Greatal

GoUI软件速查表 V1.28

说明：
属性——控件的内部变量，可读写，通过改变其值，达到更新控件外观的目的；
方法——控件的内部函数，在界面运行时，通过调用该函数，改变或者读取控件行为或状态；
事件——控件的内部事件类型，当事件发生时，界面可根据需要自动完成：1) 发送事件消息通知外部控制单元（只适用于CTP通讯模式）; 2) 运行控件动作脚本；

►表一： 用户控件

控件名称	属性 Property	方法 Method	事件 Event	CTP事件消息例子 PS-LCD自动发出，外部控制单元接收 只适用于CTP通讯模式 （须使能控件"事件通知"属性）	JS脚本例子 在任何通讯模式下控件动作脚本中输入、 或者在CTP通讯模式下由外部控制单元发出	备注
页面 Form	source	show() hide()	loaded raised	**E+xxx.loaded=1** 页面xxx被加载(对于"快速显示"属性选中的页面，系统启动时,才会有此事件；否则,任何时候调用sysManager.show("xxx")时，都会产生该事件) **E+xxx.raised=1** 页面xxx被切换到前台(页面的"快速显示"属性必须选中，而且系统启动完毕, 调用sysManager.show("xxx")时，才会产生该事件)	**xxx.source='test.jpg';** 设定页面xxx的背景为test.jpg图片 (请确保spf/picture目录下存在test.jpg文件) **xxx.show();** 切换到xxx页面 **xxx.show("fade-in");** 采用逐渐变暗效果切换到页面xxx **xxx.hide();** 隐藏xxx页面	快速显示属性选中的页面，系统启动后，始终位于内存中，可提高切换速度； **页面名字必须为英文字母**
标签 Label	text	clear()	textChanged	**E+xxx.text = '5678'** 标签xxx的文本内容改为5678	**xxx.text="test";** # 设定标签xxx的文本内容为test **xxx.clear();** # 清空标签xxx的文本内容	
文本按钮 Button	text flat checked	setCheckable(bool) setChecked(bool) animateClick(int)	pressed held released checked unchecked	**E+xxx.state=0** 按钮xxx处于按下状态 **E+xxx.state=1** 按钮xxx处于抬起状态 **E+xxx.state=2** 按钮xxx处于保持按下状态（只有"自动重复"属性选中才有效） **E+xxx.state=3** 按钮xxx处于保持锁定状态（只有"可锁定的"属性选中才有效） **E+xxx.state=4** 按钮xxx处于保持未锁定状态（只有"可锁定的"属性选中才有效）	**xxx.text="按钮1";** # 设定按钮xxx的显示文字为"按钮1" **xxx.setChecked(1);** # 设置按钮xxx处于按下状态 **xxx.animateClick(100);** #执行一次点击按钮xxx操作，100ms后按钮状态自动恢复 **xxx.flat=1;** #设置按钮xxx为平坦风格 **if (xxx.currentTrigger() == "pressed")** **{....}** **else** **{.....}** #一般用在按钮的动作脚本中，根据按钮的不同状态（按下或抬起），执行不同操作；	按钮可设置为普通按钮和自锁按钮
图片按钮 ImageButton	checked				**xxx.setChecked(1);** # 设置按钮xxx处于按下状态 **xxx.animateClick(100);** #执行一次点击按钮xxx操作，100ms后按钮状态自动恢复 **if (xxx.currentTrigger() == "pressed")** **{....}** **else** **{.....}** #一般用在按钮的动作脚本中，根据按钮的不同状态（按下或抬起），执行不同操作；	
单行文本框 LineEdit （多行文本框 Multi-line Edit）	text, frame cursorPosition readOnly	clear(), selectAll(), cut(), copy(), paste()	textChanged editingFinished editingCanceled	**E+xxx.text = '5678'** 文本框xxx的文本内容改为5678 如果主控制器发送**xxx.cursorPosition?**字符串，LCD将返回 **Q+xxx.cursorPosition=2**表示当前光标位置在第二个字符	**xxx.text = '1234';** #设置文本框xxx的文本内容为1234 **xxx.copy();** # 复制文本框xxx中选定的文本内容 **xxx.paste();** # 粘帖复制的内容到文本框xxx **xxx.frame=1;** # 设置文本框有边框 **xxx.readOnly=1;** # 设置文本框为只读, 点击该文本框,软键盘将不再自动弹出	cursorPosition为只读
滑动尺 Slider	value, maximum, minimum	N/A	valueChanged	**E+xxx.value= 50** 滑动块xxx的当前位置更新为50	**xxx.value=20;** # 设定滑动块xxx的当前位置为20 **xxx.maximum=100;** # 设定滑动块位置的最大值为100	
图片 Image	source scaledContents speed	play(int) pause(bool) stop() frameNumber()	sourceChanged frameChanged loopDone	N/A	**xxx.source='1.gif';** 播放1.gif动画文件 **xxx.source='test.jpg';** 显示test.jpg图片文件 **xxx.scaledContents=1;** 拉伸图片适合控件尺寸 **xxx.speed=100;** gif图片动画播放速度为原速度的100% **xxx.speed=200;** gif图片动画播放速度为原速度的200%，即原始速度两倍 **xxx.play(0)** 播放gif动画，直到调用xxx.stop()后停止 **xxx.play(1)** 播放gif动画1次 **xxx.pause(1)** 暂停gif动画播放 **xxx.pause(0)** 恢复gif动画播放 **var i=xxx.frameNumber()** 获取当前播放的gif动画帧号	支持的图片格式:bmp, jpg, png, tiff, gif等，图片必须位于spf/picture目录下
进度条 Progress Bar	value, format	reset(), setRange(int, int)	valueChanged	**E+xxx.value = 10;** 进度条xxx位置更新为10	**xxx.setRange(0, 100);** #设定进度条xxx的位置范围0-100 **xxx.value = 80;** # 设定进度条xxx的进度为80% **xxx.format = "%p"** # 设定进度条xxx的文字显示百分比 **xxx.format = "%v"** # 设定进度条xxx的文字显示实际值	
复选框 Check Box	checked	N/A	checked unchecked	**E+xxx.checked=0** 单选框xxx选择被取消 **E+xxx.checked=1** 单选框xxx被选中	**xxx.checked = 1;** # 选择单选框xxx **xxx.checked = 0;** # 取消选择单选框xxx	

下拉菜单 Combo Box	currentText, currentIndex, count, editable	addOption(string, int) removeOption(int)	indexChanged	**E+xxx.currentIndex = 1** 下拉菜单xxx当前选中索引1	**xxx.currentIndex=2;** # 设定下拉菜单xxx当前索引值为2 **xxx.addOption("test", 2);** # 在下拉菜单xxx索引2处增加test文本选项 **xxx.removeOption(2);** # 删除下拉菜单xxx索引值为2的选项	
定时器 Timer	interval	run(int) stop()	timeout	**E+xxx.timeout=1** 定时器xxx到时	**xxx.interval=500;** 设定定时器xxx的时间间隔为500ms **xxx.run(0);** 启动定时器xxx,自动重复定时和执行动作脚本，直到调用xxx.stop()停止 **xxx.run(2);** 启动定时器xxx, 自动重复定时和执行动作脚本2次或者调用xxx.stop()后停止 **xxx.stop();** 停止定时器xxx	interval值建议大于50ms
段式数字 LCD Number	value	NA	valueChanged	**E+xxx.value=yy** 段式数字数值变化为yy	**xxx.value=10;** 显示数字10	
日期时间 Clock	time date alarm	NA	alarmed	**E+xxx.alarmed=1** 闹钟定时到	**xxx.time="10:23:15"** 设定当前时间为 10:23:15 **xxx.date="2012-12-01"** 设定当前日期为2012年12月1号 **xxx.alarm="10:30:00"** 设定闹钟在10:30:00执行动作脚本	该控件为软时钟，每次上电需要初始化为正确时间
仪表盘 Gauge	value	NA	valueChanged	**E+xxx.value=yy** 仪表盘xx数值变化为yy	**xxx.value=10;** 仪表盘指针指到10%量程处	
刻度尺 Thermo	value	NA	valueChanged	**E+xxx.value=yy** 刻度尺xx数值变化为yy	**xxx.value=10;** 刻度尺指针指到10刻度处	
波形 Plot	value xAxisScaleMin xAxisScaleMax yAxisScaleMin yAxisScaleMax	NA	NA	NA	**xxx.value = '10, 20, 30';** 刷新三条曲线的Y坐标分别为10、20、30， X坐标自动加一 **xxx.xAxisScaleMin = 10;** 设定波形x轴的坐标最小值为10 **xxx.xAxisScaleMax = 100;** 设定波形x轴的坐标最小值为100	
画布 Canvas	NA	createObject() deleteObject()	loaded	**cc.createObject("mypoint", "point", 100, 100);** 在坐标(100, 100)处创建一个名字为mypoint的点对象 **cc.deleteObject("mypoint");** 删除名字为mypoint的点对象		支持的图形类型：点（piont）、线（line）、椭圆(ellipse)、多段线(polyline)、矩形(rect)
		setObjectPalette()		**cc.setObjectPalette("mypoint", "pen", "#FFFF00", "solid", 2);** 设置名字为mypoint的点对象的绘图操作调色板属性：颜色为RGB十六进制表示0xFFFF00的颜色，线条风格为solid，线条宽度为2个像素 **cc.setObjectPalette("mypoint", "brush", "#FFFF00", "solid");** 设置名字为mypoint的点对象的填充操作调色板属性：颜色为RGB十六进制表示0xFFFF00的颜色，线条风格为solid	
		setObjectExtention() extendObject()		**cc.setObjectExtention("mypoint", "after");** 设置名字为mypoint的对象的扩展方式为：后面扩展（扩展的绘图操作在原对象的右边进行） **cc.setObjectExtention("mypoint", "before");** 设置名字为mypoint的对象的扩展方式为：前面扩展（扩展的绘图操作在原对象的左边进行） **cc.extendObject("mypoint", 10, 20);** 将对象mypoint扩展一个新元素（如果mypoint为一个点对象，则会在坐标（10，20）处增加一个点，这个点的位置按setObjectExtention的设置，此时的两个点同属mypoint对象） **cc.extendObject("myline", 10, 10, 80, 80);** 将对象myline扩展一个新元素（如果myline为一个线对象，则会在坐标（10，10）到（80，80）间绘制一条直线，这条直线的位置按setObjectExtention的设置，此时的两条直线同属myline对象）	
		setObjectMoving() moveObject()		**cc.setObjectMoving("mypoint", "relative");** 设置mypoint对象的移动方式：相对移动 **cc.setObjectMoving("mypoint", "absolute");** 设置mypoint对象的移动方式：按绝对坐标移动 **cc.moveObject("mypoint", 1, 0);** 将mypoint对象向右移动一个像素，向上不移动（假设setObjectMoving设置的移动方式为相对移动） **cc.moveObject("mypoint", -1, 0);** 将mypoint对象向左移动一个像素，向上不移动（假设setObjectMoving设置的移动方式为相对移动）	
		clearObject() setActiveRegion()		**cc.clearObject("mypoint");** 清除名字为mypoint点对象内容 **cc.setActiveRegion(5, 5, 100, 100);** 设置画布cc的刷新区域为坐标（5，5）到（100，100）间的矩形区域,不在此区域的对象，执行extend, move操作，看不到屏幕有变化。设置合理的刷新区域能有效优化系统刷新速度。	

通用属性 适合所有用户控件	enabled visible x y width height verbose actionTrigger	NA	NA	**xxx.visible=0** 隐藏控件xxx **xxx.visible=1** 显示控件xxx **xxx.enabled=1** 使能控件xxx **xxx.enabled=0** 禁止控件xxx **xxx.verbose=1** 设置控件xxx的事件通知有效 **xxx.actionTrigger="yyy"** 设置控件xxx的触发条件为yyy（各控件支持的触发条件不一样，请参考各控件事件列）	x y width height属性为只读，写无效
通用方法 适合所有用户控件	NA	setFocus() move(int, int) resize(int, int) currentTrigger()	NA	**xxx.setFocus();** 让控件xxx获得焦点 **xxx.move(100, 200);** 移动控件到新位置（坐标x=100, y=200） **xxx.resize(100, 200);** 设置控件xxx宽度为100，长度为200 **var i=xxx.currentTrigger();** 读取xxx控件当前脚本触发事件（各控件支持的触发条件不一样，请参考各控件事件列）	

<table>
<tr><td rowspan="20">风格属性
适合所有用户控件</td><td rowspan="20">styleSheet</td><td rowspan="20">NA</td><td rowspan="20">NA</td><td colspan="2">styleSheet(样式表)的格式定义如下： "属性名称1：值; 属性名称2：值；....."
样式表无事件消息，可用的属性名称和值的定义如下：</td><td>各属性设置以分号隔开，属性个数不限</td></tr>
<tr><td>属性名称</td><td>可用值</td><td></td></tr>
<tr><td>color (字体颜色)</td><td>red(红色) | green(绿色) | rgb(10, 20, 255) 等等</td><td></td></tr>
<tr><td>background-color (背景色)</td><td>同上</td><td></td></tr>
<tr><td>selection-color (选中部分前景色)</td><td>同上</td><td></td></tr>
<tr><td>selection-background-color
(选中部分背景色)</td><td>同上</td><td></td></tr>
<tr><td>border-color (边框颜色)</td><td>同上</td><td></td></tr>
<tr><td>border-style (边框风格)</td><td>dashed | dot-dash | dot-dot-dash | dotted | double | groove | inset | outset | ridge | solid | none</td><td></td></tr>
<tr><td>border-width (边框宽度)</td><td>1px (一个像素) | 2px (两个像素) 等等</td><td></td></tr>
<tr><td>border-radius: (边框弧度)</td><td>同上</td><td></td></tr>
<tr><td>margin (外边框)</td><td>同上</td><td></td></tr>
<tr><td>padding （内边框）</td><td>同上</td><td></td></tr>
<tr><td>icon-size (图标大小)</td><td>同上</td><td>icon-size: 16px 表示图标大小为16x16</td></tr>
<tr><td>font-size (字体大小)</td><td>同上</td><td></td></tr>
<tr><td>font-weight (字体粗度)</td><td>normal (正常) | bold (加粗) 100 - 900</td><td></td></tr>
<tr><td>font-style (字体风格)</td><td>normal (正常) | italic (斜体)</td><td></td></tr>
<tr><td>text-transform (文本转换)</td><td>none | uppercase (大写) | lowercase (小写)</td><td></td></tr>
<tr><td>text-align (文本对齐)</td><td>top | bottom | left | right | center</td><td>只对按钮和进度条有效</td></tr>
<tr><td>text-decoration (文本装饰)</td><td>overline | underline | line-through | blink</td><td></td></tr>
<tr><td colspan="3">xxx.styleSheet = "color: green; background-color: black";
# 设置控件xxx的字体颜色为绿，背景色为黑
xxx.styleSheet = "color: rgb(100, 200, 100)";
# 设置控件xxx的字体颜色为 RGB的组合色，红色100，绿色200， 蓝色100 （每种颜色的最大值为255）
xxx.styleSheet = "icon-size: 16px; font-size: 20px; margin: 1px 2px 3px 4px";
# 设置控件xxx的图标尺寸为16x16像素, 字体大小为20像素, 上、右、下、左空白区分别占1、2、3、4个像素
xxx.styleSheet="border-style: inset; border-width: 5px ; border-color: red;";
# 设置控件xxx的风格为inset, 外框宽5个像素， 外框颜色为红
xxx.styleSheet="*::indicator { width: 13px; height: 130px}";
# 设置复选框xxx的前框宽度为13像素，高度为130像素</td><td></td></tr>
</table>

注：xxx为控件的ID，由用户在designer设计界面时指定

►表二： 系统控件 (PS-LCD固有控件，可在任何界面和控件中访问)

控件名称	属性 Property	方法 Method	事件 Event	JS脚本例子 在任何通讯模式下控件动作脚本中输入，或者在CTP通讯模式下由外部控制单元发出	备注
 **串口0 sysCom0**	ctpReply	write(char, ...) writeString(string, ...) read(int) readableBytes() clearReadBuffer()	received sent	**sysCom0.write(0x55)** 写串口发送缓冲区，串口向外部发送一字节数据 0x55 **sysCom0.write(0x55, 16, 'A')** 写串口发送缓冲区，串口向外部发送三字节数据 0x55, 0x10, 0x41 **sysCom0.writeString("12AB")** 写串口发送缓冲区，串口向外部发送字符串"12AB", 即4字节数据 0x31, 0x32, 0x41, 0x4B **var a = new Array(2);** **a = sysCom0.read(2)** 读取串口接收缓冲区，返回2字节数据存储在数组a中，未读出的数据仍保留在接收缓冲区 **var a = new Array(sysCom0.readableBytes());** **a = sysCom0.read(0)** 读取串口接收缓冲区所有数据，返回数据按字节存储在数组a中，读取完成后，接收缓冲区为空 **var size = sysCom0.readableBytes()** 读取接收缓冲区未读出数据字节数，每调用一次read(), 未读出数据字节数自动减少；每收到一个字节串口数据，未读出字节数自动加一 **sysCom0.clearReadBuffer()** 清空接收缓冲区数据	只能选择其中一种协议模式：CTP或者自定义. 在CTP协议下，当启动完毕后，PS-LCD将自动发出如下消息 **E+sys.state=ready**（后有回车字符作为结束符）
 **界面管理器 sysManager**	debugMode cursorVisibility formAnimation	messageBox(string) execute(string) currentForm()	NA	**sysManager.messageBox("Hello world")** 弹出消息框,内容为：Hello world **sysManager.debugMode=1** 打开调试功能，脚本出错立即提示 **sysManager.cursorVisibility=0** 设置光标不可见 **sysManager.formAnimation=1** 设在加载新页面时显示切换进度动画 **sysManager.execute("reboot");** 重新启动PS-LCD **sysManager.execute("shutdown");** 关闭PS-LCD(需重新启动后，才能显示界面) **var f = sysManager.currentForm();** 读取当前显示的页面名称到变量f	
 **矩阵键盘 sysHardKeypad**	debounceTime	code()	NA	**var key = sysHardKeypad.code()** 读取当前触发键盘事件（按下或者抬起）的按键键值 （键值范围0 -15） (一般用在矩阵键盘动作脚本中识别按键，键位置和键值对应关系参考软件开发文档系统控件章节） **sysHardKeypad.debounceTime=30;** 设置外接矩阵键盘的消抖时间为30ms	PS-LCD支持外接矩阵键盘（最大4x4）
 **软键盘 sysSoftKeypad**	defaultType	show() hide()	NA	**sysSoftKeypad.defaultType="letter"** 默认输入法为英文字母 （letter, number和pinyin三种可选） **sysSoftKeypad.show();** 打开软键盘输入 **sysSoftKeypad.hide();** 关闭软键盘输入	
 **触摸屏 sysTouch**	idleTimeout powerState enabled	calibrate()	idleTimeout wakenUp	**sysTouch.idleTimeout = 1000** 设置触摸屏空闲超时为1000ms, 1000ms内没有触摸事件，屏幕自动进入待机模式 **sysTouch.powerState="suspend";** 设置触摸屏进入待机模式（支持两种模式：待机suspend和激活active模式） **sysTouch.enabled = 0** 关闭触摸功能（关闭后，点击触摸屏不再响应，sysTouch.enabled = 1开启） **var I = sysTouch.powerState;** 读取触摸屏的操作模式，返回字符串为"suspend"或者"active" **sysTouch.calibrate()** 进入触摸屏校准画面	
 **背光 sysBacklight**	brightness	NA	NA	**sysBacklight.brightness = 5** 设置背光亮度为 5 （亮度范围为0——10）	
 **蜂鸣器 sysBuzzer**	interval duration	play(int) stop()	NA	**sysBuzzer.play(2)** 蜂鸣器响两声, 每次时长和间隔由属性duration和interval设定 **sysBuzzer.play(0)** 蜂鸣器保持响, 直到调用sysBuzzer.stop()时停止, 每次时长和间隔由属性duration和interval设定， **sysBuzzer.stop()** 立即停止蜂鸣器发声 **sysBuzzer.interval = 1000** 设定蜂鸣器发声时间隔1000ms **sysBuzzer.duration = 2000** 设定蜂鸣器发声时每次时长2000ms	
 **环境变量 sysVariable**	NA	write(string, string) read(string) remove(string) save()	NA	**sysVariable.write("test", "123")** 把字符串123存到环境变量test中 **Var i = sysVariable.read("test");** 读取环境变量test的值到变量i中 **sysVariable.remove()** 清除所有环境变量 **sysVariable.remove("test");** 清除环境变量test，其他变量保留 **sysVariable.save()** 强制保存所有环境变量	环境变量存储在PS-LCD内部flash中，掉电后内容不会丢失，最大存储容量为3M字节
通用属性 适合所有系统控件	verbose actionTrigger	NA	NA	**xxx.verbose=1** 设置控件xxx的事件通知有效 **xxx.actionTrigger="yyy"** 设置控件xxx的触发条件为yyy（各控件支持的触发条件不一样，请参考各控件事件列）	
通用方法 适合所有系统控件	NA	currentTrigger()	NA	**var i=xxx.currentTrigger();** 读取xxx控件当前脚本触发事件（各控件支持的触发条件不一样，请参考各控件事件列）。例如，如下脚本用在矩阵键盘脚本中判断按键的事件状态，然后执行不同的动作。 **if (sysHardKeypad.currentTrigger() == "pressed")** **…** **else** **…**	

注： 系统控件在系统启动时自动创建，而且ID为固定字符串，用户不可改