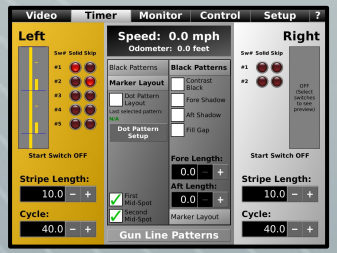
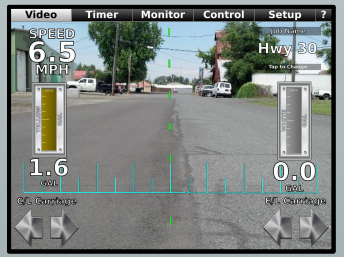
SKIP LINE

Providing industry-leading road construction electronics since 1972.





Overview

- Intro & Background
- EE vs Firmware Engineer
  - Important ECE skills
- Writing Great Code
  - Writing Good Code
- Packaging - Enclosures, ESD, Connectors
-

Credit some slides: Jack Ganssle, “Better Firmware Faster”

Intro & Background

Brad Nelson
Chief Operations Officer
Skip-Line, Inc.
La Grande, Oregon

- Class of 2011, ECE
- Fluent in Vietnamese
- Likes pina coladas and walks on the beach

EE vs Firmware Engineer

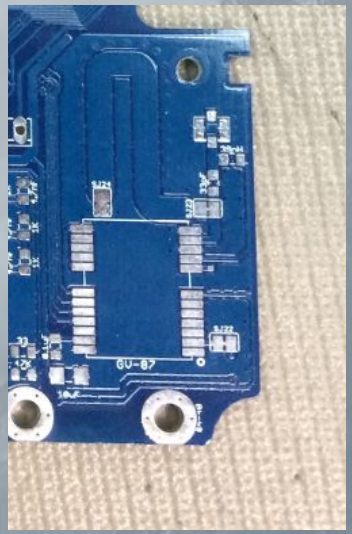
Sorry to disappoint...

- If you don’t like software, you’re going to have a rough career in ECE.
- Most EE jobs: Firmware intensive
- System design, DSP, EMC control, etc.
  - Ex: DDR RAM termination controlled in registers, not by hardware!
  - Incorrect termination caused bit 6 on a 32 bit bus to be random.

EE vs Firmware Engineer

Most Important ECE Skills

- PCB Layout:
  - High frequency signals
  - Creating impedance matched traces
  - Connector placement
  - Mechanical mounting
  - Ground currents: Where do they go?!



EE vs Firmware Engineer

Most Important ECE Skills

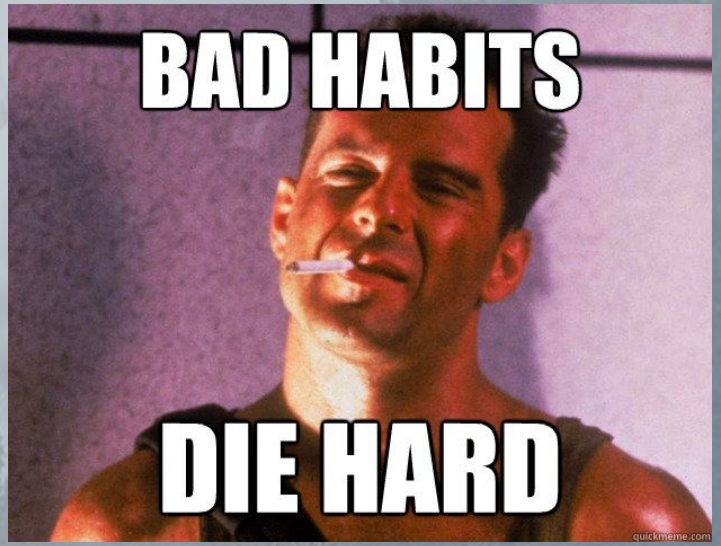
- Enclosure:
  - EMC control
  - Ground path control
  - ESD control
  - RF radiation
  - Robustness
  - Manufacturability (Tape? Screws? Pressure fit?)
- Recently:
  - Driving 15A linear motors
  - Outside enclosure: Ground no bueno!
  - Inside enclosure: Muy bien!

EE vs Firmware Engineer

Most Important ECE Skills

- Ability to learn and adapt to firmware:
  - Style Guidelines
  - Best Practices
  - Hardware
  - New platforms

It’s not what you know - it’s systems and habits that you use and apply from one project to the next that will make your career great!



Writing Great Code

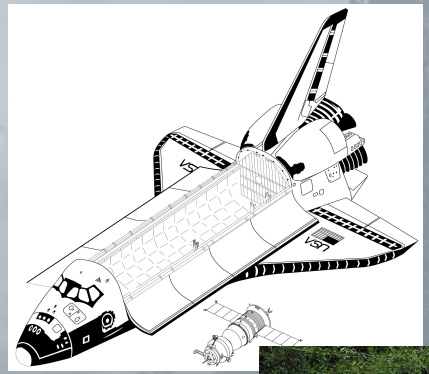
SKIP LINE

Firmware: Expensive!

Formerly: Hardware most expensive

Fighter jet manufacturers:

- No more room for analog hardware, let's make it digital!
- Old Record: Space Shuttle, $1000/LOC



Don't be fooled - quality is king!

- Toyota: $1.2B fine for uncontrolled acceleration
- 1M LOC means...
- New record: $1200/LOC + cost of development!

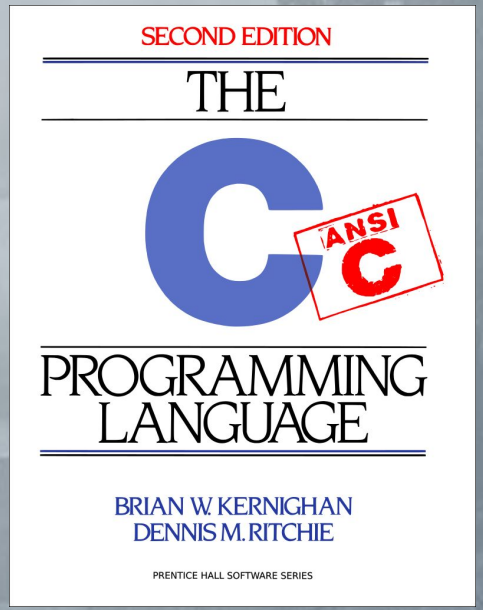
Writing Great Code

SKIP LINE

Simple and Small Units

“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?”

- Brian Kernighan



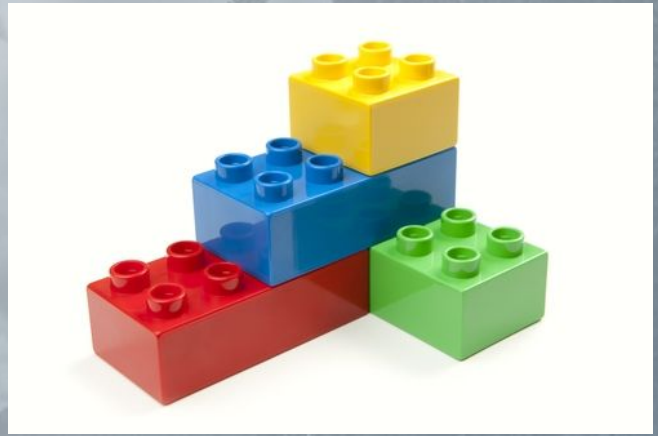
Writing Great Code

SKIP LINE

Simple and Small Units

- Write small modules
- Do thorough, standalone unit tests whenever possible
- Have as few dependencies as possible

- Code Review: ***Mission Critical!!!***



Writing Great Code

Volatiles - The Devil in Binary Form

- Volatiles:
  - Compilers can kill you!
  - See why the compiler does this?

Volatile – Use with Caution!

```
int send(void) {
  dev_out='a';
  dev_out='b';
  dev_out='c';
}
```

```
int send(void) {
  dev_out='c';
}
```

Writing Great Code

Volatiles - The Devil in Binary Form

Example Miscompilations

MSP430 compiler:

```
extern volatile int WATCHDOG;

void reset_watchdog() {
  WATCHDOG = WATCHDOG; /* load, then store */
}
```

```
reset_watchdog:
        ret
```

Writing Great Code

ISRs - The Devil's Twin Brother

- ISRs: As short as possible
  - Analyze safety carefully
  - Ex: Skips restarting every 2-3 hours?
- Be ATOMICALLY SAFE!

```
... some code ...
ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
{
    SomeValue = SharedValue;
}
... now continue!
```

Writing Great Code

SKIP LINE

ISRs - The Devil's Twin Brother

- Reentrancy
  - Be "reentrant safe"
  - Good design practices are the best defense against reentrant issues.
- Reentrant issues:
  - Failure once a week?
  - Almost impossible to reproduce!
- In a nutshell, a function is reentrant if:
  - If it uses all shared variables in an atomic way,
  - If it does not call non-reentrant functions
  - If it does not use the hardware in a non-atomic way

But note! Reentrancy is most commonly associated with ISRs or hardware, but that's not the only way!

Writing Great Code

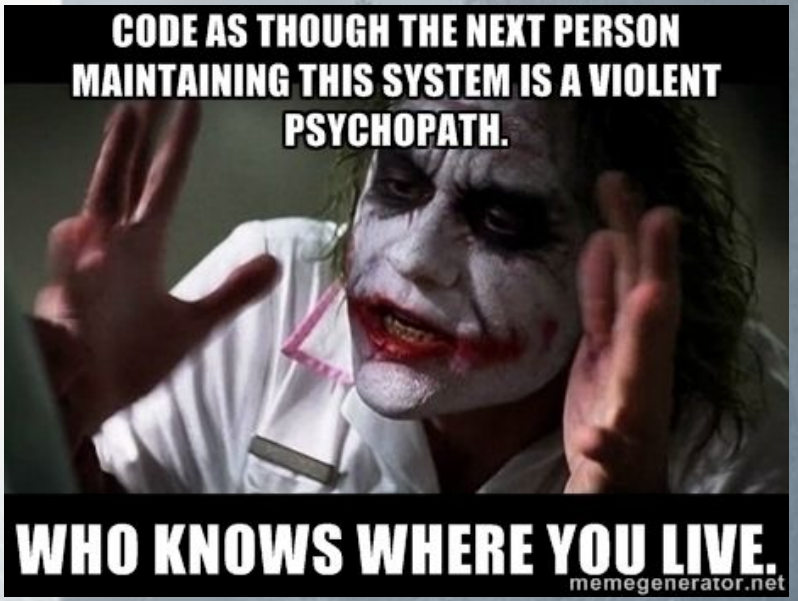
Bad Code - Actual Examples

You are not coding for yourself.

You are coding for the next person who will look at your code. Including your future self, who is not you. (Trust me.)

Always code under this premise!!

→



Writing Great Code

Bad Code - Actual Examples

Comment doesn't match value.
Writer obviously lacking understanding - hex is so you can see the binary value with fewer chars!

```
#define PRINTER_UNIT_ID                 0x05      // 0001 0101
```

Writing Great Code

Bad Code - Actual Examples

Commented-out code left in place.

No actual explanatory comments.

Magic numbers.

Why not just use strcpy? Padding LOC?

Variable naming: This code has a variable “lf”.

```
pDefFont->GetLogFont(&lf);
//lf.lfHeight = -70;
//lf.lfHeight = ((m_pTruck->WindowsPrinterType() == 2) ? -(int)(((float)
nPPIy / 8.58f) + 0.5f) : -90);
lf.lfHeight = ((m_pTruck->WindowsPrinterType() == 2) ? -32 : -90);
lf.lfWidth = 0;
//lf.lfWeight = 700;

lf.lfPitchAndFamily = 34;
lf.lfFaceName[0] = 'M';
lf.lfFaceName[1] = 'S';
lf.lfFaceName[2] = ' ';
lf.lfFaceName[3] = 'S';
lf.lfFaceName[4] = 'a';
lf.lfFaceName[5] = 'n';
lf.lfFaceName[6] = 's';
lf.lfFaceName[7] = ' ';
lf.lfFaceName[8] = 'S';
lf.lfFaceName[9] = 'e';
lf.lfFaceName[10] = 'r';
lf.lfFaceName[11] = 'i';
lf.lfFaceName[12] = 'f';
lf.lfFaceName[13] = 0;

// MS Sans Serif
```

Writing Great Code

SKIP LINE

Bad Code - Actual Examples

Make sure you don't work for someone like this!

Force yourself to just do the right thing - because it's the right thing!

Writing Great Code

Bad Code - Actual Examples

Interrupt handler complexity:

The handler itself looks simple...until you look at it and all the functions it calls.

Pragma that disables warning about an overly-complex function.

Code Complexity: > 300 code paths through one function … that happened to be an ISR.

Impossible to test, impossible to inspect, impossible to maintain.

Writing Great Code

Last Point - Allow overhead!

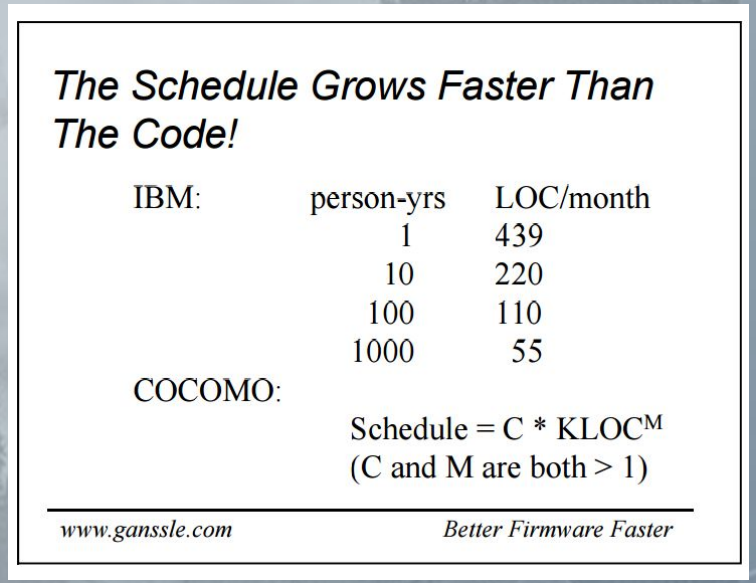
- If you run low on resources, you're screwed!
  - Memory
  - Flash space
  - I/O
  - PCB Real Estate
  - Etc.

Teamwork

Teamwork vs Coding Efficiency

The more people you add to a late project, the later it gets.

- Fred Brooks



Teamwork

Teamwork vs Coding Efficiency

- Always use version control!
- Good communications is critical. Use a tool to help.
- Code Review: Imperative for quality. If no one else available, give it a week, then review it yourself.

The Schedule Grows Faster Than The Code!

IBM:	person-yrs	LOC/month
	1	439
	10	220
	100	110
	1000	55

COCOMO:

Schedule = C * $KLOC^M$
(C and M are both > 1)

www.ganssle.com *Better Firmware Faster*

Thanks!