uC101: Introduction to Microcontrollers / Interfacing with the real world

Josh Johnson

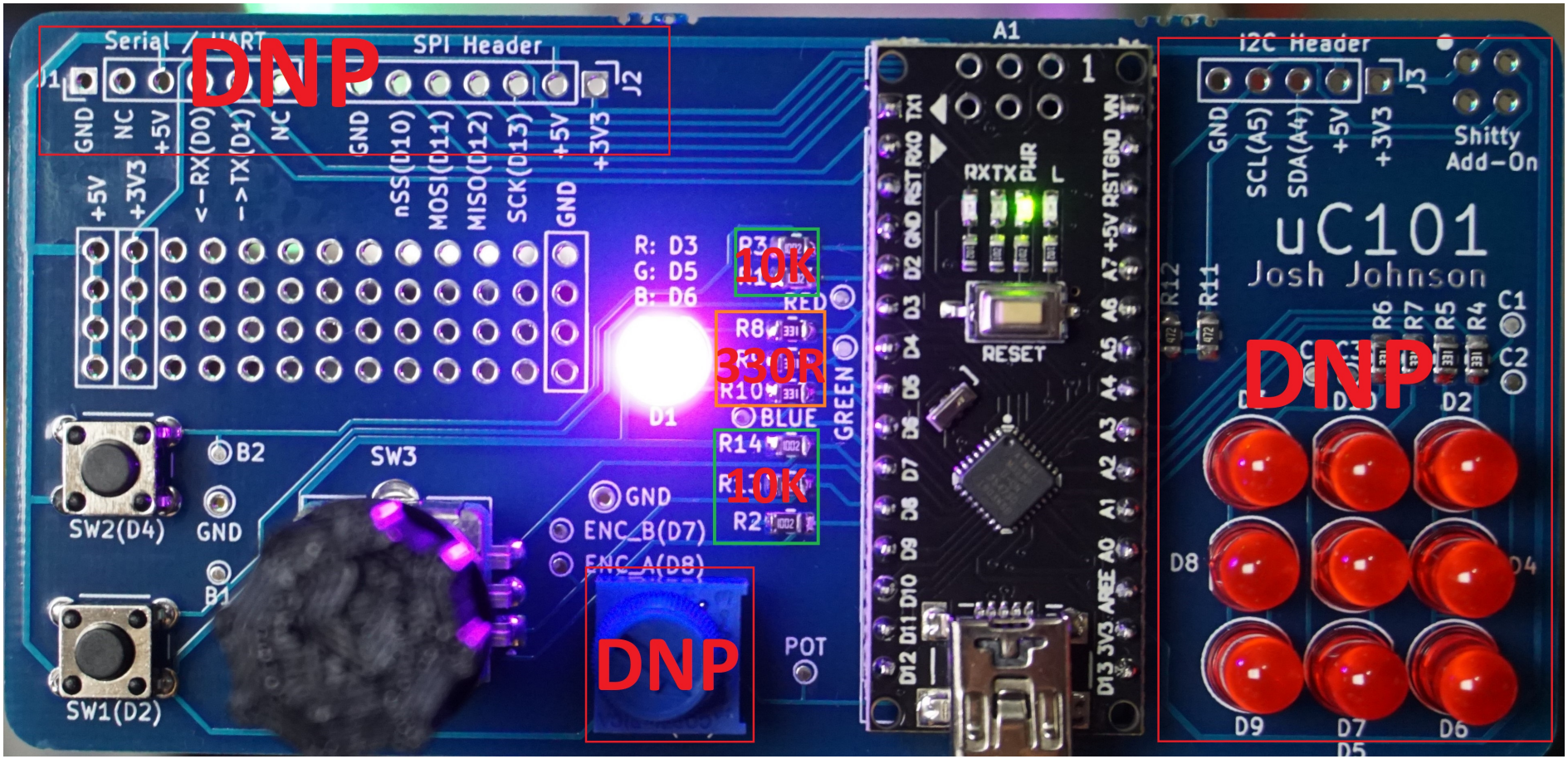
13/5/2019

Overview

- Assembly of Hardware / Installation of Software
- Microcontroller 101
- Tools
- Bit Math
- Demos
  - Blink
  - Button
  - RGB LED (PWM)
  - Rotary Encoder
  - UART
  - Charlieplexing

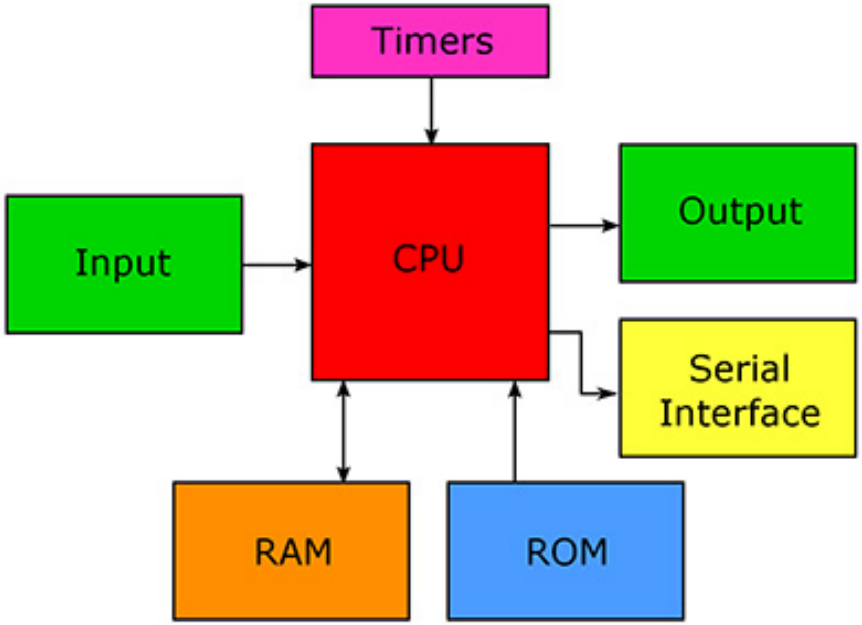
Project Files: `github.com/joshajohnson/CBRhardware`

Assembly of Hardware

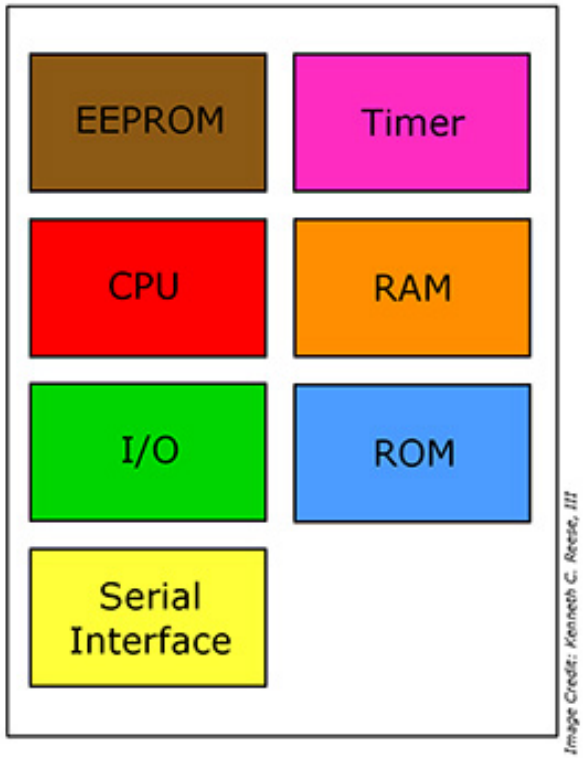


What is a microcontroller?

Microprocesser: CPU
and several supporting chips.



Microcontroller: CPU
on a single chip.



Microcontroller 101

Common Options

8 bit

- ATtiny
- ATmega (Atmel / Microchip)
- PIC (Microchip)

16 bit

- MSP430 (TI)

32 bit

- STM32 (ST)
- SAM (Atmel/Microchip)
- nRF5x (Nordic Semi)
- ESP8266/32 (Espressif)
- CCxxxx (TI)
- LPCxxxx (NXP)
- PIC32 (Microchip)

32 bit ARM cores

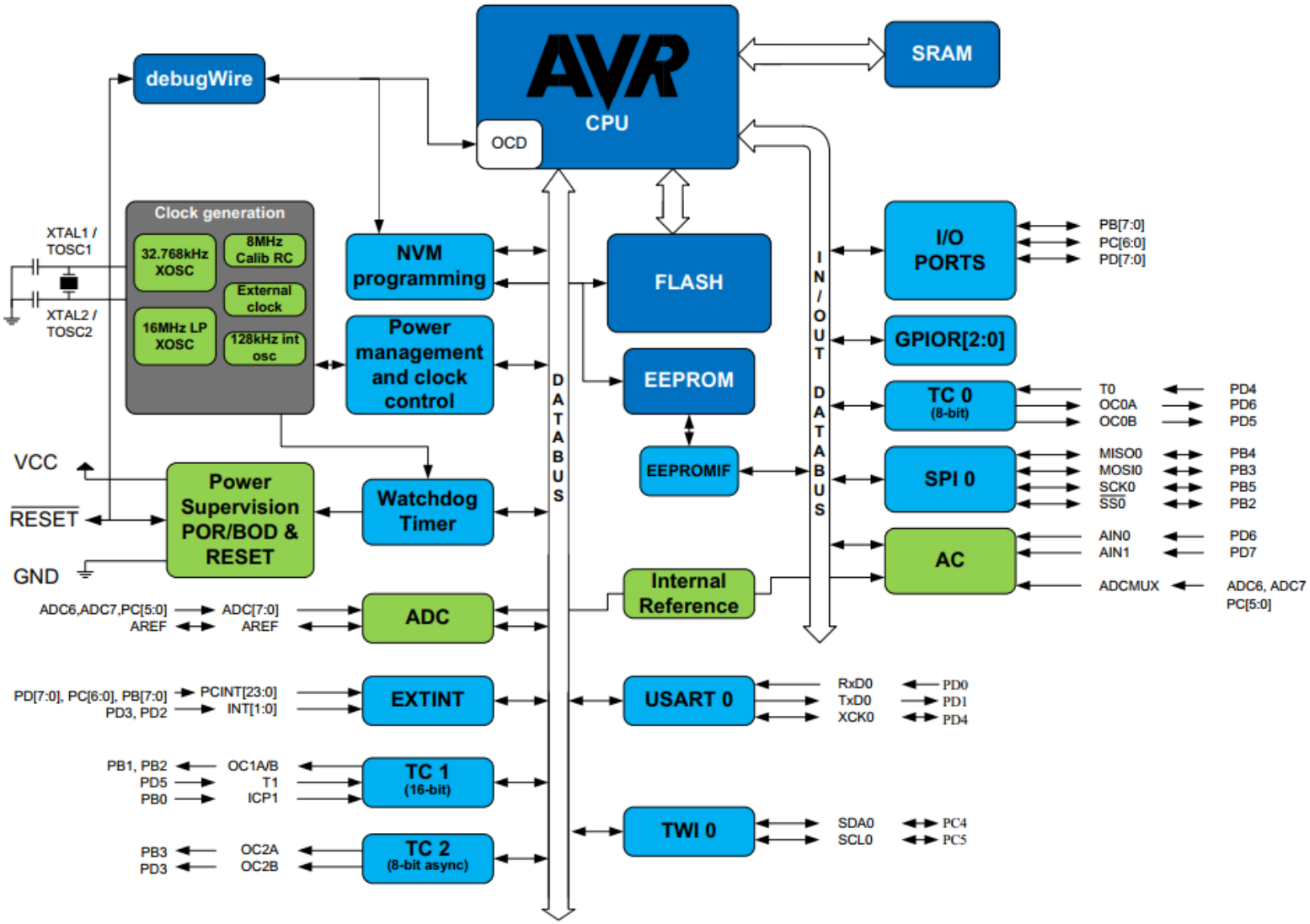
- Cortex-M0/M0+
- Cortex-M1 (FPGA only)
- Cortex-M3
- Cortex-M4 (M3 + DSP + FPU)
- ...

Microcontroller 101

How to choose?

- Compute power
  - 8 bit vs 32 bit
  - DSP / FPU
- Peripherals
  - Wireless
    - WiFi
    - Bluetooth
    - LoRa
    - Cellular
  - USB
  - ADC
  - Ethernet
  - CAN
  - Number of SPI/UART/I2C/Timers

ATmega328p Architecture

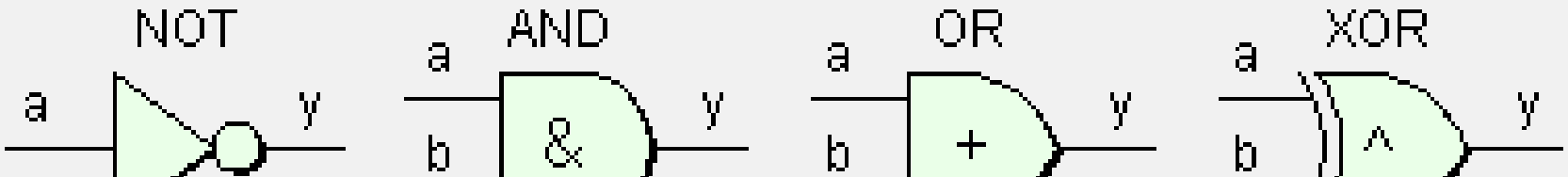


Datasheet time!

Tools

- Oscilloscope
  - + High quality time domain measurements (500 MSPS - 256 GSPS)
  - - Expensive, short record time
- Logic Analyser
  - + Easy to use UI, long sample time / depth
  - - 1 bit*, low sample rate (10 - 100 - 500 MSPS)
  - * some logic analysers have limited analog capabilities
- Debugger
  - Program and debug device
- Multimeter
  - Voltage and resistance measurements
- Development Board
  - Whilst not a tool, extremely helpful whilst bringing up firmware
  - Helpful when developing code - the issue isnt (typically) the hardware!

Bitwise Operators



a	y
0	1
1	0

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

```
y = ~a        y = a & b      y = a | b    y = a ^ b
              NOT y = a && b
```

Bit Shifting

```
a = 5;          // binary: 00000101
b = a << 3;     // binary: 00101000, 40 in decimal
c = b >> 3;     // binary: 00000101, back to 5 like we started
```

Putting it all together

```
x = 1;          // binary: 00000001
x <<= 3;        // binary: 00001000
x |= 3;         // binary: 00001011 - because 3 is 11 in binary
x &= 1;         // binary: 00000001
x ^= 4;         // binary: 00000101 - binary mask 100
x ^= 4;         // binary: 00000001 - toggle again
```

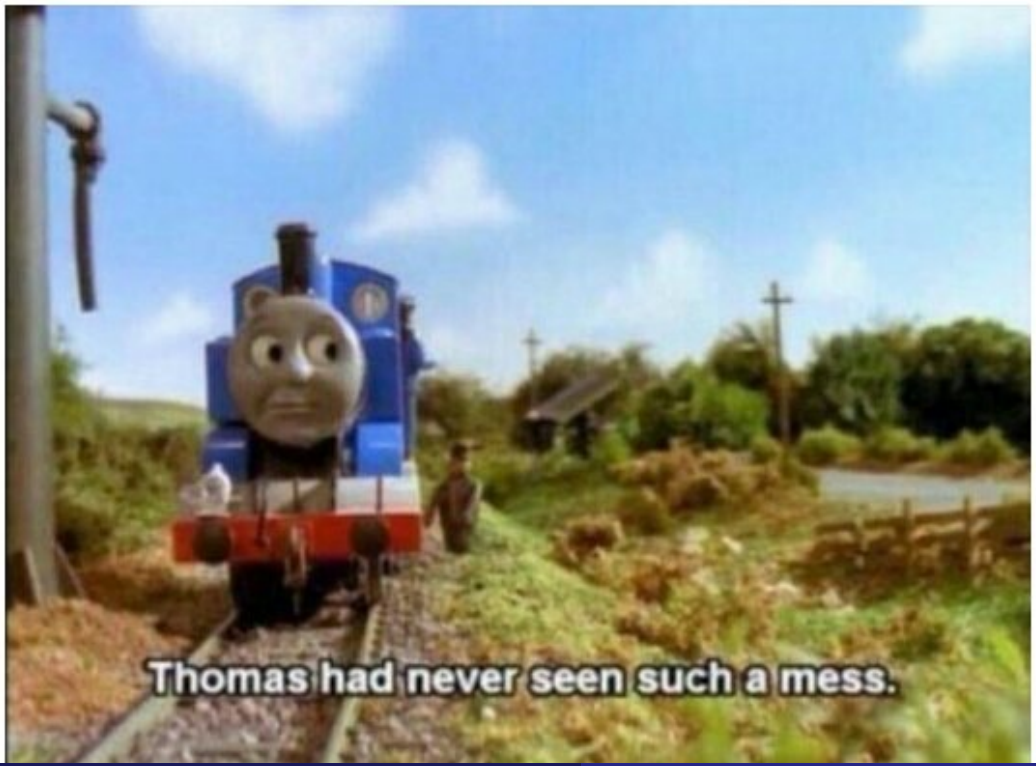
Demo Time!

Copy the uC101Library folder to `Documents/Arduino/libraries`
Open `Firmware/blink/blink.ino`
In Arduino IDE:
`Tools->Board:-> Arduino Nano`
`Tools->Processor:-> ATmega328p`
`Tools->Port:-> $comPort`
Run blink.ino and confirm that the onboard LED is blinking

Demo Time!

Me : *Explain my code to colleague*

Colleague:



- I'm a hardware person
- Not a software person
- Solder is my preferred programming language
- I know enough to be dangerous, nothing more
- More experienced folks, please jump in and correct me / answer questions I can't / point out bad practices
- I'm here to learn like everyone else

The End

Links to resources: `uC101/README.md`

Next Month

- Breadboard to Printed Circuit Board
- Mechanical Design Considerations

Say Hello!
BSidesCbr Slack: josh
Twitter: @_joshajohnson
Email: josh@joshajohnson.com

Project Files: `github.com/joshajohnson/CBRhardware`