EIE3105: Timer Programming (Chapter 10)

Dr. Lawrence Cheung

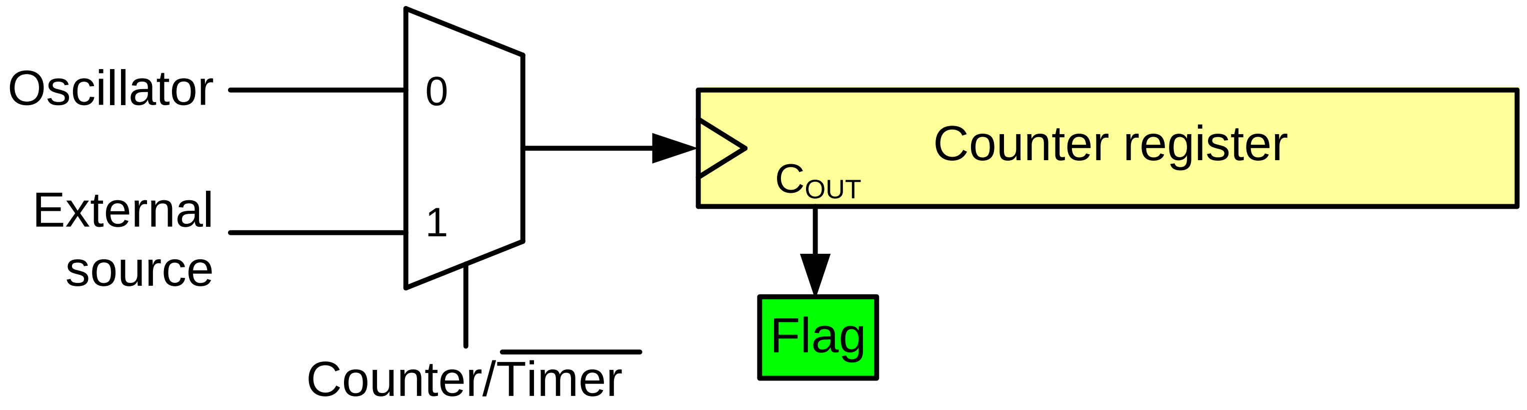
Semester 1, 2021/22

Topics

- Programming Timers 0, 1 and 2
- Counter Programming

A generic timer/counter

- Delay generating
- Counting
- Wave-form generating
- Capturing

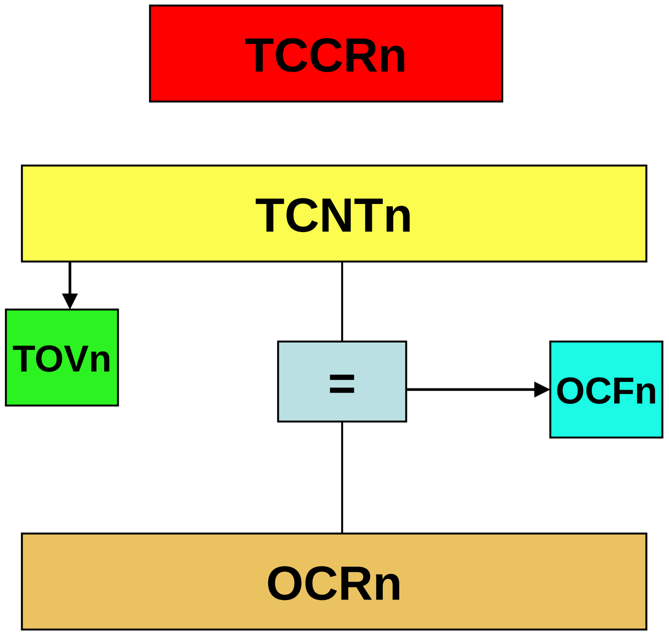


Timers in AVR

- 1 to 6 timers
  - 3 timers in ATmega32
- 8-bit and 16-bit timers
  - two 8-bit timers and one 16-bit timer in ATmega32

Timers in AVR

- Timer registers
  - TCNTn (Timer/Counter register)
  - TOVn (Timer Overflow flag)
  - TCCRn (Timer Counter control register)
  - OCRn (output compare register)
  - OCFn (output compare match flag)
- All of the timer registers are byte-addressable I/O registers.



Timer 0

- Read a timer register

```
unsigned char x;
x = TCNT2;
```

- Write into a timer register

```
TCNT2 = 0x5A;
```

- TOV0 in TIFR (Time/counter Interrupt Flag Register) will be used in Timer 0 programming.

7	6	5	4	3	2	1	0	Bit
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
RW	RW	RW	RW	RW	RW	RW	RW	R/W
0	0	0	0	0	0	0	0	Initial

Timer 0

- TCCR0 (Timer/Counter Control Register)
  - FOC0: Force compare match – Act as a wave generator if a compare match has occurred.
  - COM00, COM01: Compare Output Mode – Control the waveform generator.

7	6	5	4	3	2	1	0	Bit
W	RW	RW	RW	RW	RW	RW	RW	R/W
0	0	0	0	0	0	0	0	Initial
FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**	TCCR0

FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**	TCCR0

Clock Selector (CS)

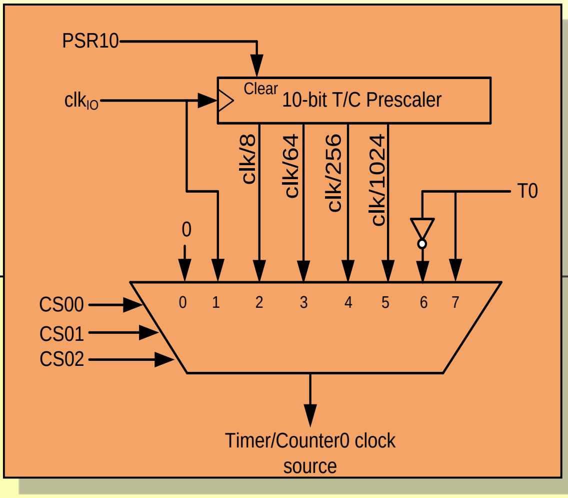
CS02	**CS01**	**CS00**	**Comment**
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

TCCR0

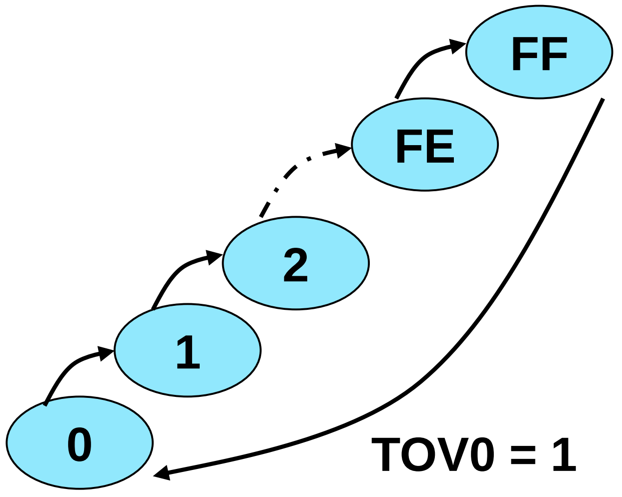
Timer Mode (WGM)

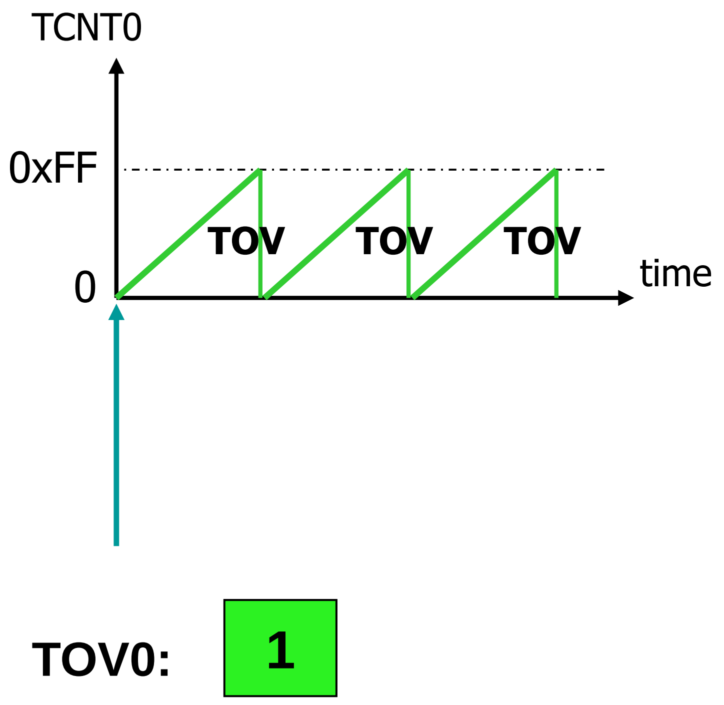
WGM00	WGM01	Comment
0	0	Normal
0	1	CTC (Clear Timer on Compare Match)
1	0	PWM, phase correct
1	1	Fast PWM



Normal mode

- TOV0: The flag is set when the counter overflows, going from $FF to $00.





Timer 0

- Steps to program Timer 0 in Normal mode

1. Load the TCNT0 register with the initial count value.
2. Load the value into the TCCR0 register, indicating which mode (8-bit or 16-bit) is to be used and the prescaler option.
3. Keep monitoring the timer overflow flag (TOV0) to see if it is raised. Get out of the loop when TOV0 becomes high.
4. Stop the timer by disconnecting the clock source.
5. Clear the TOV0 flag for the next round.
6. Go back to Step 1 to load TCNT0 again.

Timer 0

- Example: Write a C program to create a square wave of 50% duty cycle (with equal portions high (14 cycles) and low (14 cycles)) on the PORTB.5.

```
DDRB = 1<<5;
PORTB &= ~(1<<5);  //PB5=0
while (1)
{
  TCNT0 = 0xF2;
  TCCR0 = 0x01;
  while((TIFR&(1<<TOV0))==0);
  TCCR0 = 0;
  TIFR = (1<<TOV0);
  PORTB = PORTB^(1<<5);
}
```

Timer 0

- Find the value of the timer register:

1. Calculate the period of the timer clock, T = 1/f where f is the clock frequency.
2. Divide the desired time delay by T.
3. Perform 256 – n, where n is the decimal value.
4. Convert the result of Step 3 to hex, where xx is the initial hex value to be loaded into the timer's register.
5. Set TCNT0 = xx.

Timer 0

- Example: Find the value of the timer register if the time delay is 6.25 μs and XTAL = 8 MHz.

1. T = 1 / 8 MHz = 0.125 μs.
2. 6.25 μs / 0.125 μs = 50.
3. 256 – 50 = 206.
4. 206 = 0xCE.
5. TCNT0 = 206; or TCNT0 = 0xCE; (C statement)

Generating large delays

- To get the largest delay, we make TCNT0 zero. This will count up from 00 to 0xFF and then roll over to zero. What if that is not enough?
- Options:
  - Using loop
  - Prescaler
  - Bigger counters

Generating large delays

- Example: Write a C program to toggle only the PORTB.4 continuously every 70 μs. Use Timer 0, Normal mode, and 1:8 prescaler to create the delay. Assume XTAL = 8 MHz.
  - Timer value = 70 μs $\times$ 8 MHz / 8 = 70
  - TCNT0 = 256 – 70 = 186

Generating large delays

```
#include “avr/io.h”
void T0Delay();
int main(){
   DDRB = 0xFF; //PORTB output port
   while(1){
        T0Delay(); //Timer 0, Normal mode
        PORTB = PORTB ^ 0x10; //toggle PORTB.4
   }
}
void T0Delay(){
   TCNT0 = 186; //load TCNT0
   TCCR0 = 0x02; //Timer 0, Normal mode, 1 : 8 prescaler
   while ((TIFR&(1<<TOV0))==0); //wait for TOV0 to roll over
   TCCR0 = 0; //turn off Timer 0
   TIFR = 0x1; //clear TOV0
}
```

Class Exercise 1

- If the value of the timer register is 0x10 and the prescaler 256 is used, what is the time delay in seconds? Assume XTAL = 8 MHz.

Class Exercise 1 (Your work)

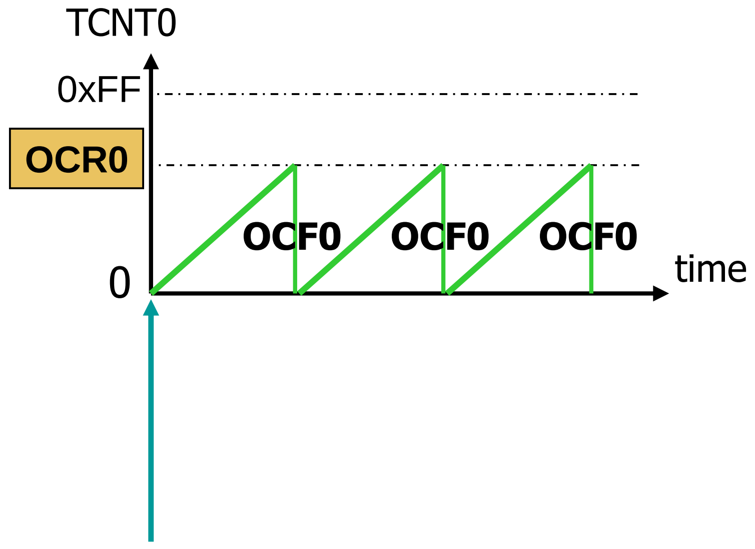
Class Exercise 1 (Answer)

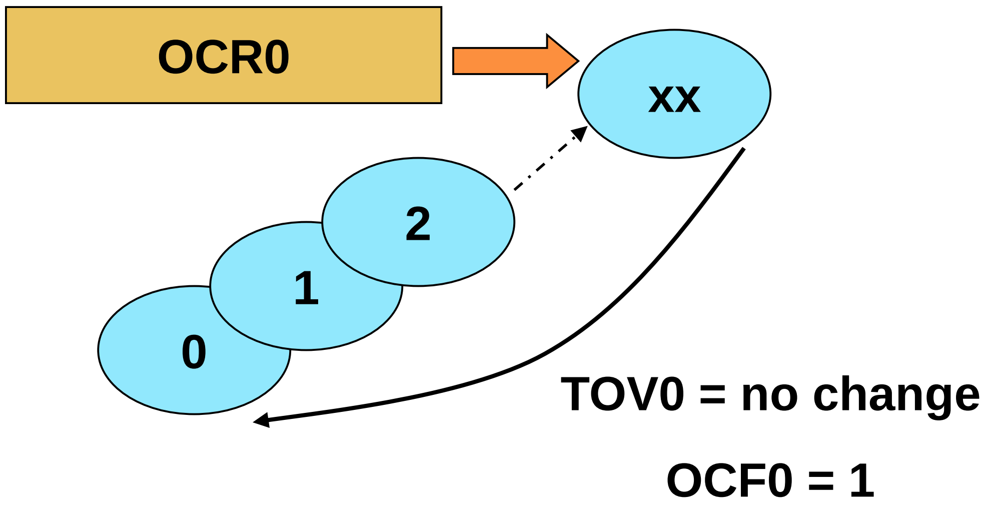
CTC mode programming

- CTC = Clear Timer on Compare match
- In CTC mode, the timer counts up until the content of the TCNT0 register becomes equal to the content of OCR0 (Compare match occurs).
- Then the timer will be cleared and the OCF0 flag will be set when the next clock occurs.
- The TOV0 and OCF0 flags are located in the TIFR (Timer/counter Interrupt Flag register).

CTC mode programming

- OCF0: The flag is set when the counter overflows, going from \$OCR0 to \$00.





TOV0: 0

OCF0: 1

CTC mode programming

- Write a program to create a square wave of 50% duty cycle (with equal portions high (5 μs) and low (5 μs)) on the PORTB.3. Assume that XTAL = 10 MHz.
  - For a square wave with T = 10 μs we must have a time delay of 5 μs. Because XTAL = 10 MHz, the counter counts up every 0.1 μs. This means that we need 5 μs / 0.1 μs = 50 clock pulses. Therefore, we have OCR0 = 49.

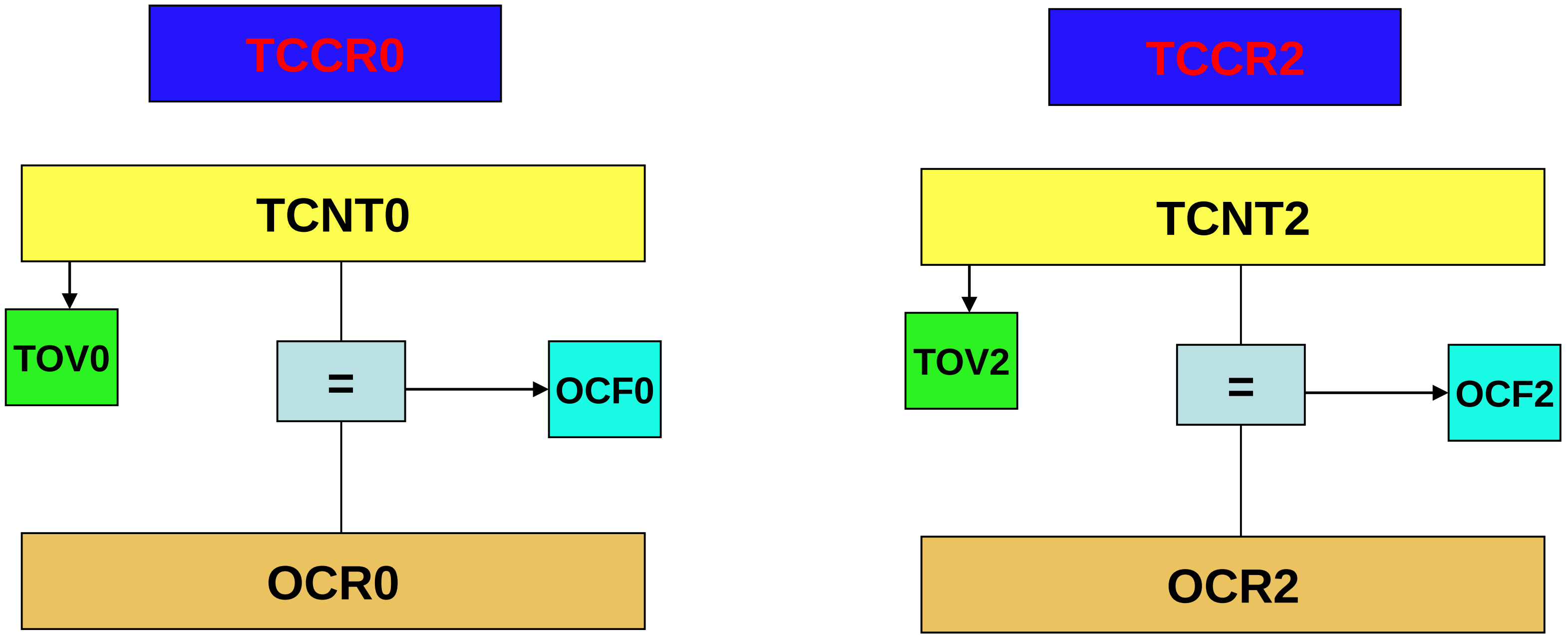
CTC mode programming

FOC0	WGM00	COM01	COM00	WGM01	**CS02**	**CS01**	**CS00**	TCCR0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

```
DDRB |= 1<<3;
PORTB &= ~(1<<3);
while (1)
{
  OCR0 = 49;
  TCCR0 = 0x09;
  while((TIFR&(1<<OCF0))==0);
  TCCR0 = 0; //stop timer0
  TIFR = 0x02;
  PORTB.3 = ~PORTB.3; // bit operation?
}
```

Timer 2

- Very similar to Timer 0



OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

Timer 2

- Differences

1. Timer 2 can be used as a real time counter. To do so, connect a crystal of 32.768 kHz to the TOSC1 and TOSC2 pins of AVR and set the AS2 bit.
   - When AS2 is zero, Timer 2 is clocked from clk$_{I/O}$. When it is set, Timer 2 works as a real time counter.

7	6	5	4	3	2	1	0
-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB

Timer 2

2. The value of CS bits in Timer 0 and Timer 2 are different.

CS02	CS01	CS00	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock (falling edge)
1	1	1	External clock (rising edge)

CS22	CS21	CS20	Comment
0	0	0	Timer/Counter stopped
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 32
1	0	0	clk / 64
1	0	1	clk / 128
1	1	0	clk / 256
1	1	1	clk / 1024

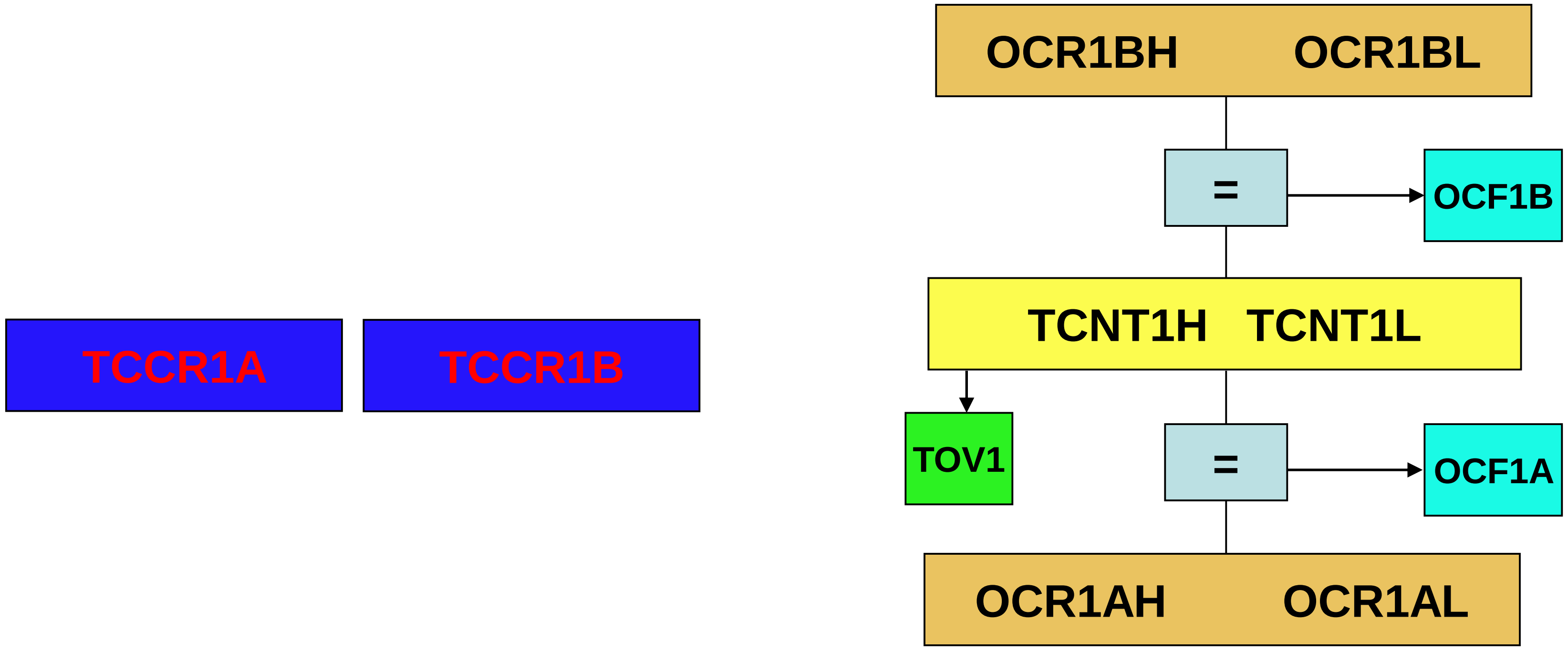
Timer 2

- Example: Using a prescaler of 64, write a C function to generate a delay of 1920 μs. Assume XTAL = 8 MHz.
  - Timer value = 1920 μs × 8 MHz / 64 = 240

```
void T2Delay()
{
   TCNT2 = 0x10; // 256 - 240 = 16 = 0x10
   TCCR2 = 0x04; // Timer 2, Normal mode, int clk, prescaler 64
   while((TIFR&(1<<TOV2))==0); // wait for roll over
   TCCR2 = 0; // turn off Timer 2
   TIFR = 0x40; // clear TOV2
}
```

Timer 1

- Timer 1 is a 16-bit timer.

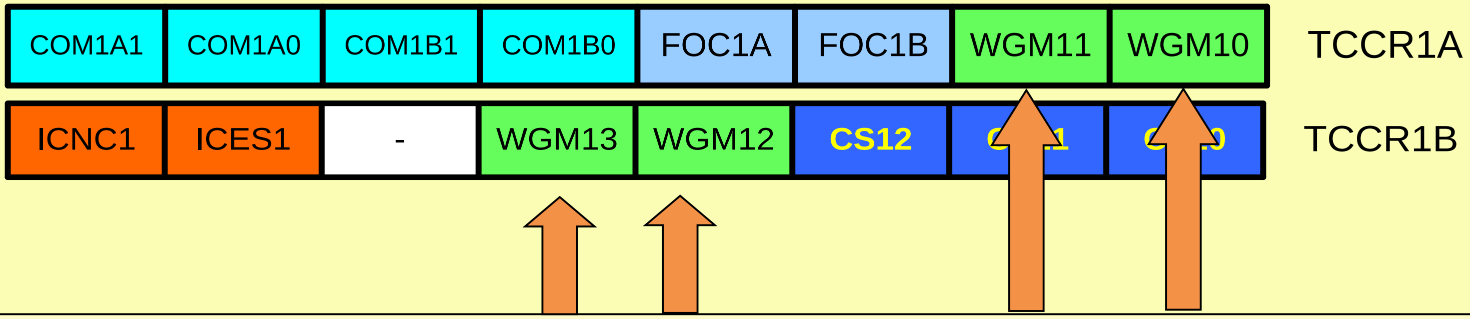


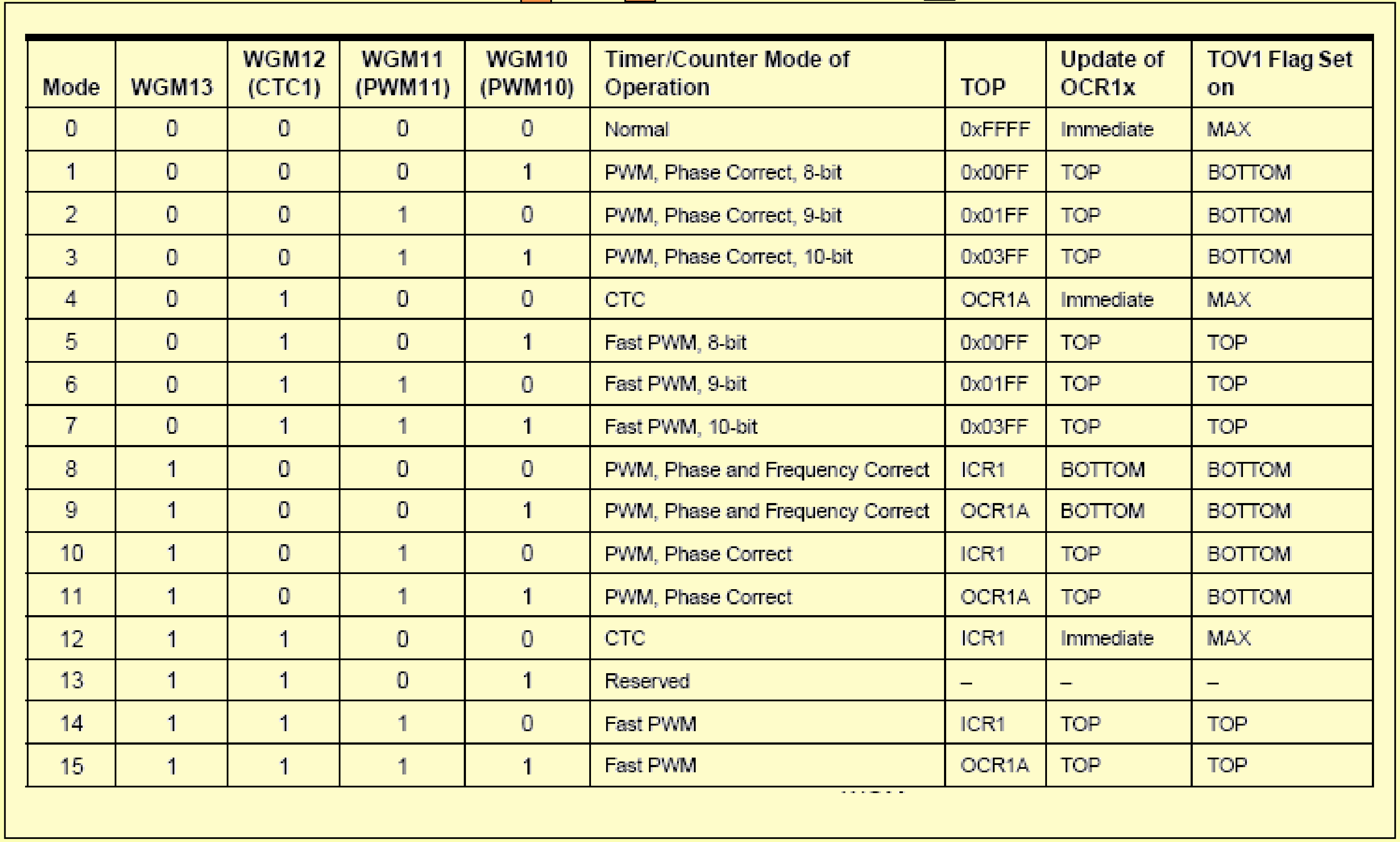
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

Clock Selector (CS)

CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge





Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Timer 1

- Example: write a C program that toggles PORTB.4 bit continuously every 2 ms. Use Timer 1, Normal mode and no prescaler to create the delay. Assuming XTAL = 8 MHz.
  - Timer value = 2 ms $\times$ 8 MHz = 16,000
  - TCNT1 = 65,536 – 16,000 = 49,536 = $C180

Timer 1

```
#include “avr/io.h”
void T1Delay();
int main(){
   DDRB = 0xFF; //PORTB output port
   while(1){
        T1Delay(); //Timer 1, Normal mode
        PORTB = PORTB ^ (1<<PB4); //toggle PORTB.4
   }
}
void T1Delay(){
   TCNT1H = 0xC1; //TCNT1 = 0xC180
   TCNT1L = 0x80;
   TCCR1A = 0x00; //Normal mode
   TCCR1B = 0x01; //Normal mode, no prescaler
   while ((TIFR&(1<<TOV1))==0); //wait for TOV1 to roll over
   TCCR1B = 0; //turn off Timer 1
   TIFR = 0x1<<TOV1; //clear TOV1
}
```

Counter Programming

- Example: Assume that a 1-Hz external clock is being fed into pin T1 (PB1). Write a C program for Counter 1 in rising edge mode to count the pulses and display the TCNT1H and TCNT1L on PORTD and PORTC, respectively.

Counter Programming

```
#include “avr/io.h”
int main(){
   // Don’t touch PB1, counter programming setting will do
   DDRC = 0xFF;        //PORTC as output
   DDRD = 0xFF;        //PORTD as output
   TCCR1A = 0x00;      //output clock source
   TCCR1B = 0x07;      //output clock source
   TCNT1H = 0x00;      //set count to 0
   TCNT1L = 0x00;      //set count to 1
   while(1)            //repeat forever
   {
       do
       {
             PORTC = TCNT1L;
             PORTD = TCNT1H;     //place value on pins
       }while ((TIFR & (0x1<<TOV1)) == 0); // wait for TOV1
       TIFR = 0x1<<TOV1;   //clear TOV1
   }
}
```

Class Exercise 2

- Assume that a 1 Hz clock pulse is fed into pin T0 (PB0). Use the TOV0 flag to extend Timer 0 to a 16-bit counter and display the counter on PORTC and PORTD. Note that it should operate in rising edge mode.

Class Exercise 2 (Your work)

Class Exercise 2 (Answer)

ATmega328p

- ATmega328p
  - Timer 0: TCCR0A, TCCR0B
  - Timer 1: TCCR1A, TCCR1B
  - Timer 2: TCCR2A, TCCR2B

Reference Readings

- Chapter 9 – *The AVR Microcontroller and Embedded Systems : Using Assembly and C*, M. A. Mazidi, S. Naimi, and S. Naimi, Pearson, 2014.

End