AVR Freaks

DESIGN NOTE #009

KEYWORDS:
ATTINY15, HIGH-SPEED TIMER, HIGH FREQUENCY PWM OUTPUT

Using the ATtiny15 High-speed Timer

Introduction

ATtiny15 includes a High-speed Timer with high frequency PWM output. This design note describes how to use the High-speed Timer to generate PWM signals with variable duty cycle and frequency.

Overview

The ATtiny15's High-speed Timer is an 8-bit timer with two compare registers. Compare Register B is the top value for the PWM. This value decides the frequency (and accuracy) of the PWM. A Compare Register B value of 255(0xff) sets a frequency of 100 kHz and 8-bit resolution with 25.6 MHz timer. If Compare Register B value is 127 (0x7f), the frequency will be 200 kHz, and the resolution 7-bit.

Compare Register A is used to control the duty cycle of the PWM. The duty cycle can be set between 0 (constant low) and Compare Register B (constant high).

The sample program below adjusts the PWM frequency by pressing PB3, and adjusts the PWM cycle by pressing PB4. A timer interrupt is used to check the status of the input pins and adjusting the PWM frequency and duty cycle every 160 ms.

```
;***** Sample program for PWM operation timer/counter 1
; Increment OCR1A value with PB2, increase PWM duty cycle
; Decrement OCR1B value with PB3. increase PWM frequency

.include "tn15def.inc"

.def    zero        = R1
.def    status      = R2    ; Storage of SREG in interrupt
.def    temp        = R16   ; Temporary register
.def    duty_cycle  = R16   ; Keeps track of duty cycle
.def    frequency   = R17   ; Keeps track of frequency

.cseg
    rjmp    RESET           ;Reset Handle. Program execution starts here
.org OVF0addr
    rjmp    TIM0_OVF        ;Timer/Counter0 Overflow Handle

;*********************************
;*
```

```
;* Timer0 overflow Interrupt Routine
;*
;*********************************
TIM0_OVF:
    in      status,SREG          ; Store status register
    sbis    PINB,PB2             ; Check if PB2 is pushed low
    inc     duty_cycle           ; If PB2 is pushed low, increase PWM duty
cycle
    cp      duty_cycle,frequency ; If duty_cycle is 100% -
    brcs    n0
    ldi     duty_cycle,0x00      ; - Set duty cycle to 0%
n0: sbis    PINB,PB3             ; Check if PB3 is pushed low
    dec     frequency            ; If PB3 is pushed low, decrease PWM
frequency
    cp      frequency,zero       ; If frequency is maximum -
    brne    n1
    ldi     frequency,0xff       ; - Reset to minimum frequency
n1: out     OCR1A,duty_cycle     ; Set duty_cycle
    out     OCR1B,frequency      ; Set frequency
    sbic    PINB,PB0             ; PB0 blinks to indicate running interrupt
    cbi     PORTB,PB0
    sbis    PINB,PB0
    sbi     PORTB,PB0
    out     SREG,status          ; Restore Status register
    reti

; Program Execution Starts Here
;*****************************************************************************

RESET:
    clr     zero
    ldi     temp, 0x03
    out     DDRB, temp           ; PB1 - outputs, PB0,2,3,4 inputs
    ldi     temp, 0xfe
    out     PORTB, temp          ; Internal pull-ups enabled
    ldi     temp, 0x04           ; Timer0 used for checking if buttons are
pushed
    out     TCCR0, temp          ; Timer0, clk/256, generate interrupt every
40ms
    ldi     temp, (1<<TOIE0)     ; Enable Timer0 interrupt
    out     TIMSK, temp
    ldi     temp, 0x61           ; Timer1 in high speed PWM mode
    out     TCCR1, temp          ; Timer1, Non-Inverted PWM, CLK*16
    ldi     duty_cycle, 0x00     ; Initialize duty_cycle to 0
    ldi     frequency, 0xff      ; Initialize frequency to 0xff
    sei                          ; Enable interrupts

e_loop:
    rjmp e_loop              ; Eternal loop, interrupted by timer0
```