LEGO robotok

XVII. rész

III.3.4.2. Konstansok

A konstansokat az *ev3_constants.h* foglalja magában.
Egy pár fontosabb konstans a következő:

Általános konstansok:

```
TRUE  1 // igaz érték
FALSE 0 // hamis érték

NUM_INPUTS     4   // bemeneti portok száma
NUM LEDS       4   // LED-ek száma
LCD WIDTH      178 // a kijelző vízszintes mérete
LCD HEIGHT     128 // a kijelző függőleges mérete
TOPLINE_HEIGHT 10  // a felső sor magassága

OWNER_NONE 0x0000 // egy erőforrás tulajdonosa
```

A téglák láncolásának konstansai:

```
LAYER MASTER 0x00 // mester
LAYER SLAVE1 0x10 // szolga 1
LAYER SLAVE2 0x20 // szolga 2
LAYER_SLAVE3 0x40 // szolga 3
LAYER_MASK   0x70 // a réteg maszk
```

A kimenet konstansai:

```
OUT A    0x01 // A port
OUT B    0x02 // B port
OUT C    0x04 // C port
OUT D    0x08 // D port
OUT_AB   0x03 // A és B portok
OUT_AC   0x05 // A és C portok
OUT AD   0x09 // A és D portok
OUT BC   0x06 // B és C portok
OUT BD   0x0a // B és D portok
OUT CD   0x0c // C és D portok
OUT_ABC  0x07 // A, B és C portok
OUT_BCD  0x0e // B, C és D portok
OUT ABCD 0x0f // A, B, C és D portok
OUT ALL  0x0f // minden port
OUT_MASK 0x0f // a kimeneti maszk

OUT_FLOAT 0x00  // motorműködés: Coast (amíg meg nem áll)
OUT_OFF   0x40  // motorműködés: Ki
OUT ON    0x80  // motorműködés: Be
OUT_REV    0x00 // motorműködés: Hátra
```

```
OUT_TOGGLE 0x40 // motorműködés: Kapcsolás
OUT_FWD    0x80 // motorműködés: Előre

OUT_POWER_DEFAULT -127 // alapértelmezett erősség

OUT REGMODE IDLE  0 // nincs kiegyenlítés
OUT REGMODE SPEED 1 // sebesség kiegyenlítés
OUT_REGMODE_SYNC  2 // két motor szinkronizálása

RESET NONE           0x00 // nincs visszaállítás
RESET_COUNT          0x08 // a belső tachométer visszaállítása
RESET_BLOCK_COUNT    0x20 // a blokk tachométer visszaállítása
RESET ROTATION COUNT 0x40 // a fordulatszámláló visszaállítása
RESET BLOCKANDTACHO  0x28 // a belső és a blokk visszaállítása
RESET_ALL            0x68 // minden visszaállítása

NUM_OUTPUTS 4   // a kimeneti portok száma
```

Gombok konstansai:

```
BUTTON_ID_UP     0x01 // Fel gomb
BUTTON_ID_ENTER  0x02 // Enter
BUTTON ID DOWN   0x04 // Le gomb
BUTTON ID RIGHT  0x08 // Jobbra gomb
BUTTON ID LEFT   0x10 // Balra gomb
BUTTON ID ESCAPE 0x20 // Kilépés gomb
BUTTON_ID_ALL    0x3f // Minden gomb

NO_OF_BTNS 6  // EV3 gombok száma
NUM_BUTTONS 6 // A rendszerben lévő gombok száma
```

Színek konstansai:

```
INPUT BLACKCOLOR  1 // Fekete
INPUT BLUECOLOR   2 // Kék
INPUT_GREENCOLOR  3 // Zöld
INPUT_YELLOWCOLOR 4 // Sárga
INPUT REDCOLOR    5 // Piros
INPUT_WHITECOLOR  6 // Fehér
```

Milliszekundumok, másodpercek, percek:

```
MS 1        1                       MS 100    100
MS 2        2                       MS 150    150
MS_3        3                       MS_200    200
MS_4        4                       MS_250    250
MS 5        5                       MS 300    300
MS 6        6                       MS 350    350
MS 7        7                       MS 400    400
MS 8        8                       MS 450    450
MS_9        9                       MS_500    500
MS_10      10                       MS_600    600
MS 20      20                       MS 700    700
MS 30      30                       MS 800    800
MS 40      40                       MS 900    900
MS 50      50                       SEC 1    1000
MS_60      60                       SEC_2    2000
MS_70      70                       SEC_3    3000
MS 80      80                       SEC 4    4000
MS_90      90                       SEC_5    5000
```

```
SEC 6    6000                    SEC 15  15000
SEC 7    7000                    SEC 20  20000
SEC_8    8000                    SEC_30  30000
SEC_9    9000                    MIN_1   60000
SEC_10  10000
```

Hangok:

```
TONE C2      65                  TONE B4      494
  // Második oktáv C-hang        TONE C5      523
TONE_CS2     69                  TONE_CS5     554
  // Második oktáv C-félhang     TONE_D5      587
TONE D2      73                  TONE DS5     622
TONE DS2     78                  TONE E5      659
TONE E2      82                  TONE F5      698
TONE F2      87                  TONE FS5     740
TONE_FS2     92                  TONE_G5      784
TONE_G2      98                  TONE_GS5     831
TONE GS2     104                 TONE A5      880
TONE A2      110                 TONE AS5     932
TONE AS2     117                 TONE B5      988
TONE B2      123                 TONE C6     1047
TONE_C3      131                 TONE_CS6    1109
  // Harmadik oktáv C-hang       TONE_D6     1175
TONE CS3     139                 TONE DS6    1245
TONE D3      147                 TONE E6     1319
TONE DS3     156                 TONE F6     1397
TONE_E3      165                 TONE_FS6    1480
TONE_F3      175                 TONE_G6     1568
TONE FS3     185                 TONE GS6    1661
TONE G3      196                 TONE A6     1760
TONE GS3     208                 TONE AS6    1865
TONE A3      220                 TONE B6     1976
TONE_AS3     233                 TONE_C7     2093
TONE_B3      247                 TONE_CS7    2217
TONE C4      262                 TONE D7     2349
  // Negyedik oktáv C-hang       TONE DS7    2489
TONE CS4     277                 TONE E7     2637
TONE D4      294                 TONE F7     2794
TONE_DS4     311                 TONE_FS7    2960
TONE_E4      330                 TONE_G7     3136
TONE F4      349                 TONE GS7    3322
TONE FS4     370                 TONE A7     3520
TONE G4      392                 TONE AS7    3729
TONE GS4     415                 TONE_B7     3951
TONE_A4      440
TONE_AS4     466
```

Ütemek:

```
NOTE WHOLE   1000            // Egész hang
NOTE HALF    (NOTE WHOLE/2)  // Félhang
NOTE QUARTER (NOTE WHOLE/4)  // Negyed hang
NOTE EIGHT   (NOTE WHOLE/8)  // Nyolcad hang
NOTE_SIXTEEN (NOTE_WHOLE/16) // Tizenhatod hang
```

III.3.4.3. A kijelző programozása

Az EV3 tégla LCD (Liquid Crystal Display) folyadékkristályos képernyőjére a következő *ev3_lcd.h* és *ev3_lcd.c* modulokban (kell használni az `#include "C:\Apps\Bricx\API\ev3_lcd.h"`-t) lévő típusok és függvények segítségével írhatunk:

Az *ev3_lcd.h* típusai:

```
typedef byte IMGDATA;
typedef IMGDATA* IP;

typedef struct
{
  int X;
  int Y;
} LocationType;

typedef LocationType* PLocationType;

typedef enum
{
  ifRAW FB0,
  ifRAW BUF,
  ifXBM,
  ifP1,
  ifP4,
  ifBMP,
  ifPNG
} ImageFormat;
```

Az *ev3_lcd.h* függvényei:

```
bool LcdText(char color, short x, short y, char* text);
```

A `color` 0 vagy 1 lehet, 0 esetében fekete alapon fehér szöveget, 1 esetében fehér alapon fekete (normális) szöveget ír ki.

Az `x` és az `y` a szöveg kezdőhelye pixelekben megadva. A képernyő mérete 178×128 pixel, így a megadható `x` értékek a 0...177, az `y` értékek pedig a 0…127 tartományból lehetnek.

A `text` a kiírandó szöveg.

A függvény **`false`** (hamis) értéket térít vissza, ha a képernyő nem volt inicializálva.

A szöveg betűtípusát a következő függvénnyel lehet megváltoztatni:

```
bool LcdSelectFont(byte FontType);
```

A `FontType` a beállítani kívánt betűtípus a 26. táblázat alapján.

Kód	Konstans	Eredmény
0	FONTTYPE_NORMAL	0-FONT
1	FONTTYPE_SMALL	1-FONT
2	FONTTYPE_LARGE	2-FONT
3	FONTTYPE_TINY	3-FONT

26. táblázat: *Betűtípusok*

```
bool LcdInit();
```
A kiírás vagy rajzolás előtt a kijelzőt inicializálni kell.

```
bool LcdInitialized();
```
Visszatéríti, hogy a kijelző inicializálva volt-e vagy sem.

```
bool LcdExit();
```
A használat után lezárjuk a kijelzőt.

```
void LcdRefresh();
```
Frissíti a kijelzőt.

```
void LcdSetAutoRefresh(bool bOn);
```
Beállítja a kijelző automatikus frissítését, ha a `bOn` **`true`**.

```
bool LcdUpdate();
```
Frissíti a kijelzőt.

```
bool LcdClean();
```
Letörli a kijelzőt.

```
void LcdClearDisplay();
```
Letörli a kijelzőt még akkor is, ha nem volt inicializálva.

```
bool LcdScroll(short y);
```
Függőlegesen görgeti a kijelzőt.

```
byte* LcdGetDisplay();
```
Visszatéríti a kijelző azonosítóját.

```
byte* LcdGetFrameBuffer();
```
Visszatéríti a kijelző memóriazónájának azonosítóját.

```
void LcdWriteDisplayToFile(char* filename, ImageFormat fmt);
```
Állományba menti a kijelző tartalmát. Állományformátumok (`ImageFormat`): `ifRAW_FB0`, `ifRAW_BUF`, `ifXBM`, `ifP1`, `ifP4`, `ifBMP`, `ifPNG`.

```
void LcdWriteFrameBufferToFile(char* filename, ImageFormat fmt);
```
Állományba menti a kijelző memóriazónájának tartalmát. Állományformátumok (`ImageFormat`): `ifRAW_FB0`, `ifRAW_BUF`, `ifXBM`, `ifP1`, `ifP4`, `ifBMP`, `ifPNG`.

```
bool LcdIcon(char Color, short X, short Y, char IconType,
char IconNum);
```
Az előredefiniált ikonokat rajzolja ki a megadott színnel (0 – fekete alapon fehér, 1 – fehér alapon fekete), a megadott `X` és `Y` koordinátákra. Az `IconType` 0 – `ICONTYPE_NORMAL`, 1 – `ICONTYPE_SMALL`, 2 – `ICONTYPE_LARGE`, 3 – `ICONTYPE_MENU`, valamint 4 – `ICONTYPE_ARROW` lehet. Az `IconNum` egy egész szám, a kívánt ikon sorszáma.

```
bool LcdBmpFile(char Color, short X, short Y, char* Name);
bool LcdPicture(char Color, short X, short Y, IP pBitmap);
```
Képet rajzolhatunk ki a megadott színnel, a megadott pozícióktól kezdődően. Az első függvényben a képet tartalmazó BMP állomány nevét kell megadni szövegként, a második függvényben pedig a kép memóriabufferét kell megadni.

```
bool LcdFillWindow(char Color, short Y, short Y1);
```
Az `Y`-tól az `Y1`-ig terjedő sávot festi ki a megadott színnel.

```
char PointOutEx(int x, int y, unsigned long options);
char LineOutEx(int x1, int y1, int x2, int y2,
unsigned long options);
char RectOutEx(int x, int y, int width, int height,
unsigned long options);
char CircleOutEx(int x, int y, byte radius,
unsigned long options);
char EllipseOutEx(int x, int y, byte radiusX, byte radiusY,
unsigned long options);
```

Függvények segítségével grafikus objektumokat tudunk kirajzolni a tégla kijelzőjére. Mindegyik függvénynek megvan az `Ex` nélküli változata is, ekkor nem kell megadni az **`unsigned long`** `options` paramétert, pontosabban ezt a `DRAW_OPT_NORMAL`-nak veszi. Például: **`char`** `PointOut(int x, int y);`

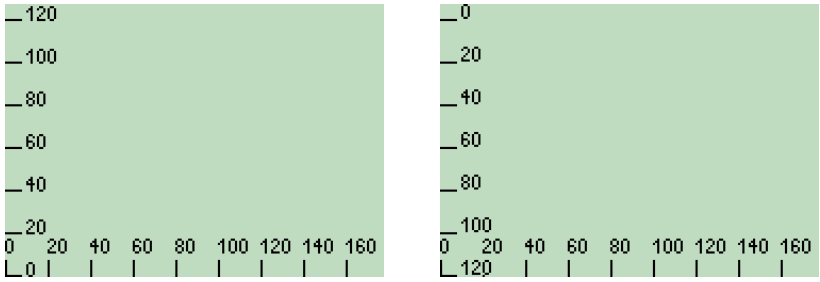
Tárgyaljuk először az `options` paramétert, amely a kirajzolás módját adja meg. Ezt a tárgyalást a 27. táblázat foglalja össze.

`DRAW_OPT_NORMAL`	`0x0000`	Normális rajzolás.
`DRAW_OPT_CLEAR_WHOLE_SCREEN`	`0x0001`	Rajzolás előtt törli a teljes képernyőt.
`DRAW_OPT_CLEAR_EXCEPT_STATUS_SCREEN`	`0x0002`	Az állapotsoron kívül törli a teljes képernyőt rajzolás előtt.
`DRAW_OPT_CLEAR_PIXELS`	`0x0004`	Pixeleket töröl rajzolás közben (fehérben rajzol).
`DRAW_OPT_CLEAR`	`0x0004`	Pixeleket töröl rajzolás közben (fehérben rajzol).
`DRAW_OPT_INVERT`	`0x0004`	Invertálja a szöveget vagy a grafikát.
`DRAW_OPT_LOGICAL_COPY`	`0x0000`	A pixelek rajzolása közben logikai MÁSOLÁS műveletet (COPY) hajt végre.
`DRAW_OPT_LOGICAL_AND`	`0x0008`	A pixelek rajzolása közben logikai ÉS műveletet (AND) hajt végre.
`DRAW_OPT_LOGICAL_OR`	`0x0010`	A pixelek rajzolása közben logikai VAGY műveletet (OR) hajt végre.
`DRAW_OPT_LOGICAL_XOR`	`0x0018`	A pixelek rajzolása közben logikai XOR műveletet hajt végre.
`DRAW_OPT_FILL_SHAPE`	`0x0020`	Kitölti az alakzatot (téglalap, kör, ellipszis, sokszög).
`DRAW_OPT_CLEAR_SCREEN_MODES`	`0x0003`	Bit maszk a képernyőtörlés módnak.
`DRAW_OPT_LOGICAL_OPERATIONS`	`0x0018`	Bit maszk a logikai műveletek számára.
`DRAW_OPT_POLYGON_POLYLINE`	`0x0400`	Nem zárja be a sokszöget, hanem törtvonalat rajzol.
`DRAW_OPT_CLEAR_LINE`	`0x0800`	Szöveg kirajzolása előtt törli a teljes sort.
`DRAW_OPT_CLEAR_EOL`	`0x1000`	Szöveg kirajzolása után törli a szöveg utáni teljes sort.

27. táblázat: *Az* options *paraméter*

Az első függvény egy adott `x`, `y` koordinátájú pixelt rajzol ki, a második egy vonalat, a harmadik egy téglalapot, a negyedik egy `x`, `y` középpontú, `radius` sugarú kört rajzol ki, az ötödik pedig egy ellipszist, amelynek *x* sugara a `radiusX`, *y* sugara pedig a `radiusY`.

A 156. ábra a kijelző parancsainak grafikus és szöveges koordinátarendszerét mutatja be. Amint láthatjuk, a két koordinátarendszer *y* koordinátája egymás fordítottjai. Tehát, ha az `LcdText` függvényt használjuk, akkor a 156. b) ábrának megfelelően kell beírnunk az *x* és *y* koordinátákat.

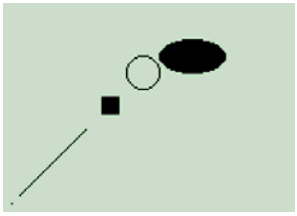


156. ábra: *a) grafikus és b) szöveges koordináták*

A 156. ábrát megvalósító program a következő:

```
1.  #include <stdio.h>
2.   #include <unistd.h>
3.   #include "c:\APPS\Bricx\API\ev3 lcd.h"
4.   #include "c:\APPS\Bricx\API\ev3 command.h"
5.
6.   int main()
7.   {
8.     LcdInit();
9.     LcdSelectFont(3);
10.    char s[3];
11.    int x;
12.    for(x = 0; x <= 170; x+= 20)
13.    {
14.       sprintf(s, "%d", x);
15.       LcdText(1, x, 110, s);
16.       LineOut(x, 0, x, 7);
17.    }
18.
19.    int y;
20.    for(y = 0; y <= 120; y += 20)
21.    {
22.       sprintf(s, "%d", y);
23.       //LcdText(1, 10, 127-y-6, s); // grafikus
24.       LcdText(1, 10, y+1, s);       // szoveges
25.       LineOut(0, y, 7, y);
26.    }
27.    Wait(SEC 1);
28.    LcdExit();
29.    return 0;
30.  }
```

Az alábbi program grafikus alakzatokat rajzol ki a tégla kijelzőjére úgy, ahogy a 157. ábrán látható.



157. ábra
A grafikus kijelző ábrái

```
1.  #include <stdio.h>
2.  #include <unistd.h>
3.  #include
    "c:\APPS\Bricx\API\ev3 lcd.h"
4.  #include
    "c:\APPS\Bricx\API\ev3 command.h"
5.
6.  int main()
7.  {
8.    LcdInit();
9.    PointOut(5, 5);
10.   LineOut(10, 10, 50, 50);
11.   RectOutEx(60, 60, 10, 10, DRAW OPT FILL SHAPE);
12.   CircleOut(85, 85, 10);
13.   EllipseOutEx(115, 95, 20, 10, DRAW OPT FILL SHAPE);
14.   Wait(SEC 1);
15.   LcdExit();
16.   return 0;
17. }
```

A következő program kirajzolja a LEGO tégla összes ikonját, ahogy az a 158. ábrán látható:

```
18. #include <stdio.h>
1.  #include <unistd.h>
2.  #include "c:\APPS\Bricx\API\ev3 lcd.h"
3.  #include "c:\APPS\Bricx\API\ev3 command.h"
4.
5.  int main()
6.  {
7.    LcdInit();
8.    int i;
9.    for (i = 0; i < 35; i++)
10.     LcdIcon(1, 0+((i % 7)*24), 5+((i / 7) * 20),
11.     ICONTYPE_NORMAL, i);
12.   Wait(SEC_1);
13.   LcdExit();
14.   return 0;
15. }
```

158. ábra: *Ikonok*

Kovács Lehel István