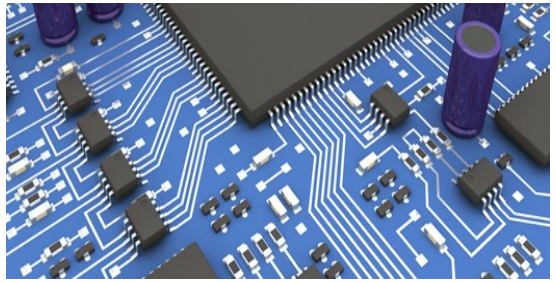
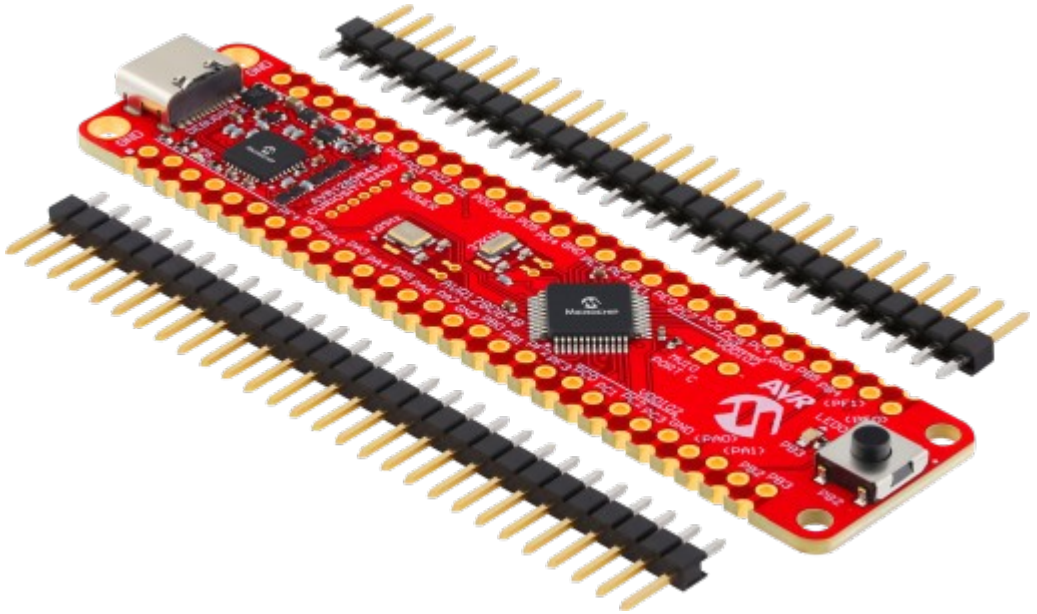
Aplikace Embedded systémů v Mechatronice





MPLAB® X IDE

Michal Bastl

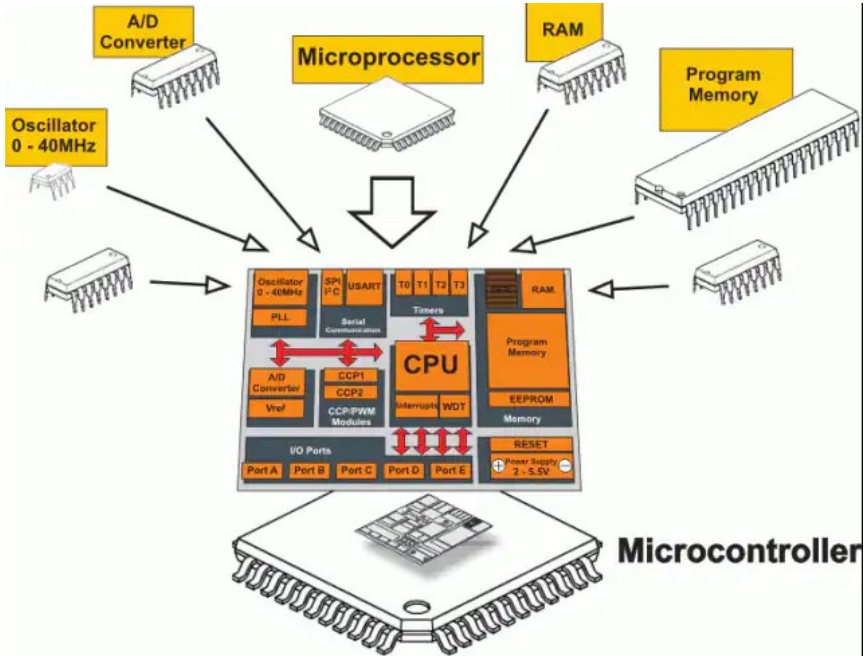
Vestavěný systém (embedded)

MCU v jednom pouzdře:

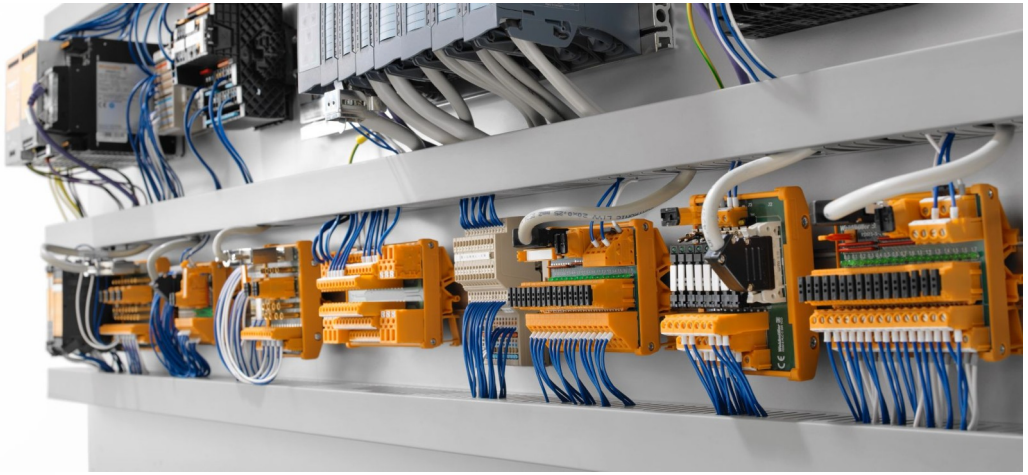
- Procesorové jádro (např. ARM Cortex-M, AVR, RISC-V, PIC)
- Paměť (RAM, Flash)
- Periferie (GPIO, UART, SPI, I2C, ADC, PWM)
- Taktovací obvody (interní/externí oscilátor)
- Napájení a řízení spotřeby
- Debuggovací jednotka

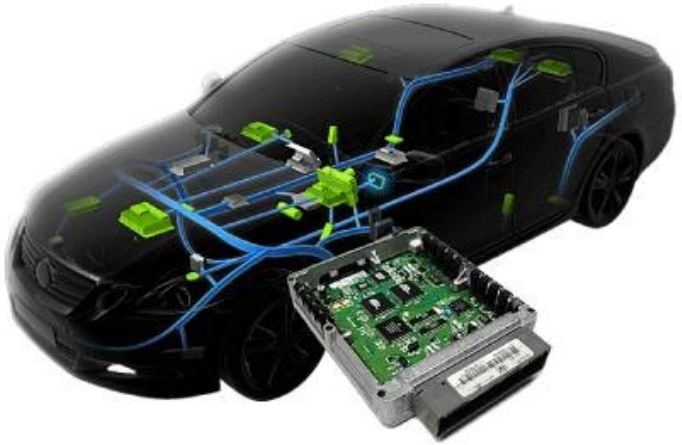
Přehled populárních MCU

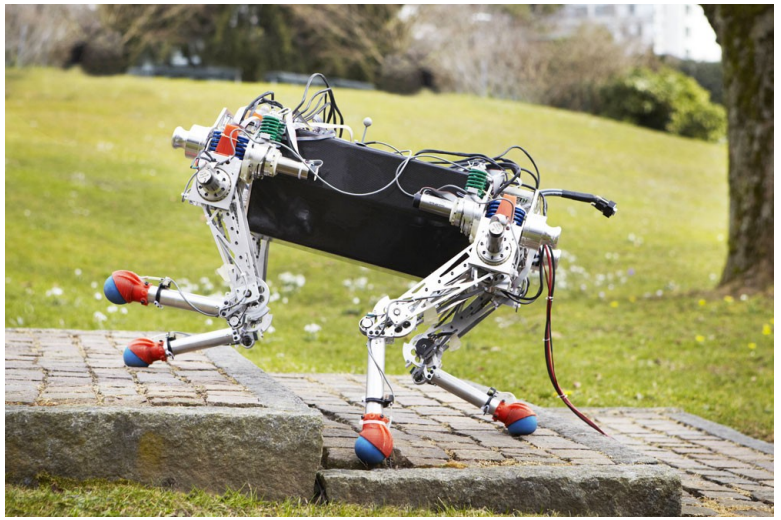
- ARM Cortex-M (STM32, NXP, TI, Nordic)
- AVR (Microchip ATmega, ATtiny)
- ESP32 (WiFi + Bluetooth)
- PIC (Microchip PIC16, PIC32)
- RISC-V (SiFive, GD32)



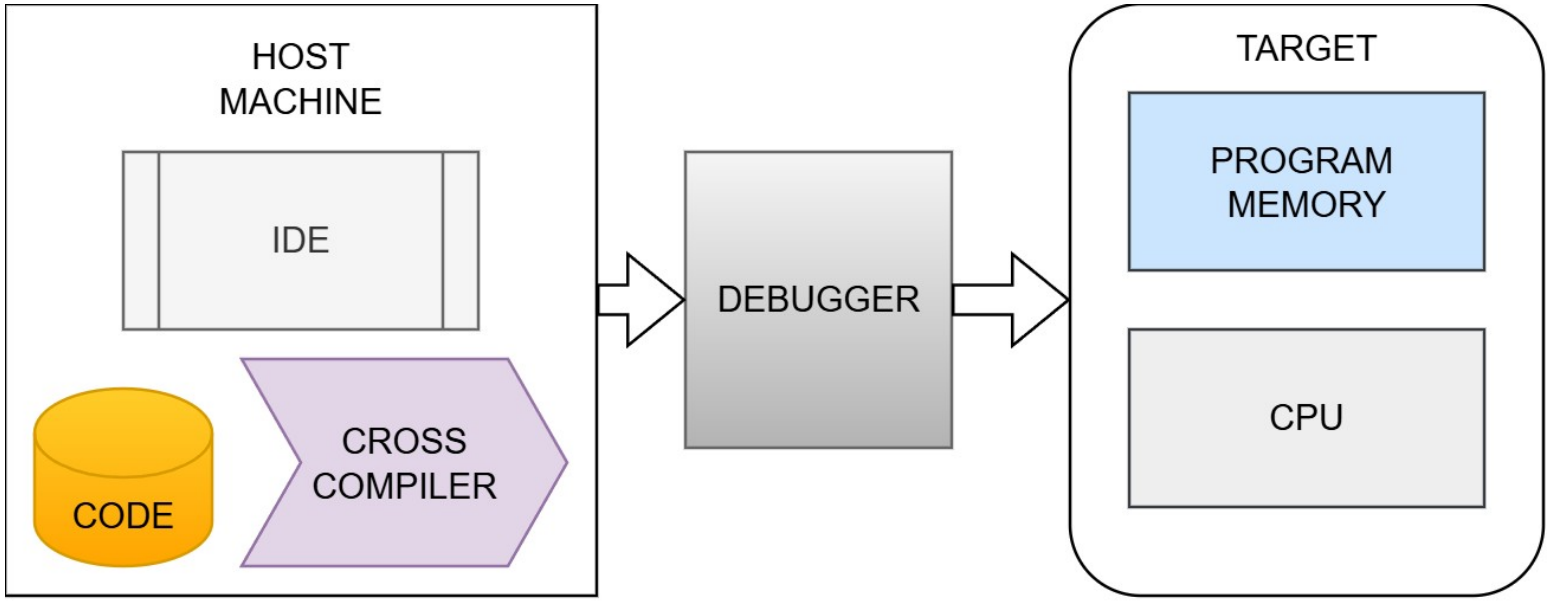
Použití MCU

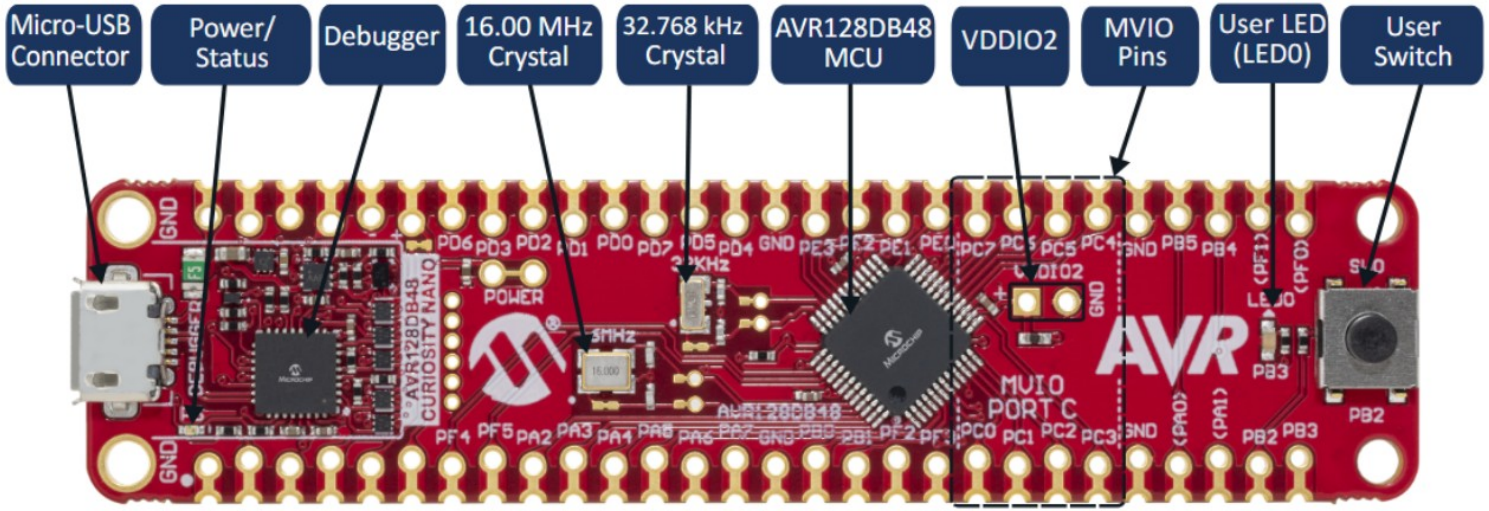






Vývoj a potřebné nástroje





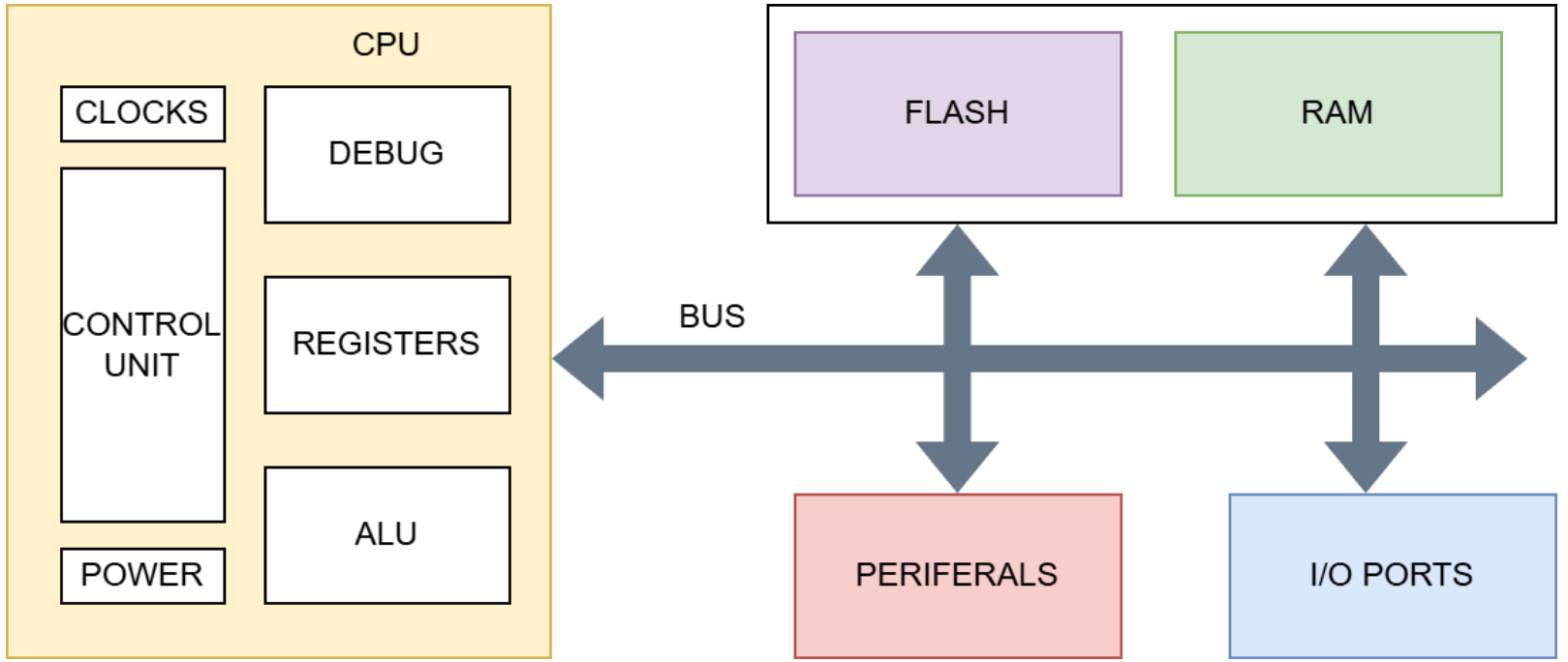
Programátorský model

LOAD/STORE architektura:

- Práce s pamětí jen přes speciální instrukce.
- CPU pak provádí operace na registrech-malých rychlých pamětech.

RISC (redukovaná instrukční sada) MCU

CISC (komplexní instrukční sada) PC



Instrukce, instrukční sada (pseudo kód)

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
    sum = sum + i;
}
```

```
ADRESA   | INSTRUKCE          | VÝZNAM
-------------------------------------------------------------
0x1000   | SET R1, 0          | sum = 0 (příprava v registru)
0x1004   | SET R2, 1          | i = 1 (příprava v registru)
         |                    |
0x1008   | CMP R2, 5          | Porovnej i se s číslem 5
0x100C   | JUMP_IF_GT 0x101C  | Pokud i > 5, skoč na KONEC
0x1010   | ADD R1, R2         | sum = sum + i
0x1014   | ADD R2, 1          | i = i + 1
0x1018   | JUMP 0x1008        | Skoč zpět na začátek porovnání
         |                    |
0x101C   | STORE [0x2000], R1 | Ulož konečný výsledek do RAM
0x1020   | HALT               | Zastav program
```

```
int data = 200;
int mask = 0x0F;
int key  = 0xAA;

data = data & mask;
data = data ^ key;
```

```
-------------------------------------------------------------
0x1000   | LOAD R1, [0x3000]  | Načti proměnnou 'data' z RAM do R1
0x1004   | LOAD R2, [0x3008]  | Načti 'mask' (0x0F) z RAM do R2
0x1008   | LOAD R3, [0x3010]  | Načti 'key' (0xAA) z RAM do R3
         |                    |
0x100C   | AND R1, R2         | Bitový součin: R1 = R1 & R2
0x1010   | XOR R1, R3         | Bitový XOR: R1 = R1 ^ R3
         |                    |
0x1014   | STORE [0x3000], R1 | Zapiš upravený výsledek zpět do RAM
0x1018   | HALT               | Konec programu
```

AVR128DB48

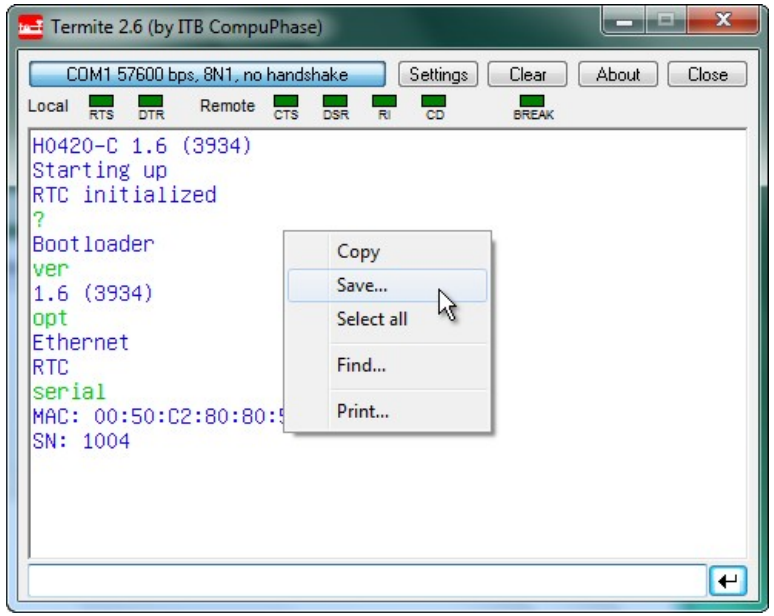
MCU - mikrokontrolér

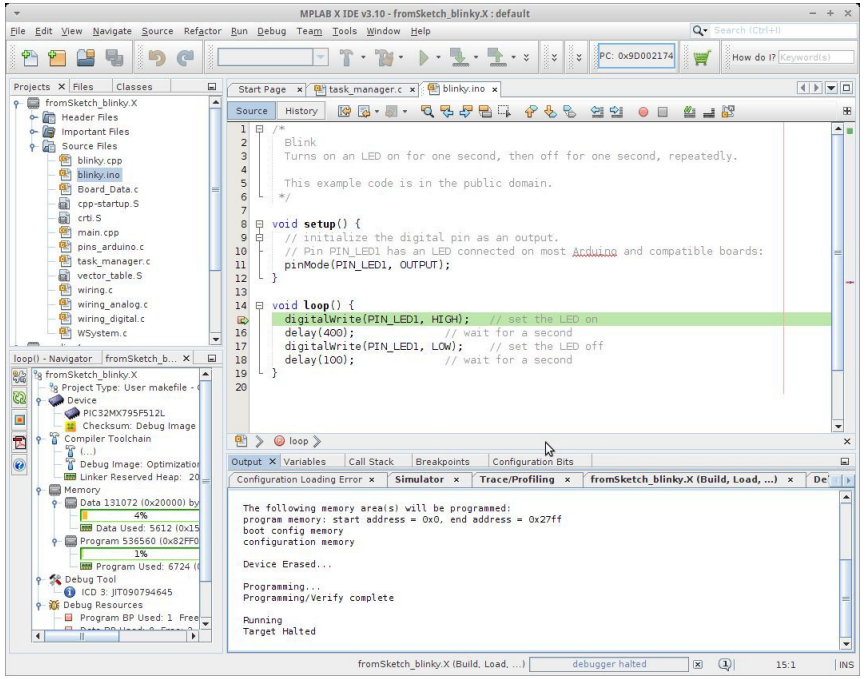
Program Memory Type	Flash
Program Memory Size (KB)	128kB
CPU Speed (MIPS/DMIPS)	24
SRAM (B)	16384B
Data EEPROM/HEF (bytes)	512
Digital Communication Peripherals:	2-UART, 2-SPI, 2-I2C2
Capture/Compare/PWM Peripherals:	2 CCP, 3 ECCP (PWM)
Timers	TCA, TCB, RTC 16bit
ADC Input	12bit
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5

Ukázka potřebných nástrojů

Software:

- Microchip MPLAB — vývojové prostředí + XC8
- Termit — sériový terminál

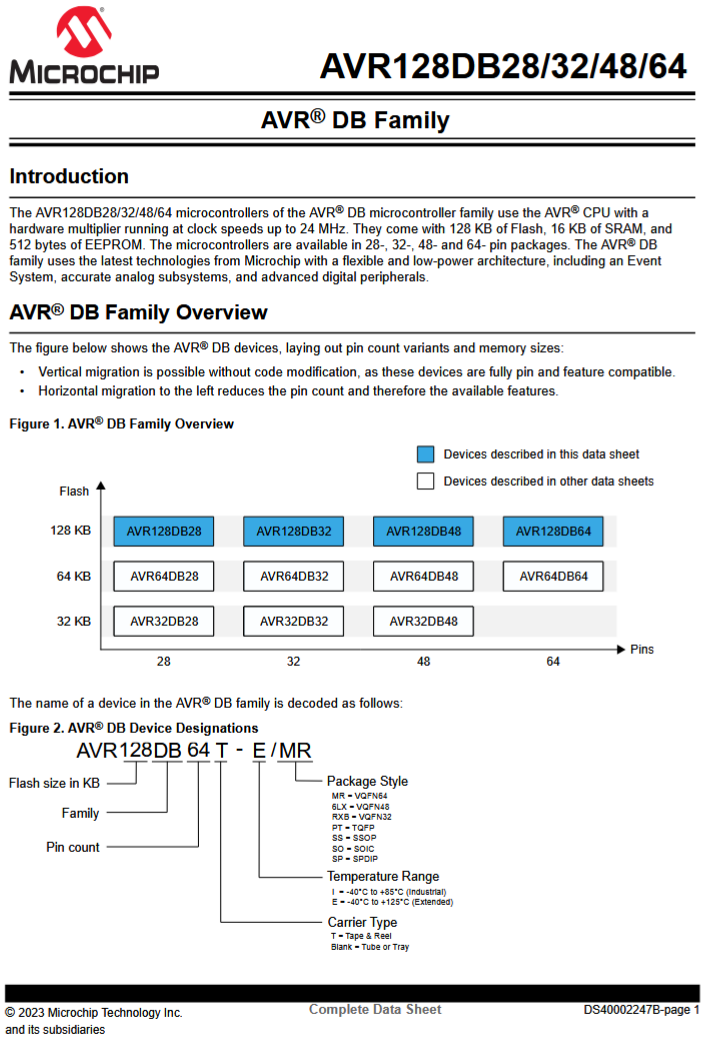




Práce s dokumentaci

- Datasheet je strukturovaný a najdeme zde kapitoly podle jednotlivých periferii. GPIO, timer, ADC apod.

- Není nutné znát přesně nastavení z hlavy. K tomu právě slouží datasheet.



MICROCHIP

AVR128DB28/32/48/64

AVR® DB Family

Introduction

The AVR128DB28/32/48/64 microcontrollers of the AVR® DB microcontroller family use the AVR® CPU with a hardware multiplier running at clock speeds up to 24 MHz. They come with 128 KB of Flash, 16 KB of SRAM, and 512 bytes of EEPROM. The microcontrollers are available in 28-, 32-, 48- and 64- pin packages. The AVR® DB family uses the latest technologies from Microchip with a flexible and low-power architecture, including an Event System, accurate analog subsystems, and advanced digital peripherals.

AVR® DB Family Overview

The figure below shows the AVR® DB devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore the available features.

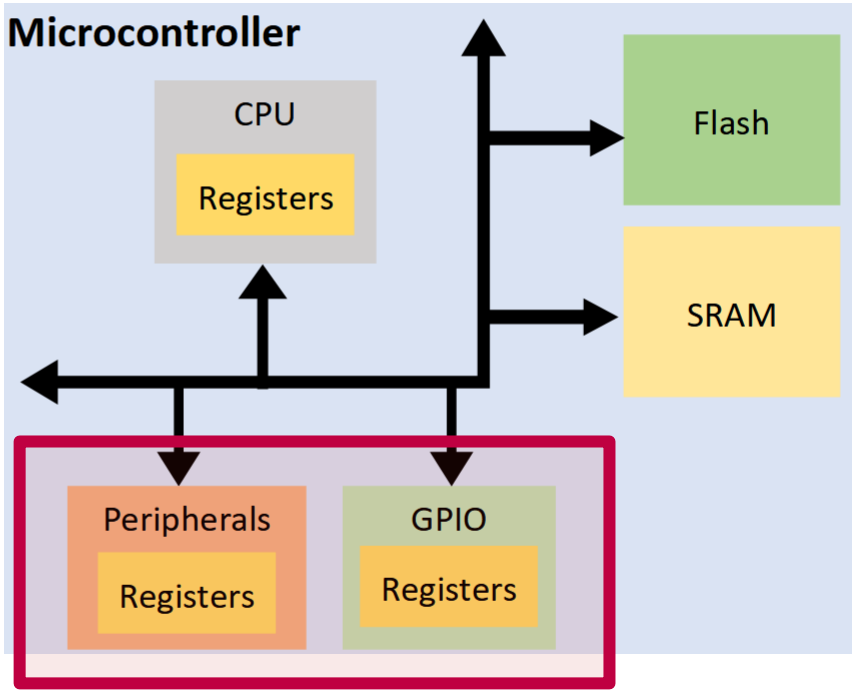
Figure 1. AVR® DB Family Overview

The name of a device in the AVR® DB family is decoded as follows:

Figure 2. AVR® DB Device Designations

Registry periferii

- Jsou speciální registry, které slouží pro práci s periferiemi MCU
- Každá periferie má své registry, ty mají svoji adresu v paměti
- Mají většinou intuitivní název PORT, ADC podle názvu v datasheetu
- V hlavičkovém souboru avr/io.h jsou zavedeny makra a struktury, které může programátor používat pro práci s periferiemi
- Je však možné pracovat přímo s adresou v dokumentaci na str.68 je Peripheral Address Map
- Například registry pro PORTB začínají na adrese 0x0420



Práce s registry

Práce s periferiemi vyžaduje manipulaci s registry.

V Datasheetu MCU nalezneme význam a popis nastavení.

Je dobré připomenout koncept bitových operátorů a bitových masek.

Operator	Function
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR (Exclusive OR)
<<	LEFT SHIFT
>>	RIGHT SHIFT

```
bits &= ~(1 << 7) ; /* clears bit 7 */
```

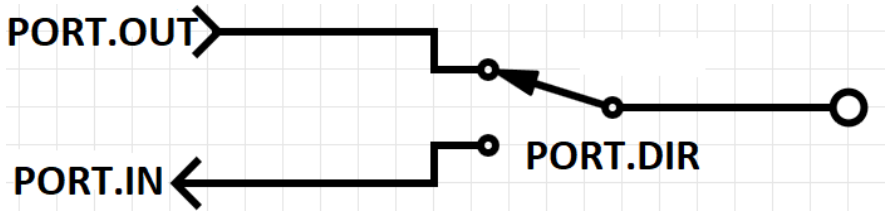
```
(1 << 7)    ➡  10000000
~(1 << 7)   ➡  01111111
```

Masky:

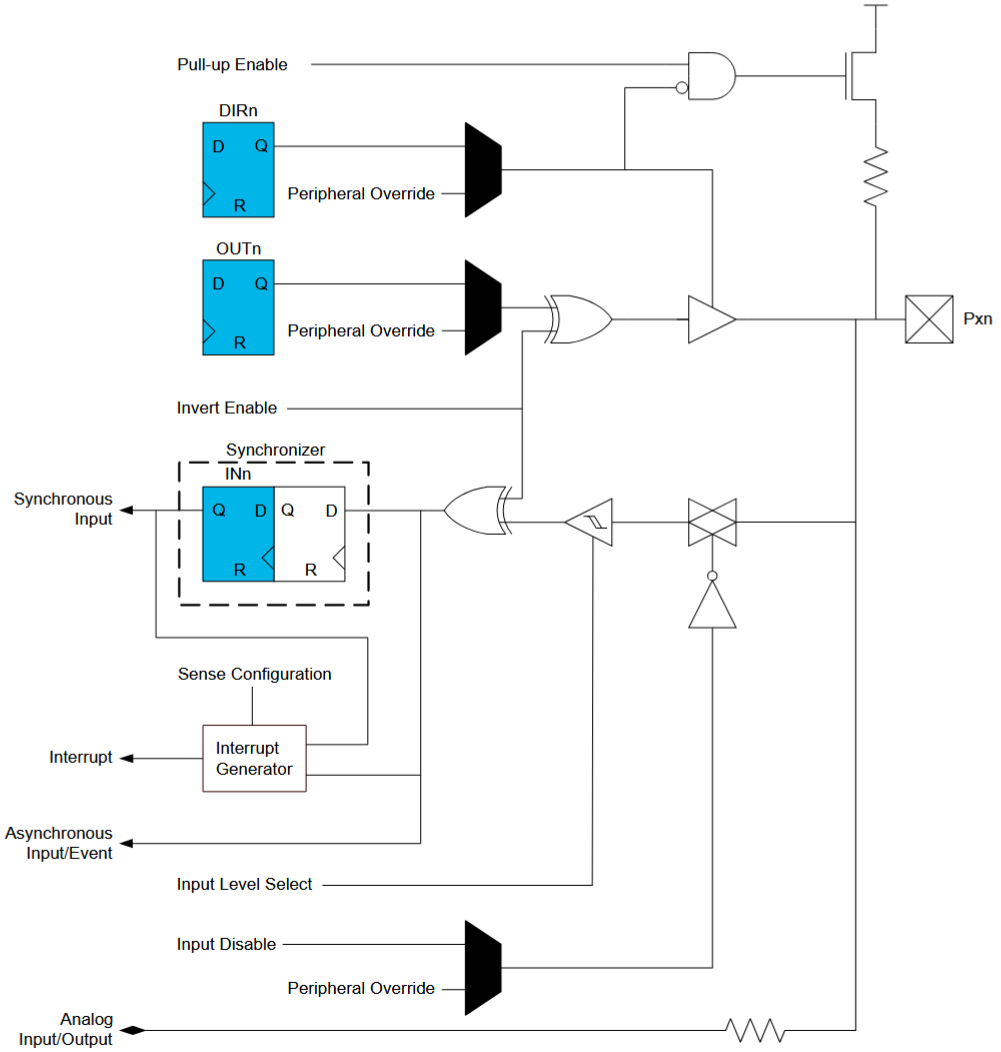
&	|	^
10011110	10011110	10011110
00001111	00000001	00001111
00001110	10011111	10010001

GPIO pin

- General purpose input/output, tedy obecný vstupně/výstupní pin.
- Slouží k základní interakci MCU s okolním světem.
- Na GPIO pin lze zapisovat 1, tedy napětí blízké napájecímu 3.3V, nebo 0 napětí blízké 0V.
- V input režimu lze pinem číst stav napětí. Pokud je blízké 0V čte se jako 0, nebo blízké 3,3V jako 1.



Zjednodušení



PORTn.DIR

Registr DIR (Data Direction):

str. 180

Určuje, zda pin funguje jako vstup nebo výstup.

- Logická 0: Pin je nastaven jako vstup (input).
- Logická 1: Pin je nastaven jako výstup (output).

Bit	7	6	5	4	3	2	1	0
	DIR[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

Bits 7:0 – DIR[7:0] Data Direction

This bit field controls the output driver for each PORTx pin.

This bit field does not control the digital input buffer. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.

The table below shows the available configuration for each bit n in this bit field.

Value	Description
0	Pxn is configured as an input-only pin, and the output driver is disabled
1	Pxn is configured as an output pin, and the output driver is enabled

```
PORTB.DIR |= (1 << 3);  // Nastaví PIN3 na 1 (out)
PORTB.DIR &= ~(1 << 2); // Nastaví PIN2 na 0 (in)
```

Využití registrů SET a CLR:

```
PORTB.DIRSET = (1 << 3); // Nastaví PIN3 na 1 out
PORTB.DIRCLR = (1 << 2); // Nastaví PIN2 na 0 in
```

PORTn.OUT

Registr OUT (Output Value)

Určuje logickou úroveň na pinech, které jsou v registru DIR nastaveny jako výstupy.

- Logická 0: Na výstupu je nízká úroveň (GND).
- Logická 1: Na výstupu je vysoká úroveň (VDD).

Name: OUT
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUT[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

Bits 7:0 – OUT[7:0] Output Value
This bit field controls the output driver level for each PORTx pin.
This configuration only affects when the output driver (PORTx.DIR) is enabled for the corresponding pin.
The table below shows the available configuration for each bit n in this bit field.

Value	Description
0	The pin n (Pxn) output is driven low
1	The Pxn output is driven high

```
PORTB.OUTSET = (1 << 3); // Nastaví PIN3 na 1
PORTB.OUTCLR = (1 << 2); // Nastaví PIN2 na 0
```

Využití maker z avr/io.h zlepšuje čitelnost:

```
PORTB.OUTSET = PIN3_bm; // Nastaví PIN3 na 1 (High)
PORTB.OUTCLR = PIN2_bm; // Nastaví PIN2 na 0 (Low)
```

PORTn.IN

Registr IN (Input Value)

Slouží k načítání aktuálního stavu napětí na pinech.

Čtení: Vrací logickou hodnotu přítomnou na fyzickém pinu (0 nebo 1).

Kontrola stavu pinu:

if(!(PORTB.IN & PIN2_bm))

Bit	7	6	5	4	3	2	1	0
	IN[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

Bits 7:0 – IN[7:0] Input Value

This bit field shows the state of the PORTx pins when the digital input buffer is enabled.
Writing a '0' to bit n in this bit field has no effect.
Writing a '1' to bit n in this bit field will toggle the corresponding bit in PORTx.OUT.
If the digital input buffer is disabled, the input is not sampled, and the bit value will not change. The digital input buffer for pin n (Pxn) can be configured in the Input/Sense Configuration (ISC) bit field in the Pin n Control (PORTx.PINnCTRL) register.
The table below shows the available states of each bit n in this bit field.

Value	Description
0	The voltage level on Pxn is low
1	The voltage level on Pxn is high

PORTn.PINnCTRL

Hlavní funkce PINnCTRL

- **PULLUPEN** (Pull-up Enable): Aktivuje interní pull-up rezistor (cca 35 kΩ). Připojuje pin k VDD, což udržuje na vstupu logickou 1, pokud není pin externě stažen k zemi.
- **INVEN** (Input Invert): Invertuje logickou hodnotu pinu. Pokud je aktivováno, fyzická log. 0 na pinu se v registru IN jeví jako log. 1.
- **ISC** (Input/Sense Configuration): Definuje chování digitálního vstupu a spouštění přerušení (Interrupt). Možnosti nastavení zahrnují:

Name: PINnCTRL
Offset: 0x10 + n*0x01 [n=0..7]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	INVEN	INLVL			PULLUPEN	ISC[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

ISC[2:0] Input/Sense Configuration
This bit field controls the input and sense configuration of pin n. The sense configuration determines how to trigger a port interrupt.

Value	Name	Description
0x0	INTDISABLE	Interrupt disabled but digital input buffer enabled
0x1	BOTHEDGES	Interrupt enabled with sense on both edges
0x2	RISING	Interrupt enabled with sense on rising edge
0x3	FALLING	Interrupt enabled with sense on falling edge
0x4	INPUT_DISABLE	Interrupt and digital input buffer disabled(1)
0x5	LEVEL	Interrupt enabled with sense on low level(2)
other	—	Reserved

Zápis do registrů v C

- Klíčové slovo volatile ještě uvidíme
- Použití tohoto slova v deklaraci proměnné znamená, že zakazujeme optimalizace této proměnné

```
int main(void) {
    // Umístíme pointer na adresu registru periferie
    char *DIRSETB   = (char*)(0x0420 + 0x01);
    char *OUTTGLB   = (char*)(0x0420 + 0x07);

    // nastavení outputu
    *DIRSETB  |= (1 << 3);

    while(1){

        // Smyčka čekání pomocí for (volatile zakáže compilatoru optimalizovat/vynechat prázdný for)
        for(volatile long i = 0; i < 100000; i++){

        }
        // zápisem 1 do OUTTGLB se stav překlopí automaticky
        *OUTTGLB |= (1 << 3);

    }
}
```

Použití definition file od výrobce

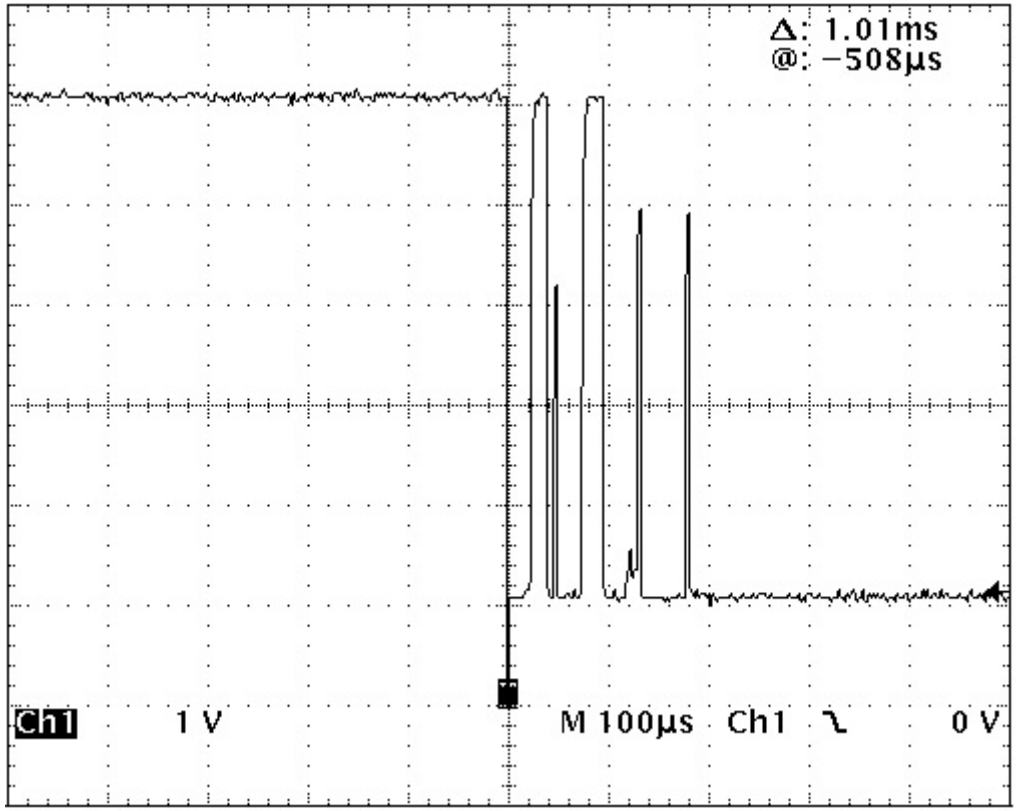
- avr/io.h
- Obsahuje makra, struktury pro práci s periferiemi

```
#define F_CPU 4000000UL    // Definice frekvence (výchozi je 4 MHz)
#include <avr/io.h>
#include <util/delay.h>


int main(void) {
    // Nastavení pinu jako VÝSTUP (Output)
    // Používáme registr DIRSET
    PORTB.DIRSET = PIN3_bm;

    while (1) {
        // Překlopení (Toggle) stavu pinu
        // Každým zápisem 1 do OUTTGL se stav LED změní (z 0 na 1 a naopak)
        PORTB.OUTTGL = PIN3_bm;

        // Čekání
        _delay_ms(500);
    }
}
```

Debouncing

- V praxi se může vyskytnout problém při stlačení tlačítka.
- Ten se projevuje tak, že se tlačítko vyhodnotí jako stisknuté vícekrát.
- Problém je třeba řešit jak vhodným HW a nejlépe i v SW

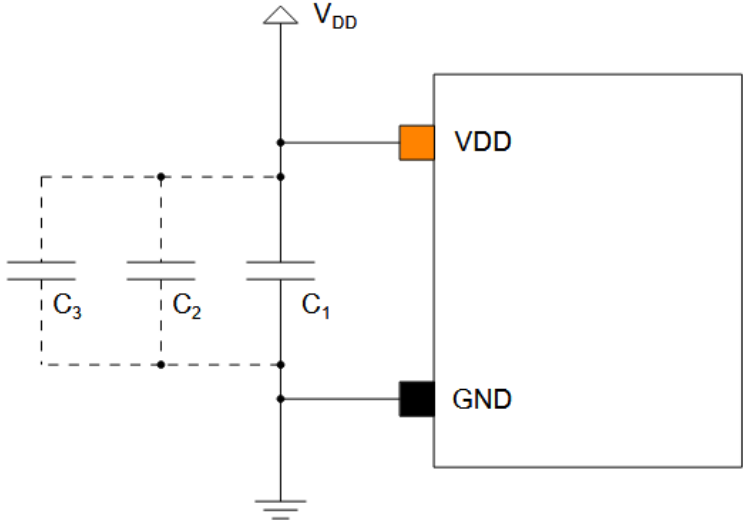
- Tento jev trvá cca 5-10ms
- Nejednoduší (ne nejlepší) řešení je přečíst tlačítko, počkat a přečíst znovu.



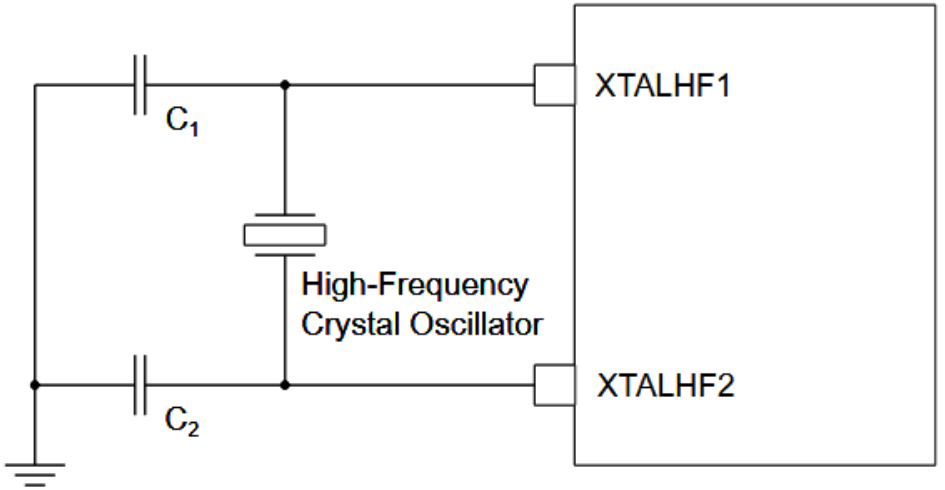
Debouncing

```
bool get_button_edge(bool raw_sample) {
    static int counter = 0;      // pocitadlo
    static bool stable_state = 0; // stabilni stav
    const int threshold = 100;     // slouzi jako filtr

    bool edge_detected = false;

    if (raw_sample != stable_state) {
        counter++;
        if (counter >= threshold) {
            // Stabilni stav po dobu threshold
            if (stable_state == 0 && raw_sample == 1) {
                edge_detected = true; // Detekce hrany
            }
            stable_state = raw_sample;
            counter = 0;
        }
    } else {
        counter = 0; // Reset pocitadla pri nesouhlasu
    }

    return edge_detected;
}
```

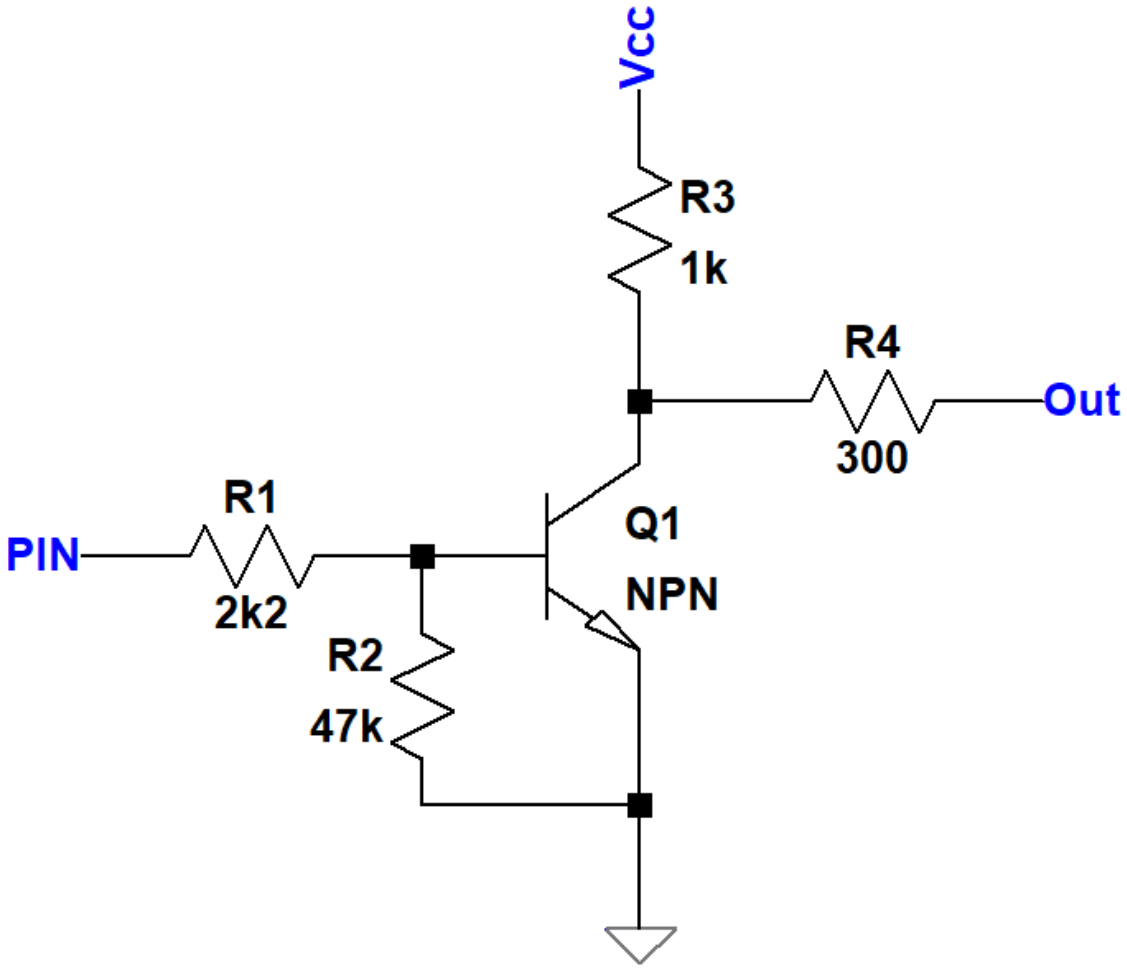
Hardware

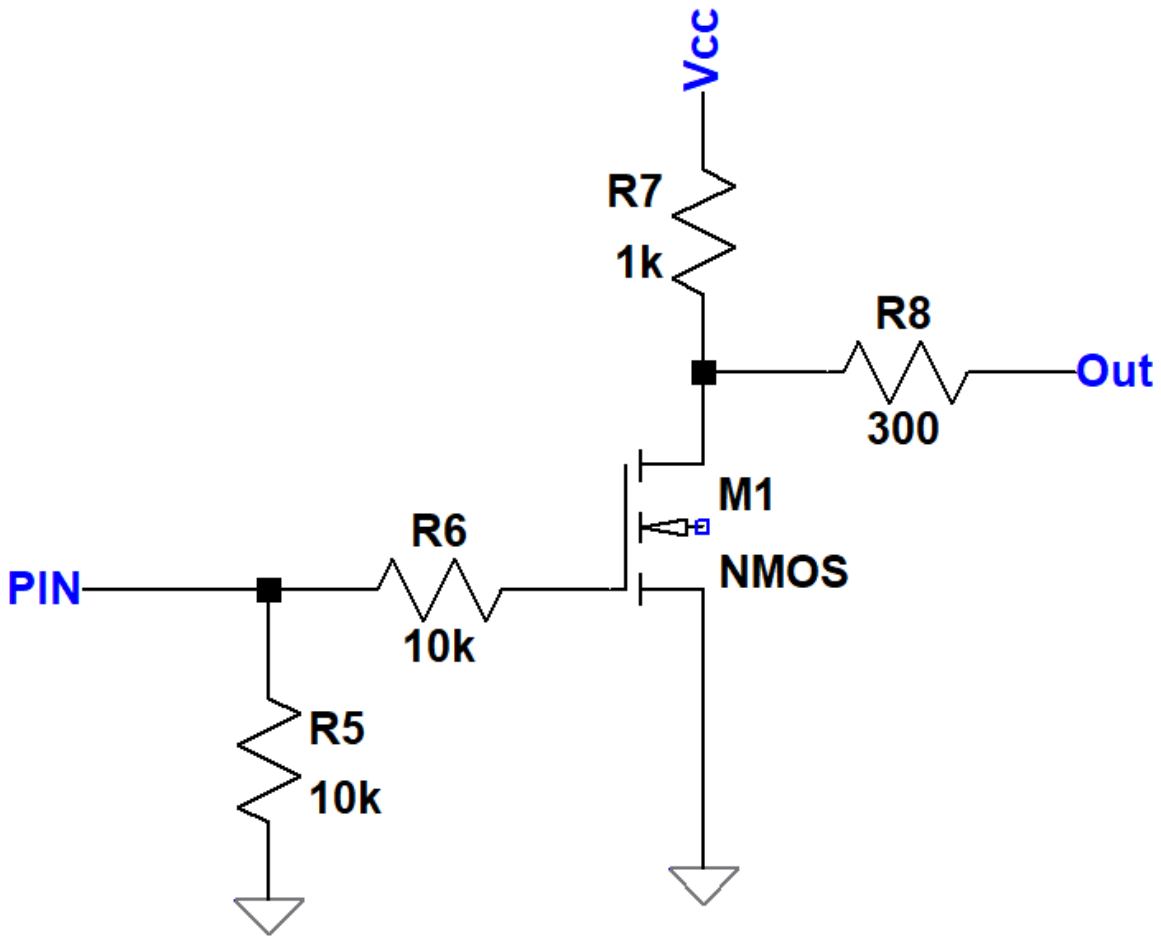


Typical values (recommended):
C_1: 100 nF (primary decoupling capacitor)
C_2: 1-10 nF (HF decoupling capacitor)
C_3(*): 1 µF (decoupling capacitor - optional)



GPIO posílení tranzistorem





GPIO zapojení