# ECED3901 Design Methods II

LECTURE #6: EMBEDDED PROGRAMMING #2

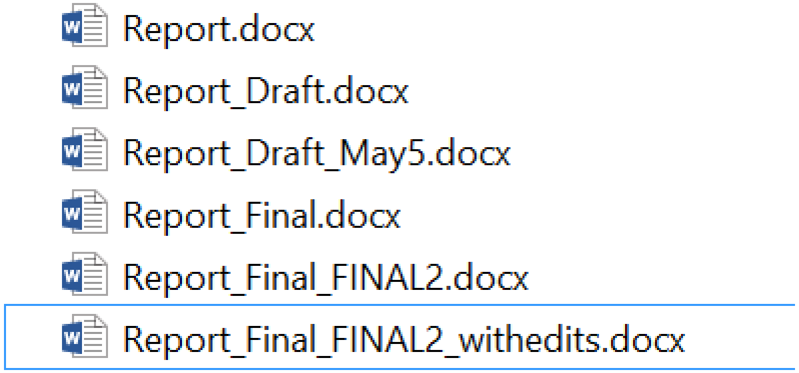
Colin O’Flynn

# What are we covering?

- Code Repositories
- Debugging Strategies
    - Blinking the lights
    - Sprinkling printf()
    - Using in-circuit emulator (ICE)

# Source Code Management

# The Problem…



Report.docx

Report_Draft.docx

Report_Draft_May5.docx

Report_Final.docx

Report_Final_FINAL2.docx

Report_Final_FINAL2_withedits.docx

# Repository

- Central “codebase”
- Lets you see what changes between versions
- Allows ”branches” which don’t affect main codebase

Tutorial example: https://www.youtube.com/watch?v=cFbCusX9bKs

# GITHub Demo

GITHub → Very popular host of GIT repositories

NOTE: To download Windows client, see:
https://windows.github.com

# Debugging

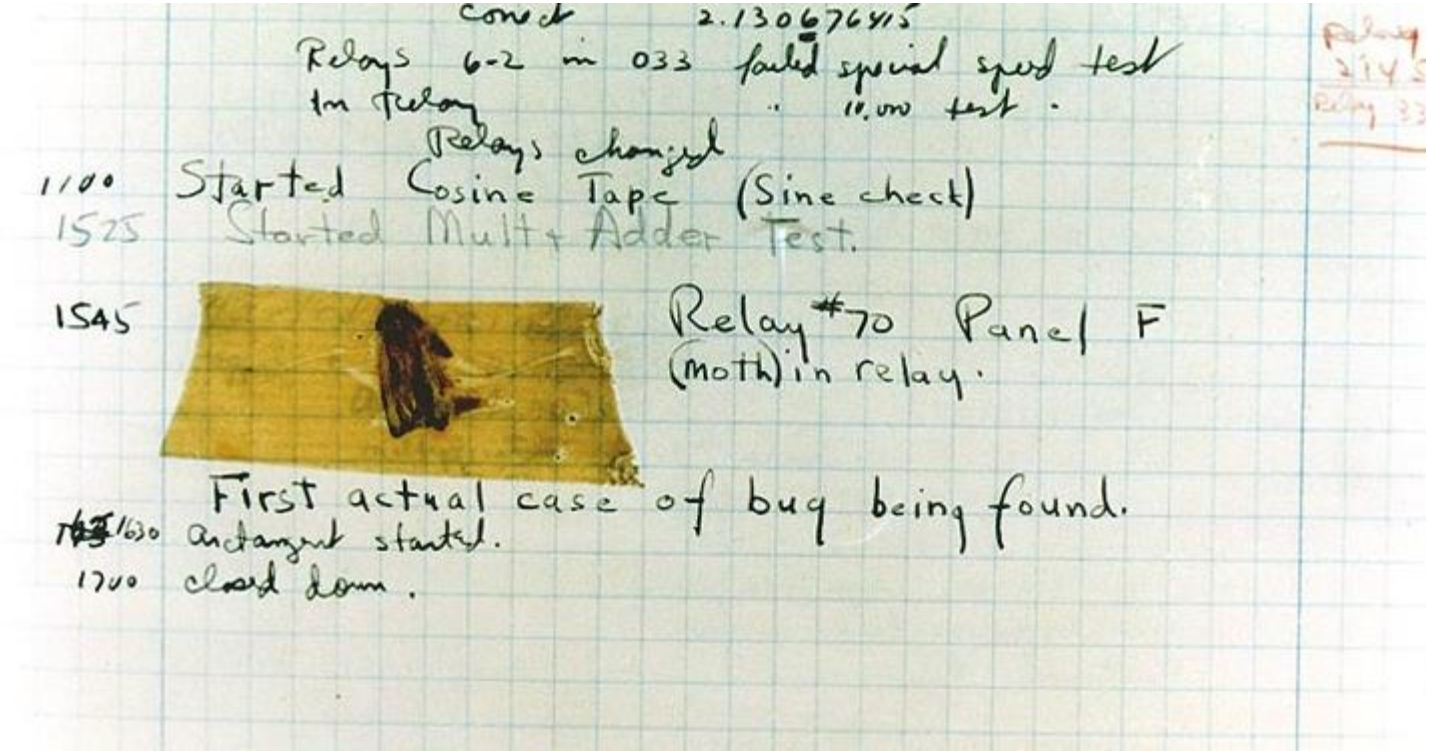
# About Debugging

**FACT:** Debugging is at least 2x as hard as programming

**THEREFOR:** If you write your “cleverest possible” code, you are too dumb to fix it.

**WHICH MEANS:** Don’t write overly complicated code! You are better off having simple code that works than complicated code that doesn’t.

# What are Bugs?



conct 2.130676415

Relays 6-2 in 033 failed special speed test
in relay . 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult+ Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1630 antangent started.

1700 closed down.

Relay
2145
Relay 33

http://en.wikipedia.org/wiki/Software_bug

Colin O’Flynn

# What is Debugging?

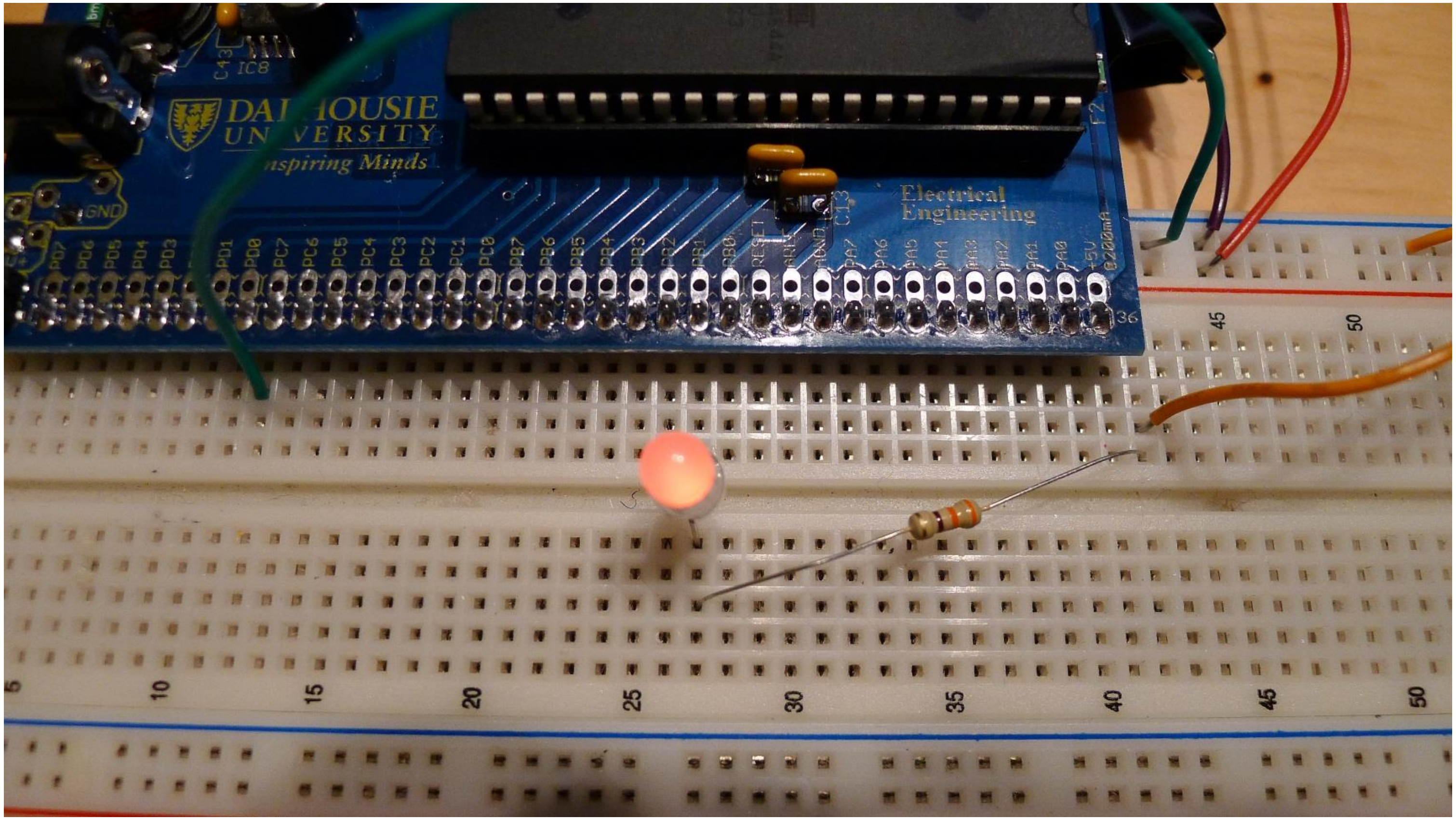
# Basic Debugging Philosophy

1. Determine what the program *should* do.
2. Determine your assumptions about how the program should be doing this.
3. Determine what program is *actually* doing.
4. Test your assumptions on the actual program to determine where things go wrong.

# Debugging Example

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

#define LED_ON()  DDRB |= 1<<1; PORTB |= 1<<1;

int main(void)
{

    while(1){

        //Turn on LED if switch on
        if (PIND & (1<<0))
            LED_ON();

    }

}
```

# Running the Program…


DALHOUSIE UNIVERSITY
Inspiring Minds
Electrical Engineering
IC8
C43
GND
PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0 PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0 PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0 RESET AREF GND PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0 +5V @200mA
C13
F2
36
5
10
15
20
25
30
35
40
45
50

# Debugging Flow?

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

#define LED_ON()  DDRB |= 1<<1; PORTB |= 1<<1;

int main(void)
{

    while(1){

        //Turn on LED if switch on
        if (PIND & (1<<0))
            LED_ON();

    }

}
```

Assumption #1

Assumption #2

# Debugging Step #1

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

#define LED_ON()  DDRB |= 1<<1; PORTB |= 1<<1;

int main(void)
{

    while(1){
        //Turn on LED if switch on
        //if (PIND & (1<<0))
        //    LED_ON();
    }

}
```

# Debugging Step #2

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

#define LED_ON()  DDRB |= 1<<1; PORTB |= 1<<1;

int main(void)
{

    while(1){
        //Turn on LED if switch on
        if (PIND & (1<<0))
           continue;
        //    LED_ON();
    }

}
```

# Debugging Fix #1

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

#define LED_ON()  DDRB |= 1<<1; PORTB |= 1<<1;

int main(void)
{

    while(1){

        //Turn on LED if switch on
        if (PIND & (1<<0)) {
            LED_ON();
        }

    }

}
```

# Debugging Fix #2

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

#define LED_ON()  {DDRB |= 1<<1; PORTB |= 1<<1;}

int main(void)
{

    while(1){

        //Turn on LED if switch on
        if (PIND & (1<<0))
            LED_ON();

    }

}
```

# Root Cause

PORTB |= 1<<1

→ Pull-Up Enable

# Debugging Hints

1. Don’t spent too long staring at code to find missed brackets, semicolons, etc.
   1. This *is* still a useful step sometimes, but:
   2. Very easy to miss them however, as in previous example due to macro

2. Keep Notes
   1. Write down your “assumptions”
   2. Write down what you have proven to work, what is unknown

3. Take Breaks
   1. Often you need a change of perspective to see what you were doing wrong
   2. Having a break helps – go for a walk, have some food, talk to someone about it, etc.

# Blinking Lights Debugging

# Debugging with LEDs

- Can turn on LEDs when certain code executing
- Possible with lots of LEDs to show more detail, or use blink pattern, etc.

```
if (readsensor() == 0){
    drive_into_wall();
} else {
    drive_left();
}
```

```
if (readsensor() == 0){
    LED_ON(0);
    drive_into_wall();
} else {
    LED_ON(1);
    drive_left();
}
```

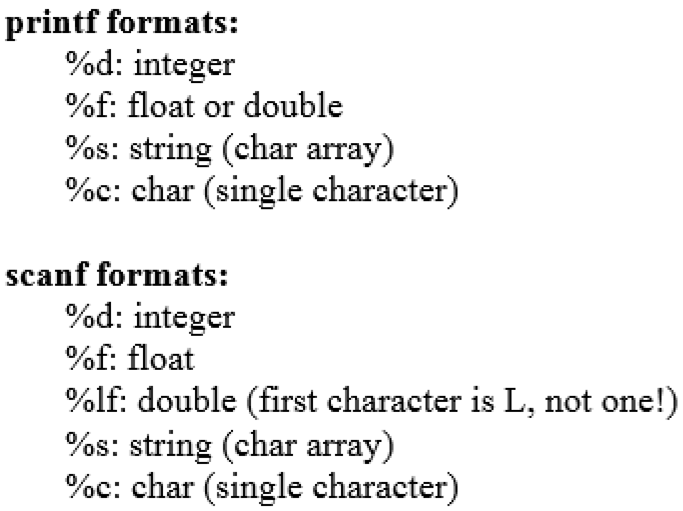
# printf() Debugging

# What is printf()

- On command-line program, prints a string to the screen
  - i.e., see Hello World program
- Also used to print variable values

# Printf() Usage

```
char name[] = "Fred";
int dollars = 25;
float waterL = 12.2;

printf("Hello %s, I have %d dollars, and %f L of
water\n", name, dollars, waterL);
```



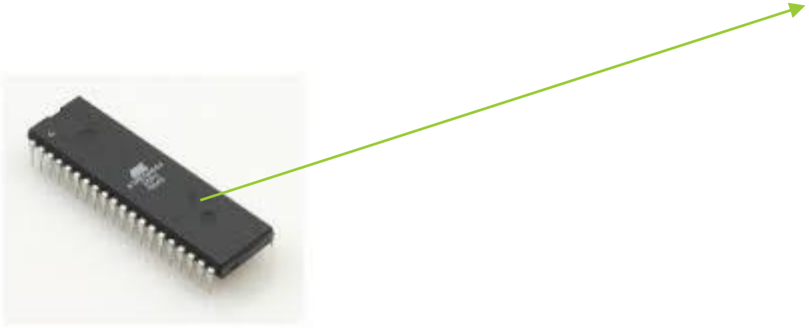
**printf formats:**

- %d: integer
- %f: float or double
- %s: string (char array)
- %c: char (single character)

**scanf formats:**

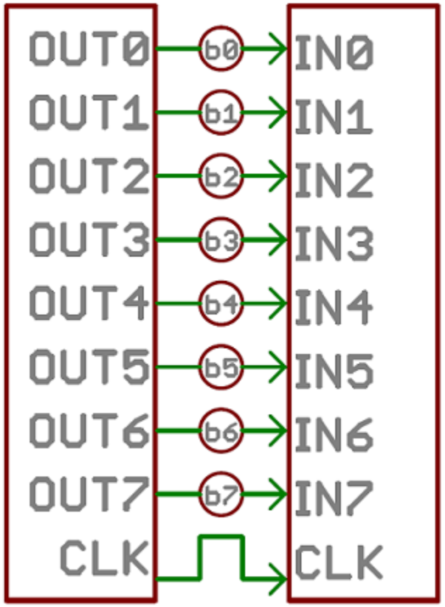
- %d: integer
- %f: float
- %lf: double (first character is L, not one!)
- %s: string (char array)
- %c: char (single character)

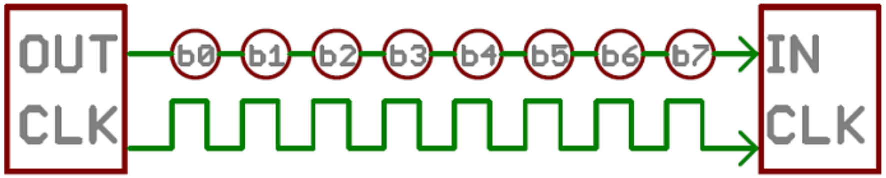
# Where does printf() go on micro?



# Serial Port

NOTE: For more detail please see this page, which is the source for the figures I've used here: https://learn.sparkfun.com/tutorials/serial-communication/all

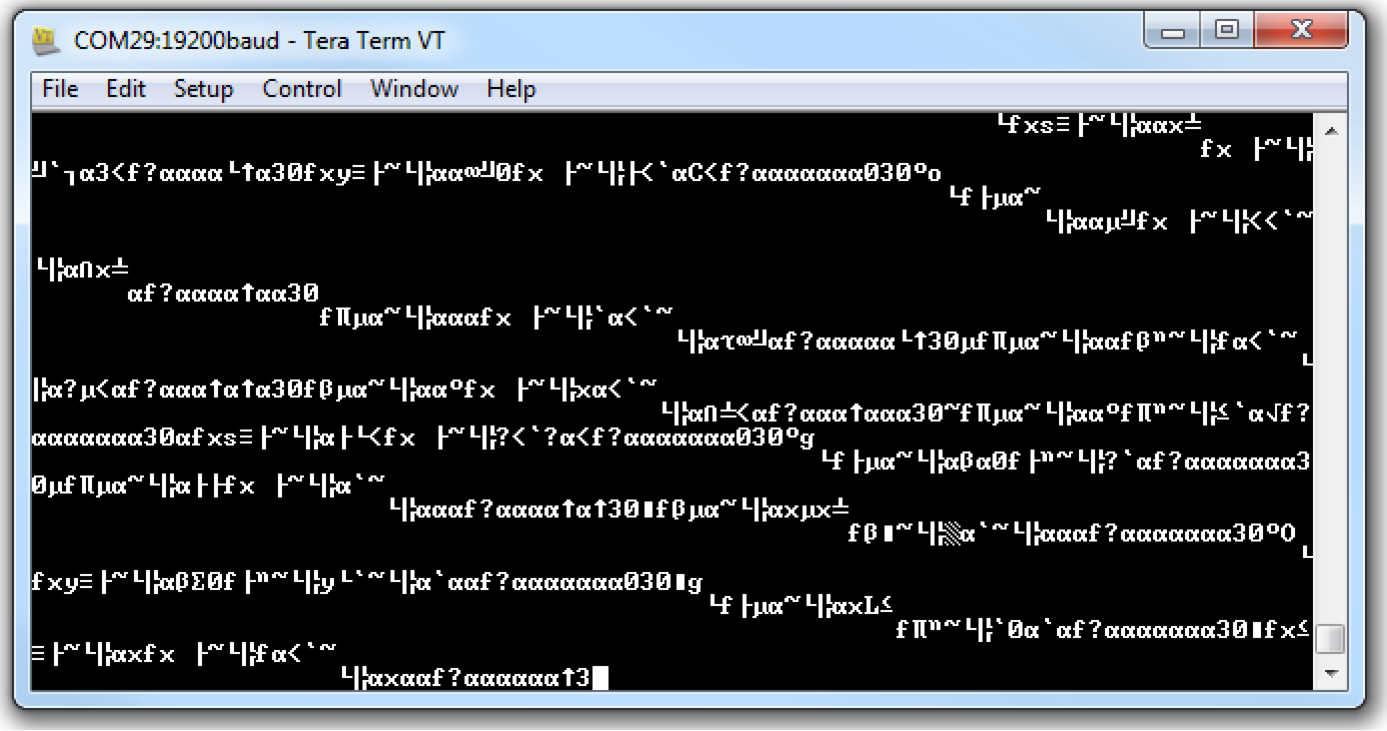




# Asynchronous Serial

- Uses an *implied* clock
- Requires both sender and receiver to know what that clock *should* be
  - This is called the BAUD RATE
- Some other things that require definition too, not important for our serial port

# Wrong Baud Rate

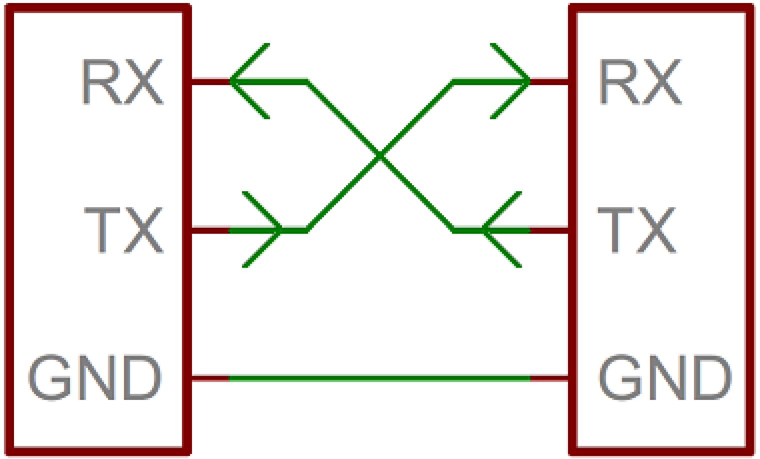


COM29:19200baud - Tera Term VT
File Edit Setup Control Window Help

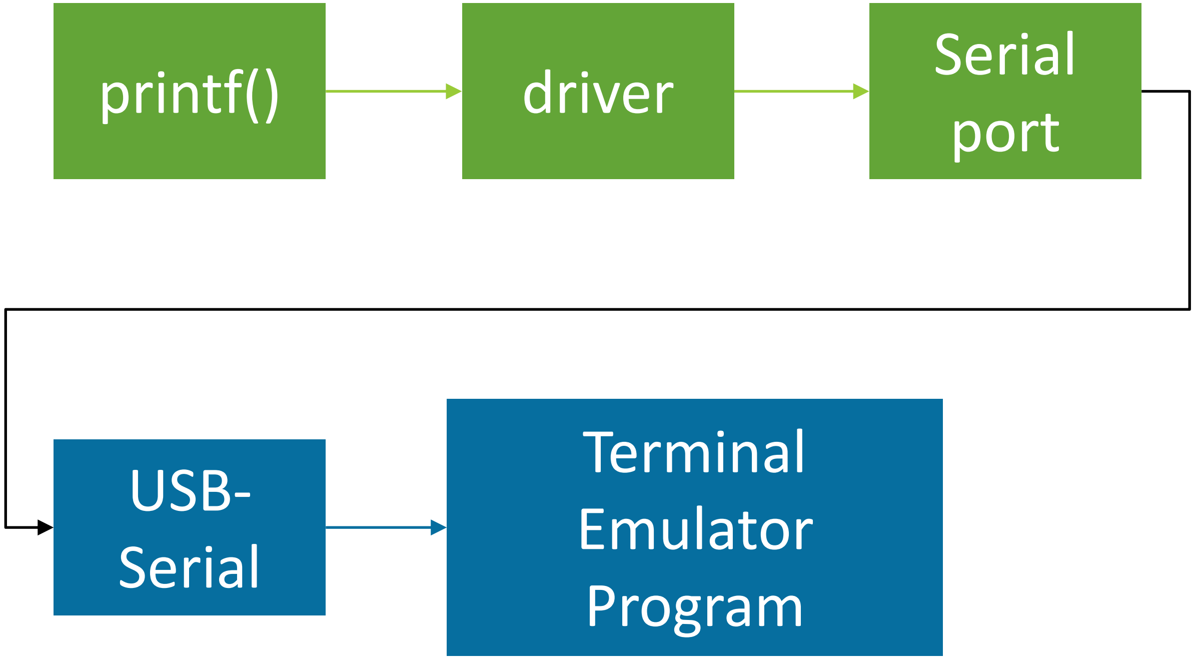
```
                                                            ╘f xs≡├~╚╠ααx±
                                                                         f x  ├~╚╠
╝`┐α3<f?αααα└↑α30f xy≡├~╚╠αα∞╝0f x  ├~╚╠├<`αC<f?ααααααα030°o
                                                         ╘f ├µα~
                                                               ╚╠ααµ╝f x  ├~╚╠<`~

╚╠αΠx±
      αf?αααα↑αα30
                  f Πµα~╚╠αααf x  ├~╚╠`α<`~
                                        ╚╠ατ∞╝αf?ααααα└↑30µf Πµα~╚╠ααf βⁿ~╚╠f α<`~
╠α?µ<αf?ααα↑α↑α30f βµα~╚╠αα°f x  ├~╚╠xα<`~
                                       ╚╠αΠ≐<αf?ααα↑ααα30~f Πµα~╚╠αα°f Πⁿ~╚╠≤`α√f?
ααααααα30αf xs≡├~╚╠α├└<f x  ├~╚╠?<`?α<f?ααααααα030°g
                                                 ╘f ├µα~╚╠αβα0f ├ⁿ~╚╠?`αf?ααααααα3
0µf Πµα~╚╠α├├f x  ├~╚╠α`~
                      ╚╠αααf?αααα↑α↑30▮f βµα~╚╠αxµx±
                                                   f β▮~╚▒α`~╚╠αααf?ααααααα30°O

f xy≡├~╚╠αβΣ0f ├ⁿ~╚╠y└`~╚╠α`ααf?ααααααα030▮g
                                          ╘f ├µα~╚╠αxL≤
                                                      f Πⁿ~╚╠`0α`αf?ααααααα30▮f x≤
≡├~╚╠αxf x  ├~╚╠f α<`~
                   ╚╠αxααf?αααααα↑3█
```

Colin O’Flynn

# Serial Connection



# Serial Port Connection



# Debugging with printf()

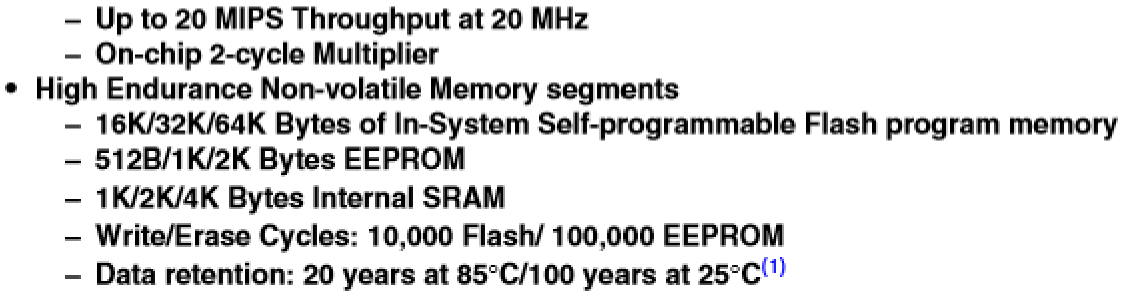
- Allows you to dump variable values
- Test analog circuits by reading output for example
- Test program flow by simply inserting printf() throughout, i.e.:

```
if (readsensor() == 0){
    drive_into_wall();
} else {
    drive_left();
}
```

```
if (readsensor() == 0){
    printf("Driving into wall\n");
    drive_into_wall();
} else {
    printf("Driving Left\n");
    drive_left();
}
```

# Special Note on printf() with AVR

SRAM vs. FLASH



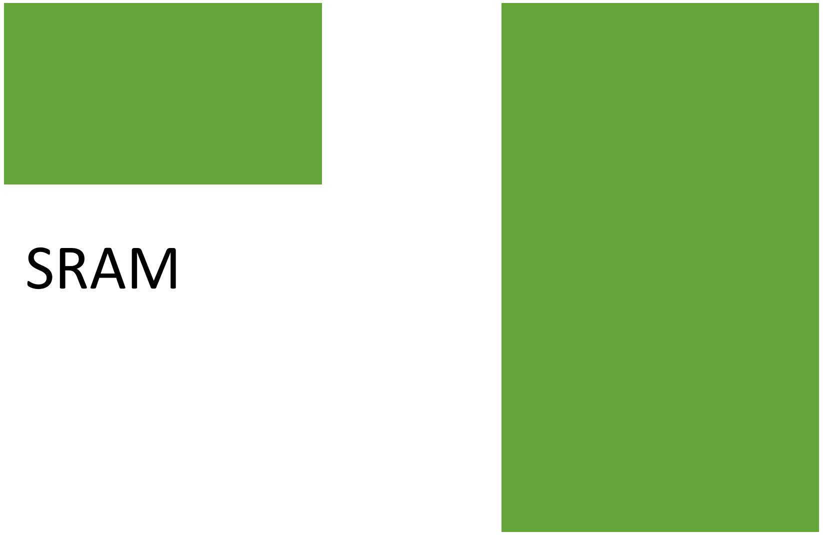
- Up to 20 MIPS Throughput at 20 MHz
- On-chip 2-cycle Multiplier

- High Endurance Non-volatile Memory segments
  - 16K/32K/64K Bytes of In-System Self-programmable Flash program memory
  - 512B/1K/2K Bytes EEPROM
  - 1K/2K/4K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]

…What happens with this:

printf(“Hello There Friend\n”);

# SRAM Usage



SRAM

Code

# Special Note on printf() with AVR

…SAVE RAM SPACE!! Instead use special _P version if doing longer or a lot of printf() statements:

```
printf_P(PSTR(“Hello There Friend\n”));
```

# Common C Errors

# …avoiding them

NOTE: Lint will catch many of these errors! See Programming #1 Lecture.

# Using = instead of ==

```
int x = 5;

if ( x = 6 )

  printf("x equals 6\n");
```

# Fixing…

```
int x = 5;

if ( x == 6 )

  printf("x equals 6\n");
```

# Off-by-one Errors

- Arrays start at 0 index

i.e. to assign values 1-16 to an array:

```
uint8_t somearray[16];
for(uint8_t i = 0; i < 16; i++){
    somearray[i] = i + 1;
}
```

Note we never assign somearray[16]! The maximum value is held in somearray[15].

# Loop Errors

```c
uint8_t somearray[16];
for(uint8_t i = 0; i < 16; i++);
{
    somearray[i] = i + 1;
}
```

THIS is loop body now!

# Forgetting a break in a switch

```
int x = 2;
switch(x) {
case 2:
  printf("Two\n");
case 3:
  printf("Three\n");
}
```

# Fixed

```c
int x = 2;
switch(x) {
case 2:
  printf("Two\n");
  break;
case 3:
  printf("Three\n");
  break;
}
```

# Moving Forward with your Microcontroller

Lab #2 – Much of this based on ECED3204 Labs. Looking at some examples…

Colin O’Flynn

# Conclusions

- Using code repositories can be very useful way of managing projects
- Debugging requires discipline to avoid wasting your time chasing down rabbit holes
- Check every assumption you are making no matter how dumb it seems
- Use static analysis tools (if possible) to catch bugs
- The ECED3204 labs have a lot of useful information on programming examples!