

Embedded Systems

Lecture 8: Timers

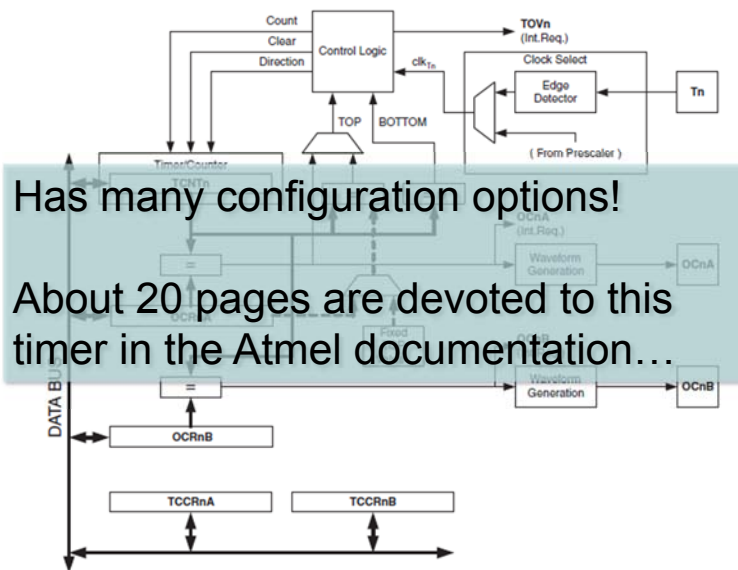


F-35 Lightning II Electro-optical Targeting System (EOTS)

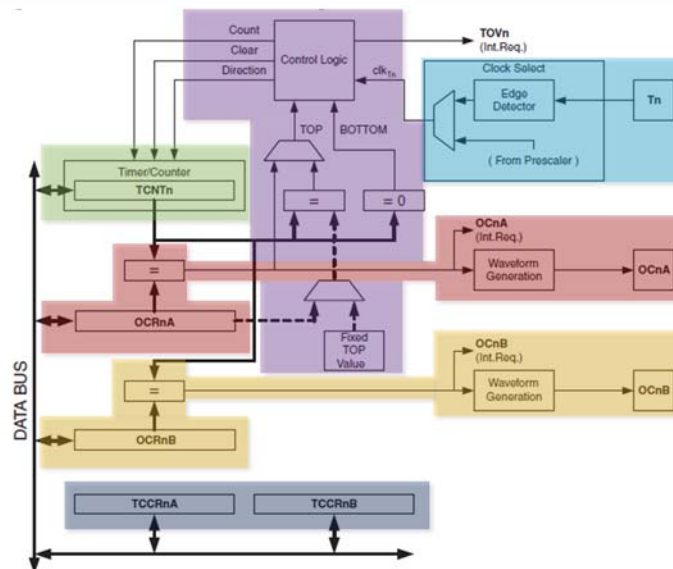
Timers and Counters

- **Very important peripherals on microcontrollers**
- **ATtiny45 has two 8-bit timers**
- **ATmega88PA has**
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
- **We will explore the ATtiny45 timers. Later in course, students will use timers on ATmega88PA**

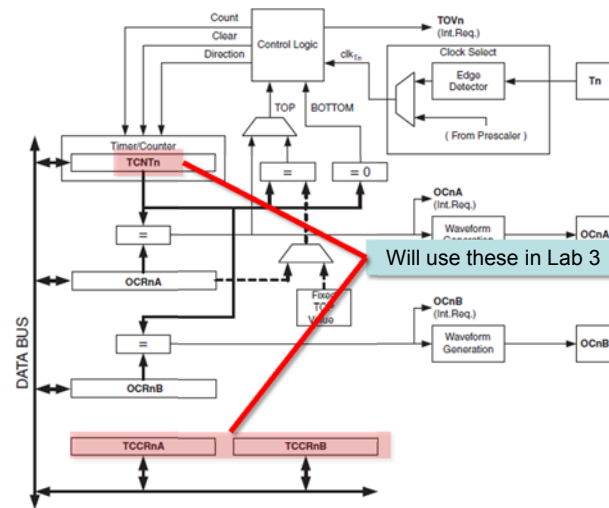
8-Bit Timer/Counter for ATtiny45



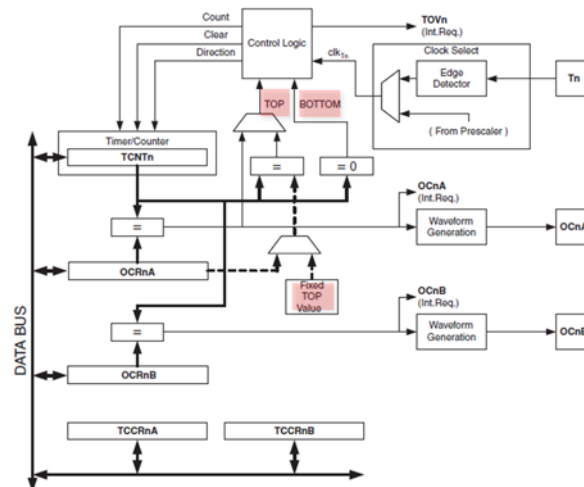
ATtiny45 8-Bit Counter/Timers – Basic Components



Many register and bit references in this section are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 0. A lower case "x" replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

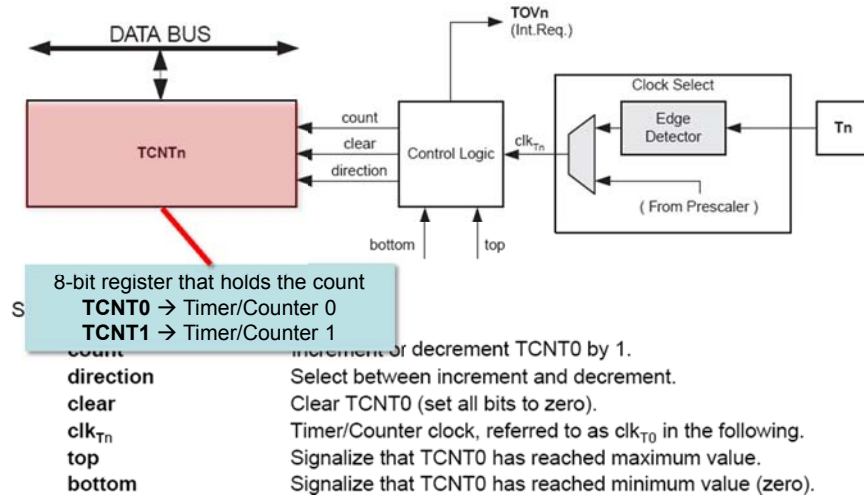


Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x00
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255)
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment depends on the mode of operation



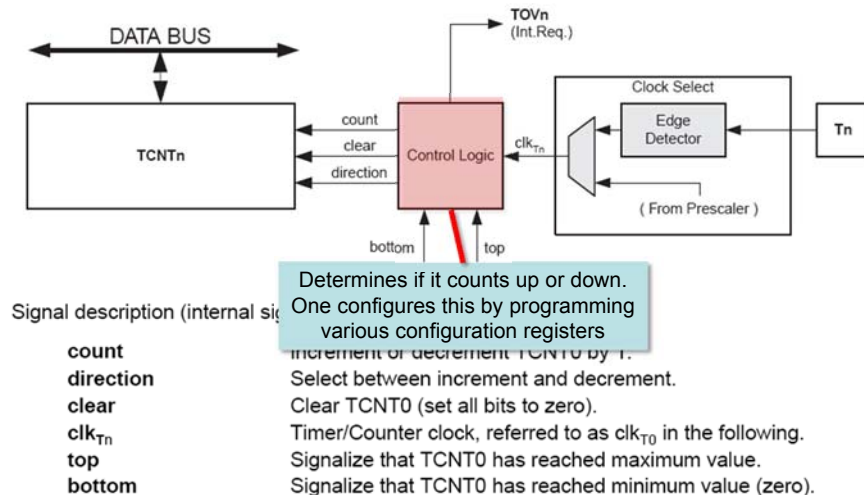
The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit.

Figure 11-4. Counter Unit Block Diagram



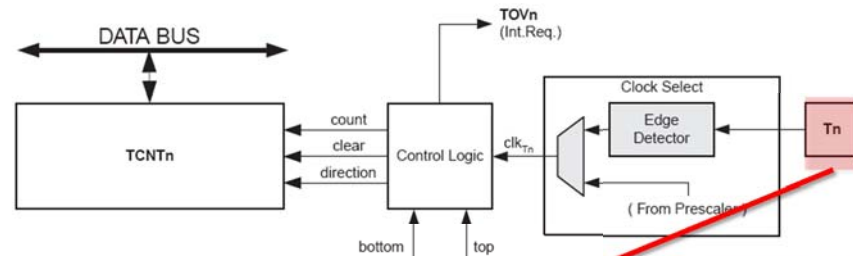
The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit.

Figure 11-4. Counter Unit Block Diagram



The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit.

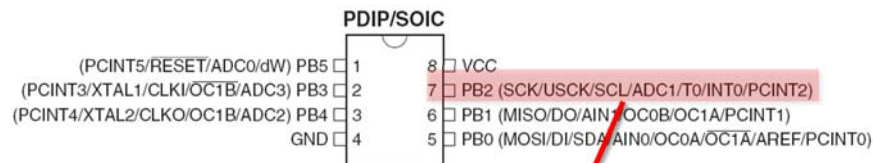
Figure 11-4. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk _{Tn}	Timer/Counter clock, referred to as clk _{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

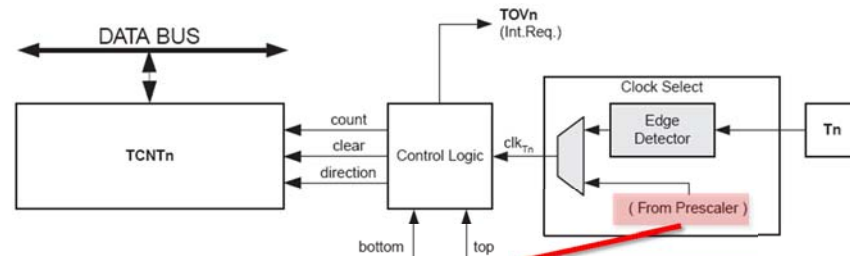
Input pulses can come from external pin **T0**. This is **PB2**, assuming it is properly configured.



Input pulses can come from external pin **T0**. This is **PB2**, assuming it is properly configured.

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit.

Figure 11-4. Counter Unit Block Diagram



Signal description (i

count
direction
clear
clk_{Tn}
top
bottom

Input pulses can come from main system clock, optionally divided or prescaled by system clock frequency:

$f_{CLK_I/O}$, $f_{CLK_I/O}/8$, $f_{CLK_I/O}/64$, $f_{CLK_I/O}/256$, or $f_{CLK_I/O}/1024$.

Assuming a 10 MHz clock this would be

10 MHz, 1.25 MHz, 156.25 KHz, 39.0625 kHz, and 9.765625 kHz

Signalize that TCNT0 has reached minimum value (zero).

TCNT0 – Timer/Counter Register

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x32	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter.

```
; Wait for TIMER0 to roll over.
delay:
    in    r24,TCNT0    ; Get counter value
    cpi   r24,0x00     ; is it zero?
    brne delay         ; no --> wait some more
    ...                ; yes --> continue on
```



“Naive Approach!”

Configuring Timers

GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C	TSM	PWM1B	COM1B1	COM1B0	FOC1B	FOC1A	PSR1	PSR0	GTCCR
Read/Write	R/W	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

This configures some of the more advanced features of the timers, see documentation. We may return to this later in course. Sensible default/initial values are provided.

Configuring Timers

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

These bits configure some of the more advanced features of the timers, see documentation. We may return to this later in course.

These are "Clock Select" bits

0x02 will clock the timer at 1/8 the system clock

Table 11-6. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{\text{IO}}/1$ (No prescaling)
0	1	0	$\text{clk}_{\text{IO}}/8$ (From prescaler)
0	1	1	$\text{clk}_{\text{IO}}/64$ (From prescaler)
1	0	0	$\text{clk}_{\text{IO}}/256$ (From prescaler)
1	0	1	$\text{clk}_{\text{IO}}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Configuring Timers

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 11-5. Waveform Generation Mode Bit Description

Mode	WGM 02	WGM 01	WGM 00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX ⁽¹⁾
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM ⁽²⁾
2	0	1	0	CTC	OCRA	Immediate	MAX ⁽¹⁾
3	0	1	1	Fast PWM	0xFF	BOTTOM ⁽²⁾	MAX ⁽¹⁾
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM ⁽²⁾
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM ⁽²⁾	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Configuring Timers – Normal Mode

Table 11-5. Waveform Generation Mode Bit Description

Mode	WGM 02	WGM 01	WGM 00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX ⁽¹⁾
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM ⁽²⁾
2	0	1	0	CTC	OCRA	Immediate	MAX ⁽¹⁾
3	0	1	1	Fast PWM	0xFF	BOTTOM ⁽²⁾	MAX ⁽¹⁾
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM ⁽²⁾
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM ⁽²⁾	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Normal Mode

The simplest mode of operation is the Normal mode (WGM0[2:0] = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Configuring T/Cs – “TOVn” in Normal Mode

11.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{TC}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 11-10 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 11-10. Timer/Counter Timing Diagram, no Prescaling

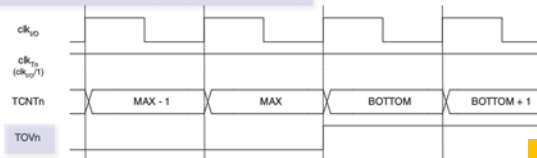
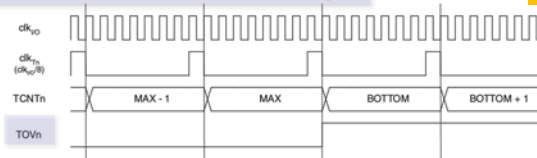


Figure 11-11 shows the same timing data, but with the prescaler enabled.

Figure 11-11. Timer/Counter Timing Diagram, with Prescaler ($f_{clk_IO}/8$)



Normal Mode:
MAX=0xFF
BOTTOM=0x00

Timer/Counter (T/C) Configuration

- **Key questions:**
 - **Clock Source?**
 - internal → μ C clock selection, prescaler, ...
 - external → rising/falling edge
 - **T/C Mode?**
 - Normal
 - Pulse With Modulation (PWM): fast, phase correct, ...
 - Clear Timer on Compare match (CTC)
 - ...
 - **T/C Integration into program?**
 - Polling
 - Interrupt
 - Waveform generation with T/C HW
 - **Initial values of TCNTn, OCRnA, OCRnB, ... ?**
 - **Initialization sequence**
- Note: code snippets provided only illustrate concepts!
- Check configuration → debugging in Atmel Studio 7 → I/O View

Timer/Counter Configuration – I/O View

The screenshot shows the AVR Studio interface with the I/O View for Timer/Counter 0. The left pane lists configuration options for Timer/Counter 0, including interrupt enables, flags, and output modes. The right pane shows the hardware register structure for the timer. A red box highlights the configuration options in the left pane.

Configuration options (left pane):

- Timer/Counter0 Output Compare Match A Interrupt Enable
- Timer/Counter0 Output Compare Match B Interrupt Enable
- Timer/Counter0 Overflow Interrupt Enable
- Timer/Counter0 Output Compare Flag 0A
- Timer/Counter0 Output Compare Flag 0B
- Timer/Counter0 Overflow Flag
- Compare Output Mode, Phase Correct PWM Mode
- Compare Output Mode, Fast PWM
- Waveform Generation Mode
- Force Output Compare A
- Force Output Compare B
- WGM02
- Clock Select
- Timer/Counter0
- Timer/Counter0 Output Compare Register
- Timer/Counter0 Output Compare Register

Hardware register structure (right pane):

Register	Address	Value	Bit
PORTA	0x00	0x00	0-7
PORTB	0x01	0x00	0-7
PORTC	0x02	0x00	0-7
PORTD	0x03	0x00	0-7
TCCR0A	0x04	0x00	0-7
TCCR0B	0x05	0x00	0-7
TCNT0	0x06	0x00	0-7
OCR0A	0x07	0x00	0-7
OCR0B	0x08	0x00	0-7
TCR0	0x09	0x00	0-7

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 19

Configuring Timers

The timers have configuration/control registers (**GTCCR**, **TCCR0A**, and **TCCR0B**). Since they are located in I/O memory space, one uses **IN/OUT** instructions to access the registers.

```
// *** Example in C code ***
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 1250.000 kHz
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A = 0x00;
TCCR0B = 0x02;
TCNT0 = 0x00;
OCR0A = 0x00;
OCR0B = 0x00;
```

```
LDI R30,0x02
OUT 0x33,R30
```

Assembly code to configure ATtiny45 **TIMER0** to use the system clock, prescaled by 8

C code to configure ATtiny45 **TIMER0** to use the system clock, prescaled by 8

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 20

Generating Square Waves – “Naive Approach!”

```
.include "tn45def.inc"
.def tmp = r23 ; Use r23 for temporary variables
.cseg

; Configure TIMER0 for input from system clock
; prescaled by 8. At 10 MHz, this will increment
; the clock at 1.250 MHz
ldi tmp,0x02
out TCCR0B,tmp

sbi DDRB,0 ; Configure PB0 as output

loop:
cbi PORTB,0 ; Pull PB0 low
rcall delay ; Wait
sbi PORTB,0 ; Pull PB0 high
rcall delay ; Wait
rjmp loop ; Start again

; Wait for TIMER0 to roll over.
delay:
in tmp,TCNT0 ; Get counter value
cpi tmp,0x00 ; is it zero?
brne delay ; no --> wait some more
ret ; yes --> return
.exit
```

Note the **.def** directive

Note the **.cseg** directive

Load 0x02 into **TCCR0B**.

Pull low and wait

Pull high and wait

Timer rolls over and become zero and continue counting up

Excluding overhead this program generates a ~2.4 kHz signal. (Why?)



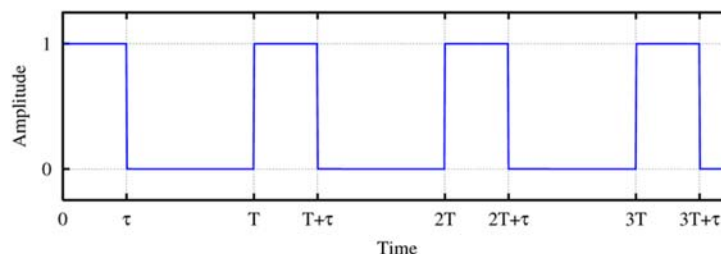
Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 21

Square Waves & Duty Cycle

The **duty cycle** is the fraction of time that a system is in an "active" state or duty cycle is the proportion of time during which a component, device, or system is operated. Suppose a disk drive operates for 1 second, and is shut off for 99 seconds, then is run for 1 second again, and so on. The drive runs for one out of 100 seconds, or 1/100 of the time, and its duty cycle is therefore 1/100, or 1 percent.

For a square wave:



$$D = \frac{\tau}{T}$$

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 22

Generating Square Waves - "Naive Approach!"

```
.include "tn45def.inc"
.def tmp = r23 ; Use r23 for temporary variables
.def count = r24 ; Use r24 for reloading timer
.cseg

; Configure TIMER0 for input from system clock
; prescaled by 8. At 10 MHz, this will increment
; the clock at 1.250 MHz
ldi tmp,0x02
out TCCR0B,tmp

sbi DDRB,0 ; Configure PB0 as output

loop:
cbi PORTB,0 ; Pull PB0 low
ldi count,255-64
rcall delay
sbi PORTB,0 ; Pull PB0 high
ldi count,64
rcall delay ; Wait
rjmp loop ; Start again

; Wait for TIMER0 to roll over.
delay:
out TCNT0,count ; Load counter
wait:
in tmp,TCNT0 ; Get counter value
cpi tmp,0x00 ; is it zero (rolled over)?
brne wait ; no --> wait some more
ret ; yes, return
.exit
```

Note the .def directives

Load 0x02 into TCCR0B.

Note how we use the Assembler to do calculations for us

Excluding overhead this program generates approx. a 4.8 KHz wave.

The duty cycle is ~ 75%.



Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 23

More Reliable Delay

The previous delay routine has some "flaws" since the timer is still counting while it is being reloaded. Further, checking for overflow by checking against zero while the counter is running may cause problems. A more reliable method is to check the timer overflow flag (TOV0) in the TIFR register:

TIFR – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38	–	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	–	TIFR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

Set when timer 0 rolls over. To clear, write logic 1 (yes "1" and not "0" ...)

The TIFR register is not in the lower 32 I/O registers, so we CANNOT use CBI, SBI etc. Rather, we must use IN, SBR, OUT:

```
; Clear over flow flag.
in tmp2,TIFR ; tmp <-- TIFR
sbr tmp2,1<TOV0 ; Clear TOV0, write logic 1
out TIFR,tmp2
```

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 24

More Reliable Delay

```

.def tmp1 = r23      ; Use r23 for temporary variables
.def tmp2 = r24      ; Use r24 for temporary values

...

; Wait for TIMER0 to roll over.
delay:

; Stop timer 0.
in  tmp1,TCCR0B      ; Save configuration
ldi tmp2,0x00        ; Stop timer 0
out TCCR0B,tmp2

; Clear over flow flag.
in  tmp2,TIFR         ; tmp <-- TIFR
sbr tmp2,1<<TOV0      ; Clear TOV0, write logic 1
out TIFR,tmp2

; Start timer with new initial count
out TCNT0,count       ; Load counter
out TCCR0B,tmp1       ; Restart timer

wait:
in  tmp2,TIFR         ; tmp <-- TIFR
sbrs tmp2,TOV0        ; Check overflow flag
rjmp wait
ret

```

Stop timer, clear overflow flag, reload, and start timer

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 25

Quiz

- An engineer wants to utilize TC0 of an ATtiny45 with 8 MHz external (μ C) clock to produce a delay of 7.5 ms.
- Q1: What prescaler should be used for highest accuracy?
- Q2: In “normal mode” (using TOV0), what value needs to be loaded into TCNT0?
- Q3: What will be the timing error in percent?
- Q4: What values need to be loaded into TCCR0A and TCCR0B?

Solution → on blackboard!

Embedded Systems, ECE:3360. The University of Iowa, 2019

Timers, Slide 26

Loading 16-Bit Timer on ATmega88PA

Many AVRs have 16-bit timer, meaning that the registers that hold the counts, are 16-bits (two bytes) wide. However, the AVR core is 8-bit, so loading the 16-bit registers must proceed byte-wise.

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

There is an internal **TEMP** register, **shared by all other 16-bit registers within each timer**, that facilitates loading.

The I/O instructions to the 16-bit timer registers involve this **TEMP** register.

However, on the ATmega88PA, **TCNT1H** and **TCNT1L** are outside the range of the IN/OUT instructions → use the **LDS/STS** instructions. (see ATmega88PA documentation)

To load the 16-bit timer, first **write** to **TCNT1H**. This does not load **TCNT1H**, but loads the internal **TEMP**. Next, **write** to **TCNT1L**. This loads **TCNT1L** and triggers the transfer from **TEMP** to **TCNT1H**.

Loading 16-Bit Timer on ATmega88PA

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Step 1: Write the high byte. This places the content of r17 in the internal TEMP, and not TCNT1H.

STS TCNT1H, r17

→ TEMP

TCNT1H

TCNT1L

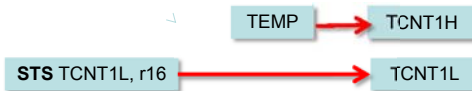
Loading 16-Bit Timer on ATmega88PA

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Step 2: Write the low byte.

This copies the contents of r16 to TCNT1L, and also triggers the transfer from the internal TEMP to TCNT1H, all in one clock cycle



Loading 16-Bit Timer on ATmega88PA

; Set TCNT1 to 0x01FF

ldi r17, 0x01

ldi r16, 0xFF

sts TCNT1H, r17

sts TCNT1L, r16

First load TCNT1H.
However, contents go to internal TEMP.

Next, load TCNT1L.
This loads TCNT1L and triggers loading of TCNT1H.

Reading 16-Bit Timer on ATmega88PA

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Step 1: Read the low byte. Use the **LDS** instruction, since the registers are outside of the range of **IN**.

This copies the contents of TCNT1L to r16, and also triggers the transfer from the TCNT1H to the internal TEMP



Reading 16-Bit Timer on ATmega88PA

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Step 2: Read the high byte.

This copies the contents of the internal TEMP to r17.



Reading 16-Bit Timer on ATmega88PA

To read the 16-bit timer, first read (using **LDS**) from TCNT1L. This also copies the contents of TCNT1H to the internal TEMP. Next read from TCNT1H.

; Read TCNT1 into r17:r16

LDS r16, TCNT1L

LDS r17, TCNT1H

...

First read TCNT1L. This also copies the contents of TCNT1H to the internal TEMP register.

Next, read TCNT1H.

From "m88PAdef.inc"

```

36 ; ***** I/O REGISTER DEFINITIONS *****
37 ; NOTE:
38 ; Definitions marked "MEMORY MAPPED" are extended I/O ports
39 ; and cannot be used with IN/OUT instructions
40 .equ 0x00, SDRB = 0x00 ; MEMORY MAPPED
41 .equ 0x01, USBRBL = 0x01 ; MEMORY MAPPED
42 .equ 0x02, USBRBH = 0x02 ; MEMORY MAPPED
43 .equ 0x03, UCSBAC = 0x03 ; MEMORY MAPPED
44 .equ 0x04, UCSABB = 0x04 ; MEMORY MAPPED
45 .equ 0x05, UCSABA = 0x05 ; MEMORY MAPPED
46 .equ 0x06, TWAMB = 0x06 ; MEMORY MAPPED
47 .equ 0x07, TWCR = 0x07 ; MEMORY MAPPED
48 .equ 0x08, TWDR = 0x08 ; MEMORY MAPPED
49 .equ 0x09, TWAR = 0x09 ; MEMORY MAPPED
50 .equ 0x0A, TWGR = 0x0A ; MEMORY MAPPED
51 .equ 0x0B, TWBR = 0x0B ; MEMORY MAPPED
52 .equ 0x0C, ASSR = 0x0C ; MEMORY MAPPED
53 .equ 0x0D, OCR2B = 0x0D ; MEMORY MAPPED
54 .equ 0x0E, OCR2A = 0x0E ; MEMORY MAPPED
55 .equ 0x0F, TCNT2 = 0x0F ; MEMORY MAPPED
56 .equ 0x10, TCCR2B = 0x10 ; MEMORY MAPPED
57 .equ 0x11, TCCR2A = 0x11 ; MEMORY MAPPED
58 .equ 0x12, OCR1L = 0x12 ; MEMORY MAPPED
59 .equ 0x13, OCR1H = 0x13 ; MEMORY MAPPED
60 .equ 0x14, OCR1A = 0x14 ; MEMORY MAPPED
61 .equ 0x15, OCR1B = 0x15 ; MEMORY MAPPED
62 .equ 0x16, TCNT1L = 0x16 ; MEMORY MAPPED
63 .equ 0x17, TCNT1H = 0x17 ; MEMORY MAPPED
64 .equ 0x18, TCCR1C = 0x18 ; MEMORY MAPPED
65 .equ 0x19, TCCR1B = 0x19 ; MEMORY MAPPED
66 .equ 0x1A, TCCR1A = 0x1A ; MEMORY MAPPED
67 .equ 0x1B, TCCR1C = 0x1B ; MEMORY MAPPED
68 .equ 0x1C, TCCR1A = 0x1C ; MEMORY MAPPED
69 .equ 0x1D, TCCR1B = 0x1D ; MEMORY MAPPED
70 .equ 0x1E, TCCR1C = 0x1E ; MEMORY MAPPED
71 .equ 0x1F, TCCR1A = 0x1F ; MEMORY MAPPED
72 .equ 0x20, TCCR1B = 0x20 ; MEMORY MAPPED
73 .equ 0x21, TCCR1C = 0x21 ; MEMORY MAPPED
74 .equ 0x22, TCCR1A = 0x22 ; MEMORY MAPPED
75 .equ 0x23, TCCR1B = 0x23 ; MEMORY MAPPED
76 .equ 0x24, TCCR1C = 0x24 ; MEMORY MAPPED
77 .equ 0x25, TCCR1A = 0x25 ; MEMORY MAPPED
78 .equ 0x26, TCCR1B = 0x26 ; MEMORY MAPPED
79 .equ 0x27, TCCR1C = 0x27 ; MEMORY MAPPED
80 .equ 0x28, TCCR1A = 0x28 ; MEMORY MAPPED
81 .equ 0x29, TCCR1B = 0x29 ; MEMORY MAPPED
82 .equ 0x2A, TCCR1C = 0x2A ; MEMORY MAPPED
83 .equ 0x2B, TCCR1A = 0x2B ; MEMORY MAPPED
84 .equ 0x2C, TCCR1B = 0x2C ; MEMORY MAPPED
85 .equ 0x2D, TCCR1C = 0x2D ; MEMORY MAPPED
86 .equ 0x2E, TCCR1A = 0x2E ; MEMORY MAPPED
87 .equ 0x2F, TCCR1B = 0x2F ; MEMORY MAPPED
88 .equ 0x30, TCCR1C = 0x30 ; MEMORY MAPPED
89 .equ 0x31, TCCR1A = 0x31 ; MEMORY MAPPED
90 .equ 0x32, TCCR1B = 0x32 ; MEMORY MAPPED
91 .equ 0x33, TCCR1C = 0x33 ; MEMORY MAPPED
92 .equ 0x34, TCCR1A = 0x34 ; MEMORY MAPPED
93 .equ 0x35, TCCR1B = 0x35 ; MEMORY MAPPED
94 .equ 0x36, TCCR1C = 0x36 ; MEMORY MAPPED
95 .equ 0x37, TCCR1A = 0x37 ; MEMORY MAPPED
96 .equ 0x38, TCCR1B = 0x38 ; MEMORY MAPPED
97 .equ 0x39, TCCR1C = 0x39 ; MEMORY MAPPED
98 .equ 0x3A, TCCR1A = 0x3A ; MEMORY MAPPED
99 .equ 0x3B, TCCR1B = 0x3B ; MEMORY MAPPED
100 .equ 0x3C, TCCR1C = 0x3C ; MEMORY MAPPED
101 .equ 0x3D, TCCR1A = 0x3D ; MEMORY MAPPED
102 .equ 0x3E, TCCR1B = 0x3E ; MEMORY MAPPED
103 .equ 0x3F, TCCR1C = 0x3F ; MEMORY MAPPED
104 .equ 0x40, TCCR1A = 0x40 ; MEMORY MAPPED
105 .equ 0x41, TCCR1B = 0x41 ; MEMORY MAPPED
106 .equ 0x42, TCCR1C = 0x42 ; MEMORY MAPPED
107 .equ 0x43, TCCR1A = 0x43 ; MEMORY MAPPED
108 .equ 0x44, TCCR1B = 0x44 ; MEMORY MAPPED
109 .equ 0x45, TCCR1C = 0x45 ; MEMORY MAPPED
110 .equ 0x46, TCCR1A = 0x46 ; MEMORY MAPPED
111 .equ 0x47, TCCR1B = 0x47 ; MEMORY MAPPED
112 .equ 0x48, TCCR1C = 0x48 ; MEMORY MAPPED
113 .equ 0x49, TCCR1A = 0x49 ; MEMORY MAPPED
114 .equ 0x4A, TCCR1B = 0x4A ; MEMORY MAPPED
115 .equ 0x4B, TCCR1C = 0x4B ; MEMORY MAPPED
116 .equ 0x4C, TCCR1A = 0x4C ; MEMORY MAPPED
117 .equ 0x4D, TCCR1B = 0x4D ; MEMORY MAPPED
118 .equ 0x4E, TCCR1C = 0x4E ; MEMORY MAPPED
119 .equ 0x4F, TCCR1A = 0x4F ; MEMORY MAPPED
120 .equ 0x50, TCCR1B = 0x50 ; MEMORY MAPPED
121 .equ 0x51, TCCR1C = 0x51 ; MEMORY MAPPED
122 .equ 0x52, TCCR1A = 0x52 ; MEMORY MAPPED
123 .equ 0x53, TCCR1B = 0x53 ; MEMORY MAPPED
124 .equ 0x54, TCCR1C = 0x54 ; MEMORY MAPPED
125 .equ 0x55, TCCR1A = 0x55 ; MEMORY MAPPED
126 .equ 0x56, TCCR1B = 0x56 ; MEMORY MAPPED
127 .equ 0x57, TCCR1C = 0x57 ; MEMORY MAPPED
128 .equ 0x58, TCCR1A = 0x58 ; MEMORY MAPPED
129 .equ 0x59, TCCR1B = 0x59 ; MEMORY MAPPED
130 .equ 0x5A, TCCR1C = 0x5A ; MEMORY MAPPED
131 .equ 0x5B, TCCR1A = 0x5B ; MEMORY MAPPED
132 .equ 0x5C, TCCR1B = 0x5C ; MEMORY MAPPED
133 .equ 0x5D, TCCR1C = 0x5D ; MEMORY MAPPED
134 .equ 0x5E, TCCR1A = 0x5E ; MEMORY MAPPED
135 .equ 0x5F, TCCR1B = 0x5F ; MEMORY MAPPED
136 .equ 0x60, TCCR1C = 0x60 ; MEMORY MAPPED
137 .equ 0x61, TCCR1A = 0x61 ; MEMORY MAPPED
138 .equ 0x62, TCCR1B = 0x62 ; MEMORY MAPPED
139 .equ 0x63, TCCR1C = 0x63 ; MEMORY MAPPED
140 .equ 0x64, TCCR1A = 0x64 ; MEMORY MAPPED
141 .equ 0x65, TCCR1B = 0x65 ; MEMORY MAPPED
142 .equ 0x66, TCCR1C = 0x66 ; MEMORY MAPPED
143 .equ 0x67, TCCR1A = 0x67 ; MEMORY MAPPED
144 .equ 0x68, TCCR1B = 0x68 ; MEMORY MAPPED
145 .equ 0x69, TCCR1C = 0x69 ; MEMORY MAPPED
146 .equ 0x6A, TCCR1A = 0x6A ; MEMORY MAPPED
147 .equ 0x6B, TCCR1B = 0x6B ; MEMORY MAPPED
148 .equ 0x6C, TCCR1C = 0x6C ; MEMORY MAPPED
149 .equ 0x6D, TCCR1A = 0x6D ; MEMORY MAPPED
150 .equ 0x6E, TCCR1B = 0x6E ; MEMORY MAPPED
151 .equ 0x6F, TCCR1C = 0x6F ; MEMORY MAPPED
152 .equ 0x70, TCCR1A = 0x70 ; MEMORY MAPPED
153 .equ 0x71, TCCR1B = 0x71 ; MEMORY MAPPED
154 .equ 0x72, TCCR1C = 0x72 ; MEMORY MAPPED
155 .equ 0x73, TCCR1A = 0x73 ; MEMORY MAPPED
156 .equ 0x74, TCCR1B = 0x74 ; MEMORY MAPPED
157 .equ 0x75, TCCR1C = 0x75 ; MEMORY MAPPED
158 .equ 0x76, TCCR1A = 0x76 ; MEMORY MAPPED
159 .equ 0x77, TCCR1B = 0x77 ; MEMORY MAPPED
160 .equ 0x78, TCCR1C = 0x78 ; MEMORY MAPPED
161 .equ 0x79, TCCR1A = 0x79 ; MEMORY MAPPED
162 .equ 0x7A, TCCR1B = 0x7A ; MEMORY MAPPED
163 .equ 0x7B, TCCR1C = 0x7B ; MEMORY MAPPED
164 .equ 0x7C, TCCR1A = 0x7C ; MEMORY MAPPED
165 .equ 0x7D, TCCR1B = 0x7D ; MEMORY MAPPED
166 .equ 0x7E, TCCR1C = 0x7E ; MEMORY MAPPED
167 .equ 0x7F, TCCR1A = 0x7F ; MEMORY MAPPED
168 .equ 0x80, TCCR1B = 0x80 ; MEMORY MAPPED
169 .equ 0x81, TCCR1C = 0x81 ; MEMORY MAPPED
170 .equ 0x82, TCCR1A = 0x82 ; MEMORY MAPPED
171 .equ 0x83, TCCR1B = 0x83 ; MEMORY MAPPED
172 .equ 0x84, TCCR1C = 0x84 ; MEMORY MAPPED
173 .equ 0x85, TCCR1A = 0x85 ; MEMORY MAPPED
174 .equ 0x86, TCCR1B = 0x86 ; MEMORY MAPPED
175 .equ 0x87, TCCR1C = 0x87 ; MEMORY MAPPED
176 .equ 0x88, TCCR1A = 0x88 ; MEMORY MAPPED
177 .equ 0x89, TCCR1B = 0x89 ; MEMORY MAPPED
178 .equ 0x8A, TCCR1C = 0x8A ; MEMORY MAPPED
179 .equ 0x8B, TCCR1A = 0x8B ; MEMORY MAPPED
180 .equ 0x8C, TCCR1B = 0x8C ; MEMORY MAPPED
181 .equ 0x8D, TCCR1C = 0x8D ; MEMORY MAPPED
182 .equ 0x8E, TCCR1A = 0x8E ; MEMORY MAPPED
183 .equ 0x8F, TCCR1B = 0x8F ; MEMORY MAPPED
184 .equ 0x90, TCCR1C = 0x90 ; MEMORY MAPPED
185 .equ 0x91, TCCR1A = 0x91 ; MEMORY MAPPED
186 .equ 0x92, TCCR1B = 0x92 ; MEMORY MAPPED
187 .equ 0x93, TCCR1C = 0x93 ; MEMORY MAPPED
188 .equ 0x94, TCCR1A = 0x94 ; MEMORY MAPPED
189 .equ 0x95, TCCR1B = 0x95 ; MEMORY MAPPED
190 .equ 0x96, TCCR1C = 0x96 ; MEMORY MAPPED
191 .equ 0x97, TCCR1A = 0x97 ; MEMORY MAPPED
192 .equ 0x98, TCCR1B = 0x98 ; MEMORY MAPPED
193 .equ 0x99, TCCR1C = 0x99 ; MEMORY MAPPED
194 .equ 0x9A, TCCR1A = 0x9A ; MEMORY MAPPED
195 .equ 0x9B, TCCR1B = 0x9B ; MEMORY MAPPED
196 .equ 0x9C, TCCR1C = 0x9C ; MEMORY MAPPED
197 .equ 0x9D, TCCR1A = 0x9D ; MEMORY MAPPED
198 .equ 0x9E, TCCR1B = 0x9E ; MEMORY MAPPED
199 .equ 0x9F, TCCR1C = 0x9F ; MEMORY MAPPED
200 .equ 0xA0, TCCR1A = 0xA0 ; MEMORY MAPPED
201 .equ 0xA1, TCCR1B = 0xA1 ; MEMORY MAPPED
202 .equ 0xA2, TCCR1C = 0xA2 ; MEMORY MAPPED
203 .equ 0xA3, TCCR1A = 0xA3 ; MEMORY MAPPED
204 .equ 0xA4, TCCR1B = 0xA4 ; MEMORY MAPPED
205 .equ 0xA5, TCCR1C = 0xA5 ; MEMORY MAPPED
206 .equ 0xA6, TCCR1A = 0xA6 ; MEMORY MAPPED
207 .equ 0xA7, TCCR1B = 0xA7 ; MEMORY MAPPED
208 .equ 0xA8, TCCR1C = 0xA8 ; MEMORY MAPPED
209 .equ 0xA9, TCCR1A = 0xA9 ; MEMORY MAPPED
210 .equ 0xAA, TCCR1B = 0xAA ; MEMORY MAPPED
211 .equ 0xAB, TCCR1C = 0xAB ; MEMORY MAPPED
212 .equ 0xAC, TCCR1A = 0xAC ; MEMORY MAPPED
213 .equ 0xAD, TCCR1B = 0xAD ; MEMORY MAPPED
214 .equ 0xAE, TCCR1C = 0xAE ; MEMORY MAPPED
215 .equ 0xAF, TCCR1A = 0xAF ; MEMORY MAPPED
216 .equ 0xB0, TCCR1B = 0xB0 ; MEMORY MAPPED
217 .equ 0xB1, TCCR1C = 0xB1 ; MEMORY MAPPED
218 .equ 0xB2, TCCR1A = 0xB2 ; MEMORY MAPPED
219 .equ 0xB3, TCCR1B = 0xB3 ; MEMORY MAPPED
220 .equ 0xB4, TCCR1C = 0xB4 ; MEMORY MAPPED
221 .equ 0xB5, TCCR1A = 0xB5 ; MEMORY MAPPED
222 .equ 0xB6, TCCR1B = 0xB6 ; MEMORY MAPPED
223 .equ 0xB7, TCCR1C = 0xB7 ; MEMORY MAPPED
224 .equ 0xB8, TCCR1A = 0xB8 ; MEMORY MAPPED
225 .equ 0xB9, TCCR1B = 0xB9 ; MEMORY MAPPED
226 .equ 0xBA, TCCR1C = 0xBA ; MEMORY MAPPED
227 .equ 0xBB, TCCR1A = 0xBB ; MEMORY MAPPED
228 .equ 0xBC, TCCR1B = 0xBC ; MEMORY MAPPED
229 .equ 0xBD, TCCR1C = 0xBD ; MEMORY MAPPED
230 .equ 0xBE, TCCR1A = 0xBE ; MEMORY MAPPED
231 .equ 0xBF, TCCR1B = 0xBF ; MEMORY MAPPED
232 .equ 0xC0, TCCR1C = 0xC0 ; MEMORY MAPPED
233 .equ 0xC1, TCCR1A = 0xC1 ; MEMORY MAPPED
234 .equ 0xC2, TCCR1B = 0xC2 ; MEMORY MAPPED
235 .equ 0xC3, TCCR1C = 0xC3 ; MEMORY MAPPED
236 .equ 0xC4, TCCR1A = 0xC4 ; MEMORY MAPPED
237 .equ 0xC5, TCCR1B = 0xC5 ; MEMORY MAPPED
238 .equ 0xC6, TCCR1C = 0xC6 ; MEMORY MAPPED
239 .equ 0xC7, TCCR1A = 0xC7 ; MEMORY MAPPED
240 .equ 0xC8, TCCR1B = 0xC8 ; MEMORY MAPPED
241 .equ 0xC9, TCCR1C = 0xC9 ; MEMORY MAPPED
242 .equ 0xCA, TCCR1A = 0xCA ; MEMORY MAPPED
243 .equ 0xCB, TCCR1B = 0xCB ; MEMORY MAPPED
244 .equ 0xCC, TCCR1C = 0xCC ; MEMORY MAPPED
245 .equ 0xCD, TCCR1A = 0xCD ; MEMORY MAPPED
246 .equ 0xCE, TCCR1B = 0xCE ; MEMORY MAPPED
247 .equ 0xCF, TCCR1C = 0xCF ; MEMORY MAPPED
248 .equ 0xD0, TCCR1A = 0xD0 ; MEMORY MAPPED
249 .equ 0xD1, TCCR1B = 0xD1 ; MEMORY MAPPED
250 .equ 0xD2, TCCR1C = 0xD2 ; MEMORY MAPPED
251 .equ 0xD3, TCCR1A = 0xD3 ; MEMORY MAPPED
252 .equ 0xD4, TCCR1B = 0xD4 ; MEMORY MAPPED
253 .equ 0xD5, TCCR1C = 0xD5 ; MEMORY MAPPED
254 .equ 0xD6, TCCR1A = 0xD6 ; MEMORY MAPPED
255 .equ 0xD7, TCCR1B = 0xD7 ; MEMORY MAPPED
256 .equ 0xD8, TCCR1C = 0xD8 ; MEMORY MAPPED
257 .equ 0xD9, TCCR1A = 0xD9 ; MEMORY MAPPED
258 .equ 0xDA, TCCR1B = 0xDA ; MEMORY MAPPED
259 .equ 0xDB, TCCR1C = 0xDB ; MEMORY MAPPED
260 .equ 0xDC, TCCR1A = 0xDC ; MEMORY MAPPED
261 .equ 0xDD, TCCR1B = 0xDD ; MEMORY MAPPED
262 .equ 0xDE, TCCR1C = 0xDE ; MEMORY MAPPED
263 .equ 0xDF, TCCR1A = 0xDF ; MEMORY MAPPED
264 .equ 0xE0, TCCR1B = 0xE0 ; MEMORY MAPPED
265 .equ 0xE1, TCCR1C = 0xE1 ; MEMORY MAPPED
266 .equ 0xE2, TCCR1A = 0xE2 ; MEMORY MAPPED
267 .equ 0xE3, TCCR1B = 0xE3 ; MEMORY MAPPED
268 .equ 0xE4, TCCR1C = 0xE4 ; MEMORY MAPPED
269 .equ 0xE5, TCCR1A = 0xE5 ; MEMORY MAPPED
270 .equ 0xE6, TCCR1B = 0xE6 ; MEMORY MAPPED
271 .equ 0xE7, TCCR1C = 0xE7 ; MEMORY MAPPED
272 .equ 0xE8, TCCR1A = 0xE8 ; MEMORY MAPPED
273 .equ 0xE9, TCCR1B = 0xE9 ; MEMORY MAPPED
274 .equ 0xEA, TCCR1C = 0xEA ; MEMORY MAPPED
275 .equ 0xEB, TCCR1A = 0xEB ; MEMORY MAPPED
276 .equ 0xEC, TCCR1B = 0xEC ; MEMORY MAPPED
277 .equ 0xED, TCCR1C = 0xED ; MEMORY MAPPED
278 .equ 0xEE, TCCR1A = 0xEE ; MEMORY MAPPED
279 .equ 0xEF, TCCR1B = 0xEF ; MEMORY MAPPED
280 .equ 0xF0, TCCR1C = 0xF0 ; MEMORY MAPPED
281 .equ 0xF1, TCCR1A = 0xF1 ; MEMORY MAPPED
282 .equ 0xF2, TCCR1B = 0xF2 ; MEMORY MAPPED
283 .equ 0xF3, TCCR1C = 0xF3 ; MEMORY MAPPED
284 .equ 0xF4, TCCR1A = 0xF4 ; MEMORY MAPPED
285 .equ 0xF5, TCCR1B = 0xF5 ; MEMORY MAPPED
286 .equ 0xF6, TCCR1C = 0xF6 ; MEMORY MAPPED
287 .equ 0xF7, TCCR1A = 0xF7 ; MEMORY MAPPED
288 .equ 0xF8, TCCR1B = 0xF8 ; MEMORY MAPPED
289 .equ 0xF9, TCCR1C = 0xF9 ; MEMORY MAPPED
290 .equ 0xFA, TCCR1A = 0xFA ; MEMORY MAPPED
291 .equ 0xFB, TCCR1B = 0xFB ; MEMORY MAPPED
292 .equ 0xFC, TCCR1C = 0xFC ; MEMORY MAPPED
293 .equ 0xFD, TCCR1A = 0xFD ; MEMORY MAPPED
294 .equ 0xFE, TCCR1B = 0xFE ; MEMORY MAPPED
295 .equ 0xFF, TCCR1C = 0xFF ; MEMORY MAPPED

```

... EOL