lua 扩展库 V1.4

目录

扩展库

1. bit

位操作库

Functions

bit.bit bit.isset bit.isclear bit.set bit.clear bit.bnot bit.band bit.bor bit.bxor

bit.lshift bit.rshift bit.arshift

number = bit.bit(position)

Generate a number with a 1 bit (used for mask generation). Equivalent to 1 << position in C.

Arguments: position - position of the bit that will be set to 1.

Returns: number - a number with only one 1 bit at position (the rest are set to 0).

flag = bit.isset(value, position)

Test if a given bit is set.

Arguments:

- **value** - the value to test.
- **position** - bit position to test.

Returns: **boolean** - true if the bit at the given position is 1, false otherwise.

flag = bit.isclear(value, position)

Test if a given bit is cleared.

Arguments:

- **value** - the value to test.
- **position** - bit position to test.

Returns: **boolean** - true if the bit at the given position is 0, false othewise.

number = bit.set(value, pos1, pos2, ..., posn)

Set bits in a number.

Arguments:

- **value** - the base number.
- **pos1** - position of the first bit to set.
- **pos2** - position of the second bit to set.
- **posn** - position of the nth bit to set.

Returns: **number** - the number with the bit(s) set in the given position(s).

number = bit.clear(value, pos1, pos2, ..., posn)

Clear bits in a number.

Arguments:

- **value** - the base number.
- **pos1** - position of the first bit to clear.
- **pos2** - position of the second bit to clear.
- **posn** - position of thet nth bit to clear.

Returns: **number** - the number with the bit(s) cleared in the given position(s).

number = bit.bnot(value)

Bitwise negation, equivalent to ~*value* in C.

Arguments: **value** - the number to negate.

Returns: **number** - the bitwise negated value of the number.

number = bit.band(val1, val2, ... valn)

Bitwise AND, equivalent to *val1 & val2 & ... & valn* in C.

Arguments:

- **val1** - first AND argument.
- **val2** - second AND argument.
- **valn** - nth AND argument.

Returns: **number** - the bitwise AND of all the arguments.

number = bit.bor(val1, val2, ... valn)

Bitwise OR, equivalent to *val1* | *val2* | *...* | *valn* in C.

Arguments:

- **val1** - first OR argument.
- **val2** - second OR argument.
- **valn** - nth OR argument.

Returns: **number** - the bitwise OR of all the arguments.

number = bit.bxor(val1, val2, ... valn)

Bitwise exclusive OR (XOR), equivalent to *val1* ^ *val2* ^ *...* ^ *valn* in C.

Arguments:

- **val1** - first XOR argument.
- **val2** - second XOR argument.
- **valn** - nth XOR argument.

Returns: **number** - the bitwise exclusive OR of all the arguments.

number = bit.lshift(value, shift)

Left-shift a number, equivalent to *value* < *shift* in C.

Arguments:

- **value** - the value to shift.

- **shift** - positions to shift.

Returns: **number** - the number shifted left

number = bit.rshift(value, shift)

Logical right shift a number, equivalent to *(unsigned)value >> shift* in C.

Arguments:

- **value** - the value to shift.
- **shift** - positions to shift.

Returns: **number** - the number shifted right (logically).

number = bit.arshift(value, shift)

Arithmetic right shift a number equivalent to *value >> shift* in C.

Arguments:

- **value** - the value to shift.
- **shift** - positions to shift.

Returns: **number** - the number shifted right (arithmetically).

2. cpu

包含 cpu 中断 id 定义

Data structures, constants and types

```
cpu.INT_GPIO_POSEDGE
cpu.INT_GPIO_NEGEDGE
```

中断 id:
INT_GPIO_POSEDGE GPIO 上升沿中断
INT_GPIO_NEGEDGE GPIO 下降沿中断

Functions

Null

3. i2c

i2c 操作接口

Functions

i2c.setup i2c.write i2c.read

speed = i2c.setup(id, speed, slave)

打开 I2C 接口

Arguments:

- **id** - i2c 接口 id，目前支持 i2c id=1 即模块的 I2C2
- **speed** - **i2c.FAST** (400KHz), **i2c.SLOW** (100KHz)
- **slave** - i2c 外设地址 0x00-0x7f

Returns: 可以根据返回的频率值判断是否成功打开 i2c

wrote = i2c.write(id, reg, data)

往指定的寄存器地址 **reg** 传输数据

Arguments:

- **id** - i2c 接口 id
- **reg** - 写入 i2c 从设备的寄存器起始地址
- **data** - number / string / table，自动根据参数类型写数据，num 只写 1 个字节，string/table 自动根据输入的长度传输相应字节

Returns: 传输成功的字节数

data = i2c.read(id, reg, num)

读取指定寄存器地址 **reg** 的数据内容

Arguments:

- **id** - i2c 接口 id
- **reg** - 读取 i2c 从设备的寄存器起始地址
- **num** - 读取数据字节数

Returns: 返回读取的数据，二进制数据会包含非可见字符，请使用 string.byte 打印数据流

speed = i2c.close(id)

关闭 I2C 接口

Arguments:

- **id** - i2c 接口 id，目前支持 i2c id=1 即模块的 I2C2

Returns: nothing

4. pack

pack 库支持将一系列数据按照格式字符转化为 lua 字符串或者将 lua 字符串按照格式字符转化成一系列值
(pack 和 unpack) 使用一个或多个格式字符来描述如何转化数据. 格式字符串格式如下:

```
[endianness]<format specifier>[count]
```

where:

- **endianness** is an optional endian flags that specifies how the numbers that are to be packed/unpacked are stored in memory. It can be either:
    1. **'<'** for little endian.
    2. **'>'** for big endian.
    3. **'='** for native endian (the platform's endian order, default).
- **format specifier** describes what kind of variable will be packed/unpacked. **The format specifier is case-sensitive**. The possible values of this parameter are summarized in the table below:

Format specifier	Corresponding variable type
'z'	zero-terminated string
'p'	string preceded by length byte
'P'	string preceded by length word
'a'	string preceded by length size_t
'A'	string
'f'	float
'd'	double
'n'	Lua number
'c'	char
'b'	byte = unsigned char
'h'	short
'H'	unsigned short
'i'	int
'I'	unsigned int
'l'	long
'L'	unsigned long

- **count** is an optional counter for the **format specifier**. For example, **i5** instructs the code to pack/unpack 5 integer variables, as opposed to **i** that specifies a single integer variable.

Functions

pack.pack pack.unpack

packed = pack.pack(format, val1, val2, ..., valn)

Packs variables in a string.

Arguments:

- **format** - format specifier (as described here).
- **val1** - first variable to pack.
- **val2** - second variable to pack.
- **valn** - nth variable to pack.

Returns: **packed** - a string containing the packed representation of all variables according to the format.

nextpos, val1, val2, ..., valn = pack.unpack(string, format, [init])

Unpacks a string

Arguments:

- **string** - the string to unpack.
- **format** - format specifier (as described here).
- **init** - **(optional)** marks where in **string** the unpacking should start (1 if not specified).

Returns:

- **nextpos** - the position in the string after unpacking.
- **val1** - the first unpacked value.
- **val2** - the second unpacked value.
- **valn** - the nth unpacked value.

5. pio

pio.P0_0 - **pio.P0_31** 表示 GPIO_0 - GPIO_31
pio.P1_0 - **pio.P1_9** 表示 GPO_0 - GPO_9
GPIO 工作方式有三种,通过 pio.pin.setdir 设置: **pio.INPUT pio.OUTPUT pio.INT**
pio.INT 中断方式来中断时通过 **rtos.MSG_INT** 消息通知,可以根据消息数据域来判断中断 id 与中断的 pin 脚:
id - 中断 id
resnum - 中断 pin

Functions

pio.pin.setdir pio.pin.setval pio.pin.getval pio.pin.sethigh pio.pin.setlow pio.decode

pio.pin.setdir(direction, pin1, pin2, ..., pinn)

Set pin(s) direction

Arguments:

- **direction** - the pin direction, can be either **pio.INPUT** or **pio.OUTPUT** or **pio.INT**
- **pin1** - the first pin
- **pin2 (optional)** - the second pin
- **pinn (optional)** - the *n*-th pin

Returns: nothing.

pio.pin.setval(value, pin1, pin2, ..., pinn)

Set pin(s) value

Arguments:

- **value** - pin value, can be either 0 or 1
- **pin1** - the first pin
- **pin2 (optional)** - the second pin
- **pinn (optional)** - the *n*-th pin

Returns: nothing.

val1, val2, ..., valn = pio.pin.getval(pin1, pin2, ..., pinn)

Get value of pin(s)

Arguments:

- **pin1** - the first pin
- **pin2 (optional)** - the second pin
- **pinn (optional)** - the *n*-th pin

Returns: The value(s) of the pin(s), either 0 or 1

pio.pin.sethigh(pin1, pin2, ..., pinn)

Set pin(s) to 1 (high)

Arguments:

- **pin1** - the first pin
- **pin2 (optional)** - the second pin
- **pinn (optional)** - the *n*-th pin

Returns: nothing.

pio.pin.setlow(pin1, pin2, ..., pinn)

Set pin(s) to 0 (low)

Arguments:

- **pin1** - the first pin
- **pin2 (optional)** - the second pin
- **pinn (optional)** - the *n*-th pin

Returns: nothing.

port, pin = pio.decode(resnum)

Convert a PIO resource number to the corresponding port and pin. This is most commonly used in GPIO edge interrupt routines to convert the Lua interrupt routine's argument to the port and pin that caused the interrupt but it can also be used on the values returned by the pin names pio.PA_0, pio.P2_15 and so on.

Arguments: **resnum** - the resource number of the pin

Returns:

- **port** - the index of the port, starting from 0 (so port A is 0, port B is 1 and so on)
- **pin** - the pin number, usually from 0 to 31

6. pmd

电源管理接口:ldo 控制,省电管理

Data structures, constants and types

```
pmd.LDO_KEYPAD
pmd.LDO_LCD--控制 LED0-LED4
pmd.KP_LEDR
pmd.KP_LEDG
pmd.KP_LEDB
pmd.LDO_VIB
pmd.LDO_VLCD--控制 POWER_VLCD
pmd.LDO_VASW -- V_ASW
pmd.LDO_VMMC -- V_MMC
```

ldo id 值

Functions

pmd.init pmd.ldoset pmd.sleep

result = pmd.init(param)

设置电源管理参数

电池充电控制,3 阶段电流充电:

一阶段：电压低于 battlevelFirst 充电电流为 currentFirst

二阶段：电压高于 battlevelFirst 低于 battlevelSecond 充电电流为 currentSecond

三阶段：电压高于 battlevelSecond 至充满 4.25v 充电电流为 currentThird

Arguments:

- **param** - 参数表,电流有效值:50,100,150,200,300,400,500,600,700,800 电压值以 mV 为单位
- **param.currentFirst** - 电池电压小于一阶段电压值时的充电电流
- **param.battlevelFirst** - 一阶段电压值节点
- **param.currentSecond** - 电池电压大于一阶段电压值小于二阶段电压值时的充电电流
- **param.battlevelSecond** - 二阶段电压值节点
- **param.currentThird** - 电池电压大于二阶段电压值时的充电电流

Returns: **result** - 1:成功 0:失败

pmd.ldoset(level, id1, [id2], ..., [idn])

ldo 控制

Arguments:

- **level** - ldo 亮度 0 - 7 级 0 级关闭
- **id1** - 要设置的第一个 ldo
- **id2 (optional)** - 要设置的第 2 个 ldo
- **idn (optional)** - 要设置的第 n 个 ldo

Returns: nothing.

pmd.sleep(value)

省电控制

Arguments: **value** - 1 - 进入睡眠，0 - 退出睡眠

Returns: nothing.

7. rtos

嵌入式系统接口:接收消息,软件定时器

Functions

rtos.init_module rtos.receive rtos.timer_start rtos.timer_stop rtos.poweron_reason rtos.poweron rtos.poweroff rtos.restart rtos.tick rtos.sleep Percentage=rtos.get_env_usage

rtos.init_module(module, [param1], ..., [paramn])

初始化模块

Arguments:

module - 按键功能 **rtos.MOD_KEYPAD**,按键功能参数表:

rtos.init_module(rtos.MOD_KEYPAD, type, [inmask, outmask])

type: 键盘类型 目前只有矩阵键盘(type = 0)

inmask outmask: 矩阵键盘有效行列标记 比如某矩阵键盘由 keyin0 keyin3 keyout1 keyout4 组成，inmask=1<0|1<3=0x09 outmask = 1<1|1<4=0x12

Returns: nothing.

msg,msgpara = rtos.receive(timeout)

接收消息

Arguments: **timeout** - timeout 超时返回以毫秒为单位，可以用#rtos.INF_TIMEOUT#表示阻塞等待消息

Returns:

如果 msg 为 table 类型，msg 根据不同的消息 msg.id 会有不同的数据：

如果 msg 为 number 类型，msg 根据不同的消息 msg 会有不同的数据

1.**rtos.MSG_TIMER** 定时器超时消息

msg.timer_id 或者 msgpara 为超时的定时器 id

2.**rtos.MSG_UART_RXDATA** 串口 ATC 数据提醒

msg.uart_id 或者 msgpara 为收到的数据的串口 id 或者 atc,收到该消息后可以通过 uart.read 接口读取数据

3.**rtos.MSG_KEYPAD** 键盘消息,必须初始化按键(#rtos.init_module#)后才会有键盘消息

msg.pressed 按键按下/弹起

msg.key_matrix_row 按键所在行值

msg.key_matrix_col 按键所在列值

4.**rtos.WAIT_MSG_TIMEOUT** 等待消息超时

5.**rtos.MSG_INT** 中断消息

msg.int_id 中断 id

msg.int_resnum 中断 pin 脚编号

6.**rtos.MSG_PMD** 电源管理消息

msg.present 电池在位状态

msg.level 百分比 0-100

msg.voltage 电池电压

msg.charger 充电器在位状态

msg.state 充电状态:0-不在充电 1-充电中 2-充电停止

rtos.timer_start(timer_id, timeout)

启动定时器

Arguments:

- **timer_id** - 定时器 id,可以是任意整数,定时器到时 msg.timer_id 值为启动时定时器
- **timeout** - 定时器延时时间以毫秒为单位

Returns: nothing.

rtos.timer_stop(timer_id)

停止定时器

Arguments: **timer_id** - 输入与启动定时器时定义的 id 即可停止定时器

Returns: nothing.

reason=rtos.poweronreason()

读取开机原因值

Arguments: nothing

Returns: 开机原因值，取值范围如下：

rtos. POWERON_KEY：按键开机

rtos. POWERON_CHARGER：充电开机

rtos. POWERON_ALARM：闹钟开机

rtos. POWERON_RESTART：软件重启开机

rtos. POWERON_EXCEPTION：异常开机

rtos. POWERON_HOST：HOST 工具控制重启开机

rtos. POWERON_WATCHDOG：软件看门狗开机

rtos.poweron(flag)

是否启动 GSM 开机

Arguments:

- **flag**- 0 表示不启动系统，1 表示启动系统

Returns: nothing.

rtos.poweroff()

软件关机

Arguments: nothing

Returns: nothing.

rtos.restart()

软件重启

Arguments: nothing

Returns: nothing.

ticks=rtos.tick()

获取系统开机运行时间总计数

Arguments: nothing

Returns: ticks，时间计数，每 tick 时长：Air200 或 Air202 是 1/16384 秒，Air810 是 4.615 毫秒。

rtos.sleep(millisecond)

堵塞等待 V103 以上版本支持

Arguments:

- **millisecond** - 堵塞等待时间 以毫秒为单位

Returns: nothing.

percentage=rtos.get_env_usage()

获取 lua 任务消息队列的使用百分比

Arguments: nothing

Returns: percentage，百分比，例如使用了 80%，则 percentage 为 80

8. uart

uart 与虚拟 AT 交互接口

Data structures, constants and types

`uart.ATC`

uart.ATC 通道可以发送通过虚拟 AT 通道收发 AT 命令

Functions

uart.setup uart.write uart.getchar uart.read uart.close

baud = uart.setup(id, baud, databits, parity, stopbits, [msgmode])

打开 uart 接口

Arguments:

- **id** - the ID of the serial port
- **baud** - serial baud rate
- **databits** - number of data bits
- **parity** - parity type, can be either **uart.PAR_EVEN**, **uart.PAR_ODD** or **uart.PAR_NONE**

  stopbits - the number of stop bits, can be either **uart.STOP_1** (for 1 stop bit), **uart.STOP_1_5** (for 1.5 stop bits) or **uart.STOP_2** (for 2 stop bits)
- **msgmode** - 包含以下设置值:
  - 0 或者默认 - 消息通知
  - 1 - 无消息上报需要用户主动轮询

Returns: The actual baud rate set on the serial port. Depending on the hardware, this might have a different value than the **baud** parameter

uart.write(id, data1, [data2], ..., [datan])

Write one or more strings or 8-bit integers (raw data) to the serial port. If writing raw data, its value (represented by an integer) must be between 0 and 255.

Arguments:

- **id** - the ID of the serial port.
- **data1** - the first string/8-bit integer to write.
- **data2 (optional)** - the second string/8-bit integer to write.
- **datan (optional)** - the *n*-th string/8-bit integer to write.

Returns: nothing.

str = uart.getchar(id, [timeout])

Read a single character from the serial port

Arguments:

- **id** - the ID of the serial port
- **timeout (optional)** - timeout of the operation, can be either **uart.NO_TIMEOUT** or 0 for non-blocking operation, **uart.INF_TIMEOUT** for blocking operation, or a positive number that specifies the timeout in microseconds. The default value of this argument is **uart.INF_TIMEOUT**.

Returns: The character read from the serial port as a string, or the empty string it timeout occured while waiting for the character.

str = uart.read(id, format, [timeout])

Reads one or more characters from the serial port according to a format specifier

Arguments:

- **id** - the ID of the serial port
- **format** - format of data to read. This can be either:
    - **'*l'** - read until an end of line character (a **\n**) is found (the **\n** is not returned) or a timeout occurs.
    - **'*n'** - read an integer. The integer can optionally have a sign. Reading continues until the first non-digit character is detected or a timeout occurs. This is the only case in which **read** returns a number instead of an integer.
    - **'*s'** - read until a spacing character (like a space or a TAB) is found (the spacing character is not returned) or a timeout occurs.

  - **a positive number** - read at most this many characters before returning (reading can stop earlier if a timeout occurs).

- **timeout (optional)** - timeout of the operation, can be either **uart.NO_TIMEOUT** or 0 for non-blocking operation, **uart.INF_TIMEOUT** for blocking operation, or a positive number that specifies the timeout in microseconds. The default value of this argument is **uart.INF_TIMEOUT**.

Returns: The data read from the serial port as a string (or as a number if **format** is **'*n'**). If a timeout occures, only the data read before the timeout is returned. If the function times out while trying to read the first character, the empty string is returned

uart.close(id)

关闭 uart 接口

Arguments: **id** - the ID of the serial port.

Returns: nothing.

9. adc

adc 操作接口

Functions

adc.open adc.read

result = adc.open(id)

Open an adc channel specified by id.

Arguments: id – adc channel id. 有效值 0~7

Returns: result – 1 if the adc channel is opened successfully, 0 otherwise.

adcValue,voltValue = adc.read(id)

Read raw measured value and coverted voltage value in mv unit for the specified adc channel.

Arguments: id – adc channel id.

Returns:

- **adcValue** - raw measured value. 无效值为 0xFFFF

  voltValue - coverted voltage value in mv unit. 无效值为 0xFFFF

10. iconv

Functions

iconv.open cd:iconv

cd = iconv.open(to, from)

返回指定的字符编码转换的处理描述符，iconv 转换操作将根据这个描述符展开

Arguments:

- **to** - 目标字符编码
- **from** - 源字符编码

Returns: **cd** - 返回的字符编码处理描述符,失败则为 nil

to_text, err = cd:iconv(from_text)

cd 为 iconv.open 操作打开的描述符,根据该描述符指定的编码集进行编码转换

Arguments:

- **from_text** - 源编码字符串

Returns:

- **to_text** - 转换为目标编码的字符串
- **err** - 错误码(整数),无错误则为 nil

11. audiocore

音频操作接口

Functions

audiocore.play audiocore.stop

ret = audiocore.play(filename)

播放音乐

Arguments:

- **filename** - 音频文件名

Returns: 播放成功或失败 true or false

audiocore.stop()

stop audio play if exists.

Arguments: nothing

Returns: nothing

result = audiocore.setchannel(channel)

set audio speaker channel.

Arguments:

- **channel** –audio speaker channel. The possible values of this parameter are summarized below:

  audiocore.HANDSET 、 audiocore.EARPIECE 、 audiocore.LOUDSPEAKER 、 audiocore.BLUETOOTH 、 audiocore.FM 、 audiocore.FM_LP 、 audiocore.TV 、 audiocore.AUX_HANSET 、 audiocore.AUX_LOUDSPEAKER 、 audiocore.AUX_EARPIECE 、 audiocore.DUMMY_HANDSET 、 audiocore.DUMMY_AUX_HANDSET 、 audiocore.DUMMY_LOUDSPEAKER 、 audiocore.DUMMY_AUX_LOUDSPEAKER

Returns:

- **result** – 1 if succeessfully, 0 otherwise.

result = audiocore.setvol(vol)

set audio speaker volume.

Arguments:

- **vol** – audio speaker volume. The possible values of this parameter are summarized below:

  audiocore.VOL0、audiocore.VOL1、audiocore.VOL2、audiocore.VOL3、audiocore.VOL4、audiocore.VOL5、audiocore.VOL6、audiocore.VOL7

Returns:

- **result** – 1 if succeessfully, 0 otherwise.

result = audiocore.setmicvol(vol)

set microphone volume.

Arguments:

- **vol** – microphone volume. The possible values of this parameter are summarized below:

- audiocore.MIC_VOL0、audiocore. MIC_VOL1、audiocore. MIC_VOL2、audiocore. MIC_VOL3、audiocore. MIC_VOL4、audiocore. MIC_VOL5、audiocore. MIC_VOL6、audiocore. MIC_VOL7 、 audiocore.MIC_VOL8 、 audiocore.MIC_VOL9 、 audiocore.MIC_VOL10 、 audiocore.MIC_VOL11 、 audiocore.MIC_VOL12 、 audiocore.MIC_VOL13、audiocore.MIC_VOL14、audiocore.MIC_VOL15

Returns:

- **result** – 1 if succeessfully, 0 otherwise.

result = audiocore.setloopback(status,channel)

set loopback test specified by a channel.

Arguments:

- **status** – 1 open loopback test, 0 close loopback test

- **channel** – audio channel. The possible values of this parameter are summarized below:

  audiocore. LOOPBACK_HANDSET 、 audiocore. LOOPBACK_EARPIECE 、 audiocore. LOOPBACK_LOUDSPEAKER 、 audiocore. LOOPBACK_AUX_HANDSET 、 audiocore. LOOPBACK_AUX_LOUDSPEAKER

Returns:

result – 1 if succeessfully, 0 otherwise.

12. apn

apn 读取库

apn,user,password = apn.get_default_apn(mcc,mnc)

Get the default apn by mcc and mnc.

Arguments:

- **mcc** – mobile country code.
- **mnc** – mobile network code.

Returns:

- **apn** – apn name.
- **user** – user name.
- **password** - password.