

# Obecný (číslicový) vstup a výstup, „porty“ mikrokontrolérů

(General Purpose Input/Output = GPIO)

**Mikroprocesorové a vestavěné systémy (IMP)**

Přednáší: doc. Ing. Richard Růžička, Ph. D., MBA

Fakulta informačních technologií VUT v Brně

# Vstupy a výstupy

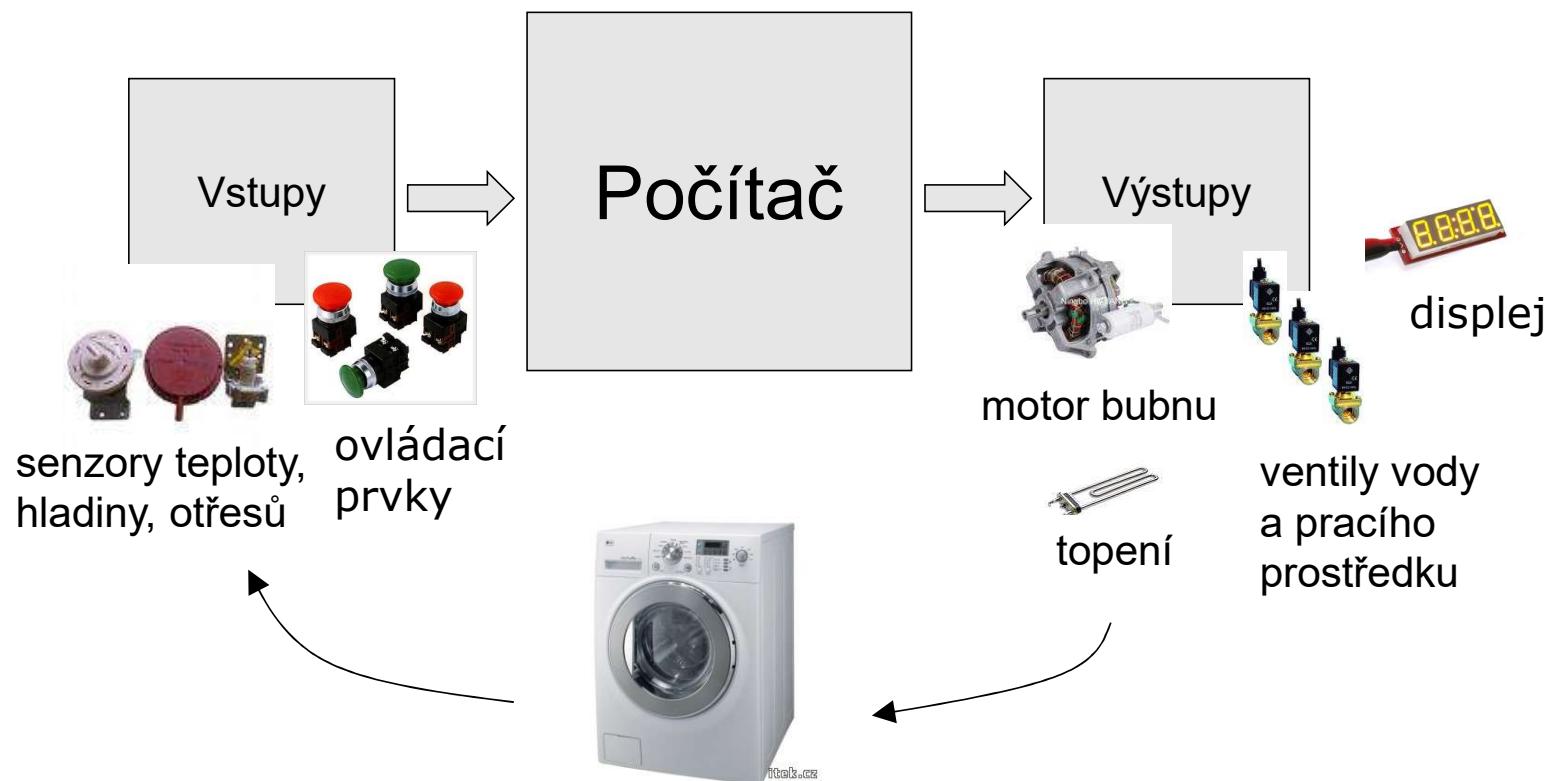
- Mikrokontrolér je počítač, určený k tomu, aby vnímal své okolí a na základě programu je ovlivňoval.
- Je zpravidla součástí nějakého systému, který má primárně jiné určení, než být „jen“ počítačem.

➡ Je to počítačový systém *vestavěný* do nějaké aplikace.

Má formu integrovaného elektronického obvodu, připojeného do aplikace prostřednictvím vývodů pouzdra.



# Vstupy a výstupy



# Vstupy a výstupy

Vstupy a výstupy u všech integrovaných obvodů bývají **buď číslicové (0/1) nebo analogové (zpravidla v rozsahu  $-V_{cc} \dots GND \dots +V_{cc}$ ).**

U mikrokontrolérů jsou **pochopitelně ovládané (výstupy) a snímané (vstupy) softwarově**, a to buď

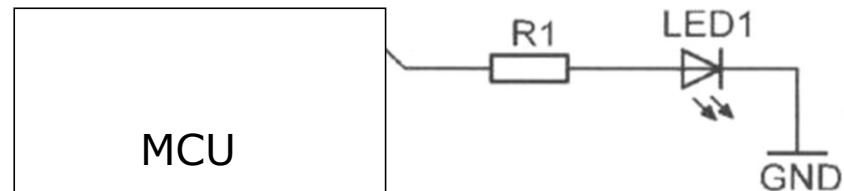
- přímo (instrukcí pro nastavení výstupu / čtení vstupu)
- nepřímo – jde o vstupy/výstupy některého modulu na čipu, který je ovládán softwarově (jeho chování je nastaveno programem).

Dnešní přednáška bude zejména o té první možnosti.

# Co se dá dokázat s číslicovým výstupem?

- Jednoduché, ale v podstatě velmi mocné věci:
  - rozsvítit/zhasnout kontrolku,
  - roztočit/zastavit motor,
  - otevřít/zavřít ventil,
  - sepnout/rozepnout relé,
  - zapnout/vypnout topnou spirálu, atd.

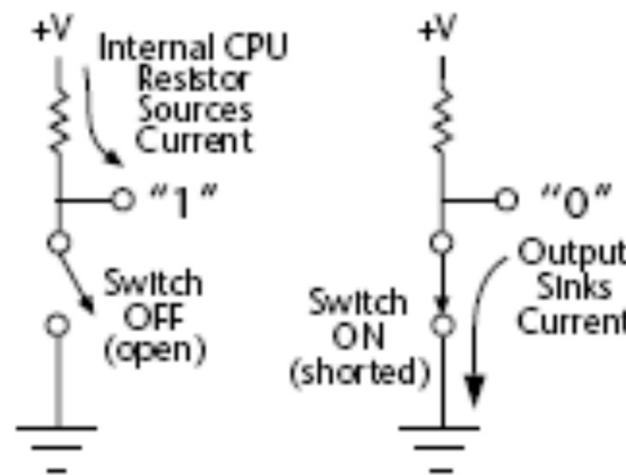
Příklad:  
ovládání LED



- Ale lze také využít celý port (nebo část) jako paralelní datové rozhraní (vyslat n-bitové slovo).

# Co se dá dokázat s číslicovým vstupem

- Opět velmi jednoduché, ale docela mocné věci:
  - snímat stav tlačítka,
  - snímat stav jakéhokoli dvoustavového snímače,
  - snímat polohu inkrementálně nebo absolutně,

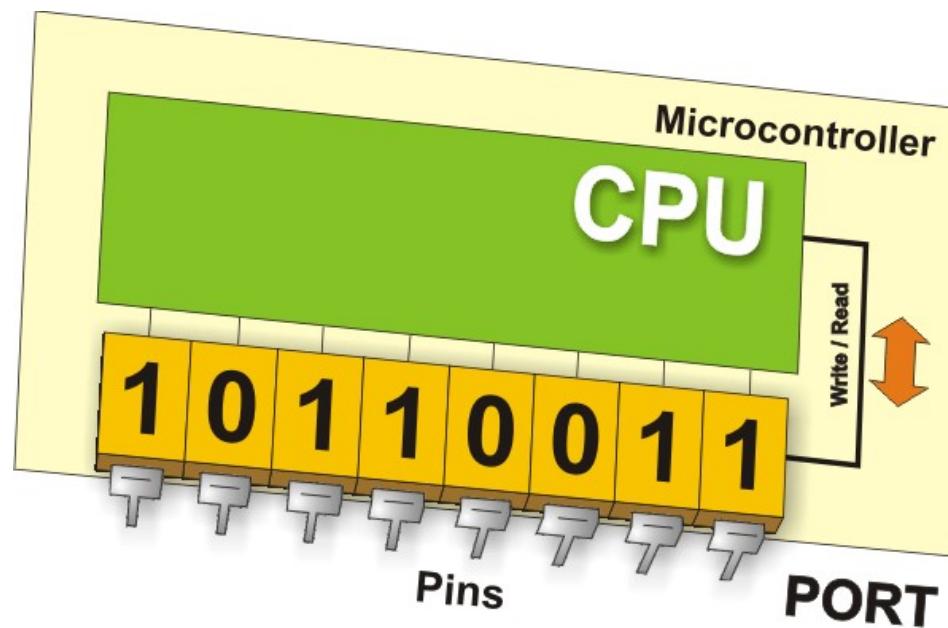


# Softwarová stránka číslicového výstupu

... jak z programu ovládnout fyzický svět.

# Přímé ovládání pinů

Přímo nastavit či zjistit stav nějakého pinu softwarově (z programu) lze pomocí datových registrů portů.



# Přímé ovládání pinů softwarem

Pro každý 32 bitový port přes sadu registrů (zde konkrétně NXP Kinetis, ale podobné všude):

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
400F_F000	Port Data Output Register (GPIOA_PDOR)	32	R/W	0000_0000h
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32	W (always reads 0)	0000_0000h
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h

Zde registry portu A, pro port B se registry jmenují GPIOB atd.

# Co dělají registry portu?

**GPIOx\_PDOR field descriptions**

Field	Description
31–0 PDO	<p>Port Data Output</p> <p>Register bits for un-bonded pins return a undefined value when read.</p> <p>0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output.</p> <p>1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.</p>

**GPIOx\_PSOR field descriptions**

Field	Description
31–0 PTSO	<p>Port Set Output</p> <p>Writing to this register will update the contents of the corresponding bit in the PDOR as follows:</p> <p>0 Corresponding bit in PDORn does not change.</p> <p>1 Corresponding bit in PDORn is set to logic 1.</p>

**GPIOx\_PCOR field descriptions**

Field	Description
31–0 PTCO	<p>Port Clear Output</p> <p>Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:</p> <p>0 Corresponding bit in PDORn does not change.</p> <p>1 Corresponding bit in PDORn is cleared to logic 0.</p>

# Příklad: dvě LED

Mějme dvě LED připojené anodami na PTB0 a PTB1, katody jsou připojeny přes rezistor na GND.

... pro rozsvícení je třeba nastavit piny PTB0 a PTB1 do log. 1:

PTB->PDOR = 0x0003;

Je-li třeba nyní jednu zhasnout, lze postupovat několika způsoby:

PTB->PDOR = 0x0001; vynuluje všechny bity, nechá jen PTB0

PTB->PDOR &= 0xFFFF; vynuluje jen PTB1, ale v několika krocích

PTB->PCOR = 0x0002; vynuluje PTB1 jedním zápisem

PTB->PTOR = 0x0002; invertuje PTB1 jedním zápisem

# Vstup nebo výstup? Nastavení směru pinu.

## 38.2.6 Port Data Direction Register (GPIOx\_PDDR)

The PDDR configures the individual port pins for input or output.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	PDD															
W																																

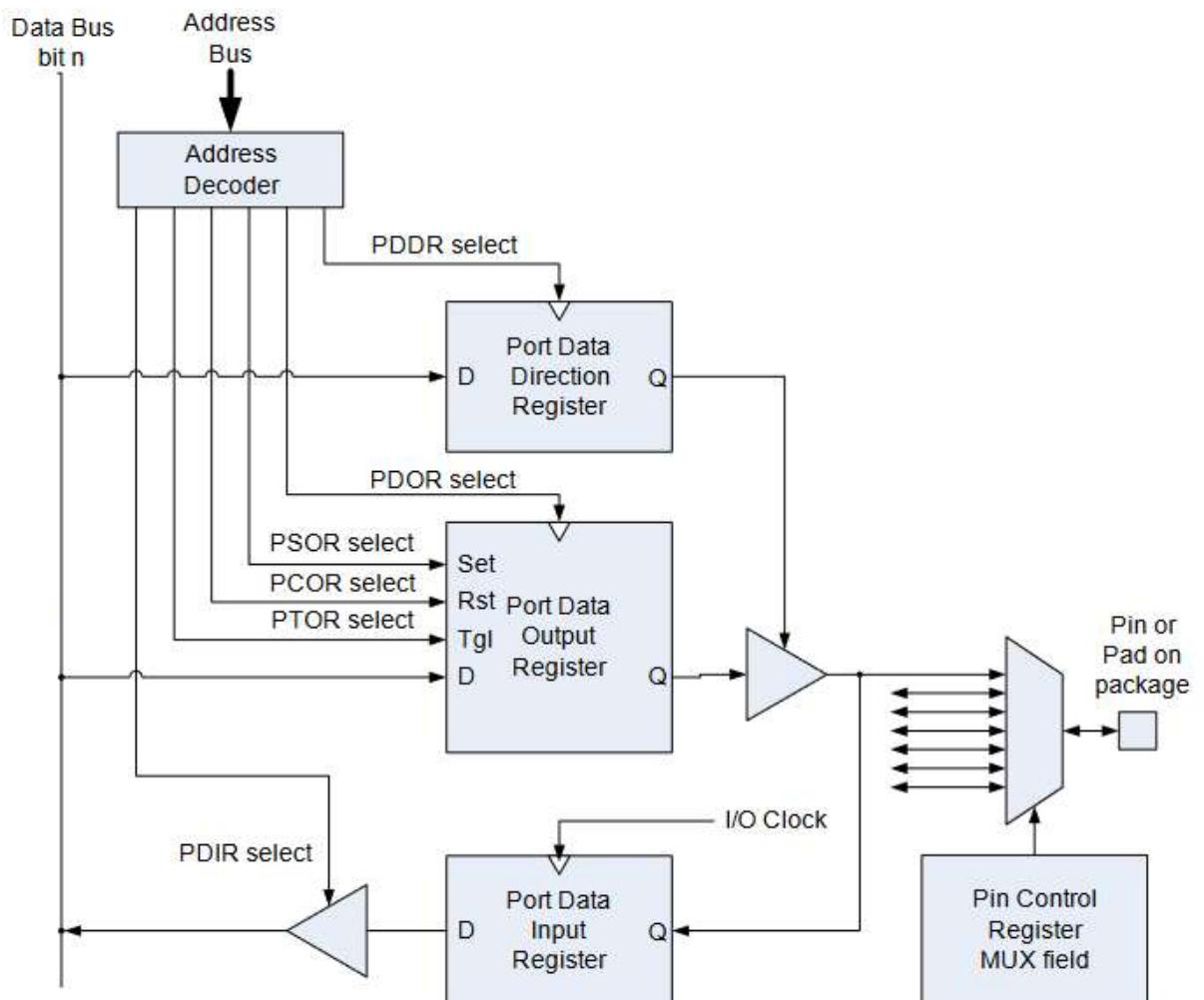
**GPIOx\_PDDR field descriptions**

Field	Description
31–0 PDD	Port Data Direction  Configures individual port pins for input or output.  0 Pin is configured as general-purpose input, for the GPIO function. 1 Pin is configured as general-purpose output, for the GPIO function.

... po RESETu jsou všechny piny portu nastaveny jako vstupy!  
Proč asi?

# Vstupní/výstupní pin MCU (GPIO)

Jak je to zapojeno uvnitř:

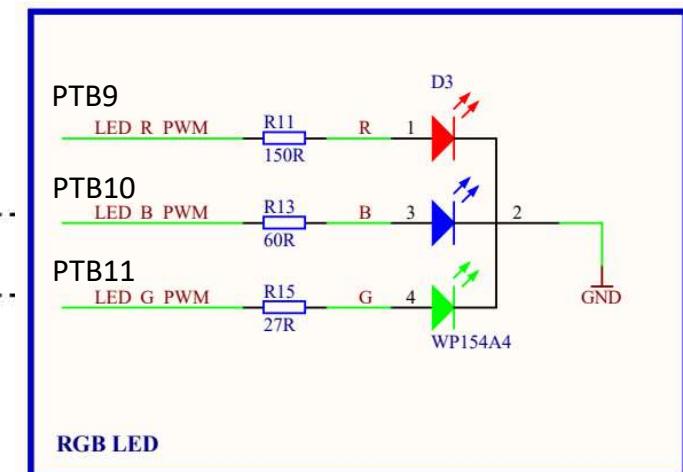
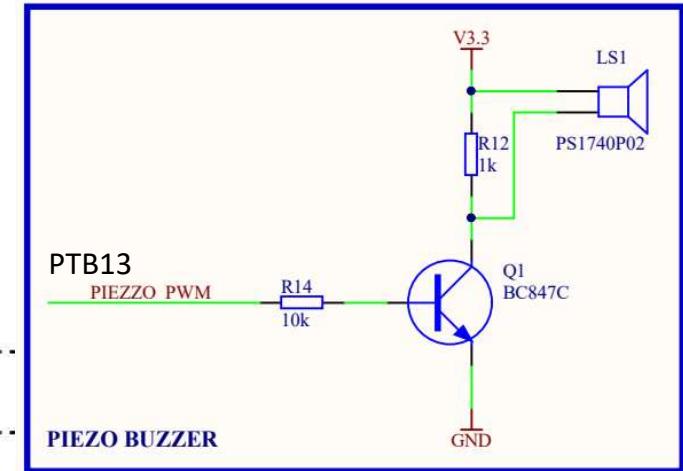


# Příklad

```
#define PIEZZO_MASK 0x2000
#define LED_R_MASK 0x200

/* -----
 * Funkce realizujici kratke pipnuti bzucaku
 * -----
void beep(void) {
    for (uint32_t q=0; q<200; q++) {
        GPIOB_PDOR |= PIEZZO_MASK; delay(200);
        GPIOB_PDOR &= ~PIEZZO_MASK; delay(200);
    }
}

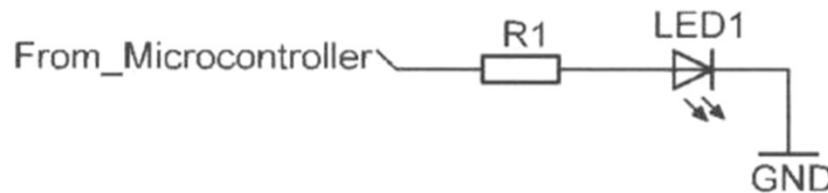
/* -----
 * Funkce realizujici kratke bliknuti LED_R
 * -----
void flash() {
    GPIOB_PDOR |= LED_R_MASK;
    delay(60000);
    GPIOB_PDOR &= ~LED_R_MASK;
}
```



# Hardwareová stránka číslicového výstupu

... co potřebujeme vědět, když chceme k číslicovému výstupu mikrokontroléra něco připojit.

# Jednabitový výstup – jeden pin

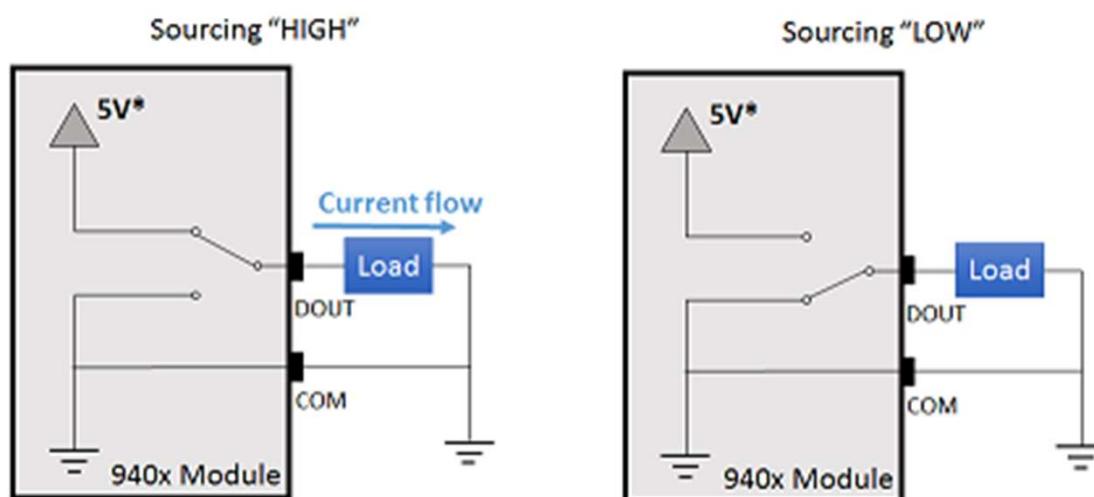


LED1 svítí pro log. 1



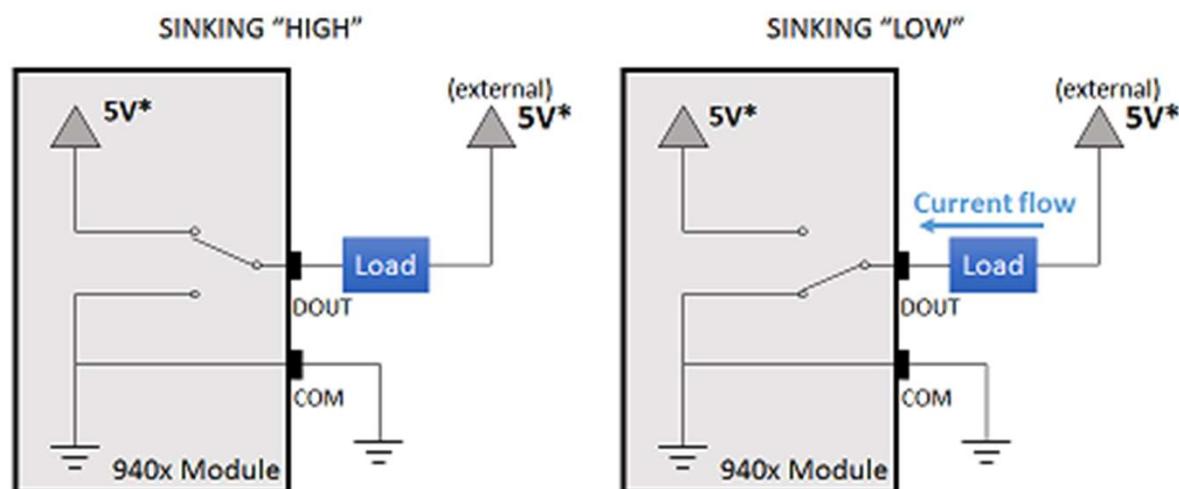
LED1 svítí pro log. 0

Připomenutí – proč pro 1 svítí a pro 0 nesvítí.



\* this is 3.3V for the 9402

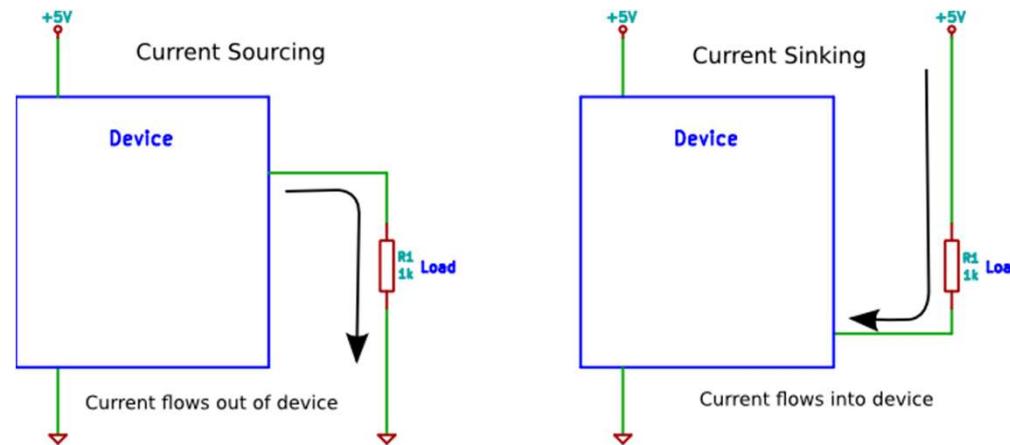
... a ten opačný případ



\* this is 3.3V for the 9402

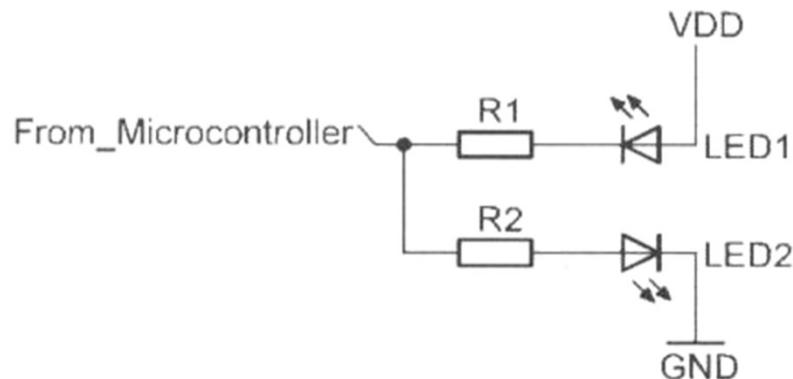
# „Sourcing“ and „Sinking“ pin

... i když se bavíme o číslicovém výstupu, proud někdy teče dovnitř a někdy ven.



Výstup to tedy není podle směru toku proudu, ale podle toho, že je to mikrokontrolér, kdo diktuje logickou hodnotu na pinu.

## Stavy pinu v „číslicovém“ režimu



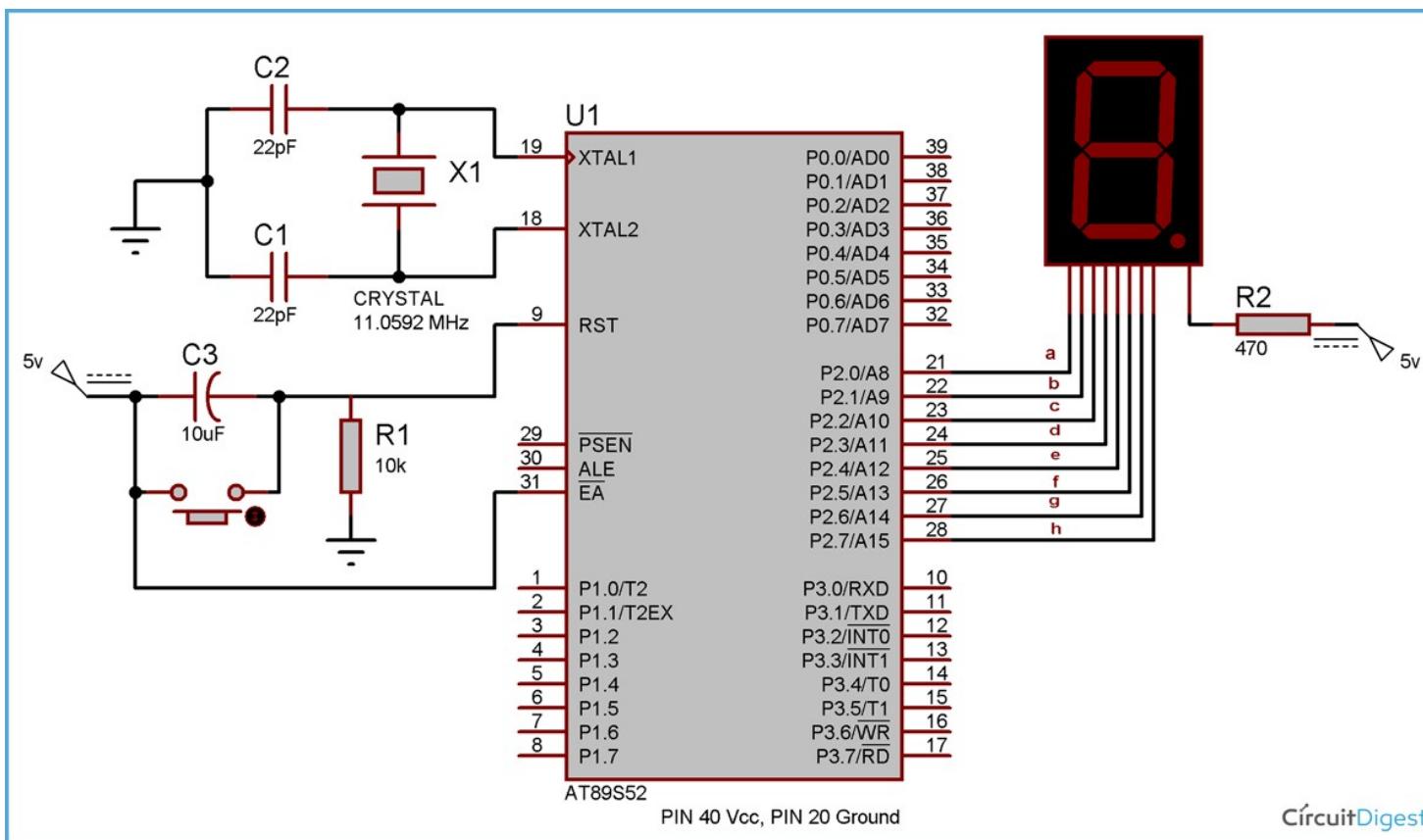
LED1 svítí pro log. 0  
LED2 svítí pro log. 1

aby nesvítila ani jedna, nastaví se port jako vstupní ;-)

Pin, pokud je konfigurován jako číslicový, může být v podstatě  
**ve třech stavech**

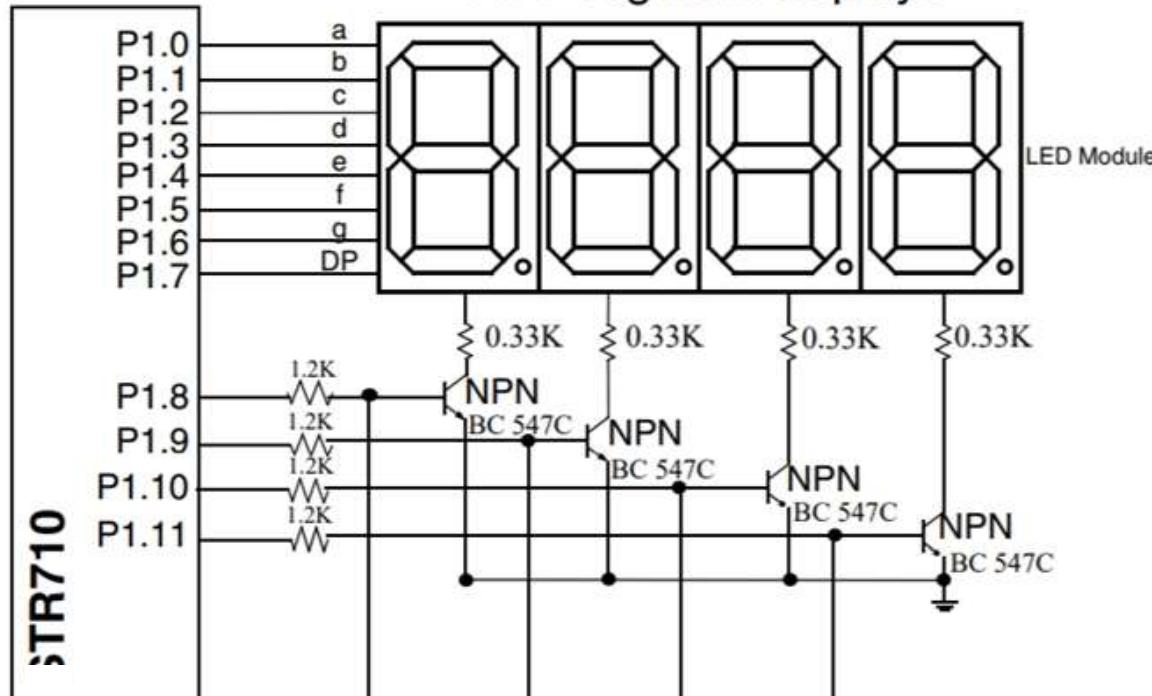
- log. 0 (na pinu je potenciál blízký GND)
- log. 1 (na pinu je potenciál blízký Vcc)
- pin je vstupní – má vůči zemi vysokou impedanci, nelze říci, zda je na něm 0 nebo 1 – pin očekává přivedení úrovně zvenku.

## Příklad užití celého portu: ovládání displeje



# Multiplexování výstupů

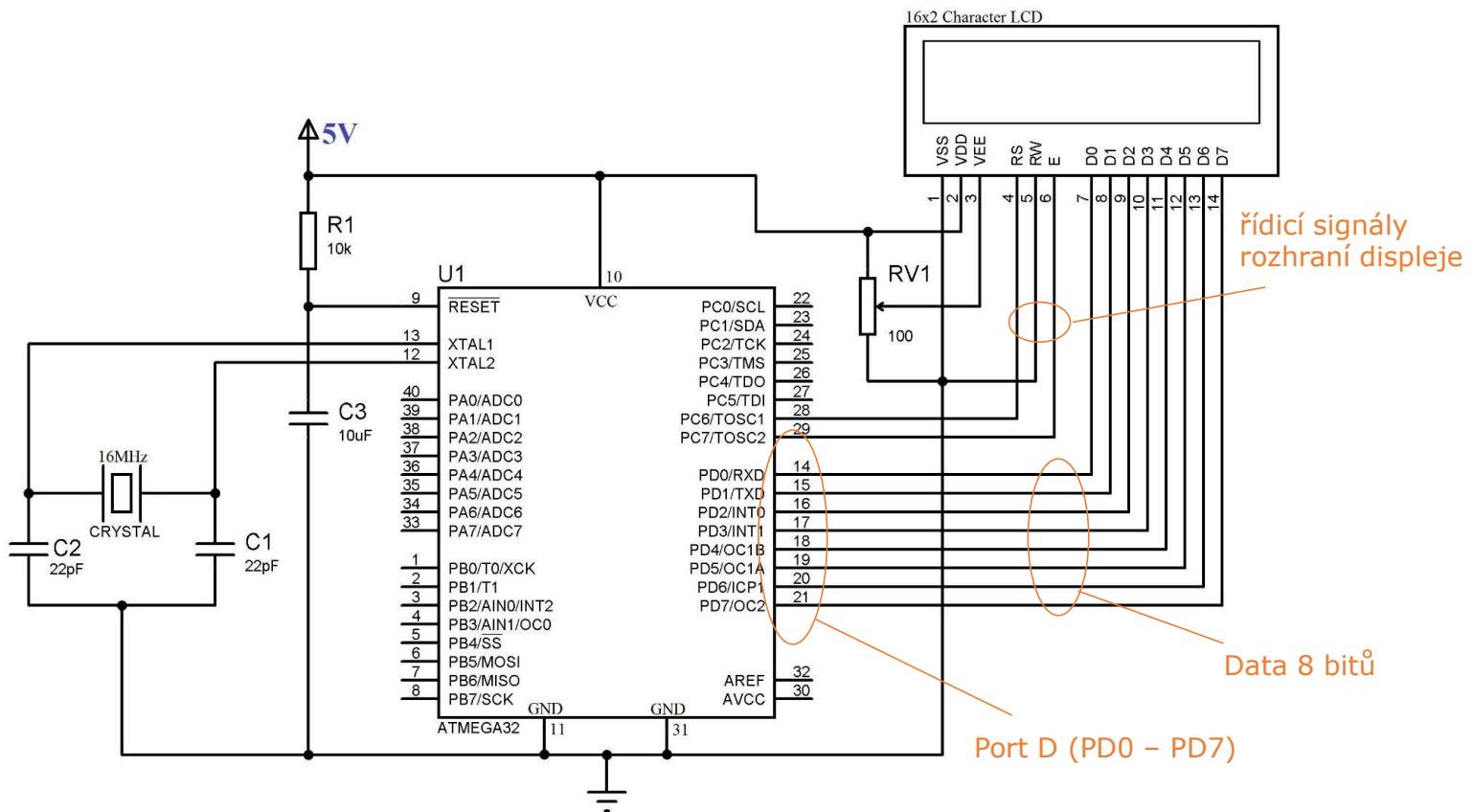
4 x 7-segment displays



Postupně se otevírá jeden z tranzistorů po druhém a na *a-f* se dávají úrovně odpovídající požadované číslici.  
To se děje rychle za sebou, dostatečně rychle na to, aby člověk nepostřehl postupné rozsvěcování,  
ale zase ne tak rychle, aby segmenty neměly čas se rozsvítit.

Uvědomte si, jaké hodnoty je třeba dávat na piny. Počítejte s tím, že svítící segmenty jsou diody, tedy svítí jen při správné kombinaci!

## Příklad užití celého portu: ovládání displeje

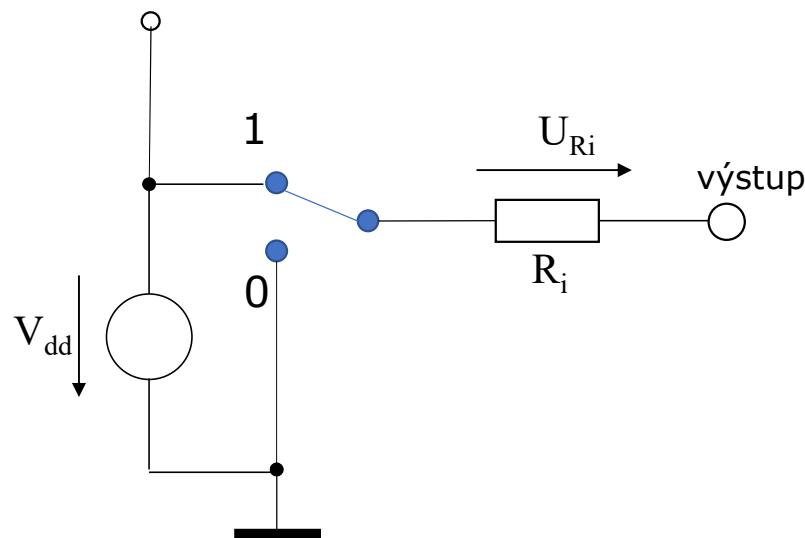


# Číslicový výstup

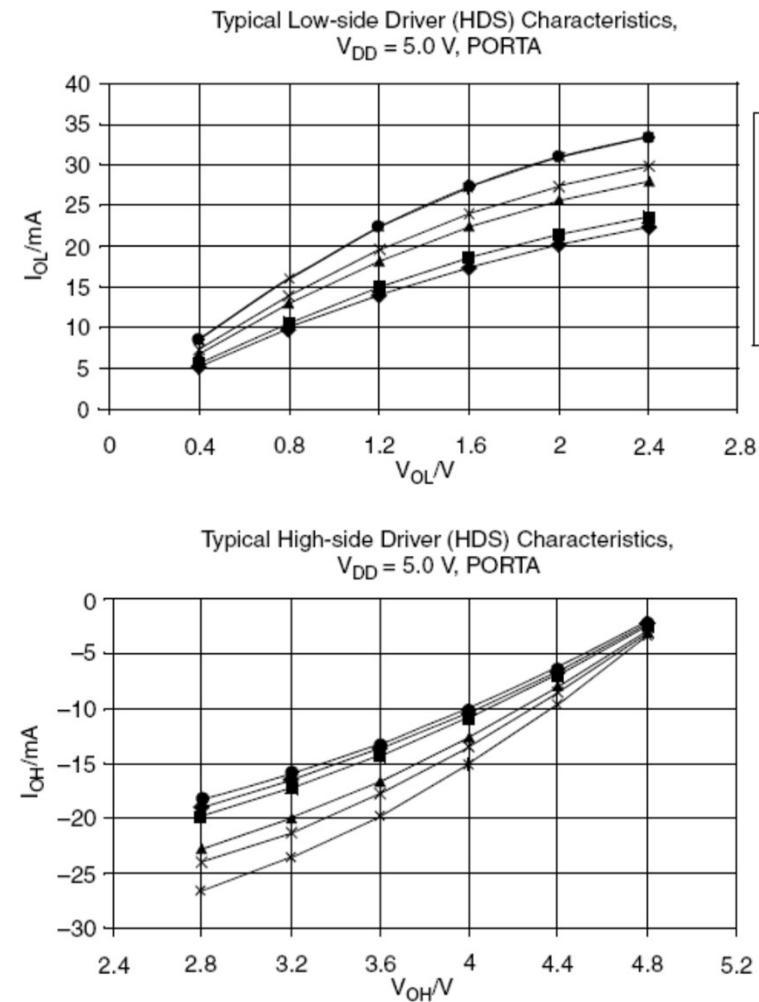
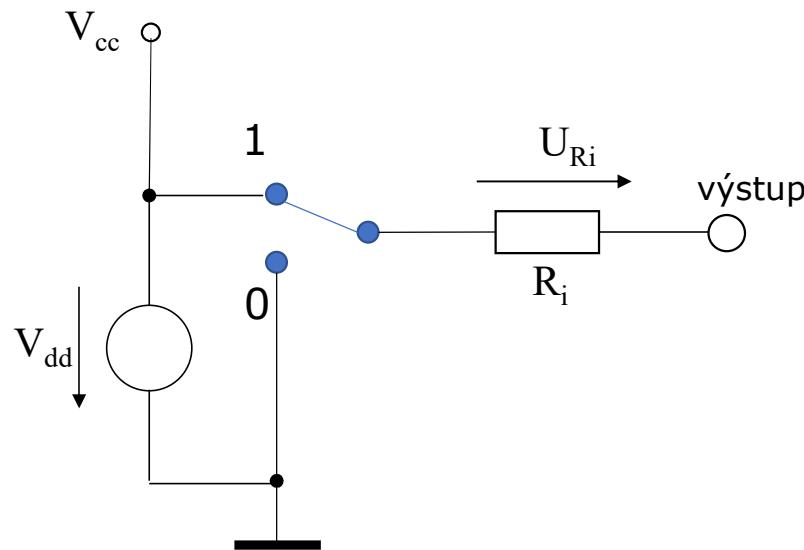
... není nic jiného, než připojení + nebo – pólu zdroje na výstup!

Musíme ale počítat s tím, že MCU je reálná součástka, nikoliv ideální!

- změna úrovně se neděje nekonečně rychle
- v obvodu jsou nenulové odpory



# Číslicový výstup

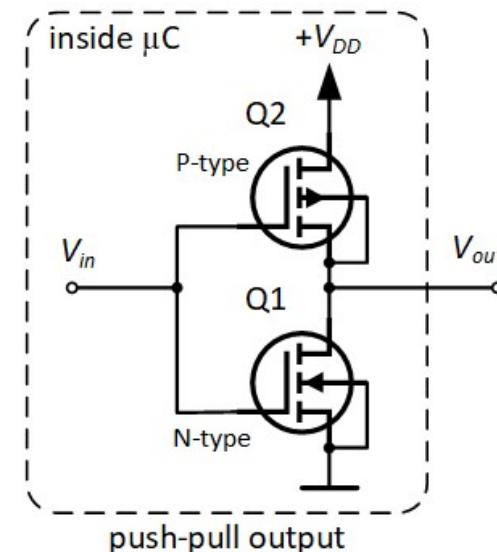
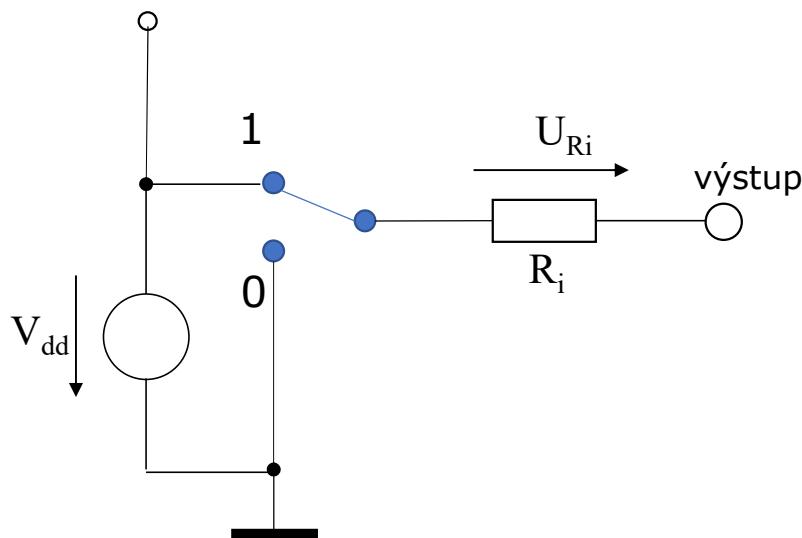


# Číslicový výstup

... není nic jiného, než připojení + nebo - pólu zdroje na výstup!

Musíme ale počítat s tím, že MCU je reálná součástka, nikoliv ideální!

- změna úrovně se neděje nekonečně rychle
- v obvodu jsou nenulové odpory



# Přechodný děj (změna úrovně)

## Drive Strength and Slew Rate



# Elektromagnetická kompatibilita

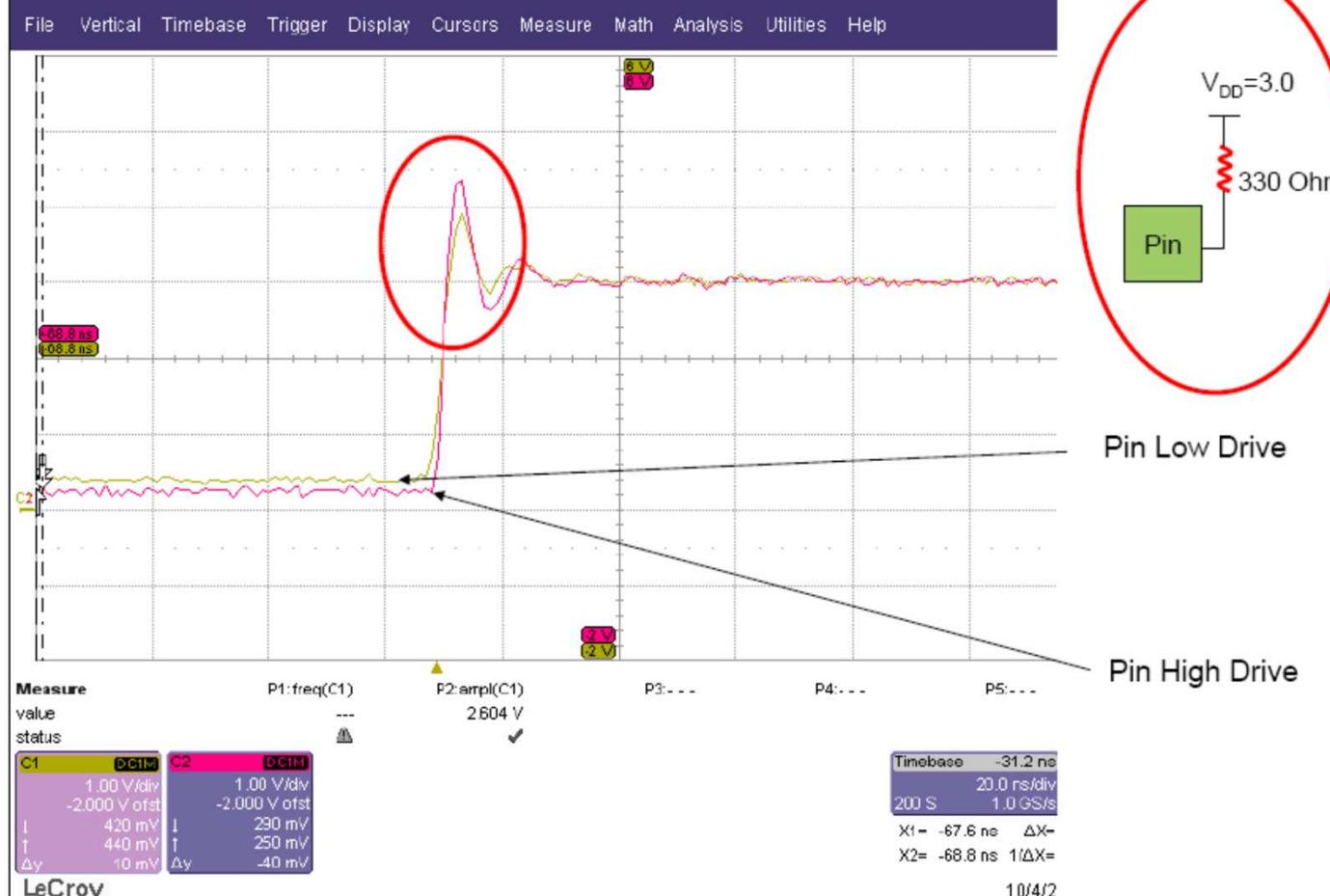
- Často je třeba, aby přechod mezi úrovněmi (hrana signálu) byl co nejrychlejší (strmá hrana).
- Strmá ostrá hrana zejména v kombinaci s dlouhým spojem a větším přenášeným výkonem ale může způsobit elektromagnetický impuls, který ovlivní okolní obvody (rušení).
- Obvykle ale naštěstí spínání větší zátěže nevyžadují tak strmou hranu a naopak.
- MCU dnes dovolují konfigurovat rychlosť přechodu úrovně (angl. **slew rate**) a také **vnitřní odpor pinu**, čímž lze omezit výkon dodaný výstupem vně MCU (angl. **drive strength**).

## Slew Rate Control Example

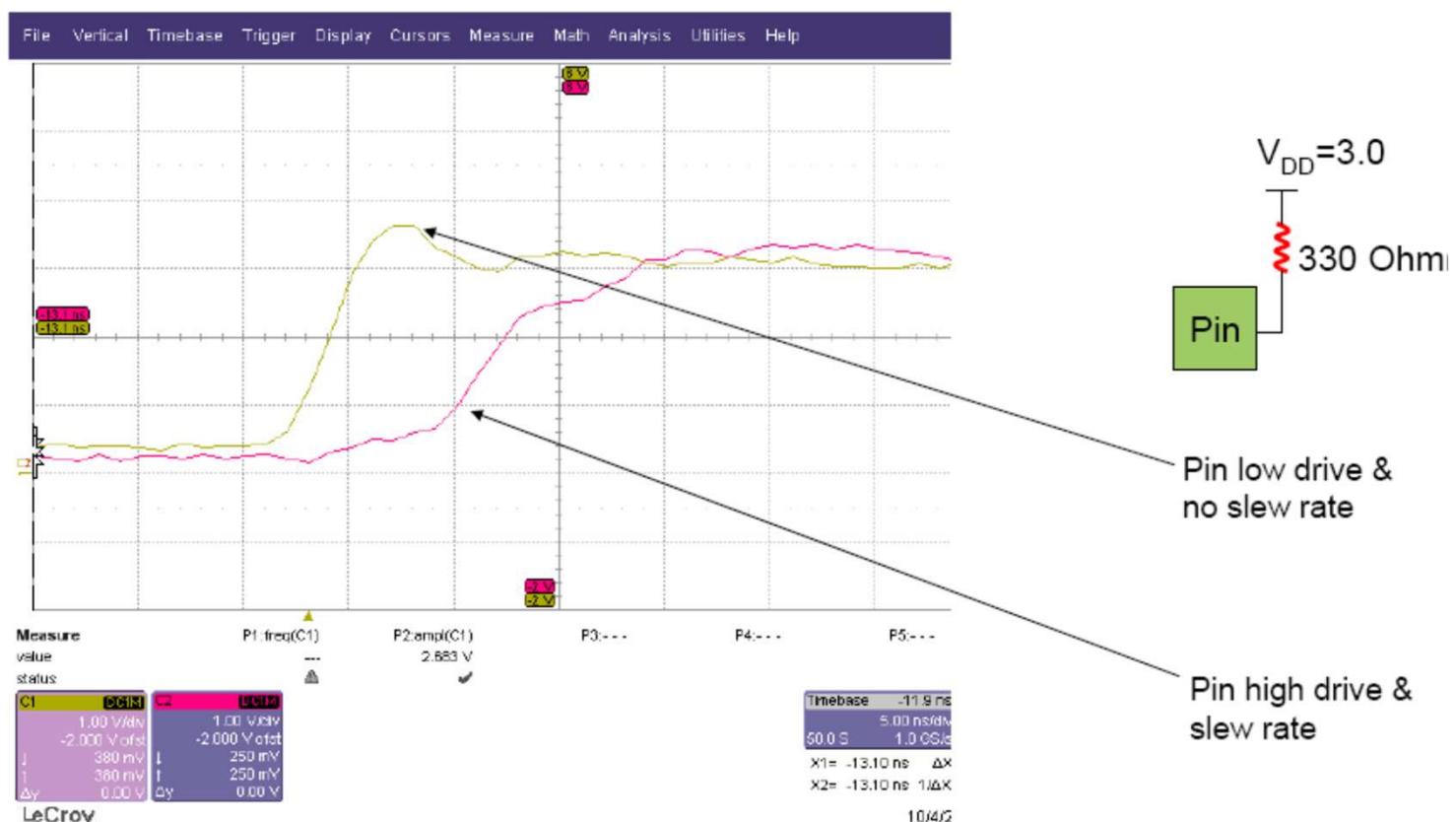


Volba „slew rate“ zlepšuje parametry EMC – méně rušení. Sama strmá hrana zvyšuje rušení. Pokud je velký překmit, zafunguje ochranná dioda a tak vyvolaný proud způsobí opět rušení. To se stává zejména u dlouhých vodičů či při malých blokovacích kapacitách.

# Impact of Drive Strength



## Drive Strength and Slew Rate



# ESD ochrana pinů

ESD = ElectroStatic Discharge

Ochrana proti  
napětí  $>V_{dd}$   
a  $<V_{ss}$  (GND)

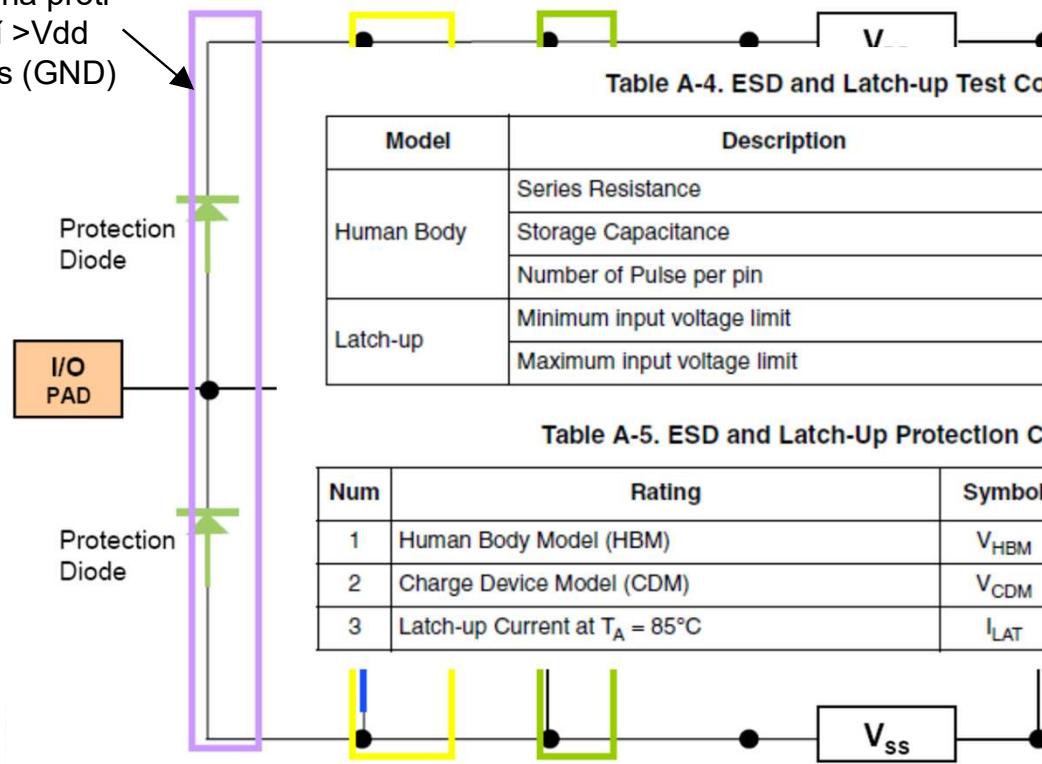


Table A-4. ESD and Latch-up Test Conditions

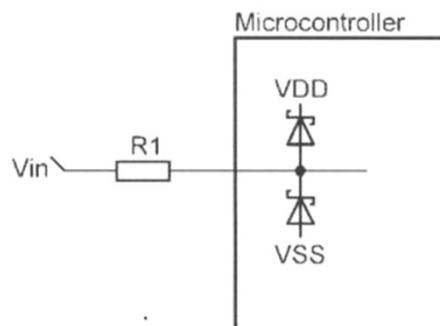
Model	Description	Symbol	Value	Unit
Human Body	Series Resistance	$R_1$	1500	$\Omega$
	Storage Capacitance	$C$	100	pF
	Number of Pulse per pin	—	3	
Latch-up	Minimum input voltage limit	—	-2.5	V
	Maximum input voltage limit	—	7.5	V

Table A-5. ESD and Latch-Up Protection Characteristics

Num	Rating	Symbol	Min	Max	Unit
1	Human Body Model (HBM)	$V_{HBM}$	$\pm 2000$	—	V
2	Charge Device Model (CDM)	$V_{CDM}$	$\pm 500$	—	V
3	Latch-up Current at $T_A = 85^\circ\text{C}$	$I_{LAT}$	$\pm 100$	—	mA

# Nestandardní využití ESD ochrany

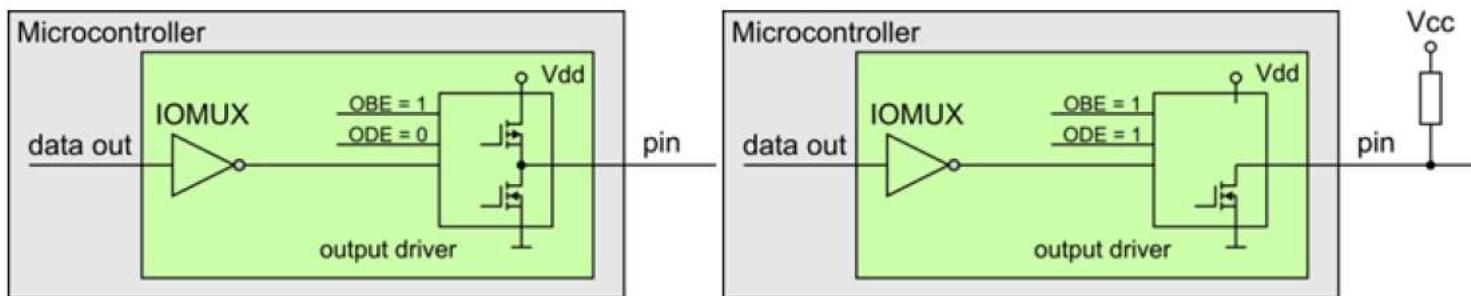
- ESD ochrany pinů lze někdy využít pro vstup signálu s vyšším napětím, než mají standardní logické úrovně. Je třeba ale postupovat opatrně!



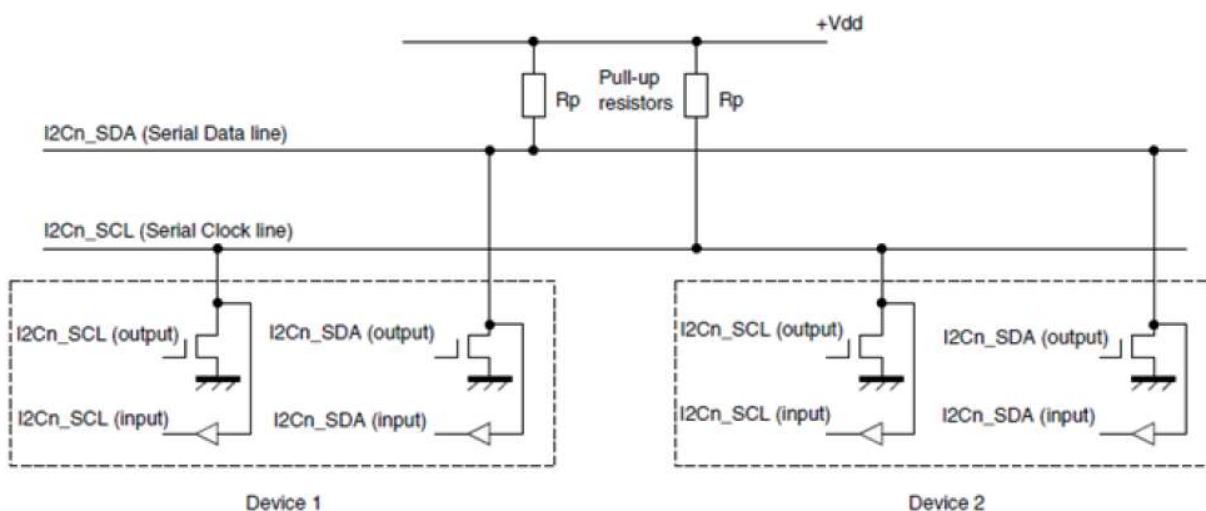
R1 je vypočten tak, aby nebyl překročen max. proud pinu.

POZOR! Ne všechny MCU mají ochranné diody na všech pinech!

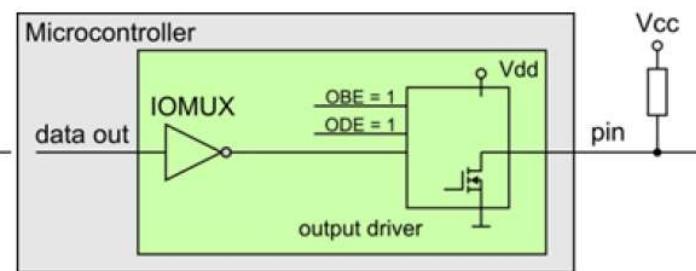
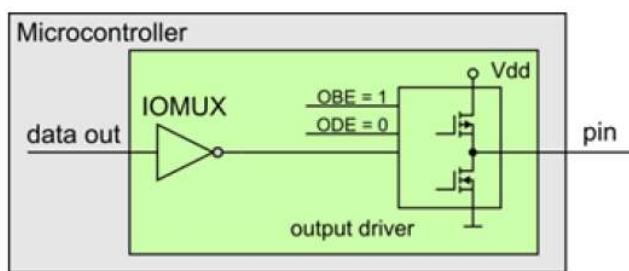
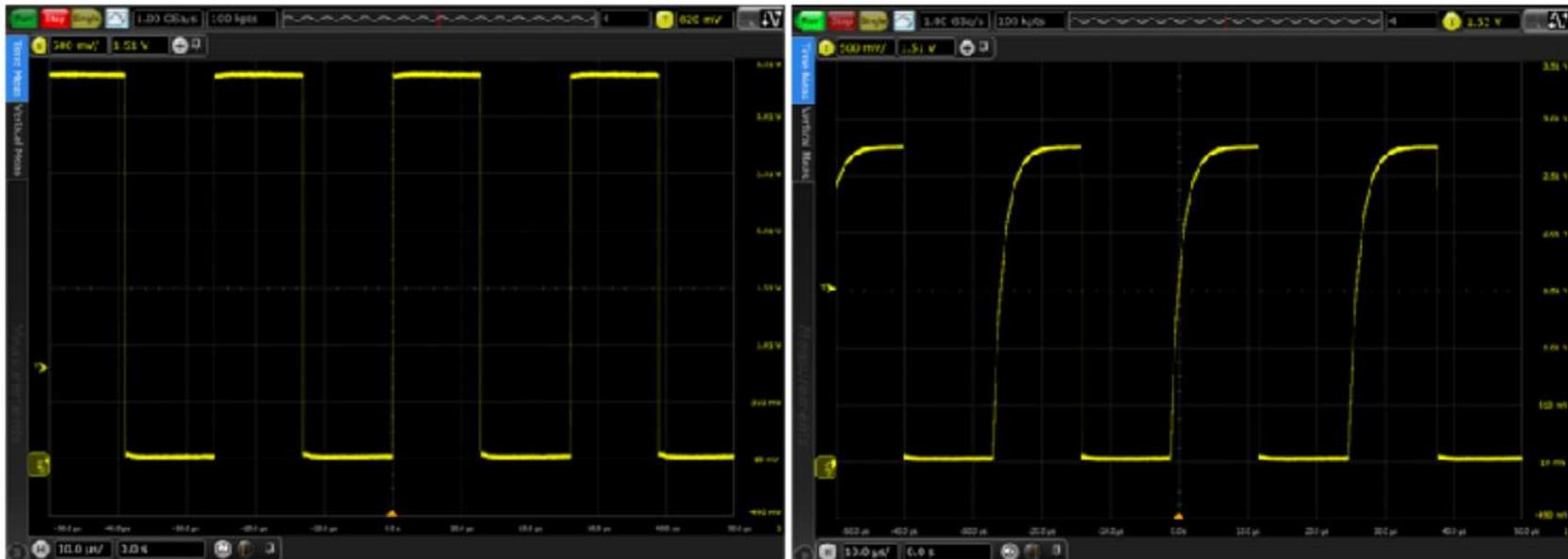
# Otevřený „kolektor“ (přesněji: „open drain“)



Typické použití: buzení sběrnice více zdroji – aby nedocházelo ke zkratům:



## Otevřený „kolektor“ – pozor na kapacitu



# Jaký proud může výstup dodávat?

Symbol	Description	Min.	Max.	Unit
$V_{DD}$	Digital supply voltage	-0.3	3.8	V
$I_{DD}$	Digital supply current	—	120	mA
$V_{IO}$	IO pin input voltage	-0.3	$V_{DD} + 0.3$	V
$I_D$	Instantaneous maximum current single pin limit (applies to all port pins)	-25	25	mA
$V_{DDA}$	Analog supply voltage	$V_{DD} - 0.3$	$V_{DD} + 0.3$	V

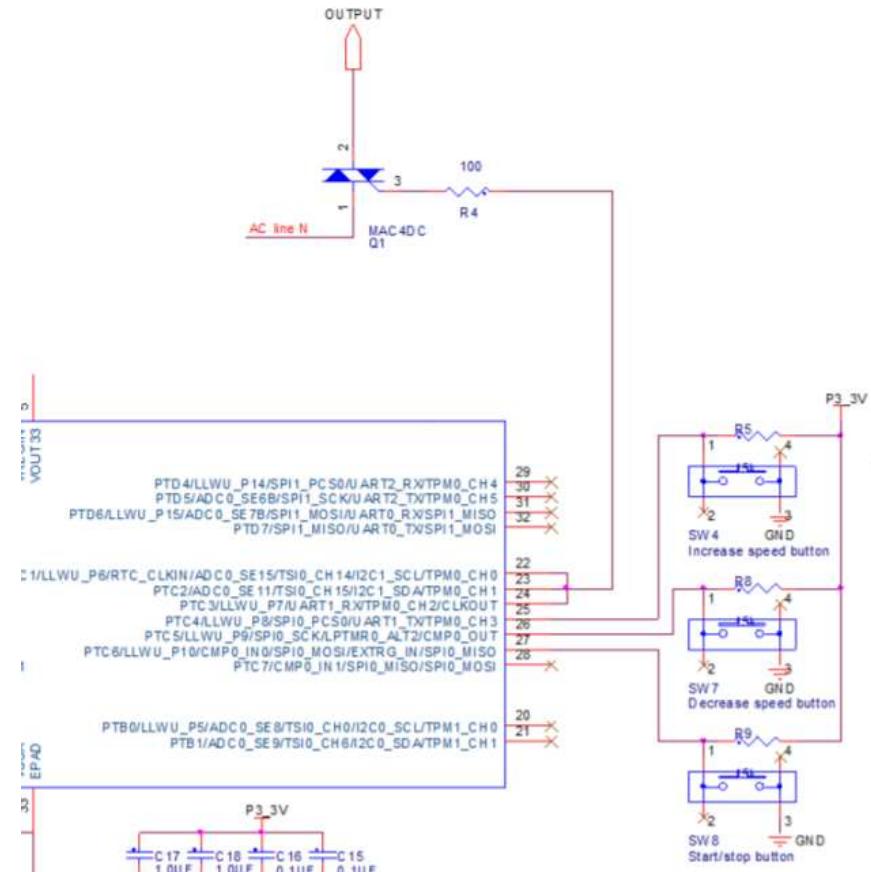
Symbol	Description	Min.	Max.	Unit	Notes
$I_{OHT}$	Output high current total for all ports	—	100	mA	
$I_{OLT}$	Output low current total for all ports	—	100	mA	

... to je tak na rozsvícení LEDky, pro náročnější spotřebiče je třeba externí budič.

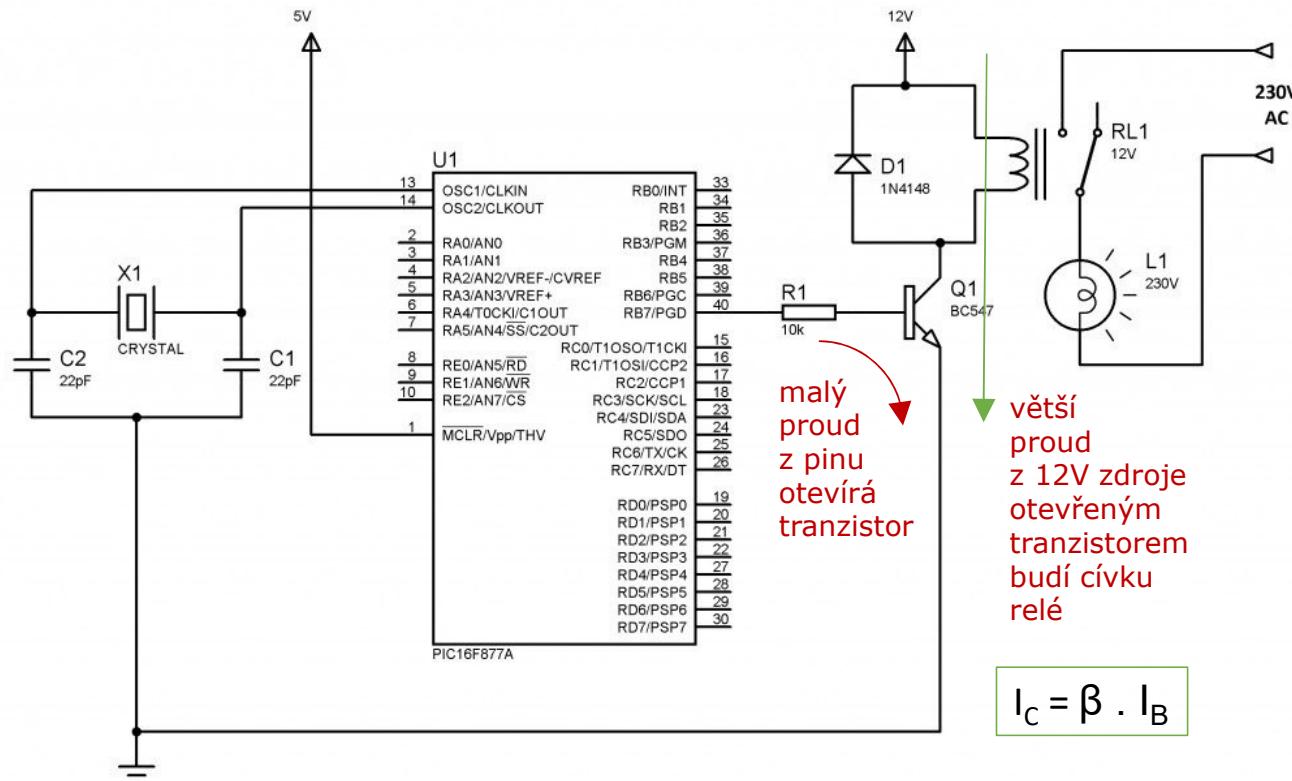
# Co když proud nestačí?

V některých případech pomůže sdružení pinů.

- Pozor na překročení celkového proudu!
- Změna stavu pinů musí být prováděna současně!



# Příklad: externí budič



# Číslicový vstup

... když v sw potřebujete vědět, co se děje „venku“.

# Číslicový vstup

- Obvody uvnitř MCU detekují, zda na pinu je **vůči zemi (GND)** potenciál
  - stejný → log. 0 nebo
  - vyšší – spíše blízký potenciálu Vcc → log. 1.

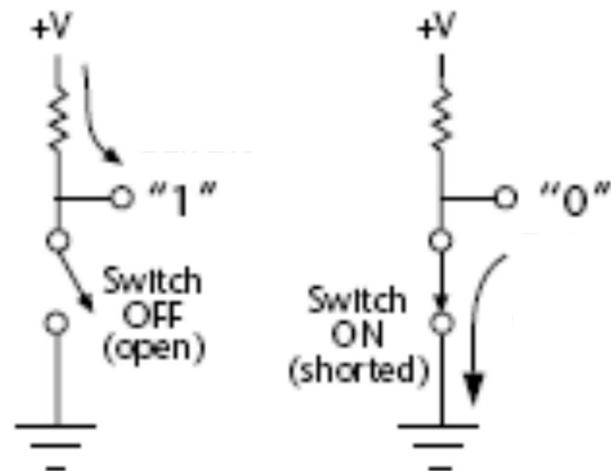
S tím souvisí otázky:

- Jak se má systém chovat, když není připojeno nic (pin je „ve vzduchu“)?  
→ programem by se mohlo nastavit, jaká tam bude úroveň
- Je třeba nějak reagovat na stav pinu či jeho změnu v software?  
→ mohlo by to spustit nějakou připravenou funkci
- Je vhodné signál nějak upravovat či tvarovat?  
→ potřebujeme filtrovat špičky či šum pomocí hystereze?



# Vstupní pin portu

- Dokáže snímat dva stavy (log. 0 a log. 1)
- Co se s tím dá dokázat?
  - snímání stavu kontaktu/spínače/tlačítka



- snímání stavu výstupu nějakého obvodu (čidla, ...)

# Jak se pozná 0 a 1 na vstupu?

Spolehlivé rozlišení logické 0 a 1 na vstupu:

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
Vih	High-level DC input voltage	–	0.7*ovdd	–	ovdd	V
Vil	Low-level DC input voltage	–	0	–	0.3*ovdd	

Uvědomte si, že signál, který přichází na pin MCU z „vnějšího světa“ nemusí být vždy ideální.

„Venku“ je analogový svět se svými šumy, zkreslením, odpory, kapacitami, indukčnostmi.

MCU se však musí rozhodnout, zda na pinu „vidí“ jedničku nebo nulu, protože v software neexistuje nic „mezi“.

## Co s tím z pohledu programátora?

- Stav pinu lze přečíst v nějakém registru, podle toho se pak program rozhodne, co udělá (if-then, ...),
- **Nejdůležitější otázka: kdy číst stav pinu/portu** = kam v programu umístit příkaz čtení registru?
- Změna stavu „něčeho vně MCU“ nastává asynchronně k běhu programu:
  - bud' čtu opakovaně tak dlouho, dokud nejistím změnu nebo
  - nechám HW aby zasáhl do provádění programu, jakmile změna nastane -> *přerušení*

# Přerušení od pinů

- Podpora generování přerušení od vstupních pinů:
  - aby program nemusel neustále testovat stav pinů.
  - lze nastavit, zda modul bude reagovat na hranu nebo na úroveň.

registr PORTx\_PCRn:

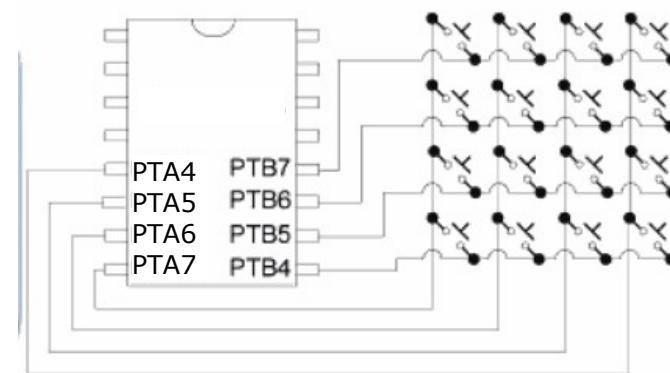
Address: Base address + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	
R	0							ISF	0					
W								w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2
R	0					MUX			0	DSE	0	PFE	0	SRE
W									*	*	0	*	0	PE
Reset	0	0	0	0	0	*	*	*	0	*	0	*	*	*

0000	Interrupt/DMA request disabled.
0001	DMA request on rising edge.
0010	DMA request on falling edge.
0011	DMA request on either edge.
1000	Interrupt when logic zero.
1001	Interrupt on rising edge.
1010	Interrupt on falling edge.
1011	Interrupt on either edge.
1100	Interrupt when logic one.
Others	Reserved.
IRQC	

# Příklad

- Snímání klávesnice:
  - PTB4 – PTB7 se nastaví jako výstupy, vyšle se určitá úroveň (např. log. 0).  
**Program nyní může na klávesnici „zapomenout“ a věnovat se jiné činnosti.**
  - V okamžiku, kdy je stisknuto některé tlačítko, dostane se přes ně úroveň log. 0 na vstup PTAx a vyvolá se přerušení.
  - V obslužné rutině přerušení se může ve 4 krocích vyzkoušet, ve které řadě je stisknuté tlačítko.



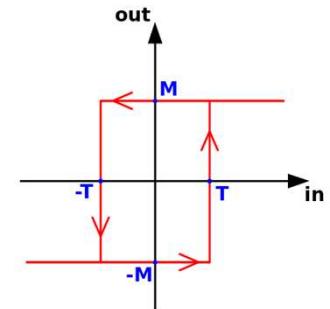
# Omezení běžného číslicového vstupu

Nepřekročitelná maxima (překročení vede ke zničení MCU):

Symbol	Description	Min.	Max.	Unit
$V_{DD}$	Digital supply voltage	-0.3	3.8	V
$I_{DD}$	Digital supply current	—	120	mA
$V_{IO}$	IO pin input voltage	-0.3	$V_{DD} + 0.3$	V
$I_D$	Instantaneous maximum current single pin limit (applies to all port pins)	-25	25	mA
$V_{DDA}$	Analog supply voltage	$V_{DD} - 0.3$	$V_{DD} + 0.3$	V

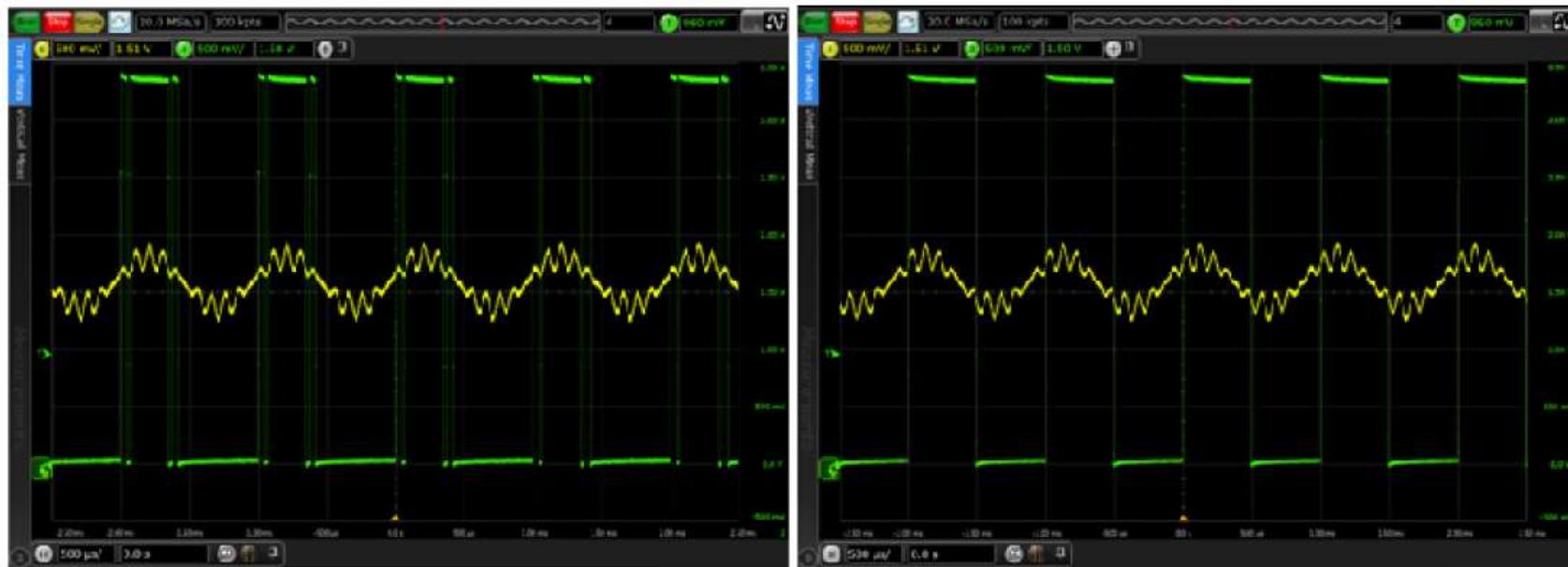
Na pinu jsou sice většinou ochranné diody (ESD), ale trvalé překročení jejich mezí vede nakonec k jejich zničení. Pak dostanou příkon (který neunesou) struktury uvnitř MCU.

# Hystereze



Některé MCU mohou mít na vstupu Schmittův klopný obvod.

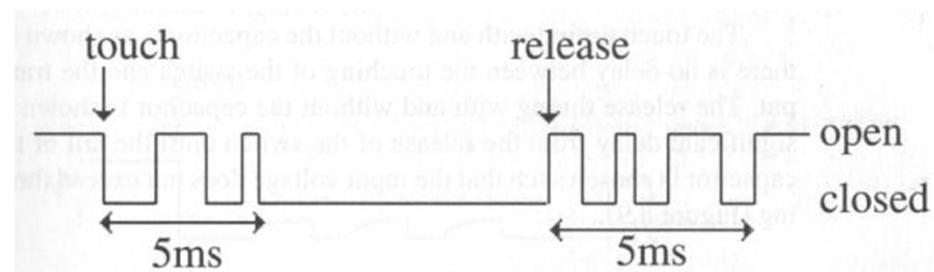
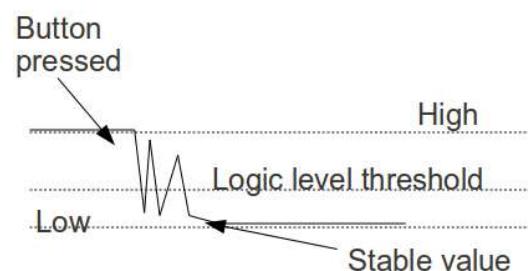
Přechod  $0 \rightarrow 1$  na vstupu se zaznamená, až když vstupní signál překoná hranici např.  $\frac{1}{2}V_{cc} + 125\text{mV}$ , z  $1 \rightarrow 0$  pak až signál klesne pod  $\frac{1}{2}V_{cc} - 125\text{mV}$ . Rozpětí  $250\text{mV}$  mezi prahy pro směr  $0 \rightarrow 1$  a  $1 \rightarrow 0$  pomůže odfiltrovat šum.



# Problém mechanického kontaktu - zákmity

Při sepnutí či rozepnutí kontaktu vždy dochází k přechodovému ději – zakmitávání kontaktu, takže z pohledu MCU dojde k několikerému rychlému rozepnutí a sepnutí.

- Řešení:
  - **softwarové** – při první detekci změny stavu kontaktu se chvíli stav kontaktu ignoruje, dokud se neustálí.
  - **hardwareové** – kontakt se řeší jako přepínač a klopným obvodem se zachytí a podrží úroveň nebo použití RC filtru a hystereze.



## SW řešení problému zákmitů kontaktu

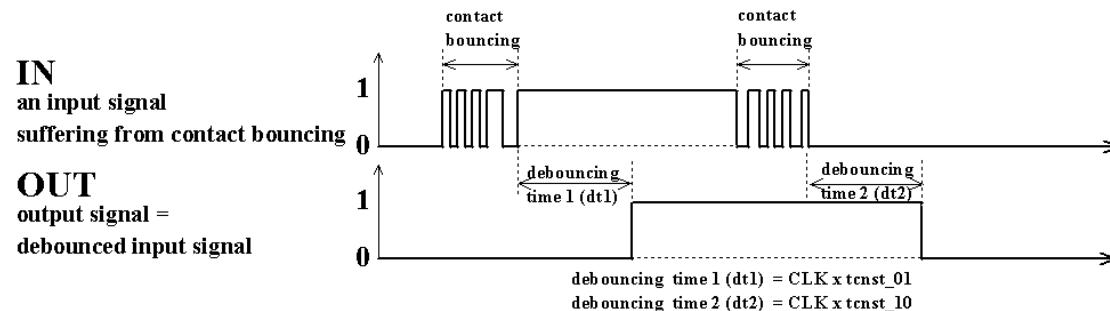
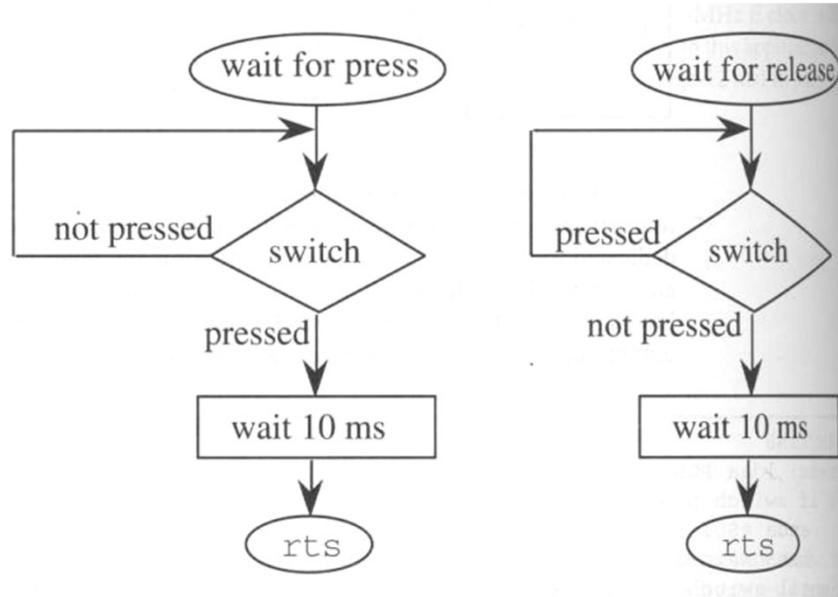
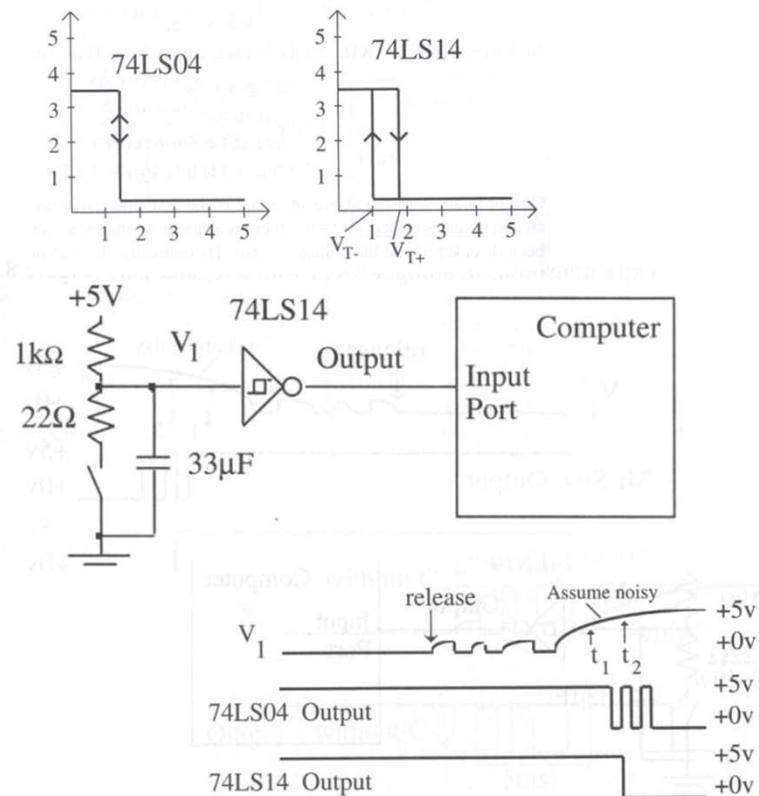


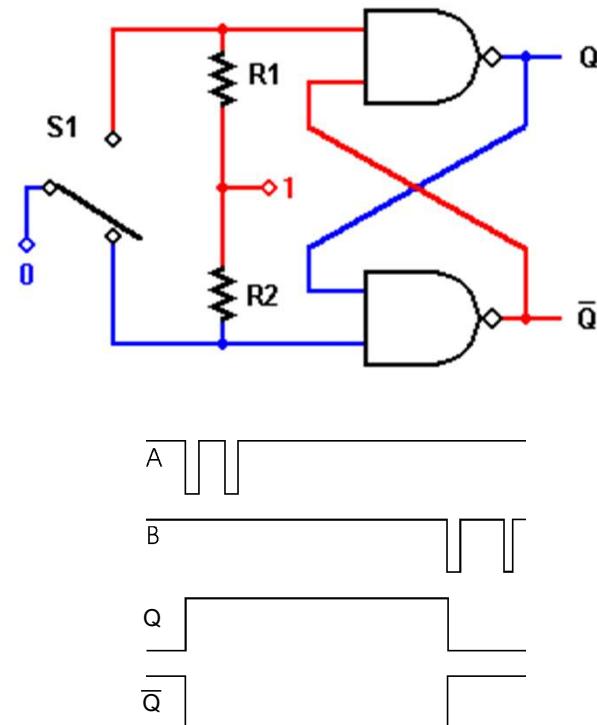
Figure 5. Timing diagram of each channel of the generic independent eight bit I/O contact debouncer.

# HW řešení problému zákmitů kontaktu

„RC filtr“ + hradlo s hysterezí.



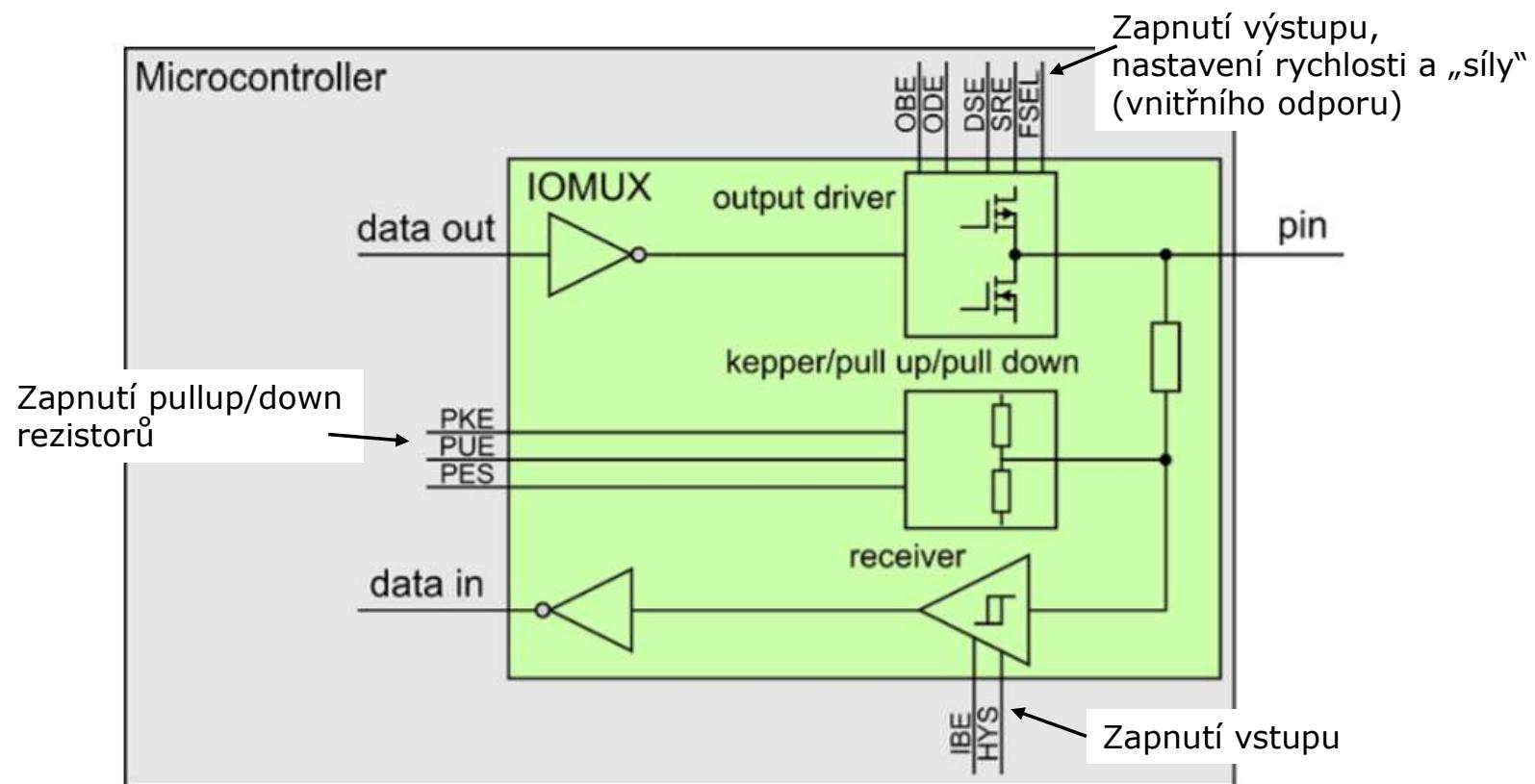
RS klopný obvod, přepínací kontakt.



# Připojovací pin moderního MCU a s ním související nastavení

(sw konfigurace GPIO pro přizpůsobení konkrétní aplikaci)

# Jak to vypadá uvnitř MCU?



# Nastavení pinu:

## registrov PORTx\_PCRn pro každý pin

Address: Base address + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					0				ISF			0				
W								w1c					IRQC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					0				MUX			0	DSE	0	PFE	0
W													SRE	PE	PS	0
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

000 Pin disabled (analog).  
 001 Alternative 1 (GPIO).  
 010 Alternative 2 (chip-specific).  
 011 Alternative 3 (chip-specific).  
 100 Alternative 4 (chip-specific).  
 101 Alternative 5 (chip-specific).  
 110 Alternative 6 (chip-specific).  
 111 Alternative 7 (chip-specific).

0 Low drive strength is configured on the corresponding pin.  
 1 High drive strength is configured on the corresponding pin.

0 Passive input filter is disabled on the corresponding pin.  
 1 Passive input filter is enabled on the corresponding pin.

0 Fast slew rate  
 1 Slow slew rate

0 Internal pullup or pulldown resistor is not enabled on the pin.  
 1 Internal pullup or pulldown resistor is enabled on the pin.

Internal pulldown resistor is enabled on the pin.  
 Internal pullup resistor is enabled on the pin.

# Hromadné nastavení konfigurace pinů portu

## Registry PORTx\_GPLCR a PORTx\_GPHCR

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R																0																0	
W																	GPWE															GPWD	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**PORTx\_GPCHR field descriptions**

Field	Description
31–16 GPWE	Global Pin Write Enable  Selects which Pin Control Registers (31 through 16) bits [15:0] update with the value in GPWD.  0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.
15–0 GPWD	Global Pin Write Data  Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.

# Testování, zda některý pin vyvolal přerušení

## Registr PORTx\_ISFR

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**PORTx\_ISFR field descriptions**

Pozor při zápisu! Write 1 to Clear!  
Proč to tak je? Smazání příznaku může být atomická operace.

Field	Description
31–0 ISF	Interrupt Status Flag  Each bit in the field indicates the detection of the configured interrupt of the same number as the field.  0 Configured interrupt is not detected. 1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic one is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared.

... není třeba číst všechny registry PORTx\_PCRn – stačí tento jeden, a je jasno, zda a který pin portu vyvolal přerušení.

# Nastavení pinu: výjimky

I když všechny piny mají svůj registr, **ne u všech jsou všechna nastavení platná!**

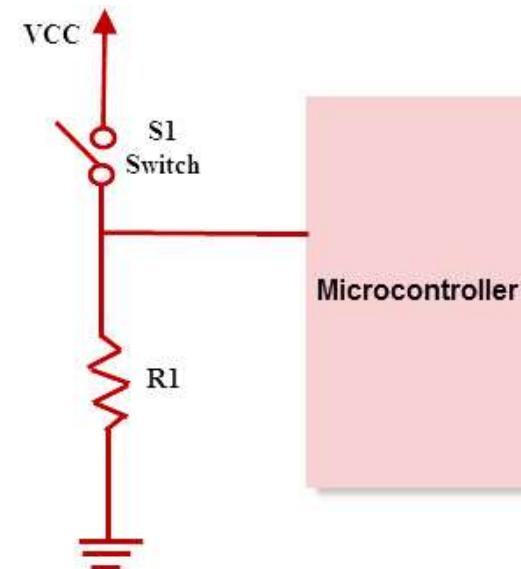
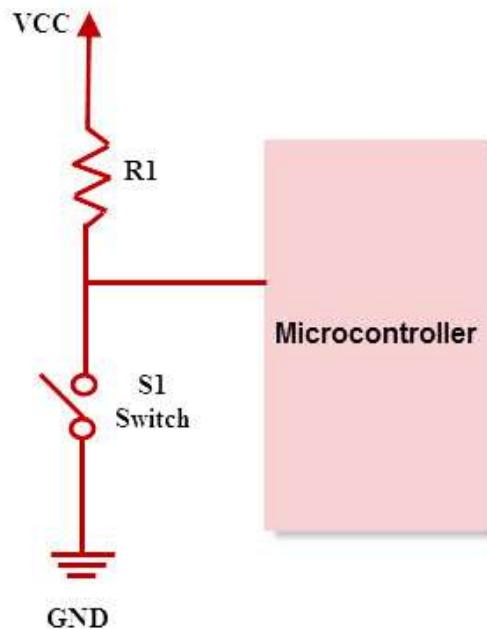
Je třeba se pečlivě podívat do dokumentace ke konkrétnímu MCU, zda požadované nastavení pinu je možné! Zejména u levnějších a jednodušších variant výrobce šetří.

Například pro náš MKL05Z:

- nemá pulldown odpory (kromě RESET pinu),
- high-drive strength lze nastavit jen pro PTB0, PTB1, PTA11, PTA 12,
- vstupní filtr má jen NMI (PTB5).

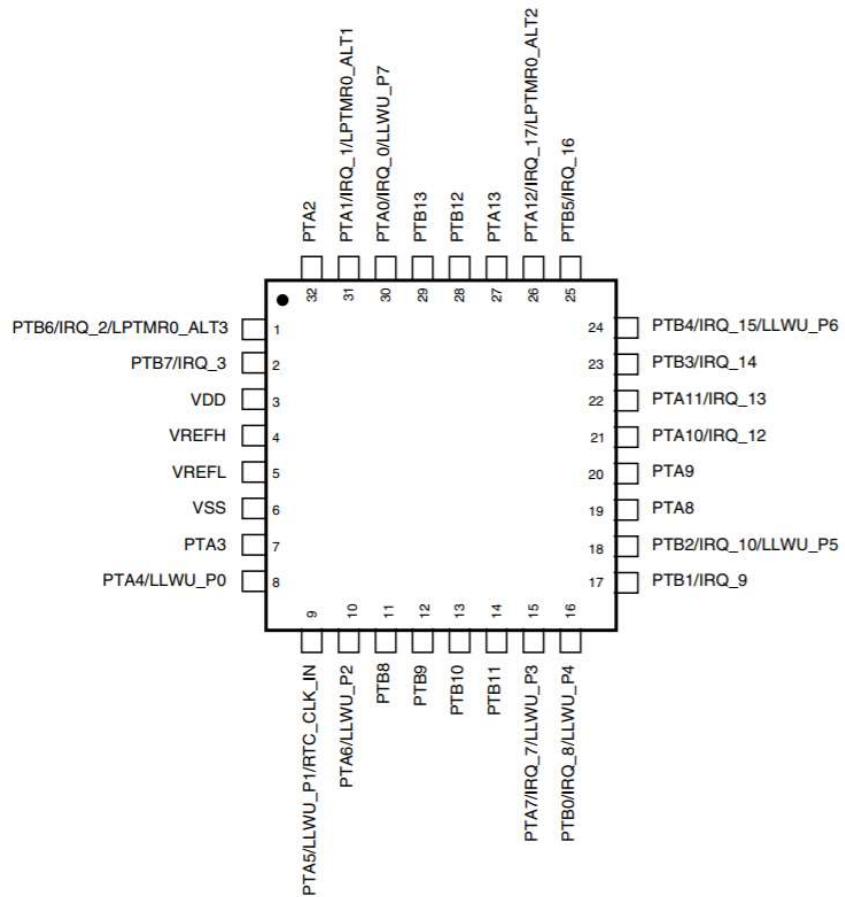
# Pull-up/pull-down rezistor

- Slouží pro definování úrovně na pinu, na němž není připojeno nic.



Typická situace je připojení jednoduchého kontaktu (tlačítka).

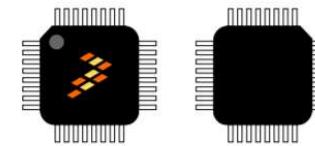
# Pouzdro MCU a jeho vývody



Příklad - náš

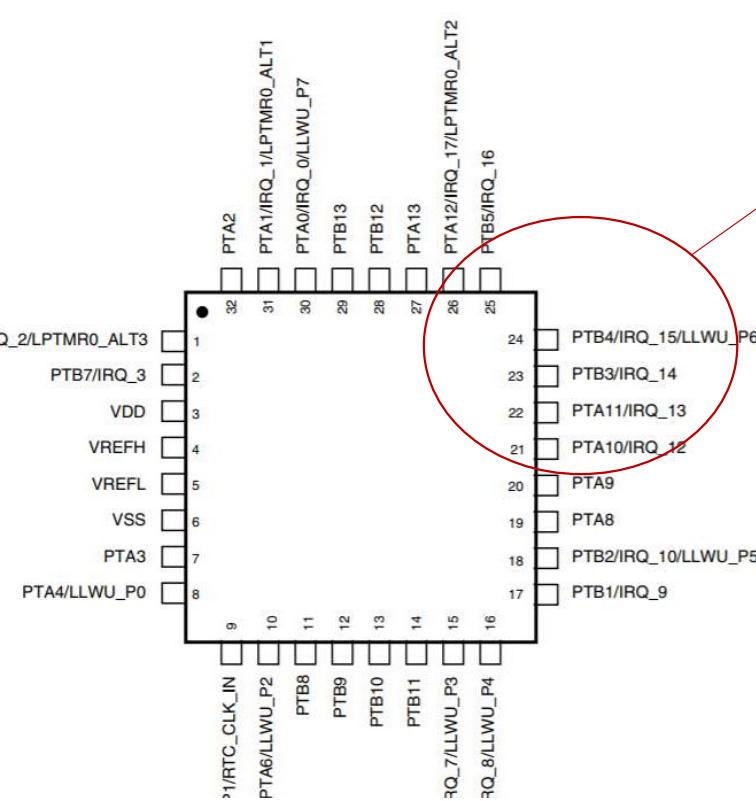
**MKL05Z32VLC4,**

který máme na kitu v laboratoři



32-pin LQFP (LC)  
7 x 7 x 1.4 Pitch 0.8  
mm

# Pouzdro MCU a jeho vývody



48 LQFP	32 QFN	32 LQFP	24 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3
33	21	21	—	PTA10/ IRQ_12	DISABLED	TSI0_IN11	PTA10/ IRQ_12		
34	22	22	—	PTA11/ IRQ_13	DISABLED	TSI0_IN10	PTA11/ IRQ_13		
35	23	23	17	PTB3/ IRQ_14	DISABLED	DISABLED	PTB3/ IRQ_14	I2C0_SCL	UART0_TX
36	24	24	18	PTB4/ IRQ_15/ LLWU_P6	DISABLED	DISABLED	PTB4/ IRQ_15/ LLWU_P6	I2C0_SDA	UART0_RX
37	25	25	19	PTB5/ IRQ_16	NMI_b	ADC0_SE1/ CMP0_IN1	PTB5/ IRQ_16	TPM1_CH1	NMI_b
38	26	26	20	PTA12/ IRQ_17/ LPTMR0_ALT2	ADC0_SE0/ CMP0_IN0	ADC0_SE0/ CMP0_IN0	PTA12/ IRQ_17/ LPTMR0_ALT2	TPM1_CH0	TPM_CLKIN0

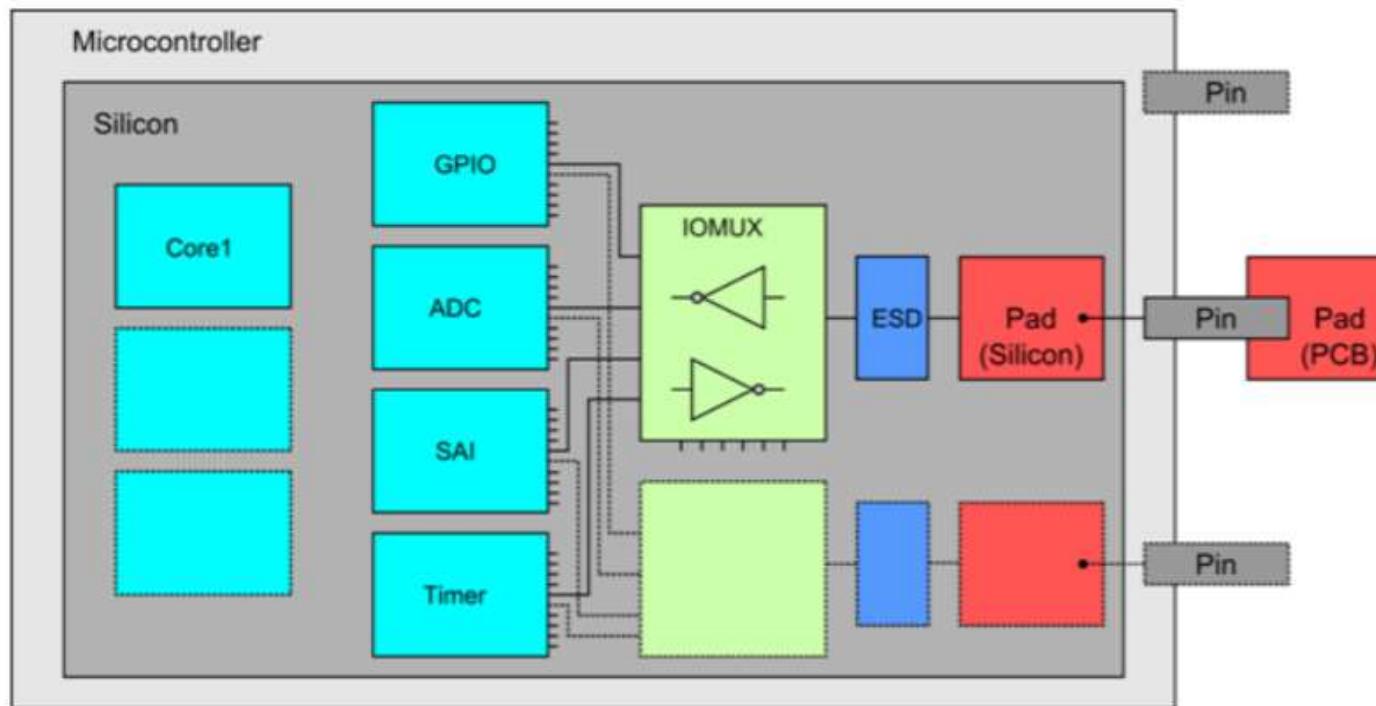
Povšimněte si, kolik významů může mít každý pin!

→ až 5 různých!

Jde o to, co uvnitř MCU bude k pinu připojeno.

To lze nastavit programem.

## Konfigurace významu (funkce) pinu MCU



# Jak nastavit MCU pro využití GPIO?

## 1. Zapnout hodiny modulu PORTu

```
SIM->SCGC5 = (SIM_SCGC5_PORTB_MASK);
```

## 2. Nastavit konfiguraci příslušného pinu

```
PORTB->PCR[0] = (0 | PORT_PCR_MUX(0x01));
```

## 3. Nastavit směr pinu

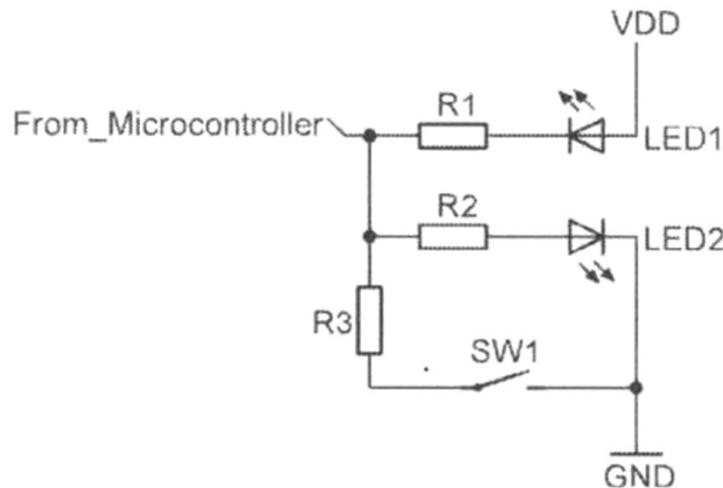
```
PTB->PDDR = 0x0001;
```

## 4. Zapisovat/číst hodnotu

...registry PDOR, PDIR, PSOR, PCOR, PTOR

# A ještě jedna zajímavost

- Využití pinu na maximum.



LED1 svítí pro log. 0  
LED2 svítí pro log. 1

aby nesvítila ani jedna, nastaví se port jako vstupní, pak je možné číst tlačítko.

Toto zapojení využívá skutečnosti, že je možné velmi rychle změnit směr pinu. Pokud má pin zároveň fungovat jako vstup i výstup, je třeba měnit směr dostatečně často, aby člověk nepoznal, že LED blikají.

Odpór R3 je dobré volit tak velký, aby stisknutí SW1 nerozsvítilo LED1, ale ne tak velký, aby MCU nedetekoval stisk tlačítka. Jeho velikost závisí především na velikosti pull-up rezistoru v MCU (pro HCS08 je vhodný R3 kolem 15k).