MINUTES                             ; increment minutes counter
   jmp    CHECK_OSC                           ; jump to check for oscillator
                                              ; change request

INC_HOUR:
   inc    HOURS                               ; increment hours counter
   jmp    CHECK_OSC                           ; jump to check for oscillator
                                              ; change request

;Oscillator changes----------------------------------------------------------------

CHECK_OSC:
   jbc    SET_EXT_OSC, EXT_OSC                ; check for external oscillator
                                              ; select
   jbc    SET_INT_OSC, INT_OSC                ; check for internal oscillator
                                              ; select
   jmp    END_ISR                             ; exit

EXT_OSC:                                      ; switch system clock to
                                              ; external oscillator
   mov    ACC, OSCICN                         ; check current system clock
```

```
    jb      ACC.3, END_ISR                        ; exit if already using external
                                                  ; oscillator

    orl     CKCON, #20h                           ; select system clock (divide by 1)
                                                  ; for Timer 2
    clr     TR2                                   ; disable Timer 2 during clock change
    clr     CT2                                   ; select SYSCLK as Timer 2 input

    mov     A, #LOW(EXT_COR)                      ; load correction value into
                                                  ; accumulator
    add     A, TL2                                ; add correction value to Timer 2
                                                  ; counter register
    mov     TL2, A                                ; store updated Timer 2 value

    orl     OSCICN, #08h                          ; select external oscillator as
                                                  ; system clock
    setb    TR2                                   ; enable Timer 2 after clock change

    jmp     END_ISR                               ; exit

INT_OSC:                                          ; switch system clock to internal
                                                  ; oscillator

    mov     ACC, OSCICN                           ; check current system clock
    jnb     ACC.3, END_ISR                        ; exit if already using internal
                                                  ; oscillator

    clr     TR2                                   ; disable Timer 2 during clock change
    anl     OSCICN, #0f7h                         ; select internal oscillator as
                                                  ; system clock

    mov     A, #LOW(INT_COR)                      ; load correction value into
                                                  ; accumulator
    add     A, TL2                                ; add correction value to Timer 2
                                                  ; register
    mov     TL2, A                                ; store updated Timer 2 value

    setb    CT2                                   ; select external Timer 2 input
    setb    TR2                                   ; enable Timer 2 after clock change

    jmp     END_ISR                               ; exit

END_ISR:

    pop     ACC                                   ; restore ACC
    pop     PSW                                   ; restore PSW
    reti

;-----------------------------------------------------------------------------------
; Counter Save Routine
;-----------------------------------------------------------------------------------

SAVE:

    mov     C, ET2                                ; preserve ET2 in Carry

    clr     ET2                                   ; disable Timer 2 interrupt
                                                  ; during copy
```

```
    mov     STORE_T, TENTHS                     ; copy all counters
    mov     STORE_S, SECONDS
    mov     STORE_M, MINUTES
    mov     STORE_H, HOURS
    mov     STORE_D, DAYS

    mov     ET2, C                              ; restore ET2
    ret

;---------------------------------------------------------------------------------

END
```

Contact Information

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Email: productinfo@silabs.com
Internet: www.silabs.com