ATmega128 Timer/Counters

Prof. Ken Short

Timer/Counters Overview

- A hardware timer/counter is a counter that counts clock pulses from either an internal clock source or an external clock source, independently of program execution
- Elapsed time can be determined as the product of the number of counts during the interval in question multiplied by the timer/counter's clock period
- A typical timer/counter can be configured to count up or down, loaded with an initial value, started, and stopped under program control
- A typical timer/counter can generate interrupt requests when events, such as overflow, occur during counting

Common Uses of Timer/Counters

- Measure elapsed time
- Measure period or frequency of an external signal
- Generate periodic waveforms
- Generate hardware delays
- Generate pulses of specified widths
- Generate pulse width modulated waveforms

ATmega128 Timer Counters

- The ATmega128 has four independent timer/counters
- Timer/Counter0 is an 8-bit asynchronous timer/counter
- Timer/Counter1 is a 16-bit synchronous timer/counter
- Timer/Counter2 is an 8-bit synchronous timer/counter
- Timer/Counter3 is a 16-bit synchronous timer/counter

Timer/Counter Events

- If a timer/counter event occurs, its status flag, in the Timer Interrupt Flag Register (TIFR) or Extended Timer Interrupt Flag Register (ETIFR) register is set
- Timer overflow – If a *b*-bit counter counts beyond its maximum value (2^b-1) the timer overflow flag (TOVn) in TIFR is set
- Compare match – If the counter's value matches the value in the Output Compare Register (OCRn), the output compare flag (OCFn) in TIFR is set
- Input capture - A signal change at the input capture pin ICPn causes the current count to be stored in an input capture register and the input capture flag (ICFn) in TIFR to be set (Timer/Counter1 and 3 only, the 16-bit timer/counters only)

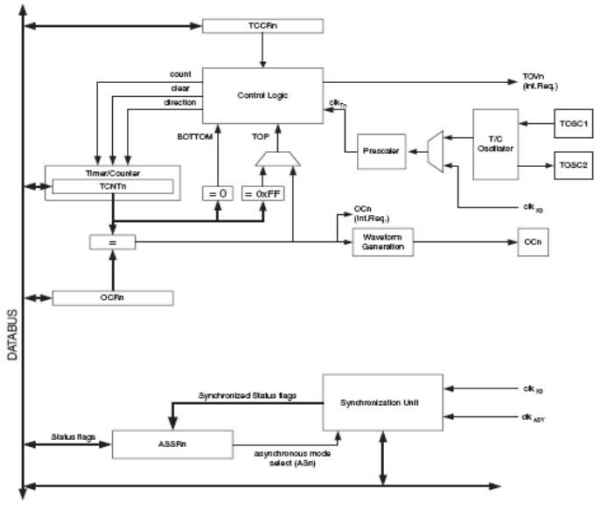
Ways to React to Timer/Counter Events

- The occurrence of a timer/counter event is indicated by the MCU setting the corresponding flag (interrupt request flag) in the Timer Interrupt Flag Register (TIFR) or Extended Timer Interrupt Flag Register (ETIFR)
- There are three different ways to monitor timer/counter events
  - Poll the interrupt request flags in TIFR or ETIFR
  - Enable the interrupt for a specific event
  - Change the level of an output pin (output compare events only)

Focus on Timer/Counter0

- The focus of the remainder of this lecture is on Timer/Counter0 since it is one of the simpler timer/counters. The more complex 16-bit Timer/Counters are the subject of the another lecture.
- The general concepts covered for Timer/Counter0 provide the background necessary to understand the data sheet and application note descriptions of this and other timer/counters

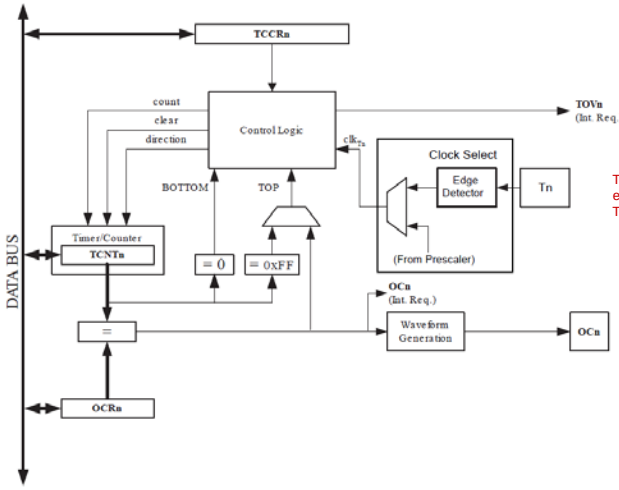
Timer/Counter0 Block Diagram



Timer/Counter 0 can operate from an external crystal (usually 32kHz) .

Timer/Counter 2 cannot.

Timer/Counter2 Block Diagram

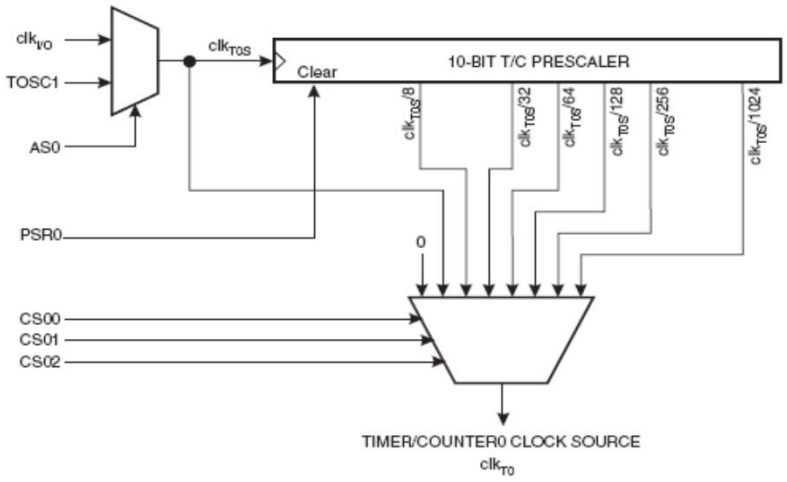


Timer/Counter 2 can count external events.
Timer/Counter 1 cannot.

Timer/Counter0 Clock

- Timer/Counter0 can be clocked:
  - Externally by an onboard oscillator controlled by an external crystal
  - Directly from the system clock
  - From the system clock divided by a prescaler
  - No clock – timer/counter is stopped

Prescalar for Timer/Counter0



TCCR0 – Timer/Counter0 Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

- **Bit 7 – FOC0: Force Output Compare**
- **Bit 6, 3 – WGM01:0: Waveform Generation Mode**
- **Bit 5:4 – COM01:0: Compare Match Output Mode**
- **Bit 2:0 – CS02:0: Clock Select**

Clock Select Bits

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

- Making CS02,CS01,CS00 = 0,0,0 stops the counter

2/21/2016 © Copyright Kenneth Short 2005 13

Resetting Prescalars Using PSR0 Bit in SFIOR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	–	ACME	PUD	PSR0	PSR321	SFIOR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

  Writing the TSM bit to 1 activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSR0 and PSR321 bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSR0 and PSR321 bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

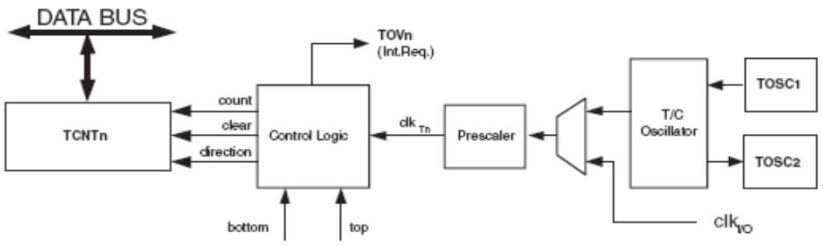
2/21/2016 © Copyright Kenneth Short 2005 14

Resetting Prescalars Using PSR0 Bit in SFIOR

- **Bit 1 – PSR0: Prescaler Reset Timer/Counter0**

  When this bit is one, the Timer/Counter0 prescaler will be reset. This bit is normally cleared immediately by hardware. If this bit is written when Timer/Counter0 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

Timer/Counter0 Counter Unit Block Diagram



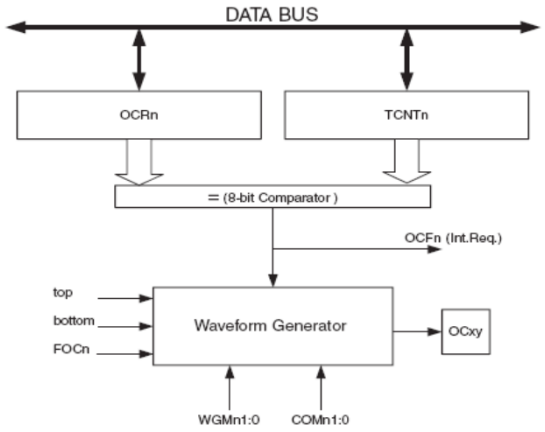
Signal Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes zero (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 OCR0 Register. The assignment is dependent on the mode of operation.

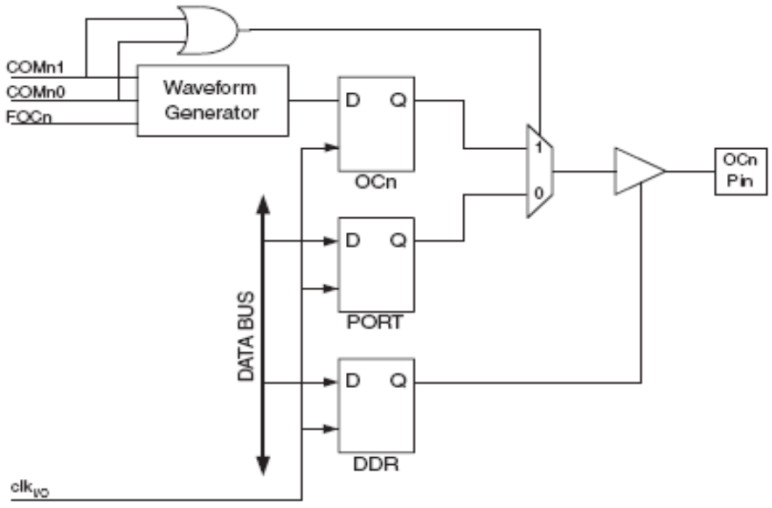
Internal Signals

count	Increment or decrement TCNT0 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{T0}	Timer/Counter clock.
top	Signalizes that TCNT0 has reached maximum value.
bottom	Signalizes that TCNT0 has reached minimum value (zero).

Output Compare Unit



Compare Match Output Unit



Compare Output Mode, Non-PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on Compare Match
1	0	Clear OC0 on Compare Match
1	1	Set OC0 on Compare Match

Timer/Counter0 Interrupts

16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow

Summary of Timer/Counter0 Registers

- TCCR0 – Timer/Counter0 Control Register
- TCNT0 – Timer/Counter0
- OCR0 – Output Compare Register 0
- ASSR – Asynchronous Status Register
- TIMSK – Timer/Counter Interrupt Mask Register
- TIFR - Timer/Counter Interrupt Flag Register
- SFIOR Special Function IO Register

TCCR0 – Timer/Counter0 Control Register

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	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	

- **Bit 7 – FOC0: Force Output Compare**
- **Bit 6, 3 – WGM01:0: Waveform Generation Mode**
- **Bit 5:4 – COM01:0: Compare Match Output Mode**
- **Bit 2:0 – CS02:0: Clock Select**

TCNT0 – Timer/Counter0 Register

Bit	7	6	5	4	3	2	1	0	
	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	

- Holds the count of the counter

OCR0 – Output Compare Register 0

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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	

- Holds the value to be compared with the current count

ASSR – Asynchronous Status Register

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3 – AS0: Asynchronous Timer/Counter0**

  When AS0 is written to zero, Timer/Counter0 is clocked from the I/O clock, clkI/O. When AS0 is written to one, Timer/Counter0 is clocked from a crystal Oscillator connected to the Timer Oscillator 1 (TOSC1) pin. When the value of AS0 is changed, the contents of TCNT0, OCR0, and TCCR0 might be corrupted.

- **Bit 2 – TCN0UB: Timer/Counter0 Update Busy**

  When Timer/Counter0 operates asynchronously and TCNT0 is written, this bit becomes set. When TCNT0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT0 is ready to be updated with a new value.

ASSR – Asynchronous Status Register 2

- **Bit 1 – OCR0UB: Output Compare Register0 Update Busy**

  When Timer/Counter0 operates asynchronously and OCR0 is written, this bit becomes set. When OCR0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR0 is ready to be updated with a new value.

- **Bit 0 – TCR0UB: Timer/Counter Control Register0 Update Busy**

  When Timer/Counter0 operates asynchronously and TCCR0 is written, this bit becomes set. When TCCR0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCCR0 is ready to be updated with a new value.

ASSR – Asynchronous Status Register 3

- If a write is performed to any of the three Timer/Counter0 Registers while its update busy flag is set, the updated value might get corrupted and cause an unintentional interrupt to occur.
- The mechanisms for reading TCNT0, OCR0, and TCCR0 are different. When reading TCNT0, the actual timer value is read. When reading OCR0 or TCCR0, the value in the temporary storage register is read.

TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

- **Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable**
- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

TIFR – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

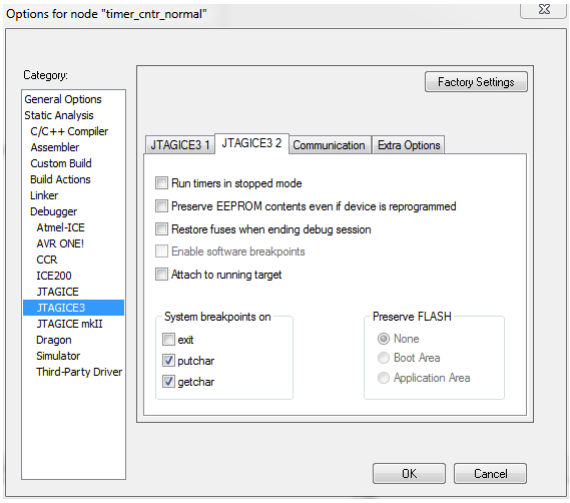
- **Bit 1 – OCF0: Output Compare Flag 0**
- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

SFIOR Special Function IO Register

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	–	ACME	PUD	PSR0	PSR321	SFIOR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**
- **Bit 1 – PSR0: Prescaler Reset Timer/Counter0**

Disabling Run timers in stopped mode



Modes of Operation

- The mode of operation of Timer/Counter0 is established by the values written to the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits
- There are four modes:
  - Normal mode
  - Clear Timer on Compare Match
  - Fast Pulse With Modulated (PWM) mode
  - Phase Correct PWM mode

Normal Mode

- **Normal Mode:** The simplest mode of operation is the normal mode (WGM01:0 = 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.

Counter 0 Test Program for Normal Mode

```
#include <iom128.h> //File with register addresses for ATmega128

// Normal Mode - Timer/Counter 0
// clock = 16 MHz, period = 1/16 Mhz = 0.0625 us
// Want a 10 ms delay
// prescaling by 1024 gives 64 us/count
// 10,000us/64us per count = 156.25 counts
// Need the 8- bit counter to count 156 times before overflow
// Load counter with 100
// Three ways to sense end of interval:
// TOV0, OCF0, and PB4 toggle
```


Counter 0 Test Program for Normal Mode (cont.)

```
unsigned char temp = 0; // variable used to breakpoint

int main(void)
{

//  OCR0 = 0;           // OCR0 not used
  DDRB = 1<<PB4;  //OC0 output

  TCNT0 = 100;  // it will take 156 counts to overflow
  // prescale by 1024, toggle pins OC0 on compare match
  // each count in the counter is 64 us

  TCCR0 = (1<<CS00) | (1<<CS01) | (1<<CS02) | (1<<COM00);

  while (1)
   temp += 1;

}
```


Counter 0 Test Program for Normal Mode with Timer Overflow Interrupt

```
unsigned char temp = 0; // variable used to breakpoint

int main(void)
{
//  OCR0 = 0;           // OCR0 not used
  DDRB = 1<<PB4;  //OC0 output

  TCNT0 = 100;  // it will take 156 counts to overflow
  // prescale by 1024, toggle pins OC0 on compare match
  // each count in the counter is 64 us

  TIMSK = (1 << TOIE0); // enable timer 0 overflow interrupt

   __enable_interrupt(); // global interrupt enable

  TCCR0 = (1<<CS00) | (1<<CS01) | (1<<CS02) | (1<<COM00);

  while (1)
   temp += 1;

}
```

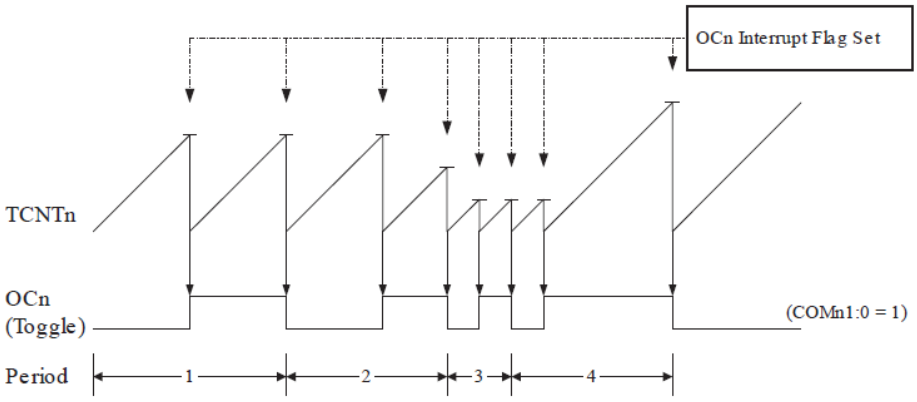

Counter 0 Test Program for Normal Mode with Timer Overflow Interrupt (cont.)

```
#pragma vector = TIMER0_OVF_vect       // insert interrupt vector
__interrupt void timer_0_overflow(void)

{
  // If counter is not stopped its next "count" will be 1024 system
  // clocks later.

  // TCCR0 = (1<<COM00);  // Stop the timer CS00 = CS01 = CS02 = 0

  // Do whatever task was to be done after the delay.
  // You could reload TCNT0 and start the counter again here
  // to repeat the delay or just leave it stopped, if only one
  //  occurence of the delay was desired.

}
```


Clear Timer on Compare or CTC mode

- In this mode (WGM01:0 = 2), the OCR0 Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0. The OCR0 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency.

CTC Mode Timing Diagram



Counter 0 Test Program for CTC Mode

```
// Test program for Timer/Counter 0 - Clear Timer on Compare
// Match Mode
// clock = 16 MHz, period = 1/16 Mhz = 0.0625 us
// Toggle OC0 to generate square wave
// fOC0 = fclk_IO / 2 * N * (1 + OCR0), where N is prescalar
// OCR0          fOC0 kHz        Meas (DSO-X 3012A)
// 16            470.588         470.6
// 32            242.424         242.41
// 64            123.076         123.07
// 128           062.015         062.012
// Why does scope give more percesion for some measurements?
```

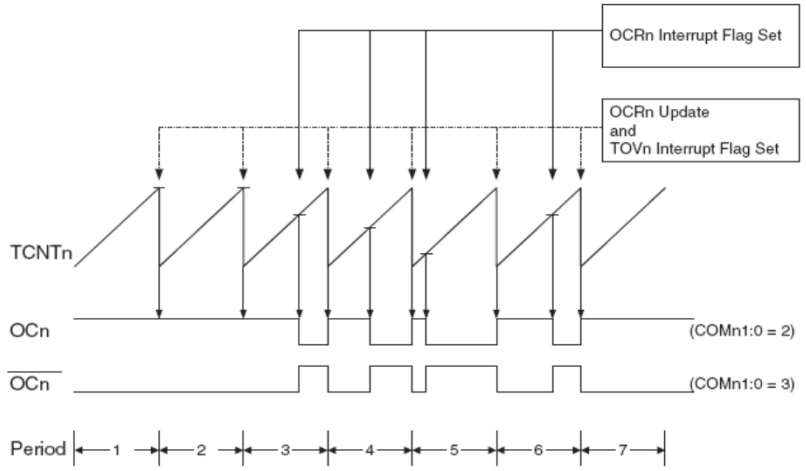
Counter 0 Test Program for CTC Mode

```
#include <iom128.h> //File with register addresses for ATmega128

int main(void)

{

  OCR0 = 16;
  DDRB = 1<<PB4;  //OC0 output

  TCNT0 = 0;
  // ctc mode, no prescale, toggle pins OC0 (PB4) on compare match
  TCCR0 = (1<<WGM01) | (1<<CS00) | (1<<COM00);

  while (1) ;

}
```


Fast Pulse Width Modulation Mode

- The fast Pulse Width Modulation or fast PWM mode (WGM01:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the output compare (OC0) is cleared on the compare match between TCNT0 and OCR0, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM.

Fast PWM Mode Timing Diagram



Compare Output Mode, Fast -PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on Compare Match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on Compare Match, clear OC0 at BOTTOM, (inverting mode)

Counter 0 Test Program for Fast PWM Mode

```
// Timer/Counter 0 Test Program - Fast PWM Mode
// fOC0 = fclkIO / (N * 256), where N is prescalar
// clock = 16 MHz, period = 1/16 Mhz = 0.0625 us
// Prescalar = 1, fOC0 = 16/256 = 62.5 kHz (meas 62.496)
// generating a fast PWM signal
// OCR0          Duty Cycle %
// 64            25
// 128           50
// 192           75
```

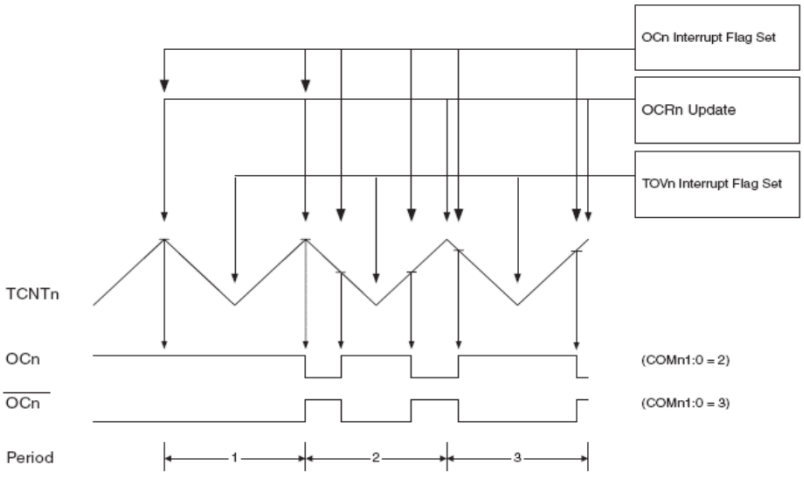

Counter 0 Test Program for Fast PWM Mode

```
#include <iom128.h> //File with register addresses for ATmega128

int main(void)
{

  OCR0 = 192;
  DDRB = 1<<PB4;  //OC0 output
  TCNT0 = 0;
  // fpwm mode, no prescale, non-inverting mode
  TCCR0 = (1<<WGM01) | (1<<WGM00) | (1<<CS00) | (1<<COM01);

while (1) ;

}
```


Phase Correct PWM Mode

- The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In noninverting Compare Output mode, the output compare (OC0) is cleared on the compare match between TCNT0 and OCR0 while counting up, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

Phase Correct PWM Mode Timing Diagram



Compare Output Mode, Phase Correct PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on Compare Match when up-counting. Set OC0 on Compare Match when downcounting.
1	1	Set OC0 on Compare Match when up-counting. Clear OC0 on Compare Match when downcounting.

Waveform Generation Mode Bits

Mode	WGM01[1] (CTC0)	WGM00[1] (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Compare Output Mode Bits

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

References

- *AVR130: Setup and Use the AVR Timers*, Atmel application note (on Blackboard).
- ATMega128 Data Sheet, pages 194 through 216

