ECE3411 – Spring 2019

Lecture 8

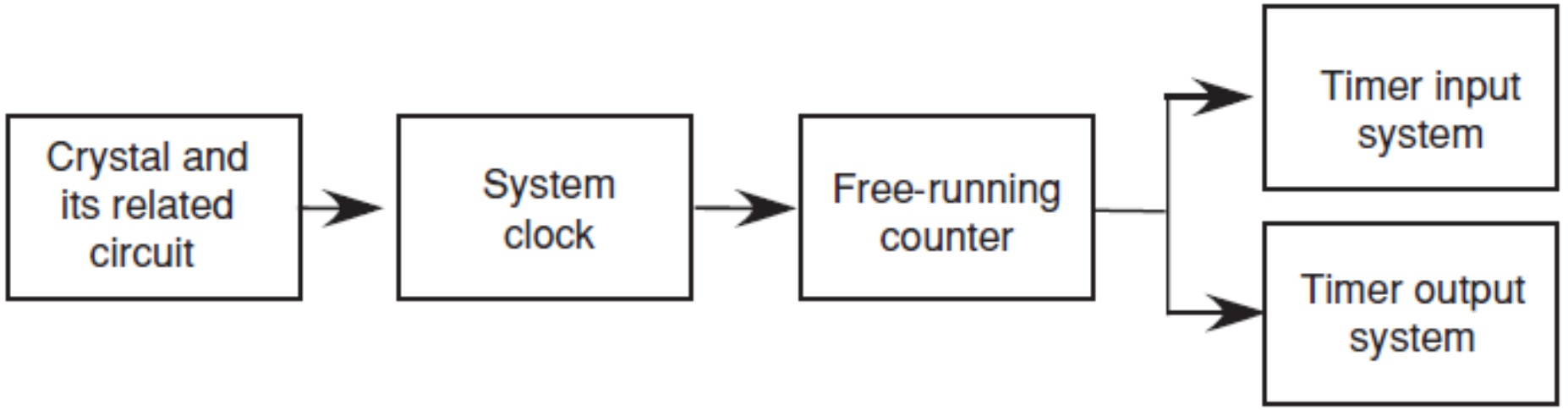
Timers

Sung Yeul Park
Department of Electrical & Computer Engineering
University of Connecticut
Email: sung_yeul.park@uconn.edu

Copied from Lecture 3b, ECE3411 – Fall 2015, by
Marten van Dijk and Syed Kamran Haider

Based on the Atmega328PB datasheet

Timer/Counter Subsystem



- Turn on or off an external device at a programmed time
- Generate complex digital waveforms with varying pulse widths to control the speed of a DC motor

Terminology for Timer Subsystem

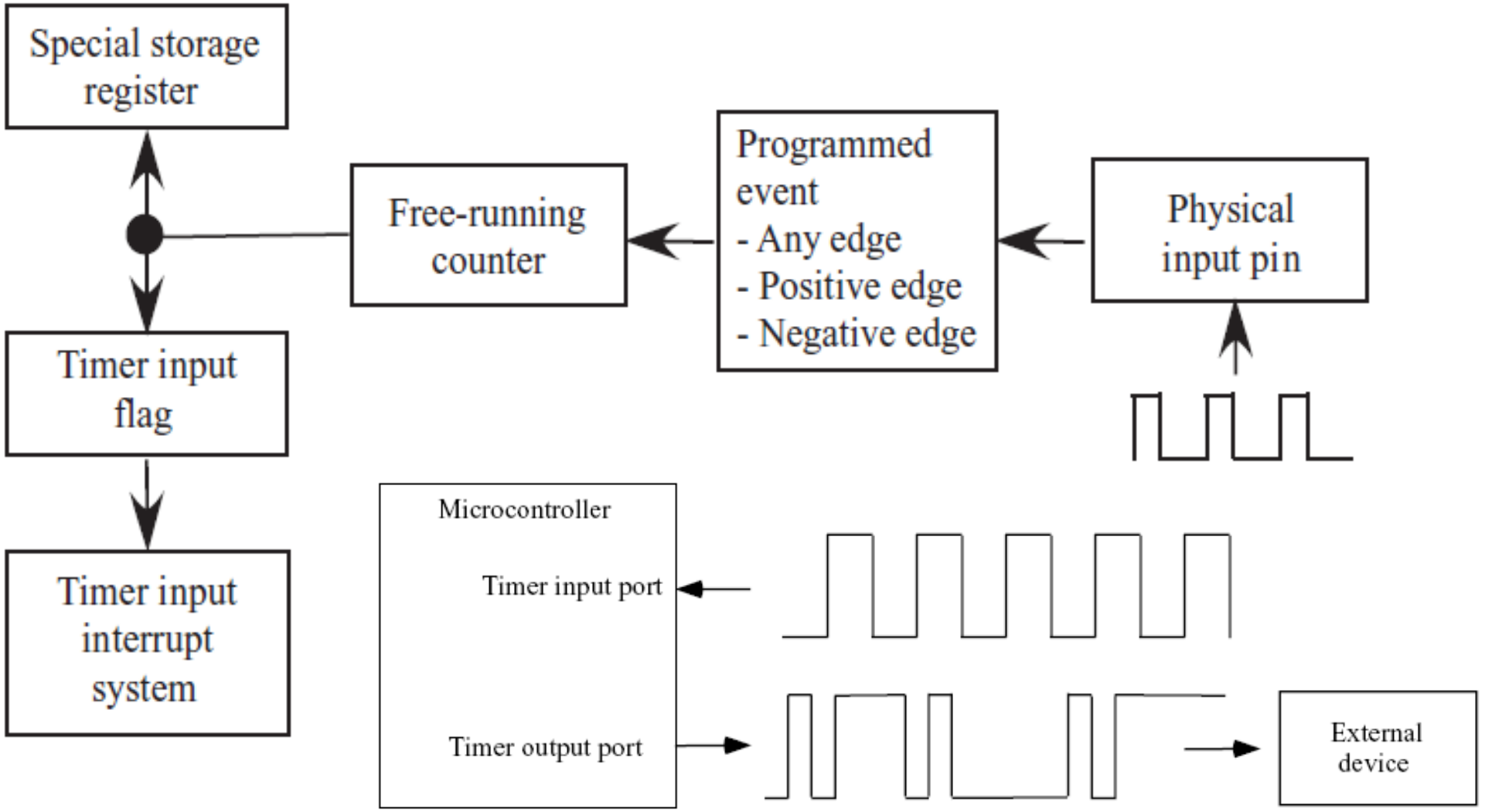
• Frequency: Signal frequency is the number of cycles per second completed by a repetitive signal. It is expressed in units of Hertz (Hz).

• Period: The period is the time increment in seconds required for a repetitive signal to complete a single cycle. The period is the reciprocal of the frequency ($T = 1/ f$).

• Duty cycle: The duty cycle indicates the percentage of time for which the signal is active in a single period.

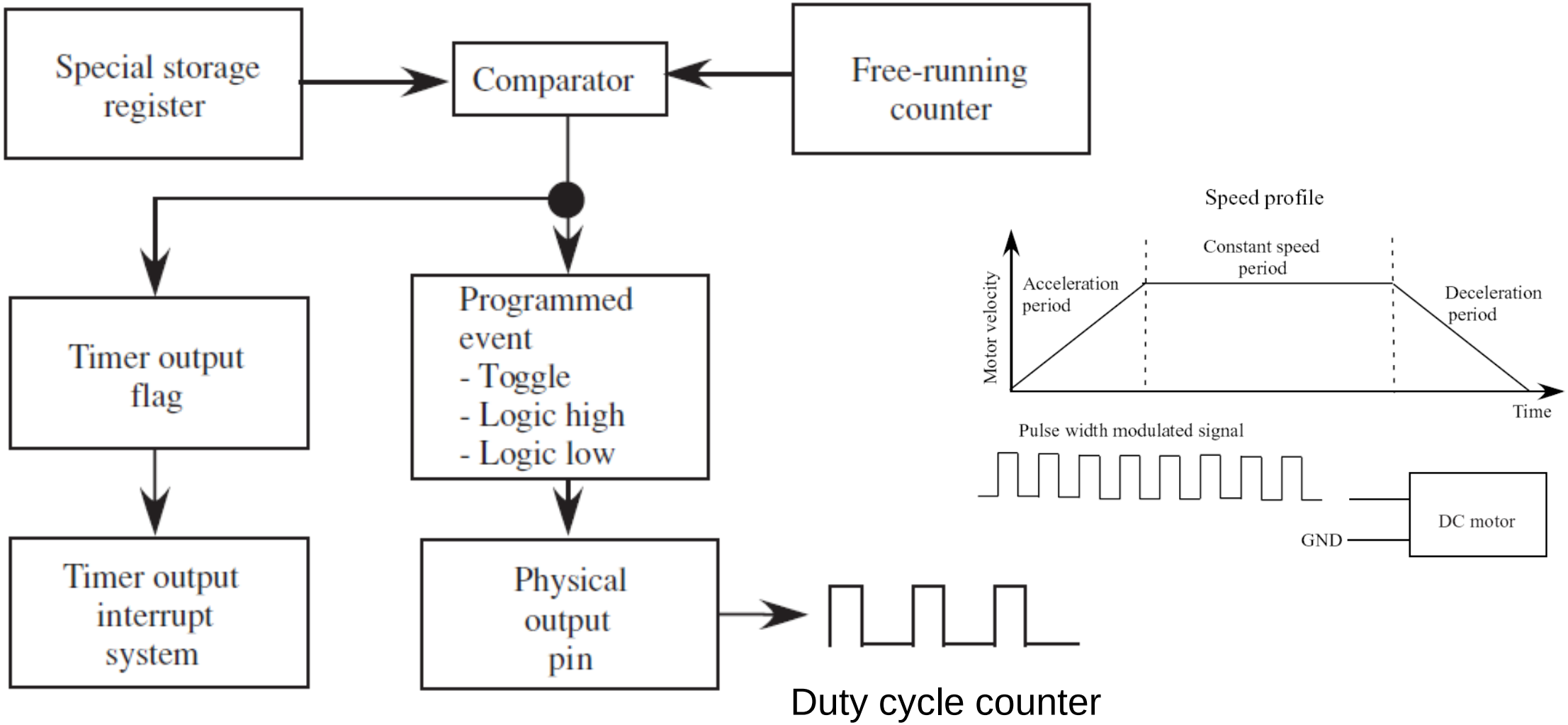
• Pulse width modulation (PWM):PWM signals are frequently used to control motor speed.

Timer/Counter Input System

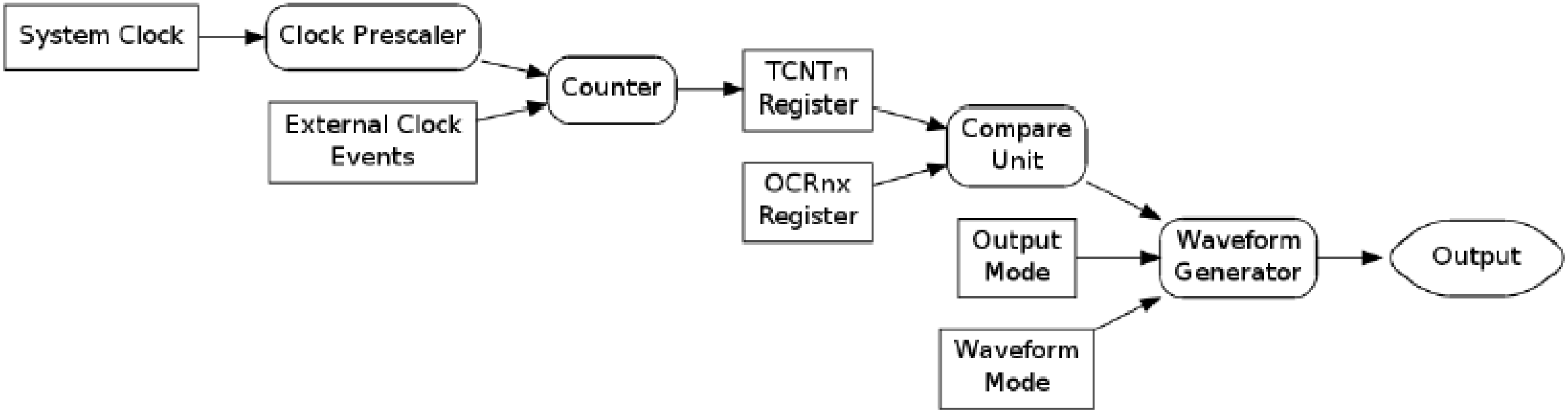


Timer input/output

Timer/Counter Output System



How does the Timer/Counter work?

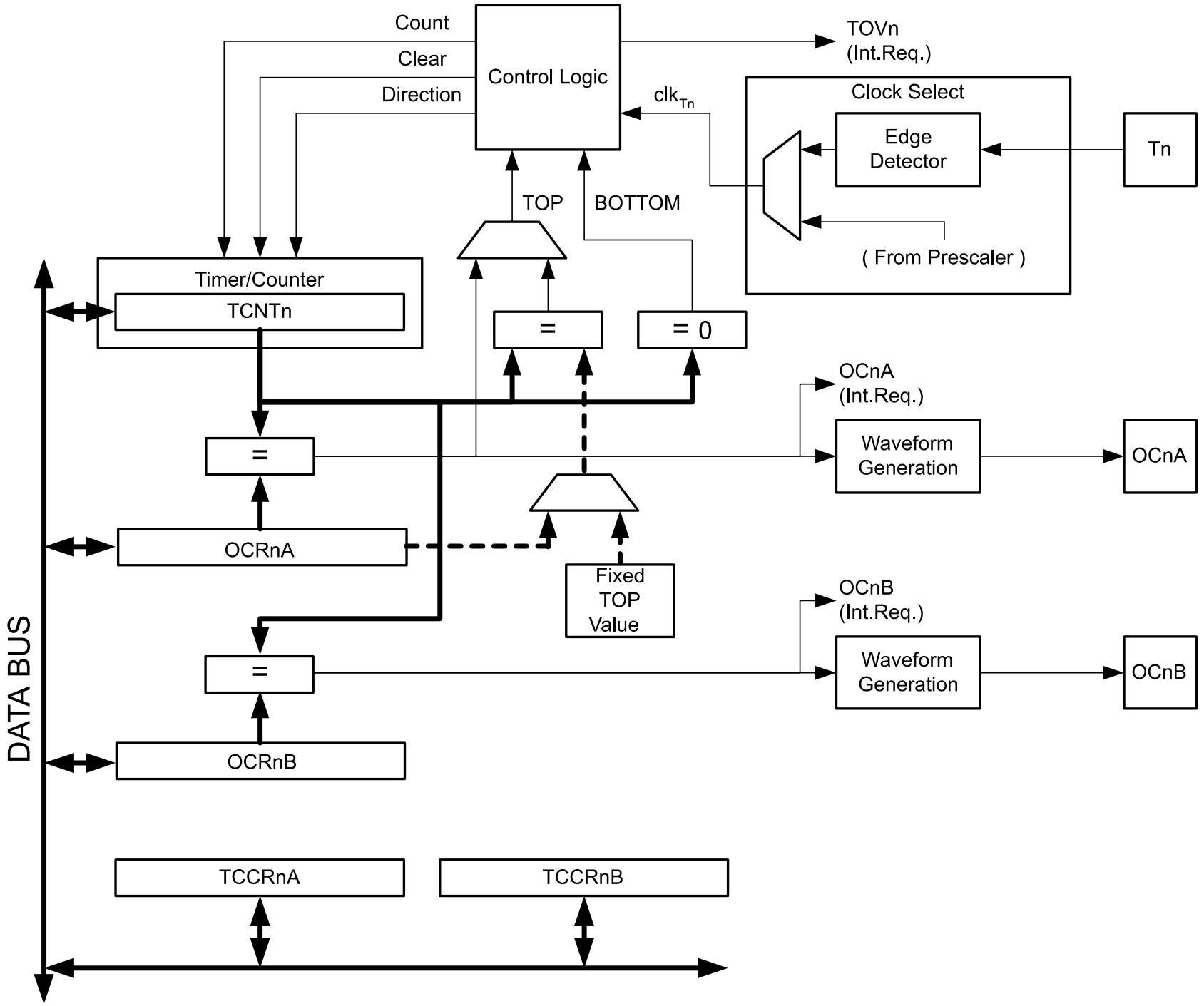


Timers

- ATMega328PB has 5 timers
  - 2 8-bit timers: Timer0, Timer2
  - 3 16-bit timers: Timer1, Timer3, Timer4

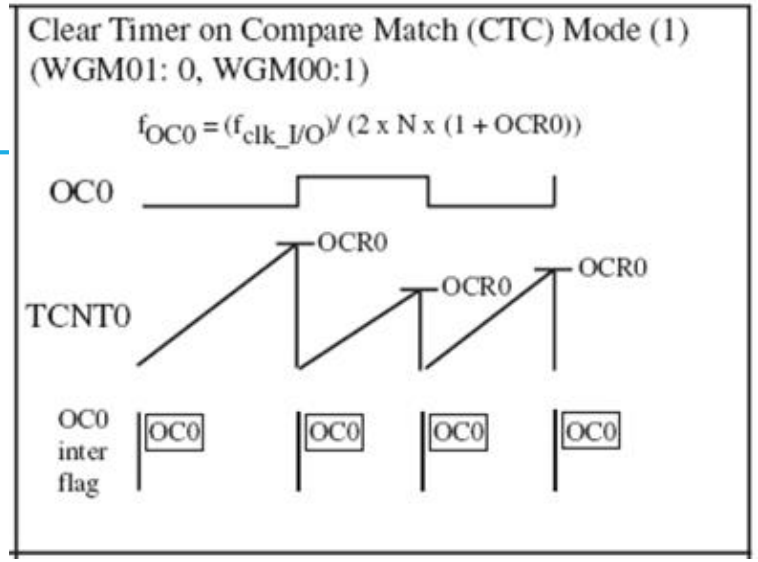
Timer 0	Timer 2	Timer1, 3, 4
8bit Timer/Counter	8bit Timer/Counter	16bit Timer/Counter
8bit PWM	8bit PWM	16bit PWM
Interrupt Source	Interrupt Source	Interrupt Source
	Asynchronous Operation	External Event Counter
		One Input Capture

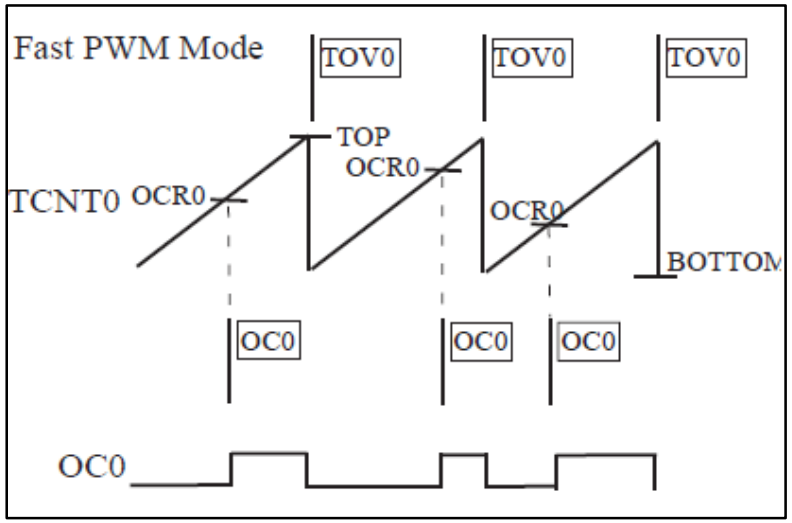
Timer 0

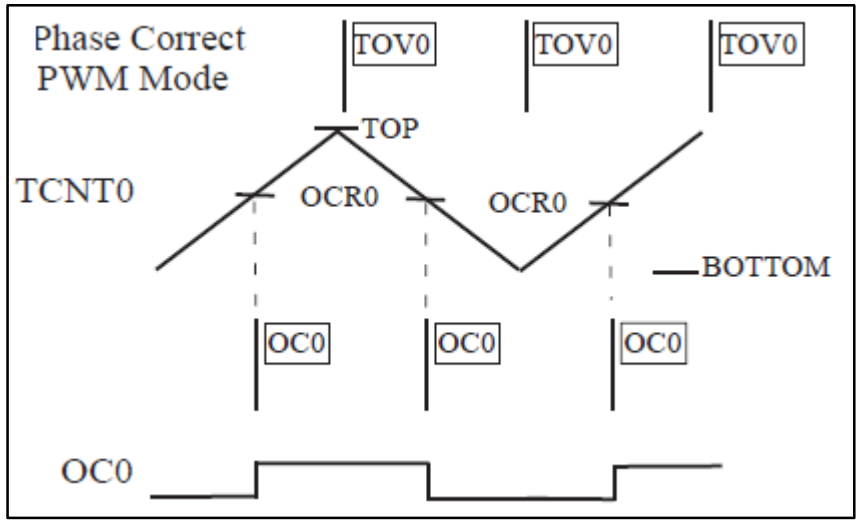

Count
Clear
Direction
Control Logic
TOVn
(Int.Req.)
clk$_{Tn}$
Clock Select
Edge
Detector
Tn
(From Prescaler)
TOP
BOTTOM
Timer/Counter
TCNTn
=
= 0
OCnA
(Int.Req.)
Waveform
Generation
OCnA
OCRnA
Fixed
TOP
Value
OCnB
(Int.Req.)
Waveform
Generation
OCnB
OCRnB
DATA BUS
TCCRnA
TCCRnB

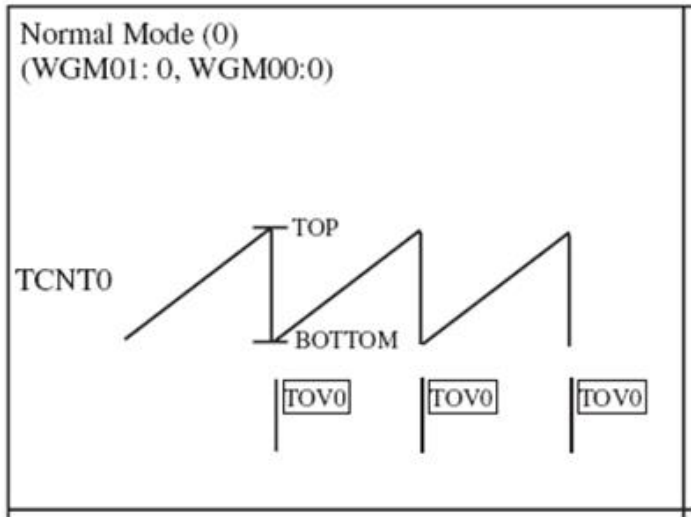
Timer 0

- Normal mode
  - Timer counts upward and continues counting when it hits 0xFF
  - To generate a periodic “clock tick” that may be used to calculate elapsed real time or provide delays within a system.
- CTC (clear timer on compare match) mode
  - Timer resets to 0 when counter reaches match
  - Used to generate a precision digital waveform such as a periodic signal or a single pulse
- PWM modes
  - Timer counts upward and resets to 0 when counter reaches match until 0xFF.









TCCR0A

TC0 Control Register A

Name: TCCR0A

Bit	7	6	5	4	3	2	1	0
	COM0A[1:0]		COM0B [1:0]				WGM0[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

COM0A[1]	COM0A[0]	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.

Mode	WGM0[2]	WGM0[1]	WGM0[0]	Timer/Counter Mode of Operation	TOP	Update of OCR0x 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	OCR0A	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCR0A	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCR0A	BOTTOM	TOP

TCCR0B

TC0 Control Register B

Name: TCCR0B

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM0 [2]	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

CS0[2]	CS0[1]	CS0[0]	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	clkI/O/256 (from prescaler)
1	0	1	$clk_{I/O}$/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.

TIMSK0

TC0 Interrupt Mask Register

Name: TIMSK0

Bit	7	6	5	4	3	2	1	0
						OCIE0B	OCIE0A	TOIE0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCIE0B Timer/Counter0, Output Compare B Match Interrupt Enable
When the OCIE0B bit is written to one, and the I-bit in the Status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in TIFR0.

Bit 1 – OCIE0A Timer/Counter0, Output Compare A Match Interrupt Enable
When the OCIE0A bit is written to one, and the I-bit in the Status register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in TIFR0.

Bit 0 – TOIE0 Timer/Counter0, Overflow Interrupt Enable
When the TOIE0 bit is written to one, and the I-bit in the Status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in TIFR0.

TIFR0

TC0 Interrupt Flag Register

Name: TIFR0

Bit	7	6	5	4	3	2	1	0
						OCF0B	OCF0A	TOV0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCF0B Timer/Counter 0, Output Compare B Match Flag
The OCF0B bit is set when a compare match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

Bit 1 – OCF0A Timer/Counter 0, Output Compare A Match Flag
The OCF0A bit is set when a compare match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

Bit 0 – TOV0 Timer/Counter 0, Overflow Flag
The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter 0 Overflow interrupt is executed.

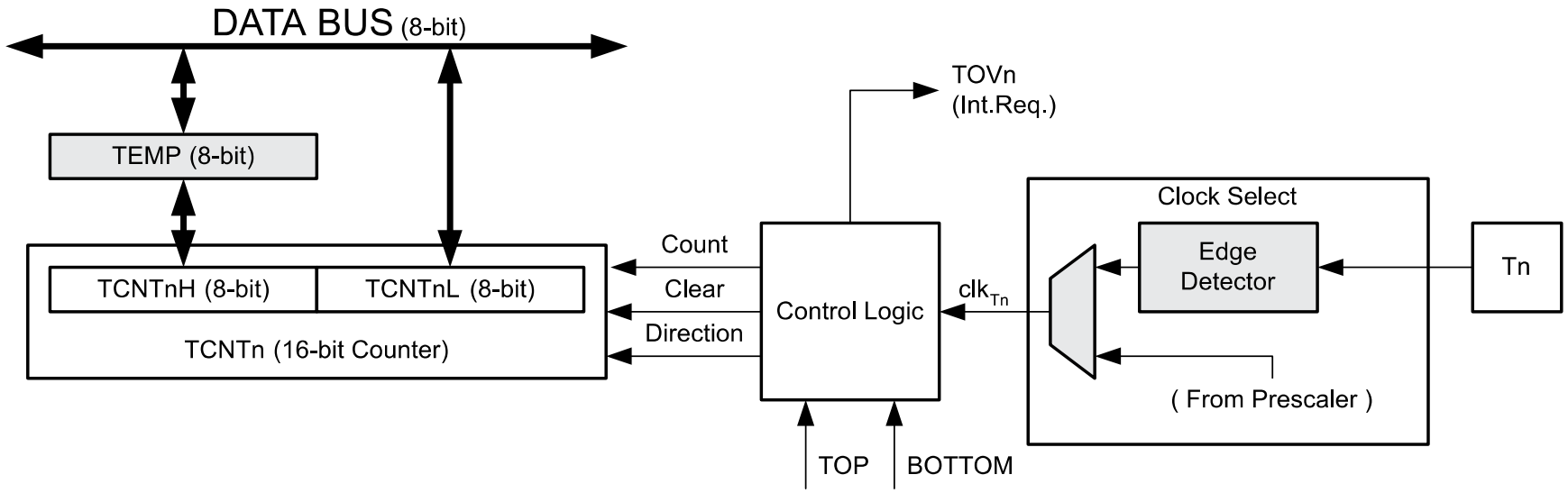
The setting of this flag is dependent on the WGM0[2:0] bit setting. Refer to bit description of WGM0 in TCCR0A.

Example Timer 0

- 16MHz, 1ms ticks:

```
// 1ms ISR for Timer 0 assuming F_CPU = 16MHz
void InitTimer0(void)
{
   TCCR0A |= (1<<WGM01); // Turn on clear-on-match with OCR0A
   OCR0A = 249;          // Set the compare register to 250 ticks
   TIMSK0 = (1<<OCIE0A); // Enable Timer 0 Compare A ISR
   TCCR0B = 3;           // Set Prescalar to divide by 64 & Timer 0 starts
}
….
```

Timer 1, 3, 4


DATA BUS (8-bit)
TEMP (8-bit)
TCNTnH (8-bit)
TCNTnL (8-bit)
TCNTn (16-bit Counter)
Count
Clear
Direction
Control Logic
TOVn
(Int.Req.)
clk$_{Tn}$
Clock Select
Edge
Detector
Tn
(From Prescaler)
TOP
BOTTOM

Register Description Timer 1

TCCR1A

Bit	7	6	5	4	3	2	1	0
	COM1A[1:0]		COM1B[1:0]				WGM1[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

WGM=0000 Normal mode

WGM=0100 CTC mode

TCCR1B

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM1[3]	WGM1[2]	CS1[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

TCCR1C

Bit	7	6	5	4	3	2	1	0
	FOC1A	FOC1B						
Access	R/W	R/W						
Reset	0	0						

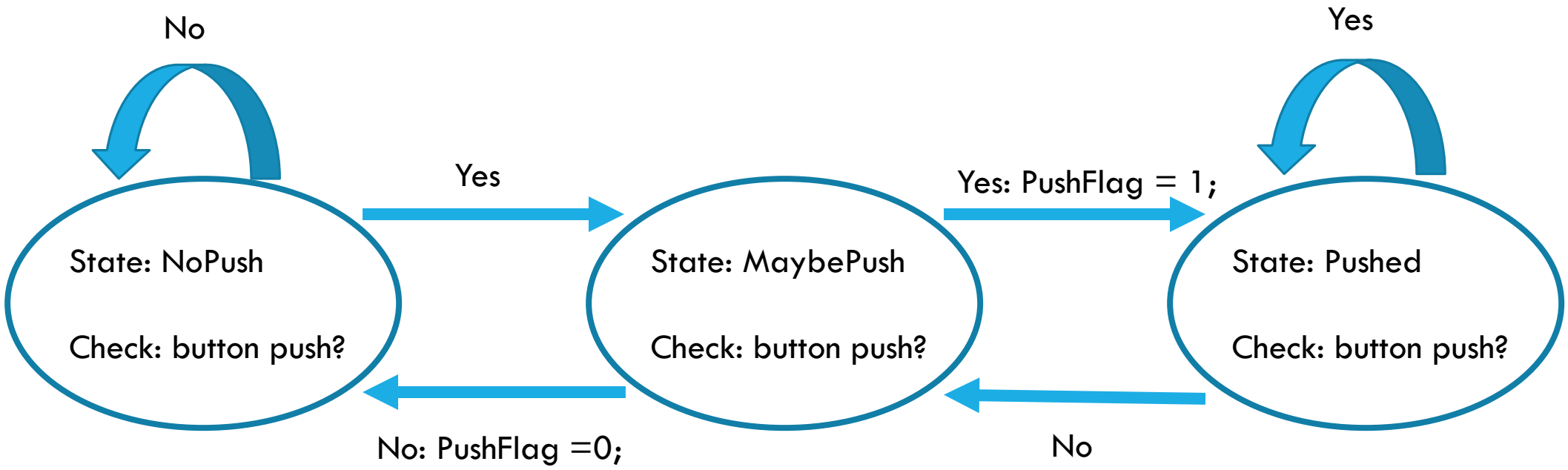
Timer interrupts

```
….
int EVENT_TIME = 500;
volatile int timerCount = EVENT_TIME;

ISR(TIMER0_COMPA_vect)
{
   if (timerCount >0) {timerCount--;}
}

// 1 ms ISR for Timer 0 assuming F_CPU = 16MHz
void InitTimer0(void)
{
   TCCR0A |= (1<<WGM01); // Clear on Compare A
   OCR0A = 249; // Set number of ticks for Compare A
   TIMSK0 = (1<<OCIE0A); ;  // Enable Timer 0 Compare A ISR
   TCCR0B = 3;  // Set Prescalar & Timer 0 starts
}
….
```

Timer-Driven State Machine

- State machines are a common programming paradigm that make it easier to deal with complex input behaviors
- If you always know the state of the program, you have less undefined behavior
- State transitions are usually defined by events (I/O, interrupts, etc.)
- Can be time driven as well
- Example: debouncing state machine

Debounce State Machine



Debounce State Machine

```
char push_debounce;

char pushState; //state variable
#define NoPush 1
#define Maybe 2
#define Pushed 3

void buttonSM(void)
{
  switch (pushState)
  {
     case NoPush:
        if (~PINB & 0x08)
            pushState=Maybe;
        else
            pushState=NoPush;
        break;
```

```
     case Maybe:
        if (~PINB & 0x08)
        {
            pushState=Pushed;
            push_debounce=1;
         }
         else
        {
            pushState=NoPush;
            push_debounce=0;
         }
         break;
     case Pushed:
        if (~PINB & 0x08)
            pushState=Pushed;
        else
            pushState=Maybe;
        break;
    }
}
```

Timer interrupts

```
#define DEBOUNCE_TIME 10
volatile int timerCount = DEBOUNCE_TIME;

ISR(TIMER0_COMPA_vect)
{
   if (timerCount >0) {
      timerCount--;
   } else {
      buttonSM();
      timerCount=DEBOUNCE_TIME;
   }
}

// 1 ms ISR for Timer 0 assuming F_CPU = 16MHz
void InitTimer0(void)
{
   TCCR0A |= (1<<WGM01); // Clear on Compare A
   OCR0A = 249; // Set number of ticks for Compare A
   TIMSK0 = (1<<OCIE0A); ;  // Enable Timer 0 Compare A ISR
   TCCR0B = 3;  // Set Prescalar & Timer 0 starts
}
….
```

Debounce State Machine

```
char pushFlag_Debounce;

char pushState; //state variable
#define NoPush 1
#define Maybe 2
#define Pushed 3

void buttonSM(void)
{
  switch (pushState)
  {
     case NoPush:
        if (~PINB & 0x08)
            pushState=Maybe;
        else
            pushState=NoPush;
        break;
```

```
     case Maybe:
        if (~PINB & 0x08)
        {
            pushState=Pushed;
            pushFlag_Debounce=1;
         }
         else
        {
            pushState=NoPush;
            pushFlag_Debounce=0;
         }
         break;
     case Pushed:
        if (~PINB & 0x08)
            pushState=Pushed;
        else
            pushState=Maybe;
        break;
    }
}
```

Timer interrupts

```
#define DEBOUNCE_TIME 10
ISR(TIMER0_COMPA_vect)
{
   if (debounceFlag == 1) {
      timerCount--;
      if (timerCount == 0) {
         push_debounce = 1;
         debounceFlag = 0;
      }
   }
}

ISR(INT1_vect)
{
   timerCount = DEBOUNCE_TIME;
   debounceFlag = 1;
}

void InitINT1(void)
{
   DDRD &= ~(1<<DDR3); // INT1 is an input
   EICRA |= (1<<ISC11);// falling edge of INT1
   EIMSK |= (1<<INT1); // enable INT1 interrupt
}
```