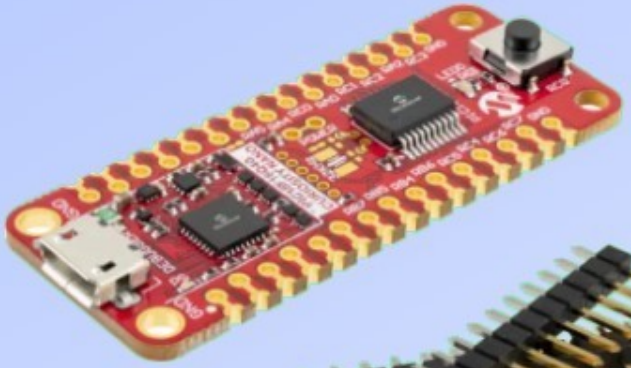
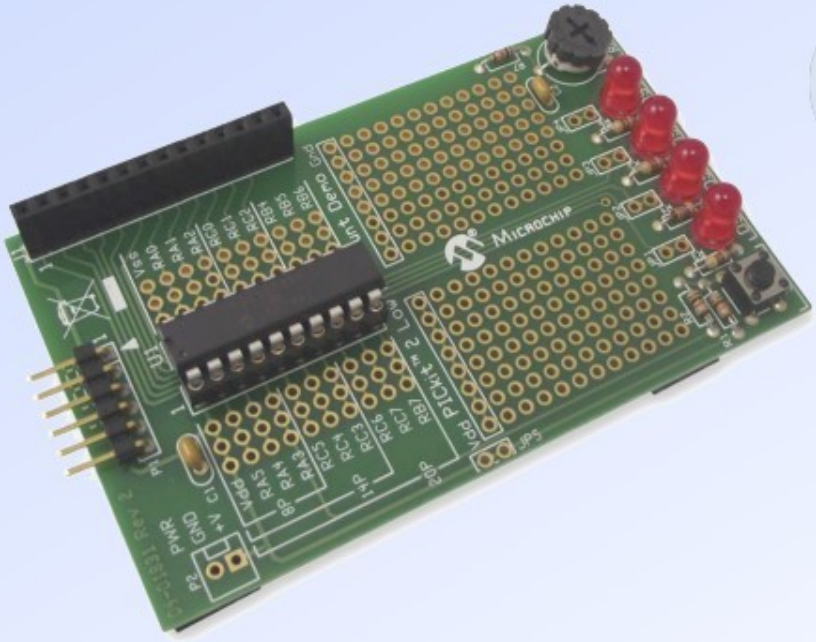


MICROCHIP

PIC mikrovezérlők

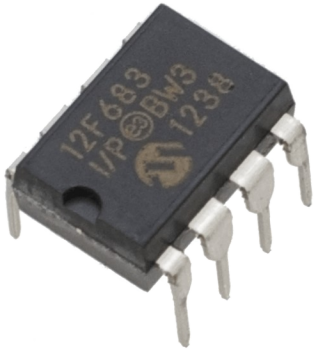
3. rész



Felhasznált és ajánlott irodalom

- **Milan Verle:** PIC MicrocontrollersProgramming in Assembly
- **Microchip:** PICmicro Mid-Range MCU Family Reference Manual
- **T&T:** Közepes teljesítményű PIC mikrovezérlők Felhasználói Kézikönyv
- **SimulIDE Community:** SimulIDE Tutorials
- **The Jallib Team:**
  - Have fun with PIC microcontrollers, Jal v2 and Jallib
  - Jal v2 Compiler Documentation

Adatlapok:



- **PIC12F683** adatlap és termékinfo
- **MCP41xxx** SPI Digital Potentiometer adatlap
- **Microchip:** PICkit2 programmer User's Guide
- **Icircuit Technologies:** iCP02v2 USB PIC/EEPROM programmer manual

Ismerkedés a JAL programnyelvvel

Eljárások és függvények

- Az Algolhoz és a Pascalhoz hasonlóan a **JAL** nyelvben is megkülönböztetjük az **eljárást** (procedure), ami elvégez egy műveletsort és a **függvényt** (function) ami egy számolás vagy egyéb tevékenység végén visszaad egy eredményt

```
PROCEDURE identifier [ '(' [VOLATILE] type { IN | OUT |
  IN OUT } identifier2 [',' ...] ')' IS
    statement_block
END PROCEDURE
```

```
FUNCTION identifier [ '(' [VOLATILE] type { IN | OUT |
  IN OUT } identifier2 [',' ...] ')' RETURN type IS
    statement_block
END FUNCTION
```

- Az utasításblokkban a `RETURN` [*expr*] utasítás hatására a szubprogramból (függvény/eljárás) azonnal visszatérünk
- **Függvény** esetén kötelező, **eljárás** esetén pedig tilos a `RETURN` kulcsszó utáni *expr* kifejezés megadása

Paraméterátadás

- A függvények/eljárások paraméterei ***érték szerint*** kerülnek átadásra, ami nem mindig vezet a kívánt eredményre, mert a bemenő paraméterek csak belépéskor, a kimenő paraméterek pedig csak kilépéskor, egyszer lesznek kiolvasva, illetve felülírva
- A **VOLATILE** kulcsszó gondoskodik arról, hogy a paraméter minden hivatkozáskor újraolvasásra vagy írásra kerüljön
- Tömbök átadása kétféle módon történhet:
  - **Fix méret specifikációval** (érték szerinti átadással, s minden híváskor csak ugyanekkora méretű lehet!). Például:

```
PROCEDURE string_write (BYTE IN str[5]) IS...
```

  - **Méretmegadás nélkül** (ez cím szerinti hivatkozással adja át), ekkor különböző hívásoknál különböző méretű lehet a tömb. Például:

```
PROCEDURE string_write (BYTE IN str[]) IS…
```

    Az átadott tömb méretét a `COUNT` operátorral határozhatjuk meg

Példa függvény definiálására

- Az alábbi függvény az ***n*** paraméter négyzetgyökét számolja ki

```
FUNCTION square_root (WORD IN n) RETURN WORD IS
  WORD result = 1
  WORD ix = 1
  WHILE ix < n LOOP
    n = n - ix
    result = result + 1
    ix = ix + 2
  END WHILE
  RETURN result
END FUNCTION

xx = square_root(81)
```

- **A rekurzió** teljes mértékben támogatott, de erőforrás igényessége miatt lehetőleg kerülendő

Pszeudo változók

- A pszeudo változók olyan alprogramok, vagy alprogrampárok, amelyek úgy működnek, mintha változók lennének. Ha egy **azonosító'PUT** eljárás definiálva van, az azonosítónak történő értékadás az **azonosító'PUT** eljárást hívja meg. Hasonlóképpen, ha egy **azonosító'GET** függvény definiálva van, az azonosító értékének minden lekérdezésekor a függvény implicit hívása történik
- Ha mind a **'GET**, mind a **'PUT** alprogram definiálva van, a **'PUT** paramétertípusának meg kell egyeznie a **'GET** típusával
- Példa az előző előadásban használ **serial_software.jal** könyvtárból

```
procedure serial_sw_data'put(byte in data) is
   serial_sw_write(data)  -- usage example: serial_sw_data = 0x33
end procedure

function serial_sw_data'get() return byte is
   var byte data
   serial_sw_read_wait(data) -- usage example: char = serial_sw_data
   return data
end function
```

Az spi_master_sw.jal programkönyvtár

- Ez a könyvtár szoftveresen emulálja az SPI adatküldést/fogadást
- Definiálnunk kell az alábbiakat, és be kell állítani az irányt:

```
alias spi_master_sw_sdi              is pin_sdi           -- adatbemenet
alias spi_master_sw_sdi_direction    is pin_sdi_direction
alias spi_master_sw_sdo              is pin_sdo           -- adatkimenet
alias spi_master_sw_sdo_direction    is pin_sdo_direction
alias spi_master_sw_sck              is pin_sck           -- órajel
alias spi_master_sw_sck_direction    is pin_sck_direction
spi_master_sw_sdi_direction = input    -- spi adatbemenet
spi_master_sw_sdo_direction = output   -- spi adatkimenet
spi_master_sw_sck_direction = output   -- spi kimenő órajel
```

- Becsatoljuk és inicializáljuk a könyvtárat

```
include spi_master_sw
spi_init(SPI_MODE_00)      -- spi inicializálás, üzemmód választás
```

- Eljárások és függvények:
  spi_master_sw'put(byte in data) – adat kiküldés
  spi_master_sw_exchange(byte in data) return byte – adatcsere
  spi_master_sw'get() return byte – adat beolvasása
- Ha szükséges, a $\overline{CS}$, vagy más névre hallgató kiválasztójelet szoftveres GPIO kezeléssel nekünk kell megoldani

spi_master_sw.jal (részletek)

```
53  --------------------------------------------------------------------------------
54  -- exchange data on the spi bus.
55  --------------------------------------------------------------------------------
56  function spi_master_sw_exchange(byte in data) return byte is
57     var bit bit_out at data : 7
58     var bit bit_in at data : 0
59     if (_spi_mode == SPI_MODE_00) | (_spi_mode == SPI_MODE_01) then
60        for 8 loop
61           spi_master_sw_sdo = bit_out
62           data = data << 1
63           spi_master_sw_sck = high
64           bit_in = spi_master_sw_sdi
65           spi_master_sw_sck = low
66        end loop
67     else ;(_spi_mode == SPI_MODE_11 then) | (_spi_mode == SPI_MODE_10) then
68        for 8 loop
69           spi_master_sw_sdo = bit_out
70           data = data << 1
71           spi_master_sw_sck = low
72           bit_in = spi_master_sw_sdi
73           spi_master_sw_sck = high
74        end loop
75     end if
76
77     return data
78  end function
```

spi_master_sw.jal (részletek)

```
80  ------------------------------------------------------------------------
81  -- put one byte of data onto the spi bus (pseudo variable)
82  ------------------------------------------------------------------------
83  procedure spi_master_sw'put(byte in data) is
84     var bit bit_out at data : 7
85     if (_spi_mode == SPI_MODE_00) | (_spi_mode == SPI_MODE_01) then
86        for 8 loop
87           spi_master_sw_sdo = bit_out
88           data = data << 1
89           spi_master_sw_sck = high
90           spi_master_sw_sck = low
91        end loop
92     else ;(_spi_mode == SPI_MODE_11 then) | (_spi_mode == SPI_MODE_10) then
93        for 8 loop
94           spi_master_sw_sdo = bit_out
95           data = data << 1
96           spi_master_sw_sck = low
97           spi_master_sw_sck = high
98        end loop
99     end if
100 end procedure
```

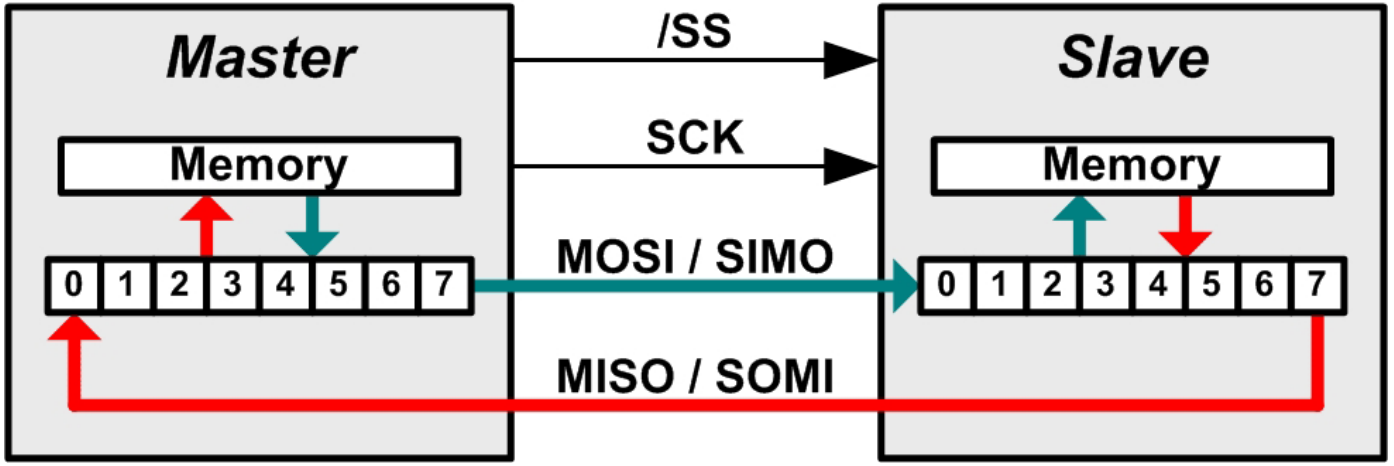
spi_master_sw.jal (részletek)

```
102  --------------------------------------------------------------------------
103  -- read one byte of data from the spi bus (pseudo variable)
104  --------------------------------------------------------------------------
105  function spi_master_sw'get() return byte is
106     var byte data
107
108     var bit bit_out at data : 0
109     if (_spi_mode == SPI_MODE_00) | (_spi_mode == SPI_MODE_01) then
110        for 8 loop
111           data = data << 1
112           spi_master_sw_sck = high
113           bit_out = spi_master_sw_sdi
114           spi_master_sw_sck = low
115        end loop
116     else ;(_spi_mode == SPI_MODE_11 then) | (_spi_mode == SPI_MODE_10) then
117        for 8 loop
118           data = data << 1
119           spi_master_sw_sck = low
120           bit_out = spi_master_sw_sdi
121           spi_master_sw_sck = high
122        end loop
123     end if
124     return data
125  end function
```

SPI kommunikáció PIC12F683 mikrovezérlővel

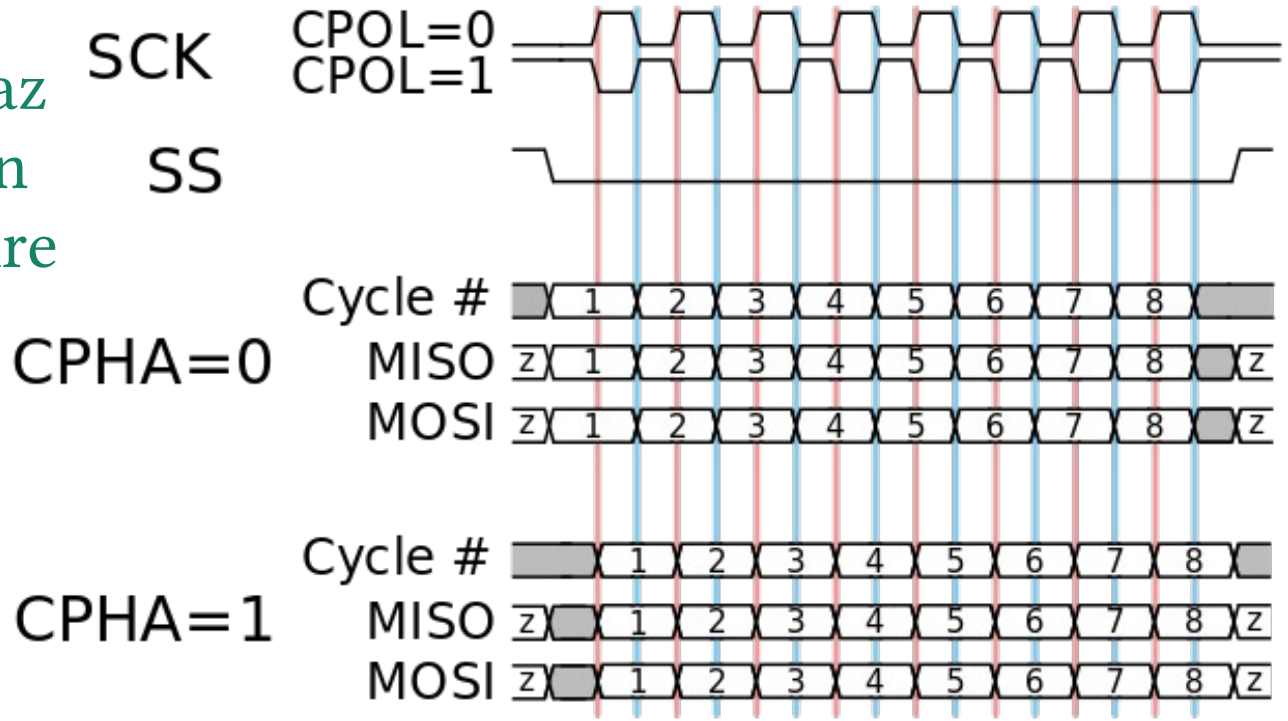
SPI adatküldés PIC12F683-ból

- Periféria bővítési lehetőségek:
  - Soros vezérlésű perifériabővítő IC-k használata (pl. MCP23S17, MCP23S08)
  - Shift regiszterek (75HC165, 74HC164, 74HC595)
  - LED kijelző vezérlő IC-k használata (TLC5940, MAX7219)
- SPI – Serial Peripheral Interface
- Jellemzői: kétirányú szinkron soros kommunikáció. Az órajel és az adatjel mellett kiválasztójel, vagy „betöltésjelző” jel is kell (/SS, /CS, vagy /LOAD)

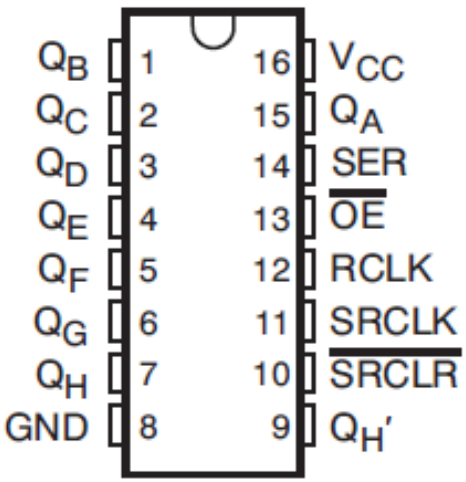


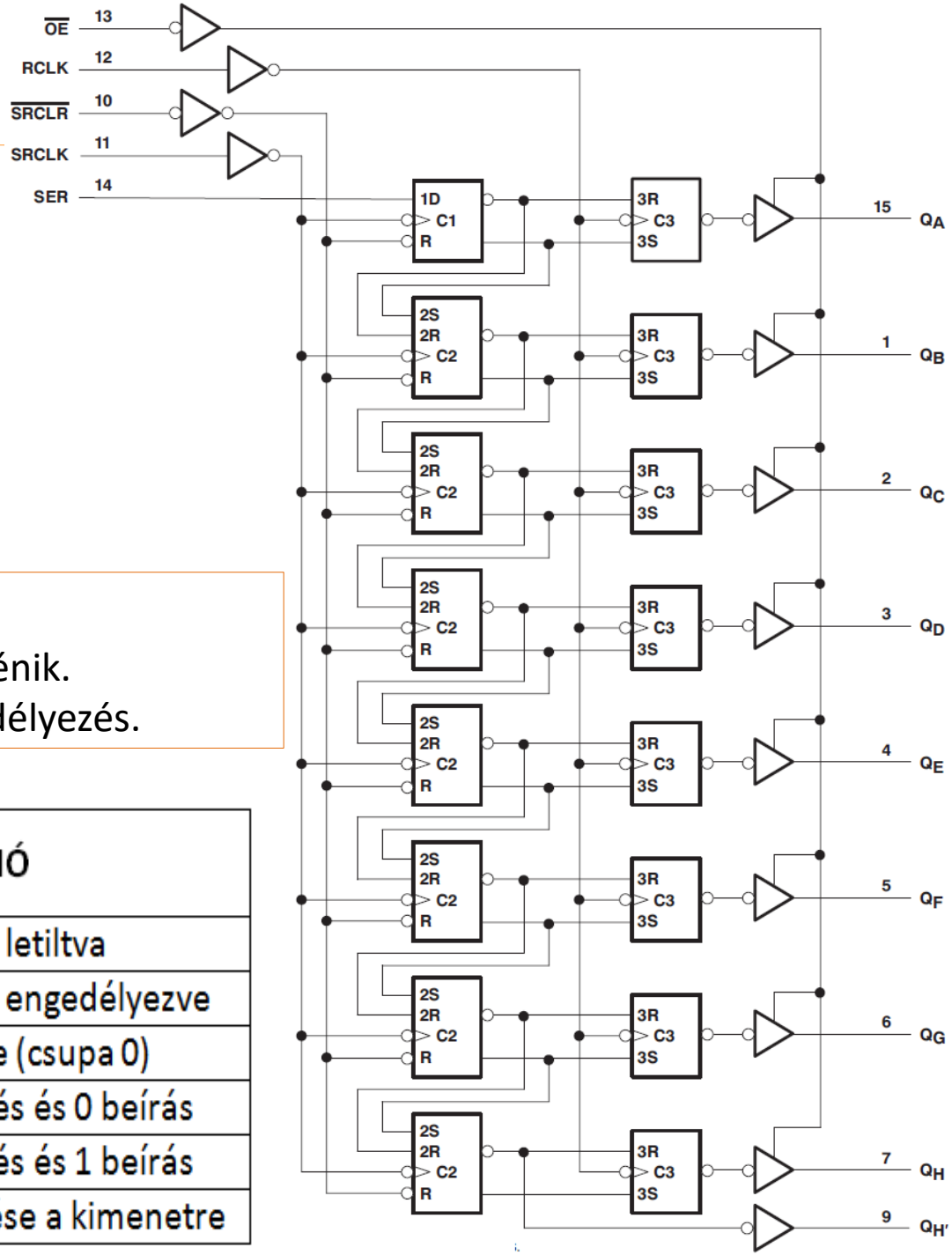
SPI üzemmódok

- A szinkronizáló órajel polaritása (CPOL) és fázisa (CPHA) szerint négy üzemmódot különböztetünk meg: ***mode 00, 01, 10*** és ***11***
  - CPOL=0, ha nyugalmi helyzetben az órajel alacsony szintű
  - CPOL=1, ha nyugalmi helyzetben az órajel alacsony szintű
  - CPHA=0, ha az adat az órajel felfutó élénél már rendelkezésre áll
  - CPHA=1, ha az adat csak az órajel lefutó élénél áll rendelkezésre
- Az egyes perifériák adatlapja közli, hogy az eszköz milyen módban tud működni (többnyire mode0,0 és mode1,1)



SN74HC595





A **74HC595** IC-k sorbaköthetőek ($QH'_n \rightarrow SER_{n+1}$)

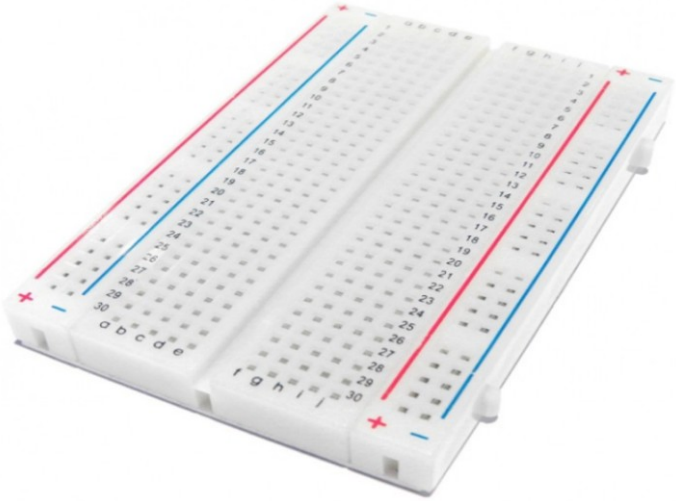
Az adatok áttöltése az **RCLK** jel felfutó élénél történik.

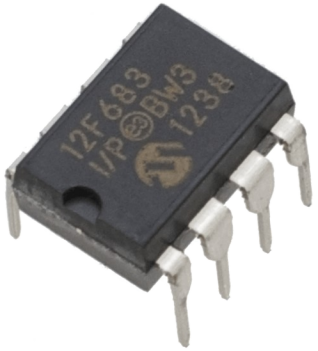
SRCLR = shift regiszter törlés, **OE** = kimenet engedélyezés.

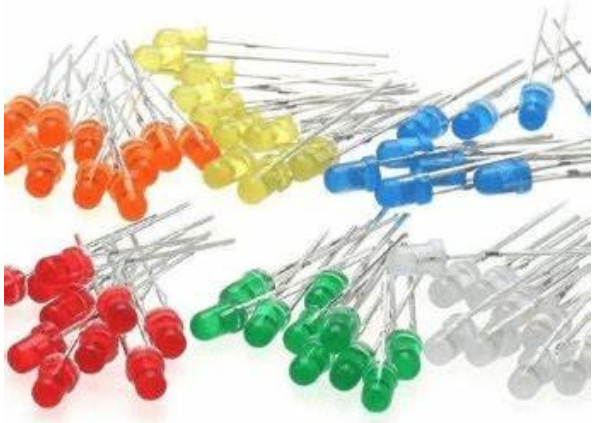
BEMENETEK					FUNKCIÓ
SER	SCLK	SCLR	RCLK	OE	
X	X	X	X	H	A Q_A – Q_H kimenetek letiltva
X	X	X	X	L	A Q_A – Q_H kimenetek engedélyezve
X	X	L	X	X	Shift regiszter törlése (csupa 0)
L	↑	H	X	X	Shift regiszter léptetés és 0 beírás
H	↑	H	X	X	Shift regiszter léptetés és 1 beírás
X	X	X	↑	X	Shift regiszter áttöltése a kimenetre

Ledek vezérlése 74HC595 shift regiszterrel

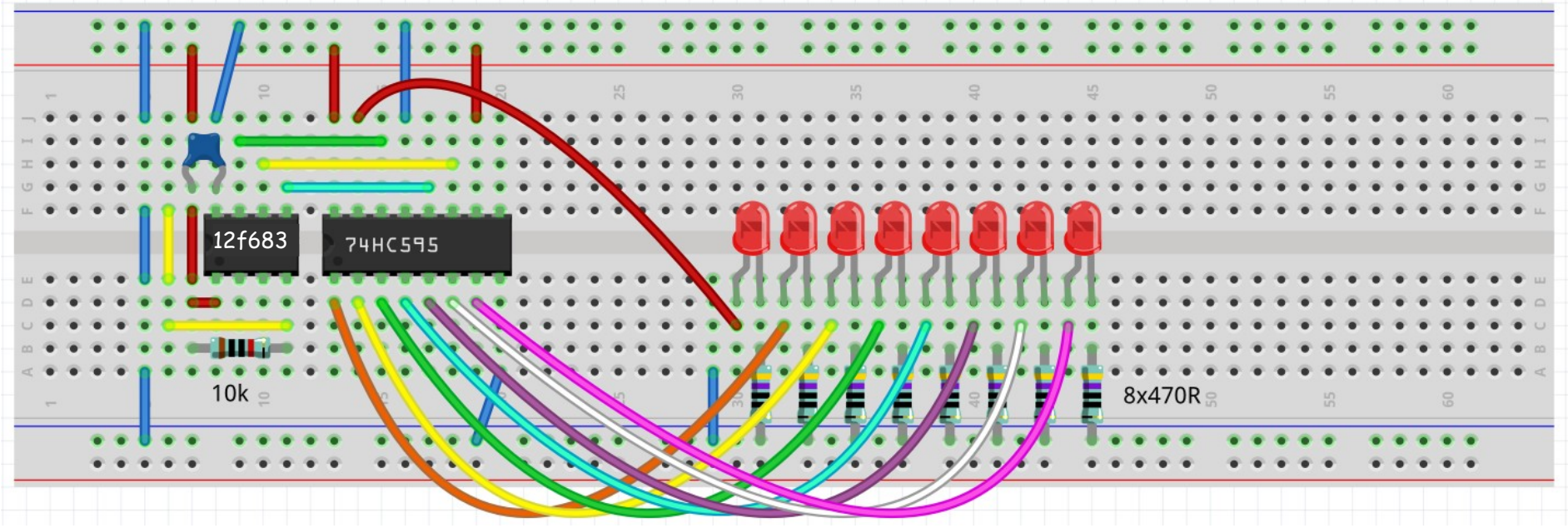
- A **PIC12F683** mikrovezérlővel egy vagy több **74HC595** shift regisztert vezérelve 8 vagy többször 8 LED-et kapcsolgathatunk

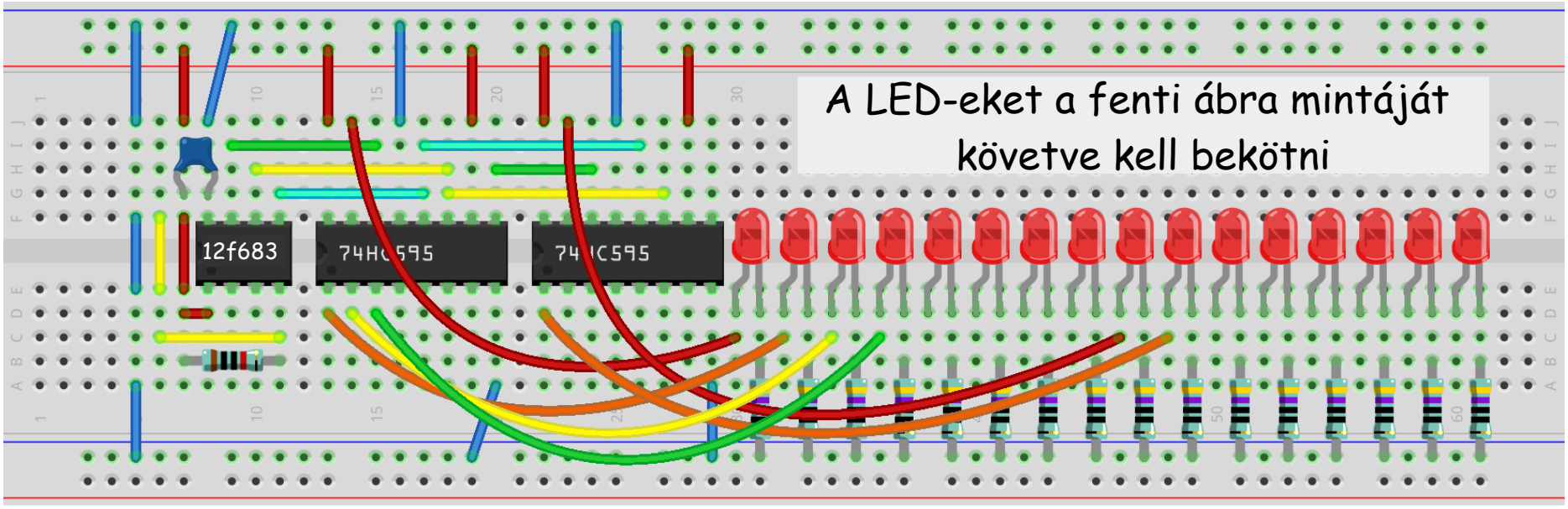






A kapcsolási vázlat





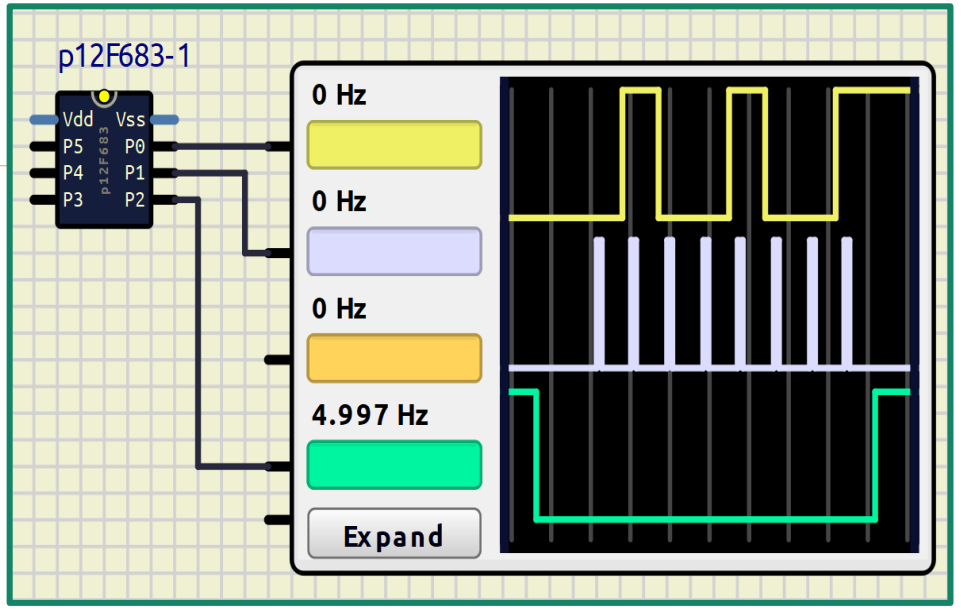
spi_demo.jal

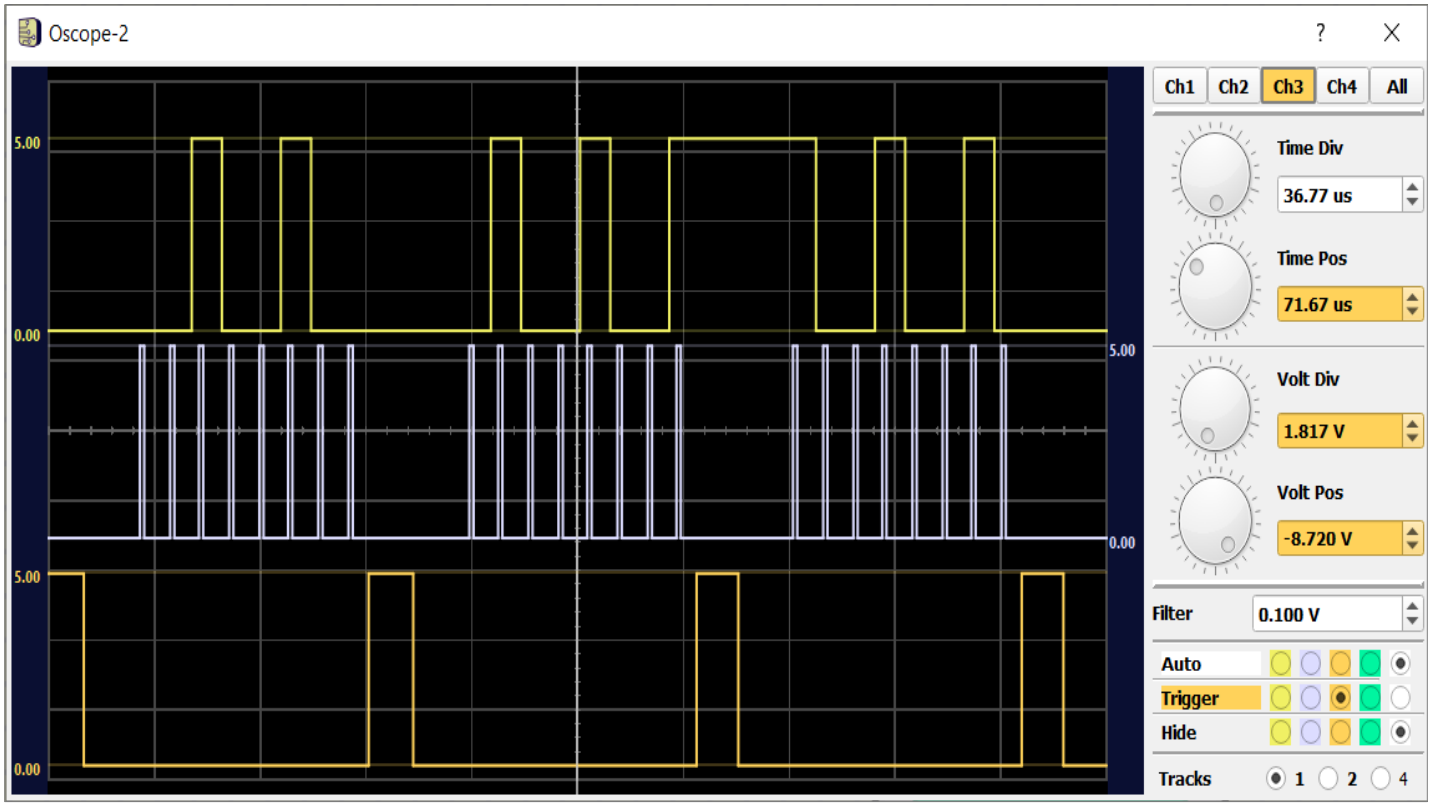
```
include 12f683                                  -- PIC12F683 az MCU
pragma target clock     8_000_000               -- oszcillátor frekvencia
pragma target OSC       INTOSC_NOCLKOUT         -- belső oszcillátor
pragma target WDT       DISABLED                -- watchdog letiltás
OSCCON_IRCF = 0b111                             -- 8 MHz
enable_digital_io()
const byte pin_dummy = 1                        -- virtual pin, reads 1
const byte data[] = {0x49, 0x92, 0x24}          -- sample data
alias spi_master_sw_sdi           is pin_dummy  -- dummy spi data input
alias spi_master_sw_sdo           is pin_A0     -- spi data out
alias spi_master_sw_sdo_direction is pin_A0_direction
alias spi_master_sw_sck           is pin_A1     -- spi clock
alias spi_master_sw_sck_direction is pin_A1_direction
alias spi_master_sw_sel           is pin_A2     -- spi select
alias spi_master_sw_sel_direction is pin_A2_direction
spi_master_sw_sdo_direction = output            -- spi output
spi_master_sw_sck_direction = output            -- spi clock
spi_master_sw_sel_direction = output            -- spi select
spi_master_sw_sel = high                        -- unselect
include spi_master_sw
spi_master_sw_init(SPI_MODE_00)                 -- init spi, choose mode
forever  loop
  for 3 using i loop
    spi_master_sw_sel = low                     -- chip select
    spi_master_sw = data[i]
    spi_master_sw_sel = high                    -- chip unselect
  end loop
end loop
```

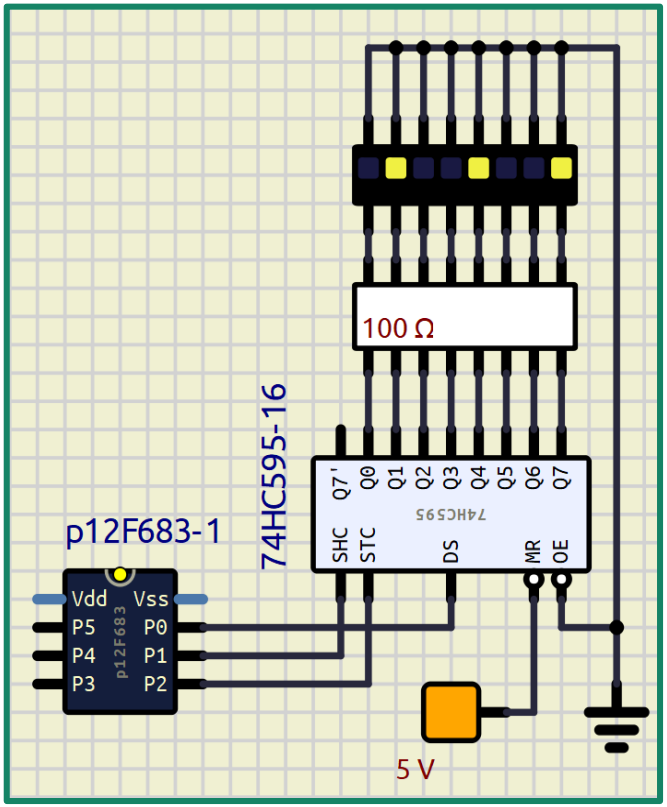
Késleltetést adjunk hozzá ízlés szerint

SimulIDE szimuláció

- Az `{0x49, 0x92, 0x24}` adatokkal a LED soron egy irányban vándorló fény hatást keltünk (háromfázisú jelek)
- A jobb alsó ábrán balról jobbra vándorolnak a fények…

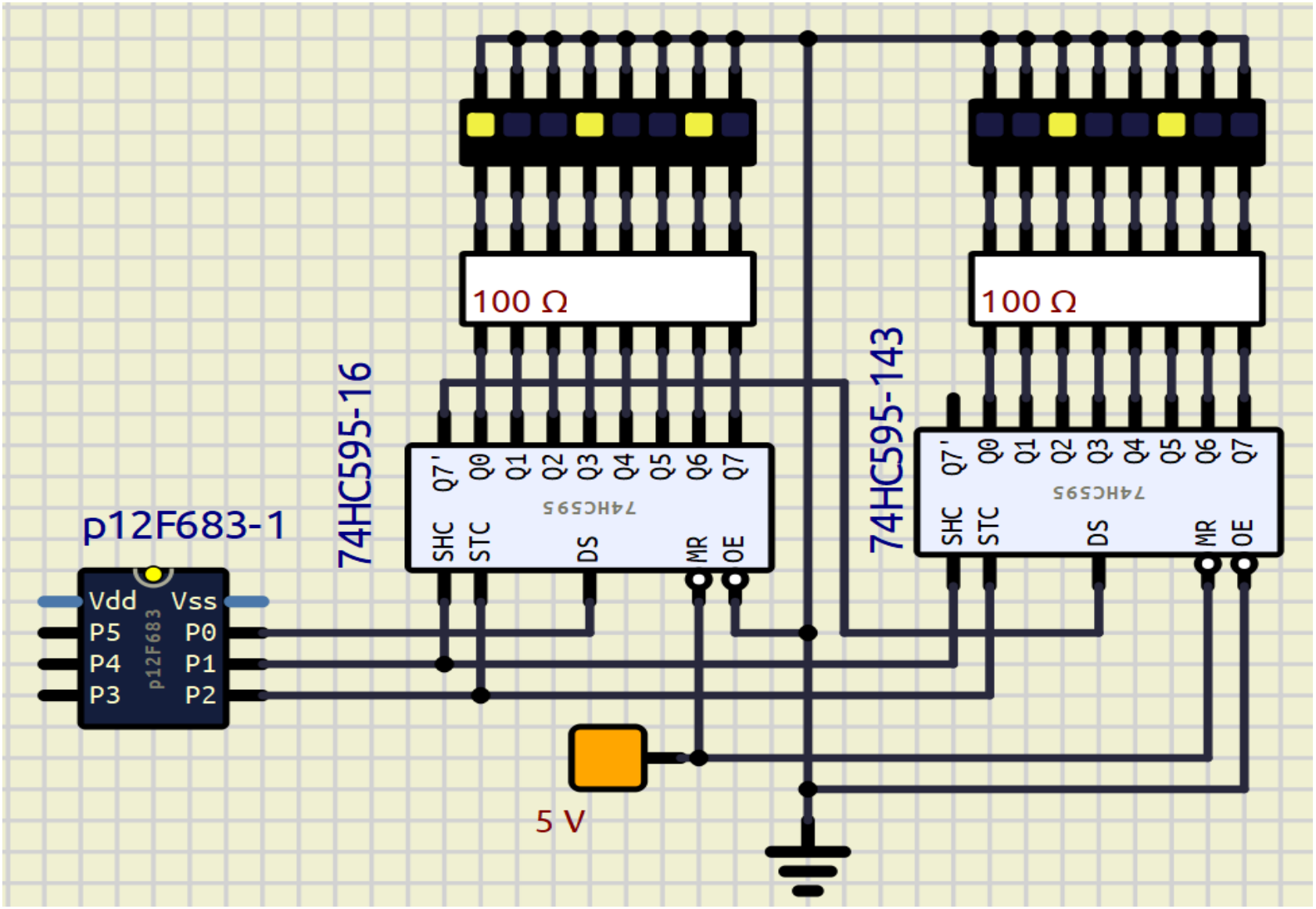






A kapcsolás tetszés szerint bővíthető...

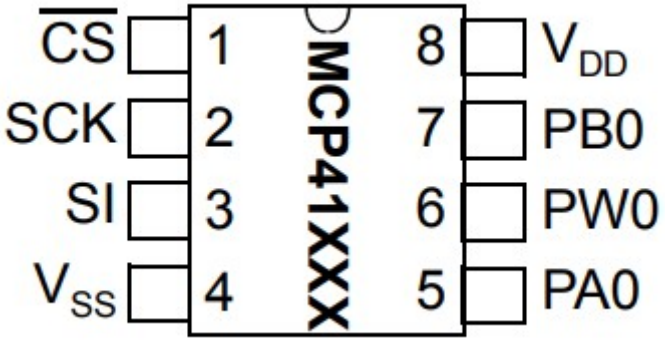
- A kiterjesztett kapcsolást is ugyanaz a szoftver működteti



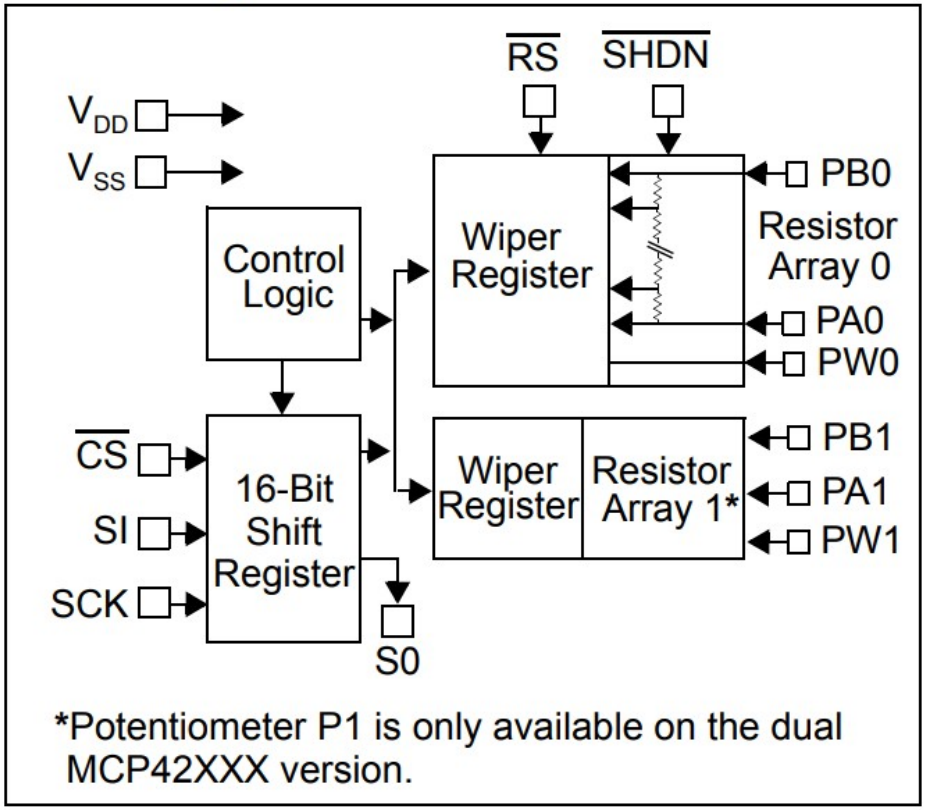
Digitális potméter vezérlése

Digitális potméter vezérlése

- A **Microchip MCP41xxx/MCP42xxx** termékcsalád SPI illesztővel ellátott digitális potmétereket egy, vagy két potmétert tartalmaznak, amelyek 8 bitesek (256 állású)
- Az általunk használt **MCP41050** csak egy potmétert tartalmaz és nincs $\overline{RS}$, $\overline{SHDN}$, ill. SO kivezetése
- A kiküldendő 16 bit első fele egy parancs, ami esetünkben **0x11,** a második fele pedig a 8 bites adat
- Adat = 0 esetén a **PW0** csúszka a potméter **PB0** végéhez zár



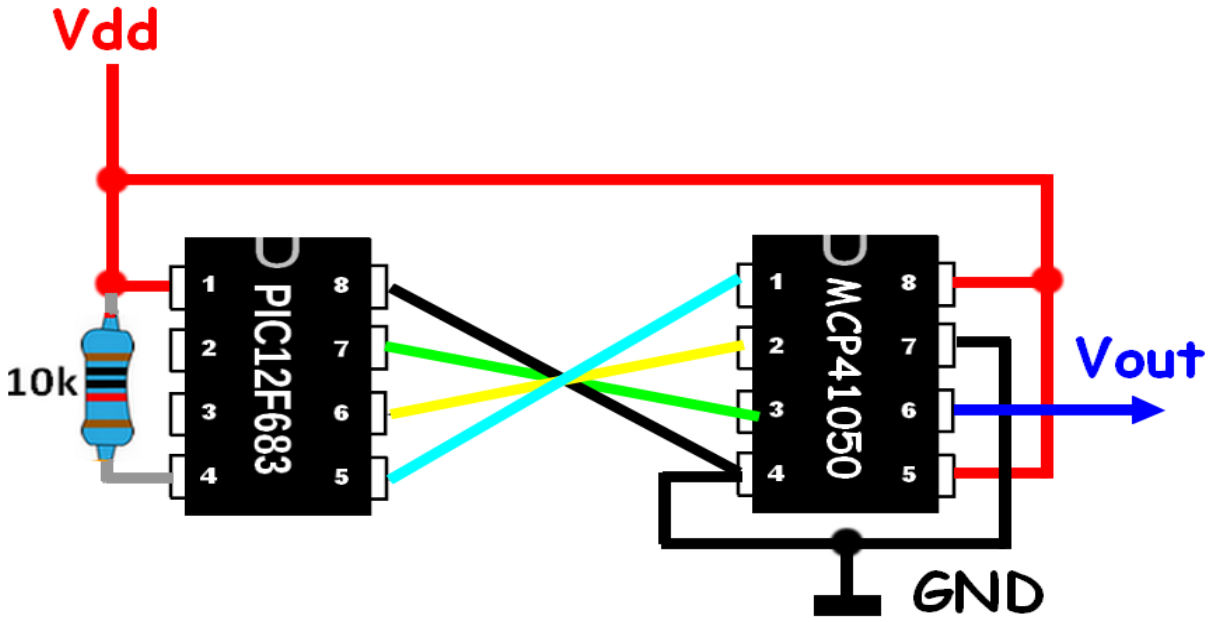
Block Diagram



Kapcsolási vázlat

- Az alábbi kapcsolás szerint a potméter az 5 V-os tápfeszültséget fogja leosztani
- Megjegyzés: Arra ügyeljünk, hogy az **MCP41050** nagy ellenállása (50 kΩ) miatt a **Vout** kimenet nem terhelhető!

PIC	MCP41050		Funkció
1	8	VCC	tápfeszültség
5	1	CS	SPI select
6	2	CLK	SPI clock
7	3	MOSI	SPI data
8	4	GND	GND

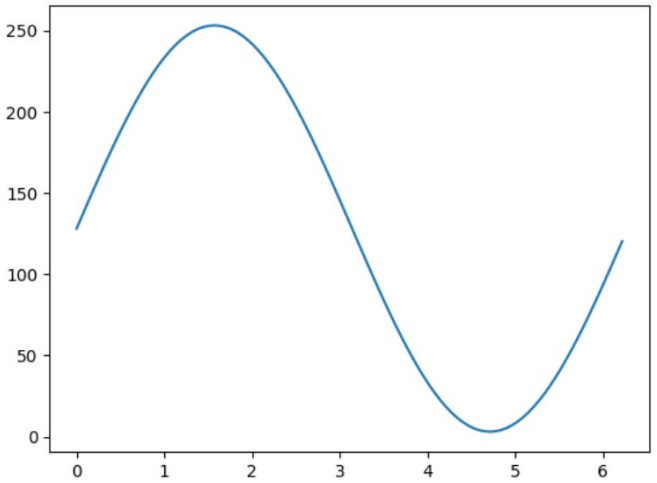


Készítsünk függvénytáblát!

- Az alábbi Python kóddal a szinusz függvény egy periódusát 100 mintára bontva számoljuk ki, a 8 bites (0 – 255) értékkészlet tartományra transzformálva
- A mintákat majd konstans tömbként tároljuk a **JAL** programban

```
In [19]: import numpy as np
         import matplotlib.pyplot as plt

In [22]: to = 2*np.pi
         step=to/100
         x=np.arange(0,to,step)
         y=125*np.sin(x)+128
         print(y.astype(int))
         plt.plot(x,y)
```




```
[128 135 143 151 159 166 174 181 188 194 201 207 213 219 224 229 233 237
 241 244 246 249 250 252 252 253 252 252 250 249 246 244 241 237 233 229
 224 219 213 207 201 194 188 181 174 166 159 151 143 135 127 120 112 104
  96  89  81  74  67  61  54  48  42  36  31  26  22  18  14  11   9   6
   5   3   3   3   3   3   5   6   9  11  14  18  22  26  31  36  42  48
  54  61  67  74  81  89  96 104 112 120]
```

spi_digipot.jal – 2/1.

```
include 12f683                                  -- PIC12F683 az MCU
pragma target clock     8_000_000               -- oszcillátor frekvencia
pragma target OSC       INTOSC_NOCLKOUT         -- belső oszcillátor
pragma target WDT       DISABLED                -- watchdog letiltás
OSCCON_IRCF = 0b111                             -- 8 MHz
enable_digital_io()
const byte pin_dummy = 1                        -- virtual pin, reads 1
const byte digipot_write = 0x11                 -- write to pot0
const byte data[] = {128, 135, 143, 151, 159, 166, 174, 181, 188, 194,
                     201, 207, 213, 219, 224, 229, 233, 237, 241, 244,
                     246, 249, 250, 252, 252, 253, 252, 252, 250, 249,
                     246, 244, 241, 237, 233, 229, 224, 219, 213, 207,
                     201, 194, 188, 181, 174, 166, 159, 151, 143, 135,
                     127, 120, 112, 104,  96,  89,  81,  74,  67,  61,
                      54,  48,  42,  36,  31,  26,  22,  18,  14,  11,
                       9,   6,   5,   3,   3,   3,   3,   3,   5,   6,
                       9,  11,  14,  18,  22,  26,  31,  36,  42,  48,
                      54,  61,  67,  74,  81,  89,  96, 104, 112, 120}
alias spi_master_sw_sdi           is pin_dummy  -- dummy spi data input
-- alias spi_master_sw_sdi_direction is pin_sdi_direction
alias spi_master_sw_sdo           is pin_A0     -- spi data out
alias spi_master_sw_sdo_direction is pin_A0_direction
alias spi_master_sw_sck           is pin_A1     -- spi clock
alias spi_master_sw_sck_direction is pin_A1_direction
```

spi_digipot.jal – 2/2.

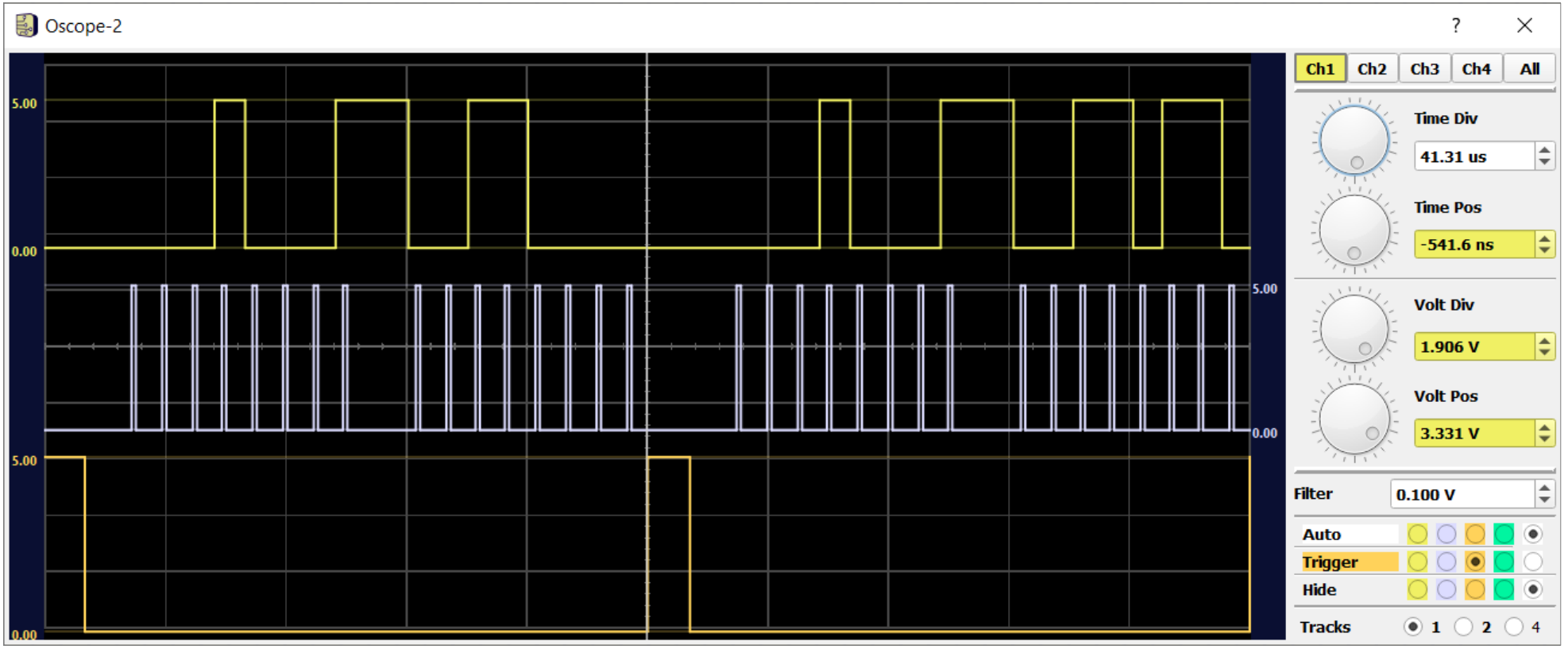
```
alias spi_master_sw_sel           is pin_A2      -- spi select
alias spi_master_sw_sel_direction is pin_A2_direction
-- spi_master_sw_sdi_direction = input          -- spi input
spi_master_sw_sdo_direction = output            -- spi output
spi_master_sw_sck_direction = output            -- spi clock
spi_master_sw_sel_direction = output            -- spi select
spi_master_sw_sel = high                        -- unselect
include spi_master_sw
spi_master_sw_init(SPI_MODE_00)                 -- init spi, choose mode

var byte i = 0
forever  loop
  for 100 using i loop
    spi_master_sw_sel = low                     -- chip select
    spi_master_sw = digipot_write               -- write command
    spi_master_sw = data[i]                     -- write data
    spi_master_sw_sel = high                    -- chip unselect
  end loop
end loop
```

- A 16 bites adatküldést egyszerűen, két bájt kiküldésével oldjuk meg, s a szinkronizáló órajel miatt nem okoz gondot, ha a két bitcsoport küldése között némi szünet van

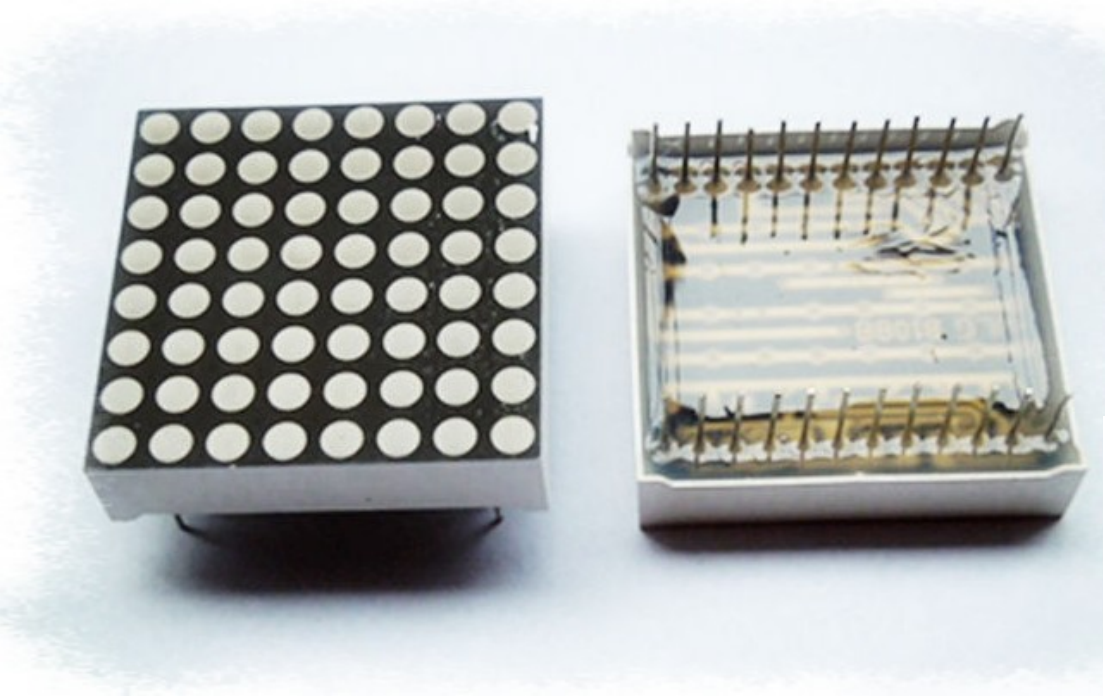
spi_digipot.jal szimuláció

- A szimulációt a **SimulIDE** programmal végeztük, s egy tipikus részlet az alábbi ábrán látható, ahol éppen a *48* (0x30) és *54* (0x36) értékek kiküldése látható
- Az adatminták kiírása kb. 5 kHz frekvenciával ismétlődik, tehát 100 mintával max. 50 Hz-es frekvenciájú jelet tudunk kelteni. Szinuszos jel esetén periódusonként 16 – 20 minta is elegendő lehet (f_{MAX} =*250-300 Hz*)



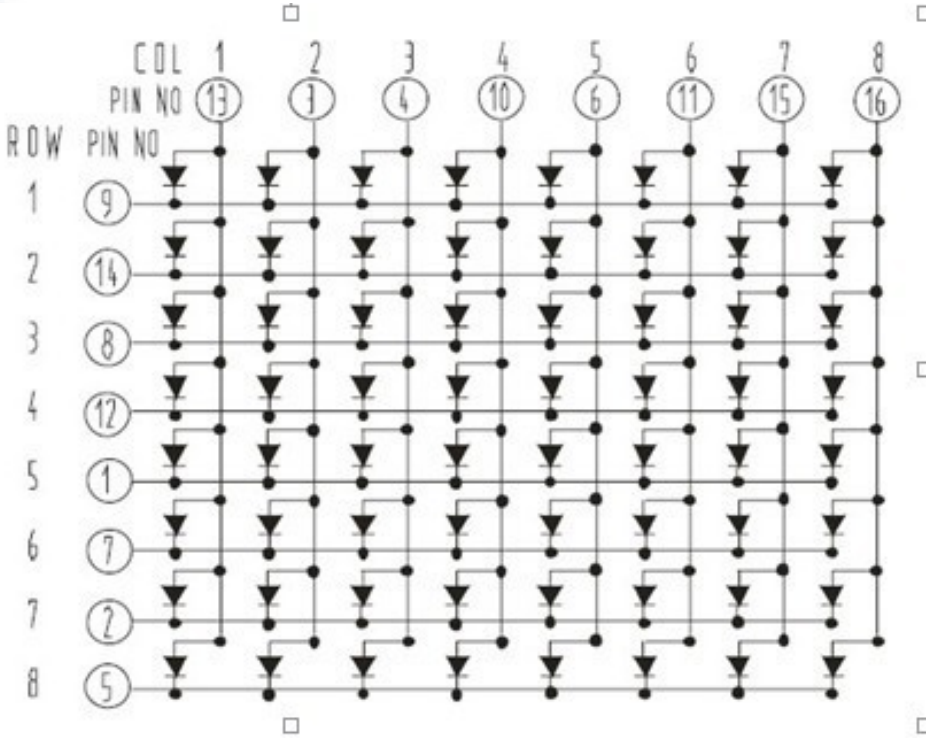
LED mátrix vezérlése

8x8 LED mátrix



3 mm-es piros LED-ek 8x8-as mátrixba szervezve

A **1088AS** típusnál a sorkiválasztó vonalak a katódokat közösítik



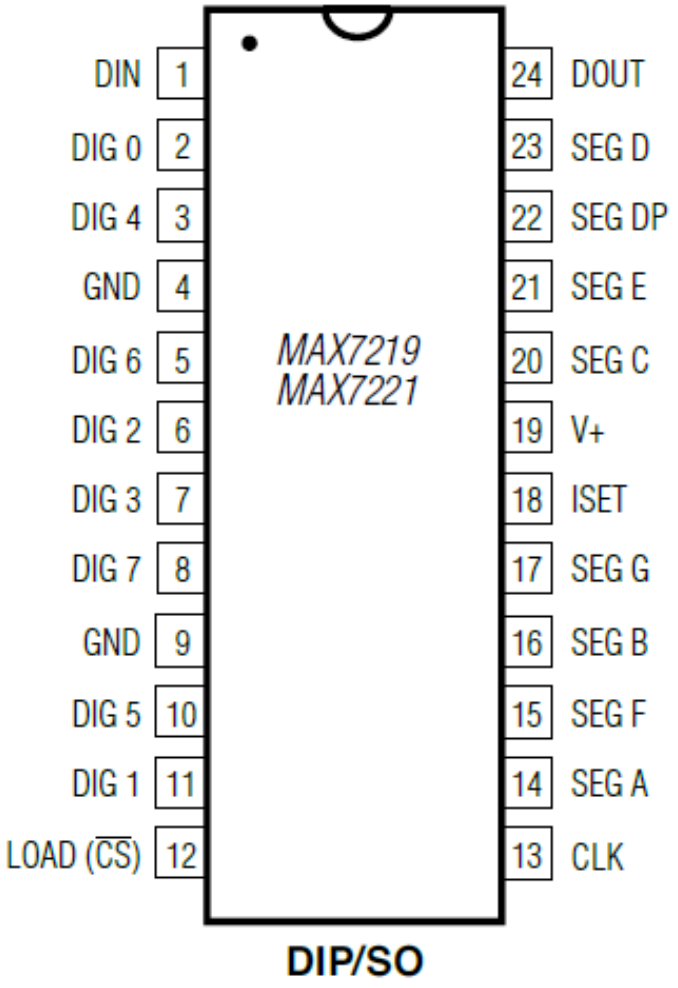
Multiplex kijelzés, egyidejűleg legfeljebb egy sor, vagy egy oszlop lehet aktív.

Kényelmes meghajtás:

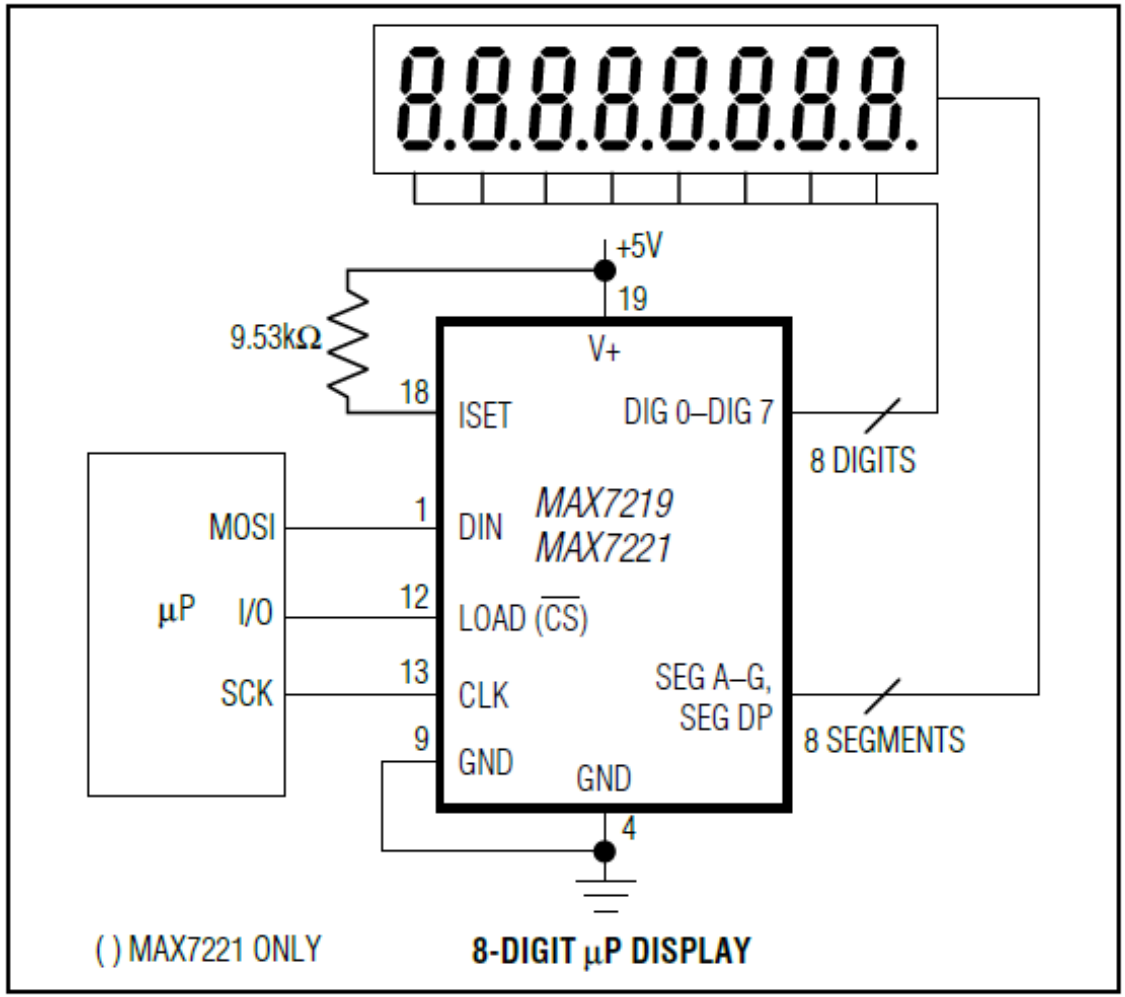
- 1 db **MAX7219**, vagy
- 2 db **74HC595** (+ meghajtó +áramkorlátozás)
- 1 db **MCP23017** (+ meghajtó +áramkorlátozás)

MAX7219

LED meghajtó IC, beépített áramkorlátozással. 7 szegmens kijelző esetén 1-8 db számjegy meghajtása (opcionálisan beépített dekódolással), vagy 8x8 LED mártix meghajtása. Az IC vezérlése SPI buszon történhet.

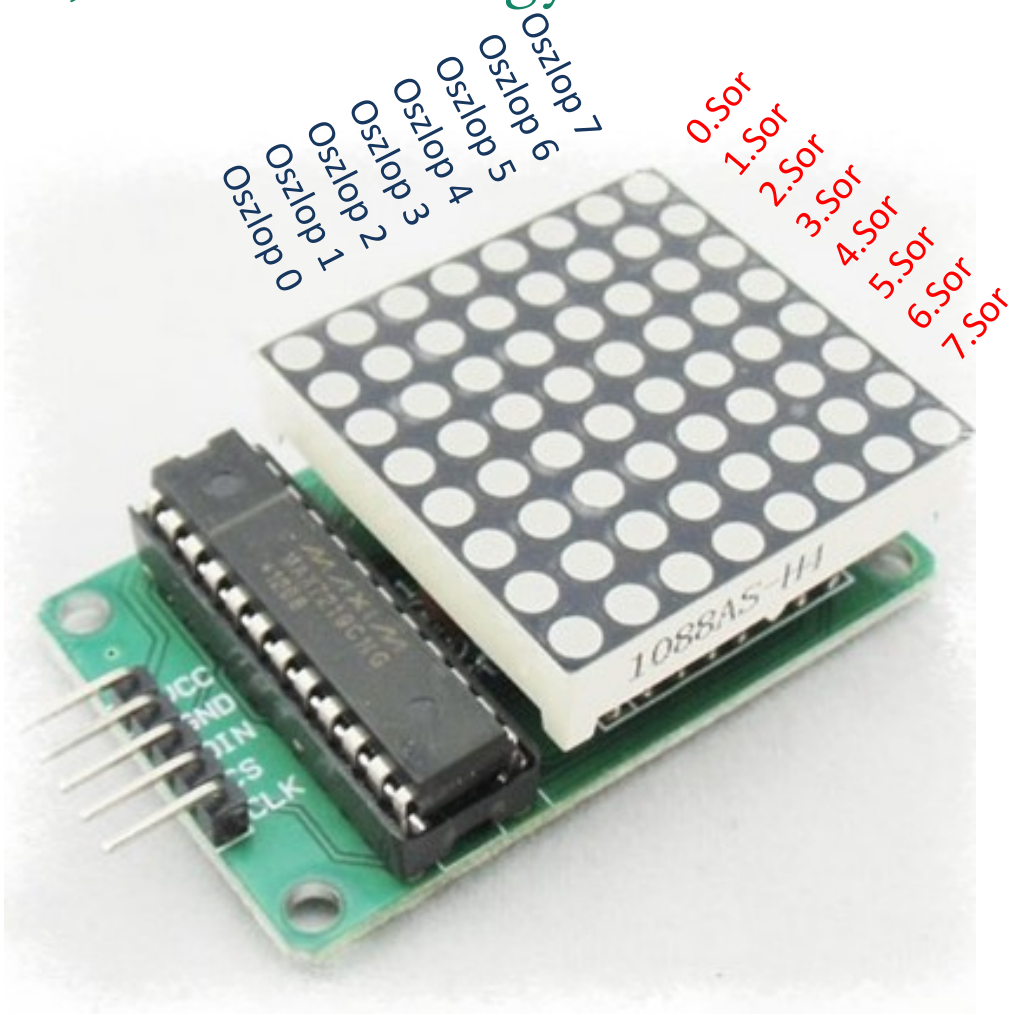


DIP/SO



Kijelző modul MAX7219 IC-vel

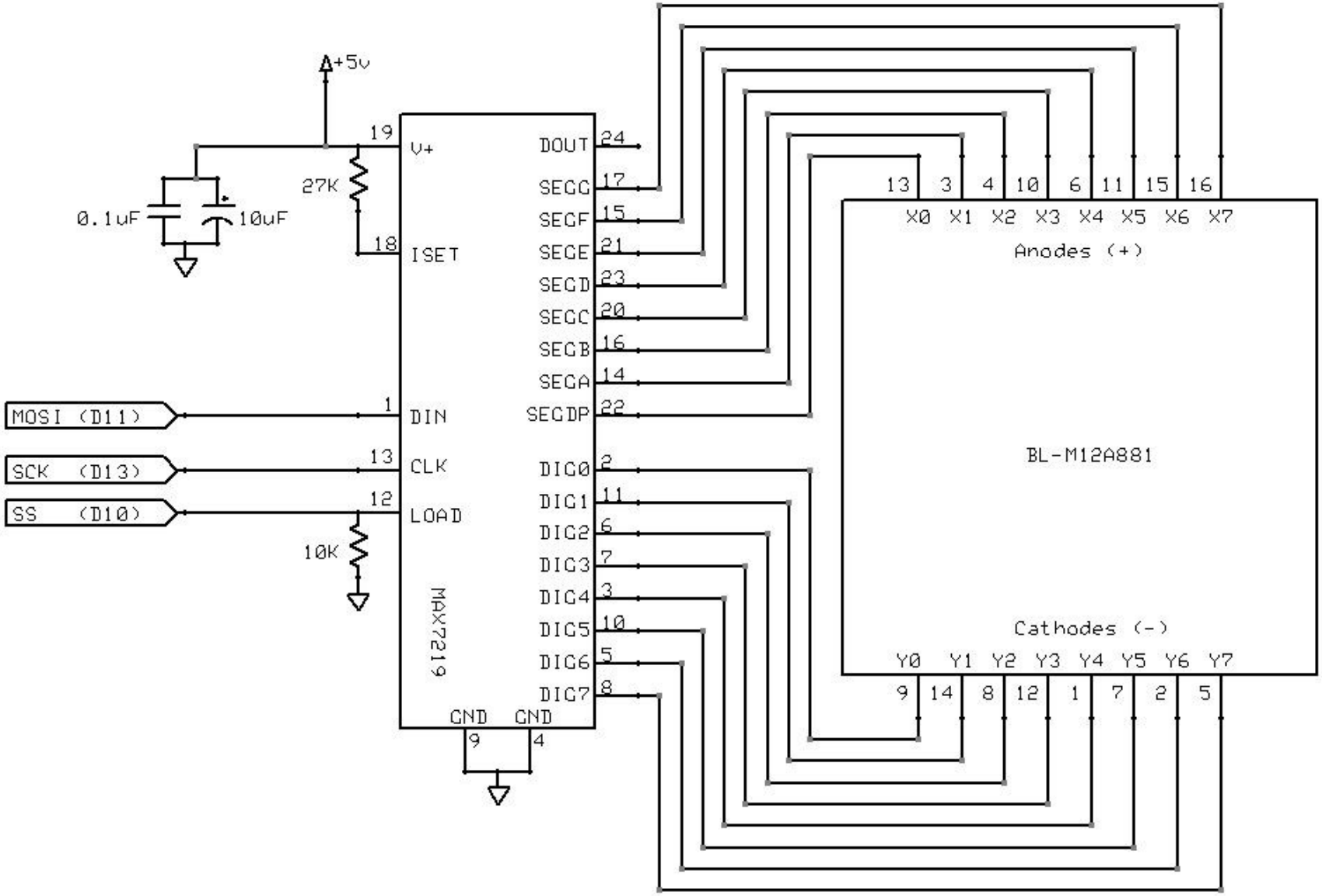
- Az E-bay kínálatában kapható, furatszerelt vagy felületszerelt kivitelben
  - 8x8 LED mátrix
  - MAX7219 vezérlő
  - Felfűzhető kivitel
  - Tápellátás: 3,5 – 5 V



Bemenetek	**Kimenetek**
1 VCC	1 VCC
2 GND	2 GND
3 DIN	3 DOUT
4 CS	4 CS
5 CLK	5 CLK

- **DIN/DOUT** – soros adat, **CLK** – szinkron órajel, **CS** – eszközkiválasztó jel, **VCC** – tápfeszültség, **GND** – a tápegység közös pontja („föld”)

Kapcsolási rajz



Technikai részletek az inicializáláshoz

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Adat

Nem használ adatot
DP a b c d e f g

0: no decode 1: decode

0 – 0xF

0 – 7

0: shutdown 1: normal mode

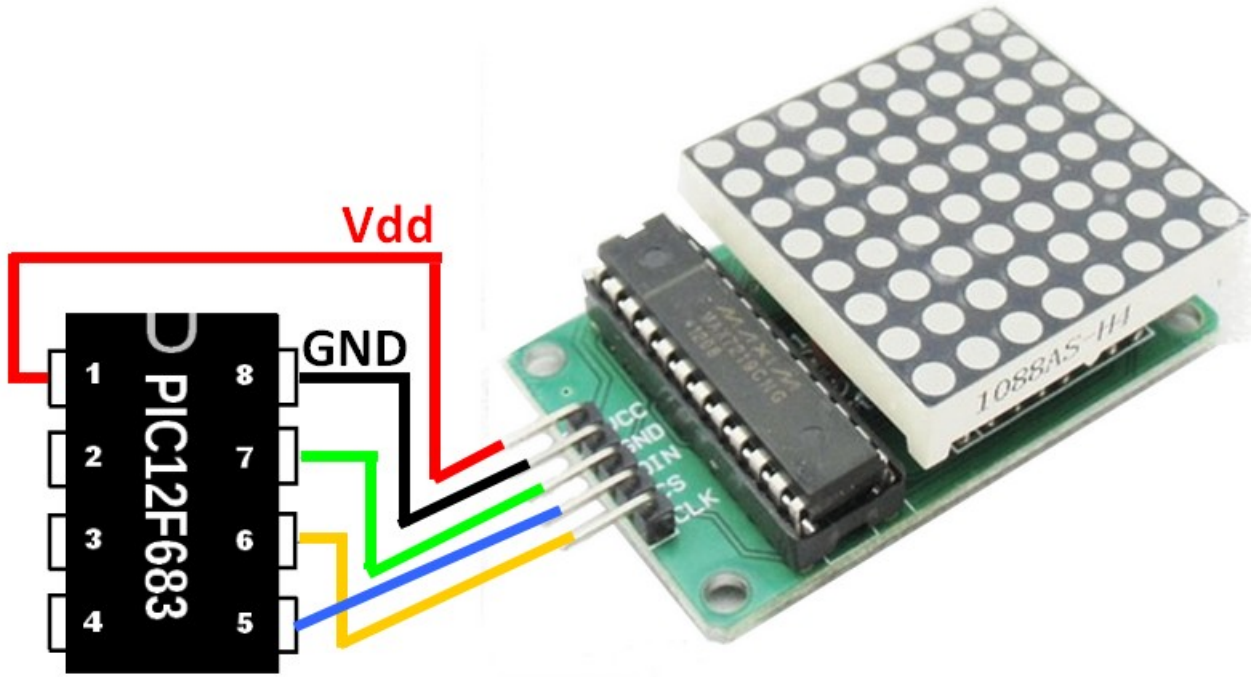
1: test mode 0: normal mode

Bekötési vázlat

- A **MAX7219** IC-vel vezérelt LED8x8 modult **PIC12F683** mikrovezérlővel vezéreljük, az alábbi bekötéssel
- A program két, 8x8-a rajzolatot (**I♥** és **HE**) jelenít meg, felváltva, 1 mp időközzel. A rajzmintákat konstans tömbként tároljuk

```
const byte led1[]= {0xFF,0x00,0x30,0x48,0x90,0x90,0x48,0x30} -- I♥
const byte led2[]= {0xFF,0x18,0x18,0xFF,0x00,0xF8,0xA8,0xA8} -- HE
```

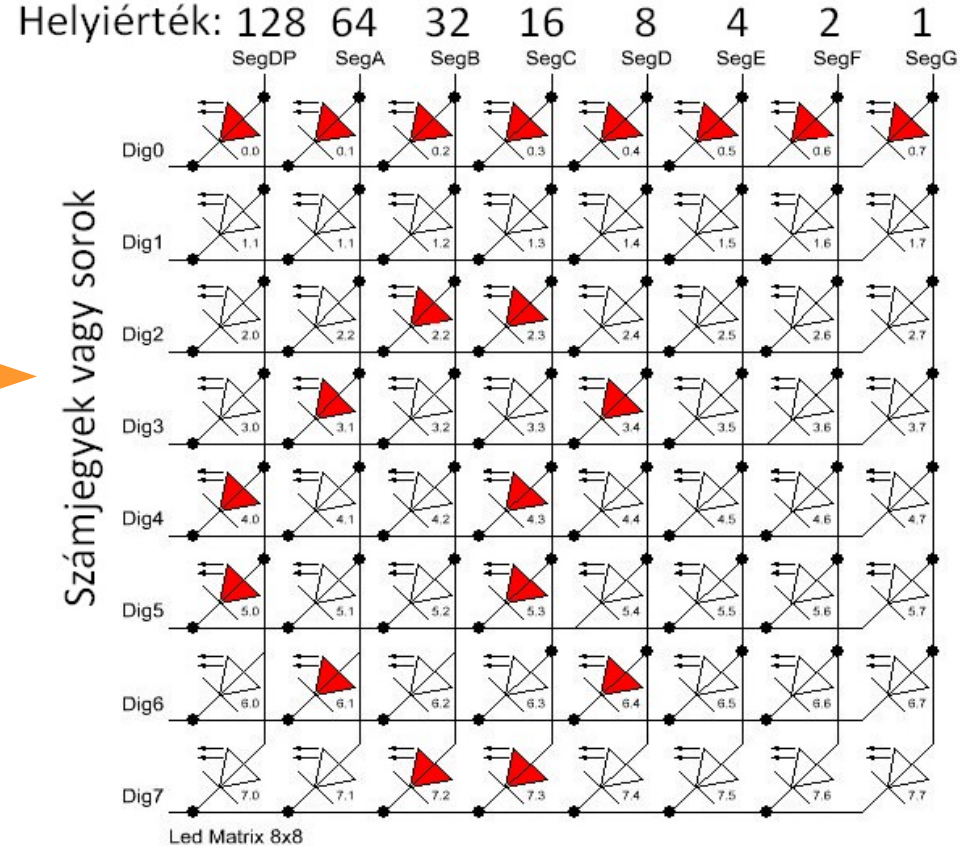
PIC	LED8x8 modul	
1	VCC	tápfeszültség
5	CS	SPI select
6	CLK	SPI clock
7	DIN	SPI data
8	GND	GND

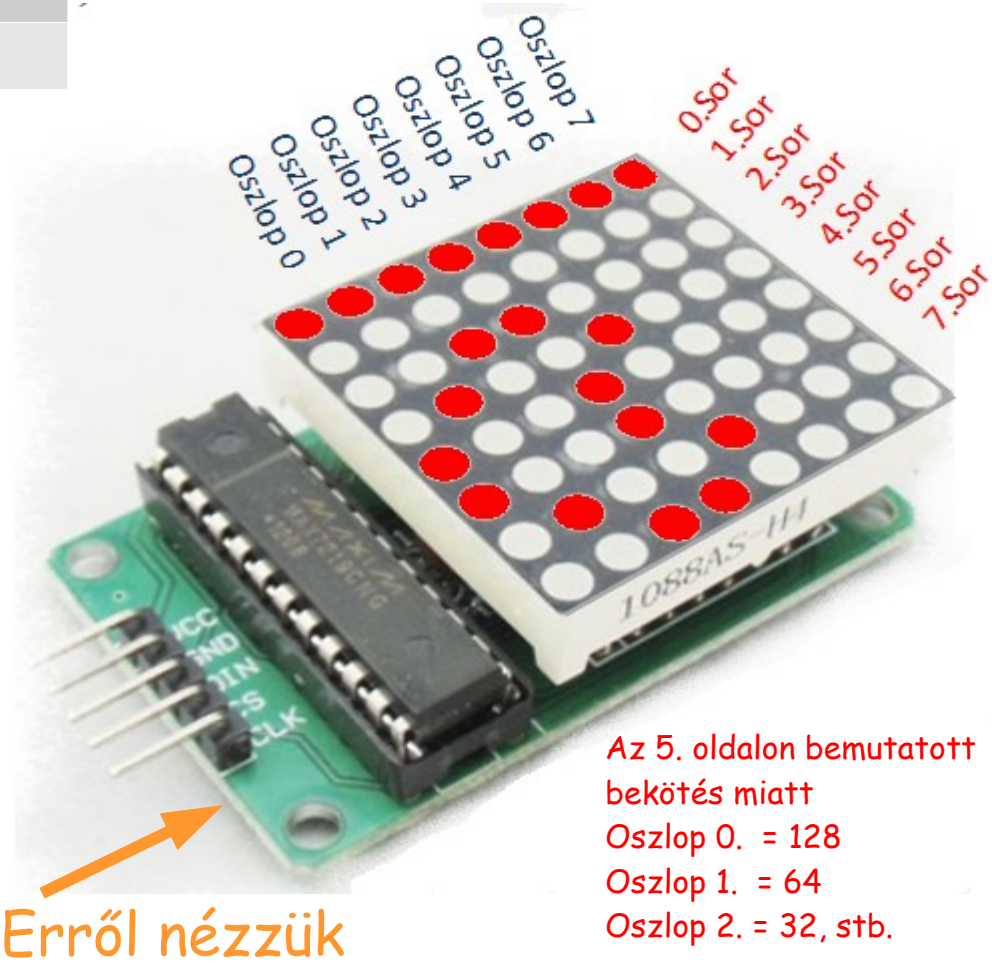


Technikai részletek a kiíráshoz

Sor	Bin	Hex	Sor	Bin	Hex
0	11111111	0xFF	4	10010000	0x90
1	00000000	0x00	5	10010000	0x90
2	00110000	0x30	6	01001000	0x48
3	01001000	0x48	7	00110000	0x30

```
const byte led1[]={
    0xFF, 0x00, 0x30, 0x48,
    0x90, 0x90, 0x48, 0x30
}   -- I♥
```





Az 5. oldalon bemutatott bekötés miatt
Oszlop 0. = 128
Oszlop 1. = 64
Oszlop 2. = 32, stb.

spi_max7219_led8x8.jal – 3/1.

```
include 12f683                                  -- PIC12F683 az MCU
include delay                                   -- késleltető függvények
pragma target clock    8_000_000                -- oszcillátor frekvencia
pragma target OSC      INTOSC_NOCLKOUT          -- belső oszcillátor
pragma target WDT      DISABLED                 -- watchdog letiltás
OSCCON_IRCF = 0b111                             -- 8 MHz
enable_digital_io()
const byte pin_dummy = 1                        -- virtual pin, reads 1
const byte led1[]= {0xFF,0x00,0x30,0x48,0x90,0x90,0x48,0x30} -- I♥
const byte led2[]= {0xFF,0x18,0x18,0xFF,0x00,0xF8,0xA8,0xA8} -- HE
alias spi_master_sw_sdi           is pin_dummy  -- dummy spi data input
-- alias spi_master_sw_sdi_direction is pin_sdi_direction
alias spi_master_sw_sdo           is pin_A0     -- spi data out
alias spi_master_sw_sdo_direction is pin_A0_direction
alias spi_master_sw_sck           is pin_A1     -- spi clock
alias spi_master_sw_sck_direction is pin_A1_direction
alias spi_master_sw_sel           is pin_A2     -- spi select
alias spi_master_sw_sel_direction is pin_A2_direction
-- spi_master_sw_sdi_direction = input          -- spi input
spi_master_sw_sdo_direction = output            -- spi output
spi_master_sw_sck_direction = output            -- spi clock
spi_master_sw_sel_direction = output            -- spi select
spi_master_sw_sel = high                        -- unselect
include spi_master_sw
spi_master_sw_init(SPI_MODE_00)                 -- init spi, choose mode
```

spi_max7219_led8x8.jal – 3/2.

```
procedure spi_write2(byte in addr, byte in data) is
  spi_master_sw_sel = low                -- CS aktiválás
  spi_master_sw = addr                   -- regisztercím kiküldése
  spi_master_sw = data                   -- adat kiküldése
  spi_master_sw_sel = high               -- CS deaktiválás
end procedure

procedure init_max7219() is
 spi_write2(0x09, 0x00)                  -- Dekódolást kikapcsoljuk
 spi_write2(0x0A, 0x0C)                  -- Fényerő beállítása
 spi_write2(0x0B, 0x07)                  -- Pásztázási korlát beállítása
 spi_write2(0x0C, 0x01)                  -- Megjelenítési mód beállítása
 spi_write2(0x0F, 0x0F)                  -- Display teszt engedélyezés
 delay_1ms(500)
 spi_write2(0x01, 0x00)                  -- 0. sor törlése
 spi_write2(0x02, 0x00)                  -- 1. sor törlése
 spi_write2(0x03, 0x00)                  -- 2. sor törlése
 spi_write2(0x04, 0x00)                  -- 3. sor törlése
 spi_write2(0x05, 0x00)                  -- 4. sor törlése
 spi_write2(0x06, 0x00)                  -- 5. sor törlése
 spi_write2(0x07, 0x00)                  -- 6. sor törlése
 spi_write2(0x08, 0x00)                  -- 7. sor törlése
 spi_write2(0x0F, 0x00)                  -- Display teszt tiltás
 delay_1ms(500)
end procedure
```

spi_max7219_led8x8.jal – 3/3.

```
var byte i = 0
init_max7219()                          -- A kijelző inicializálása

forever  loop
  -- Első bitkép kiírása (I♥)
  for 8 using i loop
    spi_write2(i+1,led1[i])
  end loop
  delay_1ms(1000)

  -- Második bitkép kiírása (HE)
  for 8 using i loop
    spi_write2(i+1,led2[i])
  end loop
  delay_1ms(1000)

end loop
```

```
Code area: 298 of 2048 used (words)
Data area: 14 of 128 used
Software stack available: 79 bytes
Hardware stack depth 3 of 8
0 errors, 0 warnings
```

SimulIDE szimuláció