Napsal uživatel Vašek Král
Sobota, 09 Duben 2011 07:33

Kompatibilní zapojení: LED panel s ATmega8
Ke stažení: LED_005.c == LED_005.pdf == LED_005.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 { } --> /* V minulém programu jsme si ukázali, jak používat proměnnou, jak vytvářet cykly s určitým počtem opakování (for) a jak používat podmínku (if). Umíme už tedy rosvěcet a zhasínat ledky v námi zvoleném pořadí. */ /* Naše dosavadní programy mají ale jeden velký nedostatek. Pokud jsme chtěli něco změnit (délku hada, rychlost), museli jsme v programu upravit potřebné konstanty, program zkompilovat a znovu nahrát do mikrokontroléru. Hodilo by se nám proto, aby bylo možné tyto konstanty měnit přímo za běhu programu bez nutnosti jej upravovat a znovu nahrávat. Potřebujeme tedy, aby program nějakým způsobem zjistil, že jsme se rozhodli některou konstantu změnit, a změnil jí sám. K tomuto účelu můžeme použít tlačítka. Jakmile program zjistí, že jsme stiskli tlačítko, změní konstantu, a pokračuje dál. Nyní tedy potřebujeme v programu nějakým způsobem zjistit, že je stisknuté tlačítko. */ /* Náš přípravek ATmega8_LED_start_2 má 3 tlačítka. Jedno je použito jako reset (resetuje mikrokontrolér) a dvě jsou volná pro všeobecné použití. Tato dvě tlačítka jsou připojena k pinům PD2 a PD3. Zkusíme pomocí jednoho tlačítka (třeba PD2) prodlužovat hada z minulého příkladu Budeme tedy potřebovat nastavit port PD2 jako VSTUP a nějakým způsobem pak zjistit, jaká je jeho hodnota. Nastavení PD2 jako vstup, by se provedlo pomocí registru DDRD (registr DDR portu D) tak, že by se na příslušné místo (2. bit) zapsala "0". Protože však po resetu mikrokontroléru jsou všechny I/O porty nastaveny na "0" - tedy vstup, nemusíme nic nastavovat. Otázkou tedy pouze je, jak z portu číst. K tomuto účelu jsou v mikrokontroléru registry PIN. Registr PIN se podobně jako registry PORT a DDR označuje písmenem portu (PINA, PINB, PINC...). Protože tlačítko je připojeno k portu "D", budeme potřebovat PIND. PIND je stejně jako ostatní registry osmibitový (obsahuje 8 jedniček a nul). Během chodu programu se do tohoto registru automaticky ukládá hodnota všech nožiček daného portu. Pokud je na nožičce napětí 0 - 0,8 V, objeví se v registru PIN na příslušném místě "0". Pokud však je k nožičce připojeno napětí mezi 2 a 5 V, je příslušný bit registru PIN nastaven na "1". Tlačítka jsou na našem "ATmega8_LED_start_2" zapojeny tak, že pokud není tlačítko stisknuto, je na příslušnou nožičku mikrokontroléru přivedeno 5 V.Pokud je tlačítko stisknuto, je na nožičce 0 V. Když tedy chceme zjistit, zda je stisknuto tlačítko PD2, musíme se podívat do registru PIND, zda je třetí bit vynulován (3 bit proto, že se nožičky i bity počítají od "0" - 0, 1,

2, (třetí pozice)). Abychom zjistili stav jednoho bitu z celého registru, musíme si jej takzvaně "vymaskovat". Vymaskovat znamená, že provedeme nějakou početní operaci, která nám odstraní ostatní (nepotřebné) bity. Zůstane nám tak osmibitové číslo, kde budou všechny bity "0" jen náš požadovaný bit zůstane nezměněn ("0" nebo "1"). Nyní můžeme zjistit, zda je tento registr větší než "0" (bit byl "1"), nebo ne (bit byl "0"). Vymaskování se dá provést pomocí logického součinu (&). Logický součin porovná dvě proměnné tak, že postupně porovnává bity které leží na stejných místech a do výsledku zapisuje "1" pouze tehdy, pokud byly jedničky na tomto místě v obou proměnných. Například: proměnná1: 10001101 (dekadicky: 141) proměnná2: 01101011 (dekadicky: 107) Výsledek: 00001001 (dekadicky: 9) Když tedy chceme zjistit jestli je nastaven třetí bit, provedeme to takto: promenna: 00110101 (dekadicky: 53) maska: 00000100 (dekadicky: 4) vysledek: 00000100 (dekadicky: 4 - je větší než "0" - bit byl "1") Zápis je následující: vysledek = promenna & maska nebo: vysledek = maska & promenna Nyní tedy umíme zjistit stav napětí na nožičkách mikrokontroléru (registr PIN) a vymaskovat si potřebnou nožičku (bitový součin - &). Můžeme se tedy pustit do psaní programu: */

```
#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
#define RYCHLOST 5
//Počet kroků hada za sekundu
#define DELKA_MIN 1
//Počáteční délka hada (počet ledek)
#define DELKA_MAX 10
//Konečná délka hada (kam až se může prodloužit)
int
main
(
void
) {
unsigned
char
delka;
//nadefinujeme si proměnnou pro délku hada (bude se měnit)
delka=DELKA_MIN;

//nastavíme délku hada na minimální hodnotu
DDRB
=
0xff
;

//všechny piny na portu "B" budou výstupní //DDRD = 0 nastavovat nemusíme - je nastaveno implicitně po resetu
```

```
for
(;;)
{
//hlavní smyčka
for
(
unsigned
char
n=
0
;n<(
8
+delka);n++)
//Nadefinováním proměnné "n" přímo v

//cyklu zajistíme, že mimo cyklus tato

//proměnná neexistuje a tudíž nezabírá

//místo v paměti.

{

if
(n<delka)
//pokud jsme rozsvítili méně ledek než je délka hada

{

PORTB
<<=
1
;
//přidáme článek hada (bity se posunou doleva a

//vpravo se doplní "0"

PORTB
++;
//přičteme "1" (nastavíme nultý bit na "1")

}

else

//pokud není počet opakování menší než délka hada
```

```
{

//to znamená, že hada už jsme nakreslili..

PORTB
<<=
1
;
//pouze posuneme bity (hada) doleva a doplní se nula...

}

_delay_ms
(
1000
/RYCHLOST);
//Čekání (1000 ms = 1 sekunda)

//V tuto chmíli proběhlo vykreslení jednoho kroku hada a proběhlo čekání.

//Než ale program necháme skočit na začátek, a nakreslit další krok hada

//zkusíme se podívat, jestli někdo zrovna teď nemačká tlačítko:

if
(PIND&0b00000100)
//pokud je 3. bit nastaven na "1"

{

//(tlačítko není stisknuté)

;

//nic nedělej a jdi dál (; = prázdný příkaz)

}

else

//pokud není 3. bit nastaven na "1" (někdo drží tlačítko)

{

delka++;
```

```
//prodloužíme hada o 1

//Teď by se ale mohlo stát, že jsme už hada prodloužili příliž.

//Musíme tedy zjistit, jestli už had není delší než DELKA_MAX

if
(delka>DELKA_MAX)

//Pokud je tedy délka větší než maximum, které

{

//jsme si na začátku programu nastavili...

delka=DELKA_MIN;
//...zkrátíme hada na minimum a můžeme zase

}

//prodlužovat.

}

}
//...a znovu na začátek smyčky
}
//hlavní smyčka...
}
//konec funkce main      /*  Tento program by měl po nahrání do mikrokontroléru vytvořit "světelného hada",   o délce definované konstantou DELKA_MIN. Had se bude pohybovat rychlostí udanou   konstantou RYCHLOST. Po stisku tlačítka se had příští kolo objeví o jeden článek   delší.  */    //Pro radioklub OK1KVK naspal Vašek Král
```