Detailed Explanation of `BlinkingLEDUsingTimer0WithOVFInterrupt.asm` Code

In a previous note, entitled *Detailed Explanation of `BlinkingLEDUsingTimer0.asm` Code*, I described ATmega328P assembly code which causes an LED to blink when the timer rolls over to zero. In this note, I describe how an interrupt can be used to jump to code that causes the blinking; the result is same (except for an added blinking pin), but studying this code helps one understand how interrupts work in the ATmega328P. This is the code `BlinkingLEDUsingTimer0WithOVFInterrupt.asm` . I will assume that the reader has studied and understood the previous note; only new features will be described here.

Walking through the assembly source code

The `BlinkingLEDUsingTimer0WithOVFInterrupt.asm` code is in the directory `/home/pi/Programming/Assembly/BlinkingLED/` . When assembled using `avra -l BlinkingLEDUsingTimer0WithOVFInterrupt.lst BlinkingLEDUsingTimer0WithOVFInterrupt.asm` , the file `BlinkingLEDUsingTimer0WithOVFInterrupt.lst` contains the following (labels are shown in bold face):

```
AVRA   Ver. 1.3.0 BlinkingLEDUsingTimer0WithOVFInterrupt.asm Tue Jun 23 10:54:12 2015


          .LIST
          .DEF   overflowCount    = R22
          .DEF   statusFlags      = R23
          .org   0x0000
C:000000 940c 0034       jmp      RESET
C:000002 9518            reti             ; EXT_INT0
C:000003 0000            nop
C:000004 9518            reti             ; EXT_INT1
C:000005 0000            nop
C:000006 9518            reti             ; PCINT0
C:000007 0000            nop
C:000008 9518            reti             ; PCINT1
C:000009 0000            nop
C:00000a 9518            reti             ; PCINT2
C:00000b 0000            nop
C:00000c 9518            reti             ; WDT
C:00000d 0000            nop
C:00000e 9518            reti             ; TIM2_COMPA
C:00000f 0000            nop
C:000010 9518            reti             ; TIM2_COMPB
C:000011 0000            nop
C:000012 9518            reti             ; TIM2_OVF
C:000013 0000            nop
C:000014 9518            reti             ; TIM1_CAPT
C:000015 0000            nop
C:000016 9518            reti             ; TIM1_COMPA
C:000017 0000            nop
C:000018 9518            reti             ; TIM1_COMPB
C:000019 0000            nop
C:00001a 9518            reti             ; TIM1_OVF
C:00001b 0000            nop
C:00001c 9518            reti             ; TIM0_COMPA
C:00001d 0000            nop
C:00001e 9518            reti             ; TIM0_COMPB
C:00001f 0000            nop
C:000020 940c 004b       jmp      TIM0_OVF; TIM0_OVF
C:000022 9518            reti             ; SPI_STC
C:000023 0000            nop
C:000024 9518            reti             ; USART_RXC
C:000025 0000            nop
C:000026 9518            reti             ; USART_UDRE
C:000027 0000            nop
C:000028 9518            reti             ; USART_TXC
C:000029 0000            nop
C:00002a 9518            reti             ; ADC
C:00002b 0000            nop
C:00002c 9518            reti             ; EE_RDY
C:00002d 0000            nop
C:00002e 9518            reti             ; ANA_COMP
C:00002f 0000            nop
C:000030 9518            reti             ; TWI
C:000031 0000            nop
C:000032 9518            reti             ; SPM_RDY
C:000033 0000            nop

         ;**********************  Reset Handler  **********************

          RESET:                          ; Program initialization
C:000034 e008      ldi   r16,high(RAMEND) ; Initialize stack pointer
C:000035 bf0e      out   SPH,r16
C:000036 ef0f      ldi   r16,low(RAMEND)
```

```
C:000037 bf0d      out   SPL,r16


         ; Have timer/counter-0 cause a toggle of the state of OC0A (PIND6) each time the count
         ;    becomes 0.
         ; Also, use the timer's overflow interrupt to count interrupts in a byte-sized variable,
         ;    and then to toggle PIND3 each time that count wraps around to zero.
         ; System clock, by default, is the calibrated RC oscillator operating at 8 MHz.
         ; The pre-scaler, by default, is set to divide by 8 so ClkIO is 1 uS.
         ; The timer/counter-0 divider is set to divide by 64 so it will tick every 64 uS
         ; Since this is an 8-bit counter it will rollover every 256*64=16384 uS and toggle PIND6.
         ; The overflow interrupt will be triggered and the overflow interrupt handler will toggle PIND3
         ;    each time it rolls over, every 256*256*64=4194304 uS or about every 4 seconds.
         ;    If the overflow count rolls over PIND3 will be toggled.  To see it change,
         ;    connect PD3 to Buf2 on the Gertboard, and set the output jumper on B2.
         ; The LED will then have a cycle time of 2*256*256*64 us = 8388608 uS

C:000038 e402      ldi   r16,(0<<COM0A1)|(1<<COM0A0)|(0<<COM0B1)|(0<<COM0B0)|(1<<WGM01)|(0<<WGM00)
C:000039 bd04      out   TCCR0A,r16               ; Set up the CTC (Clear Timer on Compare Match) mode.
C:00003a ef0f      ldi   r16,0xFF
C:00003b bd07      out   OCR0A,r16                ; Set the compare value to 0xFF


C:00003c 9a56      sbi   DDRD,DDD6                ; Prepare PIND6 to be an output to drive LED.
C:00003d 9a53      sbi   DDRD,DDD3                ; Prepare PIND3 to be an output to drive another LED

C:00003e e003      ldi   r16,(0<<FOC0A)|(0<<FOC0B)|(0<<WGM02)|(0<<CS02)|(1<<CS01)|(1<<CS00)
C:00003f bd05      out   TCCR0B,r16               ; Set the timer clock divider to ClkIO/64

C:000040 e001      ldi   r16,(0<<OCIE0B)|(0<<OCIE0A)|(1<<TOIE0)
C:000041 9300 006e sts   TIMSK0,r16               ; Enable overflow interrupt for timer/counter-0

         ; To minimize power consumption, disable power to all subsystems except the timer/counter-0.
         ; Note: Default is 0x00, all subsystems powered.
C:000043 ec0f      ldi     r16,(1<<PRTWI)|(1<<PRTIM2)|(0<<PRTIM0)|(1<<PRTIM1)|(1<<PRSPI)|(1<<PRUSART0)|(1<<PRADC)
C:000044 9300 0064 sts     PRR,r16

C:000046 9478      sei                            ; Enable interrupts globally

C:000047 e001      ldi   r16,(0<<SM2)|(0<<SM1)|(0<<SM0)|(1<<SE)    ; Go to ADC noise-reduction sleep
C:000048 bf03      out   SMCR,r16
          REPEAT:
C:000049 9588      SLEEP                          ; Sleep until interrupt occurs
C:00004a cffe      rjmp REPEAT                    ; After interrupt is handled, go back to sleep

         ;**********************  Timer0 Overflow Handler  **********************

          TIM0_OVF:
C:00004b b77f      in    statusFlags,SREG

C:00004c 9563      inc   overflowCount
C:00004d f409      brne  END_TIM0_OVF
C:00004e 9a4b      sbi   PIND,PIND3

          END_TIM0_OVF:
C:00004f bf7f      out   SREG,statusFlags
C:000050 9518      reti


Segment usage:
   Code      :        81 words (162 bytes)
   Data      :         0 bytes
   EEPROM    :         0 bytes

Assembly completed with no errors.
```

Section 12 (pages 57-70) of the ATmega328P datasheet gives the detailed specifications concerning interrupt usage. When interrupts are used, one must establish an interrupt jump table at the start of memory. The format for this table requires 2 words (1 word = 2 bytes) for each of the 26 possible interrupts (See Table 14.1, page 57). Each entry is either a `jmp` <label> requiring both words of memory, or a `nop` (no operation) requiring 1 word followed by a `reti` instruction (return from interrupt) requiring another word of memory. The comments in the interrupt table give the name of the interrupt corresponding to each location, most of which are inactive.

The very first instruction `jmp RESET` is executed when the reset pin (pin 1) goes from low to high. This happens when power is applied to the processor and also just after a program has been loaded into its program memory. That `jmp` instruction (`0x940c`) causes the program counter register named `PC` to be loaded with the address (`0x0034`) of the `RESET` label found in the code after the interrupt table.

The code from word `0x0034` to `0x0048` sets up the timer registers and sleep registers, turns on the interrupts with an `sei` instruction, and then goes to sleep. When an interrupt occurs, it will wake up, disable further interrupts, check the interrupt jump table for where to find the interrupt handling routine (in this case TIM0_OVF at word address 0x004b), save the current value of the PC on the stack, and load the PC with that address. When it reaches a reti (return from interrupt) instruction, reloads the PC with the value that had been saved on the stack, enables interrupts again, executes the relative jump instruction (rjmp REPEAT) at word address 0x004a), and goes back to sleep.

The ldi instruction at word address 0x0040 sets bit TOIE0 (timer overflow interrupt enable for timer 0) in the register TIMSK0 (timer mask for timer 0) so that later when the sei instruction enables the interrupt system, the timer interrupt can be triggered.

The in statusFlags,SREG instruction at word address 0x004b and the out SREG,statusFlags instruction at word address 0x004f are necessary at the start and end of every interrupt routine. These save the flag values at the start and restore them at the end of the interrupt routine so that the code that was running when the interrupt occurred has its correct flag values. The statusFlags variable refers to general-purpose register r23 as defined at the start of the code.

A variable overflowCount (held in r22) is incremented at word address 0x004c in the overflow interrupt handling code thereby counting overflows. Code at word address 0x003d sets PIND3 for output and that pin gets toggled at word address 0x004e when the overflowCount value wraps around to zero. An LED attached to PIND3 will therefore blink at a 256-times slower rate than one attached to PIND6.