Lecture 6 – ATmega328 Timers and Interrupts

CSE P567
Arduino Digital and Analog I/O Pins

- **Digital pins:**
  - Pins 0 – 7: PORT D [0:7]
  - Pins 8 – 13: PORT B [0:5]
  - Pins 14 – 19: PORT C [0:5] (Arduino analog pins 0 – 5)
  - digital pins 0 and 1 are RX and TX for serial communication
  - digital pin 13 connected to the base board LED

- **Digital Pin I/O Functions**
  - `pinMode(pin, mode)`
    - Sets pin to INPUT or OUTPUT mode
    - Writes 1 bit in the DDRx register
  - `digitalWrite(pin, value)`
    - Sets pin value to LOW or HIGH (0 or 1)
    - Writes 1 bit in the PORTx register
  - `int value = digitalRead(pin)`
    - Reads back pin value (0 or 1)
    - Read 1 bit in the PINx register
Port Pin Definitions

```c
#define PINB _SFR_IO8(0x03)
#define PINB0 0
...
#define PINB7 7

#define DDRB _SFR_IO8(0x04)
#define DDB0 0
...
#define DDB7 7

#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
...
#define PORTB7 7

#define PINC _SFR_IO8(0x06)
#define PINC0 0
...
#define PINC6 6

#define DDRC _SFR_IO8(0x07)
#define DDC0 0
...
#define DDC6 6

#define PORTC _SFR_IO8(0x08)
#define PORTC0 0
...
#define PORTC6 6

#define PIND _SFR_IO8(0x09)
#define PIND0 0
...
#define PIND7 7

#define DDRD _SFR_IO8(0x0A)
#define DDD0 0
...
#define DDD7 7

#define PORTD _SFR_IO8(0x0B)
#define PORTD0 0
...
#define PORTD7 7
```
Interrupts

- Allow program to respond to events when they occur
- Allow program to ignore events until they occur
- External events e.g.:
  - UART ready with/for next character
  - Signal change on pin
    - Action depends on context
  - # of edges arrived on pin
- Internal events e.g.:
  - Power failure
  - Arithmetic exception
  - Timer “tick”
# ATmega328 Interrupts

<table>
<thead>
<tr>
<th>VectorNo.</th>
<th>Program Address(^{(2)})</th>
<th>Source</th>
<th>Interrupt Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000(^{(1)})</td>
<td>RESET</td>
<td>External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset</td>
</tr>
<tr>
<td>2</td>
<td>0x0002</td>
<td>INTO</td>
<td>External Interrupt Request 0</td>
</tr>
<tr>
<td>3</td>
<td>0x0004</td>
<td>INT1</td>
<td>External Interrupt Request 1</td>
</tr>
<tr>
<td>4</td>
<td>0x0006</td>
<td>PCINT0</td>
<td>Pin Change Interrupt Request 0</td>
</tr>
<tr>
<td>5</td>
<td>0x0008</td>
<td>PCINT1</td>
<td>Pin Change Interrupt Request 1</td>
</tr>
<tr>
<td>6</td>
<td>0x00A</td>
<td>PCINT2</td>
<td>Pin Change Interrupt Request 2</td>
</tr>
<tr>
<td>7</td>
<td>0x00C</td>
<td>WDT</td>
<td>Watchdog Time-out Interrupt</td>
</tr>
<tr>
<td>8</td>
<td>0x00E</td>
<td>TIMER2 COMPA</td>
<td>Timer/Counter2 Compare Match A</td>
</tr>
<tr>
<td>9</td>
<td>0x010</td>
<td>TIMER2 COMPB</td>
<td>Timer/Counter2 Compare Match B</td>
</tr>
<tr>
<td>10</td>
<td>0x012</td>
<td>TIMER2 OVF</td>
<td>Timer/Counter2 Overflow</td>
</tr>
<tr>
<td>11</td>
<td>0x014</td>
<td>TIMER1 CAPT</td>
<td>Timer/Counter1 Capture Event</td>
</tr>
<tr>
<td>12</td>
<td>0x016</td>
<td>TIMER1 COMPA</td>
<td>Timer/Counter1 Compare Match A</td>
</tr>
<tr>
<td>13</td>
<td>0x018</td>
<td>TIMER1 COMPB</td>
<td>Timer/Counter1 Compare Match B</td>
</tr>
<tr>
<td>14</td>
<td>0x01A</td>
<td>TIMER1 OVF</td>
<td>Timer/Counter1 Overflow</td>
</tr>
<tr>
<td>15</td>
<td>0x01C</td>
<td>TIMER0 COMPA</td>
<td>Timer/Counter0 Compare Match A</td>
</tr>
<tr>
<td>16</td>
<td>0x01E</td>
<td>TIMER0 COMPB</td>
<td>Timer/Counter0 Compare Match B</td>
</tr>
</tbody>
</table>
## ATmega328 Interrupts (cont)

<table>
<thead>
<tr>
<th>VectorNo.</th>
<th>Program Address</th>
<th>Source</th>
<th>Interrupt Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0x0020</td>
<td>TIMER0 OVF</td>
<td>Timer/Counter0 Overflow</td>
</tr>
<tr>
<td>18</td>
<td>0x0022</td>
<td>SPI, STC</td>
<td>SPI Serial Transfer Complete</td>
</tr>
<tr>
<td>19</td>
<td>0x0024</td>
<td>USART, RX</td>
<td>USART Rx Complete</td>
</tr>
<tr>
<td>20</td>
<td>0x0026</td>
<td>USART, UDRE</td>
<td>USART, Data Register Empty</td>
</tr>
<tr>
<td>21</td>
<td>0x0028</td>
<td>USART, TX</td>
<td>USART, Tx Complete</td>
</tr>
<tr>
<td>22</td>
<td>0x002A</td>
<td>ADC</td>
<td>ADC Conversion Complete</td>
</tr>
<tr>
<td>23</td>
<td>0x002C</td>
<td>EE READY</td>
<td>EEPROM Ready</td>
</tr>
<tr>
<td>24</td>
<td>0x002E</td>
<td>ANALOG COMP</td>
<td>Analog Comparator</td>
</tr>
<tr>
<td>25</td>
<td>0x0030</td>
<td>TWI</td>
<td>2-wire Serial Interface</td>
</tr>
<tr>
<td>26</td>
<td>0x0032</td>
<td>SPM READY</td>
<td>Store Program Memory Ready</td>
</tr>
</tbody>
</table>
Interrupt Model

- When an interrupt event occurs:
  - Processor does an automatic procedure call
  - CALL automatically done to address for that interrupt
    - Push current PC, Jump to interrupt address
  - Each event has its own interrupt address
  - The global interrupt enable bit (in SREG) is automatically cleared
    - i.e. nested interrupts are disabled
    - SREG bit can be set to enable nested interrupts if desired

- Interrupt procedure, aka “interrupt handler”
  - Does whatever it needs to, then returns via RETI
  - The global interrupt enable bit is automatically set on RETI
  - One program instruction is always executed after RETI
Interrupts

- **Type 1** – Event is remembered when interrupt is disabled
  - If interrupt is not enabled, flag is set
    - When interrupt is enabled again, interrupt takes place, and flag is reset

- **Type 2** – Event is not remembered when interrupt is disabled
  - Signal level causes interrupt
  - If level occurs when interrupt is enabled, interrupt takes place
  - If interrupt is not enabled, and level goes away before the interrupt is enabled, nothing happens
Interrupt Model

- Interrupt hander is invisible to program
  - Except through side-effects, e.g. via flags or variables
  - Changes program timing
    - Can’t rely on “dead-reckoning” using instruction timing
- Must be written so they are invisible
  - Cannot stomp on program state, e.g. registers
  - Save and restore any registers used
    - Including SREG
Interrupt Vectors

- Table in memory containing the first instruction of each interrupt handler
- Typically at program address 0

<table>
<thead>
<tr>
<th>Address</th>
<th>Labels</th>
<th>Code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>jmp</td>
<td>RESET</td>
<td>; Reset Handler</td>
</tr>
<tr>
<td>0x0002</td>
<td>jmp</td>
<td>EXT_INT0</td>
<td>; IRQ0 Handler</td>
</tr>
<tr>
<td>0x0004</td>
<td>jmp</td>
<td>EXT_INT1</td>
<td>; IRQ1 Handler</td>
</tr>
<tr>
<td>0x0006</td>
<td>jmp</td>
<td>PCINT0</td>
<td>; PCINT0 Handler</td>
</tr>
<tr>
<td>0x0008</td>
<td>jmp</td>
<td>PCINT1</td>
<td>; PCINT1 Handler</td>
</tr>
<tr>
<td>0x000A</td>
<td>jmp</td>
<td>PCINT2</td>
<td>; PCINT2 Handler</td>
</tr>
<tr>
<td>0x000C</td>
<td>jmp</td>
<td>WDT</td>
<td>; Watchdog Timer Handler</td>
</tr>
<tr>
<td>0x000E</td>
<td>jmp</td>
<td>TIM2_COMPA</td>
<td>; Timer2 Compare A Handler</td>
</tr>
<tr>
<td>0x0010</td>
<td>jmp</td>
<td>TIM2_COMPB</td>
<td>; Timer2 Compare B Handler</td>
</tr>
<tr>
<td>0x0012</td>
<td>jmp</td>
<td>TIM2_OVF</td>
<td>; Timer2 Overflow Handler</td>
</tr>
<tr>
<td>0x0014</td>
<td>jmp</td>
<td>TIM1_CAPT</td>
<td>; Timer1 Capture Handler</td>
</tr>
<tr>
<td>0x0016</td>
<td>jmp</td>
<td>TIM1_COMPA</td>
<td>; Timer1 Compare A Handler</td>
</tr>
<tr>
<td>0x0018</td>
<td>jmp</td>
<td>TIM1_COMPB</td>
<td>; Timer1 Compare B Handler</td>
</tr>
<tr>
<td>0x001A</td>
<td>jmp</td>
<td>TIM1_OVF</td>
<td>; Timer1 Overflow Handler</td>
</tr>
<tr>
<td>0x001C</td>
<td>jmp</td>
<td>TIM0_COMPA</td>
<td>; Timer0 Compare A Handler</td>
</tr>
<tr>
<td>0x001E</td>
<td>jmp</td>
<td>TIM0_COMPB</td>
<td>; Timer0 Compare B Handler</td>
</tr>
</tbody>
</table>
Interrupt Vectors

- If interrupts are not used, this memory can be used as part of the program
  - i.e. nothing special about this part of memory

Example interrupt routine

- RESET: Sets up the stack pointer

```
0x0033:      ldi    r16, high(RAMEND); Main program start
0x0034:      out    SPH,r16           ; Set Stack Pointer to top of RAM
0x0035:      ldi    r16, low(RAMEND)
0x0036:      out    SPL,r16
0x0037:      sei                ; Enable interrupts
0x0038:      <instr>   xxx
```
Defined ISR’s

```c
#define INT0_vect VECTOR(1)    /* External Interrupt Request 0 */
#define INT1_vect VECTOR(2)    /* External Interrupt Request 1 */
#define PCINT0_vect VECTOR(3)  /* Pin Change Interrupt Request 0 */
#define PCINT1_vect VECTOR(4)  /* Pin Change Interrupt Request 0 */
#define PCINT2_vect VECTOR(5)  /* Pin Change Interrupt Request 1 */
#define WDT_vect VECTOR(6)     /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect VECTOR(7)  /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect VECTOR(8)  /* Timer/Counter2 Compare Match B */
#define TIMER2_OVF_vect VECTOR(9)  /* Timer/Counter2 Overflow */
#define TIMER1_CAPTURE_vect VECTOR(10) /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect VECTOR(11) /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect VECTOR(12) /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect VECTOR(13) /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect VECTOR(14) /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect VECTOR(15) /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect VECTOR(16) /* TimerCounter0 Overflow */
#define SPI_STC_vect VECTOR(17)  /* SPI Serial Transfer Complete */
#define USART_RX_vect VECTOR(18) /* USART Rx Complete */
#define USART_UDRE_vect VECTOR(19) /* USART, Data Register Empty */
#define USART_TX_vect VECTOR(20) /* USART Tx Complete */
#define ADC_vect VECTOR(21)      /* ADC Conversion Complete */
#define EE_READY_vect VECTOR(22) /* EEPROM Ready */
#define ANALOG_COMP_vect VECTOR(23) /* Analog Comparator */
#define TWI_vect VECTOR(24)      /* Two-wire Serial Interface */
#define SPM_READY_vect VECTOR(25) /* Store Program Memory Read */
```
Interrupts

- Global interrupt enable
  - Bit in SREG
  - Allows all interrupts to be disabled with one bit
  - sei() – set the bit
  - cli() – clear the bit

- Interrupt priority is determined by order in table
  - Lower addresses have higher priority

- ISR(vector) – Interrupt routine definition

- reti() – return from interrupt
  - automatically generated for ISR
External Interrupts

- Monitors changes in signals on pins
- What causes an interrupt can be configured
  - by setting control registers appropriately

Pins:
- INT0 and INT1 – range of event options
  - INT0 – PORT D [2]
  - INT1 – PORT D [3]
- PCINT[23:0] – any signal change (toggle)
  - PCINT[7:0] – PORT B [7:0]
  - PCINT[14:8] – PORT C [6:0]
  - PCINT[23:16] – PORT D [7:0]
- Pulses on inputs must be slower than I/O clock rate
INT0 and INT1

- External Interrupt Control Register:

<table>
<thead>
<tr>
<th>Bit (0x69)</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EICRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ISC00</td>
<td>ISC01</td>
<td>ISC10</td>
</tr>
</tbody>
</table>

- Sense Control (INT0 is the same)

<table>
<thead>
<tr>
<th>ISC11</th>
<th>ISC10</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The low level of INT1 generates an interrupt request.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Any logical change on INT1 generates an interrupt request.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The falling edge of INT1 generates an interrupt request.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The rising edge of INT1 generates an interrupt request.</td>
</tr>
</tbody>
</table>

Table 12-1. Interrupt 1 Sense Control
INT0 and INT1

- External Interrupt Mask Register
  - If INT# bit is set (and the SREG I-bit is set), then interrupts are enabled on pin INT#

- External Interrupt Flag Register
  - Interrupt flag bit is set when a change triggers an interrupt request
  - Flag is cleared automatically when interrupt routine is executed
  - Flag can be cleared by writing a 1 to it
Arduino Language Support for External Interrupts

- `attachInterrupt(interrupt, function, mode)`
  - `interrupt`: 0 or 1
  - `function`: interrupt function to call
  - `mode`: LOW, CHANGE, RISING, FALLING

- `detachInterrupt(interrupt)`

- `interrupts() – Enable interrupts : sei()`

- `noInterrupts() – Disable interrupts : cli()`
### PCINT[23:0]

- **Pin Change Interrupt Control Register**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0x68)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PCIE2</td>
<td>PCIE1</td>
<td>PCIE0</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- PCIE2 enables interrupts for PCINT[23:16]
- PCIE1 enables interrupts for PCINT[14:8]
- PCIE0 enables interrupts for PCINT[7:0]

- **Pin Change Interrupt Flag Register**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1B (0x3B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PCIF2</td>
<td>PCIF1</td>
<td>PCIF0</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- PCIF# set if corresponding pins generate an interrupt request
- Cleared automatically when interrupt routine is executed
PCINT[23:0]

- Pin Change Mask Register 2

Each bit controls whether interrupts are enabled for the corresponding pin
- Change on any enabled pin causes an interrupt
- (Mask registers 1 and 0 are similar)
8-bit Timer/Counter 0
(1 and 2 are very similar)
Prescaler for Timer/Counter 0 & 1

Figure 16-2. Prescaler for Timer/Counter0 and Timer/Counter1 (1)
**Clock Source Select (CS0[2:0])**

<table>
<thead>
<tr>
<th>CS02</th>
<th>CS01</th>
<th>CS00</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No clock source (Timer/Counter stopped)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>clk_UO/(No prescaling)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>clk_UO/8 (From prescaler)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>clk_UO/64 (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>clk_UO/256 (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>clk_UO/1024 (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>External clock source on T0 pin. Clock on falling edge.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>External clock source on T0 pin. Clock on rising edge.</td>
</tr>
</tbody>
</table>

- **T0 pin** – PORT D[4]
- **T1 pin** – PORT D[5]
- Pin can be configured as output pin
  - Program can generate clock
External Clock Source

Figure 16-1. T1/T0 Pin Sampling
Timer/Counter Registers

- TCNT0 – Timer/Counter Register (8-bit)
- OCR0A – Output Compare Register A
- OCR0B – Output Compare Register B
- TCCR0A/B – Timer/Counter Control Registers
- TIMSK0 – Timer/Counter Interrupt Mask Register
  - TOV interrupt
  - Compare A&B interrupts
- TIFR0 – Timer/Counter Interrupt Flag Register
  - TOV interrupt
  - Compare A&B interrupts
Normal Mode (0)

- Timer increments
- Wraps around at TOP = 0xFF
- Starts again at 0
- TOV0 interrupt flag set when TCNT0 reset to 0

- Useful for generating interrupts every N time units
- Useful for generating an interrupt in N time units
  - Set TCNT0 to an initial value (255 – N)
CTC (Clear Timer on Compare) Mode (2)

- Timer increments
- Wraps around at OCR0A
  - OCR0A defines top value of counter
- Starts again at 0
- OCF0A interrupt flag set when TCNT0 reset to 0
- Pin OC0A can be made to toggle when counter resets
  - Generate output waveform
Fast PWM Mode (3/7)

- Timer increments
- Wraps around at 0xFF (3) or OCR0A (7)
- Start again at 0
- Pin OC0x generates waveform
  - Set (reset) when timer=0
  - Reset (set) when timer=OCR0x
Phase-Correct PWM Mode (1/5)

- Timer increments then decrements
- Increments from 0
- Up to 0xFF (1) or OCR0A (5)
- Than back down to 0
- Pin OC0x generates waveform
  - Set when timer=OCR0x while incrementing
  - Reset when timer=OCR0x while decrementing
Timer/Counter Control Registers

- Timer/Counter Control Register A

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24 (0x44)</td>
<td>COM0A1</td>
<td>COM0A0</td>
<td>COM0B1</td>
<td>COM0B0</td>
<td>-</td>
<td>-</td>
<td>WGM01</td>
<td>WGM00</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Timer/Counter Control Register B

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25 (0x45)</td>
<td>FOC0A</td>
<td>FOC0B</td>
<td>-</td>
<td>-</td>
<td>WGM02</td>
<td>CS02</td>
<td>CS01</td>
<td>CS00</td>
</tr>
<tr>
<td>Read/Write</td>
<td>W</td>
<td>W</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Mode Summary

### Table 14-8. Waveform Generation Mode Bit Description

<table>
<thead>
<tr>
<th>Mode</th>
<th>WGM02</th>
<th>WGM01</th>
<th>WGM00</th>
<th>Timer/Counter Mode of Operation</th>
<th>TOP</th>
<th>Update of OCRx at</th>
<th>TOV Flag Set on</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Normal</td>
<td>0xFF</td>
<td>Immediate</td>
<td>MAX</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>PWM, Phase Correct</td>
<td>0xFF</td>
<td>TOP</td>
<td>BOTTOM</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CTC</td>
<td>OCRA</td>
<td>Immediate</td>
<td>MAX</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Fast PWM</td>
<td>0xFF</td>
<td>BOTTOM</td>
<td>MAX</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Reserved</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>PWM, Phase Correct</td>
<td>OCRA</td>
<td>TOP</td>
<td>BOTTOM</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Reserved</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Fast PWM</td>
<td>OCRA</td>
<td>BOTTOM</td>
<td>TOP</td>
</tr>
</tbody>
</table>

**Notes:**
1. MAX = 0xFF
2. BOTTOM = 0x00
COM0A[1:0]  (COM0B[1:0] similar)

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

<table>
<thead>
<tr>
<th>COM0A1</th>
<th>COM0A0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC0A disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Toggle OC0A on Compare Match</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC0A on Compare Match</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC0A on Compare Match</td>
</tr>
</tbody>
</table>

Table 14-3. Compare Output Mode, Fast PWM Mode

<table>
<thead>
<tr>
<th>COM0A1</th>
<th>COM0A0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>WGM02 = 0: Normal Port Operation, OC0A Disconnected.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>WGM02 = 1: Toggle OC0A on Compare Match.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC0A on Compare Match, set OC0A at BOTTOM, (non-inverting mode).</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC0A on Compare Match, clear OC0A at BOTTOM, (inverting mode).</td>
</tr>
</tbody>
</table>

Table 14-4. Compare Output Mode, Phase Correct PWM Mode

<table>
<thead>
<tr>
<th>COM0A1</th>
<th>COM0A0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC0A disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>WGM02 = 0: Normal Port Operation, OC0A Disconnected.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>WGM02 = 1: Toggle OC0A on Compare Match.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.</td>
</tr>
</tbody>
</table>
Timer/Counter 0 Interrupts

- **Timer/Counter 0 Interrupt Mask**
  - Bit 7: -
  - Bit 6: -
  - Bit 5: -
  - Bit 4: -
  - Bit 3: -
  - Bit 2: OCIE0B
  - Bit 1: OCIE0A
  - Bit 0: TOIE0
  - Initial Value: 0, 0, 0, 0, 0, 0, 0, 0, 0
  
  - **TOIE0** – Timer Overflow interrupt
  - **OCIE0A/B** – Compare A/B interrupt

- **Timer/Counter 1 Interrupt Flags**
  - Bit 7: -
  - Bit 6: -
  - Bit 5: -
  - Bit 4: -
  - Bit 3: -
  - Bit 2: OCF0B
  - Bit 1: OCF0A
  - Bit 0: TOV0
  - Initial Value: 0, 0, 0, 0, 0, 0, 0, 0, 0
  
  - **TOV0** – Timer Overflow flag
  - **OCF0A/B** – Compare A/B interrupt flag
Timer/Counter 2 (16-bit)

- Similar to Timer/Counter 1
  - 16-bit counter vs. 8-bit counter
  - 16-bit registers
    - Uses shared temporary 8-bit register to enable 16-bit read/write
  - Input capture register
Input Capture Unit
Input Capture Unit

- Event on input causes:
  - Counter value (TCNT1) to be written to ICR1
    - Time-stamp
  - Interrupt flag ICF1 to be set
    - Causing an interrupt, if enabled
- Pin ICPI – Port B [0]
- Noise Canceller
  - Pulses less than 4 clock cycles long are filtered
- Useful for measuring frequency and duty cycle
  - PWM inputs
Timer/Counter 1

- ICR can also be used as the TOP value
  - Allows OCR1A to be used for PWM generation
### Timer/Counter 1 Modes

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

- Identical to Timer/Counter 1
- Except for clock sources

*Figure 17-12. Prescaler for Timer/Counter2*
## External and Pin Change Interrupts

```c
#define PCIFR _SFR_IO8(0x1B)
#define PCIF0 0
#define PCIF1 1
#define PCIF2 2

#define EIFR _SFR_IO8(0x1C)
#define INTF0 0
#define INTF1 1

#define EIMSK _SFR_IO8(0x1D)
#define INT0 0
#define INT1 1

#define PCICR _SFR_MEM8(0x68)
#define PCIE0 0
#define PCIE1 1
#define PCIE2 2

#define PCMSK0 _SFR_MEM8(0x6B)
#define PCINT0 0

#define PCMSK1 _SFR_MEM8(0x6C)
#define PCINT8 0

#define PCMSK2 _SFR_MEM8(0x6D)
#define PCINT14 6

#define EICRA _SFR_MEM8(0x69)
#define ISC00 0
#define ISC01 1
#define ISC10 2
#define ISC11 3
```
Timer/Counter 0 Registers

#define TCCR0A _SFR_IO8(0x24) #define OCR0A _SFR_IO8(0x27)
#define WGM00 0 #define OCROA_0 0
#define WGM01 1 ... #define COM0B0 4 #define OCROA_7 7
#define WGM02 3 #define COM0B1 5 #define FOC0B 6
#define COM0A0 6 #define COM0A1 7 #define FOC0A 7
#define TCNT0 _SFR_IO8(0x26) #define OCR0B _SFR_IO8(0x28)
#define TCNT0_0 0 #define OCR0B_0 0 ... #define TCNT0_7 7
Timer/Counter Interrupts

#define TIFR0 _SFR_IO8(0x15)
#define TOV0 0
#define OCF0A 1
#define OCF0B 2

#define TIFR1 _SFR_IO8(0x16)
#define TOV1 0
#define OCF1A 1
#define OCF1B 2
#define ICF1 5

#define TIFR2 _SFR_IO8(0x17)
#define TOV2 0
#define OCF2A 1
#define OCF2B 2

#define TIMSK0 _SFR_MEM8(0x6E)
#define TOIE0 0
#define OCIE0A 1
#define OCIE0B 2

#define TIMSK1 _SFR_MEM8(0x6F)
#define TOIE1 0
#define OCIE1A 1
#define OCIE1B 2
#define ICIE1 5

#define TIMSK2 _SFR_MEM8(0x70)
#define TOIE2 0
#define OCIE2A 1
#define OCIE2B 2
Timer/Counter 1

#define TCCR1A _SFR_MEM8(0x80)
#define WGM10 0
#define WGM11 1
#define COM1B0 4
#define COM1B1 5
#define COM1A0 6
#define COM1A1 7

#define TCCR1B _SFR_MEM8(0x81)
#define CS10 0
#define CS11 1
#define CS12 2
#define WGM12 3
#define WGM13 4
#define ICES1 6
#define ICNC1 7

#define TCNT1 _SFR_MEM16(0x84)
#define TCNT1L _SFR_MEM8(0x84)
#define ICNT1L0 0
#define TCNT1L7 7

#define TCNT1H _SFR_MEM8(0x85)
#define TCNT1H0 0
#define ICNT1H7 7

#define ICR1 _SFR_MEM16(0x86)
#define ICR1L _SFR_MEM8(0x86)
#define ICR1L0 0
#define ICR1L7 7

#define ICR1H _SFR_MEM8(0x87)
#define ICR1H0 0
#define ICR1H7 7
#define OCR1A _SFR_MEM16(0x88)

#define OCR1AL _SFR_MEM8(0x88)
#define OCR1AL0 0
...
#define OCR1AL7 7

#define OCR1AH _SFR_MEM8(0x89)
#define OCR1AH0 0
...
#define OCR1AH7 7

#define OCR1B _SFR_MEM16(0x8A)

#define OCR1BL _SFR_MEM8(0x8A)
#define OCR1BL0 0
...
#define OCR1BL7 7

#define OCR1BH _SFR_MEM8(0x8B)
#define OCR1BH0 0
...
#define OCR1BH7 7
Timer/Counter 2

#define TCCR2A _SFR_MEM8(0xB0)
#define OCR2A _SFR_MEM8(0xB3)
#define WGM20 0
#define OCR2_0 0
#define WGM21 1
... #define OCR2_7 7
#define COM2B0 4
#define COM2B1 5
#define WGM22 3
#define OCR2B _SFR_MEM8(0xB4)
#define COM2A0 6
#define COM2A1 7
#define OCR2AUB 3

#define TCCR2B _SFR_MEM8(0xB1)
#define ASSR _SFR_MEM8(0xB6)
#define CS20 0
#define TCR2BUB 0
#define CS21 1
#define TCR2AUB 1
#define CS22 2
#define OCR2BUB 2
#define WGM22 3
#define OCR2AUB 3
#define FOC2B 6
#define TCN2UB 4
#define FOC2A 7
#define AS2 5
#define TCNT2 _SFR_MEM8(0xB2)
#define EXCLK 6
... #define TCNT2_0 0
#define TCNT2_7 7
Timer Interrupt Program Example

```c
void setup() {
    DDRB = ; // Pin 13 as output
    // Using timer 2
    // Set to Normal mode, Pin OC0A disconnected
    TCCR2A = ;
    // Prescale clock by 1024
    // Interrupt every 256K/16M sec = 1/64 sec
    TCCR2B = ;
    // Turn on timer overflow interrupt flag
    TIMSK2 = ;
    // Turn on global interrupts
    sei();
}
char timer = 0;

ISR( _vect) {
    timer++;
    PORTB = ;
}

void loop()
{
    // Nothing to do
}
```
void setup() {
    DDRB = ; // Pin 13 OUTPUT
    // Using timer 2
    // Set to CTC mode, Pin OC0A disconnected
    TCCR2A = ;
    // Prescale clock by 1 (no prescale)
    TCCR2B = ;

    // Set compare register
    OCR2A = ;
    // Turn on timer compare A interrupt flag
    TIMSK2 = ;
    // Turn on global interrupts
    sei();
}
char timer = 0;

ISR(vect) {
    timer++;
    PORTB = ;
}
void loop()
{
    // Nothing to do
}
# External Interrupt Example

```cpp
#define pinint0
#define pinint1

void setup() {
    pinMode(pinint0,   );
    pinMode(pinint1,   );
    Serial.begin(9600);
    // External interrupts 0 and 1
    // Interrupt on rising edge
    EICRA =          ;
    // Enable both interrupts
    EIMSK =          ;
    // Turn on global interrupts
    sei();
}

ISR( _vect) {

}
ISR( _vect) {

}

// Print out the information
void loop() {
    Serial.print("X: ");
    Serial.print(percent0);
    Serial.print(" Y: ");
    Serial.println(percent1);
}
```