

# MCUSH 基础命令使用说明

说明：此文档适用于 MCUSH 平台提供的基础、通用、开放型命令。

作者：彭树林

更新日期：2019-9-4

## 1、SCPI 接口类

### \*idn?

别名：无

功能：查询设备信息（型号，版本号和序列号等）

语法：无参数

示例	注释
<pre>=&gt;*idn? ShellLab,1.0 3C002C000547373435343731 =&gt;</pre>	查询基本信息 第一行：设备型号、固件版本号，逗号隔开 第二行：设备硬件序列号

注：强制所有 MCUSH 衍生产品支持此命令，否则 Python 模块初始化设备对象时会调用失败。

### \*rst

别名：无

功能：设备状态复位

语法：无参数

示例	注释
<pre>=&gt;*rst =&gt;</pre>	设备复位

注：建议所有 MCUSH 衍生产品支持此命令，用于系统不重启的情况下恢复成初始状态。

## 2、基本查询类

### help

别名：?

功能：打印使用帮助

语法: `help [-a] [-c <command>]`  
 options:  
`-a/--all` show all  
`-c/--check` check if command exists

示例	注释
<pre>=&gt;help help/? print command list   help [-a] *idn? print device info   *idn? ... ... =&gt;</pre>	打印所有（不包含隐藏）命令 每个命令占两行，包括简要说明和简要用法。
<pre>=&gt;help -c log 1 =&gt;</pre>	检查系统是否支持 <code>log</code> 命令 返回 <code>0</code> -不支持 <code>1</code> -支持 常用于自动测试脚本检查运行环境

注：仅列出了相应命令最常用的参数语法，完整的参数语法通过命令加`--help`参数打印。

## uptime

别名: 无  
 功能: 打印上电时间  
 语法: 无参数

示例	注释
<pre>=&gt;uptime 0:17:04.248 =&gt;</pre>	返回: 时/分/秒/毫秒

- 注:
- 通过系统节拍定时器获得，最小时间颗粒取决于系统节拍的定义。
  - 对定时器的溢出不做检查。

## 3、系统查询类

### sys

别名: 无  
 功能: FreeRTOS 状态查询  
 语法: `sys <type>`

options:  
`type` (t)ask | (q)ueue | (k)ern | heap | stack | (i)dle | v(f)s

示例	注释
<pre>=&gt;sys t  2  mcushT X 0x20004EA8 3/3 0x20000EA0 0x2000482C (free 14048)  1  vcp/txT R 0x20000DF0 3/3 0x20000CC0 0x20000D4C (free 48)  5   idleT R 0x200061A8 0/0 0x20006010 0x20006134 (free 304)  3  blinkT B 0x20004FD0 3/3 0x20004F00 0x20004F44 (free 56)  4   logT B 0x20005FB8 1/1 0x200053B0 0x20005AFC (free 1864)  6  tmrSvrT B 0x200064B0 6/6 0x20006318 0x2000642C (free 232) =&gt;</pre>	<p>查询任务信息 每行任务包含： 任务编号、任务名、 运行状态（<b>X</b> 运行、<b>R</b> 就绪、<b>B</b> 阻塞、<b>S</b> 休眠、<b>D</b> 删除）、 TCB 地址，优先级/基准优先级、任务栈及栈顶、栈剩余字节（按水印检查从未使用过的部分）。</p>
<pre>=&gt;sys q  logQ 0x20005080 20 16 0 0x200050D0 - 0x20005210 (0x00000140) logMQ 0x20005218 20 16 0 0x20005268 - 0x200053A8 (0x00000140)  TmrQ 0x20006200 16 12 0 0x20006250 - 0x20006310 (0x000000C0) =&gt;</pre>	<p>查询队列信息 每行队列包含： 队列名、控制块地址、队列总长度、消息长度、消息长度、队列首/尾地址及消息总字节数。</p>
<pre>=&gt;sys k CurrentNumberOfTasks: 6 TopReadyPriority: 9 PendedTicks: 0 NumOfOverflows: 0 CurrentTCB: 0x20004EA8 mcushT ReadyTaskLists[0]: 0x100002D8 idleT ReadyTaskLists[1]: 0x100002EC ReadyTaskLists[2]: 0x10000300 ReadyTaskLists[3]: 0x10000314 mcushT ReadyTaskLists[4]: 0x10000328 ReadyTaskLists[5]: 0x1000033C ReadyTaskLists[6]: 0x10000350 DelayedTaskList1: 0x10000364 DelayedTaskList2: 0x10000378 DelayedTaskList: 0x10000364 OverflowDelayedTList: 0x10000378 PendingReadyList: 0x10000394 SuspendedTaskList: 0x100003C0 tmrSvrT,logT =&gt;</pre>	<p>查询系统内核信息</p>
<pre>=&gt;sys idle 100 % 99 % 99 % =&gt;</pre>	<p>检查系统空闲比例 比例越大系统越空闲</p>
<pre>=&gt;sys f mount: 2 umount: 0</pre>	<p>检查虚拟文件系统接口调用统计</p>

```
open: 10307 / 6
close: 10301 / 0
read: 2606 / 0
write: 8118 / 0
flush: 7670 / 0
=>
```

## 4、内存调试类

### dump

别名: x

功能: 打印内存变量

语法: dump [-b <address>] [-l <length>] [-w <width>] [-c] [-f] [-C] [-i] [-I]  
options:

- b/--address base address
- l/--length default 16
- w/--width 1(default)|2|4
- c/--compact compact output
- f/--float float output (width=4)
- C/--ascii ascii output (width=1)
- i/--int signed integer output
- I/--uint unsigned integer output

示例	
<p>打印 0x20000000 地址 64 字节内容，并解析成 ASCII 内容</p> <pre>=&gt;x -b 0x20000000 -l64 -C 20000000: 00 00 00 00 EC 02 00 20 54 03 00 20 BC 03 00 20  ..... T.. ...   20000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....  20000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....  20000030: 00 00 00 00 74 10 05 08 00 00 00 00 00 00 00 00  ....t.....  =&gt;</pre>	
<p>打印 0x20000000 地址 64 字节内容，按 2 字节一组解析成 16 位有符号整数</p> <pre>=&gt;x -b 0x20000000 -l64 -w2 -I 20000000: 0000 0000 02EC 2000 0354 2000 03BC 2000  0 0 748 8192 852 8192 956 8192  20000010: 0000 0000 0000 0000 0000 0000 0000 0000  0 0 0 0 0 0 0 0  20000020: 0000 0000 0000 0000 0000 0000 0000 0000  0 0 0 0 0 0 0 0  20000030: 0000 0000 1074 0805 0000 0000 0000 0000  0 0 4212 2053 0 0 0 0  =&gt;</pre>	
<p>打印 0x20000000 地址 64 字节内容，按 4 字节一组解析成 32 位浮点数</p> <pre>=&gt;x -b 0x20000000 -l64 -w4 -f 20000000: 00000000 200002EC 20000354 200003BC   0.000000e+00 1.084298e-19 1.084312e-19 1.084325e-19  20000010: 00000000 00000000 00000000 00000000   0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  =&gt;</pre>	

```

20000020: 00000000 00000000 00000000 00000000 | 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00|
20000030: 00000000 08051074 00000000 00000000 | 0.000000e+00 4.004256e-34 0.000000e+00 0.000000e+00|
=>

用精简模式打印 0x20000000 地址 64 字节内容，常用于自动测试脚本提高传输速率
=>x -b 0x20000000 -l64 -c
00000000EC02002054030020BC030020
00000000000000000000000000000000
00000000000000000000000000000000
00000000741005080000000000000000
=>

```

## write

别名: w

功能: 写入内存变量

语法: write [-b <address>] [-w <bus width>] <data>

options:

-b/--address base address

-w/--width 1(default)|2|4

data data to be written

### 示例

```

将 0x20000000 地址按字节写入数据: 0x00, 0x01, 0x02, 0x03
=>w -b 0x20000000 0 1 2 3
=>x -b 0x20000000
20000000: 00 01 02 03 EC 02 00 20 54 03 00 20 BC 03 00 20
=>

将 0x20000000 地址按 16 位整数写入数据: 0x0000, 0x0001, 0x0002, 0x0003
=>w -b 0x20000000 -w2 0 1 2 3
=>x -b 0x20000000
20000000: 00 00 01 00 02 00 03 00 54 03 00 20 BC 03 00 20
=>

将 0x20000000 地址按 32 位整数写入数据: 0x00000000, 0x00000001, 0x00000002, 0x00000003
=>w -b 0x20000000 -w4 0 1 2 3
=>x -b 0x20000000
20000000: 00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00
=>

```

## mfill

别名: 无

功能: 指定模式填充内存变量

语法: mfill [-b <address>] [-l <length>] [-w <bus width>] [-p <pattern>] [-t]  
 options:  
 -b/--address base address  
 -l/--length memory length  
 -w/--width 1(default)|2|4  
 -p/--pattern data to be written  
 -t/--test test\_mode

示例
将 0x20000000 地址按字节填充数据 0x5A, 总长度 32 =>mfill -b 0x20000000 -l32 -w1 -p0x5A =>x -b 0x20000000 -l32 20000000: 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 20000010: 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A =>
将 0x20000000 地址按 16 位整数填充数据 0x00A5, 总长度 16 =>mfill -b 0x20000000 -l16 -w2 -p0xA5 =>x -b 0x20000000 -l32 20000000: A5 00 A5 00 A5 00 A5 00 A5 00 A5 00 A5 00 A5 00 20000010: 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A =>

## mapi

别名: 无

功能: 控制内存 API

语法: mapi [-t] [-i] [-m] [-r] [-f] [-b <address>] [-l <length>]  
 options:  
 -t/--test test heap memory  
 -i/--info print mallinfo  
 -m/--malloc allocate new memory  
 -r/--realloc re-allocate memory  
 -f/--free free memory  
 -b/--address base address  
 -l/--length memory length

示例	注释
=>mapi -m -l 100 0x20006728 =>	申请 100 字节内存 返回申请到的内存地址 返回 0 为失败
=>mapi -f -b 0x20006728 =>	释放刚申请的 100 字节。
=>mapi -t [1] 0x20006728 60074 [2] 0x200151D8 30037	内存申请测试 从大内存块开始申请, 失败则容量减半, 反复尝试; 打印所有申请到的内存和统计总和。

<pre>[3] 0x2001C738 3754 [4] 0x2001D5E8 1877 [5] 0x2001DD48 469 [6] 0x2001DF28 117 [7] 0x2001DFA8 58 [8] 0x2001DFE8 7 total: 96393 =&gt;</pre>	<p>最后全部释放还原。 注：执行该操作可能会对当前运行的任务的内存申请请求造成干扰。</p>
--	---

## mkbuf

别名：无

功能：创建数据内存（申请足够内存并填入数据）

语法：mkbuf [-f]

options:

-f/--float float mode

示例	注释
<pre>=&gt;mkbuf &gt;0 1 2 3 4 5 6 7 8 9 &gt; address: 0x20006728 length: 10 =&gt;x -b 0x20006728 20006728: 00 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 =&gt;mapi -f -b 0x20006728 =&gt;</pre>	<p>申请 <b>16</b> 位整数缓存，填入 <b>10</b> 个整数（从 <b>0~9</b>）。</p> <p>空行结束输入</p> <p>返回创建结果（地址和长度）</p> <p>查看内容</p> <p>释放缓存</p>
<pre>=&gt;mkbuf -f &gt;0.1 1.2 3.14159 2.71828 &gt; address: 0x20006930 length: 4 =&gt;x -b 0x20006930 -w4 -f 20006930: 3DCCCCCD 3F99999A 40490FD0 402DF84D   1.000000e-01 1.200000e+00 3.141590e+00 2.718280e+00  =&gt;mapi -f -b 20006930 =&gt;</pre>	<p>申请 <b>32</b> 位浮点数缓存，填入 <b>4</b> 个浮点数</p> <p>空行结束输入</p> <p>返回创建结果（地址和长度）</p> <p>查看内容</p> <p>释放缓存</p>

## 5、硬件控制类

### reboot

别名：无

功能：系统重启

语法： `reboot [-c] [-r]`

options:

`-c/--count`          print counter

`-r/--reset`          reset counter

示例	注释
<pre>=&gt;reboot =&gt;</pre>	系统重启
<pre>=&gt;reboot -c 10 =&gt;</pre>	打印重启次数
<pre>=&gt;reboot -r =&gt;reboot -c 0 =&gt;</pre>	重置重启次数为零

注：重启次数功能需 BSP 支持（通常需要有不受复位影响的备用 SRAM）。

### wdg

别名：无

功能：控制硬件看门狗

语法： `wdg <command>`

options:

`command`                  enable|disable|clear|reset

示例	注释
<pre>=&gt;wdg enable =&gt;</pre>	检查硬件看门狗是否工作 常用于判断当前固件是否为调试版或正式版

注：需 BSP 支持。

### led

别名：无

功能：控制 BSP 注册的 LED 灯

语法： `led [-s] [-t] [-c] [-i <led_index>] [-n] [-T]`



```
options:
-s/--set      on
-t/--toggle   invert
-c/--clr      off
-i/--index    index from 0
-n/--number   query
-T/--test     blink all
```

示例	注释
=>led -i0 -s =>	点亮第 0 编号的 LED
=>led -i0 -c =>	关闭第 0 编号的 LED
=>led -i0 -t =>	翻转第 0 编号的 LED
=>led -T =>	测试模式，闪烁所有 LED，按 Ctrl-C 中止 常用于多个设备时检查确认端口号
=>led -n 4 =>	查询注册的 LED 数量 LED 0~3

注：建议所有 MCUSH 衍生产品支持此命令，且至少支持 1 个 LED。

## gpio

别名：无

功能：控制 BSP 注册的 GPIO 端口

语法： gpio [--loop[=<loop\_delay\_ms>]] [-p <port\_bit\_name>]  
 [--input[=<input\_mode>]] [--output[=<output\_mode>]]  
 [--set[=<set\_high\_val>]] [--clr[=<set\_low\_val>]]  
 [--toggle[=<toggle\_val>]] [-n] [-U] [-D]

```
options:
-l/--loop      default 1000ms
-p/--port      port[.bit] name, eg 0[.0]
-i/--input     set input mode mask
-o/--output    set output mode mask
-s/--set       set output high mask
-c/--clr       set output low mask
-t/--toggle    toggle output mask
-n/--number    query
-U/--pullup    with pullup resister
-D/--pulldown with pulldown resister
```

示例	注释
=>gpio -p0.0 1	检查端口 0.0 状态 返回 0-低, 1-高

=>	
=>gpio -p0 0x0000B7CF =>	检查端口 0 状态 返回 32 比特位
=>gpio -p0.0 -o =>	设置端口 0.0 为输出模式
=>gpio -p0.0 -s =>	设置端口 0.0 为高电平
=>gpio -p0.0 -c =>	设置端口 0.0 为低电平
=>gpio -p0.0 -t -l =>	循环翻转端口 0.0, 1 秒节拍, 按 Ctrl-C 中止
=>gpio -p0.0 -t -l 100 =>	循环翻转端口 0.0, 0.1 秒节拍
=>gpio -p2 -o 0xFFFF =>gpio -p2 -s 0xFFFF =>	设置端口 2 低 16 位输出高电平
=>gpio -n 9 =>	查询注册的 GPIO 数量 GPIO 0~8

注:

- 建议所有 MCUSH 衍生产品支持此命令, 支持所有 GPIO。
- STM32 平台的端口 0/1/2...对应 GPIO A/B/C...。
- 上/下拉电阻功能需要 BSP 支持。

## rtc

别名: 无

功能: 控制实时时钟

语法: `rtc [-s] <setting>`

options:

`-s/--set`            `set rtc`

setting            `format: YYYY-MM-DD HH:MM:SS`

示例	注释
=>rtc 2017-12-5 14:19:27 =>	查看 RTC
=>rtc -s 2017-12-5 14:20:00 =>rtc 2017-12-5 14:20:00 =>	修改 RTC

注: 需 BSP 支持。

## beep

别名: b

功能: 控制蜂鸣器

语法: `beep [-f <frequency>] <ms>`

options:

`-f/--frequency` 20~10000 (default 4000)hz

`ms` 1~10000 (default 50)ms

示例	注释
<code>=&gt;b</code>	响一声 (4k 频率, 50ms)
<code>=&gt;</code>	

注: 需 BSP 支持。

## spi

别名: 无

功能: IO 口模拟 SPI 控制

语法: `spi [-w <bits>] [--delay=<delay_us>] [--sdi=<sdi_pin>] [--sdo=<sdo_pin>] [--sck=<sck_pin>] [--cs=<cs_pin>] [-I] [-D] [-r] [--cpol] [--cpha] [--lsb] <value>`

options:

`-w/--width` default 8

`--delay` default 5

`--sdi` default 0.0

`--sdo` default 0.1

`--sck` default 0.2

`--cs` default 0.3

`-I/--init` init pins

`-D/--deinit` deinit pins

`-r/--read` print readout

`--cpol` clk polarity

`--cpha` clk phase

`--lsb` lsb first

`value` data

示例	注释
<code>=&gt;spi --init</code> <code>=&gt;</code>	按默认 IO 口初始化
<code>=&gt;spi 0x55 0xAA</code> <code>=&gt;</code>	SPI 写入 2 字节: 0x55、0xAA, 忽略读回的值
<code>=&gt;spi -r 1 2 3 4</code> <code>0xFF 0xFF 0xFF 0xFF</code> <code>=&gt;</code>	SPI 写入 4 字节: 0x01、0x02、0x03、0x04 读回 4 字节: 0xFF、0xFF、0xFF、0xFF

=>spi --deinit =>	IO 口恢复
----------------------	--------

注：用于调试外部器件、模块。

## i2c

别名：无

功能：IO 口模拟 I<sup>2</sup>C 控制

语法： `i2c [--delay=<delay_us>] [-a <address>] [--sda=<sda_pin>] [--scl=<scl_pin>] [-I] [-D] [-l] [-n] [-r <read_cycle>] <value>`  
options:  
--delay default 5  
-a/--address default 0  
--sda default 0.0  
--scl default 0.1  
-I/--init init pins  
-D/--deinit deinit pins  
-l/--lsb lsb first  
-n/--nostop no stop bit  
-r/--read default 0  
value data

示例	注释
=>i2c -a 0x68 --init =>	按默认 IO 口初始化，指定器件地址
=>i2c 0x00 =>	i2c 写入 1 字节：0x00， 忽略读回的值
=>i2c -r 4 0xFF 0xFF 0xFF 0xFF =>	i2c 读回 4 字节：0xFF、0xFF、0xFF、0xFF
=>i2c --deinit =>	IO 口恢复

注：用于调试外部器件、模块。

## 6、文件控制类

### ls

别名：1

功能：打印文件列表

语法： `ls <path>`

options:

path path name

示例	注释
<pre>=&gt;ls /r:   120  readme     15  build /s:   192  logger =&gt;</pre>	挂载点说明: <i>/r</i> 为集成编译在 FLASH-ROM 中的只读系统 <i>/s</i> 为外部 SPI-FLASH 芯片中的日志系统 <i>/c</i> 为后期烧入 FLASH-ROM 中的配置内容 <i>/f</i> 为外部 SD 卡中的 FAT 系统
<pre>=&gt;ls /r /r:   120  readme     15  build =&gt;</pre>	仅显示指定挂载点的文件
<pre>=&gt;ls /r/readme /r:   120  readme =&gt;</pre>	仅显示单个文件

## cat

别名: 无

功能: 打印/写入文件内容

语法: `cat [-b] [-w] [-a] [-d <delay>] <file>`

options:

-b/--b64 base 64 code  
 -w/--write write mode  
 -a/--append append mode  
 -d/--delay output delay in ms  
 file file name

示例	注释
<pre>=&gt;cat /r/readme Shell Lab is based on MCUSH platform. http://mcush.com/shell-lab/ Shanghai Linkong Software Technologies Co., Ltd. 2019 =&gt;</pre>	明文打印文件内容
<pre>=&gt;cat -b /r/readme U2h1bGwgTGFiIGlzIGJhc2VkaW9uIE1DVVNIIHBSYXRmb3JtLgpodHRwOi8vbWN1c2guY29tL3NoZWxslWxhYi8KU2hhbmdoYWkgTGlua29uZyBTb2Z0d2FyZSBUZWNobm9sb2dpZXMgQ28uL0CBMdgQwIDlwMTkk</pre>	BASE64 格式打印文件内容

<pre>=&gt; =&gt;cat -w /s/test &gt;abcdefghijklmnopqrstuvwxy &gt; =&gt;cat /s/test abcdefghijklmnopqrstuvwxy  =&gt;</pre>	<p>创建文件并写入内容</p>
<pre>=&gt;cat -a /s/test &gt;1234567890 &gt; =&gt;cat /s/test abcdefghijklmnopqrstuvwxy 1234567890  =&gt;</pre>	<p>追加文件内容</p>
<pre>=&gt;cat -w -b /s/test2 &gt;YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXoKMTIzNDU2Nzg5MAo= &gt; =&gt;cat /s/test2 abcdefghijklmnopqrstuvwxy 1234567890  =&gt;cat -b /s/test2 YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXoKMTIzNDU2Nzg5MAo=  =&gt;</pre>	<p>以 <b>BASE64</b> 编码方式创建文件并写入内容 可写入非 <b>ASCII</b> 的内容 (常用于自动化测试)</p>

## cp

别名: 无

功能: 复制文件

语法: cp <file>

options:

file src -> dst

示例	注释
<pre>=&gt;cp /r/readme /s/readme =&gt;l /s /s: ... 120 readme =&gt;</pre>	

## rm

别名: 无

功能: 删除文件

语法: `rm <file>`

options:

`file`                    `file name`

示例	注释
<code>=&gt;rm /s/readme</code>	删除成功
<code>=&gt;rm /r/readme</code>	删除失败
<code>!&gt;</code>	

## rename

别名: 无

功能: 重命名文件

语法: `rename <file>`

options:

`file`                    `old -> new`

示例	注释
<code>=&gt;rename /s/test test.old</code>	新文件不能包含路径
<code>=&gt;</code>	

## crc

别名: 无

功能: 计算文件 CRC32 校验值

语法: `crc <file>`

options:

`file`                    `file name`

示例	注释
<code>=&gt;crc /r/readme</code> <code>0x69C24F04</code>	
<code>=&gt;</code>	

## spiffs

别名: s

功能: spiffs 文件系统控制

语法: spiffs [-b <address>] [-c <cmd\_name>] [-C] [--compact]

options:

-b/--address base address

-c/--command

id|erase|read|write|mount|umount|test|format|check|info

-C/--ascii ascii

--compact compact output

示例	注释
<pre>=&gt;s total: 8033255 used: 502 =&gt;</pre>	检查系统容量
<pre>=&gt;s -c umount =&gt;s -c mount =&gt;</pre>	卸载文件系统 重新挂载文件系统
<pre>=&gt;s -c id EF6017 =&gt;</pre>	检查 SPI FLASH 芯片 ID 号
<pre>=&gt;s -c check 0 =&gt;</pre>	检查修复文件系统的完整性
<pre>=&gt;s -c format =&gt;</pre>	格式化文件系统
<pre>=&gt;s -c test =&gt;ls /s /s:   38 test.dat =&gt;cat /s/test.dat abcdefghijklmnopqrstuvwxy z0123456789 =&gt;</pre>	测试文件系统 创建/s/test.dat 并写入内容
<pre>=&gt;s -c read -b 0 00000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00000020: 00 00 00 00 00 00 00 00 01 80 01 00 FF FF FF FF 00000030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00000090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 000000A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF</pre>	读取 SPI FLASH 芯片原始内容



```
000000B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
=>
```

## 7、网络控制类

### netstat

别名: 无

功能: 查询网络接口状态

语法: netstat [-c <command>]

options:

-c/--cmd info|up|down|dhcp|ip|dns

示例	注释
<pre>=&gt;netstat mac: 00:11:22:33:44:55 dhcp: 0 ip: 10.168.2.227 netmask: 255.255.255.0 gateway: 10.168.2.1 dns1: 10.168.2.1 dns2: 114.114.114.114 =&gt;</pre>	<p>查询当前接口状态  <b>dhcp: 0-静态指定 1-自动获取</b></p>
<pre>=&gt;netstat -c down =&gt;netstat -c up =&gt;</pre>	<p>重置接口</p>
<pre>=&gt;netstat -c ip &gt;10.168.2.100 &gt;255.255.255.0 &gt;10.168.2.1 &gt; =&gt;</pre>	<p>手动修改 IP/网络掩码/网关地址</p>
<pre>=&gt;netstat -c dns &gt;114.114.114.114 &gt;8.8.8.8 &gt; =&gt;</pre>	<p>手动修改 DNS1/DNS2</p>

## ping

别名: 无

功能: PING 测试

语法: ping <host>

options:

host                    hostname or ip addr

示例	注释
<pre>=&gt;ping www.baidu.com dns resolve: www.baidu.com dns resolved: 180.101.49.12 ping: send 180.101.49.12 ping: recv 180.101.49.12  20 ms ping: send 180.101.49.12 ping: recv 180.101.49.12  12 ms ping: send 180.101.49.12 ping: recv 180.101.49.12  20 ms ping: send 180.101.49.12 ping: recv 180.101.49.12  16 ms =&gt;</pre>	测试 baidu.com
<pre>=&gt;ping www.error_server.com dns resolve: www.error_server.com dns resolve failed !&gt;</pre>	DNS 解析错误
<pre>=&gt;ping 10.168.2.200 ping: send 10.168.2.200 ping: send 10.168.2.200 ping: send 10.168.2.200 ping: send 10.168.2.200 =&gt;</pre>	测试无响应

## nc

别名: 无

功能: TCP 测试

语法: nc <host> <port>

options:

host                    hostname or ip addr

port                    port

示例	注释
<pre>=&gt;nc www.baidu.com 80 dns resolve: www.baidu.com</pre>	创建 TCP 连接 DNS 解析

<pre>dns resolved 180.101.49.12 connected GET /index.html HTTP/1.1  HTTP/1.1 200 OK Accept-Ranges: bytes Cache-Control: no-cache Connection: Keep-Alive Content-Length: 14615 Content-Type: text/html Date: Wed, 04 Sep 2019 02:41:33 GMT Etag: "5d64e2cf-3917" Last-Modified: Tue, 27 Aug 2019 07:59:11 GMT P3p: CP=" OTI DSP COR IVA OUR IND COM " Pragma: no-cache Server: BWS/1.1 Set-Cookie: BAIDUID=E1A5DD0AEDEC007B8E607FBB11855C4B:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com Set-Cookie: BIDUPSID=E1A5DD0AEDEC007B8E607FBB11855C4B; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com Set-Cookie: PSTM=1567564893; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com Vary: Accept-Encoding X-Ua-Compatible: IE=Edge,chrome=1  &lt;!DOCTYPE html&gt;&lt;!--STATUS OK--&gt; &lt;html&gt; &lt;head&gt;   &lt;meta http-equiv="content-type" content="text/html; charset=utf-8"&gt;   &lt;meta http-equiv="X-UA-Compatible" content="IE=Edge"&gt;   &lt;link rel="dns-prefetch" href="//s1.bdstatic.com"/&gt;   &lt;link rel="dns-prefetch" href="//t1.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//t2.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//t3.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//t10.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//t11.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//t12.baidu.com"/&gt;   &lt;link rel="dns-prefetch" href="//b1.bdstatic.com"/&gt;   &lt;title&gt;百度一下, 你就知道&lt;/title&gt;   ...   ...   ... &lt;/body&gt;&lt;/html&gt;</pre>	<p>连接成功 输入 GET 指令 空行确认 返回结果</p> <p>服务器仍保持连接, Ctrl-C 中止连接</p>
--	--

=>

## wget

别名: 无

功能: 通过网络下载文件

语法: `wget [-u <url>] [-f <output file>]`

options:

`-u/--url`            `http://...`

`-f/--file`           `output file name`

示例	注释
<pre>=&gt;wget -u http://www.baidu.com/index.htm l -f /s/baidu.html dns resolve: www.baidu.com dns resolved: 180.101.49.11 14615 bytes saved =&gt;ls /s/baidu.html /s:  14615  baidu.html =&gt;</pre>	<p>下载百度首页</p>

注: 目前仅支持 HTTP/GET 方式下载, 不支持 FTP 下载。

## lwip

别名: 无

功能: 查询 LWIP 网络栈资源状态

语法: 无参数

示例	注释
<pre>=&gt;lwip  LINK   xmit: 0   recv: 0   fw: 0   drop: 0   chkerr: 0   lenerr: 0   memerr: 0   rterr: 0   proterr: 0   opterr: 0</pre>	<p>包含以下分类组:</p> <p>LINK ETHARP IP ICMP UDP TCP MEM HEAP MEM RAW_PCB MEM UDP_PCB MEM TCP_PCB MEM TCP_PCB_LISTEN MEM TCP_SEG</p>

err: 0	MEM NETBUF
cachehit: 0	MEM NETCONN
ETHARP	MEM TCPIP_MSG_API
xmit: 4	MEM TCPIP_MSG_INPKT
recv: 3	MEM SYS_TIMEOUT
fw: 0	MEM NETDB
drop: 0	MEM PBUF_REF/ROM
chkerr: 0	MEM PBUF_POOL
lenerr: 0	SYS
memerr: 0	
rtterr: 0	
protterr: 0	
opterr: 0	
err: 0	
cachehit: 12	
IP	
xmit: 13	
recv: 34	
fw: 0	
drop: 3	
chkerr: 0	
lenerr: 0	
memerr: 0	
rtterr: 0	
protterr: 0	
opterr: 0	
err: 0	
cachehit: 0	
ICMP	
xmit: 0	
recv: 0	
fw: 0	
drop: 0	
chkerr: 0	
lenerr: 0	
memerr: 0	
rtterr: 0	
protterr: 0	
opterr: 0	
err: 0	
cachehit: 0	

UDP

xmit: 1  
recv: 15  
fw: 0  
drop: 0  
chkerr: 0  
lenerr: 0  
memerr: 0  
rtterr: 0  
proterr: 0  
opterr: 0  
err: 0  
cachehit: 0

TCP

xmit: 7  
recv: 16  
fw: 0  
drop: 0  
chkerr: 0  
lenerr: 0  
memerr: 0  
rtterr: 0  
proterr: 0  
opterr: 0  
err: 0  
cachehit: 16

MEM HEAP

avail: 10240  
used: 0  
max: 0  
err: 0

MEM RAW\_PCB

avail: 4  
used: 0  
max: 0  
err: 0

MEM UDP\_PCB

avail: 6  
used: 1  
max: 1  
err: 0

```
MEM TCP_PCB
  avail: 20
  used: 1
  max: 1
  err: 0

MEM TCP_PCB_LISTEN
  avail: 5
  used: 1
  max: 1
  err: 0

MEM TCP_SEG
  avail: 20
  used: 0
  max: 1
  err: 0

MEM NETBUF
  avail: 2
  used: 0
  max: 0
  err: 0

MEM NETCONN
  avail: 4
  used: 0
  max: 0
  err: 0

MEM TCPIP_MSG_API
  avail: 8
  used: 0
  max: 0
  err: 0

MEM TCPIP_MSG_INPKT
  avail: 8
  used: 0
  max: 1
  err: 0

MEM SYS_TIMEOUT
  avail: 10
```

```
used: 6
max: 6
err: 0

MEM NETDB
  avail: 1
  used: 0
  max: 0
  err: 0

MEM PBUF_REF/ROM
  avail: 100
  used: 0
  max: 0
  err: 0

MEM PBUF_POOL
  avail: 40
  used: 0
  max: 1
  err: 0

SYS
  sem.used: 0
  sem.max: 0
  sem.err: 0
  mutex.used: 0
  mutex.max: 0
  mutex.err: 0
  mbox.used: 1
  mbox.max: 1
  mbox.err: 0

=>
```

## 8、日志控制类

### log

别名: 无

功能: 控制 logger 任务实现日志管理

语法: `log [-d] [-e] [-b] [--delete] [-t] [-D] [-I] [-W] [-E] [-M <module>] [-H <head>] [-m <message>]`



```
options:
  -d/--disable    disable logging to file
  -e/--enable    enable logging to file
  -b/--backup    backup history files
  --delete       delete history files
  -t/--tail      list tail 10 lines from log file
  -D/--debug     DEBUG type filter
  -I/--info      INFO type filter
  -W/--warn      WARN type filter
  -E/--error     ERROR type filter
  -M/--module    module filter
  -H/--head      message head filter
  -m/--msg       log message
```

### 示例

查看实时日志, Ctrl-C 中止

```
=>log
2019-9-4 09:00:11 I dhcpc: cable connected
2019-9-4 09:00:11 D modbus: listening on port 502, pcb=0x1000CCC8
2019-9-4 09:00:11 I alink: server ip 106.15.100.2
2019-9-4 09:00:11 D alink: bind port 50987
2019-9-4 09:00:12 I alink: connected
=>
```

查看实时日志, 过滤出所有警告和错误级别的消息

```
=>log -W -E
...(waiting for WARN and ERROR level messages)...
```

查看实时日志, 过滤出 client 模块的消息

```
=>log -M client
...(waiting for client module messages)...
```

查看实时日志, 过滤出 dhcpc 模块的内容以“cable ”引导的消息

```
=>log -M dhcpc -H "cable "
...(waiting for dhcpc module messages leading with "cable ")...
```

查看末尾 10 行的日志记录

```
=>log -t
2019-9-4 09:00:05 I init: device_name ENG001
2019-9-4 09:00:05 I init: server iot.linkongsoft.com, port 10000
2019-9-4 09:00:06 I dhcpc: mac: 00:11:22:33:44:55
2019-9-4 09:00:06 I dhcpc: config ip: 10.168.2.243 netmask: 255.255.255.0 gateway:
10.168.2.1
2019-9-4 09:00:06 I dhcpc: cable disconnected
2019-9-4 09:00:11 I dhcpc: cable connected
2019-9-4 09:00:11 D modbus: listening on port 502, pcb=0x1000CCC8
2019-9-4 09:00:11 I alink: server ip 106.15.100.2
2019-9-4 09:00:11 D alink: bind port 50987
2019-9-4 09:00:12 I alink: connected
=>
```

禁止写日志文件 <pre>=&gt;log -d =&gt;</pre>
允许写日志文件 <pre>=&gt;log -e =&gt;</pre>
手动追加日志 <pre>=&gt;log -m "test message" =&gt;</pre>
备份所有日志文件，所有/s/logger[.N]被重命名为/s/logger[.N].bak <pre>=&gt;log --backup =&gt;ls ...   977  logger.bak 20017 logger.1.bak 20003 logger.2.bak 20048 logger.3.bak 20002 logger.4.bak 20003 logger.5.bak 20048 logger.6.bak ... =&gt;</pre>
删除所有日志文件，删除所有/s/logger[.N]，但不删除备份文件/s/logger[.N].bak <pre>=&gt;log --delete =&gt;</pre>
删除所有日志文件（包含备份），删除所有/s/logger[.N]和/s/logger[.N].bak <pre>=&gt;log -b --delete =&gt;</pre>

注：此命令需要启动 **logger** 任务，位于 **appLogger** 目录下。

## 9、其它

### loop

别名：无

功能：循环执行参数指令

语法： `loop [-l <loop_delay_ms>] [-n <cycle_limit>] <command>`  
 options:  
   -l/--loop            default 1000ms  
   -n/--number        cycle\_limit  
 command              cmd with args

示例	注释
----	----

<pre>=&gt;loop led -i0 -t =&gt;</pre>	循环翻转第 0 编号的 LED，节拍 1 秒
<pre>=&gt;loop -l100 led -i0 -t =&gt;</pre>	同上，节拍 0.1 秒
<pre>=&gt;loop -l100 -n100 led -i0 -t =&gt;</pre>	同上，限制 100 次循环（LED 闪烁约 10 秒）

注：为调试方便，某些常用命令也会集成循环参数（如 `gpio`）。

## error

别名：e

功能：控制 blink 任务实现 LED 闪烁错误号

语法： `error [-s] <errno>`  
 options:  
   -s/--stop           stop  
 errno                0~100000000

示例	注释
<pre>=&gt;e 0 =&gt;</pre>	打印当前错误号
<pre>=&gt;e 12 =&gt;</pre>	修改当前错误号为 12
<pre>=&gt;e -s =&gt;e stop =&gt;</pre>	任务停止，释放响应 LED 的控制权

注：此命令需要启动 blink 任务，位于 `appBlinkErrorNumber` 目录下。