Download

Download or clone the repository at:

https://github.com/i-n-g-o/attiny_examples

AVR Programming with the ATtiny45

Becky Stewart and Ingo Randolf

What is a microcontroller?

- Processor, memory, some other useful stuff.

- Arduino is a microcontroller platform in that it contains a microcontroller and a bunch of other things to make interacting with that microcontroller easier.

What is on the Arduino Uno board?

- Microcontroller (ATmega328P) with bootloader

- Voltage regulator

- USB interface

- External oscillator
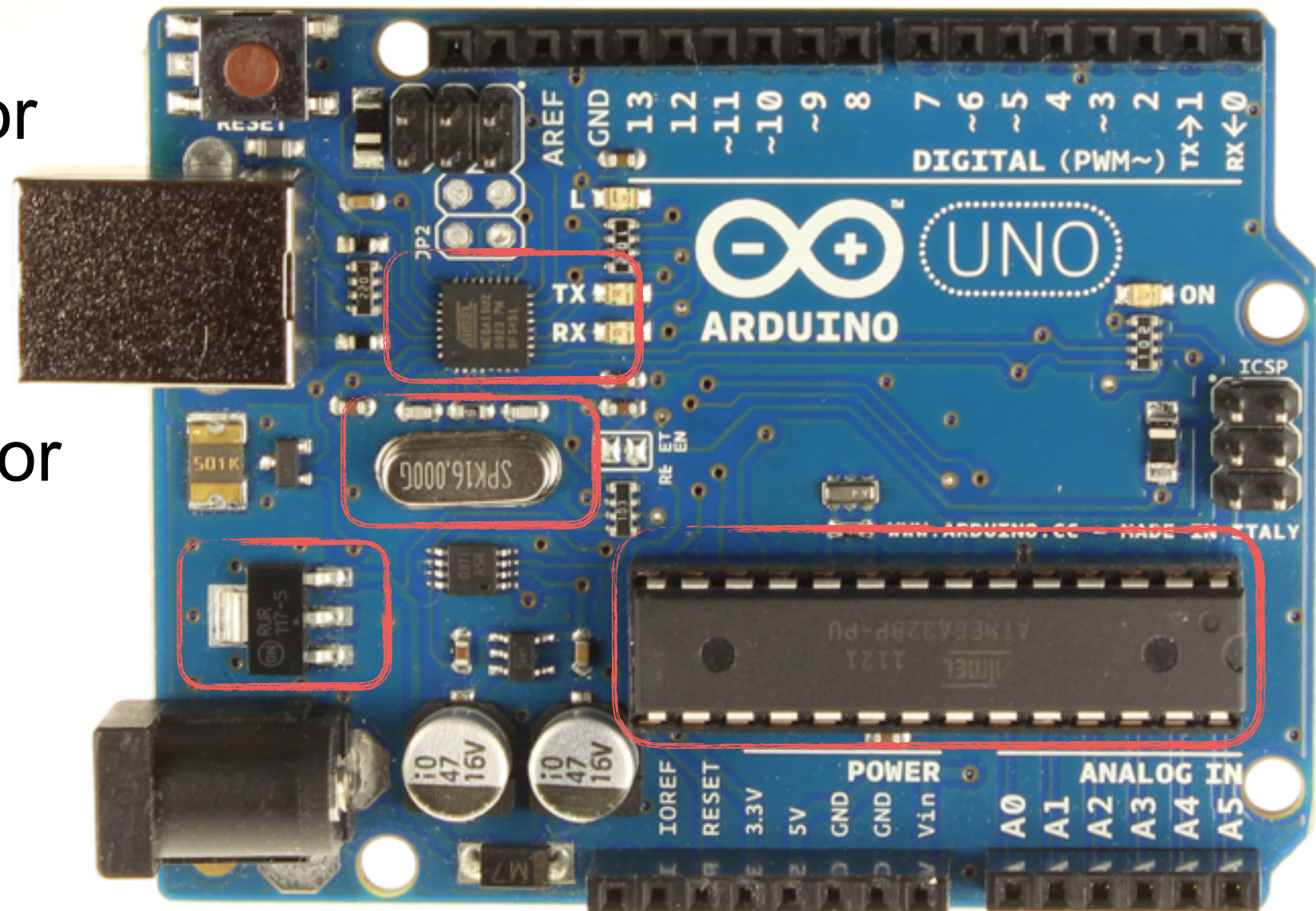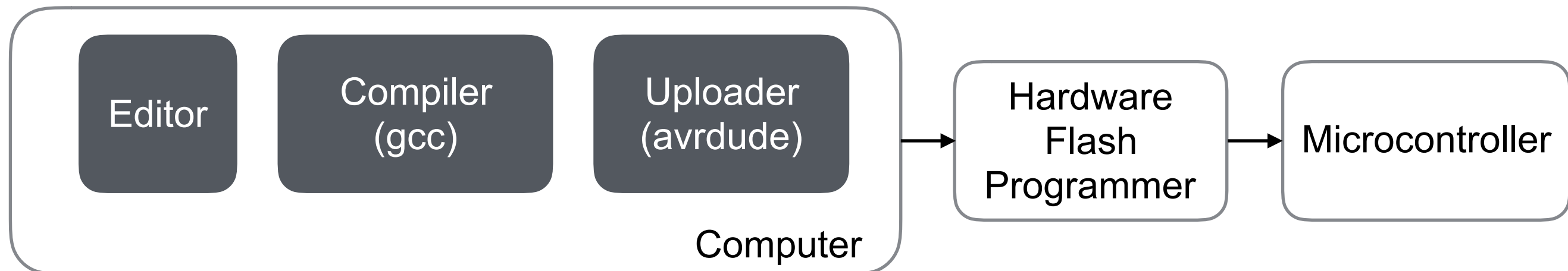
Why Not Just Use an Arduino?

- Lots of extra stuff on there you might not need

- Cost

- It's physically big and awkward

- Want more flexibility on pins and wiring

- Arduino IDE not great

- Arduino library can be slow

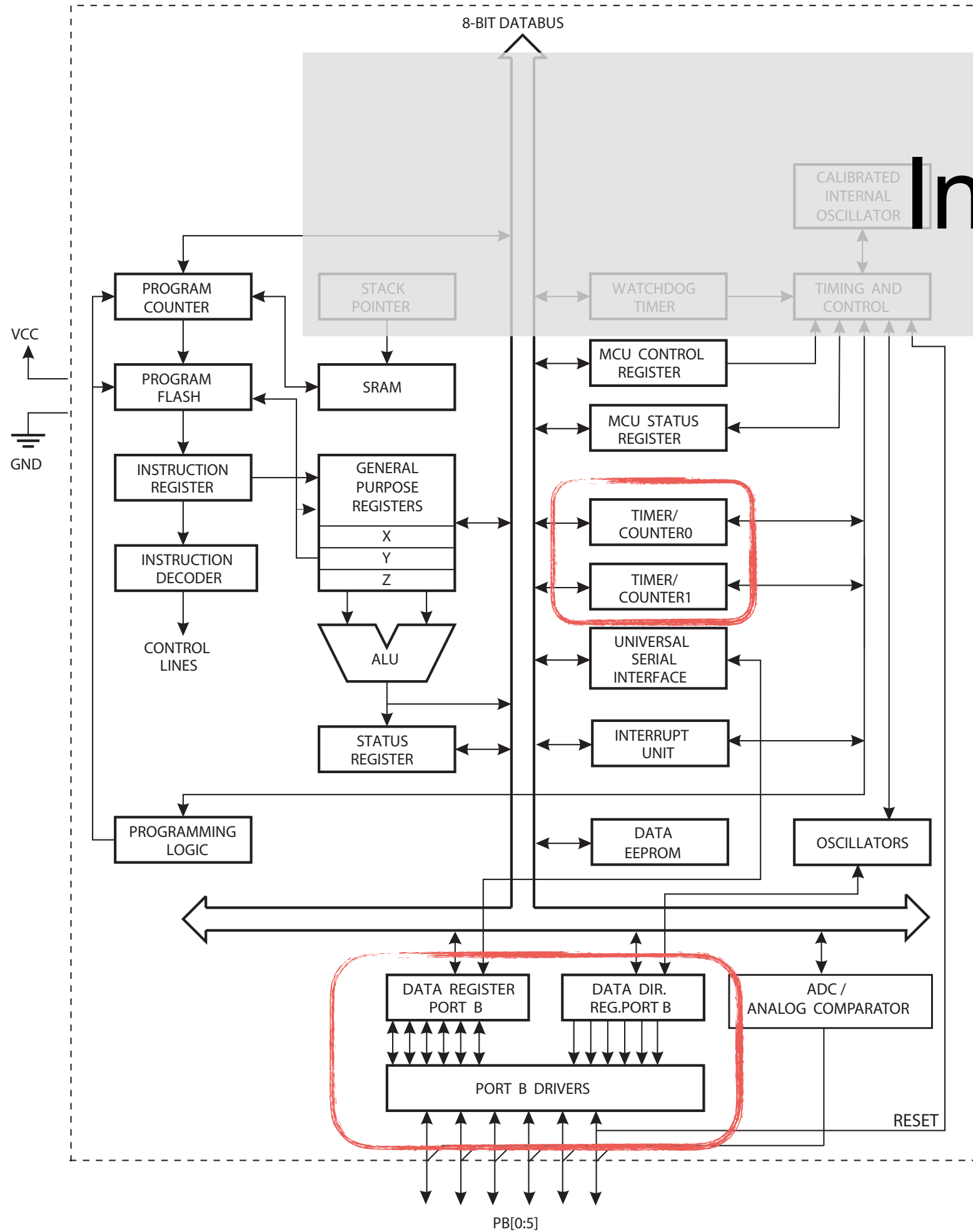- To understand what is actually going on

Toolchain

With Arduino

Arduino IDE → Arduino Board

Computer

Without Arduino

Editor | Compiler (gcc) | Uploader (avrdude) → Hardware Flash Programmer → Microcontroller

Computer

Inside the ATtiny

8-BIT DATABUS

VCC

GND

PROGRAM COUNTER

STACK POINTER

WATCHDOG TIMER

CALIBRATED INTERNAL OSCILLATOR

TIMING AND CONTROL

MCU CONTROL REGISTER

PROGRAM FLASH

SRAM

MCU STATUS REGISTER

INSTRUCTION REGISTER

GENERAL PURPOSE REGISTERS

X
Y
Z

TIMER/ COUNTER0

TIMER/ COUNTER1

INSTRUCTION DECODER

ALU

UNIVERSAL SERIAL INTERFACE

CONTROL LINES

STATUS REGISTER

INTERRUPT UNIT

PROGRAMMING LOGIC

DATA EEPROM

OSCILLATORS

DATA REGISTER PORT B

DATA DIR. REG.PORT B

ADC / ANALOG COMPARATOR

PORT B DRIVERS

RESET

1 Hardware Register

PB[0:5]

Outside the ATtiny

PDIP/SOIC/TSSOP

(PCINT5/$\overline{\text{RESET}}$/ADC0/dW) PB5 ▢ 1   8 ▢ VCC

(PCINT3/XTAL1/CLKI/$\overline{\text{OC1B}}$/ADC3) PB3 ▢ 2   7 ▢ PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)

(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 ▢ 3   6 ▢ PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)

GND ▢ 4   5 ▢ PB0 (MOSI/DI/SDA/AIN0/OC0A/$\overline{\text{OC1A}}$/AREF/PCINT0)

Outside the ATtiny

This marks which end is 'up', it may be a notch on the chip or a circle in the corner.

PDIP/SOIC/TSSOP

(PCINT5/$\overline{\text{RESET}}$/ADC0/dW) PB5	1	8	VCC
(PCINT3/XTAL1/CLKI/$\overline{\text{OC1B}}$/ADC3) PB3	2	7	PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4	3	6	PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
GND	4	5	PB0 (MOSI/DI/SDA/AIN0/OC0A/$\overline{\text{OC1A}}$/AREF/PCINT0)

Outside the ATtiny

PDIP/SOIC/TSSOP

(PCINT5/$\overline{\text{RESET}}$/ADC0/dW) PB5 ⊟ 1   8 ⊟ VCC

(PCINT3/XTAL1/CLKI/$\overline{\text{OC1B}}$/ADC3) PB3 ⊟ 2   7 ⊟ PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)

(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 ⊟ 3   6 ⊟ PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)

GND ⊟ 4   5 ⊟ PB0 (MOSI/DI/SDA/AIN0/OC0A/$\overline{\text{OC1A}}$/AREF/PCINT0)

1.8 - 5.5V
or
2.7 - 5.5V

Outside the ATtiny

PDIP/SOIC/TSSOP

(PCINT5/$\overline{\text{RESET}}$/ADC0/dW) PB5 ☐ 1   8 ☐ VCC

(PCINT3/XTAL1/CLKI/$\overline{\text{OC1B}}$/ADC3) PB3 ☐ 2   7 ☐ PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)

(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 ☐ 3   6 ☐ PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)

GND ☐ 4   5 ☐ PB0 (MOSI/DI/SDA/AIN0/OC0A/$\overline{\text{OC1A}}$/AREF/PCINT0)
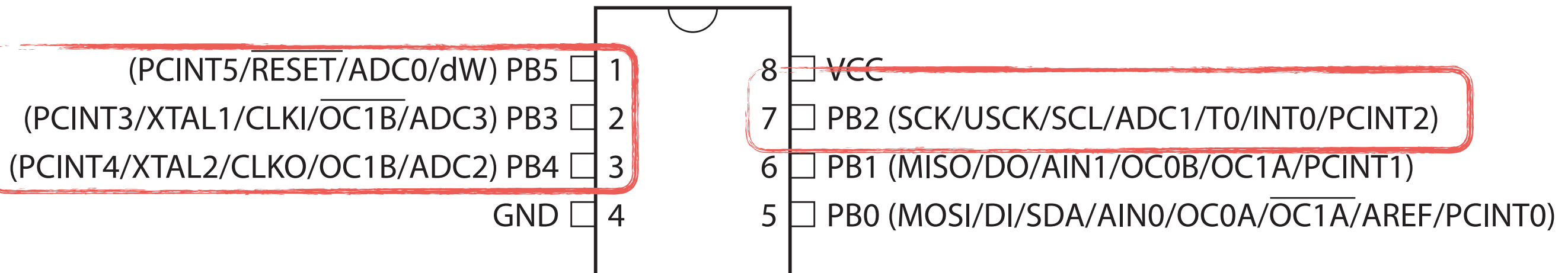
Analog Inputs

Outside the ATtiny

PDIP/SOIC/TSSOP

(PCINT5/RESET/ADC0/dW) PB5 ☐ 1   8 ☐ VCC
(PCINT3/XTAL1/CLKI/OC1B/ADC3) PB3 ☐ 2   7 ☐ PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 ☐ 3   6 ☐ PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
GND ☐ 4   5 ☐ PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
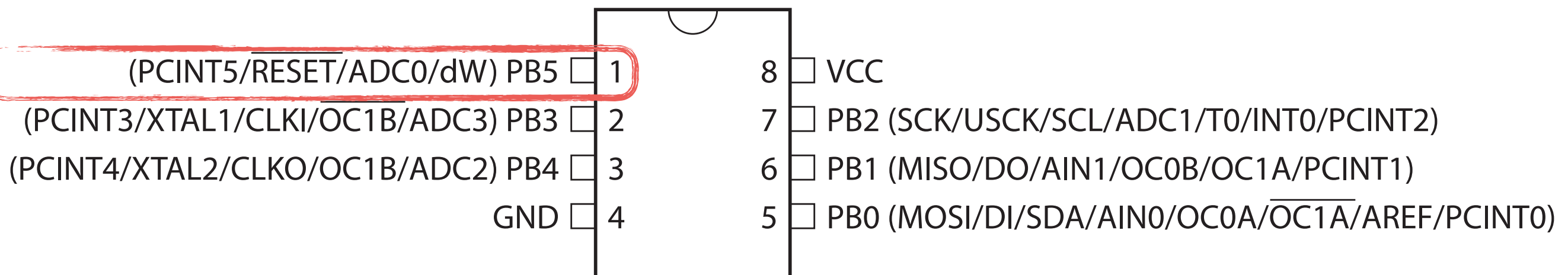
If RESET is held low for long enough, will reset the chip

Using `avrdude`

- Pro-tip: gently bend legs of ATtiny so fits cleanly into holder/breadboard

- In a terminal type:

```
> avrdude -p t45 -c usbtiny
```

Using `avrdude`

- Pro-tip: gently bend legs of ATtiny so fits cleanly into holder/breadboard

- In a terminal type:

```
> avrdude -p t45 -c usbtiny
```

ATtiny45   Using a USBTiny programmer

Blinking

- Download code if you haven't already

- In a terminal window, use `cd` to go to the directory

  `/etextile-summercamp/00-blink_with_delay`

- In the terminal type

  `> make`

- Then type

  `> make install`

Looking at the Source

```c
#include <avr/io.h>                          /* Defines pins, ports, etc */
#include <util/delay.h>                       /* Functions to waste time */


int main(void) {

  // -------- Inits --------- //
  DDRB = 0b0000001;                 // Data Direction Register B: writing a one to the bit
                                    //enables output.


  // ------ Event loop ------ //
  while (1) {

    PORTB = 0b00000001;           /* Turn on first LED bit/pin in PORTB */
    _delay_ms(1000);                                          /* wait */

    PORTB = 0b00000000;           /* Turn off all B pins, including LED */
    _delay_ms(1000);                                          /* wait */

  }                                              /* End event loop */
  return 0;                            /* This line is never reached */
}
```

Looking at the Source

```
#include <avr/io.h>            /* Defines pins, ports, etc */
#include <util/delay.h>         /* Functions to waste time */
```

same as Arduino

```
int main(void) {

  // -------- Inits --------- //
  DDRB = 0b0000001;           // Data Direction Register B: writing a one to the bit
                              //enables output.


  // ------ Event loop ------ //
  while (1) {

    PORTB = 0b00000001;        /* Turn on first LED bit/pin in PORTB */
    _delay_ms(1000);                                    /* wait */

    PORTB = 0b00000000;        /* Turn off all B pins, including LED */
    _delay_ms(1000);                                    /* wait */

  }                                          /* End event loop */
  return 0;                        /* This line is never reached */
}
```

Looking at the Source

```
#include <avr/io.h>              /* Defines pins, ports, etc */
#include <util/delay.h>          /* Functions to waste time */
```

same as Arduino

```
int main(void) {

  // --------- Inits --------- //
  DDRB = 0b0000001;             // Data Direction Register B: writing a one to the bit
                                //enables output.

  // ------ Event loop ------ //
  while (1) {

    PORTB = 0b00000001;         /* Turn on first LED bit/pin in PORTB */
    _delay_ms(1000);                                        /* wait */

    PORTB = 0b00000000;         /* Turn off all B pins, including LED */
    _delay_ms(1000);                                        /* wait */

  }                                          /* End event loop */
  return 0;                      /* This line is never reached */
}
```

like `setup()` and `pinMode()`

Looking at the Source

```
#include <avr/io.h>                 /* Defines pins, ports, etc */
#include <util/delay.h>              /* Functions to waste time */
```

same as Arduino

```
int main(void) {

  // -------- Inits --------- //
  DDRB = 0b0000001;                 // Data Direction Register B: writing a one to the bit
                                     //enables output.
```

like `setup()` and `pinMode()`

```
  // ------ Event loop ------ //
  while (1) {

    PORTB = 0b00000001;             /* Turn on first LED bit/pin in PORTB */
    _delay_ms(1000);                                          /* wait */

    PORTB = 0b00000000;             /* Turn off all B pins, including LED */
    _delay_ms(1000);                                          /* wait */

  }                                              /* End event loop */
  return 0;                         /* This line is never reached */
}
```

like `loop()`, `digitalWrite()` and `delay()`

Hardware Registers

- Registers are (tiny) physical switches on the chip and you turn them on and off by setting them to 1 or 0

- The ATtiny45 has one set of registers name B

  - Bigger microcontrollers like the ATmega have more like C and D

PDIP/SOIC/TSSOP

```
(PCINT5/RESET/ADC0/dW) PB5 □  1        8  □ VCC
(PCINT3/XTAL1/CLKI/OC1B/ADC3) PB3 □  2        7  □ PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 □  3        6  □ PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
                          GND □  4        5  □ PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
```
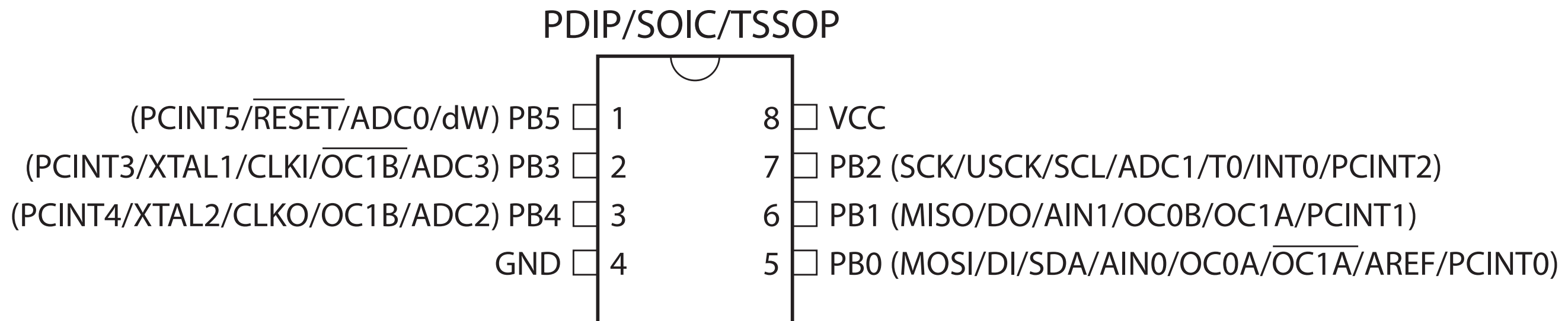
Hardware Registers

- Data direction registers are like `pinMode()`

  - `1` sets that pin to output, `0` to input

    DDRB =
0	0	0	0	1
PB5	PB3	PB2	PB1	PB0

- PORT is the register for setting outputs 0 for LOW and 1 for HIGH for each pin

    PORTB =
0	0	0	0	1
PB5	PB3	PB2	PB1	PB0

- PIN is the register for reading in inputs (not using this yet)

Binary and Hex Numbers

- Decimal (Base 10) - 0 1 2 3 4 5 6 7 8 9

```
x = 49;
```

- Binary (Base 2) - 0 1

```
x = 0b0101; // 5 in base 10
```

- Hexadecimal (Base 16) - 0 1 2 3 4 5 6 7 8 9 A B C D E F

```
x = 0xFF; // 255 in base 10, 11111111 in base 2
```

Change the delay time

Task

Change the LED pin

Build the LED circuit using a breadboard

Reading Input

- In a terminal window, use `cd` to go to the directory

  `/etextile-summercamp/01-button_read`

- On a breadboard connect a switch to PB1 and ground.

- In the terminal type

  `> make`

- Then type

  `> make install`

```
PB5  □ 1        8 □ VCC
PB3  □ 2        7 □ PB2
PB4  □ 3        6 □ PB1
GND  □ 4        5 □ PB0
```
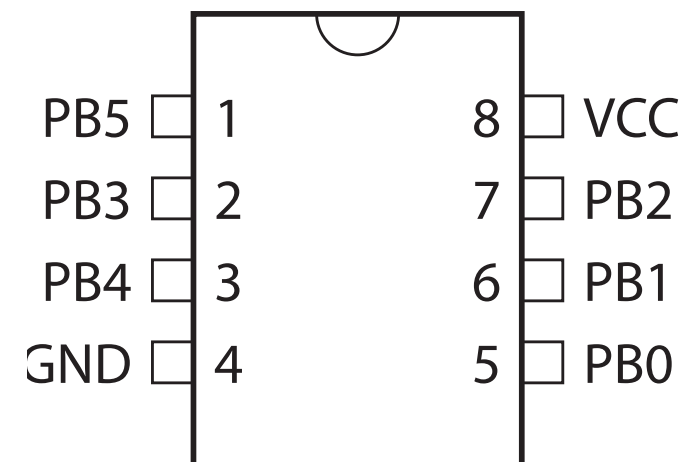
Reading Input

- The PINB holds the values of the input on each pin, whether HIGH or LOW

- Setting the PORTB HIGH for an input pin turns on the pull-up resistor

Bitmasks

OR

A	B	out
0	0	0
0	1	1
1	0	1
1	1	1

AND

A	B	out
0	0	0
0	1	0
1	0	0
1	1	1

XOR

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Bitmasks

OR

A	B	out
0	0	0
0	1	1
1	0	1
1	1	1

AND

A	B	out
0	0	0
0	1	0
1	0	0
1	1	1

XOR

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

```
1 << 2 = 00000100

00000100 & 00010100 = 00000100     want to clear all bits and set
                                    the mask

00000100 | 00010100 = 00010100     want to leave all other bits
                                    alone and set the mask

00000100 ^ 00010100 = 00010000     want to leave all other bits
                                    alone and toggle the mask
```

Blinking (Reprise)

- We are forcing the microcontroller to sit and do nothing while it waits an amount of time.

- Instead, we can set an internal alarm clock so we can do other things and it will just tell us when time is up.

- Called the Timer Compare Unit.

Timers in the ATtiny
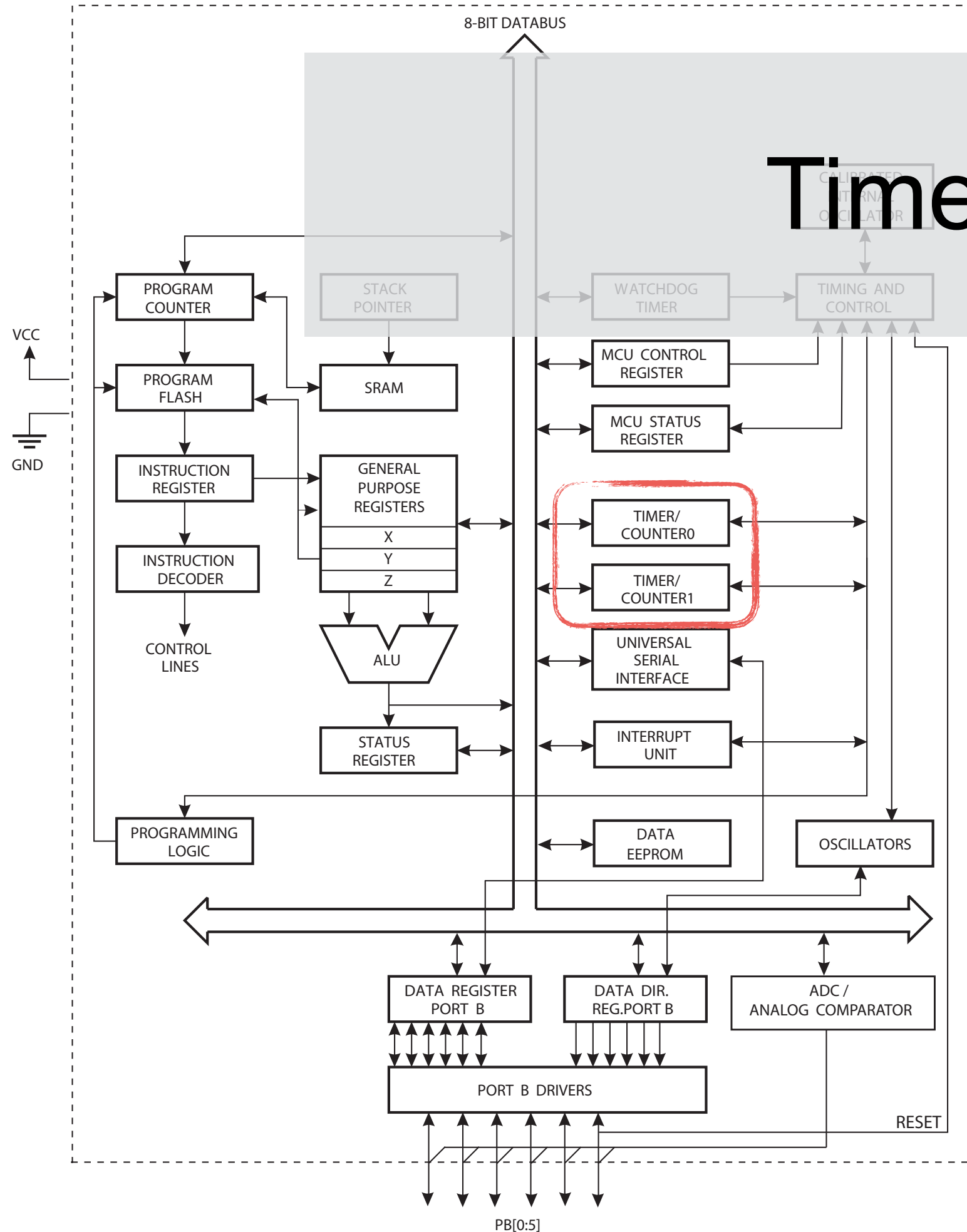
2 Timers/Counters

Blinking (Reprise)

- In a terminal window, use `cd` to go to the directory

  `/etextile-summercamp/02-blink_with_timer`

- In the terminal type

  `> make`

- Then type

  `>make install`

Clear Timer on Compare (CTC)
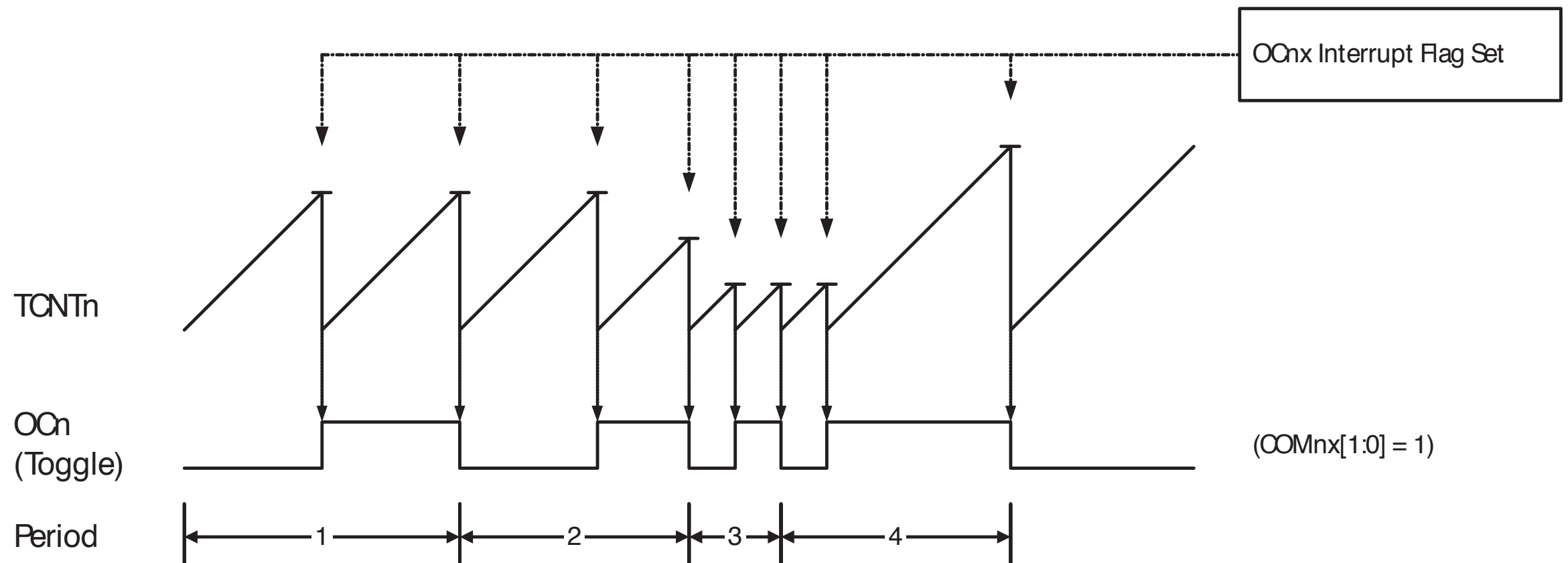
Figure 11-7 on page 72 in datasheet
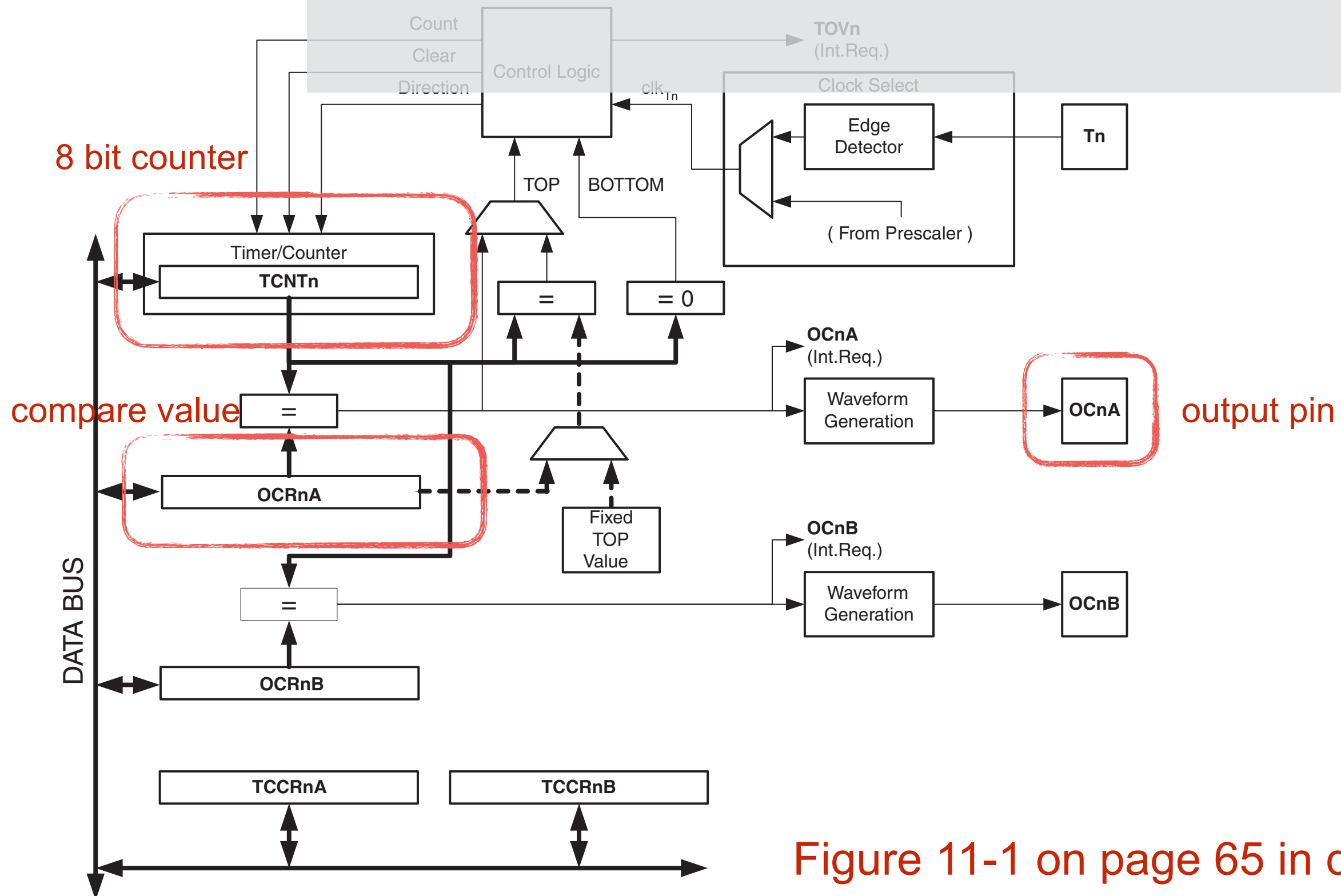
8 bit Timer/Counter

Figure 11-1 on page 65 in datasheet

(A Portion of the) Atmel AVR Family

ATMega

- ATMEGA168

- ATMEGA328P

- ATMEGA32U4

ATtiny

- ATTINY45-20PU

- ATTINY85V-20SU

(A Portion of the) Atmel AVR Family

ATMega

- ATMEGA168
- ATMEGA328P
- ATMEGA32U4

ATtiny

- ATTINY45-20PU
- ATTINY85V-20SU

Name of product family

(A Portion of the) Atmel AVR Family

ATMega

- ATMEGA168

- ATMEGA328P

- ATMEGA32U4

Size of Memory in KB

ATtiny

- ATTINY45-20PU

- ATTINY85V-20SU

(A Portion of the) Atmel AVR Family

ATMega

- ATMEGA168
- ATMEGA328P
- ATMEGA32U4

Atmel picoPower (low power consumption)

USB controller

ATtiny

- ATTINY45-20PU
- ATTINY85V-20SU

Max clock speed

(A Portion of the) Atmel AVR Family

ATMega

- ATMEGA168

- ATMEGA328P

- ATMEGA32U4

ATtiny

Package

- ATTINY45-20PU

- ATTINY85V-20SU

Can run at 1.8V instead of 2.7V

Resources

- Make: AVR Programming by Elliot Williams

  - Example code: https://github.com/hexagon5un/AVR-Programming

- http://www.atmel.com/images/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf

- http://www.avrfreaks.net/

- http://www.ladyada.net/learn/avr/index.html