Informativna, dodatna vsebina

ARM

Praktični projekt za STM32F4 vgrajen sistem (informativno, dodatno gradivo)

Programiranje v Micro Pythonu

LED, dvojna zanka, PWM, pospeškomer

VIN Projekt – Osnovna platforma

STM Discovery F4 (Cortex M4)

- STM32F407VGT6 microcontroller featuring 32-bit Arm® Cortex®-M4 with FPU core, 1-Mbyte Flash memory and 192-Kbyte RAM in an LQFP100 package
- USB OTG FS
- ST MEMS 3-axis accelerometer
- ST-MEMS audio sensor omni-directional digital microphone
- Audio DAC with integrated class D speaker driver
- User and reset push-buttons
- Eight LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 3.3 V power on
  - Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)
- Board connectors:
  - USB with Micro-AB
  - Stereo headphone output jack
  - 2.54 mm pitch extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing
- External application power supply: 3 V and 5 V

STM32F407 ST Discovery

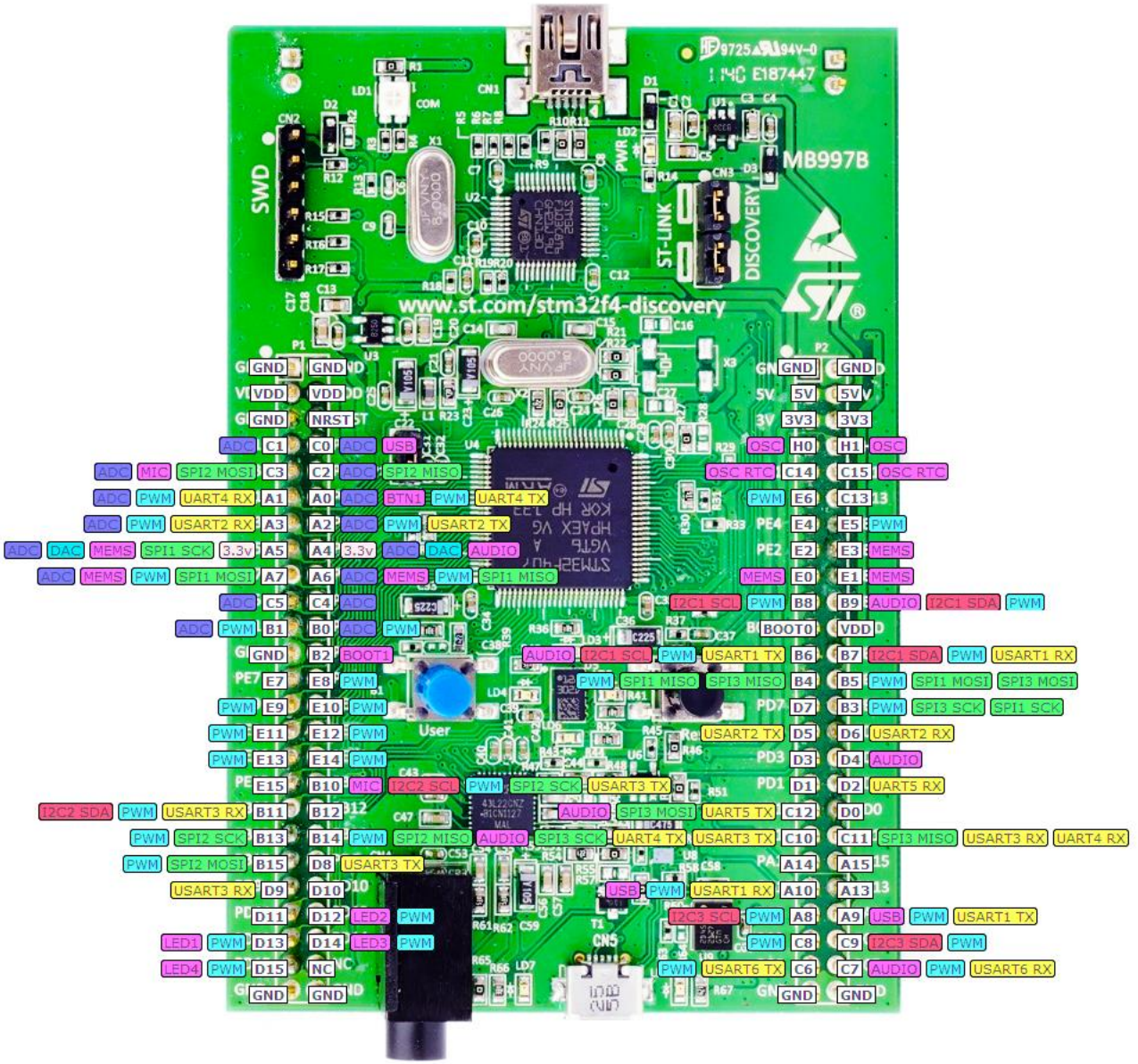
STM32

Informativna, dodatna vsebina

Razpored in funkcije priključkov („pinov“) :

STM32F4-DISCOVERY

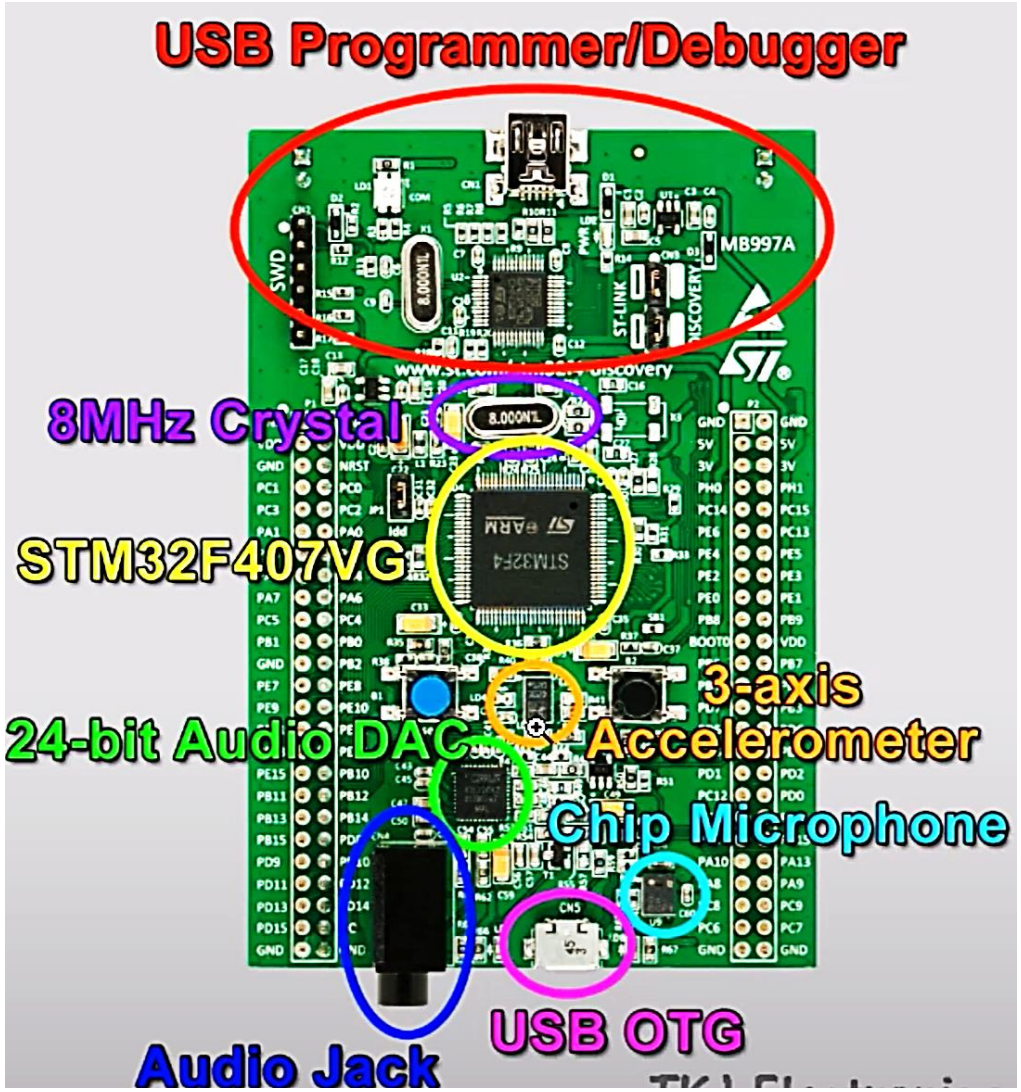
3.3V !!!



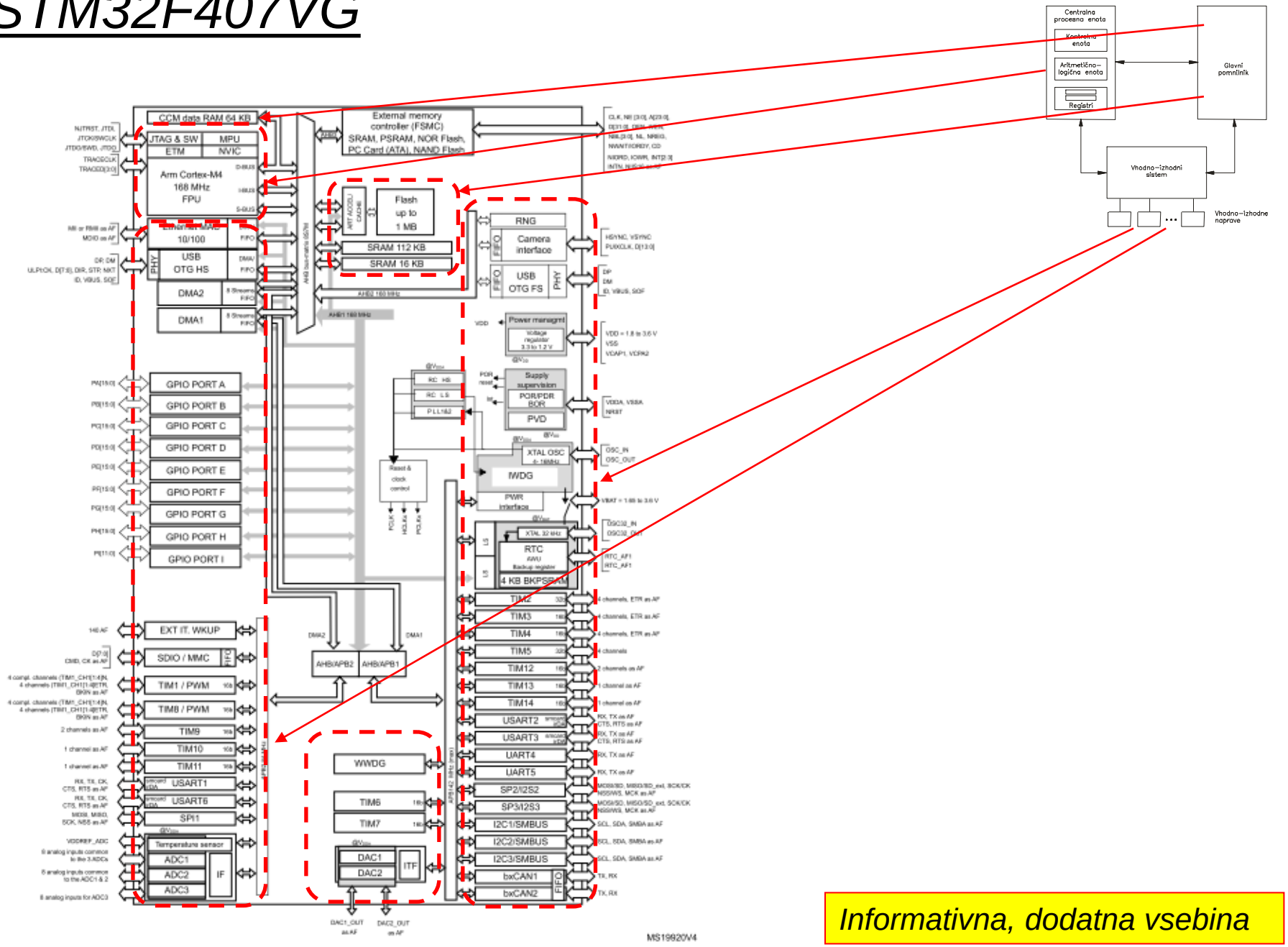
STM32F4-DISCOVERY

3.3V !!!

Razpored elementov (čipov) :



STM32F407VG



Informativna, dodatna vsebina

ARM Cortex M – ISA

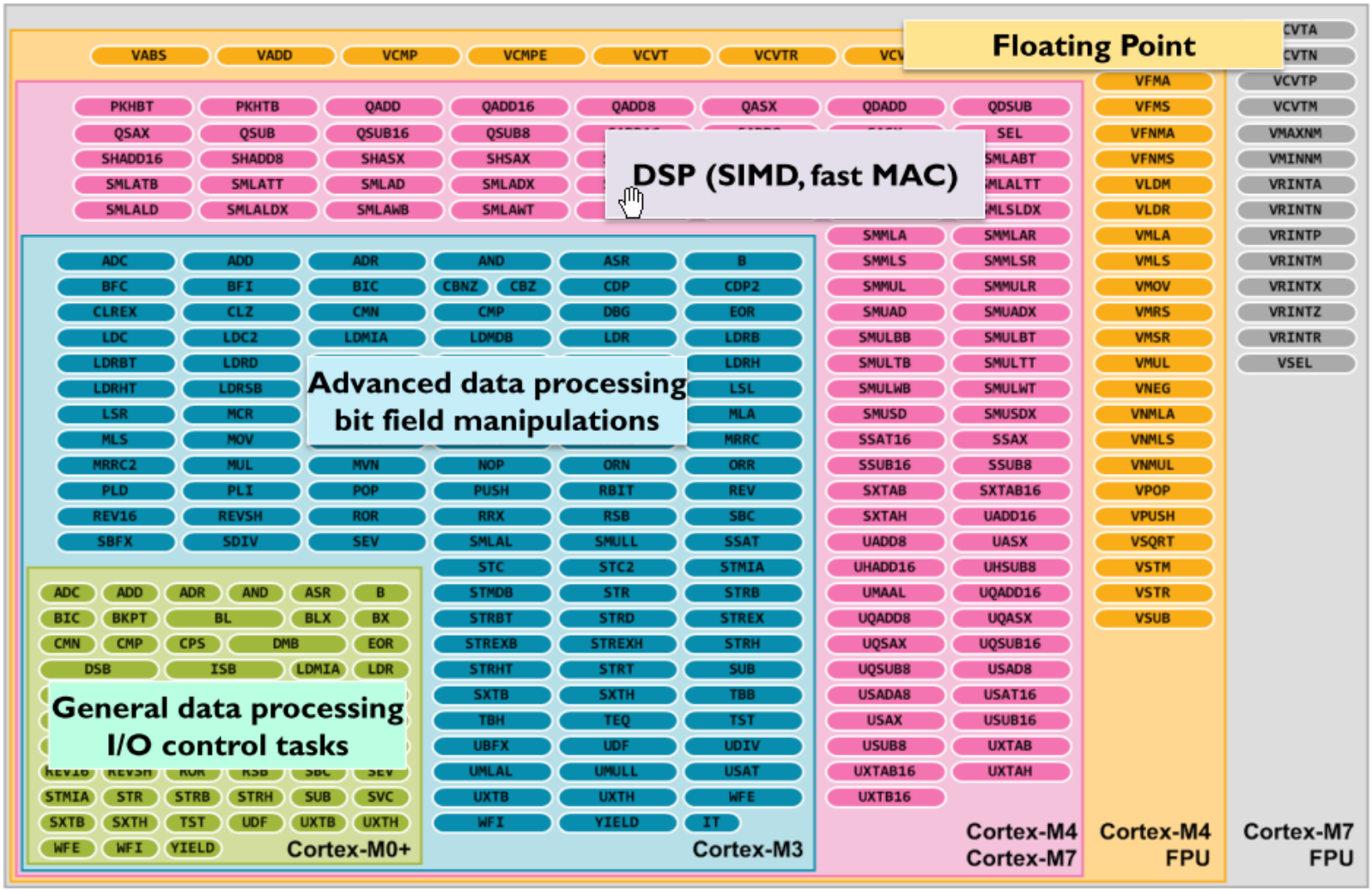
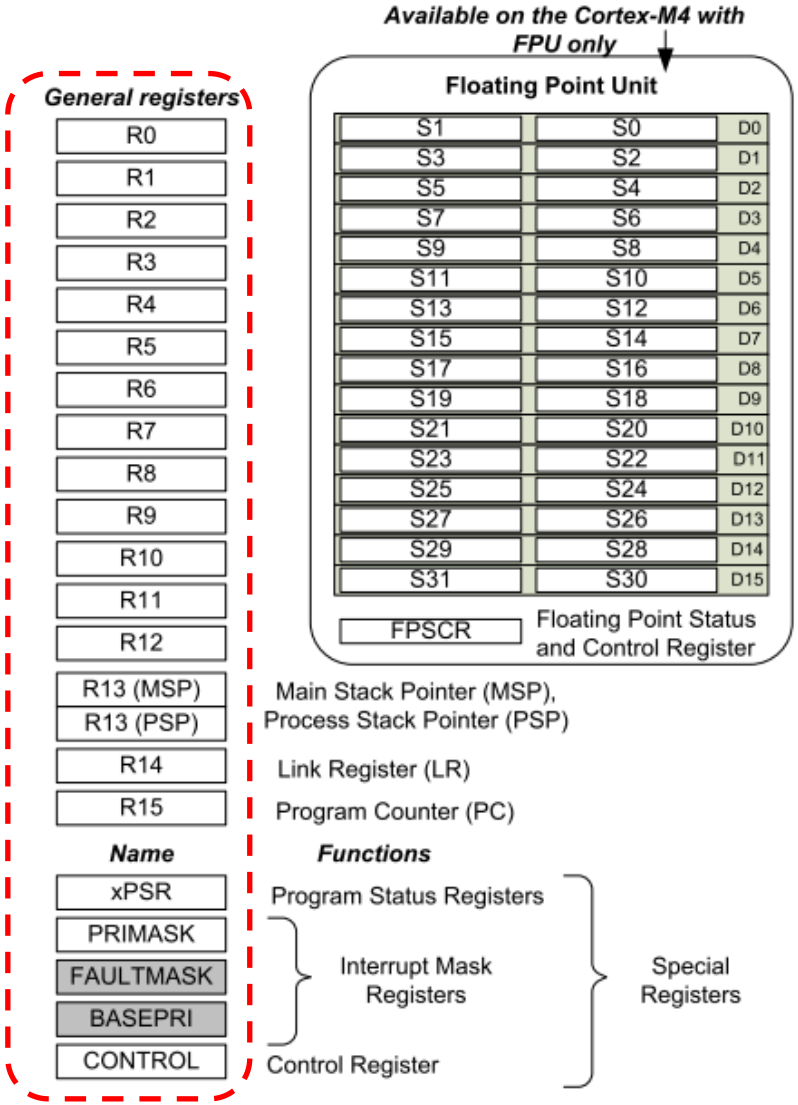


Figure 3: Instruction Set support in the Cortex-M processors

ARM Cortex M – Programski model



Delo na STM32F4 razvojnem sistemu – vklop in odklop

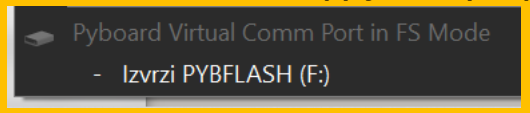
STLINK (Mini USB): napajanje, programiranje Flasha

Vklop in povezava s PC :

1. Povežemo STLINK
2. VCOM povežemo kasneje !

Izklop in prekinitev povezave s PC :

1. Izvrzi Flash Disk (.py skripte)



2. Odklop VCOM
3. Odklop STLINK

USB VCOM port (Micro USB): Micro-Python command line

Lastni viri (programiranje v zbirniku, Cju) :

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

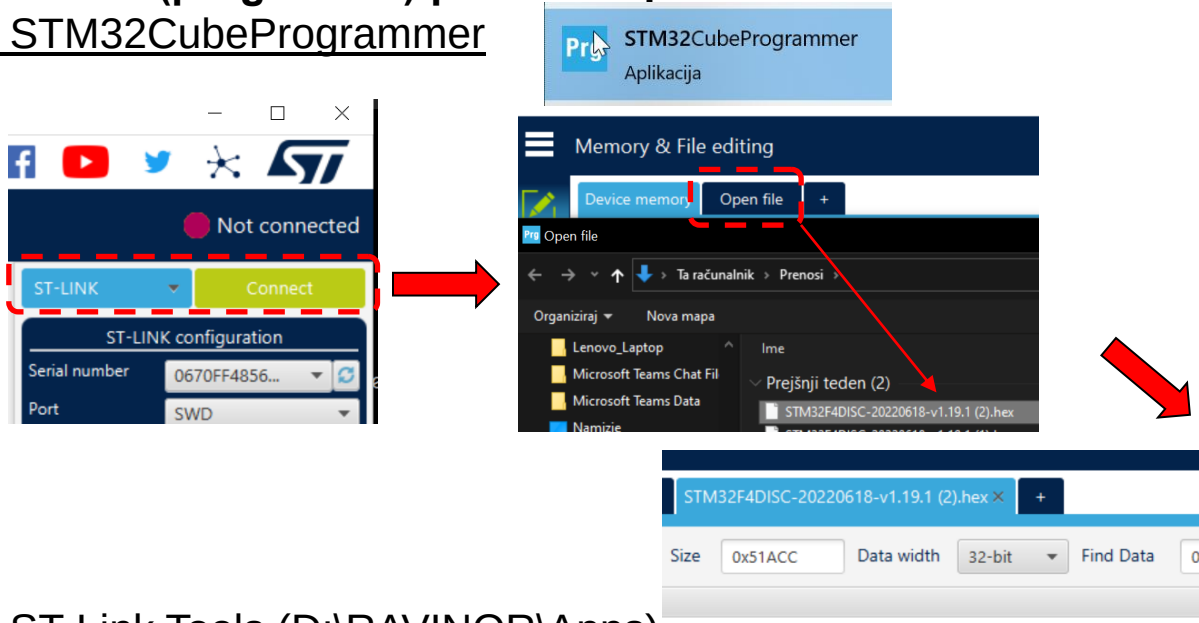
https://github.com/LAPSyLAB/ORLab-STM32

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects

Delo na plošči STM32F4 Discovery

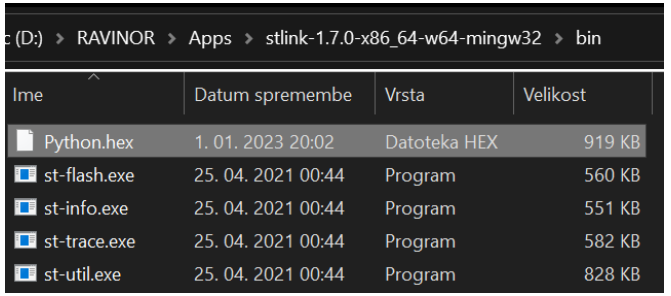
Micro Python :

- **Vpis v FLASH (programski) pomnilnik plošče STM32F4 – 2 možnosti:**
  - 1. STM32CubeProgrammer



  - 2. ST-Link Tools (D:\RAVINOR\Apps)
    - D:\RAVINOR\Apps\stlink-1.7.0-x86_64-w64-mingw32\bin>

st-flash.exe --format ihex write Python.hex



  - Datoteka (lokalno shranimo):
    - https://micropython.org/resources/firmware/STM32F4DISC-20220618-v1.19.1.hex
    - e-učilnica (datoteke za praktično LAB vajo)

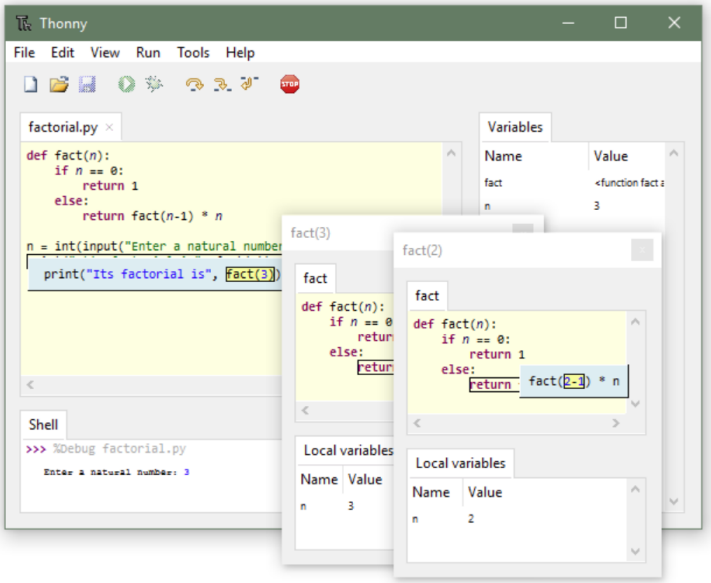
Delo na plošči STM32F4 Discovery

IDE za podporo Micro-Python in STM32F4 :

- ***Thonny (portable)***:
  - Podpira MicroPython
  - Enostaven za uporabo
  - Instaliramo „portable“ različico

Thonny
Python IDE for beginners

Download version 4.0.1 for Windows • Mac • Linux



Thonny

Download version 4.0.1 for Windows • Mac • Linux

Official downloads for Windows

Installer with 64-bit Python 3.10, requires 64-bit Windows 8.1 / 10 / 11
thonny-4.0.1.exe (20.4 MB) ⇐ *recommended for you*

Installer with 32-bit Python 3.8, suitable for all Windows versions since 7
thonny-py38-4.0.1.exe (18.9 MB)

NB! The installers have been signed with a new certificate which hasn't built up its reputation yet. You may need to click through your browser warning (e.g. choose "Keep" instead of "Discard" in Chrome) and Windows Defender warning (More info ⇒ Run anyway).

Portable variant with 64-bit Python 3.10
thonny-4.0.1-windows-portable.zip (30.5 MB)

Portable variant with 32-bit Python 3.8
thonny-py38-4.0.1-windows-portable.zip (28.6 MB)

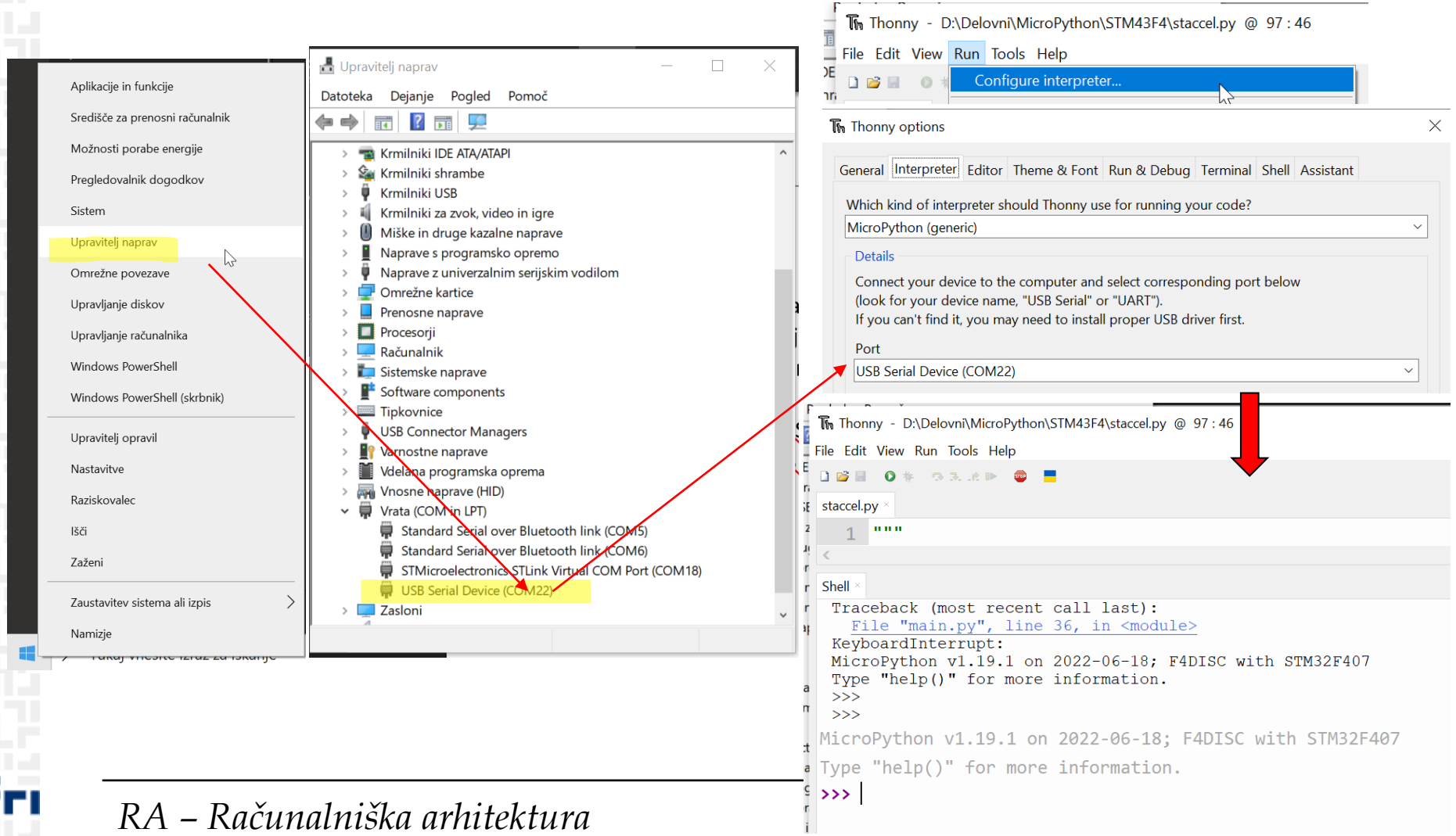
Re-using an existing Python installation (for advanced users)
`pip install thonny`

https://github.com/thonny/thonny/releases/download/v4.0.1/thonny-4.0.1-windows-portable.zip

Delo na plošči STM32F4 Discovery

Nastavitve za MicroPython in STM32F4 (Thonny IDE)

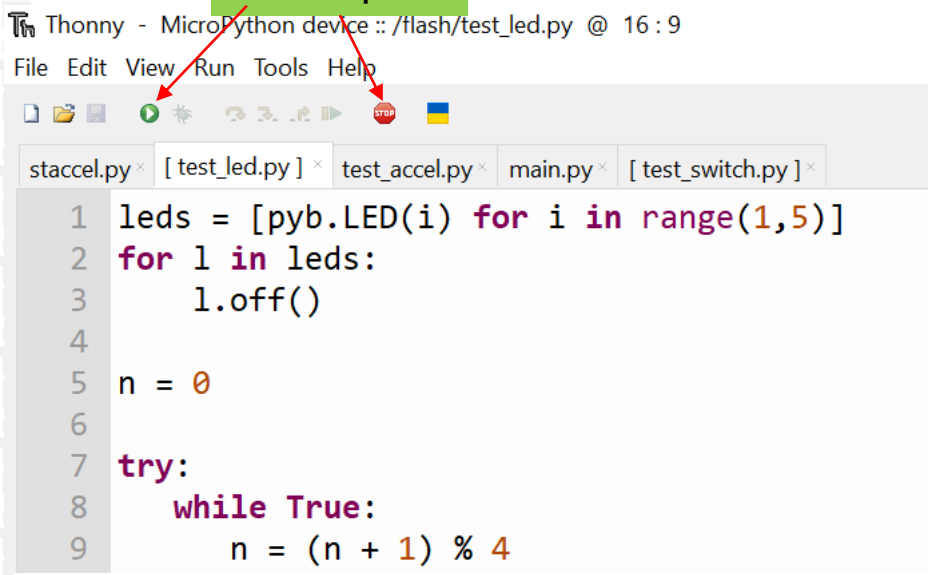
- ***Virtual COM port***
  - Command line vmesnik za MicroPython
  - Nastaviti v Thonny IDE številko COM porta

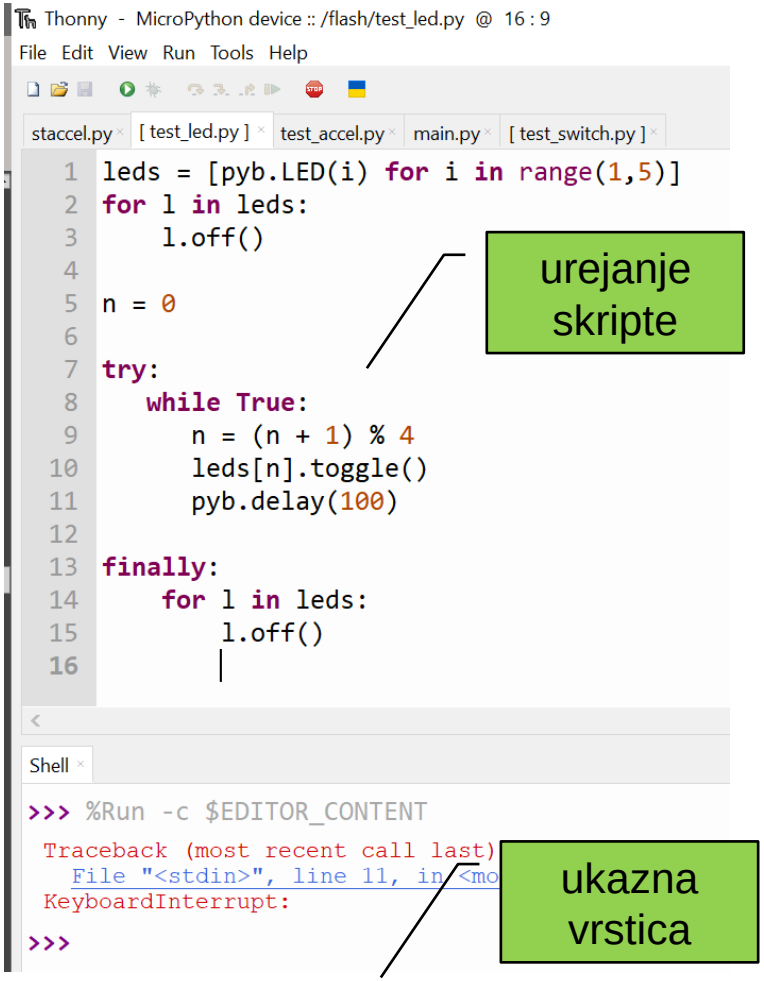


Delo na plošči STM32F4 Discovery

Delo: MicroPython, STM32F4 v Thonny IDE

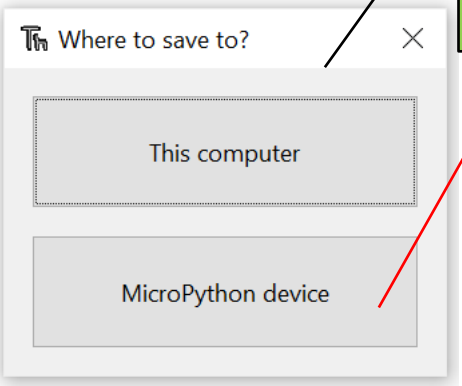
run/stop

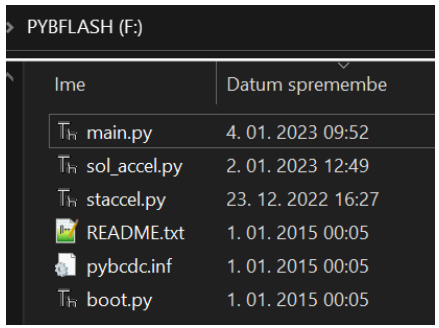




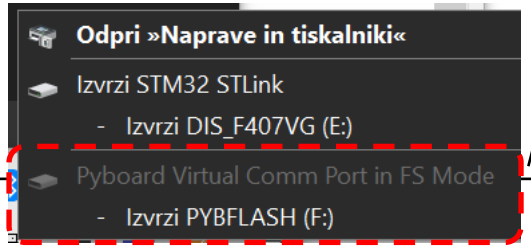
urejanje skripte

shranjevane py. skript





ukazna vrstica



Pozor: Izvrzi pred odklopom!!!

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Osnovni ukazi :

```
>>> led=pyb.LED(1)
>>> led.on()
>>> led.off()
>>> led.toggle()
```

```
>>> pyb.info()
ID=45004600:04513831:33393538
DEVID=0x0413
REVID=0x1007
S=168000000
H=168000000
P1=42000000
P2=84000000
_etext=806e174
_sidata=806e17c
_sdata=20000000
_edata=20000010
_sbss=20000010
_ebss=200022a0
_sstack=2001bff8
_estack=2001fff8
_ram_start=20000000
_heap_start=200022a0
_heap_end=2001bff8
_ram_end=20020000
qstr:
  n_pool=3
  n_qstr=62
  n_str_data_bytes=668
  n_total_bytes=1740
GC:
  103360 total
  73792 : 29568
  1=3788 2=47 m=46
LFS free: 84992 bytes
```

```
>>> help()
Welcome to MicroPython!

For online help please visit http://micropython.org/help/.

Quick overview of commands for the board:
  pyb.info()     -- print some general information
  pyb.delay(n)   -- wait for n milliseconds
  pyb.millis()   -- get number of milliseconds since hard reset
  pyb.Switch()   -- create a switch object
                    Switch methods: (), callback(f)
  pyb.LED(n)     -- create an LED object for LED n (n=1,2,3,4)
                    LED methods: on(), off(), toggle(), intensity(<n>)
  pyb.Pin(pin)   -- get a pin, eg pyb.Pin('X1')
  pyb.Pin(pin, m, [p]) -- get a pin and configure it for IO mode m, pull mode p
                    Pin methods: init(..), value([v]), high(), low()
  pyb.ExtInt(pin, m, p, callback) -- create an external interrupt object
  pyb.ADC(pin)   -- make an analog object from a pin
                    ADC methods: read(), read_timed(buf, freq)
  pyb.DAC(port)  -- make a DAC object
                    DAC methods: triangle(freq), write(n), write_timed(buf, freq)
  pyb.RTC()      -- make an RTC object; methods: datetime([val])
  pyb.rng()      -- get a 30-bit hardware random number
  pyb.Servo(n)   -- create Servo object for servo n (n=1,2,3,4)
                    Servo methods: calibration(..), angle([x, [t]]), speed([x, [t]])
  pyb.Accel()    -- create an Accelerometer object
                    Accelerometer methods: x(), y(), z(), tilt(), filtered_xyz()

Pins are numbered X1-X12, X17-X22, Y1-Y12, or by their MCU name
Pin IO modes are: pyb.Pin.IN, pyb.Pin.OUT_PP, pyb.Pin.OUT_OD
Pin pull modes are: pyb.Pin.PULL_NONE, pyb.Pin.PULL_UP, pyb.Pin.PULL_DOWN
Additional serial bus objects: pyb.I2C(n), pyb.SPI(n), pyb.UART(n)

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL-D        -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
```

Power related functions¶

```
>>> pyb.freq()
(168000000, 168000000, 42000000, 84000000)
```

pyb.freq([*sysclk*[, *hclk*[, *pclk1*[, *pclk2*]]]])¶

If given no arguments, returns a tuple of clock frequencies: (sysclk, hclk, pclk1, pclk2). These correspond to:

- sysclk: frequency of the CPU
- hclk: frequency of the AHB bus, core memory and DMA
- pclk1: frequency of the APB1 bus
- pclk2: frequency of the APB2 bus

Z naslova <https://docs.micropython.org/en/latest/library/pyb.html>

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Izzivi, teme:

- ***LED diode + tipka*:**
  - test_led.py
  - *Izziv1: sw_delay -> dvojna zanka ?*
  - test_switch.py
  - *Izziv2: svoja ideja ?*

- **Pospeškomer**
  - test_accel.py
  - *Izziv: vklop diode glede na naklon plošče ?*

- **Timer, Buzzer**
  - priklop brenčača na PB8 in GND
  - „Kuža pazi“: c-d-e-d-c (c=262Hz, d=290Hz, e=330Hz)
  - test_buzzer_pwm.py
  - *Izziv: melodija ?*

- **Vgrajeni zbirnik (Inline assembler) - rekurzija**
  - test_fibonacci.py (rekurzivni izračun)
  - https://docs.micropython.org/en/latest/reference/asm_thumb2_hints_tips.html

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Izzivi, teme:

- ***LED diode + tipka*:**
  - test_led.py
  - *Izziv1: sw_delay -> dvojna zanka ?*
  - test_switch.py
  - *Izziv2: svoja ideja ?*

```
leds = [pyb.LED(i) for i in range(1,5)]
for l in leds:
    l.off()

n = 0

try:
   while True:
      n = (n + 1) % 4
      leds[n].toggle()
      pyb.delay(100)

finally:
    for l in leds:
        l.off()
```

```
@micropython.asm_thumb

def ms_delay(r0):

    label(LOOP_MSEC)
#    movwt(r1, 42000)  # pseudo instruction to move 32b constant into register.
                       # Equivalent of LDR rd,=imm32
    movwt(r1, 28000)   # it seems that inline assembler is slower by 1/3
    label(LOOP)
    sub(r1, r1, 1)
    bne (LOOP)
    sub (r0, r0, 1)
    bne (LOOP_MSEC)
```

```
def ButtonPressed():
    pyb.LED(1).toggle()
    print('Button press!')

sw = pyb.Switch()

sw.callback(ButtonPressed)

while True:

      pyb.delay(500)
      print('Button state [%d] ' % sw.value())
```

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Izzivi, teme:

- **Pospeškomer**
  - potreben tudi staccel.py (modul)
  - test_accel.py

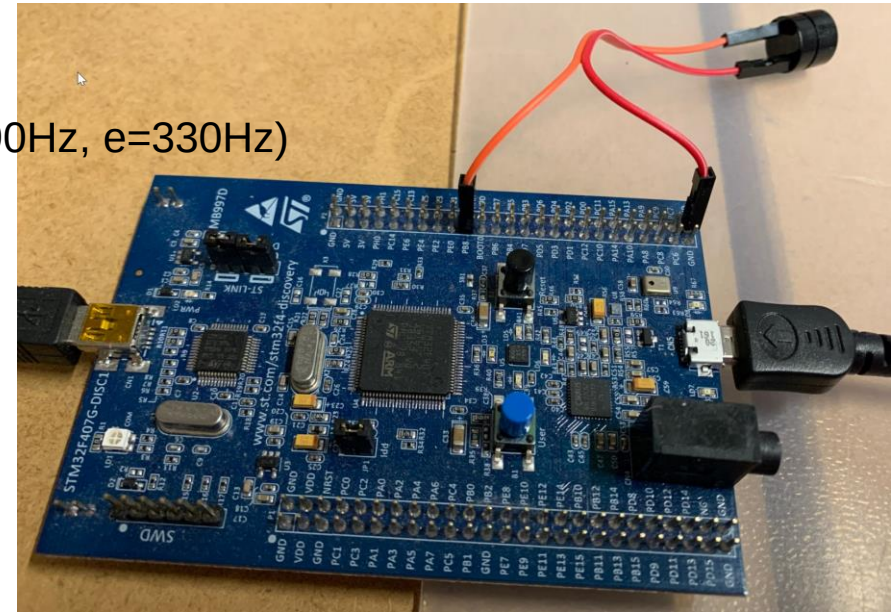
```
from staccel import STAccel
accel = STAccel()

while True:
    x = accel.x()
    y = accel.y()
    xyz=accel.xyz()

    print('Accelerator [ID=0x%x] ' % accel.read_id(),xyz)

    pyb.delay(500)
```

- *Izziv: vklop diode glede na naklon plošče ?*
  - *plošča ima 4 LED diode – vsako na svoji strani*
  - *Napišite program, ki prižge ustrezno LED diodo pri dovolj velikem naklonu v smeri lege LED diode.*

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Izzivi, teme:

- **Timer, Buzzer**
  - priklop brenčača na PB8 in GND
  - „Kuža pazi“: c-d-e-d-c (c=262Hz, d=290Hz, e=330Hz)
  - test_buzzer_pwm.py
  - *Izziv: druga melodija ?*



```
# create Timer (use TIM4)
tim = pyb.Timer ( 4 , prescaler = 83, period = 4545 ) # 220 Hz (for A3) : timer clock to 1MHz

# Kuza pazi c-d-e-d-c  (C=262Hz, D=290Hz, E=330Hz)
freqs=[262,262,262,262,290,290,290,290,330,330,290,290,262,262,262]
n=0

try:
   while True:
      tim.init (prescaler = 83, period = 4545) # timer clock to 1MHz
      # Choose PB8 pin for TIM4_CH3 or PB9 pin for TIM4_CH4 for micro secs
      pwm = tim.channel ( 3 , pyb.Timer.PWM , pin = pyb.Pin.board.PB8 , pulse_width = 0 )

      tim.period(int(1000000/(freqs[n])))
      pwm.pulse_width (tim.period() // 2 ) # 50% duty cycle

      print('Tone frequency[%d]: %d' % (n,freqs[n]))
      print ( 'frequency : {:>8} [Hz]' . format ( tim . freq ()) )
      print ( 'period : {:>8} [us]' . format ( tim . period ()) )
      print ( 'pulse width : {:>8} [us]' . format ( pwm . pulse_width ()) )
      pyb.delay(500)

      n = ((n + 1) % 15)
      tim.deinit()
      pyb.delay(100)


finally:
    tim.deinit()
    print ( 'Done' )
```

Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Izzivi, teme:

- **Vgrajeni zbirnik (Inline assembler) - rekurzija**
  - test_fibonacci.py (rekurzivni izračun)
  - https://docs.micropython.org/en/latest/reference/asm_thumb2_hints_tips.html

test_fibonacci.py:

```
@micropython.asm_thumb
def fib(r0):
    b(START)
    label(DOFIB)
    push({r1, r2, lr})
    cmp(r0, 1)
    bls(FIBDONE)
    sub(r0, 1)
    mov(r2, r0) # r2 = n -1
    bl(DOFIB)
    mov(r1, r0) # r1 = fib(n -1)
    sub(r0, r2, 1)
    bl(DOFIB)   # r0 = fib(n -2)
    add(r0, r0, r1)
    label(FIBDONE)
    pop({r1, r2, lr})
    bx(lr)
    label(START)
    bl(DOFIB)

for n in range(10):
    print(fib(n))
```

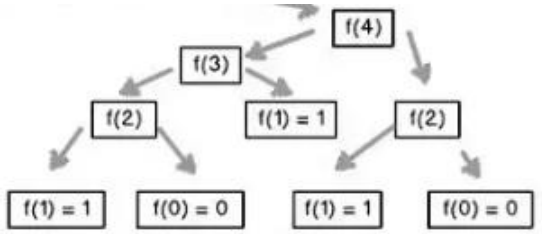
ARM assembly rewrite (CPUlator demo link below):

```
            mov   r0,#5
            bl    DOFIB
_end:       b     _end        @ r0 has the result !!!

DOFIB:      push  {r1, r2, lr}

            cmp   r0, #1
            bls   FIBDONE

            sub   r0, r0, #1
            mov   r2, r0      @ r2 = n -1
            bl    DOFIB
            mov   r1, r0      @ r1 = fib(n -1)
            sub   r0, r2, #1
            bl    DOFIB       @ r0 = fib(n -2)
            add   r0, r0, r1  @ r0=fib(n -1)+fib(n -2)

FIBDONE:    pop   {r1, r2, lr}
            bx    lr
```

https://cpulator.01xz.net/?sys=arm&loadasm=share/sHaGQiX.s

```
def fibonacci(number)
  if number <= 1
    number
  else
    fibonacci(number - 1) + fibonacci(number - 2)
  end
end
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55



Delo na plošči STM32F4 Discovery + MicroPython (Thonny IDE)

Dodatni viri (pyboard ni enak STM32F4) :

pyb – functions related to the board¶

The pyb module contains specific functions related to the board.

Z naslova <https://docs.micropython.org/en/latest/library/pyb.html>

stm – functionality specific to STM32 MCUs¶

This module provides functionality specific to STM32 microcontrollers, including direct access to peripheral registers.

```
>>> import stm
>>> help(stm)
```

Z naslova <https://docs.micropython.org/en/latest/library/stm.html>

MicroPython tutorial for the pyboard

This tutorial is intended to get you started with your pyboard. All you need is a pyboard and a micro-USB cable to connect it to your PC. If it is your first time, it is recommended to follow the tutorial through in the order below.

1. Introduction to the pyboard
2. Running your first script
3. Getting a MicroPython REPL prompt
4. Turning on LEDs and basic Python concepts
5. The Switch, callbacks and interrupts
6. The accelerometer
7. Safe mode and factory reset
8. Making the pyboard act as a USB mouse
9. The Timers
10. Inline assembler
11. Power control

Tutorials requiring extra components

1. Controlling hobby servo motors
2. Fading LEDs
3. The LCD and touch-sensor skin
4. The AMP audio skin
5. The LCD160CR skin

Tips, tricks and useful things to know

1. Debouncing a pin input
2. Making a UART - USB pass through

Z naslova <https://docs.micropython.org/en/v1.9.3/pyboard/pyboard/tutorial/index.html>

Dodatni izzivi

- **Krajšanje periode vklopa/izklopa (LED, brenčač)**
- **Spreminjanje razmerja časa vklopa/izklopa (LED dimmer)**