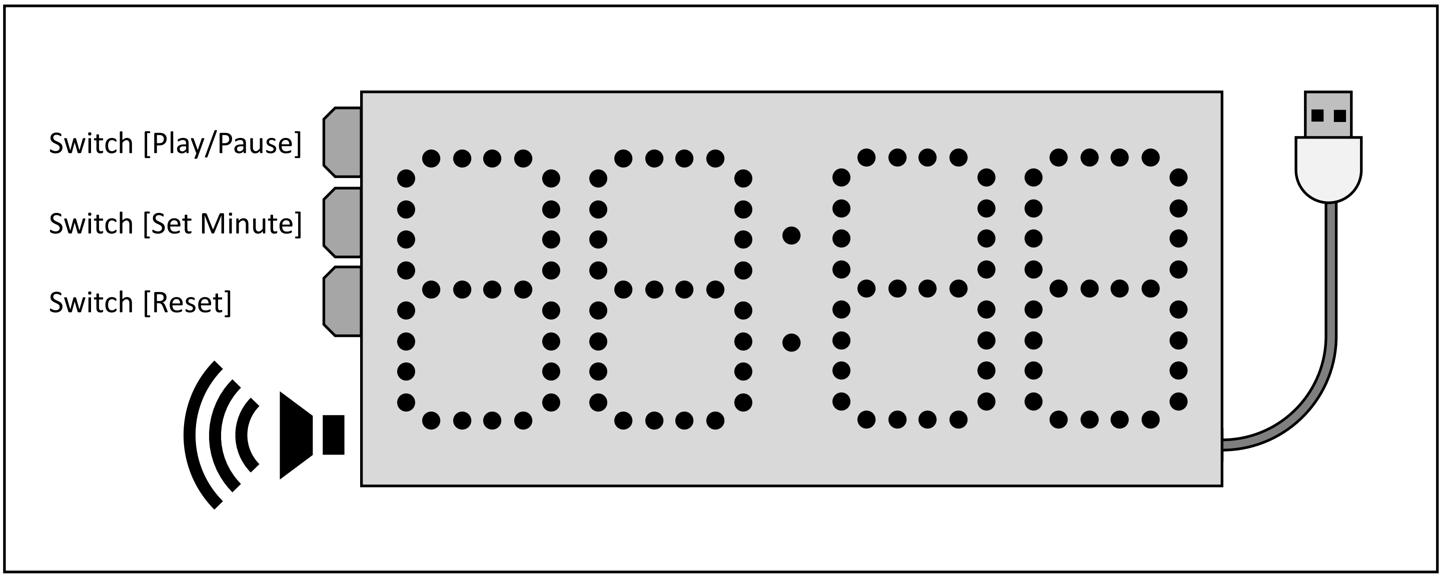
Project Countdown Timer Using Arduino

by *Sawyer Kao*, 29/08/2019

Part 1 – Introduction

● User Interface



● Power Supply

A standard USB type-A port that supply voltage with 5V and current more than 1A. Such as a common mobile power.

● Control

This device has 3 switches to control all functions.

1. Setup Mode

- [Reset]: Set time to 0.
- [Set Minute]: Short click to add 1 minute, Long click to add 10 minutes continuously.
- [Play/Pause]: Start countdown timer. (Switch to counting mode)

2. Counting Mode

- [Play/Pause]: Stop countdown timer. (Switch to setup mode)
- [Reset] and [Set Minute] are locked.

When countdown to “00:00”, speaker will sing a melody 3 times and show “End” on display then back to setup mode.

Part 2 – Hardware

Specification

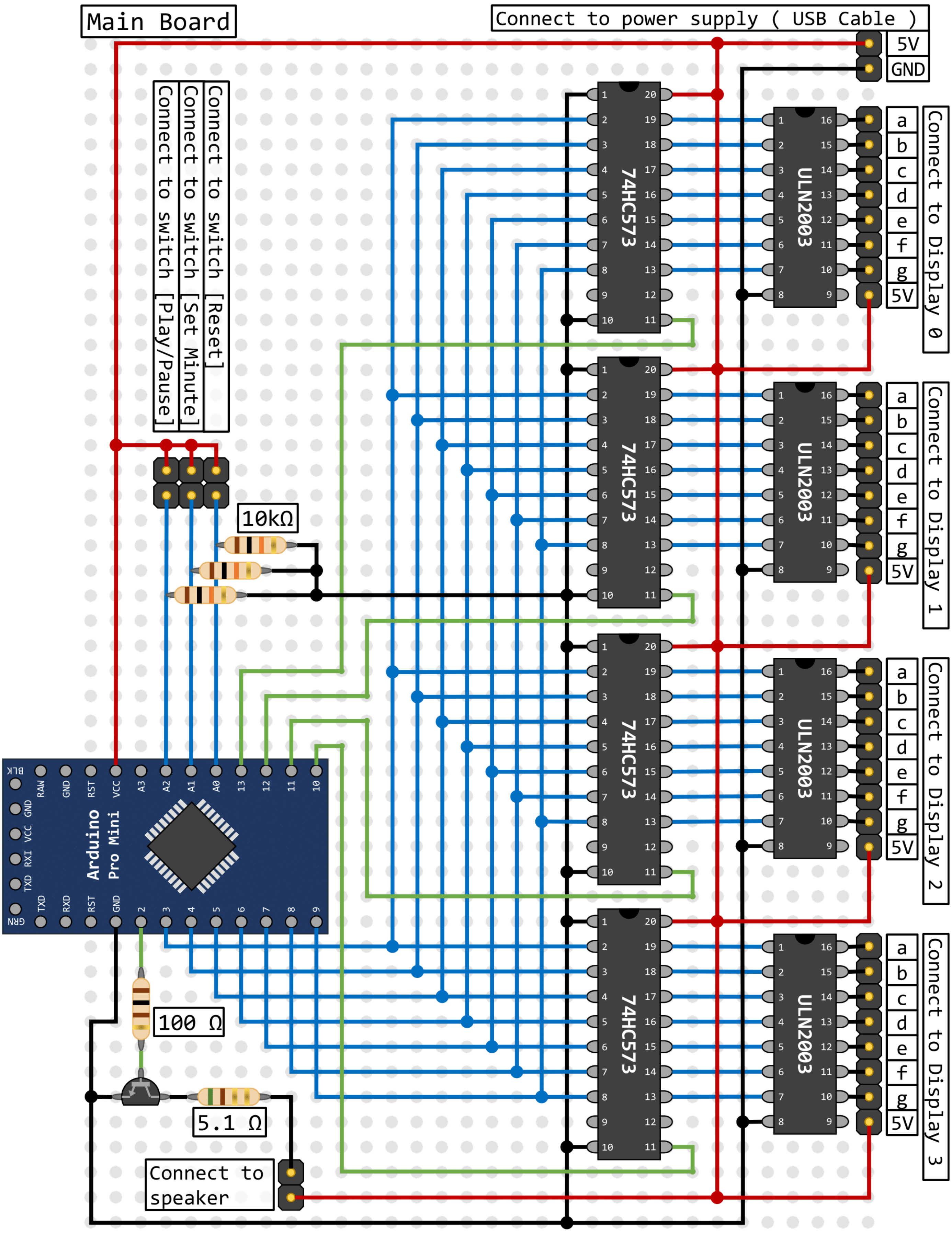
Suit for common mobile power.
- Working Voltage: 5V
- Maximum Current: 1A (All led turn on)

Components

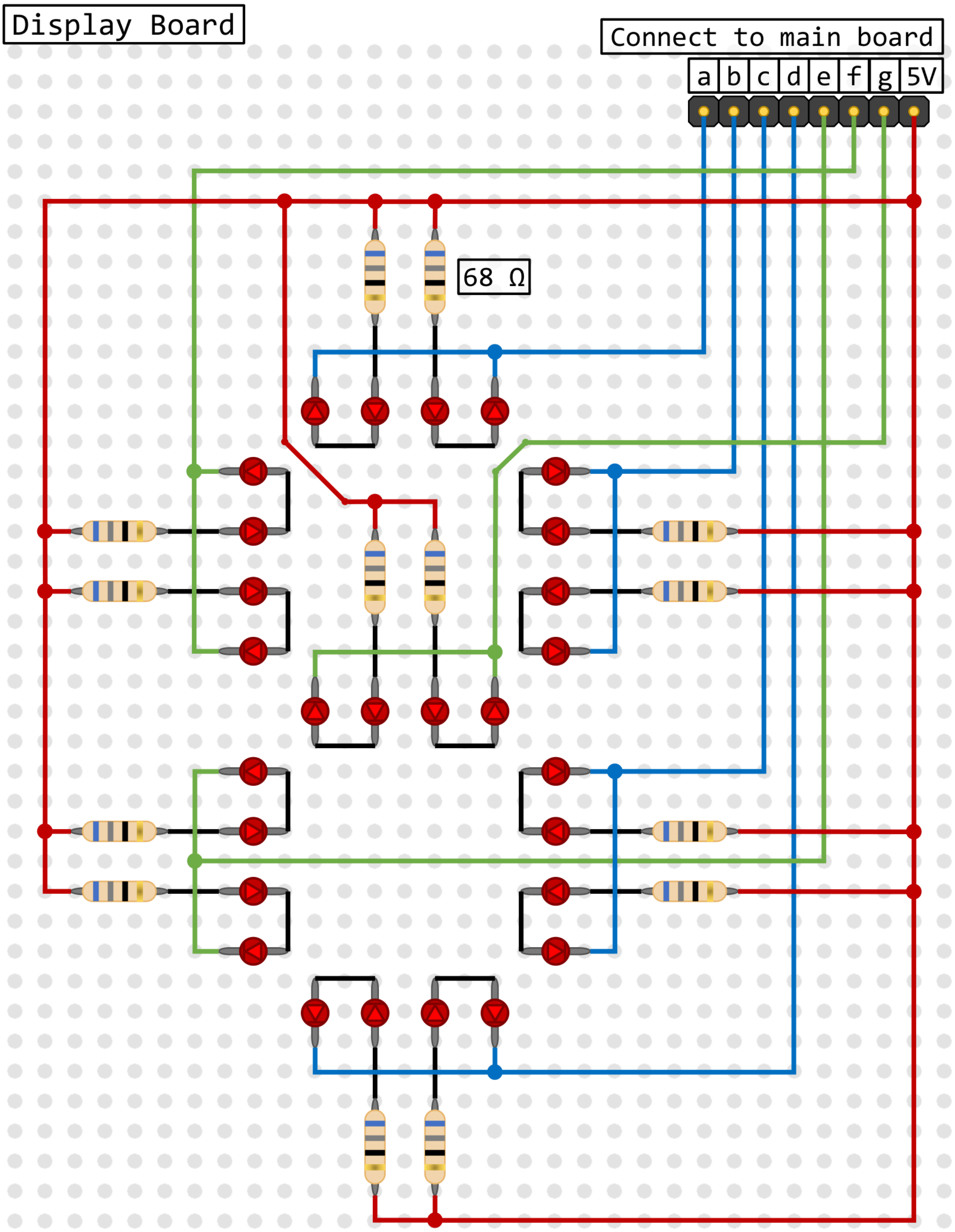
Breadboard	(2.54mm, 160mm*100mm) * 5 (pcs.)
USB Type-A Cable	1 (pc.)
5mm Red Led	2 (pcs.)
Resistor	68Ω * 1 (pc.)
DuPont Line	2pin * 1 (pc.)
Pin Header	2pin * 2 (pcs.)
Controller IC	
Arduino Pro Mini	1 (pc.)
2.54mm Female Header	12pin * 2 (pcs.)
Latch IC	
IC 74HC573	4 (pcs.)
20 pin IC Socket	4 (pcs.)
Relay IC	
IC ULN2003	4 (pcs.)
16 pin IC Socket	4 (pcs.)
DuPont Line	8pin * 4 (pcs.)
Pin Header	8pin * 4 (pcs.)
Switches	
switch	3 (pcs.)
Resistor - 10kΩ	3 (pcs.)
DuPont Line	2pin * 3 (pcs.)
Pin Header	2pin * 6 (pcs.)
Speaker	
5W Speaker	1 (pc.)
NPN Transistor	1 (pc.)
Resistor	5.1Ω * 1 (pc.)
	100Ω * 1 (pc.)
DuPont Line	2pin * 1 (pc.)
Pin Header	2pin * 2 (pcs.)
Led-Based 7 Segment Display (using 4 breadboards for 4-digit number)	
5mm Red Led	112 (pcs.)
Resistor	68Ω * 56 (pcs.)
Pin Header	8pin * 4 (pcs.)

- Circuit Design

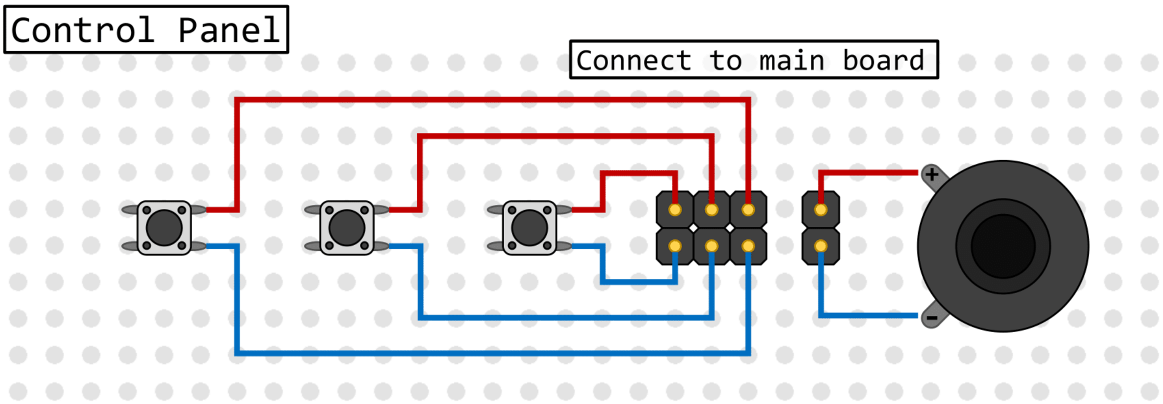
1. Main Board (1 pcs)



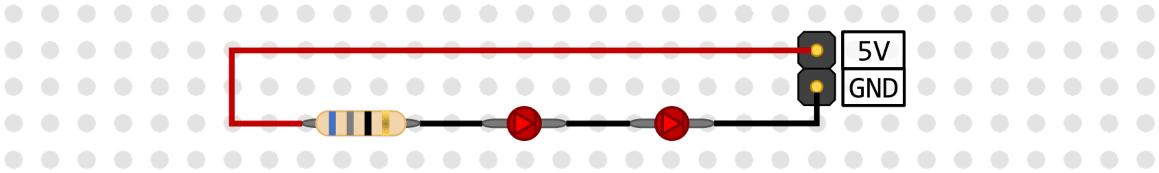
2. Display Board (4 pcs)



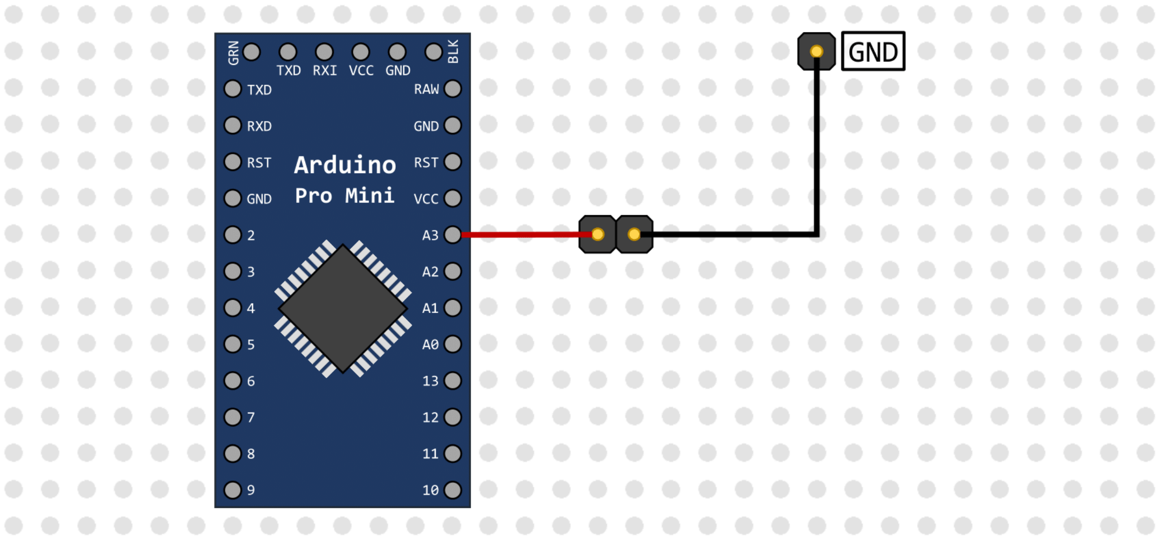
3. Control Panel (1 pcs)



4. Led in middle (Optional)

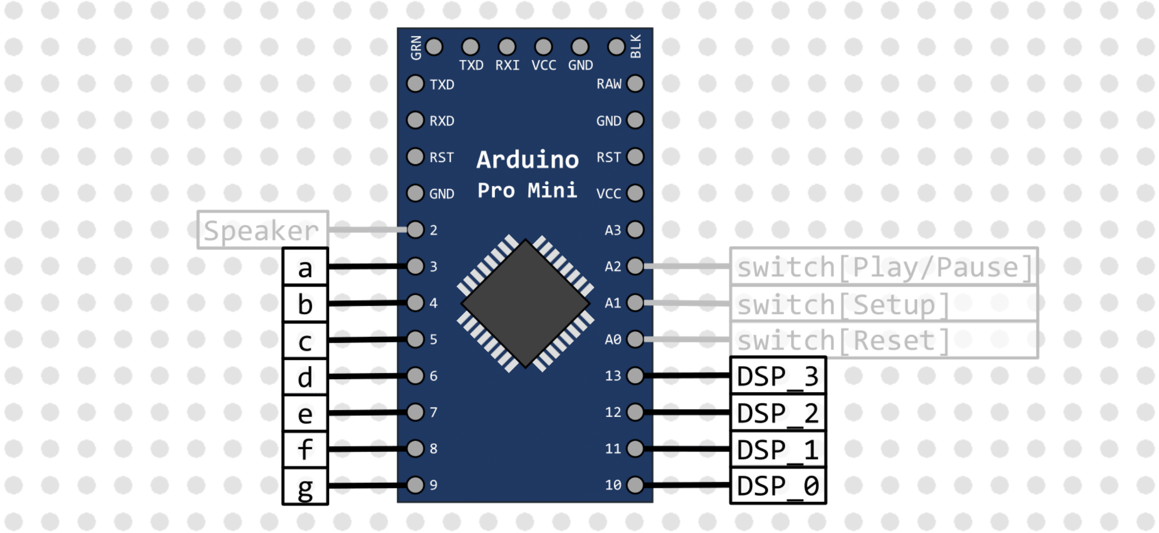


Above circuit can connect to power directly to be always turn-on,
or connect to Arduino pin A3/17 to be controlled programmatically.



Part 3 – Software

● Led Driver



1. Input number and target display to function.
2. Load segment data of number from array that represents **truth table* of 7-segment display. (Store data in byte type to save memory space)

```
const byte numberTo7Seg[] = {
  0x3F, 0x06, 0x5B, 0x4F, 0x66,  /* 0, 1, 2, 3, 4, */
  0x6D, 0x7D, 0x07, 0x7F, 0x6F   /* 5, 6, 7, 8, 9  */
};
```

*For more information about this truth table, please check *Appendix 2*.

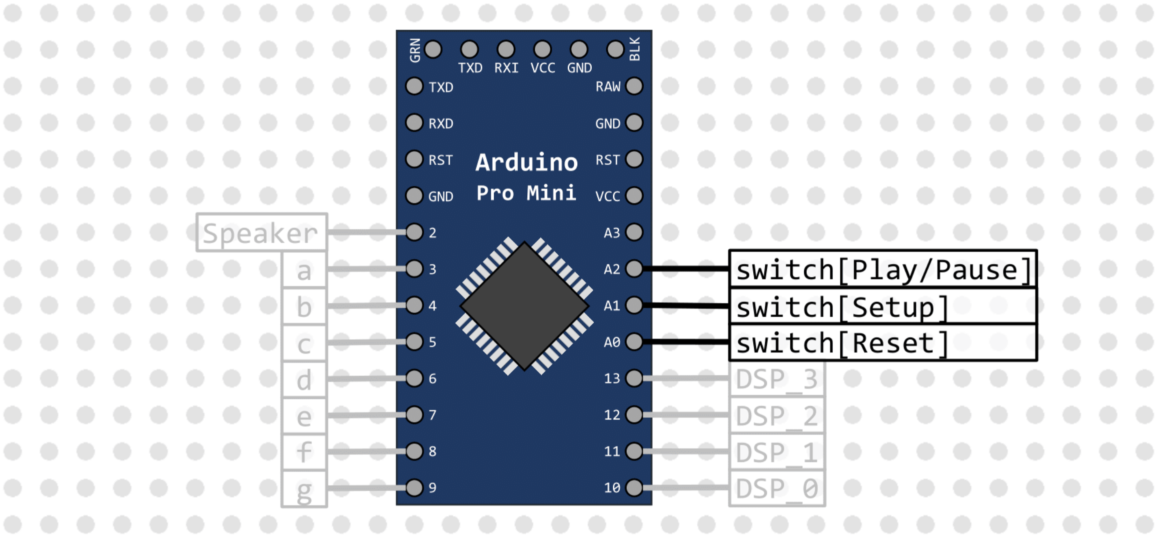
3. Write **Boolean value* from pin_3 to pin_9 (a to g)

   *True for HIGH, False for LOW.

   *Get value by shift (>>) and mask (& 0x01).

4. Set pin of target display as HIGH for 20 microseconds then set as LOW.
   The pin connects to pin_11(LE) of IC-74573. Setting this pin as HIGH will make IC-74573 set its outputs same as inputs. Setting as LOW will lock outputs.

● Switch



1. Read states of switches from `pin_A0/14`, `pin_A1/15`, `pin_A2/16`.

2. Compare current state with previous state:

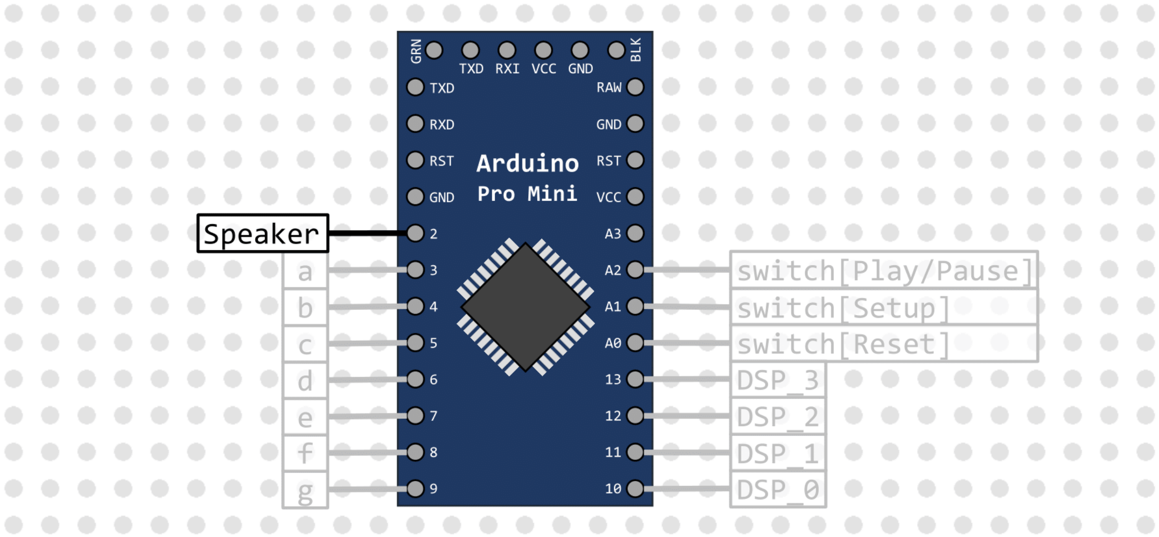
LOW	to	HIGH	Rise event
HIGH	to	HIGH	Hold event
HIGH	to	LOW	Fall event

3. According to events, do specific function.

4. Wait for 25 milliseconds in order to debounce.

5. Repeat.

- Speaker



1. Call `void song()` function every 25 milliseconds.
2. In `song()` function, check the following conditions:

   - the previous tone is completed
   - the melody is not completed

   If so, play next tone by call Arduino built in function:

```
tone(pin_2, melody[playPosition], duration[playPosition] * 0.9);
```

The arrays in argument[1] and argument[2] are showed following:

```
const int melody[] = {
  NOTE_,
  NOTE_E5,  NOTE_D5,  NOTE_FS4, NOTE_GS4,
  NOTE_CS5, NOTE_B4,  NOTE_D4,  NOTE_E4,
  NOTE_B4,  NOTE_A4,  NOTE_CS4, NOTE_E4,  NOTE_A4
};
```

```
const int duration[] = {
  0,  // millisecond to start play
  125, 125, 250, 250,
  125, 125, 250, 250,
  125, 125, 250, 250, 500
};
```

*This is “Nokia tune”.

Timer

Arduino Pro Mini (ATmega328P) has 3 timers.

Because `delay()` and `delayMicroseconds()` functions which use `Timer0` and `tone()` function which uses `Timer2` are already used, `Timer1` will be used in this case.

1. About `Timer1` and Arduino Pro Mini clock:
   - `Timer1` is a 16-bit timer. It means maximum value of its counter is 65536(2^{16}).
   - The clock rate of Arduino Pro Mini (ATmega328P) is 16MHz.

2. Using 1/64 prescaler.
   - Set register `CS12`, `CS11`, `CS10` as `0`, `1`, `1`.

   ```
   TCCR1B |= (1<<CS10) | (1<<CS11);
   ```

   - `Timer1` is 250K ticks per second now. (250KHz)

3. Using CTC mode (Clear Timer on Compare match).
   - Set register `WGM12` as `1`.

   ```
   TCCR1B |= (1<<WGM2);
   ```

4. Timer Interrupt occurs every 10 milliseconds.
   - Set register OCR1A as `2500`.

   ```
   OCR1A = 2500;
   ```

   - This countdown timer is designed to show minute and second, and second, hundred-millisecond and ten-millisecond while rest time is less than a minute. So the minimum time unit is designed as 10 milliseconds. It means timer Interrupt occurs every 10 milliseconds is suit for this case.

5. Enable timer compare interrupt by setting register OCIE1A as 1.

   ```
   TIMSK1 |= (1<<OCIE1A);    // Mask xxxxxx1x
   ```

6. Disable timer compare interrupt by setting register OCIE1A as 0.

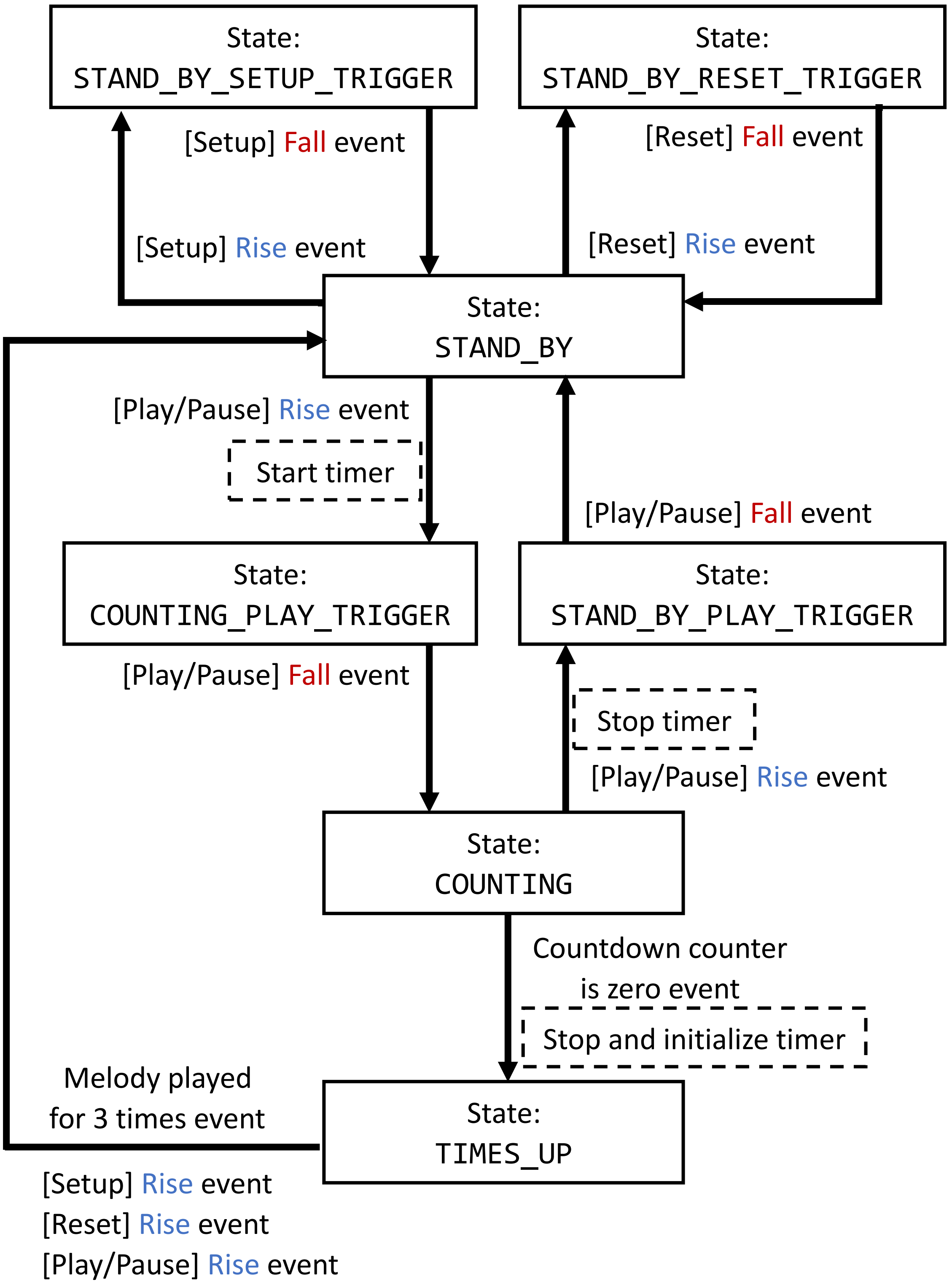
   ```
   TIMSK1 &= (~(1<<OCIE1A)); // Mask xxxxxx0x
   ```

7. While timer compare interrupt occurs, Interrupt Service Routine will be called.

   ```
   ISR(TIMER1_COMPA_vect){ /* Do something */ }
   ```

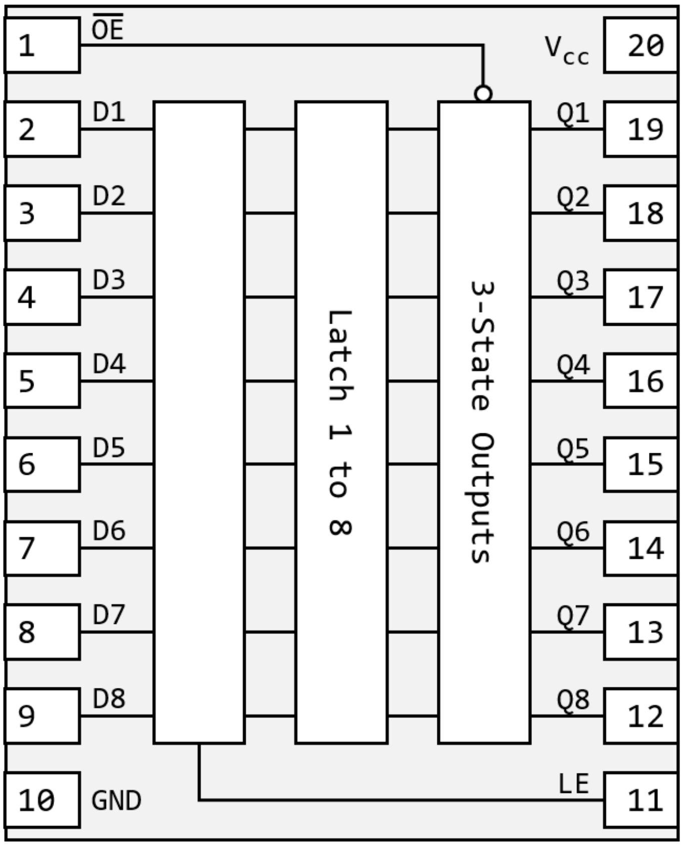
*For more information about timer and register, please check *Appendix 3*.

- State Diagram



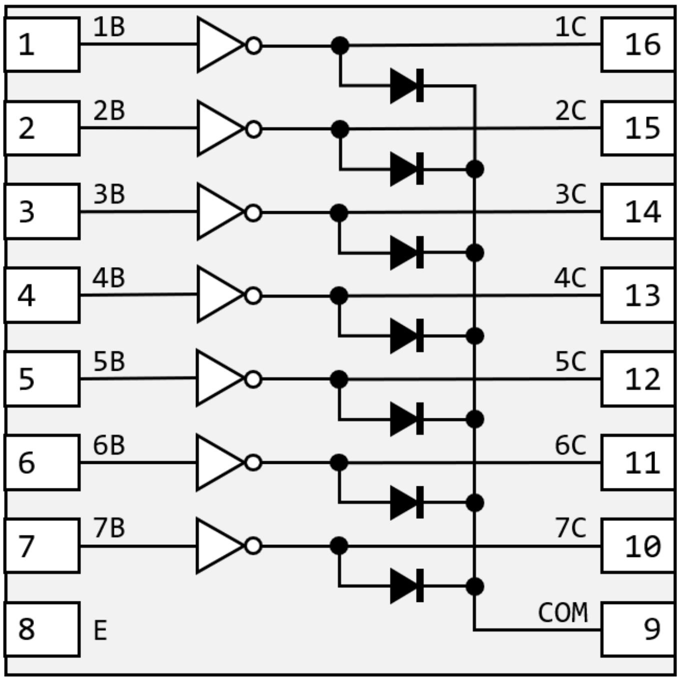
Appendix 1 - IC Datasheets

- 74HC573



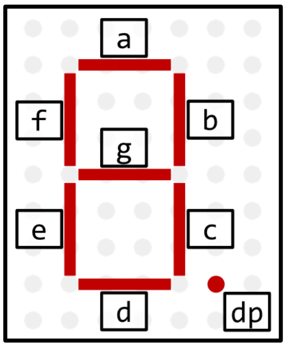
INPUTS			OUTPUT
$\overline{OE}$	LE	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

- ULN2003

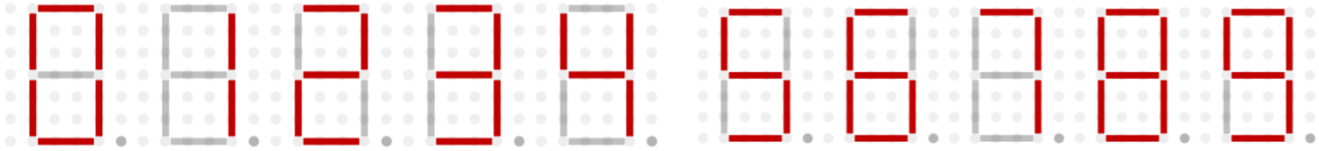


Appendix 2 - Seven Segment Display

- Segments Definition



- Segments of Decimal Numbers



- 7-Segment Display Truth Table

Segment Number	dp	g	f	e	d	c	b	a	Byte code
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

Appendix 3 - Arduino Timers and Registers

● Arduino Pro Mini (ATmega328P) Timers

1. `Timer0`
   - This is an 8-bit timer.
   - `delay()`, `millis()`, `micros()` functions use this timer.

2. `Timer1`
   - This is a 16-bit timer.
   - `Servo library` uses this timer.

3. `Timer2`
   - This is an 8-bit timer.
   - `tone()` function uses this timer.

● Timer1 Registers on ATmega328P

■ **TCCR1A** - **T**imer/**C**ounter1 **C**ontrol **R**egister A

Bit	7	6	5	4	3	2	1	0
(0x80)	**COM1A1**	**COM1A0**	**COM1B1**	**COM1B0**	**-**	**-**	**WGM11**	**WGM10**
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

■ **TCCR1B** - **T**imer/**C**ounter1 **C**ontrol **R**egister B

Bit	7	6	5	4	3	2	1	0
(0x81)	**ICNC1**	**ICES1**	**-**	**WGM13**	**WGM12**	**CS12**	**CS11**	**CS10**
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **COM1A1, COM1A0**: **C**ompare **O**utput **M**ode for Channel A
- **COM1B1, COM1B0**: **C**ompare **O**utput **M**ode for Channel B
- **WGM13, WGM12, WGM11, WGM10**: **W**aveform **G**eneration **M**ode

Mode	4
WGM13, WGM12, WGM11, WGM10	0, 1, 0, 0
Timer/Counter Mode of Operation	**CTC*
TOP	OCR1A
Update of OCR1x at	Immediate
TOV1 Flag Set on	MAX

*CTC: **C**lear **T**imer on **C**ompare match mode.

*For more mode setting by these registers, please check ATmega328 official datasheet.

- **CS12, CS11, CS10**: **C**lock **S**elect

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	No prescaling
0	1	0	1/8 prescaler
0	1	1	1/64 prescaler
1	0	0	1/256 prescaler
1	0	1	1/1024 prescaler
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.

■ **TCNT1**: **T**imer/**C**ounter1

Bit	7	6	5	4	3	2	1	0
(0x85)	TCNT1[15:8]							
(0x84)	TCNT1 [7:0]							
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

■ **OCR1A**: **O**utput **C**ompare **R**egister 1 A

Bit	7	6	5	4	3	2	1	0
(0x89)	OCR1A[15:8]							
(0x88)	OCR1A [7:0]							
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

This register is continuously compared with **TCNT1**. A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1x pin.

■ **TIMSK1**: **T**imer/**C**ounter1 **I**nterrupt **M**ask **R**egister

Bit	7	6	5	4	3	2	1	0
(0x6F)	-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **OCIE1A**: Timer/Counter1, **O**utput **C**ompare A Match **I**nterrupt **E**nable
  When this bit is written to 1 and interrupts globally enabled, the Timer/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the **OCF1A** Flag, located in **TIFR1**, is set.
- In Arduino programming, use ISR (Interrupt Service Routine) to do user defined work when the flag is set.
- In Arduino, TIMER1_COMPA_vect is represented as Timer/Counter1 Compare Match A Interrupt Vector, located at 0x00B, in ATmega328P.

```
ISR(TIMER1_COMPA_vect){ /* Do something */ }
```

*For more details about timer and register, please check ATmega328 official datasheet.