AVR Freaks

DESIGN NOTE #046

AUTHOR: MICHAEL KLOKOV

KEYWORDS: TIMER, INTERRUPT, ACCURACY

Timer Interrupt with a Maximum Accuracy

Introduction

This document describes a way to execute any time-dependent routine in the proper time exactly using simple Timer Overflow Interrupt. This method can be used for time-critical applications like precision signal generation or PLL.

Overview

Usually, a Timer Interrupt handler does not start immediately after Timer Overflow occurs. The AVR-core needs some time to finish current instruction. This time can vary from zero to three cycles depending on current instruction and the moment when Timer Overflow occurs.

It can be a problem if you want some code to be executed in the certain moment exactly. There are some ways to solve this:

- For a VERY simple single-task program you can precalculate all the instruction execution times. This way you can write any time-critical routines even without any timer or interrupt.
- You can use SLEEP instruction to make AVR enter Idle mode. Then, when the Timer Overflow occurs it will awake AVR to execute the interrupt handler. In this case AVR-core does not have any instruction to finish, so the interrupt handler will be executed in the proper time exactly. This method is most useful for single-task applications. In cases when AVR has to execute some “background” routine, it would be difficult to enter Idle mode “just before” Timer Interrupt occurs.
- Finally, you can read the Timer to find out how much time passed after overflow occurred. You can compensate the time error executing some extra instructions before your critical code.

Code

Let's say, you need to generate 10 cycles length pulse every 100 cycles. You also need to execute some "background" routine. The code is written in AVR Studio® for AT90S2313, but can be used for any AVR MCU (not FPSLIC).

Note that you can write any complex "background" routine using any tools (like AVRGCC) if it does not use CLI instruction and other interrupt(s) that can block your time-critical task.

The color marked instructions compensate the time error.

```
.include "2313def.inc"
.org 0
  rjmp    Reset
.org OVF0addr
  rjmp    OVF0

Reset:
  ldi     r16,0xDF            ; Stack pointer setup
  out     SPL,r16
  ldi     r16,1               ; Pin D0 - output
  out     DDRD,r16
  ldi     r16,2               ; Timer interrupt setup
  out     TIFR,r16
  out     TIMSK,r16
  ldi     r16,201             ; Starting timer value
  out     TCNT0,r16
  ldi     r16,1               ; Prescaler = 1 (OSC frequency)
  out     TCCR0,r16
  sei                         ; Enable interrupt

ethernal:                     ; "Background" routine
  nop                         ; 1 It's a sample routine
  rcall   _ret                ; 3 Instructions with different execution times
  rjmpethernal; 2
_ret:ret; 4

OVF0:                         ; ! Here in the "accidental" time
                              ; TCNT0 varies from 6 to 9
  push    r0                  ; 2 Save registers
  push    r16                 ; 2
  in      r16,TCNT0           ; 1 ==== THE REAL DEAL ====  Fix the time
  sbrc    r16,1               ; 1/2
  lpm                         ; 3 (lpm r0,Z for some AVR parts)
  sbrs    r16,0               ; 1/2
  rjmp    _next               ; 2
_next:                        ; ! Here in the fixed time
                              ; TCNT0 = 0x12 always
  ldi     r16,0xb0            ; 1 Reload the Timer0 (0 - 100 + 0x12 + 2 )
  out     TCNT0,r16           ; 2
```

```
sbi     PORTD,0             ; Start pulse
nop                         ; Wait 10 cycles
nop
nop
nop
nop
nop
nop
nop
cbi     PORTD,0             ; Stop pulse

pop     r16                 ; Restore registers
pop     r0
reti
```

Here is the beginning of simulator port logging file generated by AVR Studio.

```
000000000:00
000000090:01
000000100:00
000000190:01
000000200:00
000000290:01
000000300:00
000000390:01
000000400:00
   . . .
```