Vysoká škola chemicko-technologická v Praze
Fakulta chemicko-inženýrská
Ústav počítačové a řídicí techniky

Aplikace mikroprocesorů

Deska Evb IO

Návod k použití

Lukáš Lahoda 2010

Obsah

1 Deska Evb IO

Deska Evb IO (IO = input/output) je univerzální vstupně/výstupní deska určená k propojení s deskou EvbHCS08. Její schéma můžete prostudovat na *obr*. 1. Deska obsahuje několik konektorů pro další zařízení (BNC, J1, J2), 8 tlačítek (TL1–TL8), 8 LED diod (D1–D8) a 4 osmibitové přepínače typu switch (SW1–SW4). Pokud k desce nejsou připojena žádná další zařízení, můžeme využít diody a tlačítka.

1.1 Propojení desky Evb IO s deskou EvbHCS08

Tyto desky propojte přes paralelní port pomocí plochého kabelu, který je k dispozici v laboratoři. V tomto návodu budeme předpokládat propojení desek přes port PTA desky EvbHCS08.

1.2 Diody na Evb IO

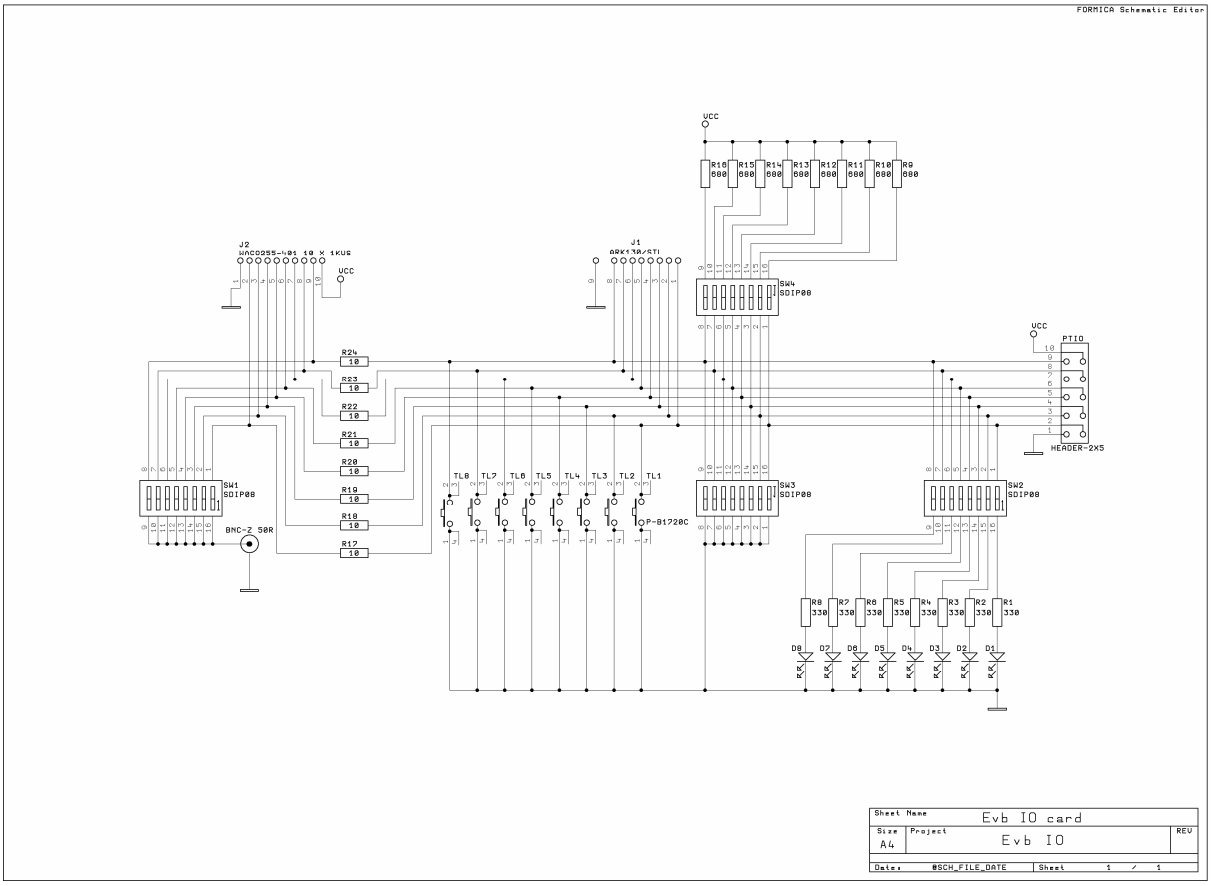
Na desce se nachází 8 LED diod, které mohou sloužit jako přídavné výstupy desky EvbHCS08 a dají se využít např. jako indikátory nejrůznějších stavů. Abychom mohli přistupovat přes paralelní port k těmto diodám, musíme správně nastavit přepínače na switch SW1 až SW4. Můžeme si vybrat, a to pro každou diodu zvlášť, jestli bude svítit pří zápisu log 0, nebo log 1 na příslušný pin portu. V tohoto návodu nastavíme switche tak, aby se všechny diody rozsvěcely zápisem log 1 na příslušné piny paralelního portu desky EvbHCS08. Nastavte tedy řady přepínačů následovně: **SW1**, **SW3** a **SW4** všechny přepínače do polohy **OFF**, **SW2** všechny přepínače do polohy **ON**. Takto nastavenou desku propojte s portem **PTA**[1] desky EvbHCS08 a v programu nastavte všechny piny tohoto portu pro zápis. Zapsáním log 1 na pin **PTA***x* by se měla rozsvítit dioda **D**(*x*+1) (diody jsou číslovány od 1 do 8, kdežto piny portů od 0 do 7).

1.3 Tlačítka na Evb IO

Máme k dispozici 8 tlačítek, které mohou sloužit jako přídavné vstupy k desce EvbHCS08. Jelikož je tato deska sama osazena pouze dvěma tlačítky, můžou být další tlačítka užitečná. Piny č. 10 všech desetipinových paralelních portů PTA až E složí jako zdroj napětí, kterým je možné napájet přídavné desky. Toto napětí nyní využijeme jako zdroj logického signálu, který budou tlačítka spínat. Sledujte následující úvahy společně se schématem na *obr. 1*. Abychom přivedli napětí, tedy náš logický signál, na tlačítka, musíme přepnout všechny přepínače switche **SW4** do polohy **ON**. Tím ale zároveň přivedeme signál log 1 na piny 2 až 9 konektoru **PTIO** (tyto piny odpovídají pinům 0–7 zvoleného paralelního portu desky EvbHCS08), což za předpokladu, že jsme naše desky propojili přes port **PTA**, znamená, že při uvolněném tlačítku **TL**(*x*+1) získáme čtením pinu **PTA***x* hodnotu log 1 (tlačítka jsou číslována od 1 do 8, kdežto piny portů od 0 do 7). Přepínače na **SW3** se nyní musí všechny nacházet v poloze **OFF**, jinak by nám napětí touto cestou unikalo na zem. Pokud přepneme přepínače na **SW2** na **ON**, můžeme sledovat stav tlačítek pomocí LED diod (dioda svítí = příslušné tlačítko uvolněno a naopak). **SW1** zůstává v poloze **OFF**. Stiskem některého

[1] POZOR! Na deskách EvbHCS08 přítomných v laboratoři AL02 je prohozeno označení portů PTA a PTB!

z tlačítek propojíme zdroj napětí se zemí a tím napětí na příslušném pinu portu, ze kterého čteme, klesne na 0. Čteme-li hodnotu pinu **PTA***x* a má-li tento hodnotu 0, znamená to, že bylo stisknuto tlačítko **TL**(*x*+1).



Obr. 1. Schéma desky Evb IO

2 Vzorové programy

Vytvoříme nyní program, který bude v určitých časových intervalech s využitím časovače TPM1 na EvbHCS08 zapisovat data na port PTA a tím rozsvěcet/zhasínat diody na desce Evb IO. Propojte tedy obě desky pře port PTA, přepněte přepínače switch dle odst. 1.2, nastavte všechny piny portu PTA pro zápis a inicializujte časovač TPM1.

```
void main(void) {

  SOPT_COPE=0;

  TPM1SC=0b01001010;

  PTAD= 0b00000000;
  PTADD=0b11111111;

  EnableInterrupts;

  for(;;) {

  }

}
```

Nyní budete mít za úkol vytvořit několik různých implementací funkce pro přerušení časovače, každá z nich bude představovat jeden program pro LED diody. Na závěr vytvoříme jeden program pro otestování funkčnosti tlačítek a ověření možnosti jejich použití jako přídavných vstupů.

2.1 Program č.1 – Binární čísla

Zadání

Úkolem je ve zvoleném časovém intervalu postupně zobrazovat čísla 0 až 255 v binárním kódu pomocí 8 diod na desce Evb IO.

Řešení

Sestavíme funkci pro přerušení časovače TMP1. Na začátku programu máme na portu PTA zapsanou hodnotu 0b00000000, poté při každém přerušení zvýšíme tuto hodnotu o 1. Funkce obsluhující přerušení vypadá následovně:

```
interrupt 8 void tpm1ovf (void) {

  TPM1SC_TOF=0; //nulování příznaku přerušení

  asm {
    lda PTAD
    inca
    sta PTAD
  }

}
```

2.2 Program č.2 – Had

Zadání

Naprogramujte funkci pro přerušení časovače tak, aby výsledkem byl „had“ o délce 3 diod, který se ze své počáteční pozice posune při každém přerušení o 1 pozici zvoleným směrem. Pokud had přeteče, objeví se znovu na druhém konci. Výsledkem by měla být následující sekvence (symbol „x“ představuje rozsvícenou diodu, symbol „o“ zhasnutou):

ooooooxxx → oooooxxxo → oooxxxoo → ooxxxooo → oxxxoooo → xxxoooooo → xxooooox → xoooooxx → ooooooxxx → atd.

Řešení

Počáteční stav hada vytvoříme zápisem hodnoty 0b00000111 na port PTA ve funkci main.

```
PTAD= 0b00000111;
```

Posun hada řešíme v obslužné funkci přerušení rotací registru PTAD. Musíme si však dát pozor v okamžiku, kdy dochází k rotaci jedničky přes carry bit z bitu 7 na bit 0. Instrukce **`rol`** standardně funguje tak, že v jedné iteraci provede rotaci a hodnotu bitu 7 uloží do carry bitu. Tuto hodnotu pak až v další iteraci uloží do bitu 0. Jak ale znáte z teorie přerušení, dochází před voláním obslužné funkce k uložení hodnot registrů CPU do zásobníku, odkud jsou po návratu z obslužné funkce obnoveny. Tím dojde ke ztrátě informace v carry bitu, kam jsme v obslužné funkci dočasně uložili hodnotu bitu 7 a tato hodnota při dalším přerušení již není k dispozici. Had by nám tedy odjel za okraj a na druhé straně by se už neobjevil. Před návratem z obslužné funkce přerušení tedy musíme po rotaci zkontrolovat hodnotu v carry bitu a v případě, že je nastavena na 1, manuálně nastavit bit 0 na 1. Had se nám poté po přetečení vrací na začátek a rotuje v nekonečné smyčce.

```
interrupt 8 void tpm1ovf (void) {

      asm {

        rol PTAD //rotace
        bcc next //branch if carry bit clear

        lda PTAD
        ora #0b00000001
        sta PTAD

        next:
      }

}
```

2.3 Program č.3 – Cihly

Zadání

Pomocí LED diod na desce Evb IO vizualizujte padání cihel na hromadu. Začněte se všemi diodami zhasnutými a poté postupně vytvářejte „cihly“ rozsvěcením bitu 0 a následným posunem směrem k bitu 7. První cihla se zastaví na bitu 7, druhá na bitu 6 atd. až dojde k zaplnění všech osmi bitů jedničkami. Poté opět všechny bity nastavte na 0 a cyklus opakujte v nekonečné smyčce. Cílem je realizovat následující sekvenci:

ooooooo → ooooooox → ooooooxo → oooooxoo → ooooxooo → oooxoooo → ooxooooo → oxooooo → xooooooo → xooooox → xoooooxo → xooooxoo → xoooxooo → xooxoooo → xoxooooo → xxooooo → … → xxxooooo → … → xxxxoooo → … → xxxxxooo → … → xxxxxxoo → … … → xxxxxxxx → oooooooo → atd.

Řešení

Řešení tohoto úkolu je ze všech nejsložitější. Kromě registru PTAD budeme muset využít 3 další bajty v paměti, a to na adresách 0x80, 0x81 a 0x82. Aktuální stav hromady budeme udržovat na adrese 0x80. Cihla bude padat na adrese 0x81, kde vždy nastavíme bit 0 na 1 a poté pomocí instrukce **`lsl`** (logical shift left) posuneme v každé iteraci cihlu o 1 pozici dolů. Pro zobrazení situace sečteme logickým součtem (**`ora`**) hromadu (0x80) s cihlou (0x81) a výsledek zapíšeme do registru PTAD, čímž dojde k rozsvícení příslušných diod. Dál je nutné kontrolovat, zda cihla již dopadla na hromadu a zareagovat zvětšením hromady o 1 cihlu, zrušením stávající cihly a vytvořením cihly nové. K tomu nám poslouží bajt na adrese 0x82, kam si vždy na začátku iterace uložíme aktuální hromadu zvětšenou o 1 cihlu a tu pak porovnáme s aktuální situací. Pokud je aktuální situace rovna se situací na adrese 0x82, došlo k dopadu cihly na hromadu a můžeme na to zareagovat. Poslední věcí, kterou je třeba v každé iteraci zkontrolovat, je zda nedošlo k zaplnění všech polí cihlami a pokud ano, hromadu zrušit a začít ji plnit od začátku.

Ve funkce main inicializujte hromadu a cihlu:

```
asm {
    mov #0b00000000,0x80 //hromada cihel
    mov #0b00000001,0x81 //padající cihla
}
```

Obslužnou funkci přerušení časovače sestavte následovně:

```
interrupt 8 void tpm1ovf (void) {

     asm {

        lda 0x80
        lsra
        ora #0b10000000
        sta 0x82 //hromada zvětšená o cihlu

        lda 0x80
        ora 0x81 //hromada+padající cihla

        sta PTAD //zobraz situaci
```

```
        cmp 0x82 //porovnej aktuální situaci s hromadou zvětšenou o 1
        beq dopad //pokud cihla dopadla na hromadu, jdi na dopad

        lsl 0x81 //pád cihly = posunutí cihly o 1 pozici doleva
        bra end

        dopad:
        mov 0x82,0x80 //ulož zvětšenou hromadu
        mov #0x00000001,0x81 //vytvoř novou cihlu na startovní pozici

        lda 0x80
        cmp #0b11111111
        blt end //hromada není plná?

        mov #0b00000000,0x80 //vyprázdni hromadu, pokud je plná

        end:
      }

}
```

2.4 Program č. 4 – Tlačítka

V následujícím velice krátkém příkladě ověříme funkčnost tlačítek. Nastavte desku dle odst. 1.3., propojte desky přes port PTA a vytvořte následující krátký program:

```
void main(void) {

  SOPT_COPE=0; //zákaz watchdog

  PTADD=0b00000000; //nastavení PTA pro čtení

  for(;;) {

    if (PTAD_PTAD0==0) {
      asm {
        lda 0b00001111;
      }
    }
    else {
      asm {
        lda 0b11110000;
      }
    }

  }
```

Vytvořený program krokujte a sledujte hodnotu v registru A. Tato hodnota by se měla měnit v závislosti na stavu tlačítka TL1. Při stisku jakéhokoli tlačítka by zároveň mělo dojít ke zhasnutí jemu odpovídající LED diody.

3 Literatura

[1] Freescale Semiconductor MC9S08GB/GT Data Sheet