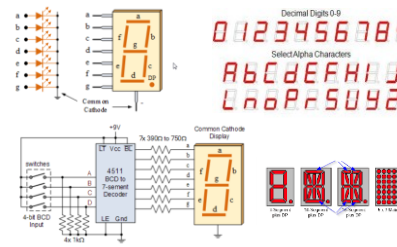
Mikropočítačové Systémy - MIPS

Prednáška 11

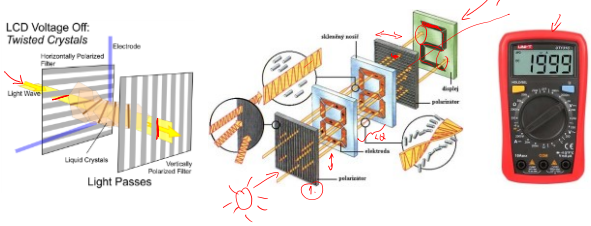
LCD displej

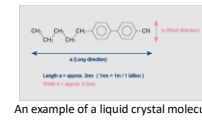
1

7-segmentový LED



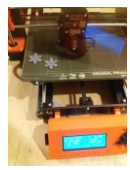
7-segmentový LCD

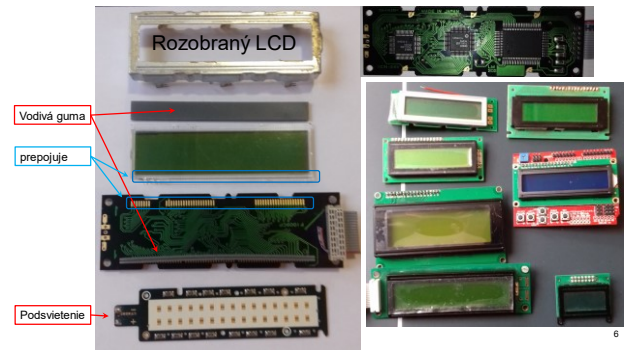




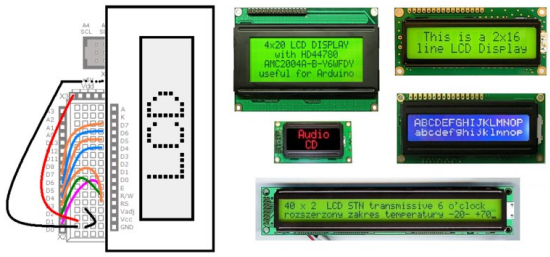
An example of a liquid crystal molecule

Aplikácie



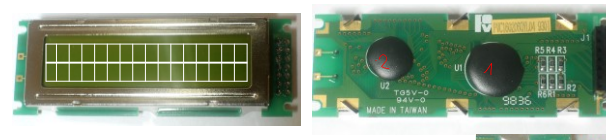


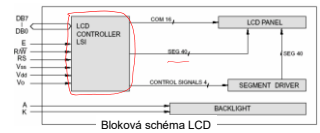
6



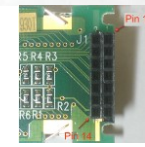
7

Alfanumerické LCD zobrazovače s radičom **HD44780**

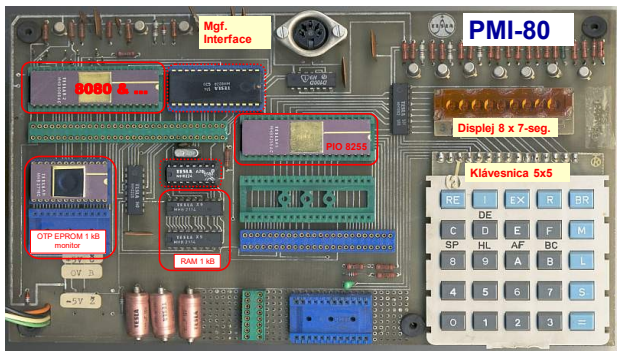




Bloková schéma LCD

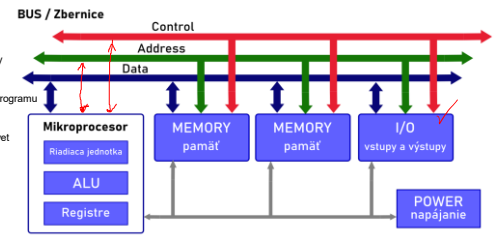


8

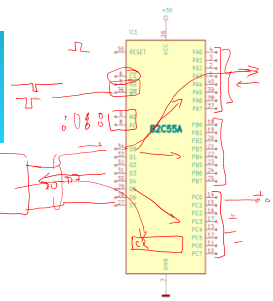


Štruktúra počítača

Procesor
riadi, realizuje výpočty
Pamäť
pamäť dát & pamäť programu
Input/Output
interface na okolitý svet
Napájanie

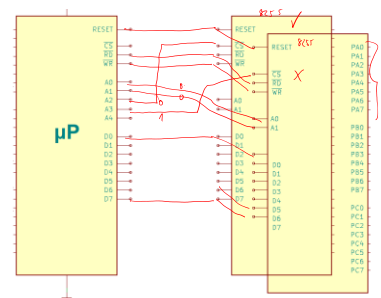


i8255

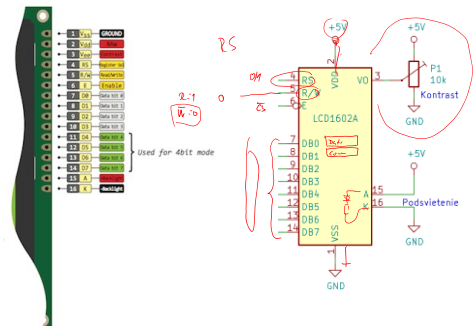


A_1	A_0	Port selected
0	0	port A
0	1	port B
1	0	port C
1	1	control register

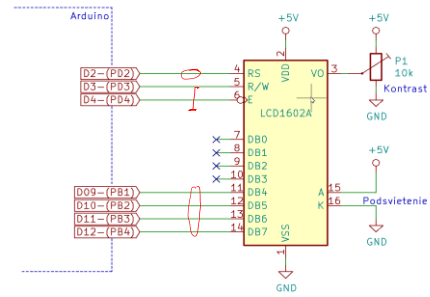
2x i8255



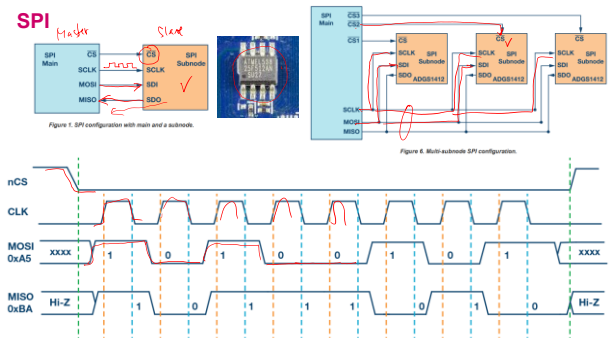
LCD



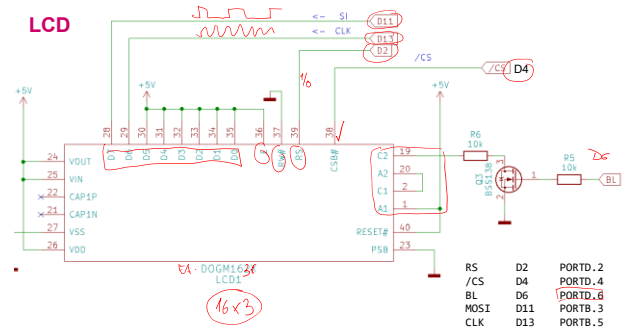
LCD



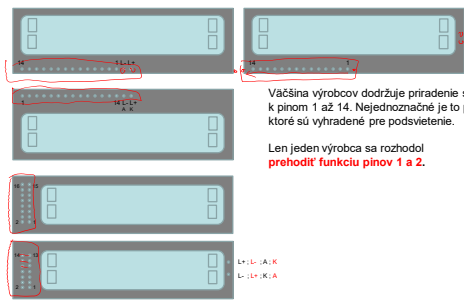
SPI



LCD



Niekoľko možností umiestnenia konektora a priradenie signálov k pinom:



Väčšina výrobcov dodržuje priradenie signálov k pinom 1 až 14. Nejednoznačné je to pre piny, ktoré sú vyhradené pre podsvietenie.

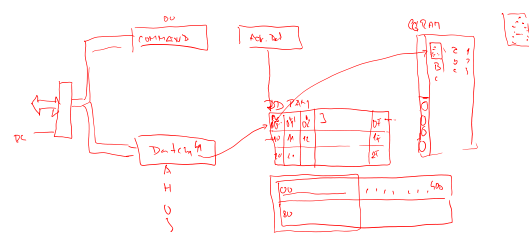
Len jeden výrobca sa rozhodol **prehodiť funkciu pinov 1 a 2**.

Číslo pinu	Symbol
1	Vss
2	Vcc
3	Vee
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

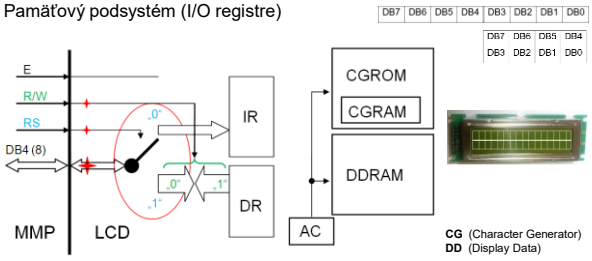
L+ ; L- ; A ; K
L- ; L+ ; K ; A

17

LCD



Pamäťový podsystém (I/O registre)



CG (Character Generator)
DD (Display Data)

IR (Instruction Register) – register inštrukcií.
DR (Data Register) – dátový register.
AC – adresný čítač.

AC

b7	b6	b5	b4	b3	b2	b1	b0
(BF)							

Riadiacu zbernicu - CB tvoria:

- EN (Enable). Výberový signál vo funkcii CS.
  - Signál je aktívny do „1"
  - Dobežná hrana vykoná príkaz definovaný pomocou: DB, RS, a RW
  - ale nemôžeme tvrdiť, že stačí vygenerovať, dobežnú hranu.
  - Ak bude High úroveň kratšia ako PW_{EH}, viď. KL, dobežná hrana nespôsobí nič.
- RS (Register Select).
  - RS = „0" ⇒ dáta sa spracujú ako inštrukcia LCD (napr.: zmaž LCD)
  - RS = „1" ⇒ dáta sa zobrazia na LCD ako znak
- RW(Read/Write ≡ „1"/„0").
  - RW = „0" informácia na DB sa zapíše do LCD
  - RW = „1"
    - Test stavu LCD (Get LCD Status)
    - „čítanie" z LCD

AC

b7	b6	b5	b4	b3	b2	b1	b0
(BF)							

DD RAM (Display Data)

Display Position

DDRAM Address (hexadecimal)	1	2	3	4	5	6	.....	14	15	16
	00	01	02	03	04	05	.....	0D	0E	0F
	10	11	12	13	14	15	.....	1D	1E	1F
	20	21	22	23	24	25	.....	2D	2E	2F

Fig. 11 3-Line Display

DDRAM (Display Data RAM).

Kapacita DDRAM je 80 znakov (5*7 bodov) :
• Kapacita pamäte je 80 B. Jeden byte je priradený k jednému znaku. Teoreticky máme 256 možností toho čo na danej pozícii zobrazíme.
• Nezobrazované znaky môžeme použiť ako pamäť RAM.
•Organizácia LCD panela môže byť : 1*8; 1*16 (akože) = (2*8); 1*20; 1*40
  • 2*16; 2*20; 2*40
  • 2*40
  • atď.

1	2	3		20	21			40
0x00	0x01	…	…	0x13	0x14			0x27
0x40	0x41	…	…	0x53	0x54			0x67

$0x27 \cong 39\ dekadicky$

0x00	0x01	…	…	0x13	0x14			0x27
0x40	0x41	…	…	0x53	0x54			0x67

0x00	0x01	…	…	0x13	0x14			0x27
0x40	0x41	…	…	0x53	0x54			0x67

Jedno-riadkové vs. Viac-riadkové displeje

N = 0, → Jednoriadkový display

Displej 1x8

1	2	3	4	5	6	7	8	Dek.
0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	Hex.

N = 1, → "Dvojriadkový" display. V opačnom prípade sa znaky na pozíciách 08H – 39H nezobrazia!

Displej 1x16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Dek.
0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	Hex.

INSTRUCTION			CODE								DESCRIPTION	Time (250 kHz)
	RS	R/W	db 7	db 6	db 5	db 4	db 3	db 2	db 1	db 0		
E=?												
Set the DD RAM address	0	0	1	MSB			ADD			LSB	DDRAM data is sent and received after this setting.	40 [us]

„1." riadok	0	0	0	0	0	0	0	Adresy: 0x00 až 0x3F
	0	1	1	1	1	1	1	
„2." riadok	1	0	0	0	0	0	0	Adresy: 0x40 až 0x7F
	1	1	1	1	1	1	1	

Jedno- a viac-riadkové displeje

Displej 2x16

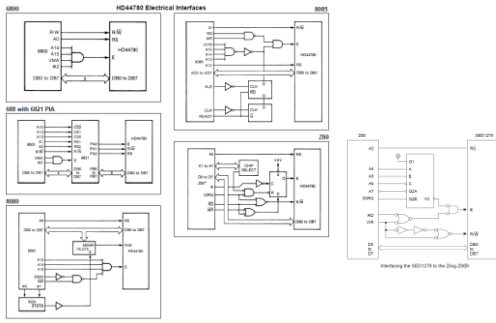
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Dek.
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	Hex.
0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F	Hex.

N = 1, → "Dvojriadkový" display
!!!! Pozor druhý riadok nezačína na „konci" prvého !!!

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Dek.
0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x10	Hex.
0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F	0x50	Hex.

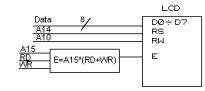
0x00	0x01	…	…	0x0F					0x27
0x40	0x41	…	…	0x4F					0x67

Možnosti pripojenia LCD k systémovej zbernici MMP: R/W ↔ /R, /W



25

Číslo pinu	Symbol	I/O	Funkcia
1	V_{SS}	-	0V napájanie
2	V_{DD}	-	+5V napájanie
3	V_{EE}	-	Kontrast
4	RS	I	0 = vstup je inštrukcia 1 = vstup sú dáta
5	R/W	I	0 = zápis dát do LCD 1 = čítanie dát z LCD
6	E	I	Aktivácia displeja
7	DB0	I/O	Dáta, bit 0
8	DB1	I/O	Dáta, bit 1
9	DB2	I/O	Dáta, bit 2
10	DB3	I/O	Dáta, bit 3
11	DB4	I/O	Dáta, bit 4
12	DB5	I/O	Dáta, bit 5
13	DB6	I/O	Dáta, bit 6
14	DB7	I/O	Dáta, bit 7



Čas trvania pulzu Enable: min. 450ns

Nech: Procesor rady 80C51 (80C552 má frekvenciu oscilátora $f_{osc} = 12MHz$.

Čas trvania signálu RD/WR je podľa KL

$t_{RD\ WR\ min.} = 6 * t_{OSC} - 100ns = 400\ ns$

Vyhovuje $f_{osc} = 11{,}0592MHz$.

a10 = 1/0 : R/W ,
a14 = 1/0 : data/inštrukcia ,
a15 = 1: (CS "displej")

a15	a14	a13	a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
1	RS	x	x	x	R/W	x	x	x	x	x	x	x	x	x	x

26

Prepojenie: MMP ⇔ Display. Priradenie pinov. Orientácia.

1.

b7	b6	b5	b4	b3	b2	b1	b0	PORT B
			D7	D6	D5	D4		Display
			D3	D2	D1	D0		DATA

write: `DDRB |= (1<<1)|(1<<2)|(1<<3)|(1<<4);  // Piny 1,2,3,4, PORTB ako output (Data pre display)`

read: `DDRB &= ~((1<<1)|(1<<2)|(1<<3)|(1<<4)); // Piny 1,2,3,4, PORTB ako input (Data pre MMP)`

2.

```
#define PORTB_DATA_H(x) PORTB &=0b11100001; PORTB |= (x & 0xF0 )>>3;
#define PORTB_DATA_L(x) PORTB &=0b11100001; PORTB |= (x & 0x0F )<<1;
```

D7	D6	D5	D4	D3	D2	D1	D0	⟨x⟩

$\hat{=}$ *Display DATA*

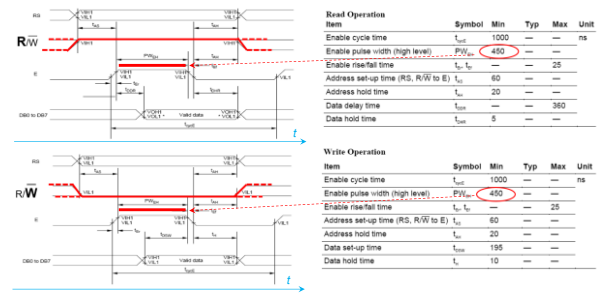
3.

b7	b6	b5	b4	b3	b2	b1	b0	PORT D
			E	R/W	RS			

```
DDRD |= (1<<En_pin);    // Pin D4 (Enable)  PORTD  output
DDRD |= (1<<RW_pin);    // Pin D3 (RW)      PORTD  output
DDRD |= (1<<RS_pin);    // Pin D2 (RS)      PORTD  output
```

27

Časovanie kontrolera HD44780U



28

Generovanie ENABLE impulzu: $f_{osc} = 16MHz \Rightarrow t_{SC} = 62{,}5\ ns$

```
#define NOP() asm("nop")  nop ≅ 1SC
#define LCD_DELAY   NOP();NOP();NOP();NOP();NOP();NOP();  LCD_DELAY ≅ 6 SC
```

$t_{EN\ High} = 6 * t_{SC} + 2 * t_{SC} = 500\ ns$

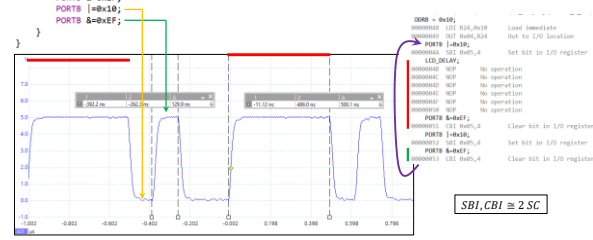
```
void En_imp(void) {
    PORTD |= (1<<En_pin);   // -> "log.1"
    LCD_DELAY;
    PORTD &= ~(1<<En_pin);  // -> "log.0"
}
```

```
void En_imp(void) {
    PORTD |= (1<<En_pin);
0000005A  SBI 0x08,4    Set bit in I/O register
    LCD_DELAY;
0000005B  NOP           No operation
0000005C  NOP           No operation
0000005D  NOP           No operation
0000005E  NOP           No operation
0000005F  NOP           No operation
00000060  NOP           No operation
    PORTD &= ~(1<<En_pin);
00000061  CBI 0x08,4    Clear bit in I/O register
}
```

$SBI, CBI \cong 2\ SC$

29

```
#define NOP() asm("nop")
#define LCD_DELAY   NOP();NOP();NOP();NOP();NOP();NOP();
#include <avr/io.h>
void main(void)
{   DDRB = 0x10;
    while(1){
        PORTB |=0x10;
        LCD_DELAY;
        PORTB &=0xEF;
        PORTB |=0x10;
        PORTB &=0xEF;
    }
}
```



$SBI, CBI \cong 2\ SC$

30

INSTRUCTION	CODE										DESCRIPTION	Time (250 kHz)
E=?	RS	R/W	db7	db6	db5	db4	db3	db2	db1	db0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter. Sets I/D=1.	1.64 [ms]
Return home	0	0	0	0	0	0	0	0	1	*	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.64 [ms]
Entry mode set	0	0	0	0	0	0	0	1	I/D	S=0	I/D = 0/1 (Decrement/ Increment) These operations are performed during data R/W of DDRAM/CGRAM	40 [us]
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	D= 0/1 (OFF/ON Display) C= 0/1 (OFF/ON Cursor) B= 0/1 (OFF/ON Blink Cursor)	40 [us]
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	*	*	S/C=0/1 Cursor/Display shift R/L=0/1 Left /Right	40 [us]
Function set	0	0	0	0	1	DL	N	F	*	*	DL= 0/1 (4/8 bits) N= 0/1 (1/2(1) lines) F= 0 (5*7 dots)	40 [us]
Set the CGRAM address	0	0	0	1	MSB	ACG				LSB	CGRAM data is sent and received after this setting.	40 [us]
Set the DDRAM address	0	0	1	MSB	ADD					LSB	DDRAM data is sent and received after this setting.	40 [us]
Read busy flag & address	0	1	BF	MSB	AC					LSB	Reads Busy flag & AC	40 [us]
Write data to CG or DD RAM	1	0	MSB							LSB	Writes data into DDRAM or CGRAM.	40 [us]
Read data from CG or DD RAM	1	1	MSB							LSB	Reads data from DDRAM or CGRAM.	40 [us]

410 clocks

I = cursor incr.
S = disp. incr,
10 clocks

?!? 0 [us] ?!?

31

```
//zapise data do Data Register
void lcd_data(unsigned char data){
    //while(busy_flag());
    while(rd_BF());
    PORTD |= (1<<RS_pin);       //    (RS = High)
    PORTD &= ~(1<<RW_pin);      //    (RW = Low, write)
    wr_data (data);
}

// zapise command do Instruction Register
void lcd_command(unsigned char command){
    //while(busy_flag());
    while(rd_BF());
    PORTD &= ~(1<<RS_pin);      //    (RS = Low)
    PORTD &= ~(1<<RW_pin);      //    (RW = Low, write)
    wr_data (command);
}

void wr_data (unsigned char data) {
    PORTB_DATA_H(data);         // data High nibble
    En_imp();

    PORTB_DATA_L(data);         // data Low nibble
    En_imp();
}
```

32

```
/* Funkcia zapise jeden byte do riadiaceho (Control) registra */
void lcd_command( char instruction )
{
clear_bit(DISPLAY_RS_PORT,DISPLAY_RS_PIN);
_delay_us( 1 );
lcd_write( instruction );
}


/* Funkcia zapise jeden byte do datoveho (Data) registra */
void lcd_data( char data )
{
set_bit(DISPLAY_RS_PORT,DISPLAY_RS_PIN);
_delay_us( 7 );
lcd_write( data );
}
```

33

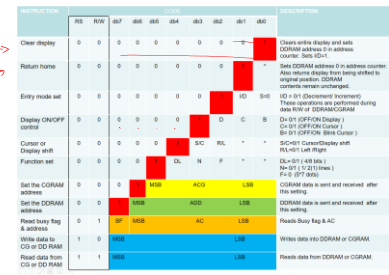
```
lcd_command(0x01);

lcd_command(0x02);

lcd_command(0b0000 1DCB);

lcd_command(0x80+0x40+3);

lcd_data(0x??);
```

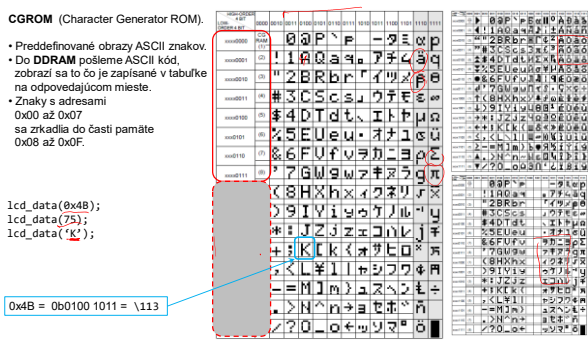


CGROM (Character Generator ROM).

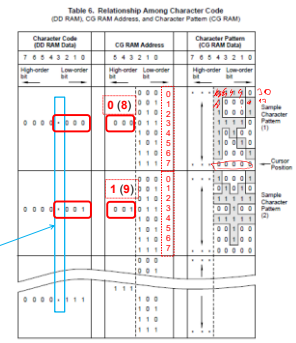
- Preddefinované obrazy ASCII znakov.
- Do **DDRAM** pošleme ASCII kód, zobrazí sa to čo je zapísané v tabuľke na odpovedajúcom mieste.
- Znaky s adresami
  0x00 až 0x07
  sa zrkadlia do časti pamäte
  0x08 až 0x0F.

```
lcd_data(0x4B);
lcd_data(75);
lcd_data('K');
```

0x4B = 0b0100 1011 = \113



35



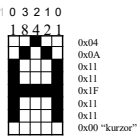
36

Preddefinované znaky ASCII

Znak (písmeno) *odpovedá* číselnému kódu.

Otom ako znak „vyzerá“ nám hovorí ***Font***.

Nazačiatku to vyzeralo asi takto:
Pr.:Font = 5*8(7+1)

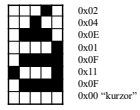


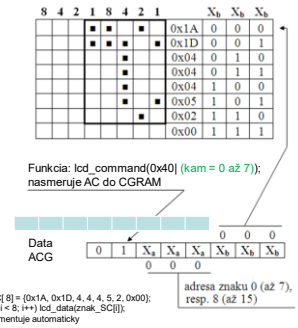
Znak: **A**

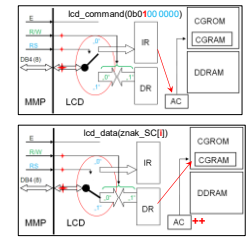
ASCII kód: 0x41

Znak: **á**

ASCII kód: 0x??







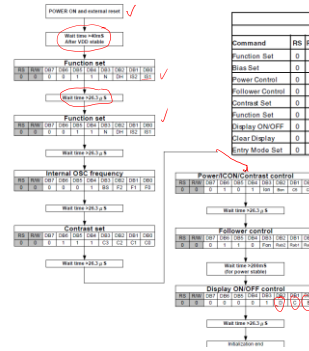

```
char znak_SC[ 8] = {0x1A, 0x1D, 4, 4, 4, 5, 2, 0x00};
for(char i = 0; i < 8; i++) lcd_data(znak_SC[i]);
// AC sa inkrementuje automaticky
```

Nesmiem zabudnúť na pôvodnú hodnotu AC.
Pôvodne AC ukazoval niekam do DDRAM.

I/D = 1: increment, I/D = 0: Decrement S = 1: Display Shift On S/C = 1: Shift Display, S/C = 0: Move Cursor R/L = 1: Shift Right, R/L = 0: Shift Left DL = 1: 8-Bit, DL = 0: 4-Bit N = 1: Dual Line, N = 0: Single Line BF = 1: Internal Operation, BF = 0: Ready for instruction	DD RAM: Display Data RAM CG RAM: Character Generator RAM A_{CG}: Character Generator RAM Address A_{DD}: Display Data RAM Address AC: Address Counter

Inicializácia

- Pred prvým použitím treba LCD inicializovať a konfigurovať.
- Po pripojení napájania, ak je +5V potom, sa radič nastaví do základného módu .
  - Jeden riadok, 8 znakov v riadku, 4-bitové pripojenie, ...
- Ak sa nevykoná, hardwarový RST a inicializácia, alebo nám nevyhovuje nastavenie, treba vykonať softwarovú inicializáciu.



Example of initialisation: 8 bit and SPI 5V

EA DOGM163

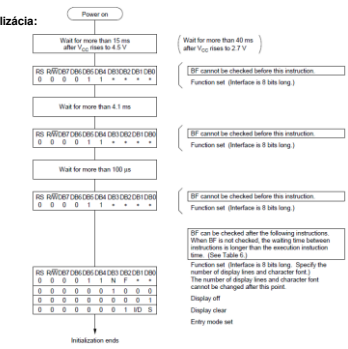
Command	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Hex	Note
Function Set	0	0	0	0	1	1	1	0	0	1	$39	8 bit data length, 2 lines, instruction table
Bias Set	0	0	0	0	0	1	1	1	0	1	$1D	BS: 1/4, 3 line LCD
Power Control	0	0	0	0	1	0	1	0	0	0	$50	booster off, contrast C5, set C4
Follower Control	0	0	0	0	1	1	0	1	1	0	$6C	set voltage follower and gain
Contrast Set	0	0	0	1	1	1	1	1	0	0	$7C	set contrast C3, C2, C1
Function Set	0	0	0	0	1	1	1	0	0	0	$38	switch back to instruction table 0 -
Display ON/OFF	0	0	0	0	0	0	1	1	1	1	$0F	display on, cursor on, cursor blink
Clear Display	0	0	0	0	0	0	0	0	0	1	$01	delete display, cursor at home
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	$06	cursor auto-increment

LCD inicializácia

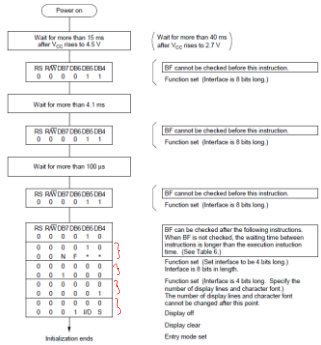
Reset vyvolaný po nábehu napájania:

1. Clear **Display (Kód inštr. = 0x01)**
   Kurzor: Ľavý horný roh. AC = 0x00. I/D = 1 (inkrement)
2. **Function Set (Kód inštr.= 0x30** (0x20|0x10))
   DL = 1 ... DB je 8-bitová.
   N = 0 ... Jednoriadkový display
   F = 0 .... Font 5x7
3. **Display ON/OFF Control (Kód inštr.= 0x08)**
   D = 0 ... Displej vypnutý.
   C = 0 ... Kurzor vypnutý.
   B = 0 .... Blikanie kurzora vypnuté.
4. **Entry Mode Set (Kód inštr.= 0x06** (0x04|0x2))
   I/D = 1 ... Inkrementovanie.
   S = 0 ... Posúvanie displeja vypnuté.

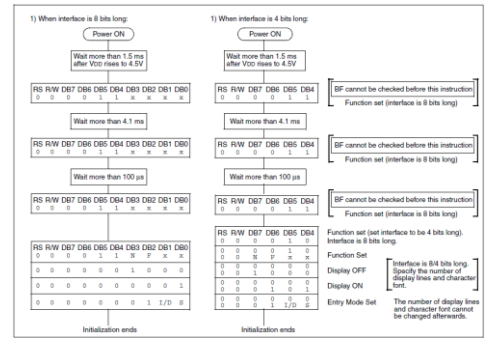
Softvérová inicializácia:
8 - bit Interface



Softvérová inicializácia:
4 - bit Interface



43



44

Správne vykonávanie operácií

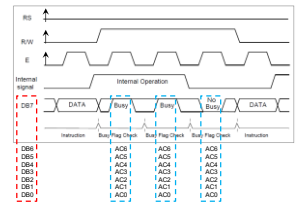
Čas trvania inštrukcií t_v je premenlivý, viď. tab. inštrukcie display-a.
T_v= f($f_{OSC\ disp}$, typ instr.); $f_{OSC\ disp}$ =270, 250kHz ±30%

Počas spracovávania príkazu, LCD ďalší príkaz neakceptuje.
Máme dve možnosti:

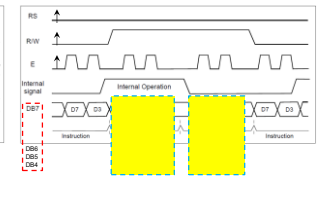
- „Počkáme“
- Testujeme stav signálu BUSY (bitová premenná)
  - Ak je LCD BUSY, potom je DB7 = „1“, ak je predchádzajúca operácia ukončená DB7 = „0“

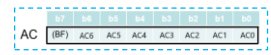
45

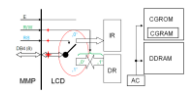
Časovanie:8-b DB



Časovanie:4-b DB

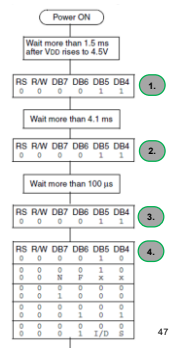




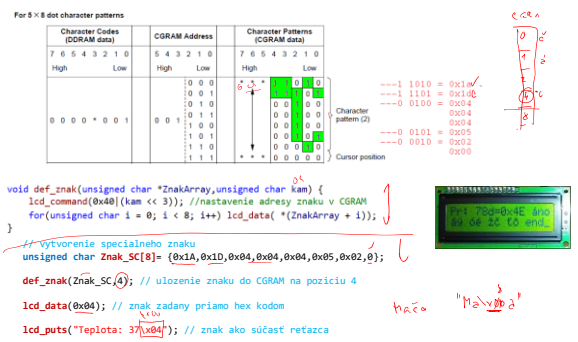


46

```
char lcdInit4(void)
{  // 4 bitove pripojenie display-a
   _delay_ms(15);
   // 1. ------------------
   PORTD &= ~(1 << RS_pin);       // set RS  - to "log. 0"  Instruction Register (IR)
   PORTD &= ~(1 << RW_pin) ;      // set R/W - to "log. 0" - Write
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   1   1     x   x   x   x     = 0x30
   PORTB_DATA_H(0x30); // 8-bitove pripojenie
   En_imp();
   // 2. ------------------
   // zopakujem  8-bitove pripojenie
   _delay_ms(5); En_imp();
   // 3. ------------------
   // zopakujem  8-bitove pripojenie
   _delay_ms(1); En_imp(); // -  stacilo by delay 0.1ms, momentalne najkratsie nastavitelne je 1ms
   // 4. ------------------
   // zmenie na 4-bitove pripojenie
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   1   0     x   x   x   x     = 0x20
   PORTB_DATA_H(0x20); // 4-bitove pripojenie
   _delay_ms(1); En_imp(); // -  stacilo by delay 0.04ms
   // ------------------
   // LCD function set     mod: DL = 0 - 4-bitove data, N = 1 - 2 riadky, F = 0 - 5x7 dots
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   1   DL    N   F   x   x     = 0x28
   _delay_ms(1); wr_IR(0x28);
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   0   0     0   0   0   1     = 0x01 , Display clear
   _delay_ms(2); wr_IR(0x01);
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   0   0     0   1   I/D S     = 0x06, I/D = 1 - inkrement
   _delay_ms(2); wr_IR(0x02);
   // RS R/W   DB7 DB6 DB5 DB4   DB3 DB2 DB1 DB0
   // 0  0     0   0   0   0     1   D   C   B     = 0x0E, D = C = 1 - Display a kurzor zapnute
   _delay_ms(1); wr_IR(0x0E);                      // B = 0 - blikanie vypnute
}
```



47



48

```
void Vypis_na_display(void){
    // zapíšeme niekolko znakov do 1 riadku
    lcd_data(0x50); // P
    lcd_data('r');      // r
    lcd_data(':');      // :
    // namiesto príkazu goto riadok, pozicia
    // len nastavíme kurzor na 1. riadok 4. pozicia (5-ta)
    lcd_command(0x80 + 4);// 0b1000 0000 + 0b0000 0100
    // a pokračujeme vo výpise
    // "formátovaný výpis" pomocou printf
    // číslo CIS vypíšeme dekadicky a hexadecimálne
    char CIS = 78;  // hexadecimálne 0X4E a pridáme text áno
    // ak by sme zapísali adresu znaku á, teda \000 další text by sa neobjavil
    // nula je koniec retazca
    // sprintf je formatovany vypis do retazca
    sprintf(Riadok,"%d=0x%X \010no ", CIS,CIS );
    zob_text(Riadok);
    // len nastavíme kurzor na 2. riadok 1. pozicia (resp. 0)
    lcd_command(0xC0 + 0);  // 0b1100 0000
    // a vypíšeme špeciálne znaky z CGRAM a pridame text end
    sprintf(Riadok,"\010\1 \2\3 \4\5 \6\7 e\156d" );
    zob_text(Riadok);
}
void zob_text(char *s){
    register unsigned char c;
    while((c = *s++))   lcd_data(c); // retazec konci "nulou"
}
```

49

```
#include <avr/io.h>
#include "lcd.h"

int main(void)
{
    lcd_init();                 // inicializacia portov a displeja
    lcd_data('A');              // zobraz pismeno A na aktualnej pozicii
    lcd_data(104);              // zobraz pismeno 'h' = 104dec = 0x68 ASCII

    lcd_command(0xc0);
    lcd_command(0x80 + 4);

                // 01234567890123456 +1 na koniec retazca
 char riadok[]= {"                "};
 int value = 42;                // nejaka premenna

 sprintf(riadok,"Hodnota = %d ",value); // Zobrazíme ‘value’ desiatkovo %d
 lcd_puts(riadok);

  while(1); /* stop here */
}
```