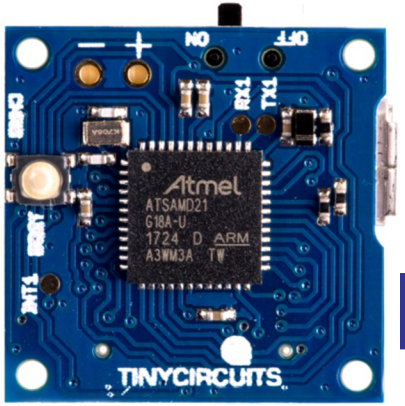
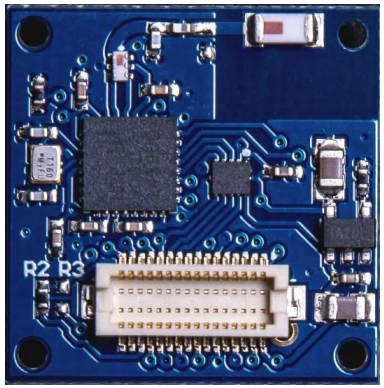
CSE190 Fall 2022



Lecture 9

Debugging Embedded Systems



Wireless Embedded Systems

Aaron Schulman

Hardware is hard.

Who do you trust when things are going wrong?

- Your software?
- Your users?
- Your test equipment?
  - Your test equipment configuration?
- The device designer?
  - The datasheet authors?
- The circuit board designer?
  - The manufacturer and assembler?
- …
- Physics?

Data sheet errors are common

Which one of these is correct?
SAM21 data sheet in 2017, and 2018 for I2C

2017

28.10.9 Address

Name: ADDR
Offset: 0x24
Reset: 0x0000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	LEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TENBITEN	HS	LENEN			ADDR[2:0]		
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

2018

28.10.9 Address

Name: ADDR
Offset: 0x24
Reset: 0x0000
Property: Write-Synchronized

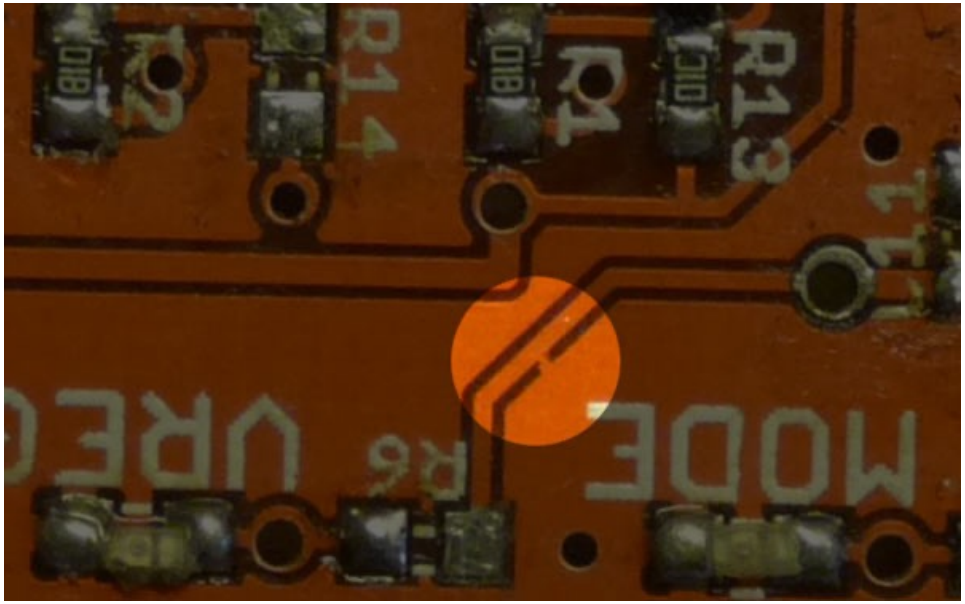
Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	LEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

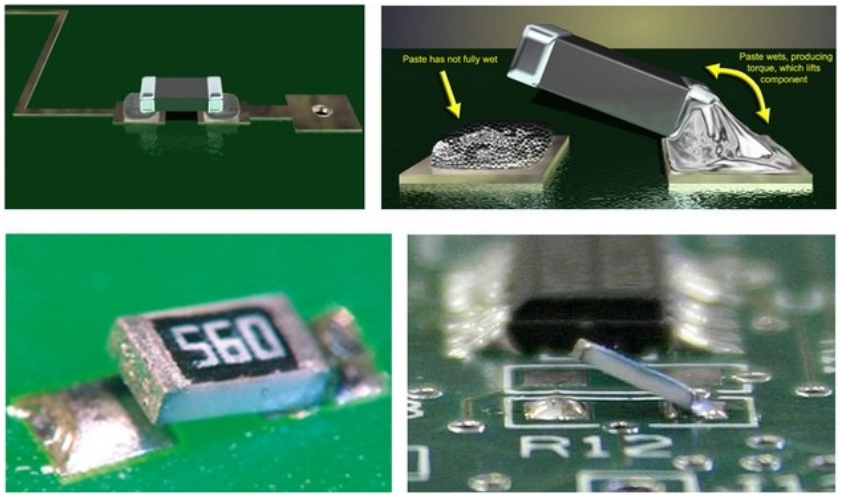
Bit	15	14	13	12	11	10	9	8
	TENBITEN	HS	LENEN			ADDR[10:8]		
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Manufacturing errors are common



Circuit board manufacturing fault (signal shorted to GND)



PCB assembly fault (“tombstoning”) resistors

When all else fails, the bootloader can save the day!

When your software has a bug in it that causes the processor to hang or crash, you can't do anything!

- Programming over USB requires the processor to be responsive (a reset message is sent over USB at the beginning of the programming)
- The bootloader is a small bit of code run at boot time that doesn't get overwritten
  - It has just enough code in it to reflash over USB
  - Normally it just jumps into your code, but you can force it to wait for a program flash switching the power on while you hold the reset button.

https://github.com/arduino/ArduinoCore-samd/tree/master/bootloaders/zero

Even one LED can be critical for debugging

Think of an LED as a breakpoint (very efficient!)

- Turn it on before a line of code
  - Lets you know you reached that line
- Turn it off after a line of code
  - Lets you know you exited that code segment
- Toggle it each time a repeated event is called
  - This can show you visually how often the event happens

Need some way to introspect code: printf() is essential

Think of prints like tracing + inspection (not efficient)

- Print variables you need to inspect
  - Register states (flags, sensor readings)
- Print when a specific function is run
  - Lets you know the order functions are run
  - Can you use printf for interrupt handlers?