Low Level Arduino Programming

Making Things Faster & More Versatile

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Jan Burchard, Tobias Schubert, Bernd Becker

Computer Architecture

High vs. Low Level

- Thus far:
  - Programming in Arduino (C)
  - With functions provided by Arduino
  - Simple
  - Limited
  - Possibly slower

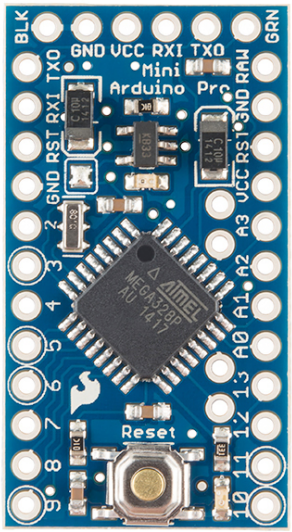
High vs. Low Level

- Thus far:
  - Programming in Arduino (C)
  - With functions provided by Arduino
  - Simple
  - Limited
  - Possibly slower
- Today:
  - Programming in Arduino (C) and assembler
  - Low level functions
  - Direct access to microcontroller registers
  - Harder
  - More possibilities
  - Potentially faster

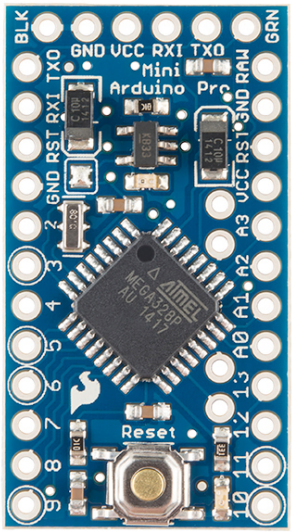
High vs. Low Level

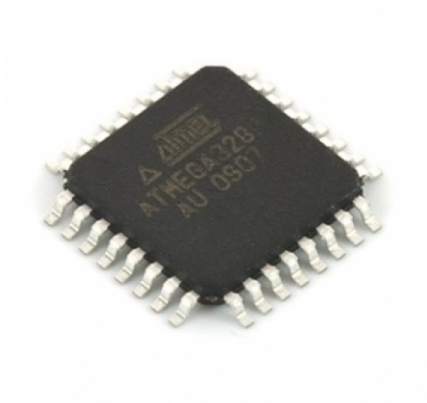
- Thus far:
  - Programming in Arduino (C)
  - With functions provided by Arduino
  - Simple
  - Limited
  - Possibly slower
- Today:
  - Programming in Arduino (C) and assembler
  - Low level functions
  - Direct access to microcontroller registers
  - Harder
  - More possibilities
  - Potentially faster
- Also: How to extract information from a 650 page datasheet

A Low Level Look onto the Arduino

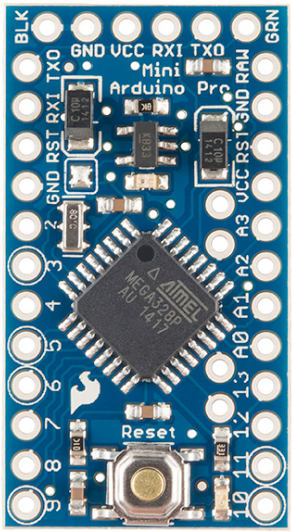


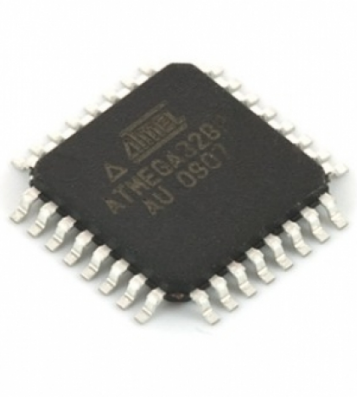
A Low Level Look onto the Arduino

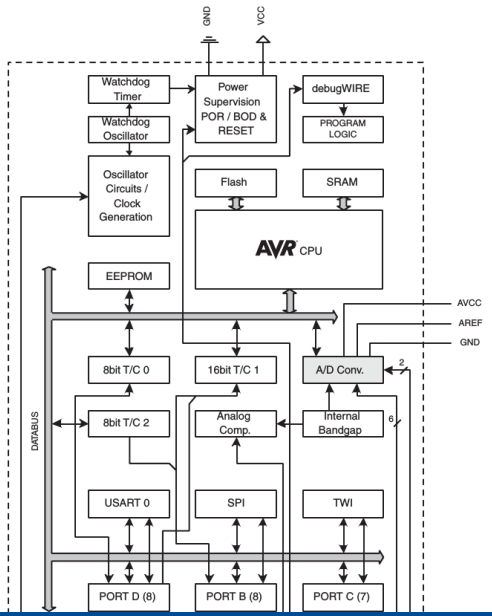


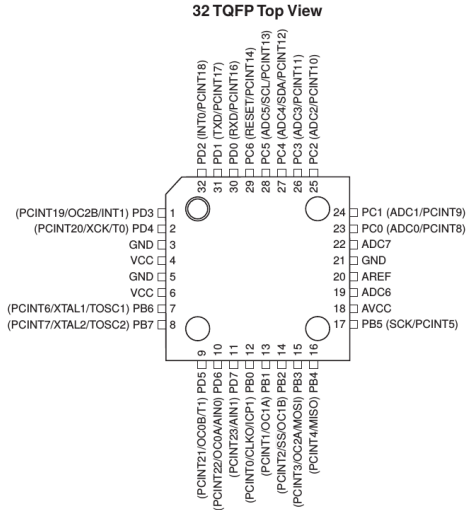


A Low Level Look onto the Arduino

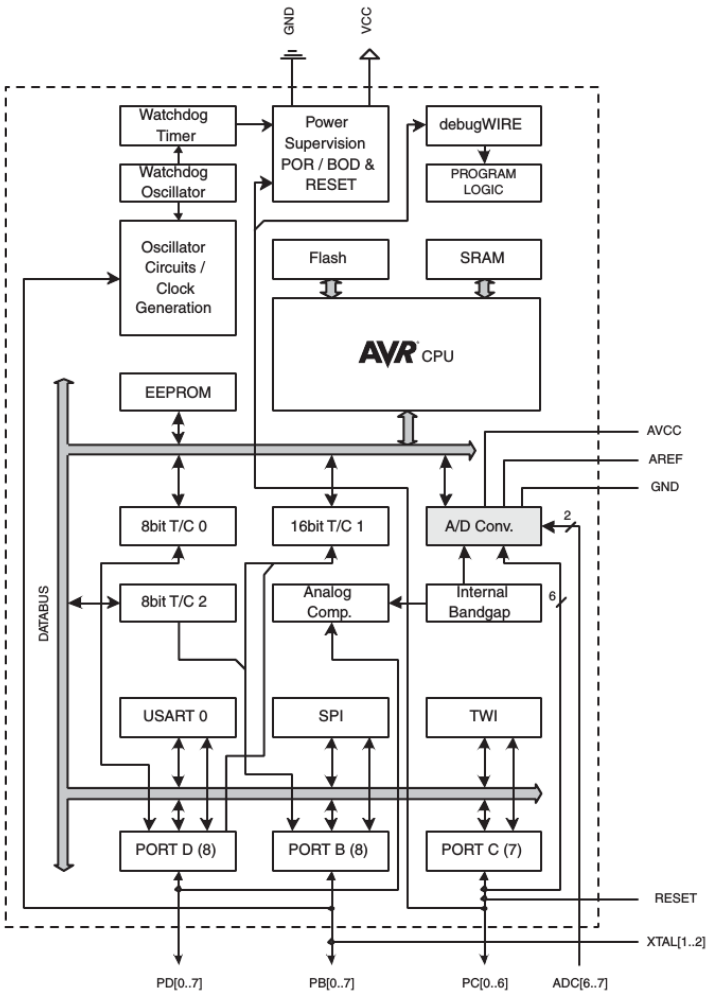




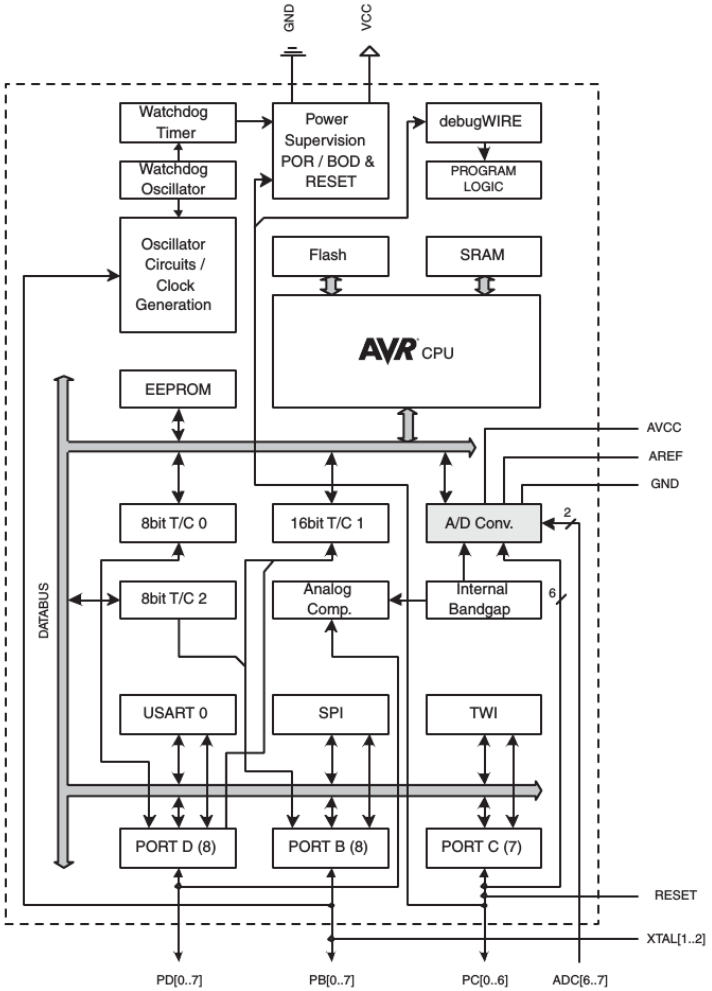


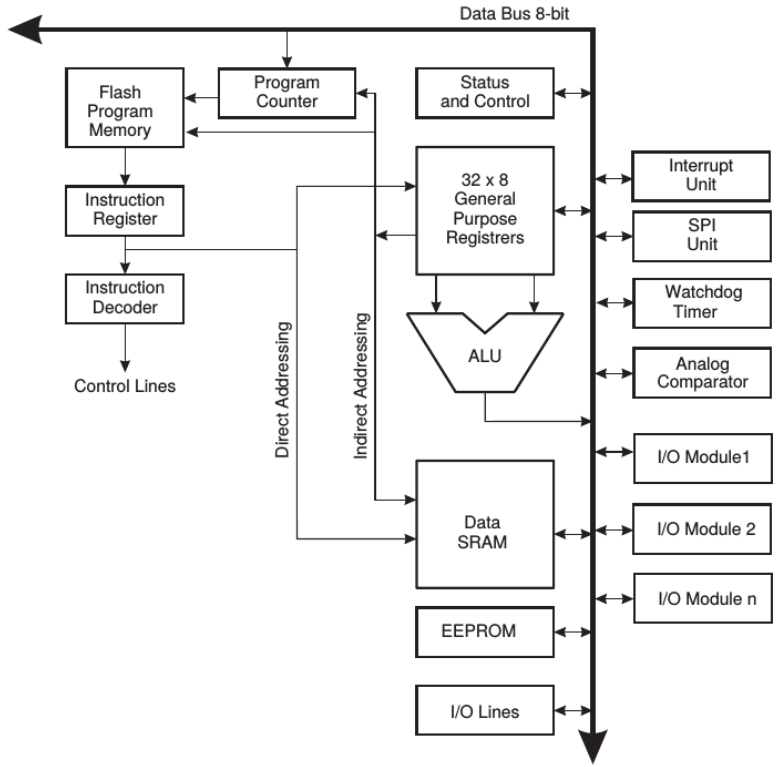


An Even Closer Look



An Even Closer Look





Registers

- 8 bit registers:
  - Status and control registers
  - General purpose registers
- Accessing registers allows for direct control
- Setting a single bit in a 8 bit register R:
  - Set the fifth bit to 1: $R \mid = B00100000;$
  - With shift: $R \mid = (1 << 5);$
  - Set the fifth bit to 0: $R \,\&= B11011111;$
  - With shift: $R \,\&= \sim(1 << 5);$
- Don't overwrite any other bits by accident:
  - $R = B00100000$ sets all 8 bits!

The Datesheet

- 650 pages full of information
- 41 chapters
- Table of contents in the **back**
- Explains *everything*!
  - Registers (with initial values)
  - Setup of all parts of the controller
  - Assembler commands
- In combination with Arduino: Do not trust initial values in control registers

Pin Toggling

- How to get a pin to 1 or 0?

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff
  - Register: PORTxn

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff
  - Register: PORTxn
  - Arduino “Pin 13” is actually PORTB5

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff
  - Register: PORTxn
  - Arduino “Pin 13” is actually PORTB5

  → Register: PORTB, 5th bit

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff
  - Register: PORTxn
  - Arduino “Pin 13” is actually PORTB5

  → Register: PORTB, 5th bit
- How about only toggling the pin?
  - Register: PINB, 5th bit

Pin Toggling

- How to get a pin to 1 or 0?
- Thus far: “digitalWrite(pin, VALUE);”
  - Slow
- Low level: write to output register
  - I/O Ports: Datasheet page 75ff
  - Register: PORTxn
  - Arduino “Pin 13” is actually PORTB5

  → Register: PORTB, 5th bit
- How about only toggling the pin?
  - Register: PINB, 5th bit
- Very low level: use assembler
  - “asm volatile (commands : outputs : inputs);”

PWM

- Toggle a pin with a specific frequency

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);
  - Does not work on all pins

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);
  - Does not work on all pins
- Idea 1: write you own

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);
  - Does not work on all pins
- Idea 1: write you own
  - Works
  - . . . but not in the background

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);
  - Does not work on all pins
- Idea 1: write you own
  - Works
  - . . . but not in the background
- Low level: use timers

PWM

- Toggle a pin with a specific frequency
- Thus far: analogWrite(pin, VALUE);
  - Does not work on all pins
- Idea 1: write you own
  - Works
  - . . . but not in the background
- Low level: use timers
  - Atmega328: 3 different timers available
  - Timer 0 is used for millis(), etc. $\rightarrow$ do not use
  - Let's use timer 2 (datasheet page 141ff)

Timer 2 (and other timers)

- Counter that is incremented in the background
  - Mode: normal, PWM, CTC . . . (page 146ff)
  - Register: TCCR2A + TCCR2B (Timer/Counter Control Register A + B)
  - Bits: WGM22, WGM21, WGM20

Timer 2 (and other timers)

- Counter that is incremented in the background
  - Mode: normal, PWM, CTC ... (page 146ff)
  - Register: TCCR2A + TCCR2B (Timer/Counter Control Register A + B)
  - Bits: WGM22, WGM21, WGM20
- Configurable count frequency
  - Prescaler: Divide input clock by specific value (page 156)
  - Register: TCCR2B (Timer/Counter Control Register B)
  - Timer can count to 255, so frequency is $\frac{8\,MHz}{Prescaler \cdot 256}$

Timer 2 (and other timers)

- Counter that is incremented in the background
  - Mode: normal, PWM, CTC ... (page 146ff)
  - Register: TCCR2A + TCCR2B (Timer/Counter Control Register A + B)
  - Bits: WGM22, WGM21, WGM20
- Configurable count frequency
  - Prescaler: Divide input clock by specific value (page 156)
  - Register: TCCR2B (Timer/Counter Control Register B)
  - Timer can count to 255, so frequency is $\frac{8\,MHz}{Prescaler \cdot 256}$
- Can trigger an action when a specific value is reached
  - Triggered when value in OCR2A or OCR2B is reached
  - Interrupt (Register: TIMSK2)
  - Toggle Pin (Register: TCCR2A, Bits: COM2A1, COM2A0, COM2B1, COM2B0)

Interrupts

- Tell the microcontroller that an (external) event has occured
- On an interrupt an interrupt service routine (ISR) is called
- Stops the current computation until ISR is finished

$\Rightarrow$ Make ISRs small and fast!

- ISR are distinguished by ISR vectors
- List of vectors can be found on the Internet, link in the forum
- Disable all interrupts with “cli();” before changing any interrupt settings, re-enable with “sei();”

Back to the PWM

- We need:
  - A timer which counts from 0 to 255 at a given frequency
  - An interrupt which is triggered at *value* to make the pin 1
  - An interrupt for an overflow the make the pin 0 again
  - ISRs for the pin toggling

What else can we do with timers?

- Toggle a pin with a **specific** frequency
  - Reset the timer on a compare match
  - ISR: Toggle pin on compare match

What else can we do with timers?

- Toggle a pin with a **specific** frequency
  - Reset the timer on a compare match
  - ISR: Toggle pin on compare match
- Trigger things regularly, e.g., an ADC conversion

ADC

- Converts a voltage (0-3.3V) into a digital value
- Datasheet: page 237ff
- Modes: Single conversion, auto triggered, free run

1. Select the channel & reference (Register: ADMUX)
2. Select prescaler, and enable (Register: ADCSRA)
3. Trigger conversation (ADSC)
4. Finished when ADIF is high
5. Read results from ADCL and ADCH (ADCL first!!!)

Image Sources

- Slide 2: `https://www.sparkfun.com/products/11114` (Abgerufen: 19.12.2016)
- Slide 2: `https://www.sparkfun.com/products/retired/9261` (Abgerufen: 19.12.2016)
- Slide 2: Atmel ATmega328/P DATASHEET COMPLETE (Atmel-8271I-AVR- ATmega-Datasheet_10/2014)
- Slide 3: Atmel ATmega328/P DATASHEET COMPLETE (Atmel-8271I-AVR- ATmega-Datasheet_10/2014)