GlobalLogic®

# Basecamp: C/Embedded

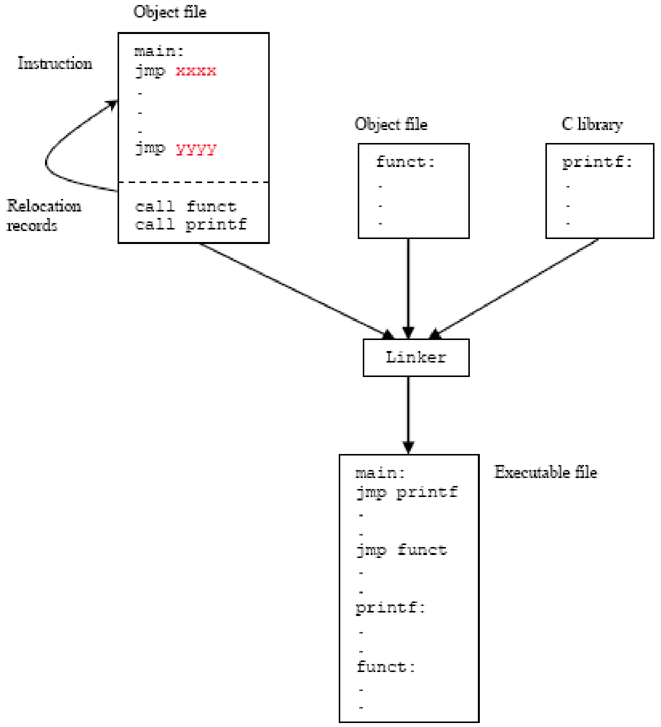
12

## Agenda

1. Homework review
2. gcc-avr
   - Makefile
3. UART
4. PWM
5. Practice

# Linking process

- C toolchain consists of several utilities
- Source files are compiled to the object files with a corresponding sections
- Object files are linked together in order to merge sections as well as update calls to other object files
- Linker produces executable file
  - For Linux it's ELF format
  - readelf utility
- Executable file is loaded with OS loader
  - MCUs don't have loader, therefore ELF should be converted to the firmware: the bunch of properly placed sections
  - Firmware is loaded with MCU at the start
    - actually, bootloader is passing control to the firmware



## Toolchain

- Arduino IDE uses gcc toolchain tailored for 8-bit MCUs
- Arduino uses avrdude bootloader, so you can upload your firmware from the CLI
- http://www.nongnu.org/avr-libc/
- Regular Arduino setup is the following:
    - Call to internal init function
    - Call to user-defined setup function
    - In the infinite loop it calls for loop function

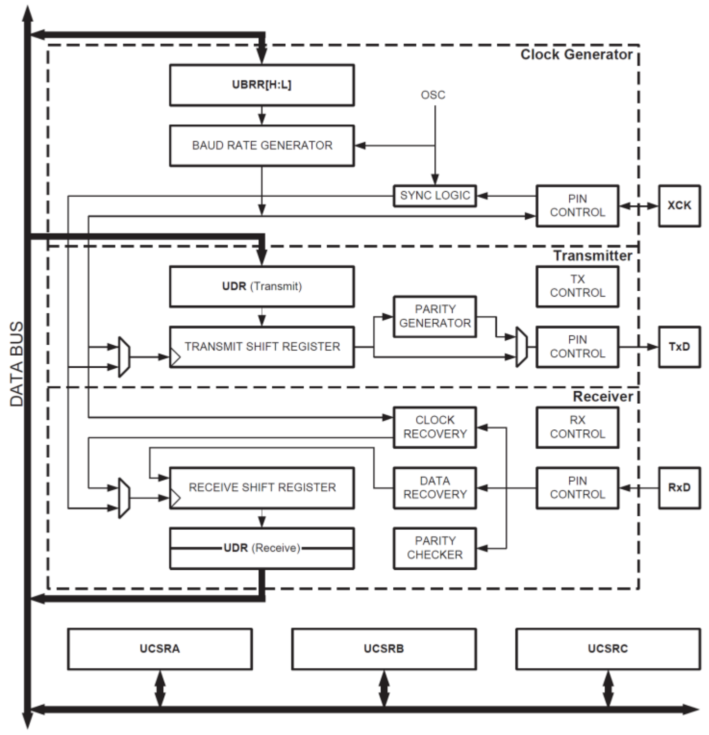
```
sudo apt-get install gcc-avr avrdude

avr-gcc -Wall -Os -DF_CPU=16000000UL \
    -mmcu=atmega2560 main.c
avr-objcopy -j .text -j .data -O ihex a.out \
    main.hex
avr-size -format=avr --mcu=atmega2560 main.hex

avrdude -v -patmega2560 -cwiring -P/dev/ttyACM0 \
    -b115200 -D -V -U flash:w:main.hex:i
```

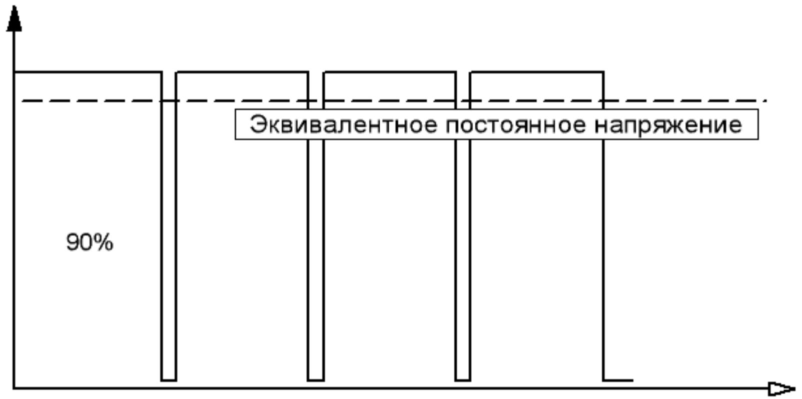
# U(S)ART

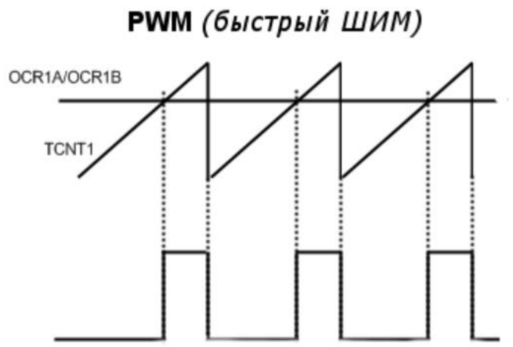
- UART is universal asynchronous receiver-transmitter
- UART is feed with its own clock generator
  - You need to properly set the baudrate in UBRR register
- Control registers are UCSRA/B/C
  - In order to get access to synchronous operation or high speeds at UART, you need to configure properly UCSRnA
  - You need to enable UART transmission/receiving in UCSRnB register
  - Frame configuration is done in UCSRnC
- UDR is used both for TX and RX
- UCSRnA has flags for frame errors and completion status

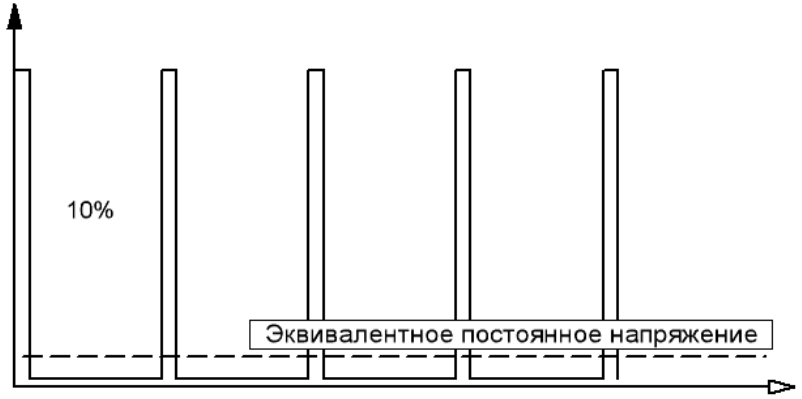


## PWM

- AVR8 MCUs have several PWMs: fast, phase-correct
- PWM is coupled with timers







## Practice agenda

1. Write the Makefile, that compiles empty main function
2. Rewrite the last example to the gcc-avr style
   - You will loose access to Arduino framework, including Serial.println()
3. Write UART handling routine via direct access to UDR
4. Connect encoder as a button

## Connect everything together

1. Use encoder button via interrupt in order to disable LED
2. Use encoder in order to regulate PWM
3. Use sleep mode of the MCU in order to powersave
   - Explore more power saving settings and measure the power consumption
4. Write with regular gcc-avr
   - Code should be clean