Eduino
A better ATmega32U4 breakout board

doc version 31, February 10, 2023

The goal of this project was to come up with a replacement for the campus lab exercises in the courses 1TE663 and 1TE723 on microcontroller programming at Uppsala University during the Corona fall semester of 2020.

Contents

1 Background 2
2 Design phase 2
3 Design challenges 2
4 Parts 4
5 Architecture 6
6 Bootloader 7
7 CDC serial port 7

1 Background

During the history of the course I have migrated from using an Atmel STK500 to using bare ATmega328 chips on solderless breadboards together with an USB-attached serial programmer. Originally a commercial programmer before I introduced my own design which plugged right into the power rail of the breadboard.

However, getting everything to work for all students even in on-site labs was not easy:

- the necessary drivers would repeatedly disappear from Windows installations
- there was no direct way of communicating back from the program code on the microcontroller to the host computer without additional hardware
- wiring 5 cables between the programmer and the microcontroller could be done in 120 different combinations, of which only one is correct

When I got the idea of home-labs early summer 2020 I started looking into the *Pro Micro* (see Fig. 1) platform, originally developed by *Spark Fun*, based on the *Arduino Leonardo*. I liked the idea of utilizing a microcontroller with built-in USB interface which can directly connect to any modern operating system since it can emulate a CDC-type serial interface which does not need any additional drivers under either Windows, MacOS or Linux.

However, I found the *Pro Micro* itself quite unattractive for the course. The reasons can be found in the schematic diagram of the *Pro Micro* (see Fig. 2):

- out of the 26 GPIO pins of the ATmega32U4 (see Fig. 3) only 18 are available
- no complete 8-bit port is available, since PB0 and PD5 are misused for one of the on-board LEDs, and PB7 and PD6 are "not used"
- the SPI-interface is not fully functional, because pin SS or PB0 is used for the LED
- all pins are more or less randomly placed on the pinheaders

2 Design phase

Designing the circuit board for the Eduino took about 2 days during my summer vacation. I had the goal to

- make all 8 pins of ports `B` and `D` available, like on the ATmega328
- sort the pins according to their bit-values in the ports
- make as many of the other pins available as possible
- use 3.3×10^0 V as the main supply voltage for compatibility with modern peripherals
- don't hide anything from view and use a clean layout with labels for all parts

The design was done in KiCAD v5 and I took the opportunity to order a few boards both assembled and unassembled from *PCBgogo* in Shenzhen in China. It was my first order of preassembled boards and I wanted to see what they could offer and how much it would cost.

I made one mistake by not reading the datasheet of the ATmega32U4 too carefully. For 3.3×10^0 V operation the datasheet forbids the use of a 16×10^0 MHz quartz resonator but would allow the use of 12×10^0 MHz. So this was my choice, but I then had to figure out that the USB interface requires the use of either 8×10^0 MHz or 16×10^0 MHz. Therefore I had to replace the surface mounted quartz resonators (part Y1) from the first units.

3 Design challenges

Drawing up the schematic diagram of the circuit around the ATmega32U4 was not a big issue, see Fig. 6. I started with the USB interface and the power supply part. Since I wanted to have as many pins available as possible I decided to go with a single LED on the board apart from a power-indicator LED. Pin `PE6` is quite lonely on port `E` of the ATmega32U4 since the only other pin from port `E`, `PE2`, is signalling the use of the bootloader code after a reset. I would have to change the bootloader code to use a single LED on `PE6` then instead of using the two LEDs on `PB0` and `PD5`, but that can be easily done.

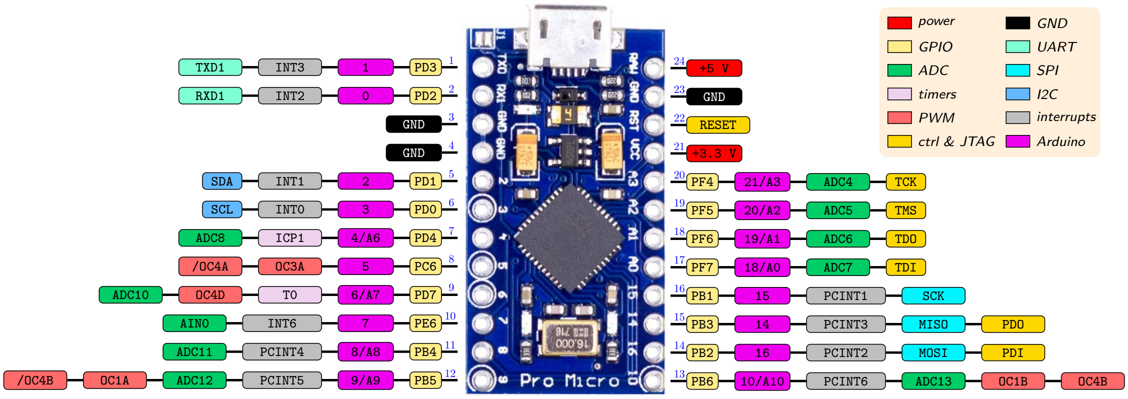


Figure 1: Pinout of the Pro Micro *by* Spark Fun.

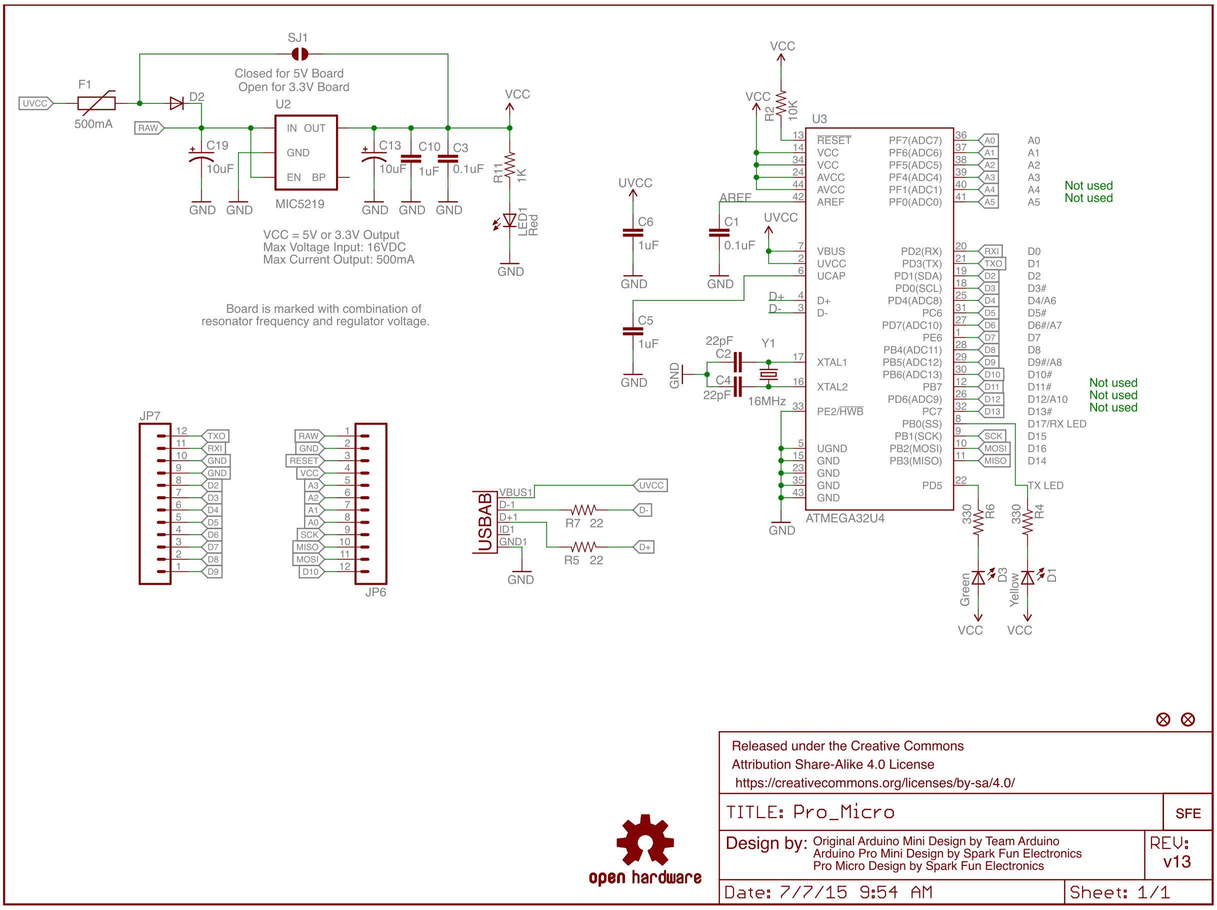


Figure 2: Schematic diagram of the Pro Micro *by* Spark Fun.

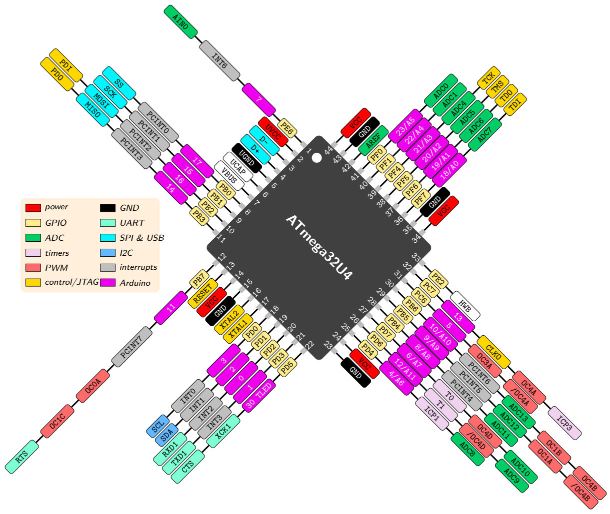
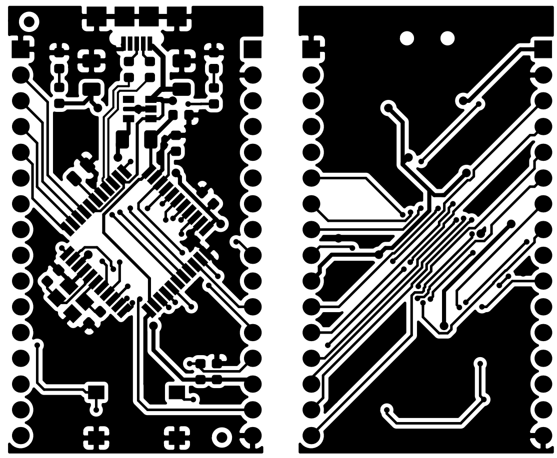


Figure 3: Pinout of the ATmega32U4 microcontroller.



front side back side

Figure 4: Copper layers of the breakout board. The USB connector is at the top and the back side is mirrored horizontally to line up with the front side.

The *Pro Micro* itself has two pin headers with 12 pins each. With 26 GPIO pins available for my design plus the need of power supply pins it was clear that I would need more pins. When laying out the circuit I ended up with two pin headers with 16 pins each for a total of 32 connection points.

The real obstacle came then when looking at the pinout of the ATmega32U4 itself, see Fig. 3. To my surprise the pins on the 44-pin package of the chip were by far not as nicely sorted as on an ATmega16, ATmega328 och ATmega644, to which I was used from the past. Just as an example, why are the pins from port `D` not sorted in order, but rather pins `PD4` and `PD5` are swapped? Why are three pins from port `B` on the wrong side of the package?

I understand that it most probably made the layout of the actual chip inside the package easier for the designers at *Microchip* (or at *Atmel* at that time), but really?

Still it was possible to untangle this maze on a two-layer PCB (see Fig. 4), but in order to make my own work not to complicated I decided to go for the maximum width of the board at 26×10^0 mm which would fit onto a standard solderless breadboard, leaving one column of holes accessible on each side. The final arrangement of the pins on the breakout board can be seen in Fig. 5.

4 Parts

Resistors

R1	$22 \times 10^0\ \Omega$	0603
R2	$22 \times 10^0\ \Omega$	0603
R3	10^0 kΩ	0603
R4	10^0 kΩ	0603
R5	10×10^0 kΩ	0603

Capacitors

C1	10×10^0 µF, 16×10^0 V	1206
C2	10×10^0 µF, 16×10^0 V	1206
C3	100×10^0 nF, 16×10^0 V	0603
C4	100×10^0 nF, 16×10^0 V	0603
C5	100×10^0 nF, 16×10^0 V	0603
C6	10^0 µF, 16×10^0 V	0805
C7	100×10^0 nF, 16×10^0 V	0603
C8	22×10^0 pF, 16×10^0 V	0603
C9	22×10^0 pF, 16×10^0 V	0603

Inductors

L1	10×10^0 µH	1206

LEDs

D1	LED green	0603
D2	LED red	0603

Integrated circuits

U1	ATmega32U4-AU	TQFP44
U2	MIC5219-3.3YM5	SOT23-5

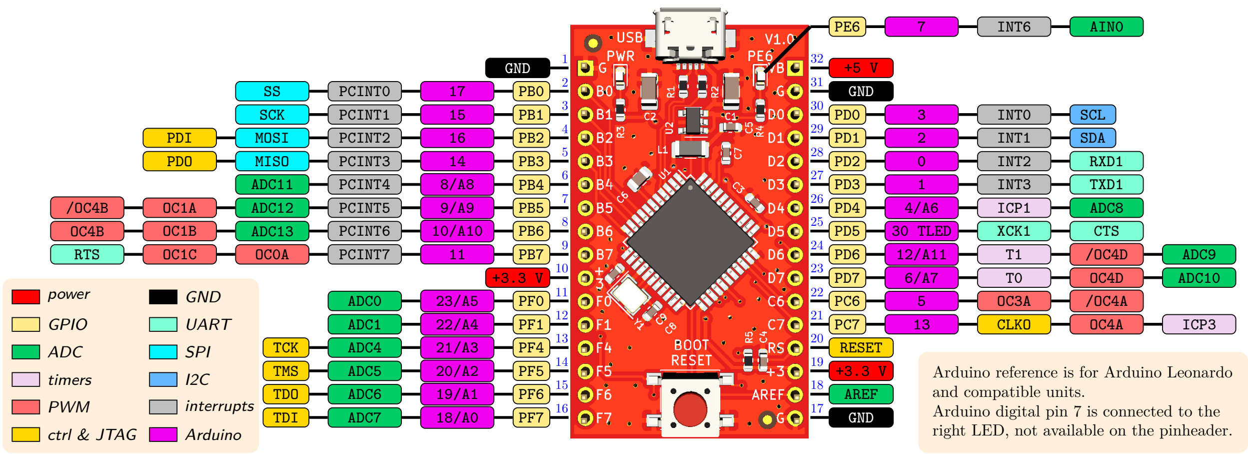


Figure 5: Pinout of the breakout board.

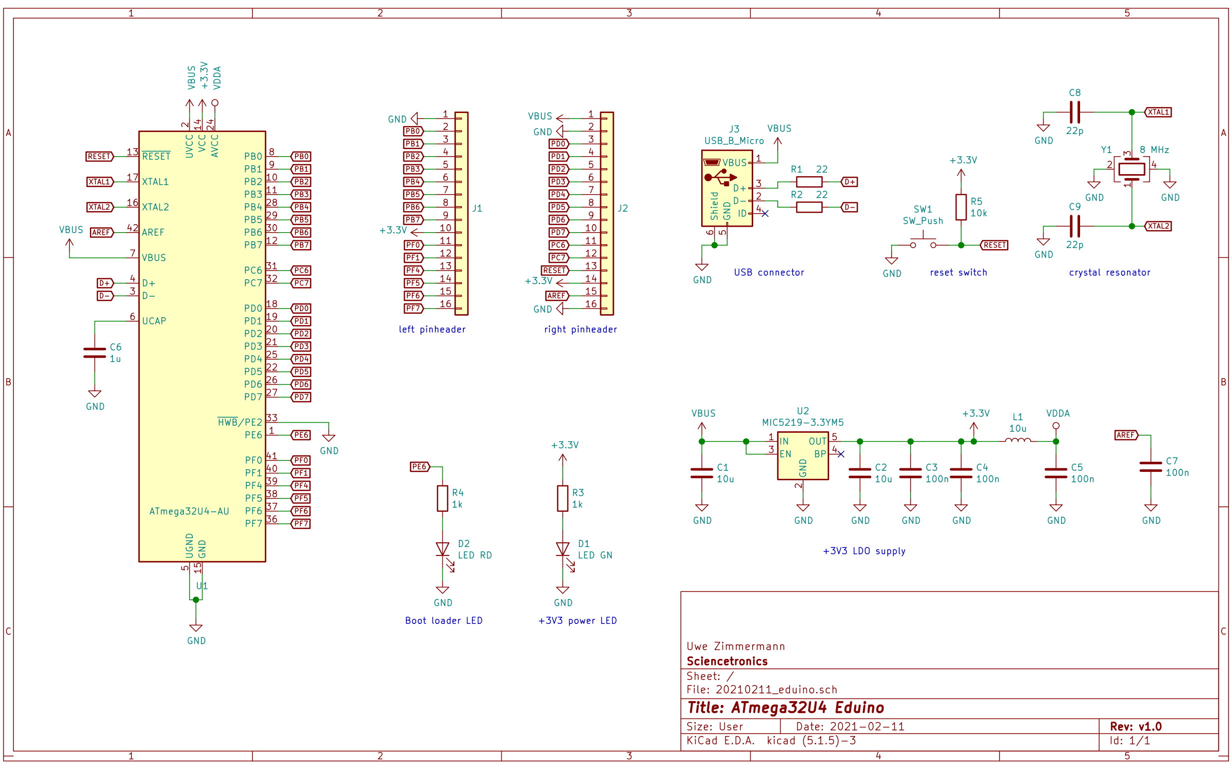


Figure 6: Schematic diagram of the breakout board.

Crystals		
Y1	8×10^{0} MHz xtal	3225-4
Electromechanical		
SW1	push button	6x6mm 4pin
J1	pinheader	16x2.56mm
J2	pinheader	16x2.56mm
J3	micro-USB	C40942

5 Architecture

5.1 USB connection

The ATmega32U4 contains all the hardware for a full speed USB-2 interface, including the differential drivers, supply voltage control, timing and pull-up resistors.

In order to use the hardware USB-interface, the microcontroller has to be clocked with 8×10^{0} MHz or 16×10^{0} MHz, an internal PLL (phase locked loop) generates the necessary timing for the data transfer.

5.2 Supply voltage

The breakout board is supplied via the USB cable from either an attached computer, a power bank or a wall adapter. The USB supply voltage of nominally 5×10^{0} V is directly available on pin 32 of the breakout board, with the corresponding GND at pins 1, 17 and 31.

The ATmega32U4 itself is supplied through a LDO (low dropout) regulator with a 3.3×10^{0} V supply voltage which is available on pins 10 and 19 of the breakout board for the connection of external sensors, peripherals, etc.

On pin 18 of the brakout board the `AREF` pin of the ATmega32U4 is available for the possible connection of an external voltage reference or to be used as a voltage reference of an external circuit.

5.3 PORT B

Port B of the ATmega32U4 is an 8-bit GPIO port with individual control over all 8 pins `PB0` to `PB7`. Data direction is controlled via the `DDRB` register, output data is sent to the `PORTB` register and data from the outside is read via the `PINB` register.

Some pins of port B have additional functional layers:

- `PB0` to `PB3` are alternatively used by the SPI interface and for serial programming of the chip
- `PB4` to `PB6` can also be used as additional analog inputs for the internal ADC
- `PB5` to `PB7` can also carry PWM signals generated by timer 0, 1 and 4
- `PB7` is also used as control signal `RTS` in full UART operation

5.4 PORT C

Port C of the ATmega32U4 is internally an 8-bit GPIO port, but only pins `PC6` and `PC7` are available on the outside of the package. Data direction is controlled via the `DDRC` register, output data is sent to the `PORTC` register and data from the outside is read via the `PINC` register.

The pins of port C have additional functional layers:

- `PC6` and `PC7` can also carry PWM signals generated by timer 3 and 4
- `PC7` can be used an output pin for the internal CPU clock
- `PC7` can also be used as a capture signal input for timer 3

5.5 PORT D

Port D of the ATmega32U4 is an 8-bit GPIO port with individual control over all 8 pins `PD0` to `PD7`. Data direction is controlled via the `DDRD` register, output data is sent to the `PORTD` register and data from the outside is read via the `PIND` register.

Some pins of port B have additional functional layers:

- `PD0` and `PD1` are alternatively used by the I2C or TWI interface
- `PD2`, `PD3` and `PD5` are used by the UART interface
- `PD4`, `PD6` and `PD7` can also be used as additional analog inputs for the internal ADC

- PD4 can also be used as a capture signal input for timer 1
- PD6 can be used as an external clock source for timer 1
- PD7 can be used as an external clock source for timer 0

5.6 PORT E

Port E of the ATmega32U4 is an 8-bit GPIO port, but only pins PE2 and PE6 are available on the outside of the package. Data direction is controlled via the DDRE register, output data is sent to the PORTE register and data from the outside is read via the PINE register.

The breakout board does not give access to either of these pins.

- PE2 is tied to ground to enable the use of the internal bootloader
- PE6 is hardwired to the red status LED on the board

5.7 PORT F

Port F of the ATmega32U4 is internally an 8-bit GPIO port, but only pins PF0, PF1 and PF4 to PF7 are available on the outside of the package. Data direction is controlled via the DDRF register, output data is sent to the PORTF register and data from the outside is read via the PINF register.

The pins of port C have additional functional layers:

- all pins on port F can also be used as additional analog inputs for the internal ADC
- PF4 to PF7 are used for the JTAG interface – if enabled

6 Bootloader

The *Caterina* bootloader is activated by a double-click of the reset button on the breakout board. When in bootloader mode the PE6 LED signals a pumping flash and the breakout board can be recogninzed on an attached computer as a CDC-compatible serial port. The bootloader will timeout after about 8×10^0 s and normal program execution will restart if no connection from a host computer is made.

Uploaders like avrdude need to be provided with the address of this serial port (COMxx under Windows, /dev/ttyXX or similar under MacOS and Linux) in order to connect to the ATmega32U4 and be able to upload new program code into the flash memory.

The breakout boards are preliminary identifying themselves with a VID of 0x1B4F and a PID of 0x9205 as a *Spark Fun* 5V *Pro Micro* to which they are mostly compatible, apart from the lower clock frequency of 8×10^0 MHz – but this is not relevant for the functioning of the bootloader.

7 CDC serial port

The USB connection can also be used for a continuous serial data connection to a host computer during the normal program execution. For this a separate library, the *M2 USB communication subsystem* by J. Fiene & J. Romano is recommended and will be used during the course.

External links

- Schematic drawing of the *Pro Micro* from *Spark Fun*
  `https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Pro_Micro_v13b.pdf`
- ATmega32U4 datasheet
  `http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf`
- The original *Caterina* bootloader by Dean Camera on GitHub
  `https://github.com/adafruit/Caterina-Bootloader`
- *M2 USB communication subsystem* by J. Fiene & J. Romano
  `http://medesign.seas.upenn.edu/index.php/Guides/MaEvArM-usb`