



Kompatibilní zapojení: [LED panel s ATmega8](#)

Ke stažení: [LED 007.c](#) == [LED 007.pdf](#) == [LED 007.htm](#)

.

Untitled <!- body { color: #000000; background-color: #FFFFFF; }
.cpp1-assembler { } .cpp1-brackets { } .cpp1-comment { color: #008000; font-style: italic; }
.cpp1-float { color: #000080; } .cpp1-hexadecimal { color: #000080; } .cpp1-character { }
.cpp1-identifier { } .cpp1-illegalchar { } .cpp1-number { color: #000080; } .cpp1-octal { color:
#0000FF; } .cpp1-preprocessor { } .cpp1-reservedword { font-weight: bold; } .cpp1-space {
color: #008080; } .cpp1-string { color: #800000; } .cpp1-symbol { } --> /* Nyní již
umíme používat příkazy k větvení programu (podmínky) "if" a "switch". Umíme také rozložit
program na jednoduché funkce a používat cyklus "for". Co se týče cyklů, zbývá nám ještě
vysvětlit cyklus "while" a "do-while" a příkazy "break" a "continue". /* /* Cyklus "while" je velmi
podobný cyklu "for". Rozdíl mezi cyklem "for" a "while" je ten, že cyklus "for" má předem daný
počet opakování, cyklus "while" jej předem daný mít nemusí. Funguje to tak, že cyklus "while"
má ve své hlavičce zadanou podmínku, která se při každém průchodu cyklem testuje. Jakmile
tedy program narazí na cyklus "while", otestuje jestli platí podmínka (podobně jako u "if") a
pokud platí, vykoná příkazy ve svém těle. Poté se program přesune zpět na začátek, a znovu
otestuje podmínku. Pokud je podmínka vyhodnocena jako neplatná, příkazy v těle cyklu se
nevykonají. Rozdíl mezi "for" a "while" tedy spočívá v tom, že pokud bude podmínka
vyhodnocena jako neplatná hned napoprvé, příkazy v těle cyklu se neprovedou vůbec. Zápis je
následující: while (podmínka) { příkazy; } /* /* Cyklus "do-while" se od cyklu "while" liší
tím, že příkazy v těle cyklu "do-while" se minimálně jednou provedou. Zápis je následující: do
{ příkazy; } while (podmínka); Podmínka je u tohoto cyklu zapsána až na konci, a program
tedy před jejím otestováním musí vykonat příkazy v těle cyklu. Na konci cyklu narazí na
podmínku, kterou otestuje. Pokud je podmínka platná, cyklus se opakuje (od začátku), pokud
ne, je cyklus ukončen, a program pokračuje za cyklem. /* /* S příkazem "break" jsme se
setkali již u podmíněného příkazu "switch". Podobně jako u příkazu "switch", se i u cyklů dá
použít příkaz "break" k jejich okamžitému ukončení. Příkaz break se většinou zadává jako
jedna větev podmínky ("if" nebo "switch"). Zápis: while (1) { příkazy; if (podmínka)
break; } V tomto případě příkaz break slouží k "vyskočení" z nekonečné smyčky. Příkazy v
těle smyčky se tedy budou vykonávat do té doby, dokud nebude platná podmínka "if". Ve výše
uvedeném příkladu si lze také povšimnout, že příkaz "break" není zapsán ve složených
závorkách, jak by se v těle příkazu "if" dalo očekávat. Pokud totiž do těla nějaké podmínky
nebo cyklu zapisuje pouze jeden příkaz, není nutné jej psát do složených závorek {}. Pokud
bychom do těla nějaké podmínky nebo smyčky zapsali více příkazů bez složených závorek,
bral by se jako tělo pouze první příkaz. Ostatní by se vykonaly až za tímto cyklem nebo
podmínkou. Například: if (n<5) n++; b=n; je totéž co: if (n<5) { n++; } b=n;

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

`*/ /*` Příkaz "continue" způsobí skok na začátek smyčky. Neukončí tedy smyčku jako "break", ale pouze vynechá všechny příkazy, které po něm následují až do konce těla smyčky. Cyklus pak pokračuje zase od začátku, jako by se nic nedělo. Zápis: `while (1) { příkazy; if (podmínka) continue; příkazy; //tyto příkazy budou vynechány, pokud bude platit podmínka }` `*/ /*` Doteď jsme se zabývali, jak na našem přípravku "ATmega8_LED_start_2" rozsvěcovat a zhasínat ledky. Co když ale budeme chtít řídit jejich jas? Jas ledky je možné řídit tak, že ji rozsvěcujeme a zhasínáme tak rychle, že to lidské oko nepostřehne. Když potom budeme různě měnit periodu světla a tmy, bude se zdát, že ledka svítí více či méně jasně. Tato metoda se označuje PWM a je to vlastně jeden ze způsobů, jak převést digitální signál (nuly a jedničky) na analogový (jas žárovky, výkon motoru, aj.). `*/ /*` Můžeme si tedy vyzkoušet, jak bude regulace jasu fungovat u našeho starého známého hada. Program had bude podobný jako v minulých programech, pouze místo příkazů: `PORTB=...; _delay_ms(...);` zaměníme za funkci: `rozsvit(...)`. Funkce "rozsvit()" bude zajišťovat rychlé blikání ledek (viz výše). Délku periody světla vůči periodě tmy, bude tato funkce přebírat z proměnné "jas". `*/ /*` Program by mohl vypadat třeba takto: `*/`

```
#define F_CPU 1000000UL
// 1 MHz (základní frekvence) kvůli delay.h
#include <avr/io.h>
//Knihovna vstupů a výstupů (PORT, DDR, PIN)
#include <util/delay.h>
//Knihovna čekacích funkcí
#include <avr/interrupt.h>
//Knihovna přerušení (kvůli vektoru ISR(TIMERO_OVF_vect))
#define VZOR 0b00000111
//tato konstanta určuje délku hada
#define DOBA 50
//tato konstanta ovlivňuje rychlost hada
```

```
//(čas na jeden krok v ms)
```

```
volatile
unsigned
char
jas=
64
;
```

```
/*tato proměnná je globální, protože se nastavuje v přerušení (čtení tlačítek) a zároveň ji
potřebujeme číst ve funkci "rozsvit". V této proměnné bude uloženo číslo (0-128), které bude
určovat jas ledek.*/
/*****
/* * Deklarace funkcí * */
*****/
```

```
*****/
/* Deklarace funkce je vlastně hlavička funkce, se
středníkem. Hlavička se píše tak jak je uvedena v definici. Pomocí deklarace funkce, dáváme
překladači najevo, že daná funkce existuje. Od tohoto místa dále tedy můžeme funkci volat, i
když je ve skutečnosti nadefinovaná až na konci programu. */
void
```

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

```
rozsvit(
unsigned
char
co,
unsigned
char
jakdlouho);
/* Nadeklarujeme si funkci, která zařídí regulaci jasu ledek, jak bylo popsáno výše. Tato
funkce bude přebírat parametry: "co" = ledky které budou svítit, "jakdlouho" = jak dlouho
budou svítit [ms] Dále tuto funkci ovlivňuje globální proměnná "jas", která udává jas ledek v
procentech (0 - 128) */
/* ***** */
/* ***** */
/* */
/* Hlavní funkce */
/* ***** */

/* ***** */
int
main
(
void
) {
unsigned
char
had=
0
;
//nadefinujeme si proměnnou, ve které bude uložen had
DDRB
=
0xff
;

//všechny piny na portu "B" budou výstupní /*Nyní si nastavíme přerušení od časovače "0" -
bude časovat čtení klávesnice*/
TIMSK|=
1
;

//nastavíme nultý bit na "1" (vybereme přerušení od časovače "0")
SREG
|=
(
1
<<
7
);

//"globální" povolení přerušení (nastavíme 7. bit registru SREG)
TCCR0
```

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

```
=
5
;

//Zapneme časovač "0" s předděličkou 1024.
for
(;;)
{
//hlavní smyčka
had<<=
1
;
//posuneme hada
if
(had<VZOR)
//pokud jsme rozsvítili méně ledek než je délka hada

had++;

//rozsvítíme poslední ledku
rozsvit(had,
DOBA);
//zavoláme funkci "rozsvit" a řekneme jí které ledky má rozsvítit
}
//konec hlavní smyčky
}
//konec funkce main      /****** /
***** */ /* *   Definice funkcí * */ /* ***** */

/****** /      /****** / /*      Rozsvit      */
/****** /

void
rozsvit(
unsigned
char
co,
unsigned
char
jakdlouho)
{
for
(
unsigned
char
i=
```

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

```
0
;
i<jakdlouho;
i++)
//jeden cyklus trvá cca 1 ms

{

for
(
unsigned
char
j=
1
;j<=
128
;j++)

{

if
(jas>=j)
//pokud je v proměnné "j" menší číslo než v proměnné "jas"

PORTB=co;
//Rozsvítíme požadované ledky

else

//jinak

PORTB
=
0
;
//zhasneme všechny ledky

_delay_us(
7
);
//počkáme 7 us

}

}
```

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

```
/* Je snahou, aby tato smyčka obsahovala co nejméně příkazů, protože vykonání každého příkazu zabere procesoru nějaký čas. Jeden cyklus tedy netrvá pouze 128 * 7 us = 896 us, nýbrž o něco déle. */
```

```
}  
/*****/ /* Čtení tlačítek */ /*****/
```

```
ISR(TIMER0_OVF_vect)
```

```
{  
TCNT0=  
157
```

```
;  
//Časovač začne počítat od 157 (255-157=98) předdělička je 1024.
```

```
//Hodinový kmitočet procesoru je 1MHz. Takže:
```

```
//1 000 000 / 1024 / 98 = 10 Hz (klávesnice se kontroluje přibližně
```

```
//10x za vteřinu, neboli jednou za cca 100 ms)
```

```
if  
(!(PIND&0b00000100))
```

```
//pokud je stisknuté tlačítko 1 (jas+)
```

```
{
```

```
if  
(jas==  
0  
)  
//pokud je jas "0"
```

```
jas=  
1  
;
```

```
else
```

```
//pokud jas není "0"
```

```
{
```

```
if  
(jas<  
128  
)  
//pokud je menší než 128 (maximální jas)
```

AVR - LED panel - #7 Program

Napsal uživatel Vašek Král
Úterý, 04 Říjen 2011 07:09

```
jas<<=
1
;
//zvýšíme jas na dvojnásobek (bitový posun doleva)

/*   Proměnná "jas" tedy může nabývat hodnot:    0, 1, 2, 4, 8, 16, 32, 64 a 128   */

}

TCNT0=
60
;
//Časovač začne počítat od 60 místo 157 (počítá do 255), takže

}

if
(! (PIND & 0b00001000))
//pokud je stisknuté tlačítko 2 (jas-)

{

jas>>=
1
;

//snížíme jas na polovinu (bitový posun doprava)

TCNT0=
60
;
//Časovač začne počítat od 60 místo 157 (počítá do 255), takže

//příští čtení tlačítka bude za cca 200 ms místo původních 100 ms.

}
}

//Pro radioklub OK1KVK napsal Vašek Král
```