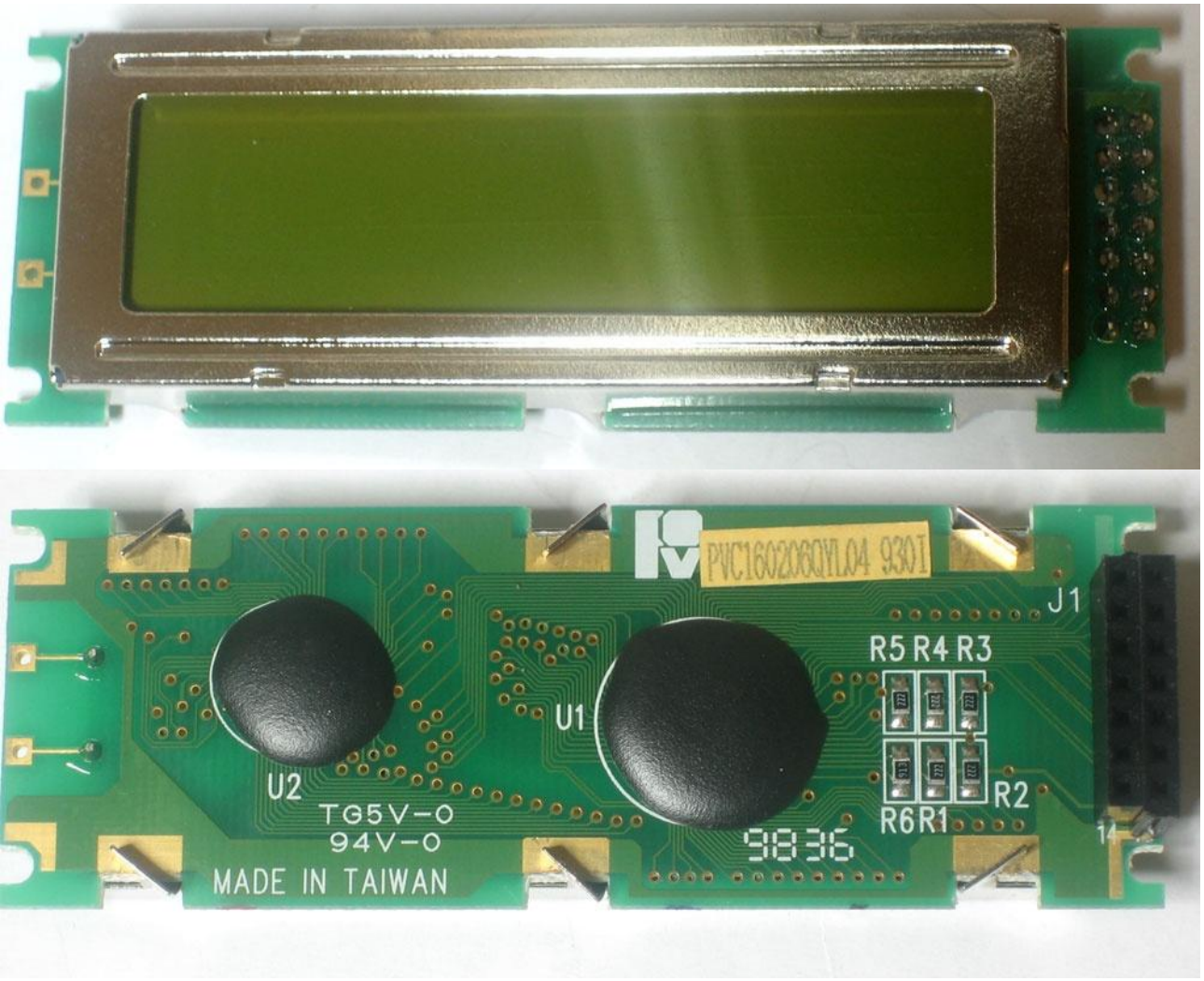
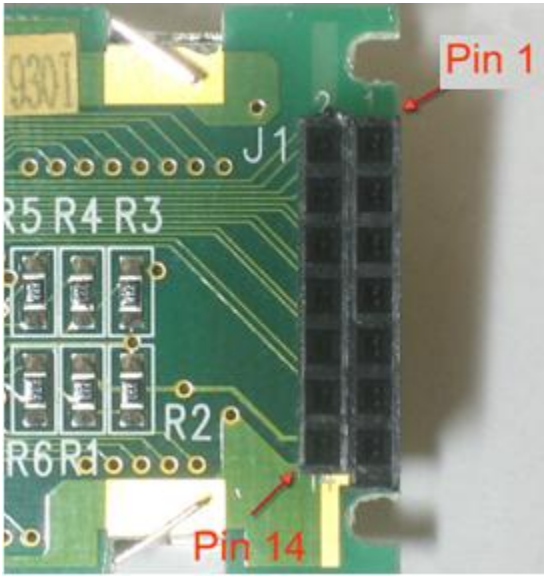
Alfanumerické LCD zobrazovače





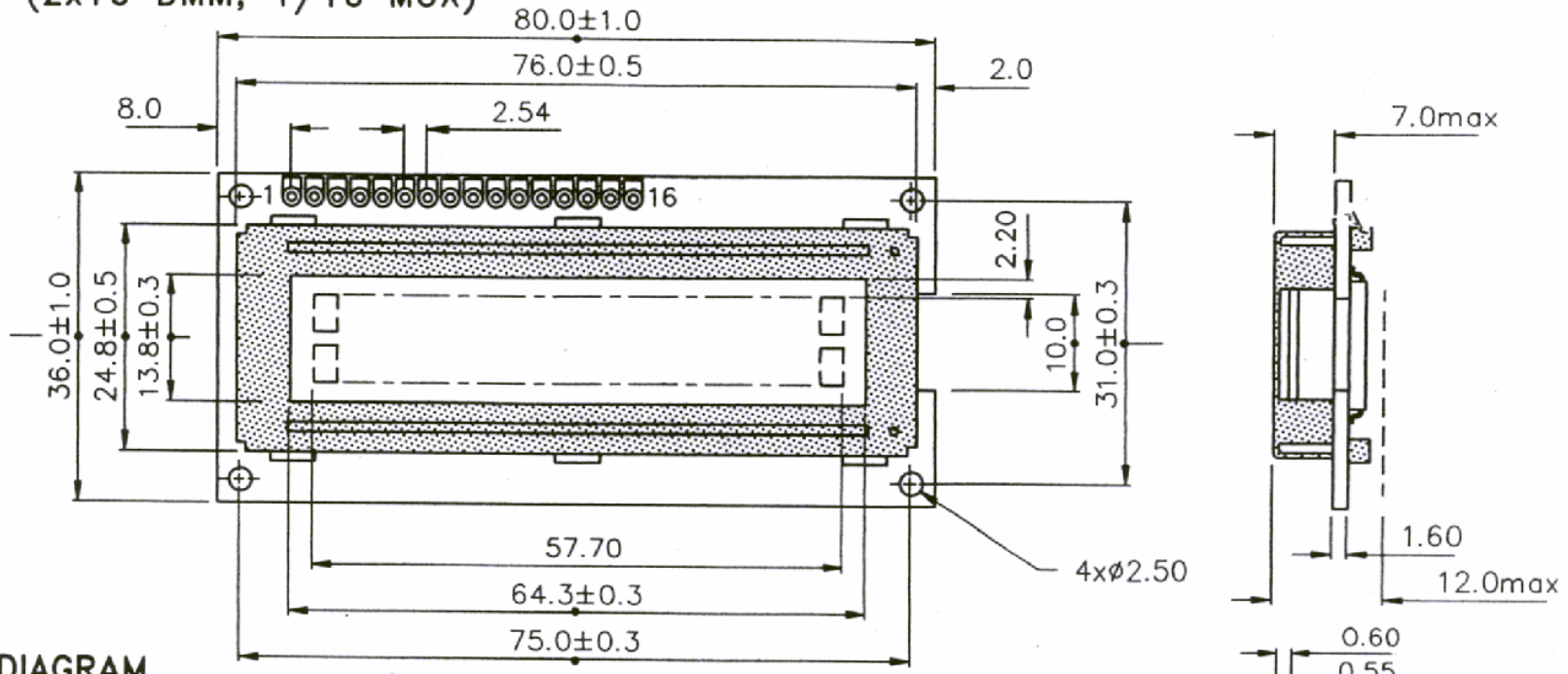
ALPHANUMERIC DOT MATRIX MODULES WITH BUILT-IN LED BACK LIGHTS

MODULE DIMENSIONS

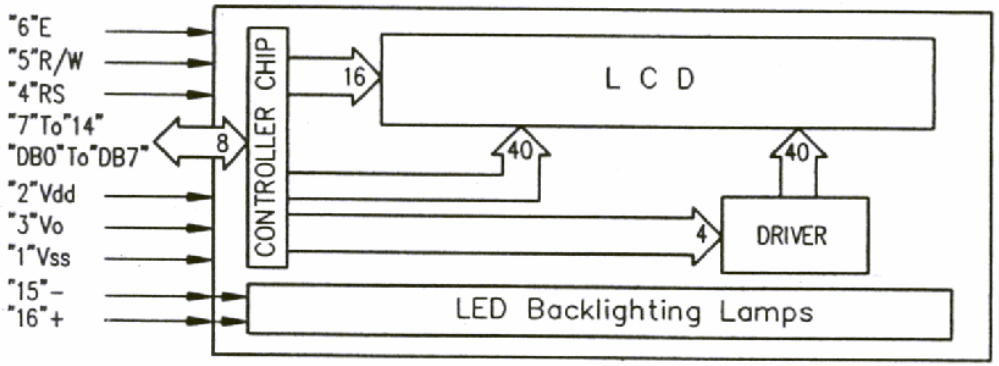
VK2123 (2x16 DMM, 1/16 MUX)

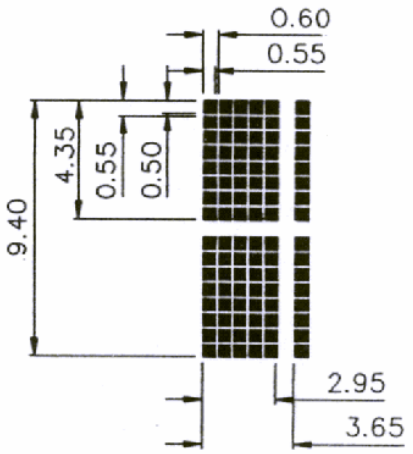
DIMENSION IN MM

"L"



BLOCK DIAGRAM

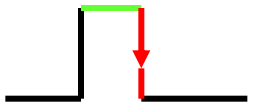




- LCD je pre MMP výstupné zariadenie
- Vyrábajú sa 16(znakov)x2(riadky), 20x2, …
- Na strane LCD je zabudovaný „driver“ 44780
  - Štandard v LCD displajoch
  - Umožňuje pripojenie pomocou:
    - 8-(4) bitovej DB
    - 3-bitovej CB

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

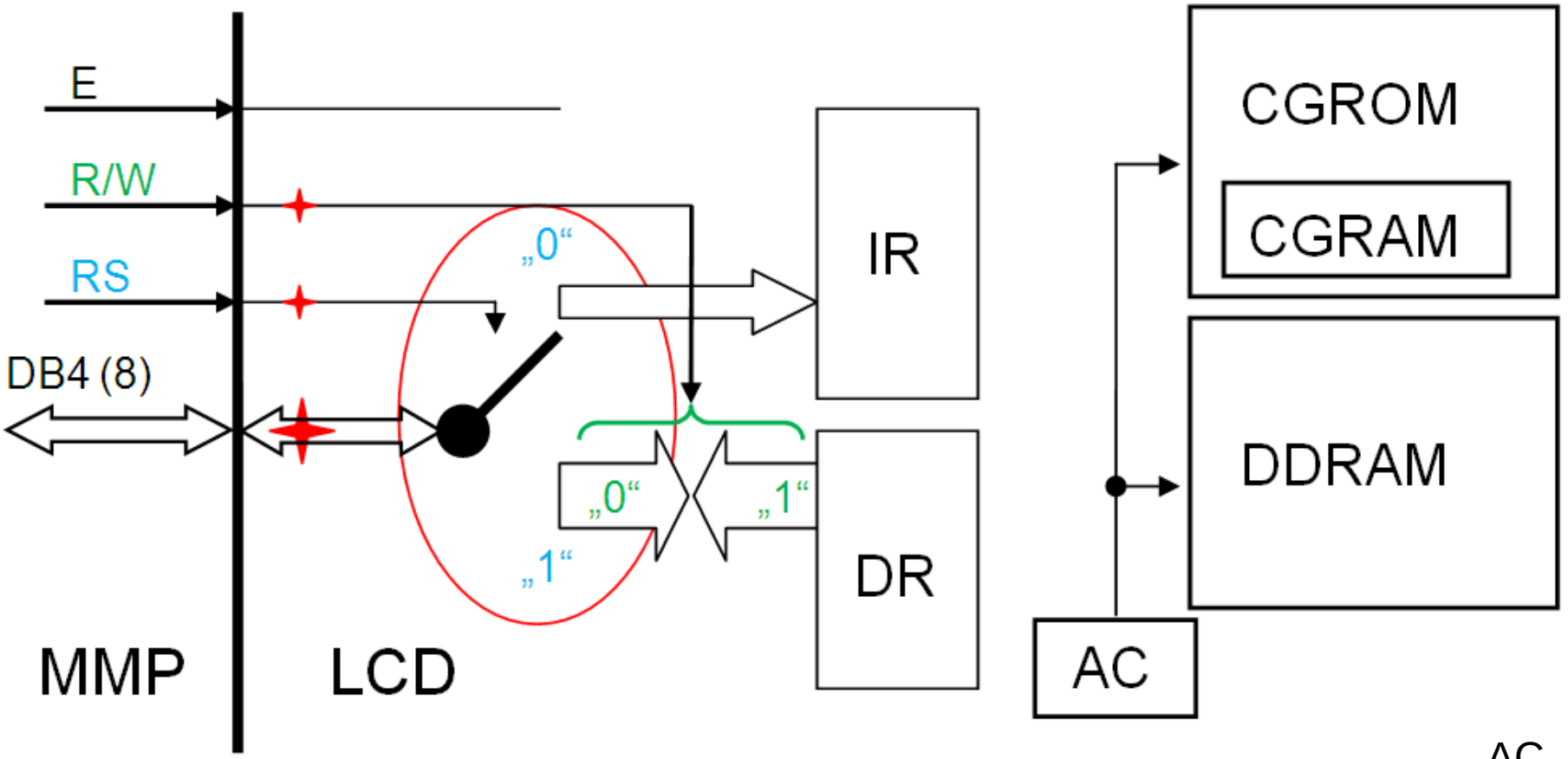


- RS (Register Select).
  - RS = „0“ ⇒ dáta sa spracujú ako inštrukcia LCD (napr.: zmaž LCD)
  - RS = „1“ ⇒ dáta sa zobrazia na LCD ako znak
- RW(Read/Write ≡ „1“/„0“).
  - RW = „0“ informácia na DB sa zapíše do LCD
  - RW = „1“
    - Test stavu LCD (Get LCD Status)
    - „čítanie“ z LCD

Pripojenie LCD k MMP

- Pomocou systémovej zberníce: DB,CB a AB
  - Aj keď „44780“ sa objavil v čase „8080“ nedá sa priamo pripojiť na zbernice DB, CB a AB
  - Pre CB procesora 8080 je typické
    - /RD – aktívny do nuly
    - /WR – aktívny do nuly
  - atď.
- Pomocou obojsmerných portov:
  - 7 bitov = 4b (dátové) + 3b(riadiace)
  - 8 + 3 bitov = 8b (dátové) + 3b(riadiace)

Pamäťový podsystém
(I/O registre)



AC

b7	b6	b5	b4	b3	b2	b1	b0
(BF)							

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

DDRAM (Display Data RAM).

Kapacita DDRAM je 80 znakov :

- Kapacita pamäte je 80 B.
- 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.

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

CGROM (Character Generator ROM).

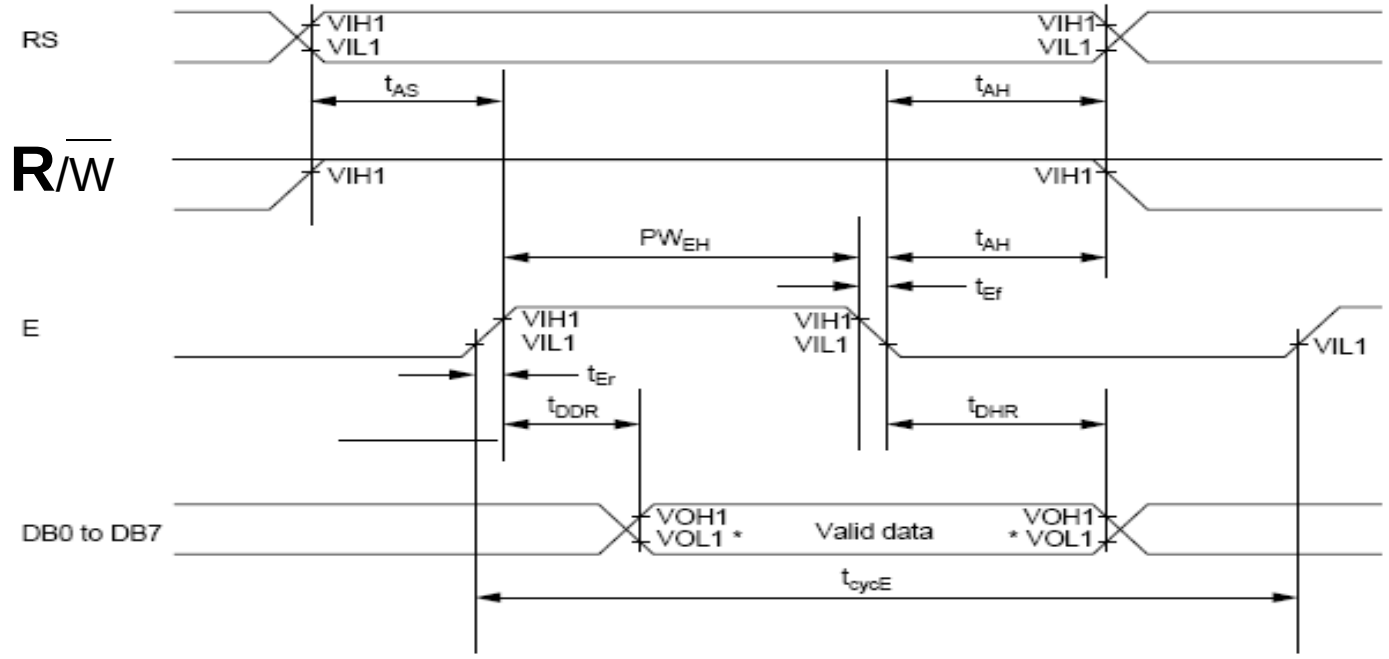
- Preddefinované obrazy ASCII znakov
- Do **DDRAM** pošleme ASCII kód, zobrazí sa to čo je zapísané v tabuľke na odpovedajúcom mieste.

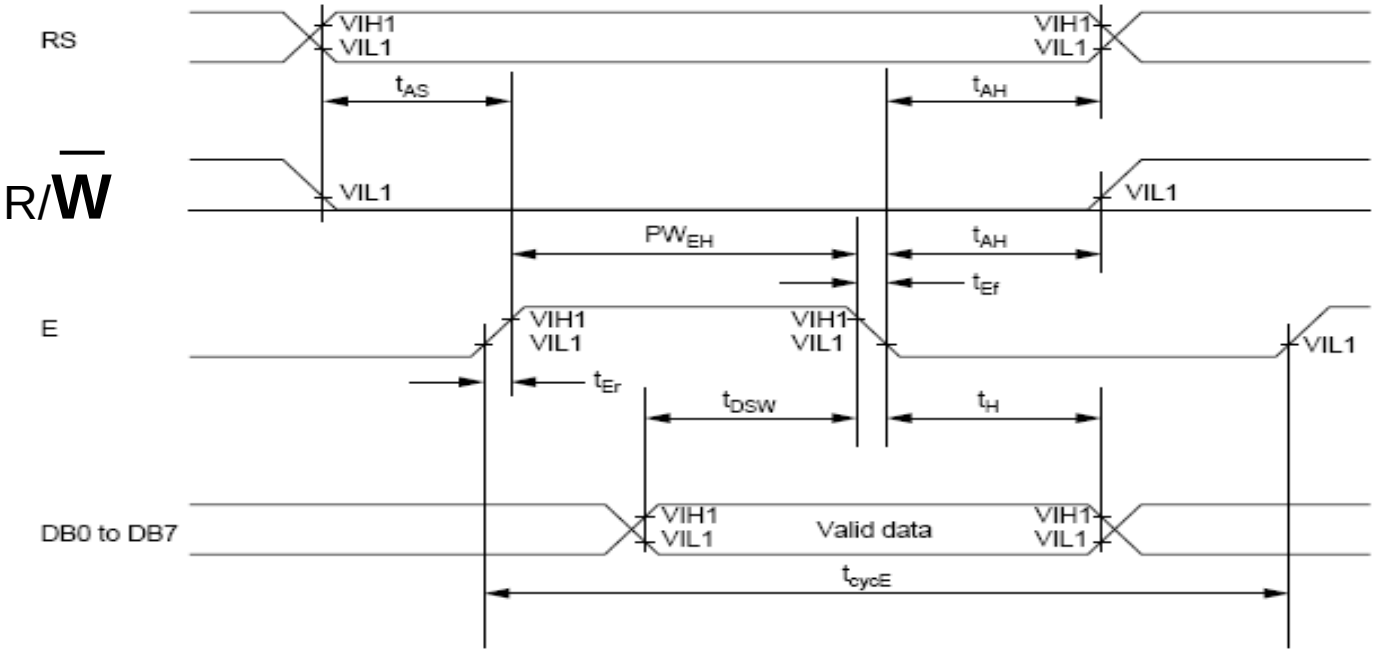
CGRAM (Character Generator RAM – 8 znakov)

- Časť pamäte znakov (vzorov) - môžeme naprogramovať sami.
- CGRAM - ASCII kódy 0x00 až 0x0F.
- Znaky s adresami 0x00 až 0x07 sa zrkadlia do časti pamäte 0x08 až 0x0F.

```
const unsigned char Znak[][8]={
  {0x02,0x04,0x0e,0x01,0x0f,0x11,0x0f,0},    // á,        0
  {0x02,0x04,0x11,0x11,0x0f,0x01,0x0e,0},    // ý,        1
   {0x0a,0x04,0x1f,0x02,0x04,0x08,0x1f,0},   // ž,        2
  {0x0a,0x04,0x0e,0x10,0x10,0x11,0x0e,0},    // č,        3
  {0x0a,0x0a,0x1c,0x08,0x08,0x09,0x06,0},    // ť,        4
  {0x04,0x0a,0x0e,0x11,0x11,0x11,0x0e,0},    // ô,        5
  {0x02,0x04,0x0c,0x04,0x04,0x04,0x0e,0},    // í,        6
  {0x0a,0x04,0x0e,0x10,0x0e,0x01,0x1e,0},    // š,         7
  {0x0a,0x00,0x0e,0x01,0x0F,0x11,0x0f,0},    // ä,        8
  {0x0A,0x04,0x0e,0x11,0x1f,0x10,0x0e,0},    // ě ,       9
  {0x0A,0x04,0x16,0x19,0x10,0x10,0x10,0},    // ř ,       10
  {0x0A,0x00,0x0e,0x11,0x11,0x11,0x0e,0},    // ö,        11
  {0x0A,0x0a,0x0e,0x11,0x11,0x11,0x0e,0},    // ő,        12
  {0x06,0x0c,0x11,0x11,0x11,0x13,0x0d,0},     // ű,       13
  {0x0A,0x04,0x0A,0x11,0x1F,0x11,0x11,0},    // Ä  krátke čiarky
 };
```

Časovanie Kontroler HD44780U





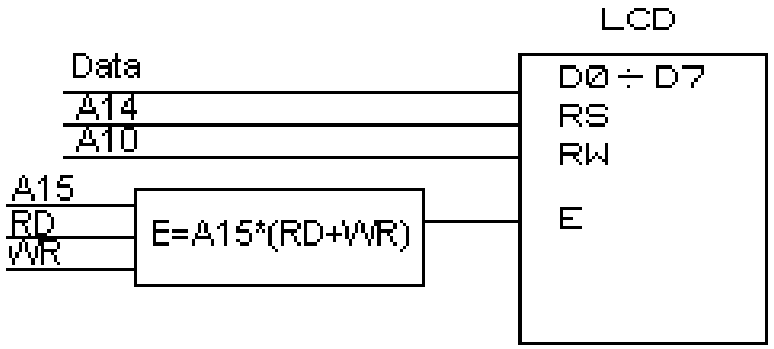
Read Operation

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	1000	—	—	ns
Enable pulse width (high level)	PW_{EH}	450	—	—	
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	25	
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	60	—	—	
Address hold time	t_{AH}	20	—	—	
Data delay time	t_{DDR}	—	—	360	
Data hold time	t_{DHR}	5	—	—	

Write Operation

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	1000	—	—	ns
Enable pulse width (high level)	PW_{EH}	450	—	—	
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	25	
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	60	—	—	
Address hold time	t_{AH}	20	—	—	
Data set-up time	t_{DSW}	195	—	—	
Data hold time	t_{H}	10	—	—	

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

čas trvania RD/WR je $6t_{ck}$

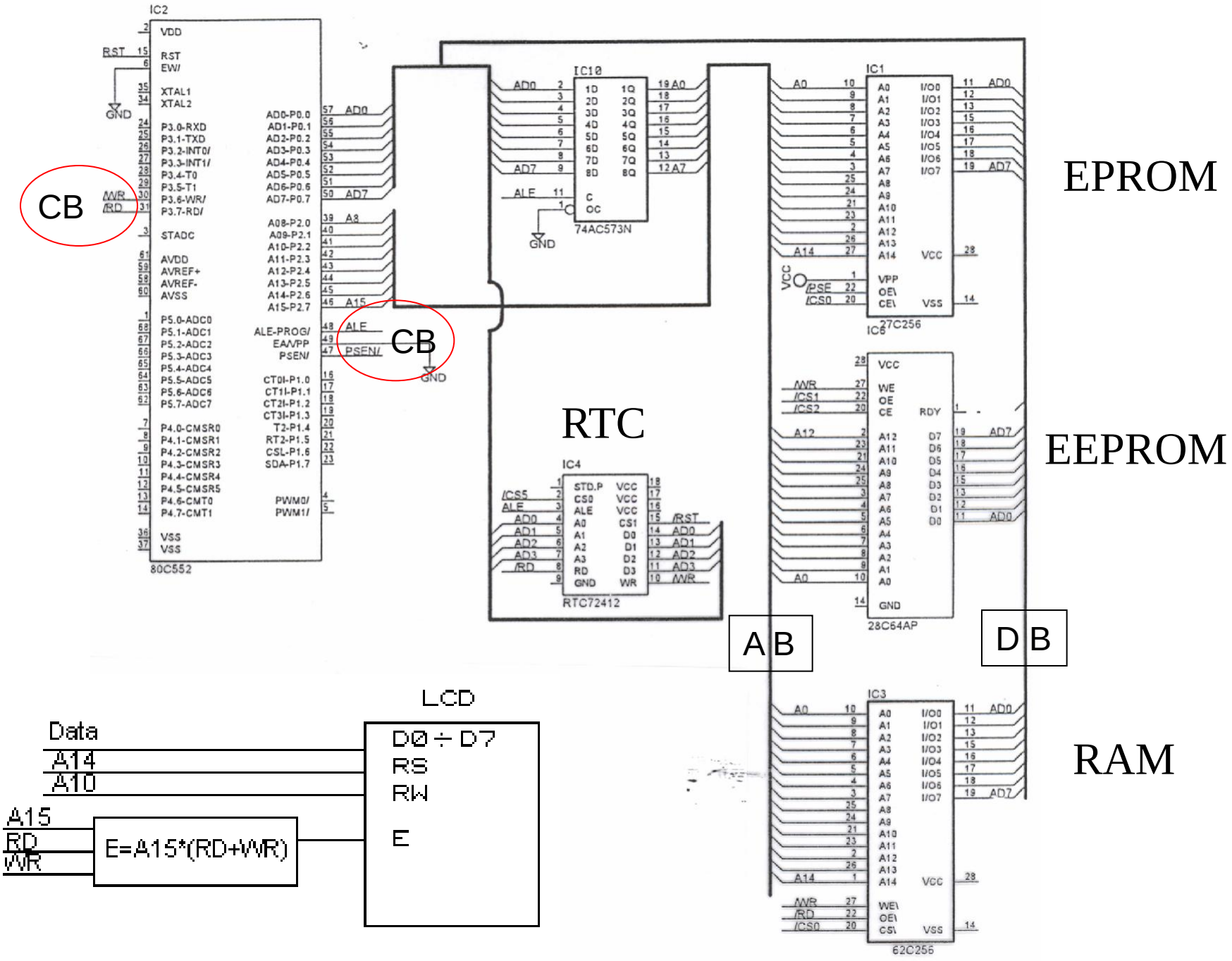
t.j. pre f_{osc}= 12MHz =>

t_{ck} = 1/12MHz = 0.08333us

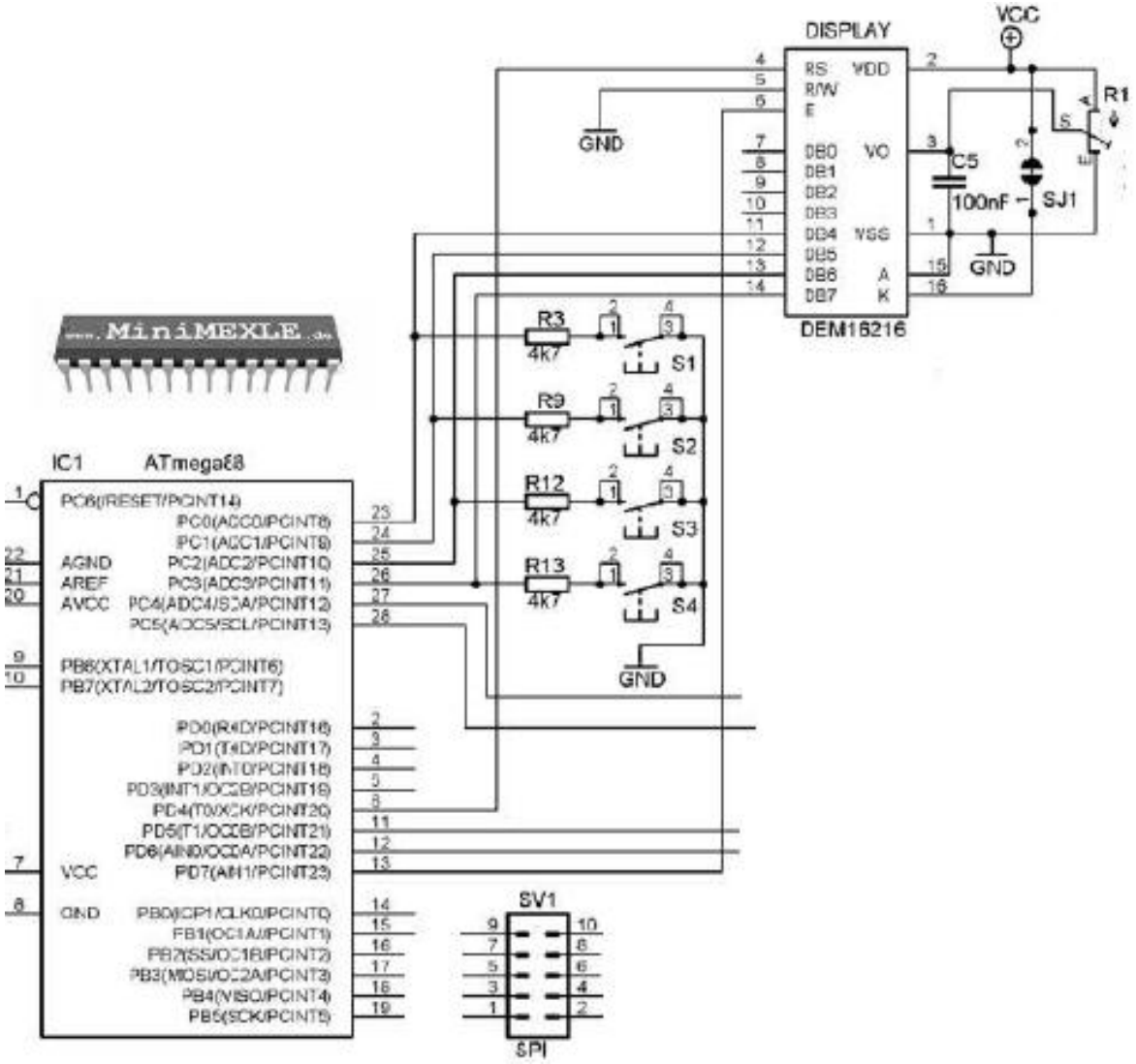
$6t_{ck}$ = 500ns

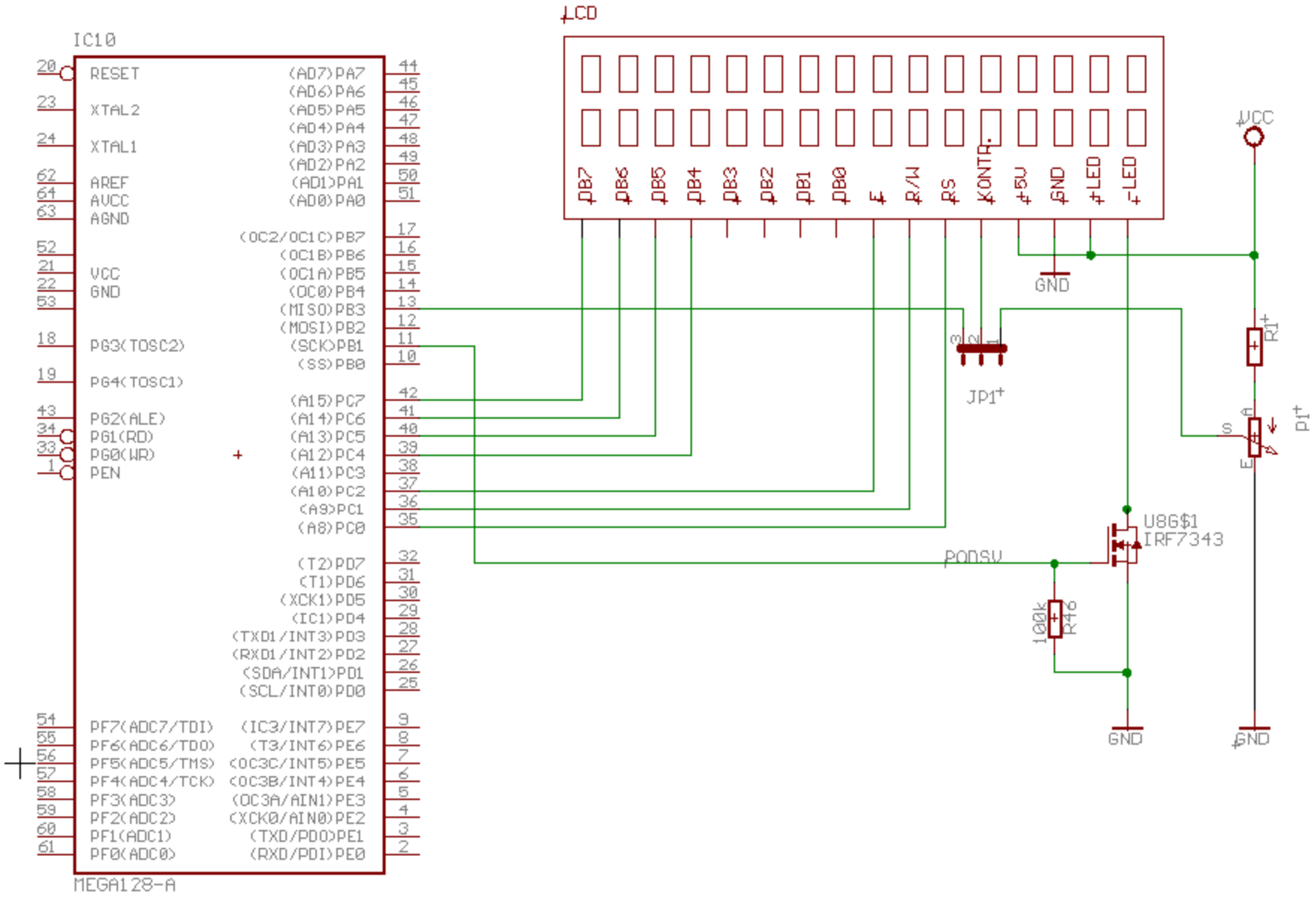
a15	a14	a13	a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
1	RS	x	x	x	R/W	x	x	x	x	x	x	x	x	x	x

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


IC2
80C552
CB
CB
RTC
IC4
RTC72412
IC10
74AC573N
IC1
27C256
IC6
28C64AP
IC3
62C256
EPROM
EEPROM
RAM
AB
DB
LCD
Data
A14
A10
A15
RD
WR
E=A15*(RD+WR)
D0÷D7
RS
RW
E

Prepojenie LCD, klávesnice a AVR




IC10
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
PG3(TOSC2)
PG4(TOSC1)
PG2(ALE)
PG1(RD)
PG0(WR)
PEN
(AD7)PA7
(AD6)PA6
(AD5)PA5
(AD4)PA4
(AD3)PA3
(AD2)PA2
(AD1)PA1
(AD0)PA0
(OC2/OC1C)PB7
(OC1B)PB6
(OC1A)PB5
(OC0)PB4
(MISO)PB3
(MOSI)PB2
(SCK)PB1
(SS)PB0
(A15)PC7
(A14)PC6
(A13)PC5
(A12)PC4
(A11)PC3
(A10)PC2
(A9)PC1
(A8)PC0
(T2)PD7
(T1)PD6
(XCK1)PD5
(IC1)PD4
(TXD1/INT3)PD3
(RXD1/INT2)PD2
(SDA/INT1)PD1
(SCL/INT0)PD0
PF7(ADC7/TDI)
PF6(ADC6/TDO)
PF5(ADC5/TMS)
PF4(ADC4/TCK)
PF3(ADC3)
PF2(ADC2)
PF1(ADC1)
PF0(ADC0)
(IC3/INT7)PE7
(T3/INT6)PE6
(OC3C/INT5)PE5
(OC3B/INT4)PE4
(OC3A/AIN1)PE3
(XCK0/AIN0)PE2
(TXD/PDO)PE1
(RXD/PDI)PE0
MEGA128-A
LCD
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
RS
KONTR.
+5V
GND
+LED
-LED
JP1
VCC
R1
P1
U86$1
IRF7343
PODSV
100k
R46
GND
GND
GND

Preddefinované znaky ASCII

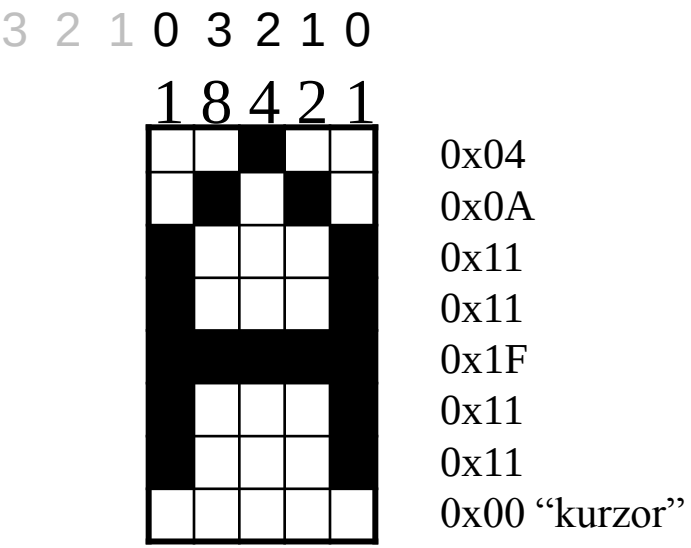
Znak (písmeno) *odpovedá* číselnému kódu.

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

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

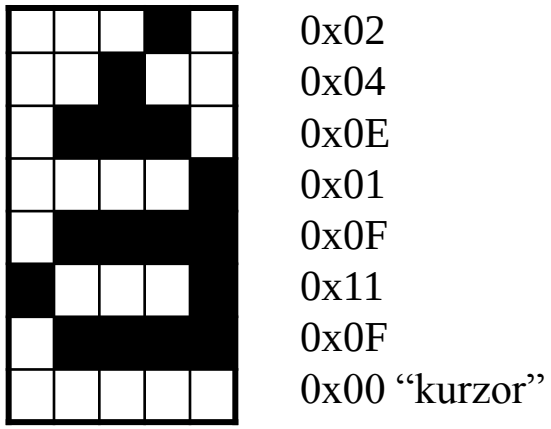
Znak: **A**

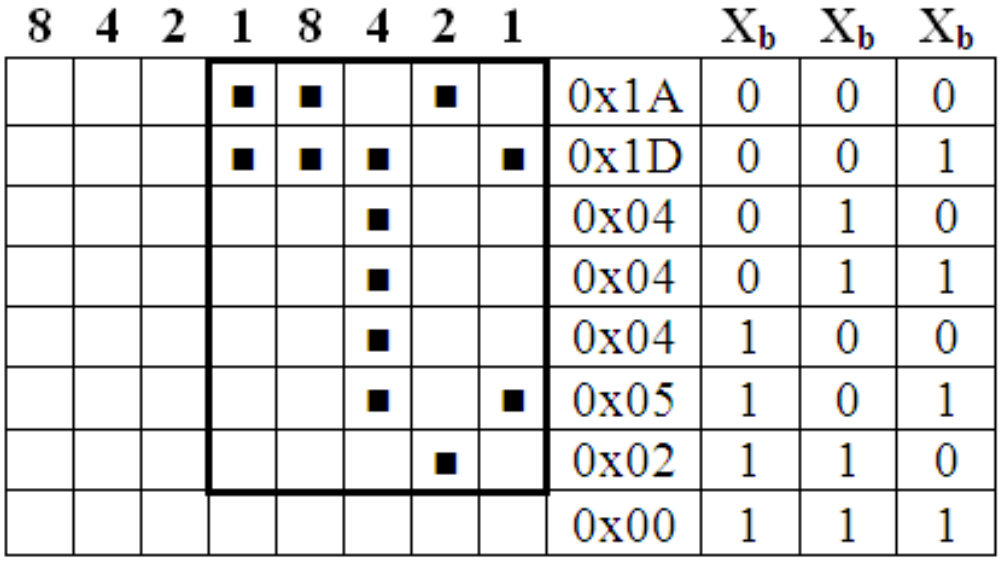
ASCII kód: 0x41



Znak: **á**

ASCII kód: 0x??





Nastavíme AC do CGRAM:
wr_d(in_da, FALSE);

Nesmiem zabudnúť na pôvodnu hodnotu AC.
Pôvodne AC ukazovalo niekam do DDRAM.

in_da = 0 1 Xa Xa Xa Xb Xb Xb

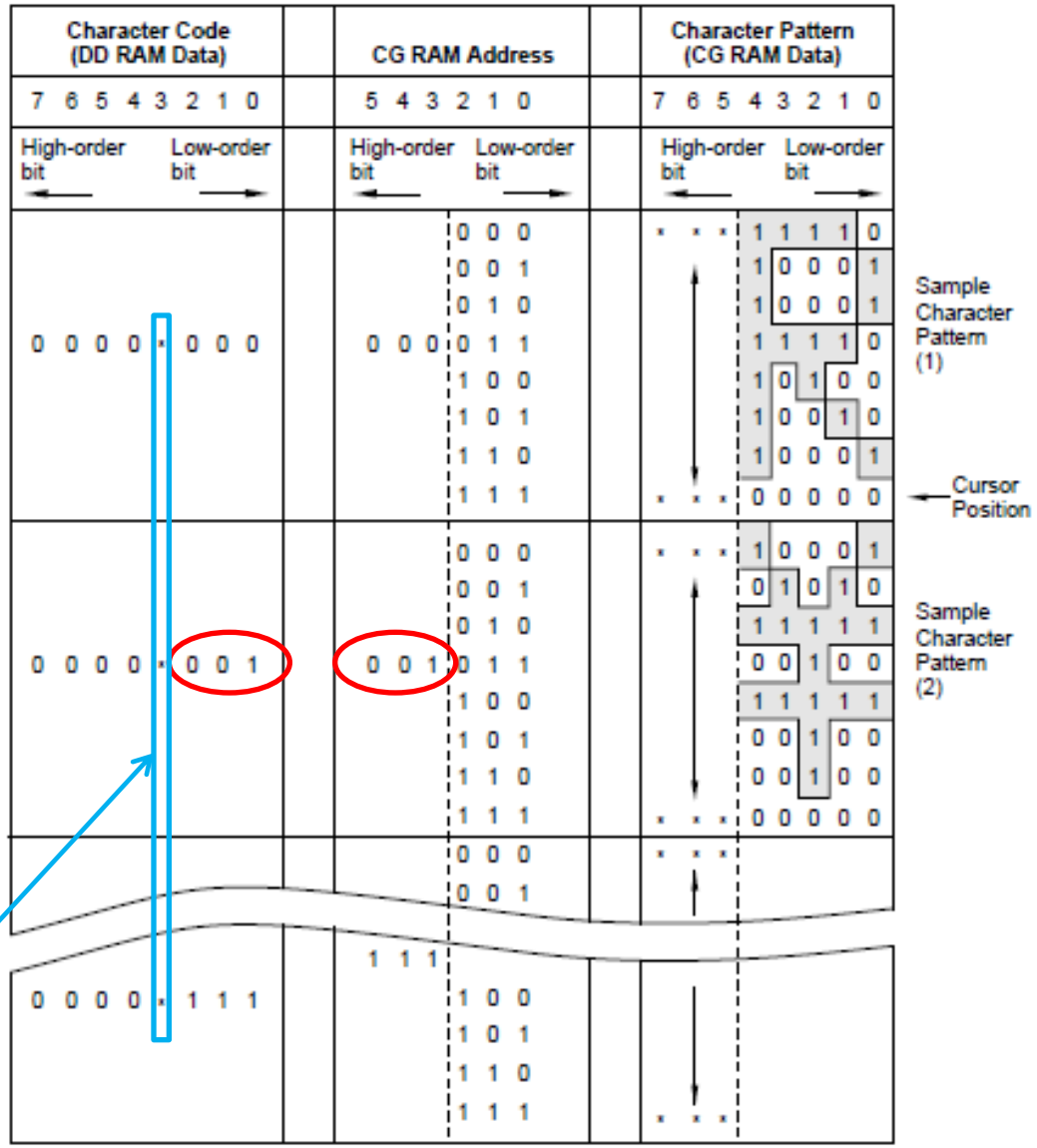
0 0 0

0 0 0

adresa znaku 0 (až 7), resp. 8 (až 15)

```
code unsigned char zn[0]= {0x1A,0x1D,4,4,4,5,2,0};
for(i=0;i<8;i++) wr_d(zn0[i],TRUE);
// AC sa inkrementuje automaticky
```

Table 6. Relationship Among Character Code
(DD RAM), CG RAM Address, and Character Pattern (CG RAM)



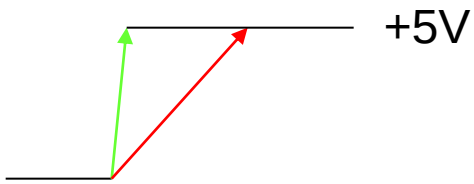
Zrkadlenie
b3 = *

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

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



potom, sa radič nastaví do základného módu .
  - Jeden riadok, 8 znakov v riadku, 4-bitové pripojenie, ...
- Ak sa nevykoná, hardwarový RST a inicializácia, treba vykonať softwarovú inicializáciu.

Reset vyvolaný po nábehu napájania:

- 1. Clear Display (Instr.= 0x01)

  Kurzor: Ľavý horný roh. AC = 0x00. I/D = 1 (inkrement)
- 2. Function Set (Instr.= 0x30)

  DL = 1 ... DB je 8-bitová.

  N = 0 ... Jednoriadkový display

  F = 0 .... Font 5x7
- 3. Display ON/OFF Control (Instr.= 0x08)

  D = 0 ... Displej vypnutý.

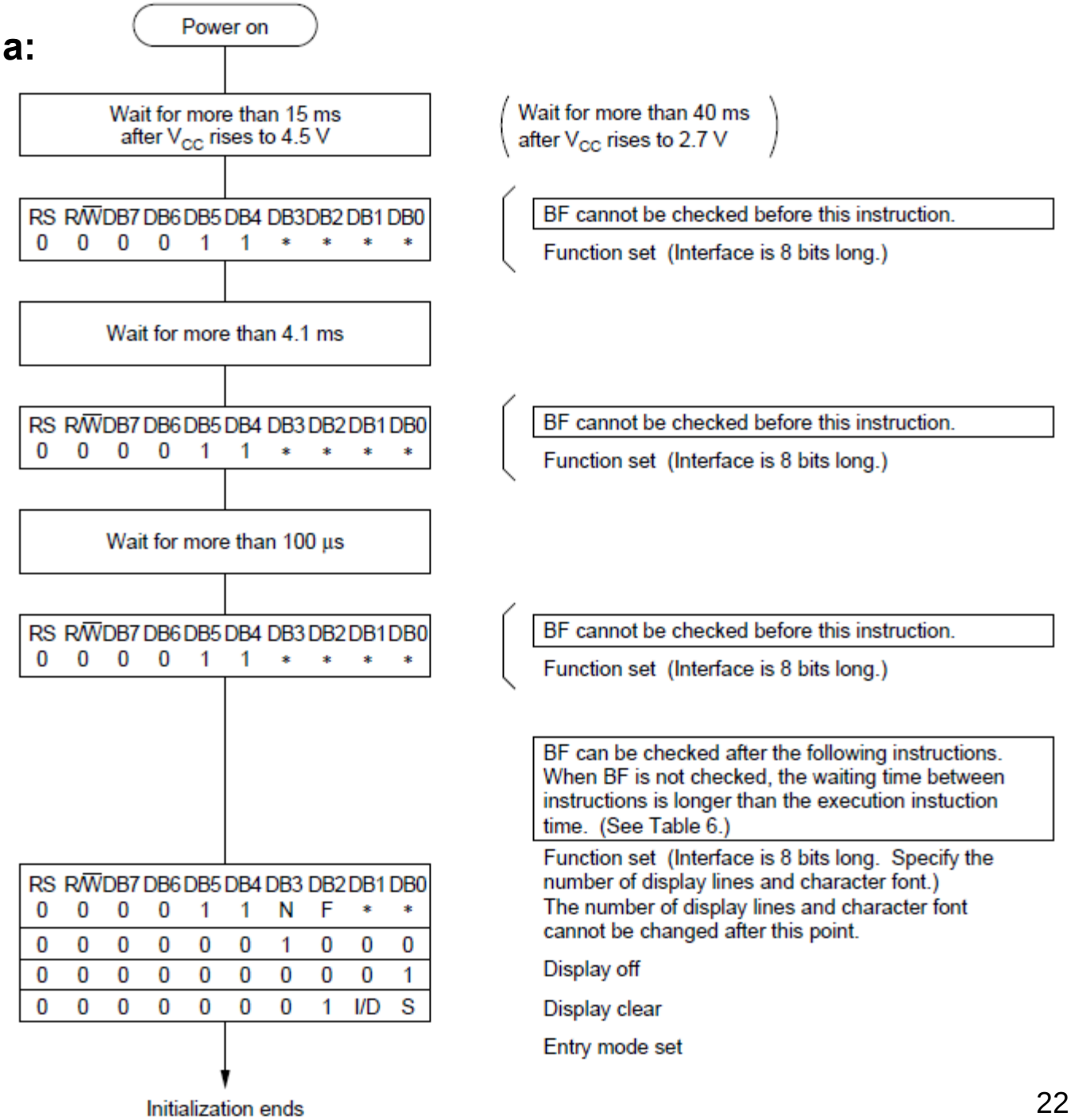
  C = 0 ... Kurzor vypnutý.

  B = 0 .... Blikanie kurzora vypnuté.
- 4. Entry Mode Set (Instr.= 0x06)

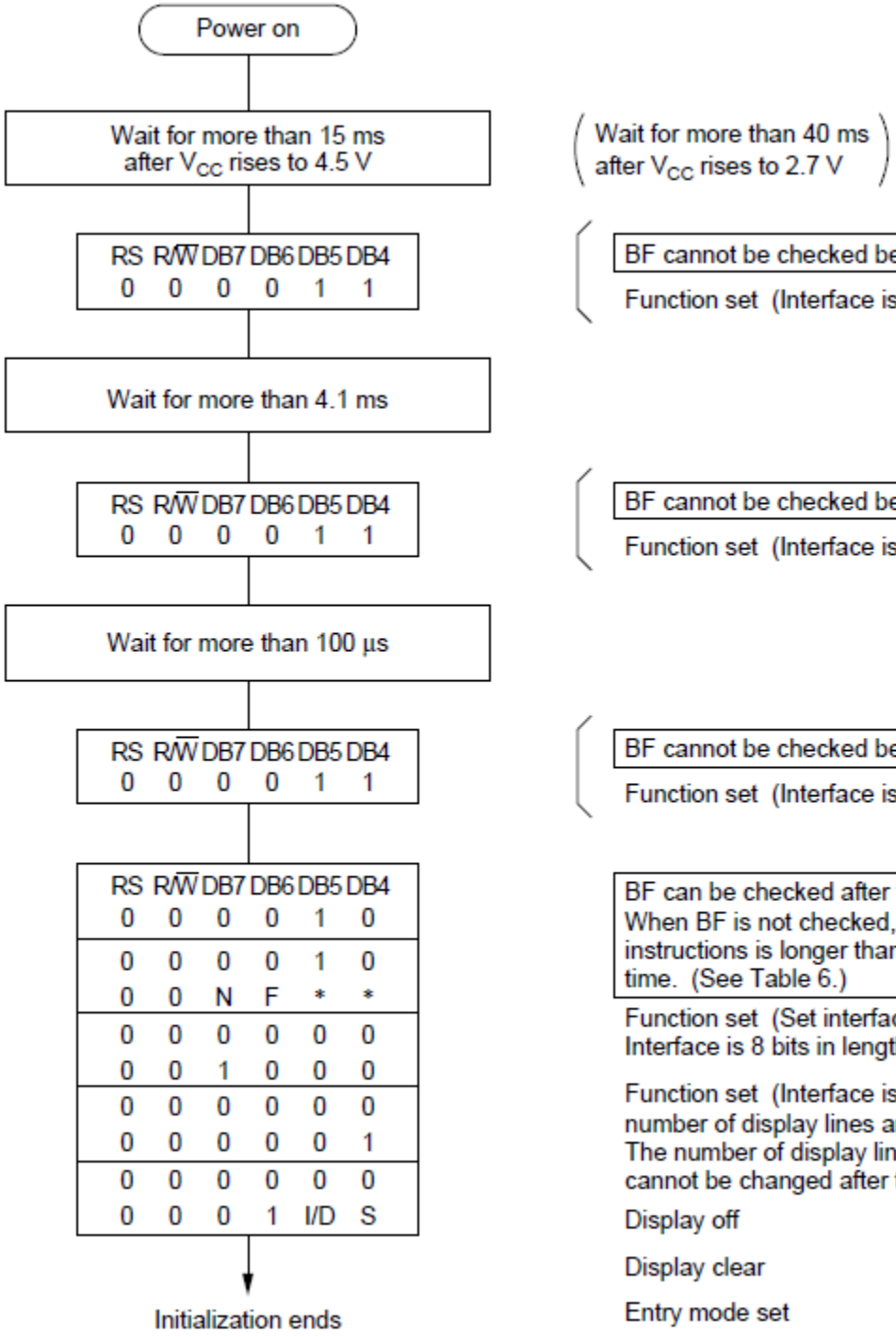
  I/D = 1 ... Inkrementovanie.

  S = 0 ... Posúvanie displeja vypnuté.

Softvérová inicializácia:
8 - bit Interface



Softvérová inicializácia: 4 - bit Interface



Jedno-riadkové vs. Viac-riadkové displeje

Displej 1x8

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

N = 0 → jednoriadkový display

Jedno-riadkové vs. Viac-riadkové displeje

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.

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

Jedno-riadkové vs. 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

Správne vykonávanie operácií

Čas trvania inštrukcií t_v je premenlivý, Tab.1.

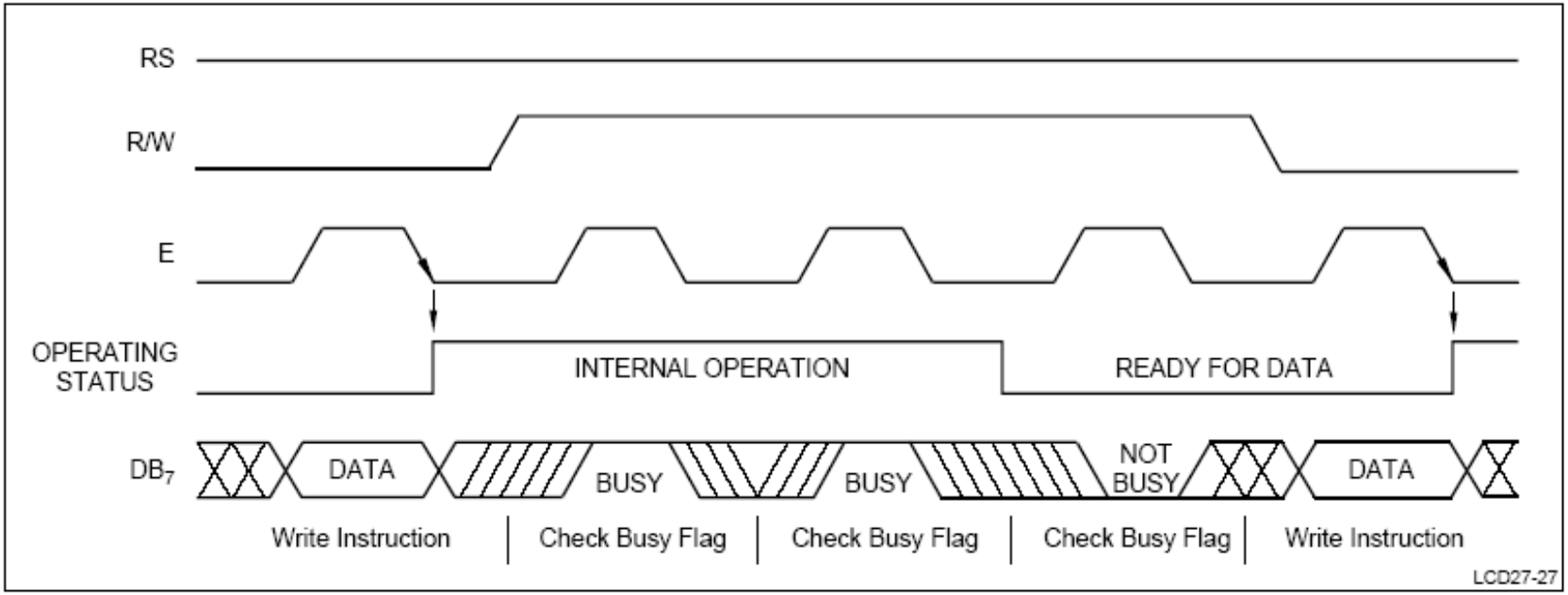
$T_v = f(f_{OSC\ disp}$, typ instr.); $f_{OSC\ disp} = 270, 250kHz \pm 30\%$

Počas spracovávania príkazu, LCD ďalší príkaz neakceptuje.

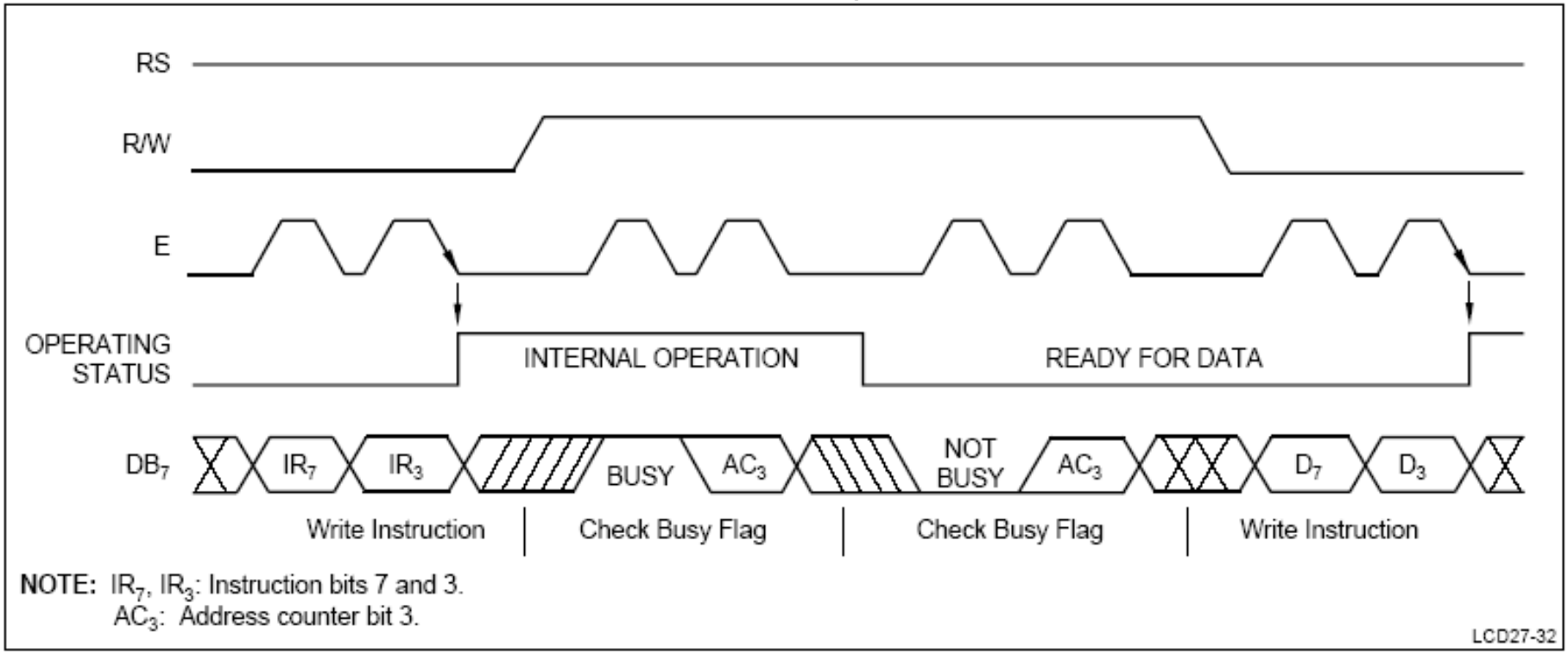
Máme dve možnosti:

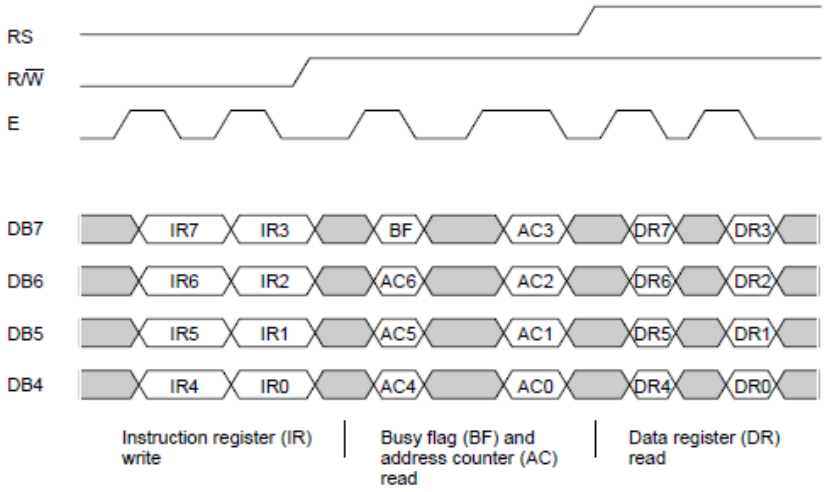
- „Počkáme“
- Testujeme stav signálu BUSY (bitová premenná)
  - Ak je LCD BUSY, potom je DB7 = „1“, ak je predchádzjúa operácia ukončená DB7 = „0“

Príklad osem bitovej komunikácie



Príklad štvor-bitovej komunikácie





Práca v systéme reálneho času.

! !
! ! ! !
! ! Pri kontrole bitu BF nesmieme zabudnúť na timeout . ! !
! ! ! !
! !

Nespomenuli sme hardwarový spôsob „genorovania“ kontrastu displeja.

- D/A prevodník. Takúto periférium má zabudovaný len malý počet jednočipových mikrokotrolerov.

- PWM signál. Ak nevhodné nastavíme periódu opakovania PWM signálu, zistíme, že v „danom okamžiku“ zobrazuje len časť displeja. Zobrazovaná časť sa posúva . Spôsob posúvania je daný reláciou $f_{OSC} < = > f_{PWM}$.