VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VÝVOJ SIMULINK BLOKU PRO AUTOMATICKÉ GENEROVÁNÍ KÓDU PRO EMBEDDED PROCESOR

THE DEVELOPMENT OF AN SIMULINK BLOCK FOR AUTOMATED CODE GENERATION TARGETED FOR EMBEDDED PROCESSOR

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — ANDREAS LUERMANN
AUTHOR

VEDOUCÍ PRÁCE — ING. VOJTĚCH LAMBERSKÝ
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Andreas Luermann

který/která studuje v **bakalářském studijním programu**

obor: **Mechatronika (3906R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Vývoj Simulink bloku pro automatické generování kódu pro embedded procesor

v anglickém jazyce:

The development of an Simulink block for automated code generation targeted for embedded processor

Stručná charakteristika problematiky úkolu:

Aby bylo možné efektivně programovat procesory, vyvíjí se stále nové programovací jazyky (mezi ně patří například Simulink) a rozšiřuje se podporovaný hardware (možnost zkompilovat kód pro určitý typ procesoru).
Tématem této práce bude vyvinout blok pro grafické programovací rozhraní Simulink, který umožní generovat C kód, přeložitelný pro embedded procesoru, zajišťující komunikaci s řadičem displeje. Tento blok umožní významně zrychlit vývoj software pro embedded procesory tam, kde je potřeba uživateli zobrazovat informace na segmentovém display.

Cíle bakalářské práce:

1. Vytvořit blok v matlabu, který zajistí generování kódu pro obsluhu displeje.
2. Vytvořit návod jak navrhovat mex funkce v Matlabu, které se používají pro generování kódu pro embedded zařízení.
3. Vytvořit návod jak napsat tlc funkci v Matlabu, která bude generovat inlinovaný C kód.
4. Vytvořit knihovnu v Matlabu (registrace bloku s vytvořeným blokem pro generování kódu pro výpis hodnot na display), tak aby šla snadno přenést na jiný počítač.

Seznam odborné literatury:

P. Herout, Učebnice jazyka C, 2009
Simulink Coder [dokumentace k softwaru]
TI teaching rom [soubor elektronických výukových materiálů]

Vedoucí bakalářské práce: Ing. Vojtěch Lamberský

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2012/2013.

V Brně, dne 24.10.2012

L.S.

prof. Ing. Jindřich Petruška, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

ABSTRAKT

Táto práce se zabývá vytvořením bloku v prostředí Simulink, který bude generovat automatický kód pro obsluhu alfanumerického LCD (Liquid-crystal display).

ABSTRACT

This thesis describes development of an Simulink block for automated code generation, targeted for setting an alphanumeric LCD (Liquid-crystal display).

KLÍČOVÁ SLOVA

Simulink, Automatické generování kódu, Řadič Hitachi HD44780, dsPIC mikro-kontroler, Explorer 16 Development Board, LCD

KEYWORDS

Simulink, Automated code generation, Hitachi HD44780 controller, dsPIC micro-controller, Explorer 16 Development Board, LCD

BIBLIOGRAFICKÁ CITACE

LUERMANN, A. Vývoj Simulink bloku pro automatické generování kódu pro embedded procesor. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 32 s. Vedoucí bakalářské práce Ing. Vojtěch Lamberský.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci na téma Vývoj Simulink bloku pro automatické generování kódu pro embedded procesor vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který tvoří přílohu této práce.

22. května 2013 ……

Andreas Luermann

PODĚKOVÁNÍ

Děkuji tímto vedoucímu mé bakalářské práce Ing. Vojtěchovi Lamberskému za jeho odbornou pomoc.

OBSAH

1. ÚVOD

Dôležitou schopnosťou moderných zariadení je podanie potrebných informácií o ich aktuálnom stave užívateľovi. Z toho dôvodu je potrebné myslieť na užívateľské rozhranie už počas návrhu software.

To je dôvodom popularity alfanumerických displejov v niektorých aplikáciách. Ich využívanie zjednodušuje množstvo informácií a návodov na internete. Napriek tomu však spojazdnenie displeja môže zabrať niekoľko hodín práce. V rámci efektivity výroby a návrhu zariadení je teda vhodné vytvoriť nástroj, ktorý pomocou niekoľkých zadaných parametrov vytvorí kód potrebný na správne nastavenie displeja.

V nasledujúcich kapitolách bude uvedený postup, akým je možné vytvoriť tento nástroj – blok v prostredí firmy MathWorks – Simulink, jeho pridanie do knižnice a taktiež jednoduchý príklad použitia tohto nástroja.

Ďalšie zvýšenie efektivity môže byť dosiahnuté, ak tento blok bude fungovať ako rozšírenie pre Kerhuel toolbox. Ten bol vytvorený na prácu s mikrokontrolermi dsPIC, ktoré budeme k ovládaniu displeja využívať.

2. CIEĽ PRÁCE

Hlavným cieľom tejto práce je vytvoriť blok v prostredí Simulink, ktorý vygeneruje kód na ovládanie alfanumerického displeja s radičom od firmy Hitachi – HD44780. Tento blok by mal zároveň fungovať ako doplnenie Kerhuel toolboxu. Súčasťou tohto cieľa je aj demonštračný príklad využitia bloku.

Ďalším cieľom je vytvorenie návodu na tvorbu bloku v prostredí Simulink.

3. OVLÁDANIE ZNAKOVÉHO LCD DISPLEJA

Zobrazovanie na displeji zabezpečuje tzv. radič. Jedná sa o integrovaný obvod, ktorého funkciou je ovládanie displeja[1] a jeho komunikácia s nadradeným zariadením[2]. Vďaka unifikácii radičov budú s rovnakým ovládacím kódom fungovať displeje od rôznych výrobcov.

3.1. SIGNÁLY OVLÁDAJÚCE RADIČE

Ovládanie radičov vo všeobecnosti zabezpečujú:

a) riadiace signály

E – enable data transmit – umožnenie odovzdania dát – ide o údaj z hodín nadradeného zariadenia

R/W - read / write – smer toku dát (čítanie z displeja, alebo zápis na displej)

RS – Register Select – určuje druh dát – resp. či je radiču posielaný znak, alebo príkaz

b) dátové signály

DBn – Dátový bit (n – poradie bitu)

Pri jednoduchých alfanumerických displejoch býva dátových bitov 4 / 8.

3.2. RADIČ HITACHI HD44780

Radič, ktorý bol vybraný pre našu prácu vyvinula firma Hitachi. Jeho označenie je HD44780.

Tento radič bol vybraný s ohľadom na to, že je to jeden z najbežnejších radičov znakových LCD displejov. Taktiež poskytuje možnosť 4 a 8 bitovej komunikácie[3], čo znižuje nároky na hardware[4]. Minimálny počet potrebných pinov môže byť teda znížený na 9 (3 riadiace, 4 dátové bity, 2 na napájanie). Pre viac informácií o radiči viď [7].

---

[1] To znamená napr. zobrazovanie znakov na správnom mieste, nastavenie kurzora, mazanie a pod.

[2] Zariadenie, od ktorého prichádza signál.

[3] Resp. 4 / 8 dátových bitov.

[4] Keďže býva často využívaný k ovládaniu displejov určených pre embedded systémy, je to dôležitá vlastnosť.

pin	názov	Popis
1	Vss	napájanie GND
2	Vdd	napájanie +5V
3	Vo	kontrast 0V .. 5V
4	RS	Register Select (0=príkaz, 1=znak)
5	R/W	Read / Write (0=zápis, 1=čítanie)
6	E	Enable
7	DB0	Data Bus 0
8	DB1	Data Bus 1
9	DB2	Data Bus 2
10	DB3	Data Bus 3
11	DB4	Data Bus 4
12	DB5	Data Bus 5
13	DB6	Data Bus 6
14	DB7	Data Bus 7
15	A	podsvietenie - anóda LED
16	K	podsvietenie - katóda LED

TAB. 3.1: ALOKÁCIA PINOV RADIČA HITACHI HD44780, PREVZATÉ Z: [7]

3.2.1. SADA ZNAKOV

TAB. 3.2: SADA ZNAKOV RADIČA HITACHI HD44780, PREVZATÉ Z: [7]

Radič HD44780 má okrem znakov uložených v DDRAM (display data) aj funkciu tvorby vlastných znakov a ich uloženie do pamäte CGRAM (character generator RAM). Maximálny počet takýchto znakov je 8. Pre zápis nového znaku do CGRAM sa používa funkcia v tvare: *lcd_cgram(adresa v CGRAM, pole nového znaku)* a jeho výpis na displej *lcd_putc(adresa v CGRAM).* Podrobnejší popis vytvorenia užívateľského znaku nájdete na: [6]

3.2.2. SADA PRÍKAZOV

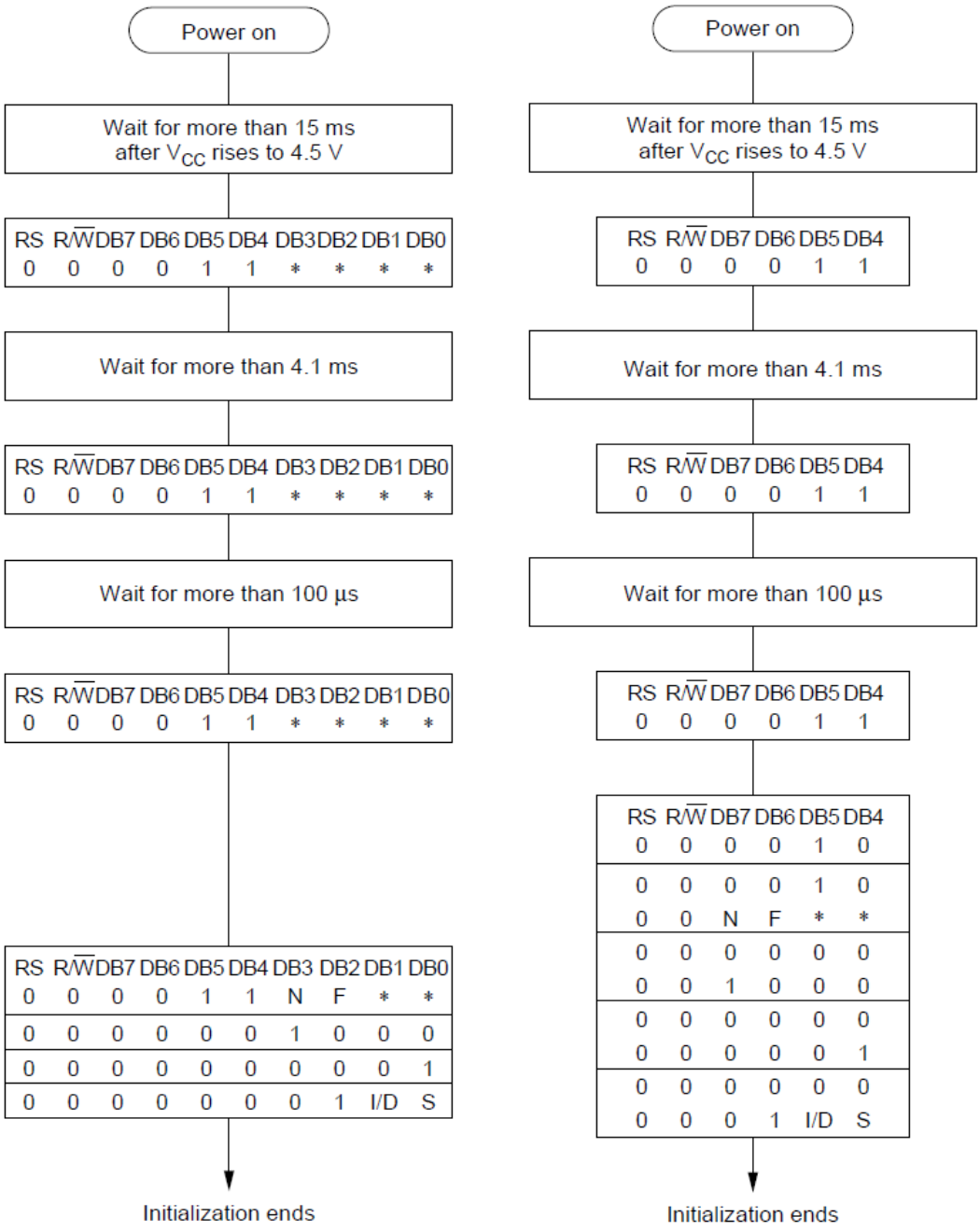
Príkazy	Signál									
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Zmazať displej	0	0	0	0	0	0	0	0	0	1
Návrat na pozíciu 0	0	0	0	0	0	0	0	0	1	*
Nastavenie módu	0	0	0	0	0	0	0	1	I/D	S
Kontrola displeja zap/vyp	0	0	0	0	0	0	1	D	C	B
Posun kurzoru, displeja	0	0	0	0	0	1	S/C	R/L	*	*
Nastavenie funkcie	0	0	0	0	1	DL	N	F	*	*
Nastavenie adresy CGRAM	0	0	0	1	CGRAM adresa					
Nastavenie adresy DDRAM	0	0	1	DDRAM adresa						
Čítanie príznaku Busy Flag a adresy	0	1	BF	CGRAM/DDRAM adresa						
Zápis dát do CGRAM nebo DDRAM	1	0	zápis dát							
Čítanie dát z CGRAM nebo DDRAM	1	1	čítanie dát							

TAB. 3.3: SADA PRÍKAZOV RADIČA HITACHI HD44780, PREVZATÉ Z: [5]

Vysvetlivky k tab 3.3:
* - na hodnote nezáleží
I/D – zvýši (1), alebo zníži (0) DDRAM adresu o 1
S – posunie displej (1)
D – zapnutý (1) / vypnutý (0) displej
C – zobrazenie kurzoru zapnuté (1) / vypnuté (0)
B – znak, na ktorý ukazuje kurzor bliká (1) / bez blikania (0)
S/C – posun kurzoru (0) / displeja (1)
R/L – posun doprava (1) / doľava (0)
DL – 4-bitová (0) / 8-bitová(1) komunikácia
N – počet riadkov displeja
F – font písma
BF – busy flag => môže (0) / nemôže (1) prijať príkaz

3.2.3. 4- A 8-BITOVÁ INICIALIZÁCIA PRÍKAZMI

Po pripojení displeja ku zdroju napätia je pre komunikáciu potrebné vykonať inicializačnú sekvenciu – sled príkazov poslaných na displej. Pretože radič HD44780 podporuje tak 4-bitovú, ako aj 8-bitovú komunikáciu, rozlišujeme dva varianty inicializácie. Viď obr. 3.1.



OBR. 3.1: INICIALIZAČNÉ SEKVENCIE (8-BITOVÁ NAĽAVO, 4-BITOVÁ VPRAVO) PREVZATÉ Z: [7]

3.2.4. ZÁPIS DÁT A PRÍKAZOV

Adresa každého znaku v pamäti sa skladá z ôsmich bitov. Napríklad písmeno A má adresu 0100-0001. Prvé štyri čísla tvoria „horný“ byte a posledné štyri tvoria „dolný“ byte. Pri osembitovej komunikácii je celá adresa poslaná naraz. V prípade štvorbitovej komunikácie je nutné poslať ich oddelene. Posielanie príkazov funguje analogicky.

8-Bitový zápis
EN = 0
R/S = 0 (príkaz) / 1(znak)
Poslanie potrebných hodnôt na DB7 – DB0
EN = 1
Čakať aspoň 450ns
EN = 0
Čakaj 5ms pre príkaz, 200µs pre zápis
4-Bitový zápis
EN = 0
R/S = 0 (príkaz) / 1(znak)
Pošli horný byte znaku, alebo príkazu na DB7-DB4
EN = 1
Čakať aspoň 450ns
EN = 0
Čakaj 5ms pre príkaz, 200µs pre zápis
Pošli dolný byte znaku, alebo príkazu na DB7-DB4
Čakať aspoň 450ns
EN = 0
Čakaj 5ms pre príkaz, 200µs pre zápis

TAB. 3.4: 8- A 4-BITOVÁ KOMUNIKÁCIA, PREVZATÉ Z: [5]

Slovne:

8-bitová komunikácia – využívame všetky dátové piny (DB0 - DB7). Najprv nastavíme pin RS na 1, ak chceme poslať znak, alebo 0, ak chceme poslať príkaz. Na piny DB0 - DB7 pošleme potrebné údaje a aktivujeme pin E. Po jeho deaktivácii je znak, resp. príkaz odoslaný. Pin R/W zostáva počas celého procesu na 0 (na displej zapisujeme).

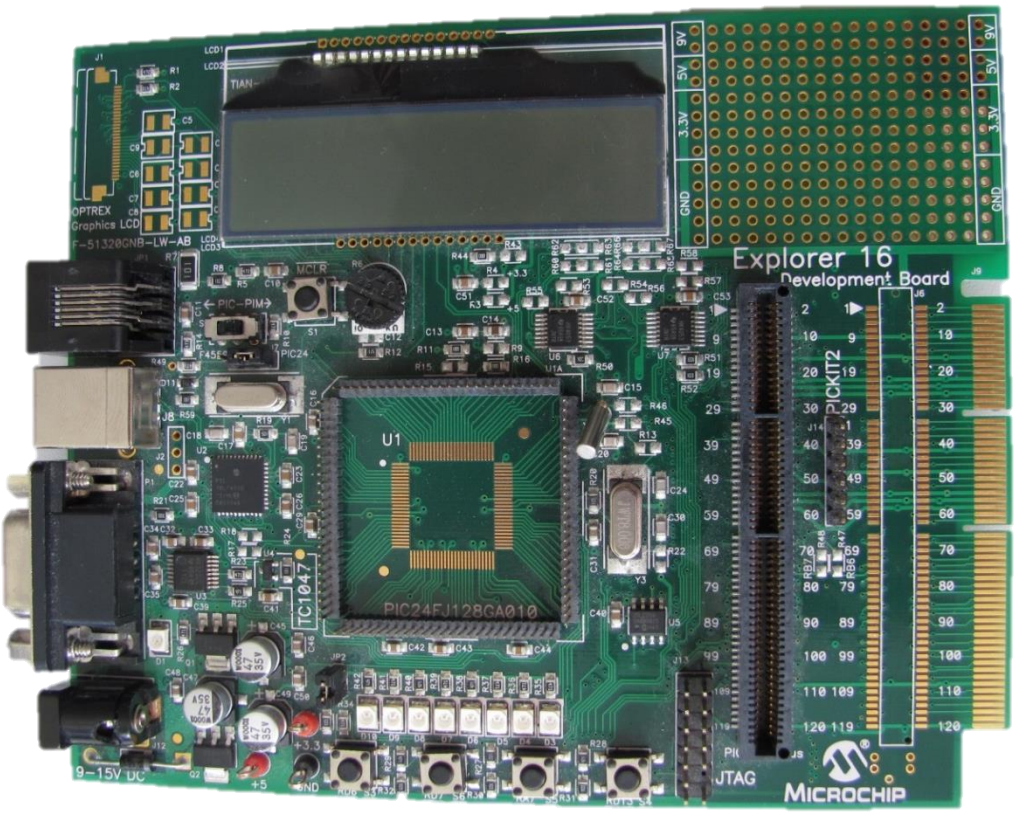
4-bitová komunikácia – využívame len dátové piny DB4 - DB7 (zvyšné sú uzemnené). Posielanie dát funguje analogicky. Najprv je ale odoslaný horný byte, a po prepnutí pinu E proces opakujeme pre dolný byte.

4. VÝVOJOVÁ DOSKA A MIKROKONTROLER

4.1. VÝVOJOVÁ DOSKA

Bola zvolená vývojová doska Explorer 16 od firmy Microchip. Táto vývojová doska je určená pre 16-bitové a 32-bitové mikrokontrolery[5]. Viac informácií o vývojovej doske nájdete v data sheet [3].

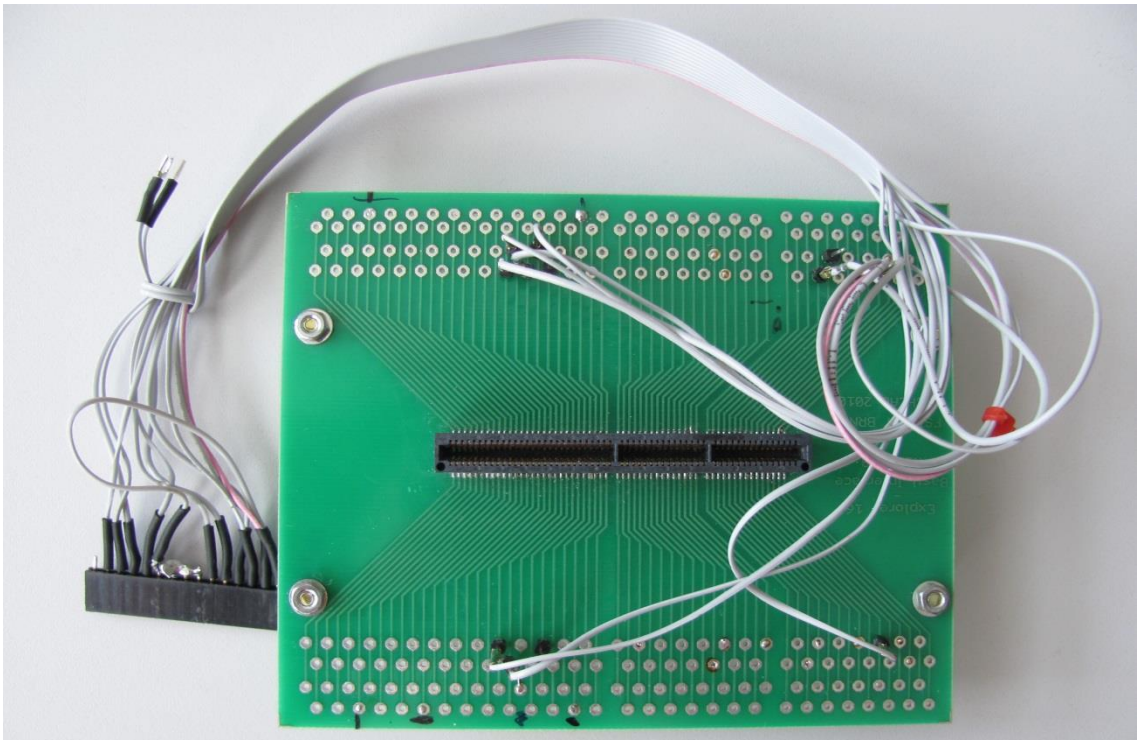
V doske je integrovaný dvojriadkový displej. Jeho radič je však nevhodný pre našu aplikáciu. Budeme využívať externý displej.



OBR. 4.1: EXPLORER 16 DEVELOPMENT BOARD

Výhodou tejto vývojovej dosky je tzv. PICTail™ Plus daughter card konektor, ktorý uľahčuje prístup k jednotlivým pinom mikrokontrolera. Na pripojenie pinov použijeme *Explorer 16 Bastl interface R100119*, vyvinutý v laboratóriu MechLab.

[5] Presnejšie: mikrokontrolery z rodín PIC24, dsPIC a PIC32

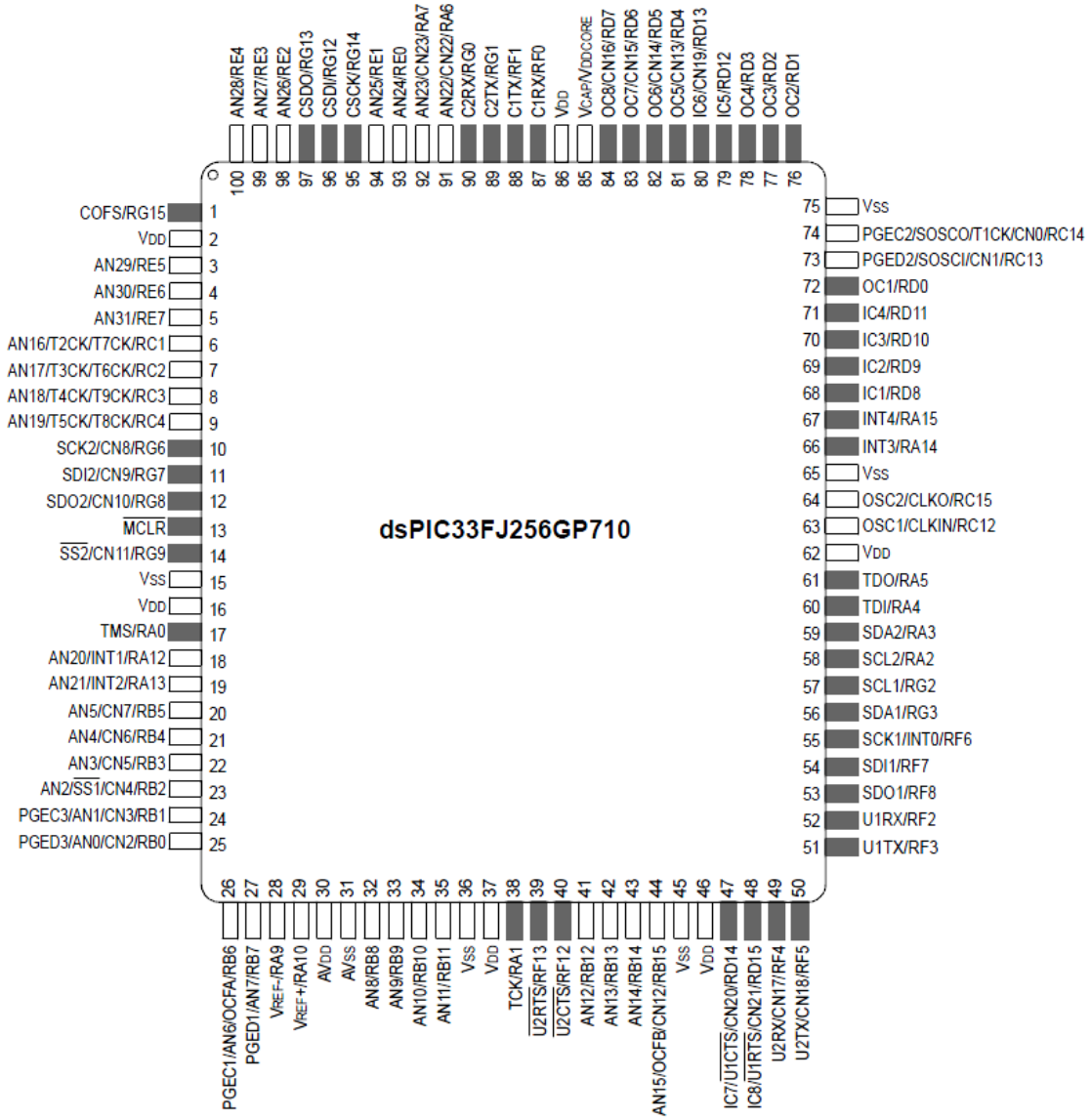


OBR. 4.2: EXPLORER 16 BASTL INTERFACE R100119 S KONEKTOROM NA PRIPOJENIE DISPLEJA

4.2. MIKROKONTROLER

Budeme používať mikrokontroler Microchip dsPIC33FJ256GP710 – mikrokontroler so 16-bitovou architektúrou (upravený Harvardský typ). Rýchlosť procesora je 40 MIPS, programová flash pamäť má veľkosť 256 kB. Obsahuje 85 I/O pinov (celkový počet je 100 – viď obr. 4.4). Viac informácií v datasheete [2].

OBR. 4.3: MICROCHIP DSPIC33FJ256GP710

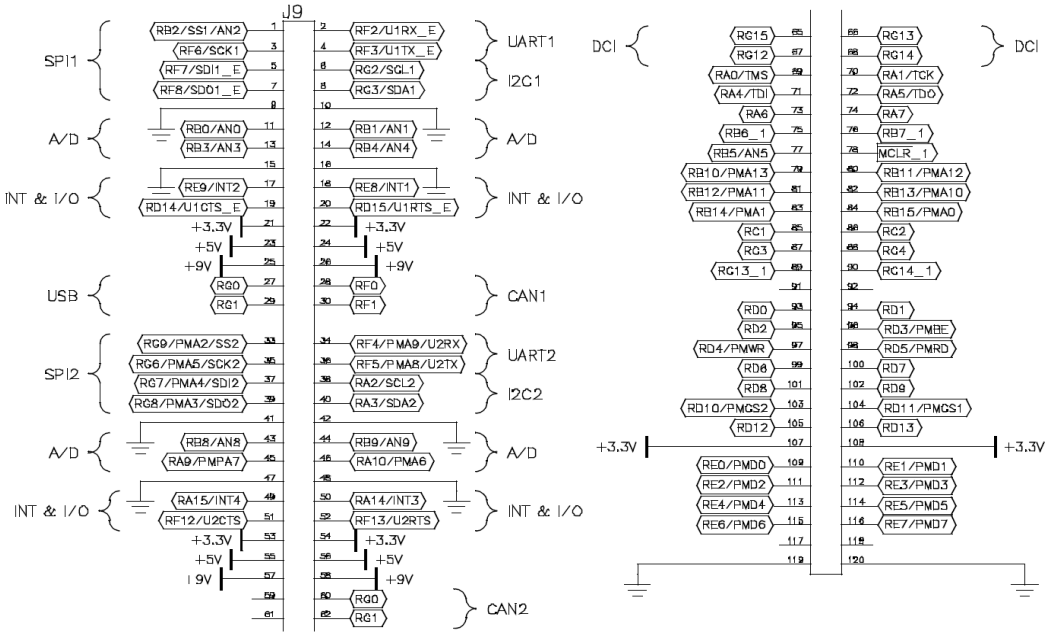


OBR. 4.4: PIN DIAGRAM, PREVZATÉ Z: [3]

5. PREPOJENIE MIKROKONTROLERA S DISPLEJOM

Na prepojenie mikrokontrolera s displejom použijeme už spomínaný[6] PICTail™ Plus konektor a Explorer 16 Bastl interface R100119.

Prvým krokom pri prepojení je vyhľadanie portu vhodného na ovládanie displeja. Ten musí mať 7 pinov, z toho 4 by mali byť vedľa seba (uľahčuje to tvorbu software). Na tieto 4 piny pripojíme DB4-DB7. Zvyšné tri sú potrebné pre E, R/W a RS. Takýto port vyhľadáme v data sheet-e vývjovej dosky.



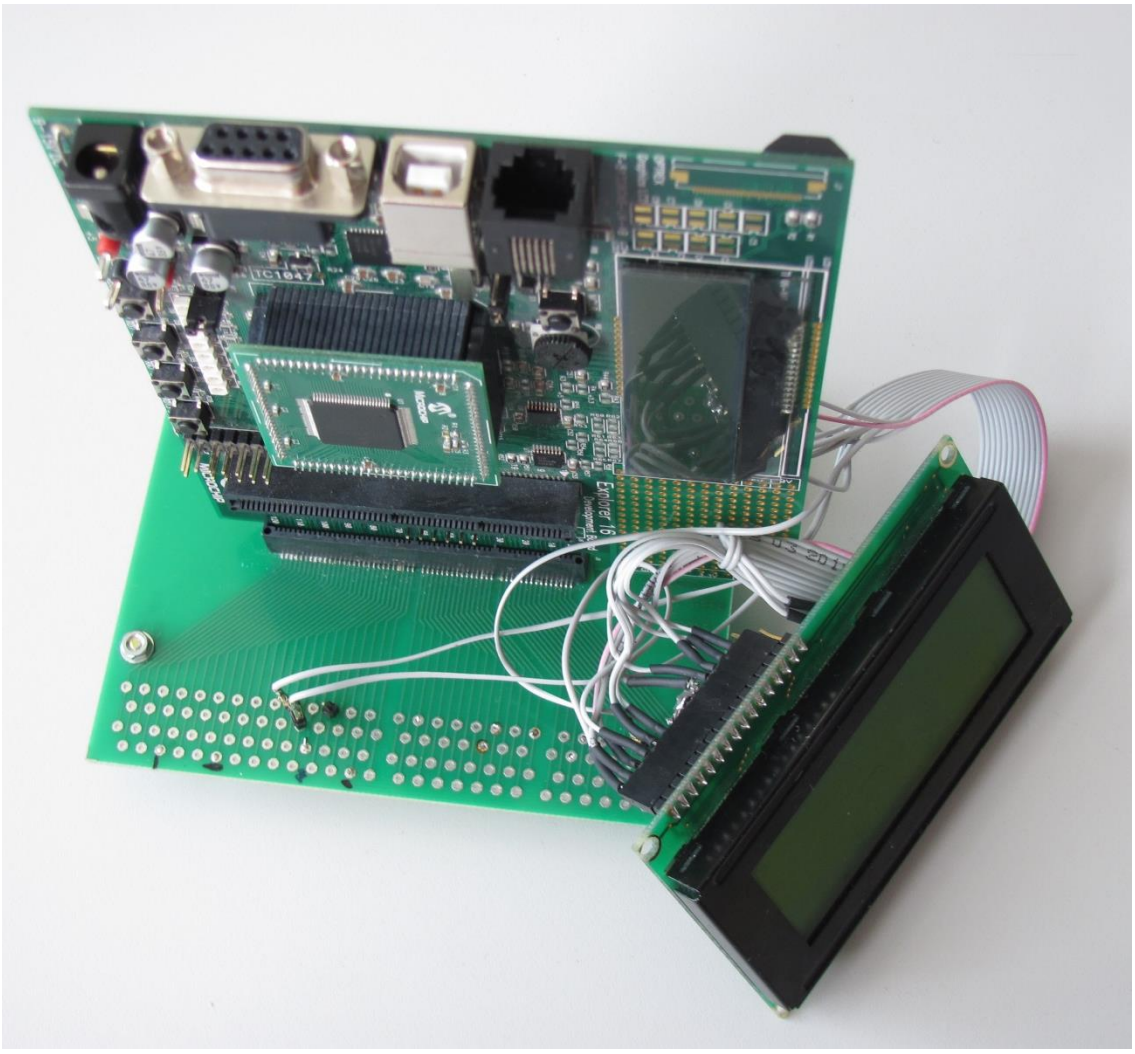
OBR. 5.1: SCHÉMA VÝVOJOVEJ DOSKY, PREVZATÉ Z: [3]

V našom prípade bol zvolený port B. Konkrétne rozloženie pinov je uvedené v tabuľke 5.1. Zvyšné výstupy displeja je potrebné uzemniť.

Radič		Vývojová doska	
Pin číslo:	Symbol	Pin číslo:	Pin
4	RS	12	RB1
5	R/W	79	RB10
6	EN	77	RB5
11	DB4	81	RB12
12	DB5	82	RB13
13	DB6	83	RB14
14	DB7	84	RB15

TAB. 5.1: PRIRADENIE PINOV

[6] Kapitola 4.1.



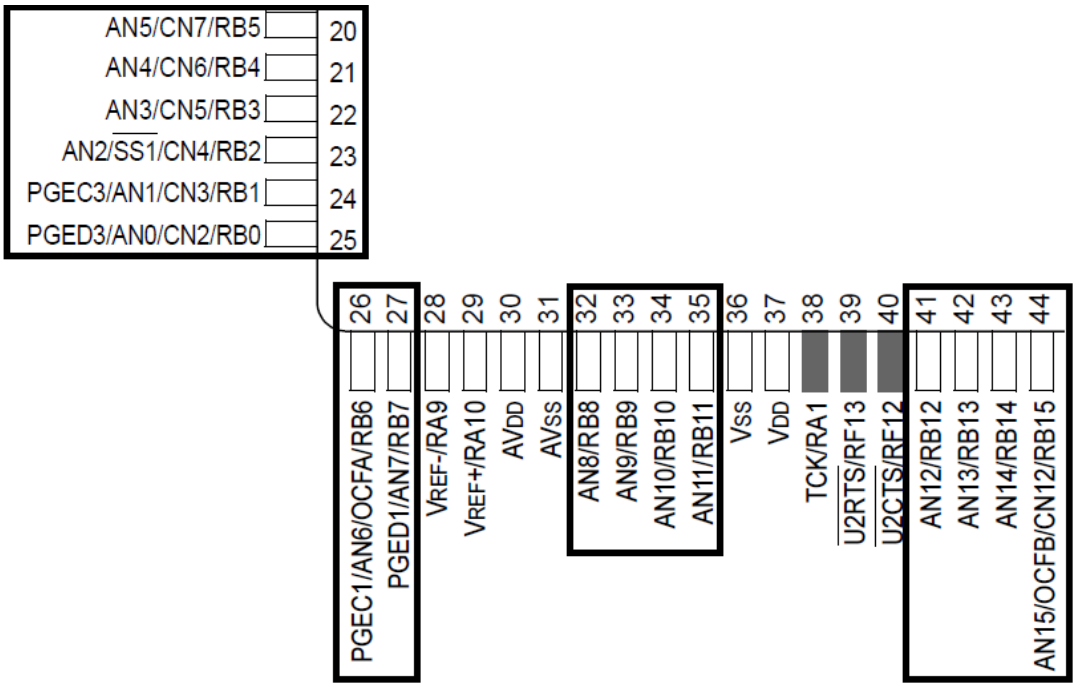
OBR. 5.2: PREPOJENIE MIKROKONTROLERA, DEVELOPMENT BOARD A DISPLEJA

6. REGISTRE AD#PCFGL

Na vytvorenie bloku na ovládanie LCD displeja bol potrebný kód, ktorý by správne odosielal dáta. Vhodný software je možné nájsť na internete. Kvôli využitiu iného mikrokontrolera a iného portu je pre správnu funkciu nutné tento kód upraviť.

Základným krokom bolo prepísanie pinov podľa nášho zapojenia. Keďže port B zdieľa piny s analógovým vstupom AD prevodníka, a ten je implicitne nastavený, je nutné prepnúť vstup pinov na port B[7].

Prepínanie vstupu pinov zabezpečujú registre. Ďalším krokom preto bude vyhľadať zodpovedajúci register v data sheet-e mikrokontrolera [2].



OBR. 6.1: DIAGRAM PINOV DSPIC33FJ256GP710 (ZVÝRAZNENÝ PORT B), PREVZATÉ Z: [2]

[7] Ide o tzv. multiplexing. Počet pinov mikrokontrolera je redukovaný tým, že ich využívajú rôzne periférie. Viď obr. 6.1.

REGISTER 21-10: ADxPCFGL: ADCx PORT CONFIGURATION REGISTER LOW(1,2,3)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **PCFG<15:0>:** ADC Port Configuration Control bits
1 = Port pin in Digital mode, port read input enabled, ADC input multiplexer connected to AVss
0 = Port pin in Analog mode, port read input disabled, ADC samples pin voltage

Note 1: On devices without 16 analog inputs, all PCFG bits are R/W by user. However, PCFG bits are ignored on ports without a corresponding input on device.

2: On devices with two analog-to-digital modules, both AD1PCFGL and AD2PCFGL will affect the configuration of port pins multiplexed with AN0-AN15.

3: PCFGx = ANx, where x = 0 through 15.

OBR. 6.2: REGISTER 21-10, PREVZATÉ Z: [2]

Na obr. 6.2: poznámka 2 hovorí, že na zariadeniach s dvomi AD modulmi budú registre AD1PCFGL a AD2PCFGL ovplyvňovať porty multiplexované s AN0-AN15.

V časti zvýraznenej modrou farbou ďalej zistíme, že konfiguračné bity AD prevodníka PCFG<15-0> prepínajú digitálny (1) a analógový (0) mód. V našom prípade musí teda byť na začiatku kódu priradená registrom AD1PCFGL a AD2PCFGL hodnota 1 – digitálny mód.

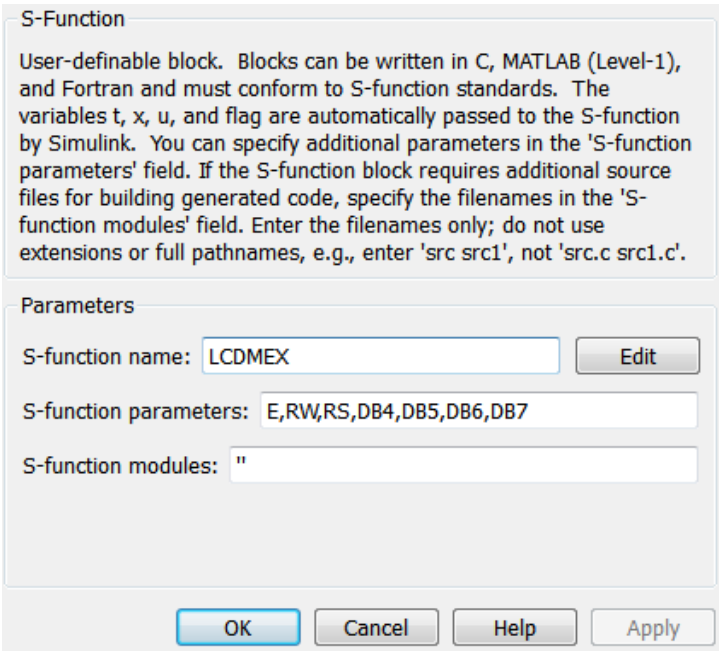
```
////////////////////////////////////
//Display Init
void LCD_Init(void)
{

  AD1PCFGL = 0xFFFF;
  AD2PCFGL = 0xFFFF;
```

OBR. 6.3: UKÁŽKA Z KÓDU – AD#PCFGL

7. AUTOMATICKÉ GENEROVANIE KÓDU

7.1. VYTVORENIE BLOKU

Teraz je potrebné vytvoriť blok, ktorý bude generovať kód na obsluhu displeja. V Simulinku založíme nový model a z knižnice doň vložíme blok s názvom S-function[8]. Následne otvoríme dialóg s parametrami bloku (viď obr. 7.1).



OBR. 7.1: BLOCK PROPERTIES

Do voľby *S-function name* je potrebné zadať názov C-MEX funkcie, ktorá sa bude vykonávať po spustení simulácie. Táto funkcia obsahuje informácie o vstupoch, výstupoch a ďalších parametroch bloku). Do *S-function parameters* uložíme parametre, ktoré budeme predávať kódu na obsluhu displeja. V našom prípade budú parametrami označenia pinov nášho mikrokontrolera.

[8] Cesta k bloku: User-Defined Functions → S-Function

7.1.1. VYTVORENIE C-MEX FUNKCIE

Ak nemáme vhodnú C-MEX funkciu, musíme ju vytvoriť. Napísanie C-MEX funkcie uľahčujú šablóny, ktoré je možné nájsť v knižnici Simulinku pod názvom *Basic C-MEX Template*[9].

V prvom kroku pri písaní S-funkcie definujeme meno bloku pomocou makra. Napr.:
#define S_FUNCTION_NAME LCDMEX

Nastavíme počet parametrov (7 pinov na ovládanie displeja), ktoré budeme vkladať do generovaného kódu funkciou *ssSetNumSFcnParams* a zrušíme možnosť zmeny hodnoty počas simulácie, pomocou príkazu *ssSetSFcnParamTunable*.

Ďalej nastavíme počet vstupných portov na 4 (teda počet riadkov displeja a vstupných reťazcov). Využijeme funkciu *ssSetNumInputPorts* (analogicky vynulujeme počet výstupných portov príkazom *ssSetNumOutputPorts*). Šírku každého vstupu je potrebné nastaviť na 21 (20 znakov/riadok + posledný člen vstupného vektoru musí byť nulový). To je možné nastaviť funkciou *ssSetInputPortWidth.*

Vstupný port musí byť typu uint8. Tento typ má rozmer 8 bitov, t.j. rovnaký rozmer, aký má aj dátový typ char (s ním pracuje náš kód).

Využitie týchto funkcií môžeme vidieť na obr. 7.2.

```
61  static void mdlInitializeSizes(SimStruct *S)
62  {
63      ssSetNumSFcnParams(S, 7);  /* Number of expected parameters */
64      if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
65      return;
66      }
67      int i;
68      for (i = 0, i = 6, i++){
69          ssSetSFcnParamTunable(S, i, false);
70      }
71
72      ssSetNumContStates(S, 0);
73      ssSetNumDiscStates(S, 0);
74
75      if (!ssSetNumInputPorts(S, 4)) return;
76      ssSetInputPortWidth(S, 0, 21);
77      ssSetInputPortWidth(S, 1, 21);
78      ssSetInputPortWidth(S, 2, 21);
79      ssSetInputPortWidth(S, 3, 21);
80
81      ssSetInputPortDataType(S, 0, SS_UINT8);
82      ssSetInputPortDataType(S, 1, SS_UINT8);
83      ssSetInputPortDataType(S, 2, SS_UINT8);
84      ssSetInputPortDataType(S, 3, SS_UINT8);
```

OBR. 7.2: UKÁŽKA KÓDU – POUŽITIE FUNKCIÍ

[9] Cesta: User-Defined Functions → S-Function Examples → C-file S-functions → Basic C-MEX Template

Ďalej potrebné nastaviť *Sample time* – vzorkovacia perióda bloku – na „zdedená“. Viď obr. 7.3.

```
121  static void mdlInitializeSampleTimes(SimStruct *S)
122  {
123      ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
124      ssSetOffsetTime(S, 0, 0.0);
125
126  }
```

OBR. 7.3: UKÁŽKA KÓDU – NASTAVENIE SAMPLE TIME

Parametre, ktoré zadávame do bloku S-function musíme vo funkcii C-MEX uložiť ako makro preprocesora (obr. 7.4). Najprv uložíme každý parameter do makra využitím príkazu *ssGetSFcnParam.*

```
222  #define E(S)        (ssGetSFcnParam(S,0))
223  #define RS(S)       (ssGetSFcnParam(S,1))
224  #define RW(S)       (ssGetSFcnParam(S,2))
225  #define DB4(S)      (ssGetSFcnParam(S,3))
226  #define DB5(S)      (ssGetSFcnParam(S,4))
227  #define DB6(S)      (ssGetSFcnParam(S,5))
228  #define DB7(S)      (ssGetSFcnParam(S,6))
229  #if defined(MATLAB_MEX_FILE)
230  #define MDL_RTW
```

OBR. 7.4: DEFINOVANIE MAKIER PREPROCESORA

Následne ich uložíme do poľa premenných typu char. Rozmer poľa volíme s rezervou, aby nedochádzalo ku chybám spôsobeným prístupom k nesprávnym častiam pamäte. Z týchto polí vytvoríme funkciou *mxGetString* reťazce (obr. 7.5).

```
232  static void mdlRTW(SimStruct *S)
233  {
234   char E[10];
235   char RS[10];
236   char RW[10];
237   char DB4[10];
238   char DB5[10];
239   char DB6[10];
240   char DB7[10];
241   mxGetString(E(S),E,10);
242   mxGetString(RS(S),RS,10);
243   mxGetString(RW(S),RW,10);
244   mxGetString(DB4(S),DB4,10);
245   mxGetString(DB5(S),DB5,10);
246   mxGetString(DB6(S),DB6,10);
247   mxGetString(DB7(S),DB7,10);
```

OBR. 7.5: VYTVORENIE POĽA PREMENNÝCH

Posledným krokom zostáva zapísanie reťazcov do štruktúry pomocou príkazu *ssWriteRTWParamSettings*. Zadáme doň parameter *SSWRITE_VALUE_STR*, ktorý sa používa pri práci s reťazcami. Vytvorený reťazec má názov *SFcnParamSettings* (obr. 7.6).

```
248   if (!ssWriteRTWParamSettings(S, 7, SSWRITE_VALUE_STR,"E",&E,
249                                      SSWRITE_VALUE_STR,"RS",&RS,
250                                      SSWRITE_VALUE_STR,"RW",&RW,
251                                      SSWRITE_VALUE_STR,"DB4",&DB4,
252                                      SSWRITE_VALUE_STR,"DB5",&DB5,
253                                      SSWRITE_VALUE_STR,"DB6",&DB6,
254                                      SSWRITE_VALUE_STR,"DB7",&DB7)) return;
```

OBR. 7.6: UKÁŽKA KÓDU – ZÁPIS PARAMETROV DO ŠTRUKTÚRY

Názov takto vytvorenej C-MEX funkcie vložíme do parametru bloku S-funkcie.

7.1.2. VYTVORENIE MASKY BLOKU

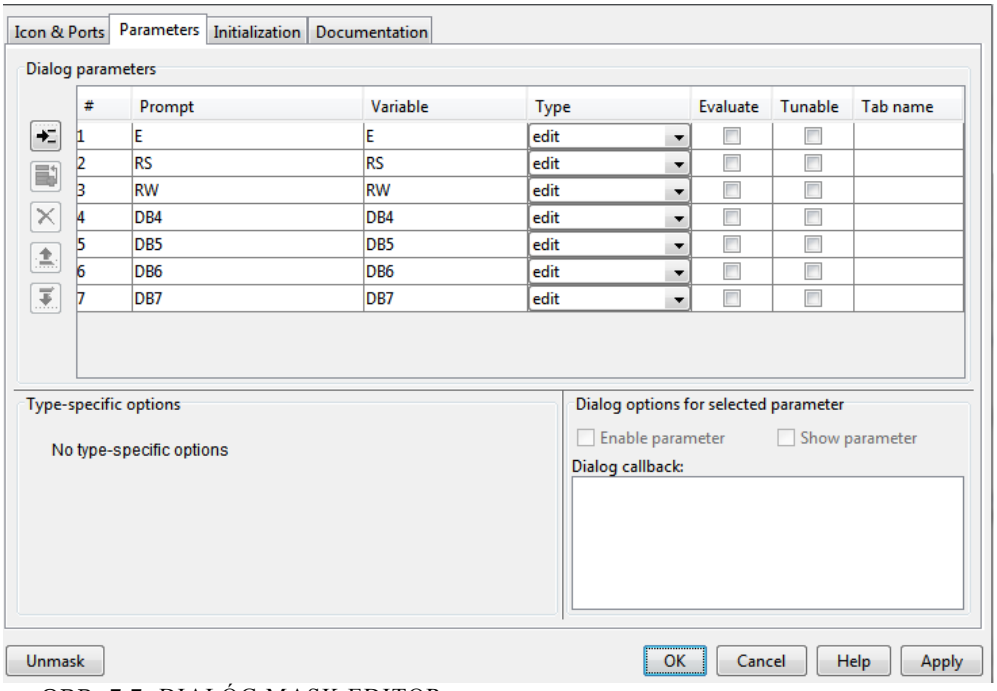
Z dôvodu sprehľadnenia užívateľského rozhrania je vhodné využiť tzv. maskovanie bloku. Parametre je potom možné bloku predávať prostredníctvom dialógu.

Masku vytvoríme označením bloku a stlačením klávesovej skratky Ctrl+M. V dialógu otvoríme voľbu „Parameters“. Tu pridáme požadovaný počet parametrov (obr. 7.7).

Do stĺpca „Variable“ napíšeme názov premennej, do ktorej uložíme označenie pinu (s týmto názvom bude pracovať C-MEX funkcia). Do stĺpca „Prompt“ vložíme text, ktorý bude zobrazený v dialógu na vloženie parametrov.

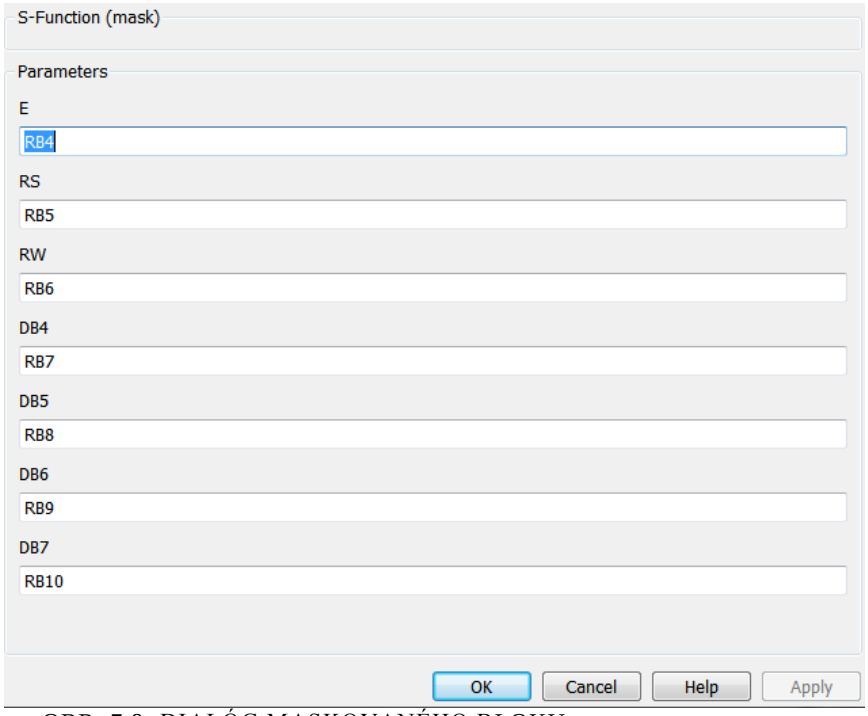
Stĺpec „Evaluate“ určí, či bude zadaný text vyhodnocovaný (tj. dosadenie hodnoty z workspace), alebo bude ponechaný ako reťazec – v našom prípade musíme možnosť „Evaluate“ zrušiť.

Podobne aj voľbu „Tunable“ je potrebné zakázať. Táto voľba totiž rozhoduje o možnosti zmeny zadaného parametera v priebehu simulácie, čo je v našom prípade nežiadúce.



OBR. 7.7: DIALÓG MASK EDITOR

Vykonané zmeny uložíme stlačením „OK“. Po dvojkliknutí na blok zamaskovanej S-funkcie sa otvorí dialóg, v ktorom zadávame parametre. Tento dialóg môžeme vidieť na obr. 7.8.



OBR. 7.8: DIALÓG MASKOVANÉHO BLOKU

7.1.3. VYTVORENIE TLC FUNKCIE

O samotné generovanie kódu v jazyku C sa stará funkcia tlc. Táto funkcia vytvorí súbory, do ktorých vloží zodpovedajúci kód. Pri vytváraní tlc funkcie je možné využiť príklady modelov s S-funkciami z knižnice Simulinku (nachádzajú sa v S-function examples). V týchto modeloch sú odkazy na C-MEX aj tlc funkcie.

Naša tlc funkcia musí vytvoriť hlavičkový súbor LCD.h, súbor s funkciami LCD.c a súbor delay.h, ktorý definuje oneskorenie v mili- a mikro-sekundách. Tieto 3 kódy vytvorí funkcia *%openfile*. Sekcia zapisovaná do kódu je ukončená funkciou *%closefile.*

```
7  %openfile temp="LCD.h"
8  #ifndef __LCD_H
9  #define __LCD_H
10 void LCD_Init(void);
11 void LCD_Busy (void);
12 inline void Toggle_E(void);
13 inline void sendHEX(unsigned char xx);
14 inline void sendChar(unsigned char xxx);
15 void sendStr(char * str, unsigned char CR);
16 void gotoXY(unsigned char col, unsigned char row);
17 #endif
18 %closefile temp
```

OBR. 7.9: UKÁŽKA KÓDU – VYTVORENIE SÚBORU LCD.H

Vytvorenie hlavičkového súboru z obr. 7.9 nevyžadovalo predávanie parametrov. V súbore LCD.c je ale potrebné uložiť označenie pinov na správne miesto. Tie uložila C-MEX funkcia do štruktúry *SFcnParamSettings*. Pomocou funkcie *CAST* uložíme hodnoty z tejto štruktúry do premenných.

Príkaz na priradenie hodnoty zo štruktúry *SFcnParamSettings.E* ako reťazec do premennej E: *%assign E = CAST("String", SFcnParamSettings.E).*

Analogicky uložíme všetky hodnoty.

Do kódu teraz vložíme potrebný text nasledovne: *%<premenná>*

```
43  %assign E = CAST("String", SFcnParamSettings.E)
44  %assign RS = CAST("String", SFcnParamSettings.RS)
45  %assign RW = CAST("String", SFcnParamSettings.RW)
46  %assign DB4 = CAST("String", SFcnParamSettings.DB4)
47  %assign DB5 = CAST("String", SFcnParamSettings.DB5)
48  %assign DB6 = CAST("String", SFcnParamSettings.DB6)
49  %assign DB7 = CAST("String", SFcnParamSettings.DB7)
50
51
52  ////////////////////////////////////////////////////
53  //Toggle enable
54  inline void Toggle_E(void)
55  {
56      PORTBbits.%<E> = 1; //Toggle_Enable
57      //__delay32(10);
58          asm("NOP");asm("NOP");asm("NOP");
59      PORTBbits.%<E> = 0;
60  }
```

OBR. 7.10: UKÁŽKA KÓDU – VLOŽENIE TEXTU DO LCD.C

Nakoniec je potrebné zadať do sekcie „start“ inicializáciu displeja a do sekcie „outputs“ funkcie na odoslanie reťazca na displej. Priradíme teda premenným S0 – S3 vstupné hodnoty bloku. Využijeme funkciu *LibBlockInputSignal* (viď obr. 7.11).

```
208 %% Function: Start ================================
209 %function Start(block, system) Output
210 LCD_Init(void)
211 %endfunction
212
213 %% Function: Outputs ==============================
214 %function Outputs(block, system) Output
215
216 %assign S0 = LibBlockInputSignal(0, "", "", 0)
217 %assign S1 = LibBlockInputSignal(1, "", "", 0)
218 %assign S2 = LibBlockInputSignal(2, "", "", 0)
219 %assign S3 = LibBlockInputSignal(3, "", "", 0)
220
221 %% set register
222 /* S-Function "dsPIC_PWM_OC_HW" Block: %<Name> */
223 {
224 %% posli string na display
225 sendStr((char *) %<S0>, 00);
226 sendStr((char *) %<S1>, 01);
227 sendStr((char *) %<S2>, 02);
228 sendStr((char *) %<S3>, 03);
229 }
```

OBR. 7.11: UKÁŽKA KÓDU – SEKCIA „START“ A „OUTPUTS“

7.2. PRIDANIE BLOKU DO KNIŽNICE

Posledným krokom pri tvorbe bloku je jeho uloženie do knižnice Simulinku. Tento krok je potrebný pre jednoduché použitie v ďalších modeloch.

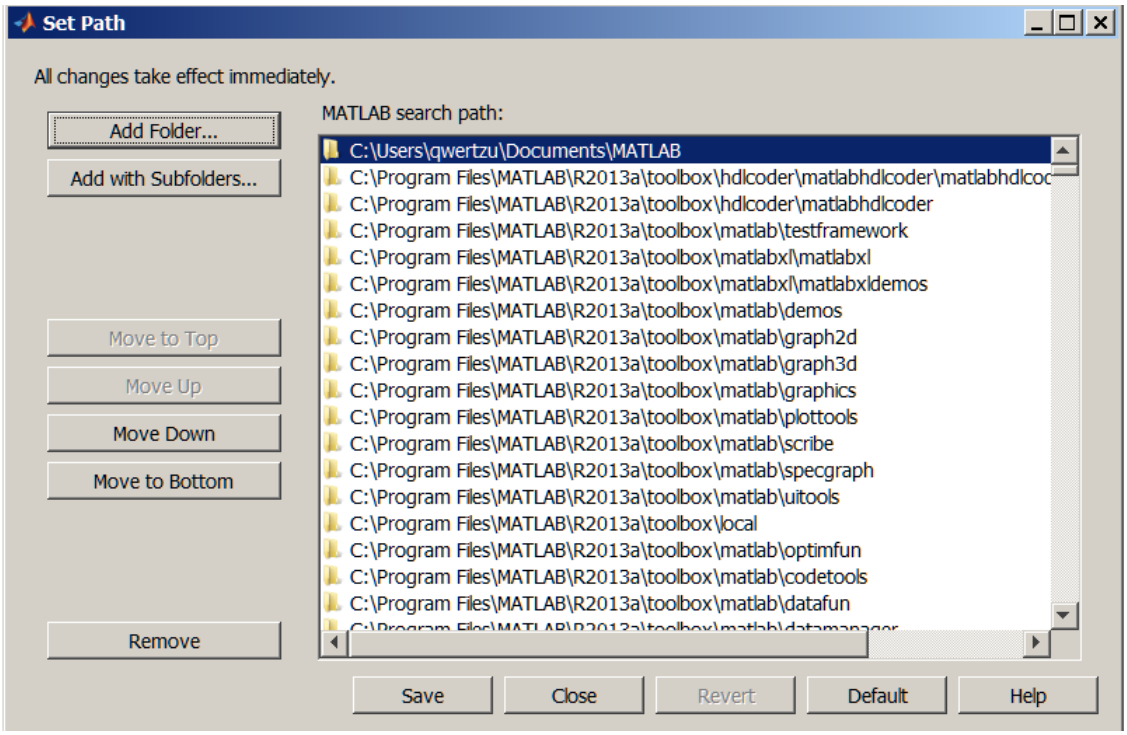
Postup pridania bloku do knižnice simulinku:

1) Vytvoríme priečinok so súbormi:
   - zdrojový kód bloku v tlc
   - súbory vygenerované z mex funkcie (t.j. zdrojový kód v C a súbor s príponou mexw64)
   - model, v ktorom je uložený blok
   - súbor *slblocks.m* (zatiaľ prázdny m-file)
2) Do súboru slblocks.m vložíme meno bloku a knižnice. Viď obr. 7.12.

```
Creating function blkStruct = slblocks
  Browser.Library = 'LCD_BLOK';     % meno modelu, v ktorom je blok ulozeny
  Browser.Name    = 'LCD_BLOK';     % nazov novej kniznice
  blkStruct.Browser = Browser;
```

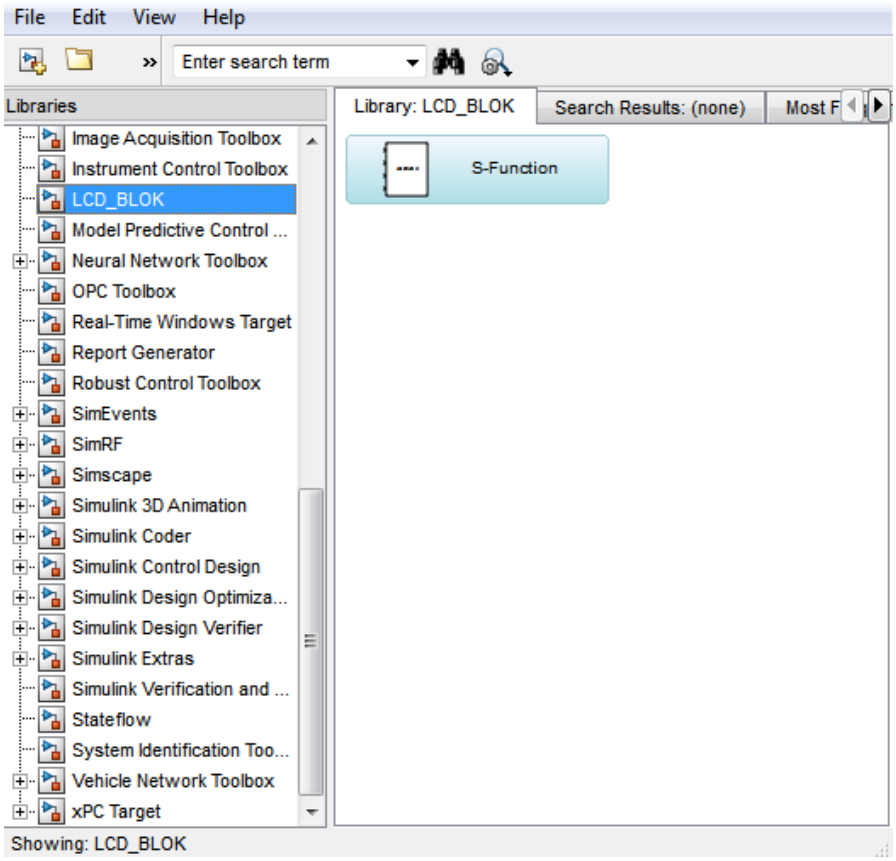
OBR. 7.12: UKÁŽKA KÓDU – FUNKCIA SLBLOCKS.M

3) Pomocou *pathtool* pridáme priečinok so súbormi z bodu č.1 do ciest v matlabe (obr. 7.13)



OBR. 7.13: MATLAB PATHTOOL

Nový blok nájdeme v knižnici Simulinku po reštartovaní matlabu (obr.7.14).



OBR. 7.14: KNIHOVNA LCD_BLOK

7.3. POUŽITIE BLOKU

Blok, ktorý týmito nastaveniami vznikol, je možné použiť spolu s Kerhuel toolbox. Ten vygeneruje súbor main.c, ktorý sa bude vykonávať po zapnutí mikrokontrolera.

Použitie bloku v spojení s Kerhuel toolbox-om je demonštrované na príklade. Model (obr. 7.15) obsahuje bloky:

Master – nastavenie mikrokontrolera

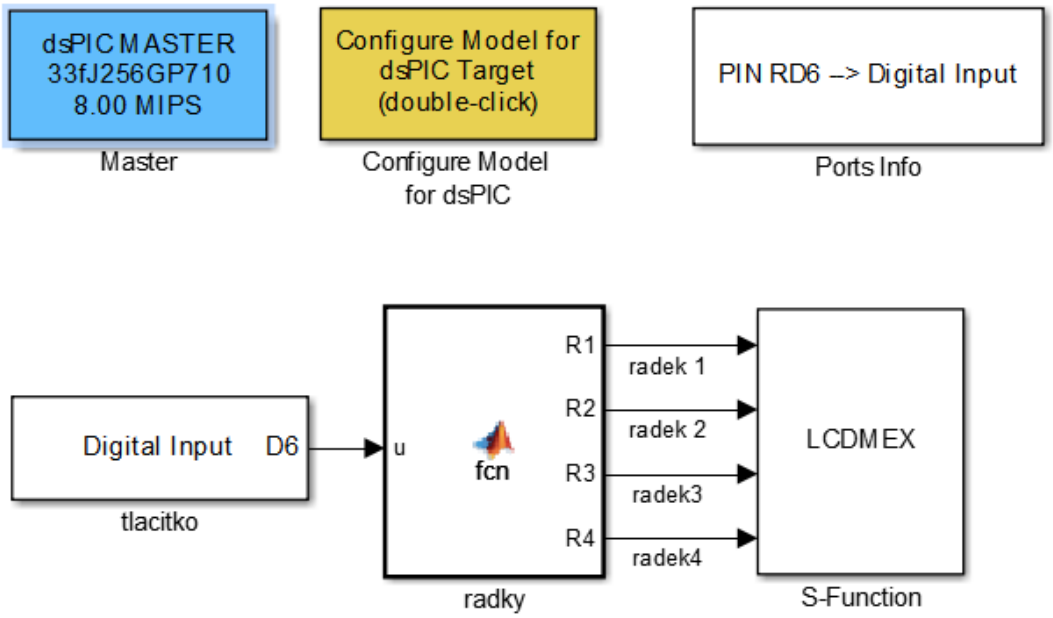
Configure Model for dsPIC

Ports Info

Digital Input – Digitálny vstup z tlačidla

Matlab fcn (obr. 7.16) – zapíše do vektorov R1 – R4 hodnoty v závislosti na vstupe z tlačidla. Počet znakov v každom vektore musí byť 20 (= počet znakov na jednom riadku). Poslednou hodnotou vektorov je 0, ktorá zabezpečí prechod na ďalší riadok. Výsledný rozmer vektorov je teda 21.

LCDMEX – blok na obsluhu displeja

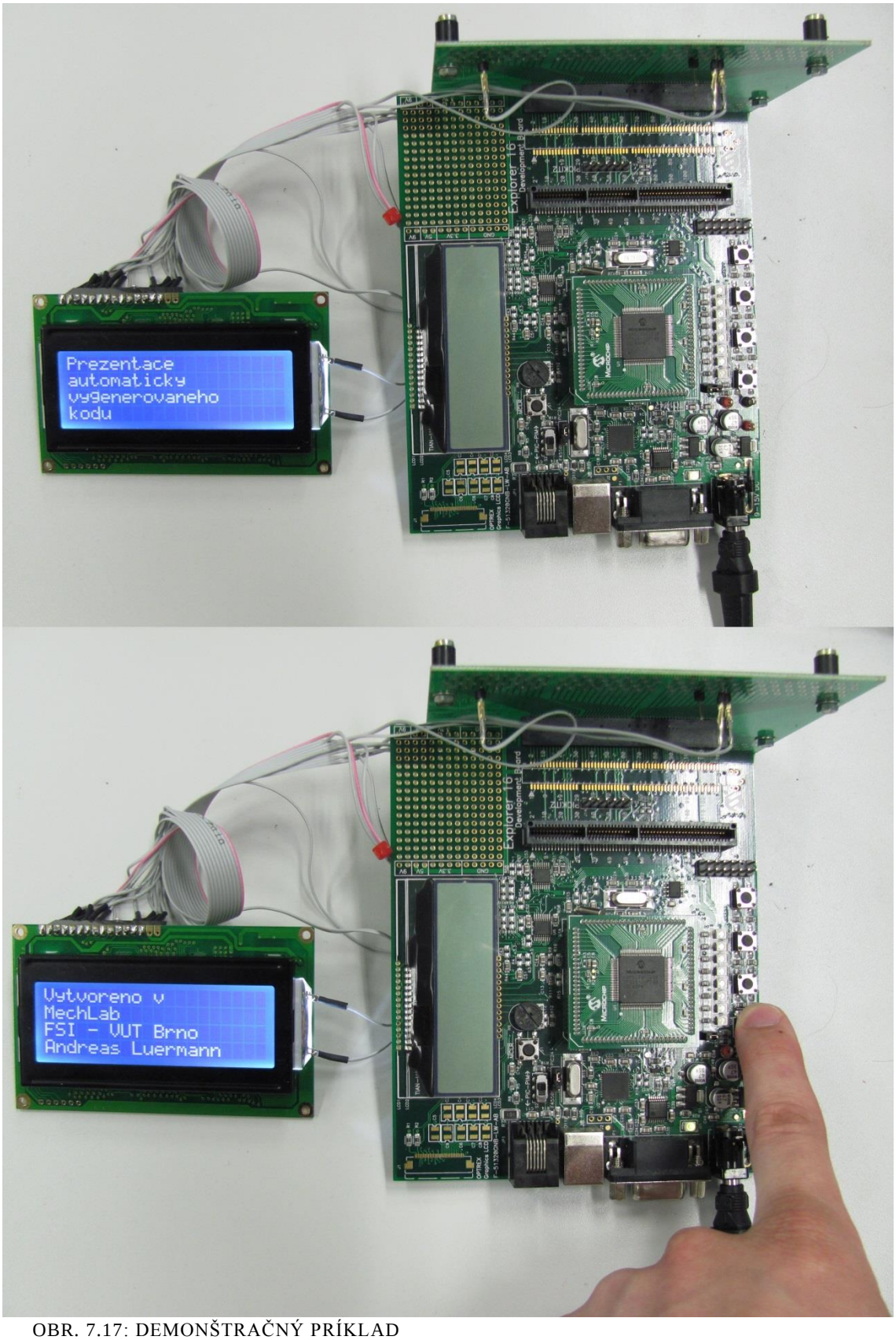


OBR. 7.15: MODEL DEMONŠTRAČNÉHO PRÍKLADU

```
function [R1, R2, R3, R4] = fcn(u)
if (u)
    R1 = [uint8('Prezentace          '),0]
    R2 = [uint8('automaticky         '),0]
    R3 = [uint8('vygenerovaneho      '),0]
    R4 = [uint8('kodu                '),0]
else
    R1 = [uint8('Vytvoreno v         '),0]
    R2 = [uint8('MechLab             '),0]
    R3 = [uint8('FSI - VUT Brno      '),0]
    R4 = [uint8('Andreas Luermann    '),0]
end
```

OBR. 7.16: MATLAB FCN – „RADKY“

Tento model vytvorí zdrojové súbory, ktoré je možné pomocou programu MPLab nahrať na mikrokontroler. Pre funkčnosť modelu viď obr. 7.17.



OBR. 7.17: DEMONŠTRAČNÝ PRÍKLAD

8. ZÁVER

Hlavným cieľom bakalárskej práce bolo vytvoriť blok v prostredí Simulink od firmy MathWorks, ktorý bude generovať software na obsluhu displeja. Tento cieľ bol v plnom rozsahu dosiahnutý. Blok taktiež spolupracuje s Kerhuel toolboxom, ktorý vygeneruje súbor main.c. Vďaka tomu je dosiahnutá vysoká efektivita návrhu, ako môžeme vidieť na demonštračnom príklade v kapitole 7.2. Použiteľnosť bloku zvyšuje aj jeho uloženie v knižnici matlabu, vďaka čomu je jednoducho vyhľadateľný.

Prínosom tejto bakalárskej práce je jednak blok v Simulinku, ktorý urýchli vývoj software a zjednoduší prácu s displejom. Zároveň môže byť táto práca použitá ako návod pri tvorbe blokov v prostredí Simulink (táto téma je podrobne rozobraná v kapitole 7.1.) a ich uloženie do knižnice (viď kapitola 7.3.). Okrem toho tu môžeme nájsť inštrukcie k tvorbe software pre mikro-kontrolery dsPIC od firmy Microchip.

Vytvorený blok môže byť ďalej upravený tak, aby bolo možné jednoducho adresovať potrebný text na požadované pole displeja. Takýto blok by bol doplnený o ďalšie parametre pomocou funkcie tlc.

9. ZOZNAM POUŽITÝCH ZDROJOV

[1] LAMBERSKÝ, V.: *Automatické generování kódu* [interné materiály]. Vysoké učení technické v Brně, Fakulta strojního inženýrství

[2] Microchip: *dsPIC33FJXXXGPX06/X08/X10 Data Sheet.*
URL <http://ww1.microchip.com/downloads/en/DeviceDoc/70286C.pdf>

[3] Microchip: *Explorer 16 Development Board User's Guide.*
URL <http://ww1.microchip.com/downloads/en/DeviceDoc/Explorer%2016%20User%20Guide%2051589a.pdf>

[4] GALLOWAY, J.: *HD44780 Controlled LCD* [online]. [cit. 2013-04-20]
URL <http://joshuagalloway.com/lcd.html>

[5] DOVEDA BOYS. *Znakové LCD displeje* [online]. [rev. 2007-02-25]. [cit. 2013-04-20].
URL <http://doveda.byl.cz/lcd/>

[6] LUBOSS17: *Znakový LCD displej s radičom HD44780* [online]. [cit. 2013-04-20]
URL <http://lubosweb.php5.sk/clanky/05_znakovy_displej.php>

[7] Hitachi: *HD44780 (LCD-II) data sheet*
URL <http://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780.html >

[8] HEROUT, P.: *Učebnice jazyka C – 1. Díl*. 6.vyd. České Budějovice: KOPP, 2011. 271 s. ISBN 978-80-7232-383-8

[9] HEROUT, P.: *Učebnice jazyka C – 2. Díl*. 4.vyd. České Budějovice: KOPP, 2012. 611 s. ISBN 978-80-7232-367-8

[10] Mathworks: Documentation center
URL <http://www.mathworks.com/help/>

10. ZOZNAM OBRÁZKOV A TABULIEK