ATmega1284 Timer0/Counter0 Interrupt

Costas Foudas, University of Ioannina, Physics Department, F3.303

The Need for Processor Interrupts

Up to now if you wanted to do something in a program as soon as a bit was set (key pressed, bit set in a register, voltage exceeded a given threshold,…) you had to keep reading the bit till it changed !

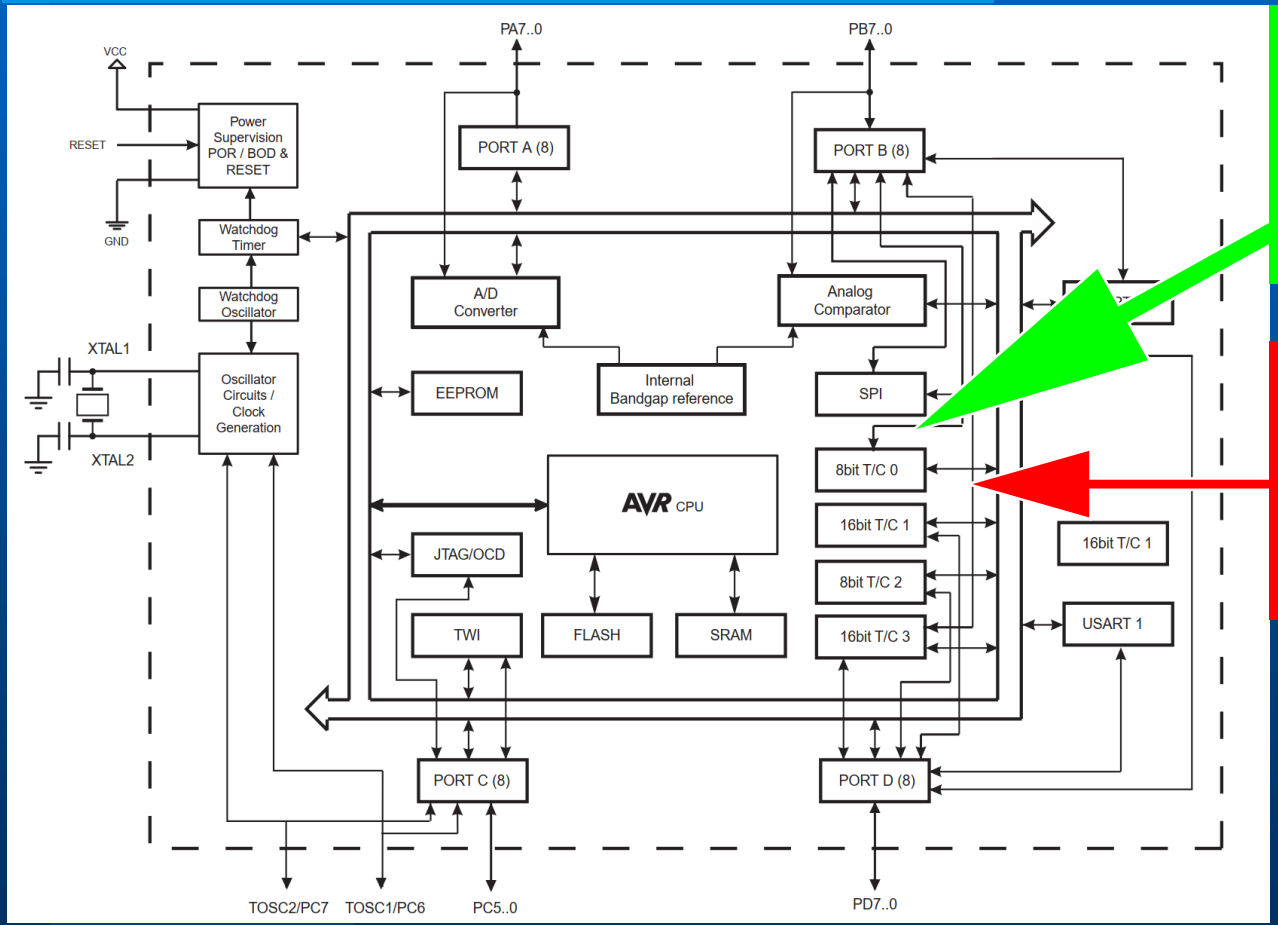
This is clearly not an efficient was of doing it…..

This is why the interrupts were introduced and are associated with bits that users tend to use frequently

Processor Interrupts

Interrupts are subroutine calls initiated not by an *rcall* command but by hardware signals. These hardware calls direct your program to the interrupt service routine which is executed. At the end your program returns where it was when the interrupt occurred.

The ATmega1284P Timers and Interrupts



A device that counts time (Timer) could be one of the reasons for interrupt (internal)

34 different reasons to interrupt the CPU and execute an interrupt service routine.

Seven External Interrupt Sources

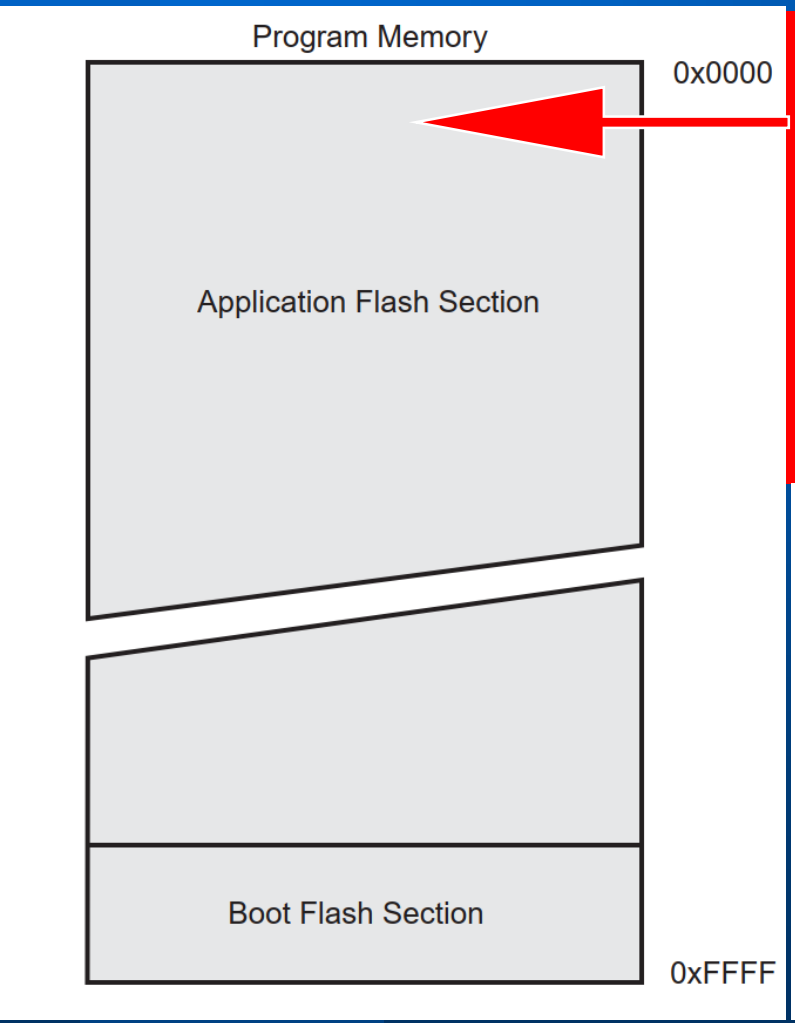
Status Register Interrupt Bit:

D7	D6	D5	D4	D3	D2	D1	D0
I	T	H	S	V	N	Z	C
★		SREG		?	?	?	?

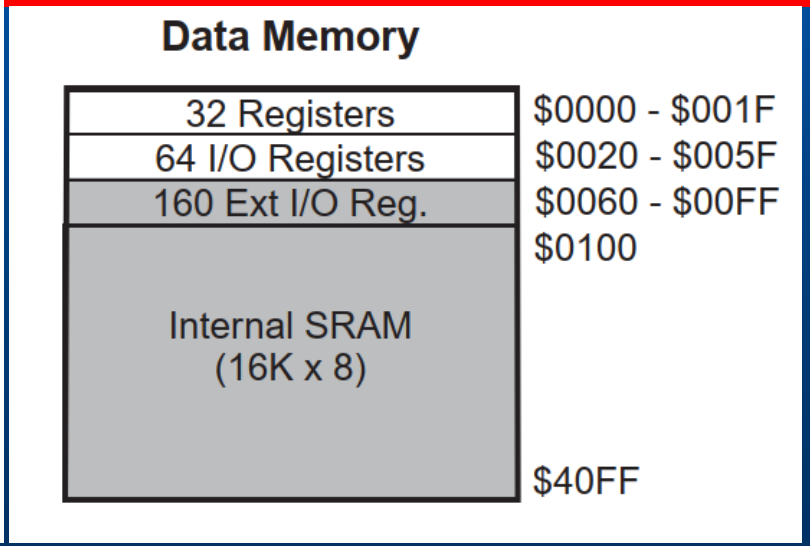
By now you should know what The V,N,Z,C bits are.

In order to use any interrupts on ATmega103 you must set the ★ bit using the command *sei*

The ATmega1284P Memory Map



The first 35 2-byte addresses in the program (00-44$_H$) memory are dedicated to interrupts



Interrupt vector addresses in the Program Memory

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	PCINT0	Pin Change Interrupt Request 0
6	$000A	PCINT1	Pin Change Interrupt Request 1
7	$000C	PCINT2	Pin Change Interrupt Request 2
8	$000E	PCINT3	Pin Change Interrupt Request 3
9	$0010	WDT	Watchdog Time-out Interrupt
10	$0012	TIMER2_COMPA	Timer/Counter2 Compare Match A
11	$0014	TIMER2_COMPB	Timer/Counter2 Compare Match B
12	$0016	TIMER2_OVF	Timer/Counter2 Overflow
13	$0018	TIMER1_CAPT	Timer/Counter1 Capture Event
14	$001A	TIMER1_COMPA	Timer/Counter1 Compare Match A
15	$001C	TIMER1_COMPB	Timer/Counter1 Compare Match B
16	$001E	TIMER1_OVF	Timer/Counter1 Overflow
17	$0020	TIMER0_COMPA	Timer/Counter0 Compare Match A
18	$0022	TIMER0_COMPB	Timer/Counter0 Compare match B
19	$0024	TIMER0_OVF	Timer/Counter0 Overflow
20	$0026	SPI_STC	SPI Serial Transfer Complete
21	$0028	USART0_RX	USART0 Rx Complete
22	$002A	USART0_UDRE	USART0 Data Register Empty
23	$002C	USART0_TX	USART0 Tx Complete
24	$002E	ANALOG_COMP	Analog Comparator
25	$0030	ADC	ADC Conversion Complete
26	$0032	EE_READY	EEPROM Ready
27	$0034	TWI	2-wire Serial Interface

Vector No.	Program Address(2)	Source	Interrupt Definition
28	$0036	SPM_READY	Store Program Memory Ready
29	$0038	USART1_RX	USART1 Rx Complete
30	$003A	USART1_UDRE	USART1 Data Register Empty
31	$003C	USART1_TX	USART1 Tx Complete
32	$003E	TIMER3_CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3_COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3_COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3_OVF	Timer/Counter3 Overflow

Typical start of a program with interrupts

Here it comes under hardware control when Timer0 overflows

```
jmp INIT ; 2 word Vect  INIT
nop ; Ext Int 0
reti ;
nop ; Ext Int 1
reti ;
nop ; Ext Int 2
reti ;
nop ; PCINT 0 - Pin Change
reti ;
nop ; PCINT 1 - Pin Change
reti ;
nop ; PCINT 2 - Pin Change
reti ;
nop ; PCINT 3 - Pin Change
reti ;
nop ; WDT  - Watchdog
Timeout
reti ;
nop ; Timer 2 Compare A
reti ;
nop ; Timer 2 Compare B
reti ;
nop ; Timer 2 Overflow
reti ;
```

```
nop ; Timer 1 Capture
reti ;
nop ; Timer 1 Compare A
reti ;
nop ; Timer 1 Compare B
reti     ;
nop ; Timer 1 Overflow
reti ;
nop ; Timer 0 Compare A
reti ;
nop ; Timer 0 Compare B
reti ;
jmp T0_OF ; Timer 0 Overflow
nop ; SPI Serial Transfer Complete
reti ;
nop ; USART0_RX  Complete
reti ;
nop ; USART0 Data Register Empty
reti ;
nop ; USART0_TX complete
reti ;
nop ; Analog Comparator
reti ;
nop ; ADC Conversion complete
reti ;
```

```
nop ; EEPROM Ready
reti ;
nop ; Two Wire Interface
reti ;
nop ; Store Program Memory Ready
reti ;
nop ; USART1 Rx Complete
reti ;
nop ; USART1 Data Register Empty
reti ;
nop ; USART1 Tx Complete
reti ;
nop ; Timer/Counter 3 capture Event
reti ;
nop ; Timer/Counter 3 Compare A
reti ;
nop ; Timer/Counter 3 Compare B
reti ;
nop ; Timer/Couner  3 Overflow
reti
```

T0_OF is the interrupt service routine name. This is where the program will jump if a Timer0 Pverflow interrupt occurs.....

At Program address 0 your program executes the initialization

Timer/Counter0 Mask Register

TIMSK0

D7	D6	D5	D4	D3	D2	D1	D0
--------	--------	--------	--------	-------	OCIE0B	OCIE0A	TOIE0

OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable
OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable
TOIE0: Timer/Counter0 Overflow Interrupt Enable
YOU NEED TO SET THIS BIT TO '1' TO ENABLE THE TIMER0/COUNER0 OVERFLOW INTERRUOT !!

Timer/Counter0 Control Registers

TCCR0A:

D7	D6	D5	D4	D3	D2	D1	D0
COM0A1	COM0A0	COM0B1	COM0B0	----	----	*WGM01*	*WGM00*

Bits 7:6 – COM0A1:0: Compare Match Output A Mode

Bits 5:4 – COM0B1:0: Compare Match Output B Mode

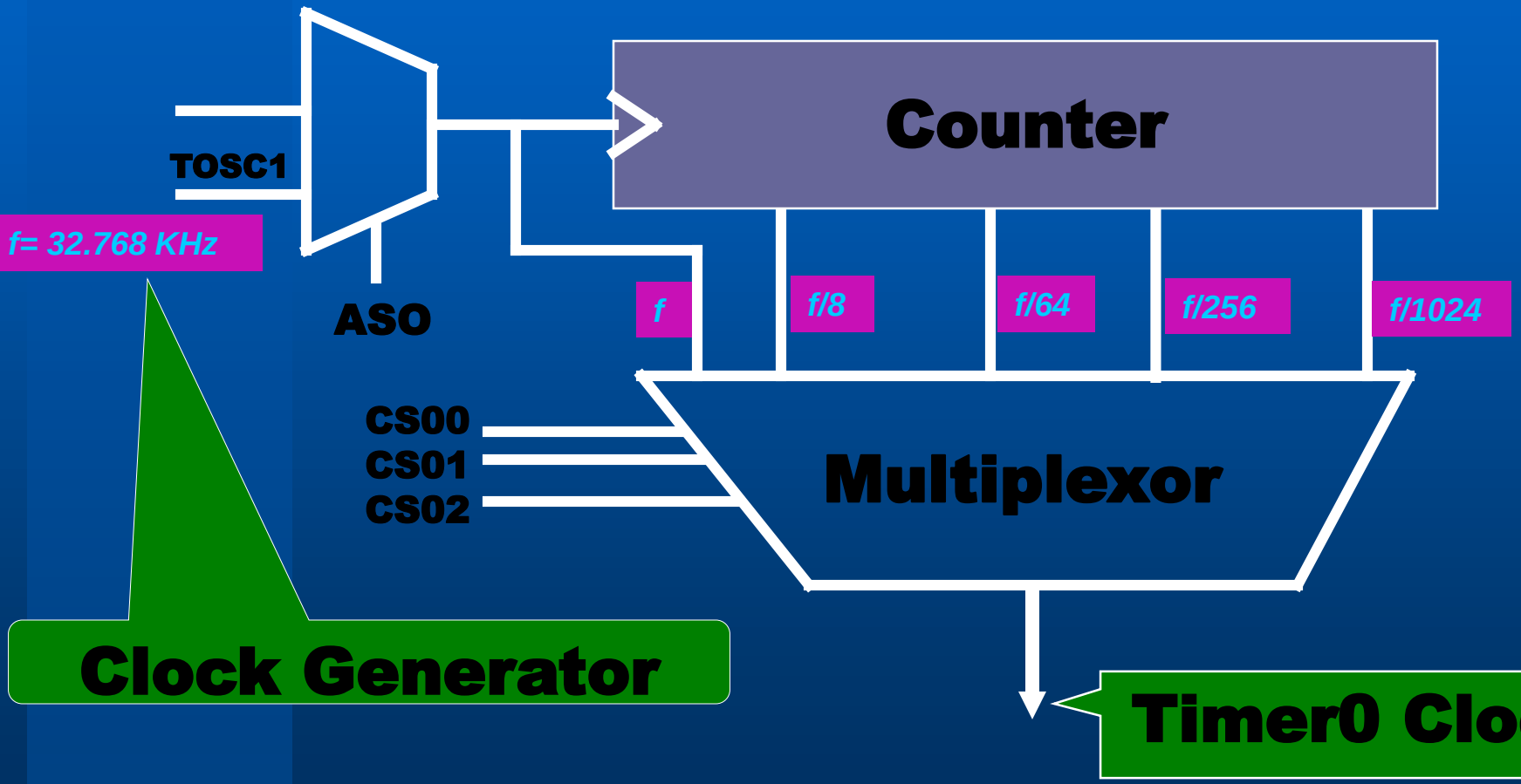
Bits 1:0 – WGM01:0: Waveform Generation Mode

TCCR0B:

D7	D6	D5	D4	D3	D2	D1	D0
FOC0A	FOC0B	---	---	WGM2	*CS02*	*CS01*	*CS00*

The timer pre-scale factor : CS02; CS01; CS00

Pre-scaling the Timer via TCCR0B (1)



Prescaling via TCCR0B (2)

CS02	CS01	CS00	Frequency f=32.768 KHz
0	0	0	Timer/Counter0 is stopped
0	0	1	f (no prescale)
0	1	0	f/8
0	1	1	f/64
1	0	0	f/256
1	0	1	f/1024
1	1	0	External Clock at T0 Clock on falling edge
1	1	1	Extranl Clock at T0 clock on rising edge

Timer/Counter0 TCNT0

D7	D6	D5	D4	D3	D2	D1	D0

This is the counter preset

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

Initialization routine (1)

```
INIT:
;
CLI          ; Disable interrupts during initialization
;
; THE NEXT 4 LINES LOAD THE STACK POINTER WITHT THE ADDRESS
; OF THE BOTTOM OF THE SRAM
;
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
;
; MAKE PORTB AN OUTPUT PORT
;
LDI R16, $FF
OUT DDRB, R16
;
;
LDI R16, $00
out TCCR0A, R16  ; NORMAL OPERATION NO PWM
LDI R16, $05
OUT TCCR0B, R16  ; NO PWM AND PRESCALE THE 32 KHZ CLOCK BY 1024
LDI R16, $01
STS TIMSK0, R16  ; ENABLE TIMER0/COUNTER0 OVERFLOW INTERRUPT
;
```

Initialization routine (2)

```
;
;                    Initializing counter registers....
;
LDI R18, $00     ; THIS HAS THE INCREMENTING NUMBERS TO BE SENT TO PORTB
LDI R17, $00     ; THIS COUNTS THE NUMBER OF INTERRUPTS
;
SEI              ; ENABLE INTERRUPTS
;
```

Interrupt Service Routine

```
T0_OF:              ; INTERRUPT SERVICE ROUTINE FOR TIMER/COUNTER 0 OVERFLOW
  ;
   CLI              ; DISABLE INTERRUPTS
   IN R19, SREG     ; STORE SREG CONTENTS ON R19
                    ; SO THAT YOU CAN RESTORE SREG AFTER
                    ; THE INTERRUPT SERVICE ROUTINE HAS BEEN EXECUTED
  ;
   INC R17          ; COUNT INTERRUPTS
   CPI R17, $15     ; EVERY 21 INTERRUPTS SEND AN INCREMENTING NUMBER FROM
                    ; R18 TO THE LEDS OF PORTB
   BRNE AGAIN
   CLR R17
  ;
   OUT PORTB, R18
   INC R18
AGAIN:
   OUT SREG, R19    ; RESTORE SREG
   SEI              ; ENABLE INTERRUPTS AGAIN
   RETI             ; GO BACK TO WHAT YOU WERE DOING AT THE TIME OF THE INTERRUPT
```

Main Program

```
Main:                ;   waisting time
;
            nop
            nop
            nop
            nop
;
            rjmp main
```

But the timer is counting in the background and when it overflows it causes an interrupt forcing the program to execute the interrupt service routine. After that it comes back in the loop where it was when the interrupt occurred……see STUDIO demo.