

类别	内容
关键词	VisualHMI 用户手册
摘 要	HMI、组态控件

版本记录

版本	日期	修改原因	页面	撰写人	审核人
V1.0	2022/02/25	创建文档	all	林青田	刘启鑫
V1.1	2022/03/17	增加 DCBUS CRC 校验说明		陈鹏	刘启鑫
V1.2	2022/09/6	增加配方、LUA api 函数、 配方、分期、安全密码修改、 绘图控件、资料采集等其他 控件补充		陈鹏	刘启鑫

前言

本教程旨在通过 VisualHMI 平台上开发 HMI 人机交互界面。文章主要介绍大彩 VisualHMI 组态软件的使用方法，详细地介绍软件安装、使用、工程创建和编辑、控件属性、编译、运行、下载等方面，引导用户快速了解 VisualHMI 的开发。此外，本文仅对 VisualHMI 工程重要部分的参数、属性参数作介绍，其余参数说明，请参考软件属性栏中的属性介绍。

广州大彩光电科技有限公司

目录

1. VisualHMI 组态软件.....	1
1.1 VisualHMI 组态软件安装.....	1
1.1.1 安装的环境需求.....	1
1.1.2 软件安装.....	1
1.2 VisualHMI 组态软件界面介绍.....	2
1.2.1 菜单区域.....	2
1.2.2 画面区域.....	3
1.2.3 画面编辑区域.....	3
1.2.4 属性区域.....	4
1.2.5 编译区域.....	4
2. 工程建立与下载.....	5
2.1 新建工程.....	5
2.2 工程设置.....	5
2.2.1 工程属性.....	6
2.2.2 串口设置.....	6
2.2.3 通信协议.....	7
2.2.4 安全登录.....	7
2.3 型号更改.....	7
2.4 工程编译与下载.....	7
2.4.1 工程编译.....	7
2.4.2 串口下载.....	8
2.4.3 SD 卡下载.....	8
3. 组态控件.....	10
3.1 位状态指示灯.....	10
3.1.1 功能设置.....	10
3.1.2 状态设置.....	11
3.2 多状态指示灯.....	11
3.2.1 功能设置.....	11
3.2.2 状态设置.....	12
3.3 位状态按钮.....	12
3.3.1 功能设置.....	12
3.3.2 状态设置.....	13
3.4 多状态按钮.....	13
3.4.1 功能设置.....	13
3.4.2 状态设置.....	14
3.5 功能按钮.....	14
3.5.1 功能设置.....	14
3.5.2 状态设置.....	15
3.6 数值.....	15
3.6.1 功能设置.....	15
3.6.2 外观设置.....	16

3.7 文本.....	17
3.7.1 功能设置.....	17
3.7.2 外观设置.....	17
3.8 滑动条.....	17
3.8.1 功能设置.....	18
3.8.2 外观设置.....	18
3.9 进度条.....	18
3.9.1 功能设置.....	19
3.9.2 外观设置.....	19
3.10 下拉选择.....	19
3.10.1 功能设置.....	20
3.10.2 外观设置.....	20
3.11 滚轮.....	21
3.11.1 功能设置.....	21
3.11.2 外观设置.....	21
3.12 动画控制.....	22
3.12.1 功能设置.....	22
3.12.2 外观设置.....	22
3.13 圆形进度条.....	23
3.13.1 功能设置.....	23
3.13.2 外观设置.....	23
3.14 仪表指针.....	23
3.14.1 功能设置.....	24
3.14.2 外观设置.....	24
3.15 流动块.....	24
3.15.1 功能设置.....	25
3.15.2 外观设置.....	25
3.16 二维码.....	25
3.16.1 功能设置.....	26
3.16.2 外观设置.....	26
3.17 动态绘图.....	26
3.17.1 指令说明.....	27
3.18 RTC.....	27
3.18.1 功能设置.....	28
3.18.2 外观设置.....	28
3.19 钟表.....	28
3.19.1 功能设置.....	29
3.19.2 外观设置.....	29
3.20 历史曲线.....	29
3.20.1 曲线数据.....	30
3.20.2 曲线设置.....	30
3.20.3 通道数量.....	30
3.20.4 控制地址.....	30
3.20.5 X 轴（时间）.....	30

3.20.6 Y 轴（值）	30
3.20.7 外观设置	31
3.20.8 显示触摸标识	31
3.21 数据记录	31
3.21.1 基本设置	31
3.21.2 外观设置	32
3.22 操作记录	32
3.22.1 基本设置	33
3.22.2 外观设置	33
3.22.3 控件操作-使用操作记录	33
3.22.4 控件操作-系统参数标签操作	34
3.23 滑动画面	34
3.23.1 功能设置	34
3.24 嵌入画面	35
3.24.1 功能设置	36
4. 资料采集	37
4.1.1 采样模式	37
4.1.2 记录条数	37
4.1.3 数据地址	37
4.1.4 数据类型	37
4.1.5 小数设置	37
4.1.6 触发地址	38
4.1.7 控制地址	38
4.1.8 采样控制地址	38
4.1.9 掉电存储	38
5. 告警记录	39
5.1 告警显示	39
5.1.1 基本设置	39
5.1.2 外观设置	40
5.2 告警条	41
5.2.1 基本设置	41
5.2.2 外观设置	42
5.3 告警设置	42
6. 安全设置（密码登录）	45
6.1 控件权限	45
6.2 用户等级	45
6.2.1 用户等级设定	45
6.2.2 控件开启用户等级	46
6.2.3 撤销用户登录状态	48
6.2.4 修改安全密码	48
7. 配方设置	50
8. 分期使用	51
8.1 属性介绍	51
8.2 系统参数	52

8.3 修改分期密码.....	53
9. 静态控件.....	55
9.1 直线.....	55
9.2 矩形.....	56
9.3 圆形.....	56
9.4 文字.....	57
9.5 图片.....	57
10. 图库.....	58
10.1 添加.....	58
10.1.1 批量添加.....	58
10.1.2 添加单个图标.....	59
10.2 修改.....	59
10.2.1 增加帧.....	60
10.2.2 替换帧.....	61
10.2.3 删除帧.....	62
11. 字库.....	66
11.1 添加多个字体.....	66
11.2 添加用户字体.....	67
11.3 替换字体.....	68
12. 地址标签.....	69
12.1 系统参数标签.....	69
12.2 用户自定义标签.....	69
12.2.1 添加标签.....	69
12.2.2 删除标签.....	72
12.2.3 标签批量导入.....	72
12.2.4 标签批量导出.....	74
13. 系统变量.....	75
14. 多语言与文字标签.....	79
14.1 多语言.....	79
14.1.1 打开多语言.....	79
14.2 文字标签.....	80
14.2.1 文字标签使用.....	80
14.2.2 文字标签导入.....	82
15. 键盘.....	84
15.1 键盘设置.....	84
15.2 启用键盘输入.....	84
15.3 自定义系统键盘.....	85
15.3.1 替换键盘 UI 风格.....	85
15.3.2 修改键值.....	86
15.3.3 键盘相关系统参数地址.....	87
16. 声音.....	88
16.1 添加音频文件.....	88
16.2 使用音频.....	89
17. 协议说明.....	90

17.1 DCBUS.....	90
17.1.1 写变量存储器指令指令（0xF1）	91
17.1.2 读变量存储器指令指令（0xF2）	91
17.1.3 屏幕修改变量上传主板（0xF2）	92
17.2 XGUS.....	92
17.3 Modbus.....	92
17.3.1 主机模式-ModbusMaster.....	92
17.3.2 主机模式-ModbusSlave.....	93
17.4 三菱-FX2N.....	94
17.4.1 协议说明.....	94
17.4.2 寄存器类型.....	95
17.5 三菱-FX3U.....	95
17.5.1 协议说明.....	95
17.5.2 寄存器类型.....	96
17.6 台达 PLC - DELTA(DVP).....	96
17.6.1 协议说明.....	96
17.6.2 寄存器类型.....	97
17.7 信捷 PLC - XINJIE(XC).....	98
17.7.1 协议说明.....	98
17.7.2 寄存器类型.....	98
17.8 信捷 PLC - XINJIE(XD).....	99
17.8.1 协议说明.....	99
17.8.2 寄存器类型.....	100
17.9 永宏 PLC - FATEK(FB).....	100
17.9.1 协议说明.....	100
17.9.2 寄存器类型.....	101
17.10 海为 PLC - HAIWELL(N/S).....	102
17.10.1 协议说明.....	102
17.10.2 寄存器类型.....	103
17.11 显控-SAMKOON.....	103
17.11.1 协议说明.....	103
17.11.2 寄存器类型.....	104
17.12 艾默生 - EMERSON.....	104
17.12.1 协议说明.....	104
17.12.2 寄存器类型.....	105
18. Lua API.....	107
18.1 常用回调函数.....	107
18.1.1 on_init().....	107
18.1.2 on_run(screen).....	107
18.1.3 on_update(slave, vtype, addr).....	107
18.1.4 on_screen_change(screen).....	108
18.1.5 on_usb_inserted(driver).....	109
18.1.6 on_usb_removed().....	109
18.1.7 on_sd_inserted(dir).....	109

18.1.8 on_sd_removed()	109
18.2 读写寄存器函数	109
18.2.1 set_notify(enable)	109
18.2.2 select_slave(slave_id)	109
18.2.3 set_endian(en)	109
18.2.4 set_bit(vtype, addr, value)	109
18.2.5 get_bit (vtype, addr)	109
18.2.6 set_uint16(vtype, addr, value)	109
18.2.7 get_uint16(vtype, addr)	110
18.2.8 set_int16(vtype, addr, value)	110
18.2.9 get_int16(vtype, addr)	110
18.2.10 set_uint32(vtype, addr, value)	110
18.2.11 get_uint32(vtype, addr,)	110
18.2.12 set_int32(vtype, addr, value)	110
18.2.13 get_int32 (vtype, addr)	110
18.2.14 set_uint64(vtype, addr, value)	110
18.2.15 get_uint64 (vtype, addr)	111
18.2.16 set_int64(vtype, addr, value)	111
18.2.17 get_int64 (vtype, addr)	111
18.2.18 set_float(vtype, addr, value)	111
18.2.19 get_float (vtype, addr)	111
18.2.20 set_double(vtype, addr, value)	111
18.2.21 get_double (vtype, addr)	111
18.2.22 set_string(vtype, addr, strings)	112
18.2.23 get_string (vtype, addr)	112
18.2.24 set_uint16_ex(vtype, addr, value1,value2, ..., value120)	112
18.2.25 set_array(vtype, addr, buff)	112
18.2.26 start_read(index,vtype, addr,quantity)	113
18.2.27 stop_read(index)	113
18.2.28 stop_all_read()	113
18.3 绘图函数	113
18.3.1 on_draw(screen,control)	113
18.3.2 redraw()	113
18.3.3 set_pen_color(color)	113
18.3.4 draw_line(x0,y0,x1,y1,width)	113
18.3.5 draw_rect(x0,y0,x1,y1,fill)	114
18.3.6 draw_rect_alpha(x0,y0,x1,y1,alpha)	114
18.3.7 draw_circle(x,y,r,fill)	114
18.3.8 draw_ellipse(x0,y0,x1,y1,fill)	114
18.3.9 draw_image(image_id,frame_id,dstx,dsty,width,height,srcx,srcy)	114
18.3.10 draw_text(text,x,y,w,h,font_id,size,color,align)	114
18.3.11 load_surface (filename)	115
18.3.12 destroy_surface (surface)	115
18.3.13 destroy_all_surface()	115

18.3.14	draw_surface (surface,dstx,dsty,width,height,srcx,srcy).....	115
18.3.15	get_surface_size (surface).....	115
18.3.16	clear_image_buffer().....	115
18.3.17	video_shoot(filepath, width, height).....	115
18.3.18	screen_shoot (filepath)	116
18.4	数据记录.....	116
18.4.1	record_get_count(sn).....	116
18.4.2	record_modify_data(sn, index, data).....	116
18.4.3	record_write_data(sn, data).....	116
18.4.4	record_read_data(sn, index).....	116
18.4.5	record_modify_string(sn, index, strings).....	117
18.4.6	record_read_string(sn, index).....	117
18.4.7	record_write_string(sn, strings).....	117
18.4.8	record_clear(sn).....	117
18.5	定时器.....	117
18.5.1	start_timer(timer_id, timeout, countdown, repeat).....	117
18.5.2	stop_timer(timer_id).....	117
18.5.3	on_timer(timer_id).....	118
18.5.4	get_timer_value(timer_id).....	118
18.6	告警.....	118
18.6.1	warning_set_mode(en).....	118
18.6.2	warning_set(warning_id,value,count).....	118
18.6.3	on_parse_warning(id, text).....	118
18.7	串口.....	120
18.7.1	uart_setup(ch,baudrate,databits,stopbit,parity).....	120
18.7.2	uart_send(ch,packet).....	120
18.7.3	on_uart_recv(ch,packet).....	120
18.8	音视频.....	120
18.8.1	play_sound(filename).....	120
18.8.2	play_video(file,left,top,width,height).....	120
18.8.3	pause_video().....	121
18.8.4	resume_video().....	121
18.8.5	stop_video().....	121
18.8.6	av_init(show,channel,left,top,width,height).....	121
18.8.7	av_get_status().....	121
18.8.8	av_select(ch).....	121
18.8.9	av_show(show).....	121
18.9	文件读写.....	121
18.9.1	dofile(name).....	121
18.9.2	list_dir(path).....	121
18.9.3	on_list_dir(path,filename,type,fsize).....	122
18.9.4	file_open(path,mode).....	122
18.9.5	file_close().....	122
18.9.6	file_size().....	122

18.9.7 file_seek(offset).....	122
18.9.8 file_read(count).....	122
18.9.9 file_write(data).....	122
18.9.10 file_delete(path).....	122
18.9.11 file_copy(src_path, dst_path).....	122
18.9.12 on_copy_file_process(status,filesize,transfersize).....	122
18.10 设置窗口控件.....	123
18.10.1 set_screen(screen).....	123
18.10.2 get_screen().....	123
18.10.3 show_dialog(screen, x, y, alpha).....	123
18.10.4 wgt_set_pos(screen,control,x,y,w,h).....	123
18.10.5 wgt_set_fcolor(screen,control, color).....	123
18.10.6 wgt_set_bcolor(screen,control, color).....	123
18.10.7 wgt_set_param(screen,control, param,value).....	123
18.10.8 on_wgt_event(screen_id,widget_id,event,value).....	124
18.10.9 refresh_screen().....	124
18.11 GPIO 控制.....	124
18.11.1 gpio_set_in (pin).....	124
18.11.2 gpio_set_out (pin).....	124
18.11.3 gpio_set_value (pin,value).....	124
18.11.4 gpio_get_value (pin).....	124
18.12 其他.....	124
18.12.1 get_platform().....	124
18.12.2 get_version().....	124
18.12.3 get_device_uuid().....	124
18.12.4 reboot().....	124
18.12.5 feed_dog().....	124
18.12.6 delay_ms(ms).....	124
18.12.7 beep(ms).....	124
18.12.8 set_run_cycle(cycle).....	125
18.12.9 get_date_time ().....	125
18.12.10 set_date_time (year,mon,day,hour,min,sec).....	125
18.12.11 make_datetime(timestamp).....	125
18.12.12 make_timestamp(year,mon,day,hour,min,sec).....	125
18.12.13 set_pwd(level,pwd).....	125
18.12.14 md5(text, key).....	125
19. 免责声明.....	126

1. VisualHMI 组态软件

1.1 VisualHMI 组态软件安装

1.1.1 安装的环境需求

该软件适合 WIN7 /WIN10 系统(32 位/64 位)运行，磁盘空间需求大于 100M，内存空间大于 128M。


1.1.2 软件安装

1. 开始安装：打开 VisualHMI 软件安装包，点击下一步，如图 1-1 所示；



图 1-1 打开安装程序

2. 选择默认安装路径，直至安装完成；

3. 安装完成后，桌面生成快捷图标 。打开软件，显示界面如图 1-2 所示；

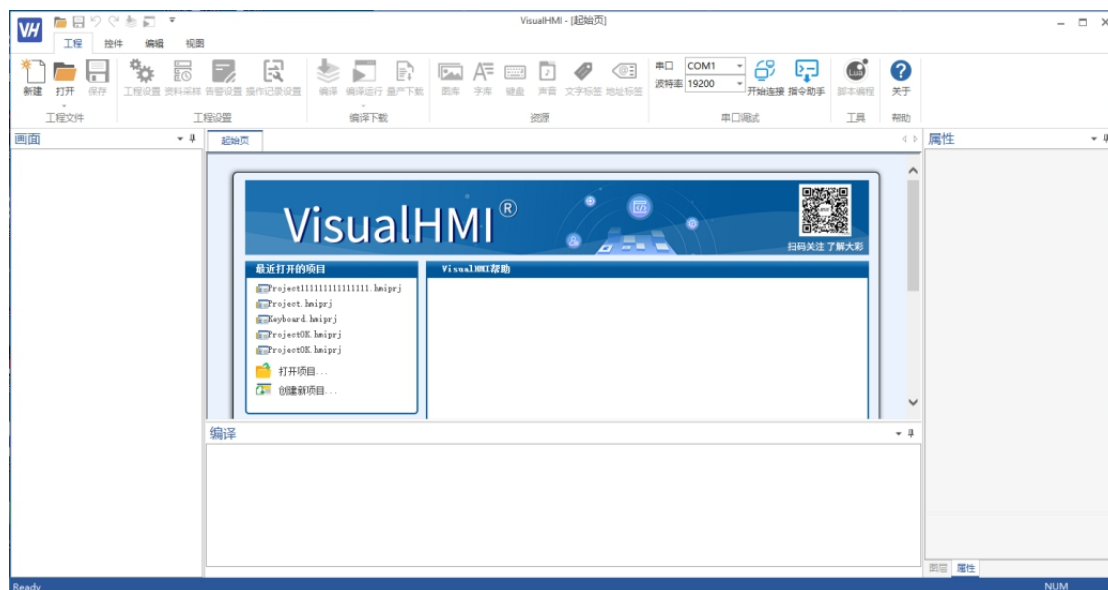


图 1-2 打开软件

1.2 VisualHMI 组态软件界面介绍

1.2.1 菜单区域

1. 工程管理区域

该区域主要用于管理工程文件、工程设置、工程编译、资源、串口调试、工具以及帮助，如图 1-3 所示：



图 1-3 工程管理区域

2. 组态控件区域

该区域主要用于选取组态控件，添加画面上，如图 1-4 所示：



图 1-4 组态控件区域

3. 画面编辑区域

该区域主要用于控件的调整，包括复制、编辑、排列、大小调整、控件上下层次切换、图标状态预览、多语言预览等，如图 1-5 所示：



图 1-5 画面编辑区域

4. 视图区域

该区域主要用于控制组态控件的标签显示/隐藏、画面放大/缩小，如图 1-6 所示：



图 1-6 视图区域

1.2.2 画面区域

1. 该区域主要用于画面的创建、删除、偏移以及选中等功能，如图 1-7 所示；

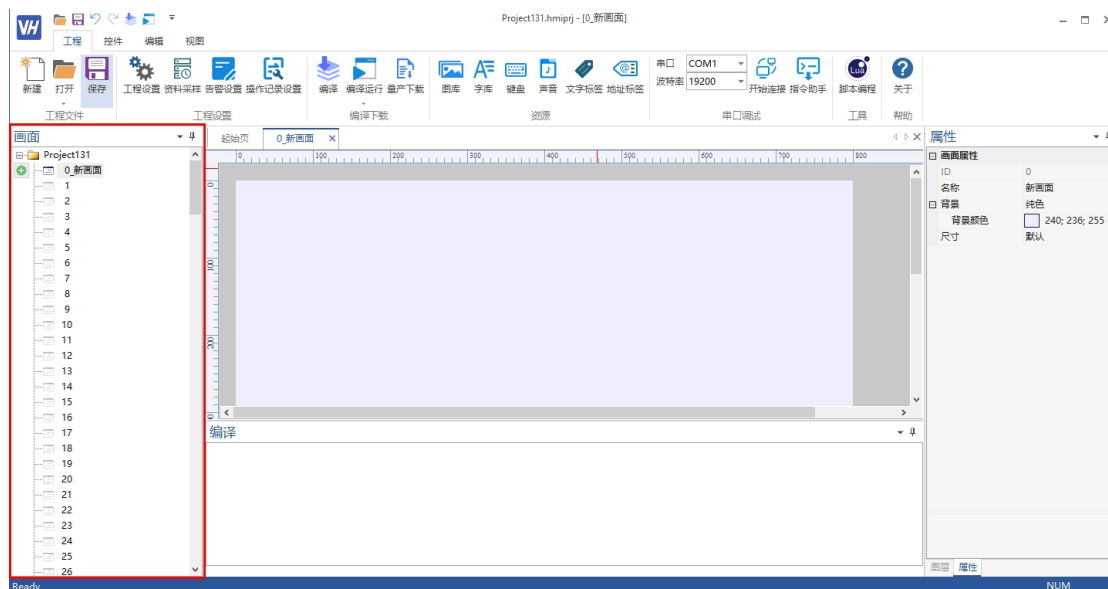


图 1-7 画面区域

2. 双击选中的画面，进入画面编辑模式；

1.2.3 画面编辑区域

该区域用于编辑当前选中的画面，如图 1-8 所示。

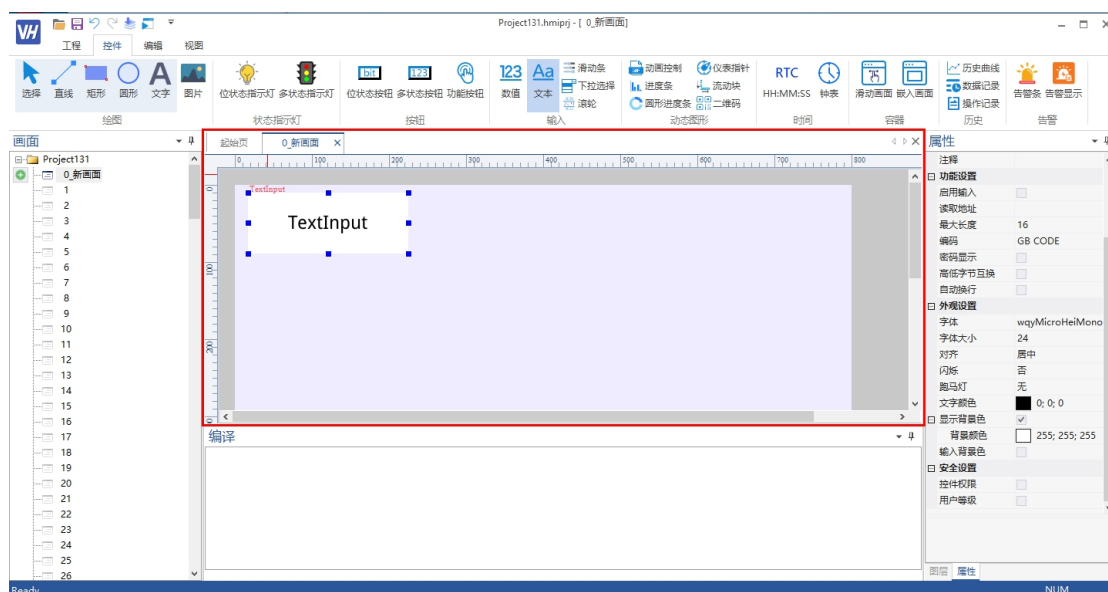


图 1-8 画面编辑区域

1.2.4 属性区域

该区域主要显示当前选中的画面或控件的属性，如图 1-9 所示

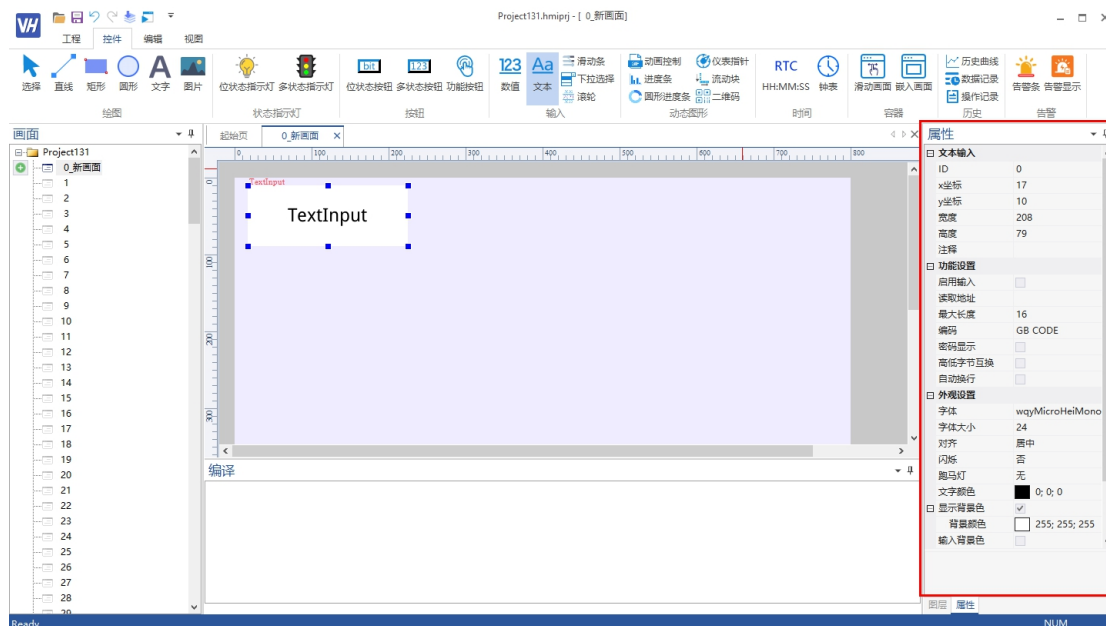


图 1-9 属性区域

1.2.5 编译区域

该区域主要显示当前工程的编译结果，如图 1-10 所示。

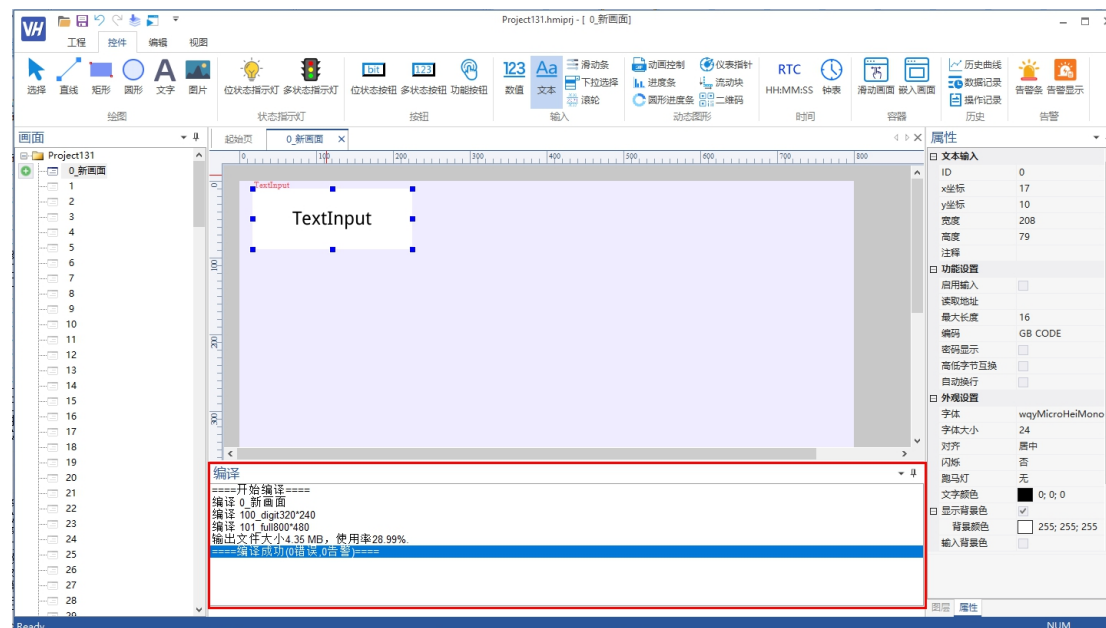



图 1-10 编译区域

2. 工程建立与下载

2.1 新建工程

新建工程步骤：

1. 点击 VisualHMI 软件菜单栏中【新建工程】按钮  新建；
2. 输入工程名字以及选择工程存放路径；
3. 根据产品选择对应的系列、型号；
4. 点击确定，生成工程。如图 2-1 所示；

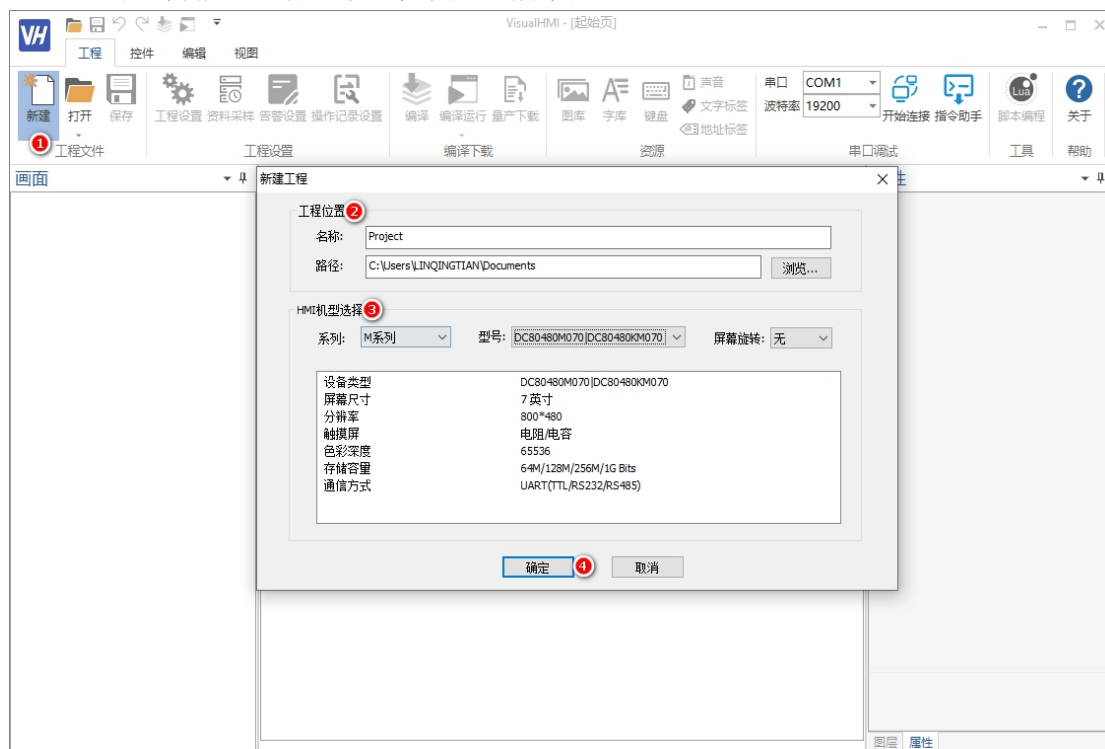



图 2-1 新建工程

2.2 工程设置

点击 VisualHMI 软件菜单栏中【工程设置】按钮  工程设置，在软件右侧属性栏中显示工程属性，如图 2-2 所示；

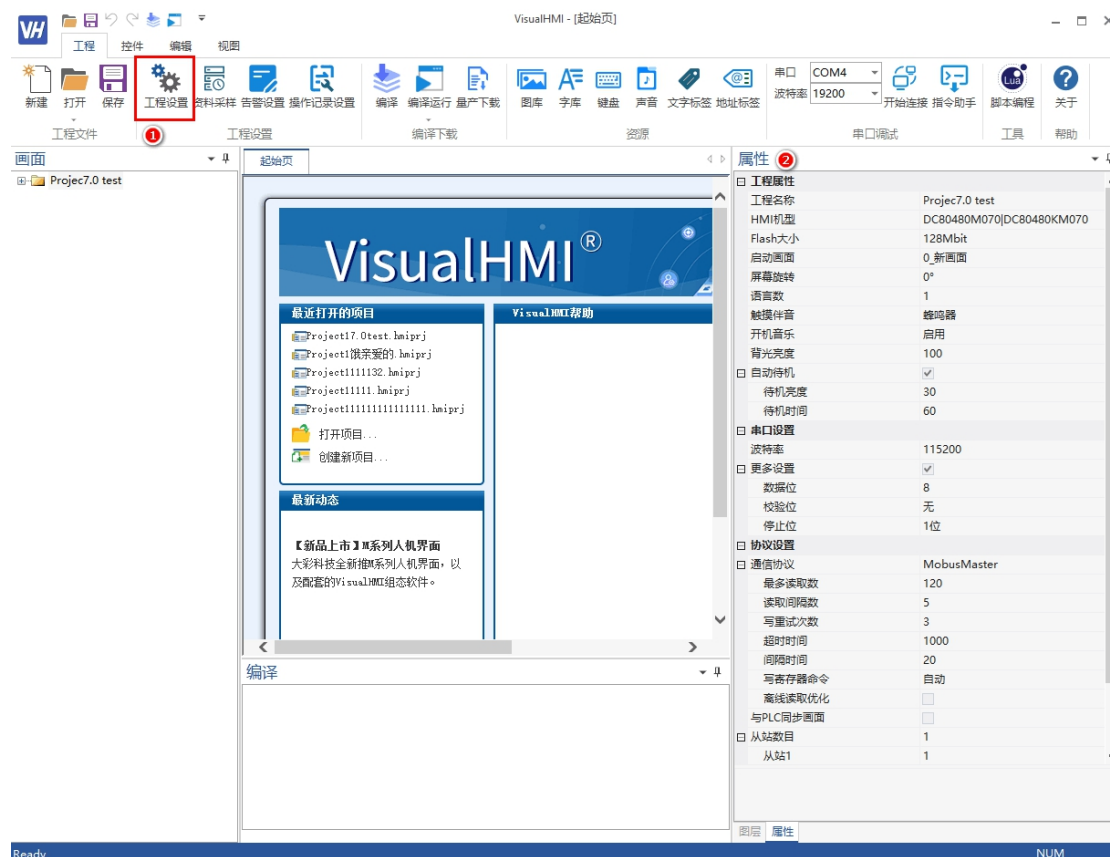


图 2-2 工程属性

此外，可以通过工程属性栏修改当前工程的名称、HMI 机型、Flash 大小，启动画面、旋转角度、语言数、开机音乐和背光调节等功能。本文仅对部分重要参数做描述，其他参数请查看 VisualHMI 属性栏中的介绍。

2.2.1 工程属性

1. 工程名称：当前工程名称；
2. HMI 机型：当前选择的产品型号；
3. Flash 大小：指串口屏存储空间大小，默认 128Mbit，可根据串口屏实际内存大小选择 128Mbit 或者 256Mbit；
4. 启动画面：开机时显示的第一个画面；
5. 屏幕旋转：工程显示角度，可以旋转 0°、90°、180° 和 270°；
6. 语言数：最大支持 30 种语言，根据客户需求自行修改；
7. 背光亮度：设定产品初始背光亮度，背光等级范围：0~100，100 最亮，0 全灭；
8. 自动待机：规定时间内无操作自动进入待机状态；
 - 待机亮度：进入待机状态后的背光亮度；
 - 待机时间：超时时间，单位秒；

2.2.2 串口设置

1. 设定串口屏串口的波特率，波特率范围：9600~921600 bps；
2. 打开更多设置可以修改串口屏串口的校验位、停止位和数据位；


2.2.3 通信协议

VisualHMI 支持多种协议开发，如：DCBUS、XGUS、ModbusMaster、ModbusSlave 和 FX2N 协议。

2.2.4 安全登录

设置用户的操作权限，最高支持 8 个等级，根据不同的用户等级，当操作权限低时，需要输入正确的密码才能操作，详细说明请参考“[第 6 章节](#)”。

2.3 型号更改

点击 VisualHMI 软件菜单栏中【工程设置】按钮 ，并在【属性区域】找到“HMI 机型”选项，点击该选项右侧导航按钮，在弹窗中修改 HMI 当前机型，如图 2-3 所示。

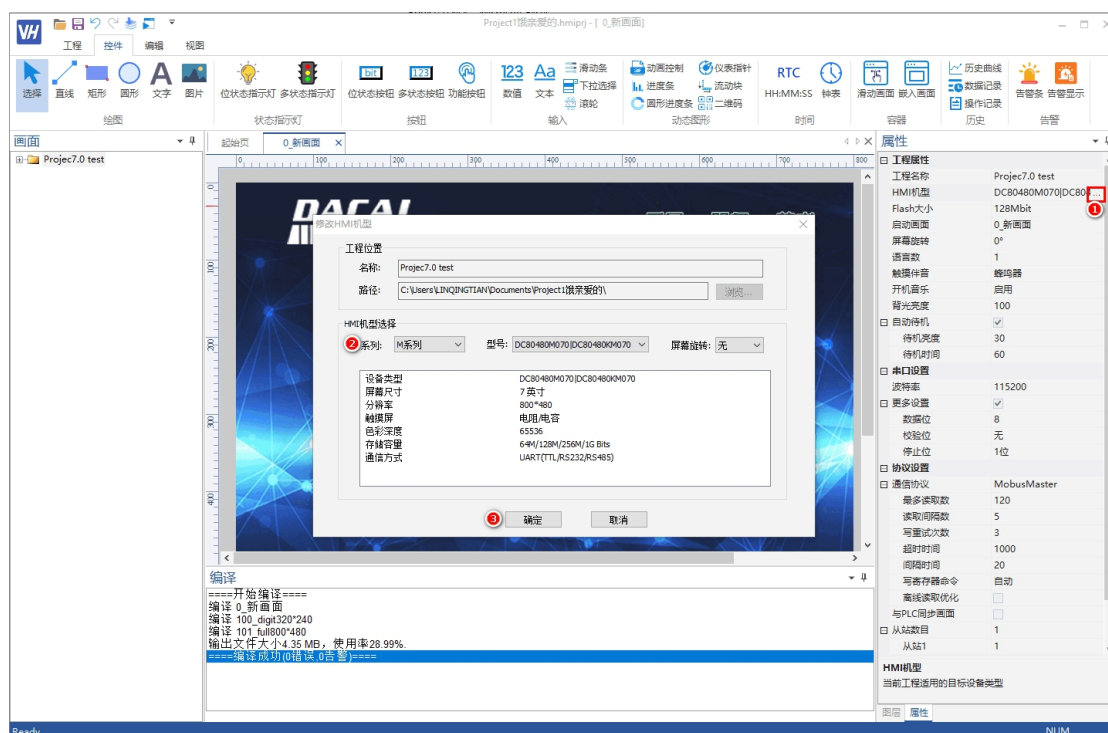


图 2-3 更改型号

2.4 工程编译与下载

2.4.1 工程编译


点击 VisualHMI 软件菜单栏中【编译】按钮 ，等待工程编译完成，如图 2-4 所示；

编译

```
====开始编译====
编译 0_新画面
编译 100_digit320*240
编译 101_full800*480
输出文件大小4.35 MB，使用率28.99%。
====编译成功(0错误,0告警)====
```

图 2-4 编译工程

2.4.2 串口下载

1. 下载前先点击 VisualHMI 软件菜单栏中 ，确认 VisualHMI 软件与大彩串口屏联

机成功 ；


2. 点击 VisualHMI 软件菜单栏中【量产下载】按钮 ，弹出【量产下载】窗口，点击串口下载，等待下载完成，如图 2-5 所示；




图 2-5 串口下载

2.4.3 SD 卡下载

SD 满足以下条件：

- ◆ **SD 容量建议 <= 8G**
- ◆ **格式为 FAT32/FAT**

1. 选中 VisualHMI 软件菜单栏中【量产下载】按钮 ，然后点击弹窗中的【SD 卡下载】按钮，把文件中所有文件拷贝到 SD 根目录下，如图 2-6 所示；

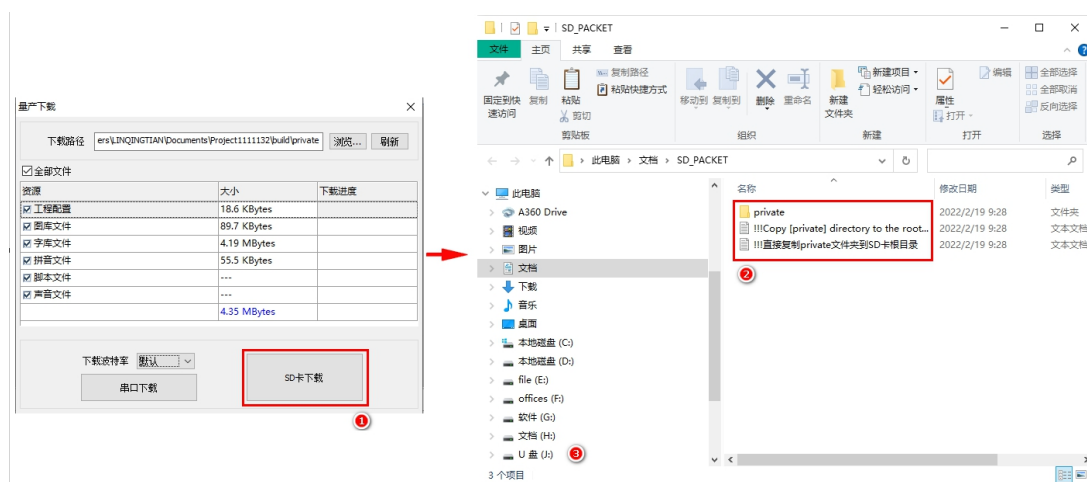


图 2-6 拷贝到 SD 卡

2. 将 SD 卡插入串口屏的卡槽，断电，上电，屏幕自动进入升级状态，如图 2-7 所示；显示“**update finished**”表示完成下载。下载完后屏幕自动加载，进入工程画面。此时，拔掉 SD 卡即可。



图 2-7 SD 下载

3. 组态控件

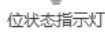
VisualHMI 组态控件如下图 3-1 所示。



图 3-1 组态控件

3.1 位状态指示灯



选中菜单栏中【位状态指示灯】, 在画面中添加一个位状态指示灯控件, 如图 3-2 所示。

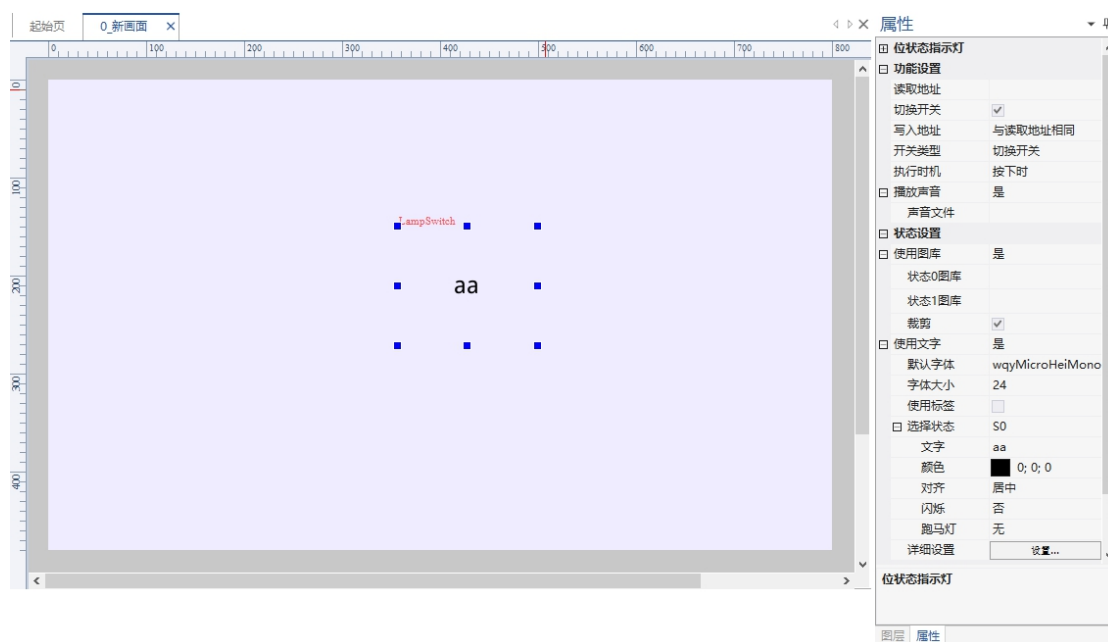


图 3-2 位状态指示灯

3.1.1 功能设置

1. 读取地址: 控件读取绑定的变量地址;
2. 切换开关: 启用状态切换开关;
3. 写入地址: 控件写入绑定的变量地址;
4. 开关类型: 设置 ON、设置 OFF、切换开关、点动开关;
 - 设置 ON: 按下目标地址写入 1;
 - 设置 OFF: 按下目标地址写入 0;
 - 切换开关: 按下时目标地址写入 1, 弹起时目标地址写入 0; 点击切换按下和弹起状态;
 - 点动开关: 按下目标地址写入 1, 弹起目标地址写入 0; 松开后自动弹起。
5. 执行时机: 可选按下时、弹起时执行;
6. 播放声音: 是或否;

- 声音文件：选择 MP3 或者 WAV 音频文件；

3.1.2 状态设置

1. 使用图库：是或否；
 - 状态 0 图库：状态 0 显示的图片文件；
 - 状态 1 图库：状态 1 显示的图片文件；
 - 裁剪：是或否，将选中的图片根据当前控件区域裁剪显示；
2. 使用文字：是或否。
 - 默认字体：默认字库文件；
 - 字体大小：字体文字大小；
 - 使用标签：是否使用文字标签，详细参考“13”；
 - 选择状态：修改指定状态 S1-Sn，全部状态选择 ALL；可设置状态的文字内容、颜色、对齐方式、是否闪烁、是否使用跑马灯效果等。

3.2 多状态指示灯



选中控件菜单栏中【多状态指示灯】，在画面中添加一个多状态指示灯控件，如图 3-3 所示。

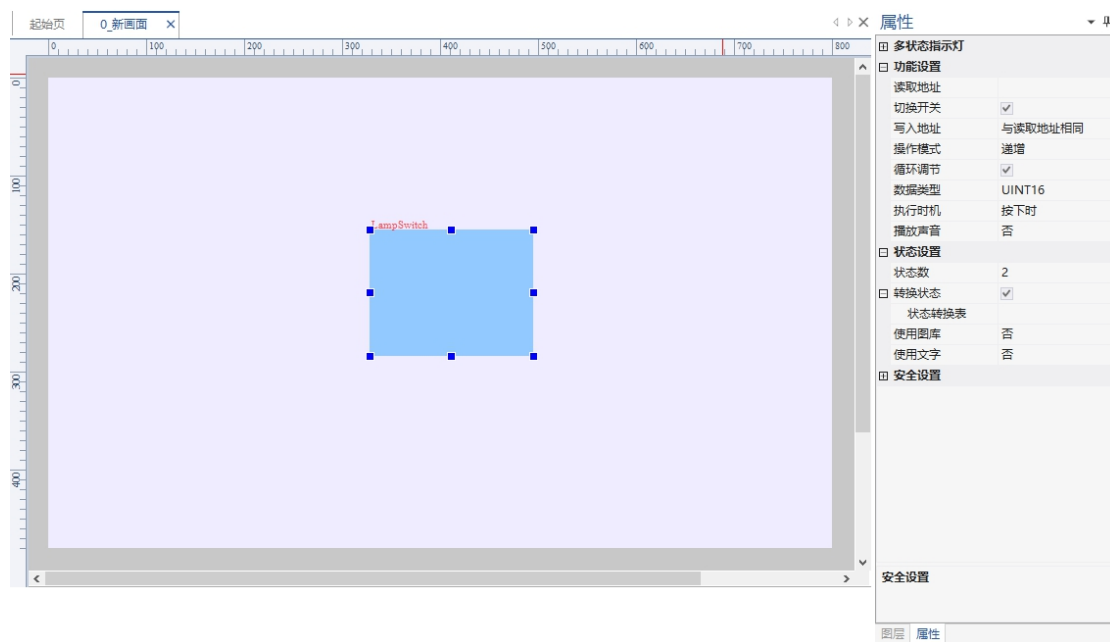


图 3-3 多状态指示灯

3.2.1 功能设置

1. 读取地址：控件变量读取绑定的地址；
2. 切换开关：启用状态切换开关；
3. 写入地址：控件变量写入绑定的地址；
4. 操作模式：状态切换改变的方式选择：递增、递减；
5. 循环调节：是否启动循环调节模式，启动后状态可循环切换；
6. 数据类型：变量数据类型选择，如 UINT16；


7. 执行时机：执行动作时的时机选择；
8. 播放声音：是或否；
 - 声音文件：选择 MP3 或者 WAV 音频文件；

3.2.2 状态设置

1. 状态数：多状态的状态数量；
2. 转换状态：状态值默认 0-N，也可重新设置每位状态对应的数值；
 - 状态转换表：设置状态索引和状态值的映射，例如：0;1;2;3;5；
3. 使用图库：是否启用图库，图库每一帧分别应一个状态；
4. 使用文字：是否使用文字，多状态文字每一状态可设置分别应一个文字状态，详细参考“13”；

3.3 位状态按钮



选中菜单栏中【位状态按钮】，在画面中添加一个位状态按钮控件，如图 3-4 所示。

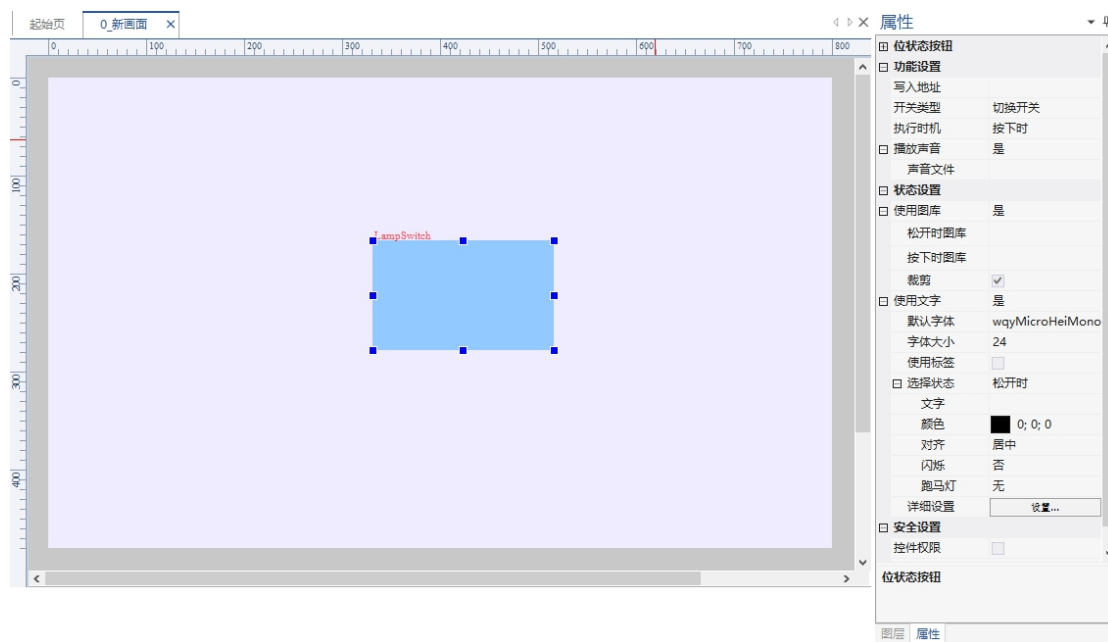


图 3-4 位状态按钮

3.3.1 功能设置

1. 写入地址：控件写入绑定的地址；
2. 开关类型：设为 ON、设为 OFF、切换开关、点动开关，本文以切换开关为例；
 - 设为 ON：按下时目标地址写入 1；
 - 设为 OFF：按下时目标地址写入 0；
 - 切换开关：按下时目标地址写入 1，弹起时目标地址写入 0；点击切换按下和弹起状态；
 - 点动开关：按下时目标地址写入 1，弹起时目标地址写入 0；松开后自动弹起；
3. 执行时机：执行动作时的状态选择；

4. 播放声音：是或否；
 - 声音文件：选择 MP3 或者 WAV 音频文件；

3.3.2 状态设置

1. 使用图库：是否使用图库。
 - 松开时图库：选择按钮松开状态下显示的图片文件；
 - 按下时图库：选择按钮按下状态下显示的图片文件；
 - 裁剪：是或否，将上一选项选中的图片按照当前控件区域裁剪显示；
2. 使用文字：是否使用文字；
 - 默认字体：选择默认字体；
 - 字体大小：设置字体文字大小；
 - 使用标签：选择设定的文字，详细参考“13”；
 - 选择状态：修改指定状态：按下时、松开时，全部状态选择 ALL；可设置状态的文字内容、颜色、对齐方式、是否闪烁、是否使用跑马灯效果等；

3.4 多状态按钮

选中菜单栏中【多状态按钮】，在画面中添加一个多状态按钮，如图 3-5 所示。

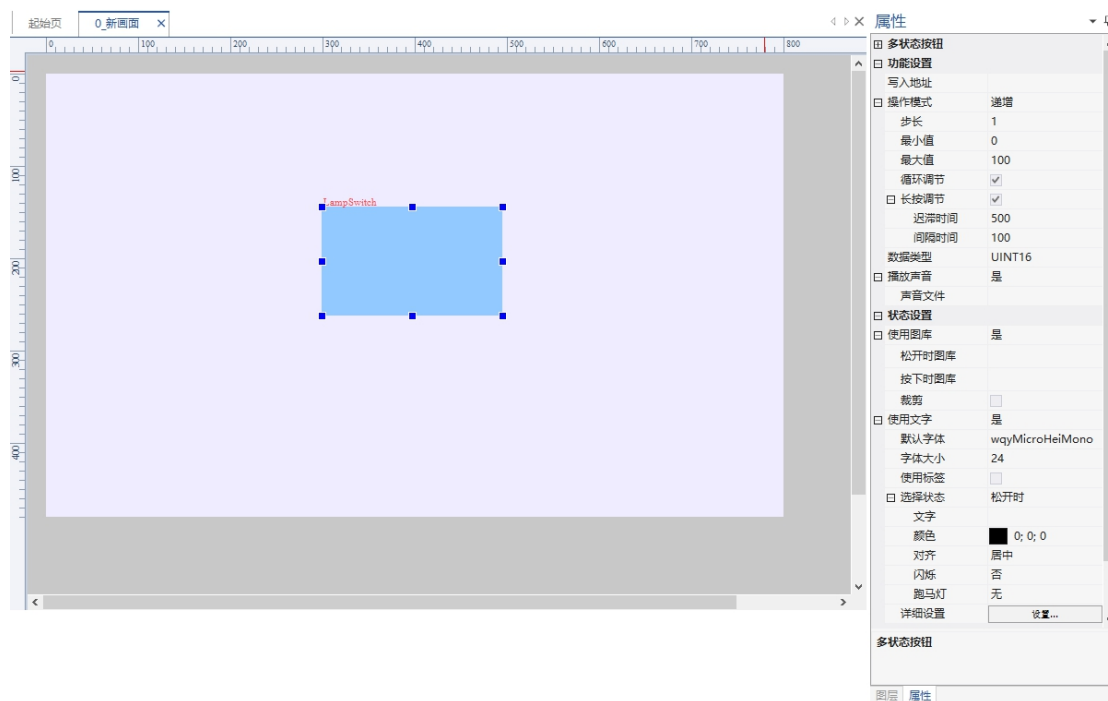


图 3-5 多状态按钮

3.4.1 功能设置

1. 写入地址：控件绑定的地址；
2. 操作模式：有三个选项：递增，递减，写入常量。本文以递增模式为例；
 - 步长：递增或递减模式下数值改变的步长；
 - 最小值：数值最小值；

- 最大值：数值最大值；
 - 循环调节：达到阈值后从最小开始；
 - 长按：是否开启；
 - 迟滞时间：长接触发时间，单位毫秒；
 - 间隔时间：指令下发间隔，单位毫秒；
3. 数据类型：数据类型选择，如 UINT16；
 4. 播放声音：是或否：
 - 声音文件：选择 MP3 或者 WAV 音频文件。

3.4.2 状态设置

1. 使用图库：属性说明和“[3.1.2](#)”章节的状态设置一样；
2. 使用文字：详细参考“[13](#)”；

3.5 功能按钮



选中控件菜单栏中【功能按钮】[功能按钮](#)，在画面中添加一个功能按钮，如图 3-6 所示。

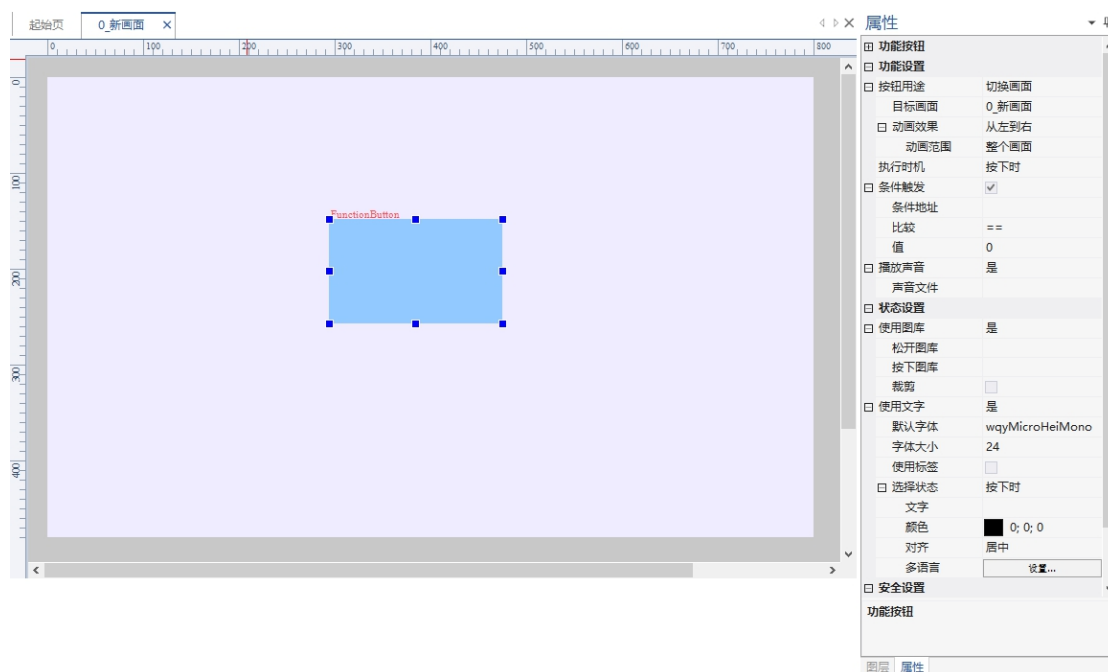


图 3-6 功能按钮

3.5.1 功能设置

1. 按钮用途：切换画面、打开对话框、关闭对话框、虚拟按键，本文以切换画面为例：
 - 目标画面：选择切换到的对应画面；
 - 动画效果：从左到右、从右到左、从上到下、从下到上；选择画面切换时的动画过渡效果；
 - 动画范围：设置动画效果的范围；
2. 执行时机：执行动作时的状态选择；
3. 条件触发：当条件成立时执行动作(触发式)；

- 条件地址：条件绑定的地址；
 - 比较：选择条件成立的比较方式；
 - 值：条件对比的值；
4. 播放声音：是或否
- 声音文件：1.wav，从工程声音文件库中选择；

3.5.2 状态设置

1. 使用图库：属性说明和“[3.1.2](#)”章节的状态设置一样；
2. 使用文字：详细参考“[13](#)”；

3.6 数值

123

选中控件菜单栏中【数值】数值，在画面中添加一个数值按钮，如图 3-7 所示。

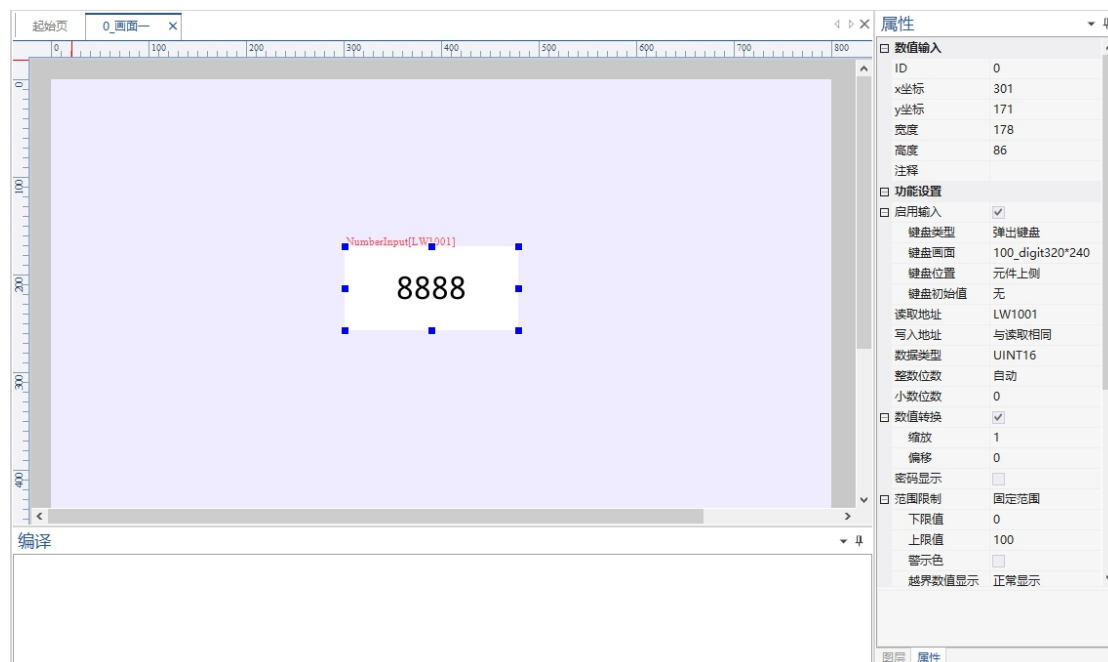


图 3-7 数值

3.6.1 功能设置

1. 启用输入：启用文字键盘输入功能；
 - 键盘类型：选择弹出键盘、外接键盘；
 - 键盘画面：可选择系统内置的键盘画面和用户自定义键盘画面；
 - 键盘位置：设置弹出键盘所在的位置；
 - 键盘初始值：设置键盘弹出时的初始值；
2. 读取地址：控件读取绑定的地址；
3. 写入地址：控件写入绑定的地址；
4. 数据类型：数据类型选择，如 UINT16；
5. 整数位数：设置数值整数部分的位数；
6. 数值转换：是否对数值进行转换运算；
 - 缩放：数值按比例进行缩放转换；


- 偏移：数值偏移转换。
- 7. 小数位数：设置数值小数部分的位数；
- 8. 密码显示：是否启用密码显示；启用后数值显示为*；
- 9. 范围限制：设置数值的上下限；可设置固定范围和可变范围，本文以固定范围为例；
 - a) 下限值：数值的下限值；
 - b) 上限值：数值的上限值；
 - c) 警示色：设置数值上下限值时的颜色；
 - 颜色用于：设置警示色用于文字或者文字背景；
 - 下限颜色：下限值时的颜色；
 - 下限闪烁：设置下限值时是否闪烁；
 - 上限颜色：上限值时的颜色；
 - 上限闪烁：设置上限值时是否闪烁；
 - d) 越界数值显示：设置数值显示超出范围时显示特殊字符或者正常显示；
 - e) 键盘输入越界时：设置键盘输入值超出范围时的效果；

3.6.2 外观设置

设置文字内容的字库、显示颜色、对齐方式、是否闪烁、是否使用跑马灯效果等。

3.7 文本



选中控件菜单栏中【文本】  ，在画面中添加一个文本控件，如图 3-8 所示。

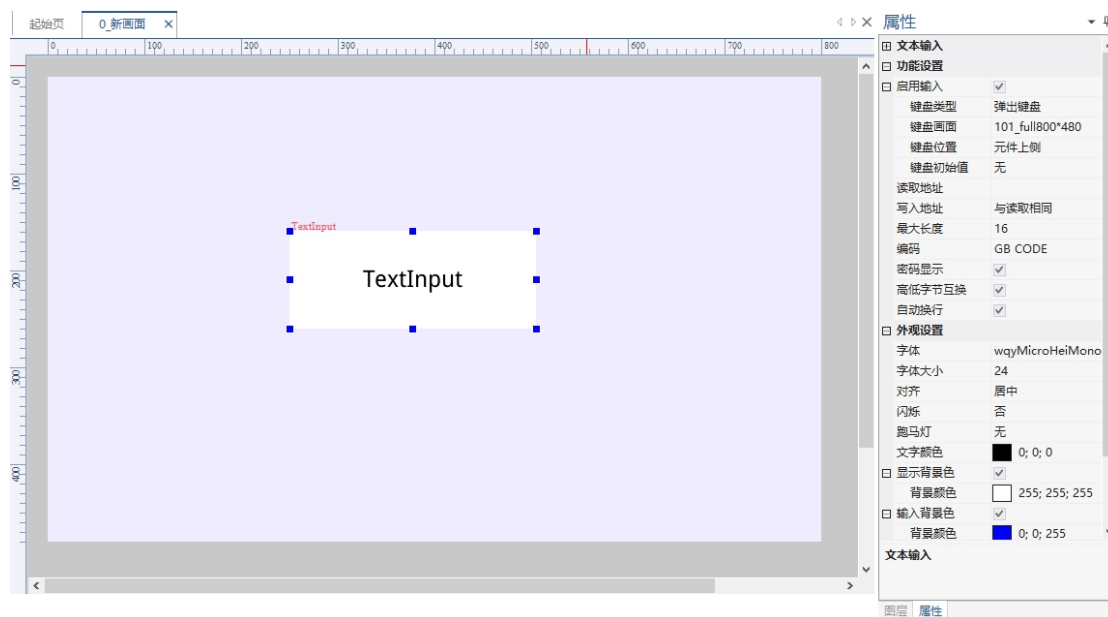


图 3-8 文本

3.7.1 功能设置

1. 启用输入：启用文字键盘输入功能；
 - 键盘类型：选择弹出键盘、外接键盘；
 - 键盘画面：可选择系统内置的键盘画面和用户自定义键盘画面；
 - 键盘位置：设置弹出键盘所在的位置；
 - 键盘初始值：设置键盘弹出时的初始值；
2. 读取地址：控件读取绑定的地址；
3. 写入地址：控件写入绑定的地址；
4. 最大长度：写入的字符串最大长度；
5. 编码：GB CODE，可选 UTF8、UNICODE 编码（键盘不支持输入）；
6. 密码显示：是或否；
7. 高低位互换：是或否；
8. 自动换行：是或否；

3.7.2 外观设置

可设置文字内容的字库、显示颜色、对齐方式、是否闪烁、是否使用跑马灯效果等。

3.8 滑动条



选中控件菜单栏中【滑动条】  ，在画面中添加一个滑动条控件，如图 3-9 所示。

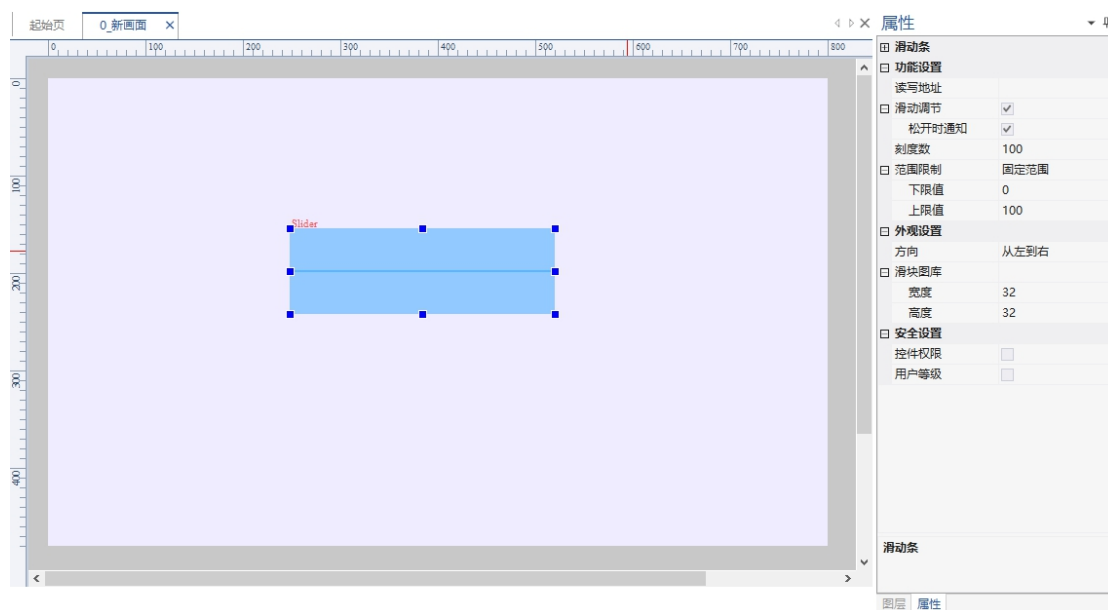


图 3-9 滑动条

3.8.1 功能设置

1. 读写地址：控件读取和写入绑定的地址；
2. 滑动调节：是否启用触摸滑动调节功能；
 - 松开时通知：开启时：滑动结束时下发通知；关闭时：滑动过程中下发通知；
3. 刻度数：滑动条显示的最大刻度数；
4. 范围限制：滑动条的范围值，可选择固定范围、可变范围，并设置对应的上下限值；

3.8.2 外观设置

1. 方向：设置滑动条的显示方向；
2. 滑块图库：设置滑动条滑块的图片文件和滑块的高度和宽度；

3.9 进度条



选中控件菜单栏中【进度条】**进度条**，在画面中添加一个进度条控件，如图 3-10 所示；

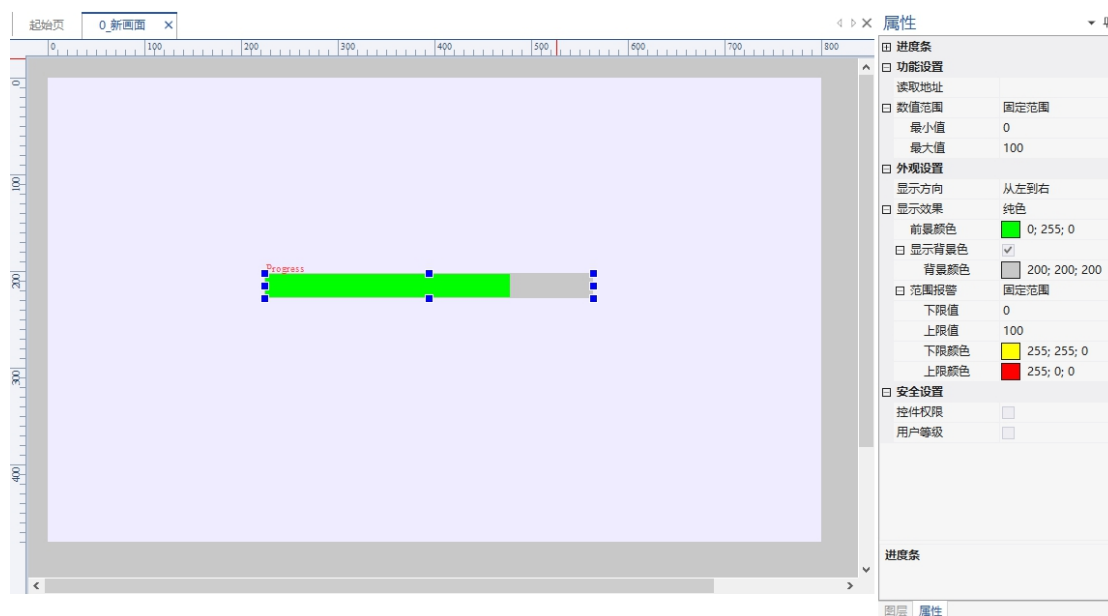


图 3-10 进度条

3.9.1 功能设置


1. 读取地址：控件读取绑定的地址；
2. 数值范围：进度条的范围值，可选择固定范围和可变范围；
固定范围：可设置最小值和最大值；必须为整数；

3.9.2 外观设置

1. 显示方向：设置进度条的显示方向；
2. 显示效果：设置进度条的显示效果，可选择纯色、渐变色、图片；
 - a) 纯色：设置进度条的前景颜色为纯色；
 - 显示背景色：可设置为纯色或渐变色时，设置进度条的背景显示颜色；
 - 范围报警：设置进度条报警时的范围值，可设置固定范围和可变范围，并设置报警上下限的颜色；
 - b) 渐变色：设置进度条的颜色从开始颜色渐变为结束颜色；
 - 开始颜色：设定起始颜色；
 - 结束颜色：设定终止颜色；
 - 范围报警：设置进度条报警时的范围值，可设置固定范围和可变范围，并设置报警上下限的颜色；
 - c) 贴图：设置进度条显示为图片，需设置进度条和背景的图库文件；

3.10 下拉选择



选中控件菜单栏中【下拉选择】, 在画面中添加一个下拉选择控件，如图 3-11 所示。

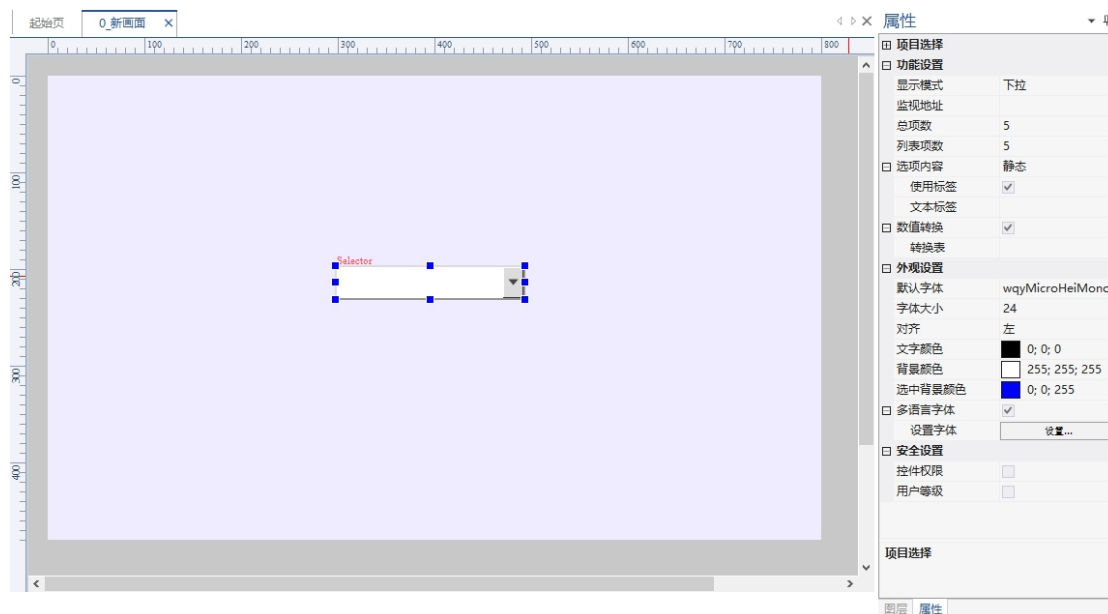


图 3-11 下拉选择

3.10.1 功能设置

1. 显示模式：设置下拉选择控件的显示模式，可设置下拉或列表；
 - 下拉：点击控件时，弹出所有可选项；
 - 列表：可选项直接以列表的形式显示；
2. 读取地址：下拉选择控件绑定的变量地址；
3. 总项数：选项中可选择的总项数；
4. 列表项数：列表的显示的项数，比如总项数：5，列表项数为 2。如所示，可以通过滚动条拖动显示、选择 5 个项的内容；

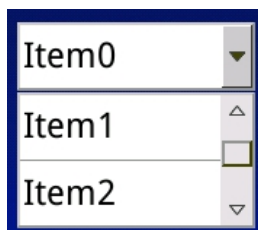


图 3-12 显示效果

5. 选项内容：设置下拉选择的选项内容，可设置静态、动态；
 - a) 静态：下拉选择选项内容为固定内容；
 - 使用标签：是否使用文字标签；
 - 文字选项：设置下拉选择选项的文字内容，如 0;1;2;3;4;5；
 - b) 动态：下拉选择选项内容为动态可变内容；
 - 变量地址：下拉选择选项动态内容的变量地址；
 - 最大长度：下拉选择选项动态内容的最大长度，单位：字节；
6. 数值转换：重新设置下拉选择每一项的值，默认为 0-N；
 - 转换表：填入下拉选项对应每一项值对应的转换值，如 30,31,32,33,34；

3.10.2 外观设置

1. 默认字体：下拉选择显示选项文字内容的默认字体；
2. 字体大小：下拉选择显示选项文字内容的字体大小；
3. 对齐：文字显示的对齐方式。
4. 文字颜色：文字内容的显示颜色。
5. 背景颜色：选项内容的默认背景颜色。
6. 选中背景颜色：点击选中选项时，选项内容的背景颜色。
7. 多语言字体：使用文本标签时，下拉选择可设置多语言，详细参考“13”；

3.11 滚轮


选中控件菜单栏中【滚轮】，在画面中添加一个滚轮控件，如图 3-13 所示。



图 3-13 滚轮

3.11.1 功能设置


1. 读取地址：滚轮控件绑定的变量地址；
2. 总项数：滚轮滚动选择的总项数；
3. 候选项：设置滚轮默认显示的项数，可设置 3 项或 5 项；
4. 惯性：可开启或关闭惯性；开启后，滑动松开后控件会有惯性滚动效果；
5. 对齐：设置滚轮滚动时，选项内容是否对齐；
6. 循环：设置滚轮滚动时，选项内容是否循环滚动；
7. 选项内容：设置滚轮的选项内容，可设置静态、动态；
 - a) 静态：滚轮选项内容为固定内容；
 - 使用标签：是否使用文字标签；
 - 文字选项：设置滚轮选项的文字内容，如 0;1;2;3;4;5；
 - b) 动态：滚轮选项内容为动态可变内容；
 - 变量地址：滚轮选项动态内容的变量地址；
 - 最大长度：滚轮选项动态内容的最大长度，单位：字节；

3.11.2 外观设置

1. 字体：滚轮显示选项文字内容的字体；
2. 字体大小：滚轮显示选项文字内容的字体大小；
3. 中间颜色：滚轮正中间选中项的文字颜色；
4. 两端颜色：滚轮两端选项内容的文字颜色；
5. 两端缩小：滚轮两端选项内容的文字缩放比例；

3.12 动画控制



选中菜单栏中【动画控制】，在画面中添加一个动画控制控件，如图 3-14 所示，可以播放 GIF。

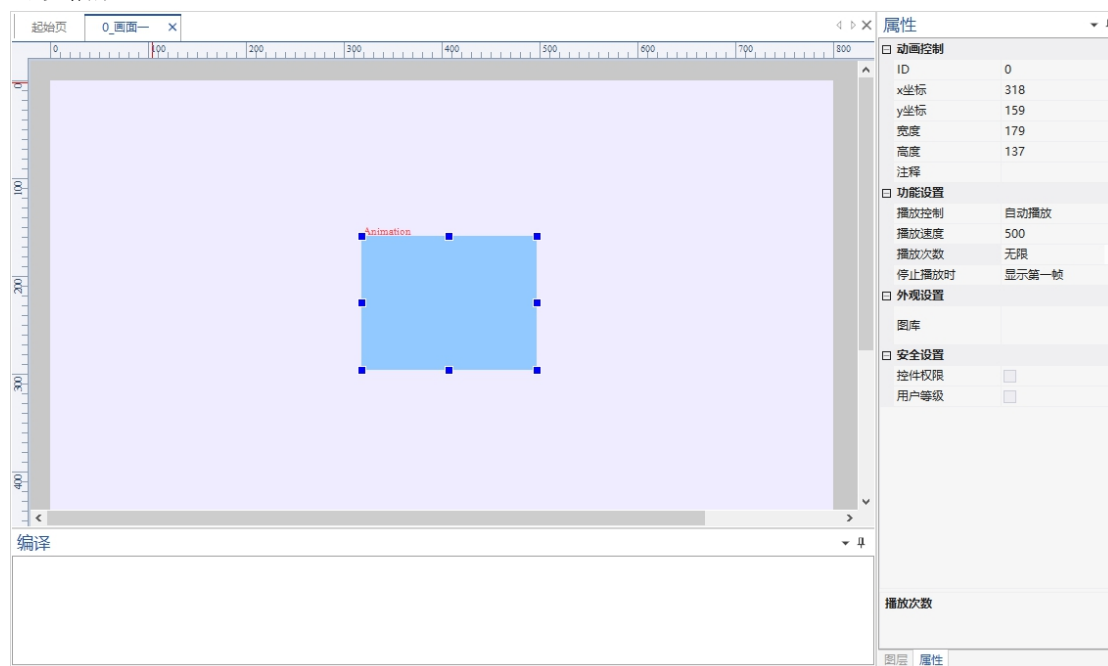


图 3-14 动画控制控件

3.12.1 功能设置


1. 播放控制
 - a) 自动播放：打开这个控件所在页面时，动画自动播放；
 - b) 条件控制：满足设置的条件时，动画开始播放，如下：
 - 读取地址：条件变量的地址
 - 比较：读取地址和值之间的比较关系，包括==、!=、<=、>=、<、>、IN：值在范围内（含边界）、OUT：值在范围外（不含边界）；
 - 值：与读取地址比较的数值，一般只有一个数值。当比较选择了 IN、OUT 时，有两个数值，包括下限值和上限值；
2. 播放速度：帧间隔时间，单位毫秒；
3. 播放次数：可以选择无限次数和指定次数；
4. 停止播放时：显示第一帧、隐藏动画、暂停动画；

3.12.2 外观设置

在图库选择控件显示的图片，设置外观。

3.13 圆形进度条



选中菜单栏中【圆形进度条】，在画面中添加一个圆形进度条控件，如图 3-15 所示，可以用来做圆形仪表。

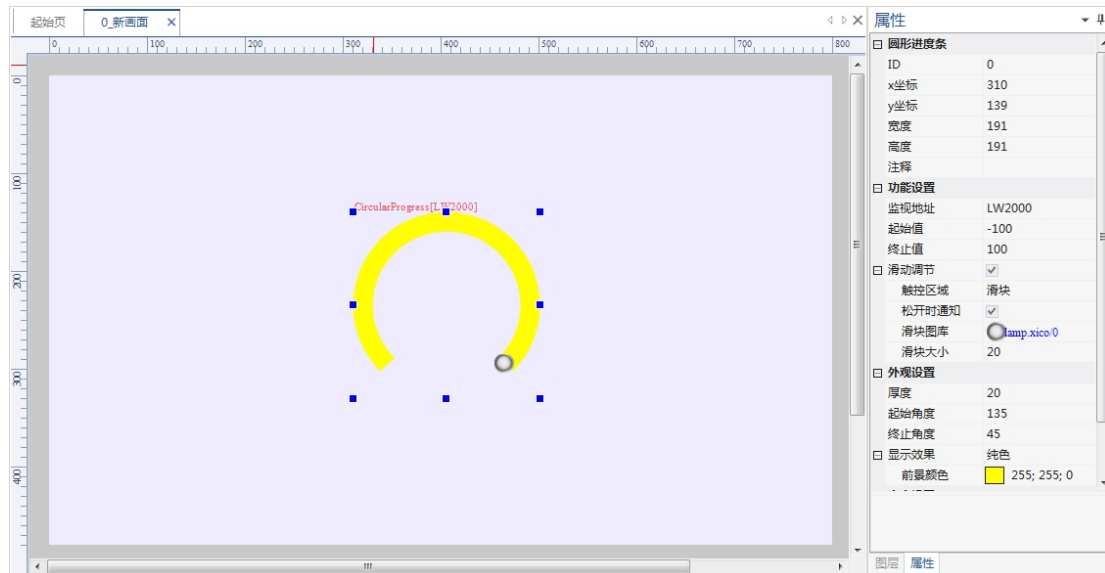


图 3-15 圆形进度条

3.13.1 功能设置


1. 读取地址：此地址的值就是进度条的旋转角度。只有读取地址的值在起始值和终止值范围内，进度条才会响应；
2. 起始值：起始数值；
3. 终止值：终止数值；
4. 滑动调节：设置之后可以触摸调节角度；
 - 触控区域：选择可以点击的位置，包括滑块、进度条；
 - 松开时通知：松开触摸时发送通知；
 - 滑块图库：选择滑块的显示样式，选择此选项后滑块可见；

3.13.2 外观设置

设置进度条的厚度、起始角、终止角，显示效果可以设置纯色或图片。起始角和终止角可限制进度条的旋转角度。

3.14 仪表指针



选中菜单栏中【仪表指针】，在画面中添加一个仪表指针控件，如图 3-16 所示，可以用来做指针仪表。

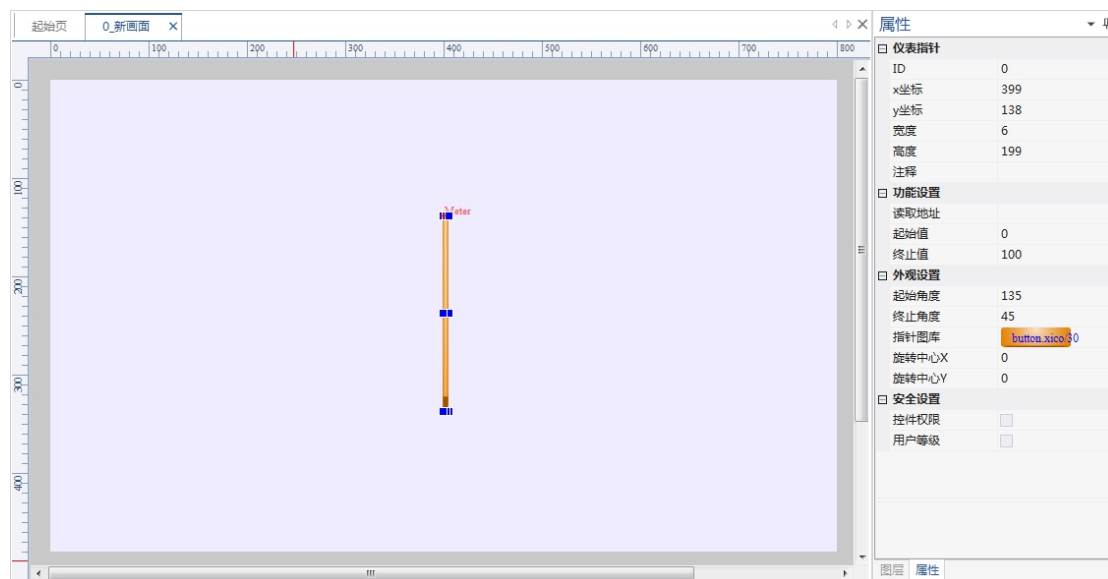


图 3-16 仪表指针

3.14.1 功能设置


1. 读取地址：此地址的值就是仪表指针的旋转角度；
2. 起始值：起始数值；
3. 终止值：终止数值；

3.14.2 外观设置

1. 起始角度、终止角度限制了仪表指针的旋转角度。起始值和终止值之间的数值平均分布在旋转角度内；
2. 指针图库用来选择指针图片；
3. 旋转中心 X、Y 设置指针的旋转中心；

3.15 流动块



选中菜单栏中【流动块】，在画面中添加一个流动块控件，如图 3-17 所示，可以用来做图片滚动。

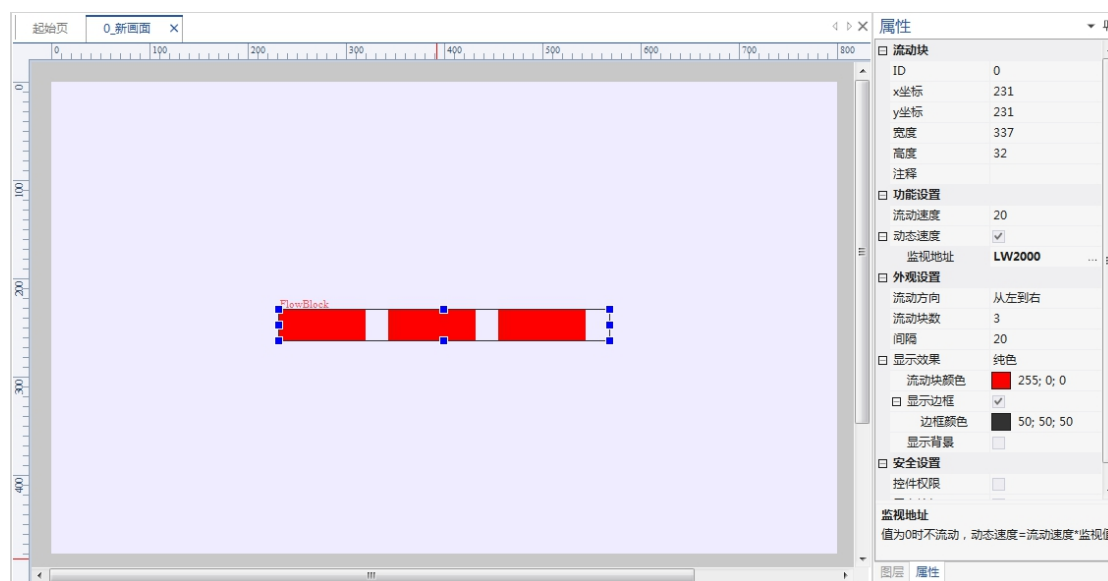


图 3-17 流动块

3.15.1 功能设置


1. 流动速度：每秒移动多少像素；
2. 动态速度：流动速度由读取地址的数值决定，设置此选项后，“流动速度”不生效；

3.15.2 外观设置

1. 流动方向：设置流动方向，可以设置从左到右、从右到左、从上到下、从下到上；
2. 流动块数：当显示效果是纯色时，可以设置块数；
3. 间隔：流动块之间的空白长度比例；
4. 显示效果：设置纯色、图片；
 - 流动块颜色/图库：显示效果是纯色时，选择流动块的颜色；显示图片时，在此选项设置图片；
 - 显示边框：设置是否显示边框、边框颜色；
 - 显示背景：显示纯色时，设置是否显示背景、背景颜色。默认是白色；

3.16 二维码



选中菜单栏中【二维码】，在画面中添加一个二维码控件，如图 3-18 所示，可以显示二维码。

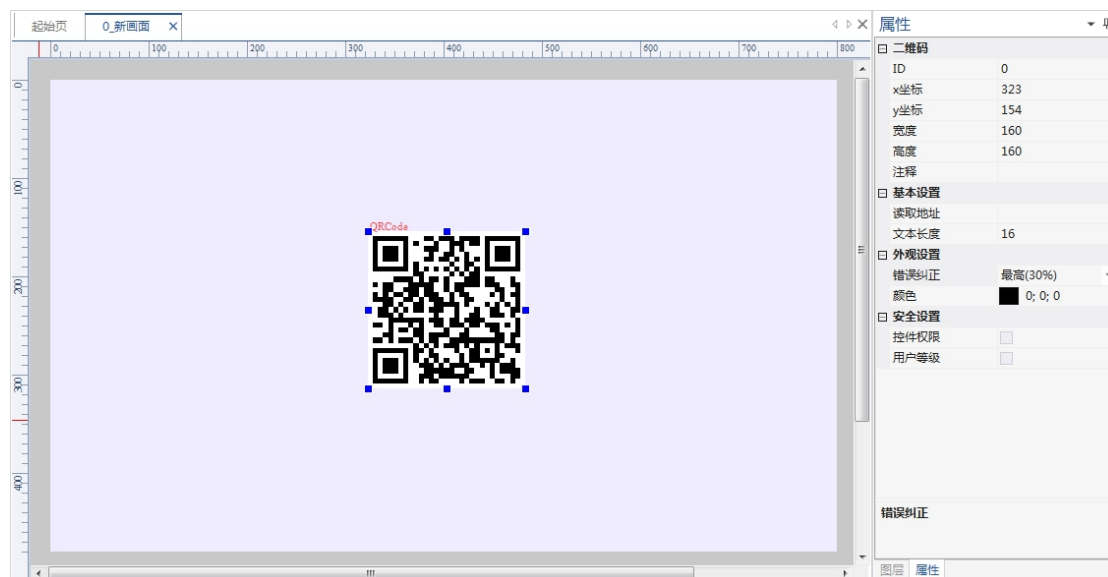


图 3-18 二维码


3.16.1 功能设置

1. 读取地址：二维码数据读取的首地址；
2. 文本长度：读取的文本长度单位字节。即从首地址开始，往后读取的字节数；

3.16.2 外观设置

1. 错误纠正：即二维码的容错能力，具有纠错功能，纠错级别越高，整体需要携带的信息越多。L 级可纠正约 7%错误、M 级别可纠正约 15%错误、Q 级别可纠正约 25%错误、H 级别可纠正约 30%错误；
2. 颜色：二维码显示颜色；

3.17 动态绘图

选中菜单栏中【动态绘图】，在画面中添加一个动态绘图控件，如图 3-18 所示。

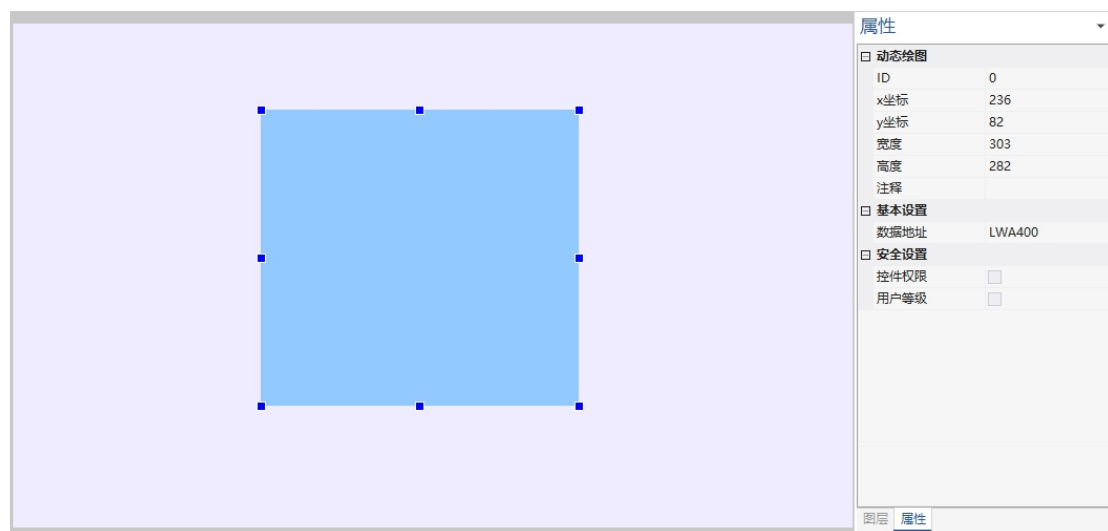


图 3-19 动态绘图

3.17.1 指令说明

通过设置变量存储器中的值，来实现绘制不同图形显示。支持显示置点、端点连线、矩形、矩形填充、圆形、频谱（垂直线段）、矩形填充反色等。指令格式（DCBUS）如下：

AA55 LEN 07 F1 1000 **CMD** **DATA_PACK_NUM_MAX** **DATA_PACK** CCCC

字段	含义
CMD 2byte	绘图指令，参考下表
DATA_PACK_NUM_MAX 2byte	最大的数据包数
DATA_PACK nbyte	数据内容，格式参考下表

绘图指令数据包说明如表所示：坐标 X、Y、Color 均两个字节

CMD 指令	操作内容	定义	说明
0x0001	置点	X+Y+Color(RGB)	将坐标 x,y，显示为 color 颜色
0x0002	端点连线	Color+(X0, Y0) + (X1, Y1) + ... + (Xn, Yn)	将(X0,Y0)...(Xn, Yn)坐标依次以 color 颜色连接
0x0003	矩形	(X0, Y0) + (X1, Y1) + Color	(X0, Y0) 为矩形左上角坐标、(X1, Y1) 为右下角坐标，
0x0004	矩形填充	(X0, Y0) + (X1, Y1) + Color	(X0, Y0) 为矩形左上角坐标、(X1, Y1) 为右下角坐标， 填充颜色
0x0005	圆形显示	(X0, Y0) + R + Color	(X, Y) 为圆心坐标，R 为半径（2byte）,Color 为圆弧的颜色
0x0009	频谱显示 (垂直线条)	Color + X0 + Y0+ Y1	将(X0,Y0)、(X0, Y1)坐标以 color 颜色连接
0x000A	线段	Color + (X0, Y0) + (X1, Y1)	将(X0,Y0)、(X1, Y1)坐标以 color 颜色连接
0x000D	矩形反色填充	(X0, Y0) + (X1, Y1) + Color	(X0, Y0) 为矩形左上角坐标、(X1, Y1) 为右下角坐标， 填充 Color 颜色

3.18 RTC

RTC

选中菜单栏中【RTC】**HH:MM:SS**，在画面中添加一个 RTC 控件，如图 3-20 所示，用来显示日期、时间。

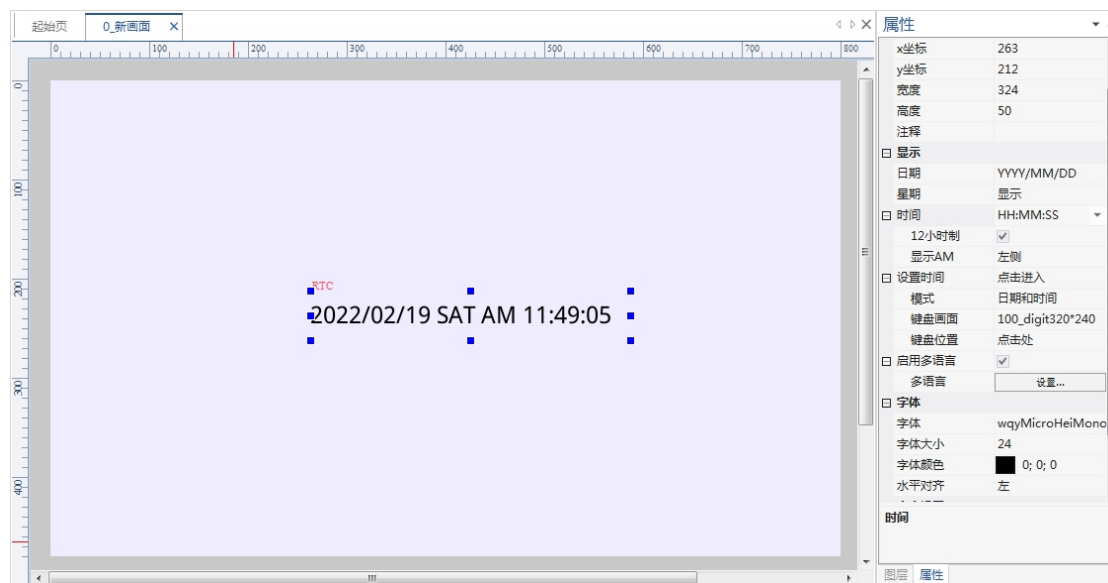


图 3-20 RTC

3.18.1 功能设置


1. 日期：设置年月日显示或不显示；
2. 星期：设置星期显示或不显示；
3. 时间：设置时分秒显示或不显示，默认显示 24 小时制；
 - 12 小时制：设置 12 小时制；
 - 显示 AM：设置 12 显示制时，AM 可以显示在左侧或右侧；
4. 设置时间：设置是否可以手动设置时间，可设置禁止、点击进入、长按 3s 进入；
 - a) 模式：可以设置日期和时间、日期、时间；
 - b) 键盘画面：选择键盘；
 - c) 键盘位置：设置键盘位置；
5. 启用多语言：详细参考“13”；

3.18.2 外观设置

设置字体、字体大小、字体颜色、水平对齐。

3.19 钟表



选中菜单栏中【钟表】，在画面中添加一个钟表控件，如图 3-21 所示，可以用来显示时针、分针、秒针。

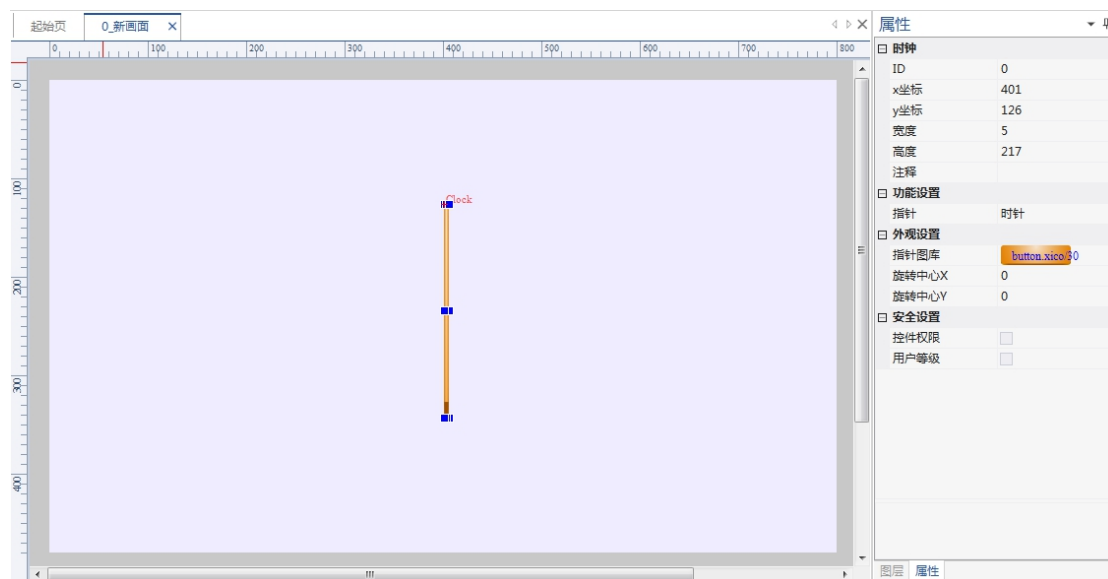


图 3-21 钟表

3.19.1 功能设置

指针：设置时钟、分针、秒针。

3.19.2 外观设置

1. 指针图库用来选择指针图片；
2. 旋转中心 X、Y 设置指针的旋转中心；

3.20 历史曲线



选中菜单栏中【历史曲线】[历史曲线](#)，在画面中添加一个历史曲线控件，如图 3-22 所示，可以用来显示历史数据的曲线图。

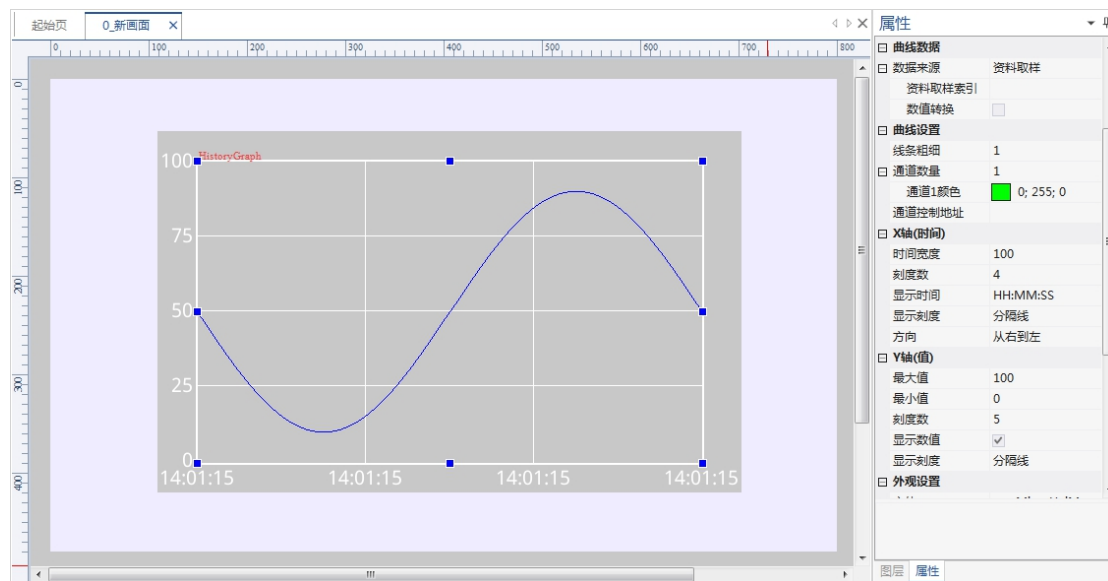


图 3-22 历史曲线

3.20.1 曲线数据

1. 数据来源：资料取样

- 打开资料采集设置，选择资料取样索引（详细参考“资料采集”），如图 3-23 所示。
- 数值转换：数值可以进行缩放在绘制到曲线上， $\text{实际值} = \text{原值} * \text{缩放} + \text{偏移}$

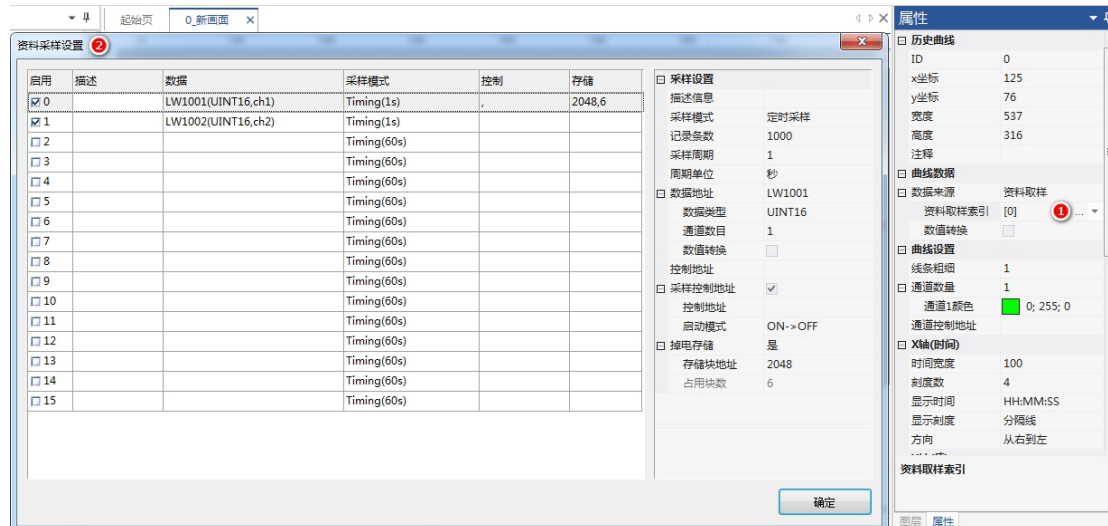


图 3-23 资料采样设置

2. 连续地址：每次采样连续读取多个数据点，并在曲线中显示多个点，并用直线连接；

- 数据点数：连续读取的点数；
- 数据地址：第一个数据点的首地址；
- 数据类型：UINT16、INT16、UINT32、INT32、UINT64、INT64、FLOAT、DOUBLE；
- 数值转换： $\text{实际值} = \text{原值} * \text{缩放} + \text{偏移}$ ；

3.20.2 曲线设置

线条粗细：1 或 2 个像素

3.20.3 通道数量

- 通道数量：最大支持 16 条
- 通道颜色：支持每条通道单独设置颜色

3.20.4 控制地址

设置通道的显示/隐藏。如填入 LW1000，则对应的 bit 为 1 时隐藏，为 0 时显示。默认为空，即全部显示；bit0 表示通道 1 的控制

翻页控制：选填

- LW1001(控制地址 + 1)：上一页设置 1，下一页设置为 2，首页设置为 3，尾页设置 4；
- LW1002(控制地址+2)：位掩码只是翻是否可行（只读）
- LW1003（控制地址+3）：时间戳定位，2 个字节

3.20.5 X 轴（时间）

设置时间宽度、刻度数、显示时间、显示刻度（刻度线、分割线）、方向；

3.20.6 Y 轴（值）

设置最大值、最小值、刻度数、显示数值、显示刻度（刻度线、分割线）；
最大值，最小值，可以在 Lua 脚本实现，调用以下 API 实现缩放。且设置曲线控件的 ID 不为 0

```
--[[/*
API:wgt_set_param(screen,control, param,value)
● screen:画面 ID
● control:控件 ID
● param: 0x31,最小值, 0x32,最大值
● value: 设置
*/--]]

--设置 Y 轴最大值为 100, 最小值为 0
wgt_set_param(0,0,0x31,0)
wgt_set_param(0,0,0x32,100)
```

3.20.7 外观设置

设置字体、大小、文字颜色、边框、背景、分割线颜色、滚动条；

3.20.8 显示触摸标识

当用户点击曲线的显示，会显示该点的时间、数值

3.21 数据记录



选中菜单栏中【数据记录】数据记录，在画面中添加一个数据记录控件，如图 3-24 所示。

可以用来显示历史数据的表格。

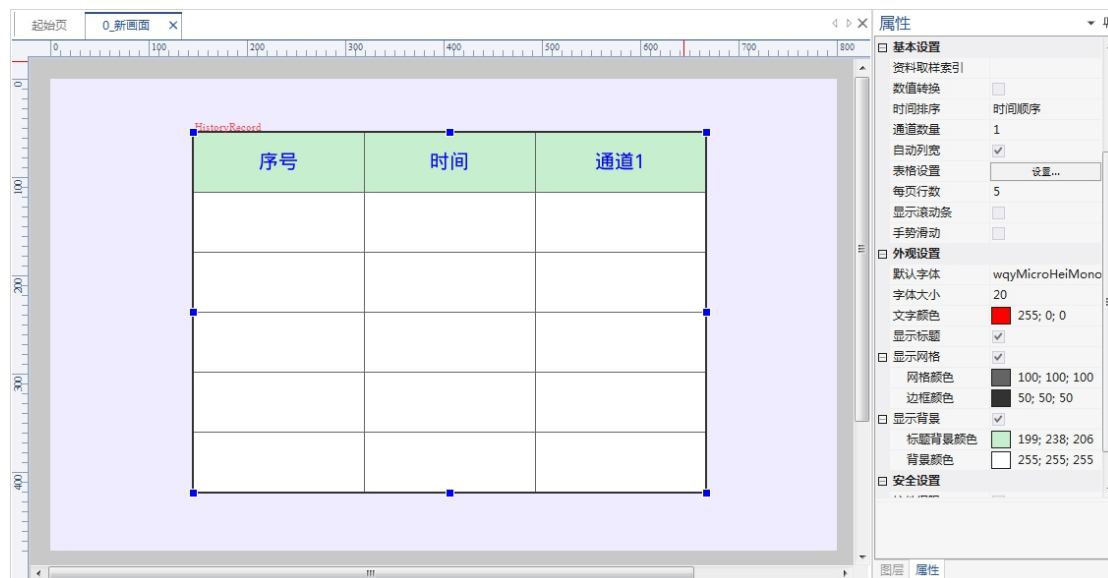


图 3-24 数据记录

3.21.1 基本设置

1. 资料取样索引：打开资料采集设置（详细参考“资料采集”），如图 3-25 所示。

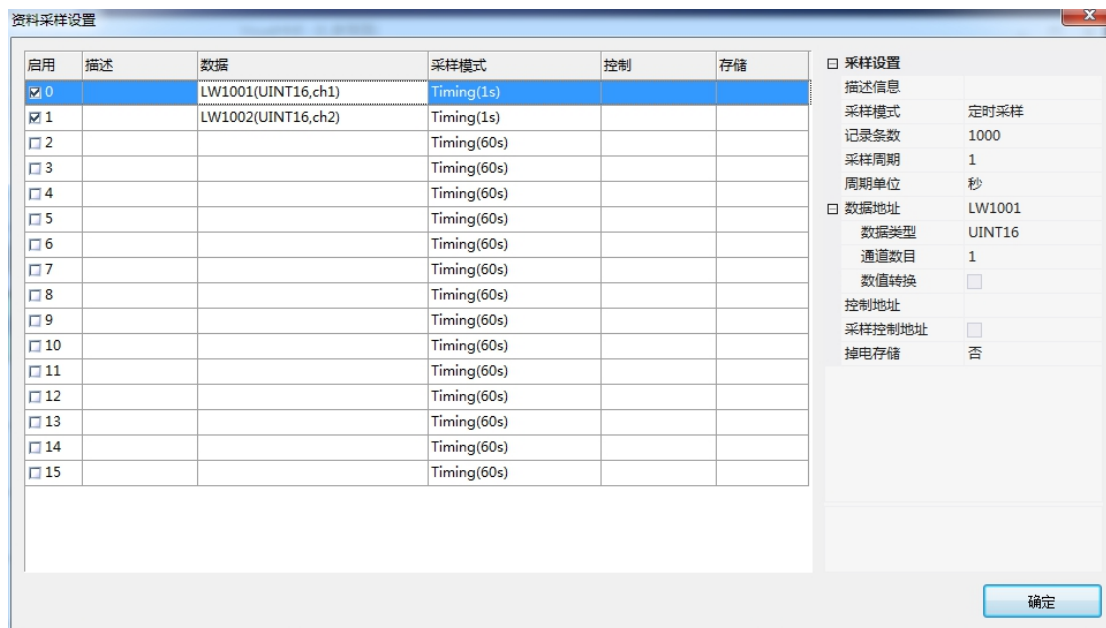


图 3-25 资源采集设置

2. 字符串表格：若该笔资料位字符串，则勾选改参数；
3. 时间排序：时间顺序、逆序；
4. 日期/时间：时间逆序、时间顺序
5. 通道数量：数据通道的数量；
6. 自动列宽：各列按比例自动调整宽度；
7. 表格设置：设置表格的标题、标题文本、宽度；
8. 每行页数：每页行数；
9. 显示滚动条：设置超过行数时，显示滚动条；
10. 手势滑动：设置可以手动滑动；
11. 翻页控制：假设控制地址为 LW1000
 - LW1000(控制地址 + 1)：上一页设置 1，下一页设置为 2，首页设置为 3，尾页设置 4；
 - LW1001(控制地址+2)：位掩码只是翻是否可行（只读）
 - LW1002（控制地址+3）：时间戳定位，2 个字节

3.21.2 外观设置

设置字体、字体大小、字体颜色、显示标题、网格、背景。

3.22 操作记录



选中菜单栏中【操作记录】操作记录，在画面中添加一个操作记录控件，如图 3-26 所示。

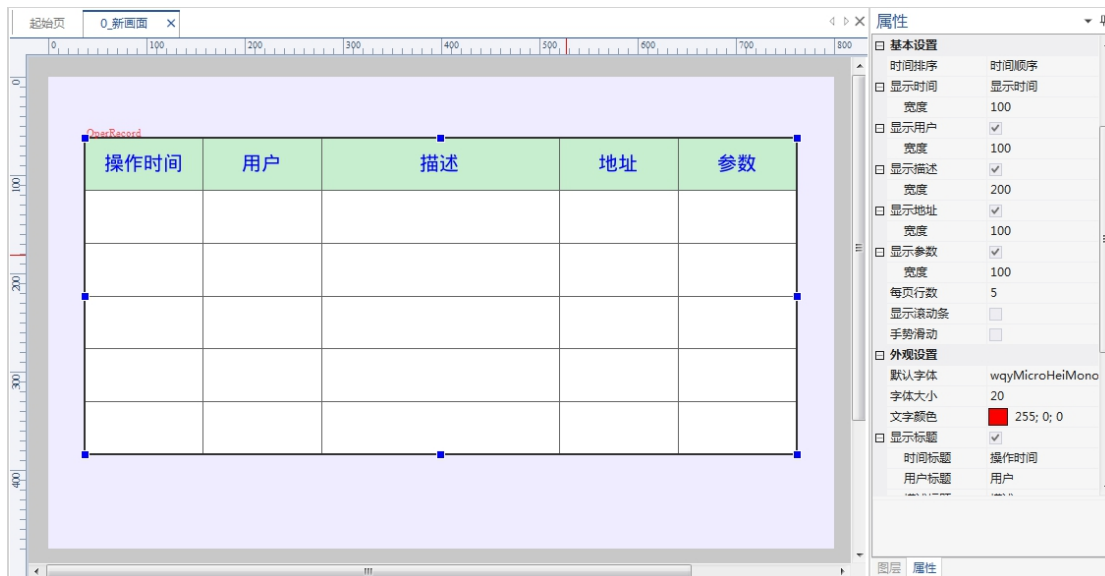


图 3-26 操作记录

3.22.1 基本设置

1. 设置时间排序、显示的时间、用户、描述、地址、表格的宽度等；
2. 每页行数：显示每页的行数；
3. 显示滚动条：是否显示滚动条；
4. 手势滑动：是否开启手势滑动；

3.22.2 外观设置

设置默认字体、字体大小、文字颜色、显示标题、显示网格、显示背景。

3.22.3 控件操作-使用操作记录

要启用操作记录，必须打开控件的操作记录选项。以下用数值控件举例，在属性的最下面选中操作记录，同时可以增加操作描述，如“测试 1”。如图 3-27 所示。



图 3-27 打开操作记录

当操作控件时，操作记录就显示到控件中，如图 3-28 所示。



操作时间	用户	描述	地址	参数
15:42:50	User0	测试1	LW2000	0->3
15:42:52	User0	测试1	LW2000	3->56
15:42:54	User0	测试1	LW2000	56->18

18

图 3-28 操作记录

3.22.4 控件操作-系统参数标签操作

操作记录可以使用系统参数地址操作，请参考“[第 13 章节](#)”。

3.23 滑动画面




在菜单工程栏→【滑动画面】，在画面中添加一个滑动画面控件，如图 3-18 所示。该控件可以实现手机桌面左右滑动的效果。



图 3-29 滑动画面

3.23.1 功能设置

1. 滑动范围：固定和动态。动态：可以设置开始画面、结束画面。如：读写地址（当


- 前地址) 为 LW1022, 开始画面为 LW1023, 结束画面为 LW1024
2. 开始画面: 关联对应的画面;
 3. 结束画面: 关联对应的画面;
 4. 滑动方向: 可以设置“左右”、“上下”;
 5. 滑动时间: 单位毫秒;
 6. 读写地址: 控件读取和写入绑定的地址; 如图 3-30 所示。

功能设置	
滑动范围	固定
开始画面	1_主画面一
结束画面	3_主画面三
滑动方向	左右
循环滑动	<input checked="" type="checkbox"/>
滑行时间	300
读写地址	LW1022 ...
安全设置	
控件权限	<input type="checkbox"/>
用户等级	<input type="checkbox"/>
读写地址	
地址0=当前画面;	
地址1=开始画面;地址2=结束画面;	

图 3-30 滑动设置

3.24 嵌入画面



菜单工程栏→【嵌入画面】, 在画面中添加一个嵌入画面控件, 如图 3-31 所示。

该控件可以实现手机拖动长图查看的效果。

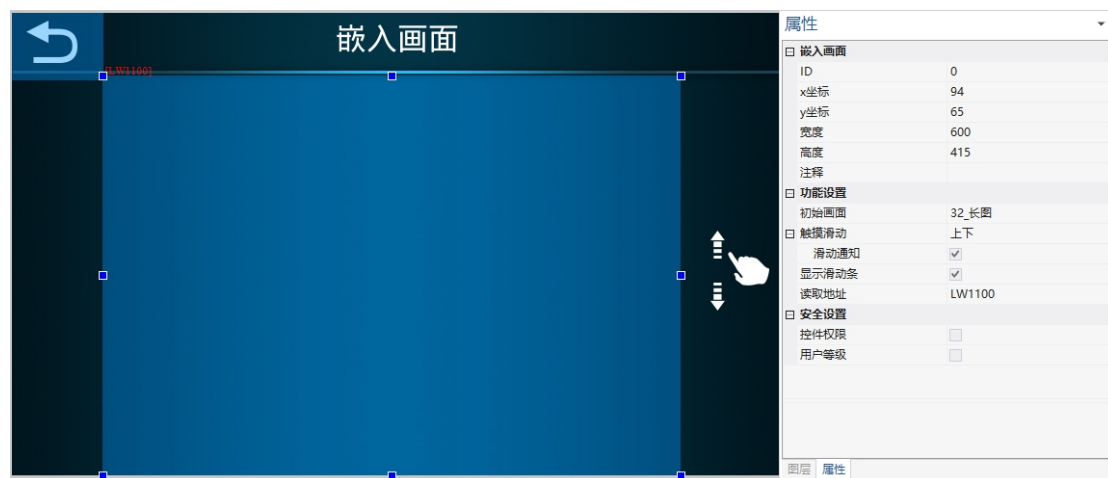


图 3-31 嵌入画面控件

3.24.1 功能设置

1. 初始画面：设置初始化的画面，该控件区域，会映射到指定画面，并显示该画面的内容
2. 触摸滑动：可以设置“左右”、“上下”；
 - 滑动通知：开启后，滑动滑动条会下发通知
3. 显示滑动条：勾选、否
4. 读写地址：控件读取和写入绑定的地址；如图 3-32 所示。如：读写地址（当前地址）为 LW1100，设置指定画面。进度条滚动位置为 LW1101

功能设置	
初始画面	32_长图
触摸滑动	上下
滑动通知	<input checked="" type="checkbox"/>
显示滑动条	<input checked="" type="checkbox"/>
读取地址	LW1100 ...
安全设置	
控件权限	<input type="checkbox"/>
用户等级	<input type="checkbox"/>
读取地址 地址+0, 设置目标画面; 地址+1, 设置偏移位置;	

图 3-32 功能设置

4. 资料采集

资料采集设置可以在数据记录控件、或历史曲线控件中打开,还可以在菜单工程栏→【资

料采集设置】 进入,如图 4-1 所示。

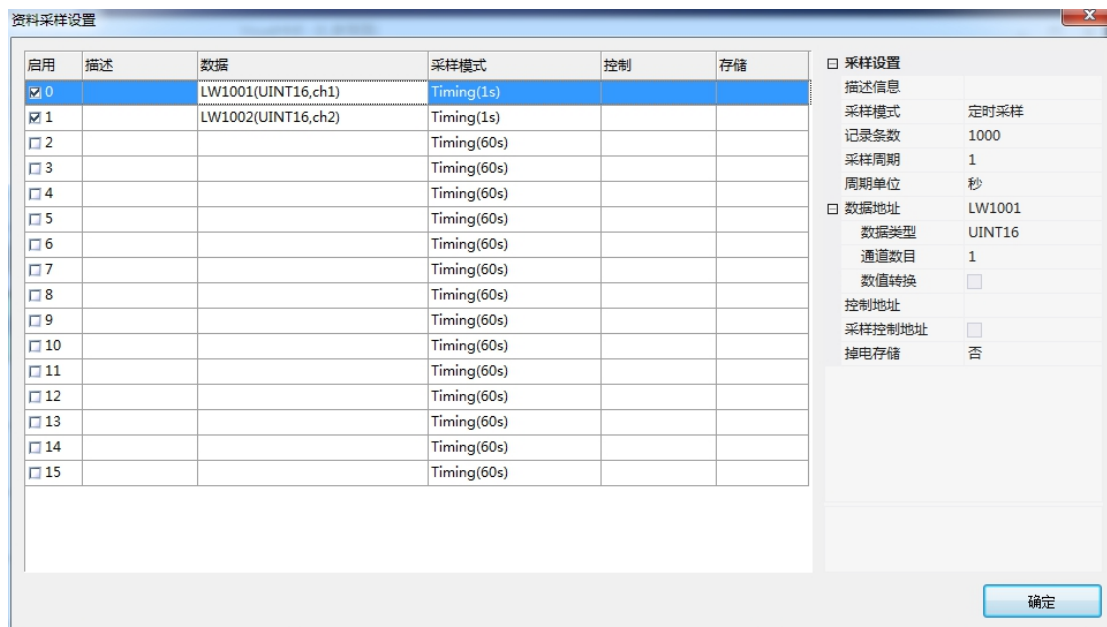


图 4-1 资源采集设置

4.1.1 采样模式

1. 定时采样: 以固定周期采样数据;
 - 采样周期: 设置定时采样周期;
 - 周期单位: 秒、毫秒;
2. 触发采样: 由某个寄存器值变化后, 触发采样;

4.1.2 记录条数

最多记录的条数, 当记录满了后, 自动覆盖。支持 10~10000000 条

4.1.3 数据地址

采样的变量起始地址。

4.1.4 数据类型

数据可以选择 UINT16、INT16、UINT32、INT32、FLOAT、DOUBLE、STRING 等类型。

数据个数该笔资料要采集数量(连续地址), 范围 1~2000

如数据地址= LW1001, 数据类型为 UINT16, 通道数目为 10, 则该笔资料采集的是 LW1001~LW100A 范围的数据。

4.1.5 小数设置

如传输值为 100，小数位数 1 位，则实际显示值为 10，支持设置为统一或单独：

- 小数位数表：在单独模式下设置；每个参数的缩放系数由“;”分割，如 0.1;0.2;0.3;0.4;...100;
- 显示无效 0:

4.1.6 触发地址

在触发采样模式下设置，填写触发寄存器的地址

- 模式：OFF->ON，上升沿触发采样（值如 0 到 1 变化）
ON->OFF，下降沿触发采样（值如 1 到 0 变化）
ON<->OFF，边缘触发采样
- 复位：自动覆盖该寄存器的值

4.1.7 控制地址

控制清除和导致的寄存器地址。在此地址写入 0x01，导出数据到 SD 卡。写入 0x0055，清除数据。

4.1.8 采样控制地址

选填地址，此地址的值控制采样使能的总开关。

4.1.9 掉电存储

数据保存到 FLASH 中,该存储方式是写在块地址中。

我司 M 系列彩屏的 Flash 大小为默认为 16M，一个块的单位为 4K。默认存储块地址 = 2048，即是从 8M 位置开始存储。占用块数，系统自动计算，如掉电存储最大条数=2000，自动计算得出占用块数=7，需要 28K 的空间。

实际应用中需要结合工程编译的大小来决定，如图 4-2 所示，编译的工程 SD 卡资源包为 13.9M，则存储地址需要与下载文件相隔 1M 空间，所以可以从 15M 开始存储，则存储块地址=(15 * 1024)/4，即为 3840，剩余 1M 的存储空间，剩余最大占用块数为 256。

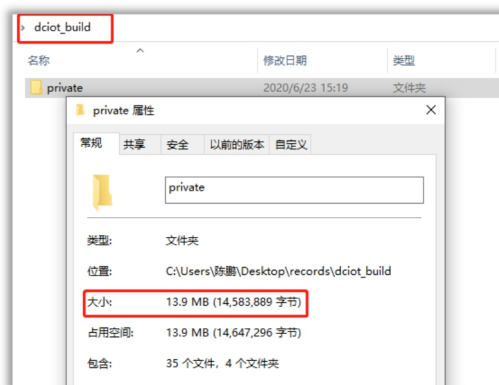


图 4-2 工程大小

5. 告警记录

本章节主要介绍常见的告警说明，如告警跑马灯（告警条）、当前告警、历史告警等。

5.1 告警显示

【告警记录】以表格形式，显示当前/历史的告警数据。



点击菜单栏中【控件】→【告警显示】告警显示，在画面中添加一个【告警显示】控件，如下图 5-1 所示。

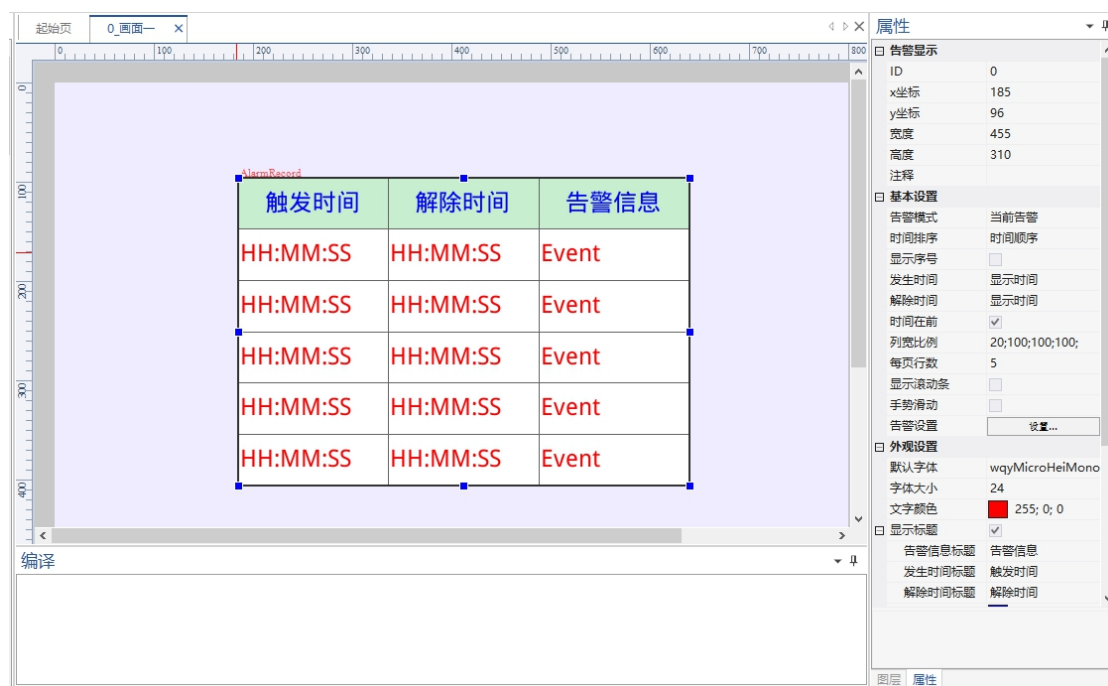


图 5-1 添加告警显示

5.1.1 基本设置

- 告警模式：可选当前告警、历史告警：
 - 当前告警：告警解除后，表格中不显示，不支持掉电存储；
 - 历史告警：告警解除后，仍记录下来，可支持掉电存储；
- 发生/解除时间：可选不显示、显示时间、显示日期、显示时间和日期。默认不显示，只显示告警内容，反之，将发生告警的时间拼接到告警后面；
- 时间在前：默认不勾选。勾选后，告警时间在前，告警内容在后；
- 列宽比例：调整每一列的宽度；
- 每页行数：显示的行数，不包括表头；
- 显示滑动条：勾选后，若告警内容小于“每页行数”时，不显示滚动条；若告警内容大于“每页行数”时，显示滚动条，并可以拖动滚动条查看数据；
- 告警设置：设置告警的内容，如下图 5-2 所示，详细说明参考“[4.5.3 告警设置](#)”

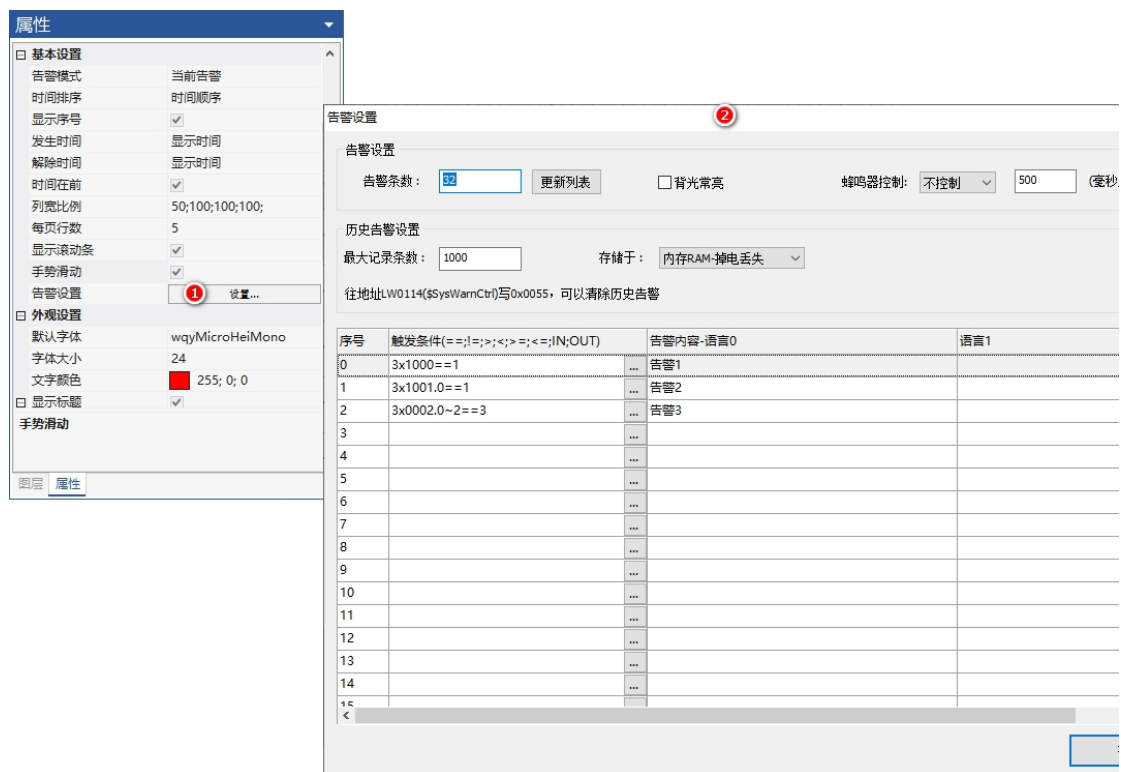


图 5-2 设置告警

5.1.2 外观设置

表格的外观可设置字体大小、文字颜色、标题、网格、背景等等，如图 5-3 所示。

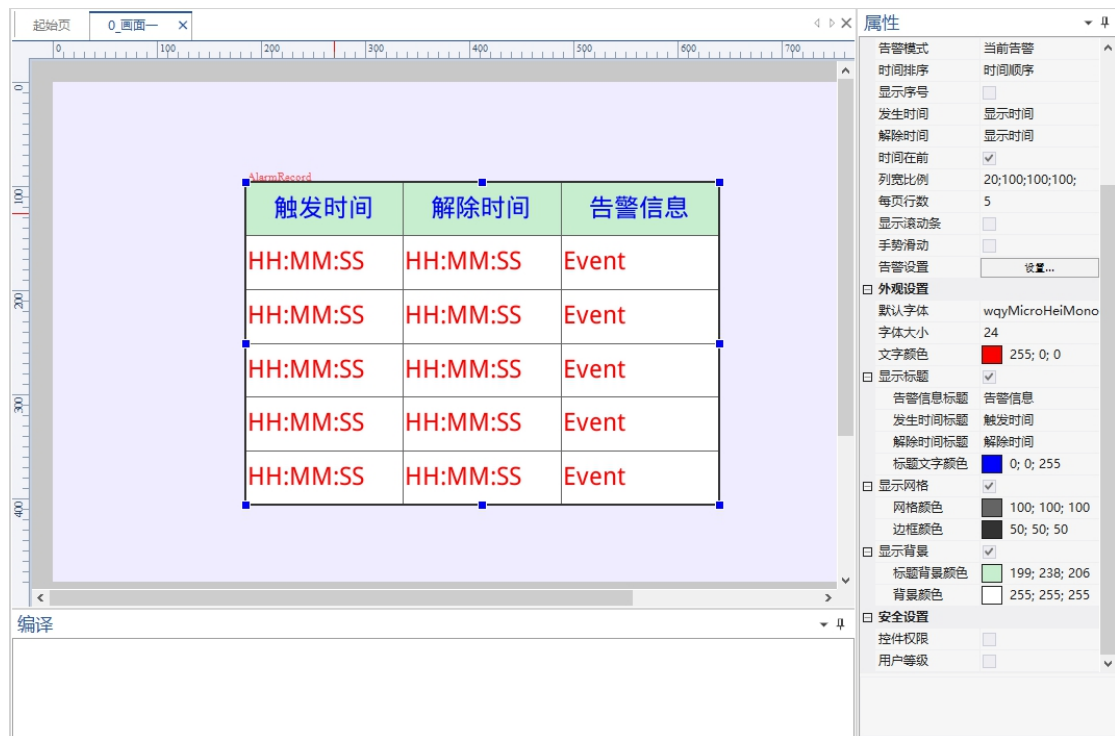


图 5-3 外观告警

5.2 告警条

【告警条】以跑马灯的方式，将所有当前告警从左往右滚动显示。




点击菜单栏中【控件】→【告警条】，在画面中添加一个【告警条】控件，如下

图 5-4 所示。

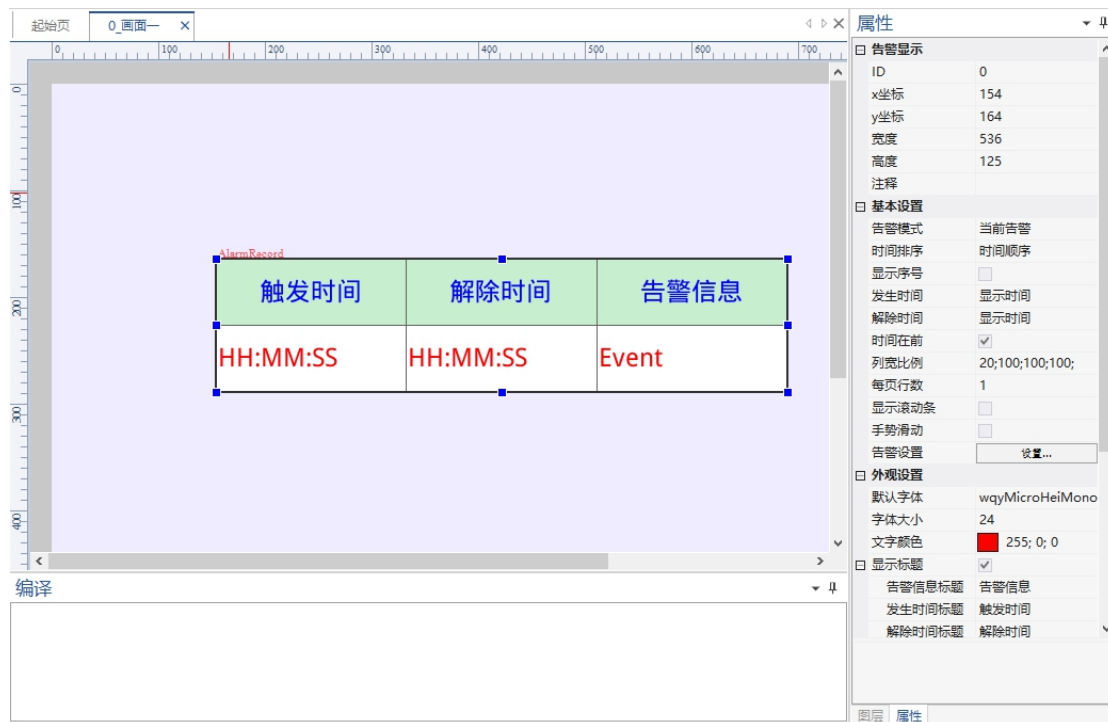


图 5-4 设置告警条

5.2.1 基本设置

1. 告警排序：可选时间顺序、时间逆序；
2. 告警时间：可选不显示、显示时间、显示日期、显示时间和日期。默认不显示，只显示告警内容，反之，将发生告警的时间拼接到告警内容后面；
3. 移动速度：告警内容从左往右滚动显示，单位毫秒（ms）；
4. 告警设置：设置告警的内容，如下图 5-5 所示，详细说明参考“[5.3 告警设置](#)”；

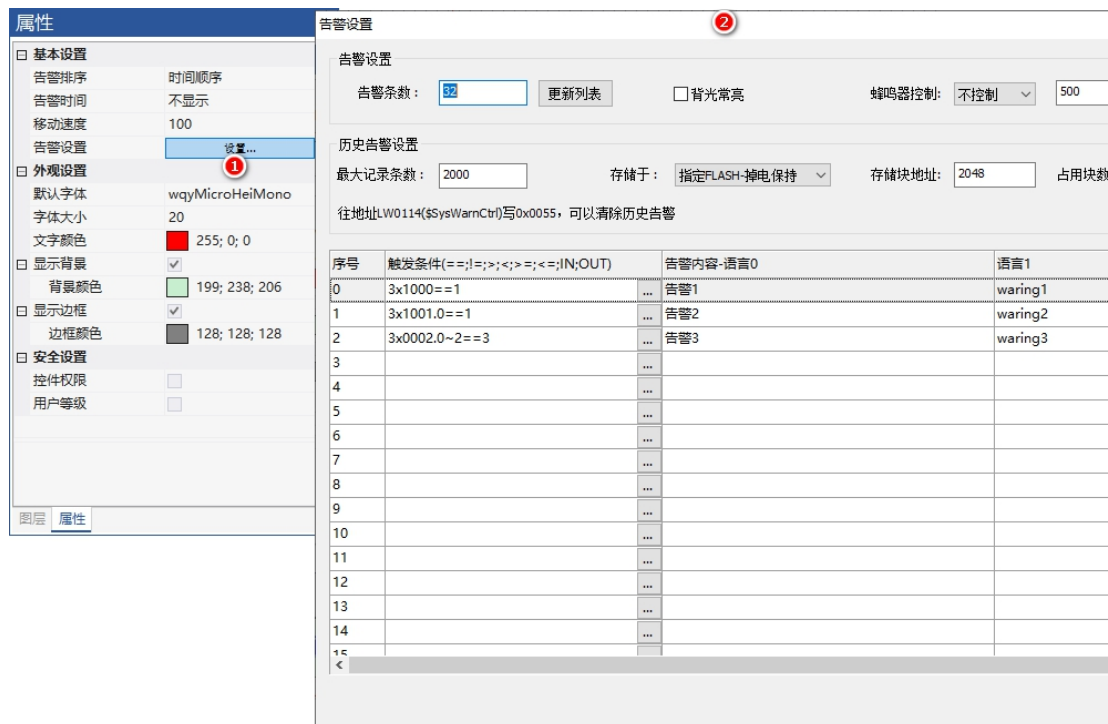


图 5-5 设置告警

5.2.2 外观设置

告警条的外观可设置字体大小、文字颜色、边框、背景等等，如图 5-6 所示。

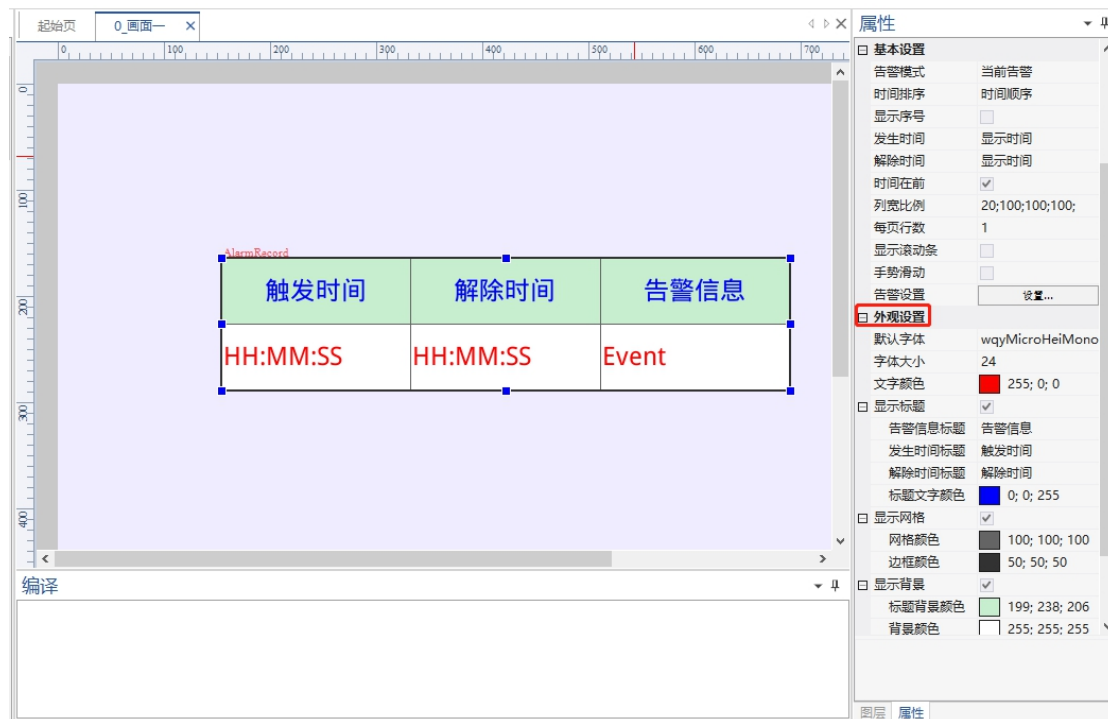


图 5-6 外观设置

5.3 告警设置

【告警设置】可设置告警内容、存储方式、告警通知方式（点亮背光、蜂鸣器）等，如

图 5-7 所示。除了在告警条、告警显示控件中进入，还可以在菜单工程栏→【告警设置】

进入 告警设置。

序号	触发条件(=, >, <, >=, <=, !N, OUT)	告警内容-语言0	语言1
0	3x1000=1	告警1	waring1
1	3x1001.0=1	告警2	waring2
2	3x0002.0~2=3	告警3	waring3
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

图 5-7 告警设置

1. 告警条数：最大支持 10000 条，点击更新，自动给序号（告警 ID）编号；
2. 背光：告警触发时点亮背光；
3. 蜂鸣器控制：控制方式可设置不控制、滴滴声、长鸣。鸣叫时间单位 ms；
4. 最大记录数：最大支持 10000 条；
5. 存储方式：
 - 内部 RAM-掉电丢失；
 - 专用 Flash 掉电存储：最大 1000 条；
 - 指定 Flash-掉电存储：最大条数，受限于剩余 Flash 的空间大小，如下图 5-8 所示；

历史告警设置

最大记录条数： 2000 存储于： 指定FLASH-掉电保持 存储块地址： 2048 占用块数： 7

往地址LW0114(\$SysWarnCtrl)写0x0055，可以清除历史告警

图 5-8 大小限制

我司 M 系列彩屏的 Flash 大小为默认为 16M，一个块的单位为 4K。默认存储块地址 = 2048，即是从 8M 位置开始存储。占用块数，系统自动计算，如掉电存储最大条数=2000，自动计算得出占用块数=7，需要 28K 的空间。

实际应用中需要结合工程编译的大小来决定，如下图 5-9 所示，编译的工程 SD 卡资源包为 13.9M，则存储地址需要与下载文件相隔 1M 空间，所以可以从 15M 开始存储，则存储块地址=(15 * 1024)/4，即为 3840，剩余 1M 的存储空间，剩余最大占用块数为 256

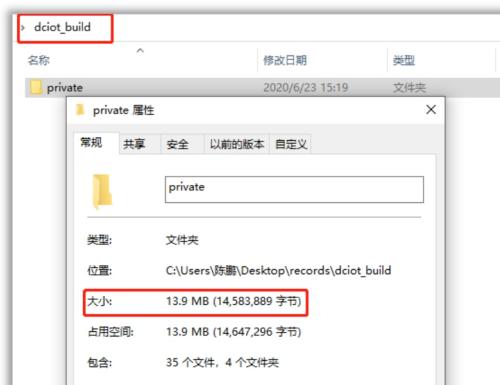


图 5-9 工程大小

- 清除历史告警：往系统变量 SysWarnCtrl (LW0114)写入 0x0055，即可清除告警信息；
- 6. 序号：若告警条数为 32，即告警 ID 为 0~31
- 7. 触发条件：支持基本的逻辑运算 “>”、“<”、“>=”、“<=”、“==”、“!=”、“IN”（相当与 $a \leq x \leq b$ ）、“OUT”（相当与 $a < b$, $x < a$, $x > b$ ）。PS：“.”表示引用位，“~”位的区间范围
 - 设定寄存器等于某个值触发：3x1000==1，如下图 5-10 所示
 - 设定寄存器某一个位的值触发：3x1001.0==1；
 - 设定某几个连续位等于某个值触发：3x0002.0~2==3；寄存器 0x0002,低 3 位等于 3 触发告警。

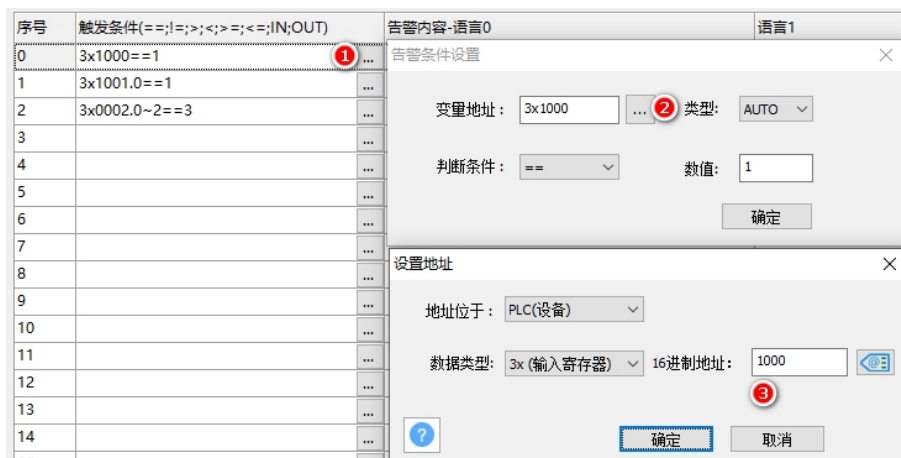


图 5-10 设定寄存器

- 8. 告警内容：告警条件成立时，触发显示的告警内容。当系统语言大于 2，可直接对不同的语言编辑告警内容。

6. 安全设置（密码登录）

VisualHMI 软件的安全设置功能，可设置用户操作控件的权限。

1. 控件权限：对地址的值进行判断，从而实现控制控件在画面中“显示/隐藏”或者“禁止/使能”触摸；
2. 用户等级：对控件功能操作的权限等级，当操作权限低，需要输入密码才可以继续操作；

注意：本章节使用功能按钮控件为例子，其他控件均可参考。

6.1 控件权限

控件权限设置是通过变量的值进行判断，当条件不满足时，可以控制控件在画面中“显示/隐藏”或者“禁止/使能”触摸。

假设需要设定地址“\$value” (地址标签)等于 0 时显示控件，地址“value”不等于 0 时隐藏控件。那么该控件权限的设置如图 6-1 控件权限所示

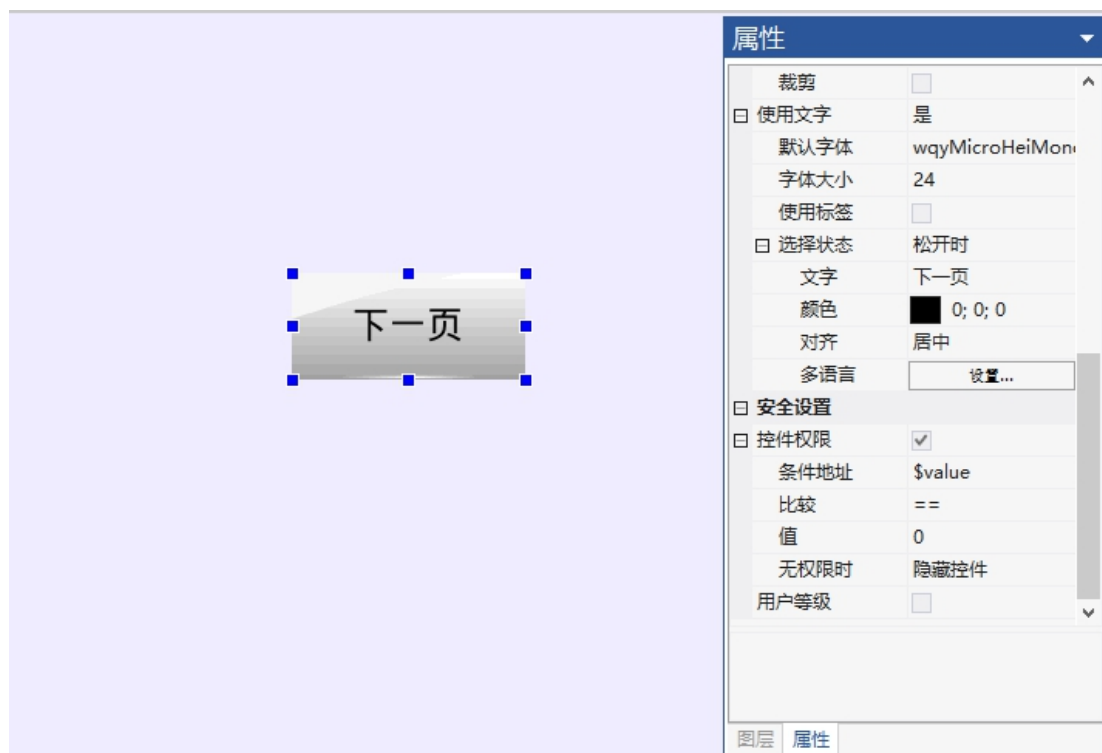


图 6-1 控件权限

6.2 用户等级

在 VisualHMI 软件中用户等级分为 0~8 级，0 级权限最低不需要输入密码，8 级权限最高级，工程所有的操作默认为 0 级（0 级不需要密码登录）。

用户等级：可以根据需求规划权限控制，当开启用户等级后，若控件的操作等级权限低，需要通过键盘输入密码，才能进行下一步操作。

6.2.1 用户等级设定

在【工程设置】的“安全登录”中，可以设定当前工程需要开启的最高等级，以及设定每个等级的登录密码，如图 6-2 所示

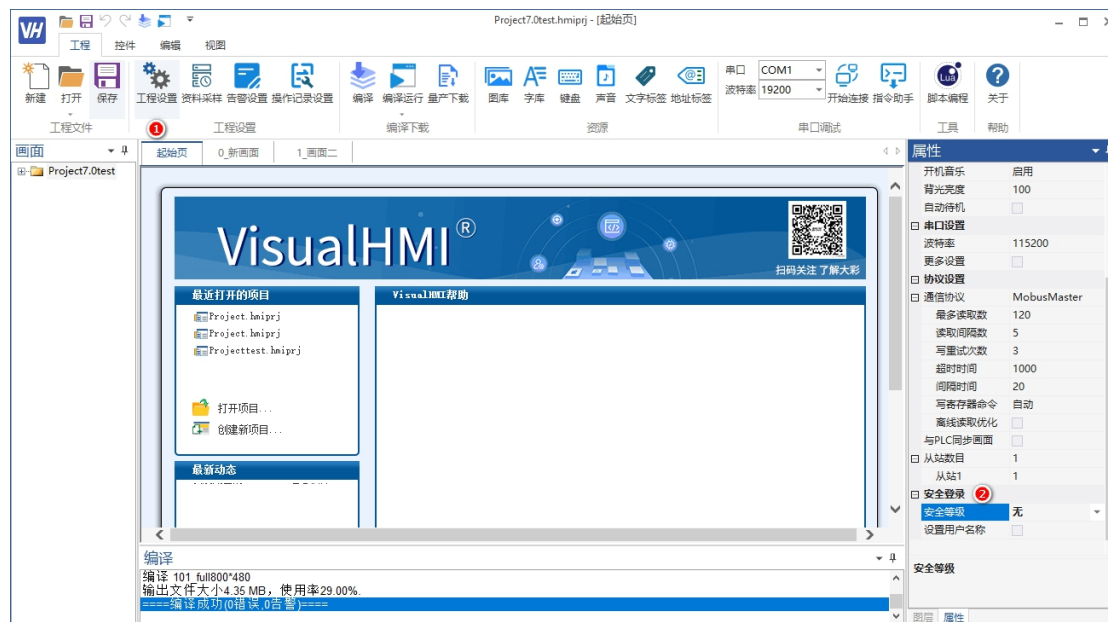


图 6-2 安全登录

假设设定开启三级用户等级，如图 6-3 所示。

- 安全等级选择 3 级。
 - 1 级用户密码：111111；
 - 2 级用户密码：222222；
 - 3 级用户密码：333333；
- 自动退出登录选择开启，设定屏幕无操作 3s 后返回“画面一”；
 - 超时时间：3 s；
 - 返回画面：0_画面一；
- 用户名称设置：设定每个用户的名称；

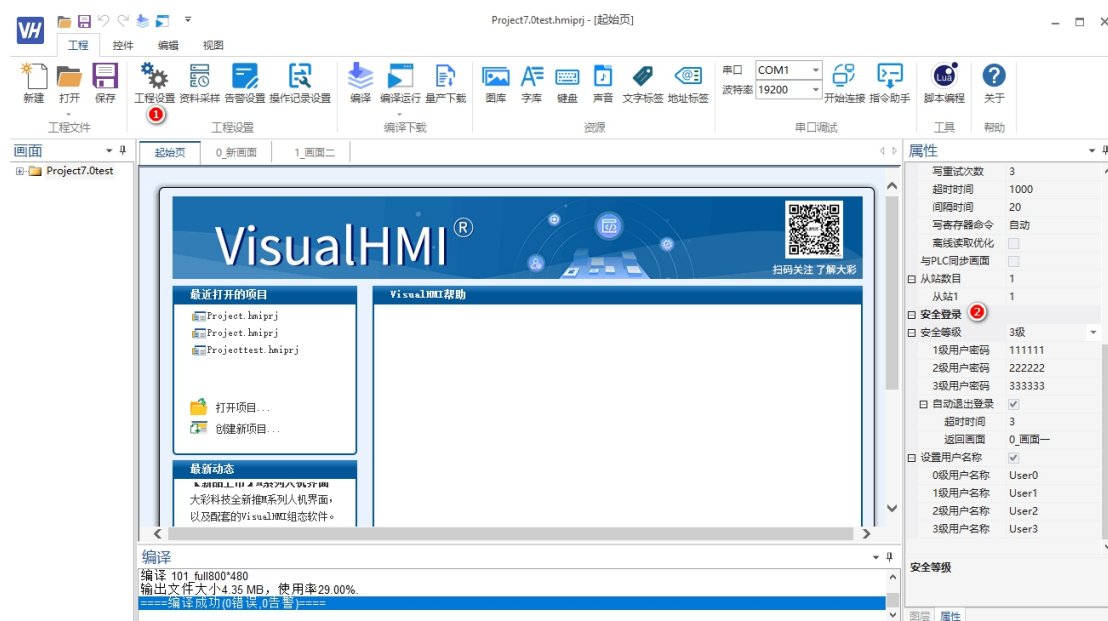


图 6-3 3 级用户

6.2.2 控件开启用户等级

当控件开启用户等级后，低于指定等级的用户，操作时都需要输入密码。如图 6-4 所示；以切换画面功能为例：

- 点击控件的属性，设置画面切换功能；
- 开启用户等级权限，设定要求等级 2 以上不需要密码输入；

注意：设定的用户等级要求不能超过工程设定的安全等级，否则该控件无正确密码

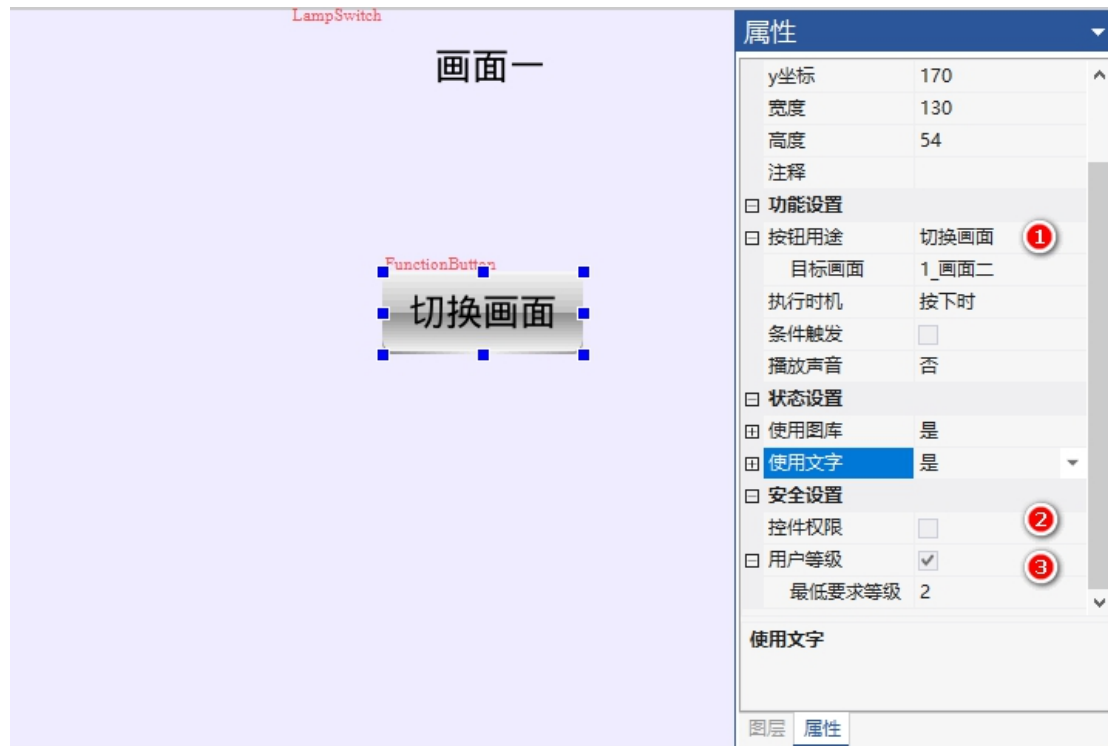


图 6-4 开启用户登陆



设定后点击菜单栏中【编译运行】按钮运行工程测试，效果如下图 6-5

- 点击画面中切换画面按钮；
- 输入正确的密码，按回车键确认后进入画面二；

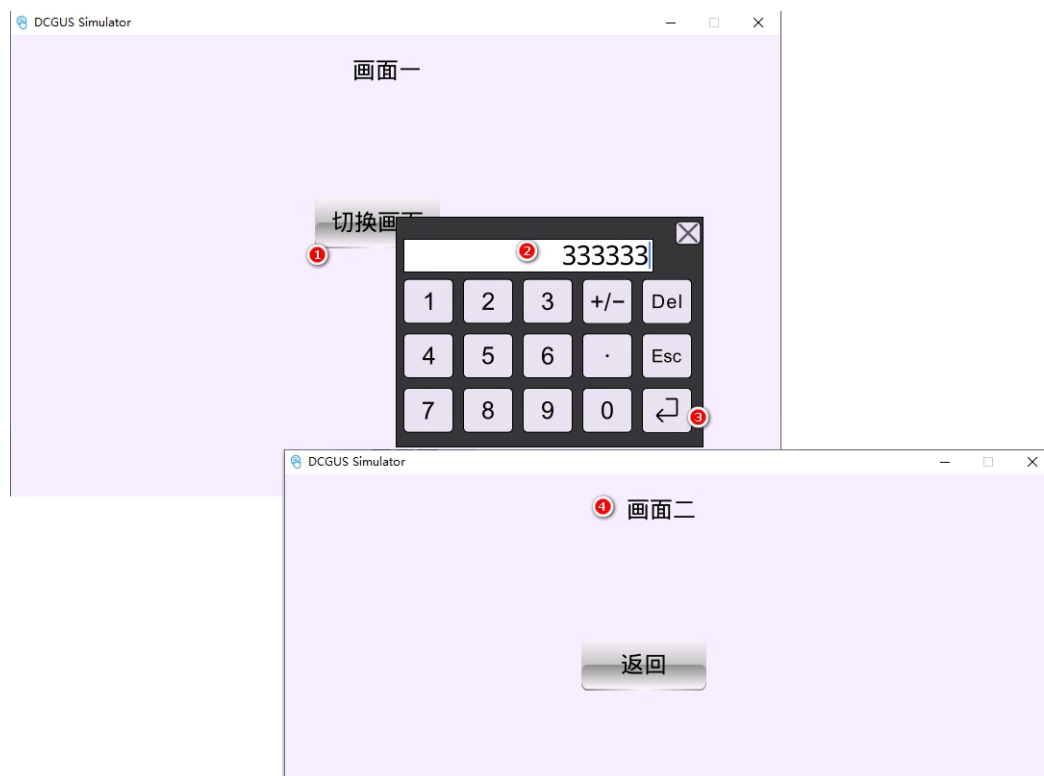


图 6-5 效果体验

6.2.3 撤销用户登录状态

撤销登录状态有两种方法：

1. 开启自动退出登录，在设定的超时时间后返回指定画面，并自动撤销登录状态；
2. 手动撤销，手动向系统地址“SysLoginLevel”填入 0，将当前用户等级登录状态更新 0 级；

6.2.4 修改安全密码

安全等级密码属于特殊、重要参数，需要在脚本处理，调用 API 进行修改：

```
--[[/*
API:set_pwd(index,pwd)
● index:安全等级索引,从0开始
● pwd:密码,UINT32
*/--]]
--设置安全等级的密码
set_pwd(0,98765)
```

若需要修改后掉电存储，需要配合系统寄存器 \$syscfgctrl(系统参数控制)、\$syscfgoption(系统参数选择)使用。若修改分期密码后，需要读取查看，用 RW 寄存器绑定显示。程序如下所示：

\$syscfgoption: 配合 syscfgctrl 使用：

- bit0-声音 syscfg0；
- bit1-多语言 syslang；

- bit2-音频音量大小 SysSndVol;
- bit3-背光设置;
- bit4-用户密码;
- bit5-分期使用参数;

\$SysCfgCtrl: 系统参数控制

- 写 0x5501 保存参数;
- 写 0x5502 加载参数;
- 写 0x5503 清除参数;

```
function on_init()
    --加载安全等级参数
    set_uint16(VT_LW, 0x011B, (1<<4))
    set_uint16(VT_LW, 0x011A, 0x5502)

    if get_uint32(VT_RW, 0xF000) ~= 1234567890 --判读是否是首次上电使用
    then
        set_uint32(VT_RW, 0xF000, 1234567890)

        --将 HMI 软件配置的默认密码复制给 RW 寄存器, 作为初始化
        local addr = 0x0100
        for i = 1, 8
        do
            set_uint32(VT_RW, addr, (100000*i+10000*i+1000*i + 100*i + 10*i + 1*i))
            addr = addr + 2
        end
    end
end


function on_update(slave,vtype,addr)
    if vtype == VT_RW
    then
        --/*安全等级、密码修改*/--
        if addr >= 0x0100 and addr <= 0x010E
        then
            local pwd = get_uint32(VT_RW, addr)
            set_pwd(((addr - 0x0100) // 2), pwd)
            set_uint16(VT_LW, 0x011B, (1<<4))
            set_uint16(VT_LW, 0x011A, 0x5501)
        end
    end
end
```


7. 配方设置

在工控行业，配方常用于以下两种场景：

- 一种是不同的产品在工艺、生产制作流程，使用不同的参数；
- 一种是在生产一种产品过程中，有很多流程，每个步骤都是有不同的参数；



点击菜单栏中【工程】→【工程设置】→配方设置 ，打开配方管理，最多 10 类配方，每类配方最多支持 1000 条配方记录数。配方成分最多包含 2000 个参数，如图 7-1 所示。

1. 配方成分数：最多 2000 个（连续寄存器）
2. 配方类型：UINT16、INT16、UINT32、INT32、UINT64、INT64、FLOAT、DOUBLE；
3. 配方记录数：最大支持 1000 条
4. 配方索引地址：如 0x1201。则配方成分数对应的地址为：0x1202~0x19D2（2000 个）
5. 配方写入 PLC 地址：若操作写入时，会将 0x1202~0x19D2 的数据写到 0xE000
6. 索引改变时候写入：是与否
7. 配方控制地址：
 - 往 0x1200 写 1，写入 PLC 地址；
 - 往 0x1200 写 2，保存并修改配方值。

配方管理

启用	配方名称	配方成分	配方记录数	配方缓存地址	写入PLC地址
<input checked="" type="checkbox"/>	0	UINT16*2000	1000	LW1201	LWE000
<input type="checkbox"/>	1	UINT16*1	1		
<input type="checkbox"/>	2	UINT16*1	1		
<input type="checkbox"/>	3	UINT16*1	1		
<input type="checkbox"/>	4	UINT16*1	1		
<input type="checkbox"/>	5	UINT16*1	1		
<input type="checkbox"/>	6	UINT16*1	1		
<input type="checkbox"/>	7	UINT16*1	1		
<input type="checkbox"/>	8	UINT16*1	1		
<input type="checkbox"/>	9	UINT16*1	1		

配方设置

配方名称: _____

1 配方成分数: 2000

2 数据类型: UINT16

3 配方记录数: 1000

配方记录数据: 设置...

4 配方索引地址: LW1201

5 配方写入PLC地址: LWE000

6 索引改变时写入: ☐

7 配方控制地址: LW1200

配方数据设置

序号	参数0	参数1	参数2	参数3	参数4	参数5	参数6	参数7	参数8	参数9	参数10	参数11	参 ^
0	1	2	3	4	5	6	7	8	9	10	11	12	1:
1	110	120	130	140	150	160	170	180	190	200	210	220	2:
2	2	0	0	0	0	0	0	0	0	8	0	0	0
3	3	0	0	0	0	0	0	0	0	7	0	0	0
4	4	0	0	0	0	0	0	0	0	6	0	0	0
5	5	0	0	0	0	0	0	0	0	5	0	0	0
6	6	0	0	0	0	0	0	0	0	3	0	0	0
7	7	0	0	0	0	0	0	0	0	2	0	0	0
8	8	0	0	0	0	0	0	0	0	2	0	0	0
9	9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0

确定


图 7-1 配方设置

8. 分期使用

HMI 系统时间到截止分期时间，HMI 会弹出分期密码输入的画面，用户输入该期的分期密码后，才可以继续正常使用。否则，一直显示在分期密码输入画面。

8.1 属性介绍



点击菜单栏中【工程】→【工程设置】分期使用 ，打开分期使用，如图 8-1 所示：

分期使用

×

☒ 启用分期：到期后自动锁定HMI，需要输入密码解锁。

分期数：

10

▼

到期锁定画面ID：

2

序号	到期日	密码
1	2022/ 9/14 ▼	123456
2	2022/ 9/23 ▼	111111
3	2022/ 9/30 ▼	222222
4	2022/10/13 ▼	333333
5	2022/11/13 ▼	444444
6	2022/12/13 ▼	555555
7	2023/ 1/13 ▼	666666
8	2023/ 2/13 ▼	777777
9	2023/ 3/13 ▼	888888
10	2023/ 4/13 ▼	999999

确定

图 8-1 分期属性

1. 启用分期：使能后才支持分期功能。
2. 分期数：最大支持 10 个分期。
3. 到期锁定画面 ID：填入对应的画面 ID。
4. 到期日：假设当前 HMI 的时间小于 分期 1（2022/9/14）。则到分期 1 时间截止前，需要输入密码才可以使用。
5. 密码：各个分期时间的密码。

8.2 系统参数

分期相关的系统参数如下所示：

ADDR	NAME	RW	TIPS	DETAILS
0x0200	SysStageCount	R	分期数(最多 10 组)	
0x0201	SysStageLockStatus	R	分期锁定状态	当前分期是否处于锁定状态, 0 未锁定, 1 到期锁定, 2 时间异常锁定
0x0202	SysStageLockScreen	R	分期锁定画面	锁定时自动切换到此画面
0x0203	SysStageUnlockLevel	R	当前已解锁分期编号	
0x0204	SysStageTimeLevel	R	当前时间所处分期编号	
0x0205	SysStageEnterPwd	W	用户输入的分期密码	输入正确密码, 自动解锁当前分期
0x0210	SysStageTime	RW	分期截止时间数组 (UINT32)	最多 10 组

假设当前 HMI 的时间为“2022/9/13”，处于分期 0（分期使用界面序号 1）“2022/9/14”，则表示“2022/9/13”~“2022/9/14”这段时间限制用户使用，判定为锁定，切换到画面 2，需要输入分期 0 的密码，如图 8-2 所示：

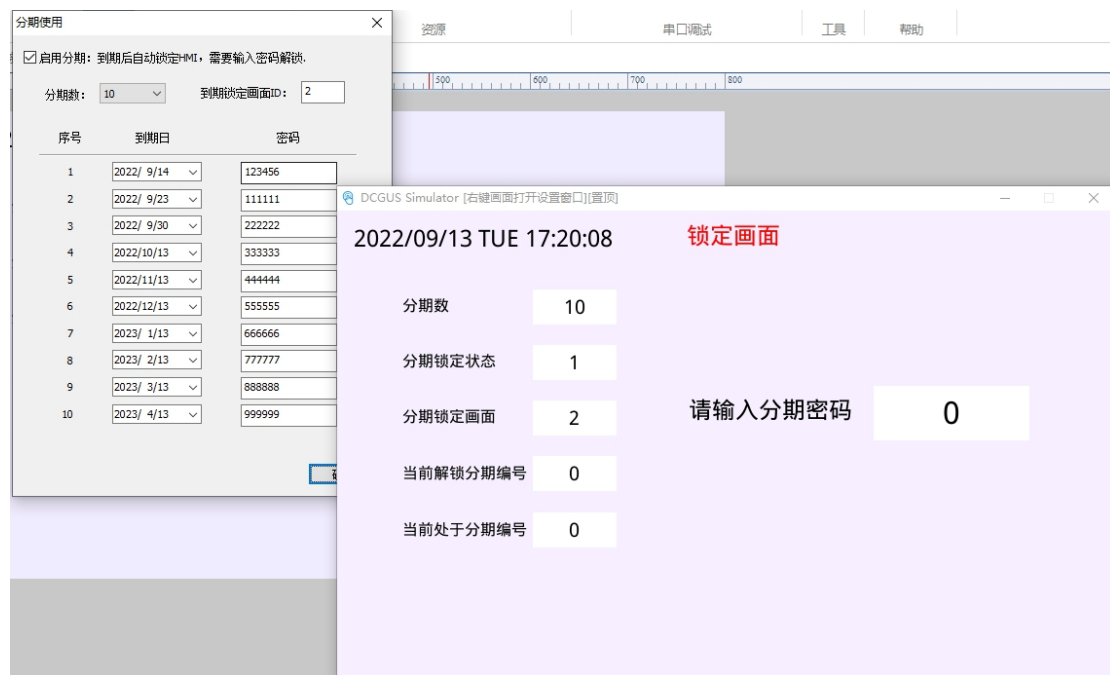


图 8-2 分期密码输入

当输入分期 0 的密码“123456”，密码一致，自动解锁，进入工作模式。则当前解锁分

期编号为 1。同理，当 HMI 时间处理“2022/9/14”，进入分期 1（分期使用界面序号 2），再次进入锁定画面，需要输入分期 1 的密码“111111”进行解锁。

8.3 修改分期密码

分期密码属于特殊、重要参数，需要在脚本处理，调用 API 进行修改：

```
--[[/*
API:set_stage_pwd(index,pwd)
● index:分期索引，从 0 开始
● pwd:密码，UINT32
*/--]]
--设置分期 1 的密码
set_stage_pwd(0,98765)
```

若需要掉电存储，需要配合系统寄存器 \$SysCfgCtrl(系统参数控制)、\$SysCfgOption(系统参数选择)使用。若修改分期密码后，需要读取查看，用 RW 寄存器绑定显示，如图 8-3 所示。程序如下所示：

\$SysCfgOption: 配合 SysCfgCtrl 使用：

- bit0-声音 SysCfg0;
- bit1-多语言 SysLang;
- bit2-音频音量大小 SysSndVol;
- bit3-背光设置;
- bit4-用户密码;
- bit5-分期使用参数;

\$SysCfgCtrl: 系统参数控制

- 写 0x5501 保存参数;
- 写 0x5502 加载参数;
- 写 0x5503 清除参数;

```
function on_init()

--加载分期参数
set_uint16(VT_LW, 0x011B, (1<<5))
set_uint16(VT_LW, 0x011A, 0x5502)

if get_uint32(VT_RW, 0xF000) ~= 1234567890 --判读是否是首次上电使用
then
    set_uint32(VT_RW, 0xF000, 1234567890)

--将 HMI 软件配置的默认密码复制给 RW 寄存器，作为初始化
set_uint32(VT_RW, 0x0000, 123456)
local addr = 0x1238
for i = 1, 9
do
    addr = addr + 2
    set_uint32(VT_RW, addr, (100000*i+10000*i+1000*i +100*i+10*i+i))
```

```

        end
    end
end

function on_update(slave,vtype,addr)
    if vtype == VT_RW
    then
        --/*分期密码修改*/--
        if addr >= 0x0000 and addr <= 0x012
        then
            local pwd = get_uint32(VT_RW, addr)
            set_stage_pwd(((addr - 0x0000) // 2), pwd)
            --保存分期参数
            set_uint16(VT_LW, 0x011B, (1<<5))
            set_uint16(VT_LW, 0x011A, 0x5501)
        end
    end
end
end

```

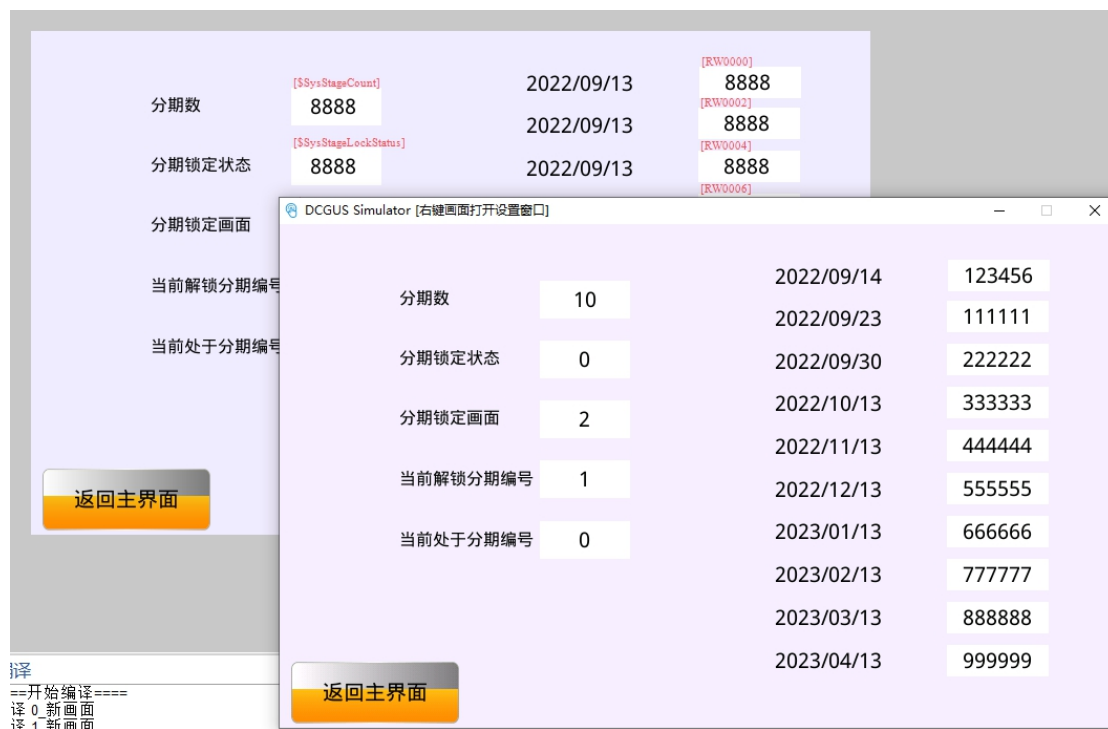


图 8-3 分期密码 RW

9. 静态控件

绘图的基本控件包括直线、矩形、圆形、文字、图片，如下图 9-1 所示

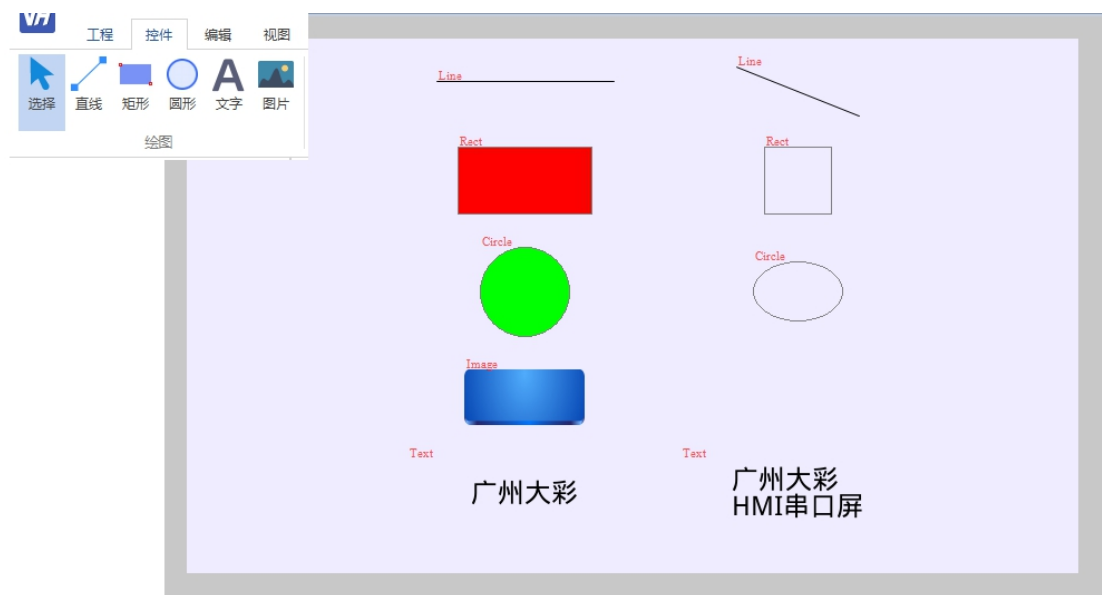


图 9-1 绘图

9.1 直线

直线控件，可编辑颜色：单色；宽度：1~4（像素），拖动线条两端，可以直接拉长和改变直线方向，如图 9-2 所示

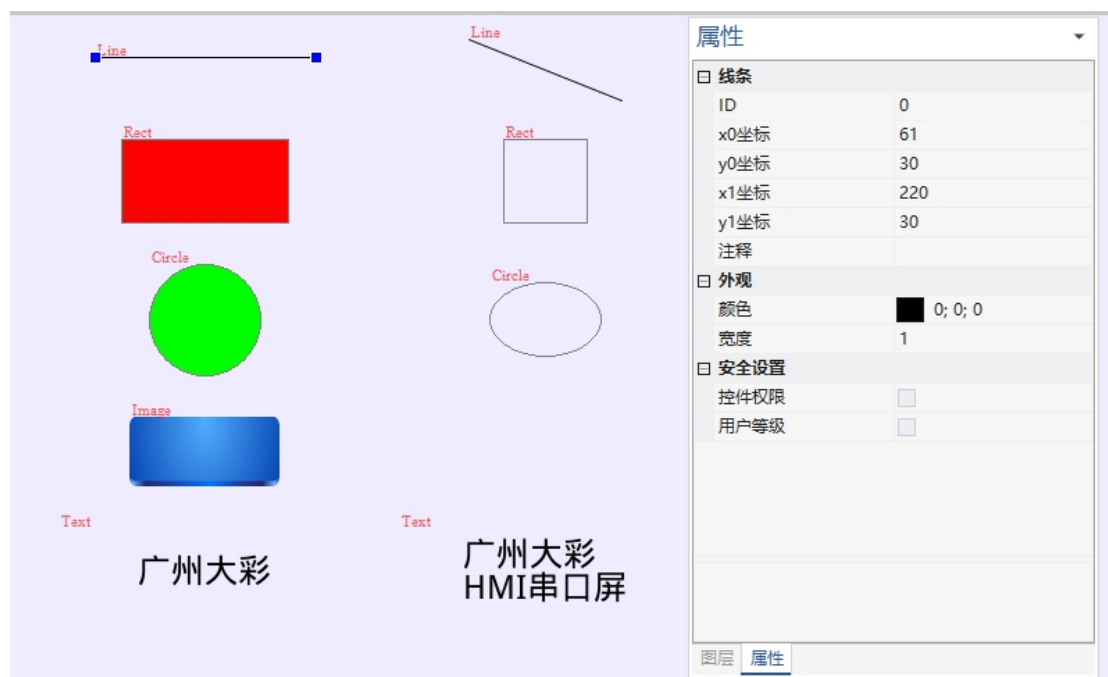


图 9-2 直线

9.2 矩形

矩形控件，可编辑颜色：单色；宽度：1~4（像素）；填充颜色：纯色；改变控件的长宽，可以变成长方形或矩形，如图 9-3 所示。

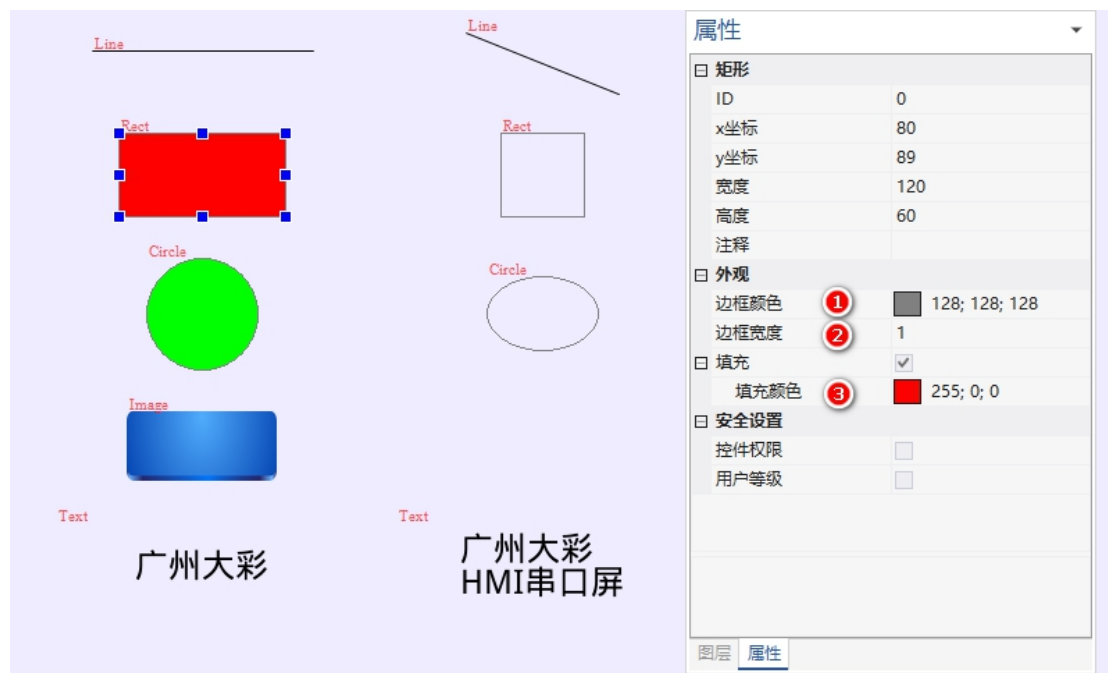


图 9-3 矩形

9.3 圆形

圆形控件，可编辑颜色：单色；宽度：1~4（像素）；填充颜色：纯色；改变控件的长宽，可以变成圆形或椭圆形，如图 9-4 所示

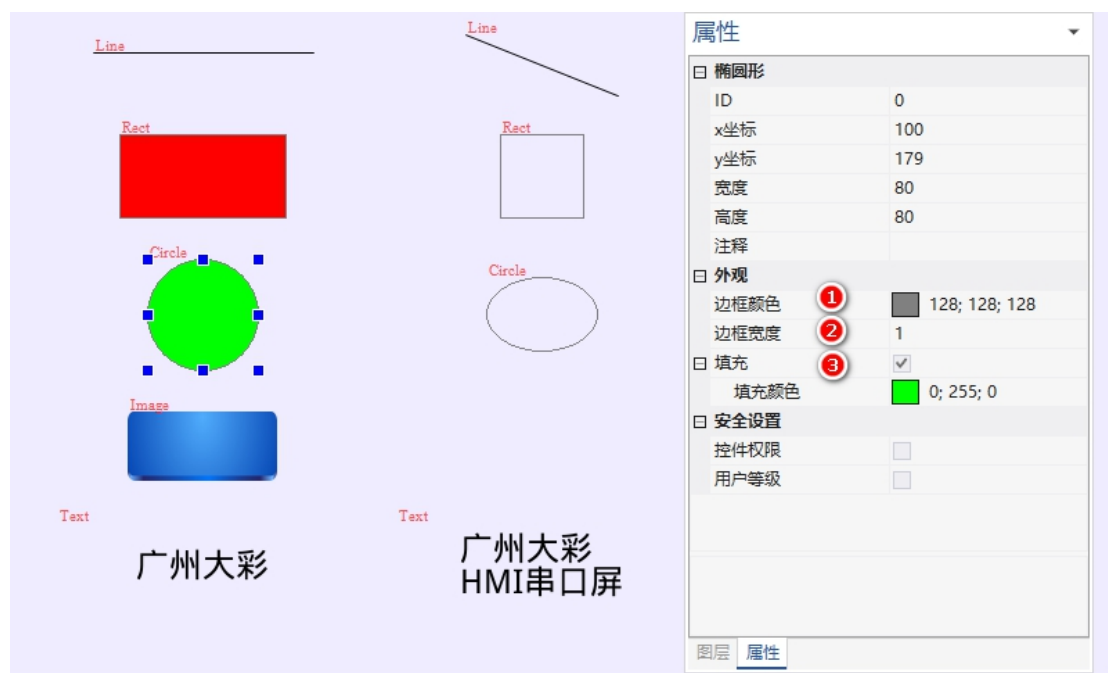


图 9-4 圆形

9.4 文字

文字控件，常用于静态字符显示，做多语言功能等。可编辑字体的颜色、大小、对齐方式等；如图 9-5 所示，标签应用，可参考“[第 13 章节](#)”

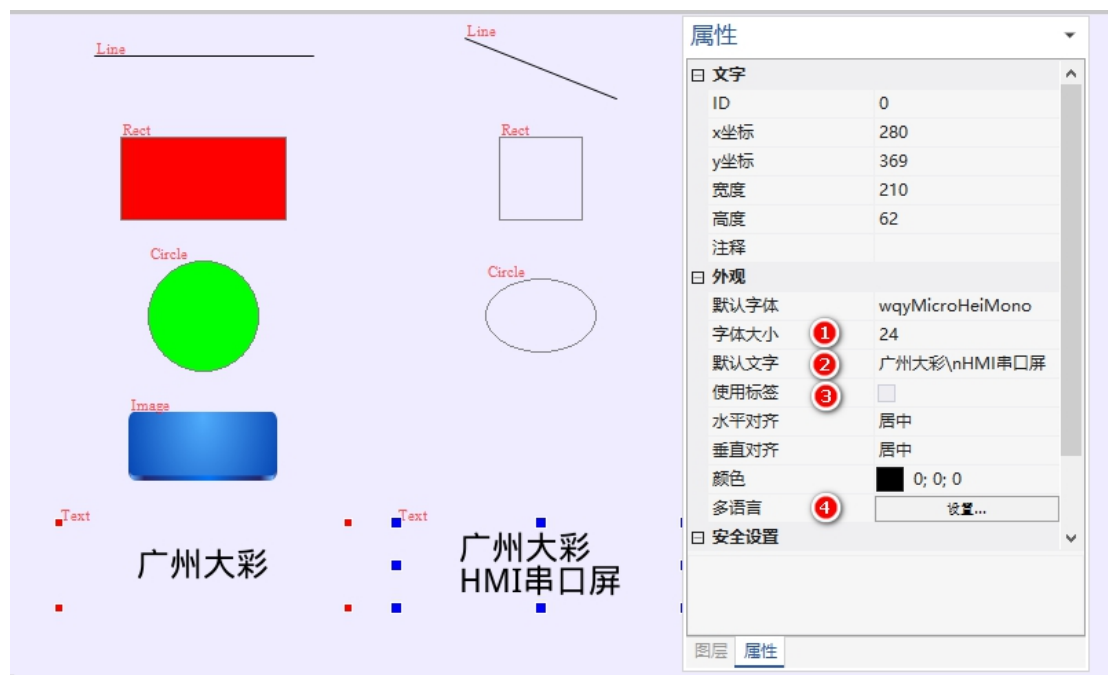


图 9-5 文字

9.5 图片


图片控件，常用于静态图片显示，需要关联图库。如图 9-6 所示，图库说明可参考“[第 9 章节](#)”



图 9-6 图库

10. 图库



菜单工程栏→图库  ,VisualHMI 所有控件使用的图片都在图库中添加和管理。其中包括工程图库（用户自定义）、系统图库（系统自带常见的按钮、开关、进度条、键盘等图片），如图 10-1 所示。

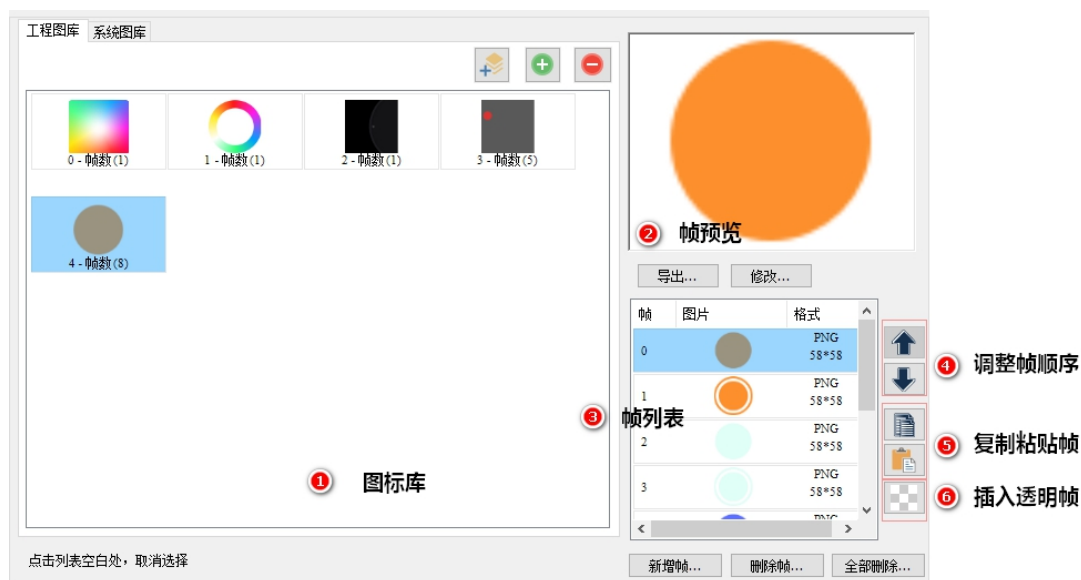


图 10-1 图库

10.1 添加

工程图库的添加支持 svg、png、jpg 格式的图片文件。

10.1.1 批量添加

【批量添加】通常用于生成仅一帧的图标，如下图 10-2 所示。

1. 点击批量添加按钮；
2. 选择对应的 UI 图片文件，如选择 4 张图片；
3. 生成 4 个图标，一张图片对应一个图标；

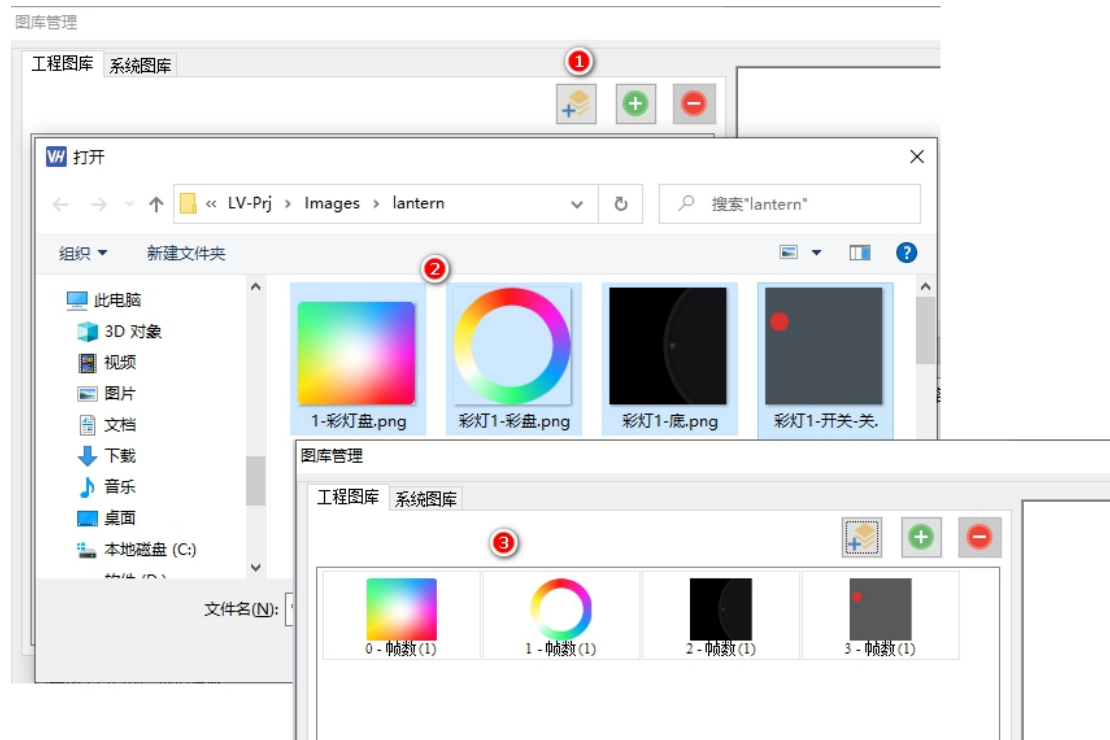


图 10-2 添加图库

10.1.2 添加单个图标

【添加单个图标】通常用于生成多帧的图标，如下图 10-3 所示

1. 点击添加单个图标按钮；
2. 选择对应的 UI 图片文件，如选择 8 张图片；
3. 生成 1 个图标，含有 8 帧图片；

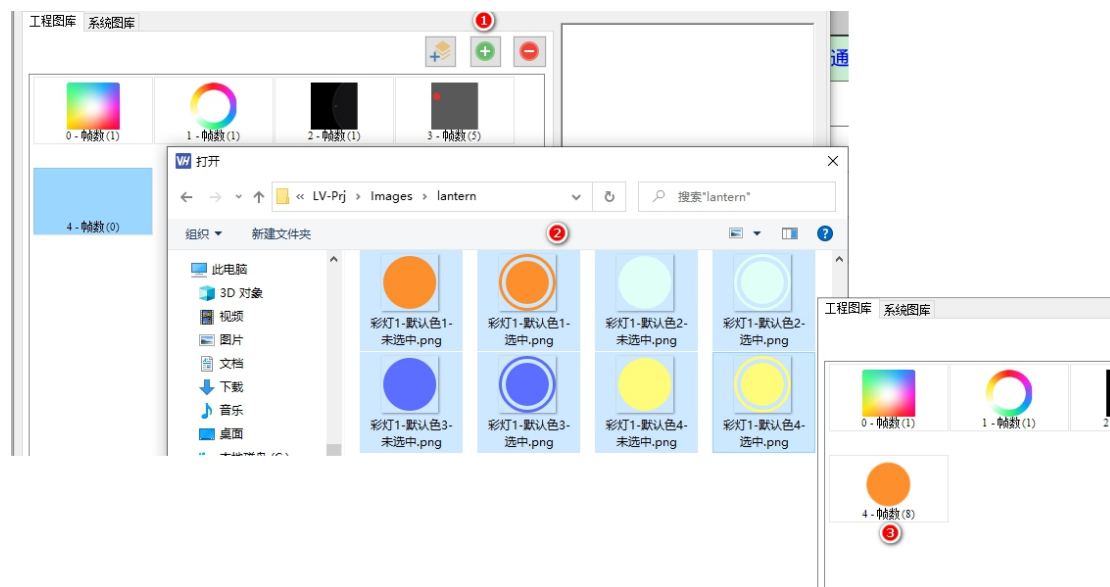


图 10-3 添加图标

10.2 修改

已有的图标文件支持增添帧、替换帧、删除帧等操作；

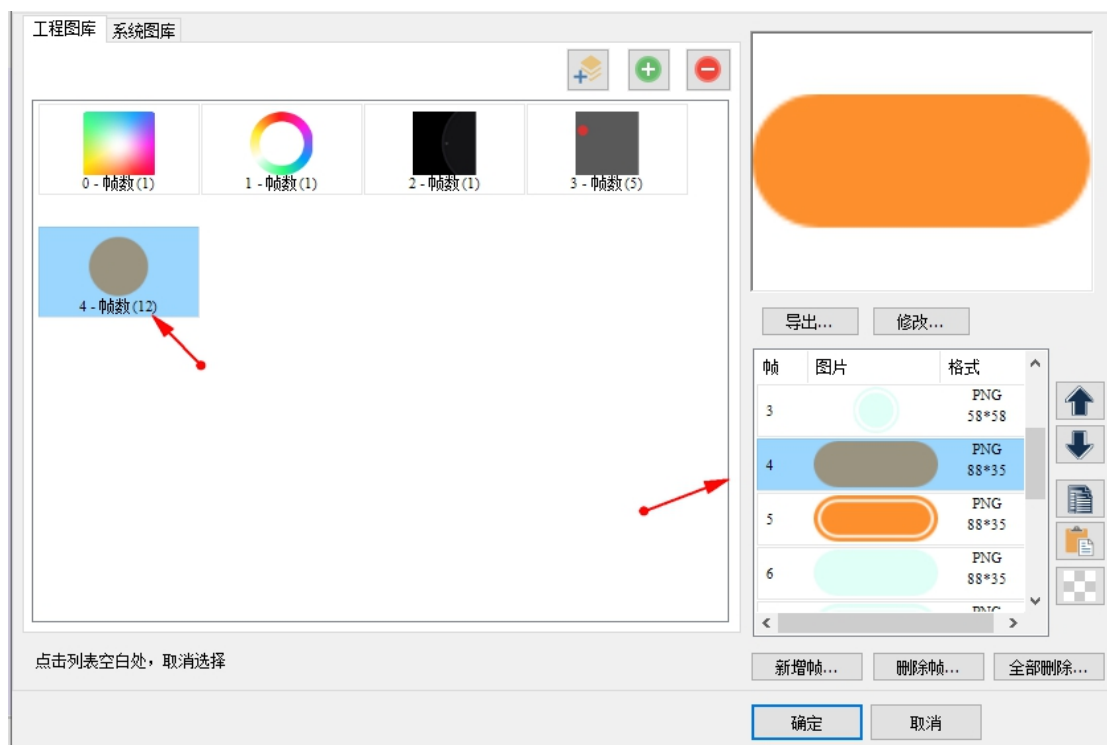


图 10-6 插入图片

10.2.2 替换帧

同时还支持对已有的图标继续替换图片，如下图 10-7 所示。

1. 选中图标 4，点击第 0 帧，点击修改；

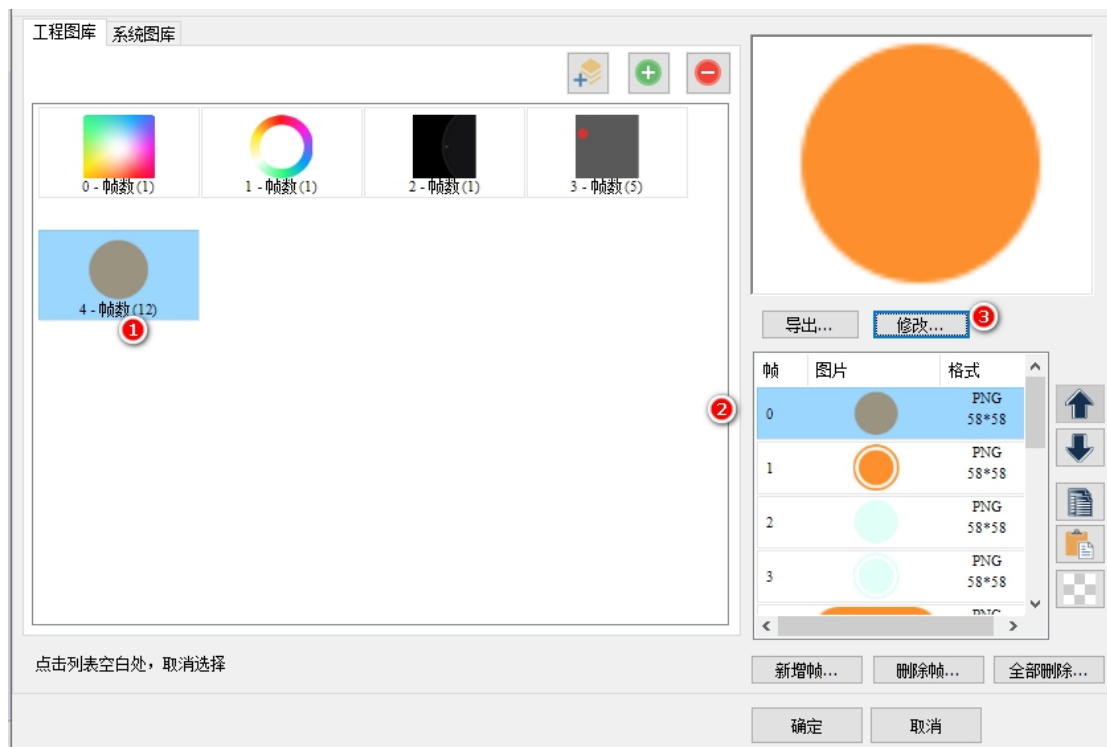


图 10-7 修改图库

2. 替换 UI：如下图 10-7 所指示；

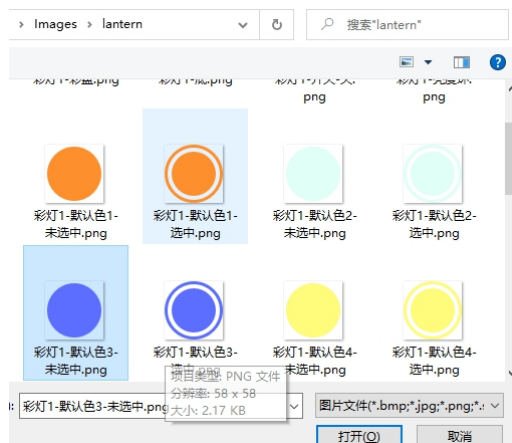


图 10-8 替换图标

3. 图标 4 第 0 帧黄色指示灯变为蓝色，如下图 10-9 所示；

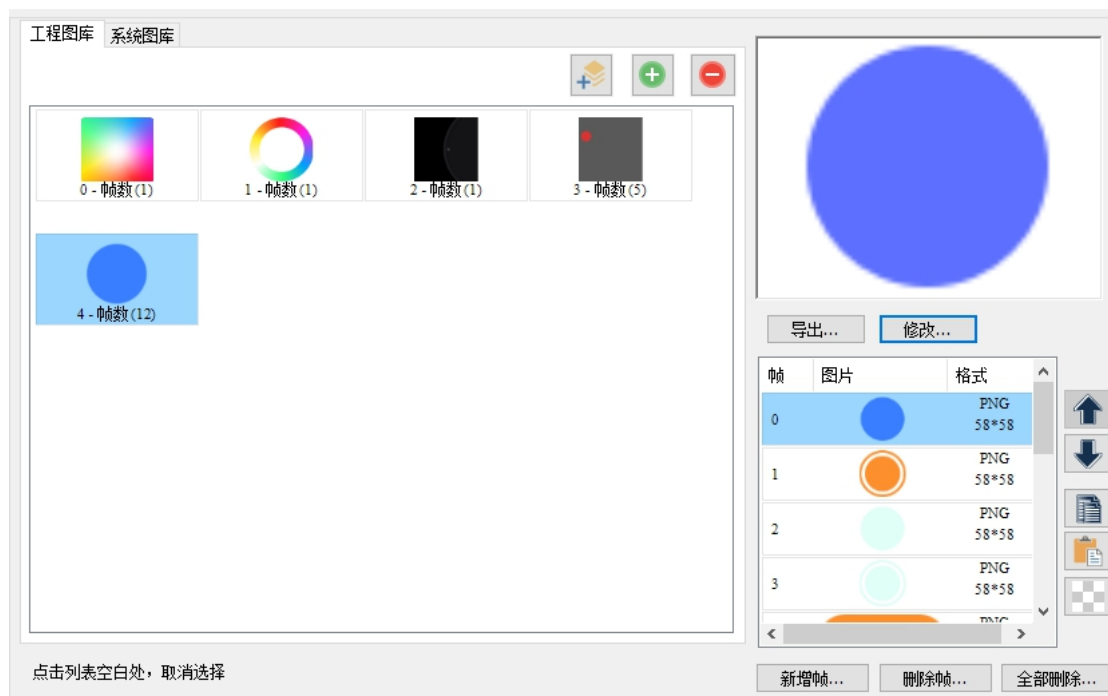


图 10-9 修改图标顺序

10.2.3 删除帧

对已有的图标可以指定删除帧、删除所有帧、删除整个图标等操作。

- 删除指定帧：如图标 4 第 0 帧，点击①，执行删除该帧操作，如图 10-10 所示；

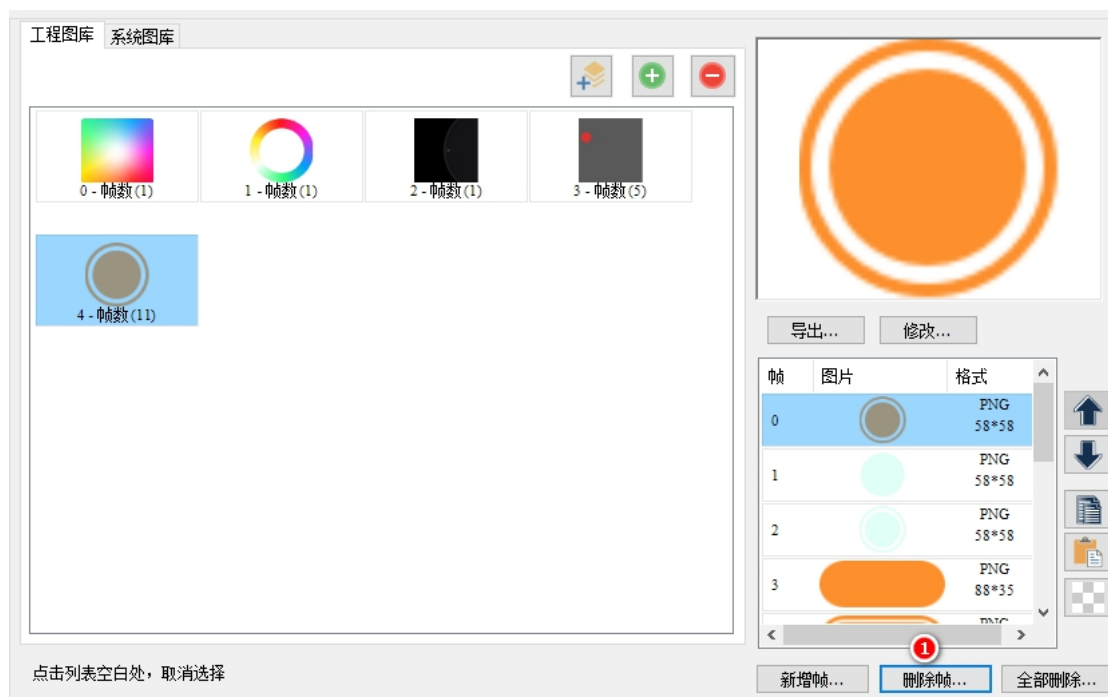


图 10-10 选中图标

- 删除所有帧：选中指定图标 4，删除所有帧，如图 10-12 所示；

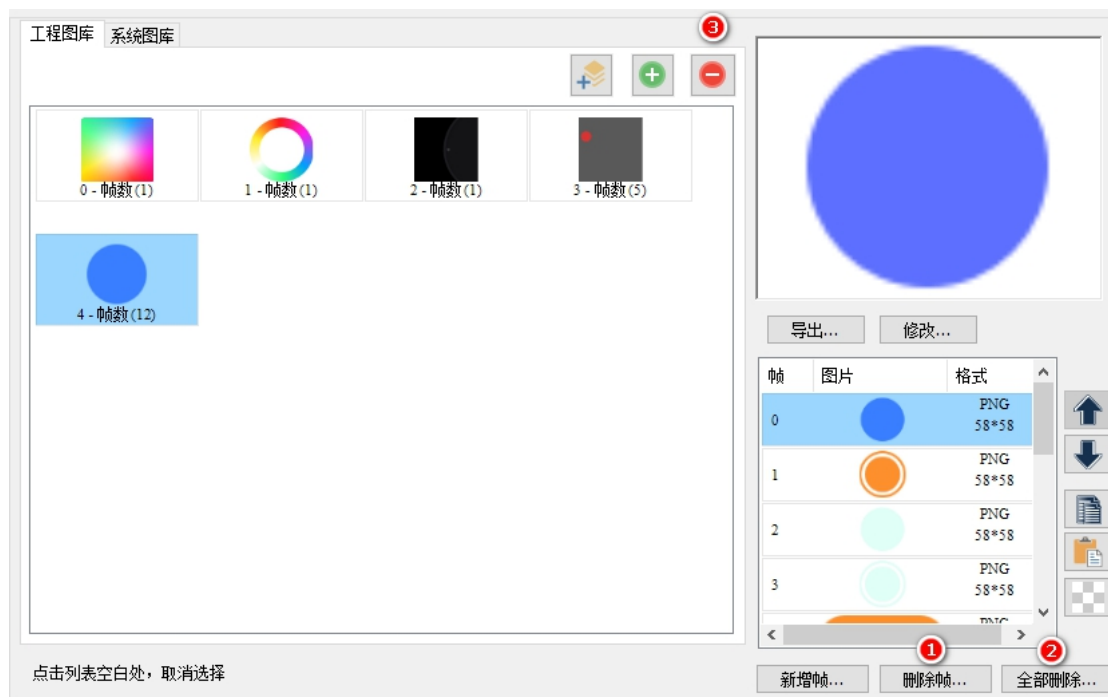


图 10-11 选中图库

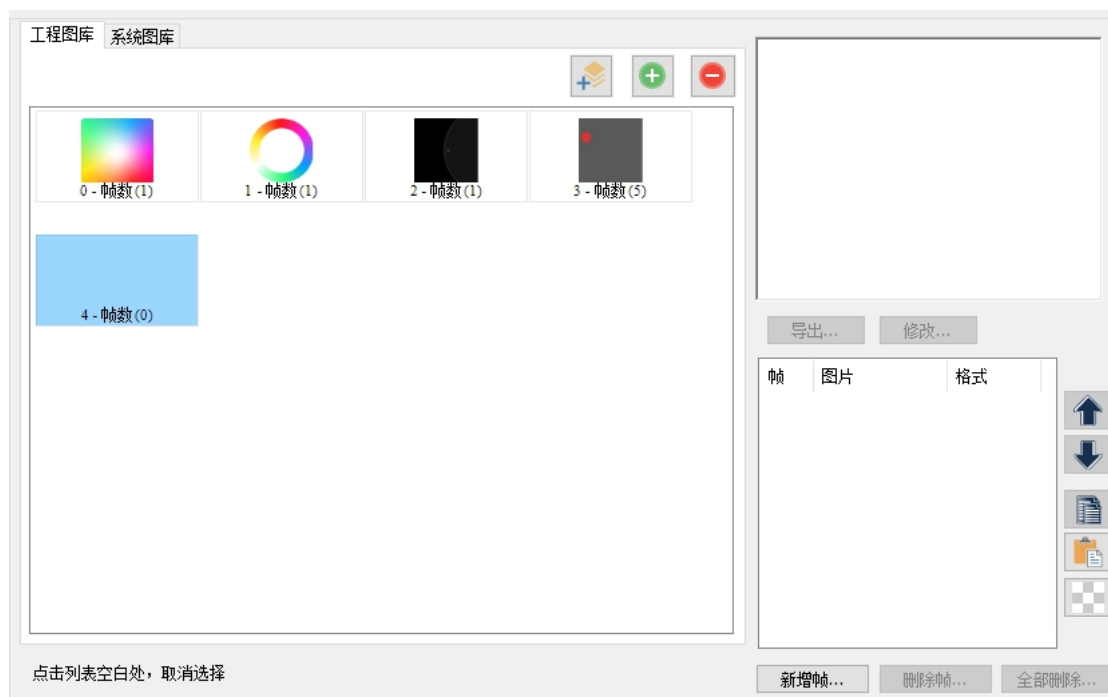


图 10-12 全部删除

- 删除图标库：选中图标 4，按住键盘【Shift】，点击③，删除图标，如图 10-13 和图 10-14 所示；

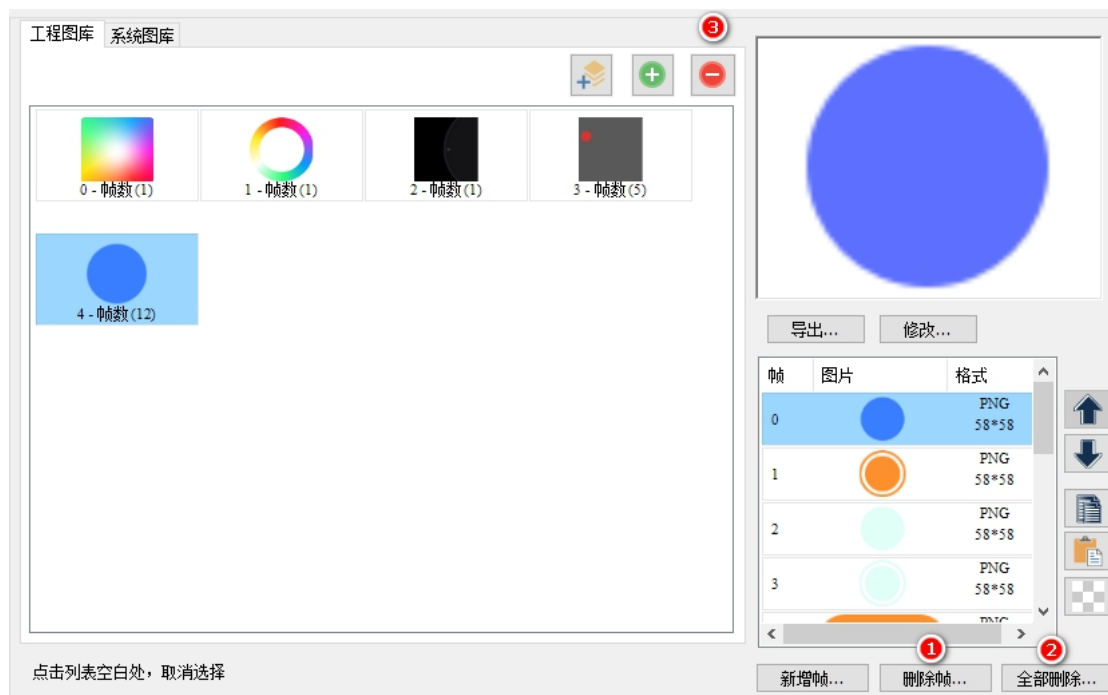


图 10-13 选中图库

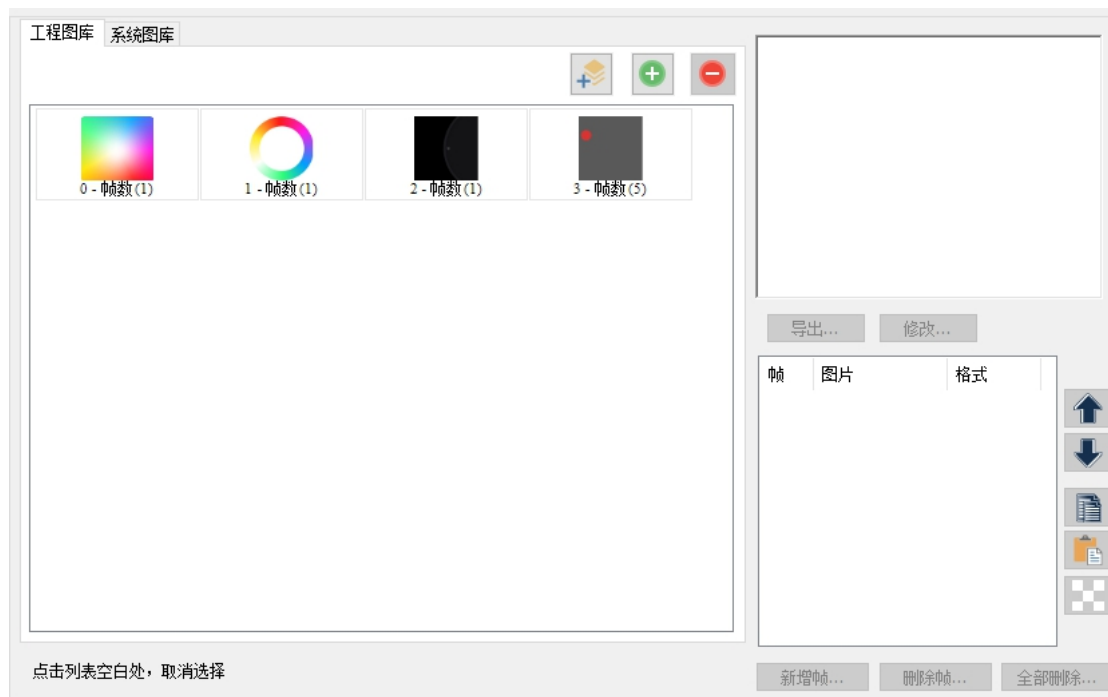


图 10-14 删除图库

11. 字库

VisualHMI 软件上的字库的显示均是矢量字库，字体的实际尺寸可以任意字号而不失真、



抗锯齿感强。点击菜单工程栏→字库 , 打开字库设置，如下图 11-1 所示。默认选中

wqyMicroHeiMono.ttf 字库，字库索引为 0。系统默认提供 5 个字库可选

新添加的控件，默认均关联索引 0 对应的字库

- DroidSans.ttf: 只含字母数字；
- DS-DIGIB.ttf: 数码管，大写字母、数字；
- fangsong_GB2312.ttf: 仿宋，支持中英文；
- songti_GB2312.ttf: 宋体，支持中英文；
- wqyMicroHeiMono.ttf: 文泉，默认字库，支持小语种，多语言使用；

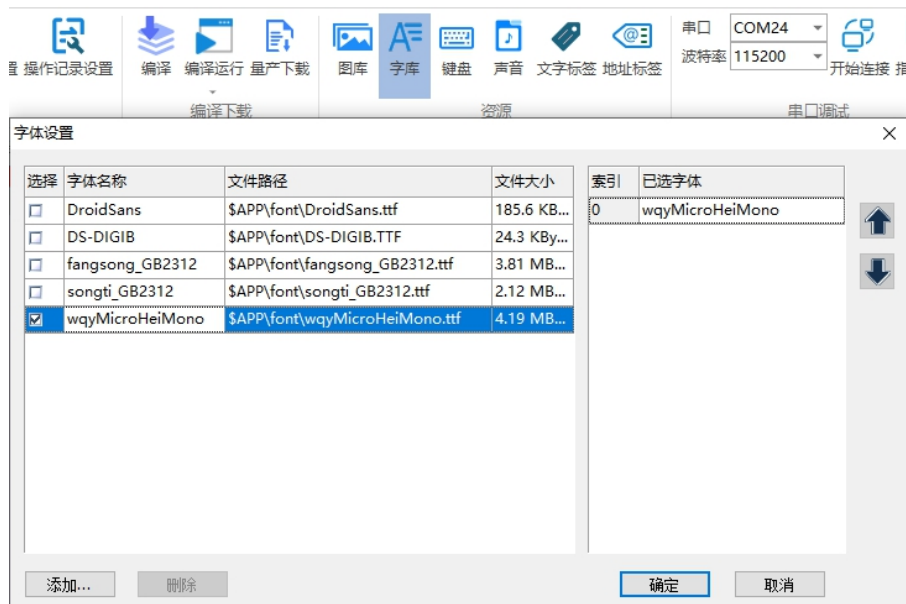


图 11-1 字库

11.1 添加多个字体

VisualHMI 可支持多个字库，如下图 11-2 所示。

1. 如勾选 DroidSans，索引为 1；
2. 在控件中，外观设置→字体即可关联所添加的字库；

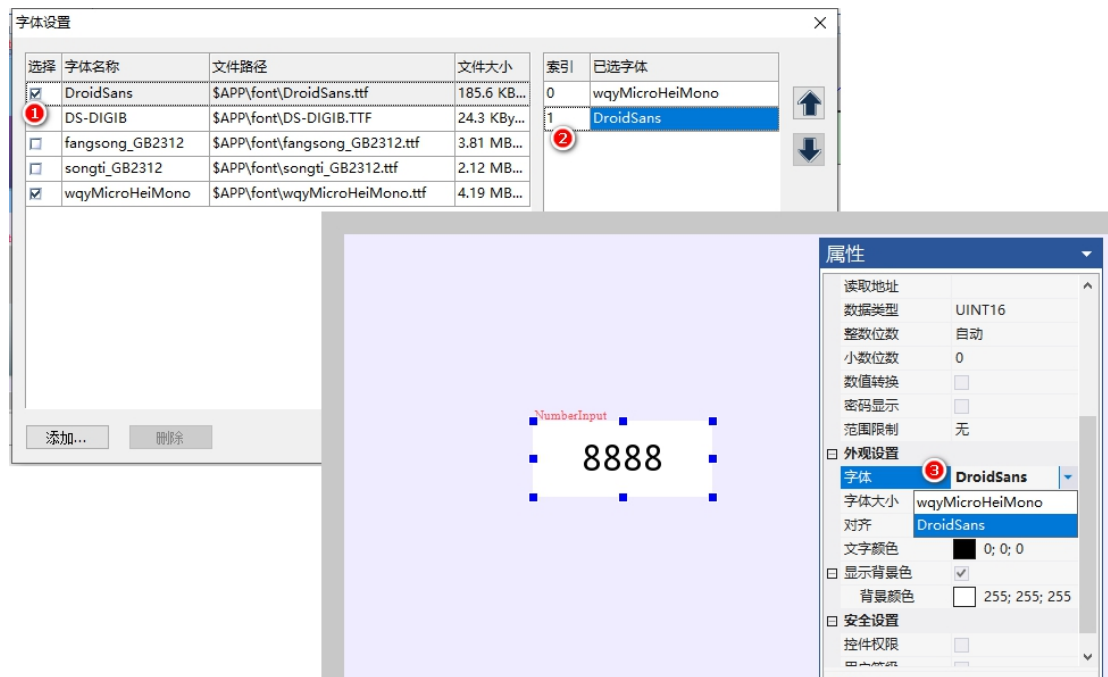


图 11-2 选择字库

11.2 添加用户字体

VisualHMI 可以添加用户的字库，如下图 11-3 所示。

1. 点击添加所需的字库，如 OPPOSansR.ttf，添加成功后，索引为 2（假设工程已有 2 个字体）；
2. 此时工程目录下的 font 文件夹，已将 OPPOSansR.ttf 拷贝到该文件夹下；
3. 在控件中即可关联所添加的字库；

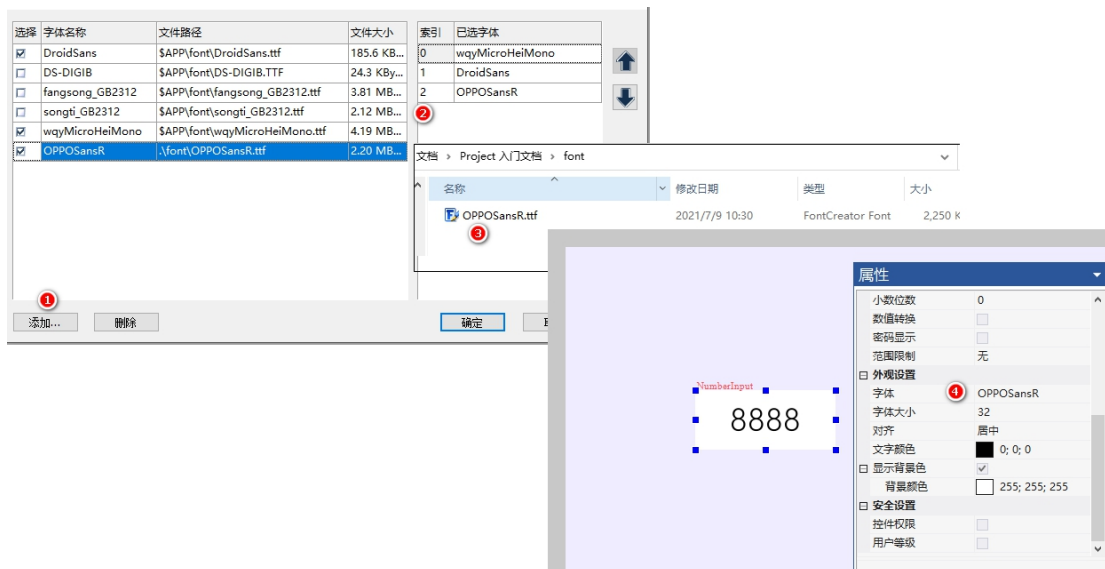


图 11-3 添加字库

11.3 替换字体

字体的替换，可以一键更改，只需要修改对应字体的索引即可。假设工程中使用 wqyMicroHeiMono.ttf（当前索引为 0），需要替换为 OPPOSansR.ttf（当前索引为 2）。只需要将索引互换即可，如下图 11-4 所示。



图 11-4 替换字库

12. 地址标签

VisualHMI 软件上的用户变量地址可支持批量导入、导出，添加意义的标签。点击菜单



工程栏→地址标签 地址标签,其中地址标签包括系统标签（不可修改，LW 地址小于 0x1000）、用户自定义标签，如下图 12-1 所示

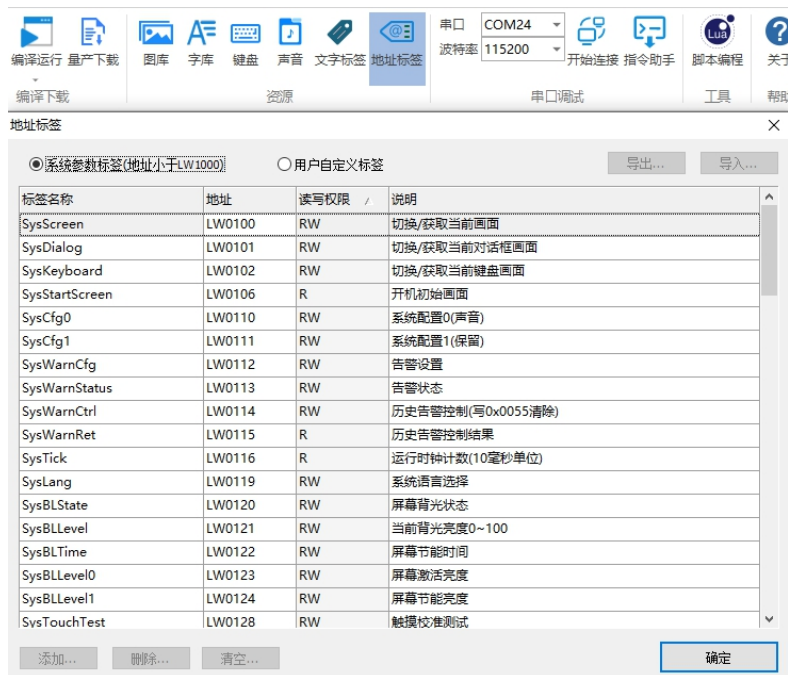


图 12-1 系统地址标签

12.1 系统参数标签

系统参数标签不支持修改、导入、导出，详细地址说明参考[“系统变量”](#)。

12.2 用户自定义标签

变量类型如下图 12-2 所示。

表格 1 用户自定义标签

地址类型	数据类型	说明
HMI 本机	LW 变量地址	内存临时变量，地址定义大于 0x1000
	RW(Flash 存储)	掉电存储：0x0000~0x8000
PLC（设备） Modbus 协议	1x(线圈)	可读可写
	2x(离散输入)	可读
	3x(输入寄存器)	可读
	4x(保持寄存器)	可读可写

12.2.1 添加标签

假设给一个地址为 LW100 的内存变量添加标签，如下图 12-2 所示；

1. 点击添加；
2. 地址位于：HMI 本机；
3. 数据类型：LW（变量地址）；
4. 16 进制地址：1001，内存变量必须 ≥ 1000 (16 进制)；

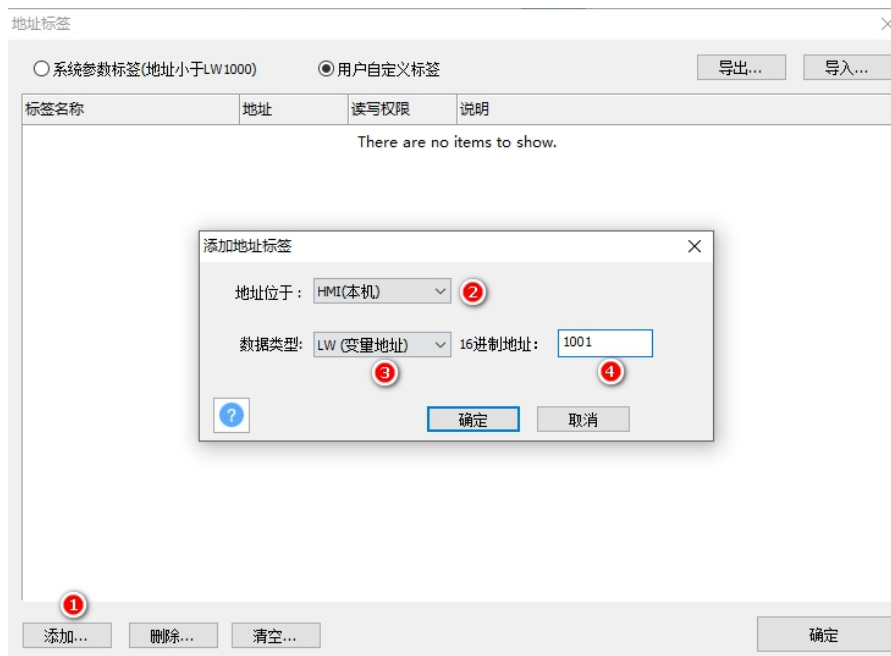


图 12-2 添加变量

若给变量的某一位设置标签则用“.”表示引用变量某个位的数据，给 modbus 保持寄存器 0x1001 地址的 bit0 位设置标签，如图 12-3 所示；

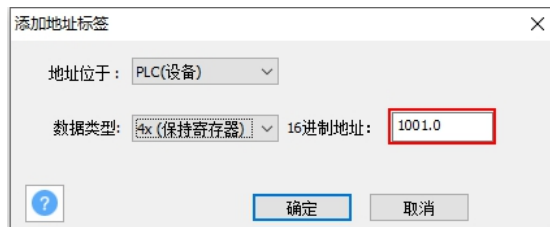


图 12-3 添加位地址标签

若给变量的连续的某几位设置标签，则用“.”和“~”引用，给 modbus 保持寄存器 0x1001 地址的 bit0~bit3 设置标签，如图 12-4 所示；

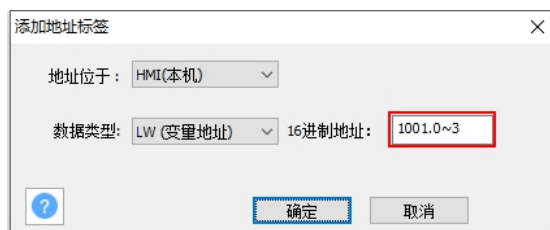


图 12-4 连续地址标签

5. 修改标签名：默认为 Addr，第一个为 Addr0，改为“临时_温度”，如图 12-5 所示；



图 12-5 修改标签名字

6. 控件选中地址标签：

以数值控件为例，绑定“临时_温度”标签，如图 12-6 所示：

- 点击控件读取地址一属性；
- 点击标签卡；
- 选择用户自定义；
- 选择“临时_温度”；
- 点击确定；

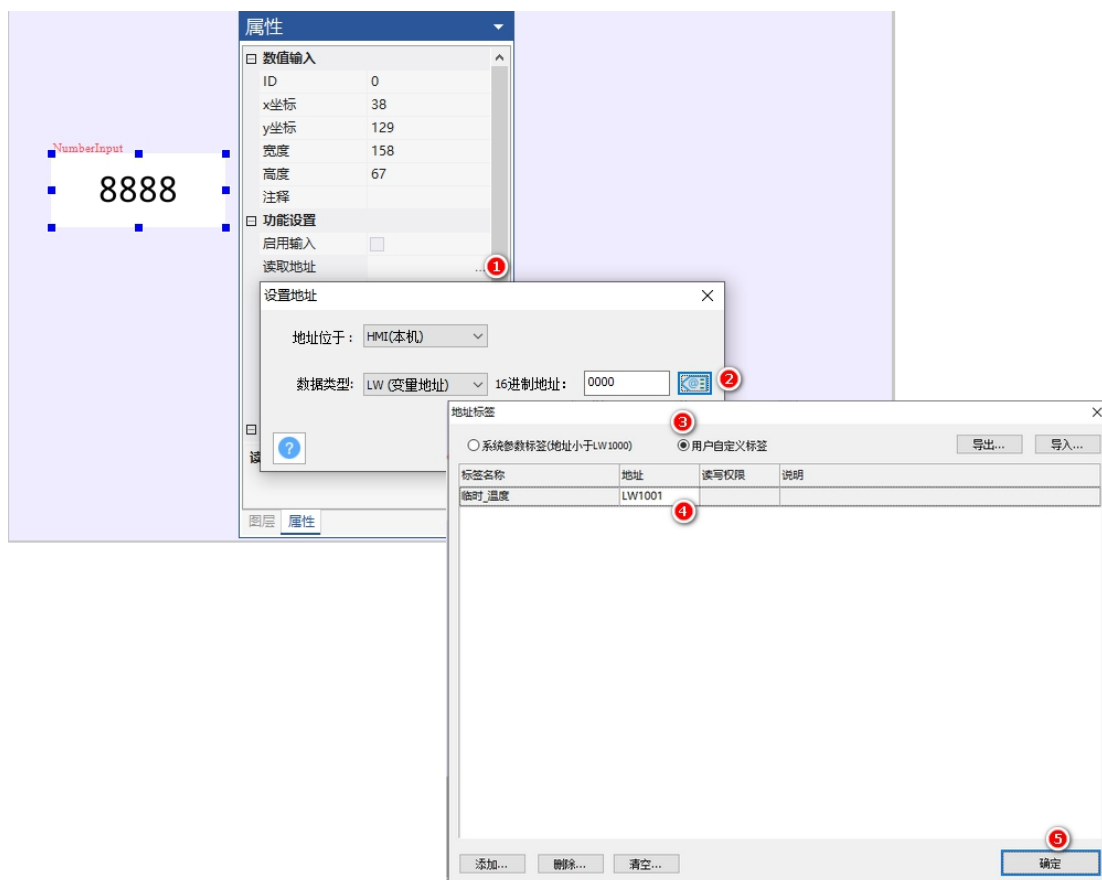


图 12-6 控件关联标签

经上述操作，控件已绑定“临时_温度”，如下图 12-7 所示；

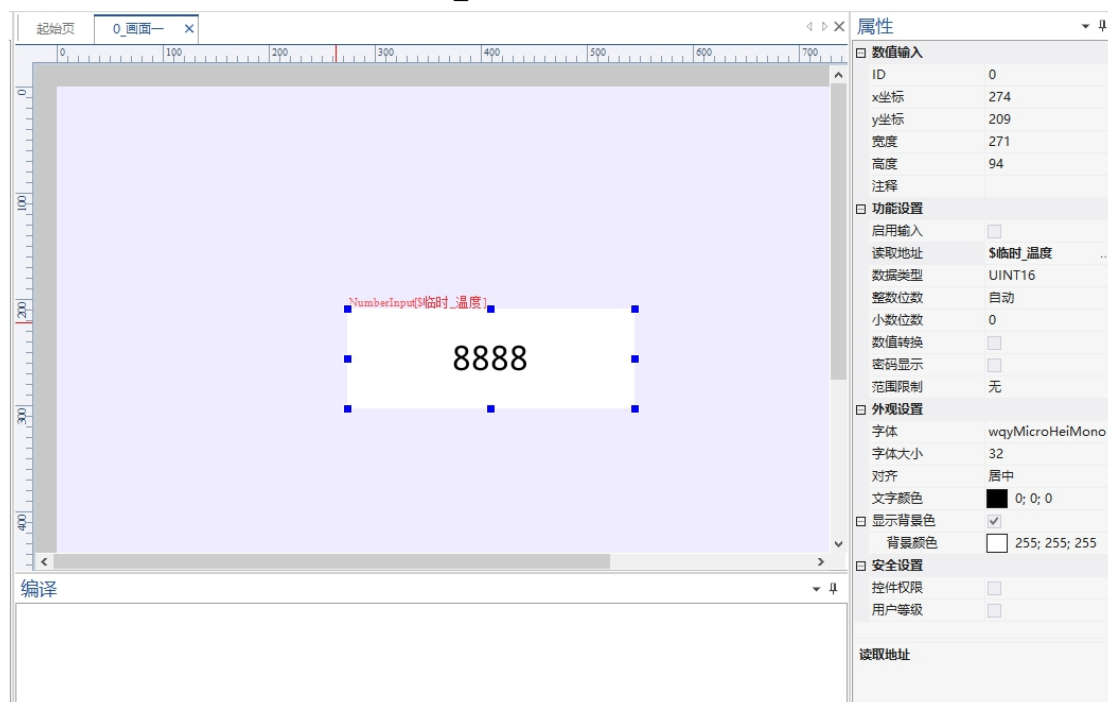


图 12-7 效果显示

12.2.2 删除标签

标签支持删除或清空，如下图 12-8 所示

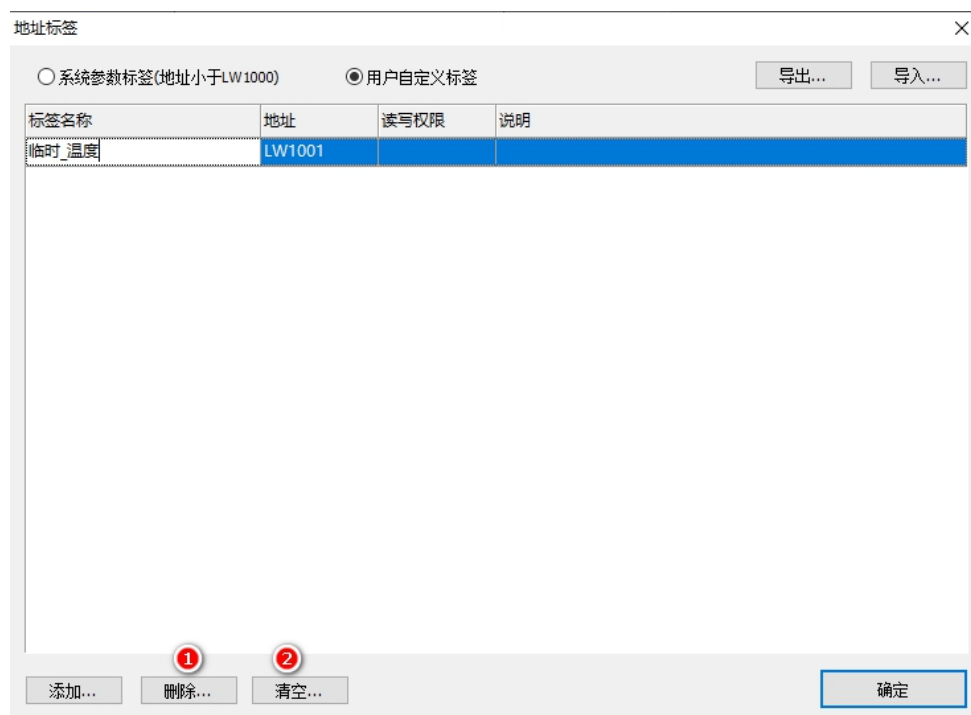


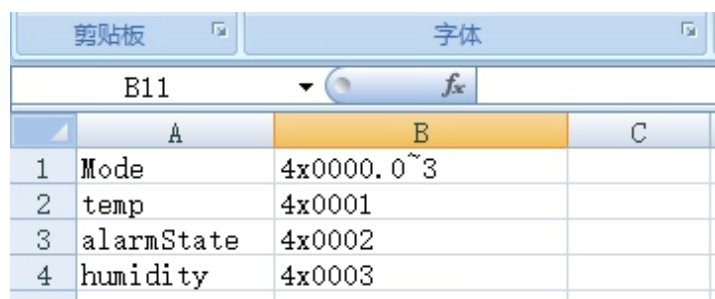
图 12-8 删除标签

12.2.3 标签批量导入

标签可以批量导入导出*.csv、*.xml 格式的文件。

1. csv, xml 编辑;

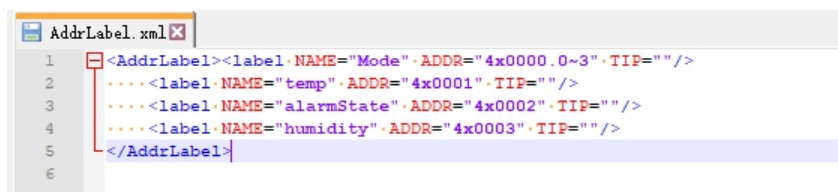
- CSV 格式编辑, 打开 office Excel 软件, 第 1 列写标签名称, 第 2 列写变量地址, 第 3 列输入空格, 如图 12-9 所示;



	A	B	C
1	Mode	4x0000.0~3	
2	temp	4x0001	
3	alarmState	4x0002	
4	humidity	4x0003	

图 12-9 CSV 格式

- xml 格式编辑, 如下图 12-10 所示;



```

1 <AddrLabel><label NAME="Mode" ADDR="4x0000.0~3" TIP="" />
2 ...<label NAME="temp" ADDR="4x0001" TIP="" />
3 ...<label NAME="alarmState" ADDR="4x0002" TIP="" />
4 ...<label NAME="humidity" ADDR="4x0003" TIP="" />
5 </AddrLabel>
6

```

图 12-10 XML 格式

2. csv、xml 存储: 将表格存储为 csv (逗号分隔) 或 xml 格式, 如图 12-11 所示;

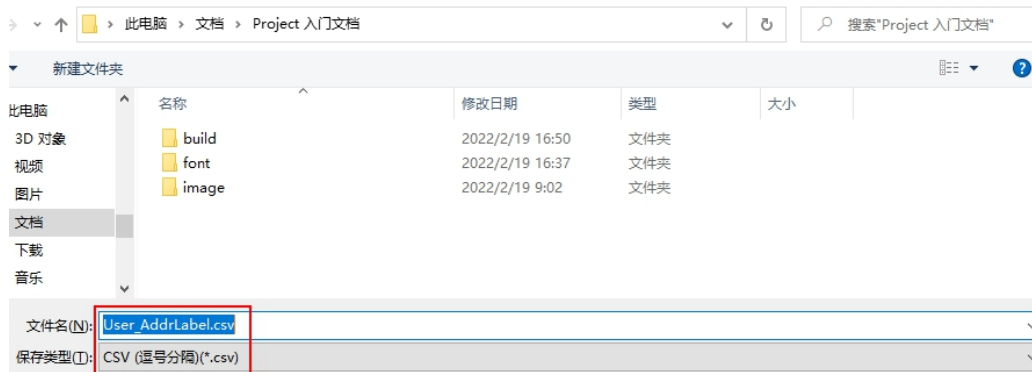


图 12-11 存储

3. 导入 csv,xml 格式, 如图 12-12 所示。

- 点击导入;
- 选择 csv 或 xml 格式的文件;
- 导入成功后, 显示编辑的地址标签;

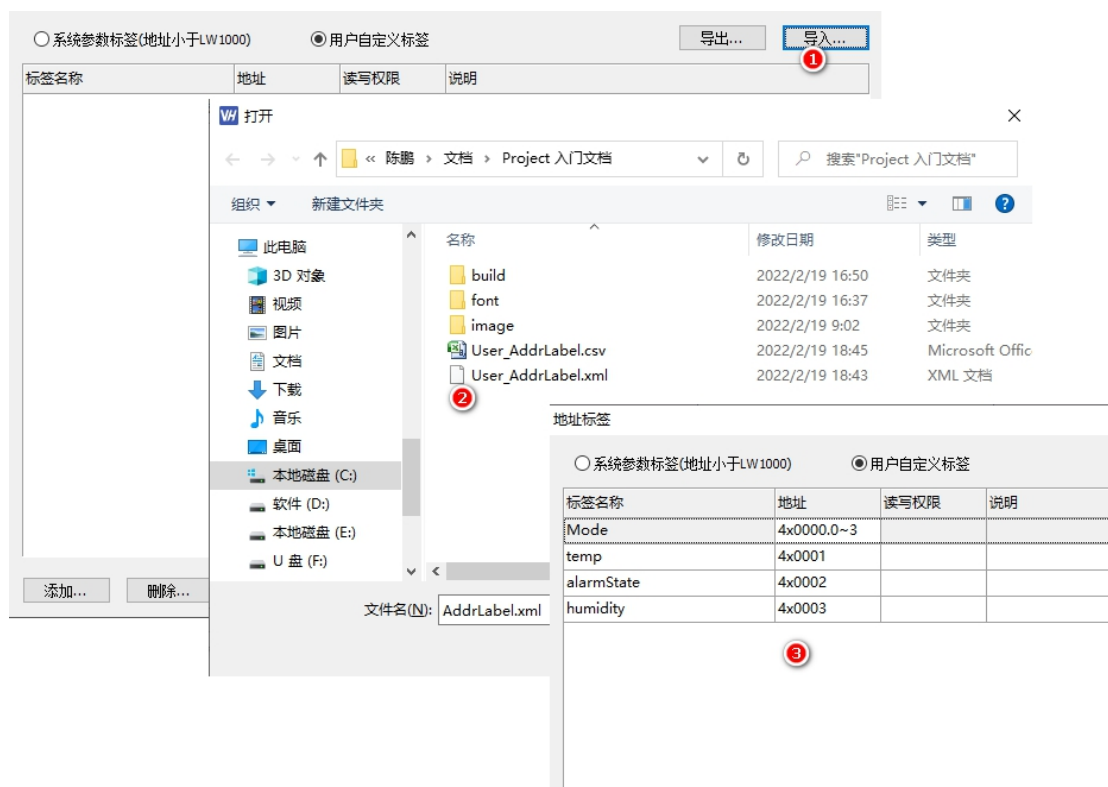


图 12-12 导入标签

12.2.4 标签批量导出

csv 、xml 格式导出，如下图 12-13 所示。

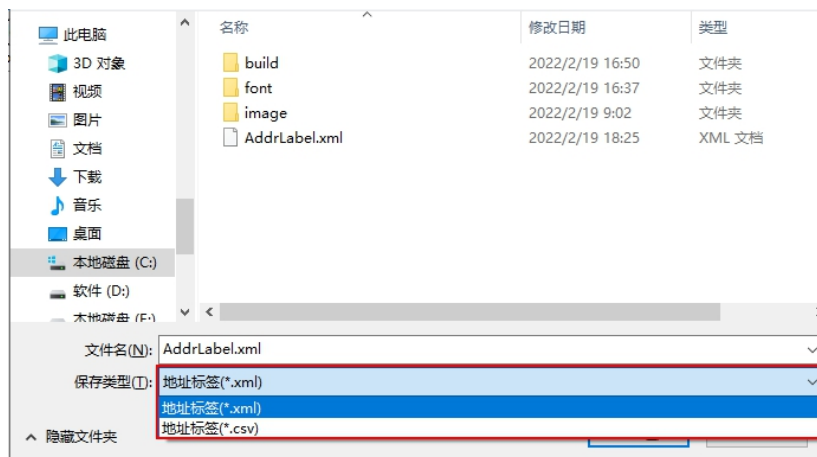


图 12-13 导出标签

13. 系统变量

系统变量：地址范围 LW0000-LW0FFF，系统已定义的变量如下表所示，其余未定义的作为系统保留(不要使用)。

变量格式：U16（WORD），U32（DWORD）

表格 2 系统变量接口一览表

地址	标签	读写	说明
0x0100	SysScreen	RW	切换/获取当前画面
0x0101	SysDialog	RW	切换/获取当前对话框画面
0x0102	SysKeyboard	RW	切换/获取当前键盘画面
0x0106	SysStartScreen	R	开机初始画面
0x0110	SysCfg0	RW	系统配置 0(声音) (bit0~bit1)-触摸蜂鸣器声音，0 静音，1 蜂鸣器声音文件;bit2-启用开机声音;
0x0111	SysCfg1	RW	系统配置 1(保留)
0x0112	SysWarnCfg	RW	告警设置 bit0-激活背光; bit1~bit2-控制蜂鸣器，0 不控制，1 滴滴叫，2 长鸣;
0x0113	SysWarnStatus	RW	告警状态、bit0-触发告警标记;
0x0114	SysWarnCtrl	W	历史告警控制 写 0x0001 导出到 SD 卡/U 盘; 写 0x0055 清除历史告警;
0x0116	SysTick	R	运行时钟计数(10 毫秒单位)
0x0119	SysLang	RW	系统语言选择
0x011A	SysCfgCtrl	RW	系统参数控制 写 0x5501 保存参数; 写 0x5502 加载参数; 写 0x5503 清除参数;
0x011B	SysCfgOption	RW	系统参数选择、配合 SysCfgCtrl 使用: bit0-声音 SysCfg0; bit1-多语言 SysLang; bit2-音频音量大小 SysSndVol; bit3-背光设置; bit4-用户密码 bit5-分期使用参数;
0x0120	SysBLState	RW	屏幕背光状态 bit0-自动背光调节开关; bit1-背光激活状态;
0x0121	SysBLLevel	RW	当前背光亮度 0~100
0x0122	SysBLTime	RW	屏幕节能时间 屏幕空闲时间(无触摸)到达后，自动进入节能状态。
0x0123	SysBLLevel0	RW	屏幕激活亮度

0x0124	SysBLLevel1	RW	屏幕节能亮度
0x0128	SysTouchTest	RW	触摸校准测试、写 0x5500 退出体验测试; 写 0x5501 进入体验测试; 写 0x5502 进入触摸校准;
0x0129	SysTouchState	R	触摸按压状态(0 松开;1 按下;)
0x012a	SysTouchX	R	触摸 X 坐标
0x012b	SysTouchY	R	触摸 Y 坐标
0x012c	SysTouchTick	R	触摸最后按下时间(UINT32):32 位长整形, 10 毫秒单位
0x0130	SysCpCmd	RW	拷贝-开始执行、拷贝完毕后自动清零; 写 0x5501 开始拷贝(自动通知); 写 0x5502 开始拷贝(不通知);
0x0131	SysCpSrcType	RW	拷贝-原始数据类型 1-LW(变量地址); 2-RW(FLASH 地址); 其它值参考协议文档;
0x0132	SysCpDstType	RW	拷贝-目的数据类型 1-LW(变量地址); 2-RW(FLASH 地址); 3-其它值参考协议文档;
0x0133	SysCpSrcAddr	RW	拷贝-原始数据地址
0x0134	SysCpDstAddr	RW	拷贝-目的数据地址
0x0135	SysCpCount	RW	拷贝-变量数量
0x0140	SysSndVol	RW	喇叭音量(0~100)
0x0141	SysSndPlay	RW	播放音频 ID
0x0142	SysSndState	RW	音频播放控制 0 空闲, 1 开始播放, 2 暂停播放, 3 停止播放;0x80 表示播放中
0x0143	SysSndPlayTime	R	音频已播放时间(秒)
0x0144	SysSndTotalTime	R	音频总时间(秒)
0x014F	SysBeep	W	蜂鸣器鸣叫、蜂鸣器鸣叫时间(10 毫秒单位), 自动清 0
0x0150	SysReboot	RW	系统重启(写 0x5555)
0x0152	SysVHMIVer	R	PC 软件版本
0x0153	SysVHMIBldTime	R	工程编译时间
0x0155	SysFWVer	R	固件版本
0x0156	SysDevType	R	屏幕型号
0x0157	SysScreenDir	R	屏幕旋转角度 0: 0 度, 1: 90 度, 2: 180 度, 3: 270 度
0x0158	SysScreenXMax	R	屏幕 X 分辨率
0x0159	SysScreenYMax	R	屏幕 Y 分辨率
0x0160	SysSetYear	RW	年(公历)
0x0161	SysSetMonth	RW	月(公历)
0x0162	SysSetDay	RW	日(公历)
0x0163	SysSetWeek	RW	星期

0x0164	SysSetHour	RW	时
0x0165	SysSetMinute	RW	分
0x0166	SysSetSecond	RW	秒
0x0167	SysSetTime	RW	设置日期时间 写 0x5501 设置日期时间; 写 0x5502 设置日期; 写 0x5503 设置时间;
0x0170	SysGetYear	R	年(公历)
0x0171	SysGetMonth	R	月(公历)
0x0172	SysGetDay	R	日(公历)
0x0173	SysGetWeek	R	星期
0x0174	SysGetHour	R	时
0x0175	SysGetMinute	R	分
0x0176	SysGetSecond	R	秒
0x0180	SysLoginLevel	RW	当前登录等级 0 游客~8 最高权限, 写 0 注销登录
0x0181	SysLoginStatus	RW	登录状态 0 空闲; 写 1 开始登录; 读 2 登录成功; 读 3 登录失败;
0x0182	SysLoginPwd	RW	用户输入的登录密码、纯数字密码, 长整型; 设置密码之后, SysLoginStatus 写 1 开始登录。
0x0184	SysLoginLevelReq	RW	当前操作需要的用户等级、当前操作需要输入的密码等级
0x0188	SysLogCtrl	RW	操作记录控制、写 0x0055 清除操作记录
0x0190	SysExportStatus	RW	告警/记录导出状态、 0 空闲, 1 正在导出, 2 导出成功, 3 导出失败;
0x0191	SysExportProg	R	告警/记录导出进度(0~100)
0x01A0	SysSite	R	当前站号
0x01A1	SysSetSite	RW	修改站号(写 0x5500 站号)
0x01A2	SysSiteTimeout	RW	从站通信超时(默认 10 秒) 连续一段时间未收到任何指令, 判断从站为离线(秒单位)。
0x01A3	SysSiteOnline	RW	从站在线状态、每一个位表示一个从站的状态。
0x01A4	SysSiteMask	RW	从站屏蔽掩码(主机模式) 屏蔽指定从机的读写功能(对应 bit 为 1)。
0x01A5	SysSiteOfflineErr	RW	从站离线错误数(默认 3 次) 从站连续多少次无应答, 判断为离线状态。
0x01A8	SysSiteErrCode	RW	最后的通信错误代码
0x01A9	SysSiteErrCount	RW	通信错误次数(UINT32)
0x01AB	SysSiteComCount	RW	通信累计次数(UINT32)
0x01B0	SysCom1Set	RW	COM1 设置、先设置波特率等参数, 然后写入 0x5501 生效
0x01B1	SysCom1Baudrate	RW	COM1 波特率(UINT32)
0x01B3	SysCom1DataBit	RW	COM1 数据位
0x01B4	SysCom1ParityBit	RW	COM1 校验位
0x01B5	SysCom1StopBit	RW	COM1 停止位

0x01C0	SysKBMax	R	键盘-数值最大值(索引)
0x01C1	SysKBMin	R	键盘-数值最小值(索引)
0x01C2	SysKBTip	RW	键盘-录入提示信息
0x01C3	SysKBSelect	RW	键盘字符选择、0 英文, 1 数字符号, 2 中文
0x01C4	SysKBStatus	RW	键盘状态, bit0-SHIFT
0x01C8	SysKBPinYin	R	键盘-拼音输入(索引)
0x01C9	SysKBHanzi	R	键盘-汉字候选(索引)
0x01CC	SysKBText	R	键盘-录入内容(索引)
0x0200	SysStageCount	RW	分期数(最多 10 组)
0x0201	SysStageLockStatus	RW	分期锁定状态 0 未锁定, 1 到期锁定, 2 时间异常锁定
0x0202	SysStageLockScreen	RW	分期锁定画面、锁定时自动切换到此画面
0x0203	SysStageUnlockLevel	RW	当前已解锁分期编号
0x0204	SysStageTimeLevel	RW	当前时间所处分期编号
0x0205	SysStageEnterPwd	RW	用户输入的分期密码、输入正确密码, 自动解锁当前分期
0x0210	SysStageTime	RW	分期截止时间数组(UINT32)、最多 10 组, 40 字节


14. 多语言与文字标签

在 VisualHMI 软件中，文字标签有利于多语言的应用，通过建立一个文字标包含所有用到的语言。用户在不同组态控件中可以引用同一个文字标签，省去了编写重复的文字的时间，尤其在使用多国语言文字的时，更便捷的管理与修改所有用到的文字。

14.1 多语言

VisualHMI 最大支持切换 30 种多语言，可满足用户多语言需求。

14.1.1 打开多语言

1. 点击 VisualHMI 软件菜单栏中【工程设置】按钮 ，在右侧工程属性中找到“语言数”，输入所需要切换的多语言种数，如图 14-1 所示

工程属性	
工程名称	Project80480M070
HMI机型	DC80480M070 DC80
Flash大小	128Mbit
启动画面	0_开机动画
屏幕旋转	0°
语言数	2
触摸伴音	蜂鸣器
开机音乐	启用
背光亮度	100
自动待机	<input type="checkbox"/>

图 14-1 多语言设置

语言数设定为 2 后，文字标签窗口中会增加第二种语言的输入列表，如图 14-2 所示。

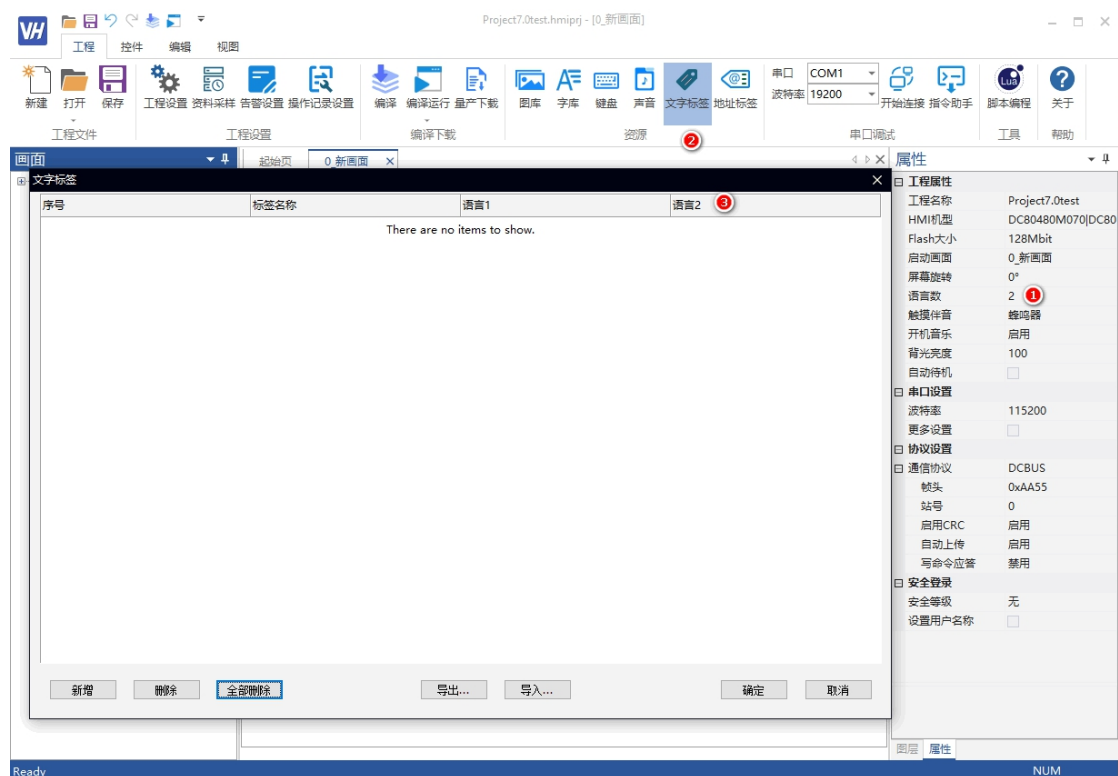



图 14-2 文字标签

14.2 文字标签

点击 Visual HMI 软件菜单栏中【文字标签】按钮  打开文字标签管理，在该弹窗中可以对文字标签进行新增、修改、删除、导出以及导入操作，如图 14-3 所示。

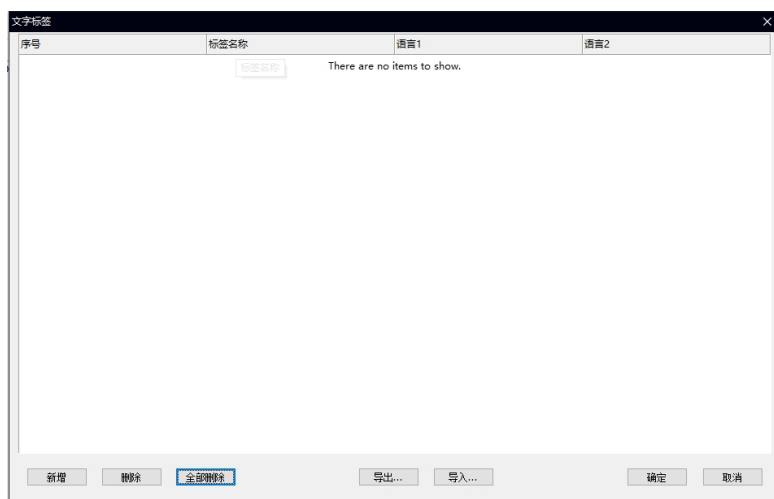


图 14-3 文字标签管理

14.2.1 文字标签使用

本文以多状态按钮为例说明如何使用文字标签：

1. 点击【新增】按钮新建文字标签，并且分别在【语言 1】栏目下方输入“中文”和【语言 2】栏目下方输入“English”后点击确认如下图 14-4 所示

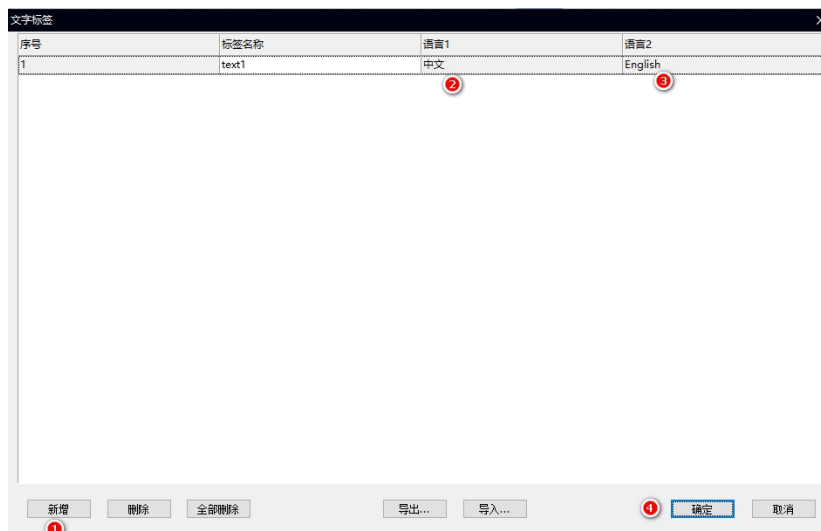


图 14-4 新增标签

2. 点击画面中多状态按钮，绑定“中文”标签，如图 14-5 所示
 - 点击控件，选择属性栏中【使用文字】该参数，设置为“是”；
 - 点击使用标签；
 - 点击标签卡按钮；
 - 选择“中文”；
 - 点击确定；

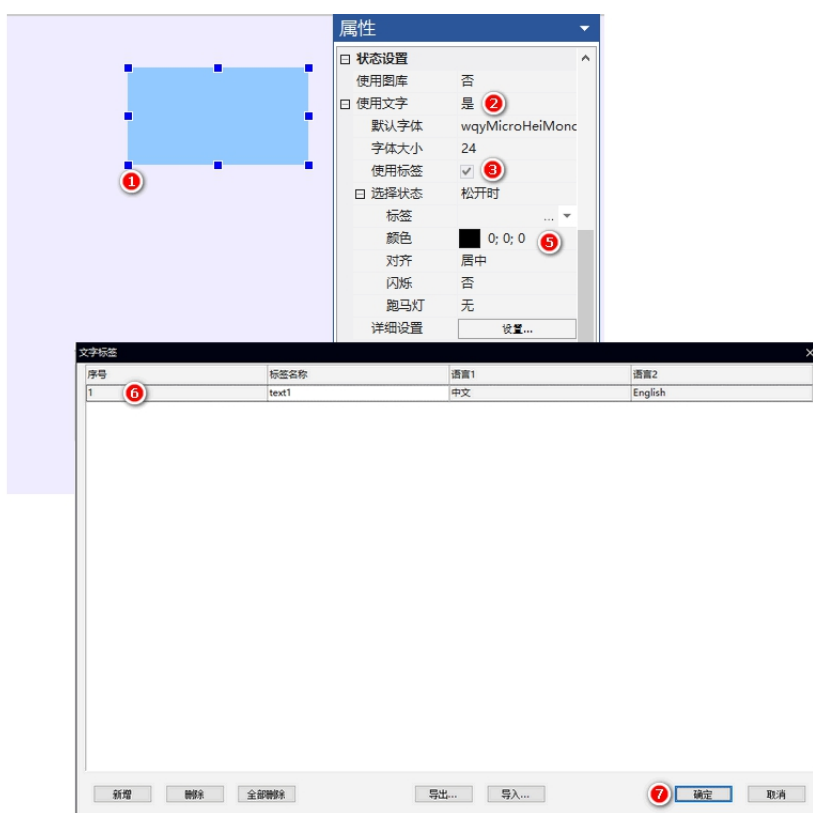


图 14-5 关联标签

经上述操作，控件已绑定“中文”文字标签，如下图 14-6 所示。

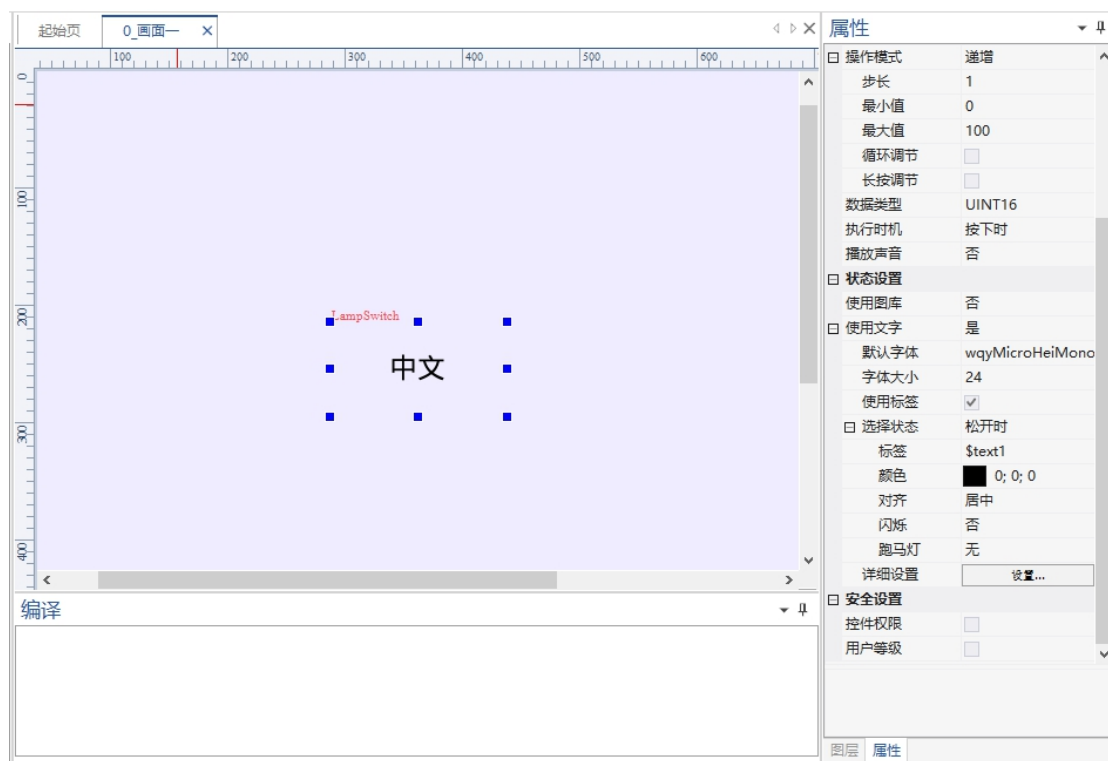


图 14-6 关联标签

14.2.2 文字标签导入

标签可以批量导入导出*.csv、*.xml 格式的文件。

1. .csv 格式文件，.xml 格式文件内容编辑；

- .csv 格式文件，打开 office Excel 软件，第 1 列写标签名称，第 2 列写语言 1 的内容，第 3 列语言 2 的内容，编辑后如图 14-7 所示；

注意：需要根据当前工程语言数依次填入多语言内容，如：当前语言数为2，导入时只识别前3列（1列标签名称+2列语言内容）

	A1			
	A	B	C	D
1	text1	中文	English	
2				
3				

图 14-7 csv 格式

- .xml 格式的文件内容，如图 14-8 所示；

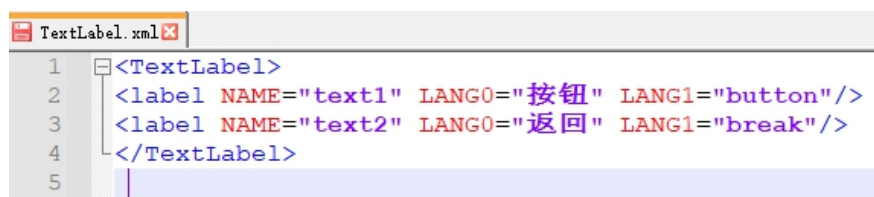


图 14-8 xml 格式

2. csv、xml 存储：将表格存储为 csv（逗号分隔）或 xml 格式，如图 14-9 所示：

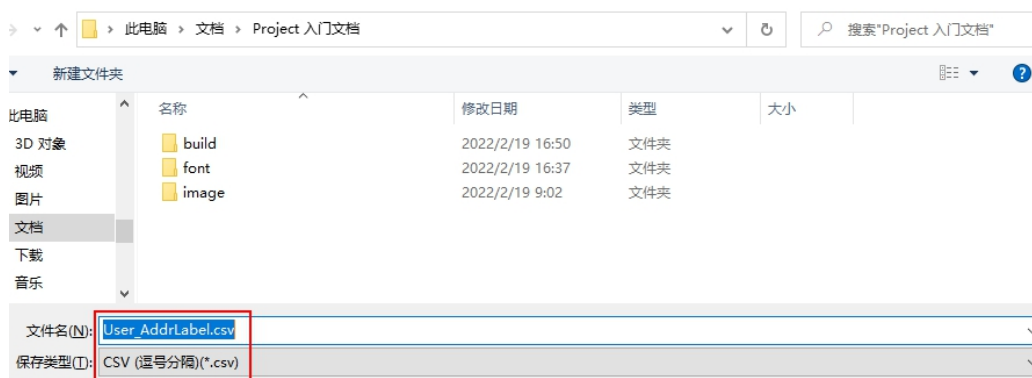


图 14-9 存储格式

3. 导入 csv,xml 格式，如图 14-10 所示。

- 点击导入；
- 选择 csv 或 xml 格式的文件；
- 导入成功后，显示编辑的文字标签

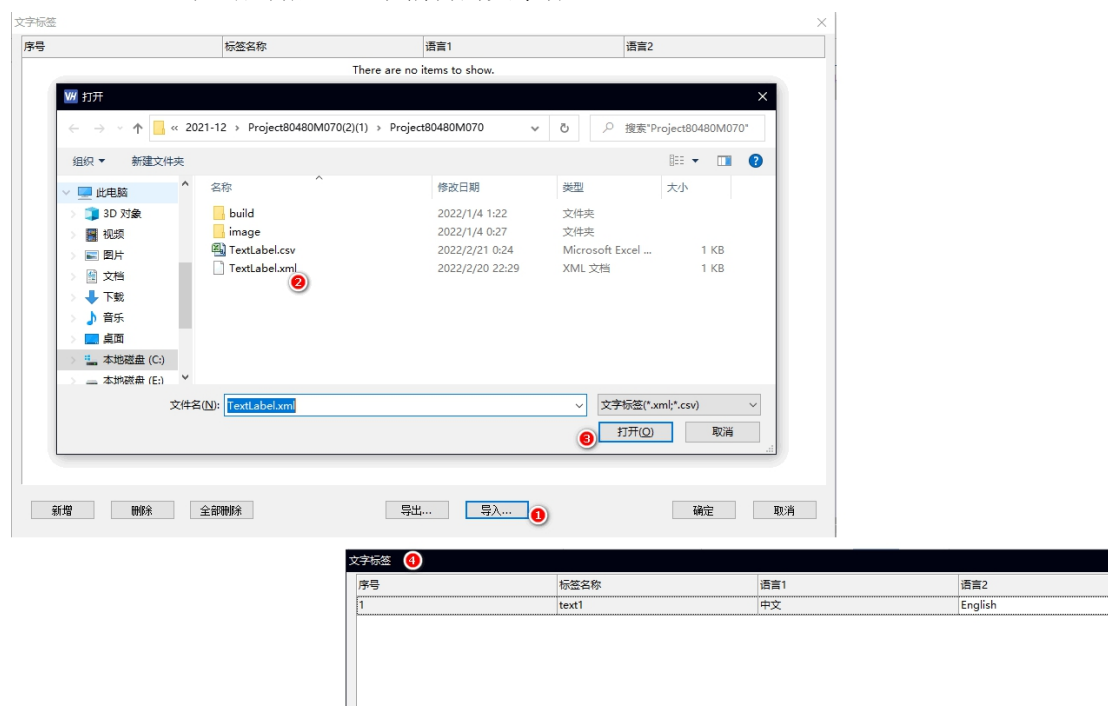


图 14-10 导入标签

15. 键盘

新建工程会默认在画面 ID100 和 ID101 自动生成键盘画面，ID100 为数字键盘，ID101 为全键盘。

15.1 键盘设置


点击软件 VisualHMI 菜单栏中【键盘】按钮 ，键盘设置的弹窗中可以修改、恢复系统绑定的键盘画面，如图 15-1 所示。



图 15-1 键盘设置

15.2 启用键盘输入

在画面中添加一个数值控件，在右侧属性栏中选中“启动输入”选项，如图 15-2 所示。

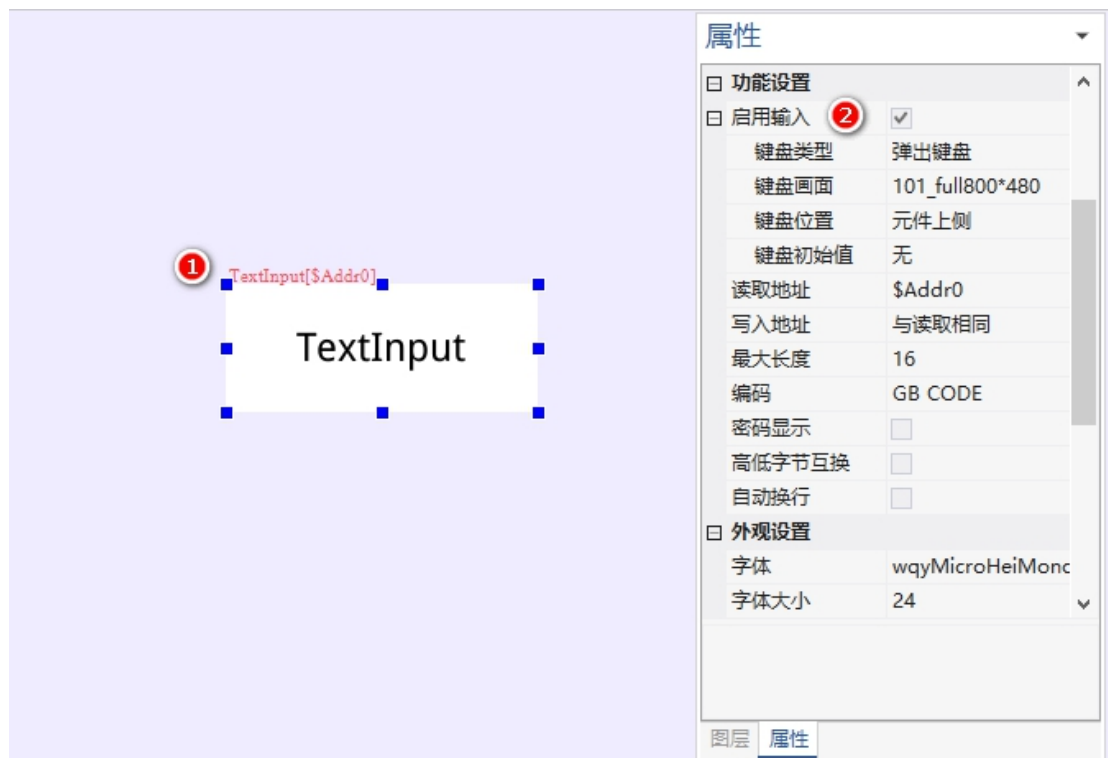


图 15-2 启动输入

15.3 自定义系统键盘

用户可以在系统盘画面上进行自定义修改，也可以生成一个新的键盘画面，在键盘设置中替换。

15.3.1 替换键盘 UI 风格

点击键盘画面的底图，在右侧属性栏中找到“图库”位置，点击修改图库文件选中想要替换的风格，如图 15-3 所示；

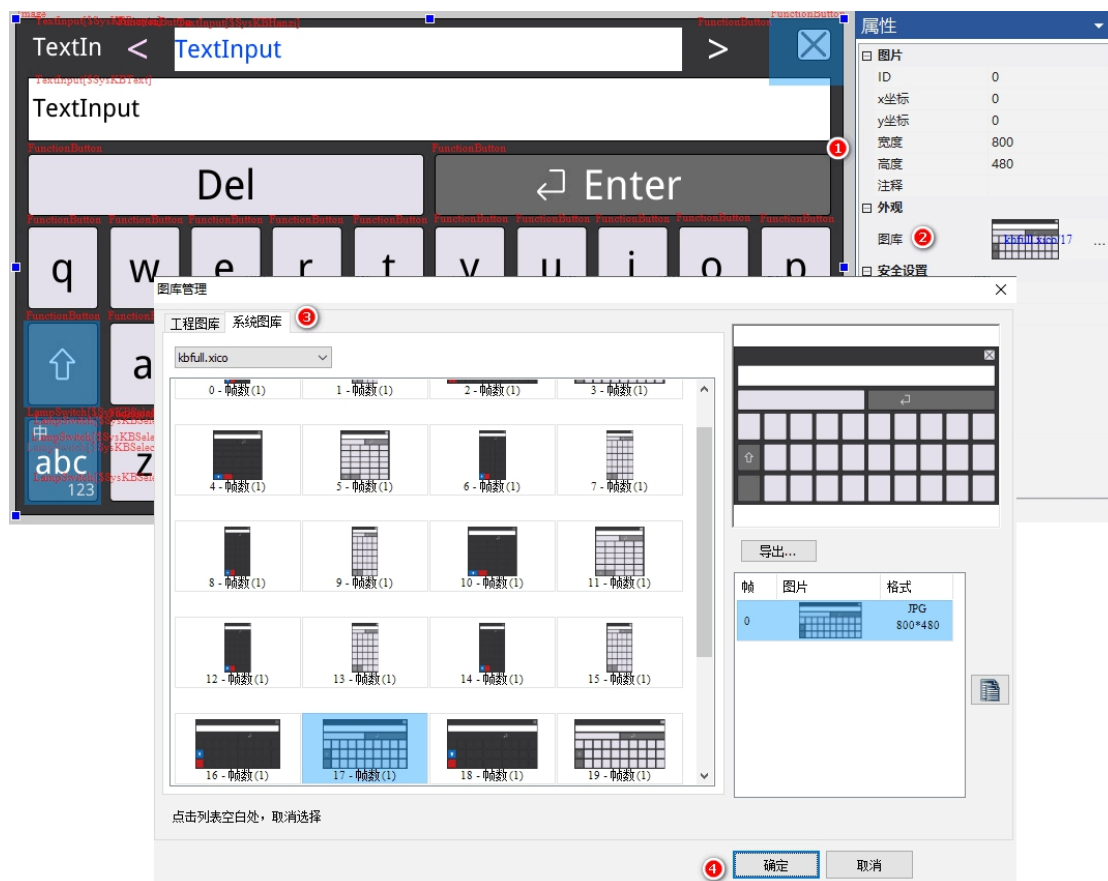


图 15-3 选择键盘背景

15.3.2 修改键值

以修改键值“y”为大写“Y”为例：

1. 点击需要修改键值的控件，在右侧属性中找到“字符”输入栏，将字符“y6”改为“Y6”，如图 15-4 所示：

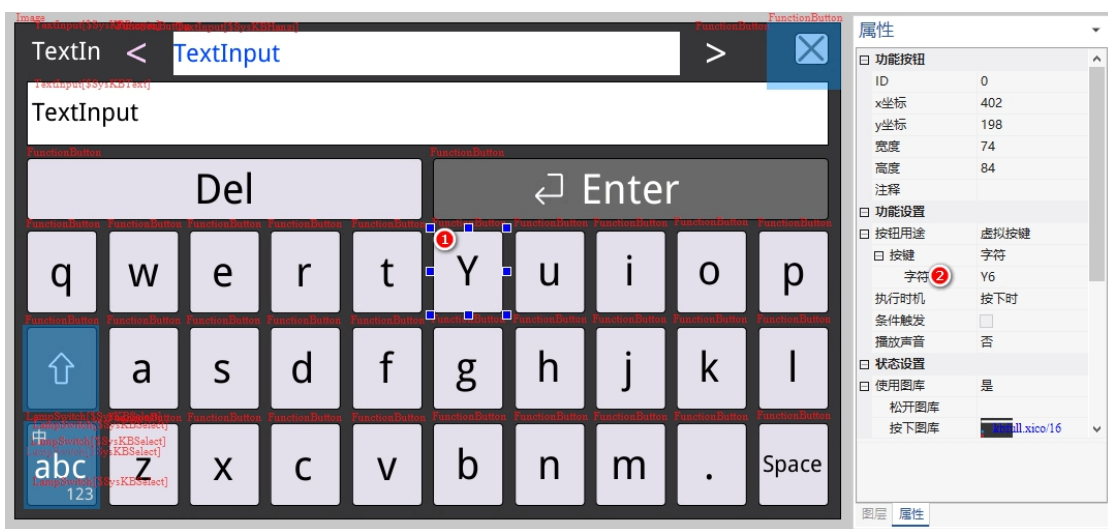



图 15-4 修改键值

2. 点击菜单栏中【编译运行】按钮运行工程测试，效果如图 15-5 所示：

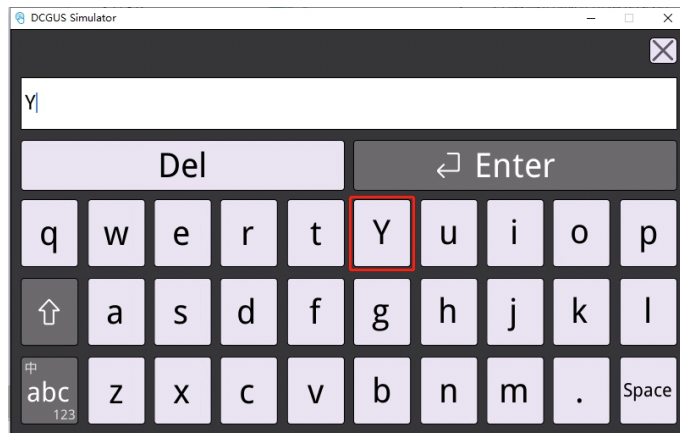


图 15-5 运行测试

15.3.3 键盘相关系统参数地址

下表仅列举部分自定义键盘可能使用到的系统参数，如需使用将控件绑定该地址即可获得对应的参数。

表格 3 部分键盘相关系统参数

键盘输入相关系统参数地址			
0x01C2	SysKBTip	RW	键盘-录入提示信息
0x01C3	SysKBSelect	RW	键盘模式选择-0 英文，1 数字符号，2 中文
0x01C8	SysKBPinYin	R	键盘-拼音输入(索引)
0x01C9	SysKBHanZi	R	键盘-汉字候选(索引)
0x01CC	SysKBText	R	键盘-录入内容(索引)

16. 声音

音频文件仅支持 MP3 和 WAV 格式，码率要求如下表格 4 所示

表格 4 音频文件格式

音频文件格式	
格式	码率
MP3	≤128Kbps
WAV	≤1411Kbps

16.1 添加音频文件

点击菜单栏中【声音】按钮，将需要的音频文件添加进工程，如图 16-1 所示

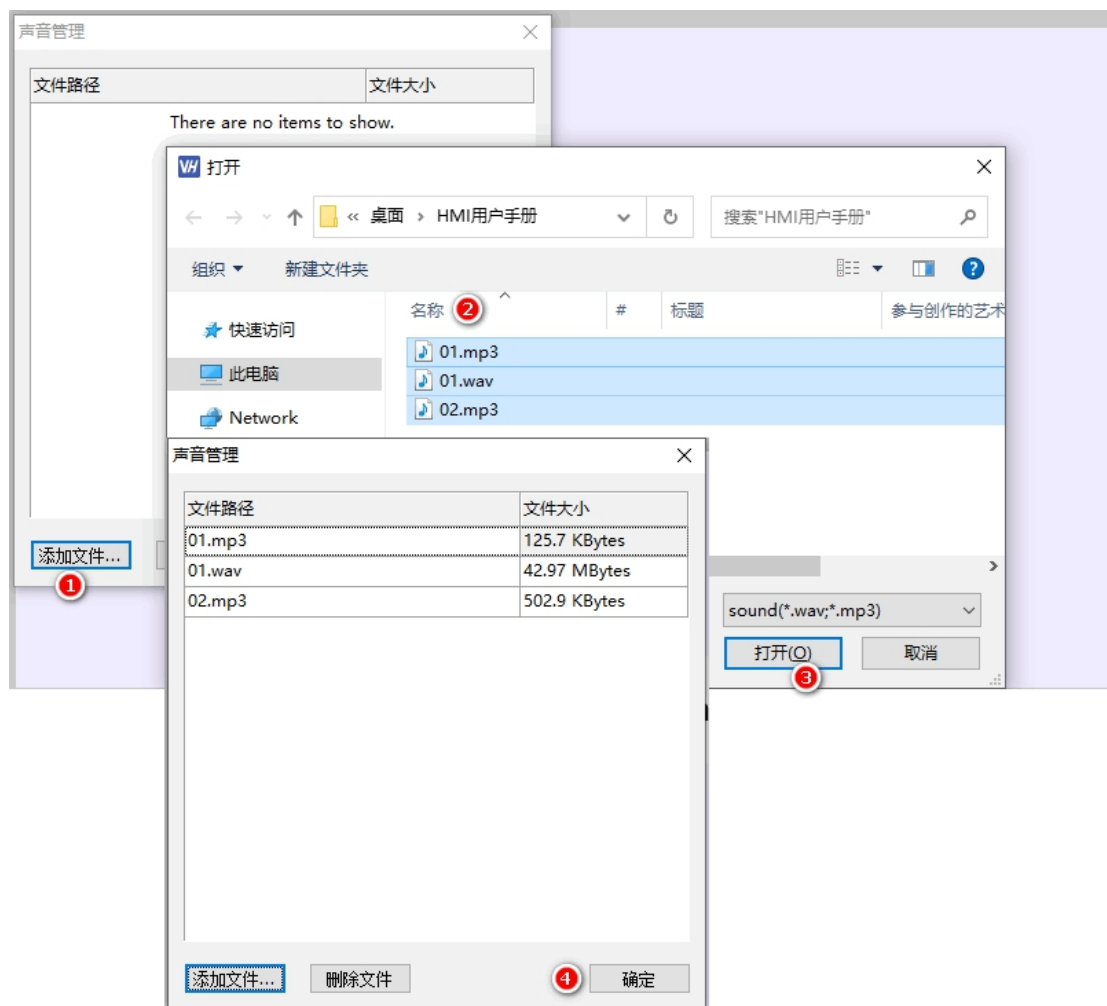


图 16-1 添加音频

16.2 使用音频

点击画面中控件，在右侧属性栏中选择需要触发的音频文件，如图 16-2 所示

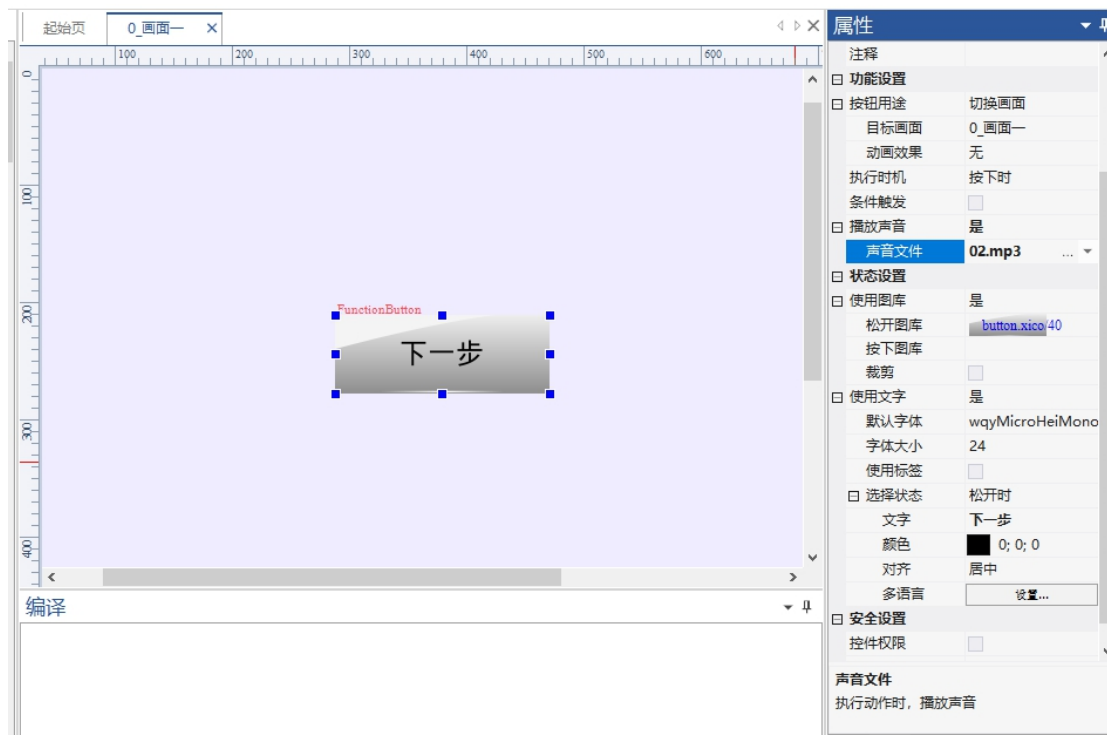


图 16-2 关联音频

17. 协议说明

VisualHMI 集成 DCBUS、XGUS、Modbus（主/从机）和 FX2N 等协议。用户开发时候，同时只能使用一种协议。用户根据产品，选择对应协议开发。屏幕端用户不需要关心如何接收或发送指令/报文，只需要将变量地址关联到对应的组态控件即可。屏幕底层会将接收的报文解析，将数据更新到控件中；操作控件时，会将相应的变量，下发到设备中。

17.1 DCBUS

DCBUS 协议：是由大彩定义的一种组态协议。在工程设置中，DCBUS 协议，如图 17-1 所示：

☐ 协议设置	
☐ 通信协议	DCBUS
帧头	0xAA55
站号	0
启用CRC	启用
自动上传	启用
写命令应答	禁用

图 17-1 DCBUS

1. 帧头：2byte，默认为 0xAA55,用户可自定义
2. 站号：1byte，默认为站号 0，255 为广播地址。**PS:一般在 RS485 总线用于区分那个从站；**
3. 启用 CRC：2byte，默认是启用，若关闭，校验字段默认为 0XCCCC；帧头不参与校验；计算方法如下所示：

```
uint16 MB_cal_crc16(uint8 *buffer,uint32 n)
{
    uint16 crc,i,j,carry_flag,a;
    crc=0xffff;
    for (i=0; i<n; i++)
    {
        crc=crc^buffer[i];
        for (j=0; j<8; j++)
        {
            a=crc;
            carry_flag=a&0x0001;
            crc=crc>>1;
            if (carry_flag==1)
                crc=crc^0xa001;
        }
    }
    return crc;
}
```

4. 自动上传：当开启后，用户操作控件会自动上传指令给用户主板

5. 写命令应答：用户主板给屏幕写寄存器时候，会自动应答写成功
读、写、应答的帧格式如下所示：

帧头 (2byte)	站号 (1byte)	长度 (1byte)	功能码 (1byte)	地址 (2byte)	数据 (n byte)	校验 (2byte)
0xAA55	0x00~0xFF	功能码+...+校验	0xF1/0xF2	0x0000~0xFFFF	0xCCCC

17.1.1 写变量存储器指令指令 (0xF1)

以向 1000 变量地址里写数值 100 为例：AA550007F110000064CCCC

- AA55 表示：帧头；
- 00 表示：站号；
- 07 表示：数据长度，功能码+...+校验的总字节长度；
- F1 表示：写寄存器；
- 1000 表示：变量地址（2 byte）；
- 0064 表示：数据 100（2byte）；
- CCCC 表示：预留字段，没有开启 CRC；

若屏幕的站号为 1：AA550107F110000064CCCC

若屏幕的站号为 1、开启 CRC：AA550107F1100000647289

屏幕开启应答：

若屏幕没有开启 CRC：AA550003F1CCCC

若屏幕的站号为 1：AA550103F1CCCC

若屏幕的站号为 1、开启 CRC：AA550103F1B0B4

17.1.2 读变量存储器指令指令 (0xF2)

以读 1000 变量地址里的数值（假设当前数值为 100）为例：AA550006F2100001CCCC

- AA55 表示：帧头；
- 00 表示：站号；
- 06 表示：数据长度；
- F2 表示：读寄存器；
- 1000 表示：变量起始地址（2 byte）；
- 01 表示：读取的地址个数，范围为 1~255 个（1byte）；
- CCCC 表示：预留字段，没有开启 CRC；

屏收到主板的指令，会返回读应答指令：AA550008F21000010064CCCC

- AA55 表示：帧头；
- 00 表示：站号；
- 08 表示：数据长度；
- F2 表示：读寄存器；
- 1000 表示：变量起始地址（2 byte）；
- 01 表示：寄存器数据的长度，单位 1 word；
- 0064 表示：寄存器值，100
- CCCC 表示：预留字段，没有开启 CRC；

若屏幕的站号为 1：

主板请求: AA550106F2100001CCCC
屏幕返回: AA550108F21000010064CCCC

若屏幕的站号为 1、开启 CRC :

主板请求: AA550106F21000017B77
屏幕返回: AA550108F210000100648CFD

17.1.3 屏幕修改变量上传主板 (0xF2)

若开启“自动上传”，用户在屏幕上修改控件，从而修改寄存器值，可自动发出指令告诉主板寄存器值变化。如，将寄存器 LW1000 修改为 100: AA550008F21000010064CCCC

- AA55 表示：帧头；
- 00 表示：站号；
- 08 表示：数据长度；
- F2 表示：读寄存器；
- 1000 表示：变量起始地址（2 byte）；
- 01 表示：寄存器数据的长度，单位 1 word；
- 0064 表示：寄存器值，100
- CCCC 表示：预留字段，没有开启 CRC；

17.2 XGUS

兼容历史协议，支持模式 1 和模式 2，如图 17-2 所示：

<input type="checkbox"/> 通信协议	XGUS
帧头	0x5AA5
启用CRC	启用
自动上传	启用
写命令应答	启用
模式	模式1
<input type="checkbox"/> 安全登录	模式1
安全等级	模式2

图 17-2 XGUS

1. 帧头：2byte，默认为 0xAA55,用户可自定义；
2. 启用 CRC：2byte，默认是启用；
3. 自动上传：当开启后，用户操作控件会自动上传指令给用户主板；
4. 写命令应答：用户主板给屏幕写寄存器时候，会自动应答写成功；
5. 模式：兼容两种模式
 - 模式 1：系统变量，使用 80 写寄存器；
 - 模式 2：系统变量 0x000~0x1000 部分，0x1001 开始才是用户变量地址；

17.3 Modbus

屏幕支持标准的 Modbus-RTU 协议，可以设置为主机或从机。

17.3.1 主机模式-ModbusMaster

当屏幕设置为主机时候，配置如图 17-3 所示：

□ 协议设置	
□ 通信协议	MobusMaster
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input checked="" type="checkbox"/>
□ 与PLC同步画面	<input checked="" type="checkbox"/>
画面写入地址	
画面取自地址	
□ 从站数目	3
从站1	2
从站2	4
从站3	3

图 17-3 ModbusMaster

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于此值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：可设置为自动和 0x10；
 - 自动：默认写单个寄存器用 0x06 指令，写多个寄存器用 0x10 指令；
 - 0x10：所有写操作均匀 0x10 写；
7. 离线优化读取：勾选后，离线的从机，不在发送指令请求；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：默认为 1 个。多个从站，常用于 RS485 总线。界面配置默认最大 10 个，若要支持更多从站，可以在 LUA 脚本设置。
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.3.2 主机模式-ModbusSlave

当屏幕设置为从机时候，配置如图 17-4 所示：

□ 协议设置	
□ 通信协议	MobusSlave
从机站号	1
延时应答	20
□ 与PLC同步画面	<input checked="" type="checkbox"/>
画面写入地址	
画面取自地址	

图 17-4 ModbusSlave

1. 从机站号：从机的站号 ID
2. 延时应答：接收到主机请求，延时一段时间后，在应答主机；
3. 与 PLC 同步画面
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址
 - 画面取自地址：PLC 地址的值变化时，切换到对于画面（自动为 0xFFFF）

17.4 三菱-FX2N

17.4.1 协议说明

屏幕支持标准的 FX2N 协议，配置如图 17-5 所示：

□ 协议设置	
□ 通信协议	FX2N
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
模式	标准
与PLC同步画面	<input type="checkbox"/>
预设字节序	默认大端

图 17-5 FX2N 协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 模式：可支持标准模式和扩展模式；
7. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
8. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.4.2 寄存器类型

支持的寄存器类型如图 17-6 所示

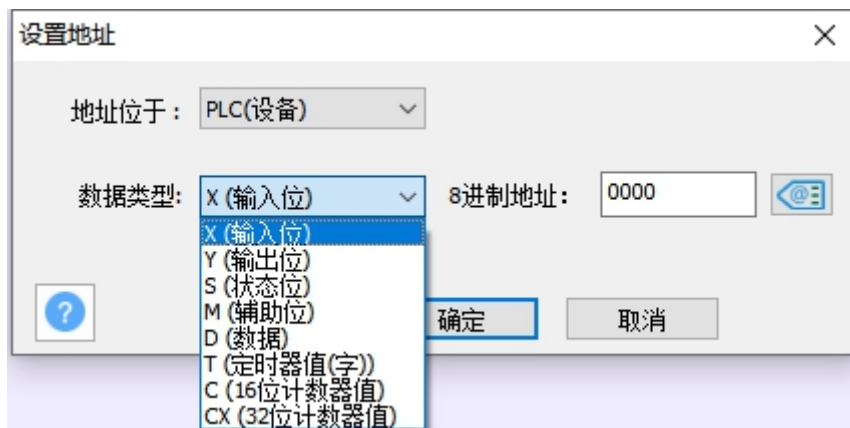


图 17-6 三菱 FX2N 寄存器

17.5 三菱-FX3U

17.5.1 协议说明

屏幕支持标准的 FX3U 协议，配置如图 17-7 所示：

<input type="checkbox"/> 通信协议	FX3U
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
模式	标准
与PLC同步画面	<input type="checkbox"/>
<input type="checkbox"/> 预设字节序	重新指定
短整型	12
长整型	1234
单精度浮点	1234
超长整形	12345678
双精度浮点	12345678

图 17-7 FX3U 协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 模式：可支持标准模式和扩展模式；
7. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；

- 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
8. 预设字节序：
- 大端模式：默认是大端
 - 重新指定：可针对短整型、长整型、单精度、超长整型、双精度浮点数指定字节序

17.5.2 寄存器类型

支持的寄存器类型如图 17-8 所示

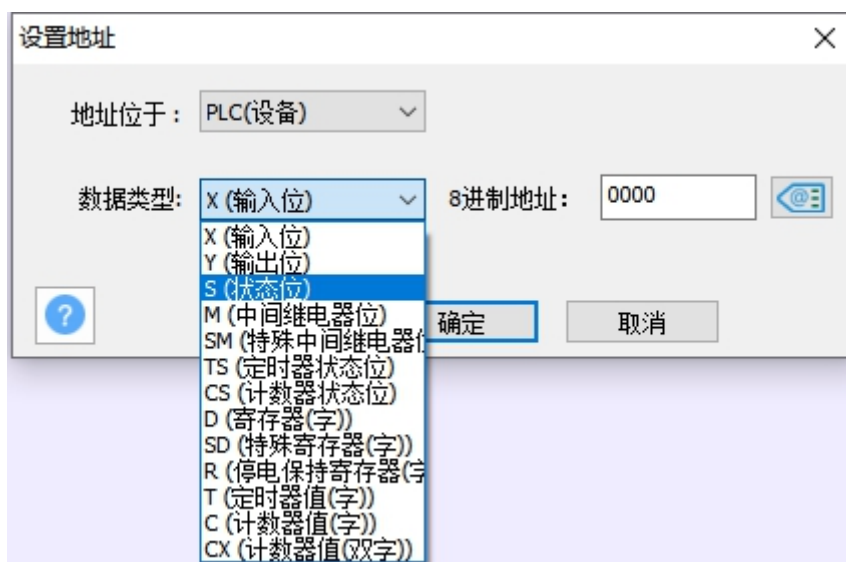


图 17-8 三菱 FX3U 寄存器

17.6 台达 PLC - DELTA(DVP)

17.6.1 协议说明

支持标准的台达协议，通讯协议 ASSIC 模式。

起始 字源	通讯地址		指令码		数据内容			校验		结束符	
STX	ADR1	ADR0	CMD1	CMD0	DATA0	...	DATA _n	LCR1	LCR0	CR	LF

协议选择配置如图 17-9 所示

□ 协议设置	
□ 通信协议	DELTA(DVP)
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
□ 从站数目	1
从站1	1
预设字节序	默认大端

图 17-9 台达 PLC-DELTA(DVP)协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.6.2 寄存器类型

支持的寄存器类型如图 17-10 所示

图 17-10 台达 PLC - DELTA(DVP)寄存器

17.7 信捷 PLC - XINJIE(XC)

17.7.1 协议说明

支持标准的信捷 PLC - XC 系列，协议图 17-11 设置如所示

□ 协议设置	
□ 通信协议	XINJIE(XC)
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
□ 从站数目	1
从站1	1
预设字节序	默认大端

图 17-11 信捷 PLC-XC 协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.7.2 寄存器类型

支持的寄存器类型如图 17-12 所示：

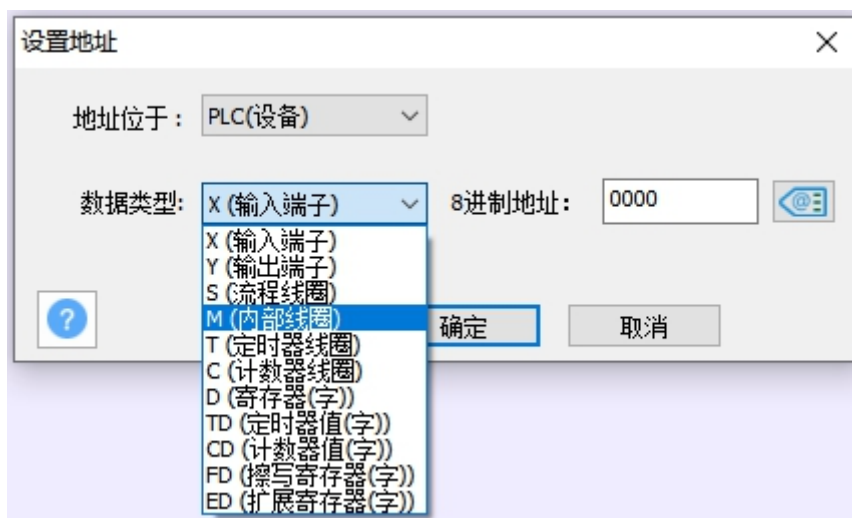


图 17-12 信捷 PLC-寄存器

17.8 信捷 PLC - XINJIE(XD)

17.8.1 协议说明

支持标准的信捷 PLC - XD 系列，协议设置如图 17-13 所示：

<input type="checkbox"/> 协议设置	
<input type="checkbox"/> 通信协议	XINJIE(XD)
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
<input type="checkbox"/> 从站数目	1
从站1	1
预设字节序	默认大端

图 17-13 信捷 PLC-XD 协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；

9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整型、单精度、超长整型、双精度浮点数指定字节序

17.8.2 寄存器类型

支持的寄存器类型如图 17-14 所示：

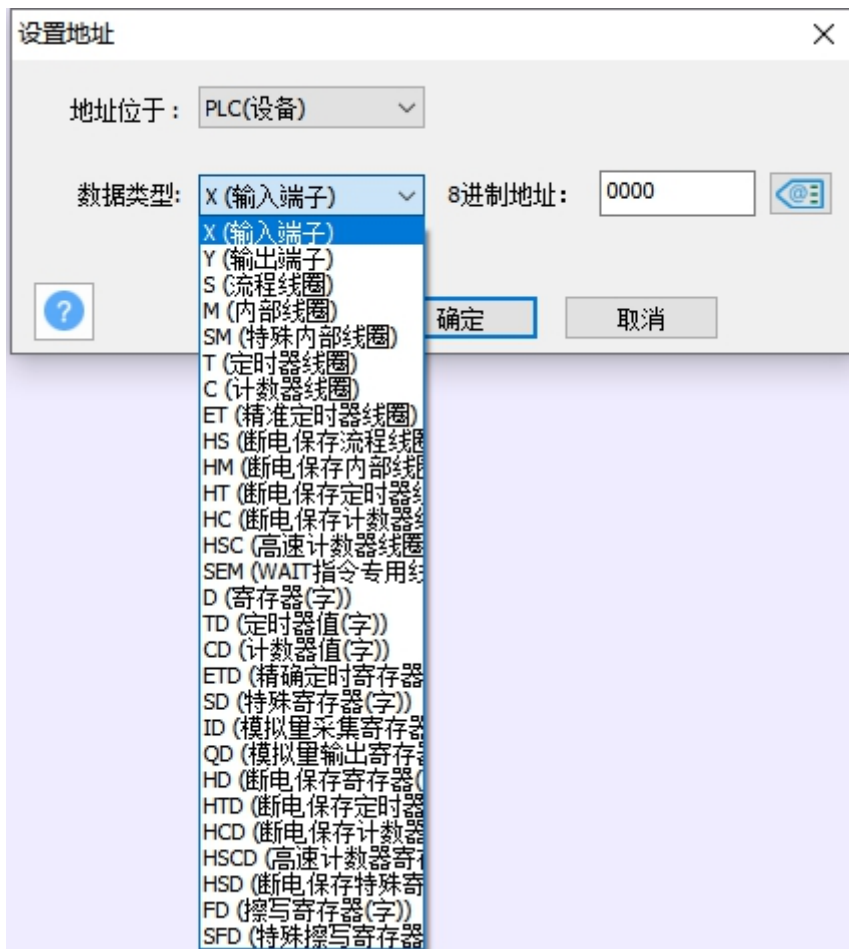


图 17-14 信捷 PLC-XD 寄存器

17.9 永宏 PLC - FATEK(FB)

17.9.1 协议说明

支持标准的永宏 PLC - FATEK(FB)系列，协议设置如图 17-15 所示

□ 协议设置	
□ 通信协议	FATEK(FB)
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
□ 从站数目	1
从站1	1
预设字节序	默认大端

图 17-15 永宏 PLC-FATEK(FB)协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.9.2 寄存器类型

支持的寄存器类型如图 17-16 所示：

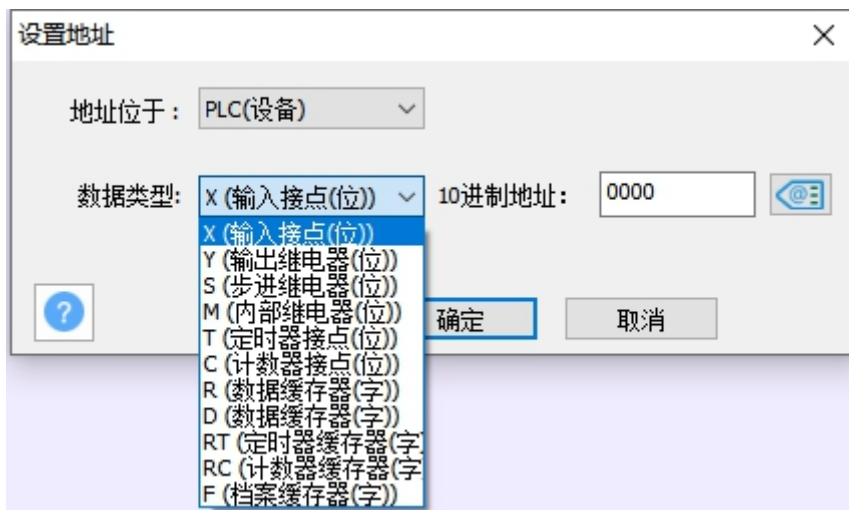


图 17-16 永宏 PLC-FATEK(FB)寄存器

17.10 海为 PLC -HAIWELL(N/S)

17.10.1 协议说明

支持标准的海为 HAIWELL(N/S)协议，协议说明如图 17-17 所示：

□ 协议设置	
□ 通信协议	HAIWELL(N/S) ▼
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
□ 从站数目	1
从站1	1
预设字节序	默认大端

图 17-17 海为 PLC-HAIWELL(N/S)协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；

9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整型、单精度、超长整型、双精度浮点数指定字节序

17.10.2 寄存器类型

支持的寄存器类型如图 17-18 所示

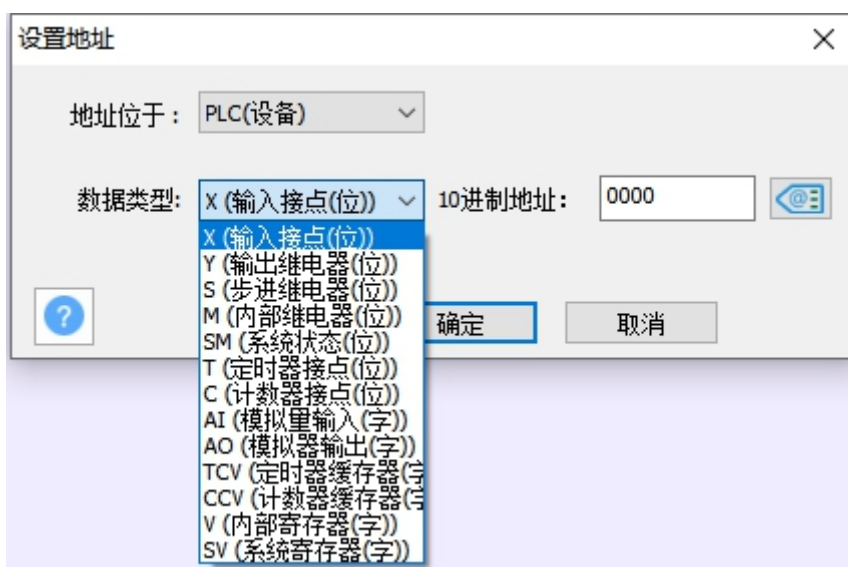


图 17-18 海为 PLC-HAIWELL(N/S)寄存器

17.11 显控-SAMKOON

17.11.1 协议说明

支持标准的显控 SAMKOO 协议，协议说明如图 17-19 所示：

<input type="checkbox"/> 通信协议	SAMKOON
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
<input type="checkbox"/> 从站数目	1
从站1	1
预设字节序	默认大端

图 17-19 显控 PLC-协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；

3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.11.2 寄存器类型

支持的寄存器，如图 17-20 所示

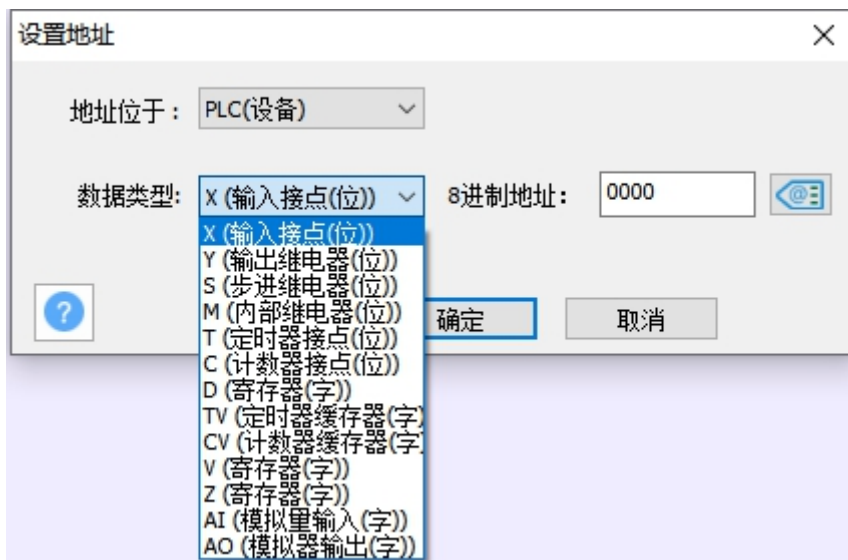


图 17-20 显控 PLC-寄存器

17.12 艾默生 - EMERSON

17.12.1 协议说明

支持标准的艾默生 PLC 协议，协议设置如图 17-21 所示

□ 协议设置	
□ 通信协议	EMERSON
最多读取数	120
读取间隔数	5
写重试次数	3
超时时间	1000
间隔时间	20
写寄存器命令	自动
离线读取优化	<input type="checkbox"/>
与PLC同步画面	<input type="checkbox"/>
□ 从站数目	1
从站1	1
预设字节序	默认大端

图 17-21 艾默生 PLC-协议说明

1. 最多读取数：单条指令读取最多字节数；
2. 读取间隔数：地址间隔小于次值，可以合并读取；
3. 写重试次数：写入失败时候，最大重试次数；
4. 超时时间：主机等待从机应答时间，单位毫秒 ms；
5. 间隔时间：主机接收到从机应答后，延时一段时间后，在发送下一条请求；
6. 写寄存器命令：自动或 0x10
7. 离线读取优化：防止从站不在线时，频发读取造成总线堵塞；
8. 与 PCL 同步画面：
 - 画面写入地址：HMI 的画面变化时，可写到指定的 PLC 地址；
 - 画面取自地址：PCL 地址的值变化时，切换到对应画面（自动为 0xFFFF）；
9. 从站数目：支持一主多从
10. 预设字节序：
 - 大端模式：默认是大端
 - 重新指定：可针对短整型、长整形、单精度、超长整形、双精度浮点数指定字节序

17.12.2 寄存器类型

支持的寄存器类型如图 17-22 所示：

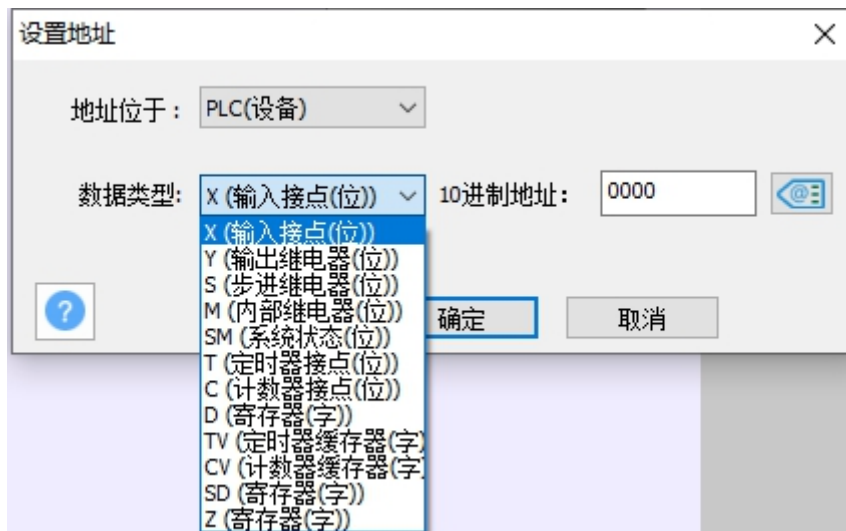



图 17-22 艾默生 PLC-寄存器

18. LUA API

VisualHMI 支持 V5.3 版本的 LUA 脚本语言,配合工程可以完成大部分的内部逻辑处理,



MCU 可以只参与数据传输部分,不参与逻辑处理。在菜单工程栏→脚本编程 ,打开默认在工程目录下生产 main.lua,且自动定义变量的数据类型,假设工程当前为 modbus 协议下,如下图 18-1 所示

```

1  ENCRYPT_0=0...--LUA脚本加密
2
3  --数据类型定义
4  VT_LW:=1...--变量地址
5  VT_RW:=2...--FLASH存储
6  VT_0x:=10...--线圈
7  VT_lx:=11...--输入点
8  VT_3x:=12...--输入寄存器
9  VT_4x:=13...--保持寄存器
10
11
12  function on_init()
13  end
14
15  function on_run(screen)
16  end
17
18  function on_update(slave,vtype,addr)
19  end
20
21  function on_draw(screen_id,control_id)
22  end
23

```

图 18-1 lua 脚本

18.1 常用回调函数

18.1.1 on_init()

系统加载 LUA 脚本文件之后,立即调用此回调函数,通常用于执行初始化操作。

18.1.2 on_run(screen)

系统不断循环调用该函数

set_run_cycle(cycle)可设置执行周期(毫秒单位)

- screen: 当前画面 ID

18.1.3 on_update(slave, vtype, addr)

变量被设置、更改后,自动执行此函数

- slave: 站号
- vtype: 变量类型,详见变量类型定义
- addr: 变量地址

DCBUS	
1	系统变量
2	内存变量
XGUS	

1	系统变量
2	内存变量
Modbus	
1	系统变量
2	内存变量
10	1x 线圈
11	2x 离散输入
12	3X 输入寄存器
13	4X 保持寄存器
Fx2N	
1	系统变量
2	内存变量
10	X 寄存器
11	Y 寄存器
12	S 寄存器
13	M 寄存器
14	D 寄存器
15	T 寄存器
16	C 寄存器
17	C32 寄存器

```
function on_update(slave, vtype, addr)
  if vtype == 13
  then
    if addr == 29 --
    then
      set_bit(13, 30, 0)
    end
    .....
  end
end
```

18.1.4 on_screen_change(screen)

当画面需要切换时，自动执行此函数

- screen: 目标画面 ID

18.1.5 on_usb_inserted(driver)

U 盘插入时，执行此回调函数，dirver 为 U 盘的盘符

18.1.6 on_usb_removed()

U 盘拔出时，执行此回调函数

18.1.7 on_sd_inserted(dir)

SD 卡插入通知，dir 盘符路径

18.1.8 on_sd_removed()

SD 卡拔出通知

18.2 读写寄存器函数

18.2.1 set_notify(enable)

使能串口通知，启用后设置变量的值可触发串口命令发送。

- enable: 0 禁止，1 使能(默认值)

18.2.2 select_slave(slave_id)

设置从站地址，用于 MODBUS 协议。

- slave_id: 0~128

18.2.3 set_endian(en)

设置变量的大小端，在主从机模式下生效

- en: 变量地址: 0-大端(默认值)，1-小端

```
set_endian(1)
```

18.2.4 set_bit(vtype, addr, value)

设置位寄存器：如 Modbus 的线圈、F2XN 的 X、Y、S、M 寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

18.2.5 get_bit(vtype, addr)

读取位寄存器：如 Modbus 的线圈、F2XN 的 X、Y、S、M 寄存器，返回数值

- vtype: 变量类型
- addr: 变量地址

```
local state = get_bit(VT_0x, 0x0000)
```

18.2.6 set_uint16(vtype, addr, value)

设置无符号短型（uint16）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

18.2.7 get_uint16(vtype, addr)

读取无符号短型（uint16）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local state = get_uint16 (VT_LW, 0x0000)
```

18.2.8 set_int16(vtype, addr, value)

设置有符号短型（int16）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

18.2.9 get_int16(vtype, addr)

读取有符号短型（int16）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local state = get_int16 (VT_LW, 0x0000)
```

18.2.10 set_uint32(vtype, addr, value)

设置无符号短型（uint32）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

18.2.11 get_uint32(vtype, addr,)

获取无符号短型（uint32）寄存器

- vtype: 类型
- addr: 变量地址

18.2.12 set_int32(vtype, addr, value)

设置有符号短型（int32）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

18.2.13 get_int32 (vtype, addr)

读取有符号整型（int32）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local value = get_uint32(VT_LW, 0x0000)
```

18.2.14 set_uint64(vtype, addr, value)

设置无符号长整型（uint64）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

```
set_uint64(VT_LW, 0x0000, 1234)
```

18.2.15 get_uint64 (vtype, addr)

读取无符号整型（int64）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local value = get_uint64(VT_LW, 0x0000)
```

18.2.16 set_int64(vtype, addr, value)

设置有符号长整型（int64）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

```
set_int64(VT_LW, 0x0000, -1234)
```

18.2.17 get_int64 (vtype, addr)

读取有符号整型（int64）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local value = get_int64(VT_LW, 0x0000)
```

18.2.18 set_float(vtype, addr, value)

设置单精度浮点数（float）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

```
set_float(VT_LW, 0x0000, 1.234)
```

18.2.19 get_float (vtype, addr)

读取单精度浮点数（float）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local value = get_float(VT_LW, 0x0000)
```

18.2.20 set_double(vtype, addr, value)

设置双精度浮点数（double）寄存器

- vtype: 类型
- addr: 变量地址
- value: 写入的值

```
set_double(VT_LW, 0x0000, 1.234)
```

18.2.21 get_double (vtype, addr)

读取单精度浮点数（float）寄存器

- vtype: 变量类型
- addr: 变量地址

```
local value = get_double(VT_LW, 0x0000)
```

18.2.22 set_string(vtype, addr, strings)

设置字符串类型寄存器

- vtype: 类型
- addr: 变量地址
- strings: 字符串数据

18.2.23 get_string (vtype, addr)

读取字符串变量

- vtype: 变量类型
- addr: 变量地址

```
local string = get_string (VT_LW, 0x0000)
```

18.2.24 set_uint16_ex(vtype, addr, value1,value2, ..., value120)

设置连续寄存器数据

- vtype: 类型
- addr: 变量起始地址，最大可以写 120 个连续寄存器
- value1: 寄存器 1 的值
- value2: 寄存器 2 的值
-
- value120: 寄存器 120 的值

```
local r_addr = 0x6060
set_uint16_ex(vtype, r_addr,
  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
  21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
  41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,
  61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,
  81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,
  101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,
  119,120)
```

18.2.25 set_array(vtype, addr, buff)

设置连续的寄存器数据

- vtype: 类型
- addr: 变量起始地址，最大可以写 120 个连续寄存器
- buff: word (字) 数组，最大 120 个字

```
local buff = {}
local r_addr = 0x6060
local index = 120
local data = 0

for i = 1, 120
do
  buff[i] = i
end
```

```
set_array(vtype, r_addr, buff)
```

18.2.26 start_read(index,vtype, addr,quantity)

开始读取多个连续的变量，系统自动发指令读取变量地址。

只能用于主机模式，例如 Modbus-Master、FX2N 等。

该函数没有返回值，需要配合 get_xx 获取变量值

当前画面用到的变量地址，系统会自动读取，无需使用 start_read。

如果要在脚本中使用某些变量做判断条件（此变量没有关联当前画面的控件），需要使用 start_read 读取。

- index: 索引，通常和停止读取搭配 stop_read
- vtype: 变量类型
- addr: 开始读取的地址
- quantity: 读取的个数

18.2.27 stop_read(index)

停止读取多个连续的变量

只能用于主机模式，例如 Modbus-Master、FX2N 等。

- index: 索引，通常和开始读取搭配 start_read

18.2.28 stop_all_read()

停止所有 start_read 开启的读取指令。

只能用于主机模式，例如 Modbus-Master、FX2N 等。

18.3 绘图函数

18.3.1 on_draw(screen,control)

控件自绘接口函数，要使用自绘功能，控件 ID 不能为 0。

当界面的显示内容需要更新时，系统自动调用此函数，用户在此函数中添加自定义的绘图操作。

此函数为系统回调函数，用户不要直接调用。

下面几种情况会触发此函数：

- 界面有动画播放、视频播放、RTC 时间显示的动态刷新；
- 用户操作屏幕控件控件；
- 通过 LUA 脚本或串口指令更新控件；
- 通过执行 redraw；

总之，界面上有任何变化，都会触发此回调函数。

18.3.2 redraw()

发送重绘请求，触发 on_draw 的执行。

18.3.3 set_pen_color(color)

设置画笔的颜色，RGB565，用于指定线、矩形、圆等的颜色

- color: RGB565

18.3.4 draw_line(x0,y0,x1,y1,width)

绘制直线

- x0,y0 起始点坐标

- x1,y1 结束点坐标
- width 为线条的厚度, 1~10

18.3.5 draw_rect(x0,y0,x1,y1,fill)

绘制矩形

- x0,y0 左上角坐标
- x1,y1 右下角坐标
- fill 为 0 不填充, 1 填充

18.3.6 draw_rect_alpha(x0,y0,x1,y1,alpha)

绘制实心的半透明矩形

- x0,y0 左上角坐标
- x1,y1 右下角坐标
- alpha 透明度 0 全透明~255 不透明

18.3.7 draw_circle(x,y,r,fill)

绘制圆形

- x,y 圆的中心坐标
- r 圆的半径
- fill 为 0 不填充, 1 填充

18.3.8 draw_ellipse(x0,y0,x1,y1,fill)

绘制椭圆

- x0,y0 左上角坐标
- x1,y1 右下角坐标
- fill 为 0 不填充, 1 填充

18.3.9 draw_image(image_id,frame_id,dstx,dsty,width,height,srcx,srcy)

绘制图片

- image_id 图片资源的 ID
- frame_id 对应图标, 可以设置帧 ID, 其他图片固定为 0
- dstx 图片显示 X 坐标
- dsty 图片显示 Y 坐标
- width 图片显示宽度
- height 图片显示高度
- srcx 图片裁剪 X 坐标
- srcy 图片裁剪 Y 坐标

18.3.10 draw_text(text,x,y,w,h,font_id,size,color,align)

显示文字, text 字符串

- x 显示 X 坐标
- y 显示 Y 坐标
- w 显示宽度
- h 显示高度
- font_id: 索引
- size: 字体大小

- color 颜色 RGB565
 - align 对齐方式
 - bit0~bit1 水平对齐方式, 0 左对齐, 1 居中对齐, 2 右对齐
 - bit2~bit3 垂直对齐方式, 0 上对齐, 1 居中对齐, 3 下对齐
- ```
draw_text('hello Dc', 220, 230, 360 ,30, 1,16, 0xFFE0, 4)
```

#### 18.3.11 load\_surface (filename)

加载图片到图层

- filename: 图片文件, 支持 JPEG/PNG

例如: `surface = load_surface ("3:/test.jpg")`

图层不再使用时, 需要调用 `destroy_surface` 进行销毁, 否则会导致内存泄漏。

#### 18.3.12 destroy\_surface (surface)

销毁图层

- surface: 图层资源指针

#### 18.3.13 destroy\_all\_surface()

销毁所有图层指针

#### 18.3.14 draw\_surface (surface,dstx,dsty,width,height,srcx,srcy)

绘制图层, 支持绘制 jpg、png 格式的图片, 在 `on_draw()` 系统函数里调用

- surface: 图层资源指针
- dstx: 图片显示 X 坐标
- dsty: 图片显示 Y 坐标
- width: 图片显示宽度[可选]
- height: 图片显示高度[可选]
- srcx: 图片裁剪 X 坐标[可选]
- srcy: 图片裁剪 Y 坐标[可选]

平铺显示: `draw_surface(surface, dstx, dsty)`

裁剪显示: `draw_surface(surface, dstx, dsty, width, height ,srcx, srcy)`

#### 18.3.15 get\_surface\_size (surface)

获取指定图层指针的图像大小, 返回, 宽度和高度

```
local cur_w,cur_h = get_surface_size (surface)
```

#### 18.3.16 clear\_image\_buffer()

清除内部图片资源缓存

#### 18.3.17 video\_shoot(filepath, width, height)

截取视频/AV 窗口的内容, 保存为 bmp 图片。

**注意**, 默认支持 AV 截图, 视频截图需要特定固件

- filepath: 图片存储路径
- width: 宽度

- height: 高度

### 18.3.18 screen\_shoot (filepath)

截取整个屏幕的内容，保存为 jpg 图片

- filepath: 图片存储路径

## 18.4 数据记录

### 18.4.1 record\_get\_count(sn)

- sn:资料记录索引，从 0 开始  
获取该笔资料的记录数

```
local cnt = record_get_count(index)
```

### 18.4.2 record\_modify\_data(sn, index, data)

修改数据记录的内容

- sn:资料记录索引，从 0 开始
- index:记录的行号，从 0 开始
- data: 字数组 (word)，下表从 1 开始

**注意：若开启存储模式，则不允许修改记录**

假设修改第 0 笔资料，为 uint16，共 100 个数据，则修改第 0 行记录为 1~100 实例如下：

```
for i = 1, 100
do
 dataTb[i] = i
end
record_modify_data(0, 0, data)
```

假设修改第 0 笔资料，为 uint32，共 100 个数据，则修改第 0 行记录为 1~100 实例如下：

```
for i = 1, 200
do
 if i % 2 == 0
 then
 dataTb[i] = i
 else
 dataTb[i] = 0
 end
end
record_modify_data(0, 0, data)
```

uint32 共 100 个参数，共占 200 个字，所有，data 的大小为 200。依次类推，64 位寄存器为 400

### 18.4.3 record\_write\_data(sn, data)

添加数据记录的内容

- sn:资料记录索引，从 0 开始
- data: 字数组 (word)，下表从 1 开始

### 18.4.4 record\_read\_data(sn, index)

读取数据记录的内容

- sn:资料记录索引, 从 0 开始
- index:记录的行号, 从 0 开始

返回两个参数: data, timestamp

- data: 字数组,数据内容, 下标从 1 开始
- timestamp:时间戳

```
local data,timestamp = record_read_data(0, 0) -- 读取第 0 笔资料, 第 1 行数据
```

#### 18.4.5 record\_modify\_string(sn, index, strings)

修改字符串类型的数据记录

**注意:** 若开启存储模式, 则不允许修改记录

- sn:资料记录索引, 从 0 开始
- index:记录的行号, 从 0 开始
- strings: 字符串。每一列用 “;” 隔开。

```
record_modify_string(0, 0, 'item1;item2;item3;item4;item5;') -- 修改第 0 笔资料, 第 1 行数据的数据位 'item1;item2;item3;item4;item5;'
```

#### 18.4.6 record\_read\_string(sn, index)

读取字符串类型的数据记录

- sn:资料记录索引, 从 0 开始
  - index:记录的行号, 从 0 开始
- 返回两个参数: strings, timestamp
- strings: 字符串。每一列以 “; ” 隔开
  - timestamp:时间戳

```
local strings, timestamp = record_read_string(0, 0)
> print('strings = '..strings)
> strings = item1;item2;item3;item4;item5;
```

#### 18.4.7 record\_write\_string(sn, strings)

添加数据记录的内容

- sn:资料记录索引, 从 0 开始
- strings: 字符串, 每一列参数以 “; ” 隔开

#### 18.4.8 record\_clear(sn)

清除记录

- sn:资料记录索引, 从 0 开始

### 18.5 定时器

#### 18.5.1 start\_timer(timer\_id, timeout, countdown, repeat)

启动定时器, 超时后系统自动调用 on\_timer

- timer\_id-定时器 ID, 0~31
- timeout-超时时间, 单位毫秒
- countdown-0 顺计时, 1 倒计时
- repeat-重复次数, 0 表示无限重复

#### 18.5.2 stop\_timer(timer\_id)

停止定时器

- timer\_id-定时器 ID, 0~31

### 18.5.3 on\_timer(timer\_id)

定时器超时回调函数

- timer\_id-定时器 ID, 0~31

### 18.5.4 get\_timer\_value(timer\_id)

获取定时器当前计时时间, 单位毫秒

- timer\_id-定时器 ID, 0~31

## 18.6 告警

一般情况使用告警设置对话框就可以满足告警需求, 不需要使用下面的 API。

此处的告警指通过脚本触发告警和动态显示告警内容。

一般用于下列情形

- 告警条目比较多, 内容具有重复规律, 只有细微差异
- 告警条件具有规律性

### 18.6.1 warning\_set\_mode(en)

开启告警模式之后, warning\_set 才能生效。

- en: 0 关闭, 1 开启告警触发模式

### 18.6.2 warning\_set(warning\_id,value,count)

触发或关闭指定 id 位置的告警, 此功能需要提前设置 warning\_set\_mode(1)。

- warning\_id: 起始告警 id
- value: 告警值, 对应的 bit 为 1 触发告警, 为 0 关闭告警。
- count: 连续告警个数

```
local err_val = get_uint16(VT_V, 0x1D00)--读取 0x1D00 寄存器值
warning_set(0, err_val , 16)--将 0x1D00 寄存器的值, 每一个位, 对应到告警 0~15 范围
```

### 18.6.3 on\_parse\_warning(id, text)

告警解析回调函数, 通过判断告警 id, 返回描述告警内容的字符串。

参数说明:

- Id: 告警 ID
- text: 告警内容

返回参数说明:

- str: 返回的字符串告警内容
- encode: 字符编码。选填, 默认为 UTF8。若 encode=1, 则表示 UTF8 编码

```
_warningTb = {
--1~96
 '翅片 1 温度探头故障',
 '翅片 2 温度探头故障',
 '翅片 3 温度探头故障',
 '翅片 4 温度探头故障',
 '回气 1 温度探头故障',
 '回气 2 温度探头故障',
```

```

 '回气 3 温度探头故障',
 '回气 4 温度探头故障',
 '排气 1 温度探头故障',
 '排气 2 温度探头故障',
 '排气 3 温度探头故障',
 '排气 4 温度探头故障',
 '蒸发 1 温度探头故障',
 '蒸发 2 温度探头故障',
 '蒸发 3 温度探头故障',
 '蒸发 4 温度探头故障',

 '冷凝 1 温度探头故障',
 '冷凝 2 温度探头故障',
 '冷凝 3 温度探头故障',
 '冷凝 4 温度探头故障',
 '房内温度探头故障',
 ...
 ...
 ...
 '室内翅片 1 探头故障',
 '室内翅片 2 探头故障',
 '室内翅片 3 探头故障',
 '室内翅片 4 探头故障',
}

function on_parse_warning(id, text)

--设备 1:id0~id95
--...
--...
--...
--设备 8:id673~id768

local slaveId = (id // 96) + 1 --第几个告警
local msgId = id % 96 --告警类型 ID
local curwarnMsg = '' --告警描述

if msgId == 0
then
 msgId = 1
else
 msgId = msgId + 1
end

```

```
curWarnMsg = math.ceil(slaveId)..'device : '.._warningTb[msgId]
return curWarnMsg
end
```

承接上文 [18.6.2](#) 对 `warning_set()` 的说明，假设读取 0x1D00 寄存器值为 0x00E，即是寄存器的 bit1~bit3 位为 1，由于前面给该寄存器对应到 0~15 的告警 id，即是变量 0x1D00 为 0x000E 的时候，对应的告警 ID 为 1~3，此时可以通过告警 1~3，进行判断，返回告警内容。

## 18.7 串口

### 18.7.1 uart\_setup(ch,baudrate,databits,stopbit,parity)

串口参数设置

- ch: 串口通道，默认 0
- baudrate: 波特率值
- databits: 数据位 7~8, 0 表示 7 位，1 表示 8 位
- stopbit: - 0-1bit, 1-1.5bit
- parity: -0 无校验，1-ODD 奇校验，2-EVEN 偶校验

### 18.7.2 uart\_send(ch,packet)

发送字节数组

- ch: 串口通道，默认 0
- packet: 数组，下标从 1 开始

```
function uart_doorState(state)
local door_buff = {0x5A,0x07,0x82,0x00,0x01}
door_buff[6] = state
uart_send_data(0, door_buff)
end
```

### 18.7.3 on\_uart\_recv(ch,packet)

串口接收回调，自定义串口协议模式时才会触发。

- ch: 串口通道，默认 0
- packet: 数组，下标从 1 开始

## 18.8 音视频

### 18.8.1 play\_sound(filename)

播放指定的音乐文件，例如播放屏内音频文件（wav/mp3）：

```
play_sound('3:/sounds/welcome.wav')
```

### 18.8.2 play\_video(file,left,top,width,height)

播放视频（mp4 格式）

- file: 文件路径
- left: 起始坐标 x
- top: 起始坐标 y
- width: 视频显示的宽度

- **height:** 视频显示的高度

全屏播放时，位置参数不需要填写

播放屏内视频: `play_ video('3:/Videos/1.mp4')`

播放屏 SD 视频: `play_ video('1:/1.mp4')`

### 18.8.3 pause\_video()

暂停视频播放

### 18.8.4 resume\_video()

恢复视频播放

### 18.8.5 stop\_video()

停止视频播放

### 18.8.6 av\_init(show,channel,left,top,width,height)

av 输入初始化

- **show:** 0 隐藏, 1 显示
- **channel:** 通道编号 0/1
- **left:** 起始坐标 x
- **top:** 起始坐标 y
- **width:** 视频显示的宽度
- **height:** 视频显示的高度

### 18.8.7 av\_get\_status()

获取 AV 播放状态

```
status = av_get_status()
```

### 18.8.8 av\_select(ch)

切换 AV 通道

- **ch:** 0-切换到通道 1, 1-切换到通道 2

### 18.8.9 av\_show(show)

显示/隐藏 AV 播放

- **show:** 0-隐藏, 1-显示

## 18.9 文件读写

### 18.9.1 dofile(name)

加载 lua 文件, 用于 lua 脚本分模块编辑, 创建的\*.lua 文件放在工程目录下, 和 main.lua 统计, 在 on\_init() 里调用即可。

```
dofile('test.lua')
```

### 18.9.2 list\_dir(path)

遍历指定目录下的文件和文件夹, 成功返回 true, 失败返回 false

通过 on\_list\_dir 回调函数返回文件夹的内容



### 18.9.3 on\_list\_dir(path,filename,type,fsize)

- path-文件路径
- filename-文件名称
- type-0 文件夹， 1 文件
- fsize-文件大小

### 18.9.4 file\_open(path,mode)

打开文件，成功返回 true，失败 false

- path-文件路径
- mode-0 只读，1 创建文件写入

例如：

```
打开文件用于读取 file_open(path, 0)
创建文件用于写入 file_open(path, 1)
```

### 18.9.5 file\_close()

关闭文件，成功返回 true，失败 false

### 18.9.6 file\_size()

获取当前文件大小，返回字节数

### 18.9.7 file\_seek(offset)

定位文件读取位置，成功返回 true，失败 false

- offset-文件偏移位置

### 18.9.8 file\_read(count)

读取文件内容，成功返回 table 数组，失败返回 nil

- count-读取字节数，最大读取 2048 个字节

### 18.9.9 file\_write(data)

写文件内容，成功返回 true，失败返回 false

- data-待写入的 table 数组，索引从 1 开始，最大一次性写 2048 个字节

### 18.9.10 file\_delete(path)

- path: 待删除的文件路径

### 18.9.11 file\_copy(src\_path, dst\_path)

- src\_path: 源文件路径
- dst\_path: 目标文件路径

### 18.9.12 on\_copy\_file\_process(status,filesize,transfersize)

文件拷贝进度，配合 file\_copy(src\_path, dst\_path)使用，当调用 file\_copy，自动回调到 on\_copy\_file\_process 函数

- status: 状态.0-失败，1-拷贝中，2-拷贝完成
- filesize: 被拷贝的文件大小
- transfersize: 当前回调拷贝的字节数

## 18.10 设置窗口控件

### 18.10.1 set\_screen(screen)

设置当前画面，执行画面切换

### 18.10.2 get\_screen()

获取当前画面 ID

### 18.10.3 show\_dialog(screen, x, y, alpha)

设置对话框，执行切换对话框操作

- screen: 画面 ID
- x,y: 左上角起始位置
- alph: 透明度

show\_dialog(0, 184, 108, 50) --显示对话框，坐标（184,105），50%的透明度  
show\_dialog(-1, 184, 108, 50) ---1，表示关闭对话框，后面参数任意

### 18.10.4 wgt\_set\_pos(screen,control,x,y,w,h)

设置控件的位置

- screen: 画面 ID
- control: 控件 ID
- x,y: 左上角起始位置
- w,h: 控件的宽度、高度

--设置画面 4 控件 ID10 显示在（476,70）位置，大小为 120,36  
wgt\_set\_pos(4,10,476,70,120,36)

### 18.10.5 wgt\_set\_fcolor(screen,control, color)

设置控件的前景色：如文本控件、进度条控件

- screen: 画面 ID
- control: 控件 ID
- color: 颜色 RGB565

### 18.10.6 wgt\_set\_bcolor(screen,control, color)

设置控件的背景色：如文本控件、进度条控件

- screen: 画面 ID
- control: 控件 ID
- color: 颜色 RGB565

### 18.10.7 wgt\_set\_param(screen,control, param,value)

设置控件的属性接口，保留扩展接口

- screen: 画面 ID
- control: 控件 ID
- param: 设置标识码  
0x30: 表示数据记录控件的通道翻页  
0x31: 表示曲线控件的 Y 轴最小值  
0x32: 表示曲线控件的 Y 轴最大值
- value: 数据内容

#### 18.10.8 on\_wgt\_event(screen\_id,widget\_id,event,value)

窗口设置触发回调，控件 id 不为 0，触碰控件，自动触发该函数

- screen: 画面 ID
- control: 控件 ID
- event: 触发的事件
- value: 控件绑定寄存器的值

#### 18.10.9 refresh\_screen()

立即刷新画面

### 18.11 GPIO 控制

#### 18.11.1 gpio\_set\_in (pin)

PIN 引脚设置为输入模式

#### 18.11.2 gpio\_set\_out (pin)

PIN 引脚设置为输出模式

#### 18.11.3 gpio\_set\_value (pin,value)

设置输出 PIN 引脚为（高电平 1/低电平 0）

#### 18.11.4 gpio\_get\_value (pin)

获取输入 PIN 引脚电平（高电平 1/低电平 0）

### 18.12 其他

#### 18.12.1 get\_platform()

获取屏幕芯片信息，返回字符串类型，设备型号

#### 18.12.2 get\_version()

获取固件版本号，返回字符串

#### 18.12.3 get\_device\_uuid()

获取设备唯一表示符号，返回长度为 20 的数字字符串

#### 18.12.4 reboot()

屏幕复位

#### 18.12.5 feed\_dog()

若在某个操作执行耗时超过 5 秒，需要执行喂狗

#### 18.12.6 delay\_ms(ms)

延时函数，单位 ms，延时不宜过长，超过 5 秒回复位。

#### 18.12.7 beep(ms)

蜂鸣器鸣叫，单位 ms

#### 18.12.8 set\_run\_cycle(cycle)

设置 on\_run 的执行周期，毫秒单位

#### 18.12.9 update\_system()

立刻加载刷新配方寄存器参数，一般用于配方切换后调用

#### 18.12.10 get\_date\_time ()

获取当前日期时间

```
local year,mon,day,hour,min,sec,week = get_date_time()
```

#### 18.12.11 set\_date\_time (year,mon,day,hour,min,sec)

设置当前日期时间

```
set_date_time(year,mon,day,hour,min,sec)
```

#### 18.12.12 make\_datetime(timestamp)

时间戳转年月日时分秒

```
local year,mon,day,hour,min,sec = make_datetime(timestamp)
```

#### 18.12.13 make\_timestamp(year,mon,day,hour,min,sec)

年月日时分秒转时间戳

```
local timestamp = make_timestamp(year,mon,day,hour,min,sec)
```

#### 18.12.14 set\_pwd(level,pwd)

设置对应等级用户的密码

- level: 用户等级 0~7
- pwd: 用户数字密码

#### 18.12.15 md5(text, key)

基于 Hash 函数和密钥进行消息认证的方法，通过这个算法可以保证通信双方之前交互的消息来自对方并且没有被篡改，返回一个长度为 32 的 16 进制字符串。

- text: 目标字符串
- key: 双方规约的密钥

## 19. 免责声明

本文档提供有关广州大彩光电科技有限公司（以下简称：大彩科技）产品的信息，旨在协助客户加速产品的研发进度，在服务过程中或者其他渠道所提供的任何例程程序、技术文档、CAD 图等资料和信息都仅供参考，客户有权不使用或自行参考修改。本公司不提供任何的完整性、可靠性等保证，若是客户使用过程中因任何原因造成的特别的、偶然的或间接的损失，本公司不承担任何责任。大彩科技产品不能在用于军事、医疗、救生或维生等用途中作为唯一控制设备。

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除大彩科技在其产品的销售条款和条件中声明的责任之外，大彩科技概不承担任何其它责任。并且，大彩科技对大彩科技产品的销售和/或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。大彩科技可能随时对产品规格及产品描述做出修改，恕不另行通知。