SNÍMAČ POLOHY HVĚZDÁŘSKÉHO DALEKOHLEDU

Cyril Koníček

4. ročník SŠPH Uh. Hradiště

ABSTRAKT

Účelem práce je navrhnout přístroj, který bude snímat polohu hvězdářského dalekohledu – jeho horizontální a vertikální natočení. Z těchto získaných údajů vypočítá, na které místo na obloze dalekohled směřuje. Výsledná pozice bude zobrazena na LCD displeji jako výška nad obzorem a azimut.

SEZNAM POUŽITÝCH SYMBOLŮ

DPS – deska plošného spoje

PC – počítač

MCU – mikroprocesor

DPI – údaj určující, kolik obrazových bodů (pixelů) se vejde do délky jednoho palce

LCD displej – zobrazovací zařízení

master – zařízení ktere ma vyžší prioritu než slave – v našem případě mikroprocesor

slave – zařízení ktere ma nižší prioritu než master – v našem případě myš

JSI – jazyk symbolických instrukcí

1 ÚVOD

Práce se zabývá návrhem snímače polohy hvězdářského dalekohledu. Pro snímání polohy dalekohledu jsou použity optické myši a pro zobrazování je použit červený LCD diplej. To všechno řídí jednočipový mikroprocesor PIC 16F84A.

2 ROZBOR PROBLÉMU

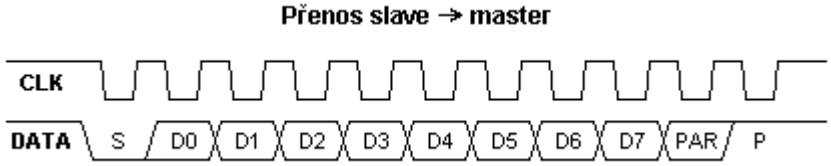
Nejprve jsem si musel určit přípravek, který je v hodný pro snímání polohy. Vybral jsem optickou myš, která je se svým rozlišením až 800DPI více než dostačující. Ovšem problém je v tom, že optická myš vysílá synchronní seriový signál – PS/2 protokol.

PS/2 protokol

Jde o synchronní obousměrný protokol master-slave, který umožňuje připojit jen jediné slave zařízení (v našem případě myš). Důležitou vlastností je, že hodiny generuje vždy slave. Master (PC nebo v našem případě MCU) je ovšem nadřazen - pomocí clk signálu povoluje, zakazuje nebo přerušuje přenos dat. Hodinový kmitočet se pohybuje v rozmezí cca 10-16 kHz.

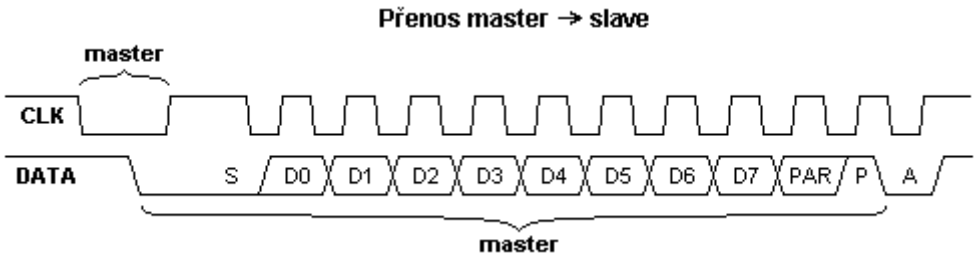
Přenos "slave >> master"

Slave může samovolně zahájit přenos jednoho bytu pokud jsou obě linky (clk i data) v úrovni high po dobu nejméně 50µs. Datový rámec obsahuje celkem 11 bitů. Přenos je zahájen start bitem s úrovní low, následuje 8 bitů dat řazených od nejméně významného, dále je zde paritní bit (lichá parita - high pro sudý počet jedniček v datech) a konečně stop bit s úrovní high. Mimo to, že hodiny generuje slave je zde ještě jedna nepříjemnost - master by měl číst stav datové linky nikoliv se sestupnou hranou hodin, ale přesně uprostřed low úrovně hodin. To odpovídá přibližně 15-25µs po sestupné hraně hodin.



Přenos "master >> slave"

Přenos bytu do zařízení se od opačného směru trochu liší. Předně master musí vyslat požadavek na přenos dat (request-to-send). To se provede následovně: Nejprve master stáhne hodiny do low úrovně na nejméně 100µs, čímž bezpečně zablokuje případný přenos opačným směrem. Následně master stáhne do low i data (což je vlastně start bit) a počká dalších alespoň 5µs, pak uvolní hodiny, načež by měl slave zareagovat nejhůře do 10ms tak, že začne generovat hodiny. Polarita hodin je při přenosu do zařízení přesně opačná - slave vzorkuje data přesně uprostřed high úrovně hodin, master mění hodnotu dat během low úrovně hodin (nejlépe přesně uprostřed).



Pro zpracování tohoto signálu jsem zvolil jednočipový mikroprocesor PIC 16F84A, který má dostatek vývodů, abych k němu připojil i LCD displej. LCD displej je pochopitelně červený, protože přípravek je určen pro provoz v noci a červená barva nezanechává „světelnou stopu“ v našem oku. Stěžejním úkolem byl napsat program, který bude zpracovávat data přijatá z optických myší. Tato data pak přepočítá na stupně a výsledek zobrazí na LCD disleji.

Návrh programu

Návrh programu jsem rozdělil na tři nejdůležitější části:

a) Vytvoření synchroní sériové komunikace mikroprocesoru a optické myši.
b) Přepočítání získaných informací na stupně.
c) Zobrazení výsledných dat na LCD displeji.

a) Vytvoření synchroní sériové komunikace mikroprocesoru a optické myši

Zde jsem musel vymyslet podprogramy pro příjem a vysílání bitů do optické myši. Myš vysílá sériovým přenosem 3x11 bitů. V prvních jedenácti bitech jsou pro nás nepodstatné informace (zmáčknuté tlačítka, pohyb kolečka....). V dalších jedenácti bitech se vysílá posun v ose X od posledního poslání dat. Posledních jedenáct bitů vypadá podobně jako předchozí, jen nesou informaci o posunu v ose Y. Tato informace o posunu je udávána v plus nebo mínus 127. Záporné číslo je posíláno jako 1.doplněk binárního čísla.

Myš má několik módů posílání dat, které můžeme libovolně nastavovat (od posílání informace každou setinu sekundy, přes posílání dat jen při změně polohy myši, až po vysílání jen když pošleme požadavek na vyslání dat). Pro mě je nejlepší mód, který posílá informace jen tehdy když si o ně požádám. Po nastavení módu požádám 1. myš, aby poslala data a poté požádám o to samé i 2. myš. Přijatá data si uložím do registrů.

b) Přepočítání získaných informací na potřebné stupně

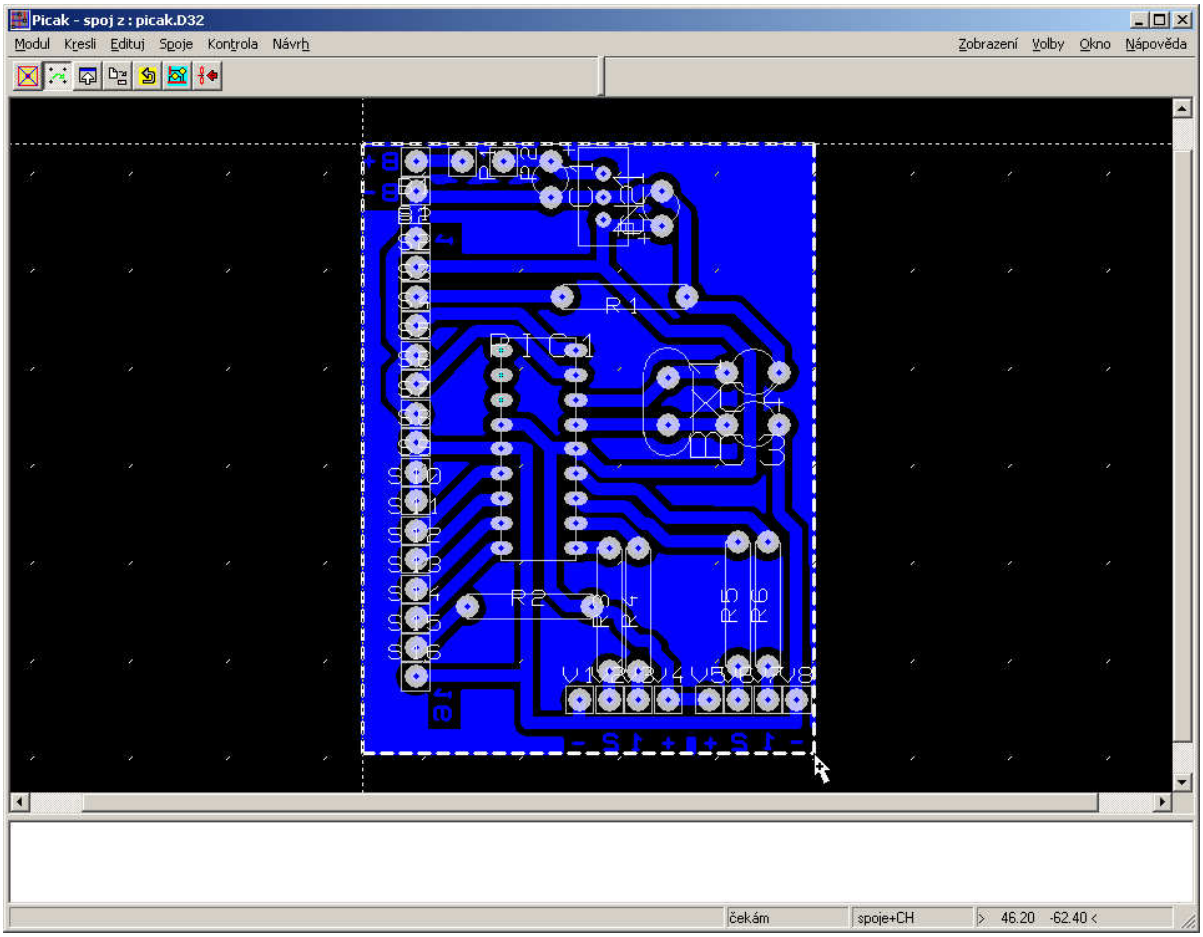
Získané informace z myší nyní připočteme nebo odečteme od minulé polohy dalekohledu a výsledek přepočítáme na stupně.

c) Zobrazení výsledných dat na LCD displeji

Tady byla potřeba nejprve inicializovat LCD dispej a poté posílat na displej data pro zobrazování správných písmen a číslic na displeji.

3 NÁVRH DESKY PLOŠNÉHO SPOJE

Pro návrh DPS jsem použil návrhový systém LSD2000. Deska je navržena jako jednostranná a to z důvodu technologie výroby. Velikost desky je určena velikosti použité krabičky. Rozmístění součástek je provedeno na základě vodivostních spojení mezi součástkami. Při návrhu bylo dbáno na to, aby se vodiče nekřížily a tudíž nebyly potřeba žádné vodivé propojky.



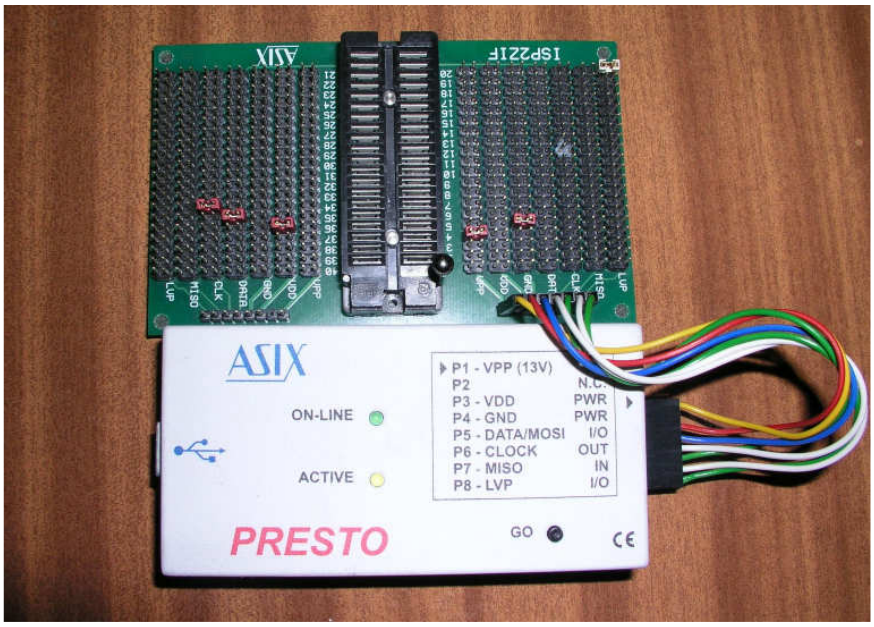
Obr. 1 Návrhový systém LSD2000

Vyrobená deska plošného spoje byla osazena součástkami a zabudována do plastové krabičky. Snímač polohy je napájen 9V akumulátorem.

4 MĚŘENÍ A OŽIVOVÁNÍ ZAŘÍZENÍ

Pro oživení přístroje bylo nutné do mikpropocesoru 16F84A nahrát program, který bude řídit činnost celého zařízení. To jsem udělal pomocí programátoru ASIX PRESTO s paticí ISP2ZIF.

Pro převod ze souboru .asm do souboru .hex jsem použil program firmy Mikrochip MPASM v5.0, pro naprogramování mikroprocesoru program UP! ver. 2.28 firmy ASIX.



Obr. 3 Programátor ASIX PRESTO s patici ISP2ZIF

5 ZÁVĚR

Zařízení funguje správně, přesnost dat získaných z myší je závislá na upevnění a kontaktu myší s „podložkou“, ale pro demonstrační účely je dostačující. Přesnost můžeme zvýšit nastavením rozlišení myší.

OObr.4 Snímač polohy zabudovaný u hvězdářského dalekohledu

SEZNAM POUŽITÉ LITERATURY A ZDROJE

[1] Manuál LSD2000 verze 5, dostupné z: http//www.lsd2000.cz

[2] Katalog GM: Součástky pro elektrotechniku, Praha 2000.

[3] http://elektronika.kvalitne.cz

[4] http://cs.wikipedia.org

[5] Jiří Hrbáček: Komunikace mikrokontroléru s okolím

SEZNAM PŘÍLOH

[1] Seznam součástek

[2] Schéma

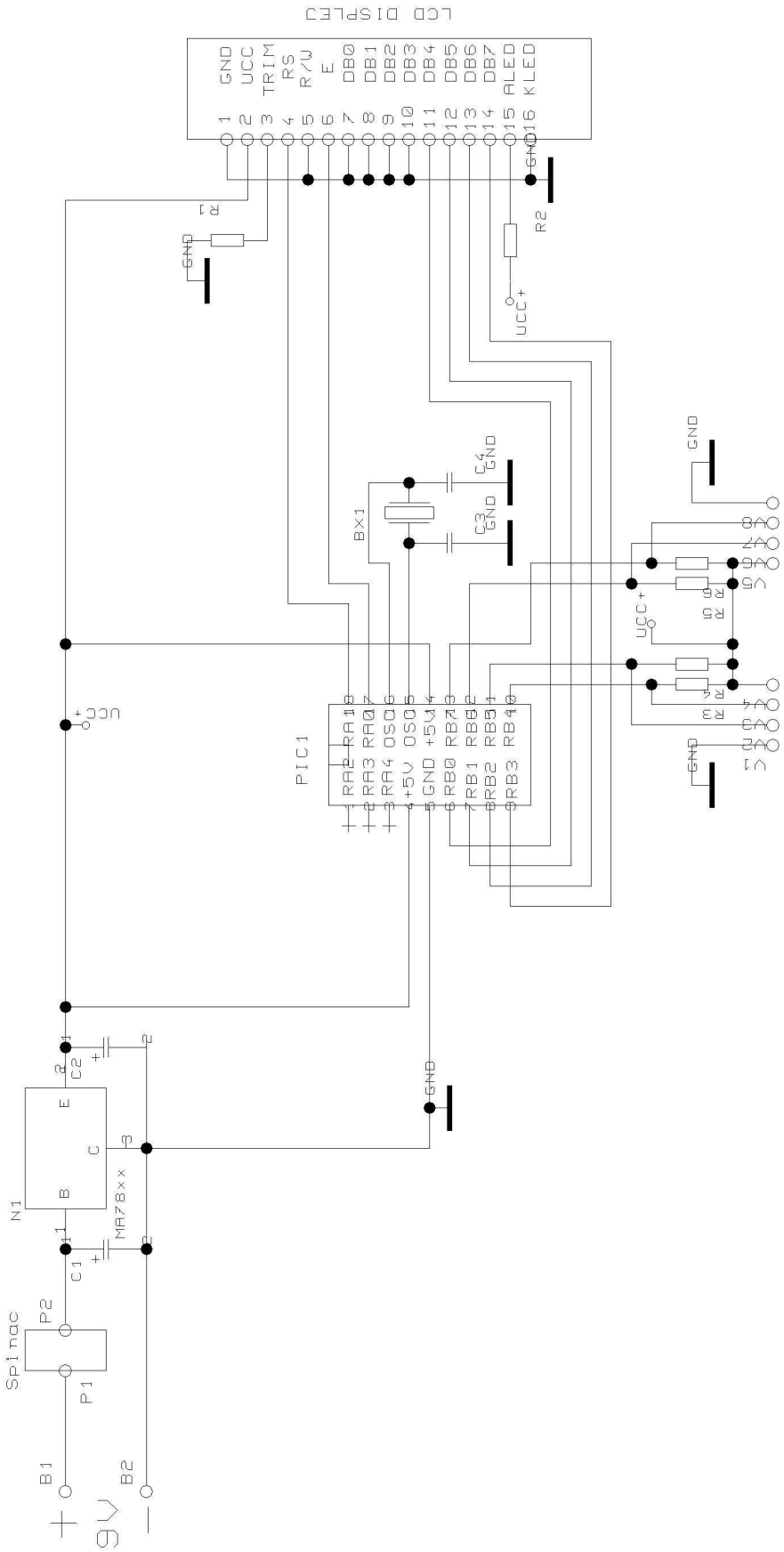
[3] Klišé

[4] Rozmístění součástek

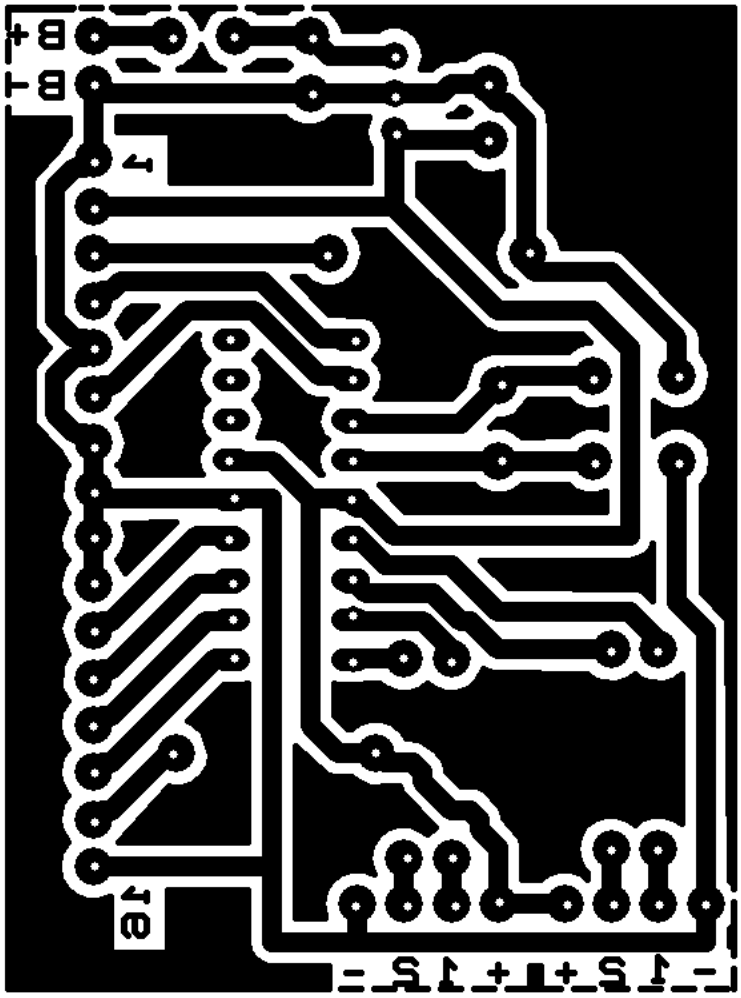
[5] Program pro mikroprocesor napsaný v JSI

	Název součástky	**Hodnota součástky**
7805	stabilizátor	
PIC16F84A	mikroprocesor	
BX1	krystal	4 Mhz
C1	kondenzátor	103 μF
C2	kondenzátor	324 μF
C3,C4	kondenzátor	33 μF
R1	rezistor	100 Ω
R2	rezistor	1000 Ω
R3,R4,R5,R6	rezistor	1800 Ω
LCD	LCD displej	

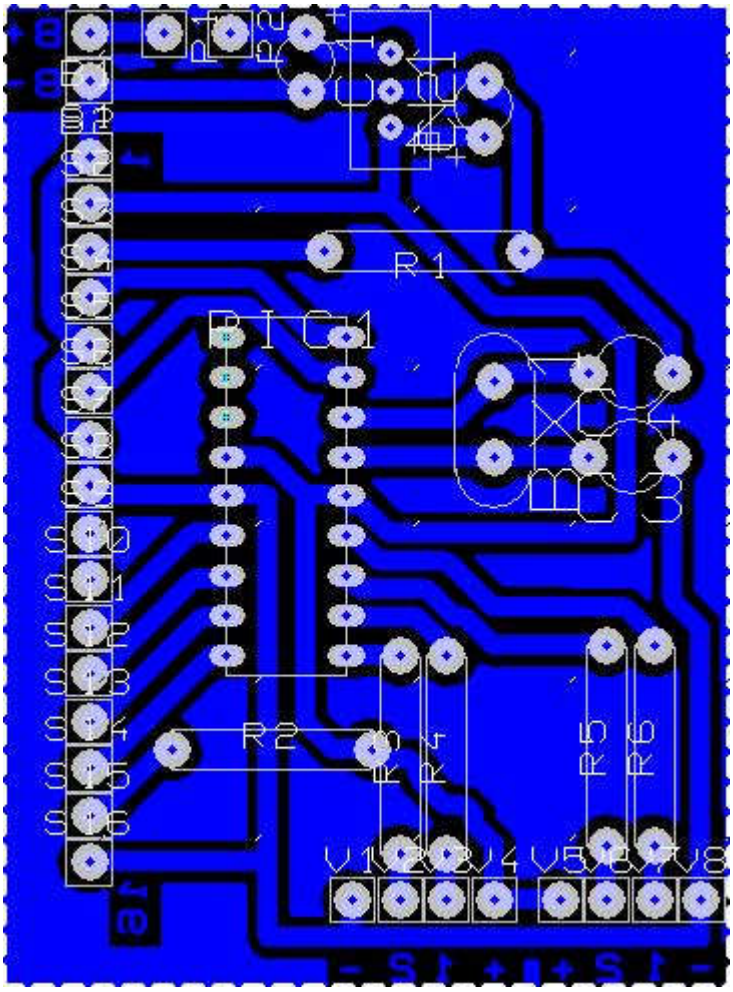
	Vypracoval: Cyril Koníček Dne:	Název: Seznam součástek	
	Schválil: Dne:	Číslo výkresu: 1/4	List:1



	Vypracoval: Cyril Koníček Dne:	Název: Schéma	
	Schválil: Dne:	Číslo výkresu: 2/4	List:1



	Vypracoval: Cyril Koníček Dne:	Název: Klišé	
	Schválil: Dne:	Číslo výkresu: 3/4	List:1



	Vypracoval: Cyril Koníček Dne:	Název: Rozmístění součástek	
	Schválil: Dne:	Číslo výkresu: 4/4	List:1

Program pro mikroprocesor napsaný v JSI

```
;*********************************************************************
;*                                PIC16F84                           *
;*                             --------\/--------                    *
;*                           -|RA2          RA1|-  LCD E             *
;*                           -|RA3          RA0|-  LCD RS            *
;*                           -|RA4         osc1|-  \ Krystal 4,00MHz *
;*                     +5V   -|MCLR        osc2|-  /                 *
;*                      0V   -|Vss          Vdd|-  +5V               *
;*          LCD data bit 4   -|RB0/INT  PGD/RB7|-  Data              *
;*          LCD data bit 5   -|RB1      PGC/RB6|-  CLK               *
;*          LCD data bit 6   -|RB2          RB5|-  Data2             *
;*          LCD data bit 7   -|RB3          RB4|-  CLK2              *
;*                             ------------------                    *
;*                                                                   *
;*********************************************************************

      include     "p16f84A.inc"
      list  p=16f84A, R=DEC

RAM         equ   0Ch   ; první adresa paměti RAM pro PIC16F84 -68bajtů
TMP_W       equ   RAM+1 ; temp preruseni
TMP_S       equ   RAM+2 ; temp preruseni
TMP_PCL     equ   RAM+3 ; temp preruseni
TMP0        equ   RAM+5 ; cekaci smycka
TMP1        equ   RAM+6 ; cekaci smycka
TMP2        equ   RAM+7 ; cekaci smycka
TMP3        equ   RAM+8 ; cekaci smycka
TEXT        equ   RAM+10      ; zobrazeni textu na LCD
ZNAK        equ   RAM+11      ; zobrazeni znaku na LCD
CISLO1      equ   RAM+12      ; pomocne cislo
CISLO2      equ   RAM+13      ; pomocne cislo
pom         equ   RAM+14      ; pomocne cislo
zn          equ   RAM+15      ; pomocne cislo
dv          equ   RAM+16      ; pomocne cislo
P1          equ   RAM+17      ; pomocne cislo
P2          equ   RAM+18      ; pomocne cislo
P3          equ   RAM+19      ; pomocne cislo
P4          equ   RAM+20      ; pomocne cislo
ACCaLO      equ   RAM+21
ACCaHI      equ   RAM+22
ACCbLO      equ   RAM+23
ACCbHI      equ   RAM+24
ACCcLO      equ   RAM+25
ACCcHI      equ   RAM+26
ACCdLO      equ   RAM+27
ACCdHI      equ   RAM+28
DX          equ   RAM+29
DY          equ   RAM+30
znX         equ   RAM+31
znY         equ   RAM+32
Zcislo      equ   RAM+33
Azim_LO     equ   RAM+34
Azim_HI     equ   RAM+35
Vysk_LO     equ   RAM+36
Vysk_HI     equ   RAM+37
```

```
#define     E     PORTA,0          ;
#define     RS    PORTA,1          ;
#define     clk   PORTB,6          ;
#define     data  PORTB,7          ;
#define     clk2  PORTB,4          ;
#define     data2 PORTB,5          ;

;     org   0x2007            ; adresa konfigurace PIC16F84
      __config _XT_OSC & _PWRTE_OFF & _WDT_OFF & _CP_OFF


      org   0           ; zacatek programu
      goto  INIT        ; skok na počáteční inicializaci
      ORG   4
      RETFIE


;*************************************************************
INIT  movlw b'00000'
      movwf PORTA       ; přednastavení PORTu A

      movlw b'00000000'
      movwf PORTB       ; přednastavení PORTu B
;       ------------------------


;     banksel     TRISA
      bsf   STATUS,RP0  ; nastavení BANKY 1

      movlw b'00000'    ; portA 4-0 vystupy
      movwf TRISA

      movlw b'11010100' ; PULL-UPy ON, 1:256
      movwf OPTION_REG

      movlw b'11110000' ; portB 7-0, 4vstupy, 4vystupy
      movwf TRISB

      bcf   STATUS,RP0  ; nastavení BANKY 0
;     banksel PORTA

;       ------------------------
      call  INI_LCD           ; inicializace LCD
;******************** Hlavni program ******************************

      call  C_LCD
      call  nulAB

      call  CEK100m
      call  CEK100m
      call  CEK100m

      movlw b'00000111' ;
      movwf CISLO1
      movlw b'11111111' ; posli FF ?
      movwf CISLO2
      call  posliB
      call  CEK100m
```

```
      movlw b'00000111' ;
      movwf CISLO1
      movlw b'11110000' ; posli F0
      movwf CISLO2
      call  posliB
      call  CEK100m

a1    movlw b'00000111' ;
      movwf CISLO1
      movlw b'11110000' ; posli F0
      movwf CISLO2
      call  poslB2
      call  CEK100m

zac0                    ; prvni mys
      movlw b'00000111' ;
      movwf CISLO1
      movlw b'11101011' ; posli EB
      movwf CISLO2
      call  posliB
      call  CEK100


      call    ctibyte         ; FA
      call    ctibyte
      call    ctibyte
      call    ctibyte

      movlw 1
      movwf znX
      BTFSS CISLO2,7
      clrf  znX

      movlw 0xFF
      btfsc znX,0
      xorwf CISLO2,1

      movf  CISLO2,0
      movwf   ACCaLO
      clrf  ACCaHI

      call  Az2B

      btfsc znX,0
      goto  bla1

      call    D_add
      goto  bla2

bla1  call  D_sub
bla2
      call  B2Az

m2                      ; druha mys
      movlw b'00000111' ;
      movwf CISLO1
      movlw b'11101011' ; posli EB ?
      movwf CISLO2
      call  poslB2
      call  CEK100
```

```
      call    ctibyt2         ; FA
      call    ctibyt2
      call    ctibyt2
      call    ctibyt2

      movlw 1
      movwf znX
      BTFSS CISLO2,7
      clrf  znX

      movlw 0xFF
      btfsc znX,0
      xorwf CISLO2,1

      movf  CISLO2,0
      movwf   ACCaLO
      clrf  ACCaHI

      call  Vy2B

      btfsc znX,0
      goto  bla10

      call    D_sub
      goto  bla20

bla10 call  D_add ; mys je opacne, tak je zameneno sub a add
bla20
      call  B2Vy


aaa   call  C_LCD

      movlw 0x41
      call  WR_DATA     ; A
      movlw 0x7A
      call  WR_DATA     ; z

      movlw 0x2D        ; minus
      btfss Azim_HI,7
      movlw 0x20        ; mezera
      call  WR_DATA

      call  Z_Azim
      call  zobr2

      movlw 0xDF
      call  WR_DATA   ; zobrazi stupne


      call  LINE2


      movlw 0x56
      call  WR_DATA     ; V
      movlw 0x79
      call  WR_DATA     ; y

      movlw 0x2D        ; minus
```

```
      btfss Vysk_HI,7
      movlw 0x20        ; mezera
      call  WR_DATA

      call  Z_Vysk
      call  zobr2

      movlw 0xDF
      call  WR_DATA   ; zobrazi stupne

      goto  zac0
      ; tady konci program a vraci se na zacatek



; podprogramy
;*****************************************************************************

poslB2      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'00000000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      bcf   clk2              ; clk na 0
      call  CEK100                  ; pockej
      call  CEK100                  ; pockej
      call  CEK40             ; pockej

      bcf   data2             ; data na 0
      call  CEK19       ; pockej 19 mikrosec
      bsf   clk2              ; clk na 1

      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'00010000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      movlw 10          ; do P4 dej 10 (pocet poslanych bitu)
      movwf P4

jed2  btfsc clk2        ; cekej na vzestupnou hranu clk
      goto  jed2


nn2   btfss clk2        ; cekej na sestupnou hranu clk
      goto  nn2
      bsf   data2       ;nastav data podle nejnizsiho bitu v CISLO2
      btfss CISLO2,0
      bcf   data2
      rrf   CISLO1,1
      rrf   CISLO2,1
      decfsz      P4,1        ; sniz pocet vyslanych bitu
      goto  jed2

      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'11110000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      return
```

```
;*********************************************************************
posliB      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'00000000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      bcf   clk               ; clk na 0
      call  CEK100                 ; pockej
      call  CEK100                 ; pockej
      call  CEK40             ; pockej

      bcf   data              ; data na 0
      call  CEK19       ; pockej 19 mikrosec

      bsf   clk               ; clk na 1

      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'01000000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      movlw 10          ; do P4 dej 10 (pocet poslanych bitu)
      movwf P4

jed   btfsc clk         ; cekej na vzestupnou hranu clk
      goto  jed

nn    btfss clk         ; cekej na sestupnou hranu clk
      goto  nn

      bsf   data        ;nastav data podle nejnizsiho bitu v CISLO2
      btfss CISLO2,0
      bcf   data

      rrf   CISLO1,1
      rrf   CISLO2,1

      decfsz      P4,1        ; sniz pocet vyslanych bitu
      goto  jed

      bsf   STATUS,RP0        ;přesun do banky 1
      movlw b'11110000'       ;binární hodonta do W
      movwf TRISB             ;nastavení portu
      bcf   STATUS,RP0        ;přesun zpět do banky 0

      return
```

```
;*******************************************************************************
ctibyte     clrf  CISLO1            ; vynuluj cislo1
      clrf  CISLO2            ; vynuluj cislo2

      movlw 11          ; do P4 dej 11 (pocet ctenych bitu)
      movwf P4

j     btfss clk         ; cekej na sestupnou hranu clk
      goto  j
n     btfsc clk
      goto  n

;     call  CEK19       ; pockej 19 mikrosec

      movlw 255
      movwf P1

      btfss data
      clrf  P1

      rrf   P1,1        ; odrotuj P1, aby bit 7 byl v C

      rrf   CISLO1,1    ; odrotuj cislo1 a cislo2
      rrf   CISLO2,1


      decfsz      P4,1        ; sniz pocet ctenych bitu
      goto  j

      movlw 6
      movwf P1

rop   rrf   CISLO1,1
      rrf   CISLO2,1
      decfsz      P1,1
      goto  rop

      return

;*******************************************************************************

ctibyt2     clrf  CISLO1            ; vynuluj cislo1
      clrf  CISLO2            ; vynuluj cislo2

      movlw 11          ; do P4 dej 11 (pocet ctenych bitu)
      movwf P4

j2    btfss clk2        ; cekej na sestupnou hranu clk
      goto  j2
n2    btfsc clk2
      goto  n2

      movlw 255
      movwf P1

      btfss data2
      clrf  P1

      rrf   P1,1        ; odrotuj P1, aby bit 7 byl v C
      rrf   CISLO1,1    ; odrotuj cislo1 a cislo2
```

```
      rrf   CISLO2,1

      decfsz      P4,1        ; sniz pocet ctenych bitu
      goto  j2

      movlw 6
      movwf P1

rop2  rrf   CISLO1,1
      rrf   CISLO2,1
      decfsz      P1,1
      goto  rop2

      return

;*********************************************************************

; podprogram zobr2  - zobrazi promennou Zcislo na aktualni pozici displeje
;                 pouziva promenne P1, P2, P3

zobr2 ;call C_LCD
      clrf  P1
      clrf  P2
      clrf  P3

p100  movlw 100
      subwf Zcislo,1
      btfss STATUS,C
      goto  p10

      incf  P1,1
      goto  p100

p10   movlw 100
      addwf Zcislo,1
p11   movlw 10
      subwf Zcislo,1
      btfss STATUS,C
      goto  p_1

      incf  P2,1
      goto  p11

p_1   movlw 10
      addwf Zcislo,1
      movf  Zcislo,0
      movwf P3

      movf  P1,0
      addlw 30h
      call  WR_DATA
      movf  P2,0
      addlw 30h
      call  WR_DATA
      movf  P3,0
      addlw 30h
      call  WR_DATA

      return
```

```
;************************************************************
LINE1 movlw 0x80        ; 1 radek, 0 znak displeje
      goto  WR_CMD
;------------------------------
LINE2 movlw 0xC0        ; 2 radek, 0 znak displeje
      goto  WR_CMD
;------------------------------
C_LCD movlw 0x01        ; smaz LCD a vrat se na pozici 0
      goto  WR_CMD
;------------------------------
WR_CMD      bcf   RS          ; RS=0, zápis instrukcí do LCD
      goto  WR_LCD
;------------------------------
WR_DATA     bsf   RS          ; RS=1, zápis dat do LCD
      goto  WR_LCD
;------------------------------
WR_LCD      movwf ZNAK        ; přepsat W do ZNAK    !!! data v registru W
!!!

      bsf   E           ; nastav Enable
      movf  PORTB,W           ; stav portu do W
      iorlw 0x0F        ; zamaskuje dolni bity portu
      movwf TMP1
;------------------------------
      swapf ZNAK,W
      iorlw 0xF0        ; zamaskuje horní bity znaku

      andwf TMP1,W            ; pošle vyšší 4 bity
      movwf PORTB       ; !!! data zapsána na PORTB 0-3 !!!
      bcf   E           ; zapiše do LCD
;------------------------------
      bsf   E           ; nastav Enable
      movf  ZNAK,W
      iorlw 0xF0        ; zamaskuje horní bity znaku

      andwf TMP1,W            ; pošle nižší 4 bity
      movwf PORTB       ; !!! data zapsána na PORTB 0-3 !!!
      bcf   E           ; zapiše do LCD
;------------------------------
      btfsc RS
      goto  CEK40       ; RS=1, zápis dat - čekej 40 us
;       ------------------------
      movlw 0x04        ; instrukce 1, 2 a 3 - čekej 1,64 ms
      subwf ZNAK,W
      btfss STATUS,C
      goto  CEK1m6            ; C=0, instrukce CLEAR - čekej 1,64 ms
      goto  CEK40       ; C=1, zápis dat - čekej 40 us
      return
```

```
;************************************************************
INI_LCD     call  CEK15m            ; 4-bitová inicializace displeje LCD
;     call  CEK15m
      bcf   RS          ; RS=0, zápis instrukcí do LCD
;       ------------------------
      movlw 03h
      movwf PORTB
      bsf   E
      bcf   E           ; zapiše do LCD
;     call  CEK4m       ; čekej 4 ms
      call  CEK15m            ; !! pro zjednoduseni dano 15ms !!
;       ------------------------
      bsf   E
      bcf   E           ; zapiše do LCD
      call  CEK100            ; čekej 100 us
;       ------------------------
      bsf   E
      bcf   E           ; zapiše do LCD
      call  CEK1m6            ; čekej 1,6 ms
;-------------------------------
      movlw 02h
      movwf PORTB
      bsf   E
      bcf   E           ; zapiše do LCD
      call  CEK1m6            ; čekej 1,6 ms
;-------------------------------
      movlw 28h         ; 00101000 - počet bitů, 2 řádky, 5x7 znak
      call  WR_CMD
      movlw 0Ch         ; 00001100 - display ON, kurzor OFF, blikání OFF
      call  WR_CMD
      movlw 01h         ; 00000001 - smaže displej
      call  WR_CMD
      movlw 06h         ; 00000110 - směr kurzoru, posunu displeje
      call  WR_CMD
      return

;************************************************************
; časy pro frekvenci krystalu 4,00Mhz
; časy jsou spočítané od call CEKxxm do návratu na volání CEKxxm
; celkový výpočet  = 2+6+(2+(TMP0-1)*3+2)+3)*TMP1-1+2
; zkrácený výpočet = 8+((TMP0-1)*3+7)*TMP1+1
; malá smyčka = (TMP2-1)*3+2
;-------------------------------
CEK2s movlw .20         ; TMP2 = 20  cekej 2 sekundu
      movwf TMP3
      goto  SMYCKA2
;-------------------------------
CEK100m     movlw 0CFh        ; cas = 100,009 ms
      movwf TMP0
      movlw 0A0h
      movwf TMP1
      goto  SMYCKA
;-------------------------------
CEK15m      movlw 0ABh        ; cas = 15,002 ms
      movwf TMP0
      movlw 01Dh
      movwf TMP1
      goto  SMYCKA
;-------------------------------
```

```
CEK4m movlw 0A5h        ; cas = 4,001 ms
      movwf TMP0
      movlw 008h
      movwf TMP1
      goto  SMYCKA
;------------------------------
CEK1m6      movlw 041h        ; cas = 1,601 ms
      movwf TMP0
      movlw 008h
      movwf TMP1
      goto  SMYCKA
;------------------------------
CEK100      movlw       01Dh        ; cas = 100 mikrosekund
      movwf TMP0
      movlw 001h
      movwf TMP1
      goto  SMYCKA
;------------------------------
CEK40 movlw 009h        ; cas = 40 mikrosekund
      movwf TMP0
      movlw 001h
      movwf TMP1
      goto  SMYCKA
;------------------------------
CEK19 movlw 002h        ; cas = 19 mikrosekund
      movwf TMP0
      movlw 001h
      movwf TMP1
      goto  SMYCKA
;------------------------------
SMYCKA2     call  CEK100m
      decfsz      TMP3,F            ; největší smyčka
      goto  SMYCKA2
      retlw 00h
;------------------------------
SMYCKA      movf  TMP0,W
      movwf TMP2
SMYCKA1     decfsz      TMP2,F            ; malá smyčka
      goto  SMYCKA1

      decfsz      TMP1,F            ; velká smyčka
      goto  SMYCKA
      retlw 00h
;*************************************************************

;
;*****************************************************************
;                 Double Precision Addition & Subtraction
;
;*****************************************************************;
;   Addition :  ACCb(16 bits) + ACCa(16 bits) -> ACCb(16 bits)
;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
;      (c) CALL D_add
;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
;
;   Performance :
;               Program Memory  :       07
;               Clock Cycles    :       08
```

```
;*****************************************************************;
;   Subtraction : ACCb(16 bits) - ACCa(16 bits) -> ACCb(16 bits)
;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
;      (c) CALL D_sub
;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
;
;   Performance :
;               Program Memory  :       14
;               Clock Cycles    :       17
;
;
;       Program:          DBL_ADD.ASM
;       Revision Date:
;                         1-13-97      Compatibility with MPASMWIN 1.40
;
;*****************************************************************;
;
;
;*****************************************************************
;         Double Precision Subtraction ( ACCb - ACCa -> ACCb )
;
D_sub   call    neg_A           ; At first negate ACCa; Then add
;
;*****************************************************************
;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
;
D_add   movf    ACCaLO,W
      addwf   ACCbLO, F       ;add lsb
      btfsc   STATUS,C        ;add in carry
      incf    ACCbHI, F
      movf    ACCaHI,W
      addwf   ACCbHI, F       ;add msb
      retlw   0
;
;
neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
      incf    ACCaLO, F
      btfsc   STATUS,Z
      decf    ACCaHI, F
      comf    ACCaHI, F
      retlw   0

;*****************************************************************
;    Load constant values to ACCa & ACCb for testing
nulAB movlw   0
      movwf   ACCaHI
      movlw   0             ; loads ACCa = 0
      movwf   ACCaLO

      movlw   0
      movwf   ACCbHI
      movlw   0            ; loads ACCb = 0
      movwf   ACCbLO

      movlw   0
      movwf   Azim_HI
      movlw   0            ; loads ACCb = 0
      movwf   Azim_LO
```

```
      movlw   0
      movwf   Vysk_HI
      movlw   0            ; loads ACCb = 0
      movwf   Vysk_LO

      return

;*********************************************************************

Az2B  movf  Azim_LO,0
      movwf ACCbLO
      movf  Azim_HI,0
      movwf ACCbHI
      retlw 0
;------------------------------

B2Az  movf  ACCbLO,0
      movwf Azim_LO
      movf  ACCbHI,0
      movwf Azim_HI
      retlw 0
;------------------------------

Vy2B  movf  Vysk_LO,0
      movwf ACCbLO
      movf  Vysk_HI,0
      movwf ACCbHI
      retlw 0
;------------------------------

B2Vy  movf  ACCbLO,0
      movwf Vysk_LO
      movf  ACCbHI,0
      movwf Vysk_HI
      retlw 0
;------------------------------
;*****************************************************************
Z_Azim      clrf  Zcislo
      call  Az2B
      btfsc Azim_HI,7
      goto  AZ2
AZ1   clrf  ACCaHI
      movlw 38          ; pocet dilku na stupen v azimutu
      movwf   ACCaLO


      call  D_sub
      btfsc ACCbHI,7
;     btfsc STATUS,C
      return

      incf  Zcislo
      goto  AZ1

AZ2   movlw 0xFF
      xorwf ACCbHI,1
      movlw 0xFF
      xorwf ACCbLO,1
      goto  AZ1
```

```
;-----------------------------------------

Z_Vysk      clrf  Zcislo
      call  Vy2B
      btfsc Vysk_HI,7
      goto  Vy2
Vy1   clrf  ACCaHI
      movlw 105         ; pocet dilku na stupen ve vysce
      movwf   ACCaLO


      call  D_sub
      btfsc ACCbHI,7
;     btfsc STATUS,C
      return

      incf  Zcislo
      goto  Vy1

Vy2   movlw 0xFF
      xorwf ACCbHI,1
      movlw 0xFF
      xorwf ACCbLO,1
      goto  Vy1

;*****************************************************************

      END
```