Arduino Internals: Atmel ATmega328P

Dr Joshua Ellul

Eclipse IDE

- Download Eclipse:
  - https://www.eclipse.org/downloads/

- Install:
  - Choose Eclipse for C/C++

- Launch, then:
  - Help > Eclipse Marketplace
    - Find “Eclipse C++ IDE for Arduino 3.0”
      - Install

Install Arduino Uno Platform

- Help > Arduino Downloads Manager
  - In the “Platforms” tab
    - Add
      - Select the “Arduino AVR Boards” platform
      - Press “OK”
      - Press “Done”

Add the board

- Window > Show View > Other > Connections > Connections
- In the “Connections” window:
  - Create a New Connection (use the icon with a +, or right click)
    - Select Arduino
    - Target Name: ArduinoUno
    - Serial port: Use the same as the Arduino IDE
    - Board type: Arduino/Genuino Uno
    - Programmer: AVR ISP

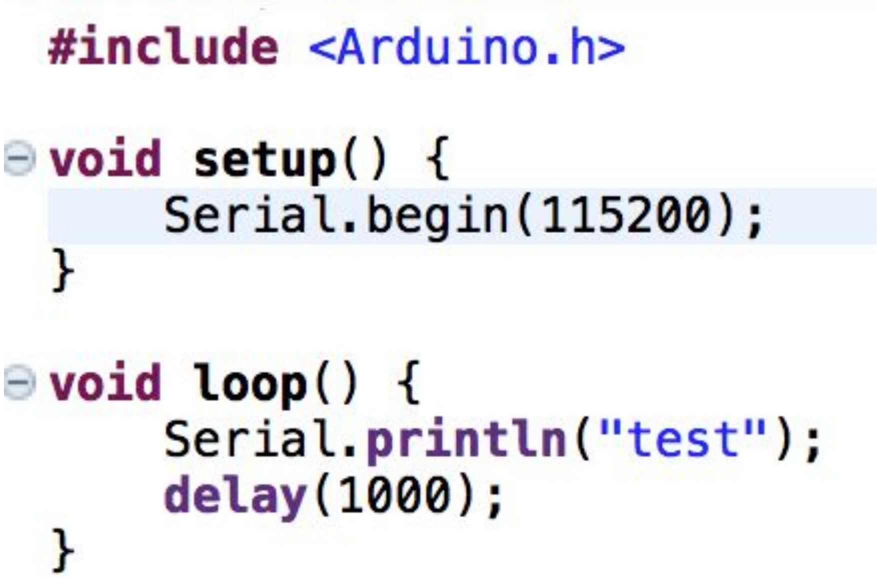
First Test Project

- File > New > Arduino Project
    - if it’s not there then: File > New > Project
        - Expand the C/C++ tree node
            - Choose Arduino Project

- Choose “Arduino C++ Sketch”
    - Project name: “Test”
    - Finish

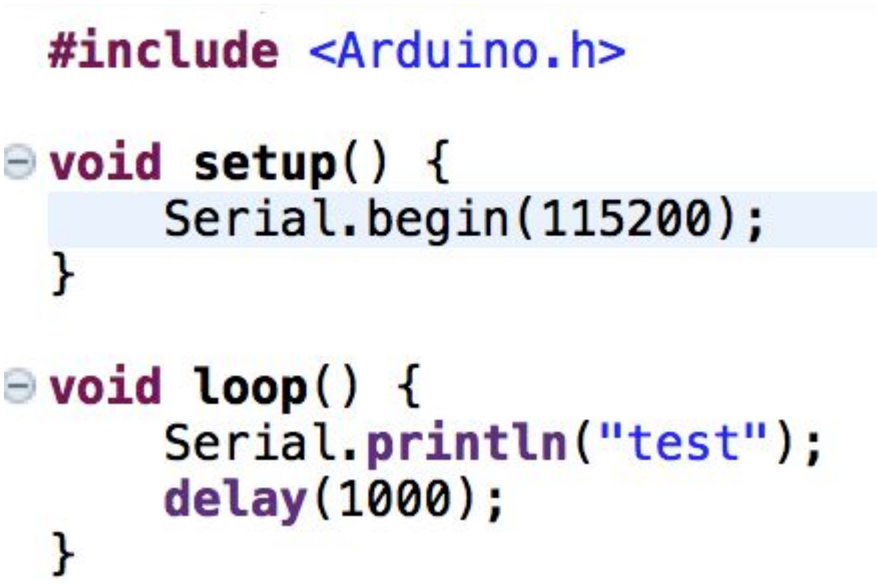
- Try building it (using the hammer)
- Try running it (using the run button next to the hammer)

Test Serial



```
#include <Arduino.h>

void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("test");
    delay(1000);
}
```

Test Serial



```
#include <Arduino.h>

void setup() {
	Serial.begin(115200);
}

void loop() {
	Serial.println("test");
	delay(1000);
}
```

Try it out

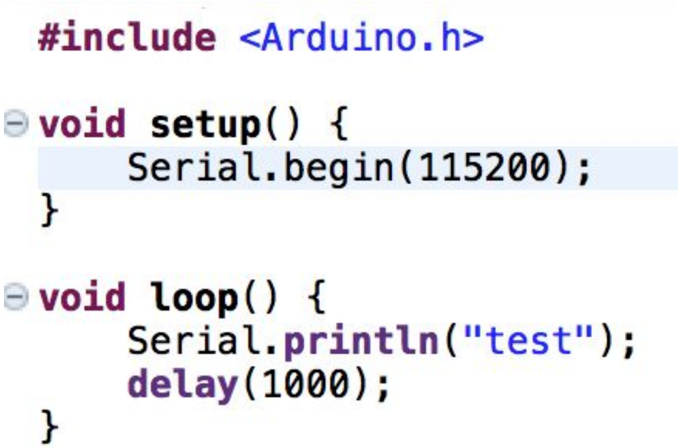
Test Serial

- Open the “Connections” window
    - Right click on “Arduino”
        - Select “Open Command Shell”

- Make sure to close the command shell by pressing the “X” in the console

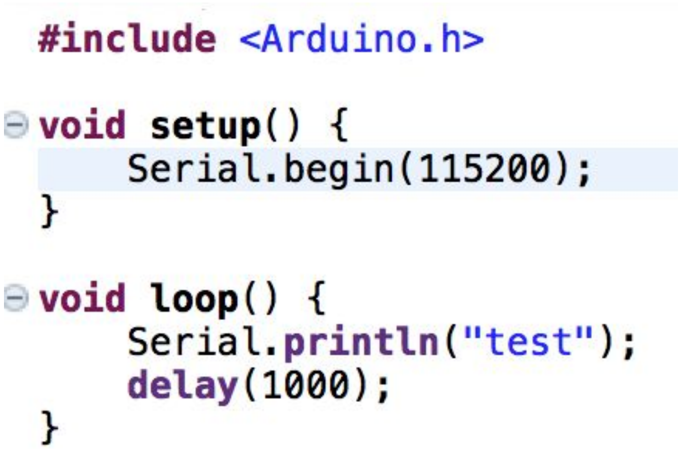
- If you have problems programming the device, make sure all command shells in the console are closed; if it still persists unplug and plug your arduino back in

But where’s main() ?

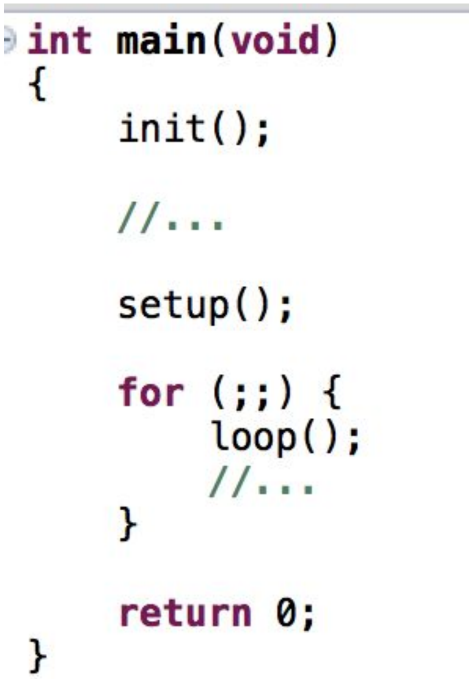


```
#include <Arduino.h>

void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("test");
    delay(1000);
}
```

But where's main() ?

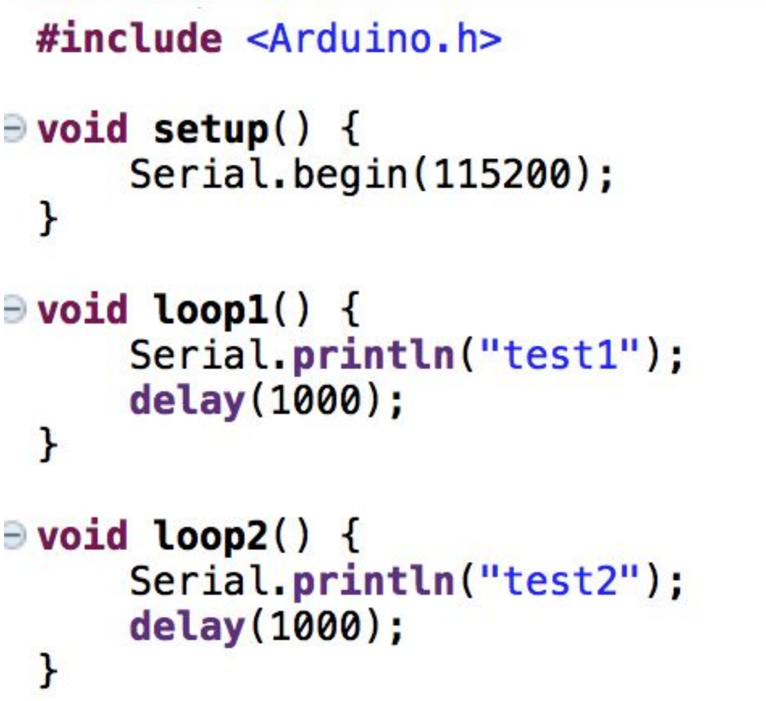


```
#include <Arduino.h>

void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("test");
    delay(1000);
}
```

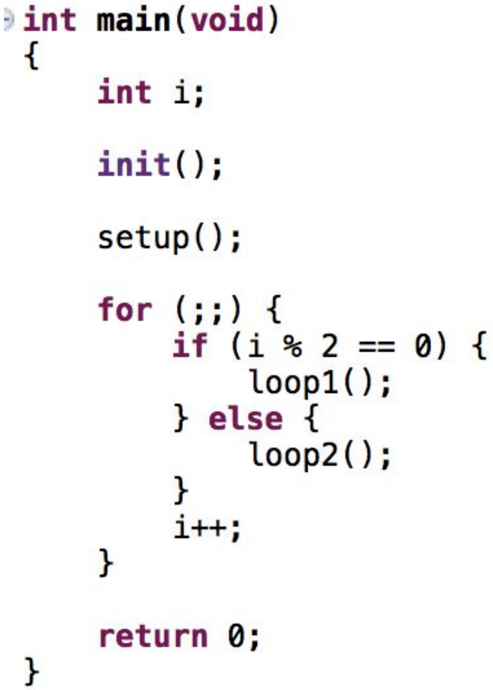


```
int main(void)
{
    init();

    //...

    setup();

    for (;;) {
        loop();
        //...
    }

    return 0;
}
```

Let’s code a different Arduino main()



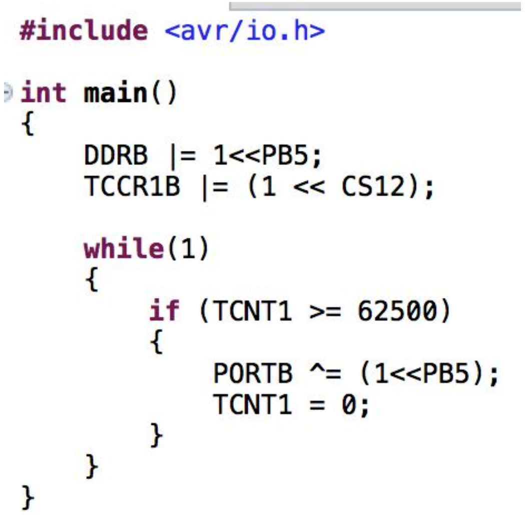
```
#include <Arduino.h>

void setup() {
    Serial.begin(115200);
}

void loop1() {
    Serial.println("test1");
    delay(1000);
}

void loop2() {
    Serial.println("test2");
    delay(1000);
}
```



```
int main(void)
{
    int i;

    init();

    setup();

    for (;;) {
        if (i % 2 == 0) {
            loop1();
        } else {
            loop2();
        }
        i++;
    }

    return 0;
}
```

Let’s get familiar with the Microcontroller

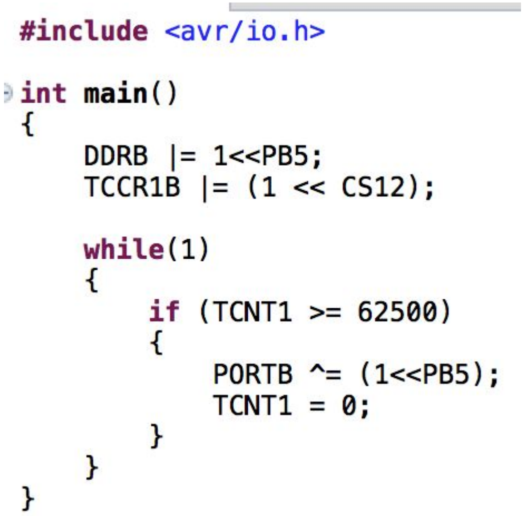
- Create a new Arduino C++ Project
- Remove the template code and use the following instead:



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Let’s get familiar with the Microcontroller

- Create a new Arduino C++ Project
- Remove the template code and use the following instead:



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Try it out… you should see the LED blink

Find the Atmel ATmega328P Datasheet

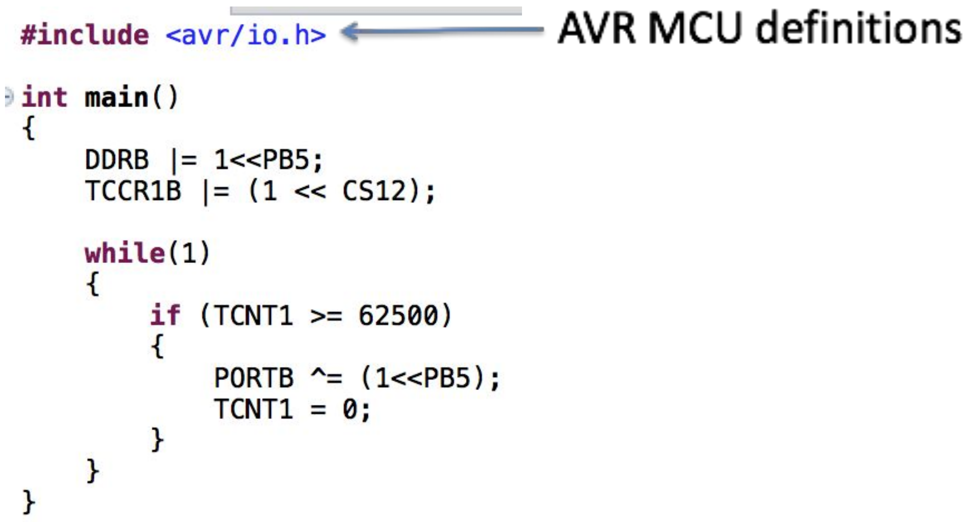
Atmel®

ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

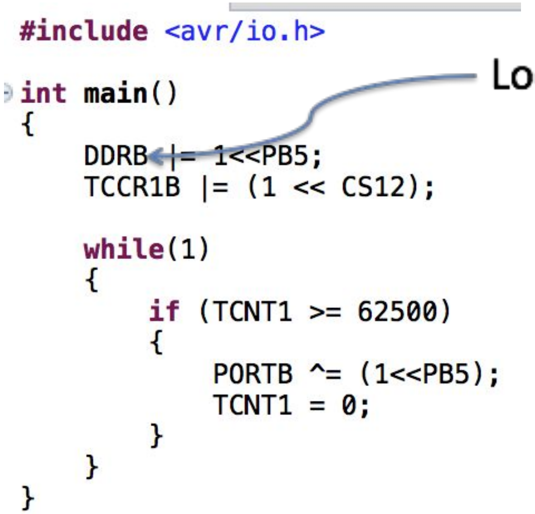
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

AVR MCU definitions

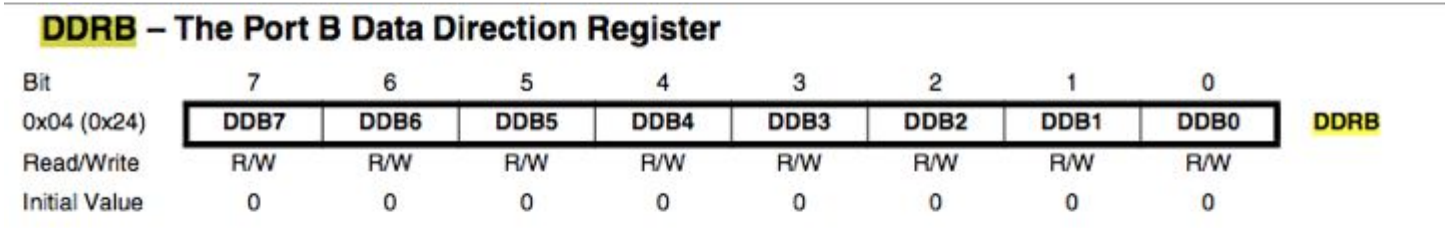
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Lookup the DDRB register in the datasheet

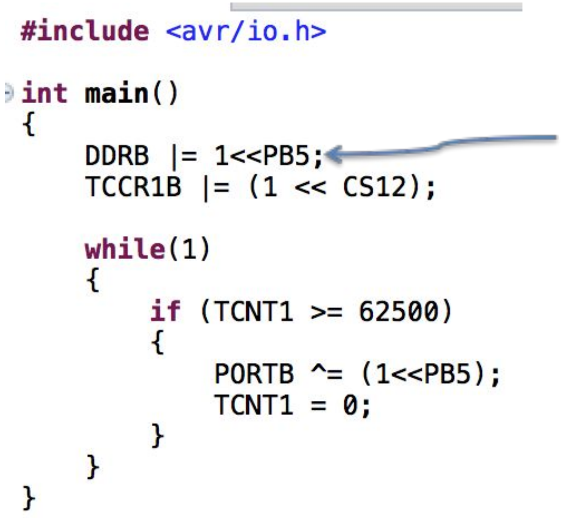
DDRB



DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

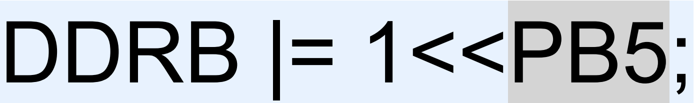
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

PB5 signifies Port B's Pin 5

PB5

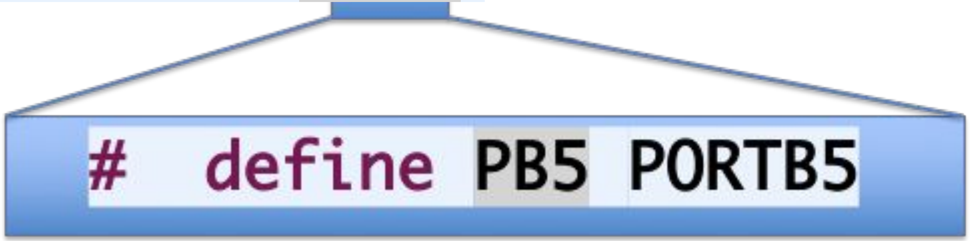


```
DDRB |= 1<<PB5;
```

Press cmd or ctrl and click on PB5

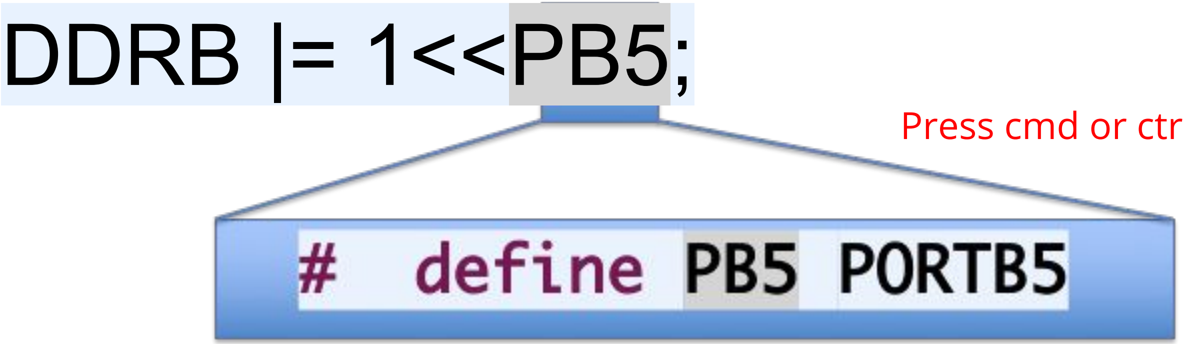
PB5

```
DDRB |= 1<<PB5;
```



```
#  define PB5 PORTB5
```

PB5

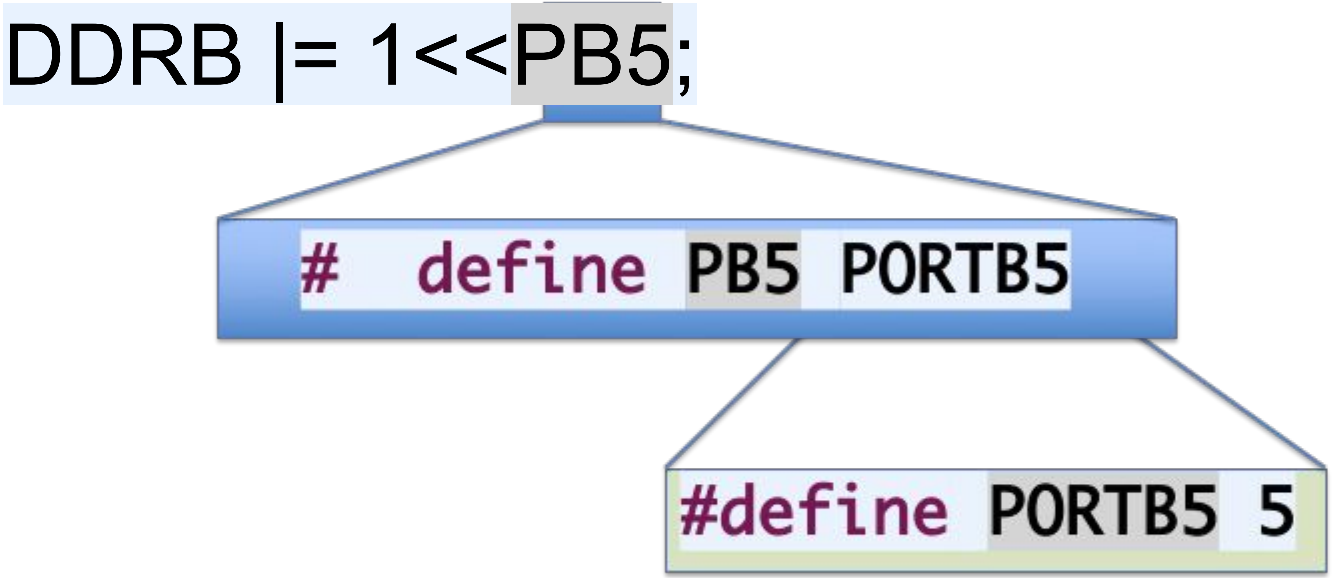


```
DDRB |= 1<<PB5;
```

```
# define PB5 PORTB5
```

Press cmd or ctrl and click on PORTB5

PB5

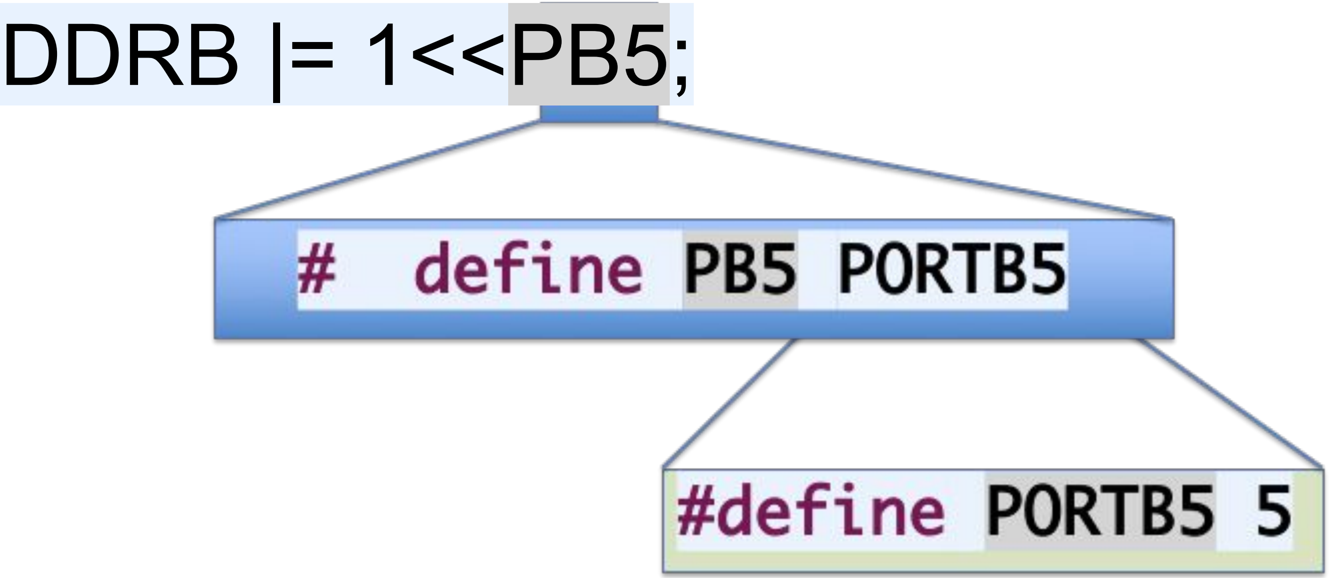


```
DDRB |= 1<<PB5;
```

```
#  define PB5 PORTB5
```

```
#define PORTB5 5
```

PB5



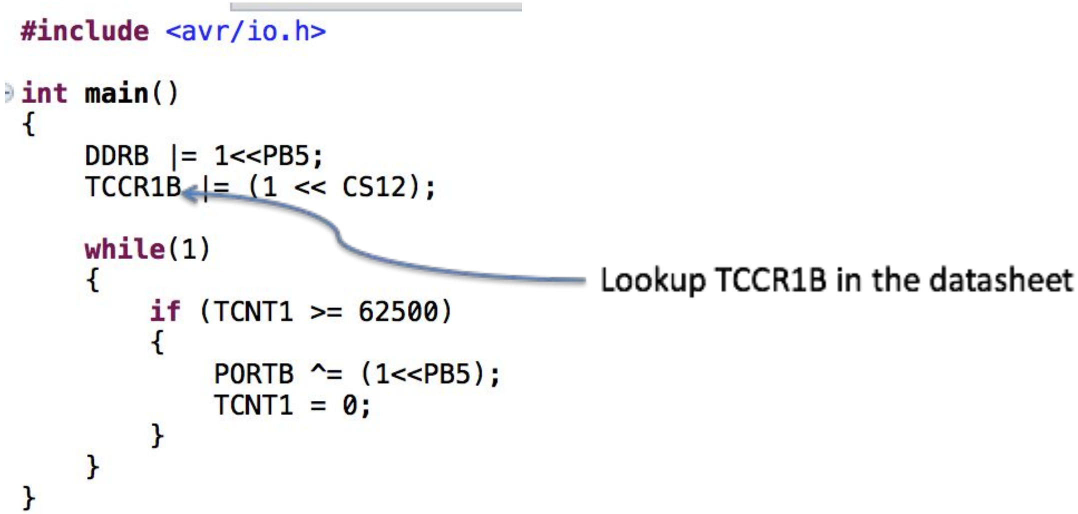
```
DDRB |= 1<<PB5;
```

```
#  define PB5 PORTB5
```

```
#define PORTB5 5
```

1<<PB5 ← this changes from pin number 5 to pin value

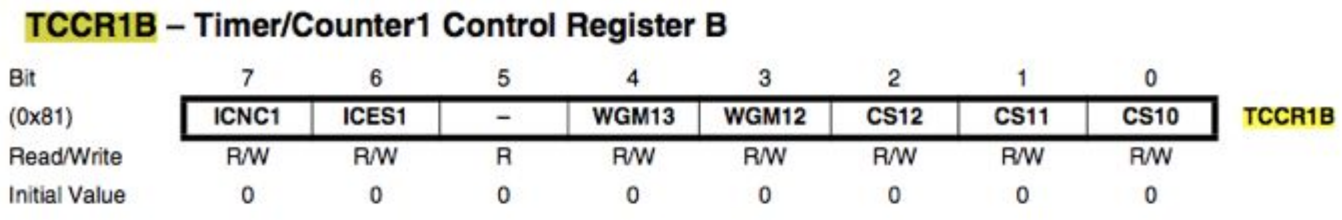
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Lookup TCCR1B in the datasheet

TCCR1B



TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
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	

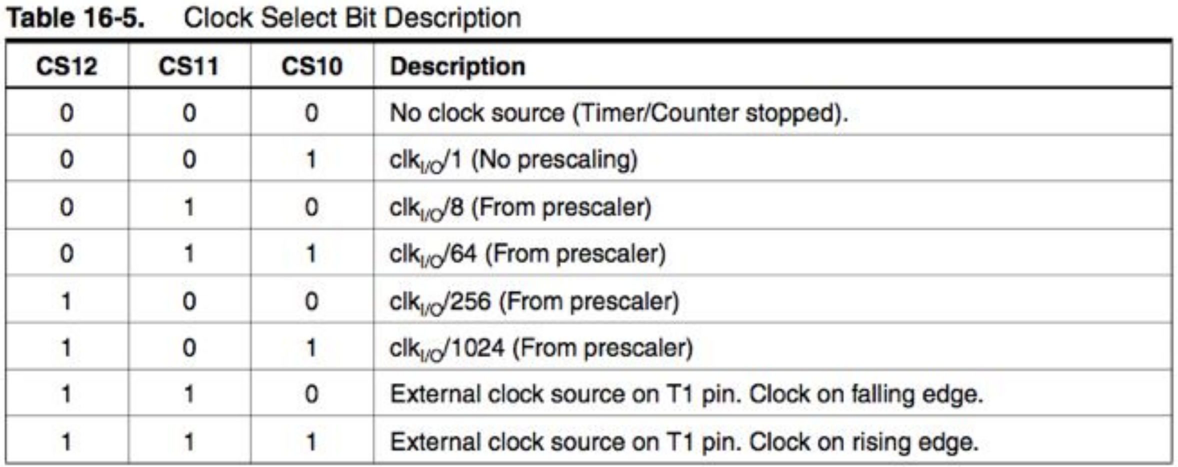


Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	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	clk$_{I/O}$/256 (From prescaler)
1	0	1	clk$_{I/O}$/1024 (From 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.

ICNC1: Input Capture Noise Canceller

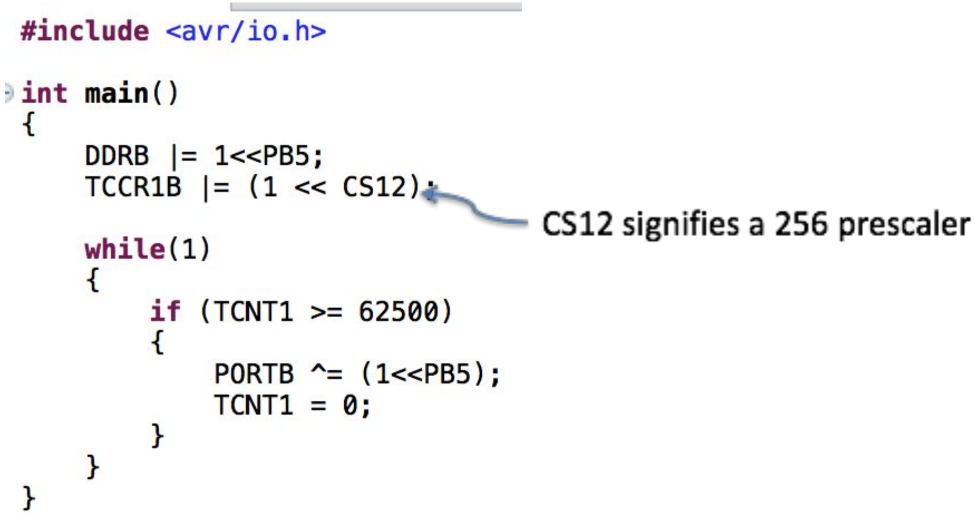
ICES1: Input Capture Edge Select

Bit 5 – Reserved

WGM13:2: Waveform Generation Mode

CS12:0: Clock Select

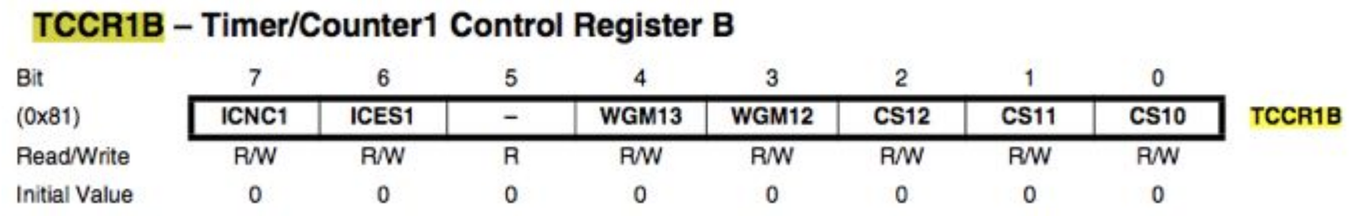
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

CS12 signifies a 256 prescaler

TCCR1B



TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
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	

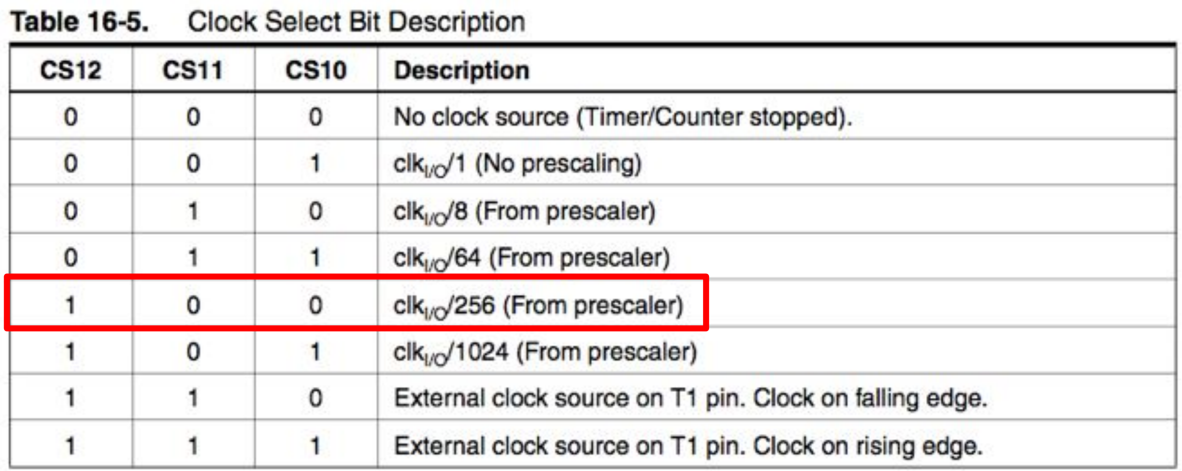


Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	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	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From 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.

ICNC1: Input Capture Noise Canceller

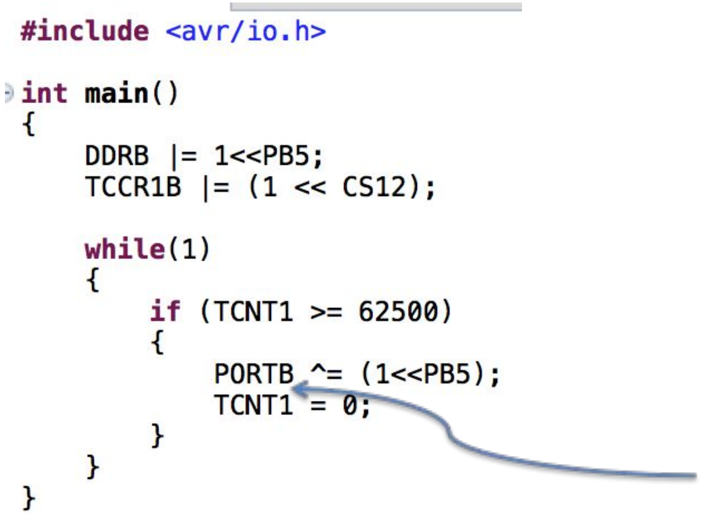
ICES1: Input Capture Edge Select

Bit 5 – Reserved

WGM13:2: Waveform Generation Mode

CS12:0: Clock Select

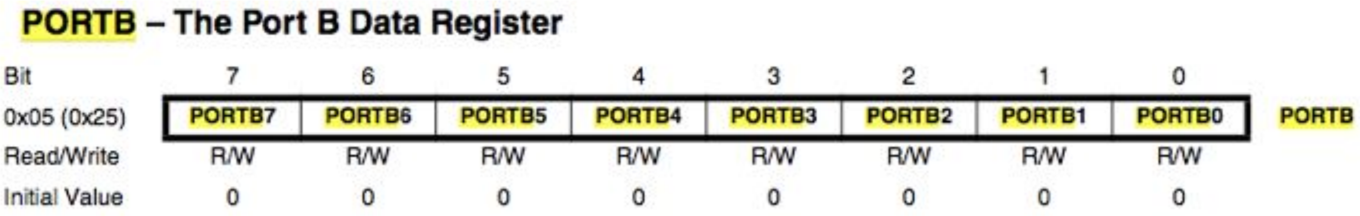
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Look it up

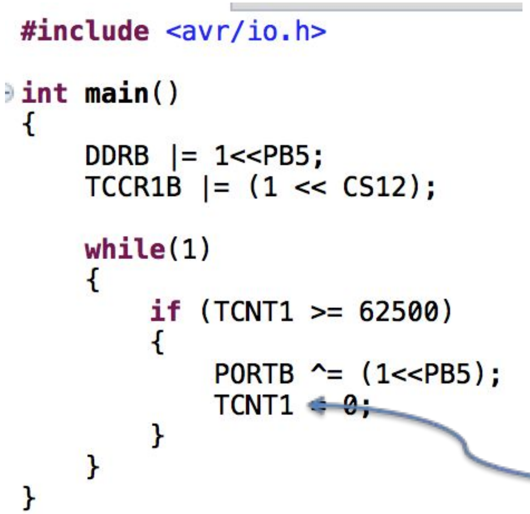
PORTB



PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

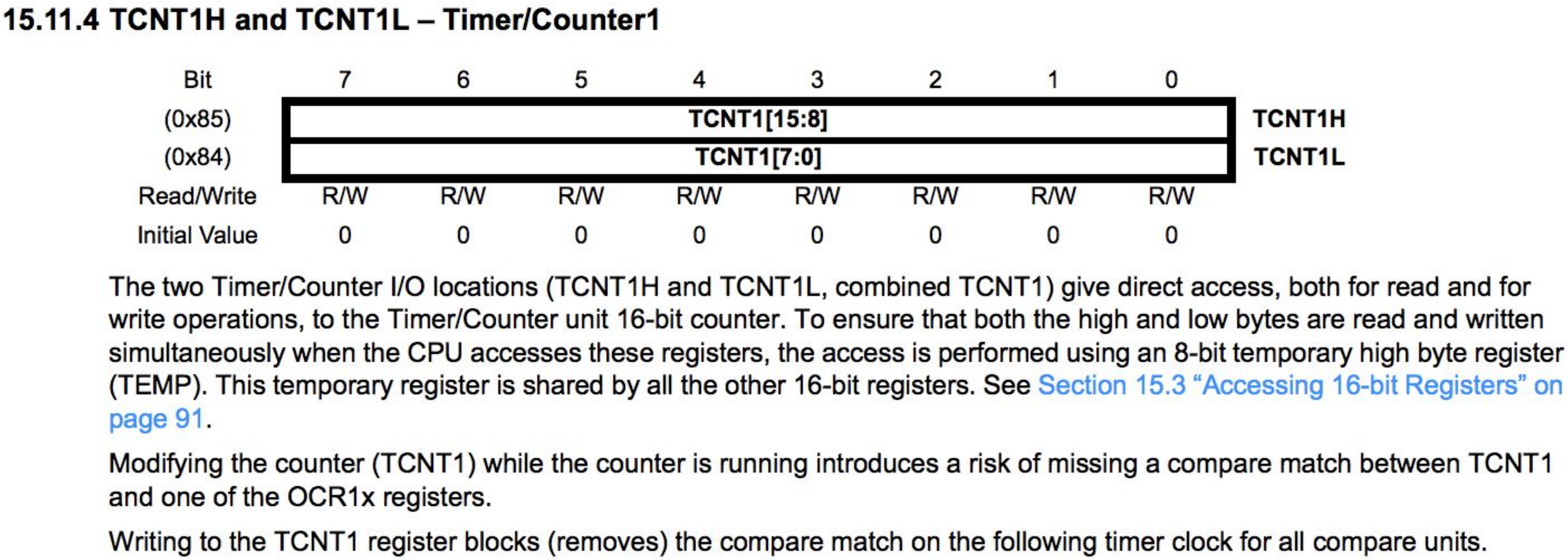
Making sense of the code



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

int main()
{
    DDRB |= 1<<PB5;
    TCCR1B |= (1 << CS12);

    while(1)
    {
        if (TCNT1 >= 62500)
        {
            PORTB ^= (1<<PB5);
            TCNT1 = 0;
        }
    }
}
```

Look it up

TCNT1



15.11.4 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

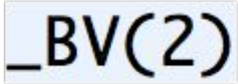
The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 15.3 "Accessing 16-bit Registers" on page 91.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x registers.

Writing to the TCNT1 register blocks (removes) the compare match on the following timer clock for all compare units.

_BV

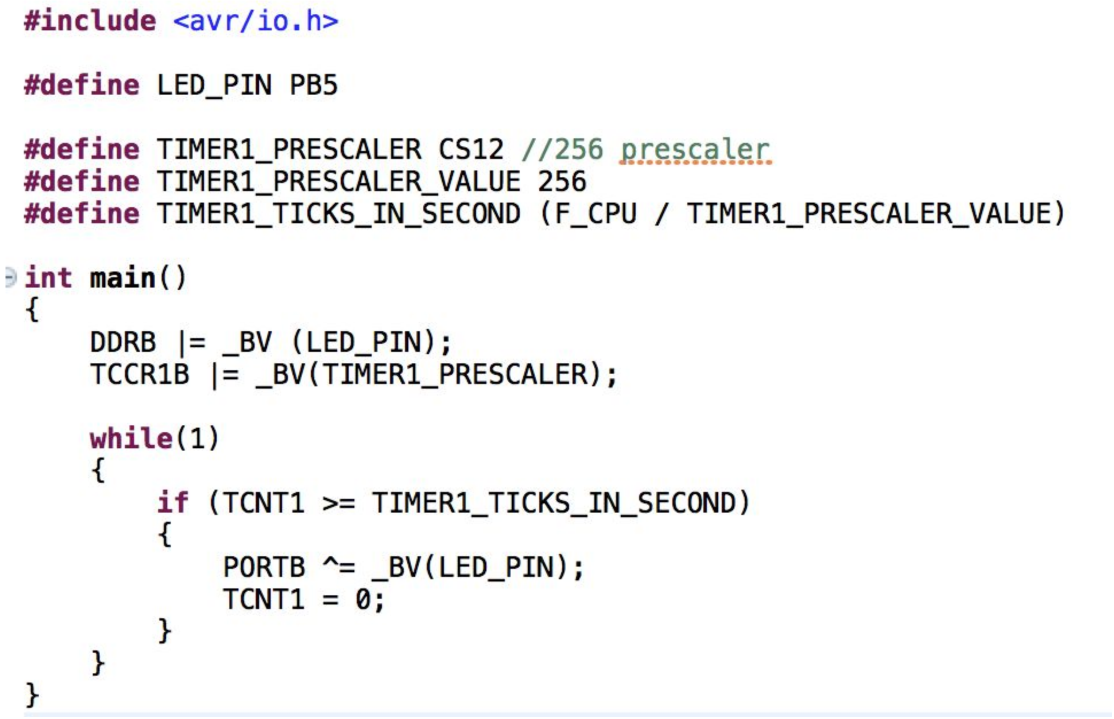
- Converting bit numbers, to byte values:



```
_BV(2)
```

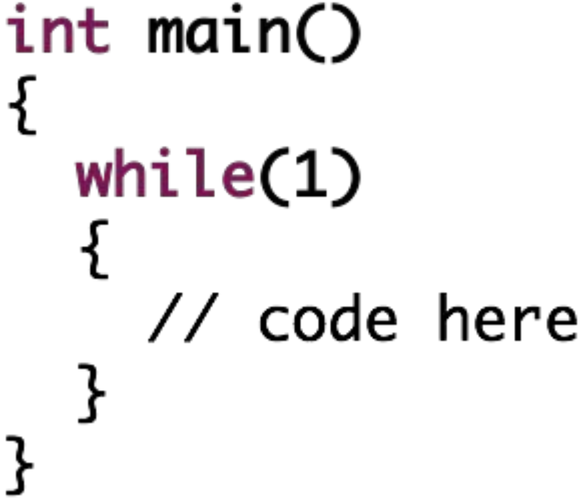
→ 4

Refactoring



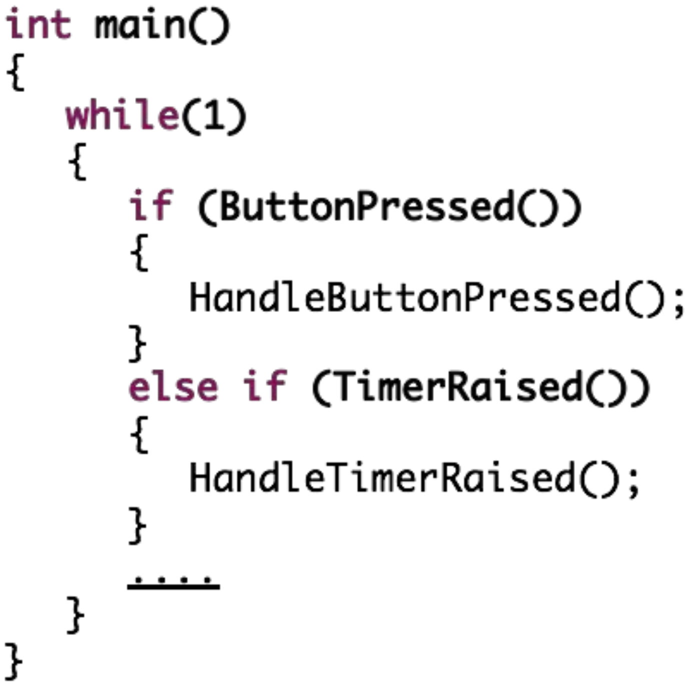
```
#include <avr/io.h>

#define LED_PIN PB5

#define TIMER1_PRESCALER CS12 //256 prescaler
#define TIMER1_PRESCALER_VALUE 256
#define TIMER1_TICKS_IN_SECOND (F_CPU / TIMER1_PRESCALER_VALUE)

int main()
{
    DDRB |= _BV (LED_PIN);
    TCCR1B |= _BV(TIMER1_PRESCALER);

    while(1)
    {
        if (TCNT1 >= TIMER1_TICKS_IN_SECOND)
        {
            PORTB ^= _BV(LED_PIN);
            TCNT1 = 0;
        }
    }
}
```

One Main Thread of Execution



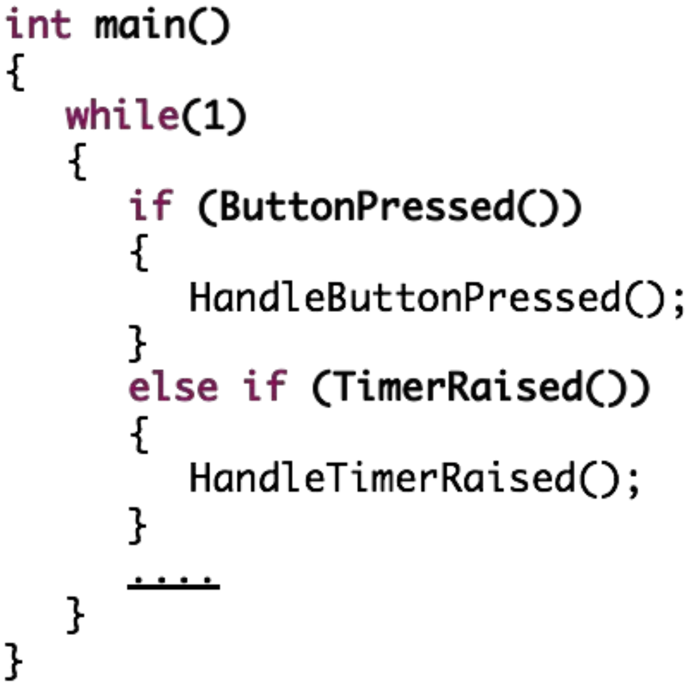
```
int main()
{
  while(1)
  {
    // code here
  }
}
```

One Main Thread of Execution



```
int main()
{
   while(1)
   {
      if (ButtonPressed())
      {
         HandleButtonPressed();
      }
      else if (TimerRaised())
      {
         HandleTimerRaised();
      }
      ....
   }
}
```

One Main Thread of Execution

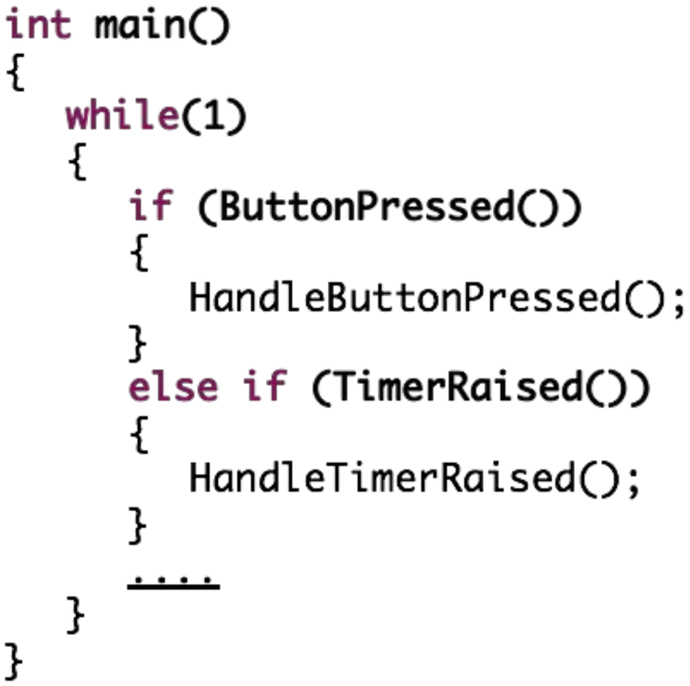


```
int main()
{
   while(1)
   {
      if (ButtonPressed())
      {
         HandleButtonPressed();
      }
      else if (TimerRaised())
      {
         HandleTimerRaised();
      }
      ....
   }
}
```

Polling:

- Check if event has occurred, if so handle it

One Main Thread of Execution

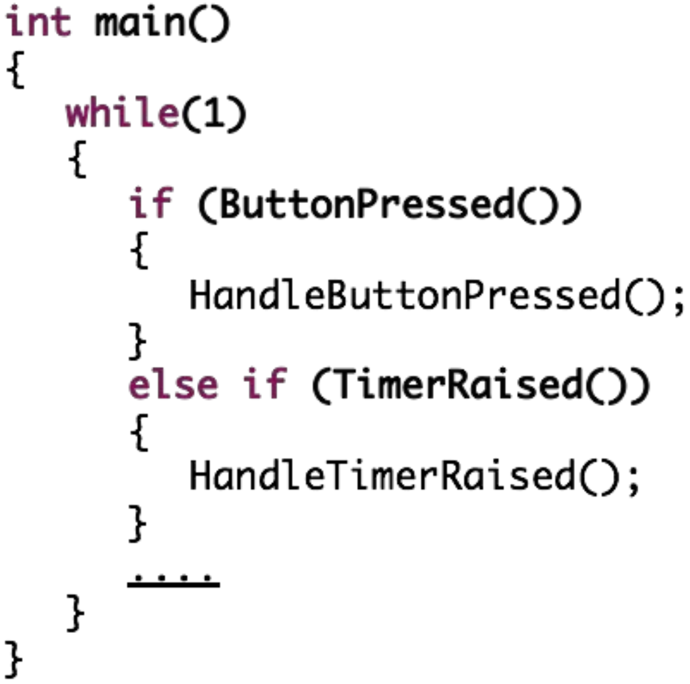


```
int main()
{
   while(1)
   {
      if (ButtonPressed())
      {
         HandleButtonPressed();
      }
      else if (TimerRaised())
      {
         HandleTimerRaised();
      }
      ....
   }
}
```

Polling:

- Check if event has occurred, if so handle it
- Only one event can be checked at a time (some events will receive priority)

One Main Thread of Execution

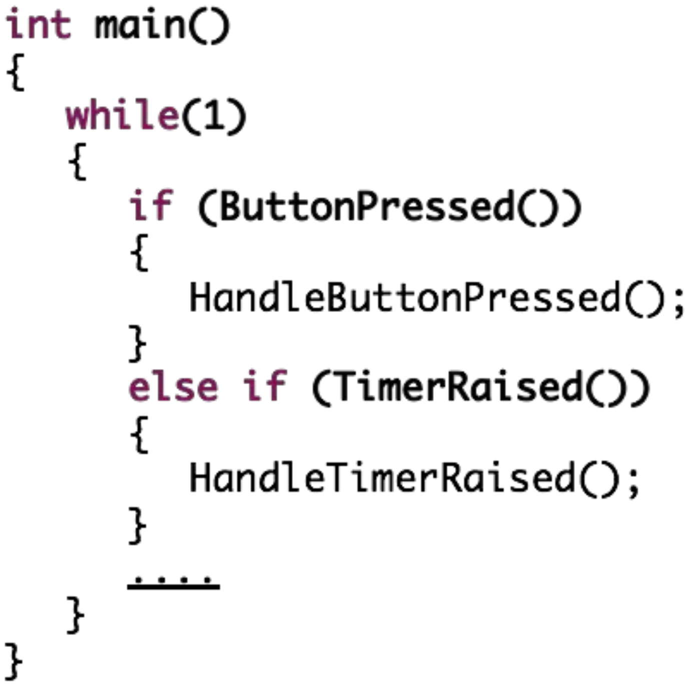


```
int main()
{
   while(1)
   {
      if (ButtonPressed())
      {
         HandleButtonPressed();
      }
      else if (TimerRaised())
      {
         HandleTimerRaised();
      }
      ....
   }
}
```

Polling:

- Check if event has occurred, if so handle it
- Only one event can be checked at a time (some events will receive priority)
- Checking itself requires computation/CPU cycles

One Main Thread of Execution

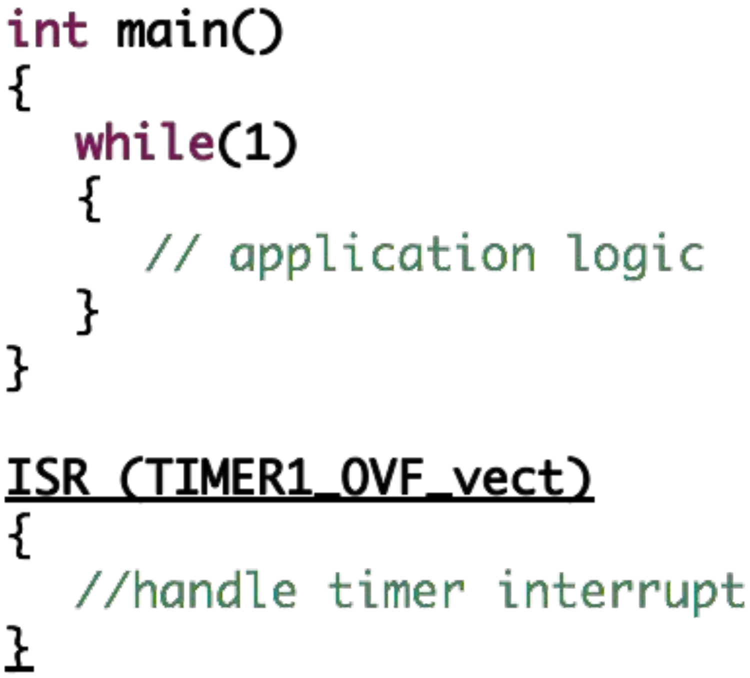


```
int main()
{
   while(1)
   {
      if (ButtonPressed())
      {
         HandleButtonPressed();
      }
      else if (TimerRaised())
      {
         HandleTimerRaised();
      }
      ....
   }
}
```

Polling:

- Check if event has occurred, if so handle it
- Only one event can be checked at a time (some events will receive priority)
- Checking itself requires computation/CPU cycles
- What if the main program is busy doing a long running task, and an important event comes in?

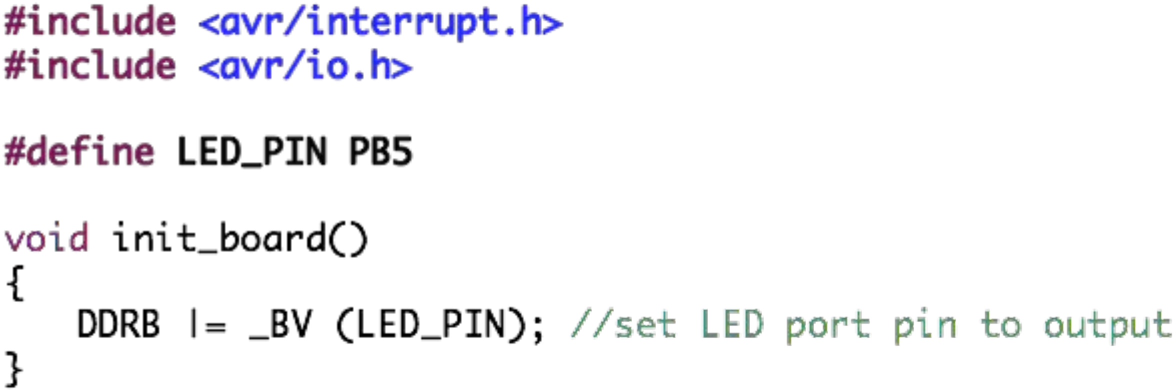
Interrupts



```
int main()
{
   while(1)
   {
      // application logic
   }
}

ISR (TIMER1_OVF_vect)
{
   //handle timer interrupt
}
```

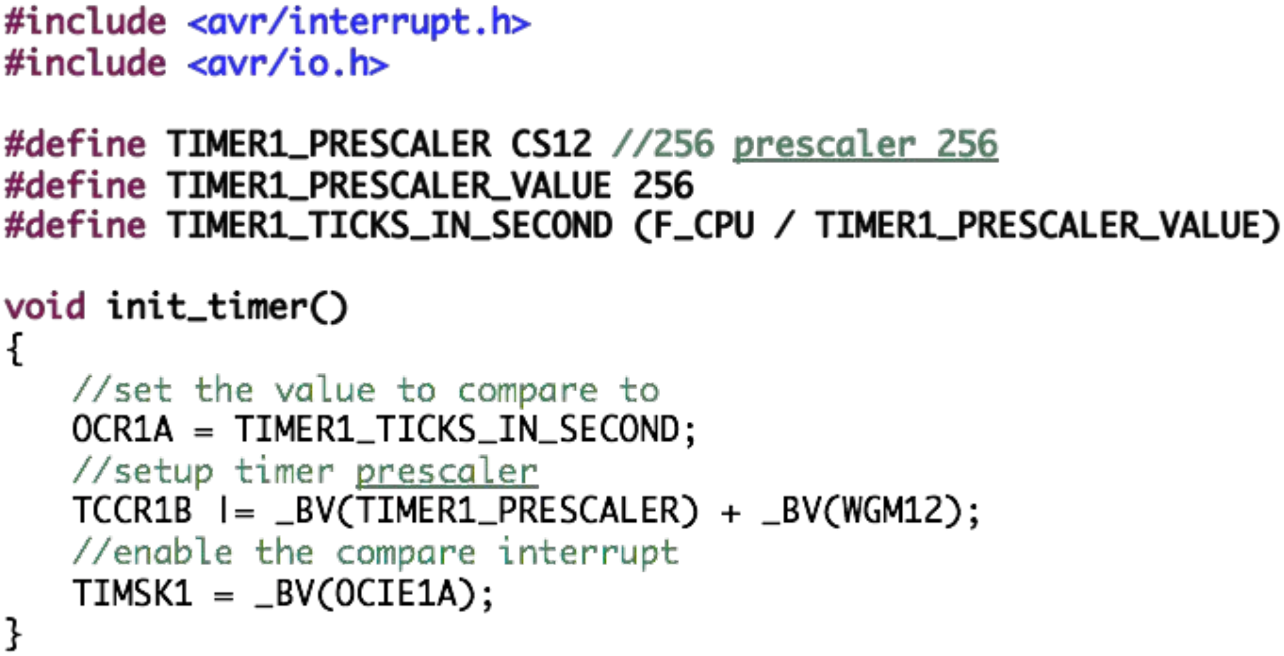
Blink using Interrupts

init_board()



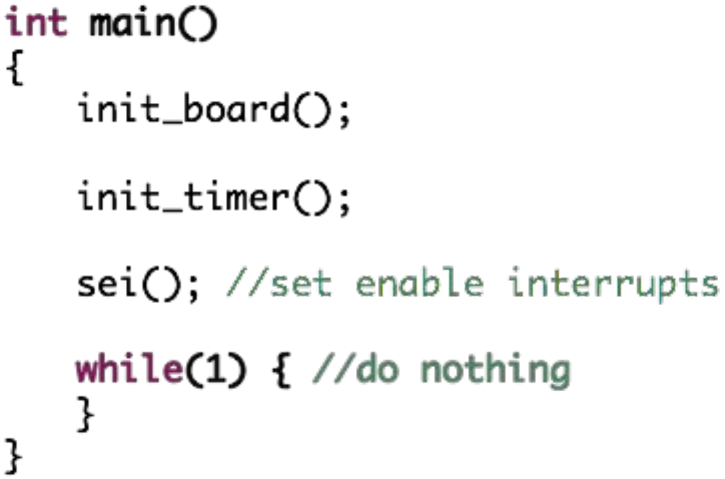
```
#include <avr/interrupt.h>
#include <avr/io.h>

#define LED_PIN PB5

void init_board()
{
    DDRB |= _BV (LED_PIN); //set LED port pin to output
}
```

init_timer()



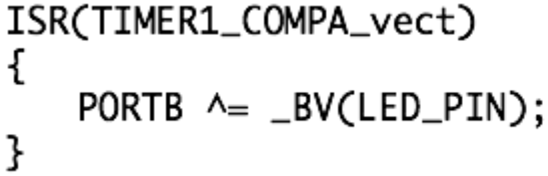
```c
#include <avr/interrupt.h>
#include <avr/io.h>

#define TIMER1_PRESCALER CS12 //256 prescaler 256
#define TIMER1_PRESCALER_VALUE 256
#define TIMER1_TICKS_IN_SECOND (F_CPU / TIMER1_PRESCALER_VALUE)

void init_timer()
{
    //set the value to compare to
    OCR1A = TIMER1_TICKS_IN_SECOND;
    //setup timer prescaler
    TCCR1B |= _BV(TIMER1_PRESCALER) + _BV(WGM12);
    //enable the compare interrupt
    TIMSK1 = _BV(OCIE1A);
}
```

main()



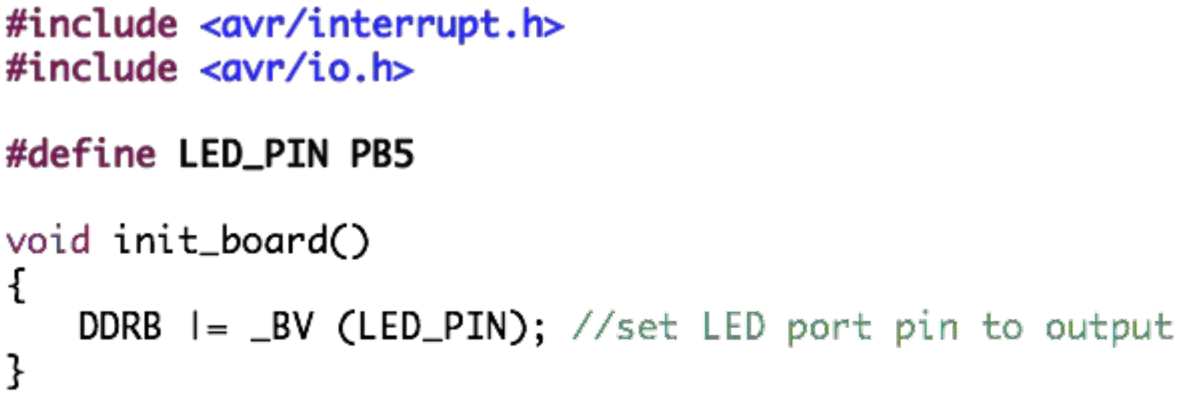
```
int main()
{
    init_board();

    init_timer();

    sei(); //set enable interrupts

    while(1) { //do nothing
    }
}
```

and, the interrupt



```
ISR(TIMER1_COMPA_vect)
{
    PORTB ^= _BV(LED_PIN);
}
```

Look at init_board() again

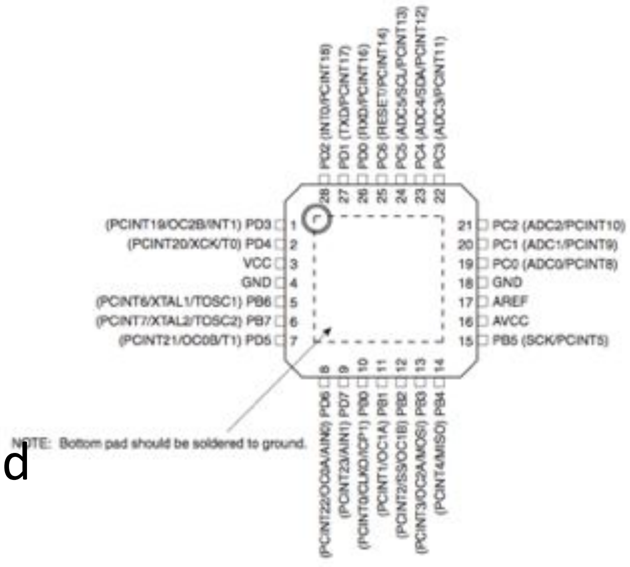


```
#include <avr/interrupt.h>
#include <avr/io.h>

#define LED_PIN PB5

void init_board()
{
    DDRB |= _BV (LED_PIN); //set LED port pin to output
}
```

What is this doing?
Blinking a led, but what does the line mean?

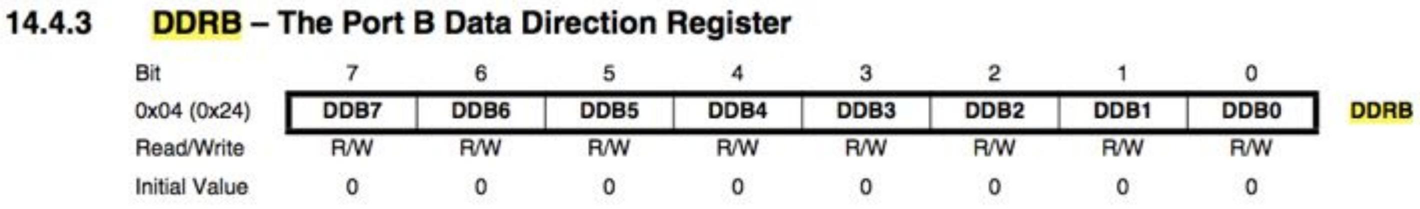
Concept of GPIO

- MCUs require to communicate with the outside world
- No MCU/CPU is an island
  - You need to attach other devices for it to do something useful
- GPIO is one mechanism of input or output
- In init_board() PB5 (Port B, Pin 5) is connected to the LED on the Arduino



GPIO Direction

- The direction register: specify direction of pins

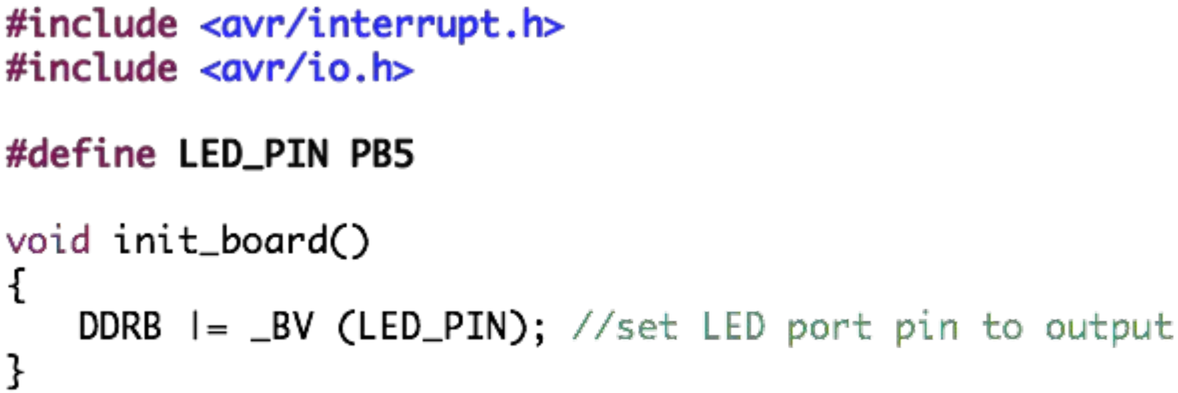


14.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

0 => input   1 => output

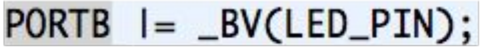
Look at init_board() again



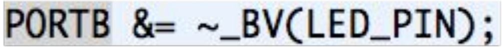
```
#include <avr/interrupt.h>
#include <avr/io.h>

#define LED_PIN PB5

void init_board()
{
    DDRB |= _BV (LED_PIN); //set LED port pin to output
}
```

LED_PIN (PB5)’s bit value will be set high (i.e. true, or 1)
in DDRB (portb’s data direction)
which means it will be an output pin.

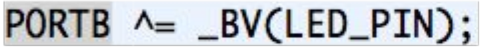
Outputting to a pin



```
PORTB  |= _BV(LED_PIN);
```



```
PORTB &= ~_BV(LED_PIN);
```



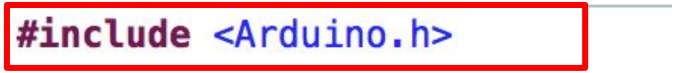
```
PORTB ^= _BV(LED_PIN);
```

Setting the value of a pin is done by setting the individual pins of the port;
or groups of pins at the same time;
or even all pins on the same port at the same time

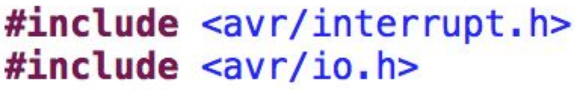
Serial Output?

- How can we output serial?
  - Use Arduino’s libraries
    - Using setup() and loop()
    - Use main, and initialise Arduino libraries
  - Use the USART (Universal Synchronous/Asynchronous Receiver/Transmitter) peripheral directly

Merging Arduino Library and Testing Serial

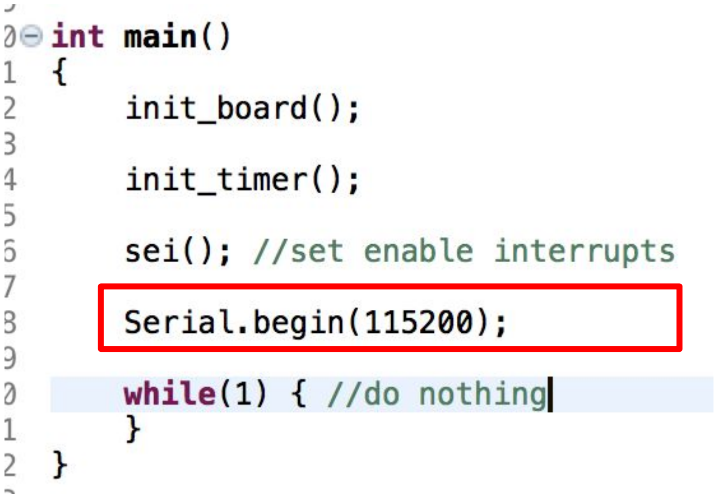


```
#include <Arduino.h>
```

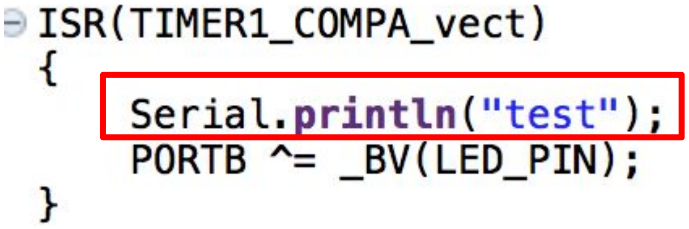


```
#include <avr/interrupt.h>
#include <avr/io.h>
```

...

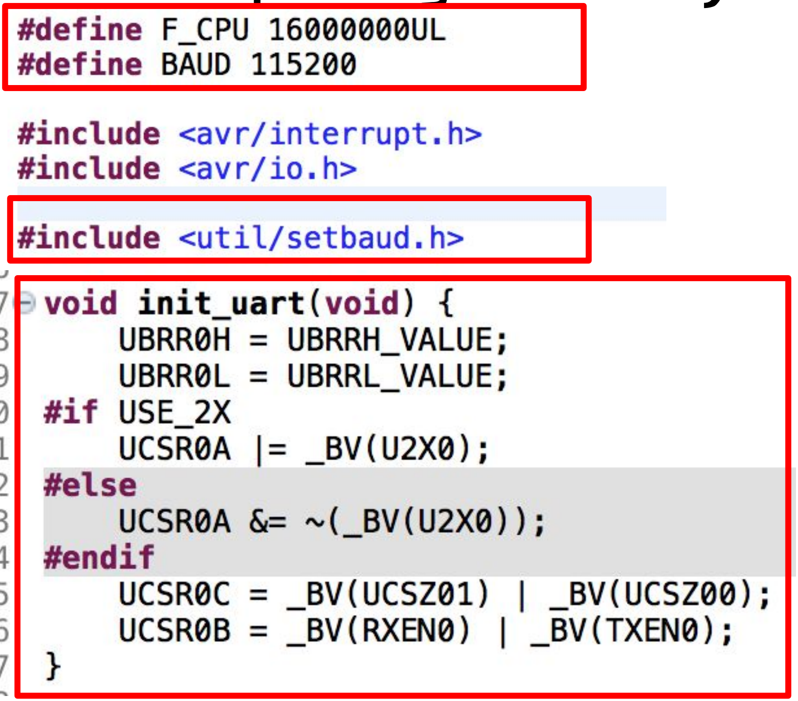


```
int main()
{
    init_board();

    init_timer();

    sei(); //set enable interrupts

    Serial.begin(115200);

    while(1) { //do nothing
    }
}
```

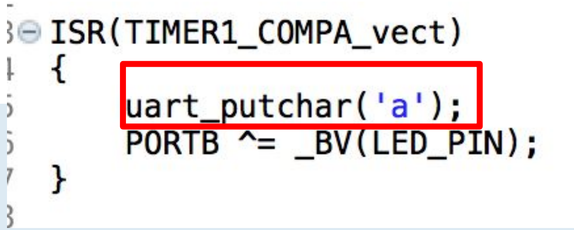


```
ISR(TIMER1_COMPA_vect)
{
    Serial.println("test");
    PORTB ^= _BV(LED_PIN);
}
```

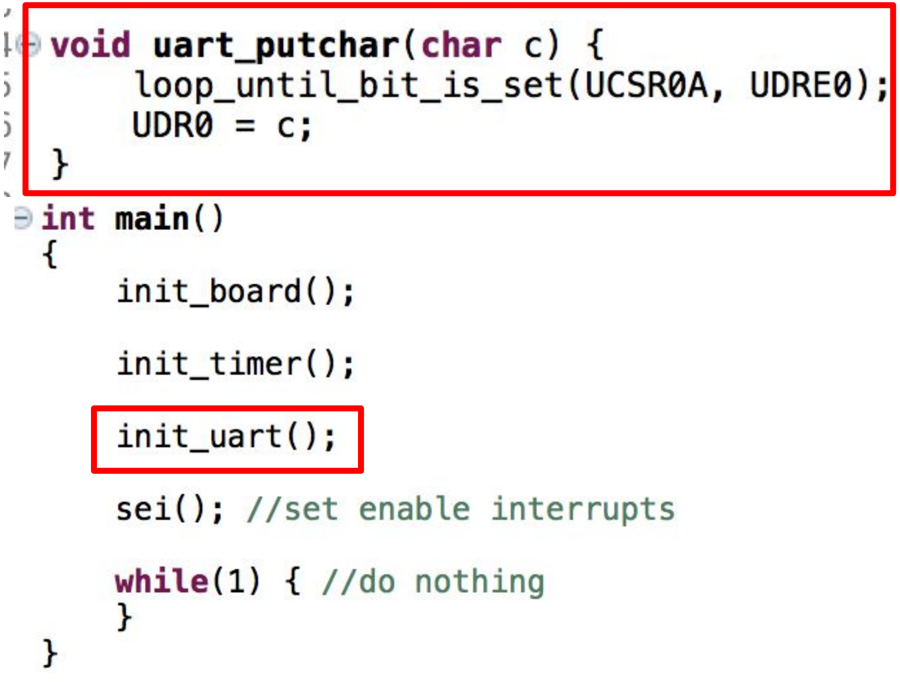
Interfacing Directly with the USART



```
#define F_CPU 16000000UL
#define BAUD 115200

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

#include <util/setbaud.h>

void init_uart(void) {
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRRL_VALUE;
#if USE_2X
    UCSR0A |= _BV(U2X0);
#else
    UCSR0A &= ~(_BV(U2X0));
#endif
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);
    UCSR0B = _BV(RXEN0) | _BV(TXEN0);
}
```



```
ISR(TIMER1_COMPA_vect)
{
    uart_putchar('a');
    PORTB ^= _BV(LED_PIN);
}
```



```
void uart_putchar(char c) {
    loop_until_bit_is_set(UCSR0A, UDRE0);
    UDR0 = c;
}
int main()
{
    init_board();

    init_timer();

    init_uart();

    sei(); //set enable interrupts

    while(1) { //do nothing
    }
}
```

https://appelsiini.net/2011/simple-usart-with-avr-libc/

To Do

1. Understand what the USART code is doing
    a. by looking up the registers and peripherals in the datasheet (UBRR0H etc)
        i. Try to see source of util/setbaud.h

    b. Search for and try to understand:
        i. F_CPU avr gcc
        ii. BAUD avr gcc
        iii. loop_until_bit_is_set

To Do

2. Implement: uart_println() function that takes in a string and send each character followed by **\n** (newline).