开发教程目录

1.文档-开发环境

开发环境：包括开发主机（development Host）和目标系统（Target），它们通过USB线连接，即可完成APP的调试和下载。

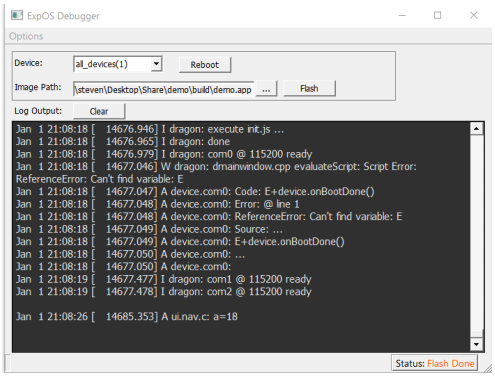
- 安装Studio

ExpOS操作系统已预装在目标系统，用户只需在Windows 7/10开发主机上一键安装最新版本ExpOS Studio即可.

注意：在Windows XP系统上，Studio不支持通过USB线与目标连接，但支持除了USB下载和日志调试的所有编辑模拟功能。Windows XP用户可将APP拷贝到U盘（U盘需格式化成FAT32格式），将U盘插入目标系统也可完成烧写，但无法实现日志方式调试APP.

- 验证USB连接

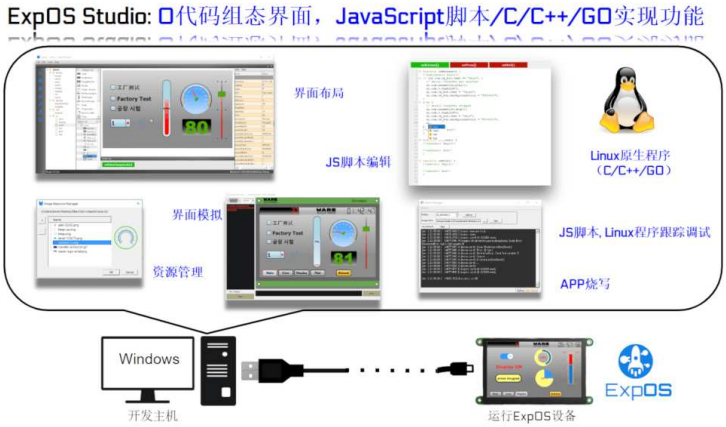
1. – 将目标系统与主机通过USB线连接，启动目标系统，待启动完毕。
2. – 启动ExpOS Studio软件，点击菜单File(文件)->New Project(新建工程) 或者Open Project(打开工程)，新建或者打开已有工程。然后点击菜单Tools(工具)->debugger(调试器)启动调试器。
3. – 如果调试器与目标连接成功，右下角状态栏显示连接状态为connected，左下角显示当前APP和OS的版本，同时可看到OS的启动日志。这时点击菜单options（选项）->device info（设备信息）可显示当前目标的设备信息。
4. – 点击Flash（烧写）按钮，可将指定路径的APP或ExpOS固件成功下载到目标



- Studio简介

可视化集成开发环境Studio提供一站式解决方案，无需再安装其他软件，即可完成所有软件开发工作，包括界面管理、配置、代码编辑、构建、模拟、调试，下载等：

- UI编辑器（通过鼠标拖，拉和点击，所见即所得，0代码生成界面）
- 脚本代码编辑器（输入JavaScript脚本描述APP行为，支持语法高亮，智能提示，代码自动补全）
- 模拟器（无需目标系统，在开发主机上快速验证APP）
- USB调试器（日志方式调试程序，烧写APP到目标系统）



2.APP开发

ExpOS操作系统软基于现代软件设计理念：面向对象操作、事件驱动执行，符合人的思维和操作习惯。但不同于其他操作系统，ExpOS对计算机软件概念进行了可视化封装，隐藏了许多编程细节，大大简化软件设计过程，减小开发工作量。无论有无经验，开发者都能短时间内设计出稳定，可靠，炫酷的应用程序.

与其他面向对象软件一样，ExpOS中所有的软件图形元素（如按钮、文本框、进度条等）和硬件单元（如串口、背光、蜂鸣器等）都被看作一个控件对象，每个对象都有名称，与对象相关，有四个基本概念：

属性 —— 对象的状态或特征，如图形元素的颜色，串口的波特率等

方法 —— 对象的行为，以函数形式出现，如时钟控件对象的setTime()，表示设置时间

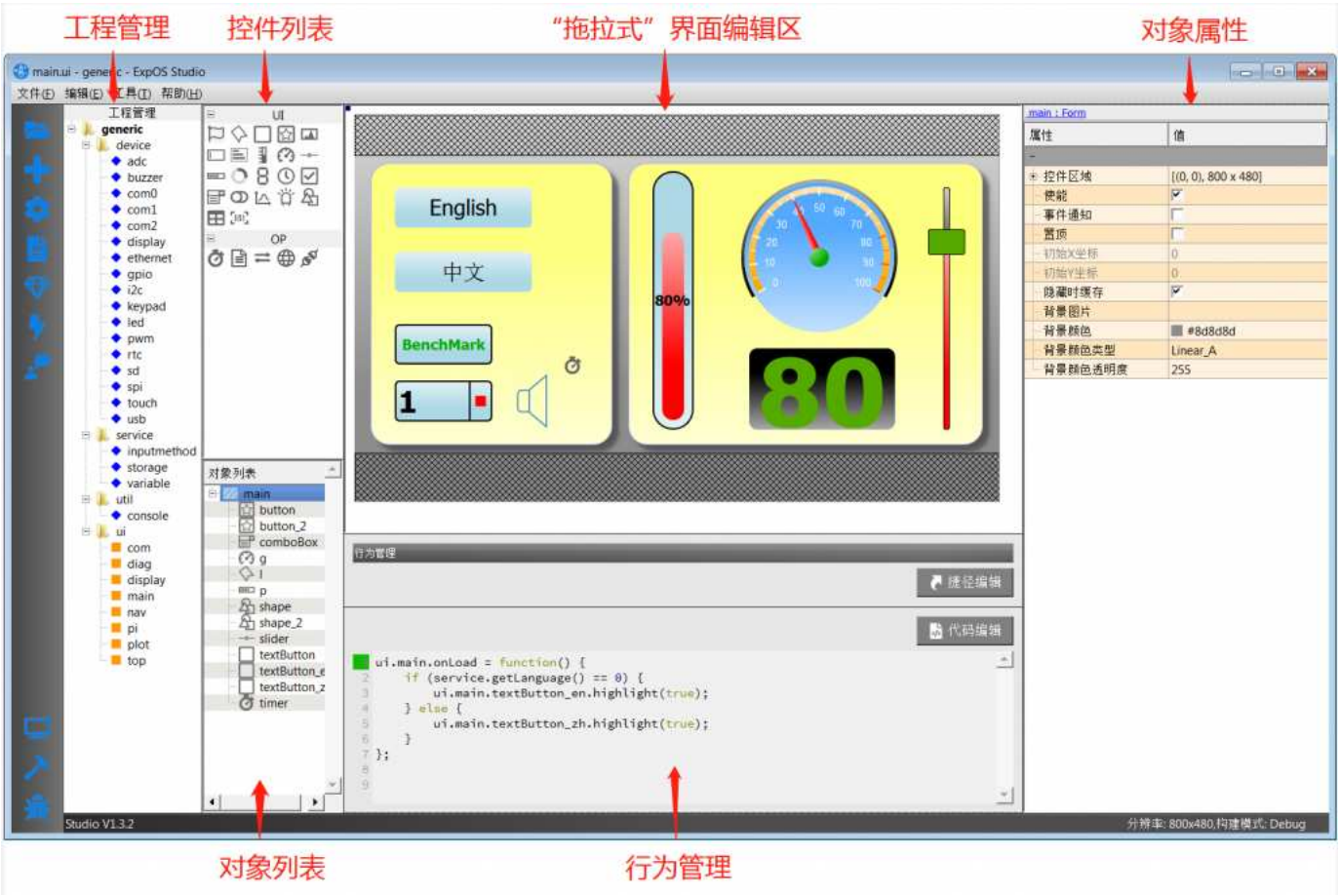
事件 —— 对象对某种条件的响应，如按下按钮，会触发"按下"事件

动作 —— 对象在事件中运行JavaScript脚本的过程

面向对象操作——通过改变某个控件对象的属性或者在软件中调用该对象的方法， 实现对该对象的访问和控制。

事件驱动执行——大部分控件对象都有自己的事件，当事件在某种条件下被触发时（如按下按钮），对应控件对象的动作将会自动执行。

针对上述概念，可视化集成开发工具Studio设计了不同的控制区域，使用户能方便快速地进行软件开发：



软件开发过程可简单概括为：

在Studio中鼠标拖/拉/点击，0代码可视化创建/设置控件对象，生成静态界面

编写各控件对象的动作脚本代码完成APP运行时的逻辑控制和动态效果

完整的APP开发步骤：

2.1.控件对象

ExpOS软件设计是完全基于可视化对象，通过Studio或者脚本操作控件对象的属性，方法，事件和动作，实现APP界面和软件功能。控件对象之间有从属关系，即父子关系，最顶层的父对象称为根对象，包括四个：

■界面对象(ui)
将不同的界面软件元素抽象为用户控件，如：文本框、按钮、图片、进度条等，一般在用户界面中可见。用户控件允许用户将控件拖拽到页面设计区创建、移动和删除，并且同一控件可以多次使用。系统启动时，根据用户的设计，动态加载和创建界面控件；访问其方法和属性时，必须确保控件所在的页面已经存在（当前页面或者有"隐藏时缓存"属性的页面）。所有的普通界面控件的父对象为页面，页面的父对象为ui。例如，设置页面xxx中的按钮yyy的属性text为"test"， 对应的JavaScript脚本为：ui.xxx.yyy.text='test'

界面对象的属性一般只支持数字和字符串类型，访问时JS脚本会做自动转换。比较特殊的属性包括：

颜色属性
ExpOS支持如下两种方式定义按三原色定义颜色：

– 16进制：格式为#RRGGBB， 其中RR表示红色，GG表示绿色，BB蓝色， 例如，#FF0000 代表红色， #FFFF00代表黄色

– 10进制：格式为rgb(r, g, b), 与16进制表示法类似，rgb(255, 0, 0) 表示红色， rgb(255, 255, 0) 代表黄色

读取相关颜色属性时，系统按16进制方式返回字符串，写入时，用户可选择任何一种方式。

图片属性
图片是由资源管理器负责管理，要使用一个图片资源，必须把该图片添加到资源管理器中。图片属性为图片名称字符串，无需指定图片路径，ExpOS会自动在资源管理器中查找到，如果要清除图片属性的设置（即无图片），设置成空字符串即可。

■设备对象（device）
将硬件相关的功能软件抽象为设备控件，如串口、触摸屏、背光、蜂鸣器等，一般在用户界面中不可见。设备控件由于与硬件一一对应，系统启动后自动创建，任何界面和控件可以无条件访问其属性和方法函数实现系统控制。用户只能对设备控件进行设置，而不允许增加、移动和删除。所有设备控件的父对象为device， 例如，向串口0发送一个字节0x55，对应的JavaScript脚本为：device.com0.write(0x55)

■服务对象（service）
将软件相关功能模块抽象为服务控件，如：输入法键盘，有的在用户界面中可见。服务控件不允许用户将控件拖拽到页面设计区创建、移动和删除，并且同一控件只有一个实例(instance)。与设备控件类似，系统启动后自动创建，任何界面和控件可以无条件访问其属性和方法函数实现服务请求。用户只能对服务控件进行设置，而不允许增加、移动和删除。所有服务控件的父对象为service， 例如，设置环境变量xx值为”test“，对应的JavaScript脚本为：service.variable.write(‘xx’,’test’)

■工具对象（util）
将软件相关工具抽象为工具控件，如：shell。工具控件不允许用户将控件拖拽到页面设计区创建、移动和删除，并且同一控件只有一个实例(instance)。与设备，服务控件类似，系统启动后自动创建，任何界面和控件可以无条件访问其属性和方法函数。用户只能对工具控件进行设置，而不允许增加、移动和删除。所有工具控件的父对象为util， 例如，重启系统，对应的JavaScript脚本为：util.shell.execute(‘reboot’)

四类控件的属性、方法可直接在动作脚本代码中调用，具体用法参看“软件速查表”。

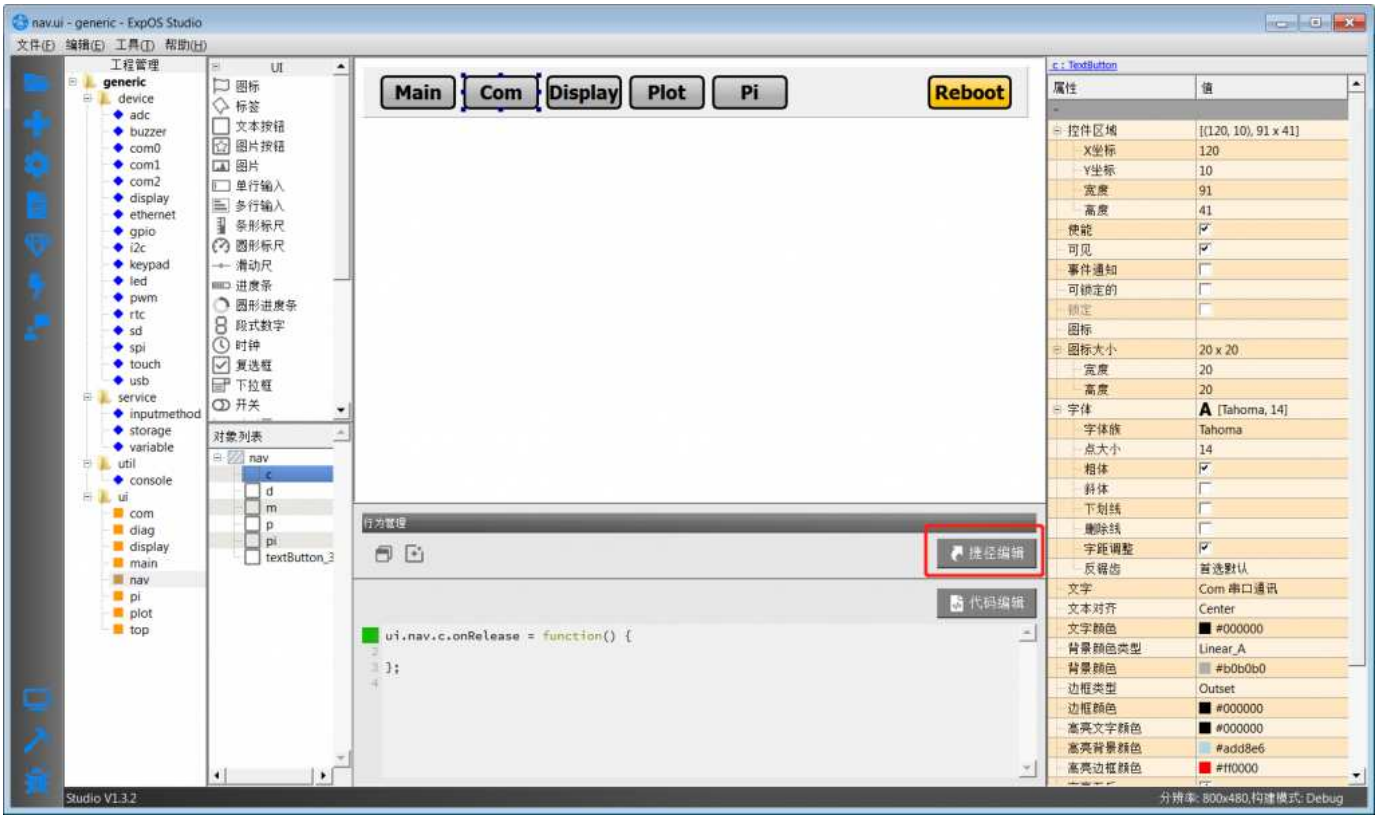
2.2.捷径

Shortcut(捷径)，顾名思义通过快捷的方式实现我们想要的一个动作行为，如切换界面，蜂鸣器滴一声，设置GPIO的输出。在我们没有引入捷径功能之前，我们要实现任何一个动作都需要写脚本来实现，如界面的切换：

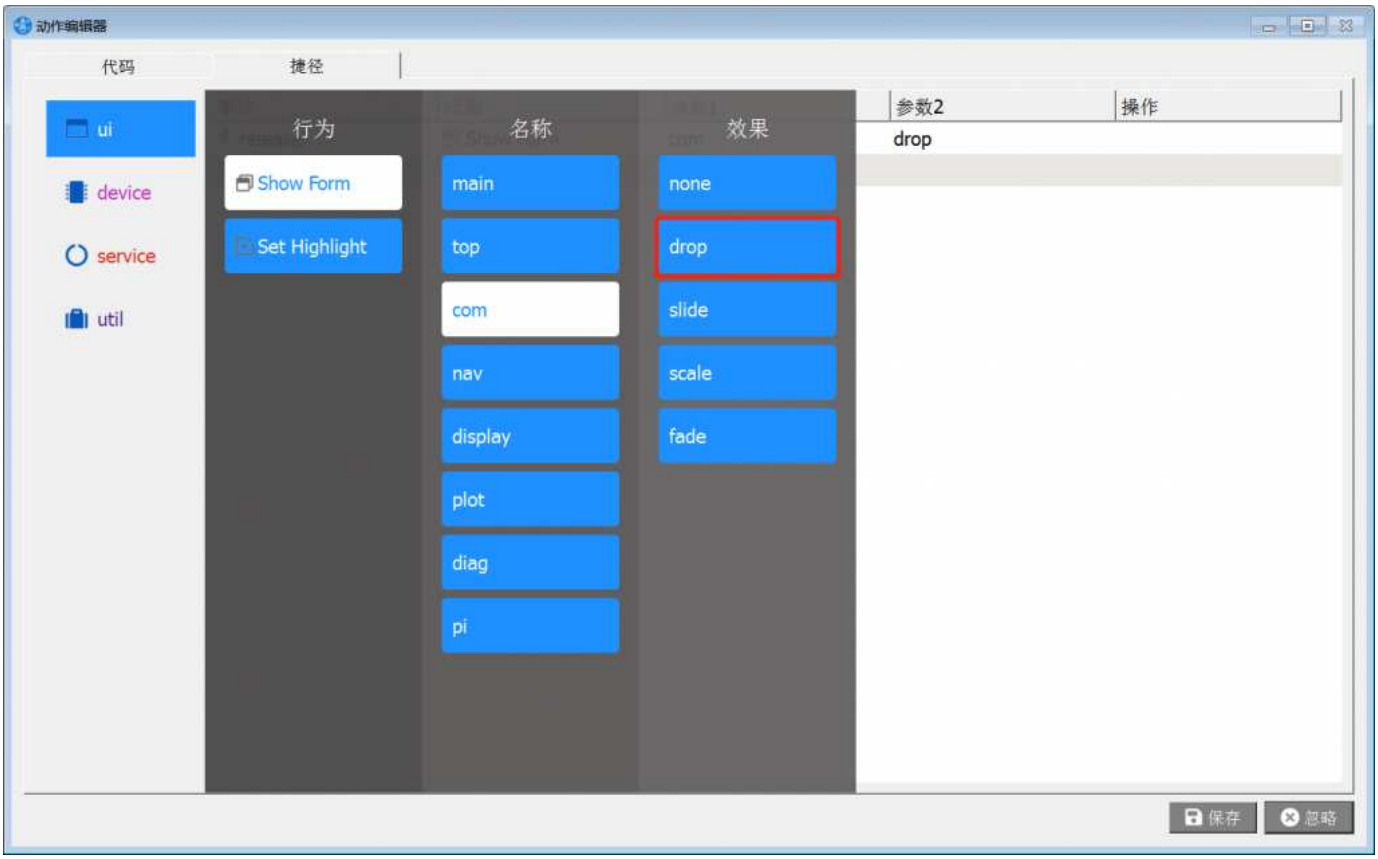
ui.page2.show(); // 切换到界面page2

那么引入捷径后，一些常用的动作行为，我们只需要点点鼠标，选择一下要切换到的界面就行，不再需要自己来写代码。下面我们以Studio自带的例程中的“综合演示”为例，演示一下捷径的使用方法。

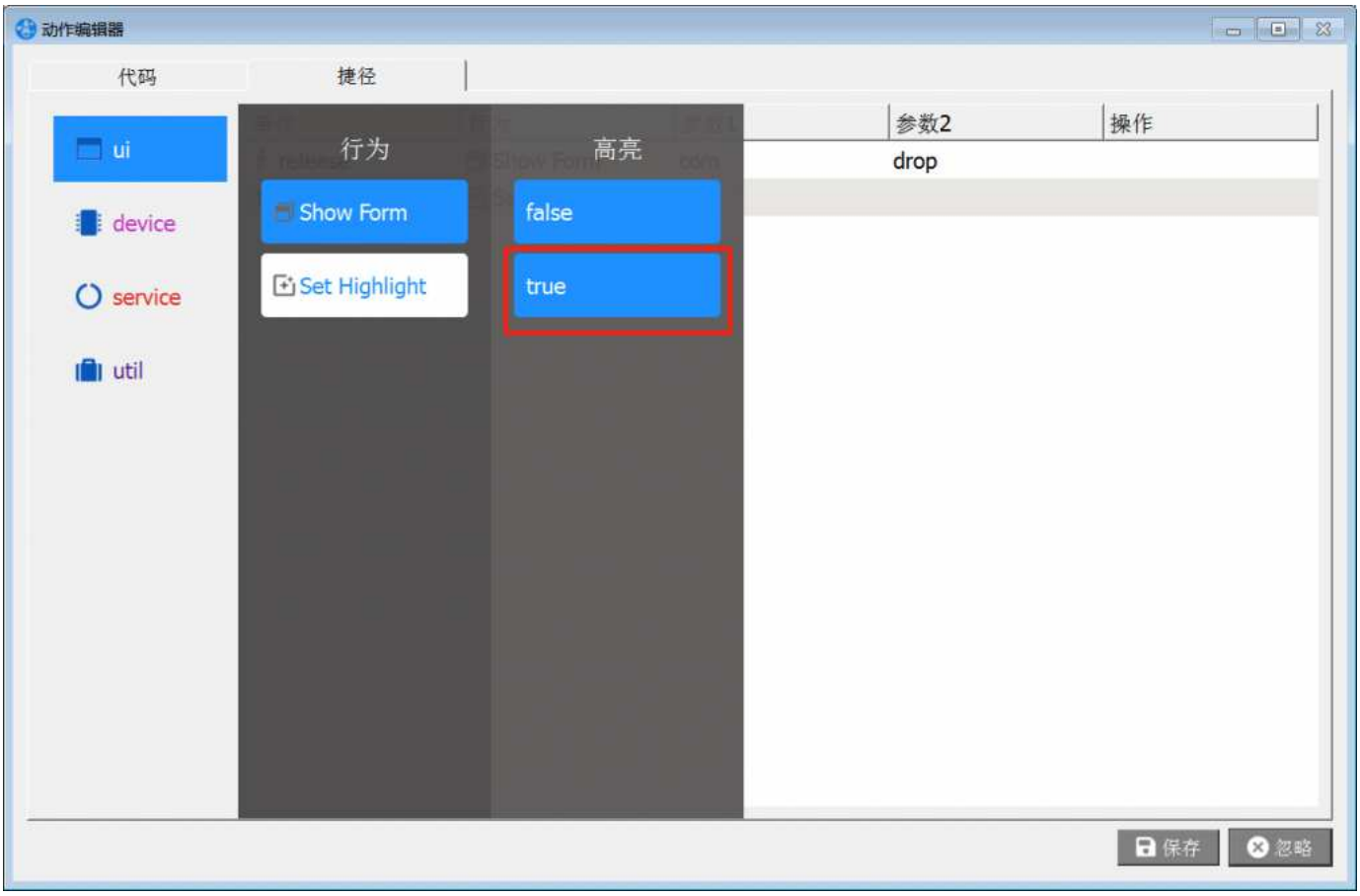
1、首先，我们打开导航页面-nav, 选择Com按钮（我们想要实现的动作行为是，点击Com按钮后，界面切换到com, 并且Com按钮显示高亮），点击行为管理区的“捷径编辑”按钮



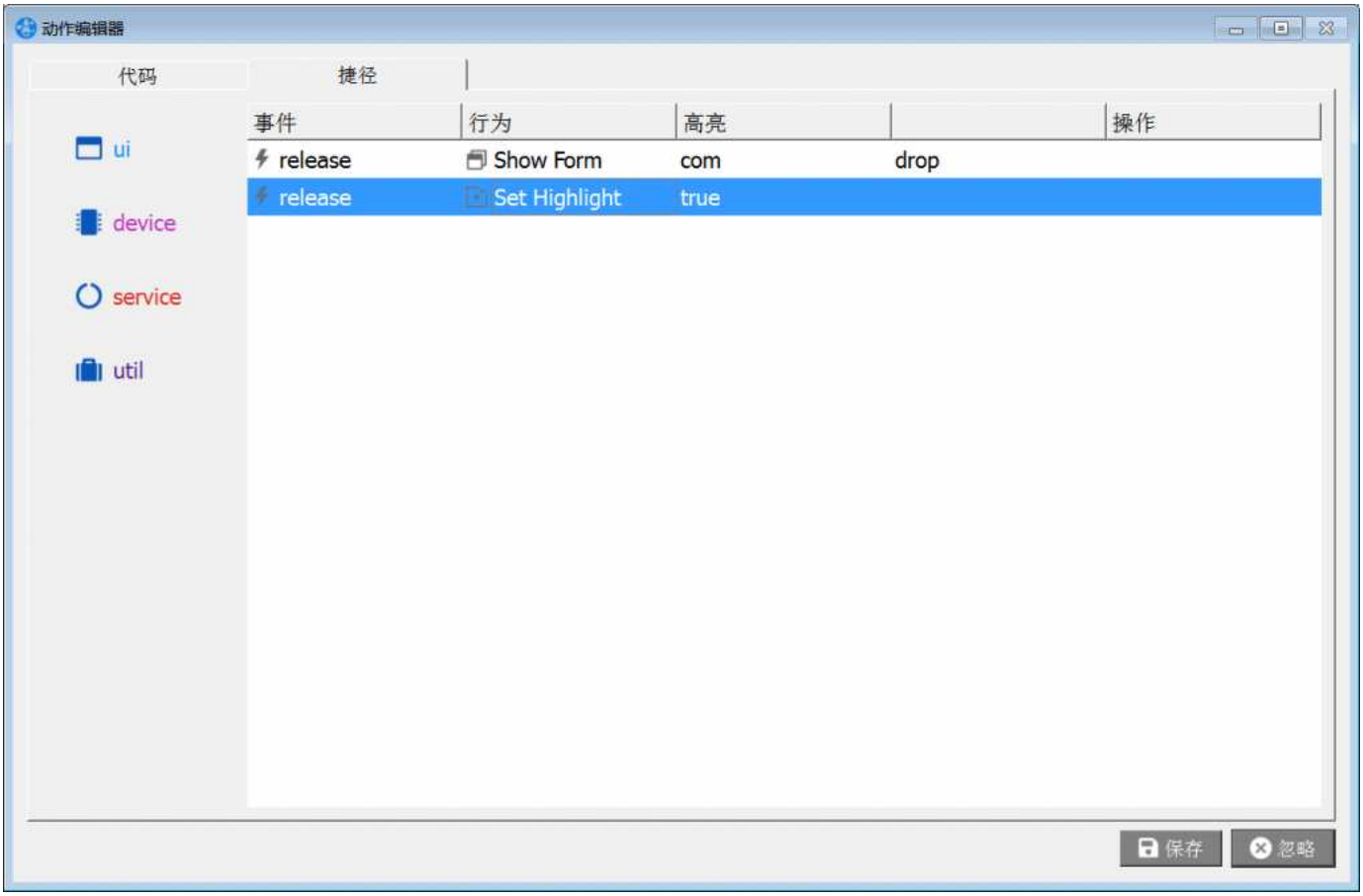
2、在打开的动作编辑器中，我们看到当前选中的是“捷径”TAB, 左边栏“ui”, “device”, “service”, “util”是对应的几种捷径的分类，切换界面属性UI操作，所以我们点击”ui”, 在弹出的面板中选择”Show Form”, 名称选择”com”, 切换效果选择”drop”, 这样我们就完成了切换界面到com的功能。



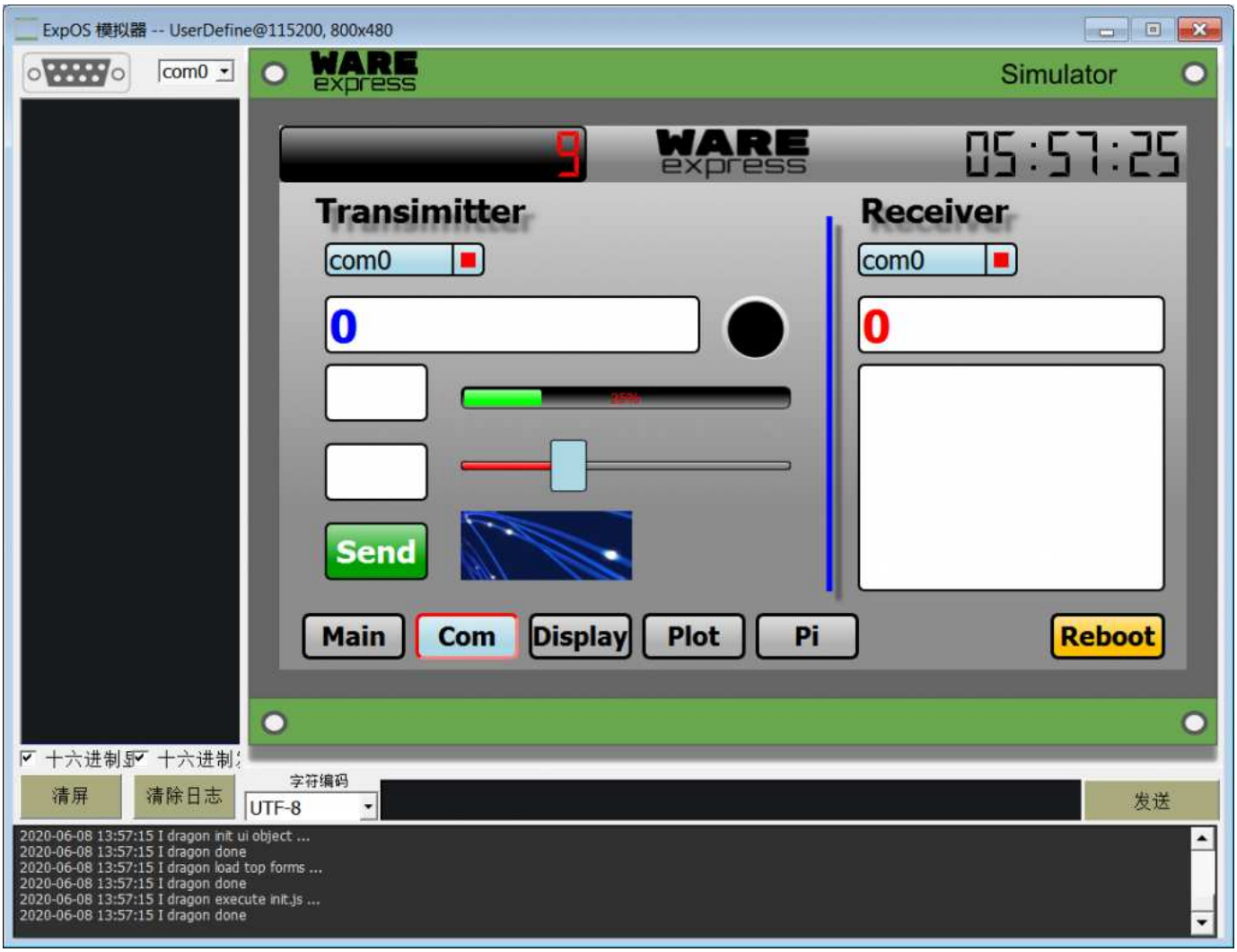
3、接着我们用同样的方法，添加Com按钮的高亮显示：



4、最后我们在捷径的列表里面能看到有两条，一条是切换界面到com，另外一条是使当前按钮高亮



5、占击”保存“按钮，运行模拟器，查看效果：



2.3.脚本

■ 脚本编程

通过Studio，不仅可设计APP静态外观、文字、风格类型等，而且也可通过控件的“动作脚本**(action)**”进行逻辑控制、运算、控件对象操作等来实现系统软件功能。

目前，动作脚本支持JavaScript核心部分ECMAScript（详细说明参看这里），语法类似C语言，在网页中大量采用，熟悉C的用户很容易掌握。

大多数控件对象支持基于事件驱动动作脚本功能，即APP在运行过程中，满足一定触发条件条件后，无需外部干预，动作脚本自动触发执行，完成预设功能，例如处理按钮事件，处理串口接收数据等。这些触发脚本执行的触发条件称为事件(event)。

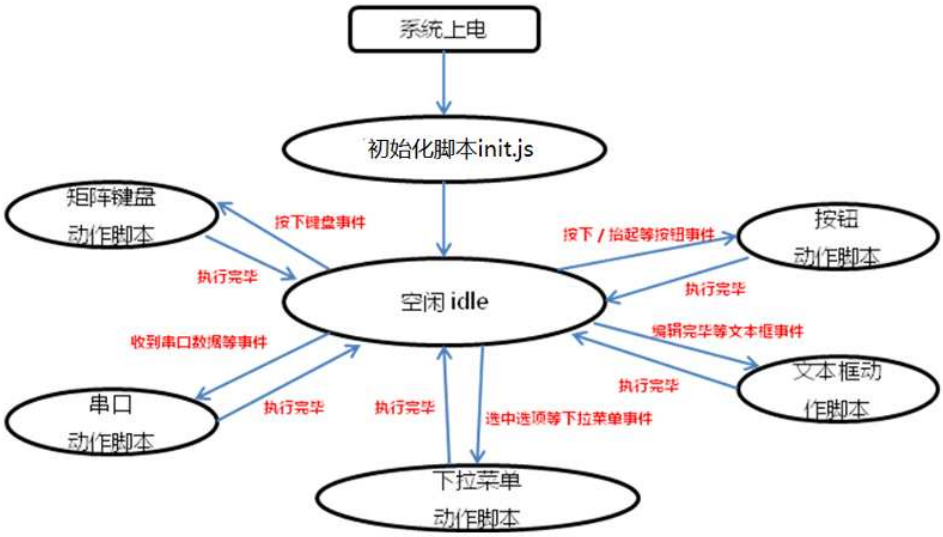
在Studio中，鼠标单击选中一个控件对象，点击Studio正下方的“代码编辑”按钮，打开动作脚本编辑器，在代码TAB中这个对象的所支持的触发事件都以函数方式显示，绿色表示该事件被选中为有效，红色表示无效。代码编辑器支持语法高亮显示，自动对齐，自动提示和补齐等功能，有效提高用户的脚本开发效率。

■ 脚本执行

当一个APP设计完成，可能会包含各种触发事件对应的脚本，ExpOS操作系统会统一管理这些脚本，保证脚本在合适的时候执行，脚本的执行过程是：

1) 目标系统上电后，首先执行一次初始化脚本init.js（初始化一些全局定义变量，定义全局函数等），然后进入空闲循环等待。

2) 当控件对象触发有效事件，立刻自动执行该控件的动作脚本（根据设计需要，用户可以进行逻辑运算、操作访问其他对象、发送串口数据等）和动作捷径，执行完毕后返回空闲循环继续等待。



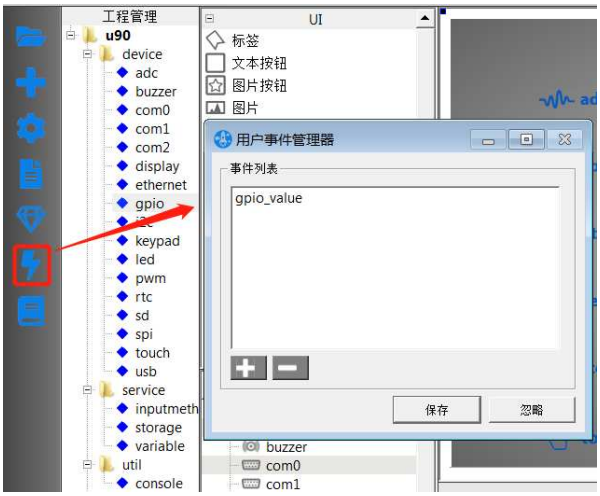
2.4.自定义事件

自定义事件

在实际的开发过程中，除了用到系统现有的事件（如按钮的释放事件）我们还可以自定义事件。

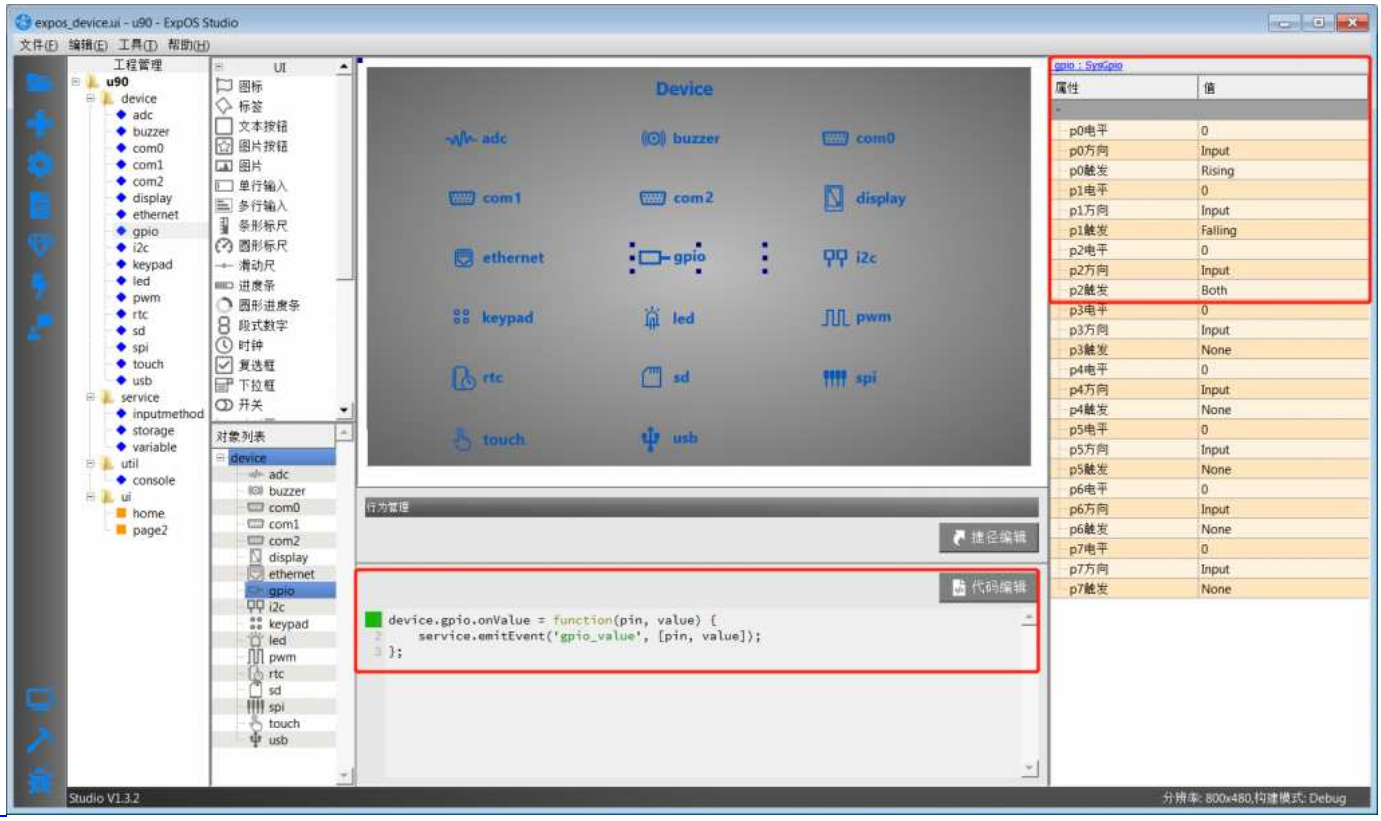
下面以自定义GPIO事件来控制LED控件的状态举例说明一下自定义事件的使用：

1、首先在Studio新建的工程中，打开用户事件管理器，点击加号按钮添加一个自定义的事件名称，如gpio_value

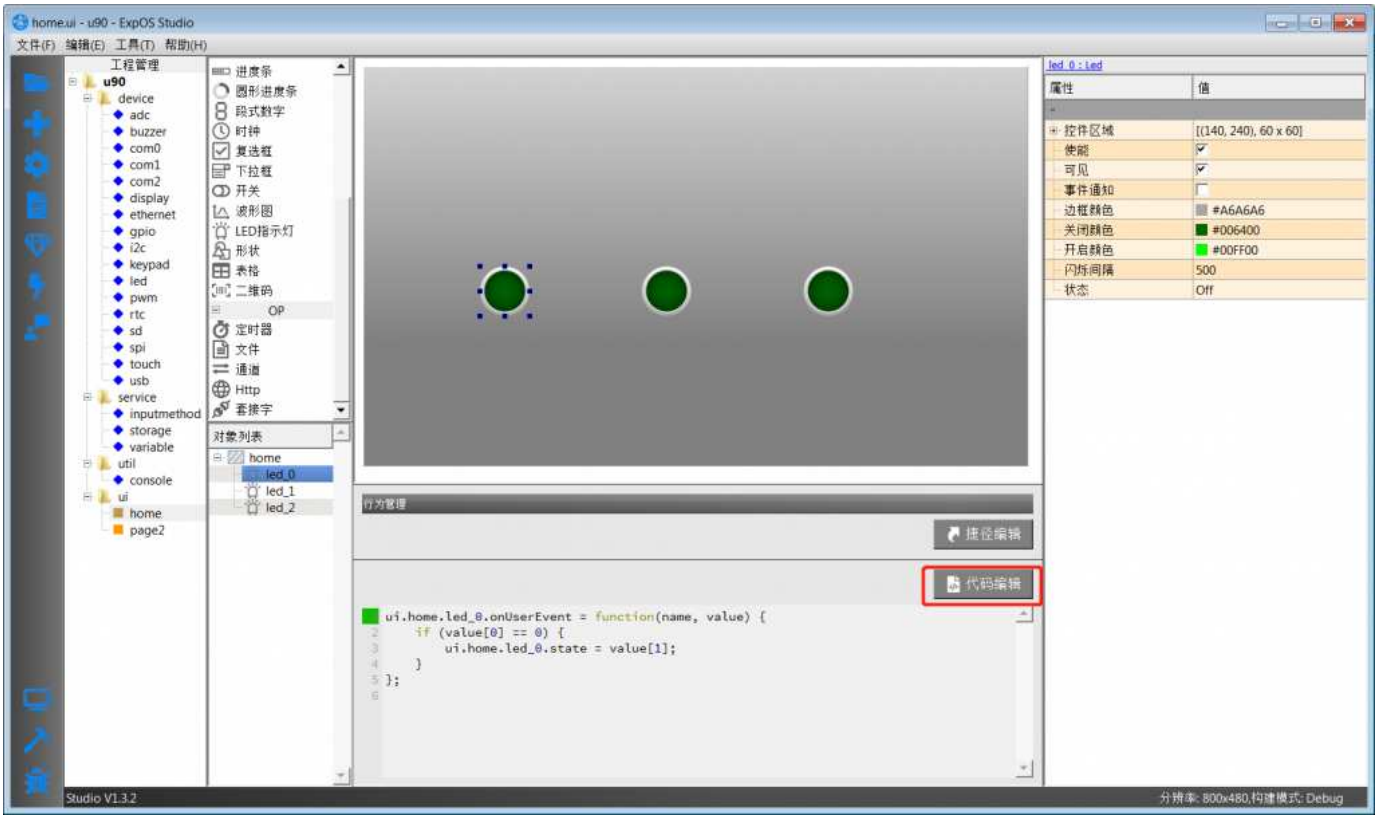


然后点“保存”按钮

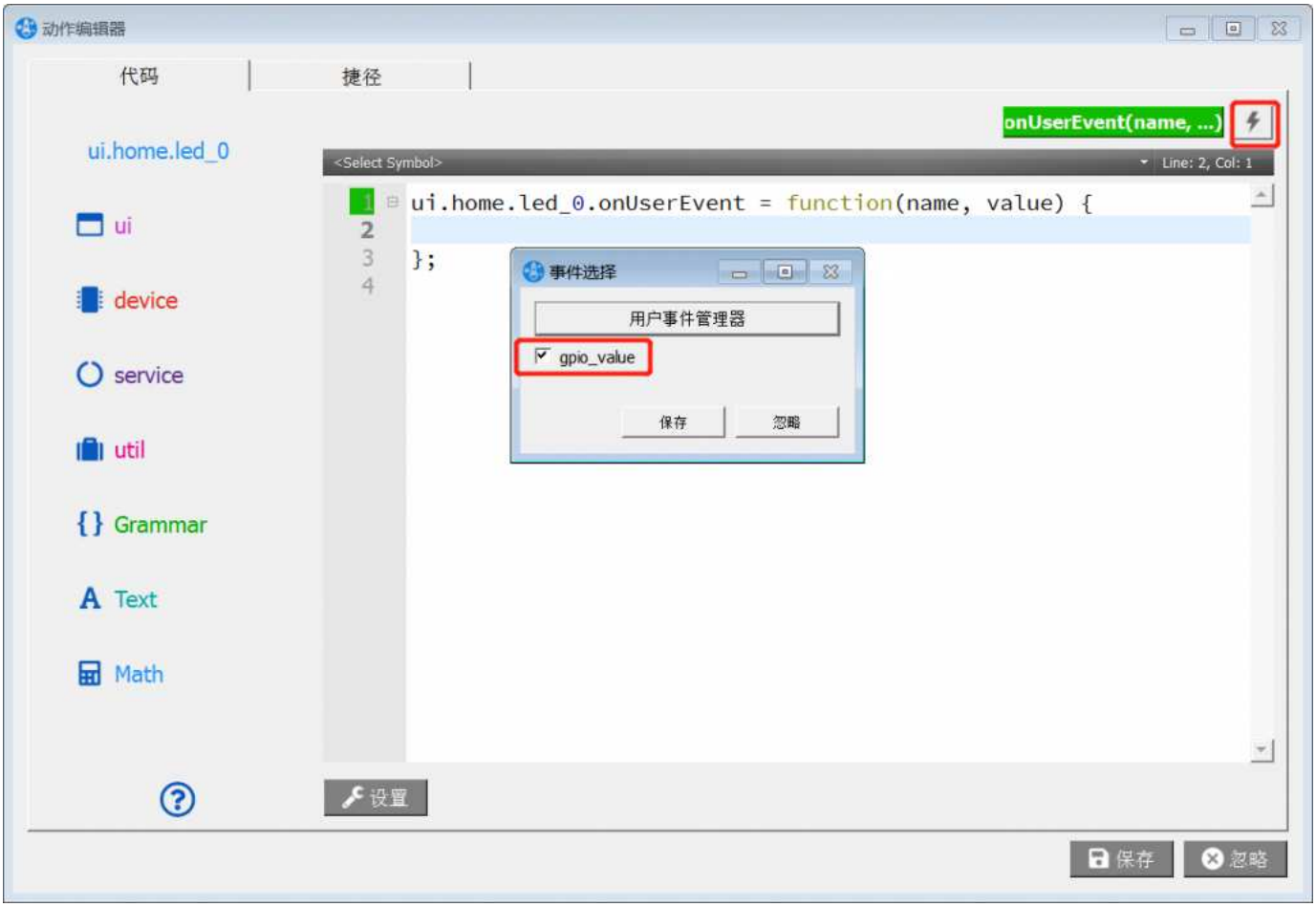
2、设置GPIO控件的p0～p2的触发属性（分别设置上升沿，下降沿触发onValue），然后在GPIO控件的onValue方法中广播自定义事件，事件的name是”gpio_value”, 事件的内容是一个数组[pin, value]（包含GPIO的端口号以及电平值）



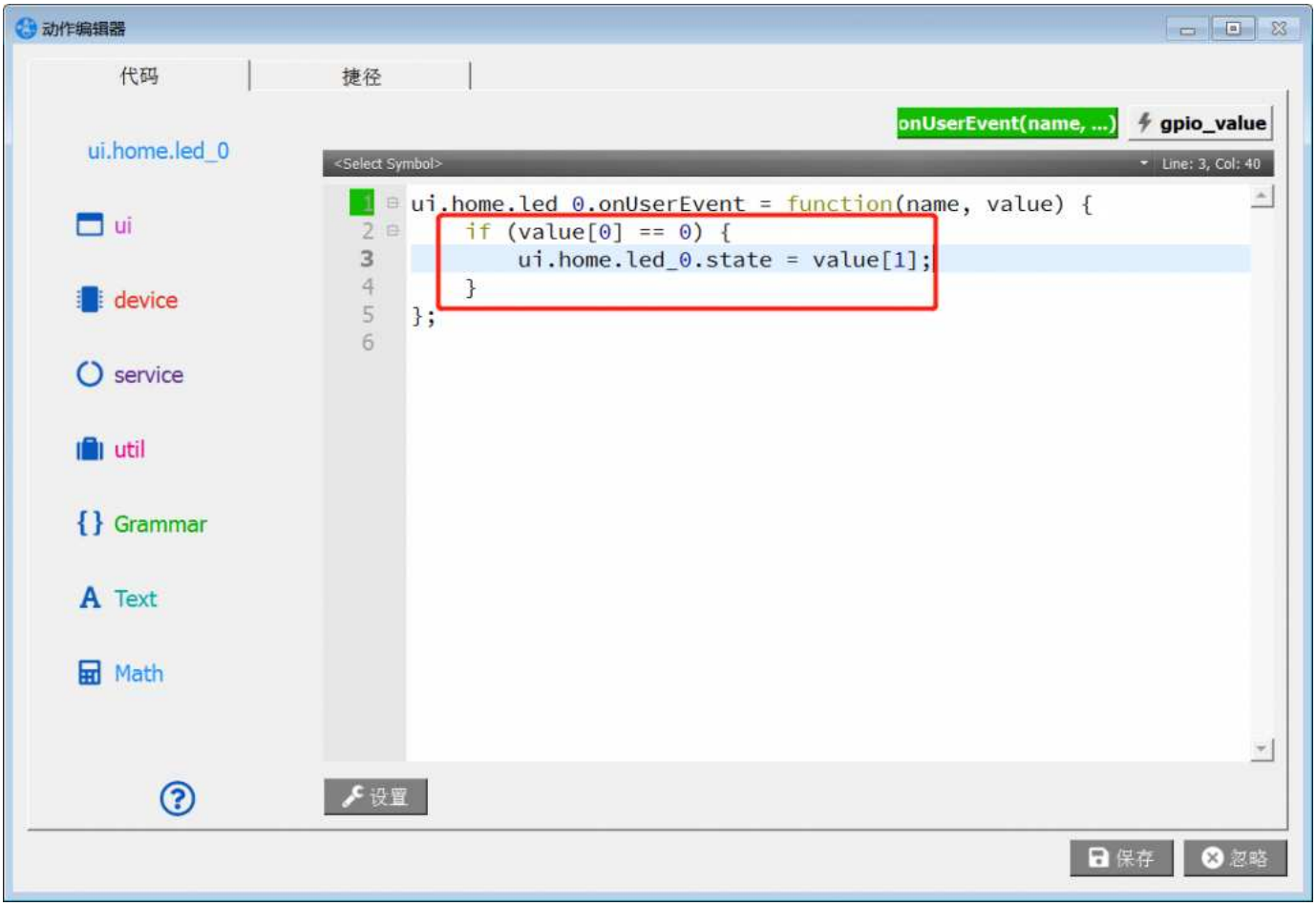
3、在主页面home上放置三个LED控件，选中led_0, 单击“代码编辑”



在动作编辑对话框的代码TAB里面，右上角点“事件管理”按钮，选择第1步添加的”gpio_value”事件，点击确定。



4、在onUserEvent方法中添加相应的代码。



5、重复上面的第3步，第4步，把剩余的led_01的led_02也添加上相应的代码。

6、构建并下载App到目标设备上，当GPIO的P0～P1有电平变化的时候，屏幕上的LED灯也会显示亮或者暗。

3.APP烧写

APP烧写

当APP设计，调试，验证完毕，Studio将生成.app文件，可下载/烧写到目标系统运行。下载方式可选择**USB**线或者**U**盘方式，二选一即可

USB线烧写

1. 将目标系统和PC通过USB线连接，然后上电，待启动完毕；
2. 在Studio中点击debugger启动调试下载器，点击…按钮，选择APP文件（扩展名为.app），点击Flash按钮烧写安装；
3. 待系统提示下载完成，5秒后APP即开始在目标系统上运行.

U盘烧写

1. 确保U盘只有一个分区， 将生成的APP文件拷贝到U盘下的根目录；
2. 上电启动目标系统，待启动完毕，将U盘插入USB接口,大约5秒后，自动进入界面更新程序，更新完毕后，按提示拔掉U盘，新APP即开始运行。

4.APP调试

■ 模拟器

在Studio中启动模拟器，PC上会运行一个模拟目标系统的虚拟设备(包括模拟显示屏，触摸屏，串口)。在无目标系统的情况下，也可验证APP界面效果，调试动作脚本逻辑功能等，加快开发调试过程。

模拟器上调试JavaScript脚本非常简单直观。APP在模拟器上运行时，如果当前运行的脚本出错，屏幕上会自动弹出错误提示框。根据提示的控件对象名称、错误类型、脚本行号等信息，用户可轻松找到脚本错误所在，一目了然。

注意：其中部分属于device, util和service的控件对象行为无法在模拟器中模拟，如蜂鸣器，环境变量，输入法等，与之相关的脚本和事件在模拟器中无效。

■ 目标系统调试

APP可分成两种版本：**debug**和**release**版，可通过Studio中的工具->配置->构建模式选定。当运行debug版本，脚本出错时，APP会自动弹出错误对话框提示，方便调试；对于Release版本，即使脚本出错，也不会弹出对话框，但是调试器中的调试日志中包含出错信息。建议在软件开发调试阶段，将APP构建成debug版本，待软件稳定后，构建成release版本烧写。

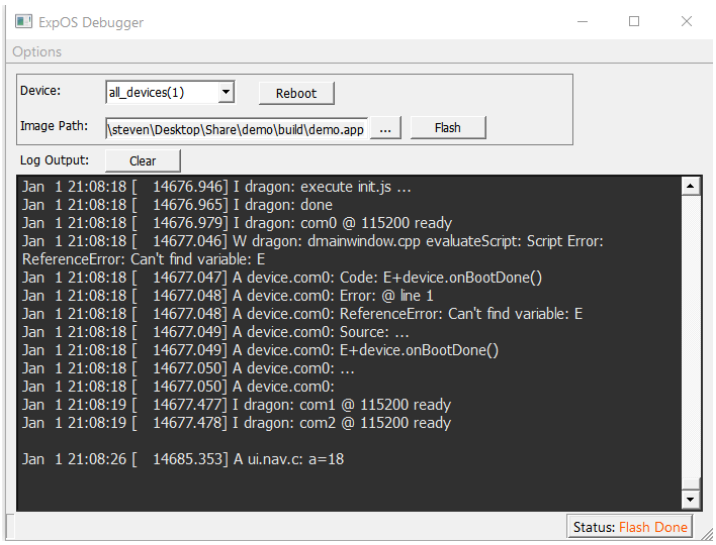
另外，调试阶段有时需要调试一些逻辑关系，变量值，跟踪事件等，可在脚本中调用util.console.log()方法，这样可在开发主机上通过调试器查看该方法输出的调试信息。

例如：在按钮控件的onRelease事件中输入如下脚本：

var a=18

util.consloe.log('a=' + a)

这样当APP在目标系统运行时，将在调试器中看到如下调试信息(最前面为系统时间戳，单位：秒, ui.nav.c为对象名，a=18即为脚本打印出的调试信息)：



5.APP OTA升级

OTA升级

ExpOS带以太网口的设备可以通过OTA的方式远程升级OS以及App

以两个不同版本的工程为例： 工程A为旧版本 ，工程B为新版本

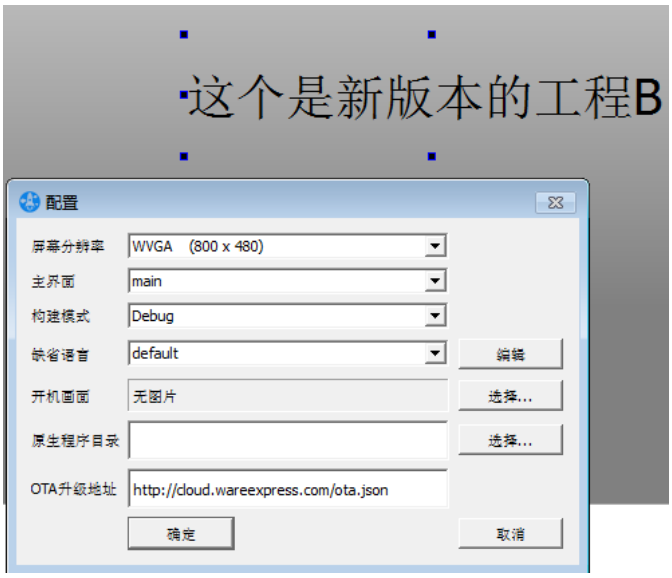
初始状态：断开网线

1、设置旧版本工程A的OTA升级地址

打开Studio，在配置页面中设置OTA升级地址为 http://cloud.wareexpress.com/ota.json，构建生成旧版本App并烧写至ExpOS设备。设备运行时，会自动查询这个地址检测是否有新版本APP。

2、生成新版本工程B

打开工程A，做些简单修改（例如改变界面某个控件），然后保存，同样设置新设置OTA升级地址为 http://cloud.wareexpress.com/ota.json，构建生成新版本App，不用烧写！



3、生成OTA配置文件

根据OS和新版本App的文件md5信息，本地编辑json格式文本文件，如ota.json

```
{
    "os_ver": "1.2.2",
    "interval": "1",
    "fw": {
        "md5": "7f9b63bb0ff596e8fb59c8607a729171",
        "url": "http://www.wareexpress.com/release/expos_V1.2.2_190706.fw"
    },
    "app": {
        "md5": "75ffba572872b7d7f5cd072ae2ad7479",
        "url": "http://cloud.wareexpress.com/ota_test_b.app"
    }
}
```

注意：json的内容都是字符串类型的，字段和值都需要双引号

os_ver 是OS固件的版本号，如 1.2.2

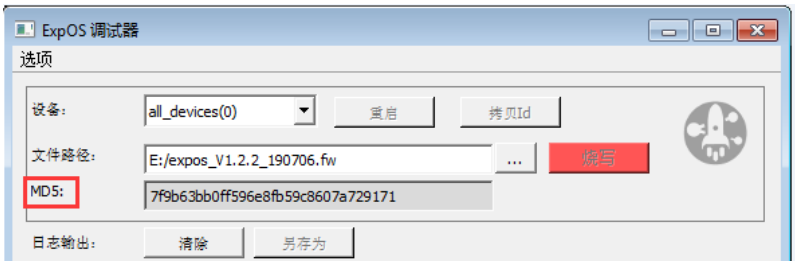
interval 是ExpOS设备上轮询检查是否有新版本的时间间隔，单位为分钟

fw 是OS固件的信息，包括md5和下载url

app 是App的信息，包括md5和下载url

文件md5信息获取：

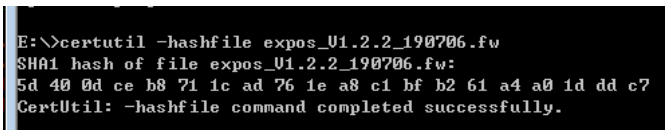
1.通过调试器获取



2.通过命令行获取

Windows命令行下执行，如 certutil -hashfile D:\expos_xxxx.fw MD5

Linux命令行下执行，如 md5sum ~/expos_xxxx.fw



4、上传OTA配置文件至OTA升级的服务器

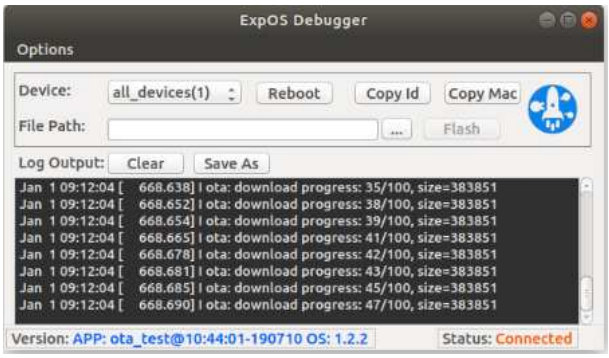
上传ota.json至web服务器目录，下载url如 http://cloud.wareexpress.com/ota.json

上传expos_xxxx.fw至服务器目录，下载url如 http://cloud.wareexpress.com/expos_xxxx.fw

上传ota_test_b.app至服务器目录，下载url如http://cloud.wareexpress.com/ota_test_b.app

5、观察OTA升级过程

打开调试器，ExpOS设备连接网线，观察调试器OTA升级日志，升级成功后ExpOS设备将显示工程B的界面



6.工程实例

6.1.工程实例-hellow-world

1) 启动Studio，新建工程hello，根据目标系统的情况，选择合适的配置;

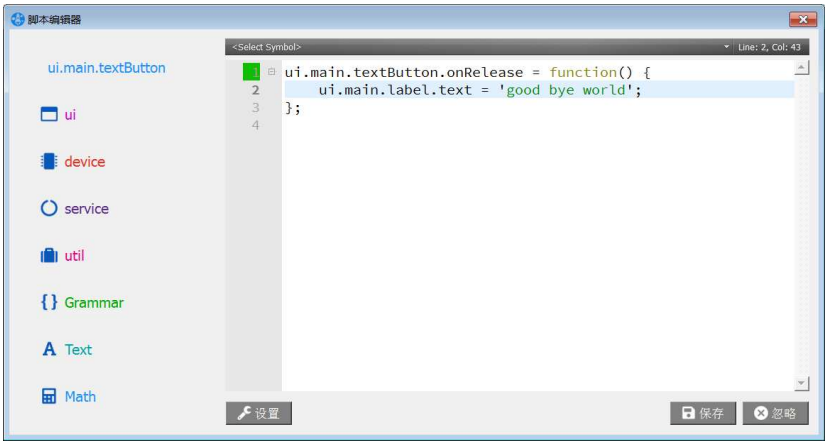
2) 从控件列表中拖入标签控件，并在对象列表区双击该控件，设定名称为label, 然后在属性区设定”文字”属性内容为”hellow world”, 并选择合适的文字大小；

3) 类似2)步，拖入文字按钮控件，设定名称为btn，然后设定”文字”属性内容为”press me”；

4) 选中按钮控件，点击动作列表区的onRelease按钮即可进入事件脚本编辑器，并在脚本编辑器的onRelease函数中输入如下脚本:

```
ui.main.label.text='good bye world'
```

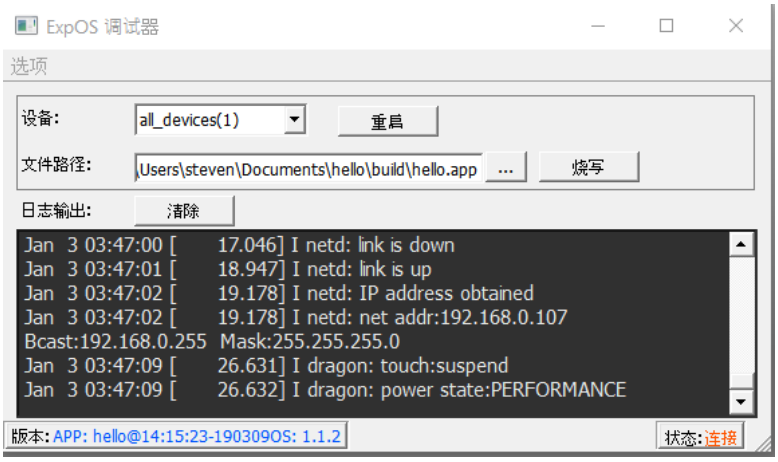
该脚本将在按钮被释放时自动执行改变标签文字内容为’good by world’.

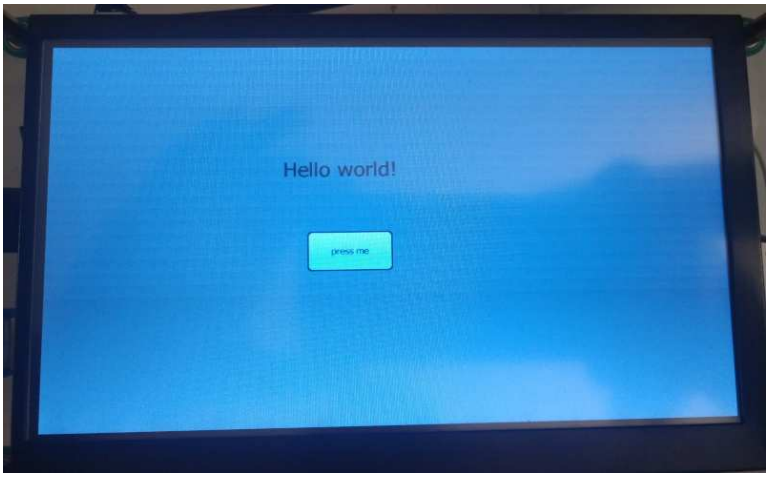


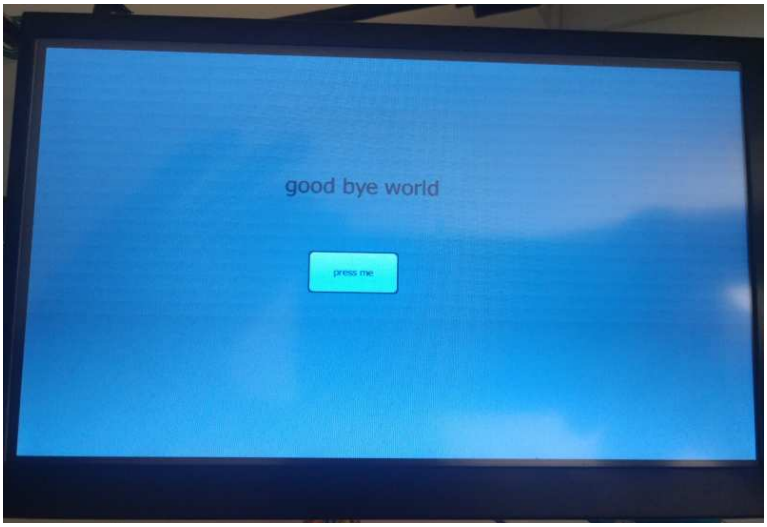
5) 点击菜单 工具->模拟器启动模拟器验证APP，打开模拟器后会看到上面设计的APP显示hello world;

6) 在模拟器中，鼠标点击按钮，这是会看到标签文字改变为good bye world;

7) 关闭模拟器，启动调试器就可以把设计的APP通过USB下载到目标系统运行，在目标系统上可以看到同样的执行结果。







至此，恭喜您已经掌握了Studio开发APP的基本方法和知识，可以继续开发你自己的更为复杂和强大的APP了！

6.2.工程实例-串口ScriptMode

本章以一个通过串口通讯的温控人机界面为例，详细说明基于ExpOS开发的通讯程序的全过程(以com0，ScriptMode通讯协议为例，串口的详细用法参看device速查表)。

■ 软件需求

– 用两个按钮来控制室内温度，按“+“， 温度上升（向串口0发送0x55，串口接收端为温度控制系统，收到0x55会升高温度），按”-”（向串口0发送0xAA，温度控制系统收到0xAA会降低温度）；

– 温度值由温度控制系统发送过来，需要仪表盘显示;

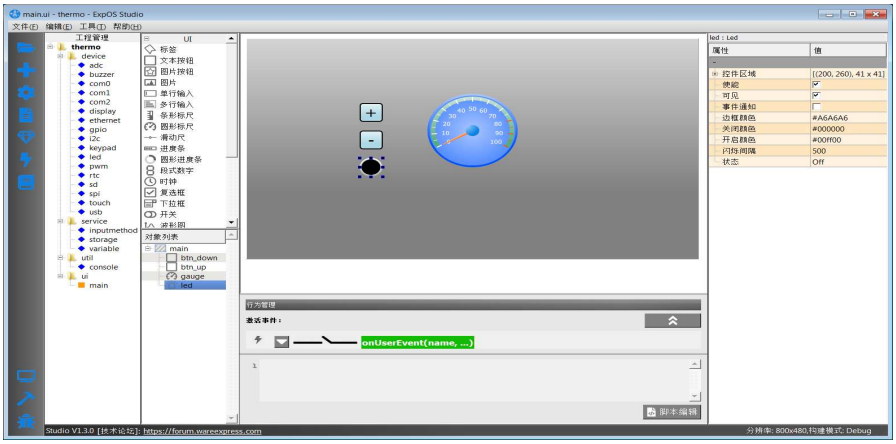
– 当温度超过30度，界面中的指示灯显示红色并闪烁；小于5度，显示蓝色并闪烁；介于5-30度之间，显示绿色.

■ 设计步骤

1) 启动Studio，新建工程thermo，根据目标系统的情况，选择合适的配置;

2) 从控件列表中拖入两个按钮控件，并在对象列表区双击该控件，设定名称btn_up和btn_down, 然后在属性区设定”文字”属性内容分别为”+”和”-“, 并选择合适的文字大小；

3) 类似2)步，拖入圆形标尺控件，设定名称为gauge，通过属性区设定各属性使外观达到自己需求，并拖拉尺寸到合适大小；类似方法拖入LED指示灯控件，并设定名称为led，设定属性区中 “关闭颜色” 为黑色和”开启颜色”为绿色。



4) 选中按钮+控件，点击动作列表区的onRelease按钮即可进入事件脚本编辑器，并在脚本编辑器的onRelease函数中输入如下脚本:

```
device.com0.write(0x55)
```

该脚本将在按钮+被释放时自动向串口0发送16进制数55.

然后选中按钮-, 在脚本编辑器中输入类似代码

```
device.com0.write(0xAA)
```

该脚本将在按钮-被释放时自动向串口0发送16进制数AA.

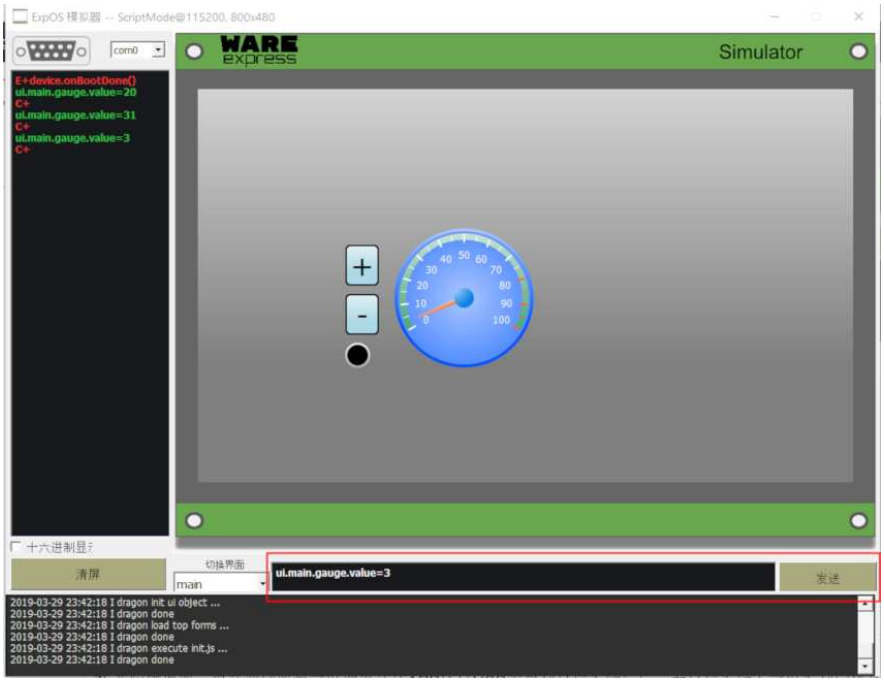
5) 点击菜单 工具->模拟器启动模拟器验证APP，打开模拟器后会看到上面设计的APP界面运行，选中左下角的”十六进制显示”单选框。用鼠标点击按钮+，可以看到左边的com0模拟结果会输出55，如果点击按钮-，输出aa。温度按钮控制按钮设计完毕，并工作。

6) 关闭模拟器，返回编辑状态，选中仪表盘，输入如下动作脚本：

```
function onValueChange(value) {
  if (ui.main.gauge.value > 30) {
    ui.main.led.state = 2;
    ui.main.led.onColor='#ff0000';
  } else if (ui.main.gauge.value < 5) {
    ui.main.led.state = 2;
    ui.main.led.onColor='#0000ff';
  } else {
    ui.main.led.state = 1;
    ui.main.led.onColor='#00ff00';
  }
}
```

以上代码将在仪表盘值变化时自动执行，根据不同的温度值，改变LED指示灯的状态。关于LED指示灯和仪表盘的属性用法，请参考软件速查。

7）再次启动模拟器，在串口发送模拟输入框内输入以下字符串”ui.main.gauge.value=3″，点”发送”按钮（模拟温度控制系统发送字符串到目标显示屏。由于com0默认为 scriptMode模式，所有收到的字符串按脚本执行），这时会看到LED变蓝并闪烁；如果输入”ui.main.gauge.value=31″并发送, LED变红并闪烁；如果输入”ui.main.gauge.value=20″并发送, LED变绿并常亮；6步的脚本执行完全达到设计要求。



8) 关闭模拟器，启动调试器就可以把设计的APP通过USB下载到目标系统运行，在目标系统上可以看到同样的执行结果。

这虽然只是一个简单的例子，但是里面包含了界面对象与界面对象，界面对象和设备对象device之间的关联编程，其实在实际的产品设计中大多数也是这种情况。所以，看完本例后，你完全可以开始实际项目的软件开发了。

6.3.工程实例-串口UserDefine

1、首先在设备控件列表选中com0, 然后在右边的属性面板中选择UserDefine协议，并使能和设置帧头“AA BB”和帧尾“CC DD”

2、在串口控件com0的动作脚本onReceive方法中读取串口数据。设置好了com0的帧头和帧尾，当串口com0收到数据并且数据中检测到有帧头和帧尾后，系统会自动调用onReceive方法。

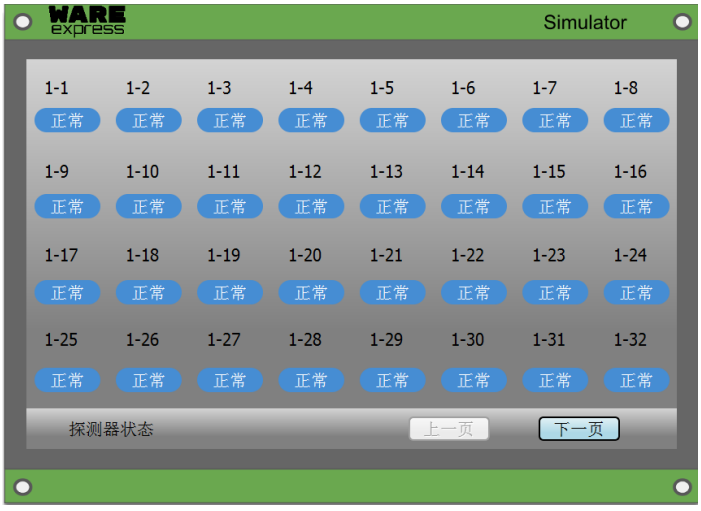
3、在主界面ui.main中，我们在“写串口”按钮的动作脚本中，从单行文本输入框ui.main.singleLineInput_write中获取所要发送的有效数据，拼接帧头，CRC校验以及帧尾，然后通过串口com0控件的write方法发送串口数据。

4、我们通过模拟器来模拟串口的发送和接收，点击屏端的”写串口“按钮，我们在下图中的左边栏能看到红色的一行是接收到屏端发过来的数据 ”AA BB 55 66 E3 2F CC DD“, 然后我们在模拟器底部的输入框中输入同样的数据，点右下角的”发送“按钮，屏端收到这个数据并读到总共5个字节 “55 66 EE A3”，CRC校验通过。

完整工程，请在最新的Studio中打开例程“UserDefine协议”

6.4.工程实例-485轮询

■ 软件需求



运行ExpOS的智能显示屏做主机，通过两个485串口(com1, com2)各自连接64台从机，总共128台从机。主机轮询从机的状态（报警/正常）并显示

串口通信协议如下：

主机巡查命令： 地址码 + 03 00 64 00 0D + 校验码（CRC-16 ModBus）

从机回应

02 03 1A 00 08 03 00 01 0A 03 00 01 0A 03 00 00 FD 03 00 00 F5 00 3C 00 3D 00 3E 00 3F 08 79

地址码 + 03 1A 00 08 +

第1路使能 + 第1路状态 + 第1路温度 +

第2路使能 + 第2路状态 + 第2路温度 +

第3路使能 + 第3路状态 + 第3路温度 +

第4路使能 + 第4路状态 + 第4路温度 +

00 + 第1路报警值 +

00 + 第2路报警值 +

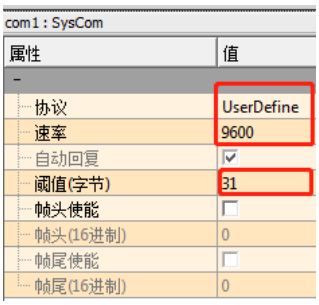
00 + 第3路报警值 +

00 + 第4路报警值 + 校验码

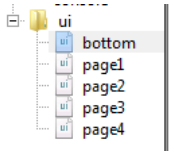
■ 实现步骤

打开Studio软件，新建一工程

1. 首先设置串口控件com1, com2的协议为UserDefine, 波特率为9600，阈值为31（从机回应的数据长度为31）



2. 删除主界面main, 添加4个界面(page1, page2, page3, page4)和一个底部导航界面(bottom)



3. 打开初始化脚本[工具菜单->初始化脚本]，添加公共方法：

```
// 定义巡检地址变量，地址从1开始到64，在此初始化为0，调用checkGroupNextAddr会自动加1
var groupAddr = [0, 0];

/**计算 CRC-16校验值
* @param data: byte数组
* @param len: CRC检验计算的长度
*/
function getCRC(data, len) {
  var j, k;
  var crc = 0xffff;
  k = len - 1;
  while(len--) {
    crc = crc ^ data[k - len];
    for(j=0;j<8;j++) {
      if(crc & 0x01 == 1) {
        crc = (crc >> 1) ^ 0xA001;
      }
      else crc = crc >> 1;
    }
  }
  var CRCH = (crc >> 8) & 0xff;
  var CRCL = crc%0x100;
  return [CRCL, CRCH];
}

/**获取主机巡检命令
* @param addr: 从机的地址码, 从地址1开始
*/
function getMasterCheckCmd(addr) {
  // 地址码 + 03 00 64 00 0D + 校验码(CRC-16)
  var cmd = [addr, 0x03, 0x00, 0x64, 0x00, 0x0D];
  // 计算CRC检验值 CRCL, CRCH
  var crc = getCRC(cmd, cmd.length);
  var len = cmd.length;
  cmd[len] = crc[0];
  cmd[len+1] = crc[1];
  return cmd;
};

/**解析从机状态
* @param data: 从串口接收到的从机返回数据(byte数组)
* @return: -1数据长度错误，-2数据格式错误，-3CRC校验错误，00正常，01报警，06断路，0A短路
*/
function parseSlaveState(data) {
  // 02 03 1A 00 08 03 00 01 0A 03 00 01 0A 03 00 00
  // FD 03 00 00 F5 00 3C 00 3D 00 3E 00 3F 08 79
  if (data.length != 31) {
    util.console.log('数据长度错误');
    return -2;
  }
  // 地址码 + 03 1A 00 08
  var magic = [0x03, 0x1A, 0x00, 0x08];
  for (var i=0; i<magic.length; i++) {
    if (data[i+1] != magic[i]) {
      util.console.log('数据格式错误');
```

```
      return -3;
    }
  }
  // 计算CRC
  var len = data.length;
  var crc = getCRC(data, len-2);
  if (crc[0] != data[len-2] || crc[1] != data[len-1]) {
    util.console.log('CRC校验失败');
    return -4;
  }
  var addr = data[0];
  // 第1路使能的起始索引
  var index = magic.length + 1;
  var s_normal = 0;
  var s_alarm = 0;
  var s_open = 0;
  var s_short = 0;
  for (var ch=1; ch<=4; ch++) { // 第1路使能控制(1byte) + 第1路状态(1byte) + 第1路温度(2bytes)
    //状态：00正常 · 01报警 · 06断路 · 0A短路
    var state = data[index+1];
    switch(state) {
      case 0x00: // 00正常
        s_normal++;
      break;
      case 0x01: // 01报警
        s_alarm++;
      break;
      case 0x06: // 06断路
        s_open++;
      break;
      case 0x0A: // 0A短路
        s_short++;
      break;
    }
    index += 4;
  }
  if (s_alarm > 0) { // 01报警
    return 0x01;
  } else if (s_short > 0) { // 0A短路
    return 0x0A;
  } else if (s_open > 0) { // 06断路
    return 0x06;
  } else if (s_normal == 4){ // 00正常
    return 0x00;
  }
}

/**巡检下一个地址
* @param index: 分组索引, 从0开始
*/
function checkGroupNextAddr(index)
{
  var group = index + 1;
  // 停止定时器
  eval('ui.bottom.timer_group'+group+'.stop();');
  // 巡检下一个地址
  if (++groupAddr[index] > 64) {
    groupAddr[index] = 1;
  }
  var data = getMasterCheckCmd(groupAddr[index]);
  util.console.log('master check: ' + group + '_' + groupAddr[index]);
  // 通过串口发送
  eval('device.com'+group+'.write(data);');
  // 启动定时器检查从机是否超时
  eval('ui.bottom.timer_group'+group+'.start();');
}
```

4. 在导航页面拖入两个定时器控件(分别命名为timer_group1, timer_group2)，设置定时器的间隔属性为100mS（从主机发送巡检命令到收到从机回应，平均时长为90mS）

在页面的onLoad方法中启动巡检

```
ui.bottom.onLoad = function() {
  // 巡检命令
  checkGroupNextAddr(0);
  checkGroupNextAddr(1);
};
```

5. 在定时器onTimeout方法中检查从机回应是否超时，以timer_group1为例

```
ui.bottom.timer_group1.onTimeout = function(counter) {
  // 从机返回超时
  var index = 0;
  updateState(index, groupAddr[index], -1);
  // 巡检下一个地址
  checkGroupNextAddr(index);
};
```

6. 在串口com1和com2的onReceive方法中处理接收的从机数据，以com1为例

```
device.com1.onReceive = function(count) {
  if (device.com1.readableBytes() == 31) {
    var index = 0;
    // 1.立即巡检下一个地址
```

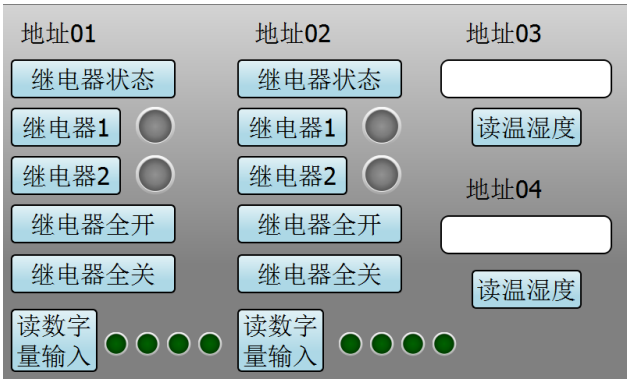
```
    checkGroupNextAddr(index);
    // 2.读取缓冲区数据，刷新界面状态
    var data = device.com1.read(31);
    var addr = data[0]; // 从机地址码
    var state = parseSlaveState(data);
    updateState(index, addr, state);
  }
};
```

完整工程，请在最新版本的Studio中打开例程“UserDefine自定义协议”

[注：本例程所用通信协议来自”沈阳宏宇光电子科技有限公司”，未经允许，不得用于商业用途]

6.5.工程实例-modbus-rtu

■ 软件需求



本例程以Modbus RTU协议为例，ExpOS设备做主机，通讯串口使用com1连接4台modbus RTU从设备(地址01～04)进行通讯，查询和控制从机的状态。

■ 设计步骤

1.首先设置串口com1协议为ModbusRTU，速率为9600，如果需要处理通讯超时，需要勾选“使能超时”属性，默认接收超时为100mS，可以根据具体情况调整

ExpOS设备目前支持的Modbus RTU功能码如下：

功能码01：读线圈（开关量输出）

功能码02：读输入状态（开关量输入）

功能码03：读保持寄存器（模拟量输出）

功能码04：读输入寄存器（模拟量输入）

功能码05：写单线圈（开关量输出）

功能码06：写单保持寄存器（模拟量输出）

功能码15：写多线圈（开关量输出）

功能码16：写多保持寄存器（模拟量输出）

功能码0F: 写多个线圈寄存器

功能码10: 写多个保持寄存器

2.主机发送Modbus指令

串口控件有专门的modbus写方法：device.com1.writeModbus(data) , 参数data可以是一个字符串（字符之间使用空格分隔），也可以是一个16进制的数组，也可以输入多个16进制做为参数

如：device.com1.writeModbus('01 01 00 00 00 02'); // 字符串

或者：device.com1.writeModbus([0x01, 0x01, 0x00, 0x00, 0x00, 0x02]); // 数组

或者：device.com1.writeModbus(0x01, 0x01, 0x00, 0x00, 0x00, 0x02); // 16进制

串口硬件输出的字节流为：01 01 00 00 00 02 BD CB

当然我们也可以不使用writeModbus方法，自己计算CRC，最后使用通用的write方法来写串口

如：var data = [0x01, 0x01, 0x00, 0x00, 0x00, 0x02];

```
var crc = util.calculateCRC16(data); // 返回长度为两个字节的数组
device.com1.write(data, crc);
```

串口硬件输出的字节流为：01 01 00 00 00 02 BD CB

3.主机接收Modbus从机回复

当第2步主机发送指令完成后，对应地址的从机回复，串口com1接收到一帧完整数据并CRC校验通过后，会调用串口控件的onReceive方法

如果串口控件设置勾选了"使能超时"属性，如果从机返回超时，或者从机返回数据帧校验有问题，会调用串口的onEvent方法

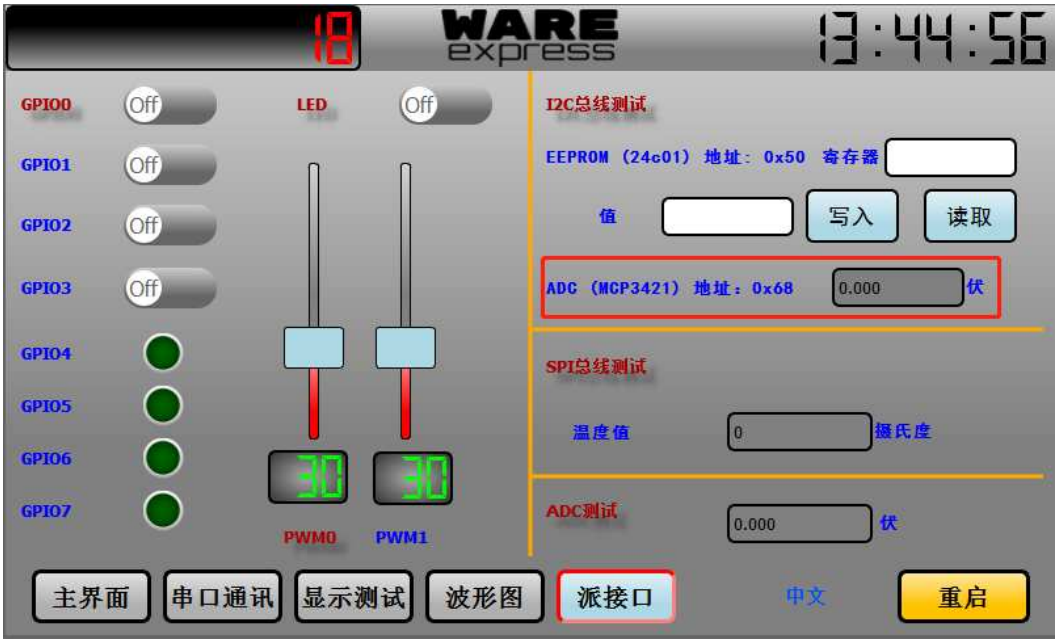
完整工程，请在最新版本的Studio中打开例程"Modbus RTU通讯"

6.6.工程实例-I2C读写MCP3421

MCP3421 为单通道低噪声、高精度、差分输入A/D转换器，分辨率高达18 位，提供微型SOT-23-6 封装。片上精密2.048V 参考电压使得差分输入电压范围为±2.048V （电压 = 4.096V ）。该器件使用2 线I2C兼容串行接口，并采用2.7V 至5.5V 单电源供电。用户通过2 线I2C 串行接口对控制配置位进行设定，该器件提供两种转换模式：a) 连续转换模式； b) 单次转换模式。在单次转换模式下，器件在完成一次转换后自动进入低电流待机模式，这样可显著降低空闲期间的电流消耗。(MCP3421中文数据手册)

本例程通过Express Pi扩展口I2C对ADC芯片MCP3421进行AD采集。

1、界面



2、根据MCP3421数据手册，我们准备配置为单次16位的数据采集模式，配置字节值为 0x88 (1000 1000)

寄存器 5-1：　　配置寄存器

R/W-1	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
$\overline{RDY}$	C1	C0	$\overline{O/C}$	S1	S0	G1	G0
1 *	0 *	0 *	1 *	0 *	0 *	0 *	0 *
bit 7							bit 0

* 上电复位时的缺省配置

图注：

R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 7　**$\overline{RDY}$：就绪标志位**

此位为数据就绪标志。在读模式，此位表示输出寄存器是否被新转换数据更新。在单次转换模式，向此位写入 1 将启动一次新的转换。

使用读命令读取 $\overline{RDY}$ 位：

1 = 输出寄存器未更新
0 = 输出寄存器被最新转换数据更新

使用写命令写 $\overline{RDY}$ 位：

连续转换模式：无影响

单次转换模式：
1 = 开始一次新的转换
0 = 无影响

bit 6-5　**C1-C0：通道选择位**

这些为通道选择位，但 MCP3421 器件未使用这些位。

bit 4　**$\overline{O/C}$：转换模式位**

1 = 连续转换模式。一旦该位被选定，器件将进行连续数据转换。
0 = 单次转换模式。器件进行单次转换并进入低功耗待机模式，直至收到新的读 / 写命令。

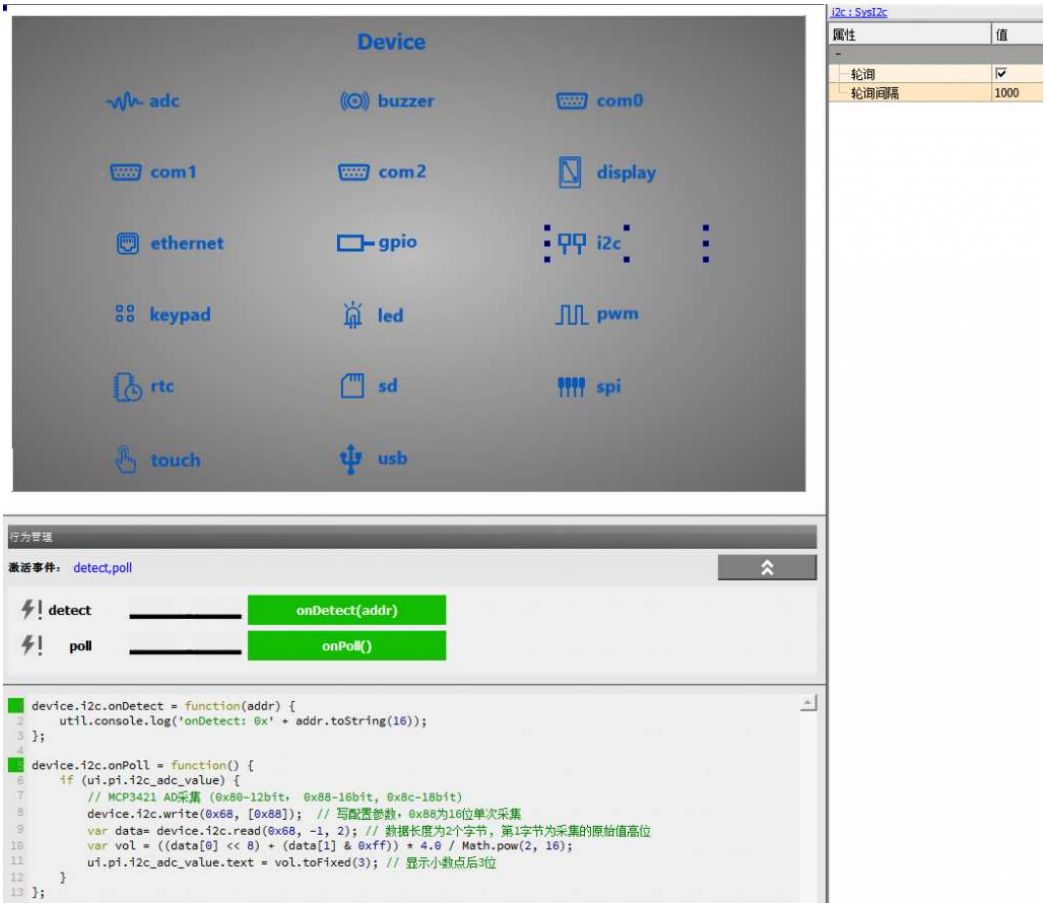
bit 3-2　**S1-S0：采样率选择位**

00 = 240 sps （12 位）
01 = 60 sps （14 位）
10 = 15 sps （16 位）
11 = 3.75 sps （18 位）

bit 1-0　**G1-G0：**PGA 增益选择位

00 = 1 V/V
01 = 2 V/V
10 = 4 V/V
11 = 8 V/V

3、系统控件device.i2c提供了轮询事件，我们使能轮询来进行读写I2C。MCP3421芯片的总线选择地址是出厂值 0xD0 (7位地址1101 000, 1位读写控制位), 在Express Pi上我们使用的是8位的地址，0xD0转换为8位的地址即为0x68 (**0**110 1000)，先写入MCP3421的配置字节，因为MCP3421芯片没有寄存器地址，所以只需要写入配置字节0x88。写完配置寄存器后，接着再读取2个字节的数据（该芯片没有寄存器地址，读数据时寄存器地址传-1），最后换算成实际的电压值(我们的测试板VIN-接的是地，所以电压值范围为0~4V)。



本实例完整工程，请在ExpOS Studio中打开例程中的“综合演示”

6.7.工程实例-I2C读写EEPROM

本工程以EEPROM 24C01芯片为例，演示I2C读写EEPROM的过程。

1、首先确定24C01的设备地址，通过查看24C01芯片手册，EEPROM 24C01的I2C设备选择地址为0xA0 (7位设备地址1010 000，1位读写控制位)，转换成8位地址0x50 (**0**101 0000)。调用device.i2c的写方法 device.i2c.write(addr, data), 参数addr为设备地址，data为字节数组，数组的第1个字节为寄存器地址0x10，第2个字节为内容如0x60

2、前面的写操作完成后，数据就存储到EEPROM相应的寄存器了，即使掉电也不会丢失。下面我们调用device.i2c的读方法来读取上面保存的内容。var data = device.i2c.read(addr, reg, count) 参数addr是设备地址，reg是寄存器地址，count是读取的字节长度。

本实例完整工程，请在ExpOS Studio中打开例程中的“综合演示”

6.8.工程实例-FTP文件传输

1、从Studio控件列表OP类别里面找到Ftp控件，拖入Ftp控件到当前界面，对象名称为ftp_2，设置相应的主机，用户，密码等属性，注意FTP端口号默认为21，一般情况下使用默认端口就行。

2、上传本地文件（本例使用File控件生成了一个本地的文件用来做上传测试），需要调用Ftp控件的put方法

ui.ftp.ftp_2.put(localPath, remotePath);

需要注意的是put方法不能在Ftp服务器端自动创建新的目录，所以远程的目录必须是一个已经存在的目录。第一个参数是本地文件路径是绝对路径，第二个参数远程路径是相对路径（相对于Ftp服务器配置的根路径），如本例中的：

本地路径：C:/tmp/EXPOSD15291A13E3E4A9.txt

远程路径：test/EXPOSD15291A13E3E4A9.txt

3、下载远程文件，需要用到Ftp控件的get方法

ui.ftp.ftp_2.get(remotePath, localPath);

跟上传刚好是相反的，第一个参数远程路径是个相对路径，第二个参数本地路径是个绝对路径

4、上传或者下载时任务是放在一个队列里面的，也就是先调用put/get方法的任务先执行，执行完一个任务后接着执行下一个任务，直到所有任务结束。如果想中止当前进行的任务以及后续还未开始的所有任务，可以使用abort方法，如：

ui.ftp.ftp_2.abort();

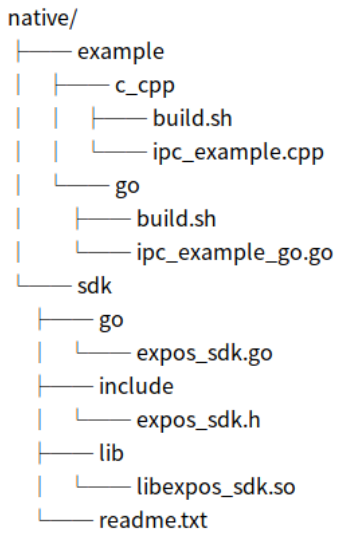
完整工程，请在Studio中打开例程”network网络“

6.9.工程实例-Native

Linux下**C/C++/GO**编写的原生程序与**JavaScript**界面程序通讯

ExpOS操作系统支持运行Linux下C/C++编写的标准原生程序，并且可与Studio生成的界面程序通讯，交换数据，方便拓展软件功能。C/C++原生程序与JavaScript程序的通讯通过SDK函数库实现，关于如何搭建编译环境和函数库说明，请参考 Linux原生程序SDK 。

下面以一个简单例子来说明通讯过程(该例程的源码位于最新的Studio安装目录，如C:\Program Files (x86)\ExpOS Studio\native， 目录结构如下：



– 编译和准备**Linux**原生程序

1.将整个native目录拷贝到Linux开发主机

export 编译工具链的bin路径，假如工具链的解压路径在 /opt/toolchain/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin

```
~/work$export EXPOS_TOOLCHAIN_PATH=/opt/toolchain/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

如果要编译C++的原生程序，请在终端命令行cd到 native/example/c_cpp目录，执行编译脚本，生成可执行文件 ipc_example_c

```
~/native/example/c_cpp$./build.sh
```

如果要编译Go的原生程序，请在终端命令行cd到 example/go 目录，执行编译脚本，生成可执行文件 ipc_example_go

```
~/native/example/go$./build.sh
```

2.拷贝原生可执行程序 ipc_example_c（或者ipc_example_go）到Studio所在的Windows环境

从Linux开发主机上将前面编译好的可执行文件 ipc_example_c（或者ipc_example_go）拷贝到Windows开发主机的工作目录备用, 如 D:\native\ipc_example

3. 在工作目录（例如上步的D:\native）新建一个名为 init.rc 的文本文件，该文件为原生程序的启动脚本，打开并编辑内容为(原生程序在后面步骤中打包到APP中时，固定拷贝到/app/native/目录)：

```
/app/native/ipc_example_c &
```

– 生成**App**

1) 在Windows主机上启动Studio，创建一个工程，如：命名为test_ipc

2) 从控件列表拖放一个通道控件(channel)到页面(main)中, 在Studio属性面板，修改“名称”属性的值为 sdk_demo

3) 编辑该通道控件动作脚本

```
ui.main.channel.onMessage = function(sender, msg, payload, responseId) {
    util.console.log(sender + ' : ' + msg); // 输出日志到调试器
    if (responseId)
        ui.main.channel.respond(sender, 0, ‘okay from JS’, ‘’);
}
```

4) 从控件列表拖放一个文本按钮控件(textButton)到页面(main)中, 编辑按钮的动作脚本:

```
ui.main.textButton.onRelease = function() {
    ui.main.channel.postMessage('ipc_example', 'this is from JS world', '');
}
```

5) 工具菜单->配置，在配置对话框中选择”原生程序目录”, 或者直接填写原生程序目录， 如 D:\native，该目录内容将在下步构建时打包到APP文件中

6) 工具菜单->构建，等待构建完成，点“下载应用”按钮，会启动调试器，在调试器窗口点击“烧写”, 不要关闭调试器，接着进行下面的测试

– 测试验证

1) 在调试器窗口的日志栏，观察日志输出

2) 在屏上点击按钮，观察日志输出

7.常见问题

7.1.常见问题-文本按钮高亮显示

在开发的过程中，我们不可避免地会遇到只让一组按钮中的一个按钮高亮显示，如导航按钮切换或者多个选项中选择一个，下面就给大家介绍一下怎么使用文本按钮的互斥高亮来实现这个功能。

所用到文本按钮的属性：

- highlightedTextColor: 高亮文字颜色
- highlightedBackgroundColor: 高亮背景颜色
- highlightedBorderColor: 高亮边框颜色
- highlightExclusive: 高亮互斥
- group: 分组

所用到文本按钮的方法：

- highlight(enabled)

1、选中左边一列的所有功能按钮，设置高亮颜色值，勾选上高亮互斥，并设置分组"func"。为什么要设置分组？因为我们使用高亮互斥功能是按照同一组名内互斥。

2、再选中右边一列的所有按钮，设置高亮颜色值，勾选上高亮互斥，并设置分组"ml"

3、在所有功能按钮的动作脚本中，调用highlight()方法设置自己高亮，同一分组其它按钮自动取消高亮

4、在所有毫升按钮的动作脚本中，调用highlight()方法设置自己高亮，同一分组其它按钮自动取消高亮

模拟运行，看下效果

7.2.常见问题-批量设置控件属性

如下图，页面main上有4个文本按钮(textButton_1, textButton_2, textButton_3, textButton_4)，我们想同时修改它们的文字以及文字颜色，我们除了一个接一个控件进行修改，能不能通过for循环的方式批量修改呢？

当然可以，有下面的两种方式来设置：

1、使用控件数组

```
ui.main.onLoad = function() {
var buttonArray = [ui.main.textButton_1, ui.main.textButton_2, ui.main.textButton_3, ui.main.textButton_4]; // 把控件对象放到一个数组里面
for (var i=0; i<4; i++) {
buttonArray[i].text = '测试-' + i;
buttonArray[i].textColor = '#ff0000';
}
};
```

2、使用eval方法 (eval是JavaScript中的一个全局方法，详情请点击)

```
ui.main.onLoad = function() {
for (var i=1; i<=4; i++) {
eval('ui.main.textButton_'+i+'.text = '测试-'+i+";');
eval('ui.main.textButton_'+i+'.textColor = '#ff0000';');
}
};
```

模拟运行，效果如下：

7.3.常见问题-如何实现掉电保存功能

有两种方式：
1、使用系统变量(service.variable)
如果只是想保存一些变量的值，可以使用系统变量的方式，数字，字符串，以及数组都可以保存
1.1 保存变量speed的值到系统变量
var speed = 100;
service.variable.write('myspeed', speed); // 注意: service.variable.write(name, value)第一个参数name是个字符串常量，不是变量! 我们可以随便起名，只要读取的时候传同样的名字就行

1.2 重新上电后读取断电前保存的系统变量
var speed = service.variable.read('myspeed');

2、通过读写内部存储器或者U盘上的文件
2.1 保存变量speed的内容到文件中 (/user 目录是ExpOS系统内部存储器的用户目录，我们可以在这个目录下读写文件操作)
先在界面ui.main中拖入一个File控件，设置路径为 '/user/test.txt'
```
var speed = 100;
if (ui.main.file.open()) { // 文件打开模式默认是可读可写
  ui.main.file.write(speed); // 写入1个字节
  ui.main.file.close();
}
```
2.2 重新上电后读取断电前保存的文件内容
```
var speed;
if (ui.main.file.open()) { // 文件打开模式默认是可读可写
  speed = ui.main.file.read(1); // 读取1个字节
  ui.main.file.close();
}
```

7.4.常见问题-多语言支持

1) 在设计界面时，通过Studio的配置菜单设定支持语言的数量，名称和默认语言；

2) 设计界面时，所有控件的text属性可设置多种语言标识(总数为1步设定的语言数), 以文本按钮控件为例，根据上步的设定，可设置两种标识：

3) 在运行时，由于1步选择的默认语言0，即英文，所以显示如下界面效果：

4) 如果执行脚本service.setLanguage(1)设定系统切换到语言1，即中文，界面立刻自动更新为如下效果:

5) 如果执行service.setLanguage(0), 界面自动切换到步骤3)效果。总之通过service.setLanguage()一行脚本即可轻松解决多语言场合的显示问题；

6) 在多语言环境下，text属性在不同语言环境下有不同的值，例如：当切换到语言0时，如果通过类似ui.form-name.widget-name.text = 'test'的脚本改变某个控件的text属性，只有语言0下的text的属性值发生变化，语言1环境下保持不变。以上步的Main按钮为例，执行完赋值操作后，语言0的标识从main变成了test，但是切换到语言1后，该按钮标识仍然为'主界面'。如果在语言0环境下设置语言1环境下的文字标识，可通过setText()方法, 类似：ui.form-name.widget-name.setText('测试', 1)

7.5.常见问题-数据类型转换

- 数组转换成字符串

如 var data = [0x61, 0x62, 0x63, 0x64]; 转换成字符串 'abcd'

ui.main.label.text = util.arrayToUtf8(data); // 使用util.arrayToUtf8方法

- 数字转换为16进制字符

如 var data = [0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff];将第2个元素0xcc显示成16进制字符'cc'到标签

ui.main.label.text = data[2].toString(16); // 使用JS方法 number.toString(radix)

- 数组转换为空格分隔的16进制字符串

如 var data = [0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff];将整个数组显示成16进制到标签

ui.main.label.text = util.arrayToHexString(data); // 使用util方法 util.arrayToHexString(array_name)

- 数组转换为浮点数

如 var data = [0x40, 0x48, 0xf5, 0xc3];将4个字节的数组转换成大端模式的浮点数 3.14

ui.main.label.text = util.arrayToFloat(data); // 使用util方法 util.arrayToFloat(array_name)

如 var data = [0xc3, 0xf5, 0x48, 0x40];将4个字节的数组转换成小端模式的浮点数 3.14

ui.main.label.text = util.arrayToFloat(data, false); // 使用util方法 util.arrayToFloat(array_name, isBigEndian)

- 浮点数只显示N位小数

如 var value = 3.1415926; 只保留两位小数显示

ui.main.label.text = value.toFixed(2); // 使用JS方法 number.toFixed(num)

- 浮点数转换成整数

如 var value = 3.1415926; 转换成整数 3

ui.main.label.text = parseInt(value);

- 字符与ASCII码的转换

将字符转为ASCII码

var str = 'a';

var code = ch.charCodeAt(); // 97

将ASCII码转为字符

var str = String.fromCharCode(97); // 'a'

8.软件速查表

8.1.用户界面-ui

8.1.1.页面-form

描述

页面，是其子界面控件对象的容器，父对象为ui，子对象为界面中的控件

事件

---

onLoad： 加载，当切换到非缓存页面（该页面属性cached=0）时触发。

onRaise：升载，当切换到缓存页面时（该页面属性cached=1）触发，第一次显示缓存页面时，会触发一次onLoad事件。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，所有子界面控件处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.enabled=0禁止form中的所有子控件

geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
backgroundImage	背景图片源	ui.form-name.backgroundImage='test.png'设置背景为资源图片test.png
alwaysOnTop	置顶：页面如果可见，永远显示在最上层， 此属性为脚本只读	var a=ui.form-name.alwaysOnTop读取alwaysOnTop属性并存入变量a
cached	隐藏时缓存：页面隐藏时，任然缓存在内存中，在后台运行。否则页面及其所有子控件释放，下次显示该页面时重新初始化所有子控件，该属性默认为0，即不缓存页面。此属性为脚本只读	var a=ui.form-name.cached读取cached属性并存入变量a
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.form-name.verbose=1设置form触发条件满足时向device.com0发送事件消息
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.form-name.backgroundColor='#FF0000'设置背景颜色为红色
backgroundColorType	背景色类型 0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.backgroundColorType=4设置form的背景颜色类型为Circle
backgroundColorAlpha	背景色透明度：0-255. 0 为不透明，255为全透明，如：128为半透明	ui.form-name.backgroundColorAlpha=128设置form的背景颜色为半透明

方法

名称	功能说明	脚本例子
preload()	预加载页面，但是不显示，脚本这时可访问该页面的子对象	ui.form-name.preload()预加载页面到内存中
show()	显示form: 可支持无参数或者一个参数，参数为切换到该form时的动态效果。参数包括"drop", "slide", "scale"和"fade"。	ui.form-name.show()按默认效果显示form，默认效果是"fade"ui.form-name.show('drop')按"drop"效果显示form
hide()	隐藏form： 无参数。如果该页面是缓存页面，即使隐藏，仍存在于内存中，脚本仍然可以访问该对象及子对象。如果为非缓存页面，该页面将从内存中释放，脚本无法访问。	ui.form-name.hide()隐藏form，同时显示其他缓存的后台form
move(x,y)	移动form：将form原点移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.move(10,10)移动form到坐标（10，10）

8.1.2.对话框-dialog

描述

弹出式界面，是其子界面控件对象的容器，父对象为ui，子对象为界面中的控件，始终处于最上层。

事件

---

onLoad： 加载，当切换到非缓存对话框（属性cached=0）时触发。

onRaise：升载，当切换到缓存对话框时（属性cached=1）触发，启动时缓存对话框切换会触发一次onLoad事件。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，所有子控件处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.dlg-name.enabled=0 禁止对话框及所有子控件
geometry	几何尺寸：定义对话框长和宽	无
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.dlg-name.verbose=1 设置对话框触发条件满足时向device.com0发送事件消息
backgroundImage	背景图片源：设置背景图片	ui.dlg-name.backgroundImage='test.png' 设置背景为资源图片test.png
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.dlg-name.backgroundColor='#00ff00' 设置背景颜色为绿色

backgroundColorType	背景色类型： 0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.dlg-name.backgroundColorType=4 设置背景颜色类型为Circle
backgroundColorAlpha	背景色不透明度：0-255. 0 为全透明，255为不透明，如：128为半透明	ui.dlg-name.backgroundColorAlpha=128 设置背景色为半透明
overlapColor	弹出对话框时，后台界面重叠颜色，颜色定义参看 控件对象	ui.dlg-name.overlapColor='#00ff00′ 设置重叠颜色为绿色
overlapColorAlpha	重叠色不透明度：0-255. 0 为全透明，255为不透明，如：128为半透明	ui.dlg-name.overlapColorAlpha=128 设置重叠色为半透明
cached	隐藏时缓存：对话框隐藏时，缓存在内存中，后台运行。否则对话框及其所有子控件释放，下次显示该对话框时重新初始化所有子控件，该属性默认为0，即不缓存	ui.dlg-name.cached=1 设置对话框隐藏时在后台继续运行

方法

名称	功能说明	脚本例子
preload()	预加载对话框，但是不显示，脚本这时可访问该对话框的子对象	ui.dlg-name.preload() 预加载对话框到内存中
show()	显示: 支持无参数或者一个参数，参数为切换到该form时的动态效果。 参数包括"drop", "slide", "scale"和"fade"。	ui.dlg-name.show() 按默认效果显示对话框，默认效果是"fade" ui.dlg-name.show('drop') 按"drop"效果显示form
hide()	隐藏： 无参数。	ui.dlg-name.hide() 隐藏对话框，同时显示其他缓存的后台form
move(x,y)	移动：将对话框原点移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.dlg-name.move(10,10) 移动对话框到坐标（10，10）

8.1.3.图标-FontIcon

描述

图标，父对象为ui，无子对象

事件

onPress： 按下，按下时触发。

onRelease：释放，松开时触发。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.fonticon-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.fonticon-name.visible=0 隐藏对象
text	文字标识	ui.form-name.fonticon-name.text='test' 设置对象的当前语言环境下的文字标识为test
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.fonticon-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
textColor	文字颜色，颜色定义参看 控件对象	ui.form-name.fonticon-name.textColor='#ff0000′ 设置文字为红色
textAlign	文字对齐方式。0-Right，1-Bottom。	ui.form-name.fonticon-name.textAlign=0 设置对象文本在图标的右边
textMargin	文本边距：文字距离图标的距离，单位：像素	ui.form-name.fonticon-name.textMargin=2 设置文本边距为2个像素
borderColor	边框颜色,颜色定义参看 控件对象	ui.form-name.fonticon-name.borderColor='#ff0000′ 设置边框为红色
borderWidth	边框宽度，单位：像素	ui.form-name.fonticon-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.fonticon-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型：0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。。	ui.form-name.fonticon-name.borderType=4 设置对象无边框

backgroundColor	背景色：设置form的背景颜色	无
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.fonticon-name.backgroundColorType=4 设置form的背景颜色类型为Circle
effect	显示效果：0-None, 1-Shadow, 2-Blur	ui.form-name.fonticon-name.effect=1 显示效果为Shadow
blurRadius	模糊半径	ui.form-name.fonticon-name.blurRadius=2 设置模糊半径为2
source	源图片：设置源图片，内容为图片对应的ID	ui.form-name.fonticon-name.source='80b6e1' 设置源图片
sourceColor	源颜色：设置图标的颜色	ui.form-name.fonticon-name.sourceColor='#ff0000' 设置源图片颜色为红色
size	尺寸：设置图标的大小，单位：像素	ui.form-name.fonticon-name.size=40 设置源图片尺寸为40像素

方法

名称	功能说明	脚本例子
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.label-name.setText('测试', 1) 设置语言环境1下的文字标识为'测试'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
append(text)	追加字符串，参数text为字符串	ui.form-name.label-name.append('A') 在最后面追加一个字符'A'
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.label-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.label-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.4.标签-label

描述

文字标签，父对象为ui，无子对象

事件

onTextChange：文本内容改变时触发， 函数定义：onTextChange(text)， text表示当前文本字符串

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.label-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.label-name.visible=0 隐藏对象
text	文字标识	ui.form-name.label-name.text='test' 设置对象的当前语言环境下的文字标识为test
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.label-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
textColor	文字颜色，颜色定义参看 控件对象	ui.form-name.label-name.textColor='#ff0000' 设置文字为红色
textAlign	文字对齐方式。0-Center, 1-Left，2-Right，3-Top，4-Bottom。	ui.form-name.label-name.textAlign=0 设置对象文本居中对齐
borderColor	边框颜色,颜色定义参看 控件对象	ui.form-name.label-name.borderColor='#ff0000' 设置边框为红色
borderWidth	边框宽度，单位：像素	ui.form-name.label-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.label-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型：0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。。	ui.form-name.label-name.borderType=4 设置对象无边框

backgroundColor	背景色：设置form的背景颜色	无
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.label-name.backgroundColorType=4 设置form的背景颜色类型为Circle
effect	显示效果：0-None, 1-Shadow, 2-Blur	ui.form-name.label-name.effect=1 显示效果为Shadow
blurRadius	模糊半径	ui.form-name.label-name.blurRadius=2 设置模糊半径为2

方法

名称	功能说明	脚本例子
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.label-name.setText('测试', 1) 设置语言环境1下的文字标识为'测试'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
backspace()	回退操作，删除最后一个字符	ui.form-name.label-name.backspace() 删除最后一个字符
append(text)	追加字符串，参数text为字符串	ui.form-name.label-name.append('A') 在最后面追加一个字符'A'
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.label-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.label-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.5.文本按钮-textbutton

描述

文字标识的按钮，父对象为ui，无子对象

支持按下(onPress)，抬起（onRelease）和保持（onHold）事件

事件

onPress： 按下，按下按钮时触发。

onRelease：释放，松开按钮时触发。

onHold：保持，按住按钮时触发，只适用于非锁定按钮（属性checkable=false）。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.textbutton-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.textbutton-name.visible=0 隐藏对象
text	按钮文字标识	ui.form-name.textbutton-name.text="test" 设置对象当前语言环境下的文字标识为'test'
icon	图标源文件	ui.form-name.textbutton-name.icon='icon.jpg' 设置图标为资源图片icon.jpg
iconSize	图标分辨率	无
autoRepeat	自动重复onHold事件。如果选中，当按钮被按下并保持时， 自动产生onHold事件，间隔时间由autoRepeatInterval指定.	ui.form-name.textbutton-name.autoRepeat=1 设置对象支持产生onHold事件
autoRepeatInterval	自动重复onHold的时间间隔，单位：ms。 只有autoRepeat选中，该属性才有效.	ui.form-name.textbutton-name.autoRepeatInterval=500 设置对象500ms产生一次onHold事件
checkable	支持状态锁定。如果选中，每按一次按钮，按钮状态只变一次。如：当按钮处于抬起状态时，按一次按钮，按钮处于按下状态并一直锁定在该状态，直到再按一次该按钮，按钮解除锁定，回到抬起状态。	ui.form-name.textbutton-name.checkable=1 设置对象支持状态锁定
checked		

	按钮的锁定状态。1：按钮处于按下和锁定状态 0：按钮处于抬起状态	ui.form-name.textbutton-name.checked=1 设置对象处于按下和锁定状态
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.form-name.textbutton-name.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
textColor	文字颜色, 颜色定义参看 控件对象	ui.form-name.textbutton-name.textColor='#ff0000′ 设置文字颜色为红色
textAlign	文字对齐方式: 0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.textbutton-name.textAlign=0 设置对象文本居中对齐
borderColor	边框颜色，颜色定义参看 控件对象	ui.form-name.textbutton-name.borderColor='#ff0000′ 设置边框颜色为红色
borderWidth	边框宽度，单位：像素	ui.form-name.textbutton-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.textbutton-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型: 0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.textbutton-name.borderType=4 设置对象无边框
backgroundColor	背景色，颜色定义参看 控件对象	ui.form-name.textbutton-name.backgroundColor='#0000ff' 设置背景色为蓝色
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.textbutton-name.backgroundColorType=4 设置form的背景颜色类型为Circle
highlightedTextColor	高亮文字颜色	ui.form-name.textbutton-name.highlightedTextColor='#ffffff' 设置高亮时文字颜色为白色
highlightedBackgroundColor	高亮背景颜色	ui.form-name.textbutton-name.highlightedBackgroundColor='#0000ff' 设置高亮时背景颜色为蓝色
highlightedBorderColor	高亮边框颜色	ui.form-name.textbutton-name.highlightedBorderColor='#ff0000′ 设置高亮时边框颜色为红色
highlightExclusive	高亮互斥，勾选表示同一分组只能有一个按钮高亮显示	ui.form-name.textbutton-name.highlightExclusive=true 设置高亮互斥
group	分组	ui.form-name.textbutton-name.group='row1′ 设置分组名为row1

方法

名称	功能说明	脚本例子
simulateTouch(state)	软件模拟触摸动作，state表示触摸的动作状态：0-release(抬起)，1-Press（按下）	ui.form-name.textbutton-name.simulateTouch(1) 模拟触摸按下按钮
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.textbutton-name.setText('测试', 1) 设置语言环境1下的文字标识为'测试'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
backspace()	回退操作，删除最后一个字符	ui.form-name.textbutton-name.backspace() 删除最后一个字符
append(text)	追加字符串，参数text为字符串	ui.form-name.textbutton-name.append('A') 在最后面追加一个字符'A'
highlight(enabled)	设置按钮高亮显示，如果设置了高亮互斥，同一分组只能有一个按钮高亮	ui.form-name.textbutton-name.highlight(true) 按钮高亮
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角	ui.form-name.textbutton-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.textbutton-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.6.图片按钮-imagebutton

描述

图片标识的按钮，无边框，可定制按下和抬起时的图片，父对象为ui，无子对象

事件

onPress： 按下，按下按钮时触发。

onRelease：释放，松开按钮时触发。

onHold：保持，按住按钮时触发，只适用于非锁定按钮（属性checkable=false）。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.imagebutton-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.imagebutton-name.visible=0 隐藏对象
releasedSource	抬起状态时的图片源文件	ui.form-name.imagebutton-name.releasedSource=’test.png’ 设置抬起时显示资源图片test.png
pressedSource	按下和保持状态时的图片源文件	ui.form-name.imagebutton-name.presssedSource=’test.png’ 设置按下时显示资源图片test.png
autoRepeat	自动重复onHold事件。如果选中，当按钮被按下并保持时， 自动产生onHold事件，间隔时间由autoRepeatInterval指定.	ui.form-name.imagebutton-name.autoRepeat=1 设置对象支持产生onHold事件
autoRepeatInterval	自动重复onHold的时间间隔，单位：ms。 只有autoRepeat选中，该属性才有效.	ui.form-name.imagebutton-name.autoRepeatInterval=500 设置对象500ms产生一次onHold事件
checkable	支持状态锁定。如果选中，每按一次按钮，按钮状态只变一次。如：当按钮处于抬起状态时，按一次按钮，按钮处于按下状态并一直锁定在该状态，直到再按一次该按钮，按钮解除锁定，回到抬起状态。	ui.form-name.imagebutton-name.checkable=1 设置对象支持状态锁定
checked	按钮的锁定状态。1：按钮处于按下和锁定状态 0：按钮处于抬起状态	ui.form-name.imagebutton-name.checked=1 设置对象处于按下和锁定状态
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.form-name.imagebutton-name.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息

方法

名称	功能说明	脚本例子
simulateTouch(state)	软件模拟触摸动作，state表示触摸的动作状态：0-release(抬起)，1-Press（按下）	ui.form-name.imagebutton-name.simulateTouch(1) 模拟触摸按下按钮
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.imagebutton-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.imagebutton-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.7.图片-image

描述

图片，支持静态图片格式bmp，jpg，png等，动态图片支持gif，默认gif动画是停止的，只有调用play方法后才开始播放。父对象为ui，无子对象

事件

onSourceChange： 设置新图片源时触发，事件函数：onSourceChange(source)， source为当前图片源名称。

onFrameChange：帧改变时触发，只适用于gif动画图片， 事件函数：onFrameChange(frameNumber)， frameNumber为当前帧序号

onLoopDone：循环完所有帧时触发，只适用于gif动画图片，事件函数：onLoopDone()。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.image-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.image-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.image-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.form-name.image-name.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
stretched	图片扩展：如果为true，图片会拉伸扩展至整个对象尺寸	ui.form-name.image-name.stretched=1 设置图片扩展至整个对象尺寸
source	图片源	ui.form-name.image-name.source=’test.png’ 设置图片源为资源图片test.png
speedRatio	动画播放速度比：用于设置播放速度，值为原始播放速度的倍数，只适用于gif图片	ui.form-name.image-name.speedRatio=1.5 设置动画播放速度为原始速度的1.5倍

方法

名称	功能说明	脚本例子
play()	播放动画：可带一个参数，标识播放次数（一个循环为一次）	ui.form-name.image-name.play(2) 播放两次动画 ui.form-name.image-name.play() 播放动画，直到调用pause()或者stop()
stop()	停止动画，再次播放时，动画从第一帧开始	ui.form-name.image-name.stop() 停止动画播放
pause()	暂停动画，恢复播放时，从暂停时帧继续	ui.form-name.image-name.pause() 暂停动画播放
resume()	恢复动画播放，从暂停时帧继续	ui.form-name.image-name.resume() 恢复动画播放
frameNumber()	当前播放动画帧序号	num = ui.form-name.image-name.frameNumber() 读取当前播放动画帧序号存入变量num
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.image-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.image-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.8.单行输入框-singlelineinput

描述

单行文本输入框，父对象为ui，无子对象

事件

onTextChange： 文本框文本发生变化时触发。

onEditFinish：当按下输入键盘OK按钮时触发。

onEditCancel：当按下输入键盘Cancel按钮时触发。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于’禁止”状态，对所有触摸操作无反应。	ui.form-name.singlelineinput-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.singlelineinput-name.visible=0 隐藏对象
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.singlelineinput-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
text	文本内容	ui.form-name.singlelineinput-name.text=”test” 设置对

		象的当前语言环境下的文本内容为’test’
maxLength	文字最大输入长度（默认32767）	ui.form-name.maxLength=5 设置最大输入长度为5
echoMode	回显模式， 值为0-Normal， 1-NoEcho，2-Password，3-PasswordEchoOnEdit	ui.form-name.singlelineinput-name.echoMode=2 设置对象回显模式为密码（星号*代替实际的字符）
cursorPosition	输入光标位置	ui.form-name.singlelineinput-name.cursorPosition=6 设置输入光标在第6个字符位置
readOnly	只读，如果为true，对象只能显示文本，无法输入	ui.form-name.singlelineinput-name.readOnly=1 设置对象为只读
numberOnly	只能输入数字，获取焦点后软键盘自动切换到数字模式	ui.form-name.singleinput-name.numberOnly=1 设置只能输入数字
validator	验证器，值为正则表达式字符串	ui.form-name.singleinput-name.validator='[0-9]+$’ 设置验证器，只能输入数字
textColor	文字颜色, 颜色定义参看 控件对象	ui.form-name.singlelineinput-name.textColor=’#ff0000′ 设置文字为红色
textAlign	文字对齐方式：	ui.form-name.singlelineinput-name.textAlign=’center’ 设置对象文本居中对齐
borderColor	边框颜色，颜色定义参看 控件对象	ui.form-name.singleinput-name.borderColor=’#0000ff’ 设置边框为蓝色
borderWidth	边框宽度，单位：像素	ui.form-name.singlelineinput-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.singlelineinput-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型。支持0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.singlelineinput-name.borderType=4 设置对象无边框
backgroundColor	背景色，颜色定义参看 控件对象	ui.form-name.singlelineinput-name.backgroundColor=’#ff0000′ 设置背景为红色
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle，5-Pure, 6-No_Color	ui.form-name.singlelineinput-name.backgroundColorType=4 设置form的背景颜色类型为Circle
readOnlyBackgroundColor	只读时背景色	无
editingBackgroundColor	编辑时背景色	无
defaultInputMethod	默认输入法：0-Letter， 1-Number	ui.form-name.singlelineinput-name.defaultInputMethod=1 设置默认输入法为数字键盘

方法

名称	功能说明	脚本例子
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.singlelineinput-name.setText(‘测试’, 1) 设置语言环境1下的文字标识为’测试’，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
backspace()	回退操作，删除最后一个字符	ui.form-name.singlelineinput-name.backspace() 删除最后一个字符
append(text)	追加字符串，参数text为字符串	ui.form-name.singlelineinput-name.append(‘A’) 在最后面追加一个字符’A’
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.singlelineinput-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.singlelineinput-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.9.多行输入框-multilineinput

描述

多行文本输入框，父对象为ui，无子对象

事件

onTextChange： 文本框文本发生变化时触发。

onEditFinish：当按下输入键盘OK按钮时触发。

onEditCancel：当按下输入键盘Cancel按钮时触发。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.multilineinput-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.multilineinput-name.visible=0 隐藏对象
text	文本内容	ui.form-name.multilineinput-name.text=’test’ 设置对象当前语言环境下的文本内容为test
autoScroll	如果输入文本超出输入框大小自动增加滑动条	ui.form-name.multilineinput-name.autoScroll=1 设置对象自动增加滑动条
readOnly	只读，如果为true，对象只能显示文本，无法输入	ui.form-name.multilineinput-name.readOnly=1 设置对象为只读
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.multilineinput-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
textColor	文字颜色, 颜色定义参看 控件对象	ui.form-name.multilineinput-name.textColor=’#ff0000′ 设置文字颜色为红色
borderColor	边框颜色，颜色定义参看 控件对象	ui.form-name.multilineinput-name.borderColor=’#00ff00′ 设置边框为绿色
borderWidth	边框宽度，单位：像素	ui.form-name.multilineinput-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.multilineinput-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型。支持0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.multilineinput-name.borderType=4 设置对象无边框
backgroundColor	背景色，颜色定义参看 控件对象	ui.form-name.multilineinput.backgroundColor=’#ff0000′ 设置背景为红色
backgroundColorType	背景色类型：linear-1,linear-2,radial-1,radial-2, circle，pure, none可选	ui.form-name.multilineinput-name.backgroundColorType=’circle’ 设置对象背景颜色类型为circle
readOnlyBackgroundColor	只读时背景色，颜色定义参看 控件对象	ui.form-name.multlineinput-name.readOnlyBackgroundColor=’#ff0000′ 设置输入框只读时背景色为红色
editingBackgroundColor	编辑时背景色，颜色定义参看 控件对象	ui.form-name.multilineinput-name.editingBackgroundColor=’#00ff00′ 设置输入框编辑时背景色为绿色
scrollBarColor	滑动条颜色，颜色定义参看 控件对象	ui.form-name.multilineinput-name.scrollBarColor=’#ff0000′ 设置滚动条颜色为红色
scrollBarBackgroundColor	滑动条背景颜色，颜色定义参看 控件对象	ui.form-name.multilineinput-name.scrollBarBackgroundColor=’#ff0000′ 设置滚动条背景为红色
scrollBarBackgroundColorType	滑动条背景颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.multilineinput-name.scrollBarbackgroundColorType=4 设置对象滑动条背景颜色类型为Circle
scrollBarWidth	滑动条宽度，单位：像素	ui.form-name.multilineinput-name.scrollBarWidth=20 设置对象滑动条宽度为20个像素
defaultInputMethod	默认输入法， 0-Letter，1-Number	ui.form-name.multilineinput-name.defaultInputMethod=1 设置默认输入法为数字键盘

方法

名称	功能说明	脚本例子
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.multilineinput-name.setText(‘测试’, 1) 设置语言环境1下的文字标识为’测试’，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.multilineinput-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.multilineinput-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.10.条形标尺-rectgauge

描述

条形标尺，又名刻度尺，父对象为ui，无子对象

事件

---

onValueChange： 条形标尺指示值变化时触发，函数定义：onValueChange(value), value为当前指示值。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果为false，控件被禁止	ui.form-name.gauge-name.enabled=0 禁止标尺
visible	可见：如果为true，对象可见，否则隐藏	ui.form-name.gauge-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.gauge-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.gauge-name.verbose=1 设置对象触发的事件有效时向设置ScriptMode的串口发送事件消息
font	字体设置	无
style	刻度显示风格， 支持0-Left, 1-Right，2-Left_Right 。	ui.form-name.gauge-name.style=0 设置刻度显示在标尺左侧
majorTickCount	主刻度数	ui.form-name.gauge-name.majorTickCount=10 设置10个主刻度标识
tickMarkColor	刻度颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.tickMarkColor='#ff0000' 设定刻度为红色
tickMarkLength	刻度长度	ui.form-name.gauge-name.tickMarkLength=8 设定刻度为8像素长
chunkMargin	指示块间距	ui.form-name.gauge-name.chunkMargin=2 设定指示块间距为2像素
chunkColor	指示块颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.chunkColor='#ff0000' 设定指示块为红色
chunkColorType	指示块颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.guage-name.chunkColorType=4 设置指示块颜色类型为Circle
textVisible	文本可见	ui.form-name.gauge-name.textVisible=0 隐藏文本
textColor	文本颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.textColor='#00ff00' 设置文本为绿色
alarmEnabled	警报使能	ui.form-name.gauge-name.alarmEnabled=1 使能警报
alarmThreshold	警报阈值	ui.form-name.gauge-name.alarmThreshold=60 标尺值超过60按alarmColor定义颜色显示
alarmColor	警报显示颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.alarmColor='#ff0000' 设置超过警报阈值时的指示块为红色
minimum	value的最小值	ui.form-name.gauge-name.minimum = 0 设定value的最小值为0
maximum	value的最大值	ui.form-name.gauge-name.maximum = 100 设定value的最大值为100
value	标尺指示位置的当前值，必须在{minimum， maximum}范围内	ui.form-name.gauge-name.value= 50 设定value的当前值为50

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.gauge-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.gauge-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.11.圆形标尺-RoundGauge

描述

圆形标尺，又名仪表盘，父对象为ui，无子对象

事件

onValueChange： 仪表盘指示值变化时触发，函数定义：onValueChange(value), value为当前指示值。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果为false，禁止该控件。	ui.form-name.gauge-name.enabled=0 禁止圆形标尺控件
visible	可见：如果为true，对象可见，否则隐藏	ui.form-name.gauge-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.gauge-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.gauge-name.verbose=1 设置对象触发的事件有效时向通讯协议为scriptMode的串口发送事件消息
minimum	value的最小值	ui.form-name.gauge-name.minimum = 0 设定value的最小值为0
minimumVisible	value最小值可见，默认值为true	ui.form-name.gauge-name.minimumVisible = false 设置最小值不可见
maximum	value的最大值	ui.form-name.gauge-name.maximum = 100 设定value的最大值为100
maximumVisible	value最大值可见，默认值为true	ui.form-name.gauge-name.maximumVisible = false 设置最大值不可见
value	表针位置的当前值，必须在{minimum， maximum}范围内	ui.form-name.gauge-name.value= 50 设定value的当前值为50
alarmEnabled	报警颜色使能：如果为true, 当值大于alarmThreshold时显示报警色	ui.form-name.gauge-name.alarmEnabled=1 使能报警色
alarmThreshold	报警门限值	ui.form-name.gauge-name.alarmThreshold=80 设置报警门限为80
alarmColor	报警色：颜色定义参看 控件对象	ui.form-name.gauge-name.alarmColor='#ff0000′ 设定报警色为红色
direction	指针旋转方向：0-Clockwise 顺时针，1-CounterClockwise 逆时针	ui.form-name.gauge-name.direction=0 设置指针按顺时针旋转
textVisible	文字可见	ui.form-name.gauge-name.textVisible=0 设置文字不可见
textColor	文字颜色：颜色定义参看 控件对象	ui.form-name.gauge-name.textColor='#ff0000′ 设置文字为红色
tickMarkVisible	刻度可见	ui.form-name.gauge-name.tickMarkVisible=1 设置刻度可见
tickMarkColor	刻度颜色：颜色定义参看 控件对象	ui.form-name.gauge-name.tickMarkColor='#ff0000′ 设置刻度为红色
tickMarkBackgroundColor	刻度背景色：颜色定义参看 控件对象	ui.form-name.gauge-name.tickMarkBackgroundColor='#ff0000′ 设置刻度背景为红色
tickMarkBackgroundColorType	刻度背景色类型：背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.gauge-name.tickMarkBackgroundColorType=1 设置刻度背景色类型为Linear_B
majorTickCount	主刻度数	ui.form-name.gauge-name.majorTickCount=10 设置显示10个主刻度
minorTickCount	次刻度数	ui.form-name.gauge-name.minorTickCount=2 设置每个主刻度显示2个次刻度
tickMarkLength	刻度长度	ui.form-name.gauge-name.tickMarkLength=10 设定刻度长度为10像素
tickMarkMargin	刻度与外边框间距	ui.form-name.gauge-name.tickMarkMargin=5 设定刻度与外边框间距为5像素
needleStartAngle	指针的起始角度，就是当value=minimum时，指针的位置角度，范围为0-360	ui.form-name.gauge-name.needleStartAngle=90 设定指针的起始角度为90度
needleSpanAngle	指针跨度，从起始角度起，指针可转动的角度范围，值为0-360	ui.form-name.gauge-name.needleSpanAngle=360 设定指针的跨度为360度
needleLength	指针长度，单位：像素	ui.form-name.gauge-name.needleLength=20 设定指针长度为20

needleInnerWidth	指针内侧宽度，单位：像素	ui.form-name.gauge-name.needleInnerWidth=8 设定指针内侧宽度为8
needleOuterWidth	指针外侧宽度，单位：像素	ui.form-name.gauge-name.needleOuterWidth=8 设定指针外侧宽度为8
needleColor	指针颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.needleColor='#ff0000′ 设定指针为红色
needleColorType	指针颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.gauge-name.needleColorType=1 设置指针颜色类型为Linear_B
pivotRadius	中枢圆的半径，单位：像素	ui.form-name.gauge-name.pivotRadius=8 设定中枢圆的半径为8
pivotColor	中枢圆的颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.pivotColor='#ff0000′ 设定中枢圆为红色
pivotColorType	中枢圆颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.gauge-name.pivotColorType=1 设置中枢圆颜色类型为Linear_B
borderRadius	外边界的圆角半径，单位：像素	ui.form-name.gauge-name.borderRadius=8 设定外边界圆角半径为8
borderWidth	外边界的宽度，单位：像素	ui.form-name.gauge-name.borderWidth=8 设定外边界宽为8像素
borderColor	外边界颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.borderColor='#ff0000′ 设定边界为红色
borderColorType	边界颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.gauge-name.borderColorType=1 设置边界颜色类型为Linear_B
borderStartAngle	外边界起始角度，范围为0-360	ui.form-name.gauge-name.borderStartAngle=90 设定外边界起始角度为90度
borderSpanAngle	外边界跨度，范围为0-360	ui.form-name.gauge-name.borderSpanAngle=180 设定外边界跨度为180度
backgroundImage	背景图片	ui.form-name.gauge-name.backgroundImage='test.png' 设定背景图片为资源文件test.png
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.form-name.gauge-name.backgroundColor='#ff0000′ 设定背景为红色
backgroundColorType	背景颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.gauge-name.borderColorType=6 设置无背景色

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.gauge-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.gauge-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.12.滑动尺-slider

描述

滑动尺，父对象为ui，无子对象

事件

onValueChange： 当前滑动块移动时触发

触发函数定义：onValueChange（value），value表示当前滑动块的位置值。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.slider-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
font	字体	无
minimum		

	最小值	ui.form-name.slider-name.minimum=0 设置value最小值为0
maximum	最大值	ui.form-name.slider-name.maximum=100 设置value最大值为100
value	滑块位置值，必须在maximum和maximum之间	ui.form-name.slider-name.value=4 设置当前值为4
singleStep	滑动一次步长	无
pageStep	点击滑轨空白处，滑动块移动一次步长	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.slider-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.slider-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
orientation	方向，0-Horizontal，1-Vertical可选	ui.form-name.slider-name.orientation=1 设置为垂直显示的进度条
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.slider-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
handleColor	滑动块颜色，颜色定义参看 控件对象	ui.form-name.slider-name.handleColor='#ff0000' 设置滑动块为红色
handleWidth	滑动块宽度，单位：像素	ui.form-name.slider-name.handleWidth=40 设置滑动块宽度为40
handleHeight	滑动块高度，单位：像素	ui.form-name.slider-name.handleHeight=30 设置滑动块高度为30
handleBorderColor	滑动块边框颜色，颜色定义参看 控件对象	ui.form-name.slider-name.handleBorderColor='#ff0000' 设置滑动块边框为红色
handleBorderWidth	滑动块边框宽度，单位：像素	ui.form-name.slider-name.borderWidth=5 设置对象边框宽为5个像素
handleBorderRadius	滑动块边框拐角弧度半径，单位：像素	ui.form-name.slider-name.borderRadius=8 设置对象边框拐角半径为8个像素
grooveColor	轨道激活区颜色，颜色定义参看 控件对象	ui.form-name.slider-name.grooveColor='#ff0000' 设置滑动轨道为红色
grooveDarkColor	轨道非激活区颜色，颜色定义参看 控件对象	ui.form-name.slider-name.grooveDarkColor='#ff0000' 设置滑动轨道非激活区为红色
grooveWidth	轨道宽度，单位：像素	ui.form-name.slider-name.grooveWidth=5 设置轨道宽为5个像素
grooveBorderColor	轨道边界颜色，颜色定义参看 控件对象	ui.form-name.slider-name.grooveBorderColor='#ff0000' 设置滑动轨道边框为红色
grooveBorderWidth	轨道边框宽度，单位：像素	ui.form-name.slider-name.grooveBorderWidth=5 设置轨道边框宽为5个像素
grooveBorderRadius	轨道边框拐角弧度半径，单位：像素	ui.form-name.slider-name.grooveBorderRadius=8 设置进度块边框拐角半径为8个像素

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.slider-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.slider-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.13.进度条-progressbar

描述

进度条，父对象为ui，无子对象

事件

onValueChange： 进度条当前值发生变化时触发

触发函数定义：onValueChange（value），value表示当前进度条的值。

onUserEvent: 接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.progressbar-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
font	字体	无
minimum	最小值	ui.form-name.progressbar-name.minimum=0 设置value最小值为0
maximum	最大值	ui.form-name.progressbar-name.maximum=100 设置value最大值为100
value	进度值，必须在maximum和maximum之间	ui.form-name.progressbar-name.value=4 设置进度当前值为4
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.progressbar-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.progressbar-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件.
orientation	方向，0-Horizontal, 1-Vertical	ui.form-name.progressbar-name.orientation=1 设置为垂直显示的进度条
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.progressbar-name.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
textVisible	进度标识文字可见	ui.form-name.progressbar-name.textVisible=0 隐藏进度标识文字
textColor	文字颜色,颜色定义参看 控件对象	ui.form-name.progressbar-name.textColor=’#00ff00′ 设置文字颜色为绿色
textAlign	文字对齐方式: 0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.progressbar-name.textAlign=0 设置对象文本居中对齐
textFormat	进度标识文字格式，值为：0-Percent， 1-Value	ui.form-name.progressbar-name.textFormat=0 设置进度标识文字为百分比显示
borderColor	边框颜色，颜色定义参看 控件对象	ui.form-name.progressbar-name.borderColor=’#ff0000′ 设置边框为红色
borderWidth	边框宽度，单位：像素	ui.form-name.progressbar-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.progressbar-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型: 0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.progressbar-name.borderType=4 设置对象无边框
backgroundColor	背景色，颜色定义参看 控件对象	ui.form-name.progressbar-name.backgroundColor=’#00ff00′ 设置背景为绿色
backgroundColorType	背景色类型 0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle，5-Pure, 6-No_Color	ui.form-name.progressbar-name.backgroundColorType=4 设置背景颜色类型为Circle
chunkColor	进度块颜色，颜色定义参看 控件对象	ui.form-name.progressbar-name.chunkColor=’#ff0000′ 设置进度块为红色
chunkColorType	进度块背景色类型: 0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle，5-Pure, 6-No_Color	ui.form-name.progressbar-name.chunkColorType=4 设置进度块颜色类型为Circle
chunkWidth	进度块宽度，单位：像素	ui.form-name.progressbar-name.chunkWidth=5 设置进度块宽为5个像素
chunkRadius	进度块拐角弧度半径，单位：像素	ui.form-name.progressbar-name.chunkRadius=8 设置进度块边框拐角半径为8个像素
chunkMargin	进度块与边框距离，单位：像素	ui.form-name.progressbar-name.chunkMargin=5 设置进度块与边框距离5个像素

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.progressbar-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.progressbar-name.resize(30,20) 设置对象宽30像素，高20像素
setText(text)	设置用户自定义的文本显示代替默认的进度值	ui.form-name.progressbar-name.setText(‘Full100%’) 设置进度值文本显示为Full 100%

8.1.14.圆形进度条-roundprogressbar

描述

圆形进度条，父对象为ui，无子对象

事件

onValueChange： 进度条当前值发生变化时触发，触发函数定义：onValueChange（value），value表示当前进度条的值。

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.progressbar-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.progressbar-name.visible=0 隐藏对象
font	字体	无
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.progressbar-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件.
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为script mode时适用。	ui.form-name.progressbar-name.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
minimum	最小值	ui.form-name.progressbar-name.minimum=0 设置value最小值为0
maximum	最大值	ui.form-name.progressbar-name.maximum=100 设置value最大值为100
value	进度值，必须在maximum和maximum之间	ui.form-name.progressbar-name.value=4 设置进度当前值为4
textVisible	进度标识文字可见	ui.form-name.progressbar-name.textVisible=0 隐藏进度标识文字
textColor	文字颜色,颜色定义参看 控件对象	ui.form-name.progressbar-name.textColor=’#00ff00′ 设置文字颜色为绿色
textFormat	文字格式：0-Percent百分比显示，1-Value实际值显示	ui.form-name.progressbar-name.textFormat=0 设置百分比显示进度
style	进度条显示风格：值为0-Donut， 1-Pie	ui.form-name.progressbar-name.textFormat=0 设置显示风格为donut
grooveBackgroundColor	轨道背景颜色，颜色定义参看 控件对象	ui.form-name.progressbar-name.grooveBackgroundColor=’#ff0000′ 设置轨道背景为红色
grooveBackgroundColorType	轨道背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.progressbar-name.grooveBackgroundColorType=0 设置轨道背景颜色类型为Linear_A
grooveWidth	轨道宽度，单位：像素	ui.form-name.progressbar-name.grooveWidth=5 设置对象轨道宽为5个像素
chunkColor	进度块颜色，颜色定义参看 控件对象	ui.form-name.progressbar-name.chunkColor=’#ff0000′ 设置进度块为红色
chunkColorType	进度块颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.progressbar-name.chunkColorType=0 设置进度块颜色类型为Linear_A
chunkWidth	进度块宽度，单位：像素	ui.form-name.progressbar-name.chunkWidth=5 设置进度块宽为5个像素
chunkCapType	进度块头部类型：0-Flat平头, 1-Round圆头	ui.form-name.progressbar-name.chunkCapType=1 设置进度块头部为圆形

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.progressbar-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.progressbar-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.15.段式数字-segmentnumber

描述

段式数字，父对象为ui，无子对象

事件

---

onTextChange： 显示的数字发生变化时触发，函数定义：onTextChange(text), text为当前显示数字值。

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.segmentnumber-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.segmentnumber-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.segmentnumber-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
digitCount	数字位数	ui.form-name.segmentnumber-name.DigitCount=5 设置显示的数字位数为5位
style	数字风格，0-Outline，1-Filled，2-Flat。	ui.form-name.segmentnumber-name.style=2 设置对象数字显示为平坦风格
text	数字值	ui.form-name.segmentnumber-name.text=’1234′ 设置对象的显示内容为数字1234
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.segmentnumber-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
textColor	文字颜色，颜色定义参看 控件对象	ui.form-name.segmentnumber-name.textColor=’#0000ff’ 设置字体为蓝色
textAlign	文字对齐方式。0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.segmentnumber-name.textAlign=0 设置对象文本居中对齐
borderColor	边框颜色,颜色定义参看 控件对象	ui.form-name.segmentnumber-name.borderColor=’#ff0000′ 设置边框颜色为红色
borderWidth	边框宽度，单位：像素	ui.form-name.segmentnumber-name.borderWidth=5 设置对象边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.segmentnumber-name.borderRadius=8 设置对象边框拐角半径为8个像素
borderType	边框类型：0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.segmentnumber-name.borderType=4 设置对象无边框
backgroundColor	背景色	无
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.segmentnumber-name.backgroundColorType=4 设置对象背景颜色类型为circle

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.segmentnumber-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.segmentnumber-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.16.时钟显示-clock

描述

时钟显示，根据硬件RTC时间，自动刷新显示时间，每秒刷新一次，父对象为ui，无子对象

事件

---

onAlarm： 设置的闹铃到点时触发，函数定义：onAlarm(time), time为闹铃设置时间。

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.clock-name.enabled=0 禁止对象
geometry	几何尺寸：定义页面的原点坐标（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.clock-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.clock-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
displayMode	显示模式：0-Time， 1-Date	ui.form-name.clock-name.displayMode=0 设置显示时间
style	显示数字风格：0-Outline，1-Filled，2-Flat	ui.form-name.clock-name.style=2 设置对象数字显示为平坦风格
textColor	文字颜色，颜色定义参看 控件对象	ui.form-name.clock-name.textColor=’#ff0000′ 设置对象数字为红色
timeFormat	时间显示格式： 0-HH_MM_SS（显示秒），1-HH_MM（不显示秒）可选	ui.form-name.clock-name.timeFormat=1 设置对象不显示秒
timezone	时区（默认为0-UTC)	ui.form-name.clock-name.timezone=8 设置时区为东8区，北京时间
alarm	闹钟时间	ui.form-name.clock-name.alarm=“12:00:00” 设置闹钟为12:00:00
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.clock-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息

方法

名称	功能说明	脚本例子
update()	立即刷新时间，适用于修改了RTC时钟后立即刷新时间显示	ui.form-name.clock-name.update()
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.clock-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.clock-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.17.复选框-checkbox

描述

复选框，父对象为ui，无子对象

事件

onCheck： 复选框选中时触发，触发函数定义：onCheck()

onUncheck：复选框取消选中时触发,触发函数定义：onUncheck()

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.checkbox-name.enabled=0 禁止对象
checked	锁定：如果勾选值为true, 如果未勾选值为false	ui.form-name.checkbox-name.checked=1 设置选中状态
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
font	字体类型，风格和尺寸	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.checkbox-name.visible=0 隐藏对象
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.checkbox-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
text	文字标识	ui.form-name.checkbox-name.text=”test” 设置对象当前语言环境下文字标识为”test”
textColor	文字颜色，颜色定义参看 控件对象	ui.form-name.checkbox-name.textColor=’#ff0000′ 设置对象文字为红色

textAlign	文字对齐方式。0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.checkbox-name.textAlign=0 设置对象文本居中对齐
spacingWidth	选择框与文字标识的距离，单位：像素	ui.form-name.checkbox-name.spacingWidth=5 设置选择框与文字标识距离5个像素
indicatorColor	选中时的颜色，黑（black）和白（white）可选	无
indicatorbackgroundColor	选择框背景色，颜色定义参看 控件对象	ui.form-name.checkbox-name.indicatorBackgroundColor='#ff0000′ 设置选择框背景为红色
indicatorSize	选择框尺寸，单位：像素	ui.form-name.checkbox-name.indicatorSize=25 设置对象选择框宽大小为25个像素
indicatorBorderWidth	选择框边框宽度，单位：像素	ui.form-name.checkbox-name.indicatorBorderWidth=2 设置选择框边框宽度为2个像素
indicatorBorderRadius	选择框边框拐角弧度半径，单位：像素	ui.form-name.checkbox-name.indicatorBorderRadius=8 设置选择框边框拐角半径为8个像素
indicatorBorderColor	选择框边框颜色，颜色定义参看 控件对象	ui.form-name.checkbox-name.indicatorBorderColor='#ff0000′ 设置选择框边框为红色

方法

名称	功能说明	脚本例子
setText(string, index)	设置语言环境为index时的文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.checkbox-name.setText('测试', 1) 设置语言环境1下的文字标识为'测试'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.checkbox-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.checkbox-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.18.下拉框-combobox

描述

下拉框，父对象为ui，无子对象

事件

onIndexChange： 选中某个下拉选项时触发，触发函数定义：onIndexChange(index)， index是当前选中的索引号

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

borderRadius

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.combobox-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
font	字体类型，风格和尺寸	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.combobox-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.combobox-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.combobox-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
textColor	文字颜色	无
textAlign	文字对齐方式：0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.combobox-name.textAlign=0 设置对象文本居中对齐
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.form-name.combobox-name.backgroundColor="#ff0000″ 设置背景为红色

backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.combobox-name.backgroundColorType=5 设置对象背景色为纯色
borderColor	边框背景色，颜色定义参看 控件对象	ui.form-name.combobox-name.borderColor=”#ff0000″ 设置边框为红色
borderWidth	边框宽度，单位：像素	ui.form-name.combobox-name.borderWidth=2 设置对象边框宽度为2像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.combobox-name.borderRadius=8 设置选择框边框拐角半径为8个像素
borderType	边框类型：0-Inset，1-Outset，2-Dotted，3-Solid，4-No_Border。	ui.form-name.combobox-name.borderType=0 设置对象边框类型为inset
listTextColor	下拉选项字体颜色，颜色定义参看 控件对象	无
listBackgroundColor	下拉选项背景颜色，颜色定义参看 控件对象	无
listBorderColor	下拉边框颜色，颜色定义参看 控件对象	无
listSelectionColor	下拉选项被选中时颜色，颜色定义参看 控件对象	无
listBorderWidth	下拉边框宽度，单位：像素	ui.form-name.combobox-name.listBorderWidth=2 设置对象下拉边框宽度为2像素
listBorderRadius	下拉边框拐角弧度半径，单位：像素	ui.form-name.combobox-name.listBorderRadius=8 设置下拉边框拐角半径为8个像素
list	下拉选项列表	ui.form-name.combobox-name.list=”option1, option2, option3, option4″ 设置当前语言环境下的下拉选项为4项，显示为option1， option2，option3, option4
index	索引值	ui.form-name.combobox-name.index=2 设置下拉列表的当前索引值为2
currentText	当前显示的文字	该属性是只读，不能修改

方法

名称	功能说明	脚本例子
addItemToList(name, index)	添加下拉选项：添加下拉选项到指定的索引位置，如果不传index参数，添加到选项列表最后。	ui.form-name.combobox-name.addItemToList(“optionX”, 1) 添加名称为optionX的选项到索引位置1 ui.form-name.combobox-name.addItemToList(“optionX”) 添加名称为optionX的选项到选项列表末尾
setListText(string, index)	设置语言环境为index时的下拉文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.checkbox-name.setText(‘选项1，选项2’, 1) 设置语言环境1下的下拉列表的两个选项：’选项1’和选项2，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.combobox-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.combobox-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.19.开关-switch

描述

开关，父对象为ui，无子对象

事件

onCheck： 开启时触发，触发函数定义：onCheck()

onUncheck：关闭时触发,触发函数定义：onUncheck()

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.switch-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
font	字体类型，风格和尺寸	无

visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.switch-name.visible=0 隐藏对象
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.switch-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
style	显示风格，支持0-Round，1-Rect	ui.form-name.switch-name.style=0 设置显示风格为圆角开关
checked	设置开关状态，如果为true，开关开启，否则关闭	ui.form-name.switch-name.checked=1 设置开关状态为开启
textVisible	文字标识可见	ui.form-name.switch-name.textVisible=0 隐藏文字标识
textOnColor	开启时文字颜色，颜色定义参看 控件对象	ui.form-name.switch-name.textOnColor='#0000ff' 开启时文字为蓝色
textOffColor	关闭时文字颜色，颜色定义参看 控件对象	ui.form-name.switch-name.textOffColor='#ff0000′ 关闭时文字为蓝色
onText	开启时文字标识	ui.form-name.switch-name.onText='test' 在当前语言环境下开启时显示文字test
offText	关闭时文字标识	ui.form-name.switch-name.offText='test' 在当前语言环境下关闭时显示文字test
handleOnColor	开启时滑块颜色，颜色定义参看 控件对象	ui.form-name.switch-name.handleOnColor='#0000ff' 开启时滑块颜色为蓝色
handleOffColor	关闭时滑块颜色，颜色定义参看 控件对象	ui.form-name.switch-name.handleOffColor='#ff0000′ 关闭时滑块颜色为红色
handleColorType	滑块颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.switch-name.handleColorType=4 设置滑块颜色类型为Circle
handleMargin	滑块间距	ui.form-name.switch-name.handleMargin=2 设置滑块与轨道的间距为2像素
grooveOnColor	开启时轨道颜色，颜色定义参看 控件对象	ui.form-name.switch-name.grooveOnColor='#0000ff' 开启时轨道颜色为蓝色
grooveOffColor	关闭时轨道颜色，颜色定义参看 控件对象	ui.form-name.switch-name.grooveOffColor='#00ff00′ 关闭时轨道颜色为绿色
grooveColorType	轨道颜色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.switch-name.grooveColorType=4 设置轨道颜色类型为Circle
grooveHeightAdjustable	轨道高度可调	ui.form-name.switch-name.grooveHeightAdjustable=1 设置轨道高度可调
grooveHeight	轨道高度	ui.form-name.switch-name.grooveHeight=10 设置轨道高度为10像素
borderOnly	只显示轨道边框，轨道内部不填充颜色	ui.form-name.switch-name.borderOnly=1 设置轨道只显示边框
borderWidth	轨道边框宽度	ui.form-name.switch-name.borderWidth=3 设置显示边框宽度为3像素

方法

名称	功能说明	脚本例子
setOnText(string, index)	设置语言环境为index时的开启文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.switch-name.setOnText('开', 1) 设置语言环境1下的开启时文字标识为'开'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
setOffText(string, index)	设置语言环境为index时的关闭文字标识：string为文字字符串，index为语言环境索引值。当通过service.setLanguage(index)设置对应的语言环境时，该文字字符串自动显示。	ui.form-name.switch-name.setOffText('开', 1) 设置语言环境1下的关闭时文字标识为'开'，当切换到语言环境1时，该标识自动显示替代之前的语言环境字符串
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.switch-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.switch-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.20.波形图-plot

描述

波形图，父对象为ui，无子对象

事件

onLoad： 波形图加载时触发，函数定义：onLoad(), 无参数。

onRefresh: 波形图刷新时触发，函数定义: onRefresh(), 无参数, 波形可定义按时间周期定时自动刷新，或者手动刷新。

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果为false，控件被禁止	ui.form-name.plot-name.enabled=0 禁止对象
visible	可见：如果为true，对象可见，否则隐藏	ui.form-name.plot-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.plot-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.plot-name.verbose=1 设置对象触发的事件有效时向设置ScriptMode的串口发送事件消息
font	字体设置	无
channelCount	波形曲线数量	ui.form-name.plot-name.channelCount=4 设置4条波形曲线显示
lineWeight	线条粗细	ui.form-name.plot-name.lineWeight=2 设置曲线粗细为2个像素
backbroundColorEnabled	使能背景颜色，颜色由属性backgroundColor定义	ui.form-name.plot-name.backgroundColorEnabled=0 不显示波形背景颜色
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.form-name.plot-name.backgroundColor='#ff0000′ 设定波形背景为红色
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.plot-name.backgroundColorType=4 设置背景颜色类型为Circle
textColor	字体颜色，颜色定义参看 控件对象	ui.form-name.plot-name.textColor='#ff0000′ 设定坐标字体为红色
textVisible	字体可见	ui.form-name.plot-name.textVisible=0 隐藏坐标字体
gridlineColor	网格颜色，颜色定义参看 控件对象	ui.form-name.plot-name.gridlineColor='#ff0000′ 设定网格为红色
gridlineVisible	网格可见	ui.form-name.plot-name.gridlineVisible=0 隐藏网格
tickMarkLength	主刻度长度，次刻度长度为主刻度2/3，单位：像素	ui.form-name.plot-name.tickMarkLength=20 设置主刻度长度为20像素
xTickColor	X轴刻度颜色，颜色定义参看 控件对象	ui.form-name.plot-name.xTickColor='#ff0000′ 设定X轴刻度为红色
xLeftTickValue	X轴左刻度值	ui.form-name.plot-name.xLeftTickValue=0 设定X轴左刻度值为0
xRightTickValue	X轴右刻度值	ui.form-name.plot-name.xRightTickValue=10 设定X轴右刻度值为10
xMajorTickCount	X轴主刻度数	ui.form-name.plot-name.xMajorTickCount=5 设定X轴主刻度为5个
xMinorTickCount	X轴次刻度数	ui.form-name.plot-name.xMinorrTickCount=10 设定X轴次刻度为10个
yTickColor	Y轴刻度颜色，颜色定义参看 控件对象	ui.form-name.plot-name.yTickColor='#ff0000′ 设定Y轴刻度为红色
yLowerTickValue	Y轴下刻度值	ui.form-name.plot-name.yLowerTickValue=0 设定Y轴下刻度值为0
yUpperTickValue	Y轴上刻度值	ui.form-name.plot-name.yUpperTickValue=10 设定Y轴上刻度值为10
yMajorTickCount	Y轴主刻度数	ui.form-name.plot-name.yMajorTickCount=5 设定Y轴主刻度为5个
yMinorTickCount	Y轴次刻度数	ui.form-name.plot-name.yMinorrTickCount=10 设定Y轴次刻度为10个
tickPerStep	步长，每刷新一个数据的X轴移动距离，单位：一个X轴次刻度	ui.form-name.plot-name.tickPerCount=0.5 每刷新一个数据，曲线X轴上移动半个次刻度
refreshMode	波形刷新模式：0-Shift_Forward(正向移动)， 1-Shift_Backward(反向移动)， 2-Replace_Forward(正向替代)，3-Replace_Backward(反向替代)	ui.form-name.plot-name.refreshMode=0 波形正向移动刷新
autoRefresh	自动刷新使能，刷新间隔由属性autoRefreshInterval定义	ui.form-name.plot-name.autoRefresh=1 使能自动刷新模式
autoRefreshInterval	自动刷新时间间隔，单位为毫秒	ui.form-name.plot-name.autoRefreshInterval=100 自动刷新间隔为100毫秒

方法

名称	功能说明	脚本例子
setChannelColor(channel, color);	设置曲线颜色，channel表示曲线的通道号，color定义参看颜色定义参看 控件对象。该方法一般在onLoad事件函数中调用初始化各曲线颜色。	ui.form-name.plot-name.setChannelColor(0, '#00ff00') 设置曲线0的颜色为绿色
setChannelVisible(channel, visible);	设置曲线可见性，channel表示曲线的通道号，visible表示可见性：0-隐藏，1-可见。	ui.form-name.plot-name.setChannelVisible(2, 0) 隐藏曲线2的显示
setData(c0, c1, …)	设置曲线的Y轴值，参数可变，c0表示通道0，c1表示通道1，以此类推。所有设置的值将缓存，直到调用refresh()方法或者自动刷新曲线时，才一次性按先入先出方式读取刷新到屏幕。	ui.form-name.plot-name.setData(8, 12) 设置曲线0和1的Y轴值分别为8和12，未明确设置的曲线将缓存与上一次相等的值。
refresh()	读取缓存的所有曲线值刷新到屏幕	ui.form-name.plot-name.refresh() 刷新曲线显示
clear()	清除当前显示的所有通道曲线	ui.form-name.plot-name.clear() 清除当前显示的所有通道曲线
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.plot-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.plot-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.21.指示灯-led

描述

LED指示灯，父对象为ui，无子对象

事件

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于"禁止"状态，对所有触摸操作无反应。	ui.form-name.led-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.led-name.visible=0 隐藏对象
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.led-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息
borderColor	边界颜色，颜色定义参看 控件对象	ui.form-name.led-name.borderColor='#0000ff' 设置LED灯边界为蓝色
onColor	开启时LED颜色，颜色定义参看 控件对象	ui.form-name.led-name.onColor='#ff0000′ 开启时LED为红色
offColor	关闭时LED颜色，颜色定义参看 控件对象	ui.form-name.led-name.offColor='#000000′ 关闭时led为黑色
blinkInterval	闪烁间隔，单位：ms	ui.form-name.led-name.blinkInterval=500 设置LED闪烁间隔500ms
state	LED的状态：0-Off，1-On，2-Blink	ui.form-name.led-name.state=2 设置LED为闪烁状态

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.led-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.led-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.22.形状-shape

描述

形状，父对象为ui，无子对象

事件

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.shape-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.shape-name.visible=0 隐藏对象
type	类型：0-Line，1-Rectangle，2-Ellipse，3-Triangle	ui.form-name.shape-name.type=0 设置形状为线
lineEndAStyle	线端点A风格：0-No_Style, 1-Arrow, 2-Square, 3-Circle	ui.form-name.shape-name.lineEndAStyle=1 设置端点A为箭头
lineEndBStyle	线端点B风格：0-No_Style, 1-Arrow, 2-Square, 3-Circle	ui.form-name.shape-name.lineEndBStyle=1 设置端点B为箭头
cornerRadius	圆角半径，单位：像素	ui.form-name.shape-name.cornerRadius=10 设置圆角半径为10像素
direction	方向：0-Clock_12， 1-Clock_3, 2-Clock_6, 3-Clock_9	ui.form-name.shape-name.direction=1 设置图形的方向为3点种方向
lineStyle	线风格：0-No_Pen, 1-Solid, 2-Dash, 3-Dot, 4-Dash-Dot, 5-DashDotDot	ui.form-name.shape-name.lineStyle=1 设置图形线为实线
lineWeight	线粗，单位：像素	ui.form-name.shape-name.lineWeight=10 设置线粗10像素
lineColor	线颜色，颜色定义参看 控件对象	ui.form-name.shape-name.lineColor=’#0000ff’ 设置线为蓝色
fillingColor	填充颜色，颜色定义参看 控件对象	ui.form-name.shape-name.fillingColor=’#ff0000′ 设置填充色为红色
fillingColorType	填充色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Radial_Center, 5-Pure, 6-No_Color可选	ui.form-name.shape-name.fillingColorType=0 设置填充色类型为Linear_A
antiAliasing	反锯齿显示	ui.form-name.shape-name.antiAliasing=1 使能反锯齿显示
effect	显示效果：0-None, 1-Shadow, 2-Blur	ui.form-name.shape-name.effect=1 显示效果为Shadow
blurRadius	模糊半径	ui.form-name.shape-name.blurRadius=2 设置模糊半径为2

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.shape-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.shape-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.23.表格-Table

描述

形状，父对象为ui，无子对象

事件

onItemClicked： 单元格被点击时触发（单元格必须有内容才触发！）触发函数定义：onItemClicked(row, column, text)，row表示行，column表示列，text表示当前单元格的文字。

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.table-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.table-name.visible=0 隐藏对象
rowCount	行数	ui.form-name.table-name.rowCount=3 设置行数为3
columnCount	列数	ui.form-name.table-name.columnCount=4 设置列数为4
columnLabel	列标签	ui.form-name.table-name.columnLabel=’aaa,bbb,ccc’ 设置列标签，用逗号分隔每列
rowLabel	行标签	ui.form-name.table-name.rowLabel=’aaa,bbb,ccc’ 设置行标签，用逗号分隔每行
rowHeaderVisible	行表头可见	ui.form-name.table-name.rowHeaderVisible=false 隐藏行表头
columnHeaderVisible	列表头可见	ui.form-name.table-name.columnHeaderVisible=false 隐藏列表头
rowWrapContent	行高适应内容	ui.form-name.table-name.rowWrapContent=true 每一行的高度按内容高度自适应
columnWrapContent	列宽适应内容	ui.form-name.table-name.columnWrapContent=true 每一列的宽度按内容宽度自适应
headerTextColor	表头文字颜色	ui.form-name.table-name.headerTextColor=’#ff0000′ 设置表头文字为红色
headerBackgroundColor	表头背景颜色	ui.form-name.table-name.headerBackgroundColor=’#0000ff” 设置表头背景颜色为蓝色
headerBackgroundColorType	表头背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.table-name.headerBackgroundColorType=4 设置表头的背景颜色类型为Circle
textAlign	文字对齐方式: 0-Center, 1-Left，2-Right，3-Top，4-Bottom	ui.form-name.table-name.textAlign=0 设置表格文本居中对齐
textColor	文字颜色, 颜色定义参看 控件对象	ui.form-name.table-name.textColor=’#ff0000′ 设置文字颜色为红色
backgroundColor	背景色，颜色定义参看 控件对象	ui.form-name.table-name.backgroundColor=’#0000ff” 设置背景色为蓝色
backgroundColorType	背景色类型：0-Linear_A, 1-Linear_B, 2-Radial_A, 3-Radial_B, 4-Circle, 5-Pure, 6-No_Color可选	ui.form-name.table-name.backgroundColorType=4 设置表格背景颜色类型为Circle
borderColor	边框颜色，颜色定义参看 控件对象	ui.form-name.table-name.borderColor=’#ff0000′ 设置边框颜色为红色
borderWidth	边框宽度，单位：像素	ui.form-name.table-name.borderWidth=5 设置边框宽为5个像素
borderRadius	边框拐角弧度半径，单位：像素	ui.form-name.table-name.borderRadius=8 设置边框拐角半径为8个像素
borderType	边框类型: 0-Solid，1-Dotted，2-No_Border	ui.form-name.table-name.borderType=2设置无边框

方法

名称	功能说明	脚本例子
setItemTextColor(row, column, color)	设置单元格的文字颜色	ui.form-name.table-name.setItemTextColor(0, 2, ‘#ff0000’) 设置第0行，第2列文字颜色为红色
setItemBackgroundColor(row, column, color)	设置单元格的背景颜色	ui.form-name.table-name.setItemBackgroundColor(0, 2, ‘#0000ff”) 设置第0行，第2列背景颜色为蓝色
getItemText(row, column)	获取单元格内容	var text = ui.form-name.table-name.getItemText(0, 2) 获取第0行，第2列的文字
setItem(row, column, text)	设置单元格内容：设置第row行，第column列文字为text, 注意行和列的索引都从0开始	ui.form-name.table-name.setItem(0, 2, ‘abc’) 设置第0行，第2列的文字为abc
setRowItem(row, text)	设置整行内容：row为行索引，text字符串，中间用逗号分隔每列的内容	ui.form-name.table-name.setRowItem(0, ‘aaa,bbb,ccc’) 设置第0行的内容为aaa bbb ccc
setColumnWidth(column, width)	设置某列的宽度：column 为列索引，width为宽度，单位像素	ui.form-name.table-name.setColumnWidth(3, 40) 设置索引为3的列宽度为40像素
setRowHeight(row, height)	设置某行的高度：row 为行索引，height为高度，单位像素	ui.form-name.table-name.setRowHeight(2, 30) 设置索引为2的行高度为30像素
getRowHeaderWidth()	获取行表头的宽度	var width = ui.form-name.table-name.getRowHeaderWidth(); 获取行表头宽度
getColumnHeaderHeight()	获取列表头的高度	var height = ui.form-name.table-name.getColumnHeaderHeight() 获取列表头高度
insertRow(row)	在指定位置插入一个空行	ui.form-name.table-name.insertRow(0) 在第0行插入一个空行

insertRowItem(row, text)	在指定位置插入一行并填充内容	ui.form-name.table-name.insertRowItem(2, ‘aaa,bbb,ccc’) 在第2行插入一个行并填充内容
removeRow(row)	删除一行：row为行索引	ui.form-name.table-name.removeRow(4) 删除索引为4的行
currentRow()	获取当前选中的行索引	var row = currentRow() 返回当前选中的行索引
currentColumn()	获取当前选中的列索引	var col = currentColumn() 返回当前选中的列索引
clearContents()	清除表格所有单元格内容（不包含表头标签）	ui.form-name.table-name.clearContents() 清除表格所有单元格内容
clear()	清除表格所有内容，包含表头自定义的标签	ui.form-name.table-name.clearContents() 清除表格所有内容
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.table-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.table-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.24.二维码-QRCode

描述

二维码，父对象为ui，无子对象

事件

onUserEvent：接收自定义事件，当订阅了某自定义事件，通过service.emitEvent(name, value)广播时触发。

属性

名称	功能说明	脚本例子
enabled	使能：如果选中，对象处于激活状态，否则处于“禁止”状态，对所有触摸操作无反应。	ui.form-name.qrcode-name.enabled=0 禁止对象
geometry	几何尺寸：对象的坐标位置（X, Y），长和宽	无
visible	可见：如果选中，对象可见，否则隐藏	ui.form-name.qrcode-name.visible=0 隐藏对象
layer	图层：显示图层，值为0-Bottom_Layer, 1-Top_Layer.	ui.form-name.qrcode-name.layer=0 将控件置于显示底层，如果其他top层的控件与该控件重合，将覆盖该控件
textColor	文字颜色	ui.form-name.qrcode-name.textColor=”#0000ff” 设置背景为蓝色
backgroundColor	背景颜色，颜色定义参看 控件对象	ui.form-name.qrcode-name.backgroundColor=”#ff0000″ 设置背景为红色
correctionLevel	设置容错率，值为 0-Level_L, 1-Level_M, 2-Level_Q, 3-Level_H	ui.form-name.qrcode-name.correctionLevel=3 设置容错率为3(Level_H)最高容错率
kanjiMode	使能日文编码(双字节编码)	ui.form-name.qrcode-name.kanjiMode=1 使能日文编码
logo	标志	ui.form-name.qrcode-name.logo=”logo.png” 设置二维码中心显示的标志为图片logo.png
text	二维码表示的文字	ui.form-name.qrcode-name.text=”wareexpress.com” 设置二维码表示的文字为 “wareexpress.com”

方法

名称	功能说明	脚本例子
move(x,y)	移动对象：将对象移动到(x,y)坐标位置，坐标系的原点在屏幕的左上角。	ui.form-name.qrcode-name.move(10,10) 移动对象到坐标（10，10）
resize(width, height)	设置对象宽和高	ui.form-name.qrcode-name.resize(30,20) 设置对象宽30像素，高20像素

8.1.25.定时器-timer

描述

定时器，父对象为ui，无子对象

事件

onTimeout： 定时时间到时触发，函数定义：onTimeout(counter), counter为timeout发生的次数。

属性

名称	功能说明	脚本例子
enabled	使能：如果为true，定时器开始工作并触发timeout事件，否则定时器停止	ui.form-name.timer-name.enabled=0 停止计时器
interval	timeout事件触发的时间间隔，单位：毫秒(ms)	ui.form-name.timer-name.interval=500 设置定时器触发间隔为500ms
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	ui.form-name.timer-name.verbose=1 设置对象触发的事件有效时向串口发送事件消息

方法

名称	功能说明	脚本例子
start	启动定时器 – 不带参数，timeout事件无限循环发生，直到调用stop()方法或者属性enabled=0是才停止； – 带一个参数，表示定时器timeout的次数，到达该次数后，定时器自动停止。	ui.form-name.timer-name.start(10) 启动定时器，10次timeout后自动停止 ui.form-name.timer-name.start() 启动定时器，
stop	停止定时器	ui.form-name.timer-name.stop() 停止定时器

8.1.26.文件-file

描述

文件，父对象为form，不可见，为功能性控件，无子对象。

事件

无

属性

名称	功能说明	脚本例子
path	文件路径，用户可以访问/user目录下的板载存储空间，目前为48MB。如果需要访问U盘，也可设置为U盘路径	ui.form-name.file-name.path=’/user/test.txt’ 设置文件路径为/user/test.txt
mode	文件打开模式，值为整型（0-ReadWrite, 1-WriteOnly, 2-Append），默认值为 0	ui.form-name.file-name.mode=0 设置打开模式为可读可写
encoding	文件操作的字符编码，支持如下编码： “UTF-8”, “GBK”, “GB2312”, “gb2312.1980-0”, “gbk-0″,”ISO-8859-1”, “latin1”, “CP819”, “IBM819”, “iso-ir-100”, “csISOLatin1”, “ISO-8859-15”, “latin9”, “UTF-32LE”, “UTF-32BE”, “UTF-32”, “UTF-16LE”, “UTF-16BE”, “UTF-16”, “mulelao-1”, “roman8”, “hp-roman8”, “csHPRoman8”, “TIS-620”, “ISO 8859-11”, “WINSAMI2”, “WS2”, “Apple Roman”, “macintosh”, “MacRoman”, “windows-1258”, “CP1258”, “windows-1257”, “CP1257”, “windows-1256”, “CP1256”, “windows-1255”, “CP1255”, “windows-1254”, “CP1254”, “windows-1253”, “CP1253”, “windows-1252”, “CP1252”, “windows-1251”, “CP1251”, “windows-1250”, “CP1250”, “IBM866”, “CP866”, “csIBM866”, “IBM874”, “CP874”, “IBM850”, “CP850”, “csPC850Multilingual”, “ISO-8859-16”, “iso-ir-226”, “latin10”, “ISO-8859-14”, “iso-ir-199”, “latin8”, “iso-celtic”, “ISO-8859-13”, “ISO-8859-10”, “iso-ir-157”, “latin6”, “ISO-8859-10:1992”, “csISOLatin6”, “ISO-8859-9”, “iso-ir-148”, “latin5”, “csISOLatin5”, “ISO-8859-8”, “ISO 8859-8-I”, “iso-ir-138”, “hebrew”, “csISOLatinHebrew”, “ISO-8859-7”, “ECMA-118”, “greek”, “iso-ir-126”, “csISOLatinGreek”, “ISO-8859-6”, “ISO-8859-6-I”, “ECMA-114”, “ASMO-708”, “arabic”, “iso-ir-127”, “csISOLatinArabic”, “ISO-8859-5”, “cyrillic”, “iso-ir-144”, “csISOLatinCyrillic”, “ISO-8859-4”, “latin4”, “iso-ir-110”, “csISOLatin4”, “ISO-8859-3”, “latin3”, “iso-ir-109”, “csISOLatin3”, “ISO-8859-2”, “latin2”, “iso-ir-101”, “csISOLatin2”, “KOI8-U”, “KOI8-RU”, “KOI8-R”, “csKOI8R”, “Iscii-Mlm”, “Iscii-Knd”, “Iscii-Tlg”, “Iscii-Tml”, “Iscii-Ori”, “Iscii-Gjr”, “Iscii-Pnj”, “Iscii-Bng”, “Iscii-Dev”, “TSCII”, “GB18030”, “CP936”, “MS936”, “windows-936”, “EUC-JP”, “ISO-2022-JP”, “Shift_JIS”, “jisx0201*-0”, “jisx0208*-0”, “JIS7”, “SJIS”, “MS_Kanji”, “EUC-KR”, “ksc5601.1987-0”, “cp949”, “Big5”, “Big5-HKSCS”, “big5-0”, “big5hkscs-0”, “Big5-ETen”, “CP950”	ui.form-name.file-name.encoding=’UTF-8′ 设置字符编码为UTF-8

方法

名称	功能说明	脚本例子
open()	打开路径为属性path的文件，返回布尔值，true 表示打开成功	ui.form-name.file-name.open() 打开指定的文件
read(count)	二进制方式读取count个字节	var a=ui.form-name.file-name.read(10) 从指定文件读取10个字节内容
readAll()	二进制方式读取文件所有内容	var a=ui.form-name.file-name.readAll() 从指定文件读取所有内容
readLine()	读取一行文本，行以'\n'作为结束符	var a=ui.form-name.file-name.readLine() 从指定文件读取一行文本
readAllLines()	读取全部行文本，行以'\n'作为结束符	var a=ui.form-name.file-name.readAllLines() 从指定文件读取所有行文本
readStartLines(count)	读取文本文件起始的几行，返回字符串数组，数组长度为count	var a=ui.form-name.file-name.readStartLines(5); 读取文件开头的5行，返回为字符串数组
readEndLines(count)	读取文本文件末尾的几行，返回字符串数组，数组长度为count	var a=ui.form-name.file-name.readEndLines(5); 读取文件末尾的5行，返回为字符串数组
seek(offset)	设置文件读写的偏移位置	ui.form-name.file-name.seek(10) 设置读写的偏移位置为10个字节
write(data)	写入数据	ui.form-name.file-name.write(0xaa) 向指定文件写入16进制数aa
writeLine(line)	写入字符串行，自动添加'\n'作为行结束符	ui.form-name.file-name.writeLine('this is a test') 向指定文件写入文本行'this is a test\n'
close()	关闭文件	ui.form-name.file-name.close() 关闭指定文件
sync()	同步，强制缓存数据写入文件	ui.form-name.file-name.sync() 强制缓存数据写入指定文件
isOpen()	返回文件是否已打开，只有打开的文件才能进行读，写，同步等操作	var opened=ui.form-name.file-name.isOpen() 返回指定文件是否已打开
atEnd()	返回文件的读写位置是否已到文件尾	var a=ui.form-name.file-name.atEnd() 返回指定文件是否已读写到结尾
offset()	返回文件的读写位置	var a=ui.form-name.file-name.offset() 返回指定文件读写位置
length()	返回文件的大小	var l=ui.form-name.file-name.length() 返回指定文件大小
exists()	返回指定文件是否存在	var l=ui.form-name.file-name.exists('/user/test.txt') 检查/user/test.txt是否存在
isReadable()	返回文件是否可读	var readable=ui.form-name.file-name.isReadable('/user/test.txt') 检查/user/test.txt是否可读
isWritable()	返回文件是否可写	var writable=ui.form-name.file-name.isWritable('/user/test.txt') 检查/user/test.txt是否可写
copyTo(destination)	拷贝指定文件到destination路径	ui.form-name.file-name.copyTo('/user/test.txt') 将指定文件拷贝到'/user/test.txt'
remove()	删除当前文件	ui.form-name.file-name.remove() 删除当前文件

8.1.27.通道-channel

描述

原生程序IPC通讯通道。ExpOS支持原生程序（Linux下C/C++/GO语言编写的程序）与JavaScript编写的APP进行进程间通讯，该控件负责APP与原生程序间的消息发送和接收. 父对象为页面，无子对象。

事件

---

onMessage：接收到原生程序消息时触发，事件函数定义：onMessage(sender, msg, payload, responseId)，其中sender是发送该消息的原生程序名称字符串，msg为消息内容字符串，payload为消息负载字符串，responseId表示回应ID，如果该值大于0，需要接收程序调用respond()回应，否则不需要。

属性

名称	功能说明	脚本例子
name	通道名称	ui.form-name.channel-name.name='test' 设定通道名称为test
interestedPublisher	订阅通道名称 (订阅多个通道，中间可用逗号分隔)	ui.form-name.channel-name.interestedPublisher="xxx,yyy" 订阅xxx和yyy原生程序发布的消息，如原生程序调用publish()发送消息, 本控件的onMessage事件会触发

方法

名称	功能说明	脚本例子
postMessage(receiver, msg, payload)	向原生程序发送消息，receiver为原生程序通道名称，msg为消息内容，payload为负载字符串。该操作为异步发送，即函数立即返回，无论接收者是否收到	ui.form-name.channel-name.postMessage('xxx', 'This is a test', 'abc') 向名称为xxx的原生程序进程发送消息字符串'This is a test',负载字符串abc

subscribe(publisher)	订阅通道，publisher为感兴趣的发布者名称	ui.form-name.channel-name.subscribe(“xxx”) 订阅xxx原生程序发布的消息，如该原生程序调用publish()发送消息, 本控件的onMessage事件会触发
unsubscribe(publisher)	取消订阅通道，publisher为感兴趣的发布者名称	ui.form-name.channel-name.unsubscribe(“xxx”) 取消订阅xxx原生程序发布的消息
publish(msg, payload)	发布消息，订阅的原生进程会收到该消息	ui.form-name.channel-name.publish(“This is a test”, ‘abc’) 发布的消息字符串’This is a test’，负载字符串为abc
respond(sender, responseId, ret, reply)	回应同步消息发送者，sender为发送者通道名称，responseId是onMessage()中的responseId参数，ret为回应的返回整数值（取决于发送者如何处理，建议0或者其他值，0表示成功，其他值表示失败），reply表示需回应的字符串。在onMessage()处理函数中，如果responseId参数>0时(表示发送者同步发送消息给接收者)调用该函数，返回结果给发送者。	ui.form-name.channel-name.respond(‘xxx’, responseId，0, “okay”) 回应同步消息发送者xxx，返回结果值为0，结果字符串为okay

8.1.28.Http-http

描述

HTTP，父对象为ui，无子对象

事件

onReceive： 网络请求返回，函数定义：onReceive(url, statusCode), url为网络请求的URL地址， statusCode为网络返回状态码，200表示OK成功。

属性

名称	功能说明	脚本例子
userAgent	用户代理	ui.form-name.http-name.userAgent=’ExpOS/1.0′ 设置用户代理字符串为 ‘ExpOS/1.0’
autoRedirect	301或302自动跳转	ui.form-name.http-name.autoRedirect=true 设置自动跳转使能

方法

名称	功能说明	脚本例子
get(url)	发送http或https协议get请求 – url，请求的服务器URL地址	ui.form-name.http-name.get(‘https://www.baidu.com’) 发送get请求百度
post(url)	发送http或https协议post请求 – url，请求的服务器URL地址	ui.form-name.http-name.post(‘https://www.baidu.com’) 发送post请求百度
post(url, content)	发送http或https协议post请求 – url，请求的服务器URL地址 - content, post请求发送的数据内容	ui.form-name.http-name.post(‘https://www.baidu.com’，’abcd’) 发送post请求百度，数据内容为’abcd’
read()	必须在onReceive(url, statusCode)方法中调用，读取当前url对应的服务器response内容	ui.form-name.http-name.read() 读取当前请求返回的response内容
header(name)	获取http服务端返回response的头内容	ui.form-name.http-name.header(‘Content-Type’) 读取服务端的回复头’Content-Type’值
setHeader(name, value)	设置http请求头内容	ui.form-name.http-name.setHeader(‘Content-Type’, ‘application/json’) 设置请求头的’content-type’值

8.1.29.文件传输-ftp

描述

Ftp，父对象为ui，无子对象

事件

onFinish： 上传或下载任务完成时触发，函数定义：onFinish(operation), 参数operation表示当前完成的任务操作，如 “C:/tmp/test.txt -> test/test.txt “表示从C盘tmp目录上传文件test.txt到服务器路径test目录;如 “C:/tmp/test.txt <- test/test.txt”表示从服务器下载test.txt到本地C盘目录下

onError: 上传或者下载出错时触发，函数定义：onError(operation), 参数operation说明同上

onProgress: 上传或者下载的百分比进度，函数定义：onProgress(operation, progress), 参数operation说明同上, progress是进度值(0~100)

属性

名称	功能说明	脚本例子
host	主机名，可以是域名或者IP地址	ui.form-name.ftp-name.host=’wareexpress.com’ 设置主机为wareexpress.com ui.form-name.ftp-name.host=’192.168.1.12′ 设置主机为192.168.1.12
username	用户名	ui.form-name.ftp-name.username=’test’ 设置用户名为test
password	密码	ui.form-name.ftp-name.password=’abcd’ 设置密码为abcd
port	端口号，默认值为21	ui.form-name.ftp-name.port=21 设置端口号为21

方法

名称	功能说明	脚本例子
put(localPath, remotePath)	开启上传任务，参数localPath为本地路径，remotePath是远程路径	ui.form-name.ftp-name.put(‘C:/tmp/test.txt’, ‘test/test.txt’) 上传本地文件test.txt到FTP服务器test目录下
get(remotePath, localPath)	开启下载任务，参数localPath为本地路径，remotePath是远程路径	ui.form-name.ftp-name.get(‘test/test.txt’, ‘C:/tmp/test.txt’) 下载远程文件test.txt到本地
abort()	中止所有任务	ui.form-name.ftp-name.abort() 中止所有任务

8.1.30.套接字-socket

套接字**-socket**

描述

套接字，父对象为ui，无子对象

事件

onConnect(ip)： 套接字建立连接成功时触发，参数ip是连接的另一端的IP地址

onDisconnect(ip)：套接字连接断开时触发，参数ip是连接的另一端的IP地址

onReceive(ip)：接收到消息时触发，参数ip是消息发送端的IP地址

属性

名称	功能说明	脚本例子
endPoint	终端 0-CLINT客户端, 1-SERVER服务端	ui.form-name.socket-name.endPoint=1 设置套接字为服务端
protocol	协议 0-TCP, 1-UDP	ui.form-name.socket-name.protocol=1 设置套接字协议为UDP

方法

名称	功能说明	脚本例子
connect(ip, port, timeout)	连接TCP服务端,参数timeout是连接超时，单位：毫秒	ui.form-name.socket-name.connect(‘192.168.1.11’, 8888, 3000) 连接TCP服务端，端口号8888
disconnect()	断开当前TCP连接	ui.form-name.socket-name.disconnect() 断开当前TCP连接
isConnected()	判断当前TCP客户端是否已连接	var connected = ui.form-name.socket-name.isConnected() 返回是否已连接
read()	读取套接字另一端发送的二进制内容，返回值是字节数组	var data = ui.form-name.socket-name.read() 读取套接字另一端发送的所有二进制内容
readAsString()	读取套接字另一端发送的文本内容，返回值是字符串	var str = ui.form-name.socket-name.readAsString() 读取套接字另一端发送的所有文本内容
write(data)	写数据到TCP套接字	ui.form-name.socket-name.write(‘abcd’); 发送字符串’abcd’ ui.form-name.socket-name.write([0xaa, 0xbb]); 发送二进制数组
write(data, ip, port)		

	写数据报到UDP套接字	ui.form-name.socket-name.write('abcd', '192.168.1.11', 8888); 发送字符串'abcd' ui.form-name.socket-name.write([0xaa, 0xbb], '192.168.1.11', 8888); 发送二进制数组
startListening(ip, port)	监听端口，TCP服务端或者UDP客户端都可以监听端口	ui.form-name.socket-name.startListening('192.168.1.11', 8888); 监听端口8888
stopListening()	停止监听端口	ui.form-name.socket-name.stopListening();

8.2.硬件设备-device

8.2.1.Adc-adc

描述

模拟数字转换，父对象为device，无子对象。

事件

---

onValue：模拟量有变化时触发，事件函数定义：onValue(channel, raw)， channel表示通道，raw表示表示原始值(0~4095)

属性

名称	功能说明	脚本例子
tolerance	误差：当采集的模拟量变化超过误差值时，才会触发onValue方法	device.adc.tolerance=1 设置模拟量变化误差原始值为1
samplingFrequency	采样频率，单位为赫兹，最大为50Hz	device.adc.samplingFrequency = 10 设置采样频率为10Hz

方法

名称	功能说明	脚本例子
isSupported()	检测硬件是否支持ADC	var supported = device.adc.isSupported();
getResolutionBits()	获取采样精度位数（12位）	device.adc.getResolutionBits() 获取采样精度位数
getValue(channel)	获取通道channel的值（原始值0～4095）	device.adc.getValue(0) 返回通道0的当前模拟量的原始值
start()	开始采集	device.adc.start() 开始采集
stop()	停止采集	device.adc.stop() 停止采集

8.2.2.串口-com0,com1,com2

描述

串口，异步全双工串口对象，可实现串口数据的接收和发送，父对象为device，无子对象。目前支持三个串口，名称分别为com0,com1和com2。以下说明以com0为例，com1和com2同样适用。

通讯协议：支持脚本模式(**ScriptMode**)协议，用户自定义（**UserDefine**）协议和**Modbus**协议，在运行时只能选择其中一种。各种协议的比较见下表

	ScriptMode协议	**UserDefine**协议	**Modbus**协议
数据发送	可选择以下两种方式中的一种或两种混合使用: – 自动发送：在设计APP时，选中某个控件的属性中的"事件通知"(verbose)，APP在运行过程中，当对象事件(该事件需在动作列表中选中)产生时，ExpOS自动发送事件消息，无需用户编写任何脚本；如：页面main中名字为xx的按钮按下一次，ExpOS自动向串口发送"ui.main.xx.onPress()"字符串； -主动发送：不选择"事件通知"属性，与自定义协议发送方式一致	在任意脚本中直接调用**device.com0.write()**或者**device.com0.writeString()**函数，实现主动发送任意自定义数据或者字符串。用户需编写发送处理程序。	ExpOS设备只能做为Modbus主机，Modbus协议支持RTU和Ascii两种。 在任意脚本中直接调用**device.com0.writeModbus()**函数，实现Modbus主机发送指令给从机设备，该函数内部会自动添加**CRC**/**LRC**校验码
数据接收	外部控制单元发送有效的JavaScript脚本程序字符串，ExpOS接收后自动执行。如：外部控制单元向串口发送"ui.main.xx.text='test'"字符串（后必须加回车字符），页面main中名字为xxx的控件文本内容自动刷新为"test"。任何有效	在device.com0的onReceive(count)事件动作脚本里调用**device.com0.read()**函数读取有效串口数据。ExpOS根据设置好的阀值、帧头和帧尾等属性，自动接收串口数据放入缓冲区并按指定条件触发	在device.com0的onReceive()事件动作脚本里面调用**device.com0.read()**函数读取一帧Modbus数据，ExpOS对接收到的数据帧已做CRC/LRC校验，如果

的脚本程序都能通过通讯接口以字符串方式发送给ExpOS立即执行。用户无需编写接收处理程序。	onReceive(count)动作脚本执行，完成数据读取。用户需编写接收处理程序。	校验失败，不会调用onReceive，而是调用**onEvent()**函数来上报错误

事件

---

onReceive： 接收到串口数据时触发（只对UserDefine或者Modbus协议有效，ScriptMode协议不会产生该事件），事件函数定义：onReceive(count), count为串口接收缓冲区内的未读出数据字节数。

onSend：发送完一次（调用write()或者writeString()一次）串口数据时触发，事件函数定义：onSend(count)， count为串口已发送数据的字节数。

onEvent(event): 串口事件上报，事件类型有Event_ReceiveTimeout（接收超时）， Event_ReceiveOverflow （接收缓冲溢出），Event_ModbusFrameError（帧地址或长度错误），Event_ModbusCRCError（CRC校验错误），Event_ModbusLRCError （LRC校验错误）

属性

名称	功能说明	脚本例子
protocol	通讯协议：脚本模式(**ScriptMode**)协议，用户自定义（**UserDefine**）协议，**ModbusAscii**协议和**ModbusRTU**协议可选	无
rate	波特率：1200, 2400, 4800, 9600, 19200, 38400, 57600和115200可选	无
scriptModeReply	只适用**ScriptMode**协议，自动回复：如果为true， ExpOS收到脚本后，自动回复字符串”C+”表示执行成功，“C-”表示执行失败	无
threshold	只适用**UserDefine**协议，接收事件触发门限值：当收到新数据，而且接收缓冲区未读字节数等于或者大于此值，触发onReceive事件	device.com0.threshold=10 设定接收缓冲区字节未读字节数等于或者大于10时触发onReceive事件
headEnabled	只适用**UserDefine**协议，帧头字节检测使能：如果为true，ExpOS自动在接收缓冲区查找head属性指定的字节，如果找到，触发onReceive事件	device.com0.headEnabled=1 使能帧头字节检测功能
head	只适用**UserDefine**协议，帧头字节定义，输入16进制字符串，如3A表示0x3A，只有headEnabled为true时才有效, 多个字节使用空格分隔，如’3A 3B’	device.com0.head=‘3A 3B’ 设置待检测的帧头为0x3A 0x3B
tailEnabled	只适用**UserDefine**协议，帧尾字节检测使能：如果为true，ExpOS自动在接收缓冲区查找head和tail属性指定的字节，如果同时找到，触发onReceive事件	device.com0.tailEnabled=1 使能帧尾字节检测功能
tail	只适用**UserDefine**协议，帧尾字节定义，输入16进制字符串，如1D表示0x1D, 多个字节使用空格分隔，如’1D 1D’	device.com0.tail=’1D 1D’ 设置待检测的帧尾为0x1D 0x1D

方法

名称	功能说明	脚本例子
write(data)	写串口：参数可以是字符串（‘abc’），也可以是数组（[0x61, 0x62, 0x63]），也可以是多个16进制数，或者是以空格分隔的16进制字符串（如 ’61 62 63’）	device.com0.write(‘abc’) device.com0.write([‘a’, ‘b’, ‘c’]) device.com0.write(0x61, 0x62, 0x63) device.com0.write(’61 62 63′) 向串口写入16进制流 61 62 63
writeInt16(data)	向串口写整型（16位），data为数值或数组，字节顺序为小端	device.com0.writeInt16(0x1234) 向串口写入16进制流 34 12
writeInt16(data,isBigEndian)	向串口写整型（16位），data为数值或数组，isBigEndian表示是否大端	device.com0.writeInt16(0x1234, true) 向串口写入16进制流 12 34
writeInt32(data)	向串口写整型（32位），data为数值或数组，字节顺序为小端	device.com0.writeInt16(0x12345678) 向串口写入16进制流 78 56 34 12
writeInt32(data,isBigEndian)	向串口写整型（32位），data为数值或数组，isBigEndian表示是否大端	device.com0.writeInt16(0x12345678, true) 向串口写入16进制流 12 34 56 78
writeModbus(data)	向串口写Modbus数据，如果协议是ModbusRTU, 会自动添加CRC校验字节，如果协议是ModbusAscii, 会自动添加帧头(3A)和尾(0D 0A)以及LRC校验字节。参数说明同上（write(data)）	device.com0.writeModbus(’01 01 00 17 00 26′); 如果是ModbusRTU协议，则向串口写入16进制流 01 01 00 17 00 26 0D D4 如果是ModbusAscii协议，则向串口写入16进制流 3A 30 31 30 31 30 30 31 37 30 30 32 36 37 34 0D 0A
writeString(string, code)	向串口写字符串: 参数string为字符串, code为编码标准。默认为utf8，可支持gb2312或其它编码	device.com0.writeString(‘This is a test’) 向串口写入字符串’This is a test’，编码标准为utf8 device.com0.writeString(‘测试’, ‘gb2312’) 向串口写入字符串’测试’，编码标准为gb2312
readableBytes()	串口接收缓冲区中可读字节数	var count = device.com0.readableBytes() 读取接收缓冲区中可读字节数
readableFrames()	串口接收缓冲区中可以读的帧数（仅适用于设置了串口属性帧头(head)或者帧尾(tail)，或者两个属性都设置了）	var count = deviceo.com0.readableFrames() 读取接收缓冲区中可读帧数
read() read(length) read(length, peek)	读串口：不带参数表示读取所有数据并清空缓冲区，参数length表示读取的字节数，参数peek表示是否在接收缓冲区中删除读过的数据；注意：如果属性设置了帧头(head)	var a = []; a = device.com0.read(2) 从串口接收缓冲区读取两个字节并存如数组a中，读完后在接收缓冲区中删除这两个字节

	或者帧尾(tail)，读取到的数据不包含头和尾，只会包含有效字节	var a = []; a = device.com0.read(2, true) 从串口接收缓冲区读取两个字节并存如数组a中，读完后在仍然保留这两个字节在接收缓冲区中 var a = []; a = device.com0.read() 从串口接收缓冲区读取所有字节并存如数组a中，读完后清空接收缓冲区
clearReadBuffer()	清空接缓冲区。如果调用read()时不带参数，默认会清空缓冲，则不需要再调用clearReadBuffer()	device.com0.clearReadBuffer() 清空串口接收缓冲区

8.2.3.蜂鸣器-buzzer

描述

蜂鸣器，父对象为device，无子对象。

事件

onBeep: 发声时触发，事件函数定义：onBeep(count)， count表示自从调用play()到现在的累计发声次数

属性

名称	功能说明	脚本例子
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为ScriptMode时适用。	device.buzzer.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
duration	蜂鸣beep声一次持续时间，单位:毫秒	device.buzzer.duration=300 设置蜂鸣器一次beep声持续300毫秒
interval	蜂鸣beep声间隔时间，单位：毫秒	device.buzzer.interval=500 设置蜂鸣器beep声间隔500毫秒
volume	蜂鸣beep声音量，1-100， 1最小，100最大	device.buzzer.volume=100 设置蜂鸣器beep声音量最大

方法

名称	功能说明	脚本例子
play()	发beep声，0或者一个参数，参数表示beep声次数	device.buzzer.play() 蜂鸣器一直保持发声，直到stop()方法被调用 device.buzzer.play(10) 蜂鸣器发10次beep声，然后停止
stop()	停止beep声	device.buzzer.stop() 蜂鸣器停止发声

上一页 | 下一页

8.2.4.显示屏-display

描述

显示屏，父对象为device，无子对象。

事件

无

属性

名称	功能说明	脚本例子
brightness	背光亮度，参数代表亮度，范围0-100， 100最亮， 0 关闭背光	device.display.brightness=100 将背光调到最亮
enabled	使能，当enabled为true，打开显示屏相关电路，否则关闭. 当关闭后，系统会切断显示屏供电，系统功耗会下降。	device.display.enabled=0 关闭显示屏供电

方法

名称	功能说明	脚本例子

上一页 | 下一页

8.2.5.触摸屏-touch

描述

触摸屏，父对象为device，无子对象。

事件

onIdleTimeout：无触摸超时，事件函数定义：onIdleTimeout(timeout)， timeout表示超时的时间，单位：毫秒

onWakeup：无触摸超时后，再次按触摸时触发，事件函数定义: onWakeup()

属性

名称	功能说明	脚本例子
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为ScriptMode时适用。	device.touch.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
enabled	触摸使能：如果为false，触摸屏对触摸无响应，默认为true	device.touch.enabled=0 禁止触摸屏
idleTimeout	空闲(无触摸)超时，单位:毫秒	device.touch.idleTimeout=10000 触摸空闲超时10秒，如果10秒内无触摸，触发onIdleTimeout事件
lowPowerAtTimeout	超时低功耗。如果为true, 空闲超时后，系统自动进入低功耗状态，直到用户再次点击触摸屏唤醒。进入低功耗状态后，第一次触摸事件只唤醒系统，对界面无效。	device.touch.lowPowerAtTimeout=1 使能超时低功耗功能

方法

名称	功能说明	脚本例子
calibrate()	无条件进入触摸校准界面。对于电阻式触摸，在界面中”划屏幕对角线“动作，也可进入校准界面。电容式触摸出厂后，无需校准。如果确实需要再次校准，通过debugger执行了工厂复位，删除了工厂校准的数据，第一次启动时，才会弹出校准界面。	device.touch.calibrate() 进入触摸校准界面

8.2.6.输入输出-gpio

描述

通用输入输出，对应硬件的8个GPIO口，父对象为device，无子对象。

事件

onValue：端口发生变化时触发， 函数定义：onValue(pin, value)，pin表示端口号，value表示高低电平(1-高电平，0-低电平)

属性

名称	功能说明	脚本例子
p0Value	GPIO端口0的电平值	device.gpio.p0Value=1 设置GPIO端口0为高电平（需设置该端口为输出口）
p0Direction	GPIO端口0的方向，0-Input，1-Output	device.gpio.p0Direction=1 设置GPIO端口0为输出口
p0Trigger	GPIO端口0的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p0Trigger=1 设置GPIO端口0为上升沿触发onLevel事件
p1Value		

	GPIO端口1的电平值	device.gpio.p1Value=1 设置GPIO端口1为高电平（需设置该端口为输出口）
p1Direction	GPIO端口1的方向，0-Input，1-Output	device.gpio.p1Direction=1 设置GPIO端口1为输出口
p1Trigger	GPIO端口1的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p1Trigger=1 设置GPIO端口1为上升沿触发onLevel事件
p2Value	GPIO端口2的电平值	device.gpio.p2Value=1 设置GPIO端口2为高电平（需设置该端口为输出口）
p2Direction	GPIO端口2的方向，0-Input，1-Output	device.gpio.p2Direction=1 设置GPIO端口2为输出口
p2Trigger	GPIO端口2的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p2Trigger=1 设置GPIO端口2为上升沿触发onLevel事件
p3Value	GPIO端口3的电平值	device.gpio.p3Value=1 设置GPIO端口3为高电平（需设置该端口为输出口）
p3Direction	GPIO端口3的方向，0-Input，1-Output	device.gpio.p3Direction=1 设置GPIO端口1为输出口
p3Trigger	GPIO端口3的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p3Trigger=1 设置GPIO端口3为上升沿触发onLevel事件
p4Value	GPIO端口4的电平值	device.gpio.p4Value=1 设置GPIO端口4为高电平（需设置该端口为输出口）
p4Direction	GPIO端口4的方向，0-Input，1-Output	device.gpio.p4Direction=1 设置GPIO端口4为输出口
p4Trigger	GPIO端口4的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p4Trigger=1 设置GPIO端口4为上升沿触发onLevel事件
p5Value	GPIO端口5的电平值	device.gpio.p5Value=1 设置GPIO端口5为高电平（需设置该端口为输出口）
p5Direction	GPIO端口5的方向，0-Input，1-Output	device.gpio.p5Direction=1 设置GPIO端口5为输出口
p5Trigger	GPIO端口5的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p5Trigger=1 设置GPIO端口5为上升沿触发onLevel事件
p6Value	GPIO端口6的电平值	device.gpio.p6Value=1 设置GPIO端口2为高电平（需设置该端口为输出口）
p6Direction	GPIO端口6的方向，0-Input，1-Output	device.gpio.p6Direction=1 设置GPIO端口2为输出口
p6Trigger	GPIO端口6的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p6Trigger=1 设置GPIO端口6为上升沿触发onLevel事件
p7Value	GPIO端口7的电平值	device.gpio.p7Value=1 设置GPIO端口7为高电平（需设置该端口为输出口）
p7Direction	GPIO端口7的方向，0-Input，1-Output	device.gpio.p7Direction=1 设置GPIO端口7为输出口
p7Trigger	GPIO端口7的触发方式，0-None, 1-Rising, 2-Falling, 3-Both	device.gpio.p7Trigger=1 设置GPIO端口7为上升沿触发onLevel事件

8.2.7.矩阵键盘-keypad

描述

矩阵键盘，父对象为device，无子对象。目前支持4×4外接矩阵键盘，16个键的编码如下：

行	列	编码	名称
1	1	0x02	KEY_1
2	1	0x03	KEY_2
3	1	0x04	KEY_3
4	1	0x01	KEY_ESC
1	2	0x05	KEY_4
2	2	0x06	KEY_5
3	2	0x07	KEY_6
4	2	0x0F	KEY_TAB
1	3	0x08	KEY_7
2	3	0x09	KEY_8
3	3	0x0A	KEY_9
4	3	0x39	KEY_SPACE
1	4	0x34	KEY_DOT
2	4	0x0B	KEY_0
3	4	0x0E	KEY_BACKSPACE
4	4	0x1C	KEY_ENTER

事件

onPress：某个键按下时触发，事件函数定义：onPress(code)， code表示该键的编码

onRelease：某个键抬起时触发，事件函数定义：onRelease(code)， code表示该键的编码

属性

名称	功能说明	脚本例子
verbose	事件通知：当触发条件满足时，ExpOS主动向device.com0发送事件消息。仅当选择device.com0通讯协议为ScriptMode时适用。	device.keypad.verbose=1 设置对象触发的事件有效时向device.com0发送事件消息
enabled	键盘使能：如果为false，屏蔽键盘事件，默认为true	device.keypad.enabled=0 禁止矩阵键盘

方法

名称	功能说明	脚本例子
code()	返回最近触发事件的键编码	var c = device.keypad.code() 读取最近触发事件的键编码

提示：

如果需要将矩阵键盘的按键与屏上的界面按钮关联，可在矩阵键盘的事件处理函数中调用界面按钮的simulateTouch()方法，如：调用ui.form-name.button-name.simulateTouch(1) 将让按钮处于”按下”状态，效果同手指按下触摸屏的按钮位置时一致，按钮的显示状态和对应的按钮脚本会执行。

8.2.8.LED-led

描述

板载LED灯，父对象为device，无子对象

事件**:** 无

属性

名称	功能说明	脚本例子
blinkInterval	闪烁间隔，单位：ms	device.led.blinkInterval=500 设置LED闪烁间隔500ms
state	LED的状态：0-Off，1-On，2-Blink	device.led.state=2 设置LED为闪烁状态

方法

名称	功能说明	脚本例子
isSupported()	检测硬件是否支持LED	device.led.isSupported();

8.2.9.实时时钟-rtc

描述

实时时钟，对应硬件的RTC 芯片，由纽扣电池供电，即使系统掉电，该芯片仍继续工作。父对象为device，无子对象。

事件

无

属性

名称	功能说明	脚本例子
ntpEnabled	使能NTP网络时间同步服务，服务器地址由ntpServer属性指定	device.rtc.ntpEnabled=1 使能网络时间同步
ntpServer	NTP服务器地址	device.rtc.ntpServer=’cn.ntp.org.cn’ 设定网络时间同步的NTP服务器地址位cn.ntp.org.cn

timezone	时区，默认为0-UTC	device.rtc.timezone=8 设置时区为东8区，北京时间

方法

名称	功能说明	脚本例子
getTime() getTime(format)	读取当前时间 无参数时返回时间字符串格式为 hh:mm:ss, 如：12:01:03；带一个参数时，参数为ms，返回自epoch(1970-1-1 00:00:00)到当前时间的毫秒数，一般用于测量时间，如：可在脚本中调用两次该函数，测量两次读取之间的时间间隔，精度为毫秒。参数为hh-mm 返回如 12:01	var time=device.rtc.getTime() 读取当前时间，并存入变量time var ms=device.rtc.getTime('ms') 读取自epoch到当前时间的毫秒数
setTime(time)	设置时间，参数time为时间字符串，时间格式： hh:mm:ss	device.rtc.setTime('01:02:00′) 调整时钟时间为1点2分0秒
getDate() getDate(format)	读取当前日期 无参数时返回固定格式：yyyy-MM-dd， 如：2018-01-20 有参数时返回自定义格式，如 yy-MM-dd, 返回 18-01-20	var date=device.rtc.getDate() 读取当前日期，并存入变量date
setDate(date)	设置日期，参数date为日期字符串，时间格式： yyyy-MM-dd	device.rtc.setDate('2018-09-20') 调整日期为2018年9月20日
getDayOfWeek()	获取星期几，1-星期一 ~ 7-星期日	var day = device.rtc.getDayOfWeek() 获取星期几
addYear(count)	加/减年，参数count为正表示加，为负表示减	device.rtc.addYear(1) 年加1
addMonth(count)	加/减月，参数count为正表示加，为负表示减	device.rtc.addMonth(1) 月加1
addDay(count)	加/减日，参数count为正表示加，为负表示减	device.rtc.addDay(1) 日加1
addHour(count)	加/减时，参数count为正表示加，为负表示减	device.rtc.addHour(1) 小时加1
addMinute(count)	加/减分，参数count为正表示加，为负表示减	device.rtc.addMinute(1) 分钟加1
addSecond(count)	加/减秒，参数count为正表示加，为负表示减	device.rtc.addSecond(1) 秒数加1

8.2.10.通用串行总线-usb

描述

通用串行总线USB，父对象为device，无子对象。

事件

onEvent：USB事件发生时触发，事件函数定义：onEvent(classname, event, payload) ，其中classname为USB类名称, event为事件类型，payload为载荷，不同类型的class有不同的定义，见下表：

事件描述	classname	event	payload
U盘插入	mass_storage	mount:ok (U盘挂载成功) mount:fail（U盘挂载失败）	如果U盘挂载成功，payload内容为挂载路径，程序可通过挂载路径访问（格式化，读写，卸载）磁盘
U盘拔出	mass_storage	umount:ok (U盘卸载成功) umount:fail（U盘卸载失败）	payload的内容为卸载路径,类似：/storage/d0

属性

无

方法

名称	功能说明	脚本例子
getStorageCount()	获取当前已经挂载的U盘数量	var udriveCount=device.usb.getStorageCount()
getStoragePath(index)	获取索引号为index的U盘的挂载路径，如果U盘没有挂载，返回invalid，路径名一般为/storage/dX，其中X为索引号。如同时插入两个U盘，分别挂载在/storage/d0, /storage/d1。	var path=device.usb.getStoragePath(0) 获取索引号为0的U盘的挂载路径

8.2.11.以太网-ethernet

描述

以太网，父对象为device，无子对象

事件

onChange： 以太网连接状态发生变化，函数定义：onChange(state), state为当前的网络状态，值为”connected”或者”disconnected”。

属性

名称	功能说明	脚本例子
enabled	使能：如果为true，使能以太网，否则关闭。	device.ethernet.enabled=0 停止以太网
verbose	事件通知：当触发条件满足时，ExpOS主动向串口发送事件消息。仅当选择串口通讯协议为script mode时适用。	device.ethernet.verbose=1 设置对象触发的事件有效时向串口发送事件消息

方法

名称	功能说明	脚本例子
getState()	返回网络状态，值为connected或者disconnected	var state=device.ethernet.getState() 读取网络状态
getIpAddress()	返回网络IP地址	var ip=device.ethernet.getIpAddress() 读取网络IP地址

8.2.12.SD卡-sd

描述

SD卡，父对象为device，无子对象

事件

onEvent： 插拔SD卡时触发，触发函数定义：onEvent(event, path)， event是事件名称，path是SD卡挂载的路径

属性： 无

方法：无

8.2.13.I2C总线-i2c

描述

I2C总线，父对象为device，无子对象

事件

onDetect(addr): 开机检测到I2C总线上的设备时触发，参数addr为设备的地址

onPoll() ：当轮询时间到时触发，无参数

属性

名称	功能说明	脚本例子
poll	使能轮询	device.i2c.poll=true 使能轮询
pollInterval	轮询间隔(单位：毫秒)	device.i2c.pollInterval=500 设置轮询间隔为500毫秒

方法

名称	功能	脚本例子
isSupported()	检查硬件是否支持I2C总线	var supported = device.isSupported();

read(addr, reg, count)	从i2c总线设备地址addr寄存器地址reg上读取count个字节(如果设备没有寄存器地址，则reg传-1), 如果寄存器地址是多字节（如16位），参数reg传数组（如 [0x11, 0x22]）
	注意：read返回值是个字节数组
		var data = device.i2c.read(0x50, 0x11, 2); //从i2c总线地址0x50的设备上对寄存器地址0x11读取2个字节(寄存器地址为8位)
		var data = device.i2c.read(0x50, [0x11,0x22], 2); //从i2c总线地址0x50的设备上对寄存器地址0x1122读取2个字节(寄存器地址为16位)
		var data = device.i2c.read(0x68, -1, 3); //从i2c总线地址0x68上读取3个字节（无寄存器地址）

write(addr, data)	写数据到地址为addr的i2c总线设备，data为字节数组 (如果设备有寄存器地址为8位，那么数组data的第1个字节即为寄存器地址，如果为16位，则data的前两个字节为寄存器地址)	device.i2c.write(0x20, [0x10, 0x25]); // 向i2c总线地址为0x20的设备写入2个字节0x10, 0x25（无寄存器地址）
		var value = [0x55, 0x66, 0x77];
		device.i2c.write(0x20, value); // 向i2c总线地址为0x20的设备写入3个字节, 如果寄存器地址为8位，则写入的有效数据为0x66, 0x77;如果寄存器地址为16位，则写入的有效数据为0x77

8.2.14.脉宽调制-pwm

描述

脉宽调制，父对象为device，无子对象

事件

---

onStop： 发送脉宽停止时触发，触发函数定义：onStop(pin)， pin是pwm的序号（总共有两路pwm）

属性

名称	功能说明	脚本例子
p0Period	p0周期（单位：微秒）	device.pwm.p0Perod=1000 设置p0的周期为1毫秒
p0DutyCycle	p0占空比(%)	device.pwm.p0DutyCycle=50 设置p0的占空比为50%
p0Polarity	p0极性, 值为 0-Normal, 1-Reversed	device.pwm.p0Polarity=1 设置p0的极性为反转
p1Period	p1周期（单位：微秒）	device.pwm.p1Perod=1000 设置p1的周期为1毫秒
p1DutyCycle	p1占空比(%)	device.pwm.p1DutyCycle=50 设置p1的占空比为50%
p1Polarity	p1极性, 值为 0-Normal, 1-Reversed	device.pwm.p1Polarity=1 设置p1的极性为反转

方法

名称	功能说明	脚本例子
isSupported()	检测硬件是否支持PWM	var supported = device.pwm.isSupported;
play(pin)	启动, 参数pin为第几路，值为0或1	device.pwm.play(1); 启动p1
play(pin, ms)	延时启动pwm, 参数pin为第几路，值为0或1，参数ms为延时毫秒数	device.pwm.play(1， 100); 延时100毫秒后启动p1
stop(pin)	停止，参数pin为第几路，值为0或1	device.pwm.stop(1); 停止p1

8.2.15.串行外设接口-spi

描述

串行外设接口，父对象为device，无子对象

事件：无

属性

名称	功能说明	脚本例子
channel	通道序号，总共两个通道，值为0或1	device.spi.channel=0 设置通道号为0

方法

名称	功能说明	脚本例子
isSupported()	检测硬件是否支持ADC	var supported = device.spi.isSupported();
xfer(tx_buf, rx_size)	传送, 参数tx_buf是发送的数据缓存(可以是整型或者整型数组), 参数rx_size是接收的数据长度	var data = device.spi.xfer([0,0], 2) 发送两个字节长度数据[0,0]，传送完成后spi返回两个字节数据到data数组

8.3.服务-service

描述

服务，根对象。

事件

无

属性

无

方法

名称	功能说明	脚本例子
setTimeout	延迟函数的执行，定义为setTimeout(func-name, ms), 其中func-name为JS函数的名称（该函数必须定义为无参数），ms为延迟时间，单位毫秒	service.setTimeout(‘test’, 500) 设置500ms后test函数被调用执行 方法名test不加引号也可以，如： service.setTimeout(test, 500) 设置500ms后test函数被调用执行
setLanguage(index)	设置语言环境，所有控件的text属性自动切换到指定语言字符串。index为语言的索引值，在Studio中可设置支持语言的种类数，index=0表示第一种语言，每个有text属性的控件可单独设置不同语言下的字符串内容。	service.setLanguage(0) 切换到语言0
getLanguage()	读取当前语言环境的索引值	var lang = service.getLanguage() 读取当前语言环境索引值并写入变量lang

8.3.1.输入法-inputmethod

描述

输入法，父对象为service，无子对象。

事件：无

属性

名称	功能说明	脚本例子
chineseinput	中文输入法：如果为true， 支持中文输入	无

方法：无

8.3.2.系统变量-variable

描述

系统变量，用于存储系统变量值，掉电后不丢失。系统变量类似于map<key, value>数据结构，包括名称(key)和内容(value). 父对象为service，无子对象。

事件：无

属性：无

方法

名称	功能说明	脚本例子
write(name, value)	存储系统变量为字符串，参数name为系统变量名称，value为值	service.variable.write(‘test’, 12) 存储名称为test的系统变量，值为12
read(name)	读取系统变量，参数name为系统变量名称，返回类型为字符串	var value=service.variable.read(“test”) 读取系统变量test，返回字符串’12’
writeInt(name, value)	存储系统变量为整形，参数name为系统变量名称，value为值	service.variable.writeInt(‘test, 12) 存储名称为test的系统变量，值为12
readInt(name)	读取系统变量，参数name为系统变量名称，返回类型为整形	var value=service.variable.readInt(‘test’) 读取系统变量test的值，返回整形12
writeIntArray(name, value)	存储系统变量为整形数组，参数name为系统变量名称，value为数组	service.variable.writeIntArray(‘test, [11, 22, 33]) 存储名称为test的系统变量，值为整形数组
readIntArray(name)	读取系统变量，参数name为系统变量名称，返回类型为整形数组	var value=service.variable.readIntArray(‘test’) 读取系统变量test的值，返回整形数组[11, 22, 33]
writeStrArray(name, value)	存储系统变量为字符串数组，参数name为系统变量名称，value为数组	service.variable.writeStrArray(‘test, [‘aa’, ‘bb’, ‘cc’]) 存储名称为test的系统变量，值为字符串数组
readStrArray(name)	读取系统变量，参数name为系统变量名称，返回类型为字符串数组	var value=service.variable.readStrArray(‘test’) 读取系统变量test的值，返回字符串数组[‘aa’, ‘bb’, ‘cc’]
contains(key)	返回是否已存在指定的名称系统变量	if (service.variable.contains(“key”)) … 检查系统中是否已存在名称为key的系统变量
size()	返回系统中已存在的系统变量总数， 无参数	var num = service.variable.size() 读取已存入系统中的系统变量个数并存入变量num中
allKeys()	返回所有已存在的系统变量名称, 无参数	var a = new Array(service.variable.size()) a = service.variable.allKeys() 读取所有系统变量的名称，并存入数组a中
remove(…)	删除系统变量，参数可为0(删除所有)，一个或者一个以上	service.variable.remove() 删除所有系统变量 service.variable.remove(“test”) 删除名称为test的系统变量 service.variable.remove(“test1”, “test2”) 删除名称为test1和test2的系统变量
save()	强制存储系统变量，无参数。即使不调用此函数，系统会自动存储，调用此方法强制系统立刻存储.	service.variable.save() 强制系统立刻存储所有系统变量
backup(filepath)	备份所有系统变量到filepath指定的文件	service.variable.backup(“/storage/d0/bak.ini”) 备份所有系统变量到文件/storage/d0/bak.ini
restore(filepath)	从filepath指定的文件恢复所有系统变量	service.variable.restore(“/storage/d0/bak.ini”) 从文件/storage/d0/bak.ini中恢复所有系统变量

8.3.3.存储-storage

描述

存储，父对象为service，无子对象

事件：无

属性：无

方法

名称	功能说明	脚本例子
formatExt4(path)	格式化path路径的U盘格式为Ext4	service.storage.formatExt4(‘/storage/d0’) 格式化path路径为Ext4格式
getFreeSize(path)	获取path路径所在分区剩余空间大小（单位为Byte）	var size = service.storage.getFreeSize(‘/storage/d0’) 返回路径所在分区的剩余空间大小
getFreeSizeInMB(path)	获取path路径所在分区剩余空间大小（单位为MB）	var size = service.storage.getFreeSizeInMB(‘/storage/d0’) 返回路径所在分区的剩余空间大小MB
getFsType(path)	获取path路径的分区格式, 返回值为ext4, vfat，或者invalid	var type = service.storage.getFsType(‘/storage/d0’) 返回路径所在分区的格式
getSize(path)	获取path路径的空间大小（单位为Byte）	var size = service.storage.getSize(‘/storage/d0’) 返回路径所在分区的空间大小
getSizeInMB(path)	获取path路径的空间大小（单位为MB）	var size = service.storage.getSizeInMB(‘/storage/d0’) 返回路径所在分区的空间大小MB
listContents(path)	获取path路径下的所有文件名称，以空格分隔的字符串，如 “1.txt 2.txt 3.txt”	var list = service.storage.listContents(‘/storage/d0’) 返回路径下所有文件列表
mount(path)	挂载路径为path的U盘	service.storage.mount(‘/storage/d0’) 挂载路径
umount(path)	卸载路径为path的U盘	service.storage.umount(‘/storage/d0’) 卸载路径

8.4.工具-util

描述

工具，根对象。

事件：无

属性：无

方法

名称	功能说明	脚本例子
getOsVersion()	读取当前ExpOS版本信息	var ver = util.getOsVersion() 读取当前系统版本信息
getModelName()	读取设置模块名，模拟器里返回”Simulator”	var model = util.getModelName() 读取模块名
getDeviceId()	读取设备标识(20个字符)，模拟器返回以EXPOS开头的20个字符	var id = util.getDeviceId() 读取设备标识
jsonToJsObject(json)	转换JSON字符串为JS对象	var obj = util.jsonToJsObject(json) 转换json为JS对象
objectToJson(obj)	转换JS对象为JSON字符串	var json = util.objectToJson(obj) 转换JS对象为json
calculateLRC(data)	计算数组data整个数数据的LRC校验值	var data = [0xaa, 0xbb, 0xcc, 0xdd]; var lrc = util.calculate(data);
calculateLRC(data, len)	计算数组data长度为len的数据LRC校验值	var data = [0xaa, 0xbb, 0xcc, 0xdd]; var lrc = util.calculate(data, 3);
calculateCRC16(data)	计算数组data整个数数据的CRC-16校验值	var data = [0xaa, 0xbb, 0xcc, 0xdd]; var crc = util.calculateCRC16(data);
calculateCRC16(data, len)	计算数组data长度为len的数据CRC-16校验值	var data = [0xaa, 0xbb, 0xcc, 0xdd]; var crc = util.calculateCRC16(data, 3);
arrayToHexString(data)	数组转换成16进制字符串（空格分隔）	var data = [0xaa, 0xbb, 0xcc, 0xdd]; var str = util.arrayToHexString(data); // ‘aa bb cc dd’
arrayToUtf8(data)	数组转换成UTF-8格式	var data = [0x61, 0x62, 0x63]; var str = util.arrayToUtf8(data); // ‘abc’
arrayToUtf8(data, encoding)	数组从encoding编码方式转换成UTF-8编码	var data = [0xd6, 0xd0]; util.arrayToUtf8(data, ‘gb2312’); // ‘中’
arrayToFloat(data)	数组（大端模式）转换成浮点数	var data = [0x40, 0x48, 0xf5, 0xc3]; var f = util.arrayToFloat(data); // 3.14
arrayToFloat(data, isBigEndian)	数组转换成浮点数，isBigEndian为true表示大端	var data = [0xc3, 0xf5, 0x48, 0x40]; var f = util.arrayToFloat(data, false); // 3.14

8.4.1.控制台-console

描述

控制台，系统的控制终端，可输出信息和执行命令，无子对象。

事件：无

属性：无

方法

名称	功能说明	脚本例子
log()	输出调试信息到系统日志终端，用于调试APP，可输出变量，固定字符等调试日志信息。 日志输出： 当开发主机与目标系统通过USB连接后，打开Studio中的调试器即可看到日志终端的输出。 日志格式： 例如看到如下信息： Jan 1 10:06:21 [202.577] A ui.main.timer: counter=89 中括号中的值为时间戳，单位为秒，即打印该信息时系统已启动了202.577秒。通过不同的时间戳差值可测量运行时间，单位精确到毫秒; – A 表示该条信息为应用程序输出 – ui.main.timer表示输出信息的对象名称 – counter=89： 在ui.main.timer对象的动作脚本里调用了类似util.console.log(‘counter=’ + counter), 当对应的事件发生触发脚本执行，就会看到该字符串输出	util.console.log(‘test’) 在调试日志终端输出调试信息test
execute()	执行系统命令，目前只支持: reboot: 重启系统 reboot-os：重启操作系统 shutdown：关闭系统	util.console.execute(‘reboot’) 重启系统