8.11 More on Timers & a peek at ECE-304/JDP

Prof. David McLaughlin

ECE Dept

UMass Amherst

Spring 2024

FInal Exam:

Time/Date/Place TBD

Topics covered: Lectures 8.1 – 8.11, Assignments 8.1 – 8.3

Online Moodle Exam. You may take the exam remotely, at a location of your choice, provided you keep your camera on throughout the exam. If you don't have a working camera, or you don't have sufficient bandwith, take the exam in Elab2, Room 119.

If you have additional time accommodations, email me: dmclaugh@umass.edu

A set of review problems will be posted on Tuesday 5/6/25 along with a set of solutions (2 separate documents)

Wednesday we will hold an optional class via Zoom to discuss any aspect of the 8-bit part of the course.

Review Timer Discussion:

Configure Timer1

- ”normal mode”
- divide system clock by 64 (pre-scaler)

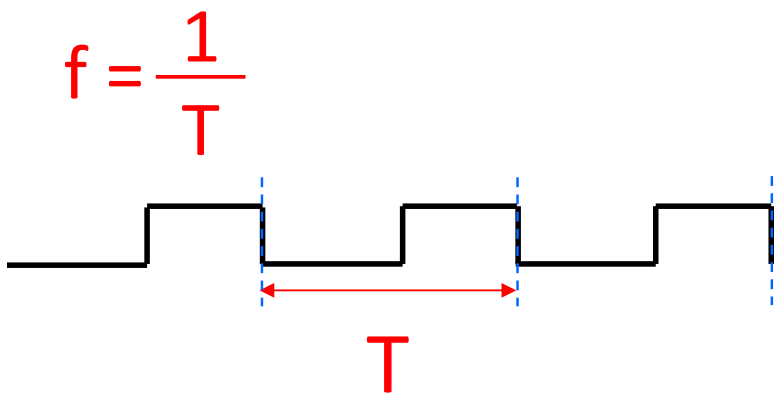
```
// Configure the timer
TCCR1A = 0;          // Normal mode (count up)
TCCR1B = 3;          // Divide clock by 64
```

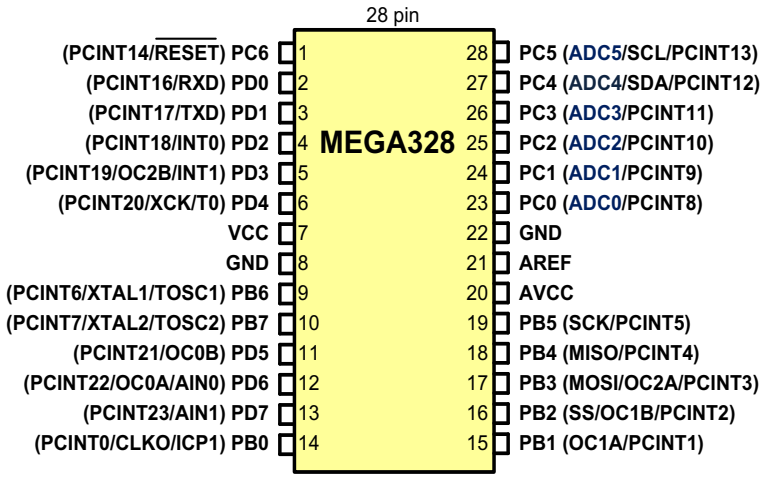
Timer delay loop

- pre-load value into timer
- timer counts from pre-load value to TOP
- 16 bit timer, so TOP = 2^{16}-1 = 65,536
- wait for overflow to occur
- clear overflow flag (a quirky feature of AVR mcus)
- toggle output pin (we happened to use PD6)

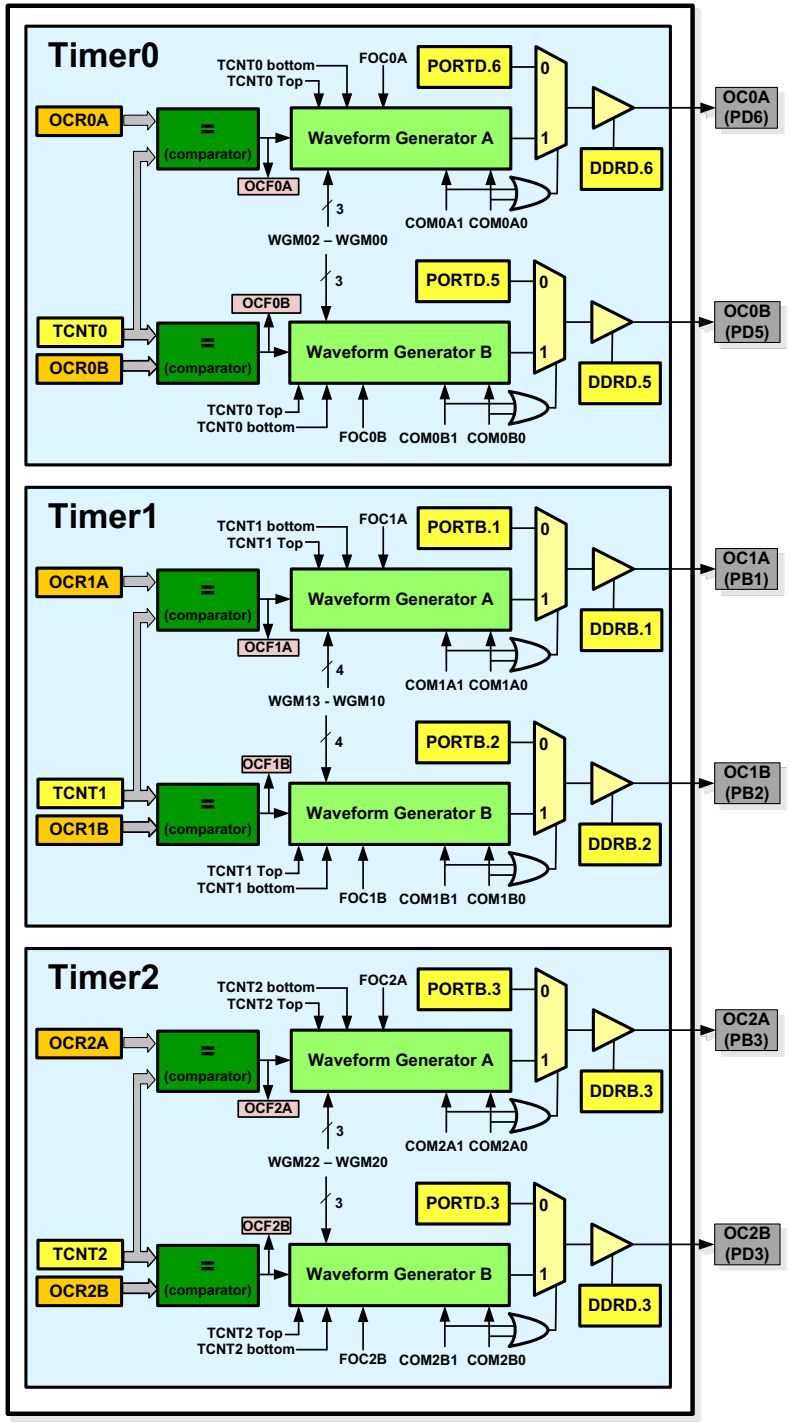
```
// Counter delay
TCNT1=65536-65536/pitch;
while ((TIFR1 & (1<<TOV1))==0);
TIFR1 |= 1<<TOV1;

PORTD ^= 1<<PORTD6;
```

LED, speaker, oscilloscope on PD6



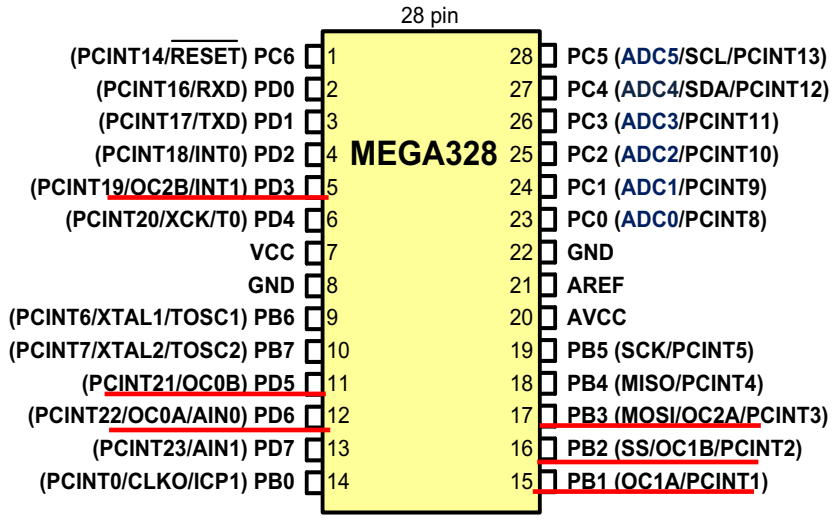


Waveform Generators and the Output Compare Mode

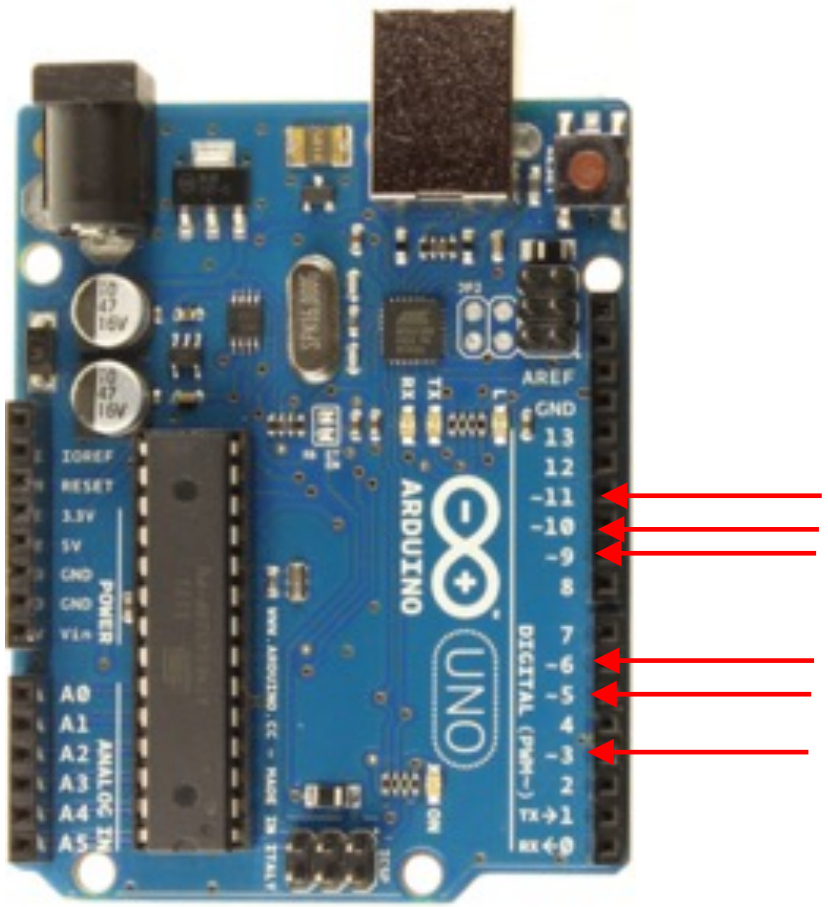


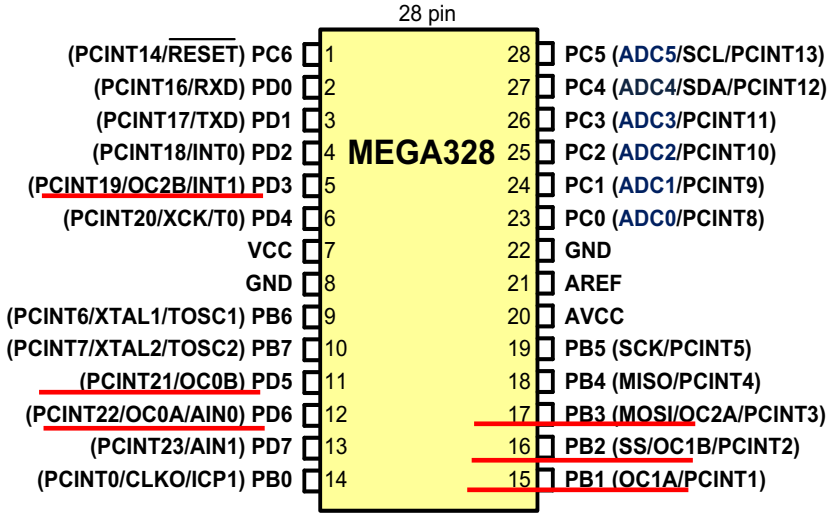
Each timer/counter has 2 waveform generators

6 waveform outputs share 6 GPIO pins:
Timer0: OC0A/PD6 & OC0B/PD5
Timer1: OC1A/PB1 & OC1B/PB2
Timer2: OC2A/PB3 & OC2B/PD3

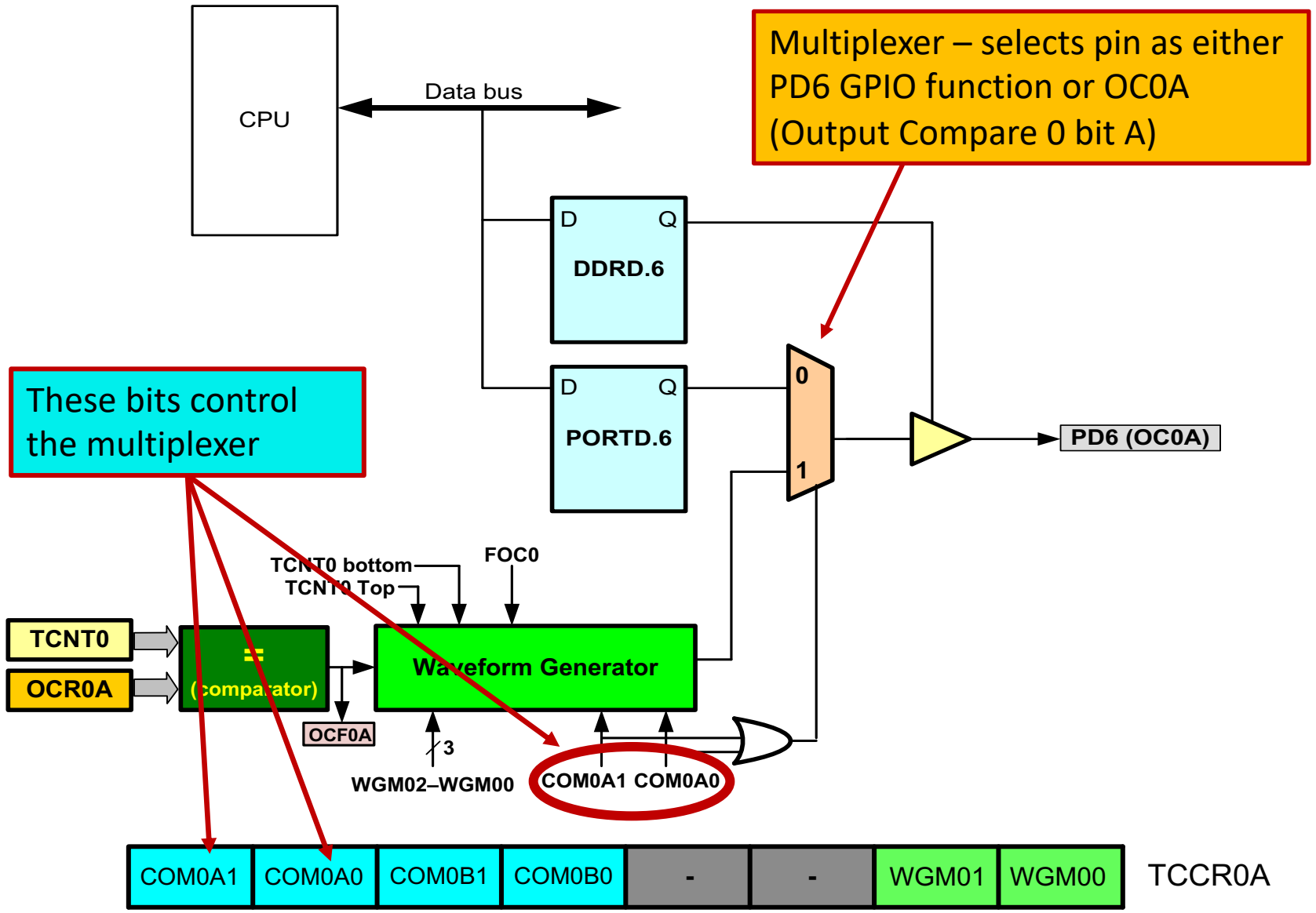


FYI: these 6 pins correspond to Arduino Uno pins 3, 5, 6, 9, 10, 11 (~PWM pins)

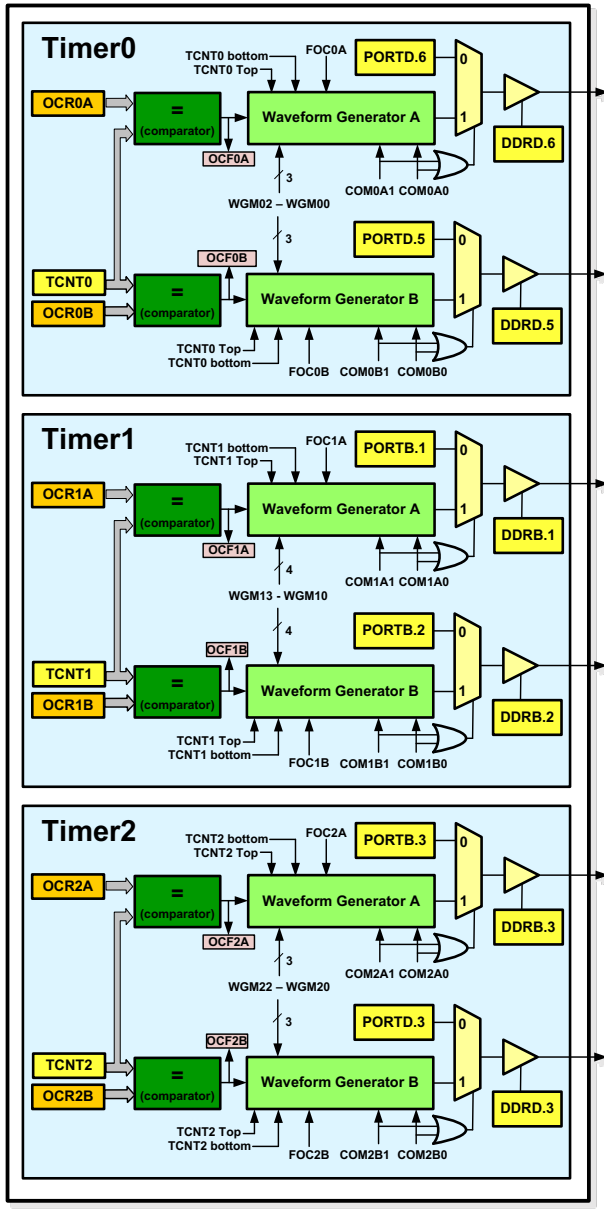




Waveform Generator 0, output A



Circuitry from previous slide is repeated for all 6 waveform generators...



Timer0 Output Comparator 0 Waveform A (OC0A)

Timer0 Output Comparator 0 Waveform B (OC0B)

Timer1 Output Comparator 1 Waveform A (OC1A)

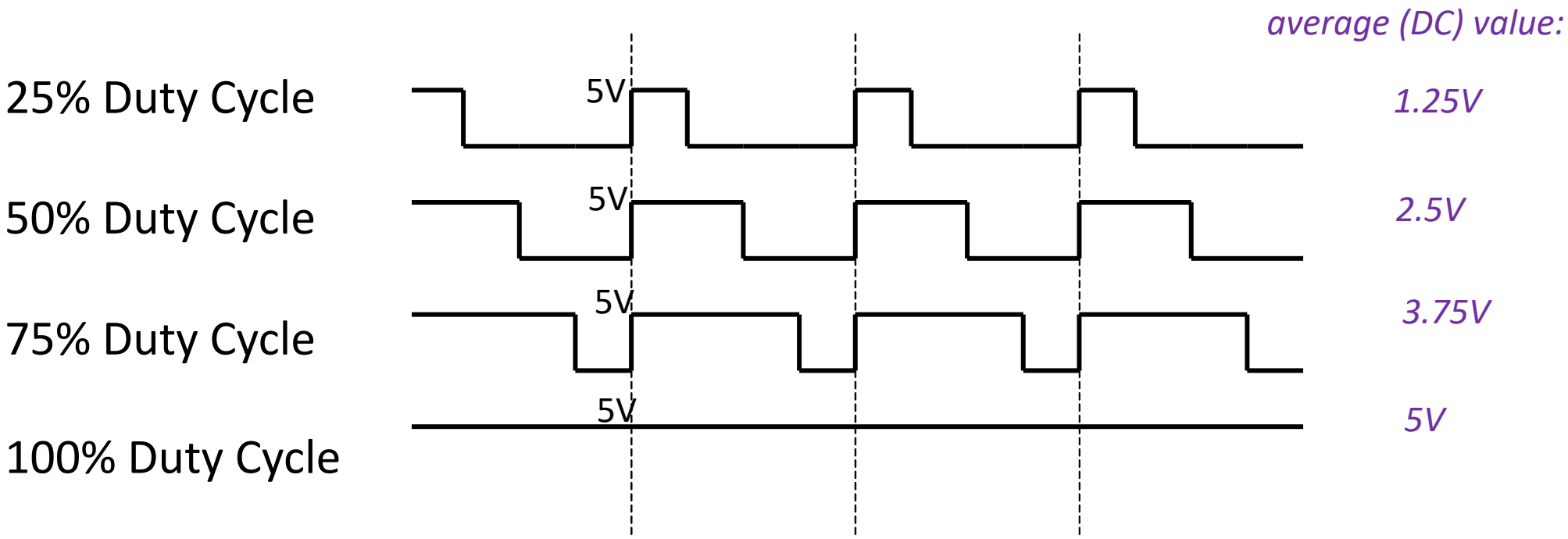
Timer1 Output Comparator 1 Waveform B (OC1B)

Timer2 Output Comparator 2 Waveform A (OC2A)

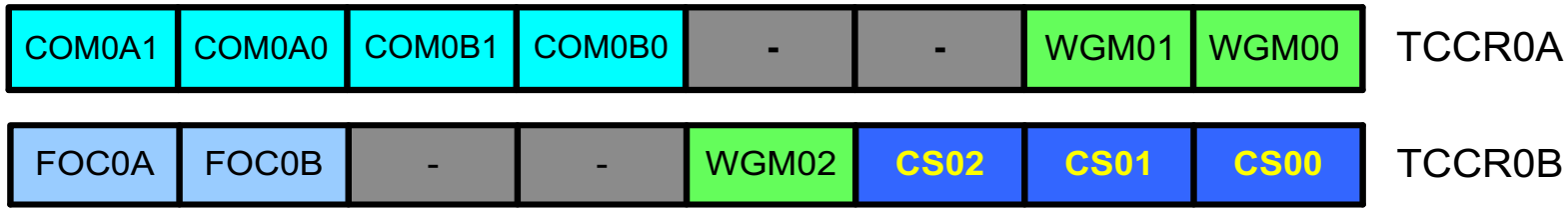
Timer2 Output Comparator 2 Waveform B (OC2B)

We will use the waveform generators for Pulse Width Modulation (variable Duty Cycle)

Can be used to dim LEDs, vary motor speed, & other applications



Fast PWM Mode
(PWM = pulse width modulation)



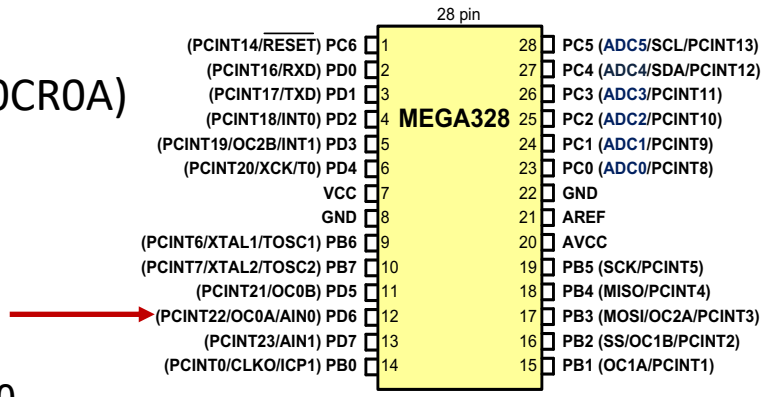
COM0A1:COM0A0 = 10 (clear OC0A on compare match; set on top)
WGM02:WGM00 = 011 (fast PWM mode)
CS02:CS00 = 011 (64 pre-scaler)

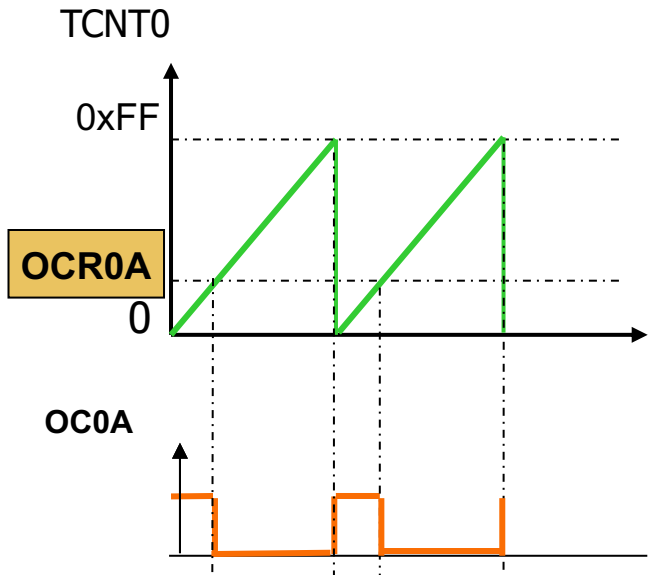
TCCR0A = 10000011=0x83

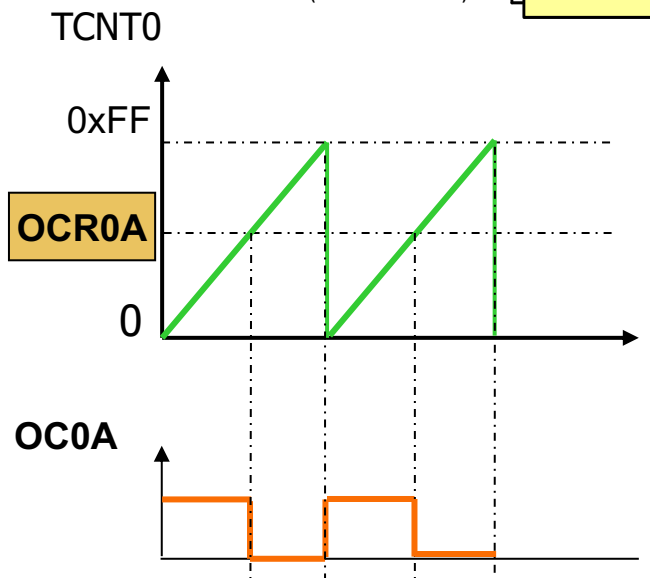
TCCR0B = 00000011 = 0x03

Fast PWM Mode
(Timer0, output OC0A)

- configure timer0 for Fast PWM Mode
- load value into OCR0A (Output Compare Register 0CR0A)
- timer continuously counts from 0x00 to 0xFF
- Output pin OC0A/PD6:
  - set when TCNT0 = 0xFF (top)
  - cleared when TCNT0=OCR0A





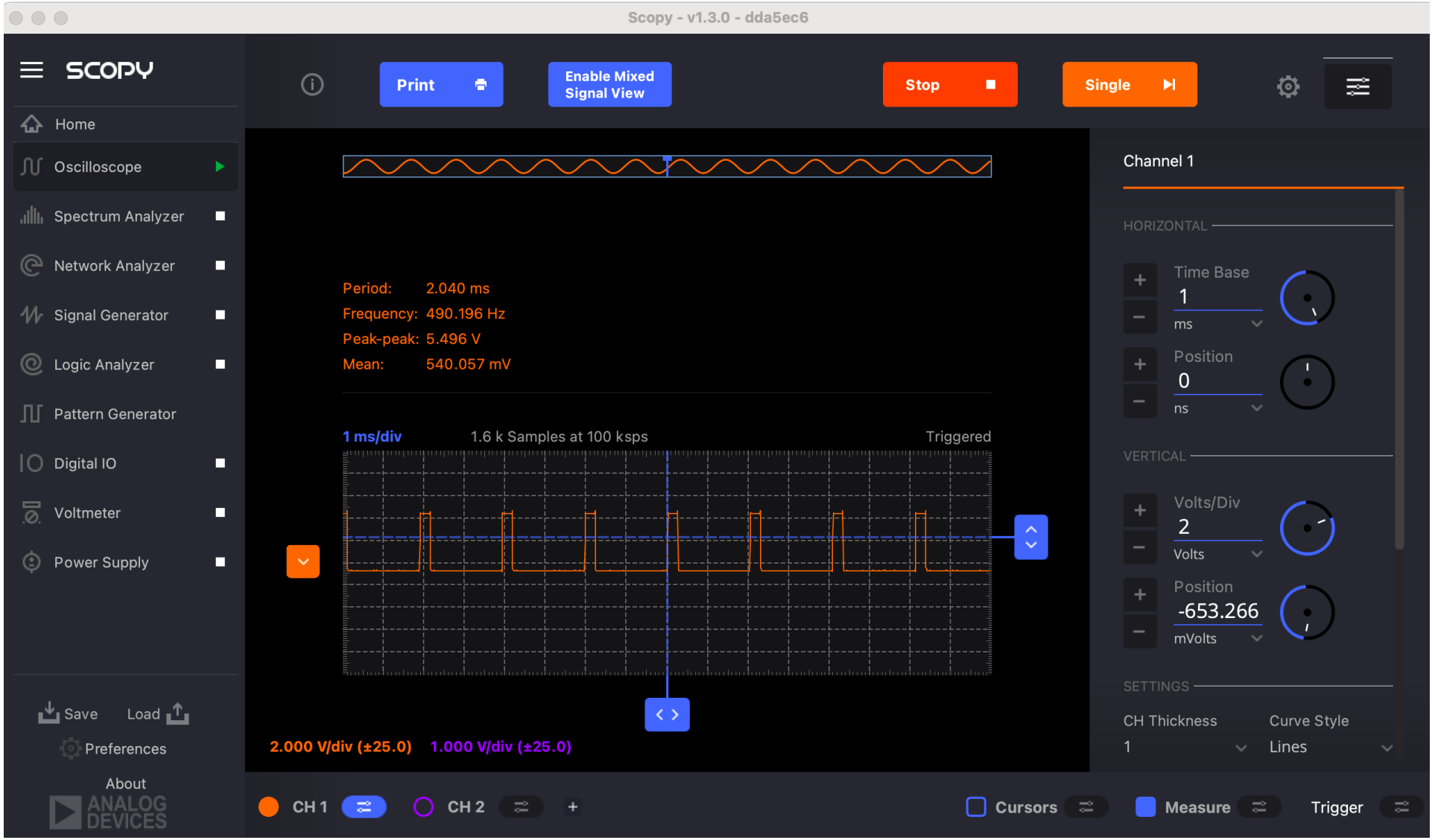


Increase & decrease duty cycle by increasing & decreasing OCR0A
OCR0A=0 → 0% duty cycle; OCR0A = 255 → 100% duty cycle

C fastpwm.c > main(void)

```
1   /* fastpwm.c  Creates 12.5%  duty cycle wafeform
2   on OC0A (PB6). Uses timer0 waveform generator A
3   with /64 pre-scaler.
4
5
6   D. McLaughlin 4/6/22 for ECE-231 Demo Spring 2022 */
7
8   #include <avr/io.h>
9   #include <util/delay.h>
10
11  int main(void){
12      DDRD = 1<<DDD6;     // Output must be PD6 for this code
13
14      // Initialize timer0 for FASTPWM, /64 prescaler
15      TCCR0A = 0x81;
16      TCCR0B = 0x03;
17
18      // Load the value into OCR0A:
19      OCR0A = 32;         // 32 is 12.5% of 256, so 12.5% duty cycle
20
21      while(1);           // Just sit there and do nothing
22  }
23
24
```

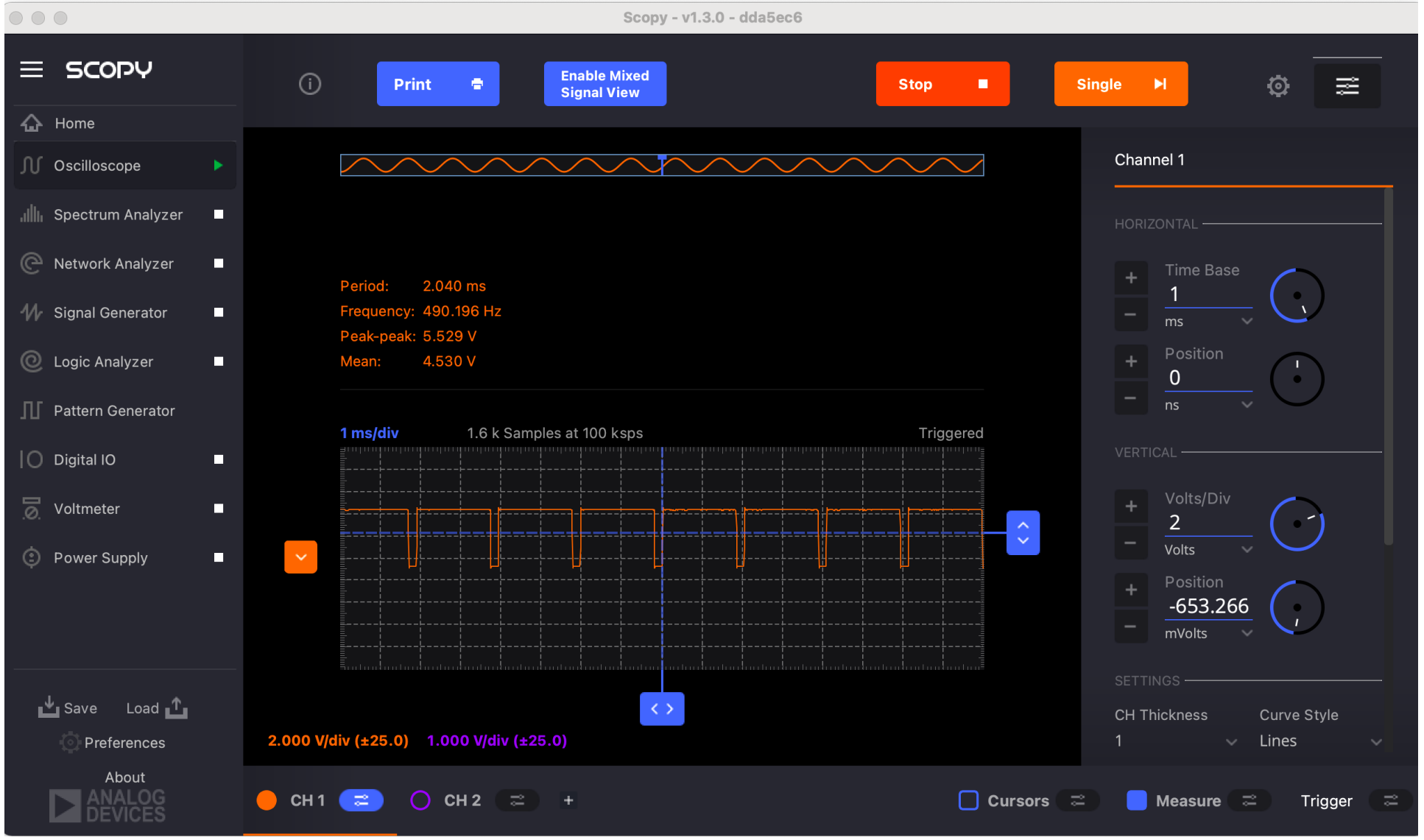
490 Hz pulse train, 12% duty cycle



fastpwm.c > main(void)

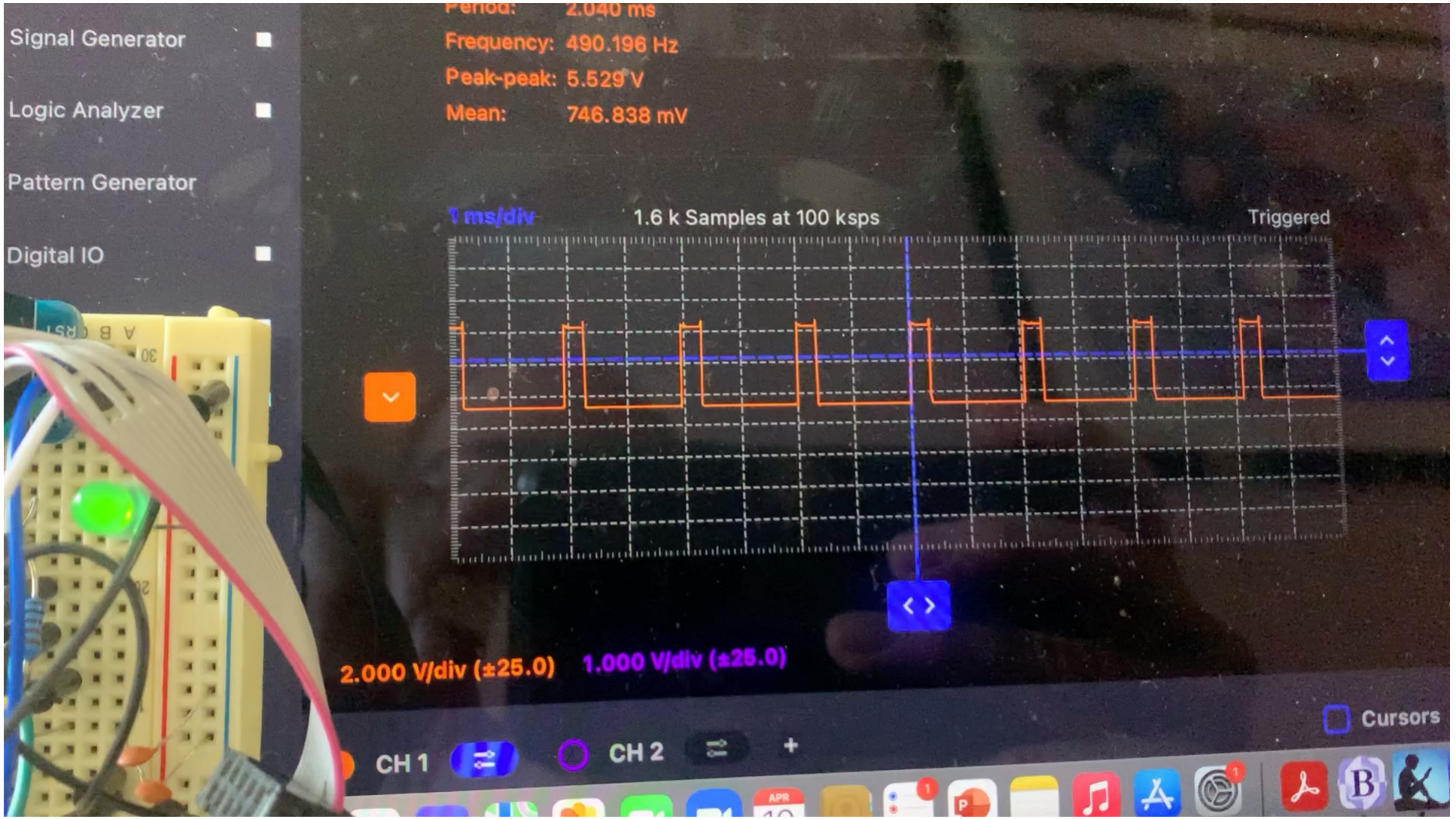
```c
/* fastpwm.c  Creates 90%  duty cycle wafeform
on OC0A (PB6). Uses timer0 waveform generator A
with /64 pre-scaler.
Waveform frequency: 16MHz/64/256/2 = 488 Hz
Duty cycle: 90% corresponds to OCR0A = 90% of 256 = 230
D. McLaughlin 4/6/22 for ECE-231 Demo Spring 2022 */

#include <avr/io.h>
#include <util/delay.h>

int main(void){
    DDRD = 1<<DDD6;     // Output must be PD6 for this code

    // Initialize timer0 for FASTPWM, /64 prescaler
    TCCR0A = 0x81;
    TCCR0B = 0x03;

    // Load the value into OCR0A:
    OCR0A = 230;     // 230 is 90% of 256 so 90% duty cycle

    while(1);           // Just sit there and do nothing
}
```

490 Hz pulse train, 90% duty cycle



C fastpwmramp.c > WAIT

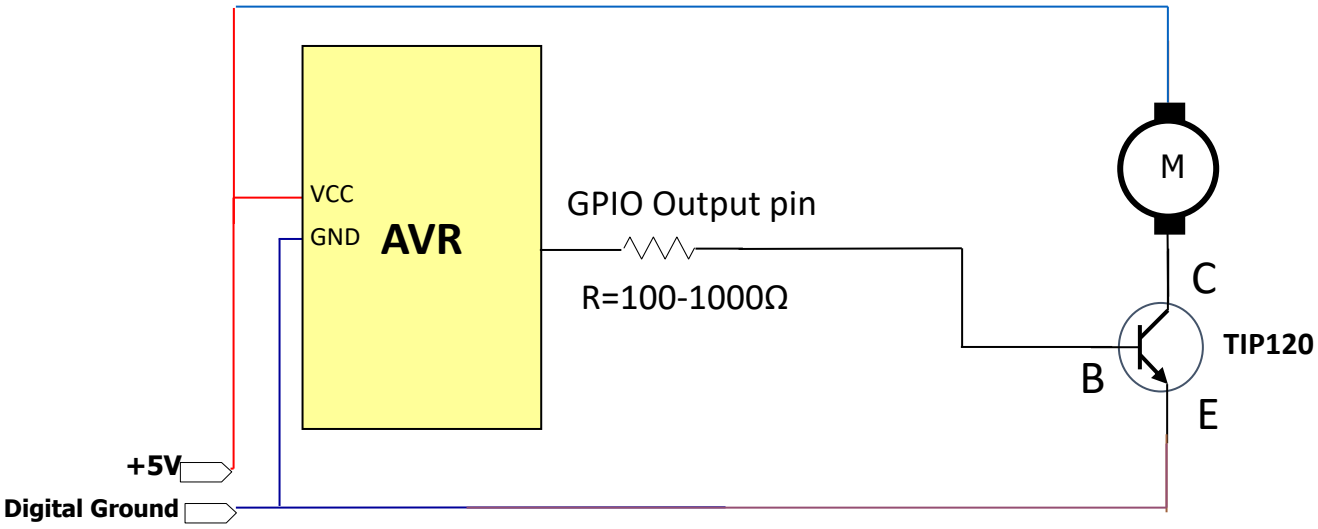
```c
/* fastpwmramp.c  Ramps up duty cycle on OC0A (PB6)
between 0 and 100%. Uses timer0 waveform generator
with /64 pre-scaler
D. McLaughlin 4/6/22 for ECE-231 Demo Spring 2022 */

#include <avr/io.h>
#include <util/delay.h>

#define WAIT 250
int main(void){
    unsigned char i=0;
    DDRD = 1<<DDD6;     // Output must be PD6 for this code

    // Initialize timer0 for FASTPWM, /64 prescaler
    TCCR0A = 0x81;
    TCCR0B = 0x03;


    while(1){
        i=i+10;
        OCR0A=i;        // pulse turns off when TCNT1 = OCR0A
        _delay_ms(WAIT);
    }
}
```

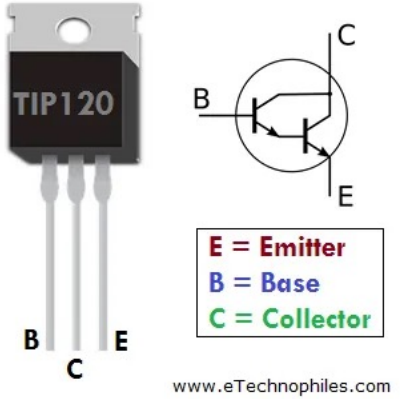

Signal Generator
Logic Analyzer
Pattern Generator
Digital IO
Period: 2.040 ms
Frequency: 490.196 Hz
Peak-peak: 5.529 V
Mean: 746.838 mV
1 ms/div
1.6 k Samples at 100 ksps
Triggered
2.000 V/div (±25.0)
1.000 V/div (±25.0)
Cursors
CH 1
CH 2

Drive a motor using a GPIO pin

GPIO pins can only source up to 40 mA
Motors draw hundreds of mA

Darlington transistor serves as motor driver: when GPIO pin is high, the transistor is on, making a connection between C & E. Motor draws its current directly from the +5V power supply rather than the GPIO pin



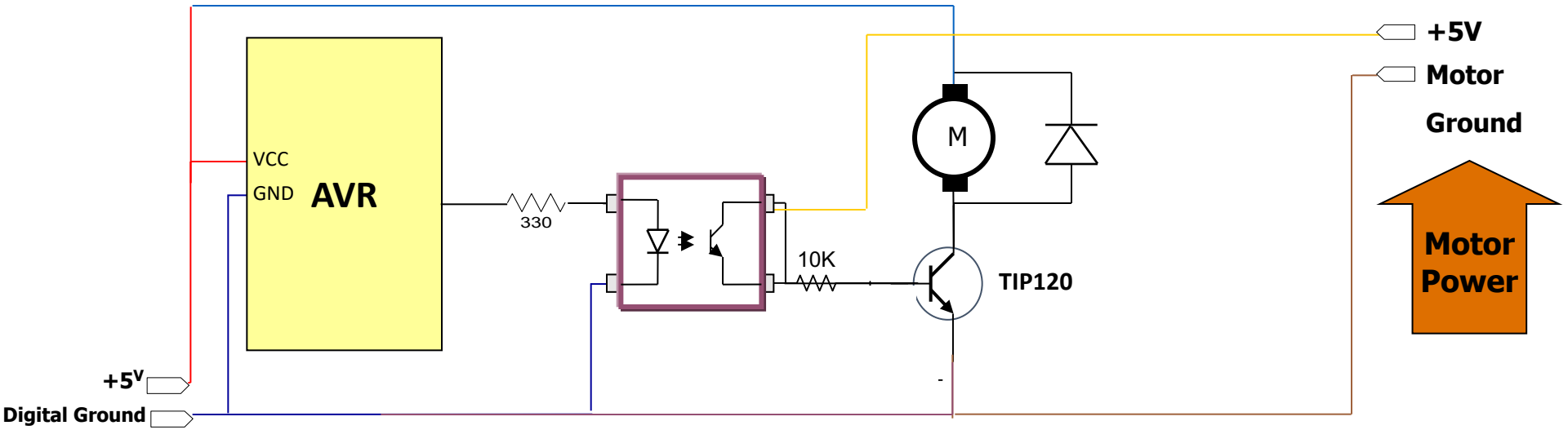


- The meter is showing the DC voltage (average value) of the waveform. The motor speed is proportional to DC voltage.
- Separate bench-top power supply for the motor.
- This experiment reset my 328p, ADALM2000 scope, and USB port repeatedly due to electrical noise from the motor.

- Motor connected directly to my bench-top variable DC power supply.
- The meter is showing the power supply voltage
- Note the impact of the motor on the power supply voltage (spikes, fluctuations)
- Adding a capacitor across the power supply (VCC & GND) suppresses some of the noise.

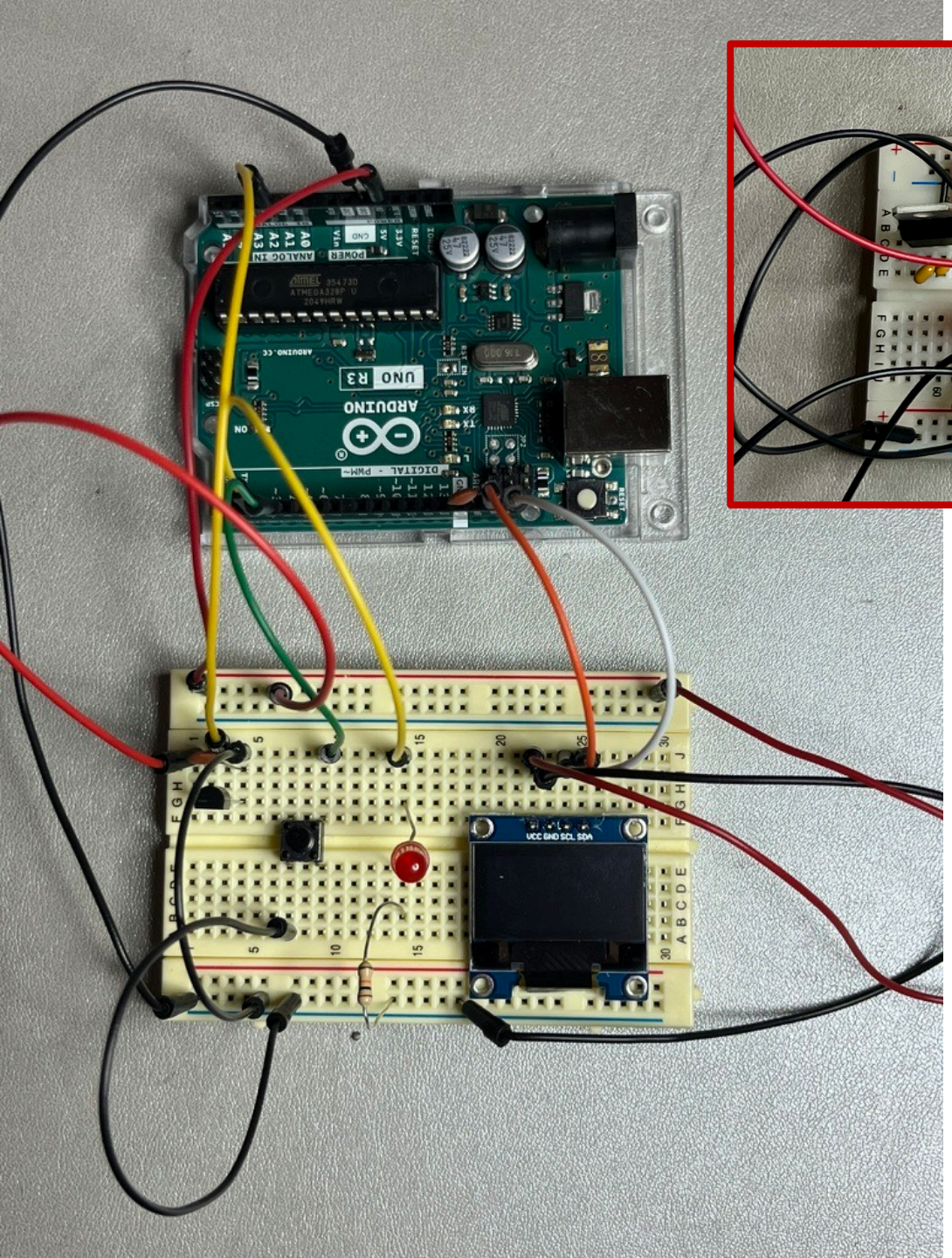
Opto-isolator, fly-back diode, separate power supplies for motor & 328P

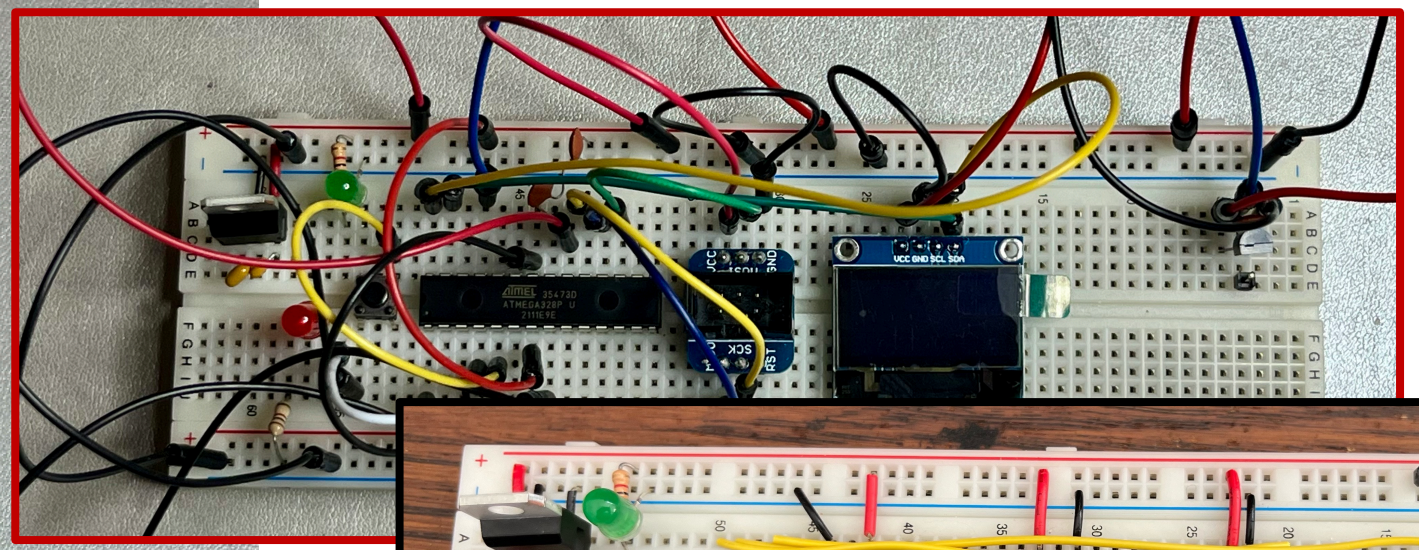
(This is a useful interface circuit)

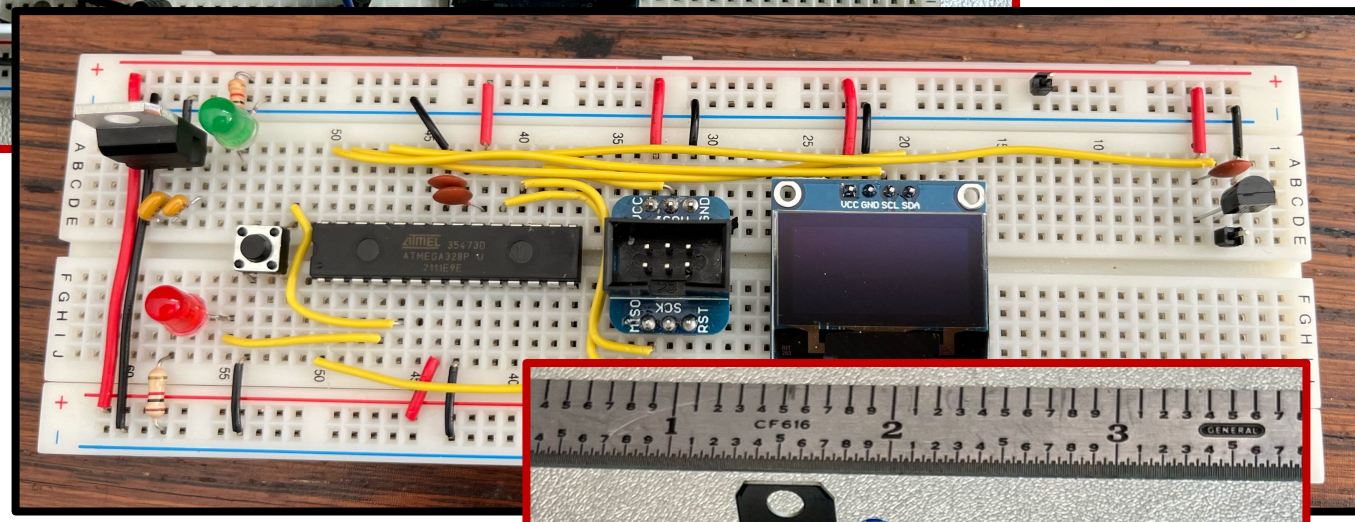


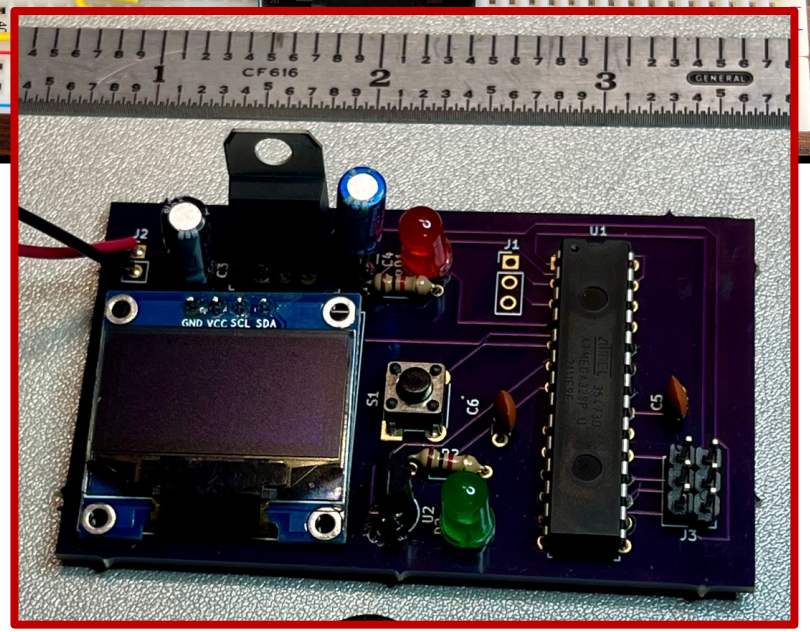
A look ahead to ECE-304...

design, build, test a battery-powered digital thermometer

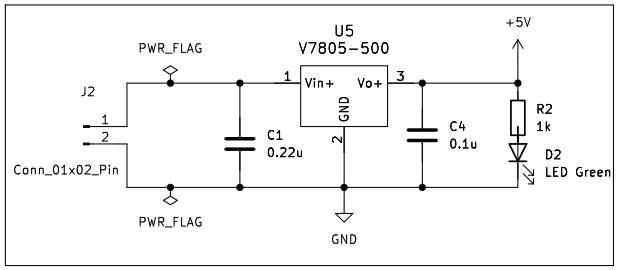

UNO R3
ARDUINO
VCC GND SCL SDA


VCC GND SCL SDA

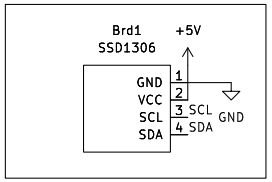

VCC GND SCL SDA


GND VCC SCL SDA

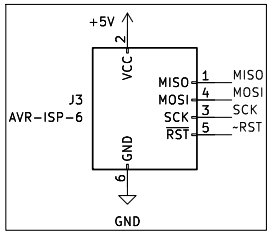
Power Supply



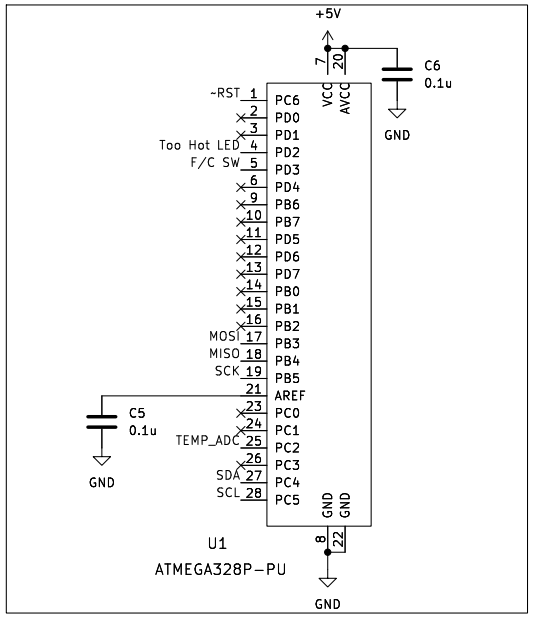
OLED Display



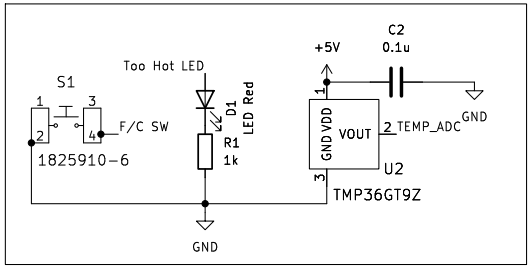
ICSP Socket



ATMega328P MCU



Temp Sensor, F/C switch, Too Hot LED

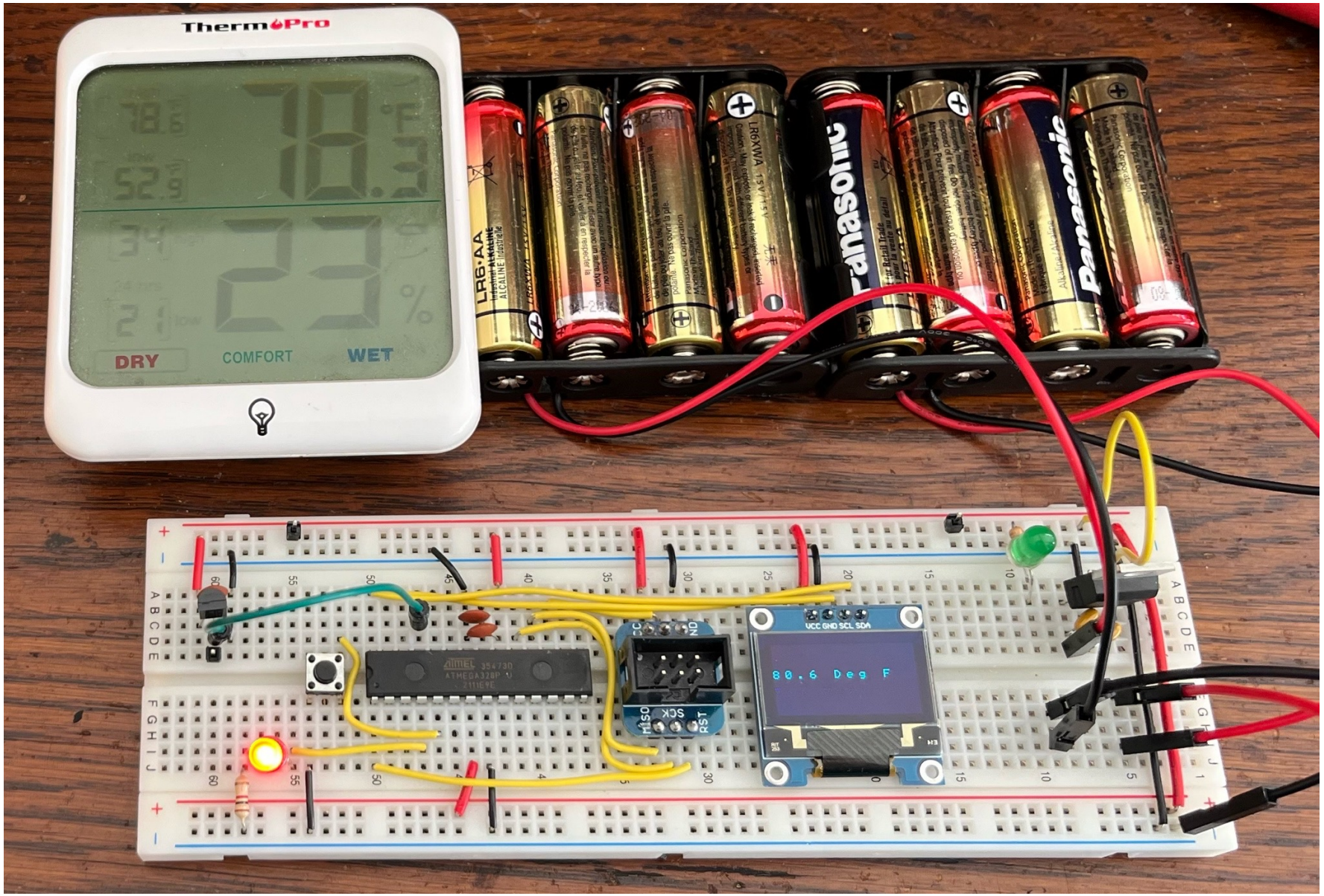


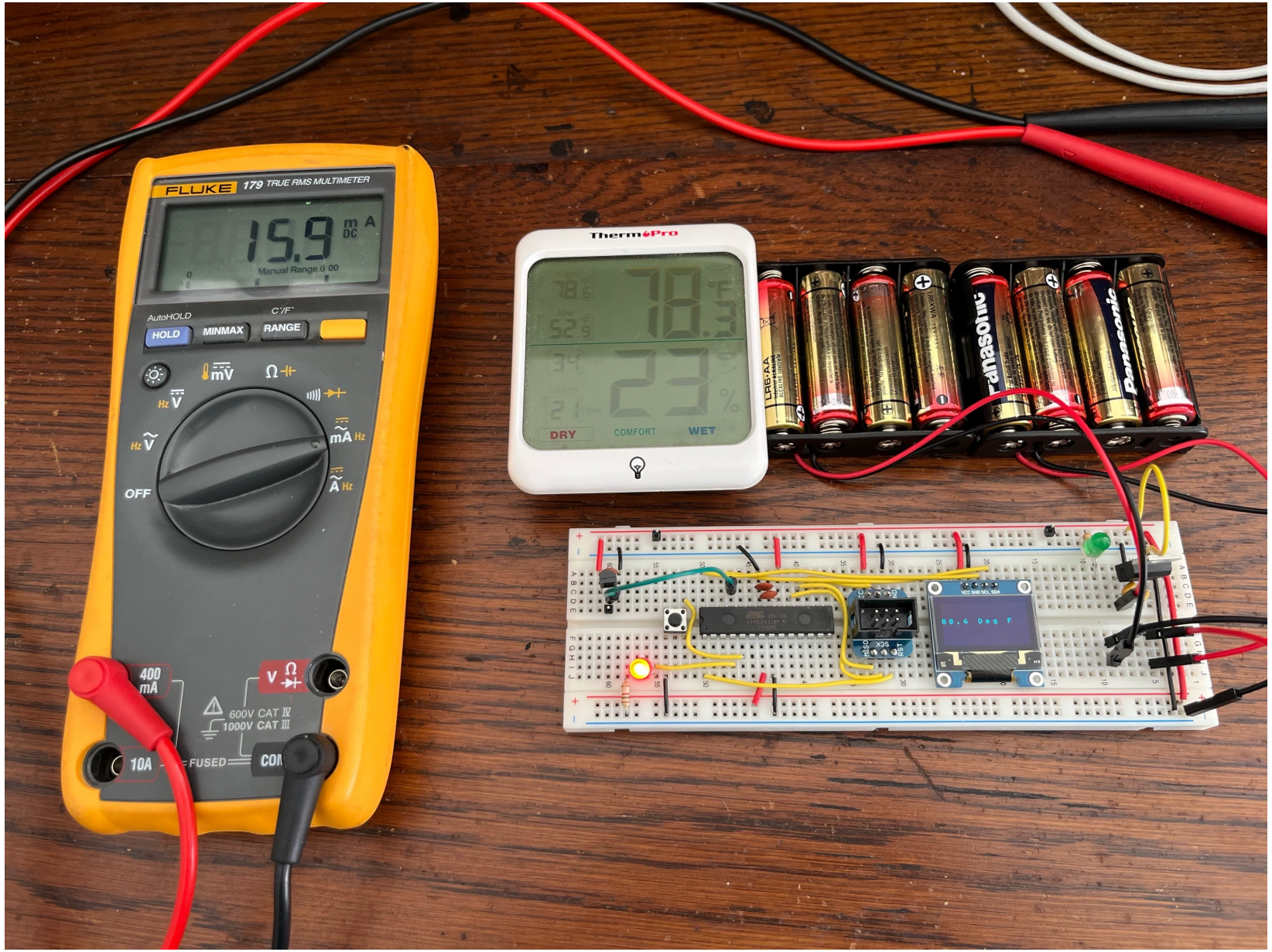
DJ McLaughlin
University of Massachusetts Amherst

Sheet: /
File: megaTempV2.kicad_sch

Title: megaTemp

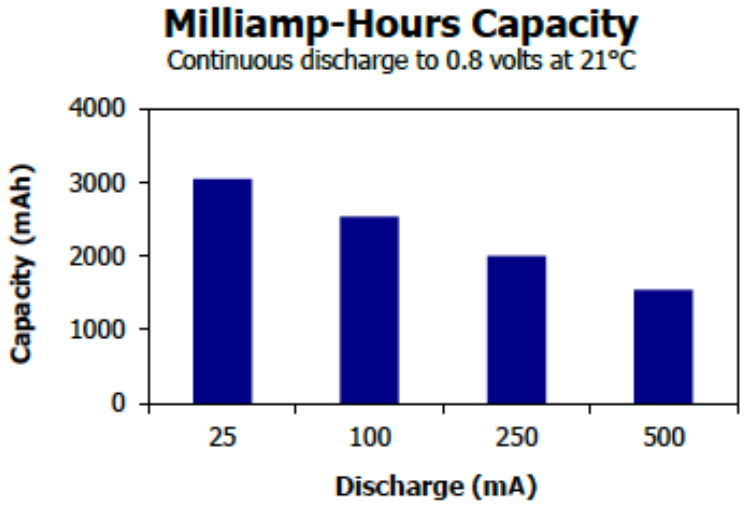
Size: USLetter	Date: 2023-12-10	**Rev: 2**
KiCad E.D.A. kicad 7.0.9		Id: 1/1


ThermoPro
78.3 °F
52 %
23 %
DRY
COMFORT
WET
LR6XWA
Panasonic
ATMEGA328P
VCC GND SCL SDA
80.6 Deg F
MISO
SCK
RST


FLUKE 179 TRUE RMS MULTIMETER
15.9 mA DC
Manual Range
AutoHOLD
HOLD
MINMAX
RANGE
OFF
400 mA
10A
600V CAT IV
1000V CAT III
FUSED
ThermoPro
78.3 °F
23 %
DRY
COMFORT
WET
Panasonic
80.6 Deg F

Battery Lifetime

Alkaline Battery
Data sheet info

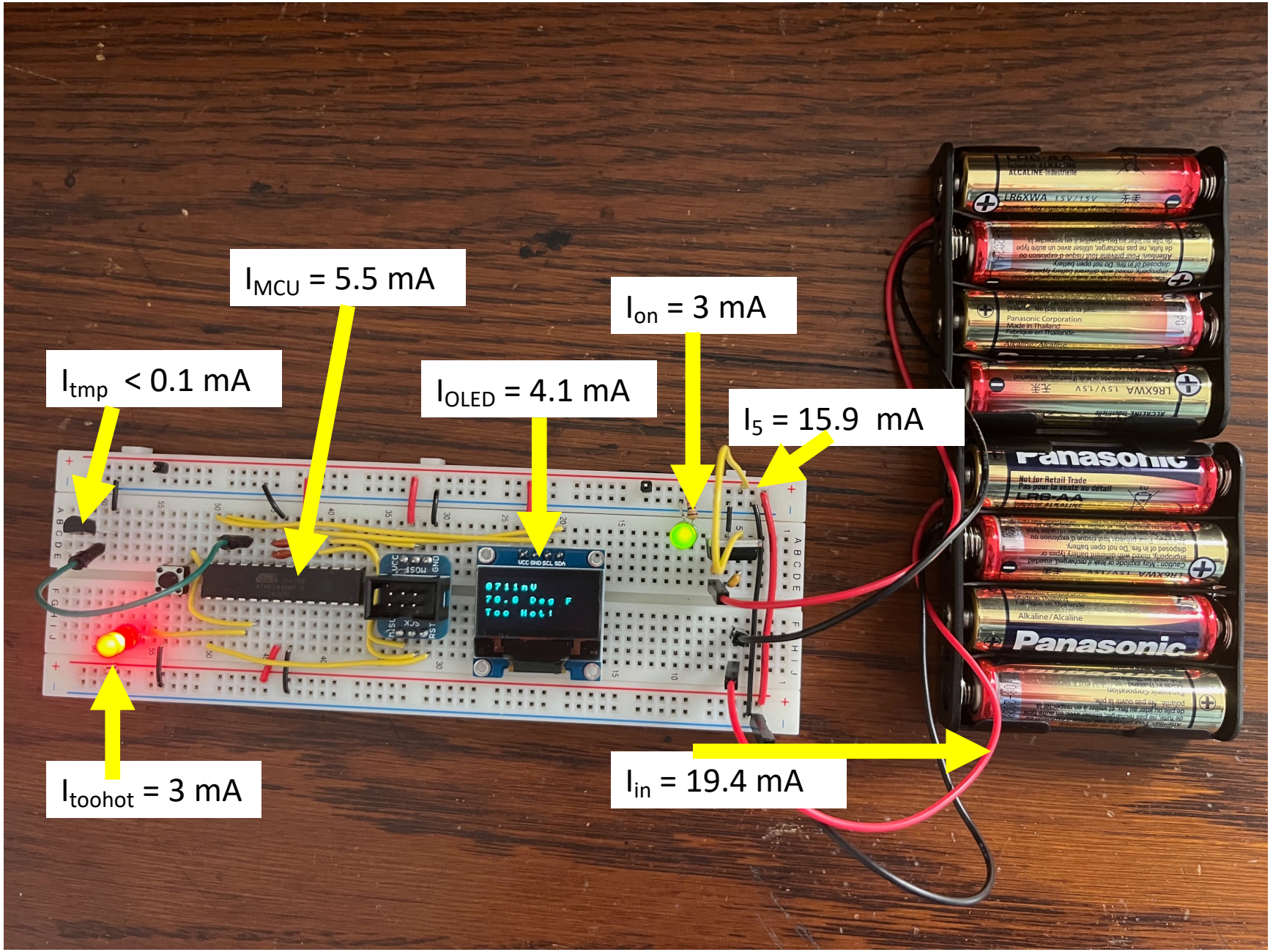


Alkaline AA battery capacity @ 25 mA current draw = 3000 mAh

1 month battery lifetime goal:
1 month x 28 days/month x 24 hours/day = 672 hours

3000 mAh/673h = 4.46 mA (assumes 3000 mAh @ 4.46 mA)

Can we keep the current draw < 4.5 mA?


I_{MCU} = 5.5 mA
I_{on} = 3 mA
I_{tmp} < 0.1 mA
I_{OLED} = 4.1 mA
I_5 = 15.9 mA
I_{toohot} = 3 mA
I_{in} = 19.4 mA
Panasonic
Alkaline/Alcaline

	Current	Redesign Goal
7805 Quiescent	3.5	0.1
Pwr ON LED	3	0.3
OLED	4.1	1.5
MCU	5.5	1.5
Too Hot LED	3	0.3
TMP36	0.1	0.1
Total	19.2	3.8

Redesign Tactics:

- increase LED series resistance
- Change from 28 pin ATmega328P to 8 pin ATtiny85
- decrease MCU clock frequency
- decrease system voltage from 5 to 3.3
- sleep the MCU between temp readings

ATmega328P MCU (28 or 32 pin package)

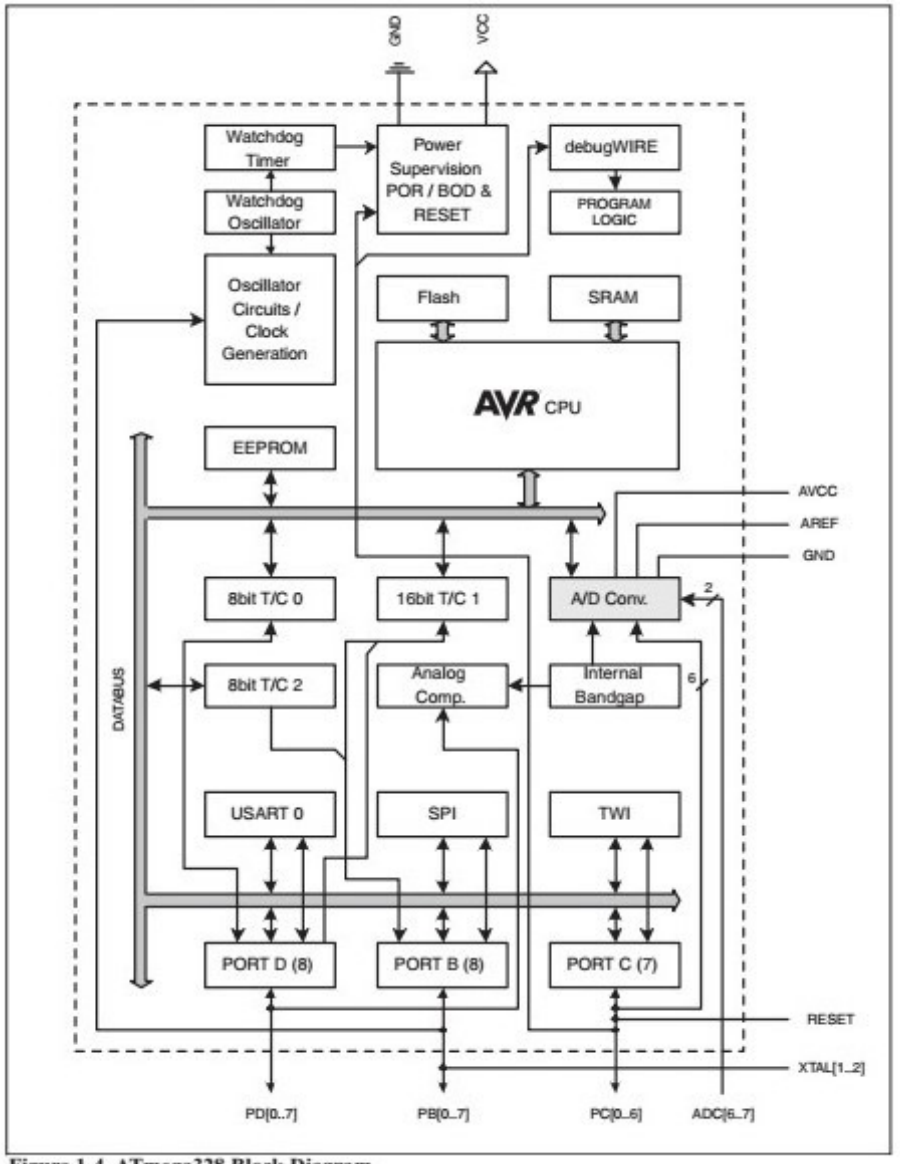


Figure 1-4. ATmega328 Block Diagram

$3.51 Digikey

28 pin PDIP

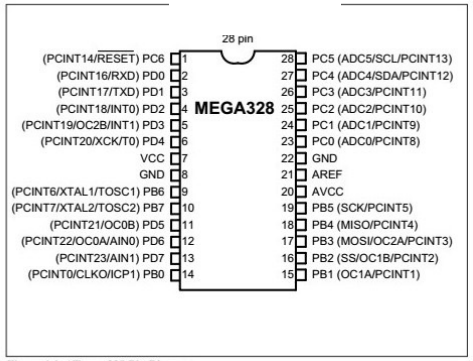


Figure 4-1. ATmega328 Pin Diagram

- 23 General Purpose I/O (GPIO) pins
  - PORTB Pins PB0-PB7
  - PORTD Pins PD0-PD7
  - PORTC Pins PC0-PC6
- 8 Analog to Digital Converter (ADC) pins
- 3 Timers
- UART for serial communication

- 32K Byte Flash ROM for Code
- 2K Byte data RAM
- 1K Byte data EEPROM

ATtiny85 MCU (8 pin package)

$1.19 from Digikey

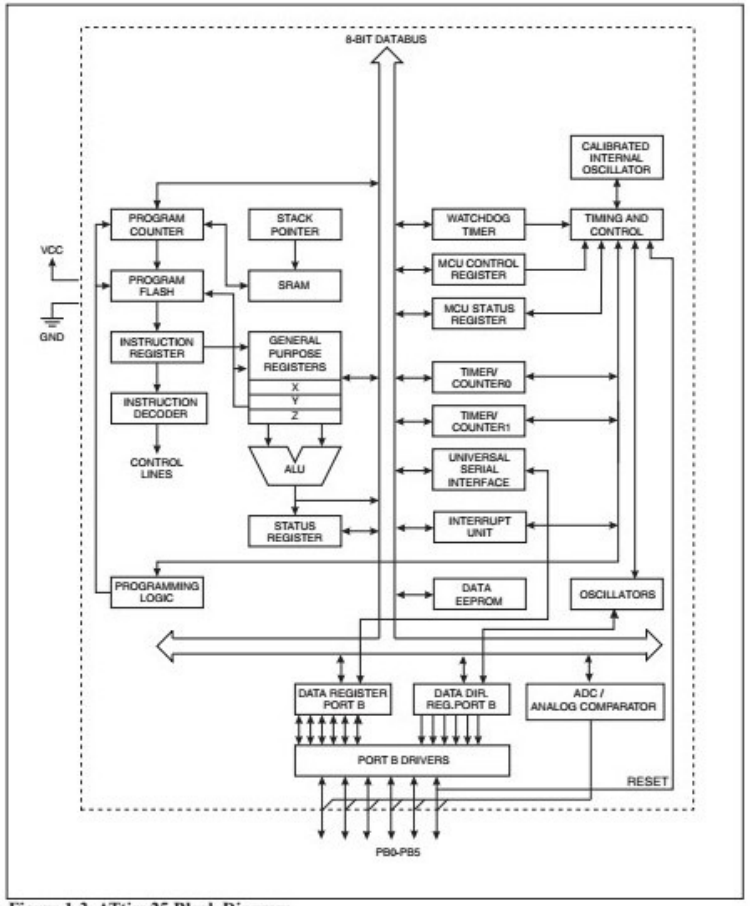
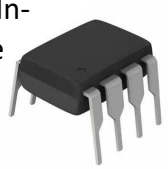
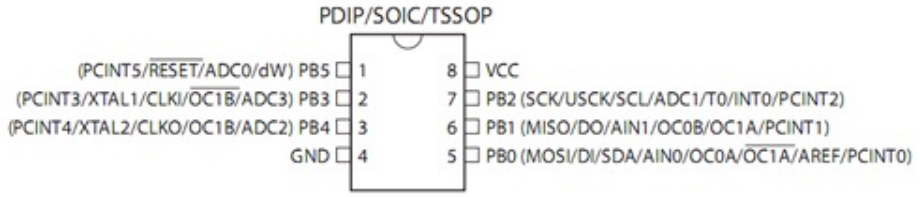


Figure 1-3. ATtiny25 Block Diagram

Plastic Dual In-Line Package (PDIP)



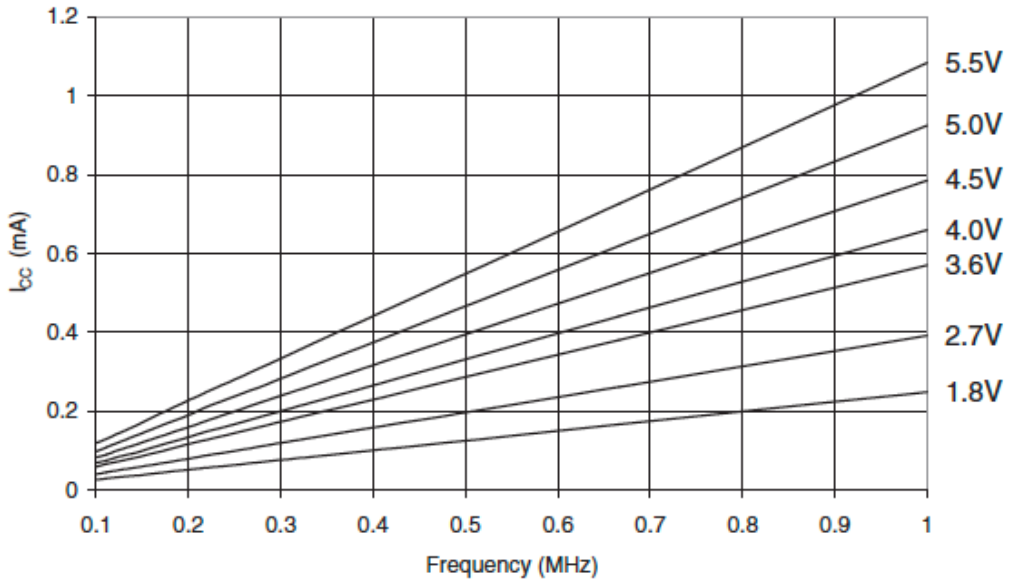
Pinout ATtiny25/45/85



- 6 General Purpose I/O (GPIO) pins
  - PORTB Pins PB0-PB5
- 4 Analog to Digital Converter (ADC) pins
- 2 Timers
- no communication

- 2K Byte Flash ROM for Code
- 128 Byte data RAM
- 128 Byte data EEPROM

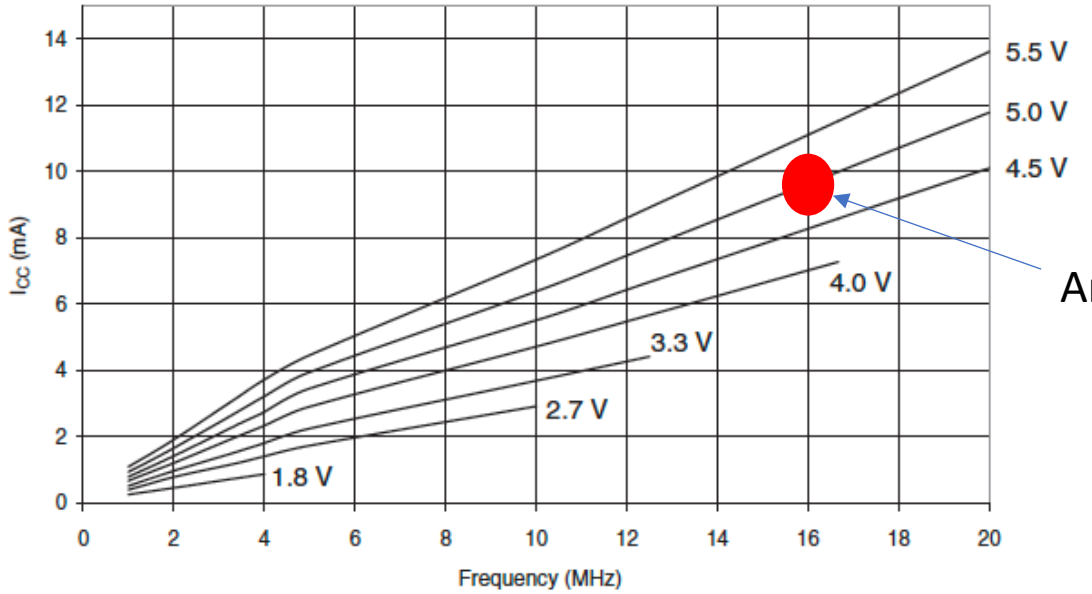
Figure 31-332. ATmega328P: Active Supply Current vs. Low Frequency (0.1-1.0MHz)



From 2018 AVR Mega Data Sheet

These curves correspond to use of an external clock.

Figure 31-333. ATmega328P: Active Supply Current vs. Frequency (1-20MHz)



Arduino Uno

	Current, I (mA)	
Vcc	**No Code**	**Do Nothing**
5.5	8.7 - 9.3	10.2
5	5.0 - 5.9	5.8
4	1.5	1.6
3.3	**0.7**	**0.8**
3	0.6	0.6
2	0.2	0.2
1.5	< 0.1	0.1

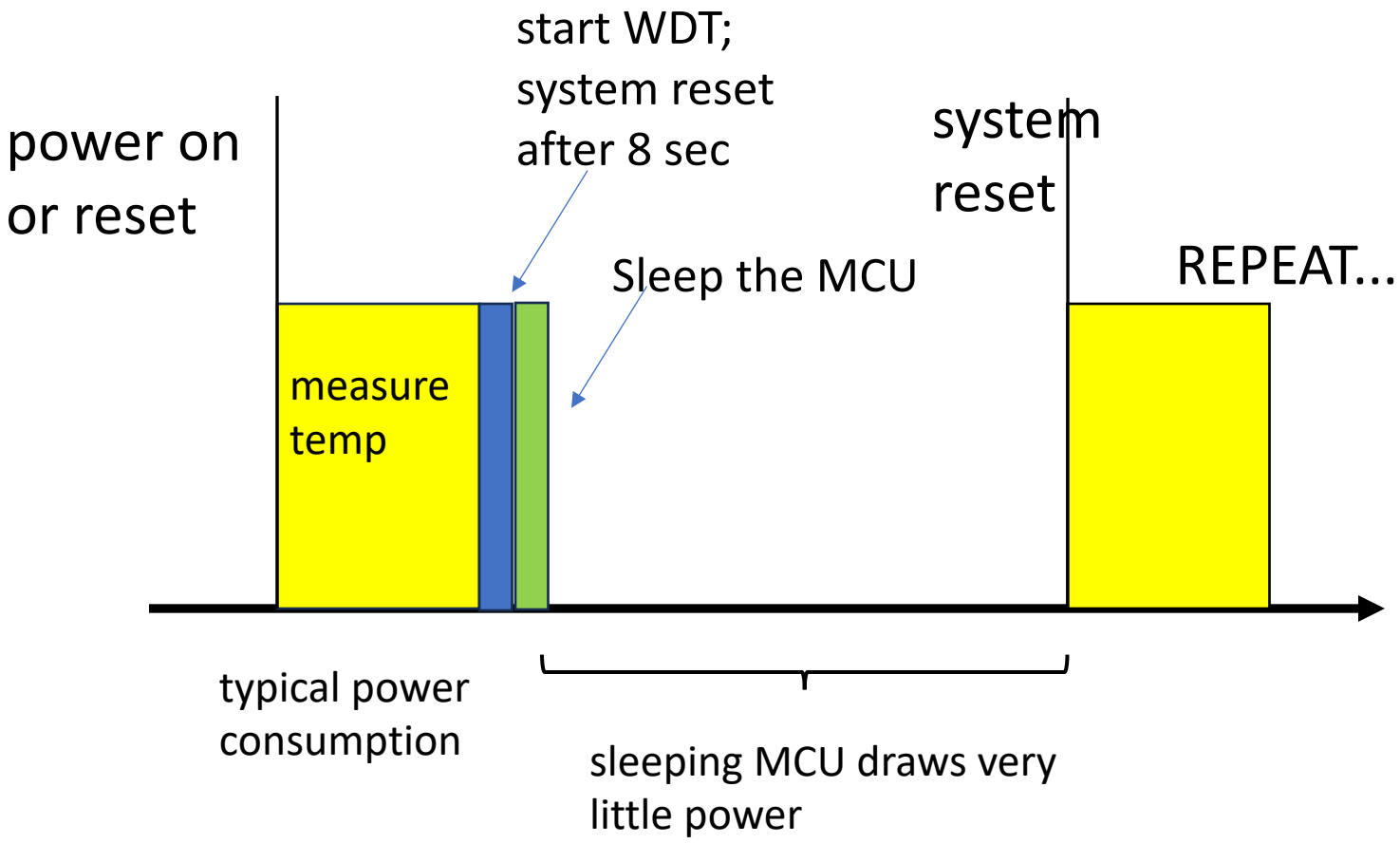
ATmega328P
Default Clock Rate
1 MHz

(8MHZ DIV8) set
by the low-byte
fuse

C doNothing.c > ...

```c
/*****************************************************************
 * donothing.c
 * ***************************************************************/

int main(void)
{
    while(1){}
}
/******* End of File *********/

```

sleeper.c using the watch dog timer (WDT)



```c
/* sleeper.c
 * Code to test ATmega328P power saving by Watchdog Timer (WDT)
 * This demo waits 3 seconds then powers down for 8 seconds
 * via SlEEP_MODE_POWER_DOWN with system reset on WDT timeout.
 * D. McLaughlin 3/5/24
 */

#include <avr/io.h>
#include <util/delay.h>
#include <avr/wdt.h>
#include <avr/sleep.h>

// Disable WTD and clear reset flag immediately at startup
void WDT_OFF()
{
    MCUSR &= ~(1 << WDRF);
    WDTCSR = (1 << WDCE) | (1 << WDE);
    WDTCSR = 0x00;
}

int main(void)
{
    WDT_OFF(); // Disable WDT & clear reset flag @ startup

    _delay_ms(3000);

    wdt_enable(WDTO_8S); // Start the WDT; reset after 8 seconds

    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_mode();       // Put the system to sleep
    while (1)
        ;
}
```

ATmega328P Current vs Vcc

Data for a previously unused, factory default IC

Default fuse settings: 8 MHz Internal RC Oscillator with DIV8 enabled for 1 MHz internal RC clock

	Current, I (mA)				
Vcc	**DataSheet**	**No Code**	**Do Nothing**	**Sleeper**	**TempSleep**
5.5	1.1	8.7 - 9.3	10.2	11.0 / 0.01	0.04
5	0.95	5.0 - 5.9	5.8	7.0 / 0.01	0.03
4	0.65	1.5	1.6	2.0 / 0.01	0.02
3.3	**0.5**	**0.7**	**0.8**	**0.9 / <0.01**	0.01
3	0.5	0.6	0.6	0.7 / <0.01	0.01
2		0.2	0.2	x	
1.5	< 0.2	< 0.1	0.1	x	

DatasSheet: Approx readings from graphs Fig 35.1, megaAVR data sheet, 2018

Configuration is external clock, not internal clock

NoCode: New IC on board without any flashed code

DoNothing: IC flashed with a while(1){ } do nothing loop

Sleeper: PD2 set as output, made high, 3 second delay, then SLEEP_MODE_POWER_DOWN for 8 seconds

x: Device not functioning below 2.2 V

TempSleep: This is a calculation that assumes a 1/50 second measurement is made once every 8 seconds

with WDT & SLEEP_MODE_POWER_DOWN in between measurements

Conclusion: Using default 8 MHZ DIV 8 clock and Vcc = 3.3V, I < 1.0 mA

Can reduce substantially, to negligible value wih power down mode & watchdog timer

AVR® MCUs: The Next Generation of ATtiny and ATmega

MT

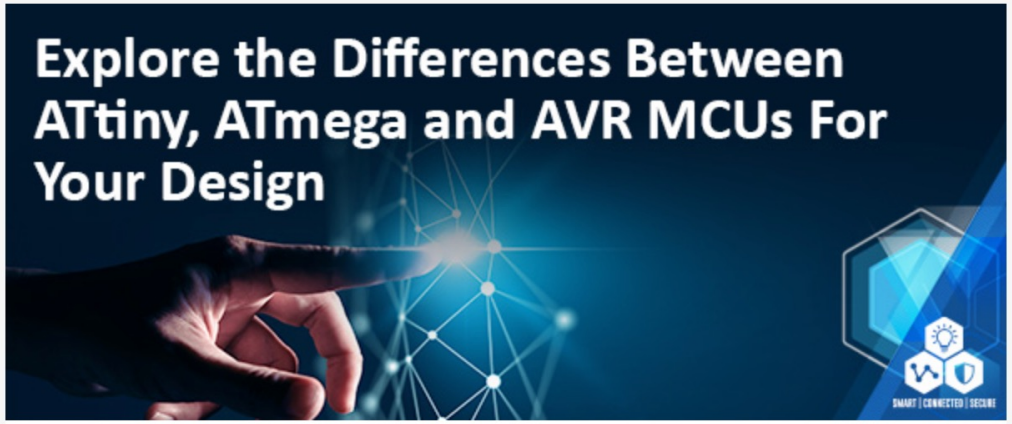
Microchip Technology Inc. <mchpnewsletters@mkto.microchip.email>

Today at 10:20 AM

To: David McLaughlin

received 3/7/24

MICROCHIP



AVR® MCUs: The Next Generation of ATtiny and ATmega

Are you navigating the landscape of microcontrollers (MCUs) and wondering about the nuances between ATtiny, ATmega and AVR® MCUs? Finding the right fit for your design is crucial. These MCUs have long been trusted choices for automotive, industrial, home appliance, medical, consumer and other innovative applications, and as technology advances, so do the capabilities of these product lines.

The recent rebranding of the ATtiny and ATmega lines under the AVR MCU name signifies a unification, accompanied by an expansion of peripherals. This shift aims to offer a more comprehensive and versatile portfolio, empowering you to develop compact, user-friendly and robust designs.

The AVR DD MCU serves as an excellent entry point into the AVR family of MCUs. It inherits the core features of both ATtiny and ATmega devices while incorporating new peripherals like Multi-Voltage I/O (MVIO), which eliminates the need for an external level shifter, and an impressive suite of analog functionalities including a 12-bit ADC, 10-bit DAC and an analog comparator.

To learn more about AVR DD MCUs, watch this overview video.


"That's all Folks!"