MicroPython WeAct Core Board 使用教程

2020-01-01 | 2020-02-07 | STM32 | 76

Micropython

MicroPython，是Python3编程语言的一个完整软件实现，用C语言编写，被优化于运行在微控制器之上。MicroPython是运行在微控制器硬件之上的完全的Python编译器和运行时系统。提供给用户一个交互式提示符（REPL）来立即执行所支持的命令。除了包括选定的核心Python库，MicroPython还包括了给予编程者访问低层硬件的模块。

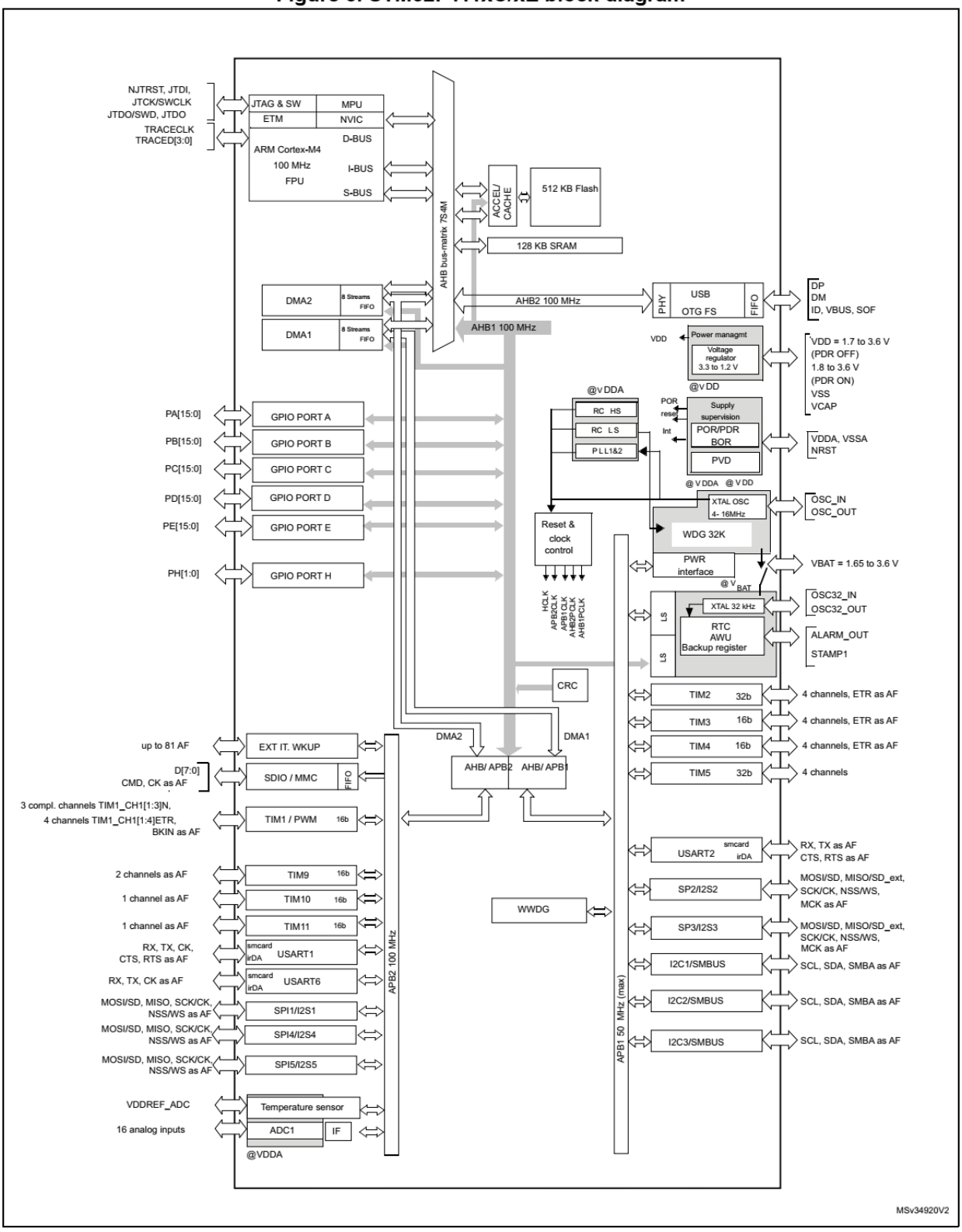
具体请参考：https://micropython.org/

核心板外设定义及使用

STM32F411CE 内部框图

> 核心板引脚丝印省略字母`P`，例如`A0`为`PA0`，micropython使用的IO统一为PAx,PBx,PCx...

Figure 3. STM32F411xC/xE block diagram



1. The timers connected to APB2 are clocked from TIMxCLK up to 100 MHz, while the timers connected to APB1 are clocked from TIMxCLK up to 100 MHz.

stm32f411CE内部框架

GPIO

pyb.Pin

定义

> PA0-PA15,PB0-PB10,PB12-PB15,PC13-PC15
> Pin.IN - configure the pin for input;
> Pin.OUT_PP - configure the pin for output, with push-pull control;
> Pin.OUT_OD - configure the pin for output, with open-drain control;
> Pin.AF_PP - configure the pin for alternate function, pull-pull;
> Pin.AF_OD - configure the pin for alternate function, open-drain;
> Pin.ANALOG - configure the pin for analog.
> pull can be one of:
> Pin.PULL_NONE - no pull up or down resistors;
> Pin.PULL_UP - enable the pull-up resistor;
> Pin.PULL_DOWN - enable the pull-down resistor;

使用

GPIO

```
from pyb import Pin
C13 = Pin('PC13',Pin.OUT_PP,Pin.PULL_NONE) # PC13推挽输出 无上下拉
C13 # REPL 打印PC13 配置
C13.high() # 输出高电平
C13.value() # 读取PC13电平
```

External interrupts 外部中断

定义

ExtInt.IRQ_RISING

ExtInt.IRQ_FALLING

ExtInt.IRQ_RISING_FALLING

pyb.Pin.PULL_NONE

pyb.Pin.PULL_UP

pyb.Pin.PULL_DOWN

使用

GPIO 外部中断

```
from pyb import Pin, ExtInt

callback = lambda e: print("PA0 <KEY> intr")
ext = ExtInt(Pin('PA0'), ExtInt.IRQ_RISING, pyb.Pin.PULL_NONE, callback)
ext.swint() # 手动触发回调 运行一次callback()
```

ADC

pyb.adc

定义

STM32F411 I/O定义

ADC0

> 外部通道：`PA0,PA1,PA2,PA3,PA4,PA5,PA6,PA7,PB0,PB1`
>
> 内部通道：Temperature,VBAT,VFEF

使用

ADC 外部通道

```
from pyb import Pin,ADC
IN0  = ADC(Pin('PA0'))
IN0.read()

IN1  = ADC(Pin('PA1'))
IN1.read()
```

ADC 内部通道

```
from pyb import Pin,ADC
adc_in = pyb.ADCAll(12, 0x70000) # 12 bit resolution, internal channels
adc_in.read_core_temp() # Temperature
adc_in.read_core_vbat() # VBAT
adc_in.read_vref() # VFEF
```

Timer 定时器

pyb.Timer

定义

STM32F411 I/O定义

STM32F4x1C 定时器引脚分布

当使用USB时，PA11已被占用

Timer	Channel 1	Channel 2	Channel 3	Channel 4
TIM1	PA8	PA9	PA10	PA11
TIM2	PA0, PA5, PA15	PA1, PB3	PA2,PB10	PA3
TIM3	PA6, PB4	PA7, PB5	PB0	PB1
TIM4	PB6	PB7	PB8	PB9
TIM5	PA0	PA1	PA2	PA3
TIM9	PA2	PA3		
TIM10	PB8			
TIM11	PB9			

定时器输入时钟频率：

> 96Mhz(411)/84Mhz(401):TIM1,TIM9,TIM10,TIM11
>
> 48Mhz(411)/42Mhz(401):TIM2,TIM3,TIM4,TIM5

使用

Timer 定时功能

```
from pyb import Timer
TIM1 = Timer(1, freq=2) # 频率：2Hz
TIM1.counter() # get counter value
TIM1.callback(lambda t: pyb.LED(1).toggle())
TIM1.freq(5) # 更改定时器频率 5 Hz
TIM1.deinit() # 取消使用TIM1
```

Timer PWM功能

```
from pyb import Pin, Timer
TIM2 = Timer(2, freq=1000) # 频率：1KHz
# Timer.PWM_INVERTED：占空比：  0%->高电平 100%->低电平
# Timer.PWM ：占空比：  0%->低电平 100%->高电平
TIM2_CH1 = Pin('PA0',Pin.PULL_NONE) # PA0 : TIM2, CH1, 无上下拉
TIM2_CH1 = TIM2.channel(1, Timer.PWM, pin=TIM2_CH1,pulse_width_percent = 0) # 占空比 0%
TIM2_CH1.pulse_width_percent(50) # 50% 占空比,0 - 100%
TIM2_CH1.pulse_width(int(TIM2.period()/2)) # 直接设置比较值 50% 占空比
```

Timer 输入捕获功能

polarity can be one of:

> Timer.RISING - 上升沿捕获
>
> Timer.FALLING - 下降沿捕获
>
> Timer.BOTH - 上升/下降沿都捕获.

```
from pyb import Pin, Timer
import pyb

TIM1 = Timer(1,prescaler=96-1, period=65535-1,mode=Timer.UP) # 96分频：1Mhz,重装载值(周期
TIM1_CH1 = Pin('PA8',Pin.PULL_NONE) # PA8 : TIM1, CH1, 无上下拉
TIM1_CH1 = TIM1.channel(1, Timer.IC,polarity=Timer.BOTH, pin=TIM1_CH1)
TIM1_CH1.callback(lambda t: print(TIM1_CH1.capture()))
pyb.delay(5000)
TIM1_CH1.callback(None)
TIM1.deinit()
print('TIM over')
```

RTC (Real Time Clock)

pyb.RTC

定义

> 无。

使用

```
from pyb import RTC

RTC = RTC()
```

```
RTC.datetime((2020, 1, 1, 3, 12, 00, 0, 0)) # 设置时间 2020.01.01 周三,12:00:00
RTC.datetime() # 获取时间
hex(RTC.info()) # 获取有关启动时间和重置源的信息
callback = lambda e: print("RTC WekeUP")
RTC.wakeup(2000,callback)
# pyb.stop() # 停止模式，如果当前使用USB连接，USB CDC 会进入假死状态，
# RTC.wakeup(0)
pyb.standby() # 如果当前使用USB连接，USB CDC 会断开连接，恢复时会自动复位
```

UART 串口

pyb.UART

定义

> USB转串口 交叉连接

STM32F411 I/O定义

> UART1: `TX`-PA9, `RX`-PA10
>
> UART2: `TX`-PA2, `RX`-PA3
>
> UART6: `TX`-PA11, `RX`-PA12 (`USB`占用)
>
> UART_REPL: UART1

使用

UART2 串口

```
from pyb import UART

uart2 = UART(2, 9600)
uart2.write('hello')
uart2.read(5) # 从数据流里读取5个字节，没有则返回None
uart2.readline() # 读取一行，以换行符结尾
uart2.any()   # 返回可以读取的字节数
```

SPI 总线

pyb.SPI

定义

STM32F411 I/O定义

> SPI2: `NSS`-PB12, `SCK`-PB13, `MISO`-PB14, `MOSI`-PB15
>
> SPI4: `NSS`-PB12, `SCK`-PB13, `MISO`-PA1, `MOSI`-PA11 (`USB`占用)
>
> SPI5: `NSS`-PB1, `SCK`-PA10, `MISO`-PA12, `MOSI`-PB0 (`USB`占用)

使用

```
from pyb import SPI

spi2 = SPI(2, SPI.MASTER, baudrate=200000, polarity=1, phase=0)
spi2.send('hello')
spi2.recv(5) # receive 5 bytes on the bus
spi2.send_recv('hello') # send and receive 5 bytes
```

I2C 总线

machine.I2C

定义

STM32F411 I/O定义

> I2C1: `SCL`-PB6, `SDA`-PB7
>
> I2C2: `SCL`-PB10, `SDA`-PB9
>
> I2C3: `SCL`-PA8, `SDA`-PB8

使用

```
from machine import I2C

i2c = I2C(1, freq=400000)                 # create hardware I2c object: I2C1,PB6,PB7
i2c = I2C(scl='PB6', sda='PB7', freq=100000)  # create software I2C object,软件I2c

i2c.scan()                            # returns list of slave addresses
i2c.writeto(0x42, 'hello')            # write 5 bytes to slave with address 0x42
i2c.readfrom(0x42, 5)                 # read 5 bytes from slave

i2c.readfrom_mem(0x42, 0x10, 2)       # read 2 bytes from slave 0x42, slave memory 0x10
i2c.writeto_mem(0x42, 0x10, 'xy')     # write 2 bytes to slave 0x42, slave memory 0x10
```

Servo Control 舵机控制

pyb.Servo

定义

> Servo模块使用了TIM5,故两者不能同时使用，只能二选一。
>
> 舵机ID(1-4)对应引脚：PA0,PA1,PA2,PA3

使用

```
from pyb import Servo
s1 = Servo(1) # servo on position 1 (PA0, VIN, GND)
s1.angle(45) # move to 45 degrees
s1.angle(-60, 1500) # move to -60 degrees in 1500ms
s1.speed(50) # for continuous rotation servos
```

Switch 按键

pyb.Switch

定义

> STM32F4x1Cx核心板 按键为(`KEY`): PA0

使用

```
from pyb import Switch

sw = Switch()
sw.value() # returns True or False
sw.callback(lambda: pyb.LED(1).toggle())
```

LED

pyb.LED

定义

> STM32F4x1 核心板 蓝色LED（`C13`): PC13

使用

```
from pyb import LED

led = LED(1) # 1=blue
led.toggle()
led.on()
led.off()
```

WeAct

一个致力于设计独一无二电子模块的工作室

STM32 # 2020 # micropython

STM32 下载烧录问题汇总

New Boards Coming Soon