三方组件API文档

中断模块

- 概述
  - 支持了异步的发布订阅式的中断, 用户只需要关注其对应的topic消息
  - 支持原生封装接口的二次开发, 和现有接口的封装
- 功能
  - 封装了中断和中断+定时器的调用
  - 对外只提供
    - 初始化
    - 开启功能
    - 关闭功能

BusExIntInterrupter

- 概述
  - 支持发布订阅式的中断的处理
  - 我们初始化的时候传入一个字符串topic, 然后一个函数通过sys_bus去订阅这个topic, 当发生对应的中断是, 我们订阅的函数, 将会受到对应中断的消息
  - 优点
    - 用户无需关心, 回到和维护callback
    - topic可以批量订阅,并且是异步形式, 不影响当前其他线程
    - 此接口支持自定义topic类型传递, 用户可以根据此类进行继承,并且二次定制
  - 缺点
    - 自己定义topic, 用户需要维护那个topic, 形成了一定的强关联
- 功能
  - 继承于ExIntInterrupter, 支持了中断的功能, 用户可以自定义topic, 拥有发布功能
  - 功能包含ExIntInterrupter所有功能
  - 新增功能
    - 支持自定义topic后发布相关的事件
  - 对外只提供
    - 初始化
    - 开启功能
    - 关闭功能

创建对象

> **BusExIntInterrupter(topic, gpio, trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE)**

- 参数

参数	类型	是否必传	说明
topic	string	必传	接收到中断索要发发布的topic
	GPIOn	必传项	引脚号 EC100YCN平台引脚对应关系如下（引脚号为外部引脚编号）： GPIO1 – 引脚号22 GPIO2 – 引脚号23 GPIO3 – 引脚号38 GPIO4 – 引脚号53 GPIO5 – 引脚号54 GPIO6 – 引脚号104 GPIO7 – 引脚号105 GPIO8 – 引脚号106 GPIO9 – 引脚号107 GPIO10 – 引脚号178 GPIO11 – 引脚号195 GPIO12 – 引脚号196 GPIO13 – 引脚号197 GPIO14 – 引脚号198 GPIO15 – 引脚号199 GPIO16 – 引脚号203 GPIO17 – 引脚号204 GPIO18 – 引脚号214 GPIO19 – 引脚号215 EC600SCN/EC600NCN平台引脚对应关系如下（引脚号为模块外部引脚编号）： GPIO1 – 引脚号10 GPIO2 – 引脚号11 GPIO3 – 引脚号12 GPIO4 – 引脚号13 GPIO5 – 引脚号14 GPIO6 – 引脚号15 GPIO7 – 引脚号16 GPIO8 – 引脚号39 GPIO9 – 引脚号40 GPIO10 – 引脚号48 GPIO11 – 引脚号58 GPIO12 – 引脚号59 GPIO13 – 引脚号60 GPIO14 – 引脚号61 GPIO15 – 引脚号62 GPIO16 – 引脚号63 GPIO17 – 引脚号69 GPIO18 – 引脚号70 GPIO19 – 引脚号1

参数	类型	是否必传	说明
			GPIO20 – 引脚号3 GPIO21 – 引脚号49 GPIO22 – 引脚号50 GPIO23 – 引脚号51 GPIO24 – 引脚号52 GPIO25 – 引脚号53 GPIO26 – 引脚号54 GPIO27 – 引脚号55 GPIO28 – 引脚号56 GPIO29 – 引脚号57 EC600UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号61 GPIO2 – 引脚号58 GPIO3 – 引脚号34 GPIO4 – 引脚号60 GPIO5 – 引脚号69 GPIO6 – 引脚号70 GPIO7 – 引脚号123 GPIO8 – 引脚号118 GPIO9 – 引脚号9 GPIO10 – 引脚号1 GPIO11 – 引脚号4 GPIO12 – 引脚号3 GPIO13 – 引脚号2 GPIO14 – 引脚号54 GPIO15 – 引脚号57 GPIO16 – 引脚号56 EC200UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号27 GPIO2 – 引脚号26 GPIO3 – 引脚号24 GPIO4 – 引脚号25 GPIO5 – 引脚号13 GPIO6 – 引脚号135 GPIO7 – 引脚号136 GPIO8 – 引脚号133 GPIO9 – 引脚号3 GPIO10 – 引脚号40 GPIO11 – 引脚号37 GPIO12 – 引脚号38 GPIO13 – 引脚号39 GPIO14 – 引脚号5

参数	类型	是否必传	说明
			GPIO15 – 引脚号141 GPIO16 – 引脚号142
trige_mode	int	非必传(默认 ExtInt.IRQ_RISING)	设置触发方式 IRQ_RISING – 上升沿触发 IRQ_FALLING – 下降沿触发 IRQ_RISING_FALLING – 上升和下降沿触发
pull_mode	int	非必传(默认 ExtInt.PULL_DISABLE)	PULL_DISABLE – 浮空模式 PULL_PU – 上拉模式 PULL_PD – 下拉模式
callback	int	必传项	中断触发回调函数

初始化示例

- 初始化可发布订阅的处理器

```
>>> from ql_interrupter import BusExIntInterrupter
>>> from machine import Pin, ExtInt
>>> extint = BusExIntInterrupter("topic1", Pin.GPIO14,
trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE,
callback=callback)
```

开启中断

使能extint对象外部中断，当中断引脚收到上升沿或者下降沿信号时，会调用callback执行 。

> **extint.start()**

- 参数

  无

- 返回值

  使能成功返回整型值0，使能失败返回整型值-1。

关闭中断

> **extint.stop()**

禁用与extint对象关联的中断 。

- 参数

无

- 返回值

使能成功返回整型值0，使能失败返回整型值-1。

订阅

订阅的函数的函数参数说明

参数	类型	说明
topic	string	收到的topic
msg	dict	例{'gpio': 18, 'pressure': 0}, gpio代表使能的gpio引脚号 pressure代表1代表高电平, 0代表低电平

```
import sys_bus
from ql_interrupter import BusExIntInterrupter
from machine import Pin, ExtInt

# 初始化中断处理器
extint = BusExIntInterrupter("topic1", Pin.GPIO14)
# 开启中断
extint.start()

# 订阅函数
def callback(topic, msg):
    pass

# 订阅topic
sys_bus.subscribe("topic1", callback)
```

ExIntProcess

- 概述
  - 继承于BusExIntInterrupter具有bus和ExInt的功能
  - 区别是相对于BusExIntInterrupter相比, 用户需要手动传入topic, 维护topic
  - 此类不需要用户传入topic, 用户只需关注固定的topic即可, 订阅固定的topic, 当中断发生时, 对应的topic将收到相关的消息
  - 对应的topic规则是如下:
    - GPIO1_EXINT
    - GPIO2_EXINT

        - GPIO?_EXINT
    - 更多详细说明请看订阅章节
- 功能
    - 不用传入topic模式
    - 用户只需关注topic事件

创建对象

> **ExIntProcess(gpio, trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE)**

参数	类型	是否必传	说明
	GPIOn	必传项	引脚号 EC100YCN平台引脚对应关系如下（引脚号为外部引脚编号）： GPIO1 – 引脚号22 GPIO2 – 引脚号23 GPIO3 – 引脚号38 GPIO4 – 引脚号53 GPIO5 – 引脚号54 GPIO6 – 引脚号104 GPIO7 – 引脚号105 GPIO8 – 引脚号106 GPIO9 – 引脚号107 GPIO10 – 引脚号178 GPIO11 – 引脚号195 GPIO12 – 引脚号196 GPIO13 – 引脚号197 GPIO14 – 引脚号198 GPIO15 – 引脚号199 GPIO16 – 引脚号203 GPIO17 – 引脚号204 GPIO18 – 引脚号214 GPIO19 – 引脚号215 EC600SCN/EC600NCN平台引脚对应关系如下（引脚号为模块外部引脚编号）： GPIO1 – 引脚号10 GPIO2 – 引脚号11 GPIO3 – 引脚号12 GPIO4 – 引脚号13 GPIO5 – 引脚号14 GPIO6 – 引脚号15 GPIO7 – 引脚号16 GPIO8 – 引脚号39 GPIO9 – 引脚号40 GPIO10 – 引脚号48 GPIO11 – 引脚号58 GPIO12 – 引脚号59 GPIO13 – 引脚号60 GPIO14 – 引脚号61 GPIO15 – 引脚号62 GPIO16 – 引脚号63 GPIO17 – 引脚号69 GPIO18 – 引脚号70 GPIO19 – 引脚号1 GPIO20 – 引脚号3

参数	类型	是否必传	说明
			GPIO21 – 引脚号49 GPIO22 – 引脚号50 GPIO23 – 引脚号51 GPIO24 – 引脚号52 GPIO25 – 引脚号53 GPIO26 – 引脚号54 GPIO27 – 引脚号55 GPIO28 – 引脚号56 GPIO29 – 引脚号57 EC600UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号61 GPIO2 – 引脚号58 GPIO3 – 引脚号34 GPIO4 – 引脚号60 GPIO5 – 引脚号69 GPIO6 – 引脚号70 GPIO7 – 引脚号123 GPIO8 – 引脚号118 GPIO9 – 引脚号9 GPIO10 – 引脚号1 GPIO11 – 引脚号4 GPIO12 – 引脚号3 GPIO13 – 引脚号2 GPIO14 – 引脚号54 GPIO15 – 引脚号57 GPIO16 – 引脚号56 EC200UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号27 GPIO2 – 引脚号26 GPIO3 – 引脚号24 GPIO4 – 引脚号25 GPIO5 – 引脚号13 GPIO6 – 引脚号135 GPIO7 – 引脚号136 GPIO8 – 引脚号133 GPIO9 – 引脚号3 GPIO10 – 引脚号40 GPIO11 – 引脚号37 GPIO12 – 引脚号38 GPIO13 – 引脚号39 GPIO14 – 引脚号5 GPIO15 – 引脚号141 GPIO16 – 引脚号142

参数	类型	是否必传	说明
trige_mode	int	非必传(默认 ExtInt.IRQ_RISING)	设置触发方式 IRQ_RISING – 上升沿触发 IRQ_FALLING – 下降沿触发 IRQ_RISING_FALLING – 上升和下降沿触发
pull_mode	int	非必传(默认 ExtInt.PULL_DISABLE)	PULL_DISABLE – 浮空模式 PULL_PU – 上拉模式 PULL_PD – 下拉模式

初始化示例

```
>>> from ql_interrupter import ExIntProcess
>>> from machine import Pin, ExtInt
>>> extint = ExIntProcess(Pin.GPIO14, trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE)
```

开启中断

使能extint对象外部中断，当中断引脚收到上升沿或者下降沿信号时，会调用callback执行 。

> **extint.start()**

- 参数

  无

- 返回值

  使能成功返回整型值0，使能失败返回整型值-1。

关闭中断

> **extint.stop()**

禁用与extint对象关联的中断 。

- 参数

无

- 返回值

使能成功返回整型值0，使能失败返回整型值-1。

订阅

订阅的函数的函数参数说明

参数	类型	说明
topic	string	收到的topic
msg	dict	例{'gpio': 18, 'pressure': 0}, gpio代表使能的gpio引脚号 pressure代表1代表高电平, 0代表低电平

- 用户只需要关注对应的topic就可以
- 示例

```
import sys_bus
from ql_interrupter import ExIntProcess
from machine import Pin, ExtInt

# 初始化中断处理器
extint = ExIntProcess(Pin.GPIO14)
# 开启中断
extint.start()

# 订阅函数
def callback(topic, msg):
    pass

# 订阅topic, 订阅的TOPIC为要处理中断TOPIC值 + _EXINT这种格式
"""
例
    Pin.GPIO14 需订阅 GPIO14_EXINT
    Pin.GPIO14 需订阅 GPIO15_EXINT
    Pin.GPIOXX 需订阅 GPIOXX_EXINT
"""
sys_bus.subscribe("GPIO14_EXINT", callback)
```

GPIO读写处理器

- 概述
  - 优点
    - 为了简便用户的操作, 并支持统一的二次开发, 封装了gpio的模块
    - 支持电平的高低闪烁切换, 提供对外的API(blinker接口), 用户只需要传入对应的从高电压切换到低电压的时间即可
    - 支持电平的限定频率的高低闪烁切换, 提供对外的API(freq_blinker接口), 用户只需要关注频率和切换的时间即可
    - 支持设置高电平

        - 支持设置低电平
        - 简化了客户的操作, 模块化了相关组件
- 功能
    - 对外只提供
        - 设置电平
        - 查看电平
        - 电频高低闪烁

PinInterrupter

- gpio的处理器
- 支持读写gpio
- 支持连续读写

创建对象

> **PinInterrupter(gpio, trige_mode=Pin.OUT, pull_mode=Pin.PULL_DISABLE, mode=0)**

参数	类型	是否必传	
gpio	int	必传项	引脚号 EC100YCN平台引脚对应关系如下（引脚号为外部引脚编号）： GPIO1 – 引脚号22 GPIO2 – 引脚号23 GPIO3 – 引脚号38 GPIO4 – 引脚号53 GPIO5 – 引脚号54 GPIO6 – 引脚号104 GPIO7 – 引脚号105 GPIO8 – 引脚号106 GPIO9 – 引脚号107 GPIO10 – 引脚号178 GPIO11 – 引脚号195 GPIO12 – 引脚号196 GPIO13 – 引脚号197 GPIO14 – 引脚号198 GPIO15 – 引脚号199 GPIO16 – 引脚号203 GPIO17 – 引脚号204 GPIO18 – 引脚号214 GPIO19 – 引脚号215 EC600SCN/EC600NCN平台引脚对应关系如下（引脚号为模块外部引脚编号）： GPIO1 – 引脚号10 GPIO2 – 引脚号11 GPIO3 – 引脚号12 GPIO4 – 引脚号13 GPIO5 – 引脚号14 GPIO6 – 引脚号15 GPIO7 – 引脚号16 GPIO8 – 引脚号39 GPIO9 – 引脚号40 GPIO10 – 引脚号48 GPIO11 – 引脚号58 GPIO12 – 引脚号59 GPIO13 – 引脚号60 GPIO14 – 引脚号61 GPIO15 – 引脚号62 GPIO16 – 引脚号63 GPIO17 – 引脚号69 GPIO18 – 引脚号70 GPIO19 – 引脚号1 GPIO20 – 引脚号3

参数	类型	是否必传	
			GPIO21 – 引脚号49 GPIO22 – 引脚号50 GPIO23 – 引脚号51 GPIO24 – 引脚号52 GPIO25 – 引脚号53 GPIO26 – 引脚号54 GPIO27 – 引脚号55 GPIO28 – 引脚号56 GPIO29 – 引脚号57 EC600UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号61 GPIO2 – 引脚号58 GPIO3 – 引脚号34 GPIO4 – 引脚号60 GPIO5 – 引脚号69 GPIO6 – 引脚号70 GPIO7 – 引脚号123 GPIO8 – 引脚号118 GPIO9 – 引脚号9 GPIO10 – 引脚号1 GPIO11 – 引脚号4 GPIO12 – 引脚号3 GPIO13 – 引脚号2 GPIO14 – 引脚号54 GPIO15 – 引脚号57 GPIO16 – 引脚号56 EC200UCN平台引脚对应关系如下（引脚号为模块外部引脚编号） GPIO1 – 引脚号27 GPIO2 – 引脚号26 GPIO3 – 引脚号24 GPIO4 – 引脚号25 GPIO5 – 引脚号13 GPIO6 – 引脚号135 GPIO7 – 引脚号136 GPIO8 – 引脚号133 GPIO9 – 引脚号3 GPIO10 – 引脚号40 GPIO11 – 引脚号37 GPIO12 – 引脚号38 GPIO13 – 引脚号39 GPIO14 – 引脚号5

参数	类型	是否必传	
			GPIO15 – 引脚号141 GPIO16 – 引脚号142
trige_mode	int	非必传,默认 Pin.OUT	IN – 输入模式，OUT – 输出模式
pull_mode	int	非必传,默认 Pin.PULL_DISABLE	PULL_DISABLE – 浮空模式 PULL_PU – 上拉模式 PULL_PD – 下拉模式
mode	int	非必传,默认是0[低电平]	0 - 设置引脚为低电平, 1- 设置引脚为高电平

初始化示例

```
>>> from ql_interrupter import PinInterrupter
>>> from machine import Pin, ExtInt
>>> pin_inter = PinInterrupter(Pin.GPIO1, rige_mode=Pin.OUT,
pull_mode=Pin.PULL_DISABLE, mode=0)
```

设置电平

> **pin_inter.write(value)**

设置PIN脚电平,设置高低电平前需要保证引脚为输出模式。

- 参数

参数	类型	说明
value	int	0 - 当PIN脚为输出模式时，设置当前PIN脚输出低; 1 - 当PIN脚为输出模式时，设置当前PIN脚输出高

- 返回值

设置成功返回整型值0，设置失败返回整型值-1。

- 示例

```
>>> from machine import Pin
>>> gpio1 = PinInterrupter(Pin.GPIO1, trige_mode=Pin.OUT,
pull_mode=Pin.PULL_DISABLE, mode=0)
>>> gpio1.write(1)
0
>>> gpio1.read()
1
```

获取电平

> **pin_inter.read()**

获取PIN脚电平。

- 参数

无

- 返回值

PIN脚电平，0-低电平，1-高电平。

电平高低闪烁

> **pin_inter.blinker(keep_time)**

执行是会在keep_time时间后完成, 电瓶切换

- 参数

参数	类型	说明
keep_time	int	切换时间, 单位是s级别

- 返回值
  - 无

电平频率高低闪烁

> **pin_inter.freq_blinker(freq,keep_time)**

代表在freq次数执行是会在keep_time时间后完成, 电瓶切换, -1代表一直切换

- 参数

参数	类型	说明
freq	int	频率, 切换的评率, -1代表持续一直切换, 1~N代表切换1~N次
keep_time	int	电平从高电平至低电平的切换时间, 单位是s级别

- 返回值
  - 无

定时处理器

- 概述
  - 优点:
    - 将定时器的操作进行了封装， 提供了统一对外的接口，支持初始化，开启，关闭定期功能

- 功能
  - 拥有定时器的功能,并封装了其对外的接口
  - 对外只提供
    - 开启定时器
    - 关闭定时器

TimerInterrupter

- 定时功能

创建对象

> **TimerInterrupter(keep_time, callback, period=1)**

参数	类型	是否必传	说明
keep_time	int	是	定时时间默认是毫秒
callback	function	是	回调函数, 用于接收定时器的回调
period	int	否	默认是1持续, 如果是0定时器只执行一次

初始化对象

```
>>> from ql_interrupter import TimerInterrupter
>>> from machine import Pin, ExtInt
>>> timer = TimerInterrupter(30000, callback, period=1)
```

开启定时器

开启timer

> **timer.start()**

- 参数

  无

- 返回值
  - 无

关闭定时器

> **timer.stop()**

停止timer

- 参数

无

- 返回值
  - 无

看门狗

- 概述
  - 优点
    - 封装了看门狗的操作，提供定时喂狗的功能
    - 提供了关闭喂狗操作
    - 如果用户关心喂狗的过程可以订阅喂狗的主题，和喂狗成功的主题（通过sys_bus）
- 功能
  - 拥有看门狗定时喂狗功能
  - 支持订阅对应的topic
  - 对外只提供
    - 开启喂狗
    - 关闭喂狗

    - 订阅消息

WatchDog

topic标准

- GPIO主题规则
    - 以GPIO18举例, 如下
    - GPIO18_EXINT
- 喂狗的主题
    - WDT_KICK_TOPIC
- 喂狗后，硬件狗拉GPIO 告知模块喂狗成功主题
    - WDT_KICK_TOPIC

示例

```
def pin_interrupt_callback(topic, message):
    # topic = GPIO18_EXINT, message={'gpio': 18, 'pressure': 0}
    # gpio是对应的gpio号，pressure是电压值
    print("pin_interrupt_callback, topic: {}, message: {}".format(topic,
message))


def ext_interrupt_callback(topic, message):
    # topic = WDT_KICK_TOPIC, message={'gpio': 17, 'pressure': 0}
    # gpio是对应的gpio号，pressure是电压值

    print("ext_interrupt_callback, topic: {}, message: {}".format(topic,
message))


def kick_feed_interrupt_callback(topic, message):
    print("kick_feed_interrupt_callback, topic: {}, message:
{}".format(topic, message))


# 订阅gpio的中断
sys_bus.subscribe("GPIO18_EXINT", pin_interrupt_callback)

# 订阅WDT_KICK_TOPIC订阅喂狗中断的回调
sys_bus.subscribe("WDT_KICK_TOPIC", ext_interrupt_callback)

# 订阅喂狗的回调，每次喂狗都会触发次订阅的回调
sys_bus.subscribe("WDT_KICK_TOPIC_FEED", kick_feed_interrupt_callback)
```

创建对象

> **WatchDog(gpio,mode, keep_time, done_pin=None, trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE)**

参数	类型	是否必传	说明
gpio	int	是	参考pin下面的gpio
mode	int	是	喂狗电平，高电平或者低电平,可选（0、1）
keep_time	int	是	喂狗时间,默认是毫秒级别
done_pin	int	否	喂狗后，硬件狗拉GPIO 告知模块喂狗成功，可以不配置,默认是100,配置需要订阅相关1的topic
trige_mode	int	否	无需传,参考中断参数
pull_mode	int	否	无需传.参考中断参数

初始化对象

```
>>> from ql_interrupter import WatchDog
>>> from machine import Pin, ExtInt
>>> watch_dog = WatchDog(Pin.GPIO1, 0, 10000,
done_pin=100,trige_mode=ExtInt.IRQ_RISING, pull_mode=ExtInt.PULL_DISABLE)
```

开启喂狗

开启喂狗模式

> **watch_dog.start()**

- 参数

  无

- 返回值
  - 无

关闭喂狗

> watch_dog.stop()

关闭喂狗模式

- 参数

无

- 返回值
  - 无

订阅示例

```
from ql_interrupter import WatchDog
from machine import Pin, ExtInt
import sys_bus

# 初始化watch dog
wd = WatchDog(Pin.GPIO15, 1, 10000)
wd.start()


def topic1_cb(topic, msg):
    pass

def topic2_cb(topic, msg):
    pass

# 订阅喂狗后，硬件狗拉GPIO 告知模块喂狗成功
sys_bus.subscribe("WDT_KICK_TOPIC", topic1_cb)

# 订阅喂狗后的回调
sys_bus.subscribe("WDT_KICK_TOPIC_FEED", topic2_cb)
```