```
;*****************************************************
; timer_ex1.asm  24 Mar 2011    Rev. 1
;
; Original Author:  Adam Mills
; Editted by:       Dr. Schwartz
;*****************************************************
;* Flashes the LED on our board.  Example using Timer1.
;*****************************************************
GPAMUX1     .set    0x6F86
GPATOGGLE   .set    0x6FC6
GPADIR      .set    0x6F8A
GPADAT      .set    0X6FC0
INT13_VECT  .set    0x00001A    ;Vector for Timer1 interrupt, VMAP = 0
TIMREGPG    .set    0xC00>>6    ;start of timer register page
TIMER1PRD   .set    0x0C0A      ;holds period of counter
TIMER1PRDH  .set    0x0C0B      ;holds period of counter [high]
TIMER1TCR   .set    0x0C0C      ;timer control register
TIMER1PR    .set    0x0C0E      ;msb current prescaler count, prescale amount [low bytes]
TIMER1PRH   .set    0x0C0F      ;msb current prescaler count, prescale amount [high bytes]
PERIODL     .set    0xFFFF      ;16-bit lower count value
PERIODH     .set    0x000F      ;16-bit upper count value
PRESCALEL   .set    0x3         ;8-bit lower prescale value
PRESCALEH   .set    0x00        ;8-bit upper prescale value

            .global _c_int00    ;This assembler directive allows _c_int00 to be a
                                ;global variable. This tells the linker where your
                                ;program (.text code) begins and where to boot fro
;*******************************************************************************
.text   ;program section, see the command linker file, program code
        ;should be placed in the text section which starts at 0x9000

_c_int00:
    LC INIT_CPU         ;initialize CPU (objmode, disable watchdog)
    LC INIT_OUTPUT0     ;initialize LED (output, mux = 00)
    LC INIT_TIMER1_INT  ;intitialize timer (PRS = 0x132, TIM = 0xF FFFF, enable
interrupts)
    OR IER,#0x1000      ;enable INT13 (bit 12 in ier) must be done with an OR
    CLRC INTM           ;enable global interrupts (by clearing the global interrupt mask)

WAIT:
    B WAIT, UNC         ;wait for interrupt

INIT_CPU:
    PUSH DP
    SETC OBJMODE        ;allow 32 bit mov instructions
    EALLOW              ;allow write access to GPIO regs
    MOVZ DP,#0x7029>>6  ;turn off that pesky watchdog timer
    MOV @7029h,#0x68
    POP DP
    LRET

INIT_OUTPUT0:
    PUSH AR0
    PUSH AR1
    MOV AR0,#0          ;set Mux for I/O purposes
    MOV AR1,#GPAMUX1
    MOV *AR1,AR0
    MOV AR0,#0x1        ;set GPIO0 as an output
    MOV AR1,#GPADIR
    MOV *AR1,AR0
    POP AR1
    POP AR0
```

```
    LRET

INIT_TIMER1_INT:
    PUSH DP                     ;save neede register
    CLRC VMAP                   ;set interrupt vectors to beginning of memory map
                                ;  instead of the end. This allows user to write the
                                ;  address of the isr directly into the interrupt vector,,
                                ;  since this is ram instead of rom (which is at the end)
    MOV DP, #TIMREGPG           ;set data page to timer registers
    MOV @TIMER1PRD, #PERIODL    ;set period
    MOV @TIMER1PRDH, #PERIODH   ;  ...
    MOV @TIMER1PR, #PRESCALEL   ;set prescaler
    MOV @TIMER1PRH, #PRESCALEH  ;  ...
    TSET @TIMER1TCR, #14        ;set bit 14 to enable Timer1
    TSET @TIMER1TCR, #5         ;load prescaler and period
    MOVL XAR0, #INT13_VECT      ;need to load interrupt vector with isr address
    MOV *XAR0++, #TIMER1_ISR    ;load lower byte (little endian)
    MOV *XAR0, #TIMER1_ISR>>16  ;load upper byte
    POP DP                      ;restore register
    LRET

; several registers auto-saved, including dp, ar1 and ar0
TIMER1_ISR:
    MOV AR1, #GPATOGGLE         ;load AR1 with address of GPIO0
    MOV *AR1, #1                ;toggle  GPIO0
    MOV DP, #TIMREGPG           ;set DP for direct addressing
    TSET @TIMER1TCR, #15        ;clear timer interupt flag, int13 flag auto-clears
    IRET                        ;registers reloaded
```