CG1112

Engineering Principles and Practices II for CEG

Week 6 Studio 2

Timers

colint@NUS.edu.sg

National University of Singapore

School of Computing

Timer Programming

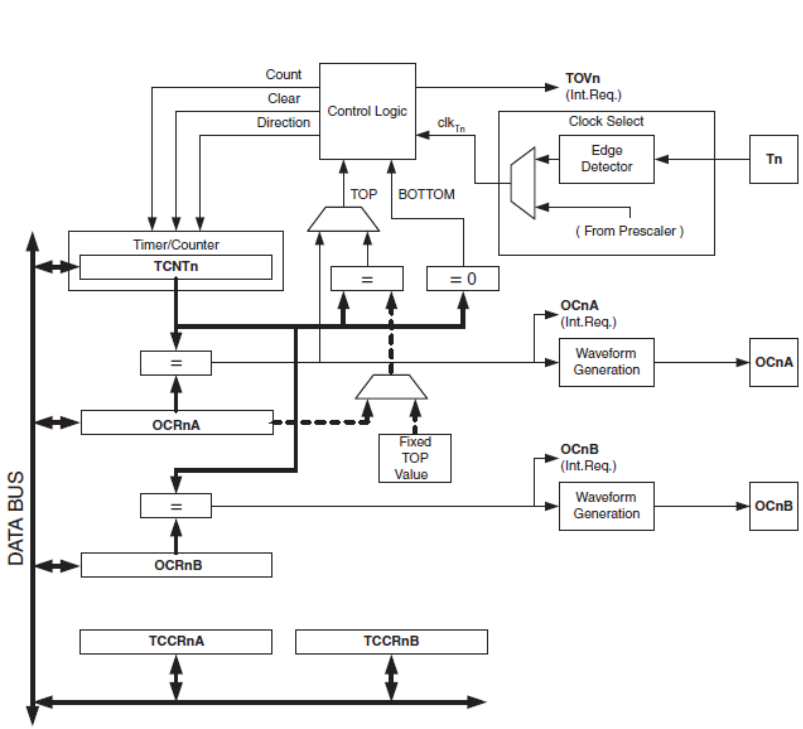
- **Timers are important in real-time systems:**
  - They allow us to read sensors or write to actuators at precise times.
  - They ensure that time-sensitive algorithms (e.g. the PID control algorithm) run at the correct timing.
  - Timers are often needed to switch between tasks in a multi-tasking OS (more in CG2271).
  - Timers are used to generate analog output signals through pulse-width modulation (PWM).

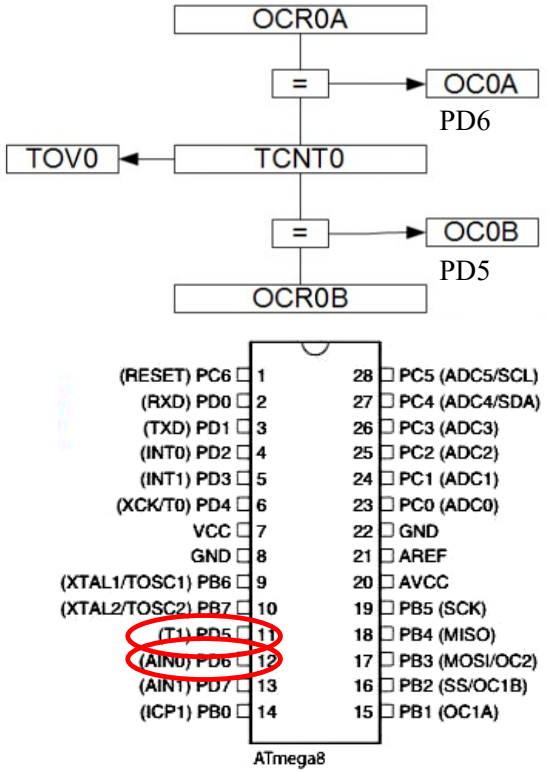
Timers in the Atmega AVR

- **There are 3 timers available on the Atmega328:**
  - Two 8-bit timers, Timers 0 and 2.
    - ✓**Counts from 0 to 255, then back to 0, etc.**
    - ✓**Triggers an interrupt (TOV0 or TOV2) whenever the counter rolls over from 255 to 0. Also triggers interrupts if counter matches a value.**
    - ✓**One 16-bit timer, Timer 1.**
    - ✓**Counts from 0 to 65535 and back to 0, etc.**
    - ✓**Triggers an interrupt (TOV1) whenever the counter rolls over from 65535 to 0. Also triggers interrupts if counter matches a value.**
  - In this lecture we will focus on Timer 0.
    - ✓**Techniques for Timer 1 and 2 are identical, just with different registers.**
  - We will also look at how to generate waveforms on the OC0A and OC0B pins (corresponding to PD5 and PD6) on the AVR.
    - ✓**This is not usually necessary but we do it just for demo purposes.**

Timer 0/Timer 2 Block Diagram

Diagram Credits: Midity. http://www.midity.com





Timer 0 Programming
CTC Mode

- **The (second) simplest timer mode is CTC (Clear Timer on Compare) mode.**
  - Counter TCNT0 starts at 0, counts to a preset "top value" *V*, then rolls over back to the 0. Process repeats.
  - A "timer match" interrupt is triggered each time TCNT0 matches OCR0A or OCR0B.
- **This is very useful for generating an interrupt at fixed times, e.g. every 1 ms.**
  - Perfect for devices that need to be read at fixed times.
  - Perfect also for programs that need to "sleep" for fixed times.

Timer 0 Programming
CTC Mode

- **To program CTC mode in Timer 0, you need to:**
  - Decide on the time period that you want, e.g. 1 ms.
  - From this:
    - ✓**Decide on a prescaler value *P* and count value *V*.**
    - ✓**These are done simultaneously, subject to I/O clock rate and register size restrictions.**
  - Configure the registers.
    - ✓**Set up the ISR for OC0A (Timer 0 Output Compare Match A).**
    - ✓**Set the initial timer value of 0 into TCNT0.**
    - ✓**Set the top count value *V* into OCR0A.**
    - ✓**Set CTC mode in TCCR0A.**
    - ✓**Set the prescaler into TCCR0B to start the timer.**

Timer 0 Programming
CTC Mode

- Deciding on a prescaler value P
  - ✓**This determines the resolution *res* of the timer. I.e. the amount of time between "increments" of the counter.**
  - ✓**The prescalar frequency is given by:**

$$res = \frac{F_{clock}}{P}$$

The resolution for the prescalar is given by the 1 / res (see next slide).

NUS National University of Singapore
School of Computing

Timer 0 Programming
CTC Mode

- The prescalar is chosen using bits 2 to 0(CS02:00) of TCCR0B:

$$\text{Resolution} = \frac{1}{(F_{clk} / P)}$$

CS02	CS01	CS00	Prescalar P	Resolution (F_{clk}=16 MHz)
0	0	0	Stops the timer	-
0	0	1	1	0.0625 microseconds
0	1	0	8	0.5 microseconds
0	1	1	64	4 microseconds
1	0	0	256	16 microseconds
1	0	1	1024	64 microseconds
1	1	0	External clock on T0. Clock on falling edge.	-
1	1	1	External clock on T0. Clock on rising edge	-

Timer 0 Programming
CTC Mode

- **The TIMER0_COMPA (TIMER0_COMPA_vect) interrupt is triggered whenever TCNT0 reaches the top timer value *V* and rolls back to 0.**
- **We first decide the period T_{cycle}, which is the time which the timer triggers the timer interrupt.**
- **We need to decide what *V* is!**

$$V = \frac{T_{cycle}}{res}$$

- **We want our TIMER0_COMPA interrupt to be triggered every 1 ms, or 1000 microseconds. Using the formula above, the possible values are shown in the table on the next page.**

Timer 0 Programming
CTC Mode

- **The possible *V* are:**

Prescaler	Resolution (16 MHz Clock)	Count *V* (for 1 ms or 1000 microseconds)
1	0.0625 microseconds	16000
8	0.5 microseconds	2000
64	4 microseconds	250
256	16 microseconds	62.5
1024	64 microseconds	15.63

- **It is not possible to load 16000 or 2000 into 8-bit registers. The possible values are 250, 62.5 and 15.63.**
  - We choose the largest value 250 because it gives us the best possible resolution of 4 microseconds.
    - ✓**Better resolution = more accurate timings.**
- **This gives us a prescaler value of 64.**

Timer 0 Programming
CTC Mode

- **Now that we have chosen *P* and *V*, we can begin configuring the timer.**
  - Set the initial timer value in the Timer 0 Control register TCNT0. We use an initial value of 0.

```
TCNT0=0;
```

  - Set the timer value *V* into the Output Compare Register OCR0A. For 250 steps, *V=249*, since we start from 0.

```
OCR0A=249;
```

Timer 0 Programming
CTC Mode

- Set up Timer/Counter Control Register TCCR0A:

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	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	

https://www.darpa.mil/news-events/2020-01-10

- **COM0A1 and COM0A0 control what to do with OC0A each time we hit the top value *V* in OCR0A:**

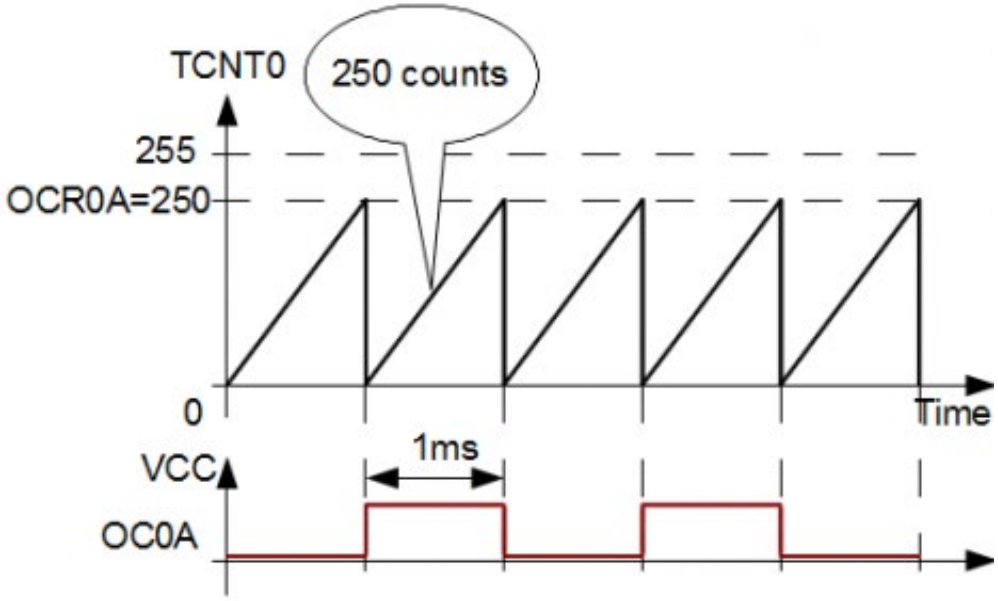
Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

- **We want to toggle OC0A on match, so we choose CM0A1 and COM0A0 to "01" respectively.**

Timer 0 Programming
CTC Mode

Diagram Credits: Midity. http://www.midity.com/



Timer 0 Programming
CTC Mode

- Set up Timer/Counter Control Register TCCR0A:
  - **The WGM00, WGM01 and WGM02 bits control the waveform generation:**

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

  - **We will use CTC mode so WGM01 and WGM00 are “01”. WGM02 is in TCCR0B.**

Timer 0 Programming
CTC Mode

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	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	

- **Taken together, we will program TCCR0A using the following statement:**

```
TCCR0A=0b01000010;
```

- **We need to enable the OCIE0A interrupt (output compare match A) by writing a 1 to bit 1 of the Timer/Counter Interrupt Mask Register TIMSK0:**

```
TIMSK0|=0b10;
```

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer 0 Programming
CTC Mode

- **Finally we will start the timer by writing selecting the prescaler value *P* in TCCR0B, using bits CS02:CS00.**
  - We want a prescaler value of 64 so we use 011.
  - We also want WGM02 to be 0 (see previous slide).

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	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	

```
TCCR0B=0b00000011;
```

  - Note: Setting CS02:CS00 bits to anything other than 000 automatically begins the timer.
  - We must also enable global interrupts using the sei(); function call.

Timer 0 Programming
CTC Mode

- **We will now look at an example program that:**
  - Increments a counter every millisecond.
  - On every 100th increment, it toggles an LED at pin 13 on or off.
    - ✓**Effectively causes the LED to blink on and off every 100 ms.**

Timer 0 Programming
CTC Mode Example 1

```
#include <avr/io.h>
#include <avr/interrupt.h>

int count;
//Initialize Timer0
void InitTimer0(void)
{
        //Set Initial Timer value
        TCNT0=0;
        //Place TOP timer value to Output compare register
        OCR0A=249;

        //Set CTC mode
        //      and make toggle PD6/OC0A pin on compare match
        TCCR0A=0b01000010;
        // Enable interrupts.
        TIMSK0|=0b10;
}
```

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	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	

Timer 0 Programming
CTC Mode Example 1

```
void StartTimer0(void)
{
        //Set prescaler 64 and start timer
        TCCR0B=0b00000011;
        // Enable global interrupts
        sei();
}

void ledOn()
{
        PORTB|=0b00100000;
}

void ledOff()
{
        PORTB&=0b11011111;
}
```

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	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	

Timer 0 Programming
CTC Mode Example 1

```
// Toggle the LED
void toggleLED()
{
        static state=1;

        if(state==1)
        {
                ledOn();
                state==0;
        }
        else
        {
                ledOff();
                state=1;
        }
}
```

Timer 0 Programming
CTC Mode Example 1

```
// Set up the ISR for TOV0A
ISR(TIMER0_COMPA_vect)
{
        count++;
        if((count % 100)==0)
                toggleLED();
}
int main(void)
{
        // Set up LED on pin 13 (port B pin 5) for output.
        DDRB|=0b00100000;
        count=0;
        InitTimer0();
        StartTimer0();
    while(1)
    {           // Do nothing.
    }
}
```

Timer Programming in General

- **Programming the other times (timer 1 and 2) is exactly the same, but with different register names.**
  - E.g. use TCCR2A, TCCR2B, TCNT2, OCR2A, etc.
- **Timer 1 is a 16-bit timer, giving you better resolution.**

Prescaler	Resolution (16 MHz Clock)	Count *V* (for 1 ms or 1000 microseconds)
1	0.0625 microseconds	16000
8	0.5 microseconds	2000
64	4 microseconds	250
256	16 microseconds	62.5
1024	64 microseconds	15.63

- We can load a value of 16000 into OCR1AH/OCR1AL to get a 1ms count, allowing use to use a prescaler of 1 and a resolution of 0.0625 microseconds.
  - ✓**This can potentially lead to more accurate timings.**