



欣瑞达 LUA 脚本 API_V1.3

深圳市欣瑞达信息技术有限公司

欣瑞达液晶/欣瑞达串口屏
深圳市南山区高新园北松坪山路5号嘉达研发大厦A座3层
0755-26018666 400-069-8808
www.xrd-lcd.com



淘宝扫一扫



微信扫一扫



抖音扫一扫

目录

欣瑞达 LUA 脚本 API_V1.3.....	1
1. 适用范围.....	11
2. LUA 脚本介绍.....	11
3. API 接口函数.....	11
3.1 控件属性类.....	11
3.1.1 change_screen(screen)	11
3.1.2 set_value(screen, control, value)	11
3.1.3 value = get_value(screen, control)	11
3.1.4 set_visiable(screen, control, visiable)	11
3.1.5 set_enable(screen, control, enable)	12
3.1.6 set_fore_color(screen, control, color)	12
3.1.7 set_back_color(screen, control, color)	12
3.1.8 set_text(screen, control, text)	12
3.1.9 string = get_text(screen, control)	12
3.1.10 currentScreen = get_current_screen()	12
3.1.11 set_text_flicker(screen, control, cycle)	12
3.1.12 play_animation(screen, control)	12
3.1.13 stop_animation(screen, control)	13
3.1.14 histGraph_reset(screen, control)	13
3.1.15 histGraph_load(screen, control, file)	13
3.1.16 histGraph_export(screen, control, path)	13

3.2 常用回调函数.....	13
3.2.1 on_init()	13
3.2.2 on_systick()	13
3.2.3 on_control_notify(screen, control, value)	13
3.2.4 on_screen_change(screen)	13
3.2.5 on_press(state, x, y)	13
3.2.6 on_usb_inserted(driver)	14
3.2.7 on_usb_removed()	14
3.2.8 on_sd_inserted(dir)	14
3.2.9 on_sd_removed()	14
3.3 绘图函数.....	14
3.3.1 on_draw(screen)	14
3.3.2 redraw()	14
3.3.3 set_pen_color(color)	14
3.3.4 draw_line(x0, y0, x1, y1, width)	14
3.3.5 draw_rect(x0, y0, x1, y1, fill)	14
3.3.6 draw_circle(x, y, r, fill)	15
3.3.7 surface = load_surface(file_name)	15
3.3.8 destroy_surface(surface)	15
3.3.9 draw_surface(surface, dstx, dsty, width, height, srcx, srcy)	15
3.3.10 screen_shoot(file_name, x, y, width, height, quality)	15
3.3.11 draw_rect_alpha(x0, y0, x1, y1, alpha)	16
3.4 寄存器访问.....	16

3.4.1	val = get_variant(slave, type, addr_name, port)	16
3.4.2	set_variant(slave, type, addr_name, val, port)	16
3.4.3	mb_set_timeout(timeout)	16
3.4.4	buf = mb_read_coil_01 (slave, addr, quantity, pattern, port)	16
3.4.5	buf = mb_read_input_02(slave, addr, quantity, pattern, port)	17
3.4.6	buf = mb_read_reg_03(slave, addr, quantity, pattern, port)	17
3.4.7	buf = mb_read_input_reg_04(slave, addr, quantity, pattern, port)	17
3.4.8	value = mb_write_coil_05(slave, addr, status, pattern, port)	17
3.4.9	value = mb_write_reg_06(slave, addr, reg, pattern, port)	18
3.4.10	value = mb_write_coil_15(slave, addr, quantity, coils, pattern, port)	18
3.4.11	value = mb_write_reg_16(slave, addr, regs, pattern, port)	18
3.5	定时器	19
3.5.1	start_timer(timer_id, timeout, countdown, repeat)	19
3.5.2	stop_timer(timer_id)	19
3.5.3	on_timer(timer_id)	19
3.5.4	get_timer_value(timer_id)	19
3.6	串口	19
3.6.1	uart_send_data(packet, port)	19
3.6.2	uart_set_timeout(timeout, timeout_inter, port)	19
3.6.3	uart_set_baudrate(baudrate, port)	19
3.6.4	baudrate = uart_get_baudrate(port)	20
3.6.5	on_uart_rcv_data(packet, port)	20
3.6.6	uart_setup(baudrate, parity, stop_bit, data_bits, port)	20

3.7 CAN 接口	20
3.7.1 canbus_open(index, baudrate, listen_mode, loop_back)	20
3.7.2 canbus_close(index)	20
3.7.3 canbus_write(index, identifier, dlc, rtr, ide, data)	20
3.7.4 on_canbus_recv(index, identifier, dlc, rtr, ide, data)	21
3.8 音视频	21
3.8.1 play_sound(file_name)	21
3.8.2 on_audio_callback(state)	21
3.8.3 set_volume(level)	21
3.8.4 level = get_volume()	21
3.8.5 play_video(path_name, repeat)	21
3.8.6 pause_video()	21
3.8.7 resume_video()	21
3.8.8 stop_video()	22
3.8.9 play_video(file, left, top, width, height)	22
3.8.10 on_video_notify(msg, v1, v2)	22
3.9 记录控件	22
3.9.1 record_set_event(screen, control, event_id)	22
3.9.2 record_reset_event(screen, control, event_id)	22
3.9.3 record_add(screen, control, record)	22
3.9.4 record_insert(screen, control, position, record)	22
3.9.5 record_clear(screen, control)	23
3.9.6 record_setoffset(screen, control, offset)	23

3.9.7	record_get_count(screen, control)	23
3.9.8	data = record_read(screen, control, position)	23
3.9.9	record_modify(screen, control, position, record)	23
3.9.10	record_delete(screen, control, position)	23
3.9.11	record_select(screen, control, position)	23
3.9.12	record_export(screen, control)	24
3.9.13	freeA, totalA, percentage = fs_getDiskFreeSize(drive)	24
3.9.14	assign_record_export(screen, control, folder_path, file_name)	24
3.9.15	readRecordFile(screen, control, file_path)	24
3.9.16	start_copy_file(file_type, from_path, to_path)	24
3.9.17	deleteAssignFile(file_type, folder_path, file_name)	24
3.10	数据记录表格控件	25
3.10.1	import_data_record (screen, control, file_type, source_path)	25
3.10.2	export_data_record (screen, control, file_type, dest_path)	25
3.10.3	add_data_record (screen, control, data_num, data_array)	25
3.10.4	delete_data_record (file_path)	25
3.10.5	clear_data_record (screen, control)	25
3.10.6	copy_data_record (source_path, dest_dir)	25
3.11	网络相关	26
3.11.1	wifi_mode, secumode, ssid, password = get_wifi_cfg()	26
3.11.2	set_wifi_cfg(wifi_mode, secumode, ssid, password)	26
3.11.3	state = get_network_state()	26
3.11.4	set_network_cfg(dhcp, ip_addr, netmask, gateway, dns)	26

3.11.5 dhcp, ip_addr, netmask, gateway, dns = get_network_cfg()	26
3.11.6 save_network_cfg()	27
3.11.7 set_network_service_cfg(wificom, mode, port, server_addr)	27
3.11.8 wificom, mode, port, server_addr = get_network_service_cfg()	27
3.11.9 ap_count = scan_ap()	27
3.11.10 ssid, security, quality = get_ap_info(index)	27
3.11.11 client_send_data(packet)	27
3.11.12 server_send_data(clinet_id, packet)	27
3.11.13 on_client_rcv_data(packet)	28
3.11.14 on_server_rcv_data(clinet_id, packet)	28
3.11.15 http_request(task_id, uri, method, content_type, post_data)	28
3.11.16 on_http_response(task_id, response)	28
3.11.17 http_download(task_id, uri, save_path)	28
3.11.18 on_http_download(task_id, status)	28
3.11.19 sockfd = udp_create(port)	29
3.11.20 udp_close(sockfd)	29
3.11.21 ret, ip, port, packet = udp_recvfrom(sockfd)	29
3.11.22 udp_sendto(sockfd, ip, port, packet)	29
3.11.23 postdata = get_wifi_mac()	29
3.12 FLASH 存储器读写	29
3.12.1 write_flash(addr, data)	29
3.12.2 data = read_flash(addr, length)	29
3.12.3 write_flash_string(addr, str)	30

3.12.4	str = read_flash_string(addr)	30
3.12.5	flush_flash()	30
3.13	文件系统操作	30
3.13.1	list_dir(path)	30
3.13.2	state = file_open(path, mode)	30
3.13.3	state = file_close()	31
3.13.4	size = file_size()	31
3.13.5	state = file_seek(offset)	31
3.13.6	data = file_read(count)	31
3.13.7	state = file_write(data)	31
3.14	摄像头	31
3.14.1	update_camera_status(screen, control, status)	31
3.14.2	size_table = get_camera_frame_descriptors()	32
3.15	数据库 SQL	32
3.15.1	database_creation(file_name, data1, data2...)	32
3.15.2	database_insertion(file_name, data1, data2...)	32
3.15.3	buf = database_sheet_query(file_name, data)	32
3.15.4	buf = database_all_query(file_name)	32
3.15.5	database_update(file_name, data1, data2)	33
3.15.6	database_delete(file_name, data)	33
3.16	配方	33
3.16.1	load_recipe()	33
3.16.2	save_recipe (group_id)	33

3.16.3	export_recipe (dir)	33
3.16.4	remove_recipe_group (group_id)	33
3.16.5	set_recipe_group_name (group_id, group_name)	33
3.16.6	group_name = get_recipe_group_name (group_id)	33
3.16.7	set_recipe_group_values (group_id, val_table)	34
3.16.8	val_table = get_recipe_group_values (group_id)	34
3.16.9	remove_recipe_item (group_id,item_id)	34
3.16.10	set_recipe_item_name (group_id,item_id,item_name)	34
3.16.11	item_name = get_recipe_item_name (group_id,item_id)	34
3.16.12	save_recipe_item (group_id,item_id,item_value)	34
3.16.13	set_recipe_item_value (group_id,item_id,item_val)	34
3.16.14	item_val = get_recipe_item_value (group_id,item_id)	34
3.16.15	copy_recipe_group_vals (source_group_id,dest_group_id)	34
3.17	其他.....	35
3.17.1	set_backlight(level)	35
3.17.2	level = get_backlight()	35
3.17.3	beep(time)	35
3.17.4	internal_lua_cmd(cmd)	35
3.17.5	set_systime(year,month,day,hour,minute,second)	35
3.17.6	year,month,day,hour,minute,second = get_systime()	35
3.17.7	set_language(lang)	35
3.17.8	lang = get_language()	36
3.17.9	standbymode(on)	36

3.17.10	<code>gpio_set_in()</code>	36
3.17.11	<code>gpio_set_out(pin)</code>	36
3.17.12	<code>gpio_set_value(pin, value)</code>	36
3.17.13	<code>value = gpio_get_value(pin)</code>	36
3.17.14	<code>start_copy_file(from, to)</code>	36
3.17.15	<code>str = get_version()</code>	36
3.17.16	<code>system_keyboard(screen, control, x, y, len, type)</code>	36
3.17.17	<code>set_text_input_scope(screen, control, type, value)</code>	37
3.17.18	<code>value = get_text_input_scope(screen, control, type)</code>	37
4.	声明与服务	37

1. 适用范围

文档仅适合串口屏产品

2. LUA 脚本介绍

LUA 脚本初学者可以通过下面链接进行学习。

<http://www.runoob.com/lua/lua-arrays.html>

3. API 接口函数

3.1 控件属性类

3.1.1 `change_screen(screen)`

切换到指定画面

screen: 目标画面 ID(如:第一个画面 id 为 0)

3.1.2 `set_value(screen, control, value)`

设置的控件数值:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

value: 要设置的控件数值

按钮控件: value--0 为按下, 1 为弹起

文本控件: value--整数或小数

也可以设置进度条、滑块、仪表等

3.1.3 `value = get_value(screen, control)`

获取控件数值, 按钮、文本、进度条、滑块、仪表等:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

value: 控件数值

按钮控件: value--0 为按下, 1 为弹起

文本控件: value--获取的数值(数值是整形的话获取到的是一个一位小数的浮点型)

其他控件: 进度条为当前的进度等等……

3.1.4 `set_visiable(screen, control, visiable)`

设置控件是否可见:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

visiable: 0 为隐藏, 1 为显示

3.1.5 set_enable(screen, control, enable)

设置控件是否可触摸:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

enable: 0 为禁止触摸, 1 为启用触摸

3.1.6 set_fore_color(screen, control, color)

设置控件前景色(例如文本控件文字颜色, 进度条显示颜色等):

color 为 RGB565(如:黑色为 0x000000 , 红色为 0xFF0000)

3.1.7 set_back_color(screen, control, color)

设置控件背景色(例如文本控件背景颜色, 进度条背景颜色):

color 为 RGB565(如:黑色为 0x000000 , 红色为 0xFF0000)

3.1.8 set_text(screen, control, text)

设置控件显示内容(字符串)(例如文本控件, 二维码控件等):

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

text 为要显示的字符串

3.1.9 string = get_text(screen, control)

获取控件字符串内容(字符串)(例如文本控件, 二维码控件等):

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

string 获取到的字符串

3.1.10 currentScreen = get_current_screen()

获取当前画面 ID:

currentScreen: 目标画面 ID(如:第一个画面 id 为 0)

3.1.11 set_text_flicker(screen, control, cycle)

设置文本控件闪烁周期, 单位秒:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

cycle: 为 0 停止闪烁

3.1.12 play_animation(screen, control)

设置动画控件播放:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

3.1.13 stop_animation(screen, control)

设置动画控件停止

3.1.14 histGraph_reset(screen, control)

清除曲线, 重新采集:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

3.1.15 histGraph_load(screen, control, file)

导入曲线数据(文件的数据格式需与被导入的曲线控件一致):

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

file: 文件路径

3.1.16 histGraph_export(screen, control, path)

导出文件数据:

(其他参数上同)

path: 可选参数(默认根目录)

3.2 常用回调函数

3.2.1 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。

3.2.2 on_systick()

系统每隔 1 秒钟自动调用此回调函数。

3.2.3 on_control_notify(screen, control, value)

用户触摸修改控件后, 执行此回调函数。

点击按钮控件, 修改文本控件、修改滑动条都会触发此事件。

value-为数值类型, 如果需要获取文本控件的字符串值, 使用 `get_text(screen, control)`。

3.2.4 on_screen_change(screen)

当画面需要切换时, 执行此回调函数, screen 切换到的画面(如:第一个画面 id 为 0)。

(注意, 此函数内部调用 `change_screen`, 不会嵌套执行 `on_screen_change`)

3.2.5 on_press(state, x, y)

用户点击触摸屏时, 执行此回调函数。

State: 0 松开, 1 按下, 2 持续按压

x, y: 为触摸坐标

3.2.6 on_usb_inserted(driver)

U 盘插入时, 执行此回调函数:

Dirver: 为 U 盘的盘符

3.2.7 on_usb_removed()

U 盘拔出时, 执行此回调函数

3.2.8 on_sd_inserted(dir)

SD 卡插入通知, dir 盘符路径

3.2.9 on_sd_removed()

SD 卡拔出通知

3.3 绘图函数

3.3.1 on_draw(screen)

重绘时执行此回调函数(通常所有绘图操作都在此函数中实现):

screen: 目标画面 ID(如:第一个画面 id 为 0)。

3.3.2 redraw()

发送重绘请求, 触发 on_draw 的执行。

3.3.3 set_pen_color(color)

设置画笔的颜色, RGB565(如:黑色为 0x000000, 红色为 0xFF0000), 用于指定线、矩形、圆等的颜色。

3.3.4 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(值的范围: 1~10)

3.3.5 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: 0 不填充, 1 填充

3.3.6 draw_circle(x, y, r, fill)

绘制圆形:

x, y: 圆的中心坐标

r: 圆的半径

fill: 0 不填充, 1 填充

3.3.7 surface = load_surface(file_name)

加载图片到图层(例如: surface = load_surface(“c:/test.jpg”)):

file_name: 图片文件(路径), 支持 JPEG/PNG

surface: 图层资源指针

(注: 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。)

3.3.8 destroy_surface(surface)

销毁图层:

surface: 图层资源指针

3.3.9 draw_surface(surface, dstx, dsty, width, height, srcx, srcy)

绘制图层(相比于 draw_image_file, 此方法效率较高):

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]

例如:

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

缩放显示: draw_surface(surface, dstx, dsty, width, height)

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

3.3.10 screen_shoot(file_name, x, y, width, height, quality)

截取屏幕窗口范围, 存储到指定文件路径

(例如: screen_shoot(“b:/shoot.jpg”, 0, 0, 480, 272, 95)):

file_name: 图片文件存放路径

quality: JPEG 图片质量, 默认 95

3.3.11 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

3.4 寄存器访问

LUA 中访问 MODBUS/PLC 等协议中定义的变量, 需要通过下面的变量访问接口

3.4.1 val = get_variant(slave, type, addr_name, port)

获取协议变量的数值:

slave: 从机 id

type: 变量类型(线圈:1, 离散输入:2, 保持寄存器:3, 输入寄存器:4, 本地变量:-1)

addr: 变量地址(十六进制), 也可以是变量的名称

port: 串口 id(可不写: 单端口时, 会默认为 modbus 串口, 多端口时, 如果两个串口都是 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为 modbus 串口, 本地变量为-1(串口中没有 modbus 协议默认为本地变量), 串口: 232 的 id 为 0, 485 为 1)。

val: 根据变量的类型返回不同类型的值

3.4.2 set_variant(slave, type, addr_name, val, port)

设置协议变量的数值:

slave: 从机 id

type: 变量类型(线圈:1, 离散输入:2, 保持寄存器:3, 输入寄存器:4, 本地变量:-1)

addr: 变量地址(十六进制), 也可以是变量的名称

val: 要设置的值(注: 变量类型不同, 要设置的值类型也要不同)

port: 串口 id(可不写: 单端口时, 会默认为 modbus 串口, 多端口时, 如果两个串口都是 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为 modbus 串口, 本地变量为-1(串口中没有 modbus 协议默认为本地变量), 串口 232 的 id 为 0, 485 为 1)。

3.4.3 mb_set_timeout(timeout)

设置从机应答超时时间:

Timeout: 超时时间(范围 1~255, 10 毫秒单位)

3.4.4 buf = mb_read_coil_01 (slave, addr, quantity, pattern, port)

01 功能码, 读取线圈:

slave: 从机 id

addr: 变量地址(十六进制)

quantity: 要读取的地址个数

pattern: 1 为异步模式(不会阻塞, 发送读取指令不会等待数据返回)

0 为同步模式(会阻塞, 发送后会等待数据返回, 可能影响 lua 的线程运行导致卡顿(主线程))

(注: 不写时默认为同步模式)

port: 串口 id(不写时: 单端口时, 默认为 modbus 串口, 多端口时, 如果

两个串口都是 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为 modbus 串口)

buf: 成功时返回字节数组, 一个字节 8 个 coil, 失败返回 nil(异步通信只返回一个序号标记, 用于在异步通信接受回调函数对应)

(数组下标从 0 开始)

3.4.5 buf = mb_read_input_02(slave, addr, quantity, pattern, port)

02 功能码, 读取离散输入:

(其他参数同上)

buf: 成功时返回字节数组, 一个字节 8 个 input, 失败返回 nil (数组下标从 0 开始)

3.4.6 buf = mb_read_reg_03(slave, addr, quantity, pattern, port)

03 功能码, 读取保持寄存器:

(其他参数同上)

buf: 成功时返回 WORD 数组, 失败返回 nil (数组下标从 0 开始)

3.4.7 buf = mb_read_input_reg_04(slave, addr, quantity, pattern, port)

04 功能码, 读取输入寄存器:

(其他参数同上)

buf: 成功时返回 WORD 数组, 失败返回 nil (数组下标从 0 开始)

3.4.8 value = mb_write_coil_05(slave, addr, status, pattern, port)

05 功能码, 写单个线圈:

slave: 从机 id

addr: 变量地址(十六进制)

status: 1 为 ON, 0 为 OFF

pattern: 1 为异步模式(不会阻塞, 发送读取指令不会等待数据返回)

0 为同步模式(会阻塞, 发送后会等待数据返回, 可能影响 lua 的线程运行导致卡顿(主线程))

(注: 不写时默认为同步模式)

port: 串口 id(不写时: 单端口时, 默认为 modbus 串口, 多端口时, 如果两个串口都是 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为 modbus 串口)

value: 成功时返回 true, 失败返回 false(异步通信只返回一个序号标记, 用于在异步通信接受回调函数对应)

3.4.9 value = mb_write_reg_06(slave, addr, reg, pattern, port)

06 功能码, 写入单个保持寄存器:

(其他参数同上)

Reg: 寄存器值

value: 成功时返回 true, 失败返回 false (异步通信只返回一个序号标记, 用于在异步通信接受回调函数对应)

3.4.10 value = mb_write_coil_15(slave, addr, quantity, coils, pattern, port)

15 功能码, 写多个线圈:

slave: 从机 id

addr: 变量地址(十六进制)

quantity: 要读取的地址个数

coils: 字节数组, 一个字节 8 个 coil(数组下标从 0 开始)

pattern: 1 为异步模式(不会阻塞, 发送读取指令不会等待数据返回)

0 为同步模式(会阻塞, 发送后会等待数据返回, 可能影响 lua 的线程运行导致卡顿(主线程))

(注: 不写时默认为同步模式)

port: 串口 id(不写时: 单端口时, 默认为 modbus 串口, 多端口时, 如果两个串口都是 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为 modbus 串口)

value: 成功时返回 true, 失败返回 false

3.4.11 value = mb_write_reg_16(slave, addr, regs, pattern, port)

16 功能码, 写入多个保持寄存器:

(其他参数同上)

Regs: 寄存器 WORD 数组(数组下标从 0 开始)

value: 成功时返回 true, 失败返回 false

3.5 定时器

3.5.1 start_timer(timer_id, timeout, countdown, repeat)

启动定时器:

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

3.5.2 stop_timer(timer_id)

停止定时器:

timer_id: 定时器 ID(0~31)

3.5.3 on_timer(timer_id)

定时器超时回调函数:

timer_id: 超时的定时器号

3.5.4 get_timer_value(timer_id)

获取定时器当前计时时间:

timer_id: 定时器 ID(0~31)

3.6 串口

3.6.1 uart_send_data(packet, port)

通过串口发送自定义格式数据:

packet: 字节数组(数组下标从 0 开始)
(如: packet = {0x5A, 0xA5, 0x00, 0x01, 0x0D, 0xFF})
port: 串口 id(不写时: 单端口时, 默认为非 modbus 串口, 多端口时, 如果两个串口都是非 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为非 modbus 串口, 如果没有非 modbus 协议, 则不会发送数据)

3.6.2 uart_set_timeout(timeout, timeout_inter, port)

设置串口接收超时时间:

Timeout: 接收总超时
timeout_inter: 字节间隔超时
port: 参数说明同上

3.6.3 uart_set_baudrate(baudrate, port)

设置波特率:

baudrate: 波特率(如: baudrate = 115200)

port: 参数说明同上

3.6.4 baudrate = uart_get_baudrate(port)

获取波特率:

port: 参数说明同上

baudrate: 返回值

3.6.5 on_uart_recv_data(packet, port)

自由协议串口接收数据的回调函数:

packet: 字节数组, 返回的数据, 需要用户解析。(数组下标从 0 开始)

port - 参数说明同上

3.6.6 uart_setup(baudrate, parity, stop_bit, data_bits, port)

串口参数设置:

baudrate: 波特率值

parity: 0-无校验 1-奇校验 2-偶校验

stop_bit: 0-1bit, 1-1.5bit

data_bits: 数据位 5~8

port: 串口 id(不写时: 单端口时, 默认为非 modbus 串口, 多端口时, 如果两个串口都是非 modbus 协议, 那么默认为 232 串口, 如果不同, 则默认为非 modbus 串口, 如果没有非 modbus 协议, 则不会发送数据)

3.7 CAN 接口

3.7.1 canbus_open(index, baudrate, listen_mode, loop_back)

打开 CANBUS 接口:

index: 索引号 (0~1)

baudrate: 波特率 (单位 K), 可选 125, 250, 500, 1000

listen_mode: 只读模式

loop_back: 自发自收 (环回/自测)

3.7.2 canbus_close(index)

关闭 CANBUS 接口:

index: 索引号 (0~1)

3.7.3 canbus_write(index, identifier, dlc, rtr, ide, data)

发送 CAN 报文:

index: 索引号 (0~1)
identifier: 报文 ID
dlc: 数据长度
rtr: 1 远程帧 0 数据帧
ide: 1 扩展帧 0 标准帧
data: 数据, table 格式(字节数组(数组下标从 0 开始))

3.7.4 on_canbus_recv(index, identifier, dlc, rtr, ide, data)

CAN 报文回调函数(收到报文后, 系统自动调用):

(参数说明同上)

3.8 音视频

3.8.1 play_sound(file_name)

播放指定音频:

file_name: 要播放的声音文件路径, 例如 play_sound('b:/sounds/welcome.wav')

3.8.2 on_audio_callback(state)

声音播放结束回调通知:

state: 保留未使用。

3.8.3 set_volume(level)

设置音量:

level: 音量大小 (0~100)

3.8.4 level = get_volume()

获取音量:

level: 获取到的音量大小 (0~100)

3.8.5 play_video(path_name, repeat)

播放视频(全屏播放):

path_name: 为视频路径

repeat: 为重复次数 (0 为无限制次)

(例如: 播放屏内视频: play_sound('b:/Videos/1.mp4' , 0))

3.8.6 pause_video()

暂停视频播放

3.8.7 resume_video()

恢复视频播放

3.8.8 stop_video()

停止视频播放

3.8.9 play_video(file, left, top, width, height)

播放视频(自定义位置大小):

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

3.8.10 on_video_notify(msg, v1, v2)

视频播放回调函数:

msg: 1-播放中, 0-播放完毕

v1: 当前播放进度, 当前已播时长, 单位 s

v2: 播放总进度, 当前视频总时长, 单位 s

3.9 记录控件

3.9.1 record_set_event(screen, control, event_id)

告警类型-触发告警:

screen: 目标画面 ID(如:第一个画面 id 为 0)

control: 目标控件 ID

event_id: 警告事件 id(0 开始)

3.9.2 record_reset_event(screen, control, event_id)

告警类型-解除告警:

(其他参数同上)

event_id: 要解除的警告事件 id(0 开始)

3.9.3 record_add(screen, control, record)

在末尾添加一条记录:

(其他参数同上)

record: 要添加的内容, 为字符串, 例如 "item1;item2;item3;"

3.9.4 record_insert(screen, control, position, record)

在指定位置插入一条记录:

(其他参数同上)

position: 要插入到的位置

record: 要插入的内容, 为字符串, 例如 “item1;item2;item3;”

3.9.5 record_clear(screen, control)

清除记录数据:

(其他参数同上)

3.9.6 record_setoffset(screen, control, offset)

设置滚动显示位置:

(其他参数同上)

offset: 要滚动到的位置

3.9.7 record_get_count(screen, control)

获取记录条数:

(其他参数同上)

3.9.8 data = record_read(screen, control, position)

读取一条记录:

(其他参数同上)

position: 要读取的行数

data: 返回的数据, 字符串, 例如 “item1;item2;item3;”

3.9.9 record_modify(screen, control, position, record)

修改一条记录:

(其他参数同上)

position: 要修改的位置

record: 要修改的内容, 字符串, 例如 “item1;item2;item3;”

3.9.10 record_delete(screen, control, position)

删除一条记录:

(其他参数同上)

position: 要删除的位置

3.9.11 record_select(screen, control, position)

选中一条记录:

(其他参数同上)

position: 要选中的位置

3.9.12 record_export(screen, control)

导出记录到 SD 卡/U 盘:

(其他参数同上)

3.9.13 freeA, totalA, percentage = fs_getDiskFreeSize(drive)

查看 flash 里面分区的剩余空间(AB 两大分区, A 区占只要空间):

drive: 分盘, 字符, 如: freeA, totalA, percentage = fs_getDiskFreeSize('A')

freeA 为剩余的空间, 单位字节

totalA 为总空间, 单位字节

percentage 为剩余空间占总空间的百分比(字符串)

3.9.14 assign_record_export(screen, control, folder_path, file_name)

保存数据记录控件的数据(指定路径和文件名):

screen: 目标画面 ID(如: 第一个画面 id 为 0)

control: 目标控件 ID

folder_path: 要保存到的路径

file_name: 保存数据的文件名字

3.9.15 readRecordFile(screen, control, file_path)

打开一个 CSV 文件, 并把数据放到一个表格控件中:

(其他参数同上)

file_path: 要打开 csv 文件的路径

3.9.16 start_copy_file(file_type, from_path, to_path)

拷贝文件:

file_type: 文件类型, 4 为文件夹, 8 为文件(目前只支持这两种)

from_path: 要拷贝的文件路径

to_path: 拷贝好的文件路径以及文件名字

3.9.17 deleteAssignFile(file_type, folder_path, file_name)

删除文件夹或者文件:

deleteAssignFile(8, folderpath, filename)—删除文件,

deleteAssignFile(4, folderpath)—删除文件夹。

file_type: 文件类型, 4 为文件夹, 8 为文件(目前只支持这两种)

folder_path: 要删除的文件夹路径

file_name: 要删除的文件

3.10 数据记录表格控件

3.10.1 import_data_record (screen, control, file_type, source_path)

导入数据记录到数据记录表格控件:

screen: 控件所在页面 ID

control: 控件 ID

file_type: 文件类型, 0 为 Bin 文件; 1 为 CSV 文件

source_path: 文件源路径

3.10.2 export_data_record (screen, control, file_type, dest_path)

将表格数据导出到文件:

screen: 控件所在页面 ID

control: 控件 ID

file_type: 文件类型, 0 为 Bin 文件; 1 为 CSV 文件

dest_path: 导出文件的路径

3.10.3 add_data_record (screen, control, data_num, data_array)

添加数据记录到数据记录表格控件:

screen: 控件所在页面 ID

control: 控件 ID

data_num: 数据数组大小

data_array: 数据数组

3.10.4 delete_data_record (file_path)

删除数据记录:

file_path: 文件夹路径或单个文件路径, 即删除整个文件夹下的记录文件或者删除单个文件

3.10.5 clear_data_record (screen, control)

清空表格的记录数据:

screen: 控件所在页面 ID

control: 控件 ID

3.10.6 copy_data_record (source_path, dest_dir)

复制文件:

source_path: 文件源文件夹目录或文件路径

dest_dir: 复制到的文件夹目录

3.11 网络相关

3.11.1 `wifi_mode, secumode, ssid, password = get_wifi_cfg()`

获取 wifi 的参数(返回 4 个参数):

(如: `wifi_mode, secumode, ssid, password = get_wifi_cfg()`)

wifi_mode: 无线网络模式 0-禁用无线网络, 1-无线网卡模式, 2-AP 热点模式

secumode: 加密模式 0-AUTO(默认值) 1-WEP 2-WPAPSK 3-WPAPSK2

ssid: 无线网络名称

password: 无线网络密码

3.11.2 `set_wifi_cfg(wifi_mode, secumode, ssid, password)`

设置 wifi 的参数:

(参数同上)

3.11.3 `state = get_network_state()`

获取网络的状态:

(如: `state = get_network_state()`)

state: 网络的状态

状态位说明:

bit0: 无线网络连接

bit1: 有线网络连接

bit2: 是否连上服务器

bit3: 是否有客户端连上

3.11.4 `set_network_cfg(dhcp, ip_addr, netmask, gateway, dns)`

设置以太网的参数:

dhcp: 启用 DHCP, 0 禁用 1 启用, 禁用时后面的参数才有效

ip_addr: 静态 IP

netmask: 掩码

gateway: 子网掩码

dns: 域名服务器

3.11.5 `dhcp, ip_addr, netmask, gateway, dns = get_network_cfg()`

获取以太网的状态:

(参数同上)

3.11.6 save_network_cfg()

保存网络设置，并重连网络

(注意：修改网络配置、服务器参数等需要调用此函数进行保存生效。)

3.11.7 set_network_service_cfg(wificom, mode, port, server_addr)

设置网络服务参数：

wificom: 默认为 0，为 1 时启用透传模式（即无线串口屏）

mode: 0 禁用网络服务，1 客户端模式，2 服务器模式

port: 服务端口，默认 5050

server_addr: 服务器地址，（屏作客户端时）

3.11.8 wificom, mode, port, server_addr = get_network_service_cfg()

获取网络服务参数：

(参数同上)

3.11.9 ap_count = scan_ap()

扫描无线热点：

ap_count: 热点数目

3.11.10 ssid, security, quality = get_ap_info(index)

获取指定热点的信息：

Index : 热点索引，索引从 0 开始

ssid: 热点名称

security: 加密方式

quality: 信号质量

3.11.11 client_send_data(packet)

通过客户端 SOCKET 发送报文：

packet: local packet = {} - 定义数组

packet[0] = 0x01

packet[1] = 0x02

...

client_send_data(packet): 发送字节数组 packet，下标从 0 开始

3.11.12 server_send_data(clinet_id, packet)

通过服务端 SOCKET 发送报文：

clinet_id: 目标客户端 ID

packet: 发送字节数组 packet (下标从 0 开始)

3.11.13 on_client_recv_data(packet)

当客户端 SOCKET 接收到数据时, 系统自动回调此函数:

packet: 接收的字节数组 (下标从 0 开始)

3.11.14 on_server_recv_data(clinet_id, packet)

当服务端 SOCKET 接收到数据时, 系统自动回调此函数:

clinet_id: 客户端 ID

packet: 接收的字节数组 (下标从 0 开始)

(处理方法与 on_client_recv_data 类似)

3.11.15 http_request(task_id, uri, method, content_type, post_data)

发送 HTTP 请求到服务器:

task_id: 请求任务编号 (可任意设置)

uri: 资源路径

method: 方法, 0 为 GET, 1 为 POST

以下参数 POST 方法才需要:

content_type: 数据类型, 字符串, 例如: json, xml, text 等

post_data: POST 数据, 字符串地址

3.11.16 on_http_response(task_id, response)

HTTP 响应, 系统自动回调此函数:

task_id: 响应任务编号, 与 http_request 匹配

response: 响应数据

3.11.17 http_download(task_id, uri, save_path)

使用 HTTP 协议下载文件:

task_id: 请求任务编号 (可任意设置)

uri: 资源路径

save_path: 要存放路径 (文件名字可自定义)

3.11.18 on_http_download(task_id, status)

下载响应, 系统自动回调此函数:

task_id: 响应任务编号, 与 http_download 匹配

status: 下载状态: 0 为下载失败, 1 为下载成功但存储失败, 2 为下载并存储成功

3.11.19 sockfd = udp_create(port)

创建 UDP 套接字，并绑定服务端口：

port: 要创建的 UDP 套接字(例如: sockfd = udp_create(12345))

sockfd: 返回的服务端口

3.11.20 udp_close(sockfd)

关闭 UDP 套接字：

sockfd: 服务端口

3.11.21 ret, ip, port, packet = udp_recvfrom(sockfd)

接收 UDP 数据报文：

sockfd: 服务端口

ret: -1 表示发生错误， 0 表示无数据，其他值表示数据长度

ip: 发送端的 IP

port: 发送端的端口

packet: 为数据报文，table 类型

3.11.22 upd_sendto(sockfd, ip, port, packet)

发送 UDP 数据报文：

sockfd: UDP 套接字

ip: 发送端的 IP

port: 发送端的端口

packet: 为数据报文，table 类型(下标从 0 开始)

3.11.23 postdata = get_wifi_mac()

获取 mac 地址：

Postdata: 返回的字符串

3.12 FLASH 存储器读写

屏幕提供 128K 用户 FLASH，可用于存储配置参数。

3.12.1 write_flash(addr, data)

写用户 FLASH 数据：

addr: 写入地址

data: 字节数组(下标从 0 开始)

3.12.2 data = read_flash(addr, length)

读用户 FLASH 数据：

addr: 写入地址
length: 读取字节数
data: 返回类型为字节数组(下标从 0 开始)

3.12.3 write_flash_string(addr, str)

写字符串到指定 FLASH 地址:

addr: 写入地址
str: 写入的字符串

3.12.4 str = read_flash_string(addr)

从指定 FLASH 地址读取字符串:

addr: 写入地址
str: 成功返回字符串, 失败返回 nil

3.12.5 flush_flash()

系统会对 FLASH 写入操作进行缓存优化, 以提高写入效率:

(flush_flash 操作会立即把数据写入 FLASH。)

3.13 文件系统操作

文件系统读写接口, 支持对屏内、SD 卡、U 盘读写。

3.13.1 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。

(

通过以下回调函数返回文件夹的内容:

on_list_dir(path, filename, type, fsize):

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

)

3.13.2 state = file_open(path, mode)

打开文件:

path: 文件路径
mode: 打开模式, 如下组合方式:

FA_OPEN_EXISTING 0x00

FA_READ 0x01
FA_WRITE 0x02
FA_CREATE_NEW 0x04
FA_CREATE_ALWAYS 0x08
FA_OPEN_ALWAYS 0x10

例如:

打开文件用于读取: `file_open(path, 0x01)`

创建文件用于写入: `file_open(path, 0x02|0x08)`

state: 成功返回 true, 失败 false

3.13.3 state = file_close()

关闭文件:

state: 成功返回 true, 失败 false

3.13.4 size = file_size()

size: 获取当前文件大小, 返回字节数

3.13.5 state = file_seek(offset)

定位文件读取位置:

offset: 文件偏移位置

state: 成功返回 true, 失败 false

3.13.6 data = file_read(count)

读取文件内容:

count: 读取字节数, 最大读取 2048 个字节

data: 成功返回 table 数组, 失败返回 nil

3.13.7 state = file_write(data)

写文件内容:

data: 待写入的 table 数组(索引从 0 开始, 最大一次性写 2048 个字节)

state: 成功返回 true, 失败返回 false

3.14 摄像头

3.14.1 update_camera_status(screen, control, status)

设置摄像头状态:

screen: 目标画面 ID

control: 目标控件 ID

status: 0 为停止; 1 为播放; 2 为暂停

3.14.2 size_table = get_camera_frame_descriptors()

获取摄像头支持的分辨率等信息

3.15. 数据库 SQL

3.15.1 database_creation(file_name, data1, data2...)

创建或者打开数据库并创建表:

file_name: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

data1, data2...: 创建表的成员, 如: 'Name text', 'Gebder text',

'BirthdayMonth int' Name 为成员的 名字, text 为类型(如:字符串、整形、字符)

3.15.2 database_insertion(file_name, data1, data2...)

写入数据:

filename: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

data1, data2...: 要写入的数据, 如: "" 成龙'", "" 男'", 11, 25, 1973, 175, 75

(注: 数据要按创建表格的成员的顺序, 字符串要加 ' ', 如: "" 成龙'")

3.15.3 buf = database_sheet_query(file_name, data)

查找单行数据:

file_name: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

data: 要查找的数据, 如: "Name = ' 成龙'" 或者 "BirthdayMonth = 11"

(注: 字符串要加 ' ', 如: "" 成龙'"),

buf: 返回查到的数据, 字符串, 如: "1;成龙;男;11;25;1973;175;75/"

3.15.4 buf = database_all_query(file_name)

查找全部数据:

file_name: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

buf: 返回查到的数据, 一个字符串, 如: "1;成龙;男;11;25;1973;175;75/2;李连杰;

男;11;25;1973;175;75/"

3.15.5 database_update(file_name, data1, data2)

更新数据:

file_name: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

data1: 要更新行的成员, 类型为字符串, 如: "Name = '程心'"

(注: 字符串要加 ' ', 如: "'成龙'")

data2: 要更新这行成员的数据, 如: "BirthdayMonth = 120", 类型为字符串

3.15.6 database_delete(file_name, data)

删除数据:

file_name: 数据库的名字和路径, 如: 'B:\\FangHuaUserData.db'

(在 A 盘创建打开名字为 FangHuaUserData.db 的数据库)

data: 要更新行的成员, 类型为字符串, 如: "Name = '程心'"

(注: 字符串要加 ' ', 如: "成龙")

3.16. 配方

3.16.1 load_recipe()

加载配方

3.16.2 save_recipe (group_id)

保存配方数据, 参数可选:

group_id: 配方组 ID; 若不输入配方组 ID, 则默认保存所有配方数据

3.16.3 export_recipe (dir)

导出配方数据:

dir: 目的文件夹目录

3.16.4 remove_recipe_group (group_id)

删除配方组:

group_id: 配方组 ID

3.16.5 set_recipe_group_name (group_id, group_name)

设置配方组名称:

group_id: 配方组 ID

group_name: 配方组名称

3.16.6 group_name = get_recipe_group_name (group_id)

获取配方组名称: (参数同上)

3.16.7 set_recipe_group_values (group_id, val_table)

设置配方组数据:

group_id: 配方组 ID

val_table: 数据表 (或者数组)

3.16.8 val_table = get_recipe_group_values (group_id)

获取配方组数据: (参数同上)

3.16.9 remove_recipe_item (group_id, item_id)

获取配方组数据:

group_id: 配方组 ID

item_id: 配方项 ID

3.16.10 set_recipe_item_name (group_id, item_id, item_name)

设置配方项名称:

group_id: 配方组 ID

item_id: 配方项 ID

item_name: 配方项名称

3.16.11 item_name = get_recipe_item_name (group_id, item_id)

获取配方项名称: (参数同上)

3.16.12 save_recipe_item (group_id, item_id, item_value)

保存配方项数据:

group_id: 配方组 ID

item_id: 配方项 ID

item_value: 可选参数, 设置配方项数据后再保存;

3.16.13 set_recipe_item_value (group_id, item_id, item_val)

设置配方项数据:

group_id: 配方组 ID

item_id: 配方项 ID

item_val: 配方项数据

3.16.14 item_val = get_recipe_item_value (group_id, item_id)

获取配方项数据: (参数同上)

3.16.15 copy_recipe_group_vals (source_group_id, dest_group_id)

将配方组数据拷贝到目的配方组:

source_group_id: 源配方组 ID

dest_group_id: 目的配方组 ID

item_val: 配方项数据

3.17 其他

3.17.1 set_backlight(level)

设置背光亮度:

level: 背光亮度(0~100)

3.17.2 level = get_backlight()

获取背光亮度:

level: 返回的背光亮度

3.17.3 beep(time)

蜂鸣器叫:

time: 蜂鸣器叫的时长(单位毫秒)

3.17.4 internal_lua_cmd(cmd)

对内指令:

cmd: 要发送的内部指令 table 数组(索引从 0 开始, 帧头: EE,

帧尾: FF FC FF FF)

例如: EE XX XX FF FC FF FF

3.17.5 set_systime(year, month, day, hour, minute, second)

设置系统时间

3.17.6 year, month, day, hour, minute, second = get_systime()

设置系统时间:

返回: 年月日时分秒

例如, 获取时间数组:

```
local time_vals = { 2020, 11, 20, 10, 0, 0 }
```

```
time_vals[1], time_vals[2], time_vals[3], time_vals[4], time_vals[5],
```

```
time_vals[6] = get_systime()
```

3.17.7 set_language(lang)

设置当前语言选项:

lang: 要选择的语言的标志(从 0 开始)

3.17.8 lang = get_language()

获取当前语言选项:

lang: 当前语言标志

3.17.9 standbymode(on)

进入待机低功耗模式, 屏幕的串口可正常运行:

on: 0 为开启, 1 为关闭

3.17.10 gpio_set_in()

PIN 引脚设置为输入模式:

(初始化 13、14、15、16 引脚)

3.17.11 gpio_set_out(pin)

PIN 引脚设置为输出模式:

(初始化 17、18、26 引脚)

3.17.12 gpio_set_value(pin, value)

设置输出 PIN 引脚的电平:

pin: 引脚

value: 电平 (高电平 1/低电平 0)

3.17.13 value = gpio_get_value(pin)

获取输入 PIN 引脚电平:

pin: 引脚

value: 返回的电平 (高电平 1/低电平 0)

3.17.14 start_copy_file(from, to)

文件拷贝:

from: 源路径

to: 目标路径

3.17.15 str = get_version()

获取固件版本号:

str: 返回字符串

3.17.16 system_keyboard(screen, control, x, y, len, type)

打开系统键盘:

screen: 目标画面 ID(如: 第一个画面 id 为 0)

control: 目标控件 ID

x, y: 坐标

len: 最大文本长度

type: 输入类型: 0:正常字符;1:密码;2:时间;3:关闭键盘

3.17.17 `set_text_input_scope(screen, control, type, value)`

改变设置文本的最大最小值以及小数位数:

(其他参数同上)

type: 最大数值:1;最小数值:2;小数位数:3

value: 数值

3.17.18 `value = get_text_input_scope(screen, control, type)`

获取设置文本的最大最小值以及小数位数:

(其他参数同上)

type: 最大数值:1;最小数值:2;小数位数:3

value: 返回的数值

4. 声明与服务

感谢您选用欣瑞达系列产品, 若您对文档有什么异议或疑问, 欢迎随时与我们取得联系。电话: 0755-26018666, 网址: www.xrd-lcd.com。当然若文档有什么错误或误解之处, 欢迎给我们提出批评和建议, 我们将及时纠正和改进。