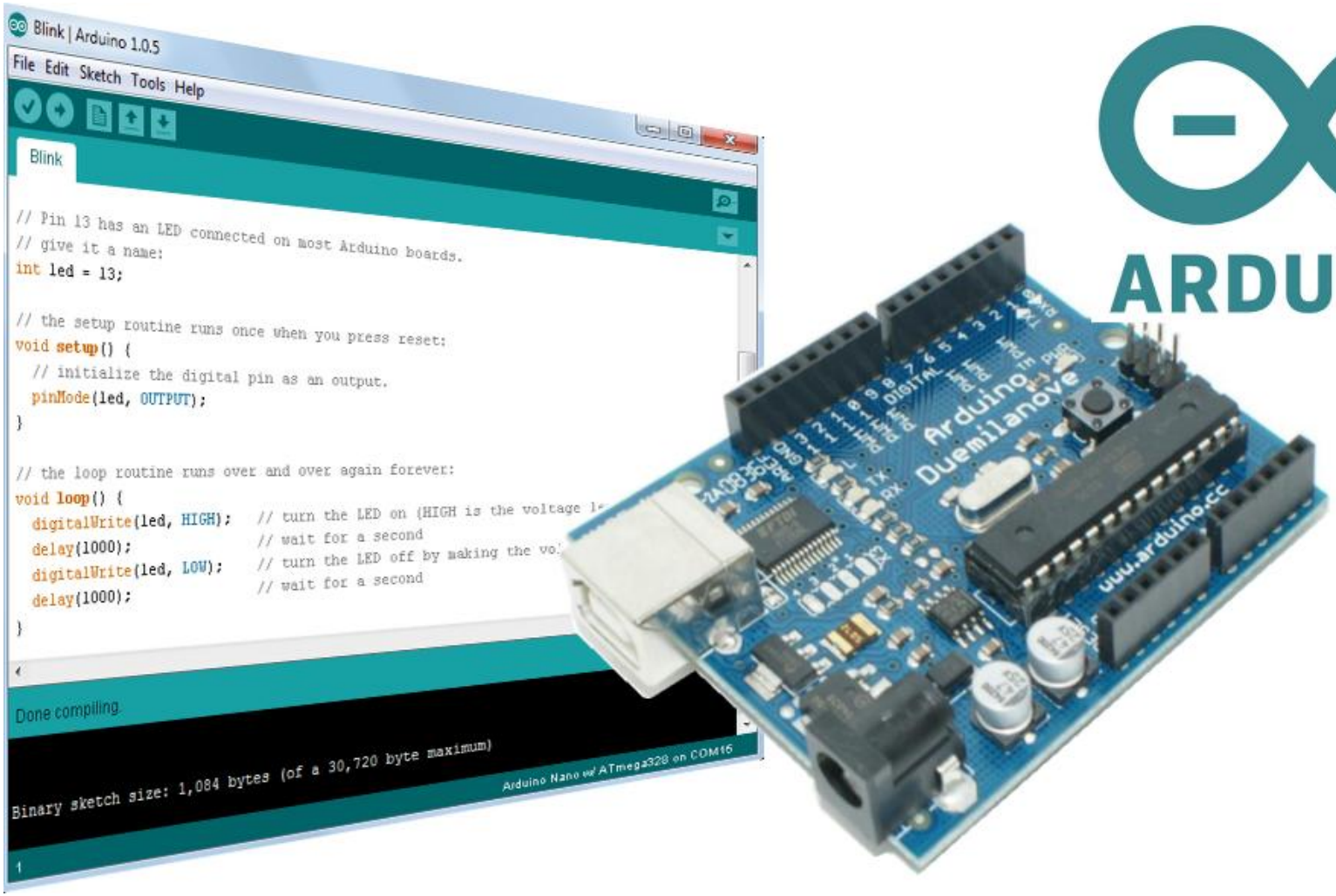


Bevezetés a mikrovezérlők programozásába:

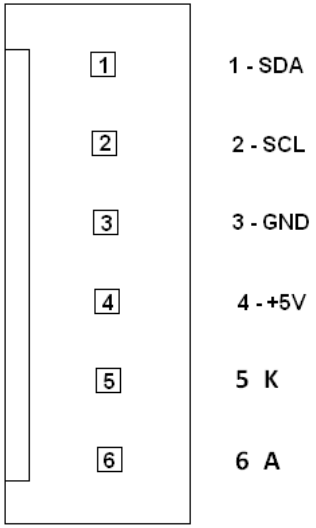
A PM6025 (7 és 16 szegmenses) LCD kijelző vezérlése

Lab 21 projektek

- **MiniPirate.ino** – Arduino Mini Pirate, interaktív vizsgálóprogram , amelyet itt az I2C busz „kézivezérlésére” használunk fel.
- **PM6025_test.ino** – tesztprogram a PM6025 LCD kijelző kipróbálásához
- **libraries/PM6025** – új Arduino programkönyvtár és API a PM6025 LCD kijelző kezeléséhez

A Displaytech PM6025 kijelző

- Többnyire DIBAL mérlegekben használják
- 2 x 13 karakter 7 szegmenses kijelző,
  9 pozícióban tizedesponttal vagy vesszővel
- 1 x 14 karakter 16 szegmenses kijelző
- 4 + 5 oldalsó szimbólum kijelzéssel
- Vezérlés I2C buszon (cím 0x38)
- 3 db PCF8576 vezérlőt tartalmaz



1 - SDA

2 - SCL

3 - GND

4 - +5V

5 K

6 A

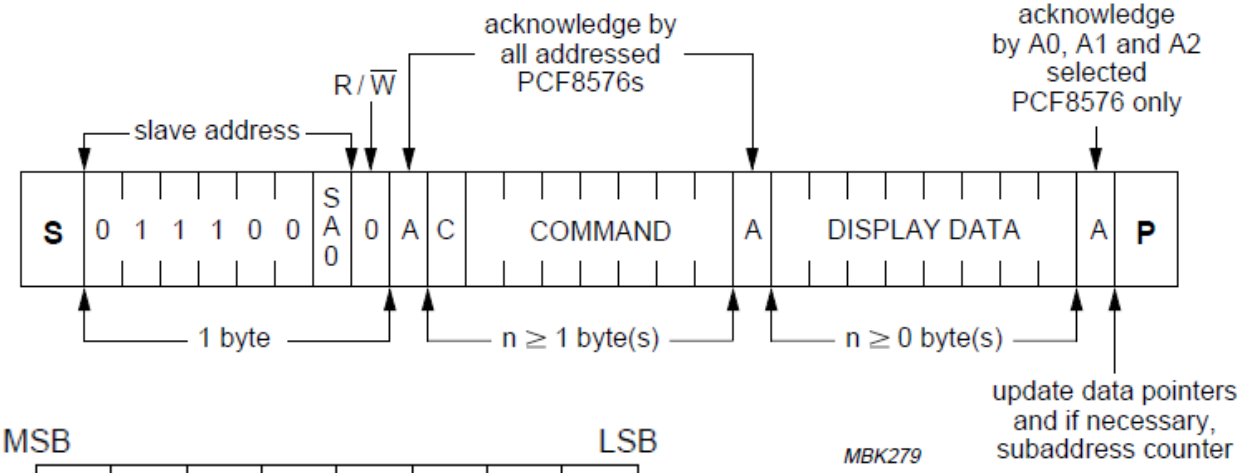
A Displaytech PM6025 kijelző

A sorok számozása fentről lefelé, a karakterek számozása pedig jobbról balra történik – a belső memóriakiosztást követve…

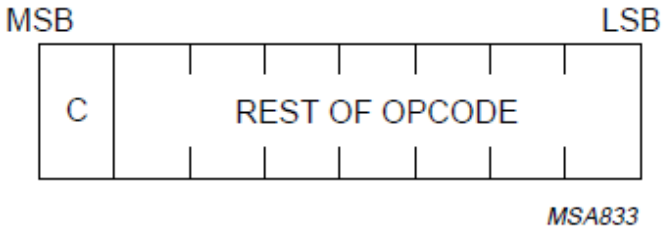
I2C kommunikáció

- A kijelzőt 3 db **PCF8576** IC vezérli. A közös **I2C cím 0x38,** az egyedi címzés a parancsbájtoknál adható meg (**0xE0, 0xE1** vagy **0xE2** parancs). Mindegyik IC legfeljebb 160 szegmens vezérlésére képes, az 1:4 multiplex módot használva.
- Egymás után több parancsbájt is kiadható, amíg a C bit (0x80 helyiérték) ’1’ értékű. Az utolsó parancsnál (ami általában egy memória kezdőcím beállítás) a C bit ’0’ legyen!
- Üzemmód beállítására **0xC8** parancs ajánlott.

Az I2C tranzakciók szerkezete



Parancsbájt szerkezete:



C = 0; last command.

C = 1; commands continue.

PCF8576 parancsok

Table 4 Definition of PCF8576 commands

COMMAND	OPCODE	OPTIONS	DESCRIPTION
MODE SET	C 1 0 LP E B M1 M0 Low power Enable Bias Mode	Table 5	Defines LCD drive mode.
		Table 6	Defines LCD bias configuration.
		Table 7	Defines display status. The possibility to disable the display allows implementation of blinking under external control.
		Table 8	Defines power dissipation mode.
LOAD DATA POINTER	C 0 P5 P4 P3 P2 P1 P0	Table 9	Six bits of immediate data, bits P5 to P0, are transferred to the data pointer to define one of forty display RAM addresses.
DEVICE SELECT	C 1 1 0 0 A2 A1 A0	Table 10	Three bits of immediate data, bits A2 to A0, are transferred to the subaddress counter to define one of eight hardware subaddresses.
BANK SELECT	C 1 1 1 1 0 I O No effect in 1:4 mode!	Table 11	Defines input bank selection (storage of arriving display data).
		Table 12	Defines output bank selection (retrieval of LCD display data). The BANK SELECT command has no effect in 1 : 3 and 1 : 4 multiplex drive modes.
BLINK	C 1 1 1 0 A BF1 BF0 Blinking frequency	Table 13	Defines the blinking frequency.
		Table 14	Selects the blinking mode; normal operation with frequency set by BF1, BF0 or blinking by alternation of display RAM banks. Alternation blinking does not apply in 1 : 3 and 1 : 4 multiplex drive modes.

Kiegészítő táblázatok

Table 5 MODE SET option 1

LCD DRIVE MODE		BITS	
DRIVE MODE	**BACKPLANE**	**M1**	**M0**
Static	1 BP	0	1
1 : 2	MUX (2 BP)	1	0
1 : 3	MUX (3 BP)	1	1
1 : 4	MUX (4 BP)	0	0

Table 6 MODE SET option 2

LCD BIAS	BIT B
$\frac{1}{3}$bias	0
$\frac{1}{2}$bias	1

Table 7 MODE SET option 3

DISPLAY STATUS	BIT E
Disabled (blank)	0
Enabled	1

Table 8 MODE SET option 4

MODE	BIT LP
Normal mode	0
Power-saving mode	1

Table 9 LOAD DATA POINTER option 1

DESCRIPTION	BITS					
6-bit binary value of 0 to 39	P5	P4	P3	P2	P1	P0

Table 10 DEVICE SELECT option 1

DESCRIPTION	BITS		
3-bit binary value of 0 to 7	A2	A1	A0

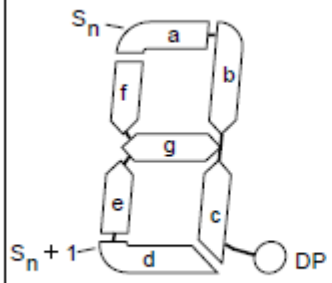
Table 13 BLINK option 1

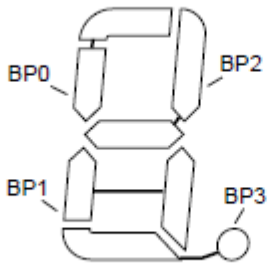
BLINK FREQUENCY	BITS	
	BF1	**BF0**
Off	0	0
2 Hz	0	1
1 Hz	1	0
0.5 Hz	1	1

Az adatmemória szerepe

1 : 4

multiplex





		n	n+1
bit/ BP	0	a	f
	1	c	e
	2	b	g
	3	DP	d

MSB LSB

a	c	b	DP	f	e	g	d

A LOAD DATA POINTER paranccsal 4-bájtos egységekben címezhetünk, de mindig 8 bites egységekben történik az adatátvitel. Egy 7-szegmeneses karakter 8 bitet foglal le.

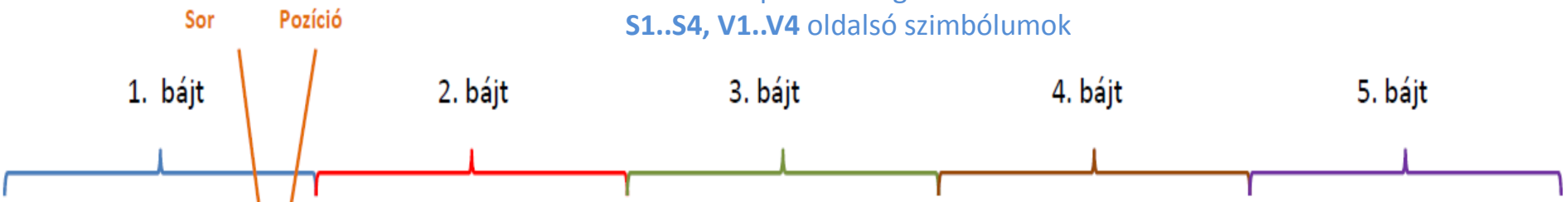
Az adatlappal ellentétben ennél a kijelzőnél a tizedespontok illetve tizedesvesszők vezérlése külön félbájtos adatrekeszek felhasználásával történik. Ez a címzést egy kicsit megbonyolítja.

A 16-szegmenses karakterek vezérlése karakterenként Két bájtot használ fel **cdhj aeko bflp gimn** bitsorrendben.

Bittérkép

A kijelző három vezérlője 3 x 20 = 60 bájtnyi információt kezel. Az egyes bitek szerepe az alábbi ábrán látható:

x nem használt bit
p tizedespont
v A tizedespontok kiegészítő vessző
S1..S4, V1..V4 oldalsó szimbólumok



MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB	MSB	LSB
a b c x 2/1	f g e d 2/1	a b c x 1/1	f g e d 1/1	p v p v (1a)	a b c x 2/2	f g e d 2/2	a b c x 1/2	f g e d 1/2	p v p v (2a)
a b c x 2/3	f g e d 2/3	a b c x 1/3	f g e d 1/3	p v p v (3a)	a b c x 2/4	f g e d 2/4	a b c x 1/4	f g e d 1/4	p v p v (4a)
a b c x 2/5	f g e d 2/5	a b c x 1/5	f g e d 1/5	p v p v (5a)	a b c x 2/6	f g e d 2/6	a b c x 1/6	f g e d 1/6	p v p v (6a)
a b c x 2/7	f g e d 2/7	a b c x 1/7	f g e d 1/7	p v p v (1b)	a b c x 2/8	f g e d 2/8	a b c x 1/8	f g e d 1/8	a b c x 2/9

f g e d 2/9	a b c x 1/9	f g e d 1/9	a b c x 2/10	f g e d 2/10	a b c x 1/10	f g e d 1/10	a b c x 2/11	f g e d 2/11	a b c x 1/11
f g e d 1/11	p v p v (3b)	a b c x 2/12	f g e d 2/12	a b c x 1/12	f g e d 1/12	a b c x 2/13	f g e d 2/13	a b c x 1/13	f g e d 1/13
c d h j 3/1	a e k o 3/1	b f l p 3/1	g i m n 3/1	c d h j 3/2	a e k o 3/2	b f l p 3/2	g i m n 3/2	c d h j 3/3	a e k o 3/3
b f l p 3/3	g i m n 3/3	c d h j 3/4	a e k o 3/4	b f l p 3/4	g i m n 3/4	c d h j 3/5	a e k o 3/5	b f l p 3/5	g i m n 3/5

c d h j 3/6	a e k o 3/6	b f l p 3/6	g i m n 3/6	c d h j 3/7	a e k o 3/7	b f l p 3/7	g i m n 3/7	c d h j 3/8	a e k o 3/8
b f l p 3/8	g i m n 3/8	c d h j 3/9	a e k o 3/9	b f l p 3/9	g i m n 3/9	c d h j 3/10	a e k o 3/10	b f l p 3/10	g i m n 3/10
c d h j 3/11	a e k o 3/11	b f l p 3/11	g i m n 3/11	c d h j 3/12	a e k o 3/12	b f l p 3/12	g i m n 3/12	c d h j 3/13	a e k o 3/13
b f l p 3/13	g i m n 3/13	c d h j 3/14	a e k o 3/14	b f l p 3/14	g i m n 3/14	S1 S2 S3 S4	p v p v (2b)	V1 V2 V3 V4	x x x V5

Karakterpozíciók címzése

A karakterek címzése a LOAD DAPA POINTER paranccsal történhet (félbájtos címeket kezel!)
Természetesen a címeket az E0, E1, E2 modulon belül modulo 40 kell venni!
Például a 2b pozíciójú tizedesvesszők (1 és 2. sor 10. karaktere mellett) az alábbi MiniPirate paranccsal jeleníthető meg: `w0xc8 0xE2 0x25 0xF0`
ahol 0xC8 az üzemmód, 0xE2 a modul címzés, 0x25 a 117. félbájt címe a modulon belül (117 – 80 = 37 = 0x25), 0xF0 pedig a pvpv bitek 1-be állításához szükséges adat…

Karakter pozíció	1.sor	2. sor	3.sor	DP/DC
1	2	0	60	
2	7	5	64	1a: 4
3	12	10	68	2a: 9
4	17	15	72	3a: 14
5	22	20	76	4a: 19
6	27	25	80	5a: 24
7	32	30	84	6a: 29
8	37	35	88	
9	41	39	92	1b: 34
10	45	43	96	2b: 117
11	49	47	100	3b: 51
12	54	52	104	
13	58	56	108	
14			112	

MiniPirate példák

Inicializálás és képernyőtörlés:

```
w0XC8 0xE0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
w0xE1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
w0xE2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Teszt mód:

```
w0XC8 0xE0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255
w0xE1 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255
w0xE2 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255
```

HELLO WORLD kiírása a 3. sorba

```
w0xC8 0xE1 20 0x0D 0xB9 0x91 0x10 0xB8 0x8e 0x99 0x99 0x91 0x39 0 0
0x99 0x99 0x91 0x10 0x91 0x10 0xB9 0x94 0xB0 0x0D
```

Az 0xE1 20 címzés a 60. félbájthoz pozícionálja a (képzeletbeli) kurzort (20 = 60%40)

Az utolsó helyről visszafelé írunk, mint a rák…

(D = 0x0DB9; L = 0x9110; R = 0xB88e; O = 0x9999; W = 0x9139; E = 0xB994; H = 0xB00D)

Számkijelzés: `w0XC8 0xE0 0 0x60 0 0x0C 0x70 0x03 0xE5`

Ez a parancs 3,21-et ír ki a 2. sor jobb szélére (1, 2, 3. pozíció)

(w – I2C write parancs, 0xC8 – üzemmód beállítás, 0xE0 – eszköz kiválasztás,
0 – adatmutató beállítása, 0x60 0 0x0C 0x70 0x03 0xE5 – a kiírandó adatok)

7-szegmens hexadecimális font

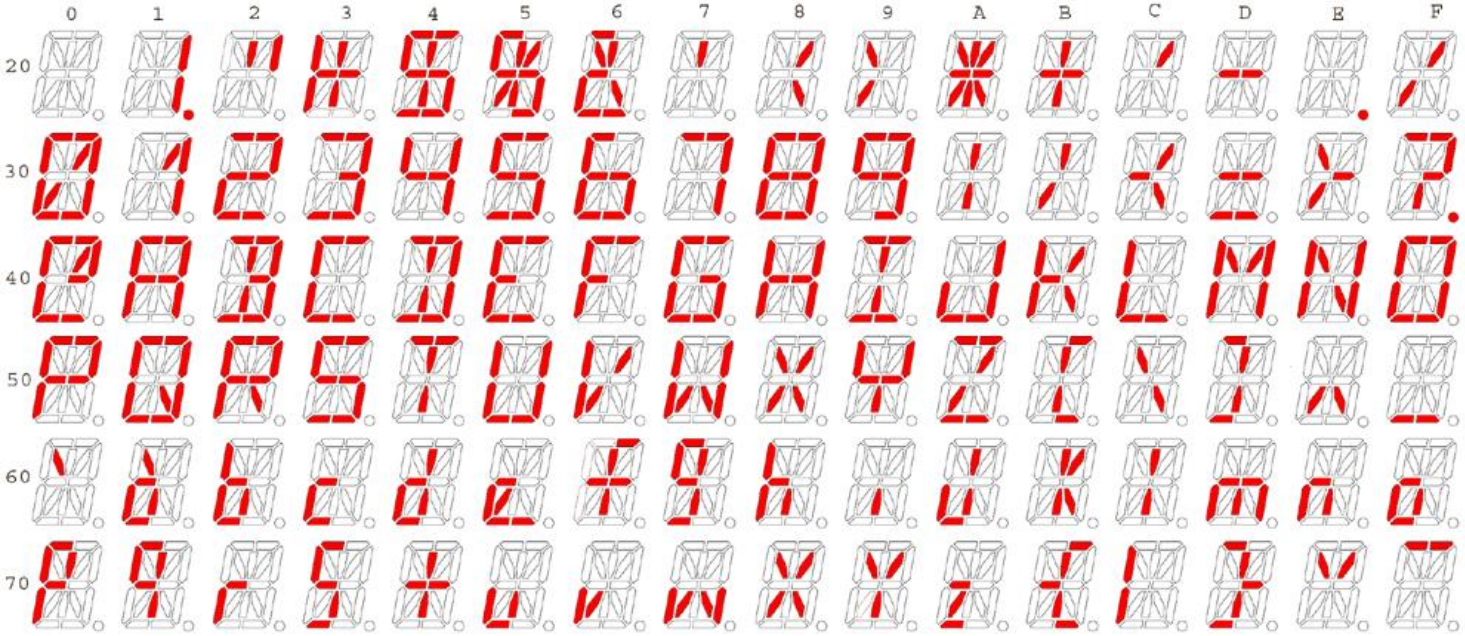
```
const static uint8_t numTable[] = {
    0xEB, 0x60, 0xC7, 0xE5, 0x6C, 0xAD, 0xAF, 0xE0, 0xEF,
    0xED, 0xEE, 0x2F, 0x8B, 0x67, 0x8F, 0x8E, 0x00, 0x04 };
/*          abcx fged           abcd efgx
        0b1110 1011,      //0 1111 1100
        0b0110 0000,      //1 0110 0000
        0b1100 0111,      //2 1101 1010
        0b1110 0101,      //3 1111 0010
        0b0110 1100,      //4 0110 0110
        0b1010 1101,      //5 1011 0110
        0b1010 1111,      //6 1011 1110
        0b1110 0000,      //7 1110 0000
        0b1110 1111,      //8 1111 1110
        0b1110 1101       //9 1111 0110
        0b1110 1110       //A 1110 1110
        0b0010 1111       //b 0011 1110
        0b1000 1011       //C 1001 1100
        0b0110 0111       //d 0111 1010
        0b1000 1111       //E 1001 1110
        0b1000 1110       //F 1000 1110
        0b0000 0000       //SPACE
        0b0000 0100       //MINUS SIGN
*/
```

A hexadecimális karaktereken kívül egy üres karaktert (szóköz)és egy mínusz jelet is definiáltunk.

A bitsorrendet a kijelző memóriájának bitkiosztása miatt egy kicsit meg kellett forgatni…

Fontkép 16-szegmensre

A 16-szegmenses kijelzőhöz használandó fontot az alábbi ábra alapján készíthetjük el. Az egyszerűség kedvéért csak a decimális számjegyeket és a nagybetűket tartalmazó fonttáblát készítettünk. A szóköznek megfelelő üres karakter itt a kettőspont helyére kerül, ezért kiíratásnál a SPACE kódját ide transzponáljuk automatikusan.



A PM6025 programkönyvtár

Állományok: PM6025.h és PM6025.cpp

A PM6025.h fejléc állomány definiálja a 7- és 16-szegmenses fontokat, a karakterpozíciók és a tizedespontok/vesszők félbájtos címeit és a szimbólumokhoz tartozó bitmaszkokat. Emellett deklarálja a **PM6025** objektumosztályt, valamint annak privát és publikus tagfüggvényeit.

A PM6025 objektumosztály publikus tagfüggvényei:

```
void init(uint8_t data = 0x00);                          //Initialize the I2C master mode
void setDigit(uint8_t row, uint8_t pos, uint8_t c);      //Display a 7-seg digit   (only in row 1 or 2)
void setChar(uint8_t pos, char c);                       //Display a 16-seg character (only in row 3)
void setSymbols(uint16_t flags);                         //Display/clear sidebar symbols (S1..S4, V1..V5)
void setDP(uint8_t row, uint8_t pos);                    //Display a decimal comma (pos 1..6, 8..10; row 1 or 2)
void clearDP(uint8_t row, uint8_t pos);                  //Remove a decimal comma (pos 1..6, 8..10; row 1 or 2)
void write(uint8_t row, uint8_t pos, long data);         //Display signed integers (only in row 1 or 2)
```

A repertoár még nem teljes, ízlés szerint tovább bővíthető…

PM6025_test.ino

```
#include <Wire.h>
#include "PM6025.h"
PM6025 lcd;

void setup() {
  lcd.init(0xFF);     //Test mode
  delay(3000);
}

void loop() {
  long a = 314159265;
  int b = -6025;
  long c = -5729578;
//---- Clear screen ----------
  lcd.init();  delay(1000);
//-- Write signed numbers ----
  lcd.write(1,1,a);
  lcd.setDP(1,8);     //3,14159265
  lcd.write(2,1,c);
  lcd.setDP(2,5);     //-57,29578
  lcd.write(2,9,b);   //-6025
  delay(5000);
```

```
//---- Clear screen ---------------
  lcd.init();

//---- Write digits into Row 1 ----
  for(int k=0; k<13; k++) {
    delay(250);
    lcd.setDigit(1,(13-k),k);
  }
  delay(1000);
//---- Write digits into Row 2 -----
  for(int k=0; k<13; k++) {
    delay(250);
    lcd.setDigit(2,(13-k),k+3);
  }
  delay(1000);
//---- Write letters into Row 3 ----
  for(int k=14; k>0; k--) {
    delay(250);
    lcd.setChar(k,('N'-k));
  }
  delay(1000);
```

Folyatatás a következő oldalon…

PM6025_test.ino (folytatás)

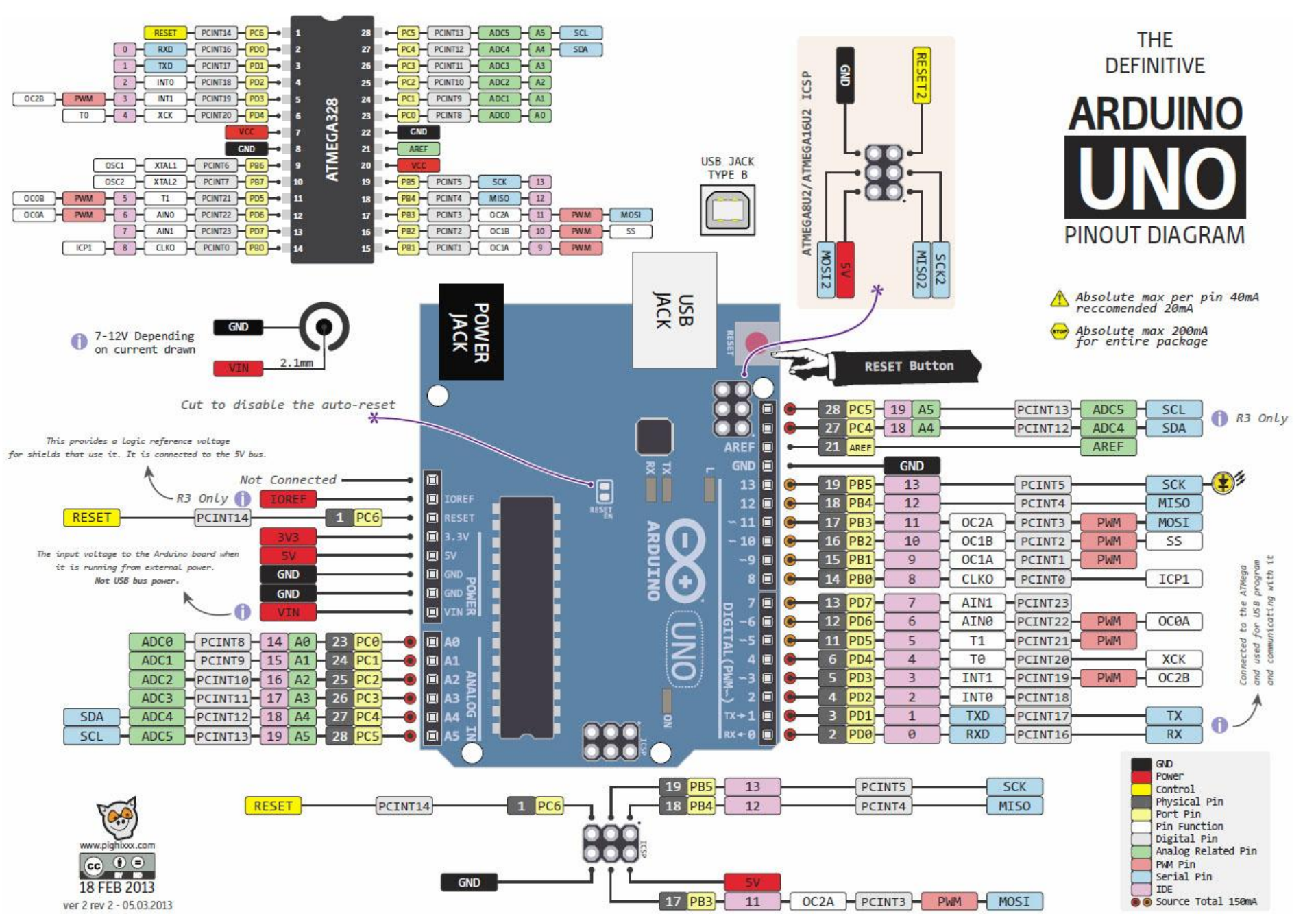
```
//---- Write decimal commas --------
  for(int j=1; j<3; j++) {
    for(int k=1; k<11; k++) {
      lcd.setDP(j,k); delay(500);
     }
  }
  delay(1000);
//---- Clear decimal commas --------
  for(int j=1; j<3; j++) {
    for(int k=1; k<11; k++) {
      lcd.clearDP(j,k);
       delay(250);
    }
  }
  delay(1000);
//---- Show sidebar symbols --------
  for(int k=0; k<5; k++) {
    lcd.setSymbols(sV1 | sNull);     delay(500);
    lcd.setSymbols(sV2 | sNet);      delay(500);
    lcd.setSymbols(sV3 | sBalance); delay(500);
    lcd.setSymbols(sV4 | sBattery); delay(500);
    lcd.setSymbols(sV5);             delay(500);
  }
//---- Show all sidebar symbols ----
  lcd.setSymbols(0xF0F1);            delay(2000);
}
```

Mindkét sorban egyenként kiírja a tizedesvesszőket

Mindkét sorban egyenként kitörli a tizedesvesszőket

Az oldalsó szimbólumok megjelenítése egyenként…,

ill. együttesen


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
2.1mm
Cut to disable the auto-reset
This provides a Logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Connected to the ATMega and used for USB program and communicating with it
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

Emlékeztető: Arduino nano v3.0