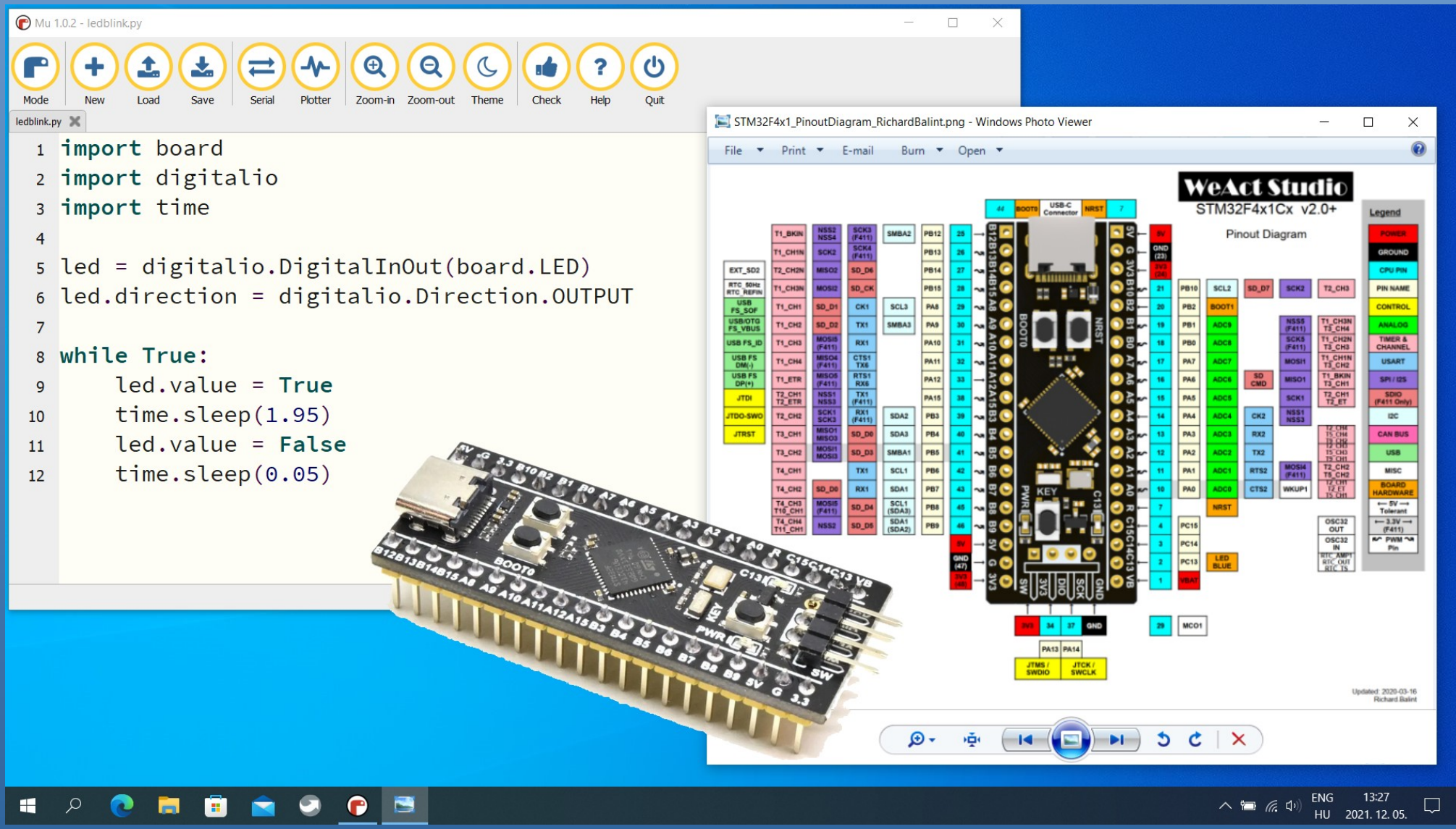
# CircuitPython tanfolyam




# 9. Az MCP23017 portbővítő használata - 2. rész

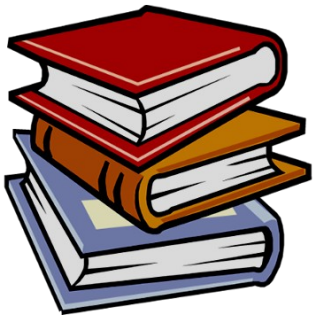
# Felhasznált és ajánlott irodalom

## Python:

- Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
- P. Wentworth et al. (ford. Biró Piroska, Szeghalmy Szilvia és Varga Imre): Hogyan gondolkozz úgy, mint egy informatikus: Tanulás Python 3 segítségével

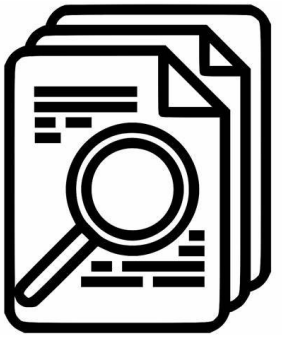
## CircuitPython:

- Adafruit: https://circuitpython.org/downloads
- Adafruit: Adafruit CircuitPython API Reference
- Learn Adafruit: CircuitPython Essentials
- Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle



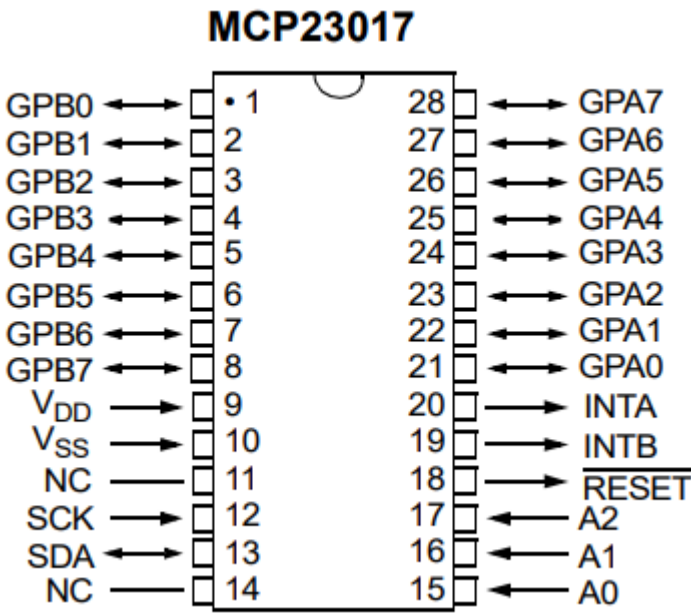
## Adatlapok és dokumentáció:

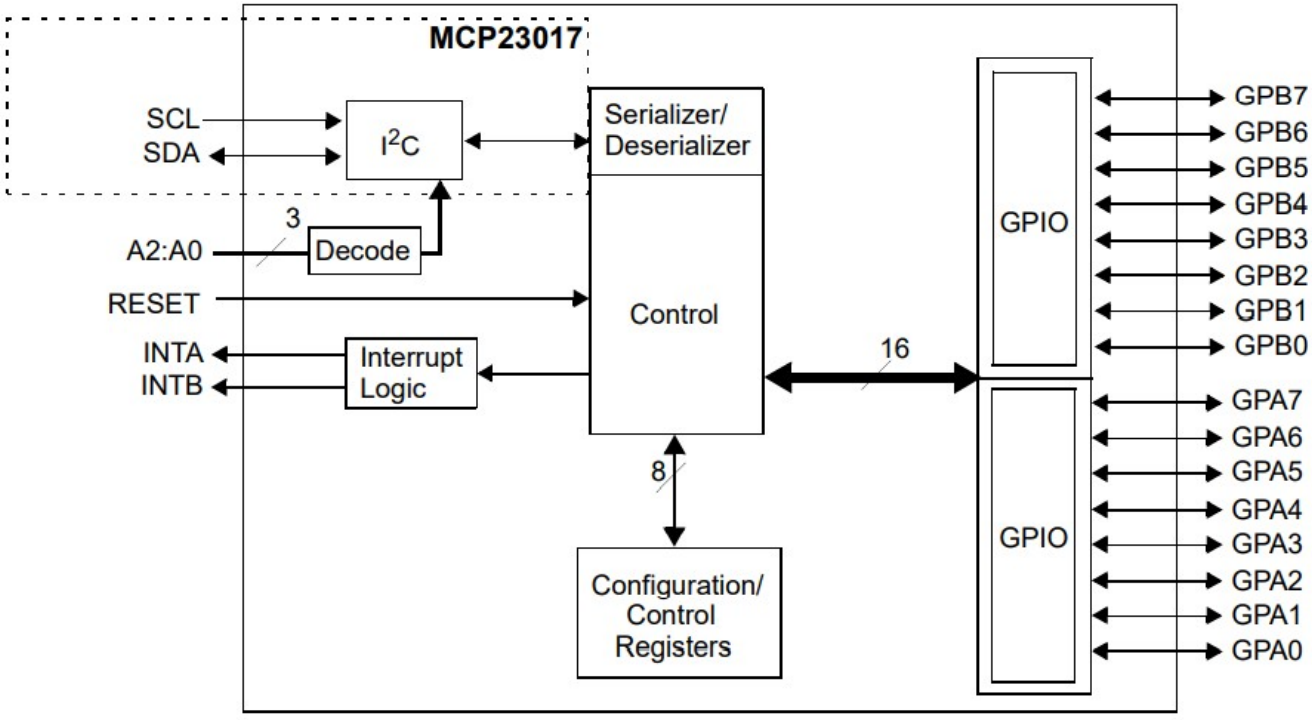
- **MCP23 017/MCP23S17:** adatlap és termékinfo
- **Microchip:** AN1081 Interfacing a 4x4 Matrix Keypad ...
- **STM32F411CE** adatlap és termékinfo
- **STM32F411xC/E** Family Reference Manual
- **WeAct Studio:** STM32F4x1 MiniF4



# Emlékeztető: Az MCP23017 periféria bővítő

- A Microchip gyártmányú **MC23017** IC egy 16 bites kétirányú portbővítő, ami az **I2C** buszra csatlakozik (100, 400 és 1700 kHz)
- Három címvonallal nyolcféle cím állítható be (0x20-0x27)
- A 2x8 bites portok megszakításkérő jelet is adnak
- Konfigurálható a megszakításkérő jel forrása, a 8, vagy 16 bites mód, a bemenő jel polaritása és a belső felhúzás
- Tápfeszültség: 1.8–5.5 V
- Hardver RESET



# Emlékeztető: Az MCP23017 periféria bővítő

- **Bájt módú** címzésnél nincs automatikus cím inkrementálás, `IOCON.BANK=0` esetén azonban 16 bites kiolvasás van: kiolvasáskor a regiszterek **A** és **B** fele váltakozik
- **Szekvenciális módnál** a cím minden kiolvasás után automatikusan inkrementálódik

**FIGURE 3-4: I²C CONTROL BYTE FORMAT**

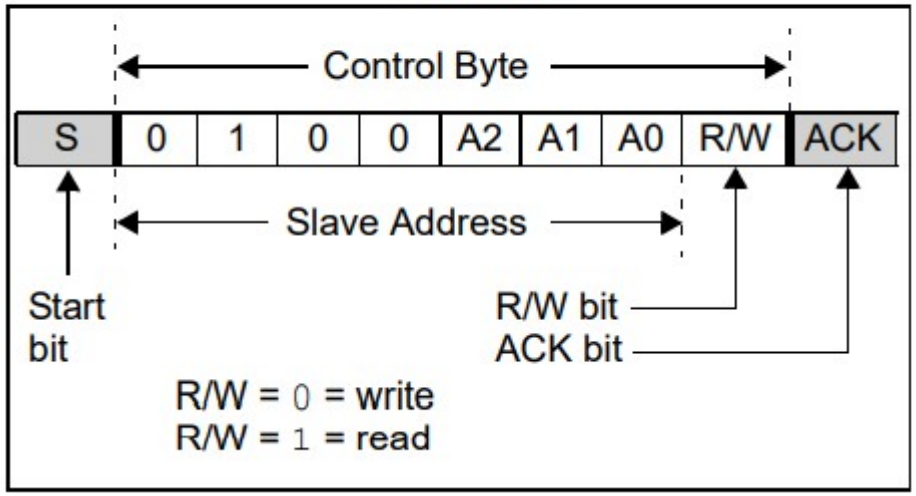



**TABLE 3-1: REGISTER ADDRESSES**

| Address IOCON.BANK = 1 | Address IOCON.BANK = 0 | Access to: |
|---|---|---|
| 00h | 00h | IODIRA |
| 10h | 01h | IODIRB |
| 01h | 02h | IPOLA |
| 11h | 03h | IPOLB |
| 02h | 04h | GPINTENA |
| 12h | 05h | GPINTENB |
| 03h | 06h | DEFVALA |
| 13h | 07h | DEFVALB |
| 04h | 08h | INTCONA |
| 14h | 09h | INTCONB |
| 05h | 0Ah | IOCON |
| 15h | 0Bh | IOCON |
| 06h | 0Ch | GPPUA |
| 16h | 0Dh | GPPUB |
| 07h | 0Eh | INTFA |
| 17h | 0Fh | INTFB |
| 08h | 10h | INTCAPA |
| 18h | 11h | INTCAPB |
| 09h | 12h | GPIOA |
| 19h | 13h | GPIOB |
| 0Ah | 14h | OLATA |
| 1Ah | 15h | OLATB |

# Emlékeztető: Az MCP23017 periféria bővítő

**A regiszterkészlet:**

- **IODIRA/B** – adatáramlási irány (0: kimenet, 1: bemenet)
- **IPOLA/B** – bemeneti polaritás (0: normál, 1: inverz)
- **GPINTENA/B** – interrupt-on-change engedélyezése
- **DEFVALA/B** – összehasonlító regiszter (default value)
- **INTCONA/B** – megszakítási mód (0: az előző értékhez hasonlít, 1: az összehasonlító regiszterhez (DEFVAL) hasonlít)
- **IOCON** – konfigurációs regiszter (`BANK,MIRROR,SEQOP,INTPOL...`)
- **GPPUA/B** – belső felhúzás engedélyezése
- **INTFA/B** – megszakításkérő jelzőbitek
- **INTCAPA/B** – megszakításkor rögzített állapot
- **GPIOA/B** – portkivezetések aktuális állapota
- **OLATA/B** – kimeneti adatregiszter

# Az Adafruit_MCP230xx könyvtár használata

- Az **MCP23 017** IC-t az **adafruit_mcp230xx** programkönyvtár segítségével kezelhetjük ( az adafruit-circuitpython-bundle-7.x csomagból is telepíthetjük)
- Az egyszerű I/O kezelést az alábbi példa mutatja be:

```
import board
import busio
import digitalio
from adafruit_mcp230xx.mcp23017 import MCP23017        ← Telepíteni kell!

i2c = busio.I2C(board.SCL, board.SDA)
mcp = MCP23017(i2c, address=0x21)       # MCP23017 w/ A0 set
# For the MCP23017 you specify a pin number from
# 0 to 15 for the GPIOA0...GPIOA7, GPIOB0...GPIOB7 pins.
pin0 = mcp.get_pin(8)    # B0
pin1 = mcp.get_pin(9)    # B1

pin0.switch_to_output(value=True)
pin1.direction = digitalio.Direction.INPUT
pin1.pull = digitalio.Pull.UP
while True:
    pin0.value = not pin1.value
```

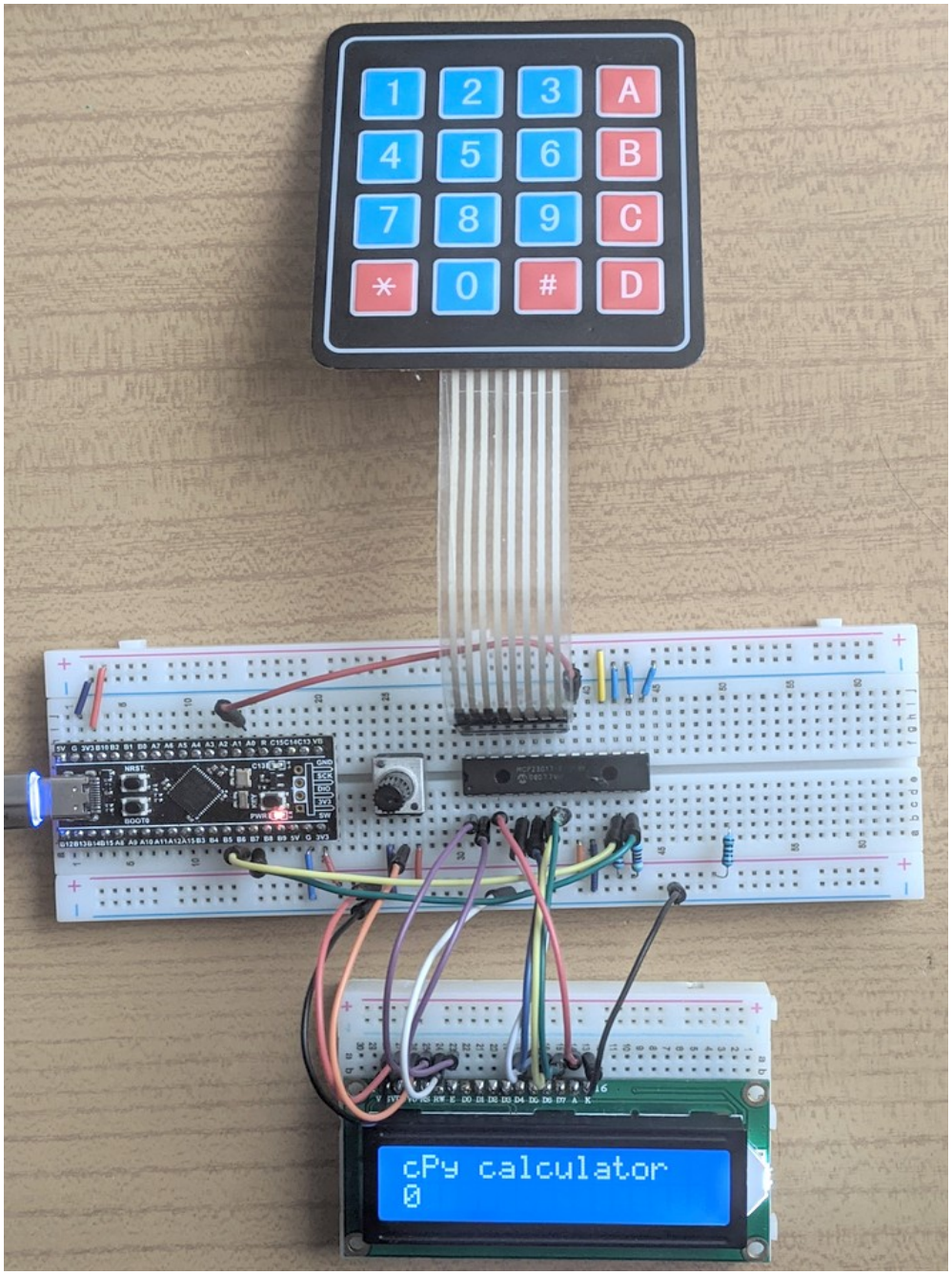
# CircuitPython Kalkulátort építünk

**A kalkulátort az alábbi lépésekben valósítjuk meg:**

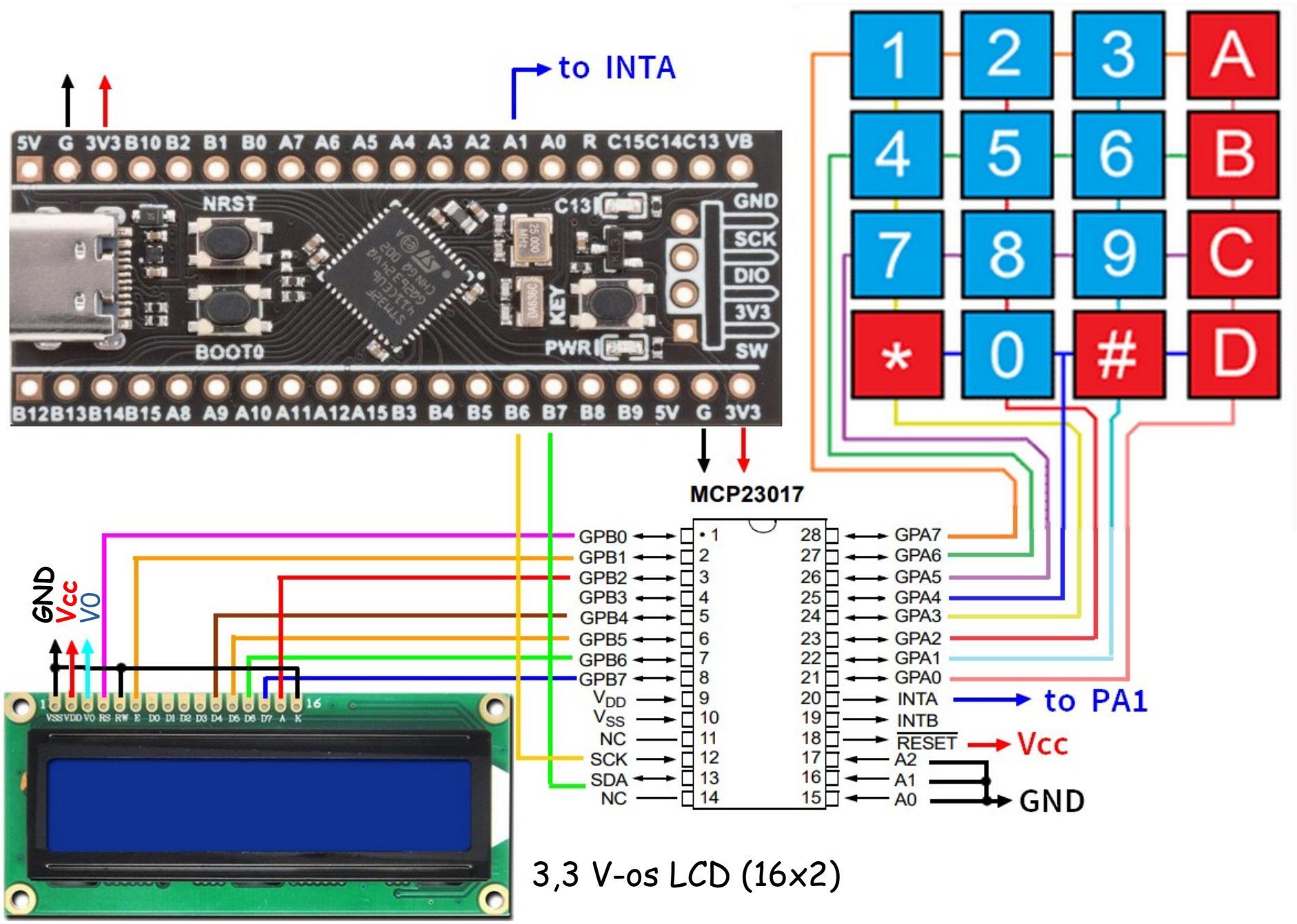
- A kapcsolás kialakítása
- A 4x4 billentyűzet kezelése
- Egyszerű kalkulátor programjának tanulmányozása
- Kalkulátor soros porti kiíratással
- LCD kezelés **MCP23 017** IC-vel
- Kalkulátor LCD kiíratással

**Telepítendő könyvtárak:**

- **adafruit_character_lcd**
- **adafruit_mcp230xx**



# Kapcsolási vázlat



# Példaprogramok

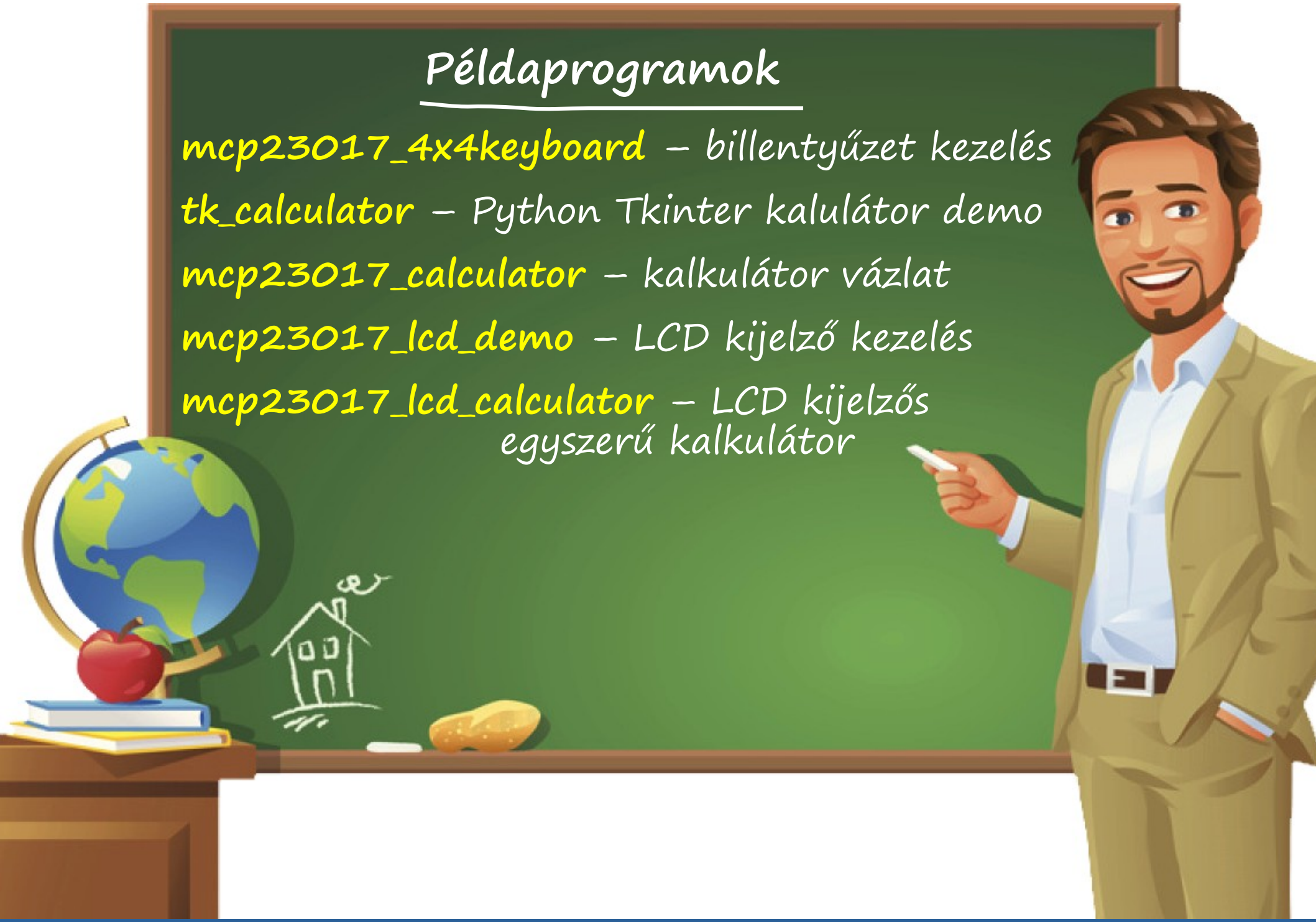
**mcp23017_4x4keyboard** – billentyűzet kezelés

**tk_calculator** – Python Tkinter kalulátor demo

**mcp23017_calculator** – kalkulátor vázlat

**mcp23017_lcd_demo** – LCD kijelző kezelés

**mcp23017_lcd_calculator** – LCD kijelzős egyszerű kalkulátor

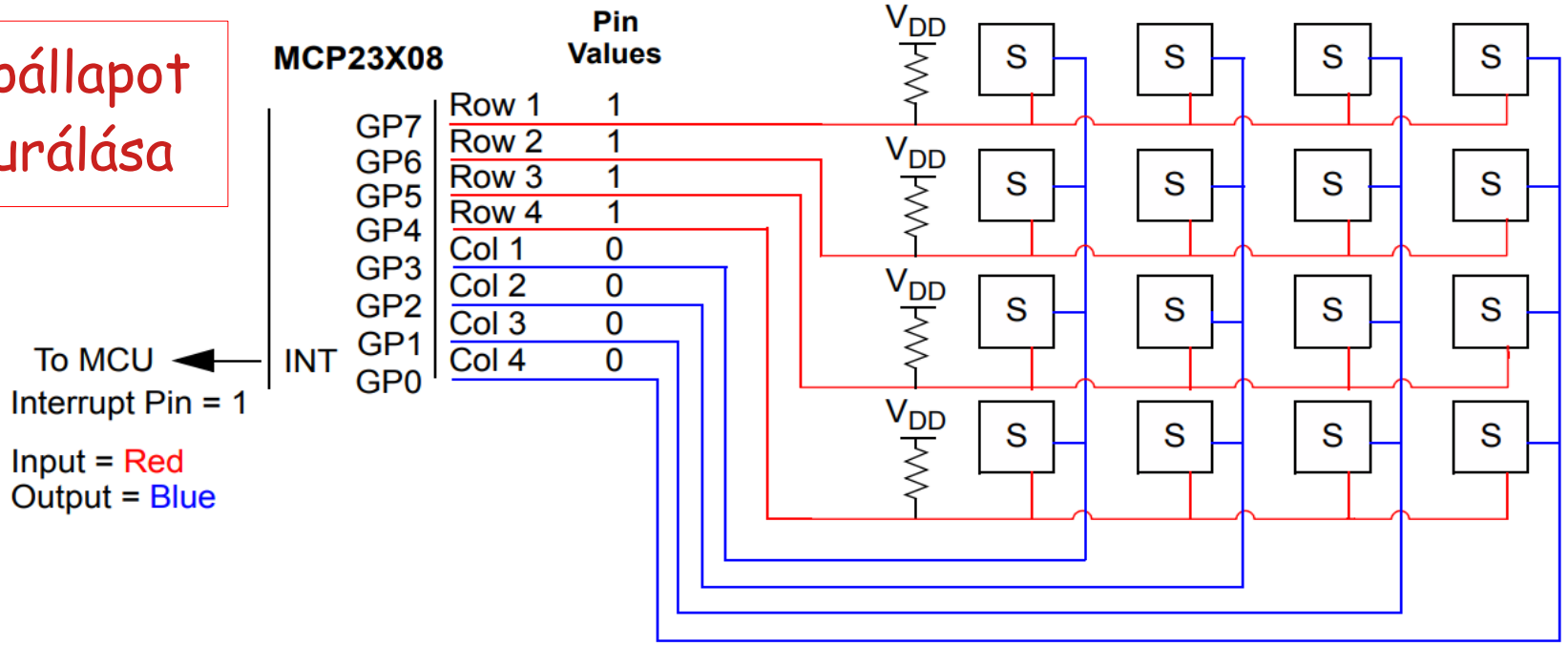


# 4x4 billentyűzet kezelése

- A Microchip **AN1081** alkalmazási mintapéldája egy **MCP23 008** IC felhasználásával oldja meg a 4x4-es billentyűzet illesztését, mi ugyanerre a feladatra az **MCP23 017 GPIOA** portját használjuk

| Register | Row 1 | Row 2 | Row 3 | Row 4 | Col 1 | Col 2 | Col 3 | Col 4 | |
|---|---|---|---|---|---|---|---|---|---|
| GPIO | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |
| INTCAP | — | — | — | — | — | — | — | — | — |
| IODIR | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |
| INTCON | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |
| DEFVAL | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |
| GPINTEN | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |
| GPPU | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xF0 |

Az alapállapot konfigurálása

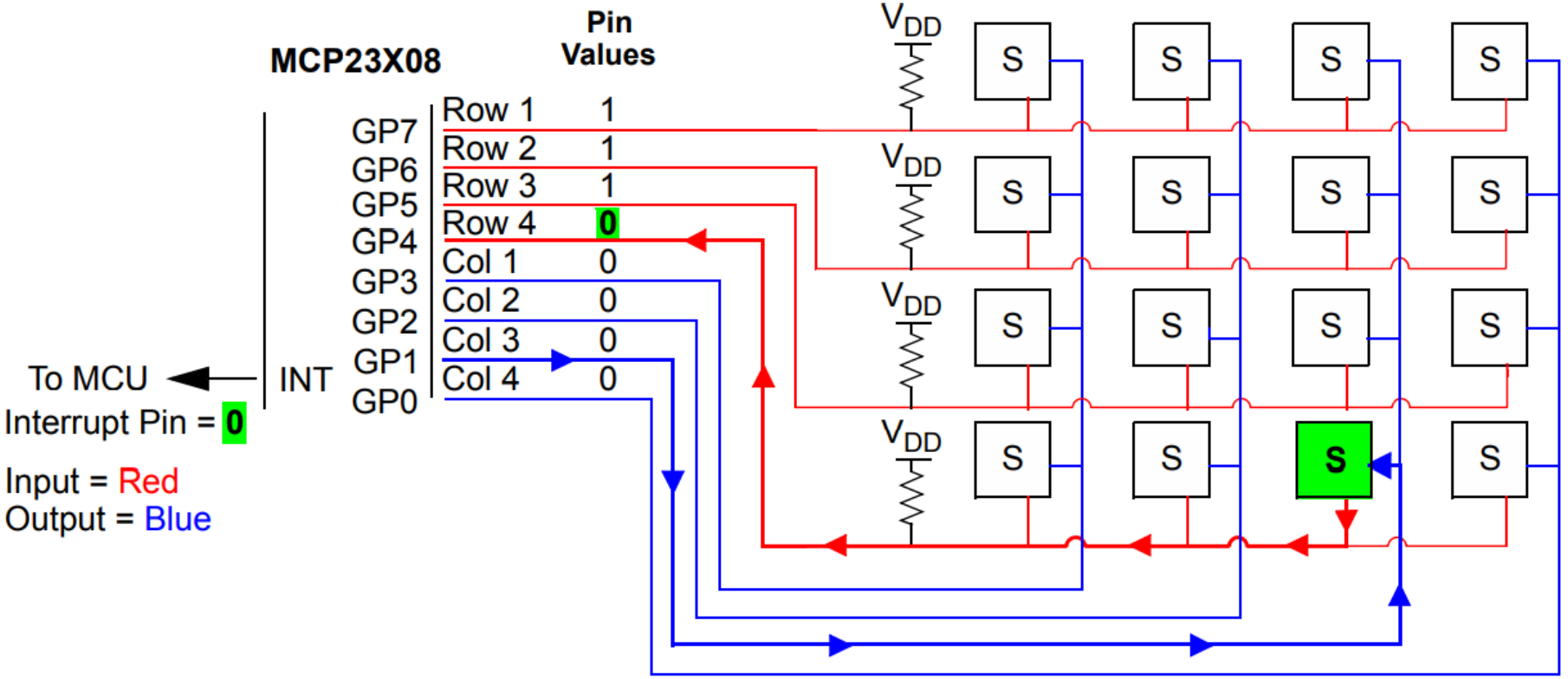


# Gomblenyomás – sor detektálása

- Gomblenyomáskor valamelyik sor (Row1..Row4) alacsony szintre vált, ennek pozíciója megadja a lenyomott gomb sorszámát
  A példánkban **0xE0** ⟶ **Row4**-et jelenti

| Pin Values | 1 | 1 | 1 | **0** | 0 | 0 | 0 | 0 | 0xE0 |
|---|---|---|---|---|---|---|---|---|---|

| Interrupt Pin Value | **0** |
|---|---|

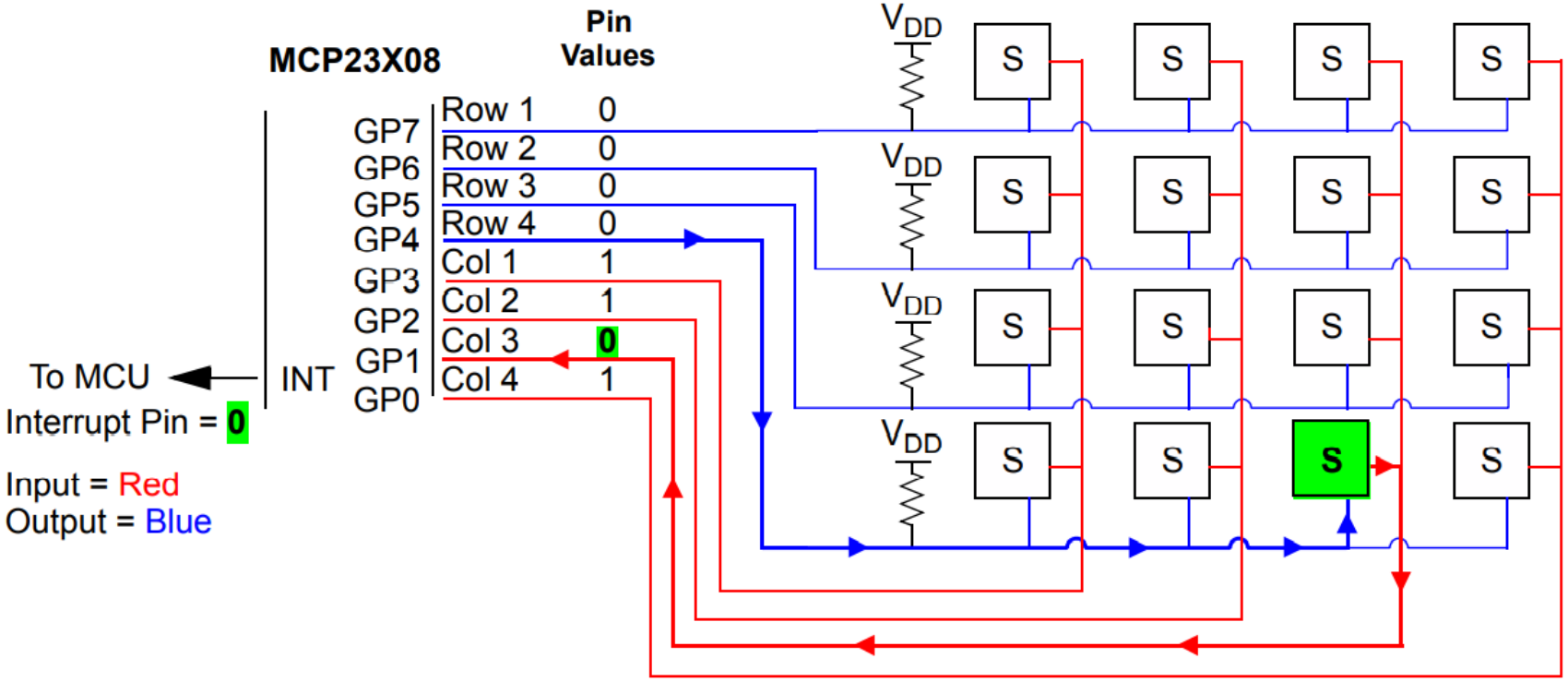


# Gomblenyomás – oszlop detektálása

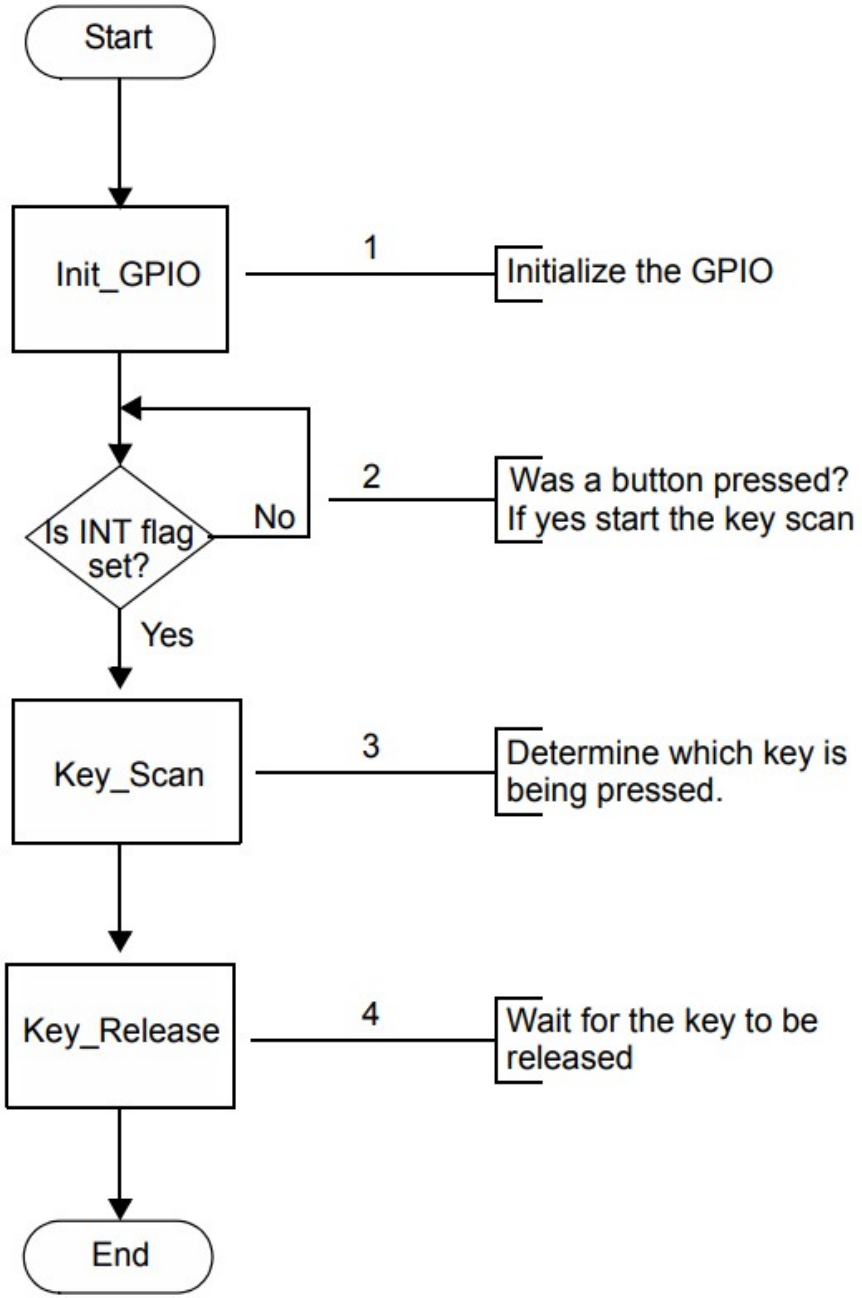
- Ha az I/O port két felét átváltjuk (a port felső fele lesz alacsony szintű kimenet, az alsó fele pedig belsőleg felhúzott bemenet), akkor meghatározhatjuk a lenyomott gomb oszlopát. Az alábbi példában beolvasott **0x0D** érték a harmadik oszlopot jelenti

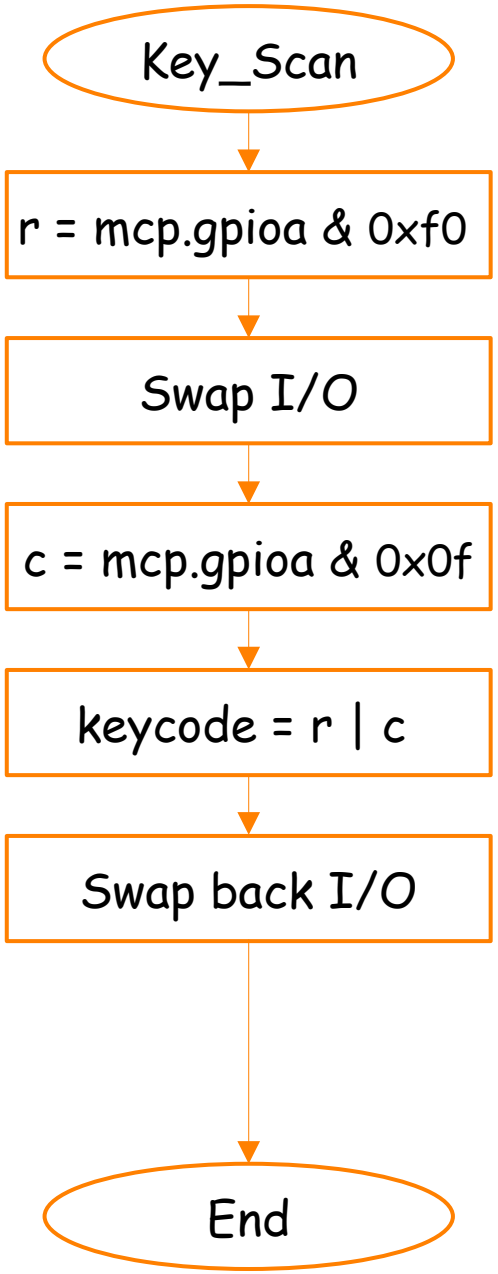
| Pin Values | 0 | 0 | 0 | 0 | 1 | 1 | **0** | 1 | 0x0D |
|---|---|---|---|---|---|---|---|---|---|

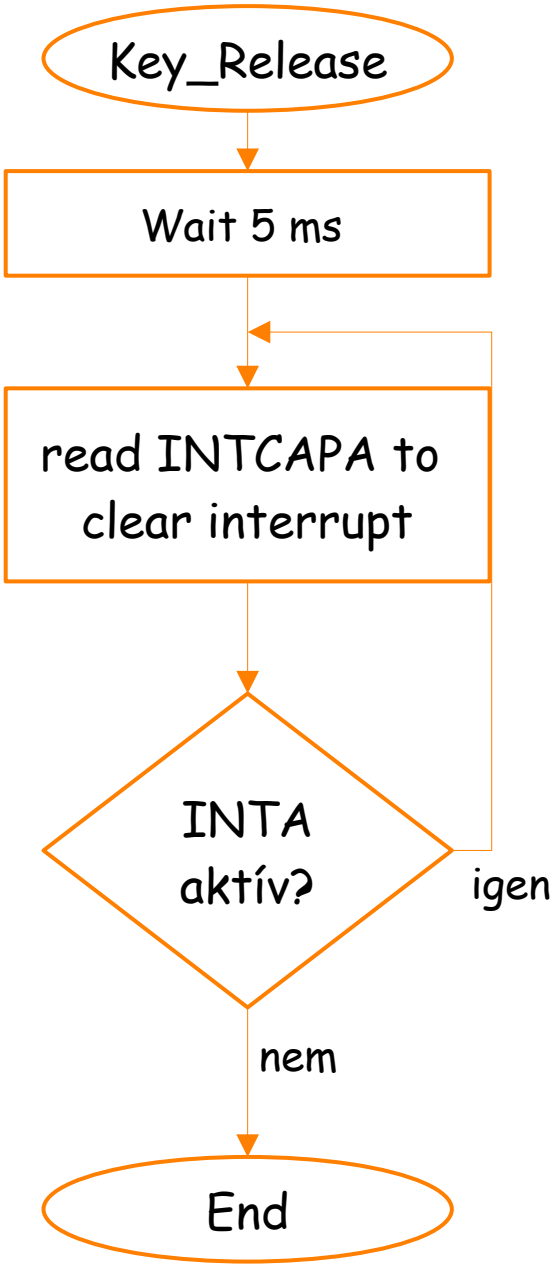
| Interrupt Pin Value | **0** |
|---|---|



# Folyamatábrák



# mcp23017_4x4keyboard.py – 3/1.

- A program érzékeli a gomblenyomásokat és a soros porton kiírja a lenyomott gombhoz rendelt értéket (0 - 0xF)

```
import time
import board
import busio
from digitalio import *
from adafruit_mcp230xx.mcp23017 import MCP23017

# Key codes and the corresponding data values
decoder = { 0x77 : 1,   0x7B : 2, 0x7D : 3,   0x7E : 0XA,
            0xB7 : 4,   0xBB : 5, 0xBD : 6,   0xBE : 0XB,
            0xD7 : 7,   0xDB : 8, 0xDD : 9,   0xDE : 0XC,
            0xE7 : 0XE, 0xEB : 0, 0xED : 0XF, 0xEE : 0XD
          }

# board.A1 accepts the interrupt signal of the MCP230xx
irq_a = DigitalInOut(board.A1)
irq_a.direction = Direction.INPUT
irq_a.pull = Pull.UP

# Initialize the I2C bus:
i2c = busio.I2C(board.SCL, board.SDA)
mcp = MCP23017(i2c, address=0x20, reset=True)    # MCP23017
```

# mcp23017_4x4keyboard.py – 3/2.

- A **key_scan** függvény az **AN1081**-ben leírt algoritmust valósítja meg a lenyomott gomb sor és oszlop koordinátáinak meghatározására

```
def key_scan()-> int:
    r = (mcp.gpioa) & 0xF0 # Read row
    #--- Flip I/O --------------------
    mcp.iodira = 0x0F
    mcp.gppua = 0x0F
    mcp.interrupt_configuration = 0x0F
    mcp.default_value = 0x0F
    mcp.interrupt_enable = 0x0F
    #--- Determine and print key coordinates (r, c):
    c = (mcp.gpioa) & 0x0F # Read column
    print("Row: {0}, Column: {1}".format(hex(r>>4),hex(c)))
    #--- Re-initialize I/O ------------
    mcp.iodira = 0xF0
    mcp.gppua = 0xF0
    mcp.interrupt_configuration = 0xF0
    mcp.default_value = 0xF0
    mcp.interrupt_enable = 0xF0
    return (r | c)
```

# mcp23017_4x4keyboard.py – 3/3.

- A főprogramban csak az egyedi gomblenyomásokat fogadjuk el

```
def gpio_init():                              # MCP23017 kezdeti konfigurálás
    mcp.iocon = 0
    mcp.iodira = 0xF0                         # Felső félbájt bemenet, alsó kimenet
    mcp.gppua = 0xF0                          # Belső felhúzás bekapcsolása
    mcp.interrupt_configuration=0xF0 # Megszakítás, ha a bemenet eltér
    mcp.default_value = 0xF0                  # Ehhez az értékhez hasonlítunk
    mcp.interrupt_enable = 0xF0               # Megszakítás engedélyezése

def wait_for_release():                       # Várakozás a nyomógomb elengedésére
    while not irq_a.value:
        time.sleep(0.005)
        dummy = mcp.int_capa                  # A kiolvasása törli a megszakítást

gpio_init()

while True:
    if not irq_a.value:                       # Megszakításkor INTA alacsonyra vált
        keycode = key_scan()
        if keycode in decoder:                # A többszörös gomblenyomást eldobjuk
            ch = decoder[keycode]
            print("code = {0}".format(hex(ch)))
        wait_for_release()                    # Várakozás a nyomógomb elengedésére
```

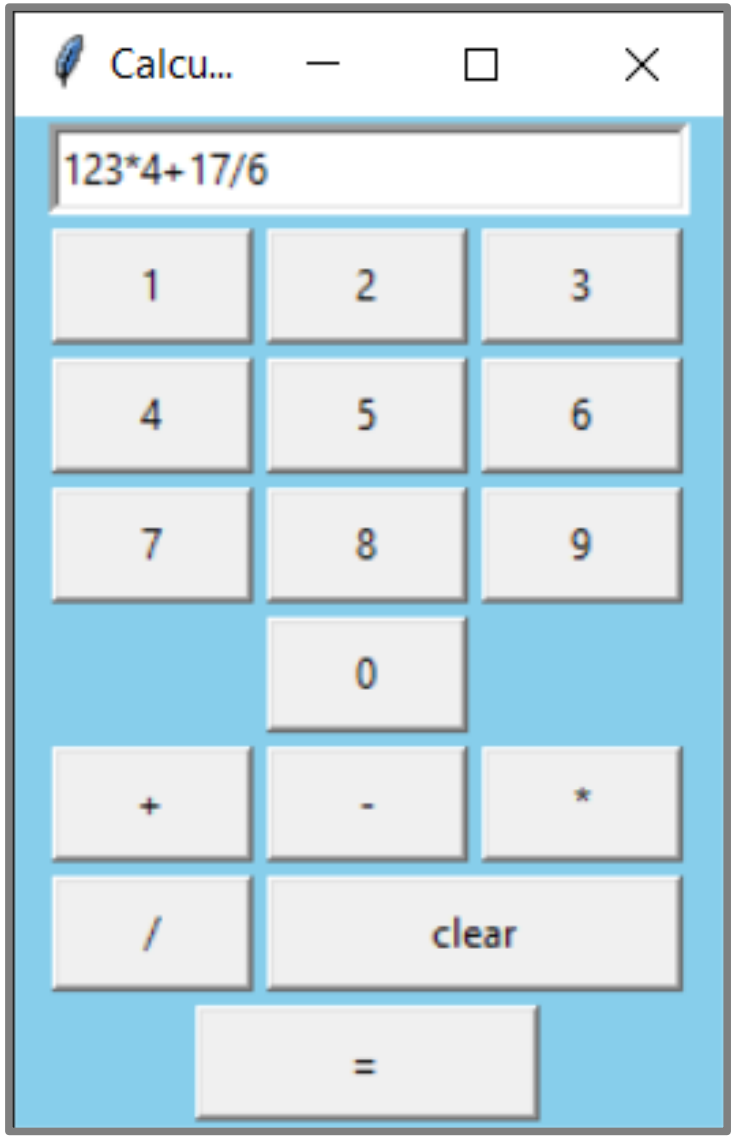
# mcp23017_4x4keyboard.py futási eredménye

- Az 1,2,3,A,5,9,E és D gombokat lenyomva a terminálon ezt látjuk:

```
Auto-reload is on. Simply save files over USB to run them or enter
REPL to disable.
code.py output:
Row: 0x7, Column: 0x7
code = 0x1
Row: 0x7, Column: 0xb
code = 0x2
Row: 0x7, Column: 0xd
code = 0x3
Row: 0x7, Column: 0xe
code = 0xa
Row: 0xb, Column: 0xb
code = 0x5
Row: 0xd, Column: 0xd
code = 0x9
Row: 0xe, Column: 0x7
code = 0xe
Row: 0xe, Column: 0xe
code = 0xd
```

# Basic calculator program using Python

- A **Geeks for geeks oldalon közzétett** Python programból kileshetjük, hogy hogyan lehet egy egyszerű kalkulátort megvalósítani Pythonban
- A kalkulátor a 0 – 9 számjegyeken kívül csak a négy alapművelet gombját, valamint törlő gombot és az eredményt produkáló egyenlőségjelet használ, így jól illeszkedik a 4x4 gombos billentyű-zetünkhöz
- Tizedespont gomb nincs, így némileg korlátozott a használat (körülményesebb egy kicsit a törtek beírása)
- A program a **Tkinter** grafikus bővítményt használja



# tk_calculator.py – 2/1.

```
import tkinter as tk
import tkinter.messagebox
from tkinter.constants import SUNKEN

window = tk.Tk()
window.title('Calculator-GeeksForGeeks')
frame = tk.Frame(master=window, bg="skyblue", padx=10)
frame.pack()
entry = tk.Entry(master=frame, relief=SUNKEN, borderwidth=3, width=30)
entry.grid(row=0, column=0, columnspan=3, ipady=2, pady=2)

def myclick(number):
    entry.insert(tk.END, number)          # Adatbevitel

def equal():
    try:
        y = str(eval(entry.get()))        # A beírt kifejezés kiszámítása
        entry.delete(0, tk.END)
        entry.insert(0, y)                # Az eredmény kijelzése
    except:
        tkinter.messagebox.showinfo("Error", "Syntax Error")

def clear():
    entry.delete(0, tk.END                # Törli az adatbeviteli mezőt
```

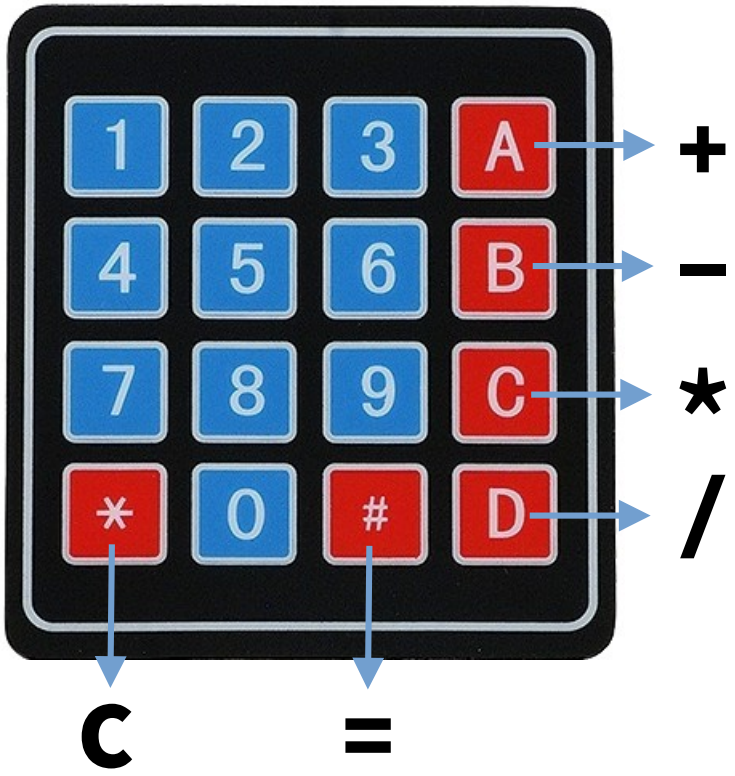
# tk_calculator.py – 2/2. (részlet)

```
button_1 = tk.Button(master=frame, text='1', padx=15, pady=5, width=3,
command=lambda: myclick(1))
button_1.grid(row=1, column=0, pady=2)
  . . .
button_add = tk.Button(master=frame, text="+", padx=15, pady=5, width=3,
command=lambda: myclick('+'))
button_add.grid(row=5, column=0, pady=2)
button_subtract = tk.Button(master=frame, text="-", padx=15, pady=5, width=3,
command=lambda: myclick('-'))
button_subtract.grid(row=5, column=1, pady=2)
button_multiply = tk.Button(master=frame, text="*", padx=15, pady=5, width=3,
command=lambda: myclick('*'))
button_multiply.grid(row=5, column=2, pady=2)
button_div = tk.Button(master=frame, text="/", padx=15, pady=5, width=3,
command=lambda: myclick('/'))
button_div.grid(row=6, column=0, pady=2)
button_clear = tk.Button(master=frame, text="clear", padx=15, pady=5,
width=12, command=clear)
button_clear.grid(row=6, column=1, columnspan=2, pady=2)
button_equal = tk.Button(master=frame, text="=", padx=15, pady=5, width=9,
command=equal)
button_equal.grid(row=7, column=0, columnspan=3, pady=2)

window.mainloop()
```

# mcp23017_calculator.py

- Az előző példából kiindulva készítsünk egy **CircuitPython** kalkulátort, amihez használjuk fel a 4x4 billentyűzet kezelést
- A kiíratás – az egyszerűség kedvéért – a **print** paranccsal történjen, az eredményt a terminál ablakban figyelhetjük meg
- A billentyűzet gombjait egy kicsit „át kell értelmezni”, az alábbi ábra szerint:




- A billentyűzet kódtáblája így alakul:

```
decoder = {}
decoder[0x77] = '1'
decoder[0x7B] = '2'
decoder[0x7D] = '3'
decoder[0x7E] = '+'
decoder[0xB7] = '4'
decoder[0xBB] = '5'
decoder[0xBD] = '6'
decoder[0xBE] = '-'
decoder[0xD7] = '7'
decoder[0xDB] = '8'
decoder[0xDD] = '9'
decoder[0xDE] = '*'
decoder[0xE7] = 'C'
decoder[0xEB] = '0'
decoder[0xED] = '='
decoder[0xEE] = '/'
```

# mcp23017_calculator.py – 2/1.

```
import time
import board
import busio
from digitalio import *
from adafruit_mcp230xx.mcp23017 import MCP23017

# Key codes and the corresponding data values
decoder = { 0x77 : '1', 0x7B : '2', 0x7D : '3', 0x7E : '+',
            0xB7 : '4', 0xBB : '5', 0xBD : '6', 0xBE : '-',
            0xD7 : '7', 0xDB : '8', 0xDD : '9', 0xDE : '*',
            0xE7 : 'C', 0xEB : '0', 0xED : '=', 0xEE : '/'
          }

entry = "0"


# board.A1 accepts the interrupt signal of the MCP230xx
irq_a = DigitalInOut(board.A1)
irq_a.direction = Direction.INPUT
irq_a.pull = Pull.UP

# Initialize the I2C bus:
i2c = busio.I2C(board.SCL, board.SDA)
mcp = MCP23017(i2c, address=0x20, reset=True)     # MCP23017
```

Csak itt van változás a korábban bemutatott **mcp23017_4x4 keyboard.py** programhoz képest

# mcp23017_calculator.py – 2/2.

```
gpio_init()
print(entry)
while True:
    if not irq_a.value:
        keycode = key_scan()
        if keycode in decoder:
            ch = decoder[keycode]
            if ch == 'C':
                entry = '0'
            elif ch == '=':
                try:
                    y = str(eval(entry))
                    entry = y
                except SyntaxError:
                    print('Syntax error!')
                    entry ='0'
                except ZeroDivisionError:
                    print('division by zero')
                    entry = '0'
            else:
                if entry == '0' and ch in '0123456789':
                    entry = ''
                entry = entry + ch
        wait_for_release()
        print(entry)
```

A **gpio_init()**, **key_scan()** és a **wait_for_release()** függvények Ugyanazok, mint a korábban bemutatott mcp23017_4x4keyboard.py programban

# mcp23017_calculator.py futási eredménye

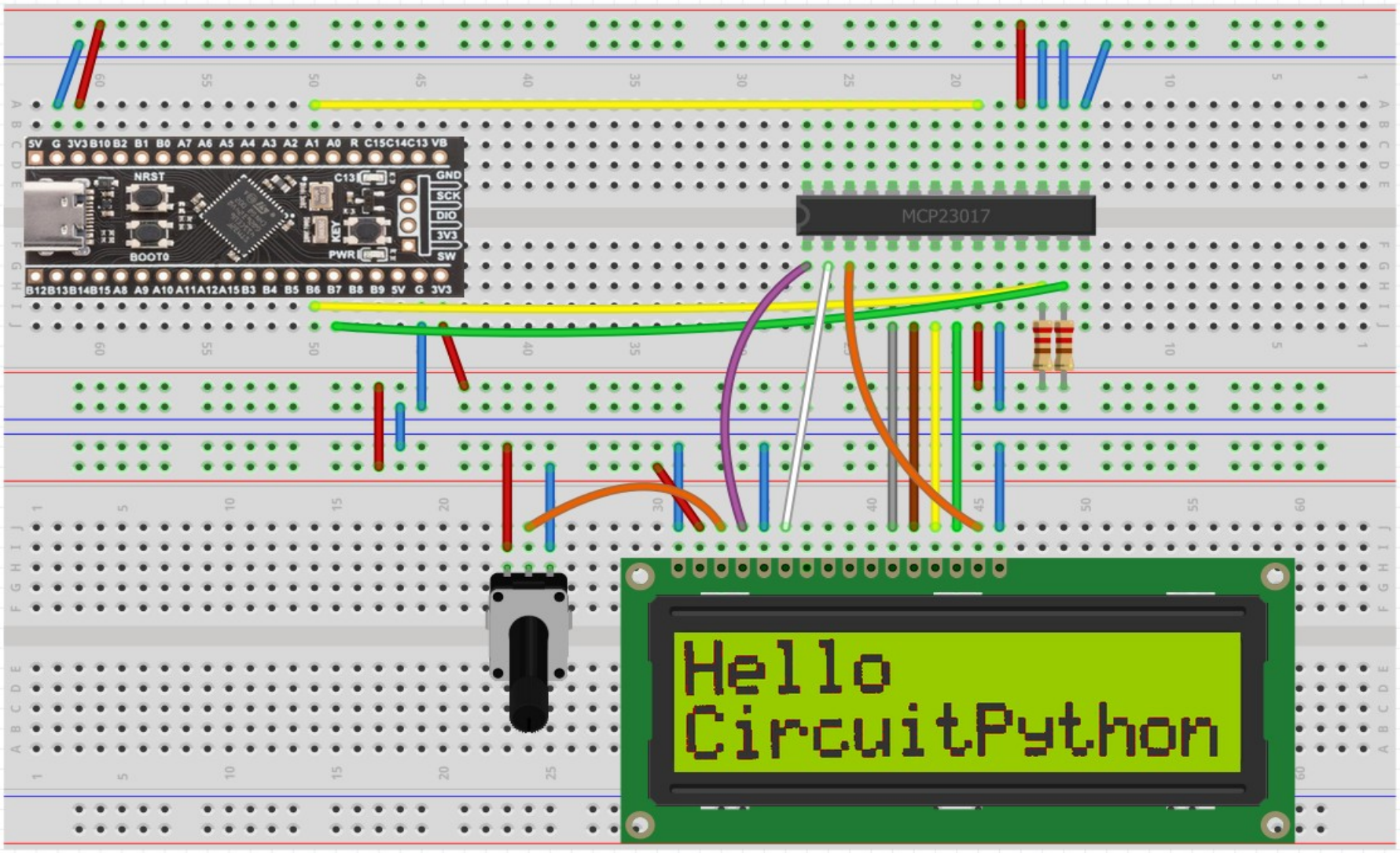
- Amint látható, az ’=’ gomb lenyomása előtt összetett kifejezéseket is beírhatunk
- Bónusz: dupla szorzásjel hatványozást jelent, dupla ’/’ jel pedig egész osztást végez (például: 2**3 = 8,  37//8 = 4)

```
Code stopped by auto-reload. Reloading soon.
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
0
1
12
12*
12*3
12*3+
12*3+4
12*3+45
12*3+45/
12*3+45/5
45.0
```

# LCD kijelző kezelése

- Lady Ada: Character LCDs leírását annyival bővítjük, hogy az LCD kijelzőt az **MCP23 017** I/O bővítőn keresztül vezéreljük. A hardver absztrakció miatt a kijelző teljesen transzparensen kezelhető



# mcp23017_lcd_demo.py – 2/1.

A program forrása: lcd_mono_simpletest.py
az Adafruit_CircuitPython_CharLCD könyvtár mintapéldája

```
import time
import board
import busio
from digitalio import *
import adafruit_character_lcd.character_lcd as characterlcd
from adafruit_mcp230xx.mcp23017 import MCP23017

i2c = busio.I2C(board.SCL, board.SDA)                # Initialize the I2C bus:
mcp = MCP23017(i2c, address=0x20, reset=True)        # MCP23017

# LCD is connected to the GPIOB port of the MCP23017 IC:
lcd_rs = mcp.get_pin(8)     # B0
lcd_en = mcp.get_pin(9)     # B1
lcd_d7 = mcp.get_pin(15)    # B7
lcd_d6 = mcp.get_pin(14)    # B6
lcd_d5 = mcp.get_pin(13)    # B5
lcd_d4 = mcp.get_pin(12)    # B4
lcd_backlight = mcp.get_pin(10) # B2

lcd_columns = 16
lcd_rows = 2
lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_backlight
)
```

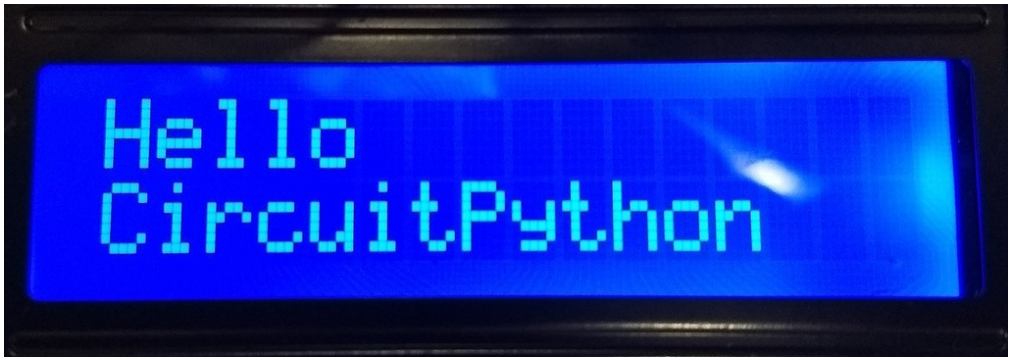
Itt már nem látszik, hogy közvetlenül, vagy portbővítőn keresztül kezeljük az LCD-t

# mcp23017_lcd_demo.py – 2/2.

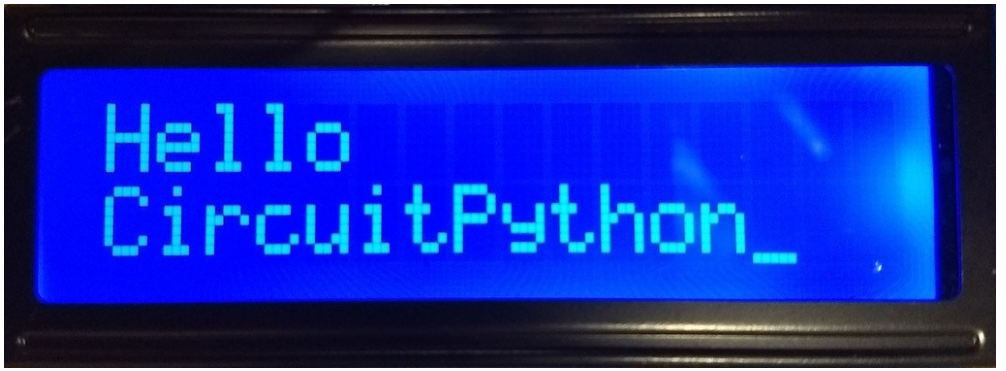
```
lcd.backlight = True                    # Turn backlight on
lcd.message = "Hello\nCircuitPython"    # Print a two line message
time.sleep(5)                           # Wait 5s
lcd.clear()
lcd.text_direction=lcd.RIGHT_TO_LEFT    # Print message right to left
lcd.message = "Hello\nCircuitPython"
time.sleep(5)                           # Wait 5s
lcd.text_direction=lcd.LEFT_TO_RIGHT    # Text direction back to left to right
lcd.clear()
lcd.cursor = True                       # Display cursor
lcd.message = "Cursor! "
time.sleep(5)                           # Wait 5s
lcd.clear()
lcd.blink = True                        # Display blinking cursor
lcd.message = "Blinky Cursor!"
time.sleep(5)                           # Wait 5s
lcd.blink = False; lcd.clear()
scroll_msg = "<-- Scroll"               # Create message to scroll
lcd.message = scroll_msg
for i in range(len(scroll_msg)):        # Scroll message to the left
    time.sleep(0.5); lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
time.sleep(3)                           # Wait 3s
lcd.backlight = False                   # Turn backlight off
```

# mcp23017_lcd_demo.py futási eredmény

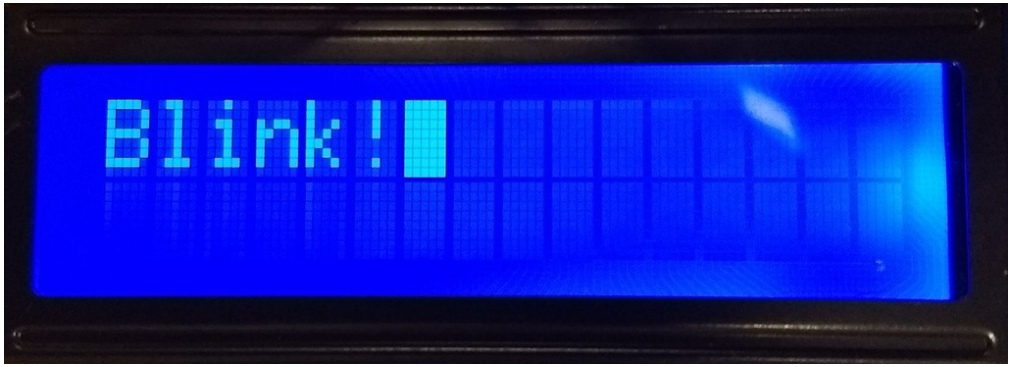
- Sikeres programfuttatásnál a kezdőképernyő így néz ki:




- Bekapcsolt kurzorral:




- Villogó kurzorral:



# Állítsuk össze a kalkulátort!

- Gyakorlatilag minden készen áll az LCD kijelzős kalkulátor elkészítéséhez
- A programot összeollózhatjuk a korábbi programokból:
  - mcp23017_calculator.py
  - mcp23017_lcd_demo.py
- A kijelzést annyival megbonyolítjuk, hogy a kétsoros kijelzőnél az alsó sorban jelenítjük meg az eredményt, a felső sorban pedig azt a kifejezést, aminek az eredményét az alsó sor mutatja
- Számjegy és műveleti jel beírásakor „terjeszkedik a kiírás”, de a többi esetben kiírás előtt törölnünk kell a képernyőt (hogy ne maradjon szemét)
  - Törlés (**C** gomb)
  - Hibajelzések (= gomb)
  - Új eredmény megjelenítése (= gomb)

# mcp23017_lcd_calculator.py – 3/1.

A program eleje a megjelölt sor kivételével megegyezik az mcp23017_calculator.py programmal

```
import time
import board
import busio
from digitalio import *
from adafruit_mcp230xx.mcp23017 import MCP23017
import adafruit_character_lcd.character_lcd as characterlcd

# Key codes and the corresponding data values
decoder = { 0x77 : '1', 0x7B : '2', 0x7D : '3', 0x7E : '+',
            0xB7 : '4', 0xBB : '5', 0xBD : '6', 0xBE : '-',
            0xD7 : '7', 0xDB : '8', 0xDD : '9', 0xDE : '*',
            0xE7 : 'C', 0xEB : '0', 0xED : '=', 0xEE : '/'
          }

entry = "0"

# board.A1 accepts the interrupt signal of the MCP230xx
irq_a = DigitalInOut(board.A1)
irq_a.direction = Direction.INPUT
irq_a.pull = Pull.UP

# Initialize the I2C bus:
i2c = busio.I2C(board.SCL, board.SDA)
mcp = MCP23017(i2c, address=0x20, reset=True)     # MCP23017
```

# mcp23017_lcd_calculator.py – 3/2.

```
# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

lcd_rs = mcp.get_pin(8)     # B0
lcd_en = mcp.get_pin(9)     # B1
lcd_backlight = mcp.get_pin(10)    # B2
lcd_d4 = mcp.get_pin(12)    # B4
lcd_d5 = mcp.get_pin(13)    # B5
lcd_d6 = mcp.get_pin(14)    # B6
lcd_d7 = mcp.get_pin(15)    # B7

# Initialise the LCD class
lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows,
lcd_backlight
)

def gpio_init():
def key_scan()-> int:
def wait_for_release():

lcd.backlight = True                # Turn backlight on
lcd.clear()                         # Clear screen
lcd.message = "cPy calculator\n0"
gpio_init()                         # Initialize keyboard
```

A program ezen része megegyezik az mcp23017_lcd_demo.py programmal

Megegyeznek a korábbiakkal (lásd: mcp23017_4x4keyboard.py, illetve mcp23017_calculator.py)

# mcp23017_lcd_calculator.py – 3/3.

```
while True:
    if not irq_a.value:
        keycode = key_scan()
        if keycode in decoder:
            ch = decoder[keycode]
            if ch == 'C':
                Entry = '0'; lcd,clear()
                lcd.message = "cPy calculator\n0"
            elif ch == '=':
                lcd.clear()
                try:
                    y = str(eval(entry))
                    lcd.message = entry + '\n' + y
                    entry = y
                except SyntaxError:
                    lcd.message = entry + '\nSynatax error!; entry ='0'
                except ZeroDivisionError:
                    lcd.message = entry + '\nDiv by zero!'; entry = '0'
            else:
                if entry == '0' and ch in '0123456789':
                    entry = ''
                    lcd.message = "cPy calculator\n"
                entry = entry + ch
                lcd.message = lcd.message + ch
        wait_for_release()
```

A kijelzés formátuma:

cPy calculator
0

12*3+4*5
=56

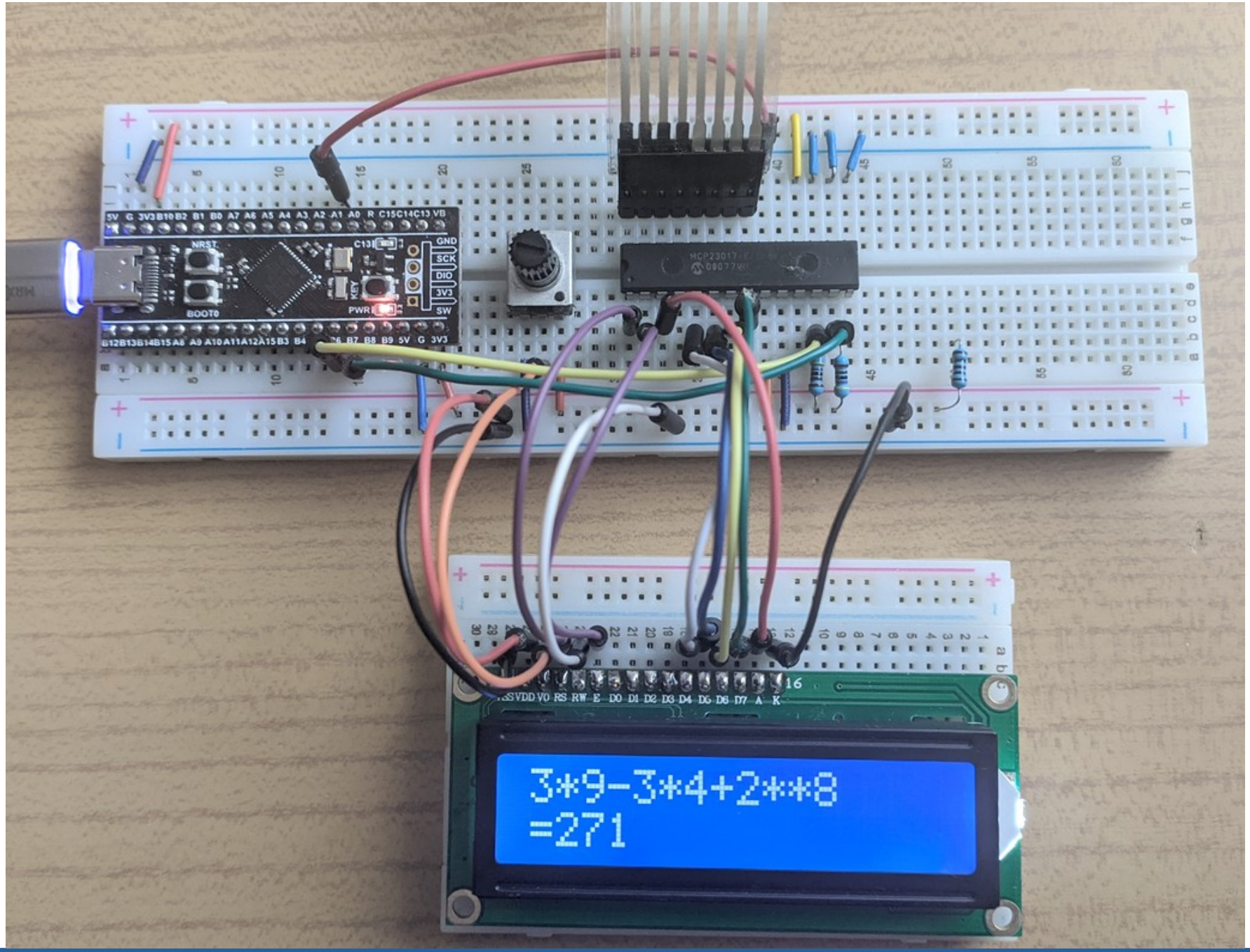
12***3+4*5
Syntax error!

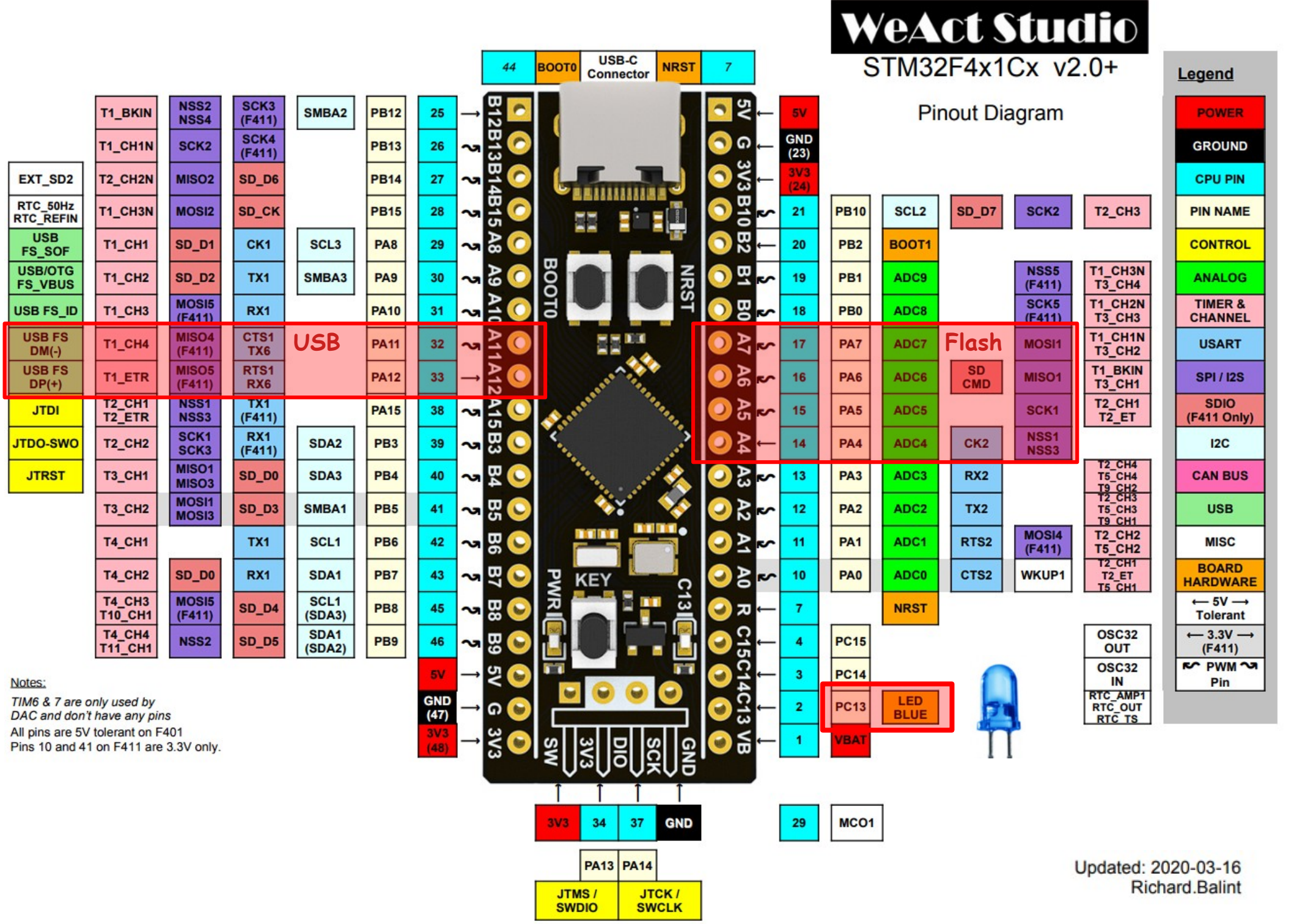
# Egy bosszantó apróság

- Az LCD kezelés felülírja **OLATA** értékét, ezért a **key_scan()** függvényben újra kell írni az értékét!

```
def key_scan()-> int:
    mcp.gpioa = 0           # Csak az LCD kezelés mellékhatása miatt kell
    r = (mcp.gpioa) & 0xF0 # Read row
    #--- Flip I/O ---------------------
    mcp.iodira = 0x0F
    mcp.gppua = 0x0F
    mcp.interrupt_configuration = 0x0F
    mcp.default_value = 0x0F
    mcp.interrupt_enable = 0x0F
    #--- Determine key code
    c = (mcp.gpioa) & 0x0F # Read column
    # print("Row: {0}, Column: {1}".format(hex(r>>4),hex(c)))
    #--- Re-initialize I/O ------------
    mcp.iodira = 0xF0
    mcp.gppua = 0xF0
    mcp.interrupt_configuration = 0xF0
    mcp.default_value = 0xF0
    mcp.interrupt_enable = 0xF0
    return (r | c)
```

# mcp23017_lcd_calculator.py futási eredmény




WeAct Studio
STM32F4x1Cx v2.0+
Pinout Diagram
Legend
POWER
GROUND
CPU PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI / I2S
SDIO (F411 Only)
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V Tolerant
3.3V (F411)
PWM Pin
USB
Flash
BOOT0
NRST
KEY
PWR
C13
LED BLUE
USB-C Connector
JTMS / SWDIO
JTCK / SWCLK
Notes:
TIM6 & 7 are only used by DAC and don't have any pins
All pins are 5V tolerant on F401
Pins 10 and 41 on F411 are 3.3V only.
Updated: 2020-03-16
Richard.Balint